De ‘branch’ en ‘branch’

Récupération d’un FW d’ECU sur mémoire FAT “nettoyée”

Philippe AZALBERT - *@Phil_BARR3TT*

Quarkslab

Contexte

- Analyse en **boîte noire** d'un calculateur additionnel d'auto-partage
- **1 seul équipement** disponible pour l'audit

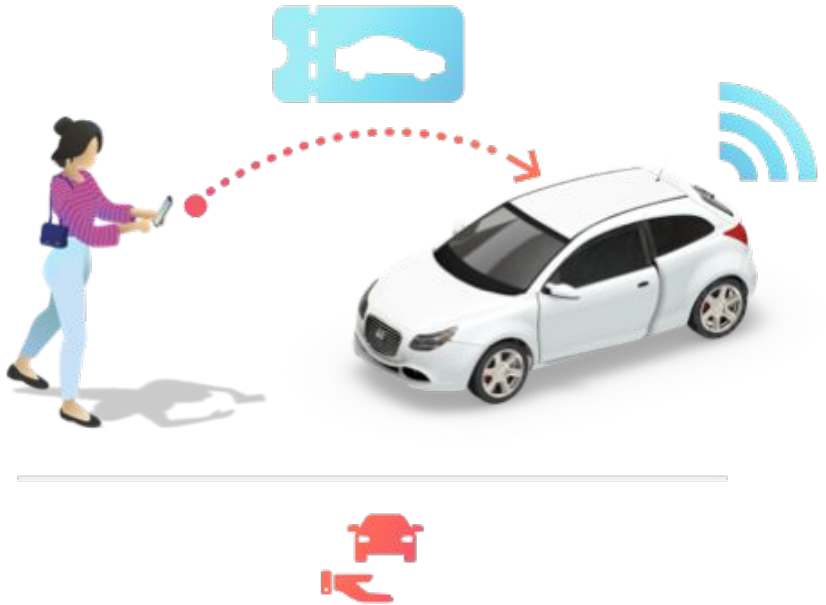
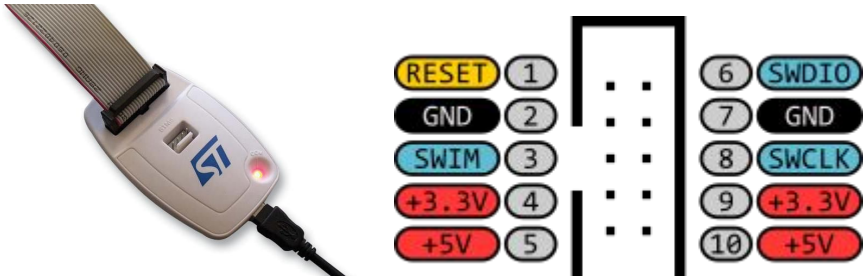


Illustration : [lien]

Analyse hardware : microcontrôleur STM32F7

- Puce principale : **STM32F7**
  - Architecture : Cortex M7 216 MHz
  - 2 Mo Flash
  - 512 Ko RAM
  - Connectivités CAN, LIN, USB
- Package **BGA 176**
- Accès de debug propriétaire **SWD**



Illustrations : [lien]

Analyse hardware : recherche des ports SWD

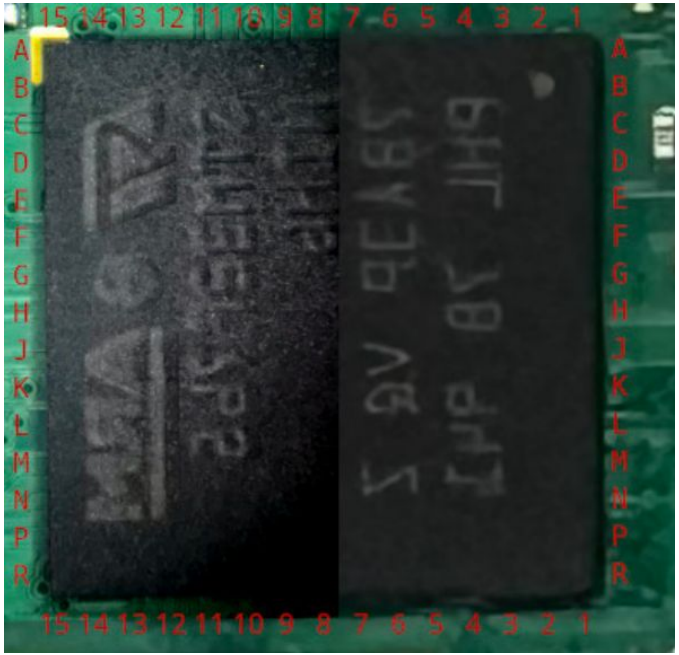
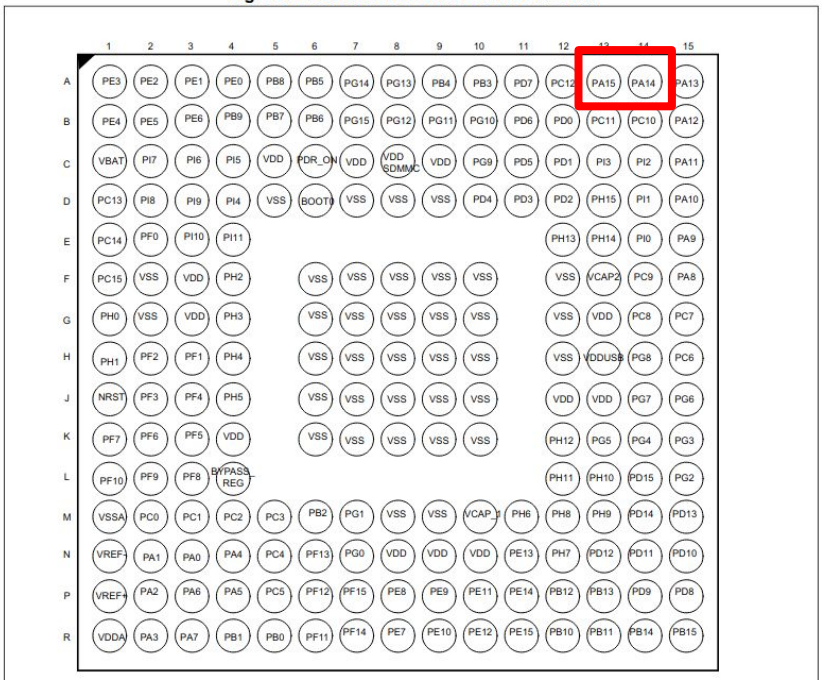


