Polypyus

Firmware History Based Binary Diffing

Master Thesis Final Presentation

TECHNISCHE UNIVERSITÄT DARMSTADT
SEEMOO
SECURE MOBILE NETWORKING

Polybius	Polypyus
Ancient greek historian **Input**: • A set of historical events. **Output**: • The “Histories”, a 40 volumes work on the rise of Rome to a world power. 	Novel firmware historian **Input**: • A set of “historical” firmware • At least one target firmware **Output**: • A mapping of known functions in the firmware history to regions in the target firmware.

This Thesis
Context

Analyzing the Broadcom/Cypress ARM Bluetooth firmware family, e.g., **stripped Thumb2 binaries**.

This Thesis
Broadcom/Cypress Bluetooth Firmware

Table 4: RNG implementation variants in even more than 17 *Broadcom* and *Cypress* chips.

Chip	Device	Build Date	Variant	HRNG Location	PRNG	Cache
BCM2046A2	iMac Late 2009	2007	1	0xE9A00, 3 regs	Minimal (inline)	No
BCM2070B0	MacBook 2011	Jul 9 2008	1	0xE9A00, 3 regs	Minimal (inline)	No
BCM20702A1	Asus USB Dongle	Feb (?) 2010	1	0xEA204, 3 regs	Minimal (inline)	No
BCM4335C0	Google Nexus 5	Dec 11 2012	2	0x314004, 3 regs	Yes (inline)	No
BCM4345B0	iPhone 6	Jul 15 2013	2	0x314004, 3 regs	Yes (inline)	No
BCM43430A1	Raspberry Pi 3/Zero W	Jun 2 2014	2	0x352600, 3 regs	Yes (inline)	No
BCM4345C0	Raspberry Pi 3+/4	Aug 19 2014	2	0x314004, 3 regs	Yes (inline)	No
BCM4358A3	Samsung Galaxy S6, Nexus 6P	Oct 23 2014	2	0x314004, 3 regs	Yes (inline)	
BCM4345C1	iPhone SE	Jan 27 2015	2	0x314004, 3 regs	Yes (inline)	
BCM4364B0	MacBook/iMac 2017–2019	Aug 21 2015	2	0x352600, 3 regs	Yes (inline)	
BCM4355C0	iPhone 7	Sep 14 2015	2	0x352600, 3 regs	Yes (inline)	
BCM20703A2	MacBook/iMac 2016–2017	Oct 22 2015	2	0x314004, 3 regs	Yes (inline)	No
CYW20719B1	Evaluation board	Jan 17 2017	2	0x352600, 3 regs	Yes (inline)	No
CYW20735B1	Evaluation board	Jan 18 2018	3	0x352600, 3 regs	Yes (rbg_get_psrng), 8 regs	Yes, breaks after 32 elements
CYW20819A1	Evaluation board	May 22 2018	3	0x352600, 3 regs	Yes (rbg_get_psrng), 5 regs	Yes, with minor fixes
BCM		2016	4	None	Only option	No
BCM4347B1	iPhone 8/X/XR	Oct 11 2016	5	0x352600, 4 regs	None	Asynchronous 32x cache
BCM4375B1	Samsung Galaxy S10/Note 10/S20	Apr 13 2018	5	0x352600, 4 regs	None	Asynchronous 32x cache
BCM4378B1	iPhone 11	Oct 25 2018	5	0x602600, 4 regs	None	Asynchronous 32x cache

Symbols available

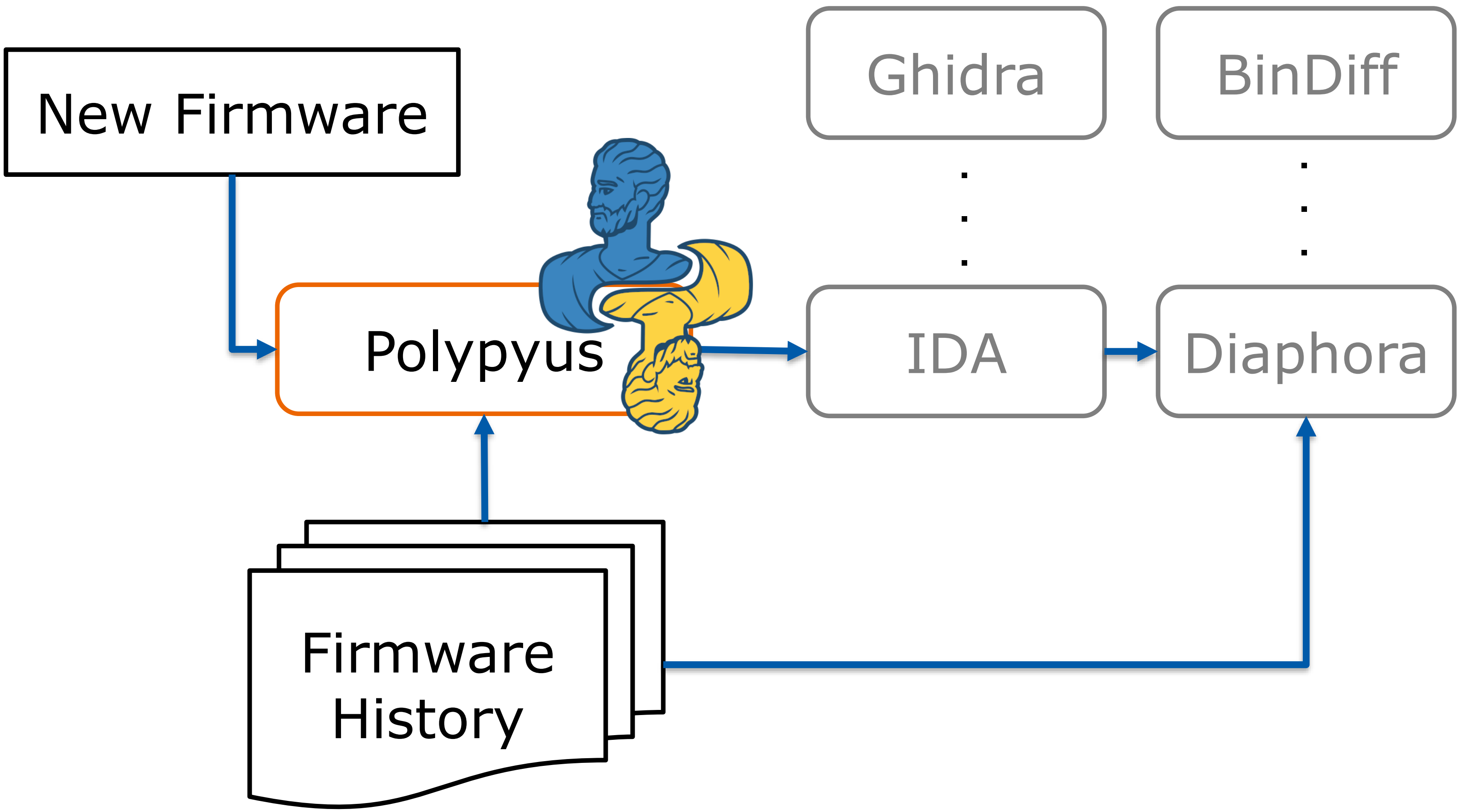
@ „On Randomness in Bluetooth Chips“ by Jörn Tillmanns, Jiska Classen, Felix Rohrbach, Matthias Hollick. 2020

The Thesis

- **Goals**:
  1. **Identify the problem** with binary diffing
  2. **Propose a solution**
     1. **That is fast**

- **Results**:
  1. **Existing** state-of-the-art **tools <u>work</u>** given **correct function starts**
  2. **Polypyus finds correct function starts** and is capable of binary diffing

The Big Picture



State-of-the-art Binary Diffing

Binary Diffing

- Finds differences between binaries
- Usually between two binaries
- Differences are either on
  - a function level
  - or basic block level
- ⟹ Produces a weighted mapping between functions/blocks
- State-of the-art tools:
  - IDA
    - BinDiff
    - Diaphora
  - Ghidra (version tracking, BinDiff)
  - Radare2 (rdiff2)
  - . . .

