

Configuring Secure Authentication on Cisco Routers and Switches

Michael Martin

Welcome to the second of three articles on how to implement a cost-effective and secure system auditing and accounting (SAA) system for Cisco networks based on Terminal Access Controller Access Control (TACACS). In this article, we will cover configuration options for secure access to Cisco routers and switches in a multi-user access environment. Topics will include using SSH 1.5 for VTY access on IOS based devices and configuring IOS and CatOS-based devices (used on Catalyst 4xxx, 5xxx and 6xxx series switches) to use TACACS+ for login and to enable authentication and event account reporting.

SAA and TACACS+

System auditing and accounting are the activities involved with the adding and removal of user accounts, access privilege assignment, and logging of system accesses and events, including user command line executions. An SAA system is typically part of a computer's operating system. In the case of application specific operating systems and hardware devices, there is often little in terms of local security and system and event accounting facilities. This is largely due to the limited resources available for processing and storing user accounting and event data. To provide these services, developers use authentication, authorization and accounting (AAA) client/server network protocols such as RADIUS and TACACS+. These allow the functions to be offloaded to systems that can scale to accommodate large system densities and user populations and process the accompanying command and event data. What do SAA, AAA and TACACS+ mean to you, the network administrator/manager? They offer the ability to centrally manage and audit the user access to your Cisco routers and switches in a secure and manageable fashion.

Secure And Restricted Communications Channel: Configuring SSH Under IOS

In Part One, we discussed the desirability to communicate over a secure channel. With routers and switches the urgency is increased -- if compromised, an attacker has access to the entire network. Cisco supports SSH 1.5 as part of its IPsec VPN IOS code set. The Data Encryption Standard (DES) is the only supported encryption method (Blowfish is the exportable encryption that is also supported by most SSH implementations). The IOS "DES Only" SSH implementation comes in two flavors of DES: DES-56 (or standard DES) and 3DES encryption. The DES-56 version is available for all of the Cisco router platforms from the 8xx up. The 3DES version is only supported on the 8xx, 17xx, 26xx, 36xx, 47xx, 7xxx, and 12xxx platforms, and is not exportable. Both of the 56-DES and 3DES require appropriate licensing, but are available on the CCO web site (3DES requires that you complete non-export use discloser form). The DES-56 support creates problems on the client side, and because it is considered by most to be too weak to ensure a secure channel, it is not widely supported. There is very limited support for DES-56 on Unix, however, there are a number of Mac and Windows clients that support DES-56. The IOS configuration steps for SSH DES-56 and 3DES are the same.

There are four steps for enabling SSH 1.5 support on IOS. The first is enabling user-based authentication, which we will cover in a moment. Second, the router's DNS domain is configured using the `<ip domain-name {domain}>` configuration mode: command.

```
RT01-outland-nw(config)#ip domain-name outland.net
```

Step three is the generation of the RSA session encryption keys. This is accomplished using the configuration command `<crypto key generate rsa>`. The Cisco IOS provides support for key sizes of 360 to 2048 bits in length. The default is 512, with a recommended size of 1024. The bigger the key,

Configuring Secure Authentication on Cisco Routers and Switches

Michael Martin

the longer the generation time; on low-end routers large keys can take some time. Once the keys are generated, if you are on the console, (which you should be if you are making these configuration changes, or if you have evoked <terminal monitor> while on a VTY session) you will see a system notification that SSH has been enabled.

```
RT01-outland-nw(config)#crypto key generate rsa
```

The name for the keys will be: RT01-outland-nw.outland.net

Choose the size of the key modulus in the range of 360 to 2048 for your General Purpose Keys. Choosing a key modulus greater than 512 may take a few minutes.

```
How many bits in the modulus [512]: 1024
```

```
Generating RSA keys ...
```

```
[OK]
```

```
7w1d: %SSH-5-ENABLED: SSH 1.5 has been enabled
```

```
RT01-outland-nw(config)#
```

The final step (number four) is to enable SSH transport support for the VTYs. This requires that you go into line configuration mode and define the supported transport modes using the <transport input {telnet|SSH|rlogin}>:

```
RT01-outland-nw(config)#line VTY 0 4
```

```
RT01-outland-nw(config-line)#transport input SSH
```

```
RT01-outland-nw(config-line)#exit
```

```
RT01-outland-nw(config-line)^Z
```

By default, VTY's transport is telnet; in the example above telnet access has been disabled and only SSH is supported. If support for SSH and telnet is required, add the keyword *telnet* after *SSH*. Now you are ready to configure user-based authentication.

