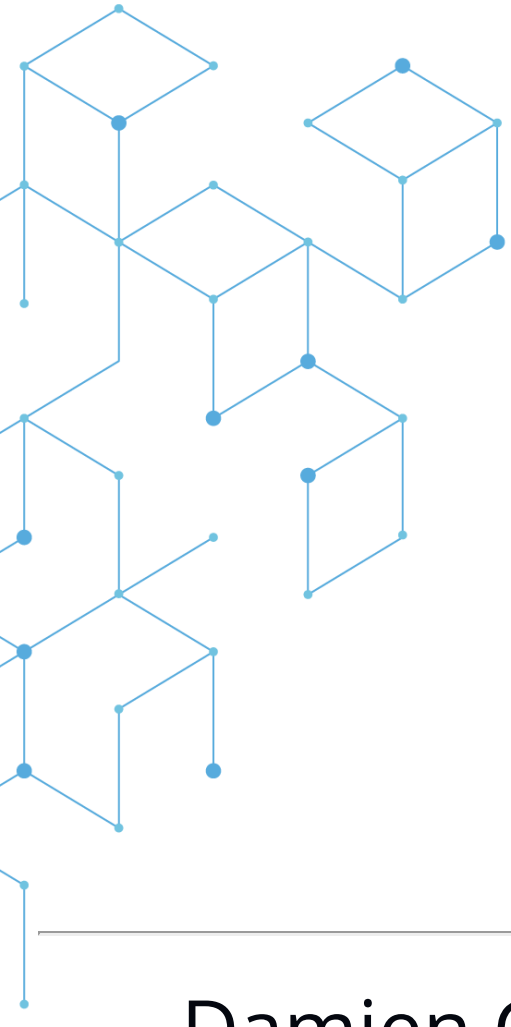


Quarkslab

Binbloom Reloaded

Damien Cauquil | virtualabs | PTS 2022

Introduction

Who am I ?

- Security Engineer (Quarkslab)
- Hardware/software RE
- Bluetooth Low Energy, sometimes 😏

A bit of context

- Guillaume Heilles (**PapaZours**) published **binbloom** in 2020
- In 2021, he gave me **some insights of binbloom** before leaving
- A month after, I stumbled upon an **unknown firmware** ...

Unknown firmware, really ?

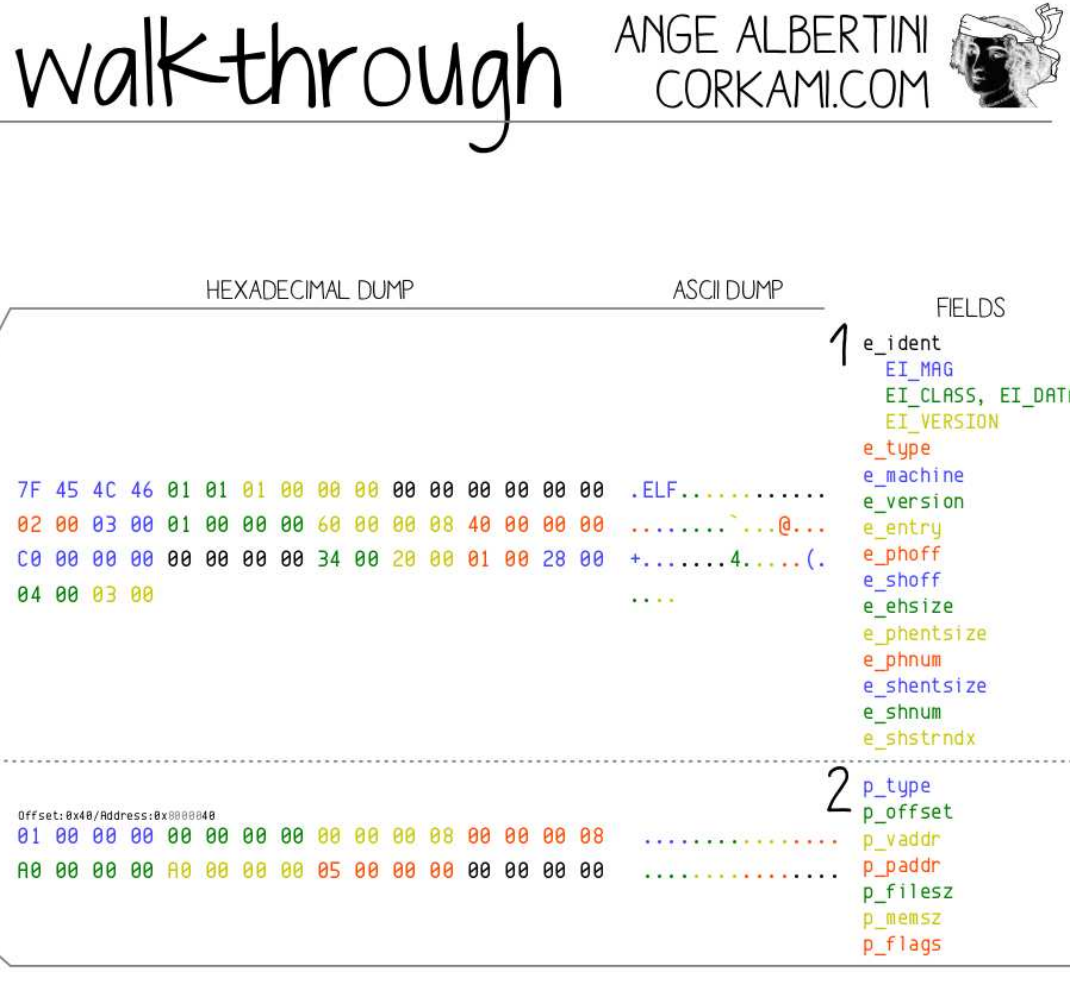
- Designed for **AArch64**
- **64-bit** architecture
- No idea of its **base address**
- **Not supported** by binbloom !

