Chapter 1. Router Configuration and File Management

1.0. Introduction

You can think of a Cisco router as a special-purpose computer. It has its own operating system, which is called the Internetwork Operating System (IOS), as well as files and filesystems. So we'll start with a discussion of the basic system administration functions that a router engineer must perform. This includes managing your router's filesystems, upgrading the operating system, doing backups, and restoring the system configuration.

Cisco routers use flash memory, rather than disks, for storing information. Flash storage media is significantly more expensive and slower than disk storage, but the amount of storage needed to run a router is relatively small compared to the amount needed to run a general-purpose computer. Flash also has the important benefit that it tends to be more reliable than disk storage.

Flash storage is similar to Random Access Memory (RAM), but it doesn't need power to retain information, so it is called non-volatile. And, unlike Read Only Memory (ROM), you can erase and rewrite flash easily. There are other types of non-volatile solid state storage, such as Erasable Programmable Read Only Memory (EPROM) and Electronically Erasable Programmable Read Only Memory (EEPROM). EPROM is not suitable for routers because it generally requires an external device such as an ultraviolet light shone through a window on the chip to erase it. EEPROM, on the other hand, can be erased by simply sending an erase signal to the chip. But there is a key difference between EEPROM and flash memory: when you erase something from an EEPROM device, you must erase the entire device, while flash devices allow selective deletion of parts of the medium.

This is an important feature for routers, because you don't always want to erase the entire storage medium in order to erase a single file. In Recipe 1.11 and Recipe 1.12, we discuss ways to erase single files on some types of routers, depending on the type of file-system used.

There are at least two main pieces of non-volatile storage in a Cisco router. The router's configuration information is stored in a device called the Non-Volatile RAM (NVRAM), and the IOS images are stored in a device called the flash (lowercase). It's important to keep these names straight because, of course, all Flash memory is nonvolatile RAM. And, in fact, most routers use Flash technology for their NVRAM. So it's easy to get confused by the terms.

On most Cisco routers, the NVRAM area is somewhere between 16 and 256Kb, depending on the size and function of the router. Larger routers are expected to have larger configuration files, so they need more NVRAM. The flash device, on the other hand, is usually upgradeable, and can be anywhere from a few megabytes to hundreds of megabytes.

We often talk about a router's configuration file, but there are actually two important configuration files on any router. There is the configuration file that describes the current running state of the router, which is called the running-config. Then, there is the configuration file that the router uses to boot, which is called the startup-config. Only the startup-config is stored in NVRAM, so it is important to periodically check that the version of the configuration in the NVRAM is synchronized with the version that the router is currently running. Otherwise you could get a surprise from ancient history the next time the router reboots. You can synchronize the two configuration files by simply copying the running-config onto the startup-config file:

```
Router1#copy running-config startup-config
```

Many Cisco engineers, including the authors, still use the old-fashioned version of this command out of force of habit:

```
Router1#write memory
```

However, this command is not only deprecated, it's also less descriptive of what the router is doing.

The router uses the larger flash storage device for holding the operating system, or IOS. Unlike the operating systems on most computers, the IOS is a single file containing all of the features and functions available on the router. You can obtain the IOS image files from Cisco on CD or, if you have an account on their system, you can download IOS files from the Cisco web site using FTP.

Most of the examples throughout this book assume that you have IOS Version 12. However, many of the features we discuss are also available in earlier versions. Although there may be slight syntax changes, we expect that Cisco will continue to support all of the features we describe well into the future. It is important to be flexible because if you work with Cisco routers a lot, you will encounter a large variety of different IOS versions, with various subtle differences. Unfortunately, some of these subtle differences are actually bugs. Cisco offers a detailed bug tracking system on their web site for registered users.

There are several important things to consider when you go to change the IOS version on a router. First is the feature set. For each IOS release, Cisco produces several different versions. They usually offer an Enterprise Feature Set, which includes all of the different feature options available at a given time. Because the IOS is a monolithic file containing all features and all commands, the Enterprise IOS files are usually quite large. The Enterprise version is generally much more expensive than the various stripped-down versions.

The simplest IOS version is usually the IP Only Feature Set. As the name suggests, this includes only TCP/IP based functionality. In most networks, you will find that the IP Only Feature Set is more than sufficient. In fact, almost all of the recipes in this book will work with the IP Only version of IOS.

If you require other protocols such as IPX or AppleTalk, Cisco produces an IOS Feature Set called Desktop that contains these protocols. They also offer several other important variations such as IP Plus, IP Plus IPSec 56, IP Plus IPSec 3DES, and so forth. The contents of these different versions (and even their names to some extent) vary from release to release. We encourage you to consult Cisco's feature matrixes to ensure that the features you need are in the IOS version that you have.

One of the most important considerations with any IOS release is whether you have sufficient RAM and Flash memory to support the new version. You can see how much storage your router has by looking at the output of the show version command.

The other important thing to remember about IOS images on Cisco routers is that every router has a fallback image located in the router's ROM. This IOS image cannot be changed or upgraded without physically replacing the ROM chips in the router.

The router's ROM contains three items: the power on self test (POST), the bootstrap program, and a limited version of the router's operating system. The router uses the bootstrap program while booting. The IOS image in ROM is usually an extremely stripped-down version that doesn't support many common features (routing protocols, for example). In the normal boot cycle, the router will first load the POST, then the bootstrap program followed by the appropriate IOS image. Please refer to Recipe 1.7 for more information about booting from different IOS files.

Recipe 1.7 also shows how to adjust the configuration register values. These values set a variety of boot options, and even allow you to force the router to stop its boot process before loading the IOS. This can be useful if the IOS image is corrupted, or if you need to do password recovery.

1.1. Configuring the Router via TFTP

Problem

You want to load configuration commands via the Trivial File Transfer Protocol (TFTP).

Solution

You can use the copy tftp: command to configure the router via the TFTP:

```
Router1#copy tftp://172.25.1.1/NEWCONFIG
 running-config
Destination filename [running-config]? <enter>
Accessing tftp://172.25.1.1/NEWCONFIG...
Loading NEWCONFIG from 172.25.1.1 (via FastEthernet0/0.1): !
[OK - 24 bytes]


24 bytes copied in 0.192 secs (125 bytes/sec)
Router1#
```

Note

IOS versions before 12.0 used the command configure network. This command is still available in more recent versions, but it is now deprecated and may not continue to be available in the future.

Discussion

Generally, most people configure their routers using Telnet and the configure terminal command. For large configuration changes, people tend to resort to cutting and pasting a large set of commands. While this method works, it is inefficient and slow, particularly if you have to configure large numbers of routers. When you use TFTP to download a large set of configuration commands, the router doesn't need to echo each character to your screen, which reduces the overhead and increases the speed.

In our example, we configured the router by making it download the file called NEWCONFIG from the server at 172.25.1.1 using TFTP. The router copies the entire file via TFTP before entering the commands into the running configuration. This is extremely useful because using some commands in the middle of a configuration could disrupt your access to the router—but the rest of the commands might fix the problem. If you tried to enter them manually using Telnet and configure terminal, you would simply lock yourself out of the router. A typical example of this problem happens when you replace an active access list. When you enter the first line, the router puts an

implicit deny all at the end, which could break your session. However, entering commands using TFTP avoids this problem.

The last line of any configuration file that you copy into the router like this should be the end command. This lets the router know that it has reached the end of the file. If you don't do this, the router will still accept all of the commands normally, but it will put the following error into its logs:

Jan 19 11:26:38: %PARSER-4-BADCFG: Unexpected end of configuration file.

If you have the end command in your configuration file, seeing this message will tell you that the router didn't get all of the configuration commands. But if you don't terminate the file properly, it's impossible to tell if the transfer was successful.

Instead of TFTP, you can use the FTP protocol to download configuration files. FTP has a number of advantages over TFTP in terms of reliability and security. Recipe 1.14 shows how to load configuration commands using FTP instead of TFTP.

See Also

Recipe 1.14

1.2. Saving Router Configuration to Server

Problem

You want to store a backup copy of your router's configuration on a TFTP server.

Solution

This example shows how to use TFTP to upload a copy of the router's active configuration to a remote server:

Freebsd% touch /tftpboot/router1-confg

Freebsd% chmod 666 /tftpboot/router1-confg

Freebsd% telnet Router1

Trying 172.25.1.5...

Connected to Router1.

Escape character is '^]'.

User Access Verification

Password: <vtypassword>

Router1>en

Password: <enablepassword>

Router1#copy running-config tftp://172.2

5.1.1/router1-confg

Address or name of remote host [172.25.1.1]? <enter>

Destination filename [router1-confg]? <enter>

!!!

9640 bytes copied in 3.956 secs (2437 bytes/sec)

Router1#

Discussion

We cannot overstress the importance of making regular backups of your router configuration files, and keeping copies of these files in a safe place. If a serious failure damages a router's hardware or software, your configuration will be destroyed. Anybody who has had to reconstruct a complex router configuration file from memory can tell you how difficult and stressful this task is! But, if you have a backup of the last working configuration file, you can usually get a router working again within minutes of fixing any hardware problems.

Typical Mean Time Between Failure (MTBF) estimates for Cisco routers tend to be about 16 years. This sounds like a long time, but in a large network it means that you can expect to see a few failures every year. Unfortunately, human errors resulting in complete or partial loss of the configuration file are far more common than device failures.

In the example, we created an empty backup configuration file on the TFTP server, then instructed the router to send its running configuration to this server. It is important to adjust the file permissions with the Unix chmod command. The transfer will fail if the configuration file is not world writable. We highly recommend moving the configuration files out of the TFTP directory to ensure that the file isn't read by unauthorized users, or accidentally overwritten.

Reading files located in the TFTP directory is trivial, because the TFTP program needs this directory to be both world readable and world writeable. Since router configuration files contain passwords and IP addresses, you should take steps to protect these files as much as possible. In fact, you don't even need to be logged into the TFTP server to read these files. In the following example, we are able to access the TFTP server and read a router configuration file from another router:

Router1#more tftp://172.25.1.1/router1

-confg

!

