

6

Passwords and Security

Passwords and security are of utmost importance for the security of your system. This chapter provides the information you need to configure your ERX system to be secure for all levels of users.

Topic	Page
Overview	6-1
Setting Basic Password Parameters	6-2
Setting and Erasing Passwords	6-5
Vty Line Authentication	6-11
Virtual Terminal Access Lists	6-16
Secure System Administration with SSH	6-16
Restricting User Access	6-27

Overview

One of your major management responsibilities is to secure your system. To do this, assign passwords or secrets to the system. In Global Configuration mode, you can set passwords or secrets to prevent unauthorized users from accessing the system in Privileged Exec mode.

Passwords and secrets have the same degree of security on your system, and they are used interchangeably. You can define either a password or a secret for your system, but not both.

Setting Basic Password Parameters

This section shows how to set up basic passwords and secrets on your system. You cannot create your own encrypted passwords and secrets. You must use encrypted passwords and secrets that the system generates.



Note: See *Setting and Erasing Passwords* later in this chapter for additional commands for erasing and monitoring passwords.

Creating Encrypted Passwords

This example encrypts password `timeout1` and creates a password for privilege level 10.

- 1 Enable and configure the password. The **0** keyword specifies that you are entering an unencrypted password.

```
host1(config)#enable password level 10 0 timeout1
```

- 2 Display the encrypted password.

```
host1(config)#exit
host1#show secrets
```

Current Password Settings

level	encryption type	encrypted password/secret	mode
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
10	7 (password)	dq]XG`,%N"SS7d}o)_?Y	configured

You or users with high privilege levels can now use the encrypted password, `dq]XG`,%N"SS7d}o)_?Y`, with the **password** command.

Creating Secrets

This example generates a secret for the password *rocket*, and creates a secret for privilege level 15.

- 1 Enable and configure the secret. The **0** keyword specifies that you are entering an unencrypted secret.

```
host1(config)#enable secret level 15 0 rocket
```

- 2 Display the secret.

```
host1(config)#exit
host1#show secret
```

```

                                Current Password Settings
                                -----
                                encryption      encrypted
                                type            password/secret      mode
                                -----
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15      5 (secret)  bcA"+1aeJD8)/[1ZDP6  configured

```

You or users with high privilege levels can now use the encrypted password, *bcA"+1aeJD8)/[1ZDP6*, with the **password** command.

Encrypting Passwords in Configuration File

You can also direct the system software to encrypt passwords saved in the configuration file by using the **service password-encryption** command. This command is useful to keep unauthorized individuals from viewing your password in your configuration file. It is important to remember that this command uses a simple cipher and is not intended to protect against serious analysis. You can tell if a string is encrypted if it is preceded by an 8.

Commands and Guidelines

Use the following commands and guidelines to set passwords or secrets for the privilege levels.

enable password

- Use to set a password, which controls access to Privileged Exec mode and some configuration modes.
- Enter the password in plain text (unencrypted) or cipher text (encrypted). In either case, the system stores the password as encrypted.
- The first time you define a password, you must enter it in plain text. To view its encrypted form, use the **show config** display. To redefine the password at a later date, you can enter the password in its encrypted form.
- You can use the following keywords:
 - › **0** (zero) – specifies an unencrypted password
 - › **7** – specifies an encrypted password
- Example 1 (unencrypted password)

```
host1(config)#enable password 0 mypassword
```
- Example 2 (encrypted password)

```
host1(config)#enable password 7 x13_2
```
- Use the **no** version to remove the password.

erase secrets

- Use to set a secret, which controls access to the Privileged Exec mode and some configuration modes.
- Enter the secret in plain text (its unencrypted form) or cipher text (its encrypted form). In either case, the system stores the secret as encrypted.
- The first time you define a secret, you must enter it in plain text. To view its encrypted form, use the **show config** display. To redefine the secret at a later date, you can enter the secret in its encrypted form.
- You can use the following keywords:
 - › **0** (zero) – specifies an unencrypted secret
 - › **5** – specifies an encrypted secret
- Example 1 (unencrypted secret)

```
host1(config)#enable secret 0 yalta45
```
- Example 2 (encrypted secret)

```
host1(config)#enable secret 5 y13_x
```
- Use the **no** version to remove the secret.

service password-encryption

- Use to encrypt passwords that are saved in the system's configuration file. The command converts plain text to cipher text. The default is no encryption.
- Use of this command prevents casual observers from viewing passwords, for example, in data obtained from **show config** displays. The command is not intended to provide protection from serious analysis.
- This command does NOT apply to passwords set with **enable secret**, **enable password**, or **password** (Line Configuration mode).
- This command does apply to authentication key passwords and BGP neighbor passwords.
- Example

```
host1(config)#service password-encryption
```
- Use the **no** version to remove the encryption assignment.

Setting and Erasing Passwords

You can set the following passwords:

- Enable passwords that control access to different groups of commands.
- A console password that controls access to the console.
- Passwords for individual vty lines or groups of vty lines.

Privilege Levels

Different groups of commands are associated with privilege levels (Table 6-1). You can set enable passwords to allow users to access commands at different privilege levels.

Table 6-1 Commands available at different privilege levels

Privilege Level	Commands Available
0	help , exit , enable , and disable commands
1	User Exec commands plus commands at level 0
5	Privileged Exec show commands plus commands at levels 0 and 1
10	All commands except support commands
15	Support commands that Juniper Technical Support may provide and all other commands

To maximize security and usability, set different passwords for levels 1, 5, 10, and 15. By default, no enable passwords exist.

Accessing Privilege Levels

If users have access to the console, they automatically have access to privilege level 0. To access higher levels of privilege, they must enter the **enable** *privilege-level* command. When users specify a privilege level, the system checks to see if there is a password at that level. If there is not, the system prompts the user for the password for the lower level closest to the requested level.

