Sticky Business

All the things you do not want to know about USB devices

Corstian Boerman

Nils Frahm

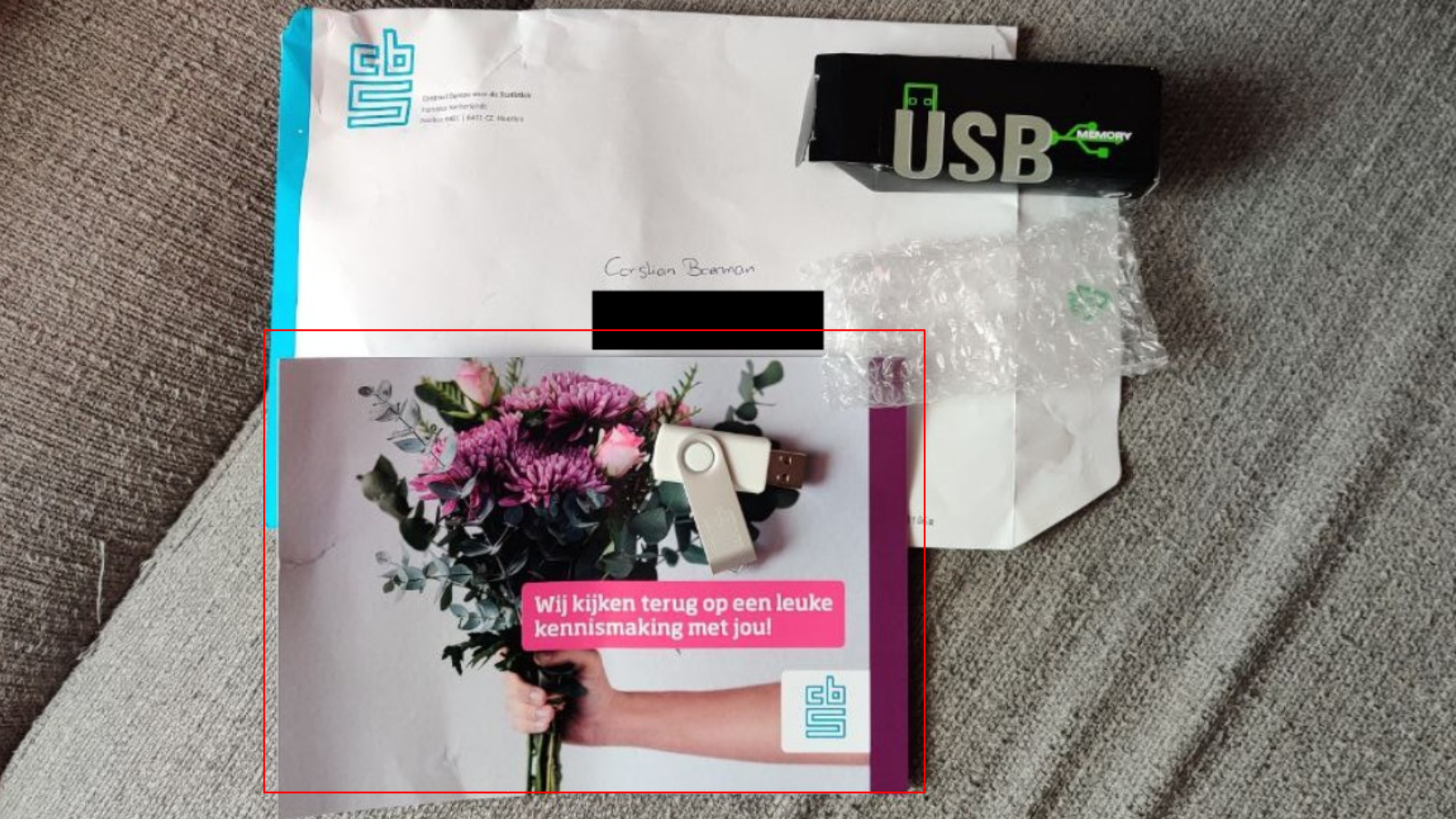
September 14th 2023


USB
MEMORY
Corstian Boerman
[redacted]
Wij kijken terug op een leuke kennismaking met jou!


USB
MEMORY
Corstian Boerman
[redacted]
Wij kijken terug op een leuke kennismaking met jou!


USB
MEMORY
Corstian Boerman
[redacted]
Wij kijken terug op een leuke kennismaking met jou!


USB
MEMORY
Corstian Boerman
[redacted]
Wij kijken terug op een leuke kennismaking met jou!


USB
MEMORY
Corstian Boerman
[redacted]
Wij kijken terug op een leuke kennismaking met jou!

Hypothesis:

“Part of the interview process..?”

Solid advice:

Do not ever plug a USB found on the street into your system

Ps. if one is directly addressed to you; RUN

DISCLAIMER:

This presentation is a highly stylized version of what had actually happened

Pluggin’ it in…

Pluggin’ it in… aaaand it’s empty.

Poke it with a stick:

Try copying a file

Intermezzo; what actually to do

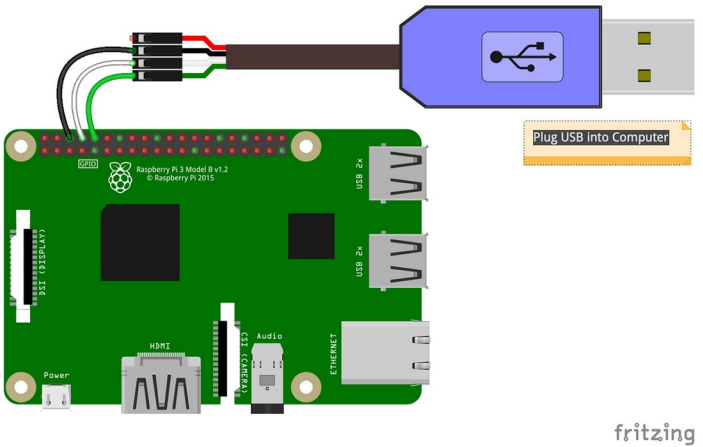
1) Grab a (freshly installed) Raspberry PI and SPI cable

2) Create a serial connection

3) Disable USB auto-mount

4) Plug in the weird usb device

5) Create a drive image
   (dd if=/dev/sda of=image.dd bs=512)

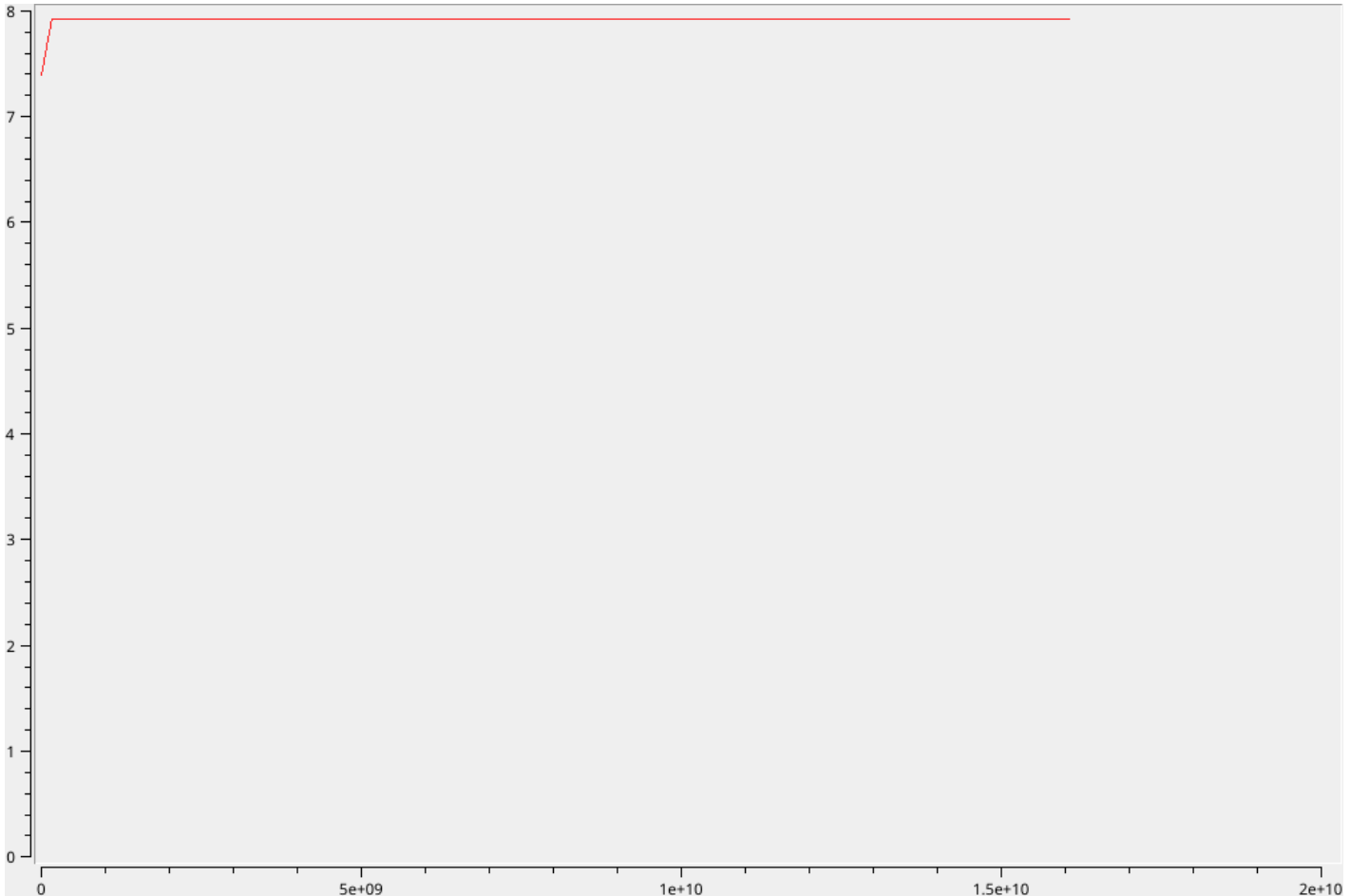


Binwalk!

https://github.com/ReFirmLabs/binwalk

ps. I liked V2 better

Entropy is sky-high!



