251 – Time is a Masterpiece

Team Information

Team Name : LuckyVicky

Team Member : Eungchang Lee, Hyun Yi, Juho Heo, Dongkyu Lee

Email Address : dfc_luckyvicky@googlegroups.com

Instructions

Description In digital forensics, integrity is paramount. Digital video recorder (DVR) video is a critical piece of evidence, but as sophisticated video editing technology develops, techniques are being adopted to maintain its integrity. One of them is the regularity of the time information between frames, which can be used to detect forgery such as frame skipping or changes in playback speed. However, if the time information used by the system is simple, it can be used for forgery, so manufacturers have their own proprietary time information format. In this problem, you need to extract the time information from the additional information between frames, analyze the form of the time information, and identify the specific time information. (Hint: Time information starts on January 1, 2000 at 00:00:00, The time standard is UTC+0)

Target	Hash (MD5)
Problem.bin	d39139828121885ea04c0ce015e53481

Questions

1. Submit the name of the painting recorded on November 4, 2016 at 15:23:21 [hour:minute:second] (150 points)
2. Submit the time information of November 4, 2063, 15:23:21

[hour:minute:second] as a hex value(big endian) (100 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:	HxD	Publisher:	Maël Hörz
Version:	2.5.0.0		
URL:	https://www.mh-nexus.de		

Name:	kmplayer	Publisher:	PandoraTV
Version:	4.2.2.68		
URL:	https://www.kmplayer.com/kr/home		

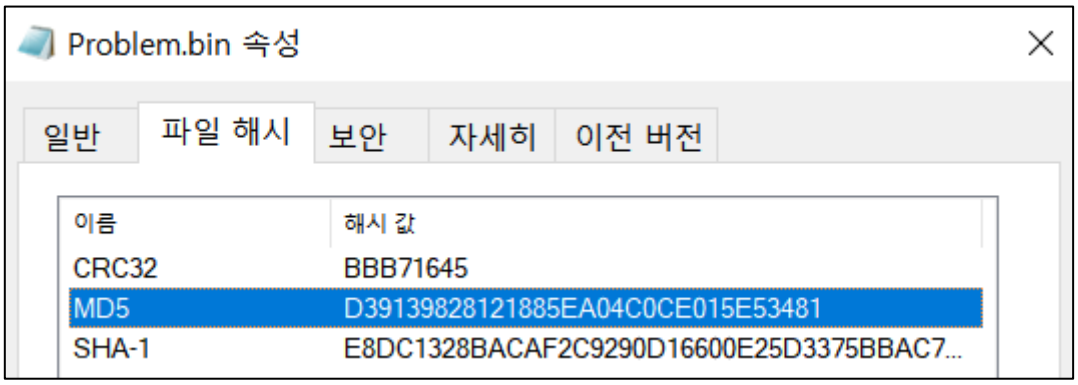
Name:	DCode	Publisher:	Digital Detective
Version:	5.6		
URL:	https://www.digital-detective.net/dcode/		

Name:	Mediainfo	Publisher:	MediaArea.net
Version:	23.04		
URL:	https://mediaarea.net/en/MediaInfo		

Name:	ffplay	Publisher:	FFmpeg developers
Version:	6.0		
URL:	https://www.ffmpeg.org/		

Name:	ffmpeg	Publisher:	FFmpeg developers
Version:	6.0		
URL:	https://www.ffmpeg.org/		

Step-by-step methodology:

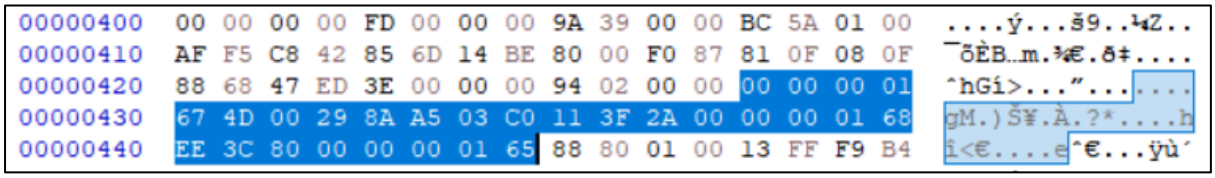


[그림 1] Problem.bin md5 hash 값 확인

주어진 target file인 Problem.bin의 md5 hash가 일치함을 확인하였습니다.

