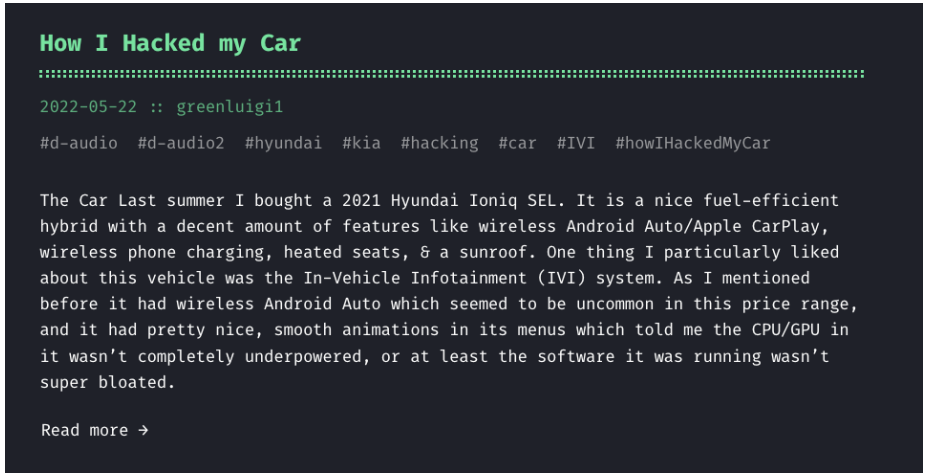
An example of what recently happened during an attack on a specific car brand

A walk through of the pitfalls based on examples

Thomas Rosenstatter, Research in Cybersecurity @RISE

Security pitfalls: Hyundai case



How I Hacked my Car

2022-05-22 :: greenluigi1

#d-audio #d-audio2 #hyundai #kia #hacking #car #IVI #howIHackedMyCar

The Car Last summer I bought a 2021 Hyundai Ioniq SEL. It is a nice fuel-efficient hybrid with a decent amount of features like wireless Android Auto/Apple CarPlay, wireless phone charging, heated seats, & a sunroof. One thing I particularly liked about this vehicle was the In-Vehicle Infotainment (IVI) system. As I mentioned before it had wireless Android Auto which seemed to be uncommon in this price range, and it had pretty nice, smooth animations in its menus which told me the CPU/GPU in it wasn't completely underpowered, or at least the software it was running wasn't super bloated.

Read more →

What was needed?

- The car
- Programming skills
- **Determination**
- **Patience**
- Cheap off the shelf hardware (e.g., USB to Ethernet adapter)

Difference to Miller and Valasek:

- Pure focus on the infotainment system from a software side, without disassembling the hardware in the process
- Remote attacks are not demonstrated nor the aim

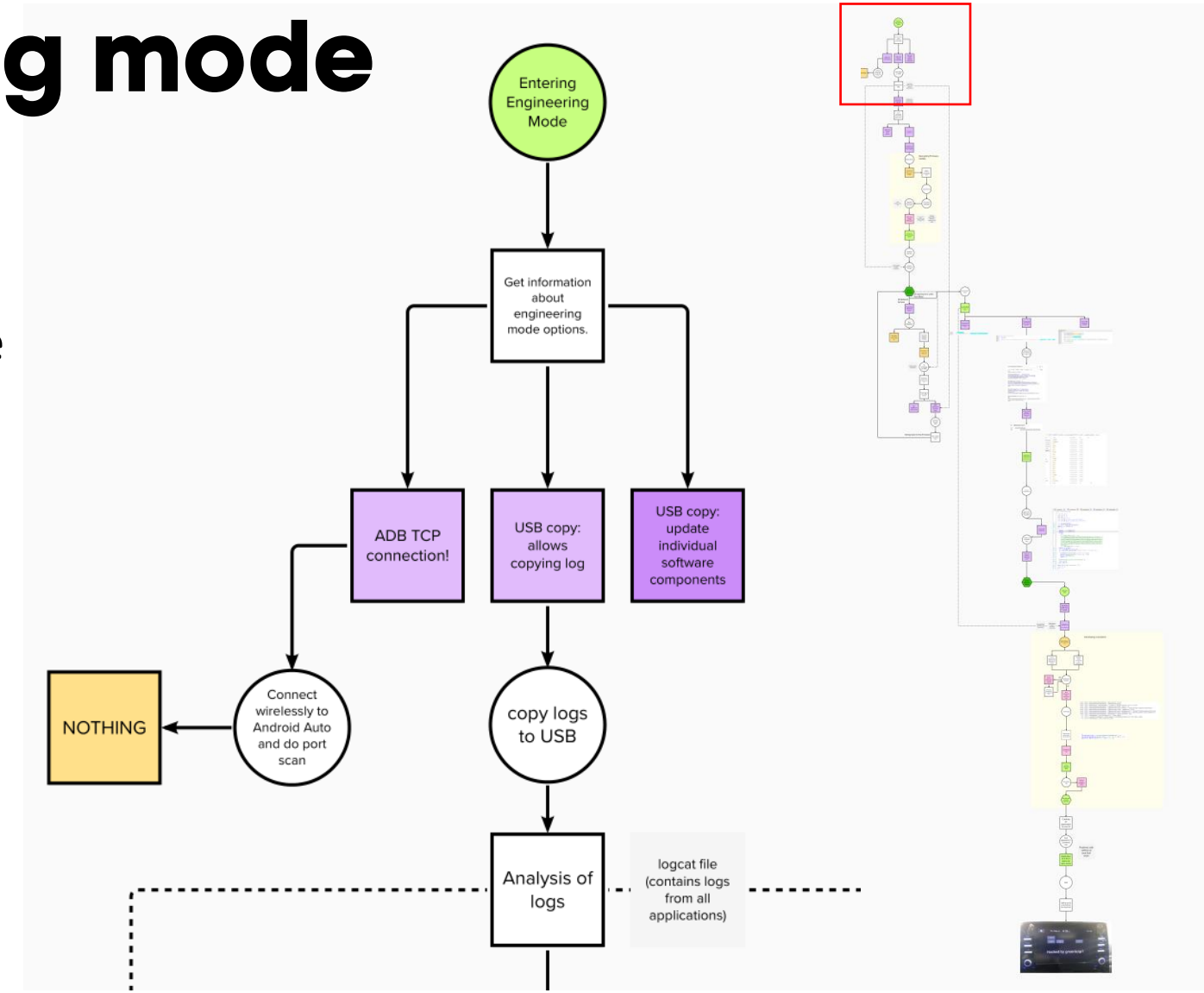
Why was this car chosen?

