CCS6224 Network Security

Lecture 3
Firewall Technologies

What is a Firewall?

- a **choke point** of control and monitoring
- interconnects networks with differing trust
- imposes restrictions on network services
  - only authorized traffic is allowed
- auditing and controlling access
  - can implement alarms for abnormal behavior
- is itself immune to penetration
- provides **perimeter defence**

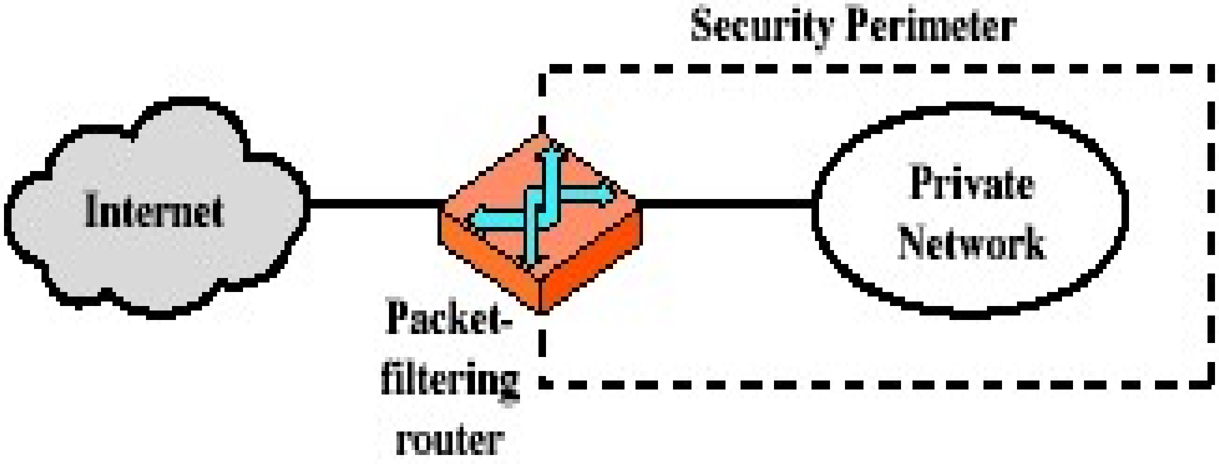
Benefits of Firewall

- Prevents exposing sensitive hosts and applications to untrusted users
- Firewalls prevent malicious data from being sent to servers and clients
- Properly configured firewalls make security policy enforcement simple, scalable and robust
- Firewall reduces the complexity of security management by offloading most of the network access control to a couple of points in the network

Firewall Limitations

- cannot protect from attacks bypassing it
  - sneaker net, utility modems, trusted organisations, trusted services (eg SSL/SSH)
- cannot protect against internal threats
  - disgruntled employee
- cannot protect against transfer of all virus infected programs or files
  - because of huge range of O/S & file types

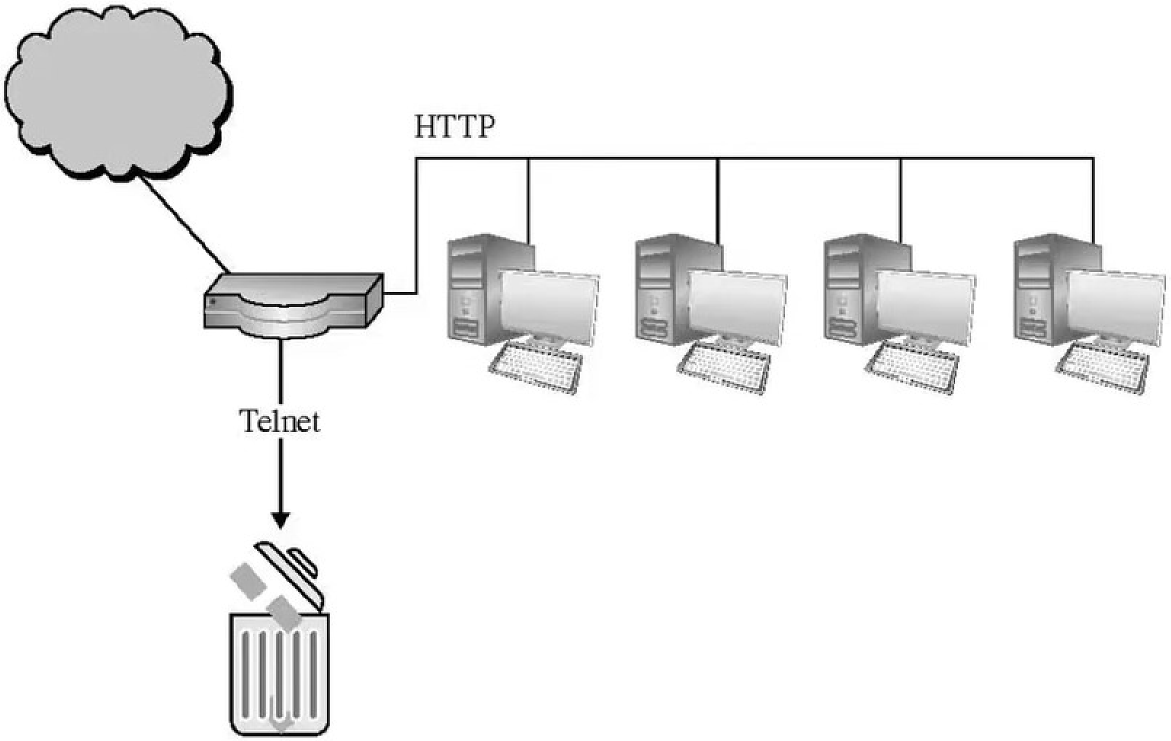
Packet Filters (Firewall)



(a) Packet-filtering router

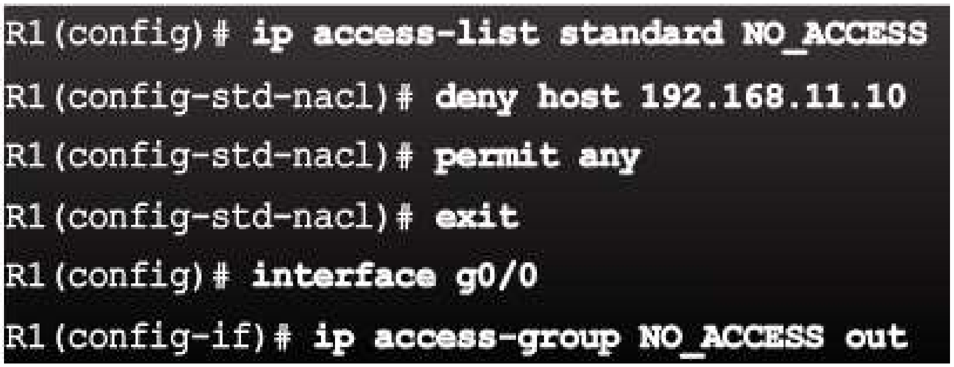
Packet Filters

- simplest of components
- foundation of any firewall system
- examine the source and/or destination IP address for each packet
- Examine the type of transport protocol for each packet (HTTP,FTP, Telnet) – port filtering
- hence restrict access to services (ports)
- possible default policies
  - Default = discard: that not expressly permitted is prohibited
  - Default = forward: that not expressly prohibited is permitted


HTTP
Telnet

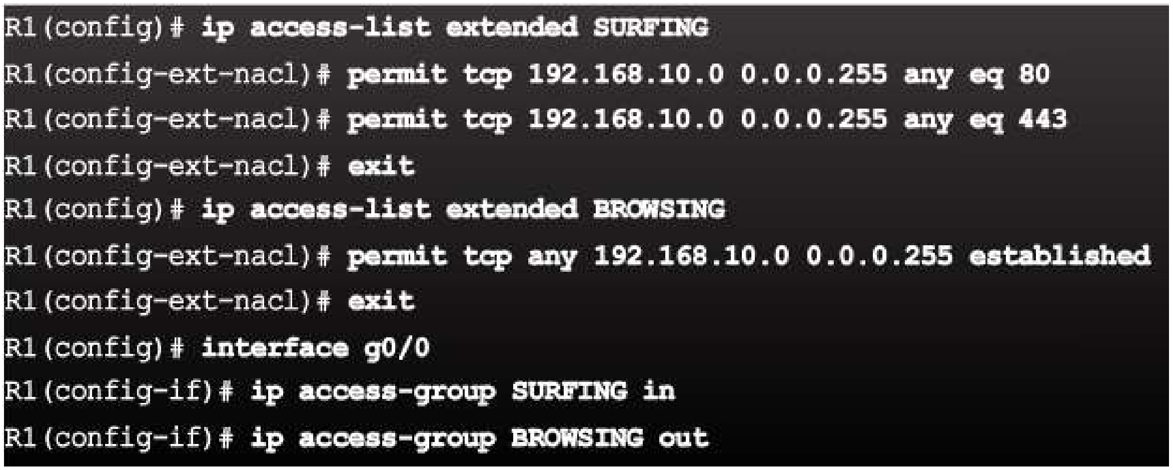
Packet Filters

Example 1
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
```

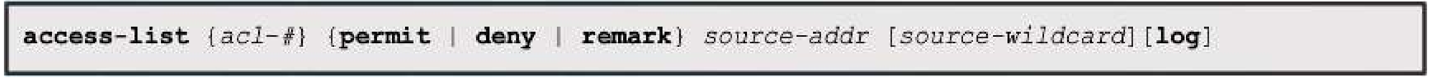
Example 2
Extended
ACL