Bingo!

```
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
45238826      0x2B24A2A       MPEG transport stream data
56781204      0x3626994       Uncompressed Adobe Flash SWF file, Version 77, File size
                              (header included) 2698231425
57507841      0x36D8001       mcrypt 2.2 encrypted data, algorithm: blowfish-448, mode: CBC,
                              keymode: 8bit
76350477      0x48D040D       PGP RSA encrypted session key - keyid: 1005801E 1478776A RSA
                              (Encrypt or Sign) 1024b
82073172      0x4E45654       MySQL MISAM compressed data file Version 6
94300930      0x59EEB02       Uncompressed Adobe Flash SWF file, Version 28, File size
                              (header included) 2769373577
117629626     0x702E2BA       EBML file
132057858     0x7DF0B02       Uncompressed Adobe Flash SWF file, Version 27, File size
                              (header included) 2232505748
164259563     0x9CA66EB       QEMU QCOW Image
169814786     0xA1F2B02       Uncompressed Adobe Flash SWF file, Version 26, File size
                              (header included) 1695637919
207571714     0xC5F4B02       Uncompressed Adobe Flash SWF file, Version 25, File size
                              (header included) 1158770090
218158077     0xD00D3FD       QNX4 Boot Block
```

```
4328626692    0x102019A04    Intel x86 or x64 microcode, sig 0xe1f7db10, pf_mask 0xa5e79d4b, 1A0F-01-01, rev 0xf108611, size 412877
4328627204    0x102019C04    Intel x86 or x64 microcode, sig 0xe1f99d10, pf_mask 0xa5ece34b, 1C0F-01-01, rev 0xf10a411, size 413568
4328627716    0x102019E04    Intel x86 or x64 microcode, sig 0xe1fb5f10, pf_mask 0xa5f2294b, 1E0F-01-01, rev 0xf10c211, size 414259
4328628228    0x10201A004    Intel x86 or x64 microcode, sig 0xe1fd2110, pf_mask 0xa5f76f4b, 200F-01-01, rev 0xf10e011, size 414950
4328692228    0x102029A04    Intel x86 or x64 microcode, sig 0xe2d8db10, pf_mask 0xa88a9d4b, 1A0F-01-02, rev 0xf1f8611, size 718540
4328692740    0x102029C04    Intel x86 or x64 microcode, sig 0xe2da9d10, pf_mask 0xa88fe34b, 1C0F-01-02, rev 0xf1fa411, size 725452
4328693252    0x102029E04    Intel x86 or x64 microcode, sig 0xe2dc5f10, pf_mask 0xa895294b, 1E0F-01-02, rev 0xf1fc211, size 732364
4328693764    0x10202A004    Intel x86 or x64 microcode, sig 0xe2de2110, pf_mask 0xa89a6f4b, 200F-01-02, rev 0xf1fe011, size 739276
4328757764    0x102039A04    Intel x86 or x64 microcode, sig 0xe3b9db10, pf_mask 0xab2d9d4b, 1A0F-01-03, rev 0xf2e8611, size 160327
4328758276    0x102039C04    Intel x86 or x64 microcode, sig 0xe3bb9d10, pf_mask 0xab32e34b, 1C0F-01-03, rev 0xf2ea411, size 161018
4328758788    0x102039E04    Intel x86 or x64 microcode, sig 0xe3bd5f10, pf_mask 0xab38294b, 1E0F-01-03, rev 0xf2ec211, size 161710
4328759300    0x10203A004    Intel x86 or x64 microcode, sig 0xe3bf2110, pf_mask 0xab3d6f4b, 200F-01-03, rev 0xf2ee011, size 162401
4328823300    0x102049A04    Intel x86 or x64 microcode, sig 0xe49adb10, pf_mask 0xadd09d4b, 1A0F-01-04, rev 0xf3d8611, size 248801
4328823812    0x102049C04    Intel x86 or x64 microcode, sig 0xe49c9d10, pf_mask 0xadd5e34b, 1C0F-01-04, rev 0xf3da411, size 249492
4328824324    0x102049E04    Intel x86 or x64 microcode, sig 0xe49e5f10, pf_mask 0xaddb294b, 1E0F-01-04, rev 0xf3dc211, size 250183
4328824836    0x10204A004    Intel x86 or x64 microcode, sig 0xe4a02110, pf_mask 0xade06f4b, 200F-01-04, rev 0xf3de011, size 250874
4328888836    0x102059A04    Intel x86 or x64 microcode, sig 0xe57bdb10, pf_mask 0xb0739d4b, 1A0F-01-05, rev 0xf4c8611, size 337274
4328889348    0x102059C04    Intel x86 or x64 microcode, sig 0xe57d9d10, pf_mask 0xb078e34b, 1C0F-01-05, rev 0xf4ca411, size 337966
4328889860    0x102059E04    Intel x86 or x64 microcode, sig 0xe57f5f10, pf_mask 0xb07e294b, 1E0F-01-05, rev 0xf4cc211, size 338657
4328890372    0x10205A004    Intel x86 or x64 microcode, sig 0xe5812110, pf_mask 0xb0836f4b, 200F-01-05, rev 0xf4ce011, size 339348
4328954372    0x102069A04    Intel x86 or x64 microcode, sig 0xe65cdb10, pf_mask 0xb3169d4b, 1A0F-01-06, rev 0xf5b8611, size 425748
4328954884    0x102069C04    Intel x86 or x64 microcode, sig 0xe65e9d10, pf_mask 0xb31be34b, 1C0F-01-06, rev 0xf5ba411, size 426439
4328955396    0x102069E04    Intel x86 or x64 microcode, sig 0xe6605f10, pf_mask 0xb321294b, 1E0F-01-06, rev 0xf5bc211, size 427130
4328955908    0x10206A004    Intel x86 or x64 microcode, sig 0xe6622110, pf_mask 0xb3266f4b, 200F-01-06, rev 0xf5be011, size 427822
4329019908    0x102079A04    Intel x86 or x64 microcode, sig 0xe73ddb10, pf_mask 0xb5b99d4b, 1A0F-01-07, rev 0xf6a8611, size 847252
4329020420    0x102079C04    Intel x86 or x64 microcode, sig 0xe73f9d10, pf_mask 0xb5bee34b, 1C0F-01-07, rev 0xf6aa411, size 854164
4329020932    0x102079E04    Intel x86 or x64 microcode, sig 0xe7415f10, pf_mask 0xb5c4294b, 1E0F-01-07, rev 0xf6ac211, size 861076
4329021444    0x10207A004    Intel x86 or x64 microcode, sig 0xe7432110, pf_mask 0xb5c96f4b, 200F-01-07, rev 0xf6ae011, size 867988
4329085444    0x102089A04    Intel x86 or x64 microcode, sig 0xe81edb10, pf_mask 0xb85c9d4b, 1A0F-01-08, rev 0xf798611, size 173198
4329085956    0x102089C04    Intel x86 or x64 microcode, sig 0xe8209d10, pf_mask 0xb861e34b, 1C0F-01-08, rev 0xf79a411, size 173890
4329086468    0x102089E04    Intel x86 or x64 microcode, sig 0xe8225f10, pf_mask 0xb867294b, 1E0F-01-08, rev 0xf79c211, size 174581
4329086980    0x10208A004    Intel x86 or x64 microcode, sig 0xe8242110, pf_mask 0xb86c6f4b, 200F-01-08, rev 0xf79e011, size 175272
4329150980    0x102099A04    Intel x86 or x64 microcode, sig 0xe8ffdb10, pf_mask 0xbaff9d4b, 1A0F-01-09, rev 0xf888611, size 261672
4329151492    0x102099C04    Intel x86 or x64 microcode, sig 0xe9019d10, pf_mask 0xbb04e34b, 1C0F-01-09, rev 0xf88a411, size 262363
4329152004    0x102099E04    Intel x86 or x64 microcode, sig 0xe9035f10, pf_mask 0xbb0a294b, 1E0F-01-09, rev 0xf88c211, size 263054
4329152516    0x10209A004    Intel x86 or x64 microcode, sig 0xe9052110, pf_mask 0xbb0f6f4b, 200F-01-09, rev 0xf88e011, size 263746
```

“Intel x86 or x64 microcode”

Lets just show the file instead...

Conclusion: it is *not* empty

Throwing stuff against the wall

- Try if it boots
- See if it interacts with QNX
- Pull it through Sleuthkit
- (all the) YARA rules
- `*strings*` search
- (Static) analysis tools (capa)
- DIE (Detect it Easy)
- Gzip extraction
- MPEG frame retrieval
- Attempt to find key material

YARA Rules

CRC32c_poly_Constant
CRC32_poly_Constant
CRC32b_poly_Constant
BLOWFISH_Constants
MD5_Constants
RC6_Constants
RIPEMD160_Constants
SHA1_Constants
SHA512_Constants
TEAN
Qemu_Detection
VBox_Detection
VMWare_Detection
VM_Detect_VirtualPC_XEN_W
ine
vmdetect
maldoc_function_prolog_signature
maldoc_structured_exception_handling
maldoc_find_kernel32_base_method_1
maldoc_OLE_file_magic_number
maldoc_suspicious_strings
PEiD_00321_BopCrypt_v1_0_
PEiD_00810_FSG_v1_10__Eng_____dulek_xt_____M
0___7_0__
PEiD_01272_Neolite_v2_0_
PEiD_01400_Obsidium_v1_0_0_61_
PEiD_01410_Obsidium_V1_3_0_0____Obsidium_Soft
PEiD_01526_PE_PACK_v1_0_by_ANAKiN_1998______
PEiD_01597_PEEncrypt_v4_0b__JunkCode__
PEiD_01628_PEQuake_V0_06____forgat_
PEiD_01635_PESHiELD_v0_25_
PEiD_01693_pex_V0_99____params_

```
/*
XORSearch wildcard rule(s):
    Find kernel32 base method 1:10:648B(B;00???101)30000000
    Find kernel32 base method 1bis:10:64A130000000
*/
rule maldoc_find_kernel32_base_method_1
{
    meta:
        author = "Didier Stevens (https://DidierStevens.com)"
    strings:
        $a1 = {64 8B (05|0D|15|1D|25|2D|35|3D) 30 00 00 00}
        $a2 = {64 A1 30 00 00 00}
    condition:
        any of them
}
```

Binocle

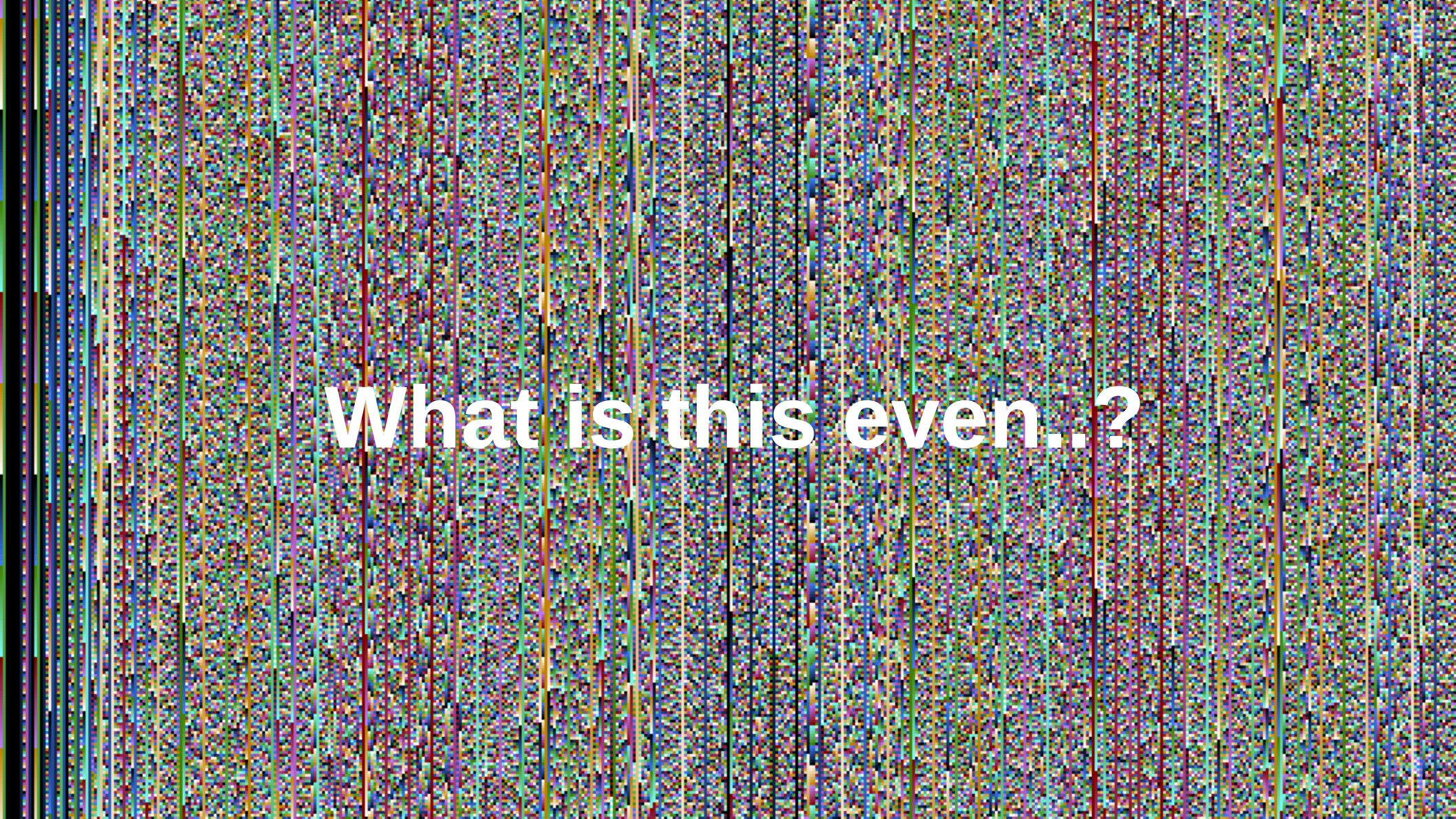
https://github.com/sharkdp/binocle

Binocle

https://github.com/sharkdp/binocle

Demonstration Time! :)

SEIZURE WARNING


What is this even..?

Seems like a pattern…

Seems like a pattern…

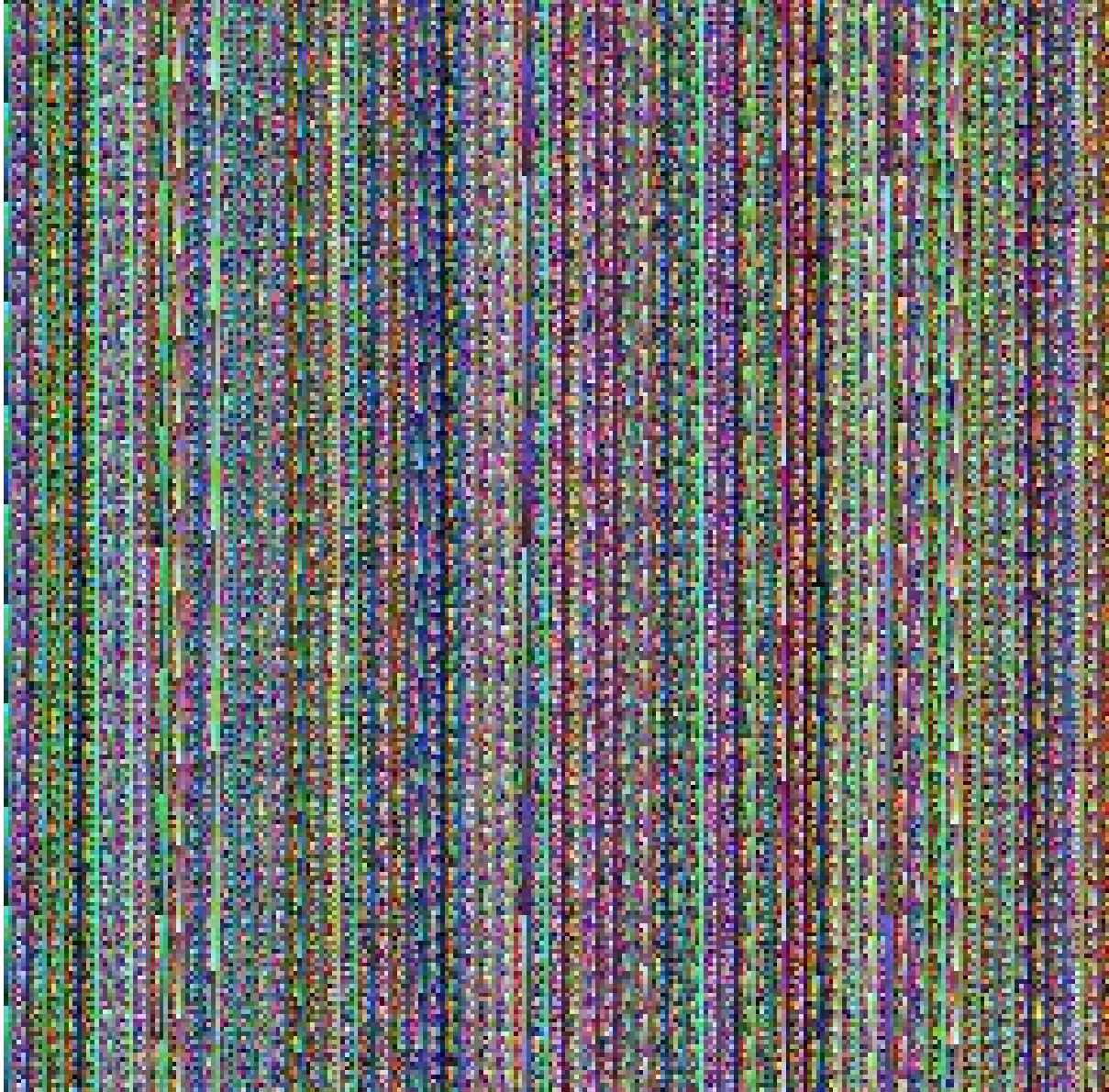
What about steganography?

