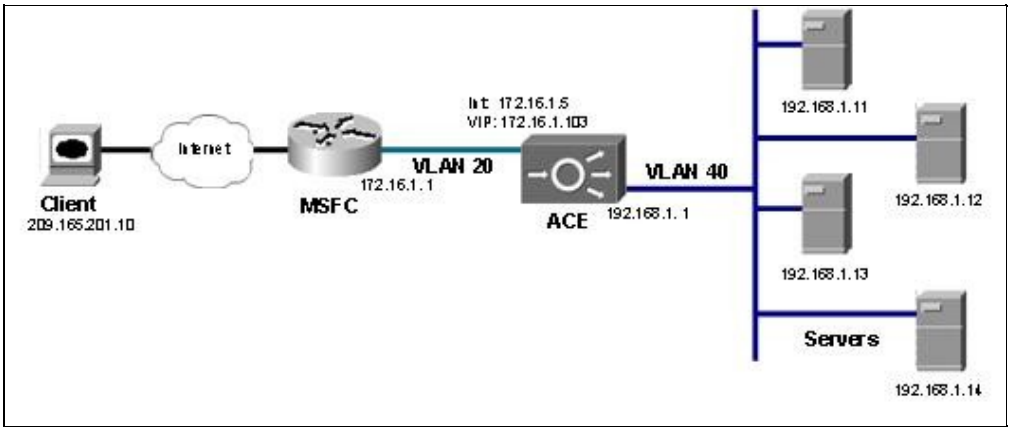
Contents

Goal

Configure basic FTP load balancing where client traffic enters on one network and is directed to FTP servers residing on a second network. All traffic will be sent to a single serverfarm containing five real servers. ACE will be configured to inspect FTP control channel messages, and dynamically open the appropriate data channel ports. Additionally, ACE will be configured with strict FTP inspection, denying any unwanted FTP commands from being used.

Design

Clients will send application requests through the MFSC, which routes them to a virtual IP address (VIP) within ACE. The VIP used in this example resides in an ACE context which is configured with a client VLAN and a server VLAN. Client requests will hit the VIP, ACE will pick the appropriate server, and then destination NAT the client request to send to the server. The server will respond using the interface VLAN of ACE as its default gateway to the client. ACE will then change the source IP to be the VIP and forward the response to the client via the MSFC. When the client and server negotiate the tcp port to be used for the data channel, ACE performs address translation on the embedded IP?s within the FTP protocol. ACE will also open the necessary port for the dynamically assigned data-channel.



Configuration

The ACE needs to be configured via ACLs to allow traffic into the ACE data plane. After the ACL checks are made, a service-policy which is applied to the interface is used to classify traffic destined to the VIP. The VIP is associated to a load balancing action within the multi-match policy. The load balancing action tells ACE how to handle traffic which has hit a VIP. In this example, all FTP connections will be serviced by a serverfarm containing five real servers. The ACE configuration is layered, such that it builds from the real

IPs to finally applying the VIP on an interface. Due to this structure, it is optimal to create the configuration working backwards in terms of how the flow is processed. Thus, to enable server load balancing you need to configure the following objects:

- Enable ACLs to allow data traffic through the ACE device, as it is denied by default.
- Configure the IPs of the servers (define rservers)
- Group the real servers (create a serverfarm)
- Define the virtual IP address (VIP)
- Define how traffic is to be handled as it is received (create a policy-map for load balancing)
- Define how traffic is to be inspected (create a policy-map for inspection)
- Associate A VIP to a handling action (create a multi-match policy-map (aka service-policy))
- Create client and server facing interfaces
- Apply the VIP and ACL permitting client connections to the interface (apply access-group and service policy to interface)

To begin the configuration, create an access list for permitting client connections.

```
ACE-1/routed(config)# access-list everyone extended permit ip any any
ACE-1/routed(config)# access-list everyone extended permit icmp any any
```

Note: While this example shows a permit any any, it is recommend ACLs be used to only permit the traffic you want allow through ACE. In the past SLB devices have used the VIP and port alone to protect servers. Within ACE ACLs are processed first, thus dropping traffic using an ACL requires less resources than dropping traffic once it passes the ACLs and hits the VIP.

