

작성자	윤지원
분석 일자	2024.05.23 ~ 27
작성 일자	2024.05.23 ~ 27
분석 대상	trudy_pc.ad1
문서 버전	2.0
작성자 E-mail	yoonjw0827@gmail.com

0. 목차

1. 문제3

2. 분석 도구3

3. 환경3


4. Write-Up.....4

5. Flag..... 18

6. 별도 첨부 19

7. Reference 20

1. 문제

URL	-
문제 내용	<p>Description Investigators raided the office after receiving an anonymous tip that a spy was targeting someone. Investigators collected some data while the spy was destroying evidence. Analyze the collected image to find the orders and missions the spy received.</p> <p>Questions</p> <ol style="list-style-type: none"> 1) When is the spy's mission date? 2) Where is the spy's mission location? 3) Who is the spy targeting?
문제 파일	 trudy_pc.ad1
문제 유형	System forensics
난이도	3 / 3

2. 분석 도구

도구명	다운로드 링크	Version
FTK Imager	https://www.exterro.com/digital-forensics-software/ftk-imager	4.7.1.2
DB Browser(SQLite)	https://sqlitebrowser.org/dl/	3.12.2
Wireshark	Wireshark · Download	3.4.7
Audacity	https://www.audacityteam.org/download/	3.5.1

3. 환경

OS
Windows 11 64-bit

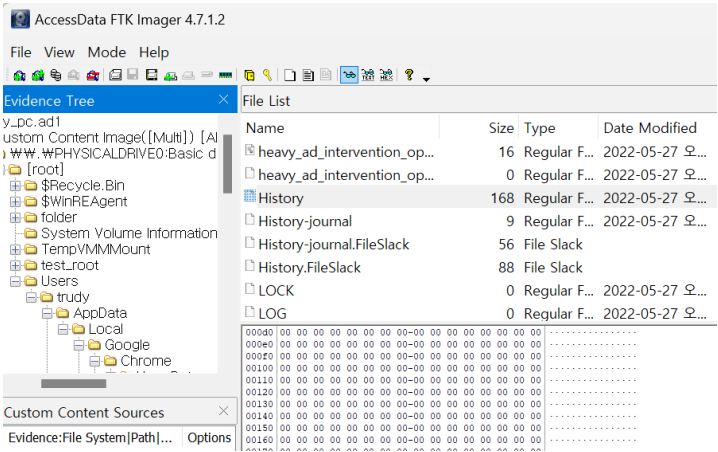
4. Write-Up

파일명	trudy_pc.ad1
용량	67MB
SHA256	b6718d717d16ca2fe049ac81001b94cde4fc433998d4b2bda4a578b71f595675
Timestamp	2022-05-30 16:15:47

문제를 요약하자면 스파이가 누군가를 목표로 하고 있으며, 수집된 이미지를 분석하여 스파이가 받은 명령과 임무를 찾는 문제이다.

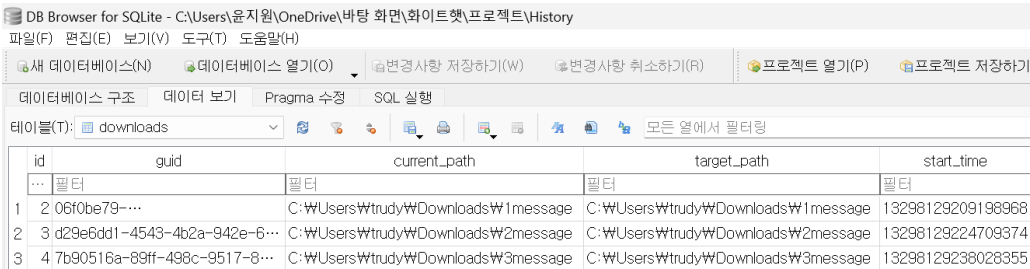
1. 스파이의 임무 수행일은 언제인가?

우선 이 스파이가 언제 임무를 수행했는지를 보기 위해서는 과거 기록을 살펴보는 것이 좋을 것이라고 생각하여 FTK Imager 를 이용하여 파일을 열고, History 라는 파일이 있는지 살펴보았다. 그 결과, [root]\Users\trudy\AppData\Local\Google\Chrome\User Data\Default\History 에서 History 파일을 발견할 수 있었다.



[사진 1] History 파일을 발견한 모습

이 파일을 Export한 다음, DB Browser for SQLite 도구를 이용하여 열어보았다. 데이터베이스 구조를 살펴보니 downloads와 urls라는 테이블을 볼 수 있었다. 스파이가 다운받은 파일과 방문한 url 주소를 알아보면 임무를 수행한 방법과 시간을 알 수 있을 것 같아서 데이터 보기를 이용하여 각각의 테이블을 열어보았다.



id	guid	current_path	target_path	start_time
1	2 06f0be79-...	C:\Users\trudy\Downloads\W1message	C:\Users\trudy\Downloads\W1message	13298129209198968
2	3 d29e6dd1-4543-4b2a-942e-6...	C:\Users\trudy\Downloads\W2message	C:\Users\trudy\Downloads\W2message	13298129224709374
3	4 7b90516a-89f-498c-9517-8...	C:\Users\trudy\Downloads\W3message	C:\Users\trudy\Downloads\W3message	13298129238028355

[사진 2] downloads 테이블



DB Browser for SQLite - C:\Users\윤지환\OneDrive\배달의민서\사이트\프로젝트\History

파일(F) 편집(E) 보기(V) 도구(T) 도움말(H)

새 데이터베이스(N) 데이터베이스 열기(O) 변경사항 저장하기(W) 변경사항 취소하기(R) 프로젝트 열기(P) 프로젝트 저장하기(V) 데이터베이스 연결(A) 데이터베이스 닫기(C)