Result of Binary Diffing

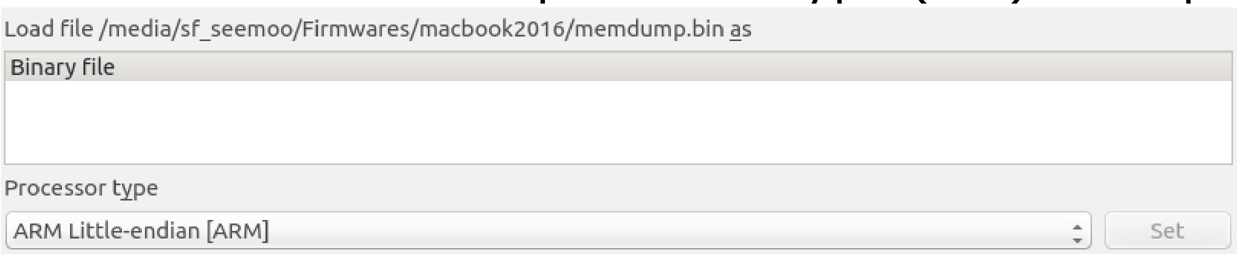
Similarity	Cor	Change	EA Primary	Name Primary	EA Secondary	Name Secondary	Co	Algorithm
1.00	0.99	-------	00042718	f_2587	000D34C4	wiced_rtos_push_to_queue		edges flowgraph MD index
1.00	0.99	-------	0004290E	f_2606	000D36BC	wiced_rtos_get_semaphore		edges flowgraph MD index
1.00	0.99	-------	000446FC	f_2683	000A3E60	_scanTaskCheckAbortRxPkt		edges flowgraph MD index
1.00	0.99	-------	00045C2E	f_2739	0002AF5A	rfm_RegisterWrite		hash matching
1.00	0.99	-------	00045C4E	f_2740	0002AF7A	rfm_RegisterRead		hash matching
1.00	0.99	-------	000471F6	f_2805	000B62F4	_bcsulp_getLrmOffset		edges flowgraph MD index
1.00	0.99	-------	000479AE	f_2822	000BDC1A	bcsulp_encryptTxBuffer		hash matching
1.00	0.99	-------	000479E2	f_2823	000BDC4E	bcsulp_decryptData		hash matching
1.00	0.99	-------	00048038	f_2844	00055AAC	_ll_shift_l		hash matching
1.00	0.99	-------	00049CB8	f_2934	0008B2BE	bcs_coexSlaveUpdateDeferCounter		edges flowgraph MD index
1.00	0.99	-------	0004AF7A	f_2990	0002CE80	bcsulp_advTaskIsActive		edges flowgraph MD index
1.00	0.99	-------	0004CB4C	f_3056	0005B318	apep_cpyFifo2Mem8		hash matching
1.00	0.99	-------	0004CB5E	f_3057	0005B32A	apep_cpyMem2Fifo8		hash matching
1.00	0.99	-------	0004CB70	f_3058	0005B33C	apep_cpyFifo2Mem16		hash matching
1.00	0.99	-------	0004CB84	f_3059	0005B350	apep_cpyMem2Fifo16		hash matching
1.00	0.99	-------	0004CB98	f_3060	0005B364	apep_cpyMemFifo2Mem8		hash matching
1.00	0.99	-------	0004CBB2	f_3061	0005B37E	apep_cpyMem2MemFifo8		hash matching
1.00	0.99	-------	0004CBC6	f_3062	0005B392	apep_cpyMem2PcmSwap		hash matching
1.00	0.99	-------	0004CBDC	f_3063	0005B3A8	apep_cpyMem2PcmMute		hash matching
1.00	0.99	-------	0004CBF0	f_3064	0005B3BC	apep_cpyPcm2MemSwap		hash matching
1.00	0.99	-------	0004CC08	f_3065	0005B3D4	apep_cpyPcm2MemMute		hash matching
1.00	0.99	-------	0004CC24	f_3066	0005B3F0	apep_cpyMem2Pcm_8padTo16		hash matching
1.00	0.99	-------	0004CC36	f_3067	0005B402	apep_cpyPcm2Mem_16unpadTo8		hash matching

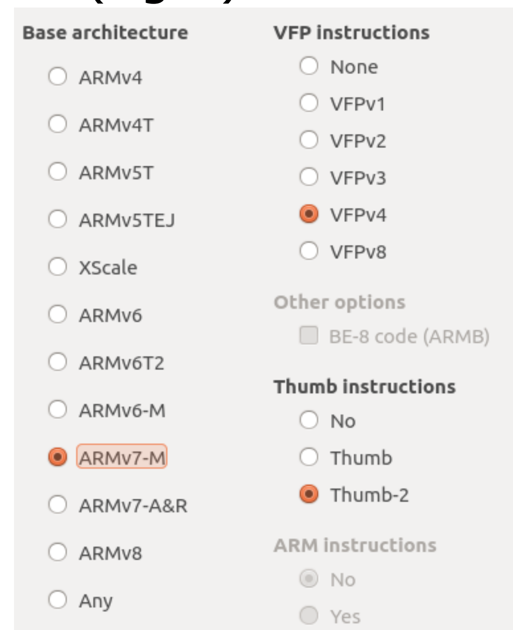
BinDiff result for perfect function starts in CYW20719B1 and CYW20819A1

IDA Investigation
How to Import Symbols?

For stripped firmware dumps:

- The user states processor type (left) and options (right)



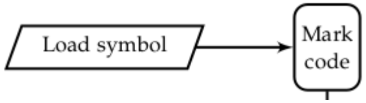


- Then, the user must provide initial locations for the RDD algorithm
- The algorithm disassembles starting from these locations.

Several challenges arise when importing symbols this way

1. IDA finds ARM32 code in Thumb-2 only base architecture!
2. Next slide

IDA Investigation
How to Import Symbols?



State-of-the-Art Tools

- IDA
  - BinDiff
  - Diaphora
- Ghidra (version tracking, BinDiff)
- Radare2 (rdiff2)
- . . .

Require correct function starts

Did not finish after 24 hours of diffing

Why require pre-marked functions in both binaries?

