TARLOGIC
CYBERSECURITY EXPERTS

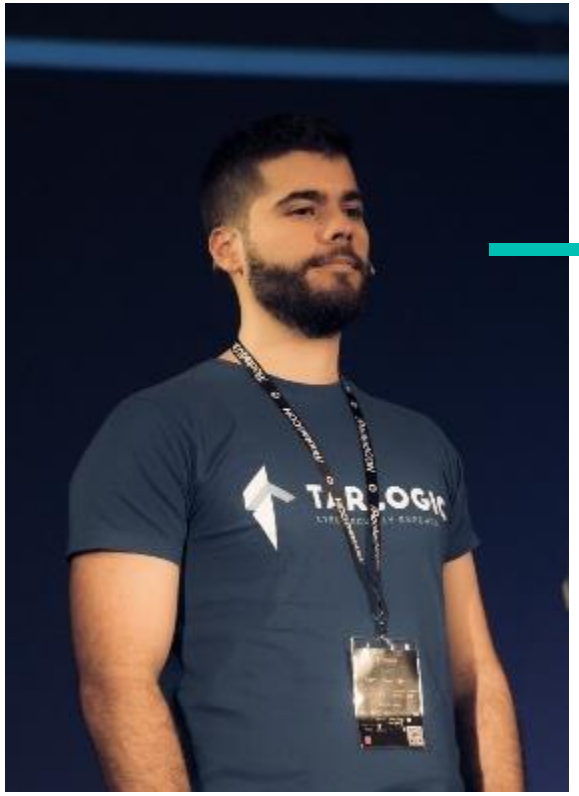
/RootedCON

Marzo 2025

Attacking Bluetooth the easy way

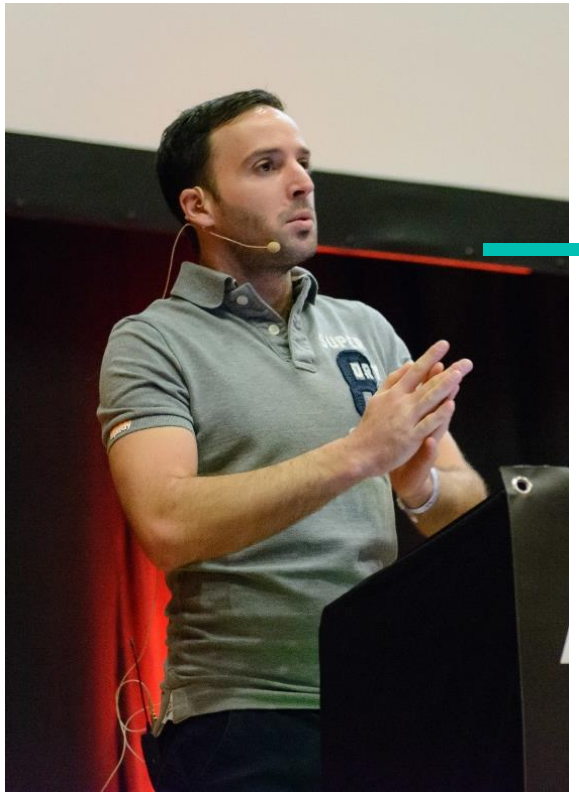
(Or at least a little bit easier…)

$ WHOAMI



Antonio Vázquez Blanco

- antonio.vazquez@tarlogic.com
- @antoniovazquezblanco@mastodon.social
- @antonvblanco



Miguel Tarascó Acuña

- miguel.tarasco@tarlogic.com

TARLOGIC
CYBERSECURITY EXPERTS

01 SITUACIÓN ACTUAL

SITUACIÓN ACTUAL

2021

Necesidad de realizar auditorías Bluetooth, pero información dispersa e incompleta

Investigación para la comprensión en profundidad de la tecnología Bluetooth

2022

BlueTrust
Clonado de identidades Bluetooth e identificación de relaciones de confianza entre dispositivos

Desarrollo de una documentación completa y accesible

2023

BSAM
Metodología de auditorías Bluetooth completa, pública y open source

SITUACIÓN ACTUAL

› **Bluetooth Security Timeline**:

https://darkmentor.com/bt.html

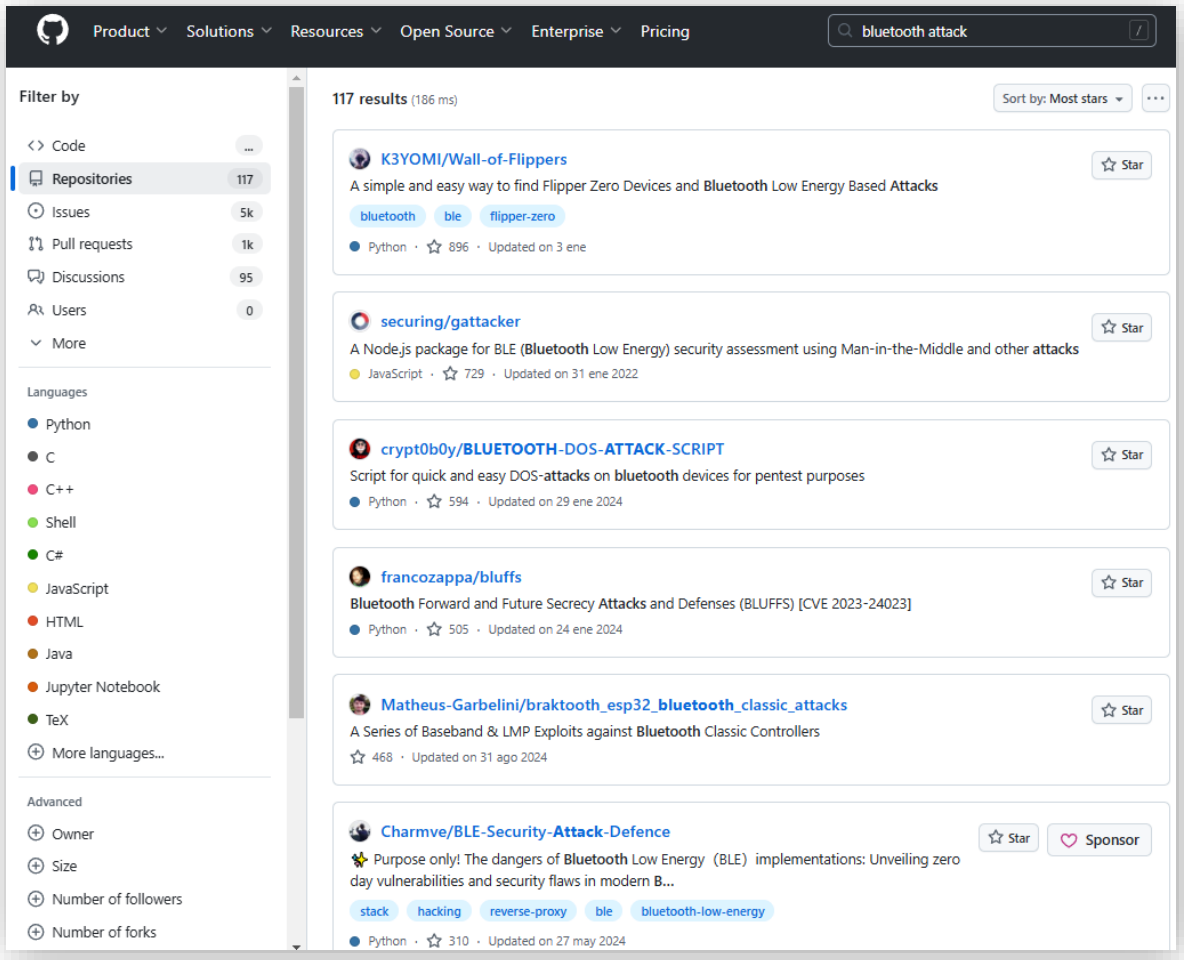
› **En 2024 se ha publicado sobre Bluetooth**:

- 30 ataques/ publicaciones
- 7 tienen PoC/ Herramienta
- 3 pueden ser usadas con hardware genérico
- Únicamente 1 es compatible con múltiples sistemas operativos…

