submit date:2022-02-01

DFRWS Challenge 2021

(Team: DFRC)

Korea University, Institute of Cyber Security & Privacy (ICSP), Anam-dong 5-ga Seongbuk-gu, Seoul, Korea.

DFRWS 2021 Challenge Solution

Table of Contents

Table of Contents
🕵️Contact Information
🔧 Tool
📖 Solution
- Part 0. Answer
  - Preconditions
  - [Integrity]
  - [Time Zone]
  - [**The Challenge Answer**]
- Part 1. Skimming Device
  - [Introduction]
  - [Description]
  - [Development]
  - [Additional Analysis]
- Part 2. Raspberry Pi
  - [Introduction]
  - [Description]
  - [Conclusion]
- Part 3. Samsung Smartphone
  - [Introduction]
  - [Description]
  - [Conclusion]
- Part 4. QNAP NAS
  - [Introduction]
  - [Description]
  - [Implementation]
  - [Conclusion]
- Part 5. Multisource Analysis
  - [Introduction]
  - [Approach]
  - [Preparation]
  - [Proposed Process]
  - [Interpretation & Examination of Challenge Data]
  - [Conclusion]

🎓Appendix
🎓References

🕵️Contact Information

Team Name : DFRC

Team Member :

Sang Hyuk An (starson1@korea.ac.kr)

Seung Ah Kang (kyn0503121@korea.ac.kr)

Byeongchan Jeong (naaya@korea.ac.kr)

Sueun Jung (whrgk10@korea.ac.kr)

Jaeseong Kwon (kjs000804@korea.ac.kr)

Jungheum Park (jungheumpark@korea.ac.kr)

Team Nationality : Republic of Korea 🇰🇷

Contact : starson1@korea.ac.kr (+82-10-4170-2391)

🔧 Tool

Tool Table

Name	Type	Part	Version	URL
Autopsy	Freeware	QNAP NAS, Raspberry PI, Samsung Smartphone, Skimming device	4.19.2	https://sleuthkit.org/autopsy/
Sonic Visualizer	Freeware	Skimming device	4.4	https://www.sonicvisualiser.org/download.html
CC Checker	web-open	Skimming device	NaN	https://www.mobilefish.com/services/credit_card_number_checke
Magstripper	Freeware	Skimming device	0.3a	https://sourceforge.net/projects/magstripper/
AccessData FTK Imager	Freeware	QNAP NAS, Raspberry PI, Samsung Smartphone, Skimming device	4.3.0	https://accessdata.com/product-download/ftk-imager-version-4-3
Notepad++	Freeware	Raspberry PI	7	https://notepad-plus-plus.org/downloads/v7.0/
Ultimaker Cura	Freeware	Raspberry PI	4.13	https://ultimaker.com/software/ultimaker-cura
DB browser for SQLite	Freeware	QNAP NAS	3.12.2	https://sqlitebrowser.org/
ChromeCacheView	Freeware	QNAP NAS	2.30	https://www.nirsoft.net/utils/chrome_cache_view.html
jadx	Freeware	Samsung Smartphone	1.3.1	https://varaneckas.com/jad/
CyberChef	web-open	Samsung Smartphone	NaN	https://gchq.github.io/CyberChef
Carpe	Developed, Freeware	Multisource Analysis	20210620	https://github.com/dfrc-korea/carpe

📖 Solution

Part 0. Answer

Preconditions

[Integrity]

Every Image file hash was checked at the introduction of each part.

Exporting files and reading files were done by **AccessData FTK Imager** in read-only mode.

[Time Zone]

Every time given without a specific timezone is considered as **UTC +01:00, UTC +02:00 (DST)**

Since it was broadly found in the given image files.

Daylight Saving Time (DST) period in Europe runs from 01:00 UTC (Coordinated Universal Time) on the last Sunday of March to 01:00 UTC on the last Sunday of October every year.

[The Challenge Answer]

Table captured below out team's answers to the challenge questions.

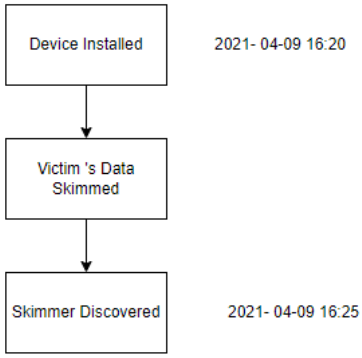
Challenge Questions Response

Aa Question	Answer
1. Skimming Device	- Card number found in recording.mp3. (4334 2250 2436 4939) - Train ticket found from unallocated area (Lausanne → Aosta)
2. Raspberry Pi	- Raspberry Pi has been used to control the 3D printer. - 3D printer printed illicit "Liberator" at "2021-04-13 14:27:57.389504 ~ 2021-04-14 17:29:46.340420".
3. Samsung Smartphone	- Encoded SMS messages was decoded. - Conversations done with E-Mail.
4. QNAP NAS	- Mattermost server hosted for communication system. - traces of Vault was found. file including CC number was found.
5. Multisource analysis and correlation	- Proposed Process includes 6-steps. - Process will cut off time to get forensics by selection and relation analyzing.

Part 1. Skimming Device

[Introduction]

The given device is a physical image dump of a micro-SD card. which was discovered inside a skimmer device which was installed in Lausanne, Switzerland (GMT +02:00)(DST). Sample timeline Figure [1-1] can be constructed according to the provided information. As the skimmer was used, victim's stolen CC number could be found in the files "present" on the memory card.



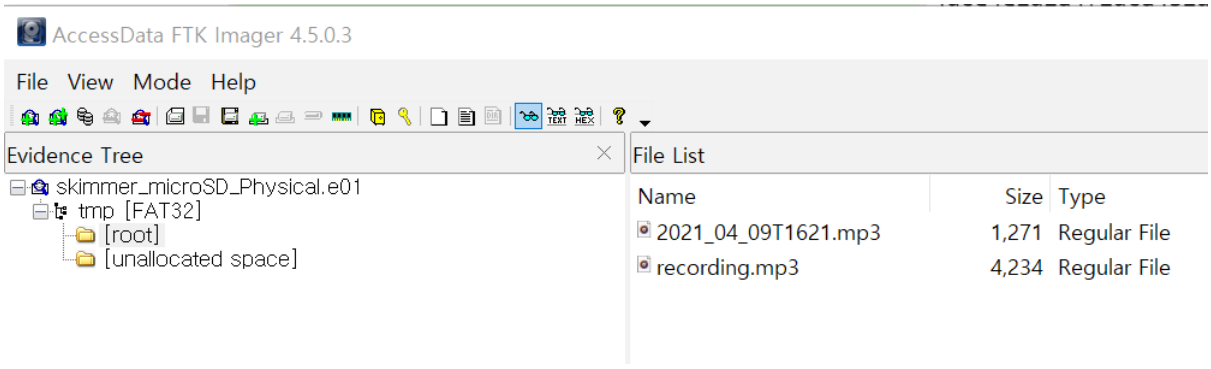
[1-1]. Sample Timeline

[Description]

Image file Description

File Name	skimmer_microSD_Physical.e01
Size (Byte)	36,663,358
SHA256	6CF9C4BC23FE60DDBF79B734C06E85DFFE8007D86339994776F6927E62DC4942
MD5	CF8B1089F4CAFE576B80CAF81160C59C

Using **AccessData FTK Imager**, 2 mp3 files were found in the root directory of the filesystem as Figure [1-2].



[1-2]. File Directory Structure of skimmer_microSD_Physical.e01

Basic information discovered in mp3 files are listed in Table [1-1].

Table [1-1] root directory of skimmer device

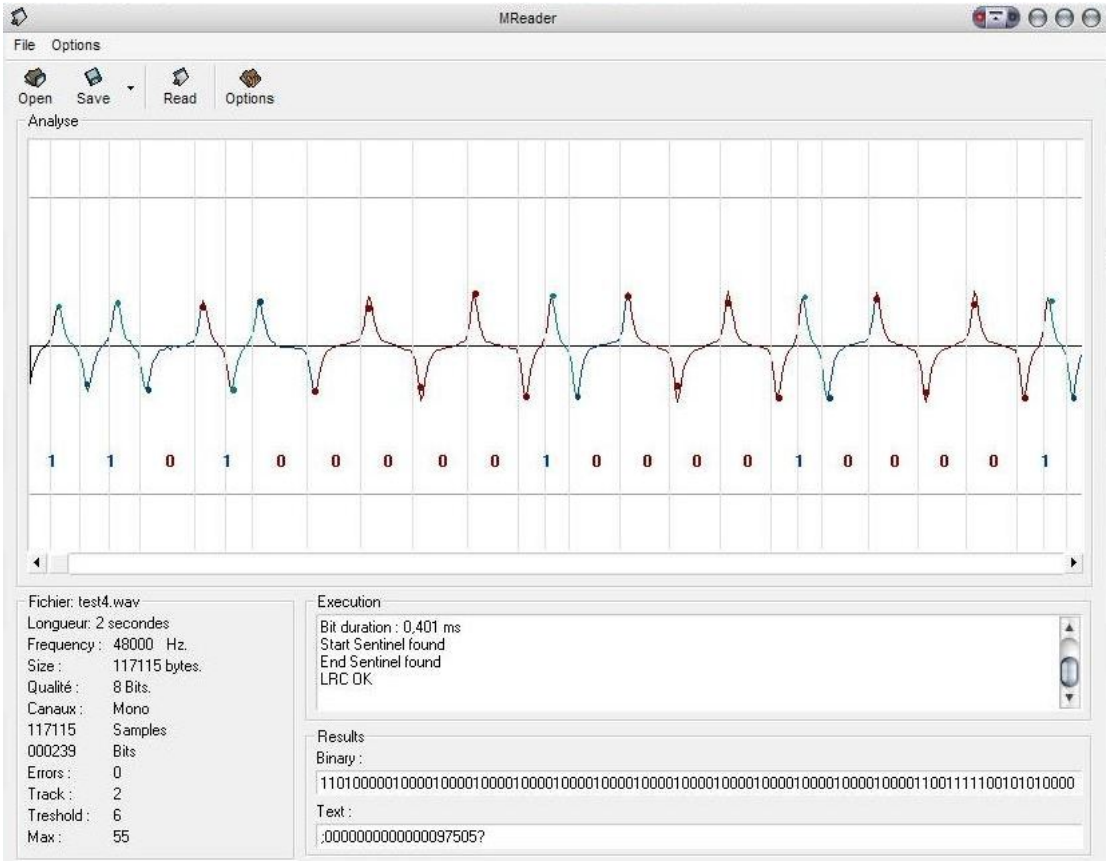
File name	File Type	Created Time	Modified Time	Hash(MD5)
2021_04_09T1621	mp3	@April 9, 2021 4:21 PM (CEST)	@April 9, 2021 4:21 PM (CEST)	066c187f3010f62a56c82298116ec3f8
recording	mp3	@April 9, 2021 4:20 PM (CEST)	@April 9, 2021 4:24 PM (CEST)	b52421a7547369a770b892026d1b25d0

Creation time of recording.mp3 and 2021_04_09T1621.mp3 corresponds with the time mentioned in the introduction, and the length of the mp3 file with modified time. So we could assume that the file was created after the device was installed. The objective was to find the CC number of the victim which was given as “4334 2550 2436 4939”.

Hearing the mp3 files, there was no hearable data. We researched about ATM skimming devices using audio files from the url posted in Crooks Rock Audio-based ATM Skimmers Figure [1-3].

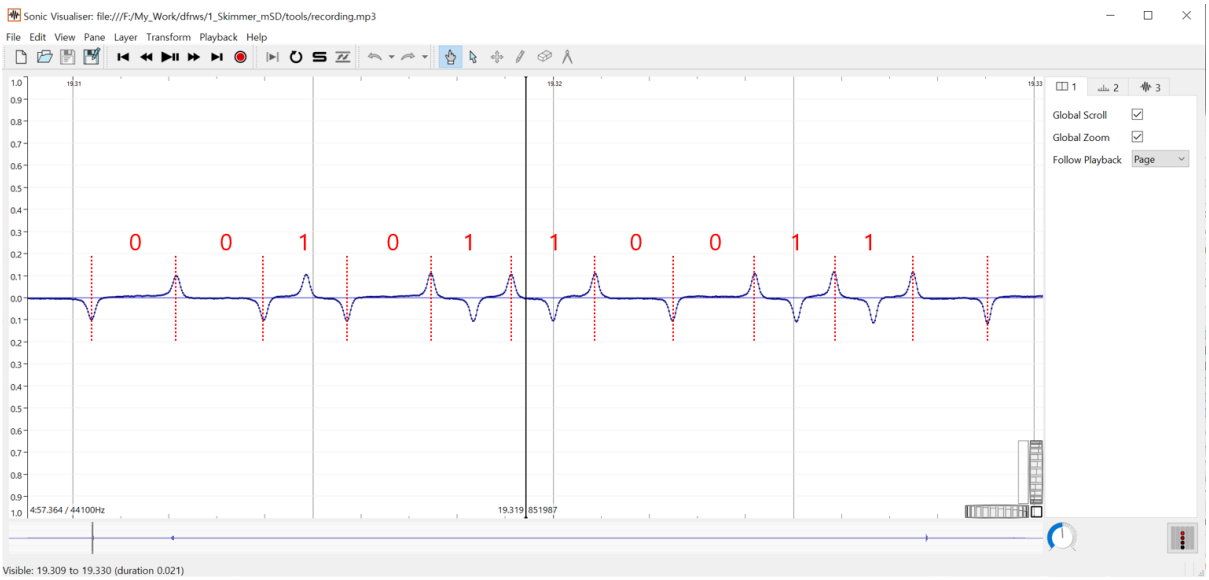
[1-3]. Image example of an ATM skimmer

These skimmers convert magnetic data’s into audio signals that can express 0 & 1. The sample image of encoded signals could be seen as Figure [1-4] the signal 0 could be interpreted as 3 peaks in one interval and the signal is formed like axisymmetry. The signal 1 could be interpreted as 2 peaks in one interval formed like point symmetry.



[1-4] sample of signals in mp3 file

The principle of the skimming device which is legal or illegal. It should have the same logic reading credit cards. The logic reading magnetic credit card is written as an ISO standard. **Information technology - Identification cards - Financial transaction cards** which is very similar to the sonic waves(Figure [1-5]) that could be found on the files given in our case. Visualization of the audio data is done by **Sonic Visualize.**



[1-5] visualized audio of recording.mp3

Standard data format of the audio file is descripted at Table [1-2] about Track1 data, Table [1-3] which explains Track2 data. There is maximum number of 3 tracks on magnetic cards known as Track 1,2,3. Most of cards used on the market reads track 1, or track 2.

Track 1 data is composed of 7bits per character (6 data bits + 1 parity bit) with 6 bits, it can express 64 characters in ASCII Table. from 0x20 to 0x60. With this reason Track 1 data can contain alphabetical data(Upper case only). such as name of the card owner.

Track 2 data is composed of 5 bits per character (4 data bits = 1 parity bit) with 4 bits it can express 16 character ins ASCII Table which is number 0 to 9 and : ; > < = ?.

[Table 1-2] Magnetic Stripe Data Format (Track 1)

Aa Name	# Size(Max)	Description
Start Sentinel	1	‘%’
Format Code	1	‘B’
PAN	19	up to 19 numbers, Creditcard numbers written on the front
Field Separator	1	‘^’
Name	26	up to 26 characters
Field Separator	1	'^'
Expiration date	4	YYMM
Service code	3	
Discretionary data	4	Card Verification Code(CVC)
End Sentinel	1	‘?’
LRC	1	validation character calculated from other data(Longitudinal Redundancy Check)

[Table 1-3] Magnetic Stripe Data Format (Track 2)

Aa Name	# Size(Max)	Description
Start sentinel	1	‘;’
PAN	19	up to 19 numbers, Creditcard numbers written on the front
Separator	1	‘=’
Expiration Date	4	MMYY
Service Code	3	interchange rules / authorization / Service range
Discretionary Data	4	Card Verification Code(CVC)
End Sentinel	1	‘?’
LRC	1	validation character calculated from other data(Longitudinal Redundancy Check)

Considering the data format, we made a binary decoder for the magnetic data reader. In the perspective of forensic science, we considered interpreting sonic data into binary data relatively unnecessary compared to decoding the data. So our first work was to read binary data directly with inspectors judgement. As a result we were able to get binary data in Figure [1-6] and decoded results with our implemented code.

```
00000000000000000000000000000000000000001111110000100000000110000100111100100001100001000010000100001000000001100000001000001100001001100111001110011100100101100000000000000000
```

Implemented code is explained in the **Development** section. Using the developed tool, there was 2 CC data. former PAN number is given in the introduction. and the other PAN number is found. Found CC data is analyzed by **CC Checker**.

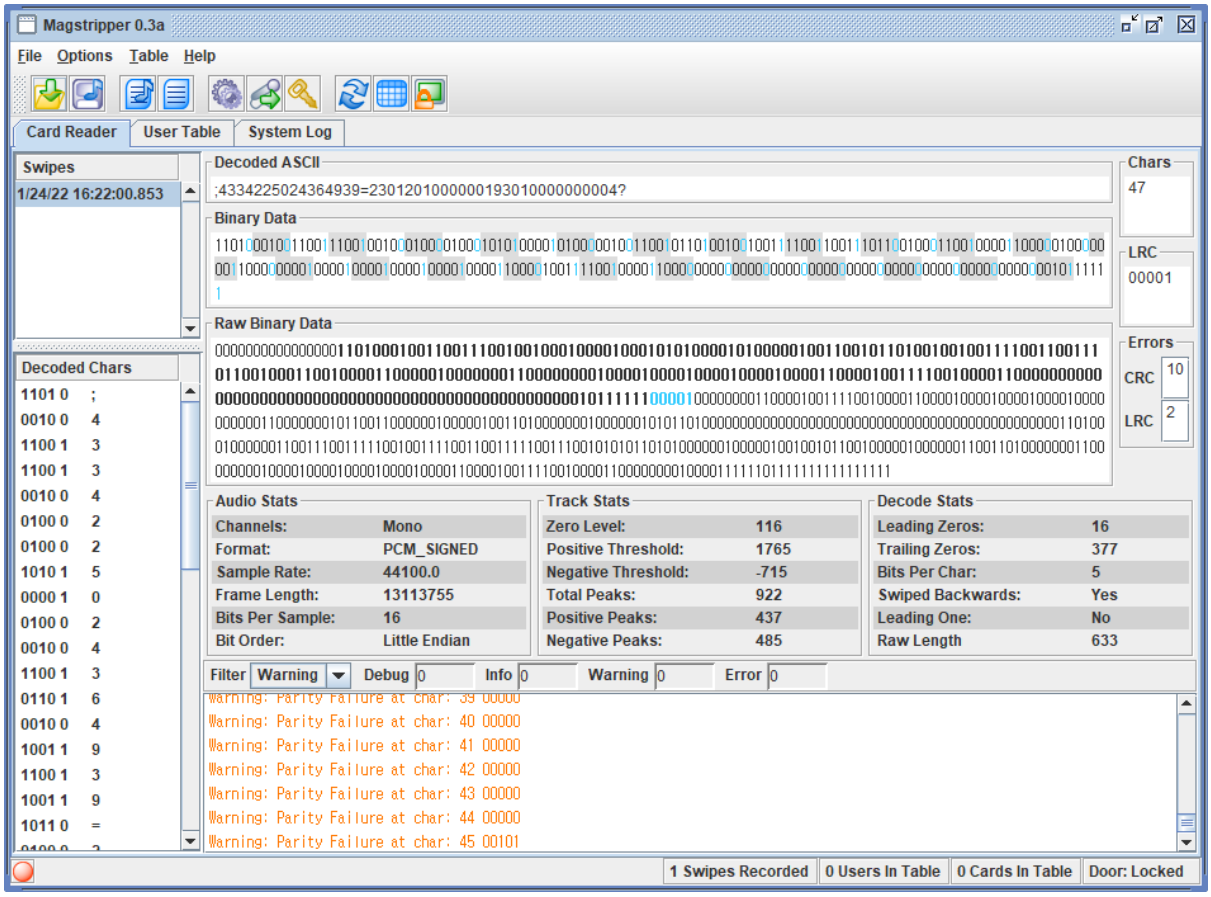
- 4334 2250 2436 4939

```
Credit card number: 43342250 2436 4939
Personal account number: 502436493
Major Industry Identifier: (4) Travel and entertainment and banking/financial
Bank identification number (BIN): 433422
Issuing bank: Aval Card, S.a. De C.v.
Card type: Visa Credit
Country: Honduras
Phone bank: 506-2090800
```

- 4099 7939 7756 2804

```
Credit card number: 4099 7939 7756 2804
Personal account number: 397756280
Major Industry Identifier: (4) Travel and entertainment and banking/financial
Bank identification number (BIN): 409979
Issuing bank: Star Networks, Inc
Card type: Visa
Country: United States Of America
Phone bank:
```

For accuracy, we found a tool named **Magstripper** that does similar action decoding wav files. Figure [1-7] shows the result, we can get exactly the same value done with implemented code above. By double checking with implemented code and freeware developed previously, it is safe to say that accuracy is guaranteed.



[1-7] Result with Magstripper

In National Conference of State Legislatures, almost every state executes sentence for the purpose of defraud. For some states, possession of a credit card skimming device is illegal. Which makes the user of the skimming device, and the person who possess data of skimmed credit card.

[Development]

The following code converts wav file into binary file, then decodes binary data into readable ASCII values. A simple explanation of methodology is included. implemented code is uploaded in **appendix [1-1]**

[bin2dat]

Under various conditions different data can be derived. This is because of the minimum size of peak that will be considered as a bit signal. For this reason this function controls the minimum size of the peak and prints the data, decoded data using different functions for each data format “track1”,”track2”. Also, the direction of drawing the card can make data that is in opposite order. Considering these cases, this code also decode data when it is inversed.

[track1]

This function decodes data consisted of “Track1” data.
checks parity bit for error detection.

[track2]

This function decodes data consisted of “Track2” data.
checks parity bit for error detection.

[find_sentinel]

Using string search algorithm included in python, finds every sentinel signatures for each data format “Track1”, “Track2” and returns found indexes.

[parity_chk]

Simple parity checker, makes sure there is odd numbers of ‘1’ in group set of binary.

[dab]

code referenced by **Magstripe**’s **dab.py**

decodes wav file into binary files in condition of silence level (’sil’).

Appendix) The path of the file is `DFRC_Appendix.zip/Part 1 Skimming Device/[1-1]main.py`

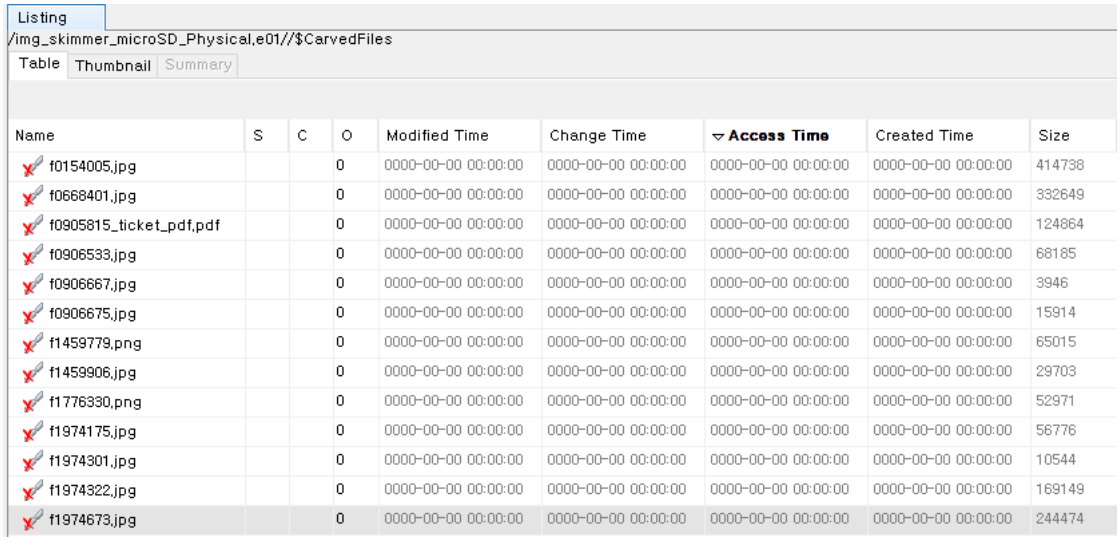
```
class MagDataDecoder:
    def __init__(self,filename):
        self.filename = filename
    def bin2dat(self,mode):
        if mode ==1:
            sil = 0
            while(1):
                if sil == 60: break
                print("sil : "+str(sil))
                data = self.dab(self.filename,sil)
                res = self.track1(data)
                res_inv = self.track1(data[::-1])
                print(res)
                print("")
                print(res_inv)
                print("")
                sil += 0.01
        elif mode ==2:
            sil = 0
            while(1):
                if sil == 60: break
                print("sil : " + str(sil))
                data = self.dab(self.filename,sil)
                self.track2(data)
                self.track2(data[::-1])

                sil += 0.01
  def track1(self,data):
        start, end = self.find_sentinel(1, data)
```

```
        for i in start:
            for j in end:
                if (i < j) and ((j - i) % 7 == 0):
                    result = ""
                    # print("sentienel index -> "+str(i)+", "+str(j))
                    for idx in range(i, j, 7):
                        if self.parity_chk(data[idx:idx + 7]) == -1:
                            break
                        else:
                            char = chr(int(data[idx:idx + 6][::-1], 2) + 32)
                            result += char
                    print(result)
    def track2(self,data):
        start,end = self.find_sentinel(2,data)
        for i in start:
            for j in end:
                if (i<j) and ((j - i)%5 == 0):
                    result = ""
                    #print("sentienel index -> "+str(i)+", "+str(j))
                    for idx in range(i,j,5):
                        if self.parity_chk(data[idx:idx+5]) == -1:
                            break
                        else:
                            char = chr(int(data[idx:idx + 4][::-1], 2) + 48)
                            result += char
                    print(result)
    def find_sentinel(self,trk,data):
        if trk == 1:
            start = [i for i in range(len(data)) if data.startswith('101000' + '1', i)]
            end = [i for i in range(len(data)) if data.startswith('111110' + '0', i)]
            return start,end
        if trk == 2:
            start = [i for i in range(len(data)) if data.startswith('1101'+'0',i)]
            end = [i for i in range(len(data)) if data.startswith('1111' + '1', i)]
            return start,end
        else:
            print("trk input error....")
            return -99,-99
    def parity_chk(self,dat):
        flag = dat.count('1')
        if flag % 2 == 0:
            return -1
        else:
            return 1
    def wav2bin(self,filename):
        samplerate, data = wavfile.read(filename)
        print(len(data))
        print(type(data))

        epoch = 0
        for i in range(1,len(data)-1):
            if (data[i] > data[i-1] and data[i] > data[i+1])\
                    and data[i] > 20:
                print("point : "+str(i)+" DB : "+str(data[i]))
    def dab(self,filename,sil):
        if sil != 0:
           thresh= 100 / float(sil)
        else:
           thresh= 100 / 33
        track=wave.open(filename)
        params=track.getparams()
        frames=track.getnframes()
        channels=track.getnchannels()
        if not channels == 1:
           sys.stderr.write("track must be mono!")
           return -1
        n= 0
        max= 0
        samples= []
        # determine max sample and build sample list
        while n < frames:
           n += 1
           # make sample an absolute value to simplify things later on
           current= abs(unpack("h",track.readframes(1))[0])
           if current > max:
              max= current
           samples.append(current)
        # set silence threshold
        silence= max / thresh
        # create a list of distances between peak values in numbers of samples
        # this gives you the flux transition frequency
        peak= 0
        ppeak= 0
        peaks= []
        n= 0
        while n < frames:
           ppeak= peak
```

```
            # skip to next data
            while n < frames and samples[n] <= silence:
               n= n+1
            peak= 0
            # keep going until we drop back down to silence
            while n < frames and samples[n] > silence:
               if samples[n] > samples[peak]:
                  peak= n
               n= n+1
            # if we've found a peak, store distance
            if peak - ppeak > 0:
               peaks.append(peak - ppeak)
         zerobl = peaks[2]
         n= 2
         # allow some percentage deviation
         freq_thres= 60
         output= ''
         while n < len(peaks) - 1:
            if peaks[n] < ((zerobl / 2) + (freq_thres * (zerobl / 2) / 100)) and peaks[n] > ((zerobl / 2) - (freq_thres * (zerobl / 2)
               if peaks[n + 1] < ((zerobl / 2) + (freq_thres * (zerobl / 2) / 100)) and peaks[n + 1] > ((zerobl / 2) - (freq_thres * (z
                  output += '1'
                  zerobl= peaks[n] * 2
                  n= n + 1
            else:
                if peaks[n] < (zerobl + (freq_thres * zerobl / 100)) and peaks[n] > (zerobl - (freq_thres * zerobl / 100)):
                  output += '0'
                  zerobl= peaks[n]
            n= n + 1
         sys.stderr.write("number of bits: " + str(len(output))+"\n")
         #sys.stderr.write("\n")
         print(output)
         return output
```

[Additional Analysis]

Using **Autopsy** Figure [1-9] we recovered data in the unallocated area, extracting 12 image files, and single PDF file.



