Zone Based Policy Firewalls

Dr Christopher D. McDermott

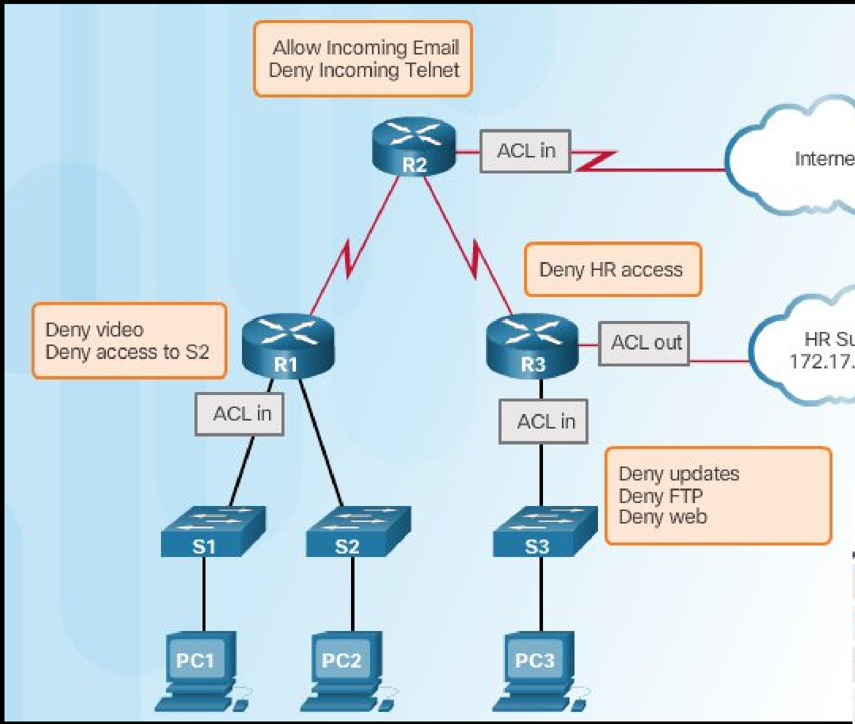
Outline

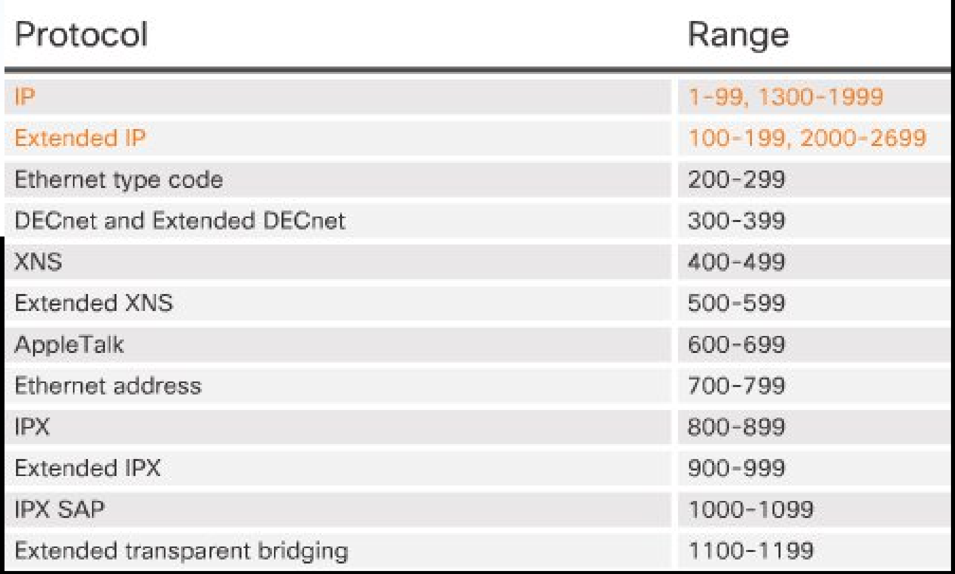
1. Introduction
2. Access Control Lists
   - Creating and editing
   - Mitigating attacks with ACLs
3. Firewalls
   - Types
   - Classic firewall
   - Demilitarised Zones
4. Zone-based Policy Firewalls

Introduction

Access Control Lists

Access Control Lists (Recap)

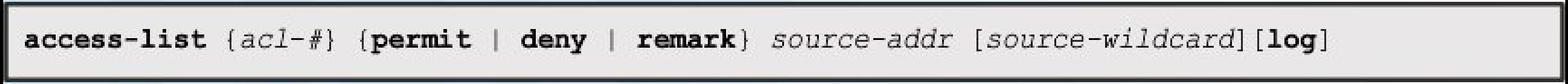




Protocol	Range
IP	1-99, 1300-1999
Extended IP	100-199, 2000-2699
Ethernet type code	200-299
DECnet and Extended DECnet	300-399
XNS	400-499
Extended XNS	500-599
AppleTalk	600-699
Ethernet address	700-799
IPX	800-899
Extended IPX	900-999
IPX SAP	1000-1099
Extended transparent bridging	1100-1199

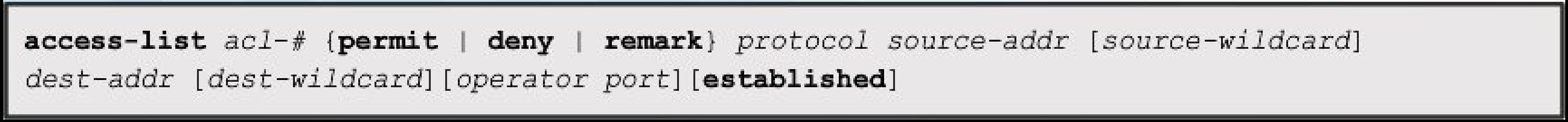
Configuring Numbered and Named ACLs

Standard Numbered ACL Syntax



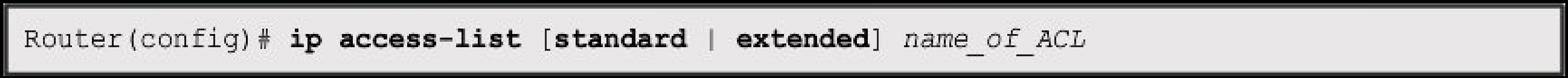
```
access-list {acl-#} {permit | deny | remark} source-addr [source-wildcard][log]
```

Extended Numbered ACL Syntax



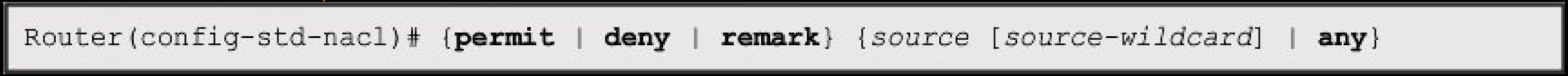
```
access-list acl-# {permit | deny | remark} protocol source-addr [source-wildcard]
dest-addr [dest-wildcard][operator port][established]
```

Named ACL Syntax



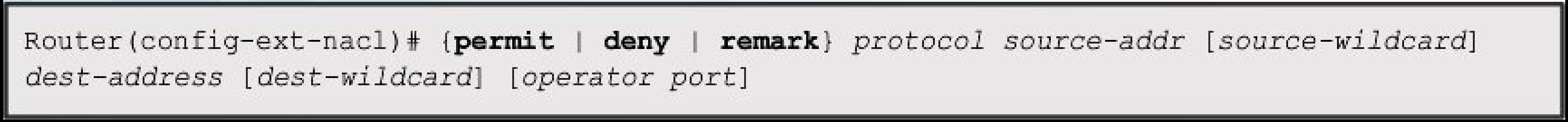
```
Router(config)# ip access-list [standard | extended] name_of_ACL
```

Standard ACE Syntax



```
Router(config-std-nacl)# {permit | deny | remark} {source [source-wildcard] | any}
```

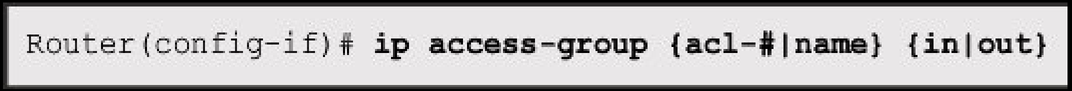
Extended ACE Syntax



```
Router(config-ext-nacl)# {permit | deny | remark} protocol source-addr [source-wildcard]
dest-address [dest-wildcard] [operator port]
```

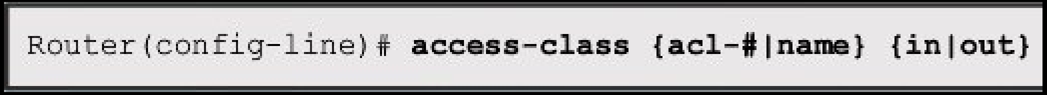
Applying an ACL

Syntax - Apply an ACL to an interface



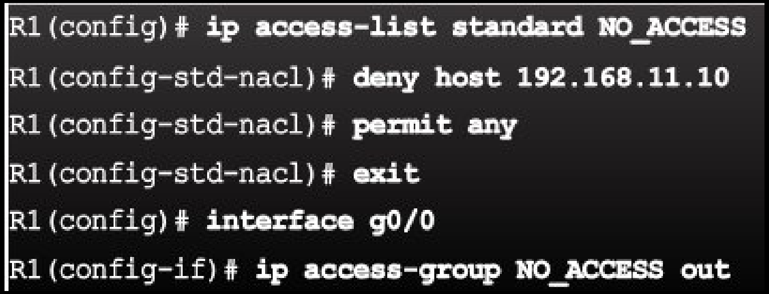
```
Router(config-if)# ip access-group {acl-#|name} {in|out}
```

Syntax - Apply an ACL to the VTY lines



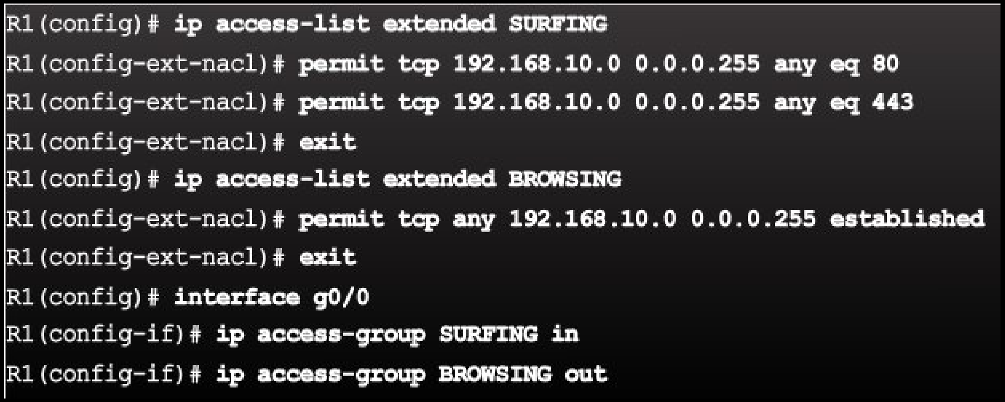
```
Router(config-line)# access-class {acl-#|name} {in|out}
```

Example - Named Standard ACL



```
R1(config)# ip access-list standard NO_ACCESS
R1(config-std-nacl)# deny host 192.168.11.10
R1(config-std-nacl)# permit any
R1(config-std-nacl)# exit
R1(config)# interface g0/0
R1(config-if)# ip access-group NO_ACCESS out
```

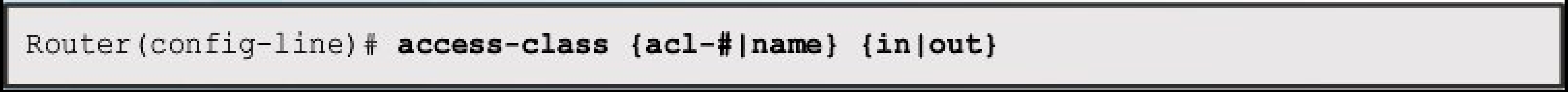
Example - Named Extended ACL



```
R1(config)# ip access-list extended SURFING
R1(config-ext-nacl)# permit tcp 192.168.10.0 0.0.0.255 any eq 80
R1(config-ext-nacl)# permit tcp 192.168.10.0 0.0.0.255 any eq 443
R1(config-ext-nacl)# exit
R1(config)# ip access-list extended BROWSING
R1(config-ext-nacl)# permit tcp any 192.168.10.0 0.0.0.255 established
R1(config-ext-nacl)# exit
R1(config)# interface g0/0
R1(config-if)# ip access-group SURFING in
R1(config-if)# ip access-group BROWSING out
```

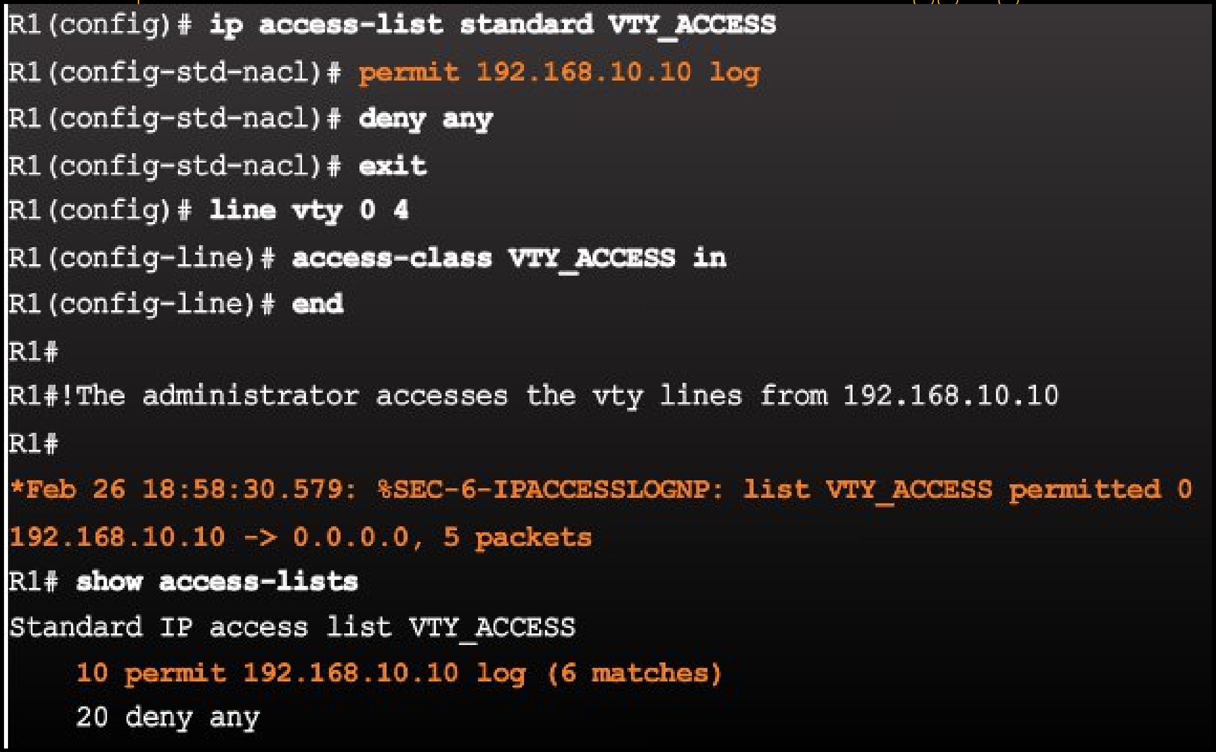
Applying an ACL (Cont.)

