CHAPTER 6

Deploying Basic Security Services

Your ability to leverage the security services of IOS is crucial to running a highly effective and robust Cisco network. Security services enforce your organization's policies and safeguard your network against misuse.

Proper deployment of IOS security increases the utility and improves the stability of the network. Remote access security, for example, enables your network to reach mobile users and telecommuters securely, and services that guard against hacking attacks preserve the integrity of networking devices and increase reliability of the network service. In the end, enhancing security increases the trust users have in the network and enables your organization to deploy applications over the network with greater confidence.

This chapter explains some basic concepts and common practices that are key to maintaining security on a Cisco network. Chapter 7, "Advanced Security Services, Part I: IPsec," and Chapter 8, "Advanced Security Services, Part II: IOS Firewall Feature Set," extend the discussion with more advanced security services.

The main topics of this chapter are

- Controlling Traffic with Access Control Lists
- Securing Access to the Router
- Deploying Authentication, Authorization, and Accounting
- Other IOS Commands for Basic Security

Controlling Traffic with Access Control Lists

The *access control list (ACL)*, or *access list*, is a key IOS feature whose programming syntax is used across many Cisco features. As the name implies, access lists are commonly used as security filters to block traffic from entering or exiting parts of the network. Over the years, however, Cisco has extended the access list syntax to other features, such as

- Routing and routing filters (see Chapter 3, "Managing Routing Protocols")
- Packet classification for QoS and queuing (see Chapter 4, "Deploying Basic Quality of Service Features," and Chapter 5, "Deploying Advanced Quality of Service Features")

- Encryption (see Chapter 7)
- Dial-on-demand routing (see the IOS Configuration Guides)

An understanding of access lists is fundamental to designing and maintaining a Cisco network.

This section covers

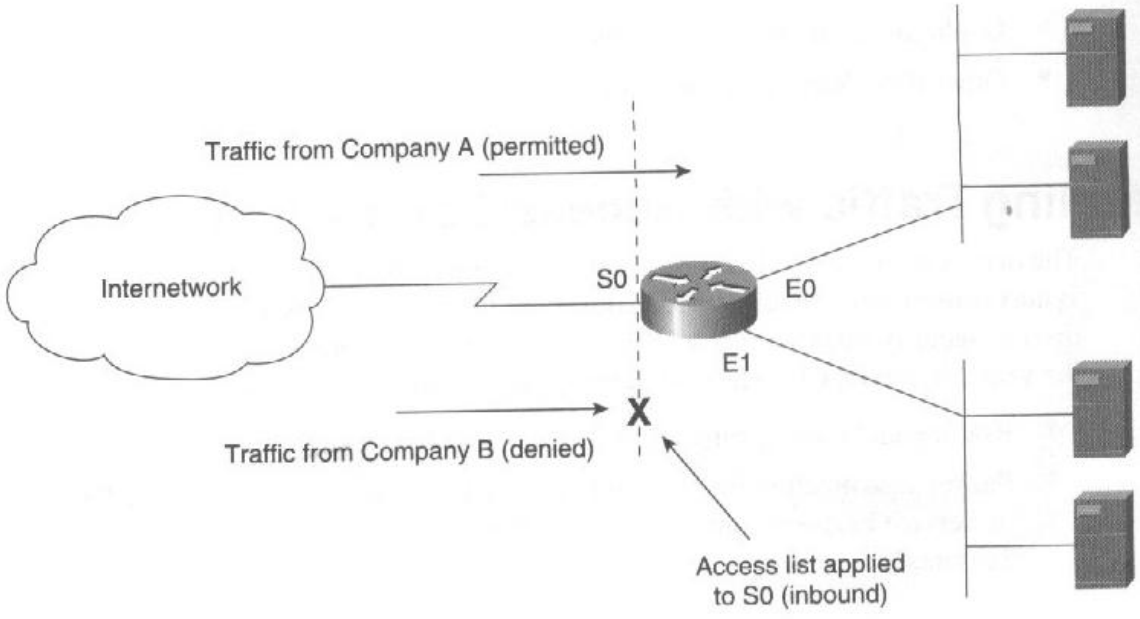
- Filtering Traffic with Access Lists
- Standard IP Access Lists
- Important Points for Designing Access Lists
- The Invisible Rule in Every Access List
- Extended IP Access Lists
- Access Lists for Combating Spoofing Attacks

Filtering Traffic with Access Lists

The most popular use of an access list is for filtering traffic on a router interface. As a traffic filter, the access list defines the traffic you want to permit and deny through an interface. After writing an access list that meets your filtering criteria, you apply it to the appropriate router interface (or multiple router interfaces). The router inspects traffic flowing through the interface and rejects the packets that are denied by your access list. Packets are filtered in either the inbound or outbound direction on an interface. (To filter in both directions, apply two access lists; see "Important Points for Designing Access Lists" later in this chapter.)

Figure 6-1 shows a simple application of an access list that inspects traffic inbound on interface Serial0. The purpose of this access list is to deny traffic from Company B.

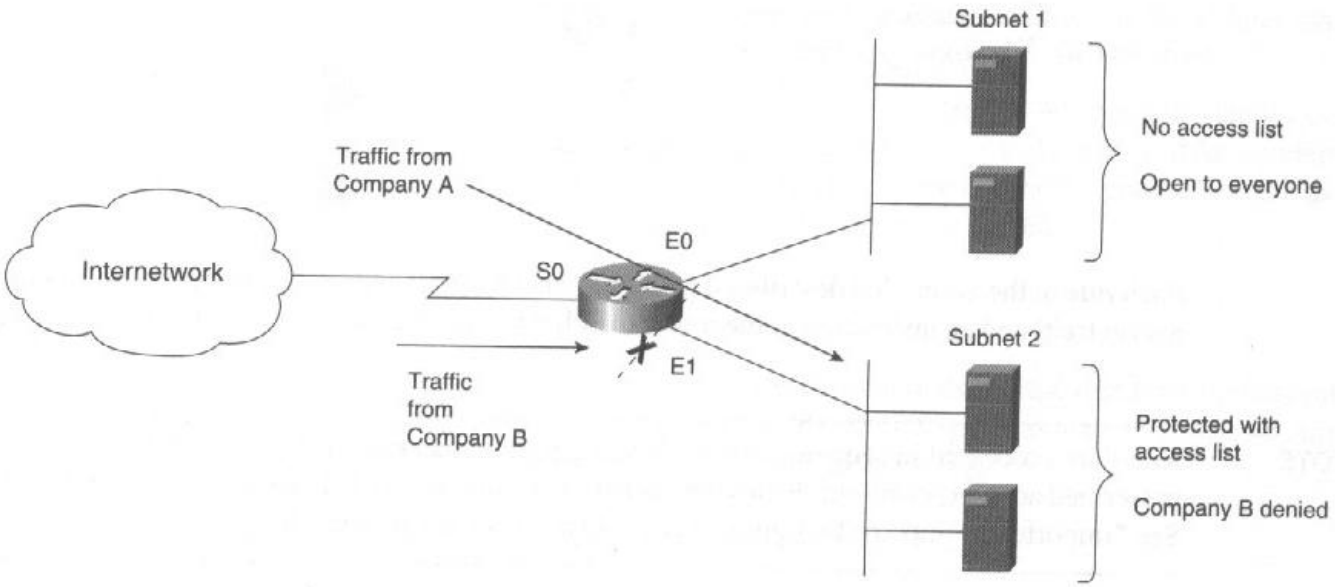
Figure 6-1 *A Simple Application of an Inbound Access List*



Company A's traffic is allowed through the interface and can reach the two network segments to the right of the router (assuming no other access lists are in effect), but Company B is denied access. Traffic not originating from Company A or Company B will be denied or permitted depending on the policy and how the access list was written. For example, the policy might dictate that only Company A's traffic should be allowed through the router and all other traffic (including Company B's) must be denied. On the other hand, if Company B is the real concern, the policy might allow everyone *except* Company B through the router. The syntax of access lists allows for flexible rules to cover these and more complex criteria. The access list in Figure 6-1 is an example of an *inbound access list*, which inspects packets entering a router's interface.

Figure 6-2 is an application of an *outbound access list* applied to an interface. The outbound access list filters traffic *exiting* a router interface.

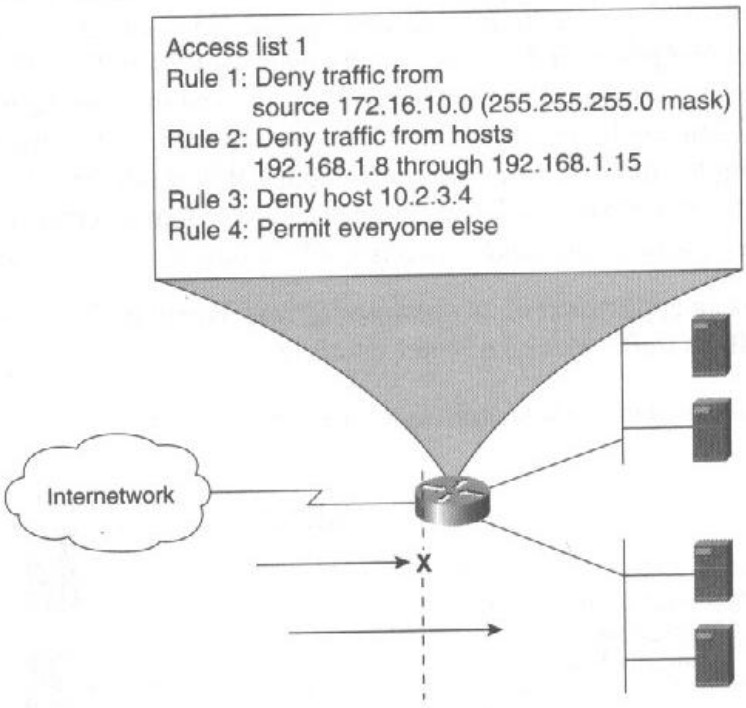
Figure 6-2 *A Simple Application of an Outbound Access List*



In this example, the outbound access list is applied to the router's Ethernet1 interface. Now, Company B's traffic is denied to subnet 2 only. All traffic, including traffic from Company B, is allowed to subnet 1 because there are no access lists denying traffic into Serial0 or out of Ethernet0.

Figure 6-3 is a conceptual view of a fictitious access list and illustrates the internal logic. An access list is a sequential progression of rules that describe your filtering policy.

Figure 6-3 *Conceptual View and Internal Logic of an Access List*



Each rule in the access list describes a pattern to match (an IP address, a protocol type, or other packet trait) and an instruction to the router to either permit (transmit) or deny (drop) the packet.

