

Time-saving techniques for JUNOS software configuration

JUNOS Cookbook™



O'REILLY®

Aviva Garrett

Basic Router Security and Access Control

2.0 Introduction

In the last few years, routers have increasingly become targets of malicious hackers attempting to launch distributed denial-of-service (DDoS) and other attacks across the Internet. Having control of a router, especially one with high-speed links, provides an even greater opportunity for mischief than just controlling PCs. A hacker in control of your router can reconfigure the system and take over your entire autonomous system (AS). Hackers are often able to log in to and take over routers simply because of negligence on the part of a router administrator who doesn't implement basic security precautions, such as setting a password for the root account or using a password that can easily be discovered, such as `juniper`, `cisco`, `root`, or `admin`. Given the increasing number of malicious attacks occurring on the Internet, it is vital for you to secure your router.

This chapter talks about how to configure router access, including setting up login accounts, and other basic security measures you should take to control access to the router and to protect your router from undesired access.

There is nothing complicated about what you need to do to protect your router. Basic router security consists of three components. Two of these—limiting physical access to your router and configuring the JUNOS software to minimize the vulnerability of your router—are under your control. Properly configuring the router to be as secure as possible, while at the same time ensuring that you don't misconfigure the router to increase its vulnerability to attack, is often called *hardening* the configuration. The third component of security is some of the default behaviors of the JUNOS software that help protect the router.

To limit physical access to your router, we strongly recommend keeping your router in an area that has restricted access, such as a room that is locked or has badge access, and then limiting the number of people who have access to that area. Anyone who can physically get to a router can do a lot of damage, from removing hardware or cables from the router to connecting a PC to the router's console port, which

lets them gain access to the router as root and gives full access to and control of the router's configuration and files. You should also never leave a modem connected to the router's console port to ensure that no one can gain access this way.

In the basic router configuration that you set up (described in Recipe 1.1), the following default software behaviors are in place to protect the security of your router:

- Only console access to the router is enabled by default. Remote management access to the router and all management access protocols, including Telnet, FTP, and SSH (secure shell), are disabled. When you initially configure the router, you connect a terminal to the router's console port. After this, you want to keep the router in an area that has limited physical access, so you need to enable a way to remotely log in to the router. For the best security, you should enable only SSH access.
- The JUNOS software does not support the SNMP Set capability for editing configuration data, although it does support this capability for monitoring and troubleshooting the network. There are no known security issues associated with this. (You can configure the software to disable this SNMP Set capability.)
- The JUNOS software does not forward directed broadcast messages. (Directed broadcasts are datagrams with a destination address of an IP subnetwork broadcast address.) Directed broadcasts are open to spoofing, which is used in DoS attacks.
- The JUNOS software ignores martian addresses that contain the following prefixes: 0.0.0.0/8, 127.0.0.0/8, 128.0.0.0/16, 191.255.0.0/16, 192.0.0.0/24, 223.255.55.0/24, and 240.0.0.0/4. (Martian addresses are reserved host or network addresses about which all routing information should be ignored.) You may want to add other prefixes to the martian list, such as RFC 1918 address space and bogon prefixes (see Recipe 9.5).

A key to router security is controlling who can log in to the router and what they can do once they are logged in. For each user who is allowed to work on the router, you should create a login account that defines the user's login name and password and the class of operations that they can perform on the router.

Strategies for Choosing Passwords

Passwords for the root account and for user accounts are often the weakest links in router security. For root and for any user who can log in to the router, you should always set a password, and the password you should choose should be a strong password, one that is hard to crack, not a weak one. You want to make it impossible for a person with malicious intentions to be able to gain login access to your router, especially as root or any user who has root permission or who has permission to modify the router's configuration or any files on the router, or to shut down or reboot the router.

All JUNOS passwords are encrypted, but this means only that the password stored on the router or in a configuration file is stored in an encrypted form. Someone reading the configuration on the router won't be able to see the plain-text password, and if you copy the file over the network and someone sniffs the session, they won't see the passwords in the file. Even though the passwords are encrypted in the configuration, you should take care not to let them circulate. It's still possible to use programs such as `crack` to guess clear-text passwords, encrypt them, and compare them to a list of encrypted strings (although this is not the case with, for example, SSH public keys). For this reason you always need to use strong passwords and prevent even encrypted versions of your passwords from falling into the wrong hands.

To understand what a strong password is, we should look first at what constitutes a weak password. It should go without saying, but bears repeating anyway, that the weakest password is no password at all. A number of groups that monitor network security still find routers that have no passwords set on them. Other weak passwords are those that are easy to guess and include common words such as the name of your router vendor (such as `juniper`), the string `admin`, using the username as the password (for example, username `root`, password `root`, or `admin/admin`), and using the string `password` or `Password`. Other guessable passwords are words or strings like your birthday, spouse's name, or the name of any person. Weak passwords are also those that are vulnerable to brute force attacks, in which an automated program tries a large number of possible passwords, and to dictionary attacks, which are automated programs that try all words in a dictionary in an attempt to crack an account's password. Keep in mind that dictionaries for all languages are now available on the Internet, as are dictionaries specific to technical and other fields, so all words that might be present in them are weak passwords, even derivations that substitute numbers for letters.

A strong password is everything a weak password is not, and then some. It should include numbers, symbols, and a mix of uppercase and lowercase characters. Other suggestions are to pick a couple of letters from a phrase you know well or to pick some unrelated words and connect them into a single string with punctuation marks or other symbols. Remember that a strong password is a good password only if you can remember it without writing it down.

User Authentication

Each user must have a login account and password to be able to log in to the router. The JUNOS software supports three methods of user authentication: local password authentication, Remote Authentication Dial-In User Service (RADIUS), and Terminal Access Controller Access Control System Plus (TACACS+). With local password authentication, you set a password for each user in the router's configuration file. RADIUS and TACACS+ are centralized authentication databases for validating users who attempt to access the router using any access method. They are both distrib-

uted client-server systems—the RADIUS and TACACS+ clients run on the router, and the server runs on a remote network system.

You can configure the router to be both a RADIUS and TACACS+ client, and you can also configure authentication passwords in the JUNOS configuration file. If you use multiple authentication methods, you can set the order in which the router tries the different authentication methods when verifying user access. If you do not set the order, the router uses the local password first.

Password Encryption

All passwords that you enter in a JUNOS configuration are encrypted. The JUNOS software supports several methods for securing passwords using encryption and hashing algorithms (encryption is a one-to-one mapping, so it's possible to decrypt, while hashing is a many-to-many mapping, so it's impossible to unhash):

SHA1

Secure Hash Algorithm 1 is the newest algorithm, developed in 1995. It is a secure hashing algorithm that produces a 160-bit message digest that is used as a signature for a message and that must be verified by the recipient. SHA1 is considered secure because it is computationally infeasible to find a message that corresponds to a given message digest or to find two different messages that produce the same message digest. Any change to a message in transit results in a different message digest, so the signature fails to verify. However, SHA1 has recently been proven not to be as strong as originally thought.

MD5

Message Digest 5, developed in 1991, is a message hashing algorithm that takes a message of arbitrary length and produces a 128-bit hash function. When developed, it was thought to be computationally infeasible to produce two messages with the same message digest. The use of MD5 has recently been deprecated by the U.S. Department of Defense.

