IoT-Honeypot on esp32

Stefan Aistleitner, Matthäus Förster, Marlin Ortner

Motivation

Elective lecture IoT project

Option to use esp32 and explore functionalities with platformio

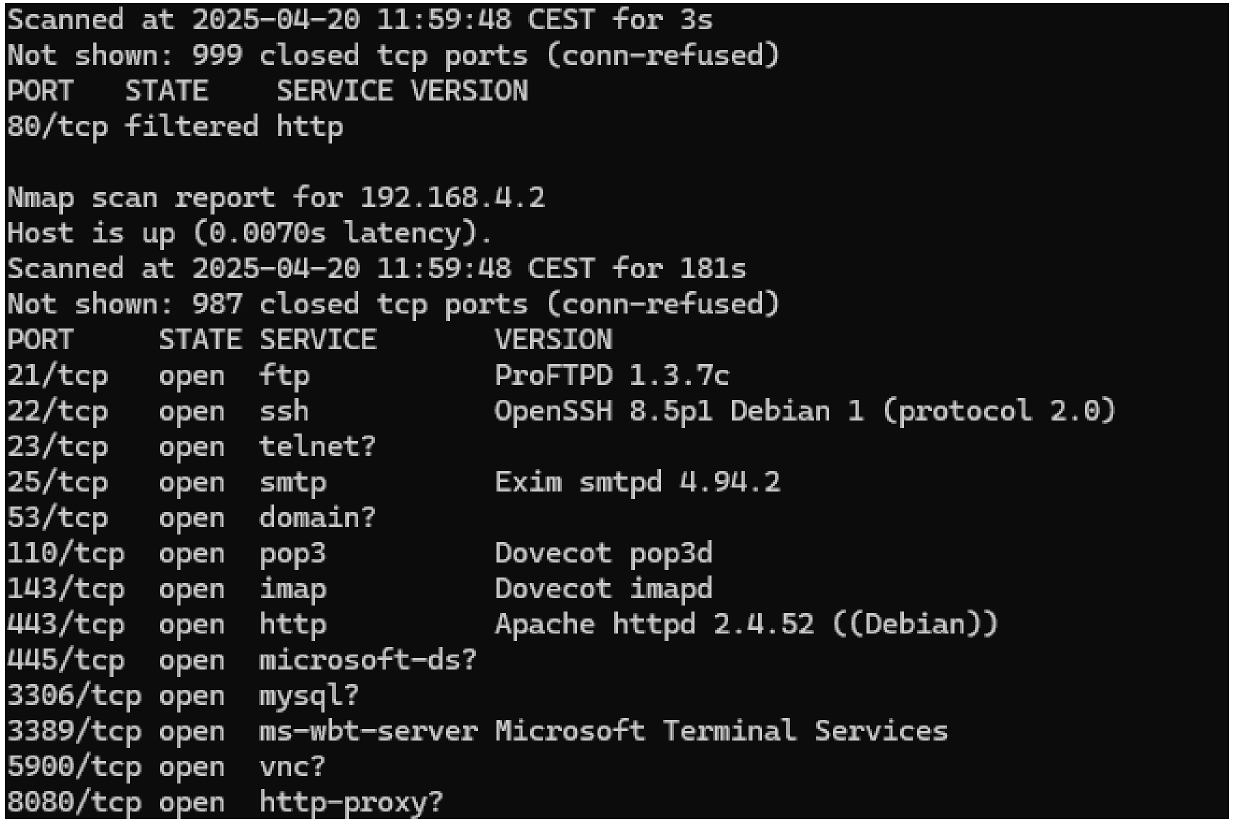
Expand experience with esp32

Related work

- Project built on existing GitHub Project NanoC6-ESP32-Honeypot from 7h30th3r0n3
- ESP32 Honeypot configuration ready to use

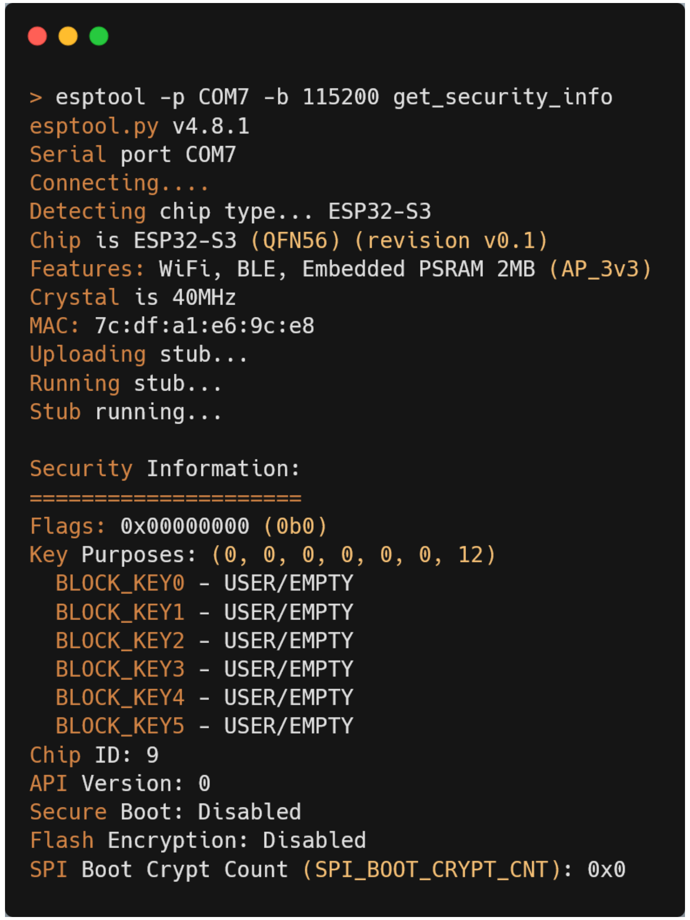
Results & showcasing

- Honeypot running on ESP32
- Discord webhook
- Web GUI for WiFi-setup
- NTP-synchronized logging
- LED blinking color feedback when accessing honeypot port
- RAW HTTP WebService



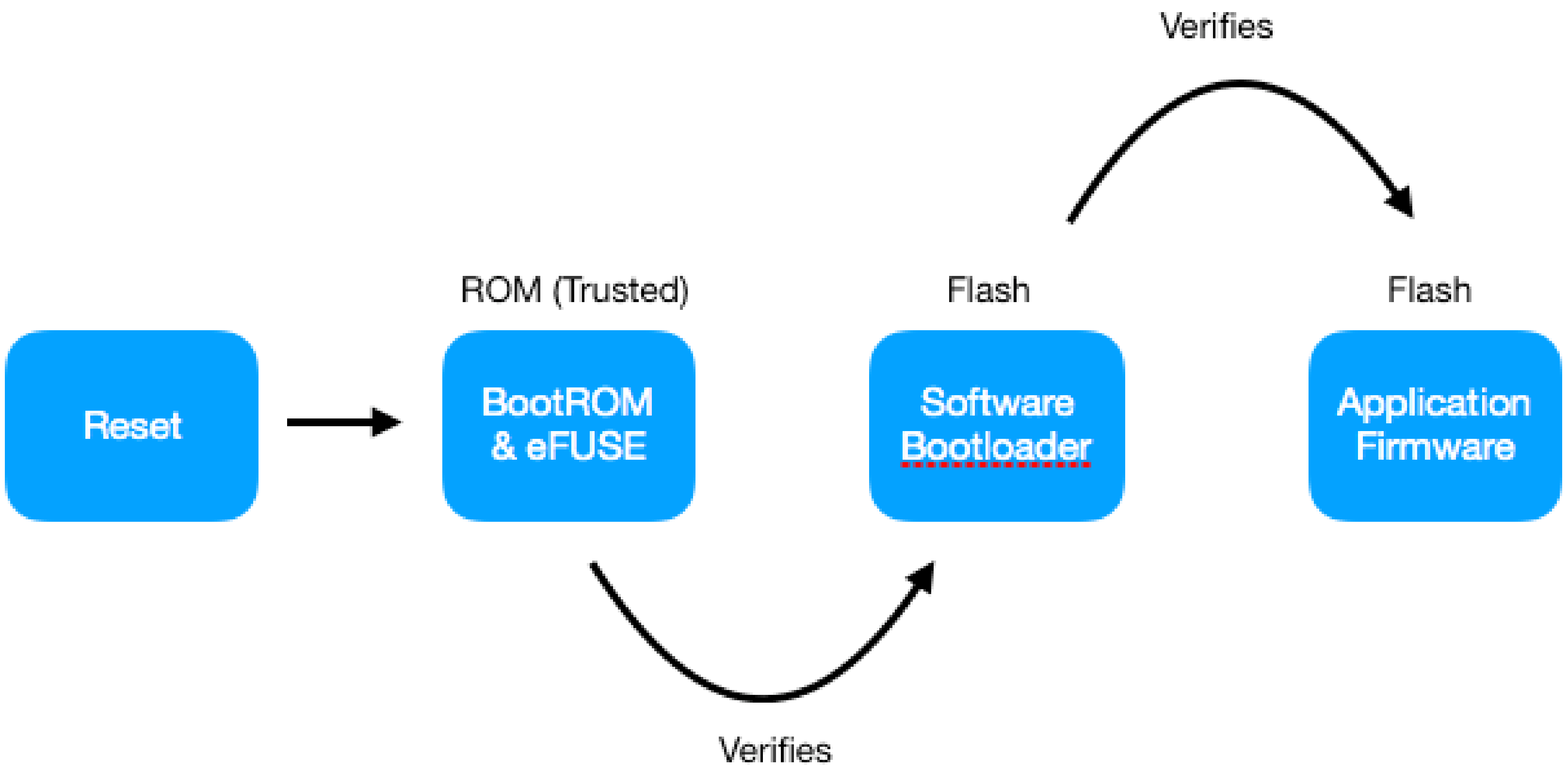
```
Scanned at 2025-04-20 11:59:48 CEST for 3s
Not shown: 999 closed tcp ports (conn-refused)
PORT   STATE    SERVICE VERSION
80/tcp filtered http

Nmap scan report for 192.168.4.2
Host is up (0.0070s latency).
Scanned at 2025-04-20 11:59:48 CEST for 181s
Not shown: 987 closed tcp ports (conn-refused)
PORT     STATE SERVICE       VERSION
21/tcp   open  ftp           ProFTPD 1.3.7c
22/tcp   open  ssh           OpenSSH 8.5p1 Debian 1 (protocol 2.0)
23/tcp   open  telnet?
25/tcp   open  smtp          Exim smtpd 4.94.2
53/tcp   open  domain?
110/tcp  open  pop3          Dovecot pop3d
143/tcp  open  imap          Dovecot imapd
443/tcp  open  http          Apache httpd 2.4.52 ((Debian))
445/tcp  open  microsoft-ds?
3306/tcp open  mysql?
3389/tcp open  ms-wbt-server Microsoft Terminal Services
5900/tcp open  vnc?
8080/tcp open  http-proxy?
```

