\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Policy Based Routing

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Author*:

Gabriel Rosas



Purpose

Imagine a scenario where you decide to host a server on the internet. Perhaps a website, or maybe a file server. Potentially something else entirely. Without adding any traffic *rules* to your router, anyone from anywhere using any protocol would have access to the server with no restraints. *Policy-based Routing* (PBR) is a routing technique that forwards or denies traffic based on an implementation of rules and filters, known as *policies*. Suddenly, those unused ports on the network can be blocked, and only the necessary ones left open.

Background Information

Introduction

Two servers would run the same two protocols on the same network. However, only one type of traffic was to be permitted to each server from an external network. The goal of this project was to use *policy-based routing* to permit a certain protocol to each server. In this case, we chose to permit HTTP to one server and SSH to the other, with both servers running HTTP and SSH.

Access Lists

Routers and firewalls can use *Access-lists* (ACL) to permit or deny incoming or outgoing traffic. They function as a set of rules, read from the first to last rule. If a packet matches a rule before it is denied, then it “passes” the ACL and continues its path. If a packet does not match any rule in the ACL, then it is denied implicitly, a feature known as the *implicit deny*. There are two main types of ACL: standard and extended.

* Standard
  + Can only match source IP.
  + Cannot match port.
  + Doesn’t distinguish between TCP / UDP / other traffic.
  + Standard ACL identifiers: [*1-99*] & [*1300-1999*] or *named*.
  + *“ip access-list standard 10”  
    “permit 10.0.0.0 255.255.255.0”.*
* Extended
  + Can match traffic based on source IP, destination IP, source port, destination port.
  + Can distinguish between TCP / UDP / other traffic.
  + Extended ACL identifiers: [*100-199*] & [*2000-2699*] or *named*.
  + *“ip access-list extended PERMIT\_HTTP”  
    “permit tcp any host 10.0.0.1 eq www”.*

For example, let’s suppose I want to discard all ping (*icmp*) packets entering my network. A flood of ping requests can overwhelm a network device, denying users service. I could configure the following on a Cisco router:

*ip access-list extended DENY\_ICMP  
deny icmp any any  
permit ip any any*

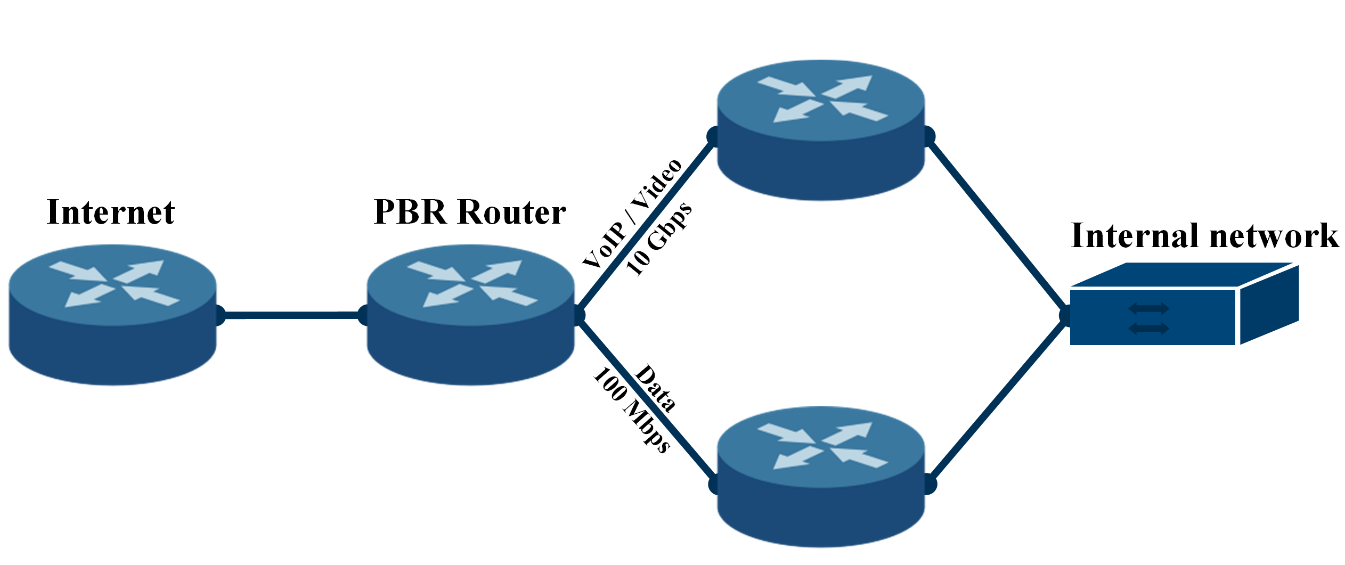
If applied on the correct interface, this access list will deny *icmp* packets entering the network but permit all other types of traffic. But what if an administrator from outside wants to ping our internal network? We would edit the ACL to permit the admin *before* we deny all icmp traffic. Since access lists are read from top to bottom, a permit line before the deny would save the admin’s traffic from being dropped. The admin’s ip address is *232.45.6.23*.