NOTE Rules are processed in sequential order. When a packet matches a rule, the router takes the prescribed action (permits or denies the packet) and stops access-list processing for that packet. See "Important Points for Designing Access Lists" later in this chapter.

Access lists are very flexible; you can configure them to be granular or broad, depending on your requirements. For example, you can define an access list to deny a specific protocol from a single host, an entire network address space, or even all packets.

When you create an access list, you must give it a number (or a name in the case of named access lists—see the IOS Configuration Guides for named access lists). The number identifies the access list as a single entity in the router so it can be conveniently applied to one or more interfaces. This means you create an access list only once and then apply it to as many interfaces as needed.

There are restrictions on what numbers you can use, depending on the protocol being filtered. For IP, two types of access lists exist, each with its own number range:

- **Standard IP access lists**—To which you may assign numbers 1–99.
- **Extended IP access lists**—To which you may assign numbers 100–199.

Other number ranges for access lists (600–699, 800–899, and so on) are reserved for filtering such protocols as IPX, AppleTalk, DECnet, and XNS. The scope of this book is IP and so will focus on IP access lists: the standard IP access list and the extended IP access list. See the IOS Configuration Guides for configuring access lists for other protocols.

Standard IP Access Lists

The standard IP access list filters packets based on their *source* address. Each rule in the access list contains a bit pattern (expressed in dotted decimal notation like an IP address) that the router compares to source addresses. When a packet's source address matches the bit pattern, the router takes your prescribed action to either permit or deny the packet. Using a very flexible syntax, you can define granular rules that match a single source IP address or broad rules that match ranges of addresses (from multiple hosts, subnets, or networks, for example).

To create a standard IP access list, you must choose a number from 1–99 that will identify the access list in the router. The number range 1–99 is defined by Cisco and is reserved for standard IP access lists only. It doesn't matter which number you choose to identify your access list as long as it's in the range and unique from other access lists in the router.

Creating an Access List

To create your access list, you start with a rule. You include in this initial rule (and subsequent rules if there are any) the number of the access list. To create a standard access list that permits traffic from host 172.16.1.1, for example, configure the following:

```
MyRouter#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
MyRouter(config)#access-list 1 permit 172.16.1.1 0.0.0.0
MyRouter(config)#exit
```

where **access-list 1 permit 172.16.1.1 0.0.0.0** adds a rule to access list 1 that permits packets with source address 172.16.1.1.

0.0.0.0 is the *wildcard* (also called the *mask* or *wildcard mask*) for the pattern 172.16.1.1. The wildcard tells the router which bits of the pattern to match and which to ignore. Like IP addresses, wildcards are 32 bits written in dotted decimal notation, like this:

0.0.0.0 equals 0000-0000.0000-0000.0000-0000.0000-0000

A zero in a bit position tells the router to match the bit; that is, to compare the bit in the packet's source address to the same bit in the pattern. Conversely, a one tells the router to ignore the bit—to skip a comparison of the bit in the packet to the same bit in the pattern. The one bit is often

called the *don't care* bit because it indicates a bit position in the pattern that you want to ignore—you don't care if the bit is a one or a zero. Therefore, a wildcard 0.0.0.0 applied to the pattern 172.16.1.1 tells the router to match all 32 bits of the pattern 172.16.1.1. Because the wildcard has no don't care bits, the packet's source address must match 172.16.1.1 exactly for it to be permitted.

Using Wildcards to Match Address Ranges

Using one bits (don't care bits) in the wildcard is convenient for matching ranges of addresses. Consider the following access list, which permits addresses 192.169.1.128 through 192.169.1.159, inclusive:

```
access-list 2 permit 192.169.1.128 0.0.0.31
```

A good way to see what's going on is to convert the last octet of the pattern and wildcard to binary, line up the bits, and examine the don't care bits. You don't need to convert the other octets to binary in this example, because the first three octets of the wildcard are all zeros (to match all eight bits in each octet). Focusing on the last octet, the wildcard 0.0.0.31 tells the router to match the first three bits and ignore the last five bits.

Pattern: 192.169.1.128 equals 192.169.1.1000-0000
Wildcard: 0.0.0.31 equals 0.0.0.0001-1111

Lining up the pattern to the wildcard, the criteria to match is 192.169.1.100*x*-*xxxx*, where *x* indicates a don't care bit—the bit may be a zero or a one.

Line up the pattern with the wildcard:

```
192.169.1.100| 0-0000
    0.0.0.000| 1-1111
```

The pattern to match is 192.169.1.100*x*-*xxxx*.

If the last five bits can be any combination of zeros and ones, the rule permits addresses 192.169.1.100**0-0000**, 192.169.1.100**1-1111**, and everything between:

- Lowest address that matches (last 5 bits all zeros): 192.169.1.100**0-0000** (192.169.1.128)
- Highest address that matches (last 5 bits all ones): 192.169.1.100**1-1111** (192.169.1.159)

This bit-level analysis is how you determine that 192.169.1.128 with wildcard 0.0.0.31 matches the range 192.169.1.128 through 192.169.1.159, inclusive.

As you configure wildcards, consult Table 6-1 for assistance (or memorize it). Table 6-1 provides a handy reference for mapping wildcards that cover ranges to their binary equivalents. Notice that each wildcard contains binary zeros (match bits) on the left and binary ones (don't care bits) on the right.

Table 6-1 *Common Octets Used for Wildcards and Their Binary Equivalents*

Octet in Decimal[1]	Binary Equivalent
1	0000-0001
3	0000-0011
7	0000-0111
15	0000-1111
31	0001-1111
63	0011-1111
127	0111-1111
255	1111-1111

1. For example, the octet *x* in 0.0.0.*x* or the octet *y* in 0.0.*y*.255.

The previous example used a wildcard of 0.0.0.31. Using Table 6-1, you can verify that the binary expansion of 31 is indeed 0001-1111.

NOTE Masking addresses with access list wildcards is typically the reverse of subnet masking (see Chapter 1, "Managing Your IP Address Space"). Wildcards usually begin with a series of zeros and end with a series of ones (0.0.255.255, for example). Subnet masks begin with ones and end with zeros (255.255.0.0, for example).

Adding Rules to an Access List

If one pattern-wildcard pair is all you need to define the traffic you wish to filter, all you need is a one-rule access list like the previous example. However, you'll often need to add rules to your access list to permit or deny more types of traffic (from multiple addresses, for example).

To add more rules to an access list, you simply continue entering one-line rules with the same access list number. For instance, you could add more rules to the first example (access list 1) like this:

```
MyRouter(config)#access-list 1 deny 172.16.1.0 0.0.0.255
MyRouter(config)#access-list 1 permit 192.168.2.8 0.0.0.3
```

By issuing the **show running-config** command, you can view all three rules in the access list:

```
MyRouter(config)#exit
MyRouter#sh run
Building configuration...
Current configuration:
[partial listing, some lines not shown]

access-list 1 permit 172.16.1.1      (prior existing rule)
access-list 1 deny   172.16.1.0 0.0.0.255      (new rule)
access-list 1 permit 192.168.2.8 0.0.0.3       (new rule)
!
line con 0
line 1 8
 transport input all
line aux 0
line vty 0 4
 exec-timeout 0 0
 password cisco
 login
!
end
```

There is also a **show access-lists** enable mode command:

```
2503#sh access-l
Standard IP access list 1
    permit 172.16.1.1
    deny   172.16.1.0, wildcard bits 0.0.0.255
    permit 192.168.2.8, wildcard bits 0.0.0.3
```

Notice that the output doesn't include the wildcard 0.0.0.0 even though it was typed in during configuration. This is a shorthand notation; for access lists that match an address exactly, you do not have to type the wildcard 0.0.0.0. In fact, the three following commands are equivalent:

```
2503(config)#access-list 3 deny 192.168.1.1 0.0.0.0
2503(config)#access-list 3 deny 192.168.1.1
2503(config)#access-list 3 deny host 192.168.1.1
```

In the last line, the **host** keyword tells the router that the next parameter is an exact host address to match.

The following frequently used rule matches *any* address:

```
2503(config)#access-list 4 permit 0.0.0.0 255.255.255.255
```

The wildcard 255.255.255.255 contains all ones and means to ignore all 32 bits of the address pattern. The pattern 0.0.0.0 isn't very interesting because the pattern doesn't matter. Because the wildcard includes 32 don't care bits, the meaning of **access-list 4 permit 0.0.0.0 255.255.255.255** is "permit all packets" and all 32 bits will be ignored. This is commonly called the *permit-any* rule. The shorthand notation for 0.0.0.0 255.255.255.255 is simply the keyword **any**; thus, the following commands are equivalent:

```
2503(config)#access-list 4 permit 0.0.0.0 255.255.255.255
2503(config)#access-list 4 permit 123.123.123.123 255.255.255.255
2503(config)#access-list 4 permit any
```

NOTE The permit-any rule often follows one or more deny rules. In other words, it's typical to write an access list that denies certain addresses but permits all others. See "Important Points for Designing Access Lists" later in this chapter for more design considerations.

Applying the Access List to an Interface

An access list is just a collection of rules occupying router memory and doesn't do anything until you tell the router how to use it. For example, you can write an access list that denies all traffic from host 172.16.1.1, but until you apply the access list to an interface, the access list will just sit on the router in a *ready for use* state.

To apply an access list to an interface and put it into action, use the **ip access-group** interface command:

```
RTA#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RTA(config)#interface s0
RTA(config-if)#ip access-group 1 in
```

The command **interface s0** selects an interface for configuration and changes the prompt to interface configuration mode. The command **ip access-group 1 in** applies access list number 1 to interface Serial0 (s0). The **in** keyword tells the router to use the access list to filter inbound traffic. Alternatively, you can use the **out** keyword to filter outbound traffic—that is, traffic leaving the router through this interface.

There's no harm in writing an access list and never applying it, but it would be a waste of time because the access list would sit on the router passively without inspecting a single packet. Therefore, remember these two tasks when deploying access lists:

- Write the access list with the rules you want.
- Apply the access list to the appropriate interface to begin filtering packets with it.

Also, don't forget to specify whether the access list should inspect *inbound* or *outbound* packets when you apply it to the interface.

To verify that the access list is properly applied to an interface, issue the **show ip interface** command:

```
RTA#sh ip int e0
Ethernet0 is up, line protocol is up
  Internet address is 192.168.80.138/30
  Broadcast address is 255.255.255.255
  Address determined by non-volatile memory
  MTU is 1500 bytes
  Helper address is not set
  Directed broadcast forwarding is enabled
  Outgoing access list is not set
  Inbound  access list is 1
```

The output **Inbound access list is 1** confirms that access list number 1 is active on this interface and is filtering inbound traffic.