Figure 24. STM32F723xx UFBGA176 ballout



STM32F7 : protection RDP

Figure 8. Example of RDP protections (STM32L4 series)

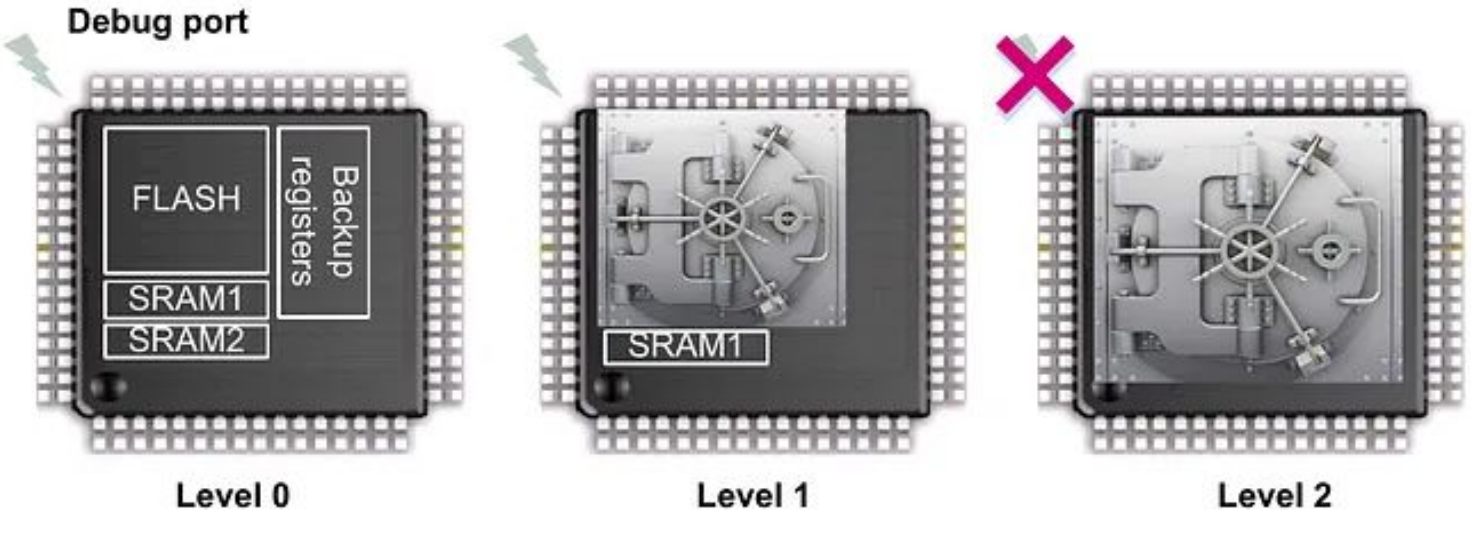
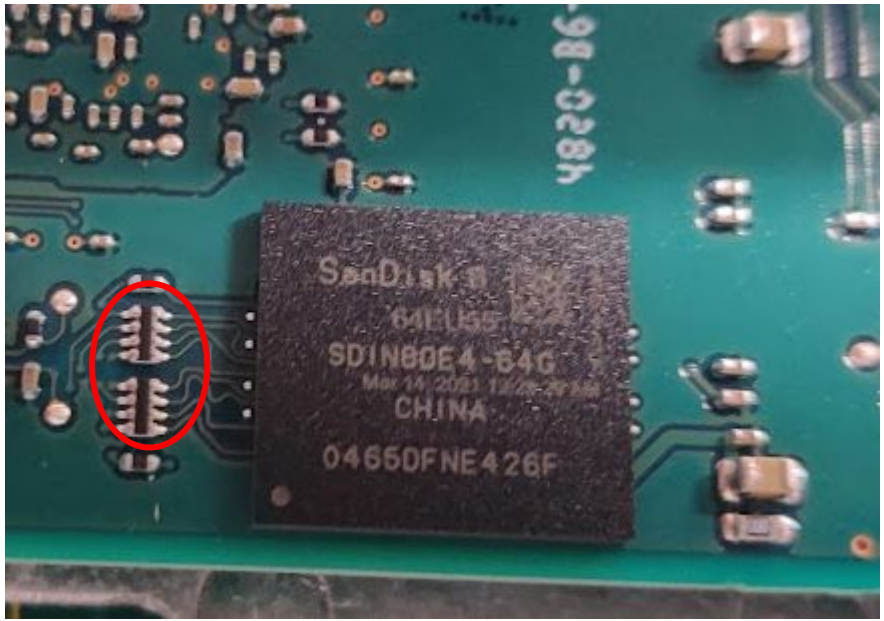
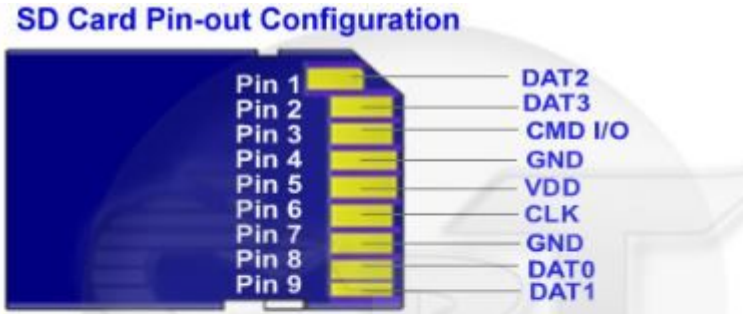


Illustration : [lien]