DES

Data Encryption Standard, an encryption algorithm developed in 1976, uses a 56-bit key. Many people never thought DES was very strong in the first place.

SSH

Secure shell, Version 1 (RSA) and Version 2 (DSA), is a security protocol that was originally developed with the Unix BSD software.

Even when you configure a plain-text password, the JUNOS software encrypts it immediately after you type it. Also, the software forces you to use a somewhat strong password, because the password must be at least six characters long and must include either a change of case or a special character.

2.1 Allowing Access to the Router

Problem

You just installed your router and can log in to it only through the console port. You want to allow administrators to securely log in to it over the network.

Solution

You should use SSH to provide secure encrypted sessions to the router:

```
aviva@router1# set system services ssh
```

Discussion

With SSH, both the password you type and the connection itself are encrypted using a well-tested industry-standard protocol, so both are protected. The systems that you use to connect to the router must have SSH client software. For greater security, you should use SSH keys on the client. You can find information about obtaining SSH software at <http://www.ssh.com> and <http://www.openssh.com>.

When you log in to the router with SSH, you are prompted for your password:

```
aviva-server1% 122: ssh router1
The authenticity of host 'router1-mycompany.com (192.168.71.246)' can't be
established.
DSA key fingerprint is 2c:a9:35:c5:2a:db:12:5b:b6:6e:0b:17:ae:ec:d4:55.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'router1-mycompany.com' (DSA) to the list of known hosts.
aviva@router1-mycompany.com's password:
--- JUNOS 7.4R1.7 built 2005-10-24 08:10:28 UTC
aviva@router1>
```

You can also allow users to connect to the router with Telnet, but if security is your highest priority, you should not use Telnet. Telnet connections and passwords are not encrypted so they can be intercepted. However, if your network itself is well protected with firewalls, you can enable Telnet to let users access the router:

```
aviva@router1# set system services telnet
```

The only user who can never log in using Telnet is root. To log in as root, you must use SSH or the console.

SSH and Telnet provide terminal sessions to the router so you can log in to the router. The commands in the JUNOS software that copy files to and from the router use SSH, but they can also use FTP. Because FTP is not secure in and of itself, if you want to use it to copy files, the best thing to do is to enable FTP just before you need to copy the files:

```
aviva@router1# set system services ftp
aviva@router1# commit
```

You need to enable FTP only if you want to FTP something to the router—that is, when the router is the FTP server. If the router is the FTP client and you are fetching a file from an FTP server, you do not need to enable FTP on the router. The FTP client on the router is always present and running.

Then disable FTP after you have copied the files:

```
aviva@router1# delete system services ftp
aviva@router1# commit
```

One way to secure FTP is to create a firewall filter that uses source address filters to limit access to the FTP port, particularly if the source addresses are forced to come through an encrypted tunnel. Recipe 9.8 discusses how to create firewall filters.

If you are using a router that supports the J-Web browser for configuring and monitoring the router, you can enable secure HTTP on the router:

```
[edit system]
aviva@router1# set services web-mangement https
```

See Also

Recipes 2.14 and 9.8

2.2 Controlling Root Authentication

Problem

When you first installed your router, you created a password for the root user (see Recipe 1.1). With this initial configuration, anyone who knows the root password can log in to the router using Telnet. You want to make the root login more secure.

Solution

There are two solutions, depending on the desired level of security. One solution is to use SSH for the root password. You can specify the root password in plain text as you are configuring the router:

```
[edit]
root@router1# set system root-authentication ssh $1991poppl
```

You can also load an SSH key file from a server:

```
[edit]
aviva@router1# set system root-authentication load-key-file server1:/homes/aviva/.ssh/id_dsa.pub
.file.19692 | 0 KB | 0.3 kB/s | ETA: 00:00:00 | 100%
aviva@router1# show
system {
    root-authentication {
```

```

ssh-rsa "1024 35
9727638204084251055468226757249864241630322207404962528390382038690141584534964170019
6106083587229615634757849182736033612764418742659468932077391083448101268312595772262
5461667999278316123500438660915866283822489746732605661192181489539813965561563786211
94032768780653816960202749164163735913269396344008443 root@mynetwork.com"; # SECRET-
DATA
}
}

```

The second solution for providing root authentication forces the root user to log in using the router's console port:

```

[edit system]
aviva@router1# set services ssh root-login deny

```

Discussion

There are two schools of thought about root access to the router. One suggests using SSH for the root password because SSH is more secure than using just the password you initially configured on the router. SSH provides inband access to the router, meaning that the root user can log in from anywhere in the network, especially if the router is part of a service provider network. A second school of thought suggests disabling SSH access for root altogether, forcing the root user to log in on the router's console port. Access using the console port is assumed to be secure in that you must be on the company's internal network to even have access to the console port.

Generally, there's very little reason to provide access to the root login inband, so unless you really need this on your network, for strict security you should not provide root SSH access.

If you use SSH to authenticate the root password, you need to first enable SSH services on the router and be running SSH on your server.

This first command in this recipe sets the root's SSH password by entering it directly in plain text in the router's configuration file. When you use the `show` command to view the configuration, you see only an encrypted version of the password. The second command copies an SSH key file from a server. After you type the command, the contents of the key file are immediately copied into the configuration file.

Any SSH password you set is in addition to the plain-text password. You should leave a local root password on the router so you can log in using the console port.

The second command in this recipe disables SSH for the root user altogether. Anyone needing to log in to the router as root must log in through the router's console. The reason for doing this is not so much that you want the root user to come in to the router through the console, but rather that you want them to log in using an individual account and then exit to the shell and use the `su` command to become root only if they need to. There are two reasons for this. First, you want to avoid habitual use of the root account. Logging in with root is like running with scissors: there are lots of ways to hurt yourself. It's much better to get in the habit of using a non-root

account and su only when required. Second, and more importantly, you want to maintain accountability. If you log in to a router, su to root and then do something horrible, there will be an audit trail to trace the source of the problem. However, if you had logged in as root in the first place, the action wouldn't be traceable to an individual router user.

See Also

Recipe 2.3

2.3 Logging In to the Router's Console

Problem

You have lost access to the router through other means and you need a way to log in.

Solution

Log in to the router using the console port:

```
aviva@server1% telnet router1-con
Trying 172.19.121.19...
Connected to router1-con.mycompany.com.
Escape character is '^]'.
router1 (ttyd0)
login:
```

Discussion

The console port on JUNOS routers is enabled by default. If you ever lose access to the router through normal login means, you can log in to using the console port. While it is possible to disable the console port, it is really not recommended.

Use of the console port is not really required other than when you are initially installing the router. However, many people use the console port as the access of last resort. You can set up a terminal server with console connections to a number of devices in the event that the network fails. Also, if someone accidentally misconfigures the router and locks themselves out, or if routing has failed, you can still get in to the router remotely using the console port.

Access over the console port is the only method that allows you to remain connected to the router during a reboot. A reboot logs you out of the session on the router, but you will still be connected and will be able to halt the reboot or watch the messages as the router reboots.