- Economical reasons (fuel efficiency, …)
- Apple CarPlay, Android Auto
- About the IVI: *”… and it had pretty nice, smooth animations in its menus which told me the CPU/GPU in it wasn’t completely underpowered, or at least the software it was running wasn’t super bloated.”*

(1) Engineering mode

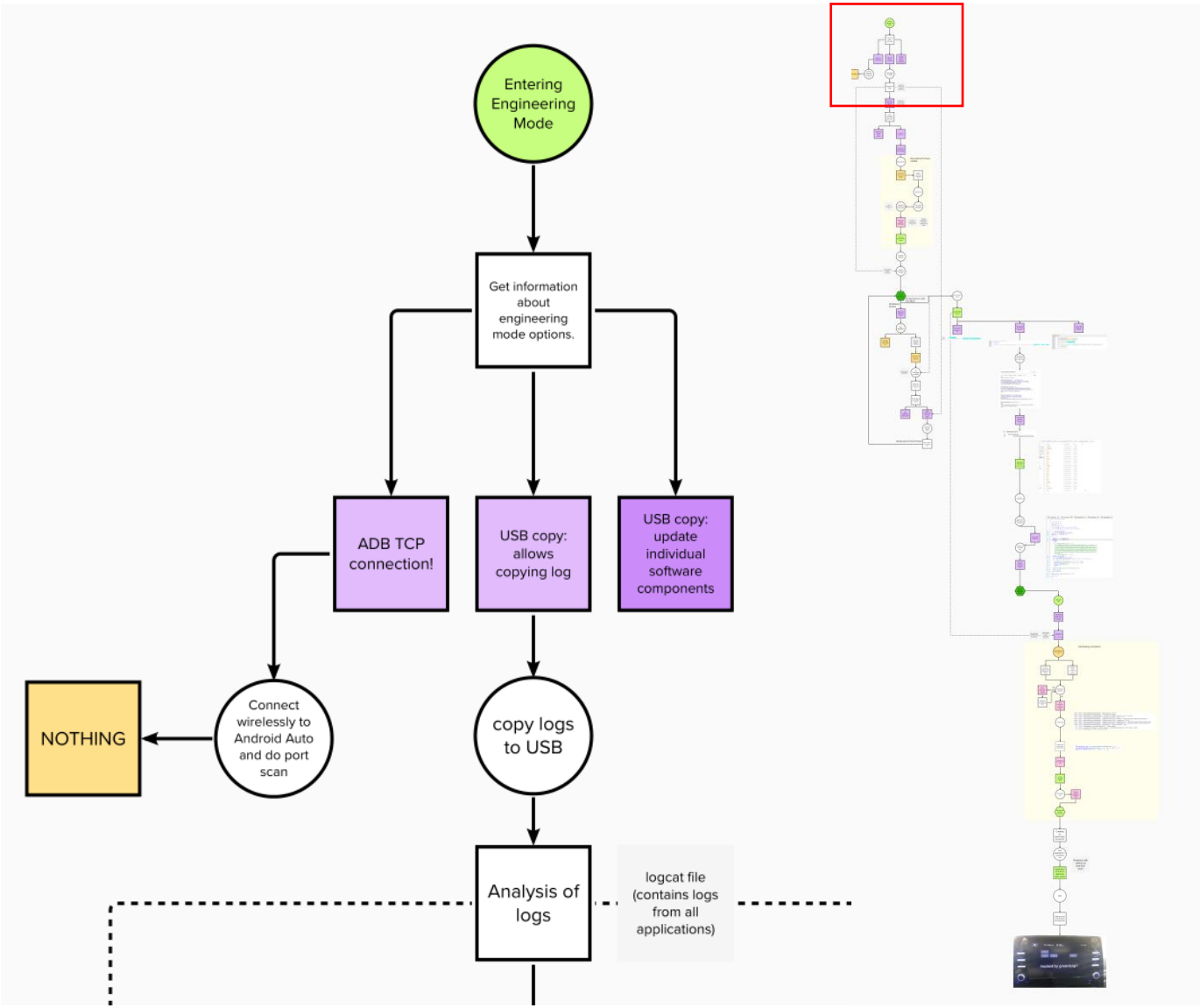
- Already extensive work done and easy to find on the web.

