Desarrolla tu propio dispositivo BadUSB con WiFi y almacenamiento

MUNDO HACKER ACADEMY

DISCLAIMER

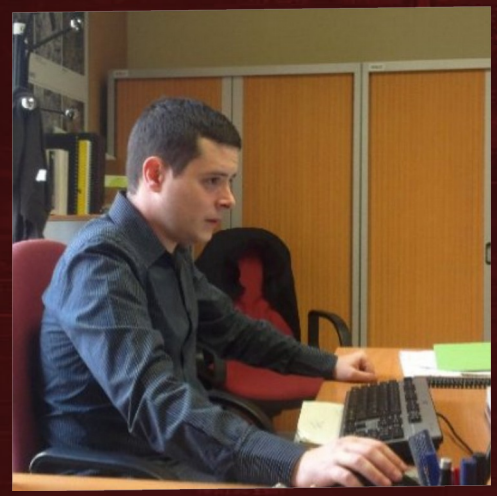

PARENTAL
ADVISORY
EXPLICIT CONTENT


I'm not a
Hacker
I'm a
"SECURITY
PROFESSIONAL"
Image courtesy: Mikhail Romanenko


PARENTAL
ADVISORY
EXPLICIT CONTENT

Sobre nosotros

Joel Serna Moreno
@JoelSernaMoreno



Ernesto Sánchez Pano
@ernesto_xload

Interesados en ciberseguridad, radio, DIY...

RESUMEN

Parte I: Introducción
- ¿BadUSB? Ataques y dispositivos
- Dispositivos comerciales con WiFi
- Firmwares

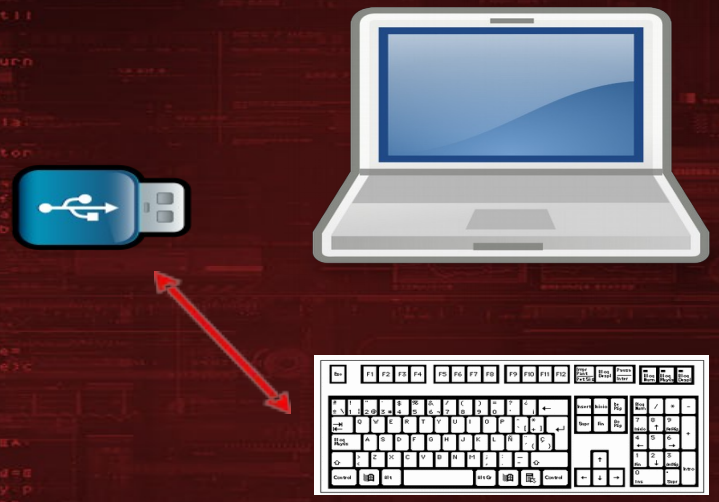
Parte II: Hardware
- Hardware utilizado
- Anatomía
- Modos de programación

Parte III: Con las manos en la masa
- Requisitos
- ¿WSD?
- Pasos a seguir
- Jugando con WSD

Parte IV: ¿Y ahora qué? / Agradecimientos

Parte I: Introducción

¿BadUSB?



ATAQUES

HID
(Human Interface Device)

- Identificación como teclado o ratón
- Inyección de teclas
- Serial

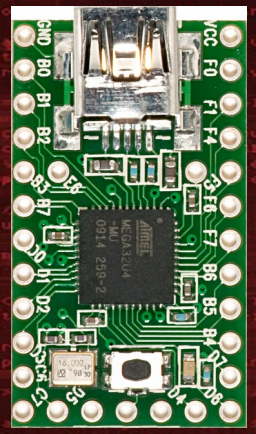
```
[  727.652188] usb 2-1.2: New USB device found, idVendor=2341, idProduct=8036
[  727.652190] usb 2-1.2: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[  727.652192] usb 2-1.2: Product: Arduino Leonardo
[  727.652193] usb 2-1.2: Manufacturer: Arduino LLC
[  727.652194] usb 2-1.2: SerialNumber: HIDPC
[  727.652524] cdc_acm 2-1.2:1.0: ttyACM3: USB ACM device
[  727.765971] hidraw: raw HID events driver (C) Jiri Kosina
[  727.775609] usbcore: registered new interface driver usbhid
[  727.775610] usbhid: USB HID core driver
[  727.895261] input: Arduino LLC Arduino Leonardo as /devices/pci0000:00/0000:00:1d.0/usb2/2-1/2-1.2/2-1.2:1.2/0003:2341:8036.0001/input/input16
[  727.952169] hid-generic 0003:2341:8036.0001: input,hidraw0: USB HID v1.01 Keyboard [Arduino LLC Arduino Leonardo] on usb-0000:00:1d.0-1.2/input2
```

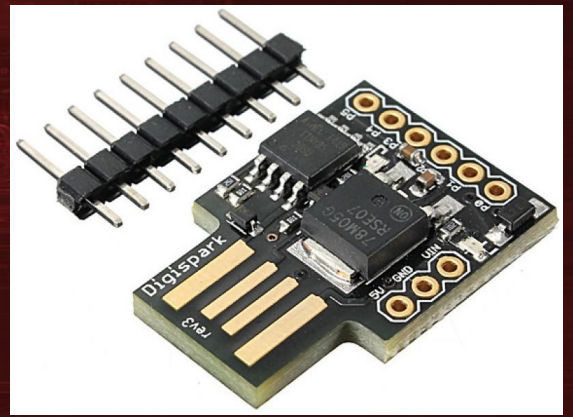
Desventajas:
- Ataques dirigidos

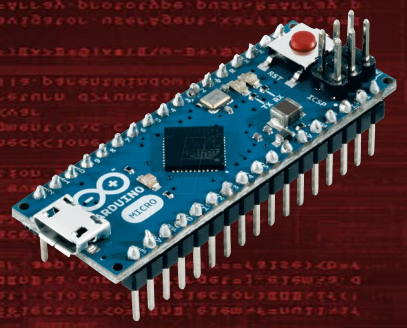
ADB
(Android Debug Bridge)