Wait, what's a base address ?

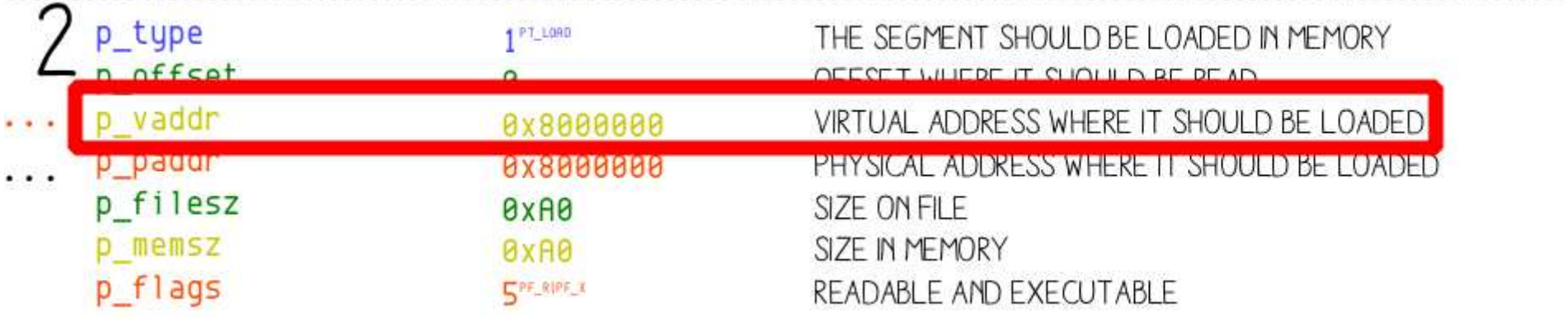
ELF file format

ble walk-through

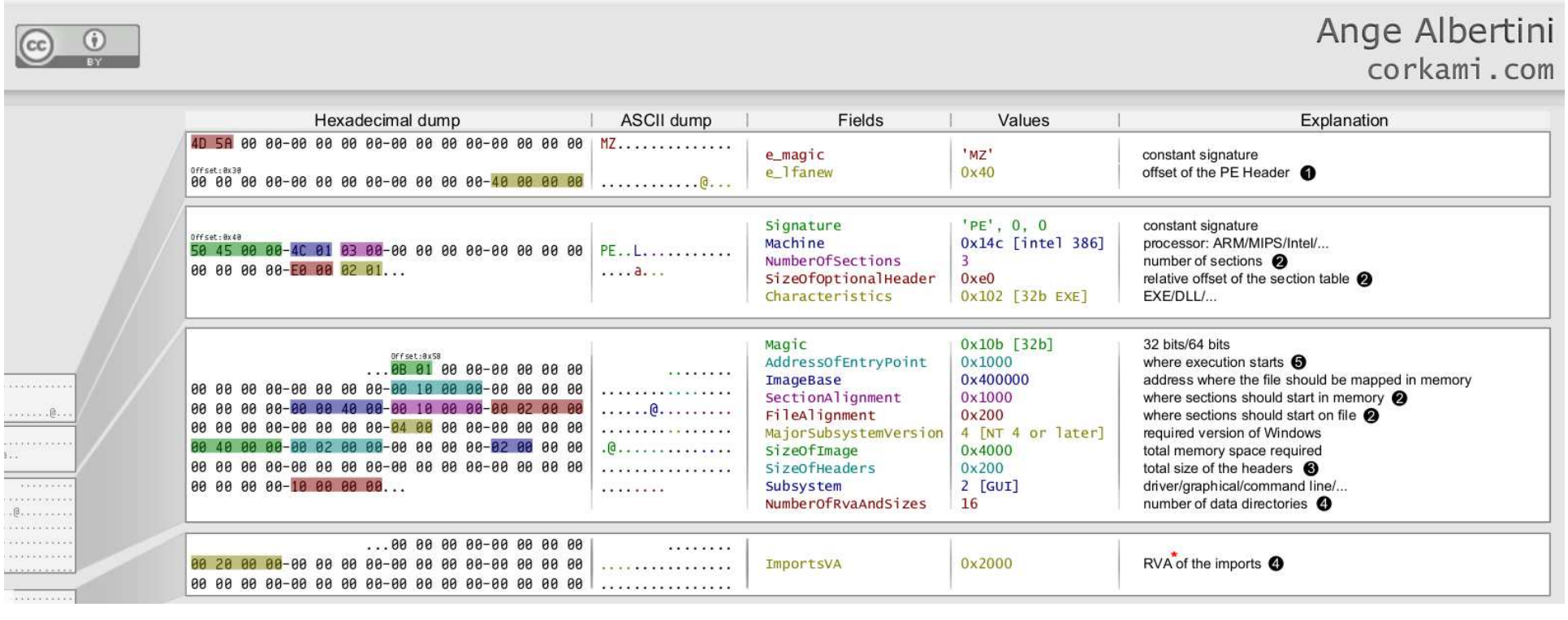
ANGE ALBERTINI
CORKAMI.COM



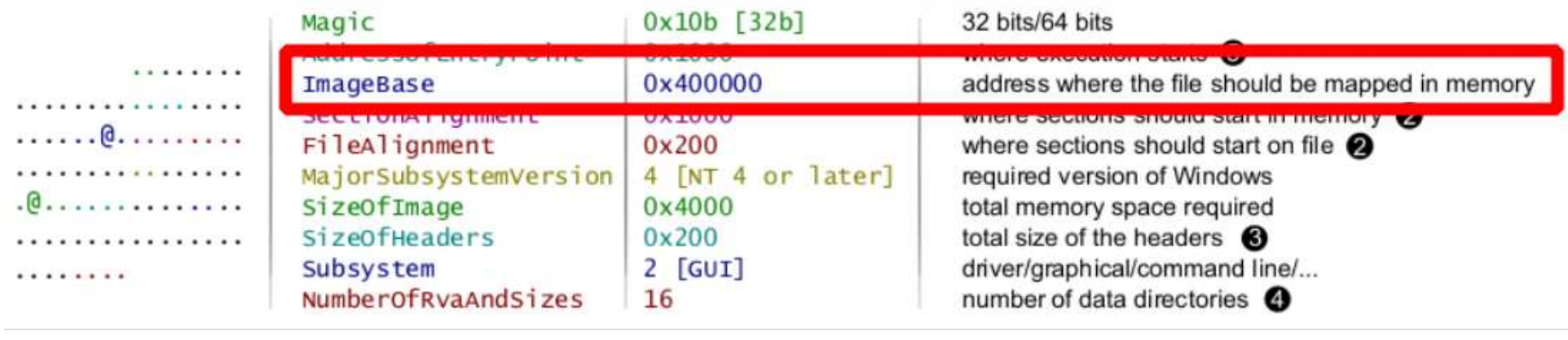
Loading address



Windows PE

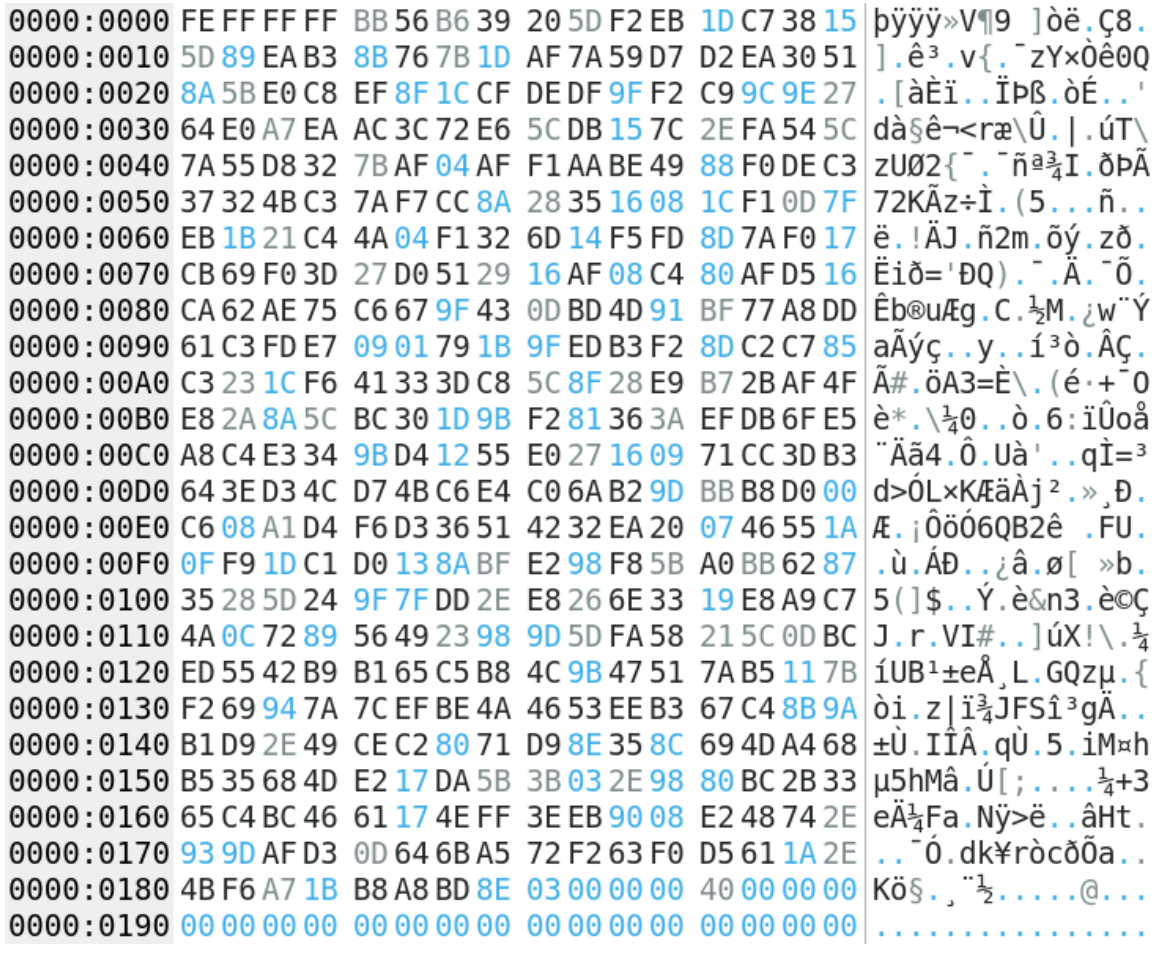

Ange Albertini
corkami.com
Hexadecimal dump
ASCII dump
Fields
Values
Explanation
4D 5A 00 00-00 00 00 00-00 00 00 00-00 00 00 00
MZ..............
e_magic
'MZ'
constant signature
Offset:0x30
00 00 00 00-00 00 00 00-00 00 00 00-40 00 00 00
............@...
e_lfanew
0x40
offset of the PE Header ❶
Offset:0x40
50 45 00 00-4C 01 03 00-00 00 00 00-00 00 00 00
PE..L...........
Signature
'PE', 0, 0
constant signature
00 00 00 00-E0 00 02 01...
....a...
Machine
0x14c [intel 386]
processor: ARM/MIPS/Intel/...
NumberOfSections
3
number of sections ❷
