SYNACKTIV

Juicing Up the Autel EV Charger: Insights from Pwn2Own Automotive

STHack

May 24th 2024

Who are we

Vincent Fargues

@Karion_

Aymeric Palhière

@bak_sec

David Berard

@_p0ly_

Thomas Imbert

@masthoon

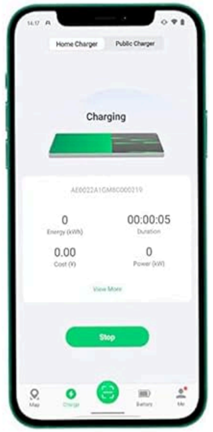
Synacktiv

- Offensive security
- 170 Experts
- Pentest, Reverse Engineering, Development, Incident Response

Reverse Engineering team

- 50 reversers
- Low level research, reverse engineering, vulnerability research, exploit development, etc.

Introduction



- `Autel MaxiCharger AC Wallbox Commercial`
- Electric Vehicle charger
- Monitoring & Management
  - Bluetooth (Low Energy)
  - WIFI
  - Ethernet
- Pwn2Own Tokyo 2024
  - Remote code execution
  - $60,000

Target

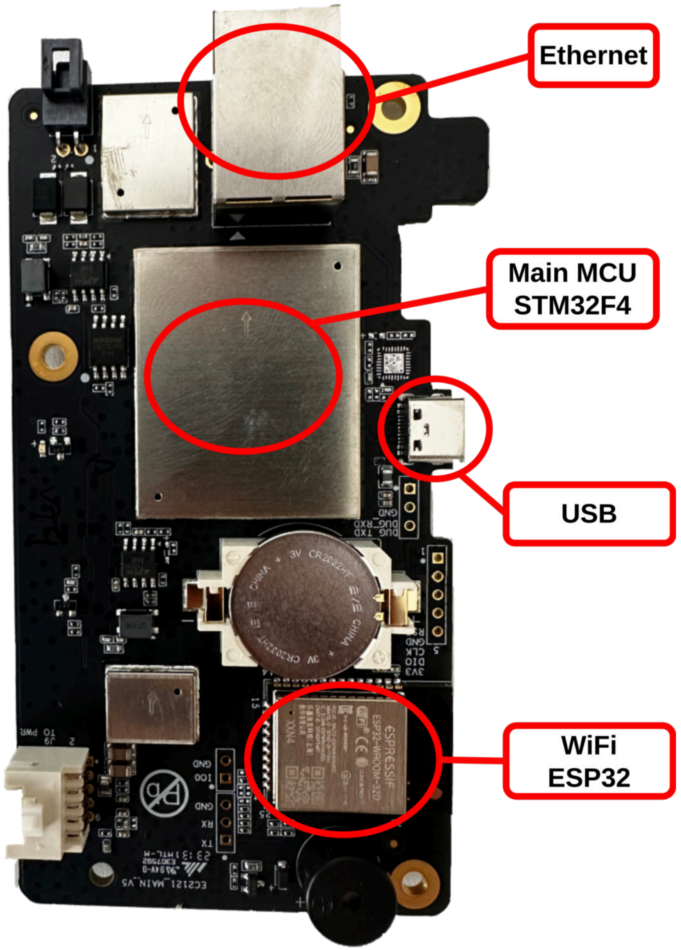
ⓘ Difficulties

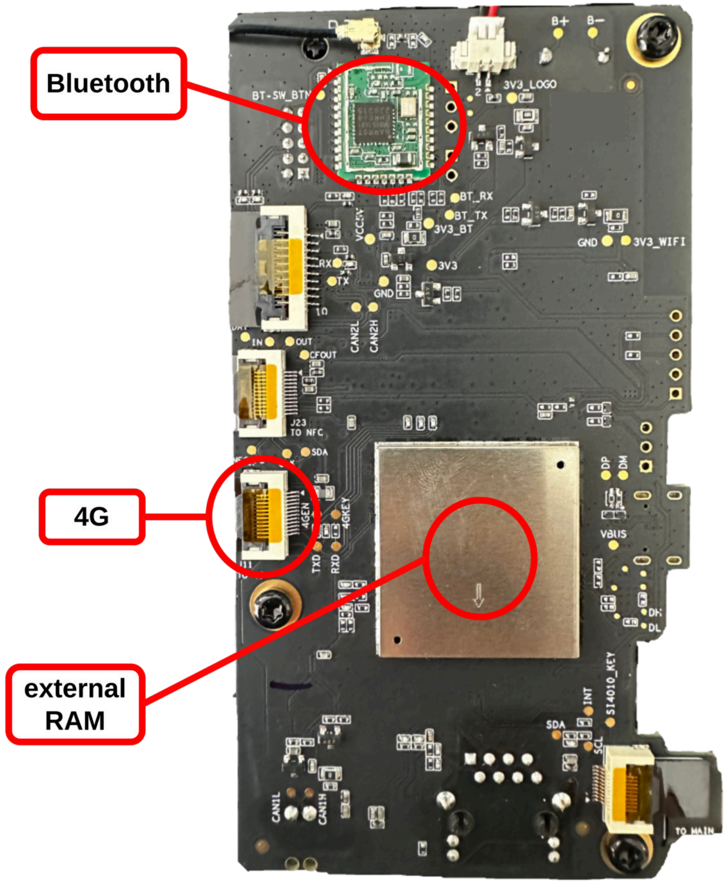
- Exact target cannot be purchased in Europe
  - Buy the European version → same PCB → different software 😒
  - We asked someone really nice in Canada to buy it and send it to us
- No public firmware available
- Packed APK
- Good HTTPs certificate validation

ⓘ Solution

- Go for hardware or software magic

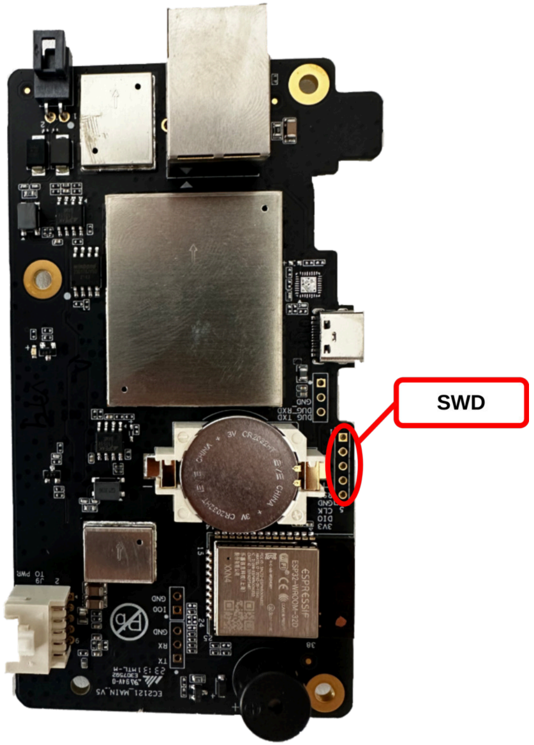
Hardware


Ethernet
Main MCU STM32F4
USB
WiFi ESP32


Bluetooth
4G
external RAM

Hardware

STM32 - JTAG / SWD



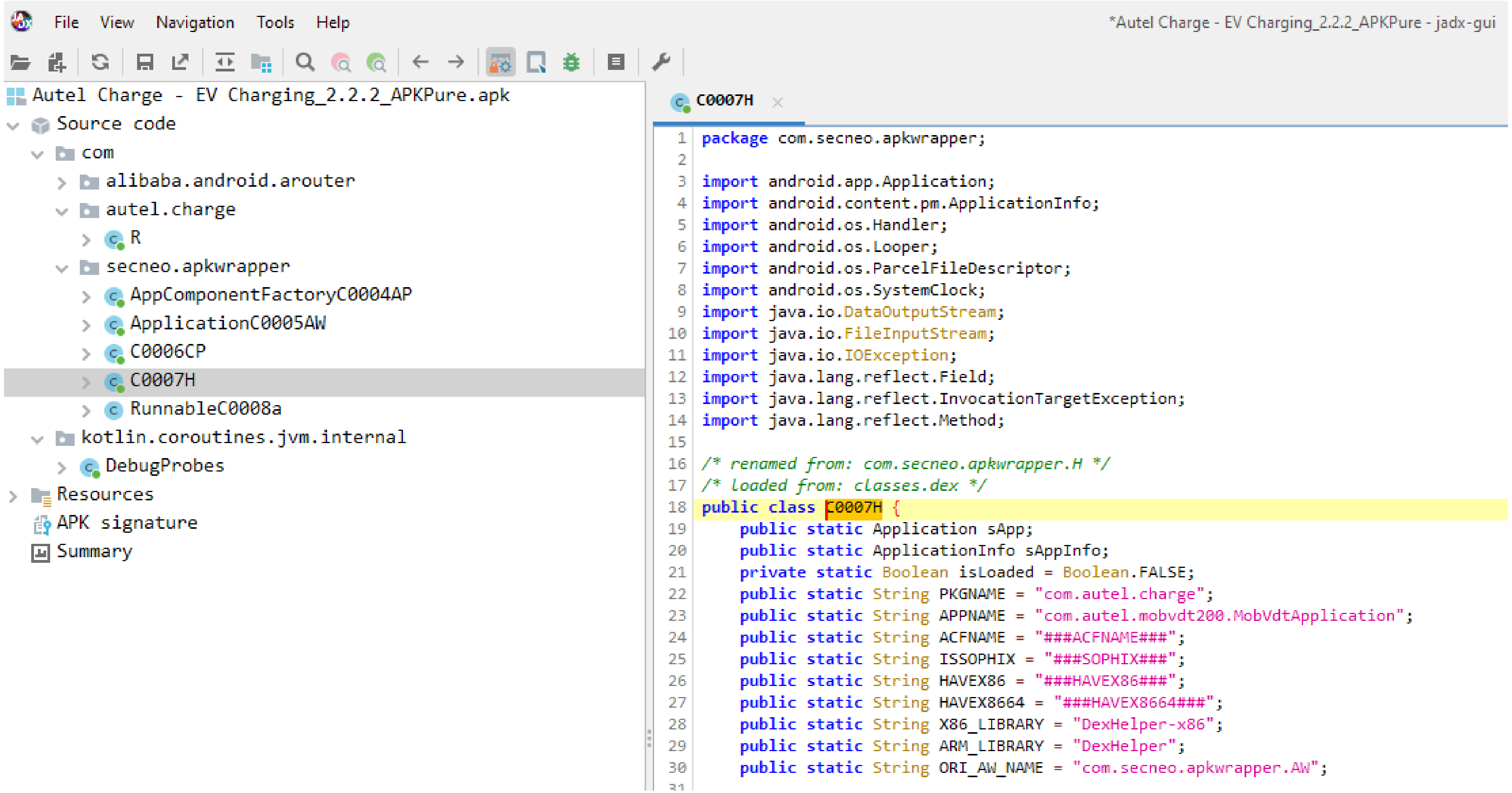
- SWD enabled
- ReaDout Protection Level 1
  - Can only read RAM memory
  - Flash is protected → can't read firmware
  - No debug
  - Can be downgraded but flash would be erased
- Hardware attack can be an option but would take time → back to software

