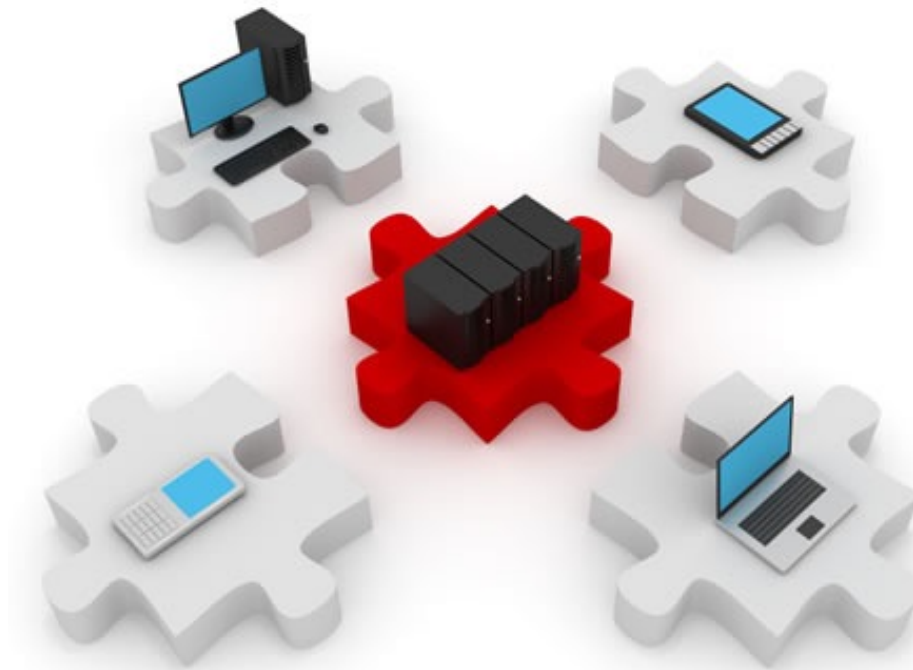


Zone-Based Firewalls. IDS/IPS.

November 11, 2014

What this lecture is about:

- ▶ Zone-based firewalls
- ▶ IDS & IPS



Zone-Based Firewalls

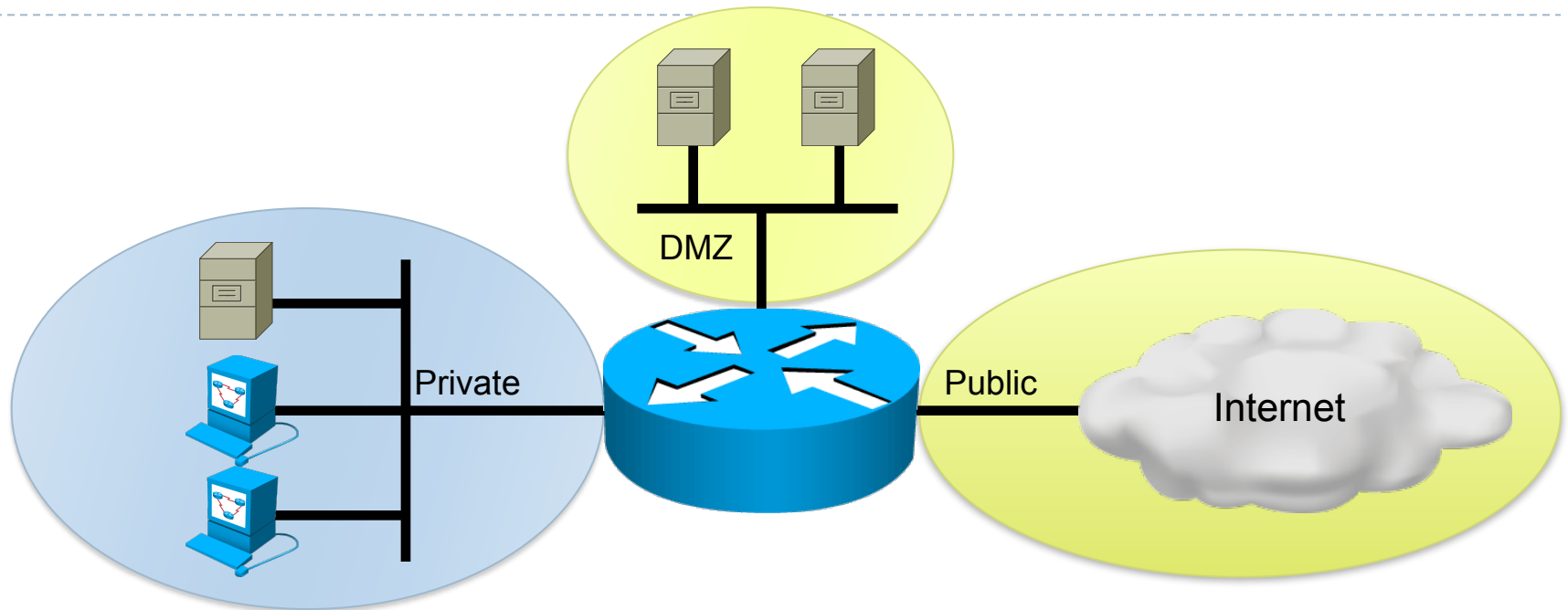
Limitations of CBAC (Classic Firewall)

- ▶ Does not have a hierarchical implementation
- ▶ Limited granularity of firewall policies
- ▶ Not fit for scenarios with more than 2 interfaces
 - ▶ All rules apply to all the traffic on one interface
- ▶ Policies cannot be tied to a group of hosts or a subnet
- ▶ Cannot protect against attacks from the local network
- ▶ Cannot inspect traffic originated from/sent to the router
- ▶ Relies on ACLs

Zone-based policy firewalls (ZPF)

- ▶ Zone-based policy firewall
 - ▶ Interfaces are assigned to zones
 - ▶ Traffic is inspected as it passes between zones
 - ▶ The router blocks all traffic unless explicitly allowed
- ▶ This type of inspection also supports:
 - ▶ Stateful packet inspection
 - ▶ Application-layer inspection
 - ▶ URL filtering
 - ▶ DoS mitigation
- ▶ Cisco Policy Language (CPL)
 - ▶ Hierarchical structure of inspection rules

Zones

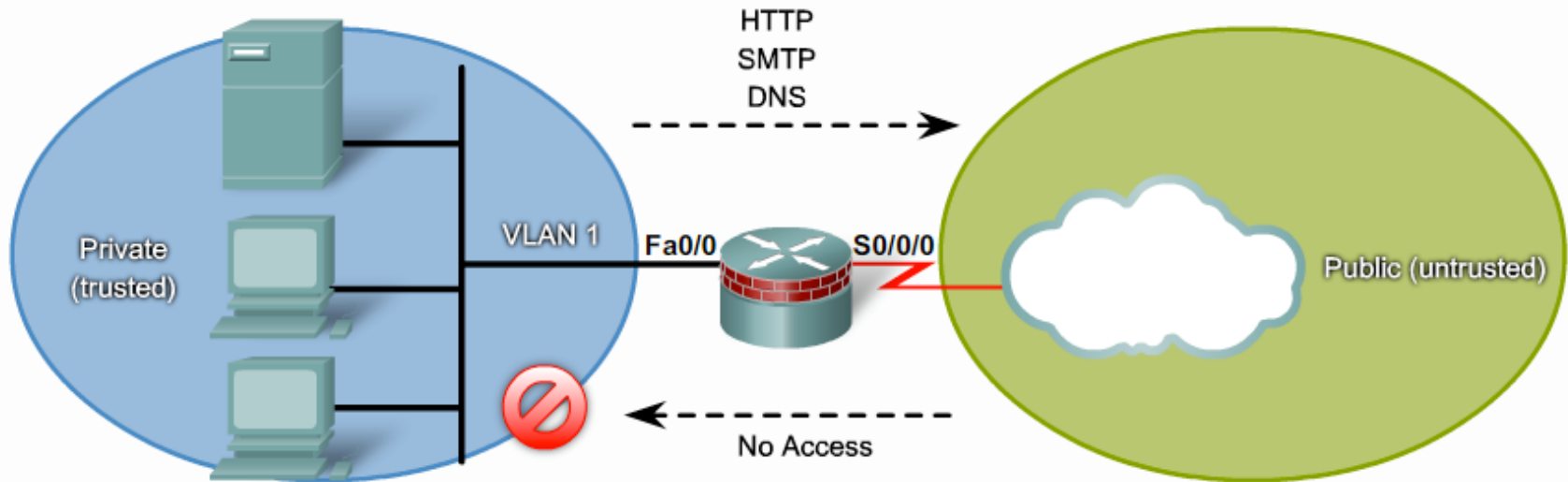


- ▶ Each interface belongs to one zone
- ▶ Policies are applied between zones (zone-pairs)
- ▶ Multiple interfaces connected to the same zone can pass traffic between each other

CBAC and ZPF

- ▶ Can coexist on the same router
- ▶ Cannot coexist on the same interface
 - ▶ One interface cannot be a security zone member and configured for inspection at the same time

Simple two-zone ZPF scenario



- ▶ The internal network should be able to access web, e-mail and DNS services
- ▶ The public network should not have any inbound access

