

Building a Scalable Network Device Management Framework with the Cisco Secure ACS TACACS+ (RBAC) Server

Abstract

Cisco Secure ACS 3.0 Shell Authorization Command sets provide the facilities to enable the construction of a scalable network device management system using familiar and efficient TCP/IP protocols and utilities supported by Cisco devices. This document discusses the key benefits of the technology and how to deploy it.

Introduction

The ongoing explosion in the many different types of IP data being communicated, along with the perennial increase in the sheer volume of data, have been facilitated by a commensurate growth in the supporting network infrastructure—routers, switches, firewalls, virtual private network (VPN) concentrators, and so on. Large-scale network infrastructures now comprise thousands of network devices, indeed in exceptional cases tens of thousands of devices. The rigorous availability requirements, driven by the increasing mission criticality of the services being provided over the network, are compounding the challenges posed by the number of devices. Increasingly, loss of connectivity is simply unacceptable.

Predictably, the requirement to adequately manage the network has resulted in the emergence of a plethora of network management protocols, tools, and technologies. Indeed, one might expect the traditional approaches to device administration—such as command-line configuration by Telnet—to be in decline or even to have disappeared altogether. In practice, despite the extensive capabilities provided by new network management tools, command-line device administration by Telnet remains a favored approach because of the familiarity, speed, power, and convenience it affords. Despite advances in other methods of network device management, Telnet-based administration is certain to remain a favored method in the future.

As the number of network devices in a typical network has grown, the number of administrators required to keep the network operating has, likewise, increased. These administrators are inevitably spread across the organization, and they tend to be employed by different departments—the larger and more complex the network and organization, the more complex the resulting system administration structure. Network administration departments are now beginning to understand that without a mechanism to establish an overall network administration system that controls which administrators can perform which commands upon which devices, problems with the security and reliability of the network infrastructure are unavoidable.

Device administration sets facilitate the establishment of a management system to control granular command authorization for different grades of administrators on specified groups of devices.



Requesting Command Authorization

For simplicity, this document concentrates on Cisco network devices running Cisco IOS® Software, and all the illustrations are based on devices using this software. Nevertheless, the solutions discussed in this document also apply—either wholly or in part—to a variety of other Cisco network devices not necessarily running Cisco IOS Software. These include:

- Cisco Catalyst® switches (running the Cisco Catalyst Operating System [CatOS])
- Cisco PIX® firewalls
- Cisco VPN 3000 Series concentrators

Cisco devices not running Cisco IOS Software (for example, Cisco Catalyst switches running CatOS, Cisco PIX firewalls running the Cisco PIX Operating System, or Cisco VPN 3000 concentrators) may also support enable privileges, TACACS+ (authentication, authorization, and accounting [AAA]) command authorization, or both. As the need for centralized management control and auditing grows, the authors expect that other Cisco network devices that presently do not support TACACS+ command authorization will be enhanced to do so. For an up-to-date understanding of the support levels of Cisco devices, consult the latest documentation for the device under consideration.

Cisco devices running Cisco IOS Software offer two solutions to network device management:

- Enable privileges
- AAA command authorization

Cisco IOS Software has 16 privilege levels, 0 to 15 (other Cisco devices may support a smaller number of privilege levels; for example, the Cisco VPN 3000 Concentrator supports two). By default, upon first connection to a device command line, a user's privilege level is set to 1. To change the default privilege level, you must run the enable command and provide both the user's enable password and the new privilege level being requested. If the password is correct, the new privilege level is granted. Note that the commands that may be executed for each privilege level on that device are stored locally in that device configuration. The default privilege levels for Cisco IOS devices when supplied by Cisco are shown in Table 1.

Table 1 Default Levels

| Privilege Level | Description |
|-----------------|--|
| 0 | Includes the disable , enable , exit , help , and logout commands |
| 1 | Includes all <i>user</i> -level commands at the router> prompt |
| 15 | Includes all <i>enable</i> -level commands at the router# prompt |



You can modify these levels and define new levels, as shown in Figure 1.

Figure 1 Example of enable Command Privilege Levels

```
enable password level 10 pswd10
privilege exec level 10 clear line
privilege exec level 10 debug ppp chap
privilege exec level 10 debug ppp error
privilege exec level 10 debug ppp negotiation
```