Syntax - Apply an ACL to the VTY lines



```
Router(config-line)# access-class {acl-#|name} {in|out}
```

Example - Named ACL on VTY lines with logging



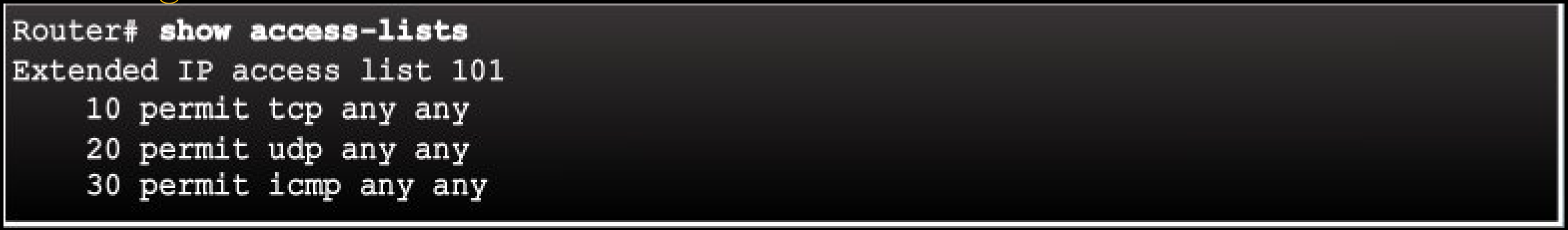
```
R1(config)# ip access-list standard VTY_ACCESS
R1(config-std-nacl)# permit 192.168.10.10 log
R1(config-std-nacl)# deny any
R1(config-std-nacl)# exit
R1(config)# line vty 0 4
R1(config-line)# access-class VTY_ACCESS in
R1(config-line)# end
R1#
R1#!The administrator accesses the vty lines from 192.168.10.10
R1#
*Feb 26 18:58:30.579: %SEC-6-IPACCESSLOGNP: list VTY_ACCESS permitted 0
192.168.10.10 -> 0.0.0.0, 5 packets
R1# show access-lists
Standard IP access list VTY_ACCESS
    10 permit 192.168.10.10 log (6 matches)
    20 deny any
```

ACL Configuration Guidelines (Summary)

- Create an ACL globally and then apply it
- Ensure the last statement is an implicit deny any or deny any any
- Statement order is important. ACLs are processed top-down. As soon as a statement is matched the ACL is exited
- Ensure that the most specific statements are the top of the list
- Only one ACL is allowed per interface, per protocol, per direction
- New statements for an existing ACL are added to the bottom of the ACL by default
- Router generated packets are not filtered by outbound ACLs
- Place standard ACLs as close to the destination as possible
- Place extended ACLs as close to the source as possible

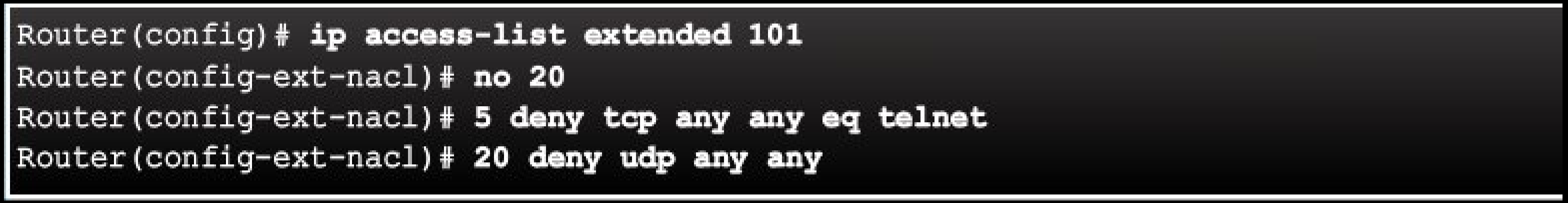
Editing Existing ACLs

Existing access list has three entries



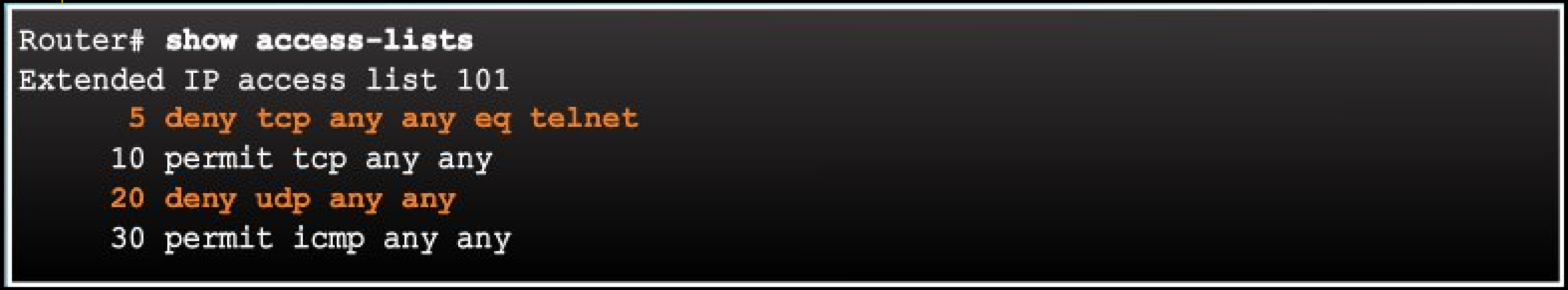
```
Router# show access-lists
Extended IP access list 101
    10 permit tcp any any
    20 permit udp any any
    30 permit icmp any any
```

Access list has been edited, which adds a new ACE and replaces ACE line 20.



```
Router(config)# ip access-list extended 101
Router(config-ext-nacl)# no 20
Router(config-ext-nacl)# 5 deny tcp any any eq telnet
Router(config-ext-nacl)# 20 deny udp any any
```

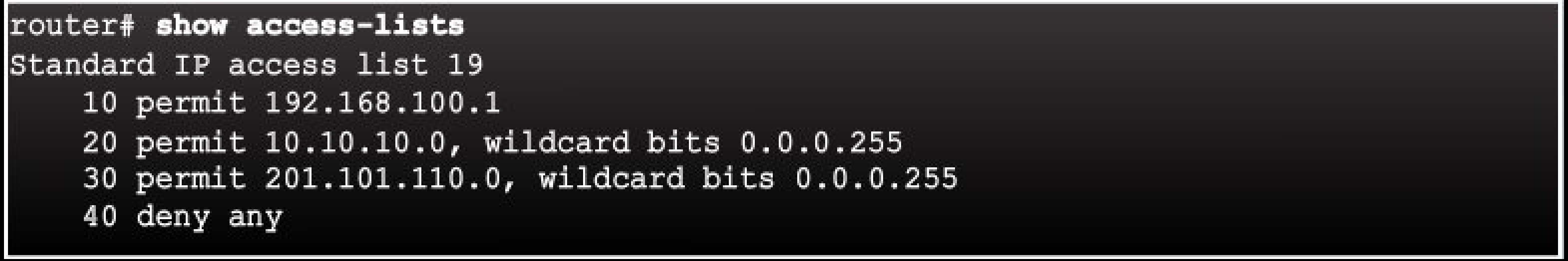
Updated access list has four entries



```
Router# show access-lists
Extended IP access list 101
      5 deny tcp any any eq telnet
     10 permit tcp any any
     20 deny udp any any
     30 permit icmp any any
```

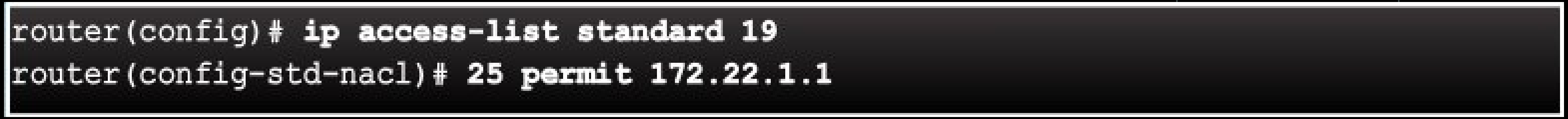
Sequence Numbers and Standard ACLs

Existing access list has four entries



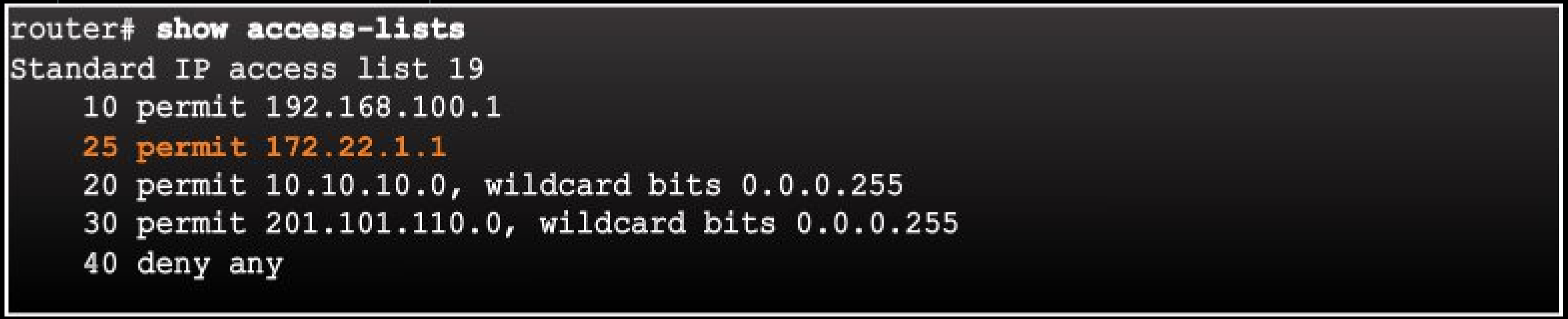
```
router# show access-lists
Standard IP access list 19
    10 permit 192.168.100.1
    20 permit 10.10.10.0, wildcard bits 0.0.0.255
    30 permit 201.101.110.0, wildcard bits 0.0.0.255
    40 deny any
```

Access list has been edited, which adds a new ACE that permits a specific IP



```
router(config)# ip access-list standard 19
router(config-std-nacl)# 25 permit 172.22.1.1
```

Updated access list places the new ACE before line 20

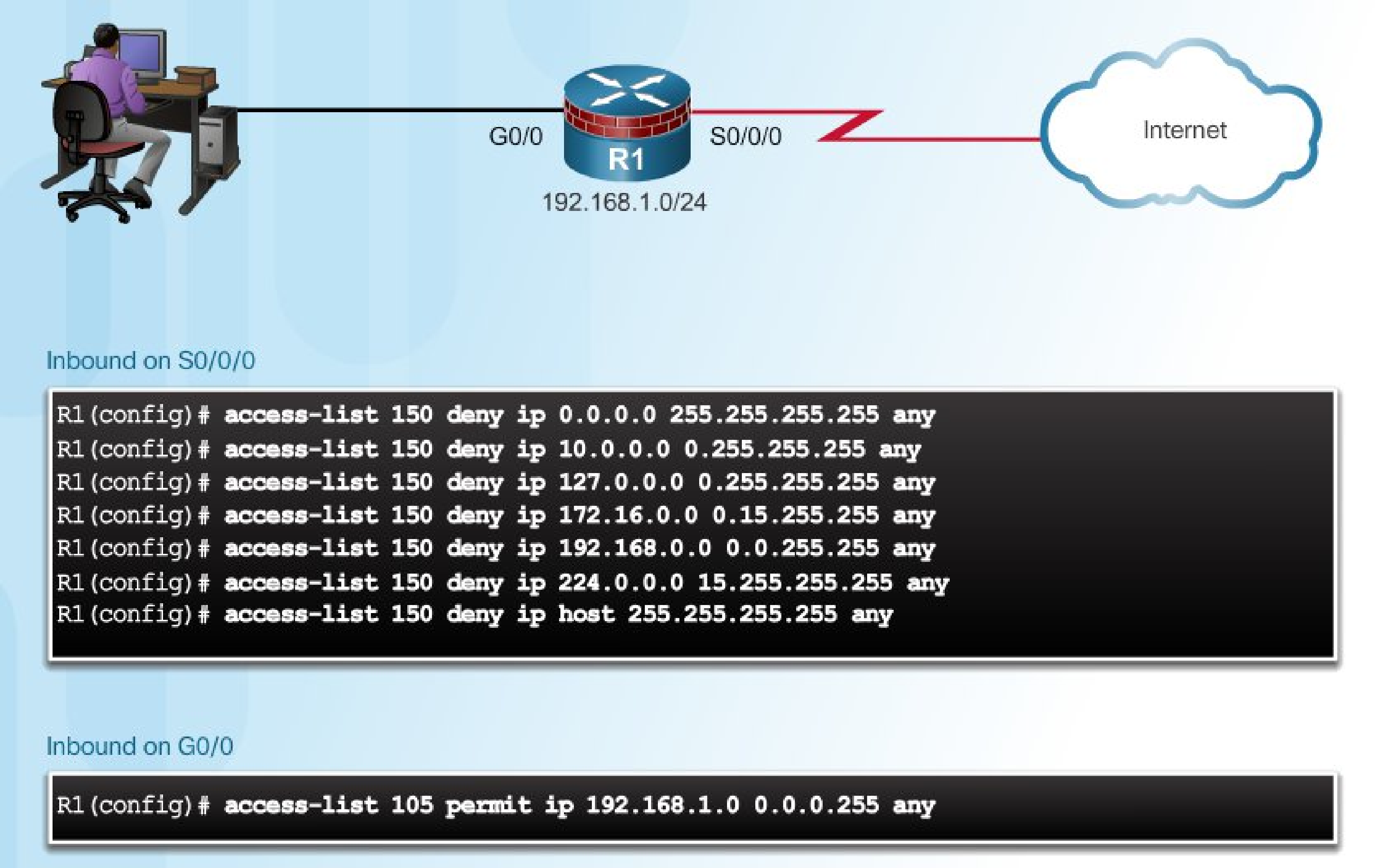


```
router# show access-lists
Standard IP access list 19
    10 permit 192.168.100.1
    25 permit 172.22.1.1
    20 permit 10.10.10.0, wildcard bits 0.0.0.255
    30 permit 201.101.110.0, wildcard bits 0.0.0.255
    40 deny any
```

