TARLOGIC
CYBERSECURITY EXPERTS

Who watches the watchmen? Reversing video surveillance equipment…

Black Alps 2025

TARLOGIC
CYBERSECURITY EXPERTS

ABOUT

- Antonio Vázquez Blanco
- Research and Innovation Team at Tarlogic
- antonio.vazquez@tarlogic.com
- @antoniovazquezblanco@mastodon.social
- @antonvblanco

TARLOGIC
CYBERSECURITY EXPERTS

CONTEXT

- Internal exercise
- Black box
- Focus on hardware security analysis
- Starting point is a physical device
- Main objective is to obtain the latest firmware

TARLOGIC
CYBERSECURITY EXPERTS

DEVICES

DAHUA DHI-NVR2104-4KS2

DAHUA DH-XVR4104HS-I

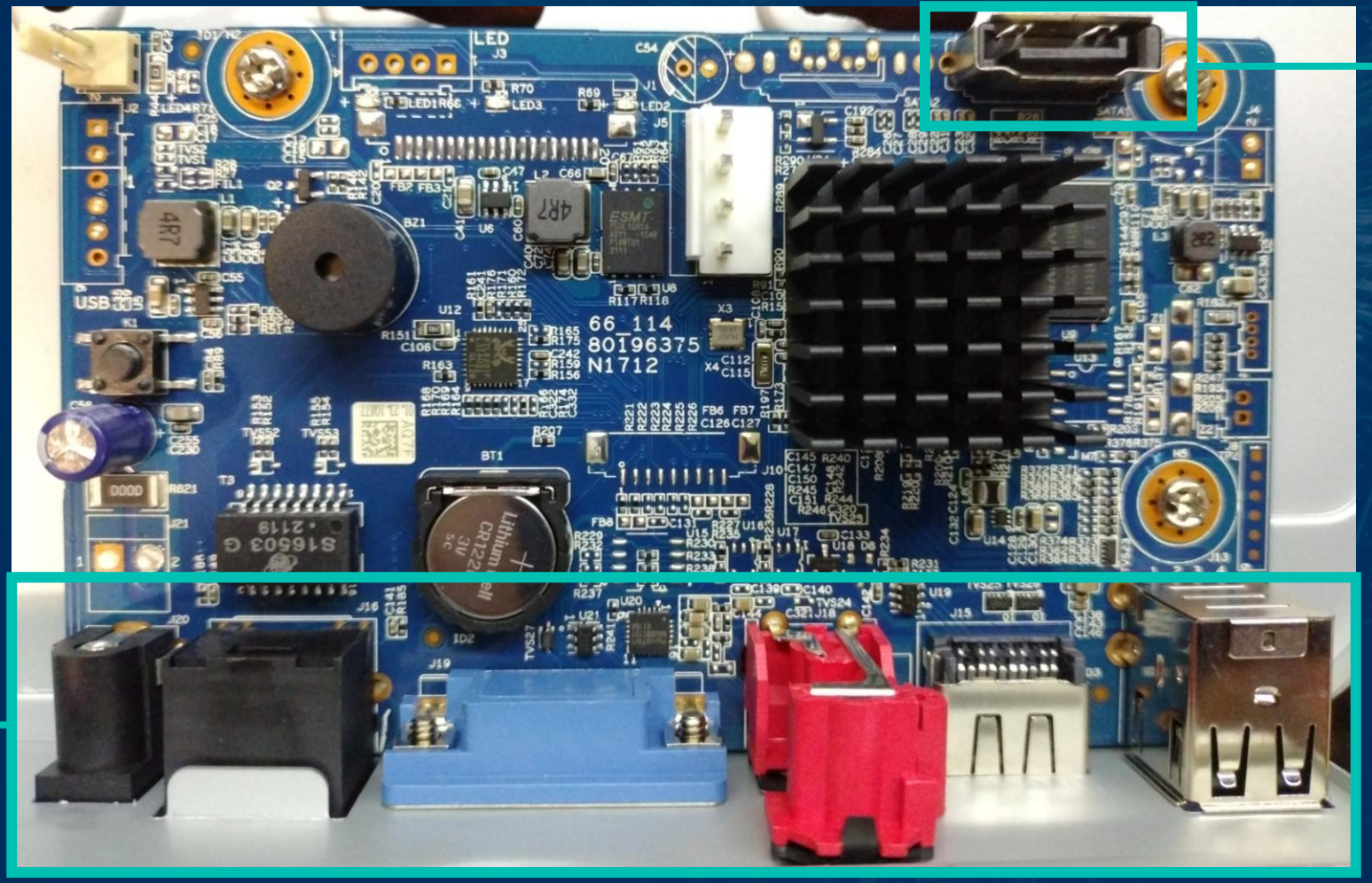
- Video recording units
- Similar specifications
- Similar hardware & firmware

TARLOGIC
CYBERSECURITY EXPERTS

HARDWARE

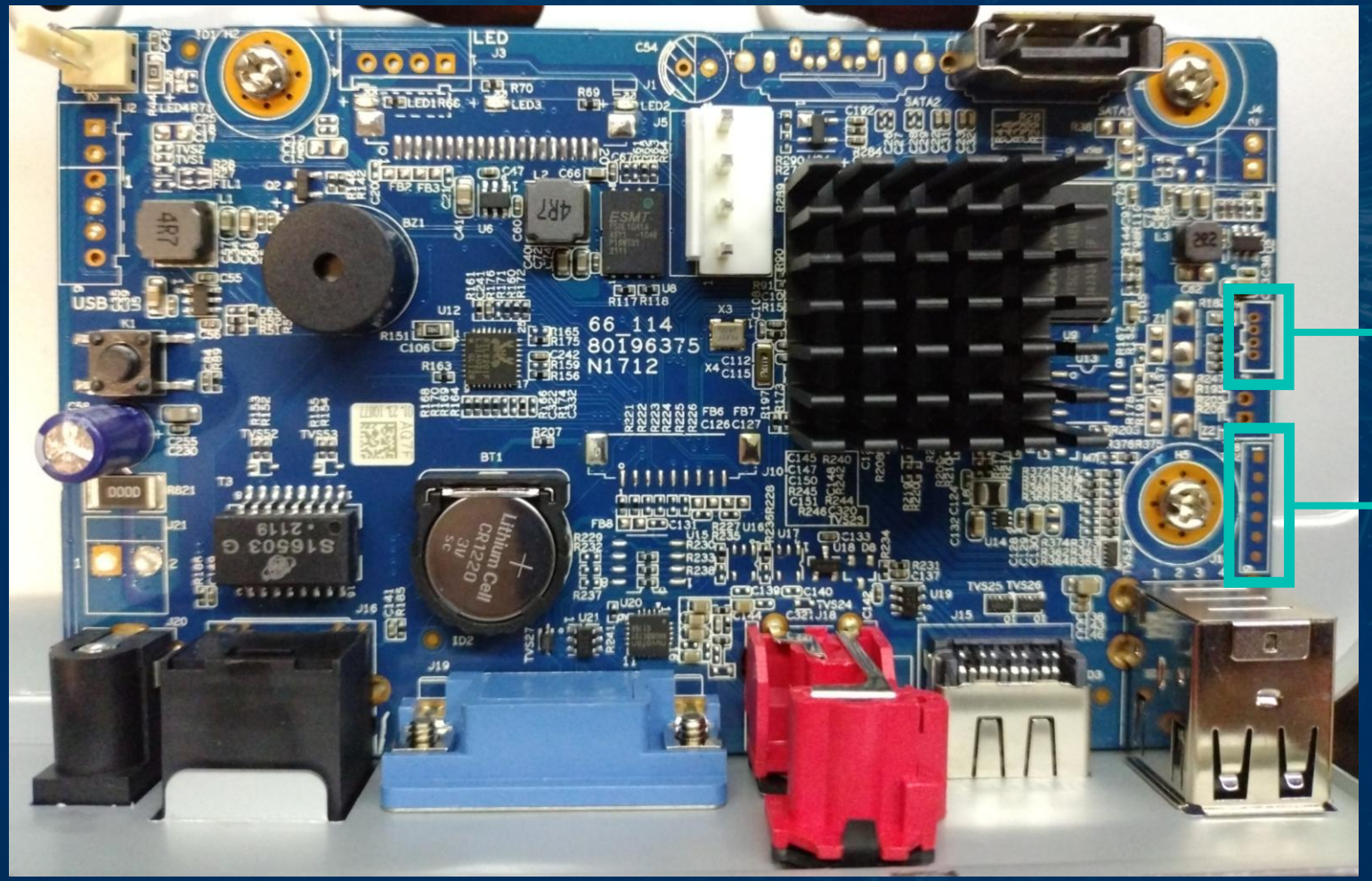
TARLOGIC
CYBERSECURITY EXPERTS

HARDWARE



TARLOGIC
CYBERSECURITY EXPERTS

HARDWARE



TARLOGIC
CYBERSECURITY EXPERTS

UART

```
System startup
U-Boot 2010.06-svn4868 (Sep 24 2020 - 09:56:33)
Check Flash Memory ContCheck Flash ... Found
ECC provided by Flash Memory Controller
Hit any key to stop autoboot: 3 ... 2 ... 1 ... 0
…
Image Name:   Linux-3.18.20
Image Type:   ARM Linux Kernel Image (uncompressed)
Data Size:    3768619 Bytes = 3.6 MiB
Load Address: 80008000
Entry Point:  80008000
Kernel data secure check, please wating ...
sec_commonSwRsaVerify run successfully!
Loading Kernel Image ... OK
Starting kernel ...
```