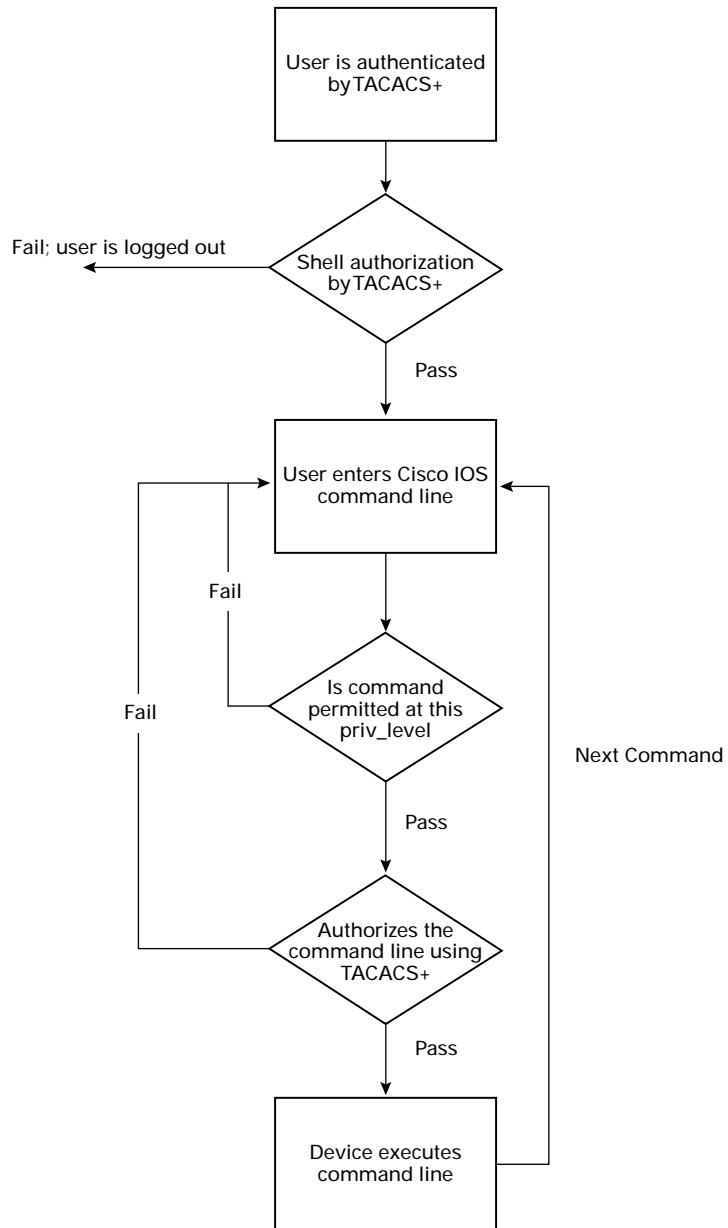
Having static local passwords associated with privilege levels on each device has a major inherent drawback: each user's enable password has to be configured on every device on which that user requires access.

To reduce the administrative scalability problem this situation poses, TACACS+ can provide the control of privilege-level authorization from a centralized location. TACACS+ servers generally allow individual users to have their own enable passwords and to obtain certain privilege levels. Thus users may be disabled or their privilege levels changed from a single, central location without affecting other administrators.

Another major problem arises because the privilege-level commands need to be properly configured on every device in the network so that administrators have a consistent experience across the devices they administer. Unfortunately, centralization of enable privilege-level control using TACACS+ does nothing to ameliorate this massive administrative scalability challenge. To alleviate this problem, you can also locate command authorization in the TACACS+ server. With this setup, any command typed on the device is first checked against the current privilege level and, if that check is satisfied, it is then issued to the TACACS+ server for a further check. Figure 2 shows the logic employed by the device to determine whether the user is authorized to execute a command line.



Figure 2 Device Logic, User is authenticated by TACACS+; Fail; user is logged out; Shell authorization...; User enters Cisco IOS command line; Is command permitted; Next Command; Authorizes the command line; Device executes command line]



TACACS+ provides the ability to predefine the initial privilege level a user receives when logging on to a device, as part of the general T+ shell service authorization (priv-lvl=). This allows an administrator to instantly gain privilege level 15 when connecting to the device.



Granting Command Authorization

In the past, the TACACS+ server stored command authorization directly against the user or group. This command authorization simply comprised a list of permitted or denied command lines, which were matched against the commands to be executed on the device. Unfortunately, a single “command set” was used globally for all devices, thus preventing device administrators from having different levels of responsibility on different devices via a single TACACS+ server; in other words, administrators could execute the same set of commands on any device to which they could gain access. In a large, complex network with many administrators, the inability to provide differentiated command authorization is a significant shortcoming.

A solution to address this shortcoming must grant a user/group a different level of access, depending on which device the user/group is attempting to administer. In other words, commands should be set according to the device being administered. One way to accomplish this would be to allow multiple command authorization sets on the group or user page for each possible device the engineer could administer. Although this setup would work, the result would probably be the same command sets defined for each user or group—simply tied to different devices. A better solution allows the command sets to be defined independent of users and groups. Then the user/groups simply reference the appropriate command sets, depending on the device to be administered. In this model, a command set equates to the network administration role, with the role being shareable across different groups or users and different “roles” being applicable for a single user, depending on which device is being administered.

Shell Authorization Command Sets

As described previously, shell authorization command sets enable the sharing of command authorization; that is, sets of commands across different users and groups. As illustrated in Figure 3, the Cisco Secure ACS graphical user interface (GUI) facilitates the independent definition of a command authorization set.

Figure 3 Shell Command Authorization Set GUI

Shell Command Authorization Set

Name:

Description:

Unmatched Commands: Permit Deny

Permit Unmatched Args



Command sets are given a name; this name is then used to reference the command set from either the user or group settings. These references can be either a straight mapping for all devices that the user may access (as per the older global control model) or a mapping based on the device issuing the requests. Following is a more detailed description of what each field represents and how it may be used.

Key Benefits

Per-Device Group Authorization

The key benefit to using shell command authorization sets is the ability to associate a user with a different command set, depending on which device, or group of devices, is being accessed.

An example of such a mapping is shown in Figure 4.

Figure 4 An Example of Mapping Command Sets

| Device Group | Command Set |
|--------------|-----------------------|
| West Coast | Local Network Enginee |
| East Coast | Network Engineer |
| Mid West | Network Engineer |

Remove Association

Device Group:


Command Set:

Add Association

Role-Based Authorization

As noted previously, a command set may be understood as a role definition. Essentially it defines the commands that are authorized and thus the kinds of tasks that may be undertaken. If command sets are defined around common administrative or organizational roles, users and groups can share them. When combined with per-network-device-group authorization, users can assume different roles for different groups of devices. An example is shown in Figure 5.