! Last configuration change at 11:23:59 EST Sat Jan 11 2003 by ijbrown

```
! NVRAM config last updated at 00:37:16 EST Sat Jan 11 2003 by ijbrown
!
version 12.2
service tcp-keepalives-in
service timestamps debug datetime msec
service timestamps log datetime localtime
service password-encryption
!
hostname Router1
<removed for brevity>
```

As you can see, any files left in the TFTP directory can be easily viewed or even deliberately corrupted. TFTP is notoriously insecure, so we recommend using care whenever you work with this protocol.

Recipe 1.18 provides an automated script that gathers the configuration files for a list of routers on a nightly basis and stores these files for 30 days, by default.

See Also

Recipe 1.14; Recipe 1.18

1.3. Booting the Router Using a Remote Configuration File

Problem

You want to boot the router using an alternate configuration.

Solution

The following set of commands allows you to automatically load a configuration file located on a remote TFTP server when the router boots:

```
Router1#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router1(config)#service config
Router1(config)#boot network tftp Network
-auto 172.25.1.1
Router1(config)#boot host tftp Router8
```

-auto 172.25.1.1

Router1(config)#end

Router1#

Discussion

By default, when the router reloads, it will read the configuration information from a file in its NVRAM. Cisco commonly refers to this file as the startup configuration file. However, you can configure the router to load all or part of its configuration from a remote server via TFTP. This feature does not prevent the router from loading its startup configuration from NVRAM. In fact, the router will load its local startup file before proceeding to the TFTP server files.

Uses for this feature vary, although most people who implement it do so because their configuration file has grown too large for their NVRAM to handle. It can also be a useful way of keeping an access list that is shared by a number of routers centralized and up-to-date. We have sometimes used it as a temporary measure when the NVRAM in a router is damaged.

However, we consider this feature to be highly risky and recommend avoiding it in most cases. If the problem is simply one of NVRAM capacity, Recipe 1.4 shows how to compress the startup configuration file to help fit more information into your existing NVRAM. Also, since routers can operate for years without reloading, using this feature to keep your routers up-to-date seems pointless.

If you choose to implement remote configuration despite these cautions, you need to understand how the boot process works. When you enable the service config option, the router attempts to load a network file, then a host file. The router assumes that network files are common to all routers, while the host file contains router-specific information. If it can't find these files, the router will generate the following error messages:

%Error opening tftp://255.255.255.255/network-confg (Timed out)

%Error opening tftp://255.255.255.255/cisconet.cfg (Timed out)

%Error opening tftp://255.255.255.255/router1-confg (Timed out)

%Error opening tftp://255.255.255.255/router1.cfg (Timed out)

Here you can see what happened when we enabled the service config option and reloaded our router, which was called router1. It attempted to load several different files automatically. The first two files have generic network file names. The router then looks for the host file under two different names. It attempts to load these configuration files from IP address 255.255.255.255 by default.

When we add the boot commands, the router looks for the specified files from the appropriate TFTP server. Again, notice the order that the router loaded the files: the network file first, followed by the host file.

Loading Network-auto from 172.25.1.1 (via Ethernet0): !

[OK - 27/4096 bytes]

Loading Router1-auto from 172.25.1.1 (via Ethernet0): !

[OK - 71/4096 bytes]

If you do not configure the router to load specific network or host filenames, it will try to load the default files, shown in the trace above. If these files don't exist, the router will pause for a significant amount of time while it tries to find them. When you use this feature, you should always include both a network and a host file to load. If you don't need a network file, you can put a file on the server that contains only the keyword end.

Note

This feature loads configuration commands only into the running configuration. It does not copy them into the startup configuration file.

The show version tells you whether the router was able to load these files successfully:

Router1#show version

Cisco Internetwork Operating System Software

IOS (tm) 2500 Software (C2500-IO-L), Version 12.2(7a), RELEASE SOFTWARE (fc2)

Copyright (c) 1986-2002 by cisco Systems, Inc.

Compiled Thu 21-Feb-02 02:07 by pwade

Image text-base: 0x0304CF80, data-base: 0x00001000

ROM: System Bootstrap, Version 5.2(8a), RELEASE SOFTWARE

BOOTLDR: 3000 Bootstrap Software (IGS-RXBOOT), Version 10.2(8a), RELEASE SOFTWARE

(fc1)

Router1 uptime is 4 minutes

System returned to ROM by reload

System image file is "flash:c2500-io-l.122-7a.bin"

Host configuration file is "tftp://172.25.1.1/Router1-auto"

Network configuration file is "tftp://172.25.1.1/Network-auto"

cisco 2500 (68030) processor (revision D) with 16384K/2048K bytes of memory.

Processor board ID 04915359, with hardware revision 00000000

Bridging software.

X.25 software, Version 3.0.0.

2 Ethernet/IEEE 802.3 interface(s)

2 Serial network interface(s)

32K bytes of non-volatile configuration memory.

16384K bytes of processor board System flash (Read ONLY)

Configuration register is 0x2102

The service config option is disabled by default. However, if the router tries to boot but cannot find its startup configuration file, it will automatically enable this option and attempt to find a configuration file through the network:

00:00:25: AUTOINSTALL: Ethernet0 is assigned 172.25.1.30

00:00:25: AUTOINSTALL: Obtain siaddr 172.25.1.3 (as config server)

00:00:25: AUTOINSTALL: Obtain default router (opt 3) 172.25.1.3

%Error opening tftp://172.25.1.3/network-confg (No such file or directory)

%Error opening tftp://172.25.1.3/cisconet.cfg (No such file or directory)

%Error opening tftp://172.25.1.3/router-confg (No such file or directory)

%Error opening tftp://172.25.1.3/ciscortr.cfg (No such file or directory)

%Error opening tftp://172.25.1.3/network-confg (No such file or directory)

%Error opening tftp://172.25.1.3/cisconet.cfg (No such file or directory)

Two interesting things happen if you reload a router with an empty configuration file. First, the router enables its autoinstall option and attempts to acquire an IP address via DHCP. In this example, the router obtained a DHCP address of 172.25.1.30. Second, after it obtains a dynamic address, it attempts to load a configuration file via TFTP.

Notice the filenames that the router cycles through in an attempt to load a configuration file. If there happens to be a file with one of these names in the TFTP directory, the router will download it and use its contents to configure itself. This can cause serious problems.

See Also

Recipe 1.4; Recipe 1.5

1.4. Storing Configuration Files Larger than NVRAM

Problem

Your configuration file has become larger than the router's available NVRAM.

Solution

You can compress your router's configuration file before saving it to NVRAM to allow you to save more configuration information. The command service compress-config will compress the configuration information when the router saves the file, and uncompress it when it is required:

Router1#configure terminal

Enter configuration commands, one per line. End with CNTL/Z.

Router1(config)#service compress-config

Router1(config)#end

Router1#

Discussion

Cisco generally ships its routers with more than enough NVRAM to store an average configuration file. However, there are times when configuration files exceed the available NVRAM. For instance, some routers contain large access lists that are hundreds of lines in length. When configuration files grow beyond the finite amount of NVRAM you will begin to have problems.

The first sign of serious problems with an overly large configuration file is usually when the router refuses to save its configuration because of size. This is a dangerous situation because the router can no longer keep a copy of the whole running-configuration file in its NVRAM storage, and it is difficult to predict how much of your configuration will be lost if you were to reload the router.

Turning on compression roughly doubles the size of the configuration file you can store. You must put the command service compress-config into the configuration with a configure terminal. Then, for this command to take effect, you need to copy the running configuration file to NVRAM:

Router1#copy running-config startup-config

Destination filename [startup-config]? <enter>

Building configuration...

Compressed configuration from 9664 bytes to 4903 bytes[OK]

Router1#

In this case, you can see that the compression reduced the configuration file to less than half of its original size. This compression algorithm will not attempt to compress a file that is three times larger than the available NVRAM space. Although this limit exists, we have never seen a router approach a 3 to 1 ratio in practice.

The actual amount of available NVRAM storage varies between different router models. You can see how much total NVRAM storage is available on a particular router with the show version command:

```
Router1#show version
Cisco Internetwork Operating System Software
IOS (tm) C2600 Software (C2600-IK9O3S-M), Version 12.2(12a), RELEASE SOFTWARE (fc1)
Copyright (c) 1986-2002 by cisco Systems, Inc.
Compiled Tue 24-Sep-02 02:05 by pwade
Image text-base: 0x8000808C, data-base: 0x8127FF40

ROM: System Bootstrap, Version 11.3(2)XA4, RELEASE SOFTWARE (fc1)

Router1 uptime is 12 hours, 15 minutes
System returned to ROM by reload
System restarted at 23:18:45 EST Fri Jan 10 2003
System image file is "flash:c2600-ik9o3s-mz.122-12a.bin"

cisco 2621 (MPC860) processor (revision 0x102) with 45056K/4096K bytes of memory.
Processor board ID JAB04130B2Q (1293133440)
M860 processor: part number 0, mask 49
Bridging software.
X.25 software, Version 3.0.0.
2 FastEthernet/IEEE 802.3 interface(s)
2 Serial network interface(s)
32K bytes of non-volatile configuration memory.
16384K bytes of processor board System flash (Read/Write)

Configuration register is 0x2102

Router1#
```

This router contains 32Kb of NVRAM in which to store configuration files. The top of the output from the show startup-config command shows how much NVRAM storage is available, and how much this particular configuration file requires. If you enable compression, it will also show the compressed and uncompressed sizes:

Router1#show startup-config

Using 5068 out of 29688 bytes, uncompressed size = 9969 bytes

Uncompressed configuration from 5068 bytes to 9969 bytes

!