SITUACIÓN ACTUAL

- **Mirando más atrás en el tiempo…**
  https://github.com/search?q=bluetooth%20attack&type=repositories

- **Muchas herramientas y ataques implementados:**
  - Poco (o nada) mantenidas
    - Issues
    - Pull requests
  - Obsoletas
    - >3 años sin actualizarse
    - Incompatibles con software/APIs actuales



SITUACIÓN ACTUAL

› Hardware Multipropósito

- Flipper
- M5Stack
- M5-NEMO
- CapibaraZero

ESP32 NRF24L01 and CC1101 Board for Flipper Zero

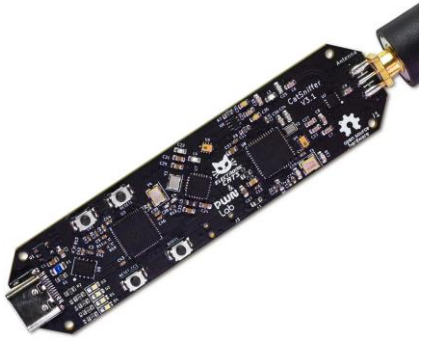
Combo ESP32/GPS/CC1101 3in1 for Flipper Zero

https://www.tindie.com/search/?q=Flipper+Zero

SITUACIÓN ACTUAL

› Mucho hardware

- Comunicaciones
  - Dongles
  - Placas de desarrollo
- Sniffers



SITUACIÓN ACTUAL

- **La situación actual a nivel de software es compleja…**
- **Pocas alternativas:**
  - APIs del OS de alto nivel y bajo nivel.
- **Cada OS tiene su propio “Framework/Stack”**
  - Bluez
  - CoreBluetooth
  - BlueSoleil
  - WidComm
  - Windows Bluetooth Stack
  - Muchas otras adhoc o propietarias…

TARLOGIC
CYBERSECURITY EXPERTS

02 ¿Y AHORA QUÉ?

¿Y AHORA QUÉ?

- Ya tenemos **BSAM y documentación**
- Necesitamos **unificar/simplificar el desarrollo** de herramientas
- Necesitamos que sea **multiplataforma**
- **No dependiente** de hardware
- **Independientemente del lenguaje de programación**

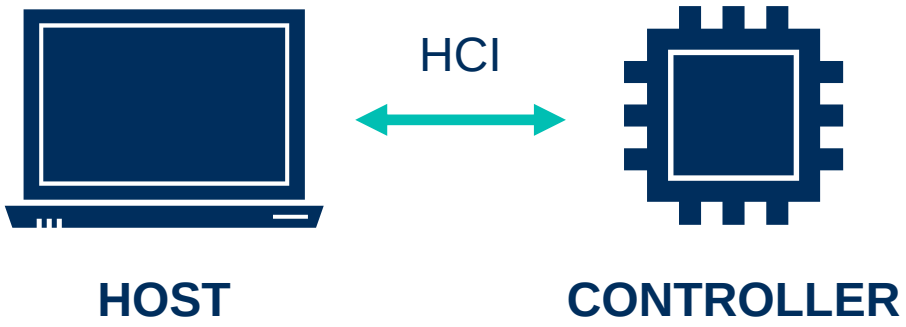
¿Y AHORA QUÉ?

› **Arquitectura física de Bluetooth**:

- **Controller** – El chip con capacidades Bluetooth que realiza las operaciones de bajo nivel
- **Host** – El PC u otro dispositivo que utiliza la conectividad del “Controller”
- **HCI o “Host Controller Interface”** – Una API estándar

› HCI esta diseñado para ser independiente del OS

› HCI puede funcionar sobre USB o UART/Serial



¿Y AHORA QUÉ?

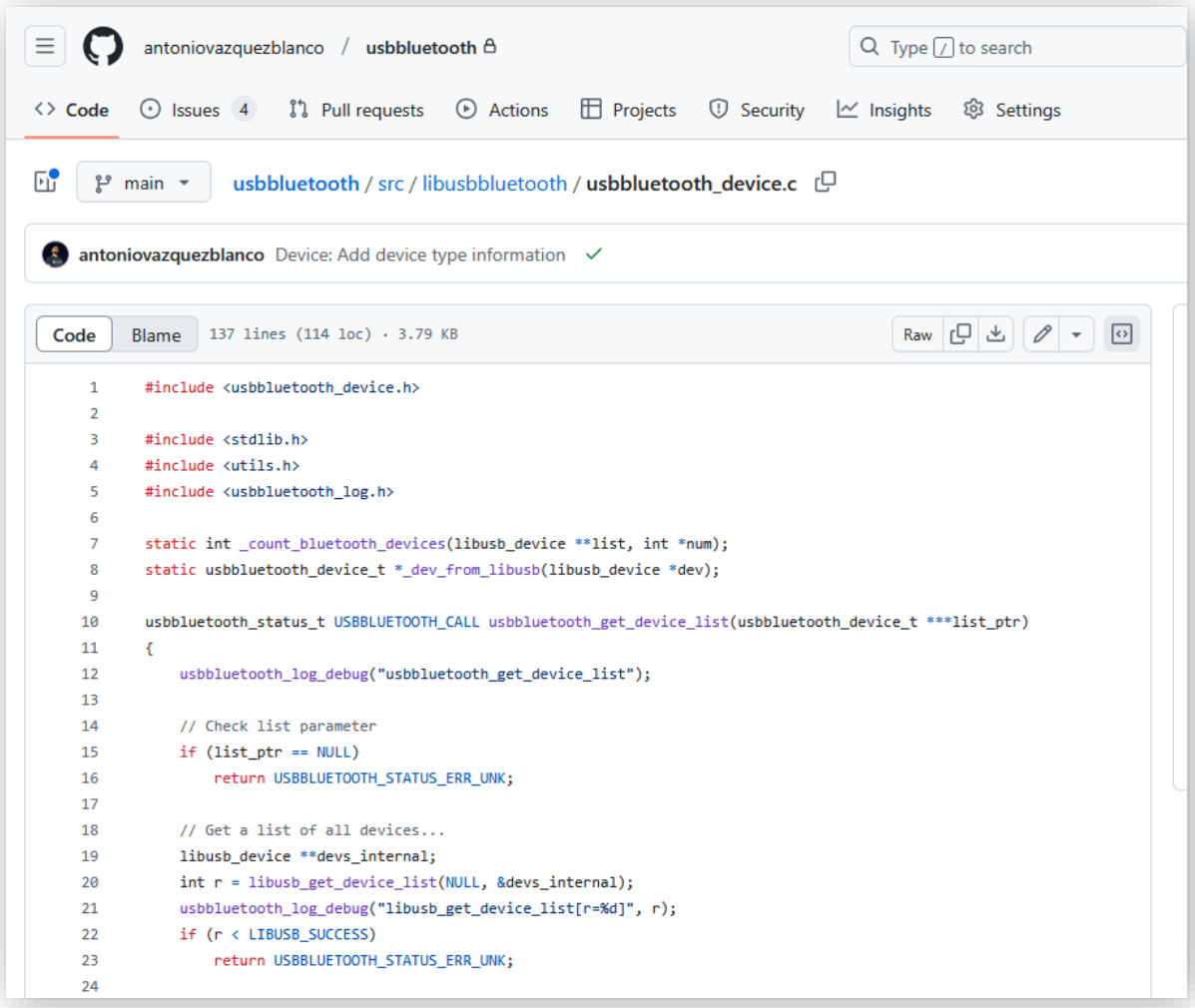
NECESIT ER HCI…

SOLUCIONES

› **USB Bluetooth:**

- Driver HCI
- Escrito en C
- Independiente de la plataforma
- Independiente del hardware

› **https://github.com/antoniovazquezblanco/usbbluetooth**



SOLUCIONES

El driver accede directamente al hardware Bluetooth a través de:

› Windows

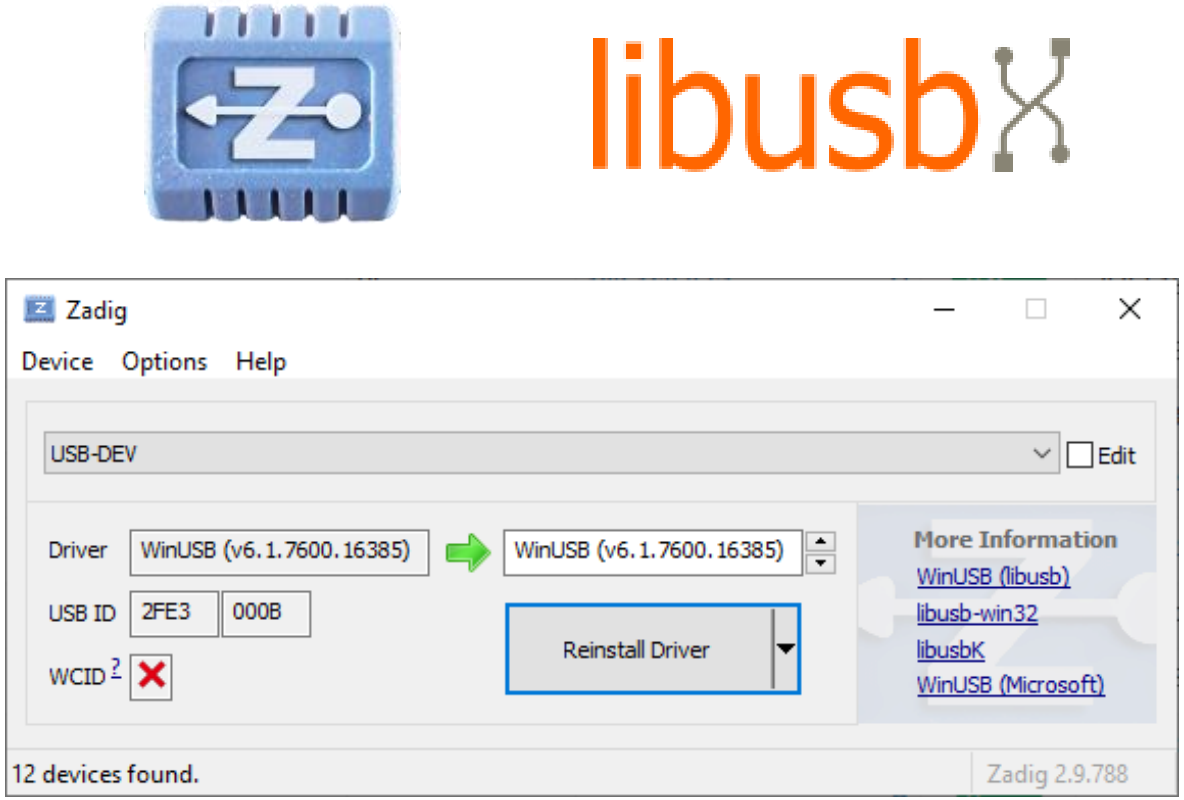
- WinUsb

› Linux

- https://libusb.info

› Mac

- https://libusb.info



SOLUCIONES


```
import usbbluetooth

# Get a list of all the available devices
devices = usbbluetooth.list_devices()
for dev in devices:
    print(dev)

# Open the device
with devices[0] as dev:
    # Send a reset command
    dev.write(b"\x01\x03\x0c\x00")
    # Read the respose
    response = dev.read()
    print(response)
```

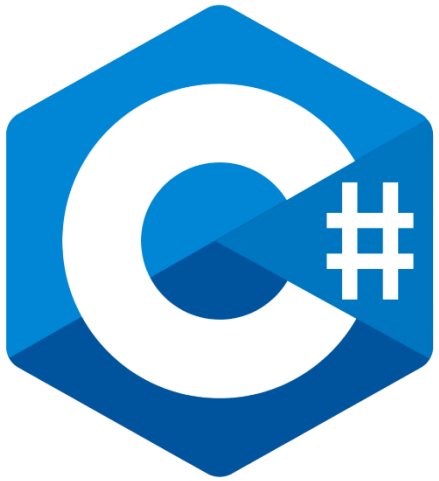
SOLUCIONES


```
from scapy.all import *
import usbbluetooth
from scapy_usbbluetooth import UsbBluetoothSocket

# Get a device
devices = usbbluetooth.list_devices()
# Obtain a socket
s = UsbBluetoothSocket(devices[0])
# Send a reset command
p = HCI_Hdr() / HCI_Command_Hdr() / HCI_Cmd_Reset()
# Await a response
r = s.sr1(p)
r.show()
```

https://github.com/antoniovazquezblanco/scapy-usbbluetooth

SOLUCIONES



```
using UsbBluetooth;

UsbBluetoothManager.Init();

// List all devices
UsbBluetoothDevice[] devices = UsbBluetoothManager.ListDevices();
foreach (UsbBluetoothDevice dev in devices) {
    Console.WriteLine($"Device found: {dev.ToString()}");
}

// Open a device
UsbBluetoothDevice device = devices[0];
device.Open();

// Reset the device
device.Write(new byte[] { 0x01, 0x03, 0x0c, 0x00 });

// Read the response
Console.WriteLine(BitConverter.ToString(device.Read()));
```

DEMO

```
// Initialize driver and get a device
UsbBluetoothManager.Init();
UsbBluetoothDevice device = UsbBluetoothManager.ListDevices()[0];
device.Open();

Task.Run(() => {// Receiver thread
    while (true) {
        byte[] data = device.Read();
        if (data == null || data.Length == 0) continue;
        Console.WriteLine($"Packet ({data.Length}):\r\n "{BitConverter.ToString(data)}");
    }
});

device.Write(new byte[] { 0x01,0x03,0x0C,0x00 }); // CMD_RESET
device.Write(new byte[] { 0x01,0x0b,0x20,0x07,0x01,0x10,0x00,0x10,0x00,0x00,0x00 }); // CMD_LE_SET_SCAN_PARAMETERS
device.Write(new byte[] { 0x01,0x0c,0x20,0x02,0x01,0x00 }); // CMD_LE_SET_SCAN_ENABLE

Thread.Sleep(10000); // Scan for 10 secs

device.Write(new byte[] { 0x01,0x0c,0x20,0x02,0x00,0x00 }); // CMD_LE_SET_SCAN_ENABLE_STOP
device.Close();
```

SOLUCIONES

RECAPITULAMOS

- ¡Tenemos un **driver independiente multiplataforma** para Bluetooth!
- Con soporte para **múltiples lenguajes**
- Nos permite **interactuar con nuestro hardware Bluetooth** sin restricciones

USOS

- **Implementación de herramientas utilizando funcionalidades estándar de bluetooth**
  - Escáner de dispositivos
  - Automatización de pruebas de conexión y pairing
  - …
- **Implementación de herramientas que requieren paquetes malformados**
  - Ataques con paquetes no estándar
  - Fuzzing!