Reasons to Require Function Starts

1. Heuristics that are based on disassembly
2. Using disassembly to compute control flow graphs

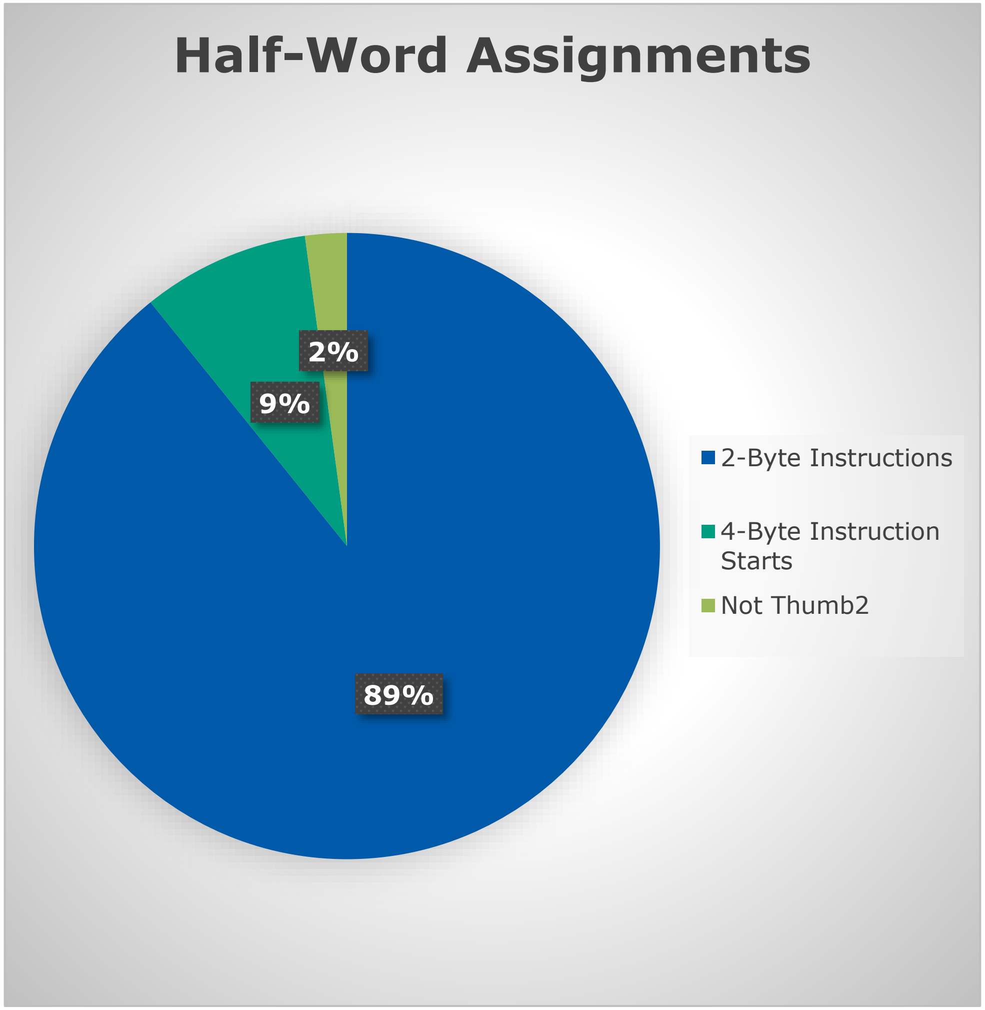
Disassembly needs code addresses to start
Because:

Differentiating between code and data:

- Is hard
- It is especially hard:
  - In stripped binaries
  - With high density instruction sets

The Function Start Problem

Thumb2 Investigations
Random Input



The 4-Byte instruction starts were

- completed in average in 67% of the cases

The completing tails were

- in 88% of cases a 2-Byte instruction themselves
- in 9.5% of cases the start of a 4-Byte instruction

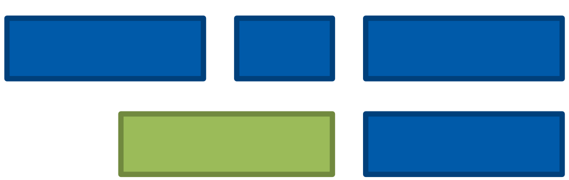
Thumb2 Investigations
Implications

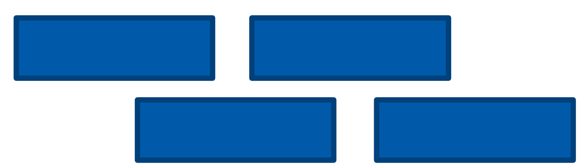
- High code density → Data looks just like code
- Variable instruction length 2-Byte and 4-Byte -> Off-by-One errors

```
correct:
        0x270D9C    bl      #0x11c6a2
        0x270DA0    pop     {r3, pc}
        0x270DA2    movs    r0, r0
        0x270DA4    push    {r4, r5, r6, lr}
        0x270DA6    mov     r6, r1
        0x270DA8    ldr     r1, [pc, #0x50]
```

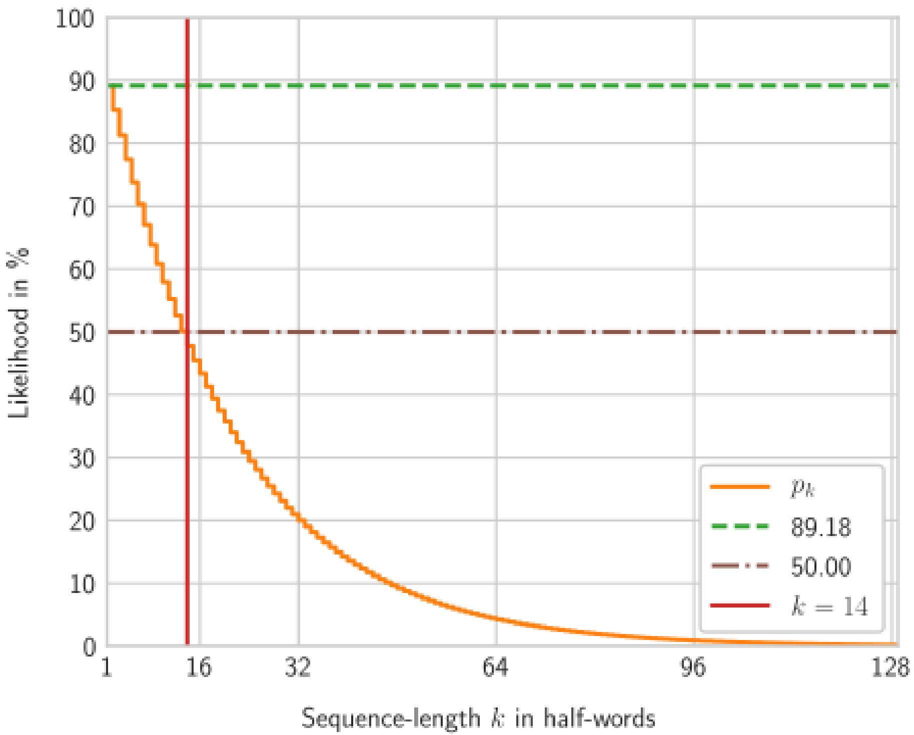
```
offset:
        0x270D9E    stc2    p13, c11, [r1], {8}

        0x270DA2    movs    r0, r0
        0x270DA4    push    {r4, r5, r6, lr}
        0x270DA6    mov     r6, r1
        0x270DA8    ldr     r1, [pc, #0x50]
```

Think about it: the „code“ starting at 0x270DA2 is maybe not reachable at all.



