



5 FASES

Oracle VM VirtualBox Administrador

archivo Máquina Ayuda

Herramientas

Nueva Configuración Descartar Mostrar

General

Nombre: xema-c8_default_1602217795622_25902

Sistema operativo: Red Hat (64-bit)

Sistema

Memoria base: 512 MB

Orden de arranque: Disquete, Óptica, Disco duro

Aceleración: VT-x/AMD-V, Paginación anidada, PAE/NX, Paravirtualización KVM

Previsualización

Pantalla

Memoria de vídeo: 16 MB

Controlador gráfico: VBoxVGA

Servidor de escritorio remoto: Inhabilitado

Grabación: Inhabilitado

Almacenamiento

Controlador: IDE

IDE primario maestro: CentOS-8-Vagrant-8.0.1905-1.x86_64.vmdk (Normal, 10.00 GB)

Audio

Controlador de anfitrión: Windows DirectSound

Controlador: ICH AC97

Red

Adaptador 1: Intel PRO/1000 MT Desktop (NAT)

USB

Inhabilitado

Carpetas compartidas

Ninguno

Descripción

Ninguno

w10

Apagada

windows servers 2012

Apagada

wifislax

Apagada

vagrant

Apagada

x_default_1602217377046...

Corriendo

xema-c8_default_...

