Write-Ups Retinal System - USB Drive Eyes Infection

Author : Choupisson

Message

Ripple effects in our industry are common. They are in the human body as well, who would have guessed.

Philip was annoyed of losing eyelashes. Long story short, he got scammed and got a cream for that, which infected his eyes, and now the Universal Sight Bridge linking his retina to the brain got impacted and he got an infection. Result? He lost color sighting. I guess that explain why he didn’t take action when our last pentest report was full of red everywhere.

Anyway. Here’s the data flux coming from his cones. See if you can find what happened.

Humm quite a big problem. Let's fight against M. Wellington's eye infection right now!

Step 1 - PCAP

First things first, we start with a pcap file containing USB frame.

Appliquer un filtre d'affichage ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Value
13	2024-01-10 00:49:02,864340	host	1.7.0	USB	36	
14	2024-01-10 00:49:02,864718	1.7.0	host	USB	60	
15	2024-01-10 00:49:02,870391	host	1.7.0	USB	36	
16	2024-01-10 00:49:02,871195	1.7.0	host	USB	28	
17	2024-01-10 00:49:02,871270	host	1.7.0	USB	36	
18	2024-01-10 00:49:02,871904	1.7.0	host	USB	28	
19	2024-01-10 00:49:02,871920	host	1.7.0	USBMS	36	
20	2024-01-10 00:49:02,872046	1.7.0	host	USBMS	29	
21	2024-01-10 00:49:02,872131	host	1.7.2	USBMS	58	
22	2024-01-10 00:49:02,872199	1.7.2	host	USB	27	
23	2024-01-10 00:49:02,872206	host	1.7.1	USB	27	
24	2024-01-10 00:49:02,872395	1.7.1	host	USBMS	63	
25	2024-01-10 00:49:02,872403	host	1.7.1	USB	27	
26	2024-01-10 00:49:02,872454	1.7.1	host	USBMS	40	

By inspecting the first USB frames, we can identify in the DEVICE DESCRIPTOR field that the equipment used in the capture is a Sony MicroVault USB Flash Drive.

```
> Frame 2: 46 bytes on wire (368 bits), 46 bytes captured (368 bits) on interface \\.\USBPcap1, id 0
> USB URB
v DEVICE DESCRIPTOR
    bLength: 18
    bDescriptorType: 0x01 (DEVICE)
    bcdUSB: 0x0200
    bDeviceClass: Device (0x00)
    bDeviceSubClass: 0
    bDeviceProtocol: 0 (Use class code info from Interface Descriptors)
    bMaxPacketSize0: 64
    idVendor: Sony Corp. (0x054c)
    idProduct: MicroVault Flash Drive (0x0243)
    bcdDevice: 0x0200
    iManufacturer: 1
    iProduct: 2
    iSerialNumber: 3
    bNumConfigurations: 1
```

After some USB initialization packets, we can remark a lot of USBMS (USB Mass Storage) packets which is a standard protocol for accessing external disks over USB. Now we have a clear overview of the situation and we can expect to recover some data transfer via the USB stick from the capture.

After some research, I find on this site a quite similar situation and some helpful information about USBMS data. To extract the data transfers through USB we have to focus on the SCSI and specifically on the Read fields. In those, the LBA field is the logical block address into the 512-byte blocks and the data field is the transferred data.

So I made some changes in the method to fit our situation and go to the practical ! 🚀

Step 1: make a JSON dump of the specific USBMS packet using Tshark.

```
tshark -T json -x -Y usbms -r eyes-infection.pcapng > usbms.json
```

