304 – The audio must be erased to be heard.

Team Information

Team Name : ForensicGPT

Team Member : Eungchang Lee, Donghyun HA, Hyunwoo Shin, Jinhyeok Son

Email Address : forensicgpt@googlegroups.com

Instructions

Description A vehicle equipped with a dash cam camera was in an accident, and the last recording time zone file was recorded abnormally. Normal files recorded in the previous time period are recorded with video data and audio data in an FTYP container with an MP4 extension. However, only the black screen is recorded for the video data, and the audio file is recorded normally. Recover audio files of MP4 files recorded due to abnormal termination. Since the mounted dash cam uses a file system with a bank structure, various time zone data remain in the abnormally terminated file due to the file slack phenomenon.

Target	Hash (MD5)
REC_1970_01_01_00_23_05_F.MP4	82395B3B85E5AF23AEEE50DBB6AE2072

Questions

1) Submit all song titles other than audio files that play from 0 to 20 seconds recorded in the target file. (300 points)
   A. First song name (150 points)
   B. Second song name (150 points)

Teams must:

- Develop and document the step-by-step approach used to solve this

problem to allow another examiner to replicate team actions and results.

- Specify all tools used in deriving the conclusion(s).

Tools used:

Name:	Hashtab	**Publisher:**	Implbits Software
Version:	6.0.0		
URL:	https://implbits.com		

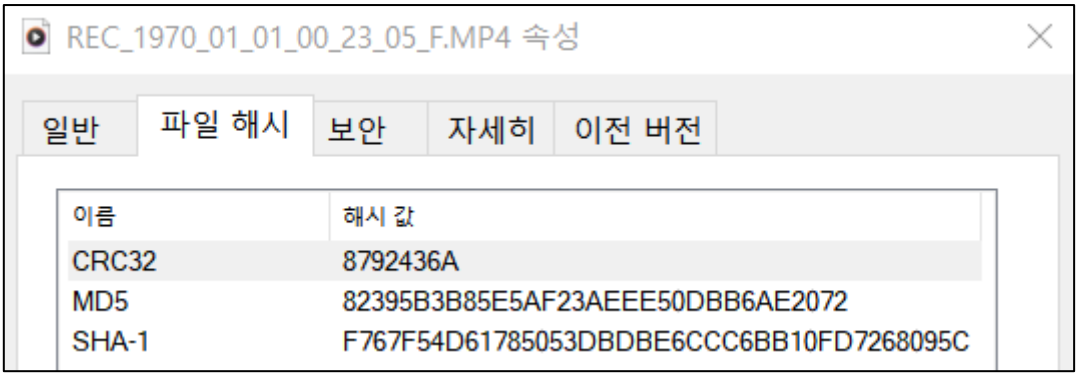
Name:	010 editor	**Publisher:**	sweetscape software
Version:	11.0.1		
URL:	https://www.sweetscape.com/010editor/		

Name:	Visual Studio Code	**Publisher:**	Microsoft
Version:	1.79.2		
URL:	https://code.visualstudio.com/download		

Name:	HxD	**Publisher:**	Maël Hörz
Version:	2.5.0.0		
URL:	https://www.mh-nexus.de		

Name:	Mp4 Explorer	**Publisher:**	CM Streaming tech
Version:	1.0.1.41163		
URL:	https://mp4-explorer.apponic.com/		

Step-by-step methodology:

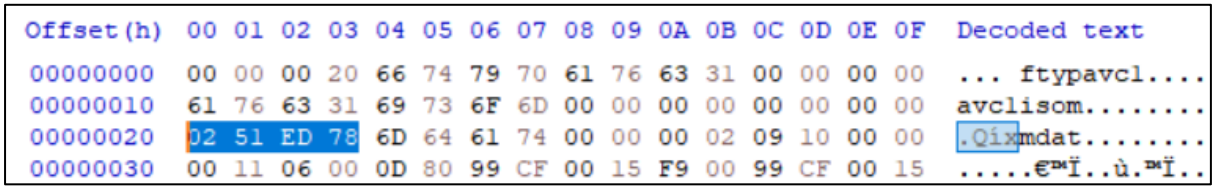


[그림 1] 파일 해시 값 확인

분석에 앞서 주어진 파일에 대한 해시 값을 산출하여, md5 해시 값이 일치함을 확인하였습니다.