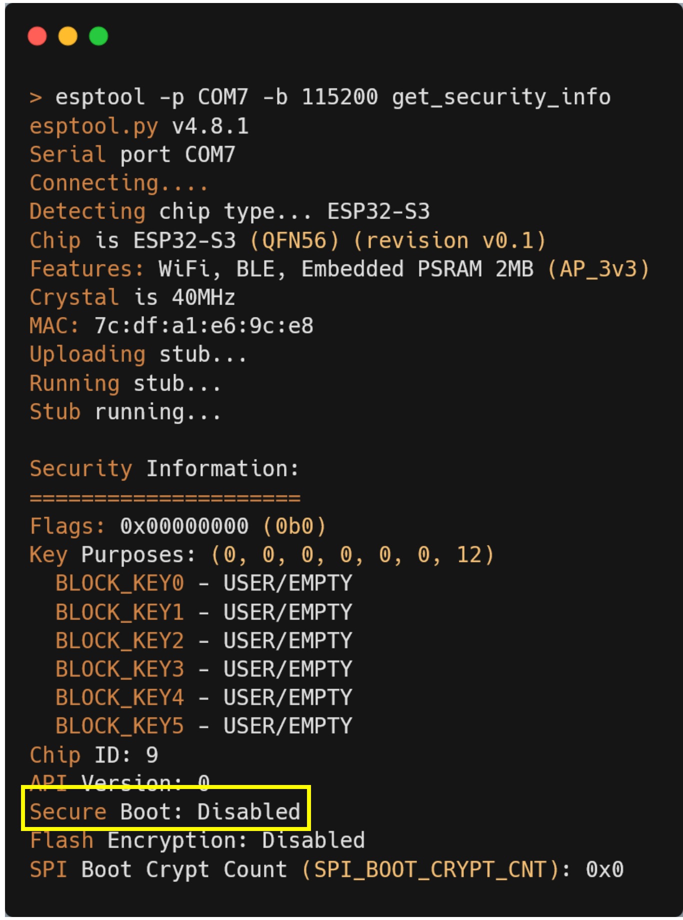
Reverse Engineering: esptool.py - securityinfo



```
> esptool -p COM7 -b 115200 get_security_info
esptool.py v4.8.1
Serial port COM7
Connecting....
Detecting chip type... ESP32-S3
Chip is ESP32-S3 (QFN56) (revision v0.1)
Features: WiFi, BLE, Embedded PSRAM 2MB (AP_3v3)
Crystal is 40MHz
MAC: 7c:df:a1:e6:9c:e8
Uploading stub...
Running stub...
Stub running...

Security Information:
=====================
Flags: 0x00000000 (0b0)
Key Purposes: (0, 0, 0, 0, 0, 0, 12)
  BLOCK_KEY0 - USER/EMPTY
  BLOCK_KEY1 - USER/EMPTY
  BLOCK_KEY2 - USER/EMPTY
  BLOCK_KEY3 - USER/EMPTY
  BLOCK_KEY4 - USER/EMPTY
  BLOCK_KEY5 - USER/EMPTY
Chip ID: 9
API Version: 0
Secure Boot: Disabled
Flash Encryption: Disabled
SPI Boot Crypt Count (SPI_BOOT_CRYPT_CNT): 0x0
```

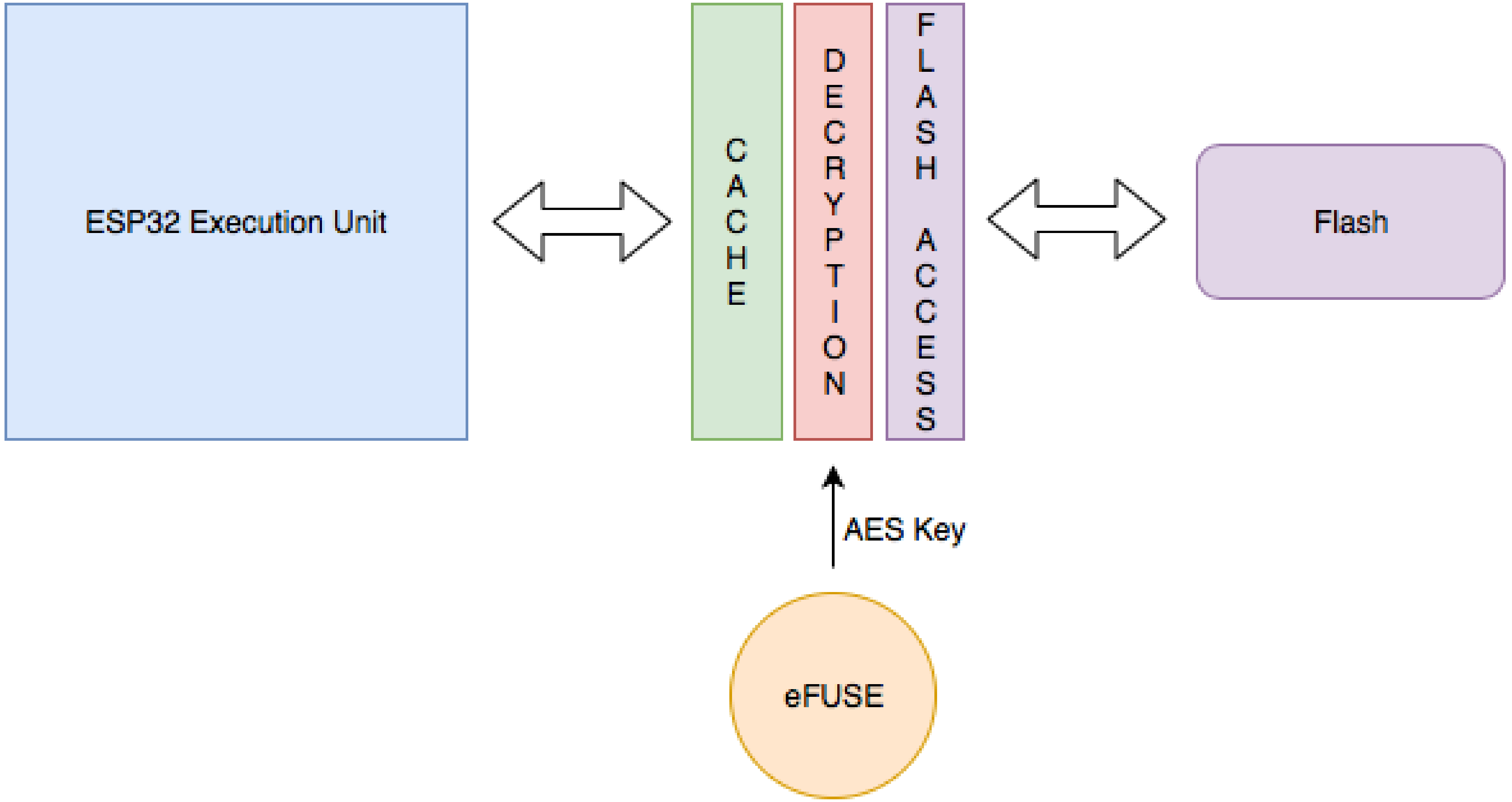
Reverse Engineering: esptool.py - securityinfo