Android application

Autel Charge

Android application

Autel Charge



```
Autel Charge - EV Charging_2.2.2_APKPure.apk
  Source code
    com
      alibaba.android.arouter
      autel.charge
        R
      secneo.apkwrapper
        AppComponentFactoryC0004AP
        ApplicationC0005AW
        C0006CP
        C0007H
        RunnableC0008a
    kotlin.coroutines.jvm.internal
      DebugProbes
  Resources
  APK signature
  Summary
```

```java
package com.secneo.apkwrapper;

import android.app.Application;
import android.content.pm.ApplicationInfo;
import android.os.Handler;
import android.os.Looper;
import android.os.ParcelFileDescriptor;
import android.os.SystemClock;
import java.io.DataOutputStream;
import java.io.FileInputStream;
import java.io.IOException;
import java.lang.reflect.Field;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;

/* renamed from: com.secneo.apkwrapper.H */
/* loaded from: classes.dex */
public class C0007H {
    public static Application sApp;
    public static ApplicationInfo sAppInfo;
    private static Boolean isLoaded = Boolean.FALSE;
    public static String PKGNAME = "com.autel.charge";
    public static String APPNAME = "com.autel.mobvdt200.MobVdtApplication";
    public static String ACFNAME = "###ACFNAME###";
    public static String ISSOPHIX = "###SOPHIX###";
    public static String HAVEX86 = "###HAVEX86###";
    public static String HAVEX8664 = "###HAVEX8664###";
    public static String X86_LIBRARY = "DexHelper-x86";
    public static String ARM_LIBRARY = "DexHelper";
    public static String ORI_AW_NAME = "com.secneo.apkwrapper.AW";
```

Android application

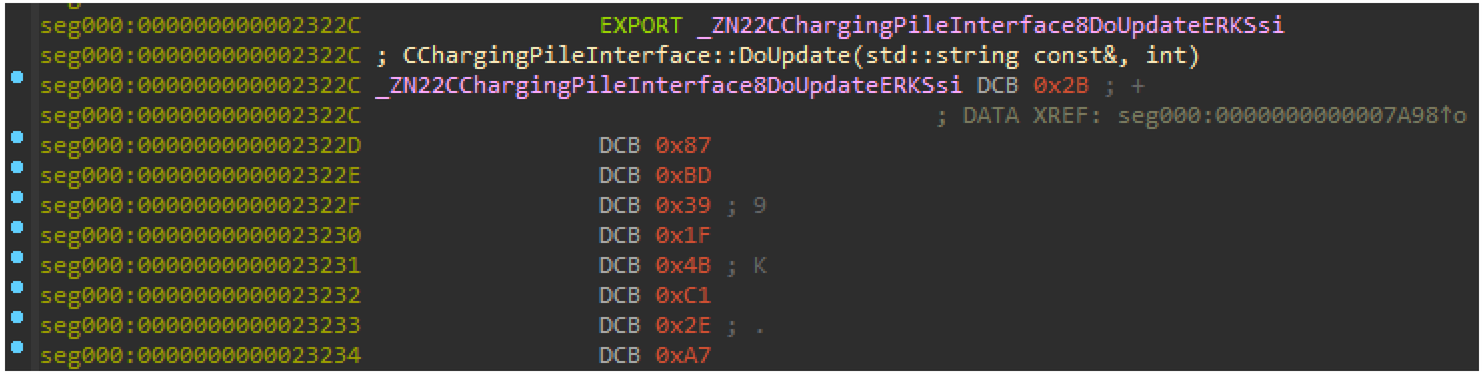
APK DEX Packer

- `Secneo apkwrapper` packer
- `DexHelper` JNI implements the packer logic
- Original application code stored encrypted

Android application

JNI library `libcp_bluetooth.so` (1/3)

- First looked at APK version 1.3
- Library Exports contain `GetOtaStatus` & `DoUpdate`
- Encrypted using `Bangcle`

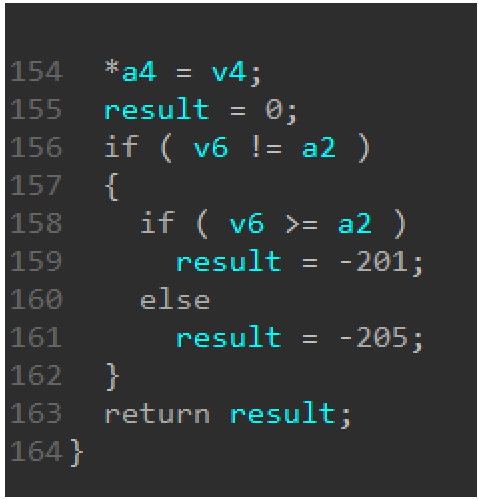


```
seg000:000000000002322C                     EXPORT _ZN22CChargingPileInterface8DoUpdateERKSsi
seg000:000000000002322C ; CChargingPileInterface::DoUpdate(std::string const&, int)
seg000:000000000002322C _ZN22CChargingPileInterface8DoUpdateERKSsi DCB 0x2B ; +
seg000:000000000002322C                                                      ; DATA XREF: seg000:0000000000007A98↑o
seg000:000000000002322D                     DCB 0x87
seg000:000000000002322E                     DCB 0xBD
seg000:000000000002322F                     DCB 0x39 ; 9
seg000:0000000000023230                     DCB 0x1F
seg000:0000000000023231                     DCB 0x4B ; K
seg000:0000000000023232                     DCB 0xC1
seg000:0000000000023233                     DCB 0x2E ; .
seg000:0000000000023234                     DCB 0xA7
```

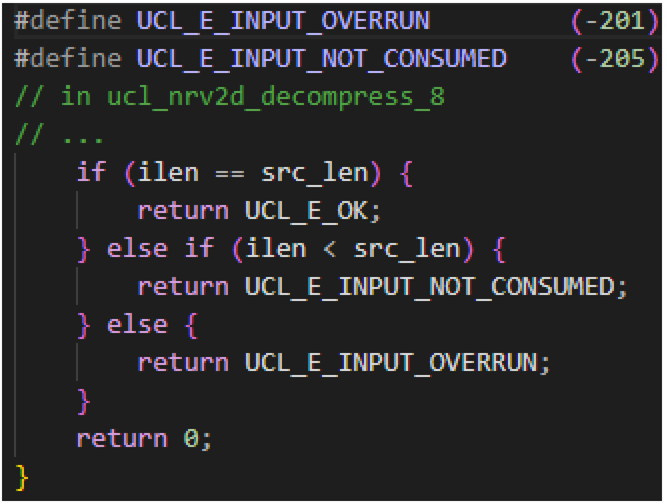
Android application

JNI library `libcp_bluetooth.so` (2/3)

- Code decompression library identified using magic return values: `UCL`



```
  *a4 = v4;
  result = 0;
  if ( v6 != a2 )
  {
    if ( v6 >= a2 )
      result = -201;
    else
      result = -205;
  }
  return result;
}
```



```
#define UCL_E_INPUT_OVERRUN          (-201)
#define UCL_E_INPUT_NOT_CONSUMED     (-205)
// in ucl_nrv2d_decompress_8
// ...
    if (ilen == src_len) {
        return UCL_E_OK;
    } else if (ilen < src_len) {
        return UCL_E_INPUT_NOT_CONSUMED;
    } else {
        return UCL_E_INPUT_OVERRUN;
    }
    return 0;
}
```

- Scan binary for magic `__b_a_n_g_c_l_e__check1234567_`
- Decompress the code using UCL (`nrv2d_decompress`)
  - Lossless data compression library
  - Released in 2000

Android application

JNI library `libcp_bluetooth.so` (3/3)

- Looked at the latest APK (`version 2.2`)
- **No obfuscation on this version...**
- Don't forget to check other versions before further analysis
- `CChargingPileInterface::DoUpdate`
  - Reads the firmware from the phone filesystem (but our device is up to date)
  - BLE command to initialize the update process
  - Transmission of the encrypted firmware by blocks of 0x200 bytes

Android application

«Unpacking»

Packer runtime protection

- `Secneo apkwrapper` prevents Frida from being present / injected
- Injected libs are allowed!

Lib injection

```
aarch64-linux-android34-clang lib.c -o libc.so -shared -fPIC
adb push libc.so /data/local/tmp/libc.so
adb shell su -c 'setprop wrap.com.autel.charge "LD_PRELOAD=/data/local/tmp/libc.so"'
adb shell su -c '/system/bin/setenforce 0'
```

Android application

«Unpacking»

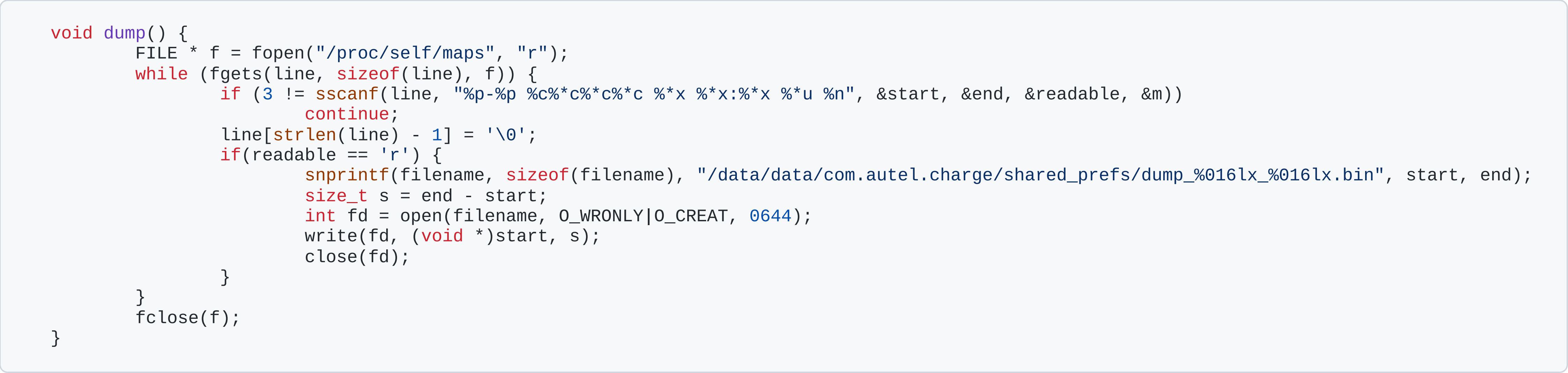
Dump once the application is fully loaded (and unpacked in memory)

```c
int __system_property_get(const char *name, char *value) {
        if(!strcmp("ro.arch", name) && !strcmp(caller_lib(), "libfcfp.so")) {
                log("[%d] %#lx __system_property_get %s => dump\n", getpid(), __builtin_return_address(0), name);
                list_threads();
                dump();
        }
        const prop_info *pi = __system_property_find(name);
        if(pi != 0) {
                return __system_property_read(pi, 0, value);
        } else {
                value[0] = 0;
                return 0;
        }
}
```