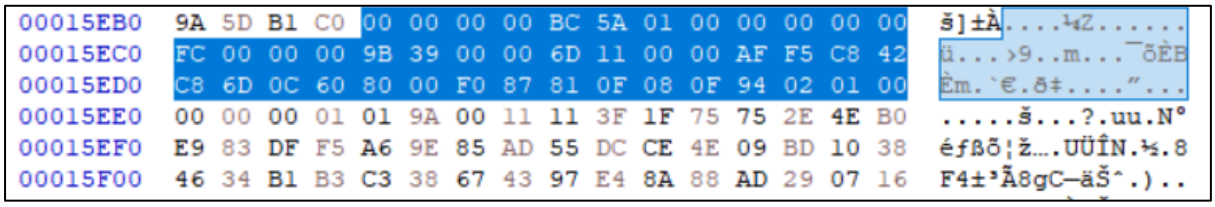
1. Submit the name of the painting recorded on November 4, 2016 at 15:23:21 [hour:minute:second] (150 points)

주어진 Problem.bin에서 시간 정보를 알아내기 위해 여러 데이터를 분석해 보았습니다.

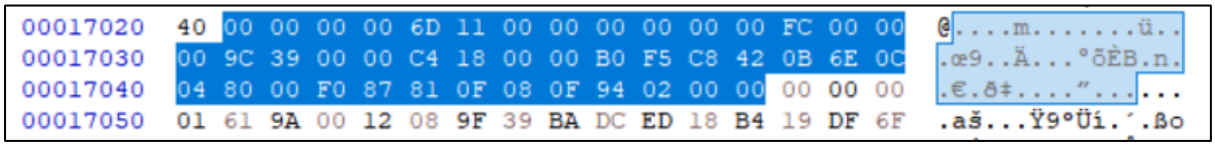


[그림 2] sps, pps, IDR frame 확인

먼저 start prefix 와 NAL Type으로 구성된 NAL Unit이 확인되며, sps와 pps, 그리고 I-Frame 까지 존재하는 것을 알 수 있습니다. 그리고 그 전에, 아래 그림 3과 같이 0x400 offset부터 0x42B까지 0x2C만큼의 특정 데이터가 스트림에 추가로 존재하는 것을 알 수 있습니다.