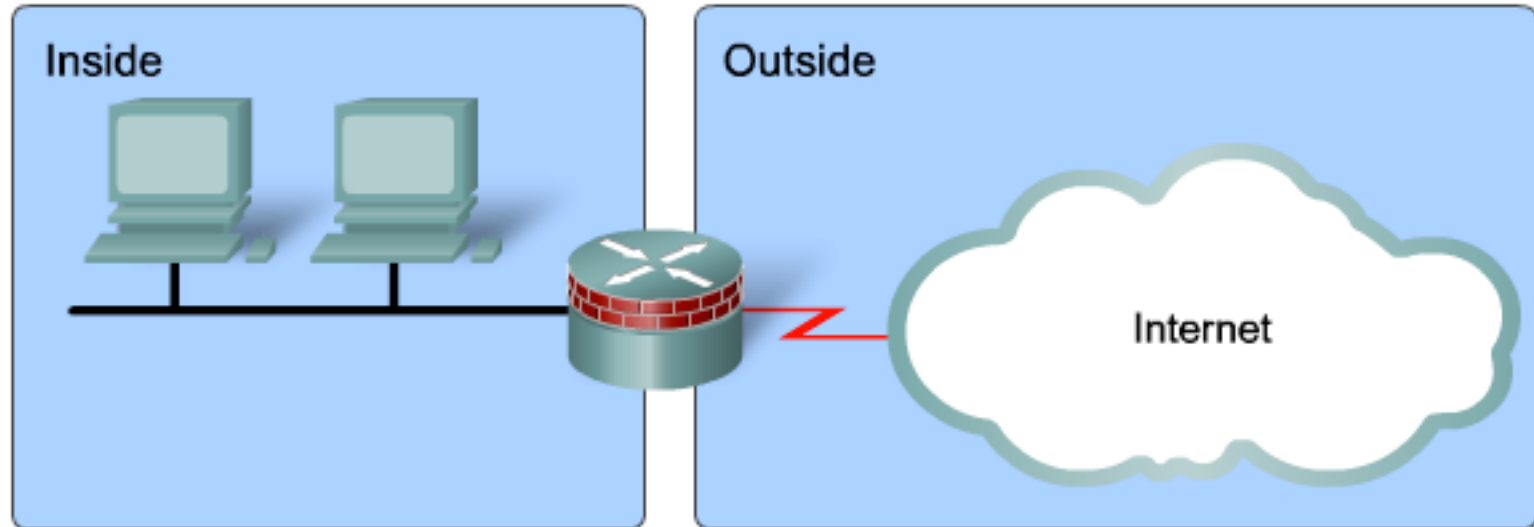
ZPF design steps

- ▶ **Determine the zones**
 - ▶ The entire infrastructure must be separated into zones
 - ▶ Each zone has a specific security level
 - ▶ Zones are designed regardless of physical implementation
- ▶ **Establish policies between zones**
 - ▶ For each “source-destination” pair between two zones
 - ▶ Define accessible destinations
 - ▶ Define services that can be requested
 - ▶ Identify session protocols (TCP, UDP, ICMP)
 - ▶ No physical setup is involved

ZPF design steps continued

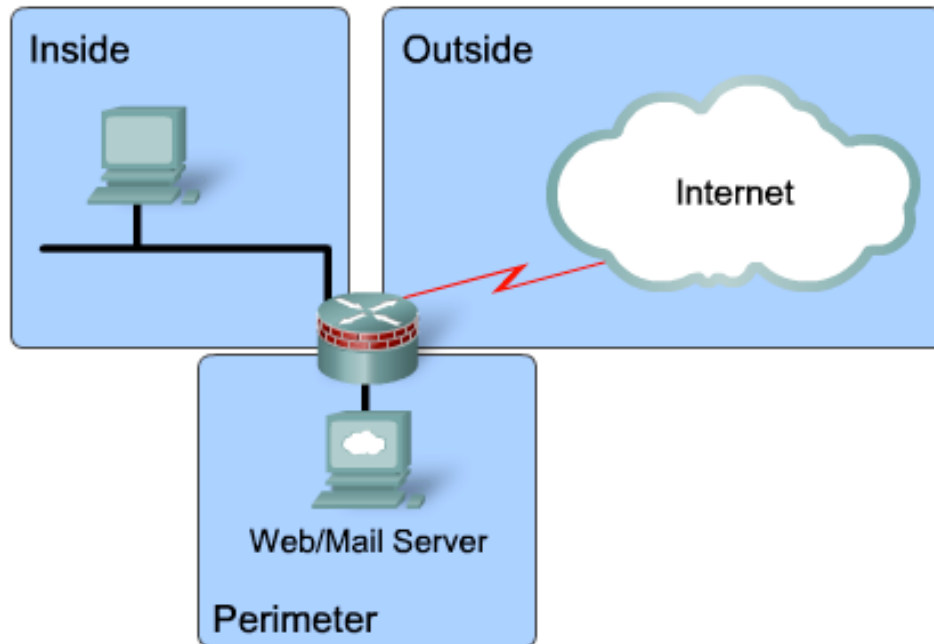
- ▶ Design the physical infrastructure
 - ▶ Take into account security and availability requirements
 - ▶ Decide the number of devices between the least secure zones and the most secure zones
 - ▶ Consider redundancy
- ▶ Identify zone subsets
 - ▶ A zone can have subsets
 - ▶ All subsets are indirectly connected to the same firewall interface
 - ▶ Policies can be defined between subsets, too.
 - ▶ But we won't go that far 😊

ZPF design model: LAN to Internet



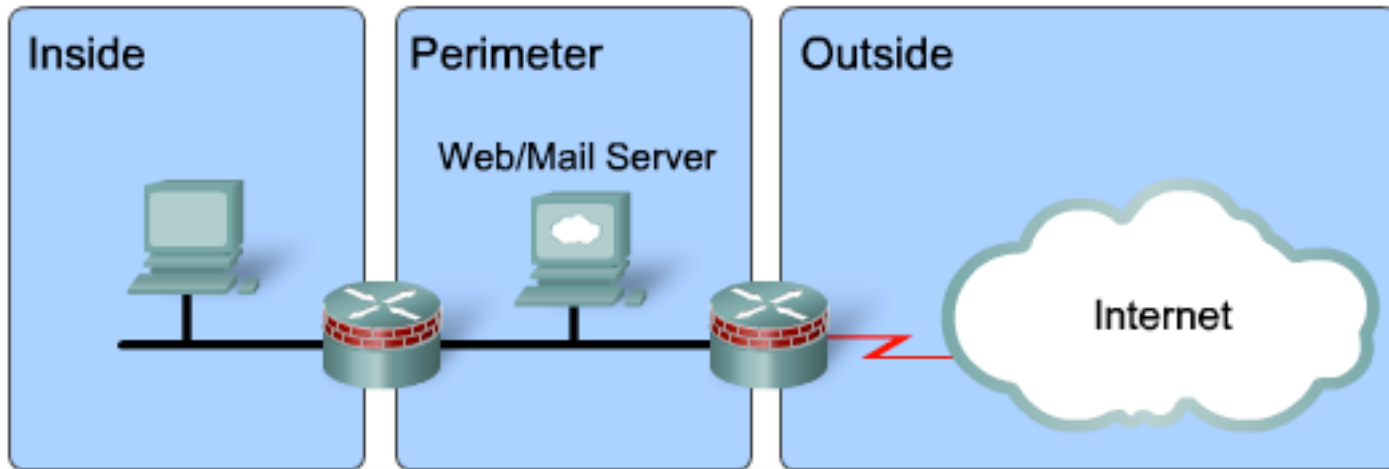
- ▶ No special zones involved
- ▶ Simple physical setup:
 - ▶ One **trusted** interface for the LAN
 - ▶ One **untrusted** interface for the Internet
- ▶ All policies implemented on a single firewall

ZPF design model: Public servers on interface



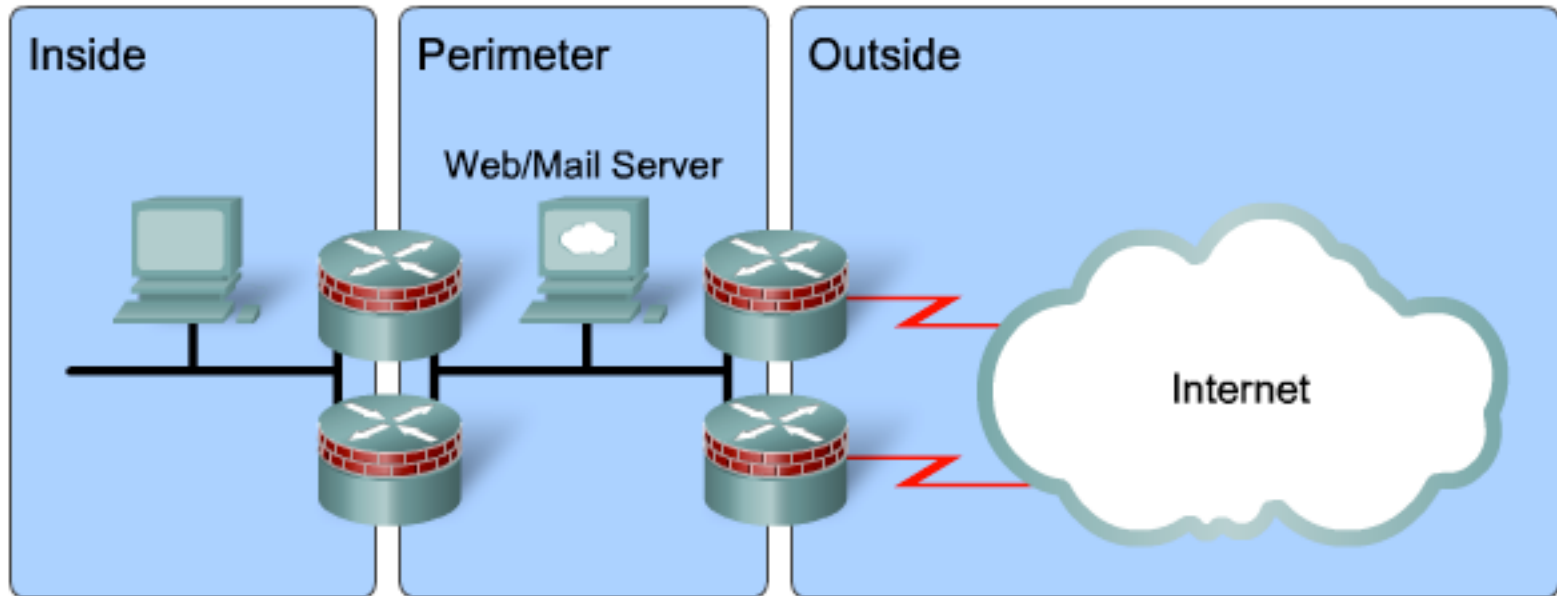
- ▶ The DMZ interface is associated with a special zone
- ▶ The DMZ zone is accessible from the outside
- ▶ Policies prohibit the DMZ from contacting the local network in case it becomes compromised