Name	S	C	O	Modified Time	Change Time	Access Time	Created Time	Size
f0154005.jpg			0	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	414738
f0668401.jpg			0	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	332649
f0905815_ticket_pdf.pdf			0	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	124864
f0906533.jpg			0	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	68185
f0906667.jpg			0	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	3946
f0906675.jpg			0	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	15914
f1459779.png			0	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	65015
f1459906.jpg			0	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	29703
f1776330.png			0	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	52971
f1974175.jpg			0	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	56776
f1974301.jpg			0	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	10544
f1974322.jpg			0	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	169149
f1974673.jpg			0	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00	244474

[1-9] Carved Files in skimmer_microSD_Physical.e01

From the pdf file Figure [1-10] name of a suspect could be found “Arnim Zola”. The reason why Arnim Zola is the suspect is mentioned in Part 2, “it was possible to identify a location of interest in Aosta.”

SBB CFF FFS

Ticket-ID 390708058822

Economy Ticket				Arnim Zola		
Valid: 2021				20.05.1941		
30		von/de/da/from	➔ nach/a/a/to	30		KL. CL.
28.03 *	09:21 *	Lausanne *	➔Aosta ➔ *	28.03 *	18:23 *	2 *

Full Price	CHF 26.40 PCD

(L)(SPEZ)(SPEZ)(70)
No ordine: 438781486

No item: 4004
incl. 7.70% IVA/FFS

[1-10] Train Ticket carved from skimmer_microSD_Physical.E01

5 of 15 images were considered associated with the given information. from Figure [1-11] **Arnim Zola** was found as the name of a character from the marvel universe. Also in the unallocated area, 4 picture of a single man was discovered as Arnim Zola via google image analysis.

[1-11] Arnim Zola

And there is a afffiliation named **Hydra** which is a terrorist organization. This organization uses an Emblem Figure [1-12] same as the carved image found in the unallocated area.

[1-12] Hydra Emblem

Part 2. Raspberry Pi

[Introduction]

Evidence Item 2-2_Raspberry_Pi_mSD

Based on forensic analysis of the SDCard from the seized skimming device, it was possible to identify a location of interest in Aosta.

Italian authorities searched the residence on April 18th, 2021. The investigators discovered a laboratory, but most of the devices and manufacturing equipment were destroyed. However, a specialist from the Reparto Investigazioni Scientifiche (RIS) of the Carabinieri identified a Raspberry Pi, connected to a 3D printer, seemingly forgotten in the destruction process. A forensic copy of the microSD card of the Raspberry Pi was acquired and is available for download here.

- Filename : 2_Raspberry_Pi_mSD.zip
- SHA2-256 : aabec0c1305e785d1ba5b4ba01c5dacd27cc128fdd32078758be826e75449953

No traces of 3D printed objects were found on site. Given the presence of the 3D printer connected to the Raspberry Pi, particular attention should be given to:

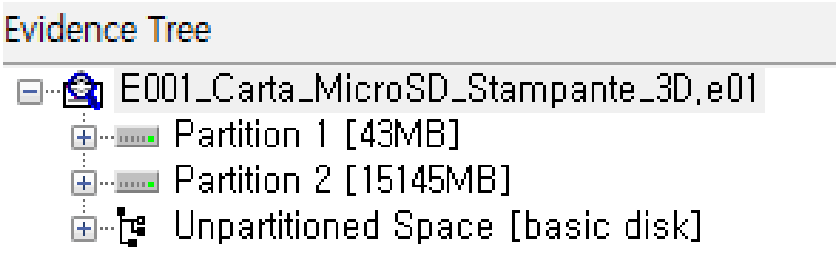
- Establishing whether the Raspberry Pi has been used to control the 3D printer.
- Establishing whether objects of possible illicit use have been printed, when and which ones.

[Description]

Image file Description

File Name	E001_Carta_MicroSD_Stampante_3D.e01
Size (Byte)	2,161,644,391
SHA256	1f6420581b901d647d98cb6fd1a4c6c8802f6a83a88ea4003405fa46b30e72a9
MD5	19C862E8ACAD9F9349CC034F7C0E3BF4

The given device is a forensic copy of the microSD card of the Raspberry Pi. It discovered in Aosta laboratory on April 18th, 2021 (UTC +01:00, **UTC+02:00 at DST**). Our analysis is mainly about 3D printer connected to the host device(Raspberry PI) and which data was printed.



[2-1] AccessData FTK Imager : Image Partition Structure

Using **AccessData FTK Imager** two number of partition is identified as Figure [2-1].

Partition 1 [43MB] is FAT32 filesystem, Partition 2 [15145MB] is EXT4. both of filesystem is analyzed and descripted in this solution.

Booting Partition

#1. **octopi-wpa-supplicant.txt**

`boot/octopi-wpa-supplicant.txt` is configuring WiFi file to boot Raspberry Pi from OctoPi Image. OctoPi is A Raspberry Pi distribution for 3d printers. It includes the OctoPrint host software for 3d printers out of the box and mjpg-streamer with RaspiCam support for live viewing of prints and timelapse video creation.

Details of WiFi configuration section is attached in Firgure [2-2]. The SSID of WiFi network is "**Ctulhuuuu - Media**", and the password for this WiFi is **"r9yqU-WpoX-lCDL-ZAzy-QRAYl".**

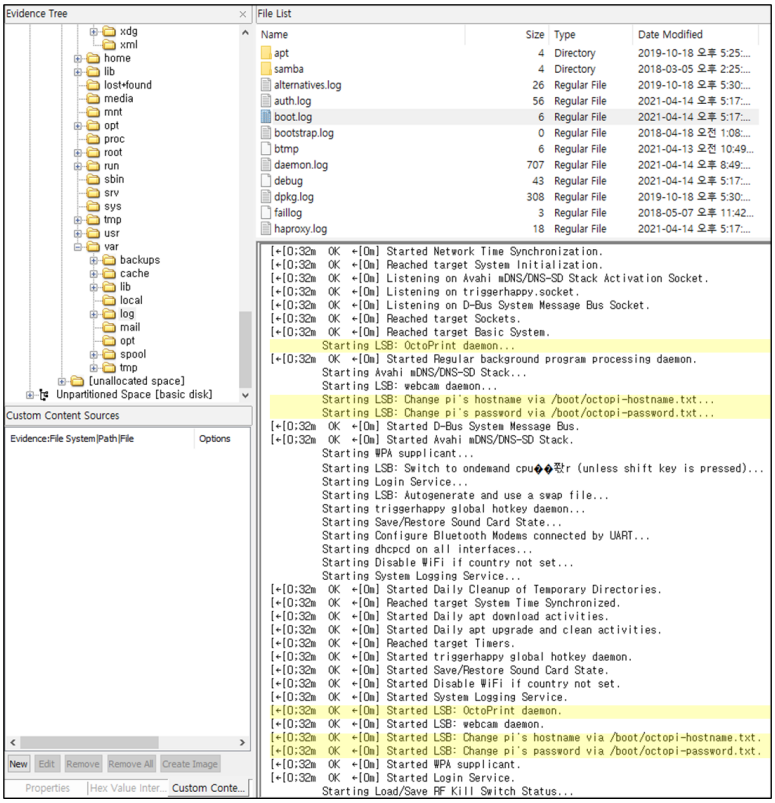
```
# Use this file to configure your wifi connection(s).
#
# Just uncomment the lines prefixed with a single # of the configuration
# that matches your wifi setup and fill in SSID and passphrase.
#
# You can configure multiple wifi connections by adding more 'network'
# blocks.
#
# See https://linux.die.net/man/5/wpa_supplicant.conf
# (or 'man -s 5 wpa_supplicant.conf') for advanced options going beyond
# the examples provided below (e.g. various WPA Enterprise setups).
#
# !!!!! HEADS-UP WINDOWS USERS !!!!!
#
# Do not use Wordpad for editing this file, it will mangle it and your
# configuration won't work. Use a proper text editor instead.
# Recommended: Notepad++, VSCode, Atom, SublimeText.
#
# !!!!! HEADS-UP MACOSX USERS !!!!!
#
# If you use Textedit to edit this file make sure to use "plain text format"
# and "disable smart quotes" in "Textedit > Preferences", otherwise Textedit
# will use none-compatible characters and your network configuration won't
# work!

## WPA/WPA2 secured
## WPA/WPA2 secured
network={
  ssid="Cthulhuuuu - Media"
  psk="r9yqU-WpoX-lCDL-ZAzy-QRAYl"
}

## Open/unsecured
#network={
#  ssid="put SSID here"
#  key_mgmt=NONE
#}

## WEP "secured"
##
## WEP can be cracked within minutes. If your network is still relying on this
## encryption scheme you should seriously consider to update your network ASAP.
#network={
#  ssid="put SSID here"
#  key_mgmt=NONE
#  wep_key0="put password here"
#  wep_tx_keyidx=0
#}
```

[2-2] boot/octopi-wpa-supplicant.txt

```
network={
  ssid="Cthulhuuuu - Media"
  psk="r9yqU-WpoX-lCDL-ZAzy-QRAYl"
}
```

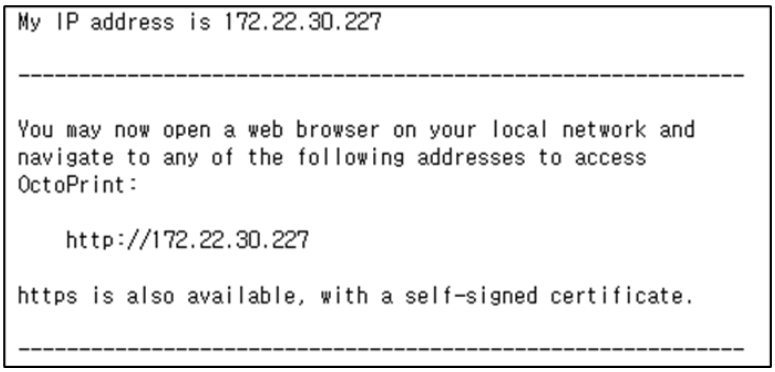
Data Partition

#1. boot.log



[2-3] /var/log/boot.log

Figure [2-3] is the content of `/var/log/boot.log`. This logfile records message that is printed while OS boots. The highlighted part is the evidence that OctoPrint daemon was executed.



[2-4] /var/log/boot.log

Also by another section in `/var/log/boot.log` Figure [2-4], address “http://172.23.0.227” was used to connect OctoPrint. Which makes remote connection available. Through the evidence listed in Figure [2-3], Figure [2-4], it is assured that the user of Raspberry Pi booted the host device in order to connect with the 3D printer. So we analyzed files that is related with **Octoprint**.

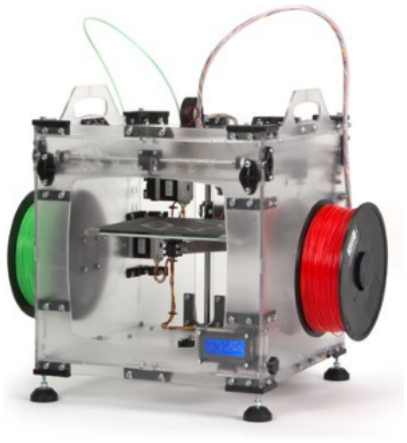
Raspberry Pi IP : 172.23.0.227

#2. printer profiles

In `/home/pi/.octoprint/printerProfiles/_default.profile`, which is Figure [2-5], the model of the 3D printer was identified as "**Velleman Vertex K8400**".

```
axes:
  e:
    inverted: false
    speed: 300
  x:
    inverted: false
    speed: 6000
  y:
    inverted: false
    speed: 6000
  z:
    inverted: false
    speed: 200
color: default
extruder:
  count: 1
  nozzleDiameter: 0.4
  offsets:
  - - 0.0
    - 0.0
  sharedNozzle: false
heatedBed: false
heatedChamber: false
id: _default
model: Velleman Vertex K8400
name: K8400
volume:
  custom_box: false
  depth: 180.0
  formFactor: rectangular
  height: 190.0
  origin: lowerleft
  width: 190.0
```

[2-5] /home/pi/.octoprint/printerProfiles/_default.profile : 3D Printer Model Name



[2-6] Velleman Vertex K8400 Printer

Figure [2-6] 3D Printer (**Velleman Vertex K8400**), https://www.velleman.eu/products/view/?id=417866.

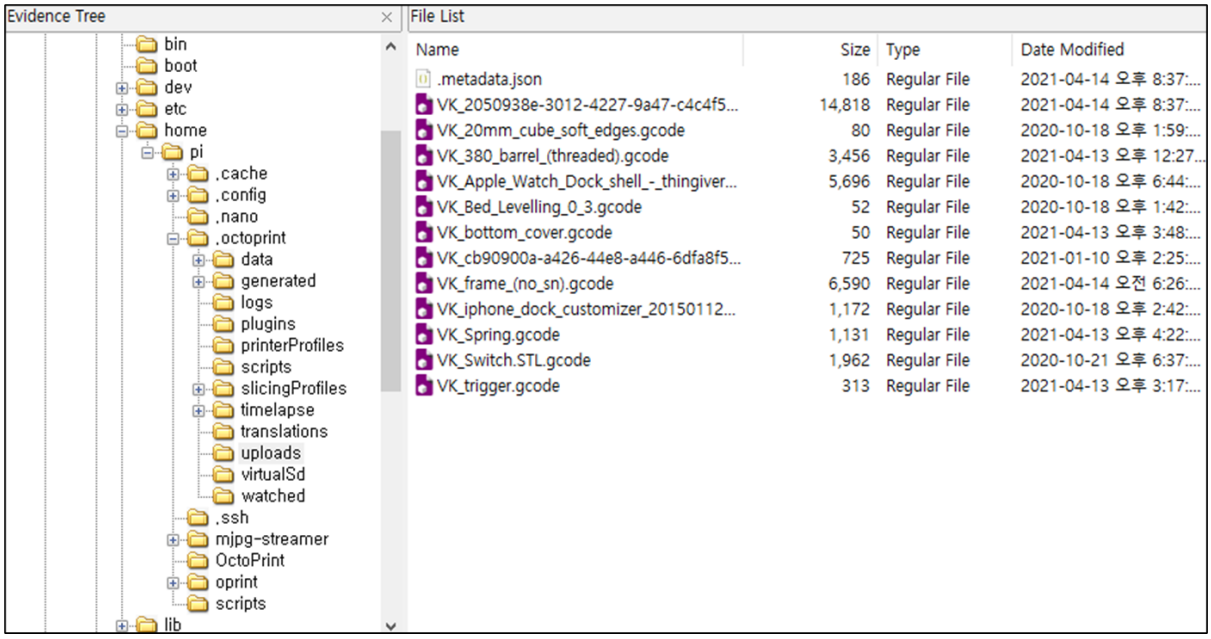
#3. config.yaml

`/home/pi/.octoprint/config.yaml file` indicates configuration information about Octoprint. **Notepad++** was used to open and view this file.

```
accessControl:
    salt: lo3NcKz2baszfxUQj8XjtS5Ex4pGD57g
api:
    key: 2C859052F51649AC8D82FAFCF6305459
appearance:
    name: Cthulhuuuu's 3D Printer
feature:
    sdSupport: false
plugins:
    PrintTimeGenius:
        print_history:
        -   analysisFirstFilamentPrintTime: 14.066955238828996
            analysisLastFilamentPrintTime: 439.07652791635525
            analysisPrintTime: 459.158660518098
            compensatedPrintTime: 459.158660518098
            firstFilamentPrintTime: 7.7476996589998635
            lastFilamentPrintTime: 435.7885079699997
            payload:
                file: /home/pi/.octoprint/uploads/VK_20mm_cube_soft_edges.gcode
                filename: VK_20mm_cube_soft_edges.gcode
                name: VK_20mm_cube_soft_edges.gcode
                origin: local
                owner: arnimzola
                path: VK_20mm_cube_soft_edges.gcode
                size: 89675
                time: 464.4652355179999
            timestamp: 1571479906.162659
        printer_config:
        - M92 X134.74 Y134.74 Z4266.66 E148.70
        - M203 X160.00 Y160.00 Z10.00 E10000.00
        - M201 X9000 Y9000 Z100 E10000
        - M204 S6000.00 T6000.00
        - M205 B20000 E20.00 S0.00 T0.00 X10.00 Z0.50
```

[2-7] /home/pi/.octoprint/config.yaml

In the column named “accessControl” in `config.yaml`, the name of the printer is “Cthulhuuuu’s 3D Printer” that was using OctoPrint. Also in the “Plugins” column, the username “arnimzola”, and the history information for the gcode(command file for 3D printing process) file. “http://127.0.0.1:8080/?action=snapshot” was the address that snapshots were saved from the webcam setting.



[2-8] /home/pi/.octoprint/uploads

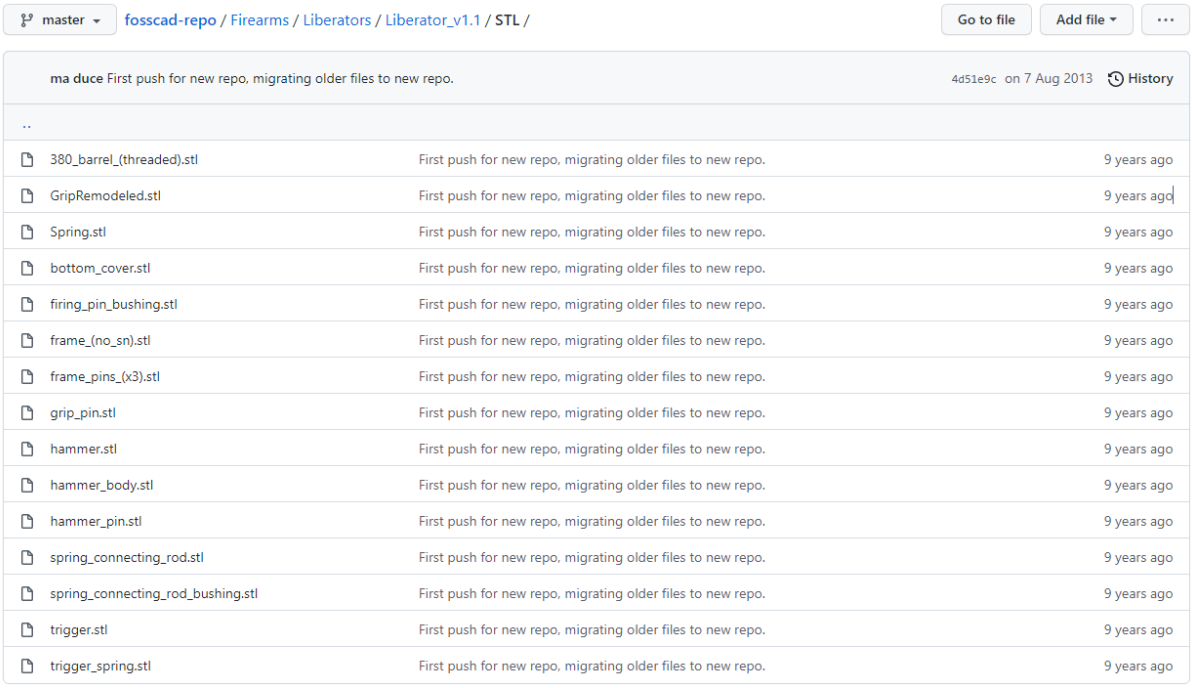
Figure [2-8] is file list of `/home/pi/.octoprint/uploads`. The Upload folder was to upload gcode files using OctoPrint. When the files are uploaded, analysis sequence is carried out. Gcode file contains commands in the G-Code, which is a language used to describe how a 3D printer should print a job. It stores instructions in plain text with each line representing a different command, such as how fast the printer should print, the temperature it should be set at, and where the printing parts should move.

[Table 2-1] File distribution of upload file

File name	Created Time	Accessed Time	Modified Time
VK_Bed_Levelling_0_3.gcode	2020-10-18 PM 3:42:33	2020-10-18 PM 3:42:33	2020-10-18 PM 3:42:33
VK_20mm_cube_soft_edges.gcode	2020-10-18 PM 3:59:13	2020-10-18 PM 3:59:13	2020-10-18 PM 3:59:13
VK_iphone_dock_customizer_20150112-3763-zv7a1d-0.gcode	2020-10-18 PM 4:42:53	2020-10-18 PM 4:42:53	2020-10-18 PM 4:42:53
VK_Apple_Watch_Dock_shell_-_thingiverse.gcode	2020-10-18 PM 8:44:29	2020-10-18 PM 8:44:30	2020-10-18 PM 8:44:30
VK_Switch.STL.gcode	2020-10-21 PM 8:37:42	2020-10-21 PM 8:37:43	2020-10-21 PM 8:37:43
VK_cb90900a-a426-44e8-a446-6dfa8f5beb7d.gcode	2021-01-10 PM 3:25:32	2021-01-10 PM 3:25:32	2021-01-10 PM 3:25:32
VK_380_barrel_(threaded).gcode	2021-04-13 PM 2:27:56	2021-04-13 PM 2:27:57	2021-04-13 PM 2:27:57
VK_trigger.gcode	2021-04-13 PM 5:17:29	2021-04-13 PM 5:17:29	2021-04-13 PM 5:17:29
VK_bottom_cover.gcode	2021-04-13 PM 5:48:25	2021-04-13 PM 5:48:25	2021-04-13 PM 5:48:25
VK_Spring.gcode	2021-04-13 PM 6:22:57	2021-04-13 PM 6:22:57	2021-04-13 PM 6:22:57
VK_frame_(no_sn).gcode	2021-04-14 AM 8:26:15	2021-04-14 AM 8:26:16	2021-04-14 AM 8:26:16
VK_2050938e-3012-4227-9a47-c4c4f5ab1e18.gcode	2021-04-14 PM 10:36:58	2021-04-14 PM 10:37:01	2021-04-14 PM 10:37:01

It was possible to assume that user of the 3D printer was going to print gun parts from the files named “trigger”, “barrel” and “frame”. While on research about 3D printing gun parts, we found the github link that prints **Liberator** (physible (3D-printable) single shot handgun, the first such printable firearm design made widely available online).

https://github.com/maduce/fosscad-repo/tree/master/Firearms/Liberators/Liberator_v1.1/STL



master fosscad-repo / Firearms / Liberators / Liberator_v1.1 / STL / Go to file Add file ···

ma duce First push for new repo, migrating older files to new repo. 4d51e9c on 7 Aug 2013 History

..		
380_barrel_(threaded).stl	First push for new repo, migrating older files to new repo.	9 years ago
GripRemodeled.stl	First push for new repo, migrating older files to new repo.	9 years ago
Spring.stl	First push for new repo, migrating older files to new repo.	9 years ago
bottom_cover.stl	First push for new repo, migrating older files to new repo.	9 years ago
firing_pin_bushing.stl	First push for new repo, migrating older files to new repo.	9 years ago
frame_(no_sn).stl	First push for new repo, migrating older files to new repo.	9 years ago
frame_pins_(x3).stl	First push for new repo, migrating older files to new repo.	9 years ago
grip_pin.stl	First push for new repo, migrating older files to new repo.	9 years ago
hammer.stl	First push for new repo, migrating older files to new repo.	9 years ago
hammer_body.stl	First push for new repo, migrating older files to new repo.	9 years ago
hammer_pin.stl	First push for new repo, migrating older files to new repo.	9 years ago
spring_connecting_rod.stl	First push for new repo, migrating older files to new repo.	9 years ago
spring_connecting_rod_bushing.stl	First push for new repo, migrating older files to new repo.	9 years ago
trigger.stl	First push for new repo, migrating older files to new repo.	9 years ago
trigger_spring.stl	First push for new repo, migrating older files to new repo.	9 years ago

[2-9] Github : Liberator stl file

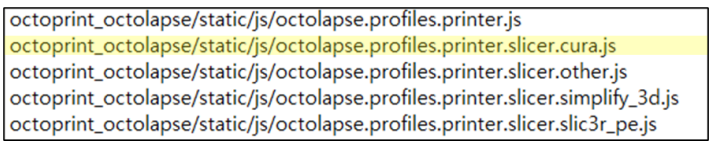
STL is a file format native to the stereolithography CAD software created by 3D Systems. In Figure [2-9], under the path `/home/pi/.octoprint/uploads` a specific file name was identical to the stl files that was composing Liberator. stl file uses program such as **Ultimaker Cura Program** in order to convert stl files into gcode files and print it with the 3D printer. Finally, it is clear that "**arnimzola**" who is the user of the 3D printer uploaded gcode files that is presumed to be Liberator.

The Liberator is a physible (3D-printable) single shot handgun, the first such printable firearm design made widely available online. Aosta, Italy needs approval in order to purchase firearms. License is issued after approval, without these approval, it is illicit for a individual to have a Liberator.

> **Ultimaker Cura Program**
> Cura is the world's most popular 3D printing software. Cura can prepare prints with a few clicks, integrate with CAD software for an easier workflow, or dive into custom settings for in-depth control. Cura suppots slicing stl files and converting the results into a gcode file.

With `/home/pi/slicingProfiles/cura existing`, it was possible to think that Cura was installed. Also `/home/pi/oprint/lib/python2.7/site/packages/Octolapse-0.3.4-py2.7egg-info/SOURCES.txt` shows "cura.js " that is highlighted in Figure [2-10]. Which shows that cura was used as a stl slicer.

by using **Autopsy** in Figure [2-11], deleted files were found that was related with cura, under `/home/pi/.cache/pip/http/c/f`

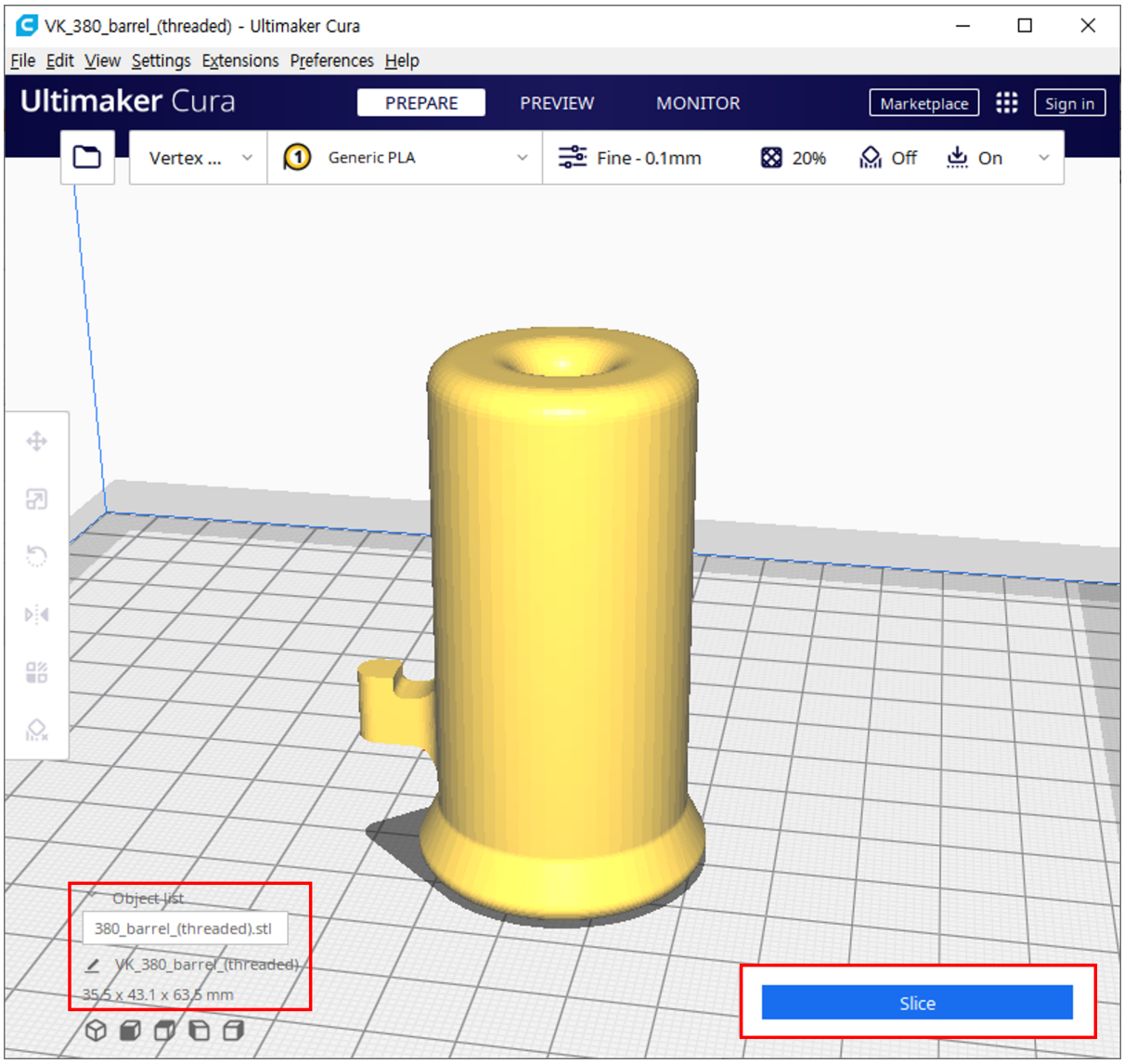


[2-10] /home/pi/oprint/lib/python2.7/site/packages/Octolapse-0.3.4-py2.7.egg-info/SOURCES.txt : cura artifact

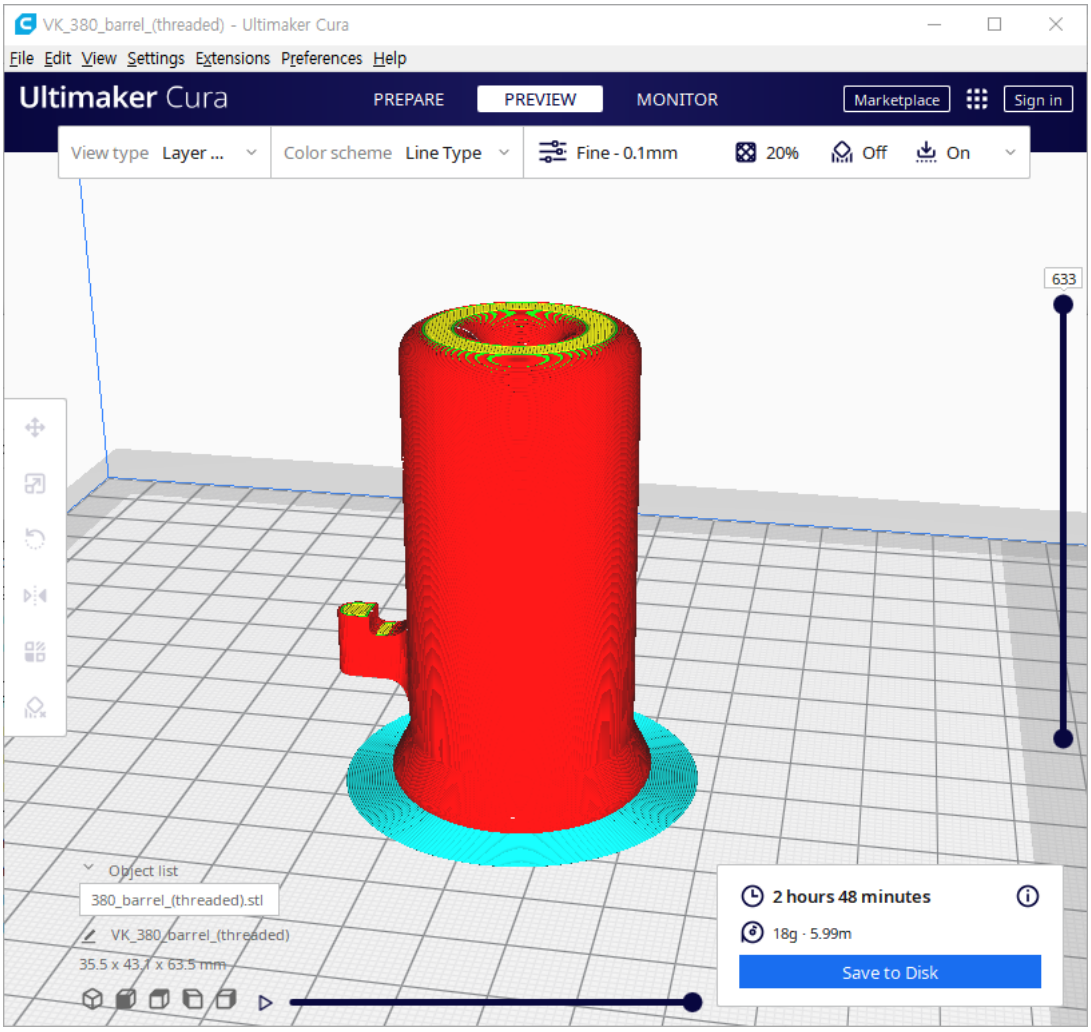
Name	S	C	O	Modified Time	Change Time	Access Time	Created Time
printer_profile_cura.png				2021-04-15 00:21:33 KST	2021-04-15 00:21:33 KST	2021-04-14 16:21:03 KST	2021-04-14 16:21:03 KST
printer_profile_cura_material.png				2021-04-15 00:21:33 KST	2021-04-15 00:21:33 KST	2021-04-14 16:21:08 KST	2021-04-14 16:21:08 KST
printer_profile_cura_speed.png				2021-04-15 00:21:33 KST	2021-04-15 00:21:33 KST	2021-04-14 16:21:13 KST	2021-04-14 16:21:13 KST
printer_profile_cura_travel.png				2021-04-15 00:21:33 KST	2021-04-15 00:21:33 KST	2021-04-14 16:21:18 KST	2021-04-14 16:21:18 KST

[2-11] Autopsy : /home/pi/.cache/pip/http/c/f : cura unallocated file

From the following process descripted above **Ultimaker Cura** is highly suspected to be used as a slicer. The following example Figure [2-12] is slicing `380_barrel_(threaded).stl` which is downloaded from github. Ultimaker Cura was used.