SizeOfOptionalHeader
0xe0
relative offset of the section table ❷
Characteristics
0x102 [32b EXE]
EXE/DLL/...
Offset:0x58
...0B 01 00 00-00 00 00 00
....
00 00 00 00-00 00 00 00-00 10 00 00-00 00 00 00
....
00 00 00 00-00 00 40 00-00 10 00 00-00 02 00 00
......@.........
00 00 00 00-00 00 00 00-04 00 00 00-00 00 00 00
....
00 40 00 00-00 02 00 00-00 00 00 00-02 00 00 00
.@..............
00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
....
00 00 00 00-10 00 00 00...
....
Magic
0x10b [32b]
32 bits/64 bits
AddressOfEntryPoint
0x1000
where execution starts ❺
ImageBase
0x400000
address where the file should be mapped in memory
SectionAlignment
0x1000
where sections should start in memory ❷
FileAlignment
0x200
where sections should start on file ❷
MajorSubsystemVersion
4 [NT 4 or later]
required version of Windows
SizeOfImage
0x4000
total memory space required
SizeOfHeaders
0x200
total size of the headers ❸
Subsystem
2 [GUI]
driver/graphical/command line/...
NumberOfRvaAndSizes
16
number of data directories ❹
...00 00 00 00-00 00 00 00
....
00 20 00 00-00 00 00 00-00 00 00 00-00 00 00 00
....
00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
ImportsVA
0x2000
RVA* of the imports ❹

ImageBase



Magic	0x10b [32b]	32 bits/64 bits
ImageBase	0x400000	address where the file should be mapped in memory
FileAlignment	0x200	where sections should start on file ❷
MajorSubsystemVersion	4 [NT 4 or later]	required version of Windows
SizeOfImage	0x4000	total memory space required
SizeOfHeaders	0x200	total size of the headers ❸
Subsystem	2 [GUI]	driver/graphical/command line/...
NumberOfRvaAndSizes	16	number of data directories ❹