Corriendo

Seleccionar Windows PowerShell

```

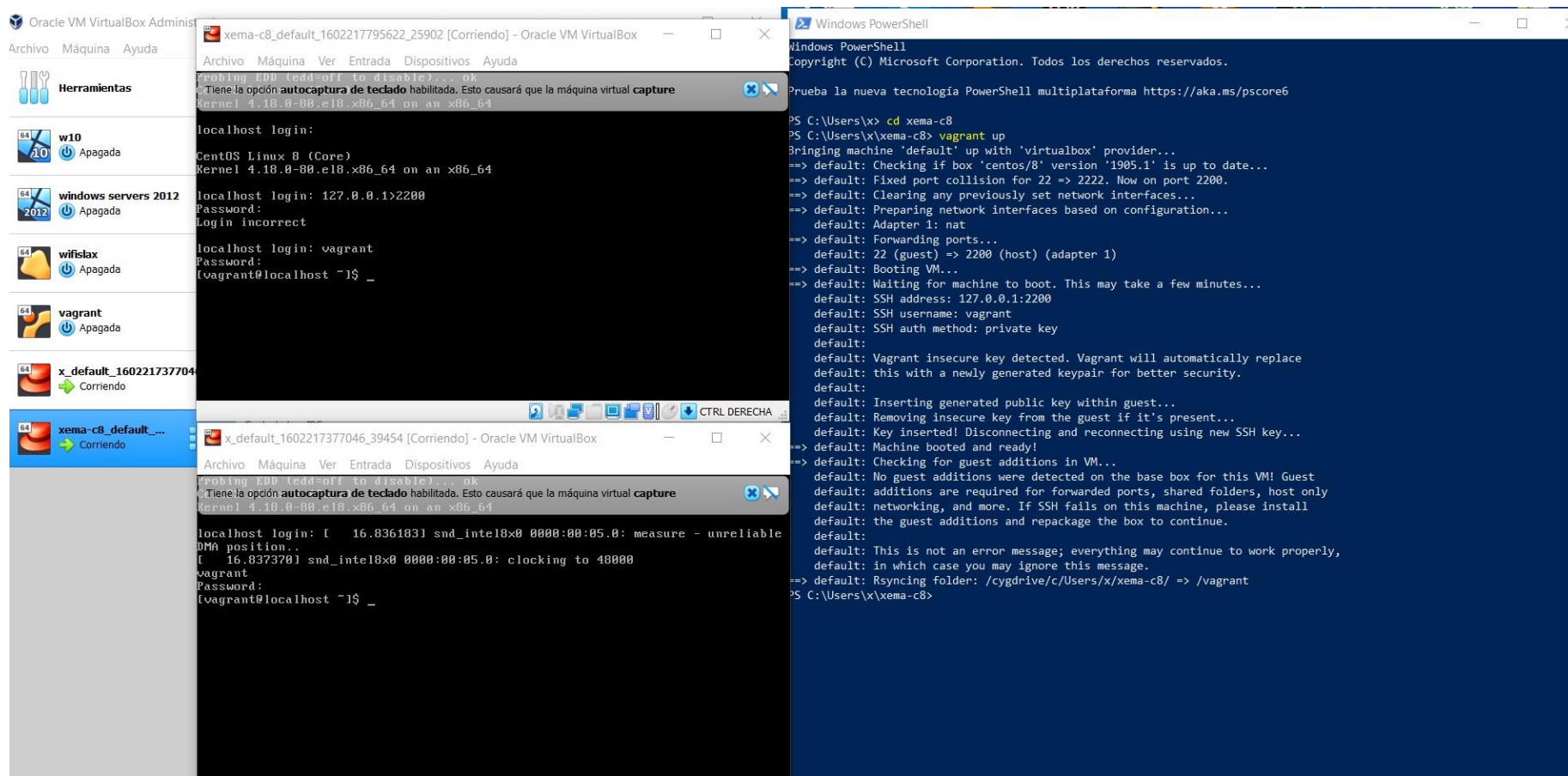
PS C:\Users\x> vagrant init centos/8
A 'Vagrantfile' has been placed in this directory. You are now
ready to 'vagrant up' your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
'vagrantup.com' for more information on using Vagrant.

PS C:\Users\x> vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Box 'centos/8' could not be found. Attempting to find and install...
default: Box Provider: virtualbox
default: Box Version: >= 0
==> default: Loading metadata for box 'centos/8'
default: URL: https://vagrantcloud.com/centos/8
==> default: Adding box 'centos/8' (v1905.1) for provider: virtualbox
default: Downloading: https://vagrantcloud.com/centos/boxes/8/versions/1905.1/providers/virtualbox.box
Download redirected to host: cloud.centos.org
default:
default: Calculating and comparing box checksum...
==> default: Successfully added box 'centos/8' (v1905.1) for 'virtualbox'!
==> default: Importing base box 'centos/8'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'centos/8' version '1905.1' is up to date...
==> default: Setting the name of the VM: x default 1602217377046_39454