Setting Enable Passwords

To set up enable passwords, use the commands described in *Setting Basic Password Parameters* earlier in this chapter.

Erasing Enable Passwords

If you forget an enable password or secret, you can erase all enable passwords and secrets.

Two commands allow you to erase passwords and secrets: **erase secrets** and **service unattended-password-recovery**. It is important to fully understand the purpose of these commands and how they work with each other.

The **erase secrets** command can be used to delete all existing passwords. To use this command, you must be physically present at the router to complete the operation. Once the command is executed, you have a finite number of seconds to press the software reset button on the SRP module. You can execute this command from the console or any vty.

The **service unattended-password-recovery** command provides you with a way to delete existing passwords and secrets without physically being present at the router. You must have the proper privilege level to execute the command, and you can execute it from either the console or any vty.

When you execute **service unattended-password-recovery**, you change the behavior of **erase secrets**. You can now delete passwords and secrets from the console by executing **erase secrets** without a time restraint or having to be physically present at the router. When you use the **no** version of **service unattended-password-recovery**, you revert the functionality of erase secrets to the factory default setting.

To erase all enable passwords or secrets:

- 1 Log in to the system.
- 2 Erase the existing enable password or secret. Specify the number of seconds to allow for the erase operation.

```
host1>erase secrets 60
```

- 3 Within the time limit that you specified for the **erase secrets** command, press the recessed software reset button on the primary SRP module (see Figure 6-1).

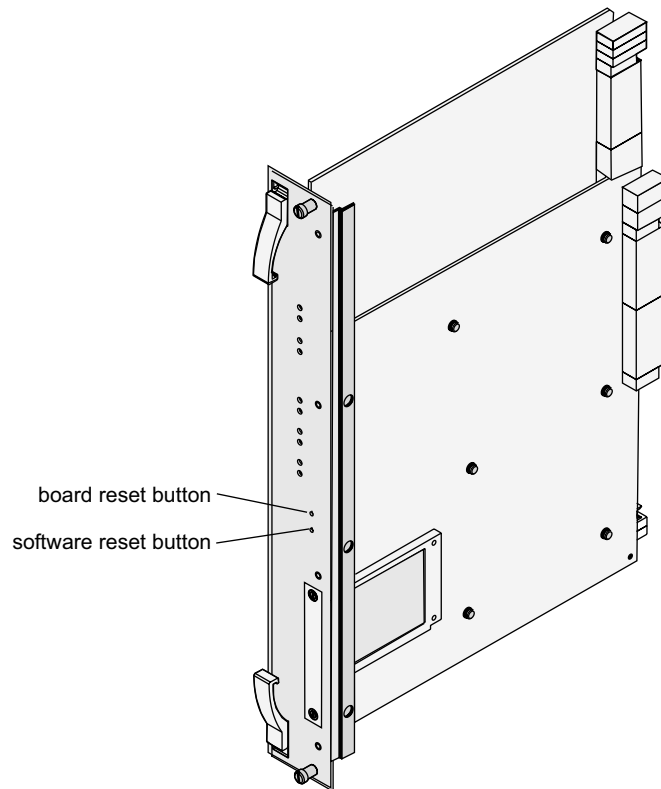


Figure 6-1 Location of the software reset button



Note: If you do not press the software reset button within the time limit, the system will not erase the password, and you will need to repeat the process.

erase secrets

- Use to delete all CLI passwords and secrets.
- After you issue this command, press the software reset button (see Figure 6-1) within the time you specify for this command.

- Allows you to set the number of seconds (1–60) for this procedure to be accomplished.
- Allows you to set a new password when you have forgotten your password.
- Can be used with the **service unattended password-recovery** command.
- Example

```
host1>erase secrets 60
```
- There is no **no** version.

service unattended password-recovery

- Use to allow you to delete all passwords and secrets from the console without being physically present at the router.
- When executed, this command changes the behavior of the **erase secrets** command, which will not take any parameters and will not be available through a vty session.
- Example

```
host1(config)#service unattended password-recovery
```
- Use the **no** version to revert **erase secrets** to factory default settings.

Setting a Console Password

By default, there is no console password. To set a console password:

- 1 Make sure that you know the enable password for the system.

If you need to reset the enable password, see *Privilege Levels* earlier in this chapter.

- 2 Access Privileged Exec mode, and enter the enable password if prompted.
- 3 Access Global Configuration mode.
- 4 Access Line Configuration mode.

```
host1(config)#line console 0
```

- 5 Enable password checking at login.

```
host1(config-line)#login
```

- 6 Specify a password.

```
host1(config-line)#password 7 dq]XG`,%N"SS7d}o)_?Y
```

line

- Use to specify the vty lines or the console.
- Example


```
host1(config)#line vty 1 4
```
- Use the **no** version to remove a vty line or a range of lines from your configuration; users will not be able to run Telnet, SSH, or FTP to lines that you remove. When you remove a vty line, the system removes all lines above that line. For example, **no line vty 6** causes the system to remove lines 6 through 19. You cannot remove lines 0 through 4.

login

- Use to enable password checking at login.
- The default setting is to enable a password.
- Example


```
host1(config)#line vty 1 4
host1(config-line)#login
```
- Use the **no** version to disable password checking and allow access without a password.

password

- Use to specify a password on the console, a line or a range of lines.
- If you enable password checking, but do not configure a password, the system will not allow you to access virtual terminals.
- Use the following keywords to specify the type of password you will enter:
 - › **0** (zero) – unencrypted password
 - › **5** – secret
 - › **7** – encrypted password



Note: To use an encrypted password or a secret, you must follow the procedure in *Setting Basic Password Parameters* earlier in this chapter to obtain the encrypted password or secret. You cannot create your own encrypted password or secret; you must use a system-generated password or secret.