1) Submit all song titles other than audio files that play from 0 to 20 seconds recorded in the target file. (300 points)

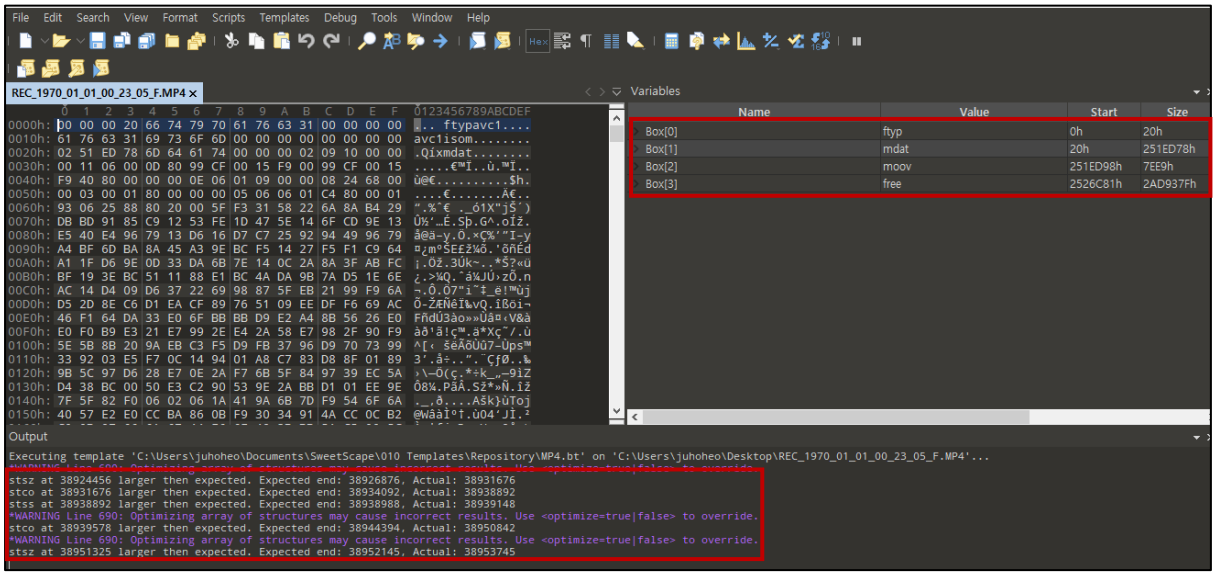
A. First song name (150 points)



Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	00	00	00	20	66	74	79	70	61	76	63	31	00	00	00	00	... ftypavc1....
00000010	61	76	63	31	69	73	6F	6D	00	00	00	00	00	00	00	00	avc1isom........
00000020	02	51	ED	78	6D	64	61	74	00	00	00	02	09	10	00	00	.Qíxmdat........
00000030	00	11	06	00	0D	80	99	CF	00	15	F9	00	99	CF	00	15	.....€™Ï..ù.™Ï..

[그림 2] mdat size 수정

앞서 101번 문제에서 수행해주었던 것 과 같이, 원본 파일에 대한 mdat 영역의 size를 데이터 형식에 맞도록 수행해주었습니다.