*ip access-list extended DENY\_ICMP  
permit icmp host 232.45.6.23 any  
deny icmp any any  
permit ip any any*

Once the desired access list is created and written, we must apply it on an interface and specify whether we want the traffic coming in or going out to be examined. If I wanted to apply the icmp ACL on all packets *entering* the *GigabitEthernet0/0/0* interface of my router, then the command would be: *ip access-group DENY\_ICMP in* (in interface configuration mode).

Policy Based Routing

Now we know a little about access lists; so, how do they differ from *policy-based routing*? Policy-based routing uses *route maps* – containing access lists –to manipulate traffic like generic access lists. However, route maps provide the additional feature of editing and adding parts to traffic after it has passed or failed the access lists. Let’s look at an example.



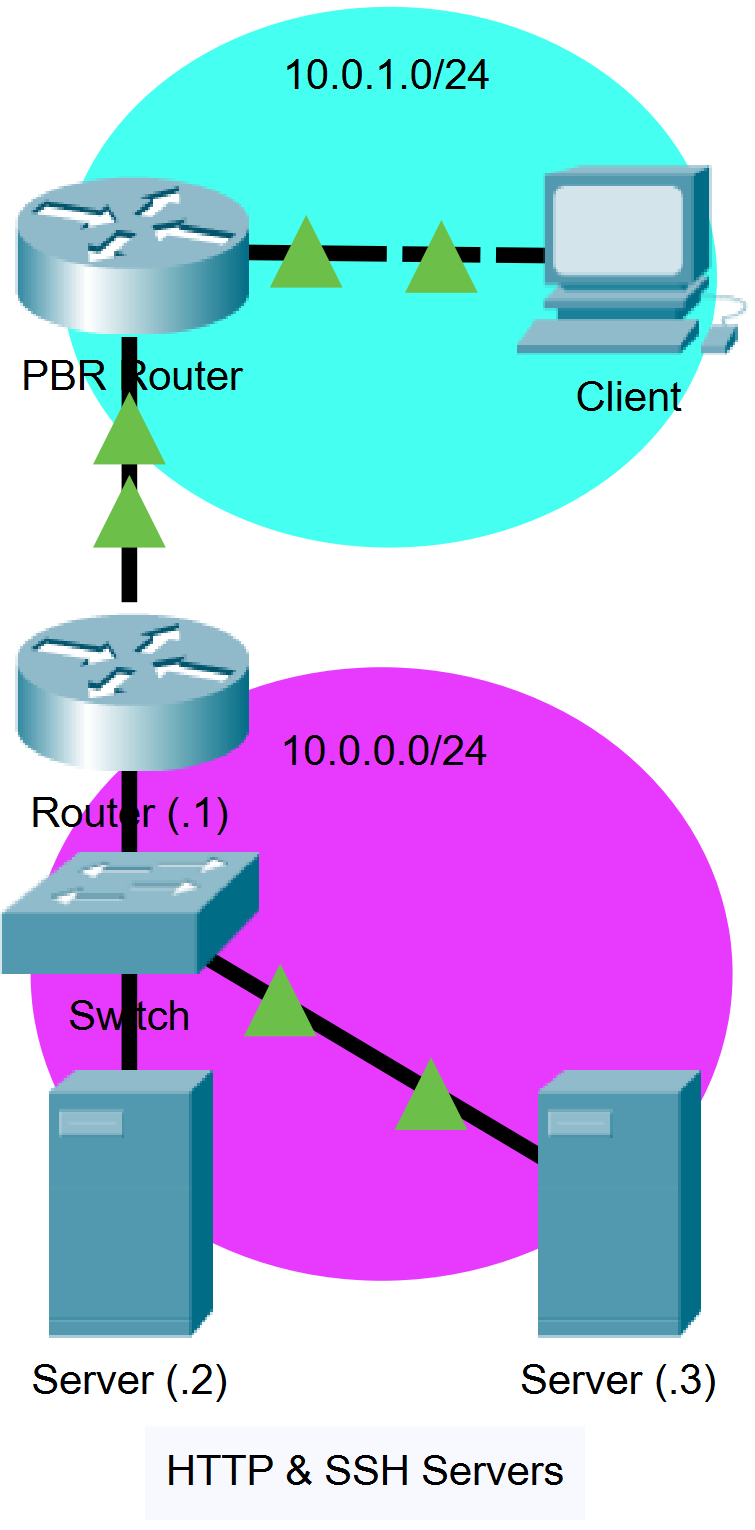
In this case, we have a router (*PBR Router*) that forwards traffic from the internet towards an internal network. The PBR Router has two choices of which to send the traffic: through a *10-gigabit* ethernet connection or through a *100-megabit* ethernet connection. Both routes eventually lead to the internal network. The network administrator wants to divide traffic so that VoIP and Video traffic is sent down the 10 *Gbps* link. With *policy-based routing*, we can *set* a *next-hop ip* based on the type of the source traffic (determined by an access list). The extra step of setting a next hop ip would not be possible with just access lists.

