

## Understanding ACL Concepts

This section describes ACL concepts.

### Using Masks

Masks are used with IP addresses in IP ACLs to specify what should be permitted and denied. Masks to configure IP addresses on *interfaces* start with 255 and have the large values on the left side (for example, IP address 209.165.202.129 with a 255.255.255.224 mask). Masks for IP ACLs are the reverse (for example, mask 0.0.0.255). This is sometimes called an inverse mask or a wildcard mask. When the value of the mask is broken down into binary (0s and 1s), the results determine which address bits are to be considered in processing the traffic. A 0 indicates that the address bits must be considered (exact match); a 1 in the mask is a "don't care". The following table further explains this concept.

Mask Example	
network address (traffic that is to be processed)	10.1.1.0
mask	0.0.0.255
network address (binary)	00001010.00000001.00000001.00000000
mask (binary)	00000000.00000000.00000000.11111111

Based on the binary mask, you can see that the first three sets (octets) must match the given binary network address exactly (00001010.00000001.00000001). The last set of numbers are "don't cares" (.11111111). Therefore, all traffic beginning with 10.1.1. will match since the last octet is "don't care". So, with this mask, network addresses 10.1.1.1 through 10.1.1.255 (10.1.1.x) will be processed.

The ACL inverse mask can also be determined by subtracting the normal mask from 255.255.255.255. In the following example, the inverse mask is determined for network address 172.16.1.0 with a normal mask of 255.255.255.0.

255.255.255.255 - 255.255.255.0 (normal mask) = 0.0.0.255 (inverse mask)

Note the following ACL equivalents.

- The source/source-wildcard of 0.0.0.0/255.255.255.255 means "any".
- The source/wildcard of 10.1.1.2/0.0.0.0 is the same as "host 10.1.1.2".

### Summarizing ACLs

**Note:** Subnet masks can also be represented as a fixed length notation. For example, 192.168.10.0/24 would represent 192.168.10.0 255.255.255.0.

The following describes how to summarize a range of networks into a single network for ACL optimization. Consider the following networks.

192.168.32.0/24  
192.168.33.0/24  
192.168.34.0/24  
192.168.35.0/24

192.168.36.0/24  
 192.168.37.0/24  
 192.168.38.0/24  
 192.168.39.0/24

The first two octets and the last octet are the same for each network. The following is an explanation of how to summarize these into a single network.

The third octet for the above networks can be written as follows, according to the octet bit position and address value for each bit.

Decimal	128	64	32	16	8	4	2	1
32	0	0	1	0	0	0	0	0
33	0	0	1	0	0	0	0	1
34	0	0	1	0	0	0	1	0
35	0	0	1	0	0	0	1	1
36	0	0	1	0	0	1	0	0
37	0	0	1	0	0	1	0	1
38	0	0	1	0	0	1	1	0
39	0	0	1	0	0	1	1	1
	M	M	M	M	M	D	D	D

Since the first five bits match, the above eight networks can be summarized into one network (192.168.32.0/21 or 192.168.32.0 255.255.248.0); all eight possible combinations of the three low-order bits are relevant for the network ranges in question. The following command defines an ACL that permits this network. Subtracting 255.255.248.0 (normal mask) from 255.255.255.255 yields 0.0.7.255.

```
access-list acl_permit permit ip 192.168.32.0 0.0.7.255
```

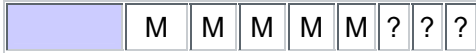
For further explanation, consider the following set of networks.

192.168.146.0/24  
 192.168.147.0/24  
 192.168.148.0/24  
 192.168.149.0/24

The first two octets and the last octet are the same for each network. The following is an explanation of how to summarize these.

The third octet for the above networks can be written as follows, according to the octet bit position and address value for each bit.

Decimal	128	64	32	16	8	4	2	1
146	1	0	0	1	0	0	1	0
147	1	0	0	1	0	0	1	1
148	1	0	0	1	0	1	0	0
149	1	0	0	1	0	1	0	1



Unlike previous example, you cannot summarize these networks into a single network—you need a minimum of two networks. The above networks can be summarized into two networks, as shown below.

