Chapter 8

Active Directory Domain Services Security

In this chapter:

One of the primary reasons for deploying a directory service like Active Directory Domain Services (AD DS) is to provide security on the corporate network. Every company stores business-critical information on file servers on the network. E-mail has become one of the primary means for exchanging business information. Intranet or Internet sites may contain confidential information, and access to the sites may need to be restricted to specific users. Managing secure access to these types of information is critical to ensure that only properly authorized users have access to the data. Microsoft Windows Server 2008 AD DS provides the directory service that enables security in these and many other scenarios.

This chapter begins by introducing the basics of AD DS security. AD DS uses several basic building blocks and concepts to provide security on a Windows Server 2008 network. After an introduction to the security basics, this chapter will focus on the authentication and authorization functions used by AD DS to ensure that users are who they say they are (authentication) and to provide access to the resources to which the user should have access (authorization). Windows Server 2008, like Microsoft Windows 2000 and Windows Server 2003, uses Kerberos as the primary authentication protocol, so much of the first part of this chapter will focus on understanding the role of Kerberos in authentication.

After discussing authentication and authorization, this chapter moves on to consider AD DS domain controller security and developing secure administrative practices. This is an essential second component in creating a secure AD DS environment.

More Info This chapter provides a basis for understanding and implementing security in a Windows Server 2008 network. Later chapters, such as Chapter 9, "Delegating the Administration of Active Directory Domain Services," and Chapter 13, "Using Group Policy to Manage Security," expand on the concepts discussed in this chapter.

AD DS Security Basics

There are some basic concepts needed to understand how AD DS security works on a Windows Server 2008 network. Essentially, AD DS security consists of two types of objects and the interaction between the two objects. The first object is a *security principal,* or an object that represents a user, group, or computer that needs access to some resource on the network. The second object is the resource itself, which is the object to which the security principal needs access. To provide the proper level of security, AD DS must have some way of determining the identity of the security principal and then giving the right level of access to the resources.

Security Principals

Security principals are the only objects in AD DS that can be granted permission to access resources on the network. Every security principal is assigned a security identifier (SID) when the object is created. The SID is made up of two parts. The first part is a domain identifier, and all security principals in a domain have the same domain identifier. The second part of the SID is the relative identifier (RID), which is unique for each security principal in an AD DS domain.

The SID is an essential component when configuring security for resources on a Windows Server 2008 network. When you grant permission to a resource, you use the security principal's display name, but Windows Server 2008 actually uses the SID to manage access to the resource. When a user tries to access a resource on a server in the domain, the operating system grants permission to the user's SID, rather than the person's name. This means that if a user's display name is changed, the permissions granted to the user do not change. However, if a user object is deleted and then re-created with the same name, the user will not be able to access the same resources, because the SID will be different.

Direct from the Field: Security IDs

A security identifier, or SID, is a numerical representation that uniquely identifies a security principal. SIDs are made up of three components: Revision Level, Identifier Authority, and Subauthority or Relative Identifier (RID).

SIDS use the following syntax:

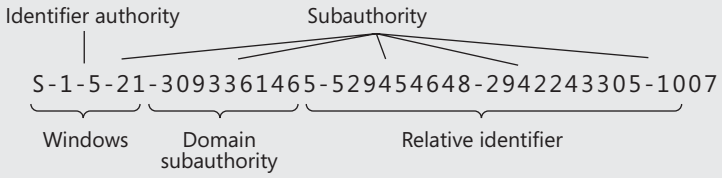
```
S-R-I-S-S
```

The letter *S* indicates that the information following is a SID. The letter *R* indicates the Revision Level of the security identifier. The letter *I* represents the Identifier Authority, and the next *S* is for the Subauthority or RID. There can be more than one Subauthority/RID value:

- **Revision Level** The Revision Level represents the revision Level of the SID Structure. The current Revision Level is 1.
- **Identifier Authority** The Identifier Authority is a 48-bit value that identifies the authority that issued the SID.
- **Subauthority/Relative Identifier** Subauthority/RIDs are used to uniquely identify the security principal relative to the authority that issued the SID. Subauthority/RIDS are used to ensure that no two SIDS are identical. This is accomplished by not allowing any SID-issuing authority to assign a RID more than once.

Windows creates security identifiers using the identifier authority of 5 and a subauthority of 21. Therefore, SIDs created on Windows start with S-1-5-21. The next subauthority is derived from either the domain or the local computer. This depends on whether the new security principal is a domain or local security principal. The remaining three subauthorities derive from the relative identifier. Each domain controller is allocated a pool of relative identifies from the RID manager FSMO role. The domain controller that creates the new security principal assigns a relative identifier, from its RID pool, to the new security principal. This creates the full security identifier:

S-1-5-21-3093361465-529454648-2942243305-1007



Mike Stephens

Microsoft Support Engineer

Access Control Lists

The other component that is included in AD DS security is the object that a security principal needs to access. This object may be an AD DS organizational unit (OU), printer object, or even a security principal. The object may also be a resource such as a file on a server running Windows Server 2008 or a mailbox on a server running Microsoft Exchange Server 2007.

The permissions that have been granted to these objects are located in an access control list (ACL). Every object in AD DS or on an NTFS file system partition has a security descriptor. The security descriptor includes the SID of the security principal that owns the object as well as the SID for the object's primary group. In addition, every object has two separate ACLs: a discretionary access control list (DACL) and a system access control list (SACL). The DACL lists the security principals or trustees that have been assigned permission to the object, as well as the level of permissions that have been assigned to each security principal. The DACL is made up of a series of access control entries (ACEs). Each ACE lists one SID and then identifies the level of access that the SID has to the object. The ACE can include an entry for all types of security principals. For example, a user account might have Read permissions to a file and a security group might have Full Control. The DACL for the file will have (at least) two ACEs, one granting the user Read permission and another granting the group Full Control.