Important Points for Designing Access Lists

You should be aware of some important points when designing your access lists:

- Rules are processed in sequential order. That is, the router compares a packet to the access list, starting with the first rule on the list, and—if the first rule does not match the packet—progresses down the list until it finds a rule that matches the packet. The first rule that matches the packet is the one the router uses to either permit or deny the packet.
- After permitting or denying the packet, the router stops access-list processing for that packet and does not continue down the rules list. When the next packet arrives on the interface, the router restarts the filtering process, beginning with the first rule on the access list.
- Try to write your access list so that the majority of traffic is matched in the first few rules at the top of your access list. This way, most of the packets will be quickly matched and fewer packets will be compared against all of the rules in the access list. This will improve access-list processing on your router.
- As you write rules for your access list, each rule is added to the end of the access list in the order you enter it. You cannot insert rules between two rules that already exist in your access list. If you need to insert a rule, you must delete the entire access list and rewrite it in its new form; therefore, a good idea is to save your router configurations and use copy-and-paste editing techniques when your access lists are long.

  If you are using a terminal emulation program, issue the **show running-config** command and paste the old access list into a text editor. Make the changes offline in the editor. Then, in IOS, delete the old access list with the **no access-list** command and paste the modified access list from the editor to the IOS configuration prompt.

TIP To delete an access list, use the **no access-list** command:

```
RTA(config)#no access-list 1
```

- You cannot edit rules that already exist in an access list. As with inserting rules, you must delete the access list and reenter the list in its new form. Again, use copy-and-paste editing to avoid lengthy reconfiguration.
- When adding a rule that denies a particular packet, be certain that no preceding rules in the list match and undesirably permit that same packet. Remember, the *first* rule with a match is the one that is used.

- All packets will match at least one rule in the access list, because the last rule is a default, invisible, catchall rule that denies *all* packets. See "The Invisible Rule in Every Access List" later in this chapter for details.
- Access lists are unidirectional. When you apply an access list to an interface, you should specify whether the list is for matching inbound or outbound packets. If you do not specify the direction, outbound is the default. If you want to filter traffic both inbound and outbound on an interface, you must apply two access lists: one for inbound and another for outbound. You may apply the same access list for both the inbound and outbound directions.
- You may apply the same access list to multiple interfaces.
- Access list numbers are unique only within a router. An access list on one router is meaningless to another router.
- Try not to make your access lists too long. If they contain many rules, performance of the router might be slowed because the router must compare each packet to each rule in the access list (this is a minimal concern on routers that use hardware-based access lists). How long is too long depends on the horsepower of your router, the amount of throughput you need to sustain on a link, and the acceptable trade-off you can handle between security and performance.

NOTE Enabling services such as Cisco's NetFlow™ switching can minimize the performance impact of long access lists. Consult the IOS Configuration Guides for NetFlow switching information.

- Try to write access lists with as few rules as possible while still meeting your filtering requirement. You might be able to consolidate multiple rules into one by leveraging the don't care bits to match ranges of addresses.
- Standard IP access lists can only match on bit patterns in the source address of packets. To match on the destination address and other fields of a packet, see the "Extended IP Access Lists" section later in this chapter.

The Invisible Rule in Every Access List

You need to be keenly aware that IOS automatically adds a default rule at the end of every access list. This terminating rule is invisible: It does not appear in outputs of **show running-config** or **show access-lists**, but it's there and you cannot disable it. The invisible rule is a *deny-any* rule that denies all packets that haven't matched the user-defined rules in your access list. For example, consider that you have configured an access list like this:

```
2509(config)#access-list 1 permit 172.16.1.1 0.0.0.0
2509(config)#access-list 1 deny 172.16.1.0 0.0.0.255
2509(config)#access-list 1 permit 192.168.2.8 0.0.0.3
```

The access list looks like this logically:

```
access-list 1 permit 172.16.1.1 0.0.0.0
access-list 1 deny 172.16.1.0 0.0.0.255
access-list 1 permit 192.168.2.8 0.0.0.3
(access-list 1 deny any)
```

where **(access-list 1 deny any)** is the invisible terminating rule at the end of this and every other access list, including extended access lists (covered later in "Extended IP Access Lists").

The invisible rule is a safety measure to prevent accidental leakage through the access list in case you make a configuration mistake—that is, in case you inadvertently missed some packets in your rules that should be denied by the access list. Because you most likely configured the access list for security, Cisco decided that denying packets that don't match your rules is the way to be safe. In a sense, better safe than sorry. If you find that packets are undesirably blocked by the invisible deny-any rule, you can configure a permit rule that matches the packets that should be allowed through.

TIP When you're configuring access lists, don't forget that routing protocol packets (such as OSPF hellos) and other control packets are circulating on the network and might be blocked by your access list. If an access list undesirably blocks these packets because of the invisible deny-any rule, go back and add rules to your access list that permit them. You might have to use extended access lists for more granular matching rules (see the following section, "Extended IP Access Lists").

Extended IP Access Lists

Extended access lists enable you to build more elaborate rules for filtering than do standard access lists. Although standard access lists only match on source address, extended access lists can match on

- Source address
- Destination address
- Protocol (ICMP, TCP, UDP, EIGRP, IGRP, OSPF, and others)
- Protocol-specific options (Telnet, FTP, HTTP, SMTP, Echo, SNMP, and others)
- Precedence level
- Type of service (TOS)

Extended Access List Syntax

The syntax for extended access lists is

```
access-list access-list-number {deny|permit} protocol source source-wildcard
destination destination-wildcard [precedence precedence] [tos tos]
```

That rule looks like quite a lot because you have more options, but it's really not much more than a standard access list. Consider the following example:

```
access-list 100 permit ip 172.16.1.0 0.0.0.255 192.168.25.0 0.0.0.31
```

where 100 is the access list number that for extended access lists must lie in the range 100–199, inclusive. Any number in the range will work as long as it does not conflict with other extended access lists in the router. 172.16.1.0 0.0.0.255 is the source-matching criteria and has the same meaning and syntax as standard access lists. To quickly review:

- **172.16.1.0 0.0.0.255** means the router will compare the first three octets of the pattern 172.16.1.0 to the source address of the packet—the last octet contains don't care bits, as indicated by the 0.0.0.255 wildcard.
- **192.168.25.0 0.0.0.31** is the destination matching criteria and also follows the same pattern-wildcard syntax.

For a packet to match this rule and be permitted, it must match both the source and destination criteria.

TIP You must specify both source and destination matching criteria in rules for extended access lists.

As with standard access lists, the shorthand notations for **any** and **host** matching are allowed. Here's an example:

```
access-list 101 permit ip any host 192.168.25.10
```

Here, the source criteria is **any** (from any source) and the destination criteria is **host 192.168.25.10** (an exact match of the address 192.168.25.10). In more human terms, the previous rule might read:

> Permit packets from any source to host 192.168.25.10.

To filter packets based on protocol information, you build on the source or destination criteria. To block Telnet sessions from one range of addresses to another, for example, you might configure an access list similar to this one:

```
access-list 102 deny tcp 192.168.25.0 0.0.0.31 172.16.1.0 0.0.0.255 eq telnet
access-list 102 permit ip any any
```

where the keyword **tcp** in the first rule tells the router that the rule filters TCP packets only.

TIP

To get a listing of other protocol options, use the question mark key and context-sensitive help:

```
MyRouter(config)#access-list 102 deny ?
  <0-255>  An IP protocol number
  eigrp    Cisco's EIGRP routing protocol
  gre      Cisco's GRE tunneling
  icmp     Internet Control Message Protocol
  igmp     Internet Gateway Message Protocol
  igrp     Cisco's IGRP routing protocol
  ip       Any Internet Protocol
  ipinip   IP in IP tunneling
  nos      KA9Q NOS compatible IP over IP tunneling
  ospf     OSPF routing protocol
  tcp      Transmission Control Protocol
  udp      User Datagram Protocol
```

The source criteria is **192.168.25.0 0.0.0.31** and is a straightforward match on an address pattern. The destination criteria is **172.16.1.0 0.0.0.255 eq telnet**, where the keywords **eq telnet** impose an additional requirement for a destination match: The destination port number must be equal to Telnet (port 23). The keyword **eq** is a logical operand and is short for "equal to." The operands you can use for matching port numbers are

- **eq**—equal to
- **neq**—not equal to
- **gt**—greater than
- **lt**—less than
- **range**—to specify the starting and ending port numbers in a range

To get a listing of well-known port numbers and their names, use the question mark key and context-sensitive help built into IOS. Here's a listing for TCP:

```
RTA(config)#access-list 101 deny tcp 192.168.25.0 0.0.0.31 172.16.1.0 0.0.0.255 eq ?
  <0-65535>  Port number
  bgp        Border Gateway Protocol (179)
  chargen    Character generator (19)
  cmd        Remote commands (rcmd, 514)
  daytime    Daytime (13)
  discard    Discard (9)
  domain     Domain Name Service (53)
  echo       Echo (7)
  exec       Exec (rsh, 512)
  finger     Finger (79)
  ftp        File Transfer Protocol (21)
  ftp-data   FTP data connections (used infrequently, 20)
  gopher     Gopher (70)
  hostname   NIC hostname server (101)
  ident      Ident Protocol (113)
```

```
irc          Internet Relay Chat (194)
klogin       Kerberos login (543)
kshell       Kerberos shell (544)
login        Login (rlogin, 513)
lpd          Printer service (515)
nntp         Network News Transport Protocol (119)
pop2         Post Office Protocol v2 (109)
pop3         Post Office Protocol v3 (110)
smtp         Simple Mail Transport Protocol (25)
sunrpc       Sun Remote Procedure Call (111)
syslog       Syslog (514)
tacacs       TAC Access Control System (49)
talk         Talk (517)
telnet       Telnet (23)
time         Time (37)
uucp         Unix-to-Unix Copy Program (540)
whois        Nicname (43)
www          World Wide Web (HTTP, 80)
```

TIP You can type port numbers instead of the keywords **telnet**, **ftp**, **www**, and so on.

Look again at the previous two-line example (access list 102):

```
access-list 102 deny tcp 192.168.25.0 0.0.0.31 172.16.1.0 0.0.0.255 eq telnet
access-list 102 permit ip any any
```

The rule **access-list 102 permit ip any any** at the end of the list permits all other IP traffic—that is, traffic not matching the first rule. This *permit-any-any* rule is necessary because the invisible terminating rule would deny all other traffic, which is undesirable in this example. Refer to "The Invisible Rule in Every Access List," earlier in this chapter.