데이터베이스 구조 데이터 보기 Pragma 수정 SQL 실행

데이터베이스: urls

	id	url	title	visit_count	typed_count
	
1	1	https://gmail.com/	Gmail	1	1
2	2	https://www.google.com/gmail/	Gmail	1	0
3	3	https://mail.google.com/mail/	Gmail	1	0
4	4	https://mail.google.com/mail/u/0/	받은편지함 (15) - dcf.tudy@gmail.com - Gmail	2	0
5	5	https://accounts.google.com/ServiceLogin?service=mail&passive=120960&osid=1&continue=https://mail.google.com/mail/u/0/&flow=https://mail.google.com/mail/u/0/&emr=1	Gmail	2	0
6	6	https://accounts.google.com/signin/v2/identifier?...	Gmail	1	0
7	7	https://accounts.google.com/signin/v2/challenge/pwd?...	Gmail	1	0
8	8	https://accounts.google.com/CheckCookie?...	계정 복구 옵션	1	0
9	9	https://accounts.google.com/ServiceLogin?continue=https://mail.google.com/mail/u/0/&flow=https://mail.google.com/mail/u/0/&emr=1	계정 복구 옵션	1	0
10	10	https://myaccount.google.com/accounts/SetOSID?authuser=0&continue=https://mail.google.com/mail/u/0/&flow=https://mail.google.com/mail/u/0/&emr=1	계정 복구 옵션	1	0
11	11	https://myaccount.google.com/security/signinoptions/recovery-options-collection?...	계정 복구 옵션	1	0
12	12	https://myaccount.google.com/signinoptions/recovery-options-collection?...	계정 복구 옵션	1	0
13	13	https://myaccount.google.com/signinoptions/recovery-options-collection?...	계정 복구 옵션	2	0
14	14	https://accounts.google.com/ServiceLogin?...	받은편지함 (15) - dcf.tudy@gmail.com - Gmail	1	0
15	15	https://mail.google.com/accounts/SetOSID?...	받은편지함 (15) - dcf.tudy@gmail.com - Gmail	1	0
16	16	https://accounts.google.com/accounts/SetOSID?ssdc=1&sid=ALWU2csQkAgU7R4b4m1O1DaeWntfy2Eee0/1Q2E5EmuLZM70S2S6ZJqpZga&agSrvXSE9vL7aJq1Bk5Zu/...	받은편지함 (15) - dcf.tudy@gmail.com - Gmail	1	0
17	17	https://accounts.google.com/accounts/SetOSID?...	받은편지함 (15) - dcf.tudy@gmail.com - Gmail	1	0
18	18	https://mail.google.com/mail/u/0/?pli=1	받은편지함 (15) - dcf.tudy@gmail.com - Gmail	1	0
19	19	https://mail.google.com/mail/u/0/#inbox:	받은편지함 (15) - dcf.tudy@gmail.com - Gmail	4	0
20	20	https://mail.google.com/mail/u/0/#inbox/PfIczGpGBFGGpcwVwmrM2TB8bqPfs	Ah... Ah... - dcf.tudy@gmail.com - Gmail	1	0
21	21	https://mail.google.com/mail/u/0/#inbox/PfIczGpGBFGGTPd4rcwaFxbWt2dv	Trudy, Good luck... - dcf.tudy@gmail.com - Gmail	4	0
22	22	https://mail.google.com/mail/u/0/#inbox/PfIczGpGBFGGTPd4rcwaFxbWt2dv?project=1&messagePartId=0.1	Trudy, Good luck... - dcf.tudy@gmail.com - Gmail	3	0
23	23	https://mail.google.com/mail/u/0/#inbox/PfIczGpGBFGGTPd4rcwaFxbWt2dv?project=1&messagePartId=0.1	Trudy, FYI - dcf.tudy@gmail.com - Gmail	5	0
24	24	https://mail.google.com/mail/u/0/#inbox/PfIczGpGBFGGTPd4rcwaFxbWt2dv?project=1&messagePartId=0.1	Trudy, FYI - dcf.tudy@gmail.com - Gmail	4	0
25	25	https://mail.google.com/mail/u/0/#inbox/PfIczGpGBFGGTPd4rcwaFxbWt2dv?project=1&messagePartId=0.1	Trudy, God Bless You - dcf.tudy@gmail.com - Gmail	2	0
26	26	https://mail.google.com/mail/u/0/#inbox/PfIczGpGBFGGTPd4rcwaFxbWt2dv?project=1&messagePartId=0.1	Trudy, God Bless You - dcf.tudy@gmail.com - Gmail	2	0
27	27	https://chrome.google.com/webstore?hl=ko	Chrome 앱 스토어 - 확장 프로그램	1	0
28	28	https://chrome.google.com/webstore/category/extensions?hl=ko	Chrome 앱 스토어 - 확장 프로그램	1	0

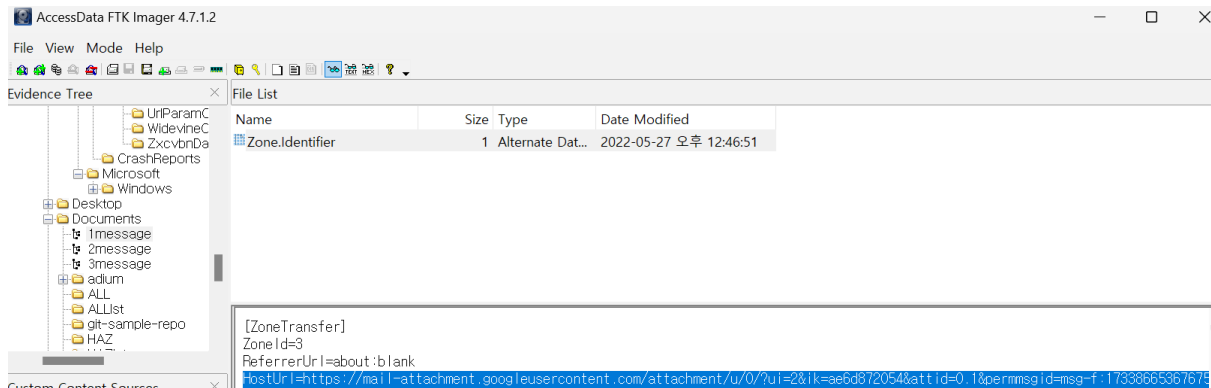
[사진 3]을 살펴보면 스파이가 gmail로 이메일을 받은 것을 볼 수 있는데, 메일 내용은 크게 3개로, 차례대로 **'Trudy, good luck', 'Trudy, FYI', 'Trudy, God Bless You'**이다. 그리고 [사진 2]의 다운로드 경로들을 보면 **1message, 2message, 3message**가 있는 것으로 보아 이 메시지들을 차례로 다운받았을 것이라고 추측할 수 있다. 이를 확실히 하기 위해 downloads 테이블의 뒷부분까지 확인한 결과, [사진 4]와 같이 다운로드 경로가 [사진 3]의 메일 주소와 일치하는 것을 확인할 수 있다.

The screenshot shows the AccessData FTK Imager 4.7.1.2 application window. The 'Evidence Tree' on the left displays a directory structure with folders like 'UriParamC', 'WidevineC', 'ZxcvbnDa', 'CrashReports', 'Microsoft', and 'Windows'. The 'File List' on the right shows a table of files and folders with columns for Name, Size, Type, and Date Modified.