ZPF design model: Public servers on segment



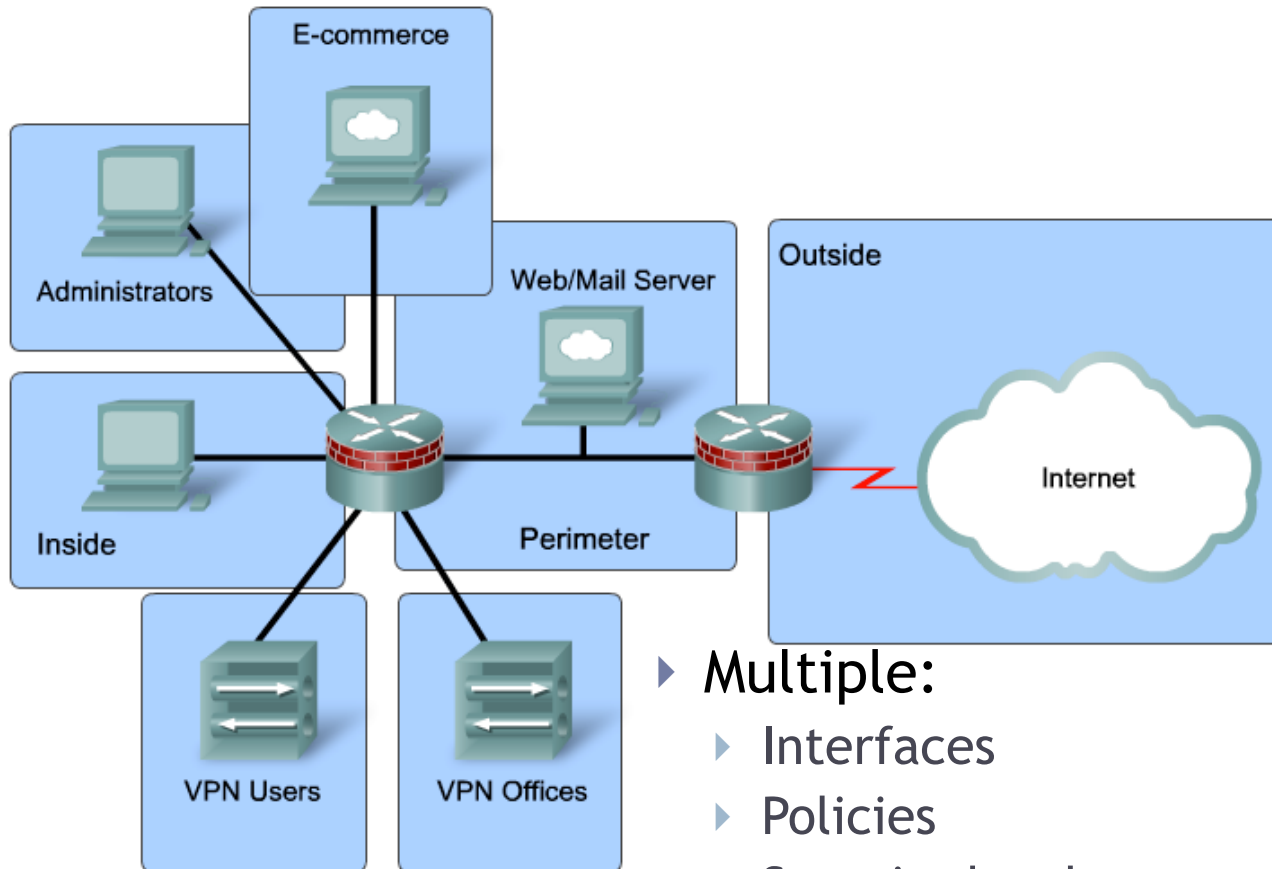
- ▶ Traffic between the untrusted zone and the trusted one must pass through the DMZ
- ▶ Two firewalls involved
- ▶ Can be implemented using layered security
- ▶ Multiple points of failure
- ▶ Different policies for different types of traffic

ZPF design model: Redundant firewalls



- ▶ One DMZ for one or several Internet connections.
- ▶ All interfaces belonging to the same zone implement the same policies
- ▶ Layered approach without single points of failure
- ▶ Load-balancing opportunity

ZPF design mode: Complex firewall



- ▶ Multiple:
 - ▶ Interfaces
 - ▶ Policies
 - ▶ Security levels
- ▶ Single point of failure

ZPF actions

- ▶ The Cisco IOS zone-based firewall can take three actions:
- ▶ **Inspect**
 - ▶ Similar to “ip inspect” from CBAC
 - ▶ Can handle application sessions
- ▶ **Pass**
 - ▶ Similar to “permit” in ACLs
 - ▶ Connection state is not tracked
 - ▶ One-way only
- ▶ **Drop**
 - ▶ Similar to “deny” in ACLs
 - ▶ Silent drop
 - ▶ Dropped packets can be logged

Rules of interfaces and zones

- ▶ Configure the zone before assigning any interfaces
- ▶ An interface can belong to only one security zone
- ▶ For traffic to flow between all interfaces, each must belong to a zone
- ▶ Traffic flows between different zones (and interfaces) must be permitted or inspected by a policy

Rules of interfaces and zones (continued)

- ▶ Interfaces of the same zone allow all traffic between them
- ▶ Interfaces not assigned to a zone can run CBAC
- ▶ If an interface does not need any special policies but has to pass traffic, it can be assigned to a zone with an all-pass policy (dummy policy)

Quick test

| Source interface member of a zone? | Destination interface member of a zone? | Zone-pair is defined? | Is there a policy in place? | Result |
|------------------------------------|---|-----------------------|-----------------------------|---------------|
| NO | NO | N/A | N/A | Normal flow |
| YES | NO | N/A | N/A | Drop |
| NO | YES | N/A | N/A | Drop |
| YES (zone 1) | YES (zone 2) | NO | N/A | Drop |
| YES (zone 1) | YES (zone 2) | YES | NO | Drop |
| YES (zone 1) | YES (zone 2) | YES | YES | Policy action |

Router's traffic

- ▶ Attaching a router's interface to a zone causes all hosts in that network to become members of the zone
- ▶ But the router's interface is not controlled by the zone's policies
 - ▶ Neither inbound nor outbound traffic
- ▶ All router's interfaces are part of the “self” zone
- ▶ To filter traffic going to or originating from the router, policies between other zones and the “self” zone must be implemented
- ▶ This “self” policy does not apply to traffic traversing the router
- ▶ The “self” zone is the only exception to the default “deny all” policy
 - ▶ In the absence of any policy, all traffic is permitted

Steps for configuring ZPF

1. Create the zones
2. Define traffic classes
3. Define firewall policies
4. Apply policy maps to zone pairs
5. Assign router interfaces to zones

1. Creating the zones

- ▶ Create the zones from a security perspective
 - ▶ Interfaces with similar security requirements should be placed in the same zone
 - ▶ Different security policies will require multiple zones

```
Firewall(config)#zone security INSIDE
```

```
Firewall(config-sec-zone)#description Our local network
```

```
Firewall(config)#zone security OUTSIDE
```

```
Firewall(config-sec-zone)#description Internet connection
```

2. Define traffic classes

- ▶ Traffic classes allow you to define traffic flows in a granular fashion

- ▶ Example:

```
Firewall(config)#access-list 101 permit ip 192.168.0.0 0.0.0.255 any
Firewall(config)#class-map type inspect match-all EXAMPLEMAP
Firewall(config-cmap)#match access-group 101
Firewall(config-cmap)#match protocol telnet
```

- ▶ The syntax for creating ZPF traffic classes

- ▶ Inspecting layers 3 and 4:

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

- ▶ Inspecting the application layer:

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

2. Layer-7 class-maps (application inspection)

Firewall(config)#class-map type inspect ?

| WORD | class-map name |
|-----------|---|
| aol | Configure CBAC class-map for IM-AOL protocol |
| edonkey | eDonkey |
| fasttrack | FastTrack Traffic - KaZaA, Morpheus, Grokster... |
| gnutella | Gnutella Version2 Traffic - BearShare, Shareeza, Morpheus ... |
| http | Configure CBAC class-map for HTTP protocol |
| imap | Configure CBAC class-map for IMAP protocol |
| kazaa2 | Kazaa Version 2 |
| match-all | Logical-AND all matching statements under this classmap |
| match-any | Logical-OR all matching statements under this classmap |
| msnmsgr | Configure CBAC class-map for IM-MSN protocol |
| pop3 | Configure CBAC class-map for POP3 protocol |
| smtp | Configure CBAC class-map for SMTP protocol |
| sunrpc | Configure CBAC class-map for RPC protocol |
| ymsgr | Configure CBAC class-map for IM-YAHOO protocol |

2. Layer-4 class-maps

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

- ▶ The syntax for referencing an ACL from the class map

```
Firewall(config-cmap)# match access-group {access-group | name  
access-group-name}
```

- ▶ Matching protocols from within the class map
 - ▶ TCP, UDP, ICMP or application services (HTTP, SMTP, etc.)

```
Firewall(config-cmap)# match protocol protocol-name
```

- ▶ Matching other class maps from within the class map

```
Firewall(config-cmap)# match class-map class-map-name
```

3. Define firewall policies

► Example:

```
Firewall(config)#policy-map type inspect INSIDE_TO_OUTSIDE
Firewall(config-pmap)#class type inspect EXAMPLEMAP
Firewall(config-pmap-c)#?
```

Policy-map class configuration commands:

| | |
|----------------|---|
| drop | Drop the packet |
| exit | Exit from class action configuration mode |
| inspect | Context-based Access Control Engine |
| no | Negate or set default values of a command |
| pass | Pass the packet |
| police | Police |
| service-policy | Deep Packet Inspection Engine |
| urlfilter | URL Filtering Engine |
| <cr> | |

```
Firewall(config-pmap-c)#inspect
```

%No specific protocol configured in class EXAMPLEMAP for inspection. All protocols will be inspected



Policy options

► The default class matching all remaining traffic (drop):

```
Firewall(config-pmap)#class class-default
```

4. Apply policy maps to zone pairs

- ▶ The firewall policies are applied to traffic between two zones (a “zone-pair”)

- ▶ Create zone-pair

- ▶ Define the **source** and **destination** zones:

```
Firewall(config)#zone-pair security IN_OUT_ZONE_PAIR source INSIDE  
destination OUTSIDE
```