Figure 5 An Example of Role-Based Shell Command Authorization Sets

| Shell Command Authorization Sets  | |
|--|--|
| Name | Description |
| Help desk | Limited fault finding |
| Local Network Engineer | Full access to device configuration |
| Network Engineer | Read only access to device configuration |



The emerging industry nomenclature that is gaining wider acceptance for a server that supports generic role-based authorization is a role-based access control (RBAC) server. The support for device administration sets over TACACS+ by Cisco Secure ACS establishes it as a domain-specific RBAC server, providing a solution to the network device management command authorization problem.

Scalability

A central location for command authorization is only the first step toward a scalable command authorization policy. Shell command authorization sets offer scalability through sharing common command sets (roles) with many users and groups.

By associating command sets with device groups, as opposed to individual devices, the introduction of a new device into the network entails simply placing it into the correct network device group (NDG). All users and groups then automatically derive the correct level of access to this new device without further work by the network administrators. There is no need to visit each user or group and define the appropriate access.

For example, Fred is a West Coast network engineer with the network engineer role for all devices on the West Coast. For East Coast devices he has only a help desk role that allows him to perform some fault diagnostics. Adding a new device to the West Coast device group means that Fred automatically gets the network engineer role for that device, while maintaining his help desk access to East Coast devices.

Granularity of Individual Command Authorization

Command authorization extends past the command portion of the command line; all the arguments for the command can be considered when deciding whether to authorize a command line. The level of granularity of the authorization to be imposed is up to the AAA administrator who defines the command set. In some cases simply restricting what commands are allowed is sufficient.

Consider the case of defining a simple set to be used for help desk personnel, whose tasks involve checking that a device is alive and executing some simple IP diagnostic tools to check connectivity (for example, ping, traceroute).

Figure 6 Example of Granular Authorization

The screenshot shows a configuration window for command authorization. On the left, under 'Unmatched Commands:', there is a list box containing the following commands: 'debug', 'interface', 'show', 'terminal', and 'traceroute'. The 'traceroute' command is currently selected. To the right of this list, there are two radio buttons: 'Permit' (which is unselected) and 'Deny' (which is selected). Below the radio buttons, there is a checked checkbox labeled 'Permit Unmatched Args', which is circled in red. At the bottom of the window, there are two buttons: 'Add Command' and 'Remove Command'.

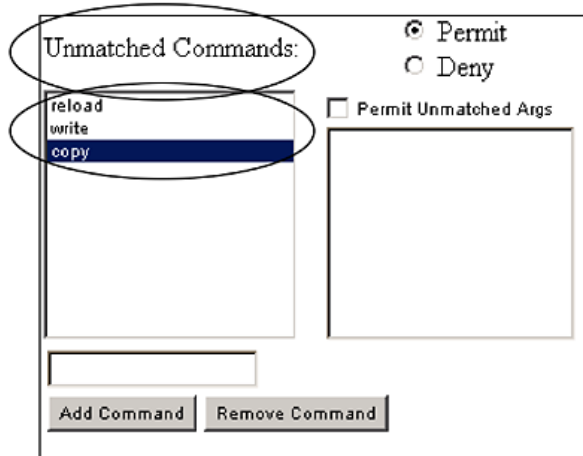
Note: When simply defining authorization at just the command level, the *Permit Unmatched Args* check box must be selected for each command for the commands to be authorized.



When only a few commands must be denied, it may be easier to proceed as follows:

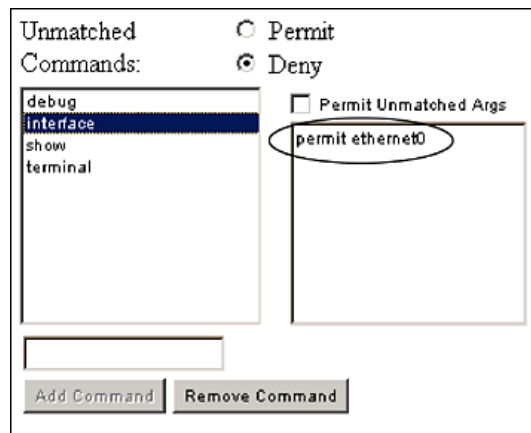
1. Set Unmatched Commands to Permit.
1. Add the commands to deny, ensuring that the Permit Unmatched Args is not checked for these commands.

Figure 7 Example of Specific Command Denial



Constraining the use of *ping* and *traceroute* to specific hosts is probably more restrictive than is necessary; however, restricting the interfaces an administrator can access while in configuration mode might be very important. Figure 8 shows an example.

Figure 8 Example of Command Argument Limitation



A user mapped to this command set is allowed to issue the following command line:

```
interface ethernet0
```

but is not allowed to issue the following command line:

```
interface serial0
```




Command-Line Matching

The TACACS+ server uses a UNIX *grep* style, regular expression, pattern-matching algorithm to match the command line sent by the device and the command line defined in the command authorization set. This pattern-matching approach offers a great deal of flexibility because any commands may be matched without changes required to the functionality of Cisco Secure ACS. It has, however, two possible drawbacks:

- You must have a good understanding of the pattern-matching rules (refer to Appendix A for examples).
- No validity checking is performed by Cisco Secure ACS, so the burden for typing the commands correctly is entirely on the Cisco Secure ACS system administrator—if the administrator types the command wrong it will never be executed (if the filter being defined is a permit filter).

Despite these drawbacks, the power and flexibility make this a risk worth taking, particularly because any individual organization is not likely to have hundreds or even tens of command sets (roles) that need to be defined. In addition, these command sets can be carefully checked for correct operation in a test environment before being deployed in production.

Audit Trail

Having a central point for authorization enables a central point for logging all administration activity. This includes both the commands that authorized successfully and those that failed to authorize.