[그림 3] 불일치 값 확인

이후, 010 editor의 Template 기능을 통해 파일의 구조를 살펴보게 되면, 기존에 인식되지 않던 moov와 free 영역이 인식되고, stsz, stco, stss 등의 atom 의 size가 실제로 차지하고 있는 데이터의 크기와 불일치 함을 볼 수 있었습니다. 이는 본문에서 설명된 내용 중 사고를 당한 차량의 블랙박스임을 통해, 녹화 도중 사고가 발생하여 entry/sample count의 업데이트가 누락된 것으로 추론할 수 있습니다. 추가적인 분석을 위해 이를 할당된 크기에 맞도록 count를 수정하였습니다.

[표 1] 불일치 영역 수정 값

영역	수정 값
offset 38924456 stsz의 sample_count	0x0708
offset 38931676 stco의 entry_count	0x0708
offset 38938892 stss의 entry_count	0x003c

offset 38939578 stco의 entry_count	0x0AFC
offset 38951325 stsz의 sample_count	0x0258

이후 문제에서 요구한 기존의 오디오에서 들을 수 있는 노래가 아닌, 추가적으로 찾을 수 있는 노래 두 곡을 요구하였고, 위에 언급했던 것 과 같이 사고가 발생한 블랙박스의 데이터로서 MDAT 영역에는 기재되었으나 ATOM에 등록이 되어있지 않았을 경우라 생각되어 해당 방향성으로 진행하였습니다.

다음 코드를 통해 MDAT 데이터 중, 존재하는 3개의 TRAK에서 참조하고 있는 데이터를 제외하고 미사용 되고 있는 MDAT 영역을 추출하는 코드는 다음과 같습니다.

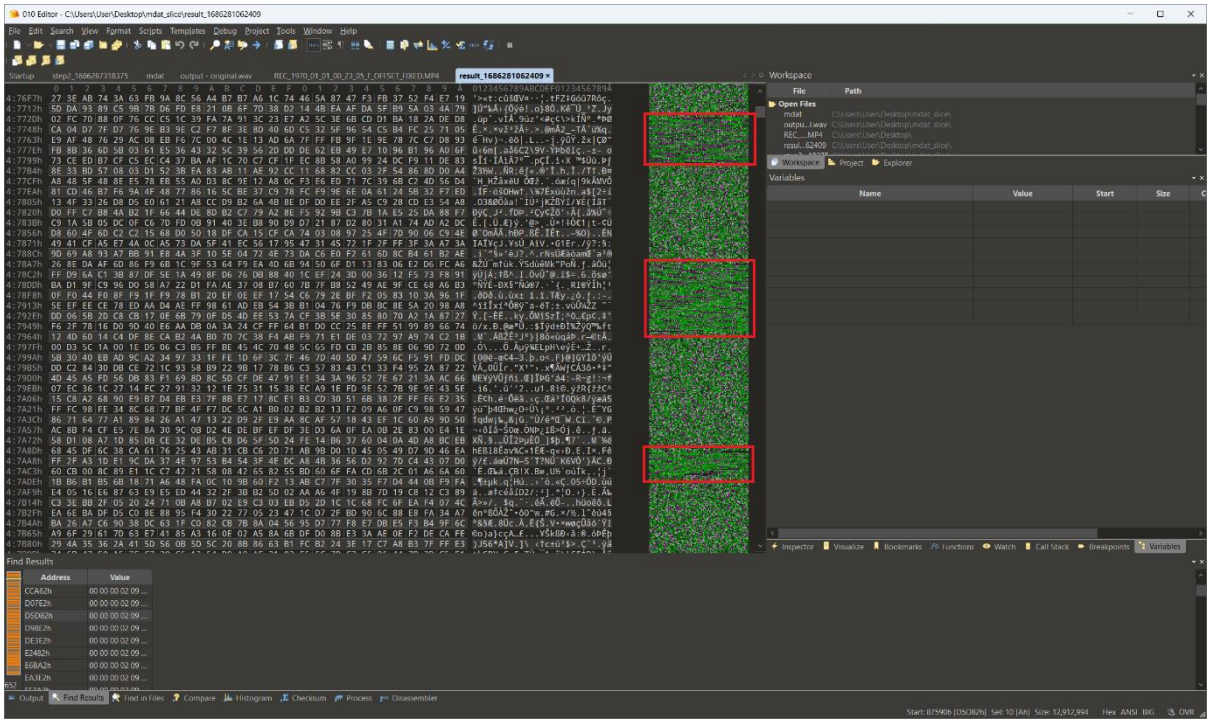
video/subtitle에 대한 stco/stsz는 010 Editor에서 Template을 통해 추출된 데이터를 사용하였으며, audio에 대한 chunk의 오프셋은 stco, chunk의 사이즈는 존재하지 않아 알고 있는 video/subtitle에 대한 데이터를 mdat 영역에서 마스킹 한 후 2048으로 추론하여 진행하였습니다.