2.4 Setting the Login Authentication Methods

Problem

You want to use a RADIUS or TACACS+ server to authenticate user logins to the router, and you want to specify a backup login authentication method in case the primary method is unavailable.

Solution

Use the following command to set RADIUS as the primary authentication method and to set the backup method to be the user accounts configured on the local router:

```
[edit]
aviva@router1> set system authentication-order [ radius password ]
```

If you are using TACACS+, you can set up something similar:

```
[edit]
aviva@router1# set system authentication-order [ tacacs password ]
```

Discussion

When users log in to the router, the JUNOS software can authenticate the username and password against an account that is configured locally in the router configuration file or against an account that is configured on a remote RADIUS or TACACS+ server.

There are a number of methods to authenticate users attempting to log in to the router. The default method is to use the username and password configured on the router and to try no other method if the authentication fails. This method is the equivalent of using the `set system authentication-order password` command. You should always configure passwords in the configuration file for at least a few users so someone can always log in to the router (see Recipe 2.8).

To have the router use a RADIUS or TACACS+ server as the primary user authentication method, you must change the order in which the JUNOS software tries different authentication methods. The first command in the recipe configures RADIUS to be the primary user authentication method, and the second command configures TACACS+ as the primary method. Both commands set the user account configured on the router (`password`) as the backup authentication method. Providing a backup method means that users will always be able to log in to the router if there are problems with the RADIUS or TACACS+ server. (Recipes 2.12 and 2.13 describe how to configure RADIUS and TACACS+ user authentication.)

With the configuration in this recipe, when a user tries to log in to the router, the router first checks the username and password against the RADIUS or TACACS+ server. If they match, the user is authenticated and the router logs them in. If the remote authentication fails, the router checks its local configuration. If the user has a

local account and the password matches, the user is logged in. If there is no match in either place, the user is denied access to the router.

A slight twist to this recipe is to use only a single authentication, specifying a remote method. The following command uses only RADIUS authentication:

```
[edit]
aviva@router1> set system authentication-order radius
```

This configuration allows users to log in to the router only if the RADIUS server has an account for them and only if the RADIUS server is up. This means that as long as the RADIUS server is up, users not listed in the RADIUS database won't be able to log in to the router even if there is a configured account for them on the router. However, if the RADIUS server fails or becomes unreachable, the JUNOS software authenticates the user locally. If you configure multiple RADIUS servers, the software checks for locally configured user accounts only after all the servers fail.

Make sure you configure user accounts and assign passwords in the JUNOS configuration for some users (see Recipe 2.5) so that login access to the router will be possible if the RADIUS or TACACS+ servers fail.

See Also

Recipes 2.5, 2.8, 2.12, and 2.13

2.5 Setting Up Login Accounts on the Router

Problem

You want a number of people to be able to work on the router to monitor and configure it.

Solution

Set up a login account for each person who is allowed to log in to the router:

```
[edit system login]
aviva@router1# set user sage class operator
aviva@router1# set user sage full-name "sage david"
aviva@router1# set user sage uid 1991
aviva@router1# set user sage authentication plain-text-password
New password:
Retype new password:
```

Discussion

For each user who you want to log in to the router, create a login account, providing information about the user that is similar to what you set for Unix accounts. The JUNOS software uses this account to locally authenticate the user.

Each account requires two pieces of information: a login name (configured with the user statement) and a login class (configured with the class statement), which associates a set of privileges with the user, defining the scope of operations that can be performed on the router. As with Unix account names, the username must be unique on the network and cannot contain spaces, colons, or commas.

This recipe configures an account for the username *sage*, who has a privilege level operator, which allows her to perform most operational commands but not enter configuration mode. operator is one of the predefined privilege classes. Recipe 2.10 explains the other predefined classes and how to create custom privilege classes.

The `set user sage full-name` command configures the user's complete name. Setting this is optional, but you may find it convenient or easier to read the person's given name rather than their account name when checking the configuration to see who has access to a router. Enclose the full name in quotation marks if it contains spaces.

The third command, `set user sage uid`, assigns a user ID (UID) of 1991. The UID is basically the same as the Unix UID. It's included in the JUNOS configuration primarily because UIDs are integral to Unix systems and the JUNOS software runs on FreeBSD. As with Unix, the JUNOS software uses the UID when establishing and enforcing file permissions and file access. Configuring the UID is optional. If you do not configure it, the JUNOS software assigns one, generally using the lowest available UID number, starting at 2000. Here's an example of automatic assignment:

```
[edit system login]
aviva@router1# set user sage class operator
aviva@router1# commit check
configuration check succeeds
aviva@router1# show
user sage {
  uid 2006;
  class operator;
}
```

You see that UID 2006 has been assigned to the user *sage*. You might want to explicitly configure the UID if users will be transferring files to Unix systems to ensure that users have the same UID on both JUNOS and Unix systems so that there are no file ownership issues.

The last command in this recipe establishes a plain-text password for the login account. You are prompted for the password, and nothing is displayed when you type and retype it.

To set up a user account, only the username and privilege class are required, but the password and other information are optional. If you omit the class, the CLI displays a warning:

```
[edit system login]
aviva@router1# set user sage full-name "sage david"
aviva@router1# show
```

```
user sage {
    full-name "sage david";
    ## Warning: missing mandatory statement(s): 'class'
}
```

Also you will not be able to commit a configuration if you forget to assign a class:

```
[edit system login]
aviva@router1# commit check
[edit system login]
'user sage'
Missing mandatory statement: 'class'
error: configuration check-out failed: daemon file propagation failed
```

If you configure a password with a user's login account, the software authenticates the user against this password during the login process. This is the default authentication method. You should always configure passwords in the configuration file for at least a few users so there is always someone who can log in to the router.

If you configure a password with the user's login account, but want the router to have a RADIUS or TACACS+ database authenticate the user before checking against the password configured on the router, you must change the authentication order so that the RADIUS or TACACS+ server is checked before the local user account (see Recipe 2.4). You must also configure RADIUS or TACACS+ user authentication (see Recipes 2.12 and 2.13). In this situation, users will be able to log in to the router if the remote authentication server fails.

If you do not configure a password with the user's login account, authentication is done remotely only, using a centralized RADIUS or TACACS+ authentication database, instead of locally based on the router configuration file. This type of account is analogous to a Unix account that has a * in the password field, which does not allow logins based on the password file, but can allow logins based on other valid means of authentication, such as RADIUS. For this to work, you must change the authentication order so that RADIUS or TACACS+ is checked first (see Recipe 2.4), and you must configure the RADIUS or TACACS+ user authentication (see Recipes 2.12 and 2.13). Users with this type of account will not be able to log in to the router if the authentication server is down or if there are network problems accessing the server.

You can also create a generic login account (see Recipe 2.8) or a group login account (see Recipe 2.9) instead of configuring individual accounts for each user. These types of accounts authenticate against the RADIUS or TACACS+ database. Here, too, you must change the authentication order so that RADIUS or TACACS+ is checked first (see Recipe 2.4), and you must configure the RADIUS or TACACS+ user authentication (see Recipes 2.12 and 2.13). Users with this type of account will not be able to log in to the router if the authentication server is not available.