- For networks 192.168.146.x and 192.168.147.x, all bits match except for the last one, which is a "don't care". This can be written as 192.168.146.0/23 (or 192.168.146.0 255.255.254.0).
- For networks 192.168.148.x and 192.168.149.x, all bits match except for the last one, which is a "don't care". This can be written as 192.168.148.0/23 (or 192.168.148.0 255.255.254.0).

The following defines a summarized ACL for the above networks.

```
access-list 10 permit ip 192.168.146.0 0.0.1.255
access-list 10 permit ip 192.168.148.0 0.0.1.255
```

### Processing ACLs

Traffic coming into the router is compared to ACL entries based on the order that the entries occur in the router. New statements are added to the end of the list. The router keeps looking until it has a match. If no matches are found when the router reaches the end of the list, the traffic is denied. For this reason, you should have the frequently hit entries at the top of the list. There is an "implied deny" for traffic that is not permitted. A single-entry ACL with only one "deny" entry has the effect of denying all traffic. You must have at least one "permit" statement in an ACL or all traffic will be blocked. The following two ACLs (101 and 102) have the same effect.

```
access-list 101 permit ip 10.1.1.0 0.0.0.255 172.16.1.0 0.0.0.255
```

```
access-list 102 permit ip 10.1.1.0 0.0.0.255 172.16.1.0 0.0.0.255
access-list 102 deny ip any any
```

In the following example, the last entry is sufficient. You do not need the first three entries because TCP includes Telnet, and IP includes TCP, User Datagram Protocol (UDP), and Internet Control Message Protocol (ICMP).

```
access-list 101 permit tcp host 10.1.1.2 host 172.16.1.1 eq telnet
access-list 101 permit tcp host 10.1.1.2 host 172.16.1.1
access-list 101 permit udp host 10.1.1.2 host 172.16.1.1
access-list 101 permit ip 10.1.1.0 0.0.0.255 172.16.1.0 0.0.0.255
```

### Defining Ports and Message Types

In addition to defining ACL source and destination, it is possible to define ports, ICMP message types, and other parameters. A good source of information for well-known ports is

[RFC 1700](#).  ICMP message types are explained in [RFC 792](#). 

The router can display descriptive text on some of the well-known ports. Use a ? for help.

```
access-list 102 permit tcp host 10.1.1.1 host 172.16.1.1 eq ?
  bgp          Border Gateway Protocol (179)
  chargen     Character generator (19)
  cmd         Remote commands (rcmd, 514)
```

During configuration, the router also converts numeric values to more user-friendly values. The following is an example wherein typing the ICMP message type number causes the router to convert the number to a name.

```
access-list 102 permit icmp host 10.1.1.1 host 172.16.1.1 14
```

becomes

```
access-list 102 permit icmp host 10.1.1.1 host 172.16.1.1 timestamp-reply
```

### Applying ACLs

You can define ACLs without applying them. However, the ACLs will have no effect until they are applied to the router's interface. It is a good practice to apply the ACL on the interface closest to the source of the traffic. As shown in the example below, when you are trying to block traffic from source to destination, you can apply an inbound ACL to E0 on router A instead of an outbound list to E1 on router C.



### Defining In, Out, Source, and Destination

The terms "in", "out", "source", and "destination" are used as referenced by the router. Traffic on the router could be compared to traffic on the highway. If you were a law enforcement officer in Pennsylvania and wanted to stop a truck going from Maryland to New York, the truck's source would be Maryland and the truck's destination would be New York. The roadblock could be applied at the Pennsylvania–New York border ("out") or the Maryland–Pennsylvania border ("in").

When referring to a router, these terms have the following meanings.

- **Out** - Traffic that has already been through the router and is leaving the interface; the source would be where it's been (on the other side of the router) and the destination is where it's going.
- **In** - Traffic that is arriving on the interface and which will go through the router; the source would be where it's been and the destination is where it's going (on the other side of the router).

The "in" ACL has a source on a segment of the interface to which it is applied and a destination off of any other interface. The "out" ACL has a source on a segment of any interface other than the interface to which it is applied and a destination off of the interface to which it is applied.

