Cisco IOS Firewall
Zone-Based Policy Firewall
Release 12.4(6)T
Technical Discussion
February 2006

Agenda

- **Background**
  - **Functional Discussion**
  - **Configuration Overview**
  - **Comparison/Contrast with Legacy CBAC/Stateful Inspection Model**
- **Configuration for Use Cases**
  - **Two-Interface Firewall**
  - **Three-Interface Firewall**
  - **Firewall with VPN**
  - **Application Inspection**

Introduction and background

In the Beginning

- **ACLs had to be configured on router interfaces to block traffic to provide initial access policy**
- **Cisco IOS Software Stateful Inspection (formerly CBAC) offered interface-based firewall service**

  Traffic entering or leaving an interface is inspected for service conformance; if traffic matches requirements, the return traffic is allowed back through the firewall
- **Inspection policy and ACL policy combined to define firewall policy**

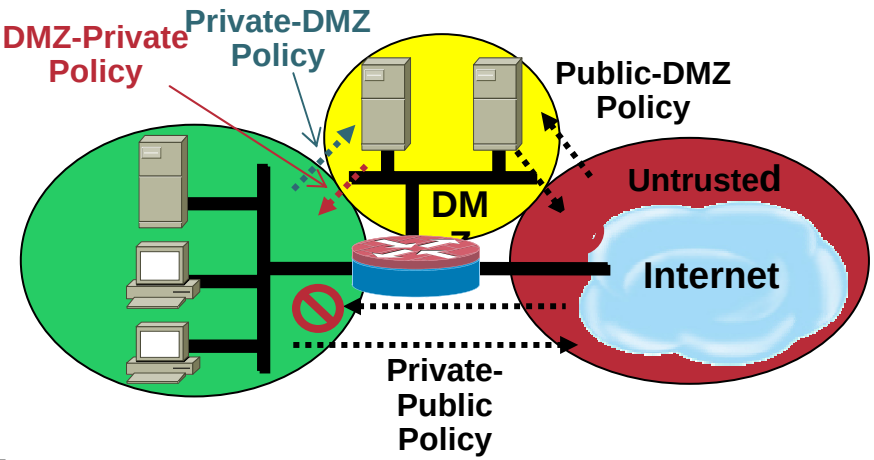
Legacy Cisco IOS Software Stateful Inspection

- **Multiple inspection policies and ACLs on several interfaces in a router make it difficult to correlate the policies that will be applied to traffic between multiple interfaces**
- **Very little inspection policy granularity**

  Policies could not be tied to a host group or subnet with an ACL. All traffic through a given interface was subject to the same inspection
- **Classic Stateful Inspection relies too heavily on ACLs**

The New Era: Zone-Based Policy Firewall

- **Zone-Based Policy introduces a new firewall configuration model**
- **Policies are applied to traffic moving between zones, not interfaces**



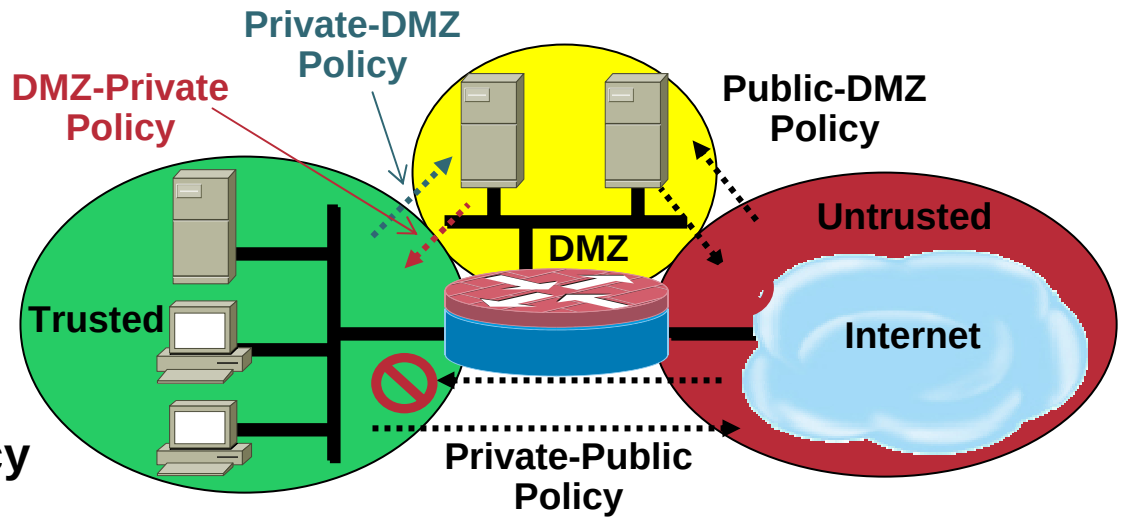
- **Subnet- and host-specific policies**
- **Offers functionality similar to PIX object-groups**
  - **Service lists can be combined with network and host address lists**
- **Firewall policies can be more clearly understood**
  - **Only policy from *Zone A* to *Zone B impacts traffic***
  - **No interference between multiple inspection policies or ACLs**

Zone-Based Policy Firewall

- **Unidirectional policy is applied between zones**
- **Default policy for inter-zone traffic is *DENY ALL***
- **Multiple traffic classes and actions can be applied per zone-pair**
- **Connection parameters are global unless zone-pair-specific parameters are applied**

Policies can define combinations of

- **IP address/subnet/ACL**
- **TCP/UDP/ICMP**
- **Application Service**
- **Application-Specific Policy**



Benefits

- **Removes dependence on ACLs**

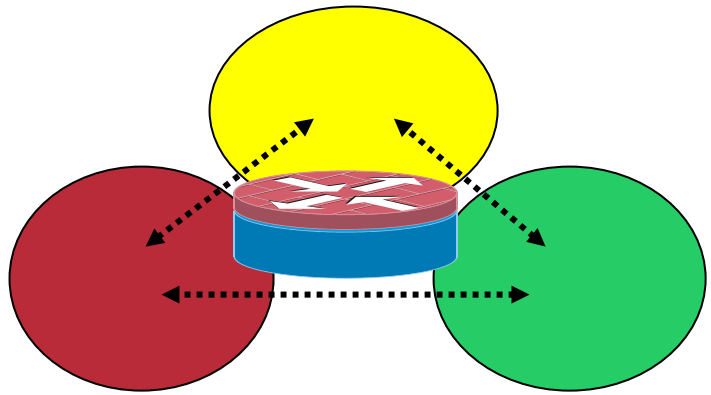
  Changes router security posture to “block unless explicitly allowed”

- **Policies are easy to read and troubleshoot**
- **One policy affects any given traffic, instead of multiple ACLs and inspection actions**

Firewall Functionality Supported in ZBP

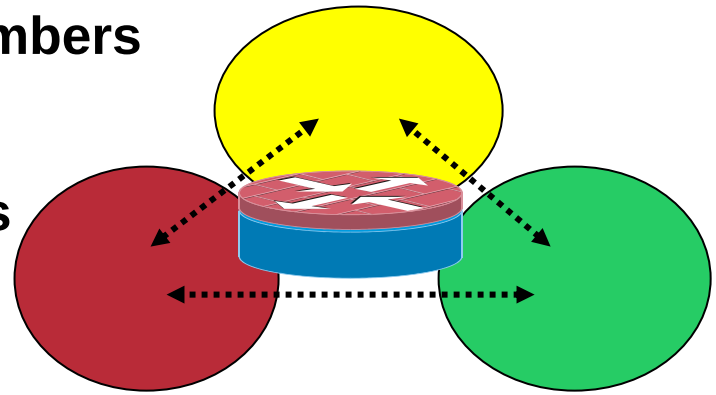
- **Layer 3 Stateful Inspection (Classic CBAC)**
- **Layer 2 Stateful Inspection (Transparent Firewall)**
- **Application Inspection**