M.1 Often 4-digit pin to guess
(default, easy depends on year, or time + 6 last digits of SW version)



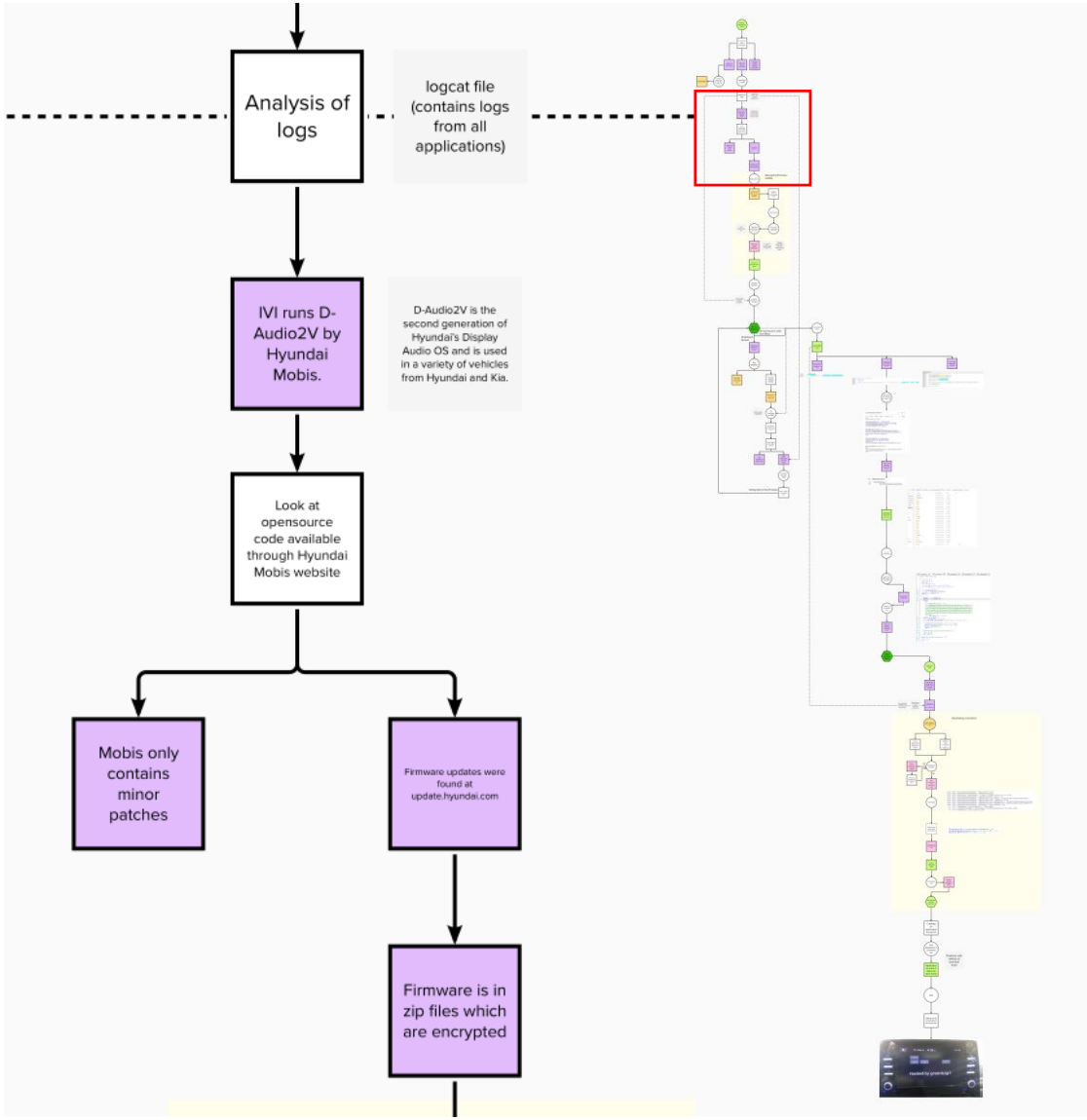
(2) Options

- **ADB secured, no other ports available**
- USB copy for logs
- Upgrade individual software