ACE needs to know the IP address of the servers available to handle client connections. The rserver command is used to define the ip address of the service. In addition each rserver must be placed inservice for it to be used. The benefit of this design is no matter how many applications or services an rserver hosts, the entire real server can be completely removed from the load balancing rotation by issuing a single ?no inservice? or ?no inservice-standby? command at the rserver level. This is very beneficial for users need to upgrade or patch an rserver, so users no longer have to go to each application and remove each instance of the rserver.

```
ACE-1/routed(config)# rserver lnx1
ACE-1/routed(config-rserver-host)# ip add 192.168.1.11
ACE-1/routed(config-rserver-host)# inservice
ACE-1/routed(config-rserver-host)# rserver lnx2
ACE-1/routed(config-rserver-host)# ip add 192.168.1.12
ACE-1/routed(config-rserver-host)# inservice
ACE-1/routed(config-rserver-host)# rserver lnx3
ACE-1/routed(config-rserver-host)# ip add 192.168.1.13
ACE-1/routed(config-rserver-host)# inservice
ACE-1/routed(config-rserver-host)# rserver lnx4
ACE-1/routed(config-rserver-host)# ip add 192.168.1.14
ACE-1/routed(config-rserver-host)# inservice
ACE-1/routed(config-rserver-host)# rserver lnx5
ACE-1/routed(config-rserver-host)# ip add 192.168.1.15
ACE-1/routed(config-rserver-host)# inservice
ACE-1/routed(config-rserver-host)# exit
ACE-1/routed(config)#
```

Now group the rservers to be used to handle client connections using a serverfarm. Again the rservers must be placed inservice. This allows a single instance of an rserver to be manually removed from rotation.

```
ACE-1/routed(config-cmap)# serverfarm ftpfarm
ACE-1/routed(config-sfarm-host)# rserver lnx1
```

```
ACE-1/routed(config-sfarm-host-rs)# inservice
ACE-1/routed(config-sfarm-host-rs)# rserver lnx2
ACE-1/routed(config-sfarm-host-rs)# inservice
ACE-1/routed(config-sfarm-host-rs)# rserver lnx3
ACE-1/routed(config-sfarm-host-rs)# inservice
ACE-1/routed(config-sfarm-host-rs)# rserver lnx4
ACE-1/routed(config-sfarm-host-rs)# inservice
ACE-1/routed(config-sfarm-host-rs)# rserver lnx5
ACE-1/routed(config-sfarm-host-rs)# inservice
ACE-1/routed(config-sfarm-host-rs)# exit
ACE-1/routed(config-sfarm-host)# exit
ACE-1/routed(config)#
```

Use a class-map to define the VIP where clients will be sending their requests.

```
ACE-1/routed(config)# class-map slb-vip
ACE-1/routed(config-cmap)# match virtual-address 172.16.1.104 tcp eq ftp
ACE-1/routed(config-cmap)# exit
ACE-1/routed(config)#
```

Next define the action to take when a new client request arrives. In this case we have only one possible action. Due to this, we can simply use the built-in ?class-default? class to handle all traffic.

```
ACE-1/routed(config)# policy-map type loadbalance http first-match slb-logic
ACE-1/routed(config-pmap-lb)# class class-default
ACE-1/routed(config-pmap-lb-c)# serverfarm ftpfarm
ACE-1/routed(config-pmap-lb-c)# exit
ACE-1/routed(config-pmap-lb)# exit
ACE-1/routed(config)#
```

Use a class-map to define the FTP protocol commands which should be prevented from reaching the FTP server.

```
ACE-1/routed(config)# class-map type ftp inspect match-any badcommands
ACE-1/routed(config-cmap)# match request-method put
ACE-1/routed(config-cmap)# match request-method dele
ACE-1/routed(config-cmap)# match request-method rmd
ACE-1/routed(config-cmap)# exit
ACE-1/routed(config)#
```

Next define the way ACE should handle the offending FTP commands when they are detected during inspection. In this case, we are denying the offending commands, resulting in the FTP connection being reset.

```
ACE-1/routed(config)# policy-map type inspect ftp first-match check-commands
ACE-1/routed(config-pmap-ftp-ins)# class badcommands
ACE-1/routed(config-pmap-ftp-ins-c)# deny
```