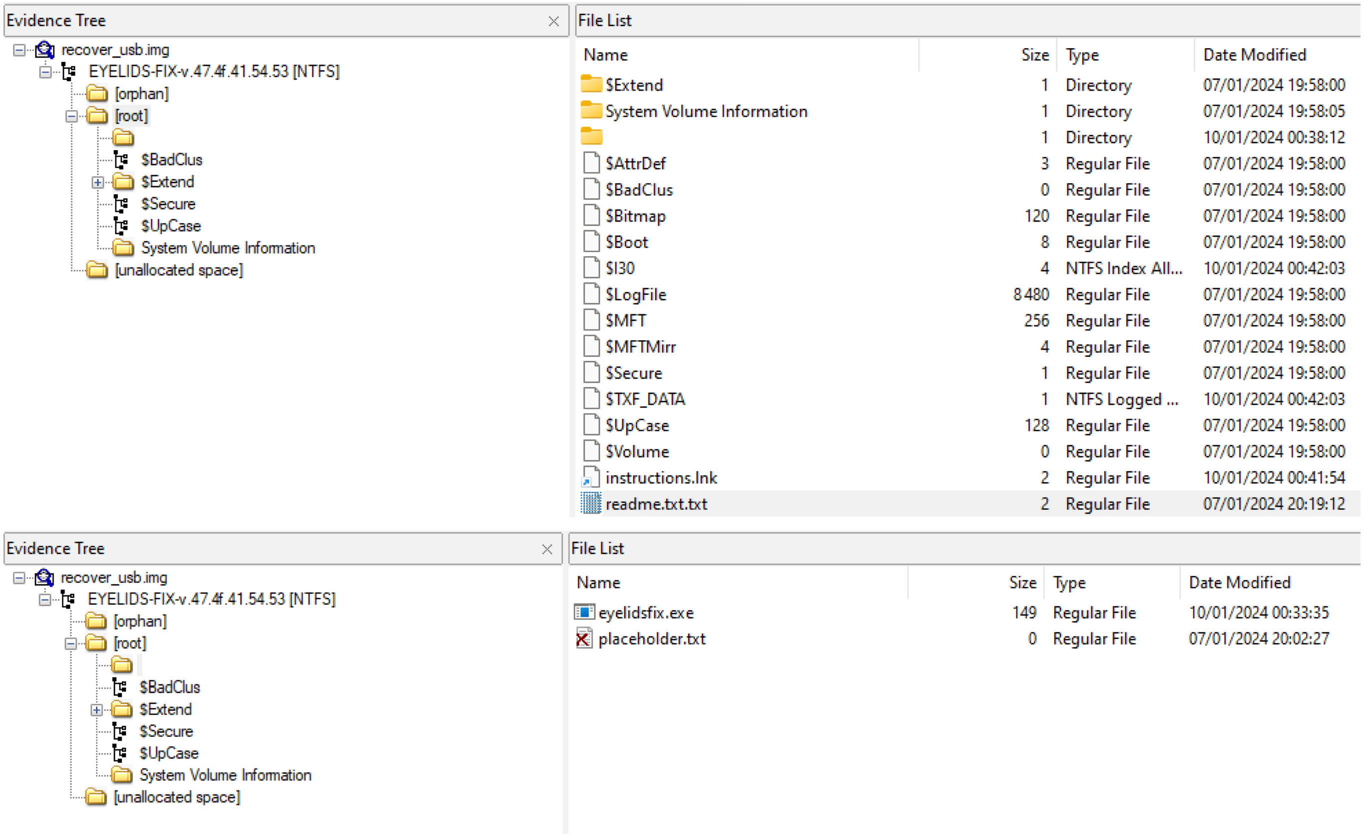
Step 2: write a Python script, that reads the JSON dump file, selects the good frames, and writes the data transfer in an output file.

