Configuring Kerberos

Kerberos is a secret-key network authentication protocol, developed at the Massachusetts Institute of Technology (MIT), that uses the Data Encryption Standard (DES) cryptographic algorithm for encryption and authentication. Kerberos was designed to authenticate requests for network resources. Kerberos, like other secret-key systems, is based on the concept of a trusted third party that performs secure verification of users and services. In the Kerberos protocol, this trusted third party is called the key distribution center (KDC).

- Finding Feature Information, on page 1
- Prerequisites for Controlling Switch Access with Kerberos, on page 1
- Information About Kerberos, on page 2
- How to Configure Kerberos, on page 6
- Configuration Examples for Kerberos, on page 12
- Additional References, on page 22
- Feature Information for Kerberos, on page 23

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table at the end of this module.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to https://cfnng.cisco.com/. An account on Cisco.com is not required.

Prerequisites for Controlling Switch Access with Kerberos

The following are the prerequisites for controlling switch access with Kerberos.

- So that remote users can authenticate to network services, you must configure the hosts and the KDC in the Kerberos realm to communicate and mutually authenticate users and network services. To do this, you must identify them to each other. You add entries for the hosts to the Kerberos database on the KDC and add KEYTAB files generated by the KDC to all hosts in the Kerberos realm. You also create entries for the users in the KDC database.
- A Kerberos server can be a switch that is configured as a network security server and that can authenticate users by using the Kerberos protocol.

When you add or create entries for the hosts and users, follow these guidelines:

- The Kerberos principal name *must* be in all lowercase characters.
- The Kerberos instance name *must* be in all lowercase characters.
- The Kerberos realm name *must* be in all uppercase characters.

Information About Kerberos

Kerberos and Switch Access

This section describes how to enable and configure the Kerberos security system, which authenticates requests for network resources by using a trusted third party.

Note In the Kerberos configuration examples, the trusted third party can be any switch that supports Kerberos, that is configured as a network security server, and that can authenticate users by using the Kerberos protocol.

Kerberos Overview

Kerberos is a secret-key network authentication protocol, which was developed at the Massachusetts Institute of Technology (MIT). It uses the Data Encryption Standard (DES) cryptographic algorithm for encryption and authentication and authenticates requests for network resources. Kerberos uses the concept of a trusted third party to perform secure verification of users and services. This trusted third party is called the *key distribution center* (KDC).

Kerberos verifies that users are who they claim to be and the network services that they use are what the services claim to be. To do this, a KDC or trusted Kerberos server issues tickets to users. These tickets, which have a limited life span, are stored in user credential caches. The Kerberos server uses the tickets instead of user names and passwords to authenticate users and network services.

Note A Kerberos server can be any switch that is configured as a network security server and that can authenticate users by using the Kerberos protocol.

The Kerberos credential scheme uses a process called *single logon*. This process authenticates a user once and then allows secure authentication (without encrypting another password) wherever that user credential is accepted.

This software release supports Kerberos 5, which allows organizations that are already using Kerberos 5 to use the same Kerberos authentication database on the KDC that they are already using on their other network hosts (such as UNIX servers and PCs).

Kerberos supports these network services:

- Telnet
- rlogin

- rsh

This table lists the common Kerberos-related terms and definitions.

Table 1: Kerberos Terms

Term	Definition
Authentication	A process by which a user or service identifies itself to another service. For example, a client can authenticate to a switch or a switch can authenticate to another switch.
Authorization	A means by which the switch identifies what privileges the user has in a network or on the switch and what actions the user can perform.
Credential	A general term that refers to authentication tickets, such as TGTs[1] and service credentials. Kerberos credentials verify the identity of a user or service. If a network service decides to trust the Kerberos server that issued a ticket, it can be used in place of re-entering a username and password. Credentials have a default life span of eight hours.
Instance	An authorization level label for Kerberos principals. Most Kerberos principals are of the form *user@REALM* (for example, smith@EXAMPLE.COM). A Kerberos principal with a Kerberos instance has the form *user/instance@REALM* (for example, smith/admin@EXAMPLE.COM). The Kerberos instance can be used to specify the authorization level for the user if authentication is successful. The server of each network service might implement and enforce the authorization mappings of Kerberos instances but is not required to do so. **Note** The Kerberos principal and instance names *must* be in all lowercase characters. **Note** The Kerberos realm name *must* be in all uppercase characters.
KDC[2]	Key distribution center that consists of a Kerberos server and database program that is running on a network host.
Kerberized	A term that describes applications and services that have been modified to support the Kerberos credential infrastructure.
Kerberos realm	A domain consisting of users, hosts, and network services that are registered to a Kerberos server. The Kerberos server is trusted to verify the identity of a user or network service to another user or network service. **Note** The Kerberos realm name *must* be in all uppercase characters.
Kerberos server	A daemon that is running on a network host. Users and network services register their identity with the Kerberos server. Network services query the Kerberos server to authenticate to other network services.
KEYTAB[3]	A password that a network service shares with the KDC. In Kerberos 5 and later Kerberos versions, the network service authenticates an encrypted service credential by using the KEYTAB to decrypt it. In Kerberos versions earlier than Kerberos 5, KEYTAB is referred to as SRVTAB[4].