Analyse hardware, round 2 : mémoire eMMC

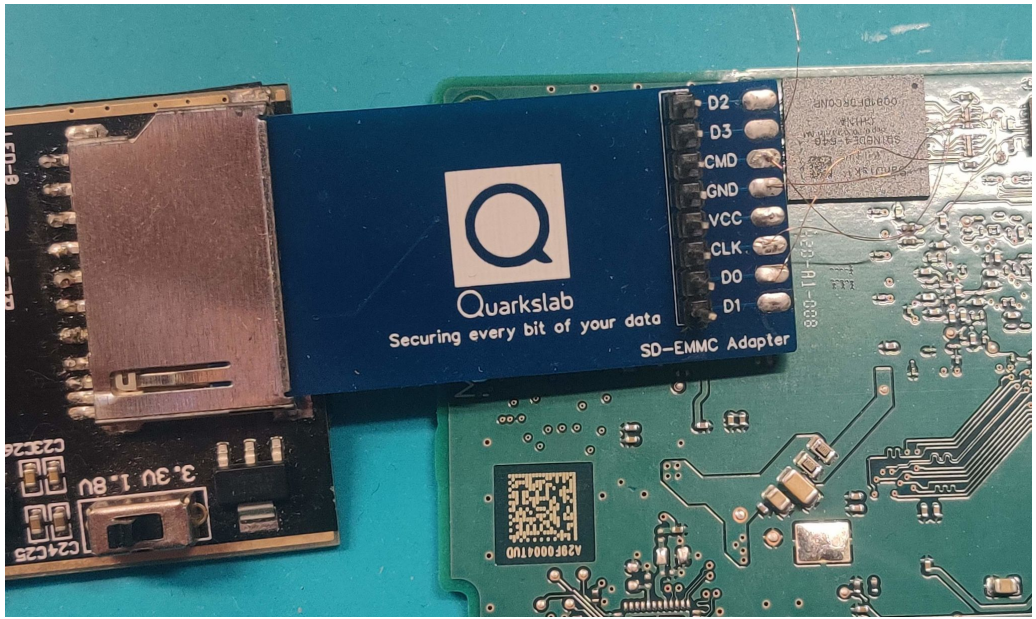




Pin	Signal	Description
1	Data2	Data signal 1
2	Data3	data signal 2
3	CMD I/O	input and output command
4	GND	supply voltage negative
5	VDD	supply voltage positive
6	CLK	clock signal
7	GND	supply voltage negative
8	Data0	data signal 0
9	Data1	data signal 1

Illustration : [lien]

Extraction de la mémoire eMMC



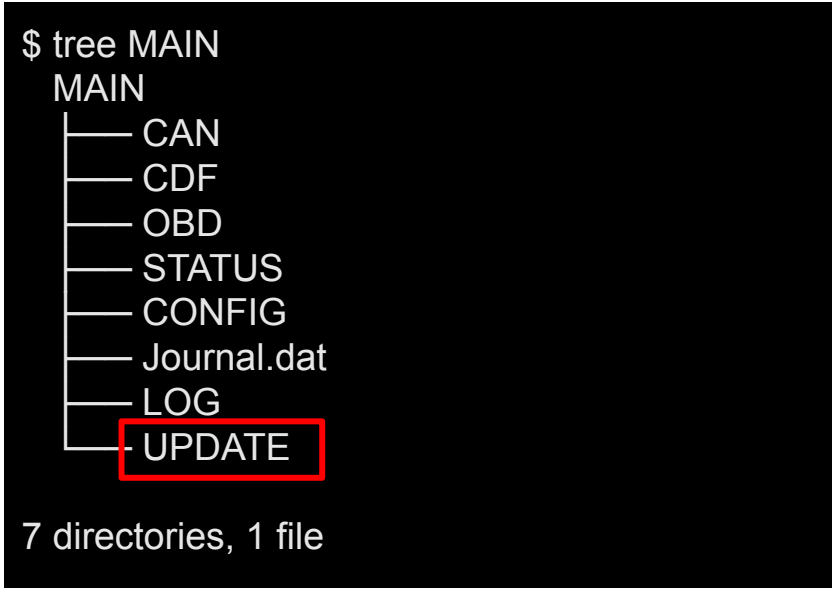
- Connexion d'un adaptateur SD sur les pins
  - VCC/GND
  - CMD
  - CLK
  - DAT0
- Arrêt du **STM32F7** via le debug **SWD**

```
Telnet localhost
Open On-Chip Debugger
> halt
target was in unknown state when halt was requested
target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x0800067a msp: 0x20001ff0
>
```

Illustration : [lien]

Analyse de la mémoire

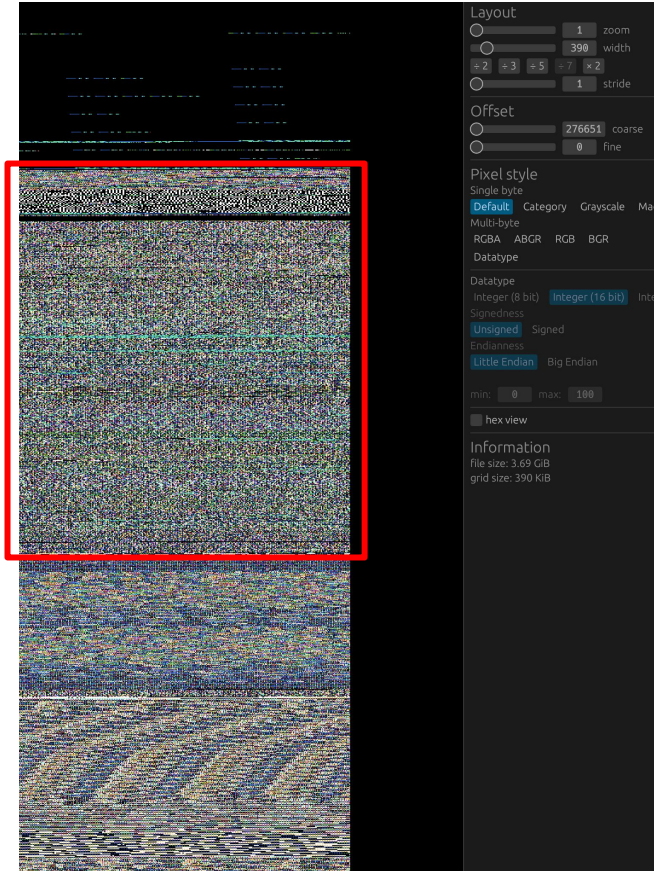
- 3 partitions FAT16
  - MAIN
  - EVENTS
  - LOGS
- Aucun autre résultat d'intérêt avec **unblob** ou **binwalk**
- La partition **MAIN** semble être utilisée pour les mises à jour, via le dossier **UPDATE**



