ONEKEY

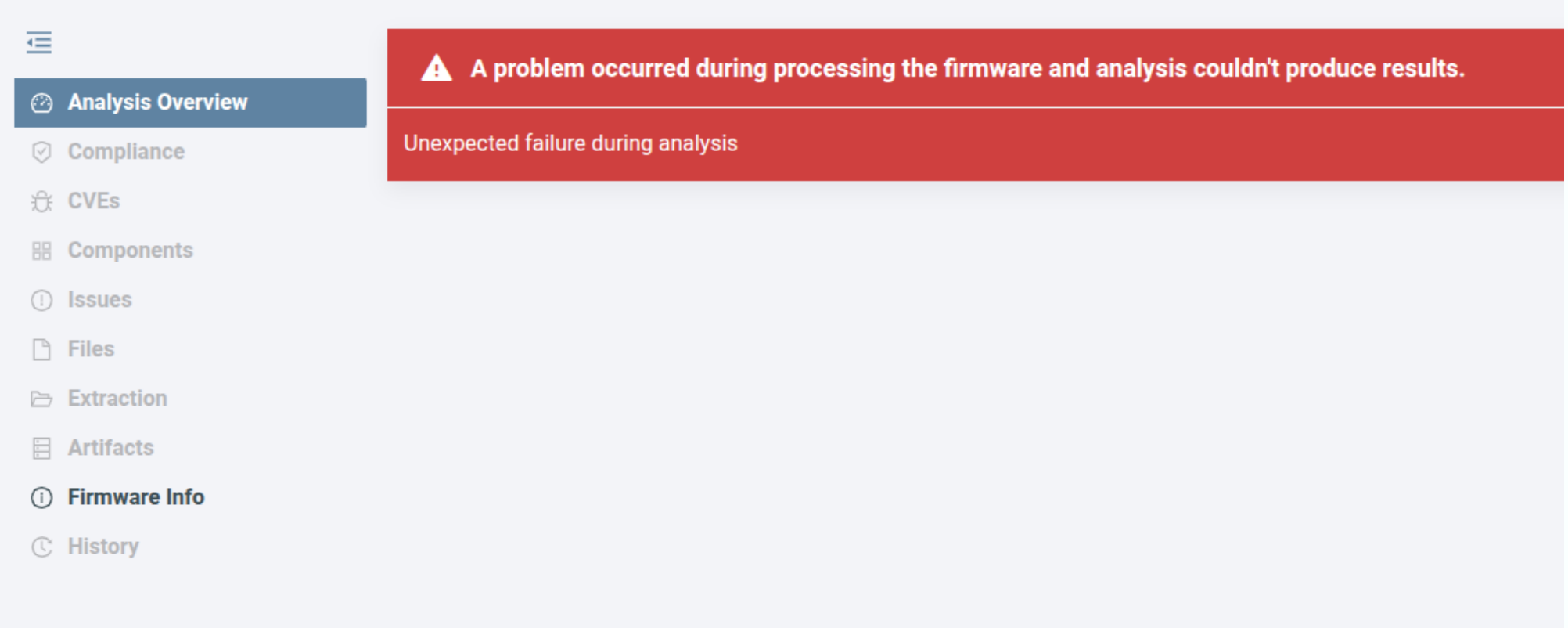
__SALTY FIRMWARE

ADVENTURES IN FIRMWARE ENCRYPTION REVERSING

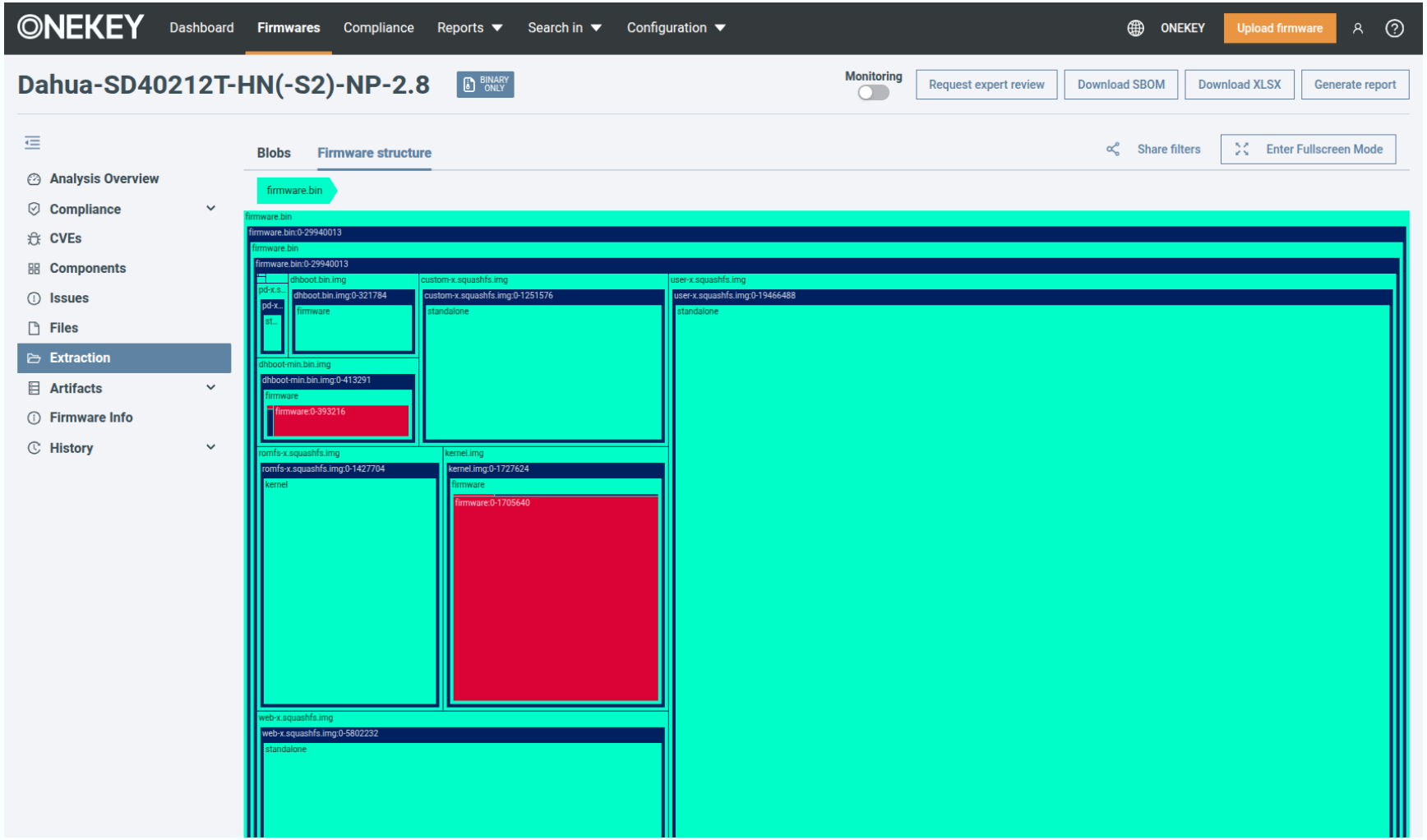
HI !

- OSS maintainer (unblob, jefferson, ubi-reader, sasquatch, arpy)
- Reverse engineer working in R&D on a vast array of embedded firmware (Linux, Android, QNX, VxWorks, FreeRTOS, Zephyr, eCOS, …)
- Trying to apply/develop modern techniques and tooling to analyze those firmware
- @qkaiser everywhere