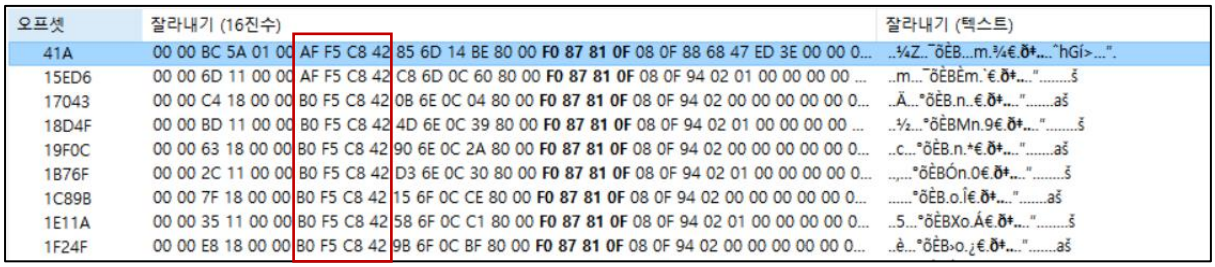


[그림 3] IDR-frame 뒤 0x2C만큼의 data 존재



[그림 4] non-IDR frame 뒤 0x2C만큼 data 존재

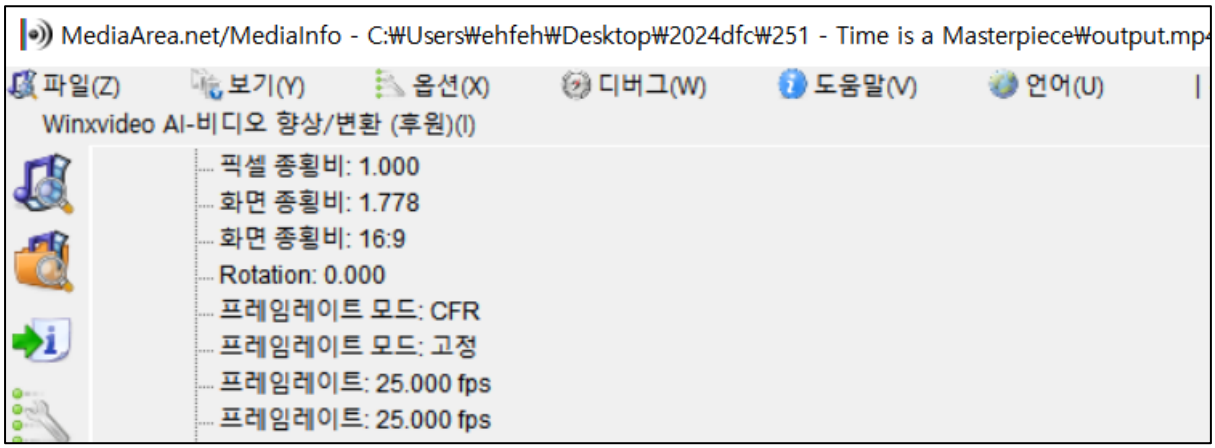
그리고, 이러한 0x2C 데이터는 그 뒤로 IDR-frame과 non-IDR frame의 마지막 부분에 각각 존재했습니다. 여기서 초반 0x08 offset부터 0x0F offset 사이에 있는 0xAF, 0xF5, 0xC8, 0x42, 0x12, 0xF6, 0xC8, 0x42 라는 값과, 반복되는 0x2C data 내 존재하는 비슷한 hex값인 0xC8, 0x42 위주의 데이터 간의 관계를 유심히 살펴보았습니다.



오프셋	잘라내기 (16진수)	잘라내기 (텍스트)
41A	00 00 BC 5A 01 00 **AF F5 C8 42** 85 6D 14 BE 80 00 **F0 87 81 0F** 08 0F 88 68 47 ED 3E 00 00 0...	..¼Z..¯õÈB...m.¾€.ð‡...ˆhGí>...".
15ED6	00 00 6D 11 00 00 **AF F5 C8 42** C8 6D 0C 60 80 00 **F0 87 81 0F** 08 0F 94 02 01 00 00 00 00 ...	..m...¯õÈBÈm.`€.ð‡..."".....š
17043	00 00 C4 18 00 00 **B0 F5 C8 42** 0B 6E 0C 04 80 00 **F0 87 81 0F** 08 0F 94 02 00 00 00 00 00 0...	..Ä...°õÈB.n..€.ð‡..."".....aš
18D4F	00 00 BD 11 00 00 **B0 F5 C8 42** 4D 6E 0C 39 80 00 **F0 87 81 0F** 08 0F 94 02 01 00 00 00 00 ...	..½...°õÈBMn.9€.ð‡..."".....š
19F0C	00 00 63 18 00 00 **B0 F5 C8 42** 90 6E 0C 2A 80 00 **F0 87 81 0F** 08 0F 94 02 00 00 00 00 00 0...	..c...°õÈB.n.*€.ð‡..."".....aš
1B76F	00 00 2C 11 00 00 **B0 F5 C8 42** D3 6E 0C 30 80 00 **F0 87 81 0F** 08 0F 94 02 01 00 00 00 00 0...	..,...°õÈBÓn.0€.ð‡..."".....š
1C89B	00 00 7F 18 00 00 **B0 F5 C8 42** 15 6F 0C CE 80 00 **F0 87 81 0F** 08 0F 94 02 00 00 00 00 00 0...	.....°õÈB.o.Î€.ð‡..."".....aš
1E11A	00 00 35 11 00 00 **B0 F5 C8 42** 58 6F 0C C1 80 00 **F0 87 81 0F** 08 0F 94 02 01 00 00 00 00 0...	..5...°õÈBXo.Á€.ð‡..."".....š
1F24F	00 00 E8 18 00 00 **B0 F5 C8 42** 9B 6F 0C BF 80 00 **F0 87 81 0F** 08 0F 94 02 00 00 00 00 00 0...	..è...°õÈB›o.¿€.ð‡..."".....aš