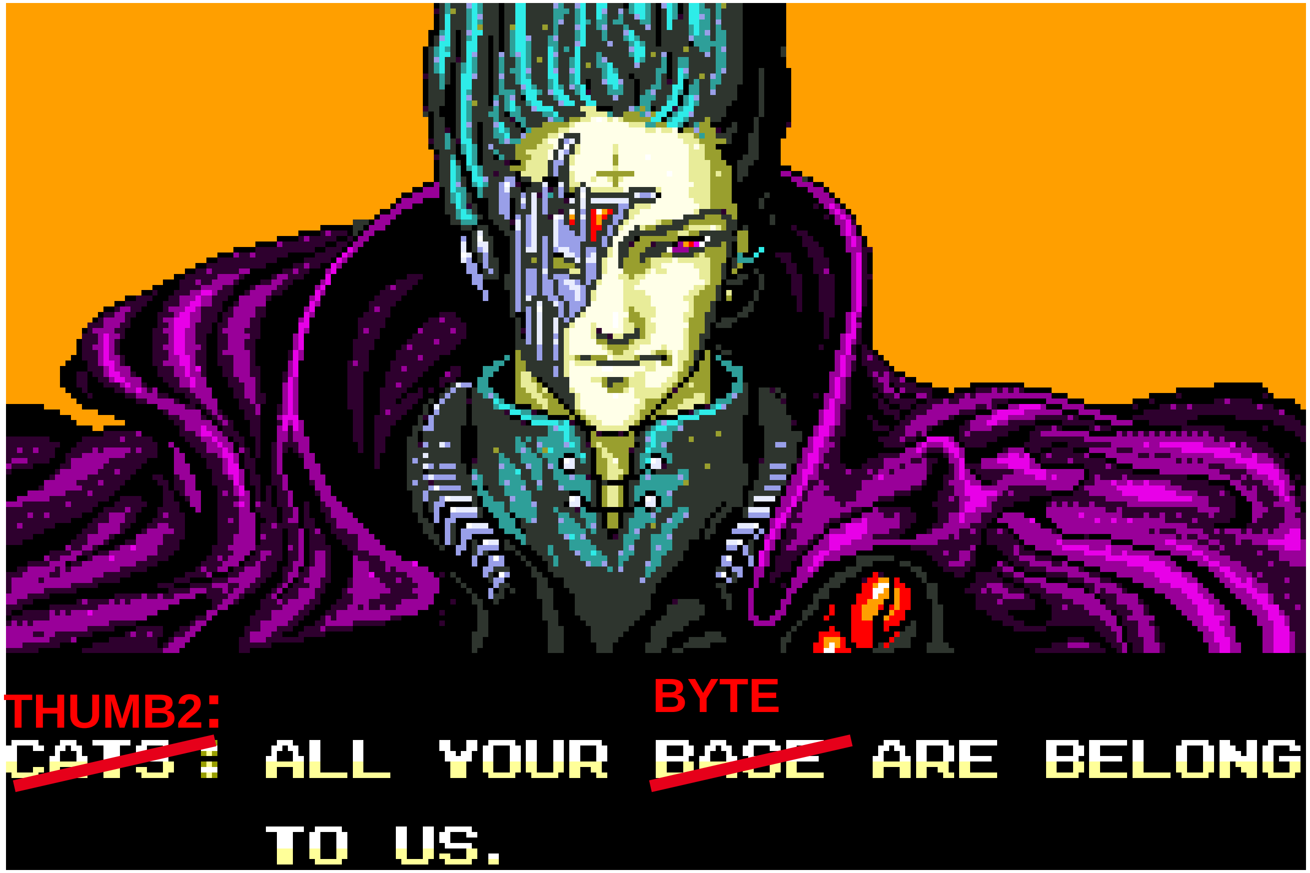


Thumb2 Investigations
Likelihood of Random Code



- Likelihood for a uniformly sampled half-word sequence to be syntactically valid Thumb2 code
- Defined as a recursive function that considers all combinations of 2-Byte and 4-Byte instructions
- Break even point at 28 Bytes

Is this the End of the Story?



But Can We do any Better?

We might have an approach

Polypyus: Using Binary Differences

Polypyus
Motivation

```
:000C2636 89 49             LDR     R1, =rm_deviceInfo ; fcn.lc
:000C2638 B1 F8 54 00       LDRH.W  R0, [R1,#(word_280BBC - 0x2
:000C263C B0 F5 00 6F       CMP.W   R0, #0x800
:000C2640 88 BF             IT HI
:000C2642 02 20             MOVHI   R0, #2
:000C2644 05 D8             BHI     loc_C2652
:000C2646 B1 F8 52 20       LDRH.W  R2, [R1,#(word_280BBA - 0x2
:000C264A 82 42             CMP     R2, R0
:000C264C 0C BF             ITE EQ
:000C264E 00 20             MOVEQ   R0, #0
:000C2650 01 20             MOVNE   R0, #1
:000C2652
:000C2652         loc_C2652                      ; CODE XREF
:000C2652 CA 78             LDRB    R2, [R1,#(rm_deviceInfo+3 -
:000C2654 60 F3 05 12       BFI.W   R2, R0, #4, #2
:000C2658 CA 70             STRB    R2, [R1,#(rm_deviceInfo+3 -
:000C265A 70 47             BX      LR
:000C265A         ; End of function f_7895
```