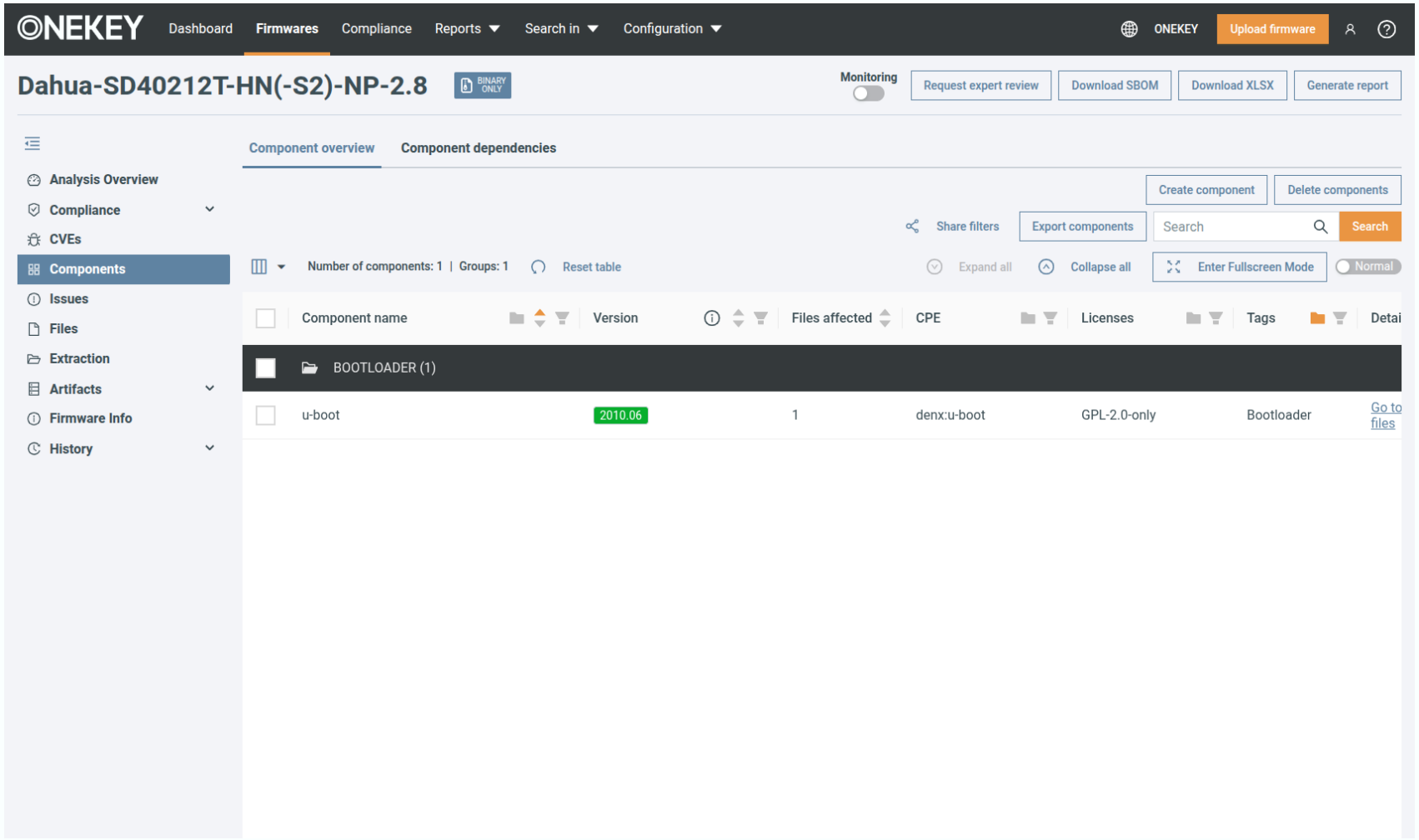
OUR VERY OWN BSOD



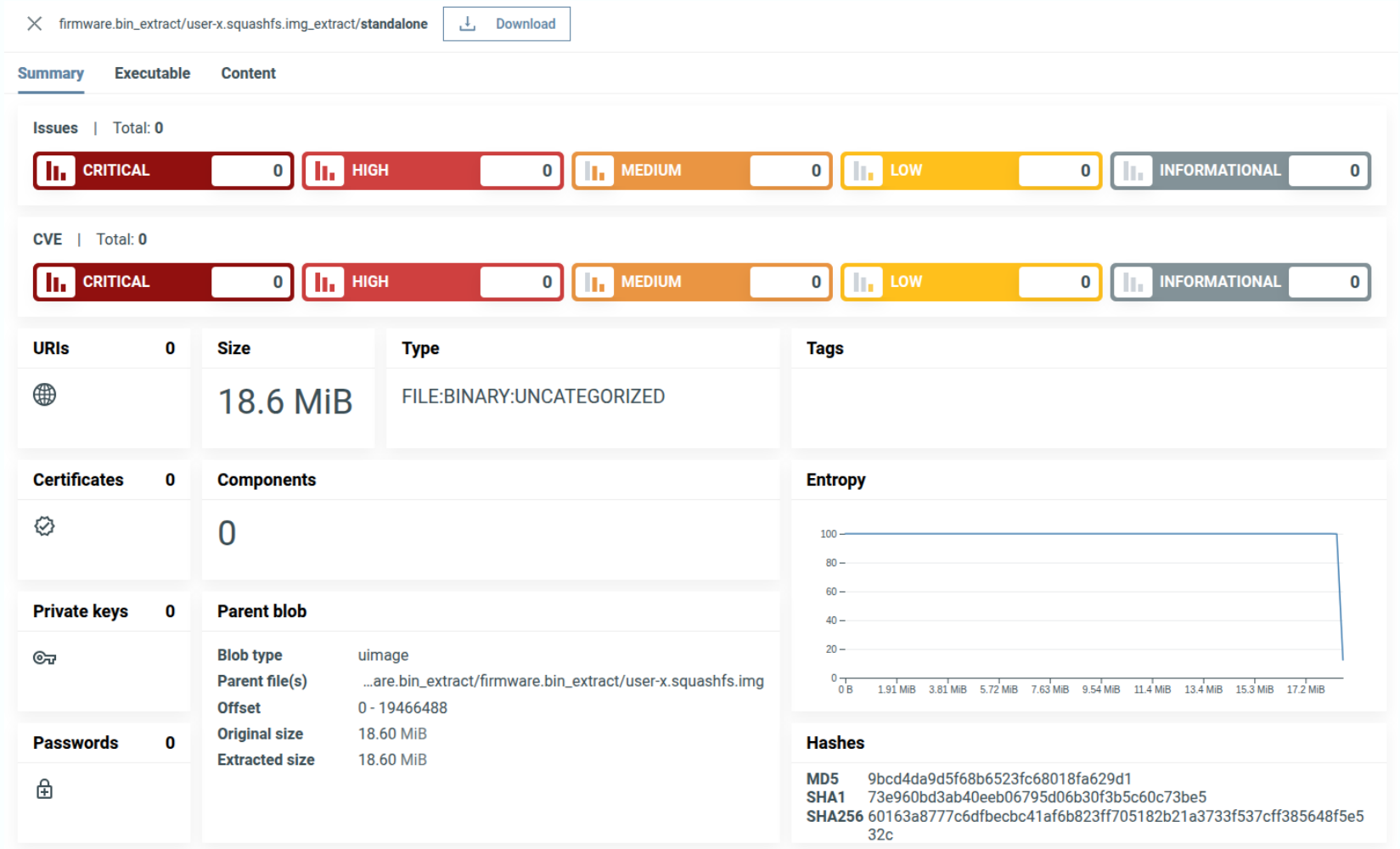
OUR VERY OWN BSOD



OUR VERY OWN BSOD



OUR VERY OWN BSOD



AGENDA

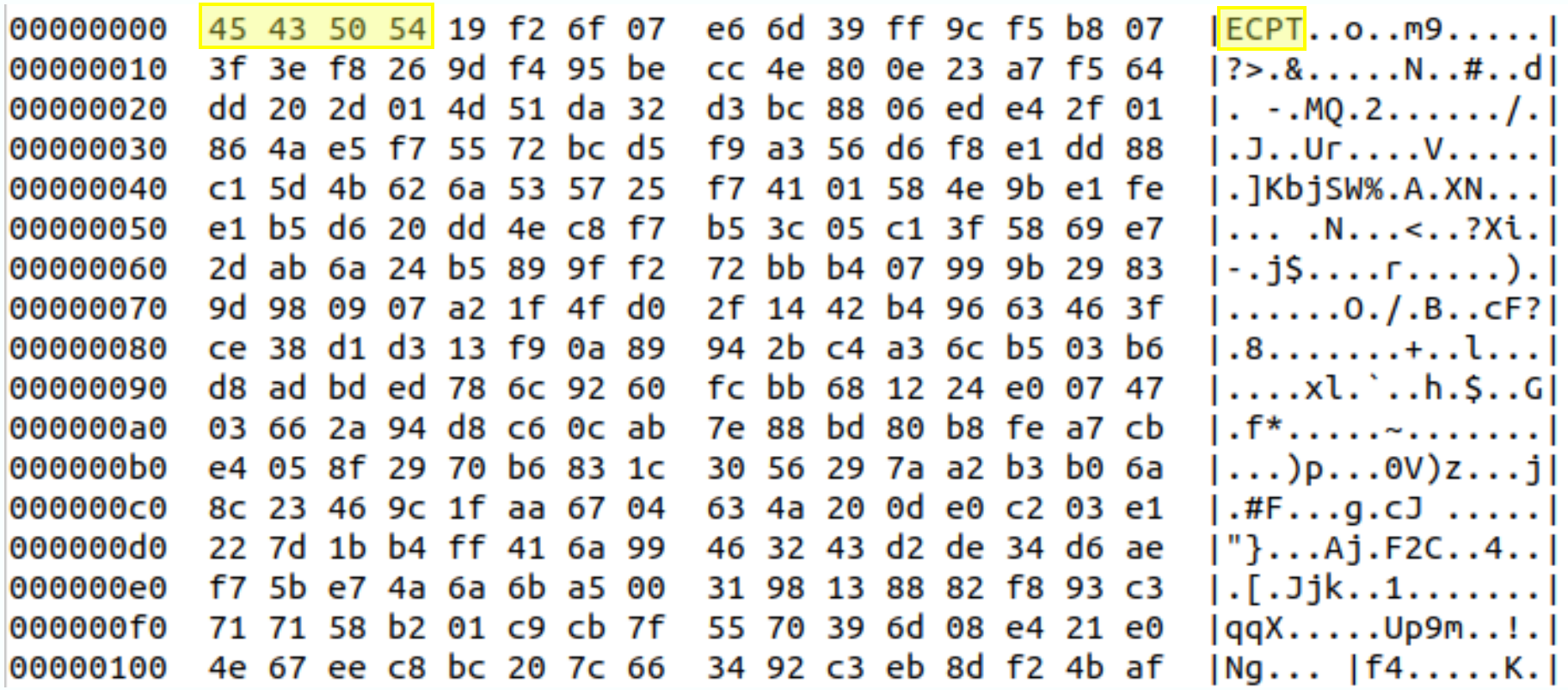
- TARGET A – BOOTLOADER
- TARGET B – BOOTLOADER
- TARGET C – LINUX KERNEL DRIVER
- TARGET D – LINUX KERNEL CUSTOM FS
- TARGET E – UPDATE BINARY