This recipe configures a plain-text password for the user's authentication. The password must be at least six characters long and must contain at least one case or one letter-to-number change. (Recipe 2.6 explains how to modify the default password

format.) Here you type the plain-text-password keyword slightly differently than for some other JUNOS configuration statements. After you type plain-text-password, press Enter. The software then prompts you to type and then retype the password. Type the password in plain text, and the JUNOS management process, MGD, immediately encrypts it using SHA1 encryption by default. (To change the default encryption, see Recipe 2.7.) You then see only the encrypted version of the password. Notice that the keyword plain-text-password has changed to reflect the fact that the password is now encrypted:

```
encrypted-password "$1$b01I/WUw$bfaYF0LHxHxVCm7XyS7eG."; ## SECRET-DATA
```

To insert a previously encrypted DES, MD5, SHA1, SSH Version 1 (RSA), or SSH Version 2 (DSA) password, cut and paste (or type) the encrypted password into the configuration statement, enclosing it in quotation marks. Here is an example of an encrypted MD5 password:

```
aviva@router1# set user sage authentication encrypted-password  
"$1$EpZ4gDEb$52KHLKA2QuqfJ83tFUvwd1"
```

You can define both encrypted and SSH passwords for a single user:

```
aviva@router1# show  
user sage {  
  authentication {  
    encrypted-password "$1$kfpFHEom$wrWwtk69gvdbWInzsoIOb."; ## SECRET-DATA  
    ssh-rsa "1024 35 1463306454911004878538350206193424119843602248584695395  
55532106068887517531015448370922784460860638276095178917479848571459866004412524  
44671184497730460934239780966471256093384818219663350688876263790111832271705295  
56264636153739986671412936949237931460389138872790447839157168037660941582648407  
66391853943503 sage@red.juniper.net"; ## SECRET-DATA  
  }  
}
```

For basic router security considerations, you should limit access to the router to only those people who really need access, and you should carefully consider which privileges each person is given, so that a person can perform their job function and responsibilities, and nothing more.

See Also

Recipes 2.4, 2.6, 2.7, 2.8, 2.9, 2.10, 2.12, and 2.13

2.6 Changing the Format of Plain-Text Passwords

Problem

You want to require the passwords for user accounts to be longer than six characters and to have more than one case change.

Solution

Set all plain-text passwords to be from 8 to 20 characters long and to contain at least two case changes:

```
[edit system login]
aviva@router1# set password maximum-length 20
aviva@router1# set password minimum-length 8
aviva@router1# set password minimum-changes 2
```

Discussion

By default, plain-text passwords must be at least six characters long and must contain one change from either letters to numbers (or vice versa) or from lowercase to uppercase (or vice versa). You can harden the router's security even more by increasing the minimum password length and the minimum number of case and letter-to-number changes.

The commands in this recipe require that all plain-text passwords be from 8 to 20 characters long and contain at least 2 case changes. The changes take effect when you next configure a plain-text password for a user:

```
[edit system login]
aviva@router1# set user sage authentication plain-text-password
New password:T91912
error: minimum password length is 8
error: require 2 changes of case, digits or punctuation
```

This password is not acceptable because it is shorter than eight characters and has only one change from a letter to a number. An example of a valid password with these conditions is \$1991poppI.

When you change the requirements for plain-text passwords, the new parameters affect only newly created passwords, so already existing passwords may not be as secure as your new password policy.

See Also

Recipe 2.5

2.7 Changing the Plain-Text Password Encryption Method

Problem

When setting up passwords for login accounts on the router, if you assigned plain-text passwords, the default encryption is SHA1. You want to change this to either DES or MD5.

Solution

Use the following command to change the encryption used for plain-text passwords to DES:

```
[edit]
aviva@router1> set system login password format des
```

For MD5 encryption, use the following command:

```
[edit]
aviva@router1> set system login password format md5
```

Discussion

All passwords that you enter in a JUNOS configuration are encrypted. For plain-text passwords, you can use one of three types of encryption: SHA1 (the default and the strongest), MD5, or DES. The encryption type that you configure is used for all plain-text passwords. You cannot specify different encryption types for different users.

See Also

Recipe 2.5

2.8 Creating a Login Account for Remote Authentication

Problem

You want to use a RADIUS or TACACS+ database to authenticate users instead of setting up individual login accounts for them on the router.

Solution

Create a login account that has the username remote:

```
[edit system]
aviva@router1# set login user remote class operator
aviva@router1# set login user remote full-name "remote account"
aviva@router1# set login user remote uid 9999
```

Then set the authentication order so that the remote authentication server is checked before the router's configuration file. The following command uses a RADIUS server:

```
[edit system]
aviva@router1> set authentication-order [ radius password ]
```

Use the following command for TACACS+:

```
[edit system]
aviva@router1# set authentication-order [ tacacs password ]
```

Discussion

When you want users to be able to log in to and work on the router, but always want to use a central authentication server, you can set up a placeholder account named `remote` instead of creating login accounts on the router for these users. When a user with no account in the local configuration files tries to log in to the router using her regular username, the authentication is handled by the `remote` account, which queries the RADIUS or TACACS+ server to authenticate the user. If the user's name and password match what is on the server, the user is authenticated and the router logs her in. (Recipes 2.12 and 2.13 explain how to configure the RADIUS and TACACS+ server information.)

As with an individual user account (see Recipe 2.5), you configure a privilege level with the `set user remote class` command and a user ID with the `set user remote uid` command. This recipe sets the privilege level to `operator`, which allows these users to perform most operational commands but not enter configuration mode. (Recipe 2.10 discusses privilege classes.)

This recipe includes the `set user remote full-name` command to provide a description of this account. This command is not required.

Users who are authenticated only by the `remote` account will not be able to log in to the router if the authentication server is down. You should always configure some individual user accounts with passwords on the router so someone can always log in to the router (see Recipe 2.5).

You can create only one `remote` account on the router. This means that all users who don't have an individual user account on the router and who are authenticated by RADIUS or TACACS+ share the same privilege level, which is configured in the `set user remote class` command. Recipe 2.9 describes how to set up remote accounts that have different privilege levels.

See Also

Recipes 2.5, 2.9, 2.10, 2.12, and 2.13

2.9 Creating a Group Login Account

Problem

You want to use a RADIUS or TACACS+ database to authenticate a group of users who perform similar job functions and tasks on the router, instead of setting up individual login accounts for them on the router.

Solution

Create a group account on the router to allow multiple users to be authenticated by the same RADIUS or TACACS+ server account:

```
[edit system login]
aviva@router1# set user noc class operator
aviva@router1# set user noc full-name "NOC team"
```

Then set the authentication order so that the remote server is checked before the router's configuration file. The following command uses TACACS+:

```
[edit system]
aviva@router1# set authentication-order [ tacacs password ]
```

Finally, map the users on the server to the account name configured on the router. The following is the map on a TACACS+ server:

```
user = mike {
    service = junos-exec {
        local-user-name = noc
    }
}
user = sage {
    service = junos-exec {
        local-user-name = noc
    }
}
```

Discussion

When you want a group of users to be able to log in to and work on the router, but always want to use a central authentication server, you can set up a common account instead of creating login accounts on the router for these users. Then in the RADIUS or TACACS+ database, you map the username to the common account name.