  HTTP, SMTP/ESMTP, POP, IMAP, SunRPC
- **URL Filtering**
- **VRF-Aware Firewall**



Traffic Specification Attributes

- **Policies can define combinations of**
  - **IP address/subnet/ACL**
    - **Address groups are defined by associating an ACL with a policy**
  - **TCP/UDCP/ICMP**
  - **Application Service**
    - **As defined by Port-Application Mapping**
    - **Include user-definable port numbers**
  - **Application-Specific Policy**
    - **Application Inspection Engines**
    - **HTTP, IM, P2P, POP, IMAP**



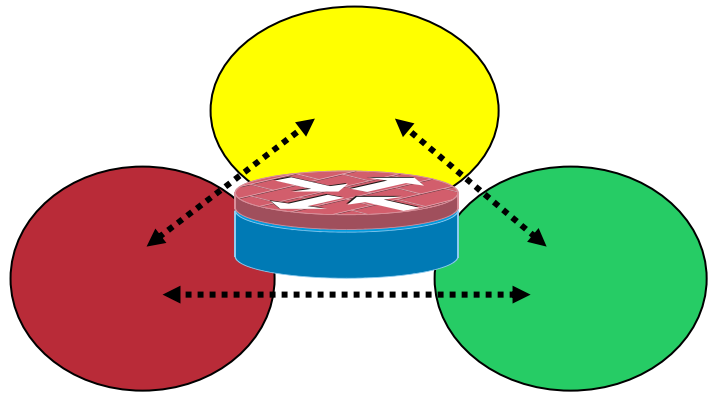
Feature Interoperability

- **Interoperates with all existing features**
  - **IPS, NAT, QoS, etc.**
- **Supports all physical and virtual interface types**
  - **Ethernet, Dialer, VTI, Loopback, etc.**
- **Works with all flavors of IPsec VPN**
  - **SVTI/DVTI, Legacy IPsec and EasyVPN, GRE+IPsec, DMVPN, SSLVPN**
- **Can co-exist with legacy firewall configuration**
  - **Not on the same interface**
  - **Interface ACLs are still relevant**

Configuration

Step-by-Step: Configure a ZBP Firewall

1. **Identify interfaces of “similar” security and group them into security zones**
2. **Determine both directions’ traffic between zones**
3. **Set up zone pairs for any policy other than deny all**
4. **Define class-maps to describe traffic between zones**
5. **Associate class-maps with policy-maps to define actions applied to specific policies**
6. **Assign policy-maps to zone-pairs**



Zoning Rules

- **Policy application and default policy for traffic is applied according to these rules**

Source interface member of zone?	Destination interface member of zone?	Zone-pair exists?	CPL policy exists?	RESULT
NO	NO	N/A	N/A	No impact of zoning/C3PL
YES (zone name *foo)*	YES (zone name *foo)*	Not allowed*	N/A	No policy lookup (PASS)
YES	NO	N/A	N/A	DROP
NO	YES	N/A	N/A	DROP
YES (zone name *foo)*	YES (zone name *bar)*	NO	N/A	DROP
YES (zone name *foo)*	YES (zone name *bar)*	YES	NO	DROP
YES (zone name *foo)*	YES (zone name *bar)*	YES	YES	C3PL policy actions

* **Zone-pair MUST have different zones as source and destination**

Zoning Rules (Cont.)

Source interface member of zone?	Destination interface member of zone?	Zone-pair exists?	C3PL policy exists?	RESULT
SELF	YES	NO	-	PASS
SELF	YES	YES	NO	PASS
SELF	YES	YES	YES	C3PL policy actions
YES	SELF	NO	-	PASS
YES	SELF	YES	NO	PASS
YES	SELF	YES	YES	C3PL policy actions

Traffic sourced by the router (router generated traffic)

Traffic destined for the router (router terminated traffic)

Zoning Rules Summarized

- **If two interfaces are not in zones, traffic flows freely between them**
- **If one interface is in a zone, and another interface is not in a zone, traffic may never flow between them**
- **If two interfaces are in two different zones, traffic will not flow between the interfaces until a policy is defined to allow the traffic**

Specifying Policy - Basics

- **Applies CPL framework**

  Based on existing MQC framework in Cisco IOS Software

- **Only 3 constructs**

  Class-map – Specifies interesting traffic via “match” conditions

  Policy-map – Associates actions with the above specified traffic

  Parameter-map – Operating parameters for the classification and action application

- **Each of the constructs is a specific feature- or protocol-specific type**

  Example: class-map type inspect match-all my-cmap

The ‘inspect’ type class-map

- **Applies logical qualifiers ‘match-all’ and ‘match-any’; determines the way a packet is matched against filters in a class-map**
- **Applies three types of match statements (filters)**

  match protocol <protocol-name>

  match access-group <number | name>

  match class <class-map-name>

Defining Class-Maps

- **Match-all – AND logic; traffic must match all filters; exit on first non-match; the default if match-all/match-any is not specified**
- **Match-any – OR logic; traffic must match at least one filter; exit on first match**
- **Filter specification order is very important to**
  - **Correctly apply service inspection**
  - **Optimize efficiency**
- **Changing a class from match-all to match-any (or vice-versa) may change the behavior of the policy**

Examples of class-map type inspect

```
class-map type inspect match-all c1
    description Web traffic which ALSO matches ACL 101
    match protocol http
    match access-group 101
class-map type inspect match-any c2
    description Traffic which is bound for ANY OF these 3 protocols
    match protocol http
    match protocol ftp
    match protocol smtp
class-map type inspect match-all c3
    description Traffic bound for ANY OF the 3 protocols in c2 AND which also
    matches ACL 199
    match access-group 199
    match class c2
```

Defining Class-Maps (Cont.)

- **‘Match protocol’ filter determines which service match the class-map, and how the traffic will be inspected, if the policy-map applies the inspect action; the traffic will be expected to behave as the specified service if the traffic matches the “protocol” filter**

Defining Class-Maps (Cont.)

- **If a packet matches a class, but there is insufficient information on what protocol matched (absence or non-execution of a match protocol filter), class-map selects service inspection by comparing traffic against services known by PAM; if no PAM mapping is present, L4 (ie: TCP/UDP/ICMP) inspection is performed**
- **Examples**
  - **A single ‘match access-group’**
  - **A ‘match not protocol’ filter in a class**

