



udp UNIVERSIDAD
DIEGO PORTALES

Facultad de Ingeniería y Ciencias
Escuela de Informática y Telecomunicaciones

Laboratorio N3: Redes de Datos
“ Creación de paquetes utilizando Scapy y validación con Wireshark ”

Profesor: José Pérez
Ayudante: Alexis Inzunza
Fecha: 12-05-2017

Integrantes:
Benjamín Alvarez
Nicolás Reyes

ÍNDICE

I.	Actividad 1: Instalación de Software(Scapy)	3
II.	Actividad 2: Creación de paquete	4
III.	Cuestionario e Invesstigación	5
IV.	Conclusión	6
	Referencias	7

ÍNDICE DE FIGURAS

1.	Comando para instalación de Scapy	3
2.	Comando para ejecutar Wireshark	3
3.	Comandos de ls()	4
4.	Comando de Ether()	4
5.	Comandos de IP()	4
6.	Comandos de ICMP()	5
7.	Comandos de Raw() y send() , sendp()	5

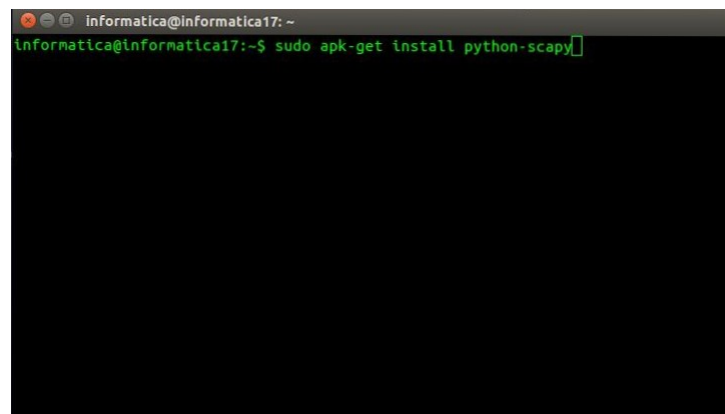
Actividades de Laboratorio

En este laboratorio se busca aprender a crear paquetes personalizados para corroborar el funcionamiento de algún servicio o dispositivo, para esto se utilizará Scapy, un programa hecho en python que permite enviar cualquier tipo de paquete y enviarlo por la red.

I. ACTIVIDAD 1: INSTALACIÓN DE SOFTWARE(SCAPY)

Esta actividad solo se realiza si es que el computador a utilizar no tiene el software necesario para realizar la actividad. Para hacer esto debemos haber encendido nuestro computador y tener la sesión iniciada.

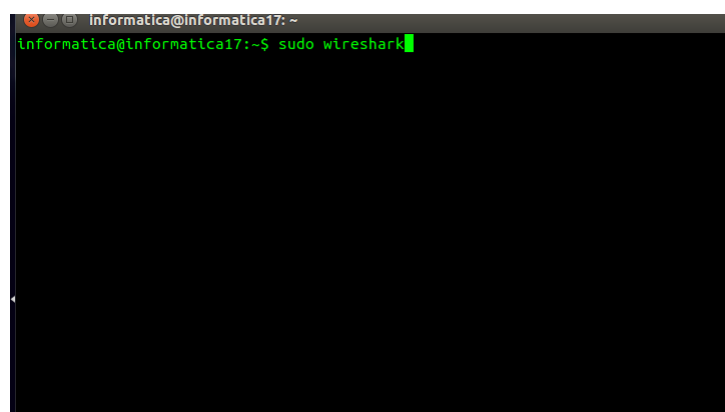
Luego se abre el terminal y se debe escribir "sudo apt-get install python-scapy" para empezar con la instalación del software.



```
informatica@informatica17: ~  
informatica@informatica17:~$ sudo apt-get install python-scapy
```

Figura 1. Comando para instalación de Scapy

Una vez finalizado este proceso (para saber si la instalación se realizó correctamente se debe revisar aptitude) se debe ejecutar Wireshark utilizando el comando "sudo wireshark"



```
informatica@informatica17: ~  
informatica@informatica17:~$ sudo wireshark
```