recover_usbms_data.py

```
#!/usr/bin/env python3
from scapy.all import *
import os
import json

dir_path = os.path.dirname(__file__)
usbms_path = os.path.join(dir_path, "usbms.json")
```

```
#Read each frame from usbms dump
usbms = json.load(open(usbms_path))
frames = {int(frame["_source"]["layers"]["frame"]["frame.number"]):frame for
frame in usbms}

#Create the out image file
file_path = os.path.join(dir_path, "recover_usb.img")
out = open(file_path, 'wb')

for framenr in sorted(frames):
    frame = frames[framenr]
    #Select specific scsi frames to bypass initialization or useless frames
    if "scsi_raw" in frame["_source"]["layers"] and "scsi_sbc.opcode_raw" not
in frame["_source"]["layers"]["scsi"] and "scsi.request_frame" in
frame["_source"]["layers"]["scsi"]:
        reqframe = frames[int(frame["_source"]["layers"]["scsi"]
["scsi.request_frame"])]
        opcode_str = frame["_source"]["layers"]["scsi"]["scsi_sbc.opcode"]
        opcode = int(opcode_str, 16)
        if opcode not in (40, 42):
            continue

        #Extract lba and data of the frame
        lba = int(reqframe["_source"]["layers"]["scsi"]
["scsi_sbc.rdwr10.lba"])
        reqlen = int(reqframe["_source"]["layers"]["scsi"]
["scsi_sbc.rdwr10.xferlen"])
        data = bytes.fromhex(frame["_source"]["layers"]["scsi_raw"][0])
        print(framenr, opcode, lba, reqlen, data[:512].hex())

        #Write extract data to the output file
        out.seek(lba * 512)
        out.write(data)
```

Once we execute these two steps, we can try to open the image file in FTK Imager, and ...
Magic it works !



We have the name of the USB volume "EYELIDS-FIX-v.47.4f.41.54.53" and diving deep inside we find in the root directory a "readme.txt.txt" file with some interesting information for M. Wellington.

```
You are sick and tired of the inconvenience with the eyelash mechanism,
install this patch to never be blinded by rogue eyelashes landing in your
visual sensor.

To upgrade the system and prevent eyelashes from getting in your visual
sensors, follow these steps:

1. Assessment:
Figure out what's going on with the eyelash mechanism.

2. Backup:
Save all the important stuff related to visual sensors.

3. Shutdown:
Turn the sensor off safely.

4. Maintenance Mode:
Access the internal systems.
```

```
5. Upgrade:
Install the latest software patch for eyelash optimization. This is included
in this upgrade kit.
Navigate to the automatic installation process instructions.

6. Calibration:
Make sure everything fits together smoothly.

7. Reboot and Test:
Turn the sensor back on and check if the upgrade did the trick.

8. Preventive Measures:
Add in some extra steps to prevent future eyelash trouble.

9. User Manual Update:
Let the user know about the fancy new upgrade.

10. Documentation:
Keep records of what you did for future reference.
```

And most importantly a PE exe file named "eyelidsfix.exe" that we can extract for further analysis.

Okay okay that's the clever method, but what if I told you it's possible to recover the binary just by performing a binwalk on the pcap file 😅.

```
binwalk --dd=".*" eyes-infection.pcapng
```

But don't be too mad, with the first method you will have more easily some useful information 😉.

Anyway, the "eyelidsfix.exe" file seems like a good way to cure M. Wellington ! Hurry up !

Properly extract data from
the USBMS frame like a pro

Execute random binwalk
on each file of the CTF

Step 2 - Reverse

First, I execute the program into my sandbox VM but nothing really happened. I see a terminal pop up and quit instantly.

I try to get the strings of the file with :

```
strings eyelidsfix.exe
```

Nothing that much interesting either except some strings related to DotNet file.