! Last configuration change at 12:36:22 EST Sat Jan 11 2003 by ijbrown

! NVRAM config last updated at 13:34:57 EST Sat Jan 11 2003 by ijbrown

!

version 12.2

<removed for brevity>

In this case we have used about 5Kb of the available 29Kb for this router's configuration file. However, the show version output indicates a total of 32Kb NVRAM, leaving 3Kb unaccounted for. The router's NVRAM used to contain the startup configuration file only, but this is no longer strictly the case. Recent IOS releases also use the same NVRAM space to store information such as private keys for SSH or IPSec, and interface numbers for SNMP. You can see information about all of these files with the dir nvram: command:

Router1#dir nvram:

Directory of nvram:/

```
  20  -rw-       5068         <no date> startup-config
  21  ----       2302         <no date> private-config
   1  ----          0         <no date> persistent-data
   2  -rw-        133         <no date> ifIndex-table

29688 bytes total (20218 bytes free)
```

Router1#

Note that the second column from the left in this output contains file attributes similar to those used by the Unix ls command. In this case, both the startup-config and ifIndex-table files are readable and writable. For example, you could look at your router's startup-config file using the following command:

Router1#more nvram:/startup-config

You can view any file that has an “r”, and you can modify any file that has a “w”. The two files in this router’s NVRAM that have neither “r” nor “w” can’t be displayed, modified, or deleted. Note that although the ifIndex-table is readable, it is in a binary format that isn’t very meaningful when displayed with the more command.

See Also

Recipe 1.3

1.5. Clearing the Startup Configuration

Problem

You want to clear an old configuration out of your router and return it to a factory default configuration.

Solution

You can delete the current startup configuration files and return the router to its factory default settings with the erase nvram: command:

Router1#erase nvram:

Erasing the nvram filesystem will remove all files! Continue? [confirm]

<enter>

[OK]

Erase of nvram: complete

Router1#reload

System configuration has been modified. Save? [yes/no]: no

Proceed with reload? [confirm] <enter>

You can achieve the same result with the erase startup-config command:

Router1#erase startup-config

Erasing the nvram filesystem will remove all files! Continue? [confirm]

<enter>

[OK]

Erase of nvram: complete

Router1#reload

Proceed with reload? [confirm] <enter>

Discussion

Before you redeploy an old router that you have previously used for another purpose, it is a good idea to completely erase the old configuration. This ensures that the router starts with a clean configuration. However, if you did this on a production router, it would wipe out the configuration and disable all of the interfaces. Fortunately, completely deleting your configuration requires two steps: you must erase the startup configuration, then reload the router.

After you erase your startup configuration file and reload, the router will enter its configuration dialog mode. Most experienced Cisco engineers prefer to skip this mode:

--- System Configuration Dialog ---

Would you like to enter the initial configuration dialog? [yes/no]: no

Would you like to terminate autoinstall? [yes]: yes

Press RETURN to get started!

Router>

At this point, the router's configuration has been returned to the factory defaults:

Router#show running-config

Building configuration...

Current configuration : 431 bytes

!

version 12.2

service timestamps debug uptime

service timestamps log uptime

no service password-encryption

!

hostname Router

```
!
!
ip subnet-zero
!
!
!
!
interface Ethernet0
 no ip address
 shutdown
!
interface Ethernet1
 no ip address
 shutdown
!
interface Serial0
 no ip address
 shutdown
!
interface Serial1
 no ip address
 shutdown
!
ip classless
ip http server
ip pim bidir-enable
!
```

!

line con 0

line aux 0

line vty 0 4

!

end

Router#

You can now safely reconfigure the router for its new function. Note that the factory defaults vary, depending on the level of IOS you are running and the hardware installed in the router.

If you accidentally erase the startup configuration file, you can still recover it if the router has not yet been reloaded. Simply copy the running configuration back to the startup configuration:

Router1#show startup-config

startup-config is not present

Router1#copy running-config startup-config

Building configuration...

[OK]

Router1#show startup-config

version 12.2

service timestamps debug datetime msec

service timestamps log datetime localtime

service password-encryption

!

hostname Router1

<removed for brevity>

If the router's configuration is erased and the router is reloaded, it will either need to be reconfigured manually from memory, or preferably, from a backup copy (as in Recipe 1.2).

See Also

Recipe 1.2

1.6. Loading a New IOS Image

Problem

You want to upgrade the IOS image that your router uses.

Solution

The copy tftp command allows you to use TFTP to download a new IOS version into the router's Flash memory:

Router1#copy tftp://172.25.1.1/c2600-ik9o3s-mz

.122-12a.bin flash:

Destination filename [c2600-ik9o3s-mz.122-12a.bin]? <enter>

Accessing tftp://172.25.1.1/c2600-ik9o3s-mz.122-12a.bin...

Erase flash: before copying? [confirm] <enter>

Erasing the flash filesystem will remove all files! Continue? [confirm]

<enter>

Erasing device... eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee ...

erased

Erase of flash: complete

Loading c2600-ik9o3s-mz.122-12a.bin from 172.25.1.1 (via FastEthernet0/0.1):

!!

!!

!!

!!

[OK - 11135588 bytes]

Verifying checksum... OK (0xE643)

11135588 bytes copied in 82.236 secs (135410 bytes/sec)

Router1# reload

Proceed with reload? [confirm] <enter>

Discussion

Sooner or later you will need to upgrade your router's IOS image. Common reasons for upgrading the IOS include new features, bug fixes, and security vulnerabilities. Before you attempt to upgrade your IOS, you should save a backup copy of your current IOS to your TFTP server, as discussed in Recipe 1.9.

You should always start by analyzing how much free space is available in your router's flash to ensure that there is enough room to load the new IOS image. If there isn't enough, then you may have to erase existing image(s) from flash—as we did in our example. In some cases, you may not have enough flash to load the new image at all. You can use the show flash command to see how much flash memory is available:

Router1#show flash

System flash directory:

File Length Name/status

 1 11135588 c2600-ik9o3s-mz.122-12a.bin

[11135652 bytes used, 5117276 available, 16252928 total]

16384K bytes of processor board System flash (Read/Write)

Router1#

Some router models can support additional flash memory in the form of PCMCIA cards. The show slot0: and show slot1: commands will give you details about these additional storage locations. You can store IOS images on the flash cards, but keep in mind that the router will load the first available IOS image in the router's main flash by default. Recipe 1.7 shows how to boot the router using IOS images located on flash cards.

Before you can load an IOS image to your router, you must download the appropriate image from Cisco or purchase it on CD media. You will then need to move the new image into the TFTP directory and ensure that the file is world readable. On Unix systems, you can use the chmod command to do this. If the file is not world readable, the TFTP process will not be able to access it, and the IOS upgrade will fail. You should also do some simple sanity checks on the file by confirming the file size is correct and that the checksum or MD5 "fingerprint" match the values provided by Cisco. Note that it is extremely unwise to use an IOS image that is not created or supported by Cisco.

The router will do several tests while loading a new IOS image to ensure that the process goes smoothly. The first test is to see whether the TFTP server has the specified file:

Router1#copy tftp://172.25.1.1/c2600

-ik9o3s-mz.122-14.bin flash:

Destination filename [c2600-ik9o3s-mz.122-14.bin]?

Accessing tftp://172.25.1.1/c2600-ik9o3s-mz.122-14.bin...

%Error opening tftp://172.25.1.1/c2600-ik9o3s-mz.122-14.bin (No such file or

directory)

Router1#

Here you can see that the router tried to find the file on the TFTP server before going any further, and discovered that it was not present. If the file exists and the permissions are correct, the router will continue with the dialogue. If your TFTP server keeps detailed logs of activity you will see an aborted TFTP file transfer as the router checked to see if the file was available. These aborted attempts are normal and shouldn't cause concern. If the requested file is present but not world readable, the router will show an error message and abort the upgrade process:

Router1#copy tftp://172.25.1.1/c2600

-ik9o3s-mz.122-12a.bin flash:

Destination filename [c2600-ik9o3s-mz.122-12a.bin]? <enter>

Accessing tftp://172.25.1.1/c2600-ik9o3s-mz.122-12a.bin...

%Error opening tftp://172.25.1.1/c2600-ik9o3s-mz.122-12a.bin (Permission denied)

Router1#

Aborting the upgrade early in the process like this ensures that you don't erase the flash unless there is a suitable replacement image available for download.

In the next step of the download process, you must tell the router whether to erase the flash before downloading a new image. If there is enough room available in flash, you can load the new image without erasing the existing image or images. However, if you attempt to download an image and you don't have enough flash space available, the router will protest and abort the procedure:

Router1#copy tftp://172.25.1.1/c2600-ik9o

3s-mz.122-12a.bin flash:

Destination filename [c2600-ik9o3s-mz.122-12a.bin]? <enter>

Accessing tftp://172.25.1.1/c2600-ik9o3s-mz.122-12a.bin...

Erase flash: before copying? [confirm]n

Loading c2600-ik9o3s-mz.122-12a.bin from 172.25.1.1 (via FastEthernet0/0.1): !

%Error copying tftp://172.25.1.1/c2600-ik9o3s-mz.122-12a.bin (Not enough space on

device)

Router1#

The process of upgrading your router's IOS image is fairly forgiving. The router performs sanity checks throughout the process to ensure that image integrity is maintained. After downloading the image, the router verifies the image's checksum. This ensures that the IOS image was not corrupted during transmission. If the image does not pass the verification test, attempt your download again without reloading the router.

You can manually check to see if the IOS checksum is correct by using the verify command. We show how to use the verify command in Recipe 1.10.

If the IOS upgrade goes smoothly and the checksum verifies correctly, then it is safe to reboot your router and load the new IOS image. Once the router becomes reachable again, you should verify that the new IOS loaded correctly with the show version command:

Router1#show version

Cisco Internetwork Operating System Software

IOS (tm) C2600 Software (C2600-IK9O3S-M), Version 12.2(12a), RELEASE SOFTWARE (fc1)

Copyright (c) 1986-2002 by cisco Systems, Inc.