- ▶ “self” can be used as a zone name here

- ▶ Apply **policy-map** to zone-pair:

```
Firewall(config-sec-zone-pair)#service-policy type inspect  
INSIDE_TO_OUTSIDE
```

- ▶ Add a **description** for the zone-pair:

```
Firewall(config-sec-zone-pair)#description Going outside
```

5. Assigning interfaces

- ▶ Interfaces must be assigned to the appropriate security zones:

```
Firewall(config)#interface FastEthernet0/0
```

```
Firewall(config-if)#zone-member security INSIDE
```

```
Firewall(config)#interface Serial0/1/1
```

```
Firewall(config-if)#zone-member security OUTSIDE
```

ZPF final configuration

- ▶ Access list to define traffic for inspection:

```
access-list 101 permit ip 192.168.0.0 0.0.0.255 any
```

- ▶ Class map defining a **traffic class**:

```
class-map type inspect match-all EXAMPLEMAP  
  match access-group 101  
  match protocol telnet
```

- ▶ Policy map setting the “inspect” **action** on the specified **traffic class**:

```
policy-map type inspect INSIDE_TO_OUTSIDE  
  class type inspect EXAMPLEMAP  
    inspect  
  class class-default
```

ZPF final configuration continued

- ▶ Defining two security zones:

```
zone security INSIDE  
  description Our local network
```

```
zone security OUTSIDE  
  description Internet connection
```

- ▶ Defining a **zone pair** between these two zones to specify a **policy map** for all traffic:

```
zone-pair security IN_OUT_ZONE_PAIR source INSIDE destination  
OUTSIDE  
  description Going outside  
  service-policy type inspect INSIDE_TO_OUTSIDE
```

Testing ZPF

- ▶ Session established after a successful Telnet attempt through the firewall:

```
Firewall#show policy-map type inspect zone-pair sessions
```

```
Zone-pair: IN_OUT_ZONE_PAIR
```

```
Service-policy inspect : INSIDE_TO_OUTSIDE
```

```
Class-map: EXAMPLEMAP (match-all)
```

```
Match: access-group 101
```

```
Inspect
```

```
Established Sessions
```

```
Session 65DA2000 (192.168.0.2:59848)=>(199.0.0.2:23) telnet SIS_OPEN
```

```
Created 00:00:04, Last heard 00:00:02
```

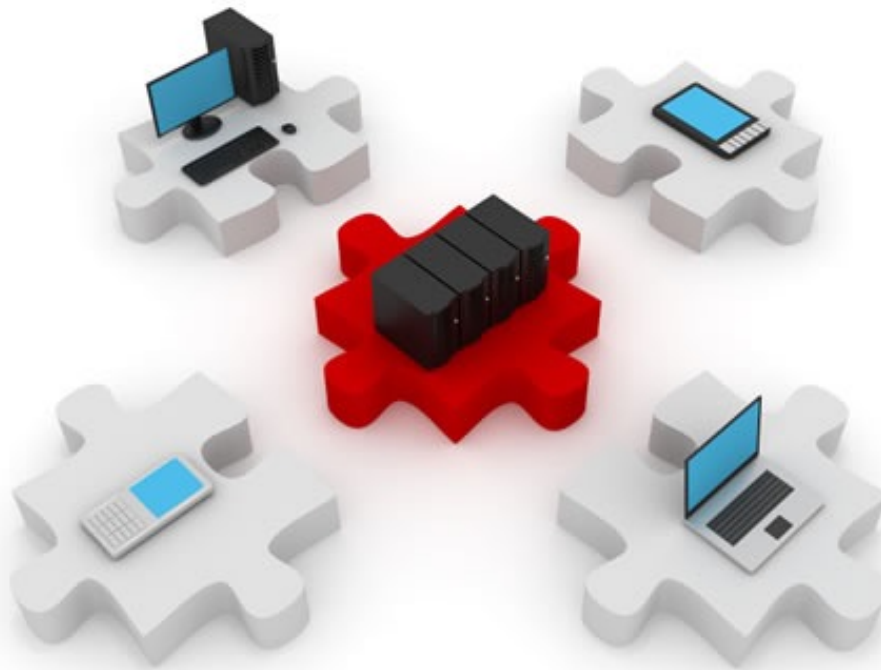
```
Bytes sent (initiator:responder) [37:80]
```

```
Class-map: class-default (match-any)
```

```
Match: any
```

```
Drop (default action)
```

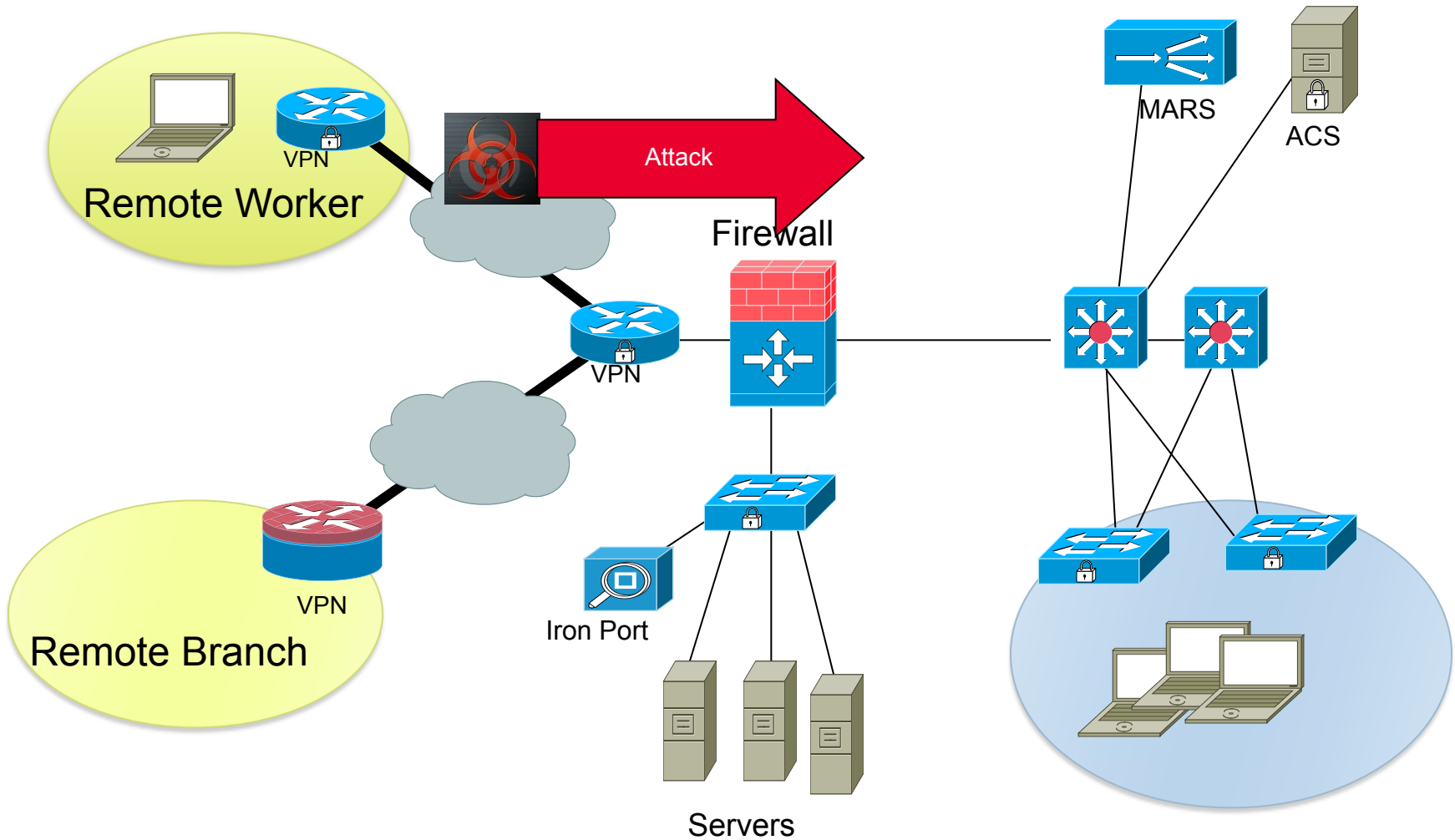
```
0 packets, 0 bytes
```



IDS & IPS

Feel the beat...

Network intrusions



Zero-day attacks

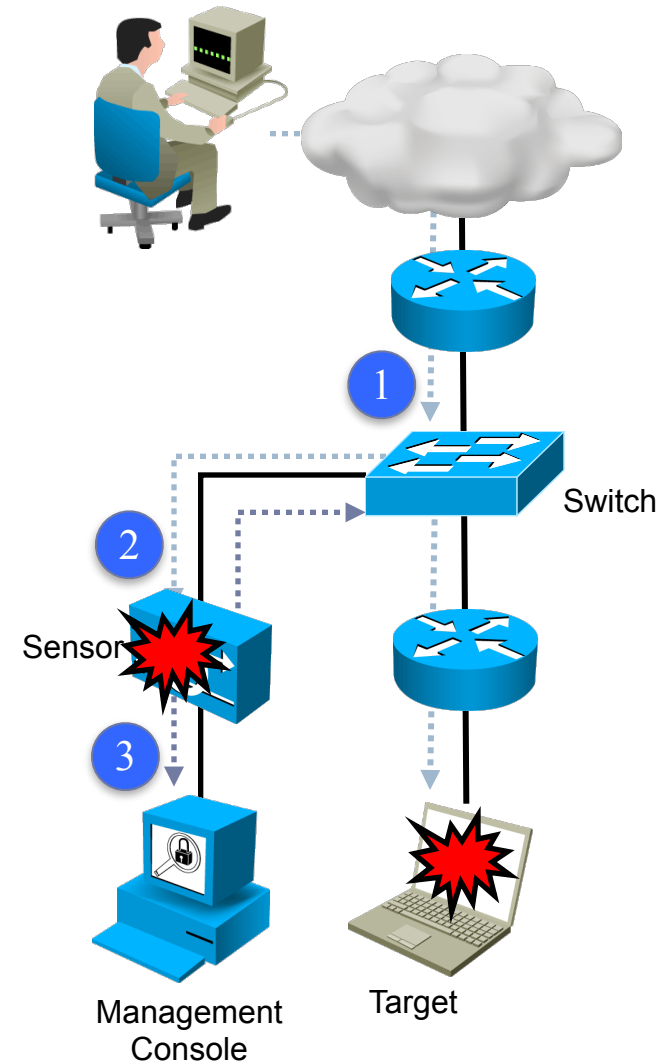
- ▶ A zero-day attack/threat/exploit attacks vulnerabilities unknown (yet) to the software vendor
- ▶ During the time it takes the software vendor to develop and release a patch, all networks are vulnerable to this exploit
- ▶ A firewall can only protect against known and well-documented threats and anomalies
- ▶ Defending against these kind of attacks requires a different perspective

Intrusion Detection System (IDS)

- ▶ Copy (“mirror”) the traffic stream from your network
 - ▶ Send it to an IDS device
 - ▶ The device analyses the traffic in real time
 - ▶ Detects attacks
-
- ▶ It is considered a passive device (it only listens)
 - ▶ Runs in promiscuous mode (receives all traffic)
 - ▶ DOES NOT analyse the actual forwarded packets
 - ▶ Only copies of them
 - ▶ What does this mean?
 - ▶ It can **ONLY DETECT**, **NOT PREVENT** attacks!

