

HackTM CTF 2020 Writeup


Since the CTF is still active I won't be dropping the flags. You can follow along and complete the challenges for yourself here: <https://ctfx.hacktm.ro/>

0x01 Strange PCAP

Strange PCAP 144 Points

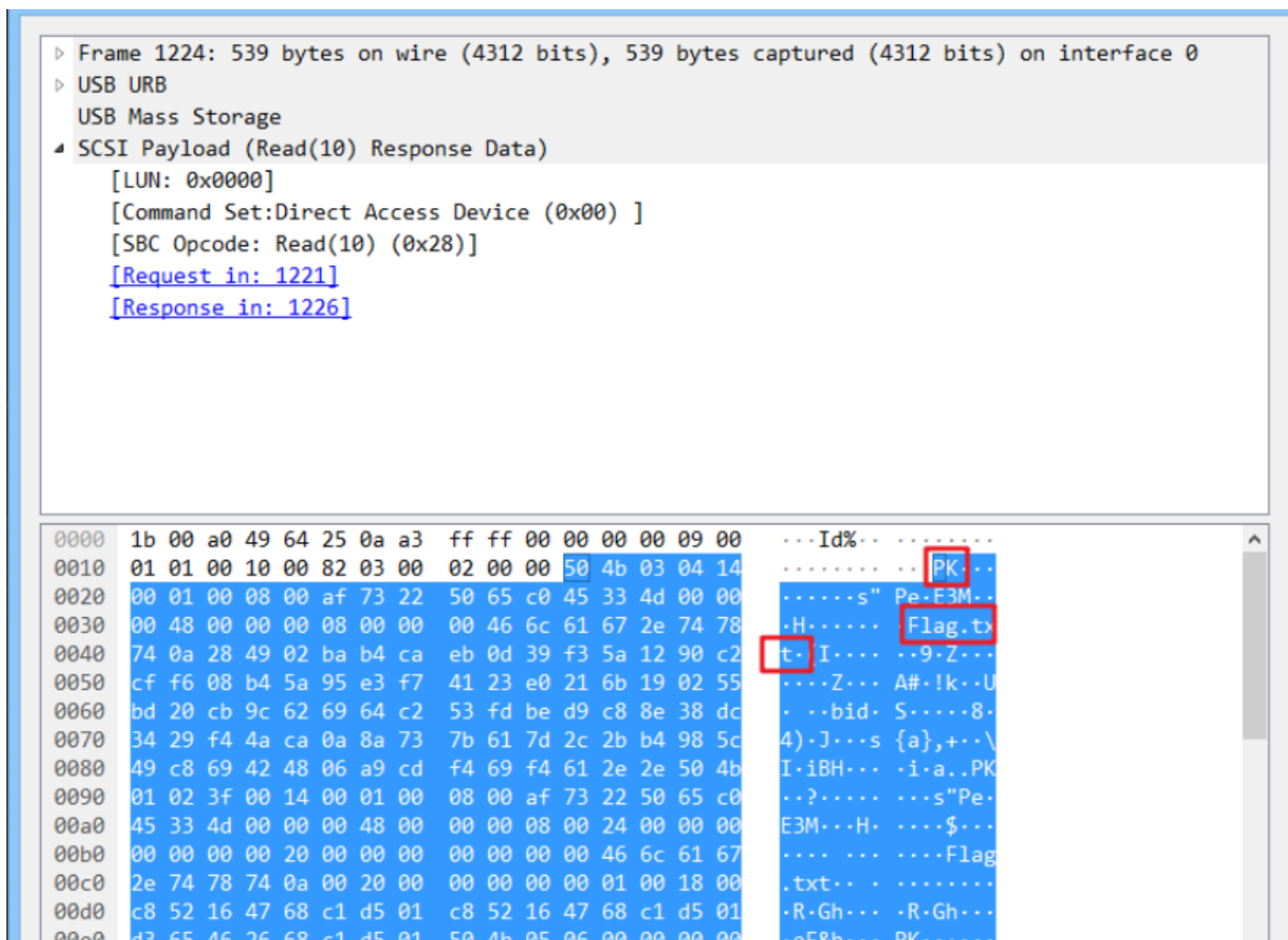
We managed to get all the data to incriminate our CEO for selling company secrets. Can you please help us and give us the secret data that he has leaked?