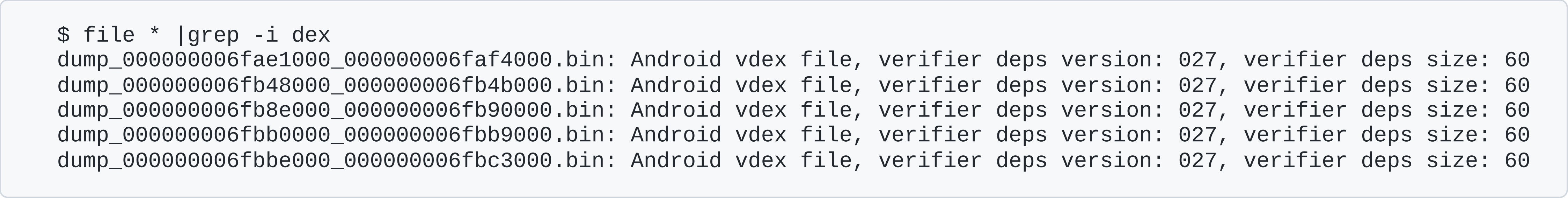
Android application

«Unpacking»

Dump once the application is fully loaded (and unpacked in memory)



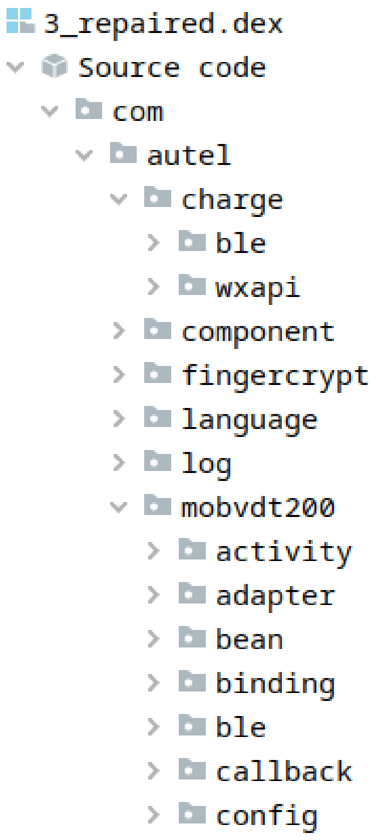
```c
void dump() {
        FILE * f = fopen("/proc/self/maps", "r");
        while (fgets(line, sizeof(line), f)) {
                if (3 != sscanf(line, "%p-%p %c%*c%*c%*c %*x %*x:%*x %*u %n", &start, &end, &readable, &m))
                        continue;
                line[strlen(line) - 1] = '\0';
                if(readable == 'r') {
                        snprintf(filename, sizeof(filename), "/data/data/com.autel.charge/shared_prefs/dump_%016lx_%016lx.bin", start, end);
                        size_t s = end - start;
                        int fd = open(filename, O_WRONLY|O_CREAT, 0644);
                        write(fd, (void *)start, s);
                        close(fd);
                }
        }
        fclose(f);
}
```



```
$ file * |grep -i dex
dump_000000006fae1000_000000006faf4000.bin: Android vdex file, verifier deps version: 027, verifier deps size: 60
dump_000000006fb48000_000000006fb4b000.bin: Android vdex file, verifier deps version: 027, verifier deps size: 60
dump_000000006fb8e000_000000006fb90000.bin: Android vdex file, verifier deps version: 027, verifier deps size: 60
dump_000000006fbb0000_000000006fbb9000.bin: Android vdex file, verifier deps version: 027, verifier deps size: 60
dump_000000006fbbe000_000000006fbc3000.bin: Android vdex file, verifier deps version: 027, verifier deps size: 60
```

Android application

Decompiled DEX



```
3_repaired.dex
⌄ Source code
  ⌄ com
    ⌄ autel
      ⌄ charge
        › ble
        › wxapi
      › component
      › fingercrypt
      › language
      › log
      ⌄ mobvdt200
        › activity
        › adapter
        › bean
        › binding
        › ble
        › callback
        › config
```

- Retrieve application code (*decompiled*)
- Look for OTA download functions

Android application

OTA API

- API to retrieve the firmware links:
  - `https://gateway-eneprodeu.autel.com/api/data-service/device/pile/version/upgrade/ota`
- Parameters:
  - Device serial number
  - Authorization token (from `/login`)
  - "X-Token" provided by the AntiCheat library `libNetHTProtect.so`

```java
public static void getFirmwareInfoV2(String str, a<?> aVar) {
        HashMap hashMap = new HashMap();
        hashMap.put("sn", str);
        try {
                b.j("/api/data-service/device/pile/version/upgrade/ota", hashMap, aVar);
        } catch (Exception e) {
                e.printStackTrace();
        }
}
```

Android application

X-Token

- 4MB library `libNetHTProtect.so`
  - Cryptographic code
  - Hard to reverse engineer
- Easier way to retrieve `X-Token` through the application logs:

```
f10080a = HTProtect.getToken(30000, "f925ce7884a6ae6695f961e0ea181613").token;
c.c = f10080a;
// Log to filesystem
C0207d.m9805h("YiDunSdk", "yidun updateToken = " + f10080a);
```

Android application

Logger storage

- Logs are stored in the application data folder
  - `/data/data/com.autel.charge/Log/AutelCharge-20231026_125734.034-log.txt`
- Logs are encrypted:

```
?=?> <= ?;-<?78;78:#<:=-<944> <944>-D"@bo[iyL}}adnlydbc7-`ldc-}▫bnh~~7nb`#lxyha#nel▫jh-
?=?> <= ?;-<?78;78:#<:<-<944> <944>-I"l7-AbjdcNbcyhuy-dcdy!-▫▫▫▫▫-03kla~h-
?=?> <= ?;-<?78;78:#<::-<944> <944>-I"@bo[iyL}}adnlydbc7-nx▫▫hcy-abjdc-~ylyx~7kla~h-
?=?> <= ?;-<?78;78:#<::-<944> <944>-I"@bo[iyL}}adnlydbc7-chz-abjdc-~ylyx~7-kla~h-
?=?> <= ?;-<?78;78:#<5<-<944> <944>-D"Ih{dnhKdcjh▫]▫dcy7-Ih{dnhKdcjh▫]▫dcyRdcdy###kdcjh▫}▫dcyDi7!ih{dnhKdcjh▫}▫dcy7######-
?=?> <= ?;-<?78;78:#?<>-<944> <944>-I"L}}OlnfAd~yhch▫7-bcLnyd{dytN▫hlyhi-lnyd{dyt-0--nb`#lxyha#`bo{iy?==#lnyd{dyt#^}al~eLnyd{dytM<;n=?;?-!-lnyNbxcy-0--=-
?=?> <= ?;-<?78;78:#?>5-<944> <944>-I"L}}OlnfAd~yhch▫7-bcLnyd{dytN▫hlyhi-lnyd{dyt-0--nb`#lxyha#`bo{iy?==#lnyd{dyt#Abjb]ljh▫Lnyd{dytM4il?<88-!-lnyNbxcy-0--=-
?=?> <= ?;-<?78;78:#?>4-<944> <944>-I"^n▫hhcLil}yXyda7-▫▫▫▫▫▫▫-~B▫djdcIhc~dyt-0->#8-!~B▫djdcKbcy^nlah-0-<#=-!-ihc~dytI}d-0-8;=-
?=?> <= ?;-<?78;78:#?>4-<944> <944>-I"^n▫hhcLil}yXyda7-33333333333333▫▫▫-ihc~dyt-0->#8-kbcy^nlah-0-<#=-!~nlahIhc~dyt-0->#8--!ihc~dytI}d-0-8;=-
?=?> <= ?;-<?78;78:#?>4-<944> <944>-I"^n▫hhcLil}yXyda7-33333333333333▫▫▫-ihc~dyt-0->#8;=>98-kbcy^nlah-0-<#=-!~nlahIhc~dyt-0->#8;=>98--!ihc~dytI}d-0-8;4-
?=?> <= ?;-<?78;78:#><<-<944> <944>-I"GZho^bnfhyNadhcy^h▫{dnh7-nab~hNbcchny-
?=?> <= ?;-<?78;78:#>?5-<944> <944>-I"L}}OlnfAd~yhch▫7-bcLnyd{dyt^yl▫yhi-lnyd{dyt-0--nb`#lxyha#`bo{iy?==#lnyd{dyt#Abjb]ljh▫Lnyd{dytM4il?<88-!-lnyNbxcy-0--=-
```

Android application

Logger encryption

- Single byte XOR encryption

```
// Xor file encryption
public static String m9743c(String str) {
        if (TextUtils.isEmpty(str)) {
                return "";
        }
        byte[] bytes = str.getBytes(StandardCharsets.UTF_8);
        byte[] bArr = new byte[bytes.length];
        for (int i = 0; i < bytes.length; i++) {
                bArr[i] = (byte) (bytes[i] ^ 13);
        }
        return new String(bArr);
}
```

Android application

Decrypted logs

- `X-Token` can be found in the decrypted logs
- With it, interacting with the *Autel API server* is possible!

```
$ python3 decrypt_hardcore_crypto.py AutelCharge-20231026_125734.034-log.txt | grep -A 2 -B 2 X-Token

2023-10-26 13:13:33.075 16906-17082 D/----: ---- --> GET https://gateway-eneprodeu.autel.com/./chargingDataStatistics?pileSN=AE0.. h2
2023-10-26 13:13:33.075 16906-17082 D/----: ---- X-Region: us
2023-10-26 13:13:33.075 16906-17082 D/----: ---- X-Token: u6D8b99UhX0+GCP2avodkaFbEuPerXM4Yfws2pg==
2023-10-26 13:13:33.076 16906-17082 D/----: ---- X-Model: Pixel 8 Pro
2023-10-26 13:13:33.076 16906-17082 D/----: ---- X-Version: 2.1.1;2.00.50
```