Three reports are used to track a user’s entire administration session.

- *The TACACS+ Accounting Report* records the start and end of an administration session; accounting must be enabled on the AAA client.
- *The TACACS+ Administration Report* records all the successfully authorized commands issued on a device; accounting must be enabled on the AAA client.
- *The Failed Attempts Report* records all failed attempts to log in to a device and all failed command authorizations on a device; accounting for this report is enabled on the AAA client by default.

Figure 9 Audit Example—Logging On

```

LA-Gateway

Login: andy
Password: *****

LA-Gateway>

```

When a user begins a new administration session on a device, it is recorded in the TACACS+ Accounting Report. An entry also is created when the administration session terminates. The Acct-Flags field differentiates between the two events.

Figure 10 Audit Example—Accounting Report Excerpt [CALL OUT IN TEXT]

| Date ↓ | Time | User-Name | Group-Name | Caller-Id | Acct-Flags | Access Device | Network Device Group | elapsed time | service | task id |
|------------|----------|-----------|---------------|-----------|------------|---------------|----------------------|--------------|---------|---------|
| 11/14/2001 | 15:51:35 | andy | Default Group | 01454 | start | LA-Gateway | West Coast | .. | shell | .. |
| | | .. | Default | | .. | .. | .. | .. | .. | .. |



When the user gains access to the device, all successfully executed commands are sent to the TACACS+ server as TACACS+ accounting requests. The TACACS+ server then logs these accounting requests in the TACACS+ Administration Report. Figure 11 shows an example administration session.

Figure 11 Audit Example—Accounting Requests

```
LA-Gateway> enable
Password: *****

LA-Gateway#config terminal
Enter configuration commands, one per line. End with CNTL/Z.
LA-Gateway(config)#interface bri0
LA-Gateway(config-if)##ppp authentication chap
```

The extract from the TACACS+ Administration Report shown in Figure 12 lists all the commands a user successfully performed on a given device, in chronological order.

Figure 12 Audit Example—Administration Report Excerpt

| Date ↑ | Time | User-Name | Group-Name | cmd | cmd-arg | priv-lvl | service | Access Device | Network Device Group |
|------------|----------|-----------|---------------|-----------|---------------------|----------|---------|---------------|----------------------|
| 11/14/2001 | 15:54:36 | andy | Default Group | enable | | | shell | LA-Gateway | West Coast |
| 11/14/2001 | 15:54:53 | andy | Default Group | configure | terminal | 15 | shell | LA-Gateway | West Coast |
| 11/14/2001 | 15:55:07 | andy | Default Group | interface | bri0 | 15 | shell | LA-Gateway | West Coast |
| 11/14/2001 | 16:02:53 | andy | Default Group | ppp | authentication chap | 15 | shell | LA-Gateway | West Coast |

Note: This report contains only commands successfully authorized and executed. It does not contain command lines that contain typos or commands that were not authorized.

In the example shown in Figure 13, a user began by executing some authorized commands, and then attempted to execute a command for which that user did not have authorization (configure terminal).

Figure 13 Audit Example—Unauthorized Request

```
NY-Gateway

Login: andy
Password: *****

NY-Gateway>
NY-Gateway> enable
Password: *****

NY-Gateway#show run

. . .

NY-Gateway#configure terminal
```



Figure 14 gives an extract from the *TACACS+ Administration Report*, detailing the commands that user *andy* executed on the NY-Gateway machine.

Figure 14 Audit Example—Administration Report Excerpt

| Date ↑ | Time | User-Name | Group-Name | cmd | cmd-arg | priv-lvl | service | Access Device | Network Device Group |
|------------|----------|-----------|---------------|--------|----------------|----------|---------|---------------|----------------------|
| 11/14/2001 | 15:54:36 | andy | Default Group | enable | . | 1 | shell | NY-Gateway | East Coast |
| 11/14/2001 | 15:54:53 | andy | Default Group | show | running-config | 15 | shell | NY-Gateway | East Coast |

When *andy* attempted to change the configuration of the NY-Gateway machine, the TACACS+ server did not grant authorization, and the attempt was recorded in the Failed Attempts Report (Figure 15).

Figure 15 Audit Example—Failed Attempts Report Excerpt

| Date ↓ | Time | Message-Type | User-Name | Group-Name | Access Device | Network Device Group | Device Command Set | Authen-Failure-Code | Authen-Failure-Code | Author-Data |
|------------|----------|---------------|-----------|---------------|---------------|----------------------|--------------------|---------------------|---------------------|--|
| 11/14/2001 | 17:18:45 | Author failed | andy | Default Group | NY-Gateway | East Coast | Network Engineer | . | Command unknown | service=shell cmd=configure terminal |

Tip: If the *Network Device Group* and *Device Command Set* columns are included in the *Failed Attempts Report*, you can easily determine why user *andy* was denied the use of the configure command.

Combined, these three reports provide a complete record of the administrative activities that have been attempted and authorized.



Sample Deployment

Problem

The ACME organization has its network infrastructure managed by three teams, the LAN team, the VoIP team, and the WAN team. The primary role of each team is to administer its own devices. There are times, however, when a team needs to examine the configuration of another team's device. Further, it is essential that when accessing another team's devices the level of access is read-only.

The organization's directors have made the following mandatory requirements:

- All devices must use a single AAA infrastructure.
- For audit purposes, all commands issued on a device must be logged.

Solution

To solve this problem, use the following features in Cisco Secure ACS v3.0:

- Network Device Groups
- Shell Authorization Command Sets
- CSV Reports

Segmenting the Network

Because you want to base your authorization decisions partly on who manages a given device, you need to organize the devices into management groups. To do this, create three NDGs called

- LAN
- VoIP
- WAN

As you add each network device to the AAA server, place it into the appropriate NDG based on which group has responsibility for managing the device.