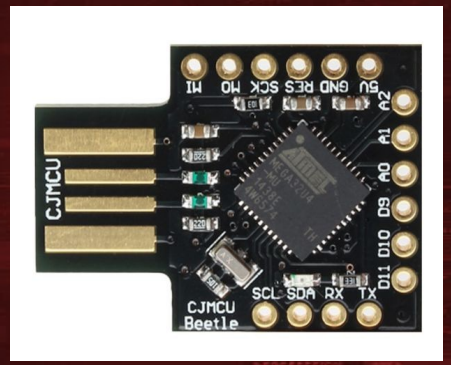
- Ejecución de comandos ADB en Android
- Serial y USB Host Shield
- No es necesario OTG
- Ataques “menos” dirigidos

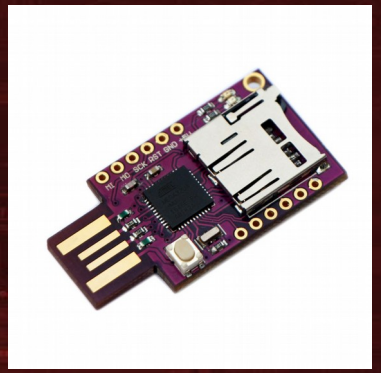
Desventajas:
- Modo depuración USB activado




Digispark




CJMCU
CJMCU
Beetle



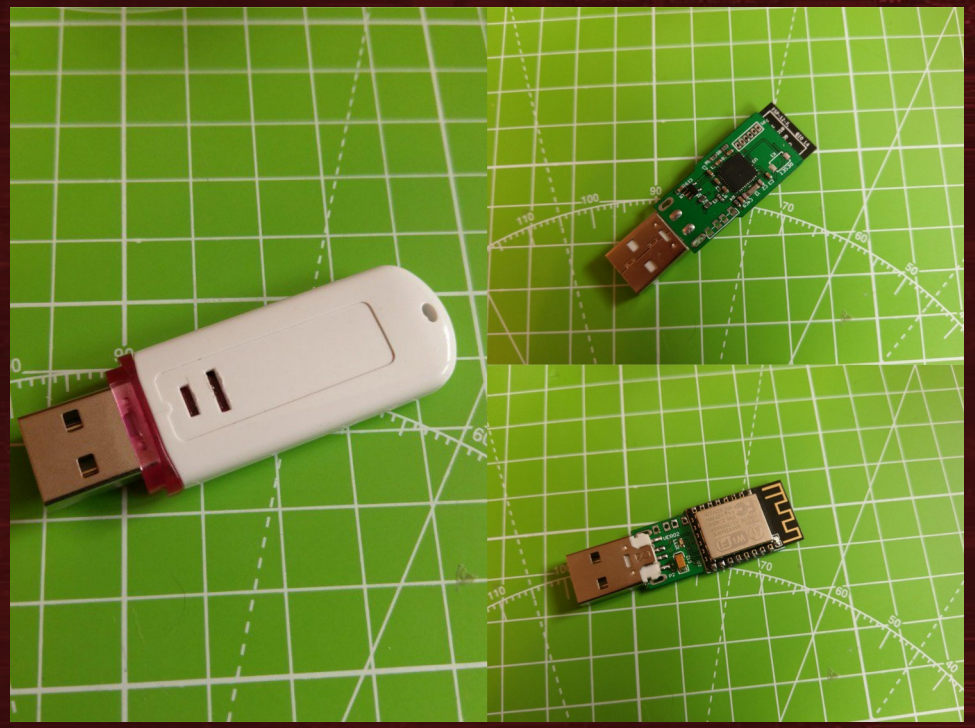

Adrian Crenshaw's - Defcon 2010
Samy Kamkar - 2014

Dispositivos comerciales I: WHID Injector (Luca Bongiorni)

Hardware:
- Atmega32u4
- ESP8266
- Sensor Hall

Funcionalidades:
- BadUSB
- Tecnologías inalámbricas (WiFi)

Donde comprar:
- Aliexpress
- Tindie
- AprilBrother



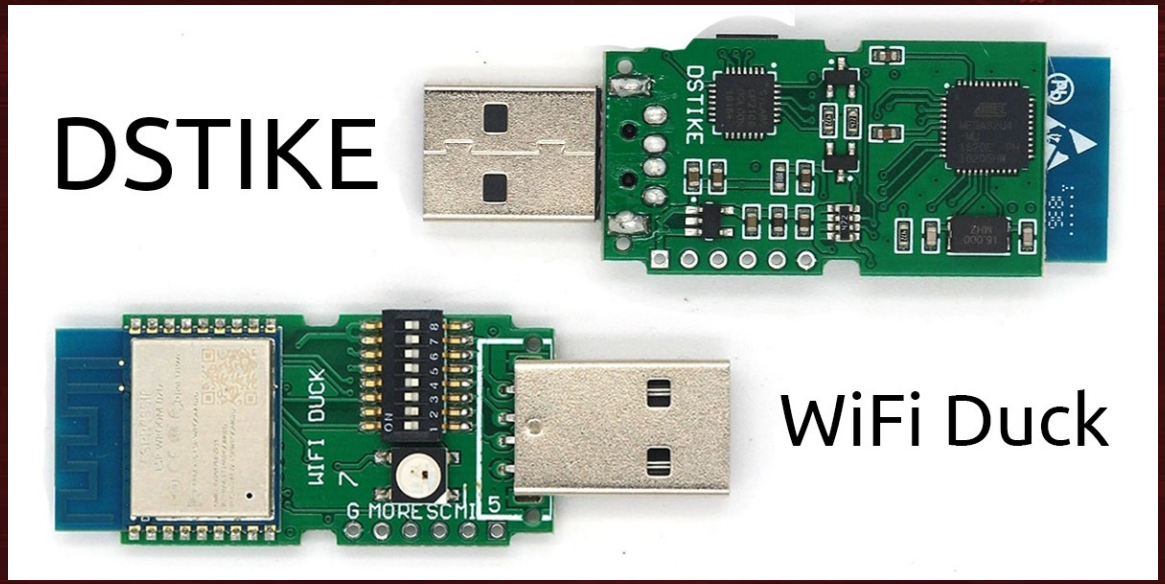
Precio: 16$ + envío

Dispositivos comerciales III: DSTIKE WiFi Duck (Stefan Kremser)

Hardware:
- Atmega32u4
- ESP8266
- 8bit DIP-switch

Funcionalidades:
- BadUSB
- Tecnologías inalámbricas (WiFi)
- Multi opciones (switch)