Android application

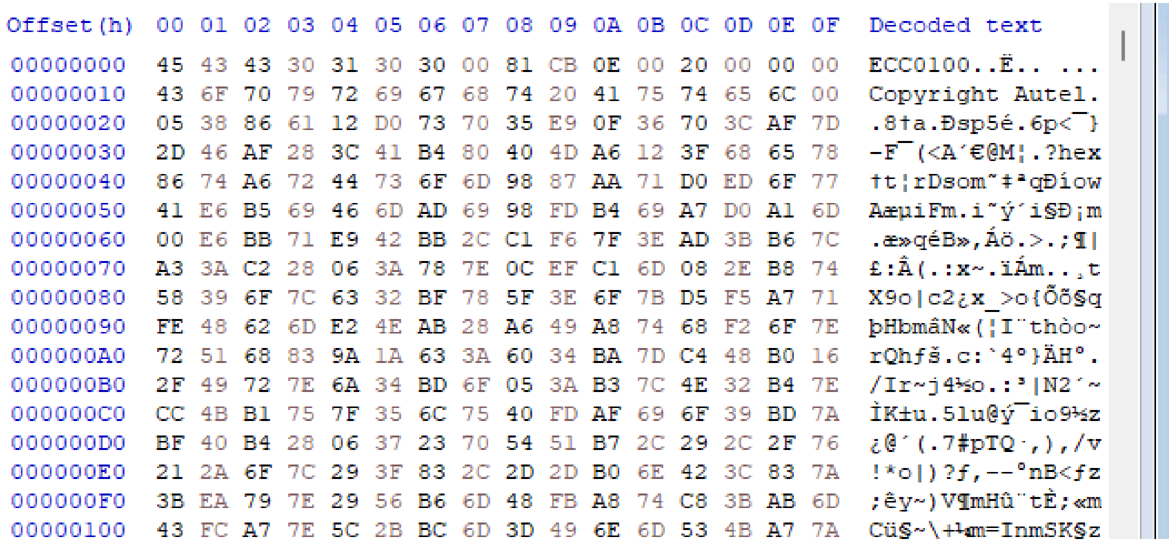
API firmware version downgrade

- `/upgrade/ota` endpoint returns *No new version available*
- `/firmware/syncByApp` endpoint is used to report the charger version
  - Can be used to change the charger version (*fake downgrade*)
- With outdated version, `/upgrade/ota` returns:
  - **Temporary link to download the latest** `firmware.aut`

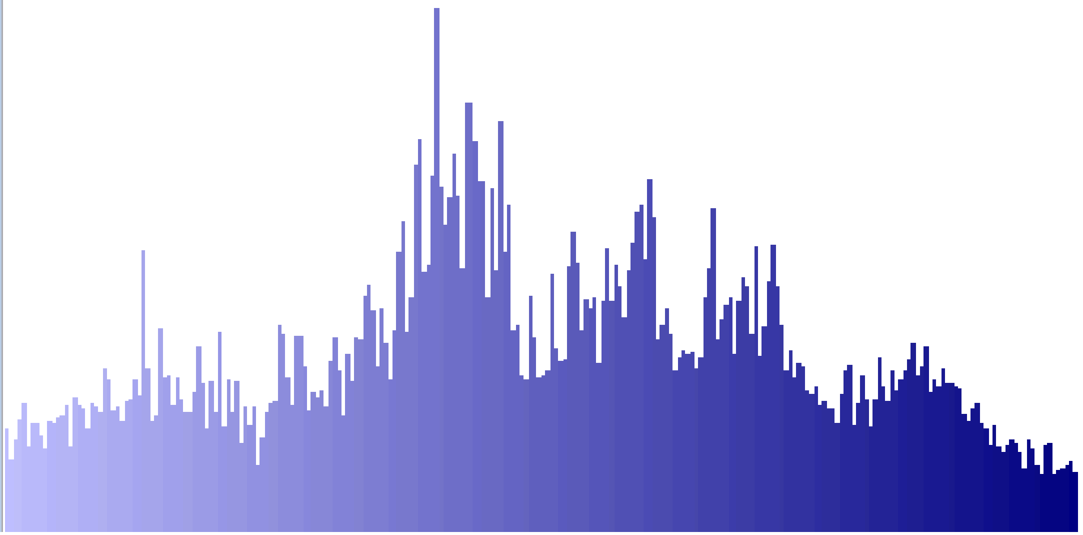
Firmware

Firmware

Firmware_ECC01_V1.42.00.aut



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text

00000000  45 43 43 30 31 30 30 00 81 CB 0E 00 20 00 00 00  ECC0100..Ë.. ...
00000010  43 6F 70 79 72 69 67 68 74 20 41 75 74 65 6C 00  Copyright Autel.
00000020  05 38 86 61 12 D0 73 70 35 E9 0F 36 70 3C AF 7D  .8†a.Ðsp5é.6p<¯}
00000030  2D 46 AF 28 3C 41 B4 80 40 4D A6 12 3F 68 65 78  -F¯(<A´€@M¦.?hex
00000040  86 74 A6 72 44 73 6F 6D 98 87 AA 71 D0 ED 6F 77  †t¦rDsom˜‡ªqÐíow
00000050  41 E6 B5 69 46 6D AD 69 98 FD B4 69 A7 D0 A1 6D  AæµiFm.i˜ý´i§Ð¡m
00000060  00 E6 BB 71 E9 42 BB 2C C1 F6 7F 3E AD 3B B6 7C  .æ»qéB»,Áö.>.;¶|
00000070  A3 3A C2 28 06 3A 78 7E 0C EF C1 6D 08 2E B8 74  £:Â(.:x~.ïÁm..¸t
00000080  58 39 6F 7C 63 32 BF 78 5F 3E 6F 7B D5 F5 A7 71  X9o|c2¿x_>o{Õõ§q
00000090  FE 48 62 6D E2 4E AB 28 A6 49 A8 74 68 F2 6F 7E  þHbmâN«(¦I¨thòo~
000000A0  72 51 68 83 9A 1A 63 3A 60 34 BA 7D C4 48 B0 16  rQhfš.c:`4º}ÄH°.
000000B0  2F 49 72 7E 6A 34 BD 6F 05 3A B3 7C 4E 32 B4 7E  /Ir~j4½o.:³|N2´~
000000C0  CC 4B B1 75 7F 35 6C 75 40 FD AF 69 6F 39 BD 7A  ÌK±u.5lu@ý¯io9½z
000000D0  BF 40 B4 28 06 37 23 70 54 51 B7 2C 29 2C 2F 76  ¿@´(.7#pTQ·,),/v
000000E0  21 2A 6F 7C 29 3F 83 2C 2D 2D B0 6E 42 3C 83 7A  !*o|)?ƒ,--°nB<ƒz
000000F0  3B EA 79 7E 29 56 B6 6D 48 FB A8 74 C8 3B AB 6D  ;êy~)V¶mHû¨tÈ;«m
00000100  43 FC A7 7E 5C 2B BC 6D 3D 49 6E 6D 53 4B A7 7A  Cü§~\+¼m=InmSK§z
```



- Probably encrypted or compressed
- Byte value distribution seems **skewed**

Firmware

Firmware block analysis

Number of repeated 16-byte blocks in the firmware

```
$ xxd -c 16 -p Firmware_ECC01_V1.42.00.aut | sort | \
  uniq -c  | sort -nr | head -n 10
     49 8a74ae208d6fb3788f7ba30a3f686578
     46 8674a67244736f6d9887aa7197366e6f
     46 5d218641553066688170e2e976aac75
     43 9434b461466dad699734b3615472a265
     42 9e72a664463e2d38533a324a214a4153
     42 86646074827974248a6fa16693767472
     42 32706f769b6eb0675074a674956ca576
     41 9782657b8f4d6032936ead75856fad0e
     41 977b6074a674b07098806073a62da469
```

Firmware

Firmware large block analysis

Number of repeated 256-byte blocks in the firmware

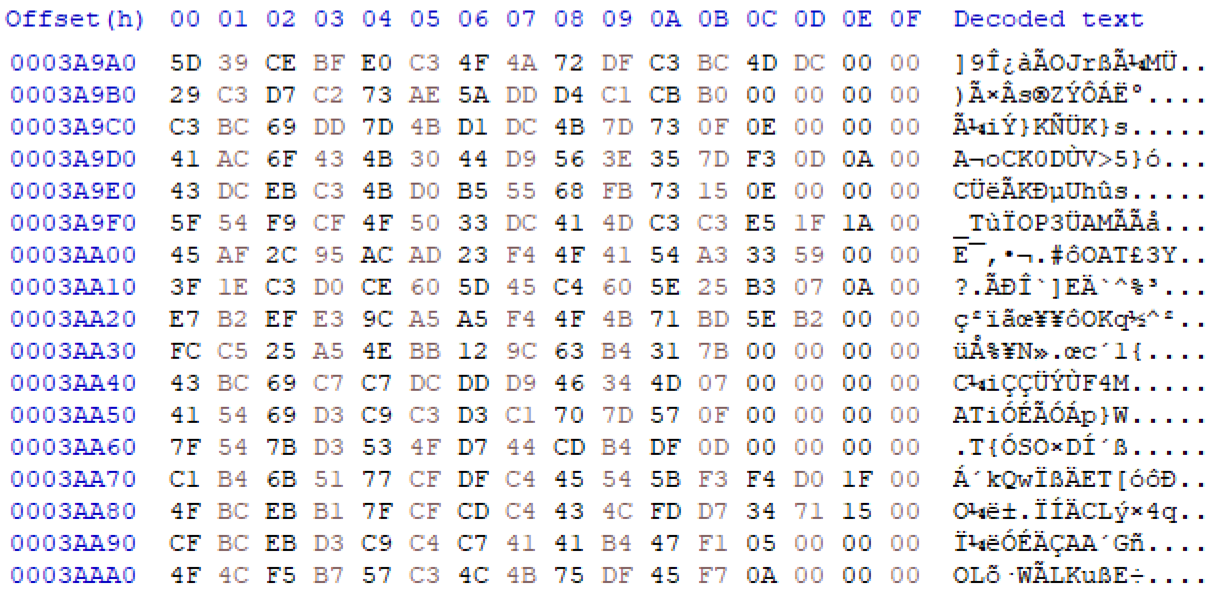
```
$ xxd -c 256 -p Firmware_ECC01_V1.42.00.aut | sort | \
  uniq -c  | sort -nr | head -n 2
     36 aa25a4766365af65...
      1 ff22917ea34fbdb5...
```

- Block of `null` bytes?
- Block of `0xff` bytes?

Firmware

Cryptanalysis: Attempt 1: XOR

XOR the firmware with the repeated block of 256 bytes

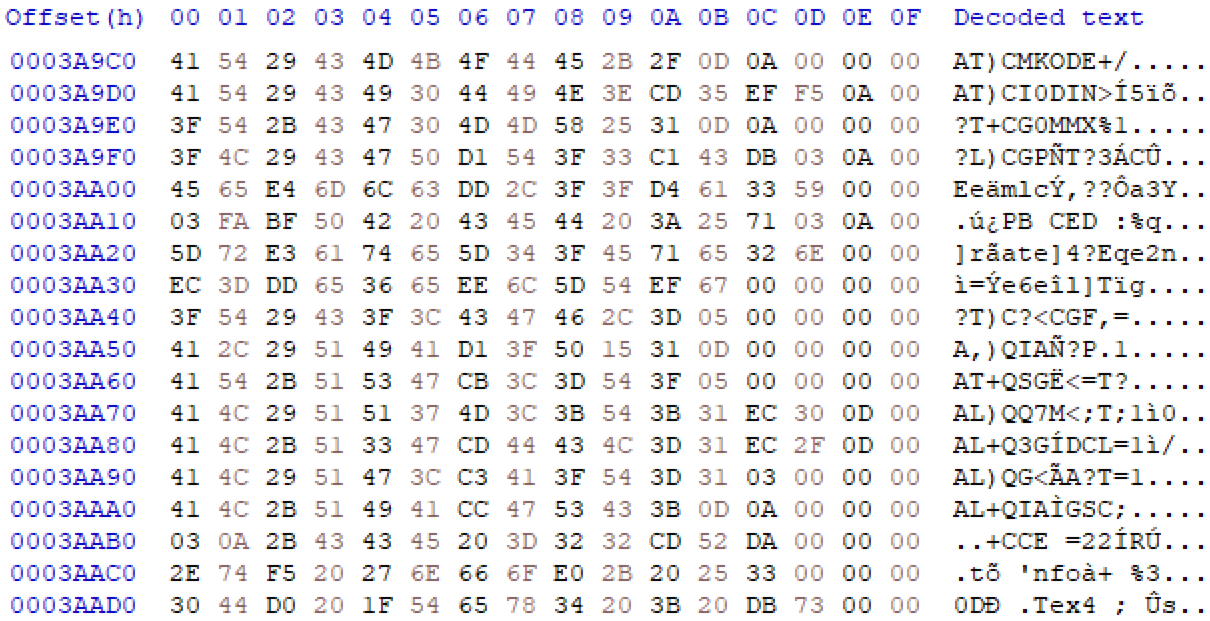