Recherche d’instructions ARM

```
$ binwalk -A dump.bin

DECIMAL    HEXA             DESCRIPTION
--------------------------------------------------------------------------------
377182     0x5C15E          ARM instructions, function prologue
400790     0x61D96          ARM instructions, function prologue
1298808    0x13D178         ARM instructions, function prologue
1481976    0x169CF8         ARM instructions, function prologue
1575914    0x180BEA         ARM instructions, function prologue
1576006    0x180C46         ARM instructions, function prologue
1576282    0x180D5A         ARM instructions, function prologue
2202406    0x219B26         ARM instructions, function prologue
.....
```



Reverse : 1er essai

```
                        LAB_0007e03c                                    XREF[1]:     0007e02e(j)
0007e03c 04 a8                  add         r0,sp,#0x10
0007e03e a9 f0 06 fb            bl          WAVE_000ea800+249422
0007e042 04 46                  mov         r4,r0

                        LAB_0007e044                                    XREF[1]:     0007e03a(j)
0007e044 32 0d                  lsrs        r2,r6,#0x14
0007e046 03 d0                  beq         LAB_0007e050
0007e048 01 20                  movs        r0,#0x1
0007e04a 60 f3 1f 55            bfi         r5,r0,#0x14,#0xc
0007e04e 05 e0                  b           LAB_0007e05c

                        LAB_0007e050                                    XREF[1]:     0007e046(j)
0007e050 02 a8                  add         r0,sp,#0x8
0007e052 a9 f0 fc fa            bl          WAVE_000ea800+249422
0007e056 02 46                  mov         r2,r0
0007e058 dd e9 02 75            ldrd        r7,r5,[sp,#local_38]

                        LAB_0007e05c                                    XREF[1]:     0007e04e(j)
0007e05c a2 42                  cmp         r2,r4
```

FAT 101

- Un système de fichiers FAT se compose de :
  - **Volume Boot Record** (VBR)
  - 1+ **File Allocation Table** (FAT)
  - **Root Directory**
  - **Data Area**
- Concept de **sectors** et **clusters**
- Un fichier est stocké sur un ou plusieurs **clusters**

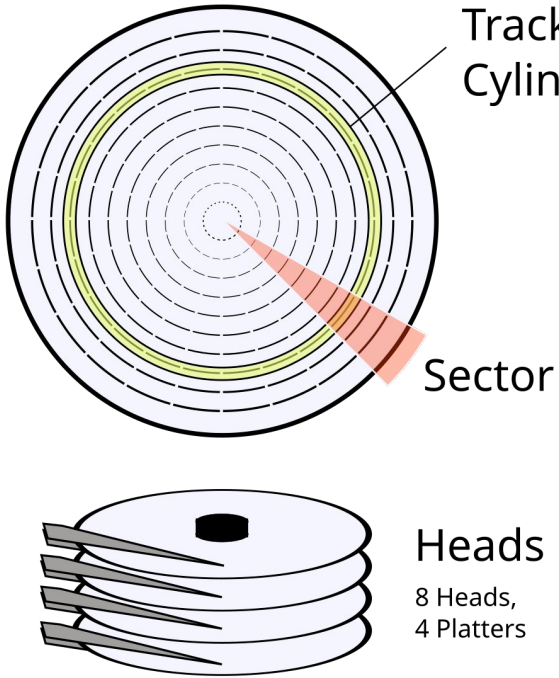


Illustration : [lien]

Volume Boot Record

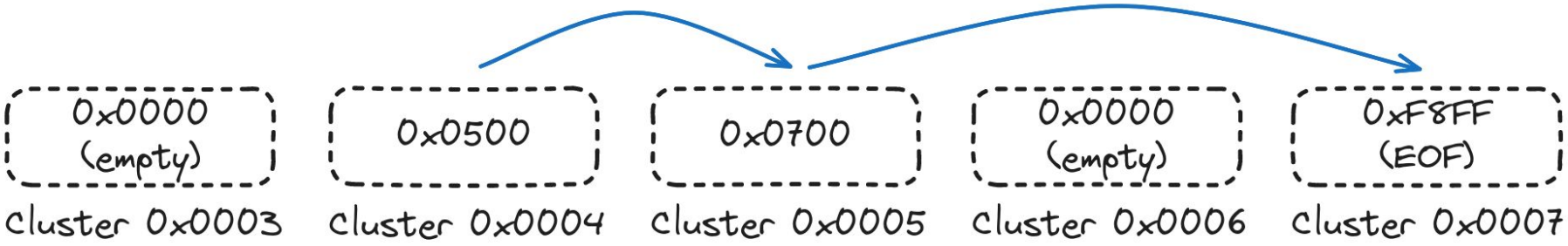
- 1er **sector** d'une partition FAT
- Spécifie notamment la taille d'un **sector** et d'un **cluster**
  - Sector : 512 octets (0x0200)
  - Sector per cluster : 8
- Magic `**0x55 0xAA**` en fin de VBR

```
0001:F800 E9 00 00 4D 53 57 49 4E 34 2E 31 00 02 08 01 00  é..MSWIN4.1.....
0001:F810 02 00 02 00 00 F8 9E 00 3F 00 FF 00 00 00 00 00  .....ø..?.ÿ.....
0001:F820 DD E9 04 00 80 00 29 67 45 23 01 4E 4F 20 4E 41  Ýé....)gE#.NO NA
0001:F830 4D 45 20 20 20 20 46 41 54 31 36 20 20 20 00 00  ME    FAT16   ..
0001:F840 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001:F850 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001:F860 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001:F870 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001:F880 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001:F890 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001:F8A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001:F8B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001:F8C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001:F8D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001:F8E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001:F8F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001:F900 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001:F910 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001:F920 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001:F930 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001:F940 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001:F950 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001:F960 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001:F970 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001:F980 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001:F990 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001:F9A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001:F9B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001:F9C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001:F9D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001:F9E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0001:F9F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA  ..............Uª
```