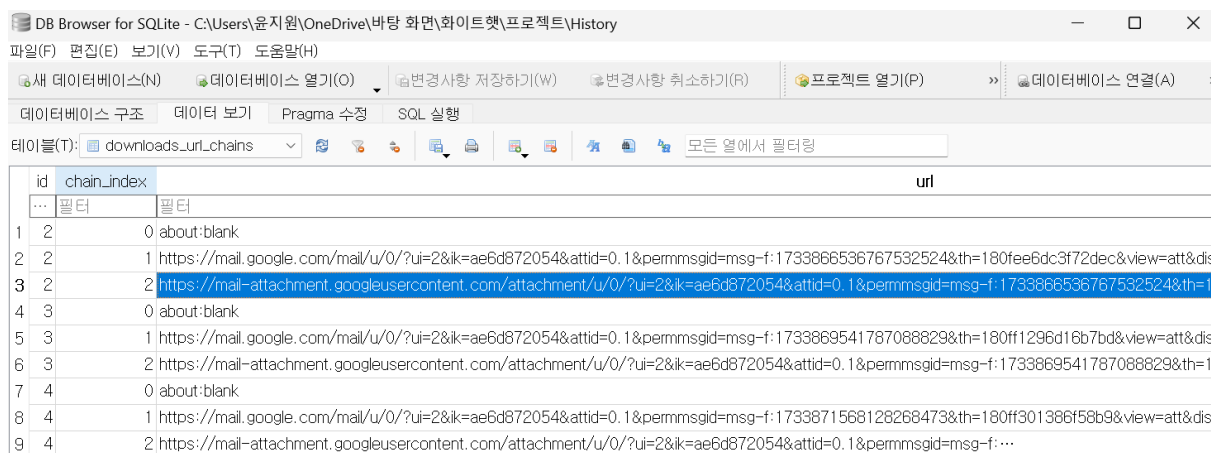
Name	Size	Type	Date Modified
ASP.NET Monsters #93 R...	3,107	Regular File	2021-10-19 오전 11:51:02
asbuitinventory.lst	1	Regular File	2021-10-18 오후 4:19:45
ADReport.htm	2	Regular File	2021-10-19 오전 3:48:41
3message	10,176	Regular File	2022-05-27 오후 12:47:19
2message	1,009	Regular File	2022-05-27 오후 12:47:05
1message	1,735	Regular File	2022-05-27 오후 12:46:51
!\$30	16	NTFS Index A...	2022-05-27 오후 12:47:42

5

[WHS-2] .iso



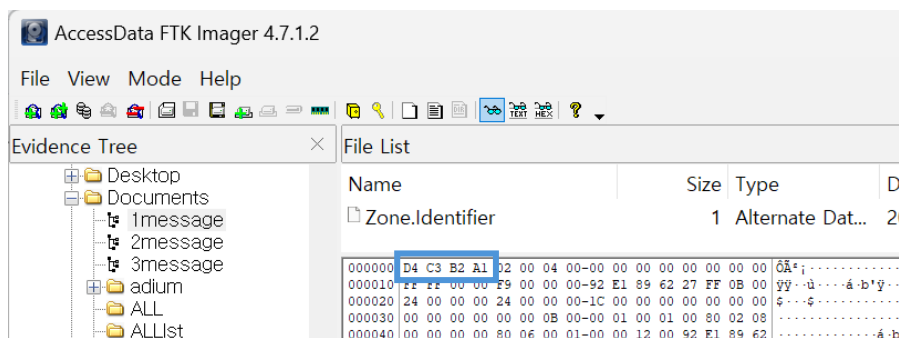
[사진 6] 1message의 Zone.Identifier의 HostUrl



[사진 7] downloads_rul_chains 테이블의 HostUrl

혹시나 다른 파일일 수도 있어서 메시지 파일들을 클릭해보니, 각각 [사진 6]과 같이 Zone.Identifier라는 파일이 들어있었다. 이곳의 HostUrl과 [사진 7]에서 확인할 수 있는 HostUrl이 동일하다는 것을 확인할 수 있기 때문에 [사진 4]와 [사진 5]의 message 파일들은 같은 파일이라고 할 수 있다.

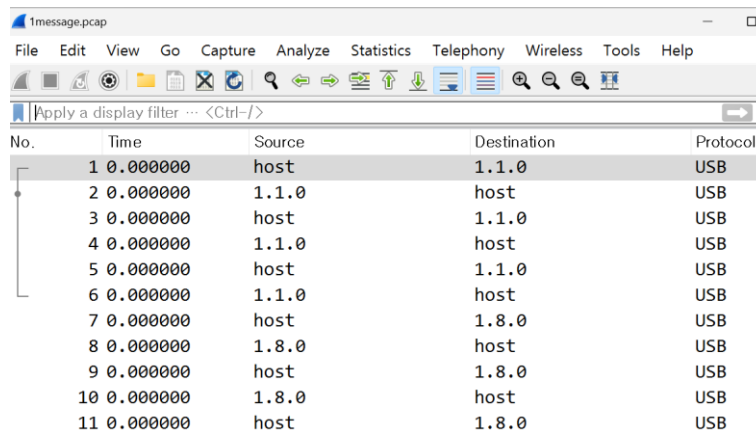
이제 이 message 파일들을 분석하기 위해 살펴보니, [사진 8]과 같이 3개의 파일 모두 'D4 C3 B2 A1'로 시작한다는 것을 알 수 있었다. 이것은 pcap 파일의 시그니처라는 것을 알아냈고, Wireshark를 이용한 분석이 필요할 것이라고 생각하였다.



[사진 8] 1message 파일의 헤더

[WHS-2] .iso

3개의 파일은 모두 Export 한 다음, 확장자를 pcap로 수정하고 Wireshark를 이용하여 열어보았다. 이를 살펴보니, usb를 이용한 통신이라는 것을 확인할 수 있었다.



No.	Time	Source	Destination	Protocol
1	0.000000	host	1.1.0	USB
2	0.000000	1.1.0	host	USB
3	0.000000	host	1.1.0	USB
4	0.000000	1.1.0	host	USB
5	0.000000	host	1.1.0	USB
6	0.000000	1.1.0	host	USB
7	0.000000	host	1.8.0	USB
8	0.000000	1.8.0	host	USB
9	0.000000	host	1.8.0	USB
10	0.000000	1.8.0	host	USB
11	0.000000	host	1.8.0	USB

[사진 9] 1message.pcap 파일

쭉 살펴보니 주소가 1.1.0에서 1.9.0까지 존재했고, 각각 사용한 장치가 다른 것 같았다. 이는 [사진 10]을 통해 확인할 수 있다.