Donde comprar:
- Aliexpress
- Tindie
- DSTIKE

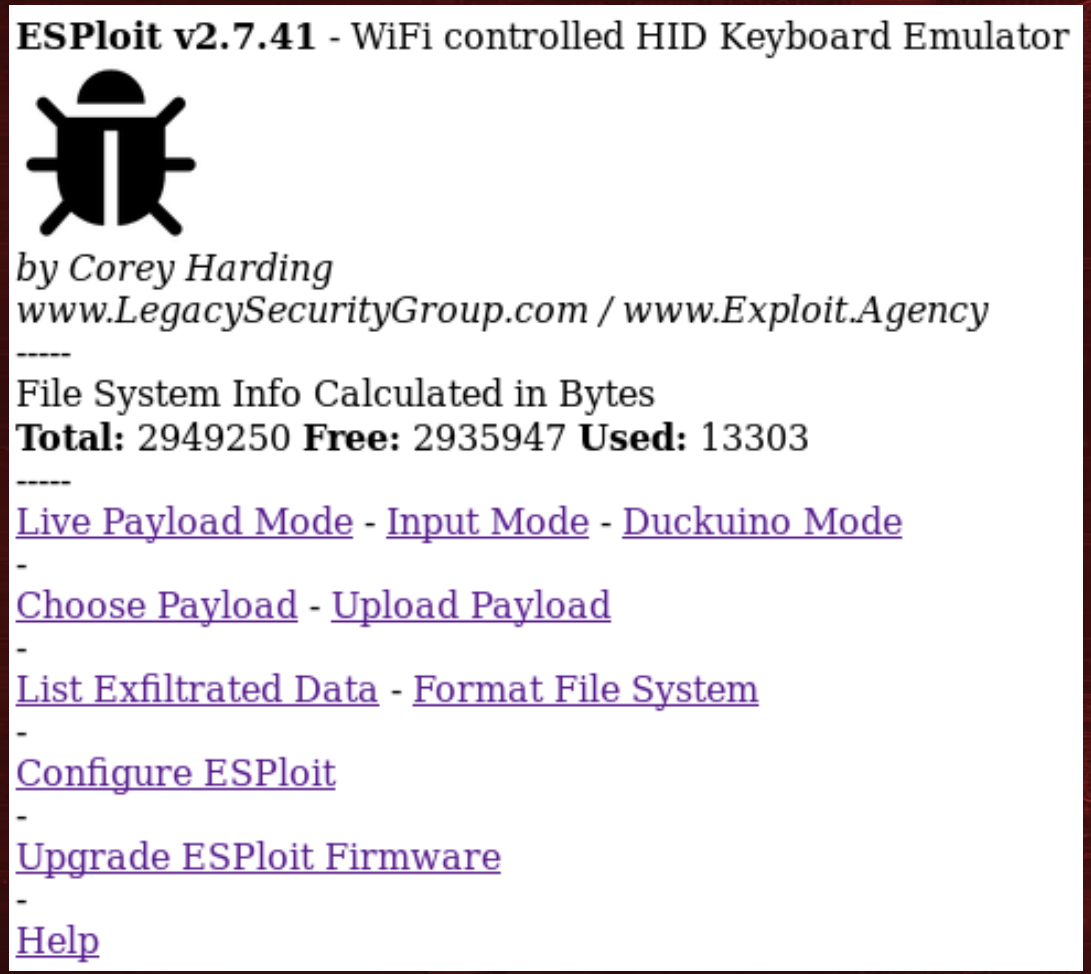


Precio: 27$ + envío

Firmware I: ESPloit V2 (Corey Harding)

Funcionalidades:
- Live Payload
- Upload Payload
- Teclado virtual
- Ducknuino
- Exfiltrated Data
- FTP server
- Changing the VID/PID

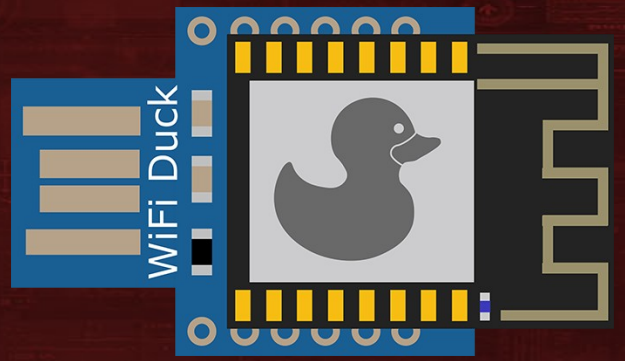
https://github.com/exploitagency/ESPloitV2



Firmware II: WiFi Duck (Stefan Kremser)

Funcionalidades base:

- Soporte para atmega32u4 y Digispark (Attiny85)
- Cambiar SSID y contraseña desde web
- Mouse
- Autoejecución de payloads
- Sintaxis Rubber Ducky



Funcionalidades añadidas:

- Soporte Multi Layout
- Payloads no limitados
- Serial sustituido por i2c

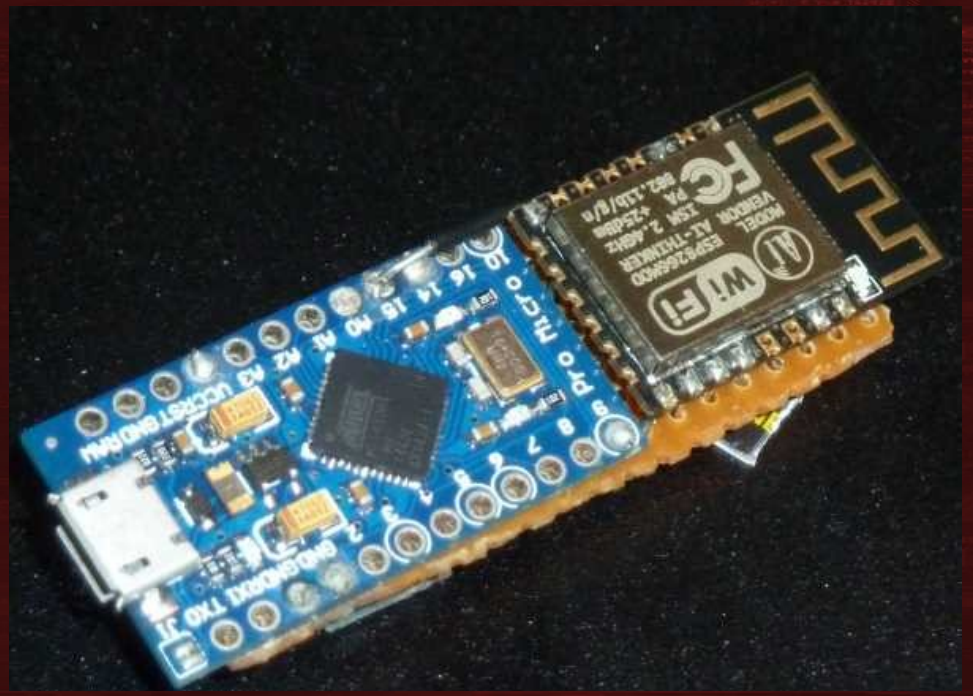
V1:
https://github.com/spacehuhn/wifi_ducky/