File Allocation Table

- Permet de lier les différents clusters utilisés par un fichier, 2 octets / cluster en FAT16

- Valeurs spéciales
  - 0x0000 : cluster non utilisé
  - 0xFFF7 : erreur
  - 0xFFF8 - 0xFFFF : dernier cluster du fichier



Root Directory

- Liste les **dossiers** et **fichiers** stockés à la racine
- Format historique **Short File Name** (**SFN**) sur 32 octets, nom de 8 caractères ASCII et extension de 3 caractères
- Format **Long File Name** (**LFN**) pour accepter plus de 8 caractères et les jeux de caractères étendus
- Un fichier devant être effacé voit son premier caractère remplacé par **0xE5**

Root Directory SFN Entry Data Structure

Bytes	Purpose
0	First character of file name (ASCII) *or* allocation status (0x00=unallocated, 0xe5=deleted)
1-10	Characters 2-11 of the file name (ASCII); the "." is implied between bytes 7 and 8
11	File attributes (see File Attributes table)
12	Reserved
13	File creation time (in tenths of seconds)*
14-15	Creation time (hours, minutes, seconds)*
16-17	Creation date*
18-19	Access date*
20-21	High-order 2 bytes of address of first cluster (0 for FAT12/16)*
22-23	Modified time (hours, minutes, seconds)
24-25	Modified date
26-27	Low-order 2 bytes of address of first cluster
28-31	File size (0 for directories)

File Attributes

Flag Value	Description
0000 0001 (0x01)	Read-only
0000 0010 (0x02)	Hidden file
0000 0100 (0x04)	System file
0000 1000 (0x08)	Volume label
0000 1111 (0x0f)	Long file name
0001 0000 (0x10)	Directory
0010 0000 (0x20)	Archive

** Bytes 13-22 are unused by DOS*

Illustration : [lien]

```
00000000 | 54 45 53 54  46 49 4c 45  42 49 4E 20  00 00 10 75 | TESTFILEBIN ...u
00000010 | 56 57 00 00  00 00 10 75  56 57 0F 00  24 00 00 00 | fW.....ufW..$..
```

Analyse forensic - recherche des fichiers effacés

- Analyse des entrées de la **File Allocation Table**
- Recherche de l'octet **0xE5** dans les **SFN**

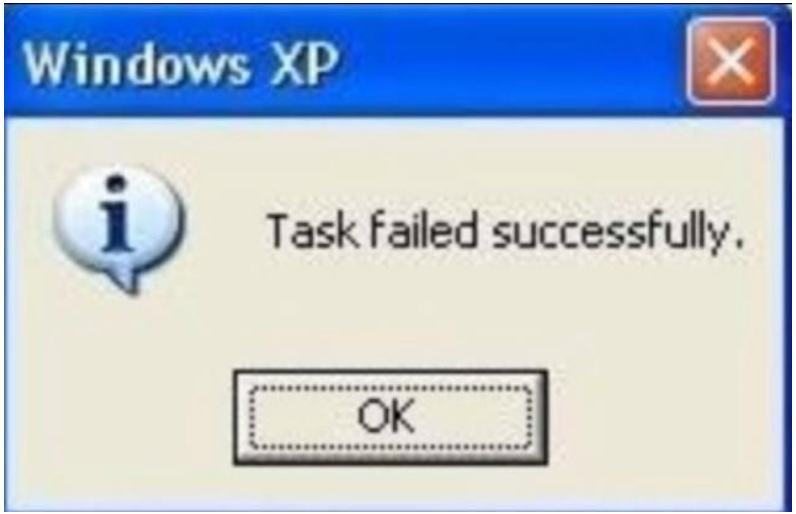


Illustration : [lien]

Réorientation de notre recherche

- Tout ou partie du firmware semble présent dans différents **clusters**
- **Objectif #1** : comment identifier des clusters contenant des parties du firmware
- **Objectif #2** : comment ordonner les fragments