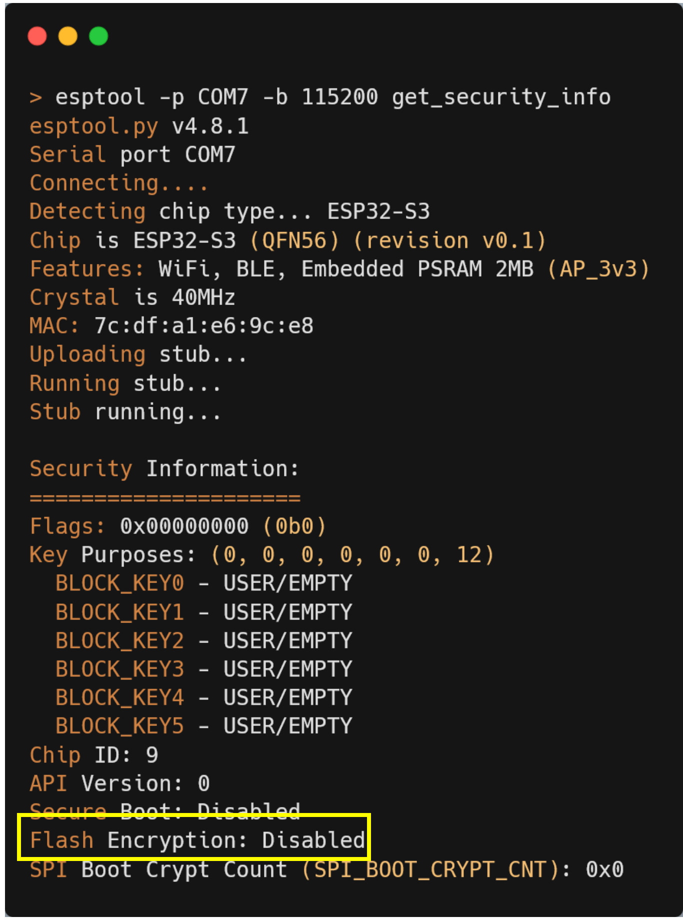




```
> esptool -p COM7 -b 115200 get_security_info
esptool.py v4.8.1
Serial port COM7
Connecting....
Detecting chip type... ESP32-S3
Chip is ESP32-S3 (QFN56) (revision v0.1)
Features: WiFi, BLE, Embedded PSRAM 2MB (AP_3v3)
Crystal is 40MHz
MAC: 7c:df:a1:e6:9c:e8
Uploading stub...
Running stub...
Stub running...

Security Information:
=====================
Flags: 0x00000000 (0b0)
Key Purposes: (0, 0, 0, 0, 0, 0, 12)
  BLOCK_KEY0 - USER/EMPTY
  BLOCK_KEY1 - USER/EMPTY
  BLOCK_KEY2 - USER/EMPTY
  BLOCK_KEY3 - USER/EMPTY
  BLOCK_KEY4 - USER/EMPTY
  BLOCK_KEY5 - USER/EMPTY
Chip ID: 9
API Version: 0
Secure Boot: Disabled
Flash Encryption: Disabled
SPI Boot Crypt Count (SPI_BOOT_CRYPT_CNT): 0x0
```

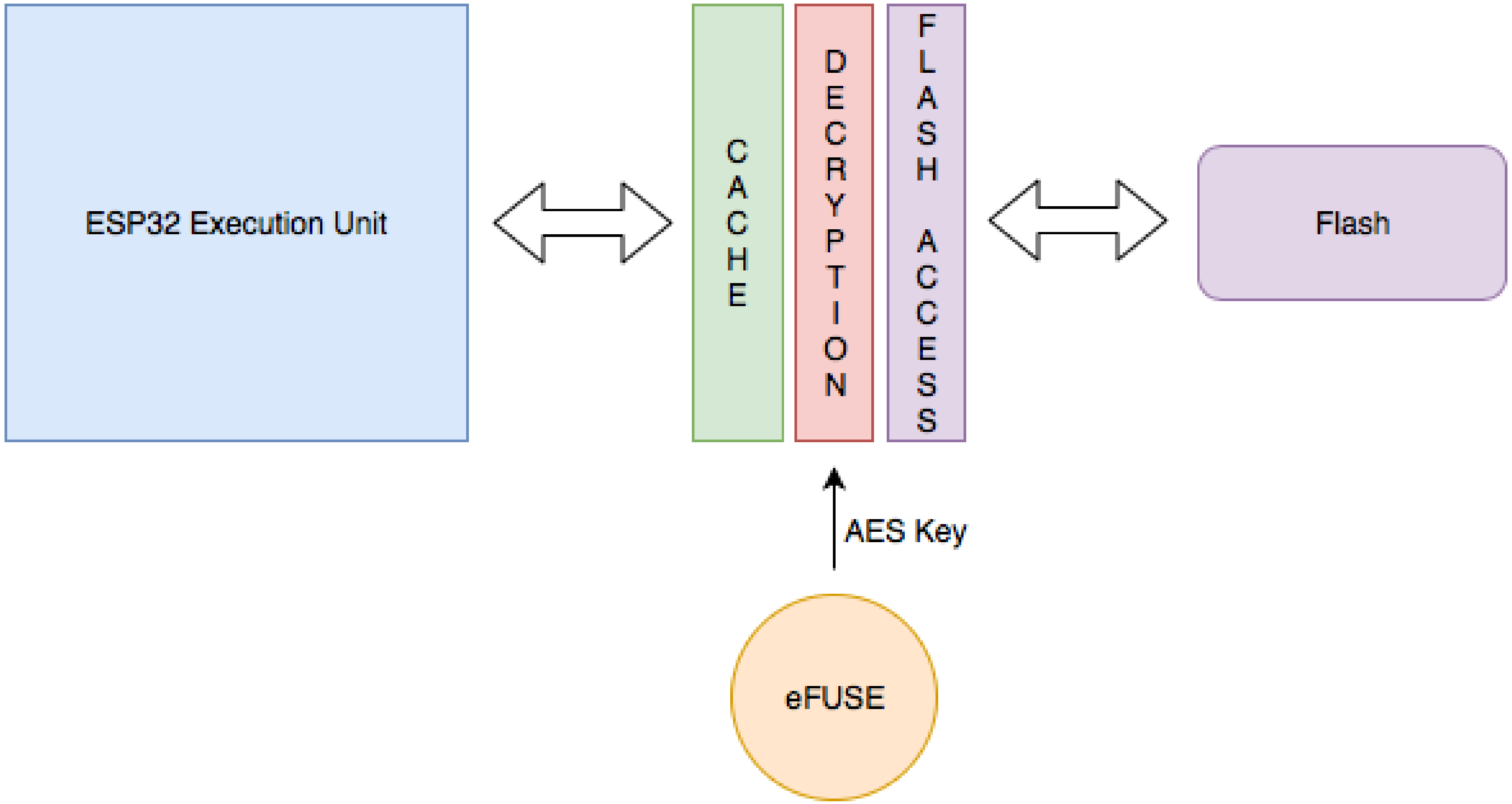
Reverse Engineering: esptool.py - securityinfo

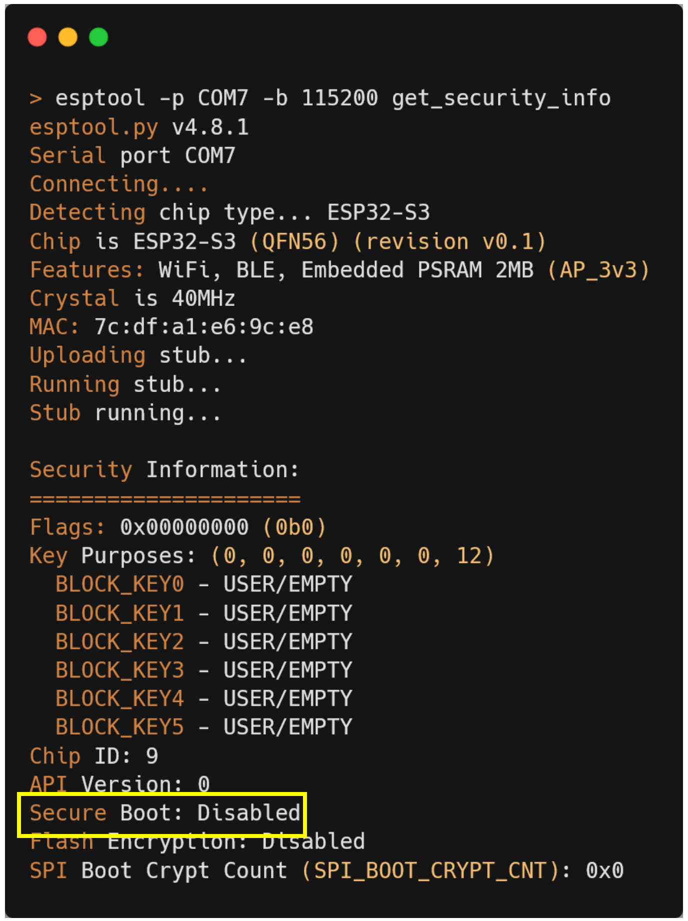




```
> esptool -p COM7 -b 115200 get_security_info
esptool.py v4.8.1
Serial port COM7
Connecting....
Detecting chip type... ESP32-S3
Chip is ESP32-S3 (QFN56) (revision v0.1)
Features: WiFi, BLE, Embedded PSRAM 2MB (AP_3v3)
Crystal is 40MHz
MAC: 7c:df:a1:e6:9c:e8
Uploading stub...
Running stub...
Stub running...

Security Information:
=====================
Flags: 0x00000000 (0b0)
Key Purposes: (0, 0, 0, 0, 0, 0, 12)
  BLOCK_KEY0 - USER/EMPTY
  BLOCK_KEY1 - USER/EMPTY
  BLOCK_KEY2 - USER/EMPTY
  BLOCK_KEY3 - USER/EMPTY
  BLOCK_KEY4 - USER/EMPTY
  BLOCK_KEY5 - USER/EMPTY
Chip ID: 9
API Version: 0
Secure Boot: Disabled
Flash Encryption: Disabled
SPI Boot Crypt Count (SPI_BOOT_CRYPT_CNT): 0x0
```