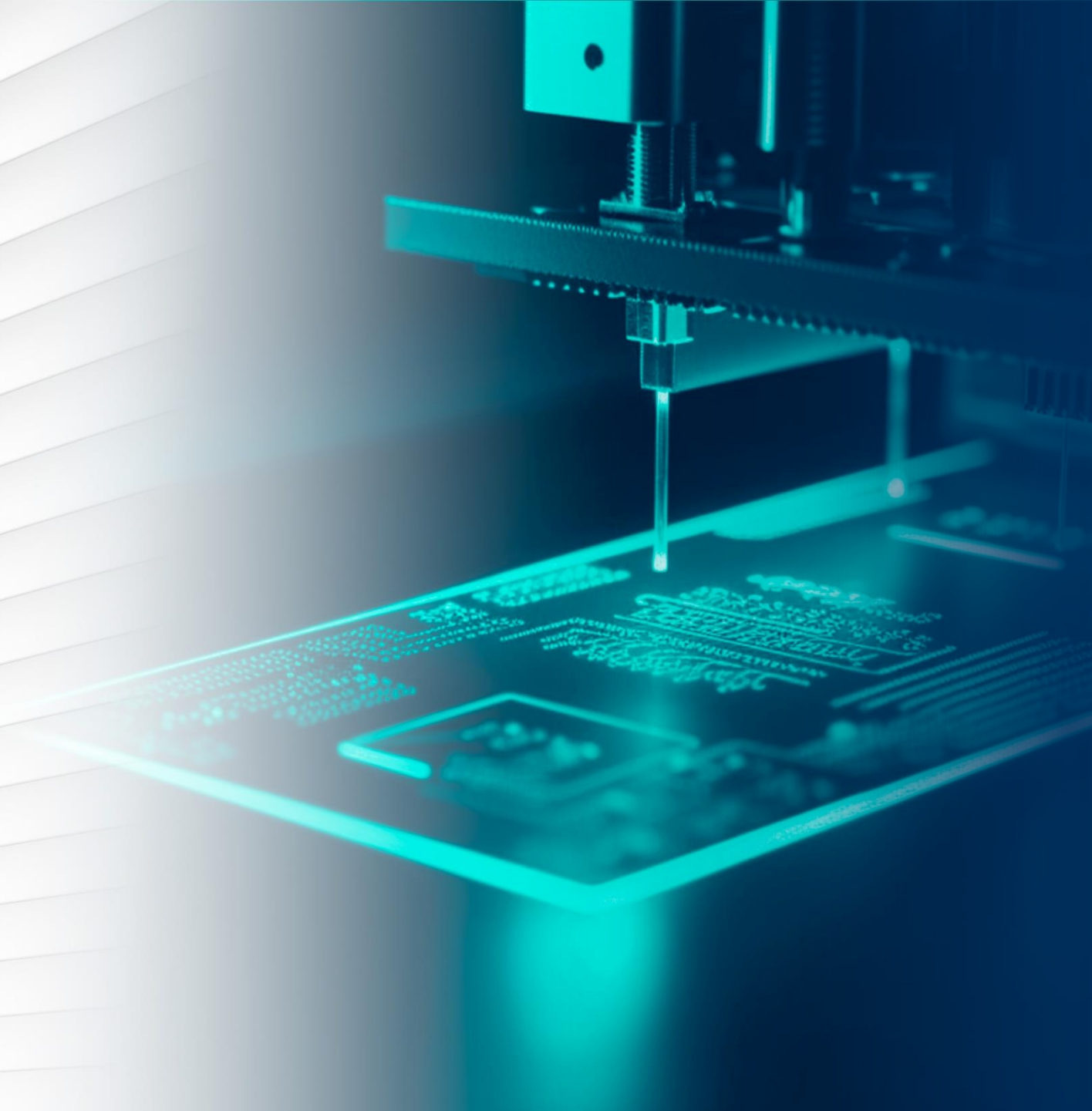
SOLUCIONES

LIMITACIONES

- Está **limitado al comportamiento estándar** del controlador
- Seguimos necesitando un **hardware específico** que nos permita **implementar ataques avanzados**
- Usar el hardware de tu equipo, por lo general chip dual **WiFi+BT no es buena idea** (te quedas sin wifi)

TARLOGIC
CYBERSECURITY EXPERTS

03 HARDWARE



HARDWARE

› **Necesitamos hardware con capacidades Bluetooth avanzadas…**

- Disponible
- Fácil de comprar en cualquier parte del mundo
- ¡Barato!

› **Idealmente que soporte Bluetooth Classic y Low Energy**

HARDWARE

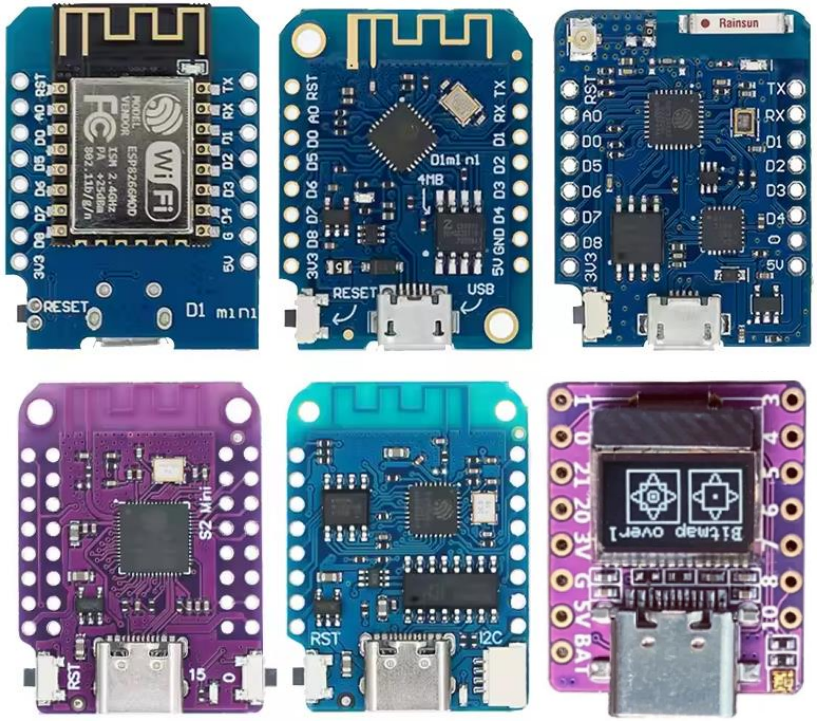
ESP32 NRF24L01 and CC1101 Board for Flipper Zero

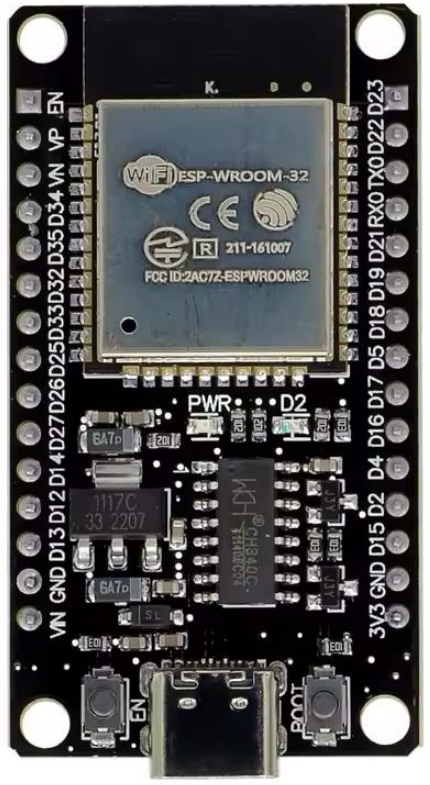
Combo ESP32/GPS/CC1101 3in1 for Flipper Zero