4 0.000000	1.1.0	host	14 0.000000	1.3.0	host	USB
5 0.000000	host	1.1.0	15 0.000000	host	1.3.0	USB
6 0.000000	1.1.0	host				
7 0.000000	host	1.8.0				
			EVICTION DESCRIPTOR bLength: 18 bDescriptorType: 0x01 (DEVICE) bcdUSB: 0x0200 bDeviceClass: Device (0x00) bDeviceSubClass: 0 bDeviceProtocol: 0 (Use class code info from Interface Descriptor) bMaxPacketSize0: 8 idVendor: Lenovo (0x17ef) idProduct: ThinkPad Compact Keyboard with TrackPoint (0x6047)			

[사진 10] 1.1.0은 키보드, 1.3.0은 ThinkPad Compact Keyboard with TrackPoint 사용

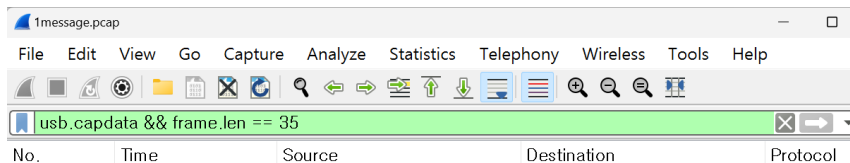
이렇게 쭉 보던 중, 1.5.1을 발견하였는데, USB function 부분이 다른 것들은 USB function이라고 적혀 있는 데에 비해, 여기는 [사진 11]을 통해 interrupt라는 단어가 있는 것을 발견했다. 따라서 차별점이 있다고 생각하여 더 알아보니, 여기서 사용한 장치는 HID 장치임을 알 수 있었다.

65 0.277463	1.5.1	host	US
66 0.277491	host	1.5.1	US
67 0.285411	1.5.1	host	US
IRP USB_D_STATUS: USB_D_STATUS_SUCCESS (0x00000000) USB Function: USB_FUNCTION_BULK_OR_INTERRUPT_TRANSFER (0x0009) > IRP information: 0x01, Direction: PDO -> FDO USB bus id: 1 Device address: 5 > Endpoint: 0x81, Direction: IN USB transfer type: USB_INTERRUPT (0x01) Packet Data Length: 8 [Request in: 62] [Time from request: 0.015954000 seconds] [bInterfaceClass: HID (0x03)] HID Data: 0000fc000000fcff			

[사진 11] 1.5.1의 장치는 HID 장치

[WHS-2] .iso

따라서 HID 데이터를 살펴보기 위해 위에 필터에 'usb.capdata && frame.len == 35'를 입력해보았다. 이것은 usb 패킷이면서 프레임 길이가 35인 패킷을 검색하는 것인데, 길이를 35로 정한 이유는 일반적으로 USB HID 장치에서 전송할 때 사용되는 고정된 크기이기 때문이다. 그러나 검색 결과는 아무것도 나오지 않았다. 그러나 [사진 13]과 같이 앞의 capdata 조건을 제외하고 검색하면 프레임 길이가 35인 패킷들이 전부 나타났다. 그래서 우선 cmd를 이용하여 **'frame.len == 35'** 조건에 해당하는 패킷들의 capdata를 **tshark를 통해 1m_151.txt**라는 텍스트 파일로 추출했다.



[사진 12] capdata와 프레임 길이 조건으로 검색한 결과

No.	Time	Source	Destination	Protocol
31	0.141412	1.5.1	host	USB
33	0.149407	1.5.1	host	USB
35	0.157438	1.5.1	host	USB
37	0.165469	1.5.1	host	USB
39	0.173413	1.5.1	host	USB
41	0.181498	1.5.1	host	USB
43	0.189424	1.5.1	host	USB

[사진 13] 프레임 길이 조건으로만 검색한 결과

```
C:\Users\윤지원\OneDrive\바탕 화면\화이트햇프로젝트>tshark -r "C:\Users\윤지원\OneDrive\바탕 화면\화이트햇프로젝트\1message.pcap" -Y "frame.len == 35" -T fields -e usb.capdata > 1m_151.txt
```

[사진 14] 1m_151.txt 생성 명령어

이후로 막혀서 write-up을 참고한 결과, 1m_151.txt에 출력된 8바이트 데이터가 마우스 패킷 데이터와 유사하여 이를 **파싱하면 마우스가 움직인 경로를 시각화**할 수 있다는 정보를 얻었다. 따라서 파싱을 진행하기 위해 UsbMiceDataHacker 코드를 사용한다고 했는데, 처음에는 이 코드가 이미지 파일 안에 존재하는 줄 알고 열심히 찾아봤는데, 결국 찾지 못했다. 그래서 풀 기미가 안 보이던 중에, 혹시나 해서 깃허브에 검색해봤더니 해당 코드를 발견할 수 있었다. 따라서 <https://github.com/laziok/UsbMiceDataHacker2021> 에 있는 코드를 알맞게 수정하여 파싱하면 될 것 같다고 생각했다.

UsbMiceDataHacker.py 코드는 Wireshark에서 캡처된 USB 마우스 데이터를 분석하고 시각화하는 기능을 가지고 있었다. 따라서 이를 [사진 15]와 같이 수정하고 UsbMiceDataHacker_1.py라는 이름으로 저장하였다.


```
# get data of pcap
.....
command = "tshark -r %s -T fields -e usb.capdata > %s" % (
    pcapFilePath, DataFileName)
print(command)
os.system(command)

# read data
with open(DataFileName, "r") as f:
    for line in f:
        data.append(line[0:-1])

# handle move
.....
with open(pcapFilePath, "r") as f:
    for line in f:
        data.append(line[0:-1])

for i in data:
    Bytes = i.split(":")
    if len(Bytes) == 8:
        horizontal = 1 # -
        vertical = 2 # |
    elif len(Bytes) == 4:
        horizontal = 1 # -
        vertical = 2 # |
    else:
        continue
```