The ‘match protocol <xxx>’ Filter

- **Matches the protocol in the packet headers against the specified protocol**

  L4 protocols - match protocol <tcp|udp|icmp>

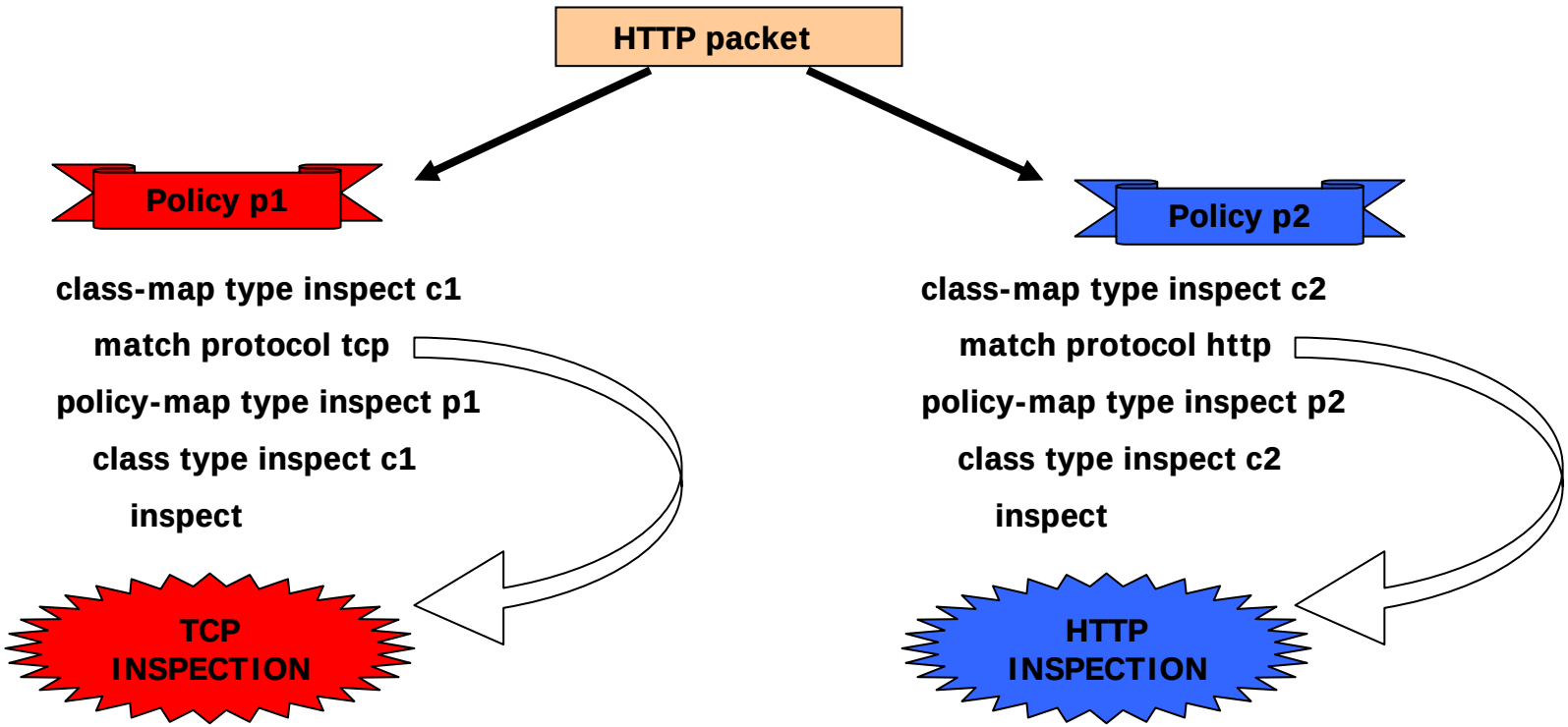
  L7 protocols - match protocol <http|smtp|telnet|…> (all protocols available in ‘ip inspect name <> ?’ are available here)

- **In case of L7 protocols, the ports associated with the protocol are dictated by the existing PAM feature**

  For example, ‘match protocol http’ will match packets bound for port 8080 (in addition to port 80) if the configuration has ‘ip port-map http port 8080’

The ‘match protocol <xxx>’ Filter

- **Determines the protocol for which the packet will be inspected, if ‘inspect’ action is configured in the policy-map**



match protocol (Examples)

```
class-map type inspect match-any c1
    match protocol tcp
    match protocol http
class-map type inspect match-any c2
    match protocol http
    match protocol tcp
class-map type inspect match-all c3
    match protocol tcp
    match protocol http
policy-map type inspect p1
    class type insp cx
        inspect
```

HTTP	SMTP	Any TCP	REASON
TCP	*TCP*	*TCP*	First-match-exit for match-any class. TCP is 1st filter
HTTP	*TCP*	*TCP*	HTTP pak matches 1st filter. Other TCP match 2nd
HTTP	*NONE*	*NONE*	Match-all semantics. HTTP insp is explicitly requested

Not a really useful configuration; maybe useful in some cases

The ‘match access-group’ Filter

- **Matches the packet against the specified ACL**

  User can specify anything in the ACL; everything is honored (meaning IP addresses/subnets, ports, dscp, precedence etc.); gives us the full ACL functionality for free

- **Recommended usage is to specify only IP addresses/subnets (and use ‘match protocol’ for protocol information); typical usage is in conjunction with ‘match protocol’ in a match-all class-map**

The ‘match access-group’ Filter (Cont.)

- **What if ‘permit tcp any any eq 21’ and ‘match protocol http’ are specified in a match-all class-map?**

  With the default PAM, no packet will match this class; however, we wont complain; deemed to be a misconfiguration

The ‘match access-group’ Filter (Cont.)

```
access-list 101 permit ip 192.168.1.0 0.0.0.255 any
class-map type inspect match-all c1
    match protocol tcp
    match access-group 101
policy-map type inspect p1
    class type inspect c1
        inspect
```

What protocol do we inspect for here? (assume HTTP packet)

- **TCP. Reason – User *asked* for TCP inspection via the ‘match protocol tcp’ filter. This is *not* a special case; there is sufficient information**
- **But we were getting L7 inspection with just match access-group; why TCP now? Rule – match protocol decides which protocol to inspect for; here user explicitly asked for TCP inspection; in the previous case, he did not ask for anything – so we went for L7 returned by PAM**

The ‘match access-group’ Filter (Cont.)

```
access-list 101 permit ip 192.168.1.0 0.0.0.255 any
class-map type inspect c1
    match access-group 101
policy-map type inspect p1
    class type inspect c1
        inspect
```

What protocol do we inspect for here?

- **This is a special case; reason – insufficient information; config is leaving us guessing on the protocol; we have to *define* the behavior**
- **Inspection will be performed for the L7 protocol based on PAM mappings; if no PAM mapping is found, relevant L4 inspection is performed; ie: HTTP – http inspection, TFTP – tftp inspection, port 9737 - TCP inspection**
- **Remember that we got into this situation because sufficient information on protocol was not conveyed by the user; special case which can be very useful**

The ‘match access-group’ Filter (Cont.)

```
access-list 101 permit ip 192.168.1.0 0.0.0.255
class-map type inspect match-any c1
   match protocol tcp
   match access-group 101
policy-map type inspect p1
   class type inspect c1
       inspect
```

What protocol do we inspect for here?

- **For TCP connections, we get TCP inspection; reason – first-match semantics of match-any class-map**
- **For UDP packets, we again have insufficient information on protocol; so, this is equivalent to the match access-group special case; result – L7 inspection as dictated by PAM, or L4 if there is no PAM mapping**
- **Not a recommended configuration**

The ‘match access-group’ Filter (Cont’d)

```
access-list 101 permit ip 192.168.1.0 0.0.0.255
class-map type inspect match-all c1
    match access-group 101
    match class my-prots
policy-map type inspect p1
    class type inspect c1
        inspect
```

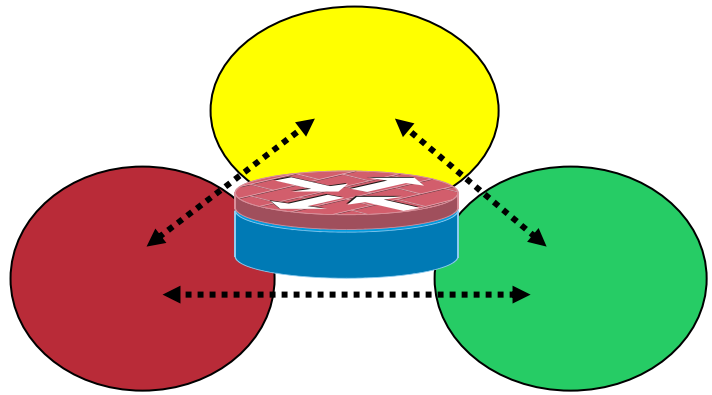
```
class-map type inspect match-any my-prots
   match protocol http
   match protocol smtp
   match protocol ftp
```

So, is the ‘match access-group’ filter confusing and evil?

