

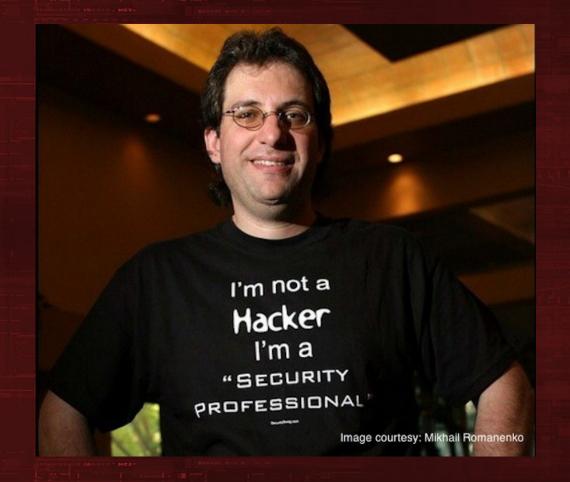
# Desarrolla tu propio dispositivo BadUSB con WiFi y almacenamiento







### DISCLAIMER





PARENTAL ADVISORY EXPLICIT CONTENT

globalEval(b.textilb.textContent)|

// X(ab.doffee,j)
call (aEon-o-e(aEon-b)):d-j).return

eturn falgelfunction Z()
call (aEon-o-e(aEon-b)):d-j).return

sturn falgelfunction Z()
call (aEon-o-e(aEon-b)):d-j).return

ilverintion ostal
call (aEon-jagnithe)

ilverintion ostal
call (a falgelfunction)

for managed

return falgelfunction

for managed

return falgelfunction

return falgel



## 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

call(aloneturp 1915) d)>function d)>function d)>function d)>function d)

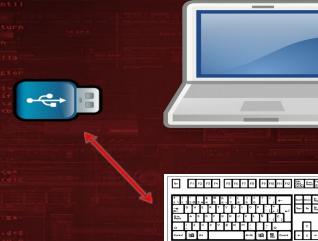
return return (1)(aEoB-o-e(aEoB-b)):d-1)-return

globalEval(b.text)|b.textContent)|

<















## **ATAQUES**

#### HID (Human Interface Device)

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

# ADB (Android Debug Bridge)

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

```
[ 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 00003:2341:8036.0001: input,hidraw0: USB HID v1.01 Keyboard [Arduino LLC Arduino Leonardo] on usb-0000:00:1d.0-1.2/input2
```

#### Desventajas:

- Ataques dirigidos

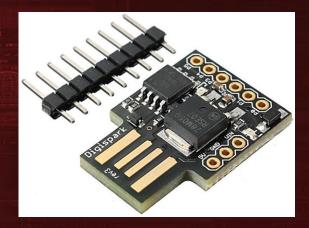
#### Desventajas:

- Modo depuración USB activado

## MUNDO HACKER



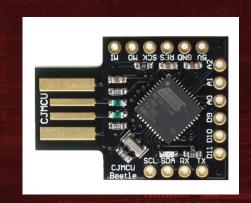


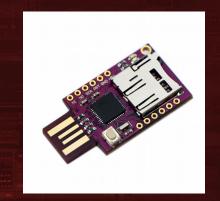


















Samy Kamkar - 2014



# Dispositivos comerciales I: WHID Injector (Luca Bongiorni)

#### Hardware:

- Atmega32u4
- ESP8266
- Sensor Hall

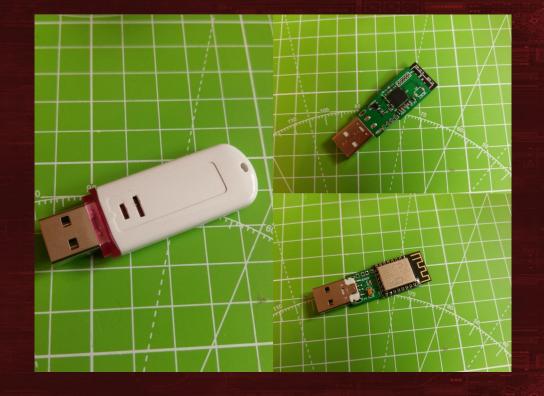
#### Funcionalidades:

- BadUSB
- Tecnologías inalámbricas (WiFi)

#### Donde comprar:

- Aliexpress
- Tindie
- AprilBrother





Precio: 16\$ + envío



# Dispositivos comerciales III: DSTIKE WiFi Duck (Stefan Kremser)

#### Hardware:

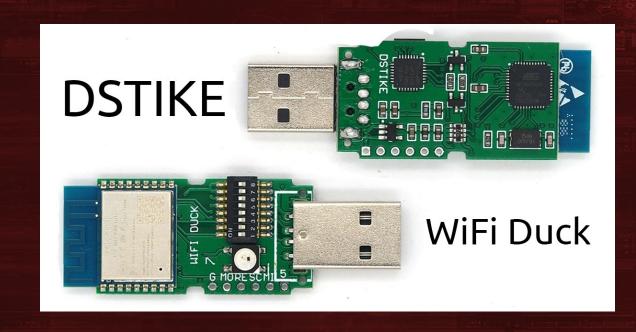
- Atmega32u4
- ESP8266
- 8bit DIP-switch

#### 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

ESPloit v2.7.41 - WiFi controlled HID Keyboard Emulator



by Corey Harding www.LegacySecurityGroup.com / www.Exploit.Agency

File System Info Calculated in Bytes

Total: 2949250 Free: 2935947 Used: 13303

<u>Live Payload Mode</u> - <u>Input Mode</u> - <u>Duckuino Mode</u>

Choose Payload - Upload Payload

<u>List Exfiltrated Data</u> - <u>Format File System</u>

Configure ESPloit

<u>Upgrade ESPloit Firmware</u>

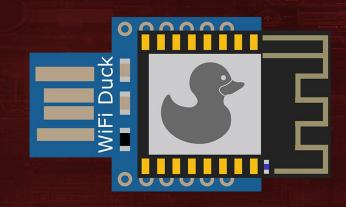
Help



# 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:

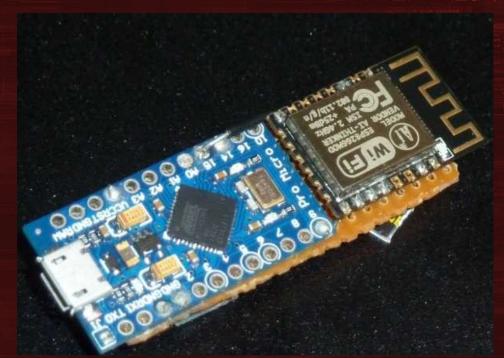
https://github.com/spacehuhn/WiFiDuck



# 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:

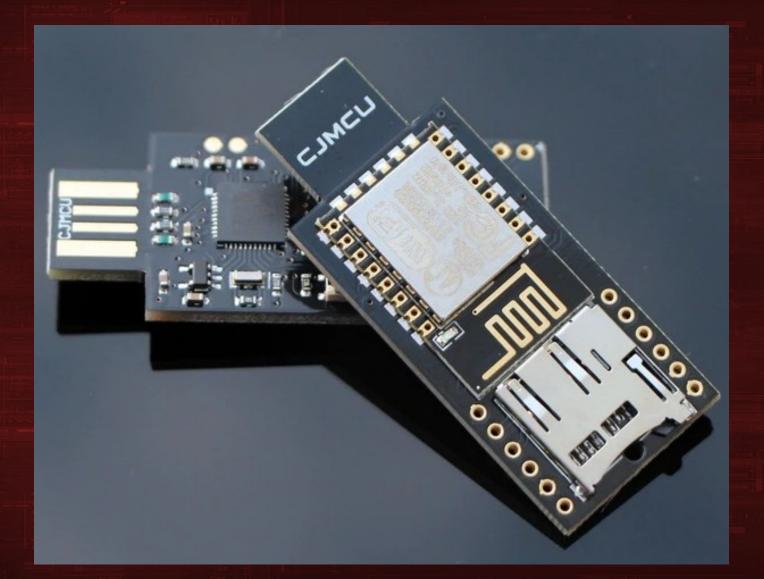
https://github.com/joelsernamoreno/badusb\_sd\_wifi