https://cyberchef.io

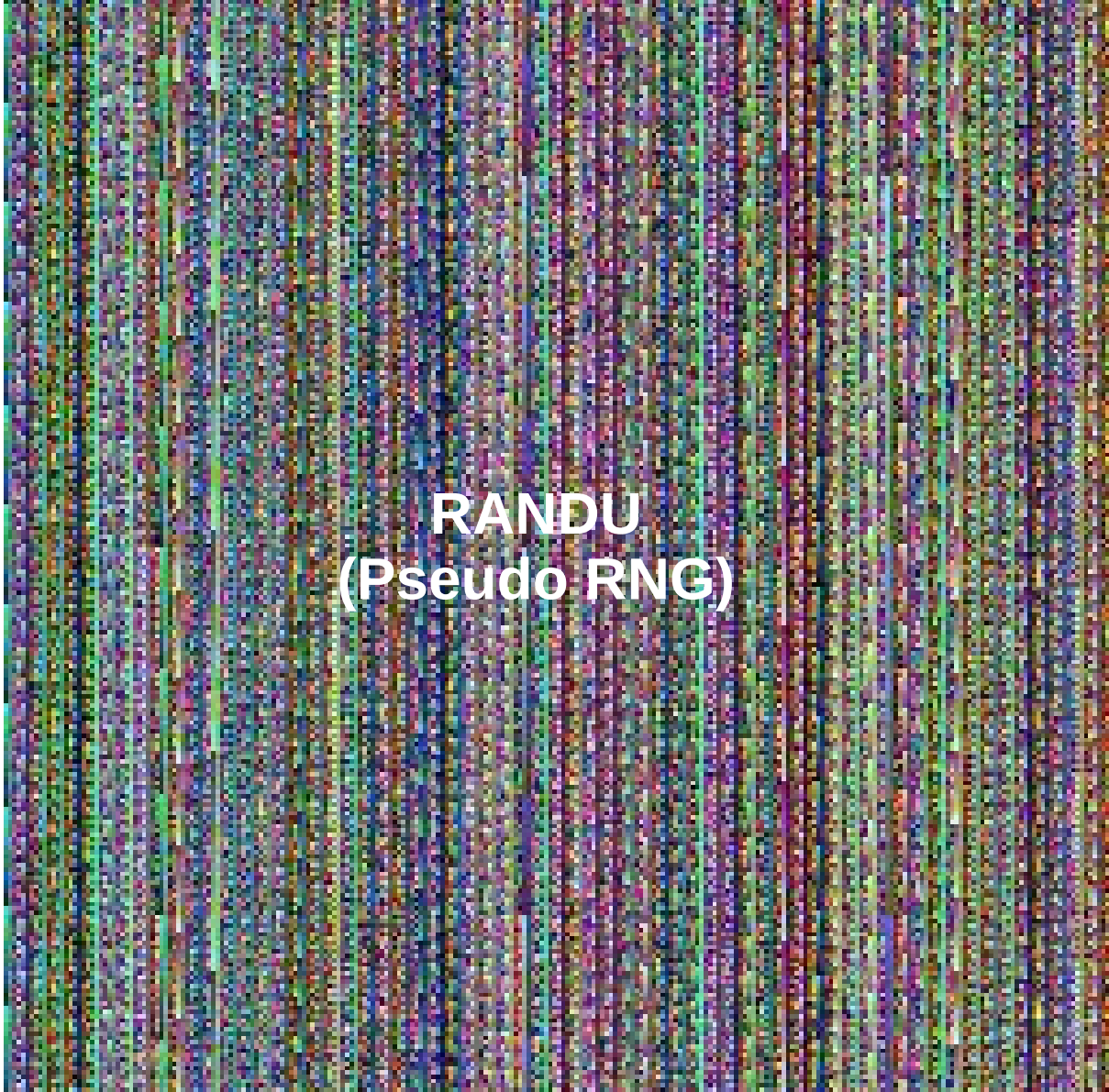
Does it execute?

https://github.com/corstian/brainfuck-interpreter

Reverse image search



Reverse image search



Pseudo-random number generators (PRNGs)!

- High-entropy
- Predictable contents

USB Dump shows vertical binary patterns.

Patterns can be data as well as code.

yara

Stick identified itself in a strange way.

Possibility of an attack on the USB controller?

USB Stick Analysis

Binwalk shows file signatures.

Most notably:

- Chunk of assembly code
- Chunk of encrypted data

Boot headers suggest relationship to QNX OS.

encryption

Blowfish

Unsuitable for brute force

USB Dump shows vertical binary patterns.

Patterns can be data as well as code.

Data

Code

Code can be found everywhere in the binary.

A large portion of memory is referenced at a very high memory location. Seemingly lookup tables?

Capabilities

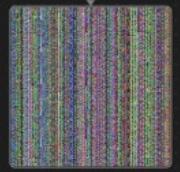
Suggested contents through analysis by capa

- TEA Encrypted portion
- CRC Checksum

RANDU is one of the closest things resembling the thing we're seeing here. See https://scruss.com/blog/tag/randu/ as well.

I'm becoming increasingly more convinced that the sequence is generated by a PRNG as a way to artificially increase entropy. See https://en.wikipedia.org/wiki/List_of_random_number_generators#Pseudorandom_number_generators_(PRNGs) as well

Pasted image 20240...



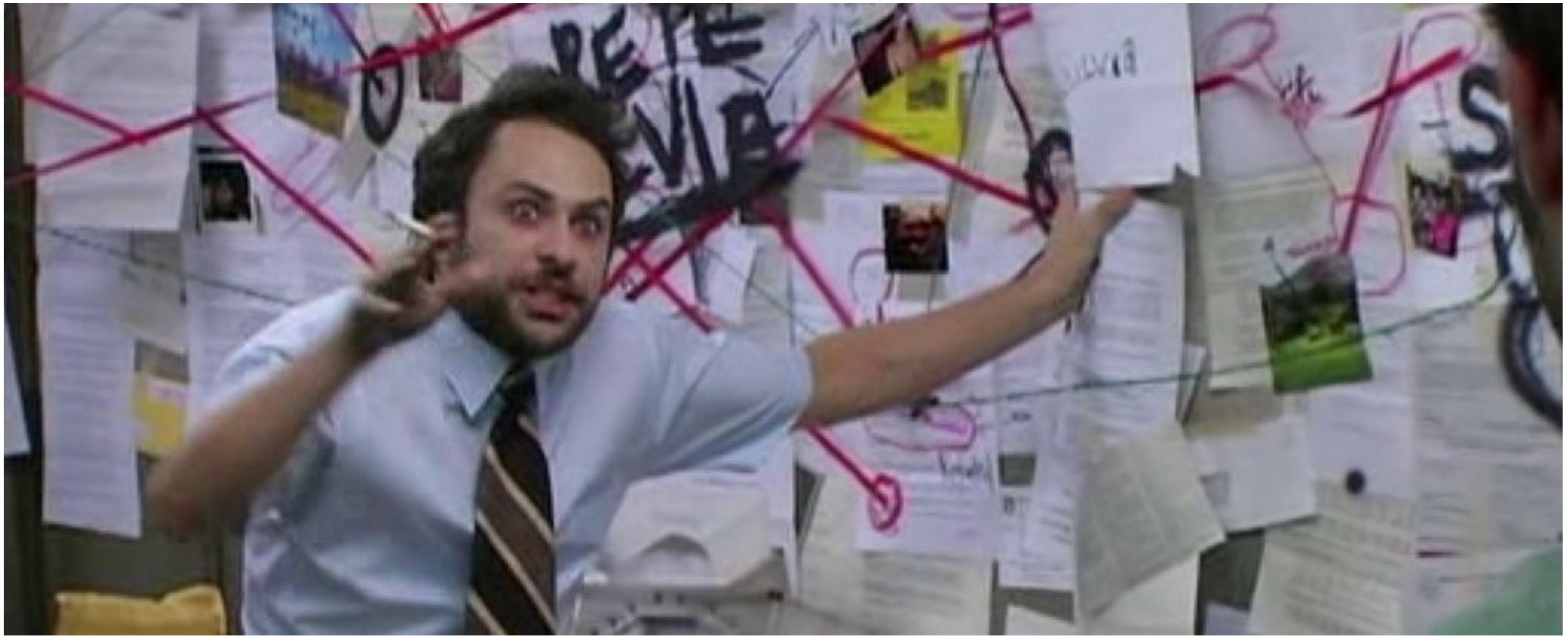
Running the file itself through binocle and radare similar patterns can be observed, and interestingly enough code apparently can be found. This is a strong signal that even in noise code can be hidden in a relatively straightforward manner.

The problem is that it is relatively difficult to identify the PRNG that had been used, and to filter out the noise.

There are various properties we can possibly derive from the sequence, though it might be interesting to generate random files and see whether the generated output matches the file we have.

The first 8 bytes seem to be an (unsigned) long, showing the offset at which it exists in the file. As such it increases by 512 for each line.


CRC32b_poly_Constant
CRC32b
Cryptanalysis
cryfind
code
para
crypto
Patterns
Det Code
Capabilities
output



So… there is a pattern

- Each row is 512 bytes (1 sector)
- Each row has 64, 64bit integers
- Each row has a constant offset:

```
512 0 7680 512 115200 15360 1728000 345600 25920000 6912000 ...
```

D	E	F	G	H	I	J	K	L
No	**Odd**	**Even**	**Odd Remainder**	**Even Remainder**	**Actual Odd**	**Actual Even**	**Actual – Calc Odd**	**Actual – Calc Even**
0	512	0	512	0	512	0	0	0
1	7680	512	7680	512	7680	512	0	0
2	115200	15360	115200	15360	115200	15360	0	0
3	1728000	345600	1728000	345600	1728000	345600	0	0
4	25920000	6912000	25920000	6912000	25920000	6912000	0	0
5	388800000	129600000	388800000	129600000	388800000	129600000	0	0
6	5832000000	2332800000	1537032704	2332800000	1537032704	-1962167295	0	-4294967295
7	87480000000	40824000000	1580654080	2169294336	1580654080	-2125672940	0	-4294967276
8	1312200000000	699840000000	2234974720	4055298048	-2.06E+09	-239668943	-4294967296	-4294966991
9	19683000000000	11809800000000	3459849728	2934903296	-835117568	-1360059418	-4294967296	-4294962714
10	295245000000000	196830000000000	358138368	238758912	358138368	238827654	0	68742
11	4428675000000000	3247695000000000	1077108224	3939522048	1077108224	-354414117	0	-4293936165
12	6.6430125E+016	5.31441E+016	3271721472	40396800	-1.023E+09	55863768	-4294967296	15466968
13	9.96451875E+017	8.63591625E+017	1831181824	3877673472	1831181824	-185289293	0	-4062962765
14	1.4946778125E+019	1.395032625E+019	1697923072	4161708032	1697923584	-948157565	512	-5109865597
15	2.24201671875E+020	2.24201671875E+020	3994025984	3994025984	-300950016	360462002	-4294976000	-3633563982
16	3.363025078125E+021	3.58722675E+021	4075814912	3774873600	-219282944	811012731	-4295097856	-2963860869
17	5.0445376171875E+022	5.7171426328125E+022	1006632960	570425344	1005723136	-938993853	-909824	-1509419197
18	7.56680642578125E+23	9.0801677109375E+023	0	0	-2.094E+09	-194282768	-2094022144	-194282768
19	1.13502096386719E+25	1.43769322089844E+25	0	0	-1.346E+09	-713296361	-1345561088	-713296361
20	1.70253144580078E+26	2.27004192773438E+26	0	0	1291420160	839895395	1291420160	839895395
21	2.55379716870117E+27	3.57531603618164E+27	0	0	-2.104E+09	1004949201	-2103534080	1004949201
22	3.83069575305176E+28	5.61835377114258E+28	0	0	-1.488E+09	85802054	-1488240128	85802054
23	5.74604362957764E+29	8.81060023201904E+29	0	0	-848765440	-201209309	-848765440	-201209309
24	8.61906544436646E+30	1.37905047109863E+31	0	0	153420288	428062233	153420288	428062233
25	1.29285981665497E+32	2.15476636109161E+32	0	0	-1.994E+09	-2015580809	-1993662976	-2015580809
26	1.93928972498245E+33	3.36143552330292E+33	0	0	159826432	2132363265	159826432	2132363265
27	2.90893458747368E+34	5.23608225745262E+34	0	0	-1.898E+09	2080504335	-1897570816	2080504335
28	4.36340188121052E+35	8.1450168449263E+035	0	0	1601208832	-754776855	1601208832	-754776855
29	6.54510282181578E+36	1.26538654555105E+37	0	0	-1.752E+09	-1130509396	-1751671296	-1130509396
30	9.81765423272367E+37	1.96353084654473E+38	0	0	-505265664	-1529443044	-505265664	-1529443044