Term	Definition
Principal	Also known as a Kerberos identity, this is who you are or what a service is according to the Kerberos server. **Note** The Kerberos principal name *must* be in all lowercase characters.
Service credential	A credential for a network service. When issued from the KDC, this credential is encrypted with the password shared by the network service and the KDC. The password is also shared with the user TGT.
SRVTAB	A password that a network service shares with the KDC. In Kerberos 5 or later Kerberos versions, SRVTAB is referred to as KEYTAB.
TGT	Ticket granting ticket that is a credential that the KDC issues to authenticated users. When users receive a TGT, they can authenticate to network services within the Kerberos realm represented by the KDC.

[1] ticket granting ticket
[2] key distribution center
[3] key table
[4] server table

Kerberos Operation

A Kerberos server can be a switch that is configured as a network security server and that can authenticate remote users by using the Kerberos protocol. Although you can customize Kerberos in a number of ways, remote users attempting to access network services must pass through three layers of security before they can access network services.

Kerberos Operation

A Kerberos server can be a switch that is configured as a network security server and that can authenticate remote users by using the Kerberos protocol. Although you can customize Kerberos in a number of ways, remote users attempting to access network services must pass through three layers of security before they can access network services.

To authenticate to network services by using a switch as a Kerberos server, remote users must follow these steps:

Authenticating to a Boundary Switch

This section describes the first layer of security through which a remote user must pass. The user must first authenticate to the boundary switch. This process then occurs:

1. The user opens an un-Kerberized Telnet connection to the boundary switch.
2. The switch prompts the user for a username and password.
3. The switch requests a TGT from the KDC for this user.
4. The KDC sends an encrypted TGT that includes the user identity to the switch.

5. The switch attempts to decrypt the TGT by using the password that the user entered.
   - If the decryption is successful, the user is authenticated to the switch.
   - If the decryption is not successful, the user repeats Step 2 either by re-entering the username and password (noting if Caps Lock or Num Lock is on or off) or by entering a different username and password.

A remote user who initiates a un-Kerberized Telnet session and authenticates to a boundary switch is inside the firewall, but the user must still authenticate directly to the KDC before getting access to the network services. The user must authenticate to the KDC because the TGT that the KDC issues is stored on the switch and cannot be used for additional authentication until the user logs on to the switch.

Obtaining a TGT from a KDC

This section describes the second layer of security through which a remote user must pass. The user must now authenticate to a key distribution center (KDC) and obtain a ticket granting ticket (TGT) from the KDC to access network services.

When a remote user authenticates to a boundary device, that user technically becomes part of the network; that is, the network is extended to include the remote user and the user's machine or network. To gain access to network services, however, the remote user must obtain a TGT from the KDC. The following process describes how remote users authenticate to the KDC:

1. The remote user, at a workstation on a remote site, launches the KINIT program (part of the client software provided with the Kerberos protocol).
2. The KINIT program finds the identity of the user and requests a TGT from the KDC.
3. The KDC creates a TGT, which contains the identity of the user, the identity of the KDC, and the expiration time of the TGT.
4. Using the user's password as a key, the KDC encrypts the TGT and sends the TGT to the workstation.
5. When the KINIT program receives the encrypted TGT, it prompts the user for a password (this is the password that is defined for the user in the KDC).
6. If the KINIT program can decrypt the TGT with the password the user enters, the user is authenticated to the KDC, and the KINIT program stores the TGT in the user's credential cache.

At this point, the user has a TGT and can communicate securely with the KDC. In turn, the TGT allows the user to authenticate to other network services.

Authenticating to Network Services

This section describes the third layer of security through which a remote user must pass. The user with a ticket granting ticket (TGT) must now authenticate to the network services in a Kerberos realm.

The following process describes how a remote user with a TGT authenticates to network services within a given Kerberos realm. Assume the user is on a remote workstation (Host A) and wants to log in to Host B.

1. The user on Host A initiates a Kerberized application (such as Telnet) to Host B.

2. The Kerberized application builds a service credential request and sends it to the KDC. The service credential request includes (among other things) the user's identity and the identity of the desired network service. The TGT is used to encrypt the service credential request.
3. The KDC tries to decrypt the service credential request with the TGT it issued to the user on Host A. If the KDC can decrypt the packet, it is assured that the authenticated user on Host A sent the request.
4. The KDC notes the network service identity in the service credential request.
5. The KDC builds a service credential for the appropriate network service on Host B on behalf of the user on Host A. The service credential contains the client's identity and the desired network service's identity.
6. The KDC then encrypts the service credential twice. It first encrypts the credential with the SRVTAB that it shares with the network service identified in the credential. It then encrypts the resulting packet with the TGT of the user (who, in this case, is on Host A).
7. The KDC sends the twice-encrypted credential to Host A.
8. Host A attempts to decrypt the service credential with the user's TGT. If Host A can decrypt the service credential, it is assured the credential came from the real KDC.
9. Host A sends the service credential to the desired network service. Note that the credential is still encrypted with the SRVTAB shared by the KDC and the network service.
10. The network service attempts to decrypt the service credential using its SRVTAB.
11. If the network service can decrypt the credential, it is assured the credential was in fact issued from the KDC. Note that the network service trusts anything it can decrypt from the KDC, even if it receives it indirectly from a user. This is because the user first authenticated with the KDC.