Configuring Local User-Based Authentication On IOS

IOS and CatOS hardware support two authentication modes. The default is "line mode" authentication, where a password is set for the **Console** and **Aux** serial ports and the VTY ports. For user-based authentication, the IOS supports two modes: **aaa old-model** and **aaa new-model**. The original motivator for the development of the TACACS+ protocol was to provide AAA services for dial-in network access servers. *Authentication* provides user identification and determines system access. *Authorization* determines what the user is permitted to do on the network or on a given system. *Accounting* logs what the user did when logged on to the network/system.

Technically, *aaa old-model* is active by default but not configured. To configure basic user-based authentication, you need to perform two tasks. First, create local user accounts using the configuration mode commands <username {user} password {usr password}>: Then, using LINE configuration mode, enable user-based login services. Here is an example:

Configuring Secure Authentication on Cisco Routers and Switches

Michael Martin

```
RT01-outland-nw(config)#username root password root
RT01-outland-nw(config)#line VTY 0 4
RT01-outland-nw(config-line)#login local
RT01-outland-nw(config-line)#exit
RT01-outland-nw(config)#service password-encryption
```

Since the user and password information is stored locally in the router configuration, it is a good idea to enable password encryption using the `<service password encryption>` configuration command.

Here is the configuration without encryption:

```
hostname RT01-outland-nw
!
enable password 7 0603022C495A000A54
!
username root password 0 root
```

Here is the configuration with encryption (notice the 0 and the 7 preceding the password; 0 indicates the password is not encrypted, 7 indicates that it is):

```
service password-encryption
!
hostname RT01-outland-nw
!
enable password 7 0603022C495A000A54
!
username root password 7 1317181D1F
```

To configure new-model user-based authentication, the configuration command `<aaa new-model>` is used. When executed, this command disables all of the old-model commands (both line and local login modes), which means that when configuring `aaa new-model` you can lock yourself out of the router if you do not complete all of the required commands. Once `aaa new-model` has been activated, locally authenticated login is enabled with the configuration command `<aaa authentication login default local>`. Here is the configuration example:

```
RT01-outland-nw(config)#aaa new-model
RT01-outland-nw(config)#aaa authentication login default local
```

As a precaution, when configuring authentication do not save configuration until you have verified that login works. Then in the event that something has gone badly, you can simply reboot and return to the old configuration. Also, when testing it's a good idea to keep your existing configuration window open and start a new VTY session to test login. Again, if something goes wrong you still have access and can make changes. If problems do come up with authenticating, enable debugging using the `exec` command `<debug aaa authentication>`. TACACS+ also has the debug command `<debug tacacs>`.

Configuring Secure Authentication on Cisco Routers and Switches

Michael Martin

Under both old and new model, enable authentication is configured using `<enable password>` or `<enable secret>`, which is stored locally.

Configuring IOS/CatOS User-Based And Enable Authentication With TACACS+

TACACS+ authentication support differs slightly between IOS versions 11.x and 12.x and CatOS, which uses a different command line. But overall, the configuration elements are the same. These are: defining the TACACS+ server, defining the pre-shared packet encryption key, defining the TACACS+ interface (which sets the source IP address of the TACACS+ packets) and, lastly, defining the TACACS+ as the user authentication protocol. Enable authentication is also supported by TACACS+ as an additional command; this overrides the locally defined enable password (or enable secret). Enable access privileges are then defined as part of the users TACACS+ account definition.