D	E	F	G	H	I	J
No	**Odd**	**Even**	**Ulong col**	**ULONG**	**Actual**	**Remainder**
0	512	0	512	0	512	0
1	7680	512	2199023263232	512	2199023263232	0
2	115200	15360	65970697781760	15360	65970697781760	0
3	1728000	345600	1484340699225600	345600	1484340699225600	0
4	25920000	6912000	2.9686813975872E+016	6912000	2.9686813975872E+016	0
5	388800000	129600000	5.566277619504E+017	129600000	5.566277619504E+017	0
6	5832000000	2332800000	1.00192997139408E+19	2332800000	1.00192997139408E+19	0
7	87480000000	40824000000	1.75337744979384E+20	40824000000	9.31704831599804E+18	1.66020696663386E+20
8	1312200000000	699840000305	3.00578991374484E+21	699840000000	1.74173738038926E+19	2.98837253994095E+21
9	19683000000000	11809800000000	Err:502	11809800000000	1.26053333562426E+19	Err:502
10	295245000000000	196830000000000	Err:502	196830000000000	1.02575696366854E+18	Err:502
11	4428675000000000	3247695000000000	Err:502	3247695000000000	1.69245470330309E+19	Err:502
12	6.6430125E+016	5.31441E+016	Err:502	5.31441E+016	2.39933059863053E+17	Err:502
13	9.96451875E+017	8.63591625E+017	Err:502	8.63591625E+017	1.76509326218068E+19	Err:502
14	1.4946778125E+019	1.395032625E+019	Err:502	1.395032625E+019	1.43744383422775E+19	Err:502
15	2.24201671875E+020	2.24201671875E+020	Err:502	2.24201671875E+020	1.5481725140347E+018	Err:502
16	3.363025078125E+021	3.58722675E+021	Err:502	3.58722675E+021	3.48327316036033E+18	Err:502
17	5.0445376171875E+022	5.7171426328125E+022	Err:502	5.7171426328125E+022	1.44137961849352E+19	Err:502
18	7.56680642578125E+23	9.0801677109375E+023	Err:502	9.0801677109375E+023	1.76123059411741E+19	Err:502
19	1.13502096386719E+25	1.43769322089844E+25	Err:502	1.43769322089844E+25	1.53831595338081E+19	Err:502
20	1.70253144580078E+26	2.27004192773438E+26	Err:502	2.27004192773438E+26	3.60732325487742E+18	Err:502
21	2.55379716870117E+27	3.57531603618164E+27	Err:502	3.57531603618164E+27	4.31622395462776E+18	Err:502
22	3.83069575305176E+28	5.61835377114258E+28	Err:502	5.61835377114258E+28	3.68517018666353E+17	Err:502
23	5.74604362957764E+29	8.81060023201904E+29	Err:502	8.81060023201904E+29	1.758255667535E+019	Err:502
24	8.61906544436646E+30	1.37905047109863E+31	Err:502	1.37905047109863E+31	1.83851329154115E+18	Err:502
25	1.29285981665497E+32	2.15476636109161E+32	Err:502	2.15476636109161E+32	9.78989041891063E+18	Err:502
26	1.93928972498245E+33	3.36143552330292E+33	Err:502	3.36143552330292E+33	9.15843048652661E+18	Err:502
27	2.90893458747368E+34	5.23608225745262E+34	Err:502	5.23608225745262E+34	8.93569808040863E+18	Err:502
28	4.36340188121052E+35	8.1450168449263E+035	Err:502	8.1450168449263E+035	1.5205002167308E+019	Err:502
29	6.54510282181578E+36	1.26538654555105E+37	Err:502	1.26538654555105E+37	1.35912431926121E+19	Err:502
30	9.81765423272367E+37	1.96353084654473E+38	Err:502	1.96353084654473E+38	1.18778362224246E+19	Err:502

$$S_0 = 512$$

$$S_{odd} = S_0 \cdot 15^n$$

$$S_{even} = \frac{S_0}{15} \cdot n$$

$$S(n) = (S_0 \cdot 15^n) \ + \ (\frac{S_0}{15} \cdot n \cdot 2^{32}) \ mod \ 2^{64}$$

Steganography is ruled out!

The existence of malicious code however is not...

Linear Feedback Shift Registers (LFSRs)

- A type of PRNG generating an (eventually) repeating sequence
- A pattern can be generated on demand...

Hypothesis: pattern generation

1) Generate first x bytes (entry point)
2) Look for pattern no longer than x as generated by the LFSR(?)
3) Add a jump to specified offset after the entry point
4) Rinse and repeat

But what is the entry point..?

Normal boot:

1) MBR is executed
2) MBR defines jump to partition
3) Partition bootstraps further operations

Emulation / Simulation!

- BOCHS <3
- Das Blinkenlights
- R2 / QEMU

To run an emulator

```
qemu-system-x86_64 ../../VirtualBox\ VMs/Windows\ 10/Windows\ 10.vdi
-usb -device usb-storage,drive=stick -blockdev
file,filename=image.original.dd,node-name=stick -boot menu=on -S -s
```

To attach a debugger

```
r2 -d gdb://127.0.0.1:1234
```

```
                    INIT_READ_SECTOR                                XREF[1]:     0000:0641(j)
0000:065d bf 05 00      MOV         DI,0x5

                    READ_SECTOR                                     XREF[1]:     0000:0671(j)
0000:0660 bb 00 7c      MOV         BX,FAT_ENTRY_POINT
0000:0663 b8 01 02      MOV         AX,0x201
0000:0666 57            PUSH        DI
0000:0667 cd 13         INT         0x13                                        ah:2
0000:0669 5f            POP         DI
0000:066a 73 0c         JNC         VALIDATE
0000:066c 33 c0         XOR         AX,AX
0000:066e cd 13         INT         0x13
0000:0670 4f            DEC         DI
0000:0671 75 ed         JNZ         READ_SECTOR
0000:0673 be a3 06      MOV         error_message,0x6a3
0000:0676 eb d3         JMP         PRINT_MESSAGE
```

```
                    s_Invalid_partition_table_0000_068b             XREF[1]:     INIT:0000:064b(R)
0000:068b 49 6e 76        ds          "Invalid partition table"
          61 6c 69
          64 20 70 ...
0000:06a3 45 72 72        ds          "Error loading operating system"
          6f 72 20
          6c 6f 61 ...
0000:06c2 4d 69 73        ds          "Missing operating system"
          73 69 6e
          67 20 6f ...
```

```
                    ****************************************************************
                    *                          FUNCTION                            *
                    ****************************************************************
                    undefined __cdecl16near DO_EXTENDED_READ(void)
                      assume BP = 0x7c00
      undefined       <UNASSIGNED>   <RETURN>
                    DO_EXTENDED_READ                                XREF[5]:     BOOTSTRAP_3RD_STAGE:0000:7cd1(c),
                                                                                 0000:7d6a(j),
                                                                                 SET_READ_POSITION:0000:80d5(c),
                                                                                 SET_READ_POSITION:0000:813f(c),
                                                                                 TARGET:0000:81bf(c)
0000:7d00 66 60           PUSHAD
0000:7d02 80 7e 02 00     CMP         byte ptr [BP + 0x2]=>COUNTER,0x0                 = 90h
0000:7d06 0f 84 20 00     JZ          SKIP_EXT_READ
0000:7d0a 66 6a 00        PUSH        0x0
0000:7d0d 66 50           PUSH        EAX
0000:7d0f 06              PUSH        ES
0000:7d10 53              PUSH        BX=>DAT_0000_8200
0000:7d11 66 68 10        PUSH        0x10010
          00 01 00
0000:7d17 b4 42           MOV         AH,0x42                                      extended read sectors
0000:7d19 8a 56 40        MOV         DL,byte ptr [BP + 0x40]=>FAT32 header_0000_7c0... DL:0x80 - first disk
0000:7d1c 8b f4           MOV         SI,SP                                        0x7bc2
                                                                                   0x7bbe
0000:7d1e cd 13           INT         0x13
0000:7d20 66 58           POP         EAX
0000:7d22 66 58           POP         EAX
0000:7d24 66 58           POP         EAX
0000:7d26 66 58           POP         EAX
0000:7d28 eb 33           JMP         READ_NEXT_SECTOR
```

```
                    LOAD_POTENTIAL_BOOTMGR_TO_8200                  XREF[1]:     0000:80f1(j)
0000:80cd bb 00 82      MOV         BX,DAT_0000_8200
0000:80d0 8b fb         MOV         DI,BX                                        di:0x8200
0000:80d2 b9 01 00      MOV         CX,0x1                                       cx:0x1
0000:80d5 e8 28 fc      CALL        DO_EXTENDED_READ                             undefined DO_EXTENDED_READ(vo

                    SEARCH_BOOTMGR                                  XREF[1]:     0000:80ee(j)
0000:80d8 38 2d         CMP         byte ptr [DI],CH
                    Triggered @ di:0x83a0
0000:80da 74 1e         JZ          BOOTMGR_NOT_FOUND
0000:80dc b1 0b         MOV         CL,0xb                                       cl:0xb
0000:80de 56            PUSH        SI
0000:80df be 6d 7d      MOV         SI,s_BOOTMGR_0000_7d6d                       "BOOTMGR"
0000:80e2 f3 a6         CMPSB.REPE  ES:DI,SI                                     di:0x8202
0000:80e4 5e            POP         SI
0000:80e5 74 1b         JZ          BOOTMGR_FOUND
0000:80e7 03 f9         ADD         DI,CX                                        di:0x820b
                                                                                   :0x822b
                                                                                   :0x824b
                                                                                   :0x826b
                                                                                   :0x828b
                                                                                   ..etc
0000:80e9 83 c7 15      ADD         DI,0x15                                      di:0x8220
                                                                                   :0x8240
                                                                                   :0x8260
                                                                                   :0x8280
                                                                                   :0x82a0
                                                                                   :0x82c0
0000:80ec 3b fb         CMP         DI,BX                                        bx:0x8400
0000:80ee 72 e8         JC          SEARCH_BOOTMGR
0000:80f0 4e            DEC         SI
0000:80f1 75 da         JNZ         LOAD_POTENTIAL_BOOTMGR_TO_8200
0000:80f3 66 58         POP         EAX
0000:80f5 e8 65 00      CALL        CHECK_MASK                                   undefined2 CHECK_MASK(void)
0000:80f8 72 bf         JC          SET_READ_POSITION

                    BOOTMGR_NOT_FOUND                               XREF[1]:     0000:80da(j)
0000:80fa 83 c4 04      ADD         SP,0x4                                       sp:7bf4?
0000:80fd e9 f6 fb      JMP         WRITE_OUTPUT                                 undefined WRITE_OUTPUT(void)
                    -- Flow Override: CALL_RETURN (CALL_TERMINATOR)
```

```
                        LAB_0000_8155                                   XREF[1]:     0000:814f(j)
0000:8155 8a 56 40        MOV         DL,byte ptr [BP + 0x40]
0000:8158 ea 00 00        JMPF        LAB_2000_0000
          00 20
```

Security Operation Center:

“Interesting, not pursuing further”
“Policy not to boot from USB”

Hypothesis:

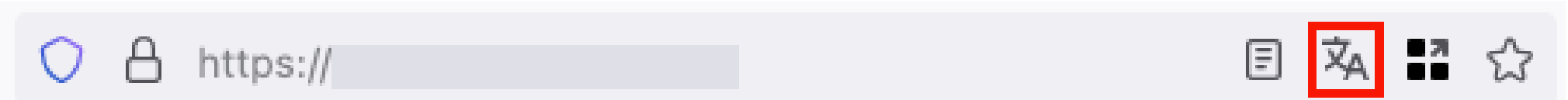
“This is a supply chain attack”

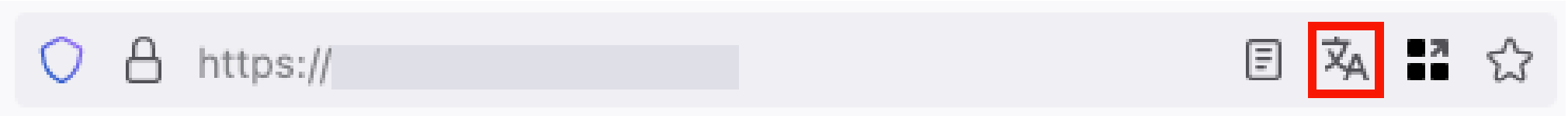
Back to hardware

Device ID

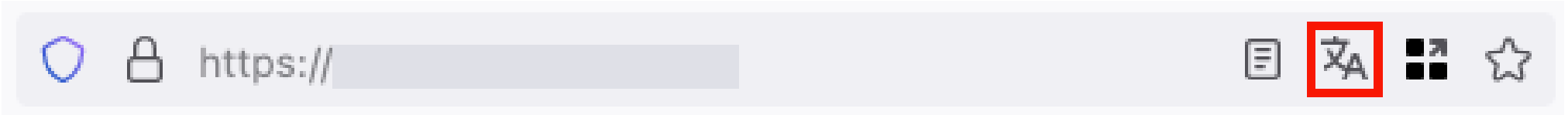
```
cat /sys/kernel/debug/usb/devices
```

```
T:  Bus=01 Lev=02 Prnt=02 Port=01 Cnt=02 Dev#=  4 Spd=480  MxCh= 0
D:  Ver= 2.00 Cls=00(>ifc ) Sub=00 Prot=00 MxPS=64 #Cfgs=  1
P:  Vendor=23a9 ProdID=ef18 Rev= 1.00
S:  Manufacturer=AI210
S:  Product=Mass Storage
C:* #Ifs= 1 Cfg#= 1 Atr=80 MxPwr=100mA
I:* If#= 0 Alt= 0 #EPs= 2 Cls=08(stor.) Sub=06 Prot=50 Driver=usb-storage
E:  Ad=02(O) Atr=02(Bulk) MxPS= 512 Ivl=0ms
E:  Ad=82(I) Atr=02(Bulk) MxPS= 512 Ivl=0ms
```


https://



https://flashboot.ru/iflash/page20/



https://flashboot.ru/iflash/page20/

https://www.usbdev.ru/f/index.php?topic=5231.0

Hi,

> Цитата: Javed sumra от июня 02, 2020, 07:39:13
> USB Device ID: VID = 23A9 PID = EF18

Highly likely the controller of your Flash Drive is one of the **SiliconGo SG1581/SG1580** family.