[표 2] mdat 영역에서 헤더정보가 누락된 데이터 추출

```
import audio_stco from "./audio_stco";
import video_stco from "./video_stco";
import video_stsz from "./video_stsz";
import subtitle_stco from "./subtitle_stco";
import subtitle_stsz from "./subtitle_stsz";
import * as fs from "fs";

const audio_chunk_size = 2048;

const ftyp = [ 0x00, 0x00, 0x00, 0x20, 0x66, 0x74, 0x79, 0x70, 0x61, 0x76, 0x63, 0x31, 0x00, 0x00, 0x00, 0x00, 0x61, 0x76, 0x63, 0x31, 0x69, 0x73, 0x6F, 0x6D, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, ];

const ftypLength = ftyp.length;

const mdat = fs.readFileSync("./mdat");

const result = fs.createWriteStream(`./result_${Date.now()}`,{flags:'w'});

const audio_stco_stsz: [number, number][] = audio_stco.map(v => [v - ftypLength, audio_chunk_size]);
const video_stco_stsz: [number, number][] = video_stsz.map((v, i) => [video_stco[i] - ftypLength, v]);
const subtitle_stco_stsz: [number, number][] = subtitle_stsz.map((v, i) => [subtitle_stco[i] - ftypLength, v]);

const all_stco_stsz = [...audio_stco_stsz, ...video_stco_stsz, ...subtitle_stco_stsz].sort((a, b) => a[0] -
```

```
b[0]);

let currIndex = 0;

while(all_stco_stsz.length > 0) {
    const st = all_stco_stsz.shift()!;

    // Skip
    if (currIndex === st[0]) {
        currIndex += st[1];
        continue;
    }

    const next_st: [number, number] | undefined = all_stco_stsz[0];

    if (next_st) {
        result.write(mdat.slice(currIndex, next_st[0]));
        currIndex = next_st[0];
    } else {
        result.write(mdat.slice(currIndex));
        break;
    }
}

result.close(() => console.log("Fin"));
```

추출된 데이터 영역을 분석해 보니, 비산되어 있는 오디오 청크들을 확인할 수 있었으나, 오디오 청크들은 특별한 시그니처가 없기에 그 외의 영역을 제거하는 방향으로 가닥을 잡았습니다.



[그림 4] 추출된 mdat 영역 중 존재하는 오디오 청크

[그림 5] 비 오디오 영역 데이터 중 일부 파트

비 오디오 영역들을 확인해본 결과, NALU 구조의 데이터로서, 오디오와 같이 사고로 인해 데이터만 기재된 비디오 데이터임을 확인하였습니다.