At this point, the user is authenticated to the network service on Host B. This process is repeated each time a user wants to access a network service in the Kerberos realm.

How to Configure Kerberos

To set up a Kerberos-authenticated server-client system, follow these steps:

- Configure the KDC by using Kerberos commands.
- Configure the switch to use the Kerberos protocol.

Configuring the KDC Using Kerberos Commands

After a host is configured to function as the KDC in the Kerberos realm, entries must be made to the KDC database (and to modify existing database information) for all principals in the realm. Principals can be network services on devices and hosts or principals can be users.

Note All Kerberos command examples are based on Kerberos 5 Beta 5 of the original MIT implementation. Later versions use a slightly different interface.

Adding Users to the KDC Database

Follow these steps to add users to the KDC and create privileged instances for those users:

Procedure

Step 1 Use the **su** command to become root on the host running the KDC.

Step 2 Use the **kdb5_edit** program to configure the commands in the next steps.

Note The Kerberos realm name in the following steps must be in uppercase characters.

Step 3 Use the **ank** (add new key) command in privileged EXEC mode to add a user to the KDC. This command prompts for a password that the user must enter to authenticate the router. For example:

Example:

```
Device # ank username@REALM
```

Step 4 Use the **ank** command to add a privileged instance of a user. For example:

```
Device # ank username/instance@REALM
```

Example

The following example adds the user *loki* to the Kerberos realm COMPANY.COM:

```
ank loki@COMPANY.COM
```

Privileged instances can be created to allow network administrators to connect to the router at the enable level so that a clear text password is not used to avoid compromising security and to enter enabled modes. See the Enabling Kerberos Instance Mapping, on page 12 for more information on mapping Kerberos instances to various Cisco IOS privilege levels.

Creating and Extracting a SRVTAB on the KDC

All devices authenticated through Kerberos must have a SRVTAB that contains the password or randomly generated key for the service principal key that was entered into the KDC database. A service principal key must be shared with the host running that service. To do this, the SRVTAB entry must be saved (extracted) to a file and copied to the device and all hosts in the Kerberos realm.

Follow these steps to make a SRVTAB entry and extract this SRVTAB to a file on the KDC in privileged EXEC mode:

Procedure

Step 1 Use the **ark** (add random key) command to add a network service supported by a host or device to the KDC. For example:

Example:

```
Device# ark
SERVICE/HOSTNAME@REALM
```

Step 2 Use the kdb5_edit command **xst** to write an SRVTAB entry to a file. For example:

Example:

```
Device# xst
 device-name host
```

Step 3 Use the **quit** command to exit the kdb5_edit program.

Example

The following example shows how to add a Kerberized authentication service for a device called *device1* to the Kerberos realm COMPANY.COM:

```
ark host/device1.company.com@COMPANY.COM
```

The following example shows how to write an entry for all network services on all Kerberized hosts that use this KDC for authentication to a file:

```
xst device1.company.com@COMPANY.COM host
```

Configuring the Device to Use the Kerberos Protocol

Defining a Kerberos Realm

For a device to authenticate a user defined in the Kerberos database, it must know the host name or IP address of the host running the KDC, the name of the Kerberos realm and, optionally, be able to map the host name or Domain Name System (DNS) domain to the Kerberos realm.

To configure the device to authenticate to a specified KDC in a specified Kerberos realm, use the following commands in global configuration mode. Note that DNS domain names must begin with a leading dot (.):

Procedure

	Command or Action	Purpose
Step 1	Device(config)# **kerberos local-realm***kerberos-realm*	Defines the default realm for the device.
Step 2	Device(config)# **kerberos server***kerberos-realm* {*hostname* | *ip-address* } [*port-number*]	Specifies to the device which KDC to use in a given Kerberos realm and, optionally, the port

	Command or Action	Purpose
		number that the KDC is monitoring. (The default is 88.)
Step 3	Device(config)# **kerberos realm** {*dns-domain* | *host* } *kerberos-realm*	(Optional) Maps a host name or DNS domain to a Kerberos realm.

What to do next

Note Because the machine running the KDC and all Kerberized hosts must interact within a 5-minute window or authentication fails, all Kerberized machines, and especially the KDC, should be running the Network Time Protocol (NTP).

The **kerberos local-realm**, **kerberos realm**, and **kerberos server** commands are equivalent to the UNIX krb.conf file. The table below identifies mappings from the Cisco IOS configuration commands to a Kerberos 5 configuration file (krb5.conf).