Linux

Like Windows and MacOS, Linux is an operating system which can run applications, but particularly excels at server-based services since it is much more lightweight than other software. In computing, something “lightweight” refers to software designed to have a small memory footprint (RAM), low CPU usage, and low overall usage of system resources. Perfect for something like a server, but less friendly to the average user. Users will typically navigate Linux through the command-line with less focus on graphical applications.

Since Linux is an open-source OS, where anyone can take the base code and manufacture it to their liking, there are many *distributions* designed for specific purposes. For example, in previous projects I worked with *Kali* Linux, a distribution focused on network penetration testing; and *Raspbian*, the official operating system of the Raspberry Pi. In this project, I hopped back on Raspbian to host an Apache Web Server as an end device.

Network Diagram



Lab Summary

I began by making a topology on packet tracer. Originally, this topology only consisted of one router. But before we could implement policies on the router, we needed to configure two servers running Apache web services and SSH.

Setting up Apache on Raspbian

Since this project required multiple servers, I prepared one on my Raspberry Pi and my partner, Harsha Bhat, set up the other on an Ubuntu virtual machine. Installing Apache was fairly straightforward on both machines: *sudo apt-get install apache2*.

Lab Commands

|  |  |
| --- | --- |
| **Command** | A statement necessary for a configuration to work, denoted in bold |
| **[*Argument*]** | An argument necessary for a command to function, denoted in bold italics. |
| *Optional-Statement*  *<Optional Argument>* | An optional argument or statement, not necessary for a command to function, denoted in italics |

Text

Configurations

PBR Router

PBR-Router#show running-config

service timestamps debug datetime msec

service timestamps log datetime msec

no platform punt-keepalive disable-kernel-core

hostname PBR-Router

boot-start-marker

boot-end-marker

vrf definition Mgmt-intf

address-family ipv4

exit-address-family

address-family ipv6

exit-address-family

no aaa new-model

subscriber templating

multilink bundle-name authenticated

license udi pid ISR4321/K9 sn FDO214421BU

spanning-tree extend system-id

redundancy

mode none

vlan internal allocation policy ascending

interface GigabitEthernet0/0/0

ip address 192.168.1.1 255.255.255.252

negotiation auto

interface GigabitEthernet0/0/1

ip address 10.0.1.1 255.255.255.0

ip policy route-map http-ssh-policy

negotiation auto

interface Serial0/1/0

no ip address

shutdown

interface Serial0/1/1

no ip address

shutdown

interface Service-Engine0/2/0

no ip address

shutdown

interface GigabitEthernet0

vrf forwarding Mgmt-intf

no ip address

shutdown

negotiation auto

interface Vlan1

no ip address

shutdown

ip forward-protocol nd

ip http server

ip http authentication local

ip http secure-server

ip tftp source-interface GigabitEthernet0

ip route 10.0.0.0 255.255.255.0 GigabitEthernet0/0/0 192.168.1.2

access-list 100 permit tcp any host 10.0.0.2 eq www

access-list 100 permit tcp any host 10.0.0.3 eq 22

route-map http-ssh-policy permit 10

match ip address 100

set ip next-hop 192.168.1.2

route-map http-ssh-policy permit 20

set interface Null0

control-plane

line con 0

stopbits 1

line aux 0

stopbits 1

line vty 0 4

login

end

Server Network Router

Router#show running-config

service timestamps debug datetime msec

service timestamps log datetime msec

platform qfp utilization monitor load 80

platform punt-keepalive disable-kernel-core

hostname Router

boot-start-marker

boot-end-marker

vrf definition Mgmt-intf

address-family ipv4

exit-address-family

address-family ipv6

exit-address-family

no aaa new-model

login on-success log

subscriber templating

multilink bundle-name authenticated

no license smart enable

diagnostic bootup level minimal

spanning-tree extend system-id

redundancy

mode none

interface GigabitEthernet0/0/0

ip address 10.0.0.1 255.255.255.0

negotiation auto

interface GigabitEthernet0/0/1

ip address 192.168.1.2 255.255.255.252

negotiation auto

interface GigabitEthernet0

vrf forwarding Mgmt-intf

no ip address

shutdown

negotiation auto

ip forward-protocol nd

ip http server

ip http authentication local

ip http secure-server

ip tftp source-interface GigabitEthernet0

ip route 10.0.1.0 255.255.255.0 GigabitEthernet0/0/1 192.168.1.1

control-plane

line con 0

transport input none

stopbits 1

line aux 0

stopbits 1

line vty 0 4

login

end

Problems

Text

Conclusion

Text