Raw firmware



Finding a needle in a haystack

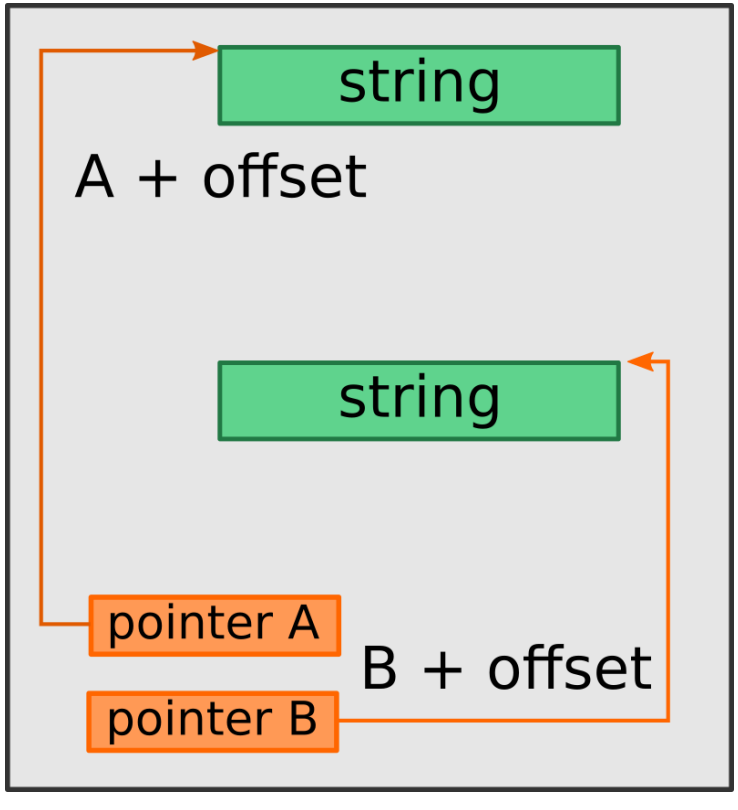
Tools !

Basefind.py

- **Brute-force** on 32-bit address space

```
for base in xrange(args.min_addr, args.max_addr, args.page_size):
    if base % args.page_size == 0:
        print "Trying base address 0x%x" % base
    score = 0
    for ptr in ptr_table.keys():
        if ptr < base:
            #print "Removing pointer 0x%x from table" % ptr
            del ptr_table[ptr]
            continue
        if ptr >= (base + size):
            continue
        offset = ptr - base
        if offset in str_table:
            score += ptr_table[ptr]
```

Brute-force



- Tries every possible offset
- Looks for pointers pointing to **text strings**
- **Count valid pointers**

basefind.cpp

```
for( offset_t base = 0; base < 0xf0000000UL; base += 0x1000 )
{
    [...]
}
```

Same in CPP 😅

rbasefind

- **Multi-threaded brute-force** on 32-bit address

```
while current_addr <= interval.end_addr {
    let mut news = FnvHashSet::default();
    for s in strings {
        match s.checked_add(current_addr) {
            Some(add) => news.insert(add),
            None => continue,
        };
    }
    [...]
    match current_addr.checked_add(config.offset) {
        Some(_) => current_addr += config.offset,
        None => break,
    };
    pb.inc();
}
```