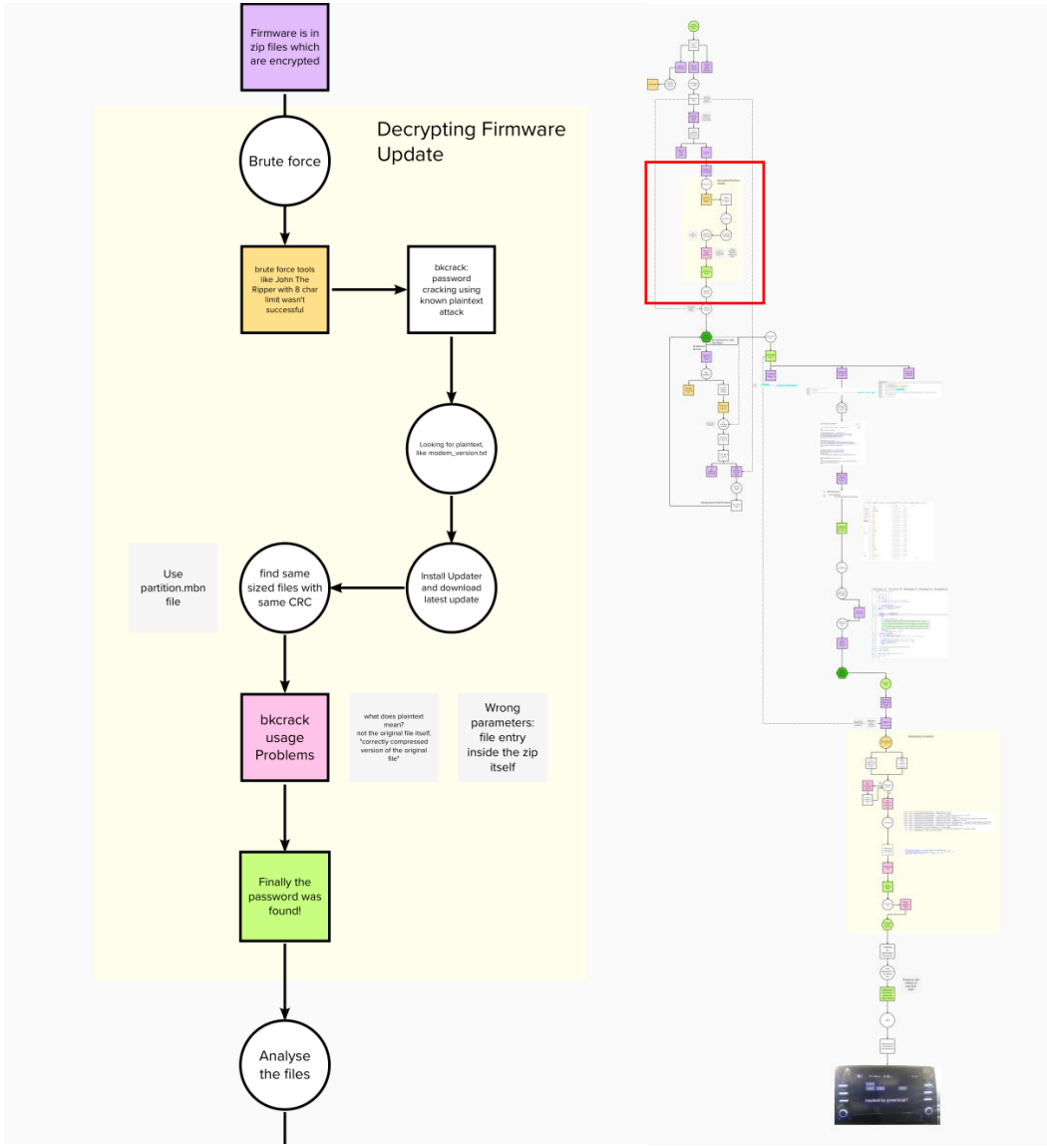
(3) Software and open source

- Software running on the in-vehicle infotainment system is D-Audio2V by Hyundai Mobis
- Search for opensource or downloads
  - Parts of the software are open source
  - Firmware updates are available, but encrypted



(4) Decrypting FW

- **Bruteforce with 8 characters failed.**
- Known plaintext attack in "*legacy zip encryption*" (PKWARE encryption) possible [1]
- Tool exists, yet the documentation/user instructions required trial and error and took a long time

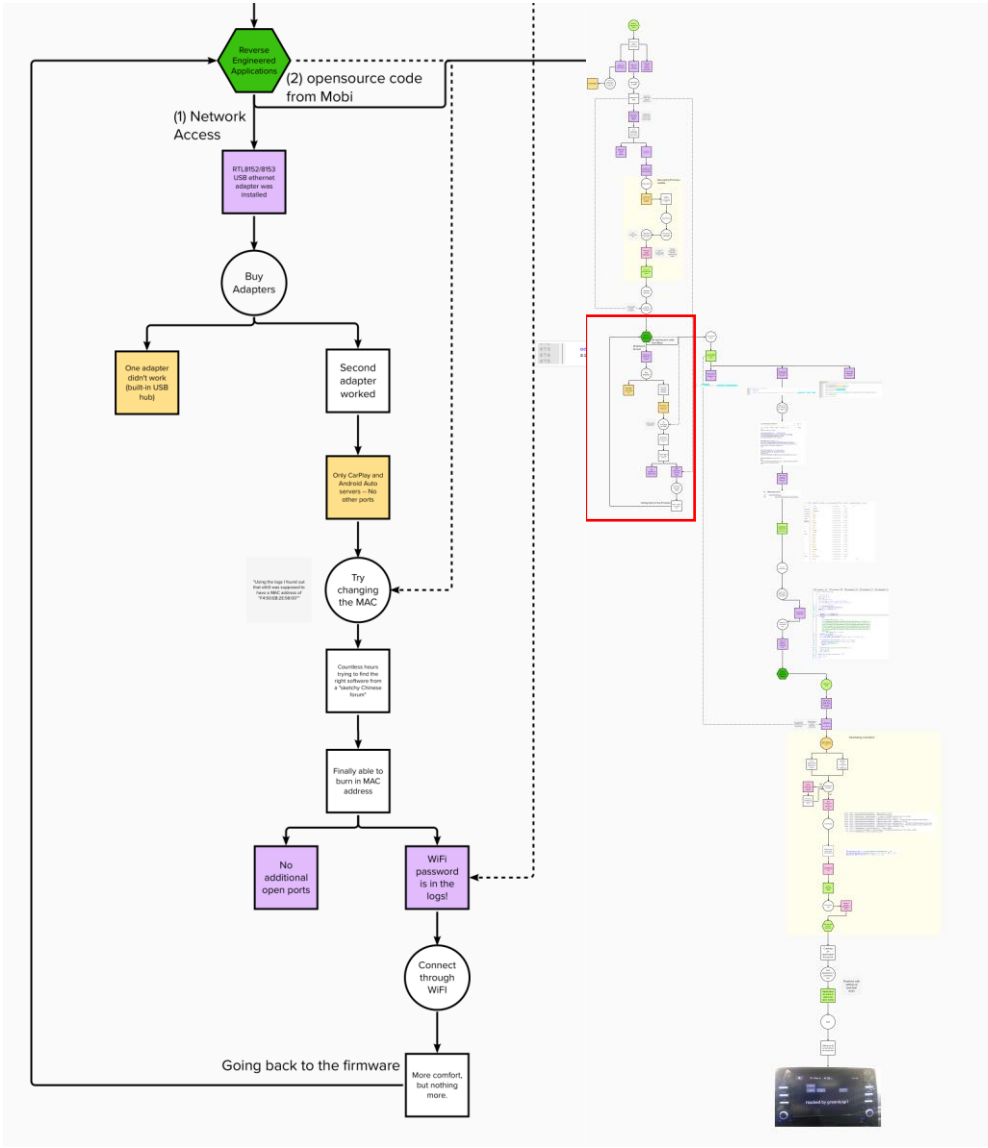


[1] Biham, E., Kocher, P.C. (1995). A known plaintext attack on the PKZIP stream cipher. In: Preneel, B. (eds) Fast Software Encryption. FSE 1994. Lecture Notes in Computer Science, vol 1008. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-60590-8_12

(5) Network Access

- Driver for USB ethernet adapter is installed
- Buy adapters and try ⌛
  - Requires specific MAC address
  - No additional ports available

M.2 WiFi password can be found in the logs!