### Editing ACLs

Editing an ACL requires special attention. For example, if you intended to delete a specific line from an existing numbered ACL as shown below, the entire ACL would be deleted.

```
router#configure terminal
```

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

```
router(config)#access-list 101 deny icmp any any
```

```
router(config)#access-list 101 permit ip any any
```

```
router(config)#^Z
```

```
router#show access-list
```

```
Extended IP access list 101
```

```
deny icmp any any
```

```
permit ip any any
```

```
router#
```

```
*Mar 9 00:43:12.784: %SYS-5-CONFIG_I: Configured from console by console
```

```
router#configure terminal
```

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

```
router(config)#no access-list 101 deny icmp any any
```

```
router(config)#^Z
```

```
router#show access-list
```

```
router#
```

```
*Mar 9 00:43:29.832: %SYS-5-CONFIG_I: Configured from console by console
```

To edit numbered ACLs, copy the configuration of the router to a TFTP server or a text editor such as Notepad, make any changes, and copy the configuration back to the router.

You can also do the following.

```
router#configure terminal  
Enter configuration commands, one per line.  
router(config)#ip access-list extended test  
router(config-ext-nacl)#permit ip host 2.2.2.2 host 3.3.3.3  
router(config-ext-nacl)#permit tcp host 1.1.1.1 host 5.5.5.5 eq www  
router(config-ext-nacl)#permit icmp any any  
router(config-ext-nacl)#permit udp host 6.6.6.6 10.10.10.0 0.0.0.255  
eq domain  
router(config-ext-nacl)#^Z  
1d00h: %SYS-5-CONFIG_I: Configured from console by consoles-1
```

```

router#show access-list
Extended IP access list test
  permit ip host 2.2.2.2 host 3.3.3.3
  permit tcp host 1.1.1.1 host 5.5.5.5 eq www
  permit icmp any any
  permit udp host 6.6.6.6 10.10.10.0 0.0.0.255 eq domain

router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
router(config)#ip access-list extended test
!--- ACL entry deleted.
router(config-ext-nacl)#no permit icmp any any
!--- ACL entry added.
router(config-ext-nacl)#permit gre host 4.4.4.4 host 8.8.8.8
router(config-ext-nacl)#^Z
1d00h: %SYS-5-CONFIG_I: Configured from console by consoles-1

router#show access-list
Extended IP access list test
  permit ip host 2.2.2.2 host 3.3.3.3
  permit tcp host 1.1.1.1 host 5.5.5.5 eq www
  permit udp host 6.6.6.6 10.10.10.0 0.0.0.255 eq domain
  permit gre host 4.4.4.4 host 8.8.8.8

```

As you can see, any deletions will be removed from the ACL and any additions will be made to the end of the ACL.

## Troubleshooting

### How do I remove an ACL from an interface?

To remove an ACL from an interface, go into configuration mode and enter **no** in front of the **access-group** command, as shown in the following example.

```

interface <interface>
no ip access-group # in|out

```

### What do I do when too much traffic is being denied?

If too much traffic is being denied, study the logic of your list or try defining and applying an additional broader list. The **show ip access-lists** command provides a packet count showing which ACL entry is being hit.

Using the **log** keyword at the end of the individual ACL entries shows the ACL number and whether the packet was permitted or denied, in addition to port-specific information.

**Note:** The **log-input** keyword exists in Cisco IOS Software Release 11.2 and later, and in certain Cisco IOS Software Release 11.1 based software created specifically for the service provider market. Older software does not support this keyword. Use of this keyword includes the input interface and source MAC address where applicable.

### How do I debug at the packet level using a Cisco router?

The steps below explain the debug process. Before beginning, be certain that there are no currently applied ACLs, that there is an ACL, and that fast switching is not disabled.

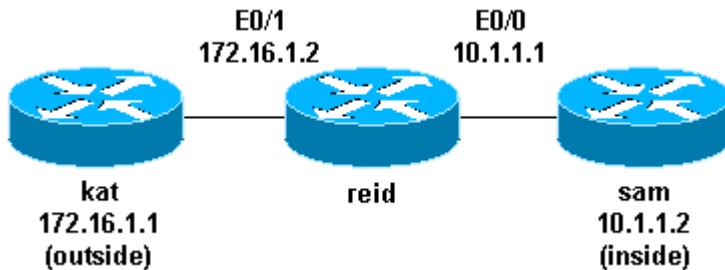
**Note:** Use extreme caution when debugging a system with heavy traffic. You can debug specific traffic using an ACL. However, be sure of the process and the traffic flow.