The SACL lists the security principals whose access to the resource needs to be audited. The list of ACEs in the SACL indicates whose access is to be audited and the level of auditing required.

Note The DACL can contain ACEs that grant access to a resource as well as ACEs that deny access. The ACEs that deny access should be listed first in the ACL, so they are evaluated first by the security subsystem. If an ACE denies access to the resource, the security subsystem does not evaluate any other ACEs. This means that an ACE that denies permission to a resource always overrides any ACE that grants access to a specific SID. For more information on how security descriptors are used to grant access to AD DS objects, see Chapter 9.

Direct from the Field: Security Descriptors

Windows uses security descriptors to protect and audit resources. A security descriptor is comprised of an owner, a primary group, a discretionary access control list, and a system access control list.

Owner and Primary Group

The owner and primary group fields are security identifiers. The owner is the security principal that owns the object. The resource owner has full permissions to the object, including the ability to add or remove permissions within the security descriptor.

The primary group remains in the security descriptor for compatibility with the POSIX subsystem. Windows does not rely on this part of the security descriptor unless you are using utilities that require POSIX interoperability. By default, the security principal that created the object will write its default primary group to the security descriptor. Windows' default primary group is Domain Users.

The primary group is an implied group membership. When a user logs on, the operating system inserts the SID for this group in the user's token. The *memberOf* attribute

does not have the primary group listed. The *memberOf* attribute includes only group memberships that are explicitly assigned.

Discretionary and System Access Control Lists

Access control lists (ACLs) include two parts. The first part of the access control list is named control flags. These settings control how Windows applies permissions within the access control list and the rules of inheritance. The second part of the access control list is the list itself. The access control list contains one or more access control entries (ACEs).

Access control flags determine how Windows applies the access control entries contained within the access control list. Windows primarily uses the protected and automatic flags. The protected flag prevents the access control list from being modified by any inherited access control lists. This flag is equivalent to clearing the "Allow inheritable permissions from parent to propagate to this object" check box. The automatic flag is equivalent to selecting the "Allow inheritable permissions from parent to propagate to this object" option. This flag automatically allows access control entries in the access control lists to propagate to child objects.

Access Control Entries

Access control lists include one or more access control entries. Windows categorizes access control entries into two types: Allow and Deny. Each ACE type has a subtype object and nonobject subtypes. Allow and Deny access control entries designate the level of access the authorization subsystem provides based on the requested right by the security principal. Object access control entries are exclusive for objects in AD DS because they provide additional fields for object inheritance. Windows uses nonobject access control entries for most of the remaining resources such as file system and registry resources. Nonobject ACEs provide container inheritance—where an object residing in a container inherits the access control entry of the container. This is similar to files inheriting permissions from their parent folders. Each access control entry type has a rights field and a trustee field. The rights fields are usually comprised of predefined numbers that represent a specific action that a security principal can request. An example of a right would be a user requesting to read or write to a file. In this example, Read and Write are two separate rights. The trustee field is a security identifier that is allowed or denied the specified right. An example of a trustee would be user or group that is allowed or denied the action specified in the right field.

Mike Stephens

Microsoft Support Engineer

Access Tokens

The connecting point between the security principal's SID and the ACL is the *access token*. When Windows authenticates the user by using Kerberos, the user is assigned an access token on the local computer during the logon process. This token includes the user's primary SID, the SIDs for any groups to which the user belongs, and the user's privileges and rights.

Note The access token may also include additional SIDs in the SIDHistory attribute. These SIDs can be populated when you move user accounts from one domain to another. For a detailed discussion on SIDHistory, see Chapter 7, "Migrating to Active Directory Domain Services."

The access token is used by the security subsystem whenever a user tries to access a resource. When the user tries to access a local resource, the token is presented by the client workstation to any thread or application that requests security information before allowing access to a resource. The access token is never transmitted across the network to another computer; rather, a local access token is created on each server where the user tries to access a resource. For example, when a user tries to access a mailbox on a server running Exchange Server 2007, an access token is created on the server. In this case, the security subsystem on the server running Exchange Server 2007 will compare the SIDs in the access token to the permissions granted in the mailbox ACL. If the permissions granted to the SID allow it, the user will be able to open the mailbox.

Authentication

In order for the security processes, including their use of SIDs and ACLs, to work, there must be some way for a user to gain access to the network. Essentially, users must be able to prove that they are who they say they are so that they can retrieve their access token from the domain controller. This process is called *authentication*.

Authentication occurs during the initial client logon process on a computer that is a member of an AD DS domain. The exact steps vary depending on the operating system that the client is logging on to. When the user sits down at a Windows 2000, Microsoft Windows XP Professional, or Windows Server 2003 computer and enters Ctrl+Alt+Del, which is known as a secure attention sequence (SAS), the Winlogon service on the local computer switches to the logon screen and loads the Graphic Identification and Authentication (GINA) dynamic-link library (DLL). By default, this is the Msgina.dll. However, third parties can build alternative GINAs (for example, the NetWare client uses the Nwgina.dll). After the user has typed in the user name and password and has selected a domain, GINA passes the entered credentials back to the Winlogon process. The Winlogon service passes the information to the Local Security Authority (LSA).

Windows Vista and Windows Server 2008 do not use the GINA dll. Windows Vista and Windows Server 2008 introduce a new authentication model in which LogonUI and Winlogon communicate directly with each other. This means that when the users provide their credentials in these operating systems, the credentials are passed directly by the LogonUI component to the Winlogon service, which passes the information to the LSA. In order to authenticate to other directories, a third party can create a credential provider, which is a module that plugs into the LogonUI, to describe the UI and to gather the credential and pass it on to WinLogon. Credential providers are completely transparent to WinLogon.