V2:
https://github.com/spacehuhn/WiFiDuck

Firmware III: WiDucky (Basic4)

Funcionalidades:

- Live Payload
- Python Server
- Windows Server
- Android Server
- Sintaxis Rubber Ducky



https://github.com/basic4/WiDucky

Firmware IV: WSD (Joel Serna)

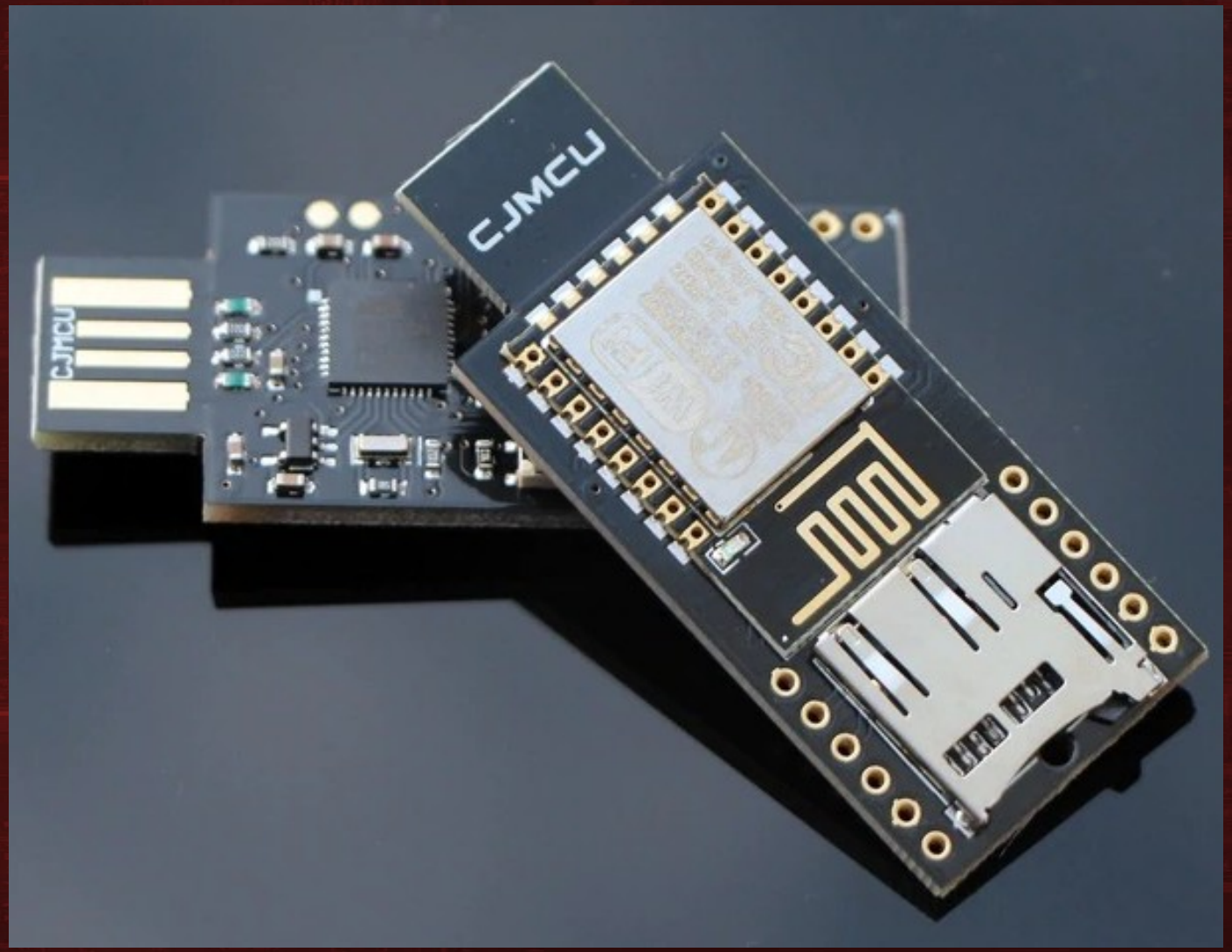
Funcionalidades base:
- Teclado virtual
- Teclado físico (@Santpapen)
- Live Payload
- Autoejecución de payloads
- Upload Payload
- Multi payloads y almacenamiento (SD)

Funcionalidades añadidas:
- Nuevo diseño (@pixeltoothless)
- Detección SO
- Exfiltración de datos vía serial

