

investigatinG bgp security in cisco virl

Roey Maor 205798440; Michal Cohen 316403070



January 26, 2017

LCCN lab

236340

Table of Contents

[Introduction 3](#_Toc473401339)

[Background 3](#_Toc473401340)

[Overview 3](#_Toc473401341)

[References 4](#_Toc473401342)

[Terminology and Acronyms 4](#_Toc473401343)

[Project General Architecture 5](#_Toc473401344)

[There are three types of ASs: 6](#_Toc473401345)

[System description 6](#_Toc473401346)

[Interfaces used by the project 6](#_Toc473401347)

[BGP attack guide for VIRL cisco 8](#_Toc473401348)

[Components in use 8](#_Toc473401349)

[How Tos: 8](#_Toc473401350)

[Study the topology 10](#_Toc473401351)

[Running the simulation 13](#_Toc473401352)

[Perform the attack 15](#_Toc473401353)

[Check if the attack succeeded 17](#_Toc473401354)

[Investigating different topologies 21](#_Toc473401355)

[Victim and attacker are in Multi Home Ass 21](#_Toc473401356)

[Both servers are closer to the attacker 21](#_Toc473401357)

[Both servers are closer to the victim 24](#_Toc473401358)

[Server-2 is closer to the attacker and server-3 is closer to the victim 26](#_Toc473401359)

[Victim is in Transit AS 30](#_Toc473401360)

[Attacker is in Transit AS 33](#_Toc473401361)

[Attacker has longer distance, but more specified mask 36](#_Toc473401362)

[Same prefix attack: 36](#_Toc473401363)

[IP forwarding 39](#_Toc473401364)

[BGP attack with impersonation 39](#_Toc473401365)

[Sanity check: a Ping between server-2 to server-1 works: 40](#_Toc473401366)

[Reminder: perform BGP hijacking 40](#_Toc473401367)

[Perform IP forwarding 41](#_Toc473401368)

[**Delivering fake news** 43](#_Toc473401369)

[Creating a custom server image 45](#_Toc473401370)

[How we did it 45](#_Toc473401371)

[MITM attack: 55](#_Toc473401372)

[Summering example 59](#_Toc473401373)

[Step 1 – sanity check 59](#_Toc473401374)

[Step 2 – Black hole attack 60](#_Toc473401375)

[Step 3 – Non-interfering MITM 61](#_Toc473401376)

[Step 4 – Interfering MITM 61](#_Toc473401377)

[Conclusions 63](#_Toc473401378)

# Introduction

## Background

The internet is composed of tens of thousands of Autonomous systems (AS). An AS is a large group of connected machines, usually representing an organization or a geographic area. Each AS decides on its intra-routing policy on its own independently from another ASs. These routing protocols aren’t scalable and differ between the different ASs. Unlike the intra-routing protocols, there is only one de-facto routing protocol between ASs and it is the BGP. The BGP algorithm purpose is to control how packets are to be routed between ASs. It is important not only from a technical aspect, but also from a business point of view.

It was written in the days where security wasn't considered, therefore there are many points of weakness to this protocol. For example, by spoofing fake AS-PATH, one can make packages redirect from their legitimate path through It’s AS.

The BGP algorithm is a path vector algorithm which has no metric function, and is affected by many parameters. Some of the parameters being considered are:

* AS-Path: The ASs identity in the path to a subnet
* The policy of the AS
* The resolution of the subnet prefix
* BGP origin
* And more

## Overview

The main idea of the project is to experiment with the relatively new technology from cisco, which is the VIRL simulation software. To use the latter to simulate BGP attacks, study these attacks and reach general conclusions regarding them such as:

* How easy is the execution of the attack from the attackers stand point?
  + In what manner did the ISP needs to cooperate with the attacker?
  + Does the attacker need to deal with race conditions?
  + Does the attacker must use special hardware?
  + How much code does the attack consume?
* What is the radius of influence (how many ASs can be infected)?
* How long does it take the attack to propagate?
* In which situations, such an attack is impossible?
* In which scenarios, such an attack is relatively easy?
* How easy is it to detect BGP attack?
* Is it easy to track the origin of the attack (Is it even possible)?
* What measures could AS manager take to prevent such an attack? Make it more unlikely?

The attack we will simulate will be "Black hole" attack, which means the packets are hijacked to the AS of the attacker, but are not re-transmitted to the original target. Such an attack cannot be held for long because many users notice they cannot connect to a certain part of the internet, and therefore investigations will start quickly and the attack will be stopped. A more sophisticated attack is "MITM" attack, in which the packets are retransmitted to the original target. This behavior is much more difficult to achieve because the internet is practically assigned to transmit these packets to the attacker and not to the original target.

An attempt to perform such an attack must make the internet look "asymmetric" in a BGP manner: at least one path to the target must stay legitimate to send the packets to the target and not receiving them back. We will try to simulate such an attack on a small topology in VIRL.

## References

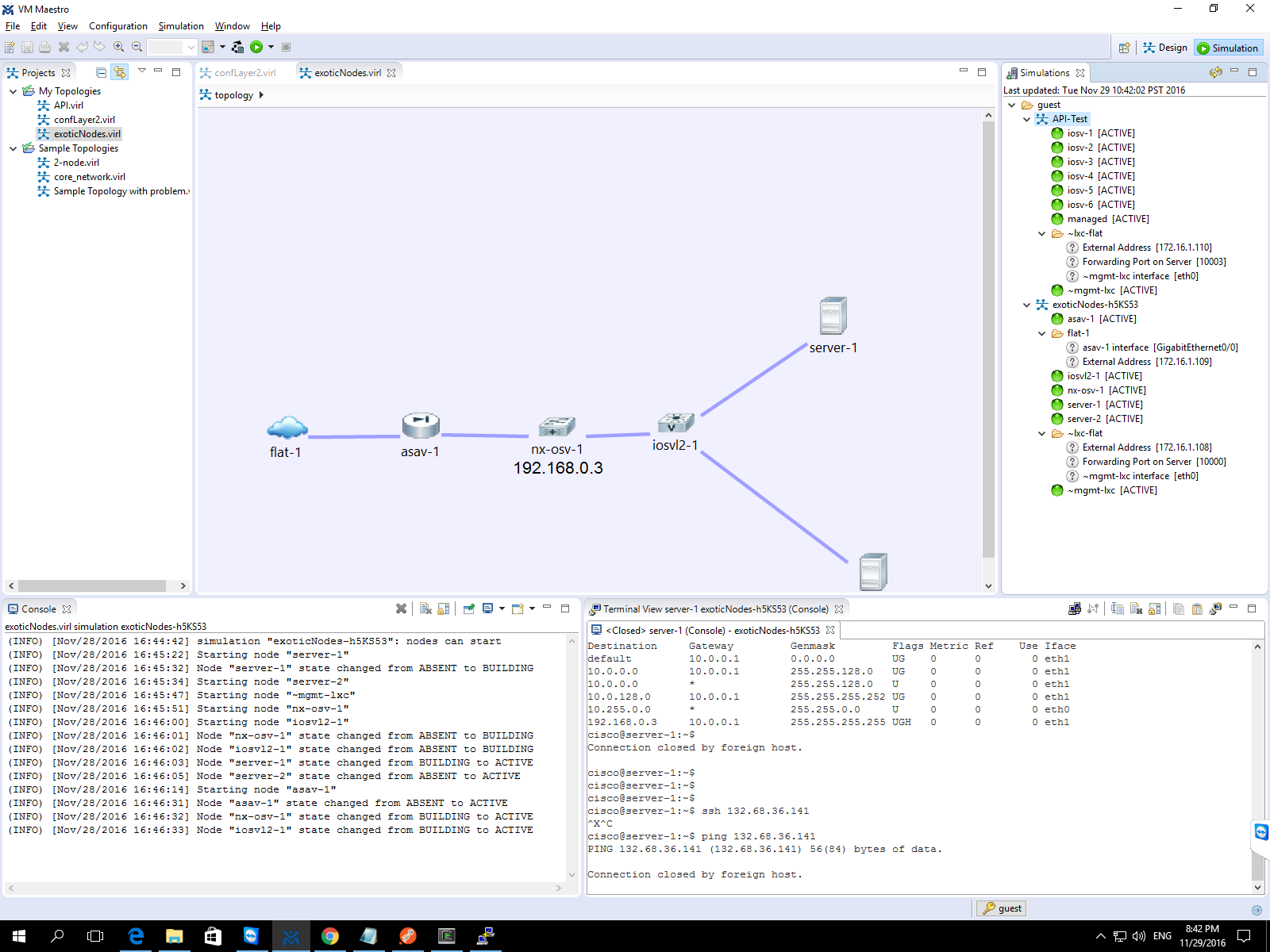
* The tutorial of cisco to VIRL: <http://virl-dev-innovate.cisco.com/vsb.php>
* Previous project using CREATE: <https://www.isi.deterlab.net/file.php?file=/share/shared/BGPhijacking>
* Example of a live BGP attack from the DEF CON conference: <https://www.youtube.com/watch?v=oWdjsfsS_Do>
* "Internet networking" webcourse: <http://webcourse.cs.technion.ac.il/236341/Winter2016-2017/>
* The YouTube channel of cisco VIRL: https://www.youtube.com/channel/UC41WuzXlJCGY5qLsuZ8aHkQ

## Terminology and Acronyms

|  |  |
| --- | --- |
| Term | Meaning |
| BGP | Border gateway protocol |
| AS | Autonomous system |
| VIRL | Virtual internet routing lab |
| VM-Maestro | The client side of the VIRL software |
| Routem | VIRL control plane traffic generator |

# Project General Architecture

The project will be held in a server in the networking lab at the Technicon. That server runs the VIRL server side software that its main purpose is to execute the simulations that are designed in the VIRL client side (VM-Maestro).



### There are three types of ASs:

* Stub AS: an AS connected only to one other AS that provides the first with internet access.
* Multihomed AS: an AS connected as a client to several AS that provides it with internet access. The decision which AS to use in order to rout packets depends on the policy of the AS (for example, A company with critical services might want a backup AS just in case the primal encountered a complete failure.
* Transit AS: a provider of internet access to other AS. The transit AS differ by their size and geographical location, factors that affect the relationships between the ASs.

We will try to perform the attack from different ASs and different policies to see if there are any differences between them.

## System description

We will configure several networks of routers and see how the different parameters affects the efficiency of the attack.

The system will consist different ASs which will be connected through routers.

Each AS will contain some of the components described:

* Routers (IOSv) that will route traffic inside the AS between subnets and between ASs.
* Routem (lxc-routem) control plane traffic generator.
* Servers, using as end-point units

We will generate attacks using the Routem support of BGP protocol, and will see the effect of the attack by the change in various ways

## Interfaces used by the project

There are several RESTful API that the VIRL environment exposes that we will use in our project:

* AutoNetKit: Gives the initial configurations to the components in the simulation.
* UWM: User Workspace Manager, provides a system management dashboard, and documentation
* Routem: control-plane traffic generator: can create custom BGP packets.



User guide to perform BGP attack in VIRL

Example topology taken from UWM

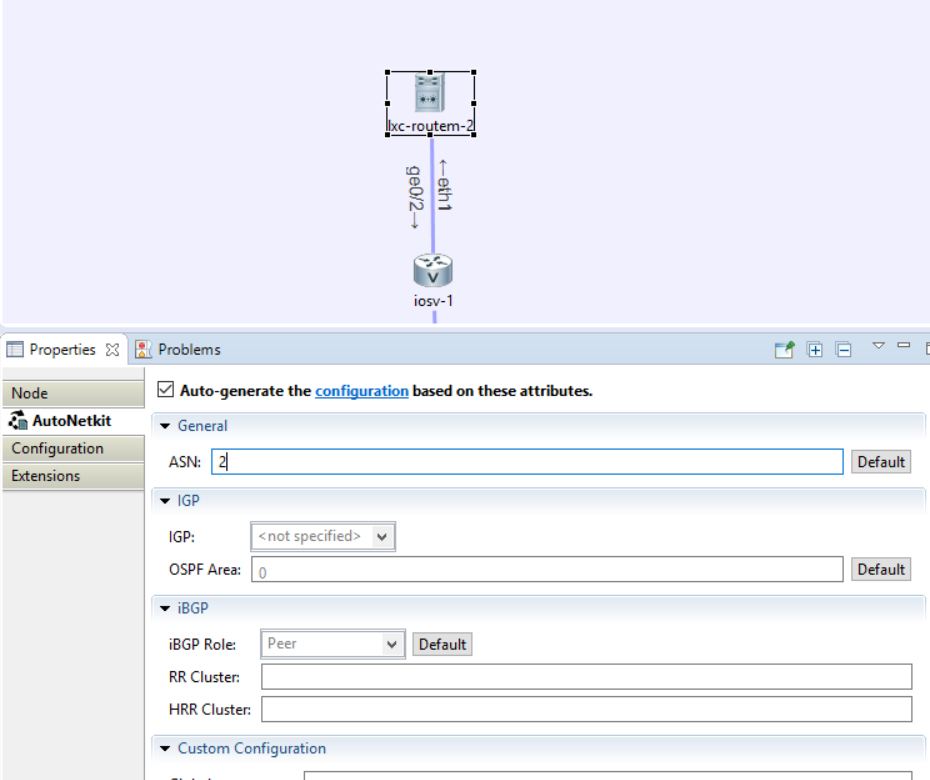


# BGP attack guide for VIRL cisco

## Components in use

* IOSv: CISCO router
* lxc routem: control plane traffic generator
* IOSv-l2: simple switch

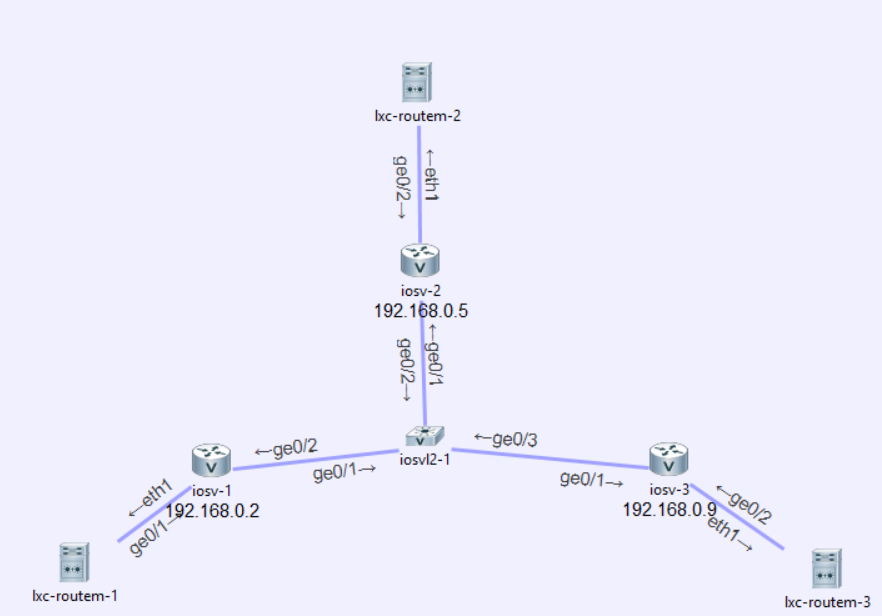
## How Tos:

* How to configure AS: Go to AutoNetKit and change ASN to a number representing your as:  
  
* How to sniff on an interface: Right click on the interface packet captures create new. The offline choice is usually good enough. You can filter here or later in Wireshark, Press ok. Later, you can right click again on the interface, then on the name of the pcap file (the date and time) and then download. Open the pcap file in Wireshark to see the sniffing.