In either case, the LSA immediately applies a one-way hash to the user's password and deletes the clear text password that the user typed in. The LSA then calls the appropriate Security Support Provider (SSP) through the Security Support Provider Interface (SSPI). Windows Server 2008 provides two primary SSPs for network authentication, the Kerberos SSP and the NT LAN Manager (NTLM) SSP. If Windows 2000 (or later) clients are logging on to a Windows Server 2008 network, the Kerberos SSP is selected and the information is passed to the SSP. The SSP then communicates with the domain controller to authenticate the user. The Kerberos authentication process will be covered in detail later in this chapter.

If authentication succeeds, the user is authenticated and granted access to the network. If the user has logged on to a domain, and if all the resources that the user needs to access are in the same forest, the user will be asked for authentication credentials only once. Until the user logs off, all the permissions the user gets on the network are based on the initial authentication. Although the user account is authenticated again each time the user accesses resources on a server to which the user has not authenticated, this authentication is transparent to the user.

Authorization

Authorization is the second step in the process of gaining access to network resources, and it takes place after authentication. During authentication, you are proving your identity by typing in the correct user name and password. During authorization, you are given access to resources on the network. Another way to think about this is to say that during authentication, the access token is created for you. During authorization, an access token is presented to a server and requests access to a resource. If the SIDs in your access token matches the SIDs in the DACL, then you are allowed or denied access, based on the access control entry on the resource.

Authorization, also known as *access control*, is the process of determining the level of access that is allowed or denied to an Active Directory object or file system object. After the Key Distribution Center (KDC) confirms the identity of the user, the security system on the authenticating domain controller generates authorization data in the form of the user's primary SID, plus SIDs for groups to which the user belongs that are recognized by all the resources on the Windows network. The authorization data is used by the computer that manages the network resource to generate an access token. The access token is used to determine the level of access that the user has to the network resource.

The access token contains the following:

- The list of SIDs that represent the user (including SIDs stored in SIDHistory)
- All groups (including nested groups) of which the user is a member
- The user's privileges (also called *user rights*) on the local computer

All objects or resources that are secured have a discretionary access control list (DACL) assigned to them that specifies the access rights of users and groups on that resource. Access to one of these objects or resources is controlled by an access check, in which the security system determines whether the requested access should be granted or denied by evaluating the contents of the access token of the requestor against the DACL on the resource.

Kerberos Security

So far this chapter has covered the basics of AD DS security without discussing the actual mechanism that implements the security. The primary mechanism for delivering authentication in AD DS is the Kerberos protocol. This protocol was first developed by engineers at the Massachusetts Institute of Technology (MIT) in the late 1980s. The current version of Kerberos is version 5 (Kerberos 5), which is described in RFC 1510, "The Kerberos Network Authentication Service (V5)." The Windows Server 2008 implementation of Kerberos is fully RFC-1510 compliant, with some extensions for public key authentication.

Kerberos is the default authentication protocol for Windows 2000 and Windows Server 2003 Active Directory, and for Windows Server 2008 AD DS. Whenever a Windows 2000 or later client authenticates to Active Directory or AD DS, the client will always try to use Kerberos. The other protocol that can be used to authenticate to AD DS is NTLM, which is supported primarily for backward compatibility for older clients. Kerberos has a number of advantages over NTLM:

- **Mutual authentication** With NTLM, authentication is only one-way; that is, the server authenticates the client. With Kerberos, the client can also authenticate the server, ensuring that the server that is responding to the client request is the correct server.
- **More efficient access to resources** When a user tries to access a network resource on an NTLM-based network (such as Microsoft Windows NT 4), the server where the resource is located has to contact a domain controller to check the user's access permissions. On a Kerberos-based network, the client connects to the domain controller and acquires a service ticket to connect to the resource server. This means that the resource server does not need to connect to the domain controller.
- **Improved trust management** NTLM trusts are always one-way, nontransitive, and manually configured. Kerberos trusts are automatically configured and maintained between all the domains in a forest and are transitive and two-way. In addition, Kerberos trusts can be configured between forests and between Windows Server 2008 Kerberos domains and other Kerberos implementations.

- **Delegated authentication** When a client connects to a server using NTLM authentication, the server can use the client credentials to access resources only on the local server. With Kerberos authentication, the server can use the client credentials to access resources on another server.

> **Note** Windows Server 2008 also supports authentication through Secure Sockets Layer/Transport Layer Security (SSL/TLS), Digest authentication, and Passport authentication. Since these authentication services are primarily used in an Internet environment for authentication to Microsoft Internet Information Services (IIS) 7.0, these authentication options will not be discussed.

Introduction to Kerberos

There are three components in a Kerberos-based system. The first is the client who needs to gain access to network resources. The second is the server that manages the network resources and ensures that only properly authenticated and authorized users can gain access to the resource. The third component is a Key Distribution Center (KDC), which serves as a central location to store user information and as a central service to authenticate users.

The Kerberos protocol defines how these three components interact. This interaction is based on two key principles. First of all, Kerberos operates on the assumption that authentication traffic between a workstation and server crosses an insecure network. This means that no confidential authentication traffic is ever sent across the network in clear text. A practical example of this is that the user password is never sent across the network, not even in an encrypted form. The second principle is that Kerberos operates based on a shared secret authentication model. In a shared secret authentication model, the client and the authenticating server share a secret that is not known by anyone else. In most cases, when users are authenticating to the network, this shared secret is the user password. When the user logs on to a network secured by Kerberos, a hash of the user's password is used to encrypt a packet of information. When the KDC receives the packet, it decrypts the information using the stored user password hash that is stored in AD DS. If the decryption is successful, then the authenticating server knows that the user knows the shared secret and access is granted.

> **Note** When the user logs on, he or she will usually type in a password. The domain controller checks to see if that password is accurate. However, because Kerberos operates with the assumption that the network is insecure, this checking is done without sending the password across the network.