BOOTLOADER - PRINTER MANUFACTURER

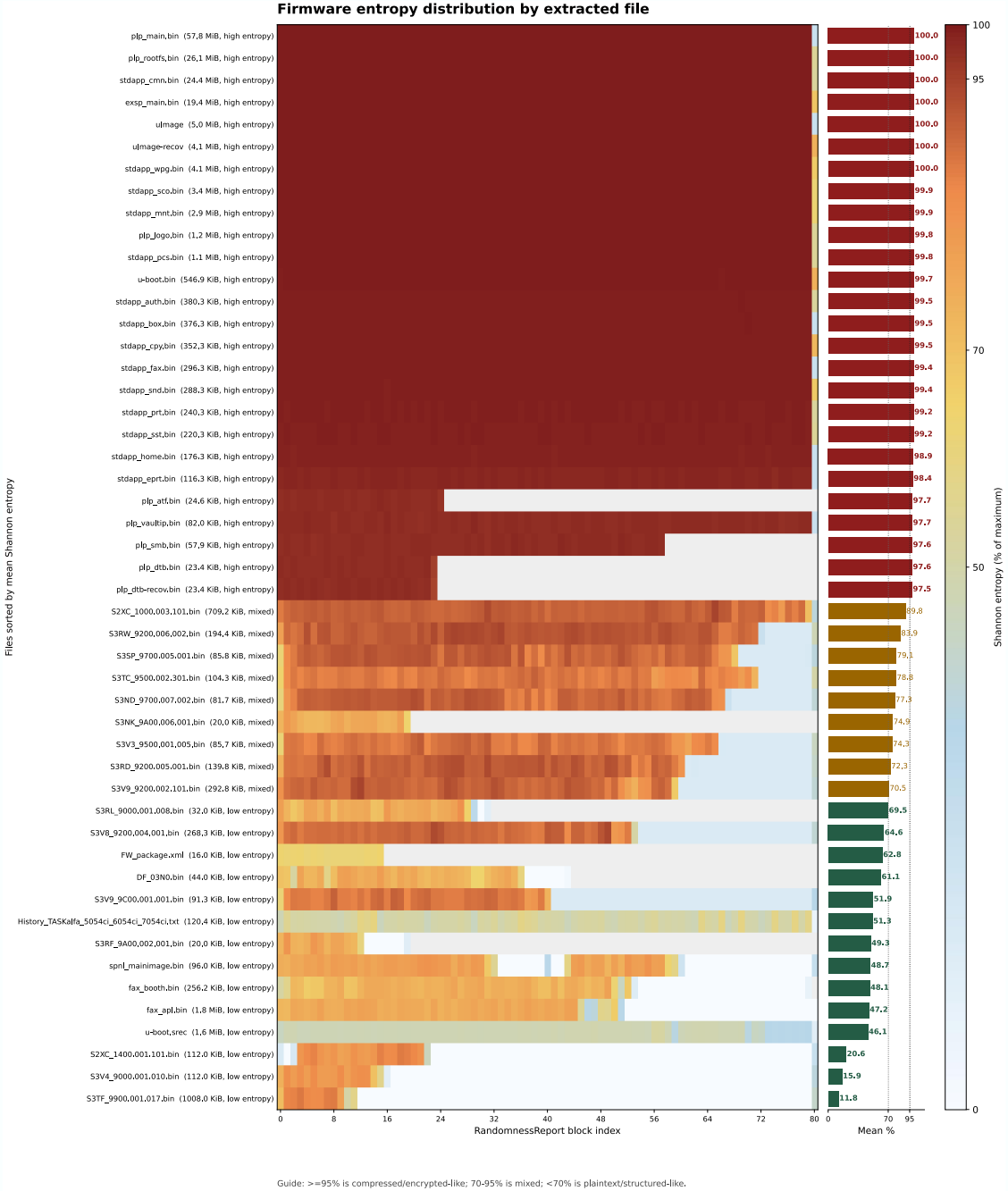
BOOTLOADER - PRINTER MANUFACTURER

2ND_03N0_cert.pem	2XC_AI_SPR_cert.pem	2XC_CTRL_STDAPP_WPG_cert.pem	DL_03TC.2XC	DL_OPT.2XC	S3TF_9900.001.017.bin
2ND_03N0_sign.bin	2XC_AI_SPR_sign.bin	2XC_CTRL_STDAPP_WPG_sign.bin	DL_03TF.2XC	DL_PKG_CTRL.2XC	S3V3_9500.001.005.bin
2ND_03ND_cert.pem	2XC_CTRL_EXSP_cert.pem	2XC_CTRL_VINF_cert.pem	DL_03V3.2XC	DL_SPNL.2XC	S3V4_9000.001.010.bin
2ND_03ND_sign.bin	2XC_CTRL_EXSP_sign.bin	2XC_CTRL_VINF_sign.bin	DL_03V4.2XC	ES_SKIP.ON	S3V8_9200.004.001.bin
2ND_03RD_cert.pem	2XC_CTRL_PLP_cert.pem	2XC_ENGN_cert.pem	DL_03V8.2XC	exsp_main.bin	S3V9_9200.002.101.bin
2ND_03RD_sign.bin	2XC_CTRL_PLP_sign.bin	2XC_ENGN_sign.bin	DL_03V9.2XC	fax_apl.bin	S3V9_9C00.001.001.bin
2ND_03RF_cert.pem	2XC_CTRL_STDAPP_AUTH_cert.pem	2XC_M2XC_cert.pem	DL_05MR.2XC	fax_booth.bin	S5MR_9D00.001.010.bin
2ND_03RF_sign.bin	2XC_CTRL_STDAPP_AUTH_sign.bin	2XC_M2XC_sign.bin	DL_AI_EHW.2XC	FW_package.xml	S5MR_9D00.001.010.bin_extract
2ND_03RL_cert.pem	2XC_CTRL_STDAPP_BOX_cert.pem	2XC_M3V9_cert.pem	DL_AI_SPR.2XC	handwriting.zip	spnl_mainimage.bin
2ND_03RL_sign.bin	2XC_CTRL_STDAPP_BOX_sign.bin	2XC_M3V9_sign.bin	DL_CTRL_EXSP.2XC	handwriting.zip_extract	stdapp_auth.bin
2ND_03RW_cert.pem	2XC_CTRL_STDAPP_CMN_cert.pem	2XC_OCR_cert.pem	DL_CTRL_PLP.2XC	ocr_data.zip	stdapp_box.bin
2ND_03RW_sign.bin	2XC_CTRL_STDAPP_CMN_sign.bin	2XC_OCR_sign.bin	DL_CTRL_STDAPP_AUTH.2XC	ocr_data.zip_extract	stdapp_cmn.bin
2RH_03NK_cert.pem	2XC_CTRL_STDAPP_CPY_cert.pem	2XC_OPT_cert.pem	DL_CTRL_STDAPP_BOX.2XC	optlangimage.zip	stdapp_cpy.bin
2RH_03NK_sign.bin	2XC_CTRL_STDAPP_CPY_sign.bin	2XC_OPT_sign.bin	DL_CTRL_STDAPP_CMN.2XC	optlangimage.zip_extract	stdapp_eprt.bin
2TJ_03SP_cert.pem	2XC_CTRL_STDAPP_EPRT_cert.pem	2XC_SPNL_cert.pem	DL_CTRL_STDAPP_CPY.2XC	plp_atf.bin	stdapp_fax.bin
2TJ_03SP_sign.bin	2XC_CTRL_STDAPP_EPRT_sign.bin	2XC_SPNL_sign.bin	DL_CTRL_STDAPP_EPRT.2XC	plp_dtb.bin	stdapp_home.bin
2XC_03TC_cert.pem	2XC_CTRL_STDAPP_FAX_cert.pem	2xc.verinfo	DL_CTRL_STDAPP_FAX.2XC	plp_dtb-recov.bin	stdapp_mnt.bin
2XC_03TC_sign.bin	2XC_CTRL_STDAPP_FAX_sign.bin	3R2_FAX_cert.pem	DL_CTRL_STDAPP_HOME.2XC	plp_logo.bin	stdapp_pcs.bin
2XC_03TF_cert.pem	2XC_CTRL_STDAPP_HOME_cert.pem	3R2_FAX_sign.bin	DL_CTRL_STDAPP_MNT.2XC	plp_main.bin	stdapp_prt.bin
2XC_03TF_sign.bin	2XC_CTRL_STDAPP_HOME_sign.bin	COLOR_TABLE	DL_CTRL_STDAPP_PCS.2XC	plp_rootfs.bin	stdapp_sco.bin
2XC_03V3_cert.pem	2XC_CTRL_STDAPP_MNT_cert.pem	COLOR_TABLE_COPY	DL_CTRL_STDAPP_PRT.2XC	plp_smb.bin	stdapp_snd.bin
2XC_03V3_sign.bin	2XC_CTRL_STDAPP_MNT_sign.bin	COLOR_TABLE_COPY_extract	DL_CTRL_STDAPP_SCO.2XC	plp_vaultip.bin	stdapp_sst.bin
2XC_03V4_cert.pem	2XC_CTRL_STDAPP_PCS_cert.pem	COLOR_TABLE_extract	DL_CTRL_STDAPP_SND.2XC	S2XC_1000.003.101.bin	stdapp_wpg.bin
2XC_03V4_sign.bin	2XC_CTRL_STDAPP_PCS_sign.bin	DF_03N0.bin	DL_CTRL_STDAPP_SST.2XC	S2XC_1400.001.101.bin	super_resolution.zip
2XC_03V8_cert.pem	2XC_CTRL_STDAPP_PRT_cert.pem	DL_03N0.2ND	DL_CTRL_STDAPP_WPG.2XC	S3ND_9700.007.002.bin	super_resolution.zip_extract
2XC_03V8_sign.bin	2XC_CTRL_STDAPP_PRT_sign.bin	DL_03ND.2ND	DL_CTRL_VINF.2XC	S3NK_9A00.006.001.bin	u-boot.bin
2XC_03V9_cert.pem	2XC_CTRL_STDAPP_SCO_cert.pem	DL_03NK.2RH	DL_ENGN.2XC	S3RD_9200.005.001.bin	u-boot.srec
2XC_03V9_sign.bin	2XC_CTRL_STDAPP_SCO_sign.bin	DL_03RD.2ND	DL_FAX.3R2	S3RF_9A00.002.001.bin	u-boot.srec_extract
2XC_05MR_cert.pem	2XC_CTRL_STDAPP_SND_cert.pem	DL_03RF.2ND	DL_M2XC.2XC	S3RL_9000.001.008.bin	uImage
2XC_05MR_sign.bin	2XC_CTRL_STDAPP_SND_sign.bin	DL_03RL.2ND	DL_M3V9.2XC	S3RW_9200.006.002.bin	uImage-recov
2XC_AI_EHW_cert.pem	2XC_CTRL_STDAPP_SST_cert.pem	DL_03RW.2ND	DL_NWDL.2XC	S3SP_9700.005.001.bin	
2XC_AI_EHW_sign.bin	2XC_CTRL_STDAPP_SST_sign.bin	DL_03SP.2TJ	DL_OCR.2XC	S3TC_9500.002.301.bin	