CapibaraZero

HARDWARE


ESP32

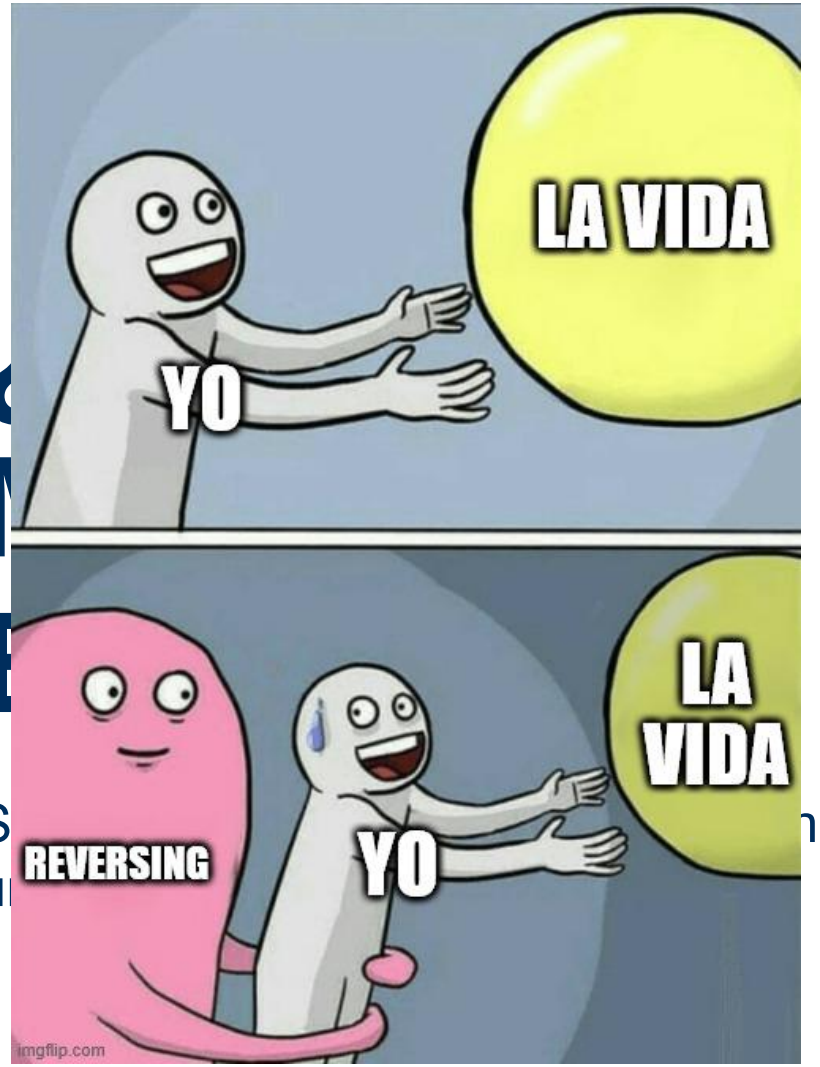

RESET
D1 mini
4MB
USB
Rainsun
TX
RX
RST
S2 Mini
I2C
bitmap over!


ESP-WROOM-32
211-161007
FCC ID:2AC7Z-ESPWROOM32
PWR
D2
EN
BOOT

HARDWARE

¿USOS DEL ESP32?

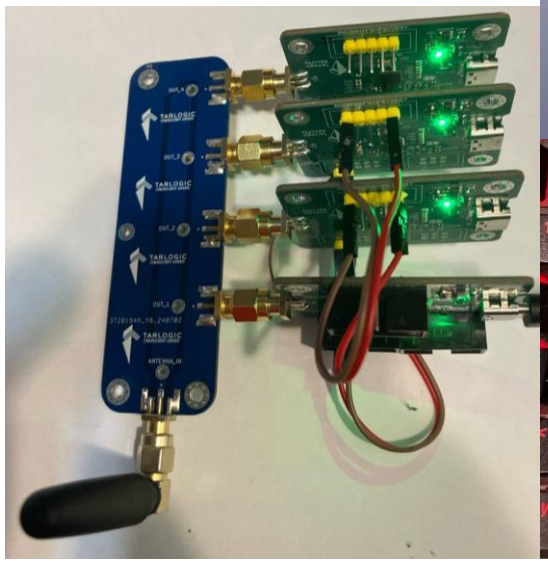
- **Mediante firmwares personalizados**
  - Tu flasheas el dispositivo
  - Funcionamiento especializado
- **Funcionalidades avanzadas**
  - Parches en el binario de Espressif
  - Problemas de estabilidad
  - Incompatibilidad entre versiones del chip/rom

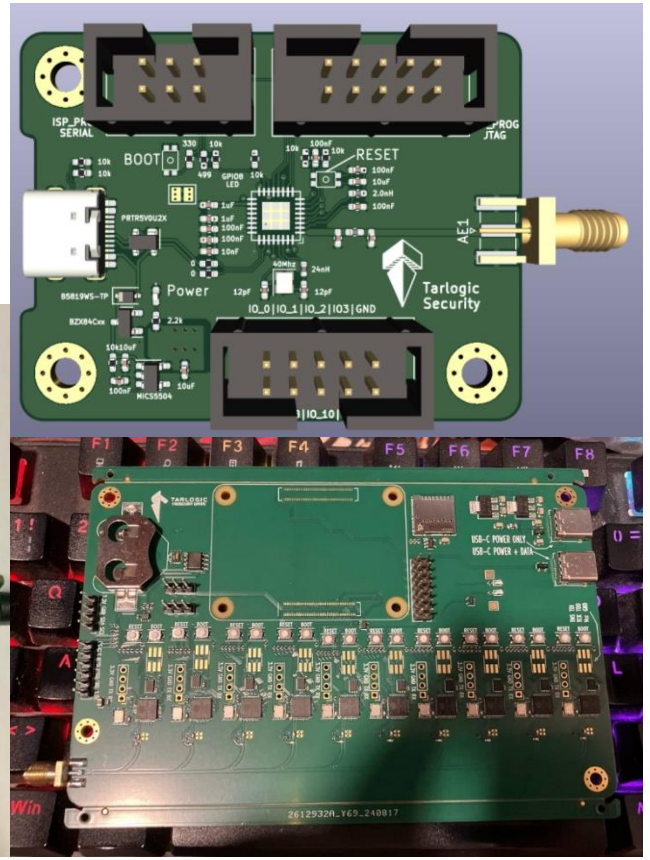


HARDWARE

- Ingeniería inversa **firmware ESP32**
- **Desarrollo de prototipos** de Hardware
- **Sniffer "modo monitor"** Bluetooth asequible
- Por desgracia, el resumen rápido es que el **silicio del ESP32 no lo permite**

Peeeeero….