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
default: Adapter 1: nat
==> default: Forwarding ports...
default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
default: SSH address: 127.0.0.1:2222
default: SSH username: vagrant
default: SSH auth method: private key
default:
default: Vagrant insecure key detected. Vagrant will automatically replace
default: this with a newly generated keypair for better security.
default:
default: Inserting generated public key within guest...
default: Removing insecure key from the guest if it's present...
default: Key inserted! Disconnecting and reconnecting using new SSH key...
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
default: No guest additions were detected on the base box for this VM! Guest
default: additions are required for forwarded ports, shared folders, host only
default: networking, and more. If SSH fails on this machine, please install
default: the guest additions and repackaging the box to continue.
default:
default: This is not an error message; everything may continue to work properly,
default: in which case you may ignore this message.
==> default: Rsyncing folder: /cygdrive/c/Users/x/ => /vagrant

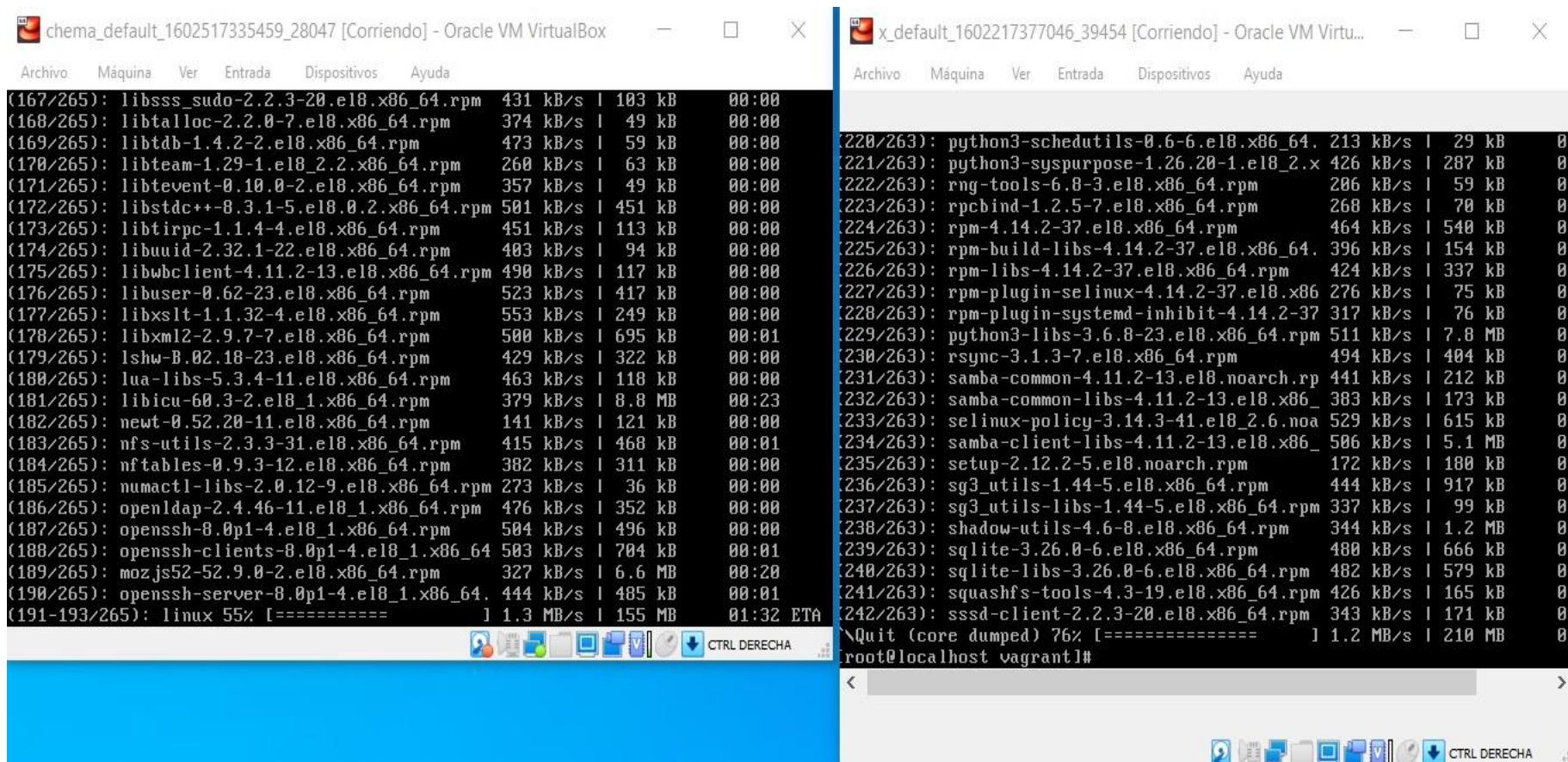
```