V1:
https://github.com/joelsernamoreno/badusb_sd_wifi

V2:
https://github.com/joelsernamoreno/WSD

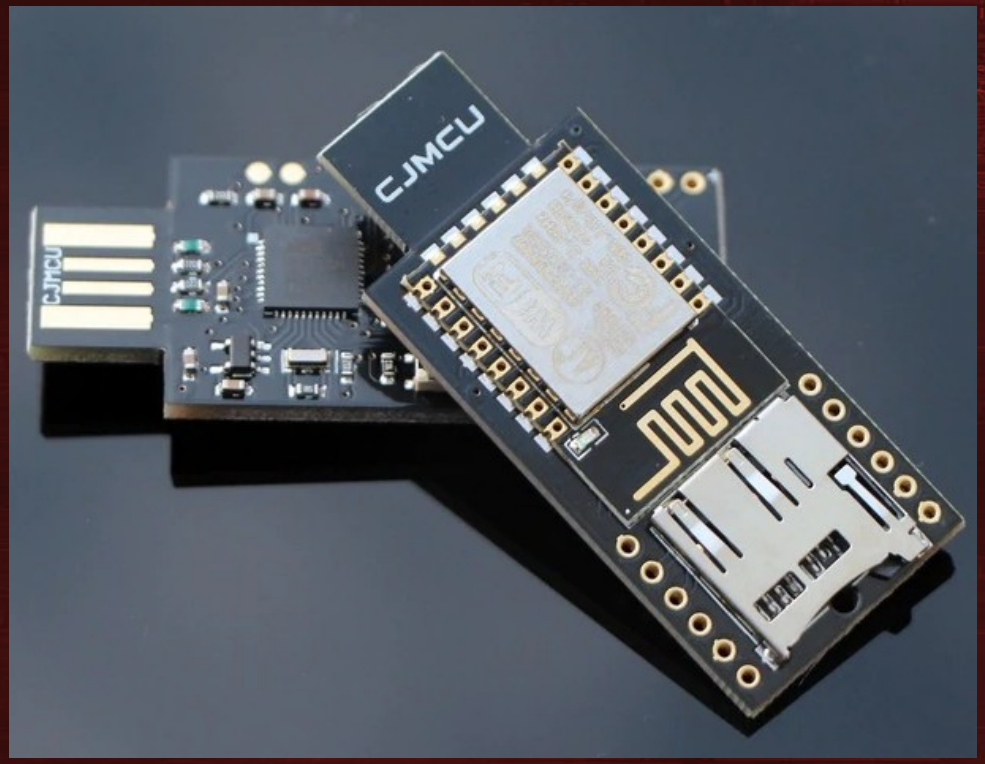
Parte II: Hardware



CJMCU 3212

Hardware:
- Atmega32u4
- ESP8266
- Slot MicroSD

Ventajas:
- Almacenamiento



Funcionalidades:
- BadUSB
- Tecnologías inalámbricas (WiFi)
- Almacenamiento payloads (SD)

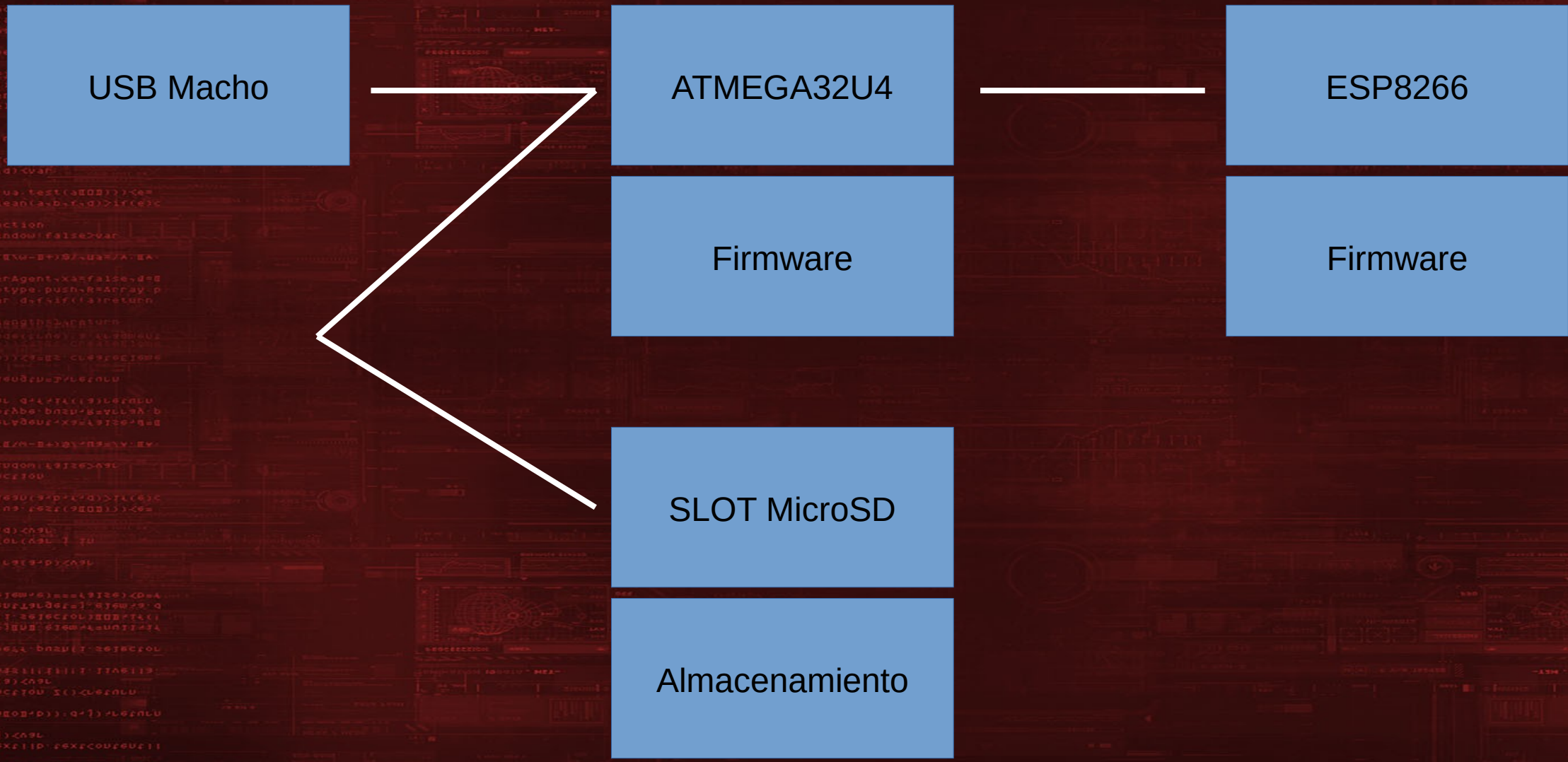
Desventajas:
- Documentación nula
- Modo programador esp8266