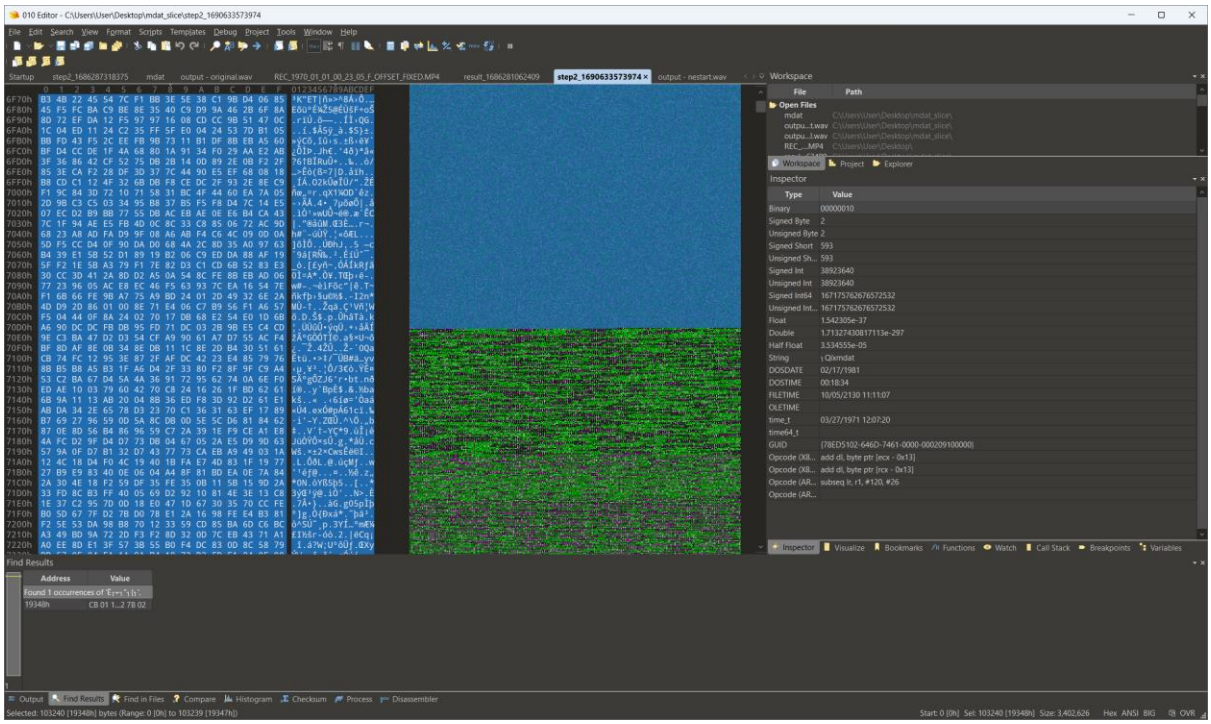
이에 해당 데이터 내부에 존재하는 NALU 청크들을 제거하는 코드를 작성하여 사운드 영역만을 추출하였습니다.

[표 3] 사운드 청크 추출 과정

```
import fs from "fs";

const mdat = fs.readFileSync("./result_1686281062409");
const chunkRanges: [number, number][] = [];
let lastFoundOffset;

lastFoundOffset = 0;
// Step 1
while(true) {
    const chunkStart =
mdat.indexOf('\x00\x00\x00\x02\x09\x30\x00\x00\x00\x0E\x06', lastFoundOffset);
    if (chunkStart < 0)
        break;

    const dataLengthOffset = chunkStart + 0x18;
    const dataLength = mdat.readIntBE(dataLengthOffset, 4);
    const dataStartOffset = dataLengthOffset + 4;
    const dataEndOffset = dataStartOffset + dataLength;

    chunkRanges.push([chunkStart, dataEndOffset]);

    lastFoundOffset = dataEndOffset;
}

lastFoundOffset = 0;
// Step 2
while(true) {
    const chunkStart =
mdat.indexOf('\x00\x00\x00\x02\x09\x10\x00\x00\x00\x13\x06', lastFoundOffset);
    if (chunkStart < 0)
        break;

    const dataLengthOffset = chunkStart + 0x38;
    const dataLength = mdat.readIntBE(dataLengthOffset, 4);
    const dataStartOffset = dataLengthOffset + 4;
    const dataEndOffset = dataStartOffset + dataLength;

    chunkRanges.push([chunkStart, dataEndOffset]);
```