(6) Open source code

M.3 The encryption key for the .zip software update file was used in production.

M.4.1 The key used to encrypt parts of the firmware update (enc_*updateboot.img*) in production shown in plain text.

(2) opensource code from Mobi

File contains the password for creating the zip file

```
cd ${UPDATE_DIR}
zip -P ahqltmTkrhk2018@@ -r
```

```
function generate_aes128_encryption()
{
    local FILE=$1
    local DIR=$2

    openssl aes-128-cbc -K 2b7e151628aed2a6abf7158809cf4f3c -iv
}
```

linux_envsetup.sh

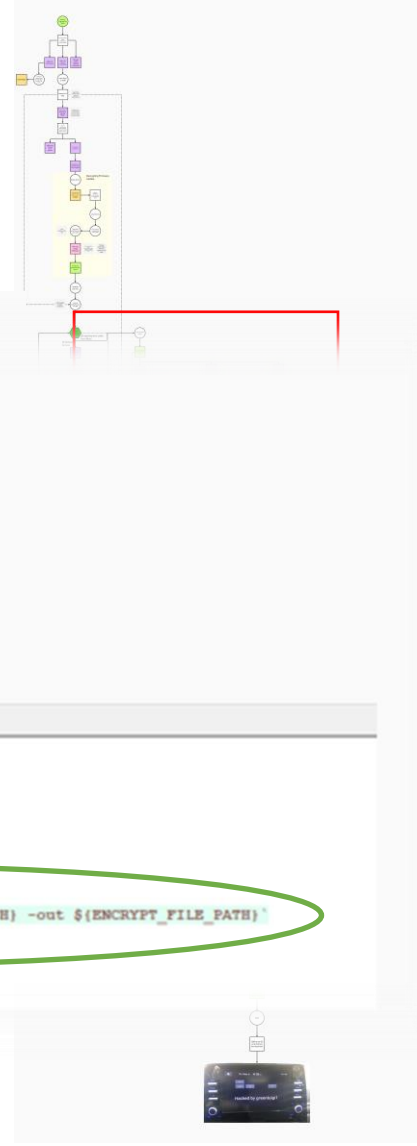
```
# update.list signing start
    local SYSTEM_SHA_PATH=$TARGET_DIR/sign/sha.txt
    local ENCRYPT_FILE_PATH=$UPDATE_DIR/update.info
    local PRI_KEY_PATH=$TARGET_DIR/sign/key.pri

    generate_sha224_hash ${UPDATE_LIST_PATH}
    echo "${SHA224_SUM} > ${SYSTEM_SHA_PATH}"

    RSA_STR='openssl rsautl -sign -in ${SYSTEM_SHA_PATH} -inkey ${PRI_KEY_PATH} -out ${ENCRYPT_FILE_PATH}'

    'rm ${SYSTEM_SHA_PATH}'

# update.list signing end
```



(6) Encryption keys

File contains also an encryption function, method, key and IV

```
function generate_aes128_encryption()
{
    local FILE=$1
    local DIR=$2

    openssl aes-128-cbc -K 2b7e151628aed2a6abf7158809cf4f3c -iv 000102030405060708090a0b0c0d0e0f -e -in ${DIR}/${FILE} -out ${DIR}/enc_${FILE}
}
```

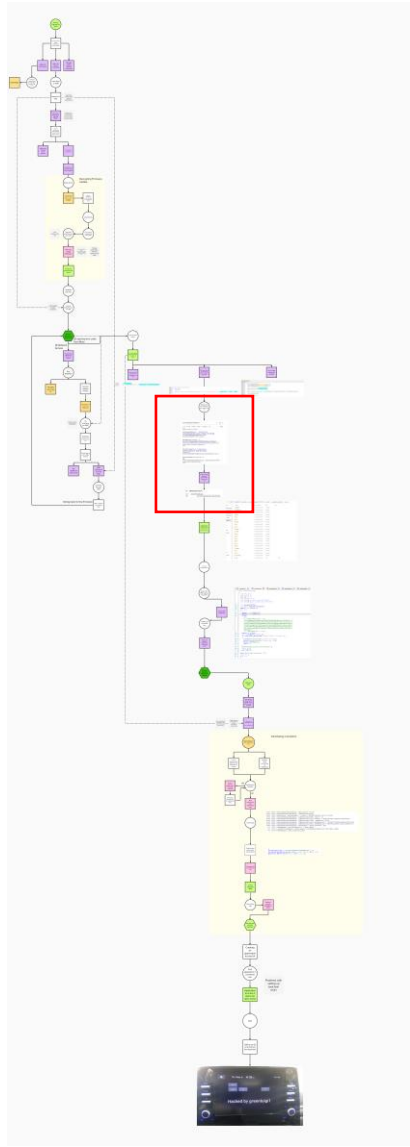
M.4.2 The hardcoded key was the example key used in many openssl tutorials.