CYW20719B1

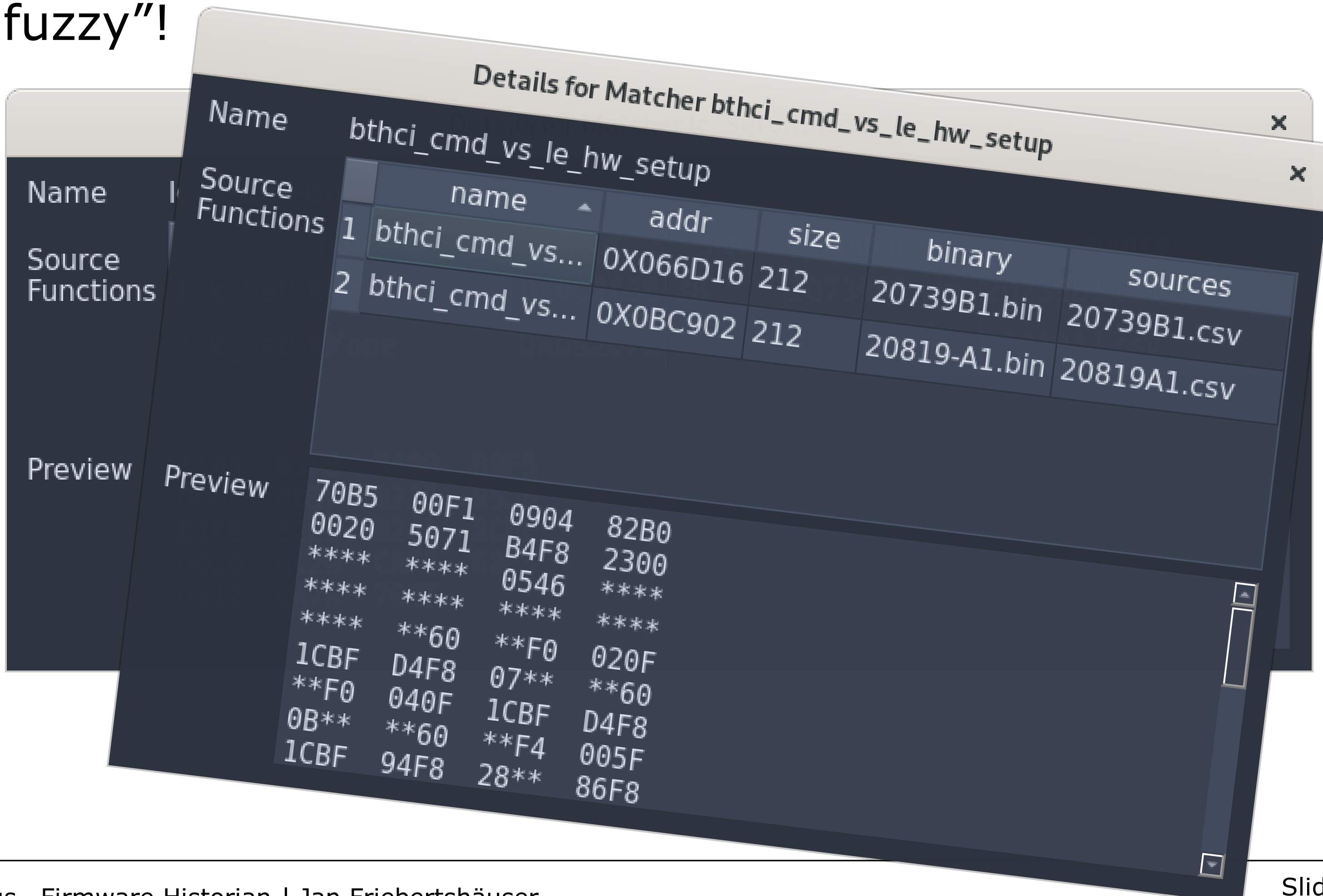
```
:0009207C         lc_SetSRMode                   ; CODE XREF: bthci_cr
:0009207C 6C 49             LDR     R1, =rm_deviceInfo ; fcn.lc_SetSRMode
:0009207E B1 F8 54 00       LDRH.W  R0, [R1,#(word_201094 - 0x201040)]
:00092082 B0 F5 00 6F       CMP.W   R0, #0x800
:00092086 88 BF             IT HI
:00092088 02 20             MOVHI   R0, #2
:0009208A 05 D8             BHI     loc_92098
:0009208C B1 F8 52 20       LDRH.W  R2, [R1,#(word_201092 - 0x201040)]
:00092090 82 42             CMP     R2, R0
:00092092 0C BF             ITE EQ
:00092094 00 20             MOVEQ   R0, #0
:00092096 01 20             MOVNE   R0, #1
:00092098
:00092098         loc_92098                      ; CODE XREF: lc_SetSR
:00092098 CA 78             LDRB    R2, [R1,#(rm_deviceInfo+3 - 0x201040)
:0009209A 60 F3 05 12       BFI.W   R2, R0, #4, #2
:0009209E CA 70             STRB    R2, [R1,#(rm_deviceInfo+3 - 0x201040)
:000920A0 70 47             BX      LR
:000920A0         ; End of function lc_SetSRMode
```

CYW20819A1

- There are minimal differences between the “same” functions in different firmware versions
- These differences come from
  - changes in ordering of functions/data
  - different function/data sizes
  - changes in source code
  - . . .

Polypyus
Individual Matchers

Marking bytes that differ between instances of the same function as “fuzzy”!



Polypyus
Matcher Creation

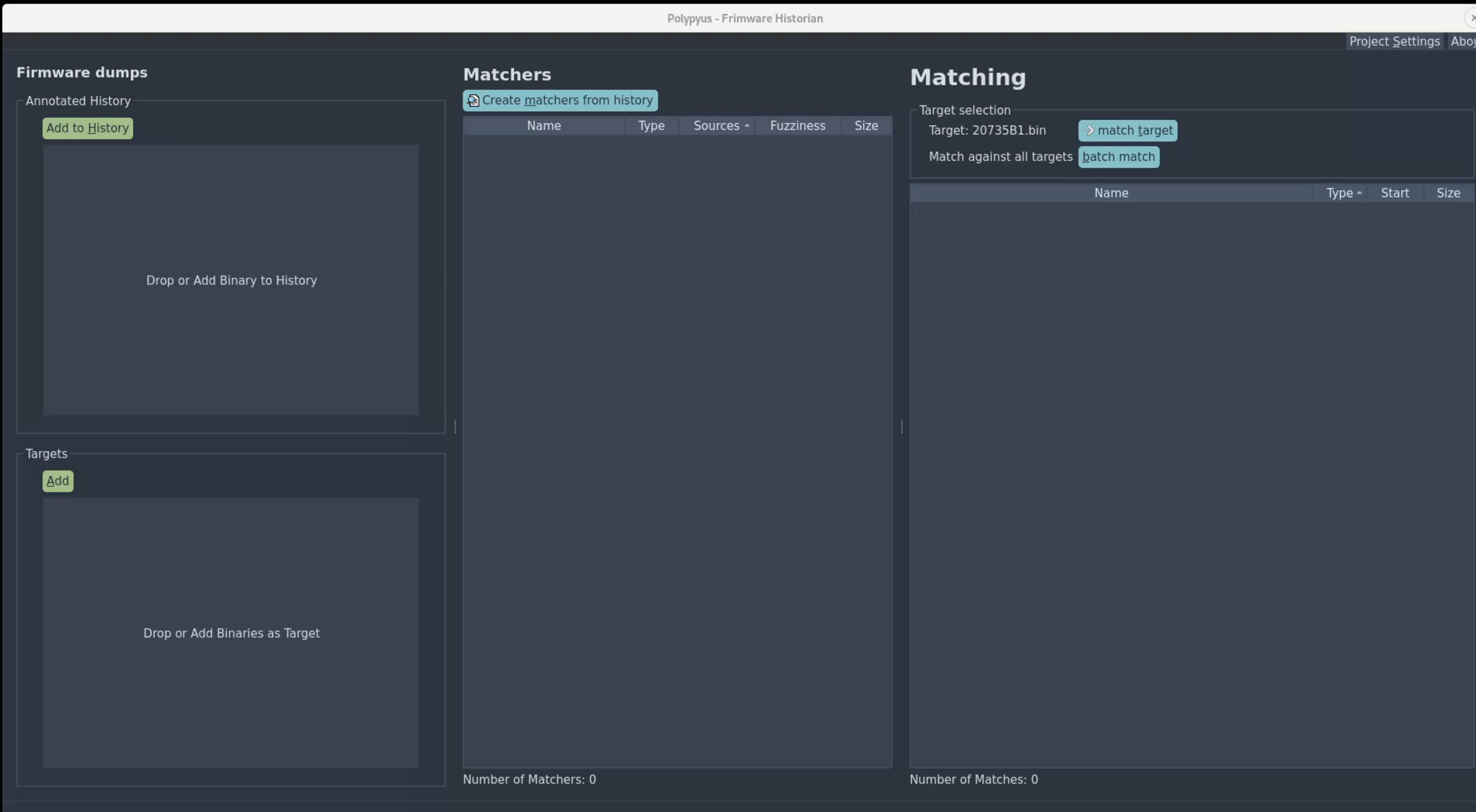
Min. distance d to neighbor-fuzziness \ Fuzzy-sequence length k	1	2	3	4	5	6	7	8	9	10
0	1.00	2.00	3.16	4.21	5.52	6.63	8.12	9.28	10.95	12.17
1	1.95	3.90	6.16	8.21	10.77	12.93	15.83	18.09	21.36	23.73
2	1.90	3.80	6.01	8.01	10.51	12.61	15.44	17.65	20.84	23.15
3	1.86	3.71	5.87	7.82	10.26	12.31	15.07	17.23	20.34	22.60
4	1.81	3.63	5.73	7.64	10.02	12.03	14.73	16.83	19.87	22.08
5	1.77	3.55	5.60	7.47	9.80	11.76	14.40	16.45	19.43	21.59
6	1.74	3.47	5.48	7.31	9.58	11.50	14.08	16.09	19.00	21.11
7	1.70	3.40	5.36	7.15	9.38	11.26	13.78	15.75	18.60	20.67

Using a cost-based approach, creation of matchers is restricted.