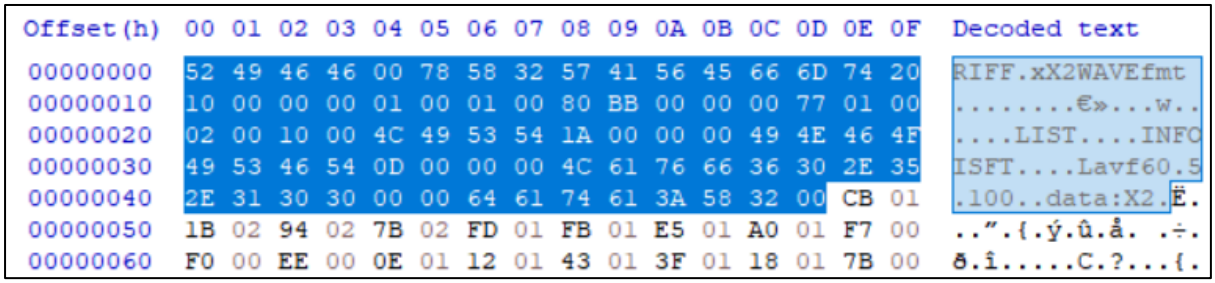
```
    lastFoundOffset = dataEndOffset;
}

chunkRanges.sort((a, b) => a[0] - b[0]);



const result = fs.createWriteStream(`./step2_${Date.now()}`,{flags:'w'});

let currCursor = 0;
for (let chunkRange of chunkRanges) {
    result.write(mdat.subarray(currCursor, chunkRange[0]));
    currCursor = chunkRange[1];
}

result.write(mdat.subarray(currCursor));

result.close();
```

이후 가공된 데이터를 확인해 본 결과 00 00 00 02 09 10 00 00 00 11 06 00 값에 대한 NALU Case가 남아있었으나, 해당 Signature에 대한 Data Length 영역을 찾을 수 없어 임의로 첫 사운드 청크 영역까지 제거하였습니다.



[그림 6] 제거에서 누락된 추가 제거 데이터 영역

모든 과정을 통해 생성된 최종 파일은 RAW_SOUND구조로서 임의의 WAV 헤더를 추가하여 Player에서 실행 가능하도록 만들었습니다.



Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	52	49	46	46	00	78	58	32	57	41	56	45	66	6D	74	20	RIFF.xX2WAVEfmt
00000010	10	00	00	00	01	00	01	00	80	BB	00	00	00	77	01	00	........€»...w..
00000020	02	00	10	00	4C	49	53	54	1A	00	00	00	49	4E	46	4F	....LIST....INFO
00000030	49	53	46	54	0D	00	00	00	4C	61	76	66	36	30	2E	35	ISFT....Lavf60.5
00000040	2E	31	30	30	00	00	64	61	74	61	3A	58	32	00	CB	01	.100..data:X2.Ë.
00000050	1B	02	94	02	7B	02	FD	01	FB	01	E5	01	A0	01	F7	00	..”.{.ý.û.å. .÷.
00000060	F0	00	EE	00	0E	01	12	01	43	01	3F	01	18	01	7B	00	ð.î.....C.?...{.

[그림 7] wav 헤더 추가

그 후, 34초에 해당하는 output – nestart.wav 파일을 실행시키게 되면, 0초에서 20초간 진행되는 노래의 제목은 다음과 같습니다.

[표 4] First song name

output – nestart.wav 파일 재생 구간	노래 제목
0 ~ 20초	**R.Schumann, Träumerei** **(슈만, 트로이메라이)**

B. Second song name (150 points)

두번째 노래는 동일한 파일에서 약 29초부터 마지막 34초까지 재생되는 노래로 제목은 다음과 같습니다.

[표 5] Second song name

output – nestart.wav 파일 재생 구간	노래 제목
29초 ~ 34초	**Satie, Gymnopedie No.1** **(사티, 짐노페디 1번)**