[DFC-2023-302] Write-Up

작성자	허은정, 김경민
분석 일자	2024.05.24~2024.05.31
작성 일자	2024.06.01
분석 대상	Seoul.MP4
문서 버전	2.0
작성자 E-mail	dmswjs4315@yonsei.ac.kr rlarudals877@gmail.com

0. 목차

1. 문제

URL	-
문제 내용	Analyze the video and prevent a terrorist attack! 1) When is the attack scheduled? (50 points) 2) What is the cryptographic key is needed to identify the location of the attack? (125 points) 3) Where is the attack scheduled? (125 points)
문제 파일	-
문제 유형	multimedia forensics
난이도	3/ 3

2. 분석 도구

도구명	다운로드 링크	Version
HxD	https://mh-nexus.de/en/downloads.php?product=HxD20	2.5.0
Dcode	https://www.digital-detective.net/dcode/	5.6
ffmpeg	https://ffmpeg.org/	4.4.2
CyberChef	https://gchq.github.io/CyberChef/	10.18.6
StegSolve	http://www.caesum.com/handbook/Stegsolve.jar	1.3

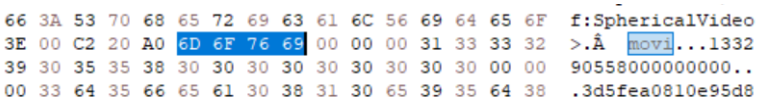
3. 환경

OS
Window 11 64-bit

4. Write-Up

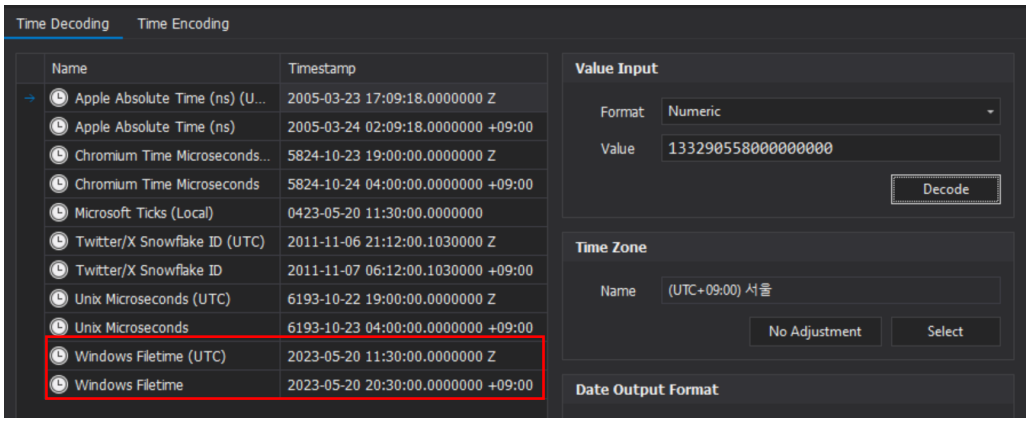
파일명	Seoul.MP4
용량	3.36GB
SHA256	b9424e47b27caffaf747e123690a7c1c85e8c25640a1c743b5cde1868b635040
Timestamp	2024-05-25 14:40:36

1) When is the attack scheduled? (50 points)



[사진 1] 해당 파일을 열어본 후 movi을 확인해본 결과

문제 파일을 열어서 MP4 파일의 구조들을 확인해보던 중 movi 가 존재하는 것을 확인하였다. movi 뒤의 이상한 데이터 값이 존재한다는 것을 알게 되었다.



[사진 2] 이상한 데이터 값을 Dcode에 넣어본 결과

Movi 뒤의 이상한 데이터인 '133290558000000000'를 Dcode 에 넣어 시각 값을 디코딩해본 결과, 예정된 공격 시간은 '2023-05-20 20:30:00 UTC+9'라는 것을 알 수 있다.