Donde comprar:
- Aliexpress

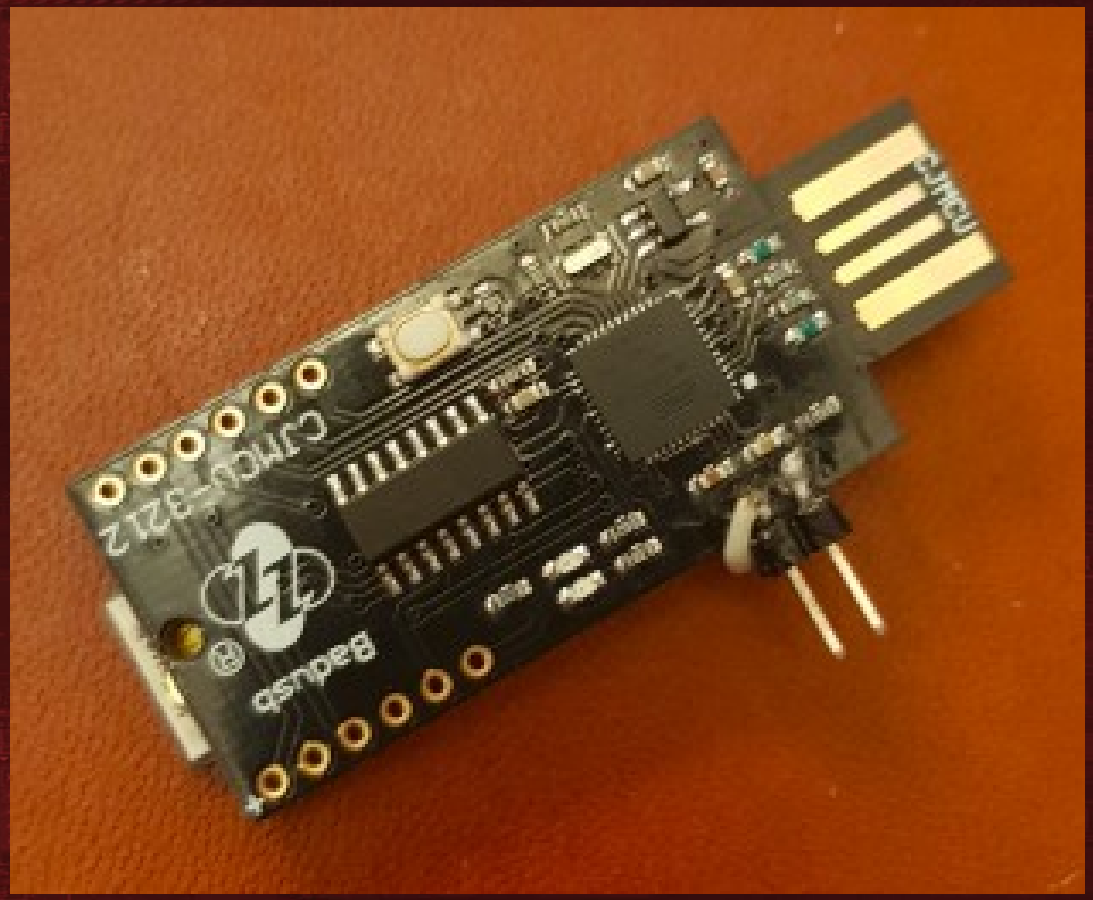
Precio: 10$ + envío

Anatomía I: WiFi

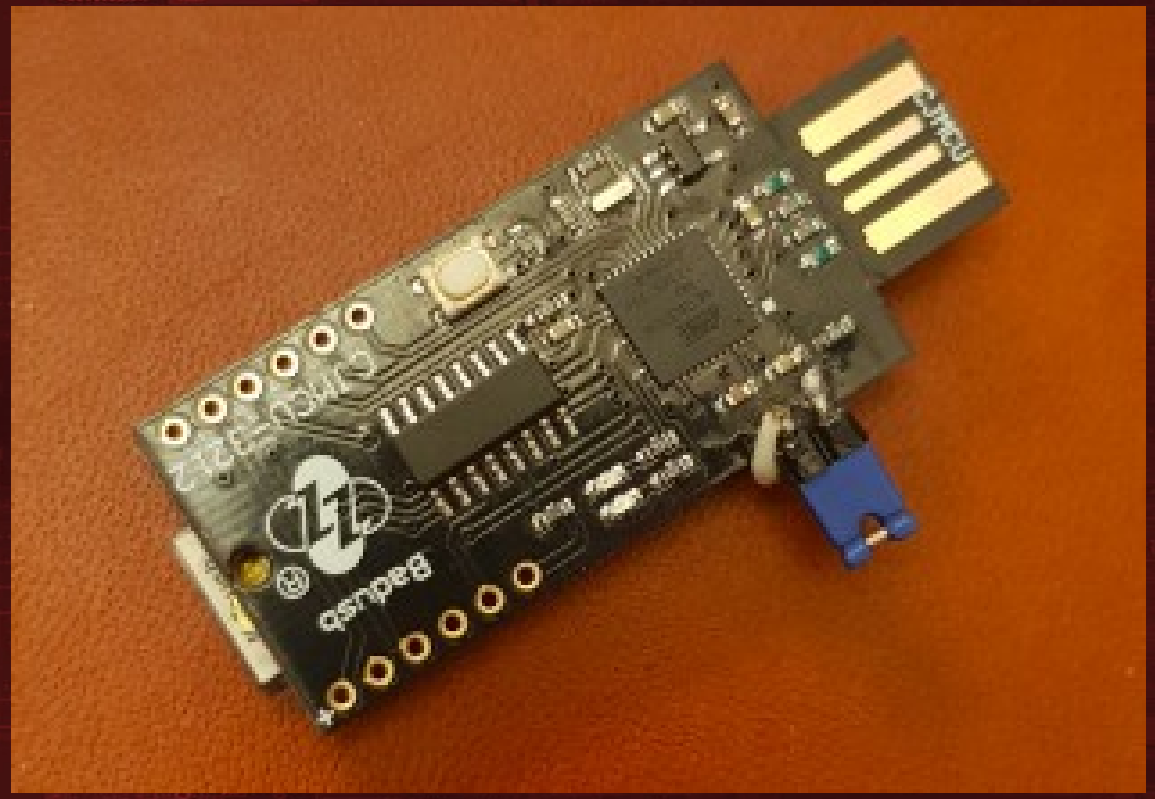
Anatomía II: WiFi + SD



Modo I: Programar Atmega32u4



Modo II: Programar ESP8266



Parte III: Con las manos en la masa

Requisitos I: Windows y Linux

- Comprar CJMCU 3212

- Descargar IDE Arduino: https://www.arduino.cc/en/main/software

- Añadir en la sección Additional Board Manager URLs del IDE de Arduino: http://arduino.esp8266.com/stable/package_esp8266com_index.json

- Instalar esp8266 by ESP8266 community version 2.3.0 en el IDE de Arduino

- Descargar FingerPrintUSBHost en el directorio de bibliotecas de Arduino: https://github.com/keyboardio/FingerprintUSBHost