Instalar los servidores con vagrant en esta captura serian los 2 CentOS 8

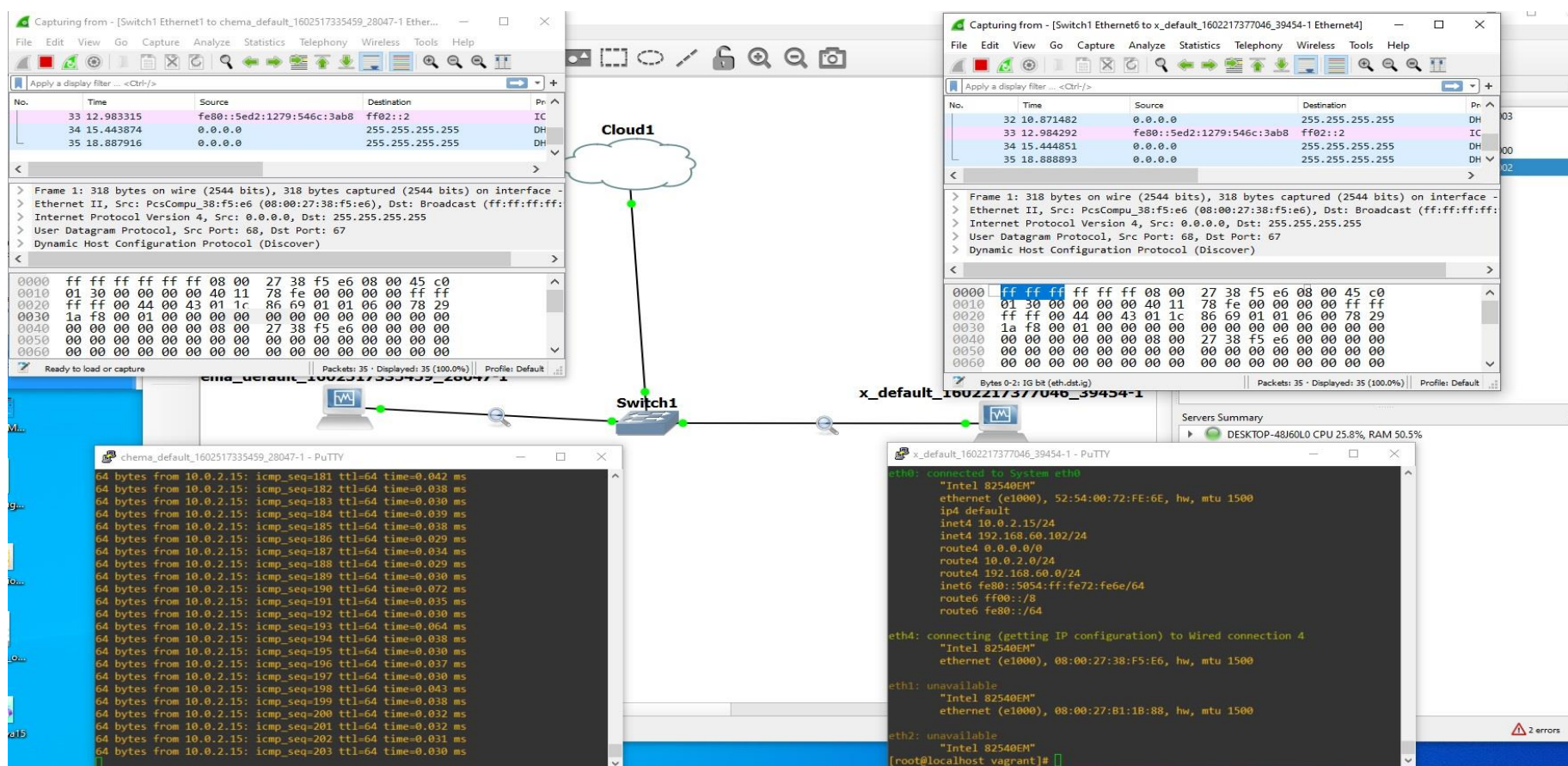
PÁGINA 1



Esta captura muestra a los servidores corriendo desde virtual box



Instalamos el Python 2 en los servidores de CentOS 8 y de igual manera configuramos las direcciones ip de cada servidor y creamos los archivos con vi para poner dentro los scripts de la conexión que trabajaremos en esta ocasión.



Instalamos el gns3 el puty y el wireshark en nuestra maquina para posteriormente poner nuestros servidores en conexi3n con un hub por medio del putty tendremos acceso a la consola y con el wireshark chequearemos el trafico de datos haremos un ping a nuestros servidores en este caso servidor es 192.168.0.16 y el cliente es 192.168.0.15 ya que vemos que el servidor funciona iniciamos el servicio del script del server por medio de Python 2.

1

No.	Time	Source	Destination	Protocol	Length	Info
622	2002.472938	fe80::5ed2:1279:546c:3ab8	ff02::1:3	LLMNR	85	Standard query 0x88ae ANY linux
623	2002.723863	fe80::5ed2:1279:546c:3ab8	ff02::1:3	LLMNR	85	Standard query 0x88ae ANY linux
624	2002.899603	fe80::5ed2:1279:546c:3ab8	ff02::1:6	ICMPv6	118	Multicast Listener Report Message v2
625	2002.996260	0.0.0.0	255.255.255.255	DHCP	318	DHCP Discover - Transaction ID 0x8858dbbe
626	2005.991099	fe80::5ed2:1279:546c:3ab8	ff02::2	ICMPv6	62	Router Solicitation
627	2007.976117	0.0.0.0	255.255.255.255	DHCP	318	DHCP Discover - Transaction ID 0xf291e150
628	2009.991958	fe80::5ed2:1279:546c:3ab8	ff02::2	ICMPv6	62	Router Solicitation
629	2016.122204	0.0.0.0	255.255.255.255	DHCP	318	DHCP Discover - Transaction ID 0xc034fc2c