Author: Legacy

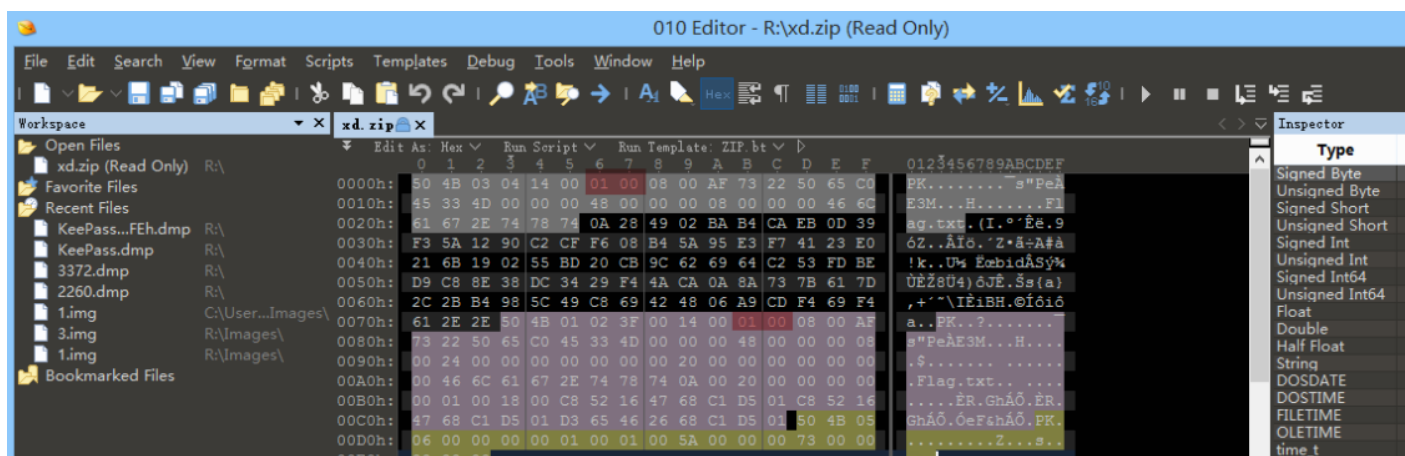
 Strange.pcapng ccd21d48fd04137551833d2e4493243e

A basic PCAP forensics question. When you get the file, you will find that there are many USB Massive Storage packages. The initial guess is to find the flag in it.