TARLOGIC
CYBERSECURITY EXPERTS

UART

```
System startup
U-Boot 2010.06-svn4868 (Sep 24 2020 - 09:56:33)
Check Flash Memory ContCheck Flash ... Found
ECC provided by Flash Memory Controller
Hit any key to stop autoboot: 3 ... 2 ... 1 ... 0
…
Image Name:   Linux-3.18.20
…
```

TARLOGIC
CYBERSECURITY EXPERTS

UART

```python
import serial
from pexpect.exceptions import TIMEOUT
from pexpect_serialspawn import SerialSpawn

def test_char(ss, character) -> bool:
    print("Please, press the reboot button on the device now.")
    # Await up to 1 minute for the device to reboot
    result = ss.expect([TIMEOUT, 'ECC provided by Flash Memory Controller'], timeout=60)
    if result == 0:
        print("Timeout waiting for device to reboot.")
        exit(1)
    # We are in the message just before the bootloader autoboot stop message
    # Send the character many times to ensure it is received
    msg = f"{character}\r\n" * 100
    ss.send(msg)
    # Check if we interrupted the boot process
    result = ss.expect([TIMEOUT, 'Image Name:   Linux-3.18.20'], timeout=5)
    if result == 0:
        print(f"Character '{character}' INTERRUPTED the boot process.")
        return True
    else:
        print(f"Character '{character}' did NOT interrupt the boot process.")
        return False
```

TARLOGIC
CYBERSECURITY EXPERTS

UART

TARLOGIC
CYBERSECURITY EXPERTS

UART

```
hisilicon # help
?       - alias for 'help'
autoup  - load update file from server
boot    - boot kernel from uboot
bootm   - boot application image from memory
decjpg  - jpgd - decode jpeg picture.
devid   - devid - set hardware id and save to flashfatload - load binary file from a
dos filesystem
fatls   - list files in a directory (default /)
fb_set  - fb_set - get shift key
fb_test - fb_test - frontboard read/write test
get_key - get_key - get shift key
help    - print command description/usage
kaimendaji - kai men da ji
lock_otp - lock_otp - otp lock
…
```

TARLOGIC
CYBERSECURITY EXPERTS

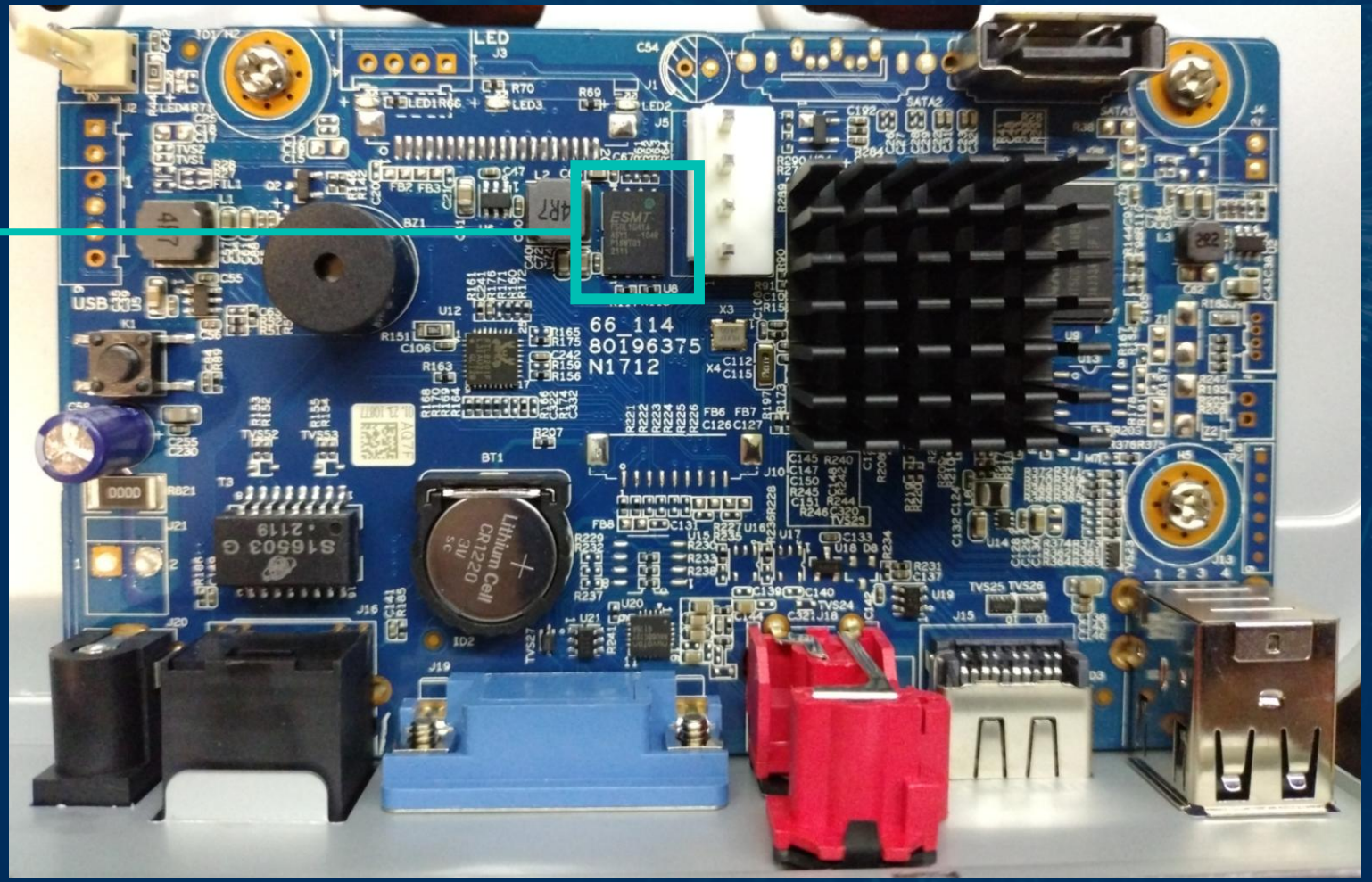
UART

```
hisilicon # devid
DEVID: DHI-NVR2104-4KS2
hisilicon # nand
nand - NAND sub-system
hisilicon # nand read
not support this cmd
hisilicon # nandops read
hisilicon # nandops read 0x0
hisilicon # xhprint
hisilicon # xhprint ID
hisilicon # xhprintenv
hisilicon # xhprintenv ID
```

TARLOGIC
CYBERSECURITY EXPERTS

HARDWARE

NAND FLASH (F50L1G41A)