Table 2: Kerberos 5 Configuration File and Commands

krb5.conf File	Cisco IOS Configuration Command
[libdefaults] `default_realm = DOMAIN.COM`	(in configuration mode) **`kerberos local-realm`** *`DOMAIN.COM`*

[domain_realm]
 	
`.domain.com = DOMAIN.COM`
 	
`domain.com = DOMAIN.COM`
	(in configuration mode)
	
	`kerberos realm`
	`.domain.com`
	`DOMAIN.COM`
	`kerberos realm`
	`domain.com DOMAIN.COM`

[realms]
 	
`kdc = DOMAIN.PIL.COM:750`
 	
`admin_server = DOMAIN.PIL.COM`
 	
`default_domain = DOMAIN.COM`
	(in configuration mode)
	
	`kerberos server`
	`DOMAIN.COM 172.65.44.2`
	`(`*`172.65.44.2`*
	`is the example IP address for DOMAIN.PIL.COM`
	`)`

Copying SRVTAB Files

To make it possible for remote users to authenticate to the device using Kerberos credentials, the device must share a secret key with the KDC. To do this, you must give the device a copy of the SRVTAB you extracted on the KDC.

The most secure method to copy an SRVTAB file to the hosts in your Kerberos realm is to copy it onto physical media and go to each host in turn and manually copy the files onto the system. To copy an SRVTAB file to the device, which does not have a physical media drive, it must be transfered over the network using TFTP.

To remotely copy an SRVTAB file to the device from the KDC, use the **kerberos srvtab remote**command in global configuration mode:

Device(config)# **kerberos srvtab remote** {*hostname* | *ip-address* } {*filename* }

When you copy the SRVTAB file from the device to the KDC, the **kerberos srvtab remote** command parses the information in this file and stores it in the running configuration of the device, in the **kerberos srvtab entry**format. To ensure that the SRVTAB is available (does not need to be acquired from the KDC) when you reboot the device, use the **write memory** configuration command to write your running configuration (which contains the parsed SRVTAB file) to NVRAM.

Specifying Kerberos Authentication

See the Configuring Authentication feature module for more information on configuring authentication on the device **aaa authentication** command is used to specify Kerberos as the authentication method.

Enabling Credentials Forwarding

With Kerberos configured thus far, a user authenticated to a Kerberized device has a TGT and can use it to authenticate to a host on the network. However, if the user tries to list credentials after authenticating to a host, the output will show no Kerberos credentials present.

You can optionally configure the device to forward users' TGTs with them as they authenticate from the device to Kerberized remote hosts on the network when using Kerberized Telnet, rcp, rsh, and rlogin (with the appropriate flags).

To force all clients to forward users' credentials as they connect to other hosts in the Kerberos realm, use the following command in global configuration mode:

Command	Purpose
`Device(config)# kerberos credentials forward`	Forces all clients to forward user credentials upon successful Kerberos authentication.

With credentials forwarding enabled, users' TGTs are automatically forwarded to the next host they authenticate to. In this way, users can connect to multiple hosts in the Kerberos realm without running the KINIT program each time to get a new TGT.

Opening a Telnet Session to a Device

To use Kerberos to authenticate users opening a Telnet session to the device from within the network, use the following command in global configuration mode:

Command	Purpose
`Device(config)# aaa authentication login {default | list-name } krb5_telnet`	Sets login authentication to use the Kerberos 5 Telnet authentication protocol when using Telnet to connect to the device.

Although Telnet sessions to the device are authenticated, users must still enter a clear text password if they want to enter enable mode. The **kerberos instance map** command, discussed in a later section, allows them to authenticate to the device at a predefined privilege level.

Establishing an Encrypted Kerberized Telnet Session

Another way for users to open a secure Telnet session is to use Encrypted Kerberized Telnet. With Encrypted Kerberized Telnet, users are authenticated by their Kerberos credentials before a Telnet session is established. The Telnet session is encrypted using 56-bit Data Encryption Standard (DES) encryption with 64-bit Cipher Feedback (CFB). Because data sent or received is encrypted, not clear text, the integrity of the dialed device or access server can be more easily controlled.

Note This feature is available only if you have the 56-bit encryption image. 56-bit DES encryption is subject to U.S. Government export control regulations.

To establish an encrypted Kerberized Telnet session from a device to a remote host, use either of the following commands in EXEC command mode:

Command	Purpose

`Device(config)# connect host [port] /encrypt kerberos`
or
`Device(config)# telnet host [port] /encrypt kerberos`
	Establishes an encrypted Telnet session.

When a user opens a Telnet session from a device to a remote host, the device and remote host negotiate to authenticate the user using Kerberos credentials. If this authentication is successful, the device and remote host then negotiate whether or not to use encryption. If this negotiation is successful, both inbound and outbound traffic is encrypted using 56-bit DES encryption with 64-bit CFB.

When a user dials in from a remote host to a device configured for Kerberos authentication, the host and device will attempt to negotiate whether or not to use encryption for the Telnet session. If this negotiation is successful, the device will encrypt all outbound data during the Telnet session.

If encryption is not successfully negotiated, the session will be terminated and the user will receive a message stating that the encrypted Telnet session was not successfully established.