Pls, pay your attention for tools to reburn your UFD here:
SiliconGo SG1581/SG1580 MPTools V1.8.6.1.778_160509 – [USBDev.ru]
https://www.usbdev.ru/files/silicongo/sg15811580mptools/

SiliconGo SG1581/SG1580 MPTools V1.8.6.1.333_160224 is preferable...

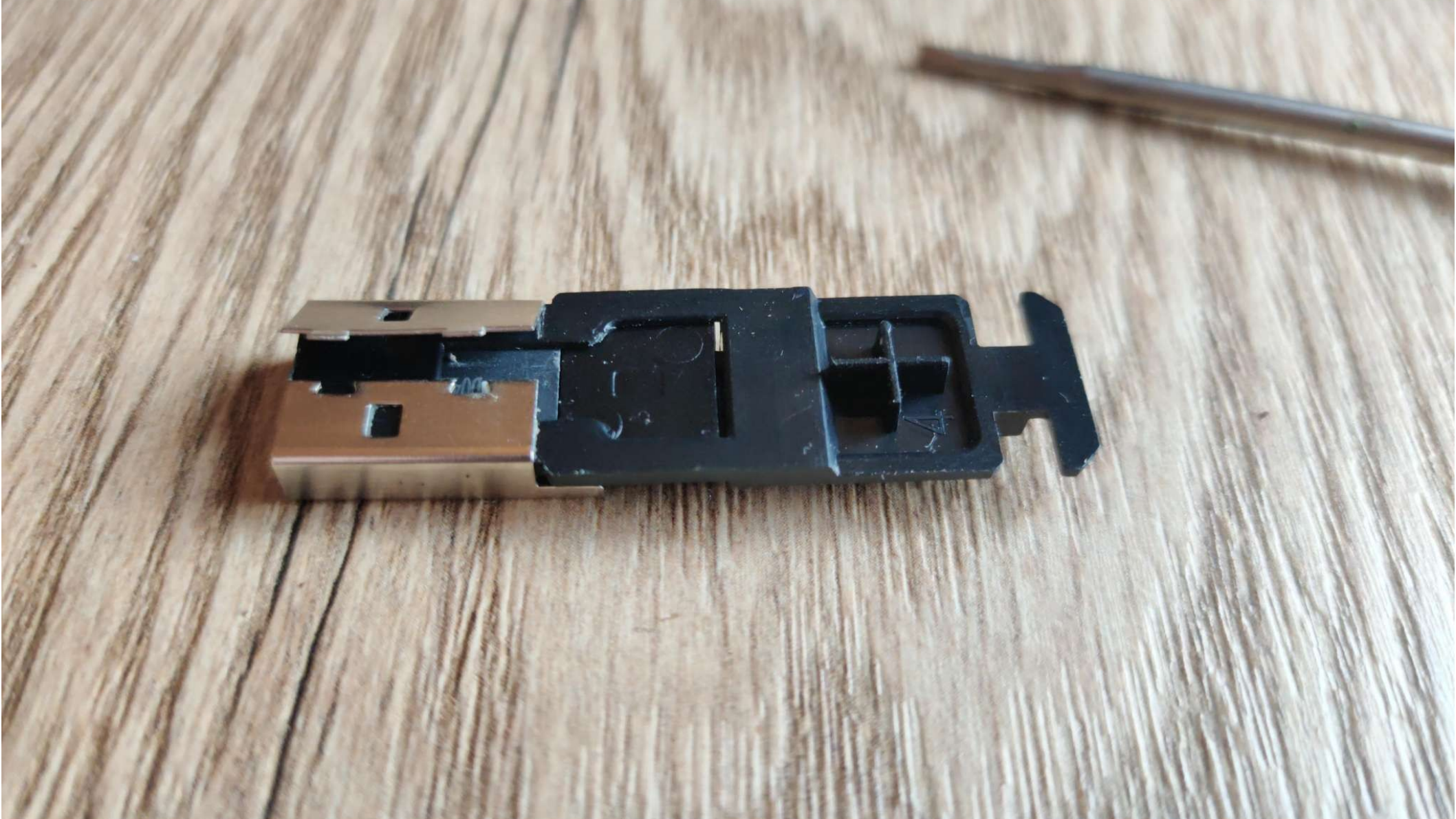
Maybe test-mode is needed, because of **FID** & **IC model** are undetected by **ChipGenius**.

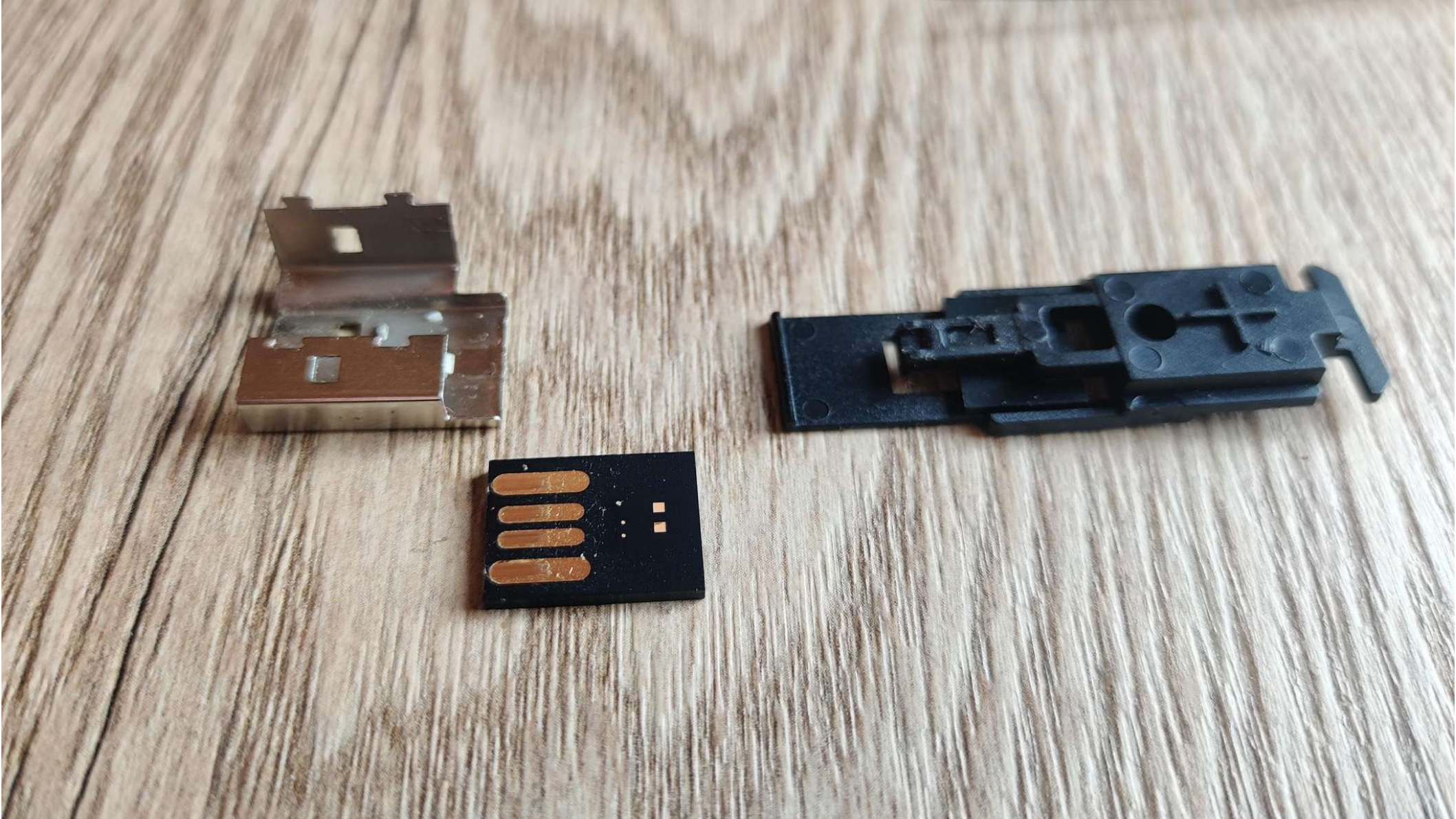
> Цитата: Javed sumra от июня 02, 2020, 07:39:13
> Controller Part-Number: Unknown

For test-mode look here:
Перевод флешки в тестовый режим – [USBDev.ru]
https://www.usbdev.ru/articles/testmod/

Good luck!