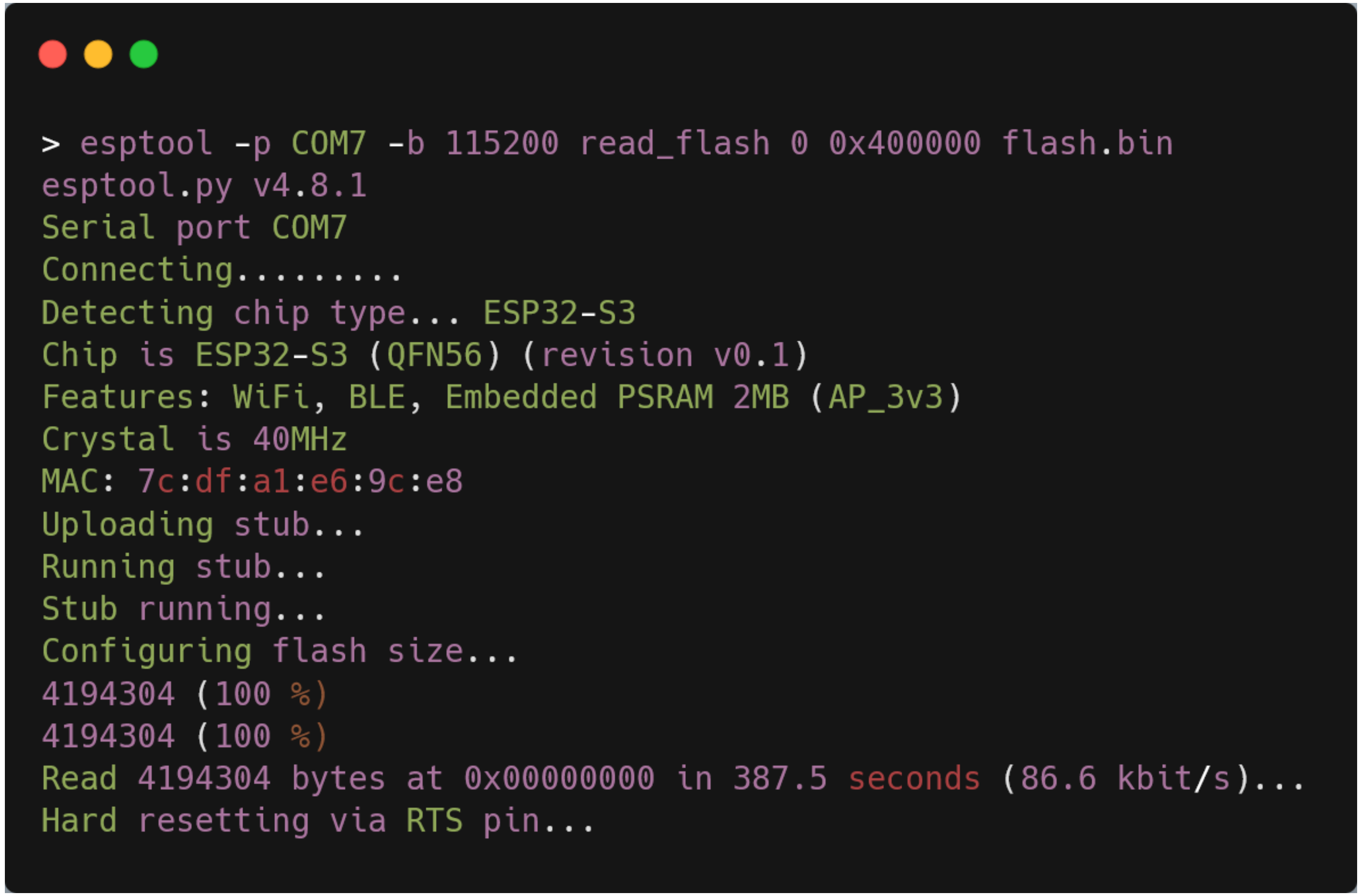
Reverse Engineering: esptool.py - securityinfo





```
> esptool -p COM7 -b 115200 get_security_info
esptool.py v4.8.1
Serial port COM7
Connecting....
Detecting chip type... ESP32-S3
Chip is ESP32-S3 (QFN56) (revision v0.1)
Features: WiFi, BLE, Embedded PSRAM 2MB (AP_3v3)
Crystal is 40MHz
MAC: 7c:df:a1:e6:9c:e8
Uploading stub...
Running stub...
Stub running...

Security Information:
=====================
Flags: 0x00000000 (0b0)
Key Purposes: (0, 0, 0, 0, 0, 0, 12)
  BLOCK_KEY0 - USER/EMPTY
  BLOCK_KEY1 - USER/EMPTY
  BLOCK_KEY2 - USER/EMPTY
  BLOCK_KEY3 - USER/EMPTY
  BLOCK_KEY4 - USER/EMPTY
  BLOCK_KEY5 - USER/EMPTY
Chip ID: 9
API Version: 0
Secure Boot: Disabled
Flash Encryption: Disabled
SPI Boot Crypt Count (SPI_BOOT_CRYPT_CNT): 0x0
```

Reverse Engineering: esptool.py - dumping

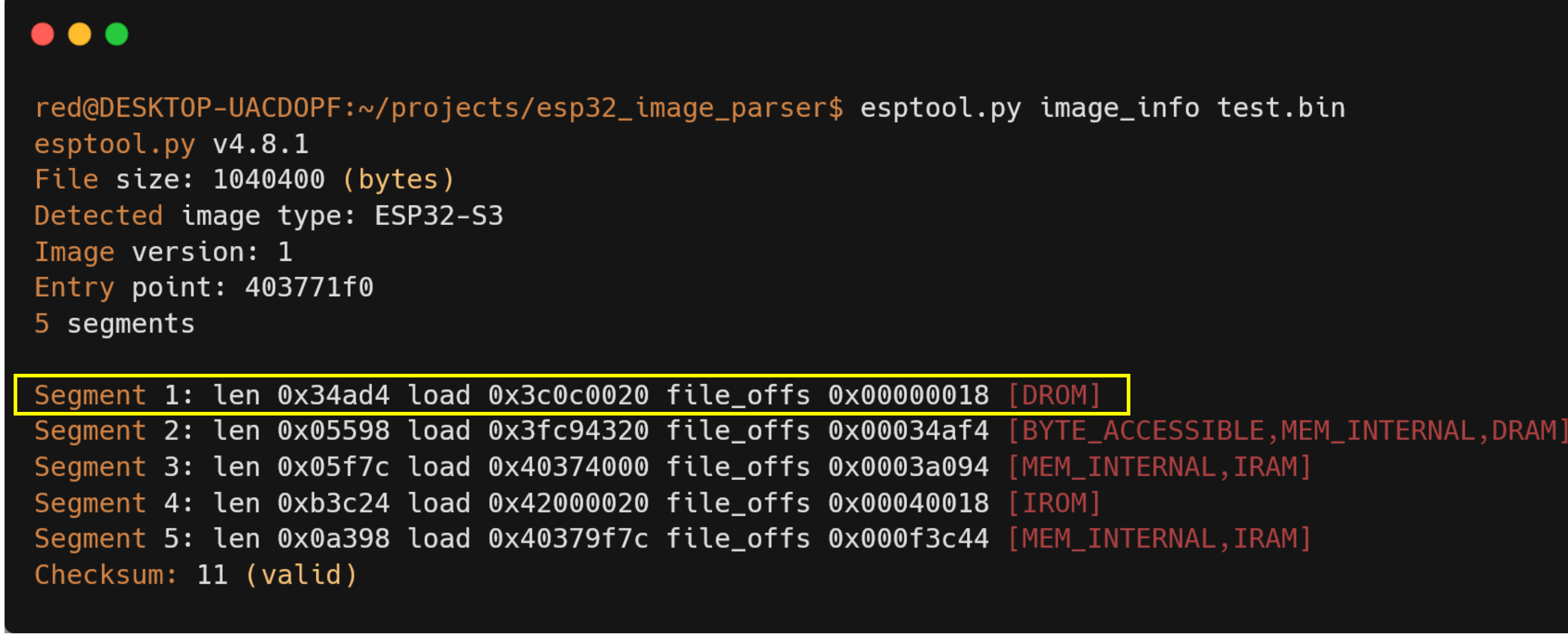


```
> esptool -p COM7 -b 115200 read_flash 0 0x400000 flash.bin
esptool.py v4.8.1
Serial port COM7
Connecting.........
Detecting chip type... ESP32-S3
Chip is ESP32-S3 (QFN56) (revision v0.1)
Features: WiFi, BLE, Embedded PSRAM 2MB (AP_3v3)
Crystal is 40MHz
MAC: 7c:df:a1:e6:9c:e8
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
4194304 (100 %)
4194304 (100 %)
Read 4194304 bytes at 0x00000000 in 387.5 seconds (86.6 kbit/s)...
Hard resetting via RTS pin...
```