| File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help | | | | | | |
|--|-------------------------|-------------------------|-------------|----------|--------|---|
| Apply a display filter ... <Ctrl-/> | | | | | | |
| No. | Time | Source | Destination | Protocol | Length | Info |
| 301 | 7.352045 | 1.16.2 | host | USBMS | 35 | SCSI: Data In LUN: 0x00 (Read Capacity(10) Response Data) |
| 302 | 7.352046 | host | 1.16.2 | USB | 27 | URB_BULK in |
| 303 | 7.352101 | 1.16.2 | host | USBMS | 40 | SCSI: Response LUN: 0x00 (Read Capacity(10)) (Good) |
| 304 | 7.352125 | host | 1.16.1 | USBMS | 58 | SCSI: Read(10) LUN: 0x00 (LBA: 0x00000000, Len: 1) |
| 305 | 7.352156 | 1.16.1 | host | USB | 27 | URB_BULK out |
| 306 | 7.352158 | host | 1.16.2 | USB | 27 | URB_BULK in |
| 307 | 7.352714 | 1.16.2 | host | USBMS | 539 | SCSI: Data In LUN: 0x00 (Read(10) Response Data) |
| 308 | 7.352728 | host | 1.16.2 | USB | 27 | URB_BULK in |
| 309 | 7.352849 | 1.16.2 | host | USBMS | 40 | SCSI: Response LUN: 0x00 (Read(10)) (Good) |
| 310 | 7.352939 | host | 1.16.1 | USBMS | 58 | SCSI: Read(10) LUN: 0x00 (LBA: 0x00000800, Len: 16) |
| 311 | 7.353003 | 1.16.1 | host | USB | 27 | URB_BULK out |
| 312 | 7.353005 | host | 1.16.2 | USB | 27 | URB_BULK in |
| 313 | 7.353832 | 1.16.2 | host | USBMS | 8219 | SCSI: Data In LUN: 0x00 (Read(10) Response Data) |
| 314 | 7.353943 | host | 1.16.2 | USB | 27 | URB_BULK in |
| 315 | 7.354004 | 1.16.2 | host | USBMS | 40 | SCSI: Response LUN: 0x00 (Read(10)) (Good) |
| 316 | 7.355069 | host | 1.16.1 | USBMS | 58 | SCSI: Read(10) LUN: 0x00 (LBA: 0x00000800, Len: 16) |
| 317 | 7.355137 | 1.16.1 | host | USB | 27 | URB_BULK out |
| 318 | 7.355138 | host | 1.16.2 | USB | 27 | URB_BULK in |
| 319 | 7.356055 | 1.16.2 | host | USBMS | 8219 | SCSI: Data In LUN: 0x00 (Read(10) Response Data) |
| > Frame 1: 36 bytes on wire (288 bits) 36 bytes captured (288 bits) on interface wireshark-ethcan1920 id 0 | | | | | | |
| 0000 | 1c 00 00 00 00 00 00 00 | 00 00 00 00 00 00 0b 00 | | | | |
| 0010 | 00 01 00 01 00 80 02 08 | 00 00 00 00 80 06 00 01 | | | | |
| 0020 | 00 00 12 00 | | | | | |



Looking through the data packets one by one, I found the ZIP file header and Flag.txt in Frame 1224, extract them and have a look.



After extracting the data, it is found to be an encrypted compressed package. Check it with 010 Editor and find that it is really encrypted. Next, we have to find the password. Let's make a preliminary guess about the HID package sent by the keyboard because we have seen the USB HID data package before Types of.