- Descargar biblioteca Keyboard en el directorio de bibliotecas de Arduino: https://github.com/ernesto-xload/arduino_keyboardlib

- Descargar el código WSD: https://github.com/joelsernamoreno/WSD

Requisitos II: Linux

- Descargar esptool (Linux): https://github.com/AprilBrother/esptool

- Instalar python-serial (Linux): sudo apt install python-serial

- Añadir usuario al grupo dialout (Linux): sudo usermod -a -G dialout test

NOTA: Se recomienda no instalar el IDE de Arduino, sólo ejecutar Arduino con el siguiente comando: ./arduino

Requisitos III: Windows

- Instalar IDE de Arduino

- Descargar NodeMCU (Windows): https://github.com/nodemcu/nodemcu-flasher

Sketches y tools incluidas en WSD

Sketch:

- ESP8266Programmer.ino: código para Atmega32u4
- ESP8266_Code.ino: código para ESP8266
- ESP8266_Code.ino.generic.bin: binario para flashear en ESP8266
- Atmega32u4_Code: código final para Atmega32u4

Tools:

- esptool.py: programa para flashear ESP8266 en Linux
- nodemcu.exe: programa para flashear ESP8266 en Windows

Bibliotecas:

- Keyboard con soporte multi layout
- FingerprintUSBHost: Identificación del SO

Pasos a seguir I

1. Descargar y realizar los requisitos mostrados en la diapositiva “Requisitos I: Windows y Linux“

2. Si tu PC es Linux, descargar y realizar los requisitos mostrados en la diapositiva: “Requisitos II: Linux”

3. Si tu PC es Linux, descargar y realizar los requisitos mostrados en la diapositiva: “Requisitos III: Windows”

4. Abrir el sketch ESP8266Programmer.ino del repositorio WSD en el IDE

5. Conectar al PC el dispositivo CJMCU 3212 sin el jumper

6. Configurar el IDE de Arduino de la siguiente forma:
   - Herramientas → Placa: Arduino Leonardo
   - Herramientas → Puerto: ttyACMX (Linux), COMX (Windows)

Pasos a seguir II

7. Verificar código: click en el bóton verificar código del IDE

8. Si no obtienes ningún error, subir el código a la placa: click en el botón subir en el IDE

9. Después de terminar de subir el código, desconectar CJMCU del PC

10. Abrir el sketch ESP8266_Code.ino del repositorio WSD en el IDE

11. Configurar el IDE de Arduino de la siguiente forma:
    - Herramientas → Placa: Generic ESP8266 Module
    - Herramientas → Upload Speed: 115200
    - Herramientas → CPU Frequenzy: 80Mhz
    - Herramientas → Flash Size: 4M (3M SPIFFS)
    - Herramientas → Flash Mode: DIO
    - Herramientas → Flash Frequency: 40Mhz
    - Herramientas → Reset Methoz: ck
    - Herramientas → Debug port: disabled
    - Herramientas → Debug Level: ninguno
    - Herramientas → Puerto: vacío

12. Verificar código: click en botón verificar

Pasos a seguir III

13. En el IDE: click en Programa → Exportar binarios compilados

14. Entrar desde terminal en el directorio esptool (Linux) o ejecutar el programa NodeMCU (Windows)

15. Conectar el dispositivo CJMCU 3212 al PC con el jumper (modo programación ESP8266)

16. Subir el binario ESP8266_Code.ino.generic.bin a la placa:

- En Linux ejecutar el siguiente comando: sudo python esptool.py --port=/dev/ttyACMX --baud 115200 write_flash 0x00000 /PATH/WSD/ESP8266_Code/ESP8266_Code.ino.generic.bin --flash_size 32m

- En Windows: Configurar el programa NodeMCU de la siguiente manera:
  Seleccionar ESP8266_Code.generic.bin
  Baud Rate: 1152000
  Flash

Pasos a seguir IV

17. Desconectar el dispositivo CJMCU 3212 y abrir el sketch Atmega32u4_Code.ino

18. Conectar el dispositivo CJMCU 3212 sin el jumper

19. Configurar el IDE de Arduino de la siguiente forma:
    - Herramientas → Placa: Arduino Leonardo
    - Herramientas → Puerto: ttyACMX (Linux), COMX (Windows)

20. Verificar código: click en el bóton verificar código del IDE

21. Si no obtienes ningún error, subir el código a la placa: click en el botón subir en el IDE

22. Visualizar las redes WiFi y acceder al punto de acceso WSD con la contraseña que indica el README del repositorio WSD

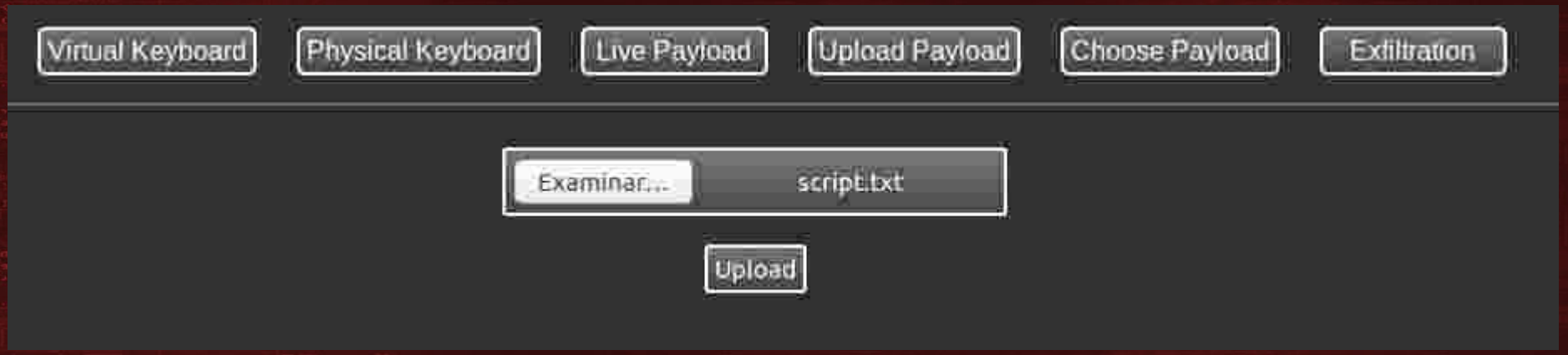
23. Ejecutar un navegador y acceder a la IP: 192.168.1.1