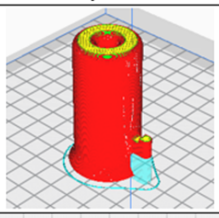
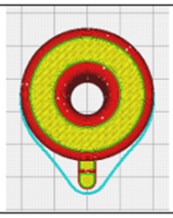
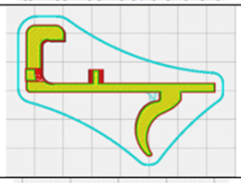
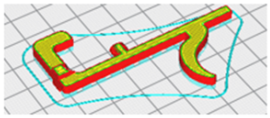
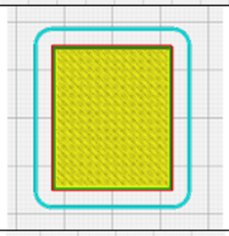
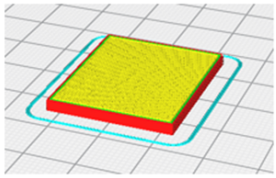
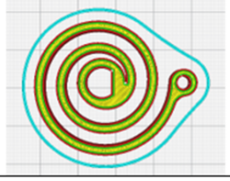
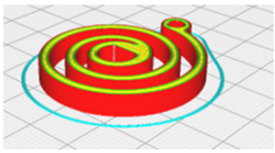
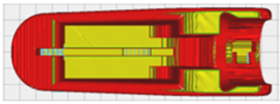
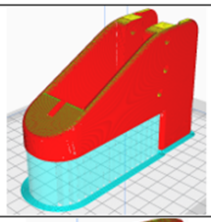
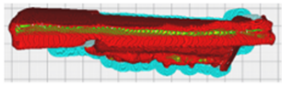
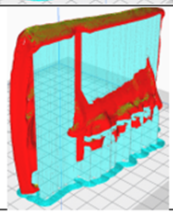
[2-12] Ultimaker Cura : stl file slice example



[2-13] Ultimaker Cura : slicing then transformed into a gcode

By using Ultimaker Cura, Figure [2-13] is the converted gcode file after slicing. “Save to Disk” in the bottom right saves the file with name “VK_380_barrel_(threaded).gcode” which has the same name format found in upload folder.

Table [2-2] show that the illicit thing printed by arnimzola was the parts used to make liberator. Files were listed in Table [2-2], which was created in **2021-04-13 ~ 2021-04-14**. The date is presumed to be the time when 3D printer was used.

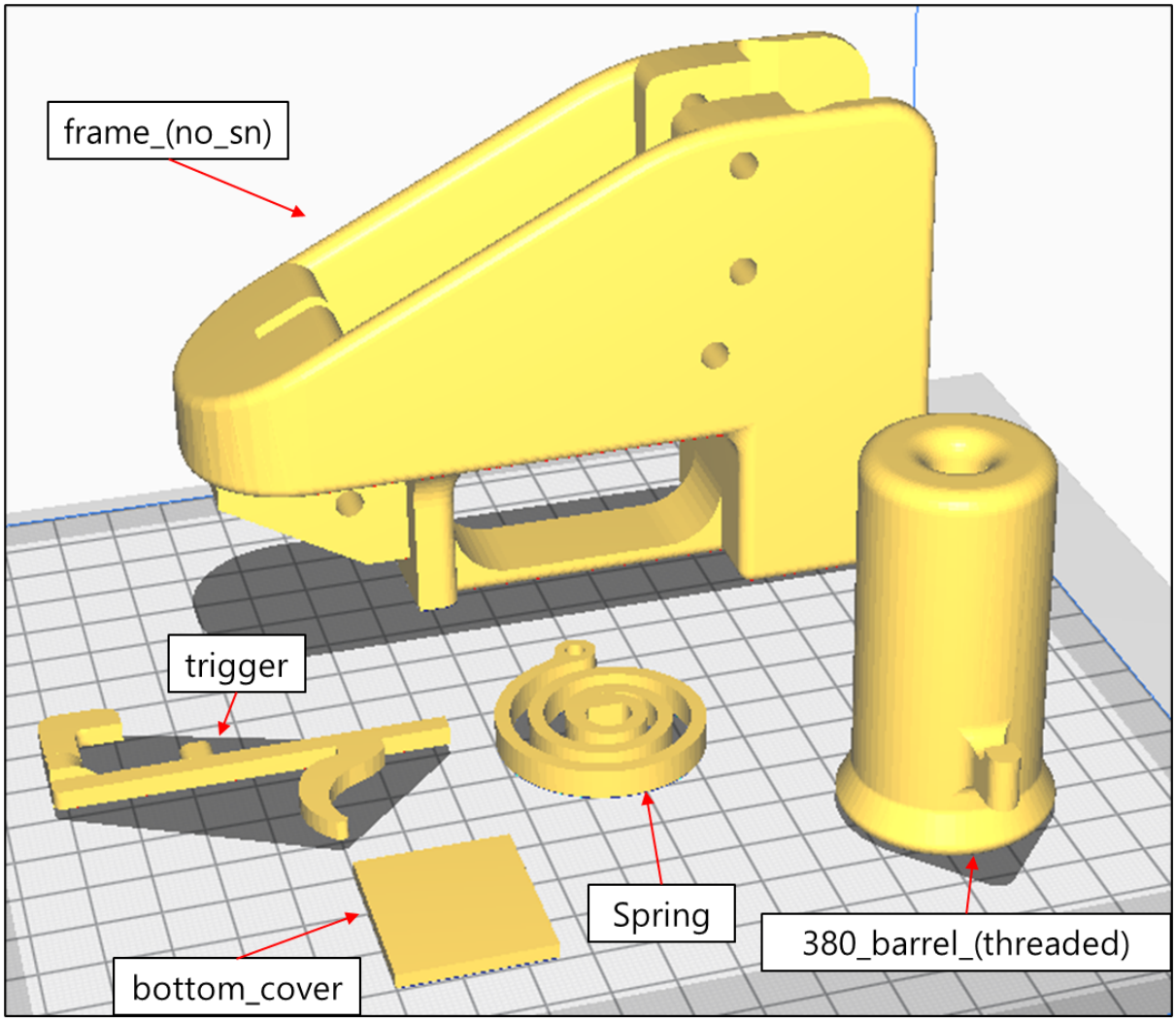
File name	Top View	3D View
VK_380_barrel_(threaded).gcode		
VK_trigger.gcode		
VK_bottom_cover.gcode		
VK_Spring.gcode		
VK_frame_(no_sn).gcode		
VK_2050938e-3012-4227-9a47-c4c4f5ab1e18.gcode		

[Table 2-2] gcode View

[Table 2-2] gcode view

File name	Top View	3D View
VK_380_barrel_(threaded).gcode		
VK_trigger.gcode		
VK_bottom_cover.gcode		
VK_Spring.gcode		
VK_frame_(no_sn).gcode		
VK_2050938e-3012-4227-9a47-c4c4f5ab1e18.gcode		

Figure [2-14] explains existing gcode files and stl file are identical. Stl files were created with Ultimaker Cura.



[2-14] stl file with similar form to gcode

But, no other stl formatted file was present that matches with “VK_2050938e-3012-4227-9a47-c4c4f5ab1e18.gcode”.

#4. .metadata.json

Octoprint records timestamps that was created while printing gcode files such as “Printing Area”, “dimensions”, “PrintTime” and “Event Timestamp” into metadat.json file Figure [2-15] that was found in Figure [2-8]. Metadata for each gcode files were found in metadata.json.

```
{
    "VK_trigger.gcode": {
        "statistics": {
            "averagePrintTime": {
                "_default": 892.5423338669989
            },
            "lastPrintTime": {
                "_default": 892.5423338669989
            }
        },
        "hash": "aadf142c913673561fee9039cb4e7b6d724277fb",
        "analysis": {
            "printingArea": {
                "maxZ": 10.699999809265137,
                "maxX": 135.85800170898438,
                "maxY": 126.05400085449219,
                "minX": 64.13899993896484,
                "minY": 73.95500183105469,
                "minZ": 0.20000000298023224
            },
            "dimensions": {
                "width": 71.71900177001953,
                "depth": 52.0989990234375,
                "height": 10.499999806284904
            },
            "compensatedPrintTime": 929.1151214688459,
            "analysisPending": false,
            "estimatedPrintTime": 929.1151214688459,
            "analysisPrintTime": 873.0214303748251,
            "analysisLastFilamentPrintTime": 852.9858522491529,
            "firstFilament": 0.0019283153450842,
            "progress": [
                [
                    0,
                    929.1151214688459
                ],
```


[2-15] /home/pi/.octoprint/uploads/.metadata.json

#5. octoprint.log

`/home/pi/.octoprint/logs/octoprint.log` shows logs that Octoprint recorded.

```
2021-04-13 10:38:24,021 - octoprint.environment - INFO - Detected environment is Python 2.7.13 under Linux (linux2). Details:
|  hardware:
|      cores: 4
|      freq: 1200.0
|      ram: 917016576
|  os:
|      bits: 32
|      id: linux
|      platform: linux2
|  plugins:
|      pi_support:
|          model: Raspberry Pi 3 Model B Rev 1.2
|          octopi_version: 0.15.1
|          throttle_state: '0x0'
|  python:
|      pip: 9.0.3
|      version: 2.7.13
|      virtualenv: /home/pi/oprint

...(omit)...


2021-04-13 10:55:07,961 - octoprint.plugins.discovery - INFO - Registered 'OctoPrint instance "Cthulhuuuu's 3D Printer"._http._tcp.
local.' for _http._tcp.local.

...(omit)...

2021-04-13 10:55:07,171 - octoprint.util.comm - INFO - Serial detection: Performing autodetection with 1 port/baudrate candidates:
 /dev/ttyUSB0@250000
```

Because Octoprint was to boot Raspberry Pi, information about environments are also found in `octoprint.log` such as “Hardware - Raspberry Pi”, “OS”, “Plugins -Octoprint” and “Python version to download Octoprint”. Following information was obtained in octoprint.log

- **Model name : Raspberry Pi 3 Model B Rev 1.2**

- **Printer name : Cthulhuuuu's 3D Printer**
- **Network ports : /dev/ttyUSB0**

```
2021-04-13 10:46:58,115 - octoprint.server.util.sockjs - INFO - New connection from client: ::ffff:172.22.30.100
2021-04-13 10:47:12,615 - octoprint.access.users - INFO - Logged in user: arnimzola
2021-04-13 10:47:12,623 - octoprint.server.api - INFO - Actively logging in user arnimzola from ::ffff:172.22.30.100
2021-04-13 10:47:12,759 - octoprint.server.util.flask - INFO - Passively logging in user arnimzola from ::ffff:172.22.30.100
2021-04-13 10:47:12,761 - octoprint.access.users - INFO - Logged in user: arnimzola
2021-04-13 10:47:13,283 - octoprint.server.util.flask.PreemptiveCache - INFO - Adding entry for / and {u'query_string': u'l10n=en',
u'path': u'/', u'_count': 1, u'_timestamp': 1618310833.283701, u'base_url': u'http://172.22.30.227/'}
2021-04-13 10:47:27,796 - octoprint.server.util.sockjs - INFO - Client connection closed: ::ffff:172.22.30.100
2021-04-13 10:47:28,552 - octoprint.server.util.sockjs - INFO - New connection from client: ::ffff:172.22.30.100
2021-04-13 10:47:28,622 - octoprint.server.util.flask - INFO - Passively logging in user arnimzola from ::ffff:172.22.30.100
2021-04-13 10:47:28,623 - octoprint.access.users - INFO - Logged in user: arnimzola
```

> ### 2021-04-13 10:47:28,622 - octoprint.server.util.flask - INFO - Passively logging in user arnimzola from ::ffff:172.22.30.100

From the log mentioned above, the IP of the user that access the server is 172.22.30.100.

Arnimzola IP address : 172.22.30.100

Other events found in `/home/pi/.octoprint/logs/octoprint.log` are organized in Table [2-3].

[Table 2-3] octoprint Event View

Aa Date (UTC+2)	Event	Log
2021-04-13 12:38:23,748	Octoprint server heartbeat	octoprint.server.heartbeat - INFO - Server heartbeat <3
2021-04-13 12:54:45,653	Start OctoPrint	octoprint.startup - INFO - Starting OctoPrint 1.5.1
2021-04-13 12:54:55,323	Initializing the file metadata for uploads folder	octoprint.filemanager.storage - INFO - Initializing the file metadata for /home/pi/.octoprint/uploads...
2021-04-13 12:55:17,784	Listening on Ocroprint Server	octoprint.server - INFO - Listening on http://127.0.0.1:5000
2021-04-13 12:58:22,433	Print job selected : VK_Switch.STL.gcode	octoprint.printer.standard.job - INFO - Print job selected - origin: local, path: VK_Switch.STL.gcode, owner: arnimzola, user: arnimzola
2021-04-13 14:27:57,248	Starting analysis of VK_380_barrel_(threaded).gcode	octoprint.filemanager.analysis - INFO - Starting analysis of local:VK_380_barrel_(threaded).gcode
2021-04-13 14:28:02,130	VK_380_barrel_(threaded).gcode analysis finished	octoprint.filemanager.analysis - INFO - Analysis of entry local:VK_380_barrel_(threaded).gcode finished, needed 4.88s
2021-04-13 14:28:03,814	Print job seleted : VK_380_barrel_(threaded).gcode	octoprint.printer.standard.job - INFO - Print job selected - origin: local, path: VK_380_barrel_(threaded).gcode, owner: arnimzola, user: arnimzola
2021-04-13 14:28:03,849	Print job started : VK_380_barrel_(threaded).gcode	octoprint.printer.standard.job - INFO - Print job started - origin: local, path: VK_380_barrel_(threaded).gcode, owner: arnimzola, user: arnimzola
2021-04-13 17:10:39,334	Starting analysis of VK_trigger.gcode	octoprint.filemanager.analysis - INFO - Starting analysis of local:VK_trigger.gcode
2021-04-13 17:10:40,943	VK_trigger.gcode analysis finished	octoprint.filemanager.analysis - INFO - Analysis of entry local:VK_trigger.gcode finished, needed 1.61s
2021-04-13 17:10:43,839	Print job selected	octoprint.printer.standard.job - INFO - Print job selected - origin: local, path: VK_trigger.gcode, owner: arnimzola, user: arnimzola
2021-04-13 17:10:43,867	Print job started : VK_trigger.gcode	octoprint.printer.standard.job - INFO - Print job started - origin: local, path: VK_trigger.gcode, owner: arnimzola, user: arnimzola
2021-04-13 17:14:09,536	Print job Cancelling : VK_trigger.gcode	octoprint.util.comm - INFO - Cancelling job on behalf of user arnimzola

Date (UTC+2)	Event	Log
2021-04-13 17:17:29,407	Starting analysis of VK_trigger.gcode	octoprint.filemanager.analysis - INFO - Starting analysis of local:VK_trigger.gcode
2021-04-13 17:17:29,551	Print job selected : VK_trigger.gcode	octoprint.printer.standard.job - INFO - Print job selected - origin: local, path: VK_trigger.gcode, owner: arnimzola, user: arnimzola
2021-04-13 17:17:30,831	VK_trigger.gcode analysis finished	octoprint.filemanager.analysis - INFO - Analysis of entry local:VK_trigger.gcode finished, needed 1.43s
2021-04-13 17:17:31,825	Print job selected : VK_trigger.gcode	octoprint.printer.standard.job - INFO - Print job selected - origin: local, path: VK_trigger.gcode, owner: arnimzola, user: arnimzola
2021-04-13 17:17:31,863	Print job started : VK_trigger.gcode	octoprint.printer.standard.job - INFO - Print job started - origin: local, path: VK_trigger.gcode, owner: arnimzola, user: arnimzola
2021-04-13 17:32:24,387	Print job done : VK_trigger.gcode	octoprint.printer.standard.job - INFO - Print job done - origin: local, path: VK_trigger.gcode, owner: arnimzola
2021-04-13 17:43:35,590	Starting analysis : VK_bottom_cover.gcode	octoprint.filemanager.analysis - INFO - Starting analysis of local:VK_bottom_cover.gcode
2021-04-13 17:43:36,721	VK_bottom_cover.gcode analysis finished	octoprint.filemanager.analysis - INFO - Analysis of entry local:VK_bottom_cover.gcode finished, needed 1.13s
2021-04-13 17:43:37,856	Print job selected : VK_bottom_cover.gcode	INFO - Print job selected - origin: local, path: VK_bottom_cover.gcode, owner: arnimzola, user: arnimzola
2021-04-13 17:43:37,884	Print job started : VK_bottom_cover.gcode	octoprint.printer.standard.job - INFO - Print job started - origin: local, path: VK_bottom_cover.gcode, owner: arnimzola, user: arnimzola
2021-04-13 17:46:46,783	Print job cancelled : VK_bottom_cover.gcode, owner	octoprint.util.comm - INFO - Cancelling job on behalf of user arnimzola
2021-04-13 17:48:25,903	Starting analysis : VK_bottom_cover.gcode	octoprint.filemanager.analysis - INFO - Starting analysis of local:VK_bottom_cover.gcode
2021-04-13 17:48:26,052	Print job selected : VK_bottom_cover.gcode	octoprint.printer.standard.job - INFO - Print job selected - origin: local, path: VK_bottom_cover.gcode, owner: arnimzola, user: arnimzola
2021-04-13 17:48:26,820	VK_bottom_cover.gcode analysis finished	octoprint.filemanager.analysis - INFO - Analysis of entry local:VK_bottom_cover.gcode finished, needed 0.92s
2021-04-13 17:48:27,880	Print job selected : VK_bottom_cover.gcode	octoprint.printer.standard.job - INFO - Print job selected - origin: local, path: VK_bottom_cover.gcode, owner: arnimzola, user: arnimzola
2021-04-13 17:48:27,907	Print job started : VK_bottom_cover.gcode	octoprint.printer.standard.job - INFO - Print job started - origin: local, path: VK_bottom_cover.gcode, owner: arnimzola, user: arnimzola
2021-04-13 18:01:44,564	Print job done : VK_bottom_cover.gcode	octoprint.printer.standard.job - INFO - Print job done - origin: local, path: VK_bottom_cover.gcode, owner: arnimzola
2021-04-13 18:22:57,864	Starting analysis :VK_Spring.gcode	octoprint.filemanager.analysis - INFO - Starting analysis of local:VK_Spring.gcode
2021-04-13 18:23:00,395	VK_Spring.gcode analysis finished	octoprint.filemanager.analysis - INFO - Analysis of entry local:VK_Spring.gcode finished, needed 2.53s
2021-04-13 18:23:01,808	Print job selected : VK_Spring.gcode	octoprint.printer.standard.job - INFO - Print job selected - origin: local, path: VK_Spring.gcode, owner: arnimzola, user: arnimzola
2021-04-13 18:23:01,829	Print job started : VK_Spring.gcode	octoprint.printer.standard.job - INFO - Print job started - origin: local, path: VK_Spring.gcode, owner: arnimzola, user: arnimzola
2021-04-13 18:50:59,282	Print job done : VK_Spring.gcode	octoprint.printer.standard.job - INFO - Print job done - origin: local, path: VK_Spring.gcode, owner: arnimzola
2021-04-14 08:26:16,580	Starting analysis :VK_frame_(no_sn).gcode	octoprint.filemanager.analysis - INFO - Starting analysis of local:VK_frame_(no_sn).gcode
2021-04-14 08:26:16,727	Print job selected : VK_frame_(no_sn).gcode	octoprint.printer.standard.job - INFO - Print job selected - origin: local, path: VK_frame_(no_sn).gcode, owner: arnimzola, user: arnimzola
2021-04-14 08:26:16,765	Print job started : VK_frame_(no_sn).gcode	octoprint.printer.standard.job - INFO - Print job started - origin: local, path: VK_frame_(no_sn).gcode, owner: arnimzola, user: arnimzola
2021-04-14 08:26:25,645	VK_frame_(no_sn).gcode analysis finished	octoprint.filemanager.analysis - INFO - Analysis of entry local:VK_frame_(no_sn).gcode finished, needed 9.07s
2021-04-14 16:28:28,529	Print job done : VK_frame_(no_sn).gcode	octoprint.printer.standard.job - INFO - Print job done - origin: local, path: VK_frame_(no_sn).gcode, owner: arnimzola

Date (UTC+2)	Event	Log
2021-04-14 16:55:23,751	Print job selected : VK_Spring.gcode	octoprint.printer.standard.job - INFO - Print job selected - origin: local, path: VK_Spring.gcode, owner: arnimzola, user: arnimzola
2021-04-14 17:01:51,024	Print job started : VK_Spring.gcode	octoprint.printer.standard.job - INFO - Print job started - origin: local, path: VK_Spring.gcode, owner: arnimzola, user: arnimzola
2021-04-14 17:29:45,776	Print job done : VK_Spring.gcode	octoprint.printer.standard.job - INFO - Print job done - origin: local, path: VK_Spring.gcode, owner: arnimzola
2021-04-14 22:37:01,823	Starting analysis :VK_2050938e-3012-4227-9a47-c4c4f5ab1e18.gcode	octoprint.filemanager.analysis - INFO - Starting analysis of local:VK_2050938e-3012-4227-9a47-c4c4f5ab1e18.gcode
2021-04-14 22:37:02,016	Print job selected : VK_2050938e-3012-4227-9a47-c4c4f5ab1e18.gcode	octoprint.printer.standard.job - INFO - Print job selected - origin: local, path: VK_2050938e-3012-4227-9a47-c4c4f5ab1e18.gcode, owner: arnimzola, user: arnimzola
2021-04-14 22:37:02,083	Print job started : VK_2050938e-3012-4227-9a47-c4c4f5ab1e18.gcode	octoprint.printer.standard.job - INFO - Print job started - origin: local, path: VK_2050938e-3012-4227-9a47-c4c4f5ab1e18.gcode, owner: arnimzola, user: arnimzola
2021-04-14 22:37:21,406	VK_2050938e-3012-4227-9a47-c4c4f5ab1e18.gcode analysis finished	octoprint.filemanager.analysis - INFO - Analysis of entry local:VK_2050938e-3012-4227-9a47-c4c4f5ab1e18.gcode finished, needed 19.58s

The following list is gcode file names printed with 3D printer in Table [2-3]

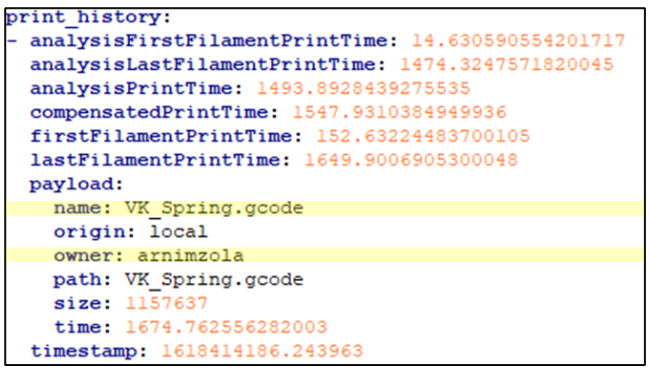
- VK_380_barrel_(threaded).gcode
- VK_trigger.gcode
- VK_bottom_cover.gcode
- VK_Spring.gcode
- VK_frame_(no_sn).gcode
- VK_2050938e-3012-4227-9a47-c4c4f5ab1e18.gcode **(Not Printed)**

For the file “VK_2050938e-3012-4227-9a47-c4c4f5ab1e18.gcode”, only the analysis process was done, but there was no start/end sequence processed. Results about analysis process was saved in `/uploads` directory.

#6. print_history.yaml

In `/home/pi/.octoprint/data/PrintTimeGenius/print_history.yaml`, following information was found. Figure [2-18] is part of print_history.yaml file.