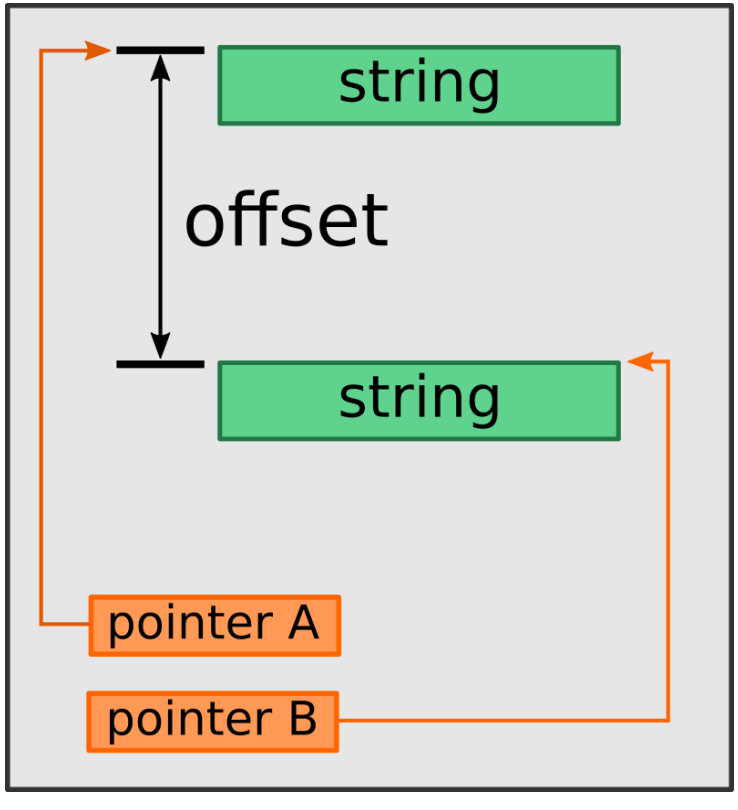
binbloom

- **Splits search space** in segments

```
uint32_t p2(uint32_t x) {
    return 1 << (32 - __builtin_clz(x - 1));
}

/* ... */
size = read_file(filename, &firmware);
/* ... */
mask_segment = ~(p2(size) - 1);
mask_pointer = p2(size) - 1;
nb_segments = mask_segment;
while ((nb_segments & 1) == 0)
    nb_segments = nb_segments >> 1;
nb_segments++;
```

basefind2



- Computes **distance between strings**
- Looks for pointers with same distance
  B - A = offset
- **No brute-force !**

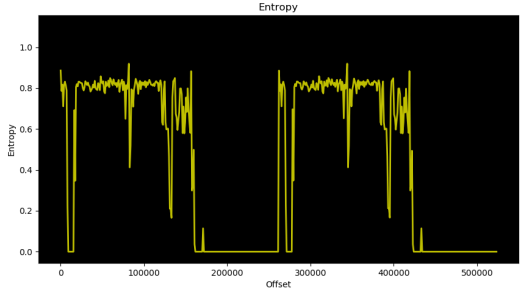
So, limited tools.

- Most of them perform a **brute-force search on 32-bit address space**
- Most of them rely on **text strings**
- **None of them supports 64-bit** architecture (more than 1^{14} possibilities !)

Burning the haystack to get the needle

1. Isolating data

Telling data and code apart



- Entropy can be used to **determine data segments**
- Data segments may contain **strings** and **pointers**
- Everything else is **code**

Avoiding useless data

- Do not consider **long series of 0x00 or 0xFF**
- Generally present in **unused areas** or used as **fillers**

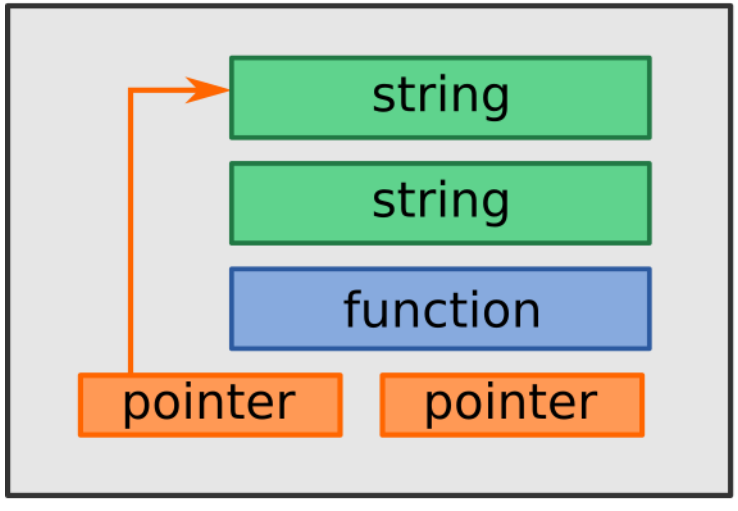
Points of Interest

- **Text strings**
- **Arrays** of similar values
- **Functions** (if possible)