```
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
```

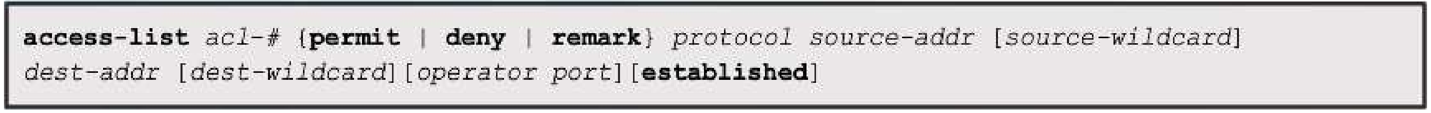
Configuring Numbered and Named ACLs

Standard Numbered ACL Syntax



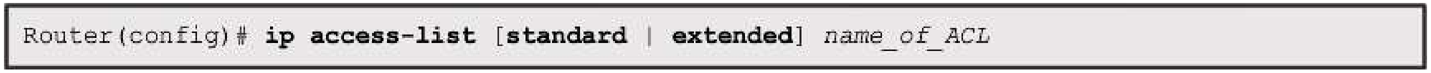
```
access-list {acl-#} {permit | deny | remark} source-addr [source-wildcard][log]
```

Extended Numbered ACL Syntax



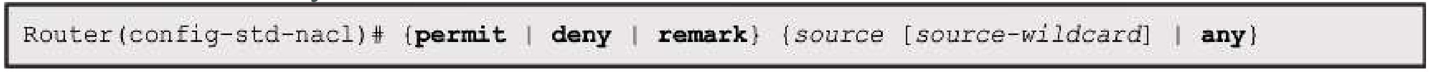
```
access-list acl-# {permit | deny | remark} protocol source-addr [source-wildcard]
dest-addr [dest-wildcard][operator port][established]
```

Named ACL Syntax



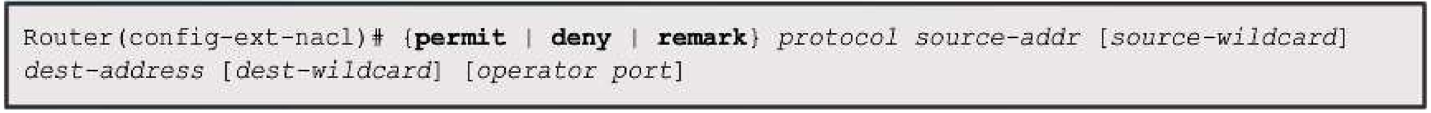
```
Router(config)# ip access-list [standard | extended] name_of_ACL
```

Standard ACE Syntax



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

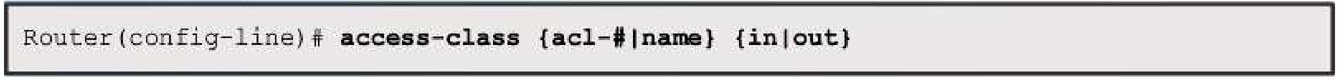
Extended ACE Syntax



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

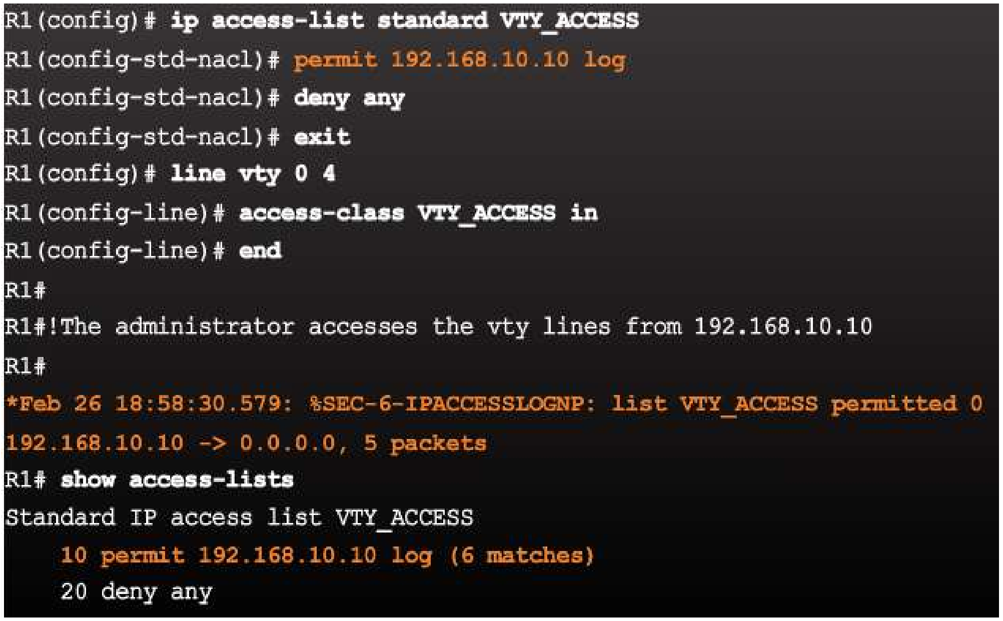
Applying an ACL

Syntax - Apply an ACL to the VTY lines



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

Example - Named ACL on VTY lines with logging



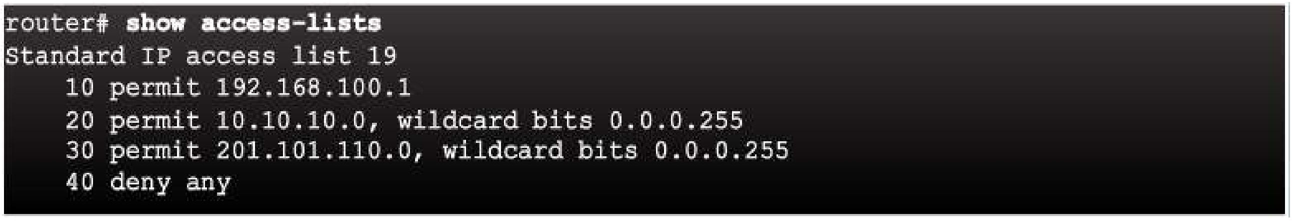
```
R1(config)# ip access-list standard VTY_ACCESS
R1(config-std-nacl)# permit 192.168.10.10 log
R1(config-std-nacl)# deny any
R1(config-std-nacl)# exit
R1(config)# line vty 0 4
R1(config-line)# access-class VTY_ACCESS in
R1(config-line)# end
R1#
R1#!The administrator accesses the vty lines from 192.168.10.10
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
```

Guidelines for ACL Configuration

- Create an ACL in global configuration mode
- Ensure the last statement is an implicit `deny any` or `deny any any`
- Remember that statement order is important because ACLs are processed top-down
- Ensure the most specific statements are at the top of the list
- Only one ACL per interface, per direction
- Remember that new statements for an existing ACL are added to the bottom of the ACL by default
- Standard ACL – place as close to the destination as possible
- Extended ACL – place as close to the source as possible

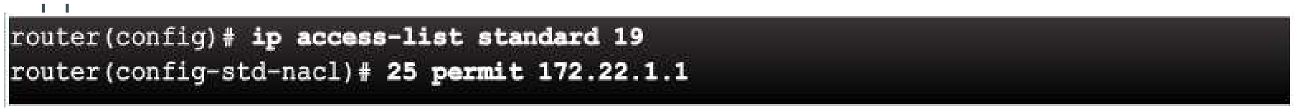
Edit Existing ACLs

Existing access list has four entries



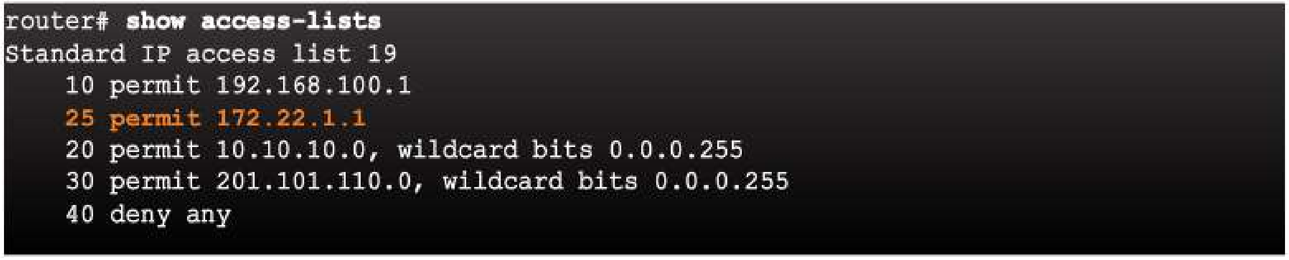
```
router# show access-lists
Standard IP access list 19
    10 permit 192.168.100.1
    20 permit 10.10.10.0, wildcard bits 0.0.0.255
    30 permit 201.101.110.0, wildcard bits 0.0.0.255
    40 deny any
```

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



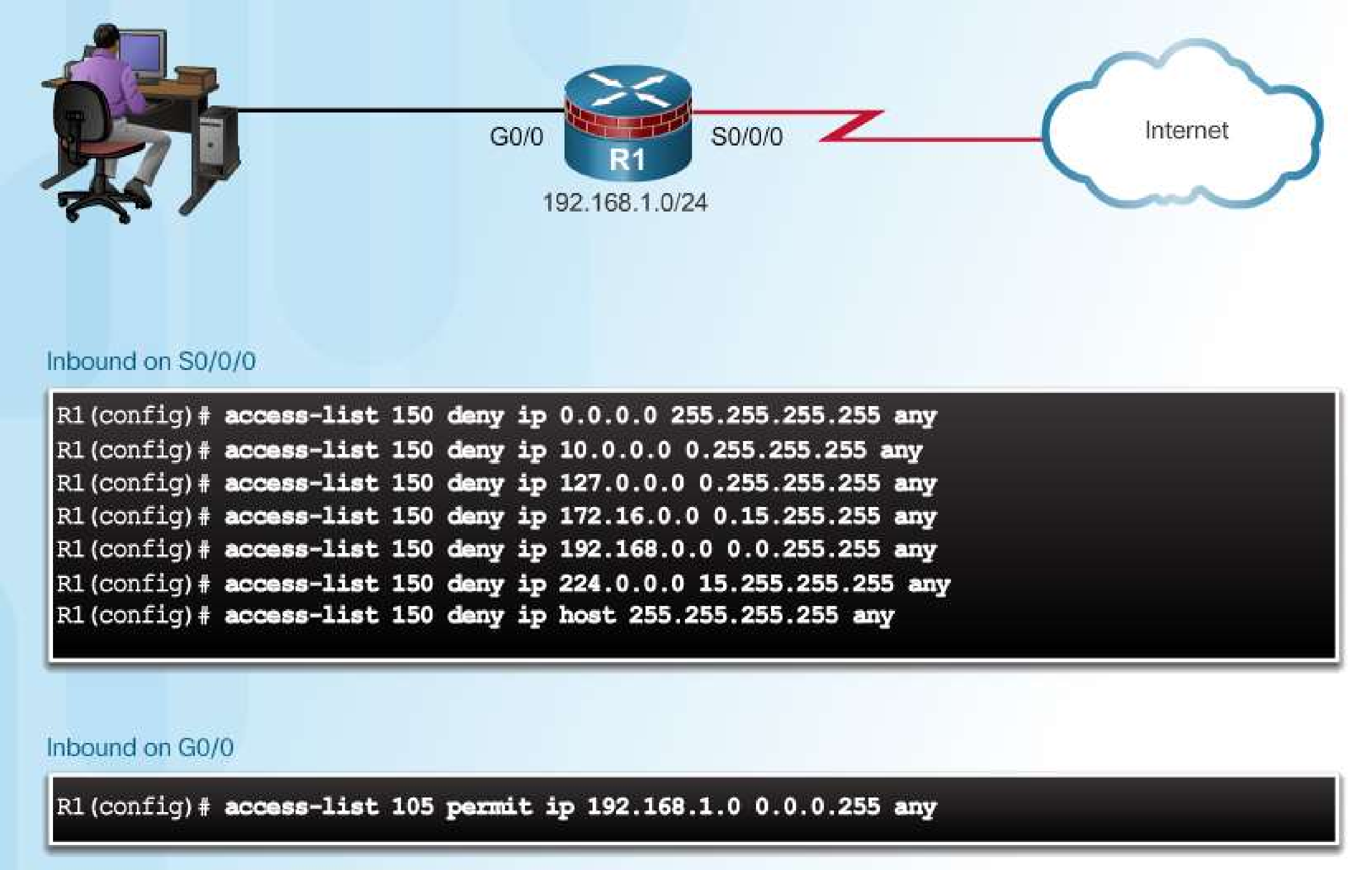
```
router(config)# ip access-list standard 19
router(config-std-nacl)# 25 permit 172.22.1.1
```

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 Network Attacks with ACLs
Antispoofing with ACLs



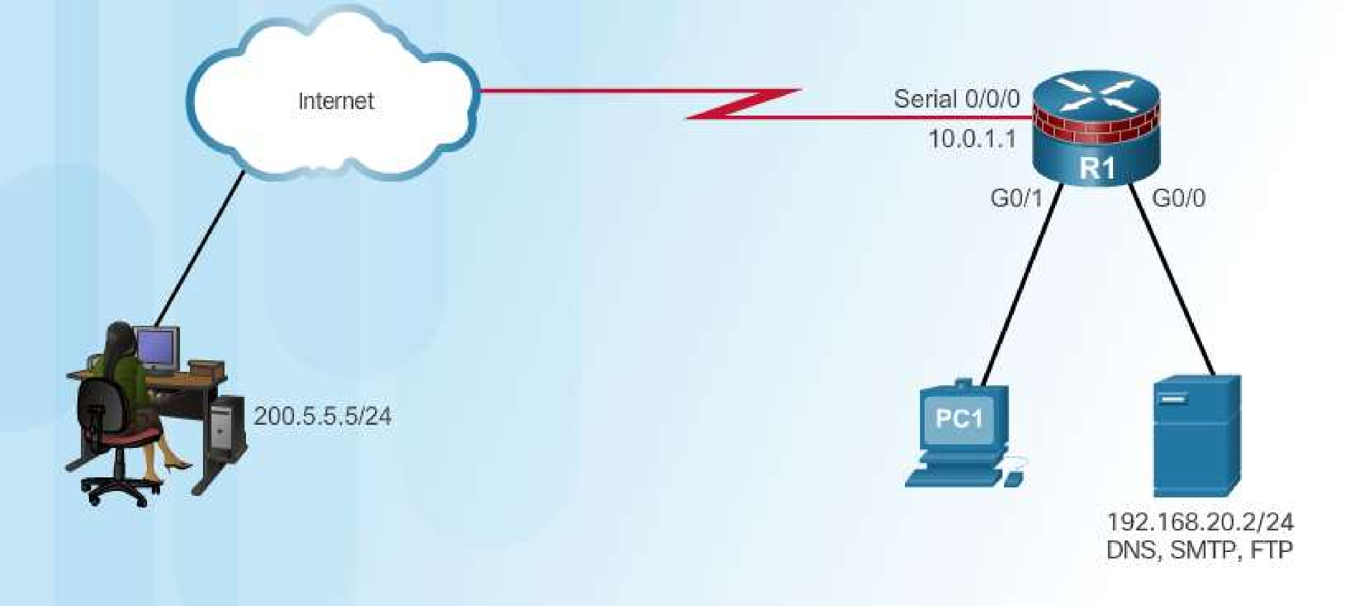
Inbound on S0/0/0