Creating Roles

When accessing a device, users will be accessing their own equipment or another team's equipment, so you need to create two command sets, one allowing complete access to the device and another allowing only restricted read-only access to the device. These command sets are called:

- Local Network Engineer
- Network Engineer

The Local Network Engineer set allows all commands, whereas the Network Engineer set allows only a restricted set of commands that allow read-only access to the configuration and a set of diagnostic commands (refer to Figure 16).



Figure 16 Network Engineer Roles

The screenshot shows two configuration panels for network roles. The left panel is for 'Network Engineer' with a description of 'Read only access to device configuration'. It has 'Unmatched Commands' set to 'Deny' and a list of commands including 'debug', 'interface', 'show', 'terminal', and 'traceroute'. The right panel is for 'Local Network Engi' with a description of 'Full access to device configuration'. It has 'Unmatched Commands' set to 'Permit' and an empty list of commands. Both panels include 'Add Command' and 'Remove Command' buttons.

Associating Users with Roles

Now create three new user groups by renaming three unused groups.

- LAN Engineering
- VoIP Engineering
- WAN Engineering

For each of these groups, enable the TACACS+ service shell and define the role the group's users should have when administering the various devices in the network.

Figure 17 shows how this is configured for users in the LAN Engineering group.

Figure 17 Network Roles Applied

The screenshot shows the 'Shell Command Authorization Set' configuration. The 'Assign a Shell Command Authorization Set on a per Network Device Group Basis' option is selected. A table lists the associations:

| Device Group | Command Set |
|--------------|------------------------|
| LAN | Local Network Engineer |
| VoIP | Network Engineer |
| WAN | Network Engineer |

Below the table, there are fields for 'Device Group' (set to WAN) and 'Command Set' (set to Network Engineer), with 'Add Association' and 'Remove Association' buttons.



When adding new engineers to Cisco Secure ACS, place them into one of the three groups. This procedure immediately gives them the correct level of access to the devices under their group's control.

Reports

In order to have full visibility of the command-line activity that has occurred—or has been attempted—ensure that the following CSV log targets are enabled:

- *Failed Attempts Report*
- *TACACS+ Administration Report*

Cisco IOS Software Configuration

The Cisco IOS Software configuration shown in Figure 18 is valid for Cisco IOS versions prior to 12.0(5)T. (From 12.0(5)T and later, the syntax changed, and you should refer to the applicable documents that can be found on the Cisco.com Web site.) The configuration represents the AAA portion of the configuration to implement centralized command administration. It assumes that the three default privilege levels are the only privilege levels defined on the device. This configuration needs to be applied to all devices, but it is done only once per device.

Figure 18 Cisco IOS Software Configuration

```
aaa new-model
aaa authentication login default tacacs+ local
aaa authentication enable default tacacs+ none
aaa authorization exec default tacacs+ none
aaa authorization commands 1 default tacacs+ none
aaa authorization commands 15 default tacacs+ none

aaa accounting commands 1 stop-only tacacs+ none
aaa accounting commands 15 stop-only tacacs+ none
tacacs-server host <your_tacacs+_server_ip>
tacacs-server key <your_secret_key>
```

Additional Solution Benefits

This solution not only satisfies the problem definition, but also provides a scalable solution. Adding new engineers or devices simply requires placing them into their appropriate groups; engineers then automatically assume the appropriate roles for the devices they are accessing and, most importantly, devices are protected from unauthorized access.

Console Port Authorization

Until now, command authorization has been enabled only for connections originating from network interfaces. Console port authorization was not added as a feature until Bug ID CSCdi82030 was resolved. Console port authorization is turned off by default to lessen the likelihood of accidentally being locked out of the router. To ensure that all authorization is obtained and recorded via a single, central location, console authorization must be enabled. This authorization is especially useful if the console port is attached via a serial port concentrator/Ethernet multiplexer. This functionality can be enabled only for images in which Bug ID CSCdi82030 has been implemented; console port authorization can be turned on under line `con 0` with the hidden command **aaa authorization console**.



Issues and Solutions in More Complex Deployment Scenarios

Heterogeneous Device Type Environments

TACACS+ command authorization is supported on three Cisco device types: Cisco IOS Software, CatOS, and the Cisco PIX Operating System. In networks where all three types are being managed, typically they must all be managed from the same AAA infrastructure. Although this setup adds some complexities, they may be overcome with appropriate implementation.

For historical reasons, both Cisco IOS Software and CatOS devices use the same TACACS+ service type, `service=shell`, when making a command authorization request against a TACACS+ server. Although this setup complicates the task slightly for Cisco Secure ACS administrators, who service requests from both types of devices, it can be resolved in two ways:

- *Construct separate NDGs for each device type*—In this scenario, the NDG network model allows for the separation of device types into discrete groups. This solution is the simplest one because device-type-specific command sets can be constructed for each device type and applied separately, as required.
- *Construct a single command set to deal with both types of devices*—In some network models it is impractical to separate device types into discrete NDGs. In such cases you can build a command authorization set that deals with the possible commands to be authorized that might be generated by devices running either operating system. As noted previously, because the Cisco Secure ACS GUI treats all command authorizations as text-matching operations, there is nothing implicit to prohibit this approach.