- Printed file name(Highlighted)
- Owner(Highlighted)
- Path
- Analysis time
- Print time



[2-18] /home/pi/.octoprint/data/PrintTimeGenius/print_history.yaml

#7. octoprint_stats.json

`/home/pi/.octoprint/logs/octoprint_stats.json` file shows “Event Type” and “Timestamp” from the gcode file. octoprint_stats.json looks similar with octoprint.log, but it is explained by Event type, which gives more detail information. A sample image can be seen in Figure [2-19]. Table [2-4] shows events that occurred in **2021-04-13 ~ 2021-04-14** from octoprint_stats.json.


```
"399": {
    "data": {
        "origin": "local",
        "tool1_target": 0,
        "name": "VK_380_barrel_(threaded).gcode",
        "event_time": "2021-04-13 12:28:03.978985",
        "bed_target": 0.0,
        "tool2_target": 0,
        "user": "arnimzola",
        "file": "VK_380_barrel_(threaded).gcode",
        "owner": "arnimzola",
        "tool0_target": 0.0,
        "size": 3538731
    },
    "event_type": "PRINT_STARTED"
},
"400": {
    "data": {
        "origin": "local",
        "bed_actual": 0.0,
        "ptime": 5096.09778866,
        "event_time": "2021-04-13 13:53:00.404119",
        "tool1_volume": 0,
        "tool2_length": 0,
        "name": "VK_380_barrel_(threaded).gcode",
        "tool2_actual": 0,
        "tool0_volume": 32.777773970642166,
        "file": "VK_380_barrel_(threaded).gcode",
        "tool0_actual": 193.7,
        "owner": "arnimzola",
        "tool1_actual": 0,
        "size": 3538731,
        "tool2_volume": 0,
        "tool1_length": 0,
        "tool0_length": 4637.1064453125
    },
    "event_type": "PRINT_DONE"
},
"401": {
    "data": {
        "event_time": "2021-04-13 15:10:39.378093",
        "name": "VK_trigger.gcode",
        "file": "VK_trigger.gcode",
        "target": "local"
    },
    "event_type": "UPLOAD"
```


[2-19] /home/pi/.octoprint/logs/octoprint_stats.json

Table [2-4]. octoprint_stats.json

File name	속성	event_time (UTC+2)
VK_380_barrel_(threaded).gcode	UPLOAD	2021-04-13 14:27:57.389504
VK_380_barrel_(threaded).gcode	PRINT_STARTED	2021-04-13 14:28:03.978985
VK_380_barrel_(threaded).gcode	PRINT_DONE	2021-04-13 15:53:00.404119
VK_trigger.gcode	UPLOAD	2021-04-13 17:10:39.378093
VK_trigger.gcode	PRINT_STARTED	2021-04-13 17:10:43.906148
VK_trigger.gcode	PRINT_CANCELLED	2021-04-13 17:14:09.634281
VK_trigger.gcode	PRINT_FAILED	2021-04-13 17:14:10.216089
VK_trigger.gcode	UPLOAD	2021-04-13 17:17:29.628354
VK_trigger.gcode	PRINT_STARTED	2021-04-13 17:17:32.016423
VK_trigger.gcode	PRINT_DONE	2021-04-13 17:32:24.911512

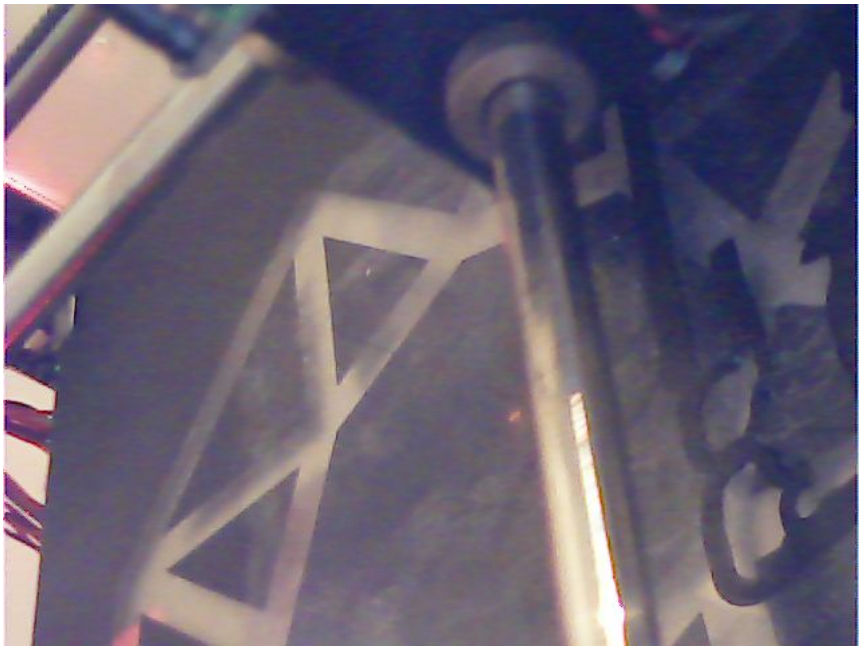
File name	속성	event_time (UTC+2)
VK_bottom_cover.gcode	UPLOAD	2021-04-13 17:43:35.738022
VK_bottom_cover.gcode	PRINT_STARTED	2021-04-13 17:43:37.971102
VK_bottom_cover.gcode	PRINT_CANCELLED	2021-04-13 17:46:46.861340
VK_bottom_cover.gcode	PRINT_FAILED	2021-04-13 17:46:47.109851
VK_bottom_cover.gcode	UPLOAD	2021-04-13 17:48:26.109604
VK_bottom_cover.gcode	PRINT_STARTED	2021-04-13 17:48:28.065077
VK_bottom_cover.gcode	PRINT_DONE	2021-04-13 18:01:45.028063
VK_Spring.gcode	UPLOAD	2021-04-13 18:22:57.896580
VK_Spring.gcode	PRINT_STARTED	2021-04-13 18:23:01.858722
VK_Spring.gcode	PRINT_DONE	2021-04-13 18:50:59.858228
VK_frame_(no_sn).gcode	UPLOAD	2021-04-14 08:26:16.921332
VK_frame_(no_sn).gcode	PRINT_STARTED	2021-04-14 08:26:17.139730
VK_frame_(no_sn).gcode	PRINT_DONE	2021-04-14 16:28:29.087254
VK_Spring.gcode	PRINT_STARTED	2021-04-14 17:01:51.151617
VK_Spring.gcode	PRINT_DONE	2021-04-14 17:29:46.340420
VK_2050938e-3012-4227-9a47-c4c4f5ab1e18.gcode	PRINT_STARTED	2021-04-14 22:37:02.155042
VK_2050938e-3012-4227-9a47-c4c4f5ab1e18.gcode	UPLOAD	2021-04-14 22:37:02.402413

Additionally, `/home/pi/.octoprint/timelapse/` directory shows the timelapse video recorded by webcam connected via USB port. Table [2-5] shows the timelapse vidoe of printed files by 3D printer during 2021-04-13 ~ 2021-04-14.

Table [2-5] Timelapse List

Num	File name
1	VK_380_barrel_(threaded)_20210413122803.mp4
2	VK_trigger_20210413151043-fail.mp4
3	VK_trigger_20210413151731.mp4
4	VK_bottom_cover_20210413154337-fail.mp4
5	VK_bottom_cover_20210413154828.mp4
6	VK_Spring_20210413162301.mp4
7	VK_frame_(no_sn)_20210414062616.mp4
8	VK_Spring_20210414150151.mp4

Video of “VK_2050938e-3012-4227-9a47-c4c4f5ab1e18.gcode” was not found. However in `/home/pi/.octoprint/timelapse/tmp/` directory, a jpg file named “VK_2050938e-3012-4227-9a47-c4c4f5ab1e18_20210414ttttmm-nn.jpg” was found.



[2-20] /home/pi/.octoprint/timelapse/tmp/ : .jpg related VK_2050938e-3012-4227-9a47-c4c4f5ab1e18.gcode

Since there is no “Print_Done” log in “VK_2050938e-3012-4227-9a47-c4c4f5ab1e18.gcode” file in Table [2-3], Table [2-4] and no video in Table [2-5], this file was not completely printed.

[Conclusion]

Analysis of SD cards found in Raspberry Pi shows that Raspberry Pi was used to control 3D printers and that parts of Liberator, a gun, were printed on 2021/04/13 14:27:57 - 2021/04/14 17:29:46.340420. octoprint_stats.json file proves this. Table [2-6], component required for liberator printed time.

[Table 2-6] Liberator printed time

File name	Upload Time	Print_Done Time
VK_380_barrel_(threaded).gcode	2021-04-13 14:27:57.389504	2021-04-13 15:53:00.404119
VK_trigger.gcode	2021-04-13 17:10:39.378093	2021-04-13 17:32:24.911512
VK_bottom_cover.gcode	2021-04-13 17:43:35.738022	2021-04-13 18:01:45.028063
VK_frame_(no_sn).gcode	2021-04-14 08:26:16.921332	2021-04-14 16:28:29.087254
VK_Spring.gcode	2021-04-13 18:22:57.896580	2021-04-14 17:29:46.340420

As a result, “arnimzola” who is the user of Raspberry Pi, used Raspberry Pi to control 3D printer. Which printed illicit “**Liberator**” at “**2021-04-13 14:27:57.389504 ~ 2021-04-14 17:29:46.340420”**.

Part 3. Samsung Smartphone

[Introduction]

Summary of analysis target information is Table [3-1].

[Table 3-1] Taget Smartphone Information

Name	Description
Model	SM-G973F
OS Version	Android 10
Phone Number	+41786909109
E-mail	vonstruckerwerner@gmail.com
IMEI	356163113740928

[Description]

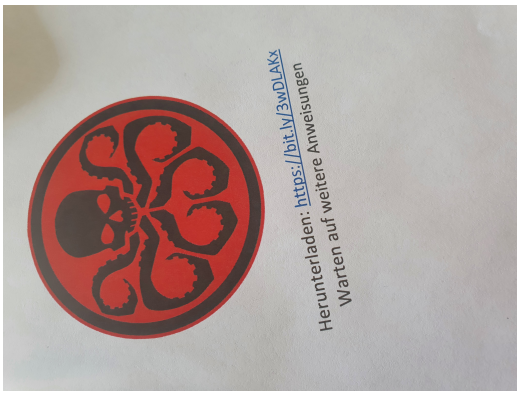
The main keyword given in this part was “encoded SMS exchanges”. Thus, the first phase was to identify applications having message exchange functions. As a result, Android’s default SMS database file stored in `/data/data/com.android.providers.telephony/databases/mmssms.db` was identified as a target for forensic analysis, because it contains encoded messages prefixed with “HHY” signature as shown in Figure [3-1]. As an initial attempt, well-known encoding algorithms were applied to decode them, but the decoding was not successful with any known algorithms. Therefore, it was necessary to identify and reverse-engineer S/W that was used for recording encoded messages in the default SMS database.

address	date	body	creator
+41786909109	2021-04-08 18:09:21	HHY6cKEMFlNU3NPs/…	it.feio.android.omninotes
+41786909109	2021-04-08 18:10:33	HHYZEuQn5pA1hKmulAHVx55cA==	com.samsung.android.messaging
+41786909109	2021-04-08 18:11:08	HHYu1+uAKNdKlj0YsLDmmJw3M2bFrI0/…	it.feio.android.omninotes
+41786909109	2021-04-08 18:12:51	HHYKvEbze+wdEalheouf7A2m98+…	com.samsung.android.messaging
+41786909109	2021-04-08 18:14:13	HHYsskk9VxsMQLIQ5jsWVGKu0cikDEZYDdgJ…	com.samsung.android.messaging
+41786909109	2021-04-08 18:16:54	HHYbRTTaGxxLatLwNLMF7ptPDTjpMnxkpyp5…	it.feio.android.omninotes
+41786909109	2021-04-09 16:10:10	HHYCIUGa/i7pLYjRGwzXrvpVg==	it.feio.android.omninotes
+41786909109	2021-04-09 16:11:13	HHYCIUGa/i7pLYjRGwzXrvpVg==	com.samsung.android.messaging
+41786909109	2021-04-09 16:13:53	HHYBNsxyjLdeJ2KD5cKwnO1A81zVE6Xad+i+…	it.feio.android.omninotes
+41786909109	2021-04-09 16:15:16	HHYoD/v/R9xC1A8TqJ/UWCf/…	com.samsung.android.messaging
+41786909109	2021-04-15 07:49:43	HHY2NToA10DeweA+V3mutF9QA==	com.samsung.android.messaging
+41786909109	2021-04-15 07:50:11	HHY2NToA10DeweA+V3mutF9QA==	it.feio.android.omninotes
+41786909109	2021-04-15 07:51:05	HHYHiiOyvP7s0llu5N2ta9y/…	com.samsung.android.messaging
+41786909109	2021-04-15 07:52:48	HHYOeHfDhFU8plFmx7J5qXltG1XGwpcbrBtUj…	it.feio.android.omninotes
+41786909109	2021-04-15 08:02:17	HHYiNf68BXbaolSBtAav5yYU7WHDHaADyWmt…	com.samsung.android.messaging
+41786909109	2021-04-15 08:03:13	HHYbm0jyH9MsblM1DsyvNU9lDm63RQa9/1Fx…	com.samsung.android.messaging
+41786909109	2021-04-15 08:05:00	HHYS6+p9lajgCnkXlm0touuBzw/…	it.feio.android.omninotes
+41786909109	2021-04-15 08:13:37	HHYcdZD5nSpEbiNiPHaMn/…	com.samsung.android.messaging
+41786909109	2021-04-15 08:16:19	HHY9frkML1o7eM6Ri32ERhNUEnmlbDg1syRQ…	it.feio.android.omninotes
+41786909109	2021-04-15 08:17:08	HHY2NToA10DeweA+V3mutF9QA==	com.samsung.android.messaging
+41786909109	2021-04-15 10:54:54	HHY6JGyTGtl0ljl8PgtU2T/w5YPNCw1PhZ/…	it.feio.android.omninotes
SIGNAL	2021-04-18 15:08:35	SIGNAL: Your code is: 194-892…	com.samsung.android.messaging
SAUTH	2021-04-18 15:16:46	<#> Account: 811182 is your Samsung acco…	com.samsung.android.messaging
+41786909109	2021-04-18 16:00:47	HHYj6DtE+EQdYzoKjUoVP8DQw==	com.samsung.android.messaging
+41786909109	2021-04-18 16:01:08	HHYj6DtE+EQdYzoKjUoVP8DQw==	it.feio.android.omninotes
506	2021-04-18 16:01:27	Cher client, votre crédit disponible est inférie…	com.samsung.android.messaging
+41786909109	2021-04-18 16:02:10	HHYqDAJ2HSbMahjfWcjFcL8H41ALcW17Jkzd…	com.samsung.android.messaging
+41786909109	2021-04-18 16:02:56	HHYRvZdXk+/…	it.feio.android.omninotes
+41786909109	2021-04-18 16:05:14	HHY97NA5dDGdqomydjZt0GGycKgRS2T2vltp…	com.samsung.android.messaging
+41786909109	2021-04-18 16:06:46	HHYgSQ2GGQd5ZEvKdOk5qRrWfoEH0baqjRk…	it.feio.android.omninotes
0766013486	2021-04-18 16:30:22	Hey Jackie! Hope you alright! Thanks for the …	com.samsung.android.messaging

[3-1] Encoded messages stored in /data/data/com.android.providers.telephony/databases/mmssms.db

The path of the file is `DFRC_Appendix.zip/Part3.Samsung Smartphone/[3-1] mmssms.db`. Also emblem was found as Figure [3-2], which was found in other devices. accessing to the URL in the image via safebox environment, OmniNotes-playRelease-6.1.0 Alpha

1.apk was downloaded. the package name was `it.feio.android.omninotes` which had the identical package name seen in mmsms.db.



[3-2] /data/media/0/DCIM/Camera/20210408_1821133.jpg

Package name that were generating these message formats were `it.feio.android.omninotes` and `com.samsung.android.messaging`. In order to find out how these format is created, packages were extracted and reverse engineered. First, `AndroidManifest.xml` was analyzed to find permission of the application, to find out that it was able to access on SMS. Analyzed applications were omninote and samsung message. Figure [3-3] is permissions that were granted to omninote. the original package that was uploaded on google playstore had no such permission about message Read/Write. Which was the main reason that omninote was the target application to send encoded sms.

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.VIBRATE"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_SMS"/>
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="com.android.alarm.permission.SET_ALARM"/>
<uses-permission android:name="com.google.android.apps.photos.permission.GOOGLE_PHOTOS"/>
<uses-permission android:name="com.android.launcher.permission.INSTALL_SHORTCUT"/>
<uses-permission android:name="com.android.launcher.permission.UNINSTALL_SHORTCUT"/>
<uses-permission android:name="com.pushbullet.android.permission.READ_MESSAGING_EXTENSION_DATA"/>
<uses-permission android:name="com.pushbullet.android.permission.SEND_MESSAGES"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
```

[3-3] AndroidManifest.xml of it.feio.android.omninotes

 # Message Application Analysis

Applications package list acquired was decompiled with **jadx**. With the decompiled source code, MainActivity class was the first thing to analyze, because it was the main screen of the application. While looking for the point where the code can proceed to another activity, a suspicious method Figure [3-4] was found which can access to CommActivity. This method executes a hidden activity if “open_comm_counter” was clicked 5 times after it is set initially.

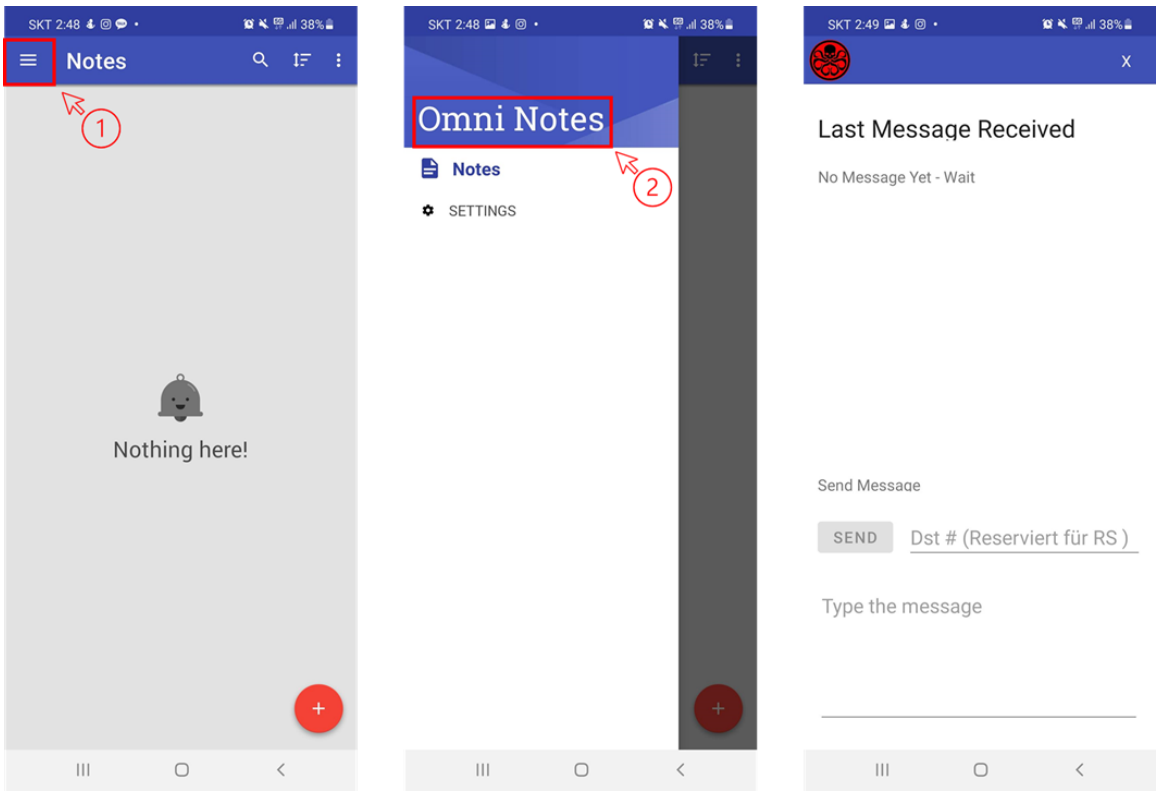
```
public void openComm(View view) {
        String str = "Open Comm in -" + String.valueOf(this.open_comm_counter);
        if (this.open_comm_counter == 0) {
            this.open_comm_counter = 5;
            startActivity(new Intent(this, CommActivity.class));
```

```
        }
        this.open_comm_counter--;
    }
```

Looking for condition that “open_comm_counter” variable decreases, it was found that `openComm` method Figure [3-5] can be called by “Navigation” under MainActivity.

```
android:text="@string/app_name" android:layout_alignParentBottom="true" android:onClick="openComm"

              <string name="app_name">Omni Notes</string>
```

Experiments were done by clicking the string “Omni Notes” 5 times. Deep understanding of “CommActivity” was needed to solve the problem.



[3-6] Hidden Activity

When `onCreate` is executed in “CommActivity”, the 3rd image of Figure [3-6] (from left) is created. `checkAndRequestReadPermissions` method is used next to create message list due to Read permission. `checkAndRequestSendPermission` method is then used to find Send permission to enable “SEND” button. This logic can be found in Figure [3-7].

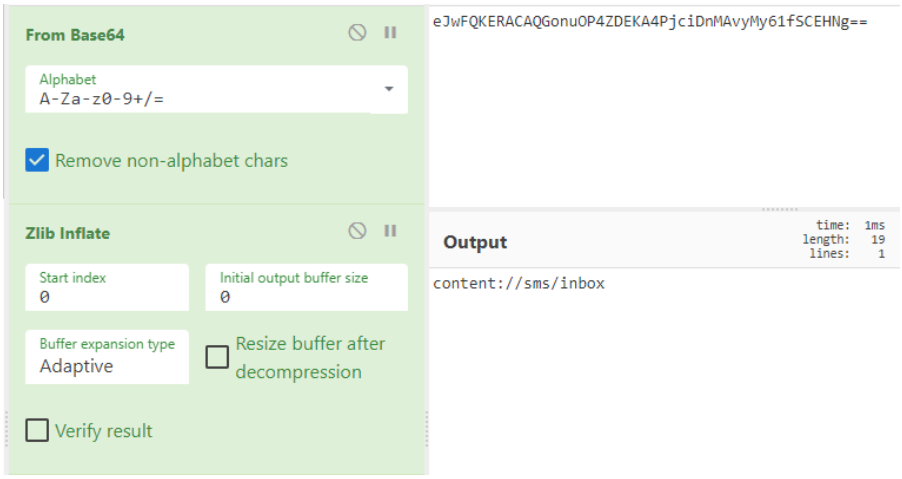
```
public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView(R.layout.activity_comm);
        StrictMode.setThreadPolicy(new StrictMode.ThreadPolicy.Builder().permitAll().build());
```

```
        if (checkAndRequestReadPermissions()) {
            listComm();
        }
        findViewById(R.id.button_send).setEnabled(checkAndRequestSendPermissions());
    }
```

Data was returned using “Content Provider”, the encoded strings were found out to be decompressed by ZipHelper classes “decomp” method. ZipHelper class is located in `it.feio.android.omninotes.utils` and the decomp method is shown as Figure [3-8].

```
public static String decomp(String str) {
        int i;
        byte[] decode = Base64.decode(str, 0);
        Inflater inflater = new Inflater();
        inflater.setInput(decode);
        byte[] bArr = new byte[1000];
        try {
            i = inflater.inflate(bArr);
        } catch (DataFormatException e) {
            e.printStackTrace();
            i = 0;
        }
        return new String(Arrays.copyOfRange(bArr, 0, i), StandardCharsets.UTF_8);
    }
```

`ZipHelper.decomp` method decodes the message with Base64 and decompresses with `Inflater.inflate (Zlib Inflate Algorithm)`. Decompressed bArr is encoded with UTF-8 and returned. The sample of decoding `eJwFQKERACAQGonuOP4ZDEKA4PjciDnMAvyMy61fSCEHNg==` is Figure [3-9].



[3-9] Decoded string as an example

In `listComm` Figure [3-10], Content Provider is used to load SMS history in inbox, Meanwhile brings messages if the message body starts with “HHY” signature.

```
private void listComm() {
        Cursor query = getContentResolver().query(Uri.parse(ZipHelper.decomp("eJwFQKERACAQGonuOP4ZDEKA4PjciDnMAvyMy61fSCEHNg==")),
 null, null, null, null);
        String str = "No Message Yet";
        if (query.moveToFirst()) {
            String str2 = BuildConfig.FLAVOR;
            boolean z = false;
            do {
```

```
                for (int i = 0; i < query.getColumnCount(); i++) {
                    str2 = str2 + " " + query.getColumnName(i) + ":" + query.getString(i);
                    if (query.getColumnName(i).equals("body")) {
                        String string = query.getString(i);
                        if (string.startsWith("HHY")) {
                            str = new TextHelper(getResources()).get(string);
                            z = true;
                        }
                    }
                }
            } while (query.moveToNext() & (!z));
            ((TextView) findViewById(R.id.text_view_msg_rec)).setText(str);
        }
        ((TextView) findViewById(R.id.text_view_msg_rec)).setText(str);
    }
```

`putComm` methods is called when the SEND button is clicked. Before analyzing `putComm` method, every string were decoded with the method given before. It was encoded, compressed with identical methods, decoded results are listed in Table [3-2].

```
<Button android:id="@+id/button_send" android:text="@string/send" android:onClick="putComm"
```

```
public void putComm(View view) {
        String put = new TextHelper(getResources()).put(((EditText) findViewById(R.id.text_multiline_msg_send)).getText().toString
());
        if (put.length() > 160) {
            Toast.makeText(this, ZipHelper.decomp("eJwFQLENwDAIe8UH5I1u6ZL2gQgs1AUPqEuuR1eanI6bVTuIV8JUxsCi/0Y832ED99UM+Q=="), 0).s
how();
        } else if (put.equals(BuildConfig.FLAVOR)) {
            Toast.makeText(this, ZipHelper.decomp("eJwFgLEJACAMBFfJMpaCK6R4QhqV+/mLsOARB/2kb8WWnaUBZ9oIuA=="), 0).show();
        } else {
            Toast.makeText(this, ZipHelper.decomp("eJwFQLEJADAMesU7+kOmXlCoSBYX7x/k0n8tDJMnHhQ48AYI") + put + " to " + rsn(), 0).sh
ow();
            PendingIntent.getActivity(this, 0, new Intent(this, CommActivity.class), 0);
            SmsManager.getDefault().sendTextMessage(rsn(), null, put, null, null);
        }
    }
```

```
private String rsn() {
        EditText editText = (EditText) findViewById(R.id.dst_person);
        if (editText.getText().toString().equals(BuildConfig.FLAVOR)) {
            return ZipHelper.decomp("eJwFgMEJAAAIAgfqoxDZ7b+YzDp/CIsCD7UCcg==");
        }
        return editText.getText().toString();
    }
```

[Table 3-2] Decoded Strings

Aa Encoded String	Decoded String
eJwFgMEJAAAIAgfqoxDZ7b+YzDp/CIsCD7UCcg==	+41786909109
eJwFQLENwDAIe8UH5I1u6ZL2gQgs1AUPqEuuR1eanI6bVTuIV8JUxsCi/0Y832ED99UM+Q==	Encoded Message Too Long, Reduce Size
eJwFgLEJACAMBFfJMpaCK6R4QhqV+/mLsOARB/2kb8WWnaUBZ9oIuA==	Error Preparing Message
eJwFQLEJADAMesU7+kOmXlCoSBYX7x/k0n8tDJMnHhQ48AYI	Sending Message:

With `put` method, if the message length written in the buffer is longer than 160 characters, toast message of “Encoded Message Too Long, Reduce Size” is printed. If error was caught, “Error Preparing Message” is printed in the same format. For normal inputs, “Sending: Message”, variable “put” which has the input string, and the receiver “+41786909109” in the toast message. Then it sends message with `sendTextMessage` method.

 # Message Encryption/Decryption Algorithm

The Class that it directly `encrypts/decrypts` the message is “TextHelper class of `it.feio.android.omninotes.helpers` . Table [3-3] is the results of strings decoded in TextHepler class.

[Table 3-3] Decoded Strings

Aa Encoded String
eJwFQLEJACEMXOWG0dLSAfLmEEHMk1g4fmiMkEnU9y+nwhz9yLeJaygcpkzk/QyW
eJwNyUsKwDAIBNATZSTbHietkH5wShUkt7frNyNe30Tm6hgPB3Z+ijztYDpM45eW3rrcunA5rQDMDhIV
eJwFwIEIAAAAACC0/a1OAEAAQA==
eJw9ylHLwiAUgOFfo3fC8ai5DbyJ2k0lwWfQvstG2GiVeKTfPyPo9nlfSi5zKg4BpQAtwAapO7PqwP5zun5dgoAmIP48ZXcrJRGn9+eoGevEKbuR0
eJwNwIEAAAAMhECiULJ4idm3u4eyBRWSA6E=

1. **Send Message**

When sending messages, `put` method is used. After the string is written, `save_msg` method is called and the output that is returned is Base64 encoded. Then, three characters “HHY” as a signature is concatenated in front of each “encrypted and encoded” message.

```
public String put(String str) {
        try {
            return "HHY" + Base64.encodeToString(save_msg(str), 2);
        } catch (IOException | InvalidKeyException | NoSuchAlgorithmException | InvalidParameterSpecException | BadPaddingException
| IllegalBlockSizeException | NoSuchPaddingException | JSONException unused) {
            return BuildConfig.FLAVOR;
        }
    }
```

```
private byte[] save_msg(String str) throws NoSuchAlgorithmException, NoSuchPaddingException, InvalidKeyException, InvalidParameterS
pecException, IllegalBlockSizeException, BadPaddingException, IOException, JSONException {
        SecretKeySpec secretKeySpec = new SecretKeySpec(get_data().getBytes(), "AES");
        Cipher instance = Cipher.getInstance("AES/ECB/PKCS5Padding");
        instance.init(1, secretKeySpec);
        return instance.doFinal(str.getBytes("UTF-8"));
    }
```

`save_msg` encrypts data via AES-ECB algorithm with `get_data` which reads the Key.

2. **Read Message**

When reading messages, `get` method is used to split “HHY” signature and calls `open_msg` method to decode the message. If error occurs, it returns “Message Expired or Unable to Decode”.

```
public String get(String str) {
        try {
            return open_msg(Base64.decode(str.substring(3), 2));
        } catch (IOException | InvalidAlgorithmParameterException | InvalidKeyException | NoSuchAlgorithmException | InvalidParamet
erSpecException | BadPaddingException | IllegalBlockSizeException | NoSuchPaddingException | JSONException unused) {
            return ZipHelper.decomp("eJwFQLEJACEMXOWG0dLSAfLmEEHMk1g4fmiMkEnU9y+nwhz9yLeJaygcpkzk/QyW");
        }
    }
```

`open_msg` method also calls `get_data` method to obtain key. With the key obtained, it decrypts the encrypted sting with AES-ECB algorithm.