One of the problems with a shared secret authentication model is that the user and the server managing the network resource must have some way of sharing the secret. If one user is trying to access a resource on one server, a user account can be created on the server with a password that only the user knows. When the user tries to access the resources on the server,

that user can present the shared secret (password) and gain access to the resource. However, in a corporate environment, there may be thousands of users and hundreds of servers. Managing distinct shared secrets for all of these users would be impractical. Kerberos deals with this issue by using a Key Distribution Center (KDC). The KDC runs as a service on a domain controller on the network and manages the shared secrets for all users on the network. The KDC has one central database of all user accounts on the network, and it stores the shared secret for each user (in the form of a one-way hash of the user's password). In an AD DS environment, these shared secrets are stored in the AD DS data store. When a user needs access to the network and resources on the network, the KDC confirms that the user knows the shared secret and then authenticates the user. The KDC also stores shared secrets for computers that are members of the AD DS domain, which are used to authenticate computers that are used to access network resources.

Note In Kerberos terminology, this central server that manages user accounts is a KDC, as discussed previously. In the Windows Server 2008 implementation of Kerberos, this server is called a domain controller. Every AD DS domain controller, including read-only domain controllers, is a KDC. In Kerberos, the boundary defined by the user database on one KDC is called a *realm*. In Windows Server 2008 terminology, this boundary is called a *domain*.

Each KDC (which runs as the Kerberos Key Distribution Center service in Windows Server 2008) is made up of two separate services: the Authentication Service (AS) and the Ticket-Granting Service (TGS). The AS is responsible for the initial client logon and issues a Ticket-Granting Ticket (TGT) to the client. The TGS is responsible for all service tickets that are used to access resources on the Windows Server 2008 network.

The KDC stores the account database used for Kerberos authentication. In the Windows Server 2008 implementation of Kerberos, the database is managed by the directory system agent (DSA), which runs within the LSA process on each domain controller. Clients and applications are never given direct access to the account database; all requests must go through the DSA, using one of the AD DS interfaces. Every object within the account database (in fact, every attribute on every object) is protected with an ACL. The DSA ensures that any attempts to access the account database are properly authorized.

Note When AD DS is installed on the first domain controller in the domain, a special account named *krbtgt* is created in the domain. The account cannot be deleted or renamed and should never be enabled or moved from the Users container. The account is assigned a password when it is created, and the password is automatically changed on a regular basis. This password is used to create a secret key that is used to encrypt and decrypt the TGTs issued by all the domain controllers (KDCs) in the domain. Each read-only domain controller is issued a unique krbtgt account when the computer is promoted. This provides cryptographic isolation between KDCs in different branches, which prevents a compromised RODC from issuing service tickets to resources in other branches or a hub site.

Kerberos Authentication

Kerberos authentication begins when the Kerberos security provider is called by the LSA on a Windows Vista workstation or a Windows Server 2008 computer. When a user logs on by typing a user name and password, the client computer applies a one-way hash to the user's password to create a secret key, which is cached in secure memory on the workstation. A one-way hash means that the password cannot be derived from the hash. This hash is also consistent—each time the hash is applied to the same password, the result will always be the same.

Note This process also applies to computers running Windows 2000 Professional or later client operating systems, and to Windows 2000 Server or later server operating systems.

To perform a client logon process, the client and server systems follow these steps:

1. The Kerberos SSP on the workstation sends an authentication message to the KDC. (See Figure 8-1.) The message includes:
   - The user name
   - The user realm (domain name)
   - A request for a TGT
   - Preauthentication data, which includes a time stamp, plus possibly other data

   The preauthentication data is encrypted using the secret key derived from the user password.
2. When the message arrives at the KDC, the server examines the user name and then checks the directory database for its copy of the secret key associated with the user's account. The server decrypts the encrypted data in the message with the secret key and checks the time stamp. If the decryption is successful and the time stamp is recorded as being within five minutes of the current time on the server, the server prepares to authenticate the user. If the decryption fails, the user must have entered the wrong password, and the authentication fails. If the time stamp is more than five minutes off the current time on the server, the authentication will also fail. The reason for the small time difference is to prevent someone from capturing the authentication packets and then replaying them at a later time. The default maximum allowable time difference of *5 minutes* can be configured on the domain security policies.

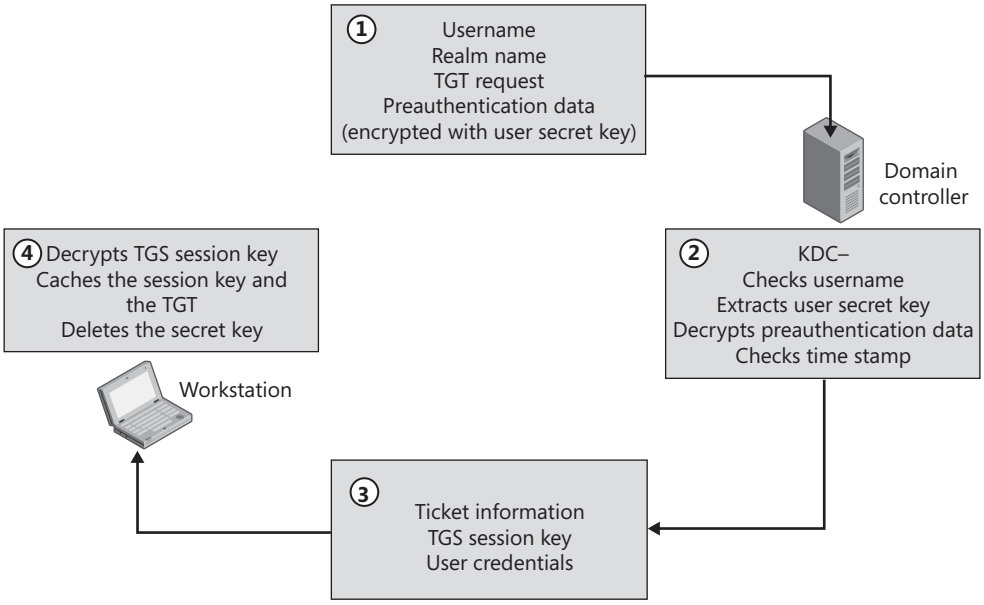


Figure 8-1 Getting a Kerberos TGT.

