

# Zone-Based Policy Firewall Design and Application Guide

Document ID: 98628

---

## **Introduction**

### **Prerequisites**

- Requirements
- Components Used
- Conventions

### **Zone-Based Policy Overview**

### **Zone-Based Policy Configuration Model**

### **Rules For Applying Zone-Based Policy Firewall**

### **Designing Zone-Based Policy Network Security**

### **Using IPSec VPN with Zone-Based Policy Firewall**

### **Cisco Policy Language (CPL) Configuration**

- Configuring Zone-Based Policy Firewall Class-Maps
- Configuring Zone-Based Policy Firewall Policy-Maps
- Configuring Zone-Policy Firewall Parameter-Maps
- Applying Logging For Zone-Based Policy Firewall Policies
- Editing Zone-Policy Firewall Class-Maps and Policy-Maps

### **Configuration Examples**

- Stateful Inspection Routing Firewall
- Stateful Inspection Transparent Firewall
- Rate Policing For Zone-Based Policy Firewall
- URL Filtering
- Controlling Access to the Router

### **Zone-Based Firewall and Wide-Area Application Services**

### **Monitoring Zone-Based Policy Firewall with show and debug Commands**

### **Tuning Zone-Based Policy Firewall Denial-of-Service Protection**

### **Appendix**

- Appendix A: Basic Configuration
- Appendix B: Final (Complete) Configuration
- Appendix C: Basic Zone-Policy Firewall Configuration for Two Zones

### **NetPro Discussion Forums – Featured Conversations**

### **Related Information**

---

## **Introduction**

Cisco IOS® Software Release 12.4(6)T introduced Zone-Based Policy Firewall (ZFW), a new configuration model for the Cisco IOS Firewall feature set. This new configuration model offers intuitive policies for multiple-interface routers, increased granularity of firewall policy application, and a default deny-all policy that prohibits traffic between firewall security zones until an explicit policy is applied to allow desirable traffic.

Nearly all classic Cisco IOS Firewall features implemented before Cisco IOS Software Release 12.4(6)T are supported in the new zone-based policy inspection interface:

- Stateful packet inspection
- VRF-aware Cisco IOS Firewall
- URL filtering

- Denial-of-Service (DoS) mitigation

Cisco IOS Software Release 12.4(9)T added ZFW support for per-class session/connection and throughput limits, as well as application inspection and control:

- HTTP
- Post Office Protocol (POP3), Internet Mail Access Protocol (IMAP), Simple Mail Transfer Protocol/Enhanced Simple Mail Transfer Protocol (SMTP/ESMTP)
- Sun Remote Procedure Call (RPC)
- Instant Messaging (IM) applications:
  - ◆ Microsoft Messenger
  - ◆ Yahoo! Messenger
  - ◆ AOL Instant Messenger
- Peer-to-Peer (P2P) File Sharing:
  - ◆ Bittorrent
  - ◆ KaZaA
  - ◆ Gnutella
  - ◆ eDonkey

Cisco IOS Software Release 12.4(11)T added statistics for easier DoS protection tuning.

Some Cisco IOS Classic Firewall features and capabilities are not yet supported in a ZFW in Cisco IOS Software Release 12.4(15)T:

- Authentication proxy
- Stateful firewall failover
- Unified firewall MIB
- IPv6 stateful inspection
- TCP out-of-order support

ZFW generally improves Cisco IOS performance for most firewall inspection activities.

Neither Cisco IOS ZFW or Classic Firewall include stateful inspection support for multicast traffic.

## Prerequisites

## Requirements

There are no specific requirements for this document.

## Components Used

This document is not restricted to specific software and hardware versions.

## Conventions

Refer to Cisco Technical Tips Conventions for more information on document conventions.

# Zone–Based Policy Overview

Cisco IOS Classic Firewall stateful inspection (formerly known as Context–Based Access Control, or CBAC) employed an interface–based configuration model, in which a stateful inspection policy was applied to an interface. All traffic passing through that interface received the same inspection policy. This configuration model limited the granularity of the firewall policies and caused confusion of the proper application of firewall policies, particularly in scenarios when firewall policies must be applied between multiple interfaces.

Zone–Based Policy Firewall (also known as Zone–Policy Firewall, or ZFW) changes the firewall configuration from the older interface–based model to a more flexible, more easily understood zone–based model. Interfaces are assigned to zones, and inspection policy is applied to traffic moving between the zones. Inter–zone policies offer considerable flexibility and granularity, so different inspection policies can be applied to multiple host groups connected to the same router interface.

Firewall policies are configured with the Cisco® Policy Language (CPL), which employs a hierarchical structure to define inspection for network protocols and the groups of hosts to which the inspection will be applied.

## Zone–Based Policy Configuration Model

ZFW completely changes the way you configure a Cisco IOS Firewall inspection, as compared to the Cisco IOS Classic Firewall.

The first major change to the firewall configuration is the introduction of zone–based configuration. Cisco IOS Firewall is the first Cisco IOS Software threat defense feature to implement a zone configuration model. Other features might adopt the zone model over time. Cisco IOS Classic Firewall stateful inspection (or CBAC) interface–based configuration model that employs the **ip inspect** command set is maintained for a period of time. However, few, if any, new features are configurable with the classical command–line interface (CLI). ZFW does not use the stateful inspection or CBAC commands. The two configuration models can be used concurrently on routers, but not combined on interfaces. An interface cannot be configured as a security zone member as well as being configured for **ip inspect** simultaneously.

Zones establish the security borders of your network. A zone defines a boundary where traffic is subjected to policy restrictions as it crosses to another region of your network. ZFW s default policy between zones is deny all. If no policy is explicitly configured, all traffic moving between zones is blocked. This is a significant departure from stateful inspection s model where traffic was implicitly allowed until explicitly blocked with an access control list (ACL).

The second major change is the introduction of a new configuration policy language known as CPL. Users familiar with the Cisco IOS Software Modular quality–of–service (QoS) CLI (MQC) might recognize that the format is similar to QoS s use of class maps to specify which traffic will be affected by the action applied in a policy map.

## Rules For Applying Zone–Based Policy Firewall

Router network interfaces membership in zones is subject to several rules that govern interface behavior, as is the traffic moving between zone member interfaces:

- A zone must be configured before interfaces can be assigned to the zone.
- An interface can be assigned to only one security zone.
- All traffic to and from a given interface is implicitly blocked when the interface is assigned to a zone, except traffic to and from other interfaces in the same zone, and traffic to any interface on the router.

- Traffic is implicitly allowed to flow by default among interfaces that are members of the same zone.
- In order to permit traffic to and from a zone member interface, a policy allowing or inspecting traffic must be configured between that zone and any other zone.
- The self zone is the only exception to the default deny all policy. All traffic to any router interface is allowed until traffic is explicitly denied.
- Traffic cannot flow between a zone member interface and any interface that is not a zone member. Pass, inspect, and drop actions can only be applied between two zones.
- Interfaces that have not been assigned to a zone function as classical router ports and might still use classical stateful inspection/CBAC configuration.
- If it is required that an interface on the box not be part of the zoning/firewall policy. It might still be necessary to put that interface in a zone and configure a pass all policy (sort of a dummy policy) between that zone and any other zone to which traffic flow is desired.
- From the preceding it follows that, if traffic is to flow among all the interfaces in a router, all the interfaces must be part of the zoning model (each interface must be a member of one zone or another).
- The only exception to the preceding deny by default approach is the traffic to and from the router, which will be permitted by default. An explicit policy can be configured to restrict such traffic.

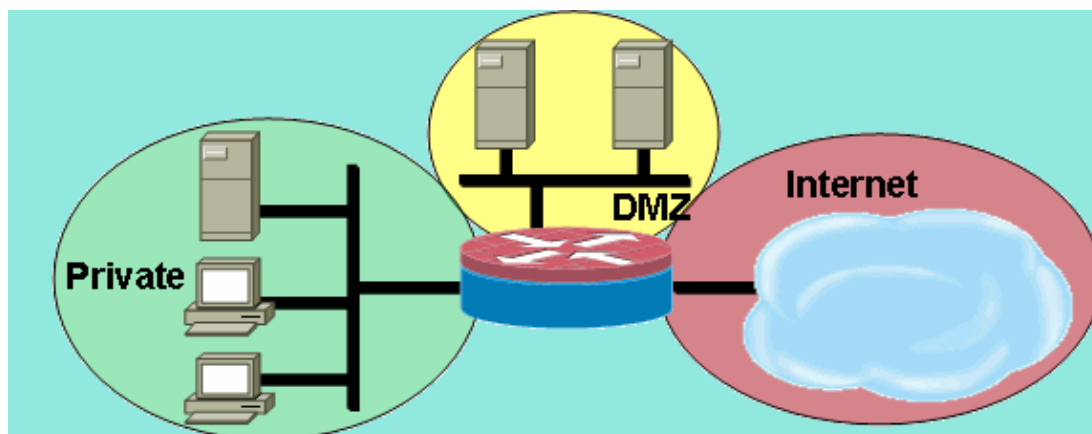
## Designing Zone–Based Policy Network Security

A security zone should be configured for each region of relative security within the network, so that all interfaces that are assigned to the same zone will be protected with a similar level of security. For example, consider an access router with three interfaces:

- One interface connected to the public Internet
- One interface connected to a private LAN that must not be accessible from the public Internet
- One interface connected to an Internet service demilitarized zone (DMZ), where a Web server, Domain Name System (DNS) server, and e–mail server must be accessible to the public Internet

Each interface in this network will be assigned to its own zone, although you might want to allow varied access from the public Internet to specific hosts in the DMZ and varied application use policies for hosts in the protected LAN. (See Figure 1.)

**Figure 1: Basic Security Zone Topology**



In this example, each zone holds only one interface. If an additional interface is added to the private zone, the hosts connected to the new interface in the zone can pass traffic to all hosts on the existing interface in the same zone. Additionally, the hosts' traffic to hosts in other zones is similarly affected by existing policies.

Typically, the example network will have three main policies:

- Private zone connectivity to the Internet
- Private zone connectivity to DMZ hosts
- Internet zone connectivity to DMZ hosts

Because the DMZ is exposed to the public Internet, the DMZ hosts might be subjected to undesired activity from malicious individuals who might succeed at compromising one or more DMZ hosts. If no access policy is provided for DMZ hosts to reach either private zone hosts or Internet zone hosts, then the individuals who compromised the DMZ hosts cannot use the DMZ hosts to carry out further attack against private or Internet hosts. ZFW imposes a prohibitive default security posture. Therefore, unless the DMZ hosts are specifically provided access to other networks, other networks are safeguarded against any connections from the DMZ hosts. Similarly, no access is provided for Internet hosts to access the private zone hosts, so private zone hosts are safe from unwanted access by Internet hosts.

## Using IPSec VPN with Zone–Based Policy Firewall

Recent enhancements to IPSec VPN simplify firewall policy configuration for VPN connectivity. IPSec Virtual Tunnel Interface (VTI) and GRE+IPSec allow the confinement of VPN site–to–site and client connections to a specific security zone by placing the tunnel interfaces in a specified security zone. Connections can be isolated in a VPN DMZ if connectivity must be limited by a specific policy. Or, if VPN connectivity is implicitly trusted, VPN connectivity can be placed in the same security zone as the trusted inside network.

If a non–VTI IPSec is applied, VPN connectivity firewall policy requires close scrutiny to maintain security. The zone policy must specifically allow access by an IP address for remote sites hosts or VPN clients if secure hosts are in a different zone than the VPN client s encrypted connection to the router. If the access policy is not properly configured, hosts that should be protected can end up exposed to unwanted, potentially hostile hosts. Refer to Using VPN with Zone–Based Policy Firewall for further concept and configuration discussion.

## Cisco Policy Language (CPL) Configuration

This procedure can be used to configure a ZFW. The sequence of steps is not important, but some events must be completed in order. For instance, you must configure a class–map before you assign a class–map to a policy–map. Similarly, you cannot assign a policy–map to a zone–pair until you have configured the policy. If you try to configure a section that relies on another portion of the configuration that you have not configured, the router responds with an error message.

1. Define zones.
2. Define zone–pairs.
3. Define class–maps that describe traffic that must have policy applied as it crosses a zone–pair.
4. Define policy–maps to apply action to your class–maps traffic.
5. Apply policy–maps to zone–pairs.
6. Assign interfaces to zones.

## Configuring Zone–Based Policy Firewall Class–Maps

Class–maps define the traffic that the firewall selects for policy application. Layer 4 class–maps sort the traffic based on these criteria listed here. These criteria are specified using the **match** command in a class–map:

- Access–group A standard, extended, or named ACL can filter traffic based on source and destination IP address and source and destination port.

- Protocol The Layer 4 protocols (TCP, UDP, and ICMP) and application services such as HTTP, SMTP, DNS, etc. Any well-known or user-defined service known to Port-Application Mapping can be specified.
- Class-map A subordinate class-map that provides additional match criteria can be nested inside another class-map.
- Not The *not* criterion specifies that any traffic that does not match a specified service (protocol), access-group or subordinate class-map will be selected for the class-map.

### Combining Match Criteria: Match-Any versus Match-All

Class-maps can apply match-any or match-all operators to determine how to apply the match criteria. If match-any is specified, traffic must meet only one of the match criteria in the class-map. If match-all is specified, traffic must match all of the class-map's criteria in order to belong to that particular class.

Match criteria must be applied in order from more specific to less specific, if traffic meets multiple criteria. For example, consider this class-map:

```
class-map type inspect match-any my-test-cmap
  match protocol http
  match protocol tcp
```

HTTP traffic must encounter the match protocol http first to make sure the traffic is handled by the service-specific capabilities of HTTP inspection. If the match lines are reversed, so traffic encounters the match protocol tcp statement before it compares it to match protocol http, the traffic is simply classified as TCP traffic, and inspected according to the capabilities of the Firewall's TCP Inspection component. This is a problem for certain services such as FTP, TFTP, and several multimedia and voice signaling services such as H.323, SIP, Skinny, RTSP, and others. These services require additional inspection capabilities to recognize the more complex activities of these services.

### Applying an ACL as Match Criteria

Class-maps can apply an ACL as one of the match criteria for policy application. If a class-map's only match criterion is an ACL and the class-map is associated with a policy-map applying the inspect action, the router applies basic TCP or UDP inspection for all traffic allowed by the ACL, except that which ZFW provides application-aware inspection. This includes (but not limited to) FTP, SIP, Skinny (SCCP), H.323, Sun RPC, and TFTP. If application-specific inspection is available and the ACL allows the primary or control channel, any secondary or media channel associated with the primary/control is allowed, regardless of whether the ACL allows the traffic.