Examining the list whole lines at a time, the first rule, **access-list 102 deny tcp 192.168.25.0 0.0.0.31 172.16.1.0 0.0.0.255 eq telnet**, translates into human terms like this:

> Deny TCP packets *from* source 192.168.25.0 (match first 27 bits) with any source port number *to* destination 172.16.1.0 (match first 24 bits) with destination port number equal to Telnet (port 23).

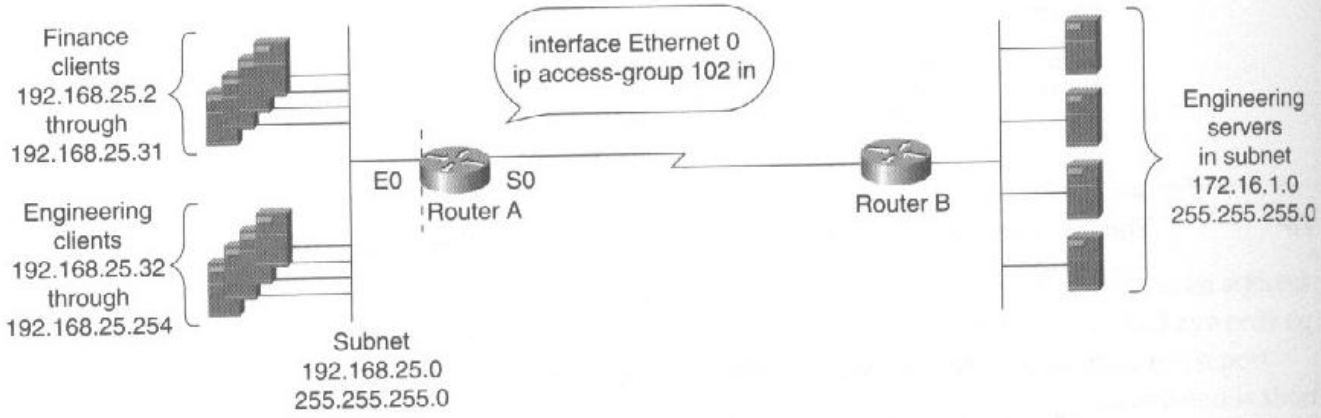
NOTE The wildcard 0.0.0.31 is 0000-0000.0000-0000.0000-0000.0001-1111 in binary (see Table 6-1).

And the last rule, **access-list 102 permit ip any any**, translates to

Permit IP packets from any source to any destination.

Applying this access list 102 as an inbound list on Router A's Ethernet0 (see Figure 6-4) prevents Telnet sessions initiated from the finance clients to the engineering servers. However, engineering clients can Telnet to the engineering servers successfully.

Figure 6-4 *Using the Example Access List 102 to Deny Telnet Sessions*



Applying the Extended Access List to an Interface

As with standard access lists, extended access lists do not filter traffic until you apply them to an interface with the **ip access-group** command. The following example applies access list 102 to Ethernet0 and specifies the inbound direction:

```
RTA(config)#int e0
RTA(config-if)#ip access-group 102 in
```

Logging Access List Activity

Sometimes it's useful to log the activity of traffic that is denied by an extended access list. This is especially true when you are using an access list for security and intentionally blocking conversations that shouldn't be happening. To log when packets match an access list rule, simply append the **log** keyword to the rule (logging is not available on standard access lists). For example, to log every time a packet is denied from sources matching 172.16.1.0 0.0.0.255, the access list might look like this:

```
access-list 103 deny ip 172.16.1.0 0.0.0.255 any log
<other access list rules>
access-list 103 permit ip any any
```

To view the log, issue the **show logging** command or check your syslog server if the router is sending output to a syslog server (see Appendix E, "A Crash Course in Cisco IOS," for syslog configuration):

```
2503#sh log
Syslog logging: enabled (0 messages dropped, 0 flushes, 0 overruns)
    Console logging: level debugging, 49 messages logged
    Monitor logging: level debugging, 0 messages logged
    Trap logging: level informational, 53 message lines logged
    Buffer logging: level debugging, 49 messages logged
         Log Buffer (4096 bytes):

%LINK-3-UPDOWN: Interface BRI0, changed state to up
%LINK-3-UPDOWN: Interface Ethernet0, changed state to up
<lines deleted>
*Apr 14 07:11:50: %SEC-6-IPACCESSLOGDP: list 103 denied icmp 172.16.1.1 -> 1
92.168.1.1 (0/0), 1 packet
*Apr 14 07:11:54: %SEC-6-IPACCESSLOGP: list 103 denied udp 172.16.1.1(0) ->
172.16.10.1(0), 23 packets
*Apr 14 07:12:23: %SEC-6-IPACCESSLOGP: list 103 denied tcp 172.16.1.1(0) ->
192.168.1.2(0), 1 packet
```

TIP If you aren't getting timestamps in your log messages, configure the router with the **service timestamps log datetime** command:

```
RTA(config)#service timestamps log datetime
```

Monitoring Extended Access List Counters

Extended access lists maintain packet counters for each rule. These are useful when you're troubleshooting and verifying the operation of your access lists. You can view these statistics by issuing the **show access-lists** command:

```
RTA#sh access-l
Extended IP access list 103
    deny ip 172.16.1.0 0.0.0.255 any log (16 matches)
    permit ip any any (7703 matches)
```

TIP Reset the counters with the **clear counters** enable mode command:

```
RTA#cle count
```

Access Lists for Combating Spoofing Attacks

IP address *spoofing* is an attack in which the hacker pretends to be a trusted computer by using an address within your range of acceptable internal addresses. Although spoofing is only one of many attacks practiced in the hacker world, it is one of the most popular and therefore should be on your list of attacks to thwart.

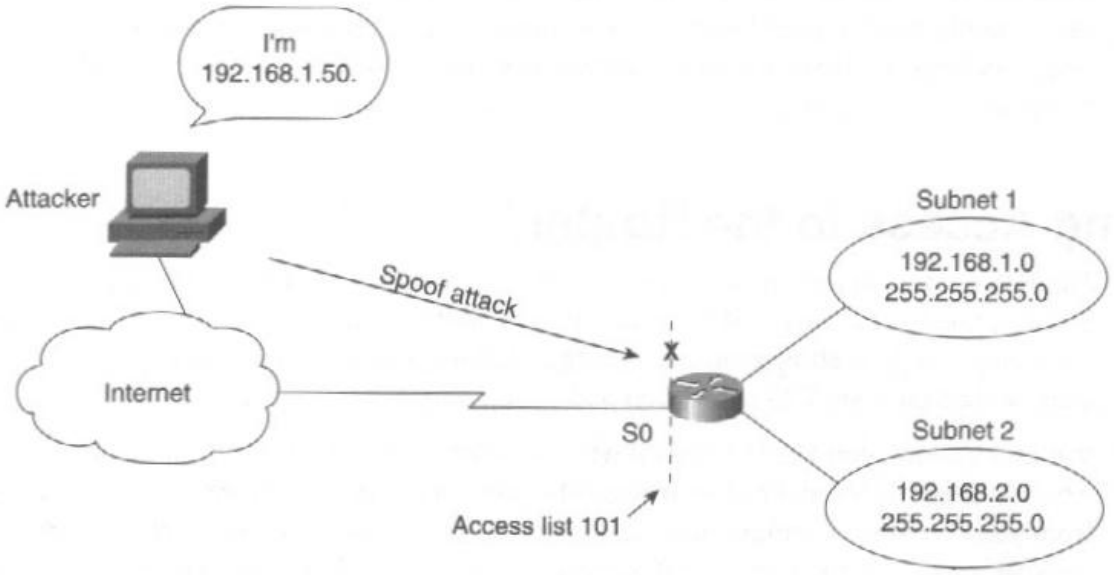
This section describes some basic anti-spoofing filters that are useful for almost any router connected to an untrusted network—for example, the Internet. The access list techniques in this section are generally considered good practices and a starting point for more sophisticated defenses.

An anti-spoofing access list is a defense against one style of attack, so it obviously cannot defend against all attacks and therefore should be used in conjunction with a dedicated firewall such as Cisco's PIX Firewall or a router running Cisco's IOS Firewall Feature Set (covered in Chapter 8). Dedicated firewalls also improve on the anti-spoof prevention performed by regular access lists. For more types of attacks and defenses, see the rest of this chapter, Chapter 7, and Chapter 8.

NOTE Few will claim there's a 100 percent hacker-proof network out there on the Internet—to do so would probably attract the best hackers to their door. Even with an infinite budget and the best security technology available, organizations cannot ignore the fact that new vulnerabilities are found and more attacks invented regularly. Luckily, remedies are usually quick and published by vendors such as Cisco; still, preventing attacks is key to maintaining confidence in the network and its effectiveness for the users. The IOS-based techniques in this chapter, Chapter 7, and Chapter 8 will get you started and build on your current security practices. You can find some security references in the Bibliography.

The router in Figure 6-5 is configured with an anti-spoofing access list on interface Serial0, pointing to the public Internet.

Figure 6-5 *Deploying Anti-spoofing Access Lists to Increase Security*



The access list configuration for the router in Figure 6-5 is as follows:

```
interface Serial0
ip address 172.16.1.1 255.255.255.252
ip access-group 101 in
!
access-list 101 deny   ip 192.168.1.0 0.0.0.255 any
access-list 101 deny   ip 192.168.2.0 0.0.0.255 any
access-list 101 deny   ip 127.0.0.0 0.255.255.255 any
access-list 101 permit ip any any
!
<lines deleted for brevity>
```

Access list 101 filters inbound traffic from the Internet on Serial0. The rule **access-list 101 deny ip 192.168.1.0 0.0.0.255 any** denies traffic from the Internet that has a source address belonging to the internal network, subnet 1 (with a subnet mask of 255.255.255.0). The filter is there because you wouldn't expect any inside address to *arrive* on the outside interface (Serial0) that points to the Internet. Any packets that come from the Internet with a source address from the internal network are highly suspicious of a spoofing attacker—that is, someone who looks like an internal computer but is coming from the Internet. If the attacker succeeds in masquerading as one of the internal clients, internal servers might think communicating with him is safe—a classic spoof job.

The rule **access-list 101 deny ip 192.168.2.0 0.0.0.255 any** is similar to the first rule. It ensures no packets are permitted from the Internet with a source address that belongs to subnet 2. This example has just two internal class C address ranges. With a larger internal network, you need to configure enough rules in your anti-spoofing access list to cover all of your internal addresses.