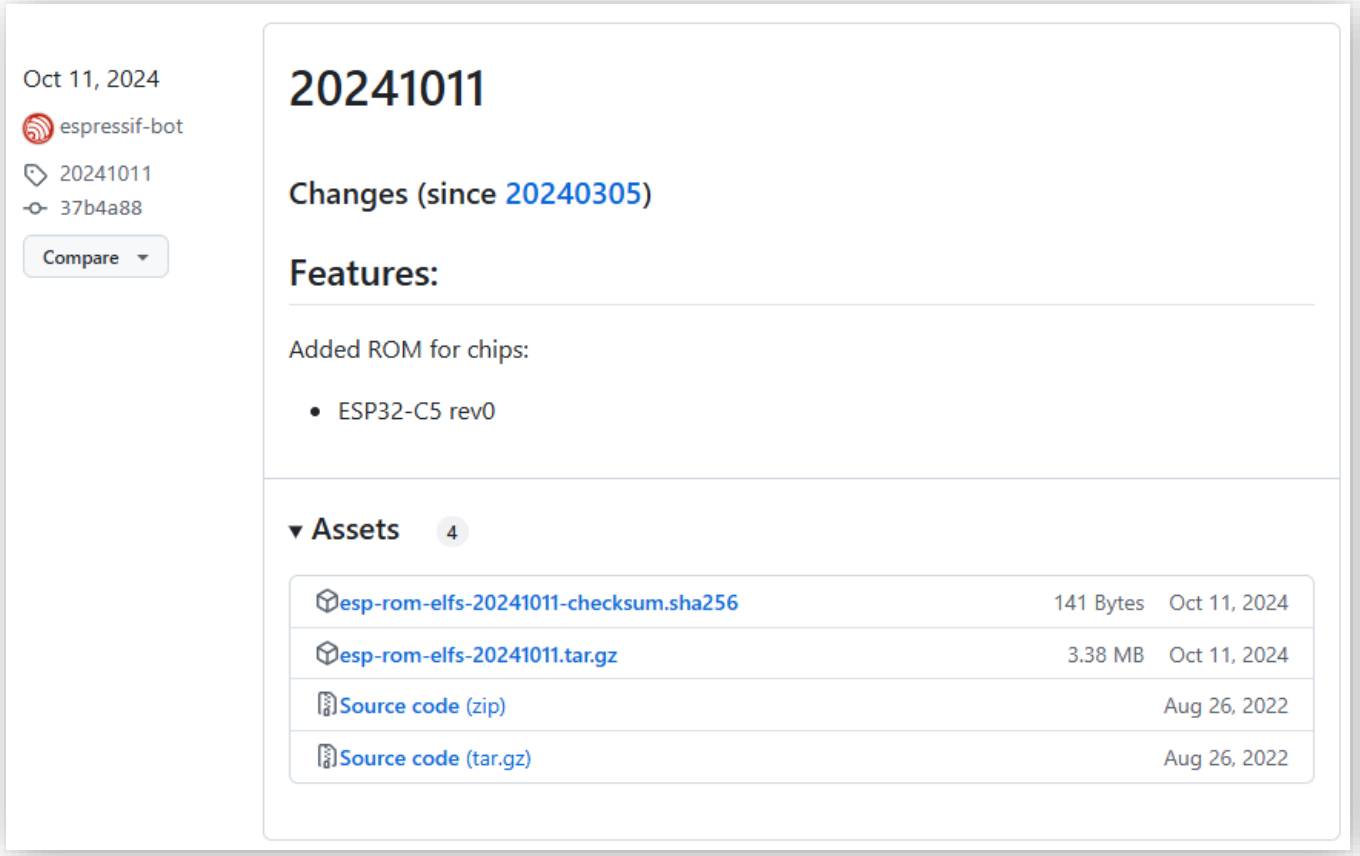




REVERSING

› Espressif facilita las **ROMs internas del ESP32**

https://github.com/espressif/esp-rom-elfs/releases



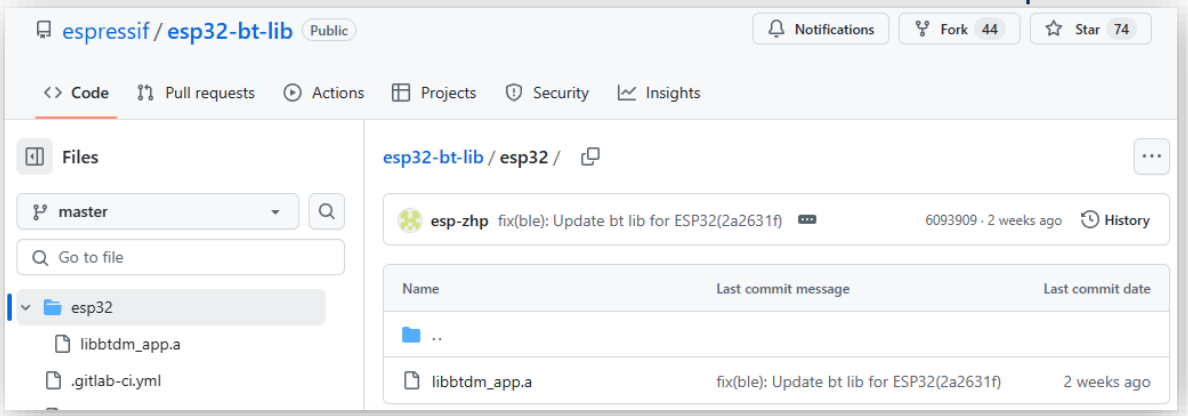
REVERSING

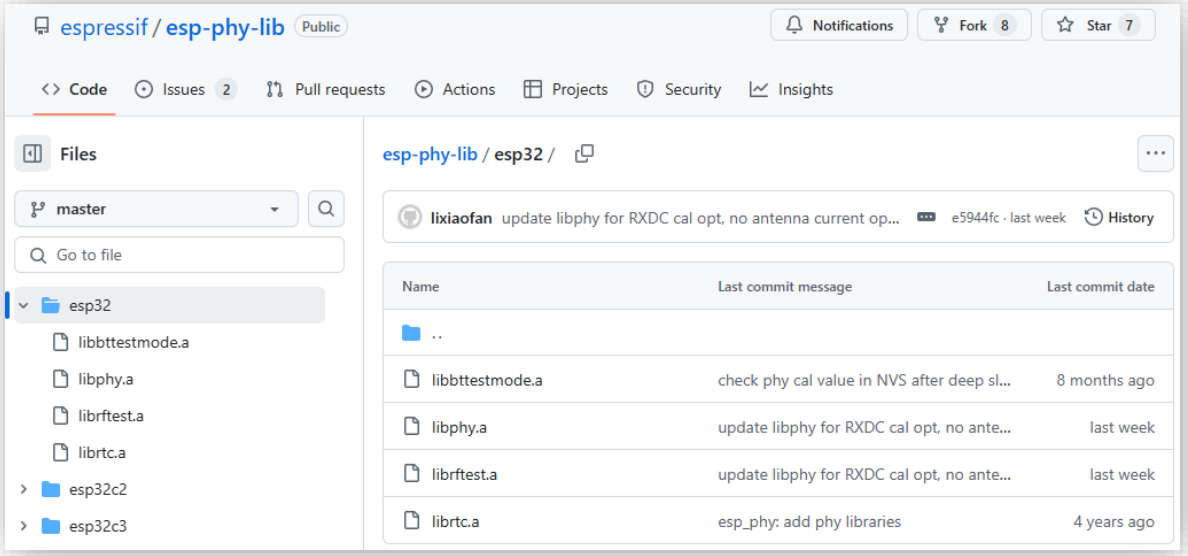
› Espressif también **ofrece las librerías con**:

- El stack del controller de Bluetooth
- Los drivers de la radio…
- Son binarios, no hay código

https://github.com/espressif/esp32-bt-lib/

https://github.com/espressif/esp-phy-lib/





REVERSING

› Muchos binarios, es necesario **simplificar el reversing:**

- Unir las librerías con el **linker**
- Se analiza **un archivo en vez de cinco**…

```
PS C:\> xtensa-esp32-elf-ld -r -o golden_bin.elf --whole-archive .\libbtdm_app.a .\libbttestmode.a
.\libphy.a .\librftest.a .\librtc.a

PS C:\> ls .\golden_bin.elf
    Directorio: C:\
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        25/02/2025      9:17        4156284 golden_bin.elf
```

REVERSING

› Espressif publica los **linker scripts**

https://github.com/espressif/esp-idf/blob/master/components/esp_rom/esp32/ld/esp32.rom.ld

› Publicaremos un **plugin de Ghidra** para cargar estos datos

https://github.com/antoniovazquezblanco/GhidraLinkerScript

esp-idf / components / esp_rom / esp32 / ld / esp32.rom.ld

BetterJincheng and espressif-bot — fix(bt/controller): Fixed some bugs... — 5e0a73b · 3 months ago

