Access Control List (ACL) for Security

- The original use of ACLs was as a **security feature** to decide whether traffic should be allowed to pass through the router.
- By default, a router allows **all traffic** to pass between its interfaces.
- When ACLs are applied, the router identifies traffic and decides whether to allow or deny it.

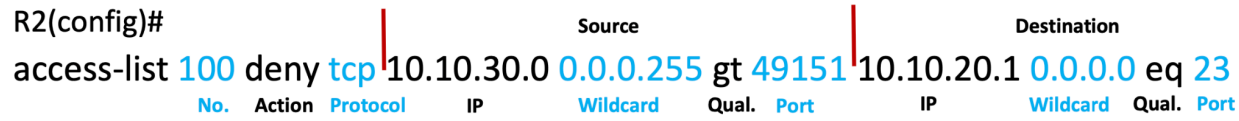
ACLs are also used in other software policies where traffic must be identified, for example:

- Identifying traffic to give better service in a QoS (Quality of Service) policy.
- Identifying traffic to translate to a different IP address in a NAT (Network Address Translation) policy.

Access Control Entries (ACE)

- An **Access Control List** is made up of **Access Control Entries (ACEs)**—a series of `permit` or `deny` rules.
- Each ACE is written on a **separate line**.

Access Control Entry Example:



Standard vs Extended ACLs (Original)

- Standard ACLs reference the source address only
- Extended ACLs check based on the **protocol, source address, and destination address, and port number**

```
Router(config)#access-list ?
  <1-99>     IP standard access list
  <100-199>  IP extended access list
```

ACL Improvement: Expanded Ranges

Cisco expanded the original ACL Ranges
Standard: 1-99, 1300-1999
Extended: 100-199, 2000-2699

Standard Access List Example:

```
R1(config) # access-list 1 deny 10.10.10.10     0.0.0.0
R1(config) # access-list 1 permit 10.10.10.0    0.0.0.255
```

- The default wildcard for a standard ACL is 0.0.0.0, meaning an individual host address

```
R1(config)# access-list 1 deny 10.10.10.10
```

- Don't forget to enter the wildcard when specifying an IP subnet

```
R1(config)# access-list 1 deny 10.10.10.0
```

Extended Access List Example

```
R1(config)# access-list 100 deny tcp 10.10.10.10 0.0.0.0 gt 49151
10.10.50.10 0.0.0.0 eq 23
R1(config)# access-list 100 permit tcp 10.10.10.0 0.0.0.255 gt 49151
10.10.50.10 0.0.0.0 eq telnet
```

- There is **no default wildcard mask for Extended ACLs**

ACL Improvement: Named ACLs

- You can now reference ACLs by number or by a name
- Named ACIs begin with the command 'ip access-list' instead of 'access-list'

```
R1(config)#ip access-list
 extended  Extended Access List
 standard  Standard Access List
```

Named ACL Syntax

```
R1(config)# ip access-list standard Flackbox-Demo
R1(config-std-nacl)#deny 10.10.10.10 0.0.0.0
R1(config-std-nacl)#permit 10.10.10.0 0.0.0.255
```

ACL Action

```
Router(config)#access-list 100 ?
  deny       Specify packets to reject
  permit     Specify packets to forward
  remark     Access list entry comment

Router(config)#access-list 100 permit ?
```

```
  ahp         Authentication Header Protocol
  eigrp       Cisco's EIGRP routing protocol
  esp         Encapsulation Security Payload
  gre         Cisco's GRE tunneling
  icmp        Internet Control Message Protocol
  ip          Any Internet Protocol
  ospf        OSPF routing protocol
  tcp         Transmission Control Protocol
  udp         User Datagram Protocol
```

- Use **TCP** or **UDP** if you want the ACE to apply to traffic for a specific **application** between a source and destination:

```
Router(config)#access-list 100 deny tcp 10.10.10.0 0.0.0.255 10.10.50.0
0.0.0.255 eq 80
```

- Use **IP** if you want the ACE to apply to **all traffic** between a source and destination:

```
Router(config)#access-list 100 deny ip 10.10.10.0 0.0.0.255 10.10.50.0
0.0.0.255
```