The first command in this recipe creates the group account `noc` that has operator privileges and can perform most operational commands but cannot enter configuration mode. This second command, `set user remote full-name`, provides a description of the account. This command is optional, but is suggested so that the meaning of the account is clear. The third command sets TACACS+ as the primary authentication method.

The TACACS+ database in this recipe has two usernames, `mike` and `sage`. When these two users try to log in to the router using their regular login names `mike` and `sage`, the login request is authenticated by the TACACS+ server, which sees that their local username (their login account name on the router) is `noc`. The server returns this information to the router, which logs them in using the `noc` account and gives them operator privileges.

Users who are authenticated only by a group account will not be able to log in to the router if the authentication server is down. You should always configure some indi-

vidual user accounts with passwords on the router so someone can always log in to the router (see Recipe 2.5).

See Also

Recipes 2.4, 2.5, 2.10, and 2.13

2.10 Customizing Account Privileges

Problem

You want to create a custom privilege class to define the operations and actions a user can perform while logged in to the router.

Solution

Create a privilege class that allows users to read but not modify the configuration, and then let them perform all operational mode commands:

```
[edit system login]
aviva@router1# set class operator-plus-read-config permissions [ admin clear
configure floppy interface network reset routing shell snmp system trace view
maintenance firewall rollback security ]
```

Discussion

When you set up login accounts on the router (see Recipe 2.5), each account must have a privilege level, or class, which defines the operations and actions the user can and cannot perform on the router. Each privilege level consists of a collection of *permission bits* that specifies what a user is allowed to do. Table 2-1 lists all the permission bits.

Table 2-1. Login class permissions

Permission	Bit name
All (superuser)	all (can perform all actions)
Delete data from system log, tracing, and other files	clear (using the clear commands)
All control-level operations (bits ending in -control)	control (can view and change all portions of the configuration)
Configure the router	configure (using the configure and commit commands)
Access removable media	floppy
Halt and reboot the router; start a shell and become superuser	maintenance (using the request system commands, and using the CLI start shell command and the su root command)
Access the network	network (using the ping, ssh, telnet, and traceroute commands)
Start and stop software processes	reset (using the restart command, and configure at [edit system processes])

Table 2-1. Login class permissions (continued)

Permission	Bit name
Return to previous configuration	rollback (using the rollback command)
Start a local shell	shell (using the start shell command)
Display router, routing table, and protocol values	view (using the show commands)
User account information (login classes, user IDs)	admin (read only, using the show configuration command) admin-control (read, and configure at [edit system login])
Firewall filters	firewall (read only, using the show configuration command) firewall-control (read, and configure at [edit firewall])
Interfaces, chassis, class of service, forwarding options	interface (read only, using the show configuration command) interface-control (read, and configure at [edit interfaces], [edit chassis], [edit ???], [edit forwarding-options])
Routing, routing protocols, routing policy	routing (read only, using the show configuration command) routing-control (read, and configure at [edit routing], [edit routing-options], [edit policy-options])
Passwords and authentication keys	secret (read only, using the show configuration command) secret-control (read and configure)
IPSec security	security (read only, using the show configuration command) security-control (read, and configure at [edit security])
SNMP	snmp (read only, using the show configuration command) snmp-control (read, and configure at [edit snmp])
Router name, RADIUS, TACACS+, NTP, and other systemwide information	system (read only, using the show configuration command) system-control (read, and configure at [edit system])
Tracing and trace files	trace (read tracing files and configuration using the show configuration command) trace-control (read and configure)

Notice that some bits have two forms, a “simple” form, which gives read-only permission, and a -control form, which gives read and write permission. Except for the all bit (which grants all permissions) and the control bit (which grants read-write permission to the entire configuration), the permission bits are not cumulative, so when you create a custom privilege class, you must list all the bits that apply. Always include the view bit so users can use the show commands in operational mode. If you want users to be able to modify the configuration, include the configure bit.

The JUNOS software has four built-in privilege levels:

superuser or super-user

Can perform any operations on the router (equivalent to the all permission bit). This is similar to the Unix superuser.

operator

Can perform all actions in operational mode available with the clear, network, reset, trace, and view permission bits. Cannot display or alter the configuration and cannot shut down or reboot the router.

read-only

Can perform all actions in operational mode available with the view permission bit, to show information about the router or network. Cannot perform any operations that delete or change files or file contents, that clear statistics, or that change the information on the router.

unauthorized

Can log in to the router but cannot perform any operations on the router except to log out.

The default privilege levels are not explicitly defined in the configuration, but if you did configure them, the first three would look like this:

```
[edit system login]
aviva@router1# set class superuser permissions all
aviva@router1# set class read-only permissions view
aviva@router1# set class operator permissions [clear network reset trace view]
```

There is no way to explicitly configure the unauthorized level.

The command in this recipe defines a custom privilege class that allows users to perform all operational mode commands and to read but not modify the configuration. The clear, network, reset, trace, and view permission bits allow this class to use all operational mode commands. The configure bit allows this class to issue the configure command to enter configuration. The remaining bits are all the read-only bits that allow this class to use the show command in configuration mode. Users in this class can view all the contents of the configuration file except for passwords and keys (we have omitted the secret bit). Because this class has no -control bits, users can't change the configuration, even though the configure bit allows them to issue the commit command:

```
[edit]
aviva@router1# set
unknown command
```

To find out what privileges you have, use the show cli authorization command. Here is a user with superuser privileges:

```
aviva@router1> show cli authorization
Current user: 'aviva' class 'superuser'
Permissions:
  admin      -- Can view user accounts
  admin-control-- Can modify user accounts
  clear      -- Can clear learned network information
  configure  -- Can enter configuration mode
  control    -- Can modify any configuration
  edit       -- Can edit full files
  field      -- Special for field (debug) support
  floppy     -- Can read and write from the floppy
  interface  -- Can view interface configuration
  interface-control-- Can modify interface configuration
  network    -- Can access the network
```

```

reset      -- Can reset/restart interfaces and daemons
routing    -- Can view routing configuration
routing-control-- Can modify routing configuration
shell      -- Can start a local shell
snmp       -- Can view SNMP configuration
snmp-control-- Can modify SNMP configuration
system     -- Can view system configuration
system-control-- Can modify system configuration
trace      -- Can view trace file settings
trace-control-- Can modify trace file settings
view       -- Can view current values and statistics
maintenance -- Can become the super-user
firewall   -- Can view firewall configuration
firewall-control-- Can modify firewall configuration
secret     -- Can view secret configuration
secret-control-- Can modify secret configuration
rollback   -- Can rollback to previous configurations
security   -- Can view security configuration
security-control-- Can modify security configuration
access     -- Can view access configuration
access-control-- Can modify access configuration
view-configuration-- Can view all configuration (not including secrets)
Individual command authorization:
Allow regular expression: none
Deny regular expression: none
Allow configuration regular expression: none
Deny configuration regular expression: none

```

Here is a user with operator privileges:

```

mike@router1> show cli authorization
Current user: 'mike' class 'operator'
Permissions:
clear      -- Can clear learned network information
network    -- Can access the network
reset      -- Can reset/restart interfaces and daemons
trace      -- Can view trace file settings
view       -- Can view current values and statistics
Individual command authorization:
Allow regular expression: none
Deny regular expression: none
Allow configuration regular expression: none
Deny configuration regular expression: none

```

If you do not have permission to perform an operation, you are either “blind” to that operation or you see some type of indication that you cannot perform it. If you try to view the configuration without permission, you see the following warnings:

```

aviva@router1> show configuration
version /* ACCESS-DENIED */;
system { /* ACCESS-DENIED */ };
interfaces { /* ACCESS-DENIED */ };
routing-options { /* ACCESS-DENIED */ };
protocols { /* ACCESS-DENIED */ };
policy-options { /* ACCESS-DENIED */ };

```

If you try to enter a command that you don't have permission to use, the CLI acts as if that command doesn't exist:

```
aviva@router1> clear
unknown command.
```

You should keep these permission levels in mind when trying to use the commands discussed in this book. If you cannot enter the command or do not see it with the CLI help, review your authorization level and check with your system administrator if you need additional permission.