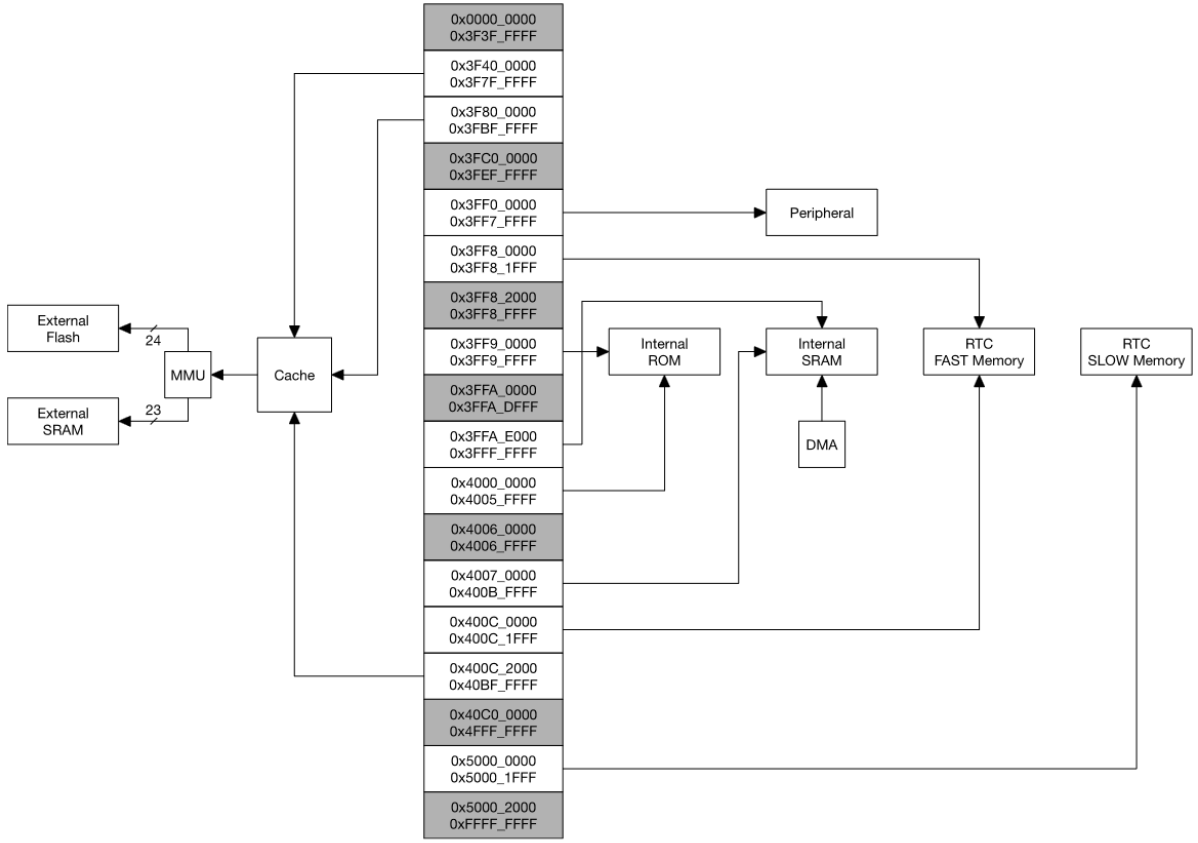
1638 lines (1631 loc) · 73.3 KB

Code | Blame | Raw

```
1   /*
2   ESP32 ROM address table
3   Generated for ROM with MD5sum:
4   ab8282ae908fe9e7a63fb2a4ac2df013  ../../rom_image/prorom.elf
5   */
6   PROVIDE ( Add2SelfBigHex256 = 0x40015b7c );
7   PROVIDE ( AddBigHex256 = 0x40015b28 );
8   PROVIDE ( AddBigHexModP256 = 0x40015c98 );
9   PROVIDE ( AddP256 = 0x40015c74 );
10  PROVIDE ( AddPdiv2_256 = 0x40015ce0 );
11  PROVIDE ( app_gpio_arg = 0x3ffe003c );
12  PROVIDE ( app_gpio_handler = 0x3ffe0040 );
13  PROVIDE ( BasePoint_x_256 = 0x3ff97488 );
```

REVERSING

- Las librerías acceden a **regiones de memoria fijas**
- Estas regiones son periféricos del **ESP32**
- Poca información en el **datasheet**…
- Es posible **cargarla a mano** en Ghidra…



REVERSING

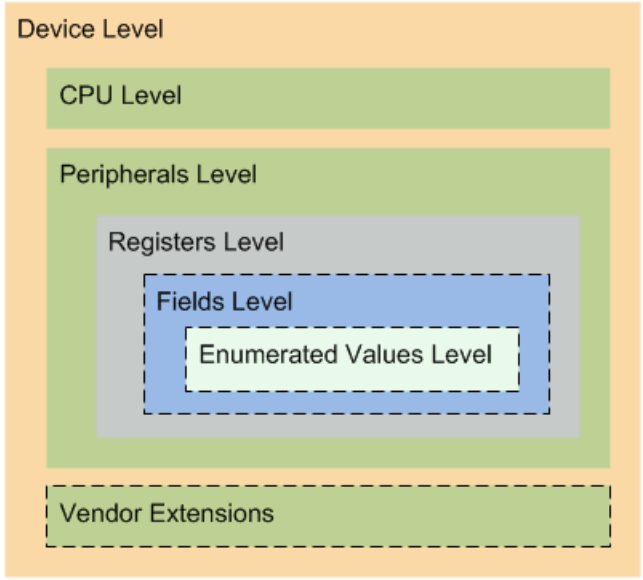
› Espressif ofrece **archivos SVD** con información de sus periféricos

https://github.com/espressif/svd

https://github.com/esp-rs/esp-pacs/tree/main/esp32/svd

› Para cargar esta información en **Ghidra** es necesario un plugin:

https://github.com/antoniovazquezblanco/GhidraSVD



CMSIS

GHIDRA

REVERSING

- El **ESP** contiene un segundo **RTOS/Kernel propietario**…
- Posible **relación** con:
  - RivieraWaves
  - CevaWaves
- **Funciones** interesantes:
  - H4 Transport Layer Init

```
4  void r_rwip_init(uint32_t error)
5
6  {
7    undefined4 uVar1;
8    int iVar2;
9    undefined *puVar3;
10   code *pcVar4;
11   char local_30;
12   uint8_t acStack_2f;
13   uint8_t length;
14
15   length = '\x01';
16   memset(&rwip_env,0,6);
17   (*r_modules_funcs_p->r_ke_init)();
18   (*r_modules_funcs_p->r_ke_mem_init)(KERNEL_MEM_ENV,&rwip_heap_env,0x22f0);
19   (*r_modules_funcs_p->r_ke_mem_init)(KERNEL_MEM_KERNEL_MSG,&rwip_heap_msg,0x1e00);
20   (*r_modules_funcs_p->r_ke_mem_init)(KERNEL_MEM_NON_RETENTION,&rwip_heap_non_ret,0x2400);
21   (*r_import_rf_phy_func_p->disable_agc_)();
22   (*r_import_rf_phy_func_p->phy_disable_agc_)(0);
23                     /* Initialize RF */
24   (*(code *)r_modules_funcs_p->r_rf_rw_init)(&rwip_rf);
25                     /* Initialize Diffie Hellman Elliptic Curve Algorithm */
26   (*(code *)r_modules_funcs_p->r_ecc_init)(0);
27                     /* Initialize H4 TL */
28   pcVar4 = (code *)r_modules_funcs_p->r_h4tl_init;
29   uVar1 = (*(code *)r_plf_funcs_p->r_rwip_eif_get)(0);
30   (*pcVar4)(0,uVar1);
```

RW

REVERSING

- Contiene funciones de **gestión de eventos.**
- Uno de esos eventos es para la **recepción de paquetes HCI**…
- Utiliza tablas de los **comandos HCI soportados**

```
2  void r_h4tl_init(byte param_1,int param_2)
3
4  {
5    int iVar1;
6    r_ke_event_callback_set *prVar2;
7
8    iVar1 = (uint)param_1 * 0x14;
9    ((h4tl_env_tag *)(h4tl_env.rx_buf + iVar1 + -0xc))->ext_if = (rwip_eif_api *)param_2;
10   (**(code **)(param_2 + 8))();
11   h4tl_env.rx_buf[iVar1 + 7] = param_1;
12   prVar2 = r_modules_funcs_p->r_ke_event_callback_set;
13   h4tl_env.rx_buf[iVar1 + 5] = '\x02';
14   (*prVar2)(KE_EVT_H4_TX,r_ke_evt_h4_tx_cb);
15   (*r_modules_funcs_p->r_ke_event_callback_set)(KE_EVT_H4_RX_HDR,r_ke_evt_h4_rx_hdr_cb);
16   (*r_modules_funcs_p->r_ke_event_callback_set)(KE_EVT_H4_RX_PAYLOAD,r_ke_evt_h4_rx_payload_cb);
17   h4tl_read_start((h4tl_env_tag *)(h4tl_env.rx_buf + iVar1 + -0xc));
18   return;
19 }
```

REVERSING

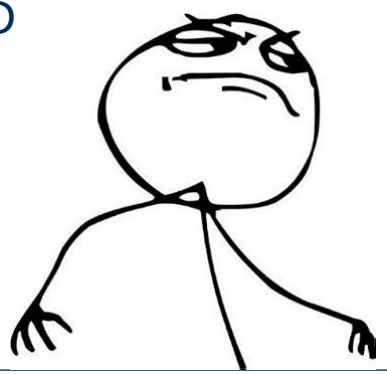
› La tabla de comandos **referencia subtablas**…