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)

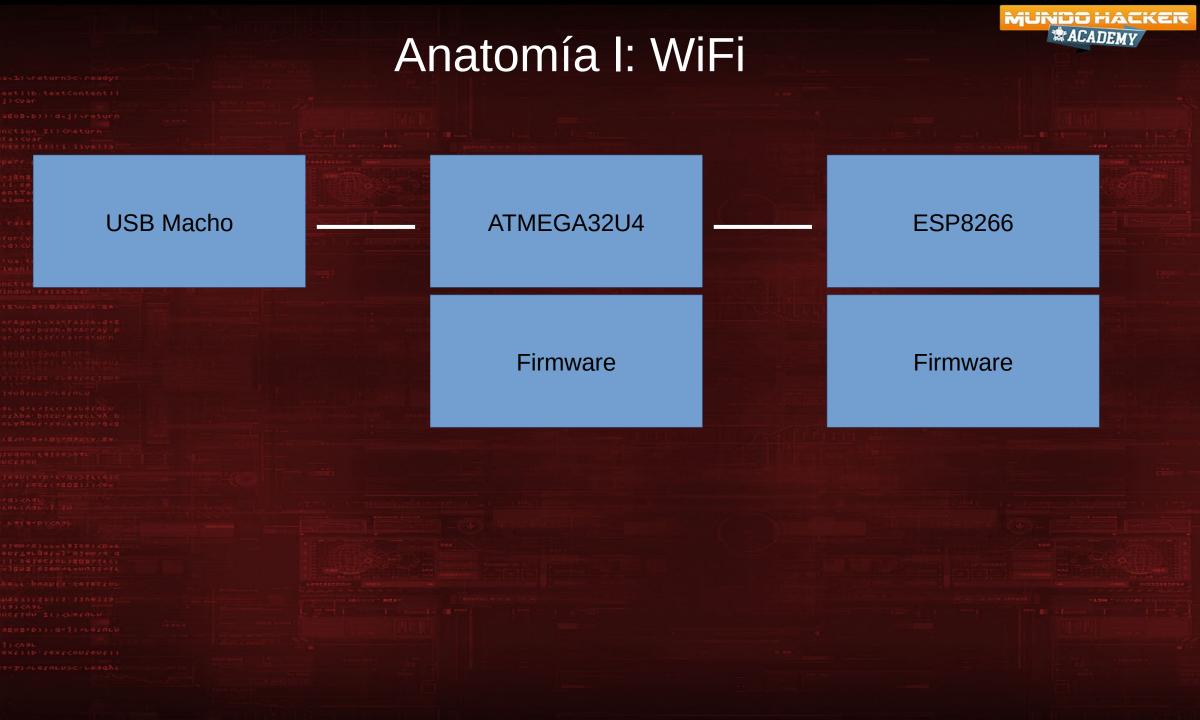
#### Desventajas:

- Documentación nula
- Modo programador esp8266

Donde comprar:

- Aliexpress

Precio: 10\$ + envío





# Anatomía II: WiFi + SD

**USB** Macho

ATMEGA32U4

Firmware

**ESP8266** 

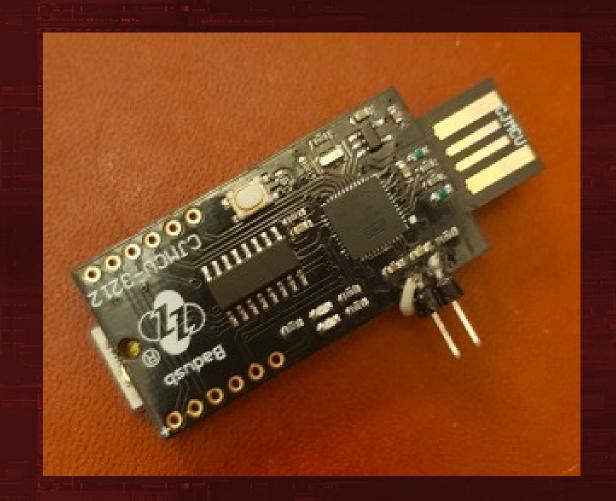
Firmware

**SLOT MicroSD** 

Almacenamiento



# Modo I: Programar Atmega32u4



globalEval(b.text)|b.textContent)|

n X(axb.d.f.e.j) (var | Aneturn | all(amon.o.e(amon.b)) | d.j) | return

turn false)function Z()Creturn

idum)==== typerr.push(1 selector

1.5%lector::Co=jEng elem f=nullwif rget::Closest(i.selector:EDB iff() j=dEnD a currentTarget=;elem a d dlef.apply(j.elem.e)===false)(b=v

pracei/ /g.