| No. | Time | Source | Destination | Protocol | Length | Leftover Capture Data | Info |
|------|-----------|--------|-------------|----------|--------|-----------------------|---|
| 1327 | 37.973014 | 1.15.0 | host | USBHID | 28 | | SET_REPORT Response |
| 1328 | 37.973471 | 1.15.0 | host | USBHID | 30 | | GET_REPORT Response |
| 1329 | 37.973535 | 1.15.3 | host | USB | 30 | 210012 | URB_INTERRUPT in |
| 1332 | 37.973585 | 1.15.1 | host | USB | 35 | 0000240000000000 | URB_INTERRUPT in |
| 1334 | 37.974323 | 1.15.0 | host | USBHID | 28 | | SET_REPORT Response |
| 1336 | 37.974789 | 1.15.0 | host | USBHID | 30 | | GET_REPORT Response |
| 1338 | 37.975545 | 1.15.0 | host | USBHID | 28 | | SET_REPORT Response |
| 1339 | 37.997791 | 1.15.1 | host | USB | 35 | 0000000000000000 | URB_INTERRUPT in |
| 1342 | 38.399042 | 1.16.1 | host | USB | 27 | | URB_BULK out |
| 1344 | 38.399115 | 1.16.2 | host | USBMS | 40 | | SCSI: Response LUN: 0x00 (Test Unit Ready) (Good) |
| 1345 | 38.645921 | 1.15.1 | host | USB | 35 | 0000190000000000 | URB_INTERRUPT in |
| 1347 | 38.737929 | 1.15.1 | host | USB | 35 | 0000000000000000 | URB_INTERRUPT in |
| 1350 | 39.401615 | 1.16.1 | host | USB | 27 | | URB_BULK out |
| 1352 | 39.401683 | 1.16.2 | host | USBMS | 40 | | SCSI: Response LUN: 0x00 (Test Unit Ready) (Good) |
| 1353 | 39.429649 | 1.15.1 | host | USB | 35 | 00000a0000000000 | URB_INTERRUPT in |
| 1355 | 39.509595 | 1.15.1 | host | USB | 35 | 0000000000000000 | URB_INTERRUPT in |
| 1357 | 40.393950 | 1.15.1 | host | USB | 35 | 00000d0000000000 | URB_INTERRUPT in |
| 1360 | 40.405221 | 1.16.1 | host | USB | 27 | | URB_BULK out |
| 1362 | 40.405290 | 1.16.2 | host | USBMS | 40 | | SCSI: Response LUN: 0x00 (Test Unit Ready) (Good) |
| 1363 | 40.477555 | 1.15.1 | host | USB | 35 | 0000000000000000 | URB_INTERRUPT in |
| 1366 | 41.408476 | 1.16.1 | host | USB | 27 | | URB_BULK out |
| 1368 | 41.408531 | 1.16.2 | host | USBMS | 40 | | SCSI: Response LUN: 0x00 (Test Unit Ready) (Good) |
| 1369 | 42.161608 | 1.15.1 | host | USB | 35 | 0000210000000000 | URB_INTERRUPT in |
| 1371 | 42.241928 | 1.15.1 | host | USB | 35 | 0000000000000000 | URB_INTERRUPT in |
| 1374 | 42.410764 | 1.16.1 | host | USB | 27 | | URB_BULK out |
| 1376 | 42.410838 | 1.16.2 | host | USBMS | 40 | | SCSI: Response LUN: 0x00 (Test Unit Ready) (Good) |
| 1377 | 42.769943 | 1.15.1 | host | USB | 35 | 0200000000000000 | URB_INTERRUPT in |
| 1379 | 43.041885 | 1.15.1 | host | USB | 35 | 0200160000000000 | URB_INTERRUPT in |

Frame 1332: 35 bytes on wire (280 bits), 35 bytes captured (280 bits) on interface 0
 USB URB
 Leftover Capture Data: 0000240000000000

Scrolling down, I saw a lot of URB_INTERRUPT types. We exported these data packets with tshark. For the detailed process, please refer to the second [level](#) of [the 2020 New Year Red Packet Writeup of Milk Ice](#) . I will not repeat it here.

Refer to the [USB Keyboard data packet format](#) , you can know that the first Byte of each packet corresponds to the state of the control key, and the third Byte corresponds to the input key.

Combined with the [USB HID Keyboard scan codes](#), the following script can be constructed to analyze the data packet.

```
usb_codes = {
    0x04: "aA", 0x05: "bB", 0x06: "cC", 0x07: "dD", 0x08: "eE", 0x09: "fF",
    0x0A: "gG", 0x0B: "hH", 0x0C: "iI", 0x0D: "jJ", 0x0E: "kK", 0x0F: "lL",
    0x10: "mM", 0x11: "nN", 0x12: "oO", 0x13: "pP", 0x14: "qQ", 0x15: "rR",
    0x16: "sS", 0x17: "tT", 0x18: "uU", 0x19: "vV", 0x1A: "wW", 0x1B: "xX",
    0x1C: "yY", 0x1D: "zZ", 0x1E: "1!", 0x1F: "2@", 0x20: "3#", 0x21: "4$",
    0x22: "5%", 0x23: "6^", 0x24: "7&", 0x25: "8*", 0x26: "9(", 0x27: "0)",
    0x2C: " ", 0x2D: "-_", 0x2E: "=", 0x2F: "[{", 0x30: "]", 0x32: "#~",
    0x33: ";:", 0x34: "`\"'", 0x36: "<,", 0x37: ">"
}

data = ''
for x in open("xd","r").readlines():
```

```

code = int(x[ 4: 6],16)
print(x[ 4: 6])
if code == 0:
    continue
if code == 0x28:
    print(' ENTER! ')
    print(data)
    data = ''
    continue
upper = 0
if int(x[ 0: 2],16) == 0x02 or int(x[ 0: 2],16) == 0x20:
    upper = 1
data += usb_codes[code][upper]
print(data)

```

After parsing, the compressed package password is obtained and completed.