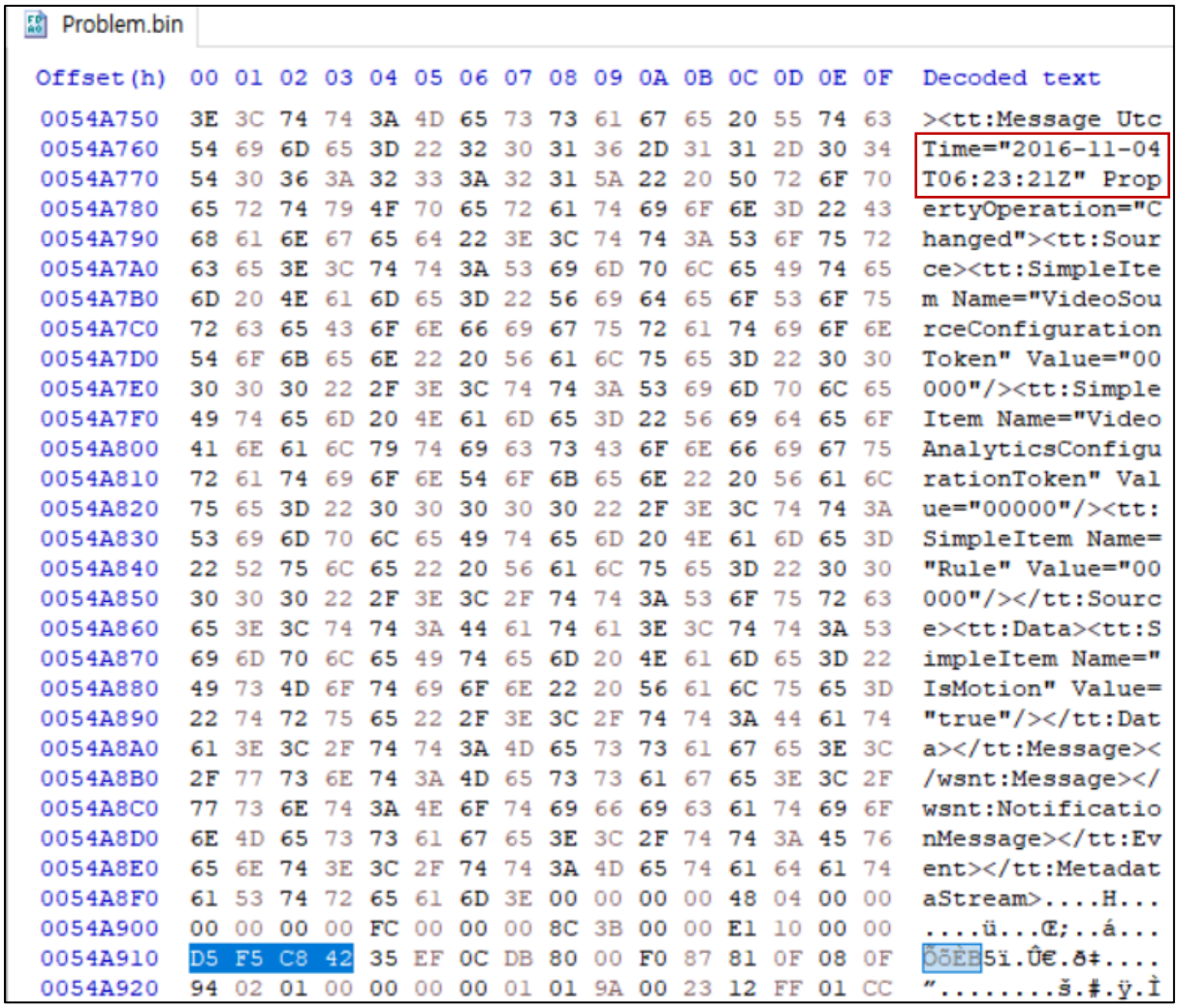
[그림 5] 반복되며 1씩 올라가는 값 확인

또한, 해당 값은 little-endian으로 1씩 증가하는 값이었으며, 조금씩 반복되는 값이었습니다. 그리고 이 값들은 총 1365개가 존재했는데, 아래 그림 6과 같이 ffmpeg로 problem.bin을 mp4로 변환 후 fps를 살펴보면 25 fps임을 알 수 있고 이를 통해 1365/25 = 54초로 영상 길이와 맞아 떨어진다는 것을 알 수 있습니다.



[그림 6] fps확인

그래서 시간 정보로 추정되는 값들은 대략 15~20개의 프레임마다 같은 값을 가지며 1씩 증가하고 있었습니다.