Enabling Mandatory Kerberos Authentication

As an added layer of security, you can optionally configure the device so that, after remote users authenticate to it, these users can authenticate to other services on the network only with Kerberized Telnet, rlogin, rsh, and rcp. If you do not make Kerberos authentication mandatory and Kerberos authentication fails, the application attempts to authenticate users using the default method of authentication for that network service; for example, Telnet and rlogin prompt for a password, and rsh attempts to authenticate using the local rhost file.

To make Kerberos authentication mandatory, use the following command in global configuration mode:

Command	Purpose
`Device(config)# kerberos clients mandatory`	Sets Telnet, rlogin, rsh, and rcp to fail if they cannot negotiate the Kerberos protocol with the remote server.

Enabling Kerberos Instance Mapping

You can create administrative instances of users in the KDC database. The **kerberos instance map** command allows you to map those instances to Cisco IOS privilege levels so that users can open secure Telnet sessions to the device at a predefined privilege level, obviating the need to enter a clear text password to enter enable mode.

To map a Kerberos instance to a Cisco IOS privilege level, use the following command in global configuration mode:

Command	Purpose
`Device(config)# kerberos instance map` *instance privilege-level*	Maps a Kerberos instance to a Cisco IOS privilege level.

If there is a Kerberos instance for user *loki* in the KDC database (for example, *loki/admin*), user *loki* can now open a Telnet session to the device as loki/admin and authenticate automatically at privilege level 15, assuming instance “admin” is mapped to privilege level 15.

Cisco IOS commands can be set to various privilege levels using the **privilege level**command.

After you map a Kerberos instance to a Cisco IOS privilege level, you must configure the device to check for Kerberos instances each time a user logs in. To run authorization to determine if a user is allowed to run an EXEC shell based on a mapped Kerberos instance, use the **aaa authorization** command with the **krb5-instance** keyword. For more information, refer to the chapter “Configuring Authorization.”

Monitoring the Kerberos Configuration

To display the Kerberos configuration, use the following commands:

- **show running-config**
- **show kerberos creds**: Lists the credentials in a current user’s credentials cache.
- **clear kerberos creds**: Destroys all credentials in a current user’s credentials cache, including those forwarded.

Configuration Examples for Kerberos

Example: Defining a Kerberos Realm

To define CISCO.COM as the default Kerberos realm, use the following command:

```
kerberos local-realm CISCO.COM
```

To tell the device that the CISCO.COM KDC is running on host 10.2.3.4 at port number 170, use the following Kerberos command:

```
kerberos server CISCO.COM 10.2.3.4 170
```

To map the DNS domain cisco.com to the Kerberos realm CISCO.COM, use the following command:

```
kerberos realm.cisco.com CISCO.COM
```

Example: Copying a SRVTAB File

To copy over the SRVTAB file on a host named host123.cisco.com for a device named device1.cisco.com, the command would look like this:

```
kerberos srvtab remote host123.cisco.com device1.cisco.com-new-srvtab
```

Example: Configuring Kerberos

This section provides a typical non-Kerberos device configuration and shows output for this configuration from the **write term** command, then builds on this configuration by adding optional Kerberos functionality. Output for each configuration is presented for comparison against the previous configuration.

This example shows how to use the kdb5_edit program to perform the following configuration tasks:

- Adding user chet to the Kerberos database
- Adding a privileged Kerberos instance of user chet (chet/admin) to the Kerberos database
- Adding a restricted instance of chet (chet/restricted) to the Kerberos database
- Adding workstation chet-ss20.cisco.com
- Adding device chet-2500.cisco.com to the Kerberos database
- Adding workstation chet-ss20.cisco.com to the Kerberos database
- Extracting SRVTABs for the device and workstations
- Listing the contents of the KDC database (with the **ldb** command)

Note In this sample configuration, host chet-ss20 is also the KDC:

```
chet-ss20# sbin/kdb5_edit
kdb5_edit:  ank chet
Enter password:
Re-enter password for verification:
kdb5_edit:  ank chet/admin
Enter password:
Re-enter password for verification:
kdb5_edit:  ank chet/restricted
Enter password:
Re-enter password for verification:
kdb5_edit:  ark host/chet-ss20.cisco.com
```

```
kdb5_edit:  ark host/chet-2500.cisco.com
kdb5_edit:  xst chet-ss20.cisco.com host
'host/chet-ss20.cisco.com@CISCO.COM' added to keytab 'WRFILE:chet-ss20.cisco.com-new-srvtab'
kdb5_edit:  xst chet-2500.cisco.com host
'host/chet-2500.cisco.com@CISCO.COM' added to keytab 'WRFILE:chet-2500.cisco.com-new-srvtab'
kdb5_edit:  ldb
entry: host/chet-2500.cisco.com@CISCO.COM
entry: chet/restricted@CISCO.COM
entry: chet@CISCO.COM
entry: K/M@CISCO.COM
entry: host/chet-ss20.cisco.com@CISCO.COM
entry: krbtgt/CISCO.COM@CISCO.COM
entry: chet/admin@CISCO.COM
kdb5_edit:  q
chet-ss20#
```

