



PROYECTO

Telecom

Descripción breve

Una practica donde veremos la comunicación entre 2 máquinas virtuales

Héctor Emilio Cantellano Gómez
Instituto tecnológico de Cancún

Introducción

En este proyecto podremos ver como es el proceso para que 2 maquinas virtuales se comuniquen entre si, se va utilizar varios programas para poder lograr estos y estos son: VirtualBox, GENS3, PuTTY y WireShark, además de que se podrá observar como son necesarios todos estos programas además de su compatibilidad.

Recursos:

-VirtualBox

-Vagrant

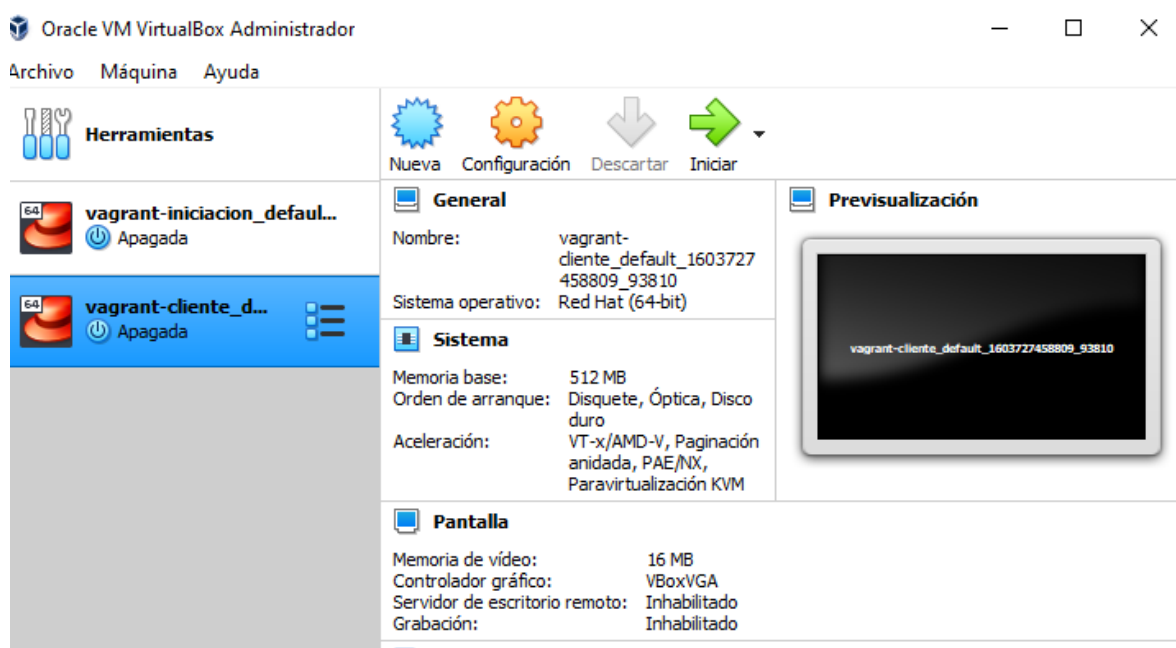
-GENS3

-PuTTY

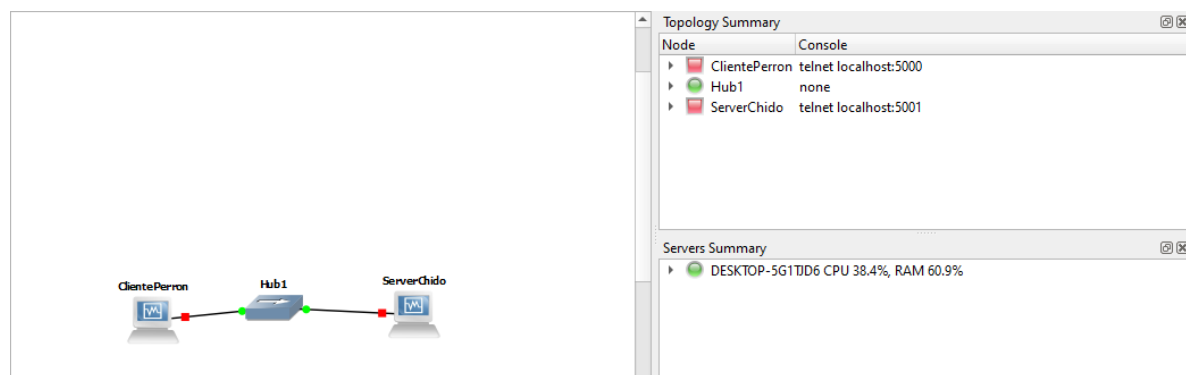
-WireShark

Proceso

Fase 1: Primero se instaló el SO de CentOS la versión 8 en 2 máquinas virtuales utilizando Vagrant y Powershell de Windows y las maquinas fueron de VirtualBox.



Fase 2: Des pues mediante GENS3 se conectaron las máquinas virtuales con un switch hub, pero antes se les asigno una ip a cada máquina y desde la VirtualBox configuramos que tendrían una conexión con cable



Fase 3: Ya al estar conectadas y puesto una ip a cada una de las maquinas, se le instalo a cada una Python para poder usar unos scripts y poder comunicarse entre sí, los comandos son algo complicados, pero con la practica se dominan enseguida.

```

ClientePerron - PuTTY
target_host = "192.168.60.102"
target_port = 3080

# 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

~
~
~
"tcpclientperron.py" [New] 19L, 354C written
[root@localhost vagrant]#

ServerChido - PuTTY
import thread

host = '192.168.60.102'
port = 3080

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print('Socket Ready...')

s.bind((host, port))