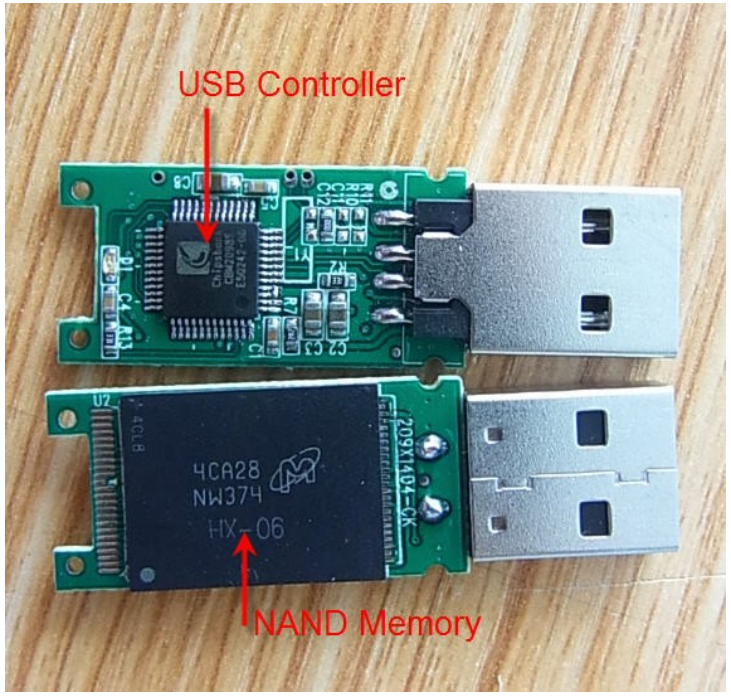
- Цитировать

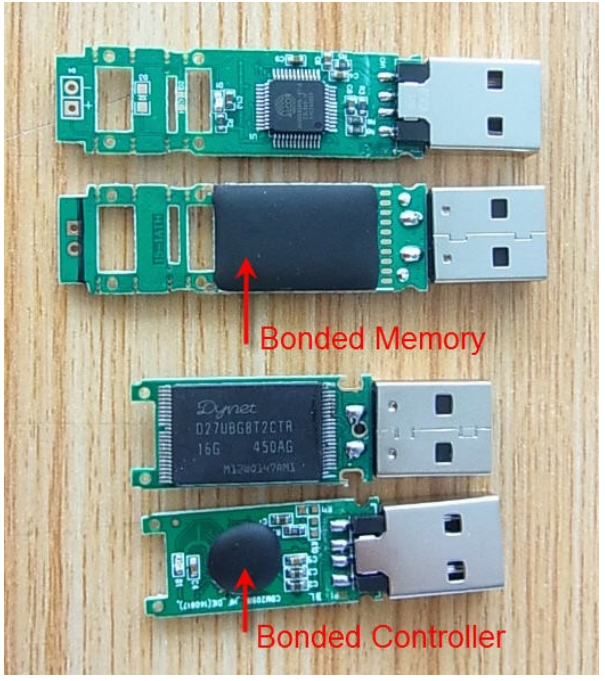


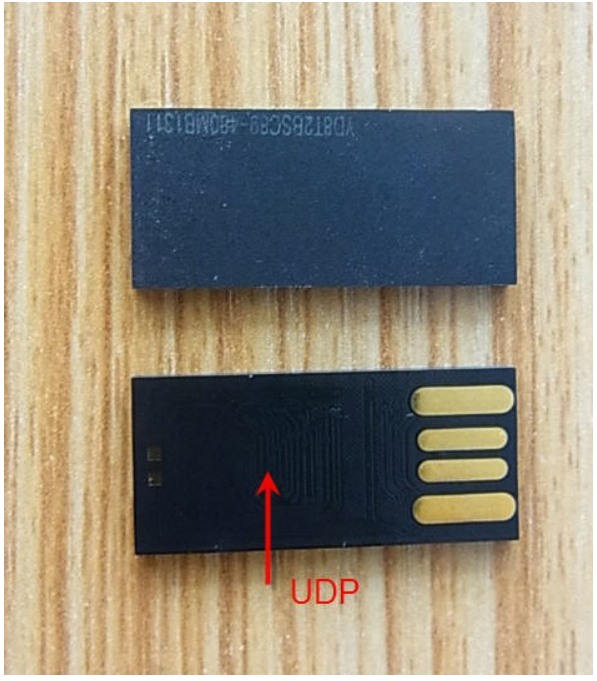


1/128in
0.05mm

PCB vs COB vs UDP



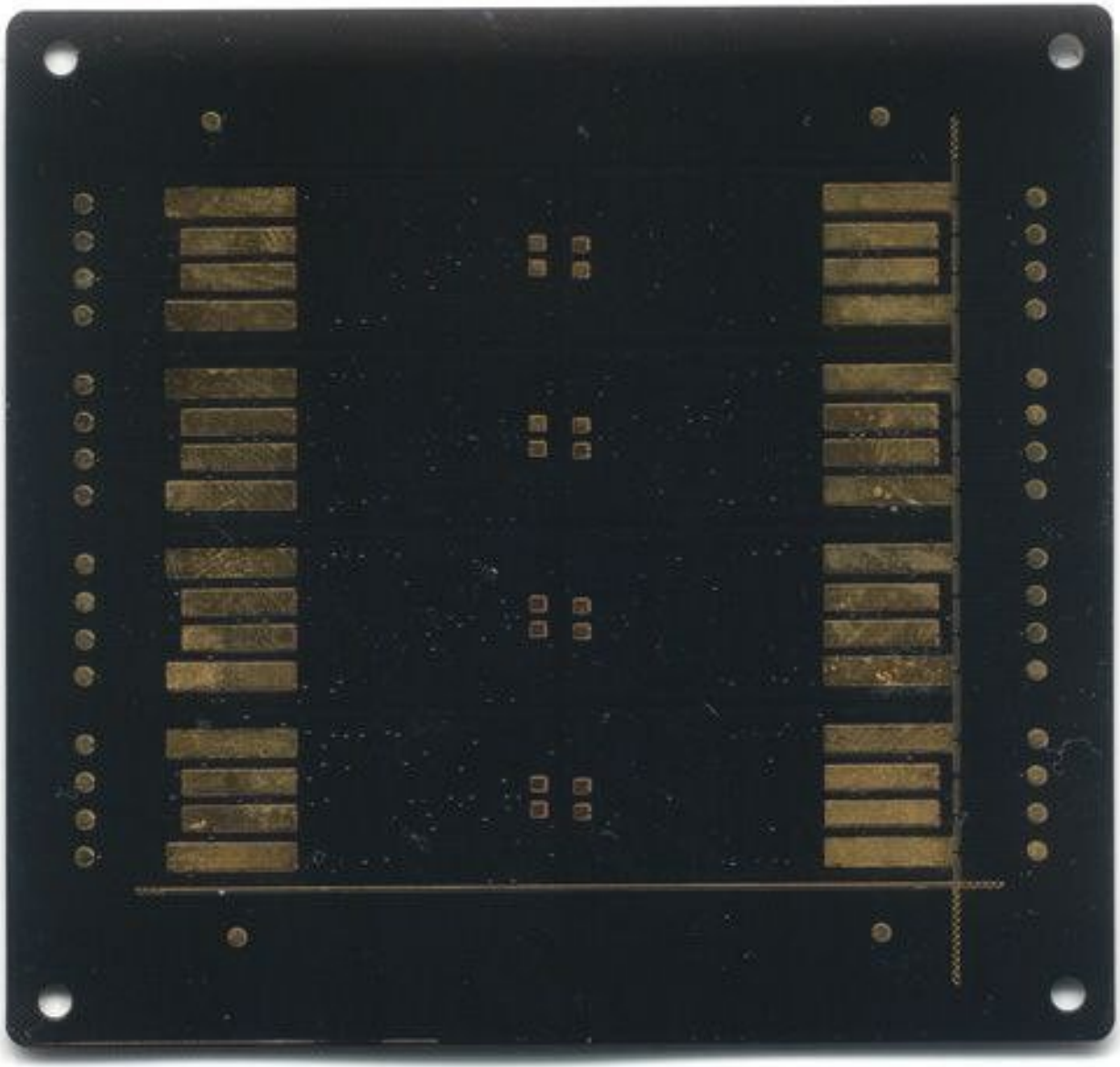


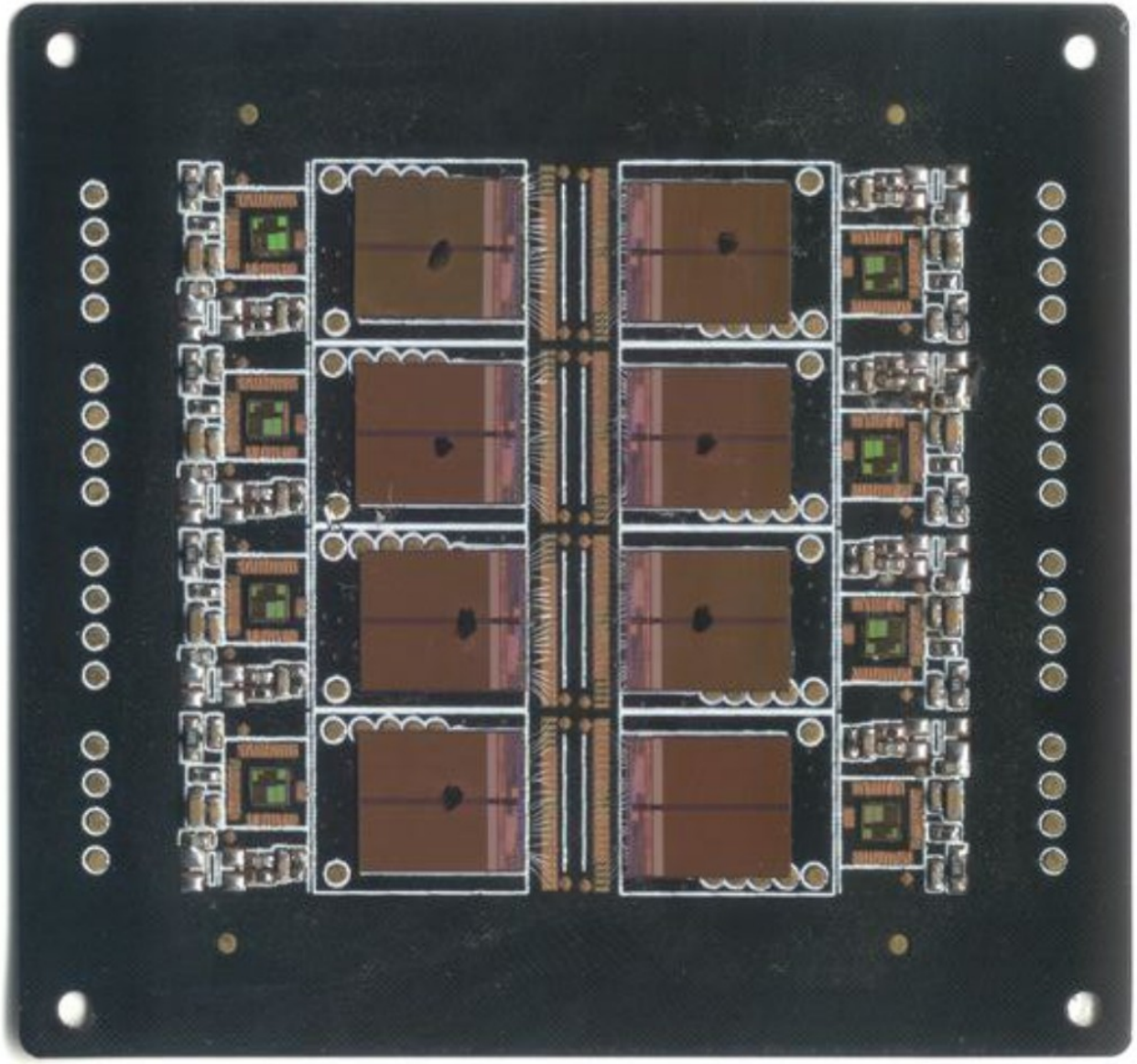


https://www.getusb.info/usb-flash-drives-pcb-cob-udp/

Manufacturing process

https://www.bunniestudios.com/blog/2013/where-usb-memory-sticks-are-born/



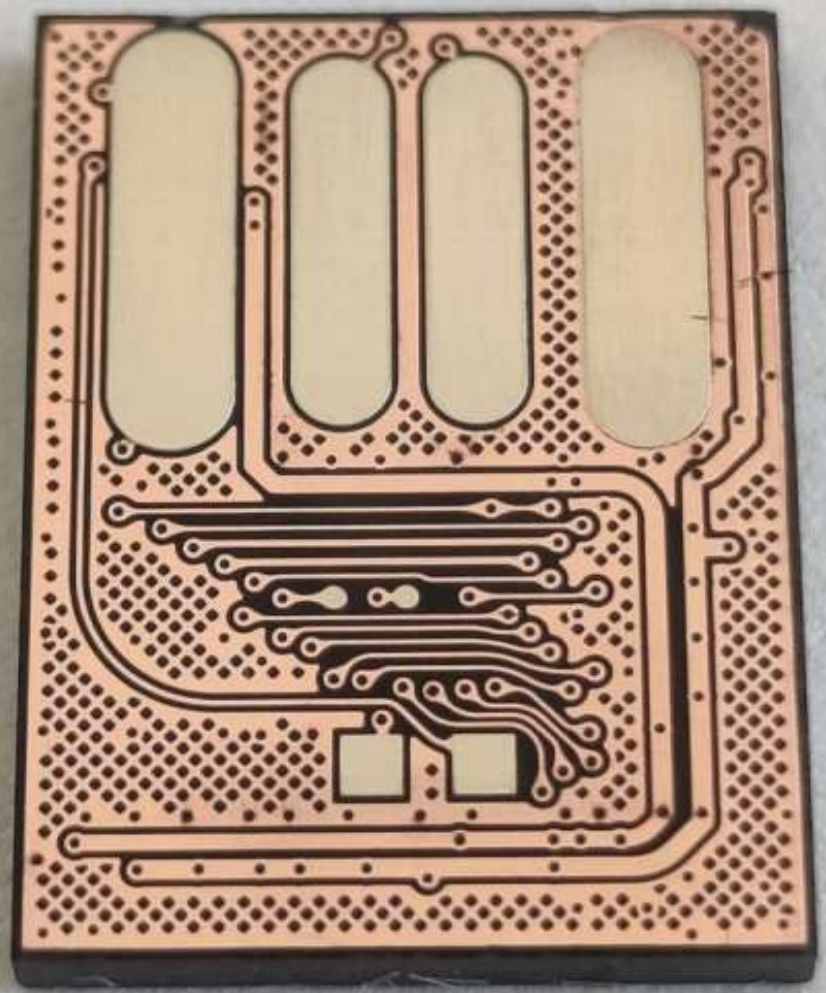


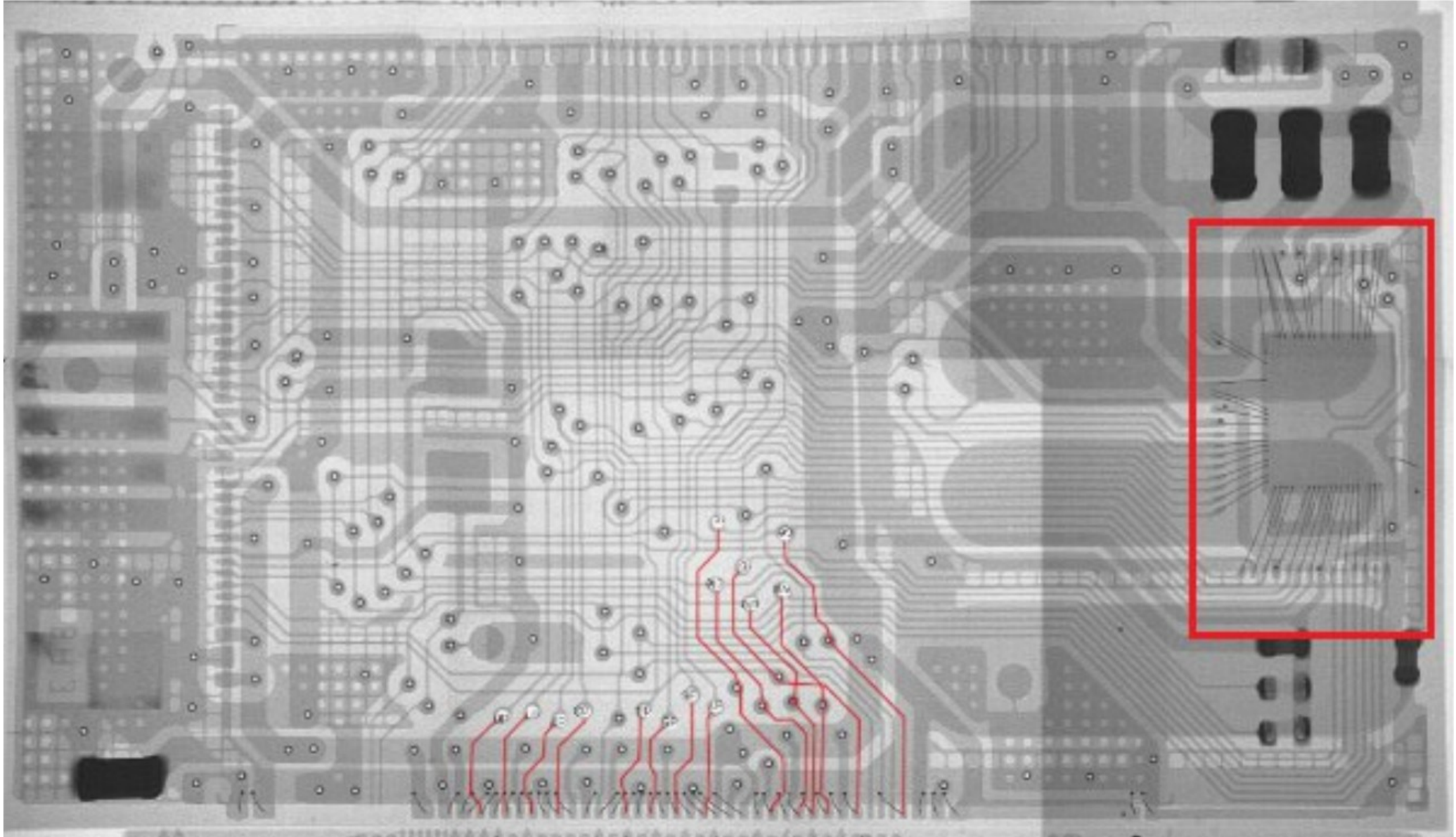
FR4 is nasty stuff!