1. Capture the desired data using the **access-list** command. In the example below, the data capture is set for the destination address of 10.2.6.6 or the source address of 10.2.6.6.
  2. `access-list 101 permit ip any host 10.2.6.6`
  3. `access-list 101 permit ip host 10.2.6.6 any`
4. Disable fast switching on the interfaces involved. You will only see the first packet if fast switching is not disabled.
5. `config interface`  
`no ip route-cache`
6. To display **debug** command output and system error messages for the current terminal and session, use the **terminal monitor** command in enable mode.
7. Begin the debug process using the **debug ip packet 101** or **debug ip packet 101 detail** command.
8. To stop the debug process, execute the **no debug all** command in enable mode, and the **interface configuration** command.
9. Restart caching.
10. `config interface`  
`ip route-cache`

## Types of IP ACLs

This section of the document describes ACL types.

### Network Diagram



### Standard ACLs

Standard ACLs are the oldest type of ACL, dating back as early as Cisco IOS Software Release 8.3. Standard ACLs control traffic by comparing the source address of the IP packets to the addresses configured in the ACL.

The following is the command syntax format of a standard ACL.

```
access-list access-list-number {permit|deny} {host|source source-wildcard|any}
```

In all software releases, the *access-list-number* can be anything from 1 to 99. In Cisco IOS Software Release 12.0.1, standard ACLs began using additional numbers (1300 to 1999). These additional numbers are referred to as expanded IP ACLs. Cisco IOS Software Release 11.2 added the ability to use list *name* in standard ACLs.

A *source/source-wildcard* setting of 0.0.0.0/255.255.255.255 can be specified as **any**. The wildcard can be omitted if it is all zeros. Therefore, host 10.1.1.2 0.0.0.0 is the same as host 10.1.1.2.

After the ACL is defined, it must be applied to the interface (inbound or outbound). In early software releases, "out" was the default when a keyword "out" or "in" was not specified. The direction must be specified in later software releases.

```
interface <interface>
ip access-group number {in|out}
```

The following is an example of using a standard ACL to block all traffic except that from source 10.1.1.x.

```
interface Ethernet0/0
ip address 10.1.1.1 255.255.255.0
ip access-group 1 in
access-list 1 permit 10.1.1.0 0.0.0.255
```

### Extended ACLs

Extended ACLs were introduced in Cisco IOS Software Release 8.3. Extended ACLs control traffic by comparing the source *and* destination addresses of the IP packets to the addresses configured in the ACL.

The following is the command syntax format of extended ACLs. (Lines are wrapped here for spacing considerations.)

### IP

```
access-list access-list-number [dynamic dynamic-name [timeout minutes]]
{deny | permit} protocol source source-wildcard
destination destination-wildcard [precedence precedence]
[tos tos] [log | log-input] [time-range time-range-name]
```

### ICMP

```
access-list access-list-number [dynamic dynamic-name [timeout minutes]]
{deny | permit} icmp source source-wildcard
destination destination-wildcard
[icmp-type | [[icmp-type icmp-code] | [icmp-message]]]
[precedence precedence] [tos tos] [log | log-input]
[time-range time-range-name]
```

### TCP

```
access-list access-list-number [dynamic dynamic-name [timeout minutes]]
{deny | permit} tcp source source-wildcard [operator [port]]
destination destination-wildcard [operator [port]] [established]
[precedence precedence] [tos tos] [log | log-input]
[time-range time-range-name]
```

### UDP

```
access-list access-list-number [dynamic dynamic-name [timeout minutes]]
```

```
{deny | permit} udp source source-wildcard [operator [port]]
destination destination-wildcard [operator [port]]
[precedence precedence] [tos tos] [log | log-input]
[time-range time-range-name]
```

In all software releases, the *access-list-number* can be 101 to 199. In Cisco IOS Software Release 12.0.1, extended ACLs began using additional numbers (2000 to 2699). These additional numbers are referred to as expanded IP ACLs. Cisco IOS Software Release 11.2 added the ability to use list *name* in extended ACLs.