Compiled Tue 24-Sep-02 02:05 by pwade

Image text-base: 0x8000808C, data-base: 0x8127FF40

ROM: System Bootstrap, Version 11.3(2)XA4, RELEASE SOFTWARE (fc1)

Router1 uptime is 2 minutes

System returned to ROM by reload

System restarted at 11:53:26 EST Sat Jan 11 2003

System image file is "flash:c2600-ik9o3s-mz.122-12a.bin"

cisco 2621 (MPC860) processor (revision 0x102) with 45056K/4096K bytes of memory.

Processor board ID JAB04130B2Q (1293133440)

M860 processor: part number 0, mask 49

Bridging software.

X.25 software, Version 3.0.0.

2 FastEthernet/IEEE 802.3 interface(s)

2 Serial network interface(s)

32K bytes of non-volatile configuration memory.

16384K bytes of processor board System flash (Read/Write)

Configuration register is 0x2102

Router1#

In this example, you can see that the router's boot sequence completed successfully and that it's running the new IOS version. Also, notice that the router's new system image file matches the name of the file we just downloaded. This indicates the IOS upgrade was completely successful.

See Also

Recipe 1.7; Recipe 1.9; Recipe 1.10

1.7. Booting a Different IOS Image

Problem

You want to boot using an alternate IOS image.

Solution

To specify which IOS image the router should load next time it reboots, use the boot system command:

Router1#configure terminal

Enter configuration commands, one per line. End with CNTL/Z.

Router1(config)#boot system flash:c3620-jk9o3s-mz.122-7a.bin

Router1(config)#boot system flash:c3620-jos56i-l.120-11.bin

Router1(config)#boot system slot0:c3620-ik9s-mz.122-13.bin

Router1(config)#boot system rom

Router1(config)#end

Note

The sequence of the boot system commands is extremely important, as the router will attempt to load the IOS images in the order that they appear in the configuration file.

Discussion

The router can store as many IOS images in its flash memory as there is space to hold. If there is only one file, it can safely assume that this must be the IOS image to load. However, if the router has several images in its flash

storage, you need to specify which one it should load, or the router will simply select one. This is particularly true on routers that have additional flash memory in the form of PCMCIA cards, which can hold many files—not all of them are necessarily IOS images.

With the default configuration register settings, the router will attempt to load the first accessible IOS image it finds in its flash storage. However, loading the first available image might not be appropriate. For instance, in our last recipe we showed that if you have space, you can download a new IOS image without erasing old images. In this case, you probably want the router to load the newer IOS image. It would be better still if the router tried the new image first, then reverted to the old image if the new one failed to load correctly for any reason. The boot system command allows you to specify not only which IOS images to boot from, but also the order in which to try them if the router has trouble booting.

In the example, this router tries a succession of three different IOS images. If they all fail, it resorts to using its boot ROM image.

As we noted earlier, the sequence of the boot system commands is important since the router will attempt to load the IOS images in order of entry. This means that the only way to add a new IOS image to the start of the list is to remove all of the old boot system commands and reenter them again in the order of preference. You can remove all of the boot system commands at once with the following command:

Router1#configure terminal

Enter configuration commands, one per line. End with CNTL/Z.

Router1(config)#no boot system

Router1(config)#end

Router1#

Once the old boot system commands have been removed, you can configure a new set in whatever order you require.

In addition to allowing you to boot from IOS images in the router's flash storage, you can also use the boot system command to boot from the IOS image in its ROM storage, or by using the TFTP or remote copy (RCP) protocols across the network. Recipe 1.8 shows an example of booting across the network using TFTP. Table 1-1 shows the options for the boot system command.

Table 1-1. Boot system command target options

Keyword	Description
flash:	On-board flash
slot0:	PCMCIA flash card in slot0

Keyword	Description
slot1:	PCMCIA flash card in slot1
mop	Load an image using the MOP protocol
bootflash:	Load bootflash image (not available on all systems)
rom:	Load the image from ROM
rcp:	Load an image using the RCP
tftp:	Load an image using the TFTP protocol

In addition to the boot system commands, you can use the router's configuration register to change which image the router boots from. The last octet in the configuration register must be set to "2" or the router will completely ignore the boot system commands. For instance, if the last octet of the configuration register is set to "1", the router will boot from ROM and ignore the boot system commands. In our example, the test router's configuration register was set to 0x2102. The config-register command allows you to set the appropriate configuration register values:

Router1#configure terminal

Enter configuration commands, one per line. End with CNTL/Z.

Router1(config)#config-register 0x2102

Router1(config)#end

Router1#

It is important to remember that, unlike any other configuration command, you don't need to save the running configuration to NVRAM when you change the configuration register setting. It will survive a reload without being saved. In fact, the new setting will not take effect until after the next reload:

Router1#show version

Cisco Internetwork Operating System Software

IOS (tm) 3000 Bootstrap Software (IGS-RXBOOT), Version 10.2(8a), RELEASE SOFTWAR

E (fc1)

Copyright (c) 1986-1995 by cisco Systems, Inc.

Compiled Tue 24-Oct-95 15:46 by mkamson

Image text-base: 0x01020000, data-base: 0x00001000

ROM: System Bootstrap, Version 5.2(8a), RELEASE SOFTWARE

Router1 uptime is 2 minutes

System restarted by reload

Running default software

cisco 2500 (68030) processor (revision D) with 16380K/2048K bytes of memory.

Processor board serial number 04915359 with hardware revision 00000000

X.25 software, Version 2.0, NET2, BFE and GOSIP compliant.

2 Ethernet/IEEE 802.3 interfaces.

2 Serial network interfaces.

32K bytes of non-volatile configuration memory.

16384K bytes of processor board System flash (Read/Write)

Configuration register is 0x2101 (will be 0x2102 at next reload)

Router1#

After setting the appropriate boot system commands and reloading the router, you can use the show version command to see which image file the router used to boot:

Router2#show version

Cisco Internetwork Operating System Software

IOS (tm) 3600 Software (C3620-IK9S-M), Version 12.2(13), RELEASE SOFTWARE (fc1)

Copyright (c) 1986-2002 by cisco Systems, Inc.

Compiled Tue 19-Nov-02 19:04 by pwade

Image text-base: 0x60008930, data-base: 0x61276000

ROM: System Bootstrap, Version 11.1(19)AA, EARLY DEPLOYMENT RELEASE SOFTWARE (fc1)

Router2 uptime is 2 hours, 4 minutes

System returned to ROM by reload

System restarted at 21:13:13 EST Wed Jan 15 2003

System image file is "slot0:c3620-ik9s-mz.122-13.bin"

cisco 3620 (R4700) processor (revision 0x81) with 41984K/7168K bytes of memory.

Processor board ID 05969532

R4700 CPU at 80Mhz, Implementation 33, Rev 1.0

Bridging software.

X.25 software, Version 3.0.0.

SuperLAT software (copyright 1990 by Meridian Technology Corp).

Basic Rate ISDN software, Version 1.1.

1 Ethernet/IEEE 802.3 interface(s)

1 FastEthernet/IEEE 802.3 interface(s)

1 Serial network interface(s)

1 ISDN Basic Rate interface(s)

DRAM configuration is 32 bits wide with parity disabled.

29K bytes of non-volatile configuration memory.

16384K bytes of processor board System flash (Read/Write)

16384K bytes of processor board PCMCIA Slot0 flash (Read/Write)

16384K bytes of processor board PCMCIA Slot1 flash (Read/Write)

Configuration register is 0x2102

Router2#

In this case, the router says that it loaded its IOS image from slot0:, as configured. After changing your boot system commands, you should make sure to reboot and verify that the router behaves as expected. You don't want the wrong IOS image to accidentally get loaded the next time the router reboots. If you do have problems with the boot system command, connect to the console and reload the router. This will display any error messages as the router boots. The router does not capture these messages anywhere; this is the only way to see them.

See Also

Recipe 1.6; Recipe 1.8

1.8. Booting Over the Network

Problem

You want to load an IOS image that is too large to store on your router's local flash.

Solution

You can load an IOS image that is larger than your router's flash by configuring the router to use TFTP to download the image before booting:

Router1#configure terminal

Enter configuration commands, one per line. End with CNTL/Z.

Router1(config)#boot system tftp c2500-io-l.122-7a.bin 172.25.1.1

Router1(config)#boot system flash

Router1(config)#end

Router1#

Discussion

We mentioned in Recipe 1.7 that it is possible to load IOS images over the network at boot time. However, booting from remote IOS images presents some unique challenges. This is why we have dedicated an entire recipe to remote booting.

One of the most important advantages of booting an IOS image over the network is that it allows you to use images that are larger than your router's flash. Like any other software, each new IOS image tends to be slightly larger than the previous versions. It is relatively common to discover that you can't load the latest IOS version because it is too big to fit in an older router's flash.

Booting over the network also provides a way of loading a backup IOS image if the primary image fails. As we discussed Recipe 1.7, it's a good idea to configure your router with at least one backup IOS image to load in case

the primary fails for any reason. Even if you have a lot of flash storage, you may find that you can't store two IOS images at once. So booting over the network is actually a reasonable way of providing a backup image.

Booting over the network also poses an important security problem because, as we discussed in Recipe 1.2, it's virtually impossible to secure a UDP-based service like TFTP. In addition, it makes the router dependent on the TFTP server for its boot images. Network booting also has performance issues. Loading an IOS over the network can significantly increase the time it takes your router to reload, particularly if it has to traverse slower WAN links. We certainly do not recommend relying solely on remote booting in a production environment.

However, in a lab or testing environment, the ability to load an IOS image that is larger than your router's flash can be extremely useful. Booting over the network lets you work with IOS versions that you could not otherwise load and test. The following show version command output is from a router that was booted this way:

Router1#show version

Cisco Internetwork Operating System Software

IOS (tm) 2500 Software (C2500-IO-L), Version 12.2(7a), RELEASE SOFTWARE (fc2)

Copyright (c) 1986-2002 by cisco Systems, Inc.