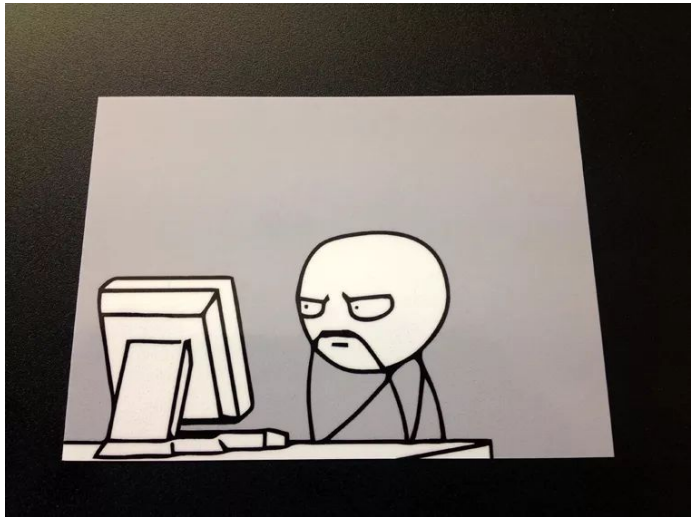
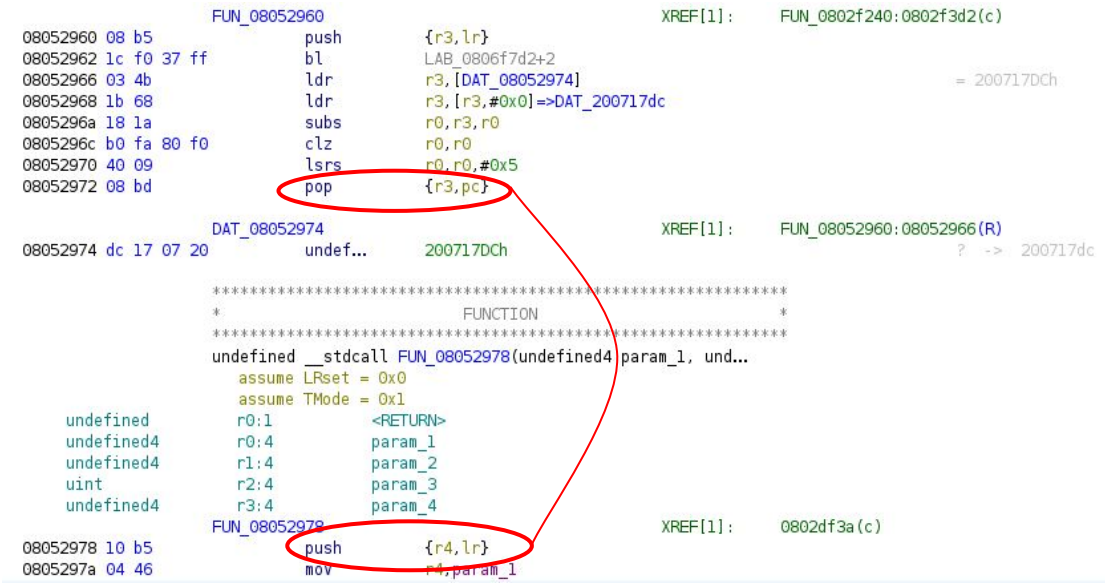


Illustration : [lien]

‘Prologue’ d’une récupération

- Des **valeurs aléatoires** peuvent produire des **opcodes ARM valides** , comment déterminer que le contenu d’un cluster est bien une partie de notre firmware ?

- **Solution :** rechercher une succession **d’épilogue** et de **prologue** typique des fonctions
  - pop - push
  - pop - branch
  - pop - [ldr/sub/add] r3



Automatisation de la recherche avec Capstone


```
#!/usr/bin/python3
from capstone import *

def cluster_contains_valid_pattern(cluster_data):
  base_addr = 0x00200000
  md = Cs(CS_ARCH_ARM, CS_MODE_THUMB)
  found_pattern = False
  got_pop = False

  for instr in md.disasm(cluster_data,base_addr):
    if "pop" in instr.mnemonic:
      got_pop = True
      prev_mnemonic = instr.mnemonic + " " + instr.op_str
    elif ("push" in instr.mnemonic or "b." in instr.mnemonic or "r3" in instr.op_str) and got_pop == True:
      got_pop = False
      found_pattern = f"{instr.address:#08x} {prev_mnemonic}; {instr.mnemonic} {instr.op_str}"
      break
    else:
      got_pop = False
  return found_pattern
```

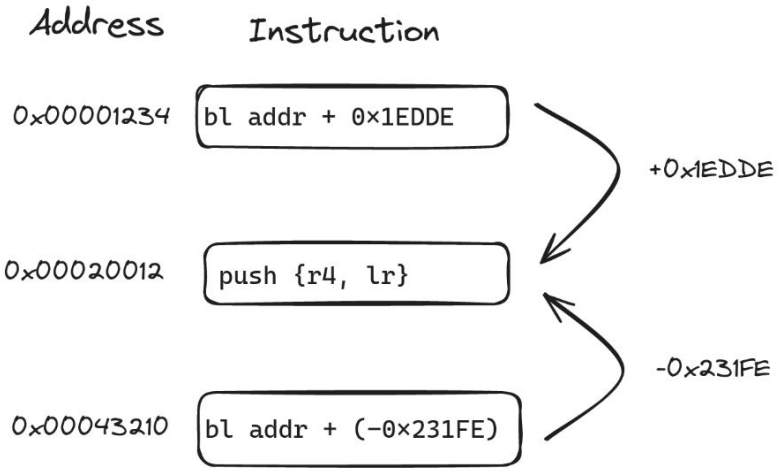
Résultat de la recherche

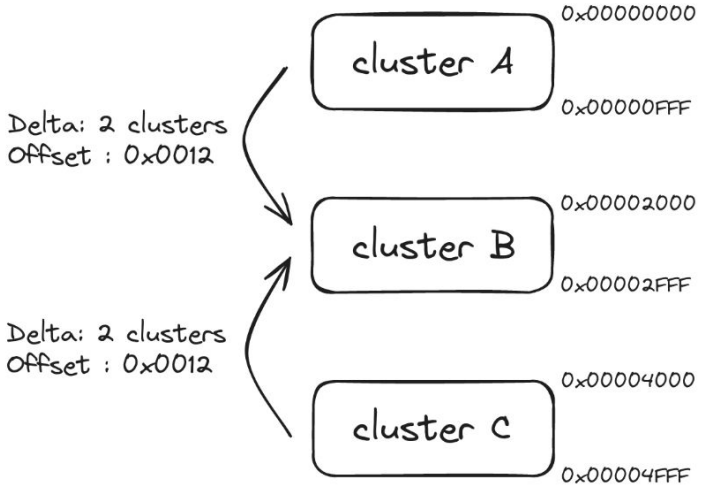
```
$ ./fat_explorer.py --decompile-all
MSWIN4.1 - sector: 512 bytes - cluster: 8 sectors
  Start address: 0x0000f800
  FAT0: 0x0000fa00 [158 sectors]
  FAT1: 0x00023600 [158 sectors]
  Root: 0x00037200 [16384 bytes]
  Data: 0x0003b200

Decompiling clusters:  100%|.....................................| 960/960 [00:40<00:00, 22.98it/s]
Cluster #20:
  0x2008c8 pop {r4, r5, r6, pc}; ldrb.w r3, [sp, #0x19]
Cluster #21:
  0x2005fe pop.w {r4, r5, r6, r7, lr}; b.w #0x26d420
Cluster #22:
  0x20056c pop.w {r4, r5, r6, lr}; b.w #0x200086
Cluster #24:
  0x2001e8 pop {r4, r5, r6, r7}; b.w #0x1ffda4
Cluster #26:
  0x200702 pop.w {r3, r4, r5, lr}; b.w #0x267920
…
```