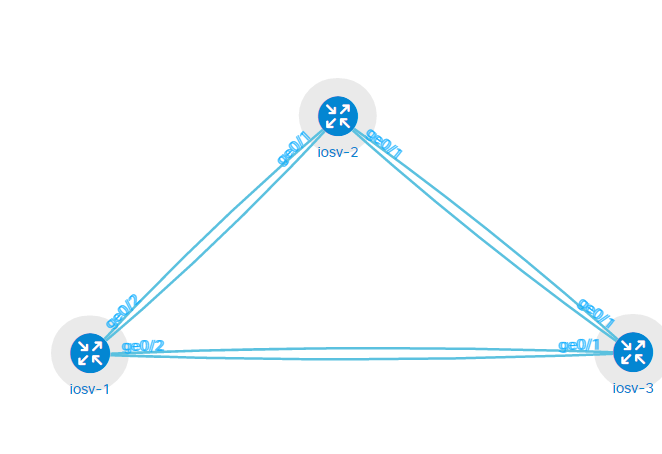
Remember these several crucial constrains:

* Each Routem must be connected to a router.
* The Error message “there is no image” in the simulation means one of the components you chose isn't supported by your server. Contact the VIRL server admin or choose another component.

create the following topology:



Configure routem-i iosv-i to be in the i-AS. Once built, the topology should look like that in an eBGP view:



And like that in an AS view: 

Study the topology

Go to each router and check its' gigabit interfaces and BGP configuration. For example, iosv-1 has the following configuration:

interface GigabitEthernet0/1 *// The interface seeing the routem*

description to lxc-routem-1

ip address 10.0.0.2 255.255.0.0 *// This is the IP of the router as seen by the routem*

ip ospf cost 1

duplex full

speed auto

no shutdown

!

interface GigabitEthernet0/2

description to iosvl2-1

ip address 10.1.0.1 255.255.128.0 *// This is the IP as seen by the other routers*

duplex full

speed auto

no shutdown

!!!

router ospf 1

network 192.168.0.2 0.0.0.0 area 0

log-adjacency-changes

passive-interface Loopback0

network 10.0.0.0 0.0.255.255 area 0

!!

router bgp 1 // the bgp configuration, notice the AS is 1.

bgp router-id 192.168.0.2 *// This is how the router advertises itself in the BGP protocol*

no synchronization

! ibgp // I have no ibgp friends in my network

!

! ebgp

!

neighbor 10.1.0.2 remote-as 2 *// These are the routers’ eBGP neighbors*

neighbor 10.1.0.2 description eBGP to iosv-2

!

neighbor 10.1.0.3 remote-as 3

neighbor 10.1.0.3 description eBGP to iosv-3

!!!

address-family ipv4

network 192.168.0.2 mask 255.255.255.255

network 10.0.0.0 mask 255.255.0.0 *// These are the advertised networks of the router*

network 10.1.0.0 mask 255.255.128.0

neighbor 10.1.0.2 activate *// With no activation, there is no communication*

neighbor 10.1.0.2 send-community

neighbor 10.1.0.2 next-hop-self

neighbor 10.1.0.3 activate

neighbor 10.1.0.3 send-community

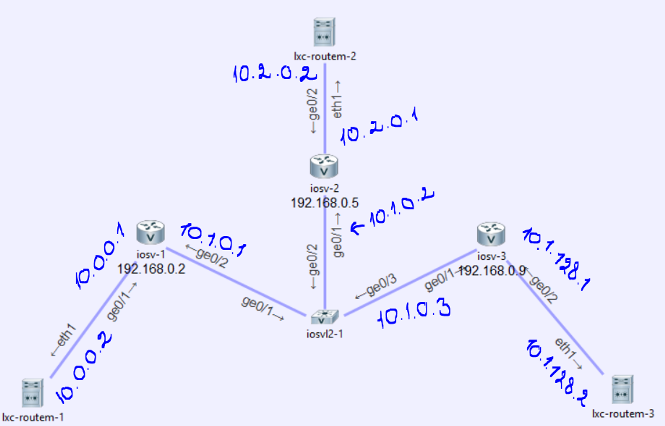
neighbor 10.1.0.3 next-hop-self

exit-address-family

!

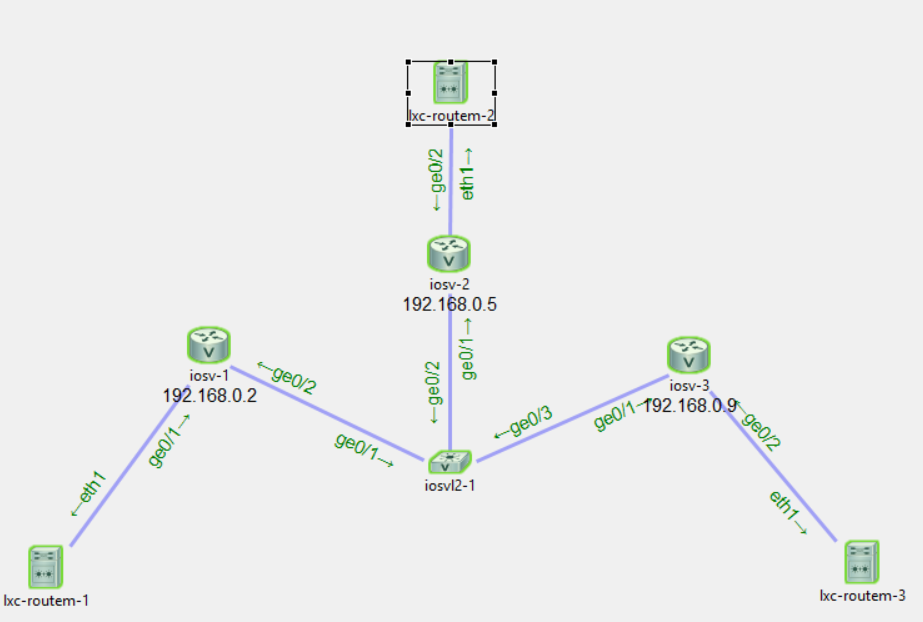
exit

Check the rest of the configuration and get (in our case):



Now you can run the simulation (It might take a few minutes)

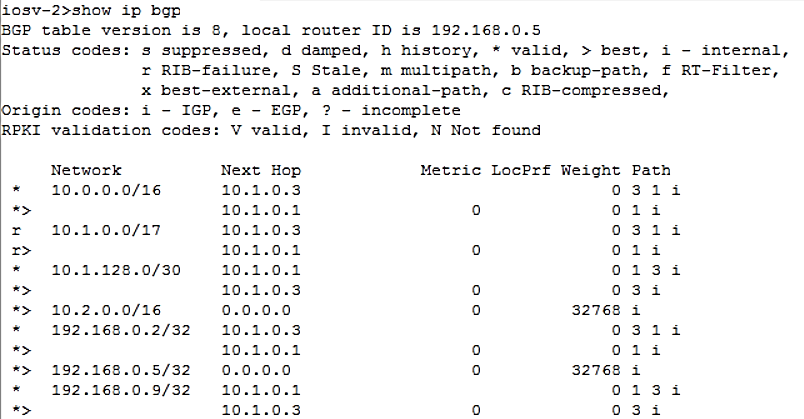
## Running the simulation



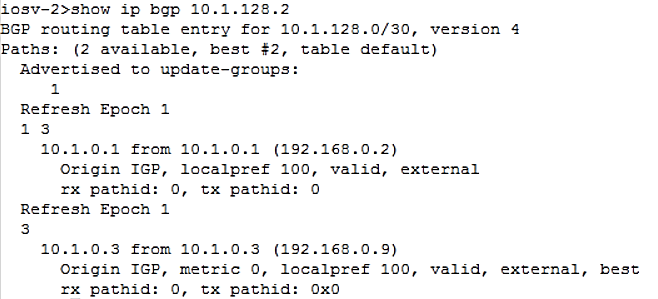
Wait until all the component are green. If a component isn't green in this stage, try to replace it by a new instance or reconfigure the topology.

In this example, routem-1 steels the identity of routem-3, so that iosv-2 thinks it needs to send its packets destined to routem-3 to iosv-1.

For start, we would like to show which networks iosv-2 sees. Open telnet communication with it and type: “show ip bgp” (right click the component and choose Telnet to its Monitor port.



We can see the prefix 10.1.128.0 (which contains our desired 10.1.128.2) has the AS path starting by 10.1.0.3 (This is the best rout, assigned by a ‘>’)



Now we would like to advertise 10.1.128.2 is in routem-1 address space. Beforehand we will do a small test with a non-existing network, so we will be sure there are no competition factors to our trial.

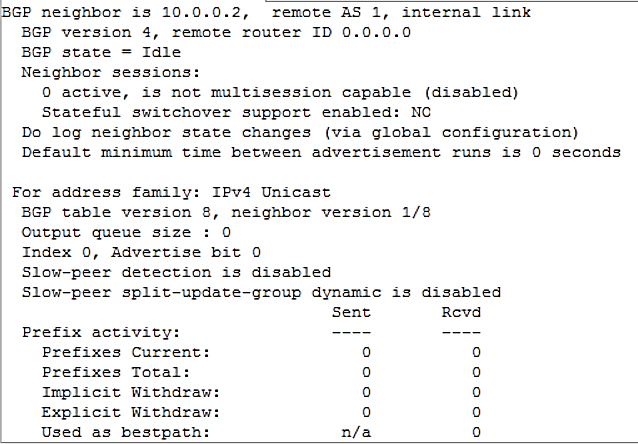
We first need to inform the router iosv-1 that he has a new BGP neighbor. Open SSH connection to iosv1 (via right click), type “configure terminal” and then “router BGP 1” (If you haven't defined otherwise, the username and password is cisco).

**neighbor 10.0.0.2 remote-as 1** *// this defines new BGP neighbor to this router*

**neighbor 10.0.0.2 activate** *// configure this neighbor as active neighbor because by default every bgp component starts “gracefully”.*

## Perform the attack

Now routem-1 is a BGP neighbor of iosv-1. Open management telnet with iosv-1 and type “show ip bgp neighbor 10.0.0.2”

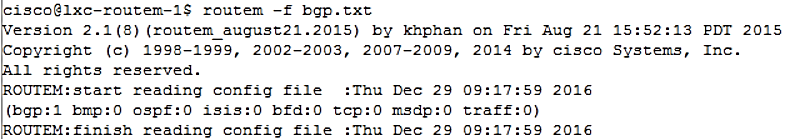


We can see 10.0.0.2 is a configured neighbor, but no active sessions or advertised networks are shown.

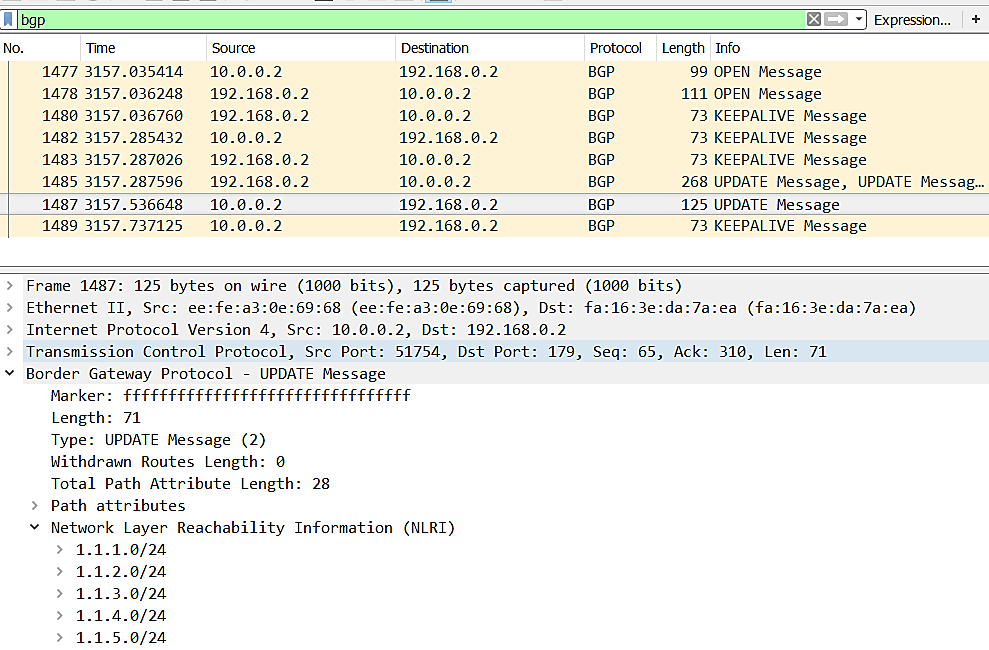
Now we need to configure the other side of the connection. Go to routem-1 and open SSH connection with it. Create new file called bgp.txt and insert the following content (you can do it via “nano” or “vi):

**router bgp 1**  
**neighbor 192.168.0.2 remote-as 1** *// configure iosv-1 as my neighbor*  
**neighbor 192.168.0.2 update-source 10.0.0.2** /*/ tell iosv-1 I am a source of updates*  
**network 1 1.1.1.0/24 5** *// advertise the following networks as my own***metric 1 5**  
**sendall**

Save, open a sniffing on the interface of the Routem, and then run “routem -f bgp.txt”, it is a blocking call.



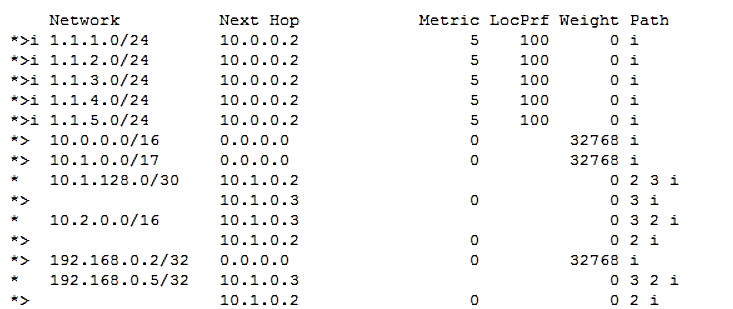
Notice you have 1 BGP session.  
Download the pcap file you’ve sniffed:



We see that after OPEN messages and several KEEPALIVE, the Routem sent update message to iosv-1 (the IP is 192.168.0.2 because this is the IP iosv-1 advertised as his BGP IP). We can see in the update message that the networks 1.1.i.0 for i from 1 to 5 have been advertised.