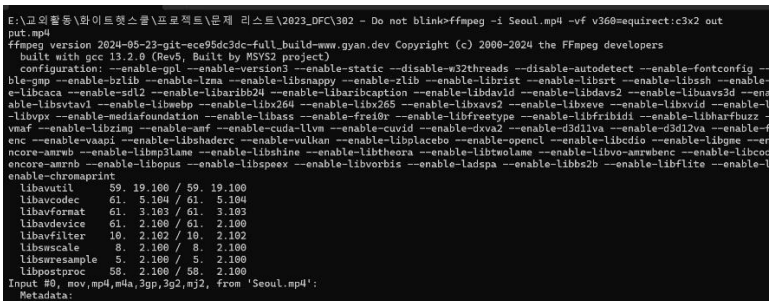
2) **What is the cryptographic key is needed to identify the location of the attack? (125 points)**

[사진 3] ffprobe Seoul.mp4를 입력하여 확인해본 결과

Seoul.mp4 의 상세 정보를 확인하기 위해 ffprobe Seoul.mp4 를 입력하여 확인해보았다. 이를 통해, 동영상의 spherical 이 equirectangular 임을 알 수 있다.

[사진 4] 문제 영상 열어본 결과

이를 확인 후 해당 영상을 열어본 결과, 왜곡된 형태로 동영상이 나오고 있다는 것을 알 수 있다.



[사진 5] 왜곡 없는 프레임을 얻기 위해 명령어 사용

ffmpeg 를 이용해 동영상의 프로젝션 타입을 cubemap 으로 변확하여 왜곡 없는 프레임을 얻고자 하였다. 이를 통해 360 도 환경을 정육면체로 보고 각 방향을 6 개의 정사각형을 표현해 왜곡 없이 모든 면을 얻고자 하였다.

```
E:\교외활동\화이트햇스쿨\프로젝트\문제 리스트\2023_DFC\302 - Do not blink>ffmpeg -i output.mp4 -filter_complex " [0:v]split=6[c0][c1][c2][c3][c4][c5]; [c0] crop=iw/3:ih/2:0:0[c0];
 [c1] crop=iw/3:ih/2:iw/3:0[c1]; [c2] crop=iw/3:ih/2:2*iw/3:0[c2]; [c3] crop=iw/3:ih/2:0:ih/2[c3];  [c4] crop=iw/3:ih/2:iw/3:ih/2[c4];  [c5] crop=iw/3:ih/2:2*iw/3:ih/2[c5]" -map "
[c0]" c0.mp4 -map "[c1]"c1.mp4 -map "[c2]" c2.mp4 -map "[c3]" c3.mp4 -map "[c4]"c4.mp4 -map "[c5]" c5.mp4
```

[사진 6] 6개의 면을 분할

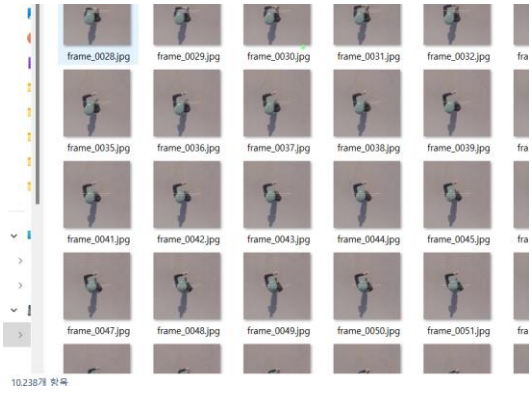
모든 면을 얻은 후 6 개의 면을 별도의 동영상으로 분할하였다.

[사진 7] 분할된 영상

분할 후 해당 영상의 확인해보면 모든 각도에서 영상을 확인할 수 있다. 해당 영상에서 자막이 있는 영상에 암호화된 키가 있다는 의심을 하였다.