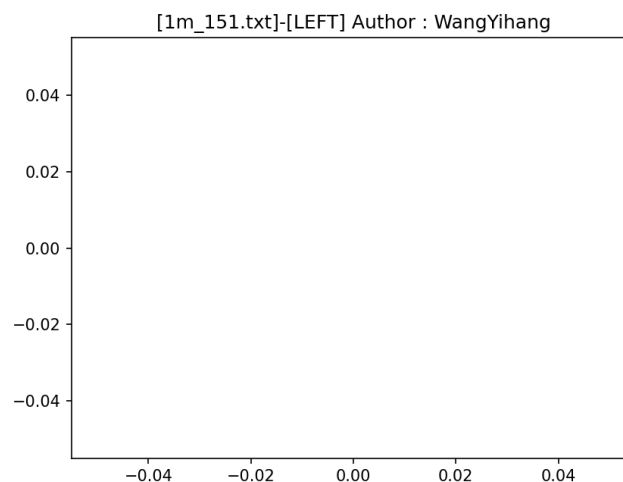
[사진 15] UsbMiceDataHacker_1.py의 일부분

기존의 코드와 달라진 점은 저 주석 처리된 초록색 부분이 tshark를 이용하여 pcapdata를 추출하는 명령인데, 이미 cmd를 통해 1m_151.txt로 생성해 놓았기 때문에 주석 처리를 해 놓았다. 또한 아래 for문에서 가로 세로 길이를 보기 편하도록 조금씩 조정해주었다.

이렇게 수정한 코드를 [사진 16]의 명령어를 이용하여 실행해주었고, [사진 17]과 같은 결과가 나왔다.

```
C:\Users\윤지원\OneDrive\바탕 화면\화이트햇\프로젝트>python UsbMiceDataHacker_1.py 1m_151.txt LEFT
'rm' 은(는) 내부 또는 외부 명령, 실행할 수 있는 프로그램, 또는
배치 파일이 아닙니다.
```

[사진 16] UsbMiceDataHacker_1.py 실행



[사진 17] UsbMiceDataHacker_1.py 실행 결과 (실패)

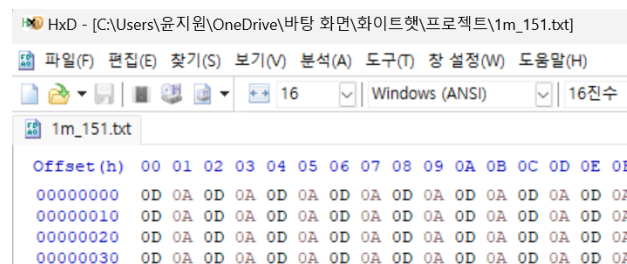
[WHS-2] .iso

시각화 창이 나온 것을 보면 명령어나 코드가 틀린 것은 아닌 것 같아서 무엇이 문제인지 알아보니, 1m_151.txt 파일이 문제였다. 원래 파일 내용에 capdata가 출력되어야 하는데, 보이는 데이터가 없는데 글자 수는 많이 존재하는 것을 확인할 수 있었다.



[사진 18] 1m_151.txt 내용

따라서 텍스트 파일에 문제가 생긴 줄 알고 HxD에도 넣어보았는데, [사진 19]와 같이 0D와 0A만 계속해서 나타나는 것을 볼 수 있었다. 이것은 tshark 명령어가 제대로 실행되지 않았거나, 필터 조건에 맞는 데이터가 없는 경우 발생한 것이라고 한다.



[사진 19] 1m_151.txt를 HxD에 넣은 모습

그러나 중간중간 출력되는 비트들이 있어서 공백들을 없애고 출력되는 값으로만 1m_151.txt를 다시 만들면 될 것이라고 생각했다. 따라서中间的 공백을 없애주는 line.py라는 코드를 만들었다. 이 코드는 파일 1m_151.txt과 동일한 usbdata1.txt을 새로 만든 다음 적용시켰고, 이것의 공백을 없애고 filtered_capdata.txt로 새로 만들어준다.

```
# filter_empty_lines.py
input_file = 'usbdata1.txt'
output_file = 'filtered_capdata.txt'

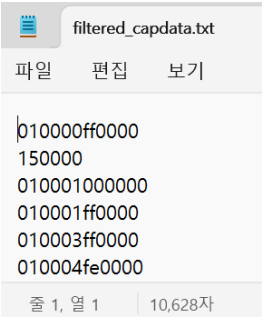
with open(input_file, 'r') as infile, open(output_file, 'w') as outfile:
    for line in infile:
        if line.strip(): # 공백이 아닌 줄만 필터링
            outfile.write(line)
```

[사진 19] line.py

이 코드를 [사진 20]의 명령어로 입력해주었더니 [사진 21]과 같이 공백이 모두 사라지고 값들만 나타나는 것을 볼 수 있었다.

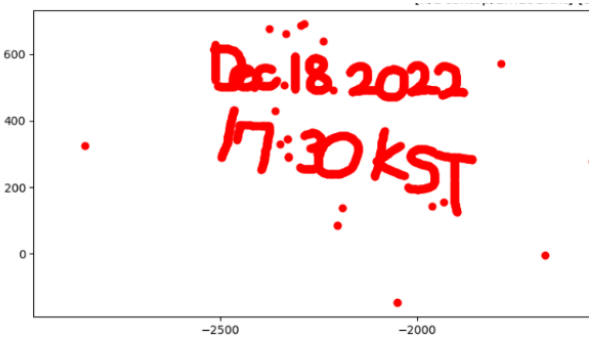
```
C:\Users\윤지원\OneDrive\바탕 화면\화이트햇프로젝트\2022 204>python line.py
```

[사진 20] line.py 실행



[사진 21] filtered_capdata.txt

이렇게 새로 만든 파일을 가지고 [사진 16]의 명령어를 적용하여 입력해주면 다음과 같은 시각화 결과가 출력된다.



[사진 22] 좌클릭 상태로 움직인 경로 시각화

따라서 스파이의 임무 수행일은 **2022년 12월 18일 17:30 KST**이다.

2. 스파이의 임무 장소는 어디인가?

이번에는 2message 파일을 wireshark로 열어보았다.