The following example shows output from a **write term** command, which displays the configuration of device chet-2500. This is a typical configuration with no Kerberos authentication.

```
chet-2500# write term
Building configuration...
Current configuration:
!
! Last configuration
change at 14:03:55 PDT Mon May 13 1996
!
version 11.2
service udp-small-servers
service tcp-small-servers
!
hostname chet-2500
!
clock timezone PST -8
clock summer-time PDT recurring
aaa new-model
aaa authentication login console none
aaa authentication ppp local local
enable password sMudgKin
!
username chet-2500 password 7 sMudgkin
username chet-3000 password 7 sMudgkin
username chetin password 7 sMudgkin
!
interface Ethernet0
 ip address 172.16.0.0 255.255.255.0
!
interface Serial0
 no ip address
 shutdown
 no fair-queue
!
interface Serial1
 no ip address
 shutdown
 no fair-queue
!
interface Async2
 ip unnumbered Ethernet0
 encapsulation ppp
 shutdown
 async dynamic routing
 async mode dedicated
 no cdp enable
```

```
 ppp authentication pap local
 no tarp propagate
!
interface Async3
 ip unnumbered Ethernet0
 encapsulation ppp
 shutdown
 async dynamic address
 async dynamic routing
 async mode dedicated
 no cdp enable
 ppp authentication pap local
 no tarp propagate
!
router eigrp 109
 network 172.17.0.0
 no auto-summary
!
ip default-gateway 172.30.55.64
ip domain-name cisco.com
ip name-server 192.168.0.0
ip classless
!
!
line con 0
 exec-timeout 0 0
 login authentication console
line 1 16
 transport input all
line aux 0
 transport input all
line vty 0 4
 password sMudgKin
!
ntp clock-period 17179703
ntp peer 172.19.10.0
ntp peer 172.19.0.0
end
```

The following example shows how to enable user authentication on the device via the Kerberos database. To enable user authentication via the Kerberos database, you would perform the following tasks:

- Entering configuration mode
- Defining the Kerberos local realm
- Identifying the machine hosting the KDC
- Enabling credentials forwarding
- Specifying Kerberos as the method of authentication for login
- Exiting configuration mode (CTL-Z)
- Writing the new configuration to the terminal

```
chet-2500# configure term
Enter configuration commands, one per line. End with CNTL/Z.
chet-2500(config)# kerberos local-realm CISCO.COM
chet-2500(config)# kerberos server CISCO.COM chet-ss20
Translating "chet-ss20"...domain server (192.168.0.0) [OK]
chet-2500(config)# kerberos credentials forward
```

```
chet-2500(config)# aaa authentication login default krb5
chet-2500(config)#
chet-2500#
%SYS-5-CONFIG_I: Configured from console by console
chet-2500# write term
```

Compare the following configuration with the previous one. In particular, look at the lines beginning with the words “aaa,” “username,” and “kerberos” (lines 10 through 20) in this new configuration.

```
Building configuration...
Current configuration:
!
! Last configuration change at 14:05:54 PDT Mon May 13 1996
!
version 11.2
service udp-small-servers
service tcp-small-servers
!
hostname chet-2500
!
clock timezone PST -8
clock summer-time PDT recurring
aaa new-model
aaa authentication login default krb5
aaa authentication login console none
aaa authentication ppp local local
enable password sMudgKin
!
username chet-2500 password 7 sMudgkin
username chet-3000 password 7 sMudgkin
username chetin password 7 sMudgkin
kerberos local-realm CISCO.COM
kerberos server CISCO.COM 172.71.54.14
kerberos credentials forward
!
interface Ethernet0
 ip address 172.16.0.0 255.255.255.0
!
interface Serial0
 no ip address
 shutdown
 no fair-queue
!
interface Serial1
 no ip address
 shutdown
 no fair-queue
!
interface Async2
 ip unnumbered Ethernet0
 encapsulation ppp
 shutdown
 async dynamic routing
 async mode dedicated
 no cdp enable
 ppp authentication pap local
 no tarp propagate
!
interface Async3
 ip unnumbered Ethernet0
 encapsulation ppp
 shutdown
 async dynamic address
```

```
 async dynamic routing
 async mode dedicated
 no cdp enable
 ppp authentication pap local
 no tarp propagate
!
router eigrp 109
 network 172.17.0.0
 no auto-summary
!
ip default-gateway 172.30.55.64
ip domain-name cisco.com
ip name-server 192.168.0.0
ip classless
!
!
line con 0
 exec-timeout 0 0
 login authentication console
line 1 16
 transport input all
line aux 0
 transport input all
line vty 0 4
 password sMudgKin
!
ntp clock-period 17179703
ntp peer 172.19.10.0
ntp peer 172.19.0.0
end
```