Compiled Thu 21-Feb-02 02:07 by pwade

Image text-base: 0x0000144C, data-base: 0x0082E874

ROM: System Bootstrap, Version 5.2(8a), RELEASE SOFTWARE

BOOTLDR: 3000 Bootstrap Software (IGS-RXBOOT), Version 10.2(8a), RELEASE SOFTWARE

(fc1)

Router1 uptime is 10 hours, 16 minutes

System returned to ROM by reload

System restarted at 01:57:47 EST Sat Jan 11 2003

System image file is "tftp://172.25.1.1/c2500-io-l.122-7a.bin"

cisco 2520 (68030) processor (revision E) with 16384K/2048K bytes of memory.

Processor board ID 03870281, with hardware revision 00000002

Bridging software.

X.25 software, Version 3.0.0.

Basic Rate ISDN software, Version 1.1.

1 Ethernet/IEEE 802.3 interface(s)

2 Serial network interface(s)

2 Low-speed serial(sync/async) network interface(s)

1 ISDN Basic Rate interface(s)

32K bytes of non-volatile configuration memory.

16384K bytes of processor board System flash (Read/Write)

Configuration register is 0x2102

Router1#

This shows that the router is running the new version of IOS, which it loaded using TFTP. In this example, we put the TFTP boot first:

Router1(config)#boot system tftp c2500-io-l.122-7a.bin 172.25.1.1

Router1(config)#boot system flash

If the TFTP file transfer had failed, the router would have loaded its old IOS image from its local flash. If we had reversed the order of these commands, the router would have tried first to boot from flash, then resorted to TFTP if it had trouble with the file on the flash.

For redundancy purposes, you can configure the router to boot from multiple TFTP servers. Simply copy the same IOS image to an alternate set of TFTP servers and include a boot system command for each server. This reduces the dependency of the router to a single TFTP server, but the router has to try each successive server and time out before moving on to the next one. This can increase the boot time.

See Also

Recipe 1.2; Recipe 1.7

1.9. Copying an IOS Image to a Server

Problem

You want to save a backup copy of your IOS image on a TFTP server.

Solution

You can upload a copy of your router's IOS image to a TFTP server with the following set of commands:

Freebsd% touch /tftpboot/c2600-ik9o3s-mz.122-12a.bin

Freebsd% chmod 666 /tftpboot/c2600-ik9o3s-mz.122-12a.bin

Freebsd% telnet Router1

Trying 172.25.1.5...

Connected to Router1.

```
Escape character is '^]'.


User Access Verification


Password: <vtypassword>


Router1>en

Password: <enablepassword>

Router1#copy flash:c2600-ik9o3s-mz.122-12a.bin tftp

Address or name of remote host []? 172.25.1.1

Destination filename [c2600-ik9o3s-mz.122-12a.bin]? <enter>

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

11135588 bytes copied in 52.588 secs (211752 bytes/sec)

Router1#
```

Discussion

It’s a good idea to save a copy of the current IOS image before attempting to upgrade the IOS version of a router. This way, if an upgrade fails or if you have problems with the new IOS version, you can revert back to the old proven IOS version. The procedure to copy an IOS image to a TFTP server is very similar to the way we backed up a configuration file in Recipe 1.2. The only real difference is the size of the file involved—IOS images are quite a bit larger than configuration files.

As we mentioned in Recipe 1.2, you have to verify the file permissions on your TFTP server. The transfer will fail if this file isn’t world writable. We highly recommend that you remove the world writable attribute on this file after uploading it. On Unix systems, you can use the chmod command to change the file attributes. This will ensure that the file isn’t accidentally overwritten. Unlike configuration files (which you should never store in your TFTP directory), world writeable IOS images pose no security concerns.

See Also

Recipe 1.2; Recipe 1.6

1.10. Copying an IOS Image Through the Console

Problem

You want to load an IOS image into your router through a serial connection to the console or AUX ports.

Solution

You can use the following set of commands to copy an IOS image onto a router through the console or the AUX port:

```
Router1#copy xmodem: slot1:
                **** WARNING ****
x/ymodem is a slow transfer protocol limited to the current speed
settings of the auxiliary/console ports. The use of the auxilary
port for this download is strongly recommended.
During the course of the download no exec input/output will be
available.
                ---- ******* ----


Proceed? [confirm] <enter>
Destination filename []? c3620-ik9s-mz.122-12a.bin
Erase slot1: before copying? [confirm] <enter>
Use crc block checksumming? [confirm] <enter>
Max Retry Count [10]: <enter>
Perform image validation checks? [confirm] <enter>
Xmodem download using crc checksumming with image validation
Continue? [confirm] <enter>
Ready to receive file.....CC  <start xmodem file transfer here>
4294967295 bytes copied in 1450.848 secs (1271445669961 bytes/sec)
Router1#
```

Note

Cisco highly recommends using the AUX port for this procedure rather than the console port because the AUX port supports hardware flow control.

Discussion

It can be quite useful to be able to load an IOS image through a serial connection, particularly if you don't have access to a TFTP server, or if the router doesn't have any accessible LAN interfaces. Although this feature is rarely used, Cisco supports xmodem and ymodem file transfers through a serial connection.

We also recommend enabling the CRC checksum feature when you use xmodem to download an IOS image through a serial connection. This will help to ensure the integrity of the file transfer.

We should stress that this process can be extremely slow. Don't even attempt to download an IOS image at the default speed of 9600bps unless you have an entire day to kill. We highly recommend increasing the speed to the highest value that your terminal emulation package will support. We have found that 115200bps provides the maximum throughput with the greatest reliability. The speed command allows you to change the speed of an asynchronous serial port:

Router1#configure terminal

Enter configuration commands, one per line. End with CNTL/Z.

Router1(config)#line aux 0

Router1(config-line)#speed 115200

Router1(config-line)#end

Router1#

In this example, we used Hyperterminal because it is included with the Windows operating system. However, almost any terminal emulation program that supports xmodem or ymodem protocols will work. In fact, we have found significant differences in download times between the various emulation packages, and HyperTerminal tends to be one of the slowest. Other packages such as ProComm tend to be somewhat faster. But they all work.

Even after we increased the speed of the AUX port to 115200bps the file transfer took nearly 25 minutes to complete. By comparison, loading the same IOS version via TFTP through an Ethernet connection took less than 4 minutes. So, in general, we don't recommend using this method unless you can't use TFTP.

The first step, once you have a copy of the IOS image on your computer, is to connect to the router's AUX port. Set the line speed to 115200bps on both the router's port and the terminal emulator, and issue the copy command. The router will prompt you to begin the file transfer with the text "Ready to receive file."

At this point, you should begin your file transfer protocol. If you are using HyperTerminal, select the "Transfer" drop-down menu, and then click on "Send-file." It will prompt you for filename and location and protocol type. Enter the name of the IOS image, then select "Xmodem" to start the file transfer.

During the file transfer, the connection is busy transferring the file, so the router can't display any messages. This is normal. However, most terminal emulator programs provide a status window to let you keep track of the file transfer.

When the transfer is complete, the terminal emulator drops out of the file transfer mode and the router puts up its normal prompt. At this point, we highly recommend checking the new IOS image to make sure that it copied successfully. You can verify the file size as follows:

Router1#show slot1:

PCMCIA Slot1 flash directory:

File Length   Name/status

  1   11922512 c3620-ik9s-mz.122-12a.bin

[11922576 bytes used, 4592496 available, 16515072 total]

16384K bytes of processor board PCMCIA Slot1 flash (Read/Write)

In this case, we loaded the image into the PCMCIA device in slot 1. If you put the image somewhere else, such as the internal flash memory, you would use the command show flash: instead.

If the file size is correct, you should check the image's checksum using the verify command:

Router1#verify slot1:c3620-ik9s-mz.122-12a.bin

Verified slot1:c3620-ik9s-mz.122-12a.bin

Router1#

1.11. Deleting Files from Flash

Problem

You want to erase files from your router's flash.

Solution

To delete all of the files from your router's flash memory, use the erase command:

Router1#erase slot1:

Erasing the slot1 filesystem will remove all files! Continue? [confirm] <enter>

Erasing device...

eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee

eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee ...erased

Erase of slot1: complete

Router1#

Note

Not all router types support the erase command.

You can remove individual files from the router's flash memory with the delete command:

Router1#delete slot1:c3620-ik9s-mz.122-13.bin

Delete filename [c3620-ik9s-mz.122-13.bin]? <enter>

Delete slot1:c3620-ik9s-mz.122-13.bin? [confirm] <enter>

Router1#

Discussion

As we have indicated, there are two ways to delete files from flash, depending on the type of router. The difference arose because Cisco routers use three different kinds of filesystems, called Class A, Class B, and Class C. Table 1-2 shows the filesystems that Cisco's most common routers use.

Table 1-2. Supported filesystems of common Cisco routers

Router type	Filesystem type
7000(RSP)	Class A
7500(RSP2,4, & 8)	Class A
12000	Class A
Route Switch Module (RSM)	Class A
1600	Class B
2500	Class B

Router type	Filesystem type
3600[a]	Class B
4000	Class B
AS5300	Class B
AS5800	Class C
7100	Class C
7200	Class C

[a] The 3600 traditionally uses the Class B filesystem. However, starting with IOS Version 12.2(4)T, the 3600 also includes Class C filesystem functionality.

Table 1-3 lists some of the different filesystem commands, their meanings, and the filesystems that they work with.