Thinking back to my old CTFs I try to extract utf16-le strings and finally, I get something !

```
strings -e l eyelidsfix.exe
```

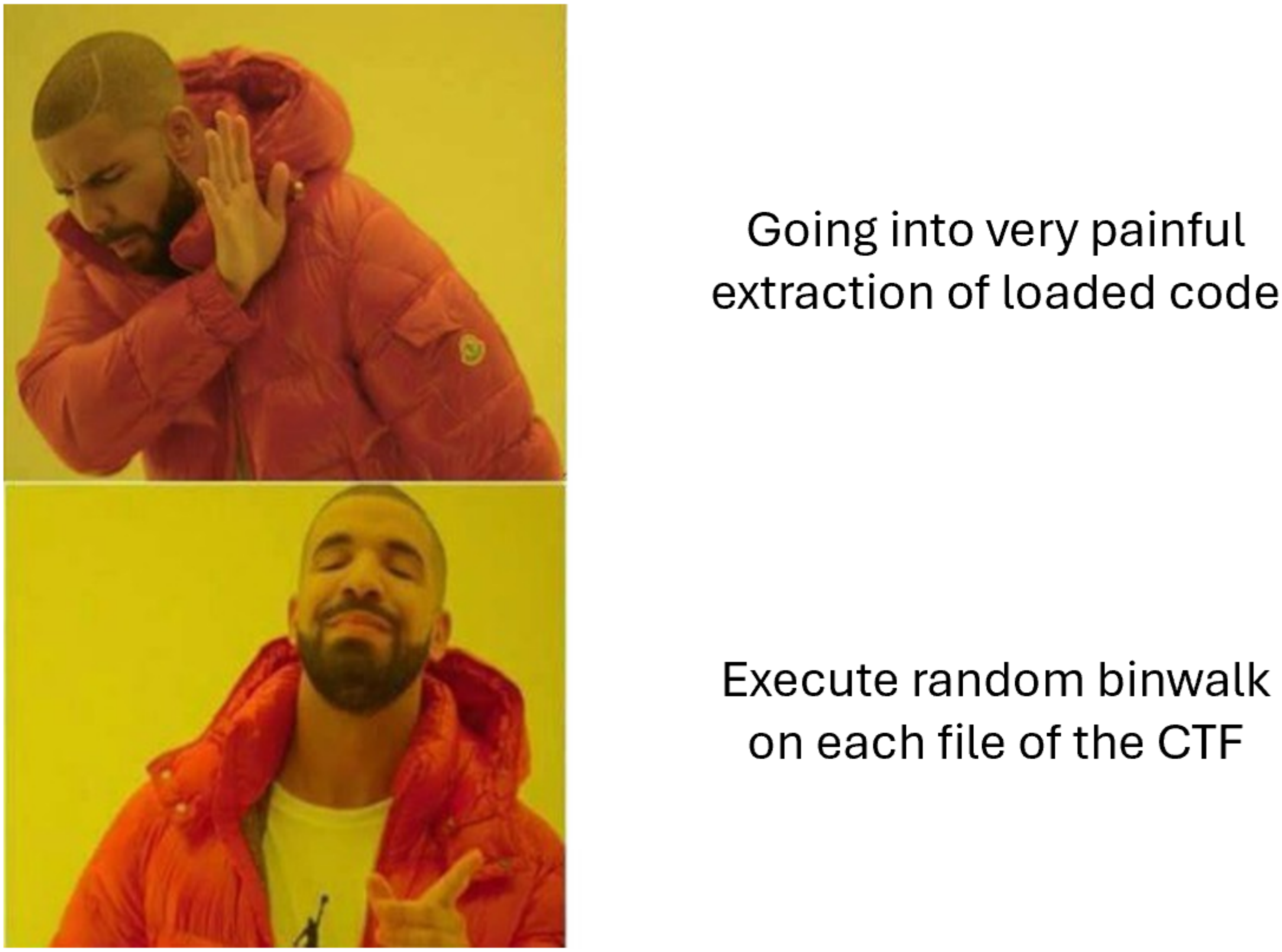
```
eyelidsfix.dll
LegalCopyright
OriginalFilename
eyelidsfix.dll
ProductName
eyelidsfix
ProductVersion
1.0.0
Assembly Version
1.0.0.0
[-] Starting up EYELIDS FIX...
[-] Gathering eyes firmware...
[+] Version compatible with fix.
[-] Shutting down device to apply fix...
[-] Patching eyes firmware...
HKEY_CURRENT_USER\SOFTWARE\Microsoft\ColorFiltering
Active
FilterType
[+] Patch applied successfully.
[*] Your cones have been disabled in this trial version.
[*] To unlock colour vision, please upgrade to the premium firmware.
[X] An error has occured applying the eyes patch.
[-] Attempting to upgrade to premium...
[+] The program is running from an approved media.
[-] Attempting to unlock premium firmware features...
FLAG
[+] The premium firmware have been activated. Activate online to receive instructions on enabling the cone receptors.
[+] Activation code:
[X] Unable to activate the premium firmware. Generated activation code is invalid.
[+] Activation Code:
[X] The program is not running from an approved media.
[-] Shutting down EYELIDS FIX...
VS_VERSION_INFO
```