2message						
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help						
Apply a display filter ... <Ctrl-/>						
No.	Time	Source	Destination	Protocol	Length	Info
31	0.190232	1.4.2	host	USB	42	URB_BULK in
32	0.190370	host	1.4.2	USB	27	URB_BULK in
33	0.192717	1.4.2	host	USB	42	URB_BULK in
34	0.192775	host	1.4.2	USB	27	URB_BULK in
35	0.201435	1.4.2	host	USB	42	URB_BULK in
36	0.201598	host	1.4.2	USB	27	URB_BULK in
37	0.223967	1.4.2	host	USB	42	URB_BULK in
38	0.224105	host	1.4.2	USB	27	URB_BULK in
39	0.226415	1.4.2	host	USB	42	URB_BULK in
40	0.226472	host	1.4.2	USB	27	URB_BULK in
41	0.235258	1.4.2	host	USB	42	URB_BULK in

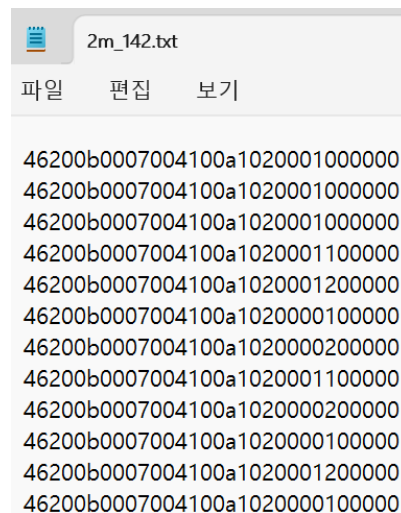
[사진 23] 2message

[WHS-2] .iso

이 파일에서 가장 많은 부분을 차지하고 있는 1.4.2 데이터가 보였고, 장치가 Unknown 형태로 이루어져있다는 것을 알 수 있었다. 따라서 이에 해당하는 패킷들을 [사진 24] 명령어를 통해 2m_142.txt로 추출하였다. 이번에는 한번에 값들이 출력되었다.

```
C:\Users\윤지원\OneDrive\바탕 화면\화이트햇프로젝트\2022_204>tshark -r "C:\Users\윤지원\OneDrive\바탕 화면\화이트햇프로젝트\2022_204\2message" -Y "usb.capdata && frame.len == 42" -T fields -e usb.capdata > 2m_142.txt
```

[사진 24] 2m_142.txt 추출



[사진 25] 2m_142.txt

이 데이터들을 쭉 살펴보면 처음의 46200b0007004100a102까지는 동일하고 그 다음은 조금씩 다르다는 것을 확인할 수 있었다. 이 뒤의 4바이트가 유의미한 부분이라고 생각했기에, 이들이 마우스 값 대응에 중요한 역할을 한다는 사실도 유추할 수 있었다. 따라서 더 자세히 어떤 값들로 이루어져 있는지 알아보았을 때, 4바이트 중 첫번째는 00 또는 01 중 하나를 가지고, 두번째와 세번째는 00~ff의 범위 안의 값을 가지는 것으로 보였다. 마지막은 00 또는 ff 중 하나였다.

이들을 각각 클릭과 가로 움직임, 세로 움직임으로 나누어 대응시키면 될 것이라고 생각하여 UsbMiceDataHacker.py를 이에 알맞게 수정하여 UsbMiceDataHacker_2.py로 만들었다.

```
if Bytes[10] == "01":
    print("[+] Left button.")
    if action == "LEFT":
        # draw point to the image panel
        X.append(mousePositionX)
        Y.append(-mousePositionY)
elif Bytes[10] == "02":
    print("[+] Right Button.")
    if action == "RIGHT":
        # draw point to the image panel
        X.append(mousePositionX)
        Y.append(-mousePositionY)
elif Bytes[10] == "00":
    print("[+] Move.")
    if action == "MOVE":
        # draw point to the image panel
        X.append(mousePositionX)
        Y.append(-mousePositionY)
```

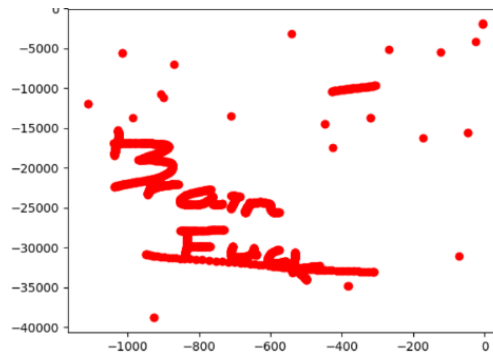
[사진 26] UsbMiceDataHacker_2.py 중 일부분

[WHS-2] .iso

이렇게 만든 코드를 이용하여 [사진 27]의 명령어를 실행시켜주면 [사진 28]과 같이 시각화 결과로 **Brain Fuck**이라는 글자를 볼 수 있다.

```
C:\Users\윤지원\OneDrive\바탕 화면\화이트햇프로젝트\2022 204>python UsbMiceDataHacker_2.py 2m_142.txt LEFT
```

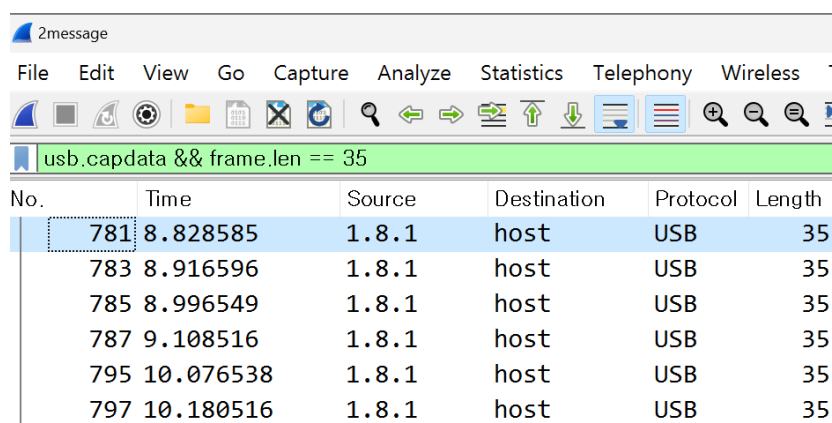
[사진 27] 시각화 명령어



[사진 28] 2m_142.txt 시각화 결과

그러나 이것만으로는 임무 장소가 나오지 않기 때문에 2message를 살펴보다가, 1message에서 했던 것과 같이 'usb.capdata && frame.len == 35'로 검색해보았는데 [사진 29]와 같이 데이터가 존재하는 것을 알 수 있었다. 조사를 통해 이것은 usb로 연결된 키보드 데이터와 유사하다는 것을 알 수 있었고, UsbKeyboardDataHacker.py를 이용할 수 있다는 사실도 알 수 있었다.

따라서 우선적으로 [사진 30]의 명령어를 이용하여 2m_181.txt 파일을 추출해주었다. 그 다음 <https://github.com/WangYihang/UsbKeyboardDataHacker/blob/master/UsbKeyboardDataHacker.py> 이 링크에서 코드 원본을 얻고, 이를 조금 수정하여 UsbKeyboardDataHacker_1.py로 만들어주었다.