The value of 0.0.0.0/255.255.255.255 can be specified as **any**. After defining the ACL, it must be applied to the interface (inbound or outbound). In early software releases, "out" was the default when a keyword "out" or "in" was not specified. The direction must be specified in later software releases.

```
interface <interface>
ip access-group {number|name} {in|out}
```

The following extended ACL is used to permit traffic on the 10.1.1.x network (inside) and to receive ping responses from the outside while preventing unsolicited pings from people outside (permitting all other traffic).

```
interface Ethernet0/1
ip address 172.16.1.2 255.255.255.0
ip access-group 101 in
access-list 101 deny icmp any 10.1.1.0 0.0.0.255 echo
access-list 101 permit ip any 10.1.1.0 0.0.0.255
```

**Note:** Some applications (such as network management) require pings for a keepalive function. If this is the case, you may wish to limit blocking inbound pings (or be more granular in permitted/denied IPs).

### Lock and Key (Dynamic ACLs)

Lock and key (also known as dynamic ACLs) was introduced in Cisco IOS Software Release 11.1. This feature is dependent on Telnet, authentication (local or remote), and extended ACLs.

Lock and key configuration starts with applying an extended ACL to block traffic through the router. Users wanting to traverse the router are blocked by the extended ACL until they Telnet to the router and are authenticated. The Telnet connection then drops and a single-entry dynamic ACL is added to the existing extended ACL. This will permit traffic for a particular time period (idle and absolute timeouts are possible).

The following is the command syntax format for configuring lock and key with local authentication.

```
username username password password
interface <interface>
ip access-group {number|name} {in|out}
```

The single-entry ACL in the following command will be dynamically added to the existing ACL after authentication.

```
access-list access-list-number dynamic name {permit|deny} [protocol]
{source source-wildcard|any} {destination destination-wildcard|any}
[precedence precedence][tostos][established] [log|log-input]
[operator destination-port|destination port]
```

```
line vty line_range
```

```
login local
```

The following is a basic example of lock and key.

```
username test password 0 test
```

```
!--- 10 (minutes) is the idle timeout.
```

```
username test autocommand access-enable host timeout 10
```

```
interface Ethernet0/0
```

```
ip address 10.1.1.1 255.255.255.0
```

```
ip access-group 101 in
```

```
access-list 101 permit tcp any host 10.1.1.1 eq telnet
```

```
!--- 15 (minutes) is the absolute timeout.
```

```
access-list 101 dynamic testlist timeout 15 permit ip 10.1.1.0
```

```
0.0.0.255
```

```
172.16.1.0 0.0.0.255
```

```
line vty 0 4
```

```
login local
```

After the user at 10.1.1.2 makes a Telnet connection to 10.1.1.1, the dynamic ACL is applied. The connection is then dropped, and the user can go to the 172.16.1.x network.

## IP Named ACLs

IP named ACLs were introduced in Cisco IOS Software Release 11.2, allowing standard and extended ACLs to be given names instead of numbers.

The following is the command syntax format for IP named ACLs.

```
ip access-list {extended|standard} name
```

The following is a TCP example

```
permit|deny tcp source source-wildcard [operator [port]]
```

```
destination destination-wildcard [operator [port]] [established]
```

```
[precedence precedence] [tos tos] [log] [time-range time-range-name]
```

The following is an example of using a named ACL to block all traffic except the Telnet connection from host 10.1.1.2 to host 172.16.1.1.

```
interface Ethernet0/0
```

```
ip address 10.1.1.1 255.255.255.0
```

```
ip access-group in_to_out in
```

```
ip access-list extended in_to_out
```

```
permit tcp host 10.1.1.2 host 172.16.1.1 eq telnet
```

## Reflexive ACLs

Reflexive ACLs were introduced in Cisco IOS Software Release 11.3. Reflexive ACLs allow IP packets to be filtered based on upper-layer session information. They are generally used to allow outbound traffic and to limit inbound traffic in response to sessions originating inside the router.