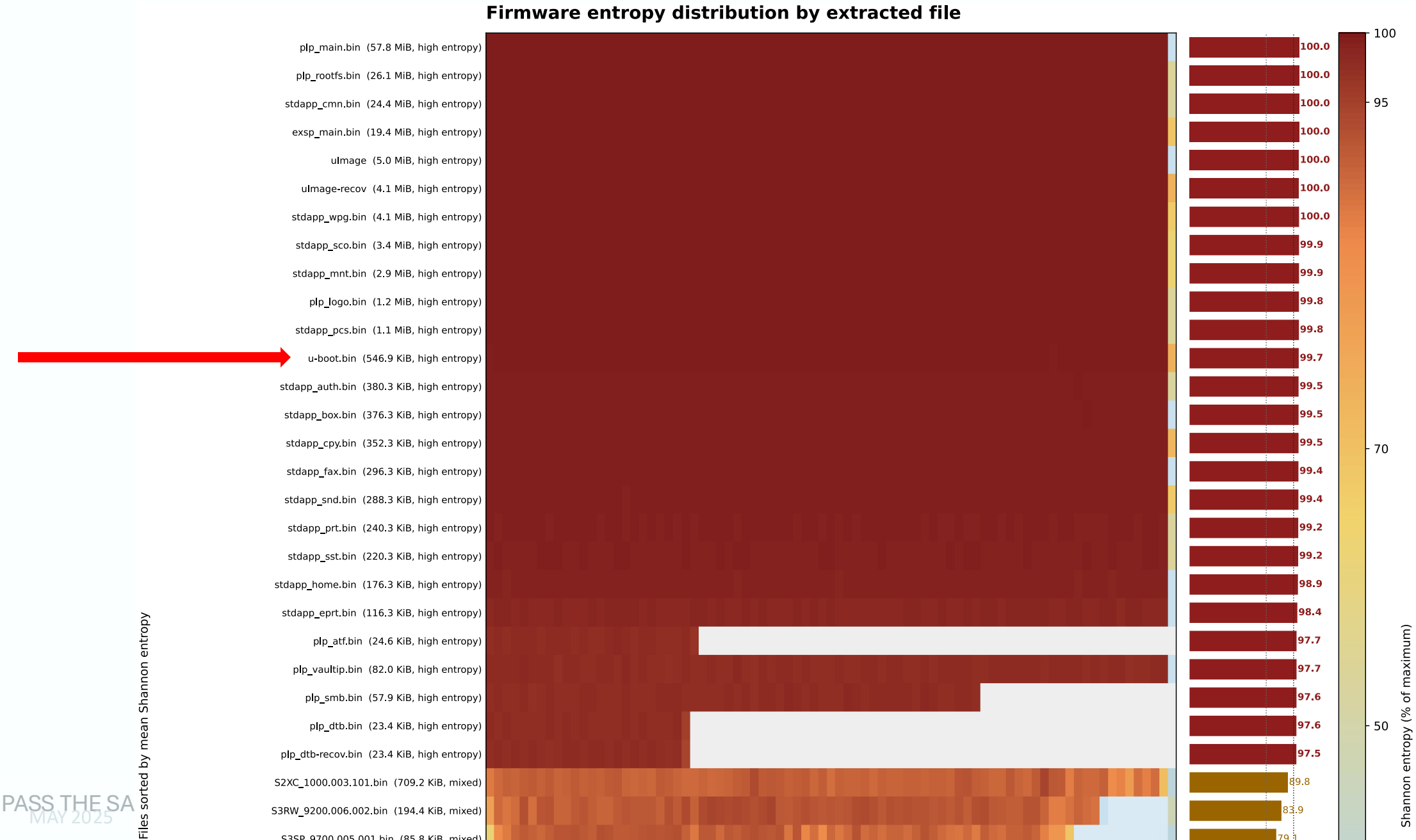
BOOTLOADER - PRINTER MANUFACTURER

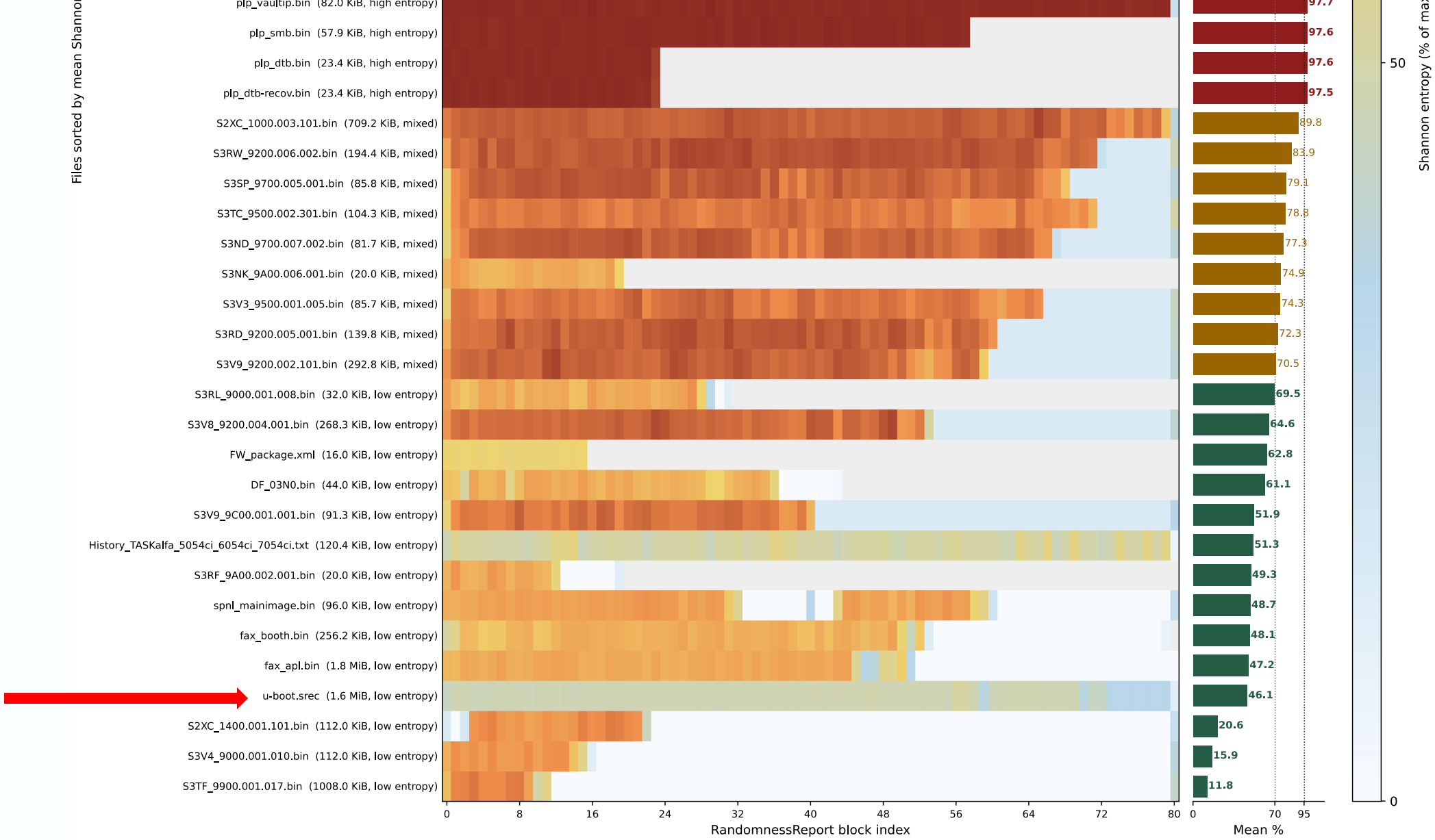


```
00000000  45 43 50 54 19 f2 6f 07  e6 6d 39 ff 9c f5 b8 07  |ECPT..o..m9.....|
00000010  3f 3e f8 26 9d f4 95 be  cc 4e 80 0e 23 a7 f5 64  |?>.&.....N..#..d|
00000020  dd 20 2d 01 4d 51 da 32  d3 bc 88 06 ed e4 2f 01  |. -.MQ.2....../.|
00000030  86 4a e5 f7 55 72 bc d5  f9 a3 56 d6 f8 e1 dd 88  |.J..Ur....V.....|
00000040  c1 5d 4b 62 6a 53 57 25  f7 41 01 58 4e 9b e1 fe  |.]KbjSW%.A.XN...|
00000050  e1 b5 d6 20 dd 4e c8 f7  b5 3c 05 c1 3f 58 69 e7  |... .N...<..?Xi.|
00000060  2d ab 6a 24 b5 89 9f f2  72 bb b4 07 99 9b 29 83  |-.j$....r.....).|
00000070  9d 98 09 07 a2 1f 4f d0  2f 14 42 b4 96 63 46 3f  |......O./.B..cF?|
00000080  ce 38 d1 d3 13 f9 0a 89  94 2b c4 a3 6c b5 03 b6  |.8.......+..l...|
00000090  d8 ad bd ed 78 6c 92 60  fc bb 68 12 24 e0 07 47  |....xl.`..h.$..G|
000000a0  03 66 2a 94 d8 c6 0c ab  7e 88 bd 80 b8 fe a7 cb  |.f*.....~.......|
000000b0  e4 05 8f 29 70 b6 83 1c  30 56 29 7a a2 b3 b0 6a  |...)p...0V)z...j|
000000c0  8c 23 46 9c 1f aa 67 04  63 4a 20 0d e0 c2 03 e1  |.#F...g.cJ .....|
000000d0  22 7d 1b b4 ff 41 6a 99  46 32 43 d2 de 34 d6 ae  |"}...Aj.F2C..4..|
000000e0  f7 5b e7 4a 6a 6b a5 00  31 98 13 88 82 f8 93 c3  |.[.Jjk..1.......|
000000f0  71 71 58 b2 01 c9 cb 7f  55 70 39 6d 08 e4 21 e0  |qqX.....Up9m..!.|
00000100  4e 67 ee c8 bc 20 7c 66  34 92 c3 eb 8d f2 4b af  |Ng... |f4.....K.|
```



Guide: >=95% is compressed/encrypted-like; 70-95% is mixed; <70% is plaintext/structured-like.





Guide: >=95% is compressed/encrypted-like; 70-95% is mixed; <70% is plaintext/structured-like.

BOOTLOADER - PRINTER MANUFACTURER

- u-boot.srec - Motorola S-Record
- Convert to raw binary with **srec_cat** ("srec_cat u-boot.srec -Output u-boot.plain.bin -Binary")
- Get the load address (0x0)
- Get the architecture (AARCH64 little-endian)
- Load into Ghidra and:
  - Search for "TPCE" (ECPT)
  - Search for "decrypt"
  - Use FindCrypt Fto find AES S-Box

BOOTLOADER - PRINTER MANUFACTURER

Decompile: FUN_020891d0 - (u-boot.bin)

```
1
2 int FUN_020891d0(long param_1,undefined8 param_2,undefined8 param_3)
3
4 {
5   int iVar1;
6
7   iVar1 = FUN_0208ddcc(param_2);
8   if (iVar1 != 0) {
9     iVar1 = FUN_0208de00(param_2,*(undefined4 *)(param_1 + 0x28),param_2,param_3);
10    if (iVar1 == 0) {
11      FUN_020d7f74(s_Decrypt_success._020e6fab);
12      iVar1 = 0;
13    }
14    else {
15      iVar1 = 3;
16      FUN_020d7f74(s_%s(%d)_Decrypt_error_%s_020e6f92,s_decrypt_firmware_020dd9b0,0x2e5,
17                   *(undefined8 *)(param_1 + 0x20));
18    }
19  }
20  return iVar1;
21 }
22
```

BOOTLOADER - PRINTER MANUFACTURER

Decompile: do_decrypt - (u-boot.bin)

```
1
2 int do_decrypt(char *param_1,int param_2,undefined8 param_3,undefined8 param_4)
3
4 {
5   int iVar1;
6
7   iVar1 = is_ECPT(param_1);
8   if (iVar1 != 0) {
9     iVar1 = FUN_0208dd08(&DAT_020e3b28,0x20,&DAT_020e3b18,0x10,param_1 + 4,param_2 + -4,param_3,
10                         param_4);
11    return iVar1;
12  }
13  return -1;
14 }
15
```

BOOTLOADER - PRINTER MANUFACTURER

Decompile: do_decrypt - (u-boot.bin)

```
1
2 int do_decrypt(char *buffer,int param_2,undefined8 param_3,undefined8 param_4)
3
4 {
5   int iVar1;
6
7   iVar1 = is_ECPT(buffer);
8   if (iVar1 != 0) {
9     iVar1 = aes_cbc_256(AES_KEY,0x20,AES_IV,0x10,buffer + 4,param_2 + -4,param_3,param_4);
10    return iVar1;
11  }
12  return -1;
13 }
14
```

BOOTLOADER - PRINTER MANUFACTURER

- Manufacturers do make mistakes
- Always make sur you look at every artifact

BOOTLOADER - DAHUA

BOOTLOADER - DAHUA

- Initial coverage by at Offzone Moscow 2022
- Talk and slides in Russian - [*insert take on language barriers still creating islands of security research in 2026*]

BOOTLOADER - DAHUA

23:58 **Quentin** You may want to activate close caption translation for this one, but I think all the answers are there https://youtu.be/01uPmePiN0A

> **YouTube**
> Юрий Васин IP камера Dahua. Как посмотреть, где почесать
>
> 
>

23:58 **Arun Magesh** damn. russian. i think i have caption feature.

23:59 **Quentin** The reduction from SHA256 hash to 16 bytes, the different slots for different files, the final key calculation. It looks really close to what we got.

YouTube can auto-translate to english, with some funny mistakes from time to time

BOOTLOADER - DAHUA

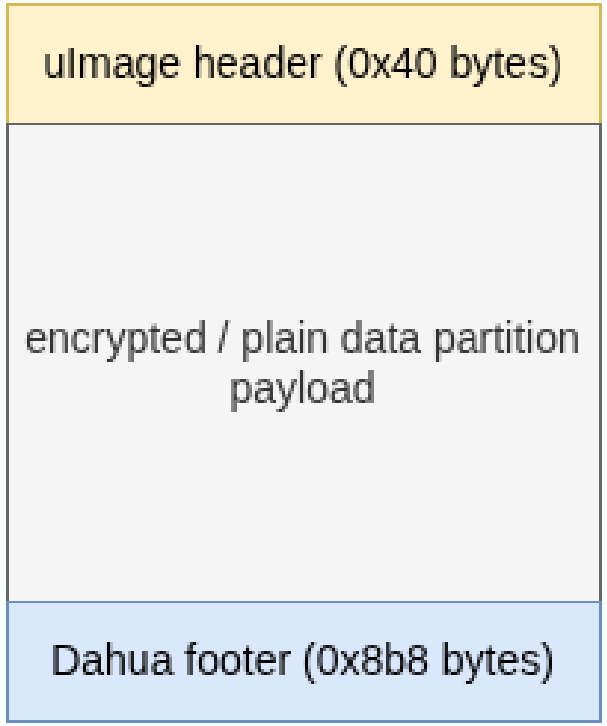

```
quentin@onekey:~ $ ls -alh
total 31M
drwxrwxr-x 2 quentin quentin 4,0K jun 22 14:47 .
drwxrwxr-x 3 quentin quentin 4,0K jun 22 11:50 ..
-rw-r--r-- 1 quentin quentin 8,5K aug  7  2024 check.img
-rw-r--r-- 1 quentin quentin 307K aug  7  2024 dhboot.bin.img
-rw-r--r-- 1 quentin quentin 7,6K aug  7  2024 hwid
-rwxr-xr-x 1 quentin quentin  322 aug  7  2024 Install
-rw-r--r-- 1 quentin quentin 2,5M aug  7  2024 kernel.img
-rw-r--r-- 1 quentin quentin 8,5K aug  7  2024 partition-x.cramfs.img
-rw-r--r-- 1 quentin quentin  21M aug  7  2024 romfs-x.squashfs.img
-rw-r--r-- 1 quentin quentin  128 aug  7  2024 sign.img
-rw-r--r-- 1 quentin quentin 7,8M aug  7  2024 web-x.squashfs.img
```

BOOTLOADER - DAHUA

```
quentin@onekey:~ $ file *

check.img:              u-boot legacy uImage, check, Linux/ARM, Firmware Image (Not compressed)
dhboot.bin.img:         u-boot legacy uImage, boot, Linux/ARM, Firmware Image (Not compressed)
hwid:                   JSON data
Install:                JSON data
kernel.img:             u-boot legacy uImage, kernel, Linux/ARM, Firmware Image (Not compressed)
partition-x.cramfs.img: u-boot legacy uImage, partition, Linux/ARM, Standalone Program (gzip)
romfs-x.squashfs.img:   u-boot legacy uImage, romfs, Linux/ARM, Standalone Program (gzip)
sign.img:               data
web-x.squashfs.img:     u-boot legacy uImage, web, Linux/ARM, Standalone Program (gzip)
```