- Example 1 (unencrypted password)


```
host1(config-line)#password 0 mypassword
```
- Example 2 (secret)


```
host1(config-line)#password 5 bcA"+1aeJD8)/[1ZDP6
```
- Example 3 (encrypted password)


```
host1(config-line)#password 7 dq]XG`,%N"SS7d}o)_?Y
```
- Use the **no** version of this command to remove the password. By default, no password is specified.

Erasing the Console Password

If you forget the console password, you can erase the existing value and configure a new one. This action deletes all authentication for the console line. To erase existing passwords:

- 1 Reboot the system by pressing the recessed software reset button on the primary SRP module (see Figure 6-1) and then pressing the <mb> key sequence during the countdown.

- 2 Disable authentication at the console level.

```
:boot##disable console authentication
```

If you remember the password at this point, you can override this action by entering:

```
:boot##no disable console authentication
```

- 3 Reload the operating system.

```
:boot##reload
```

When the operating system reloads, you can access the console without a password.



Note: You will be able to log in to the console without a password until you set a new password.

Monitoring Passwords

You can use the **show secrets** command to view all current passwords and secrets.

show secrets

- Use to display all passwords and secrets.
- Passwords and secrets appear in their encrypted form.
- In the mode column, *inherited* indicates whether a secret was inherited from a lower password level. The **show config** command displays only secrets configured by the user; it does not display inherited secrets.
- Example

```
host1#show secrets
                        Current Password Settings
                        -----
level      encryption      encrypted      mode
-----      -
0
1
2
```

```

3
4
5      7 (password)  zRFj_6>^]10kZR@e!|S$  configured
6      7 (password)  zRFj_6>^]10kZR@e!|S$  inherited
7      7 (password)  zRFj_6>^]10kZR@e!|S$  inherited
8      7 (password)  zRFj_6>^]10kZR@e!|S$  inherited
9      7 (password)  zRFj_6>^]10kZR@e!|S$  inherited
10     7 (password)  zRFj_6>^]10kZR@e!|S$  inherited
11     7 (password)  zRFj_6>^]10kZR@e!|S$  inherited
12     7 (password)  zRFj_6>^]10kZR@e!|S$  inherited
13     7 (password)  zRFj_6>^]10kZR@e!|S$  inherited
14     7 (password)  zRFj_6>^]10kZR@e!|S$  inherited
15     7 (password)  zRFj_6>^]10kZR@e!|S$  inherited

```

Vty Line Authentication

The system supports 20 virtual tty (vty) lines for Telnet, Secure Shell Server (SSH) and FTP services. Each Telnet, SSH, or FTP session requires one vty line. You can add security to your system by configuring the software to validate login requests. There are two modes of authentication for a vty line:

- Simple authentication – password-only authentication via the local configuration
- AAA authentication – username and password authentication via a set of authentication servers

Configuring Simple Authentication

To configure simple authentication:

- 1 Specify a vty line or a range of vty lines on which you want to enable the password.

```

host1(config)#line vty 8 13
host1(config-line)#

```

- 2 Specify the password for the vty lines.

```

host1(config-line)#password 0 mypassword

```

- 3 Enable login authentication on the lines.

```

host1(config-line)#login

```

- 4 Display your vty line configuration.

```

host1#show line vty 8
no access-class in

```

```

data-character-bits 8
exec-timeout never
exec-banner enabled
motd-banner enabled
login-timeout 30 seconds

```

line

- Use to specify the vty line(s) on which you want to enable the password.
- You can set a single line or a range of lines. The range is 0–19.
- Example

```

host1(config)#line vty 8 13

```

- Use the **no** version to remove a vty line or a range of lines from your configuration; users will not be able to run Telnet, SSH, or FTP to lines that you remove. When you remove a vty line, the system removes all lines above that line. For example, **no line vty 6** causes the system to remove lines 6 through 19. You cannot remove lines 0 through 4.

login

- Use to enable password checking at login.
 - The default setting is to enable a password.
 - Example
- ```

host1(config-line)#login

```
- Use the **no** version to disable password checking and allow access without a password.

**password**

- Use to specify a password on a single line or a range of lines.
- If you enable password checking but do not configure a password, the system will not allow you to access virtual terminals.
- Specify a password in plain text (unencrypted) or cipher text (encrypted). In either case, the system stores the password as encrypted.
- Use the following keywords to specify the type of password you will enter:
  - › **0** (zero) – unencrypted password
  - › **5** – secret
  - › **7** – encrypted password



**Note:** To use an encrypted password or a secret, you must follow the procedure in *Setting Basic Password Parameters* earlier in this chapter to obtain the encrypted password or secret. You cannot create your own encrypted password or secret; you must use a system-generated password or secret.

- Example 1 (unencrypted password)

```

host1(config-line)#password 0 mypassword

```

- Example 2 (secret)

```

host1(config-line)#password 5 bcA"+;+1aeJD8)/[1ZDP6

```

- Example 3 (encrypted password)
 

```
host1(config-line)#password 7 dq]xG~,%N"SS7d}o)_?Y
```
- Use the **no** version to remove the password. By default, **no password** is specified.

### ***show line vty***

- Use to display the configuration of a vty line.
- Field descriptions
  - › access-class – access-class associated with the vty line
  - › data-character-bits – number of bits per character
    - 7 – setting for the standard ASCII set
    - 8 – setting for the international character set
  - › exec-timeout – time interval that the terminal waits for expected user input
    - Never – indicates that there is no time limit
  - › exec-banner – status for the exec banner: enabled or disabled. This banner is displayed by the CLI after user authentication (if any) and before the first prompt of a CLI session.
  - › motd-banner – status for the MOTD banner: enabled or disabled. This banner is displayed by the CLI when a connection is initiated.
  - › login-timeout – time interval during which the user must log in.
    - Never – indicates that there is no time limit
- Example
 

```
host1#show line vty 0
no access-class in
data-character-bits 8
exec-timeout 3w 3d 7h 20m 0s
exec-banner enabled
motd-banner enabled
login-timeout 30 seconds
```

### *Configuring AAA Authentication*

Before you configure AAA authentication, you need to configure a RADIUS authentication server.