```
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
```

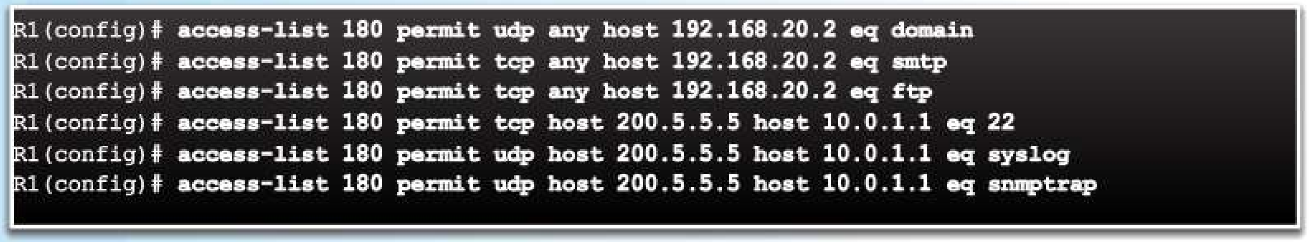
Inbound on G0/0

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

Permitting Necessary Traffic through a Firewall

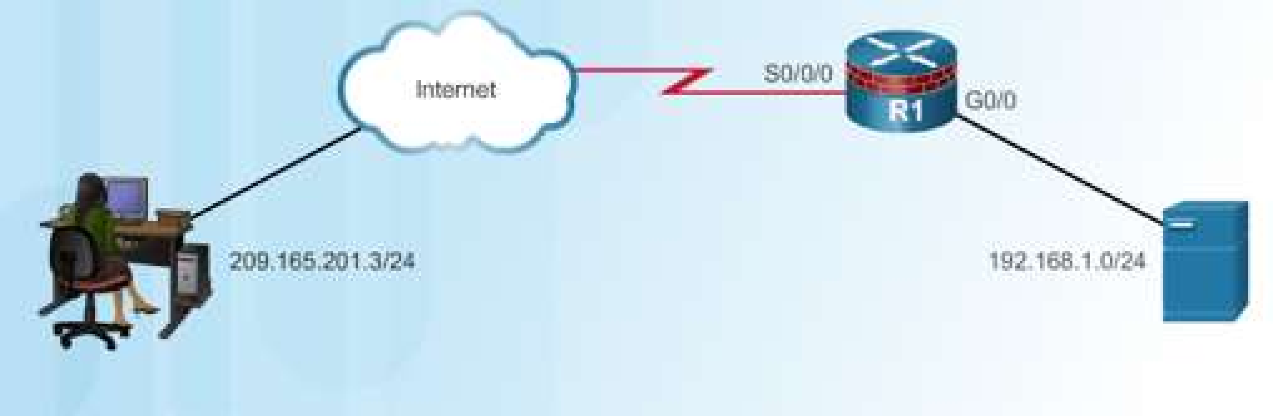


Inbound on Serial 0/0/0