› **Clasifican por OGF** (Opcode Group Field)

```
                                hci_cmd_desc_root_tab
3ff976d4        hci_cmd_desc_tab_ref[8]
  3ff976d4      hci_cmd_desc_tab_ref                          [0]
    3ff976d4      db                    1h                      ogf
    3ff976d5      ??                    00h                     field1_0x1
    3ff976d6      uint16_t              2Dh                     nb_cmds
    3ff976d8      hci_cmd_desc_tag *    hci_cmd_desc_tab_lk_ctrl cmd_desc_tab
  3ff976dc      hci_cmd_desc_tab_ref                          [1]
    3ff976dc      db                    3h                      ogf
    3ff976dd      ??                    00h                     field1_0x1
    3ff976de      uint16_t              51h                     nb_cmds
    3ff976e0      hci_cmd_desc_tag *    hci_cmd_desc_tab_ctrl_bb cmd_desc_tab
  3ff976e4      hci_cmd_desc_tab_ref                          [2]
    3ff976e4      db                    4h                      ogf
    3ff976e5      ??                    00h                     field1_0x1
    3ff976e6      uint16_t              7h                      nb_cmds
    3ff976e8      hci_cmd_desc_tag *    hci_cmd_desc_tab_info_par cmd_desc_tab
```

REVERSING

- ¡La última entrada de la tabla referencia una subtabla interesante!
- ¡El **OGF 0x3F** está reservado para **comandos propietarios**!
- Contiene **29 comandos HCI no documentados** por ESP :D



```
3ff9770c    hci_cmd_desc_tab_ref                                    [7]
  3ff9770c    db                  3Fh                               ogf
  3ff9770d    ??                  00h                               field1_0x1
  3ff9770e    uint16_t            1Dh                               nb_cmds
  3ff97710    hci_cmd_desc_tag *  hci_cmd_desc_tab_vs               cmd_desc_tab
```

```
                        ************************************************************
                        * HCI command descriptors (OGF Vendor Specific)            *
                        ************************************************************
                        hci_cmd_desc_tab_vs
3ff97714    hci_cmd_desc_tag[29]
  3ff97714    hci_cmd_desc_tag                                      [0]
    3ff97714    uint16_t        FC01h                               opcode
    3ff97716    db              88h                                 dest_field
    3ff97717    db              6h                                  par_size_max
    3ff97718    void *          s_LBB_3ff9c950                      par_fmt
    3ff9771c    void *          hci_dbg_rd_data_cmd_cmp_evt_pk      ret_par_fmt
  3ff97720    hci_cmd_desc_tag                                      [1]
    3ff97720    uint16_t        FC02h                               opcode
    3ff97722    db              48h                                 dest_field
    3ff97723    db              88h                                 par_size_max
    3ff97724    void *          hci_dbg_wr_mem_cmd_upk              par_fmt
    3ff97728    void *          s_B_3ff9c6d6+4                      ret_par_fmt
```

REVERSING

COMANDOS OCULTOS

OPCODE	COMMAND
0xFC01	Read memory
0xFC02	Write memory
0xFC03	Delete NVDS parameter
0xFC05	Get flash ID
0xFC06	Erase flash
0xFC07	Write flash
0xFC08	Read flash
0xFC09	Read NVDS parameter
0xFC0A	Write NVDS parameter
0xFC0B	Enable/disable coexistence
0xFC0E	Send LMP packet
0xFC10	Read kernel stats
0xFC11	Platform reset
0xFC12	Read memory info

OPCODE	COMMAND
0xFC30	Register read
0xFC31	Register write
0xFC32	Set MAC address
0xFC35	Set CRC initial value
0xFC36	LLCP msgs discard
0xFC37	Reset RX count
0xFC38	Reset TX count
0xFC39	RF register read (Not implemented)
0xFC3A	RF register write (Not implemented)
0xFC3B	Set TX password
0xFC40	Set LE parameters
0xFC41	Write LE default values
0xFC42	LLCP pass through enable
0xFC43	Send LLCP packet
0xFC44	LMP msgs discard

COMANDOS OCULTOS

¿QUÉ PODEMOS HACER CON ESTOS COMANDOS?

› Cambiar la MAC del dispositivo

- Suplantar otros dispositivos
- Iniciar conexiones con dispositivos “no descubribles/conectables”.

› Gestionar tráfico LMP/LLCP!!!!

- ¡Tráfico de las capas más bajas de Bluetooth!
- ¡Permite implementar ataques que requieren hardware custom!

DEMO 2

```
// Initialize driver and get a device
UsbBluetoothManager.Init();
UsbBluetoothDevice device = UsbBluetoothManager.ListDevices()[0];
device.Open();
device.Write(new byte[] { 0x01,0x03,0x0C,0x00 }); // CMD_RESET
// Start Evil Apple Juice Attack
do {
    byte[] addr = new byte[6];
    new Random().NextBytes(addr);
    // CMD_SET_RANDOM_ADDRESS
    device.Write(new byte[] { 0x01,0x05,0x20,0x06,addr[5],addr[4],addr[3],addr[2],addr[1],addr[0] |= 0xC0 });
    // CMD_SET_ADVERTISING_PARAMETERS
    device.Write(new byte[] { 0x01,0x06,0x20,0x0f,0x00,0x02,0x00,0x08,0x02,0x02,0x01,0x8a,0xaf,0x69,0x3e, … });
    // CMD_SET_ADVERTISING_DATA
    device.Write(new byte[] { 0x01,0x08,0x20,0x20,0x1f,0x1e,0xff,0x4c,0x00,0x07,0x19,0x07,0x02,0x20, …});
    // CMD_SET_ADVERTISING_ENABLE
    device.Write(new byte[] { 0x01,0x0a,0x20,0x01,0x01 });
    Thread.Sleep(3000); // Spam for 3 secs
} while (true);
```

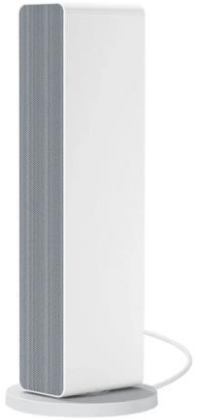
TARLOGIC
CYBERSECURITY EXPERTS

04

¿QUE MÁS SE PUEDE HACER...?

BONUS TRACK

- Mas de **1 billón (mil millones) de dispositivos usando chips ESP32** (2023)
- Uno de los **chips más usados** para conectividad **WiFi+Bluetooth low cost**, especialmente en el mundo **IoT**
- Con tantos dispositivos, ¿que más podemos hacer con estos **vendor commands**? 😈
  - Acceso directo a la **memoria de ESP32**
  - **Envío y recepción de paquetes** de bajo nivel



BONUS TRACK

› Para cualquier **dispositivo IoT con ESP32** en el que podamos **enviar comandos HCI**

- Podremos usarlo para **implementar ataques Bluetooth**
- Podremos **pivotar a otros dispositivos**
- ¡Tenemos ejecución de **código en el ESP32** a través de los co**mandos de escritura de RAM**!
- ¡**Evasión de mecanismos de verificación** de firmware del ESP32!
- El chip de comunicaciones es un buen lugar para e**sconder módulos y lograr permanencia…**
- ¿ **Rootkits/APTs** en ESP32 ? 😉 😉

CONCLUSIONES

› Existe **documentación libre y abierta** acerca de como **verificar la seguridad** de dispositivos Bluetooth

› Tenemos un **punto de partida para elaborar herramientas** con las que interactuar con Bluetooth:

- **Multiplataforma**: Linux, Windows y Mac
- **Multilenguaje**: Python y C# , fácil soportar otros lenguajes… ¿Rust?
- Integración con **Scapy**

› Desbloqueo de hardware que permite **implementar todo tipo de ataques a un muy bajo coste**…

¡Esperamos que empecéis hacer vuestras herramientas!

TARLOGIC
CYBERSECURITY EXPERTS

¡GRACIAS POR TU ATENCIÓN!

www.tarlogic.com
+34 912 919 319