If a user who has a login account but no login class tries to log in, she can get as far as the operational mode prompt, but she can't do anything except log out:

```
warning: user "aviva" does not have a valid login class
aviva@router1> exit
```

How do you find out which permissions are associated with each command and statement? On the router, you can use the help reference command to see the permissions for the configuration statements:

```
aviva@router1> help reference interface address
...
Required Privilege Level
interface--To view this statement in the configuration.
interface-control--To add this statement to the configuration.
```

For a configuration that already exists on the router, you can see the permissions for the statements in the configuration. Use this command from operational mode:

```
aviva@router1> show configuration system | display detail
```

and use this command in configuration mode:

```
[edit system]
aviva@router1# show | display detail
```

Both show the same output:

```
##
## system: System parameters
## require: admin system
## domain-name: Domain name for this router
## match (regex): ^[[:alnum:]._-]+$
## require: system
##
## domain-name mynetwork.com;
##
## name-server: DNS name servers
## require: system
##
## name-server {
##
##   ## DNS name server address
##
##   192.168.15.2;
```

```

}
##
## login: Names, login classes, and passwords for users
## require: admin
##
login {
  ##
  ## Login class name
  ## match (regex): ^[:alnum:]-]+$
  ##

```

The only way to find out the permissions for operational mode commands is to look in the JUNOS product documentation.

Login classes have one more feature to help with basic router security. You can set a time after which all users in that class are automatically logged out if they have not typed anything at the keyboard. (By default, a user can remain logged in indefinitely.) Here, the users in the class we created will be automatically logged out if the keyboard is idle for five minutes:

```

[edit system login]
aviva@router1# set class operator-plus-read-config permissions idle-timeout 5

```

Warning messages are displayed beforehand:

```

aviva@router1> show system users
 9:56PM up 18:48, 2 users, load averages: 0.16, 0.09, 0.04
USER  TTY  FROM                LOGIN@  IDLE WHAT
aviva  p0   server.juniper.net  9:42PM  4 cli

```

```

aviva@router1> Warning: session will be closed in 1 minute if there is no activity
Warning: session will be closed in 10 seconds if there is no activity
Idle timeout exceeded: closing session
Connection closed by foreign host.

```

As if all this control weren't enough, you can also control, down to the specific command and configuration hierarchy level, what commands users in a particular login class can and cannot issue and what portions of the configuration they can view and modify. For example, you can create a class that has the standard operator permissions but also can issue the request system support command to collect information to send when reporting a problem with the router:

```

[edit system login]
aviva@router1# set class operator-plus-support permissions [ clear network reset trace view ]
aviva@router1# set class operator-plus-support allow-commands "request support information"

```

Or you can take the basic operator class and modify it so users can issue all clear commands except clear system commit (which clears pending configuration commit operations) and clear system reboot (which clears pending router reboots):

```

[edit system login]

```

```
aviva@router1# set class operator-plus-support permissions [ clear network reset
trace view ]
aviva@router1# set class operator-plus-support deny-commands "clear system"
```

Parallel statements allow you to finetune what portions of the configuration can be edited or viewed in configuration mode. This is a way to lock portions of the configuration. The following command does not allow users to modify the protocols portion of the configuration:

```
[edit system login]
aviva@router1# set class all-but-protocols permissions [ all ]
aviva@router1# set class all-but-protocols deny-configuration "protocols"
```

A user in this permission class can edit all portions of the configuration except for the [edit protocols] section:

```
[edit]
aviva2@router1# edit protocols
                        ^
syntax error, expecting <statement> or <identifier>.
```

2.11 Creating a Privilege Class that Hides Encrypted Passwords

Problem

You need to have all permissions on the router, but you don't want to have all of the encrypted passwords displayed.

Solution

Create a new class that explicitly includes all the permission bits except for control and secret:

```
[edit system login]
aviva@router1# set class power-user permissions [ admin admin-control clear configure
field floppy interface interface-control network reset routing routing-control shell
snmp snmp-control system system-control trace trace-control view maintenance firewall
firewall-control secret-control rollback security security-control access access-
control view-configuration ]
```

Discussion

Many network operators like to trim shared secrets and other encrypted data out of their configurations before sharing the configurations with others. The JUNOS software uses the secret permission bit to control viewing access to the passwords, and the secret-control permission bit to control setting them. This recipe still allows shared secrets and passwords to be set on the router, but the values are not shown, copied, or saved (using the configuration mode save command) by the user during normal operations.

Password and secret settings are, of course, still preserved with the commit operation, however, and the full configuration with secret data included is still accessible to the user by virtue of the maintenance permissions.

2.12 Setting Up RADIUS User Authentication

Problem

You use RADIUS for user authentication in your network, and you want to set up the router to authenticate against the RADIUS server.

Solution

Configure information about your RADIUS server:

```
[edit system]
aviva@router1# set radius-server 192.168.63.10 secret $1991poppI
aviva@router1# show
radius-server {
    192.168.63.10 secret "$9$90m6A01EcyKWLhcYgaZji"; ## SECRET-DATA
}
```

Discussion

The Remote Authentication Dial-In User Service (RADIUS) provides a centralized method for authenticating users on the router. RADIUS uses a client-server model. A RADIUS server receives user connection requests, authenticates the user, and returns all configuration information necessary for the client—in this case, the router—to deliver service to the user. All transactions between the server and the client are authenticated by a password called a shared secret.

To configure the router as a RADIUS client, you set the IP address of your RADIUS server and the password (secret) that the router should use to access the server. The secret on the router and the RADIUS server must be the same. After you type the secret, the CLI never displays it, but shows it in a pseudo-encrypted format. The show output is a simple obfuscation to prevent someone from reading the password over your shoulder.