Reflexive ACLs can be defined only with extended named IP ACLs. They cannot be defined with numbered or standard named IP ACLs, or with other protocol ACLs. Reflexive ACLs can be used in conjunction with other standard and static extended ACLs.

The following is the syntax for various reflexive ACL commands.

```
interface
ip access-group {number|name} {in|out}

ip access-list extended name

permit protocol any any reflect name [timeoutseconds]

ip access-list extended name
```

```
evaluate name
```

The following is an example of permitting ICMP outbound and inbound traffic, while only permitting TCP traffic that had been initiated from inside (other traffic will be denied).

```
ip reflexive-list timeout 120

interface Ethernet0/1
 ip address 172.16.1.2 255.255.255.0
 ip access-group inboundfilters in
 ip access-group outboundfilters out

ip access-list extended inboundfilters
permit icmp 172.16.1.0 0.0.0.255 10.1.1.0 0.0.0.255
evaluate tcptraffic

!--- This ties the reflexive ACL part of the outboundfilters ACL,
!--- called tcptraffic, to the inboundfilters ACL.
ip access-list extended outboundfilters
permit icmp 10.1.1.0 0.0.0.255 172.16.1.0 0.0.0.255
permit tcp 10.1.1.0 0.0.0.255 172.16.1.0 0.0.0.255 reflect tcptraffic
```

### Time-Based ACLs Using Time Ranges

Time-based ACLs were introduced in Cisco IOS Software Release 12.0.1.T. While similar to extended ACLs in function, they allow for access control based on time. To implement time-based ACLs, a time range is created that defines specific times of the day and week. The time range is identified by a name and then referenced by a function. Therefore, the time restrictions are imposed on the function itself. The time range relies on the router's system clock. The router clock can be used, but the feature works best with Network Time Protocol (NTP) synchronization.

The following are time-based ACL commands.

```
!--- Defines a named time range.
time-range time-range-name
!--- Defines the periodic times.
periodic days-of-the-week hh:mm to [days-of-the-week] hh:mm
!--- Or, defines the absolute times.
```

```
absolute [start time date] [end time date]
!--- The time range used in the actual ACL.
ip access-list name|number <extended_definition>time-rangename_of_time-
range
```

In the following example, a Telnet connection is permitted from the inside to outside network on Monday, Wednesday, and Friday during business hours:

```
interface Ethernet0/0
ip address 10.1.1.1 255.255.255.0
ip access-group 101 in

access-list 101 permit tcp 10.1.1.0 0.0.0.255 172.16.1.0 0.0.0.255
eq telnet time-range EVERYOTHERDAY

time-range EVERYOTHERDAY
periodic Monday Wednesday Friday 8:00 to 17:00
```

### Commented IP ACL Entries

Commented IP ACL entries were introduced in Cisco IOS Software Release 12.0.2.T. Comments make ACLs easier to understand and can be used for standard or extended IP ACLs.

The following is the commented name IP ACL command syntax.

```
ip access-list {standard|extended} name
remark remark
```

The following is the commented numbered IP ACL command syntax.

```
access-list access-list-number remark remark
```

The following is an example of commenting a numbered ACL.

```
interface Ethernet0/0
ip address 10.1.1.1 255.255.255.0
ip access-group 101 in

access-list 101 remark permit_telnet
access-list 101 permit tcp host 10.1.1.2 host 172.16.1.1 eq telnet
```

### Context-Based Access Control

Context-based access control (CBAC) was introduced in Cisco IOS Software Release 12.0.5.T and requires the Cisco IOS Firewall feature set. CBAC inspects traffic that travels through the firewall to discover and manage state information for TCP and UDP sessions. This state information is used to create temporary openings in the firewall's access lists. This is done by configuring **ip inspect** lists in the direction of the flow of traffic initiation to allow return traffic and additional data connections for permissible sessions (sessions that originated from within the protected internal network).