Before enabling TACACS+ authentication, it is a wise idea to create a "backup" local account:

```
RT01-outland-nw(config)#username root password somepassword
```

With IOS, define the TACACS+ server and pre-shared key using the configuration command `<tacacs-server [variable] {definition}>`:

```
RT01-outland-nw(config)#aaa new-model
RT01-outland-nw(config)#tacacs-server host 172.30.71.103
RT01-outland-nw(config)#tacacs-server key secretkey
```

With CatOS, the `<set tacacs [variable] {definition}>` configuration command is used:

```
RS01-outland-nw> (enable) set tacacs server 172.30.71.103
RS01-outland-nw> (enable) set tacacs key secretkey
```

The source address of the TACACS+ packet is important. For starters, most server implementations use the packet source address to verify if a message is to even be processed. Additionally, the packet IP address is used to identify accounting events. On CatOS based devices, the management IP address is used for all messages. On IOS based devices, since there are multiple interfaces, the message IP address needs to be defined using the configuration command `<ip tacacs source-interface [phy int] >`:

```
RT01-outland-nw(config)#ip tacacs source-interface eth0
```

Under IOS 11.x, configuring TACACS + as the user-based and enable authentication protocol is accomplished with the configuration command `<aaa authentication login default [local|tacacs|radiusenable|local] >`. Here is an example that sets TACACS+ as the primary protocol and local as the secondary. Enable authentication is also configured, using the enable password (not the enable secret, which is not supported under AAA) as secondary. The secondary method is used to provide local authentication in the event the TACACS+ server is down:

```
RT01-outland-nw(config)#aaa authentication login default tacacs+ local
RT01-outland-nw(config)#aaa authentication enable default tacacs enable
```

Configuring Secure Authentication on Cisco Routers and Switches

Michael Martin

CatOS also supports the same "method ordering" configuration for login and enable. User-based authentication is configured using the enable command `<set authentication [tacacs+|radius|local|enable] >`. Similar to the above example, both login and enable authentication (with local backup) is configured:

```
RS01-outland-nw> (enable) set authentication login tacacs local
RS01-outland-nw> (enable) set authentication enable tacacs enable
```

The ability to support multiple TACACS+ server entries is supported in IOS 11.x and 12.x. Server preference is determined by the order the entries appear in the configuration, and both servers must use the same pre-shared key. The idea is that the servers are mirrors of each other in terms user accounting data. The router will send authentication and accounting data to the first active server in the configuration until it fails to respond. In the case of "mid-session" server failure, the router will send request and accounting events to the secondary server. The router will continue to use the secondary server until the session is completed, unless the secondary fails, then it will attempt to use the primary again. This implementation works well for backing up authentication services. But it plays havoc when you are trying to collect consistent accounting data. In IOS 12.x, AAA support was "enhanced" by adding support for server groups, and in 12.1 and above broadcast accounting was added (We will look at that in Part Three).

These enhancements changed the AAA authentication configuration command syntax to `<aaa authentication login default [group|local|enable] >`. The group option followed by the keyword `tacacs+` defines all of the TACACS+ servers in the configuration to be used. Alternatively, a specific group of servers (of those defined in the configuration) can be defined using a custom AAA server list. Here are two 12.x TACACS+ authentication configuration examples. The first uses all of the TACACS+ servers; the second defines a custom group using the configuration command `<aaa group server [tacacs+|radius] {groupname}>`.

Example 1:

```
RT02-outland-ne(config)#tacacs-server host 172.30.71.103
RT02-outland-ne(config)#tacacs-server host 172.17.1.2
RT02-outland-ne(config)#tacacs-server key secretkey
RT02-outland-ne(config)#aaa authentication login default group tacacs+
```

Example 2:

```
RT02-outland-ne(config)#tacacs-server host 172.30.71.103
RT02-outland-ne(config)#tacacs-server host 172.17.1.2
RT02-outland-ne(config)#tacacs-server key secretkey
RT02-outland-ne(config)#aaa group server tacacs+ CORE-TAC
RT02-outland-ne(config-sg-ta)#server 172.17.1.2
RT02-outland-ne(config-sg-ta)#exit
RT02-outland-ne(config)#aaa authentication login default group CORE-TAC
local
RT02-outland-ne(config)#aaa authentication enable default group CORE-TAC
enable
```