To configure AAA new model authentication for inbound sessions to vty lines on your system:

- 1 Specify AAA new model authentication.

```
host1(config)#aaa new-model
```

- 2 Create an authentication list that specifies the type(s) of authentication methods allowed.

```
host1(config)#aaa authentication login my_auth_list radius
line none
```

- 3 Specify the range of vty lines.

```
host1(config)#line vty 6 10
host1(config-line)#
```

- 4 If you specified that a password is required in step 2, specify a password for the vty lines.

```
host1(config-line)#password xyz
```

- 5 Apply the authentication list to the vty lines.

```
host1(config-line)#login authentication my_auth_list
```

### ***aaa authentication login***

- Use to create a list that specifies the methods of authentication.
- Once you specify AAA new model as the authentication method for vty lines, an authentication list called “default” is automatically assigned to the vty lines. To allow users to access the vty lines, you must create an authentication list and either:
  - › Name the list “default.”
  - › Assign a different name to the authentication list and assign the new list to the vty line using the **login authentication** command.
- You can enter up to three authentication methods in an authentication list.
- The system traverses the list of authentication methods to determine if a user is allowed to start a Telnet session. If a specific method is available but the user information is not valid (such as an incorrect password), the system does not continue to traverse the list and denies the user a session.
- If a specific method is unavailable, the system continues to traverse the list. For example, if **radius** is the first authentication type element on the list and the RADIUS server is unreachable, the system attempts to authenticate with the next authentication type on the list.
- The system assumes an implicit denial of service if it reaches the end of the authentication list without finding an available method.
- Example

```
host1(config)#aaa authentication login my_auth_list radius
line none
```

- Use the **no** version to remove the authentication list from your configuration.

### ***aaa new-model***

- Use to specify AAA new model as the authentication method for the vty lines on your system.
- If you specify AAA new model and you do not create an authentication list, users will not be able to access the system through a vty line.
- Example

```
host1(config)#aaa new-model
```

- Use the **no** version to restore simple authentication.

## **line**

- Use to specify the virtual terminal lines.
- You can set a single line or a range of lines. The range is 0–19.
- Example
 

```
host1(config)#line vty 6 10
```
- Use the **no** version to remove a vty line or a range of lines from your configuration; users will not be able to run Telnet, SSH, or FTP to lines that you remove. When you remove a vty line, the system removes all lines above that line. For example, **no line vty 6** causes the system to remove lines 6 through 19. You cannot remove lines 0 through 4.

## **login authentication**

- Use to apply an authentication list to the vty lines you specified on your system.
- Example
 

```
host1(config-line)#login authentication my_auth_list
```
- Use the **no** version to specify that the system should use the default authentication list.

## **password**

- Use to specify a password on a line or a range of lines if you specified the line option with the **aaa authentication login** command.
- If you enable password checking but do not configure a password, the system will not allow you to access virtual terminals.
- Use the following keywords to specify the type of password you will enter:
  - › **0** (zero) – unencrypted password
  - › **5** – secret
  - › **7** – encrypted password



**Note:** To use an encrypted password or a secret, you must follow the procedure in *Setting Basic Password Parameters* earlier in this chapter to obtain the encrypted password or secret. You cannot create your own encrypted password or secret; you must use a system-generated password or secret.

- Example 1 (unencrypted password)
 

```
host1(config-line)#password 0 mypassword
```
- Example 2 (secret)
 

```
host1(config-line)#password 5 bcA"+1aeJD8)/[1ZDP6
```
- Example 3 (encrypted password)
 

```
host1(config-line)#password 7 dq]XG`,%N"SS7d}o)_?Y
```
- Use the **no** version to remove the password. By default, **no password** is specified.

## Virtual Terminal Access Lists

---

You can provide additional security for your system by using access lists to restrict access to vty lines.

When the system attempts to authenticate a user, it always selects the first vty line that has an access class that permits that user's host. The vty line's configuration must authenticate the user to allow access. Otherwise, the user can never gain access. Consequently, it is recommended that you use identical authentication configurations for all vtys that have the same access class list.

To set up access lists:

- Associate the access list with inbound Telnet sessions.

```
host1(config)#line vty 12 15
host1(config-line)#access-class boston in
```

- Configure an access list.

```
host1(config)#access-list boston permit any
```

### ***access-class in***

- Use to associate the access list with vty lines.
- Example – this example sets the virtual terminal lines to which you want to restrict access and specifies an access class to grant access to incoming requests.

```
host1(config)#line vty 12 15
host1(config-line)#access-class boston in
```

- Use the **no** version to remove access restrictions.

### ***access-list***

- Use to configure an access list.
- Example

```
host1(config)#access-list boston permit any
```

- The **no** version of this command removes the access list.

## Secure System Administration with SSH

---

The system supports the SSH protocol version 2 as a secure alternative to Telnet for system administration.



**Note:** Versions earlier than 2.0.12 of the SSH protocol client are not supported. The SSH server embedded within the system recognizes SSH clients that report an SSH protocol version of 1.99, with the expectation that such clients are compatible with SSH protocol version 2.0. Clients that report an SSH protocol

*version of 1.99 apparently do so to determine the protocol version supported by the server.*

SSH provides the following major features:

- Server authentication via a Diffie-Hellman key exchange – Protects against hackers interjecting mimics to obtain your password. You can be confident that you are connected to your own router.
- User authentication – Ensures that the system is allowing connection from a permitted host and remote user.