ACL Source IP

```
R1(config)#access-list 100 permit tcp ?
  A.B.C.D     Source address
  any         Any source host
  host        A single source host
```

Wildcards

Wildcards simplify how you define IP addresses. These examples are equivalent:

```
R1(config)#access-list 100 permit tcp 10.10.10.10 0.0.0.0
R1(config)#access-list 100 permit tcp host 10.10.10.10

R1(config)#access-list 100 permit tcp 0.0.0.0 255.255.255.255
R1(config)#access-list 100 permit tcp any
```

Source Port Number

```
Router(config)#access-list 100 permit tcp 10.10.10.0 0.0.0.255 ?
  A.B.C.D    Destination address
  any        Any destination host
  eq         Match only packets on a given port number
  gt         Match only packets with a greater port number
  host       A single destination host
  lt         Match only packets with a lower port number
  neq        Match only packets not on a given port number
  range      Match only packets in the range of port numbers
```

- Specifying the **source port number** is optional. If omitted, it defaults to **any port**.

Destination Address Format

The destination address uses the **same format** as the source address.

Example:

```
Router(config)#access-list 100 permit tcp host 10.10.10.10 10.10.20.0
0.0.0.255
```

Final Options

After the destination address, additional options can be specified such as **destination ports**, **TCP flags**, and **logging**:

```
Router(config)#access-list 100 permit tcp host 10.10.10.10 10.10.20.0
0.0.0.255 ?
  dscp         Match packets with given DSCP value
  eq           Match only packets on a given port number
  established  Match packets with ACK or RST flags set
  gt           Match only packets with a greater port number
  lt           Match only packets with a lower port number
  neq          Match only packets not on a given port number
  precedence   Match packets with given precedence value
  range        Match only packets in the range of port numbers
```

```
  <cr>
```

Verification – Show Access Lists

Use the following command to verify ACLs:

```
Router#show access-lists 100
Extended IP access list 100
  deny tcp 10.10.10.0 0.0.0.255 10.10.50.0 0.0.0.255 eq 80
  permit tcp host 10.10.10.10 10.10.20.0 0.0.0.255
```

Access Groups (applyign to Interfaces)

- ACLs are **applied** at the **interface level** with the **Access-Group** command
- ACLs can be applied in the inbound or outbound direction
- You can have maximum of one ACL per interface per direction
- You can have both an inbound and an outbound ACL on the same interface, but not 2 inbound or outbound ACLs
- An Interface can have no ACL applied, an inbound ACL only, an outbound ACL only, or ACLs in both directions

Access-Group Configuration

```
R1(config)# interface GigabitEthernet 0/1
R1(config-if)# ip access-group 100 out
R1(config-if)# ip access-group 101 in
```

```
R3# show ip interface f1/0 | include access list
      Outgoing access list is 100
      Inbound access list is 101
```

(‘not set’ if ACL is not applied)

Access Control Entry Order

The ACL is read by the router from top to bottom
As soon as a rule matches the packet, the `permit` or `deny` action is applied and the ACL is not processed any further
The order of rules is important

Example:
This will deny 10.10.10.10 but permit the rest of the 10.10.10.0/24 subnet

```
R1(config)# access-list 1 deny host 10.10.10.10
R1(config)# access-list 1 permit 10.10.10.0 0.0.0.255
```

Flipped order now:

```
R1(config)# access-list 1 permit 10.10.10.0 0.0.0.255
R1(config)# access-list 1 deny host 10.10.10.10
```

The first way is the correct way, as the flipped order will permit all 10.10.10.0/24 traffic since that match is applied first, therefore, it wouldn't deny the 10.10.10.10 IP

Note: Keep the more specific entries at the top and the more general ones at the bottom

Injecting ACEs in an Existing ACL

ACEs are automatically numbered in increments of 10

```
R1#sh access-list 110
Extended IP access list 110
  10 deny tcp host 10.10.10.10 host 10.10.50.10 eq telnet
  20 permit tcp 10.10.10.0 0.0.0.255 host 10.10.50.10 eq telnet
  30 deny tcp host 10.10.20.10 host 10.10.50.10 eq telnet
  40 permit tcp 10.20.10.0 0.0.0.255 host 10.10.50.10 eq telnet
```