```
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
```

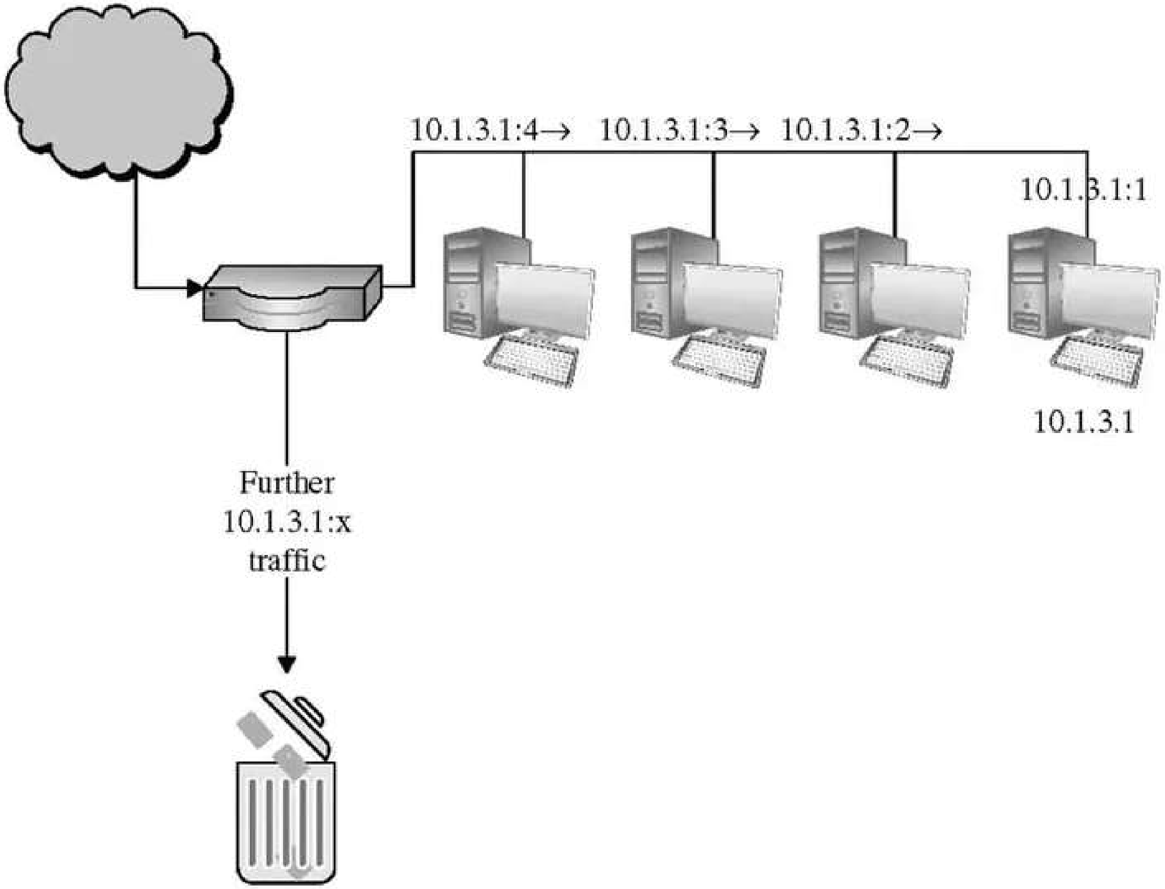
Block ICMP traffic



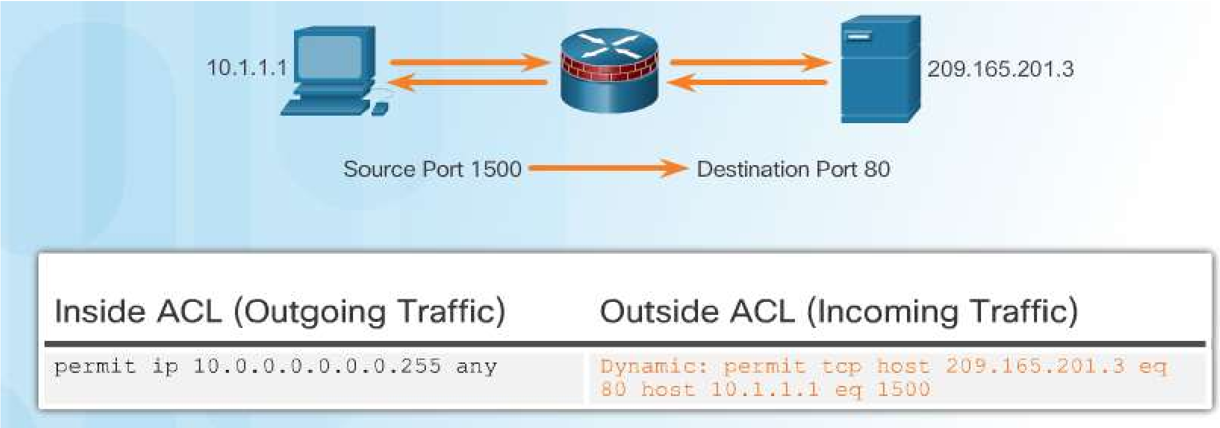
```
access-list 102 deny icmp any 192.168.1.0 0.0.0.255
access-list 102 permit icmp any any
access-list 102 permit ip any any
```

Stateful Firewall

- examine each IP packet in context
  - keeps tracks of client-server sessions
  - checks each packet validly belongs to one
  - “remember” the network activities of hosts
- The goal of a stateful inspection firewall is to identify hosts that represent a threat by accumulating evidence against them
- If the negative evidence against a host exceeds a threshold established by the firewall’s security policy, the host can be blocked.


10.1.3.1:4→
10.1.3.1:3→
10.1.3.1:2→
10.1.3.1:1
10.1.3.1
Further
10.1.3.1:x
traffic

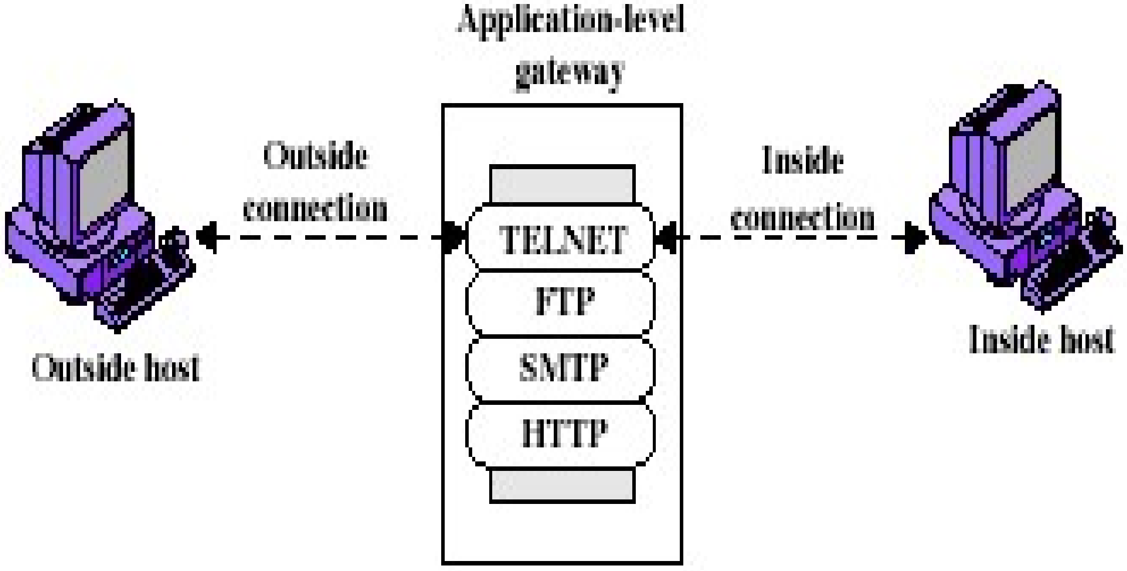
Stateful Firewall



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`

Benefits	Limitations
Primary means of defense	No Application Layer inspection
Strong packet filtering	Cannot filter stateless protocols
Improved performance over packet filters	Difficult to defend against dynamic port negotiation
Defends against spoofing and DoS attacks	No authentication support
Richer data log	

Application level Gateway Firewall



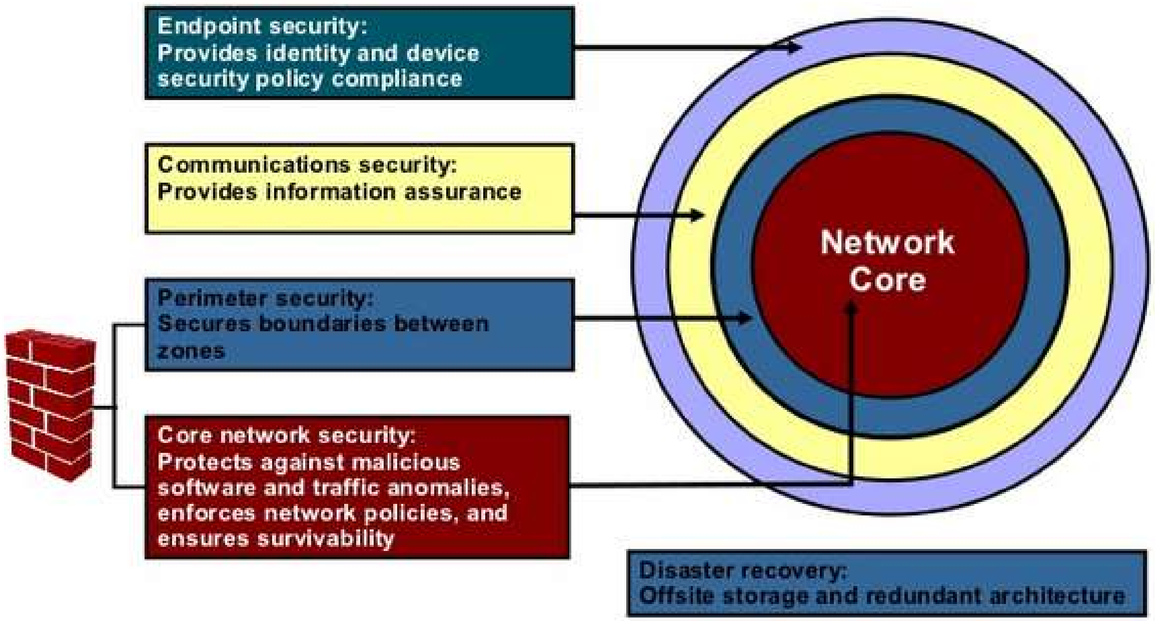