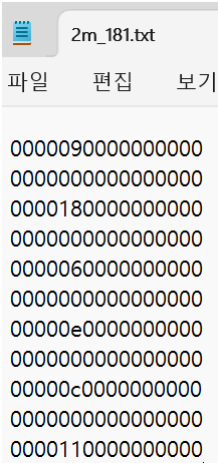
No.	Time	Source	Destination	Protocol	Length
781	8.828585	1.8.1	host	USB	35
783	8.916596	1.8.1	host	USB	35
785	8.996549	1.8.1	host	USB	35
787	9.108516	1.8.1	host	USB	35
795	10.076538	1.8.1	host	USB	35
797	10.180516	1.8.1	host	USB	35

[사진 29] 2message에 존재하는 키보드 데이터

```
C:\Users\윤지원\OneDrive\바탕 화면\화이트햇프로젝트\2022 204>tshark -r "C:\Users\윤지원\OneDrive\바탕 화면\화이트햇프로젝트\2022 204\2message" -Y "usb.capdata && frame.len == 35" -T fields -e usb.capdata > 2m_181.txt
```

[사진 30] 2m_181.txt 추출

[WHS-2] .iso



[사진 31] 2m_181.txt

```
# get argv
pcapFilePath = sys.argv[1]

# get data of pcap
# os.system("tshark -r %s -T fields -e usb.capdata 'usb.data_len == 8'

# read data
with open(pcapFilePath, "r") as f:
    for line in f:
        presses.append(line[0:-1])
```

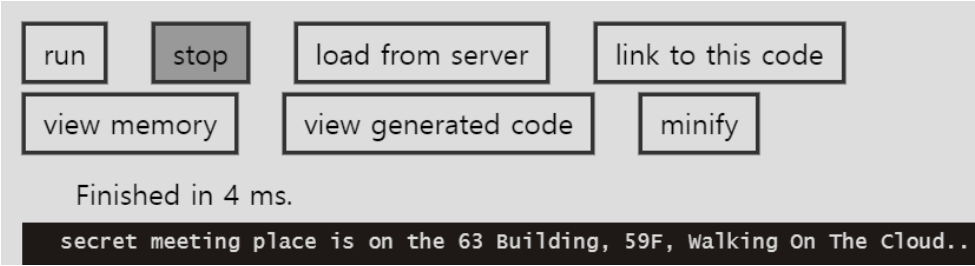
[사진 32] UsbKeyboardDataHacker_1.py 일부분

이를 [사진 33]의 명령어를 이용하여 입력해주면 다음과 같은 문자열을 확인할 수 있다.



[사진 33] UsbKeyboardDataHacker_1.py 실행

이걸 가지고 어떻게 하라는 것인지 의문이 들어 우선적으로 Brain Fuck에 대해서 조사해보았더니, [사진 33]에서 출력된 문자열과 같은 형태의 프로그래밍 언어라는 것을 알 수 있었다. 따라서 이를 번역할 수 있는 사이트가 있을 것이라고 생각하여 알아보니 <https://copy.sh/brainfuck/> 라는 사이트를 발견할 수 있었고, 여기에 해당 문자열을 넣은 결과는 [사진 34]와 같다.

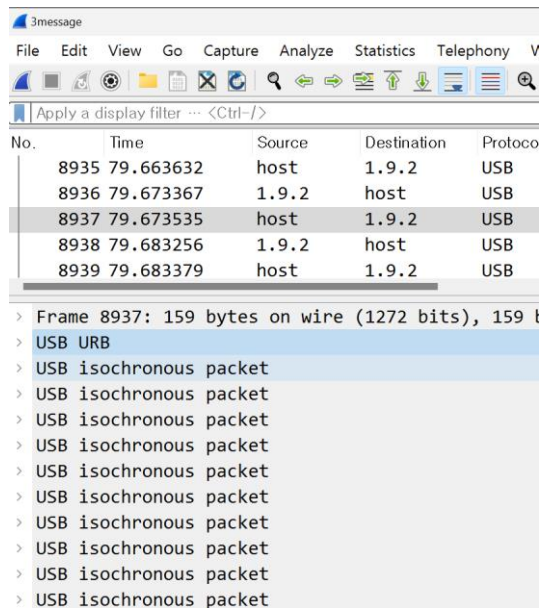


[사진 34] brainfuck 해석 결과

[WHS-2] .iso

3. 누가 스파이의 타겟인가?

이번에는 3message를 wireshark로 열어보았다.



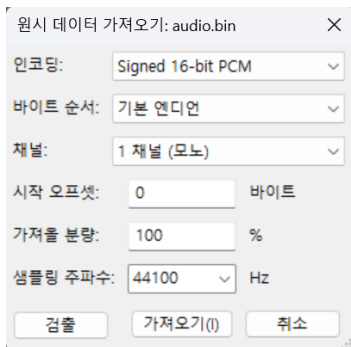
[사진 35] 3message

살펴보면 앞의 두 message와 달리 1.9.2라는 id가 많았고, 이를 살펴보니 [사진 35]와 같이 **USB isochronous**라는 것이 있었다. 조사해보니, 이는 오디오 장치와 통신하는 데이터라고 하여 이전과 같이 텍스트 파일이 아니라 오디오 파일을 추출하면 될 것이라고 생각했다. 따라서 audio.bin 파일을 추출할 수 있는 tshark 명령어를 [사진 36]과 같이 실행하였다.

```
C:\Users\윤지원\OneDrive\바탕 화면\화이트햇프로젝트\2022 204>tshark -r "C:\Users\윤지원\OneDrive\바탕 화면\화이트햇프로젝트\2022 204\3message" -Y "usb.iso.data" -T fields -e usb.iso.data | tr -d "\n" | tr -d "," | xxd -r -ps > audio.bin
```