Mitigating Attacks with ACLs

Mitigating Attacks with ACLs

Antispoofing with ACLs



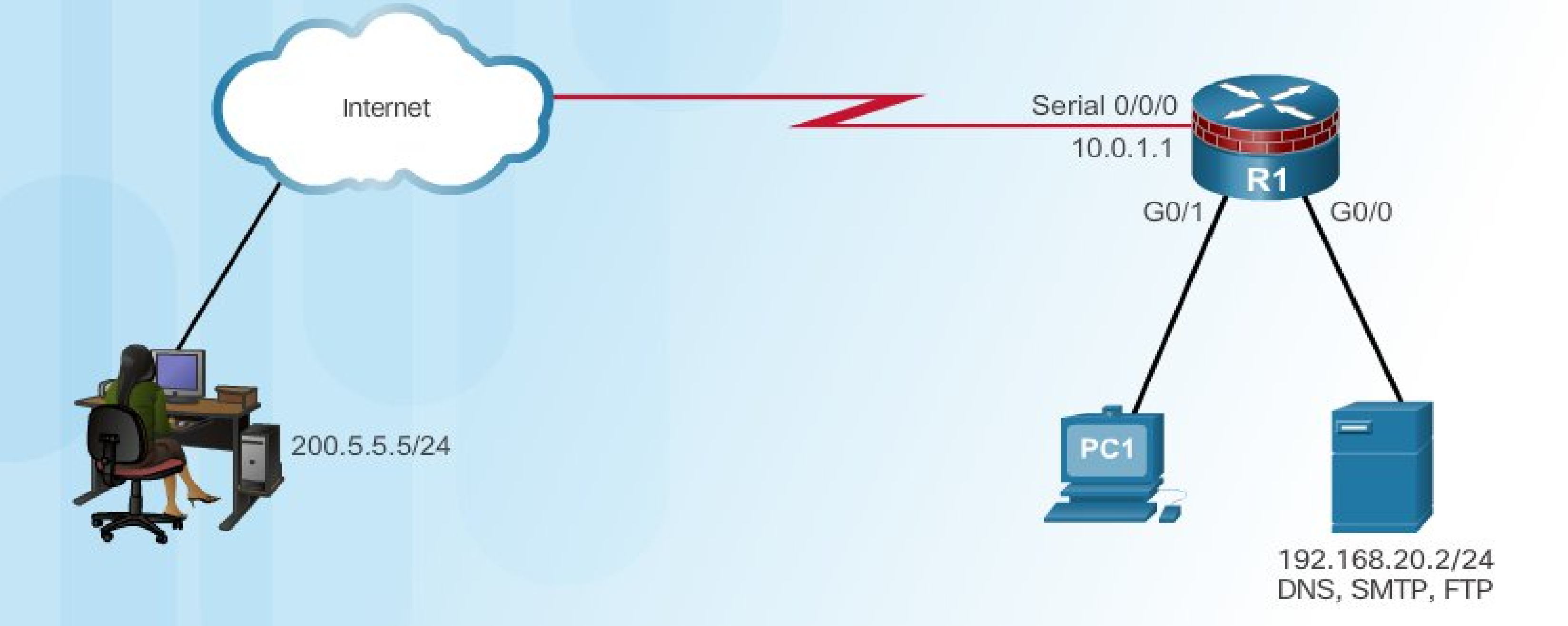
Inbound on S0/0/0

```
R1(config)# access-list 150 deny ip 0.0.0.0 255.255.255.255 any
R1(config)# access-list 150 deny ip 10.0.0.0 0.255.255.255 any
R1(config)# access-list 150 deny ip 127.0.0.0 0.255.255.255 any
R1(config)# access-list 150 deny ip 172.16.0.0 0.15.255.255 any
R1(config)# access-list 150 deny ip 192.168.0.0 0.0.255.255 any
R1(config)# access-list 150 deny ip 224.0.0.0 15.255.255.255 any
R1(config)# access-list 150 deny ip host 255.255.255.255 any
```

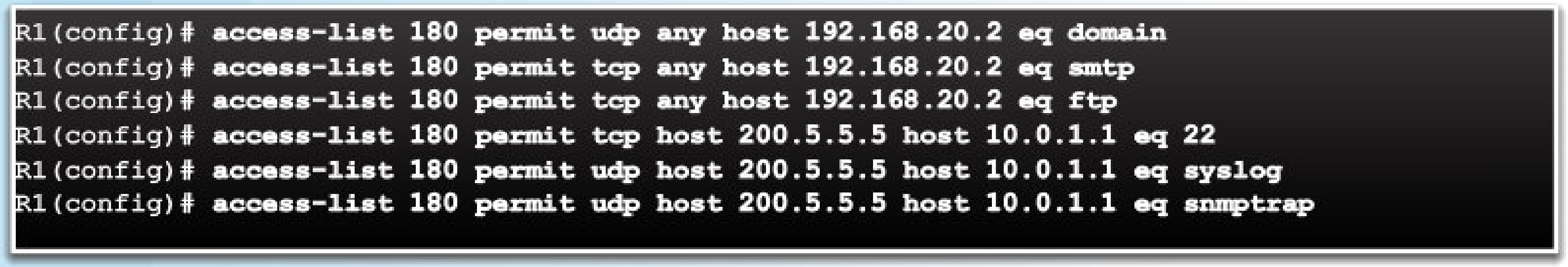
Inbound on G0/0

```
R1(config)# access-list 105 permit ip 192.168.1.0 0.0.0.255 any
```

Permitting Necessary Traffic through a Firewall

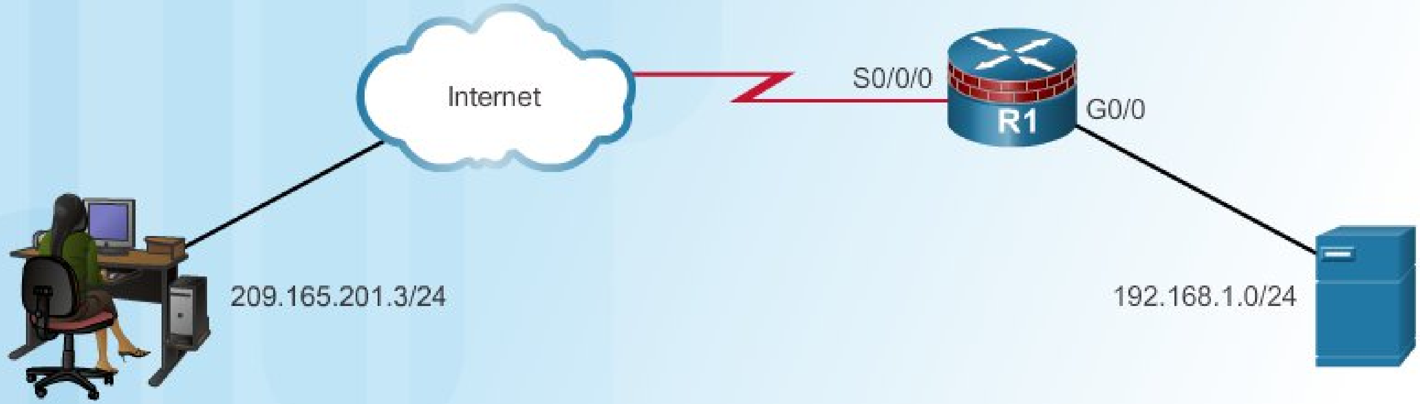


Inbound on Serial 0/0/0



```
R1(config)# access-list 180 permit udp any host 192.168.20.2 eq domain
R1(config)# access-list 180 permit tcp any host 192.168.20.2 eq smtp
R1(config)# access-list 180 permit tcp any host 192.168.20.2 eq ftp
R1(config)# access-list 180 permit tcp host 200.5.5.5 host 10.0.1.1 eq 22
R1(config)# access-list 180 permit udp host 200.5.5.5 host 10.0.1.1 eq syslog
R1(config)# access-list 180 permit udp host 200.5.5.5 host 10.0.1.1 eq snmptrap
```

Mitigating ICMP Abuse



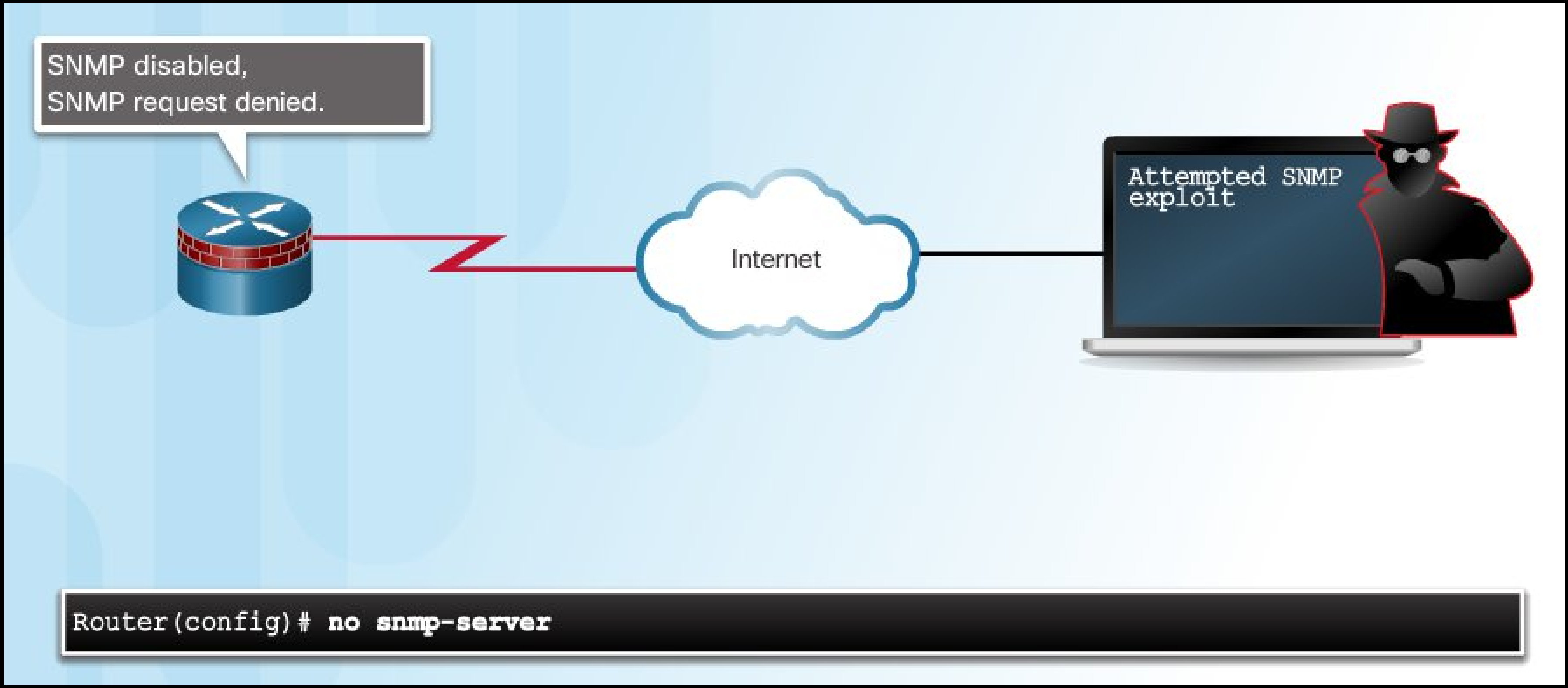
Inbound on S0/0/0

```
R1(config)# access-list 112 permit icmp any any echo-reply
R1(config)# access-list 112 permit icmp any any source-quench
R1(config)# access-list 112 permit icmp any any unreachable
R1(config)# access-list 112 deny icmp any any
R1(config)# access-list 112 permit ip any any
```

Inbound on G0/0

```
R1(config)# access-list 114 permit icmp 192.168.1.0 0.0.0.255 any echo
R1(config)# access-list 114 permit icmp 192.168.1.0 0.0.0.255 any parameter-problem
R1(config)# access-list 114 permit icmp 192.168.1.0 0.0.0.255 any packet-too-big
R1(config)# access-list 114 permit icmp 192.168.1.0 0.0.0.255 any source-quench
R1(config)# access-list 114 deny icmp any any
R1(config)# access-list 114 permit ip any any
```

Mitigating SNMP Exploits



SNMP disabled,
SNMP request denied.