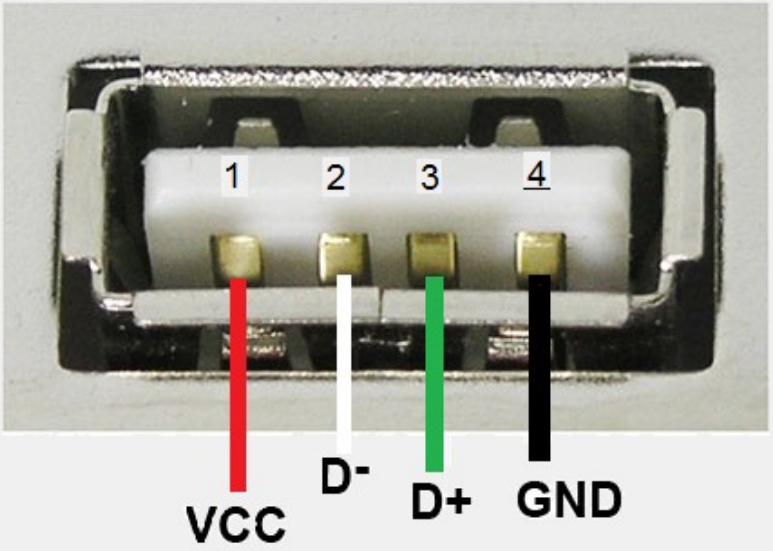
And it does not dissolve :’(

(without nasty, nasty, nasty stuff)

(I do not trust myself with)





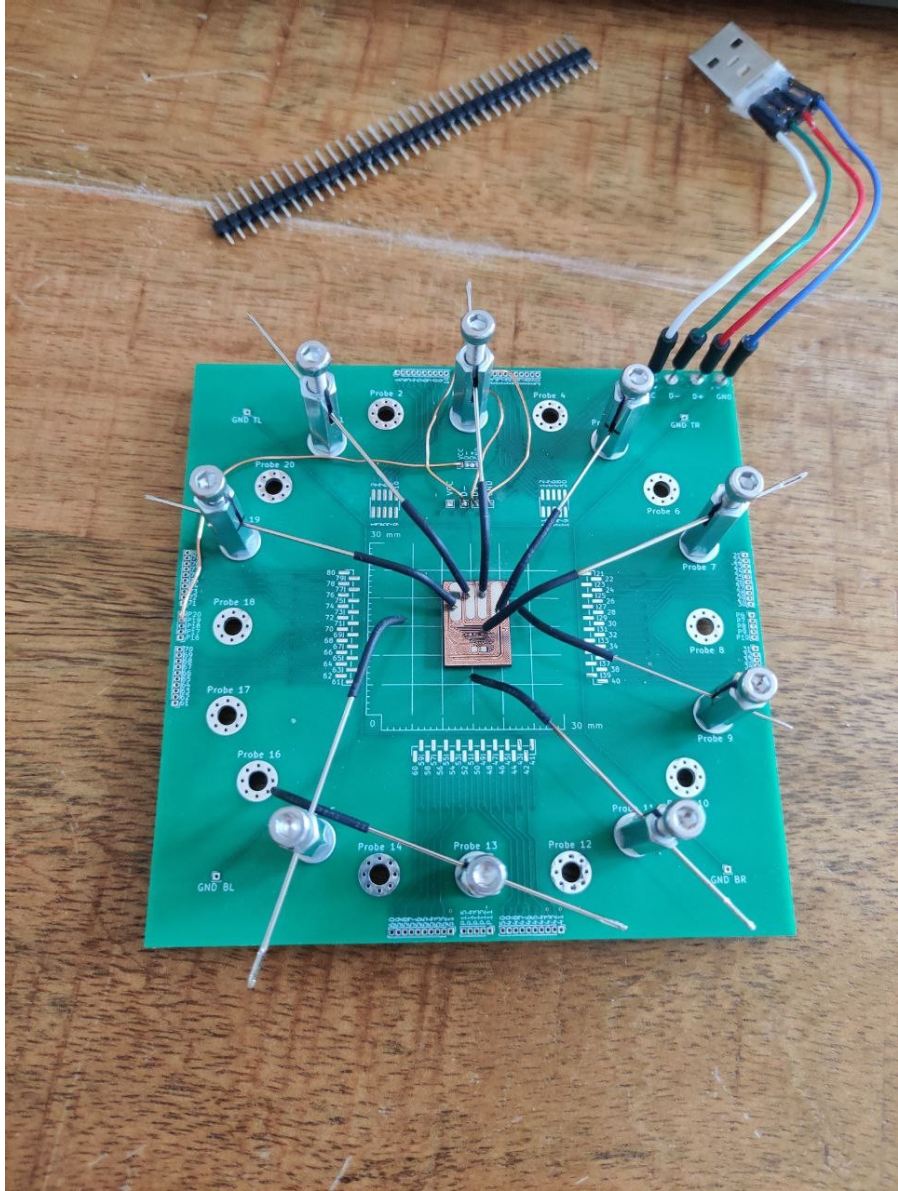


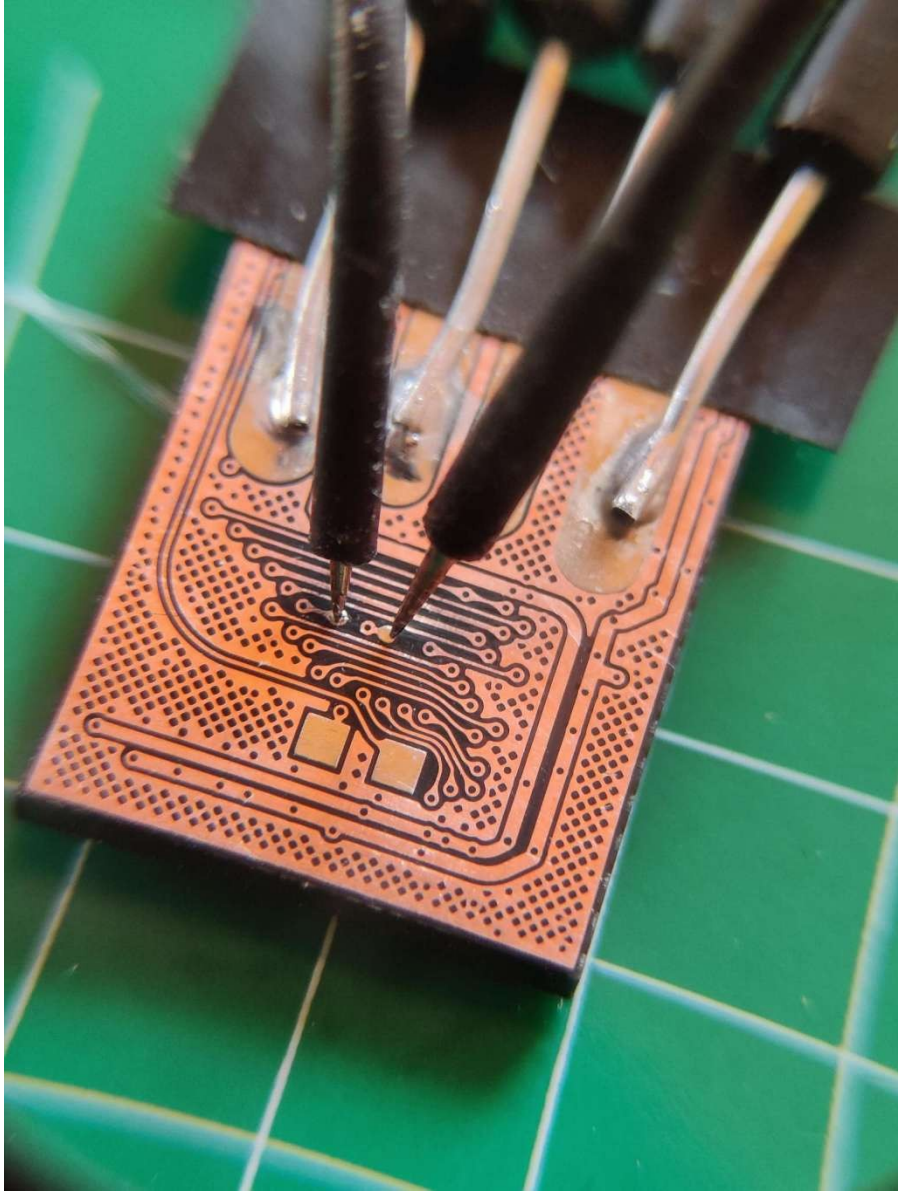
The USB Pinout: front female face

Pin	Name	Cable color	Description
1	VCC	Red	+5 VDC
2	D-	White	Data -
3	D+	Green	Data +
4	GND	Black	Ground

Probe set-up

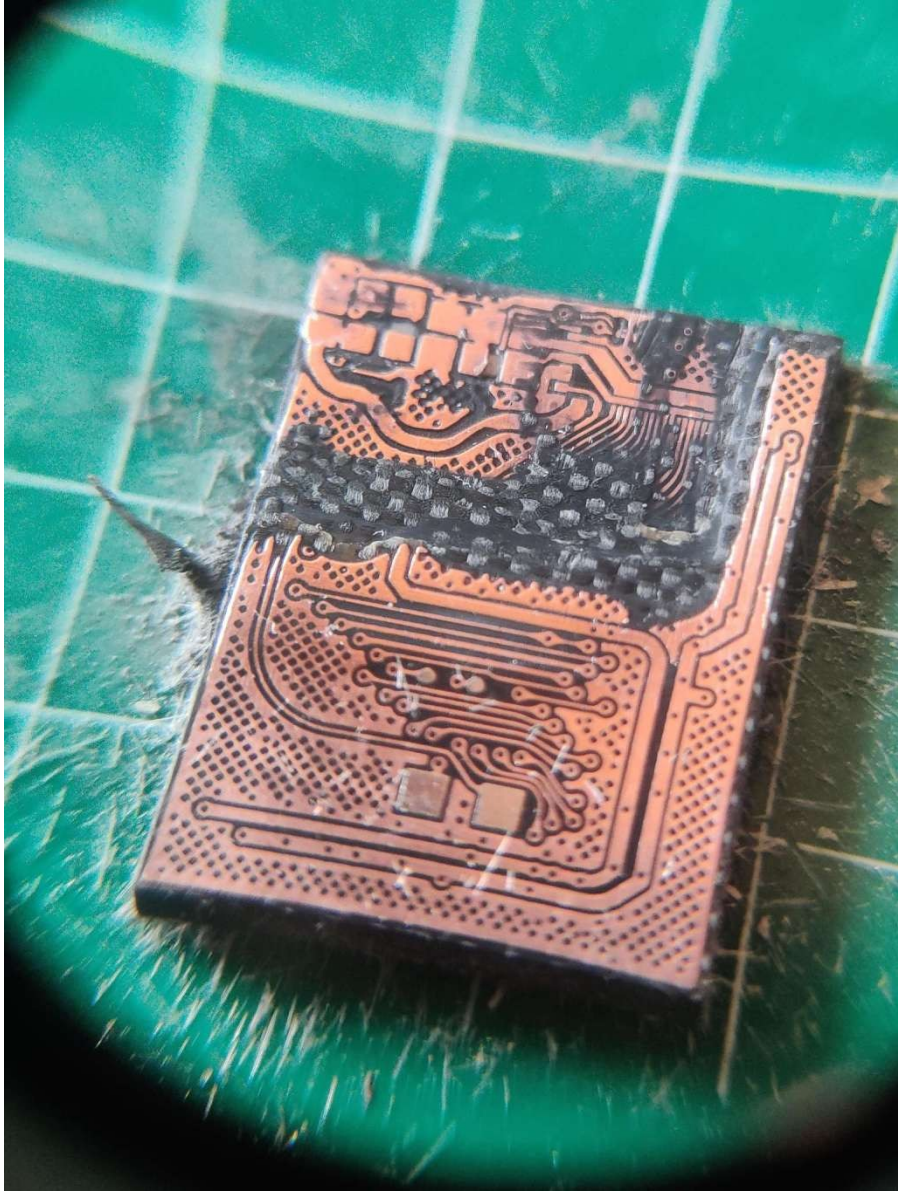
- Designed a PCB
- Created probes from needles and shrink tubes





Shorting the thing

- Wanted direct access to the controller
- Can probably repair the thing – if I reaaaly feel like it
- Skill issue tbh



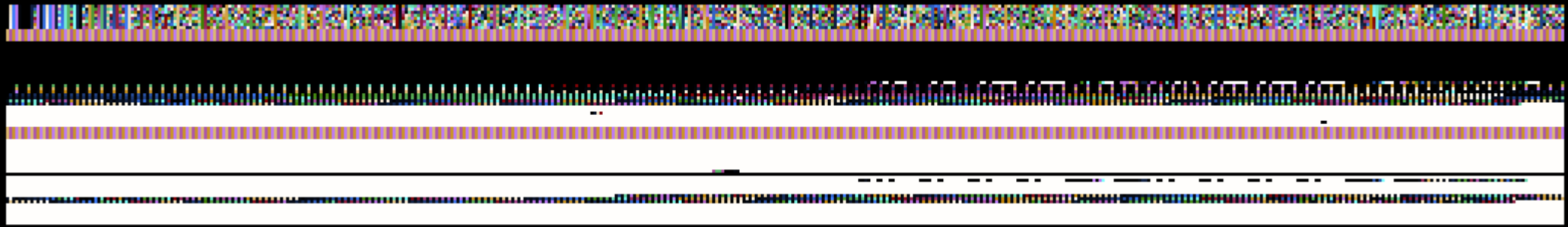
There are more USBs out there!