IDS behaviour

- ▶ 1: An attack is launched from outside the network. Traffic is mirrored and sent to the sensor, too
- ▶ 2: The IDS sensor matches the traffic with a signature and sends the switch a command to deny further similar traffic.
 - ▶ The IDS experiences the same attack as the hosts in the network.
- ▶ 3: The IDS sensor sends a log message to a management console

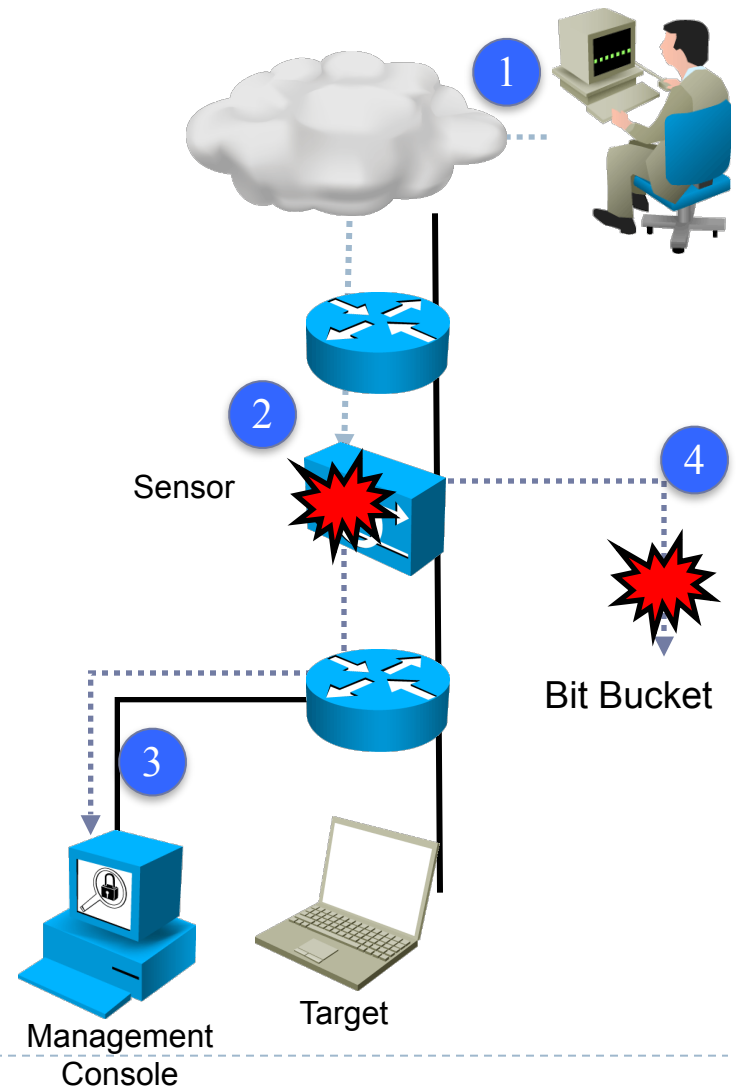


Intrusion Prevention System (IPS)

- ▶ An IPS device does mainly what an IDS does, too
- ▶ Except it is located elsewhere:
 - ▶ An IPS is located inline with the traffic flow
- ▶ The IPS responds immediately to a threat, by blocking traffic
 - ▶ The IDS cannot block traffic

IPS behaviour

- ▶ 1: An attack is launched from outside the network. Traffic goes directly to the sensor.
- ▶ 2: The IPS sensor analyzes the packets. If a signature matches, traffic is stopped immediately.
- ▶ 3: The IPS sensor notifies a management console of the event.
- ▶ 4: Further traffic in violation of certain policies can be dropped immediately.



IPS & IDS characteristics

- ▶ A sensor can be implemented as:
 - ▶ A router configured with Cisco IOS IPS software
 - ▶ A dedicated device that provides IPS/IDS services
 - ▶ A network module installed in an ASA, switch or router
- ▶ In general they use signatures to detect potential harmful traffic patterns
- ▶ Patterns detected can be:
 - ▶ Atomic - single packets identified as attacks
 - ▶ Composite - sequences of packets that form an attack

IDS advantages and disadvantages

- ▶ Zero impact on normal network performance (latency, jitter, etc.)
- ▶ No impact on network functionality if the sensor fails or is overloaded
- ▶ Cannot stop the trigger packet(s) and may not stop the connection
- ▶ Does not provide fast response time
- ▶ Vulnerable to evasion techniques



IPS advantages and disadvantages

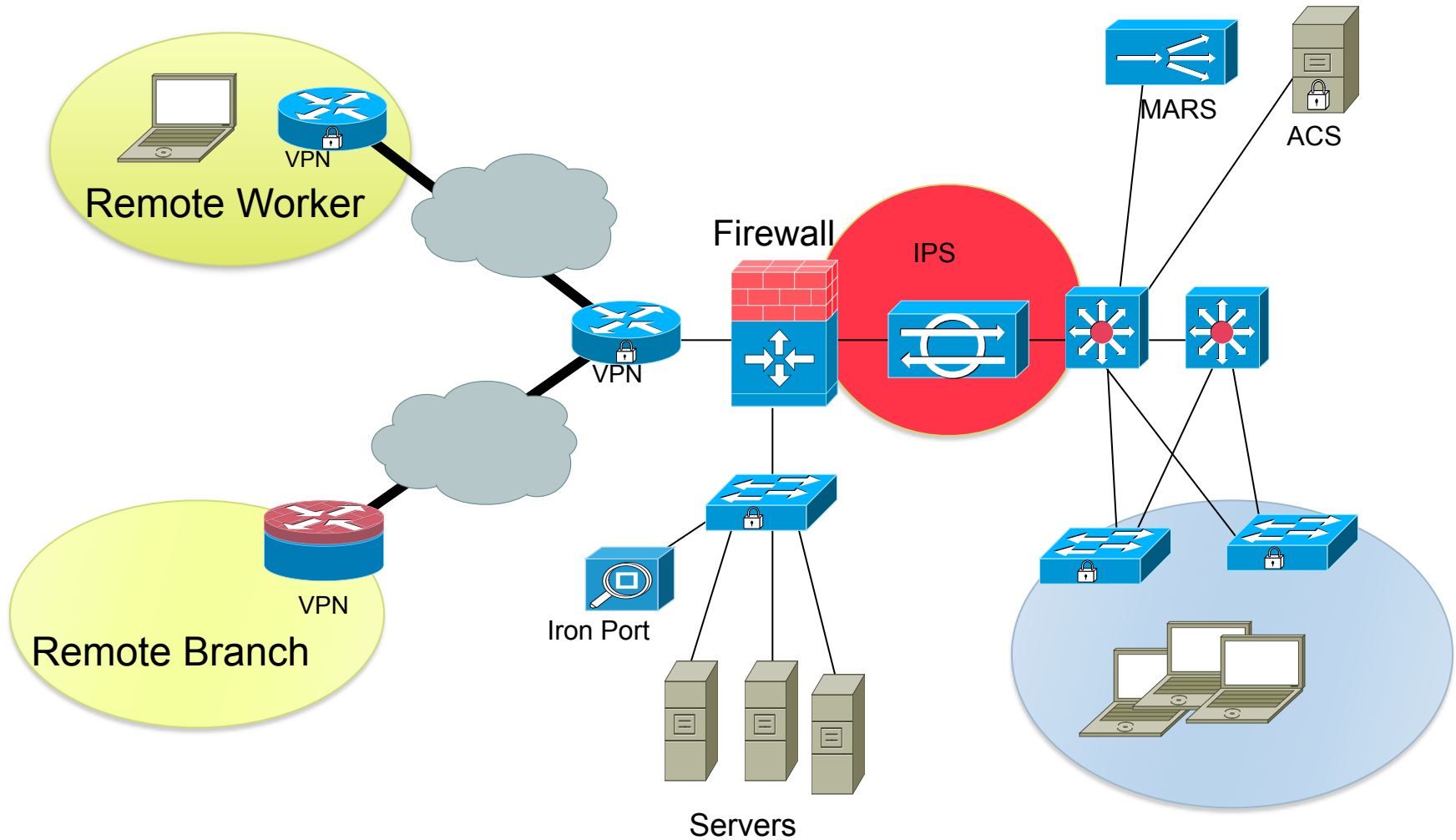
- ▶ Can stop malicious traffic (packet, connection)
- ▶ Can apply stream normalization to reduce evasion attempts
 - ▶ Abnormal streams can be used to confuse an IDS/IPS.
- ▶ Single point of failure
 - ▶ Sensor issues affect network functionality (failures or overloading)
- ▶ Fine-tuned policy required to avoid false positives
- ▶ Some impact on network performance