The following is the syntax for CBAC.

```
ip inspect name inspection-name protocol [timeoutseconds]
```

The following is an example of using CBAC to inspect outbound traffic. Extended ACL 111 would normally block the return traffic (other than ICMP) without CBAC opening holes for the return traffic.

```
ip inspect name myfw ftp timeout 3600
```

```

ip inspect name myfw http timeout 3600
ip inspect name myfw tcp timeout 3600
ip inspect name myfw udp timeout 3600
ip inspect name myfw tftp timeout 3600
interface Ethernet0/1
    ip address 172.16.1.2 255.255.255.0
    ip access-group 111 in
    ip inspect myfw out
access-list 111 deny icmp any 10.1.1.0 0.0.0.255 echo
access-list 111 permit icmp any 10.1.1.0 0.0.0.255

```

## Authentication Proxy

Authentication proxy was introduced in Cisco IOS Software Release 12.0.5.T. This requires having the Cisco IOS Firewall feature set. Authentication proxy is used to authenticate inbound or outbound users, or both. Users who would normally be blocked by an ACL can bring up a browser to go through the firewall and authenticate on a TACACS+ or RADIUS server. The server passes additional ACL entries down to the router to allow the users through after authentication.

Authentication proxy is similar to lock and key (dynamic ACLs). The differences are explained below.

- Lock and key is turned on by a Telnet connection to the router; authentication proxy is turned on by HTTP through the router.
- Authentication proxy must use an external server.
- Authentication proxy can handle the addition of multiple dynamic lists; lock and key can only add one.
- Authentication proxy has an absolute timeout but no idle timeout; lock and key has both.

Refer to the [Cisco Secure Integrated Software Configuration Cookbook](#) for examples of authentication proxy.

## Turbo ACLs

Turbo ACLs were introduced in Cisco IOS Software Release 12.1.5.T and are found only on the 7200, 7500, and other high-end platforms. The turbo ACL feature is designed to process ACLs more efficiently in order to improve router performance.

Use the **access-list compiled** command for turbo ACLs. An example of a compiled ACL is shown below.

```

access-list 101 permit tcp host 10.1.1.2 host 172.16.1.1 eq
telnet
access-list 101 permit tcp host 10.1.1.2 host 172.16.1.1 eq ftp
access-list 101 permit udp host 10.1.1.2 host 172.16.1.1 eq
syslog
access-list 101 permit udp host 10.1.1.2 host 172.16.1.1 eq tftp
access-list 101 permit udp host 10.1.1.2 host 172.16.1.1 eq ntp

```

After defining the standard or extended ACL, use the **global configuration** command to compile.

```

!--- Tells the router to compile.
access-list compiled

```

```
Interface Ethernet0/1
ip address 172.16.1.2 255.255.255.0
!--- Applies to the interface.
ip access-group 101 in
```

The **show access-list compiled** command shows statistics about the ACL.

### Distributed Time-Based ACLs

Distributed time-based ACLs were introduced in Cisco IOS Software Release 12.2.2.T to implement time-based ACLs on VPN-enabled 7500 series routers. Before the introduction of the distributed time-based ACL feature, time-based ACLs were not supported on line cards for the Cisco 7500 series routers. If time-based ACLs were configured, they behaved as normal ACLs. If an interface on a line card was configured with time-based ACLs, the packets switched into the interface were not distributed switched through the line card but forwarded to the route processor for processing.

The syntax for distributed time-based ACLs is the same as for time-based ACLs with the addition of the commands regarding the status of the Inter Processor Communication (IPC) messages between the route processor and line card.

```
debug time-range ipc
show time-range ipc
clear time-range ipc
```

### Receive ACLs

Receive ACLs are used to increase security on Cisco 12000 routers by protecting the router's gigabit route processor (GRP) from unnecessary and potentially nefarious traffic. Receive ACLs were added as a special waiver to the maintenance throttle for Cisco IOS Software Release 12.0.21S2 and integrated into 12.0(22)S. See [GSR: Receive Access Control Lists](#) for further information.

### Infrastructure Protection ACLs

Infrastructure ACLs are used to minimize the risk and effectiveness of direct infrastructure attack by explicitly permitting only authorized traffic to the infrastructure equipment while permitting all other transit traffic. See [Protecting Your Core: Infrastructure Protection Access Control Lists](#) for further information.

### Transit ACLs

Transit ACLs are used to increase network security by explicitly permitting only required traffic into your network or networks. See [Transit Access Control Lists: Filtering at Your Edge](#) for further information.