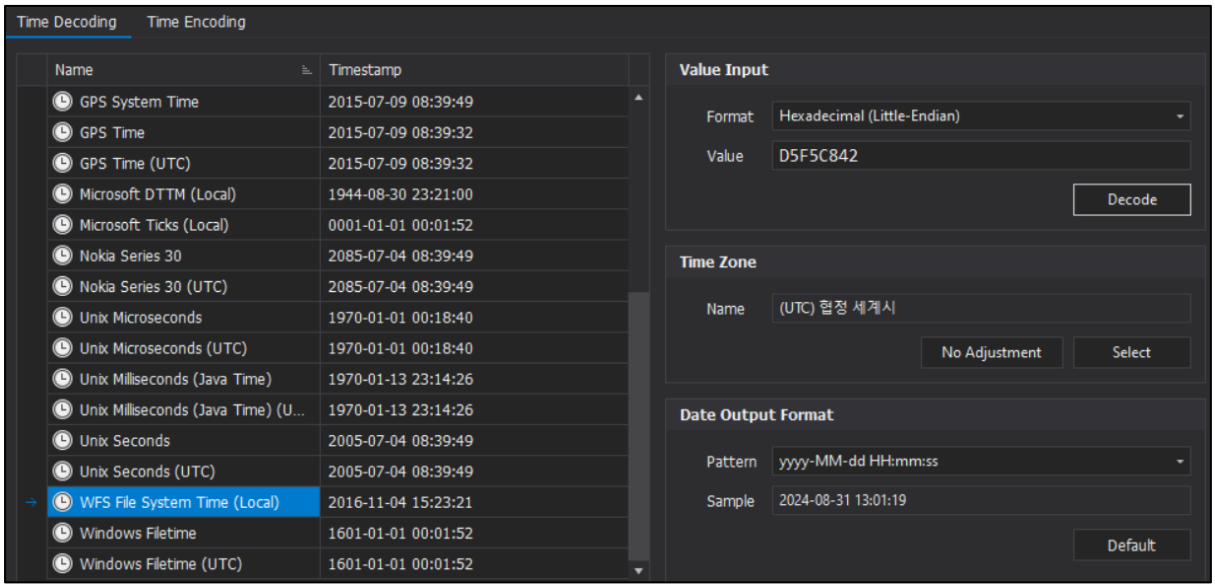
Problem.bin

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
0054A750	3E	3C	74	74	3A	4D	65	73	73	61	67	65	20	55	74	63	><tt:Message Utc
0054A760	54	69	6D	65	3D	22	32	30	31	36	2D	31	31	2D	30	34	Time="2016-11-04
0054A770	54	30	36	3A	32	33	3A	32	31	5A	22	20	50	72	6F	70	T06:23:21Z" Prop
0054A780	65	72	74	79	4F	70	65	72	61	74	69	6F	6E	3D	22	43	ertyOperation="C
0054A790	68	61	6E	67	65	64	22	3E	3C	74	74	3A	53	6F	75	72	hanged"><tt:Sour
0054A7A0	63	65	3E	3C	74	74	3A	53	69	6D	70	6C	65	49	74	65	ce><tt:SimpleIte
0054A7B0	6D	20	4E	61	6D	65	3D	22	56	69	64	65	6F	53	6F	75	m Name="VideoSou
0054A7C0	72	63	65	43	6F	6E	66	69	67	75	72	61	74	69	6F	6E	rceConfiguration
0054A7D0	54	6F	6B	65	6E	22	20	56	61	6C	75	65	3D	22	30	30	Token" Value="00
0054A7E0	30	30	30	22	2F	3E	3C	74	74	3A	53	69	6D	70	6C	65	000"/><tt:Simple
0054A7F0	49	74	65	6D	20	4E	61	6D	65	3D	22	56	69	64	65	6F	Item Name="Video
0054A800	41	6E	61	6C	79	74	69	63	73	43	6F	6E	66	69	67	75	AnalyticsConfigu
0054A810	72	61	74	69	6F	6E	54	6F	6B	65	6E	22	20	56	61	6C	rationToken" Val
0054A820	75	65	3D	22	30	30	30	30	30	22	2F	3E	3C	74	74	3A	ue="00000"/><tt:
0054A830	53	69	6D	70	6C	65	49	74	65	6D	20	4E	61	6D	65	3D	SimpleItem Name=
0054A840	22	52	75	6C	65	22	20	56	61	6C	75	65	3D	22	30	30	"Rule" Value="00
0054A850	30	30	30	22	2F	3E	3C	2F	74	74	3A	53	6F	75	72	63	000"/></tt:Sourc
0054A860	65	3E	3C	74	74	3A	44	61	74	61	3E	3C	74	74	3A	53	e><tt:Data><tt:S
0054A870	69	6D	70	6C	65	49	74	65	6D	20	4E	61	6D	65	3D	22	impleItem Name="
0054A880	49	73	4D	6F	74	69	6F	6E	22	20	56	61	6C	75	65	3D	IsMotion" Value=
0054A890	22	74	72	75	65	22	2F	3E	3C	2F	74	74	3A	44	61	74	"true"/></tt:Dat
0054A8A0	61	3E	3C	2F	74	74	3A	4D	65	73	73	61	67	65	3E	3C	a></tt:Message><
0054A8B0	2F	77	73	6E	74	3A	4D	65	73	73	61	67	65	3E	3C	2F	/wsnt:Message></
0054A8C0	77	73	6E	74	3A	4E	6F	74	69	66	69	63	61	74	69	6F	wsnt:Notificatio
0054A8D0	6E	4D	65	73	73	61	67	65	3E	3C	2F	74	74	3A	45	76	nMessage></tt:Ev
0054A8E0	65	6E	74	3E	3C	2F	74	74	3A	4D	65	74	61	64	61	74	ent></tt:Metadat
0054A8F0	61	53	74	72	65	61	6D	3E	00	00	00	00	48	04	00	00	aStream>....H...
0054A900	00	00	00	00	FC	00	00	00	8C	3B	00	00	E1	10	00	00	....ü...Œ;..á...
0054A910	D5	F5	C8	42	35	EF	0C	DB	80	00	F0	87	81	0F	08	0F	ÕõÈB5ï.Û€.ð‡....
0054A920	94	02	01	00	00	00	00	01	01	9A	00	23	12	FF	01	CC	".....š.#.ÿ.Ì