Dense clusters of fuzziness are more expensive than sparsly distributed fuzzy bytes.

More fuzzyness means higher costs.

A function's allowance is based on its length and choosable thresholds.


Polypyus - Frimware Historian
Project Settings
About
Firmware dumps
Annotated History
Add to History
Drop or Add Binary to History
Targets
Add
Drop or Add Binaries as Target
Matchers
Create matchers from history
Name
Type
Sources
Fuzziness
Size
Number of Matchers: 0
Matching
Target selection
Target: 20735B1.bin
match target
Match against all targets
batch match
Name
Type
Start
Size
Number of Matches: 0

Polypyus
Performance Goals Achieved

1. Creating matchers: **8.5s**
2. Finding new matches: **9.5s**

The Performance is achieved by several optimizations

1. A matcher prefix tree deduplicates the matchers
2. Match finding is a depth first search in the prefix tree
3. Bins in the tree reduce number of visited matcher fragments
4. Search is restricted to partitions of the binary, that potentially contain code

And a general focus on performance throughout the whole implementation.

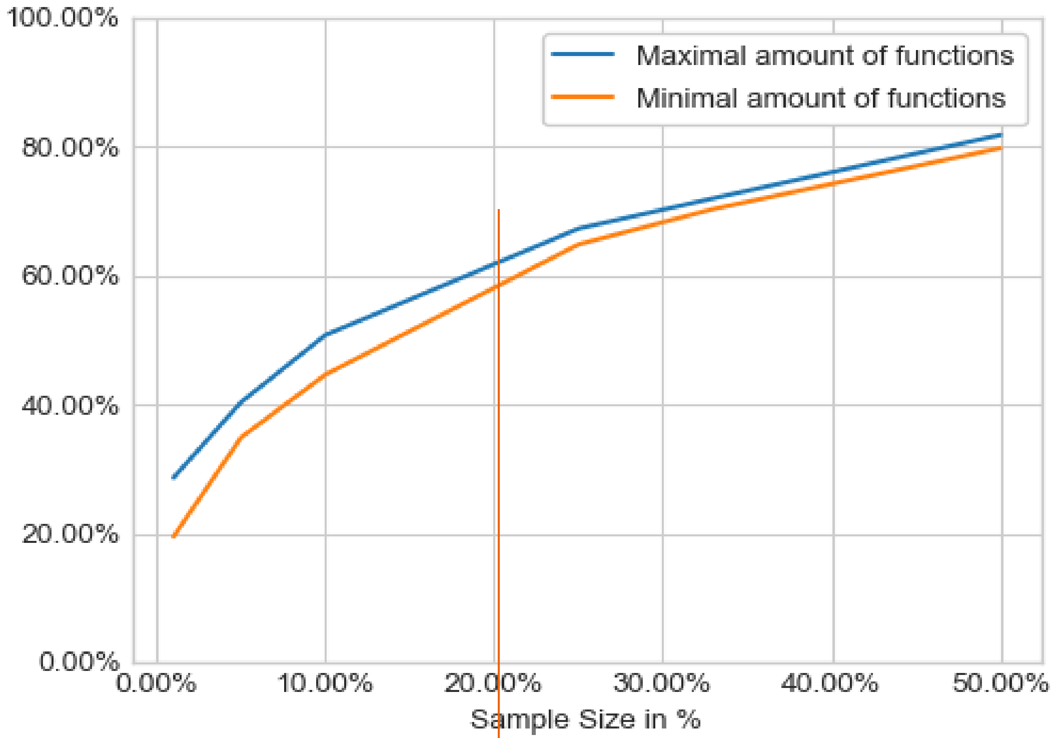
Polypyus: Finding More Functions

Polypyus offers an optional function prefix finder

1. From the firmware history all the function prefixes are collected
2. The unmatched regions of the target binary are searched for exact matches to these prefixes

- In the example in the video, the function prefix finder was disabled
- We found 2344 functions of around a maximum of ~10500 common functions
- In the same setting with function prefix finder we find 7321 functions

IDA: Amount of Functions After Import



For i = 1,5,10,25,33,50 We tested how many functions IDA finds when we import i% of symbols. We repeat this experiment for each i 100/i times with disjoint samples.

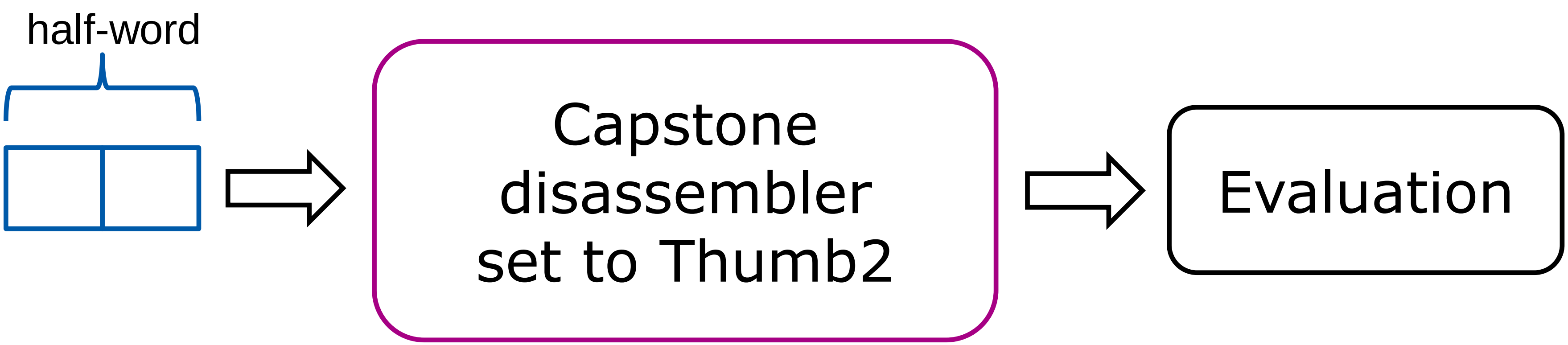
Import without function finder

The End. Questions?