```
private String open_msg(byte[] bArr) throws NoSuchPaddingException, NoSuchAlgorithmException, InvalidParameterSpecException, Invali
dAlgorithmParameterException, InvalidKeyException, BadPaddingException, IllegalBlockSizeException, IOException, JSONException {
        SecretKeySpec secretKeySpec = new SecretKeySpec(get_data().getBytes(), "AES");
        Cipher instance = Cipher.getInstance("AES/ECB/PKCS5Padding");
        instance.init(2, secretKeySpec);
        return new String(instance.doFinal(bArr), "UTF-8");
    }
```

3. **Key Extraction**

The method which gets the key while encrypting/decrypting messages is `get_data`. This method is Microsoft Azure Blob Storage server and if the encoded strings are decoded and concatenated, the url of the storage server is found. Decode url address is quoted.

> “https://hy1.blob.core.windows.net/hy-ws-1/key.json?sp=r&st=2021-04-07T14:56:07Z&se=2021-10-08T22:56:07Z&spr=https&sv=2020-02-10&sr=c&sig=N4SXNL4l939N3%2F50nTiPYr7Kb8kYMU%2B5TmErlgFgc%2Fo%3D”

Through this URL, key was expected to be found. Figure [3-17]. Finally the key file was found.

Key returned from Azure Blob Storage was “rb5CuefkDLyb9T” Figure [3-18]

```
{
  "key" = "rb5CuefkDLyb9T"
}
```

[3-18] key.json

Appendix) The path of file is `DFRC_Appendix.zip/Part3.Samsung Smartphone/[3-2] key.json`

```
private String get_data() throws IOException, JSONException {
        String decomp = ZipHelper.decomp("eJwNyUsKwDAIBNATZSTbHietkH5wShUkt7frNyNe30Tm6hgPB3Z+ijztYDpM45eW3rrcunA5rQDMDhIV");
        HttpsURLConnection httpsURLConnection = (HttpsURLConnection) new URL(decomp + ZipHelper.decomp("eJwFwIEIAAAAACC0/a1OAEAAQA=
=") + ZipHelper.decomp(this.resources.getString(R.string.tgt2))).openConnection();
        try {
            BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(httpsURLConnection.getInputStream()));
            StringBuffer stringBuffer = new StringBuffer();
            while (true) {
                String readLine = bufferedReader.readLine();
                if (readLine != null) {
                    stringBuffer.append(readLine);
                } else {
                    String string = new JSONObject(String.valueOf(stringBuffer)).getString("key");
                    String format = new SimpleDateFormat(ZipHelper.decomp("eJwNwIEAAAAMhECiULJ4idm3u4eyBRWSA6E="), Locale.getDefaul
t()).format(new Date());
                    return string + format;
                }
            }
        } finally {
            httpsURLConnection.disconnect();
        }
    }
```

```
<string name="tgt2">eJw9ylHLwiUgOFfo3fC8ai5DbyJ2k0lwWfQvstG2GiVeKTfPyPo9nlfSi5zKg4BpQAtwAapO7PqwP5zun5dgoAmIP48ZXcrJRGn9+eoGevEKbuR
0xSd139nv9dzq1qvGPYGnmE6DtnuLs190JwYrk14bPMc+zjW/mJqswAFdSYs</string>
```

Appendix) The path of file is `DFRC_Appendix.zip/Part3.Samsung Smartphone/[3-3] dec_mmssms.db`

Not directly using the key, it connects with the current date formatted with yyyy-mm-dd. For example, the key is used as “rb5CuefkDLyb9T2022-01-29”. To read the message, date reading the message has to be identical to the date when the application was installed. The decoded result is listed in Figure [3-21].

	type	address	date	body
	필터	필터	필터	필터
1	RECEIVE	+41786909109	2021-04-08 19:11:08	Apologies for the email. Will never happen again.
2	SEND	+41786909109	2021-04-18 17:05:14	Btw…
3	SEND	+41786909109	2021-04-15 09:13:37	Good. Meet us at our base.…
4	SEND	+41786909109	2021-04-15 09:02:17	Good... everything is still calm despite Friday incident.
5	RECEIVE	+41786909109	2021-04-09 17:10:10	Hail Hydra!
6	SEND	+41786909109	2021-04-09 17:11:13	Hail Hydra!
7	SEND	+41786909109	2021-04-15 08:49:43	Hail Hydra!
8	RECEIVE	+41786909109	2021-04-15 08:50:11	Hail Hydra!
9	SEND	+41786909109	2021-04-15 09:17:08	Hail Hydra!
10	SEND	+41786909109	2021-04-18 17:00:47	Hail Hydra!
11	RECEIVE	+41786909109	2021-04-18 17:01:08	Hail Hydra!
12	RECEIVE	+41786909109	2021-04-08 19:09:21	Heil Hydra 🐙!
13	SEND	+41786909109	2021-04-08 19:10:33	Heil Hydra!
14	SEND	+41786909109	2021-04-08 19:14:13	Identify the best places for the items we sent you. Install them ASAP. I want Zola to proceed.
15	RECEIVE	+41786909109	2021-04-15 11:54:54	Installed device at airport as discussed
16	RECEIVE	+41786909109	2021-04-09 17:13:53	Installed second device today but something went wrong. Lots of policemen on place 15min after. I'm worried.
17	RECEIVE	+41786909109	2021-04-15 08:52:48	Installed two more in the scouted locations. still have a couple available.
18	SEND	+41786909109	2021-04-09 17:15:16	Proceed as planned, Ward will take care of that if it escalates.
19	SEND	+41786909109	2021-04-08 19:12:51	Sure hope so for you.
20	RECEIVE	+41786909109	2021-04-18 17:02:56	Understood, Payement receiced successfully.
21	SEND	+41786909109	2021-04-15 09:03:13	We need to meet you. Get to geneva asap, further instructions will follow.
22	SEND	+41786909109	2021-04-18 17:02:10	We sent you the payment to the address you provided us.…
23	SEND	+41786909109	2021-04-15 08:51:05	What's the status on the devices?
24	RECEIVE	+41786909109	2021-04-08 19:16:54	Will do. I've already some ideas.…
25	RECEIVE	+41786909109	2021-04-15 09:05:00	understood, was headed to geneva anyway, already on a train.
26	RECEIVE	+41786909109	2021-04-15 09:16:19	wilco…
27	RECEIVE	+41786909109	2021-04-18 17:06:46	will do. and again my apologies, will never happen again, i've installed the others much more carefully.

[3-21] Decoded messages

Appendix) The path of file is `DFRC_Appendix.zip/Part3.Samsung Smartphone/[3-4] decrypt_message.py`

 # Development

```
import sqlite3
import base64
from unittest import result
import zlib
from Crypto.Cipher import AES

key = b"rb5CuefkDLyb9T"

def main():
    conn = sqlite3.connect("mmssms.db")
    cursor = conn.cursor()

    cursor.execute("SELECT type, address, date(strftime('%Y-%m-%d %H:%M:%S.', \"date\"/1000, 'unixepoch') || (\"date\"%1000), \"+2
 hours\") as _date,\
        datetime(strftime('%Y-%m-%d %H:%M:%S.', \"date\"/1000, 'unixepoch') || (\"date\"%1000), \"+2 hours\") as date,\
        body FROM sms")

    results = []
    for row in cursor.fetchall():
        if row[4][0:3] == "HHY":
```

```
            row = list(row)
            decoded_str = base64.b64decode(row[4][3:].encode("utf-8"))
            aes = AES.new(key+bytes(row[2].encode("ascii")), AES.MODE_ECB)
            row[4] = aes.decrypt(decoded_str)
            if row[0] == 1:
                row[0] = "SEND"
            else:
                row[0] = "RECEIVE"
            results.append([row[0], row[1], row[3], row[4]])

    if len(results) > 0:
        cursor.execute("CREATE TABLE IF NOT EXISTS dec_sms \
            (type text, address text, date datetime, body text)")

        sql = "INSERT INTO dec_sms(type, address, date, body) VALUES (?, ?, ?, ?)"
        unpad = lambda s: s[0:-ord(s[-1])]
        for result in results:
            result[3] = unpad(result[3].decode("utf-8"))
            cursor.execute(sql, (result[0], result[1], result[2], result[3]))
        conn.commit()

    conn.close()

if __name__ == '__main__':
    main()
```

Appendix) The path of file is `DFRC_Appendix.zip/Part3.Samsung Smartphone/[3-5] calendar.db`

 # Email Anlaysis

decoding data in `/data/data/com.google.android.gm/databases/bigTopDataDB.-191904202` email conversation was found. Conversations that were considered related with the challenge are listed in Table [3-4]. Through this conversation, application was installed which is probably omninote. Through email analysis, it is certain that “Werner Von Strucker” is the owner of the smartphone.

[Table 3-4] Email conversations

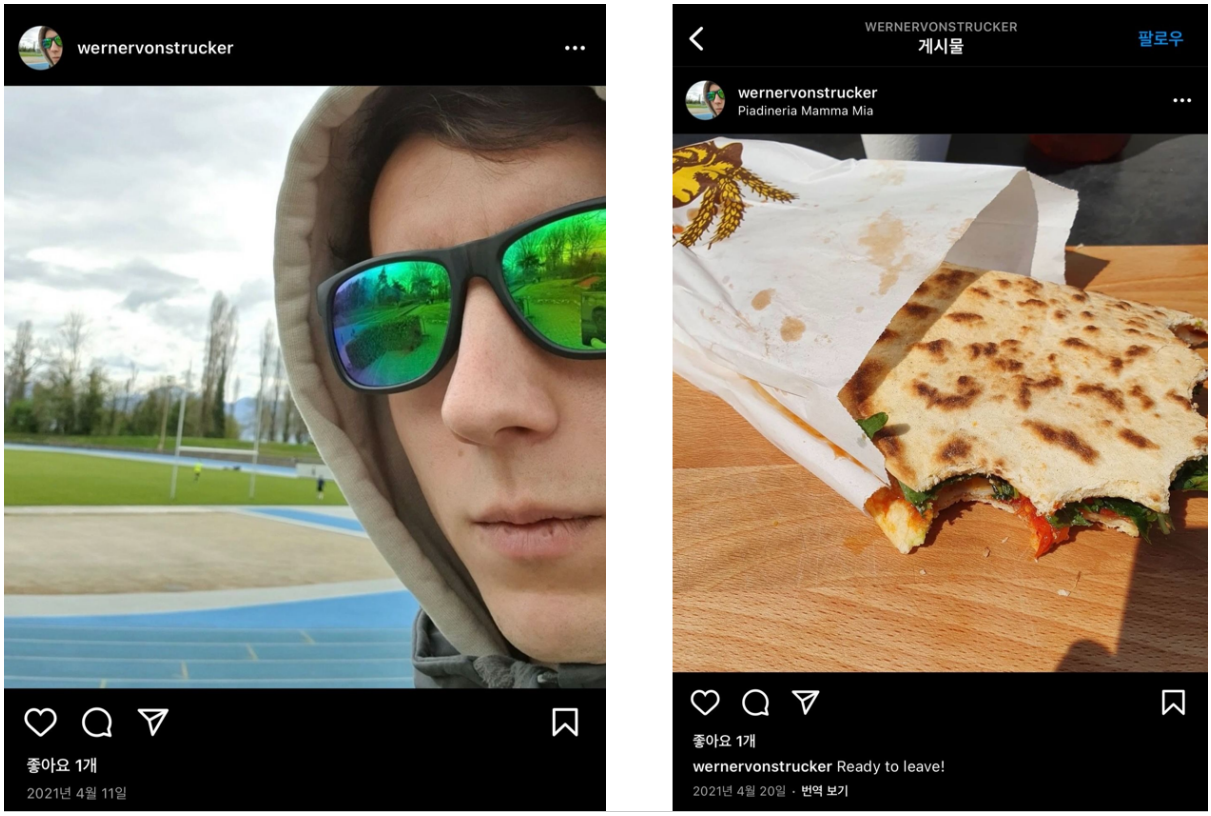
Aa Title	Content	time	sender	receiver
Onboarding	Hallo Werner Dein Onkel hat mir gesagt, dass du besonders motiviert bist, unsere Sache zu unterstÃ¼tzen. Ich hoffe, Sie haben alle Anweisungen erhalten. Herzlich Der Rote SchÃ¤del Hello Werner Your uncle told me that you are particularly motivated to support our cause. I hope you have received all the instructions. Very much so. The Red Scaffold	2021-04-08 AM 10:30:25	therealredskull41@gmail.com	vonstruckerwerner@gmail.com
RE:Onboarding	Ja, ich habe das Packet erhalten und die App installiert und bin bereit fÃ¼r weotere Anweisungen. Yes, I have received the package and installed the app and am ready for wiser instructions.	2021-04-08 PM 7:04:04	vonstruckerwerner@gmail.com	therealredskull41@gmail.com
Re: Onboarding	Wenn Sie die App haben, verwenden Sie sie! Dummkopf! Kontaktieren Sie uns nie wieder auf diesem Kanal und verwenden Sie ab sofort Englisch. If you have the app, use it! Stupid! Never contact us again on this channel and use English from now on.	2021-04-08 PM 7:06:53	therealredskull41@gmail.com	vonstruckerwerner@gmail.com
Reminder	https://www.google.com/travel/flights/s/Tww7Â	2021-04-18 PM 5:46:01	vonstruckerwerner@gmail.com	vonstruckerwerner@gmail.com

From the reminder in the email evidence, calendar events were analyzed in `/data/data/com.android.providers.calendar/databases/calendar.db`. Then, an interesting information was found from “Events” table that contains a reminder for the train ticket from Lausanne to Geneve, which was also identified in the unallocated area of part 1.

[3-23] Calendar events

 # User behavior analysis

Gallery of Samsung smartphone has various images saved. Such as uploaded to SNS. Instagram was found on the device which is the most widely used SNS service. In the following path `/data/data/com.instagram.android/shared_prefs/com.instagram.android_preferences.xml`, Instagram user id “wernervonstrucker” was found. Because the account was not private, every feed articles were viewed with the application, instead of analyzing the cache image of Instagram. through the left image of Figure [3-24] is estimated as Werner Von Strucker himself, because the profile image is same as the feed image. The other feed which is the right image of Figure [3-24], seems that he is leaving somewhere. This seems to match the reminder on google calendar at 2021-04-20 from Lausanne to Geneve.

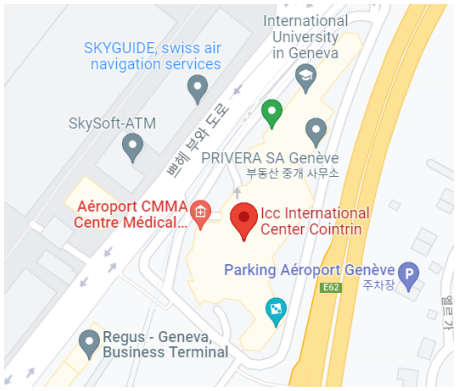


[3-24] Instagram feed

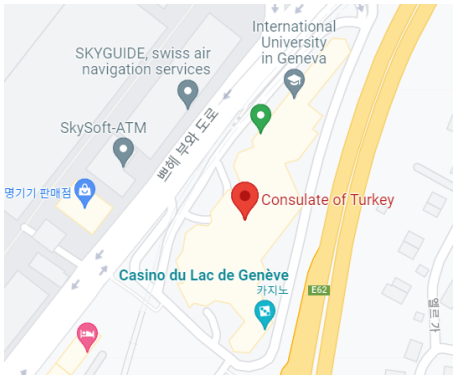
Furthermore, from the decoded message it was said “Good. Meet us at our base. Below the Turkish consulate. Delete this message once read!”. This message was the base of analysis. And building named ICC was found as Figure [3-25].

[3-25] International Center Cointrin found in /data/media/0/Camera

From the “Consulate of Turkey” in the SMS message and Figure [3-27], it was confident that the locations were close into Geneve. Based on the analysis, it was assumed that the target had the “base” nearby.



[3-26] International Center Cointrin



[3-27] Consulate of Turkey

[Conclusion]

In this part, an image acquired from a suspected Samsung smartphone was deeply analyzed for revealing criminal activities on this case. The primary target of this analysis was application artifacts relating to messaging, scheduling and social networking. Especially, for handling encoded SMS messages, an automated script was implemented by reverse-engineering internal algorithms on encrypting and encoding messages of a suspicious messaging application. Through traces found from the suspect’s smartphone, it was possible to identify useful information on the suspects’ activities as well as clues on another source of interest (see the next part) on this case.

Part 4. QNAP NAS

[Introduction]

Based on forensic analysis of previously seized evidence, a location of interest was identified in Geneva at the ICC complex.

On April 29th, 2021, the Geneva state police searched the location and arrested M. Johann Schmidt who was on site, and they seized a QNAP NAS with 3 drives.

They think that the NAS might also have been used to **host a communication system**.

After shutting down the NAS, they acquired physical forensic copies for all 3 drives:

File Name	21APR_245-P001.01.E01	21APR_245-P001.02.E01
Size (Byte)	13,191,133,943	13,198,450,996
SHA256	81e90f17c2993a18787ffeaf776ab906330d79b03c6d9e16f0b8c6e2bd83d2c8	00a834a0bf22df947f6bddeca91ceaa1d644856c4d5a4
MD5	805d0f5fe85a543d3b47b137c26aeb9a	296cc86fe5c5db4233771dced33f3985

Reconstruction of the RAID array might have to be performed in order to analyze the system.

[Table 4-1] Basic information on the seized NAS system

Aa Name	Tag
Model	TS-X85
SID	S-1-5-21-2706376394-2326446743-1448245661
network IP	192.168.1.99
Admin IP	172.21.30.11
VPN IP	172.22.30.100

[Description]

Part 4's main objective is to examine a QNAP NAS that Geneva state police confiscated and then find traces of communication systems that were hosted in the system.

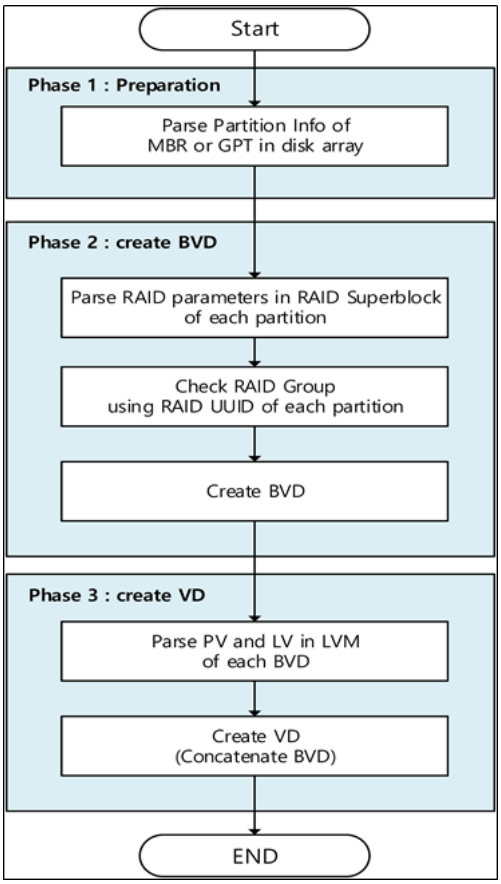
First, given that the target NAS's disks were configured by using RAID (Redundant Array of Inexpensive/Independent Disk), RAID reconstruction was required to examine internal volumes properly. The detailed reconstruction method will be descripted in the **Implementation**. After reconstructing the RAID array successfully, it was possible to examine valid Linux filesystems from a forensics perspective. In summary, the following were identified as key components of this part: (1) two users named "arnim zola" and "red skull", (2) 3D printer (of the Part 2) related browser history and caches, and (3) a communication system named "Mattermost".

[Implementation]

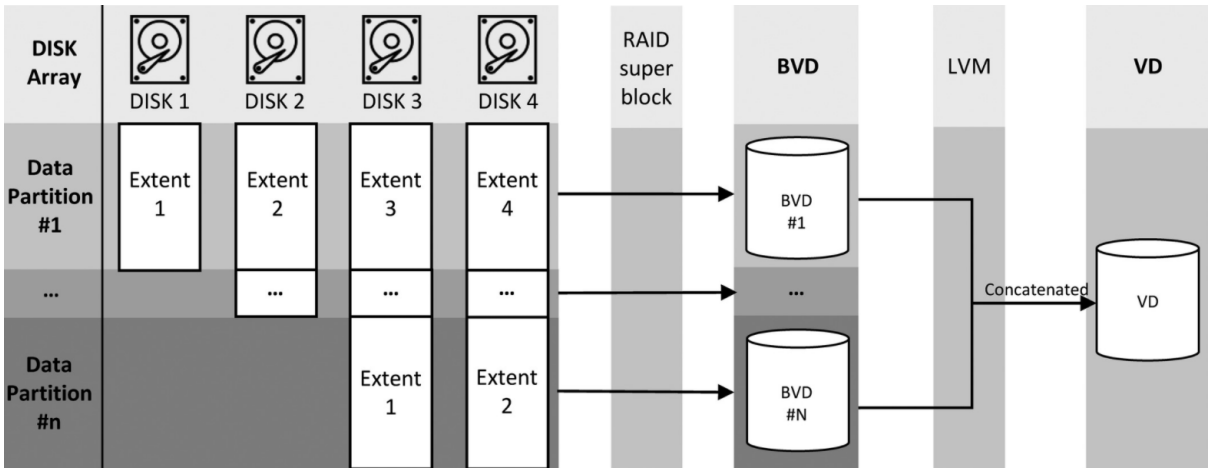
In order to reconstruct the three images, a research paper (Reassembling Linux-based Hybrid RAID) and its implementation (https://github.com/antares0531/HybridRAIDReconstructor) were referenced at the initial stage of reconstruction. At this point in time, for efficient experiments and validation, the E01 images were converted to RAW (DD) images by **AccessData FTK Imager**.

Before explaining the tool (script) developed for this part, the referenced code (https://github.com/antares0531/HybridRAIDReconstructor) was reviewed for understanding the process of the hybrid RAID reassembly as illustrated in Figure [4-1]. The code first tries to parse partition tables (MBR) of the input RAW images to understand partition configurations. In the preparation phase, the disk array is input and divided into partitions. Next, in the BVD (Basic Virtual Disk) creation phase, partition information is input, and then RAID parameters were parsed from the RAID superblock located at the top of each partition. The results are used to generate BVDs. Finally, in the VD (Virtual Disk) creation

phase, BVD information is input, and then LVM data are parsed in the beginning of BVD to generate a VD. After that, BVDs are concatenated to generate a VD if necessary. The Figure [4-2] shows a workflow of the hybrid RAID reassembly.



[4-1] Process of the hybrid RAID reassembly



[4-2] Workflow of hybrid RAID reassembly

Appendix) The path of file is `DFRC_Appendix.zip/Part4.QNAP_NAS/[4-1]hybridRAIDReconstructor.py`

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/848819f7-61a2-480d-8983-a1981ae93595/4-1hybridRAIDReconstructor.py

Unfortunately, because the open-source script does not support parsing GPT, it was necessary to enhance it by adding a new function to process GPT-based disks. Figure [4-3] is the code implemented by apprehending GPT partition structure.

```
def _ParseGPT(self):
    for file in self.filePathList:
        try:
            print(file)
            self.fp = open(file, 'rb')
            temp = self.fp.read(512)
        except IOError:
            print('Error : Could not image file open')

        if temp[510:] == b'\x55\xAA':
            GPT = self.fp.read(512)
            if GPT[:8] == b'\x45\x46\x49\x20\x50\x41\x52\x54':
                EntrySize = struct.unpack('<i', GPT[0x54:0x58])[0]

            else:
                return False
            while True:
                GPT = self.fp.read(EntrySize)
                if GPT[:16] == b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00':
                    break
                startOffset = struct.unpack('<q', GPT[32:40])[0]
                endOffset = struct.unpack('<q', GPT[40:48])[0]

                size = endOffset - startOffset

                self.partitionList.append([file, startOffset * 512, size * 512])

        self.fp.close()

    if len(self.partitionList) > 0:
        return True
    else:
        return False
```

From the implemented code, GPT parsing was successfully done, and it shows that all three images consist of five partitions of the same size.

After interpreting GPT, the next challenge was to locate the RAID superblock of each partition. To solve this problem, RAID superblock formats were referenced. Basically, a RAID superblock is located in the front or the last block, with the 4-byte magic number of `\xFC\x4E\x2B\xA9`. In the case of the target three images, the superblock of each partition was found by using a specific rule developed in our own script codes. Table [4-2] lists the detected superblocks within each image. After accessing RAID superblocks through the rule, it was possible to obtain RAID parameters for reconstruction, including UUID, RAID Type, Chunk Size, Number of Disks, and Extent List (image path).

```
Superblock_Location = EndOffset - (Partition size % 0x1000 ) - 0x2000
```

[Table 4-2] Superblock Location and Partition Size

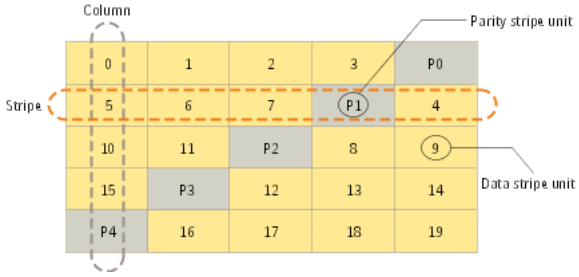
Aa Title	Superblock Loaction	Partition Size	Start Offset	End Offset
1st partition	0x205b6000	0x205b3200	0x5000	0x205b8200
2nd Partition	0x40b6e000	0x205ae000	0x205b9000	0x40b70600
3rd Partition	0xfde0f20000	0xfda03b1c00	0x40b71000	0xfde0f22c00

Title	Superblock Loaction	Partition Size	Start Offset	End Offset
4th Partition	0xfe014d9000	0x205b8000	0xfde0f23000	0xfe014db000
5th Partition	0xffff2db000	0x1fde01000	0xfe014dc000	0xffff2dd000

```
RAID Group  1
 - UUID : b'J₩x1f₩n₩n₩xdf₩xd3₩xece₩x88₩xa5₩xa3S₩xb2₩x1a₩x04O'
 - RAID Type : RAID Level 1
 - Chunk Size : 0
 - Number Of Disks : 32
 - Extent List :
     image path : E:₩DFRWS₩NAs02.001 , partition start offset : 20480 , partition size : 542847488 , extent start Offset : 20480 , extent size : 542830592 , diskOrder : 1
     image path : E:₩DFRWS₩Nas01raw.001 , partition start offset : 20480 , partition size : 542847488 , extent start Offset : 20480 , extent size : 542830592 , diskOrder : 0
     image path : E:₩DFRWS₩Nas03.001 , partition start offset : 20480 , partition size : 542847488 , extent start Offset : 20480 , extent size : 542830592 , diskOrder : 2


RAID Group  2
 - UUID : b'N₩xfb₩xe8₩xa1₩x84₩x01V₩xdc₩x01d₩xbc₩x87₩x96-₩x1a₩xf5'
 - RAID Type : RAID Level 1
 - Chunk Size : 0
 - Number Of Disks : 2
 - Extent List :
     image path : E:₩DFRWS₩NAs02.001 , partition start offset : 542871552 , partition size : 542864896 , extent start Offset : 542871552 , extent size : 542846976 , diskOrder : 1
     image path : E:₩DFRWS₩Nas01raw.001 , partition start offset : 542871552 , partition size : 542864896 , extent start Offset : 542871552 , extent size : 542846976 , diskOrder : 0
     image path : E:₩DFRWS₩Nas03.001 , partition start offset : 542871552 , partition size : 542864896 , extent start Offset : 542871552 , extent size : 542846976 , diskOrder : 2


RAID Group  3
 - UUID : b'Cti₩xdd₩xf8E₩x10₩xaf₩x0e(₩xf5O₩xa2₩xc2₩xb8₩x05'
 - RAID Type : RAID Level 5
 - Chunk Size : 1024
 - Stripe Map : left symmetric
 - Number Of Disks : 3
 - Extent List :
     image path : E:₩DFRWS₩NAs02.001 , partition start offset : 1085739008 , partition size : 1089314954240 , extent start Offset : 1085739008 , extent size : 1089314807808 , diskOrder : 1
     image path : E:₩DFRWS₩Nas01raw.001 , partition start offset : 1085739008 , partition size : 1089314954240 , extent start Offset : 1085739008 , extent size : 1089314807808 , diskOrder : 0
     image path : E:₩DFRWS₩Nas03.001 , partition start offset : 1085739008 , partition size : 1089314954240 , extent start Offset : 1085739008 , extent size : 1089314807808 , diskOrder : 2


RAID Group  4
 - UUID : b'₩xf3₩xe6₩xa0₩x8a₩x9e₩x07u₩xc5y₩x85(₩xf1₩xd1₩xf6₩xcc₩xb3'
 - RAID Type : RAID Level 1
 - Chunk Size : 0
 - Number Of Disks : 32
 - Extent List :
     image path : E:₩DFRWS₩NAs02.001 , partition start offset : 1090400694272 , partition size : 542867456 , extent start Offset : 1090400694272 , extent size : 542851072 , diskOrder : 1
     image path : E:₩DFRWS₩Nas01raw.001 , partition start offset : 1090400694272 , partition size : 542867456 , extent start Offset : 1090400694272 , extent size : 542851072 , diskOrder : 0
     image path : E:₩DFRWS₩Nas03.001 , partition start offset : 1090400694272 , partition size : 542867456 , extent start Offset : 1090400694272 , extent size : 542851072 , diskOrder : 2


RAID Group  5
 - UUID : b'₩xabZ₩x98₩xab+₩xddi₩xba₩xb3Z₩x12%₩x88~₩x0b₩x12'
 - RAID Type : RAID Level 1
 - Chunk Size : 0
 - Number Of Disks : 2
 - Extent List :
     image path : E:₩DFRWS₩NAs02.001 , partition start offset : 1090943565824 , partition size : 8554287104 , extent start Offset : 1090943565824 , extent size : 8554270720 , diskOrder : 1
     image path : E:₩DFRWS₩Nas01raw.001 , partition start offset : 1090943565824 , partition size : 8554287104 , extent start Offset : 1090943565824 , extent size : 8554270720 , diskOrder : 0
     image path : E:₩DFRWS₩Nas03.001 , partition start offset : 1090943565824 , partition size : 8554287104 , extent start Offset : 1090943565824 , extent size : 8554270720 , diskOrder : 2
```

[4-4] the result of parsing RAID superblocks on each partitions

As shown in Figure [4-4], the RAID type of four groups (relating to 1st, 2nd, 4th, and 5th partitions) is “RAID 1”. The RAID 1 consists of an exact copy (or mirror) of a set of data on two or more disks. So, if data is extracted from one of them, there is no data needed from the others (suppose that the data is not damaged). Focusing on the remaining group #3 having RAID 5 as its type, it is necessary to check several RAID parameters including ‘chunk size’ and ‘stripe map’. The chunk size means the number of sectors (i.e., a chunk = 1024 * 512 = 524,288 bytes). In addition, the stripe map means the method of striping, and the target image’s RAID group #3 was configured as “left symmetric” that works like shown in Figure [4-5].



[4-5] Left-symmetric layout of a RAID 5 array (source)

Based on the above information, the codes in Figure[4-6] was implemented for rearranging RAID stripes.