For the Cisco PIX Firewall, the administrator can configure the TACACS+ service used—the service can be set to either `service=shell`, as per Cisco IOS Software, or `service=pixshell` (from Version 6.3 onward only; Versions 6.2 and below use `service=shell` only). If the service is set to `service=pixshell` (recommended), Cisco Secure ACS can easily differentiate requests originating from a Cisco PIX Firewall from those originating from Cisco IOS Software or CatOS. This simplifies the authorization process, because with the extra information included in the request the Cisco Secure ACS can easily choose the appropriate command set to authorize it. When using `service=pixshell`, the administrator can develop a separate group of command sets in Cisco Secure ACS that can be applied to incoming requests. A Cisco PIX command set could be related in the Cisco Secure ACS GUI to a NDG in addition to a standard Cisco IOS Software command set (normally only a single command set may be mapped for a particular group/user against a particular NDG). Note that if the Cisco Secure ACS receives a `service=pixshell` command authorization request for an NDG for which no Cisco PIX command set is mapped, it will refuse the authorization—even if a matching command is contained within a Cisco IOS Software command set that is mapped for that NDG.

As support for command authorization for new device types is added to Cisco Secure ACS, each device carries its own unique service definition. This scenario facilitates operation in the same mode as for the Cisco PIX Firewall; that is, each device type has a dedicated command set that can be applied to each NDG.



Network Access Restriction Filtering

As described extensively throughout this paper, Command Authorization Sets control the commands that a device administrator may execute when logged in to a particular device with Telnet/Secure Shell Protocol (SSH). They do not, however, control initial access to a device. Control of access to a device is provided in Cisco Secure ACS through two mechanisms:

- *Authentication*—First, prospective administrators must satisfy the AAA server that they have the appropriate credentials.
- *Network Access Restriction (NAR) filtering*—When users successfully pass the authentication step, Cisco Secure ACS (if it is so configured) checks them to determine whether they are authorized to have a device administration (Telnet/SSH) session on that device.

Only after a user successfully completes both steps does the Cisco Secure ACS grant a Telnet/SSH session to that user on a device. Like device administration sets (DASs), NAR filtering may be performed at the individual device level or may be done at the NDG level. Again, like DAS, NAR filtering may be applied to individual users or to groups of users. A useful alternative way of considering NAR filtering, therefore, is as a complete command authorization deny set for that device—if a user cannot log on to a device, that user cannot execute any commands on that device.

Clearly, this is complementary functionality to command authorization sets and may be used along with them to build suitable network administration policies for any particular site.

TACACS+ Change Password

Although not directly related to the use of DASs, this feature of the TACACS+ protocol is important in large-scale managed networks.

Cisco IOS devices support a TACACS+ facility for users to change their passwords when connecting for an administration session. When prompted for their password, users who hit Return with nothing entered are allowed to enter into a *change password* routine. The changes made by the user are communicated to the Cisco Secure ACS over TACACS+ using a routine known as CHPASS.

Although useful in device administration systems where user-changeable passwords are a requirement, this facility can be somewhat problematic for some redundant AAA server configurations. Many AAA servers, including Cisco Secure ACS (up to and including v3.1), use a discrete database instance per server. In these installations, if there is no mechanism for communicating the password change to upstream replication partners, the password changes can be lost when the next replication occurs. Unfortunately, in Cisco Secure ACS v3.0 and below, this problem has no workaround, so it is best for device administrators not to use the CHPASS facility. An alternative method for allowing users to change their passwords (in a way that is not lost) is to use the Web-based User Changeable Password utility supplied with Cisco Secure ACS. If this utility is configured to “point at” the top-level replication master in the Cisco Secure ACS replication tree, changes made using it are replicated to downstream partners and are not lost. Unfortunately, replication propagation timing issues may still be present. A Cisco Secure ACS receiving TACACS+ authentication traffic from a particular device may not have yet received the password change update and so may still be operating with the older password. Although this problem is undesirable, it is a well-known problem with many distributed password control systems. Setting the Cisco Secure ACS replication frequency to something very small mitigates this problem to some degree.

In Cisco Secure ACS v3.1, this problem is alleviated more completely. A check box has been added to the Cisco Secure ACS GUI to disable Cisco Secure ACS CHPASS from its TACACS+ clients. When a user on a TACACS+ client device attempts the CHPASS procedure against a Cisco Secure ACS on which support for CHPASS is disabled, Cisco Secure ACS sends a configurable message back to the device and to the user explaining that the CHPASS functionality is not supported on this



device. The message is presented in an editable text box in the Cisco Secure ACS GUI on the same page as the check box controlling CHPASS support. This configurable message can be used to direct device administrators to perform a Telnet to a device that uses a TACACS+ server that allows CHPASS.

On a Cisco Secure ACS on which CHPASS is enabled (CHPASS should be enabled on a top-level replication master for the installation) the CHPASS sequence proceeds as at present. When a user tries to change a password on a Cisco Secure ACS on which CHPASS is enabled, that Cisco Secure ACS performs an immediate and automatic propagation of the event to its replication slave partners (all those configured in its GUI as replication slave partners). This process mitigates any possible change propagation issues inherent in earlier version solutions relying upon timed replication propagation.

Cisco Secure ACS Network Device Database Provisioning

As in the preceding section, provisioning the Cisco Secure ACS network device database is not directly related to the use of DASs, but it also merits coverage because it is an important issue for any large-scale network. In smaller-scale networks, the Cisco Secure ACS device database is populated and maintained using the facilities provided by the Cisco Secure ACS GUI. As the number of devices increases, this method of maintaining the device database becomes increasingly impractical because it consumes increasingly larger amounts of administrator time. Fortunately, Cisco Secure ACS has two programmatic interfaces for managing its network device database:

- *CSDBSync*—This utility provides an Open Database Connectivity (ODBC)-based facility that can be driven programmatically. It provides all the functionality available through the GUI—insert, modify, delete. Full details of the functionality of DBSync are provided as part of the standard Cisco Secure ACS documentation set.
- *CSUtil*—CSUtil offers capabilities similar to those of DBSync but uses a proprietary input file format for database changes. As with DBSync, full details of the functionality of CSUtil are provided as part of the standard Cisco Secure ACS documentation set.