Firmware similarity

firmware a	firmware b	unique symbols	common symbols
20735B1 (01/18)	20739B1 (01/17)	4790	10435
20819A1 (03/18)	20735B1	1927	9570
20819A1	20739B1	6115	9515

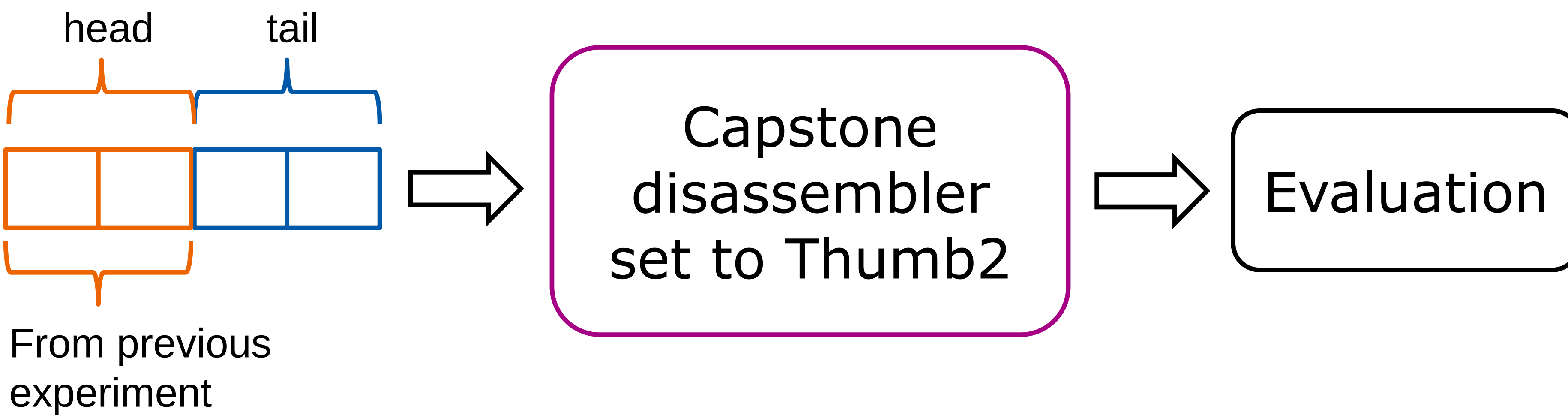
Extra: Thumb2 Investigations
1.: All Half-Word Assignments



Evaluation:

- Assignment is code:
  → Simplify mnemonic & count occurences
- *Assignment is not code:*
  → Assignment is input for next Investigation

Extra: Thumb2 Investigations
2.: 4-Byte Assignments

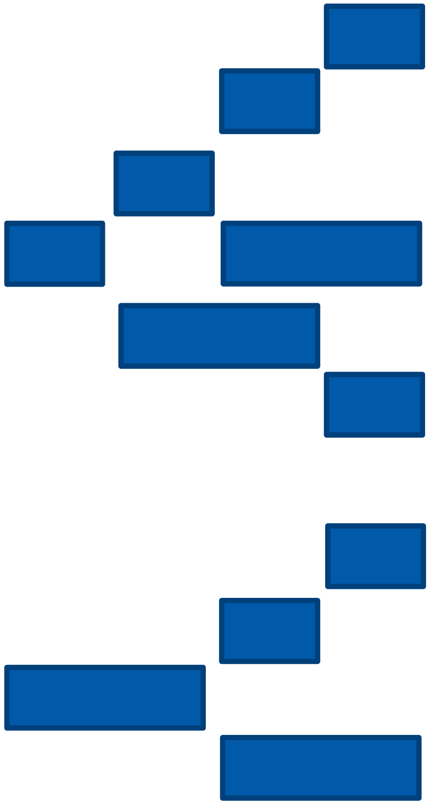


Evaluation:

- Assignment is code:
  → Simplify mnemonic & count occurences
  → test if tail is by itself code

Extra:
Random Code Function

$$p_k = \begin{cases} a \cdot p_{k-1} + b \cdot p_{k-2}, & k \geqslant 2 \\ 1, & k = 0 \\ a, & k = 1 \end{cases}$$



Where a is the Thumb2 code density in uniformly sampled half-word assignments and b is the density of 4-Byte instruction heads (half-word) Multiplied by the likelihood of another uniformly sampled half-word completing the 4-Byte instruction.

Extra:
Cost Function

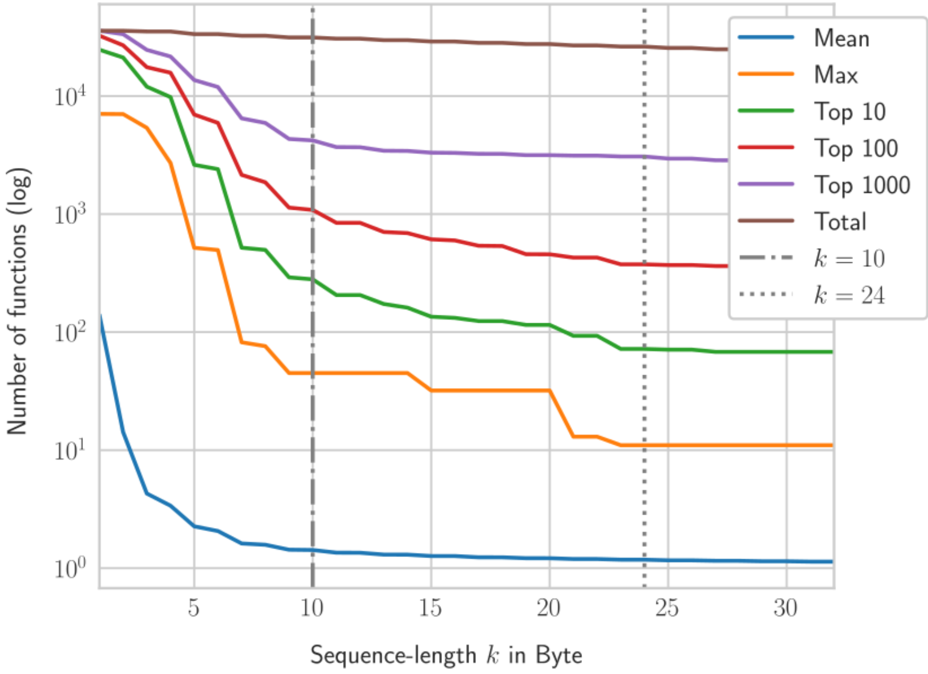
$$\text{matcher-cost}(m) = \sum_{(k,d)\in F_m} \text{sequence-cost}(k, d)$$

$$\text{matcher-cost}(m) \leqslant (|m| - \mu) \cdot \phi$$

$$\text{sequence-cost}(k, d) = \frac{k}{p(\lceil k/2 \rceil)} \cdot (1 + \text{proximity-penalty}(d))$$

$$\text{proximity-penalty}(d) = \begin{cases} 0 & , d = 0 \\ 0.9^d & , \text{ otherwise} \end{cases}$$