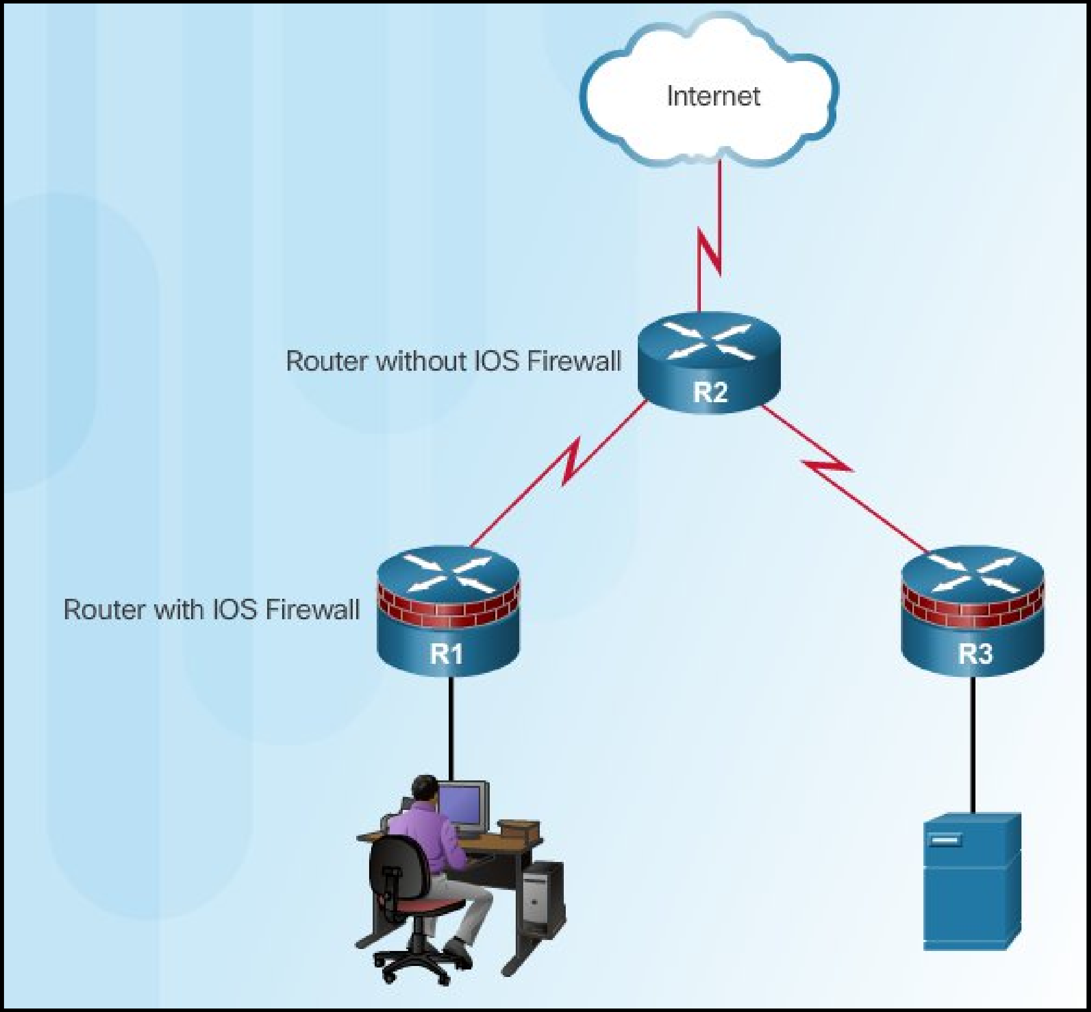
```
Router(config)# no snmp-server
```

Firewalls

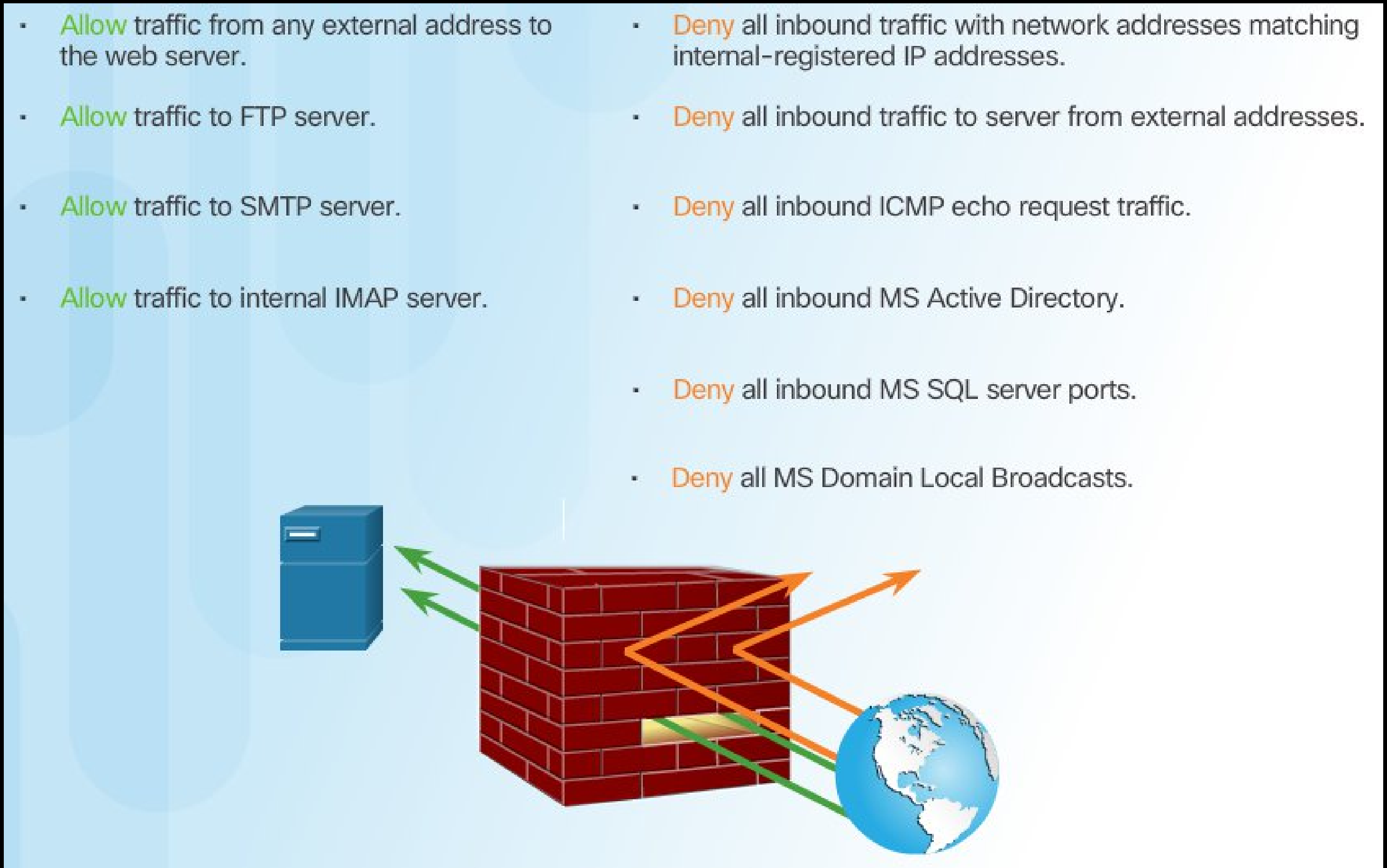
Defining Firewalls

All firewalls:

- Are resistant to attack
- Are the only transit point between networks because all traffic flows through the firewall
- Enforce the access control policy



Benefits and Limitations of Firewalls

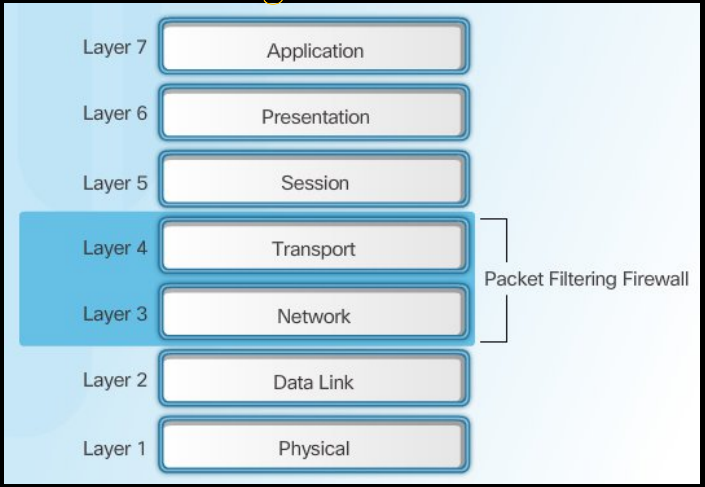


- Allow traffic from any external address to the web server.
- Allow traffic to FTP server.
- Allow traffic to SMTP server.
- Allow traffic to internal IMAP server.

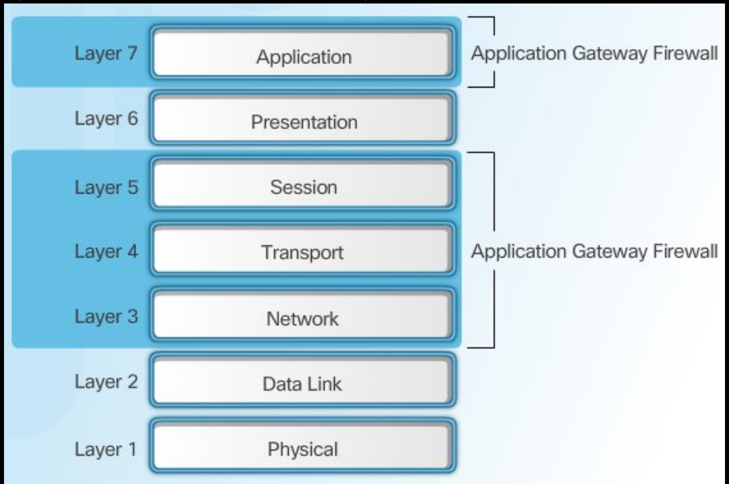
- Deny all inbound traffic with network addresses matching internal-registered IP addresses.
- Deny all inbound traffic to server from external addresses.
- Deny all inbound ICMP echo request traffic.
- Deny all inbound MS Active Directory.
- Deny all inbound MS SQL server ports.
- Deny all MS Domain Local Broadcasts.

Firewall Types

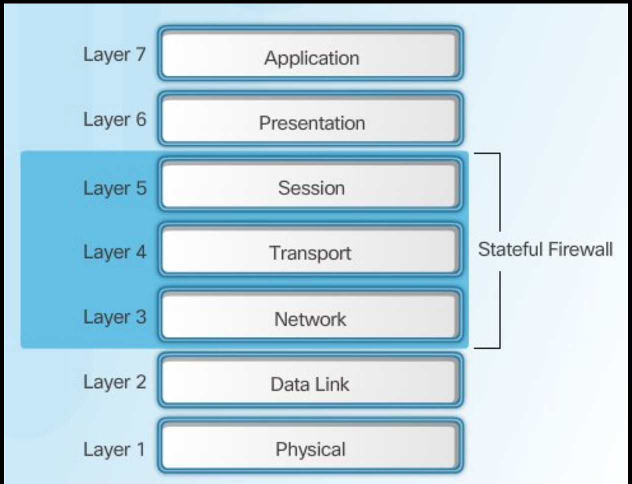
Packet Filtering Firewall



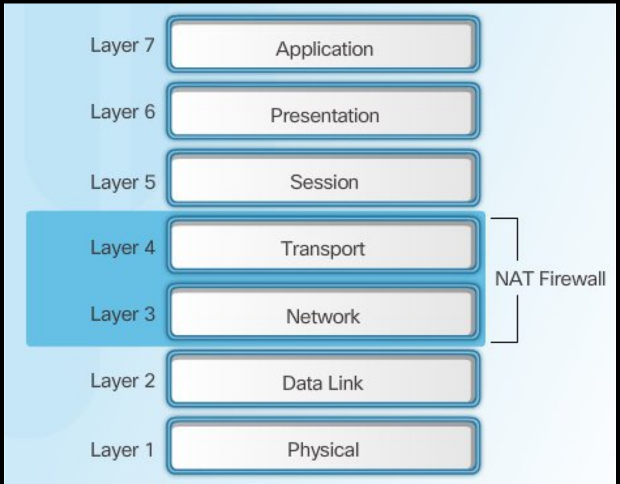
Application Gateway Firewall



Stateful Firewall



NAT Firewall



Stateful vs Stateless Firewall

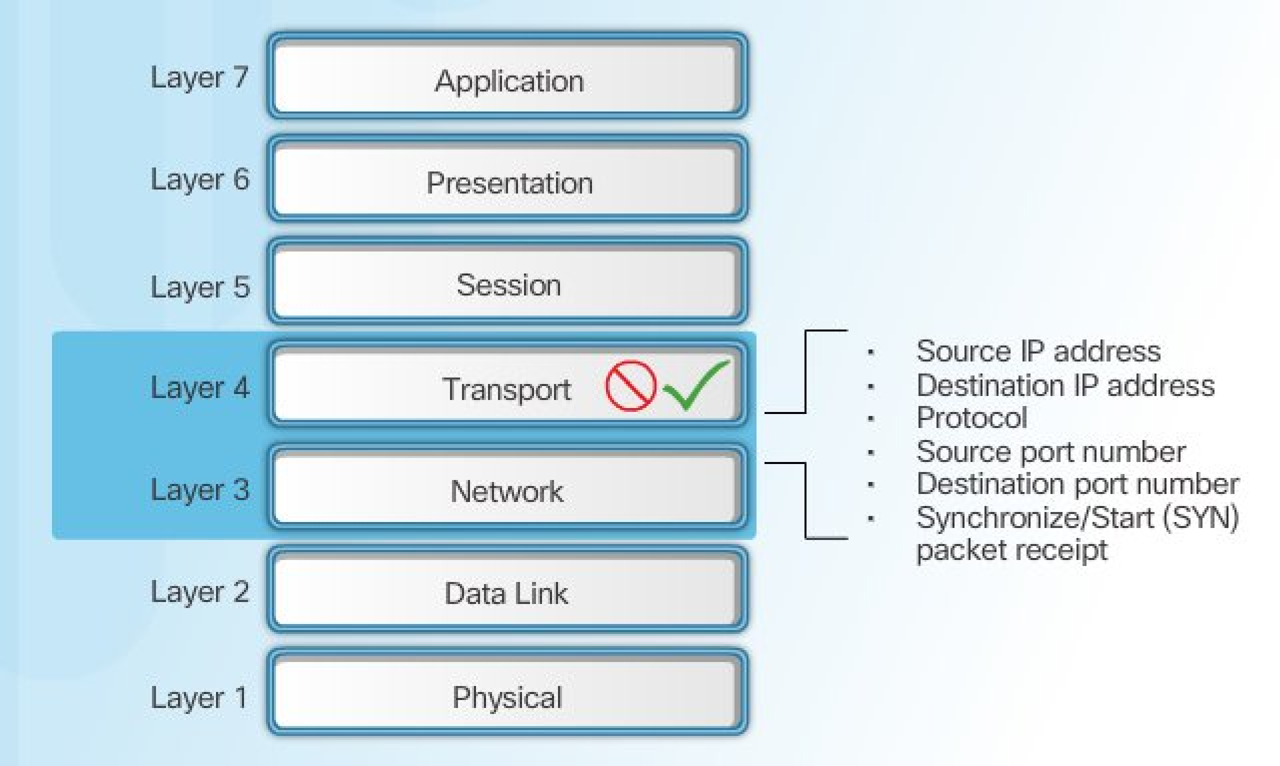
Stateless firewalls

(Packet Filtering) Stateless firewalls do not look at the state of connections but just at the packets themselves. An example of a packet filtering firewall is the Extended Access Control Lists on Cisco IOS Routers.

Stateful firewall:

Aware of the connections that pass through it. It adds and maintains information about a user's connections in a state table, referred to as a connection table. It then uses this connection table to implement the security policies for users connections. An example of the stateful firewall is PIX, ASA, Checkpoint.