To confirm it I run `capa` on the exe and I get the following capabilities :

```
capa eyelidsfix.exe
```

Capability	Namespace
contains PDB path	executable/pe/pdb
contain an embedded PE file	executable/subfile/pe
query environment variable (7 matches)	host-interaction/environment-variable
get common file path	host-interaction/file-system
print debug messages	host-interaction/log/debug/write-event
create process on Windows (2 matches)	host-interaction/process/create
query or enumerate registry value	host-interaction/registry
link many functions at runtime	linking/runtime-linking
parse PE header	load-code/pe

It seems that our "eyelidsfix.exe" is loading some code and we want to extract it. So, I run binwalk on the exe file and this time I got everything that I needed. 🙃.



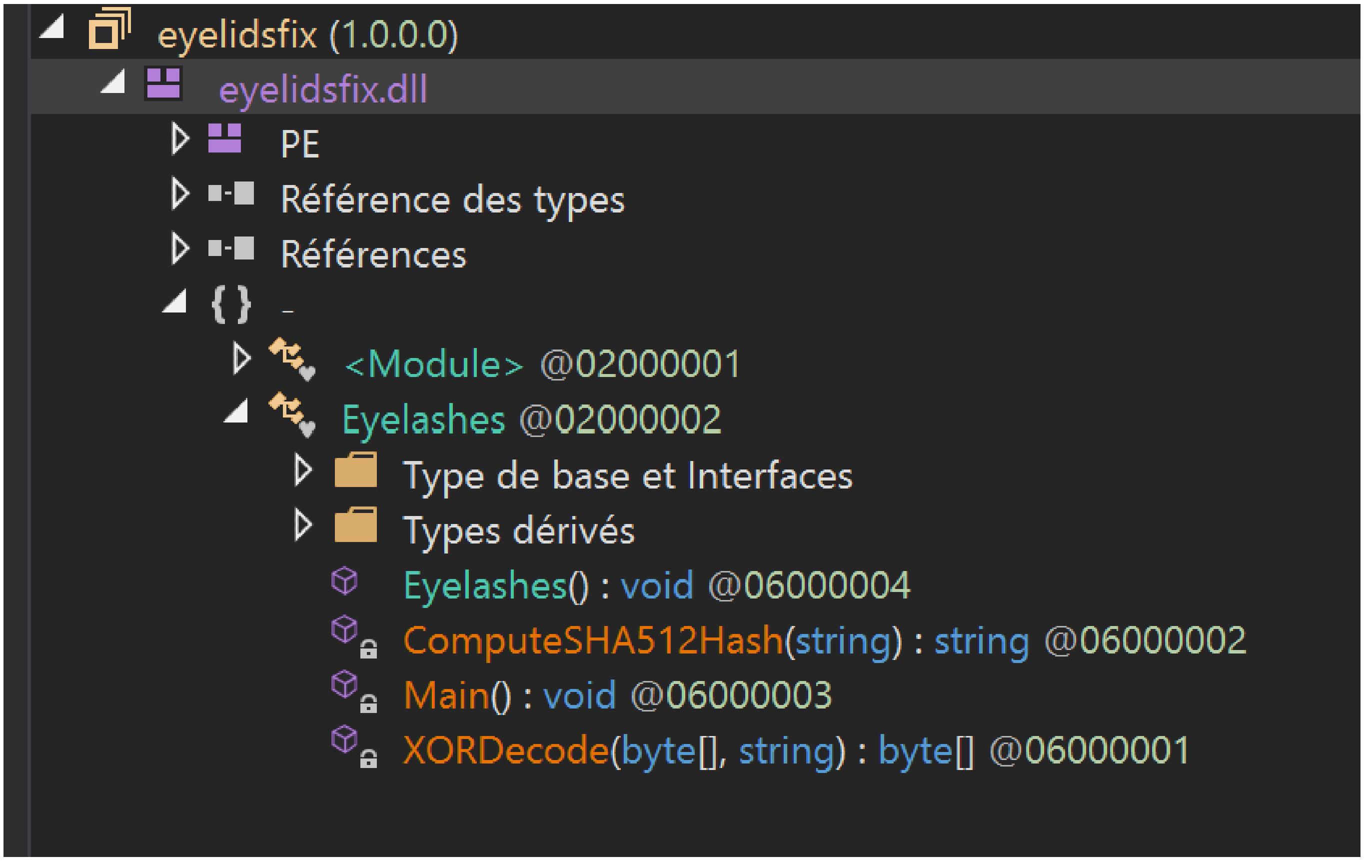

```
binwalk --dd=".*" eyelidsfix.exe
```

```
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             Microsoft executable, portable (PE)
140119        0x22357         XML document, version: "1.0"
143360        0x23000         Microsoft executable, portable (PE)
150391        0x24B77         XML document, version: "1.0"
```