Polypyus
Minimum Function Length Threshold

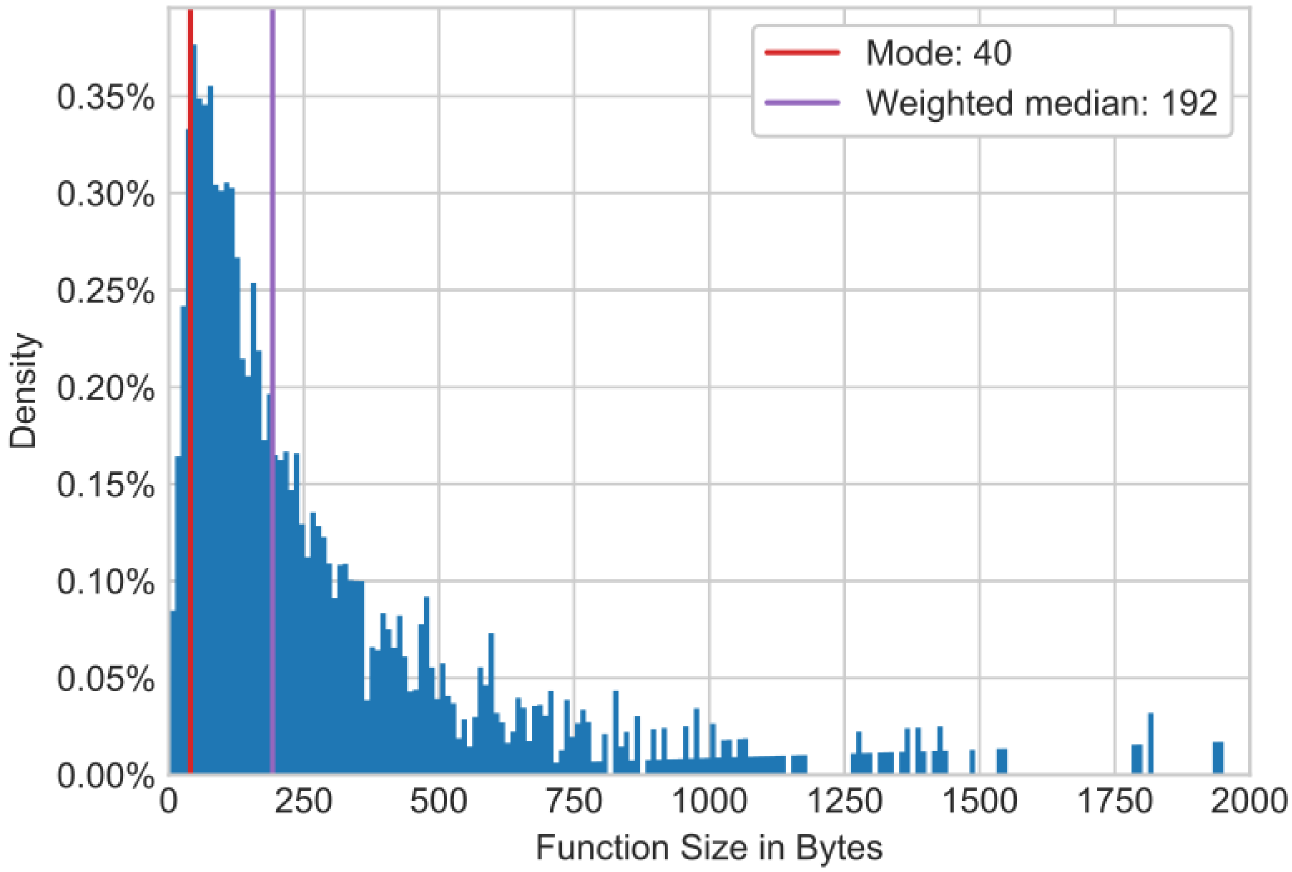


Number of functions with the same prefix of length k Byte.

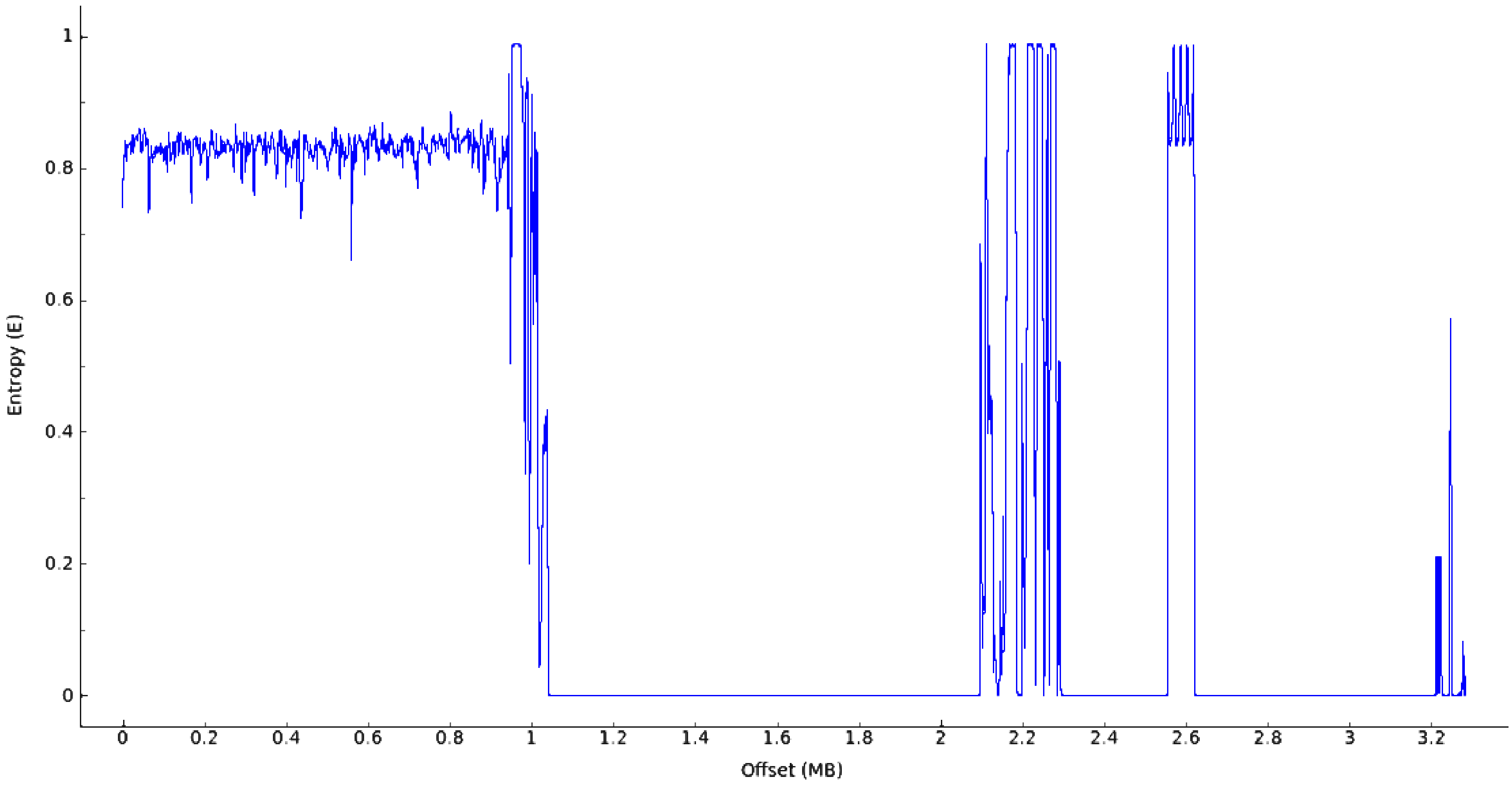
To prevent short fuzzy functions that produce many false-positives, a minimum function length is imposed.

In the firmware dumps we analyzed, the Byte thresholds 10 or 24 are reasonable.

Extra:
Cost Function - Function Size



Extra:
Binwalk Entropie CYW20819A1



TECHNISCHE UNIVERSITÄT DARMSTADT