Since the VIPs and load balancing actions are defined independently they must be associated so the ACE knows how traffic destined to a VIP should be handled. The association is made using a multi-match policy-map. Keep in mind, multi-match policy-maps are applied to interfaces as service-policies. For FTP load balancing to work properly, the ?inspect ftp? command must be added to the class. This enables the FTP data channel to be properly negotiated between the client and the server. Adding a ?strict? policy to the inspect command allows the ACE to perform the command inspection defined previously.

```
ACE-1/routed(config)# policy-map multi-match client-vips
ACE-1/routed(config-pmap)# class slb-vip
```

```
ACE-1/routed(config-pmap-c)# loadbalance policy slb-logic
ACE-1/routed(config-pmap-c)# loadbalance vip inservice
ACE-1/routed(config-pmap-c)# inspect ftp strict policy check-commands
ACE-1/routed(config-pmap-c)# exit
ACE-1/routed(config-pmap)# exit
ACE-1/routed(config)#
```

At this point the interface vlans can be created to interconnect ACE to the client side of the network and to the servers

```
ACE-1/routed(config)# interface vlan 20
ACE-1/routed(config-if)# description ?Client Side?
ACE-1/routed(config-if)# ip address 172.16.1.5 255.255.255.0
ACE-1/routed(config-if)# no shutdown

ACE-1/routed(config-if)# interface vlan 40
ACE-1/routed(config-if)# description ?Default gateway of real servers?
ACE-1/routed(config-if)# ip address 192.168.1.1 255.255.255.0
ACE-1/routed(config-if)# no shutdown
ACE-1/routed(config-if)# exit
```

The last step is to apply the ACL and service policy (policy-map multi-match) to the client side interface. Both the access-group and service policy are applied on the input side of the interface.

```
ACE-1/routed(config)# interface vlan 20
ACE-1/routed(config-if)# access-group input everyone
ACE-1/routed(config-if)# service-policy input client-vips
```

Note: There is no need to add an access group to the server side, as the ACE automatically creates pin holes to allow server response traffic to pass back to the client.

Related show commands

This section provides information you can use to confirm your configuration is working properly.

Certain show commands are supported by the Output Interpreter Tool (registered customers only), which allows you to view an analysis of show command output.

```
ACE-1/routed #show arp
ACE-1/routed #show acl
ACE-1/routed #show show service-policy clientvips
ACE-1/routed #show show service-policy clientvips detail
ACE-1/routed #show serverfarm
ACE-1/routed #show rserver
ACE-1/routed #show stats
```

Comments

Once configured verify the ACE has an ARP response for each rserver and the default route to the client. Check the ACL hits to ensure client connections are being accepted. Check the service policy output to see the client connection hits, and verify the server is responding with response packets. Checking the service policy will also show you which classes are being matched based on the URLs being requested. The show command for serverfarm and rserver can be used to display the exact rserver handling the connection and the amount of work the entire serverfarm has handled. The show stats command provides a higher level of monitoring ACE load balancing, inspection, probes, and other important metrics.

show running-config

```
ACE-1/routed# sho run
Generating configuration....

access-list everyone line 8 extended permit ip any any
access-list everyone line 16 extended permit icmp any any

rserver host lnx1
  ip address 192.168.1.11
  inservice
rserver host lnx2
  ip address 192.168.1.12
  inservice
rserver host lnx3
  ip address 192.168.1.13
  inservice
rserver host lnx4
  ip address 192.168.1.14
  inservice
rserver host lnx5
  ip address 192.168.1.15
  inservice

serverfarm host ftpfarm
  rserver lnx1
    inservice
  rserver lnx2
    inservice
  rserver lnx3
    inservice
  rserver lnx4
    inservice
  rserver lnx5
    inservice

class-map type ftp inspect match-any badcommands
  2 match request-method put
  3 match request-method dele
  4 match request-method rmd

class-map match-all slb-vip
  2 match virtual-address 172.16.1.104 tcp eq ftp

policy-map type management first-match remote-access
  class remote-access
    permit



policy-map type loadbalance first-match slb-logic
  class class-default
    serverfarm ftpfarm

policy-map type inspect ftp first-match check-commands
  class badcommands
    deny

policy-map multi-match client-vips
  class slb-vip
    loadbalance vip inservice
    loadbalance policy slb-logic
    inspect ftp strict policy check-commands
```

```
interface vlan 20
  description "Client Side"
  ip address 172.16.1.5 255.255.255.0
  access-group input everyone
  service-policy input client-vips
  no shutdown

interface vlan 40
  description "Default gateway of real servers"
  ip address 192.168.1.1 255.255.255.0
  service-policy input remote-access
  no shutdown

ip route 0.0.0.0 0.0.0.0 10.90.14.254
```

Related Information

Technical Support & Documentation - Cisco Systems