**Note:** *Digital Signature Standard (DSS) public key user authentication for SSH is not supported. RADIUS password authentication is the only method of user authentication currently supported. It is enabled by default. If RADIUS authentication is disabled, then all SSH clients that pass protocol negotiation are accepted.*

- Data encryption and key-protected hashing – Provides a secure, trustable session to the upper-layer user interface. Encryption provides confidentiality by preventing unauthorized persons from listening in on management traffic. Encryption and hashing ensure data integrity to obstruct man-in-the-middle attacks, where unauthorized persons access messages and modify them without detection.

## *Transport*

The SSH transport layer handles algorithm negotiation between the server and client over TCP/IP. Negotiation begins when the SSH client and server send each other textual information that identifies their SSH version. If they both agree that the versions are compatible, the client and server exchange lists that specify the algorithms that they support for key exchange, encryption, data integrity via a message authentication code (MAC), and compression. Each party sends two lists. One list has the algorithms supported for transmission; the other has the algorithms supported for receipt. The algorithms are specified in order of preference in each list. The client and server use the algorithm for each process that matches the client's highest preference and is supported by the server. If no intersection is found, the negotiation attempt fails and the connection is terminated.

If algorithm negotiation is successful, the server sends its public host key to the client for authentication so the client can be certain it is connected to the intended host rather than to an imposter. The client compares the key to its host key database. The client authenticates the server if the key is found in the database. If the key is not present, then the client can

accept or reject this new, unknown key depending on how you have configured the client. See *Host Key Management* later in this chapter.

When the client authenticates the server's host key, it begins the transport key exchange process by sending the key data required by the negotiated set of algorithms. The server responds by sending its own key data set. If both sides agree that the keys are consistent and authentic, the keys are applied so that all subsequent messages between client and server are encrypted, authenticated, and compressed according to the negotiated algorithms.

### *User Authentication*

User authentication begins after the transport keys are applied. The client typically asks the server which authentication methods it supports. The server responds with a list of supported methods with no preference.

The client specifies a user authentication method. If the chosen method is supported by the server, the client then challenges the user—that is, the client prompts the user for a password or public-key pass phrase. The client sends the challenge response from the user and the username to the server. The server authenticates the user based on this response.

The system software currently supports only RADIUS password authentication, which is enabled by default. The RADIUS server validates the username and password from its database. If user authentication is disabled, then all SSH clients that pass protocol negotiation are accepted.

### *Connection*

The SSH connection layer creates the user session when the user is authenticated. The server waits for a connection request. The system currently supports only shell requests, which the server interprets as a request for a hook into a CLI session. The server ignores any other requests, such as X11 or TCP/IP tunneling.

### *Key Management*

The ERX system implementation of SSH provides for management of user keys and host keys.

#### User Key Management

Key administration is still under development for the server environment.

## Host Key Management

You create a host key for the SSH server with the **crypto key generate dss** command. If a host key already exists, this command replaces it with a new key and terminates all ongoing SSH sessions. Any SSH clients that previously accepted the old host key reject the new key the next time the client and server connect. The client then typically instructs the end user to delete the locally cached host key and to try to connect again.



**Caution:** Use caution issuing the **crypto key generate dss** command from an SSH client. Issuing this command will terminate that SSH session; it will be the last command you send from that session.

The public half of the host key is sent from the server to the client as part of the transport layer negotiation. The client attempts to find a match for this key with one stored locally and assigned to the server. If the client does not find a match, it can accept or reject the key sent from the server. Refer to your client documentation for detailed information. You typically configure the client to do one of the following:

- Never accept an unknown key.
- Always accept an unknown key.
- Query the administrator before accepting an unknown key.

If you do not want the client ever to trust the server when it sends an unknown key, you must manually copy—using the **copy** command—the host key from each server to each intended client. This is the only way to be certain that each client has a local copy of the necessary keys for matching during negotiation.

If you configure the client to accept unknown keys—either automatically or with administrator approval—this acceptance policy applies only to the first time the client receives a key from a particular server. When the SSH client accepts a host key, it stores the key locally and uses it for all future comparisons with keys received from that host. If the client subsequently receives a different key—a new unknown—from that server, it is rejected.

You cannot configure an SSH client to accept a new key after it has accepted a key from an SSH server. You must delete the old key before a new key can be accepted.

## Performance

Generating a host key is computationally intensive and can take up to several minutes depending on the load of the system. The system cannot accept any CLI inputs from that session while it is generating the key.

Encryption, data integrity validation, and compression are all computationally intensive. These features can affect system performance in the following ways:

- Reduce the effective baud rate compared with Telnet or the local CLI. Users are unlikely to notice this performance degradation because user interaction is inherently slow compared with other system operations.
- Increase the general load on the system CPU.

### *Security Concerns*

There are two areas where you might be concerned about security with the current support of SSH:

- Only RADIUS user authentication is supported. If you disable user authentication, all users are accepted if the client and server successfully complete negotiation.
- Because the load on the system CPU increases with use of SSH, you might be concerned about denial-of-service attacks. However, the forwarding engine takes care of this issue, because it limits the rate at which it sends packets to the system controller. A flood of packets from a packet generator does not cause problems regardless of whether SSH is enabled.

### *Before You Configure SSH*

You must obtain and install a commercial SSH client on the host from which you want to administer the system. Versions earlier than 2.0.12 of the SSH client are not supported.

Determine your Telnet policy before you configure SSH on your system. Effective use of SSH implies that you should severely limit Telnet access to the system. To limit Telnet access, create access control lists that prevent almost all Telnet usage, permitting only trusted administrators to access the system via Telnet. For example, you might limit access to administrators who need to Telnet to the system from a remote host that does not have the SSH client installed.

You must install and configure a RADIUS server on a host machine before you configure SSH on your system. Refer to your RADIUS server documentation for information on choosing a host machine and installing the server software. You must also configure the RADIUS client on your system. See *ERX Broadband Access Configuration Guide, Chapter 1, Configuring Remote Access to the ERX System* for more information.