If a class-map applies only ACL 101 as the match criteria, an ACL 101 appears as this:

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

All traffic is allowed in the direction of the service-policy applied to a given zone-pair, and corresponding return traffic is allowed in the opposite direction. Therefore, the ACL must apply the restriction to limit traffic to specific desired types. Note that the PAM list includes application services such as HTTP, NetBIOS, H.323, and DNS. However, in spite of PAM's knowledge of the specific application's use of a given port, firewall only applies sufficient application-specific capability to accommodate the well-known requirements of the application traffic. Thus, simple application traffic such as telnet, SSH, and other single-channel applications are inspected as TCP, and their statistics are combined together in the **show** command output. If application-specific visibility into network activity is desired, you need to configure inspection for services by application name (configure match protocol http, match protocol telnet, etc.).

Compare the statistics available in the **show policy-map type inspect zone-pair** command output from this configuration with the more explicit firewall policy shown further down the page. This configuration is used

to inspect traffic from a Cisco IP Phone, as well as several workstations that use a variety of traffic, which includes http, ftp, netbios, ssh, and dns:

```
class-map type inspect match-all all-private
  match access-group 101
!
policy-map type inspect priv-pub-pmap
  class type inspect all-private
    inspect
  class class-default
!
zone security private
zone security public
zone-pair security priv-pub source private destination public
  service-policy type inspect priv-pub-pmap
!
interface FastEthernet4
  ip address 172.16.108.44 255.255.255.0
  zone-member security public
!
interface Vlan1
  ip address 192.168.108.1 255.255.255.0
  zone-member security private
!
access-list 101 permit ip 192.168.108.0 0.0.0.255 any
```

While this configuration is easy to define and accommodates all traffic that originates in the private zone (as long as the traffic observes the standard, PAM-recognized destination ports), it provides limited visibility into service activity, and does not offer the opportunity to apply ZFW s bandwidth and session limits for specific types of traffic. This **show policy-map type inspect zone-pair priv-pub** command output is the result of the previous simple configuration that uses only a permit ip [subnet] any ACL between zone-pairs. As you can see, most of workstation traffic is counted in the basic TCP or UDP statistics:

```
stg-871-L#show policy-map type insp zone-pair priv-pub
Zone-pair: priv-pub

Service-policy inspect : priv-pub-pmap

Class-map: all-private (match-all)
  Match: access-group 101
  Inspect
    Packet inspection statistics [process switch:fast switch]
    tcp packets: [413:51589]
    udp packets: [74:28]
    icmp packets: [0:8]
    ftp packets: [23:0]
    tftp packets: [3:0]
    tftp-data packets: [6:28]
    skinny packets: [238:0]

    Session creations since subsystem startup or last reset 39
    Current session counts (estab/half-open/terminating) [3:0:0]
    Maxever session counts (estab/half-open/terminating) [3:4:1]
    Last session created 00:00:20
    Last statistic reset never
    Last session creation rate 2
    Maxever session creation rate 7
    Last half-open session total 0

Class-map: class-default (match-any)
  Match: any
  Drop (default action)
    0 packets, 0 bytes
```

By contrast, a similar configuration that adds application-specific classes provides more granular application statistics and control, and still accommodates the same breadth of services that was shown in the first example by defining the last-chance class-map matching only the ACL as the last chance in the policy-map:

```
class-map type inspect match-all all-private
  match access-group 101
class-map type inspect match-all private-ftp
  match protocol ftp
  match access-group 101
class-map type inspect match-any netbios
  match protocol msrpc
  match protocol netbios-dgm
  match protocol netbios-ns
  match protocol netbios-ssn
class-map type inspect match-all private-netbios
  match class-map netbios
  match access-group 101
class-map type inspect match-all private-ssh
  match protocol ssh
  match access-group 101
class-map type inspect match-all private-http
  match protocol http
  match access-group 101
!
policy-map type inspect priv-pub-pmap
  class type inspect private-http
    inspect
  class type inspect private-ftp
    inspect
  class type inspect private-ssh
    inspect
  class type inspect private-netbios
    inspect
  class type inspect all-private
    inspect
  class class-default!
zone security private
zone security public
zone-pair security priv-pub source private destination public
  service-policy type inspect priv-pub-pmap
!
interface FastEthernet4
  ip address 172.16.108.44 255.255.255.0
  zone-member security public
!
interface Vlan1
  ip address 192.168.108.1 255.255.255.0
  zone-member security private
!
access-list 101 permit ip 192.168.108.0 0.0.0.255 any
```

The more-specific configuration provides this substantial granular output for the **show policy-map type inspect zone-pair priv-pub** command:

```
stg-871-L#sh policy-map type insp zone-pair priv-pub
Zone-pair: priv-pub

Service-policy inspect : priv-pub-pmap

Class-map: private-http (match-all)
  Match: protocol http
  Match: access-group 101
  Inspect
    Packet inspection statistics [process switch:fast switch]
    tcp packets: [0:2193]
```

Session creations since subsystem startup or last reset 731  
Current session counts (estab/half-open/terminating) [0:0:0]  
Maxever session counts (estab/half-open/terminating) [0:3:0]  
Last session created 00:29:25  
Last statistic reset never  
Last session creation rate 0  
Maxever session creation rate 4  
Last half-open session total 0

Class-map: private-ftp (match-all)

Match: protocol ftp

Inspect

Packet inspection statistics [process switch:fast switch]  
tcp packets: [86:167400]  
ftp packets: [43:0]

Session creations since subsystem startup or last reset 7  
Current session counts (estab/half-open/terminating) [0:0:0]  
Maxever session counts (estab/half-open/terminating) [2:1:1]  
Last session created 00:42:49  
Last statistic reset never  
Last session creation rate 0  
Maxever session creation rate 4  
Last half-open session total 0

Class-map: private-ssh (match-all)

Match: protocol ssh

Inspect

Packet inspection statistics [process switch:fast switch]  
tcp packets: [0:62]

Session creations since subsystem startup or last reset 4  
Current session counts (estab/half-open/terminating) [0:0:0]  
Maxever session counts (estab/half-open/terminating) [1:1:1]  
Last session created 00:34:18  
Last statistic reset never  
Last session creation rate 0  
Maxever session creation rate 2  
Last half-open session total 0

Class-map: private-netbios (match-all)

Match: access-group 101

Match: class-map match-any netbios

Match: protocol msrpc

0 packets, 0 bytes  
30 second rate 0 bps

Match: protocol netbios-dgm

0 packets, 0 bytes  
30 second rate 0 bps

Match: protocol netbios-ns

0 packets, 0 bytes  
30 second rate 0 bps

Match: protocol netbios-ssn

2 packets, 56 bytes  
30 second rate 0 bps

Inspect

Packet inspection statistics [process switch:fast switch]  
tcp packets: [0:236]

Session creations since subsystem startup or last reset 2  
Current session counts (estab/half-open/terminating) [0:0:0]  
Maxever session counts (estab/half-open/terminating) [1:1:1]  
Last session created 00:31:32  
Last statistic reset never  
Last session creation rate 0

```

Maxever session creation rate 1
Last half-open session total 0

Class-map: all-private (match-all)
  Match: access-group 101
  Inspect
    Packet inspection statistics [process switch:fast switch]
    tcp packets: [51725:158156]
    udp packets: [8800:70]
    tftp packets: [8:0]
    tftp-data packets: [15:70]
    skinny packets: [33791:0]

    Session creations since subsystem startup or last reset 2759
    Current session counts (estab/half-open/terminating) [2:0:0]
    Maxever session counts (estab/half-open/terminating) [2:6:1]
    Last session created 00:22:21
    Last statistic reset never
    Last session creation rate 0
    Maxever session creation rate 12
    Last half-open session total 0

Class-map: class-default (match-any)
  Match: any
  Drop (default action)
    4 packets, 112 bytes

```

Another added benefit of using a more granular class-map and policy-map configuration, as mentioned earlier, is the possibility of applying class-specific limits on session and rate values and specifically adjusting inspection parameters by applying a parameter-map to adjust each class's inspection behavior.

## Configuring Zone-Based Policy Firewall Policy-Maps

The policy-map applies firewall policy actions to one or more class-maps to define the service-policy that will be applied to a security zone-pair. When an inspect-type policy-map is created, a default class named class class-default is applied at the end of the class. The class class-default's default policy action is drop, but can be changed to pass. The log option can be added with the drop action. Inspect cannot be applied on class class-default.

### Zone-Based Policy Firewall Actions

ZFW provides three actions for traffic that traverses from one zone to another:

- **Drop** This is the default action for all traffic, as applied by the "class class-default" that terminates every inspect-type policy-map. Other class-maps within a policy-map can also be configured to drop unwanted traffic. Traffic that is handled by the drop action is "silently" dropped (i.e., no notification of the drop is sent to the relevant end-host) by the ZFW, as opposed to an ACL's behavior of sending an ICMP host unreachable message to the host that sent the denied traffic. Currently, there is not an option to change the "silent drop" behavior. The log option can be added with drop for syslog notification that traffic was dropped by the firewall.
- **Pass** This action allows the router to forward traffic from one zone to another. The pass action does not track the state of connections or sessions within the traffic. Pass only allows the traffic in one direction. A corresponding policy must be applied to allow return traffic to pass in the opposite direction. The pass action is useful for protocols such as IPSec ESP, IPSec AH, ISAKMP, and other inherently secure protocols with predictable behavior. However, most application traffic is better handled in the ZFW with the inspect action.
- **Inspect** The inspect action offers state-based traffic control. For example, if traffic from the private zone to the Internet zone in the earlier example network is inspected, the router maintains connection or session information for TCP and User Datagram Protocol (UDP) traffic. Therefore, the router

permits return traffic sent from Internet–zone hosts in reply to private zone connection requests. Also, inspect can provide application inspection and control for certain service protocols that might carry vulnerable or sensitive application traffic. Audit–trail can be applied with a parameter–map to record connection/session start, stop, duration, the data volume transferred, and source and destination addresses.

Actions are associated with class–maps in policy–maps:

```
conf t
policy-map type inspect z1-z2-pmap
class type inspect service-cmap
inspect|drop|allow [service-parameter-map]
```

Parameter–maps offer options to modify the connection parameters for a given class–map s inspection policy.

## Configuring Zone–Policy Firewall Parameter–Maps

Parameter–maps specify inspection behavior for ZFW, for parameters such as DoS protection, TCP connection/UDP session timers, and audit–trail logging settings. Parameter–maps are also applied with Layer 7 class and policy–maps to define application–specific behavior, such as HTTP objects, POP3 and IMAP authentication requirements, and other application–specific information.

Inspection parameter–maps for ZFW are configured as **type inspect**, similar to other ZFW class and policy–objects:

```
stg-871-L(config)#parameter-map type inspect z1-z2-pmap
stg-871-L(config-profile)#?
parameter-map commands:
  alert          Turn on/off alert
  audit-trail    Turn on/off audit trail
  dns-timeout    Specify timeout for DNS
  exit           Exit from parameter-map
  icmp          Config timeout values for icmp
  max-incomplete Specify maximum number of incomplete connections before
                clamping
  no            Negate or set default values of a command
  one-minute     Specify one-minute-sample watermarks for clamping
  sessions       Maximum number of inspect sessions
  tcp           Config timeout values for tcp connections
  udp           Config timeout values for udp flows
```

Specific types of parameter–maps specify parameters applied by Layer 7 application inspection policies. Regex–type parameter–maps define a regular expression for use with HTTP application inspection that filters traffic using a regular expression:

```
parameter-map type regex [parameter-map-name]
```

Protocol–info–type parameter–maps define server names for use with instant messaging application inspection:

```
parameter-map type protocol-info [parameter-map-name]
```

Complete configuration details for HTTP and IM application inspection are provided in the respective application inspection sections of this document.

Adjusting DoS protection is covered in a later section of this document.

Configuring application inspection is covered in a later section of this document.

## Applying Logging For Zone–Based Policy Firewall Policies

ZFW offers logging options for traffic that is dropped or inspected by default or configured firewall policy actions. Audit–trail logging is available for traffic that the ZFW inspects. Audit–trail is applied by defining audit–trail in a parameter–map and applying the parameter–map with the inspect action in a policy–map:

```
conf t
policy-map type inspect z1-z2-pmap
class type inspect service-cmap
inspect|drop|allow [parameter-map-name (optional)]
```

Drop logging is available for traffic that the ZFW drops. Drop logging is configured by adding log with the drop action in a policy–map:

```
conf t
policy-map type inspect z1-z2-pmap
class type inspect service-cmap
inspect|drop|allow [service-parameter-map]
```

## Editing Zone–Policy Firewall Class–Maps and Policy–Maps

ZFW does not presently incorporate an editor that can modify the various ZFW structures such as policy–maps, class–maps, and parameter–maps. In order to rearrange match statements in a class–map or action application to various class–maps contained within a policy–map, you need to complete these steps:

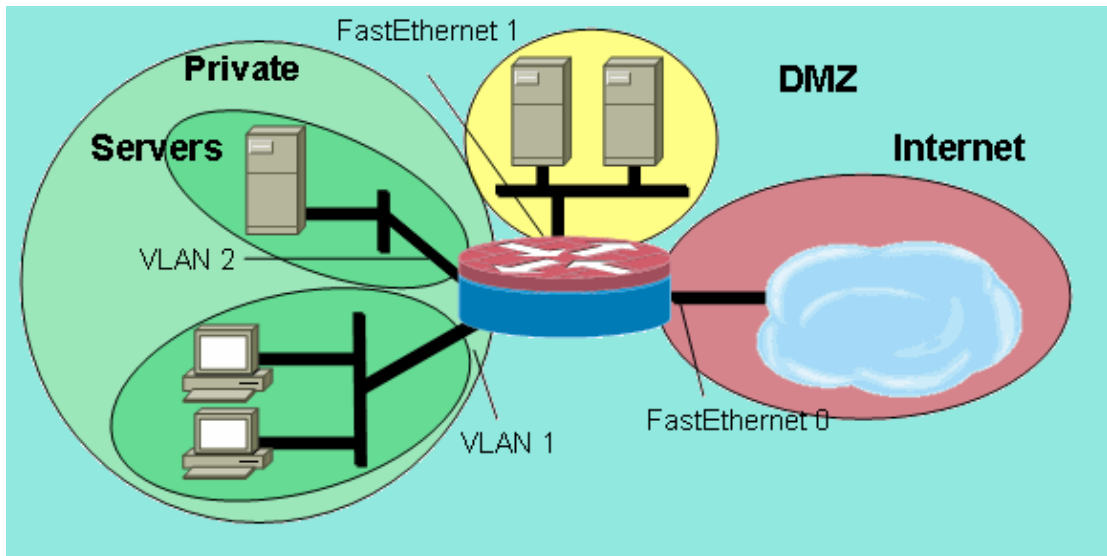
1. Copy the existing structure to a text editor such as Microsoft Windows Notepad, or an editor such as vi on Linux/Unix platforms.
2. Remove the existing structure from the router s configuration.
3. Edit the structure in your text editor.
4. Copy the structure back to the router s CLI.