BOOTLOADER - DAHUA

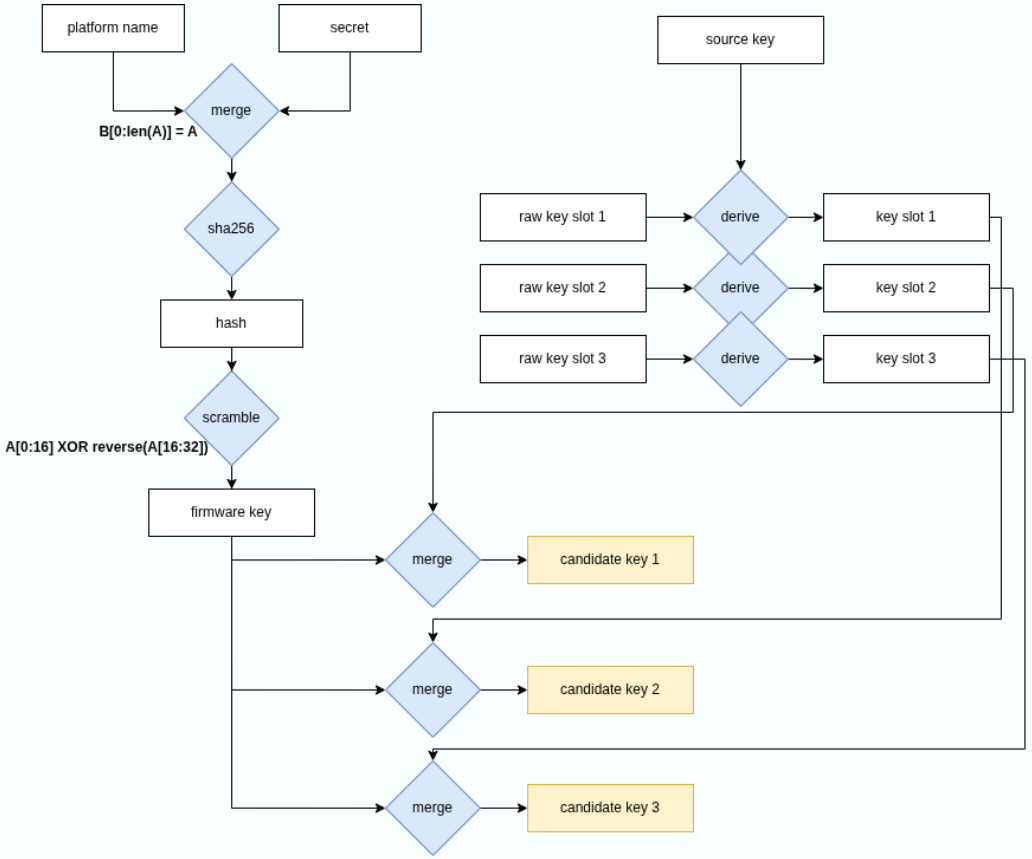


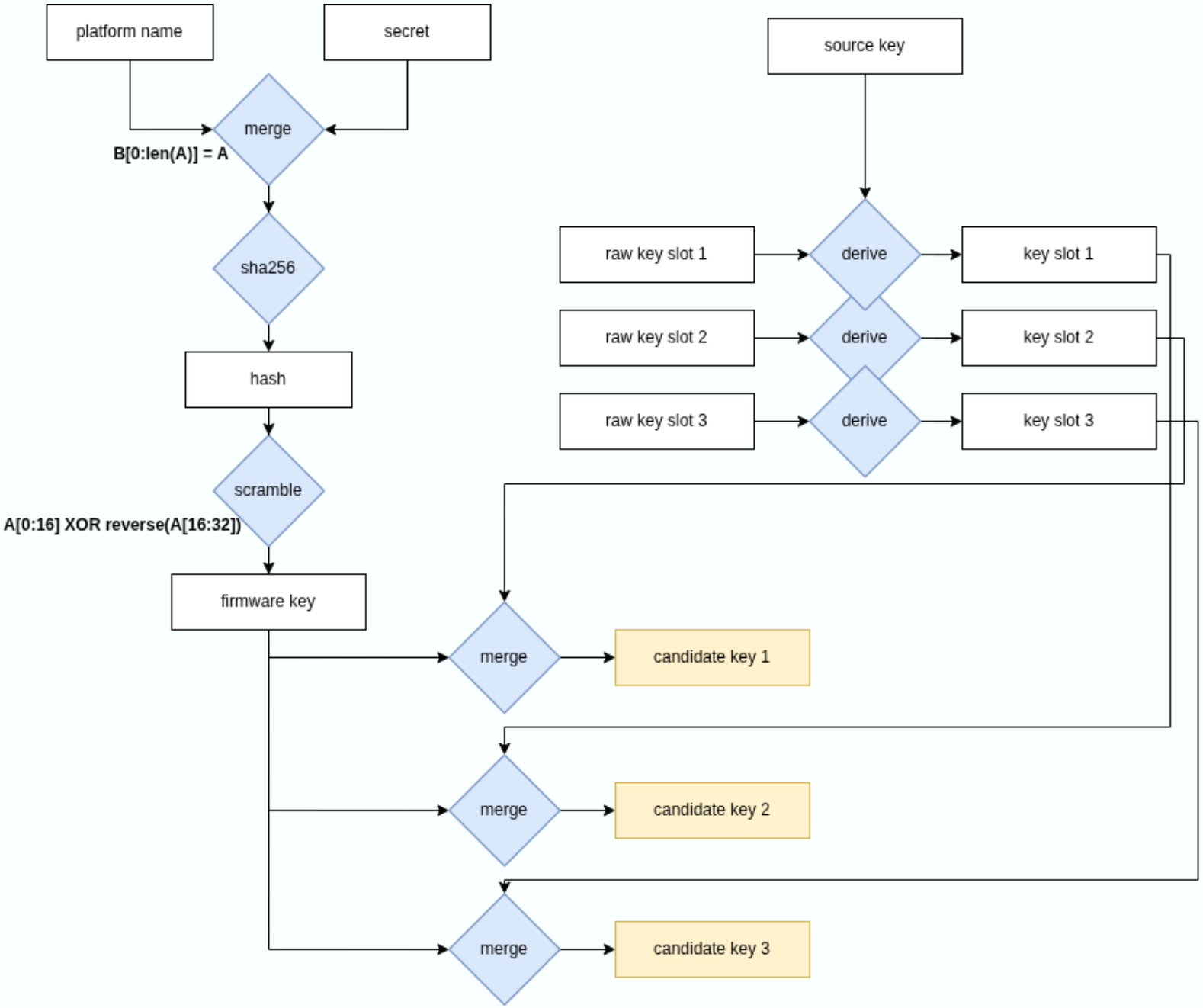

```
typedef struct secrity_footer {
    char magic[16];          // "SecrityImgMagic\0"
    char footer_version[8];
    uint32_t ih_magic;
    uint32_t ih_hcrc;
    uint32_t ih_time;
    uint32_t file_size;
    uint32_t ram_start_offset;
    uint32_t ram_end_offset;
    uint32_t ih_dcrc;
    uint8_t  ih_os;
    uint8_t  ih_arch;
    uint8_t  ih_type;
    uint8_t  ih_comp;
    char   ih_name[32];
} secrity_footer_t;
```

BOOTLOADER - DAHUA

- Encryption is AES-CBC-128, hard part is **key reconstruction**
- Firmware decryption stays the same, but:
- They change which key slot is used for kernel or filesystem over time
- They change how encryption is applied
  - All blocks
  - Even blocks
  - Every third block
- They change how IV is derived
- Manageable through proper heuristics

BOOTLOADER - DAHUA – DECRYPTION SCHEDULES




platform name
secret
merge
B[0:len(A)] = A
sha256
hash
scramble
A[0:16] XOR reverse(A[16:32])
firmware key
source key
raw key slot 1
derive
key slot 1
raw key slot 2
derive
key slot 2
raw key slot 3
derive
key slot 3
merge
candidate key 1
merge
candidate key 2
merge
candidate key 3

ONEKEY

BOOTLOADER - DAHUA

- Encryption is AES-CBC-128, hard part is **key reconstruction**
- Firmware decryption stays the same, but:
- They change which key slot is used for kernel or filesystem over time
- They change how encryption is applied
  - All blocks
  - Even blocks
  - Every third block
- They change how IV is derived
- Manageable through proper heuristics

BOOTLOADER - DAHUA – DECRYPTION SCHEDULES

Kernel Decryption Schedule

sector index	0	1	2	3	4	5	6
operation	AES	copy	AES	copy	AES	copy	AES
IV index	0		1		2		3

Filesystem Decryption Schedule (Legacy)

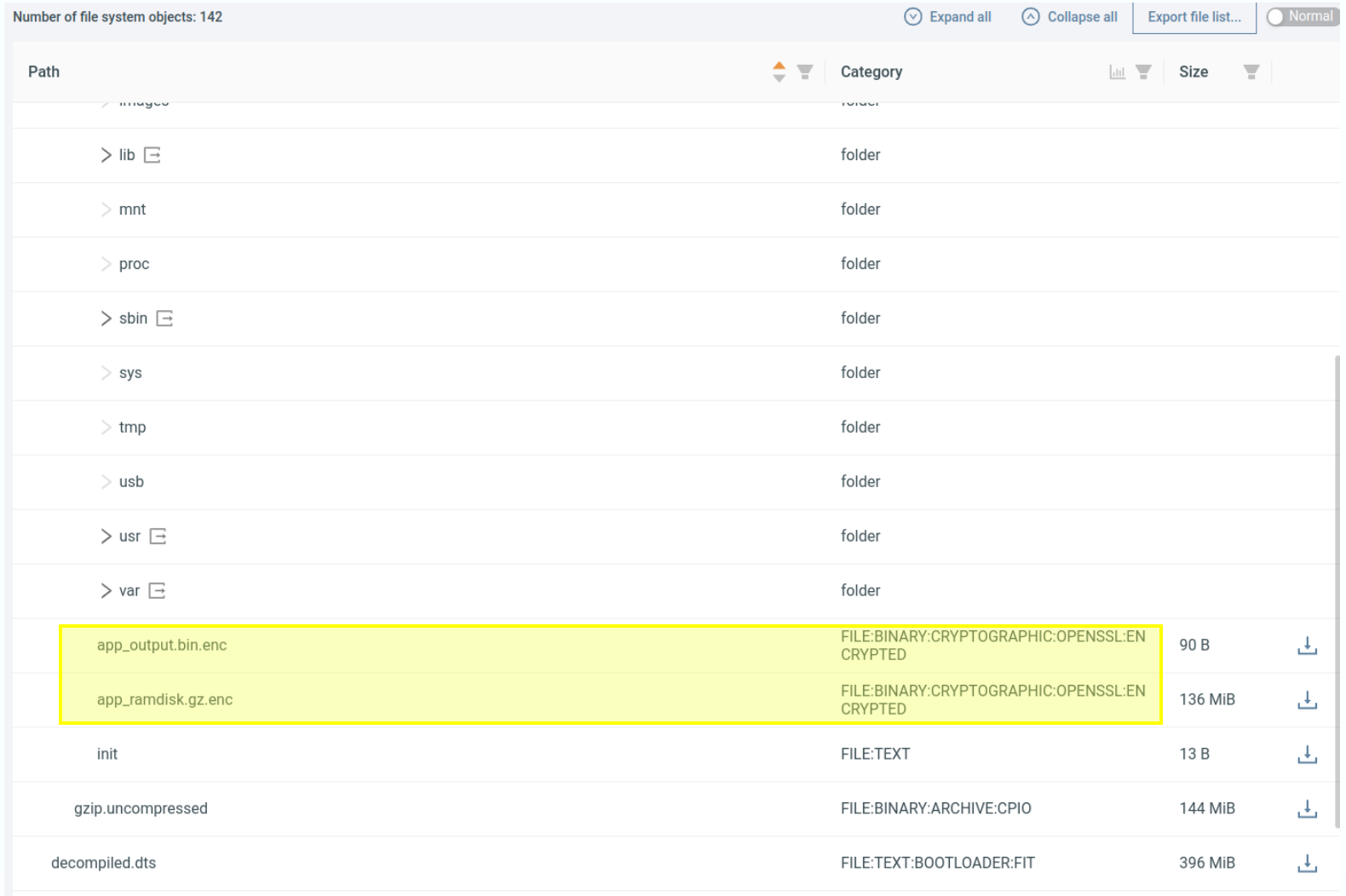
sector index	0	1	2	3	4	5	6
operation	AES	AES	AES	AES	AES	AES	AES
IV index	0	1	2	3	4	5	6

Filesystem Decryption Schedule (Current)