Table 1-3. Filesystem commands

Command	Filesystem	Description
Delete	All	Marks the file as deleted, but does not permanently remove it from flash
Squeeze	A	Permanently removes all files that have been marked as deleted
Format	A & C	Erases the entire flash device
Verify	All	Verifies that the IOS file's checksum matches the value encoded in the image

Command	Filesystem	Description
Undelete	A & B	Recovers deleted files
Erase	A & B	Erases the entire flash device

The erase command is not available on all router types. On routers that use the Class C filesystem, you can only remove files from the flash with the delete command.

The delete command marks files as deleted, but it does not permanently remove them:

Router1#show slot1:

PCMCIA Slot1 flash directory:

File Length Name/status

1 11992088 c3620-ik9s-mz.122-13.bin [deleted]

[16515072 bytes used, 0 available, 16515072 total]

16384K bytes of processor board PCMCIA Slot1 flash (Read/Write)

Router1#

You can permanently remove a file and reclaim the space on the flash device with the squeeze command. Note, however, that only routers with the type A filesystem support this command:

Router1#squeeze slot1:

Squeeze operation may take a while. Continue? [confirm] <enter>

squeeze in progress...

Squeeze of slot1 complete

Router1#

The squeeze function can take up to several minutes, so be patient. Once the squeeze command is complete, you can view the flash device to verify that the file is gone:

Router1#show slot1:

PCMCIA Slot1 flash directory:

No files in PCMCIA Slot1 flash

[0 bytes used, 16515072 available, 16515072 total]

16384K bytes of processor board PCMCIA Slot1 flash (Read/Write)

Router1#

The file has now been permanently removed and you can no longer recover it with the undelete command. On routers with filesystems that do not support the squeeze command, the only way to permanently remove deleted files is to use the erase command. However, the erase command deletes the entire flash system and will not permit you to delete individual files. In the next recipe, we look at ways to partition flash devices in order to reduce the impact of the erase command.

See Also

Recipe 1.12

1.12. Partitioning Flash

Problem

You want to change how your router's flash memory is partitioned.

Solution

The partition command allows you to create a partition in the router's flash memory:

Router1#configure terminal

Enter configuration commands, one per line. End with CNTL/Z.

Router1(config)#partition slot1: 2 8 8

Router1(config)#end

Router1#

Discussion

As we discussed in Recipe 1.11, the erase command deletes the entire contents of a flash device. On routers that don't support the delete and squeeze commands, there is no way to delete an individual file from flash without erasing all of the files in the flash device. Fortunately, you can use the partition command on flash devices to shelter some files from the effects of the erase command.

After you have partitioned a flash device, the erase command only affects one partition at a time. This command doesn't affect any of the other partitions on the same flash device. You can use this to allow you to delete individual files without having to wipe out the entire flash device.

In the next example, we partitioned a flash device into two equal parts. We then stored an IOS image on each of the partitions. You can see the partitions and their contents with the following command:

```
Router1#show slot1:


PCMCIA Slot1 flash directory, partition 1:

File  Length   Name/status

  1   7723664  c3620-ajs56i-mz.120-25.bin

[7723728 bytes used, 664880 available, 8388608 total]

8192K bytes of processor board PCMCIA Slot1 flash (Read/Write)


PCMCIA Slot1 flash directory, partition 2:

File  Length   Name/status

  1   7723664  c3620-ajs56i-mz.120-25.bin

[7723728 bytes used, 402736 available, 8126464 total]

8192K bytes of processor board PCMCIA Slot1 flash (Read/Write)


Router1#
```

Note that the router treats the two partitions as if they were separate flash devices. You can erase the contents of a particular partition by specifying the flash device name followed by the partition number and a colon:

```
Router1#erase slot1:2:

Erasing the slot1:2 filesystem will remove all files! Continue? [confirm] <enter>

Erasing device... eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee

...erased

Erase of slot1:2: complete

Router1#
```

If you view the entire flash device again, you can see that the file in partition 2 has been erased, while the contents of partition 1 remain untouched:

```
Router1#show slot1:
```

PCMCIA Slot1 flash directory, partition 1:

File Length Name/status

 1 7723664 c3620-ajs56i-mz.120-25.bin

[7723728 bytes used, 664880 available, 8388608 total]

8192K bytes of processor board PCMCIA Slot1 flash (Read/Write)

PCMCIA Slot1 flash directory, partition 2:

No files in PCMCIA Slot1 flash

[0 bytes used, 8126464 available, 8126464 total]

8192K bytes of processor board PCMCIA Slot1 flash (Read/Write)

Router1#

As we mentioned in Recipe 1.11, attempting to erase one file from this flash device without partitioning it first causes the router to erase both IOS images.

You can remove an existing set of partitions with the no partition command:

Router1#configure terminal

Enter configuration commands, one per line. End with CNTL/Z.

Router1(config)#no partition slot1: 2 8 8

ee

Router1(config)#end

Router1#

You can safely partition a flash device that already contains files as long as you don't attempt to create a partition partway through an existing file. If you attempt to create a partition that divides an existing file, the router will identify it as a problem.

See Also

Recipe 1.6; Recipe 1.11

1.13. Using the Router as a TFTP Server

Problem

You want to configure your router to act as a TFTP server.

Solution

The tftp-server command configures the router to act as a TFTP server:

```
Router1#configure terminal

Enter configuration commands, one per line.  End with CNTL/Z.

Router1(config)#tftp-server flash:c2600-ik9o3s-mz.122-12a.bin

Router1(config)#end

Router1#
```

Discussion

The ability to use a router as a TFTP server can be quite useful. We have often used this feature to upgrade several routers that are separated from the TFTP server by slow WAN connections. In situations like this, you can upgrade one of the remote routers using TFTP over the slow WAN connection as we described in Recipe 1.6. Then you can configure this router to act as a TFTP server, and use it to upgrade the remaining routers over high-speed local links.

However, the router is not a fully functional TFTP server. It can only serve files for download. You cannot use this feature to upload files into the serving router's local flash. The router is not limited to just serving IOS images: you can use your router's flash to store configuration files and make them available for download via TFTP as well. You can even use it to hold configuration files for non-Cisco equipment.

Security is a concern whenever you enable services on a router. Every extra service you enable provides the wily hacker with a new potential avenue to exploit against your network. Therefore, we don't recommend using the TFTP server feature on routers facing the public Internet or other potentially unfriendly networks. However, for internal use, we believe it is reasonably safe. You can increase the security of the router's TFTP server by using an access list like this:

```
Router1#configure terminal

Enter configuration commands, one per line.  End with CNTL/Z.

Router1(config)#access-list 99 permit 172.25.1.0 0.0.0.255

Router1(config)#access-list 99 deny any

Router1(config)#tftp-server flash:c2600-ik9o3s-mz.122-12a.bin 99

Router1(config)#end

Router1#
```

In this example, we defined an access list, 99, that will allow all devices on the 172.25.1.0/24 network to access the router's TFTP server. Then we applied the access list to the TFTP service by specifying the access list number at the

end of the tftp-server command line. This will help to ensure that only the authorized devices permitted by the access list may download the specified file via TFTP.

You can configure the router to serve multiple files via TFTP by simply adding more tftp-server commands. If security is a concern, you can configure a different access list for each file.

Although this feature can be useful, we recommend enabling it only when you need to perform a download. Disabling the service as soon as the download has completed mitigates the security concerns of running extra services from your router.

See Also

Recipe 1.6

1.14. Using FTP from the Router

Problem

You want to use FTP directly from your router to download configuration or IOS files.

Solution

The copy ftp: command lets the router exchange files using FTP:

```
Router1#configure terminal

Enter configuration commands, one per line.  End with CNTL/Z.

Router1(config)#ip ftp username ijbrown

Router1(config)#ip ftp password ianpassword

Router1(config)#end

Router1#copy ftp: running-config

Address or name of remote host [172.25.1.1]? 172.25.1.1

Source filename [ ]? test

Destination filename [running-config]? <enter>

Accessing ftp://172.25.1.1/test...

Loading /test

[OK - 24/4096 bytes]


24 bytes copied in 0.276 secs (87 bytes/sec)

Router1#
```

We explicitly defined a username and password in this example. If you don't specify a username, the router will try to connect to the server's anonymous FTP service.

Discussion

Several recipes in this chapter show how to transfer files between your router and server using TFTP. However, Cisco routers also support FTP, which is better suited for transferring files over busy and congested links. While TFTP file transfers tend to abort if they encounter persistent congestion, FTP appears to be more resilient.

FTP is also somewhat more secure than TFTP because it uses usernames and passwords. TFTP has no user level security features. However, FTP sends its passwords across the network in unencrypted cleartext, so it is still not highly secure.

In the example we explicitly configured a FTP username and password on the router. Once this information is defined, using FTP is as easy as using TFTP. You can also override the username and password settings defined in the configuration file by including them on the command line:

Router1#copy ftp://ijbrown:ianpassword@172.25.1.1/c3620-ik9s-mz.122-10a.bin slot1:

Destination filename [c3620-ik9s-mz.122-10a.bin]? <enter>

Accessing ftp://ijbrown:ianpassword@172.25.1.1/c3620-ik9s-mz.122-10a.bin...

Loading pub/c3620-ik9s-mz.122-10a.bin !!!!

Erase slot1: before copying? [confirm] <enter>

Erasing the slot1 filesystem will remove all files! Continue? [confirm] <enter>

Erasing device... eeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee ...erased

Erase of slot1: complete

Loading pub/c3620-ik9s-mz.122-10a.bin

!!

!!

!!

[OK - 11819052/4096 bytes]

Verifying checksum... OK (0x3238)

11819052 bytes copied in 266.956 secs (44273 bytes/sec)

Router1#

You can use URL format to specify the username, password, server address, and file you want to download. The format of the FTP URL looks like this:

ftp://ijbrown:ianpassword@172.25.1.1/c3620-ik9s-mz.122-10a.bin

A colon separates the username, ijbrown, from the password, ianpassword. An @ sign then separates the user information from the server information, which can be either an IP address or a DNS name. A forward slash (/) separates the server name or address from the directory and filename.