Deployment methods: NIPS and HIPS 😊

- ▶ **NIPS - Network-based IPS implementation**
 - ▶ Located between trusted and untrusted networks
 - ▶ Analyze network-wide activity
 - ▶ Deployed using ASA, routers and switches
- ▶ **HIPS - Host-based IPS implementation**
 - ▶ Installed on individual computers as a software agent
 - ▶ Supervise network activity, file systems, OS resources
 - ▶ Does not provide a global view of the network
 - ▶ Acts like a network/application firewall+antivirus software
 - ▶ Example: Cisco Security Agent (CSA)

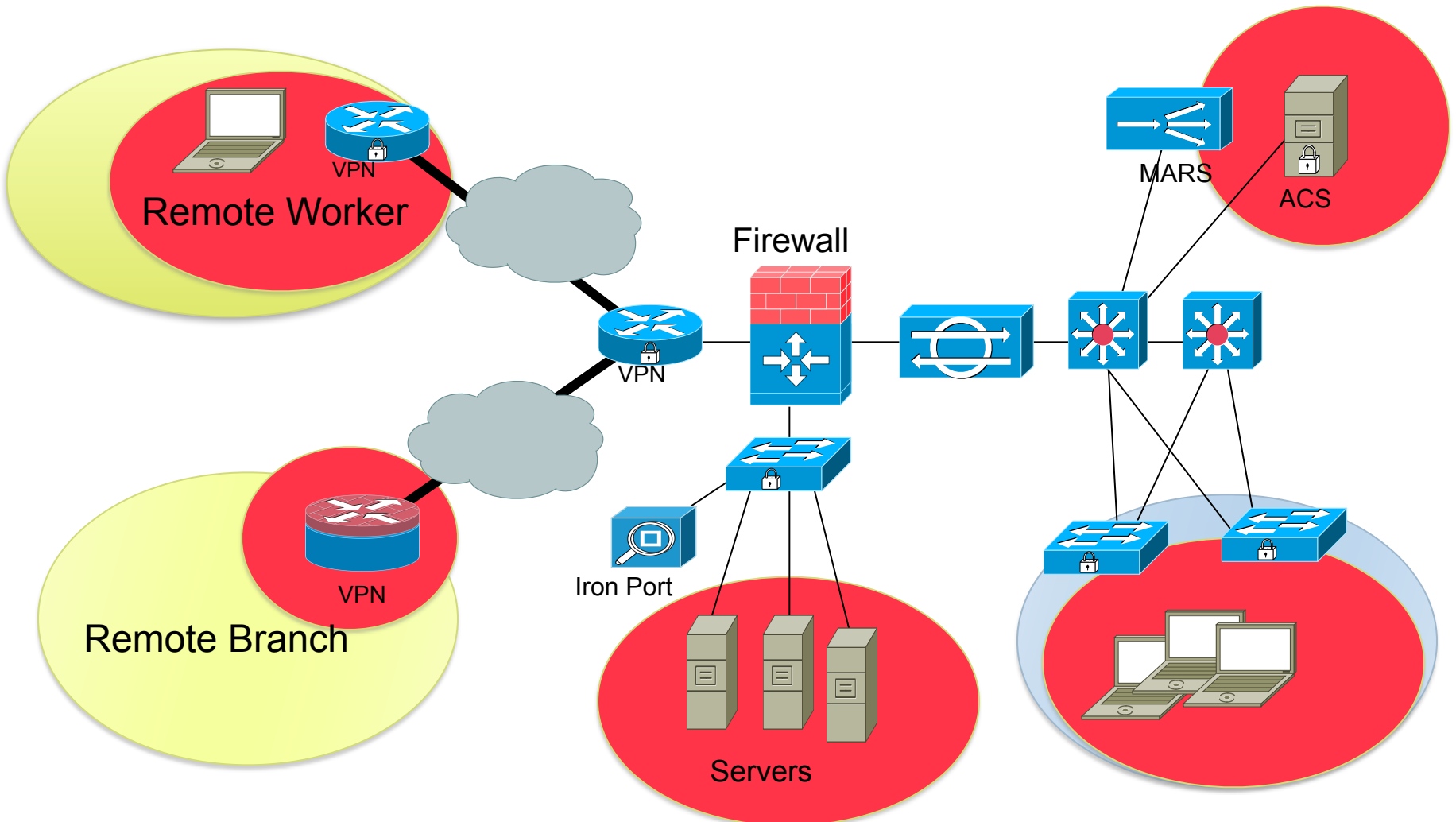
NIPS deployment example



NIPS

- ▶ NIPS implemented as a dedicated hardware device requires:
 - ▶ A NIC (Network Interface Card) adequate to the network medium (FastEthernet, Gigabit, etc)
 - ▶ Processor: real-time pattern matching between traffic and signatures require processing power
 - ▶ Memory: intrusion detection analysis is memory-intensive
- ▶ The device is transparent to the network and its users
- ▶ Is not dependent on network operating systems
- ▶ More cost-effective
- ▶ Cannot examine encrypted traffic
- ▶ Does not know whether an attack was successful or not

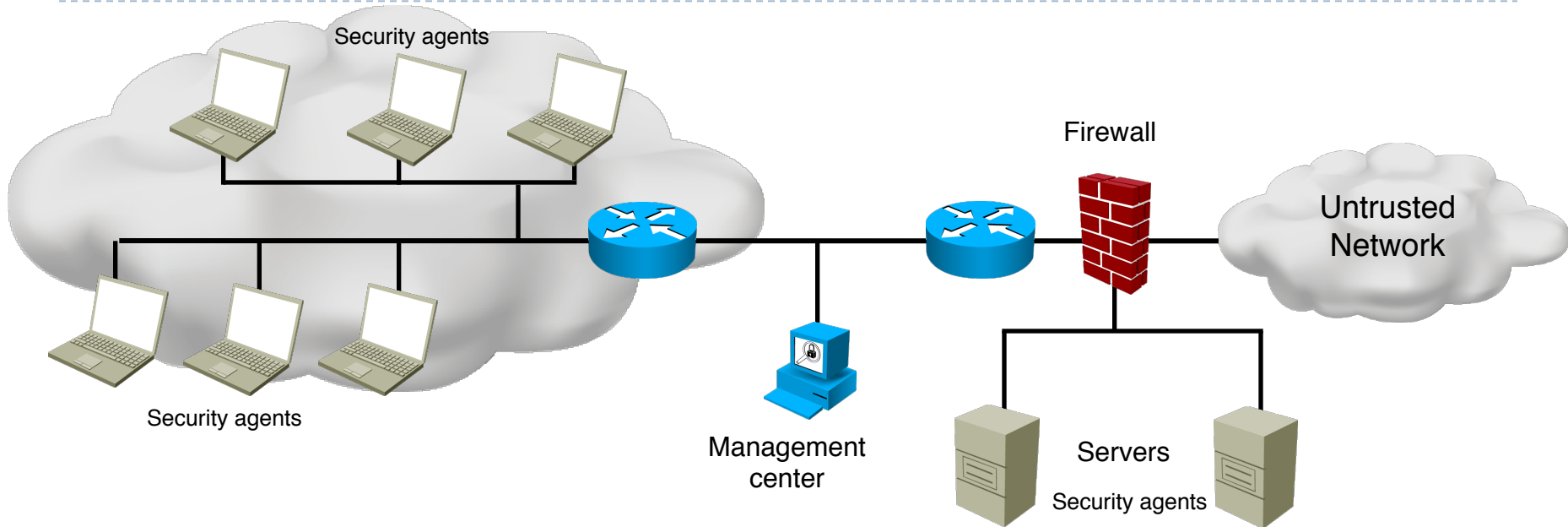
HIPS deployment example



HIPS

- ▶ A software application running on top of the OS
 - ▶ Must be run on every system on the network
 - ▶ Requires complete administrative access to the system
 - ▶ Can query the user for specific actions
 - ▶ Each decision affects only the local system
- ▶ Can immediately determine the success or failure of an attack
- ▶ Traffic received by HIPS is unencrypted
- ▶ Operating system dependent
- ▶ Runs with limited resources (one host)
- ▶ Cannot detect lower level network events

Cisco Security Agent components



- ▶ **Management center:** installed on central server, managed by system administrator
- ▶ **Security agents:** installed on all host systems
 - ▶ Constant monitoring activity

Types of IDS/IPS systems

- ▶ **IDS/IPS sensors can use four types of detection:**
 - ▶ Signature-based detection
 - ▶ Anomaly-based detection
 - ▶ Policy-based detection
 - ▶ Honeypot-based detection

Signature-based detection

- ▶ Searches for a predefined pattern
- ▶ Network traffic is cross-referenced with a database of known attacks.
- ▶ Simple to employ
- ▶ Cannot be used to detect unknown attacks
- ▶ Many false positives
- ▶ Atomic signature:
 - ▶ Detecting an ARP request with the FF:FF:FF:FF:FF:FF source address
- ▶ Composite signature:
 - ▶ Searching for a character string in a sequence of TCP Telnet packets

Policy-based detection

- ▶ Based on the network security policy
- ▶ Any traffic outside the policy will generate an alarm
- ▶ Example:
 - ▶ Detect a port sweep: number of ports scanned on a host exceeds a threshold
- ▶ Threshold levels depend on the actual utilization patterns of the network
- ▶ Requires detailed knowledge of the network traffic
- ▶ May use statistical analysis of the traffic flow

Anomaly-based detection

- ▶ Requires defining a profile that is considered “normal”
 - ▶ The network must be free of attacks when being initially inspected (learning phase)
 - ▶ Regarding traffic amount, protocol types, session initiation frequency, etc.
- ▶ Can detect new and unpublished (zero-day) attacks
- ▶ Can also generate many false positives
- ▶ As the network evolves, the definition of “normal” must be constantly updated
- ▶ Harder to track down the specific attack
 - ▶ Only indicates that abnormal traffic patterns were detected

Honeypot-based detection

- ▶ Uses a dummy server to attract attacks
- ▶ From the outside, the server looks like a valuable, yet vulnerable host, ready to be compromised
- ▶ The server concentrates and logs all attacks
- ▶ The logs can be analyzed to create new types of signatures

IPS signatures

- ▶ To stop an attack, you must be able to identify it
 - ▶ How to tell apart an attack from regular network traffic?
- ▶ Signatures
 - ▶ A set of rules used by IPS and IDS to detect typical intrusive activities
 - ▶ They “describe” attacks such as:
 - ▶ Viruses, worms
 - ▶ DoS attacks
 - ▶ Flooding attacks
 - ▶ Spoofing attacks
 - ▶ Known exploits