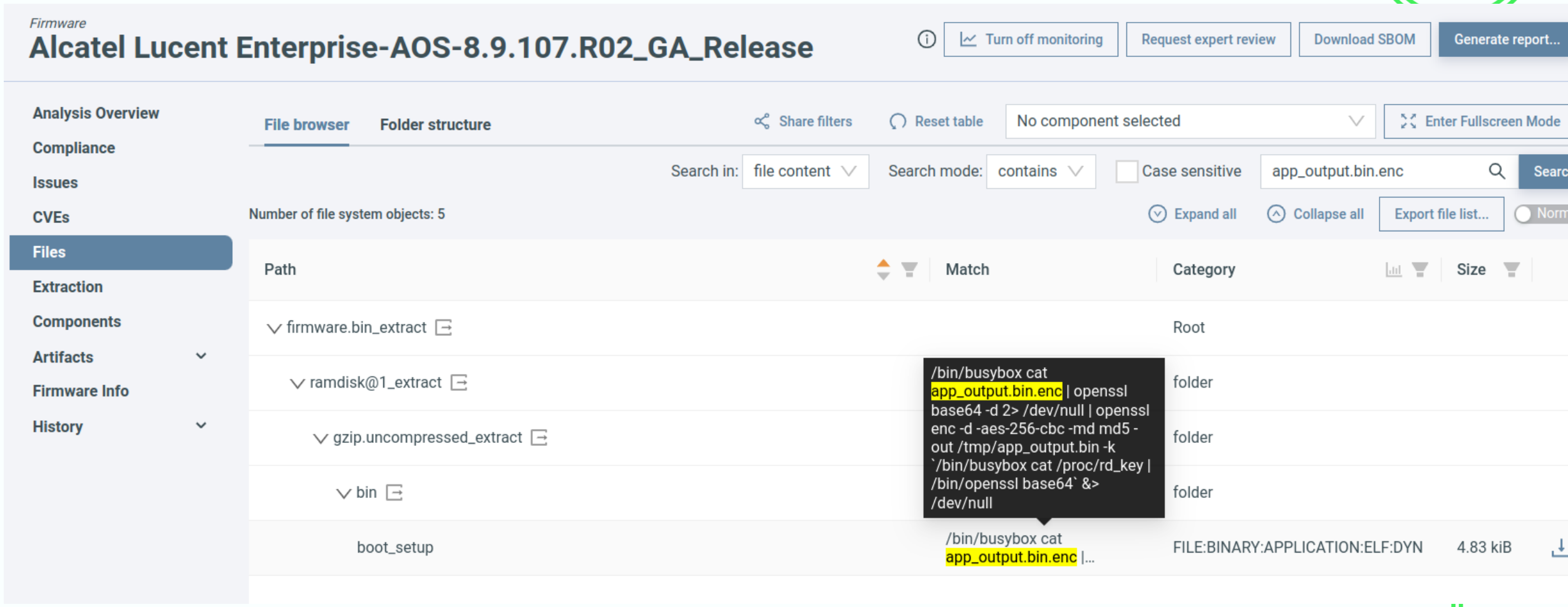
sector index	0	1	2	3	4	5	6
operation	AES	copy	copy	AES	copy	copy	AES
IV index	0			1			2

ENCRYPTED RAMDISK - ALCATEL-LUCENT

ENCRYPTED RAMDISK - ALCATEL-LUCENT



ENCRYPTED RAMDISK - ALCATEL-LUCENT



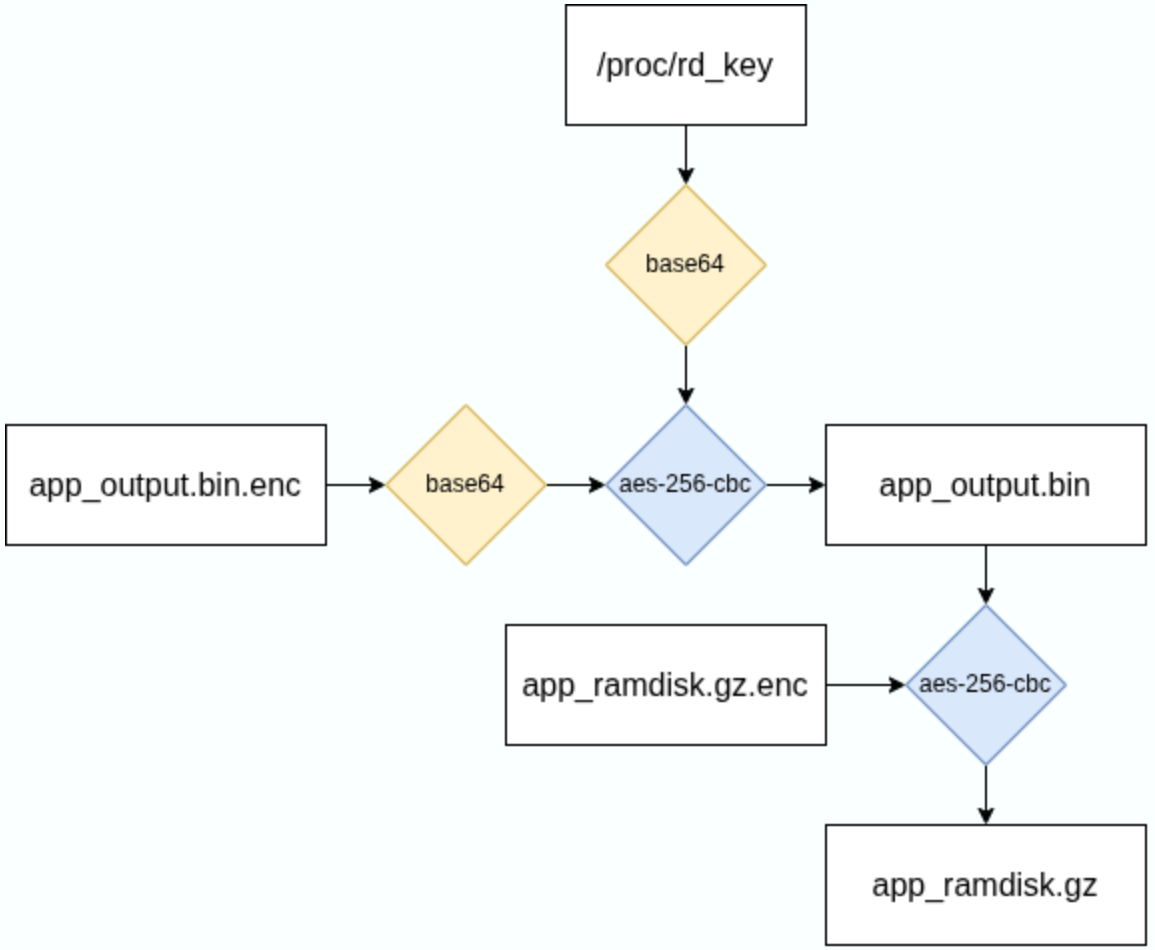
ENCRYPTED RAMDISK - ALCATEL-LUCENT

- Let's look at boot_setup !

```
/bin/busybox cat app_output.bin.enc | \
    openssl base64 -d 2> /dev/null | \
    openssl enc -d -aes-256-cbc -md md5 -out /tmp/app_output.bin \
    -k `/bin/busybox cat /proc/rd_key | /bin/openssl base64`  &> /dev/null

/bin/openssl enc -aes-256-cbc -d -md md5 \
    -in /app_ramdisk.gz.enc \
    -out /app_ramdisk.gz \
    -k `/bin/busybox cat /tmp/app_output.bin`  &> /dev/null
```