## Configuration Examples

This configuration example employs a Cisco 1811 Integrated Services Router. A basic configuration with IP connectivity, VLAN configuration, and transparent bridging between two private Ethernet LAN segments is available in Appendix A. The router is separated into five zones:

- The public Internet is connected to FastEthernet 0 (Internet zone)
- Two Internet servers are connected to FastEthernet 1 (DMZ zone)
- The Ethernet switch is configured with two VLANs:
  - ◆ Workstations are connected to VLAN1 (client zone).
  - ◆ Servers are connected to VLAN2 (server zone).
  - ◆ The client and server zones are in the same subnet. A transparent firewall will be applied between the zones, so the inter–zone policies on those two interfaces will only affect traffic between the client and server zones.
- The VLAN1 and VLAN2 interfaces communicate with other networks through the bridge virtual interface (BVI1). This interface is assigned to the private zone. (See Figure 2.)

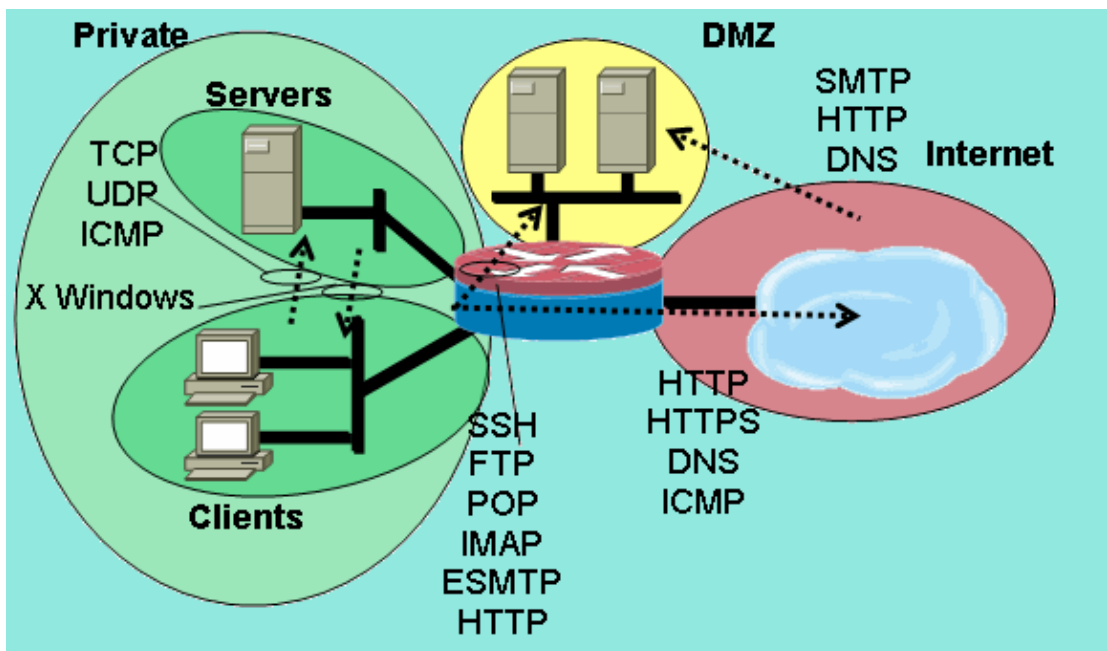
**Figure 2: Zone Topology Detail**



These policies are applied, using the network zones defined earlier:

- Hosts in Internet zone can reach DNS, SMTP, and SSH services on one server in the DMZ. The other server will offer SMTP, HTTP, and HTTPS services. The firewall policy will restrict access to the specific services available on each host.
- The DMZ hosts cannot connect to hosts in any other zone.
- Hosts in the client zone can connect to hosts in the server zone on all TCP, UDP, and ICMP services.
- Hosts in the server zone cannot connect to hosts in the client zone, except a UNIX–based application server can open X Windows client sessions to X Windows servers on desktop PCs in the client zone on ports 6900 to 6910.
- All hosts in the private zone (combination of clients and servers) can access hosts in the DMZ on SSH, FTP, POP, IMAP, ESMTP, and HTTP services, and in the Internet zone on HTTP, HTTPS, and DNS services and ICMP. Furthermore, application inspection will be applied on HTTP connections from the private zone to the Internet zone in order to assure that supported instant messaging and P2P applications are not carried on port 80. (See Figure 3.)

**Figure 3: Zone–Pair service permissions to be applied in the configuration example**



These firewall policies are configured in order of complexity:

1. Clients–Servers TCP/UDP/ICMP inspection
2. Private–DMZ SSH/FTP/POP/IMAP/ESMTP/HTTP inspection
3. Internet –DMZ SMTP/HTTP/DNS inspection restricted by host address
4. Servers–Clients X Windows inspection with a port–application mapping (PAM)–specified service
5. Private–Internet HTTP/HTTPS/DNS/ICMP with HTTP application inspection

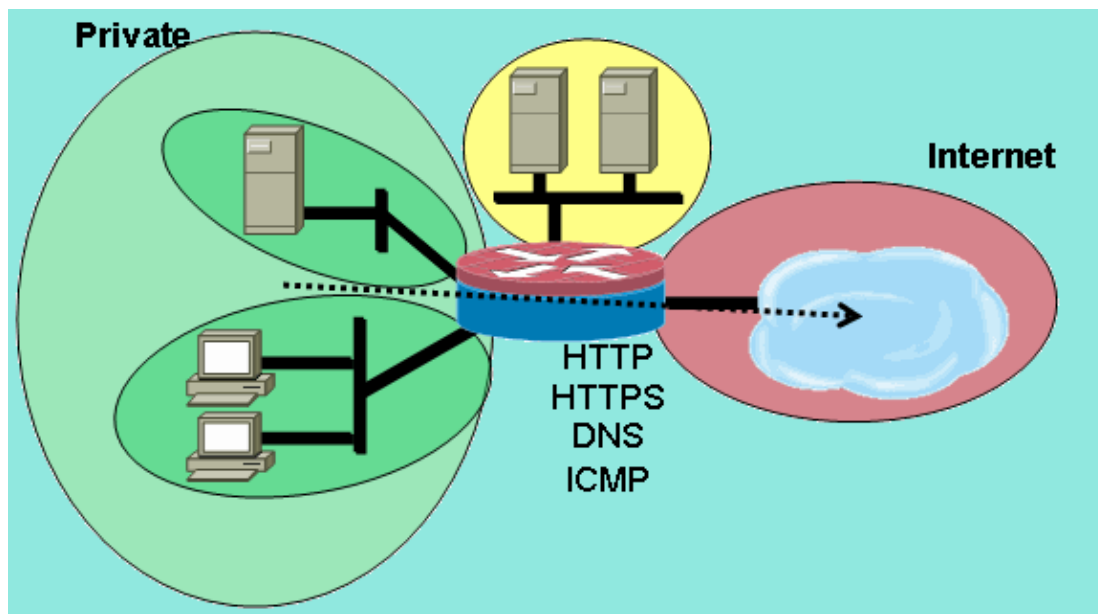
Because you will apply portions of the configuration to different network segments at different times, it is important to remember that a network segment will lose connectivity to other segments when it is placed in a zone. For instance, when the private zone is configured, hosts in the private zone will lose connectivity to the DMZ and Internet zones until their respective policies are defined.

## Stateful Inspection Routing Firewall

### Configure Private Internet Policy

Figure 4 illustrates the configuration of private Internet policy.

**Figure 4: Service inspection from private zone to internet zone**



The private Internet policy applies Layer 4 inspection to HTTP, HTTPS, DNS, and Layer 4 inspection for ICMP from the private zone to the Internet zone. This allows connections from the private zone to the Internet zone, and allows the return traffic. Layer 7 inspection carries the advantages of tighter application control, better security, and support for applications requiring fixup. However, Layer 7 inspection, as mentioned, requires a better understanding of network activity, as Layer 7 protocols that are not configured for inspection will not be allowed between zones.

1. Define class–maps that describe the traffic that you want to permit between zones, according to policies described earlier:

```
conf t
class-map type inspect match-any internet-traffic-class
  match protocol http
  match protocol https
  match protocol dns
  match protocol icmp
```

2. Configure a policy-map to inspect traffic on the class-maps you just defined:

```
conf t
  policy-map type inspect private-internet-policy
    class type inspect internet-traffic-class
    inspect
```

3. Configure the private and Internet zones and assign router interfaces to their respective zones:

```
conf t
  zone security private
  zone security internet
  int bv11
  zone-member security private
  int fastethernet 0
  zone-member security internet
```

4. Configure the zone-pair and apply the appropriate policy-map.

**Note:** You only need to configure the private Internet zone pair at present in order to inspect connections sourced in the private zone traveling to the Internet zone:

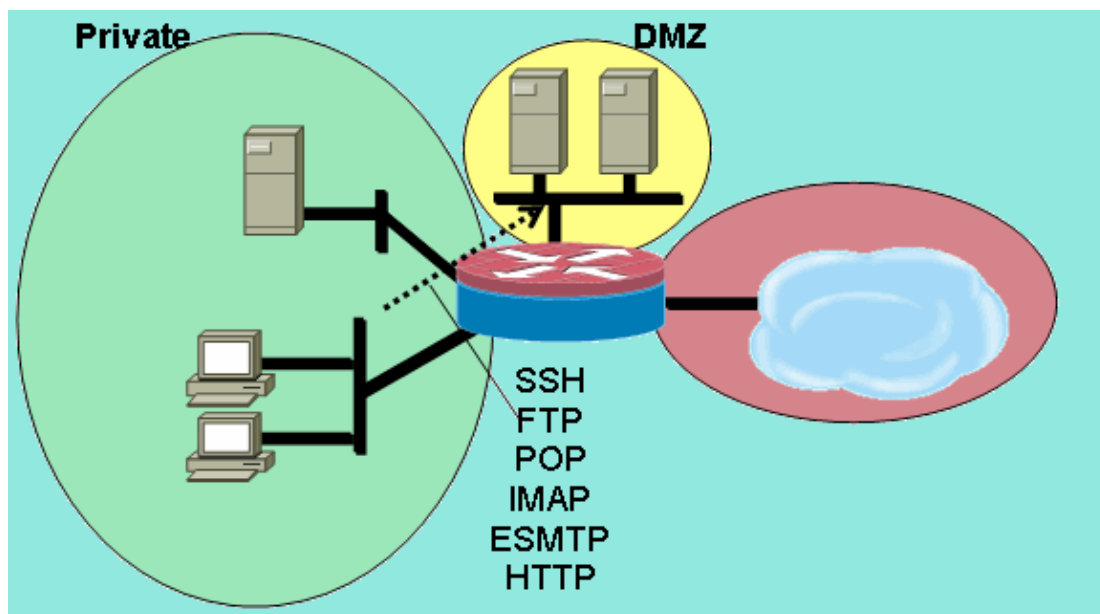
```
conf t
  zone-pair security private-internet source private destination internet
  service-policy type inspect private-internet-policy
```

This completes the configuration of the Layer 7 inspection policy on the private Internet zone-pair to allow HTTP, HTTPS, DNS, and ICMP connections from the clients zone to the servers zone and to apply application inspection to HTTP traffic to assure that unwanted traffic is not allowed to pass on TCP 80, HTTP's service port.

## Configure Private DMZ Policy

Figure 5 illustrates the configuration of private DMZ policy.

**Figure 5: Service inspection from private zone to DMZ zone**



The private DMZ policy adds complexity because it requires a better understanding of the network traffic between zones. This policy applies Layer 7 inspection from the private zone to the DMZ. This allows connections from the private zone to the DMZ, and allows the return traffic. Layer 7 inspection carries the

advantages of tighter application control, better security, and support for applications requiring fixup. However, Layer 7 inspection, as mentioned, requires a better understanding of network activity, as Layer 7 protocols that are not configured for inspection will not be allowed between zones.

1. Define class-maps that describe the traffic that you want to permit between zones, according to policies described earlier:

```
conf t
class-map type inspect match-any L7-inspect-class
  match protocol ssh
  match protocol ftp
  match protocol pop
  match protocol imap
  match protocol esmtp
  match protocol http
```

2. Configure policy-maps to inspect traffic on the class-maps you just defined:

```
conf t
policy-map type inspect private-dmz-policy
  class type inspect L7-inspect-class
  inspect
```

3. Configure the private and DMZ zones and assign router interfaces to their respective zones:

```
conf t
zone security private
zone security dmz
int bvil
zone-member security private
int fastethernet 1
zone-member security dmz
```

4. Configure the zone-pair and apply the appropriate policy-map.

**Note:** You only need to configure the private DMZ zone-pair at present in order to inspect connections sourced in the private zone traveling to the DMZ:

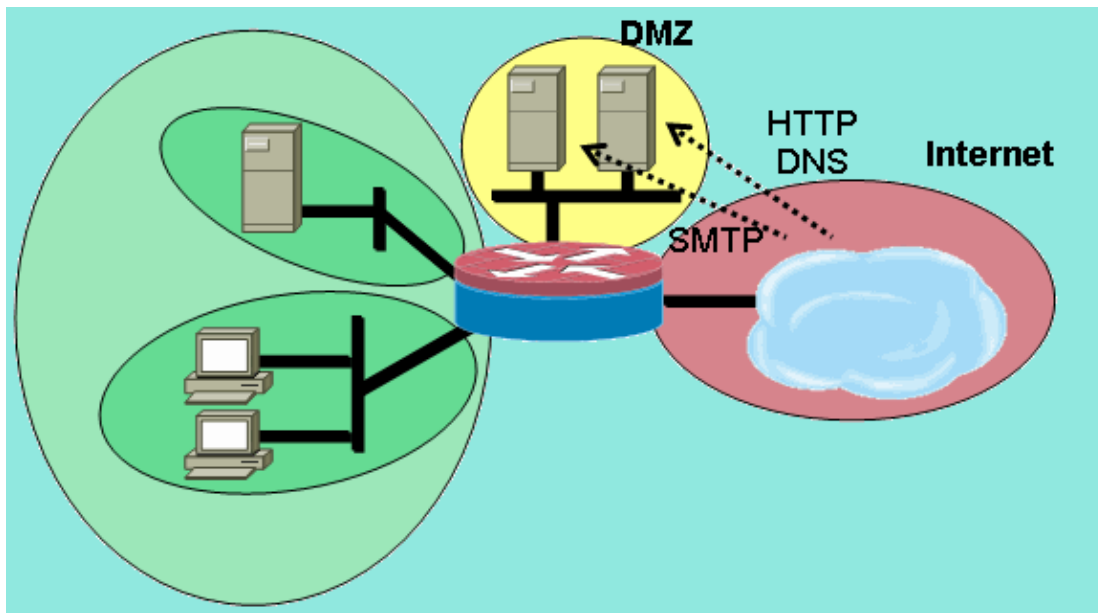
```
conf t
zone-pair security private-dmz source private destination dmz
service-policy type inspect private-dmz-policy
```

This completes the configuration of the Layer 7 inspection policy on the private DMZ to allow all TCP, UDP, and ICMP connections from the clients zone to the servers zone. The policy does not apply fixup for subordinate channels, but provides an example of simple policy to accommodate most application connections.