## Check if the attack succeeded

Go to the telnet connection of iosv-1 and type “show ip bgp”



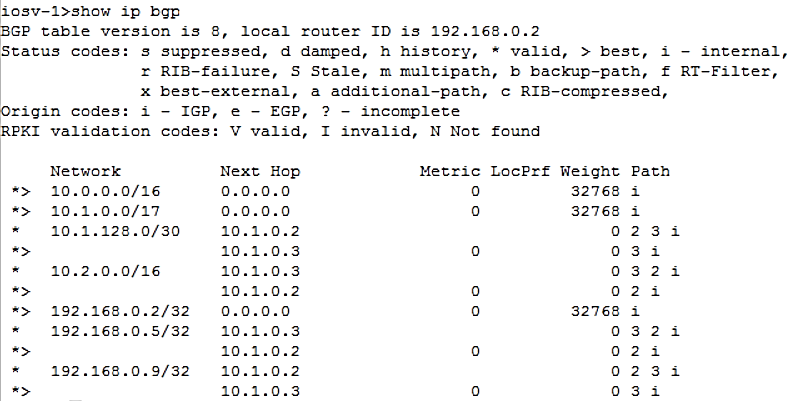
We now see that these networks are going through 10.0.0.2.   
Go to iosv-2 and open telnet communication with it. Type again “show ip bgp”



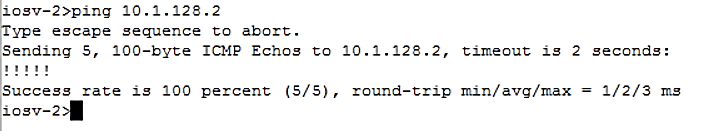
You can see that the preferred path to 1.1.1.0 is going through 10.1.0.1, A question worth asking is why is there even a possibility to go through 10.1.0.3? It is because iosv-3 also got the network advertisement, and now it's also advertising that there is a path from it to these networks. The chosen path is through 10.1.0.1 because the as path of it is of size 1 which is smaller than 2.  
This an example of BGP preference that is based on shorter AS-PATH.

Now we can turn to the original attack that relays on a greater resolution, and hijacks 1.128.0.2 IP.

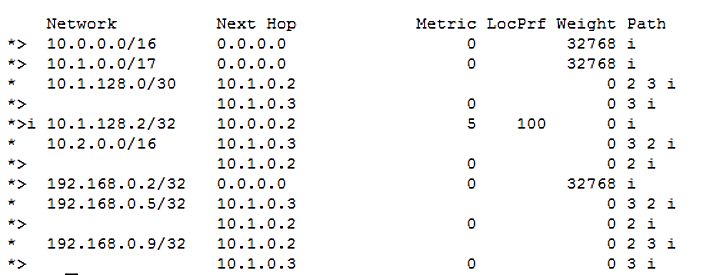
We can see that iosv-1 sees advertisement of 10.1.128.2 with mask of 30.



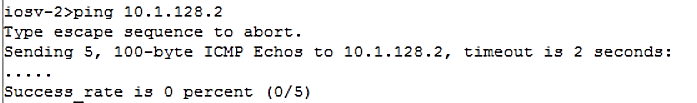
Preform a ping to that server:



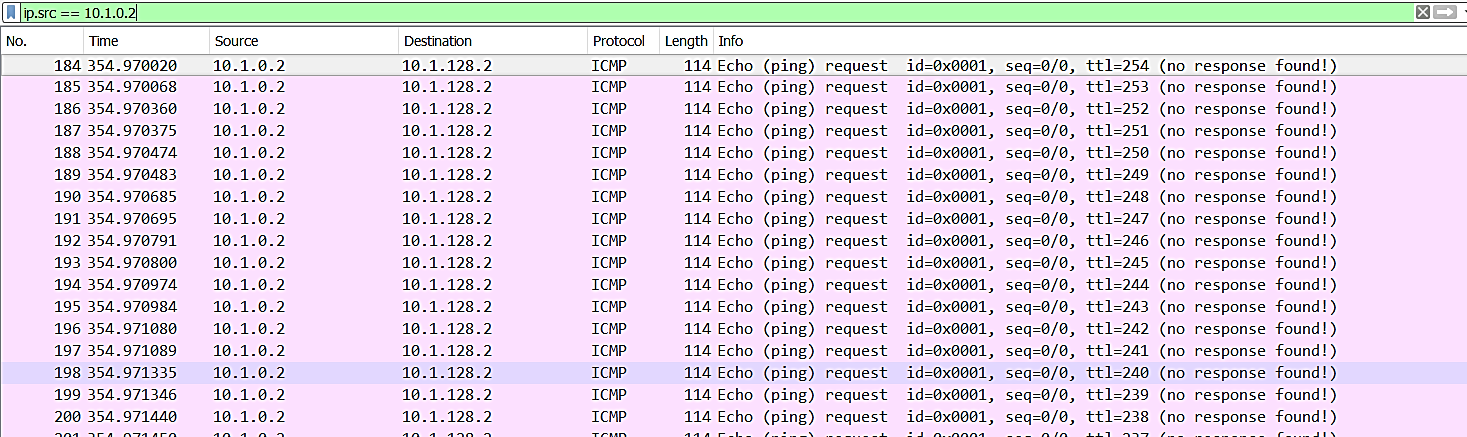
This is a ping to routem-3 from iosv-2, and it is working.  
Do the same procedure as previously, but advertise “network 1 10.1.128.2/**32** 5” instead.



Try to ping again, type in iosv-2 telnet terminal “ping 10.1.128.2”



Now the ping isn’t working. That is because the communication goes through routem-1. Download again the sniffed pcap:



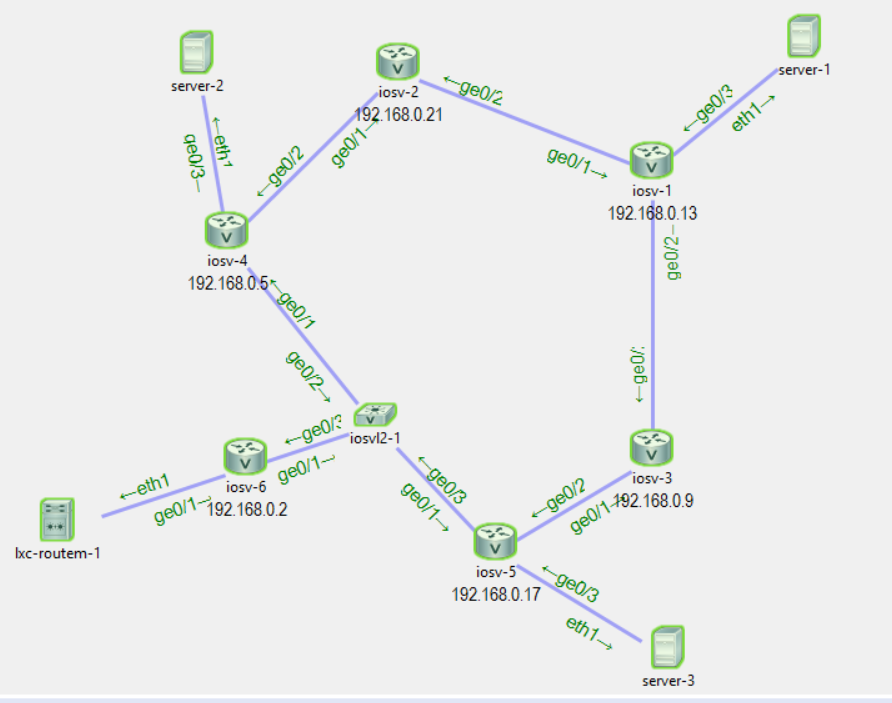
The only reason iosv-2 may choose 10.1.0.2 as next hop is because it has more specific mask for 10.1.128.2. You can see that any other IP in that network is still transmitted to 10.1.0.3.

# Investigating different topologies

We will now try to perform the attack on different topologies ranged on their policies, AS-path distance, mask specification and more.

## Victim and attacker are in Multi Home Ass

### Both servers are closer to the attacker

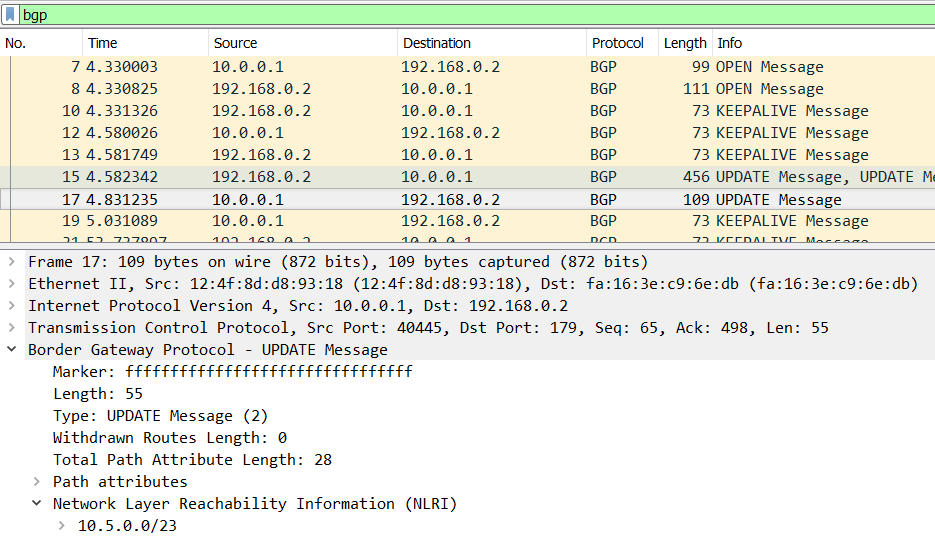


In this topology, the attacker is routem-1 and the victim is server-1. We would like to that our BGP update goes through both channels and affect both server-2 and server-1 decision on the next hop to send their massages to.

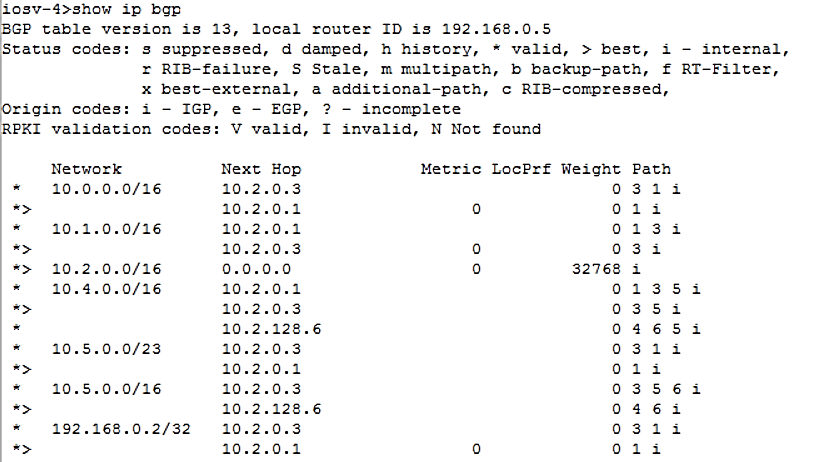
We will try several places for server-2 and server-3 to see how they are affected.

The IP of server-1 is 10.5.0.1, the Routem is 10.0.0.1, sevrver-2 is 10.2.128.1, and server-3 is 10.1.0.1.

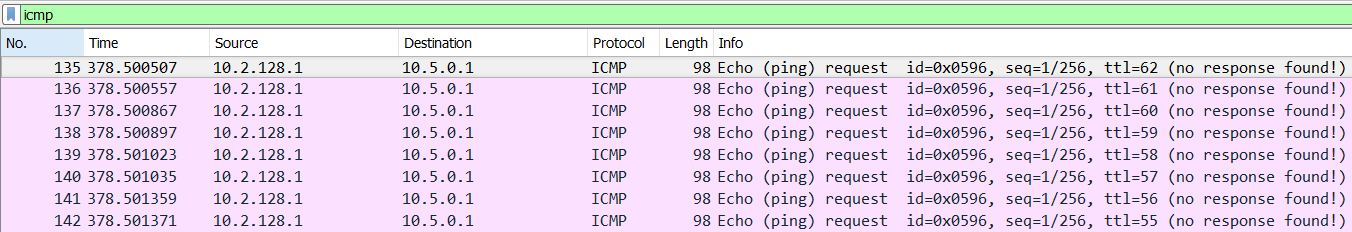
We now perform the attack as explained in the guide and see that the update BGP message was successfully sent:



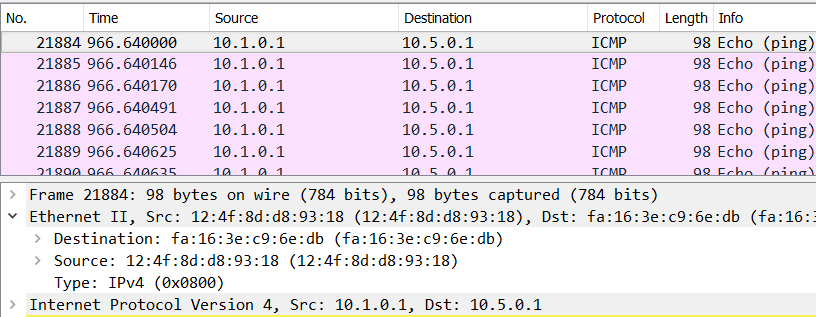
We now look at iosv-4 BGP table:



We can see that now the prefix 10.5.0.0/23 is sent through 10.2.0.1 (the IP of the attacker’s router). We intentionally set the mask to be 255.255.128.0 so the mask of the existing routing prefix and the new one will be the same. The routing goes through the attacker **only** because of his shorter AS-path.

We now try to ping from both servers:

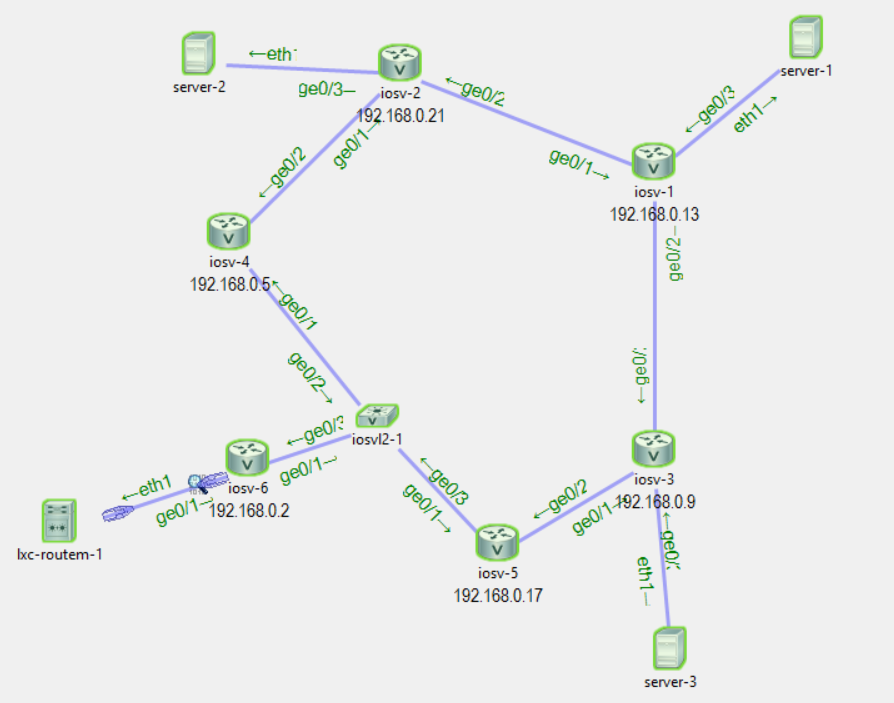
The source is server-2, the destination is server-1 and still the package enters the Routem interface.