The rule **access-list 101 deny ip 127.0.0.0 0.255.255.255 any** denies packets that start with octet 127: IP loopback addresses for hosts. These packets should never appear on a physical link and certainly not from the Internet. Some spoof attacks use these reserved addresses as the source address. In these attacks, the hacker does not need to know what your addressing is on the inside.

Securing Access to the Router

The IOS command prompt is the primary place where you monitor and configure the router. By now, you have configured the router in IOS configuration mode and used some **show** commands, such as **show running-config** and **show access-lists**, to validate your work. For a quick start on basic IOS navigation and configuration, see Appendix E.

Without a doubt, you want to control who can access the command prompt, not only to control who can alter the behavior of your router but also to protect the information the router collects from your network. Configurations and statistics gleaned from a router tell a lot about your network's policies, topologies, traffic patterns, and attached systems. Therefore, controlling who can access a router is more than just protecting one device, it's protecting an intertwined operation of computer systems and policies.

These are the two main ways to access a router and get to the IOS prompt (also called the EXEC prompt):

- Connecting to the console port
- Initiating a Telnet session

NOTE The most common way to connect to the console port is with an EIA/TIA-232 (RS-232) serial cable and client software with terminal emulation, but sometimes remotely dialing a modem connected to an AUX or other asynchronous serial port on the router is convenient. Detailed information is currently located at http://www.cisco.com/warp/public/701/6.html, or you can search for "modem, aux port" on Cisco's Web site (http://www.cisco.com). See also Appendix E.

After connecting to the router, you are placed into user mode and allowed some basic commands such as **ping**, **telnet**, **traceroute**, and so on. However, more useful administrator and configuration commands require enable mode.

This section covers

- Securing the Enable Mode of a Router
- Securing Telnet Access
- Securing Access to the Console Port

Securing the Enable Mode of a Router

The first thing you should do when configuring a router is set a password that protects the *enable mode* (the administrator level) of the router. You can think of the enable mode, also called the *privileged EXEC mode*, as the superuser level that is allowed to monitor and modify everything in the router. For basic information on navigating to and from enable mode, see Appendix E.

By default, no password is assigned to the enable mode, so you can get to it with a connection to the console port and the **enable** command:

```
RTA>enable
RTA#
```

You are not allowed into enable mode, however, when there is no enable password and you are connected to the router with Telnet. The following output is an attempt to initiate enable mode from a Telnet session on a router without an enable password. Notice that the message **No password set** is displayed and the prompt returns to user EXEC mode as indicated by the **>** character:

```
MyRouter>enable
% No password set
MyRouter>
```

To set the enable password, use the **enable secret** global command:

```
RTA#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RTA(config)#enable secret foo!enable
```

where **enable secret foo!enable** sets the enable password in this example to foo!enable (the exclamation point was used to make the password harder to guess). Attempts to change from user mode to enable mode now require the enable password:

```
RTA>en
Password: <Type the password here. Text is not displayed.>
RTA#
```

TIP The **enable secret** command uses a one-way cryptographic hashing function to store the password securely. Another command, **enable password**, also sets the enable password but is not recommended because it is less secure.

Cisco IOS supports a total of 16 configurable modes called *privilege levels*. The idea is that you can configure privilege levels with different allowable commands and with different passwords. Then, based on the passwords people type, they are put into the corresponding level with IOS commands you allow for that level. For more information on privilege levels, consult the *Security Configuration Guide* IOS manual. By default, a router has two modes: the user EXEC mode (upon login) and the enable mode.

Securing Telnet Access

By default, Cisco routers support five simultaneous Telnet sessions, allowing up to five people to log into the router at the same time. The router treats these sessions as logical interfaces called *virtual terminal (or vty) lines*.

The router doesn't have any passwords configured on its vty ports by default. Trying to Telnet to the router without vty passwords will be unsuccessful. The router will respond with a message **Password required, but none set** and immediately terminate the attempt:

```
myserver#telnet 192.168.1.2
Trying...
Connected to 192.168.1.2.
Escape character is '^]'.

Password required, but none set

Connection closed by remote host.
```

To enable Telnet sessions to the router, you must at a minimum configure a password on the vty lines—or you can configure vty lines with the **no login** command and disable password checking entirely (not recommended). The following is an example:

```
RTA#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RTA(config)#service password-encryption
RTA(config)#line vty 0 4
RTA(config-line)#login
RTA(config-line)#password foo!pass
```

In this configuration:

- The command **service password-encryption** enables a feature that encrypts passwords in the router so you cannot see them in plaintext when viewing the configuration with the **show running-config** command.
- The command **line vty 0 4** tells the router that you want to simultaneously configure all five of the vty lines numbered 0-4. This changes the prompt to line configuration mode, as indicated by the text **config-line**.
- The command **login** enables password checking for vty (Telnet) connections. This should already be enabled by default, but it doesn't hurt to enter the command and ensure password checking is on.
- The last command, **password foo!pass**, sets the password for the vty lines to foo!pass.

NOTE Cisco IOS passwords are case sensitive.

With passwords on the vty lines, you can now Telnet to the router and use the password:

```
myserver#telnet 192.168.1.2
Trying...
Connected to 192.168.1.2.
Escape character is '^]'.

User Access Verification

Password: <Type the password here. Text is not displayed.>
RTA>
```

TIP If you worry that five active Telnet sessions by other people will prevent you from accessing the router, you can create more vty lines, or you can configure a common password for vty lines 0 through 3 and a different password for vty 4 for "emergency purposes only." Use the **line vty 4** command to configure line 4 only. To create more vty lines, simply issue the **line vty** command with numbers greater than 4. The command **line vty 5 9**, for example, creates five more vty lines, numbered 5 through 9. Telnet sessions consume vty lines in the order they are numbered, starting with 0.

Additional login options allow the router to check usernames and leverage RADIUS and TACACS+ servers. See "Deploying Authentication, Authorization, and Accounting (AAA)," later in this chapter.

Controlling vty Access with Access Lists

If you want to restrict who can Telnet to a router, apply access lists to the logical vty lines that permit only authorized addresses. Here's a partial configuration listing:

```
access-list 10 permit 192.168.1.0 0.0.0.255
!
line vty 0 4
 access-class 10 in
```

The command **access-class 10 in** applies access list number 10 to all five vty lines (vty lines 0–4). Only users matching the source criteria **192.168.1.0 0.0.0.255** are allowed to Telnet to the router. See "Controlling Traffic with Access Control Lists" earlier in this chapter for more information on access lists.

Securing Access to the Console Port

If you have a direct connection to the console port, you can break into a router by using password recovery techniques. This is to recover from forgotten passwords (see Appendix D, "Password Recovery"). Still, configuring the console port with a password to discourage the casual hacker is a good idea.

To set the console port password, configure the following and replace **foo!console** with your password:

```
RTA#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RTA(config)#line console 0
RTA(config-line)#password foo!console
```

NOTE Password recovery requires a direct physical connection to the console port (see Appendix D). A hacker who has physical access to your router can do worse things than circumventing the password, so lock the router in a secure place.

Deploying Authentication, Authorization, and Accounting

Providing service to road warriors, telecommuters, and other remote users greatly extends the effective reach of the network. With the option to make the network available anytime from anywhere, organizations see opportunities for convenience, productivity, and cost savings.

Critical to these gains is ensuring that the network can extend to remote areas with sufficient security. If the network is available anytime from anywhere—perhaps just a phone call away—you must make sure it's still not accessible to *anyone*.

Some popular methods to reach a private network from the outside world include dialup modems over basic telephone service, telecommuter ISDN routers, cable modems or xDSL devices over the public Internet, wireless modems, and so on. These types of access methods can liberate new applications and new ways of doing business, but can also bring new challenges in securing a private service that is publicly accessible.

This section covers

- Authentication, Authorization, and Accounting
- Configuring Authentication for Network Access over PPP
- Using the Default Authentication List
- Configuring Authentication for Router Logins
- The Local Username Database
- Configuring Authorization
- Configuring Accounting
- Pointing the Router to the RADIUS or TACACS+ AAA Server

Authentication, Authorization, and Accounting

The networking industry has adopted three principles that address the basic needs of remotely accessible networks. They are *authentication, authorization, and accounting (AAA or "triple A").*

Authentication validates a user's identity. This is the heart of remote access security and grants a user access to the network based on an assurance that the system knows the person. Authentication is typically done with user login names and passwords.

To improve the probability that a password is known only by the owner, *one-time password (OTP)* systems are often used. OTP systems usually consist of a server that authenticates passwords generated by electronic *token cards* the size of a credit card. Users type their personal identification number (PIN) into their token cards, and the token card dynamically generates and displays a password that can be used for one login only. At the time of this writing, the CiscoSecure™ AAA product for UNIX includes an OTP system.

Authorization limits what a user is allowed to do on the network, usually defined by a profile for the user or a group to which the user belongs. Some examples of what you can authorize
to a user are permitted hours of the day (or days of the week) to access the network, protocols (IP, IPX) allowed, areas of the network to access, maximum number of concurrent sessions to the network, services (Telnet, FTP, SNMP) allowed, and date of account expiration.

Accounting tracks what the user is doing or has done on the network. This usually involves recording such information as start and stop time of the user's session, duration of a session, number of bytes or packets transmitted, commands the user issued, caller line identification (CLID), and others. These records are important for usage billing and auditing.

NOTE Security software on servers and clients works in conjunction with IOS to deliver the AAA capabilities discussed here. One such product is the CiscoSecure AAA server offered by Cisco.