F.2 CBC Example Vectors

F.2.1 CBC-AES128.Encrypt

Key 2b7e151628aed2a6abf7158809cf4f3c

IV 000102030405060708090a0b0c0d0e0f



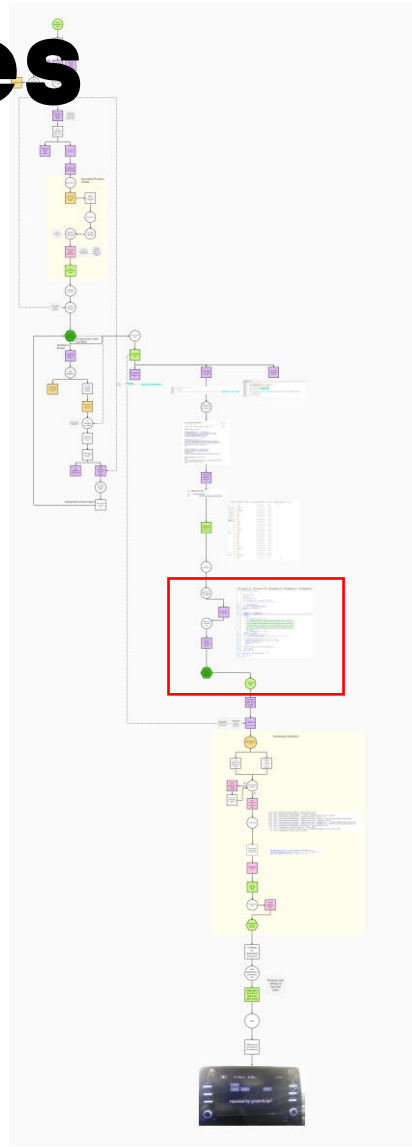
(7) Decrypting firmware images

- *updateboot.img* revealed the public key used to sign the firmware

M.5 OpenSSL example key was used. Therefore, the corresponding private key can be found easily.

IDA View-A | Pseudocode-E | Pseudocode-D | Pseudocode-A | Pseudocode-B

```
1 void *rsaDecryptHash()
2 {
3   void *v0; // r5
4   void *v1; // r4
5   int rsa; // r0
6   void *result; // r0
7   char s[256]; // [sp+Ch] [bp-2D4h] BYREF
8   char dest[468]; // [sp+10Ch] [bp-1D4h] BYREF
9
10  v0 = openUpdateInfo();
11  v1 = (void *)operator new[](57u);
12  memset(s, 0, sizeof(s));
13  if ( v0 )
14  {
15    memcpy(s, v0, sizeof(s));
16    memcpy(s, v0, sizeof(s));
17    strcpy(
18      dest,
19      "-----BEGIN PUBLIC KEY-----\n"
20      "MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAy8Dbv8prpJ/0kKhlGeJY\n"
21      "ozo2t60EG8L0561g13R29LvMR5hyvGZlGJpmn65+A4xHXInJYiPuKzrKUnApeLZ+\n"
22      "vw1HocOAZtWK0z3r26uA8kQYOKX9Qt/DbCdvsF9wF8gRK0ptx9M6R13NvBxvVQAp\n"
23      "fc9jB9nTzphOgM4JiEYvlV8FLhg9yZovMYd6Wwf3aoXK891VQxTr/kQYoq1Yp+68\n"
24      "i6T4nNq7NWC+UNVjQHxNQMQMzU6lWCX8zyg3yH88OAQkUXIXKfQ+NkvYQ1cxaMoV\n"
25      "PpY72+eVthKzpMeyHkBn7ciumk5qgLTEJAfWZpe4f4eFZj/Rc8Y8Jj2IS5kVPjUy\n"
26      "wQIDAQAB\n"
27      "-----END PUBLIC KEY-----\n");
28  memset(v1, 0, 56u);
29  rsa = createRSA((int)dest, 1);
30  if ( RSA_public_decrypt(256, (int)s, (int)v1, rsa, 1) != -1 )
31  {
32    printf("decrypt_hash:%s\n", (const char *)v1);
33    printf("decrypt_hash:%s\n", (const char *)v1);
34    operator delete[](v0);
35    return v1;
36  }
37  puts("RSA_public_decrypt fail decrypt!!");
38  result = v0;
39  goto LABEL_5;
40 }
41 puts("fail to get encryptText !!");
42 result = v1;
43 if ( v1 )
44 {
```



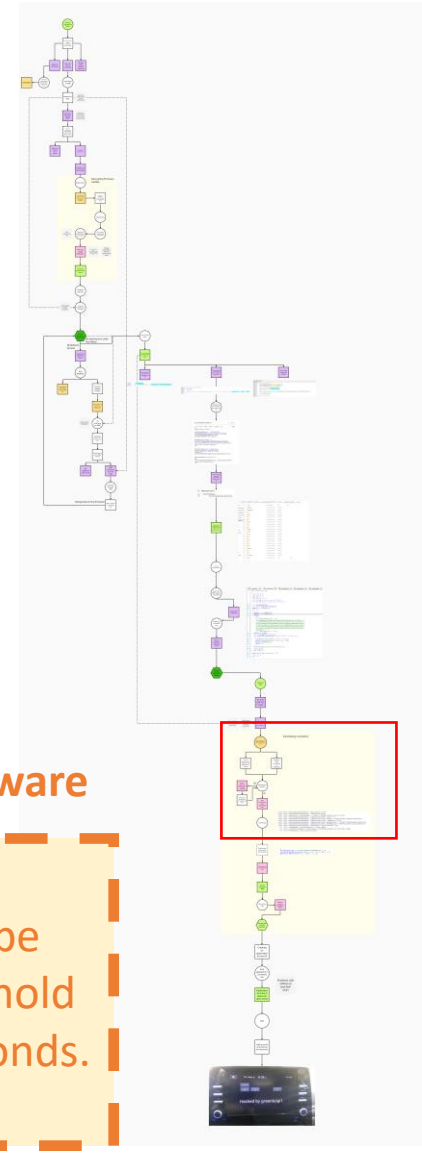
(8) Creating some backdoors

- 2 backdoors when using the Guider script that can be triggered in engineering mode
  - Execute script on pendrive when it is connected
  - Create reverse shell to fixed IP (192.168.0.2)
- 1st try, error during upgrade due to wrong hash (lead to constant reboot)
- 2nd try was successful
- **Pin to engineering mode changed!**