TARLOGIC
CYBERSECURITY EXPERTS

HARDWARE

RT809H

TARLOGIC
CYBERSECURITY EXPERTS

FLASH

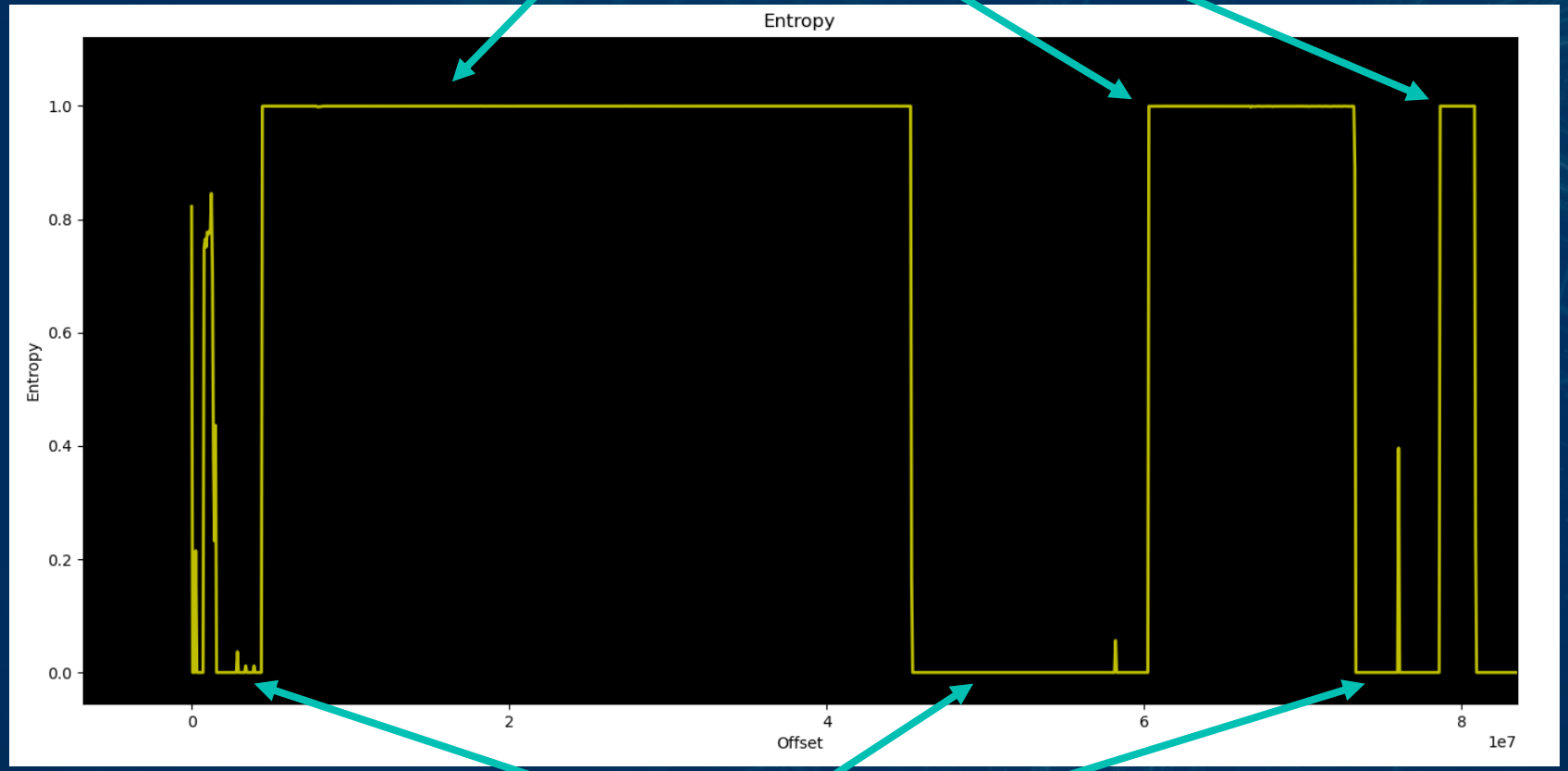
```
% binwalk dahua_nvr.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
262144        0x40000         Flattened device tree, size: 15970 bytes, version: 17
1130639       0x11408F        ESP Image segment count: 1, flash mode: QUIO, flash size: 2MB, entry
address: 0x4dd038e3, hash: none
…
1288009       0x13A749        eCos RTOS string reference: "ecos %s"
1296358       0x13C7E6        MP3 ID3 tag,
1296508       0x13C87C        SHA256 hash constants, little endian
…
1319959       0x142417        AES Inverse S-Box
…
4456448       0x440000        Squashfs filesystem, little endian, version 4.0, compression:gzip,
size: 40899891 bytes, 9 inodes, blocksize: 131072 bytes, created: 2021-03-12 06:37:13
58196508      0x378021C       Zlib compressed data, best compression
…
```

TARLOGIC
CYBERSECURITY EXPERTS

FLASH

PARTITIONS

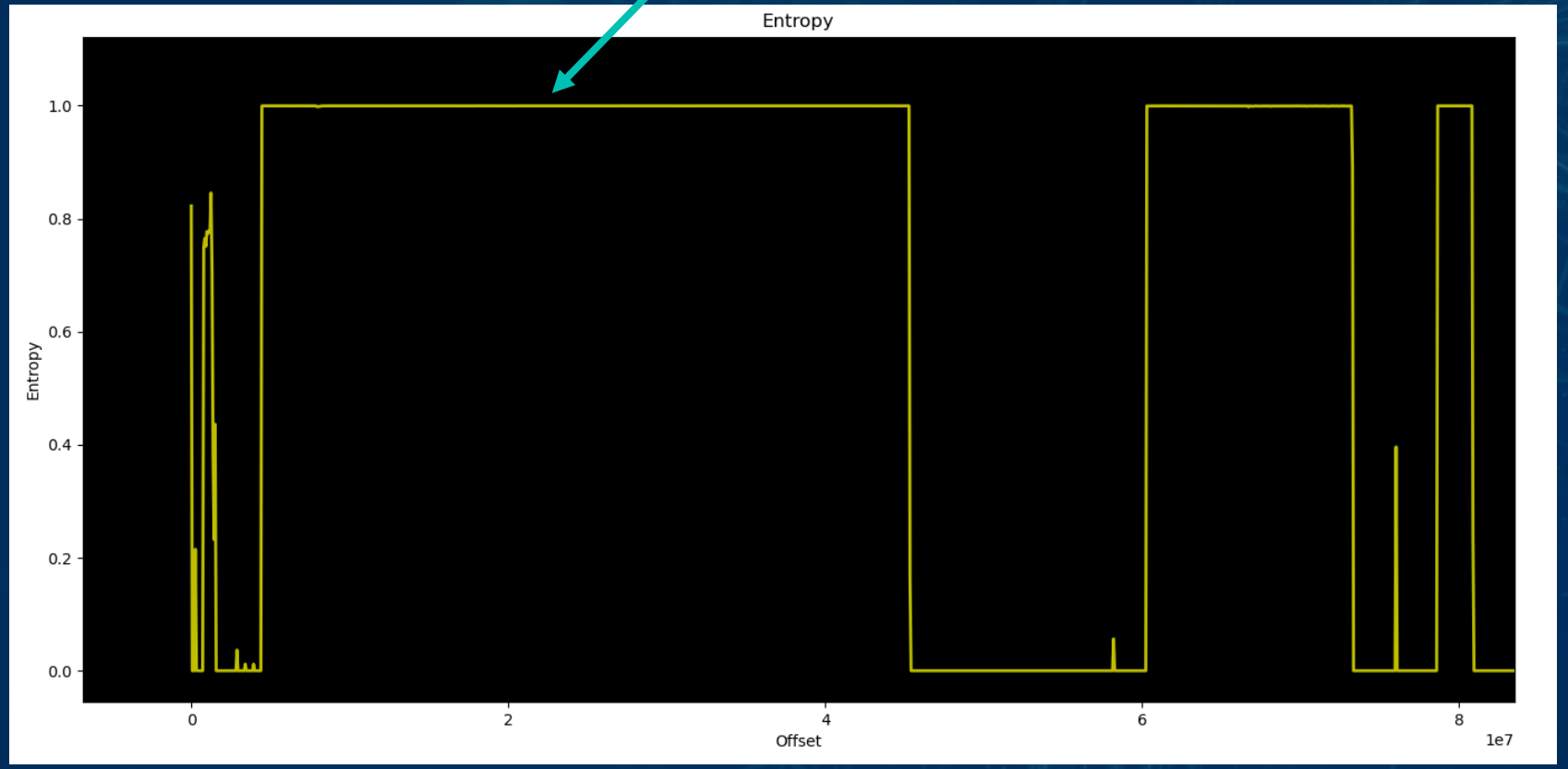


FREE SPACE

TARLOGIC
CYBERSECURITY EXPERTS

FLASH

USERSPACE?



TARLOGIC
CYBERSECURITY EXPERTS

FLASH

```
% dd if=dahua_nvr.bin of=dahua_nvr_userspace.bin bs=$((0x10000)) skip=$((0x44))
count=$((0x271))
40960000 bytes (41 MB, 39 MiB) copied, 0,614597 s, 66,6 MB/s

% binwalk dahua_nvr_userspace.bin
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             Squashfs filesystem, little endian, version 4.0,
compression:gzip, size: 40899891 bytes, 9 inodes, blocksize: 131072 bytes, created: 2021-
03-12 06:37:13

% unsquashfs dahua_xvr_userspace.bin
Parallel unsquashfs: Using 2 processors
3 inodes (314 blocks) to write

[=============================================================/] 317/317 100%
```

TARLOGIC
CYBERSECURITY EXPERTS

FLASH

```
% tree
.
├── boot
│   └── uImage
├── dev
├── romfs-x.squashfs
├── root
└── usr
    └── data
        └── hardware.lua
```

TARLOGIC
CYBERSECURITY EXPERTS

FLASH

```
% binwalk boot/uImage
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
26308         0x66C4          xz compressed data

% binwalk romfs-x.squashfs
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
…
12181334      0xB9DF56        MP3 ID3 tag,
…

% binwalk usr/data/hardware.lua

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
```