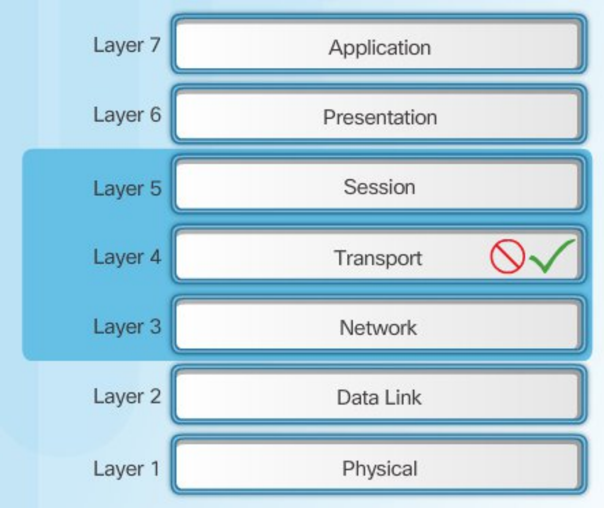
Packet Filtering Firewall Benefits & Limitations



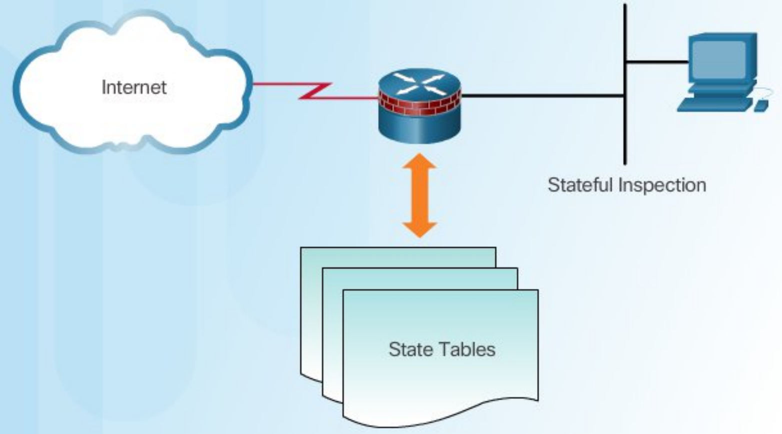
- Source IP address
- Destination IP address
- Protocol
- Source port number
- Destination port number
- Synchronize/Start (SYN) packet receipt

Stateful Firewalls

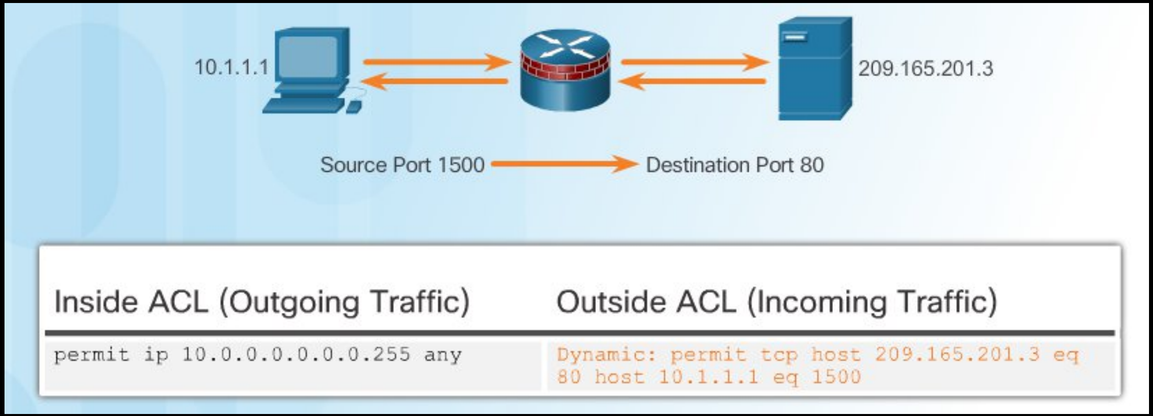
Stateful Firewalls



State Tables

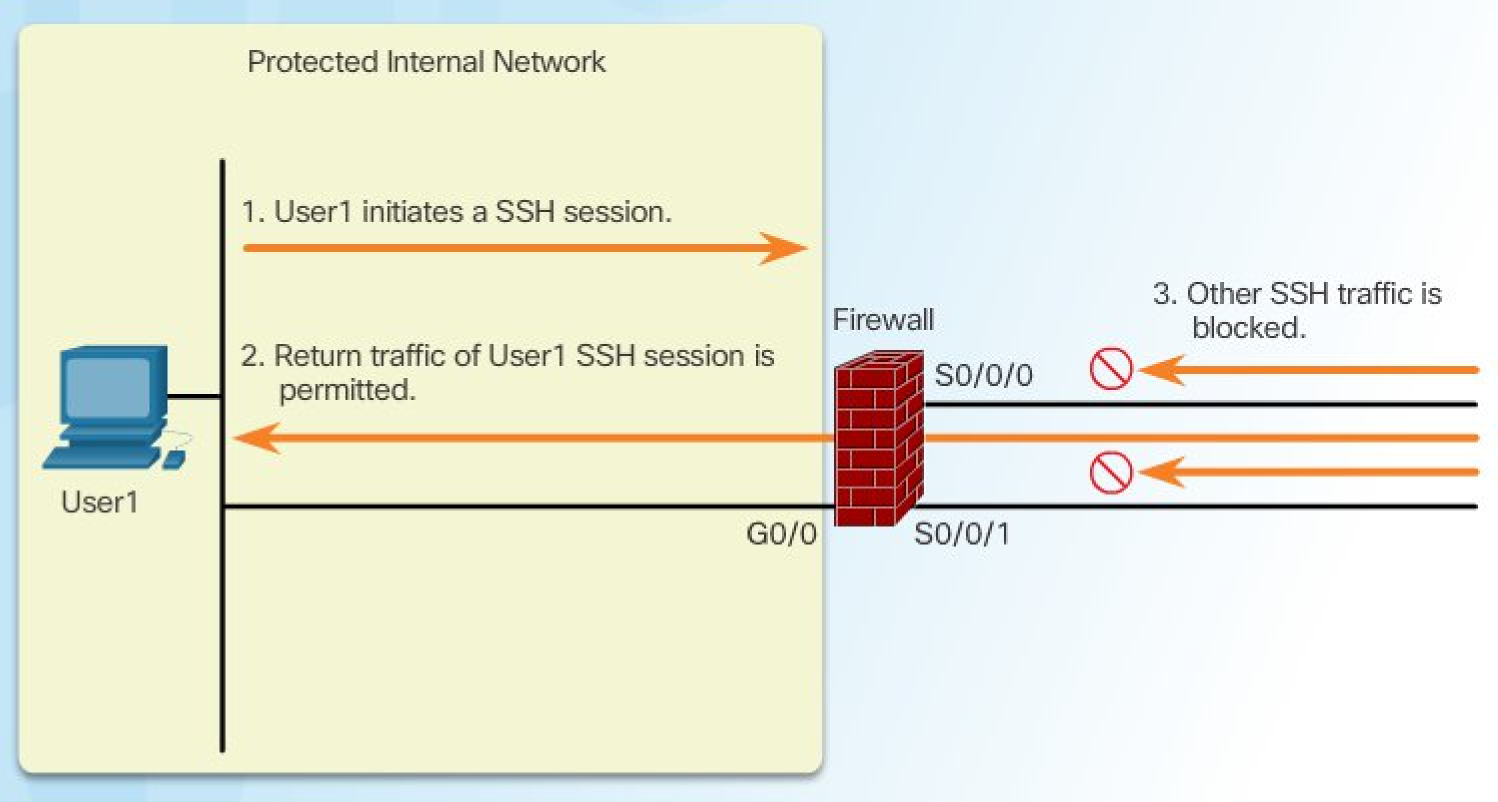


Stateful Firewall Operation

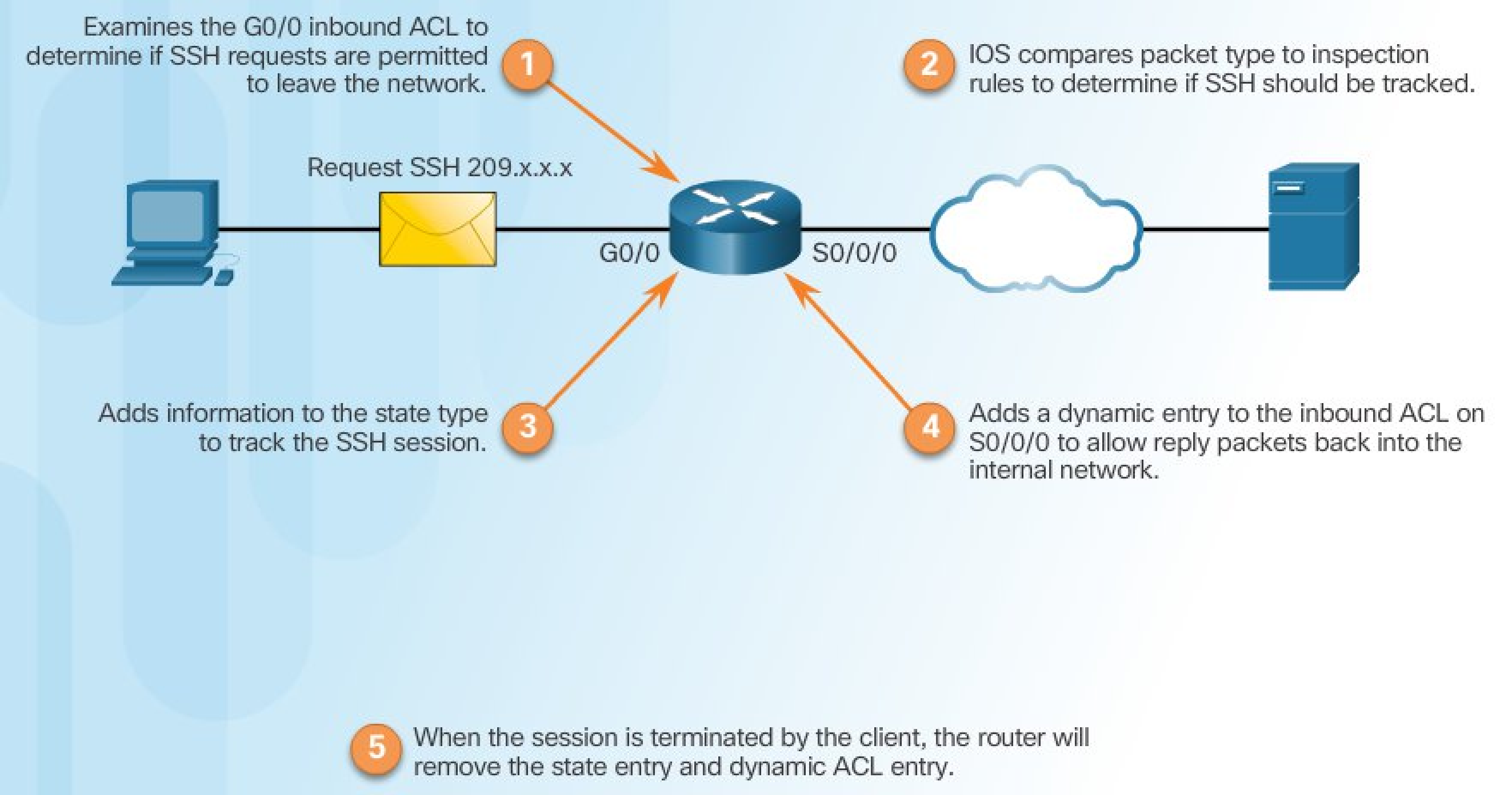


Inside ACL (Outgoing Traffic)	Outside ACL (Incoming Traffic)
`permit ip 10.0.0.0.0.0.0.255 any`	`Dynamic: permit tcp host 209.165.201.3 eq 80 host 10.1.1.1 eq 1500`

Classic Firewall


Protected Internal Network
1. User1 initiates a SSH session.
2. Return traffic of User1 SSH session is permitted.
User1
Firewall
G0/0
S0/0/0
S0/0/1
3. Other SSH traffic is blocked.

Classic Firewall Operation



1. Examines the G0/0 inbound ACL to determine if SSH requests are permitted to leave the network.

2. IOS compares packet type to inspection rules to determine if SSH should be tracked.

3. Adds information to the state type to track the SSH session.

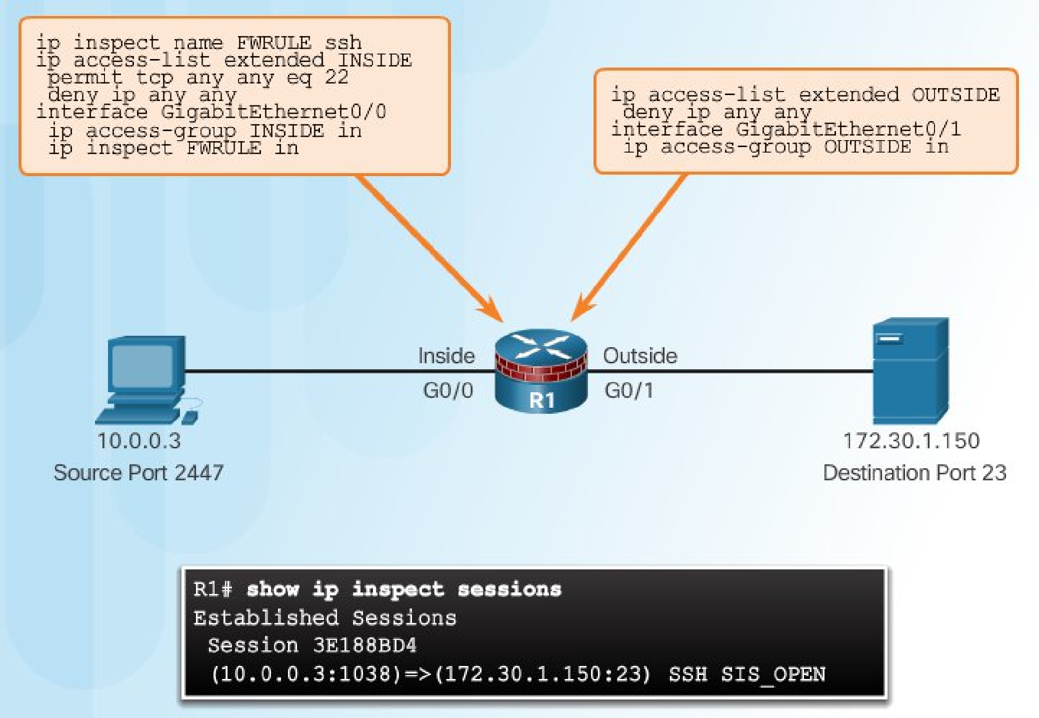
4. Adds a dynamic entry to the inbound ACL on S0/0/0 to allow reply packets back into the internal network.

5. When the session is terminated by the client, the router will remove the state entry and dynamic ACL entry.

Classic Firewall Configuration

1. Choose the internal and external interfaces.
2. Configure ACLs for each interface.
3. Define inspection rules.
4. Apply an inspection rule to an interface.