ESPTool.py image_info

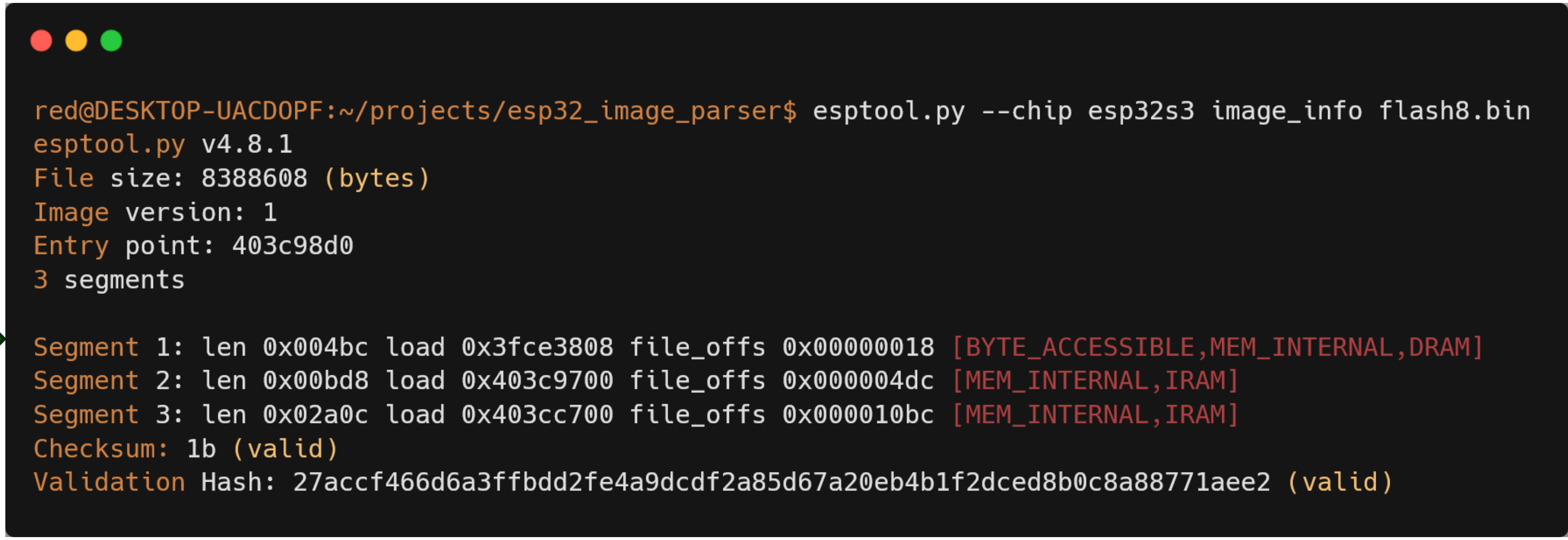
- Partitiontable

With PIO generated
ELF => .bin via esptool
elf2image
firmware.elf

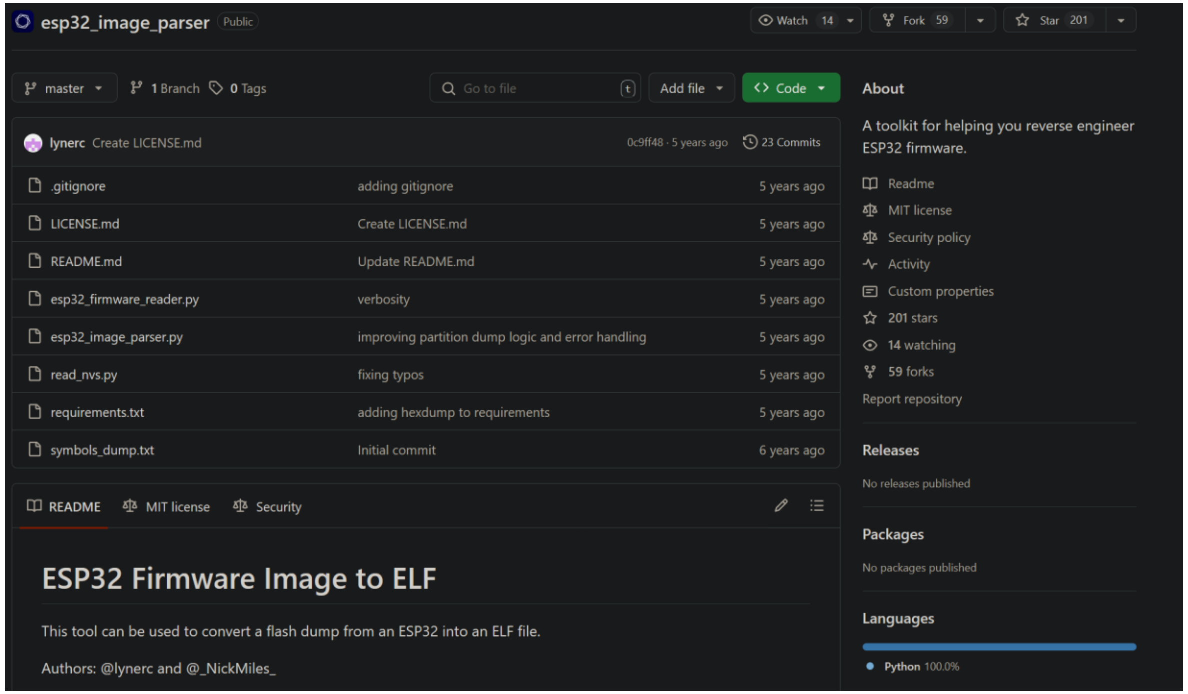


```
red@DESKTOP-UACDOPF:~/projects/esp32_image_parser$ esptool.py image_info test.bin
esptool.py v4.8.1
File size: 1040400 (bytes)
Detected image type: ESP32-S3
Image version: 1
Entry point: 403771f0
5 segments

Segment 1: len 0x34ad4 load 0x3c0c0020 file_offs 0x00000018 [DROM]
Segment 2: len 0x05598 load 0x3fc94320 file_offs 0x00034af4 [BYTE_ACCESSIBLE,MEM_INTERNAL,DRAM]
Segment 3: len 0x05f7c load 0x40374000 file_offs 0x0003a094 [MEM_INTERNAL,IRAM]
Segment 4: len 0xb3c24 load 0x42000020 file_offs 0x00040018 [IROM]
Segment 5: len 0x0a398 load 0x40379f7c file_offs 0x000f3c44 [MEM_INTERNAL,IRAM]
Checksum: 11 (valid)
```

Dumped data via esptool



```
red@DESKTOP-UACDOPF:~/projects/esp32_image_parser$ esptool.py --chip esp32s3 image_info flash8.bin
esptool.py v4.8.1
File size: 8388608 (bytes)
Image version: 1
Entry point: 403c98d0
3 segments

Segment 1: len 0x004bc load 0x3fce3808 file_offs 0x00000018 [BYTE_ACCESSIBLE,MEM_INTERNAL,DRAM]
Segment 2: len 0x00bd8 load 0x403c9700 file_offs 0x000004dc [MEM_INTERNAL,IRAM]
Segment 3: len 0x02a0c load 0x403cc700 file_offs 0x000010bc [MEM_INTERNAL,IRAM]
Checksum: 1b (valid)
Validation Hash: 27accf466d6a3ffbdd2fe4a9dcdf2a85d67a20eb4b1f2dced8b0c8a88771aee2 (valid)
```



esp32_image_parser Public

Watch 14 · Fork 59 · Star 201

master · 1 Branch · 0 Tags · Go to file · Add file · Code

lynerc Create LICENSE.md · 0c9ff48 · 5 years ago · 23 Commits

.gitignore	adding gitignore	5 years ago
LICENSE.md	Create LICENSE.md	5 years ago
README.md	Update README.md	5 years ago
esp32_firmware_reader.py	verbosity	5 years ago
esp32_image_parser.py	improving partition dump logic and error handling	5 years ago
read_nvs.py	fixing typos	5 years ago
requirements.txt	adding hexdump to requirements	5 years ago
symbols_dump.txt	Initial commit	6 years ago

README · MIT license · Security

ESP32 Firmware Image to ELF

This tool can be used to convert a flash dump from an ESP32 into an ELF file.

Authors: @lynerc and @_NickMiles_

About

A toolkit for helping you reverse engineer ESP32 firmware.

Readme
MIT license
Security policy
Activity
Custom properties
201 stars
14 watching
59 forks
Report repository

Releases

No releases published

Packages

No packages published

Languages

Python 100.0%

https://github.com/tenable/esp32_image_parser

https://www.youtube.com/watch?v=w4_3vwN_2dI

Extracting an ELF From an ESP32 - Chris Lyne and Nick Miles (Shmoocon 2020) [42:39]

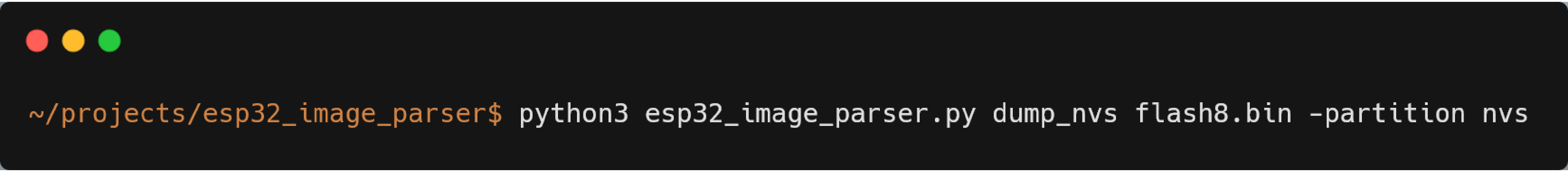
Nothing worked

- Could not map the affirmentioned segments and dump a ELF file to reverse engineer

- Github Issues
- Overwriting Python Script
- Maybe it’s the flash download? Do it again
- Change segment mapping