Configuring Secure Authentication on Cisco Routers and Switches

Michael Martin

Configuring IOS *command, exec* and *event* Accounting

If you want to implement command and system event accounting on IOS-based routers and switches, you need to use TACACS+. Up to this point, all of the previous configuration examples could have been implemented just as easily using a RADIUS server to provide user-based authentication services (except for enable authentication, which is only supported on certain RADIUS implementations). Accounting is where TACACS+ really shines. While you can configure command line and exec accounting to use a RADIUS server, it will not actually work. If you want to collect this data, you are going to use TACACS+. If you are a RADIUS shop and you are implementing TACACS+ because you want to implement command accounting, there is a compromise. For TACACS+ to report accounting data correctly, it requires that user-based accounting be implemented in some form, so a username can be correlated to an event. This means that you can use a RADIUS server or even local user-based authentication to provide user authentication and deploy TACACS+ only to collect accounting data and provide centralized enable mode authentication if your RADIUS implementation can support it.

With TACACS+, you can collect as much or as little information as you want. In Part Three of this series we will discuss how to customize TACACS account reporting. To configure basic login and logout events, you use the configuration command `<aaa authentication exec default [start-stop|stop-only] [tacacs+|radius|group|broadcast] >`. The *start-stop* and *stop-only* options require some explanation. As I mentioned, TACACS+ was originally developed for dial-in accounting. When the *start-stop* option is used, two accounting records are sent to the accounting server -- at session startup and at session termination. The *stop* record reports the amount of time on the system and when the user logs off, but the *start* record tells you when the user logs on. Here is a configuration example that enables login start-stop accounting:

```
RT02-outland-ne(config)# aaa accounting exec default start-stop tacacs+
```

Now let's take a look at configuring command accounting. To enable command accounting, use the configuration command `<aaa accounting commands [0-15] default [start-stop|stop-only] [tacacs+|radius|group|broadcast] >`. This example configures command accounting for all enable/configuration commands, using the multi-server reporting *broadcast* option:

```
RT02-outland-ne(config)# aaa accounting 15 default start-stop broadcast  
group CORE-TAC
```

In order to implement IOS command accounting effectively, an understanding of the IOS security model is needed. The IOS supports a 15 level security model. By default, levels 0, 1 and 15 are implemented. Additional levels can be implemented by assigning an exec shell command a specific "enable level." The term "enable level" is used because to access a command that has been assigned to a specific security level, you need to use the exec level 0 command `<enable [1-15] [password] >` to authenticate to that level or higher. To set the security level of a specific command, use the configuration command `<privilege exec level [0-15] [exec command] >`. To set the password for a specific enable level use the configuration command `<enable [password|secret] level [1-15] {password}>`. The following example configures the exec command ping, which operates in a limited form in exec level 1, and in extended mode in exec level 15 to operate in extended mode in exec level

Configuring Secure Authentication on Cisco Routers and Switches

Michael Martin

5. The value of this kind of model is twofold. First, quite often network support personnel need to access specific commands for troubleshooting, and nothing else. This approach gives them the access to the commands they need without affecting the access of users with higher access privileges.

```
rt04-outland-ne(config)#enable secret level 5 en5password
rt04-outland-ne(config)#privilege exec level 5 ping
rt04-outland-ne(config)#^Z
rt04-outland-ne>enable 5
Password:
rt04-outland-ne#ping
Protocol [ip]:
Target IP address: 172.30.71.1
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.30.71.1, timeout is 2 seconds:
!!!!!
```

Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms

Second, using tiered accounting allows you to create custom command accounting by defining specific commands at certain security levels and then tracking just those levels.

With your TACACS server up and routers and switches now centrally authenticated, it is time to take a look at customizing command accounting and account log processing and reporting. We will look at that in Part Three of this series. In addition to accounting, we will also discuss implementing server and IOS access security, enable authentication options and service checking to ensure that your TACACS server stays up and running.