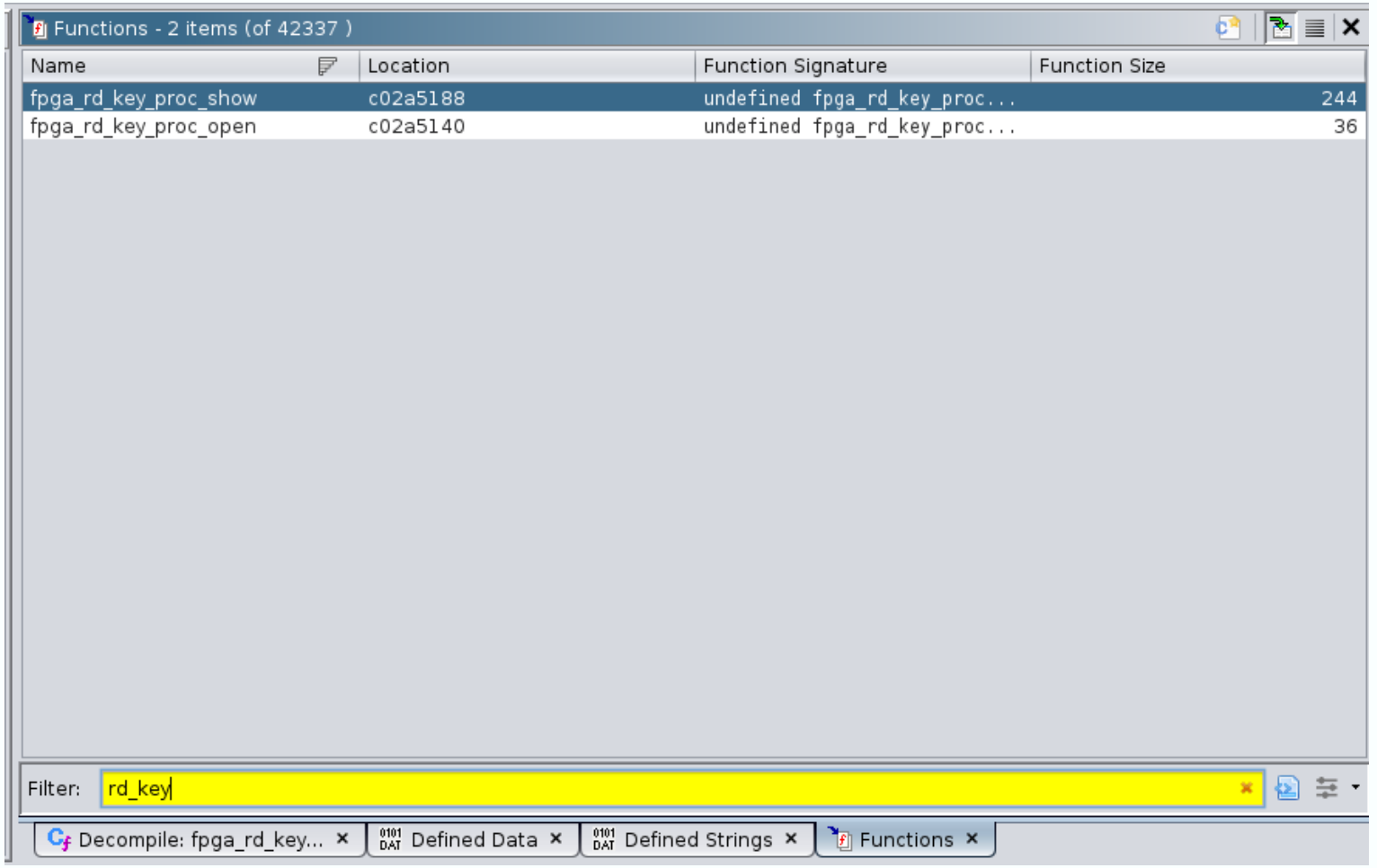
ENCRYPTED RAMDISK - ALCATEL-LUCENT



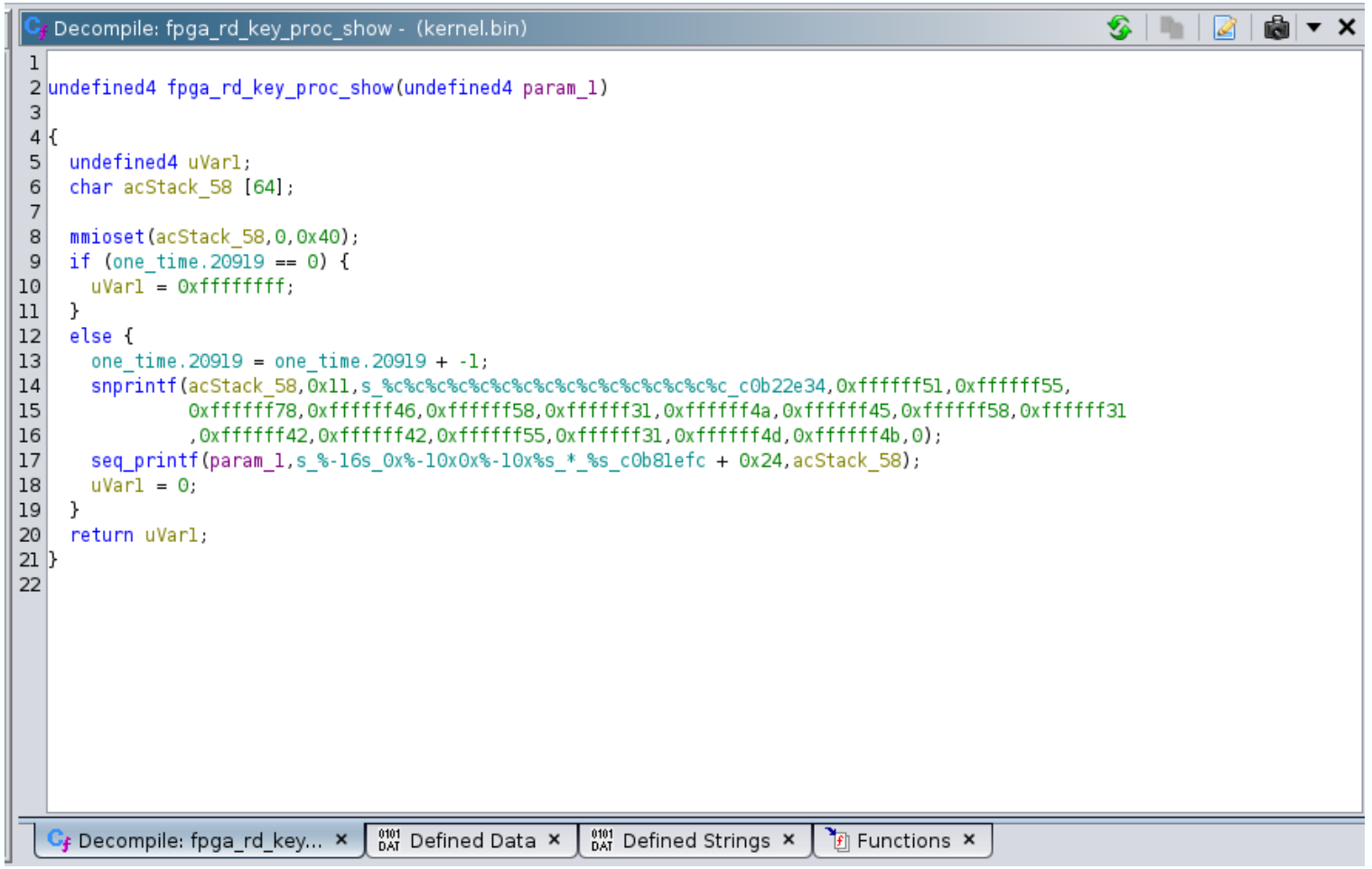
ENCRYPTED RAMDISK - ALCATEL-LUCENT


```
quentin@onekey:~ $ vmlinux-to-elf kernel@1 kernel.elf
[+] Kernel successfully decompressed in-memory (the offsets that follow will be given relative to the decompressed binary)
[+] Version string: Linux version 4.14.222 (relman@mcgrath) (gcc version 4.8.2 (GCC)) #1 SMP Thu Mar 16 11:35:58 PDT 2023
[+] Guessed architecture: armle successfully in 1.68 seconds
[+] Found kallsyms_token_table at file offset 0x00ba92f0
[+] Found kallsyms_token_index at file offset 0x00ba9620
[+] Found kallsyms_markers at file offset 0x00ba9030
[+] Found kallsyms_names at file offset 0x00b2ba10
[+] Found kallsyms_num_syms at file offset 0x00b2ba00
[i] Negative offsets overall: 0 %
[i] Null addresses overall: 0 %
[+] Found kallsyms_offsets at file offset 0x00b00520
[+] Successfully wrote the new ELF kernel to kernel.elf
```

ENCRYPTED RAMDISK - ALCATEL-LUCENT



ENCRYPTED RAMDISK - ALCATEL-LUCENT



ENCRYPTED RAMDISK - ALCATEL-LUCENT

```
content = [
    0xffffff51,
    0xffffff55,
    0xffffff78,
    0xffffff46,
    0xffffff58,
    0xffffff31,
    0xffffff4a,
    0xffffff45,
    0xffffff58,
    0xffffff31,
    0xffffff42,
    0xffffff42,
    0xffffff55,
    0xffffff31,
    0xffffff4d,
    0xffffff4b
]
content = [c & 0xff for c in content]
# [81, 85, 120, 70, 88, 49, 74, 69, 88, 49, 66, 66, 85, 49, 77, 75]
bytes(content)
# b'QUxFX1JEX1BBU1MK'
base64.b64decode(bytes(content))
# b'ALE_RD_PASS\n'
```

BINARY - STORMSHIELD

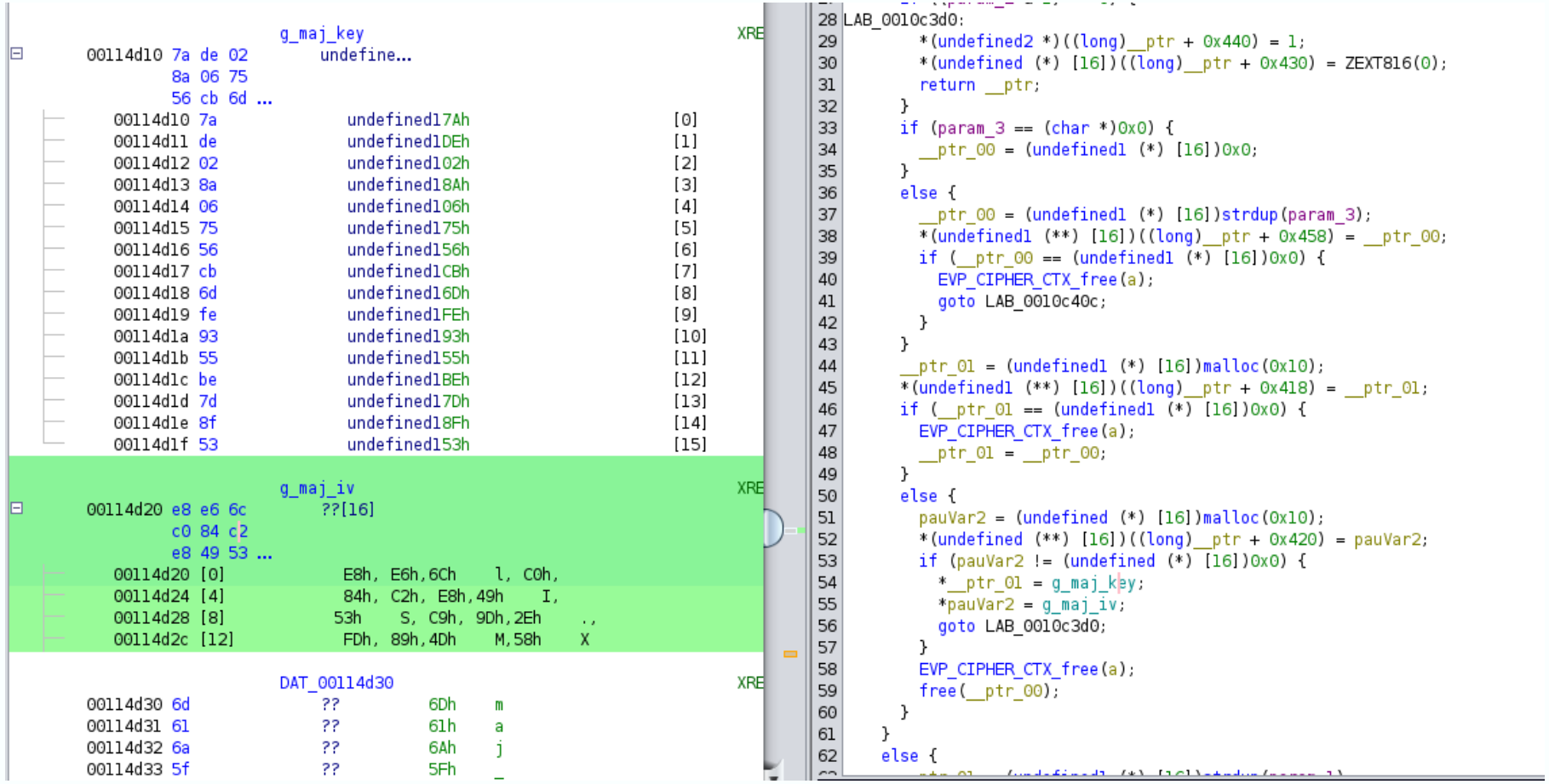
BINARY - STORMSHIELD

- StormShield is a french manufacturer of firewalls. Like any good firewalls, stormshield runs on FreeBSD.
- Firmware are encrypted, but Stormshield provides virtual machine images. Or at least did provide, I found one on Internet Archive (https://archive.org/details/MyStormshield)
- Reversing the decryption has been our secret challenge for internship candidates. A bit sad to release it :)
- Right balance of easy to find information and PITA analysis (aka setting up FreeBSD)

BINARY - STORMSHIELD

- After a quick look around, you quickly realize decryption is performed by a binary named **fwupdate**. This binary calls *fwcrypto_decverif_maj_file* from **libncrypto.so**, some kind of crypto wrapper library developed by Stormshield.
- Within *fwcrypto_upd_init*, an EVP_CIPHER_CTX object is built

BINARY - STORMSHIELD



```
                        g_maj_key                                       XRE
00114d10 7a de 02       undefine...
         8a 06 75
         56 cb 6d ...
   00114d10 7a          undefined17Ah                   [0]
   00114d11 de          undefined1DEh                   [1]
   00114d12 02          undefined102h                   [2]
   00114d13 8a          undefined18Ah                   [3]
   00114d14 06          undefined106h                   [4]
   00114d15 75          undefined175h                   [5]
   00114d16 56          undefined156h                   [6]
   00114d17 cb          undefined1CBh                   [7]
   00114d18 6d          undefined16Dh                   [8]
   00114d19 fe          undefined1FEh                   [9]
   00114d1a 93          undefined193h                   [10]
   00114d1b 55          undefined155h                   [11]
   00114d1c be          undefined1BEh                   [12]
   00114d1d 7d          undefined17Dh                   [13]
   00114d1e 8f          undefined18Fh                   [14]
   00114d1f 53          undefined153h                   [15]

                        g_maj_iv                                        XRE
00114d20 e8 e6 6c       ??[16]
         c0 84 c2
         e8 49 53 ...
   00114d20 [0]           E8h, E6h,6Ch    l, C0h,
   00114d24 [4]           84h, C2h, E8h,49h    I,
   00114d28 [8]          53h    S, C9h, 9Dh,2Eh    .,
   00114d2c [12]          FDh, 89h,4Dh    M,58h    X

                        DAT_00114d30                                    XRE
00114d30 6d             ??          6Dh    m
00114d31 61             ??          61h    a
00114d32 6a             ??          6Ah    j
00114d33 5f             ??          5Fh    _
```

```
LAB_0010c3d0:
        *(undefined2 *)((long)__ptr + 0x440) = 1;
        *(undefined (*) [16])((long)__ptr + 0x430) = ZEXT816(0);
        return __ptr;
      }
      if (param_3 == (char *)0x0) {
        __ptr_00 = (undefined1 (*) [16])0x0;
      }
      else {
        __ptr_00 = (undefined1 (*) [16])strdup(param_3);
        *(undefined1 (**) [16])((long)__ptr + 0x458) = __ptr_00;
        if (__ptr_00 == (undefined1 (*) [16])0x0) {
          EVP_CIPHER_CTX_free(a);
          goto LAB_0010c40c;
        }
      }
      __ptr_01 = (undefined1 (*) [16])malloc(0x10);
      *(undefined1 (**) [16])((long)__ptr + 0x418) = __ptr_01;
      if (__ptr_01 == (undefined1 (*) [16])0x0) {
        EVP_CIPHER_CTX_free(a);
        __ptr_01 = __ptr_00;
      }
      else {
        pauVar2 = (undefined (*) [16])malloc(0x10);
        *(undefined (**) [16])((long)__ptr + 0x420) = pauVar2;
        if (pauVar2 != (undefined (*) [16])0x0) {
          *__ptr_01 = g_maj_key;
          *pauVar2 = g_maj_iv;
          goto LAB_0010c3d0;
        }
        EVP_CIPHER_CTX_free(a);
        free(__ptr_00);
      }
    }
    else {
```