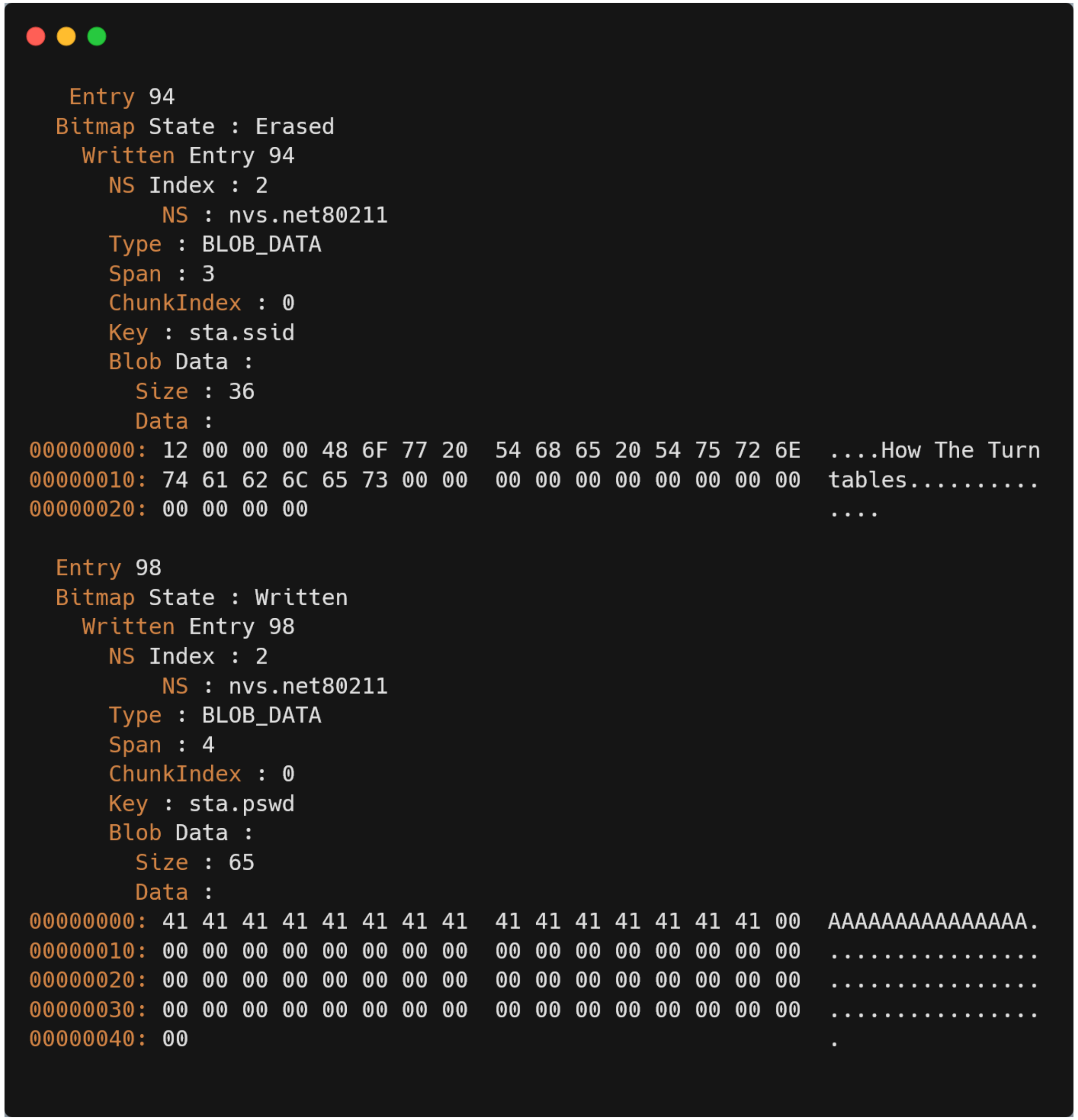
New approach: NVS

- Non-Volatile Storage Library
- designed to store key-value pairs in flash



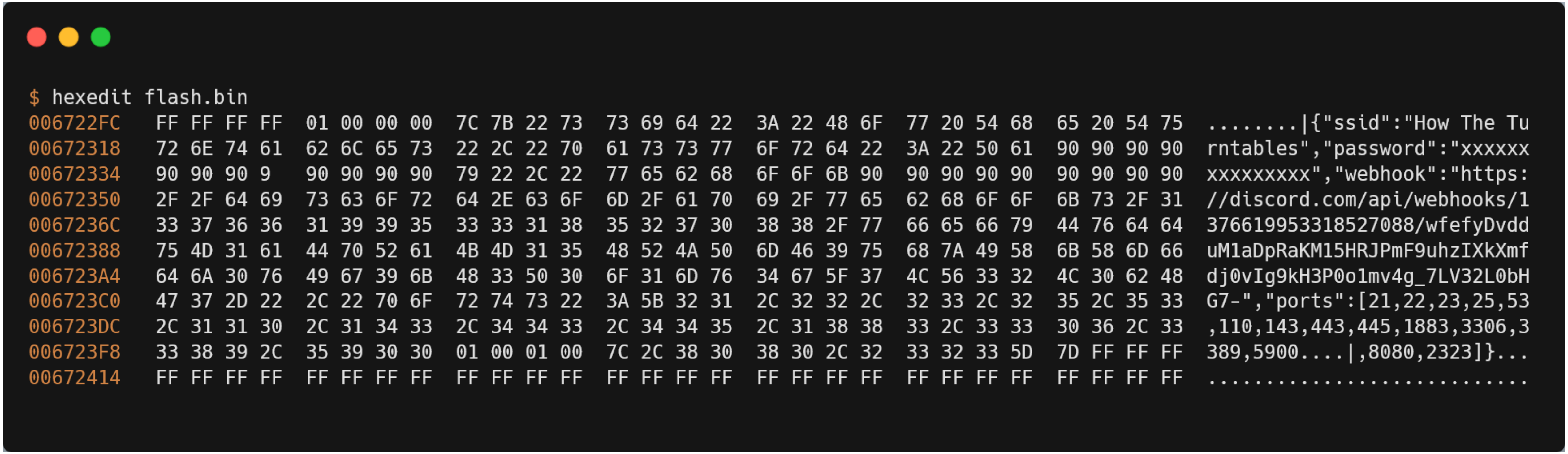
```
~/projects/esp32_image_parser$ python3 esp32_image_parser.py dump_nvs flash8.bin -partition nvs
```

AP SSID + Password in Non-Volatile Storage (dumped flash)



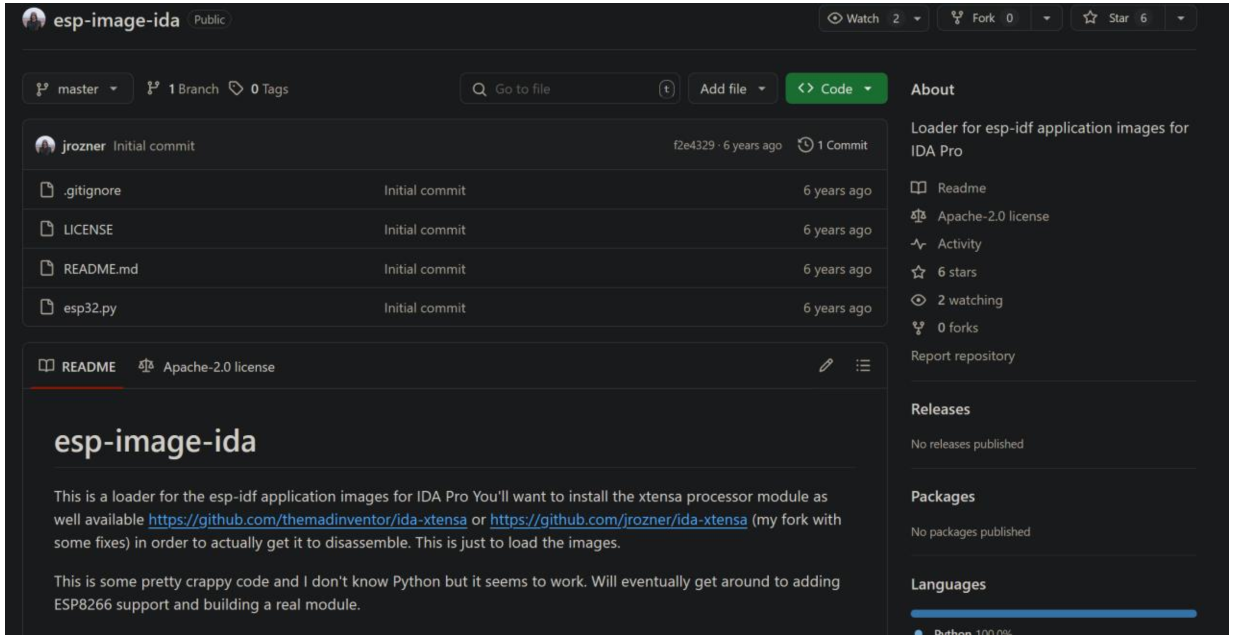
```
   Entry 94
  Bitmap State : Erased
    Written Entry 94
      NS Index : 2
          NS : nvs.net80211
      Type : BLOB_DATA
      Span : 3
      ChunkIndex : 0
      Key : sta.ssid
      Blob Data :
        Size : 36
        Data :
00000000: 12 00 00 00 48 6F 77 20  54 68 65 20 54 75 72 6E  ....How The Turn
00000010: 74 61 62 6C 65 73 00 00  00 00 00 00 00 00 00 00  tables..........
00000020: 00 00 00 00                                       ....

  Entry 98
  Bitmap State : Written
    Written Entry 98
      NS Index : 2
          NS : nvs.net80211
      Type : BLOB_DATA
      Span : 4
      ChunkIndex : 0
      Key : sta.pswd
      Blob Data :
        Size : 65
        Data :
00000000: 41 41 41 41 41 41 41 41  41 41 41 41 41 41 41 00  AAAAAAAAAAAAAAA.
00000010: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000020: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000030: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000040: 00                                                .
```

Also in hexedit (dumped flash)



```
$ hexedit flash.bin
006722FC   FF FF FF FF  01 00 00 00  7C 7B 22 73  73 69 64 22  3A 22 48 6F  77 20 54 68  65 20 54 75  ........|{"ssid":"How The Tu
00672318   72 6E 74 61  62 6C 65 73  22 2C 22 70  61 73 73 77  6F 72 64 22  3A 22 50 61  90 90 90 90  rntables","password":"xxxxxx
00672334   90 90 90 9   90 90 90 90  79 22 2C 22  77 65 62 68  6F 6F 6B 90  90 90 90 90  90 90 90 90  xxxxxxxxx","webhook":"https:
00672350   2F 2F 64 69  73 63 6F 72  64 2E 63 6F  6D 2F 61 70  69 2F 77 65  62 68 6F 6F  6B 73 2F 31  //discord.com/api/webhooks/1
0067236C   33 37 36 36  31 39 39 35  33 33 31 38  35 32 37 30  38 38 2F 77  66 65 66 79  44 76 64 64  376619953318527088/wfefyDvdd
00672388   75 4D 31 61  44 70 52 61  4B 4D 31 35  48 52 4A 50  6D 46 39 75  68 7A 49 58  6B 58 6D 66  uM1aDpRaKM15HRJPmF9uhzIXkXmf
006723A4   64 6A 30 76  49 67 39 6B  48 33 50 30  6F 31 6D 76  34 67 5F 37  4C 56 33 32  4C 30 62 48  dj0vIg9kH3P0o1mv4g_7LV32L0bH
006723C0   47 37 2D 22  2C 22 70 6F  72 74 73 22  3A 5B 32 31  2C 32 32 2C  32 33 2C 32  35 2C 35 33  G7-","ports":[21,22,23,25,53
006723DC   2C 31 31 30  2C 31 34 33  2C 34 34 33  2C 34 34 35  2C 31 38 38  33 2C 33 33  30 36 2C 33  ,110,143,443,445,1883,3306,3
006723F8   33 38 39 2C  35 39 30 30  01 00 01 00  7C 2C 38 30  38 30 2C 32  33 32 33 5D  7D FF FF FF  389,5900....|,8080,2323]}...
00672414   FF FF FF FF  FF FF FF FF  FF FF FF FF  FF FF FF FF  FF FF FF FF  FF FF FF FF  FF FF FF FF  ............................
```