- No, it is a very useful construct
- Different behaviors result because the match-any/all constructs dictate the matching logic in the class-map; simple rule: In case of insufficient information on the protocol, go for the L7 inspection returned by PAM
- The configuration shown above is highly recommended. It is what customers usually want; it doesn’t force us to guess

ZBP Policy Action

- **Inspect**
  - **Monitor outbound traffic according to permit/deny policy**
  - **Anticipate return traffic according to session table entries**
- **Drop**
- **Pass**
  - **Requires manually-configured ACL for reflexive policy**
  - **No stateful capability**



Access-List Caveat

- **Interface ACLS are still applicable, in addition to Zone-Based Policy**
  - **`ip access-group in` is applied *before* ZBP**
  - **`ip access-group out` is applied *after* ZBP**
- **Beware the implicit “deny any” at the end of ACL**
- **If you have a problematic source or destination host that you wish to address with an interface ACL, always end the ACL with “permit ip any any”**

Examples – drop traffic

```
access-list 199 permit ip host 192.168.1.13 any
class-map type inspect bad-host
   match access-group 199
policy-map type inspect mypolicy
   class type inspect bad-host
      drop
zone-pair security in-out source in-zone dest out-zone
   service-policy type inspect mypolicy
```

Specify interesting traffic – All traffic that matches ACL 199

Specify the traffic (class) on which an action is to be performed – All IP traffic coming from 192.168.1.13

The "action" – drop, performed on traffic specified by class *bad-host*

Policy in English – Drop all traffic originated by 192.168.1.13 going from zone *in-zone* to *out-zone*

Example – inspect traffic

```
class-map type inspect match-all inspect-traffic
   match protocol tcp
parameter-map type inspect insp-params
   audit-trail on
   tcp synwait-time 10
policy-map type inspect mypolicy
   class type inspect inspect-traffic
      inspect insp-params
zone-pair security in-out source in-zone dest out-zone
   service-policy type inspect mypolicy
```

specifies all TCP traffic

Parameters for inspection

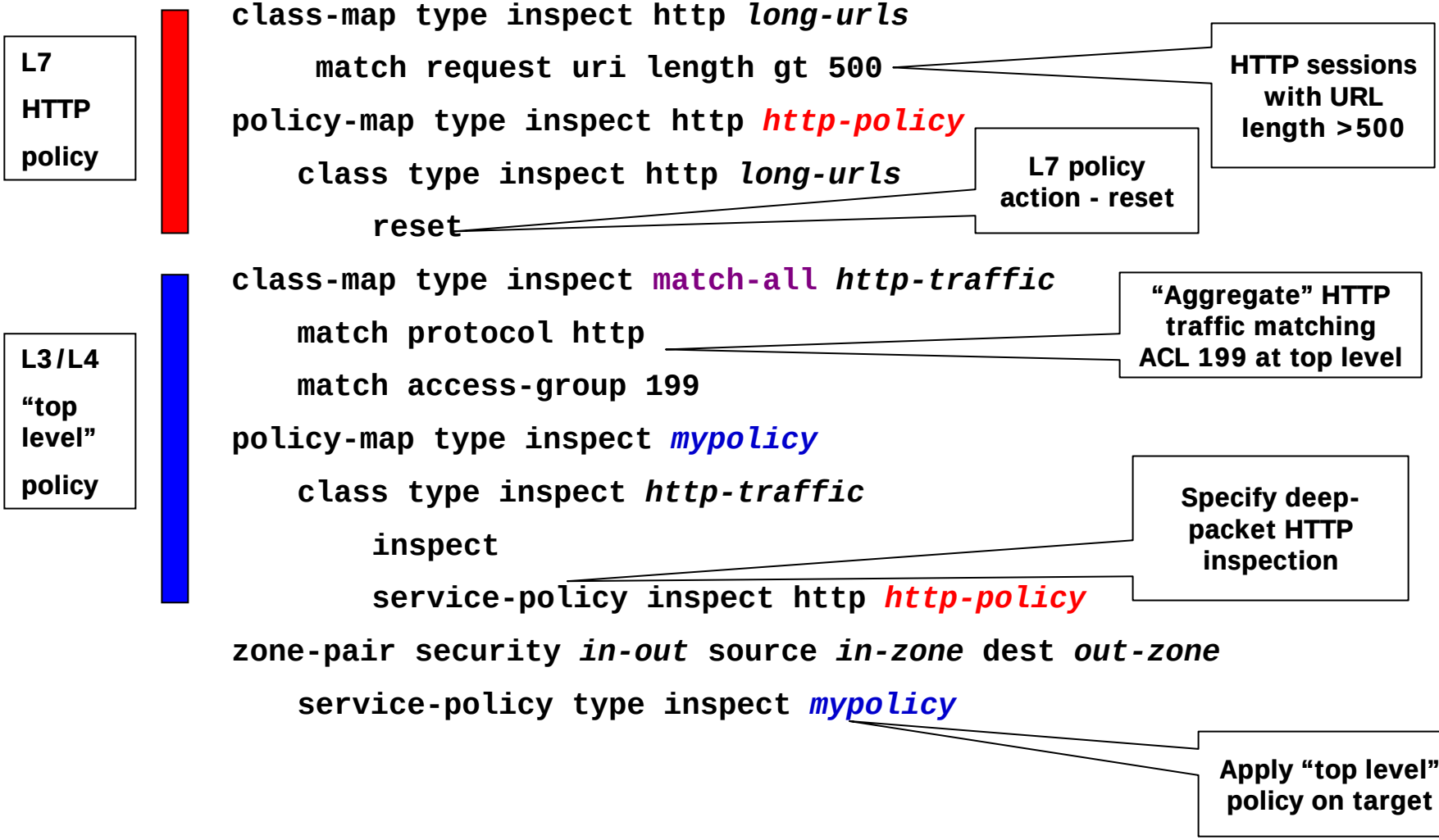
Inspect action with specified parameters

Default action is DROP if packet does not match any class in the policy
Default action is DROP if no action is specified for a class in the policy

Policy Types: Layer 3/4/7

- **L3/L4 policy is a “top level” policy which is attached to the zone-pair; “Aggregate” traffic using ‘match protocol/access-list’ selections, apply “high level” actions like drop, inspect, urlfilter and deep-inspection**
- **L7 or application policy is optional and is typically applied to control finer details of an application ie: http, smtp etc. It is contained in an L3/L4 policy and cannot be directly attached to a target**
- **Summary: L3/L4 policy suffices for basic inspection; finer application level inspection calls for creation of an L7 policy which is nested in the L3/L4 policy**

Hierarchy - example



Basic inside-outside topology – Case 1

- **Simple 2 interface topology – internal network and internet**
- **Current inspect rule style configuration**