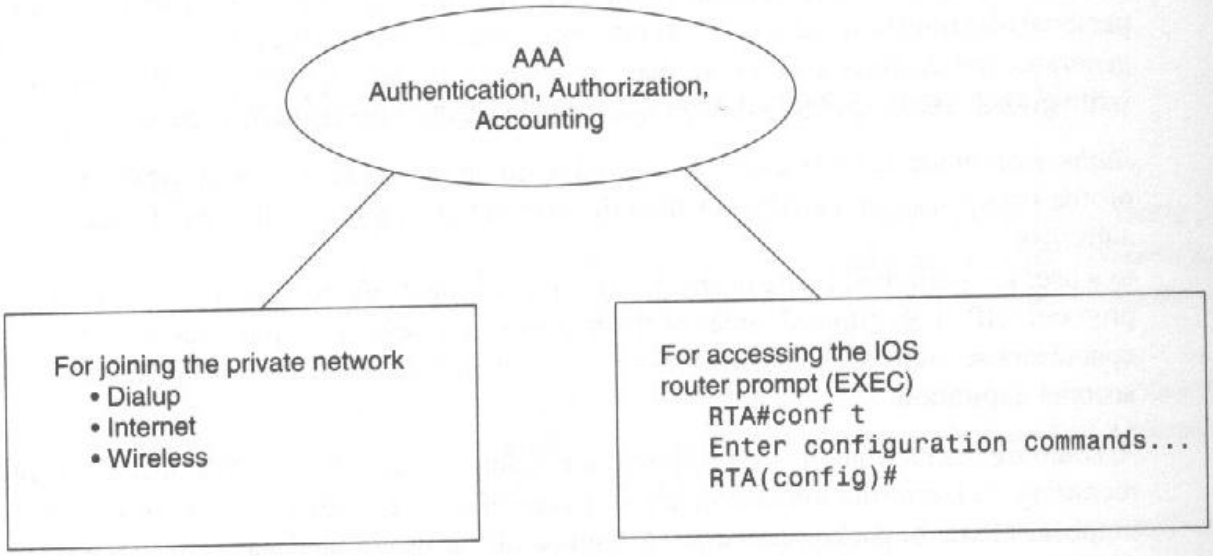
AAA is just one building block in the security infrastructure that makes your internal network available over the public switched telephone network (PSTN), Internet, or other widely accessed network. AAA is necessary but might not be sufficient. Your security policy might require more layers of network security, such as encryption, per-packet authentication, or nonrepudiation. Advanced security services, covered in Chapter 7, address such needs.

In a Cisco IOS network, the following popular services are supported by AAA:

- Connectivity for remote users who want to join the private network with functionality similar to what they have "in the office." As an example, this could be a road warrior user who dials the private network with a modem to run e-mail and Web applications.
- Logins for networking service personnel (and other people) who need to access the IOS command prompt (EXEC prompt) and monitor or configure Cisco devices.

Figure 6-6 illustrates these AAA services.

Figure 6-6 *IOS-based AAA Supports Remote Access Connectivity and Router Logins*

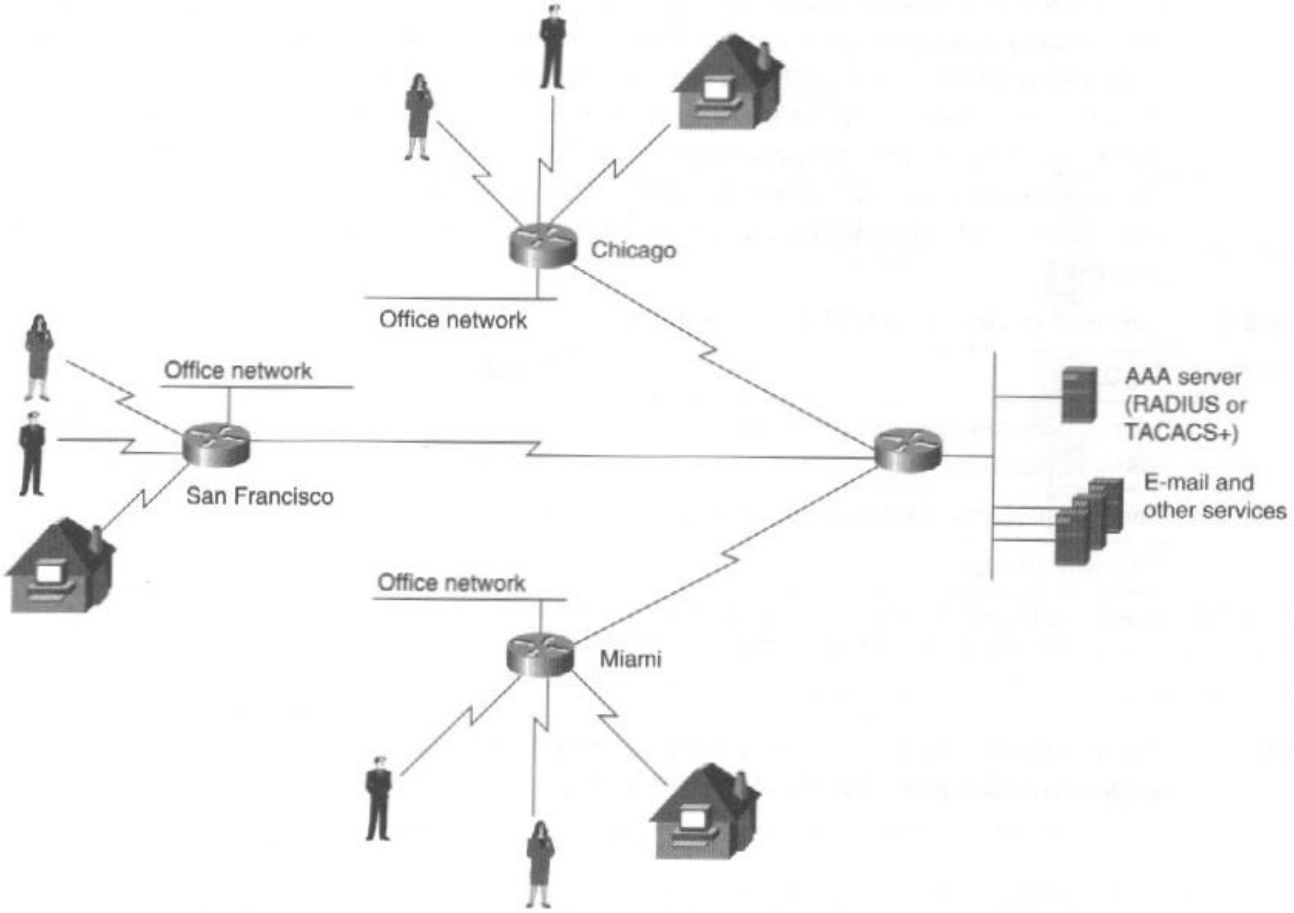


Centralized AAA Servers

A key AAA strength is a centralized security database residing on a AAA server that stores the usernames, passwords, profiles, policies, and accounting information for the organization. Centralizing the information eases administration for networks that have multiple points of access. For example, consider a network like Figure 6-7 that has multiple access points around the country.

With a centralized AAA server, each of the access routers—for example, in Chicago, Miami, and San Francisco—can consult the same database when authenticating passwords and authorizing users. This has obvious advantages in consistency and manageability because only one database needs to be maintained. Likewise, having one repository for accounting data simplifies reporting and billing.

Figure 6-7 *A Centralized AAA Server Supporting Access Routers at Multiple Branch Offices*



The alternative to the centralized AAA server model is a distributed model with separate, distributed databases on each router. Although feasible on a small network, having multiple databases on every router is cumbersome to build and maintain. Also, a distributed database approach does not deliver the variety of authorization and accounting features found in AAA servers.

Security Protocols—RADIUS and TACACS+

The AAA server and the routers converse over the network with a protocol specially designed for exchanging security information. The two most prevalent security protocols are *Remote Access Dial-In User Service (RADIUS)* and *Terminal Access Controller Access Control System (TACACS+—pronounced "tack-acks plus")*. There are technical differences in how the two protocols work, but their purpose is the same: They pass queries and responses between clients (the routers) and the AAA server for authentication, authorization, and accounting. Table 6-2 lists some of the main differences between RADIUS and TACACS+. Cisco IOS supports both protocols.

Table 6-2 *Distinctions Between RADIUS and TACACS+*

RADIUS	TACACS+
UDP: Connectionless transport without acknowledgements	TCP: Connection-oriented, reliable full-duplex transport
Encrypts password portion of packet only	Encrypts the entire packet
Primarily IP only	Multiprotocol
Standards-based	Proprietary

NOTE Some vendors add their own extensions to RADIUS. This effectively makes the RADIUS implementation proprietary.

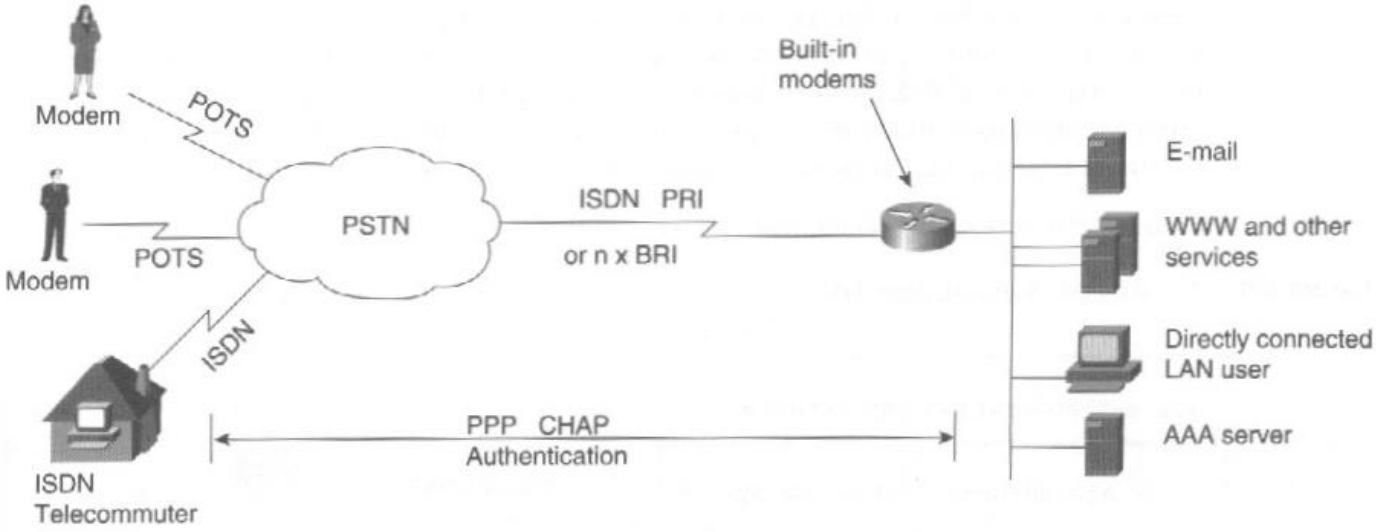
Configuring Authentication for Network Access over PPP

The *Point-to-Point Protocol (PPP)* is widely used to connect remote users and branch offices over dialup links. PPP is an OSI Layer 2 (link layer) encapsulation protocol that, in addition to IP, supports other protocols such as IPX and DECnet.

Built within PPP are two authentication methods called *Password Authentication Protocol (PAP)* and *Challenge Handshake Authentication Protocol (CHAP)*. CHAP sends a cryptographic hash of the password over the network and is therefore more secure than PAP, which sends passwords in cleartext.