IPS signature characteristics

- ▶ Signatures are stored in signature files
- ▶ These files are uploaded to IPS devices periodically
- ▶ SME (Signature Micro-Engines) are compiled groups of signatures
 - ▶ Used by Cisco IOS to improve scanning speed by searching for multiple signatures at once
- ▶ Signature files can be published weekly or even hours after an attack identification
- ▶ Incremental updates
 - ▶ Example: IOS-S361-CLI.pkg after IOS-S360-CLI.pkg

Signature types

- ▶ **Atomic**

- ▶ Single packet or event
- ▶ Does not require state information tracking
- ▶ Ex: spoofed, malformed packet

- ▶ **Composite**

- ▶ Stateful signature - tracks an entire connection
- ▶ Time to track a connection: event horizon

- ▶ **Components of a signature:**

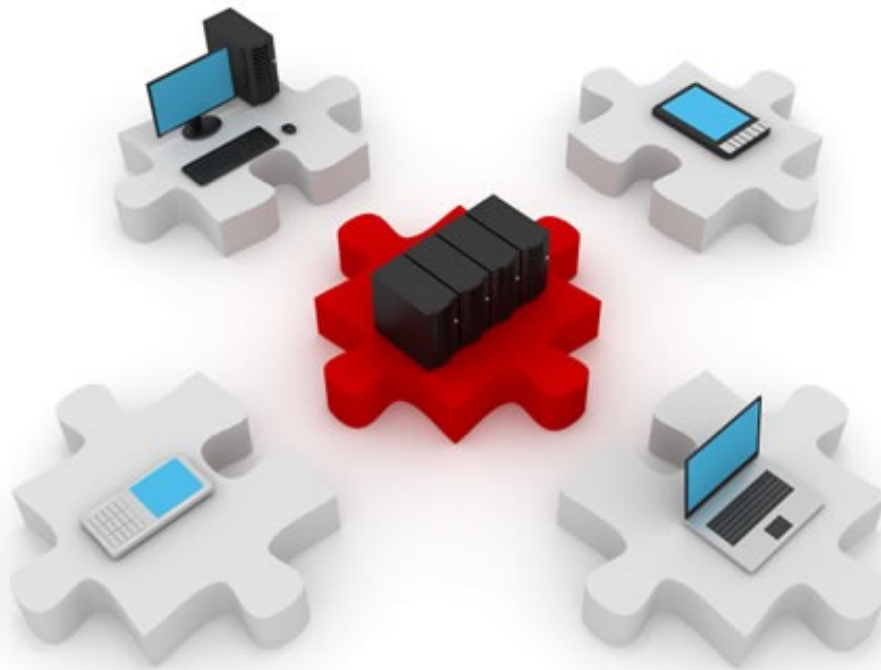
- ▶ Type (classification)
- ▶ Trigger (what kind of traffic triggers the signature action)
- ▶ Action (the action taken with the specific traffic)

IPS signature trigger

- ▶ Certain packet parameters or a packet sequence that indicates a known attack
- ▶ Values in IP, TCP, UDP, ICMP headers
- ▶ Fixed sequence of bytes
- ▶ More complex conditions
- ▶ Example: <the packet is IPv4 and TCP> and <the destination port is 2222> and <the payload contains the string “foo”>

IPS signature actions

- ▶ **Generate an alert**
 - ▶ Store locally or send an event through the network
- ▶ **Log the activity (packets)**
 - ▶ Log attacker, victim or both types of packets.
- ▶ **Drop or prevent activity**
 - ▶ Temporarily or permanently deny further traffic
 - ▶ Drop on a per-packet basis or forcefully close TCP connection
- ▶ **Block future activity**
 - ▶ A request to a switch or router can be sent to deny a certain type of traffic
- ▶ **Allow traffic**

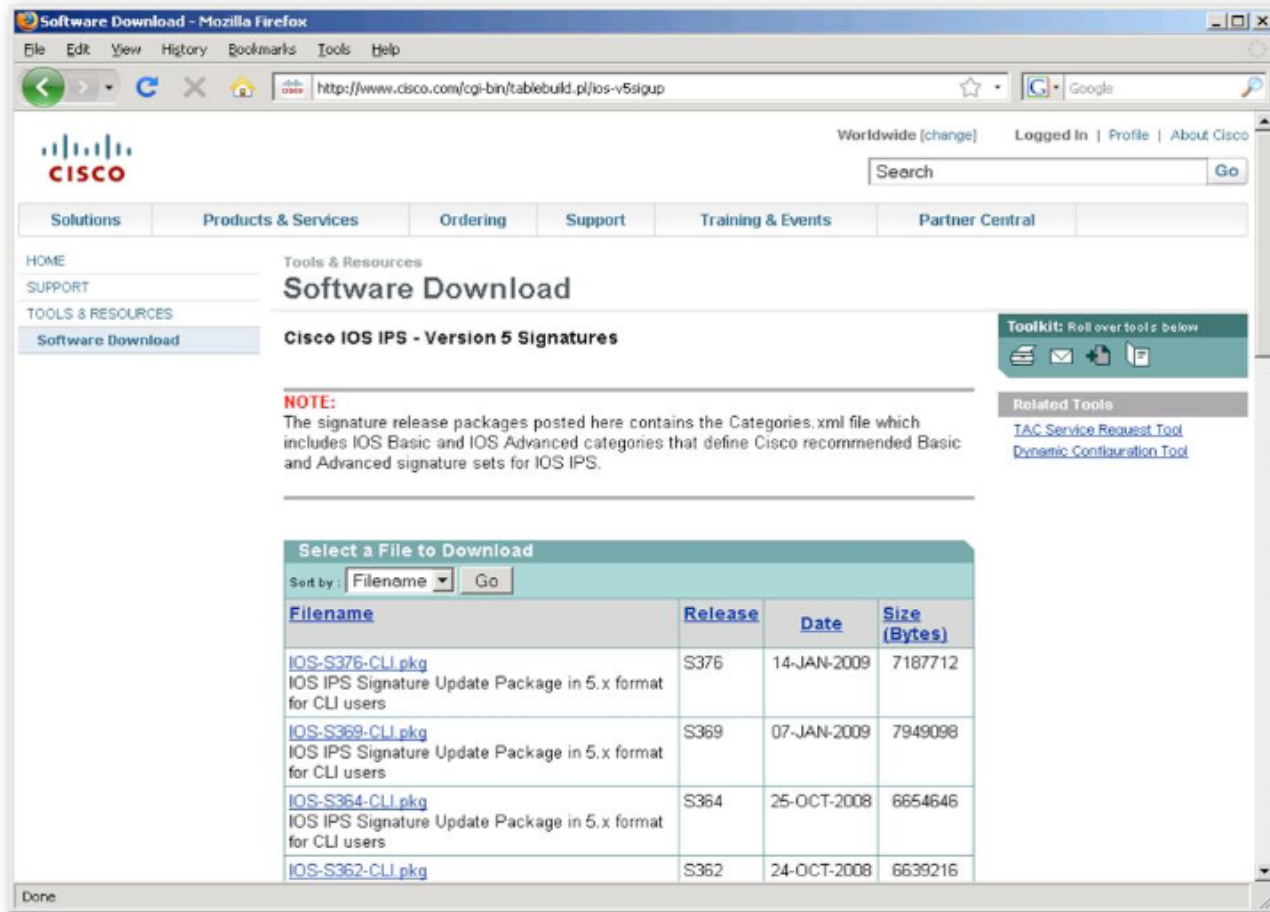


Configuring Cisco IOS IPS

Steps for implementing IOS IPS

- ▶ 1: Download the IOS IPS file
- ▶ 2: Create an IOS IPS configuration directory in Flash
- ▶ 3: Configure an IOS IPS crypto key
- ▶ 4: Enable IOS IPS
- ▶ 5: Load the IOS IPS signature package into the router

Downloading the IOS IPS file: Cisco.com



The screenshot shows the Cisco Software Download page for IOS IPS Version 5 Signatures. The page includes a navigation bar with links like Solutions, Products & Services, Ordering, Support, Training & Events, and Partner Central. A search bar is also present. The main content area features a 'Software Download' section with a 'Cisco IOS IPS - Version 5 Signatures' heading. A 'NOTE' states that the signature release packages contain a Categories.xml file. Below this, a 'Select a File to Download' section contains a table of available files.

| Filename | Release | Date | Size (Bytes) |
|--|---------|-------------|--------------|
| IOS-S376-CLI.pkg IOS IPS Signature Update Package in 5.x format for CLI users | S376 | 14-JAN-2009 | 7187712 |
| IOS-S369-CLI.pkg IOS IPS Signature Update Package in 5.x format for CLI users | S369 | 07-JAN-2009 | 7949098 |
| IOS-S364-CLI.pkg IOS IPS Signature Update Package in 5.x format for CLI users | S364 | 25-OCT-2008 | 6654646 |
| IOS-S362-CLI.pkg | S362 | 24-OCT-2008 | 6639216 |

- ▶ **IOS-Sxxx-CLI.pkg** - the signature package
- ▶ **realm-cisco-pub-key.txt** - the public crypto key used by IOS IPS

Create a configuration directory in Flash

- ▶ Create a directory in Flash to store the signature files and configurations:

```
R1#mkdir ips
```

```
Create directory filename [ips]?
```

```
Created dir flash:ips
```