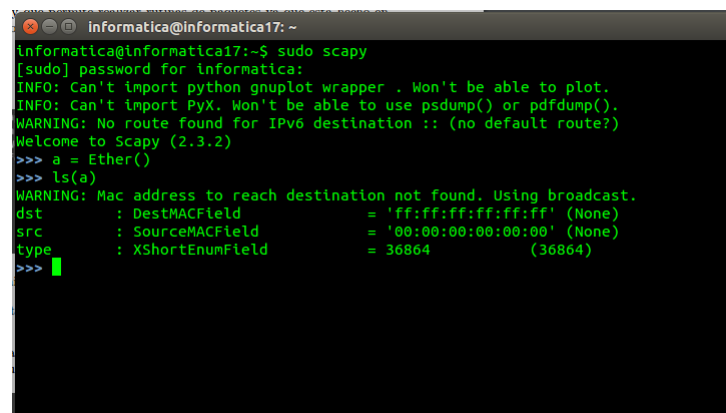
Figura 2. Comando para ejecutar Wireshark

II. ACTIVIDAD 2: CREACIÓN DE PAQUETE

Luego de tener Scapy instalado se procede a crear el paquete que será enviado. Acá se dieron dos opciones para la creación, la primera es usar Scapy desde el terminal usando el comando “sudo scapy” y la segunda es programar un script (si se usa esta opción no se debe olvidar el escribir “from scapy.all import *” para importar Scapy a nuestro programa), en ambos casos con python.

En cualquiera de los dos casos se nos fueron dados varios comandos que nos ayudaban a crear el paquete y revisar datos, estos eran los siguientes:

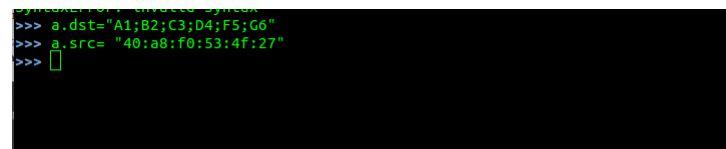
- ls(pkt): Sirve para mostrar todos los datos de un paquete.



```
informatica@informatica17: ~
informatica@informatica17:~$ sudo scapy
[sudo] password for informatica:
INFO: Can't import python gnuplot wrapper . Won't be able to plot.
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().
WARNING: No route found for IPv6 destination :: (no default route?)
Welcome to Scapy (2.3.2)
>>> a = Ether()
>>> ls(a)
WARNING: Mac address to reach destination not found. Using broadcast.
dst      : DestMACField      = 'ff:ff:ff:ff:ff:ff' (None)
src      : SourceMACField    = '00:00:00:00:00:00' (None)
type     : XShortEnumField   = 36864      (36864)
>>>
```

Figura 3. Comandos de ls()

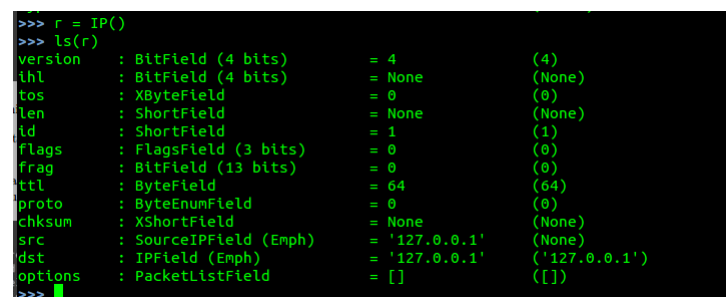
- Ether(): Basado en la capa 2 de OSI. Se le puede asignar una variable para luego revisar todos sus campos con ls(). También, una vez asignada la clase a la variable, se puede cambiar con valores válidos al antojo del usuario utilizando las instrucciones .dst y .src como se ve en la figura.



```
>>> a.dst="A1:B2:C3:D4:F5:G6"
>>> a.src= "40:a8:f0:53:4f:27"
>>>
```

Figura 4. Comando de Ether()

- IP(): Basado en la capa 3 de OSI. También se le puede asignar una variable y revisar sus datos con ls().



```
>>> r = IP()
>>> ls(r)
version  : BitField (4 bits)      = 4      (4)
ihl      : BitField (4 bits)      = None    (None)
tos      : XByteField             = 0      (0)
len      : ShortField             = None    (None)
id       : ShortField             = 1      (1)
flags    : FlagsField (3 bits)    = 0      (0)
frag     : BitField (13 bits)     = 0      (0)
ttl      : ByteField              = 64     (64)
proto    : ByteEnumField          = 0      (0)
chksum   : XShortField            = None    (None)
src      : SourceIPField (Emph)    = '127.0.0.1' (None)
dst      : IPField (Emph)         = '127.0.0.1' ('127.0.0.1')
options  : PacketListField        = []      ([])
>>>
```

Figura 5. Comandos de IP()