[그림 7] 특정 non-IDR frame 내 onvif metadata와 함께 위치하는 0x2C 데이터 속 시간추정 값

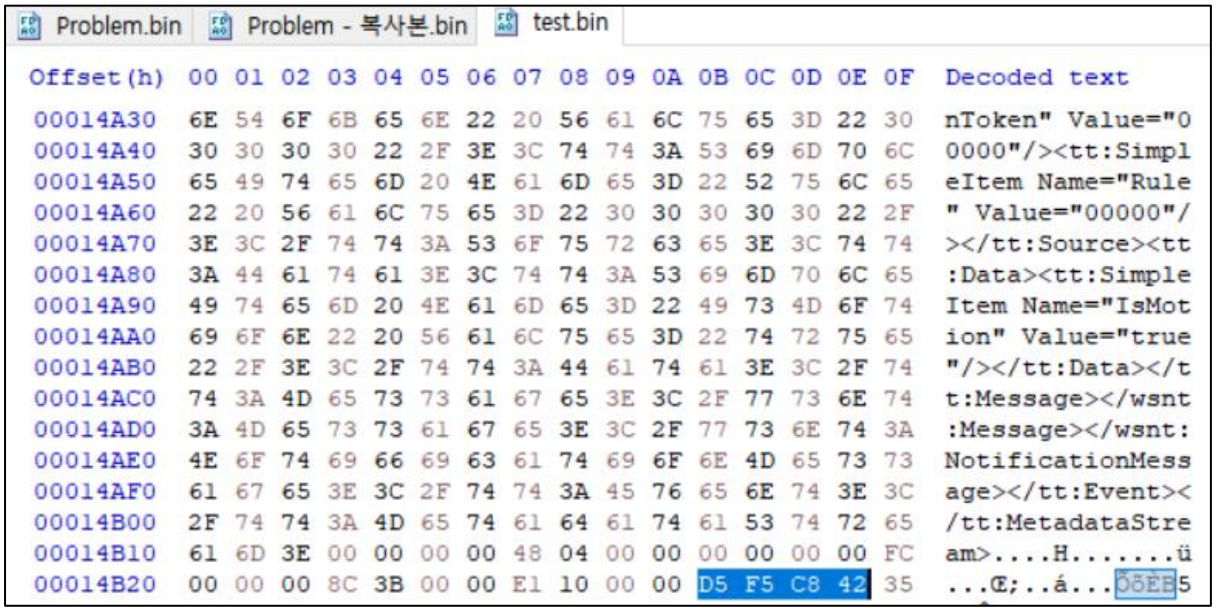
특히, 주어진 target 파일에서 프레임마다 보이는 onvif metadata가 존재했는데, 문제에서 요구하던 시간인 2016-11-04T06:23:21Z(UTC+0)을 찾을 수 있었고 그 부근인 다음 non-IDR frame이 오기전 0x2C 데이터에서는 시간 값으로 추정되는 4바이트 값을 확인할 수 있었습니다.

또한, 해당 시간 정보로 추정되는 값은 끝 값인 0x12F6C842에서 시작 값인 0xAFF5C842을 빼면 0x63으로 99초가 나오며, 메타데이터 역시 처음 시작부분부터 끝 부분까지 조회해보면, 시작시간은 2016-11-04T06:22:48Z이고 끝 시간은 2016-11-04T06:24:18Z로 약 90초 가량으로 볼 수 있는데 실질적인 비디오 play time은 54초인 것으로 보아 문제의 description에 기술된 것처럼 재생속도 변화와 같은 부분을 의심해볼 수 있습니다.



[그림 8] Dcode로 확인한 타깃 파일에 적용된 시간 정보(WFS File system Time)

그리고 Dcode 도구를 통해 의심되는 해당 hex값을 디코딩 해 본 결과, WFS file system time에서 문제에서 요구하는 시간 정보가 표출된 것을 알 수 있었습니다. 이를 통해 시간 정보로 추정되던 헥스 값이 시간 정보임을 파악할 수 있었습니다.



[그림 9] 시간정보 값이 포함된 h264 bitstream data 카빙

위 분석한 정보를 토대로 문제에서 요구하는 시간 값을 가진 프레임의 해당 장면을 재생시키기 위해 필요한 데이터인 SPS, PPS, IDR, non-IDR set 순서로 하여금 다음 SPS가 오기전까지의 NAL 데이터를 카빙하여 test.bin으로 저장하였습니다.

test.bin

H264 Bits

#	Unit type		Reference idc	Forbidden zero bit
0	**7**	Sequence parameter set	3	0
1	**8**	Picture parameter set	3	0
2	**5**	Coded slice of an IDR picture	3	0
3	**1**	Coded slice a of non-IDR picture	0	0
4	**1**	Coded slice a of non-IDR picture	3	0
5	**1**	Coded slice a of non-IDR picture	0	0
6	**1**	Coded slice a of non-IDR picture	3	0