Performing `file` on them I identify the DotNet one.

```
_eyelidsfix.exe.extracted/0:     PE32+ executable (console) x86-64, for MS Windows
_eyelidsfix.exe.extracted/22357: data
_eyelidsfix.exe.extracted/23000: PE32+ executable (console) x86-64 Mono/.Net assembly, for MS Windows
_eyelidsfix.exe.extracted/24B77: data
```

So I open it in dnSpy.



The file is a dll named "eyelidsfix.dll". We can see some interesting functions but first, let's go into the 'Main' function.

Main() : void

```
// Eyelashes
// Token: 0x06000003 RID: 3 RVA: 0x00002108 File Offset: 0x00000308
private static void Main()
{
    Console.WriteLine("[-] Starting up EYELIDS FIX...");
    Thread.Sleep(1000);
    Console.WriteLine("[-] Gathering eyes firmware...");
    Thread.Sleep(1000);
    Console.WriteLine("[+] Version compatible with fix.");
    Thread.Sleep(500);
    Console.WriteLine("[-] Shutting down device to apply fix...");
    Thread.Sleep(1000);
    Console.WriteLine("[-] Patching eyes firmware...");
    Thread.Sleep(2000);
    try
    {
        string keyName = "HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\ColorFiltering";
        Registry.SetValue(keyName, "Active", 1, RegistryValueKind.DWord);
        Registry.SetValue(keyName, "FilterType", 1, RegistryValueKind.DWord);
        Console.WriteLine("[+] Patch applied successfully.");
        Console.WriteLine("[*] Your cones have been disabled in this trial version.");
        Console.WriteLine("[*] To unlock colour vision, please upgrade to the premium firmware.");
    }
    catch (Exception)
    {
        Console.WriteLine("[X] An error has occured applying the eyes patch.");
    }
    Thread.Sleep(1000);
    Console.WriteLine("[-] Attempting to upgrade to premium...");
    Thread.Sleep(1000);
    DriveInfo driveInfo = new DriveInfo(Path.GetPathRoot(AppDomain.CurrentDomain.BaseDirectory));
    if (driveInfo.DriveType == DriveType.Removable)
    {
        Console.WriteLine("[+] The program is running from an approved media.");
        string volumeLabel = driveInfo.VolumeLabel;
        byte[] inputBytes = new byte[]
        {
            37,
            46,
            112,
            34,
            30,
            127,
            7,
            120,
            44,
            33,
            125,
            8,
            51,
            53,
            102,
            109,
            54,
            38,
            105,
            47,
            120,
            114,
            11,
            39,
            99,
            82,
            32,
            62,
            40,
            56,
            61,
            112,
            42,
            47,
            36,
            50
        };
        Console.WriteLine("[-] Attempting to unlock premium firmware features...");
        Thread.Sleep(1000);
        string key = Eyelashes.ComputeSHA512Hash(volumeLabel);
        byte[] bytes = Eyelashes.XORDecode(inputBytes, key);
        string @string = Encoding.UTF8.GetString(bytes);
        if (@string.StartsWith("FLAG"))
        {
            Console.WriteLine("[+] The premium firmware have been activated. Activate online to receive instructions on enabling the cone receptors.");
            Console.WriteLine("[+] Activation code: " + @string);
        }
        else
        {
            Console.WriteLine("[X] Unable to activate the premium firmware. Generated activation code is invalid.");
            Console.WriteLine("[+] Activation Code: " + @string);
        }
    }
    else
    {
        Console.WriteLine("[X] The program is not running from an approved media.");
    }
    Console.WriteLine("[-] Shutting down EYELIDS FIX...");
}
```