## Configure Internet DMZ Policy

Figure 6 illustrates the configuration of Internet DMZ policy.

### Figure 6: Service inspection from Internet zone to DMZ zone



This policy applies Layer 7 inspection from the Internet zone to the DMZ. This allows connections from the Internet zone to the DMZ, and allows the return traffic from the DMZ hosts to the Internet hosts that originated the connection. The Internet DMZ policy combines Layer 7 inspection with address groups defined by ACLs to restrict access to specific services on specific hosts, groups of hosts, or subnets. This is accomplished by nesting a class-map specifying services within another class-map referencing an ACL to specify IP addresses.

1. Define class-maps and ACLs that describe the traffic that you want to permit between zones, according to policies described earlier.

Multiple class-maps for services must be used, as differing access policies will be applied for access to two different servers. Internet hosts are allowed DNS and HTTP connections to 172.16.2.2, and SMTP connections are allowed to 172.16.2.3. Note the difference in the class-maps. The class-maps specifying services use the **match-any** keyword to allow any of the listed services. The class-maps associating ACLs with the service class-maps use the **match-all** keyword to require that both conditions in the class map must be met to allow traffic:

```
conf t
access-list 110 permit ip any host 172.16.2.2
access-list 111 permit ip any host 172.16.2.3
class-map type inspect match-any dns-http-class
  match protocol dns
  match protocol http
class-map type inspect match-any smtp-class
  match protocol smtp
class-map type inspect match-all dns-http-acl-class
  match access-group 110
  match class-map dns-http-class
class-map type inspect match-all smtp-acl-class
  match access-group 111
  match class-map smtp-class
```

2. Configure policy-maps to inspect traffic on the class-maps you just defined:

```
conf t
policy-map type inspect internet-dmz-policy
  class type inspect dns-http-acl-class
    inspect
  class type inspect smtp-acl-class
    inspect
```

3. Configure the Internet and DMZ zones and assign router interfaces to their respective zones. Skip the DMZ configuration if you set it up in the previous section:

```
conf t
zone security internet
zone security dmz
int fastethernet 0
 zone-member security internet
int fastethernet 1
 zone-member security dmz
```

4. Configure the zone-pair and apply the appropriate policy-map.

**Note:** You only need to configure the Internet DMZ zone pair at present, to inspect connections sourced in the Internet zone traveling to the DMZ zone:

```
conf t
zone-pair security internet-dmz source internet destination dmz
 service-policy type inspect internet-dmz-policy
```

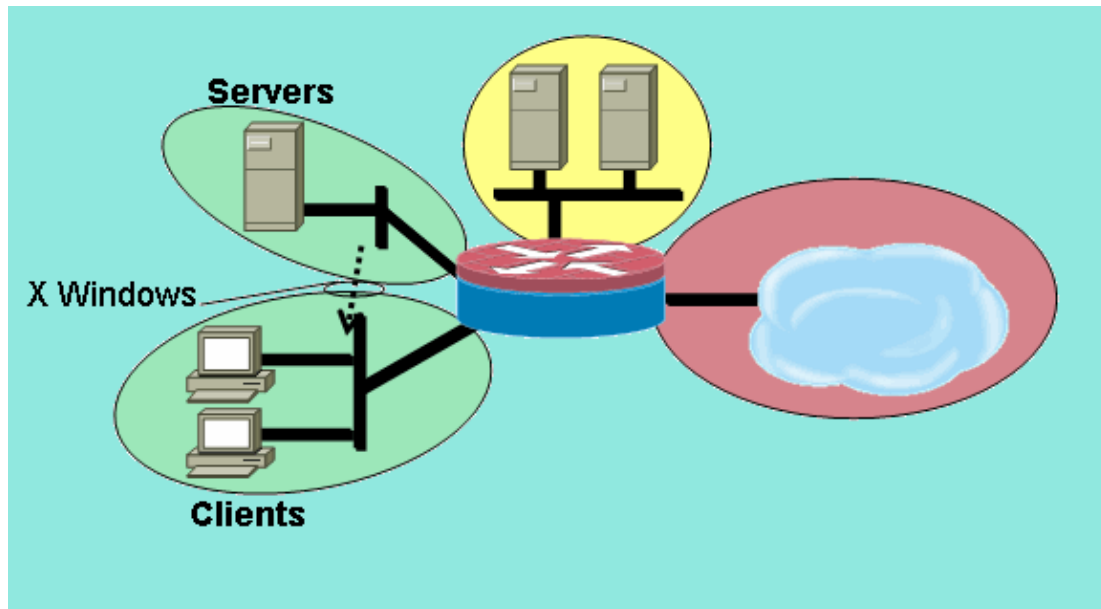
This completes the configuration of the address-specific Layer 7 inspection policy on the Internet DMZ zone-pair.

## Stateful Inspection Transparent Firewall

### Configure Servers-Clients Policy

Figure 7 illustrates the configuration of server-client policy.

**Figure 7: Service inspection from Servers zone to Clients zone**



The servers-clients policy applies inspection using a user-defined service. Layer 7 inspection is applied from the servers zone to the clients zone. This allows X Windows connections to a specific port range from the servers zone to the clients zone, and allows the return traffic. X Windows is not a natively supported protocol in PAM, so a user-configured service in PAM must be defined so the ZFW can recognize and inspect the appropriate traffic.

Two or more router interfaces are configured in an IEEE bridge-group to provide Integrated Routing and

Bridging (IRB) to provide bridging between the interfaces in the bridge-group and routing to other subnets via the Bridge Virtual Interface (BVI). The transparent firewall policy will offer apply firewall inspection for traffic crossing the bridge , but not for traffic that leaves the bridge-group via the BVI. The inspection policy only applies to traffic crossing the bridge-group. Therefore, in this scenario, the inspection will only be applied to traffic that moves between the clients and servers zones, which are nested inside the private zone. The policy applied between the private zone, and public and DMZ zones, only comes into play when traffic leaves the bridge-group via the BVI. When traffic leaves via the BVI from either the clients or servers zones, the transparent firewall policy will not be invoked.

1. Configure PAM with a user-defined entry for X Windows.

X Windows clients (where applications are hosted) open connections for display information to clients (where user is working) in a range starting at port 6900.

Each additional connection uses successive ports, so if a client displays 10 different sessions on one host, the server uses ports 6900–6909. Therefore, if you inspect the port range from 6900 to 6909, connections opened to ports beyond 6909 will fail:

```
conf t
ip port-map user-Xwindows port tcp from 6900 to 6910
```

2. Review PAM documents to address additional PAM questions or check granular protocol inspection documentation for information about the details of interoperability between PAM and Cisco IOS Firewall stateful inspection.
3. Define class-maps that describe the traffic that you want to permit between zones, according to policies described earlier:

```
conf t
class-map type inspect match-any Xwindows-class
match protocol user-Xwindows
```

4. Configure policy-maps to inspect traffic on the class-maps you just defined:

```
conf t
policy-map type inspect servers-clients-policy
class type inspect Xwindows-class
inspect
```

5. Configure the client and server zones and assign router interfaces to their respective zones.

If you configured these zones and assigned interfaces in the Clients–Servers Policy Configuration section, you can skip to the zone-pair definition. Bridging IRB configuration is provided for completeness:

```
conf t
bridge irb
bridge 1 protocol ieee
bridge 1 route ip
zone security clients
zone security servers
int vlan 1
bridge-group 1
zone-member security clients
int vlan 2
bridge-group 1
zone-member security servers
```

6. Configure the zone-pair and apply the appropriate policy-map.

**Note:** You only need to configure the servers-clients zone pair at present in order to inspect connections sourced in the servers zone traveling to the clients zone:

```

conf t
zone-pair security servers-clients source servers destination clients
service-policy type inspect servers-clients-policy

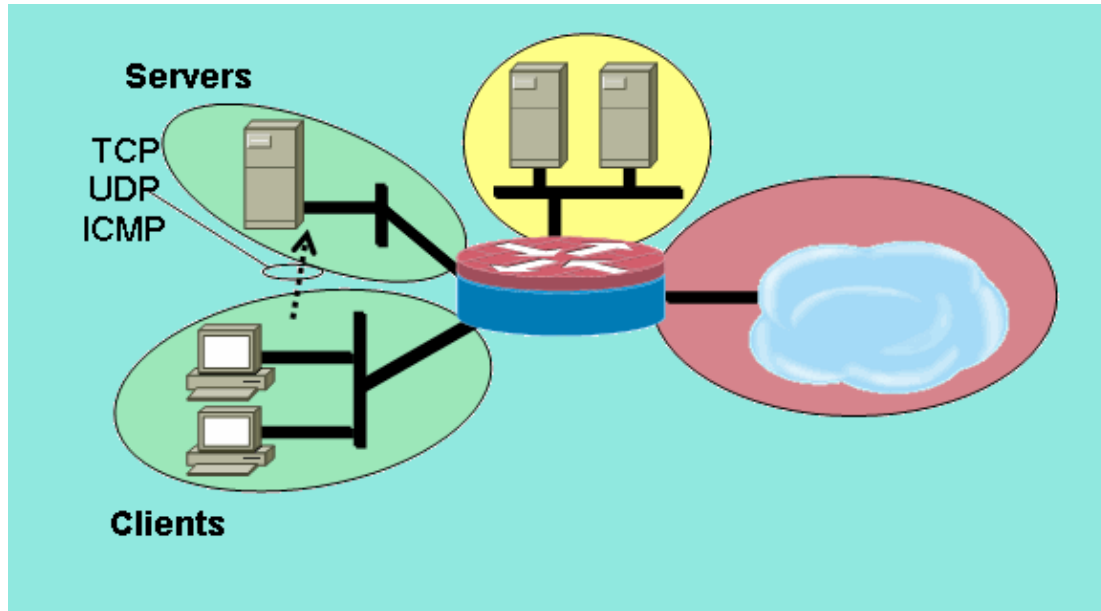
```

This completes the configuration of the user-defined inspection policy in the servers-clients zone-pair to allow X Windows connections from the server zone to the client zone.

## Configure Clients-Servers Policy

Figure 8 illustrates the configuration of client-server policy.

**Figure 8: Service inspection from Clients zone to Servers zone**



The client-servers policy is less complex than the others. Layer 4 inspection is applied from the clients zone to the servers zone. This allows connections from the clients zone to the servers zone, and allows return traffic. Layer 4 inspection carries the advantage of simplicity in the firewall configuration, in that only a few rules are required to allow most application traffic. However, Layer 4 inspection also carries two major disadvantages:

- Applications such as FTP or streaming media services frequently negotiate an additional subordinate channel from the server to the client. This functionality is usually accommodated in a service fixup that monitors the control channel dialog and allows the subordinate channel. This capability is not available in Layer 4 inspection.
- Layer 4 inspection allows nearly all application-layer traffic. If network use must be controlled so only a few applications are permitted through the firewall, an ACL must be configured on outbound traffic to limit the services allowed through the firewall.

Both router interfaces are configured in an IEEE bridge group, so this firewall policy will apply transparent firewall inspection. This policy is applied on two interfaces in an IEEE IP bridge group. The inspection policy only applies to traffic crossing the bridge group. This explains why the clients and servers zones are nested inside the private zone.

1. Define class-maps that describe the traffic that you want to permit between zones, according to policies described earlier:

```

conf t
class-map type inspect match-any L4-inspect-class

```

```
match protocol tcp
match protocol udp
match protocol icmp
```

2. Configure policy-maps to inspect traffic on the class-maps you just defined:

```
conf t
policy-map type inspect clients-servers-policy
class type inspect L4-inspect-class
inspect
```

3. Configure the clients and servers zones and assign router interfaces to their respective zones:

```
conf t
zone security clients
zone security servers
int vlan 1
zone-member security clients
int vlan 2
zone-member security servers
```

4. Configure the zone-pair and apply the appropriate policy-map.

**Note:** You only need to configure the clients-servers zone-pair at present, to inspect connections sourced in the clients zone traveling to the servers zone:

```
conf t
zone-pair security clients-servers source clients destination servers
service-policy type inspect clients-servers-policy
```

This completes the configuration of the Layer 4 inspection policy for the clients-servers zone-pair to allow all TCP, UDP, and ICMP connections from the client zone to the server zone. The policy does not apply fixup for subordinate channels, but provides an example of simple policy to accommodate most application connections.

## Rate Policing For Zone-Based Policy Firewall

Data networks frequently benefit with the ability to limit the transmission rate of specific types of network traffic, and to limit lower-priority traffic's impact to more business-essential traffic. Cisco IOS software offers this capability with traffic policing, which limits traffic's nominal rate and burst. Cisco IOS Software has supported traffic policing since Cisco IOS Release 12.1(5)T.

Cisco IOS Software Release 12.4(9)T augments ZFW with rate-limiting by adding the capability to police traffic matching the definitions of a specific class-map as it traverses the firewall from one security zone to another. This provides the convenience of offering one configuration point to describe specific traffic, apply firewall policy, and police that traffic's bandwidth consumption. ZFW policing differs from interface-based policing in that it only provides the actions transmit for policy conformance and drop for policy violation. ZFW policing cannot mark traffic for DSCP.

ZFW policing can only specify bandwidth use in bytes/second, packet/second and bandwidth percentage policing are not offered. ZFW policing can be applied with or without interface-based policing. Therefore, if additional policing capabilities are required, these features can be applied by interface-based policing. If interface-based policing is used in conjunction with firewall policing, make certain that the policies do not conflict.

### Configuring ZFW Policing

ZFW policing limits traffic in a policy-map's class-map to a user-defined rate value between 8,000 and 2,000,000,000 bits per second, with a configurable burst value in the range of 1,000 to 512,000,000 bytes.