TARLOGIC
CYBERSECURITY EXPERTS

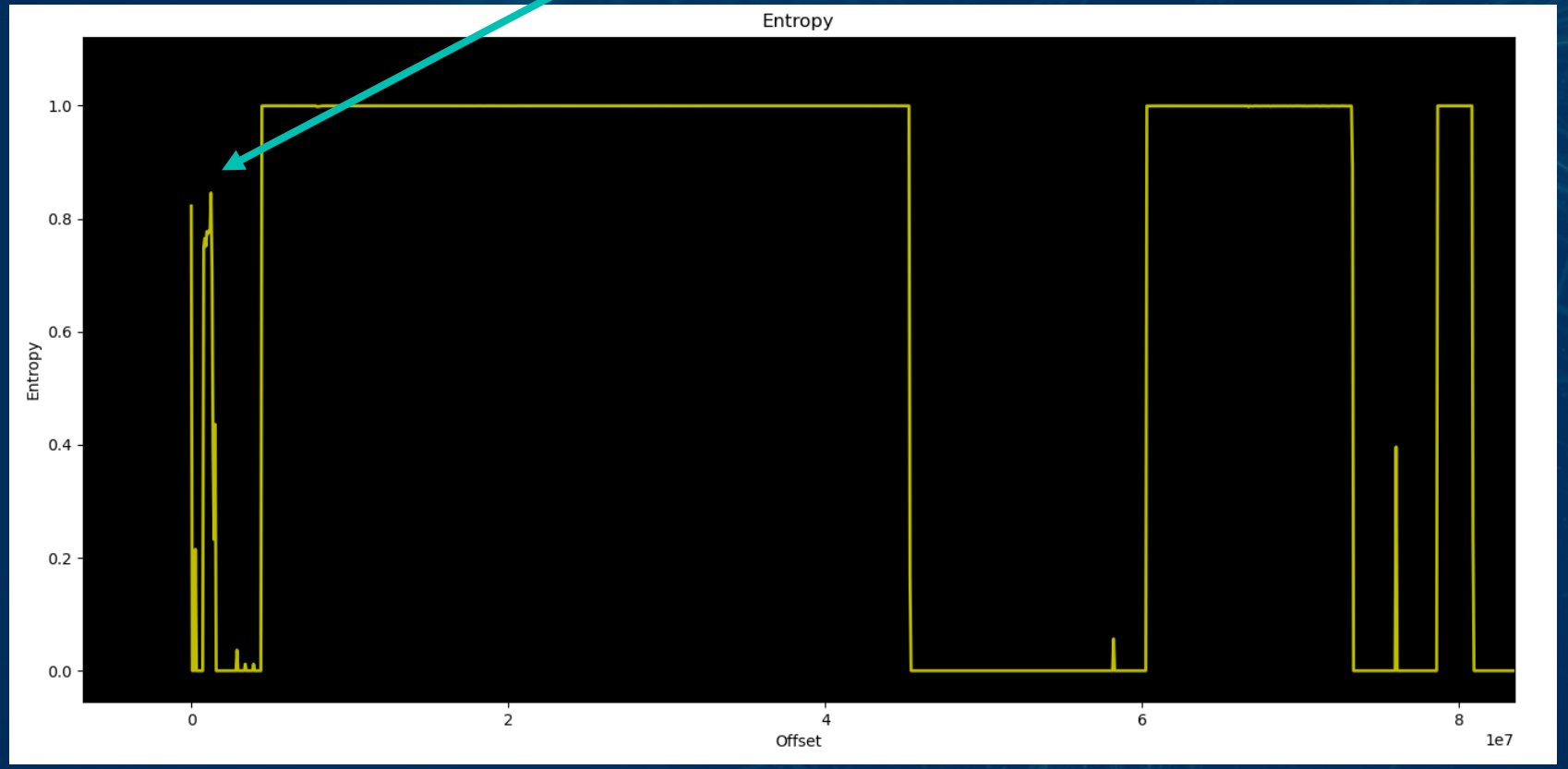
FLASH

- Based on the names, that were the kernel and userspace
- Binwalk only returns false positives
- High entropy (0.8) on all files
- Most likely, those are encrypted
- But something must decrypt to load them…

TARLOGIC
CYBERSECURITY EXPERTS

FLASH

BOOTLOADER?



TARLOGIC
CYBERSECURITY EXPERTS

FLASH

```
% dd if=dahua_nvr.bin of=dahua_nvr_bootloader.bin bs=$((0x1000)) skip=$((0x0))
count=$((0xF))
15+0 records in
15+0 records out
61440 bytes (61 kB, 60 KiB) copied, 0,00398486 s, 15,4 MB/s

% binwalk -Y dahua_nvr_bootloader.bin
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
4396          0x112C          ARM executable code, 16-bit (Thumb), little endian, at
least 1002 valid instructions
```

- Not encrypted!!!

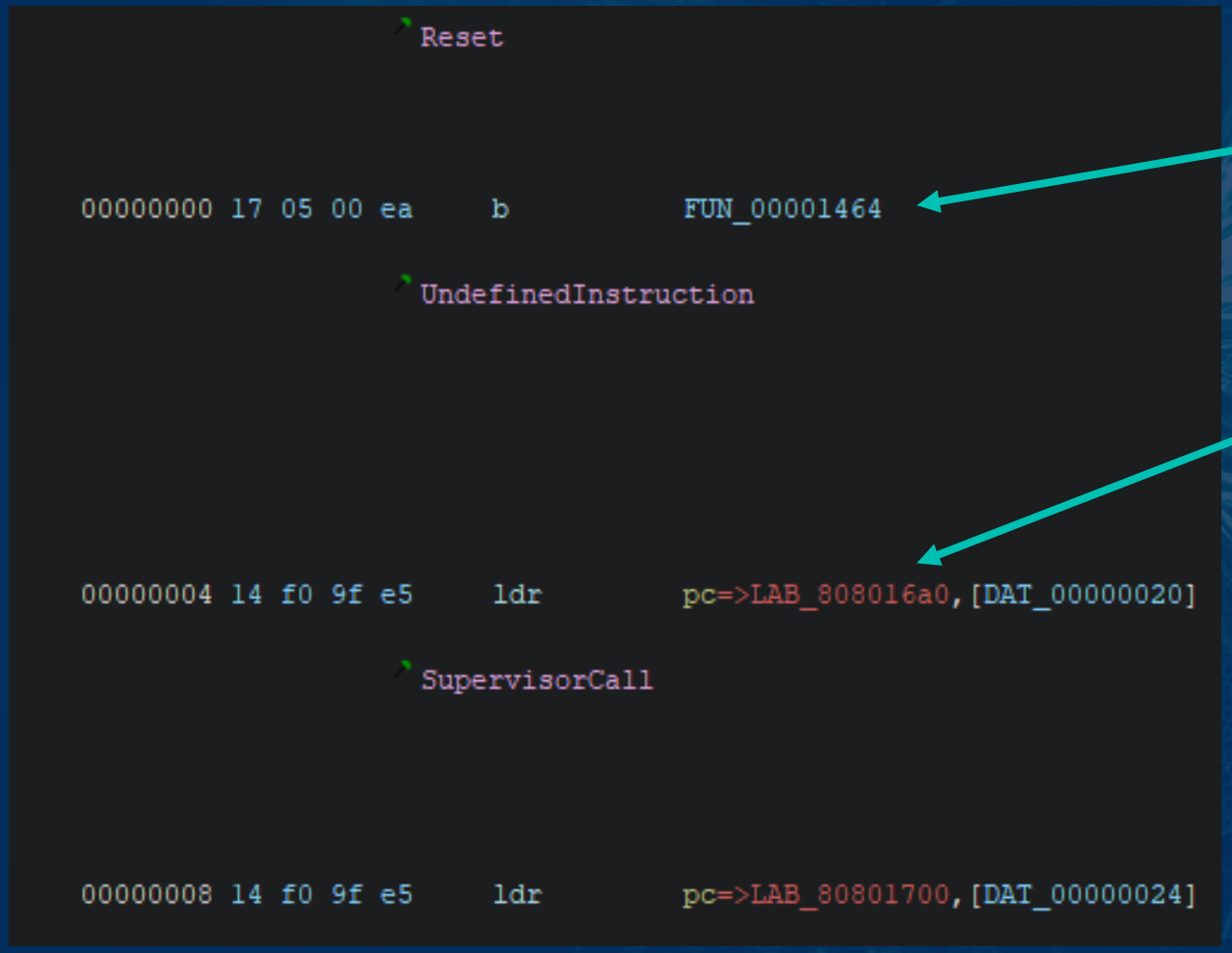
TARLOGIC
CYBERSECURITY EXPERTS

BOOTLOADER

- Does not seem encrypted
- U-Boot 2010.06-svn4868 (probably with modifications)
- Architecture is some kind of ARM little endian
- Kernel load Address is 0x80008000
- And we have a binary
- Let’s launch Ghidra and load the binary at 0x0