Inspection Rules



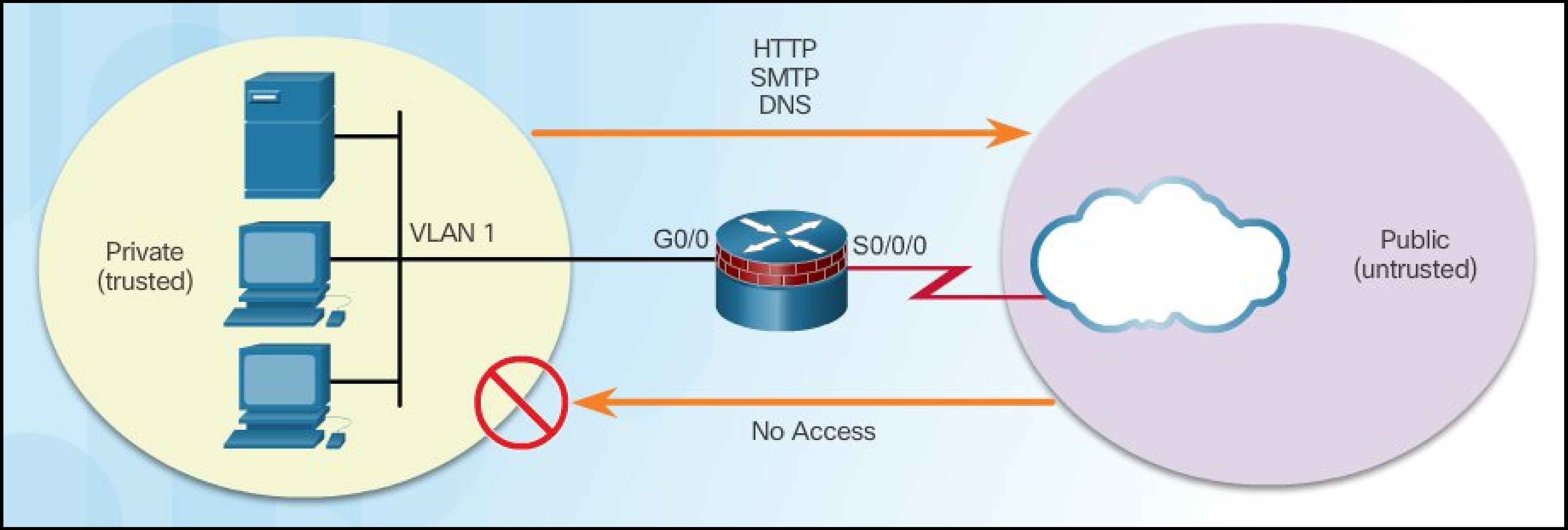
```
ip inspect name FWRULE ssh
ip access-list extended INSIDE
 permit tcp any any eq 22
 deny ip any any
interface GigabitEthernet0/0
 ip access-group INSIDE in
 ip inspect FWRULE in
```

```
ip access-list extended OUTSIDE
 deny ip any any
interface GigabitEthernet0/1
 ip access-group OUTSIDE in
```

```
R1# show ip inspect sessions
Established Sessions
 Session 3E188BD4
 (10.0.0.3:1038)=>(172.30.1.150:23) SSH SIS_OPEN
```

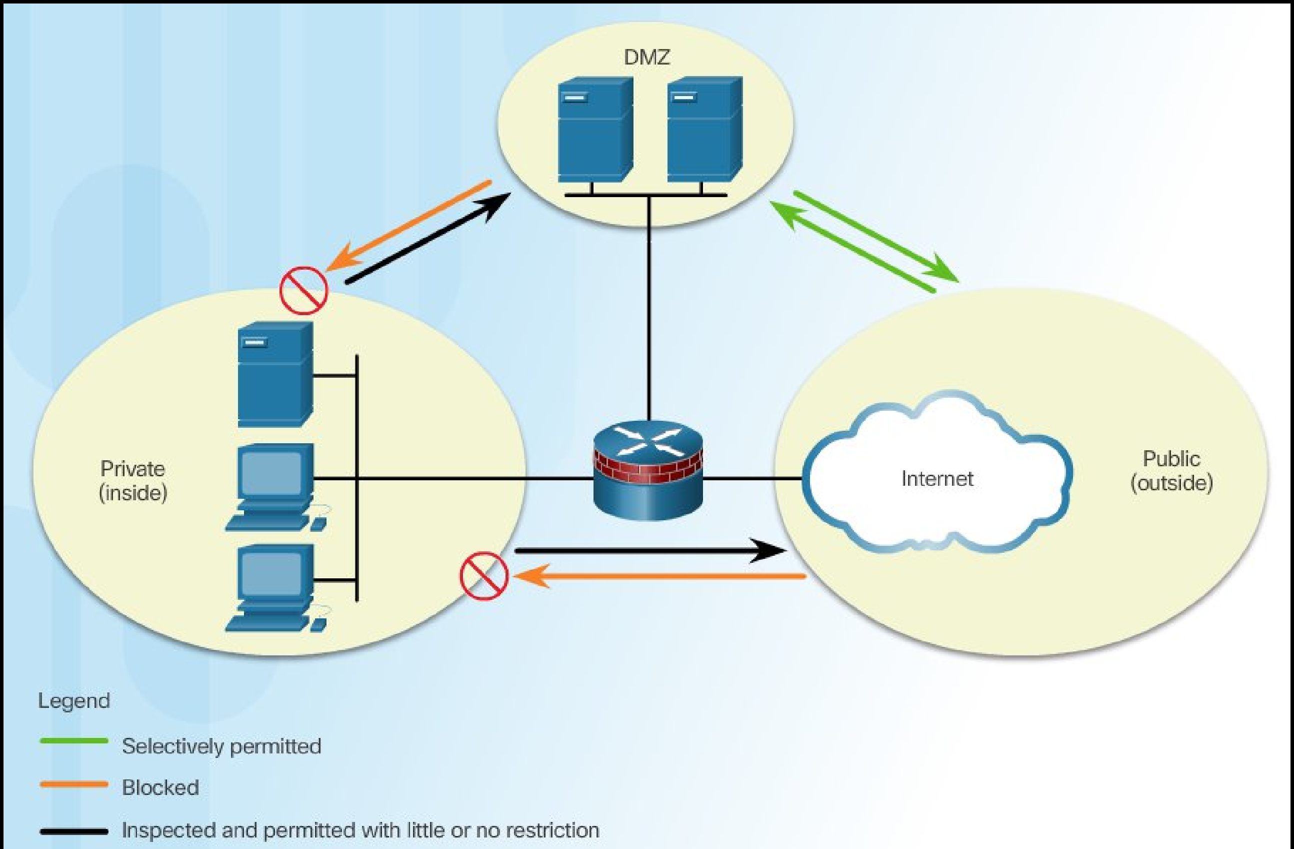
Firewalls in Network Design

Inside and Outside Networks



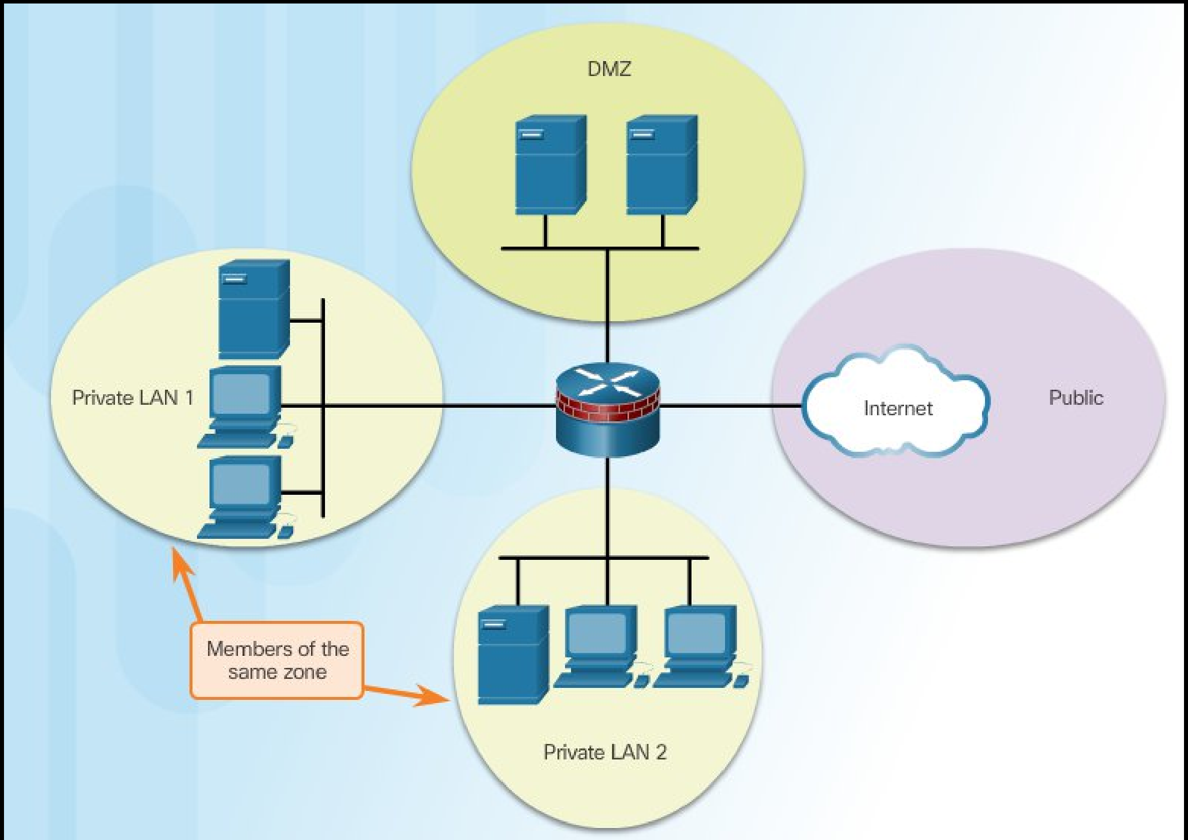
Firewalls in Network Design

Demilitarized Zones



Firewalls in Network Design

Zone-Based Policy Firewalls



Firewall Best Practice

- Position firewalls at security boundaries
- It is unwise to reply exclusively on a firewall for security
- Deny all traffic by default. Permit only services that are needed
- Ensure that physical access to the firewall is controlled
- Monitor firewall logs
- Practice change management for firewall configuration changes
- Remember that firewalls primarily protect from technical attacks originating from the outside

Comparing Approaches

Cloud-Native Firewalls

Design Philosophy

- Firewalling **as-a-service**.
- Integrated into cloud infrastructure.
- Highly scalable and programmable.

Key Features

- Identity and role-based access control (IAM integration)
- Policy-based rules applied at VMs, subnets, or regions
- Autoscaling and high availability
- API-driven configuration (great for DevOps)

Comparing Approaches

Next-Generation Firewalls

Design Philosophy

- Goes beyond port/protocol filtering.
- Inspects traffic at **Layer 7 (Application Layer)**.
- Integrated threat detection and prevention.

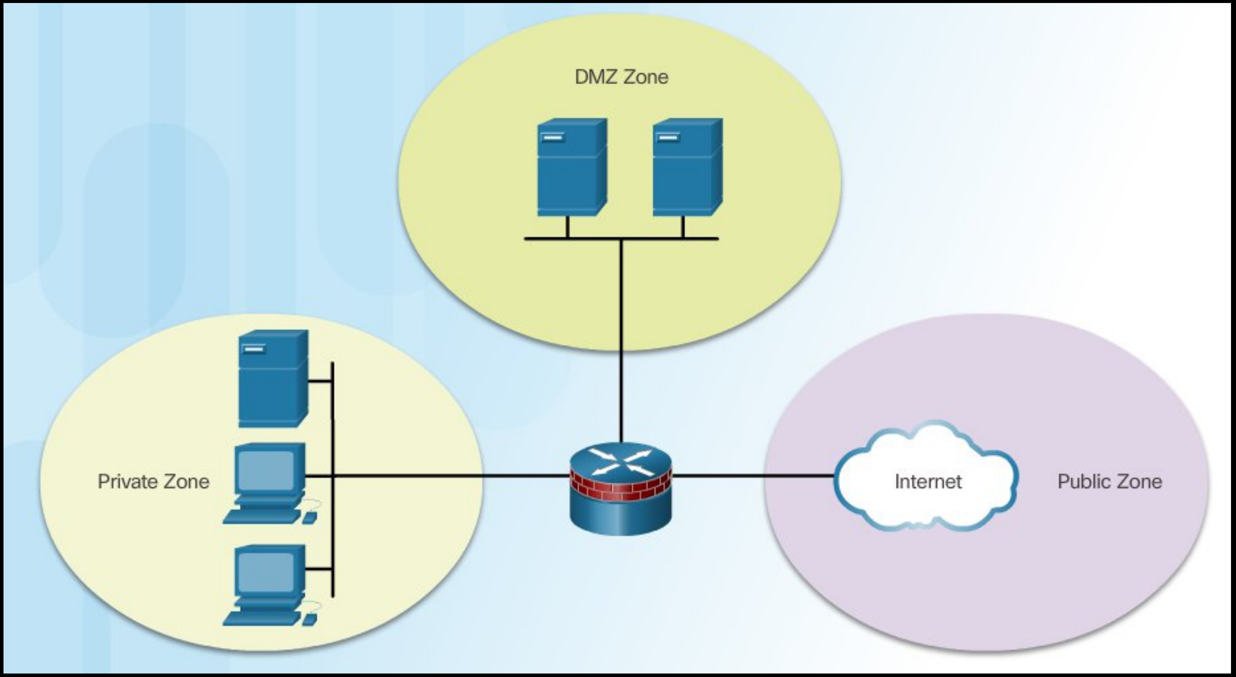
Key Features

- Deep Packet Inspection (DPI)
- Application awareness (can allow Facebook chat but block games, for example)
- Integrated intrusion prevention system (IPS)
- Malware sandboxing
- User identity-based rules (e.g., integrate with Active Directory)
- SSL/TLS inspection
- Threat intelligence feeds and auto-updates

Zone-based Policy Firewalls

Benefits of ZPF

- Not dependent on ACLs
- Router security posture is to block unless explicitly allowed
- Policies are easy to read and troubleshoot with C3PL
- One policy affects any given traffic, instead of needing multiple ACLs and inspection actions



ZPF Design

Common designs include:

- LAN-to-Internet
- Firewalls between public servers
- Redundant firewalls
- Complex firewalls

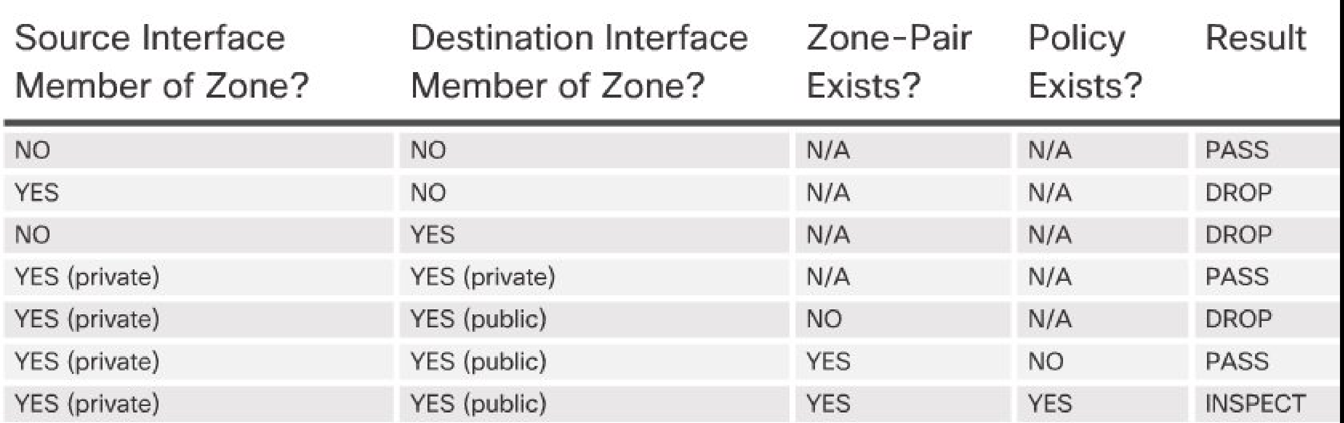
Design steps:

1. Determine the zones
2. Establish policies between zones
3. Design the physical infrastructure
4. Identify subsets within zones and merge traffic requirements

ZPF Actions

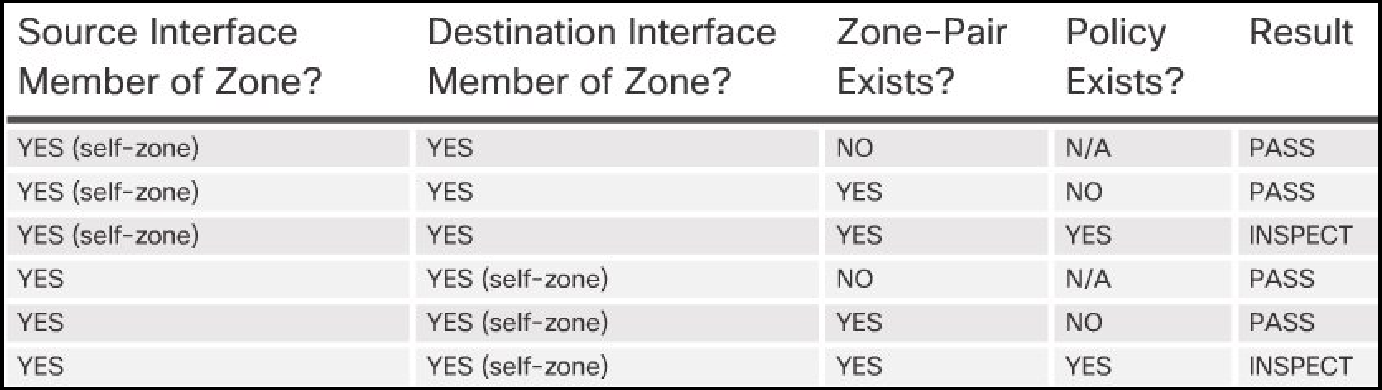
- **Inspect** - Configures Cisco IOS stateful packet inspections.
- **Drop** - Analogous to a deny statement in an ACL. A log option is available to log the rejected packets.
- **Pass** - Analogous to a permit statement in an ACL. The pass action does not track the state of connections or sessions within the traffic.