The same thing happens with server-3. We can also see that the MAC source is 12:4f:8d:d8:93:18 which is the same as the MAC address of the Routem. The reason is that the ping request is sent to the attacker, and since it is not is IP address, the attacker sends “no response found!” message.

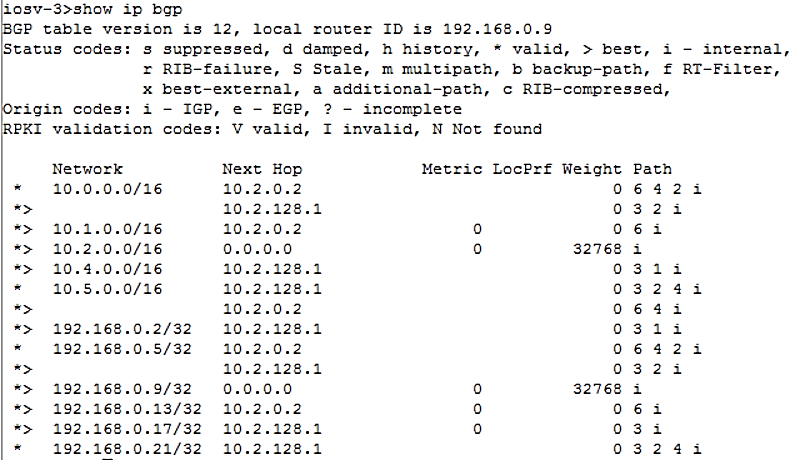


### Both servers are closer to the victim



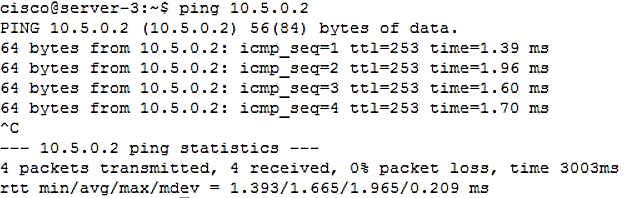
In this case both servers are closer to server-1 than to routem-1, so the BGP attack should fail.

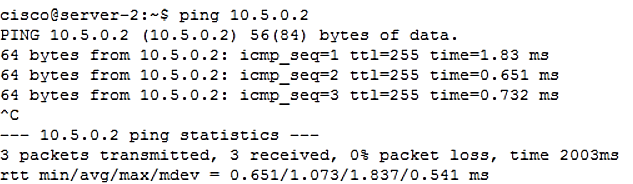
We can see that the BGP routing paths of iosv-3 haven’t changed:



The path chosen for 10.5.0.0 is still router 10.2.0.2 because it is the closest router.

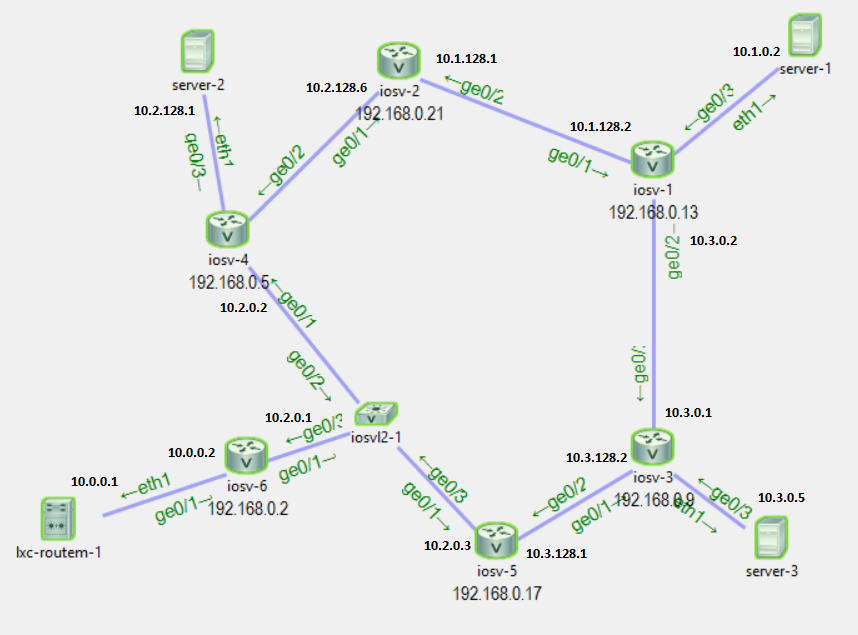
Ping from both servers:





The ping works from both directions, the attack failed.

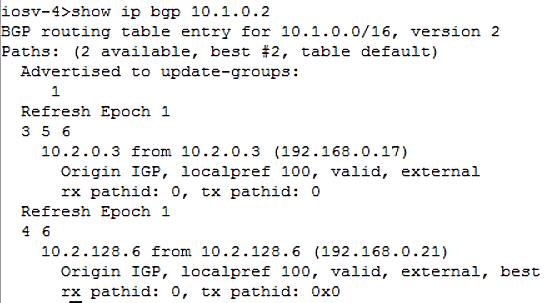
### Server-2 is closer to the attacker and server-3 is closer to the victim



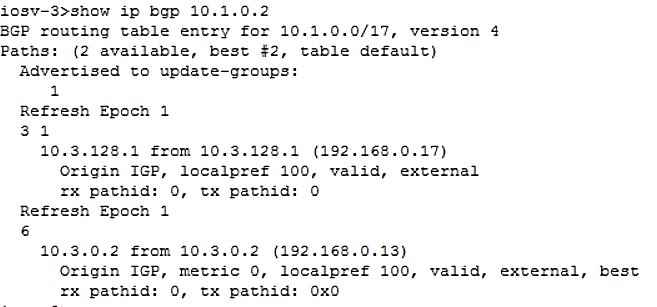
We can see that the traffic from iosv-3 goes through iosv-1 (with IP 10.3.0.2):



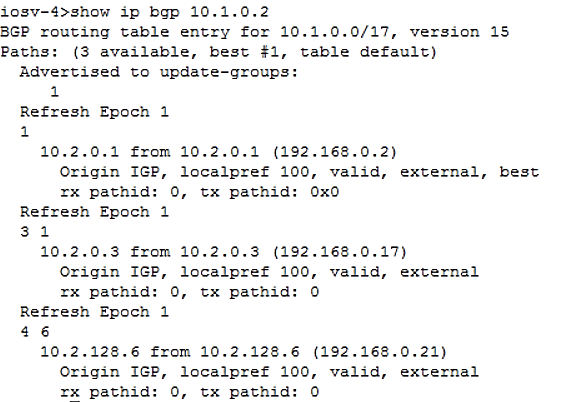
And that traffic from iosv-4 goes through iosv-2 to iosv-1



Perform the attack:



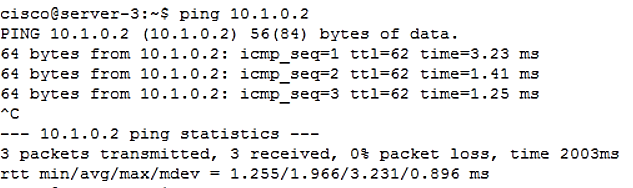
We can see iosv-3 prefers option #2 which is to go through router iosv-1 to the original server-1.



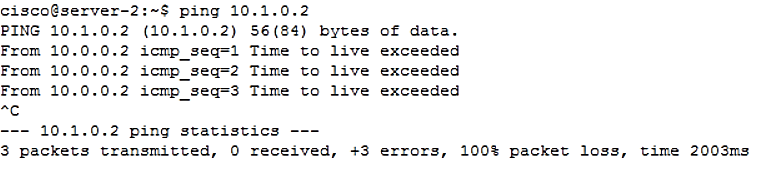
Router iosv-4 prefers the route going through iosv-6 to the victim.

After the attack is performed, we will try to ping the victim from both servers:

Server-3 receives the ping successfully:



While server-2 doesn’t succeeds:



This is a very important asymmetrical behavior that might allow MIT `M attack.



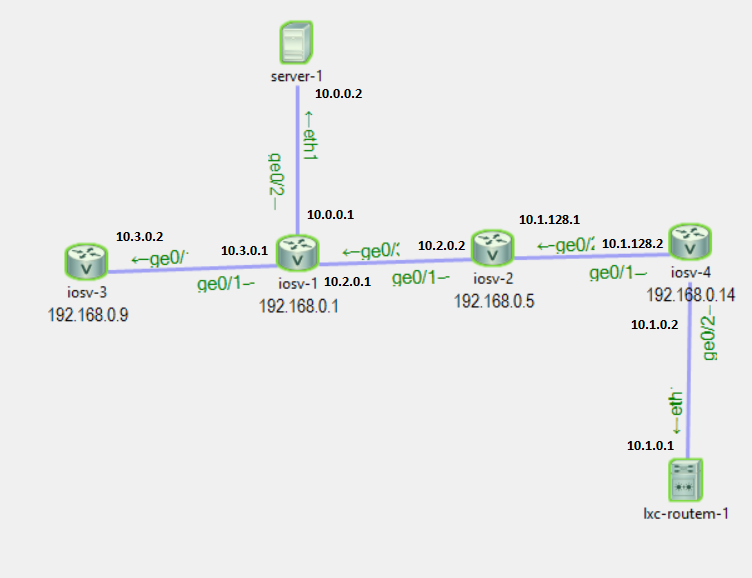
**Conclusion**

It is possible to break the symmetry in the routing graph and make the routers that are closer to the attacker route their packages to the attacker, and these who are far away from the attacker - unaffected



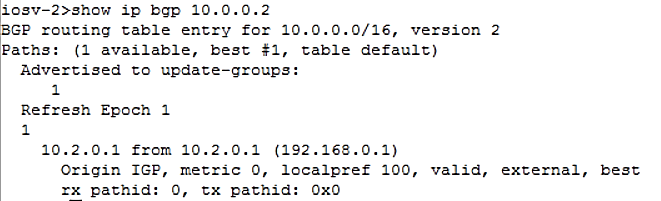
## Victim is in Transit AS

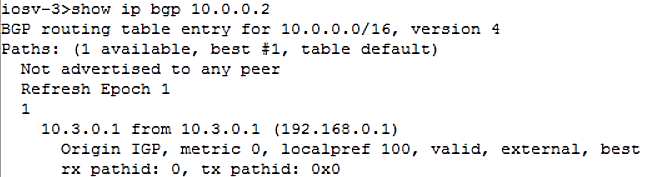
We will look at the following topology:



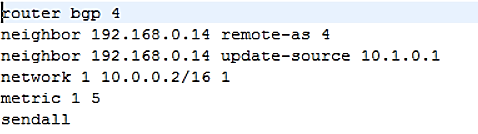
Server-1 will be our victim, whose IP address is going to be spoofed, and the routem-1 will be the attacker. Every router is in its own AS.

Before the attack both iosv-3 and iosv-2 direct their packages through iosv-1 to server-1:

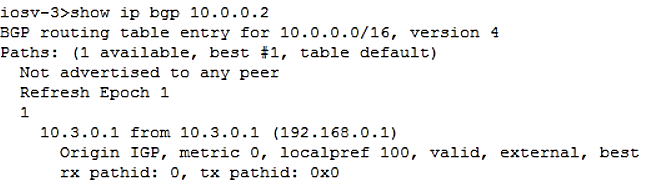




Now we will perform the attack and see what happens:

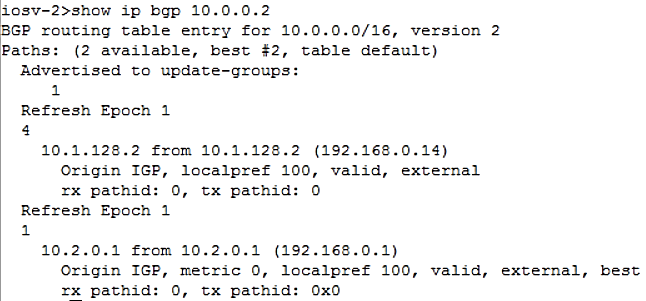


Notice the advertised prefixes are the same.



We can see that is contrary to the previous attack, now there is a router that not only it’s chosen route hasn’t been affected – it doesn’t even acknowledge the existence of a new route.

Iosv-2, in contrary, does see the new rout, but as it isn’t preferable upon the existing route, it still directs its packages to iosv-1:



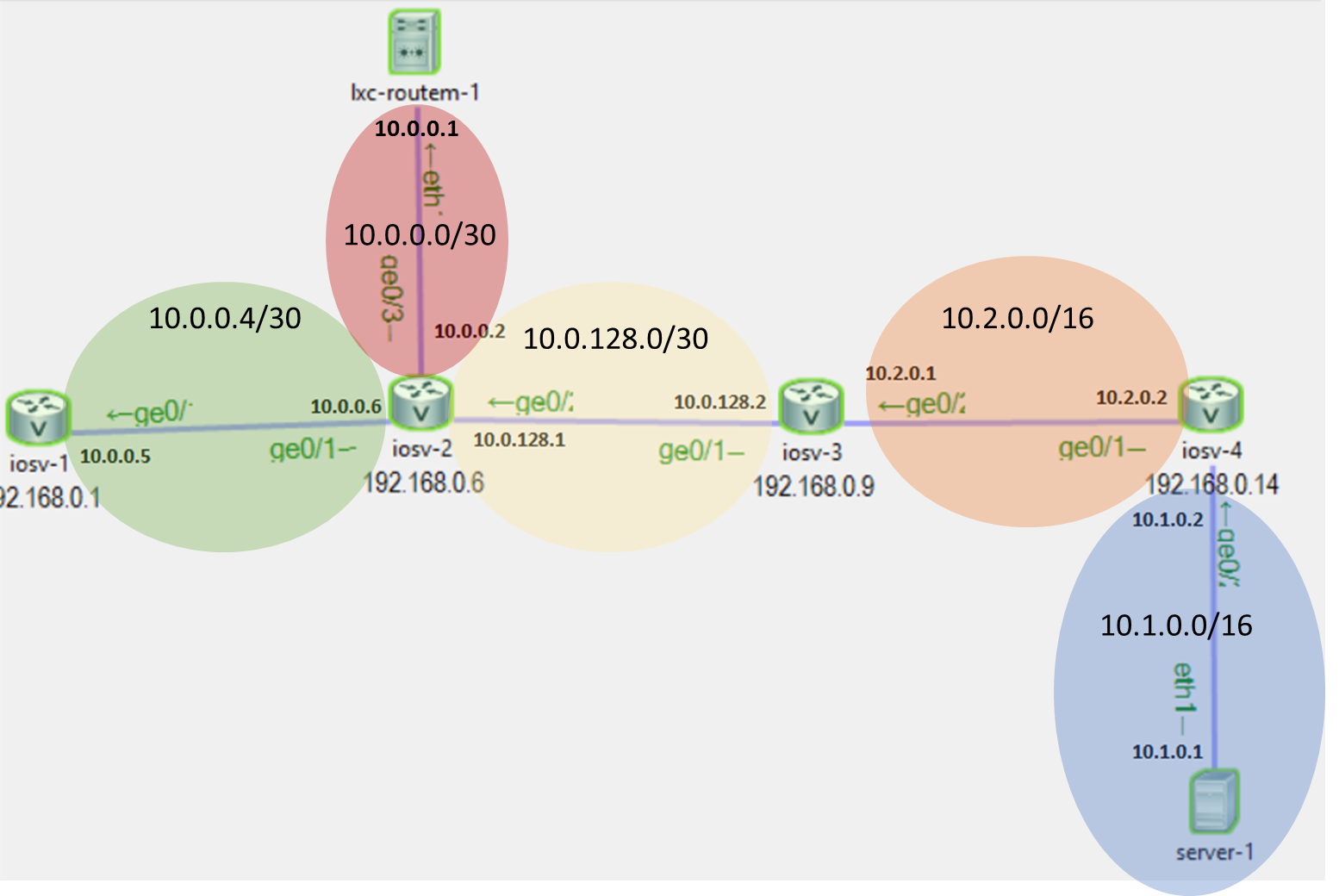


**Conclusion**

If router isn’t affected by the advertisement of the attacker, then any other router s.t. every path between the attacker and y go through will remain unaffected, since won’t advertise the new AS-path.