ZFW policing is configured by an additional line of configuration in the policy-map, which is applied after the policy action:

```
policy-map type inspect private-allowed-policy
  class type inspect http-class
  inspect
  police rate [bps rate value <8000-2000000000>] burst [value in bytes <1000-512000000>]
```

## Session Control

ZFW policing also introduced session control to limit the session count for traffic in a policy-map matching a class-map. This adds to the existing capability to apply DoS protection policy per class-map. Effectively, this allows granular control on the number of sessions matching any given class-map that cross a zone-pair. If the same class-map is used on multiple policy-maps or zone-pairs, different session limits can be applied on the various class-map applications.

Session control is applied by configuring a parameter-map that contains the desired session volume, then appending the parameter-map to the inspection action applied to a class-map under a policy-map:

```
parameter-map type inspect my-parameters
  sessions maximum [1-2147483647]

policy-map type inspect private-allowed-policy
  class type inspect http-class
  inspect my-parameters
```

Parameter-maps can only be applied to the inspect action, and are not available on pass or drop actions.

ZFW's session control and policing activities are visible with this command

```
show policy-map type inspect zone-pair
```

## Application Inspection

Application inspection introduces additional capability to ZFW. Application inspection policies are applied at Layer 7 of the OSI model, where user applications send and receive messages that allow the applications to offer useful capabilities. Some applications might offer undesired or vulnerable capabilities, so the messages associated with these capabilities must be filtered to limit activities on the application services.

Cisco IOS Software ZFW offers application inspection and control on these application services:

- HTTP
- SMTP
- POP3
- IMAP
- Sun RPC
- P2P Application Traffic
- IM Applications

Application inspection and control (AIC) varies in capability per service. HTTP inspection offers granular filtering on several types of application activity, offering capabilities to limit transfer size, web address lengths, and browser activity to enforce compliance with application-behavior standards and to limit types of content that are transferred over the service. AIC for SMTP can limit content length and enforce protocol compliance. POP3 and IMAP inspection can help ensure that users are using secure authentication mechanisms to prevent compromise of user credentials.

Application inspection is configured as an additional set of application-specific class-maps and policy-maps, which are then applied to existing inspection class-maps and policy-maps by defining the application service policy in the inspection policy-map.

## HTTP Application Inspection

Application inspection can be applied on HTTP traffic to control unwanted use of HTTP's service port for other applications such as IM, P2P file sharing, and tunneling applications that can redirect otherwise firewalled applications through TCP 80.

Configure an application inspection class-map to describe traffic that violates allowed HTTP traffic:

```
! configure the actions that are not permitted
class-map type inspect http match-any http-aic-cmap
  match request port-misuse any
  match req-resp protocol-violation
! define actions to be applied to unwanted traffic
policy-map type inspect http http-aic-pmap
  class type insp http http-aic-cmap
    reset
    log
! define class-map for stateful http inspection
class-map type inspect match-any http-cmap
  match protocol http
! define class-map for stateful inspection for other traffic
class-map type inspect match-any other-traffic-cmap
  match protocol smtp
  match protocol dns
  match protocol ftp
! define policy-map, associate class-maps and actions
policy-map type inspect priv-pub-pmap
  class type inspect http-cmap
    inspect
  service-policy http http-aic-pmap
  class type inspect other-traffic-cmap
    inspect
```

## HTTP Application Inspection Improvements

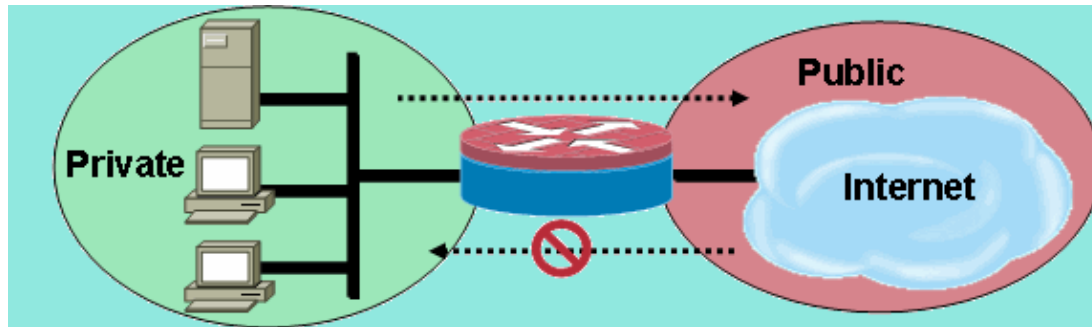
Cisco IOS Software Release 12.4(9)T introduces improvements to ZFW's HTTP inspection capabilities. Cisco IOS Firewall introduced HTTP Application Inspection in Cisco IOS Software Release 12.3(14)T. Cisco IOS Software Release 12.4(9)T augments existing capabilities by adding:

- Ability to permit, deny, and monitor requests and responses based on header name and header values. This is useful to block requests and responses that carry vulnerable header fields.
- Ability to limit the sizes of different elements in the HTTP request and response headers such as maximum URL length, maximum header length, maximum number of headers, maximum header-line length, etc. This is useful to prevent buffer overflows.
- Ability to block requests and responses that carry multiple headers of the same type; for instance, a request with two content-length headers.
- Ability to block requests and responses with non-ASCII headers. This is useful to prevent various attacks that use binary and other non-ASCII characters to deliver worms and other malicious contents to web servers.
- Ability to group HTTP methods into user-specified categories and flexibility to block/allow/monitor each of the group is offered. The HTTP RFC allows a restricted set of HTTP methods. Some of the standard methods are considered unsafe because they can be used to exploit vulnerabilities on a web server. Many of the non-standard methods have a bad security record.
- Method to block specific URIs based on a user configured regular expression. This feature gives a

user the capability to block custom URIs and queries.

- Ability to spoof header types (especially server header type) with user customizable strings. This is useful in a case where an attacker analyzes web server responses and learns as much information as possible, then launches an attack that exploits weaknesses in that particular web server.
- Ability to block or issue an alert on an HTTP connection if one or more HTTP parameter values match values entered by the user as a regular expression. Some of the possible HTTP value contexts include header, body, username, password, user agent, request line, status line, and decoded CGI variables.

Configuration examples for HTTP application inspection improvements assume a simple network:



The firewall groups traffic into two classes:

- HTTP traffic
- All other single-channel TCP, UDP, and ICMP traffic

HTTP is separated to allow specific inspection on web traffic. This allows you to configure policing in the first section of this document, and HTTP Application Inspection in the second section. You will configure specific class-maps and policy-maps for P2P and IM traffic in the third section of this document.

Connectivity is allowed from the private zone to the public zone. No connectivity is provided from the public zone to the private zone.

A complete configuration implementing the initial policy is provided in Appendix C, Basic Zone-Policy Firewall Configuration For Two Zones.

## Configuring HTTP Application Inspection Enhancements

HTTP Application Inspection (as well as other application inspection policies) requires more complex configuration than basic Layer 4 configuration. You must configure Layer 7 traffic classification and policy to recognize specific traffic that you wish to control, and to apply the desired action to desirable and undesirable traffic.

HTTP Application Inspection (similar to other types of Application Inspection) can only be applied to HTTP traffic. Thus, you must define Layer 7 class-maps and policy-maps for specific HTTP traffic, then define a Layer-4 class-map specifically for HTTP, and apply the Layer-7 policy to HTTP inspection in a Layer-4 policy-map, as such:

```
!configure the layer-7 traffic characteristics:
class-map type inspect http match-any http-l7-cmap
  match req-resp protocol-violation
  match request body length gt 4096
!
!configure the action to be applied to the traffic
!matching the specific characteristics:
policy-map type inspect http http-l7-pmap
  class type inspect http http-l7-cmap
```

```

    reset
    log
!
!define the layer-4 inspection policy
class-map type inspect match-all http-l4-cmap
  match protocol http
!
!associate layer-4 class and layer-7 policy-map
!in the layer-4 policy-map:
policy-map type inspect private-allowed-policy
  class type inspect http-l4-cmap
    inspect
  service-policy http http-l7-pmap

```

All of these HTTP Application Inspection traffic characteristics are defined in a Layer 7 class-map:

- **Header inspection** This command provides the ability to permit/deny/monitor requests or responses whose header matches the configured regular expression. Allow or reset action can be applied to a request or response matching the class-map criteria. Addition of the log action causes a syslog message:

```
APPFW-6-HTTP_HDR_REGEX_MATCHED
```

#### *Command usage:*

```
match {request|response|req-resp} header regex <parameter-map-name>
```

#### *Sample Use Case*

Configure an http appfw policy to block request or response whose header contains non-ASCII characters.

```

parameter-map type regex non_ascii_regex
  pattern [^\x00-\x80]
class-map type inspect http non_ascii_cm
  match req-resp header regex non_ascii_regex
policy-map type inspect http non_ascii_pm
  class type inspect http non_ascii_cm
  reset

```

- **Header length inspection** This command checks the length of a request or response header and applies action if length exceeds the configured threshold. Action is allow or reset. Addition of the log action causes a syslog message:

```
APPFW-4- HTTP_HEADER_LENGTH
```

#### *Command usage:*

```
match {request|response|req-resp} header length gt <bytes>
```

#### *Sample Use Case*

Configure an http appfw policy to block requests and responses having header length greater than 4096 bytes.

```

class-map type inspect http hdr_len_cm
  match req-resp header length gt 4096

policy-map type inspect http hdr_len_pm
  class type inspect http hdr_len_cm
  reset

```

- **Header count inspection** This command verifies the number of header-lines (fields) in a request/response and applies action when the count exceeds configured threshold. Action is allow or reset. Addition of the log action causes a syslog message:

```
APPFW-6- HTTP_HEADER_COUNT.
```

*Command usage:*

```
match {request|response|req-resp} header count gt <number>
```

*Sample Use Case*

Configure an http appfw policy to block a request that has more than 16 header fields.

```
class-map type inspect http_hdr_cnt_cm
  match request header count gt 16

policy-map type inspect http_hdr_cnt_pm
  class type inspect http_hdr_cnt_cm
  reset
```

- **Header field inspection** This command provides the ability to permit/deny/monitor requests/responses that contain a specific HTTP header field and value. Allow or reset action can be applied to a request or response matching the class-map criteria. The addition of the log action causes a syslog message:

```
APPFW-6- HTTP_HDR_FIELD_REGEX_MATCHED
```

*Command usage:*

```
match {request|response|req-resp} header <header-name>
```

*Sample Use Case*

Configure an http application inspection policy to block spyware/adware:

```
parameter-map type regex ref_regex
  pattern \.delfinproject\.com
  pattern \.looksmart\.com

parameter-map type regex host_regex
  pattern secure\.keenvalue\.com
  pattern \.looksmart\.com

parameter-map type regex usragnt_regex
  pattern Peer Points Manager

class-map type inspect http_spy_adwr_cm
  match request header refer regex ref_regex
  match request header host regex host_regex
  match request header user-agent regex usragnt_regex

policy-map type inspect http_spy_adwr_pm
  class type inspect http_spy_adwr_cm
  reset
```

- **Header field length inspection** This command provides an ability to limit the length of a header field line. Allow or reset action can be applied to a request or response matching the class-map criteria. The addition of the log action causes a syslog message:

```
APPFW-6- HTTP_HDR_FIELD_LENGTH.
```

### Command usage:

```
match {request|response|req-resp} header <header-name> length gt <bytes>
```

### Sample Use Case

Configure an http appfw policy to block a request whose cookie and user-agent field length exceeds 256 and 128 respectively.

```
class-map type inspect http hdrline_len_cm
  match request header cookie length gt 256
  match request header user-agent length gt 128

policy-map type inspect http hdrline_len_pm
  class type inspect http hdrline_len_cm
  reset
```

- **Inspection of header field repetition** This command checks if a request or response has repeated header fields. Allow or reset action may be applied to a request or response matching the class-map criteria. When enabled, the log action causes a syslog message:

```
APPFW-6- HTTP_REPEATED_HDR_FIELDS.
```

### Command usage:

```
match {request|response|req-resp} header <header-name>
```

### Sample Use Case

Configure an http appfw policy to block a request or response that has multiple content-length header lines. This is one of the most useful functionalities used to prevent session smuggling.

```
class-map type inspect http multi_occrrns_cm
  match req-resp header content-length count gt 1

policy-map type inspect http multi_occrrns_pm
  class type inspect http multi_occrrns_cm
  reset
```

- **Method inspection** The HTTP RFC allows a restricted set of HTTP methods. However, even some of the standard methods are considered unsafe as some methods can be used to exploit vulnerabilities on a web server. Many of the non-standard methods are used frequently for malicious activity. This necessitates a need to group the methods into various categories and have the user choose the action for each category. This command provides the user a flexible way of grouping the methods into various categories such as safe methods, unsafe methods, webdav methods, rfc methods, and extended methods. Allow or reset action can be applied to a request or response that matches the class-map criteria. Addition of the log action causes a syslog message:

```
APPFW-6-HTTP_METHOD.
```

### Command usage:

```
match request method <method>
```

### Sample Use Case

Configure an http appfw policy that groups the HTTP methods into three categories: safe, unsafe and webdav. These are shown in the table. Configure actions such that:

- ◆ all safe methods are allowed without log
- ◆ all unsafe methods are allowed with log
- ◆ all webdav methods are blocked with log.