With the device configured thus far, user chet can log in to the device with a username and password and automatically obtain a TGT, as illustrated in the next example. With possession of a credential, user chet successfully authenticates to host chet-ss20 without entering a username/password.

```
chet-ss20% telnet chet-2500
Trying 172.16.0.0 ...
Connected to chet-2500.cisco.com.
Escape character is '^]'.
User Access Verification
Username: chet
Password:
chet-2500> show kerberos creds

Default Principal:  chet@CISCO.COM
Valid Starting          Expires                 Service Principal
13-May-1996 14:05:39    13-May-1996 22:06:40    krbtgt/CISCO.COM@CISCO.COM
chet-2500> telnet chet-ss20
Trying chet-ss20.cisco.com (172.71.54.14)... Open
Kerberos:       Successfully forwarded credentials
SunOS UNIX (chet-ss20) (pts/7)
Last login: Mon May 13 13:47:35 from chet-ss20.cisco.c
Sun Microsystems Inc.   SunOS 5.4       Generic July 1994
unknown mode: new
chet-ss20%
```

The following example shows how to authenticate to the device using Kerberos credentials. To authenticate using Kerberos credentials, you would perform the following tasks:

- Entering configuration mode
- Remotely copying over the SRVTAB file from the KDC

- Setting authentication at login to use the Kerberos 5 Telnet authentication protocol when using Telnet to connect to the device
- Writing the configuration to the terminal

Note that the new configuration contains a **kerberos srvtab entry** line. This line is created by the **kerberos srvtab remote**command.

```
chet-2500# configure term
Enter configuration commands, one per line.  End with CNTL/Z.
chet-2500(config)# kerberos srvtab remote earth chet/chet-2500.cisco.com-new-srvtab
Translating "earth"...domain server (192.168.0.0) [OK]
Loading chet/chet-2500.cisco.com-new-srvtab from 172.68.1.123 (via Ethernet0): !
[OK - 66/1000 bytes]
chet-2500(config)# aaa authentication login default krb5-telnet krb5
chet-2500(config)#
chet-2500#
%SYS-5-CONFIG_I: Configured from console by console
chet-2500# write term
Building configuration...
Current configuration:
!
! Last configuration change at 14:08:32 PDT Mon May 13 1996
!
version 11.2
service udp-small-servers
service tcp-small-servers
!
hostname chet-2500
!
clock timezone PST -8
clock summer-time PDT recurring
aaa new-model
aaa authentication login default krb5-telnet krb5
aaa authentication login console none
aaa authentication ppp local local
enable password sMudgKin
!
username chet-2500 password 7 sMudgkin
username chet-3000 password 7 sMudgkin
username chetin password 7 sMudgkin
kerberos local-realm CISCO.COM
kerberos srvtab entry host/chet-2500.cisco.com@CISCO.COM 0 832015393 1 1 8 7 sMudgkin
kerberos server CISCO.COM 172.71.54.14
kerberos credentials forward
!
interface Ethernet0
 ip address 172.16.0.0 255.255.255.0
!
interface Serial0
 no ip address
 shutdown
 no fair-queue
!
interface Serial1
 no ip address
 shutdown
 no fair-queue
!
interface Async2
 ip unnumbered Ethernet0
 encapsulation ppp
```

```
 shutdown
 async dynamic routing
 async mode dedicated
 no cdp enable
 ppp authentication pap local
 no tarp propagate
!
interface Async3
 ip unnumbered Ethernet0
 encapsulation ppp
 shutdown
 async dynamic address
 async dynamic routing
 async mode dedicated
 no cdp enable
 ppp authentication pap local
 no tarp propagate
!
router eigrp 109
 network 172.17.0.0
 no auto-summary
!
ip default-gateway 172.30.55.64
ip domain-name cisco.com
ip name-server 192.168.0.0
ip classless
!
!
line con 0
 exec-timeout 0 0
 login authentication console
line 1 16
 transport input all
line aux 0
 transport input all
line vty 0 4
 password sMudgKin
!
ntp clock-period 17179703
ntp peer 172.19.10.0
ntp peer 172.19.0.0
end
chet-2500#
```

With this configuration, the user can Telnet in to the device using Kerberos credentials, as illustrated in the next example:

```
chet-ss20% bin/telnet -a -F chet-2500
Trying 172.16.0.0...
Connected to chet-2500.cisco.com.
Escape character is '^]'.
[ Kerberos V5 accepts you as "chet@CISCO.COM" ]
User Access Verification
chet-2500>[ Kerberos V5 accepted forwarded credentials ]
chet-2500> show kerberos creds
Default Principal:  chet@CISCO.COM
Valid Starting          Expires                 Service Principal
13-May-1996 15:06:25    14-May-1996 00:08:29    krbtgt/CISCO.COM@CISCO.COM
chet-2500>q
Connection closed by foreign host.
chet-ss20%
```

The following example shows how to map Kerberos instances to Cisco's privilege levels. To map Kerberos instances to privilege levels, you would perform the following tasks:

- Entering configuration mode
- Mapping the Kerberos instance admin to privilege level 15
- Mapping the Kerberos instance restricted to privilege level 3
- Specifying that the instance defined by the **kerberos instance map** command be used for AAA Authorization
- Writing the configuration to the terminal

```
chet-2500# configure term
Enter configuration commands, one per line.  End with CNTL/Z.
chet-2500(config)# kerberos instance map admin 15
chet-2500(config)# kerberos instance map restricted 3
chet-2500(config)# aaa authorization exec default krb5-instance
chet-2500(config)#
chet-2500#
%SYS-5-CONFIG_I: Configured from console by console
chet-2500# write term
Building configuration...
Current configuration:
!
! Last configuration change at 14:59:05 PDT Mon May 13 1996
!
version 11.2
service udp-small-servers
service tcp-small-servers
!
hostname chet-2500
!
aaa new-model
aaa authentication login default krb5-telnet krb5
aaa authentication login console none
aaa authentication ppp default krb5 local
aaa authorization exec default krb5-instance
enable password sMudgKin
!
username chet-2500 password 7 sMudgkin
username chet-3000 password 7 sMudgkin
username chetin password 7 sMudgkin
ip domain-name cisco.com
ip name-server 192.168.0.0
kerberos local-realm CISCO.COM
kerberos srvtab entry host/chet-2500.cisco.com@CISCO.COM 0 832015393 1 1 8 7 sMudgkin
kerberos server CISCO.COM 172.71.54.14
kerberos instance map admin 15
kerberos instance map restricted 3
kerberos credentials forward
clock timezone PST -8
clock summer-time PDT recurring
!
interface Ethernet0
 ip address 172.16.0.0 255.255.255.0
!
interface Serial0
 no ip address
 shutdown
 no fair-queue
```

```
!
interface Serial1
 no ip address
 shutdown
 no fair-queue
!
interface Async2
 ip unnumbered Ethernet0
 encapsulation ppp
 shutdown
 async dynamic routing
 async mode dedicated
 no cdp enable
 ppp authentication pap local
 no tarp propagate
!
interface Async3
 ip unnumbered Ethernet0
 encapsulation ppp
 shutdown
 async dynamic address
 async dynamic routing
 async mode dedicated
 no cdp enable
 ppp authentication pap local
 no tarp propagate
!
router eigrp 109
 network 172.17.0.0
no auto-summary
!
ip default-gateway 172.30.55.64
ip classless
!
!
line con 0
 exec-timeout 0 0
 login authentication console
line 1 16
 transport input all
line aux 0
 transport input all
line vty 0 4
 password sMudgKin
!
ntp clock-period 17179703
ntp peer 172.19.10.0
ntp peer 172.19.0.0
end
chet-2500#
```

The following example shows output from the three types of sessions now possible for user chet with Kerberos instances turned on:

```
chet-ss20% telnet chet-2500
Trying 172.16.0.0 ...
Connected to chet-2500.cisco.com.
Escape character is '^]'.
User Access Verification
Username: chet
Password:
chet-2500> show kerberos creds
Default Principal:  chet@CISCO.COM
```

```
Valid Starting          Expires                 Service Principal
13-May-1996 14:58:28    13-May-1996 22:59:29    krbtgt/CISCO.COM@CISCO.COM
chet-2500> show privilege
Current privilege level is 1
chet-2500> q
Connection closed by foreign host.
chet-ss20% telnet chet-2500
Trying 172.16.0.0 ...
Connected to chet-2500.cisco.com.
Escape character is '^]'.
User Access Verification
Username: chet/admin
Password:
chet-2500# show kerberos creds
Default Principal:  chet/admin@CISCO.COM
Valid Starting          Expires                 Service Principal
13-May-1996 14:59:44    13-May-1996 23:00:45    krbtgt/CISCO.COM@CISCO.COM
chet-2500# show privilege
Current privilege level is 15
chet-2500# q
Connection closed by foreign host.
chet-ss20% telnet chet-2500
Trying 172.16.0.0 ...
Connected to chet-2500.cisco.com.
Escape character is '^]'.
User Access Verification
Username: chet/restricted
Password:
chet-2500# show kerberos creds
Default Principal:  chet/restricted@CISCO.COM
Valid Starting          Expires                 Service Principal
13-May-1996 15:00:32    13-May-1996 23:01:33    krbtgt/CISCO.COM@CISCO.COM
chet-2500# show privilege
Current privilege level is 3
chet-2500# q
Connection closed by foreign host.
chet-ss20%
```

Example: Encrypting a Telnet Session

The following example shows how to establish an encrypted Telnet session from a device to a remote host named “host1”:

```
Device>
 telnet host1 /encrypt kerberos
```

Additional References

Related Documents

Related Topic	Document Title
Kerberos Commands	*Cisco IOS Security Command Reference*

Error Message Decoder

Description	Link
To help you research and resolve system error messages in this release, use the Error Message Decoder tool.	https://www.cisco.com/cgi-bin/Support/Errordecoder/index.cgi

MIBs

MIB	MIBs Link
All the supported MIBs for this release.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/support

Feature Information for Kerberos

Release	Feature Information
Cisco IOS Release 15.0(2)EX	This feature was introduced.