(b) Application-level gateway

Application level Gateway Firewall

- use an application specific gateway / proxy
- has full access to protocol
  - user requests service from proxy
  - proxy validates request as legal
  - then actions request and returns result to user
- need separate proxies for each service
  - some services naturally support proxy
  - others are more problematic
  - custom services generally not supported

Firewalls in Network Design

Layered Defense Scenario



Endpoint security:
Provides identity and device security policy compliance

Communications security:
Provides information assurance

Perimeter security:
Secures boundaries between zones

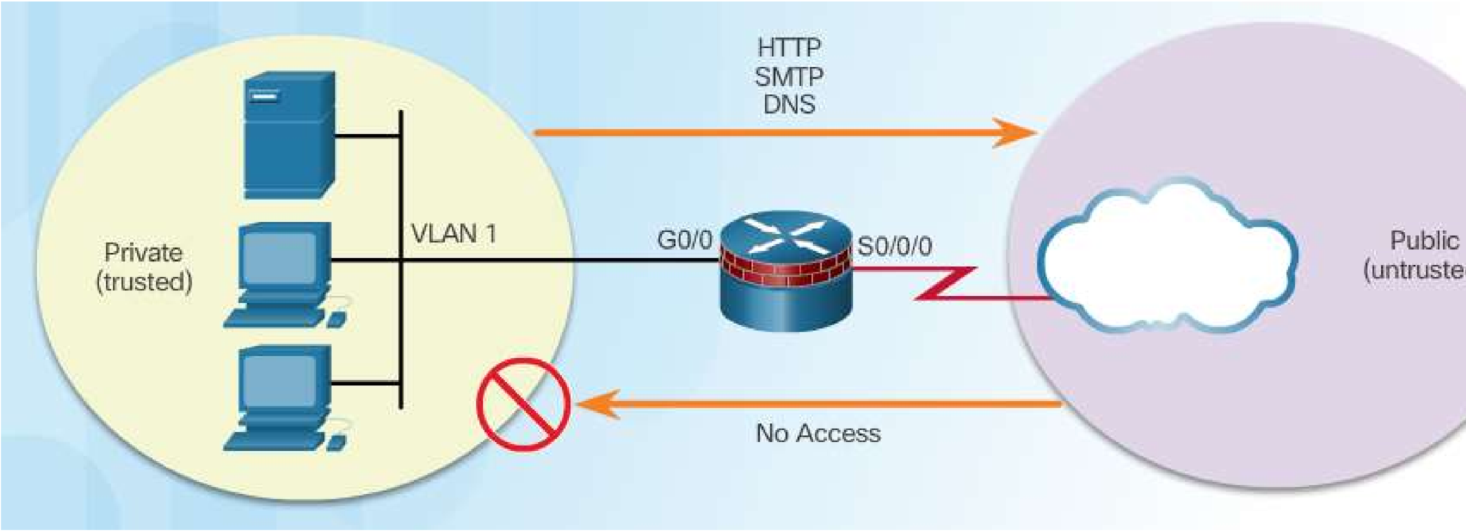
Core network security:
Protects against malicious software and traffic anomalies, enforces network policies, and ensures survivability

Disaster recovery:
Offsite storage and redundant architecture

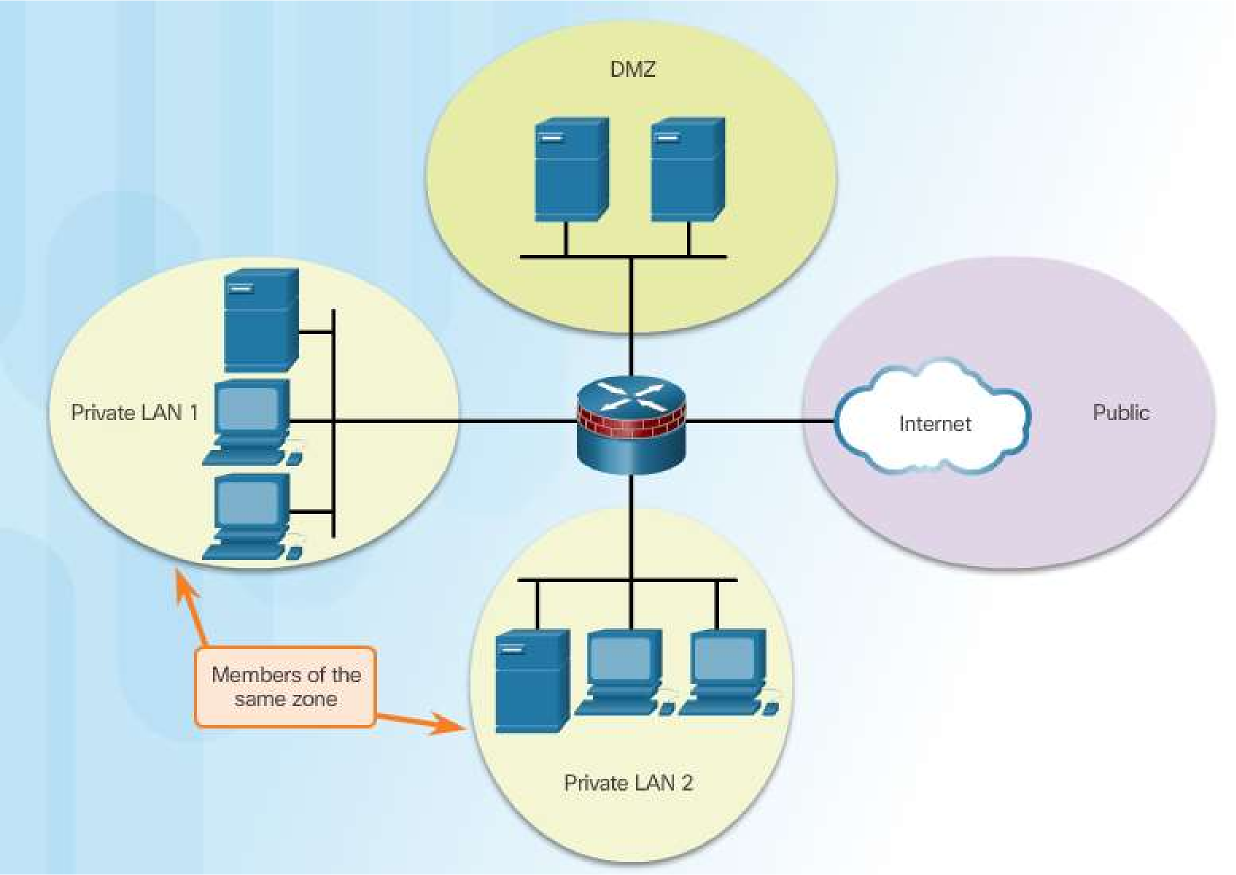
Firewall Best Practices

- Position firewalls at security boundaries
- Unwise to rely exclusively on a firewall for security, because firewalls are the primary security device
- Permit only services that are needed, deny all other traffics
- Ensure that physical access to the firewall is controlled
- Practice change management for firewall configuration changes