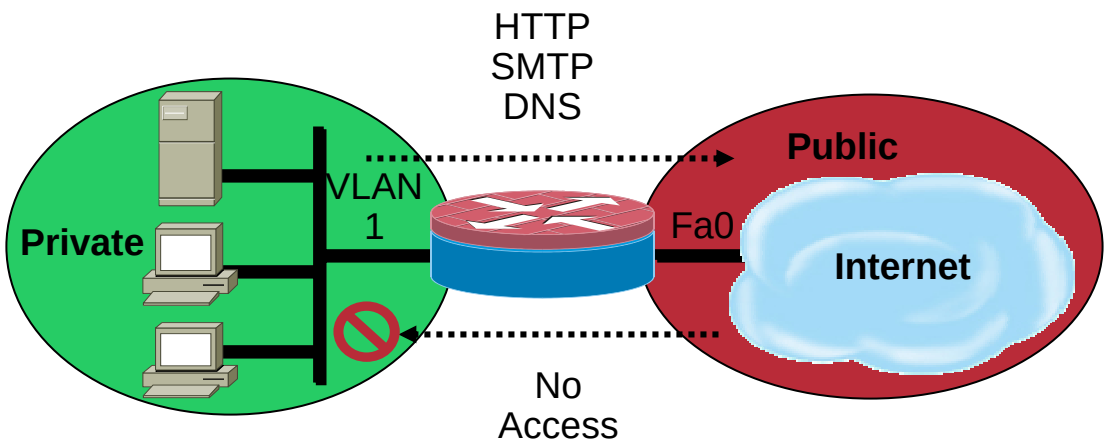
```
ip inspect name test tcp
ip inspect name test ftp
ip inspect name test http
ip inspect name test icmp

access-list 101 deny ip any any

interface ethernet0
     ip inspect test in
interface serial 0
     ip access-group 101 in
```

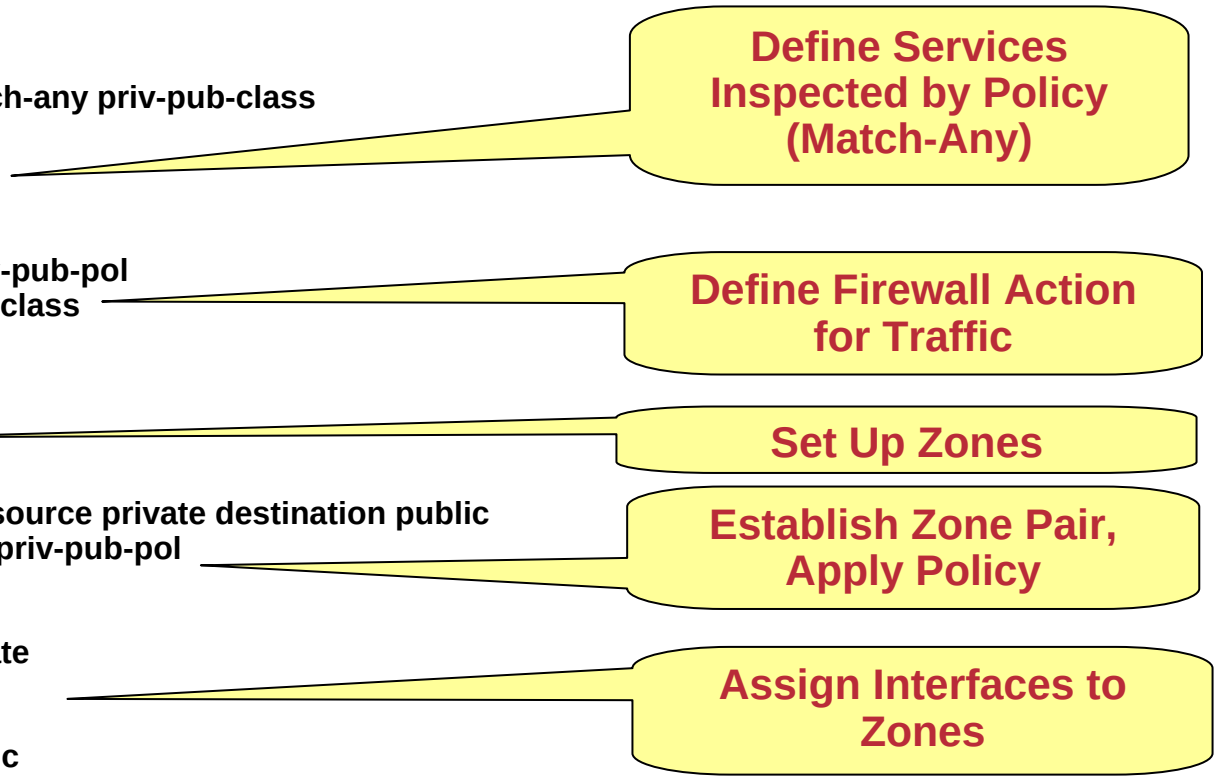
Consider a Basic Firewall

- **Private Zone must reach Internet, with access to HTTP, SMTP, and DNS services**
- **Internet should not have any inbound access**



Zone-Based Policy Firewall Configuration

```
class-map type inspect match-any priv-pub-class
 match protocol http
 match protocol smtp
 match protocol dns
!
policy-map type inspect priv-pub-pol
 class type inspect priv-pub-class
  inspect
!
zone security private
zone security public
!
zone-pair security priv-pub source private destination public
 service-policy type inspect priv-pub-pol
!
interface VLAN 1
 zone-member security private
!
interface fastethernet 0
 zone-member security public
```



Basic inside-outside topology (cont'd)

- **Policy firewall configuration for same 2 interface topology**

```
class-map type inspect match-any insp-traffic
     match protocol ftp
     match protocol http
     match protocol icmp
     match protocol tcp
policy-map type inspect mypolicy
     class type inspect insp-traffic
       inspect
zone-pair security in-out source in-zone dest out-zone
     service-policy type inspect mypolicy
```

Order of match statements important. Classification exits on first match

ACL 101 not needed anymore. *in-zone* is assumed to contain ethernet0; *out-zone* serial 0