```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text

0003A9A0   5D 39 CE BF E0 C3 4F 4A 72 DF C3 BC 4D DC 00 00  ]9Î¿àÃOJrßÃ¼MÜ..
0003A9B0   29 C3 D7 C2 73 AE 5A DD D4 C1 CB B0 00 00 00 00  )Ã×Âs®ZÝÔÁË°....
0003A9C0   C3 BC 69 DD 7D 4B D1 DC 4B 7D 73 0F 0E 00 00 00  Ã¼iÝ}KÑÜK}s.....
0003A9D0   41 AC 6F 43 4B 30 44 D9 56 3E 35 7D F3 0D 0A 00  A¬oCK0DÙV>5}ó...
0003A9E0   43 DC EB C3 4B D0 B5 55 68 FB 73 15 0E 00 00 00  CÜëÃKÐµUhûs.....
0003A9F0   5F 54 F9 CF 4F 50 33 DC 41 4D C3 C3 E5 1F 1A 00  _TùÏOP3ÜAMÃÃå...
0003AA00   45 AF 2C 95 AC AD 23 F4 4F 41 54 A3 33 59 00 00  E¯,•¬.#ôOAT£3Y..
0003AA10   3F 1E C3 D0 CE 60 5D 45 C4 60 5E 25 B3 07 0A 00  ?.ÃÐÎ`]EÄ`^%³...
0003AA20   E7 B2 EF E3 9C A5 A5 F4 4F 4B 71 BD 5E B2 00 00  ç²ïãœ¥¥ôOKq½^²..
0003AA30   FC C5 25 A5 4E BB 12 9C 63 B4 31 7B 00 00 00 00  üÅ%¥N».œc´1{....
0003AA40   43 BC 69 C7 C7 DC DD D9 46 34 4D 07 00 00 00 00  C¼iÇÇÜÝÙF4M.....
0003AA50   41 54 69 D3 C9 C3 D3 C1 70 7D 57 0F 00 00 00 00  ATiÓÉÃÓÁp}W.....
0003AA60   7F 54 7B D3 53 4F D7 44 CD B4 DF 0D 00 00 00 00  .T{ÓSO×DÍ´ß.....
0003AA70   C1 B4 6B 51 77 CF DF C4 45 54 5B F3 F4 D0 1F 00  Á´kQwÏßÄET[óôÐ..
0003AA80   4F BC EB B1 7F CF CD C4 43 4C FD D7 34 71 15 00  O¼ë±.ÏÍÄCLý×4q..
0003AA90   CF BC EB D3 C9 C4 C7 41 41 B4 47 F1 05 00 00 00  Ï¼ëÓÉÄÇAA´Gñ....
0003AAA0   4F 4C F5 B7 57 C3 4C 4B 75 DF 45 F7 0A 00 00 00  OLõ·WÃLKußE÷....
```

Firmware

Cryptanalysis: Attempt 2: SUB (1/2)

Subtract the firmware bytes with the repeated block of 256 bytes



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text

0003A9C0  41 54 29 43 4D 4B 4F 44 45 2B 2F 0D 0A 00 00 00  AT)CMKODE+/.....
0003A9D0  41 54 29 43 49 30 44 49 4E 3E CD 35 EF F5 0A 00  AT)CI0DIN>Í5ïõ..
0003A9E0  3F 54 2B 43 47 30 4D 4D 58 25 31 0D 0A 00 00 00  ?T+CG0MMX%1.....
0003A9F0  3F 4C 29 43 47 50 D1 54 3F 33 C1 43 DB 03 0A 00  ?L)CGPÑT?3ÁCÛ...
0003AA00  45 65 E4 6D 6C 63 DD 2C 3F 3F D4 61 33 59 00 00  EeämlcÝ,??Ôa3Y..
0003AA10  03 FA BF 50 42 20 43 45 44 20 3A 25 71 03 0A 00  .ú¿PB CED :%q...
0003AA20  5D 72 E3 61 74 65 5D 34 3F 45 71 65 32 6E 00 00  ]rãate]4?Eqe2n..
0003AA30  EC 3D DD 65 36 65 EE 6C 5D 54 EF 67 00 00 00 00  ì=Ýe6eîl]Tïg....
0003AA40  3F 54 29 43 3F 3C 43 47 46 2C 3D 05 00 00 00 00  ?T)C?<CGF,=.....
0003AA50  41 2C 29 51 49 41 D1 3F 50 15 31 0D 00 00 00 00  A,)QIAÑ?P.1.....
0003AA60  41 54 2B 51 53 47 CB 3C 3D 54 3F 05 00 00 00 00  AT+QSGË<=T?.....
0003AA70  41 4C 29 51 51 37 4D 3C 3B 54 3B 31 EC 30 0D 00  AL)QQ7M<;T;1ì0..
0003AA80  41 4C 2B 51 33 47 CD 44 43 4C 3D 31 EC 2F 0D 00  AL+Q3GÍDCL=1ì/..
0003AA90  41 4C 29 51 47 3C C3 41 3F 54 3D 31 03 00 00 00  AL)QG<ÃA?T=1....
0003AAA0  41 4C 2B 51 49 41 CC 47 53 43 3B 0D 0A 00 00 00  AL+QIAÌGSC;.....
0003AAB0  03 0A 2B 43 43 45 20 3D 32 32 CD 52 DA 00 00 00  ..+CCE =22ÍRÚ...
0003AAC0  2E 74 F5 20 27 6E 66 6F E0 2B 20 25 33 00 00 00  .tõ 'nfoà+ %3...
0003AAD0  30 44 D0 20 1F 54 65 78 34 20 3B 20 DB 73 00 00  0DÐ .Tex4 ; Ûs..
```

- AT commands ?

Firmware

Cryptanalysis: Attempt 2: SUB (2/2)

- Subtracting the key looks promising

```
$ strings -el sub_fw.bin

Aoemsficatigf erRor mith cenlrUd bUard
LhYs cXar_]r has a\rcady been r]servc\
Qlgpped by lappiV_ Slgp en DA<
Car\ r]a\]r cgeeuficalWgn crRor
AP f]galtin] n]lLae] abnorcal
Do yUm w_fl lo slgp char]iV_5
Gfl]rf_l gn]rlemp]raTure
PoW]r supply dgscg^n[cle\
```

- `Guessing 500 CTF`

Firmware

Cryptanalysis: Attempt 3: Fixing the errors

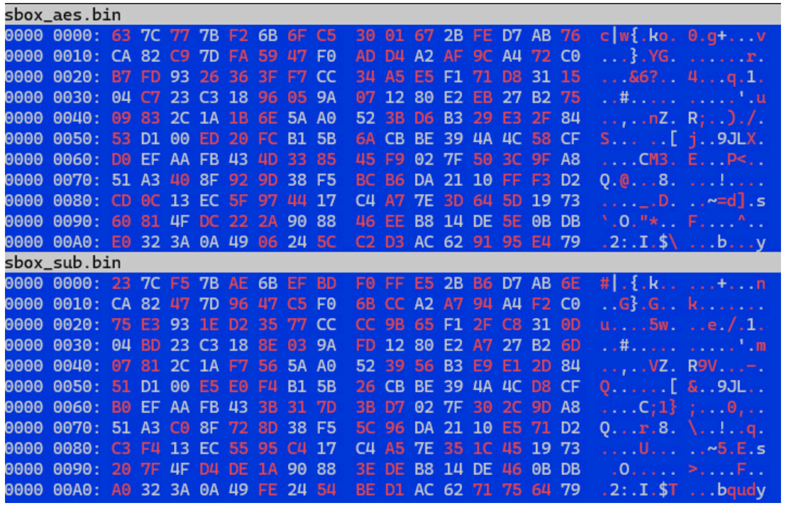
- Still many errors in strings
- Guessed one string to check hypothesis
  - Erroneous `has a\rcady been r]servc\`
  - Corrected `has already been reserved`
- No single bit errors
- No error from integer underflow in subtraction
  - on 8/16/32-bit
- Erroneous bits appear to be at random positions
  - except on NULL bytes (no error in UTF-16 strings)