TARLOGIC
CYBERSECURITY EXPERTS

BOOTLOADER



RELATIVE JUMP

ABSOLUTE JUMP

0x8080 16a0

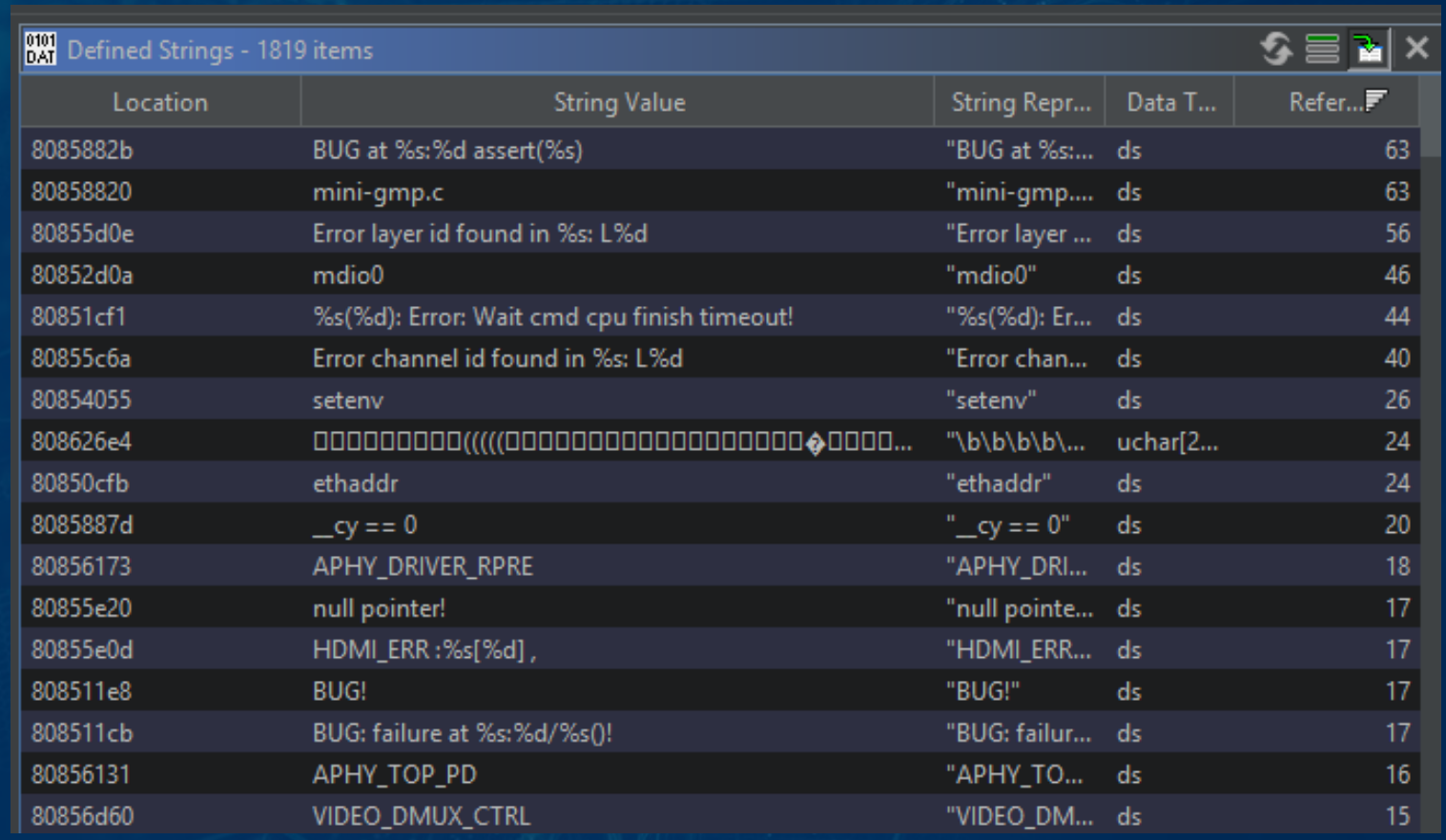
LOAD ADDRESS?

0x8080 0000

TARLOGIC
CYBERSECURITY EXPERTS

BOOTLOADER

- Load at 0x80800000
- Validate that the address is correct
- Strings have references so probably yes!



Location	String Value	String Repr...	Data T...	Refer...
8085882b	BUG at %s:%d assert(%s)	"BUG at %s:...	ds	63
80858820	mini-gmp.c	"mini-gmp....	ds	63
80855d0e	Error layer id found in %s: L%d	"Error layer ...	ds	56
80852d0a	mdio0	"mdio0"	ds	46
80851cf1	%s(%d): Error: Wait cmd cpu finish timeout!	"%s(%d): Er...	ds	44
80855c6a	Error channel id found in %s: L%d	"Error chan...	ds	40
80854055	setenv	"setenv"	ds	26
808626e4	□□□□□□□□□□((((□□□□□□□□□

TARLOGIC
CYBERSECURITY EXPERTS

BOOTLOADER

- First rule of reverse engineering?
- Do not reverse what is open source!
- https://github.com/u-boot/u-boot/tree/v2010.06
- Compare strings known to be used!

TARLOGIC
CYBERSECURITY EXPERTS

BOOTLOADER

```
static __inline__ int abortboot(int bootdelay) {
    int abort = 0;
    printf("Hit any key to stop autoboot: %2d ",
bootdelay);

    while ((bootdelay > 0) && (!abort)) {
        int i;
        --bootdelay;
        for (i=0; !abort && i<100; ++i) {
            if (tstc()) {    /* we got a key press    */
                abort  = 1; /* don't auto boot     */
                bootdelay = 0;  /* no more delay    */
                menukey = getc();
                break;
            }
            udelay(10000);
        }
        printf("\b\b\b%2d ", bootdelay);
    }
    putc('\n');
    return abort;
}
```

```
printf("Hit any key to stop autoboot: %2d ",bootdelay);
bVar1 = true;
abort = 0;
bootdelay_cpy = bootdelay;
while (0 < bootdelay_cpy && abort == 0) {
  bootdelay_cpy = bootdelay_cpy - 1;
  abort = 0x65;
  while (abort = abort - 1, abort != 0) {
    char_available = tstc();
    if (char_available != 0) {
      char = getc();
      if (char == '*') {
        char_matches = char_matches + 1;
        if (char_matches == 3) {
          bootdelay_cpy = 0;
        }
        abort = (uint)(char_matches == 3);
      }
      else {
        abort = 0;
      }
      break;
    }
    udelay(10000);
    dh_gpio_read(13,5,&pin_value);
    if (false) {
      bVar1 = false;
    }
  }
  printf("\b\b\b%2d ",bootdelay_cpy);
}
```

TARLOGIC
CYBERSECURITY EXPERTS

BOOTLOADER

```
len = readline (CONFIG_SYS_PROMPT);
flag = 0;   /* assume no special flags for now */
if (len > 0)
    strcpy (lastcommand, console_buffer);
    flag |= CMD_FLAG_REPEAT;
    return;     /* retry autoboot */
}


if (len == -1)
    puts ("<INTERRUPT>\n");
else
    rc = run_command (lastcommand, flag);
```

```
    len = readline("hisilicon # ");
    if (len < 1) {
      if (len != -1) {
        flag = (uint)(len == 0);
        goto LAB_808198ec;
      }
      puts("<INTERRUPT>\n");
    }
    else {
      strcpy(&lastcommand,&console_buffer);
      flag = 0;
LAB_808198ec:
      run_command(&lastcommand,flag);
    }
```

TARLOGIC
CYBERSECURITY EXPERTS

BOOTLOADER

- Until this moment, we have used builtin or standard types for reversing
  - char, char*, int…
- For custom types, we have to define them in Ghidra manually
- Or parse them from source…
- https://github.com/antoniovazquezblanco/GhidraExtendedSourceParser

ithub.com/u-boot/u-boot/blob/v2010.06/include/command.h

u-boot / include / command.h

Code | Blame | 128 lines (102 loc)

```
40
41  #ifndef __ASSEMBLY__
42  /*
43   * Monitor Command Table
44   */
45
46  struct cmd_tbl_s {
47      char        *name;      /
48      int     maxargs;    /* ma
49      int     repeatable; /* au
50                      /* Implem
51      int     (*cmd)(struct cmd
52      char        *usage;     /
53  #ifdef  CONFIG_SYS_LONGHELP
54      char        *help;      /
55  #endif
56  #ifdef CONFIG_AUTO_COMPLETE
57      /* do auto completion on
58      int     (*complete)(int a
59  #endif
60  };
61
62  typedef struct cmd_tbl_s    c
63
64  extern cmd_tbl_t  __u_boot_cmd_start;
65  extern cmd_tbl_t  __u_boot_cmd_end;
66
67
68  /* common/command.c */
69  int _do_help (cmd_tbl_t *cmd_start, int cmd_items, cmd_tbl_t * cmdtp, int
```