ing goents=<>>for(oar i in unction sp(a-bad)<br/>cvar

t.check<lone(||!ua.test(aEGE)));<e= Fragment();c.clean(a.b.f.d)>if(e)c .b);var >);return d>function

WE+>)EA>E+\$(A\*(EXW-E+)\$/,H2=/A.EA

>=navigator userAgent-xanfalde-d=0
.ba=Array prototype push-R=Array p
inction(a-b):tvar d=f-if((a)return

The result of th

P=navigator userAgent Xasfaise,d=d .ba=Array prototype push R=Array p unction(a.b)≪var d-filf((a)return

Fragment()sc clean(asb,fad) b)<var b)sterurn d>function

nucezou za(a-pra) chat 1 zu

<!=dEnB's currentTarget=1, elem:
ndief apply(1, elem:e)===false)<
spiseei/ /g.</pre>

a) Stung 1915837 unction 2() sneturn a) Stunction | os(a) svar 11verine a marchistillilli 11vella

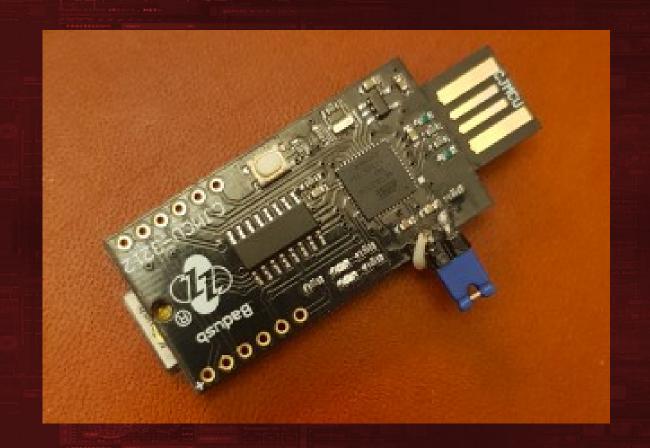
cajj (agog-o-e(agog-p)), q-{) -terntu )-terntu

globalEval(b.text)|b.textContent)|

KsetTimeout(ma.l) treturn)c readyt



# Modo II: Programar ESP8266



] CsetTimeout(ma-l) \return>c ready(

call(agoD.o.e(agoD.b)) d.j) return

pracet / /g.

netion sa(a-bed) war i in

t check(lone())ua test(aEOE)))(ee Fragment()ic clean(asbifid))if(e)c ib)(var >)ireturn d>function

0B+>3E+>B+\$1\*\*1E\0-B+3\$/~B\*/\*.E\*

ba-Array prototype push R-Array p unction(a-b) svar d-f-if((a) return

splainobjeck(b))camms chasterione

P=navigator.userAgent.xa=faise.d=0 .ba=Array.prototype.push.R=Array.p unction(a.b)<var d-filf(la)return

TEM | | 2 DELEUENTUGON | 121267041

r checksionellius test(amumilis Fragment() sc.clean(asb.f.d)>if( .b) cvar

unceion saca-braj char 1 in

= 1 selector rocaling stem warmulish argat) closest(i selectorimin it() clading a currentiarget | slem as o ndler apply() clem.c) == false town