0x02 RR

RR 298 Points

One of my drives failed and I need help recovering all my files. As far as I know the persons who have set up my PC used something like Reusing All of Internal Disks.

Author: Legacy

https://mega.nz/#!1MEQRCpD!YfSQZQSmKn520Jh8DCBb2Xh0ndqF_kPjgZsQIYtvH8A
<https://drive.google.com/open?id=1hPBksCtm3a4wGs9cd8DYkM4mcV0aNCEJ>

| Name | Type | Compressed size |
|---|-----------------|-----------------|
|  1 | Disc Image File | 150,529 KB |
|  2 | Disc Image File | 0 KB |
|  3 | Disc Image File | 150,489 KB |

I got the title and found that there are three imgs, two of which are the same size, and the other is 0.

According to the meaning of the title "One of my drives failed", it is guessed that the file with a size of 0 is the damaged disk. According to the two disks can "recovering all my files", it may be a

RAID array

```
root@boschko 14:19:44 ~/CTF/blog
# file 1.img 66 file 3.img
1.img: DOS/MBR boot sector MS-MBR Windows 7 english at offset 0x163 "Invalid partition table" at offset 0x17b "Error loading operating system" at offset 0x19a "Missing operating system", disk signature 0xd7f49e5c; p
S (0x0,32,33), end-CHS (0x41,69,4), startsector 2048, 1046528 sectors
3.img: DOS/MBR boot sector MS-MBR Windows 7 english at offset 0x163 "Invalid partition table" at offset 0x17b "Error loading operating system" at offset 0x19a "Missing operating system", disk signature 0xd7f49e5a; p
S (0x0,32,33), end-CHS (0x41,69,4), startsector 2048, 1046528 sectors

root@boschko 14:20:01 ~/CTF/blog
# binwalk 1.img 66 binwalk 3.img

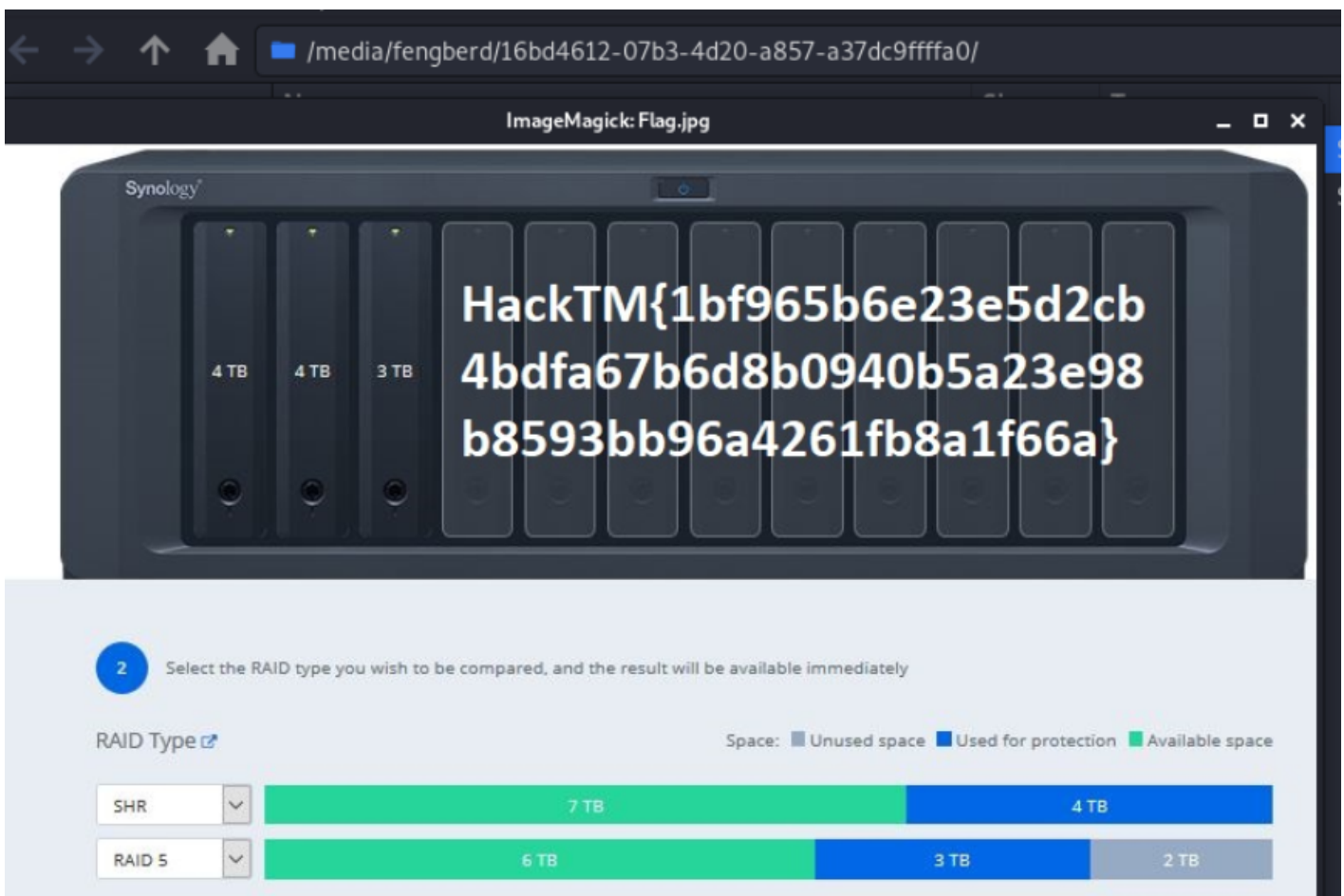
DECIMAL      HEXADECEMAL  DESCRIPTION
1048576      0x100000     Linux EXT filesystem, blocks count: 523264, image size: 535822336, rev 1.0, ext4 filesystem data, UUID=4b485c27-8a32-4116-b5c8-e541cbf3cbf3

DECIMAL      HEXADECEMAL  DESCRIPTION
1048576      0x100000     Linux EXT filesystem, blocks count: 523264, image size: 535822336, rev 1.0, ext4 filesystem data, UUID=7c004893-5625-4a6a-b3f2-562e72b272b2

root@boschko 14:20:18 ~/CTF/blog
# mdadm --misc --examine 1.img
1.img:
  MBR Magic : aa55
  Partition[0] : 1046528 sectors at 2048 (type 83)
```