## SSH Configuration Tasks

You configure SSH on individual virtual routers, rather than the global system. To configure SSH:

- 1 Access the context of the virtual router.
- 2 Configure encryption.
- 3 Configure user authentication, including connection parameters.
- 4 Configure message authentication.
- 5 Enable SSH.
- 6 Display SSH to verify configuration.

### Configuring Encryption

The embedded SSH server and external SSH client maintain separate lists of the encryption algorithms that each supports. Lists are kept for inbound and outbound algorithms. For the server:

- Inbound means the algorithms that the server supports for information coming in from a client.
- Outbound means the algorithms that the server supports for information it sends out to a client.

You must configure each list separately. Refer to your SSH client documentation for details on configuring encryption on your client. The system supports the following SSH algorithms for encryption:

- 3des-cbc – A triple DES block cipher with 8-byte blocks and 24 bytes of key data. The first 8 bytes of the key data are used for the first encryption, the next 8 bytes for the decryption, and the following 8 bytes for the final encryption.
- blowfish-cbc – A block cipher with 8-byte blocks and 128-bit keys that provides strong encryption and is faster than DES.
- twofish-cbc – A block cipher with 16-byte blocks and 256-bit keys that is stronger and faster than Blowfish encryption.

Although it is not recommended, you can also specify **none**. In this case, the system does not perform encryption.

### *ip ssh crypto*

- Use to add an encryption algorithm to the specified support list for the SSH server.

Example 1 – this example adds the blowfish-cbc algorithm to the list of supported inbound algorithms.

```
host1(config)#ip ssh crypto client-to-server blowfish-cbc
```

Example 2 – this example removes the 3des-cbc algorithm from the list of supported outbound algorithms.

```
host1(config)#ip ssh crypto server-to-client no 3des-cbc
```

- The **default** version restores the specified list to the factory default, which includes all supported algorithms (3des-cbc, twofish-cbc, and blowfish-cbc). The default list does not include the **none** option.

Example

```
host1(config)#ip ssh crypto server-to-client default
3des-cbc
```

- If you do not specify a direction (client-to-server or server-to-client), the command applies the algorithm to both inbound and outbound lists.
- Use the **no** version to remove or exclude an algorithm from the specified list.

### Configuring User Authentication

The system supports RADIUS for user authentication. RADIUS authentication is enabled by default. You must have previously configured a RADIUS server on a host machine and the RADIUS client on your system.

You can specify timeout and retry limits to control the SSH connection process. The limits apply only from the time the user first tries to connect until the user has been successfully authenticated. The timeout limits are independent of any limits configured for virtual terminals (vty). The following limits are supported:

- SSH timeout – The maximum time allowed for a user to be authenticated, starting from the receipt of the first SSH protocol packet.
- Authentication retry – The number of times a user can try to correct incorrect information—such as a bad password—in a given connection attempt.
- Sleep – Prevents a user that has exceeded the authentication retry limit from connecting from the same host within the specified period.

See the following commands.

### ***ip ssh authentication-retries***

- Use to set the number of times that a user can retry a failed authentication, such as trying to correct a wrong password. The SSH server terminates the connection when the limit is exceeded.
- Specify an integer from 0–20.
- Example

```
host1(config)#ip ssh authentication-retries 3
```
- Use the **no** version to restore the default value of 20 retry attempts.

### ***ip ssh disable-user-authentication***

- Use to disable RADIUS password authentication. If you disable RADIUS authentication, all SSH clients that pass protocol negotiation are accepted.
- RADIUS authentication is enabled by default.
- Example

```
host1(config)#ip ssh disable-user-authentication
```
- Use the **no** version to restore RADIUS authentication.

### ***ip ssh sleep***

- Use to set a sleep period in seconds for users that have exceeded the authentication retry limit. Connection attempts from the user at the same host are denied until this period expires.
- Specify any nonnegative integer.
- Example

```
host1(config)#ip ssh sleep 300
```
- Use the **no** version to restore the default value of 600 seconds.

### ***ip ssh timeout***

- Use to set a timeout period in seconds. The SSH server terminates the connection if protocol negotiation—including user authentication—is not completed within this timeout.
- Specify an integer from 10–600.
- Example

```
host1(config)#ip ssh timeout 480
```
- Use the **no** version to restore the default value of 600 seconds.

## Configuring Message Authentication

The SSH server and SSH client maintain separate lists of the message authentication algorithms that each supports. Lists are kept for *inbound* and *outbound* algorithms. For the server, *inbound* means the algorithms that the server supports for information coming in from a client. For the server, *outbound* means the algorithms that the server supports for

information it sends out to a client. You must configure each list separately. The system supports the following SSH algorithms for hash function-based message authentication:

- `hmac-sha1` – Uses Secure Hash Algorithm 1 (SHA-1) to create a 160-bit message digest from which it generates the MAC.
- `hmac-sha1-96` – Uses the first 96 bits of the SHA-1 message digest to generate the MAC.
- `hmac-md5` – Uses MD5 hashing to create a 128-bit message digest from which it generates the MAC.

Although it is not recommended, you can also specify **none**. In this case, the system does not verify the integrity of the data.

### ***ip ssh mac***

- Use to add a message authentication algorithm to the specified support list for the SSH server.

Example

```
host1(config)#ip ssh mac server-to-client hmac-md5
```

This example adds the `hmac-md5` algorithm to the list of supported outbound algorithms.

- If you do not specify a direction (client-to-server or server-to-client), the command applies the algorithm to both inbound and outbound lists.
- The **default** version restores the specified list to the factory default, which includes all supported algorithms (`hmac-md5`, `hmac-sha1`, and `hmac-sha1-96`). The default list does not include the *none* option.

Example

```
host1(config)#ip ssh mac client-to-server default hmac-sha1
```

This example restores the `hmac-sha1` algorithm to the list of supported inbound algorithms.

- Use the **no** version to remove or exclude an algorithm from the specified list.