24. A jugar!!!

Jugando con WSD I: Virtual Keyboard

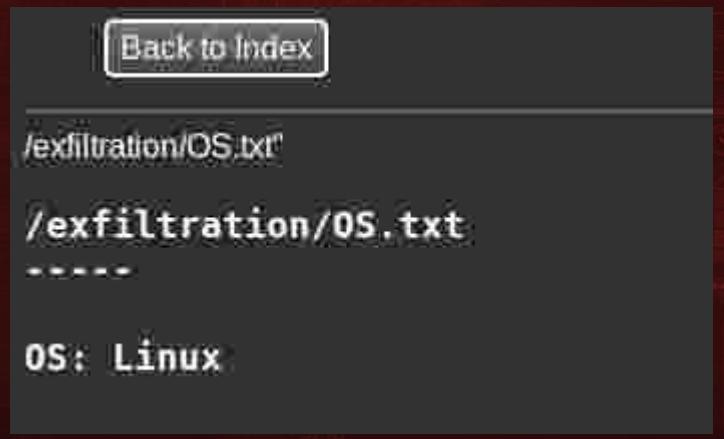
Jugando con WSD II: Physical Keyboard

Jugando con WSD III: Upload Payload

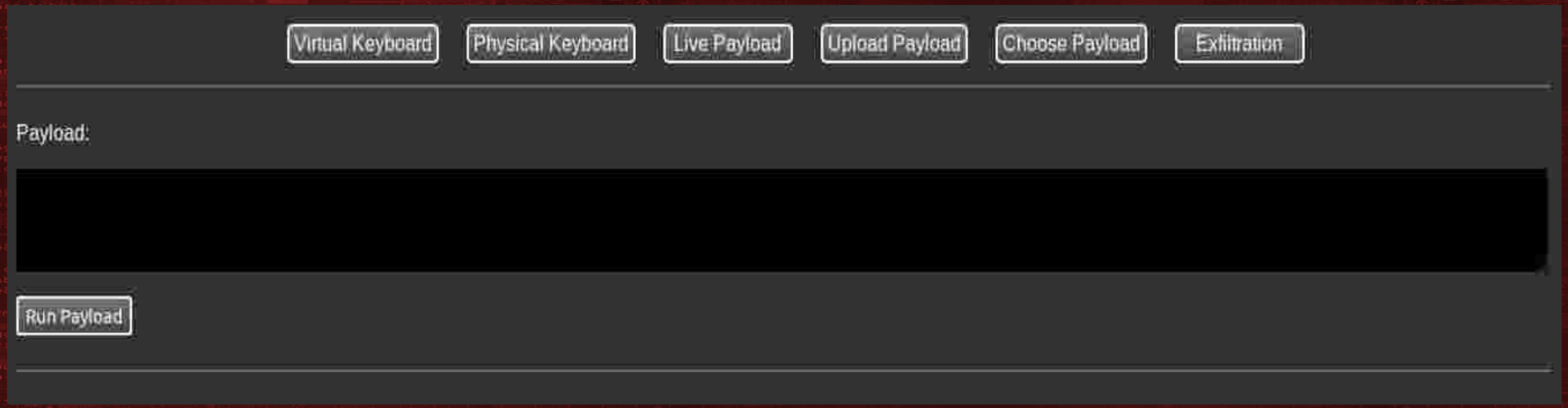


Jugando con WSD IV: Choose Payload

Jugando con WSD V: Exfiltration



Jugando con WSD VI: Live Payload

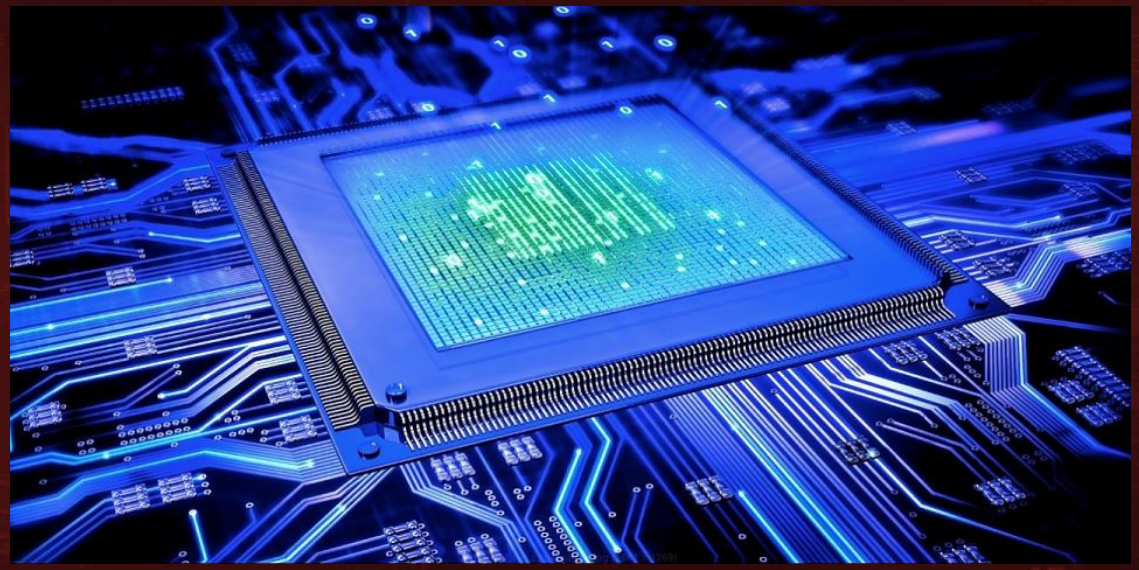


¿Y ahora qué?

- Subir otros códigos a la placa, ejemplo: ESPloitV2

- Desarrollar nuestros propios códigps

- Añadir nuevas funcionalidades a WSD, ejemplo: cambiar SSID y contraseña desde la web, Serial to WiFi, cliente Python, etc



Agradecimientos

- Carlos Barbero (@Nevnaur)
- Organización de Mundo Hacker
- Equipo I+D Phoenix Intelligence & Security

@RadioHacking

@yadox

@Santpapen

@LucaBongiorni

@C4T_13

@NN2ed_s4ur0n

@EA4FSV

@PCabreraCamara