Program Trees

u-boot.bin
ram
ram.exp

Program Tree

Symbol Tree

do_devid
do_fat_fsload
do_fatls
do_hello_wor
do_kaimenda
do_lock_otp
do_nand
do_nandops

Filter:

Data Typ...

stddef.h
stdint.h
stdio.h
stdlib.h
types.h
uboot
bool
bootargs_st

Filter:

Listing: u-boot.bin

```
                        LAB_80814eec                            XREF[1]:     80814ed0
80814eec 18 40 84 e2    add         r4,r4,#0x18
80814ef0 ef ff ff ea    b           LAB_80814eb4

                        LAB_80814ef4                            XREF[1]:     80814eb8
80814ef4 01 00 56 e3    cmp         r6,#0x1
80814ef8 0a 00 a0 01    cpyeq       p,r10
80814efc 00 00 a0 13    movne       p,#0x0
80814f00 f0 8e bd e8    ldmia       sp!,{r4,len,r6,r7,r9,r10,r11,pc}

                        LAB_80814f04                            XREF[1]:     80814ee0
80814f04 04 00 a0 e1    cpy         p,r4
80814f08 f0 8e bd e8    ldmia       sp!,{r4,len,r6,r7,r9,r10,r11,pc}

                        ****************************************************************
                        *                          FUNCTION                            *
                        ****************************************************************
                        cmd_tbl_t * __stdcall find_cmd(char * cmd)
    cmd_tbl_t *         r0:4        <RETURN>                    XREF[1]:
    char *              r0:4        cmd                         XREF[1]:
    undefined4          r0:4        cmdtblptr                   XREF[1]:
                        find_cmd                                XREF[6]:     FUN_8081
                                                                             FUN_8081
                                                                             FUN_8081
                                                                             FUN_8081
                                                                             run_comm
                                                                             FUN_8081
80814f0c 14 10 9f e5    ldr         r1=>cmd_table,[->cmd_table]
80814f10 14 30 9f e5    ldr         r3,[PTR_DAT_80814f2c]
80814f14 14 20 9f e5    ldr         r2,[DAT_80814f30]
80814f18 03 30 61 e0    rsb         r3,r1,r3
80814f1c c3 31 a0 e1    mov         r3,r3, asr #0x3
80814f20 92 03 02 e0    mul         r2,r2,r3
80814f24 d2 ff ff ea    b           find_cmd_tbl
                        -- Flow Override: CALL_RETURN (CALL_TERMINATOR)

                        PTR_cmd_table_80814f28                  XREF[1]:     find_cmd
80814f28 78 2c 86 80    addr        cmd_table
```

Decompile:...

```
1
2  cmd_tbl_t * find_cmd(char *cmd)
3
4  {
5    cmd_tbl_t *cmdtblptr;
6
7    cmdtblptr = find_cmd_tbl(cmd,&cmd_table,
8    return cmdtblptr;
9  }
10
```

Decompile: find_cmd | Defined Str

80814f24 | find_cmd | b 0x80814e74

```
58  int             (*complete)(int argc, char
58  int argc, char *argv[], char last_char, in
58  last_char, int maxv, char *cmdv[]);
69  int _do_help (cmd_tbl_t *cmd_start, int cm
```

TARLOGIC
CYBERSECURITY EXPERTS

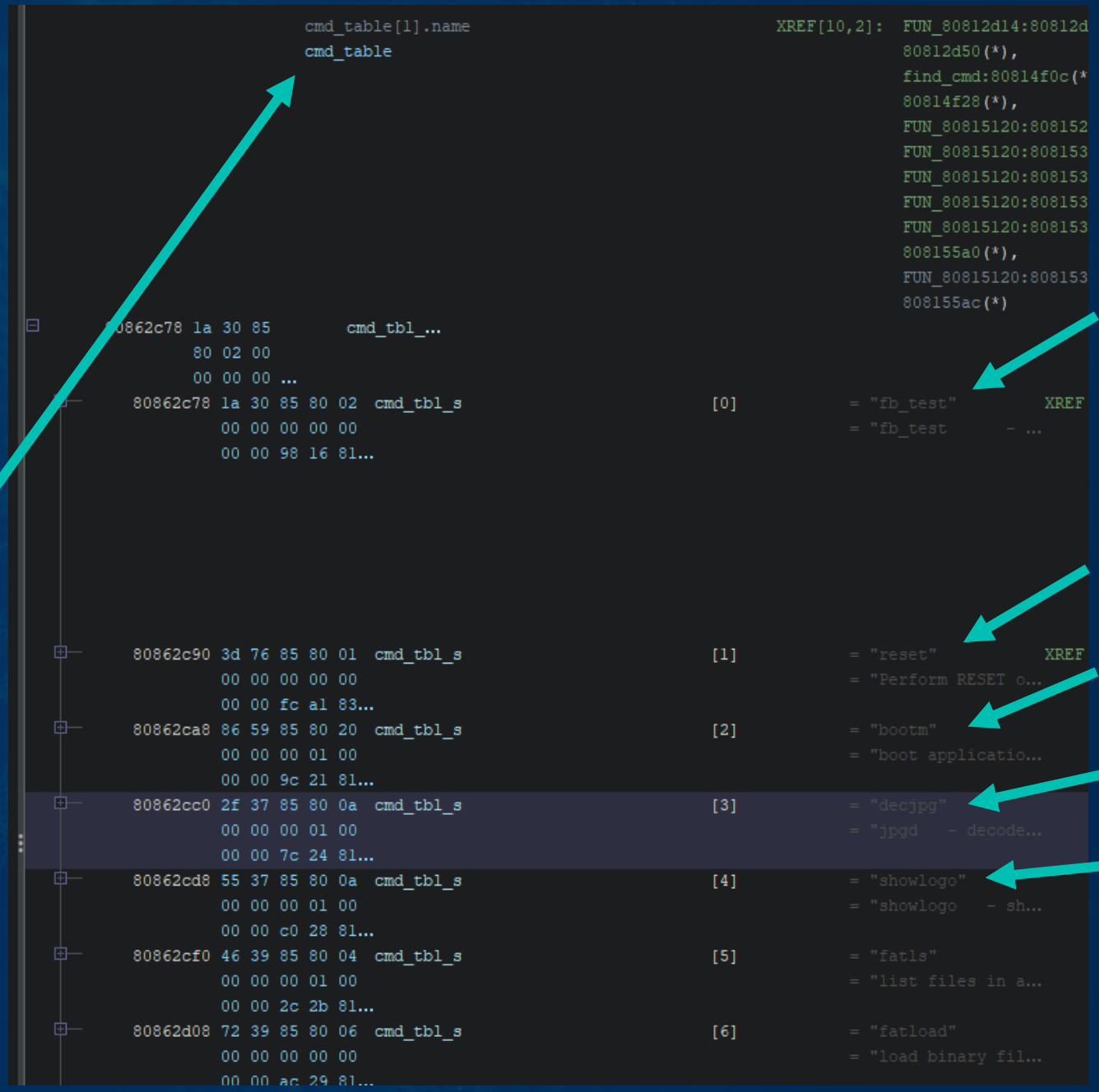
BOOTLOADER

- Follow the code…
- Define the table description types…

```
cmd_tbl_t * find_cmd(char *cmd)

{
  cmd_tbl_t *cmdtblptr;

  cmdtblptr = find_cmd_tbl(cmd,cmd_table,42);
  return cmdtblptr;
}
```



TARLOGIC
CYBERSECURITY EXPERTS

BOOTLOADER

```
80862e28 c1 3e 85 80 20  cmd_tbl_s                          [18]
         00 00 00 01 00
         00 00 1c 3a 81...
   80862e28 c1 3e 85 80     char *     s_xhprintenv_80853ec1    name
   80862e2c 20 00 00 00     int        20h                      maxargs
   80862e30 01 00 00 00     int        1h                       repeatable
   80862e34 1c 3a 81 80     uintptr_...do_xhprintenv            cmd
   80862e38 01 42 85 80     char *     s_print_environment_va... usage
   80862e3c 00 00 00 00     char *     00000000                 help
```

- Some commands are guarded behind a function that checks if the device is “locked”…
- That function checks if the device is locked and if the command is forbidden in the locked state…

```
2  int do_xhprintenv(cmd_tbl_t *cmdtp,int flag,int argc,char **argv)
3
4  {
5    int iVar1;
6    int iVar2;
7    int i;
8    char *param;
9
10   iVar1 = is_cmd_locked("xhprintenv",(char *)0x0,(char *)0x0);
11   if (iVar1 == 0) {
12     iVar1 = 0;
13     if (argc != 1) {
14       for (i = 1; i < argc; i = i + 1) {
15         param = argv[i];
16         iVar2 = FUN_808138bc(param,2);
17         if (iVar2 != 0) {
18           printf("## Error: \"%s\" not defined\n",param);
19           iVar1 = iVar1 + 1;
20         }
21       }
22       return iVar1;
23     }
24     iVar1 = FUN_808138bc((char *)0x0,1);
25     if (-1 < iVar1) {
26       printf("\nEnvironment size: %d/%ld bytes\n",iVar1,0x1fffc);
27       return 0;
28     }
29   }
30   return 1;
31 }
```