  logs + reverse engineering the new firmware revealed

  log revealed '02' + bruteforced md5 hash of 2 digit pin was hard coded in firmware

  md5 is also insecure



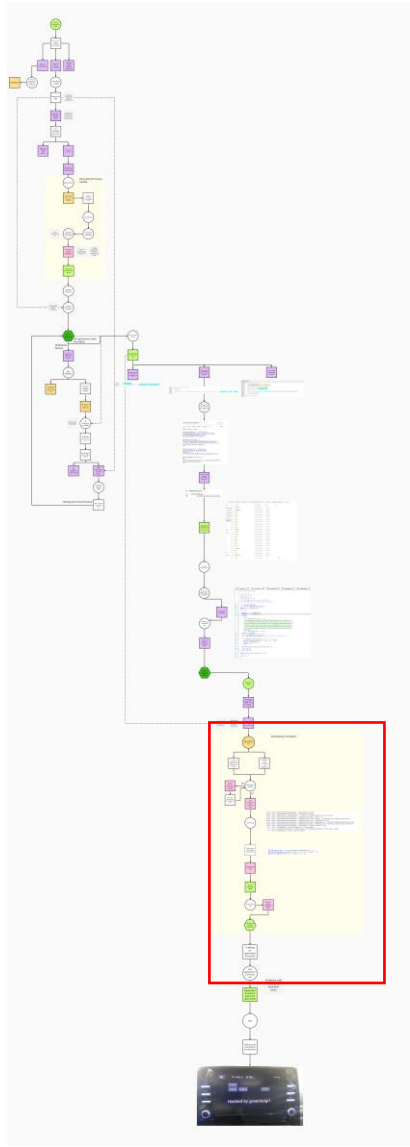
M.6 Log revealed first part of the pin

Logging to pendrive can be activated by, for example, hold the radio button for 30 seconds.

(9) Reverse shell

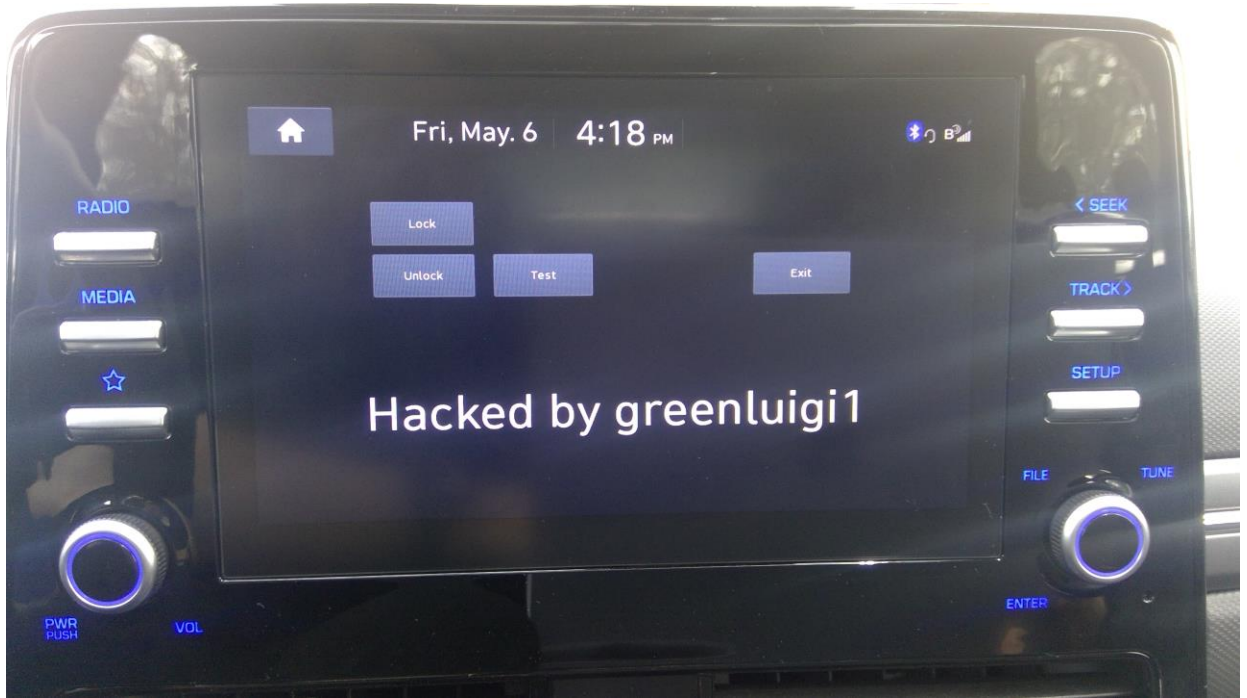
- Reverse shell executed through a script on the pendrive was successful
- Developing a command line application
  - Initial problems with using the libraries (e.g., versions) and setting up the tool chain
- First applications were possible to be executed via command line
  - E.g., checking whether the doors are closed or open