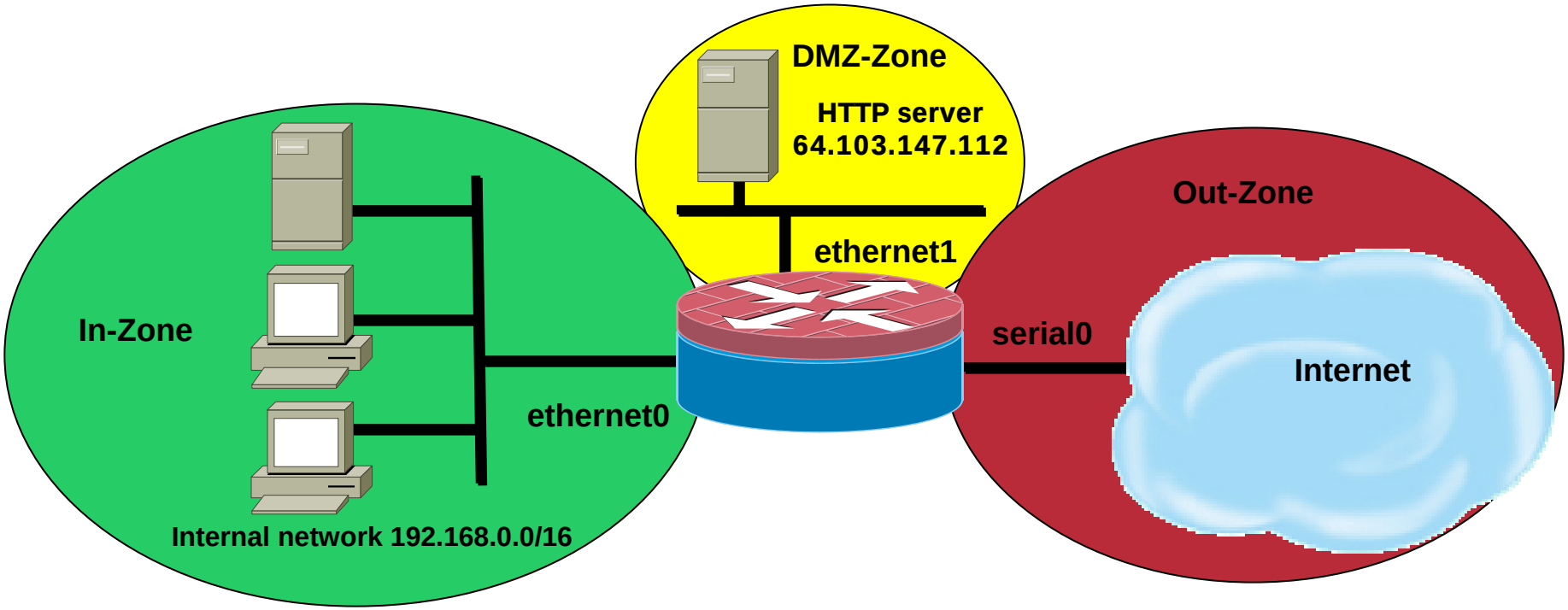
inside-outside-dmz topology – Case 2

- **Network consists of three zones**

  Out-Zone: Internet

  DMZ-Zone: 64.103.147.112

  In-Zone: Private Network, 192.168.0.0/16



inside-outside-dmz topology – Case 2

- **Inspect tcp, http, icmp from inside-outside. Allow and inspect HTTP to hosted webserver on DMZ**
- **Current inspect rule style configuration**

```
 interface ethernet0
      ip inspect test in
interface serial 0
      ip access-group 101 in
      ip inspect dmz-rule in
interface ethernet1
      ip access-group 102 in
```

```
ip inspect name test tcp
ip inspect name test http
ip inspect name test icmp

ip inspect name dmz-rule http
```

```
access-list 101 permit ip any host 64.103.147.112 eq http
access-list 101 deny ip any any
access-list 102 deny ip any any
```

inside-outside-dmz (Cont.)

- **Policy firewall configuration**

```
class-map type inspect insp-traffic
      match protocol http
      match protocol icmp
      match protocol tcp

class-map type inspect match-all myhttp
      match access-group 199
      match protocol http
```

```
policy-map type inspect p-inout
  class type inspect insp-traffic
    inspect

policy-map type inspect webtraffic
  class type inspect myhttp
    inspect
```

```
access-list 199 permit tcp any host 64.103.147.112
zone-pair security in-out source in-zone dest out-zone
      service-policy type inspect p-inout
zone-pair security out-dmz source out-zone dest dmz-zone
      service-policy type inspect webtraffic
```

inside-outside multiple flows – Case 3

- **Policy firewall configuration for interface topology with different inspection for different “flows”**
- **Problem statement**

  HTTP, SMTP, FTP inspection for traffic originating from 192.168.1.0/24 sub network. Stricter DOS thresholds to be configured for inspection

  TCP, UDP, H323 inspection for traffic originating from 192.168.2.0/24 sub network. Default inspection parameters

  No Layer 7 inspection required anywhere
- **This is not possible with the existing inspect rule CLI; all traffic entering/leaving an interface will be subjected to the same inspect rule; different policies in the context of a given target (interface) not possible presently**

inside-outside multiple flows – Case 3

Class-map definitions

```
class-map type inspect match-any proto-list-1
     match protocol ftp
     match protocol http
     match protocol smtp
class-map type inspect match-any proto-list-2
     match protocol tcp
     match protocol udp
     match protocol h323

class-map type inspect match-all first-subnet-traffic
     match access-group 198
     match class proto-list-1
class-map type inspect match-all second-subnet-traffic
     match access-group 199
     match class proto-list-2
```

Nested class-maps. Observe match-all/any semantics

permit ip 192.168.1.0 0.0.255.255 any

permit ip 192.168.2.0 0.0.255.255 any

inside-outside multiple flows – Case 3

policy-map parameter-map definitions

Inspection of different protocols using different parameters for the 2 flows

```
parameter-map type inspect first-subnet-params
     max-incomplete low 100
     max-incomplete high 150
     tcp max-incomplete host 100 block-time 10

policy-map type inspect mypolicy
     class type inspect first-subnet-traffic
               inspect first-subnet-params
     class type inspect second-subnet-traffic
               inspect

zone-pair security in-out source in-zone dest out-zone
     service-policy type inspect mypolicy
```

match access-group/inspect – Case 3.1

```
access-group 199 permit 192.168.2.0 0.0.0.255 any
class-map type inspect interesting-traffic
  match access-group 199

policy-map type inspect mypolicy
  class type inspect interesting-traffic
    inspect

zone-pair security in-out source in-zone dest out-zone
  service-policy type inspect mypolicy
```

- This is a valid configuration. Note the there is no 'match protocol xxx' configured
- All traffic matching ACL 101 is subjected to inspection
- The protocol for inspection is decided by the PAM mappings configured on the box
- For example traffic matching ACL 199 and bound for port 21 will be inspected for FTP
- If there is no PAM mapping for the port, L4 inspection will be performed
- Somewhat similar to the F1 'default-inspection-traffic' behavior

Inspect ‘global’ CLIs

Router(config)# ip inspect ?	Router(config)# parameter-map type inspect abc Router(config-profile)#?
L2-transparent dhcp-passthrough	*No equivalent at present. Use same command*
alert-off	alert <on | off> (default is on)
audit-trail	audit-trail <on | off> (default is off)
dns-timeout <N>	dns-timeout <N>
hashtable-size <N>	*No equivalent. Use same command*
Log drop-pkt	Log drop-pkt
max-incomplete < high <N> | low <N> >	max-incomplete < high <N> | low <N> >

Inspect ‘global’ CLIs (Cont.)

Router(config)# ip inspect ?	Router(config)# parameter-map type inspect abc Router(config-profile)#?
Name <name> <protocol-name> …	*Not applicable. Equivalent functionality provided through class/policy-maps*
One-minute <high <N> | low <N> >	One-minute <high <N> | low <N> >
tcp <block-non-session | finwait-time <N> | idle-time <N> | max-incomplete host <N> block-time <N>| synwait-time <N> >	tcp <finwait-time <N> | idle-time <N> | max-incomplete host <N> block-time <N>| synwait-time <N> > *(block-non-session not applicable to c3pl/zone model)*
udp idle-time <N>	udp idle-time <N>
No equivalent command. Currently done through ‘ip inspect name test icmp timeout N’ command	icmp idle-time <N>

L7 Policy – General Approach

- **L7 class/policy-maps are protocol specific; the options appearing under them depend on the protocol and the capabilities of the existing application inspection module**
- **As the inspection engines of individual protocols are enhanced, more options will be added to the corresponding L7 class/policy-maps to provision the new functionality**
- **As of now, L7 policies can be configured for the following protocols: HTTP, SMTP, POP3, IMAP and RPC**

L7 Policy – General Approach (Cont.)

- **The L7 policy-map is attached to the top-level policy using the “service-policy inspect <http | smtp | …> <policy-name>” command**
- **The class in the top-level policy for which an L7 policy-map is configured MUST have a “match protocol” filter. This protocol and the L7 policy-map protocol must be the same. If only ‘match access-group’ filters are present in the class-map, L7 policy cannot be configured for that class**
- **A single L7 policy-map may be used in multiple classes/policies**

SMTP Inspection - Case 4

Existing CLI

```
ip inspect name test smtp
```

C3PL CLI

```
class-map type inspect c1
   match protocol smtp
policy-map type inspect mypolicy
   class type inspect c1
      inspect
```