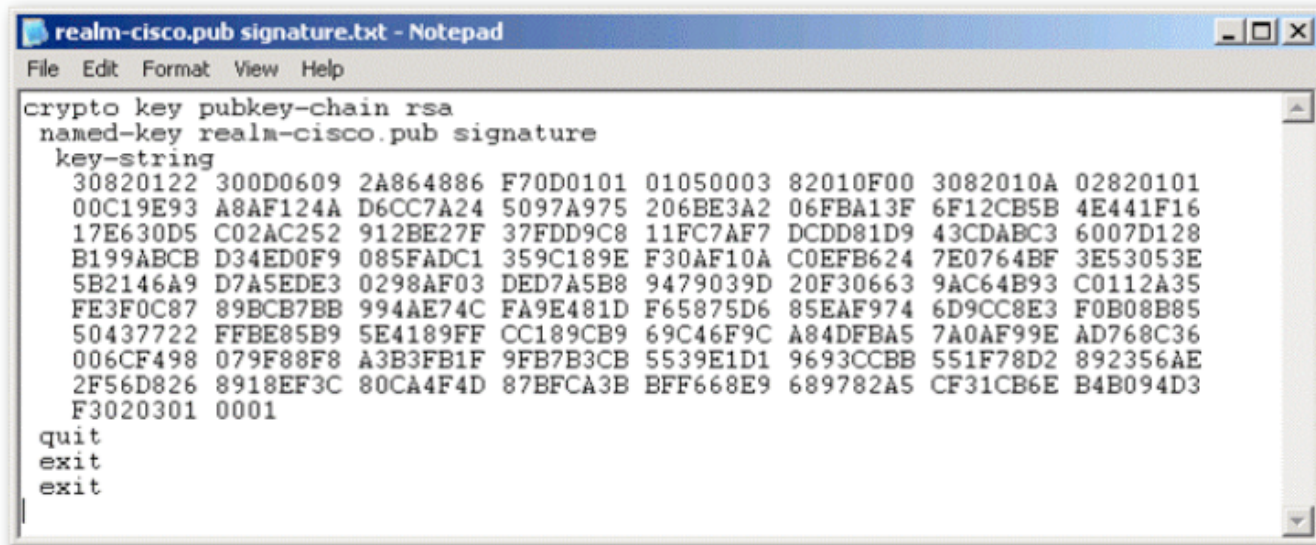
```
R1#dir flash:
```

```
Directory of flash:/
```

```
  2  -rw-          1652  Aug 13 2009 11:59:54 +00:00  
pre_autosec.cfg  
  3  -rw-          1015   Nov 6 2009 16:30:22 +00:00  srs_ac.cfg  
  5  drw-           0   Nov 7 2009 16:28:16 +00:00  ips  
  4  -rw-    30588892  Nov 10 2007 16:13:02 +00:00  c2801-  
advipservicesk9-mz.124-9.T.bin
```

```
64016384 bytes total (33415168 bytes free)
```

Configure an IOS IPS crypto key



```
realm-cisco.pub signature.txt - Notepad
File Edit Format View Help
crypto key pubkey-chain rsa
named-key realm-cisco.pub signature
key-string
30820122 300D0609 2A864886 F70D0101 01050003 82010F00 3082010A 02820101
00C19E93 A8AF124A D6CC7A24 5097A975 206BE3A2 06FBA13F 6F12CB5B 4E441F16
17E630D5 C02AC252 912BE27F 37FDD9C8 11FC7AF7 DCDD81D9 43CDABC3 6007D128
B199ABCB D34ED0F9 085FADC1 359C189E F30AF10A C0EFB624 7E0764BF 3E53053E
5B2146A9 D7A5EDE3 0298AF03 DED7A5B8 9479039D 20F30663 9AC64B93 C0112A35
FE3F0C87 89BCB7BB 994AE74C FA9E481D F65875D6 85EAF974 6D9CC8E3 F0B08B85
50437722 FFBE85B9 5E4189FF CC189CB9 69C46F9C A84DFBA5 7A0AF99E AD768C36
006CF498 079F88F8 A3B3FB1F 9FB7B3CB 5539E1D1 9693CCBB 551F78D2 892356AE
2F56D826 8918EF3C 80CA4F4D 87BFCA3B BFF668E9 689782A5 CF31CB6E B4B094D3
F3020301 0001
quit
exit
exit
```

- ▶ Copy the public key file's contents and paste it into global configuration mode
- ▶ Removing a key:
`no crypto key pubkey-chain rsa`
`no named-key realm-cisco.pub signature`

Enabling IOS IPS - rules

- ▶ Create an IPS rule:

```
R(config)#ip ips name IOSIPS
```

- ▶ Optionally, specify an ACL to select traffic that will be inspected by the IPS:

```
R(config)#ip ips name IOSIPS2 list ACL
```

- ▶ Specify the location of the IPS file:

```
R(config)#ip ips config location flash:ips
```

Enabling IOS IPS - logging

- ▶ SDEE (Security Device Event Exchange) is a protocol running between IPS clients and servers
 - ▶ Relies on HTTP/HTTPS protocols.

```
R(config)#ip http server  
R(config)#ip ips notify sdee  
R(config)#ip ips notify log
```

- ▶ IPS can also notify via syslog (for configured syslog destinations)

Enabling IOS IPS: assign the rule to an interface

- ▶ Applying the IOS **IPS rule** to an interface:

```
R3(config)#interface FastEthernet0/0
```

```
R3(config-if)#ip ips IOSIPS in
```

- ▶ The rule can be applied inbound, as well as outbound, on the same interface:

```
R3(config)#interface Serial0/1/1
```

```
R3(config-if)#ip ips IOSIPS in
```

```
R3(config-if)#ip ips IOSIPS out
```

Loading IOS IPS signature package to the router

- ▶ Load the signature package downloaded in the first step
- ▶ Use FTP or TFTP server

```
copy ftp://<user:password@server>/<package> idconf
```

- ▶ For example:

```
R3#copy ftp://cisco:cisco@10.1.1.1/IOS-S310-CLI.pkg idconf
Loading IOS-S310-CLI.pkg !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
[OK - 7608873/4096 bytes]
```

- ▶ The signature package is compiled after it is loaded on the router
- ▶ Verify the signature package is properly compiled

```
R3#show ip ips signature count
Cisco SDF release version S310.0
[...]
```

Enabling IOS IPS: signature categories

- ▶ Signatures are grouped into hierarchical categories
- ▶ Common categories: **all**, **basic**, **advanced**
- ▶ A category can be:
 - ▶ Retired: not compiled by IOS, unused
 - ▶ Not retired: compiled and used to scan traffic

```
R(config)#ip ips signature-category
R(config-ips-category)#category all
R(config-ips-category-action)#retired true
R(config-ips-category-action)#exit
R(config-ips-category)#category ios_ips basic
R(config-ips-category-action)#retired false
R(config-ips-category-action)#exit
R(config-ips-category)#exit
Do you want to accept these changes? [confirm] y
R(config)#
```

Altering individual signatures

- ▶ Retiring an individual signature with id **6130** and subsignature id **10**:

```
R1(config)# ip ips signature-definition
R1(config-sigdef)# signature 6130 10
R1(config-sigdef-sig)# status
R1(config-sigdef-sig-status)# retired true
R1(config-sigdef-sig-status)# exit
R1(config-sigdef-sig)# exit
R1(config-sigdef)# exit
Do you want to accept these changes? [confirm] y
R1(config)#
```

Altering categories

- ▶ **Unretire** an entire **category**:

```
R1(config)# ip ips signature-category
R1(config-ips-category)# category ios_ips basic
R1(config-ips-category-action)# retired false
R1(config-ips-category-action)# exit
R1(config-ips-category)# exit
Do you want to accept these changes? [confirm] y
R1(config)#
```

Changing a signature's action

```
R1(config)# ip ips signature-definition
R1(config-sigdef)# signature 6130 10
R1(config-sigdef-sig)# engine
R1(config-sigdef-sig-engine)# event-action produce-alert
R1(config-sigdef-sig-engine)# event-action deny-packet-inline
R1(config-sigdef-sig-engine)# event-action reset-tcp-connection
R1(config-sigdef-sig-engine)# exit
R1(config-sigdef-sig)# exit
R1(config-sigdef)# exit
Do you want to accept these changes? [confirm] y
R1(config)#
```

- ▶ The alert states that signature 6130 with subsignature 10 will generate an alert, drop packets and close the TCP connection when triggered

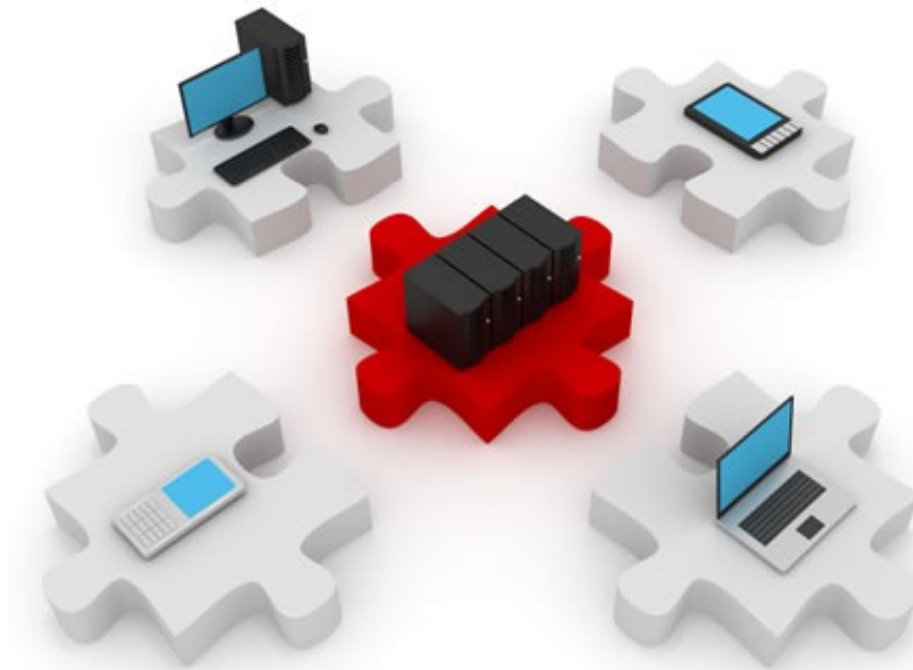
Verifying configuration

- ▶ The `show ip ips` can be used with several other parameters to provide specific IPS information
- ▶ The `show ip ips all` command displays all IPS configuration data
- ▶ The `show ip ips configuration` command displays additional configuration data that is not displayed with the `show running-config` command
- ▶ The `show ip ips interface` command displays interface configuration data. The output from this command shows inbound and outbound rules applied to specific interfaces

“Expecting the world to treat you fairly because you are a good person is a little like expecting a bull not to attack you because you are a vegetarian.”

Dennis

Wholey



Goodbye detected!