Rules for Transit Traffic



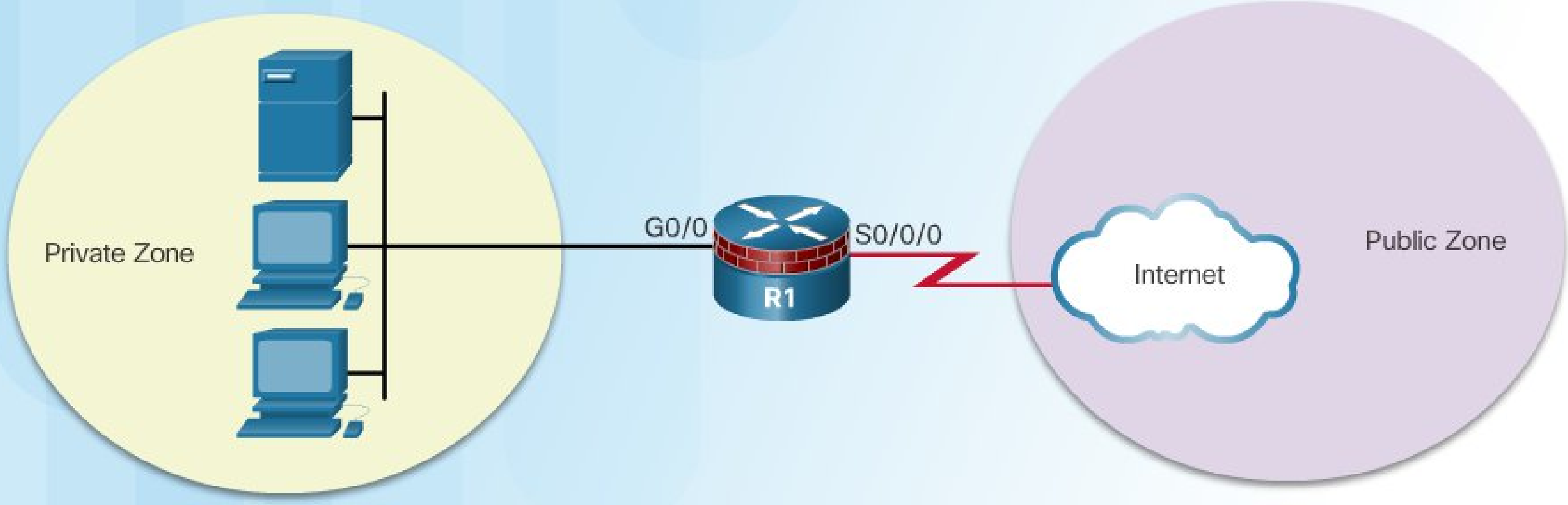
Source Interface Member of Zone?	Destination Interface Member of Zone?	Zone-Pair Exists?	Policy Exists?	Result
NO	NO	N/A	N/A	PASS
YES	NO	N/A	N/A	DROP
NO	YES	N/A	N/A	DROP
YES (private)	YES (private)	N/A	N/A	PASS
YES (private)	YES (public)	NO	N/A	DROP
YES (private)	YES (public)	YES	NO	PASS
YES (private)	YES (public)	YES	YES	INSPECT

Rules for Traffic to the Self Zone



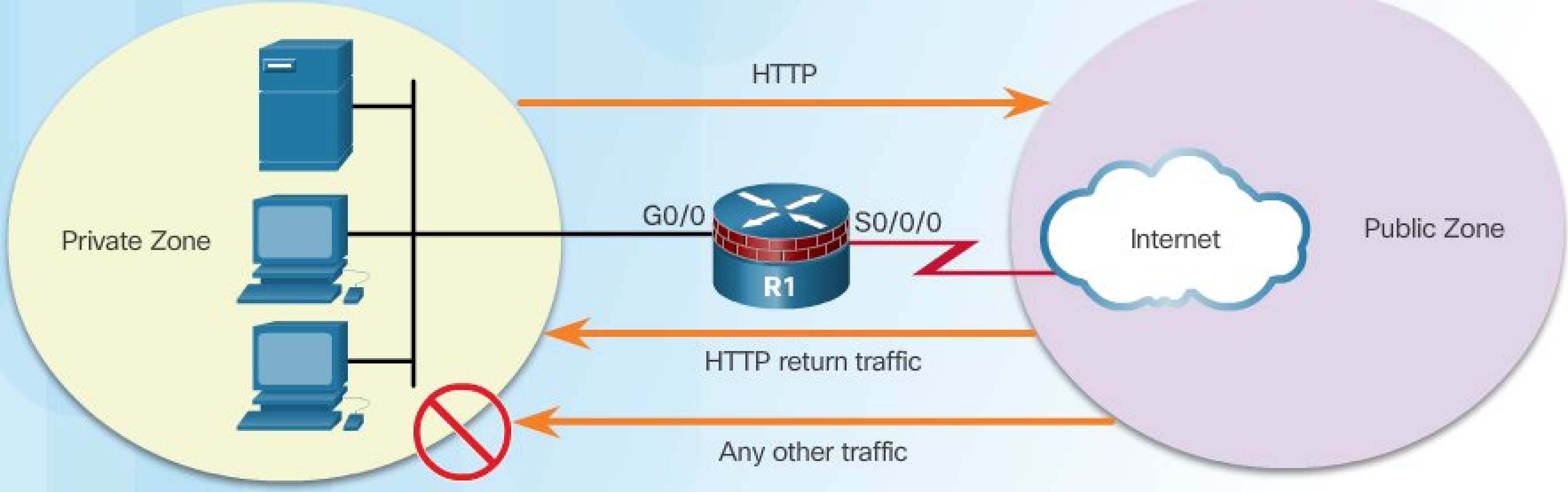
Source Interface Member of Zone?	Destination Interface Member of Zone?	Zone-Pair Exists?	Policy Exists?	Result
YES (self-zone)	YES	NO	N/A	PASS
YES (self-zone)	YES	YES	NO	PASS
YES (self-zone)	YES	YES	YES	INSPECT
YES	YES (self-zone)	NO	N/A	PASS
YES	YES (self-zone)	YES	NO	PASS
YES	YES (self-zone)	YES	YES	INSPECT

Configure ZPF



Step 1: Create the zones.
Step 2: Identify traffic with a class-map.
Step 3: Define an action with a policy-map.
Step 4: Identify a zone pair and match it to a policy-map.
Step 5: Assign zones to the appropriate interfaces.

Step 1: Create Zones



Syntax

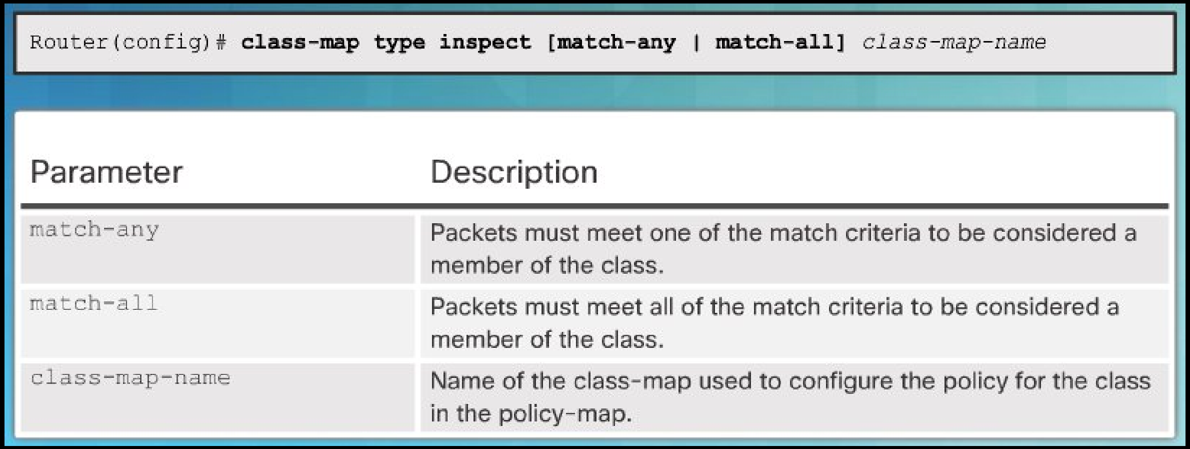
```
Router(config)# zone security zone-name
```

Example

```
R1(config)# zone security PRIVATE
R1(config-sec-zone)# exit
R1(config)# zone security PUBLIC
```

Step 2: Identify Traffic

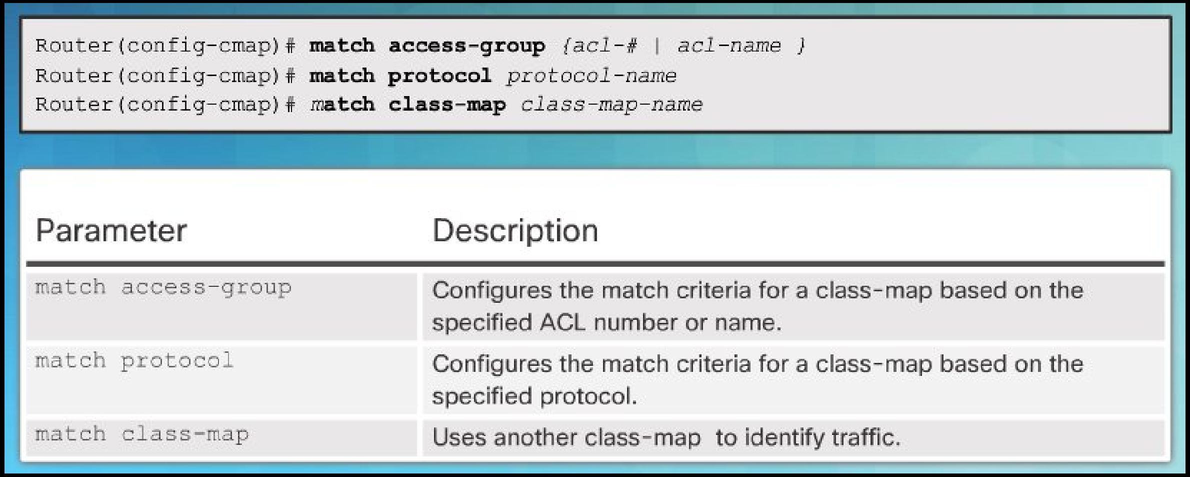
Command Syntax for `class-map`



```
Router(config)# class-map type inspect [match-any | match-all] class-map-name
```

Parameter	Description
`match-any`	Packets must meet one of the match criteria to be considered a member of the class.
`match-all`	Packets must meet all of the match criteria to be considered a member of the class.
`class-map-name`	Name of the class-map used to configure the policy for the class in the policy-map.

Sub-Configuration Command Syntax for `class-map`

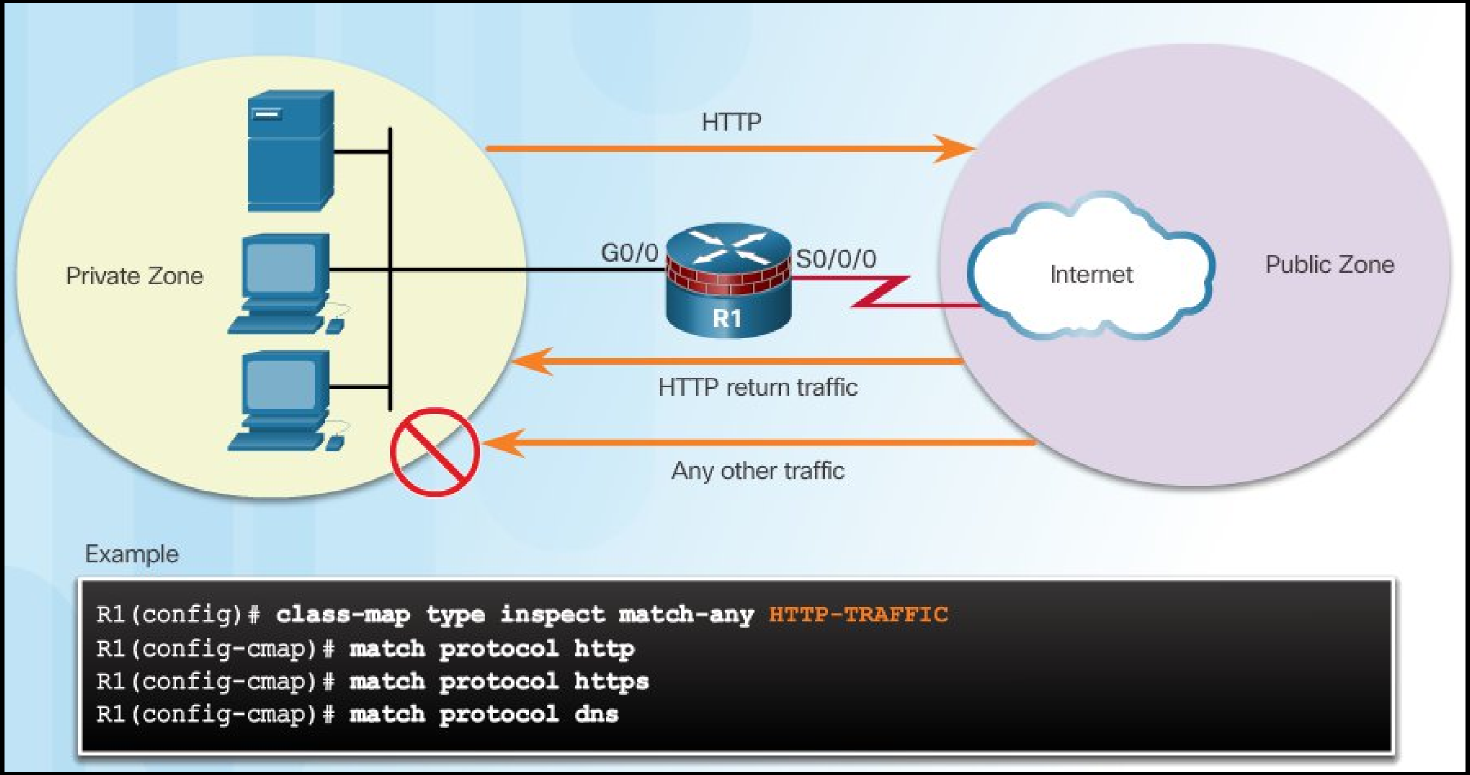


```
Router(config-cmap)# match access-group {acl-# | acl-name }
Router(config-cmap)# match protocol protocol-name
Router(config-cmap)# match class-map class-map-name
```

Parameter	Description
`match access-group`	Configures the match criteria for a class-map based on the specified ACL number or name.
`match protocol`	Configures the match criteria for a class-map based on the specified protocol.
`match class-map`	Uses another class-map to identify traffic.