Safe	Unsafe	WebDAV
GET, HEAD, OPTION	POST, PUT, CONNECT, TRACE	BCOPY, BDELETE, BMOVE

http policy:

```
class-map type inspect http safe_methods_cm
  match request method get
  match request method head
  match request method option

class-map type inspect http unsafe_methods_cm
  match request method post
  match request method put
  match request method connect
  match request method trace

class-map type inspect http webdav_methods_cm
  match request method bcopy
  match request method bdelete
  match request method bmove

policy-map type inspect http methods_pm
  class type inspect http safe_methods_cm
    allow
  class type inspect http unsafe_methods_cm
    allow log
  class type inspect http webdav_methods_cm
    reset log
```

- **URI inspection** This command provides the ability to permit/deny/monitor requests whose URI matches configured regular inspection. This gives the user a capability to block custom URLs and queries. Allow or reset action can be applied to a request or response matching the class-map criteria. Addition of the log action causes a syslog message:

```
APPFW-6- HTTP_URI_REGEX_MATCHED
```

*Command usage:*

```
match request uri regex <parameter-map-name>
```

*Sample Use Case*

Configure an http appfw policy to block a request whose URI matches any of these regular expressions:

- ◆ .\*cmd.exe
  - ◆ .\*sex
  - ◆ .\*gambling
- ```
parameter-map type regex uri_regex_cm
  pattern .*cmd.exe
  pattern .*sex
  pattern .*gambling

class-map type inspect http uri_check_cm
  match request uri regex uri_regex_cm

policy-map type inspect http uri_check_pm
```

```
class type inspect http uri_check_cm
  reset
```

- **URI length inspection** This command verifies the length of the URI being sent in a request and applies the configured action when length exceeds configured threshold. Allow or reset action can be applied to a request or response matching the class-map criteria. Addition of the log action causes a syslog message:

```
APPFW-6- HTTP_URI_LENGTH.
```

#### *Command usage:*

```
match request uri length gt <bytes>
```

#### *Sample Use Case*

Configure an http appfw policy to raise an alarm whenever URI length of a request exceeds 3076 bytes.

```
class-map type inspect http uri_len_cm
  match request uri length gt 3076

policy-map type inspect http uri_len_pm
  class type inspect http uri_len_cm
  log
```

- **Argument inspection** This command provides an ability to permit, deny or monitor request whose arguments (parameters) match configured regular inspection. Allow or reset action can be applied to a request or response matching the class-map criteria. Addition of the log action causes a syslog message:

```
APPFW-6- HTTP_ARG_REGEX_MATCHED
```

#### *Command usage:*

```
match request arg regex <parameter-map-name>
```

#### *Sample Use Case*

Configure an http appfw policy to block a request whose arguments match any of these regular expressions:

- ◆ `.*codeder`

- ◆ `.*attack`

```
parameter-map type regex arg_regex_cm
  pattern .*codeder
  pattern .*attack

class-map type inspect http arg_check_cm
  match request arg regex arg_regex_cm

policy-map type inspect http arg_check_pm
  class type inspect http arg_check_cm
  reset
```

- **Argument length inspection** This command verifies the length of the arguments being sent in a request and applies the configured action when length exceeds configured threshold. Allow or reset action can be applied to a request or response matching the class-map criteria. Addition of the log action causes a syslog message:

```
APPFW-6- HTTP_ARG_LENGTH.
```

### Command usage:

```
match request arg length gt <bytes>
```

### Sample Use Case

Configure an http appfw policy to raise an alarm whenever argument length of a request exceeds 512 bytes.

```
class-map type inspect http arg_len_cm
  match request arg length gt 512

policy-map type inspect http arg_len_pm
  class type inspect http arg_len_cm
  log
```

- **Body inspection** This CLI allows the user to specify list of regular expressions to be matched against body of the request or response. Allow or reset action can be applied to a request or response matching the class-map criteria. Addition of the log action causes a syslog message:

```
APPPFW-6- HTTP_BODY_REGEX_MATCHED
```

### Command usage:

```
match {request|response|req-resp} body regex <parameter-map-name>
```

### Sample Use Case

Configure an http appfw to block a response whose body contains the pattern `.*[Aa][Tt][Tt][Aa][Cc][Kk]`

```
parameter-map type regex body_regex
  pattern .*[Aa][Tt][Tt][Aa][Cc][Kk]

class-map type inspect http body_match_cm
  match response body regex body_regex

policy-map type inspect http body_match_pm
  class type inspect http body_match_cm
  reset
```

- **Body (Content) length inspection** This command verifies size of the message being sent through request or response. Allow or reset action can be applied to a request or response matching the class-map criteria. Addition of the log action causes a syslog message:

```
APPPFW-4- HTTP_CONTENT_LENGTH
```

### Command usage:

```
match {request|response|req-resp} body length lt <bytes> gt <bytes>
```

### Sample Use Case

Configure an http appfw policy to block an http session that carries more than 10K bytes message in a request or response.

```
class-map type inspect http cont_len_cm
  match req-resp header content-length gt 10240

policy-map type inspect http cont_len_pm
  class type inspect http cont_len_cm
```

reset

- **Status line inspection** The command allows the user to specify list of regular expressions to be matched against status-line of a response. Allow or reset action can be applied to a request or response matching the class-map criteria. Addition of the log action causes a syslog message:

```
APPFW-6-HTTP_STLINE_REGEX_MATCHED.
```

*Command usage:*

```
match response status-line regex <class-map-name>
```

*Sample Use Case*

Configure an http appfw to log an alarm whenever an attempt is made to access a forbidden page. A forbidden page usually contains a 403 status-code and the status line looks like HTTP/1.0 403 page forbidden\r\n.

```
parameter-map type regex status_line_regex
  pattern [Hh][Tt][Tt][Pp][/] [0-9][.][0-9][ \t]+403

class-map type inspect http status_line_cm
  match response status-line regex status_line_regex

policy-map type inspect http status_line_pm
  class type inspect http status_line_cm
  log
```

- **Content-type inspection** This command verifies if the message header's content-type is in the list of the supported content types. It also verifies that the header's content-type matches the content of the message data or entity body portion. If the keyword **mismatch** is configured, the command verifies the content-type of the response message against the accepted field value of the request message. Allow or reset action can be applied to a request or response matching the class-map criteria. Addition of the log action causes the appropriate syslog message:

```
APPFW-4- HTTP_CONT_TYPE_VIOLATION,
APPFW-4- HTTP_CONT_TYPE_MISMATCH,
APPFW-4- HTTP_CONT_TYPE_UNKNOWN
```

*Command usage:*

```
match {request|response|req-resp} header content-type [mismatch|unknown|violation]
```

*Sample Use Case*

Configure an http appfw policy to block an http session that carries requests and responses having unknown content-type.

```
class-map type inspect http cont_type_cm
  match req-resp header content-type unknown

policy-map type inspect http cont_type_pm
  class type inspect http cont_type_cm
  reset
```

- **Port-misuse inspection** This command is used to prevent http port (80) being misused for other applications such as IM, P2P, Tunneling, etc. Allow or reset action can be applied to a request or response matching the class-map criteria. Addition of the log action causes the appropriate syslog message:

```
APPFW-4- HTTP_PORT_MISUSE_TYPE_IM
APPFW-4-HTTP_PORT_MISUSE_TYPE_P2P
```

*Command usage:*

```
match request port-misuse {im|p2p|tunneling|any}
```

*Sample Use Case*

Configure an http appfw policy to block an http session being misused for IM application.

```
class-map type inspect http port_misuse_cm
  match request port-misuse im

policy-map type inspect http port_misuse_pm
  class type inspect http port_misuse_cm
  reset
```

- **Strict-http inspection** This command enables strict protocol conformance check against HTTP requests and responses. Allow or reset action can be applied to a request or response matching the class-map criteria. Addition of the log action causes a syslog message:

APPFW-4- HTTP\_PROTOCOL\_VIOLATION

*Command usage:*

```
match req-resp protocol-violation
```

*Sample Use Case*

Configure an http appfw policy to block requests or responses violating RFC 2616:

```
class-map type inspect http proto-viol_cm
  match req-resp protocol-violation

policy-map type inspect http proto-viol_pm
  class type inspect http proto-viol_cm
  reset
```

- **Transfer-Encoding inspection** This command provides an ability to permit, deny or monitor request/response whose transfer encoding type matches with configured type. Allow or reset action can be applied to a request or response matching the class-map criteria. Addition of the log action causes a syslog message:

APPFW-6- HTTP\_TRANSFER\_ENCODING

*Command usage:*

```
match {request|response|req-resp} header transfer-encoding
  {regex <parameter-map-name> |gzip|deflate|chunked|identity|all}
```

*Sample Use Case*

Configure an http appfw policy to block a request or response that has compress type encoding.

```
class-map type inspect http trans_encoding_cm
  match req-resp header transfer-encoding type compress

policy-map type inspect http trans_encoding_pm
  class type inspect http trans_encoding_cm
  reset
```

- **Java Applet inspection** This command checks if a response has Java applet and applies the configured action upon detection of applet. Allow or reset action can be applied to a request or response matching the class-map criteria. Addition of the log action causes a syslog message:

```
APPFW-4- HTTP_JAVA_APPLET
```

*Command usage:*

```
match response body java-applet
```

*Sample Use Case*

Configure an http appfw policy to block java applets.

```
class-map type inspect http java_applet_cm
  match response body java-applet

policy-map type inspect http java_applet_pm
  class type inspect http java_applet_cm
  reset
```

## ZFW Support for Instant Messaging and Peer-to-Peer Application Control

Cisco IOS Software Release 12.4(9)T introduced ZFW support for IM and P2P Applications.

Cisco IOS Software first offered support for IM application control in Cisco IOS Software Release 12.4(4)T. The initial release of ZFW did not support IM Application in the ZFW interface. If IM application control was desired, users were unable to migrate to the ZFW configuration interface. Cisco IOS Software Release 12.4(9)T introduces ZFW support for IM Inspection, supporting Yahoo! Messenger (YM), MSN Messenger (MSN), and AOL Instant Messenger (AIM).

Cisco IOS Software Release 12.4(9)T is the first version of Cisco IOS Software that offers native IOS Firewall support for P2P file sharing applications.

Both IM and P2P inspection offer Layer 4 and Layer 7 policies for application traffic. This means ZFW can provide basic stateful inspection to permit permit or deny the traffic, as well as granular Layer 7 control on specific activities in the various protocols, so that certain application activities are allowed while others are denied.

## P2P Application Inspection and Control

SDM 2.2 introduced P2P Application control in its Firewall configuration section. SDM applied a Network-Based Application Recognition (NBAR) and QoS policy to detect and police P2P application activity to a line rate of zero, blocking all P2P traffic. This raised the issue that CLI users, expecting P2P support in the IOS Firewall CLI, were unable to configure P2P blocking in the CLI unless they were aware of the necessary NBAR/QoS configuration. Cisco IOS Software Release 12.4(9)T introduces native P2P control in the ZFW CLI, leveraging NBAR to detect P2P application activity. This software release supports several P2P application protocols:

- BitTorrent
- eDonkey
- FastTrack
- Gnutella
- KaZaA / KaZaA2
- WinMX

P2P applications are particularly difficult to detect, as a result of port-hopping behavior and other tricks to avoid detection, as well as problems introduced by frequent changes and updates to P2P applications which modify the protocols behaviors. ZFW combines native firewall stateful inspection with NBAR s traffic-recognition capabilities to deliver P2P application control in ZFW s CPL configuration interface. NBAR offers two excellent benefits:

- Optional heuristic-based application recognition to recognize applications in spite of complex, difficult-to-detect behavior
- Extensible infrastructure offering an update mechanism to stay abreast of protocol updates and modifications

## Configuring P2P Inspection

As mentioned earlier, P2P inspection and control offers both Layer 4 Stateful Inspection and Layer 7 Application Control.

Layer 4 inspection is configured similarly to other application services, if inspection of the native application service ports will be adequate:

```
class-map type inspect match-any my-p2p-class
match protocol [bittorrent | edonkey | fasttrack | gnutella | kazaa | kazaa2 | winmx ]
[signature (optional)]
!
policy-map type inspect private-allowed-policy
class type inspect my-p2p-class
[drop | inspect | pass]
```

Notice the additional signature option in the match protocol [service-name]. By adding the signature option at the end of the match protocol statement, NBAR heuristics are applied to the traffic to search for telltales in traffic that indicate specific P2P application activity. This includes port-hopping and other changes in application behavior to avoid traffic detection. This level of traffic inspection comes at the price of increased CPU utilization and reduced network throughput capability. If the signature option is not applied, NBAR-based heuristic analysis will not be applied to detect port-hopping behavior, and CPU utilization will not be impacted to the same extent.

Native service inspection carries the disadvantage that it is unable to maintain control over P2P applications in the event that the application hops to a non-standard source and destination port, or if the application is updated to begin its action on an unrecognized port number:

| Application | Native Ports (as recognized by 12.4(15)T PAM list) |
|-------------|----------------------------------------------------|
| bittorrent  | TCP 6881-6889                                      |
| edonkey     | TCP 4662                                           |
| fasttrack   | TCP 1214                                           |
| gnutella    | TCP 6346-6349<br>TCP 6355,5634<br>UDP 6346-6348    |
| kazaa2      | Dependent on PAM                                   |
| winmx       | TCP 6699                                           |

If you wish to allow (inspect) P2P traffic, you might need to provide additional configuration. Some applications might use multiple P2P networks, or implement specific behaviors that you might need to accommodate in your firewall configuration to allow the application to work:

- BitTorrent clients usually communicate with trackers (peer directory servers) via http running on some non-standard port. This is typically TCP 6969, but you might need to check the torrent-specific tracker port. If you wish to allow BitTorrent, the best method to accommodate the additional port is to configure HTTP as one of the match protocols and add TCP 6969 to HTTP using the **ip port-map** command:

```
ip port-map http port tcp 6969
```

You will need to define http and bittorrent as the match criteria applied in the class-map.

- eDonkey appears to initiate connections that are detected as both eDonkey and Gnutella.
- KaZaA inspection is entirely dependent on NBAR-signature detection.

Layer 7 (Application) Inspection augments Layer 4 Inspection with the capability to recognize and apply service-specific actions, such as selectively blocking or allowing file-search, file-transfer, and text-chat capabilities. Service-specific capabilities vary by service.

P2P Application Inspection is similar to HTTP Application Inspection:

```
!configure the layer-7 traffic characteristics:
class-map type inspect [p2p protocol] match-any p2p-l7-cmap
  match action
!
!configure the action to be applied to the traffic
!matching the specific characteristics:
policy-map type inspect [p2p protocol] p2p-l7-pmap
  class type inspect p2p p2p-l7-cmap
    [ reset | allow ]
  log
!
!define the layer-4 inspection policy
class-map type inspect match-all p2p-l4-cmap
  match protocol [p2p protocol]
!
!associate layer-4 class and layer-7 policy-map
!in the layer-4 policy-map:
policy-map type inspect private-allowed-policy
  class type inspect p2p-l4-cmap
    [ inspect | drop | pass ]
  service-policy p2p p2p-l7-pmap
```

P2P Application Inspection offers application-specific capabilities for a subset of the applications supported by Layer 4 Inspection:

- edonkey
- fasttrack
- gnutella
- kaza2

Each of these applications offers varying application-specific match criteria options:

*edonkey*

```
router(config)#class-map type inspect edonkey match-any edonkey-l7-cmap
router(config-cmap)#match ?
  file-transfer      Match file transfer stream
```

```
flow                Flow based QoS parameters
search-file-name    Match file name
text-chat           Match text-chat
```

### *fasttrack*

```
router(config)#class-map type inspect fasttrack match-any ftrak-17-cmap
router(config-cmap)#match ?
  file-transfer      File transfer stream
  flow               Flow based QoS parameters
```

### *gnutella*

```
router(config)#class-map type inspect gnutella match-any gtella-17-cmap
router(config-cmap)#match ?
  file-transfer      Match file transfer stream
  flow              Flow based QoS parameters
```

### *kazaa2*

```
router(config)#class-map type inspect kazaa2 match-any kazaa2-17-cmap
router(config-cmap)#match ?
  file-transfer      Match file transfer stream
  flow              Flow based QoS parameters
```

New P2P protocol definitions or updates to existing P2P protocols can be loaded using the dynamic pdlm update functionality of NBAR. This is the configuration command to load the new PDLM:

```
ip nbar pdlm <file-location>
```

The new protocol is available in **match protocol ...** commands for class type inspect. If the new P2P protocol has services (sub-protocols), the new Layer 7 inspect class-map types, as well as Layer 7 match criteria, become available.