But there is a hole in this question. The beginning of the img file is filled with an invalid partition table. Using the file command or directly checking with mdadm will not be recognized as a RAID disk. Check with binwalk to find the correct offset.

Here, use dd to simply crop this file, and then use mdadm to view the detailed RAID information, it is indeed a RAID5. Next, create a loop and then do with losetup `mdadm --assemble --run /dev/md0 --readonly /dev/loop0 /dev/loop1` directly mount the hard drive on it. Here you go losetup -o rather use the dd process the file because encountered some mysterious Bug.



The screenshot shows a terminal window with a file browser view of a directory containing a file named 'Flag.jpg'. The file is opened in ImageMagick, displaying a black image with white text. The text is a long alphanumeric string: `HackTM{1bf965b6e23e5d2cb4bdfa67b6d8b0940b5a23e98b8593bb96a4261fb8a1f66a}`.

Below the image, a RAID configuration interface is visible, showing the selection of RAID type and the resulting space usage.

RAID Type

Space: ■ Unused space ■ Used for protection ■ Available space

| RAID Type | Unused space | Used for protection | Available space |
|-----------|--------------|---------------------|-----------------|
| SHR | 7 TB | 4 TB | |
| RAID 5 | 6 TB | 3 TB | 2 TB |

Revision #2

Created 3 October 2020 16:16:10 by Boschko

Updated 3 October 2020 18:26:31 by Boschko