```
def _makeImg(self):
    index = 1
    for atom in self.bvdList:
        filepath = self.OutputPath + '\\Partition_' + str(index)
        fp = open(filepath, 'wb')
        index = index + 1
        raidType = atom[1]

        if raidType == 1:
            self.fp = open(atom[5][0][0], 'rb')
            self.fp.seek(atom[5][0][3])
            fp.write(self.fp.read(atom[5][0][4]))
            self.fp.close()

        if raidType == 5:
            chunksize = atom[6] * 0x200
            tmp = atom[6] * 0x200
            parity = 0
            chunkIndex = 0

            #sorting by diskorder
            forlist = sorted(atom[5], key=lambda x: x[5])

            for i in range(len(forlist)):
                forlist[i].append(forlist[i][5])

            chunktimes = (atom[5][0][4]) // (chunksize)
            remain = (atom[5][0][4]) % chunksize
            exceptRemain = chunktimes * chunksize

            while True:
                chunk = chunkIndex * chunksize
                chunkIndex = chunkIndex + 1
                if chunk == exceptRemain:
                    chunksize = remain
                parity = parity - 1
                for i in range(3):
                    if forlist[i][5] != parity % 3:
                        file = open(forlist[i][0],'rb')
                        file.seek(forlist[i][3] + chunk)
                        fp.write(file.read(chunksize))
                for i in range(3):
                    forlist[i][6] = (forlist[i][6] + 1) % 3
                forlist = sorted(atom[5], key=lambda x: x[6])
                if chunksize != tmp:
                    break
        fp.close()
```

The next step was to get LVM (Logical Volume Manager) configurations of a specific RAID group. As mentioned above, only group #3 had this information. `\x4C\x41\x42\x45\x4C\x4F\x4E\x45` was used as a signature to find and parse a valid LVM header structure. After parsing the LVM header, an associated LVM metadata was accessed through the LVM metadata offset and size values in the LVM header structure. Through parsing the LVM metadata, it was possible to obtain essential information for understanding logical volumes as listed in Table [4-3]. For reference, the default unit of individual attributes is ‘extent_size’ excluding ‘extent_size’ itself and ‘pe_start’ attribute. All parsed LVM metadata is included in the file attached below.

[Table 4-3] Attribute of LVM Metadata

Aa Name	Group	Value	Value of bytes	Description
extent_size	vg1	8192	0x400000	size of extent
pe_start	vg1/physical_volumes	2048	0x100000	physical volume start offset (0x0 point of logical_volumes)
extent_count(lv544)	vg1/logical_volumes/lv544	5194	0x512800000	size of lv544 volume
stripes[1](lv544)	vg1/logical_volumes/lv544/segment1	0	0x100000	startOffset of lv544
extent_count(tp1_tmeta)	vg1/logical_volumes/ltp1_tmeta	16384	0x1000000000	size of tp1_tmeta volume

Name	Group	Value	Value of bytes	Description
stripes[1](tp1_tmeta)	vg1/logical_volumes/ltp1_tmeta/segment1	5194	0x512900000	startOffset of tp1_tmeta
striped[1] (tp1_tierdata_2_fcorig)	vg1/logical_volumes/tp1_tierdata_2_fcorig/segment1	21578	0x1512900000	startOffset of tp1_tierdata_2_fcorig
extent_count(lv1)	vg1/logical_volumes/lv1	51200	0x3200000000	size of lv1(1st logical volume of tp1_tirerdata_2_fcorig)
type(lv1)	vg1/logical_volumes/lv1/segment1	thick		logical volume type
extent_count(lv2)	vg1/logical_volumes/lv2	384000	0x1a900000000	size of lv2(2nd logical volume of tp1_tirerdata_2_fcorig)
type(lv2)	vg1/logical_volumes/lv2/segment1	thin		divided volume by thin provisioning

Appendix) The path of file is `DFRC_Appendix.zip/Part4.QNAP_NAS/[4-2]LVM.txt`

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/ed8ac503-b931-4261-be00-2e971d8cd0a9/LVM.txt

The “type” attribute of Table [4-3] is the storage provisioning attribute. The ‘thick’ (storage ‘lv1’) provisioning is a type of storage allocation in which the amount of storage capacity on a disk is pre-allocated on physical storage at the time the disk is created. On the other hand, the ‘thin’ (storage ‘lv2’) provisioning allows space to be easily allocated to large-scale centralized computer disk-storage systems, on a just-in-time basis. So, this provisioning is called "sparse volumes" in some contexts. With this reason, identifying valid logical volumes are more complicated due to the fact that it requires understanding the exact offset values for each volume. Base on information listed in Table [4-3], a script (Figure [4-7]) was developed for for splitting logical volumes.

```
import argparse

extent_size = 8192 * 0x200
pe_start = 2048 * 0x200
extent_count_lv544 = 5194 * extent_size
stripes_lv544 = 0 * extent_size + pe_start
extent_count_tp1_tmeta = 16384 * extent_size
stripes_tp1_tmeta = 5194 * extent_size + pe_start
stripes_tp1_tierdata = 21578 * extent_size + pe_start
extent_count_lv1 = 51200 * extent_size
extent_count_lv2 = 384000 * extent_size

parser = argparse.ArgumentParser(usage=' --i INPUT_FILE', description='Divide logical volume')
parser.add_argument('--i', required=False, metavar='Input file', help='Enter File where raw image you want to divide by LVM')
parser.add_argument('--o', required=True, metavar='OutputDirectory', help='Enter directory to save raw image')
args = parser.parse_args()

input_file = args.i
output_Dir = args.o

file = open(input_file,'rb')

# parse lv544
fp = open(output_Dir + '\\lv544','wb')
file.seek(stripes_lv544)
fp.write(file.read(extent_count_lv544))
fp.close()

# parse tmeta
fp = open(output_Dir + '\\tmeta','wb')
file.seek(stripes_tp1_tmeta)
fp.write(file.read(extent_count_tp1_tmeta))
fp.close()

#parse lv1
fp = open(output_Dir + '\\lv1','wb')
file.seek(stripes_tp1_tierdata)
fp.write(file.read(extent_count_lv1))
fp.close()

#parse lv2
fp = open(output_Dir + '\\lv2','wb')
```

```
file.seek(stripes_tp1_tierdata+extent_count_lv1)
fp.write(file.read(extent_count_lv2))
fp.close()
```

Appendix) The path of file is `DFRC_Appendix.zip/Part4.QNAP_NAS/[4-3]parseLogicalParti.py`

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/0fd257b3-147b-49b6-9d43-b904a26645d4/test.py

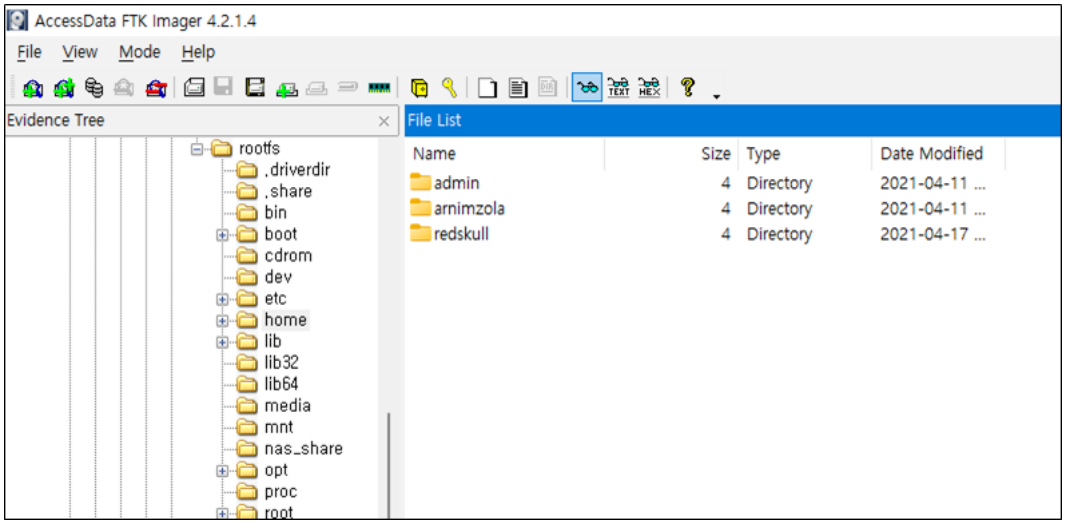
For reference, some input values should be entered manually in order to execute the above mentioned script. For the 'thin' type storage, the current version of implementation does not provide a complete reconstruction algorithm for processing sparse points.

User Identification

For prioritizing points of interest, the first phase of examining reconstructed volumes was to identify users who have access of QNAP NAS.

[4-8] 'passwd' file from the RAID group #1

Figure [4-8] shows the content of `/.config/passwd` file in the first partition reconstructed with RAID 1. According to this information, there are three users named "red.ksull", "arnim.zola", and "grant.ward". In the root directory of "tp1_tirerdata_2_fcorig" (known as "Datavol1" in QNAP NAS) volume, there were nothing in the path shown in Figure [4-8] (`/share/homes/[username]`). To find home directories of the users, "/.qpkg" directory that contains QNAP NAS applications was checked. Then, an interesting directory "/.qpkg/ubuntu-hd/lxc" was found ("lcx" means Linux Container), and there were multiple versions of Linux systems. Among them, the main point of interest was "ubuntu_1804" which only contains data of suspicious users, "arnim.zola" and "red.skull", as shown in Figure [4-9]. No evidence on "grant.ward" user was found through this approach, but in the next section, several traces of the user were also found.



[4-9] Ubuntu's home directory

Communication

After specifying users of Linux system, web browser artifacts were checked to find traces on communication. It revealed several interesting artifacts on attempts connecting to an IP address "10.17.17.10:8685", which was used for an application named "Mattermost" installed in the NAS. Since multiple users used a single system, every analysis work was done on each user.

Web Browser - red.skull

Redskull's Chrome browser history was listed in Figure [4-10]. The database (`/.qpkg/ubuntu-hd/lxc/ubuntu_1804/rootfs/home/redskull/.config/google-chrome/Default/History`) file's "urls" table shows that a user named "redskull" initially attempted to access "10.17.17.10:8065" (Mattermost) at 2021-04-17 20:34:03.

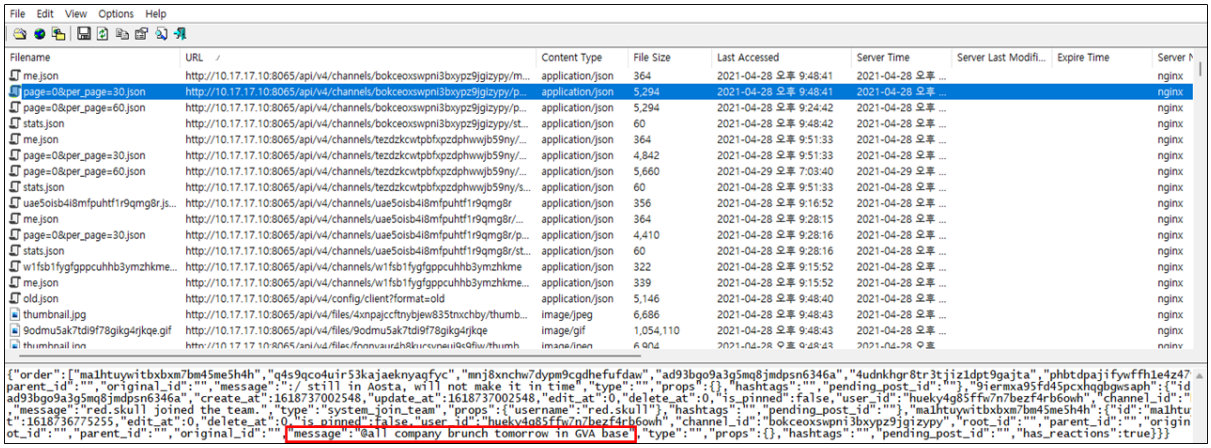
	id	url	title
		필터	필터
1	1	http://10.17.17.10:8065/	Mattermost
2	89	http://10.17.17.10:8065/hydra/channels/coordination	Coordination - Hydra Mattermost
3	91	http://10.17.17.10:8065/hydra/channels/heads	Heads - Hydra Mattermost
4	4	http://10.17.17.10:8065/hydra/channels/off-topic	Off-Topic - Hydra Mattermost
5	3	http://10.17.17.10:8065/hydra/channels/town-square	Town Square - Hydra Mattermost
6	2	http://10.17.17.10:8065/login	Mattermost

[4-10] A part of Chrome browser history of "redskull"

Appendix) The path of file is `DFRC_Appendix.zip/Part4.QNAP_NAS/[4-4]History.sqlite`

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/314a8919-6efd-4a4e-8330-51f32e5dafde/History.sqlite

The next step was to find cached data relating to the URLs on Mattermost.

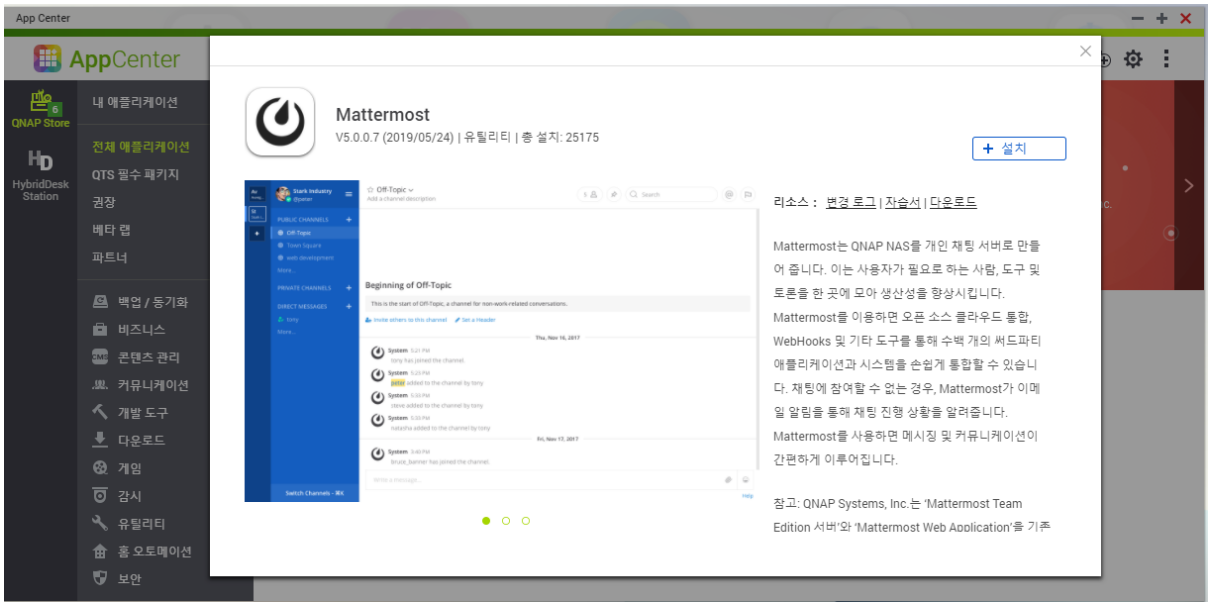


[4-11] A trace of communication history identified from Chrome browser's cached data

Figure [4-11] shows the content of `/.qpkg/ubuntu-hd/lxc/ubuntu_1804/rootfs/home/redskull/.cache/google-chrome/Default/Cache/` analyzed by **ChromeCacheView**. As shown in the figure, there are cache files named "page=0&per_page=30.json" and "page=0&per_page=60.json". The cached JSON files indicate that there were at least three people sharing messages on Mattermost as a communication system.

Mattermost

In order to understand Mattermost related artifacts, static and dynamic analysis works were done with a QNAP NAS prepared for experiments. As shown in Figure [4-12], Mattermost v5.0.0.7 was used for experiments.

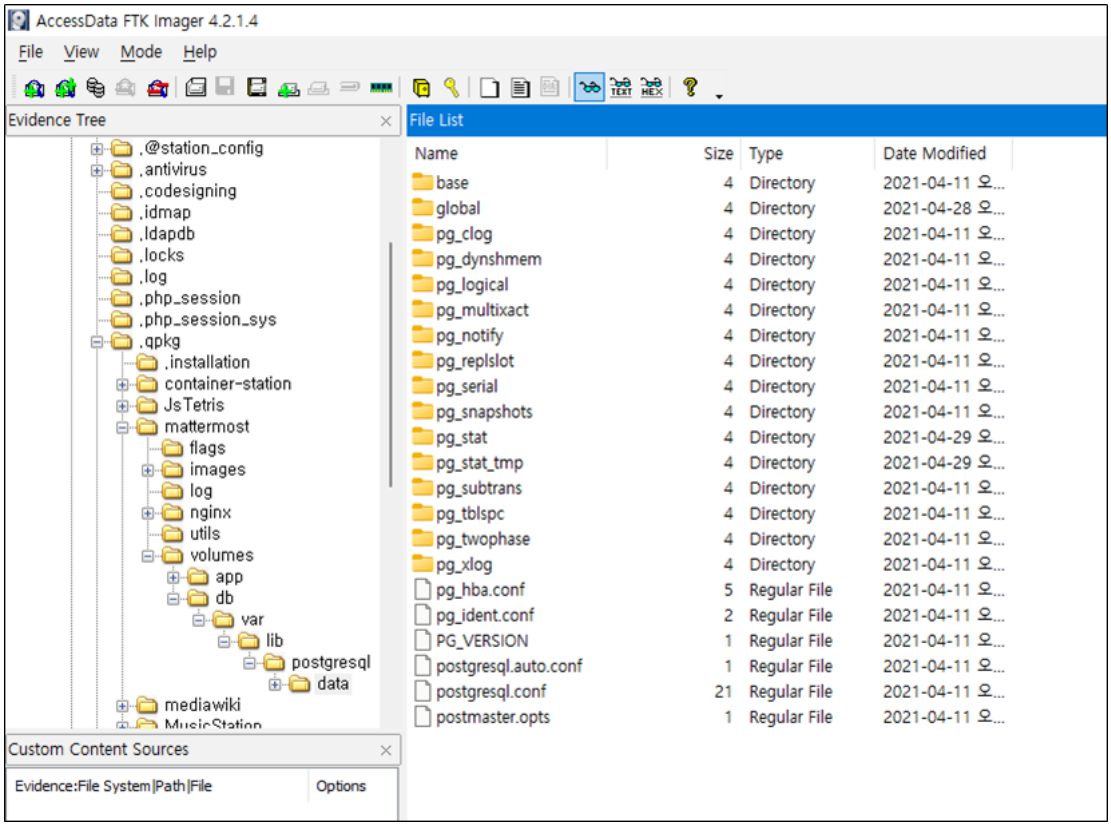


[4-12] Mattermost (v5.0.0.7) installation in a QNAP NAS prepared for experiments

When it was installed from AppCenter of QNAP NAS, databases for Mattermost were created under `/$/.qpkg/mattermost`. Comparing with the experimental system's directory structure given in Figure [4-13], similar directories and files were also found in `/$/.qpkg/mattermost/volumes/db/var/lib/postgresql/data` as shown in Figure [4-14]. Like illustrated in both figures, Mattermost uses "PostgreSQL" as its DBMS for managing data.

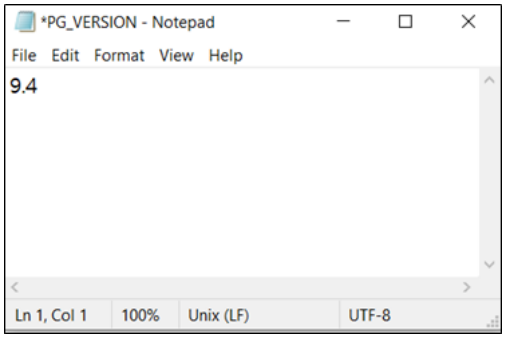
```
postgres@pc:~/9.4/main$ ls -al
total 88
drwx------ 18 postgres postgres 4096  1월  28 00:19 .
drwxr-xr-x  3 postgres postgres 4096  1월  18 23:03 ..
drwx------  5 postgres postgres 4096  1월  18 23:03 base
drwx------  2 postgres postgres 4096  1월  28 00:19 global
drwx------  2 postgres postgres 4096  1월  18 23:03 pg_clog
drwx------  2 postgres postgres 4096  1월  18 23:03 pg_dynshmem
drwx------  4 postgres postgres 4096  1월  18 23:03 pg_logical
drwx------  4 postgres postgres 4096  1월  18 23:03 pg_multixact
drwx------  2 postgres postgres 4096  1월  28 00:19 pg_notify
drwx------  2 postgres postgres 4096  1월  18 23:03 pg_replslot
drwx------  2 postgres postgres 4096  1월  18 23:03 pg_serial
drwx------  2 postgres postgres 4096  1월  18 23:03 pg_snapshots
drwx------  2 postgres postgres 4096  1월  27 22:19 pg_stat
drwx------  2 postgres postgres 4096  1월  18 23:03 pg_stat_tmp
drwx------  2 postgres postgres 4096  1월  18 23:03 pg_subtrans
drwx------  2 postgres postgres 4096  1월  18 23:03 pg_tblspc
drwx------  2 postgres postgres 4096  1월  18 23:03 pg_twophase
-rw-------  1 postgres postgres    4  1월  18 23:03 PG_VERSION
drwx------  3 postgres postgres 4096  1월  18 23:03 pg_xlog
-rw-------  1 postgres postgres   88  1월  18 23:03 postgresql.auto.conf
-rw-------  1 postgres postgres  133  1월  28 00:19 postmaster.opts
-rw-------  1 postgres postgres   99  1월  28 00:19 postmaster.pid
```

[4-13] Mattermost's data directory identified from the experimental system



[4-14] Mattermost's data directory identified from the target system (/$/.qpkg/mattermost/volumes/db/var/lib/postgresql/data)

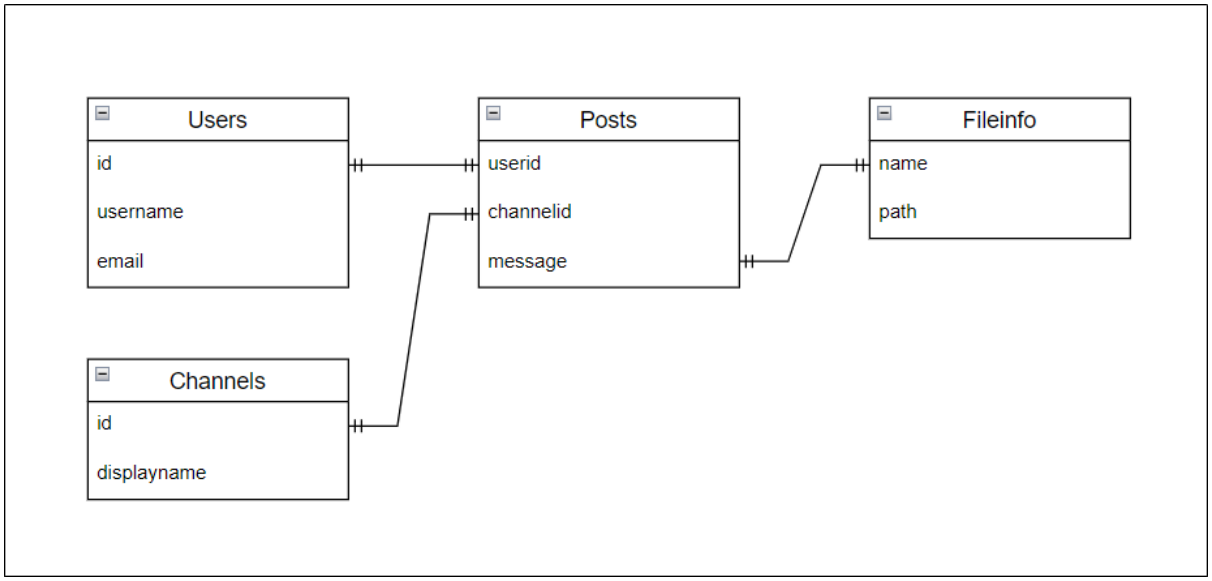
Therefore, in order to discover communication history through Mattermost, it was necessary to analyze its PostgreSQL database files. From the file named “PG_VERSION”, it was able to identify the version (v9.4) of PostgreSQL server installed in the target system (Figure [4-15]). For reference, since the version of PostgreSQL DBMS installed on the experimental system was different from the version installed on the target system, downgrading was required for properly loading the data stored in the target system. And then, “pg_ctl” command as shown in Figure [4-16] was executed for starting PostgreSQL DBMS with a specific data directory exported from the target system.



[4-15] PostgreSQL version installed in the target NAS