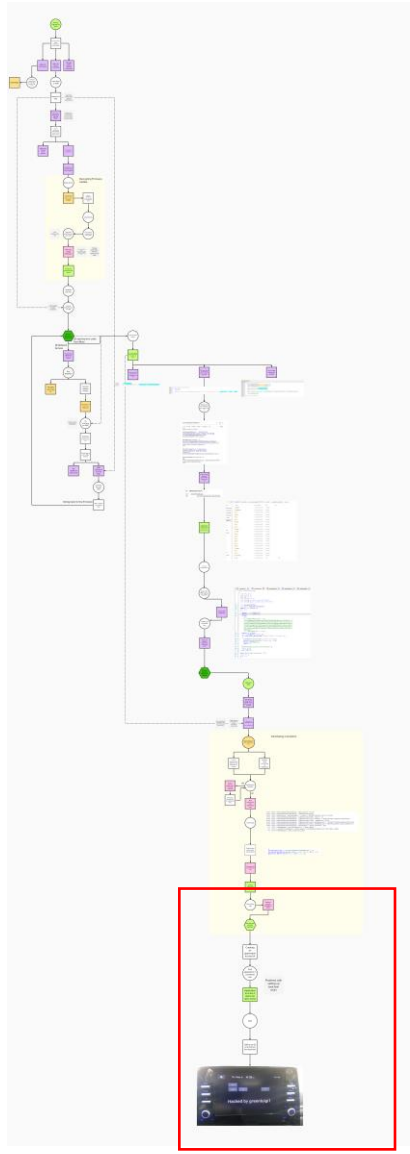
```
bash-3.2# ./test
./test
Ioniq Test Application
HLog D:GBusLIB(0:2425) gbus_client_thread() -
HLog D:GBusLIB(0:2425) gdbus_init_client() - com.mobis.canmanager_gdbus.service
HLog D:HBodyGDBus(0:2427) HBodyGDBus MSG_D 2022-06-12 13:38:19.093 HBodyGDBus.cpp:operator()          00086 HBodyGDBus SignalCallback thread start
HLog D:HBodyGDBus(0:2427) HBodyGDBus MSG_D 2022-06-12 13:38:19.094 HBodyGDBus.cpp:onSetSignalSubscribe  00135 subscribeSignal()
HLog D:HScanManagerGDBusClient(0:2423) startService() - isStart [1]. try count [2]
HLog D:HBodyImpl(0:2423) HBodyImpl MSG_D 2022-06-12 13:38:19.274 HBodyImpl.cpp:isDoorOpened          00617 isDoorOpened():
HLog D:HBodyImpl(0:2423) HBodyImpl MSG_D 2022-06-12 13:38:19.274 HBodyImpl.cpp:getDoorOpenStatusCapi  00716 getDoorOpenStatusCapi():
HLog D:HBodyGDBus(0:2423) HBodyGDBus MSG_D 2022-06-12 13:38:19.275 HBodyGDBus.cpp:isDoorOpened        01724 isDoorOpened->remotetype : 2
HLog D:HBodyGDBus(0:2423) HBodyGDBus MSG_D 2022-06-12 13:38:19.275 HBodyGDBus.cpp:isDoorOpened        01725 isDoorOpened->remotedata : 4
HLog D:HBodyGDBus(0:2423) HBodyGDBus MSG_D 2022-06-12 13:38:19.282 HBodyGDBus.cpp:isDoorOpened        01733 isDoorOpened result : 2
HLog D:HBodyGDBus(0:2423) HBodyGDBus MSG_D 2022-06-12 13:38:19.282 HBodyGDBus.cpp:isDoorOpened        01736 SUCCESS
HLog D:HBodyGDBus(0:2423) HBodyGDBus MSG_D 2022-06-12 13:38:19.282 HBodyGDBus.cpp:isDoorOpened        01741 HBodyGDBus->fRet : 0
Door Result: Open
Finished door test
HLog D:HScanManagerGDBusClient(0:2423) HScanManagerGDBusClient::isConnected mbConnection : true
HLog D:HBodyGDBus(0:2427) HBodyGDBus MSG_D 2022-06-12 13:38:19.330 HBodyGDBus.cpp:operator()          00095 HBodyGDBus SignalCallback thread done
bash-3.2#
```



(10) GUI

- Qt Creator and setup of the toolchain
- Helix as a application manager
- Configuration files





Pitfalls/Mistakes

M.1	Often 4-digit pin to guess (default, easy depends on year, or time + 6 last digits of SW version)	*Who should access it? Are individual keys or public key crypto possible?*
M.2	WiFi password can be found in the logs!	*Who should have access to the logs? (-> M.6)*
M.3	The encryption key for the .zip software update file was used in production.	*Secret keys should be marked and removed before published open source.*
M.4	The hardcoded key was the example key used in many openssl tutorials.	*Example keys are for examples!*
M.5	OpenSSL example key was used. Therefore, the corresponding private key can be found easily.	*Example keys are for examples!*
M.6	log revealed first part of the pin (e.g., holding radio button pressed for 30s triggers logging to pendrive)	*Encrypt logs*

An example of what recently happened during an attack on a specific car brand

A walk through of the pitfalls based on examples

Thomas Rosenstatter, Research in Cybersecurity @RISE

Sources

- Screenshots of the attack are taken from the blog written by *greenluigi1* (https://programmingwithstyle.com/)
- Detailed documentation about the hack can be found in https://programmingwithstyle.com/
- [1] Biham, E., Kocher, P.C. (1995). A known plaintext attack on the PKZIP stream cipher. In: Preneel, B. (eds) Fast Software Encryption. FSE 1994. Lecture Notes in Computer Science, vol 1008. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-60590-8_12