Figure 6-8 illustrates a simple scenario with dialup road warriors and telecommuters connecting to an office network for their e-mail, Web, and other applications.

Figure 6-8 *Remote Users Connecting to an Office LAN over a PSTN*



The idea is that the remote users join the network and appear as normal IP nodes with the same functionality as a user directly connected to the office LAN. Of course, performance over their remote link (modem, ISDN, and the like) will most likely lag behind the throughput of a direct LAN connection.

The steps a remote user goes through to connect to the network are the following:

1. User dials the office router over the PSTN.
2. Office router answers the call from the user. This establishes a physical (Layer 1) connection between the endpoints.
3. The two endpoints establish a data link (Layer 2) connection, using the PPP Link Control Protocol (LCP). LCP establishes, configures, and tests the link.
4. After link establishment, the router sends a packet called a *challenge* to the user.
5. PPP in the user's device formulates a packet called a *response* and sends it to the router. The response consists of a cryptographic hash value of the password and the user's login name.
6. The router checks the response against its own calculation of the hash value. If the response matches the router's calculation, the user is authenticated.
7. The two endpoints proceed to the Network Control Protocol (NCP) phase of PPP, where the network layer protocol (IP) is established and configured.
8. After NCP, the user is connected to the network at the IP layer (Layer 3).

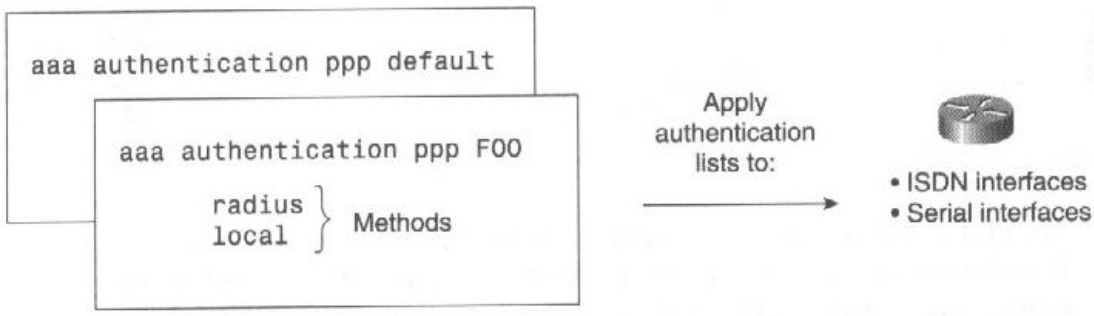
Consult RFC 1994 for more information on CHAP and RFC 1661 for more information on PPP.

Authenticating Inbound Callers with Authentication Lists

To authenticate inbound callers on a router, you first define an *authentication list* that defines a sequence of *methods* to tell the router how it should attempt authentication of the user's password. For example, an authentication list might tell the router to try the RADIUS protocol first, and if the RADIUS protocol fails (because the AAA server is offline, for example), to use the router's local username database. After creating an authentication list, you apply it to the router interface that accepts the inbound caller.

Figure 6-9 describes the relationship between authentication lists and router interfaces.

Figure 6-9 *Two Example Authentication Lists*



FOO is a user-defined authentication list with two methods: **radius** and **local**. The **default** authentication list is described in "Using the Default Authentication List," later in this chapter.

TIP When a router is conversing with the AAA server, it is a RADIUS or TACACS+ client. When you configure RADIUS or TACACS+, you enable an IOS software process in the router that handles the client-side tasks of the RADIUS or TACACS+ protocol.

Configuring Authentication Lists on Routers

Here's an example that configures an authentication list on a router:

```
RTA#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RTA(config)#aaa new-model
RTA(config)#aaa authentication ppp FOO tacacs+ local
```

where **aaa new-model** enables AAA configuration on the router. This is a required step that disables some legacy login commands on the router and enables AAA commands.

The command **aaa authentication ppp FOO tacacs+ local** creates an authentication list called FOO. This particular authentication list specifies that TACACS+ is the first authentication method. If TACACS+ authentication fails for any reason, the router will attempt to authenticate the user with its local username database as indicated by the keyword **local** following **tacacs+**. See "The Local Username Database," later in this chapter.

After creating the authentication list, you apply it to interfaces that receive the inbound callers.

For asynchronous serial interfaces (or *async* for short):

```
RTA(config)#int async 1
RTA(config-if)#ppp authentication chap FOO
```

For ISDN BRI interfaces:

```
RTA(config)#int bri 0
RTA(config-if)#ppp authentication chap FOO
```

For ISDN PRI interfaces:

```
RTA(config)#int serial 0:23
RTA(config-if)#ppp authentication chap FOO
```

NOTE Depending on the type of Cisco router, the numbering syntax for interfaces can vary slightly. Consult the documentation for your particular router for the exact numbering convention.

Additionally, your configuration might define logical interfaces that group multiple async interfaces. In the following example, async interfaces 1 through 24 are grouped as one logical interface **group-async1**, and configured as one entity for convenience:

```
RTA(config)#int group-async1
RTA(config-if)#group-range 1 24
RTA(config-if)#ppp authentication chap FOO
```

The **group-async** interface is commonly used on Cisco access servers. These routers aggregate high volumes of remote access users.

Using the Default Authentication List

Instead of creating named authentication lists (like FOO), you can define a default authentication list. The following configures the default authentication list with two methods (**tacacs+** and **local**) and enables CHAP on an interface:

```
RTA#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RTA(config)#aaa new-model
RTA(config)#aaa authentication ppp default tacacs+ local
RTA(config)#int async 1
RTA(config-if)#ppp authentication chap
```

Cisco routers use the default authentication list when CHAP is configured on an interface and no authentication list is defined.

TIP With named authentication lists, you can configure multiple lists and assign them to different interfaces.

Configuring Authentication for Router Logins

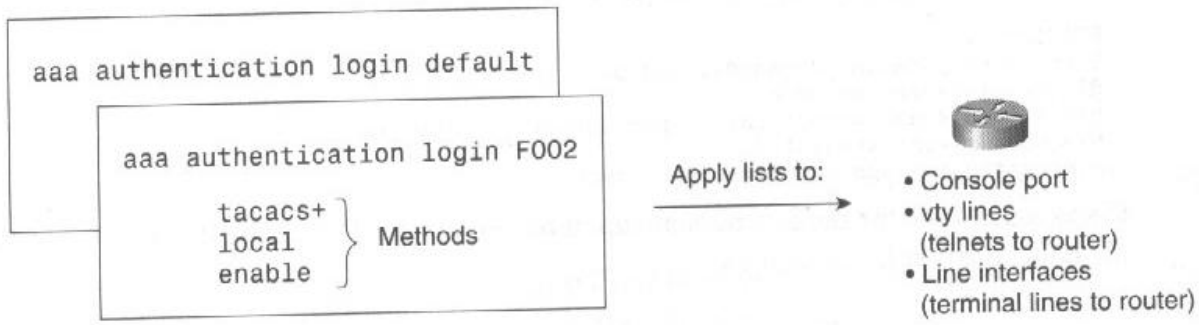
AAA is not just for remote users who want to connect to your network. Network administrators, engineers, and managers can leverage the benefits of AAA in managing a router network.

A common problem with managing a network with a team of people is tracking who makes configuration changes and when those changes happened. To solve this, you can use a common AAA database to authenticate each person, authorize IOS commands, and log changes. Even if the logging task is unimportant in your particular environment, allowing users to use their own passwords instead of the enable mode password is a good policy for secure router administration. AAA makes these capabilities possible. (Enforcing that people change those passwords on a periodic basis might also be a good idea; AAA enables this too.)

TIP AAA for IOS logins is also useful when using routers as terminal servers. Users connect to a router acting as a terminal server and, when authenticated, use the IOS prompt as a launch point to view console (tty) ports of other devices such as servers, LAN switches, modems, or other routers. See Appendix E for details on configuring a router as a terminal server (also called a *communications server*).

You configure login authentication similarly to PPP authentication by defining an authentication list and applying the list to the router's console port, vty lines, and line interfaces. Figure 6-10 illustrates the relationship between login authentication lists and router logins.

Figure 6-10 *Two Example Login Authentication Lists*



The following example configures a login authentication list and applies it to vty lines 0–4:

```
RTA#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RTA(config)#aaa new-model
RTA(config)#aaa authentication login FOO2 radius line enable
RTA(config)#line vty 0 4
RTA(config-line)#login authentication FOO2
```

where **aaa new-model** enables AAA on the router.

The command **aaa authentication login FOO2 radius line enable** creates an authentication list for logins called FOO2. This list tells the router to try RADIUS authentication first, and if that method fails for any reason, to use the password assigned to the vty line second. If no password is assigned to the vty line, the router will use the enable password as a last resort.

TIP Configuring **enable** as a backup method in a login authentication list is highly recommended. This prevents you from being locked out of the router if your AAA server is down or unreachable for some other reason.

The Local Username Database

You have the option to configure a local security database in each router, containing usernames and passwords. This is useful when you have a small set of users and a few routers that don't warrant the convenience of a AAA server. The local database can also act as a backup for usernames and passwords if the AAA server is unavailable or unreachable. To use the local database as a backup to your AAA server, configure an authentication list with the **local** keyword following the **radius** or **tacacs+** keyword:

```
RTA(config)#aaa authentication ppp FOO radius local
```

To populate the local username database, use the **username** global configuration command:

```
RTA(config)#service password-encryption
RTA(config)#username donn password apple2
RTA(config)#username shirley pass tequila55
RTA(config)#username terry pass bigvan!
RTA(config)#username wes pass parson3
RTA(config)#username alex pass toonman21
```

where **service password-encryption** (if it's not already enabled) ensures that all passwords are stored in the encrypted format.

The command **username donn password apple2** creates a user whose login name is **donn** and password is **apple2**.

The remaining lines continue to populate the local database with the **username** command and login-password pairs.

Configuring Authorization

As mentioned earlier, the authorization task of AAA limits what a user is allowed to do on the network. Your ability to control time-of-day access, destination addresses, protocols, or other privileges depends on your AAA server and the features it supports. Profiles configured in the AAA server contain the access privileges of your users. Your router then consults the AAA server over the RADIUS or TACACS+ protocol and applies those privileges to users as they
log in.

To configure a router to consult a AAA server for authorization of PPP sessions, use the **aaa authorization network** command. The following example uses the TACACS+ protocol (for RADIUS, use the keyword **radius** instead of **tacacs+**):

```
RTA(config)#aaa authorization network tacacs+ none
```

where the keyword **none** is a backup for TACACS+ and tells the router to allow authorization to succeed if the TACACS+ server is unavailable.