Reverse Engineering: IDA



jrozner Initial commit f2e4329 · 6 years ago 1 Commit

.gitignore	Initial commit	6 years ago
LICENSE	Initial commit	6 years ago
README.md	Initial commit	6 years ago
esp32.py	Initial commit	6 years ago

About

Loader for esp-idf application images for IDA Pro

Readme
Apache-2.0 license
Activity
6 stars
2 watching
0 forks
Report repository

Releases

No releases published

Packages

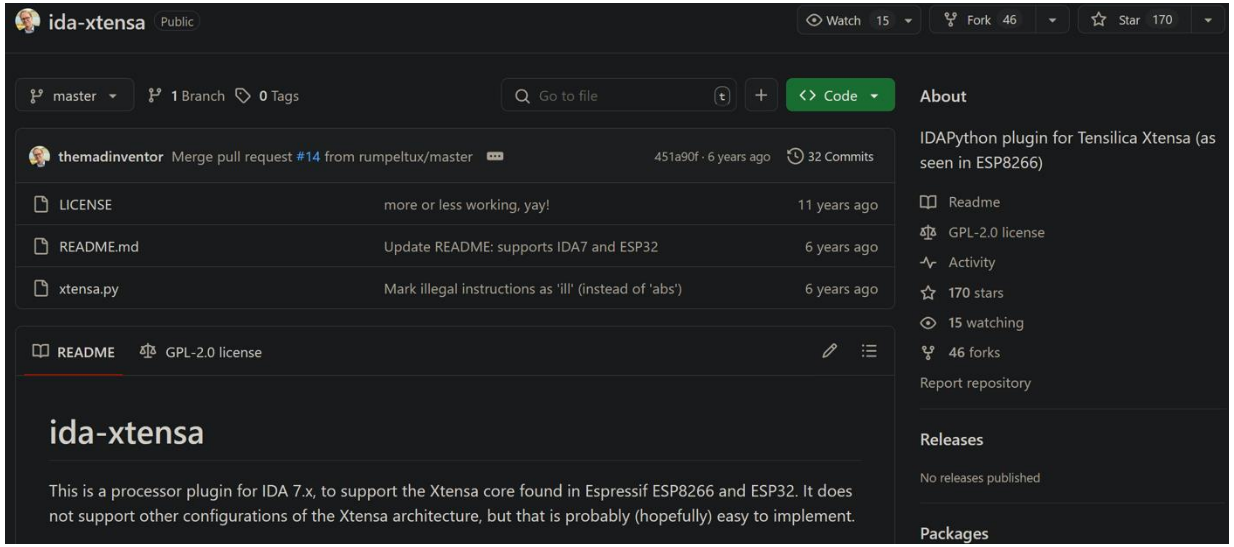
No packages published

Languages

esp-image-ida

This is a loader for the esp-idf application images for IDA Pro You'll want to install the xtensa processor module as well available https://github.com/themadinventor/ida-xtensa or https://github.com/jrozner/ida-xtensa (my fork with some fixes) in order to actually get it to disassemble. This is just to load the images.

This is some pretty crappy code and I don't know Python but it seems to work. Will eventually get around to adding ESP8266 support and building a real module.



themadinventor Merge pull request #14 from rumpeltux/master 451a90f · 6 years ago 32 Commits

LICENSE	more or less working, yay!	11 years ago
README.md	Update README: supports IDA7 and ESP32	6 years ago
xtensa.py	Mark illegal instructions as 'ill' (instead of 'abs')	6 years ago

About

IDAPython plugin for Tensilica Xtensa (as seen in ESP8266)

Readme
GPL-2.0 license
Activity
170 stars
15 watching
46 forks
Report repository

Releases

No releases published

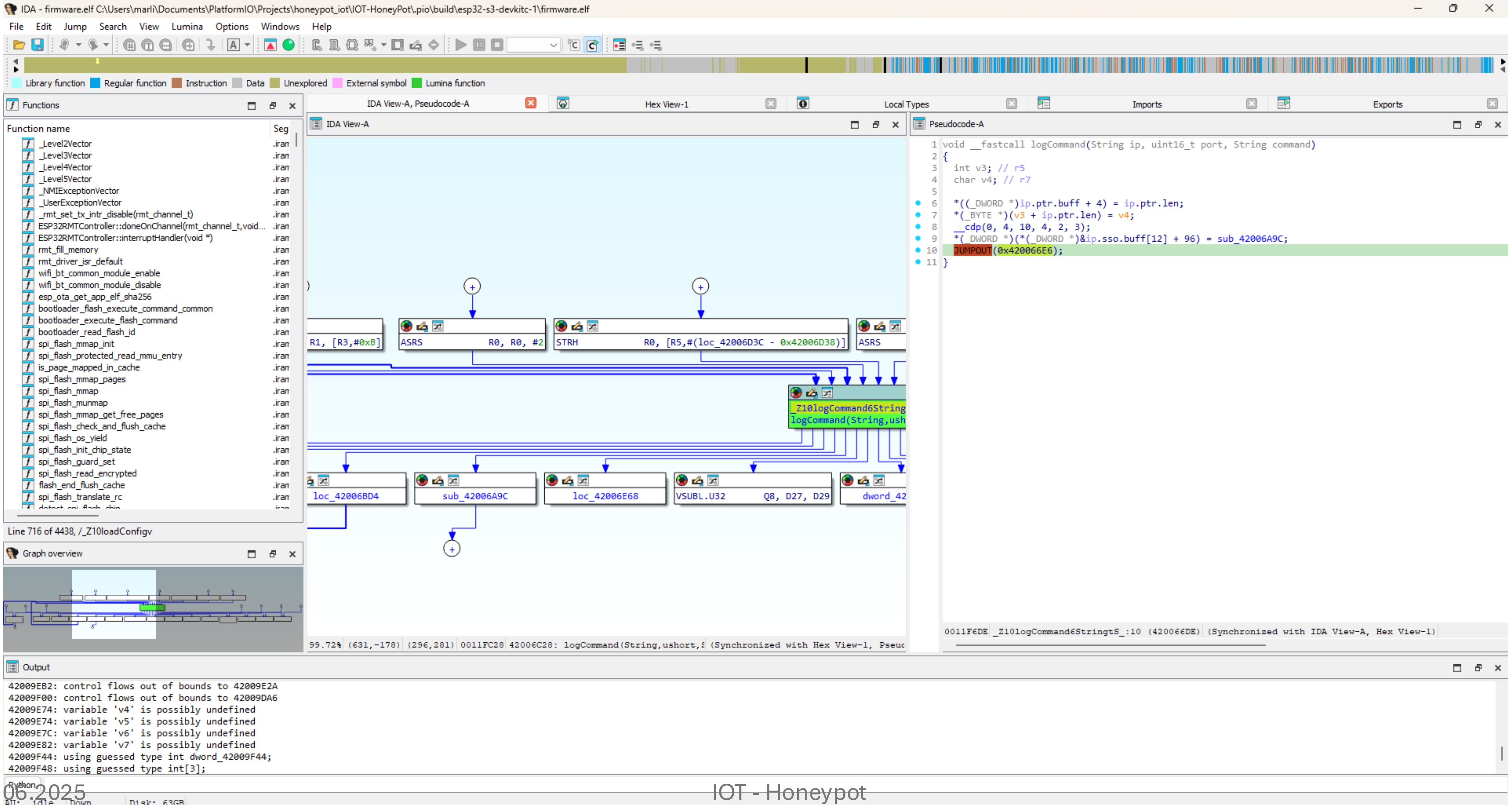
Packages

ida-xtensa

This is a processor plugin for IDA 7.x, to support the Xtensa core found in Espressif ESP8266 and ESP32. It does not support other configurations of the Xtensa architecture, but that is probably (hopefully) easy to implement.

https://github.com/jrozner/esp-image-ida https://github.com/themadinventor/ida-xtensa

Reverse Engineering: IDA



IDA - firmware.elf C:\Users\marli\Documents\PlatformIO\Projects\honeypot_iot\IOT-HoneyPot\.pio\build\esp32-s3-devkitc-1\firmware.elf