Firmware

Cryptanalysis: Attempt 4

Breakthrough

- **Idea**: look for known long plaintext
- Crypto tables are good candidates, find an area with a lot of matching bytes in sub-decrypted
- Identified `AES SBOX` and `RSBOX` in the binary



```
sbox_aes.bin
0000 0000: 63 7C 77 7B F2 6B 6F C5  30 01 67 2B FE D7 AB 76  c|w{.ko. 0.g+...v
0000 0010: CA 82 C9 7D FA 59 47 F0  AD D4 A2 AF 9C A4 72 C0  ...}.YG. ......r.
0000 0020: B7 FD 93 26 36 3F F7 CC  34 A5 E5 F1 71 D8 31 15  ...&6?.. 4...q.1.
0000 0030: 04 C7 23 C3 18 96 05 9A  07 12 80 E2 EB 27 B2 75  ..#..... .....'.u
0000 0040: 09 83 2C 1A 1B 6E 5A A0  52 3B D6 B3 29 E3 2F 84  ..,..nZ. R;..)./.
0000 0050: 53 D1 00 ED 20 FC B1 5B  6A CB BE 39 4A 4C 58 CF  S... ..[ j..9JLX.
0000 0060: D0 EF AA FB 43 4D 33 85  45 F9 02 7F 50 3C 9F A8  ....CM3. E...P<..
0000 0070: 51 A3 40 8F 92 9D 38 F5  BC B6 DA 21 10 FF F3 D2  Q.@...8. ...!....
0000 0080: CD 0C 13 EC 5F 97 44 17  C4 A7 7E 3D 64 5D 19 73  ...._.D. ..~=d].s
0000 0090: 60 81 4F DC 22 2A 90 88  46 EE B8 14 DE 5E 0B DB  `.O."*.. F....^..
0000 00A0: E0 32 3A 0A 49 06 24 5C  C2 D3 AC 62 91 95 E4 79  .2:.I.$\ ...b...y
sbox_sub.bin
0000 0000: 23 7C F5 7B AE 6B EF BD  F0 FF E5 2B B6 D7 AB 6E  #|.{.k.. ...+...n
0000 0010: CA 82 47 7D 96 47 C5 F0  6B CC A2 A7 94 A4 F2 C0  ..G}.G.. k.......
0000 0020: 75 E3 93 1E D2 35 77 CC  CC 9B 65 F1 2F C8 31 0D  u....5w. ..e./.1.
0000 0030: 04 BD 23 C3 18 8E 03 9A  FD 12 80 E2 A7 27 B2 6D  ..#..... .....'.m
0000 0040: 07 81 2C 1A F7 56 5A A0  52 39 56 B3 E9 E1 2D 84  ..,..VZ. R9V...-.
0000 0050: 51 D1 00 E5 E0 F4 B1 5B  26 CB BE 39 4A 4C D8 CF  Q......[ &..9JL..
0000 0060: B0 EF AA FB 43 3B 31 7D  3B D7 02 7F 30 2C 9D A8  ....C;1} ;...0,..
0000 0070: 51 A3 C0 8F 72 8D 38 F5  5C 96 DA 21 10 E5 71 D2  Q...r.8. \..!..q.
0000 0080: C3 F4 13 EC 55 95 C4 17  C4 A5 7E 35 1C 45 19 73  ....U... ..~5.E.s
0000 0090: 20 7F 4F D4 DE 1A 90 88  3E DE B8 14 DE 46 0B DB   .O..... >....F..
0000 00A0: A0 32 3A 0A 49 FE 24 54  BE D1 AC 62 71 75 64 79  .2:.I.$T ...bqudy
```

Firmware

Cryptanalysis: Attempt 4

Analysis with known plaintext

- With 512 bytes of plaintext:
  - ~50% of error
  - LSB always correct
- Identified more plaintext:
  - AES S-BOX / RS-BOX
  - SHA256 const table
  - 4 Camellia S-BOX
- Tables come from the open-source TLS library `mbedtls`
- **7 blocks of 256 bytes known**

Firmware

Cryptanalysis: Attempt 4

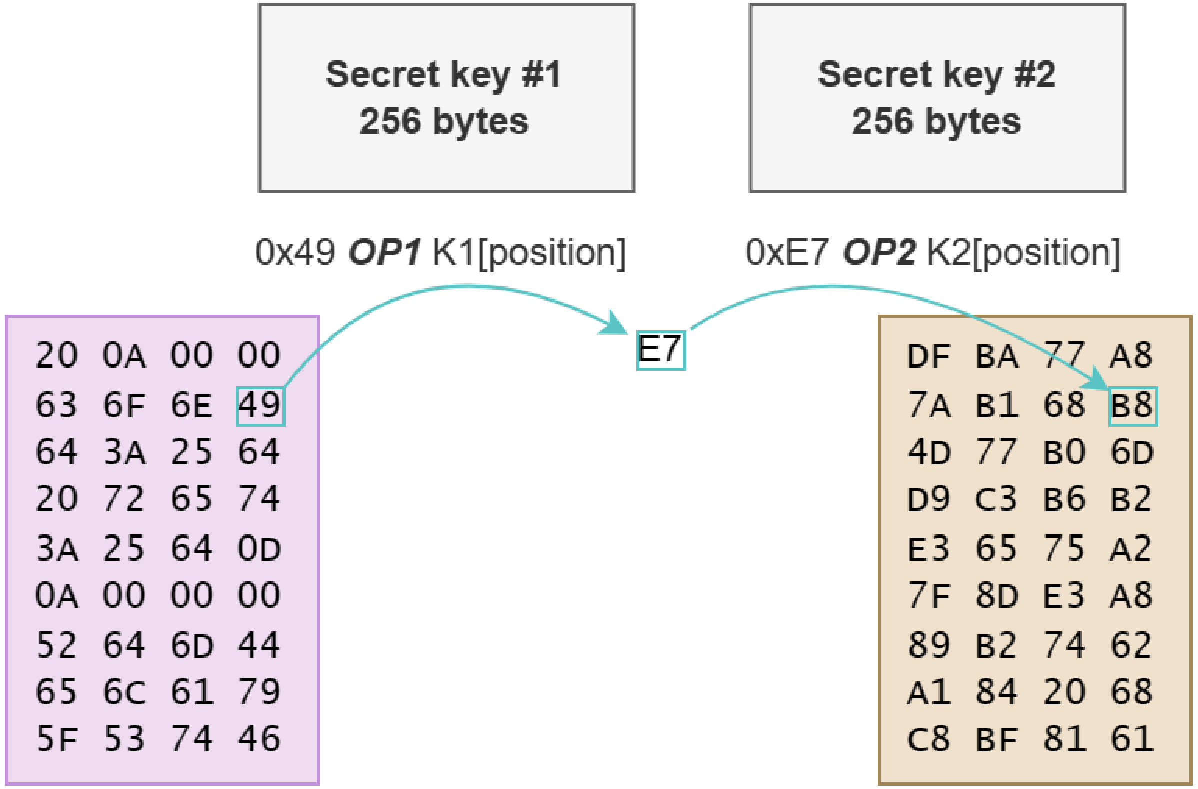
Hypothesis

- More than one operation (xor, sub, times, …)
- Two or more keys of 256 bytes repeated
- Encryption operates each byte independently
- Operations must not lose information

Firmware

Cryptanalysis: Attempt 4

Hypothesis scheme



Secret key #1
256 bytes

Secret key #2
256 bytes

0x49 ***OP1*** K1[position]

0xE7 ***OP2*** K2[position]

```
20 0A 00 00
63 6F 6E 49
64 3A 25 64
20 72 65 74
3A 25 64 0D
0A 00 00 00
52 64 6D 44
65 6C 61 79
5F 53 74 46
```

E7

```
DF BA 77 A8
7A B1 68 B8
4D 77 B0 6D
D9 C3 B6 B2
E3 65 75 A2
7F 8D E3 A8
89 B2 74 62
A1 84 20 68
C8 BF 81 61
```

Firmware

Cryptanalysis: Attempt 4

Bruteforcing operations

- SAT solver z3
- Pick two operations `op1` `op2`
- Validate at least one solution exists for each byte in known blocks, for each position

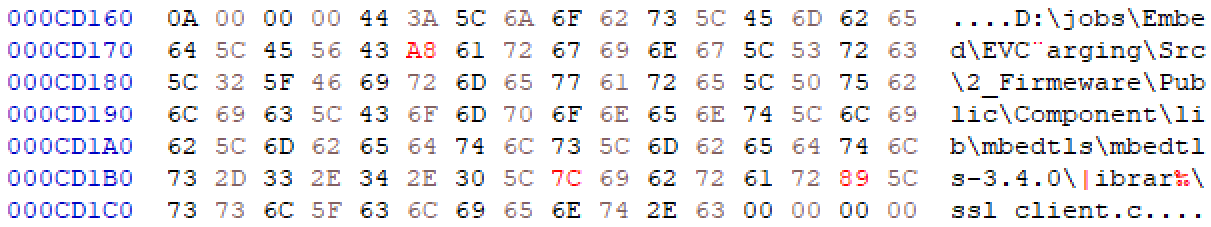
```
ops = [
    "__add__", "__mul__", "__sub__", "__and__", "__xor__",
    "__div__", "__mod__", "__rshift__", "__lshift__"
]
for op1 in ops:    # Pick operation #1
  for op2 in ops:   # Pick operation #2
    for pos in range(256): # Test for all positions
      k1 = BitVec('k1', 8) # Key 1
      k2 = BitVec('k2', 8) # Key 2
      s = Solver()
      for ciphertext, plaintext in known.items():
        s.add(plaintext[pos] == k2.__getattribute__(op2)(k1.__getattribute__(op1)(ciphertext[pos]))
      if s.check() != sat:
        print("Impossible op {op1} - {op2}")
        # ...
```

Firmware

Cryptanalysis: Attempt 4

Results

- No working result with 3+ operations
- Only `add` and `xor` have one or more solutions for each position
- Pick one solution for each position
- Then try to decrypt the whole firmware



```
000CD160  0A 00 00 00 44 3A 5C 6A 6F 62 73 5C 45 6D 62 65  ....D:\jobs\Embe
000CD170  64 5C 45 56 43 A8 61 72 67 69 6E 67 5C 53 72 63  d\EVC¨arging\Src
000CD180  5C 32 5F 46 69 72 6D 65 77 61 72 65 5C 50 75 62  \2_Firmware\Pub
000CD190  6C 69 63 5C 43 6F 6D 70 6F 6E 65 6E 74 5C 6C 69  lic\Component\li
000CD1A0  62 5C 6D 62 65 64 74 6C 73 5C 6D 62 65 64 74 6C  b\mbedtls\mbedtl
000CD1B0  73 2D 33 2E 34 2E 30 5C 7C 69 62 72 61 72 89 5C  s-3.4.0\|ibrar%\
000CD1C0  73 73 6C 5F 63 6C 69 65 6E 74 2E 63 00 00 00 00  ssl client.c....
```

- **SUCCESS**
- Fix the last errors by adding more plaintext & ciphertext of strings to have a unique solution (k1, k2) for each position