Choice of which of these two facilities to use is determined by the specifics of the implementation and the preference of those charged with administering the Cisco Secure ACS.

MultiNAS

Directly related to the previous point is the Cisco Secure ACS MultiNAS facility. MultiNAS allows multiple physical devices to share a single entry in the Cisco Secure ACS network device database. This is achieved by entering a range of IP addresses rather than a single address into a single device definition. All the other attributes of the device as defined there—for example, the shared secret—are then used when dealing with requests from any device within the specified range. In this way, a relatively small number of database entries may be used to represent a far larger population of physical clients, thus reducing the administrative overhead required to maintain the network device database by an order of magnitude (or even more).

For more information on configuration of MultiNAS, consult the Cisco Secure ACS documentation at Cisco.com.

Performance

As with all transaction processing systems, two basic axes of performance need to be considered: response time and throughput. As background, it is important to note that TACACS+ command authorization tends to put a far lower load on an AAA server than do service-facing AAA workloads such as wireless-network access control or VPN-tunnel authentication and authorization. This is a result of the profile of the workload being almost reversed:

- *Service facing*—Access through a (relatively) modest number of devices is shared by a large number of clients from a very large possible pool of clients, generating a high transaction-per-second rate.



- *Device administration*—Access to a (relatively) large number of devices is shared by a small number of clients from a small possible pool of clients, generating a fairly low transaction-per-second requirement.

Because AAA servers have to deal with the very high stress requirements of the service-facing role, in general they are quite capable of servicing the workloads generated by the authorization control role.

Throughput

Throughput involves two main issues:

- How many concurrent sessions can be processed by a given infrastructure?
- At what level of throughput does the response time for individual transactions being handled become unacceptable to the user?

Individual Server Concurrency

Testing indicates that a single Cisco Secure ACS Server can provide a throughput of approximately 100 TACACS+ requests per second without noticeable degradation of response time for individual transactions. From this fact, extrapolations may be constructed as to what size of administrator population might be supported on a single machine. If an aggressive typing and thinking time of 2 seconds is assumed for each administrator to generate each command authorization request, a single server can service the workload generated by 200 administrators acting concurrently. So what size total pool of administrators should this be extrapolated to?

It is not realistic to design an infrastructure to deal with a maximum performance requirement that will never be experienced in practice. A reasonably aggressive active versus passive concurrency ratio might be one to three—that is, for each active administrator at any point in time, two or more in the database are inactive. On this basis, a single Cisco Secure ACS can support an administrator database of 600 administrators. Using a less aggressive active/passive ratio of, say, one to five, a single server will support a database of 1000 administrators. If more active administrators need to be serviced at any one time, more Cisco Secure ACS servers can be deployed to balance the workload. (Cisco Secure ACS provides the necessary replication tools to build an integrated infrastructure.)

Network Impact on Session Concurrency

Unlike Cisco Secure ACS response-time processing, the network bandwidth between the requesting device and the Cisco Secure ACS may be of consequence for command authorization response time as perceived by the user. If the Cisco Secure ACS and the requesting device are connected over a very low bandwidth link, the intervening network may introduce significant and perceptible latencies to the response times the device administrator experiences. This is likely to be even more aggravated if the intervening networks traverse public networks, perhaps using VPN, where latencies are even more unreliable.

In general, the amount of network traffic generated by TACACS+ device administration is very small. Individual packets are fairly small, containing single commands to be authorized along with a few other small data items. In the unlikely situation in which a large number of devices on one site are connected by a low bandwidth to a remote site where the Cisco Secure ACS is situated, it is possible that the AAA traffic will in itself cause contention problems. More likely, contention with other data traffic will cause the problems.

The solution for this problem is simple: ensure that all requesting devices are connected to their Cisco Secure ACS servers with network connections of sufficient bandwidth. In other words, increase the bandwidth between the remote sites, set up suitable quality of service (QoS) rules, or implement a more local Cisco Secure ACS to the requesting devices.



Response Time

The response time a device administrator experiences for any particular command authorization request has two major components:

- The time it takes to process the authorization by the Cisco Secure ACS
- The time it takes for the request to travel the network to the Cisco Secure ACS and for the response to be sent back

Cisco Secure ACS Request Processing Response Time

The time it takes to process a single command authorization request—if it is the only request being processed—will be affected most significantly by the number of rows in the command set being authorized against. Testing indicates that using a single command authorization against a command set with 50 commands takes about 0.0045 seconds (performed on an ordinary 800-MHz Pentium III host, database of 5000 devices, submitted locally using TACACS+). Clearly, this period of time in human terms is imperceptibly small.

Batch testing confirms this level of performance. A batch job consisting of 10 authorization requests (commands) against a command set of 20 commands for 1900 devices took 33 seconds to complete. This means that the Cisco Secure ACS performed 19,000 individual command authorizations in 33 seconds; the time taken for each request averages out to about 0.0017 seconds ($33/[10*1900]$)—2/100 of a second. This testing was performed on the same 800-MHz host as the individual testing.

In conclusion, after the Cisco Secure ACS receives an individual command authorization request, the time it takes to process it is so infinitesimally small in human perception terms that it may reasonably be discounted from consideration.