- Remember that firewalls primarily protect from attacks originating from the outside

Inside and Outside Networks (Two Zones)



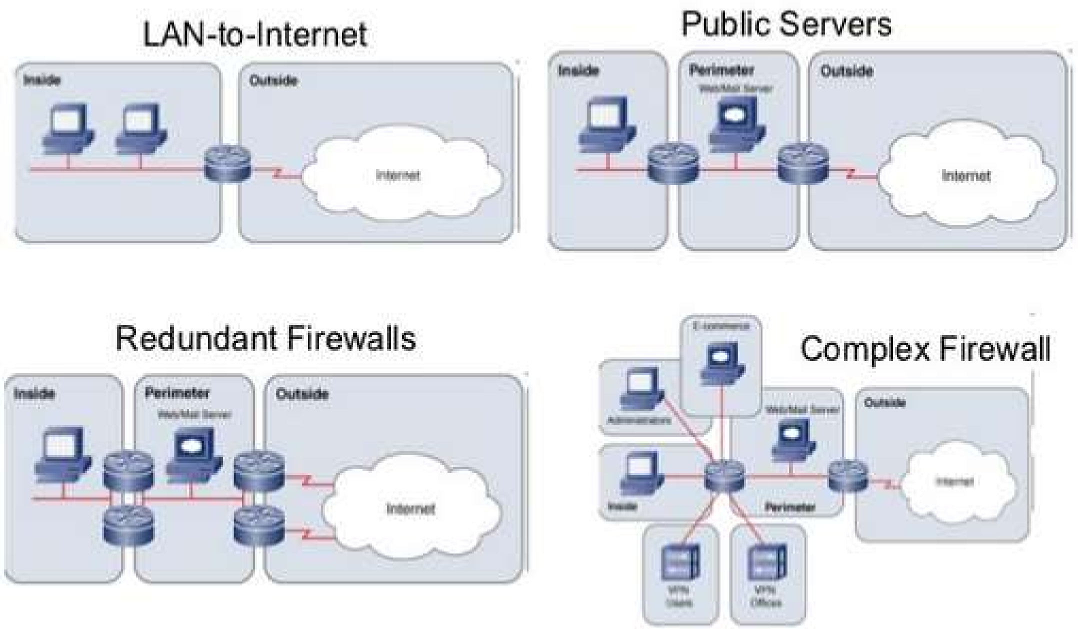
Zone-Based Policy Firewalls



Zone-Based Policy Firewalls

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

ZPF Designs



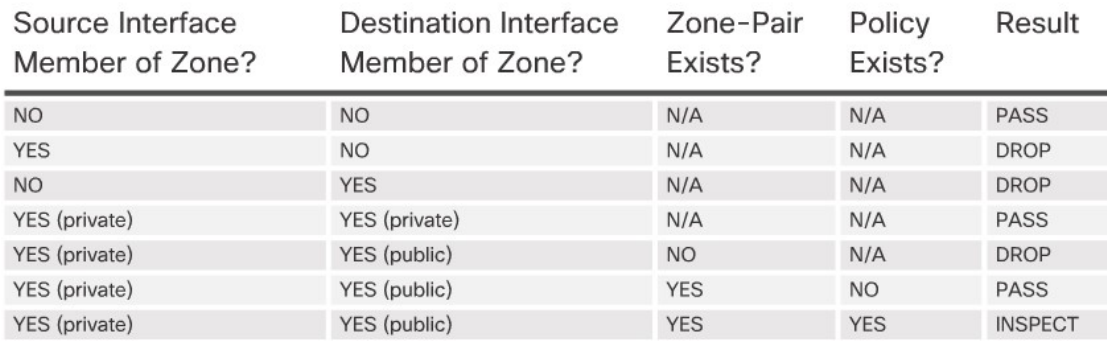
Design steps:

- Determine the zones
- Establish policies between zones
- Design the physical infrastructure
- Identify subsets within zones and merge traffic requirements

ZPF Actions

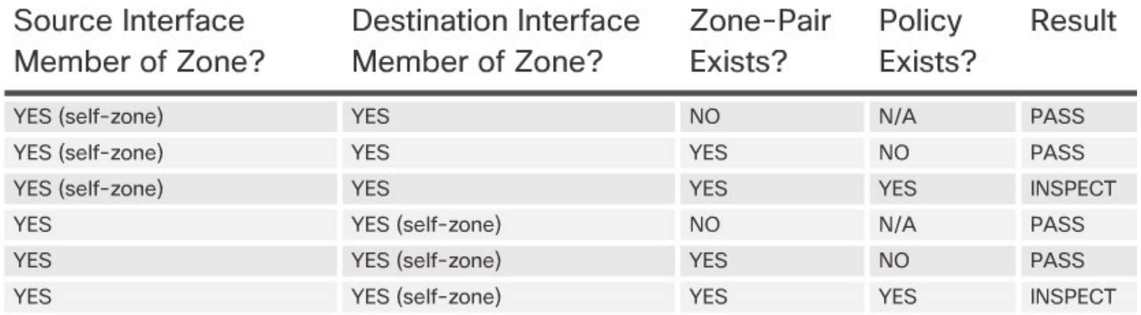
- **Inspect** - Configures Cisco IOS stateful packet inspections.
- **Drop** - Analogous to a deny statement in an ACL. A log option is available to log the rejected packets.
- **Pass** - Analogous to a permit statement in an ACL. The pass action does not track the state of connections or sessions within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

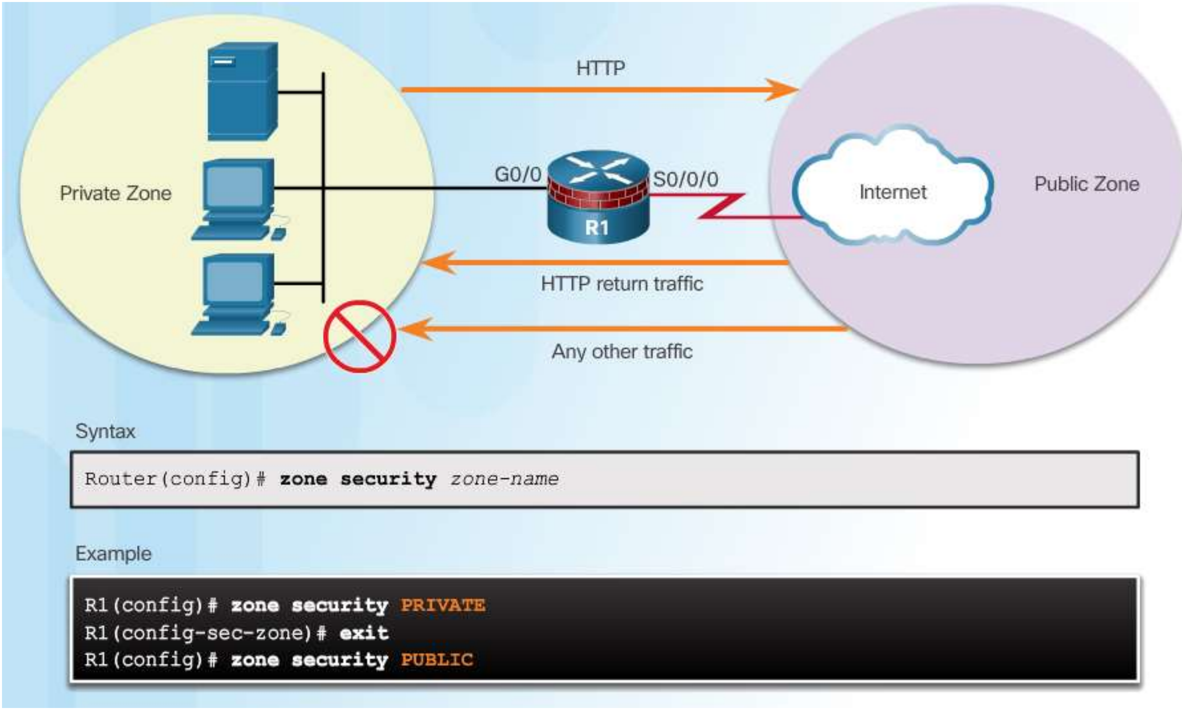
Rules for Traffic to the Self Zone



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 Zones



Syntax

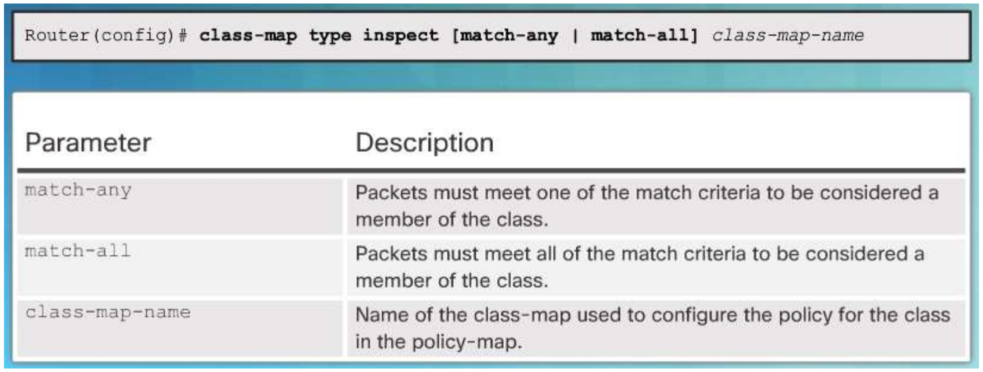
```
Router(config)# zone security zone-name
```

Example

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

Step 2: Identify Traffic

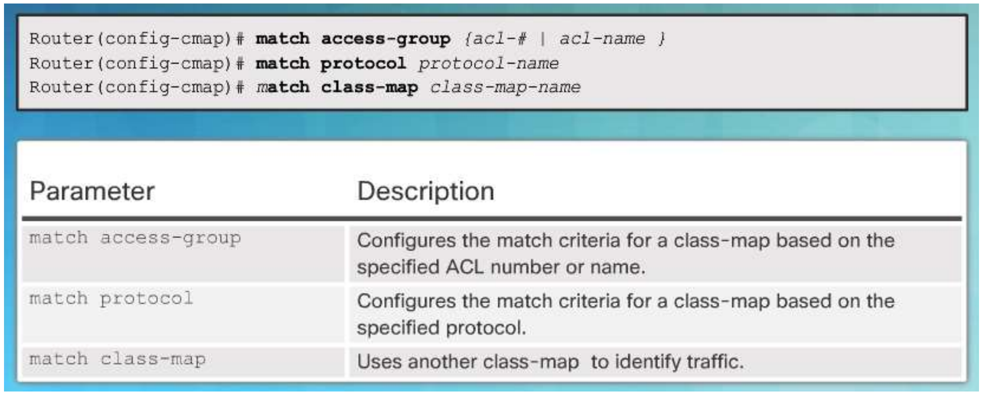
Command syntax for `class-map`