If you don't include an FTP username in the configuration or the command line, the router will default to using anonymous FTP. If no password is specified in either the router's configuration or on the command line, the router will use a default password of mailto:router@cisco.com.

It is important to remember that if you specify a username and password on the command line, it will override whatever values you have configured. If you don't specify a username or password on the command line, the router will use the configured FTP username and password. And, if you don't specify username and password in either place, the router will resort to anonymous FTP.

See Also

Recipe 1.1; Recipe 1.2; Recipe 1.6; Recipe 1.10

1.15. Generating Large Numbers of Router Configurations

Problem

You need to generate hundreds of router configuration files for a big network rollout.

Solution

When building a large WAN, you will usually configure the remote branch routers similarly according to a template. This is a good basic design principle, but it also makes it relatively easy to create the router configuration files. Example 1-1 uses a Perl script to merge a CSV file containing basic router information with a standard template file. It takes the CSV file as input on STDIN.

Example 1-1. create-configs.pl

```
#!/usr/local/bin/perl

#

$template_file_name="rtr-template.txt";

while(<>) {


  ($location, $name, $lo0ip, $frameip, $framedlci, $eth0ip, $x)

     = split (/,/);
```

```
  open(TFILE, "< $template_file_name") || die "config template file $template_file_name:
$!\n";
  $ofile_name = $name . ".txt";
  open(OFILE, "> $ofile_name") || die "output config file $ofile_name: $!\n";

  while (<TFILE>) {

    s/##location##/$location/;
    s/##rtrname##/$name/;
    s/##eth0-ip##/$eth0ip/;
    s/##loop0-ip##/$lo0ip/;
    s/##frame-ip##/$frameip/;
    s/##frame-DLCI##/$framedlci/;

    printf OFILE $_;
  }
}
```

Discussion

This Perl script is a simplified version of much longer scripts that we have used to create the configuration files for some very large networks. After loading these configuration files into the routers, we shipped them to the remote locations along with a hard copy of the configuration in case there were problems during shipment. The technician installing the router could then simply connect the appropriate cables and power on the router. He wouldn't even need to log on to the router's console unless there were unexpected problems. This methodology can save hundreds of hours in a network installation project.

The script does a relatively simple merge function and expects the input data in CSV format on STDIN. For an input file named RTR-DATA.CSV, run the script as follows:

```
Freebsd% create-configs.pl < RTR-DATA.CSV
```

The input file in this case might look something like this:

```
Toronto, Router1, 172.25.15.1, 172.25.16.6, 101, 172.25.100.1,
```

Boston, Router2, 172.25.15.2, 172.25.16.10, 102, 172.25.101.1,

San Francisco, Router3, 172.25.15.3, 172.25.16.14, 103, 172.25.102.1,

Using a CSV file like is convenient because you can keep track of the entire network in a spreadsheet and create a CSV file containing the data you need for the router configurations.

The template configuration needs to include unique variable names that the script will replace with values from the CSV file. For example, the template configuration file might look like this:

```
!
version 12.1
service timestamps debug datetime msec
service timestamps log datetime msec
service password-encryption
!
hostname ##rtrname##
!
enable password cisco
enable secret cisco
!
interface Loopback0
 ip address ##loop0-ip## 255.255.255.255
!
interface Serial0/0
 description Frame-Relay Circuit
 no ip address
 encapsulation frame-relay
 ip route-cache policy
 frame-relay lmi-type ansi
 no shutdown
!
```

```
interface Serial0/0.1 point-to-point
 ip address ##frame-ip## 255.255.255.252
 frame-relay interface-dlci ##frame-DLCI##
!
interface FastEthernet0/1
 description User LAN Segment
 ip address ##eth0-ip## 255.255.255.0
 no shutdown
!
router eigrp 99
 network 172.25.0.0
!
snmp-server location ##location##
!
line con 0
 password cisco
 login
 transport input none
line aux 0
 password cisco
 login
line vty 0 4
 password cisco
 login
 transport input telnet
!
end
```

The script expects to find this template file located in the current directory with the name rtr-template.txt by default, but you can change this easily by modifying the variable called template_file_name in the script.

Naturally, your router templates will not look like this one, and your CSV file will almost certainly contain other data that is important to your network. This script will probably need significant local modification every time you use it on a new network, but the amount of time required to modify the script is usually far less than the amount of time needed to create all of the configuration files by hand.

The output of this script will be a series of files whose names are the same as the router names, but with .txt added to the end. You can then use a terminal emulator to cut and paste the router configuration files into the routers prior to shipping them to their destinations. Always remember to save the configuration to the router's NVRAM before powering it off:

Router1#copy running-config startup-config

See Also

Recipe 1.16.

1.16. Changing the Configurations of Many Routers at Once

Problem

You want to make a configuration change to a large number of routers.

Solution

The Expect script in Example 1-2 makes the same configuration changes to a list of routers using Telnet. When it finishes running, the script produces a status report that identifies which devices, if any, failed to update properly. No arguments are required or expected.

Example 1-2. rtrchg.exp

```
#!/usr/local/bin/expect
#
#    rtrcfg.exp -- a script to perform mass configuration changes to
#                  a list of routers using Telnet and Expect
#
#
# Set Behavior
set tftp "172.25.1.1"
set workingdir /home/cisco/rtr
#
```

```
puts stdout "Enter user name:"
gets stdin userid
system stty -echo
puts stdout "Enter login password:"
gets stdin vtypasswd
puts stdout "\nEnter enable password:"
gets stdin enablepwd
system stty echo
system "cp $workingdir/NEWCONFIG /tftpboot/NEWCONFIG"
set RTR [open "$workingdir/RTR_LIST" r]
set LOG [open "$workingdir/RESULT" w]
while {[gets $RTR router] != -1} {
  if {[ string range $router 0 0 ] != "#"} {
    set timeout 10
    spawn telnet; expect "telnet>"; send "open $router\n"
    expect {
          {Username}   { send "$userid\r"
                       expect {
                              {*Password*} { send "$vtypasswd\r" }
                             }
                      }
          {Password}   { send "$vtypasswd\r" }
          timeout      { puts $LOG "$router - telnet failed"
                       close; wait; continue
                      }
        }
```

```
expect {
      {Password}    { puts $LOG "$router - vty login failed"
                      close; wait; continue
                    }
      {Username}    { puts $LOG "$router - vty login failed"
                      close; wait; continue
                    }
      {>}           { puts $LOG "$router - vty login ok" }

      timeout       { puts $LOG "$router - vty login failed"
                      close; wait; continue
                    }
    }

send "enable\r"
expect "Password"
send "$enablepwd\r"
#
expect {
       {*#}      { puts $LOG "$router - enable login ok" }

       {*>}      { puts $LOG "$router - enable login failed"
                   close; wait; continue
                 }

       timeout     { puts $LOG "$router - enable login failed"
                     close; wait; continue
```

```
                }
        }
    # CMDs
    set timeout 30
    send "copy tftp://$tftp/NEWCONFIG running-config\r"
    expect "running-config"
    send "\r"
    expect  {
          {OK}       { puts $LOG "$router - TFTP successful"}
          timeout    { puts $LOG "$router - TFTP failed"
                close; wait; continue }
        }
    send "copy running-config startup-config\r\r\r"
    expect  {
          {OK}       { puts $LOG "$router - config saved"}
          timeout    { puts $LOG "$router - config failed"
                close; wait; continue }
        }
    #CMDs
    send "exit\r"; close; wait
   }
}
close $RTR; close $LOG
system "rm /tftpboot/NEWCONFIG"
```

Discussion

This script uses the Expect language to emulate how a human router engineer might perform a series of configuration updates via TFTP. The script logs into each router in a list and uses TFTP to download a set of configuration changes into the router's running configuration. It then saves the new configuration file to NVRAM

and moves on to the next router in the list. Automating a routine but time-consuming procedure with a script like this saves time and decreases the chances of fatigue-induced errors.

The script is designed to work with either normal router passwords or the AAA-enabled username and password combinations described in Chapter 3. The script begins by asking you to supply a username. If one or more of your routers aren't configured to use AAA or local authentication, the script simply ignores this username and inserts the password.

After asking for a username, the script prompts you to enter your VTY or AAA password. Then the script asks for the enable password and begins to perform the required configuration changes.

Note

The script is designed to work with IOS versions 12.0 and greater. Unfortunately, Cisco changed the command sequence prior to 12.0, meaning the script will not work correctly with old IOS versions.

You must change two variables in this script for it to work in your network. The first variable is called tftp. You must set this value to your TFTP server's IP address. The second variable you need to change is workingdir, which must contain the name of the directory that holds the list of routers, the file of configuration changes, and where the script will put its report.

The script is written in the Expect language and requires Expect to be loaded on your server and available in the /usr/local/bin directory. For more information on the Expect language, see Appendix A or Exploring Expect (O'Reilly).

The script expects to find two files in the working directory. The first, RTR_LIST, contains a list of router names, with one name on each line. The second file, NEWCONFIG, contains all of the required configuration changes. We also recommend that you put the configuration command end on the last line of the NEWCONFIG file to avoid the error messages that we mentioned in Recipe 1.1.

Here is a typical example of the sort of things you might put in the configuration file:

```
Freebsd% cat NEWCONFIG

enable secret cisco

end
```

Using this file, the script will log into all of the routers and change the enable secret password. You could put any number of commands in this configuration file, but because the exact same set of configuration commands will be downloaded into every router, you should never change anything that is unique to a particular router (such as an IP address) this way. This method is perfect for changing passwords, SNMP community strings, access lists, and other things that can be the same on all routers.

The script will copy the NEWCONFIG file into the /tftpboot directory so it can use TFTP to transfer it to each router. It is a good idea to ensure that your server's TFTP is working correctly before launching the script.