3. After the user is authenticated, the server sends the client a message that includes a Ticket Granting Service (TGS) *session key* and a *TGT*. (See Figure 8-1.) The session key is an encryption key that is used to interact with the KDC instead of using the client's secret key. The TGT grants the user access to the TGS. For the lifetime of the TGT, the client will present the TGT to the TGS whenever the client needs access to resources on the network. All access tokens for the principal (including the user SID and security group SIDs) are also included in the TGT. This information is known as the Privilege Attribute Certificate (PAC). The TGS session key is encrypted using the user's secret key. In addition, the TGT is encrypted using the KDC's (krbrgt) long-term secret key.
4. When the packet arrives at the client computer, the user's secret key is used to decrypt the TGS session key. If the decryption is successful and the time stamp is valid, the user's computer assumes that the KDC is authentic because it knew the user's secret key. The TGS session key is then cached on the local computer until it expires or until the user logs off the workstation. This TGS session key will be used to encrypt all future connections to the KDC. This means that the client no longer needs to remember the secret key, which is deleted from the workstation cache. The TGT is stored in an encrypted form in the workstation cache.

Note The Kerberos protocol includes the Authentication Service (AS) Exchange, which is the subprotocol used to perform the initial authentication for the user. The process just described uses the AS Exchange subprotocol. The initial message sent by the client to the KDC is called a KRB_AS_REQ message. The server response to the client is called a KRB_AS_REP message.

5. At this point, the user has been authenticated, but the user still does not have access to resources on the network. The TGT grants access to the Ticket Granting Service, but to gain access to any other resources on the network, the user must acquire a service ticket from the TGS. (See Figure 8-2.) The client workstation sends a service ticket request to the TGS. The request includes the name of the target computer, the domain of the target computer, the TGT granted during authentication, the service principal name (SPN) that the user principal(UPN) wants access to , and a time stamp that is encrypted using the TGS session key that was acquired during the AS Exchange process.

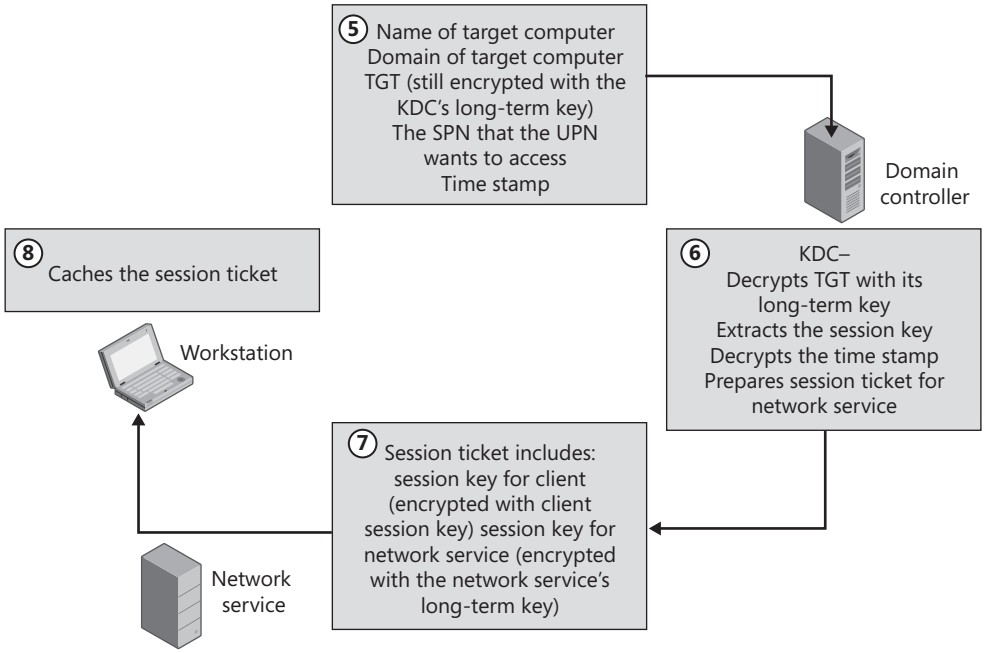


Figure 8-2 Acquiring a Kerberos session ticket for a network resource.

6. The KDC decrypts the TGT using its long-term key. It then extracts the TGS session key from the TGT and decrypts the time stamp to ensure that the client is using the correct session key and that the time stamp is valid. If the session key and time stamp are acceptable, the KDC then does an LDAP query to find the account that has the service principal registered on it. After this is done, it prepares a service ticket for the requested service.

7. The response includes two copies of a service session key. The first copy of the service session key is encrypted using the TGS session key the client obtained during the initial logon. The second copy of the service session key is intended for the service principal hosting the requested service and includes the user's access information—a service ticket. The service ticket is encrypted using the secret key of the service principal hosting the network service, which is unknown to the client workstation but known to both the KDC and the service principal, because the service principal is located in the KDC realm or a trusted Kerberos realm.

8. The client workstation caches both parts of the session ticket in memory.

> **Note** The process described in steps 5 through 8 uses the Ticket-Granting Service (TGS) Exchange subprotocol. The session ticket request sent by the client is called a KRB_TGS_REQ message; the server response is a KRB_TGS_REP message.

9. The client now presents the service ticket to the network service to gain access. (See Figure 8-3.)

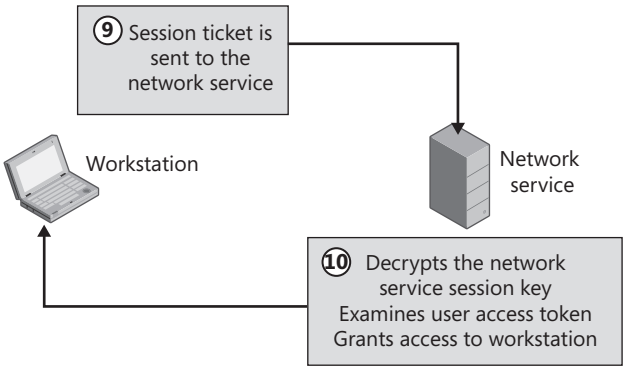


Figure 8-3 Accessing the network service.