print('Bind Ready...')

print('Listening...')
s.listen(1)

def handle_client(client_socket):
    while True:
        data = client_socket.recv(1024)
        if not data: break
        print('Client says: ' + data)
        print('Sending: ' + data)
        client_socket.send(data)
        client_socket.close()

"tcpserverchido.py" [New] 30L, 635C written
[root@localhost vagrant]#
  
```

Nota: Para mayor comodidad y facilidad utilizamos PuTTY además de que con todo esto se conceto por vía Telnet.

```

ClientePerron - PuTTY
# 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

~
~
~
"tcpclientperron.py" 19L, 355C written
[root@localhost vagrant]# python2 tcpclientperron.py
bash: python2: command not found
[root@localhost vagrant]# python tcpclientperron.py
bash: python: command not found
[root@localhost vagrant]# python2 tcpclientperron.py
GET / HTTP/1.1
Host: google.com
[root@localhost vagrant]#

ServerChido - PuTTY
data = client_socket.recv(1024)
if not data: break
print('Client says: ' + data)
print('Sending: ' + data)
client_socket.send(data)
client_socket.close()

"tcpserverchido.py" [New] 30L, 635C written
[root@localhost vagrant]# python2 tcpserverchido.py
python2: can't open file 'tcpserverchido.py': [Errno 2] No such file or directory
[root@localhost vagrant]# python2 tcpserverchido.py
Socket Ready...
Bind Ready...
Listening...
Conexion from: ('192.168.60.101', 47214)
Client says: GET / HTTP/1.1
Host: google.com

Sending: GET / HTTP/1.1
Host: google.com
  
```

Fase 4: Por último, para poder ver el proceso que tienen estas 2 máquinas virtuales se usa WireShark para ver el tráfico de paqueterías que se envían