```
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`

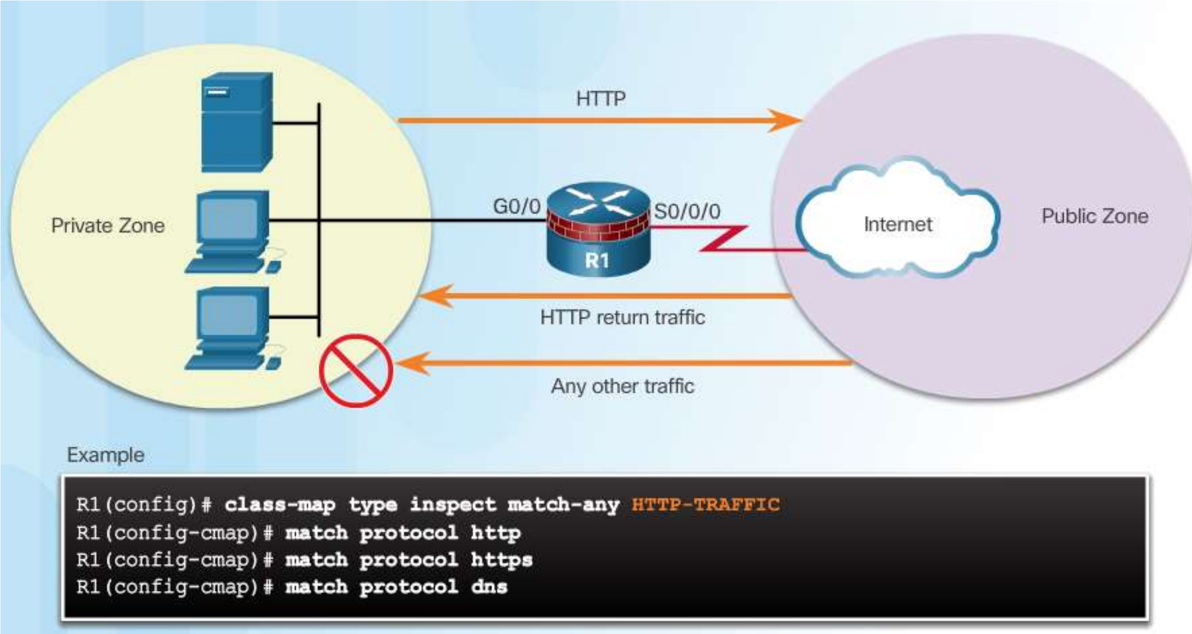


```
Router(config-cmap)# match access-group {acl-# | acl-name }
Router(config-cmap)# match protocol protocol-name
Router(config-cmap)# match class-map class-map-name
```