10. The network service decrypts the service ticket using its secret key. Using the service session key, the service then decrypts the time stamp included in the request. If both decryptions are successful and the time stamp is within five minutes of the hosting computers time, the service trusts that the ticket comes from the KDC. The network service then determines if the client requested mutual authentication. If the client requests mutual authentication, the service encrypts the time stamp it received in the request, using the service session key and replies back to the client.

    The service sends a reply back to the client when the client requests mutual authentication. If this occurs, the client decrypts the time stamp using the service session key. The client then compares the time stamp it received in the reply with the time stamp it sent in the original request. If the time stamps match, the client then trusts the service.

    After Kerberos authentication completes, the service principal hosting the requested service passes the service ticket to LSA. LSA then decrypts the service ticket and extracts the authorization data. The authorization data includes the user SID and the SIDs of groups of which the user is a member. LSA uses this data to create an access token. The authorization data is known as Privilege Attribute Certificate or PAC.

> **Note** The process described in steps 9 and 10 uses the Client/Server (CS) Exchange subprotocol. The client request is called a KRB_AP_REQ message.

Direct from the Field: Service Principal Names

You can assign service principal names (SPN) to user or computer accounts. Service principal names are stored in a multivalued Active Directory attribute on the user or computer account, which allow each user or computer to have more than one SPN. SPNs *must* be unique across the entire Active Directory forest.

An example of a service principal name attribute on an Active Directory object looks like this:

Host /DC1

Host/DC1.contoso.com

You read this example SPN as follows—the service name of **host** on security principal **DC1.** The next service principal name describes the same service but uses a different security principal, **DC1.contoso.com**. Kerberos authentication relies on names. If your clients connect by both NetBIOS and FQDN names, then you want to ensure that you register the requested service using both names, as shown in the example.

The following is an example of a service principal name for a user account under which the SQL service runs. Windows starts the SQL service using a domain user account (also called a service account). You must register the SQL service principal name on the user account, because that is the account to which you are delegating authentication. A common configuration error is to register the SPN on the computer account that hosts the SQL service.

MSSQLSvc/sqlsrvr.contoso.com:1433

MSSQLSvc/sqlsrvr:1433

Kerberos-enabled services use a preconfigured service identifier in their SPN. In this example, the Microsoft SQL service uses MSSQLSvc. This is how you read this service principal name. The Microsoft SQL service (**MSSQLSvc**) is hosted on the computer **sqlsrvr.contoso.com,** and this instance of SQL is listening on port **1433**. Again, we include both the FQDN and NetBIOS names for the server to ensure that authentication works when connecting to either name. Remember, these service principal names *must* be unique to the Active Directory forest.

Microsoft provides several utilities that you can use to view service principal names. These utilities include LDP, LDIFDE, ADSIEdit, and SETSPN.

Robert Greene

Microsoft Support Escalation Engineer

Assuming the authentication and authorization are successful, the client is given access to the server resources. If the client needs subsequent use of the same resource or service, the session ticket is pulled from the client's ticket cache and is reissued to the target resource server. If the session ticket has expired, the client has to return to the KDC to obtain a new ticket.

Note You can view the contents of the client cache by using two tools. KList.exe, which is installed on Windows Server 2008 computers, provides a command-line interface to view and delete the Kerberos tickets. The Kerberos Tray tool (Kerbtray.exe) provides a graphical user interface (GUI) for viewing the tickets. Figure 8-4 shows an example of the information provided by the Kerberos Tray tool. The Kerberos Tray tool is available as part of the Windows Server 2003 Resource Kit tools, which can be downloaded at *https://www.microsoft.com/downloads/details.aspx?FamilyID=9d467a69-57ff-4ae7-96ee-b18c4790cffd&displaylang=en*.

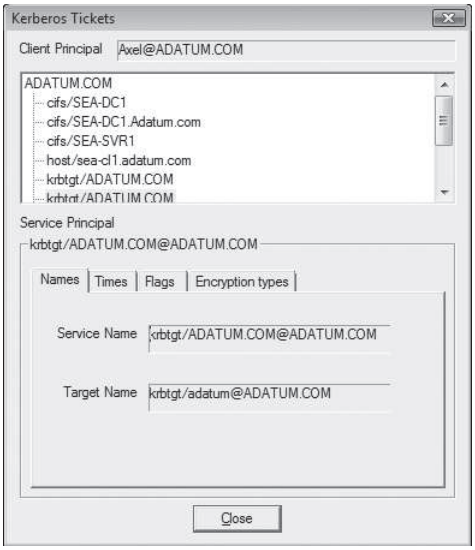


Figure 8-4 Viewing Kerberos tickets using the Kerberos Tray tool.

How It Works: Kerberos Ticket Flags

Ticket flags are configured within all tickets and are used to identify the ticket purpose and/or constraints. You can view these flags when you use the Kerberos Tray utility. Kerberos tickets use the following ticket flags:

- ***Forwardable*** Only valid for a TGT. Instructs the Ticket Granting Server that it can issue a new TGT with a different network address based on the TGT that is presented. A forwardable ticket can be used in Kerberos delegation.

- ***Forwarded*** Indicates that the TGT has been forwarded or that a ticket was issued from a forwardable TGT. The middle-tier application in Kerberos delegation should have this type of ticket.
- ***Proxiable*** A proxiable ticket is a ticket (generally only a TGT) that allows you to get a ticket for a service with IP addresses other than the ones in the TGT. This is different than aforwardable ticket in that you cannot proxy a new TGT from your current TGT; you can only proxy non-TGT service tickets.
- ***Proxy*** This ticket flag indicates that the service ticket was obtained from a proxiable TGT.
- ***Renewable*** Indicates if the ticket can be renewed. This is used in combination with the EndTime and Renew-Till Fields to cause tickets to be renewed at the KDC periodically.
- ***Initial*** Indicates that this is a Ticket Granting Ticket (TGT).