[사진 36] audio.bin 파일 추출

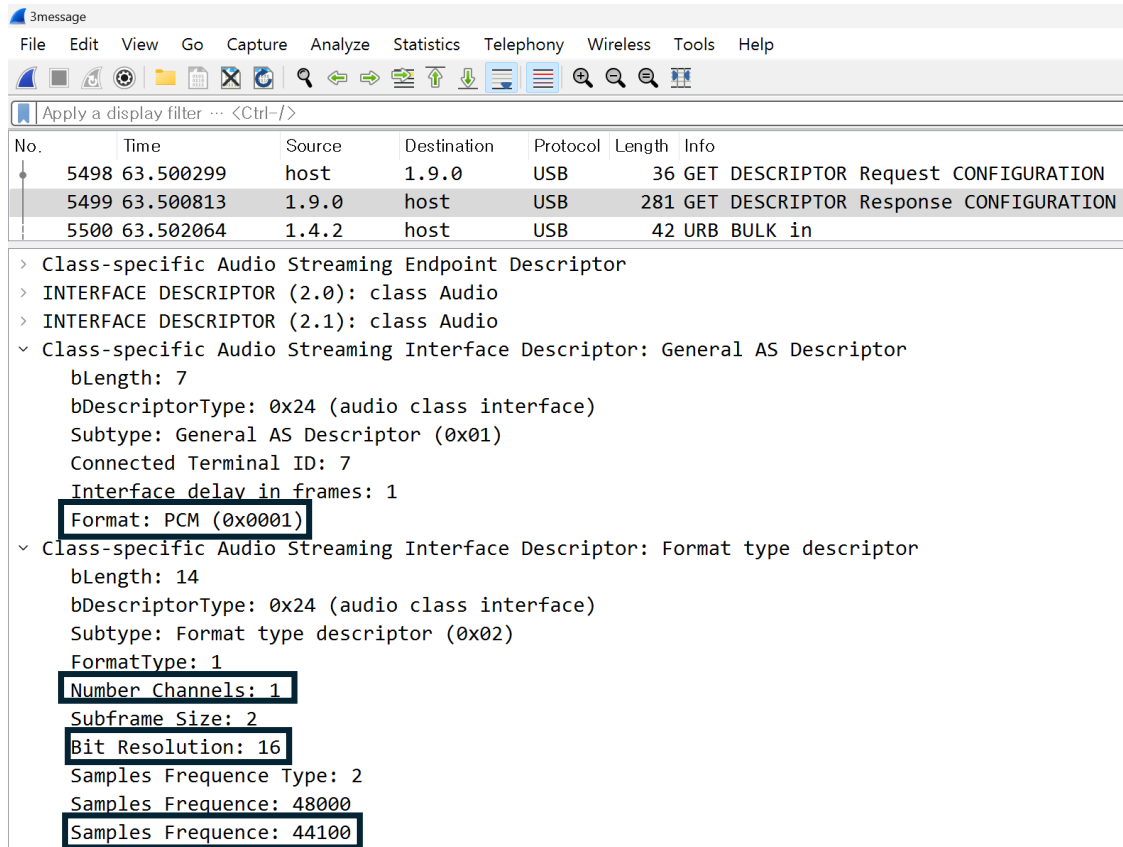
오디오 파일이니까 이에 대한 정보를 알 수 있는 오디오 포렌식에서 종종 사용하는 audacity를 이용하면 좋을 것이라고 생각하였다. 이때 일반적인 오디오 파일 형태가 아니기 때문에 raw 데이터 가져오기(원시 데이터 가져오기)를 선택하여 파일을 가져와야 한다.



[사진 37] audacity로 audio.bin을 열었을 때 나타나는 팝업

[WHS-2] .iso

파일을 열었을 때 [사진 37]이 자동으로 나타나는데, 이 파일에 대한 기본 정보일 것이라고 생각했다. 그리고 이 정보가 들어있는 패킷을 3message에서 찾을 수 있을 것이라고 생각하고 패킷들을 살펴보니 [사진 38]에서 발견할 수 있었다.

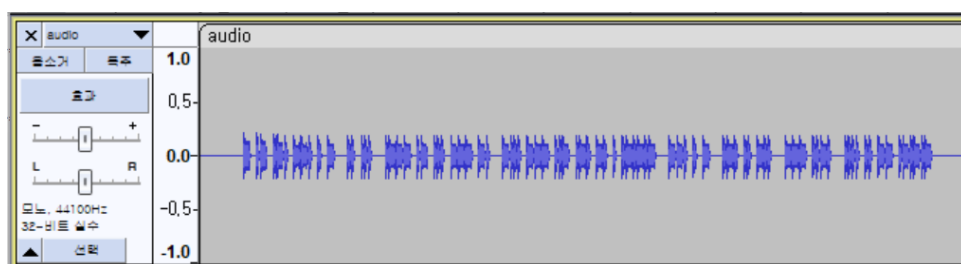


No.	Time	Source	Destination	Protocol	Length	Info
5498	63.500299	host	1.9.0	USB	36	GET_DESCRIPTOR Request CONFIGURATION
5499	63.500813	1.9.0	host	USB	281	GET_DESCRIPTOR Response CONFIGURATION
5500	63.502064	1.4.2	host	USB	42	URB BULK in

- > Class-specific Audio Streaming Endpoint Descriptor
 - > INTERFACE_DESCRIPTOR (2.0): class Audio
 - > INTERFACE_DESCRIPTOR (2.1): class Audio
 - > Class-specific Audio Streaming Interface Descriptor: General AS Descriptor
 - bLength: 7
 - bDescriptorType: 0x24 (audio class interface)
 - Subtype: General AS Descriptor (0x01)
 - Connected Terminal ID: 7
 - Interface delay in frames: 1
 - Format: PCM (0x0001)
- > Class-specific Audio Streaming Interface Descriptor: Format type descriptor
 - bLength: 14
 - bDescriptorType: 0x24 (audio class interface)
 - Subtype: Format type descriptor (0x02)
 - FormatType: 1
 - Number Channels: 1
 - Subframe Size: 2
 - Bit Resolution: 16
 - Samples Frequency Type: 2
 - Samples Frequency: 48000
 - Samples Frequency: 44100

[사진 38] audio.bin 정보

따라서 인코딩은 16비트 PCM, 채널은 1채널, 샘플 주파수는 44100로 동일하다는 사실을 알 수 있다.



[사진 39] audio.bin

audio.bin을 재생해보니 모스부호였는데, 이것도 따로 변환해주는 사이트를 찾았다. audio.bin 파일을 mp3 파일로 내보내기 하여 <https://morsecode.world/international/decoder/audio-decoder-adaptive.html> 이 사이트에 넣고 번역을 돌려보면 다음과 같은 결과가 나왔다.


Alphabet to decode into


Latin


All these alphabets can be sent in Morse using
(and includes accented characters and prosig)


Use the microphone:

Or analyse an audio file containin

Listen 

Stop 

Upload 

Play 

Filename: "audio.mp3"

TARGET IS JASON BOURNE MET NID OF HIM.

[사진 40] 모스 부호 해석

따라서 타겟은 **JASON BOURNE**이다.

5. Flag

- 1번 : 2022년 12월 18일 17:30 KST
- 2번 : 63 Building, 59F, Walking On The Cloud
- 3번 : JASON BOURNE

6. 별도 첨부

7. Reference

- [URL]