- ICMP(): Se basa en la capa 4 de OSI, se le puede asignar una variable y revisar sus campos con ls().

```

>>> t=ICMP()
>>> ls(t)
type      : ByteEnumField      = 8          (8)
code      : MultiEnumField     = 0          (0)
chksum    : XShortField        = None       (None)
id        : XShortField (Cond) = 0          (0)
seq       : XShortField (Cond) = 0          (0)
ts_ori    : ICMPTimeStampField (Cond) = 67190077 (67190077)
ts_rx     : ICMPTimeStampField (Cond) = 67190077 (67190077)
ts_tx     : ICMPTimeStampField (Cond) = 67190077 (67190077)
gw        : IPField (Cond)     = '0.0.0.0' ('0.0.0.0')
ptr       : ByteField (Cond)   = 0          (0)
reserved  : ByteField (Cond)   = 0          (0)
length    : ByteField (Cond)   = 0          (0)
addr_mask : IPField (Cond)     = '0.0.0.0' ('0.0.0.0')
nexthopmtu : ShortField (Cond) = 0          (0)
unused    : ShortField (Cond)   = 0          (0)
unused    : IntField (Cond)     = 0          (0)
>>>

```

Figura 6. Comandos de ICMP()

- Raw(): Es un payload o carga útil para nuestro modelo OSI, se le puede asignar una variable y luego revisar todos los campos con ls().

- send(pkt) o sendp(pkt): Son los encargados de enviar nuestro paquete, pero aunque parezcan iguales no lo son, ya que ambos trabajan en capas distintas, en este caso se usará sendp que trabaja en la capa 2 de OSI, lo que permite ingresar la interfaz correcta y todos los parámetros de esta capa de forma manual.

```

nextthopmtu : ShortField (Cond) = 0          (0)
unused      : ShortField (Cond) = 0          (0)
unused      : IntField (Cond)   = 0          (0)
>>> p=Raw()
>>> ls(p)
load      : StrField          = ''        ('')
>>> packete = a/r/t/p
>>> sendp(packete)

```

Figura 7. Comandos de Raw() y send() , sendp()

Luego de conocer todos estos comandos se procede a formar el paquete que se va a enviar, para esto usaremos una variable donde se deben apilar los paquetes en el mismo orden según la pila OSI basándonos en el modelo anterior capa 1, capa 2, ... , capa n como se ve en la forma REVISAR. Como último paso se debe probar enviando paquetes a 3 casos distintos:

1. A una dirección MAC=FF:FF:FF:FF:FF:FF
2. A una dirección MAC de otro computador del laboratorio.
3. A una dirección MAC que no se encuentra en la misma red.

III. CUESTIONARIO E INVESTIGACIÓN

1. ¿Qué sucede cuando se envía un paquete a la dirección FF:FF:FF:FF:FF:FF? ¿Quiénes lo reciben? ¿Por qué?
R: FF:FF:FF:FF:FF:FF que es la dirección de difusión y las direcciones de cada adaptador de red en la red, así que al enviar un paquete con esta dirección se debería mandar este a todos los dispositivos conectados a la red de donde proviene el paquete.
2. ¿Qué pasa cuando se envía un paquete a la MAC de otro equipo? ¿Quiénes lo pueden recibir? ¿Por qué?
R: Le llega un paquete al computador con la MAC asignada, lo recibe solo el o los equipos con esa MAC. Esto es porque el mensaje va dirigido solo a los que tengan la MAC de destino.
3. ¿Qué sucede si se envía un paquete a una MAC que no corresponda a ningún equipo de la red? ¿Quiénes lo pueden recepcionar? ¿Por qué?
R: Al no existir esta MAC, simplemente el paquete de datos no llega a ningún y en el caso de ser así lo más probable es que solo llegue una alerta de que este no fue recibido.

IV. CONCLUSIÓN

En este experimento aprendimos como usar lo básico de Scapy y a crear y enviar paquetes en distintas situaciones, donde nos dimos cuenta como funcionan en distintos casos. Así como en el caso de usar una la direccion MAC "FF:FF:FF:FF:FF:FF" un direccion MAC ubicada en la red, donde, aunque ambas se enviaban no tenian el mismo resultado. O como al usar una MAC que no se encuentra en la red el envio de esta simplemente falla y no le llega a nadie. Tambien pudimos conocer en detalle los datos que enviará nuestro computador a otro a través de los comandos de Scapy.

REFERENCIAS

- [1] Wireshark, *Página web Wireshark* <https://wireshark.com>