Other flags include *May Postdate, Invalid, HW Auth, OK As Delegate,* and others. Postdated tickets may be requested in advance for batch processing and so on, but they are flagged invalid until the time that they become effective. A KDC must validate the ticket and remove the Invalid flag at that time.

Robert Greene

Microsoft Support Escalation Engineer

This process of gaining access to a resource on the network means that the KDC is only involved during the initial client logon and the first time the client tries to access a specific resource on a specific server. When the user first logs on, that user is given a TGT that gives the client access to the KDC during the lifetime of the ticket. When the client tries to connect to a network resource, the client contacts the KDC again and gets a service ticket to access that resource. This service ticket includes the authorization data for the user. After successful authentication, the Local Security subsystem uses this data to create an access token on the computer hosting the service or resource so the server can determine the level of resource access the user should have.

Authenticating Across Domain Boundaries

The same authentication process applies when a user authenticates across domain boundaries. For example, a company may have a three-domain forest, as shown in Figure 8-5.

If a user with an account in TreyResearch.com travels to the NA.ADatum.com domain location and tries to log on to the network on a machine in the NA.ADatum.com domain, the client workstation must be able to connect to a domain controller (KDC) in the TreyResearch.com as well as NA.ADatum.com and ADatum.com domain. In this case,

the client computer sends the initial logon request to the NA.ADatum.com domain controller. The domain controller determines that the user account is located in the TreyResearch.com domain, so it needs to refer the client workstation to that domain. If all of the domains were configured with shortcut trusts with each other, the domain controller could directly refer the client computer to a domain controller in the TreyResearch.com domain. However, if no shortcut trusts have been created, there is no direct trust between NA.ADatum.com and TreyResearch.com. In this case, the NA domain controller will refer the client computer to a domain controller in the ADatum.com domain. The referral includes a TGT for ADatum.com that is encrypted with an inter-realm session key, which is shared by both ADatum.com and NA.Adatum.com. This TGT allows access to KDC service on the domain controller in the Adatum.com domain. The inter-realm session key was created when the NA domain was added to the Adatum.com forest and the initial trust was created between the two domains. The inter-realm session key guarantees that the logon request is coming from a trusted domain. The client computer then sends an authentication request to the ADatum.com domain. The client is then referred to a domain controller in the TreyResearch.com domain. Again, this referral includes TGT for TreyResearch.com that has an inter-realm session key to access the KDC services on the domain controller in TreyResearch.com. The client computer then sends a TGT request to the home domain controller in TreyResearch.com.

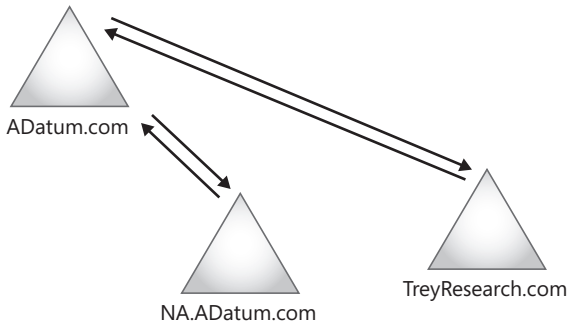


Figure 8-5 Authentication across domain boundaries.

A similar process is followed when a client tries to gain access to a resource on a domain other than the user's home domain. In this case, the client needs to acquire a service ticket from a domain controller in the domain where the resource is located, so the client will be referred through the same process until it can connect to the right domain controller.

This authentication process has implications for forest design, especially if users frequently log on to domains other than their home domain or access resources in domains other than the home domain. If you are designing a forest with multiple domains, the client may have to traverse the entire trust path between the domains. If this happens often, you may want to put domain controllers for the root domains in locations close to the users. You can also use shortcut trusts so that the domain controller referrals can be sent directly to the appropriate domains.

Delegation of Authentication

One of the issues that can complicate accessing network services is that the network service may be distributed across multiple servers. For example, the client might connect to a front-end server that must connect to a back-end database server to collect some information. In this environment, the user's credentials (rather than the front-end server's credentials) should be used to access the back-end server so that the user will only get access to authorized information. In Windows 2000, Kerberos provides this functionality in two ways: using proxy tickets and using forwarded tickets. If proxy tickets are enabled, the client will send a session ticket request to the KDC requesting access to the back-end server. The KDC grants the session ticket and sets the *Proxiable* flag on the ticket. The client then presents the session ticket to the front-end server, which uses the ticket to access information on the back-end server. The main problem with proxy tickets is that the client must know the identity of the back-end server. The other option is to use forwarded tickets. If these tickets are enabled, the client will send an AS Exchange request to the KDC requesting a TGT that the front-end server will be able to use to access back-end servers. The KDC creates a TGT and sends it to the client. The client sends the TGT to the front-end server, which then uses the TGT to acquire a session ticket to access the back-end server on behalf of the client.

There are two significant concerns with the way delegation of authentication is implemented in Windows 2000. The first concern is that delegation of authentication can only be used if the client is authenticated using Kerberos. This means that no Windows NT, Microsoft Windows 95, or Windows 98 client can use delegation of authentication. The second Windows 2000–related concern is related to the security of the delegation. In Windows 2000, after the front-end server obtains the forwarded ticket from the KDC, it can use the ticket to access any network service on behalf of the client. Windows Server 2003 and Windows Server 2008 provide the option for constrained delegation, which means you can configure the account so it is delegated for only specific services on the network (based on service principal names). Constrained delegation is available only when the domain is set to Windows Server 2003 or Windows Server 2008 functional level.