## Attacker is in Transit AS



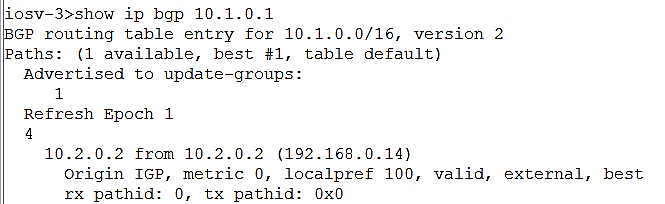
As this network contains different sizes of subnets, each subnet is colored in a different color, and it’s prefix is written upon it.

Routing of iosv-1 to server-1:

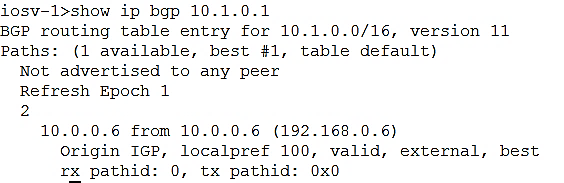


We can see it is routing its packages through the AS-path: .

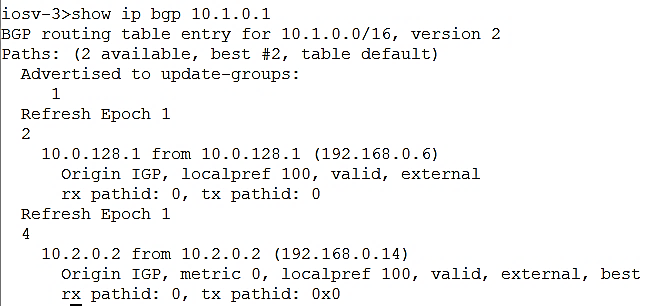
Routing of iosv-3 to server-1:



After the attack is performed:



We can see that not only iosv-1 is routed through as 2 to the attacker, it doesn’t even see the existance of the original route, since iosv-2 was the one to advertise it and iosv-2 was also affected.



We can see iosv-3 sees two different AS-routs, but still prefers the original rout to server-1, as the new route has no beneficial properties.

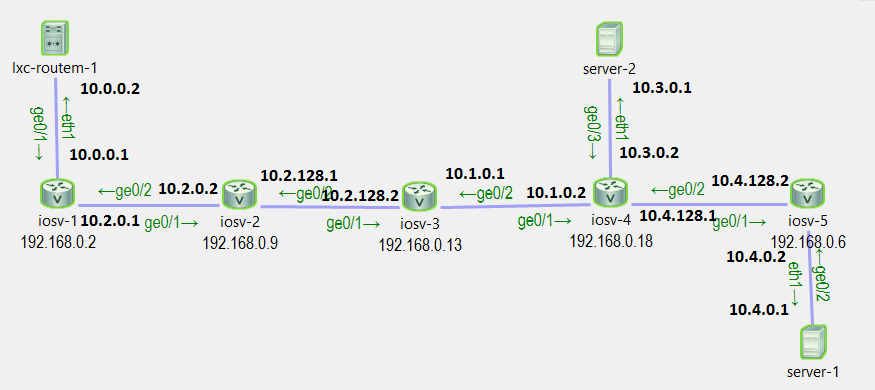


**Conclusion**

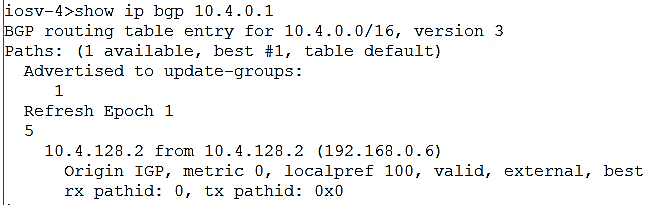
*A router sees both the route of the attacker and the original rout if and only if the advertisement of these two AS-routs are coming from different paths that doesn’t intersect.*



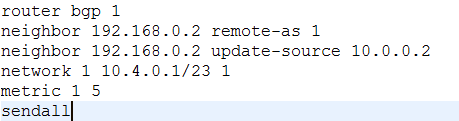
## Attacker has longer distance, but more specified mask



First iosv-4 routs its packets to server-1 via iosv-5:

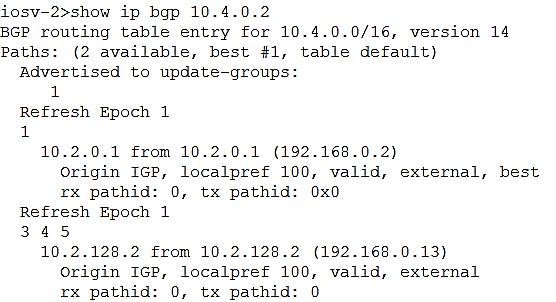


## Same prefix attack:

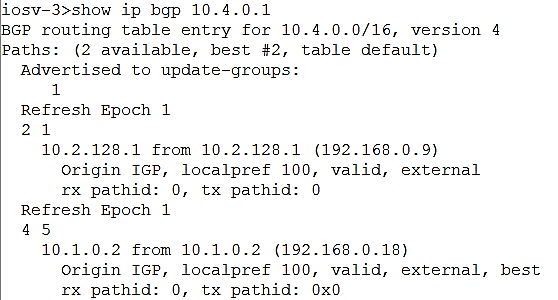


When 255.255.0.0 is the mask routing table entry for the subnet 10.4.0.0 in which server-1 is in (i.e. 16 digits of mask).

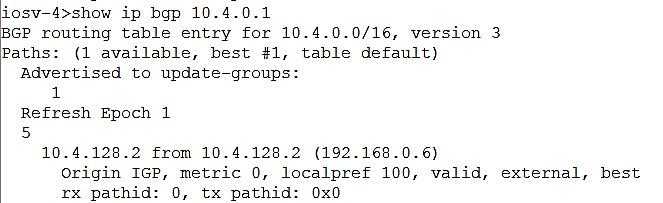
We can see that iosv-2 sees both options and chooses the shorter one to the attacker:



And that iosv-3 sees the conflict between the two routing options, and as the new option has no preferable properties upon the previous one, it still chooses the path leading to the original server-1.

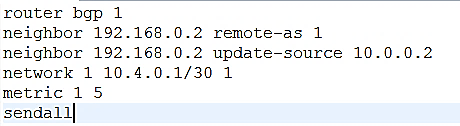


And hence iosv-4 sees only one routing which is the correct one (iosv-3 doesn’t advertise the non-preferred routing):

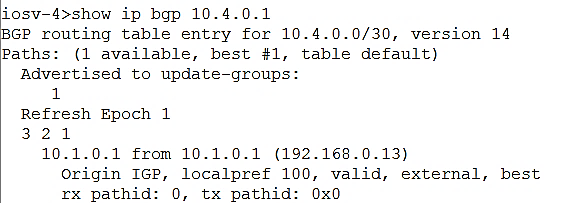


More specified prefix attack:

The attack now looks like that:



The prefix now is 30 which says that we might know a better way to reach 10.4.0.1.



Now we can see that the entry dedicated to 10.4.0.1 has a mask of size 30 and it is routed to the attacker, and doesn’t even recognize there is other path with more generalized subnet mask.

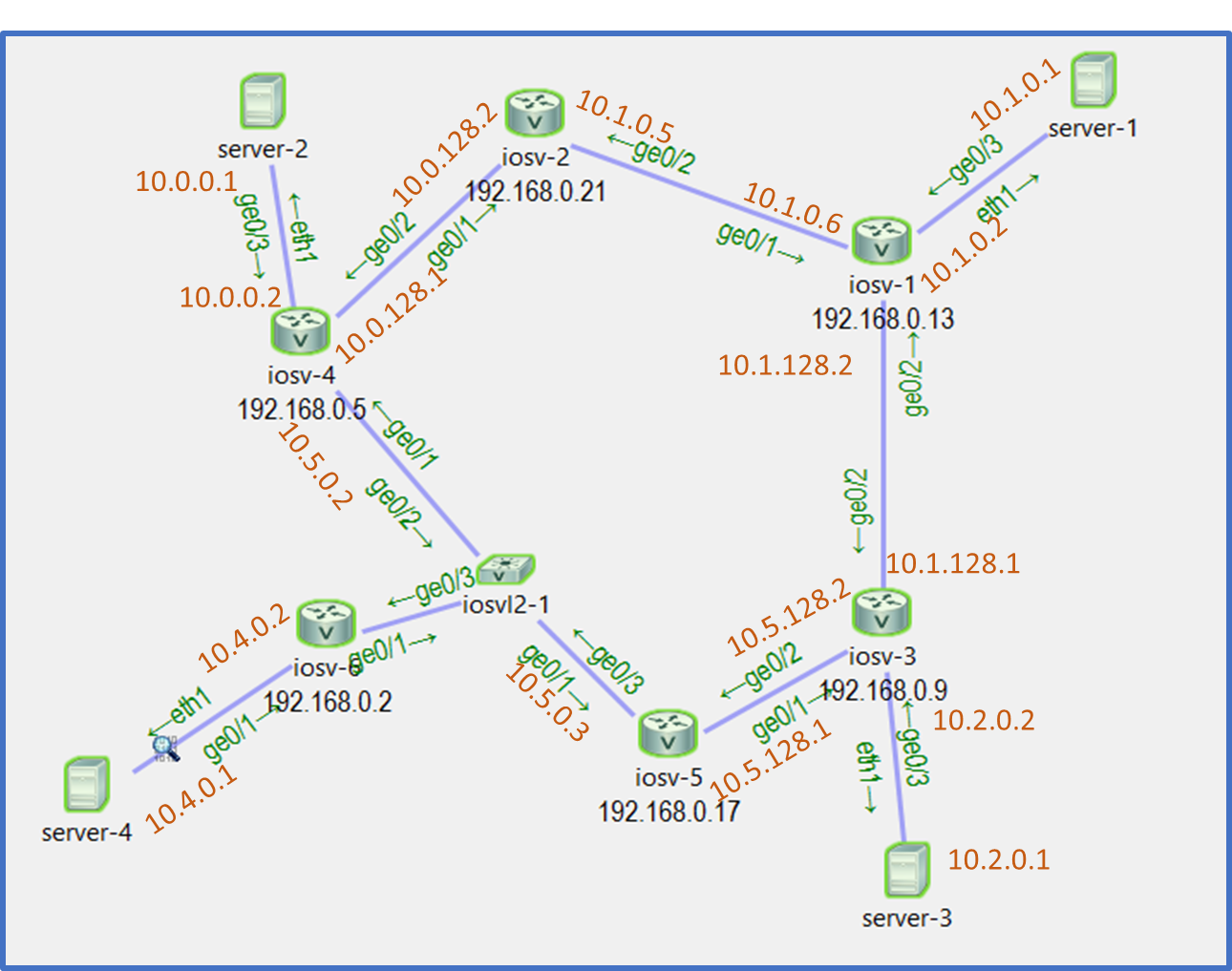
# IP forwarding

As we saw in “[Check if the attack succeeded](#_Check_if_the)”, when we are trying to receive in the attacker side the packets that we’ve stolen, they do reach the interface of the attackers server, but since the IP destination isn’t one of the attackers IPs, it is thrown in the IP layer, and no Ping response is found.

There is no point in making the internet route packets to the attacker, if he can’t read them. We would like to do IP forwarding, and hence we will use “iptables”. Due to the minimalism CISCO wished to apply on each component in VIRL, lxc-routem is very minimal container and doesn’t have “iptables”.

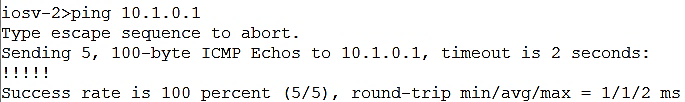
The solution we had found to this problem is to create manually a server that has both the strength of server in VIRL (together with useful other applications) and the strength of Routem. In “[Creating custom server image](#_Creating_a_custom)” there is a user manual to create a similar image to the one we used.

## BGP attack with impersonation



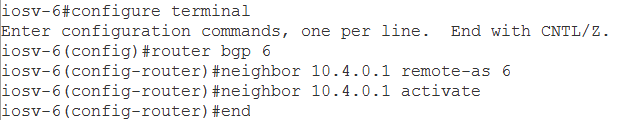
The attacker, server-4, is of the new server type we had created, and the victim is server-1.