By default, the JUNOS software sends authentication requests to UDP port 1812 on the RADIUS server, as defined in RFC 2865. Also by default, the router waits three seconds to receive a response from the RADIUS server and, if it doesn't hear from the server, tries three more times to connect. You can modify these values if necessary. Here, we allow just 1 retry and wait 10 seconds to receive a response from the server:

```
[edit system]
aviva@router1# set radius-server 192.168.63.10 retry 1
aviva@router1# set radius-server 192.168.63.10 timeout 10
```

If you use a centralized server, it represents a single point of failure if it should go down. To provide redundancy, you can configure several servers:

```
[edit system]
aviva@router1# set radius-server 192.168.0.23 secret 2lip123
aviva@router1# set radius-server 10.0.16.1 secret 883roZe
```

When you configure more than one server, initially the primary server is the one you configured first. After that, the primary server is the one that last responded. If the router cannot reach this server, it tries the remaining ones in the order configured. Use the show command to see the order in which the router tries the servers:

```
[edit system]
aviva@router1# show
radius-server {
  192.168.63.10 secret "$9$vs0W7-oJGiqm24fzF3AtKvWL7V"; ## SECRET-DATA
  10.0.16.1 secret "$9$4DojHQFnCp0TzIcrKXxs2"; ## SECRET-DATA
  192.168.0.23 secret "$9$7edYgq.5QF/iktuB1hcwY2"; ## SECRET-DATA
}
```

Notice that this example specifies different secrets for each server to improve network security. If you suspect that the password of the primary server has been compromised, you can switch to one of the secondary servers.

The JUNOS software defines vendor-specific RADIUS attributes, which are included in packets sent to the RADIUS server. You can configure your server to interpret the Juniper-specific information (see Table 2-2). The Juniper Networks vendor ID is 2636. All the Juniper attributes are used only in RADIUS Access-Accept packets.

Table 2-2. Juniper-specific RADIUS attributes

Attribute name	Description	Type field value	Length field value	String
Juniper-Local-User-Name	Name of user template.	1	3 or more	One or more ASCII octets.
Juniper-Allow-Commands	Allows user to run operational mode commands in addition to those authorized by the user's login class. Same action as the allow-command statement.	2	3 or more	One or more ASCII octets written as an extended regular expression.
Juniper-Deny-Commands	Disallows user to run operational mode commands authorized by the user's login class. Same action as the deny-command statement.	3	3 or more	One or more ASCII octets written as an extended regular expression.
Juniper-Allow-Configuration	Allows the user to modify portions of the configuration in addition to those authorized by the user's login class. Same action as the allow-statement statement.	4	3 or more	One or more ASCII octets written as an extended regular expression.
Juniper-Deny-Configuration	Disallows user to modify portions of the configuration in addition to those authorized by the user's login class. Same action as the deny-statement statement.	5	3 or more	One or more ASCII octets written as an extended regular expression.

See Also

RFC 2865, *Remote Authentication Dial In User Service (RADIUS)*

2.13 Setting Up TACACS+ User Authentication

Problem

You want to use a TACACS+ server to authenticate people who log in to the router.

Solution

Configure information about your TACACS+ server:

```
[edit system]
aviva@router1# set tacacs-server 192.168.62.10 secret $1991poppI
aviva@router1# show
tacacs-server {
    192.168.62.10 secret "$9$90m6A01EcyKWLhcYgaZji"; ## SECRET-DATA
}
```

Discussion

TACACS+ is a newer version of the older TACACS authentication software. Like RADIUS, TACACS+ uses a client-server model, with the router being the client. All transactions between the server and the client are authenticated by a shared secret.

The JUNOS configuration for TACACS+ is almost identical to that for RADIUS. You set the IP address of your TACACS+ server and the password (secret) that the router should use to access the server. The secrets on the router and the server must match. For redundancy, you can configure multiple servers.

There are also JUNOS-specific TACACS+ attributes that you can configure on the TACACS+ server. These attributes are named `local-user-name`, `allow-commands`, `deny-commands`, `allow-configuration`, and `deny-configuration`, and have the same description, length, and string as the parallel RADIUS attributes (see Table 2-2).

2.14 Restricting Inbound SSH and Telnet Access

Problem

You want to allow SSH and Telnet access to the router, but you want to restrict the access to make the router more secure.

Solution

Add a term to an existing firewall filter that restricts SSH and Telnet access:

```
[edit firewall filter protect-RE]
```

```
aviva@RouterF# set term ssh-telnet from source-address 10.0.8.0/24
aviva@routerF# set term ssh-telnet from destination-port [ ssh telnet ]
aviva@RouterF# set then accept
```

Also include a term at the end of the filter to reject access attempts from any other subnets:

```
[edit firewall filter protect-RE]
aviva@RouterF# set term allow-nothing-else then count reject-counter
aviva@RouterF# set term allow-nothing-else then log
aviva@RouterF# set term allow-nothing-else then syslog
aviva@RouterF# set term allow-nothing-else then reject
```

For the filter to affect incoming traffic, apply it to the desired interfaces:

```
[edit interfaces]
aviva@RouterF# set lo0 unit 0 family inet filter input protect-RE
```

Discussion

SSH and Telnet are two very common ways to access the router. However, SSH brute-force attempts to guess passwords are a very common way to try to compromise routers, and Telnet connections are not very secure. To protect the router, you should restrict the systems from which people can use SSH and Telnet to the router. Even though Telnet is not a secure access method, you may want to allow it because your network management tools use Telnet to access routers, and it is more of a hassle to change the access method than to just allow restricted access. As much as possible, you should lock down access to all management services on the routers and on any other systems in your network to maintain tight security. You can never be sure that people are not using passwords that are easy to guess, and Telnet sends passwords over the network in clear text so they could easily be sniffed. Restricting inbound access to the router also protects against potentially unknown vulnerabilities with SSH and Telnet.

You restrict Telnet access using a firewall filter. This recipe shows a single term in a firewall filter that acts on all TCP traffic whose destination port is the Telnet port (port 23), and it accepts Telnet connections only from the 10.0.0.0/8 subnet and rejects all other connection attempts. Each interface can have one inbound firewall filter, so you include the term shown in this recipe in the complete firewall filter that you apply on an incoming interface.

It's important to note that if you use the term in the recipe as the only filter on an interface, it will block all traffic to the Routing Engine except for Telnet from subnet 10.0.0.0/8. This means that SNMP, OSPF, IS-IS, PIM, and BGP will all be blocked. Make sure you include this term as part of a longer firewall filter.

You have to decide where in the filter to place the term. Because the terms in the firewall filter are evaluated in the order in which they appear, the placement affects the efficiency of the filter. Generally, terms for operations that need to be performed quickly, such as BGP peering, and IGP and DNS traffic, are at the beginning of the

filter. For operations that are less time-critical, including processing Telnet connections, place the term toward the end of the filter. In the incoming firewall filter `incoming-to-me` shown in Recipe 9.11, you might place the Telnet term almost at the end, just before the `allow-nothing-else` term.

Then apply the filter to the desired interfaces. Here we apply the filter to the `100` interface because we want it to apply to all traffic destined to the router's management addresses, even traffic that is coming to the address of one of the network (PIC) interfaces.

The term in this recipe is just one of several terms in a single firewall filter. As a general point, you rarely just reject a firewall term without also either logging, syslogging, or running a counter on the rejections (which gives you data that you can graph). Tracking the rejections is useful for showing abuse of your router, attacks on the router, or even misconfigurations. For example, if you forget about an automated process that uses Telnet to read configuration information on your router that comes from `192.168.0.0/16` and you only permit `10.0.0.0/8`, then the then `syslog` action (and appropriate `syslog` configuration statements) can be very handy for resolving issues.