TARLOGIC
CYBERSECURITY EXPERTS

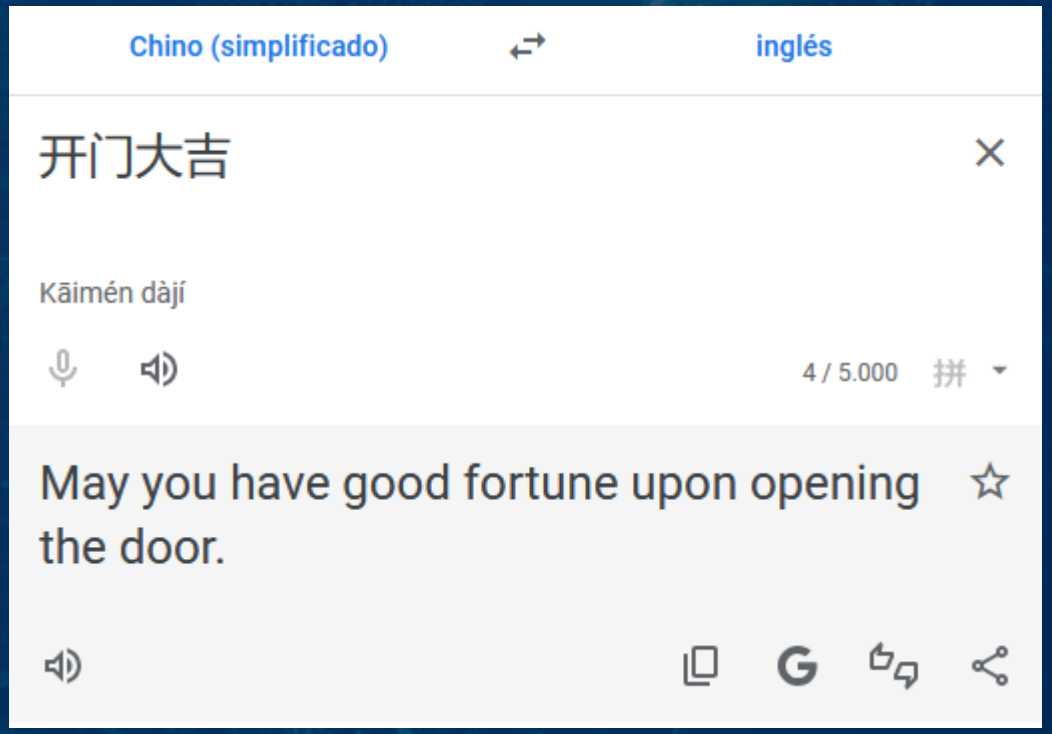
BOOTLOADER

- Following the trails of the function that checks the device “locked” state we find…
- Password user input is stored in the “dajidali” environment variable
- The input password is checked using a function
- That function is related to the custom “kaimendaji” bootloader command

```
2  int is_device_unlocked(void)
3
4  {
5    char *pw;
6    int iVar1;
7
8    DAT_80922bb8 = &PTR_s_Hello_World!!_808629c8;
9    pw = getenv("dajidali");
10   if (pw != (char *)0x0) {
11     iVar1 = kaimendaji_pw_check(pw,6);
12     return iVar1;
13   }
14   return 1;
15 }
```

KAIMEMDAJI

- But wait…



TARLOGIC
CYBERSECURITY EXPERTS

KAIMEMDAJI

- But wai

TARLOGIC
CYBERSECURITY EXPERTS

KAIMEMDAJI

- The check password function seems to use some kind of cryptography…
- https://github.com/antoniovazquezblanco/GhidraFindcrypt
- This plugin looks for cryptographic constants and tags them!
- Among others, find a function relevant to kaimendaji…
- Now, lets understand the rest…

```
4  void MD5_hash(void *data,size_t size,byte *hash)
5
6  {
7    MD5_CTX ctx;
8
9    ctx.A = 0x67452301;
10   ctx.B = 0xefcdab89;
11   ctx.C = 0x98badcfe;
12   ctx.D = 0x10325476;
13   ctx.Nl = 0;
14   ctx.Nh = 0;
15   MD5_Update(&ctx,data,size);
16   MD5_Final(hash,&ctx);
17   return;
18 }
```

TARLOGIC
CYBERSECURITY EXPERTS

KAIMEMDAJI

- “Kaimendaji” asks for a parameter, the password
- Checks the length of the password
- Calls a checking function…

```
int do_kaimendaji(cmd_tbl_t *cmdtp,int flag,int argc,char **argv)

{
  char *msg;
  size_t pwlen;
  int iVar1;

  if (argc == 1) {
    msg = "kaimendaji paswwd\n";
  }
  else {
    if (argc != 2) {
      printf("Parameter error!\n");
      return -1;
    }
    pwlen = strlen(argv[1]);
    if (pwlen != 6) {
      printf("invalid paswwd len\n");
    }
    strcpy((char *)kaimendaji_pw,argv[1]);
    kaimendaji_struct_ptr = &kaimendaji_struct;
    iVar1 = kaimendaji_pw_check((char *)kaimendaji_pw,6);
    if (iVar1 == 0) {
      setenv("dajidali",(char *)kaimendaji_pw);
      nand_save_env?();
      msg = "Hello World!\n";
    }
    else {
      msg = "invalid paswwd\n";
    }
  }
  printf(msg);
  return 0;
}
```

TARLOGIC
CYBERSECURITY EXPERTS

KAIMEMDAJI

- The password in ASCII format is converted to (3) bytes

- Another function is called to check the 3 bytes

```
2  int kaimendaji_pw_check(char *pw,uint len)
3
4  {
5    int ret;
6    byte pw_bin [4];
7    uint local_c;
8
9    pw_bin[0] = 0;
10   pw_bin[1] = 0;
11   pw_bin[2] = 0;
12   pw_bin[3] = 0;
13   local_c = len & 0xffff0000;
14   if (len == 6 && pw != (char *)0x0) {
15     hexstring_to_binary(pw_bin,pw);
16     ret = kaimendaji_pwbytes_check(pw_bin,(char *)0x0,(char *)0x0);
17   }
18   else {
19     ret = -1;
20   }
21   return ret;
22 }
```

KAIMEMDAJI

- The function internally tries to obtain the ID and MAC of the device

```
23      if ((code *)kaimendaji_struct_ptr->get_env_id != (code *)0x0) {
24        (*(code *)kaimendaji_struct_ptr->get_env_id)(id_str,0x100);
25        goto LAB_8084857c;
26      }
```

```
38 LAB_808485b0:
39      if ((code *)kaimendaji_struct_ptr->get_env_ethaddr != (code *)0x0) {
40        (*(code *)kaimendaji_struct_ptr->get_env_ethaddr)(ethaddr_str,0x100);
41      }
42      goto LAB_808485c8;
```

TARLOGIC
CYBERSECURITY EXPERTS

KAIMEMDAJI

- The function internally tries to obtain the ID and MAC of the device
- Performs an MD5 of each of those variables
- Mixes them using XOR in a new function
- Checks the result against our input

```
45 LAB_80848584:
46   len = strlen(ethaddr);
47   memcpy(ethaddr_str,ethaddr,len);
48 LAB_808485c8:
49   memset(id_md5,0,16);
50   memset(ethaddr_md5,0,16);
51   len = strlen(id_str);
52   MD5((uchar *)id_str,len,id_md5);
53   len = strlen(ethaddr_str);
54   MD5((uchar *)ethaddr_str,len,ethaddr_md5);
55   kaimendaji_xor(id_md5,ethaddr_md5,pass_good);
56   ret = memcmp(pass_good,pwbytes,6);
57   return ret;
```

TARLOGIC
CYBERSECURITY EXPERTS

KEYGEN

```python
#!/usr/bin/env python

import hashlib

def xorfun(hash1, hash2):
    enc1 = b'\x01\x03\x09\x17\x14\x12'
    enc2 = b'\x0e\x1e\x16\x07\x09\x20'
    out = b''
    for i in range(6):
        index1 = ((enc1[i] + 1) >> 1) - 1
        off1 = (enc1[i] & 1) * 4
        byte1 = (hash1[index1] >> off1) & 0xf
        index2 = ((enc2[i] + 1) >> 1) - 1
        off2 = (enc2[i] & 1) * 4
        byte2 = (hash2[index2] >> off2) & 0xf
        out += (byte1 ^ byte2).to_bytes(1, 'big')
    return out

def pass_generate(dev_addr, dev_id):
    addr_hash = hashlib.md5(dev_addr).digest()
    id_hash = hashlib.md5(dev_id).digest()
    return xorfun(id_hash, addr_hash)
```

https://github.com/TarlogicSecurity/advisories

TARLOGIC
CYBERSECURITY EXPERTS

SHOW HELLO

- A second look at the commands table reveals another custom command…

```
2  int do_hello_world(cmd_tbl_t *cmdtp,int flag,int argc,char **argv)
3
4  {
```

```
29   getenv_id(id_str,256);
30   getenv_ethaddr(ethaddr_str,256);
```

```
53   packSrc(ethaddr_clean_str,id_str,msg);
54   printf("hello world!  %s\n",msg);
55   return 0;
56 }
```