Example

```
host1(config)#ip ssh mac client-to-server no hmac-sha1
```

This example removes the `hmac-sha1` algorithm from the list of supported inbound algorithms.

### Enabling and Disabling SSH

The SSH server daemon starts only if the server host key exists when the system boots. The host key resides in NVS and is persistent across system reboots. Once started, the daemon listens for traffic on TCP port 22. The server daemon is disabled by default.

## **crypto key dss**

- Use the **generate** keyword to create the SSH server host key and enable the daemon.
- Example
 

```
host1(config)#crypto key generate dss
```
- Use the **zeroize** keyword to remove the SSH server host key and stop the SSH daemon if it is running. Issuing this command terminates any active client sessions. The next time the system boots after this command is issued, the SSH server daemon is not started.
- The command is not displayed by the **show config** command.



**Note:** SSH can be enabled or disabled regardless of the state of the Telnet daemon. If SSH is enabled, use access control lists to limit access via Telnet. See *Virtual Terminal Access Lists* in this chapter for information on using access control lists.

- Example
 

```
host1(config)#crypto key zeroize dss
```
- There is no **no** version.

## Displaying SSH Status

You can monitor the current state of the SSH server with the **show ip ssh** command.

## **show ip ssh**

- Use to display the current state of the SSH server.
- Use the **detail** keyword to display the encryption and MAC algorithm lists for the client and server. For each active session, **detail** shows the version of SSH running on the client and the algorithms in use for encryption and message authentication.
- Example
 

```
host1#show ip ssh
```
- Field descriptions
  - › daemon status – indicates whether the SSH server is enabled; if so, how long it has been up
  - › supported encryption, inbound – encryption algorithms supported inbound from the client
  - › supported encryption, outbound – encryption algorithms supported outbound to the client
  - › supported MAC, inbound – message authentication code algorithms supported inbound from the client
  - › supported MAC outbound – message authentication code algorithms supported outbound to the client
  - › connections since last system reset – number of connections made via SSH since the last time the system was reset

- › connections since daemon startup – number of connections made since the SSH server was enabled
- › active sessions – number of SSH sessions currently active
  - id – session ID number
  - username – username for the remote user that initiated the session
  - host – IP address of the remote client
  - uptime (d:h:m:s) – duration of the session
  - client version – version of the SSH software run by the remote client
  - ciphers inbound/outbound – encryption algorithms used by the client and the system for this session
  - MAC inbound/outbound – message authentication code algorithms used by the client and the system for this session
- Example

```
host1#show ip ssh detail
```

```
SSH Server version: SSH-2.0-2.0.12
```

```
daemon status: enabled, up since MON NOV 08 1999 14:38:19 UTC
```

```
supported encryption, inbound: 3des-cbc,blowfish-cbc,twofish-cbc
supported encryption, outbound: 3des-cbc,blowfish-cbc,twofish-cbc
supported MAC, inbound: hmac-shal,hmac-shal-96,hmac-md5
supported MAC, outbound: hmac-shal,hmac-shal-96,hmac-md5
```

```
connections since last system reset: 4 out of 4 attempts
connections since daemon startup: 4 out of 4 attempts
```

```
active sessions: 1
```

| id | username | host       | uptime     | client version              | ciphers           | MAC               |
|----|----------|------------|------------|-----------------------------|-------------------|-------------------|
|    |          |            | (d:h:m:s)  |                             | inbound/outbound  | inbound/outbound  |
| 3  | mcarr    | 10.0.0.145 | 0:00:00:19 | SSH-2.0-2.0.12 F-SECURE SSH | 3des-cbc/3des-cbc | hmac-md5/hmac-md5 |

To view failed connection attempts and other protocol errors logged at the error severity level, use the **show log data** command:

```
host1#show log data category ssh severity error
```

### Terminating an SSH Session

You can use the session identifier to terminate an SSH session.

#### **disconnect ssh**

- Use to terminate an active SSH session.
- Use the **show ip ssh** command to determine the session identifier for the session to terminate.

- Example

```
host1(config)#disconnect ssh 12
```



**Note:** You can also use the **clear line vty** terminal command to terminate SSH sessions. In that case, use the **show users** command to determine the virtual terminal number to specify with the **clear line vty** terminal command.

- There is no **no** version.

## Restricting User Access

Users who are authenticated via RADIUS can be restricted to certain sets of commands and virtual routers (VRs).

### *Restricting Access to Commands*

You can use RADIUS authentication to specify a level of command access for users. If you do not configure RADIUS authentication for the console or virtual terminals, all users who successfully log in are automatically granted Level 1 access.

The vendor-specific attribute (VSA) admin-auth-level supports the levels of access shown in Table 6-2.

**Table 6-2** CLI user access levels

| Access Level | Commands Available                                                                  |
|--------------|-------------------------------------------------------------------------------------|
| 0            | <b>disable</b> , <b>enable</b> , <b>exit</b> , and <b>help</b> commands             |
| 1            | Level 0 commands and all other commands available in User Exec mode                 |
| 5            | Level 1 commands and all Privileged <b>show</b> commands                            |
| 10           | All commands except support commands                                                |
| 15           | Commands that Juniper Networks Technical Support may provide and all other commands |

In addition to VSA access level support, the software provides access to levels 1 and 10 through the initial-auth-level in the standard RADIUS service-type attribute. If the RADIUS service-type attribute is included in the RADIUS access-accept message, the standard attribute overrides any VSA setting.

If you are using the RADIUS service-type attribute to assign access levels, the system sets the initial-auth-level as follows:

- If the service-type attribute is set to “administrative,” then the initial-auth-level is set to 10.
- If the service-type attribute is set to “nas prompt” or “login,” the initial-auth-level is set to 1.