Can include other match protocol / access-group statements

Holds true for all protocols.
match protocol xxx
inspect
in the c3pl model exhibits the same behavior as
ip inspect name *test* xxx

zone / zone-pair configuration not shown

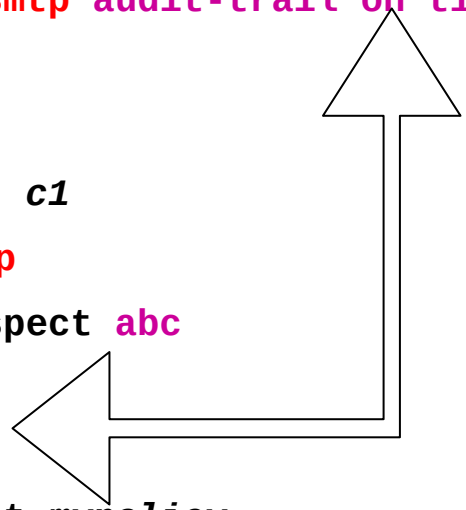
SMTP Inspection - Case 5

Existing CLI

```
ip inspect name test smtp audit-trail on timeout 360
```

C3PL CLI

```
class-map type inspect c1
    match protocol smtp
parameter-map type inspect abc
    audit-trail on
    tcp idle-time 360
policy-map type inspect mypolicy
    class type inspect c1
        inspect abc
```



audit-trail, alert and timeout are part of parameter-map (type inspect)

SMTP Inspection - Case 6

Existing CLI

```
ip inspect name test smtp max-data 100000
```

C3PL CLI

```
class-map type inspect smtp huge-mails
    match data-length gt 100000
policy-map type inspect smtp mysmtp-policy
    class type inspect smtp huge-mails
        reset
class-map type inspect c1
    match protocol smtp
policy-map type inspect mypolicy
    class type inspect c1
        inspect
        service-policy inspect smtp mysmtp-policy
```

Basic inspection

More application level control (via L7/DPI policy-map)

Sun RPC Inspection - Case 7

Existing CLI

```
ip inspect name test rpc program-number 2345 wait-time 5

class-map type inspect sunrpc rpc-prog-nums
    match program-number 2345
policy-map type inspect sunrpc myrpc-policy
    class type inspect sunrpc rpc-prog-nums
        allow wait-time 5
class-map type inspect c1
    match protocol rpc
policy-map type inspect mypolicy
    class type inspect c1
        inspect
        service-policy inspect sunrpc myrpc-policy
```

C3PL CLI

Multiple rpc program numbers are configured as multiple "match" statements in the class *rpc-prog-nums*

POP3 Inspection - Case 8

Existing CLI

```
ip inspect name test pop3 alert on secure-login

class-map type inspect pop3 pop3-class
    match login clear-text
policy-map type inspect pop3 mypop3-policy
    class type inspect pop3 pop3-class
        alarm
class-map type inspect c1
    match protocol pop3
policy-map type inspect mypolicy
    class type inspect c1
        inspect
        service-policy inspect pop3 mypop3-policy
```

C3PL CLI

secure-login option checks if the login process is happening in clear-text

POP3 Inspection - Case 9

Existing CLI

```
ip inspect name test pop3 secure-login reset

class-map type inspect pop3 pop3-class
    match login clear-text
policy-map type inspect pop3 mypop3-policy
    class type inspect pop3 pop3-class
        reset
class-map type inspect c1
    match protocol pop3
policy-map type inspect mypolicy
    class type inspect c1
        inspect
        service-policy inspect pop3 mypop3-policy
```

C3PL CLI

secure-login option in conjunction with reset tears down the connection when clear-text login is seen

POP3 Inspection - Case 10

Existing CLI

```
ip inspect name test pop3 reset
```

C3PL CLI

```
class-map type inspect pop3 pop3-class
    match invalid-command
policy-map type inspect pop3 mypop3-policy
    class type inspect pop3 pop3-class
        reset
class-map type inspect c1
    match protocol pop3
policy-map type inspect mypolicy
    class type inspect c1
        inspect
        service-policy inspect pop3 mypop3-policy
```

The reset option resets the connection when an invalid pop3 command is seen

POP3 Inspection - Case 11

Existing CLI

```
ip inspect name test pop3 alert on reset
```

C3PL CLI

```
class-map type inspect pop3 pop3-class
    match invalid-command
policy-map type inspect pop3 mypop3-policy
    class type inspect pop3 pop3-class
        reset
        alarm
class-map type inspect c1
    match protocol pop3
policy-map type inspect mypolicy
    class type inspect c1
        inspect
        service-policy inspect pop3 mypop3-policy
```

The reset+alarm options resets the connection and spews out a message when an invalid pop3 command is seen

IMAP Inspection

- **Exactly the same options as POP3 inspection**
- **Please refer to the previous 4 slides on POP3 inspection; just replace pop3 with imap (match protocol imap, class/policy-map type inspect imap <name>)**

HTTP Inspection

- **Currently provisioned via ‘appfw’ CLI, which is associated with inspect rule**
- **In C3PL model, provisioned via ‘inspect http’ class/policy-maps**

```
appfw policy-name test
  application http
    max-uri-length 300 action alarm reset

class-map type inspect http c11
  match request uri length gt 300
policy-map type inspect http myhttppolicy
  class type inspect http c11
    alarm
     reset
```

‘match’ equivalent portion of the command goes into http L7 class-map

‘action’ portion goes into the L7 policy-map. Same actions as appfw – alarm, allow, reset supported

Configuration example and mapping of all existing CLIs into C3PL L7 class-map ‘match’ commands in next slides

HTTP Inspection Example - Case 12

Existing CLI

```
appfw policy-name appfw-policy
    application http
        strict-http action alarm reset
ip inspect name test appfw appfw-policy
```

C3PL CLI

```
class-map type inspect http http-class
    match req-rsp protocol-violation
policy-map type inspect http myhttp-policy
    class type inspect http http-class
        alarm
        reset
policy-map type inspect mypolicy
    class type inspect c1
        inspect
        service-policy inspect http myhttp-policy
```

match protocol http

HTTP Inspection appfw and New CLIs

Mapping of appfw CLIs to new CLIs for HTTP inspection

strict-http	match req-resp protocol-violation
content-length minimum <N> maximum <N>	match req-rsp body length lt <N> gt <N>
content-type-verification	match req-rsp header content-type violation
content-type-verification match-req-rsp	match req-rsp header content-type mismatch
content-type-verification unknown	match req-rsp header content-type unknown
max-header-length request <N> response <N>	match {request | response| reg-resp} header length gt <bytes>
port-misuse <im | p2p | tunneling | default>	match request port-misuse <im | p2p | tunneling | any>
transfer-encoding type <chunked | compress | deflate | gzip | identity | default>	match req-rsp header transfer-encoding <chunked|compress|deflate|gzip|identity|any>

HTTP Inspection
appfw and New CLIs

- **Mapping of old CLIs to new CLIs for HTTP inspection**

max-uri-length <N>	**match request uri length gt <N>**
request-method rfc <connect | put | get …> request-method extension <copy | edit…>	**match request method <connect | put | copy …> *Note: method categorization into 'rfc' and 'extension' is now not supported. All methods are displayed at 'match request method ?'***
audit-trail <on | off> timeout <N>	**Not supported under L7 class/policy-map. Needs to be configured in type 'inspect' parameter-map at L3/L4 level**

Java Blocking - Case 13

Existing CLI

```
ip inspect name test http [java-list <ACL-number>]
```

C3PL CLI

```
class-map type inspect http http-class
    match response java-applet
policy-map type inspect http myhttp-policy
    class type inspect http http-class
        reset
policy-map type inspect mypolicy
    class type inspect c1
        inspect
        service-policy inspect http myhttp-policy
```

Only 'reset' action supported for a class configured with 'match java-applet'

class-map type inspect match-all c1
match protocol http
[match access-group N]

URL Filtering - Case 14

Existing CLI

```
ip inspect name test http urlfilter
ip urlfilter server vendor websense 10.0.0.1 port 2030
Ip urlfilter max-request 500
```

C3PL CLI

```
parameter-map type urlfilter myurlf-map
    server vendor websense 10.0.0.1 port 2030
    max-request 500
policy-map type inspect mypolicy
    class type inspect c1
        inspect
        urlfilter myurlf-map
```

class-map type inspect match-all *c1*
match protocol http
[match access-group N]