Network Latency and Bandwidth

Unlike Cisco Secure ACS response-time processing, the network latency between the requesting device and the Cisco Secure ACS is of much greater consequence for command authorization response time as perceived by the user. If the Cisco Secure ACS is geographically distant from the requesting device, the intervening network may introduce significant and perceptible latencies to the response times the device administrator experiences. This is likely to be even more aggravated if the intervening connections traverse public networks (through VPN tunnels perhaps), where latencies are even more unreliable.

The fact that TACACS+ is a TCP-based protocol further aggravates this situation. Every transaction in both directions generates an extra round trip for TCP session establishment. Thus for a single packet authorization, the network component of the response time is four one-way trips. Because this is known, it is easy to perform empirical testing using “ping” to see what time a packet takes to get from the site with the device client to the Cisco Secure ACS site. When this time is known, simply multiply it by four to get a reasonable approximation of the time required for the network processing of an individual command authorization.

Clearly, this timing depends upon the particular network implementation being considered and is beyond the scope of control of the Cisco Secure ACS. The heuristics for building a TACACS+ device management infrastructure that delivers adequate performance is fairly simple:

1. Test network performance from the envisaged remote sites and confirm that it will not impose a response-time delay that is unacceptable to the device management users.
2. If it does, test alternative sites for Cisco Secure ACS server deployment “nearer” to the sites where performance is impaired.
3. Repeat Steps 1 and 2 until you achieve adequate performance.



Fault Tolerance

With the AAA server controlling command-line access to the network devices, it is more important than ever to have a fault-tolerant AAA environment. At a minimum, AAA servers should be deployed as redundant pairs so that if one fails the second one is available. To do this, have one AAA server as the master and the second the slave and then deploy replication between the master and the slave.

Failover should be defined at TACACS+ device/client, as shown in Figure 19 (for a Cisco IOS Router).

Figure 19 Failover Configuration

```
tacacs-server host <primary ip>
tacacs-server host <secondary ip>
```

If failover is defined in any other manner, it is important to allow AAA functionality to occur *local* to the device in case the network interface providing the transport to the AAA server goes down.

The excerpt from the Cisco IOS Software configuration shown in Figure 20 allows access to the device even if the AAA server becomes unavailable.

Figure 20 Cisco IOS Software Configuration Excerpt

```
username admin privilege 15 password admin
aaa authentication login default tacacs+ local
aaa authentication enable default tacacs+ none
aaa authorization exec default tacacs+ none
```

A local user called admin is created, along with a password. If the AAA server is unavailable (an unlikely event), this account can be used to gain access to the device at privilege level 15. If the AAA server is available, then it is used as the single authority for AAA. Thus, knowledge of the fallback account is not, by itself, sufficient to give a person access to the device (it would require, in addition, that the device can no longer contact a remote AAA server).

Table 2 Examples of Command-Line Pattern Matching

| Rule | Input | Matched | Comments |
|-----------------------|----------------|---------|--|
| serial[0-9] | serial1 | Yes | Because 1 is in the range of 0-9, it matches. |
| | serial1a | Yes | The rule says that only the first character after the word <i>serial</i> must be in the range of 0 to 9; any other characters before or after the string serial1 will result in a match. |
| | serial 0 | No | The word <i>serial</i> followed by a numeric value needs to be present in a single command argument for the command line to match. |
| | enable serial0 | Yes | The word <i>serial</i> followed by a digit appears in the command line. |
| serial[0-9]\$ | serial1a | No | The \$ sign at the end of the rule means end of line; therefore, the command line must end with a digit and not an alpha. |
| ^serial[0-9]\$ | enable serial0 | No | The ^ sign at the start of the rule means that the first argument must start with the word <i>serial</i> . |



Corporate Headquarters

Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA

www.cisco.com

Tel: 408 526-4000

800 553-NETS (6387)

Fax: 408 526-4100

European Headquarters

Cisco Systems Europe
11 Rue Camille Desmoulins
92782 Issy-les-Moulineaux
Cedex 9
France

www-europe.cisco.com

Tel: 33 1 58 04 60 00

Fax: 33 1 58 04 61 00

Americas Headquarters

Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA

www.cisco.com

Tel: 408 526-7660

Fax: 408 527-0883

Asia Pacific Headquarters

Cisco Systems, Inc.
Capital Tower
168 Robinson Road
#22-01 to #29-01
Singapore 068912

www.cisco.com

Tel: +65 317 7777

Fax: +65 317 7799

Cisco Systems has more than 200 offices in the following countries and regions. Addresses, phone numbers, and fax numbers are listed on the

Cisco Web site at www.cisco.com/go/offices

Argentina • Australia • Austria • Belgium • Brazil • Bulgaria • Canada • Chile • China PRC • Colombia • Costa Rica • Croatia • Czech Republic • Denmark • Dubai, UAE • Finland • France • Germany • Greece • Hong Kong SAR • Hungary • India • Indonesia • Ireland • Israel • Italy • Japan • Korea • Luxembourg • Malaysia • Mexico • The Netherlands • New Zealand • Norway • Peru • Philippines • Poland • Portugal • Puerto Rico • Romania • Russia • Saudi Arabia • Scotland • Singapore • Slovakia • Slovenia • South Africa • Spain • Sweden • Switzerland • Taiwan • Thailand • Turkey • Ukraine • United Kingdom • United States • Venezuela • Vietnam • Zimbabwe

All contents are Copyright © 2002 Cisco Systems, Inc. All rights reserved. Catalyst, Cisco, Cisco Systems, Cisco IOS, the Cisco Systems logo, and PIX are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the U.S. and certain other countries.

All other trademarks mentioned in this document or Web site are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0203R)