### Per-User Enable Authentication

Once a user is authenticated through RADIUS, the RADIUS server provides the ERX system with the names of the privilege levels (for example, “10”) that the user has **enable** access to. When the user attempts to access a privilege level through the **enable** command, the system either denies or approves the user’s request. The decision to deny or approve the user’s request is based on the list the system received through RADIUS. See Table 6-3.

**Table 6-3** Juniper Networks-specific CLI access VSA descriptions

| VSA                              | Description                                            | Type | Length | Subtype | Subtype Length | Value                                            |
|----------------------------------|--------------------------------------------------------|------|--------|---------|----------------|--------------------------------------------------|
| Juniper-Initial-CLI-Access-Level | Specifies the initial level of access to CLI commands. | 26   | len    | 18      | sublen         | Single attribute; enter only: 0, 1, 5, 10, or 15 |
| Juniper-Alt-CLI-Access-Level     | Specifies level of access to CLI commands.             | 26   | len    | 20      | sublen         | Single attribute; enter only: 0, 1, 5, 10, or 15 |



**Note:** All levels to which a user can have access must explicitly be specified in the Admin-Auth-Set VSA.

The user is not prompted for a password, since the system knows whether or not the user should have access to the requested level. If the user is not authenticated through RADIUS, the system uses the system-wide **enable** passwords instead.

### Restricting Access to Virtual Routers

You can use RADIUS authentication to specify whether users can access all virtual routers (VRs), one specific VR, or a set of specific VRs.



**Note:** This classification is independent of the command access levels configurable via the Juniper-Initial-CLI-Access-Level VSA.

The VSA Juniper-Allow-All-VR-access controls access; the VSA Juniper-Virtual-Router controls the VR to which the user logs in, and the

VSA `Juniper-Alt-CLI-Virtual-Router-Name` specifies which VRs other than the VR specified by the VSA `Juniper-virtual-router` are accessible to restricted users. See Table 6-4.

**Table 6-4** Juniper Networks-specific virtual router access VSA descriptions

| VSA                                 | Description                                                                                                                                                                                                         | Type | Length | Subtype | Subtype Length | Value                                  |
|-------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|--------|---------|----------------|----------------------------------------|
| Juniper-Allow-All-VR-Access         | Specifies user access to all virtual routers.                                                                                                                                                                       | 26   | len    | 19      | sublen         | Integer:<br>0 – disable,<br>1 – enable |
| Juniper-Virtual-Router              | Specifies the VR to which the user logs in or the only VR to which a user has access. The default setting is the default VR.                                                                                        | 26   | len    | 1       | sublen         | String:<br><i>virtual-router-name</i>  |
| Juniper-Alt-CLI-Virtual-Router-Name | Specifies a VR, other than the VR specified by the <code>Juniper-Virtual-Router</code> VSA, to which the user has access. You can define this VSA multiple times to define a set of VRs to which a user has access. | 26   | len    | 21      | sublen         | String:<br><i>virtual-router-name</i>  |

### VSA Configuration Examples

Consider a system on which five VRs have been configured. The VRs are called Boston, Chicago, Detroit, Los Angeles, and San Francisco. The following examples illustrate how to use the VSAs to control a user's access to these VRs.

**Example 1** In this example, you want the user to have access to all VRs and to log in to the default VR. Accept the default setting or set the following VSA:

- `Juniper-Allow-All-VR-Access – 1`

**Example 2** In this example, you want the user to have access to all VRs and to log in to the VR Boston. Set the VSAs as follows:

- `Juniper-Allow-All-VR-Access – 1`
- `Juniper-Virtual-Router – Boston`

**Example 3** In this example, you want the user to have access only to the VR Boston. Set the VSAs as follows:

- `Juniper-Allow-All-VR-Access – 0`
- `Juniper-Virtual-Router – Boston`

**Example 4** In this example, you want the user to log in to VR Boston, and to have access to VRs Chicago, Los Angeles, and San Francisco. Set the VSAs as follows:

- Juniper-Allow-All-VR-Access – 0
- Juniper-Virtual-Router – Boston
- Juniper-Alt-CLI-Virtual-Router-Name – Chicago
- Juniper-Alt-CLI-Virtual-Router-Name – Los Angeles
- Juniper-Alt-CLI-Virtual-Router-Name – San Francisco

#### Commands Available to Users

If you do not configure RADIUS authentication for the console or virtual terminals, there are no restrictions on VR access for any user who successfully logs onto the system. For example, nonrestricted users can

- Issue the **virtual-router** command in Privileged Exec mode, to switch to another previously created virtual router.
- Issue the **virtual-router** command in Global Configuration mode to create a new virtual router and switch to its context.
- Access Global Configuration mode to configure the system and virtual routers.
- View all settings for the system and all virtual routers.

User restricted to one or a set of specific VRs can see and use only a limited set of commands to monitor the status of those VRs and view some configuration settings on those VRs. More specifically, such users

- Can issue the **virtual-router** command in Privileged Exec mode to switch to another previously configured VR to which they have access.
- Cannot create new VRs or access VRs other than those to which they have access.
- Cannot access Global Configuration mode and cannot configure VRs to which they have access.
- Cannot see or use any commands associated with the file system, boot settings, or system configuration.

Table 6-5 lists some, but not all, commands accessed from User Exec or Privileged Exec mode that are available only to users with no VR restriction.

**Table 6-5** User Exec or Privileged Exec mode commands

|                            |                           |                  |
|----------------------------|---------------------------|------------------|
| clear line                 | reload                    | show redundancy  |
| clock set                  | reload slot               | show secrets     |
| copy                       | rename                    | show subsystems  |
| copy running-configuration | redundancy force-failover | show timing      |
| delete                     | redundancy revert         | show users       |
| dir                        | show boot                 | show utilization |
| disconnect ssh             | show config               | srp switch       |
| configure                  | show exception dump       | synchronize      |
| erase secrets              | show ip ssh               |                  |
| halt                       | show line                 |                  |