NOTE The command **aaa authorization network** is also used to authorize Serial Line IP (SLIP) and Apple Remote Access Protocol (ARAP) sessions with AAA.

To configure authorization of router logins (EXEC prompt), use the **aaa authorization exec** global configuration command. The following example uses the TACACS+ protocol (for RADIUS use the keyword **radius** instead of **tacacs+**):

```
RTA(config)#aaa authorization exec tacacs+ none
```

Configuring Accounting

Like authorization, the accounting capabilities you have are largely dictated by the features of your AAA server. Still, you must instruct the AAA client in the router to use either RADIUS
or TACACS+ for these accounting transactions.

Cisco IOS supports three kinds of accounting methods:

- **Stop-only**—The router sends a RADIUS or TACACS+ accounting record to the AAA server only when a user's session ends. This provides a minimal level of accounting.
- **Start-stop**—The router sends a RADIUS or TACACS+ accounting record to the AAA server at the start and end of a user's session.
- **Wait-start**—The router sends a RADIUS or TACACS+ accounting record to the AAA server at the start of the session and waits for an acknowledgment from the server before starting the user's service. As in start-stop, the router also sends an accounting record at the end of the session.

Accounting records typically contain information about the session such as when the session started or stopped, the duration of the session, the number of bytes and packets transmitted, CLID information, and commands that were issued.

To enable accounting on a router, use the **aaa accounting** global configuration command. The following example configures RADIUS accounting for network sessions such as PPP:

```
RTA(config)#aaa accounting network start-stop radius
```

where the keyword **start-stop** tells the router to use the start-stop accounting method mentioned earlier. For TACACS+ use the keyword **tacacs+** instead of **radius**, and for router logins use the keyword **exec** instead of **network**. For stop-only or wait-start accounting, replace the keyword **start-stop** with **stop-only** or **wait-start**.

TIP To view the accounting information for active sessions, issue the **show accounting** command.

Pointing the Router to the RADIUS or TACACS+ AAA Server

You must configure the router with the IP address (or DNS name) and security key of your AAA server. Without these two pieces of data, the router will not communicate with the server over the RADIUS or TACACS+ protocol.

The security key is a password used to authenticate the router to the AAA server and to encrypt data exchanges between them.

The following example configures Router A with the RADIUS server's IP address and security key:

```
RTA(config)#radius-server host 192.168.60.1
RTA(config)#radius-server key Xj3lZtc8P9
```

In place of the IP address (**192.168.60.1** in the example), you can type the DNS name of the RADIUS server if your router is using DNS. See Appendix E for configuring a DNS server on a router. The key (**Xj3lZtc8P9** in the example) must match the key configured in the RADIUS AAA server.

NOTE The key is a shared secret between the RADIUS clients (the routers) and the RADIUS server and must be kept confidential.

Pointing a router to a TACACS+ server follows a similar logic. Here's an example:

```
RTA(config)#tacacs-server host 192.168.70.2
RTA(config)#tacacs-server key g7P9v23W1b
```

where **192.168.70.2** is the IP address of the TACACS+ server and **g7P9v23W1b** is the security key. The keys **Xj3lZtc8P9** and **g7P9v23W1b** in these examples were chosen so that they are hard to guess.

NOTE You can use multiple **tacacs-server host** commands to specify additional servers. IOS searches for hosts in the order in which you specify them. This also applies to the **radius-server host** command.

Other IOS Commands for Basic Security

The commands presented in this section enhance network security and are recommended for routers connected to untrusted networks such as the Internet.

Cisco routers have several minor TCP/IP services that might be abused by hackers. This section covers some common attacks and illustrates how to defend against them.

The topics of this section are

- Disable TCP and UDP Small Servers
- Disable IP Source Routing
- Disable CDP on Public Links
- Disable Directed Broadcasts on Interfaces

Disable TCP and UDP Small Servers

By default, so-called TCP and UDP *small servers* are enabled. These are relatively simple protocol services required for standards compliance. However, hackers can abuse these services, so unless you have a compelling reason for their existence, disable them as follows:

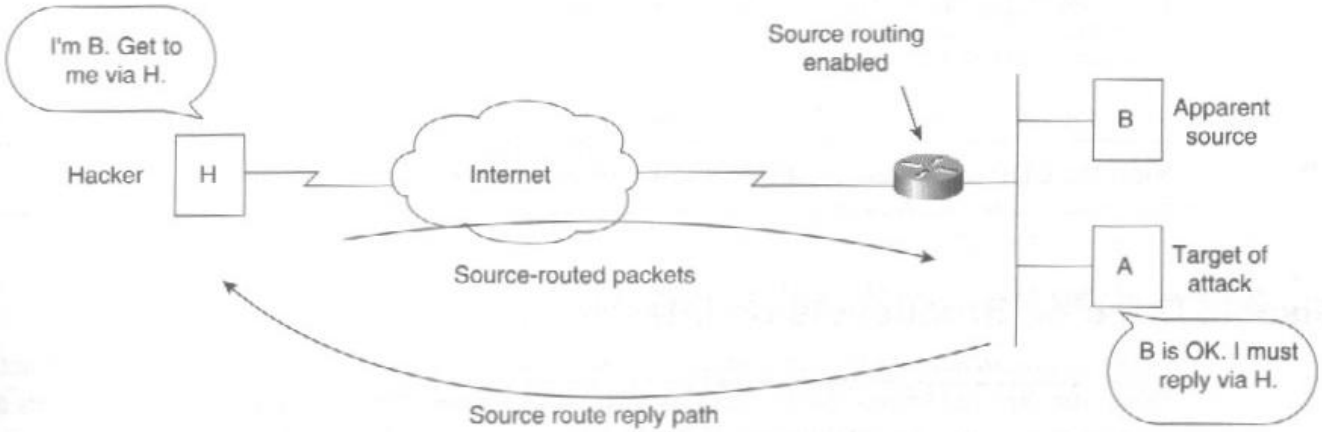
```
RTA(config)#no service tcp-small-servers
RTA(config)#no service udp-small-servers
```

TIP The TCP small servers are Echo, Discard, Chargen, and Daytime. The UDP small servers are Echo, Discard, and Chargen.

Disable IP Source Routing

IP source routing is rarely used. On occasion, it's used for troubleshooting. However, a hacker might attempt to communicate with one of your hosts by inserting himself or herself as an intermediary stop between two legitimate host addresses. Figure 6-11 illustrates the scheme.

Figure 6-11 *A Hacker Attacking with IP Source-Routing*



The hacker, H, pretends to be an intermediary hop in a source-routed path from Host B to Host A. H creates a request and a fictitious source-route path with B as the source and H as the middle hop. H sends this to A. Host A looks at the source address of the packet, sees that it's Host B, decides that B is friendly because it's on the same subnet, and sends a reply back to B along the source-routed path with H as the next hop. H is now communicating with A.

The hacker could do this if both the router and Host A have IP source-routing enabled. To comply with the standards, Cisco routers and just about all TCP/IP hosts have IP source-routing on by default. To disable IP source-routing on a router, issue the **no ip source-route** global configuration command:

```
RTA#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
RTA(config)#no ip source-route
```

TIP See RFC 1122 for the details of IP source routing.

Disable CDP on Public Links

Cisco Discovery Protocol (CDP) is a feature that gives you useful information about other Cisco devices (called *neighbors*) attached to a router. CDP information is helpful for determining the topology of a network and for troubleshooting. This information could also be useful to a hacker, however, so disabling it on interfaces that point to an untrusted network is a

good idea. CDP packets travel only between neighboring Cisco devices; they do not span multiple hops or roam around the network, so you can safely keep CDP enabled between trusted devices in a private network.

To disable CDP on an interface:

```
RTA(config)#int s1
RTA(config-if)#no cdp enable
```

To disable CDP on the whole router (all interfaces):

```
RTA(config)#no cdp run
```

TIP View the CDP information with the **show cdp neighbors** command.

Disable Directed Broadcasts on Interfaces

Directed broadcasts are IP-based broadcasts sent over a routed network to a destination subnet. When the directed broadcast reaches the destination subnet, the last router in the path issues a data link layer broadcast that is processed by all nodes on that subnet. Hosts and routers might suffer from processing overload if a flood of broadcasts storm their subnet. Therefore, if you do not have a compelling reason to use directed broadcasts it is recommended that you disable them to prevent problems from misconfigured hosts and abuse by hackers. Disabling directed broadcasts does not affect normal local broadcasts necessary for host-to-host and host-to-router communication. See RFC 1812 for more information on directed broadcasts.

Here's an example that disables directed broadcasts on an Ethernet interface:

```
RTA(config)#int e0
RTA(config-if)#no ip directed-broadcast
```

The command **no ip directed-broadcast** disables directed broadcast processing on the interface and protects the subnet on Ethernet0 from directed broadcasts.

Summary

Security is a significant area of concern for network engineers and managers. This chapter covered some basic security principles fundamental to maintaining a secure Cisco network. Although this chapter is by no means a complete set of all the network security you need, it is a good foundation for building more advanced security services. Chapter 7 and Chapter 8 cover some of these advanced services.

The following are the key concepts of this chapter:

- The most common use of access lists is for filtering traffic, but the access list syntax is also used in other IOS features because of its flexibility.
- Standard access lists match on source address only.
- Extended access lists can match on several more aspects of the packet (destination address, protocol, port number, IP precedence, and so on).
- Access list rules consist of an address bit pattern and a wildcard that defines the match and don't care bits.
- Access list rules are processed in sequential order. List processing for a packet stops when the router reaches the first matching rule.
- You need to consider several things when designing an access list. One that you should always remember is the invisible deny-any rule at the end of every access list.
- Extended access lists maintain statistics on their rules and can be configured to log activity.
- You can deploy access lists to combat spoofing attacks.
- Securing access to a router involves protecting the enable mode password, securing Telnet access, and securing the console port.
- AAA provides secure remote access for both network connections (such as PPP) and router logins.
- Centralized AAA servers simplify the management of passwords, policies, and billing.
- Cisco routers support RADIUS and TACACS+ and are clients of AAA servers.
- Cisco routers have several minor TCP/IP services that can be disabled to prevent abuse by hackers. These services are TCP/UDP small servers, IP source routing, CDP, and directed broadcasts.
- There are no 100 percent hacker-proof networks because new attacks are invented every day. Good network security is a continuous process of deploying defenses, auditing activity, and keeping current with new attacks.