## **IM Application Inspection and Control**

Cisco IOS Software Release 12.4(4)T introduced IM Application Inspection and Control. IM support was not introduced with ZFW in 12.4(6)T, so users were unable to apply IM control and ZFW in the same firewall policy, as ZFW and legacy firewall features cannot co-exist on a given interface.

Cisco IOS Software Release 12.4(9)T supports stateful inspection and application control for these IM services:

- AOL Instant Messenger
- MSN Messenger
- Yahoo! Messenger

IM inspection varies slightly from most services, as IM inspection relies on controlling access to a specific group of hosts for each given service. IM services generally rely on a relatively permanent group of directory servers, which clients must be able to contact in order to access the IM service. IM applications tend to be very difficult to control from a protocol or service standpoint. The most effective way to control these applications is to limit access to the fixed IM servers.

## Configuring IM Inspection

IM inspection and control offers both Layer 4 Stateful Inspection and Layer 7 Application Control.

Layer 4 inspection is configured similarly to other application services:

```
class-map type inspect match-any my-im-class
match protocol [aol | msnmsgr | ymsgr ]
!
policy-map type inspect private-allowed-policy
class type inspect my-im-class
[drop | inspect | pass
```

IM applications are able to contact their servers on multiple ports to maintain their functionality. If you wish to allow a given IM service by applying the inspect action, you might not need a server-list to define permitted access to the IM service's servers. However, configuring a class-map that specifies a given IM service, such as AOL Instant Messenger, and applying the drop action in the associated policy-map can cause the IM client to try and locate a different port where connectivity is allowed to the Internet. If you do not want to allow connectivity to a given service, or if you want to restrict IM service capability to text-chat, you must define a server list so the ZFW can identify traffic associated with the IM application:

```
!configure the server-list parameter-map:
parameter-map type protocol-info <name>
server name <name>
server ip a.b.c.d
server ip range a.b.c.d a.b.c.d
```

For example, the Yahoo IM server list is defined as such:

```
parameter-map type protocol-info ymsgr-pmap
server name scs.msg.yahoo.com
server name scsd.msg.yahoo.com
server ip 66.77.88.99
server ip range 103.24.5.67 103.24.5.99
```

You will need to apply the server-list to the protocol definition:

```
class-map type inspect match-any ym-l4-cmap
match protocol ymsgr ymsgr-pmap
```

You must configure the **ip domain lookup** and **ip name-server ip.ad.re.ss** commands in order to enable name resolution.

IM server names are fairly dynamic. You will need to periodically check that your configured IM server lists are complete and correct.

Layer 7 (Application) Inspection augments Layer 4 Inspection with the capability to recognize and apply service-specific actions, such as selectively blocking or allowing text-chat capabilities, while denying other service capabilities.

IM Application Inspection presently offers the capability to differentiate between text-chat activity and all other application services. In order to restrict IM activity to text-chat, configure a Layer 7 policy:

```
class-map type inspect ymsgr match-any ymsgr-text-cmap
match service text-chat

class-map type inspect ymsgr match-any ymsgr-default-cmap
match service any
```

```
policy-map type inspect im ymsgr-l7-pmap
  class type inspect im ymsgr-text-cmap
    allow
    [log]
  class type inspect im ymsgr-text-cmap
    reset
    [log]
```

Apply the Layer 7 policy to the Yahoo! Messenger policy configured earlier:

```
class-map type inspect match-any my-im-class
match protocol ymsgr
!
policy-map type inspect private-allowed-policy
  class type inspect my-im-class
    inspect
  service-policy im ymsgr-l7-pmap
```

## URL Filtering

ZFW offers URL filtering capabilities to limit access to web content to that specified by a white- or black-list defined on the router, or by forwarding domain names to a URL filtering server to verify access to specific domains. ZFW URL filtering in Cisco IOS Software Releases 12.4(6)T to 12.4(15)T is applied as an additional policy action, similar to application inspection.

For server-based URL filtering, you must define a parameter-map that describes the **urlfilter** server configuration:

```
parameter-map type urlfilter websense-parmap
  server vendor [n2h2 | websense] 10.1.1.1
```

If static white- or black-lists are preferred, you can define a list of domains or subdomains that are specifically allowed or denied, while the inverse action is applied to traffic that does not match the list:

```
parameter-map type urlfilter websense-parmap
  exclusive-domain deny .disallowed.com
  exclusive-domain permit .cisco.com
```

If a URL black-list is defined using deny options in the exclusive-domain definitions, all other domains will be allowed. If any permit definitions are defined, all domains that will be allowed must be explicitly specified, similar to the function of IP access-control lists.

Set up a class-map that will match HTTP traffic:

```
class-map type inspect match-any http-cmap
  match protocol http
```

Define a policy-map that associates your class-map with **inspect** and **urlfilter** actions:

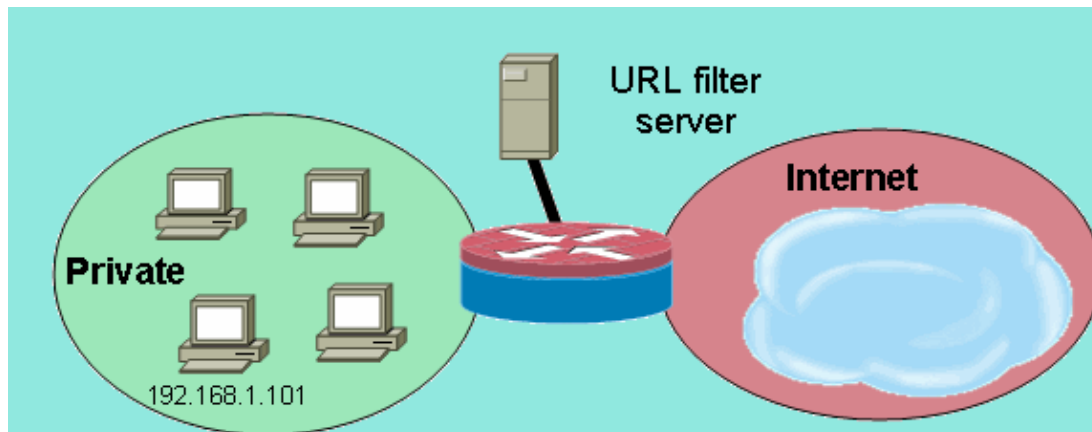
```
policy-map type inspect http-filter-pmap
  class type inspect http-cmap
    inspect
    urlfilter websense-parmap
```

This configures the minimum requirement to communicate with a URL filtering server. Several options are available to define additional URL filtering behavior.

Some network deployments might want to apply URL filtering for some hosts or subnets, while bypassing

URL filtering for other hosts. For instance, in Figure 9, all the hosts in the private zone must have HTTP traffic checked by a URL filter server, except for the specific host 192.168.1.101.

**Figure 9: URL Filtering example topology**



This can be accomplished by defining two different class-map maps:

- One class-map that only matches HTTP traffic for the larger group of hosts, which will receive URL filtering.
- One class-map for the smaller group of hosts, which will not receive URL filtering. The second class-map will match HTTP traffic, as well as a list of hosts that will be exempted from the URL filtering policy.

Both class-maps are configured in a policy-map, but only one will receive the **urlfilter** action:

```
class-map type inspect match-any http-cmap
  match protocol http
class-map type inspect match-all http-no-urlf-cmap
  match protocol http
  match access-group 101
!
policy-map type inspect http-filter-pmap
  class type inspect http-no-urlf-cmap
    inspect
  class type inspect http-cmap
    inspect
    urlfilter websense-parmap
!
access-list 101 permit ip 192.168.1.101 any
```

## Controlling Access to the Router

Most network security engineers are uncomfortable exposing the router's management interfaces (for example, SSH, Telnet, HTTP, HTTPS, SNMP, and so on) to the public Internet, and under certain circumstances, control might be needed for LAN access to the router as well. Cisco IOS Software offers a number of options to limit access to the various interfaces, which includes the Network Foundation Protection (NFP) feature family, various access control mechanisms for management interfaces, and ZFW's self-zone. You should review other features, such as VTY access control, management plane protection, and SNMP access control to determine which combination of router control features will work best for your specific application.

Generally, the NFP feature family is best suited for control of traffic destined for the router itself. Refer to Control Plane Security Overview in Cisco IOS Software for information that describes router protection using

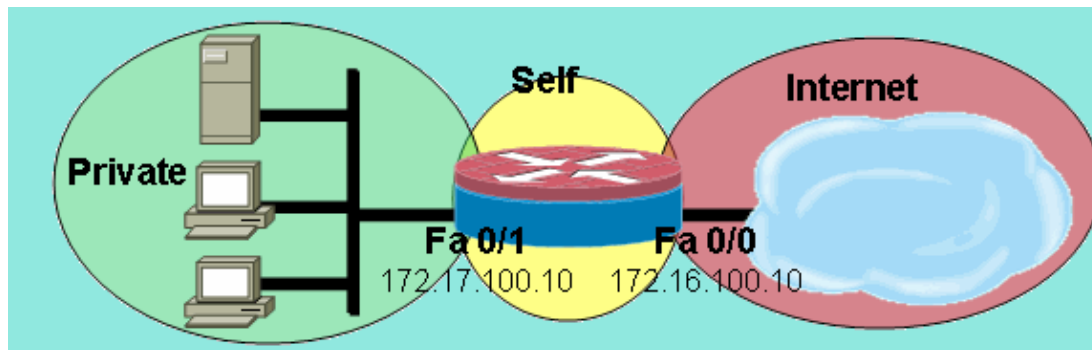
the NFP features.

If you decide to apply ZFW to control traffic to and from the IP addresses on the router itself, you must understand that the firewall's default policy and capabilities differ from those available for transit traffic. Transit traffic is defined as network traffic whose source and destination IP addresses do not match any IP addresses applied to any of the router's interfaces, and the traffic will not cause the router to send, for example, network control messages such as ICMP TTL expiration or network/host unreachable messages.

ZFW applies a default deny-all policy to traffic moving between zones, except, as mentioned in the general rules, traffic in any zone flowing directly to the addresses of the router's interfaces is implicitly allowed. This assures that connectivity to the router's management interfaces is maintained when a zone firewall configuration is applied to the router. If the same deny-all policy affected connectivity directly to the router, a complete management policy configuration would have to be applied before zones are configured on the router. This would likely disrupt management connectivity if the policy were improperly implemented or applied in the wrong order.

When an interface is configured to be a zone member, the hosts connected to the interface are included in the zone. However, traffic flowing to and from the IP addresses of the router's interfaces is not controlled by the zone policies (with the exception of circumstances described in the note following Figure 10). Instead, all of the IP interfaces on the router are automatically made part of the self zone when ZFW is configured. In order to control IP traffic moving to the router's interfaces from the various zones on a router, policies must be applied to block or allow/inspect traffic between the zone and the router's self zone, and vice versa. (See Figure 10.)

**Figure 10: Apply policy between network zones and router's self zone**



Although the router offers a default-allow policy between all zones and the self zone, if a policy is configured from any zone to the self zone, and no policy is configured from self to the router's user-configurable interface-connected zones, all router-originated traffic encounters the connected-zone to self-zone policy on its return to the router and is blocked. Thus, router-originated traffic must be inspected to allow its return to the self zone.

**Note:** Cisco IOS Software always uses the IP address associated with an interface nearest destination hosts for traffic such as syslog, tftp, telnet, and other control-plane services, and subjects this traffic to self-zone firewall policy. However, if a service defines a specific interface as the source-interface using commands that include, but not limited to **logging source-interface [type number]**, **ip tftp source-interface [type number]**, and **ip telnet source-interface [type number]**, the traffic is subjected to the zone-to-zone firewall policy for the source interface's zone and the security zone of the destination host. If a service is configured to use an interface that is assigned to a specific security zone, self-zone policy does not apply to that service's traffic.

**Note:** Some services (particularly routers' voice-over-IP services) use ephemeral or non-configurable interfaces that cannot be assigned to security zones. These services might not function properly if their traffic

cannot be associated with a configured security zone. If the service is configured to use one of the configurable physical or virtual interfaces on the device, the traffic should be handled by security zone policy relevant to that interface.

## Self-Zone Policy Limitations

Self-zone policy has limited functionality as compared to the policies available for transit-traffic zone-pairs:

- As was the case with classical stateful inspection, router-generated traffic is limited to TCP, UDP, ICMP, and complex-protocol inspection for H.323.
- Application Inspection is not available for self-zone policies.
- Session and rate limiting cannot be configured on self-zone policies.

## Self-Zone Policy Configuration

Under most circumstances, these are desirable access policies for router management services:

- Deny all Telnet connectivity, as Telnet's clear-text protocol easily exposes user credentials and other sensitive information.
- Allow SSH connections from any user in any zone. SSH encrypts user credentials and session data, which provides protection from malicious users that employ packet-capturing tools to snoop on user activity and compromise user credentials or sensitive information such as router configuration. SSH Version 2 provides stronger protection, and addresses specific vulnerabilities inherent to SSH Version 1.
- Allow HTTP connectivity to the router from the private zones, if the private zone is trustworthy. Otherwise, if the private zone harbors the potential for malicious users to compromise information, HTTP does not employ encryption to protect management traffic, and might reveal sensitive information such as user credentials or configuration.
- Allow HTTPS connectivity from any zone. Similar to SSH, HTTPS encrypts session data and user credentials.
- Restrict SNMP access to a specific host or subnet. SNMP can be used to modify router configuration and reveal configuration information. SNMP should be configured with access control on the various communities.
- Block ICMP requests from the public Internet to the private-zone address (assuming the private-zone address is routable). One or more public addresses may be exposed for ICMP traffic for network troubleshooting, if necessary. Several ICMP attacks can be used to overwhelm router resources or reconnoiter network topology and architecture.