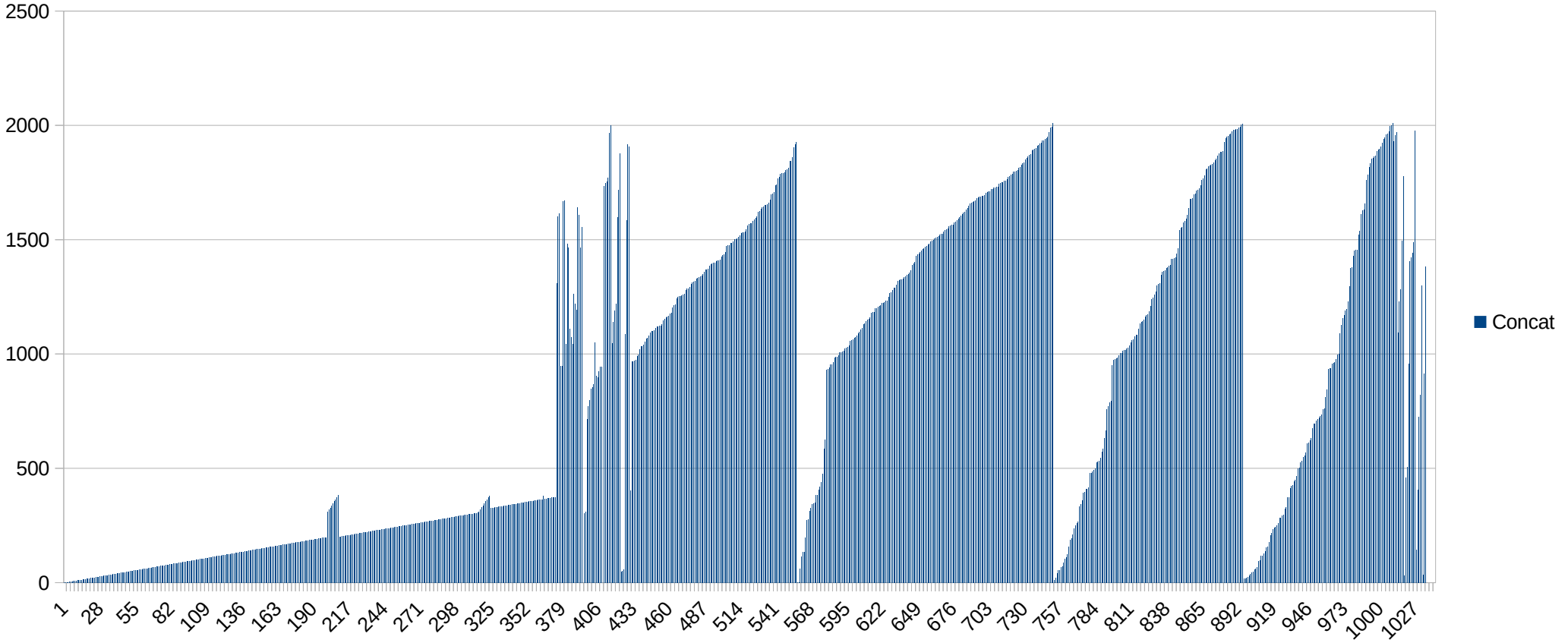
They just do not want to give ‘em to me...

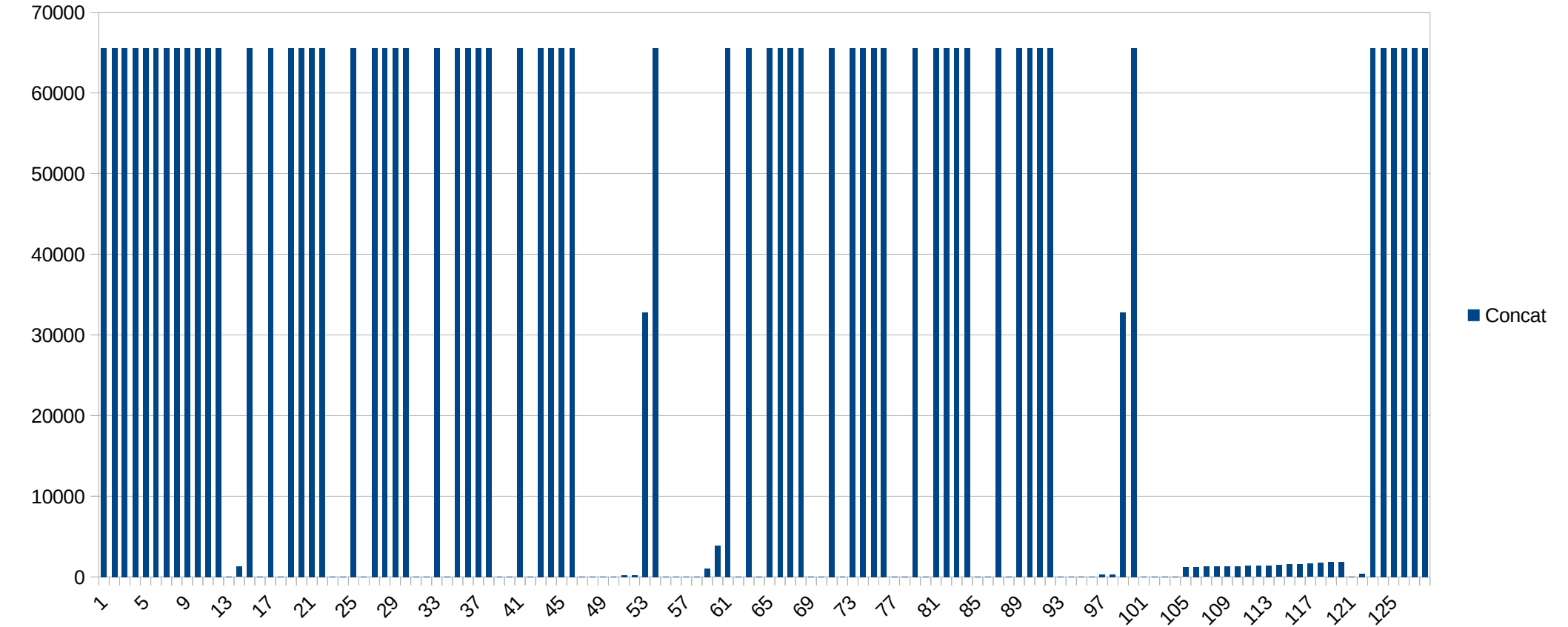
Open questions

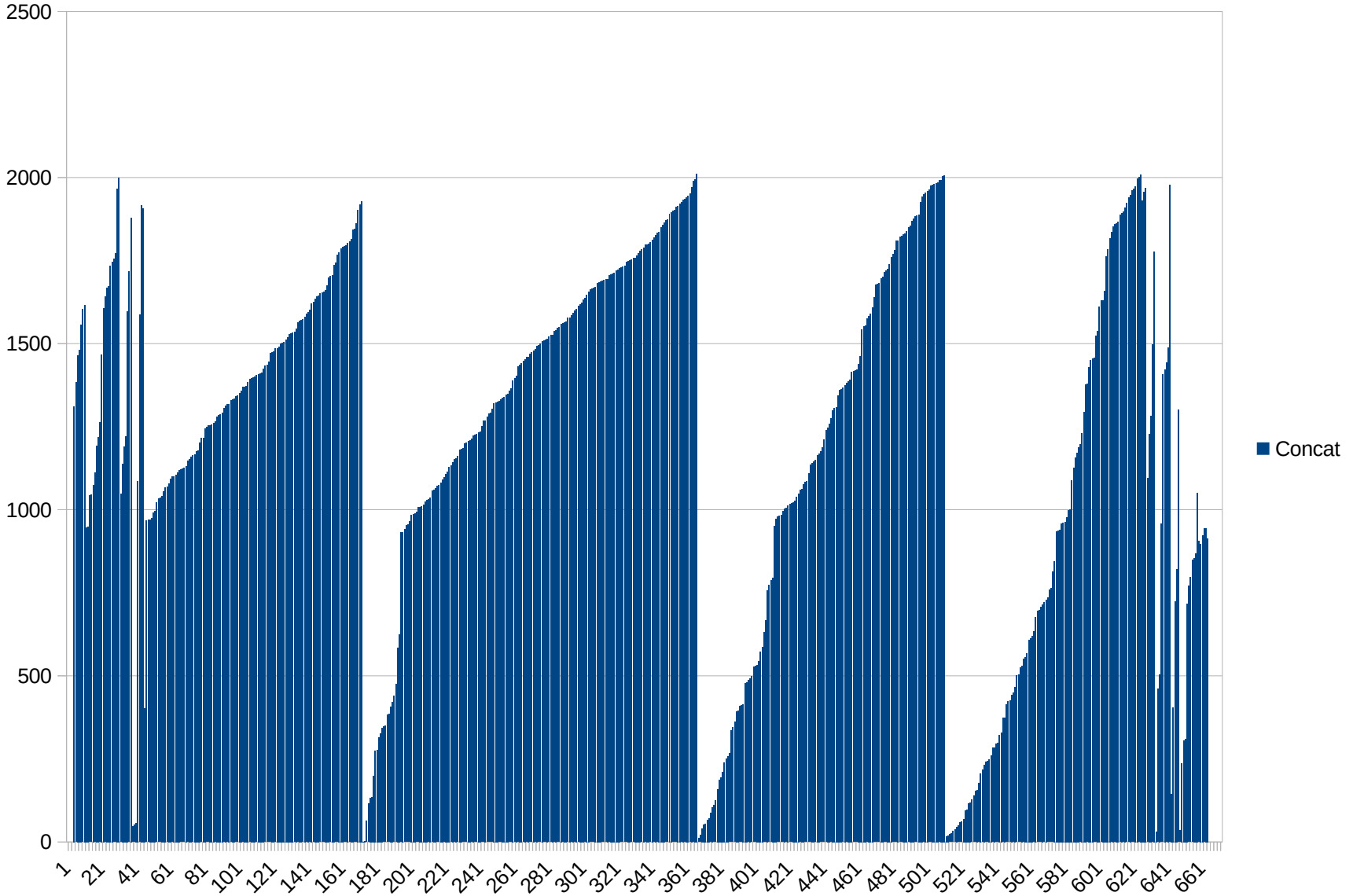
- Whom is targeted? (Windows? BOOTMGR?)
- Where is the jump to the LFSR?
- Why does the big blob indicate microcode and blowfish?
- What is the thing at the end..?

The end:




2500
2000
1500
1000
500
0
1
28
55
82
109
136
163
190
217
244
271
298
325
352
379
406
433
460
487
514
541
568
595
622
649
676
703
730
757
784
811
838
865
892
919
946
973
1000
1027
Concat


70000
60000
50000
40000
30000
20000
10000
0
1
5
9
13
17
21
25
29
33
37
41
45
49
53
57
61
65
69
73
77
81
85
89
93
97
101
105
109
113
117
121
125
Concat


2500
2000
1500
1000
500
0
1
21
41
61
81
101
121
141
161
181
201
221
241
261
281
301
321
341
361
381
401
421
441
461
481
501
521
541
561
581
601
621
641
661
Concat

Did you know…

You can bootstrap a lisp evaluator within 512 bytes?

https://github.com/jart/sectorlisp

Bonus: high level USB malware design

- Boot on start; aim for a bootkit
- Be stealthy – like for real
- Obfuscate the main payload

- Use a lisp-like language…
- Hide a needle in a haystack

Lessons learned:

Think before doing

And so much more...

Find me

- Corstian Boerman
- Mastodon: @corstian@sunny.garden
- Blog: https://disintegrated.parts

Available for interesting projects.