2

```
> Frame 1: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface -, id 0
> Ethernet II, Src: PcsCompu_66:f1:86 (08:00:27:66:f1:86), Dst: IPv6mcast_02 (33:33:00:00:00:02)
> Internet Protocol Version 6, Src: fe80::5ed2:1279:546c:3ab8, Dst: ff02::2
> Internet Control Message Protocol v6
```

3

```
0000 33 33 00 00 02 08 00 27 66 f1 86 86 dd 60 06
0010 71 07 00 08 3a ff fe 80 00 00 00 00 00 00 5e d2
0020 12 79 54 6c 3a b8 ff 02 00 00 00 00 00 00 00
0030 00 00 00 00 02 85 00 7c c7 00 00 00 00 00 00
```

chema_cliente

```
connect: Network is unreachable
[root@localhost vagrant]# cat > cliente.py
import socket

target_host = "www.google.com"
target_port = 80

# create a socket object
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# connect the client
client.connect((target_host, target_port))

# send some data
client.send("GET / HTTP/1.1\r\nHost: google.com\r\n\r\n")

# receive some data
response = client.recv(4096)

print response[root@localhost vagrant]# touch cliente.py
[root@localhost vagrant]# nmtui
```

x_server

```
import socket
import threading

bind_ip = "192.168.0.15"
bind_port = 9999

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind((bind_ip, bind_port))
server.listen(5)

print "[*] Listening on %s:%d" % (bind_ip, bind_port)

# this is our client-handling thread
def handle_client(client_socket):

    # print out what the client sends
    request = client_socket.recv(1024)

    print "[*] Received: %s" % request

    # send back a packet
    client_socket.send("ACK!")
```

Corro primero el script sin modificar para ver cómo funciona, la sección 1 muestra todos los paquetes que se están capturando en tiempo real. La sección 2 desglosa por capas cada una de las cabeceras de los paquetes seleccionados en la zona 1. Y la sección 3 representa, en formato hexadecimal, el paquete en bruto, es decir, tal y como fue capturado por nuestra tarjeta de red.

En este caso a groso modo podemos ver paquetes de Linux y paquetes de datos del protocolo ipv6 de nuestros servidores.

The image displays a Wireshark packet capture window showing network traffic between a client (chema_cliente) and a server (x_server) via a hub (Hub1). The capture filter is set to 'Interface 0'. The packet list shows several ICMP Echo (ping) requests and responses, followed by a TCP connection attempt. The packet details pane shows the structure of the captured packets, including Ethernet II, Internet Protocol Version 4, and Internet Control Message Protocol. The packet bytes pane shows the raw data in hexadecimal and ASCII. Below the Wireshark window, a network diagram illustrates the topology: chema_cliente is connected to Hub1, which is connected to x_server. Two terminal windows are also shown: 'chema_cliente - PuTTY' and 'x_server - PuTTY'. The 'chema_cliente' terminal shows a successful ping to 192.168.0.16 and a successful connection to the server. The 'x_server' terminal shows the server listening on port 8080 and receiving a connection from 192.168.0.16. A 'Servers Summary' window on the right shows the system status: DESKTOP-4B160LD CPU 40.3%, RAM 47.2%.

Reporte de conclusiones:

Como podemos ver en los paquetes de datos enviados inicialmente regresan un tráfico de datos de la tarjeta Realtek luego enseguida muestra en **TRIPLEHAND** que inicia con el paquete **SYN** que inicia la comunicación en **SYN/ACK** que hace una especie de proceso y el **ACK** que regresa el **RECEIVED** que de igual manera muestra en los paquetes un **FIN/ACK** posterior mente en rojo vemos como se finaliza la conexión y se detienen los servicios de script de **Python** tomando en cuenta todos los datos obtenidos en el wireshark vimos como se suscitó el evento de envió de paquetes para la conexión de los scripts entre los servidores CentOS/8 es una dificultad muy grande de no seguir cada paso como se debe se puede suscitar incluso no lograr que funcione nada como debe funcionar.