In order for the delegation of authentication to be successful, you must ensure that both the user account and the service or computer account are configured to support delegation of authentication. To configure this on a user account, access the user's Properties sheet through the Active Directory Users and Computers administrative tool, select the Account tab; then scroll through the Account Options list and make sure that the Account Is Sensitive And Cannot Be Delegated option is not selected. (This is not selected by default.) To configure the service account for delegation, you must first determine whether the logon account used by the service is a normal user account or whether it is the LocalSystem account. If the service runs under a normal user account, first ensure that you have added an SPN to the user account. Then access the user's Account tab and make sure the Account Is Sensitive And Cannot Be Delegated option is not selected. (This is not selected by default.) Also check the appropriate level of delegation on the delegation tab of the user account. (Figure 8-6 shows the interface.)

If the service runs under a LocalSystem account, the delegation must be configured on the computer account's Properties sheet. To implement the Windows 2000 level of authentication, select the Trust This Computer For Delegation To Any Service (Kerberos Only) option. To implement the Windows Server 2003 or Windows Server 2008 enhancements, select the Trust This Computer For Delegation To Specified Services Only option. You can then select whether the client must authenticate using Kerberos only or can use any protocol and then select the services (based on service principal names registered in AD DS) to which the computer can present delegated credentials.

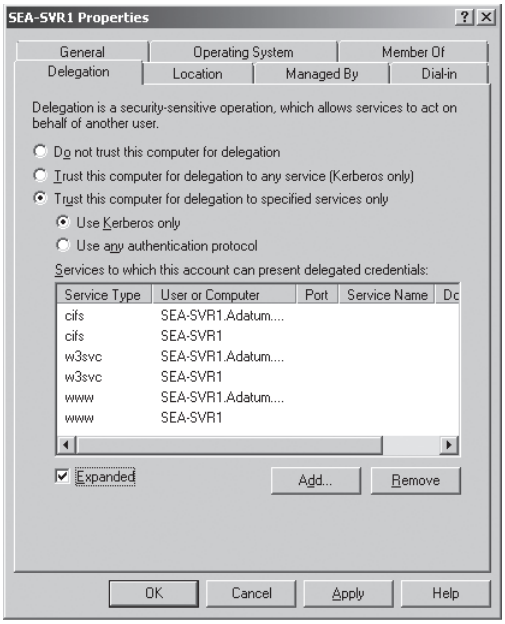


Figure 8-6 Configuring constrained delegation on a computer account.

Direct from the Field: Delegated Authentication

Delegated authentication is becoming more frequent in customer environments because of the need to limit user access to confidential data as well as a need to be able to audit what data users are accessing on a day-to-day basis. Today's applications are typically multi-tier solutions, which increases the complexity of authentication. For example, delegated authentication is required when a Web-based application must request data from a SQL database using the credentials of the user who authenticated to the Web site. Following are the most common problems that cause delegated authentication to fail:

- The client application is not configured to use Kerberos. In the example given, the Web browser is the client application that you must configure to support Kerberos authentication.
- SPNs are not registered on the correct service account. In the example, the service account that runs the Web application must have the proper SPNs registered for

the name that users use to connect to the Web application. If users connect using both NetBIOS and FQDN names, then you must register both names on the service account (http/webapp1 and http/webapp1.contoso.com). Also, the correct service account in the example is the service account used to run the Web application pool, which includes the Web application that connects to SQL. Lastly, no SPN registration is required if the Web application pool is running as the network service.

- Duplicate SPNs can also cause delegated authentication to fail. Each SPN in the domain must be unique. Two security principals, each having the same SPN, causes the delegation to fail.
- There is an incorrect configuration on the IIS server. You must configure IIS to use Kerberos authentication. Ensure that the Web application is configured for Integrated authentication. Also, make sure the Negotiate authentication provider has not been disabled. For more information, see the Microsoft Knowledge Base article 215383, "How to Configure IIS to Support Both the Kerberos Protocol and the NTLM Protocol for Network Authentication" available at *http://support.microsoft.com/kb/215383*.

Robert Greene

Microsoft Support Escalation Engineer

Configuring Kerberos in Windows Server 2008

As mentioned earlier, Kerberos is the default authentication protocol for clients using Windows 2000 or later to log on to AD DS. You can configure several Kerberos properties through the domain security policy. To access the Kerberos policy settings, open the Group Policy Management console and edit the Default Domain Policy. Under Computer Configuration, first expand Security settings and then expand the Account Policies folder. (The interface is shown in Figure 8-7.)

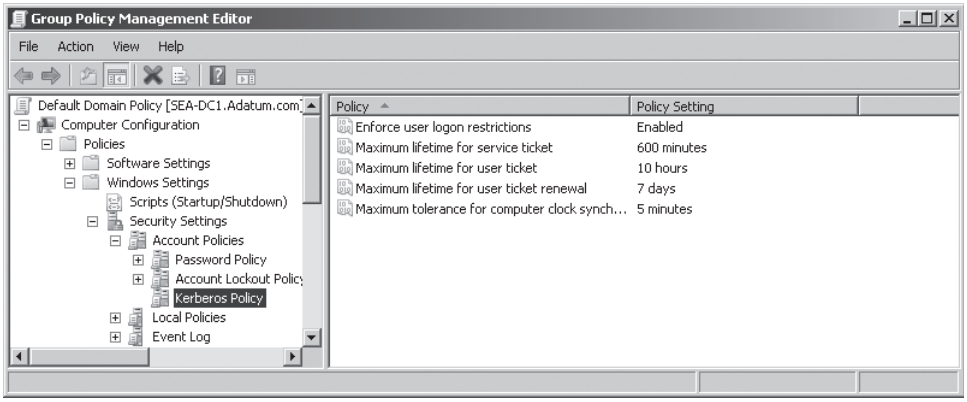


Figure 8-7 Configuring the Kerberos settings through Domain Security Policy.