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

Example `class-map` configuration

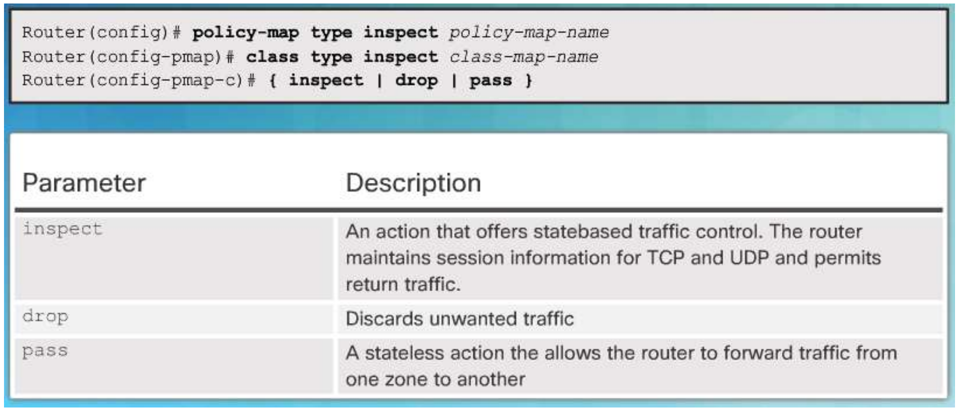


Example

```
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

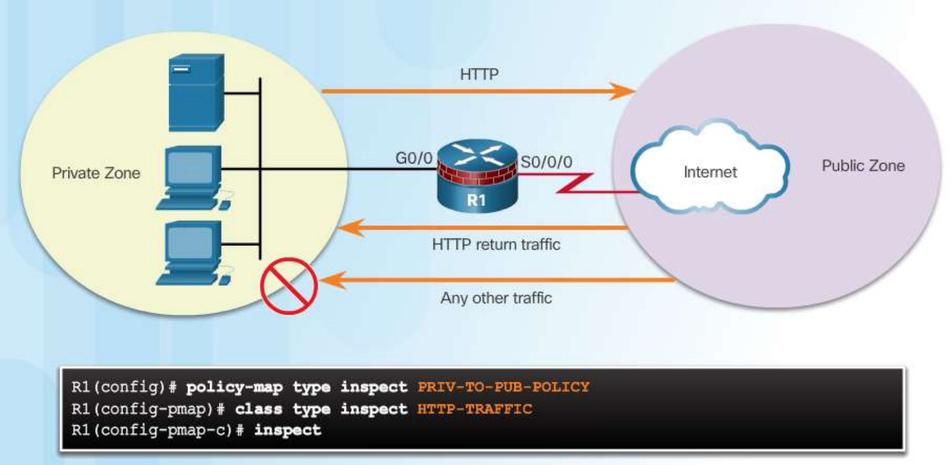
Command syntax for `policy-map`



```
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

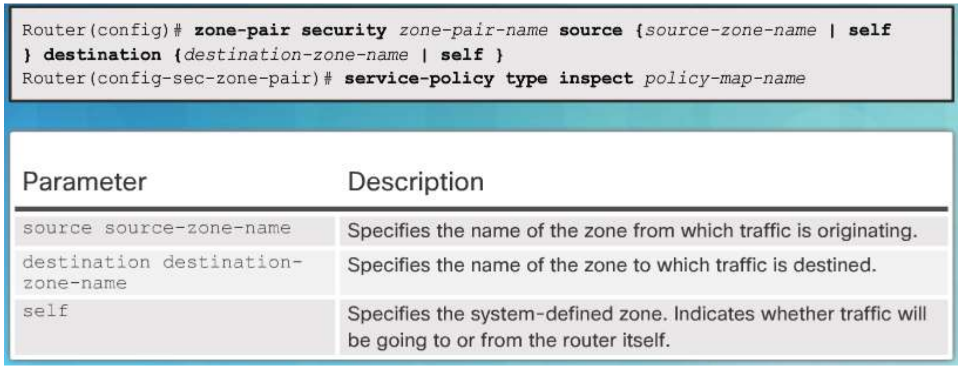
Example `policy-map` configuration



```
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

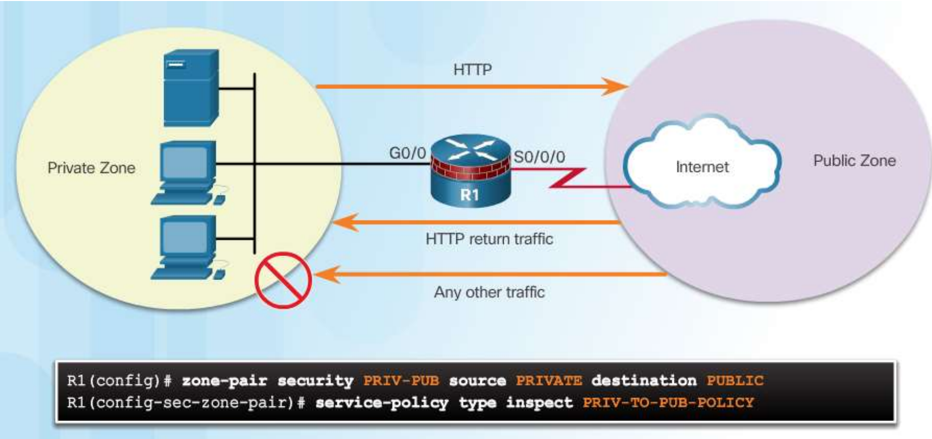
Command syntax for `zone-pair` and `service-policy`



```
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
```

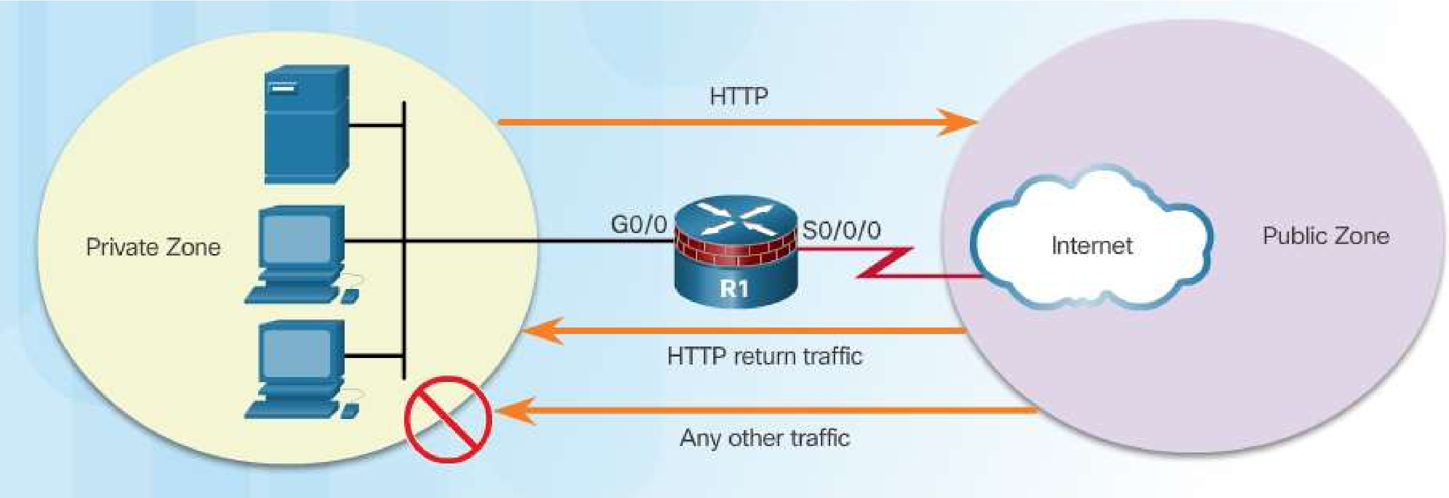
Parameter	Description
`source source-zone-name`	Specifies the name of the zone from which traffic is originating.
`destination destination-zone-name`	Specifies the name of the zone to which traffic is destined.
`self`	Specifies the system-defined zone. Indicates whether traffic will be going to or from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

Commonly used 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.
- 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.