1. **Provisioned via urlfilter action in inspect policy-map**
2. **'inspect' action MUST be configured before urlfilter**
3. **Class must be configured with 'match protocol http'**
4. **'ip urlfilter' commands now reside in the parameter-map of type urlfilter**

‘Local’ Traffic - Case 15

Existing CLI

```
ip inspect name test tcp router-traffic
interface ethernet0
  ip inspect test in
```

C3PL CLI

```
class-map type inspect local-tcp
    match protocol tcp
policy-map type inspect mylocalpolicy
    class type inspect local-tcp
        inspect

zone-pair name inz-local source in-zone dest self
    service-policy type inspect mylocalpolicy
```

***in-zone* is assumed to include interface ethernet0**

1. ‘Local’ traffic provisioned through concept of ‘self’ zone
2. ‘self’ zone is system-defined
3. ‘self’ can appear as source or destination zone in a zone-pair
4. Validations are performed to check that only allowed protocols (tcp, udp, icmp, H323) can be configured for inspection when self zone is involved

VRF Aware Configurations

- **No VRF configuration in zone/C3PL model**
- **Old inspect rule CLI had ‘vrf’ attribute because ‘global’ inspect parameters had to be provisioned on a per-vrf basis; in C3PL parameters are specified on a per-class basis (via parameter-map). No concept of ‘global**
- **In the C3PL model, internally vrf is deduced from the target (zone-pair/zone/interface) of the policy and is used by Cisco IOS Firewall and Url-filtering as it is done today**

Transparent Firewall Configuration

- **Nothing special to be done in the zone/C3PL model**
- **Add the bridging interface to a zone, configure zone-pair and apply policy. Provisioning model is same as that of ‘normal Layer3’ firewall**
- **As of now ‘ip inspect L2-transparent dhcp-passthrough’ command has not been converted to the C3PL model; for DHCP passthrough, this command is to be used even with C3PL configuration and will apply to all policies applied on bridged interfaces**

NAT and VFR

- **Will continue to be applied on the interface; do not understand zones and will work as they do today**
- **Will work with zone/C3PL inspect policies; no change in the order of feature processing because of zoning/C3PL policy**
- **So, if an inspect policy is configured on a zone-pair, it does not mean that all traffic going from source-zone to destination-zone will be processed identically by other features. Other features (NAT, VFR) will process traffic based on their interface configuration**

Inspect and crypto-map - Case 16

- **Simple 2 interface topology – crypto-map on internet facing serial interface; this shows the 'double-ACL' configuration**

```
ip inspect name test tcp
ip inspect name test http
ip inspect name test icmp
interface ethernet0
      ip inspect test in
interface serial 0
      ip access-group 101 in
      crypto map myvpn
access-list 101 permit esp any any
access-list 101 deny ip any any
```

crypto map *myvpn* local-address serial 0
crypto map *myvpn* 10 ipsec-isakmp
set peer A.B.C.D set transform-set *xxx*
set ip access-group *N* in
match address *M*

Permits ESP, AH and IKE only. Clear-text traffic is filtered using ACL *N* in the cryptomap

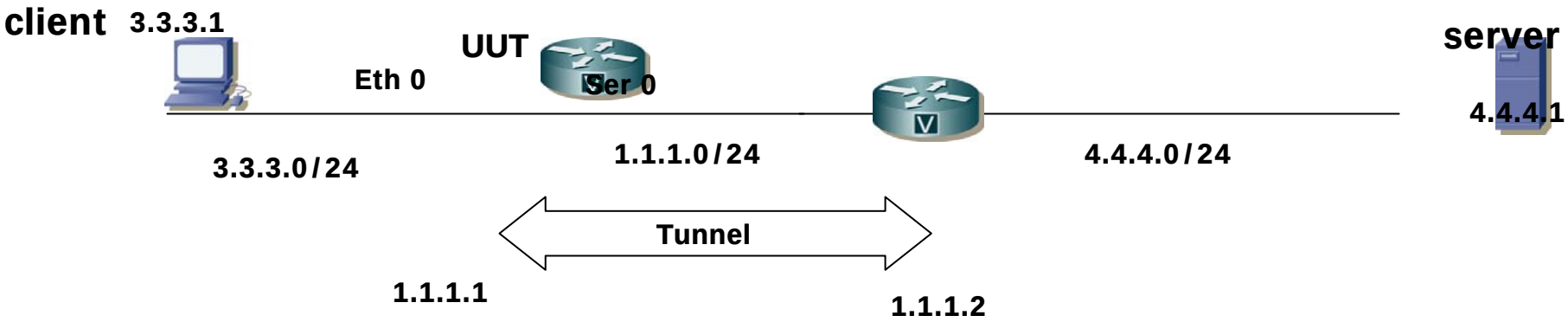
Inspect and crypto-map - Case 16

- **Policy firewall configuration for same case**

```
interface ethernet0
      zone-member security in-zone
interface serial 0
      zone-member security out-zone
      ip access-group 101 in
      crypto map myvpn
zone-pair name in-out source in-zone dest out-zone
      service-policy type inspect mypolicy
```

1. Crypto-map, policy / class-maps not shown for conciseness
2. ACL 101 is to be used to permit ESP, AH, IKE only. It must NOT attempt to filter clear-text traffic
3. The configured policy-map *mypolicy* acts only on clear-text traffic. It inspects connections initiated from inside.
4. Crypto 'double-ACL' NOT required from Cisco IOS Firewall perspective. Cisco IOS Firewall does not punch holes in the crypto ACL also

Tunnel interfaces




```
interface tunnel0
ip address 2.2.2.1
tunnel source 1.1.1.1
tunnel dest 1.1.1.2
ip route 4.4.4.0/24 tunnel0
```

```
interface tunnel0
ip address 2.2.2.2
tunnel source 1.1.1.2
tunnel dest 1.1.1.1
ip route 3.3.3.0/24 tunnel0
```

Problem: Inspect traffic initiated by client (3.3.3.1) to server (4.4.4.1) on UUT

Tunnel Interfaces – Case 17

- **Applying Cisco IOS Firewall on UUT in current style**

```
interface ethernet0
      ip address 3.3.3.2/24
interface serial 0
      ip address 1.1.1.1/24
      ip access-group 102 in
interface tunnel 0
      ip addr 2.2.2.1 255.255.255.0
      tunnel source 1.1.1.1
      tunnel dest 1.1.1.2
      ip inspect test out
      ip access-group 101 in
ip route 4.4.4.0/24 tunnel0
```

```
ip inspect name test tcp
ip inspect name test http
ip inspect name test icmp
```

Permit return tunnel (GRE, ipip) traffic inside

Rule inspects connections initiated from ethernet 0 side

ACL for clear-text (post-decap , pre-encap packets) filtering

Tunnel interfaces – Case 17

Permit return tunnel (GRE, ipip) traffic inside

```
interface ethernet0
     ip address 3.3.3.2/24
     zone-member security in-zone
interface serial 0
     ip address 1.1.1.1/24
     ip access-group 102 in
interface tunnel 0
     ip address 2.2.2.1/24
     tunnel source 1.1.1.1
     tunnel dest 1.1.1.2
     zone-member security out-zone

zone-member security in-out source in-zone dest out-zone
     service-policy type inspect mypolicy
ip route 4.4.4.0/24 tunnel0
```

- Policy *mypolicy* inspects sessions initiated from the 3.3.3.0/24 network which are going into the tunnel.
- ACL for clear-text policy is not needed. The clear-text policy *is mypolicy* itself.
- ACL 102 on serial0 must be configured only to let tunnel traffic (GRE, IPSEC, IPIP) into the router
- When "tunnel protection ipsec" is configured on tunnel 0, the tunnel becomes a crypto tunnel