When the script finishes, it creates a status file called RESULT in the working directory. This status file contains detailed status reports of what happened on each router. The easiest way to see a list of the routers that the script failed to change is to use the Unix grep command to search for the keyword "fail":

Freebsd% grep fail RESULT

toronto - enable login failed

boston - telnet failed

test   - enable login failed

frame   - enable login failed

See Also

Recipe 1.1; Appendix A; Exploring Expect, by Don Libes (O'Reilly)

1.17. Extracting Hardware Inventory Information

Problem

You need an up-to-date list of the hardware configurations and IOS levels of all of your routers.

Solution

The Bourne shell script in Example 1-3 uses SNMP to extract useful version information from a list of routers. By default, the script stores this data in CSV format so that you can easily import it into a spreadsheet for analysis. No arguments are required or expected.

Example 1-3. inventory.sh

```
#!/bin/sh

#

#    inventory.sh -- a script to extract valuable information

#                    from a list of routers. (Name, Type, IOS version)

#

#

# Set behaviour

public="ORARO"

workingdir="/home/cisco"

#

LOG=$workingdir/RESULT.csv

infile=$workingdir/RTR_LIST

snmp="/usr/local/bin/snmpget -v1 -c $public"
```

```
#
while read device
do
  $snmp $device sysName.0 > /dev/null
  if [ "$?" = "0" ] ; then
    rtr=`$snmp $device .1.3.6.1.4.1.9.2.1.3.0 | cut -f2 -d\" `
    type2=`$snmp $device .1.3.6.1.4.1.9.9.25.1.1.1.2.3 | cut -f2 -d$ `
    ios=`$snmp $device .1.3.6.1.4.1.9.9.25.1.1.1.2.5 | cut -f2 -d$ `
    prot=`$snmp $device .1.3.6.1.4.1.9.9.25.1.1.1.2.4 | cut -f2 -d$ `
    echo "$device, $rtr, $type2, $ios, $prot" >> $LOG
  fi
done < $infile
```

Discussion

The inventory.sh script extracts hardware and IOS version information directly from the routers using SNMP. This ensures that the data is up-to-date. You can even automate this script to run periodically, ensuring that your inventory information is always accurate. In a large network, this is much easier than keeping track of this information manually.

By default, the script captures the device name, router type, IOS version, and IOS feature set from each router. It stores this gathered information in a CSV format file called RESULT.csv.

This script requires NET-SNMP to gather the information via SNMP. You can use a different SNMP package if you prefer, but then you will need to modify the syntax appropriately. The script expects to find the executable snmpget in the /usr/local/bin directory. Again, if you keep this file in a different location, you will need to define the correct location in the snmp variable. For more information on NET-SNMP, see Chapter 17 or Appendix A.

Before running this script in your network, you will need to modify two variables. The first is the public variable. This value must contain your read-only SNMP community string. The script assumes that you have the same community string on all of the routers in the list. The second variable that you will need to set is workingdir, which must contain the name of the directory that you wish to run the script from.

Finally, you will need to build a file called RTR_LIST that contains the names of all of your routers, with one name on each line. The script expects to find this file in the working directory.

The output of the script is a CSV file, which you can import into a spreadsheet to analyze and sort the results as required. Table 1-4 shows an example of the script's output as it might look in a spreadsheet.

Table 1-4. Output results from the inventory.sh script

Router	Router Name	Type	IOS version	IOS feature set
Toronto	toronto	C2600	12.2(13)	IP|FIREWALL 2 PLUS 3DES
Boston	boston	C3620	12.2(13)	IP|3DES PLUS
Newyork	newyork	C3620	12.2(13)	IP|3DES PLUS
Tampa	tampa	C2500	12.0(25)	IP PLUS
Sanfran	sanfran	C7200	12.0(12a)	IP|IPX|VLAN

See Also

Chapter 17; Appendix A

1.18. Backing Up Router Configurations

Problem

You need to download all of the active router configurations to see what has changed recently.

Solution

The Perl script in Example 1-4 will automatically retrieve and store router configuration files on a nightly basis. By default, it will retain these configuration files for 30 days. The script should be run through the Unix cron utility to get the automatic nightly updates, but you can also run it manually if required. No arguments are required or expected.

Example 1-4. backup.pl

```
#!/usr/local/bin/perl

#

#       backup.pl -- a script to automatically backup a list of

#                    router configuration files on a nightly basis.

#

#

# Set behaviour
```

```
$workingdir="/home/cisco/bkup";

$snmprw="ORARW";

$ipaddress="172.25.1.1";

$days="30";

#

#

$rtrlist="$workingdir/RTR_LIST";

$storage="$workingdir/storage";

$latest="$storage/LATEST";

$prev="$storage/PREV";

if (! -d $storage) {mkdir ($storage, 0755)};

if (! -d $prev) {mkdir ($prev, 0755)};

if (! -d $latest) {mkdir ($latest, 0755)};

($sec, $min, $hr, $mday, $mon, $year, @etc) = localtime(time);

$mon++; $year=$year+1900;

$today1=sprintf("%.4d_%.2d_%.2d", $year, $mon, $mday);

$today="$storage/$today1";

system("cp -p $latest/* $prev/");

unlink <$latest/*>;

mkdir ($today, 0755);


open (RTR, "$rtrlist") || die "Can't open $rtrlist file";

open (LOG, ">$workingdir/RESULT") || die "Can't open $workingdir/RESULT file";

print LOG "Router Configuration Backup Report for $year/$mon/$mday\n";

print LOG "================================================================================\n";

print LOG "Device Name                    Status\n";

print LOG "================================================================================\n";
```

```
while (<RTR>) {
 chomp($rtr="$_");
 $oid=".1.3.6.1.4.1.9.2.1.55.$ipaddress";
 $snmpset ="/usr/local/bin/snmpset -v1 -c $snmprw -t60 -r2 $rtr";
 $rtrfile="/tftpboot/$rtr.cfg";
 unlink $rtrfile;
 open (CFG, ">$rtrfile"); print CFG " ";close CFG;
 chmod 0666, $rtrfile;
 chop ($status=`$snmpset $oid s $rtr.cfg`);
 $status=~/.+ = "(.+)".*$/;
 if($1 eq "$rtr.cfg") {
   if( -z "$rtrfile" ) {
     $result="not ok (File empty)";
     unlink $rtrfile;
   }
   else {
     $result="ok";
     chmod 0444, $rtrfile;
     system("mv $rtrfile $latest");
   }
 }
 else {
   $result="not ok";
   unlink $rtrfile;
 }

printf LOG ("%-28s      %-28s\n", $rtr,$result);
```

```
}

system ("cp -p $latest/*cfg $today");

$time=$days*86400;

print "$time\n";

($sec, $min, $hr, $mday, $mon, $year, @etc) = localtime(time-$time);

$mon++; $year=$year+1900;

$rmdir=sprintf("%s/%.4d_%.2d_%.2d",$configs, $year, $mon, $mday);

system ("rm -r -f $storage/$rmdir");
```

Discussion

As we mentioned earlier in the chapter, it is extremely important to make regular backup copies of your router configuration files. However, as the size of your network grows, it becomes quite tedious to maintain a useful archive of these backups. This script automates the task of collecting and storing router configuration files on a Unix-based TFTP server.

This script will maintain 30 days worth of configuration files. We have found that this is a reasonable length of time, allowing engineers to recover router configuration files that are up to one month old. However, if you prefer, you can change the $days variable to increase or decrease how long the script will store these files before deleting them. If you increase the length of time that the server must store these files, it will obviously increase the amount of disk space you need to hold the extra configuration files. But router configuration files are generally quite small, so this is usually not a serious problem unless you support thousands of routers.

Before executing this script, you will need to modify a few variables. First, the $workingdir variable should contain the name of the directory in which the server will run the script. Second, the $snmprw variable must contain your SNMP read-write community string. Please note that the read-only community string will not allow you to copy a configuration file; you must use the read-write string. The other variable you need to change is $ipaddress, which should contain the IP address of your TFTP server.

The script is written in Perl, and it makes a few system calls out to Bourne shell commands. The script expects to find the Perl executable in the /usr/local/bin directory. The script is also dependent on NET-SNMP and it expects to find the executable snmpset in the /usr/local/bin directory as well. If these files are in different locations on your local system, you will need to modify these paths. See Appendix A for more information on Perl and NET-SNMP.

Finally, you will need a file called RTR_LIST that contains the list of router names. This file must be in the working directory.

As we mentioned earlier, you should run this backup script from the Unix cron utility on a nightly basis. This ensures that you have an up-to-date backup of your configuration files. We recommend launching this script during the off-hours since it does generate traffic across your network, as well as causes a small amount of CPU loading on the routers. Here is an example crontab entry to start the script every night at 1:30 A.M.:

30 1 * * * /home/cisco/bkup/backup.pl

When the script runs, it creates a new directory called storage under the working directory. Under this directory, the script creates several subdirectories, including LATEST, PREV, and dated directory names (such as 2003_01_28). The LATEST directory always contains the most up-to-date router configuration files, and you can find the previously stored version of each router's configuration in the PREV directory. The dated directories contain all of the router configuration files that were captured on the date indicated in the directory name.

You can use the Unix diff command to see what changes have occurred on a given router.

Finally, the script creates a nightly status report that it stores in a file called RESULT in the working directory:

Freebsd% cat RESULT

Router Configuration Backup Report for 2003/1/28

==========

Device Name Status

==========

toronto ok

boston not ok

test ok

frame ok

With a slight modification, you can configure the script to email this report to the responsible engineer. However, since each different Unix flavor uses a different mail program, we chose not to include it here in the interest of compatibility. On a Solaris server, for example, you could add the following line to the bottom of the script to mail this report:

system ("/usr/ucb/mail -s \"Config Report for $today1\" `/bin/cat $mail` <

$workingdir/RESULT");

In this case, you would need to define the variable $mail to be an email distribution list for the report. For other Unix or Linux variants, please consult your manpages for more information on your local mail program.