Support for injecting ACEs in an existing ACL started in Named ACLs but is also supported in numbered ACLs now

```
R1(config)# ip access-list extended 110
R1(config-ext-nacl)#15 deny tcp host 10.10.10.11 host 10.10.50.10 eq telnet
```

```
R1#sh access-list 110
Extended IP access list 110
  10 deny tcp host 10.10.10.10 host 10.10.50.10 eq telnet
  15 deny tcp host 10.10.10.11 host 10.10.50.10 eq telnet
  20 permit tcp 10.10.10.0 0.0.0.255 host 10.10.50.10 eq telnet
  30 deny tcp host 10.10.20.10 host 10.10.50.10 eq telnet
  40 permit tcp 10.20.10.0 0.0.0.255 host 10.10.50.10 eq telnet
```

Implicit Deny All

- There is an implicit `deny any` rule at the bottom of ACLs
- If an ACL is not applied to an interface, all traffic is allowed
- If an ACL is applied, all traffic is denied except what is explicitly allowed
- Traffic from 10.10.10.0/24 will be permitted, everything else is denied
- I discovered that in ACLs, all traffic is denied by default unless explicitly permitted, so adding only a deny rule without any permit statements blocks all traffic

```
R1(config)# access-list 1 permit 10.10.10.0 0.0.0.255
```

Explicit Deny All

Many organizations include an explicit deny at the end of ACLs to log illegal traffic

```
R1(config)# access-list 1 permit 10.10.10.0 0.0.0.255
R1(config)# access-list 1 deny any log
```

Explicit Permit All

If an ACL is applied, all traffic is denied except what is explicitly allowed
If you want to reverse this so that all traffic is permitted except what's explicitly denied, add a permit all statement to the end of the ACL

Traffic from 10.10.10.0/24 is denied, everything else is permitted

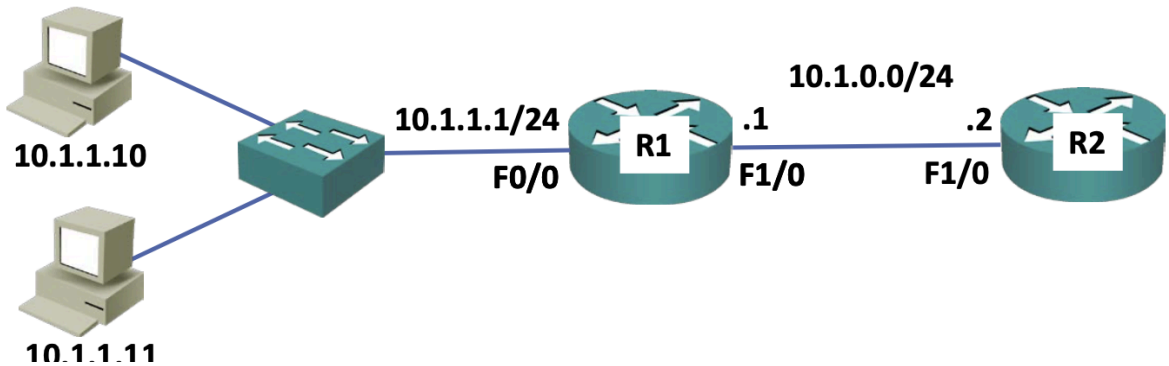
```
R1(config)# access-list 1 deny 10.10.10.0 0.0.0.255
R1(config)# access-list 1 permit any
```

Traffic Sourced From Router

ACLs applied to an interface **doesn't apply** to traffic that **originates from the router itself**

The hosts in the 10.1.1.0/24 subnet cannot telnet to R2
An administrator can telnet to R2 from the CLI on R1

```
R1(config)# access-list 100 deny tcp any any eq 23
R1(config)# interface f1/0
R1(config)# ip access-group 100 out
```


10.1.1.10
10.1.1.11
10.1.1.1/24
F0/0
R1
.1
F1/0
10.1.0.0/24
.2
F1/0
R2