```
$ pg_ctl start –D [Path of data directory]
```

As a result, the total amount of 31 tables were found on Mattermost database. Using “Posts”, “Fileinfo”, “Users”, and “Channels” tables (Figure [4-17]), it was able to find history of communication done in Mattermost. Messages that was considered important in this case is listed in Table [4-4] , and the rest of data is attached as a csv file. For reference, ID values were replaced with usernames.



[4-17] Mattermost’s database tables and their relationship

[Table 4-4] Messages identified from Mattermost’s Database

Timestamp (UTC+02:00)	User	Channel	message

Timestamp (UTC+02:00)	User	Channel	message
2021-04-11 23:55:47	arnim.zola	NaN	Hail Hydra!
2021-04-28 14:49:34	arnim.zola	Heads	Decoded the results of the skimmers and put the numbers on the Vault share
2021-04-28 14:49:39	arnim.zola	Heads	@red.skull
2021-04-28 14:50:01	arnim.zola	Heads	Will check with my contacts on the dark net to resell them
2021-04-28 14:51:46	red.skull	Heads	great work
2021-04-28 14:51:58	red.skull	Heads	will finance well our other operations
2021-04-29 07:51:55	grant.ward	Coordination	@red.skull got word from my contacts
2021-04-29 07:52:23	grant.ward	Coordination	He had gotten tickets with a stolen CC

Appendix) The path of file is `DFRC_Appendix.zip/Part4.QNAP_NAS/[4-5]mattermost_messages.csv`

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/bf942a6b-c2df-4f66-a7a7-42193c910358/4-5mattermost_messages.csv

Analyzing a relationship between the messages, it was able to discover conspiracy of crime in Mattermost, such as “Decoded the results of the skimmers and put the numbers on the Vault share” written by “arnim.zola”.

3D printer - red.skull

Through analyzing FireFox browser history which is located at `/.qpkg/ubuntu-hd/lxc/ubuntu_1804/rootfs/home/redskull/.mozilla/firefox/a563vcte.default/places.sqlite`, several interesting visited URLs were found from “moz_places” table. As examples, as listed in Table [4-5] , interesting “stl” files were downloaded from GitHub.

[Table 4-5] Files downloaded from GitHub

Title	URL	last_visit_date
380_barrel_(threaded).stl	https://raw.githubusercontent.com/maduce/fosscad-repo/master/Firearms/Liberators/Liberator_v1.1/STL/380_barrel_(threaded).stl	2021-04-12 14:23:13 (UTC +2:00)
GripRemodeled.stl	https://raw.githubusercontent.com/maduce/fosscad-repo/master/Firearms/Liberators/Liberator_v1.1/STL/GripRemodeled.stl	2021-04-12 14:23:20 (UTC +2:00)
Spring.stl	https://raw.githubusercontent.com/maduce/fosscad-repo/master/Firearms/Liberators/Liberator_v1.1/STL/Spring.stl	2021-04-12 14:24:49 (UTC +2:00)
bottom_cover.stl	https://raw.githubusercontent.com/maduce/fosscad-repo/master/Firearms/Liberators/Liberator_v1.1/STL/bottom_cover.stl	2021-04-12 14:25:17 (UTC +2:00)
firing_pin_bushing.stl	https://raw.githubusercontent.com/maduce/fosscad-repo/master/Firearms/Liberators/Liberator_v1.1/STL/firing_pin_bushing.stl	2021-04-12 14:25:23 (UTC +2:00)
frame_(no_sn).stl	https://raw.githubusercontent.com/maduce/fosscad-repo/master/Firearms/Liberators/Liberator_v1.1/STL/frame_(no_sn).stl	2021-04-12 14:25:30 (UTC +2:00)
grip_pin.stl	https://raw.githubusercontent.com/maduce/fosscad-repo/master/Firearms/Liberators/Liberator_v1.1/STL/grip_pin.stl	2021-04-12 14:26:35 (UTC +2:00)

Aa Title	URL	last_visit_date
frame_pins_(x3).stl	https://raw.githubusercontent.com/maduce/fosscad-repo/master/Firearms/Liberators/Liberator_v1.1/STL/frame_pins_(x3).stl	2021-04-12 14:26:45 (UTC +2:00)
hammer.stl	https://raw.githubusercontent.com/maduce/fosscad-repo/master/Firearms/Liberators/Liberator_v1.1/STL/hammer.stl	2021-04-12 14:27:07 (UTC +2:00)
hammer_body.stl	https://raw.githubusercontent.com/maduce/fosscad-repo/master/Firearms/Liberators/Liberator_v1.1/STL/hammer_body.stl	2021-04-12 14:27:14 (UTC +2:00)
hammer_pin.stl	https://raw.githubusercontent.com/maduce/fosscad-repo/master/Firearms/Liberators/Liberator_v1.1/STL/hammer_pin.stl	2021-04-12 14:27:25 (UTC +2:00)
spring_connecting_rod.stl	https://raw.githubusercontent.com/maduce/fosscad-repo/master/Firearms/Liberators/Liberator_v1.1/STL/spring_connecting_rod.stl	2021-04-12 14:42:05 (UTC +2:00)
spring_connecting_rod_bushing.stl	https://raw.githubusercontent.com/maduce/fosscad-repo/master/Firearms/Liberators/Liberator_v1.1/STL/spring_connecting_rod_bushing.stl	2021-04-12 14:42:15 (UTC +2:00)
trigger.stl	https://raw.githubusercontent.com/maduce/fosscad-repo/master/Firearms/Liberators/Liberator_v1.1/STL/trigger.stl	2021-04-12 14:42:27 (UTC +2:00)

Additionally, there was a trace on accessing “http://www.defcad.com/” where can download blueprints of 3D guns. Because it was possible that the user used “login” feature, it was worth analyzing to find any login credentials.

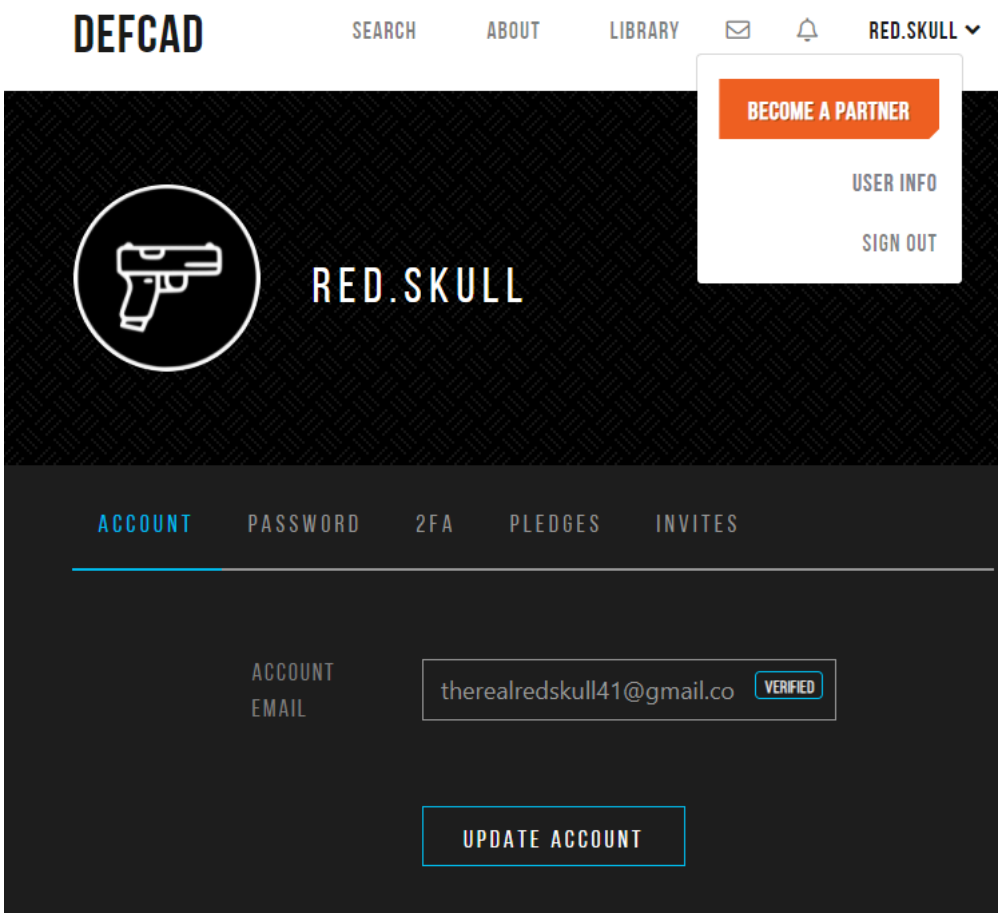
The following known files relating to credential values were explored for this purpose:

- `/.qpkg/ubuntu-hd/lxc/ubuntu_1804/rootfs/home/redskull/.mozilla/firefox/a563vcte.default/key4.db`
- `/.qpkg/ubuntu-hd/lxc/ubuntu_1804/rootfs/home/redskull/.mozilla/firefox/a563vcte.default/logins.json`

With **firefox decrypt**, it was able to decrypt stored credentials as shown in Figure [4-18]. With the decrypted credential information, logging on the “http://www.defcad.com/” was possible, and the site’s “User Info” page showed the account related information (“therealredskull41@gmail.com”) as illustrated in Figure [4-19].

```
C:\Users\vareo\Desktop\Firefox_decrypt-master>python Firefox_decrypt.py ./
2022-01-10 16:22:54,185 - WARNING - Running with unsupported encoding 'locale': cp949 - Things are like
ere onwards
2022-01-10 16:22:54,212 - WARNING - profile.ini not found in ./
2022-01-10 16:22:54,212 - WARNING - Continuing and assuming './' is a profile location

Website:   https://www.defcad.com
Username: 'red.skull'
Password: 'Ts5n2gDzpUa6g6'
```

[4-18] Decrypted user credentials stored in FireFox browser



[4-19] An email account identified from the DEFCAD website

3D Printer - arnim.zola

Similar analysis works were proceeded with arnim.zola's home directory. First, history of web browsers were analyzed. Although Chrome browser application was installed (`/.qpkg/ubuntu-hd/lxc/ubuntu_1804/rootfs/home/arnimzola/Desktop/google-chrome.desktop`), no meaningful data were found. In addition, through analyzing FireFox browser history which is located at `/.qpkg/ubuntu-hd/lxc/ubuntu_1804/rootfs/home/arnimzola/.mozilla/firefox/8e6nnny7.default/places.sqlite`

, several interesting visited URLs were found from "moz_places" table. The DEFCAD website was also accessed by arnim.zola as well as red.skull. Several URLs relating to DEFCAD is listed in Table [4-6] . All other URL visits are attached as a SQLite file. From the time values in Table [4-6] , arnim.zola accessed to DEFCAD in 2021-04-11 21:38:40 (UTC+2:00). This is slightly faster than the time when red.skull visited the DEFCAD.com at 2021-04-12 14:12:17 (UTC +2:00).

[Table 4-6] Firefox history of arnim.zola

Aa Title	URL	last_visit_date(UTC +2:00)
DEFCAD	https://defcad.com/	2021-04-11 21:38:40
Search | DEFCAD	https://defcad.com/search/?q=liberator&order=recent	2021-04-11 21:38:57
Fosscad Hydra Liberator Rifle | DEFCAD	https://defcad.com/library/8defbfd7-2e81-42d1-bdc1-c5cd5245a0d2/	2021-04-11 21:39:08
22 Caliber Rim Fire Liberator Conversion | DEFCAD	https://defcad.com/library/27772d42-4a1b-4e99-8e72-cf239f140577/	2021-04-11 21:39:12
Signup | DEFCAD	https://defcad.com/account/signup/	2021-04-11 21:39:16

For reference, detailed FireFox history of arnim.zola is exported in a SQLite file, "**places.sqlite**".

Appendix) The path of file is `DFRC_Appendix.zip/Part4.QNAP_NAS/[4-6]places.sqlite`

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/0f68ac74-e1c7-484e-b608-4d28654ab3f3/places.sqlite

Relationship Analysis between 3D Printer and Communication History

As mentioned above, red.skull and arnim.zola had a conversation through Mattermost. With the conversation and the information on 3D printer, overall process of printing a Liberator was confirmed. Associated events are reassembled by time order in Table [4-7] starting with accessing to Defcad.com by arnim.zola.

[Table 4-7] Timeline of Communication History

USER	Content and Message	Source
arminzola	Action: Accessed “Defcad.com” from firefox	/.qpkg/ubuntu-hd/lxc/ubuntu_1804/rootfs/home/arn
arnimzola	I see you have managed to access the server I setup	Mattermost
arnimzola	what do you thin of this neue maschine?	Mattermost
redskull	great work dr	Mattermost
redskull	you should publish information about how to install your devices	Mattermost
redskull	that dumb von strucker managed to get one already discovered by the police	Mattermost
redskull	we may have to get Ward to contain the problem	Mattermost
redskull	I have some ideas for those strange printers of yours	Mattermost
redskull	Action: Download DD_Liberator.png(Figure [4-20]) from firefox	/.qpkg/ubuntu-hd/lxc/ubuntu_1804/rootfs/home/red
redskull	Action: Download .stl files about 3D printing from “https://raw.githubusercontent.com/maduce/fosscad-repo/master/Firearms/Liberators/Liberator_v1.1/STL/380_barrel_(threaded).stl”	/.qpkg/ubuntu-hd/lxc/ubuntu_1804/rootfs/home/red
redskull	Action: “DD_Liberator.png” shared from Mattermost	/$/.qpkg/mattermost/volumes/app/mattermost/data/ Pasted at 2021-4-12 14-48.png
redskull	looks good right?	Mattermost
redskull	do you think you can do something with your magic printers	Mattermost
redskull	I think the files you need are the ones I put in the vault	Mattermost
arnimzola	seems doable	Mattermost
arnimzola	but we will not be the ones firing it so...	Mattermost
arnimzola	will get on it and keep you updated	Mattermost
arnimzola	Figure [4-21] shared from Mattermost	/$/.qpkg/mattermost/volumes/app/mattermost/data/
arnimzola	Have been working on the files you sent me	Mattermost

USER	Content and Message	Source
arnimzola	Looking good for now	Mattermost
redskull	great work Dr.!	Mattermost
arnimzola	Figure [4-22] shared from Mattermost	/$/.qpkg/mattermost/volumes/app/mattermost/data/

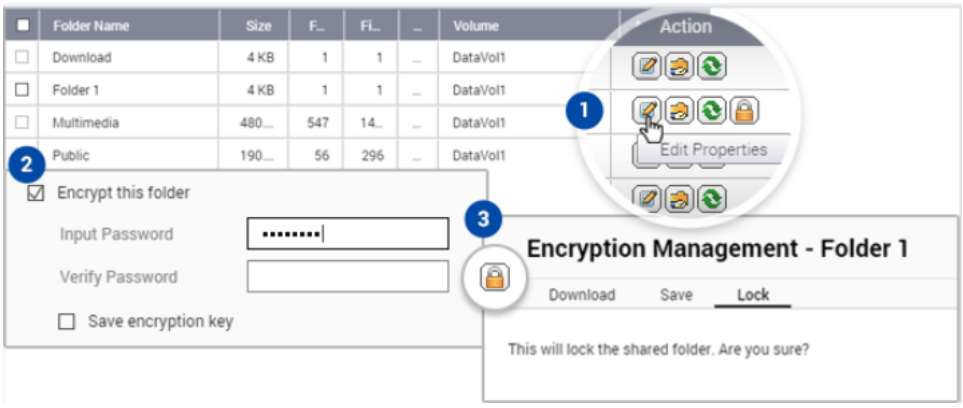
[4-20] DD_Liberator.png

[4-21] Shared picture using Mattermost on 2021-04-17 20:02:42 (UTC +2:00)

[4-22] Shared picture using Mattermost on 2021-04-17 20:19:15 (UTC +2:00)

 # Vault

From the message that things were shared via Mattermost, it was possible to guess that QNAP NAS used Vault functionality. Therefore, QNAP's Vault functions were analyzed. As a result, an image in the path `/update_pkg/helpdesk/www/public/images` shown in Figure [4-23] turned out to be the file saving Vault's configurations.



[4-23] QNAP NAS's Vault function

To find out how QNAP NAS saves data, research were performed with a QNAP NAS prepared for experiments.

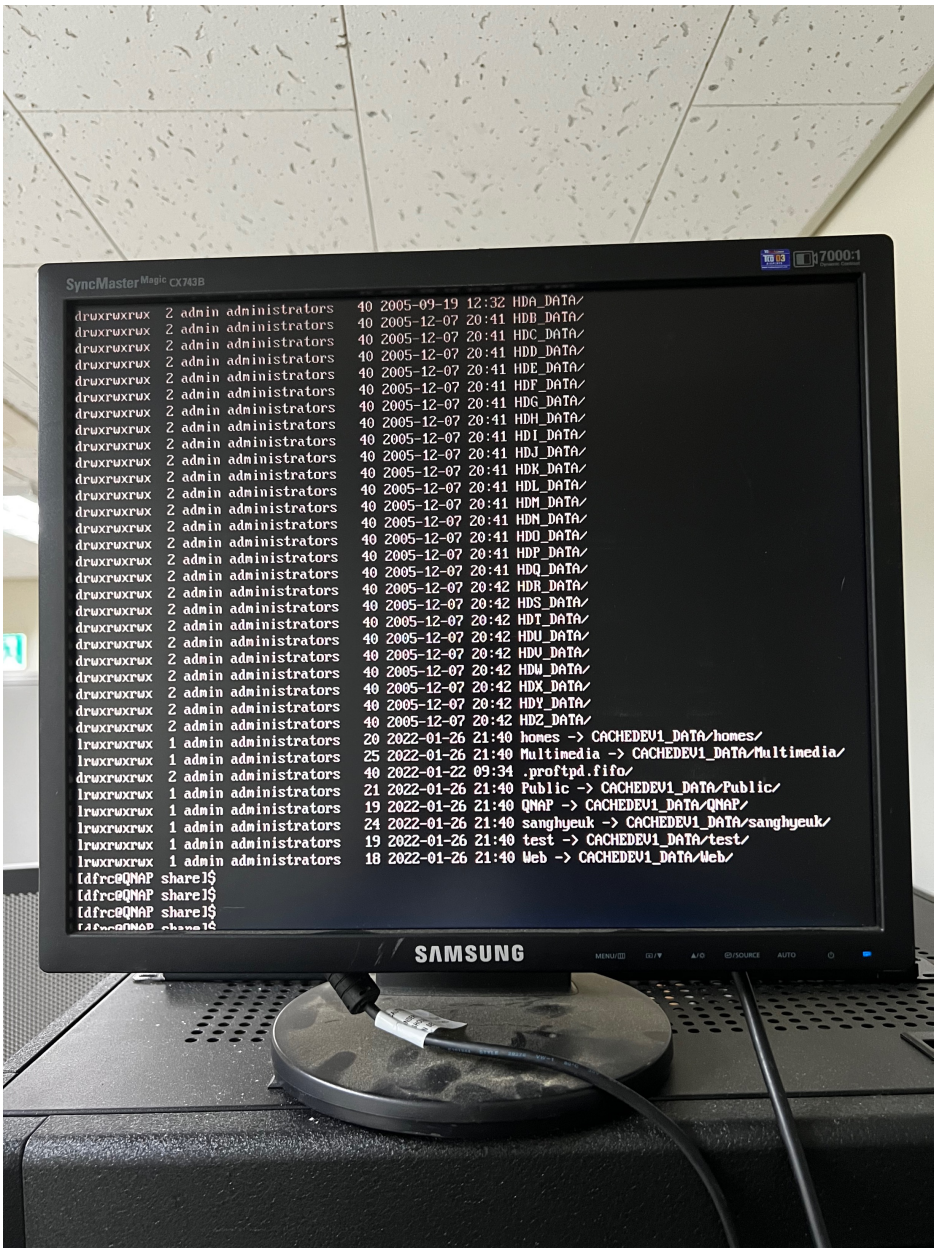
CODE: SELECT ALL

```
[/share] # ls
CACHEDEV1_DATA/ HDE_DATA/       HDM_DATA/       HDU_DATA/       Web@
CACHEDEV2_DATA/ HDF_DATA/       HDN_DATA/       HDV_DATA/       external/
Container@      HDG_DATA/       HDO_DATA/       HDW_DATA/       homes@
Download@       HDH_DATA/       HDP_DATA/       HDX_DATA/       snapshot/
HDA_DATA/       HDI_DATA/       HDQ_DATA/       HDY_DATA/
HDB_DATA/       HDJ_DATA/       HDR_DATA/       HDZ_DATA/
HDC_DATA/       HDK_DATA/       HDS_DATA/       Multimedia@
HDD_DATA/       HDL_DATA/       HDT_DATA/       Public@
```

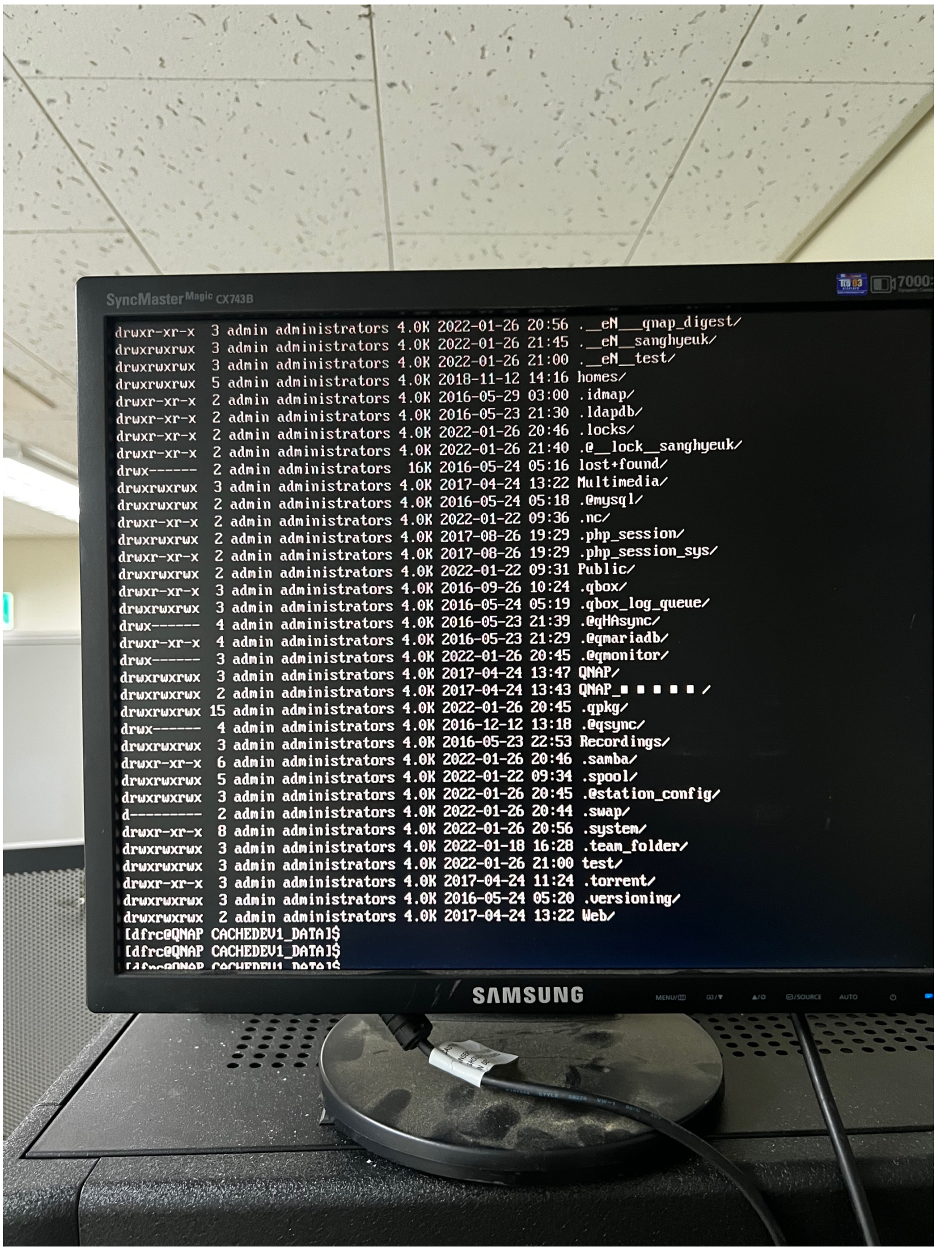
[4-24] QNAP NAS - Directory Structure

To find out that research matches the actual results, vault function was applied in physical QNAP NAS server. The directory saving the data was mounted in `/share/CACHEDEV1_DATA` (Figure [4-24]). Directory named “sanghyeuk” was created to test Vault. In Figure [4-25], directory is found under the path `/share`, which is a symbolic link to the path given above. In the real path of directory, sample data that was created could be seen through shell.

After applying Vault features, the directory was seen as Figure [4-26]. The name of directory is changed as `.@__lock__sanghyeuk/`. It can bee seen as the directory is hidden by the dot(.) in front of the directory name. In this case no files were seen through shell.



[4-25] QNAP NAS “sanghyeuk” before applying Vault feature



[4-26] QNAP NAS "sanghyuek" after applying Vault feature

Therefore from the data that was accessible, such as configuration files and web pages in which NAS service was accessed in web. Analysis was carried out to find the path where data is saved. The result of analysis is found as Figure [4-27], which designates `/.samba/lock/locking.tdb` file. From this file, it was able to presume that the file was in `/share/CACHEDEV2_DATA/Vault`. From the message mentioned in Part 3, decoded card number of the skimmer device will be found in `/share/CACHEDEV2/Vault/cards/bcv_metrop.txt`.

```
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BB8
/share/CACHEDEV2_DATA/Vault
cards/bcv_metrop.txt
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
/share/CACHEDEV1_DATA/Multimedia
```

[4-27] /$/.samba/lock/locking.tdb

Furthermore, Encrypted share folder with name “Vault” was found in `/.logs/event.log`

2021-04-11	20:49:55	admin	172.22.30.100	---	[Encrypted Share Folder : Vault] Created successfully.

[4-28] /.logs/event.log

Name of DataVol1 (lv1) volume is `/share/CACHEDEV1_DATA`. So the partition that owns the Vault folder will be DataVol2, which is lv2 found in the superblock of lv2 in Figure [4-29]. As descripted in front of Part 4, it is not known whether it was because of thin provisioning or because the filesystem was encrypted, but the filesystem was not recognized.

```
00 00 EE 02 00 00 70 17 00 00 02 00 2F 68 3C 17  ..î...p...../h<.
F3 FF ED 02 00 00 00 00 02 00 00 00 02 00 00 00  óÿí.............
00 80 00 00 00 80 00 00 00 10 00 00 FC 3B 73 60  .€...€......ü;s`
FC 3B 73 60 03 00 FF FF 53 EF 01 00 01 00 00 00  ü;s`..ÿÿSï......
3E 1F 73 60 00 00 00 00 00 00 00 00 01 00 00 00  >.s`............
00 00 00 00 0B 00 00 00 00 01 00 00 1C 00 00 00  ................
C6 02 00 00 7B 00 00 00 A7 E6 8E 4E 62 92 4F 34  Æ...{...§æŽNb'O4
AB C6 42 03 2E CF 81 9B 44 61 74 61 56 6F 6C 32  «ÆB..Ï.›DataVol2
00 00 00 00 00 00 00 00 2F 73 68 61 72 65 2F 43  ......../share/C
41 43 48 45 44 45 56 32 5F 44 41 54 41 00 00 00  ACHEDEV2_DATA...
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 04  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
08 00 00 00 00 00 00 00 00 00 00 00 29 60 94 F4  ............)`”ô
15 B2 42 AD B9 D9 0D DF E3 62 E9 3F 01 01 40 00  .²B.¹Ù.ßãbé?..@.
00 00 00 00 00 00 00 3E 1F 73 60 0A F3 01 00     ........>.s`.ó..
04 00 01 00 00 00 00 00 00 00 00 FF 7F B8 0B     ............ÿ.¸.
00 00 00 00 00 80 00 00 00 80 00 00 00 00 B9 0B  .....€...€....¹.
00 00 01 00 00 80 00 00 00 80 B9 0B 00 80 01 00  .....€...€¹..€..
00 80 00 00 00 00 BA 0B 00 00 00 00 00 00 00 40  .€....º........@
00 00 00 00 00 00 00 00 00 00 00 00 20 00 20 00  ............ . .
01 01 00 00 80 00 00 00 00 00 00 00 00 00 00 00  ....€...........
00 01 00 00 04 00 00 00 80 A4 CB 00 00 00 00 00  ........€¤Ë.....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

[4-29] Superblock of lv2 partition

[Conclusion]

In this part, communication history using Mattermost and web browser history relating to Defcad.com were identified. Based on these information, arnim.zola initialized 3D printer server. And the word “Vault” was mentioned in the conversation. Vault was used to encrypt shared folders which contain files and information that was mention in the conversation. red.skull downloaded STL files which are parts of a Liberator and shared the file via Vault.

Part 5. Multisource Analysis

[Introduction]

With the incredible speed of developments in electronic and internet industry, criminal acts became more advanced and meticulous furthermore not only a single digital source is used nowadays. In this situation, investigators must have a unified methodology to analyze and correlate potential digital evidence from multiple sources.

With the DFRWS Challenge 2021, it was truly challenging to examine each source of interest and to correlate them with a single story. There has been practical guidelines for a single source such as mobile, desktop, etc. However, research efforts connecting in between different sources has not been done enough.

So, in this Part 5, a unified guideline is proposed that is specialized with multi-sources. Believing that it will help investigators to investigate in efficiency with increasing amounts of data from multitude of sources.

Briefly, the process being introduced is designed “Top-Down” in 6-steps, with analysis done in each step, gradually helps building logical timeline that explains users’ activities at last.

[Approach]

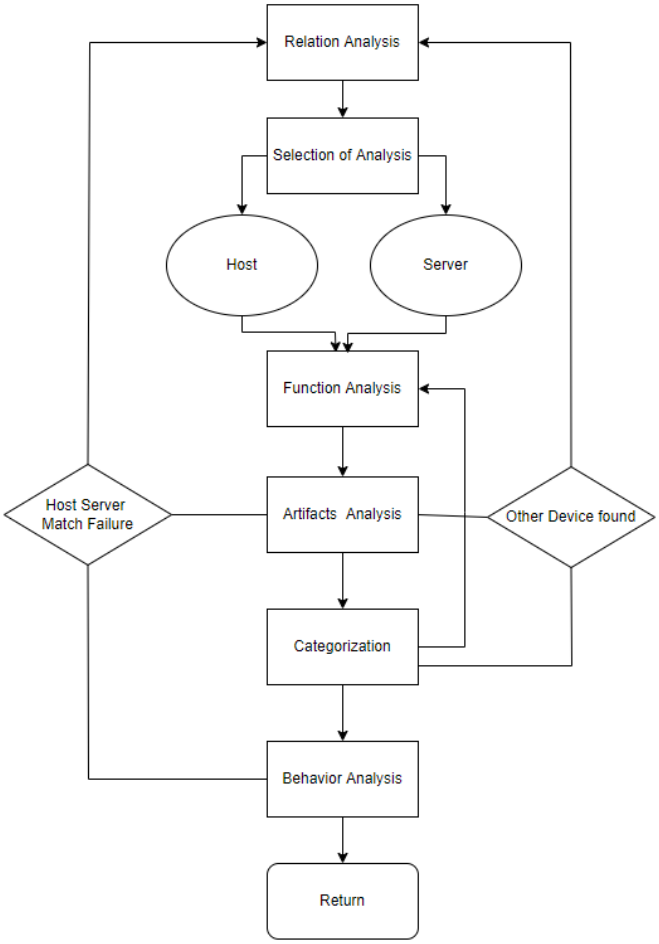
The goal of this part is to advance the state-of-the-art in multisource analysis and correlation. When there is a case with multisource, the most important thing is to save time for investigators and seeing a new view through correlation analysis on multi-source data. Because any kind of analysis method which saves lots of time needs a new approach because of characteristic of multisource analysis. This idea comes from the importance of recognizing the relationship between multiple sources not focusing on the individual one.

By solving the challenge given, the main theme was to design a new process that will cut off time no matter what kind of source device is given. The process presented in here includes 6-steps. Relation analysis, Selection of analysis, Function analysis, Artifact analysis, Categorization, and Behavior analysis. Time saving was achieved in Selection of analysis and Categorization parts, which gives convenience inspecting the entire case. In the next section, detailed description of the proposed approach will be provided.

[Preparation]

Preparing forensic knowledge for various H/W and S/W. For multisource analysis, there needs to understand basic knowledge of individual potential source of evidence, and to prepare forensic tools and techniques for handling it properly. This preparation is necessary not only for multisource analysis but also for normal forensic analysis. Therefore, we explain the proposed process on the premise that basic knowledge for handling target H/W and S/W are already shared between stakeholders. Also, it is necessary to understand about the case in detail. That is to say, digital forensic examiners need to check what case is about, and think about which of events to focus on. For example, in the case of a confidential information leakage, we should focus on file copying, moving traces, or messenger artifact.

[Proposed Process]



[5-1] A workflow of the proposed multisource analysis process

1. **Relation Analysis**

   The first step of the whole process is to define relationship between sources. Between sources, any type of relation can occur thorough linkages such as Bluetooth, LTE, Wi-Fi, USB interface, and so on. Based on the characteristic of the source device, every linkage should be listed up before analysis based on possibilities.

2. **Selection of Analysis**

   After the previous step, privilege has to be set based on the relations of source devices. Every relation is defined based on information, and the part that is significant is where these data intersects. It basically means that giving roles are needed, which device will be client or server (host). With both types, analysis works must be done in parallel, in order to validate their relationship. For this reason, top-down and recursive method was chosen.