PAYLOAD NAL

reserved_zero_2bits	0
level_idc	41
seq_parameter_set_id	0
log2_max_frame_num_minus4	9
pic_order_cnt_type	0
log2_max_pic_order_cnt_lsb_minus4	1
num_ref_frames	1
gaps_in_frame_num_value_allowed_flag	1
pic_width_in_mbs_minus1	119
pic_height_in_map_units_minus1	67
frame_mbs_only_flag	1

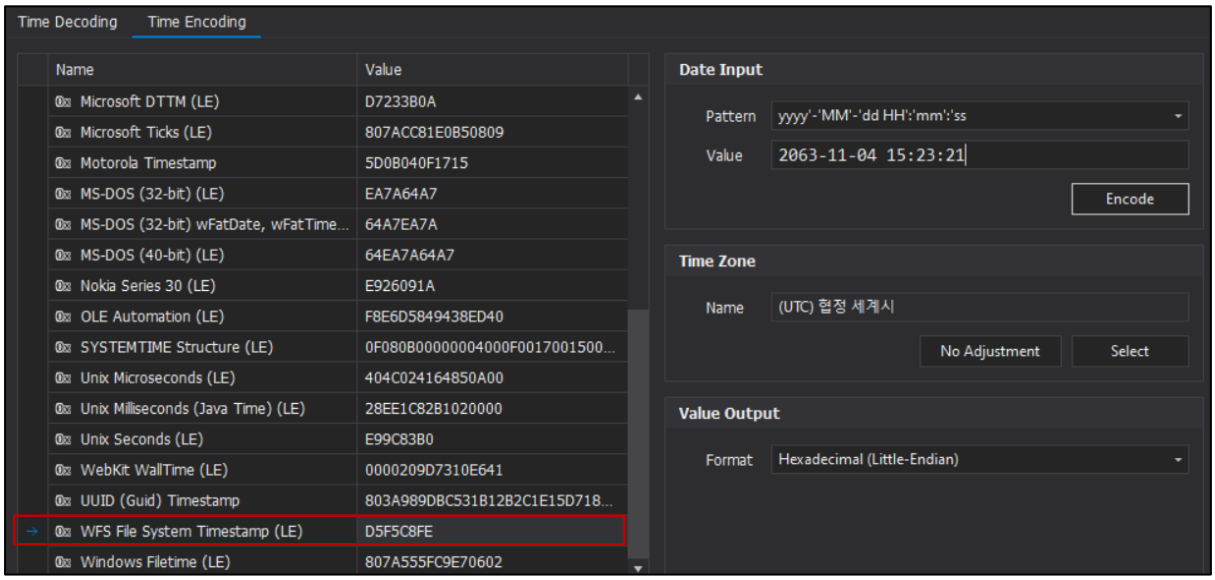
[그림 10] 카빙한 test.bin의 bitstream 구조

[그림 11] ffplay로 재생한 test.bin

그 후, ffplay를 통해 test.bin을 play해주면, 위 그림과 같이 하나의 그림이 나타납니다. 해당 그림은 **The Anatomy Lesson of Dr. Nicolaes Tulp**로 판단됩니다.

답 : The Anatomy Lesson of Dr. Nicolaes Tulp

2. Submit the time information of November 4, 2063, 15:23:21 [hour:minute:second] as a hex value(big endian) (100 points)



[그림 12] 2016-11-04T06:23:21Z에 대한 hex값 정보

앞서 1번 문제에서 분석한 내용을 토대로라면 2016-11-04T06:23:21Z에 대한 hex값은 0xD5F5C842였습니다. 따라서, 2063년 11월 4일 15:23:21에 대한 시간정보는 Dcode에서 인코딩해보면 0xD5F5C8FE로 나타낼 수 있습니다. 기존에 time decoding시에 little-endian으로 계산하여 나온 값이었기 때문에 big endian으로 나타내면 **0xFEC8F5D5**입니다.

답 : 0xFEC8F5D5