### Sanity check: a Ping between server-2 to server-1 works:

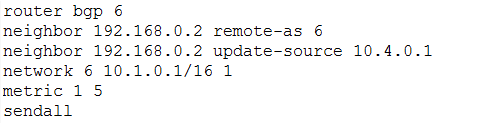


### Reminder: perform BGP hijacking

Establish SSH connection to iosv-6 and run the following commands:

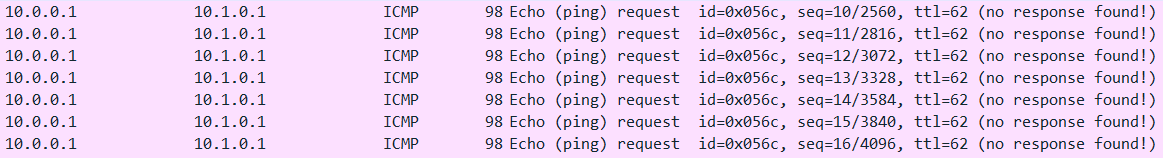


Now Establish SSH connection with server-4, create file named bgp.txt that contains:



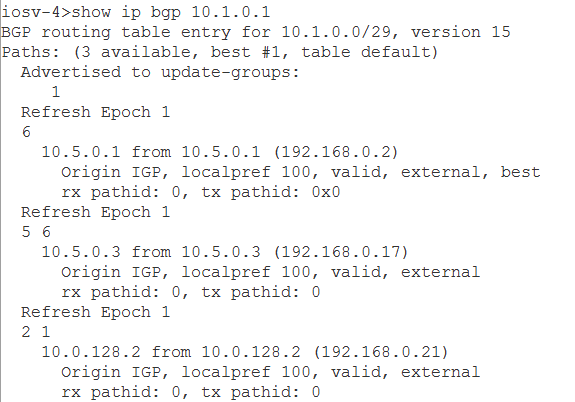
(Notice the mask of the attacked subnet)

Now run tmux, then press ctrl+b% so you will have two parallel terminals. On one of them run “./routem -f bgp.txt”. Create a sniffer on the interface of server-4 and ping again from server-2 to 10.1.0.1.

This is how the sniffing looks like:

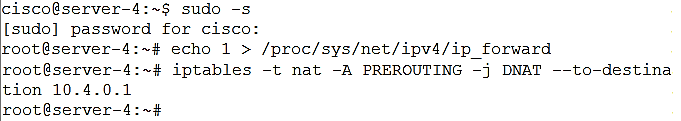
So, as described in the beginning of the chapter, the packets reach the interface of server-4, but are dropped once it’s operating system doesn’t find the IP destination in the IPs of the server.

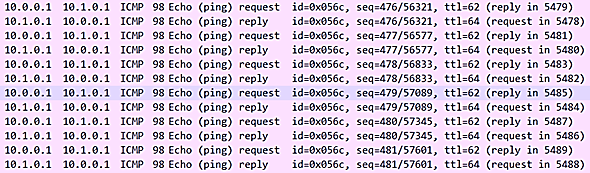
(We know that packets are routed to server 4 due to the BGP table of iosv-4: )



### Perform IP forwarding

In the SSH connection with server-4 run the following commands:



Now download again the pcap file:

Not only a ping reply have been sent – the source IP is the one of the victim, server-1.

Similarly a “nc” between server-2 to 10.1.0.1 (the ip of server-1) is under the control of the attacker:



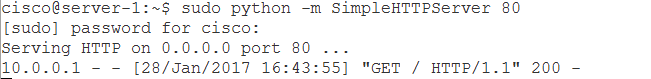


## Delivering fake news

We have set up an HTTP server at server-1, who’s ip is 10.1.0.1. this server will simulate a news site such as YNET. Home page is called index.html:



Setting up the server:

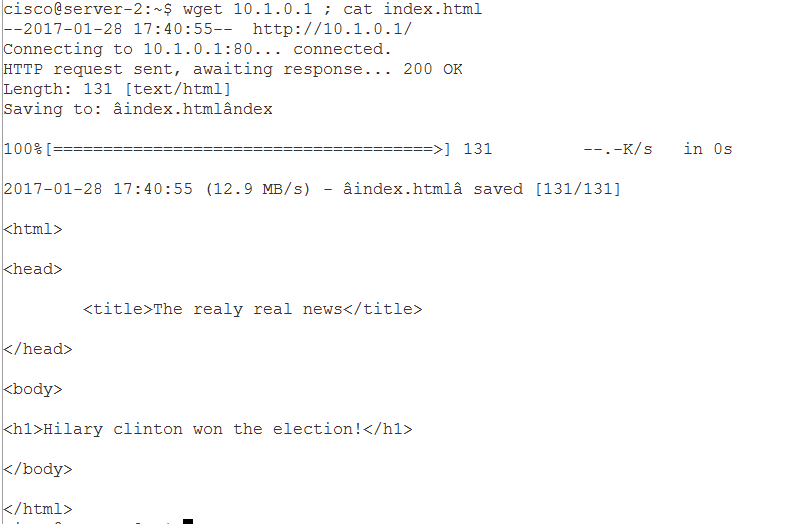


When the client requests the home page via HTTP GET:



He receives the real news page.

We have set up a second http server on the attackers node, which delivers fake news. After we activated the impersonation attack with ip forwarding described above, anyone who surfs the web for news at 10.1.0.1 will receive the following:



Now some people believe Hilary won the election!

# 

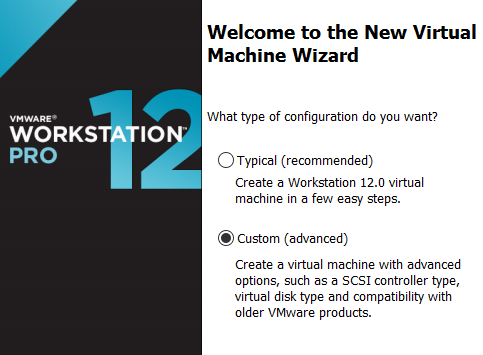
# Creating a custom server image

Creating a custom image is required for several reasons:

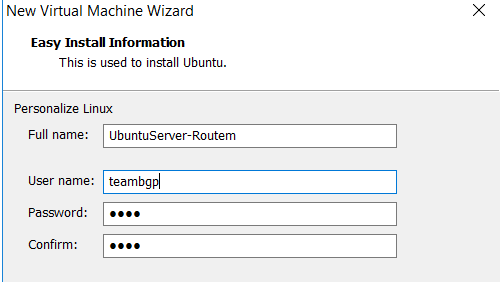
* Most importantly, as we didn’t have external connectivity from within the simulations, any tool we want to be included in the relevant node Image must be pre-loaded.
  + One of the most important such tools in our cause is Routem.
  + Scapy module for python is also useful for MITM purposes
  + Tmux for more efficient work
* Creating a custom node sub-type allows us to control the amount of resources it takes.
  + If we know our server will perform only lightweight tasks, a reduction in RAM is possible

## How we did it

Install vm workstation, and create a new vm:

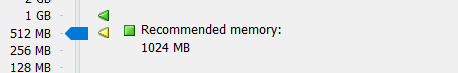


We used the latest Ubuntu server image



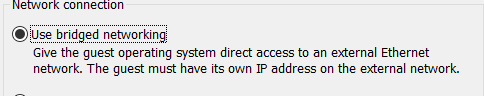
We chose a machine with a single processor.

We are then prompted to set the amount of RAM, this is crucial because we could set it lower than the original server node, thus **making our custom node more resource efficient**:

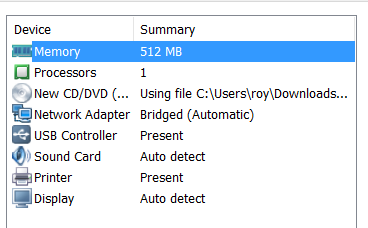


Next, we are prompted whether to use a dynamic disk image or a whole disk image. **You must select whole disk image**, because later on we would need to convert the vm workstation image format (vmdk) to qcow2 format. This is possible only with a complete whole disk image.

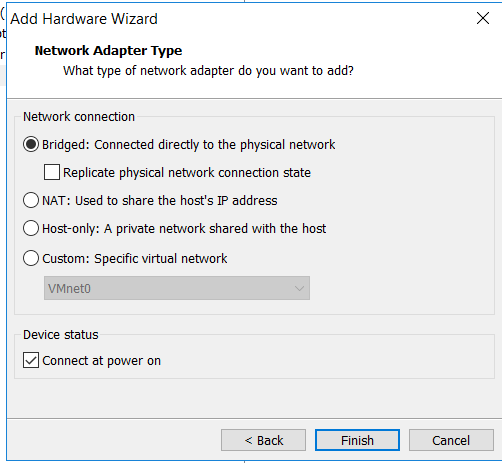
When prompted for network connectivity option, choose bridged networking to give the node direct network access:



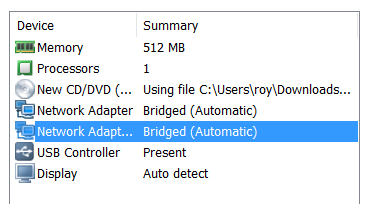
After that we are given a summary of the vm’s resources:



On that screen, we have an option to add a network adapter, so we would have two NICs on the machine. By default, you will only have 1 NIC and Routem requires 2 NICs. NIC 1(eth0) is used for management of the server if needed:

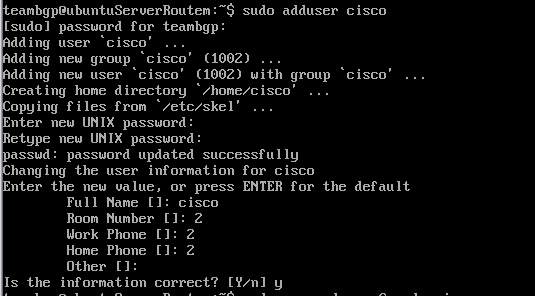


Bridged connection should be chosen for the second NIC as well. Now we can see we have two NICs:



We boot the vm, and installed tools we need like tmux, and the scapy python module.

Now, since virl automatically tries to use a user named cisco whenever it connects to a node, we add another user to our vm with the name cisco, and add it to the sudoers so it has enough permissions:

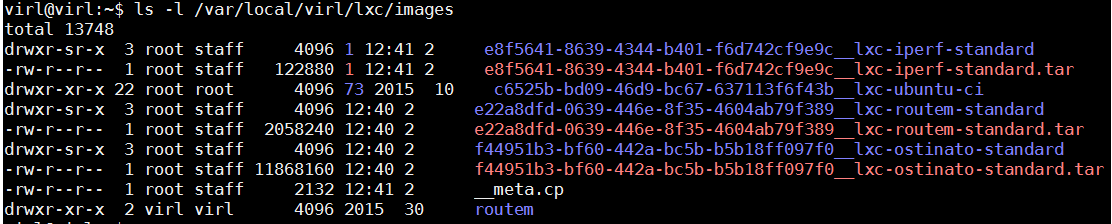






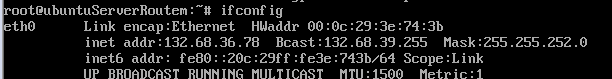
Now we need to get the routem executable onto our vm.

First, we locate the compressed tar routem lxc on the virl server, and uncompress the tar:

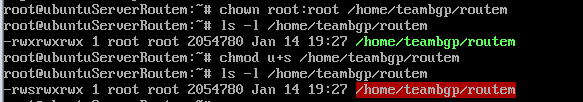




We looked up the IP of our vm



Then we copied the executable to our local machine via ssh, and then ssh’d to our vm and copied it there. We gave routem root setuid rights, meaning we switched to root, took ownership of the routem file (chown), and set the setuid bit on it (meaning that any user that will run it will run it with the privileges of it’s owner = root):



We have set it’s permissions to 777, and ran it to validate everything is ok:

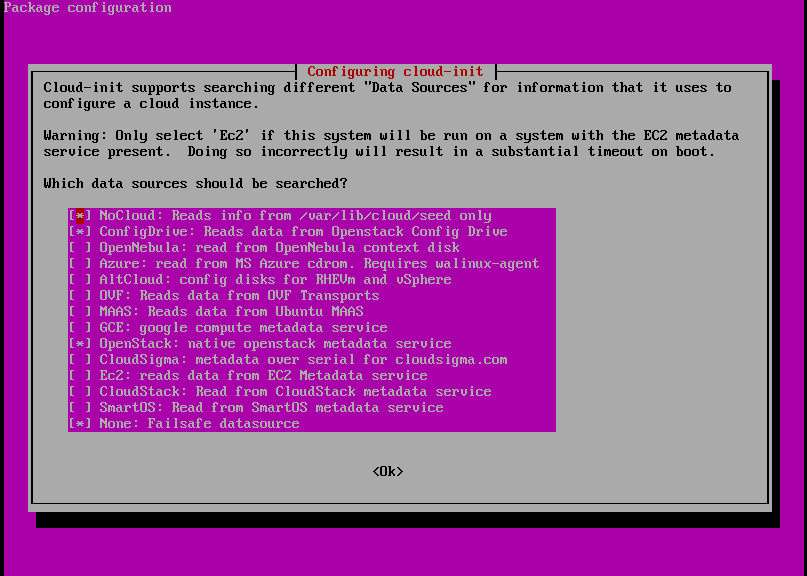


Notice that we have got an error message: config file name is NULL, that is because we didn’t pass any routem configuration file as a parameter, that is OK!

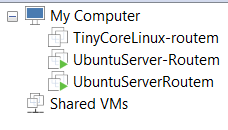
Virl uses cloud-init in order to initialize it’s nodes prior to boot.



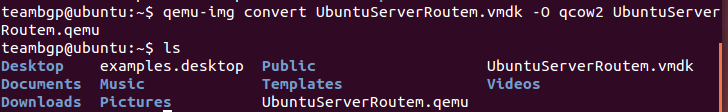
The Ubuntu server image comes pre-loaded with cloud-init. The problem is, that without external internet connectivity, cloud-init stalls the boot times drastically, because it tries to search a set of data sources over and over again. The solution to this is to reconfigure it and delete all of the problematic data sources: 



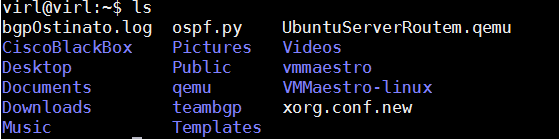
Now, since we want to convert the disk image of our vm to a different format, we opened yet another vm (since we need a linux machine, and we cant modify our vm’s disk image from within itself – inception):



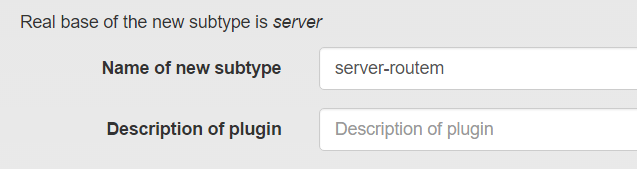
We have transferred the vm’s disk image (vmdk) to the new machine, installed and ran qemu-img in order to convert the image types:



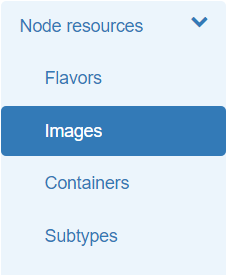
We transferred the created qcow2 image to the virl server:

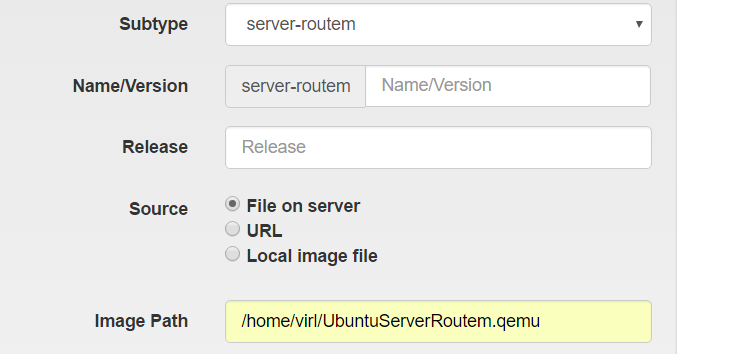


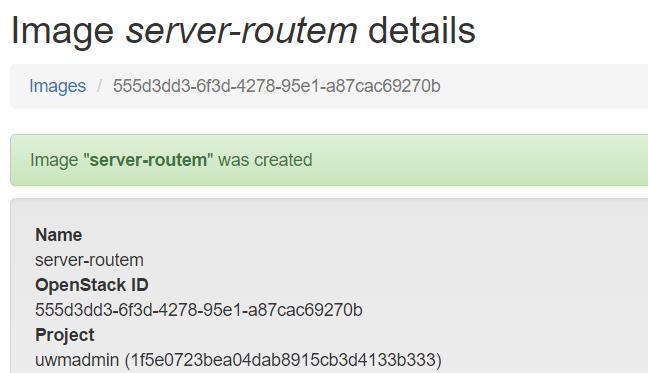
On the UWM, go to node-resources -> subtypes, and create a new subtype for our node (make sure to select cloud-init instead of disk):



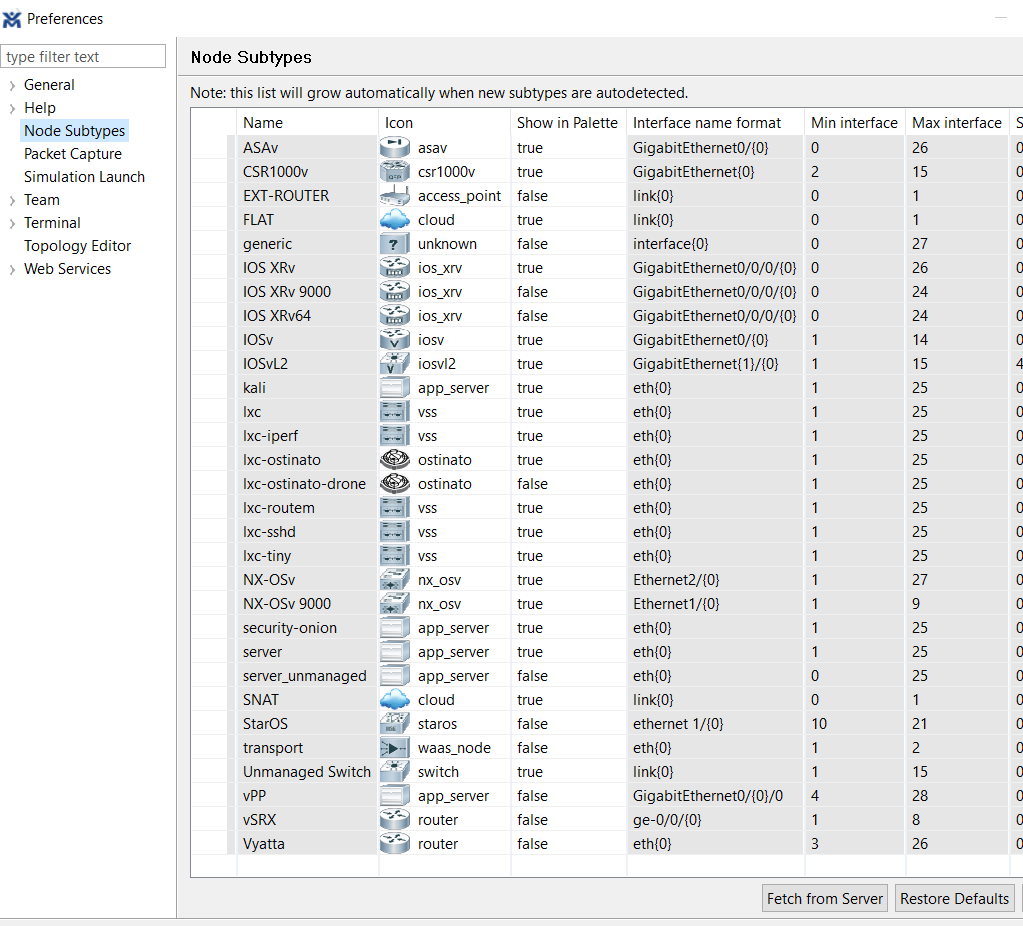
Then go to images and create a new image with our subtype and point it’s image file to the one we have uploaded to the virl server:





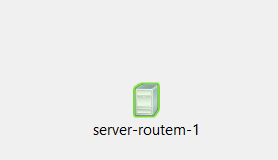


On virl, go to file -> preferences -> node subtypes -> fetch from server

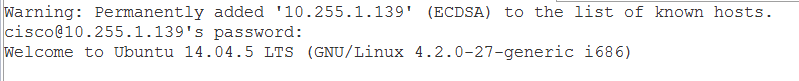




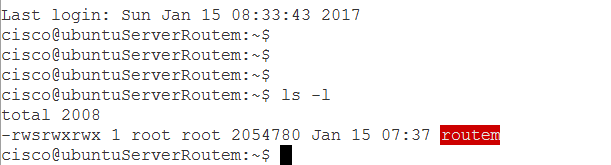
Now we can add the node to our simulation:



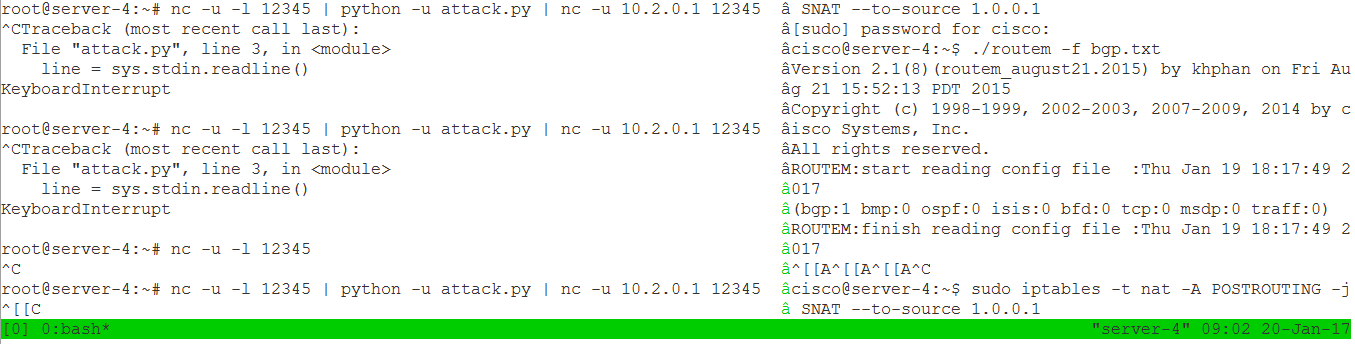
Now we ssh to our active node:



We can see that indeed we have routem!

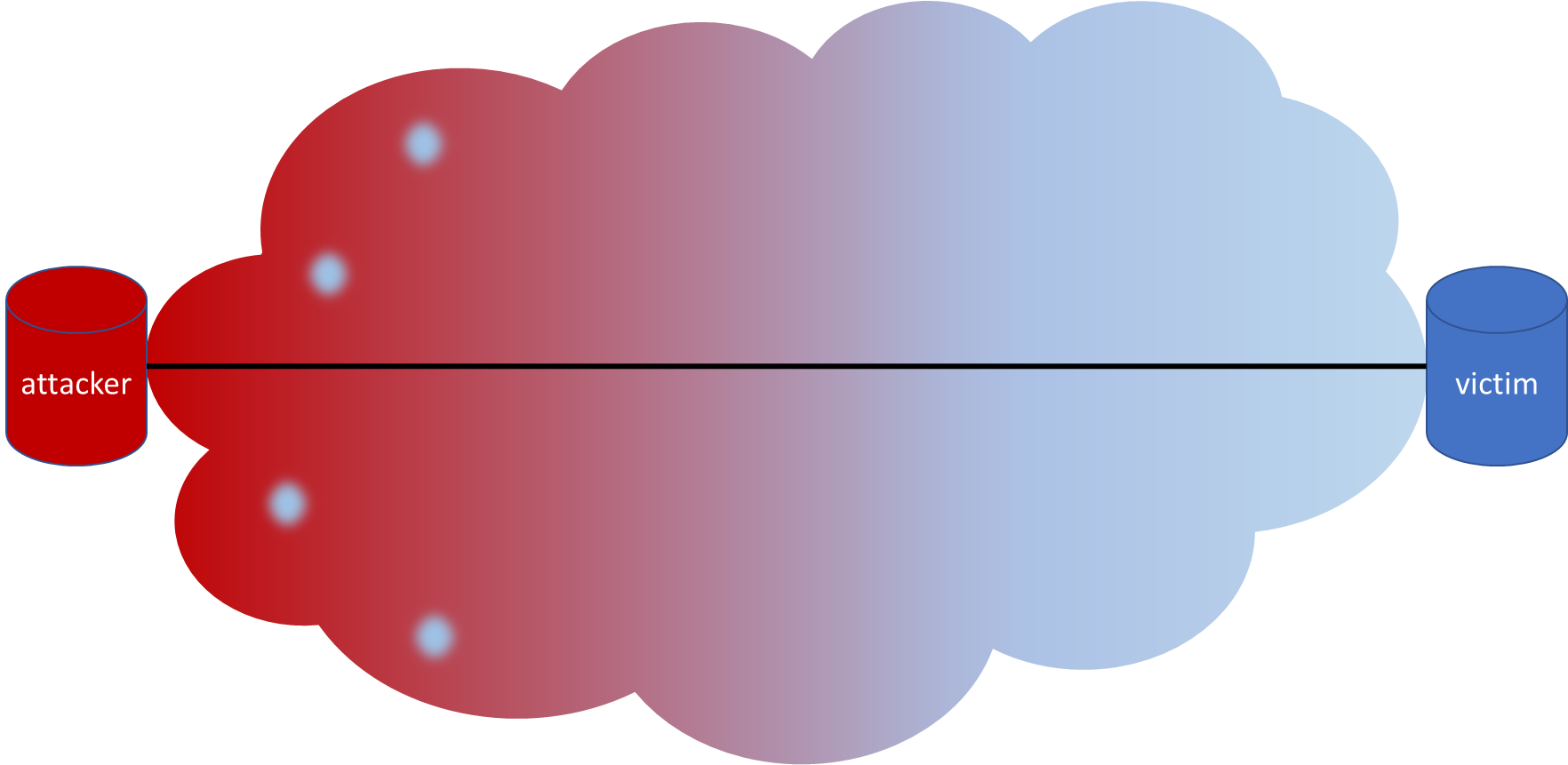


And all of the other tools we have installed , like tmux:



# MITM attack:

We now limit ourself to a BGP attack that advertises subnet mask that is equal to the original subnet mask of the victim. After such an attack the internet will look like that:



Meaning the red zone is affected by the attacker and will route it’s packets to him, and the blue zone wasn’t affected, and still route its packets to the victim. We can see that although the ‘AS’ closeness of some of the routers, due to their lack of choice (determined by the AS policy), they are still routed to the victim.

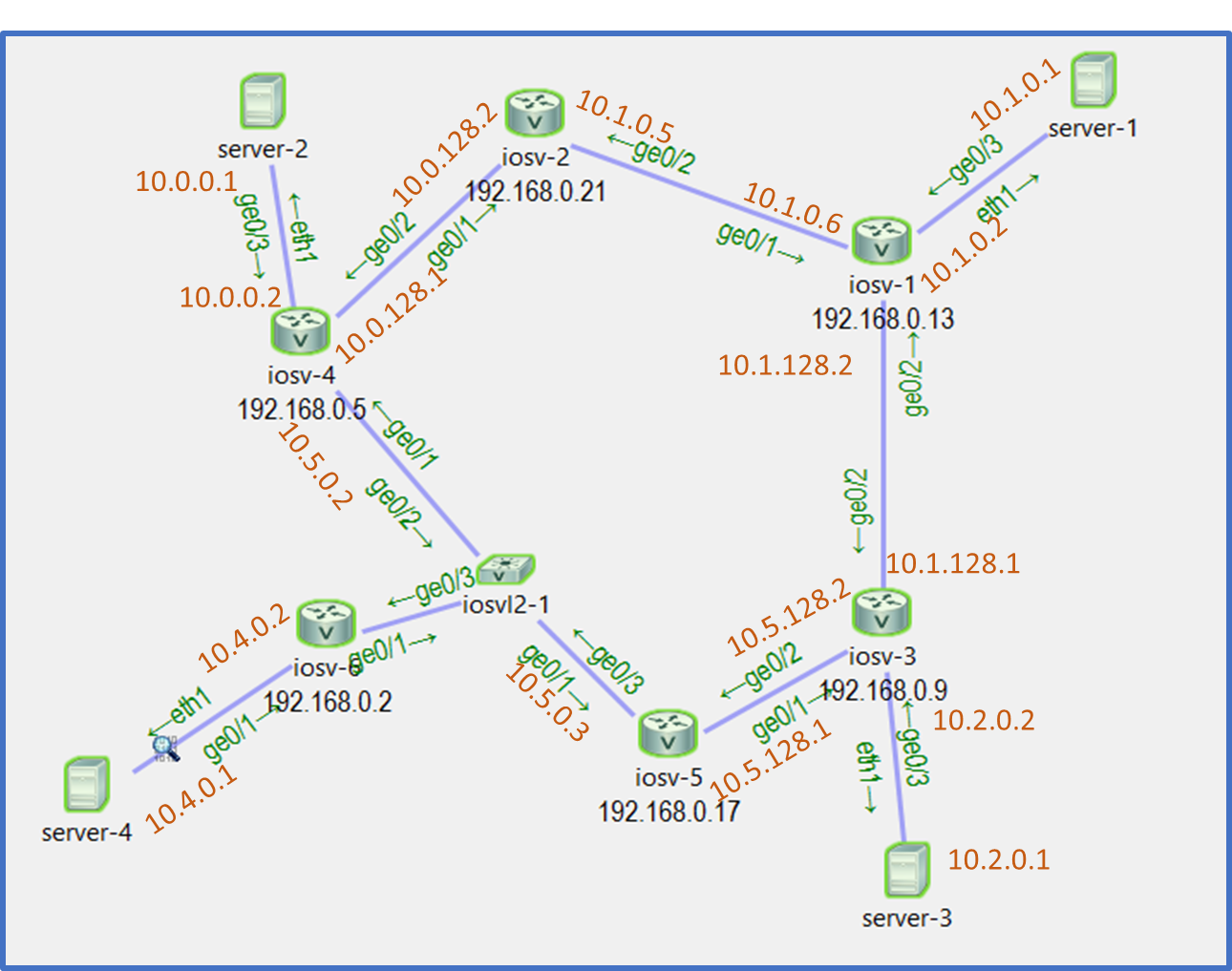
In the MITM attack, we take advantage of this asymmetric property of the internet. We will use collaborator in the “blue” side of the internet that won’t be affected by the attack, send it all the packets that needs to be sent to the victim, and then it will send to the victim these packets with no interference due to the fact that the attack didn’t affect this probe.