   After selection of sources to analyze, individual source has to be analyzed as normal single-source case is. Analysis of individual source needs understanding the source of interest, preparing appropriate forensic tools and techniques to start analysis. Then, acquisition and preservation of the evidence is needed.

3. **Function Analysis**

   Analysis of function is necessary for logical reasons. Logical means that there must be a justification when moving on to the next step. Although this process does not practically find forensically meaningful data, it helps to discover what situation could happen. It distinguishes specific functions that could have been used. Experiments and validation are required to apprehend existing features of interest.

4. **Artifact Analysis**

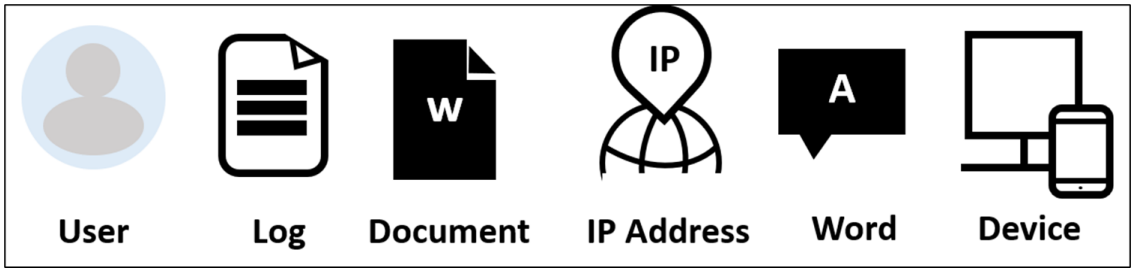
   Artifacts in this part mean as follows. It means directly or indirectly obtaining data remaining on the acquired device. This information is essential for cross checking data and putting strength on the evidence that is previously found or even helps to find a new data that is not found. Academical skills are required if necessary, such as reverse engineering knowledge and filesystem concepts. Utilizing existing tools and developing new tools are also important for doing that.

5. **Categorization**

   Results of analysis can be generated by using automated tools. However, investigators had to make their own decision based on the results and then find the path or evidence to make a conclusion. So, it would be more efficient if the results can be categorized based on various results. Categorization consists of two parts: (1) integrating data identified from multiple sources, and (2) correlating multisource information.

   “Integration” can be done by various factors. Most commonly used factor which is recommended is listed as follows:

   - Based on timestamps
   - Based on identities (name , ID, nickname, email, IP address, etc.)
   - Based on communication activities

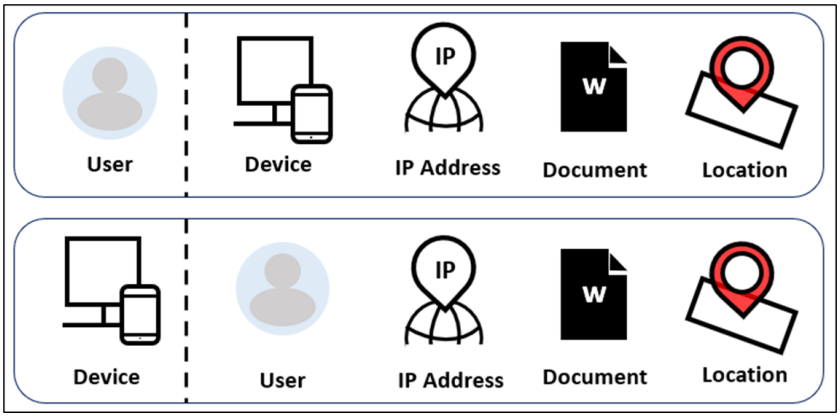


[5-2] Categorization Example

   “Correlation” of multisource information also has many factors. Some examples of those factors are provided as follows:

   - Analysis of communication activities (SMS, messenger, SNS, etc.)
   - Analysis of suspicious activities related to the case
   - Analysis of techniques used for criminal activities (destruction, deletion, encryption, obfuscation, etc.)

   Also, it is necessary to match each factor with the other factors. Factors in the “User” category can match with device, IP address, location, or etc. This step can be of great help in analyzing user behaviors.

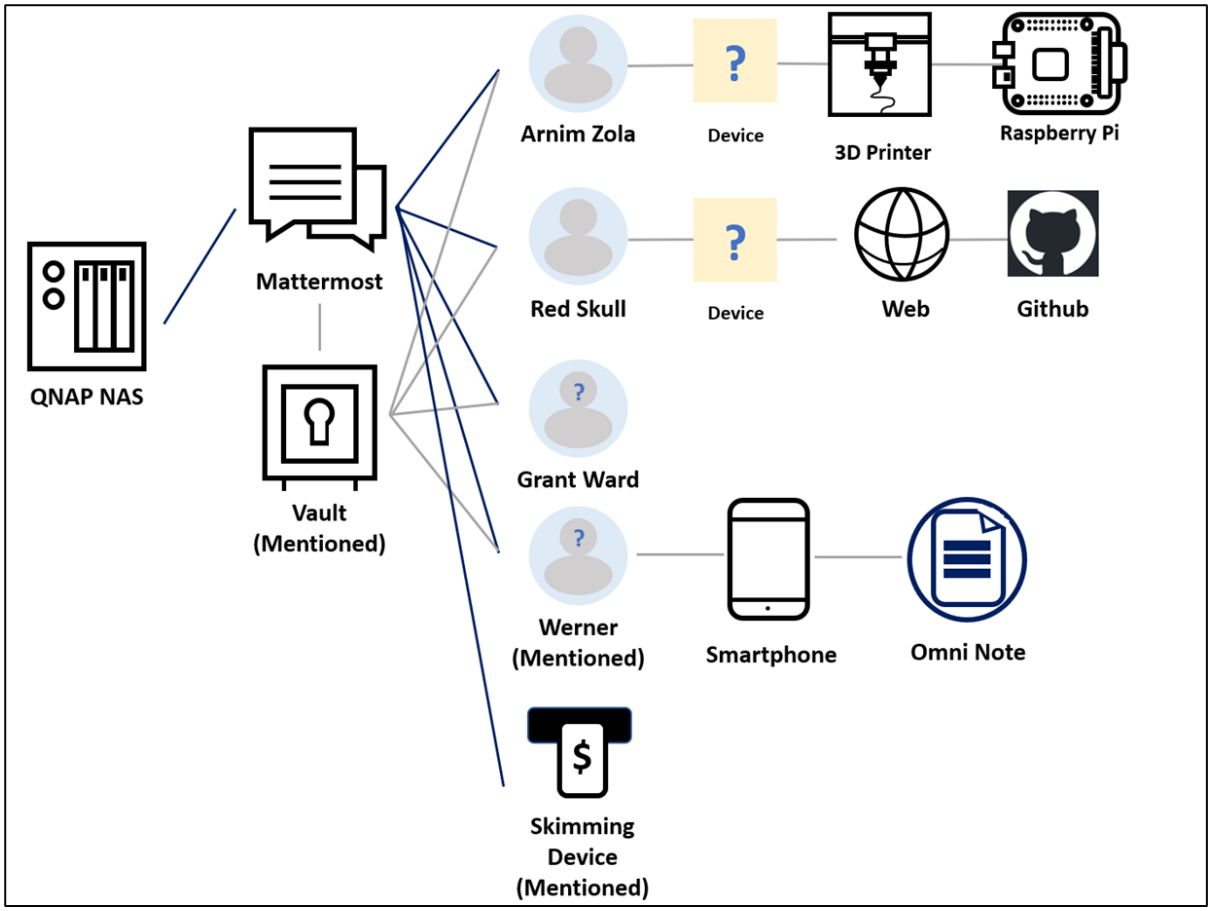


[5-3] Matching Example

6. **Behavior Analysis**

   Using the categorized data, logical results are derived. Including the relationships between devices, a specific result of analysis has to be mentioned. With this step, advanced timeline is constructed to show correlation. Through 1~5 steps, it is possible to establish the relationship between sources of interest and user behaviors more easily.

[Interpretation & Examination of Challenge Data]



[5-4] Integrative Process of Challenge Case

1. **Relation Analysis**

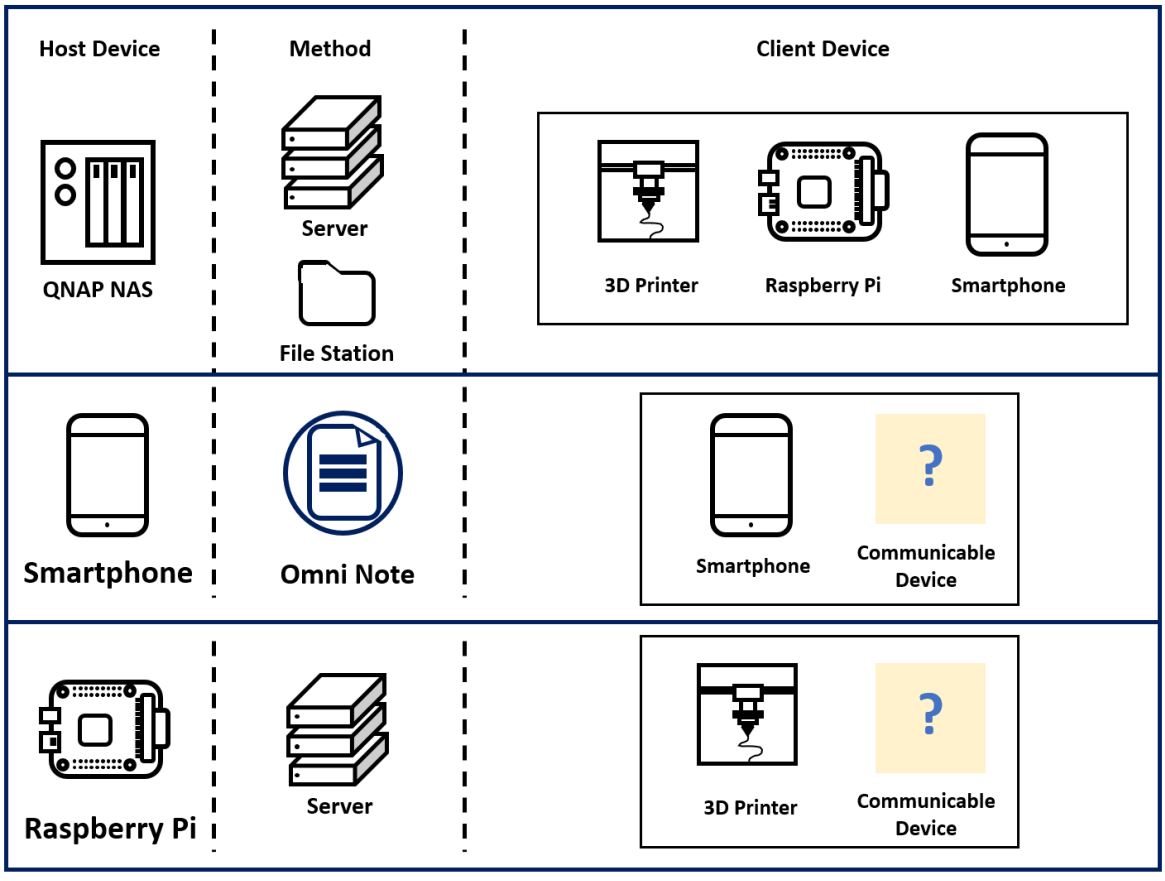
The QNAP NAS uses “qpkg” to host several services including Mattermost, which can be used as a communication system. And these services can be used with other devices such as Raspberry Pi. Even though describing these relationships are important, listing these possibilities in tables are more readable as Table [5-1].

[Table 5-1] Relation Analysis on Challenge Case

Method	source1 (Host)	source2 (Client)
Messenger	QNAP NAS Samsung Smartphone	Raspberry Pi Samsung Smartphone Unkown Device
File Station	QNAP NAS	Raspberry Pi Samsung Smartphone Unkown Device
Encrypted Folder/File Station	QNAP NAS	Raspberry Pi Samsung Smartphone Unkown Device
User Distribution	QNAP NAS Raspberry Pi	Unkown Device
Customized Application	Samsung Smartphone	Raspberry Pi Samsung Smartphone Unkown Device
Physical Connection	QNAP NAS	Raspberry Pi Samsung Smartphone Skimming Device Unkown Device
Physical Connection	Raspberry Pi	3D Printer Samsung Smartphone Unkown Device
Phone Call	Samsung Smartphone	Unkown Device

2. **Selection of Analysis**

Thinking about the image we have, there are six interaction cross-sections that QNAP NAS have. Therefore, the first forensic investigation can be performed for the QNAP NAS with the most interaction cross-sections.

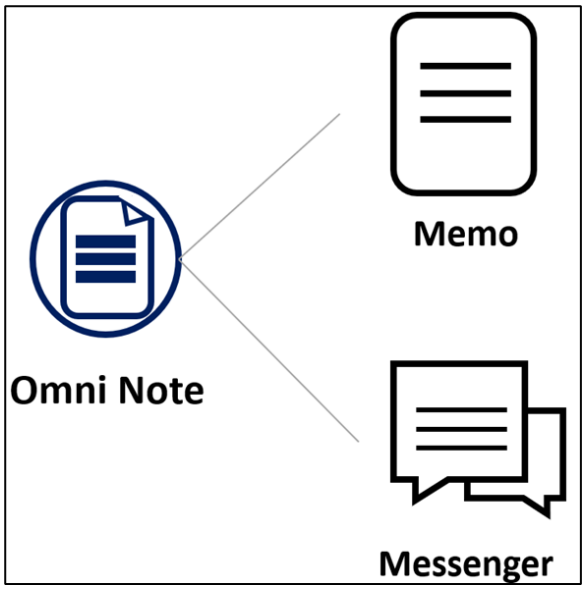


[5-5] Real-relation Analysis on Challenge Sources

The real-relation analysis picture of the device and the rest of the devices identified in this case is shown in Figure [5-5]. Among them, it can be seen that the QNAP NAS is most likely to be related with three client devices using two methods. In Table [5-1], it was confirmed that the NAS had the most connection points with other devices. Therefore, if examiners investigate and analyze the NAS first, they will be able to find connections with other deices, and it will be easy to connect to things common to analysis of other devices. There is a high probability that a device, which has the most intersections in the previous ‘Relation Analysis’, will eventually have the most intersections when it comes to real-relation analysis. Due to this reason QNAP NAS was selected initially.

3. **Function Analysis**

   A complete understanding of applications in each devices given. For a Samsung smartphone, full functions of the suspicious Omni Note application was needed to identify the message sending page. In case of the QNAP NAS, Mattermost is the application that requires knowledge of database structure of PostgreSQL. In this case, Omni Note has a hidden function on sending messages. Examiners can determine the range of checking artifacts if it is possible to know the fact that these functions exist and are executable.



[5-6] Function Analysis on Challenge Sources: Omni Note as an Example

4. **Artifact Analysis**

   In order to analyze the given images, examiners should know and understand some forensic knowledge, including but not limited to NAS, smartphone, Linux, Android, APK, web browser application, messaging service, RAID, magnetic stripe, and S/W reverse engineering skills. In addition, for every image, unallocated areas were analyzed for deleted files that could have meaningful information. Although it was not useful for every image, meaningful files were carved from the skimmer device related image. Beside the data which was manually found, existing tools (e.g., Autopsy) were used to get artifacts such as e-mails and registered users, web browser history, messages, OS information, and SNS related data. Every data found in the current step will be used in the next step.

5. **Categorization**

   Using existing tools like **Autopsy**, it was able to get forensically meaningful artifacts. But the tools usually do not provide functions to correlate between those artifacts. It is up to investigators relating the given artifacts. So, in order to do that, artifacts found in the previous step are needed to be categorized properly in this step. Table [5-2] lists the categorized results associated with multiple artifacts processed in the ‘Artifact Analysis’ step. Through categorization, relationships are made simply connected by each categorization factor. The path of evidence in the source is also given if it is a parsed information. For reference, any source is not given when it is carved from unallocated areas.

[Table 5-2] Categorized information

Categorization Factor	Sources
Skimmer Device(Keyword)	Samsung smartphone, Skimmer device
Leaving to Geneve(Event)	Samsung smartphone, Skimmer device
Symbol of Hydra(img)	QNAP NAS, Samsung smartphone, Skimmer device
3D printed Liberator(Keyword)	QNAP NAS, Raspberry Pi
Skimmer installation(Event)	Samsung smartphone, Skimmer device
Mattermost(Service)	QNAP NAS, Samsung smartphone
Vault(Service)	QNAP NAS, Raspberry Pi, Skimmer device
(Person)	QNAP NAS, Raspberry Pi, Samsung smartphone, Skimmer device
CC data shared	Samsung smartphone, Skimmer device

6. **Behavior Analysis**

   In this step, details on the case will be analyzed by every categorized factors. Information such as “why it is related” and “what is learned from the relation” can be obtained as results. For the challenge dataset, “Description” in Table [5-3] explains the detailed behaviors revealed through this analysis.

   After analysis, integrating and rearranging every events from multiple sources individually by timestamp, an integrated timeline on the case is built as listed in Table [5-4].

[Table 5-3] Behavior Analysis Result

Categorization Factor	sources	Connected	Description
Skimmer Device(Keyword)	Samsung smartphone, Skimmer device	O	ATM skimmer machine was filmed by smartphone.
Leaving to Geneve(Event)	Samsung smartphone, Skimmer device	X	SNS article that victim is leaving matches with the calendar information. Train ticket from Lausanne to Geneve was found. Date mismatches.
Symbol of Hydra(img)	QNAP NAS, Samsung smartphone, Skimmer device	O	Symbol image was found in every images.
3D printed Liberator(Keyword)	QNAP NAS, Raspberry Pi	O	Same blueprint was found, which turns out to be Liberator Also found by DEFCAD.com File stored in QNAP NAS Vault
Skimmer installation(Event)	Samsung smartphone, Skimmer device	O	Skimmer installed
Mattermost(Service)	QNAP NAS, Samsung smartphone	O	Email on the smartphone implies another channel. Mattermost found in QNAP NAS
Vault(Service)	QNAP NAS, Raspberry Pi, Skimmer device	O	Decoded CC numbers stored in Vault Blueprints(STL) stored in Vault Vault is serviced in QNAP NAS
(Person)	QNAP NAS, Raspberry Pi, Samsung smartphone, Skimmer device	O	**Arnim Zola** → Skimmer device, Raspberry Pi, QNAP NAS **Redskull** → Samsung Smartphone, QNAP NAS **Werener Von Strucker** → Samsung Smartphone, QNAP NAS **Grant Ward** → QNAP NAS

Categorization Factor	sources	Connected	Description
CC data shared	Samsung smartphone Skimmer device	O	identical mp3 file was found in each device since the file time is faster in smartphone, and data was not extracted in the file, this file seems to be a sample file.
Omninote installed	Samsung smartphone		

[Table 5-4] Integrated Timeline on the Case

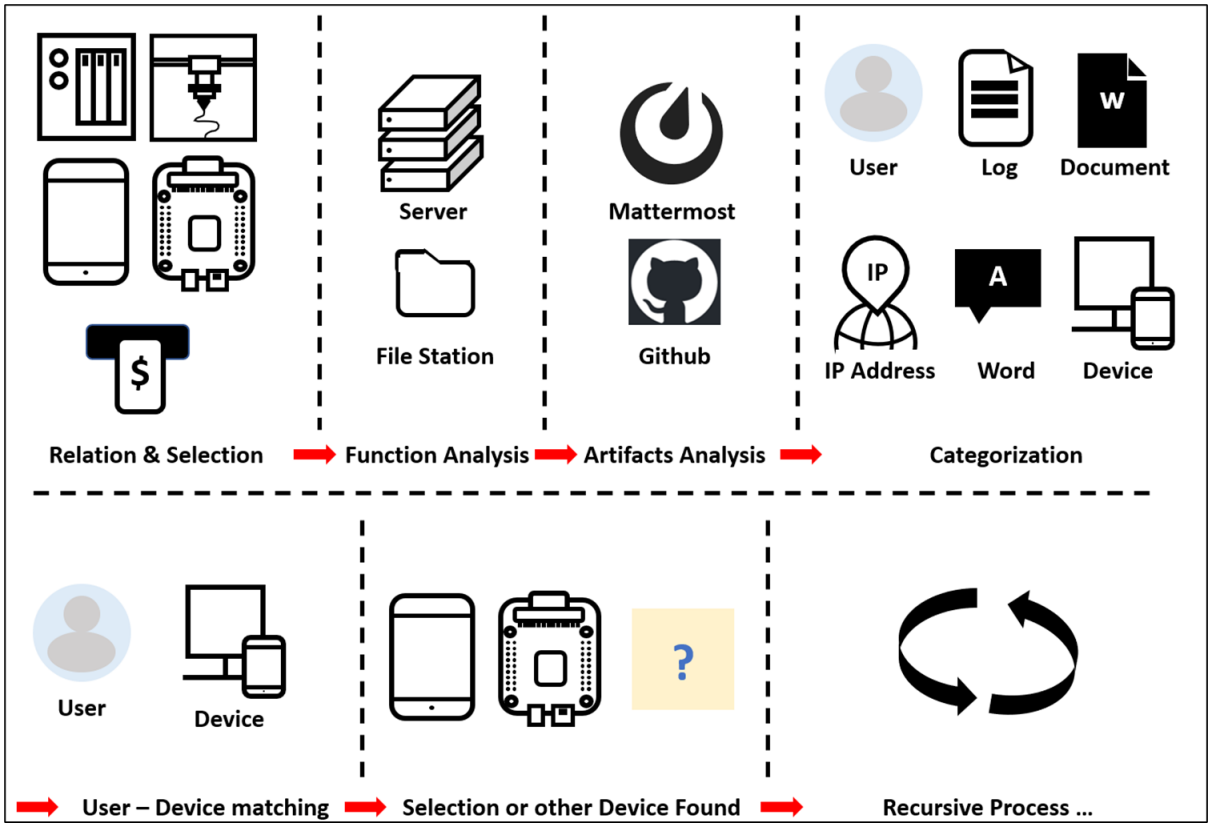
Event	Source	Date (UTC +02:00)	Description
Laussane → Geneve	Skimmer Device	2021/03/28 09:21	carved from unallocated space
Omninote installed	Smartphone	2021/04/08 11:59	
Omninote conversation	Smartphone	2021/04/09 16:10 ~ 2021/04/09 16:15	
Email	Smartphone	2021/04/08 17:04 ~ 2021/04/08 17:06	redskull & werner
Omninote conversation	Smartphone	2021/04/08 18:09 ~ 2021/04/08 18:16	
Sample mp3 file	Smartphone	2021/04/09 17:24	identical mp3 file was found in each device since the file time is faster in smartphone, and data was not extracted in the file, this file seems to be a sample file.
Sample mp3 file	Skimmer Device	2021/04/09 16:22	identical mp3 file was found in each device since the file time is faster in smartphone, and data was not extracted in the file, this file seems to be a sample file.
Victim Data skimmed	Skimmer Device	2021/04/09 16:20 ~ 2021/04/09 16:25	Modified time of Recording.mp3 which has the CC number.
Skimmer Discovered	Skimmer Device	2021/04/09 16:25	information given in the introduction.
Vault Created	QNAP NAS	2021/04/11 20:49	found in event.log file
DEFCAD visited	QNAP NAS	2021/04/11 21:38	by Arnim Zola
Mattermost conversation	QNAP NAS	2021/04/11 23:55	by Arnim zola & Red Skull
3D printer Browsing started	QNAP NAS	2021/04/12 14:06	
3D printer information Download start	QNAP NAS	2021/04/12 14:08	
Images of Liberator shared	QNAP NAS	2021/04/12 14:48	via Mattermost
Mattermost conversation	QNAP NAS	2021/04/12 14:48 ~ 2021/04/12 14:57	by Arnim Zola & Red Skull
Octoprint Server Turned On	Raspberry Pi	2021/04/13 12:38	

Event	Source	Date (UTC +02:00)	Description
Liberator parts printed	Raspberry Pi	2021-04-13 14:27 ~ 2021-04-14 17:29	
Mattermost visited	QNAP NAS	2021/04/17 20:34	via Chrome Cache by redskull
Mattermost conversation	QNAP NAS	2021/04/17 20:07 ~ 2021/04/17 20:19	by Arnim Zola & Red Skull
Raspberri Pi Lab found	Raspberry Pi	2021/04/18	Informaiton given in the introduction
Instagram	Smartphone	2021/04/20 13:01	“ready to leave”
Lausanne→Geneve	Smartphone	2021/04/20 16:45	Found in google calendar reminder
Samsung Smartphone extracted	Smartphone	2021/04/21	Information given in the introduction
Arrested in Geneve	Smartphone	2021/04/21 18:30	information given in the introduction
Vault		2021-04-28 14:49	found in Mattermost messages
Mattermost conversation	QNAP NAS	2021/04/28 14:49 ~ 2021/04/28 14:51	by Arnim Zola & Red Skull
Mattermost conversation	QNAP NAS	2021/04/29 07:51 ~ 2021/04/29 07:52	by Grant Ward
NAS discovered	QNAP NAS	2021/04/29	Information given in the introduction

Overall Process

Figure [5-7] shows an integrated process on handling the challenge dataset. As we explained above, the QNAP NAS with the most interactions are selected as the first target source. Then, examiners analyze functions that can be using the QNAP NAS and checking artifacts corresponding to it. The artifacts analyzed by using existing or self-developed tools can be categorized. Various factors in each category can be matched with factors in another category, and while proceeding with this, other devices can be detected as new sources of interest. The next step is recursively operated again. The device with many connections is selected, and then functions and artifacts are analyzed.

After this repetitive step, it is easy to judge the behavior of the user by using the categorization and matching factors. Therefore, the proposed process can discover correlation between multisource information as well as reduce the time for forensic analysis.



[5-7] Apply Process - Challenge Source

[Conclusion]

The most challenging part of multisource analysis is difficult to determine which source to analyze first and which artifacts to investigate in each source. With the proposed methodology, investigators will not get lost in the flood of information, by selection a source to analyze first and then relating sources logically. In multi source analysis, the most important thing is the correlation between different sources, and it can be found in places where data intersects.

🎓Appendix

https://github.com/dfrc-korea/DFRWS_Challenge_2021

Appendix Table

num	File name	Path
1-1	[1-1] main.py	DFRC_Appendix.zip/Part 1 Skimming Device/[1-1]main.py
3-1	[3-1] mmssms.db	DFRC_Appendix.zip/Part3.Samsung Smartphone/[3-1] mmssms.db
3-2	[3-2] key.json	DFRC_Appendix.zip/Part3.Samsung Smartphone/[3-2] key.json
3-3	[3-3] dec_mmssms.db	DFRC_Appendix.zip/Part3.Samsung Smartphone/[3-3] dec_mmssms.db
3-4	[3-4] decrypt_message.py	DFRC_Appendix.zip/Part3.Samsung Smartphone/[3-4] decrypt_message.py
3-5	[3-5] calendar.db	DFRC_Appendix.zip/Part3.Samsung Smartphone/[3-5] calendar.db
4-1	[4-1] hybridRAIDReconstructor.py	DFRC_Appendix.zip/Part4.QNAP_NAS/[4-1]hybridRAIDReconstructor.py
4-2	[4-2] LVM.txt	DFRC_Appendix.zip/Part4.QNAP_NAS/[4-2]LVM.txt
4-3	[4-3] parseLogicalParti.py	DFRC_Appendix.zip/Part4.QNAP_NAS/[4-3]parseLogicalParti.py
4-4	[4-4] History.sqlite	DFRC_Appendix.zip/Part4.QNAP_NAS/[4-4]History.sqlite
4-5	[4-5] mattermost_messages.csv	DFRC_Appendix.zip/Part4.QNAP_NAS/[4-5]mattermost_messages.csv
4-6	[4-6] places.sqlite	DFRC_Appendix.zip/Part4.QNAP_NAS/[4-6]places.sqlite

🎓References

Reference Table

Title	part	Url
Crooks Rock Audio-based ATM Skimmers	skimming device	https://krebsonsecurity.com/2010/11/crooks-rock-audio-based-atm-skimmers/
Information technology — Identification cards — Financial transaction cards	skimming device	https://www.iso.org/standard/43317.html
Octoprint	Raspberry Pi	https://github.com/guysoft/OctoPi
Cura	Raspberry Pi	https://ultimaker.com/ko/software/ultimaker-cura
Arnim Zola	multisource analysis skimming device	https://marvelcinematicuniverse.fandom.com/wiki/Arnim_Zola
Hydra	multisource analysis skimming device	https://marvelcinematicuniverse.fandom.com/wiki/HYDRA
Magstripe	skimming device	http://www.alcrypto.co.uk/magstripe/magstripe-defcon-2006-final.pdf
dab.py	skimming device	http://www.alcrypto.co.uk/magstripe/?C=M;O=A
firefox browser history	QNAP NAS multisource analysis	https://wiki.mozilla.org/images/0/08/Places.sqlite.schema.pdf
firefox decrypt	QNAP NAS multisource analysis	https://github.com/unode/firefox_decrypt
US Case Laws & State Legislations - Skimmers	skimming device	https://www.ncsl.org/research/financial-services-and-commerce/credit-card-skimming-devices-laws-and-legislation.aspx
Reassembling Linux-based Hybrid RAID	QNAP NAS	https://pubmed.ncbi.nlm.nih.gov/31868925/
Hybrid RAID Reconstructor	QNAP NAS	https://github.com/antares0531/HybridRAIDReconstructor
GPT partition Table	QNAP NAS	https://en.wikipedia.org/wiki/GUID_Partition_Table
RAID superblock formats	QNAP NAS	https://raid.wiki.kernel.org/index.php/RAID_superblock_formats
Left-symmetric layout	QNAP NAS	https://sort.veritas.com/public/documents/sf/5.1/aix/html/vxvm_admin/ch01s04s09s03.htm