The program is quite simple. We see easily that to get the FLAG, we have to complete this little challenge.

```
string key = Eyelashes.ComputeSHA512Hash(volumeLabel);
byte[] bytes = Eyelashes.XORDecode(inputBytes, key);
string @string = Encoding.UTF8.GetString(bytes);
if (@string.StartsWith("FLAG"))
```

The two functions 'ComputeSHA512Hash' and 'XORDecode' do nothing but their name, so it's just basically a function that computes SHA512 hash and a function that performs xoring operation between a byte array and a key.

The volumeLabel variable reminds me of the USB volume name that we saw earlier in FTK Imager, so I just transcript the useful part of the script in Python and give "EYELIDS-FIX-v.47.4f.41.54.53" as volumeLabel entry.

recover_code.py

```python
#!/usr/bin/env python3
import hashlib

def xor_decode(input_bytes, key):
    key_bytes = key.encode('utf-8')
    decoded_bytes = bytearray(input_bytes)
    for i in range(len(input_bytes)):
        decoded_bytes[i] ^= key_bytes[i % len(key_bytes)]
    return decoded_bytes


def compute_sha512_hash(input_str):
    sha512 = hashlib.sha512()
    sha512.update(input_str.encode('utf-8'))
    return sha512.hexdigest()


def recover_activation_code(volume_label):
    input_bytes = [
        37, 46, 112, 34, 30, 127, 7, 120, 44, 33, 125, 8, 51, 53, 102, 109,
        54, 38, 105, 47, 120, 114, 11, 39, 99, 82, 32, 62, 40, 56, 61, 112,
        42, 47, 36, 50
    ]

    key = compute_sha512_hash(volume_label)
    decoded_bytes = xor_decode(input_bytes, key)
```

```
    activation_code = decoded_bytes.decode('utf-8')
    return activation_code

if __name__ == "__main__":
    volume_label = "EYELIDS-FIX-v.47.4f.41.54.53"
    activation_code = recover_activation_code(volume_label)
    print(f"Premium firmware activation code: {activation_code}")
```

And Boom !

```
Premium firmware activation code: FLAG-M4LICI0US_USB_INF3CT3D_MY_CONES
```

Message

While our host is still under sleep, I can confirm that transmission of color coding has been restored. Dopamine secretion will be restored when his favorite color, green, will be perceived on dashboards.

System repaired.


GOOD JOB