Step 2: Identify Traffic (Cont.)

Example `class-map` Configuration

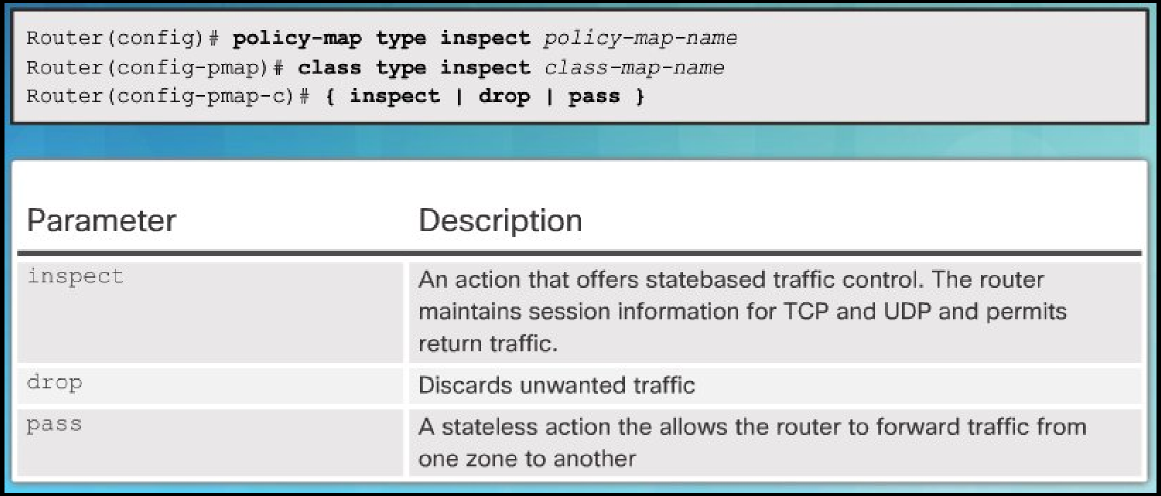


Example

```
R1(config)# class-map type inspect match-any HTTP-TRAFFIC
R1(config-cmap)# match protocol http
R1(config-cmap)# match protocol https
R1(config-cmap)# match protocol dns
```

Step 3: Define an Action

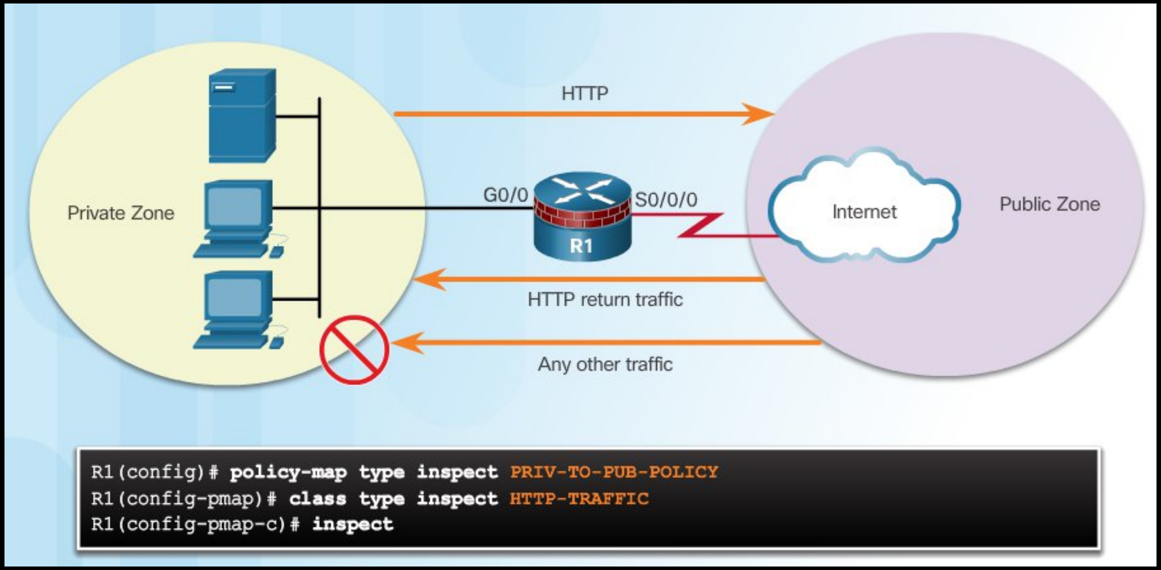
Command Syntax for `policy-map`



```
Router(config)# policy-map type inspect policy-map-name
Router(config-pmap)# class type inspect class-map-name
Router(config-pmap-c)# { inspect | drop | pass }
```

Parameter	Description
`inspect`	An action that offers statebased traffic control. The router maintains session information for TCP and UDP and permits return traffic.
`drop`	Discards unwanted traffic
`pass`	A stateless action the allows the router to forward traffic from one zone to another

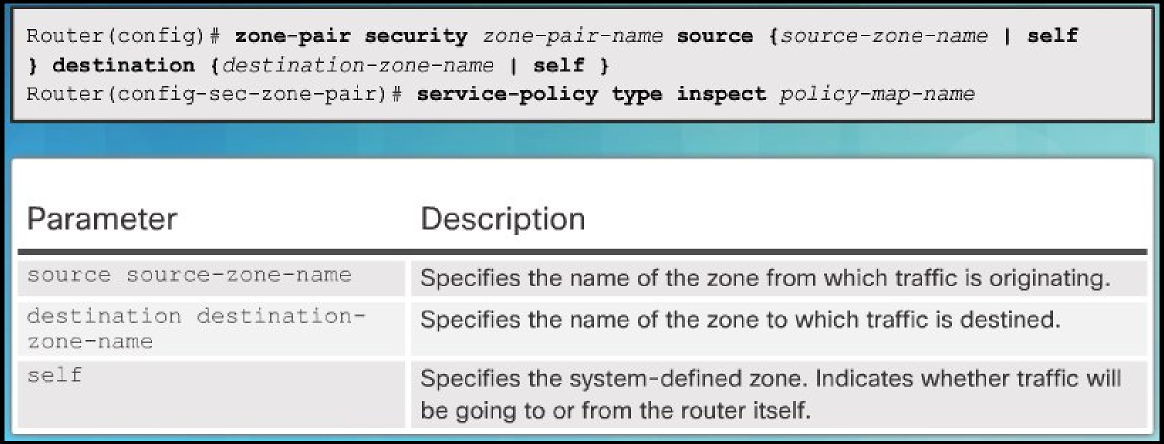
Example `policy-map` Configuration



```
R1(config)# policy-map type inspect PRIV-TO-PUB-POLICY
R1(config-pmap)# class type inspect HTTP-TRAFFIC
R1(config-pmap-c)# inspect
```

Step 4: Identify a Zone-Pair and Match to a Policy

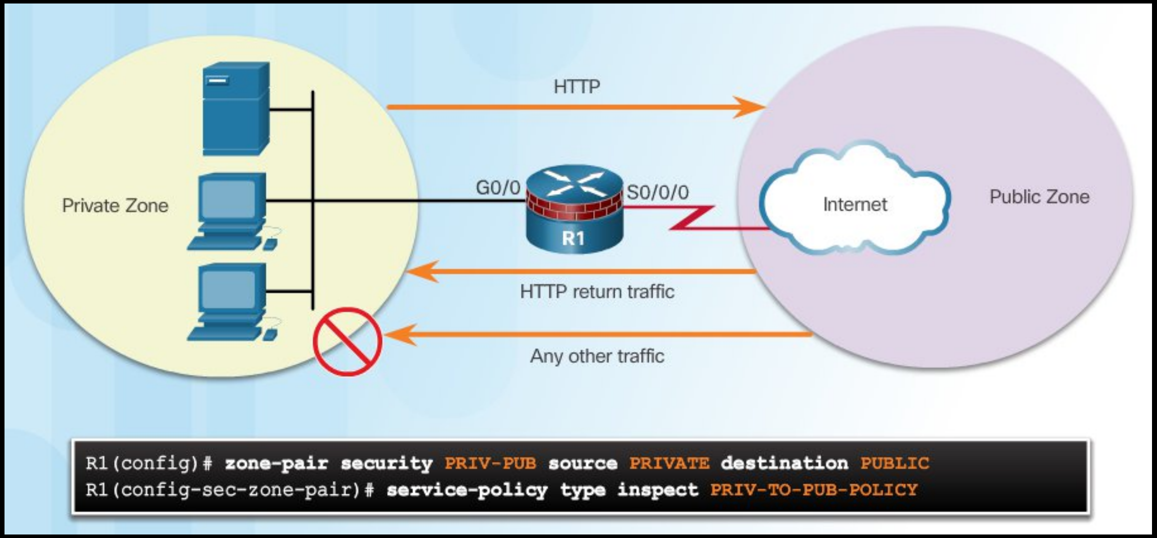
Command Syntax for `zone-pair` and `service-policy`



```
Router(config)# zone-pair security zone-pair-name source {source-zone-name | self
} destination {destination-zone-name | self }
Router(config-sec-zone-pair)# service-policy type inspect policy-map-name
```

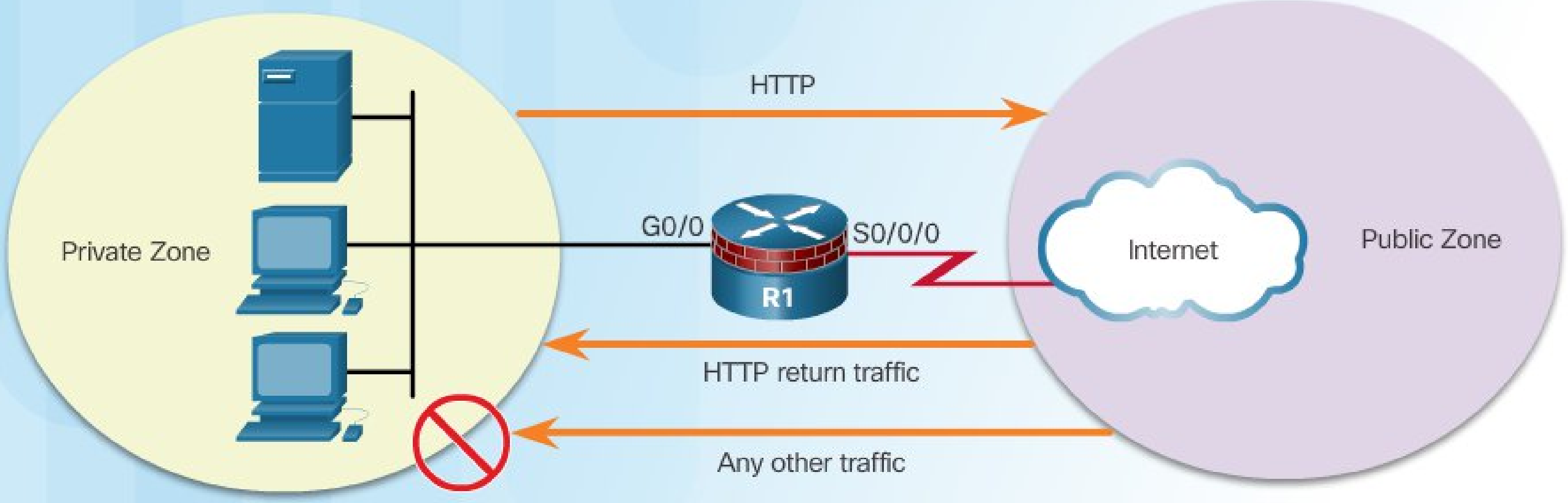
Parameter	Description
`source source-zone-name`	Specifies the name of the zone from which traffic is originating.
`destination destination-zone-name`	Specifies the name of the zone to which traffic is destined.
`self`	Specifies the system-defined zone. Indicates whether traffic will be going to or from the router itself.

Example `service-policy` Configuration



```
R1(config)# zone-pair security PRIV-PUB source PRIVATE destination PUBLIC
R1(config-sec-zone-pair)# service-policy type inspect PRIV-TO-PUB-POLICY
```

Step 5: Assign Zones to Interfaces



Syntax

```
Router(config-if)# zone-member security zone-name
```

Example

```
R1(config)# interface GigabitEthernet 0/0
R1(config-if)# zone-member security PRIVATE
R1(config-if)# interface Serial 0/0/0
R1(config-if)# zone-member security PUBLIC
```

Verify a ZPF Configuration

Verification commands:

- show run | begin class-map
- show policy-map type inspect zone-pair sessions
- show class-map type inspect
- show zone security
- show zone-pair security
- show policy-map type inspect

ZPF Configuration Considerations

- No filtering is applied for intra-zone traffic
- Only one zone is allowed per interface.
- No Classic Firewall and ZPF configuration on same interface.
- If only one zone member is assigned, all traffic is dropped.
- Only explicitly allowed traffic is forwarded between zones.
- Traffic to the self zone is not filtered.

More Advanced Testing

To further test your configured firewall think about any ‘attacks’ or ‘testing’ you have learnt in other modules and see if they can be performed through your firewall. A few examples (you will obviously need to change for your topology):

Port Scanning (Reconnaissance)

```
nmap -sS 192.168.3.3
nmap -sU 192.168.3.3
```

Spoofed Packet Injection (IP Spoofing Attempt)

```
hping3 -a 192.168.3.3 -c 5 -1 192.168.1.3
```

Malformed Packet Injection (Protocol Abuse)

```
nmap -sX 192.168.3.3
hping3 -FPU -p 80 -c 5 192.168.3.3
```

Application-Layer Attacks (HTTP Methods, Slowloris)

```
curl -X TRACE http://10.2.2.2/
python3 slowloris.py -p 80 -s 300
```

DNS Tunneling or Abuse

```
dig txt longname.example.com @8.8.8.8
```