TARLOGIC
CYBERSECURITY EXPERTS

SHOW HELLO

- A second look at the

TARLOGIC
CYBERSECURITY EXPERTS

UNPACKSRC

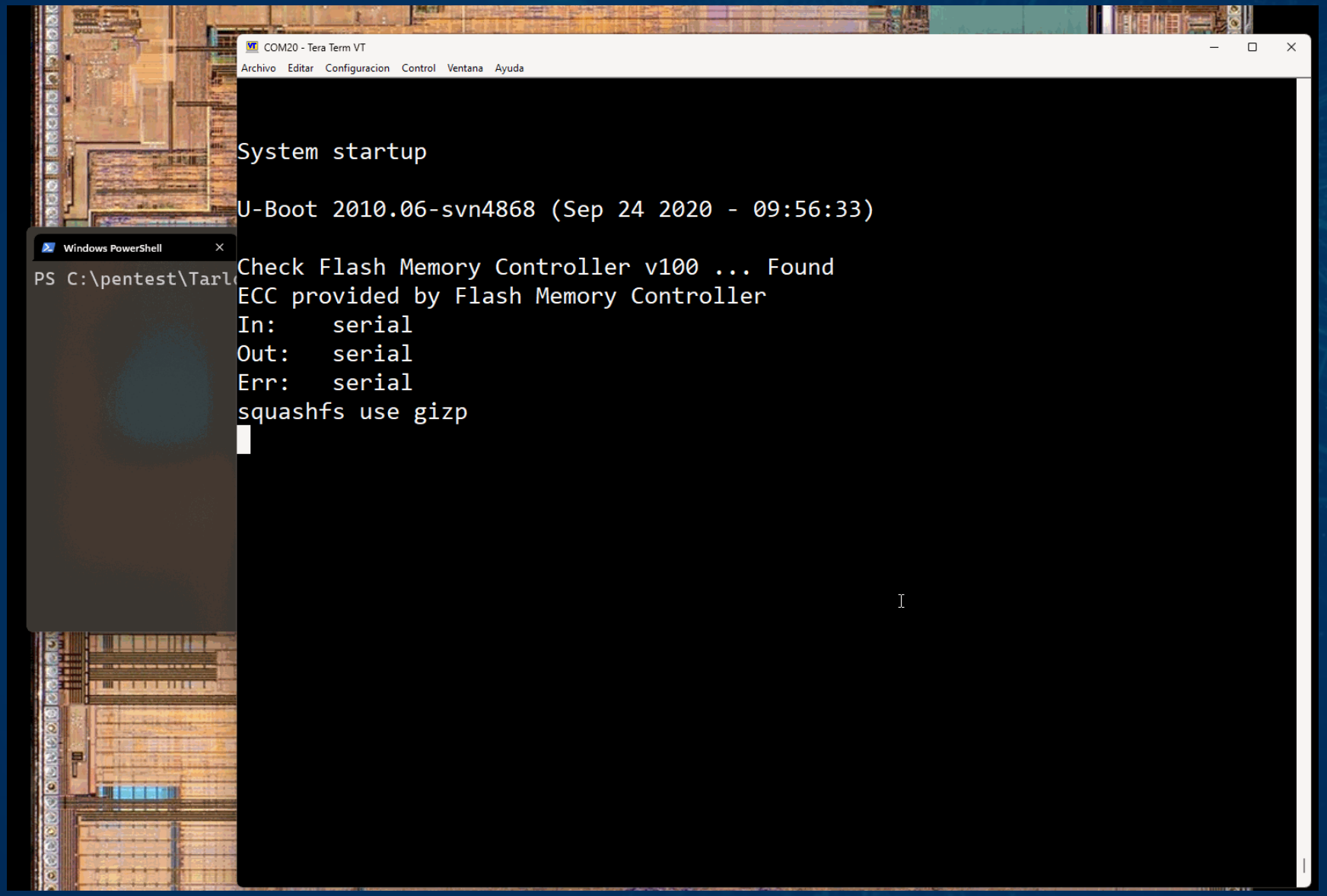
```
#!/usr/bin/env python

def unpack_src(s):
    if len(s) < 12:
        return None, None
    s_used = [False for i in range(len(s))]
    mac = ""
    for i in range(12):
        pos = (i >> 2) * 8
        bits = i & 3
        if bits == 2:
            pos += 6
        elif bits == 3:
            pos += 7
        elif bits == 1:
            pos += 3
        else:
            pos += 2
        s_used[pos] = True
        mac += s[pos]
    mac = ':'.join(mac[i:i+2] for i in range(0, 12, 2))
    id_len = len(s) - 12
    idxs = []
    i = 0
    while len(idxs) < id_len:
        pos = (i >> 2) * 8
        bits = i & 3
        if bits != 2:
            if bits == 3:
                pos += 1
            elif bits == 1:
                pos += 5
            else:
                pos += 4
        i += 1
        idxs.append(pos)
        if pos < len(s):
            s_used[pos] = True
    missing = max(idxs) - len(s) + 1
    index = s_used.index(False)
    s = s[:index] + '_'*missing + s[index:]
    id = ''
    for p in idxs:
        id += s[p]
    return mac, id
```

https://github.com/TarlogicSecurity/advisories

TARLOGIC
CYBERSECURITY EXPERTS

PoC



TARLOGIC
CYBERSECURITY EXPERTS

COMMANDS

```
# Print environment variables
hisilicon# xhprintenv <varname>

# Load and decrypt partition
hisilicon# partload <partname>

# Does not work…
hisilicon# xhprint

# Write from memory to nand (page aligned)
hisilicon# nandops 1 <memaddr> <nandoffset>

# Dump nand page
hisilicon# nand dump <nandoffset>
```

TARLOGIC
CYBERSECURITY EXPERTS

ENVIRONMENT

```
# Enables verbose kernel & services
hisilicon# setenv dh_keyboard 0

# Stops automatic services and drops a shell
hisilicon# setenv appauto 0

# Prevents from loading some manufacturer kernel modules
hisilicon# setenv load_modules 0

# Unknown
hisilicon# setenv ch_board open
```

TARLOGIC
CYBERSECURITY EXPERTS

TIMELINE

- 2024/07/24 - Initial contact with Dahua Iberia, redirected to Dahua PSIRT, initial report.
- 2025/04/21 - Initial attempt to contact MITRE (via email).
- 2025/06/01 - Second attempt to contact MITRE (via email).
- 2025/07/16 - Second attempt to contact Dahua PSIRT.
- 2025/08/19 - Third attempt to contact MITRE (via the web form).
- 2025/10/06 - MITRE request for further version information.
- 2025/10/25 - Requested update from MITRE due to lack of email responses.
- 2025/10/31 - MITRE closed the request due to an errata in a reported version.
- 2025/10/31 - Fourth attempt to obtain a CVE via MITRE web form.

TARLOGIC
CYBERSECURITY EXPERTS

REFERENCES

- Slides:
- https://github.com/TarlogicSecurity/talks

Advisories:

- https://www.tarlogic.com/blog/
- https://github.com/TarlogicSecurity/advisories

Tools:

- https://github.com/antoniovazquezblanco/pexpect-serialspawn
- https://github.com/antoniovazquezblanco/GhidraExtendedSourceParser
- https://github.com/antoniovazquezblanco/GhidraFindcrypt

TARLOGIC
CYBERSECURITY EXPERTS

CONCLUSIONS

- This was a not common device from the security point of view
- Goes beyond the basic IoT/embedded device security:
  - Partition encryption
  - Software signature verification
  - Customized bootloader for enhanced security
- Probably due to cost/development/support reasons:
  - Bootchain was not properly designed to be Secure
  - Bootloader is not signed
  - Authentication in Bootloader is not well designed
  - Encryption can be bypassed using built in features

TARLOGIC
CYBERSECURITY EXPERTS

CONCLUSIONS

- This was a nc
- Goes beyond
  - Partition
  - Software
  - Customiz
- Probably due
  - Bootchai
  - Bootload
  - Privilege
  - Encryption can be bypassed using built in features and keys are not properly protected

TARLOGIC
CYBERSECURITY EXPERTS

THANK YOU!

contacto@tarlogic.com

www.tarlogic.com
+34 912 919 319