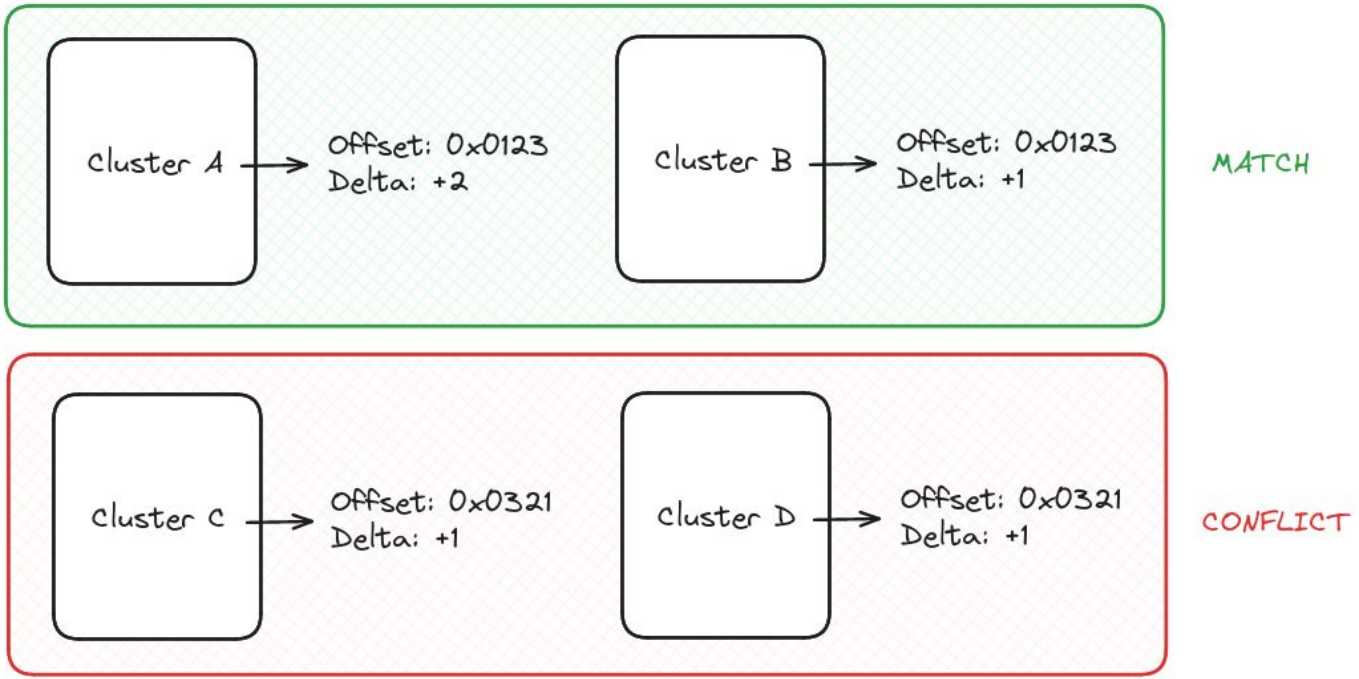
Recherche de liens entre deux clusters

- Exploitation des instructions **“branch link”** pour identifier des sauts vers une même **fonction** entre différents **clusters**