BINARY - STORMSHIELD

- Dynamic analysis on BSD ? Dtrace

```
#!/usr/sbin/dtrace -s
pid$target::EVP_CipherInit:entry {
    this->key = (char*)copyin(arg2, 16);
    this->iv = (char*)copyin(arg3, 16);
    printf("ctx: %p\n", arg1);
    printf("type: %p\n", arg2);
    printf(
        "key: %02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x\n",
        this->key[0], this->key[1], this->key[2], this->key[3],
        this->key[4], this->key[5], this->key[6], this->key[7],
        this->key[8], this->key[9], this->key[10], this->key[11],
        this->key[12], this->key[13], this->key[14], this->key[15]);
    printf(
        "iv: %02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x\n",
        this->iv[0], this->iv[1], this->iv[2], this->iv[3],
        this->iv[4], this->iv[5], this->iv[6], this->iv[7],
        this->iv[8], this->iv[9], this->iv[10], this->iv[11],
        this->iv[12], this->iv[13], this->iv[14], this->iv[15]
    );
}
pid$target::EVP_DecryptUpdate:entry {
    this->ctx = arg1;
    this->out = stringof(copyin(arg2, 256));
    this->in = stringof(copyin(arg3, 256));
    printf("in: %02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x\n", this->in
7], this->in[8], this->in[9], this->in[10], this->in[11], this->in[12], this->in[13], this->i
    printf("out: %02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x\n", this->o
his->out[7], this->out[8], this->out[9], this->out[10], this->out[11], this->out[12], this->o
}
syscall::openat:entry {
    printf("fd: %s", stringof(copyin(arg1, 32)));
}
syscall::read:entry {
    printf("%d %d %d\n", arg1, arg2, arg3);
}
```

BINARY - STORMSHIELD


```
root@freebsd:~ # dtrace -s test.d -p `pgrep fwupdate` | more
dtrace: script 'test.d' matched 10 probes
CPU     ID                    FUNCTION:NAME
--snip--
  1  72036                     openat:entry fd: fwupd-4.6.9-SNS-amd64-M.maj
  1  71062                       read:entry 21600800795584 32768 0
  1  71062                       read:entry 21600800795584 32768 0
  3  73252            EVP_aes_128_cbc:entry
  3  73250             EVP_CipherInit:entry ctx: 1e6af84b9c40
type: 1e6b15e38060
key: 7ade028a067556cb6dfe9355be7d8f53
iv: e8e66cc084c2e84953c99d2efd894d58
  3  73251          EVP_DecryptUpdate:entry in: 5daa54e40c7a1e30ec5d2c8f00000000
out: 00000000000000000000000000000000
```

BINARY - STORMSHIELD

```
binwalk -R '\x5d\xaa\x54\xe4' fwupd-4.6.9-SNS-amd64-M.maj
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
6556          0x199C          Raw signature (\x5d\xaa\x54\xe4)
```

BINARY - STORMSHIELD

```
dd if=fwupd-4.6.9-SNS-amd64-M.maj bs=6556 skip=1 |\
openssl enc -d -aes-128-cbc -iv e8e66cc084c2e84953c99d2efd894d58 \
-K 7ade028a067556cb6dfe9355be7d8f53 -nosalt | more
#!/bin/sh
#

FWPATH="/usr/Firewall"
FWBIN="${FWPATH}/sbin"
UPDLOG="${FWPATH}/System/update"
LOGFILE="${FWPATH}/System/update.log"
```

YEALINK


Yealink
Kevin
15:32
Fri, Jan 25
Peter
Adrian
Carlos
Louis
Daniel
History
Directory
DND
Menu
HD

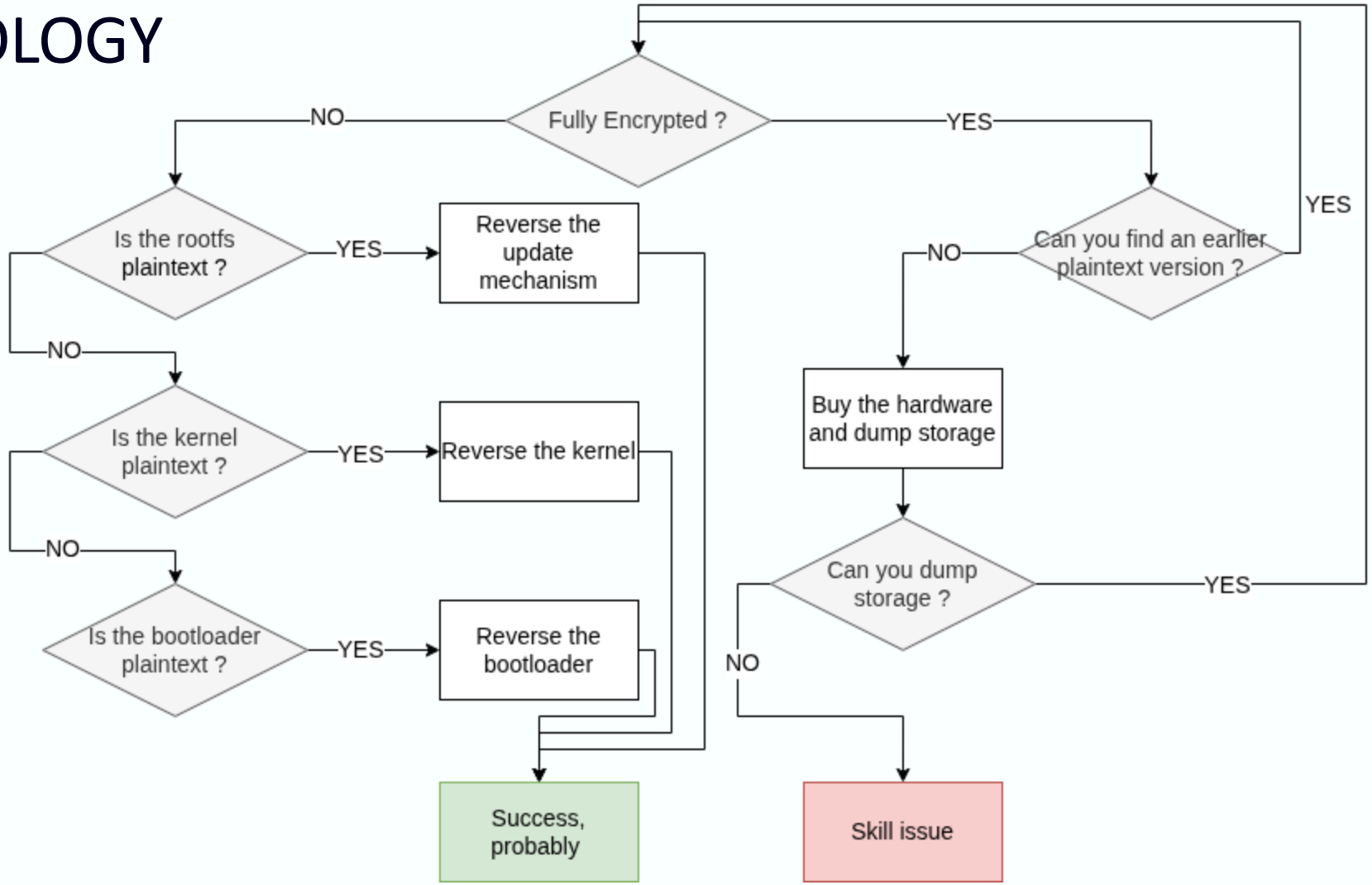
YEALINK

- Initially investigated by Synacktiv in 2019
- Since then, they continued experimenting
  - Different crypto schemes
  - Different file systems formats
  - Sometimes with hardware backed key derivation (but not really TPM)
- We had to keep track of this, and we know we're not the only ones :)
- Transparent decryption by the VFS layer in Linux kernel
- Made to mess with reversers, not protect (customer) data
  - UBIFS implementation only encrypts inode metadata, not content

YEALINK

Product Line	Version	Format
Yealink-T5x	96.86.199.10	Encrypted UBIFS v2 (AES ECB)
Yealink-W60	77.85.199.11	Encrypted with AES-CBC + TPM
Yealink-W70	146.85.199.5	Encrypted UBIFS v2 (AES ECB)
Yealink-W90	130.85.199.8	Encrypted UBIFS v2 (AES ECB)
Yealink-CP860	37.80.199.7	Encrypted YAFFS (cypher3)
Yealink-CP925	148.86.199.1	Encrypted UBIFS v2 (AES ECB)
Yealink-T46G	28.82.199.13	Encrypted YAFFS (cypher3)
Yealink-T48G	35.82.199.12	Encrypted YAFFS (cypher3)
Yealink-T42G	29.82.199.12	Encrypted SQUASHFS (cypher3)
Yealink-T4xS	66.86.199.2	Encrypted YAFFS (cypher3)

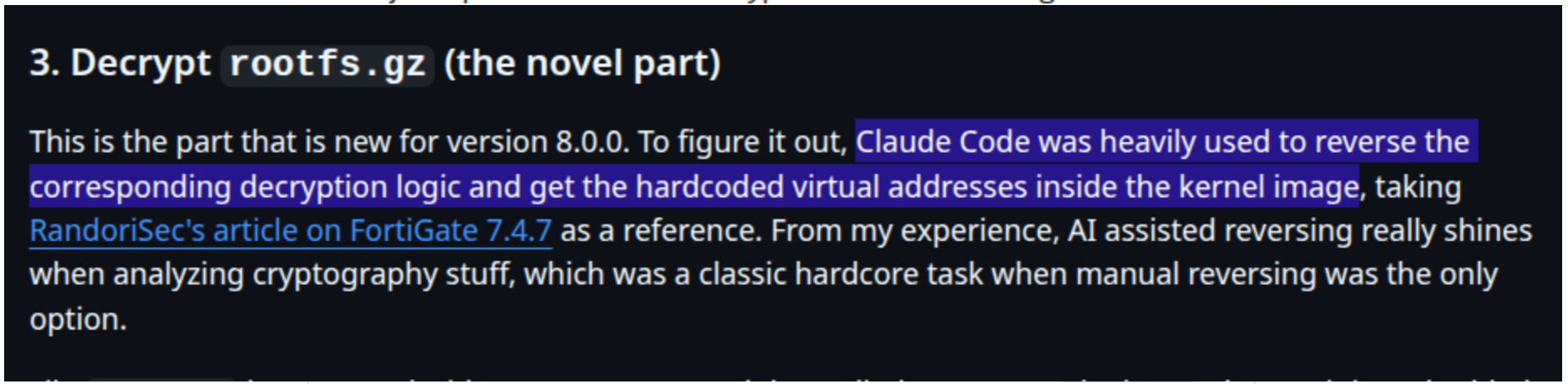
METHODOLOGY



ADVICE TO MANUFACTURERS

- Stop spending precious engineering time trying to encrypt and obfuscate.
- With current tooling - including AI frontier model – custom baked encryption can be reversed with little efforts.
- Either do it right, or don't.

ADVICE TO MANUFACTURERS



Victor FreskO
@hacefresko

I have just uploaded a tool to decrypt FortiOS v8.0.0 images. It works

Decrypt and extract FortiOS 8.0.0 firmware images.

0 Contributors · 0 Issues · 0 Stars · 0 Forks

GitHub - hacefresko/forticrack_v8: Decrypt and extract FortiOS 8.0.0 firmware images.

From github.com

6:46 PM · Jun 25, 2026 · **10K** Views

ADVICE TO MANUFACTURERS

- **If a vacuum cleaner (Ecovac) or a loudspeaker (Sonos) can implement encryption with eFuses, so can you !**
- fscrypt / LUKS for encryption
- dmverity for integrity
- eFuses or TPM for key storage / key derivation

ADVICE TO MANUFACTURERS (CAVEAT)

- TPM is hard to get right
  - Lack of parameter encryption leaves it open to hardware sniffing
  - Just using RSA primitives instead of key sealing leaves it open as decryption oracle
  - Not implementing measured boot / attestation means we can talk to the TPM anyway
- Make sure the whole Secure Boot chain is protected.
- Make sure your BL31 is bl4sty-resistant™

WORKSHOP TOMORROW !

- **Hands-on Firmware Extraction, Exploration, and Emulation**
- 2026-07-01 - 14:15–17:00 @ Room LW112

If you signed up: having a Debian derivative virtual machine at hand will speed things **a lot.** Use your hotel network to get your ISOs :)

REFERENCES

- Ghidra FindCrypt - https://github.com/TorgoTorgo/ghidra-findcrypt
- "No grave but the SIP": Reversing a VoIP phone firmware - Synactkiv - https://www.synacktiv.com/en/publications/no-grave-but-the-sip-reversing-a-voip-phone-firmware
- WHY 2025 - Die Hardcoded: Unlocking Yealink's (weakest) secrets - https://youtu.be/hIcnTBa8IEk
- Юрий Васин IP камера Dahua. Как посмотреть, где почесать - https://www.youtube.com/watch?v=01uPmePiN0A

Q&A.

Incendiary emails can be sent out to **research@onekey.com**