Firmware

Firmware encryption algorithm

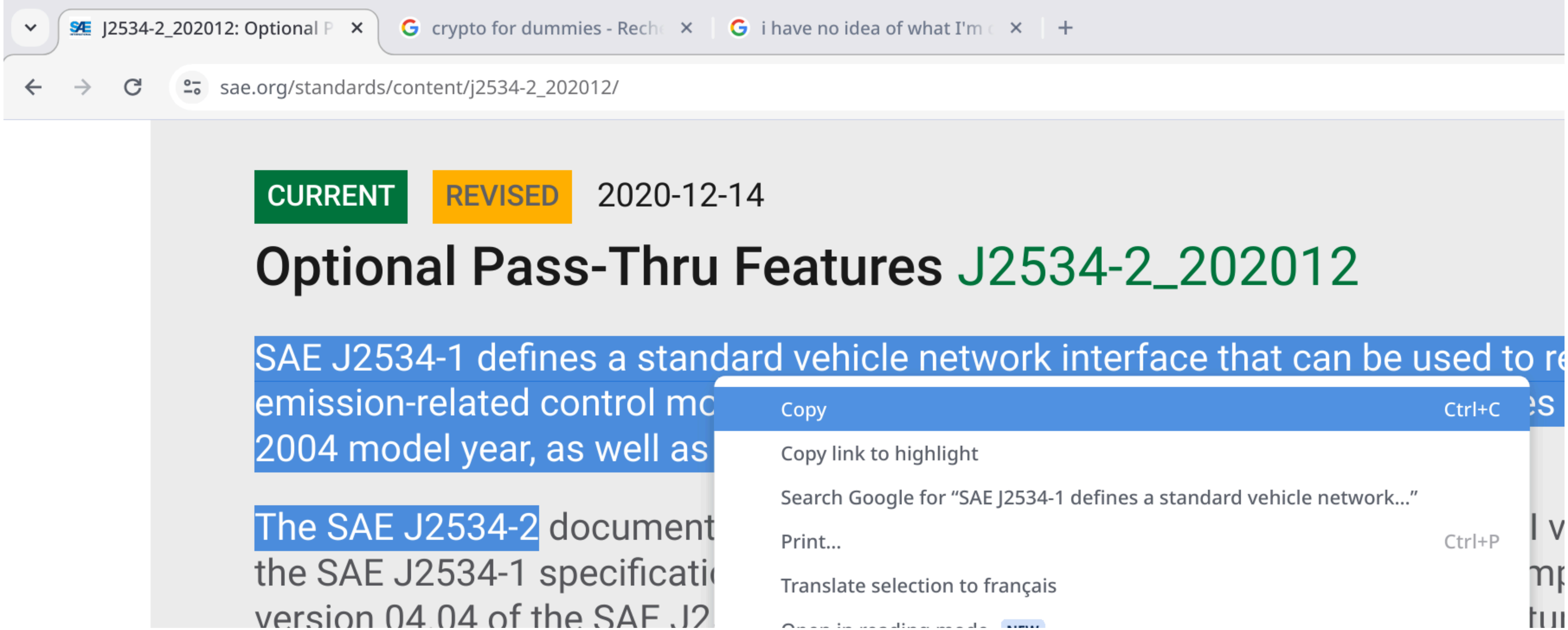
- Now, reverse engineering the real encryption is possible

```
// Decompiled code
const KEY[256] = "SAE J2534-1 defines a standard vehicle network "
  "interface that can be used to reprogram emission-related control\r\n"
  "modules. However, there is a need to support vehicles prior to the 2004 "
  "model year as well as non-emission related\r\n"
  "control modules.\r\nThe SAE J";

uint32_t C = 0x4C11DB7; // CRC32 polynomial because why not

void decrypt_firmware(char *dec, char *enc, unsigned int size) {
  for (uint32_t i = 0 ; i < size ; i++) {
    dec[i] = (C >> (8 * (i & 3))) & KEY[i & 0xFF] ^ (enc[i] - KEY[(~i) & 0xFF]);
  }
}
```

- Correctly identified SUB and XOR
- Took us a week of work to retrieve the decrypted firmware

Firmware

Firmware encryption algorithm



J2534-2_202012: Optional P
crypto for dummies - Rech
i have no idea of what I'm
sae.org/standards/content/j2534-2_202012/

CURRENT REVISED 2020-12-14

Optional Pass-Thru Features J2534-2_202012

SAE J2534-1 defines a standard vehicle network interface that can be used to re
emission-related control mo
2004 model year, as well as

The SAE J2534-2 document
the SAE J2534-1 specificati
version 04.04 of the SAE J2

Copy Ctrl+C
Copy link to highlight
Search Google for “SAE J2534-1 defines a standard vehicle network...”
Print... Ctrl+P
Translate selection to français

Reconstruction of the crime scene

Vulnerability research & exploitation

Firmware

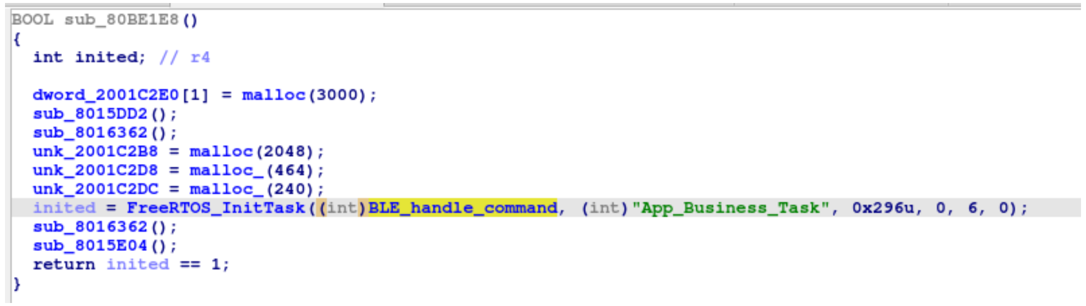
Firmware analysis

- FreeRTOS
- ARMv7 Thumb Mode
- A lot of debug strings

Vulnerability research

Reverse Engineering

Choose an Entry Point: Bluetooth Low Energy



```
BOOL sub_80BE1E8()
{
  int inited; // r4

  dword_2001C2E0[1] = malloc(3000);
  sub_8015DD2();
  sub_8016362();
  unk_2001C2B8 = malloc(2048);
  unk_2001C2D8 = malloc_(464);
  unk_2001C2DC = malloc_(240);
  inited = FreeRTOS_InitTask((int)BLE_handle_command, (int)"App_Business_Task", 0x296u, 0, 6, 0);
  sub_8016362();
  sub_8015E04();
  return inited == 1;
}
```

- BLE_Handler calls a function depending on `operationCode` and `functionCode`

SerialNum (4 bytes)	opCode (2 bytes)	funcCode (2 bytes)	Command (n-8 bytes...)	CRC (2 bytes)

- All functions require authentication except `AppAuthenOperation`

Vulnerability research

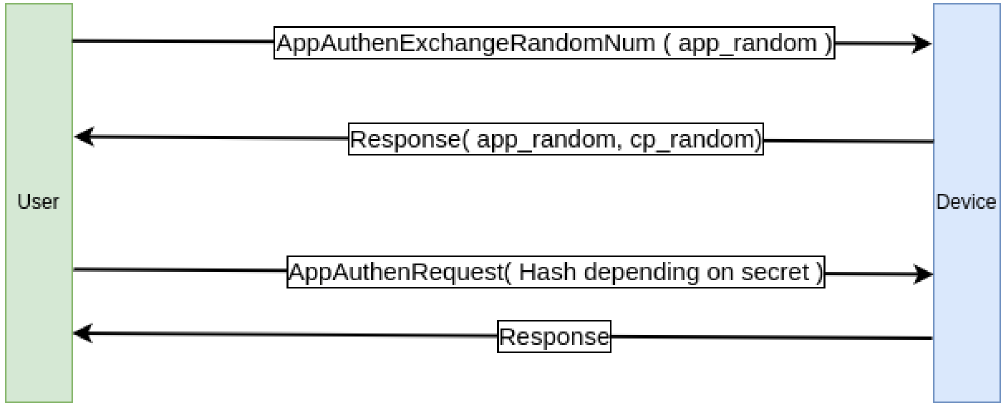
BLE Authentication

AppAuthenExchangeRandomNum

- Exchange random nonces for authentication handshake
  - `app_random` : user-controlled
  - `cp_random` : device-generated random

AppAuthenRequest

- User calculates a hash from nonces, hardcoded values, and a secret password
- Device also calculates the hash and compares the value to authenticate the user


User
Device
AppAuthenExchangeRandomNum (app_random)
Response(app_random, cp_random)
AppAuthenRequest(Hash depending on secret)
Response

Vulnerability research

AppAuthenRequest

Hash Calculation

- A hash is computed based on nonces, hardcoded value, and secret password:

```
string_to_be_hashed = snprintf(v14, authStrLen + 1, "%s:%s:%s:%s", app_random_and_cp_rand,
                        magic_key_1_depends_on_passwd, bt_hardcoded_key, mac_addr);
if ( string_to_be_hashed == authStrLen ) {
        v18 = sha256_((int)v14, string_to_be_hashed, output, 0); [...]}
```

- The hash is compared to the one sent by the user:

```
for ( idx = 0; idx < 0x20u; ++idx ) {
    if ( hash_from_user_input[idx] != computed_hash[idx] )
        auth_failure = 1;
}
```

- Impossible to authenticate without knowing the password

Vulnerability research

AppAuthenRequest

authBD alternative authent

- Classical authent:

```
generate_hash_to_compare(nonces, magic_key_derivated_with_password, computed_hash);
```

- Alternative authent:

```
generate_hash_to_compare(nonces, hardcoded_constructor_key, computed_hash);
```

- Possible to calculate the hash without knowing any secret:

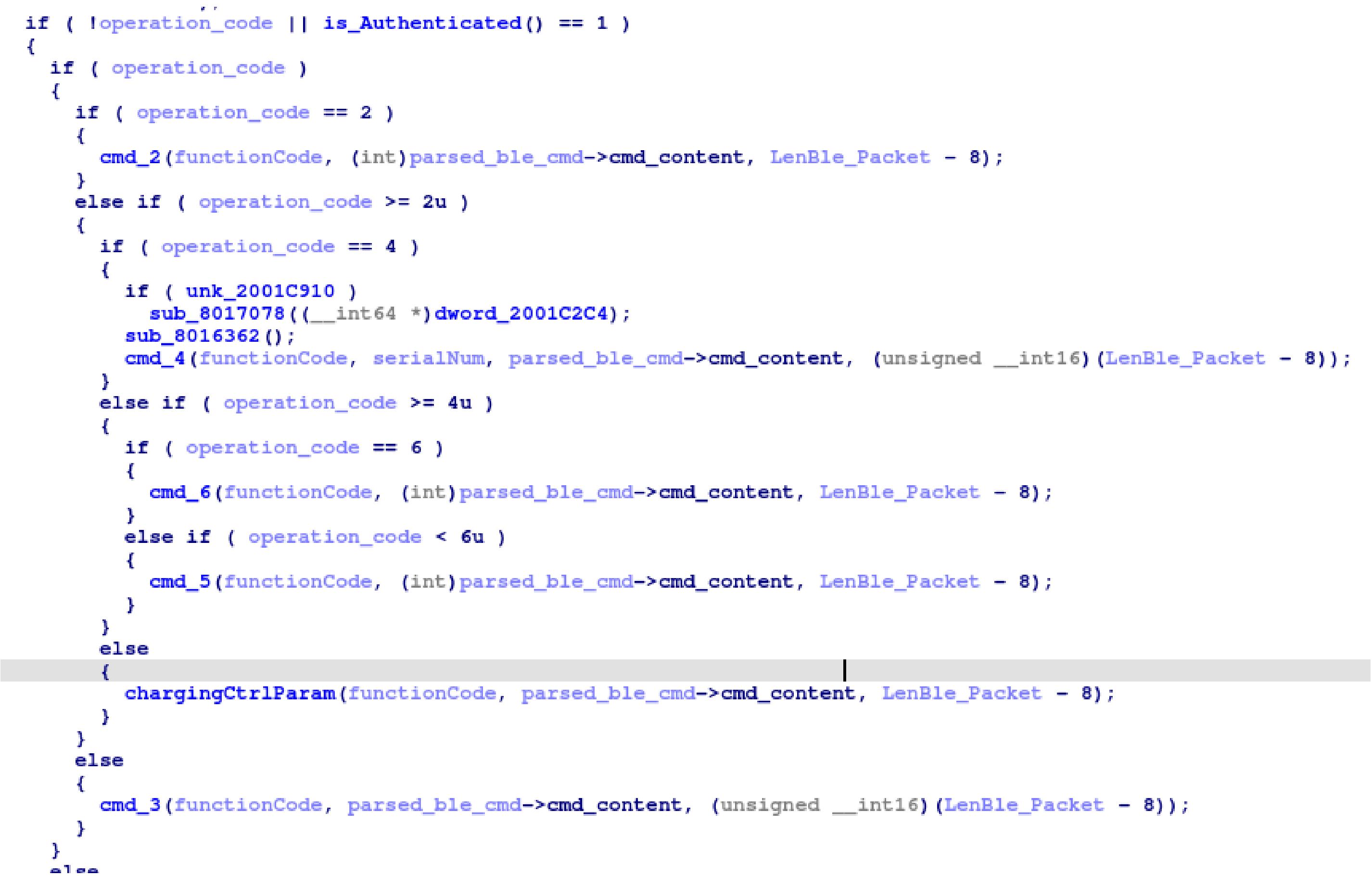
```
set_auth_status(1);
log_something("A_Ble_Bus", 2, 650, "authbd succ\r\n");
```

- What could this "BD" word mean ?

Vulnerability research & exploitation

Authenticated Commands

New attack surface once the authentication is bypassed



```
if ( !operation_code || is_Authenticated() == 1 )
{
  if ( operation_code )
  {
    if ( operation_code == 2 )
    {
      cmd_2(functionCode, (int)parsed_ble_cmd->cmd_content, LenBle_Packet - 8);
    }
    else if ( operation_code >= 2u )
    {
      if ( operation_code == 4 )
      {
        if ( unk_2001C910 )
          sub_8017078((__int64 *)dword_2001C2C4);
        sub_8016362();
        cmd_4(functionCode, serialNum, parsed_ble_cmd->cmd_content, (unsigned __int16)(LenBle_Packet - 8));
      }
      else if ( operation_code >= 4u )
      {
        if ( operation_code == 6 )
        {
          cmd_6(functionCode, (int)parsed_ble_cmd->cmd_content, LenBle_Packet - 8);
        }
        else if ( operation_code < 6u )
        {
          cmd_5(functionCode, (int)parsed_ble_cmd->cmd_content, LenBle_Packet - 8);
        }
      }
      else
      {
        chargingCtrlParam(functionCode, parsed_ble_cmd->cmd_content, LenBle_Packet - 8);
      }
    }
    else
    {
      cmd_3(functionCode, parsed_ble_cmd->cmd_content, (unsigned __int16)(LenBle_Packet - 8));
    }
  }
  else
```

Vulnerability exploitation

chargingCtrlParam

Time to control `$pc`

```
char chargingCtrlParam_stack_buffer[60];
...
memcpy(output_ble_buffer, dword_80F4754, sizeof(output_ble_buffer));
send_ble_response((int)output_ble_buffer, 0x11u);
memcpy(chargingCtrlParam_stack_buffer, cmd_content, cmd_len);
print_log(dword_80F4768);
print_log("chargingCtrlParam.chargingCtrl = 0x%x\r\n", *(_DWORD *)chargingCtrlParam_stack_buffer);
print_log("chargingCtrlParam.chargingMode = 0x%x\r\n", v16);
print_log("chargingCtrlParam.chargingParam = %d\r\n", v17);
print_log("chargingCtrlParam.accountBalance = %d\r\n", v18);
```

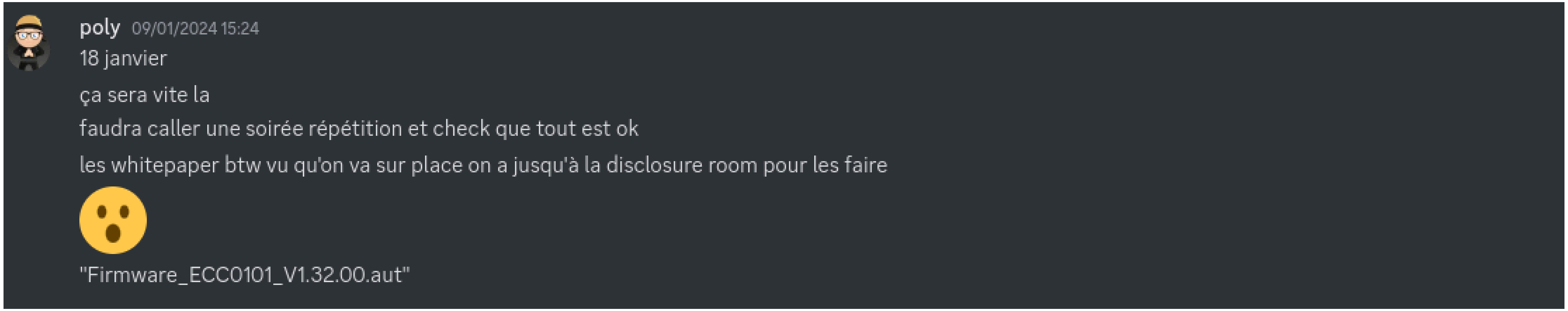
- Stack is executable
- RTOS -> no shell
- Rop to a shellcode on the stack to blink the led

Demo

Vulnerability exploitation

Last minute update

Ten days before shipping the exploit: A new version appears



poly 09/01/2024 15:24

18 janvier

ça sera vite la

faudra caller une soirée répétition et check que tout est ok

les whitepaper btw vu qu'on va sur place on a jusqu'à la disclosure room pour les faire

😮

"Firmware_ECC0101_V1.32.00.aut"

Vulnerability exploitation

Update

Change in the authentication logic

- The code responsible for the authentication has been changed 😱
- Before:

```
generate_hash_to_compare(nonces, hardcoded_constructor_key, computed_hash);
```

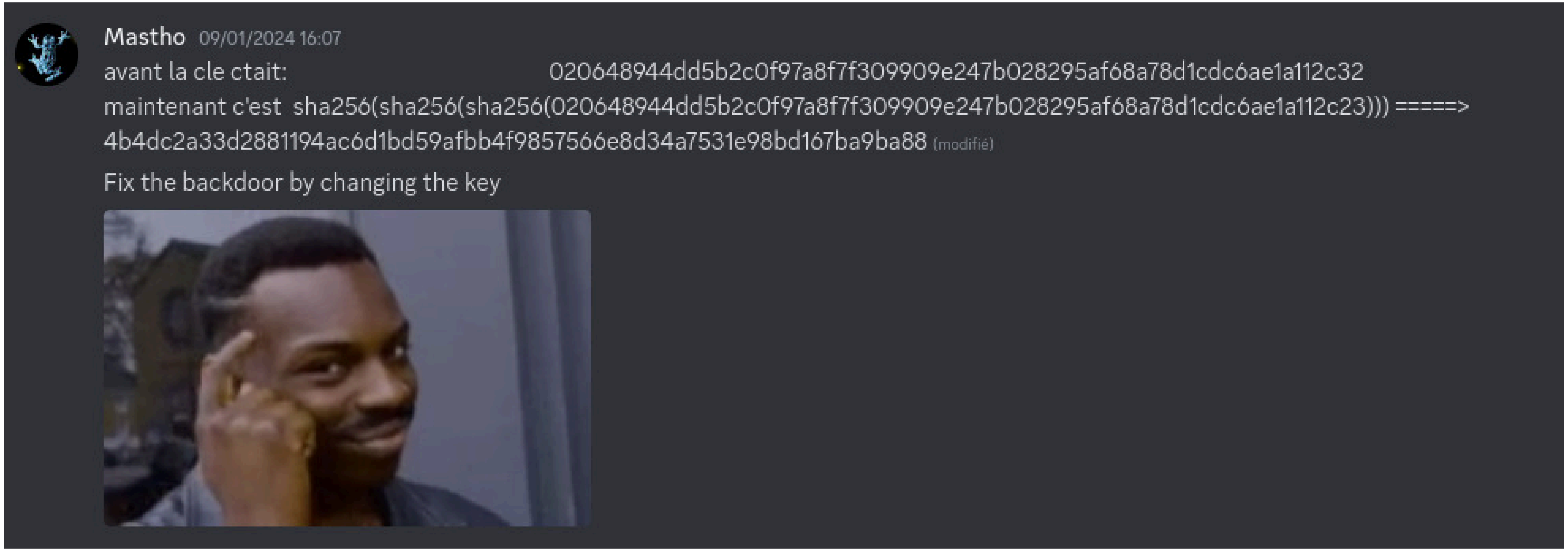
- After:

```
sha256(hardcoded_constructor_key, 32, v20, 0);
sha256(v20, 32, v20, 0);
sha256(v20, 32, v20, 0);
memcpy(hardcoded_constructor_key, v20, 0x20u);
generate_hash_to_compare(nonces, hardcoded_constructor_key, computed_hash);
```

Vulnerability exploitation

Update

Only the backdoor changed



Mastho 09/01/2024 16:07

avant la cle ctait: 020648944dd5b2c0f97a8f7f309909e247b028295af68a78d1cdc6ae1a112c32
maintenant c'est sha256(sha256(sha256(020648944dd5b2c0f97a8f7f309909e247b028295af68a78d1cdc6ae1a112c23))) =====> 4b4dc2a33d2881194ac6d1bd59afbb4f9857566e8d34a7531e98bd167ba9ba88 (modifié)

Fix the backdoor by changing the key

Conclusion

Conclusion

The target

- Most of the time spent getting a firmware
- Vuln research and exploitation were very easy
- Obtaining the firmware is (was?) much more difficult than exploiting it
- Another example of security by obscurity

Pwn2Own

- Good event to work with colleagues and share beers 🍻
- Newcomers in the competition are often good targets 🎯
- Try your luck 🍀

SYNACKTIV

https://www.linkedin.com/company/synacktiv

https://twitter.com/synacktiv

https://synacktiv.com