No.	Time	Source	Destination	Protocol	Length	Info
28	148.859809	fe80::5054:ff:fe72::ff02::2	ff02::2	ICMPv6	70	Router Solicitation from 52:54:00:72:fe:6e
29	183.162892	fe80::5054:ff:fe72::ff02::2	ff02::2	ICMPv6	70	Router Solicitation from 52:54:00:72:fe:6e
30	252.794308	fe80::5054:ff:fe72::ff02::2	ff02::2	ICMPv6	70	Router Solicitation from 52:54:00:72:fe:6e
31	379.772539	fe80::5054:ff:fe72::ff02::2	ff02::2	ICMPv6	70	Router Solicitation from 52:54:00:72:fe:6e
32	650.106071	fe80::5054:ff:fe72::ff02::2	ff02::2	ICMPv6	70	Router Solicitation from 52:54:00:72:fe:6e
33	677.167493	RealtekU_72:fe:6e	Broadcast	ARP	60	Who has 192.168.60.102? Tell 192.168.60.101
34	677.168464	RealtekU_72:fe:6e	RealtekU_72:fe:6e	ARP	60	192.168.60.102 is at 52:54:00:72:fe:6e
35	677.169441	192.168.60.101	192.168.60.102	TCP	74	44036 → 3080 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=3965459757 TSecr=0 WS=64
36	677.172373	192.168.60.102	192.168.60.101	TCP	74	3080 → 44036 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=3965459757 TSecr=3965459757 WS=64
37	677.174327	192.168.60.101	192.168.60.102	TCP	66	44036 → 3080 [ACK] Seq=1 Ack=1 Win=29248 Len=0 TSval=3965459764 TSecr=3469861394
38	677.174327	192.168.60.101	192.168.60.102	HTTP	102	GET / HTTP/1.1
39	677.174327	192.168.60.102	192.168.60.101	TCP	66	3080 → 44036 [ACK] Seq=1 Ack=37 Win=28992 Len=0 TSval=3469861396 TSecr=3965459764
40	677.176286	192.168.60.102	192.168.60.101	HTTP	102	GET / HTTP/1.1
41	677.177261	192.168.60.101	192.168.60.102	TCP	66	44036 → 3080 [ACK] Seq=37 Ack=37 Win=29248 Len=0 TSval=3965459767 TSecr=3469861398
42	677.184897	192.168.60.101	192.168.60.102	TCP	66	44036 → 3080 [FIN, ACK] Seq=37 Ack=37 Win=29248 Len=0 TSval=3965459772 TSecr=3469861398
43	677.186952	192.168.60.102	192.168.60.101	TCP	66	3080 → 44036 [FIN, ACK] Seq=37 Ack=38 Win=28992 Len=0 TSval=3469861407 TSecr=3965459772
44	677.188983	192.168.60.101	192.168.60.102	TCP	66	44036 → 3080 [ACK] Seq=38 Ack=38 Win=29248 Len=0 TSval=3965459779 TSecr=3469861407
45	682.361155	RealtekU_72:fe:6e	RealtekU_72:fe:6e	ARP	60	Who has 192.168.60.101? Tell 192.168.60.102
46	682.362133	RealtekU_72:fe:6e	RealtekU_72:fe:6e	ARP	60	192.168.60.101 is at 52:54:00:72:fe:6e

> Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0, id 0
 > Ethernet II, Src: RealtekU_72:fe:6e (52:54:00:72:fe:6e), Dst: RealtekU_72:fe:6e (52:54:00:72:fe:6e)
 > Internet Protocol Version 4, Src: 192.168.60.101, Dst: 192.168.60.102

0000 52 54 00 72 fe 6e 52 54 00 72 fe 6e 00 00 45 00 RT r-nrt r-n-r E
 0010 00 3c e3 57 40 00 40 06 5d 48 c0 a8 3c 65 c0 a8 <w@> [H] <e-
 0020 3c 66 58 70 8c 08 a5 c3 44 f1 00 00 00 00 02 <f.p.... D.....
 0030 72 10 0f 07 00 00 02 04 05 b4 04 02 00 0a c2 3a r.....:.....
 0040 da b4 00 00 00 00 01 03 03 06:.....

Conclusiones

Al concluir este proyecto se puede ver lo laborioso, pero no complicado proceso que el comunicarse entre dispositivos además de que se puede identificar como es el protocolo que se tiene que seguir, también te hace cuestionar de las posibilidades que puede tener todo esto, es impresionante ver como se usan variedad de programas para poder un mejor control y posibilidades para hacer las cosas, además que al principio se es difícil notar para que sirve cada cosa porque en si unas complementan a otras, también se pudo apreciar como por tener algo mal en una de estos programas puede afectar a todo lo demás o simplemente no funcione.