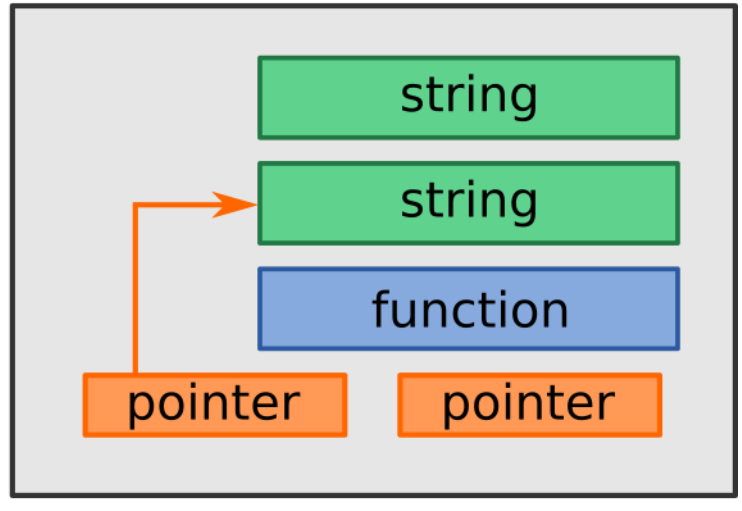
2. Inferring base address

Inference vs. brute-force

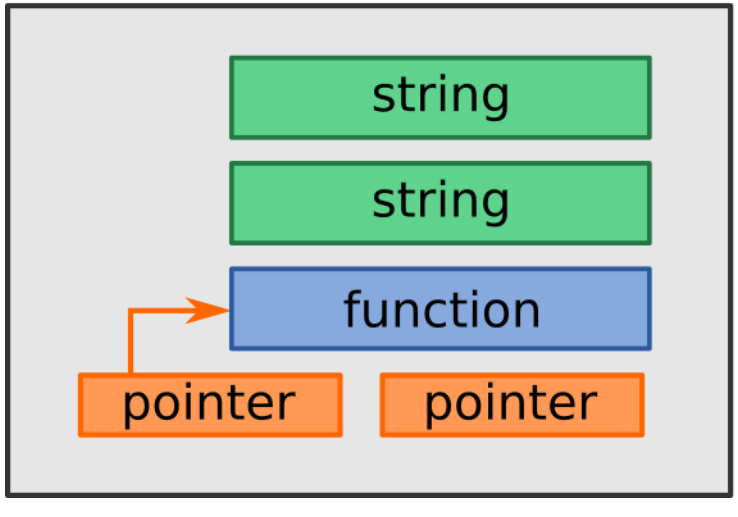
Candidate #1



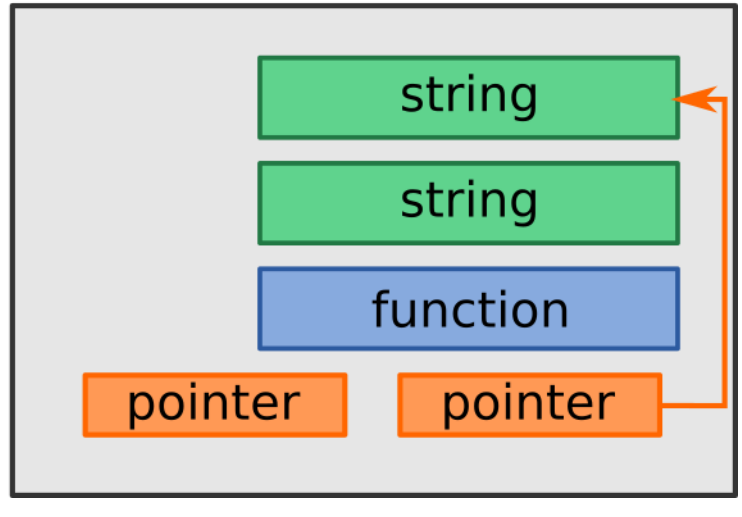
Candidate #2



Candidate #3



Candidate #4



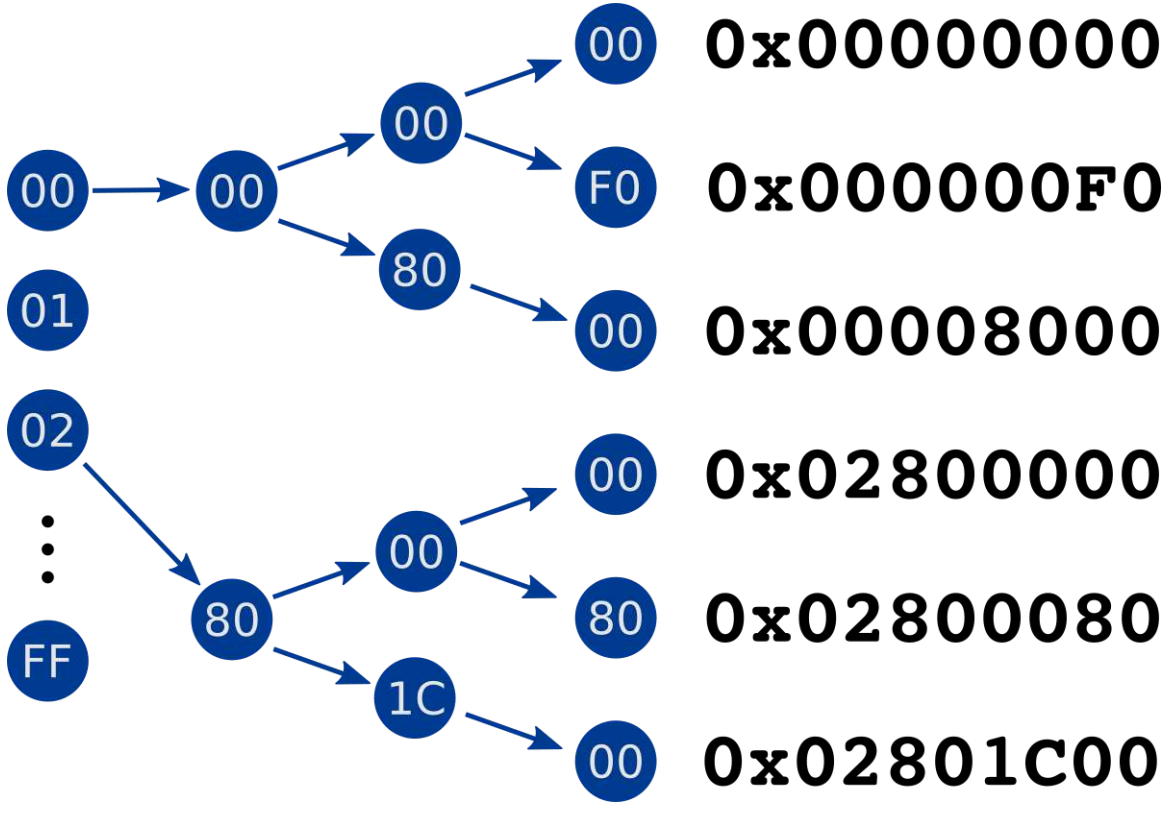
Inference vs. brute-force

- Search space is **reduced**
- Works on both **32-bit and 64-bit** arch firmwares
- Still a **lot of candidates** 😨

Memory & performances issues

- Storing candidates in a list **is not efficient**
- Decided to **use a tree** rather than a list

Candidate addresses tree



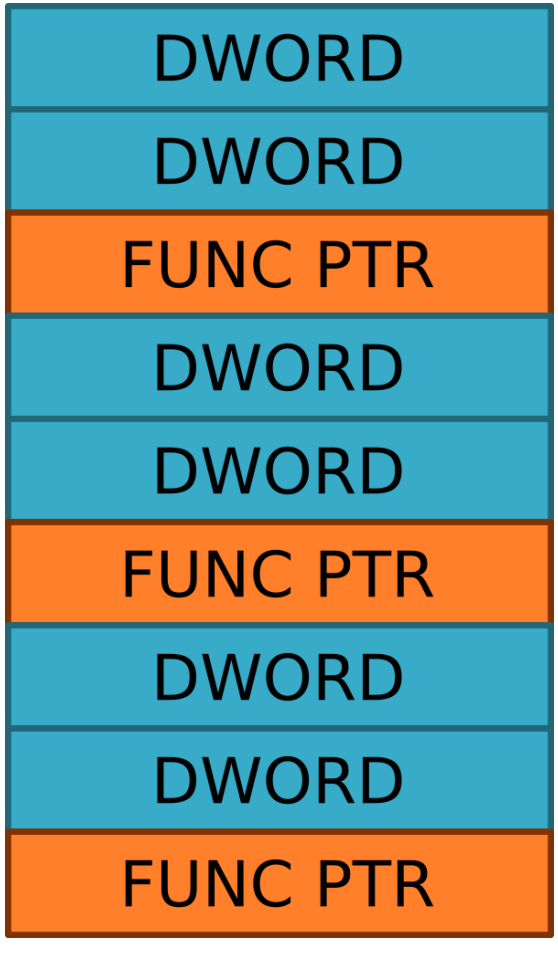
Candidate addresses tree

- **Searching/storing** only requires **8 operations**
- We **can prune the tree** to make room for new candidates

Evaluating candidate addresses

- looking for the address that will give the **best results**
- count **valid pointers**
- **Arrays** of valid pointers have **more weight**
- Compute **a score** for each candidate address

Structured data recognition



- Converting **arrays** of values **into structures**
- As a binbloom v1 legacy, only detects **UDS-related structures** 😇
- … but may do **much more** !

Demo

Binbloom vs. others

Firmware	Binbloom	Binbloom2	Rbasefind
AE5R100V	11.33	3.019	**0.916**
bootloader ARM	5.48	**0.183**	5.40
ECU external flash firmware	5.78	**5.69**	6.17
IntegrityOS application	~	**1.453**	~
UBoot standalone application	8.228	**0.723**	1.462
STM32 firmware	5.232	**0.03**	0.064
Teensy firmware	5.686	0.068	**0.053**
Google Titan M firmware (2018)	9.664	**1.288**	10.23
Flash Air firmware	**11.042**	37.52	44.184

Performances (exec time in seconds)

Improvements

- Automatic architecture detection (with **cpu_rec**)
- **Entropy thresholds** may be defined per-architecture
- Automatic **function detection** per arch
- Better **structured data recognition** algorithm

Try Binbloom v2 !

https://github.com/quarkslab/binbloom

Conclusion

Conclusion

- We presented a more **generic base address search algorithm**
- **Binbloom v2** supports both **32-bit** and **64-bit** archs
- Still some room for improvements, so **stay tuned**

Thanks ! Questions ?

Damien Cauquil

quarkslab

dcauquil@quarkslab.com

virtualabs

virtualabs