See Also

Recipes 9.8, 9.11, 9.12, and 9.13

2.15 Setting the Source Address for Telnet Connections

Problem

You want to force Telnet to use a specific IP address when connecting from the router to another system.

Solution

Include the source address in the Telnet command:

```
aviva@RouterA> telnet source 172.19.121.15 server1
Trying 172.19.121.246...
Connected to server1.mycompany.com.
Escape character is '^]'.
server1 (tty0)
login:
```

Discussion

By default, the source address included in locally generated Telnet and other TCP/IP packets is the address of the interface on which the Telnet request is sent. This

means the source address may change from connection to connection. If multiple equal-cost next hops are present for a destination, the lo0 loopback interface address is used as the source address. If you configure the system default-address-selection statement in the configuration, which uses the lo0 interface address as the router's system address, this address is used as the source for most Telnet connections (see Recipe 7.4).

The result of this behavior is that the default Telnet source address is not always the same and not always deterministic. If the source address matters when using Telnet to access another system, include it in the telnet command. One instance to do so is when filtering the source address on incoming connections, which may block packets coming to the default source address. Another instance is when using Telnet as a generic way to check and troubleshoot other TCP ports (such as connecting to a server on port 25 to see if it is listening for SNMP mail connections). The source address that you specify must be an address that's configured on the router.

See Also

Recipes 7.4 and 7.12

2.16 Creating a Login Banner

Problem

You want to display a banner during login that indicates that the router is for authorized users only.

Solution

You define a login banner:

```
[edit system login]
aviva@router1# set message "\n\
n=====Access to this device
is limited to authorized users only.\n\n  WARNING: All unauthorized access is
prohibited.\n\n=====
=====\\n\\n"
```

Discussion

A login banner is displayed each time anyone logs in to the router, before the login prompt:

```
aviva-server> telnet router1
=====
Access to this device is limited to authorized users only.
  WARNING: All unauthorized access is prohibited.
=====
router1 (ttyp0)
```

login:

It may seem rather trivial to set a login banner, and you may wonder what this has to do with router security because it doesn't do anything to restrict access to the router. Although this is true, having a login banner is good practice for legal protection of your router. From a legal point of view, you want to warn unauthorized users that they are not permitted to use the router, and you want to do so with a strongly worded message, as we've shown here. While you might think that you want to welcome users to the router, you should not use the word "welcome" or any similar words in the login banner.

You can also have a login message that is displayed after users log in to the router:

```
[edit system]
aviva@router1# set announcement "Reminder: maintenance window schedule at 0200 UTC"
```

These messages are a way to remind authorized users of network or router issues:

```
aviva-server1%> telnet router1
router1 (tty0)
login: aviva
password: *****
--- JUNOS 7.4R1.7 built by builder on 2005-10-23 02:03:58 UTC
Reminder: maintenance window schedule at 0200 UTC
aviva@router1>
```

2.17 Finding Out Who Is Logged in to the Router

Problem

You are logged in to the router and you want to see who else is logged in.

Solution

Use the `show system users` command to see who is logged in to the router:

```
aviva@router1> show system users
 9:10PM up 11 hrs, 2 users, load averages: 0.00, 0.00, 0.00
USER  TTY  FROM                LOGIN@  IDLE WHAT
mike  p0   server1.juniper.net  9:09PM - -tssh (csh)
aviva p1   server1.juniper.net  8:42PM - cli
```

Discussion

More than one person can log in to the router at one time. Each person can perform various operations, from viewing router statistics, to rebooting the router and changing the router's configuration. Once you access the router, you might also want to see who else is working on the router. You display who is logged in using the `show system users` command, which is basically the same as the Unix `w` command. This command shows the username, the terminal number through which they are connected, the server they have logged in from, when they logged in, how long they have

been idle, and what they are doing. The output in this recipe shows that the user `mike` is working in the Unix shell on the router and the user `aviva` is working in the CLI.

As you are logging in to the router, if others are logged in, the CLI does not display any messages. However, when you enter configuration mode, the CLI indicates that another user is also configuring the router:

```
aviva@router1> configure
Entering configuration mode
Users currently editing the configuration:
  mike terminal p2 (pid 5465) on since 2005-04-06 21:30:42 UTC
    [edit class-of-service scheduler-maps]
The configuration has been changed but not committed
```

Here you see that the user `mike` is also working in configuration mode and has made changes to the class of service portion of the configuration.

See Also

Recipe 2.19

2.18 Logging Out of the Router

Problem

You are done using the router and want to log out so that no one can sit down at your terminal and access the router.

Solution

Log out of the router:

```
aviva@router1> exit
[server1.mycompany.com] aviva%
```

Discussion

One of the simplest and most obvious ways to protect the security of the router is to log out when you have no reason to be logged in or when you have to step away from your terminal for a few minutes. You must be in operational mode to log out. If you are in configuration mode, exit from it first:

```
[edit]
aviva@router1# exit
aviva@router1> exit
aviva@server1%
```

When you are not at the top level of configuration mode, you can go there before exiting:

```
[edit snmp v3 vacm]
```

```
aviva@router1# top
[edit]
aviva@router1# exit
aviva@router1> exit
aviva@server1%
```

You can also exit directly from a lower level in the hierarchy:

```
[edit snmp v3 vacm]
aviva@router1# exit configuration-mode
aviva@router1> exit
aviva@server1%
```

By default, a user can remain logged in to the router for an unlimited amount of time when your login session is idle. You can limit the time by setting an idle timeout value for each login privilege class (see Recipe 2.10).

2.19 Forcibly Logging a User Out

Problem

Someone is logged in to the router who shouldn't be and you need to log them out.

Solution

Forcibly log the user out:

```
aviva@router1> request system logout user mike
```

Discussion

There are a number of situations when someone is logged in to the router and they shouldn't be. A user may have walked away from the terminal or may be in configuration mode when you must change the configuration to deal with a problem situation.

Use the `show system users` command to list who is logged in to the router:

```
aviva@router1> show system users
5:20AM up 81 days, 6:41, 1 user, load averages: 0.00, 0.00, 0.00
USER  TTY  FROM                LOGIN@  IDLE WHAT
mike  p0   172.10.28.108       5:20AM  4:07 -cli (cli)
aviva p1   172.10.28.107       2:06AM  - -cli (cli)
```

If another user is in configuration mode, you see their username in the message displayed when you enter configuration mode.

If you are logged in as root or have root privileges, you can forcibly log a user out of the router:

```
aviva@router1> request system logout user mike
```

Mike would be logged out:

```
mike@router1> Connection closed by foreign host.
```

```
[server1.mycompany.com] mike%
```

You can send the user a message beforehand using a command similar to the Unix `write` utility:

```
aviva@router1> request message user mike message "log out immediately"
```

You can send the message to a particular user, as we've done here, or to all logged-in users (similar to the Unix `wall` utility):

```
aviva@router1> request message all message "log out immediately"
```

You can also specify the terminal (TTY) to forcibly log out a user:

```
aviva@router1> request system logout terminal p0
```