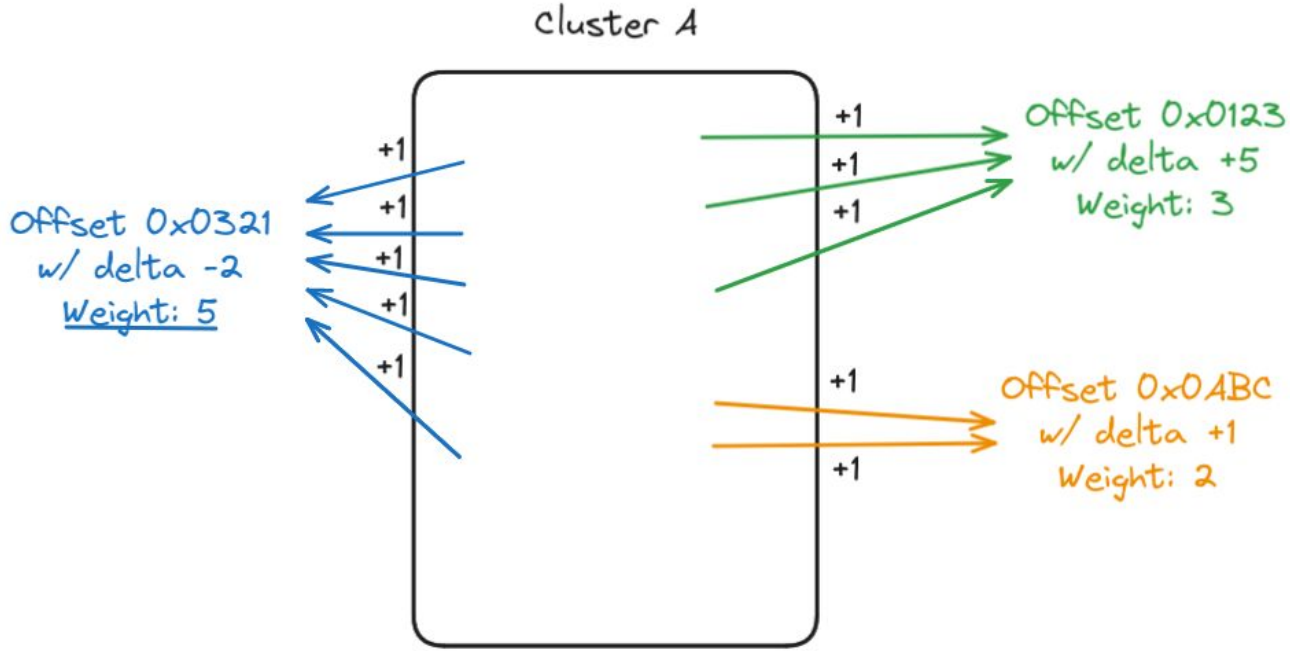


Etablissement d’une liste de relations



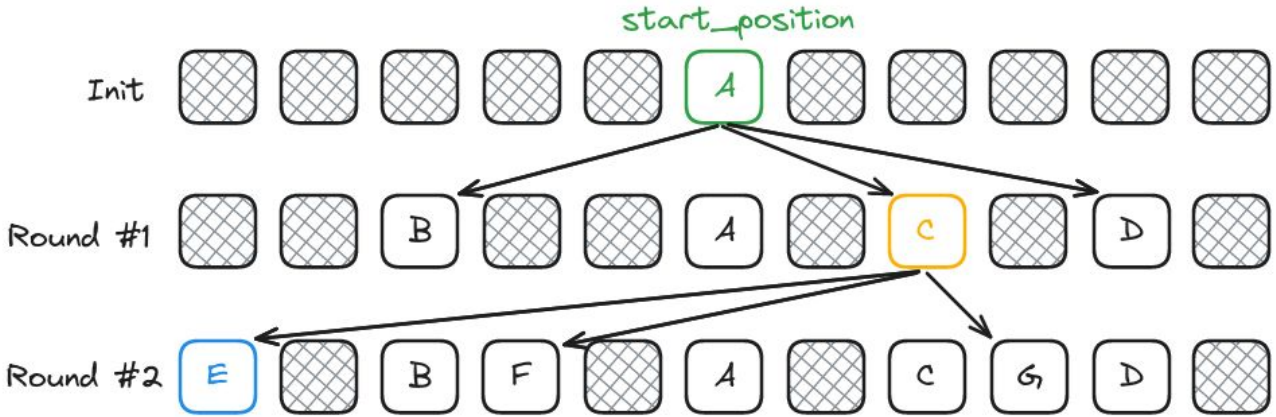
Pondération des résultats

- Plus un **cluster** a d'appels à une même fonction, plus son **poids** sera élevé



Repositionnement des clusters

- **1 -** Position de départ du **cluster** avec le poids le plus élevé au ⅔ du nombre de clusters non-nuls
- **2 -** Positionnement des **clusters** partageant les mêmes appels via la **liste de relations**
- **3 -** Identification du **cluster** ayant le poids le plus élevé parmis les **derniers positionnés** pour répéter l'étape 2



Relation list

Cluster	Higher Weight ↓
A	34
E	26
C	24
...	...

Corrections et vérifications manuelles

```
$ ./fat_explorer.py --decompile-all --compute-branch
MSWIN4.1 - sector: 512 bytes - cluster: 8 sectors
Start address: 0x0000f800
FAT0: 0x0000fa00 [158 sectors]
FAT1: 0x00023600 [158 sectors]
Root: 0x00037200 [16384 bytes]
Data: 0x0003b200

Decompiling clusters: 100%|.....| 960/960 [00:46<00:00, 20.07it/s]

positioning clusters from branches:
>  20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43,
44, 45, 355, 47, 48, 49, 50, 51, 710, 53, 54, 55, 306, 307, 308, 309, 310, 311, 312, 313, 314,
315, 316, -1, 318, -1, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333,
492, 335, 336, 337, 338, 339, 340, 341, 493, 343, 344, 345, 346, 347, 348, 349, 350, 351,
352, 353, 354, -1, 62, 357, 358, -1, -1, -1, -1, 60, 179, -1, 366, 367, 368, 369, -1, 371, 358,
373, 374, 361, 362, 377, 378, 365, 208, 142, 393, 397, 401, 71, 407, 411, 412, 413, 414,
415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, -1, 382, -1, 375, 433,
434, -1, 436, 437, 438, 439, 440, 441 [...]
```

Reverse : 2ème essai

```
)260 2d e9 f0 4f          push        {r4,r5,r6,r7,r8,r9,r10,r11,lr}
)264 b2 f5 c0 7f          cmp.w       param_3,#0x180
)268 ad f5 49 7d          sub.w       sp,sp,#0x324
)26c 01 90                str         param_1,[sp,#local_344]
)26e 00 f2 d6 80          bhi.w       LAB_080904le
)272 16 46                mov         r6,param_3
)274 0c 46                mov         r4,param_2
)276 4f f4 d0 72          mov.w       param_3,#0x1a0
)27a 00 21                movs        param_2,#0x0
)27c 60 a8                add         param_1,sp,#0x180
)27e 66 ad                add         r5,sp,#0x198
)280 20 f0 45 fd          bl          memset                    void * memset(void * __s, in
)284 1a a8                add         param_1,sp,#0x68
)286 fb f7 59 f9          bl          mbedtls_aes_init          undefined mbedtls_aes_init(v
)28a 21 46                mov         param_2,r4
)28c 33 0a                lsrs        r3,r6,#0x8
)28e 00 24                movs        r4,#0x0
)290 32 46                mov         param_3,r6
)292 28 46                mov         param_1,r5
)294 8d f8 92 31          strb.w      r3,[sp,#local_1b6]
)298 30 23                movs        r3,#0x30
)29a 8d f8 93 61          strb.w      r6,[sp,#local_1b5]
)29e 8d f8 97 31          strb.w      r3,[sp,#local_1b1]
)2a2 8d f8 90 41          strb.w      r4,[sp,#local_1b8]
)2a6 8d f8 91 41          strb.w      r4,[sp,#local_1b7]
)2aa 2d f0 96 f8          bl          memcpy                    void * memcpy(void * __dest,
)2ae 80 22                movs        param_3,#0x80
)2b0 23 46                mov         r3,r4
```

Illustration : [lien]

Merci !

Quarkslab

Thank you

Contact information:

Email: contact@quarkslab.com

Phone: +33 1 58 30 81 51

Website: www.quarkslab.com

@quarkslab

Quarkslab