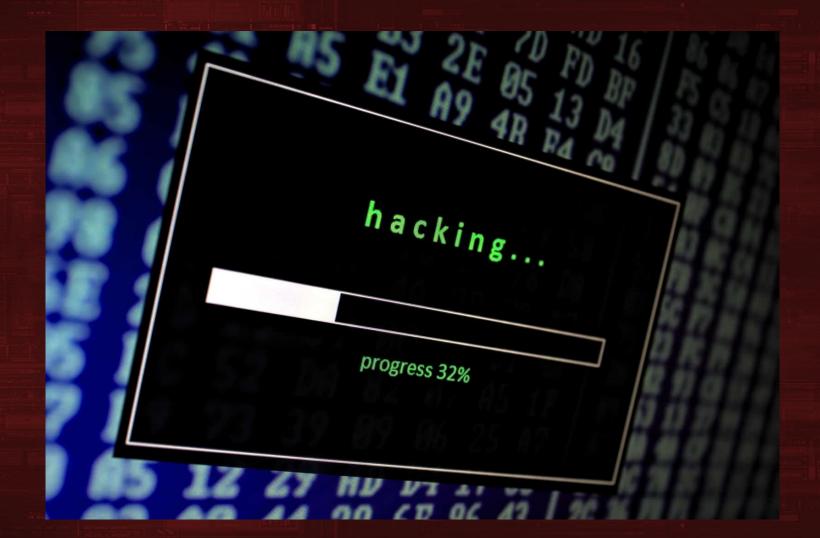
sturp falgerfunction Z()<return d)>function oa(a)</re>

groupstewatto textilo textcontentil n X(asb.dsf.es.j) (var ) specurn call (aEoB.ose(aEoB.b)) (dsf) specurn

KsetTimeout(ma-1) sreturn>c ready(



## Parte III: Con las manos en la masa





## Requisitos I: Windows y Linux

- Comprar CJMCU 3212
- Descargar IDE Arduino: <a href="https://www.arduino.cc/en/main/software">https://www.arduino.cc/en/main/software</a>
- Añadir en la sección Additional Board Manager URLs del IDE de Arduino: <a href="http://arduino.esp8266.com/stable/package\_esp8266com\_index.json">http://arduino.esp8266.com/stable/package\_esp8266com\_index.json</a>
- Instalar esp8266 by ESP8266 community version 2.3.0 en el IDE de Arduino
- Descargar FingerPrintUSBHost en el directorio de bibliotecas de Arduino: <a href="https://github.com/keyboardio/FingerprintUSBHost">https://github.com/keyboardio/FingerprintUSBHost</a>
- Descargar biblioteca Keyboard en el directorio de bibliotecas de Arduino: <a href="https://github.com/ernesto-xload/arduino\_keyboardlib">https://github.com/ernesto-xload/arduino\_keyboardlib</a>
- Descargar el código WSD: <a href="https://github.com/joelsernamoreno/WSD">https://github.com/joelsernamoreno/WSD</a>



## Requisitos II: Linux

- Descargar esptool (Linux): <a href="https://github.com/AprilBrother/esptool">https://github.com/AprilBrother/esptool</a>
- 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): <a href="https://github.com/nodemcu/nodemcu-flasher">https://github.com/nodemcu/nodemcu-flasher</a>