A router can apply this type of policy with the addition of two zone-pairs for each zone that must be controlled. Each zone-pair for traffic inbound to, or outbound from, the router self-zone must be matched by the respective policy in the opposite direction, unless traffic will not be originated in the opposite direction. One policy-map each for inbound and outbound zone-pairs can be applied that describes all of the traffic, or specific policy-maps per zone-pair can be applied. Configuration of specific zone-pairs per policy-map provides granularity for viewing activity matching each policy-map.

Assuming an example network with an SNMP management station at 172.17.100.11, and a TFTP server at 172.17.100.17, this output provides an example of the entire management-interface access policy:

```
class-map type inspect match-any self service-cmap
  match protocol tcp
  match protocol udp
  match protocol icmp
  match protocol h323
!
class-map type inspect match-all to-self-cmap
  match class-map self service-cmap
```

```

    match access-group 120
    !
class-map type inspect match-all from-self-cmap
    match class-map self service-cmap
    !
class-map type inspect match-all tftp-in-cmap
    match access-group 121
    !
class-map type inspect match-all tftp-out-cmap
    match access-group 122
    !
policy-map type inspect to-self-pmap
    class type inspect to-self-cmap
        inspect
    class type inspect tftp-in-cmap
        pass
    !
policy-map type inspect from-self-pmap
    class type inspect from-self-cmap
        inspect
    class type inspect tftp-out-cmap
        pass
    !
zone security private
zone security internet
zone-pair security priv-self source private destination self
    service-policy type inspect to-self-pmap
zone-pair security net-self source internet destination self
    service-policy type inspect to-self-pmap
zone-pair security self-priv source self destination private
    service-policy type inspect from-self-pmap
zone-pair security self-net source self destination internet
    service-policy type inspect from-self-pmap

!
interface FastEthernet 0/0
    ip address 172.16.100.10
    zone-member security internet
!
interface FastEthernet 0/1
    ip address 172.17.100.10
    zone-member security private
!
access-list 120 permit icmp 172.17.100.0 0.0.0.255 any
access-list 120 permit icmp any host 172.17.100.10 echo
access-list 120 deny icmp any any
access-list 120 permit tcp 172.17.100.0 0.0.0.255 host 172.17.100.10 eq www
access-list 120 permit tcp any any eq 443
access-list 120 permit tcp any any eq 22
access-list 120 permit udp any host 172.17.100.10 eq snmp
access-list 121 permit udp host 172.17.100.17 host 172.17.100.10
access-list 122 permit udp host 172.17.100.10 host 172.17.100.17

```

Unfortunately, the self-zone policy does not offer the capability to inspect TFTP transfers. Thus, the firewall must pass all traffic to and from the TFTP server if TFTP must pass through the firewall.

If the router will terminate IPsec VPN connections, you should also define a policy to pass IPsec ESP, IPsec AH, ISAKMP, and NAT-T IPsec (UDP 4500). This depends on which is needed based on services you will use. The following policy can be applied in addition to the policy above. Note the change to the policy-maps where a class-map for VPN traffic has been inserted with a pass action. Typically, encrypted traffic is trustworthy, unless your security policy states that you must allow encrypted traffic to and from specified endpoints.

```

class-map type inspect match-all crypto-cmap

```

```

    match access-group 123
    !
    policy-map type inspect to-self-pmap
    class type inspect crypto-cmap
    pass
    class type inspect to-self-cmap
    inspect
    class type inspect tftp-in-cmap
    pass
    !
    policy-map type inspect from-self-pmap
    class type inspect crypto-cmap
    pass
    class type inspect from-self-cmap
    inspect
    class type inspect tftp-out-cmap
    pass
    !
    access-list 123 permit esp any any
    access-list 123 permit udp any any eq 4500
    access-list 123 permit ah any any
    access-list 123 permit udp any any eq 500

```

## Zone-Based Firewall and Wide-Area Application Services

Refer to Release Note for Cisco Wide Area Application Services (Software Version 4.0.13) – New Features for Software Version 4.0.13 for an application note that provides configuration examples and usage guidance

## Monitoring Zone-Based Policy Firewall with show and debug Commands

ZFW introduces new commands in order to view policy configuration and monitor firewall activity.

Display zone description and the interfaces contained in a specified zone:

```
show zone security [<zone-name>]
```

When the zone name is not included, the command displays the information of all the zones configured.

```

Router#show zone security z1
zone z1
  Description: this is test zone1
  Member Interfaces:
    Ethernet0/0

```

Display the source zone, destination zone and policy attached to the zone-pair:

```
show zone-pair security [source <source-zone-name>] [destination <destination-zone-name>]
```

When no source or destination is specified, all the zone-pairs with source, destination, and the associated policy are displayed. When only the source/destination zone is mentioned, all the zone-pairs that contain this zone as the source/destination are displayed.

```

Router#show zone-pair security
zone-pair name zp
  Source-Zone z1 Destination-Zone z2
  service-policy pl

```

Displays a specified policy-map:

```
show policy-map type inspect [<policy-map-name> [class <class-map-name>]]
```

When the name of a policy-map is not specified, it displays all policy-maps of type inspect (including Layer 7 policy-maps that contain a subtype).

```
Router#show policy-map type inspect p1
Policy Map type inspect p1
Class c1
Inspect
```

Displays the runtime inspect type policy-map statistics existing on a specified zone-pair.

```
show policy-map type inspect zone-pair [zone-pair-name] [sessions]
```

When **no zone-pair name** is mentioned, policy-maps on all zone-pairs are displayed.

The **sessions** option displays the inspection sessions created by the policy-map application on the specified zone-pair.

```
Router#show policy-map type inspect zone-pair zp
Zone-pair: zp

Service-policy : p1

Class-map: c1 (match-all)
Match: protocol tcp
Inspect
  Session creations since subsystem startup or last reset 0
  Current session counts (estab/half-open/terminating) [0:0:0]
  Maxever session counts (estab/half-open/terminating) [0:0:0]
  Last session created never
  Last statistic reset never
  Last session creation rate 0
  Last half-open session total 0

Class-map: c2 (match-all)
Match: protocol udp
Pass
  0 packets, 0 bytes

Class-map: class-default (match-any)
Match: any
Drop
  0 packets, 0 bytes
```

The **urlfilter** keyword displays the urlfilter-related statistics that pertain to the policy-map specified (or policy-maps on all targets when no zone-pair name is specified):

```
show policy-map type inspect zone-pair [zone-pair-name] [urlfilter [cache]]
```

When the **cache** keyword is specified along with **urlfilter**, it displays the **urlfilter** cache (of IP addresses).

Summary of the **show policy-map** command for inspect policy-maps:

```
show policy-map type inspect inspect { <policy name> [class <class name>] |
zone-pair [<zone-pair name>] [sessions | urlfilter cache] }
```

# Tuning Zone-Based Policy Firewall Denial-of-Service Protection

ZFW offers DoS protection to alert network engineers to dramatic changes in network activity, and to mitigate unwanted activity to reduce the impact of network activity changes. ZFW maintains a separate counter for every policy-map s class-map. Thus, if one class-map is used for two different zone-pairs policy-maps, two different sets of DoS protection counters will be applied.

ZFW provides DoS attack mitigation as a default on Cisco IOS Software releases before 12.4(11)T. The default DoS protection behavior changed with Cisco IOS Software Release 12.4(11)T. Refer to Tuning Cisco IOS Firewall Denial-of-Service Protection for further discussion and a procedure to adjust ZFW DoS protection.

Refer to Defining Strategies to Protect Against TCP SYN Denial of Service Attacks for more information about TCP SYN DoS attacks.

## Appendix

### Appendix A: Basic Configuration

```
ip subnet-zero
ip cef
!
bridge irb
!
interface FastEthernet0
 ip address 172.16.1.88 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet1
 ip address 172.16.2.1 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet2
 switchport access vlan 2
!
interface FastEthernet3
 switchport access vlan 2
!
interface FastEthernet4
 switchport access vlan 1
!
interface FastEthernet5
 switchport access vlan 1
!
interface FastEthernet6
 switchport access vlan 1
!
interface FastEthernet7
 switchport access vlan 1
!
interface Vlan1
 no ip address
 bridge-group 1
!
interface Vlan2
 no ip address
 bridge-group 1
```

```

!
interface BV11
 ip address 192.168.1.254 255.255.255.0
 ip route-cache flow
!
ip classless
ip route 0.0.0.0 0.0.0.0 172.16.1.1
!
bridge 1 protocol ieee
bridge 1 route ip
!
end

```

## Appendix B: Final (Complete) Configuration

```

ip subnet-zero
ip cef
!
ip port-map user-Xwindows port tcp from 6900 to 6910
!
class-map type inspect match-any L4-inspect-class
 match protocol tcp
 match protocol udp
 match protocol icmp
class-map type inspect match-any L7-inspect-class
 match protocol ssh
 match protocol ftp
 match protocol pop
 match protocol imap
 match protocol esmtp
 match protocol http
class-map type inspect match-any dns-http-class
 match protocol dns
 match protocol http
class-map type inspect match-any smtp-class
 match protocol smtp
class-map type inspect match-all dns-http-acl-class
 match access-group 110
 match class-map dns-http-class
class-map type inspect match-all smtp-acl-class
 match access-group 111
 match class-map smtp-class
class-map type inspect match-any Xwindows-class
 match protocol user-Xwindows
class-map type inspect match-any internet-traffic-class
 match protocol http
 match protocol https
 match protocol dns
 match protocol icmp
class-map type inspect http match-any bad-http-class
 match port-misuse all
 match strict-http
!
policy-map type inspect clients-servers-policy
 class type inspect L4-inspect-class
 inspect
policy-map type inspect private-dmz-policy
 class type inspect L7-inspect-class
 inspect
policy-map type inspect internet-dmz-policy
 class type inspect dns-http-acl-class
 inspect
 class type inspect smtp-acl-class
 inspect
policy-map type inspect servers-clients-policy

```

```
class type inspect Xwindows-class
inspect
policy-map type inspect private-internet-policy
class type inspect internet-traffic-class
inspect
class type inspect bad-http-class
drop
!
zone security clients
zone security servers
zone security private
zone security internet
zone security dmz
zone-pair security private-internet source private destination internet
service-policy type inspect private-internet-policy
zone-pair security servers-clients source servers destination clients
service-policy type inspect servers-clients-policy
zone-pair security clients-servers source clients destination servers
service-policy type inspect clients-servers-policy
zone-pair security private-dmz source private destination dmz
service-policy type inspect private-dmz-policy
zone-pair security internet-dmz source internet destination dmz
service-policy type inspect internet-dmz-policy
!
bridge irb
!
interface FastEthernet0
ip address 172.16.1.88 255.255.255.0
zone-member internet
!
interface FastEthernet1
ip address 172.16.2.1 255.255.255.0
zone-member dmz
!
interface FastEthernet2
switchport access vlan 2
!
interface FastEthernet3
switchport access vlan 2
!
interface FastEthernet4
switchport access vlan 1
!
interface FastEthernet5
switchport access vlan 1
!
interface FastEthernet6
switchport access vlan 1
!
interface FastEthernet7
switchport access vlan 1
!
interface Vlan1
no ip address
zone-member clients
bridge-group 1
!
interface Vlan2
no ip address
zone-member servers
bridge-group 1
!
interface BVI1
ip address 192.168.1.254 255.255.255.0
zone-member private
!
```

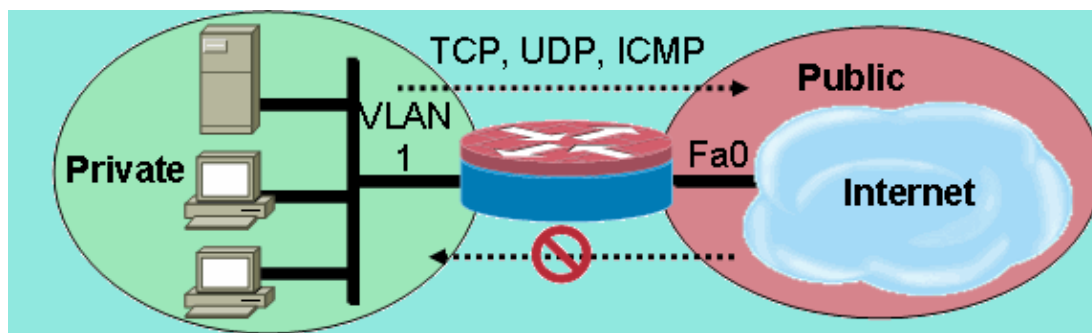
```

ip classless
ip route 0.0.0.0 0.0.0.0 172.16.1.1
!
access-list 110 permit ip any host 172.16.2.2
access-list 111 permit ip any host 172.16.2.3
!
bridge 1 protocol ieee
bridge 1 route ip
!
End

```

## Appendix C: Basic Zone–Policy Firewall Configuration for Two Zones

This example provides a simple configuration as a basis for feature testing for enhancements to the Cisco IOS Software ZFW. This configuration is a model configuration for two zones, as configured on an 1811 router. The private zone is applied to the router's fixed switch ports, so all hosts on the switch ports are connected to VLAN 1. The public zone is applied on FastEthernet 0.



```

class-map type inspect match-any private-allowed-class
  match protocol tcp
  match protocol udp
  match protocol icmp
class-map type inspect match-all http-class
  match protocol http
!
policy-map type inspect private-allowed-policy
  class type inspect http-class
    inspect my-parameters
  class type inspect private-allowed-class
    inspect
!
zone security private
zone security public
zone-pair security priv-pub source private destination public
  service-policy type inspect private-allowed-policy
!
interface fastethernet 0
  zone-member security public
!
Interface VLAN 1
  zone-member security private

```

## NetPro Discussion Forums – Featured Conversations

Networking Professionals Connection is a forum for networking professionals to share questions, suggestions, and information about networking solutions, products, and technologies. The featured links are some of the most recent conversations available in this technology.

|                                         |
|-----------------------------------------|
| Security: Intrusion Detection [Systems] |
|-----------------------------------------|

|               |
|---------------|
| Security: AAA |
|---------------|

|                   |
|-------------------|
| Security: General |
|-------------------|

|                       |
|-----------------------|
| Security: Firewalling |
|-----------------------|

---

## Related Information

- **Technical Support & Documentation – Cisco Systems**
- 

[Contacts & Feedback](#) | [Help](#) | [Site Map](#)

© 2008 – 2009 Cisco Systems, Inc. All rights reserved. [Terms & Conditions](#) | [Privacy Statement](#) | [Cookie Policy](#) | [Trademarks of Cisco Systems, Inc.](#)

---

Updated: Sep 13, 2007

Document ID: 98628

---