Function name	Seg
_Level2Vector	.iram
_Level3Vector	.iram
_Level4Vector	.iram
_Level5Vector	.iram
_NMIExceptionVector	.iram
_UserExceptionVector	.iram
_rmt_set_tx_intr_disable(rmt_channel_t)	.iram
ESP32RMTController::doneOnChannel(rmt_channel_t,void...	.iram
ESP32RMTController::interruptHandler(void *)	.iram
rmt_fill_memory	.iram
rmt_driver_isr_default	.iram
wifi_bt_common_module_enable	.iram
wifi_bt_common_module_disable	.iram
esp_ota_get_app_elf_sha256	.iram
bootloader_flash_execute_command_common	.iram
bootloader_execute_flash_command	.iram
bootloader_read_flash_id	.iram
spi_flash_mmap_init	.iram
spi_flash_protected_read_mmu_entry	.iram
is_page_mapped_in_cache	.iram
spi_flash_mmap_pages	.iram
spi_flash_mmap	.iram
spi_flash_munmap	.iram
spi_flash_mmap_get_free_pages	.iram
spi_flash_check_and_flush_cache	.iram
spi_flash_os_yield	.iram
spi_flash_init_chip_state	.iram
spi_flash_guard_set	.iram
spi_flash_read_encrypted	.iram
flash_end_flush_cache	.iram
spi_flash_translate_rc	.iram

Line 716 of 4438, /_Z10loadConfigv

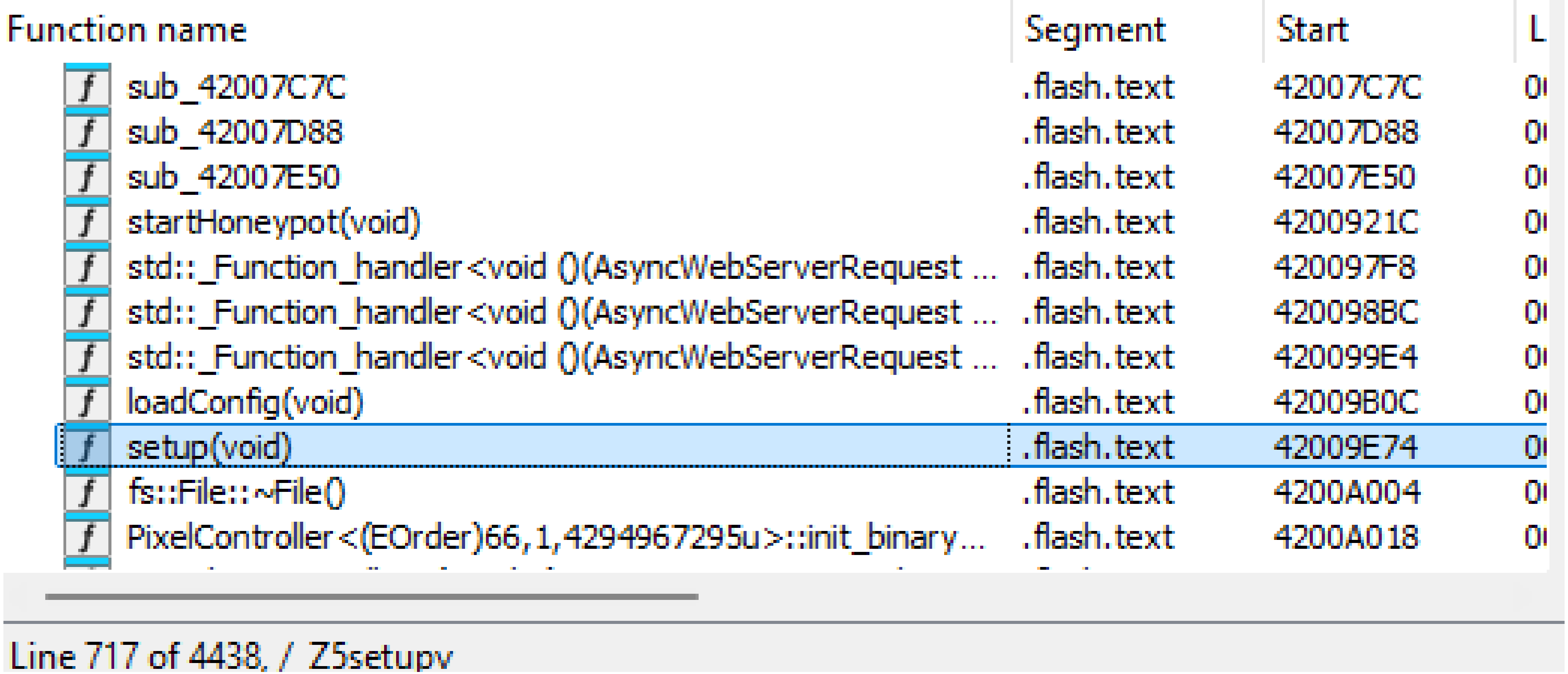
```
void __fastcall logCommand(String ip, uint16_t port, String command)
{
  int v3; // r5
  char v4; // r7

  *((_DWORD *)ip.ptr.buff + 4) = ip.ptr.len;
  *(_BYTE *)(v3 + ip.ptr.len) = v4;
  __cdp(0, 4, 10, 4, 2, 3);
  *(_DWORD *)(*(_DWORD *)&ip.sso.buff[12] + 96) = sub_42006A9C;
  JUMPOUT(0x420066E6);
}
```

```
99.72% (631,-178) (296,281) 0011FC28 42006C28: logCommand(String,ushort,S (Synchronized with Hex View-1, Pseuc
```

```
0011F6DE _Z10logCommand6StringtS_:10 (420066DE) (Synchronized with IDA View-A, Hex View-1)
```

```
42009EB2: control flows out of bounds to 42009E2A
42009F00: control flows out of bounds to 42009DA6
42009E74: variable 'v4' is possibly undefined
42009E74: variable 'v5' is possibly undefined
42009E7C: variable 'v6' is possibly undefined
42009E82: variable 'v7' is possibly undefined
42009F44: using guessed type int dword_42009F44;
42009F48: using guessed type int[3];
```

```
AU: idle Down Disk: 63GB
```



Function name	Segment	Start	L
sub_42007C7C	.flash.text	42007C7C	0
sub_42007D88	.flash.text	42007D88	0
sub_42007E50	.flash.text	42007E50	0
startHoneypot(void)	.flash.text	4200921C	0
std::_Function_handler<void ()(AsyncWebServerRequest ...	.flash.text	420097F8	0
std::_Function_handler<void ()(AsyncWebServerRequest ...	.flash.text	420098BC	0
std::_Function_handler<void ()(AsyncWebServerRequest ...	.flash.text	420099E4	0
loadConfig(void)	.flash.text	42009B0C	0
setup(void)	.flash.text	42009E74	0
fs::File::~File()	.flash.text	4200A004	0
PixelController<(EOrder)66,1,4294967295u>::init_binary...	.flash.text	4200A018	0

Line 717 of 4438, / Z5setupv

Image analysis in IDA

- A lot of function (4438)
- From a ctf perspective: either GUI application or a lot of libaries have been used

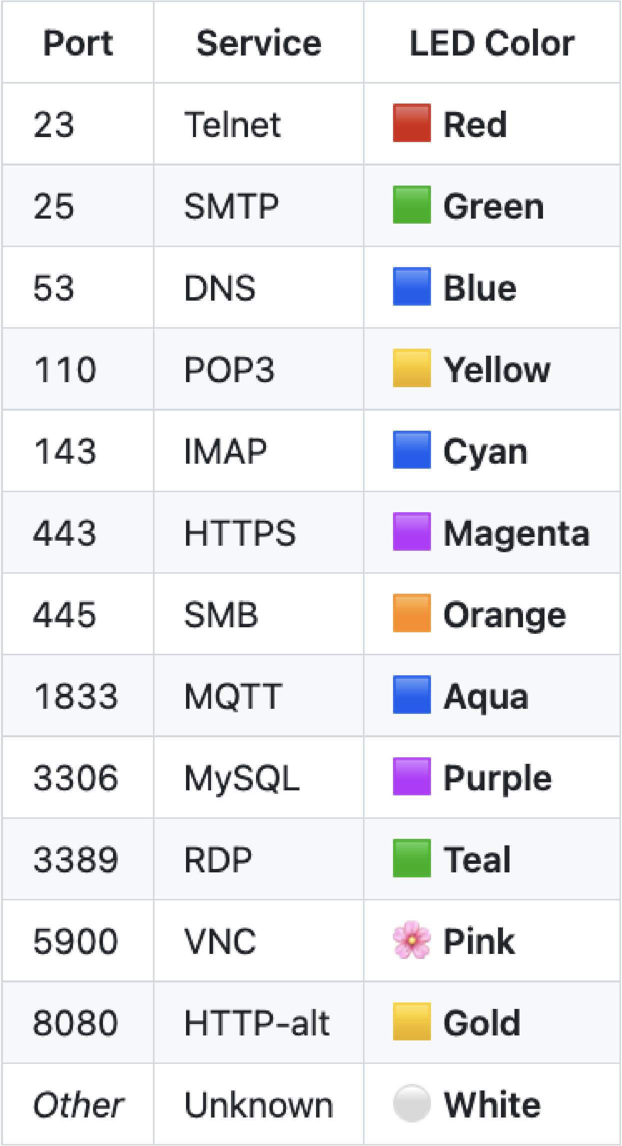
Future Work

- Add more honeypot features
- Extend implemented shell usage
- Implement more logging features
- User-friendly extraction of logs on the Filesystem
- Practical setup and collect logs

Thank you for your attention!

DEMO

SSID:	Wifi
Passwort:	pleaseEnter
IP:	192.168.137.193



Port	Service	LED Color
23	Telnet	🟥 **Red**
25	SMTP	🟩 **Green**
53	DNS	🟦 **Blue**
110	POP3	🟨 **Yellow**
143	IMAP	🟦 **Cyan**
443	HTTPS	🟪 **Magenta**
445	SMB	🟧 **Orange**
1833	MQTT	🟦 **Aqua**
3306	MySQL	🟪 **Purple**
3389	RDP	🟩 **Teal**
5900	VNC	🌸 **Pink**
8080	HTTP-alt	🟨 **Gold**
Other	Unknown	⚪ **White**