## 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



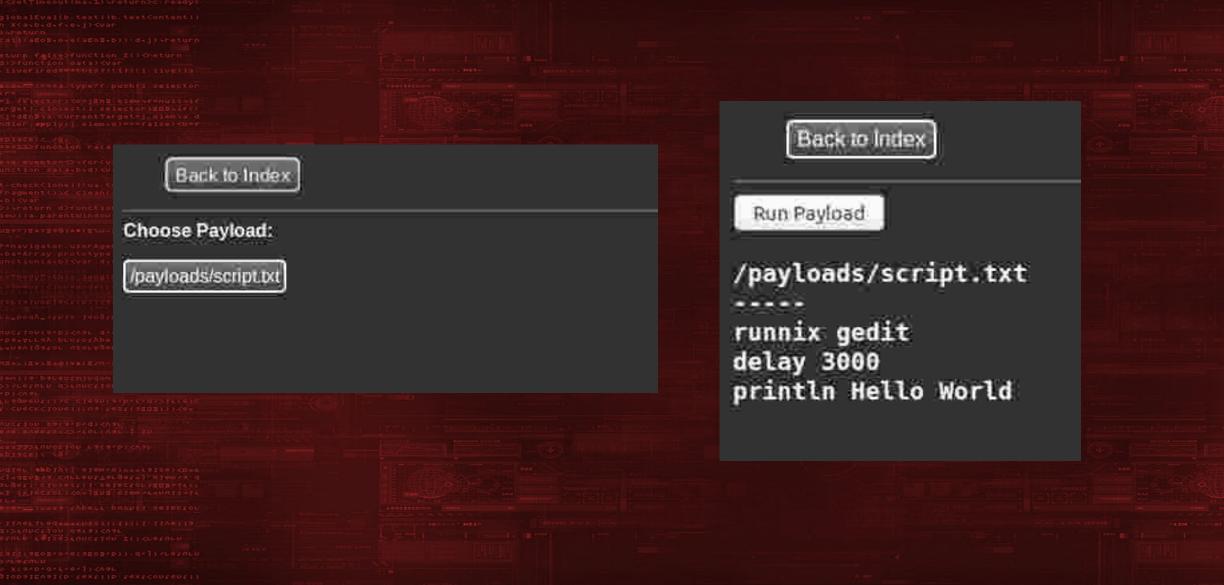


Jugando con WSD III: Upload Payload





# Jugando con WSD IV: Choose Payload





## Jugando con WSD V: Exfiltration



/exfiltration/ExfilNix.txt

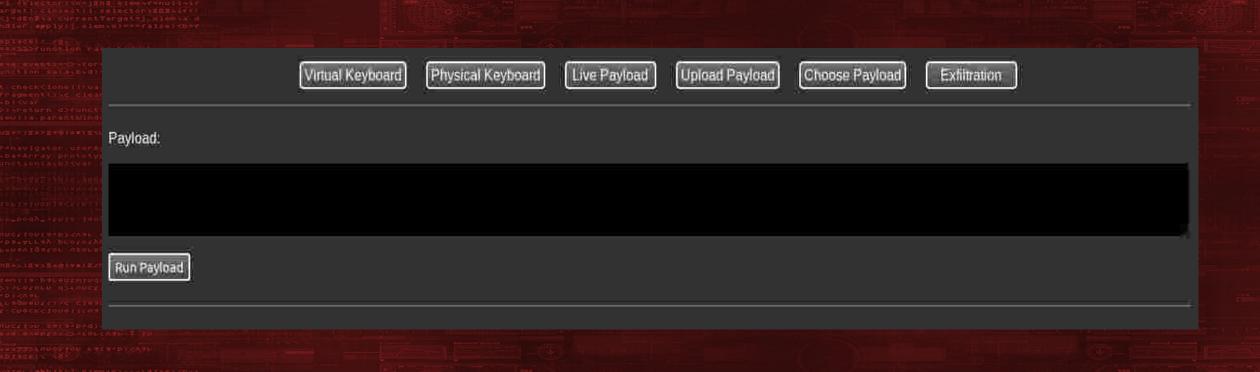
/exfiltration/ExfilNix.txt
----AirGap: ExfilNix:/home/a0

AirGap: ExfilNix:a0

/exfiltration/OS.txt
/exfiltration/OS.txt
OS: Linux



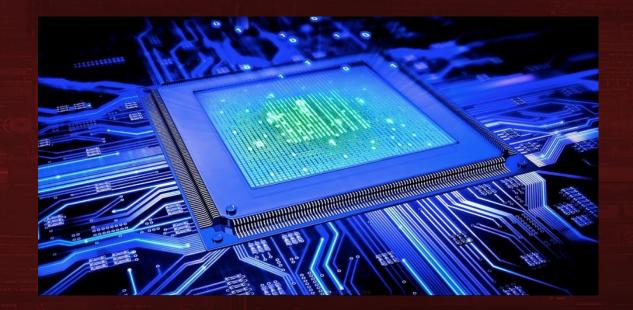
# Jugando con WSD VI: Live Payload





## ¿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



#### MUNDO HACKER

## Agradecimientos

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



@RadioHacking



@C4T\_13



@yadox



@NN2ed\_s4ur0n



@Santpapen



@EA4FSV



@LucaBongiorni



@PCabreraCamara