This collaborator is much more simple than the attacker:

* It doesn’t need the AS manager authority to use BGP
* It doesn’t have to be geographically distant from the attacker, as it can be only “half” the distance between the attacker and the victim.

We will now see a MITM attack using BGP.

We will look again at the topology given in the section “[BGP attack with impersonation](#_BGP_attack_with)”:



As discussed in the section [Server-2 is closer to the attacker and server-3 is closer to the victim](#_Server-2_is_closer), in this topology, if one uses a BGP attack that advertises subnet mask that is equal in its length to the real subnet of the victim (server-1), then routers between the attacker and the victim will be affected only by the AS-path advertised. So the only routers to change their BGP routing table are these who are connected to several routers and have a manner of free choice (i.e. not strictly policed to go through a certain AS that is not affected), and that are closer to the attacker. In our topology, server-2 will change its routing to go to server-4, and server-3, that saw no preferability in iosv-6s’ advertisement, will keep its’ BGP routing table untouched.

We will start by showing an example of traffic going from server-2 to server-1 before the attack:

Open a ‘nc’ communication between server-1 to server-2:



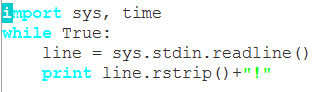


We can see that server-1 gets the message as expected. We will now perform a [BGP attack with impersonation](#_BGP_attack_with) with a slight change: the output of the ‘nc’ program of the attacker will be given to a python script that manipulates it and then will be sent to server-3 (10.2.0.1). Server-3, in turn, will take the output of its ‘nc’ program and will send it to server-1 (10.1.0.1).

The attack is:



The python script is:

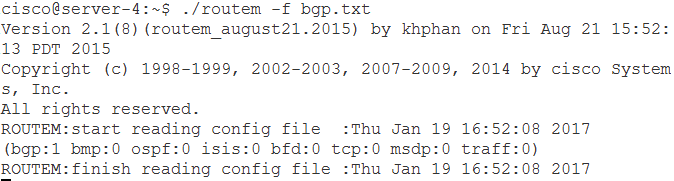


Taking each input from the user and concatenate a “!” to it.

Server-3 transmitter activation:



Server-4 attack:



And then activation of transmitter:



Server-2 message sent:  

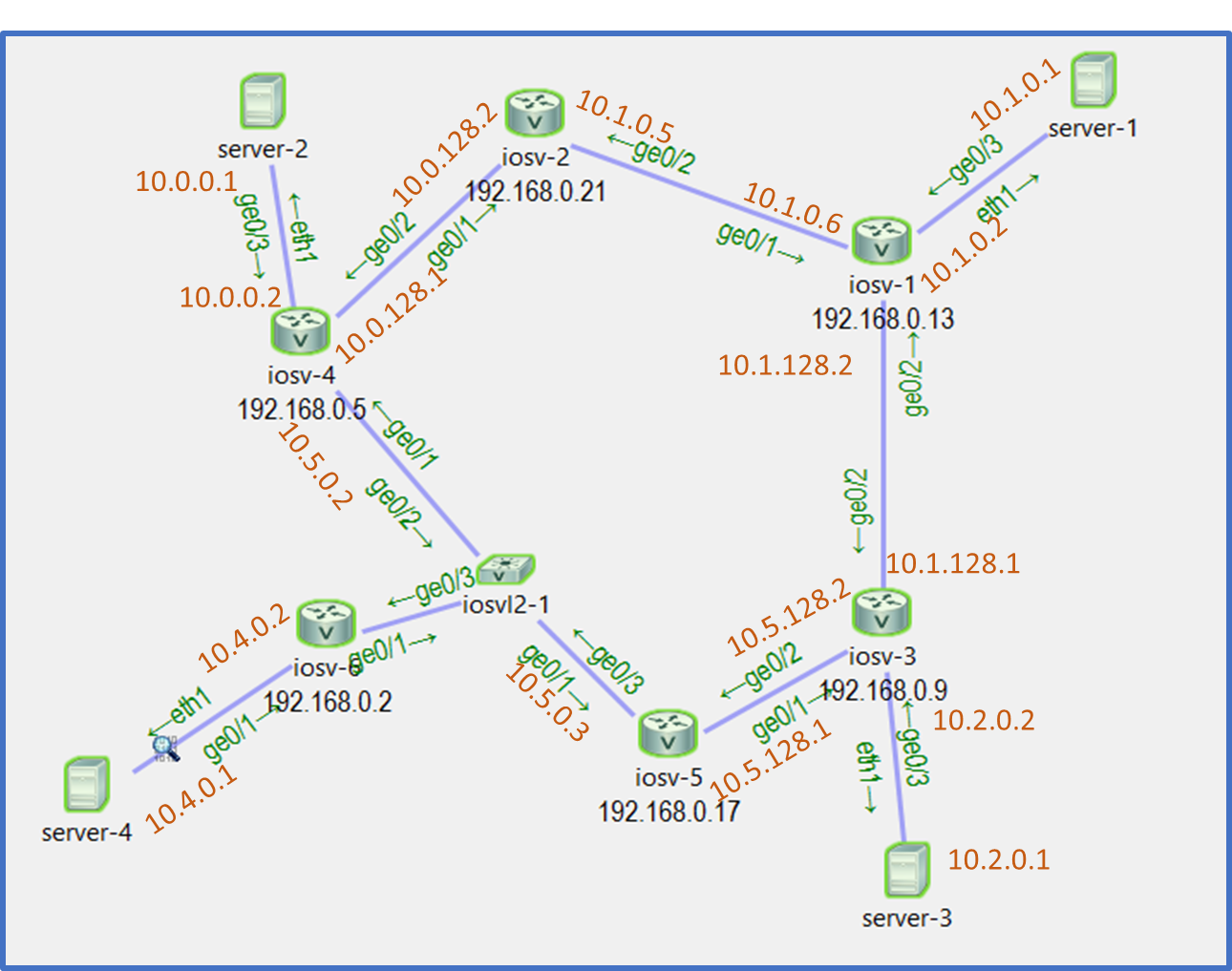

Server-1 message received:



This BGP attack is pretty weak because server-1 can see the IP source of the packet is server-3, and hence understands it is under attack or not being able to link between the received packet and an already existing session with server-2. This is why we would like to spoof the IP of server-2, In order to perform the attack in the middle of a session between server-1 and server-2.

## Summering example

Run the previous simulation.



### Step 1 – sanity check

Before the attack begins, we would like to start a new session between server-1 and server-2.

Run the following commands:

*On server-1:*

*nc -u -l 12345*

*On server-2:*

*nc -u 10.1.0.1 12345*

Send a message between them to ensure the session is up.

### Step 2 – Black hole attack

Open SSH connection to iosv-6 and run the following commands:

*configure terminal*

*router bgp 1*

*neighbor 10.4.0.1 remote-as 1*

*neighbor 10.4.0.1 activate*

*end*

Open SSH connection to server-4, create bgp.txt file with the following content:

*router bgp 1*

*neighbor 192.168.0.2 remote-as 1*

*neighbor 192.168.0.2 update-source 10.4.0.1*

*network 1 10.1.0.1/30 1*

*metric 1 5*

*sendall*

And then run:

*sudo -s*

*echo 1 > /proc/sys/net/ipv4/ip\_forward*

*iptables -t nat -A PREROUTING -j DNAT –to-destination 10.4.0.1*

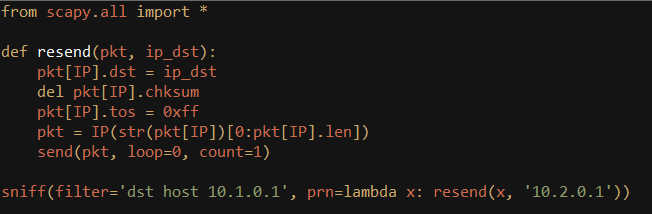
*// consider assisting tmux, as the following command is blocking*

*./routem -f bgp.txt*

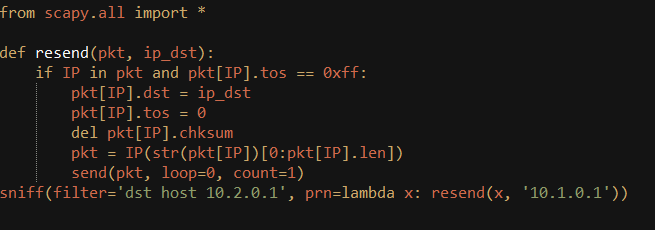
Check the messages don’t get from server-2 to server-1 anymore, and run nc -u -l 12345 in parallel on server-4 to see it does reach server-4. We would now want to redirect it to server-1.

### Step 3 – Non-interfering MITM

Create the following python script on server-4 (name it attack.py):



And the following python script on server-3:

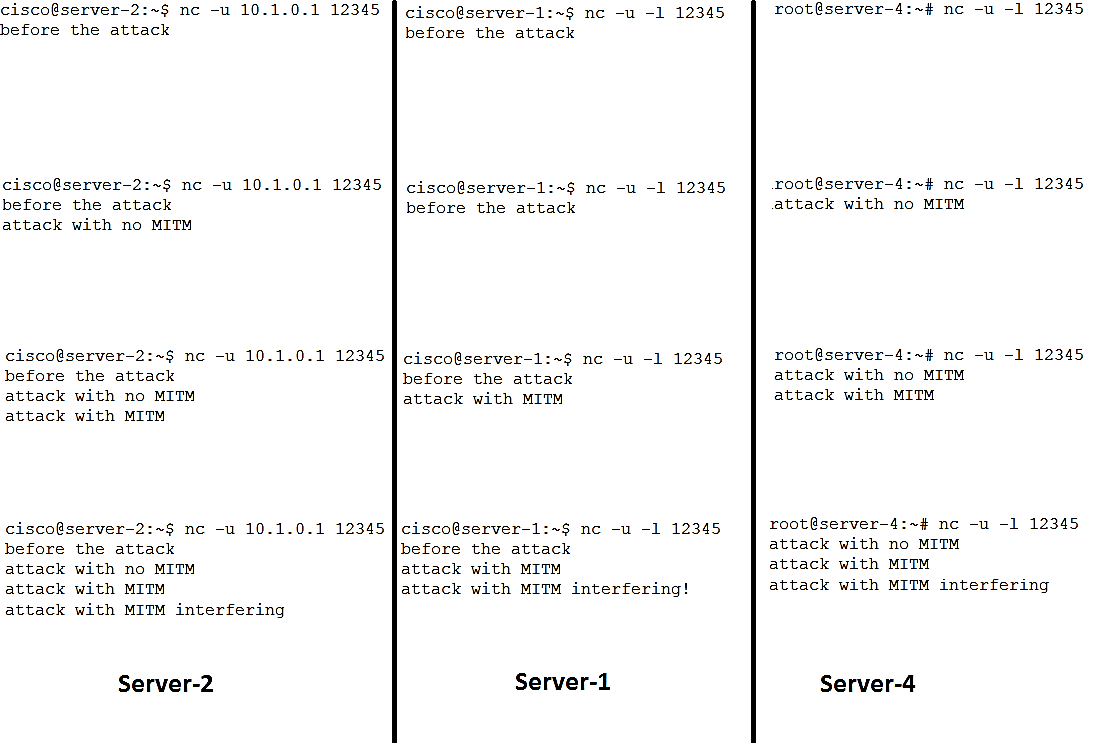


Run these scripts on the servers (with sudo permissions) and now see the messages again reach server-1, but this time, we can also see them in server-4.

### Step 4 – Interfering MITM

If you want to alter the message of server 2, add the following command to the python script of server-4 (adds a “!” in the end of the row):

*pkt[Raw].load = pkt[Raw].load.strip() + “!\n”*

This is how the change in the transmission should look like (when each row is after a step is done):

# Conclusions

One of The main ideas of the project was to use VIRL to simulate BGP attacks, study these attacks and reach general conclusions regarding them. We will now present the conclusions:

* How easy is the execution of the attack from the attackers stand point?
  + In what manner did the ISP need to cooperate with the attacker?
    - The BGP speaker of the AS needs to acknowledge the attacker as a BGP speaker as well (neighbor <ip> remote-as; neighbor <ip> update-source). The rest of the attack can be done solely by the attacker.
  + Does the attacker need to deal with race conditions?
    - If the attackers offer is “better” than the original one, then it doesn’t matter when the attacker will advertise its attack. If it is equivalent by all parameters, then the first to advertise, is the one to win, though it seems highly improbable to win this kind of race, since there is no request for network advertisement, but spontaneous advertisement occasionally of each speaker.
  + Must the attacker use special hardware?
    - Not hardware, but software: the attacker must be able to speak the BPG protocol. In our simulation, we used “Routem” software to participate in the BGP protocol.
  + How much code does the attack consume?
    - Even with MITM attack it is no more than 20 lines of bash and python code. The major difficulty is testing this code in real life. We based heavily on sniffers along the topology, a privilege real life attacker doesn’t have.
* What is the radius of influence (how many ASs can be infected)?
  + The answer depends on the length of the subnet mask being advertised. If the mask is longer (and therefore more specific) than the original one, then the attacker can make even the original AS to believe it owns the subnet. If the masks are with equal lengths, then the internet will divide half way between the attacker and the victim, the routers closer to the victim will keep their routing tables pointing to it, and the routers closer to the attacker will update the routing table to point to the attacker.
* How long does it take the attack to propagate?
  + In the simulations, it was a manner of seconds, we cannot infer from these simulations on a real-life attack.
* In which situations, such an attack is impossible?
  + Since the internet is a highly-connected mesh, there is a path between the attacker and any router in the internet so it has potentially the power attack anyone anywhere. That said, it still needs the permission of its AS manager, and if it wishes to perform MITM attack, if most the world is closer to the victim than to the attacker, the influence of the attack will be minor (You can think of a huge ISP in U.S.A for example, against small terror group in Kazakhstan).
* In which scenarios, such an attack is relatively easy?
  + Governments that have access to big ASs can use their power pretty easily on the BGP protocol. If it isn’t even suspicious that the communication goes through the attacker AS, it might be unnoticed as well.
* How easy is it to detect BGP attack?
  + The attack we have performed is very easy to detect, run “tracert” and see that the path has changed. It becomes harder if the attacker is inside a big transit AS that isn’t suspicious that the communication of the victim goes through it, or if the attacker managed to discover the previous path, and respond with false answers imitating the original path.
* Is it easy to track the origin of the attack (Is it even possible)?
  + One can see that the path has changed, but to know who advertised it, there is a need of cooperation with the AS. If the attacker didn’t advertise that he owns the subnet, rather advertised it has a path to the subnet, than it will be harder to detect.
* What measures could the AS manager take to prevent such an attack? Make it more unlikely?
  + If a BGP speaker gets an advertisement of a more specific subnet mask, it can check whether the AS advertises this mask was also the AS to advertise the previous more generic mask.
  + For a MITM attack, Suspicious AS managers can share their AS path towards one another, and see if any dramatically change was done in one direction.