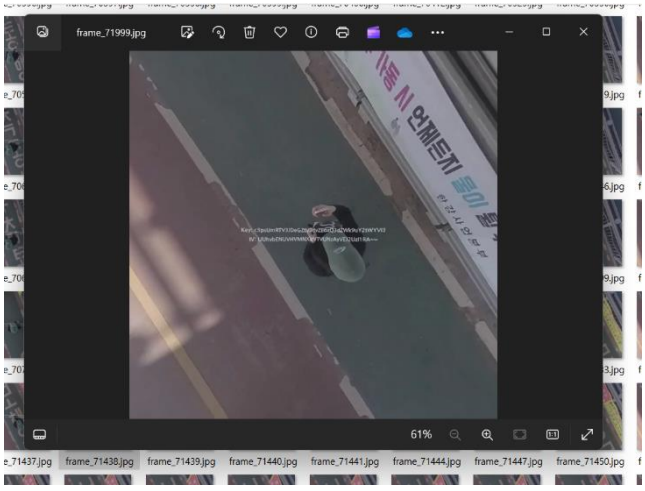
따라서 자막이 있는 부분의 RGB 값을 비교하여 자막 값이 달라지는 경우 해당 프레임을 저장하는 *코드를 만들어 실행하였다. 그 결과, 총 10238 개의 프레임이 특정 RGB 값을 벗어난다는 것을 알 수 있다.

*별도 첨부에서 확인



[사진 8] 추출된 프레임

해당 자막 부분들을 일일이 찾아본 결과 71999 번째 프레임에서 기존 자막 포맷을 벗어난 문자열이 존재했으며 해당 프레임 확인 결과 Key, Iv 를 확인할 수 있다.



[사진 9] key랑 Iv가 나온 프레임

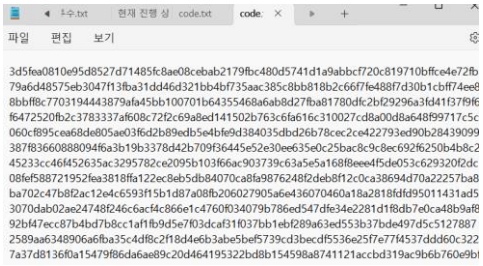
따라서 key는 c3psUmRTV3JDeGZtV3dvZEoxQ3dZWk9uY2tWYVI3이고

IV는 UhvbENUVHVMNXlzVTVUNzAyVEJ2Uzl1RA==이다.

3) **What is the cryptographic key is needed to identify the location of the attack? (125 points)**

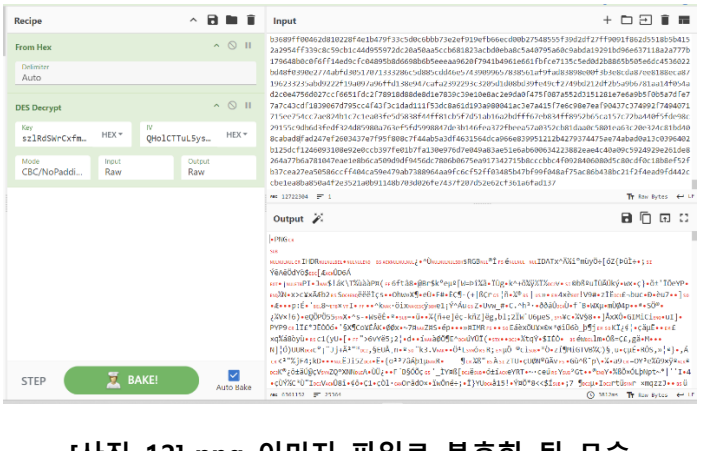
[사진 10] 발견된 hex 문자열

Seoul.mp4를 헥스를 통해서 다시 보면 movi 박스가 나오고 약 12MB 정도의 hex 문자열이 저장되어 있는 것을 볼 수 있다. 확인해 보니 엔트로피가 높았다. 따라서 암호화된 데이터라고 추측할 수 있었다. 해당 영역을 추출해보면 다음과 같다.



[사진 11] 추출된 hex 문자열

앞선 문제에서 나온 답은 base64로 디코딩 할 수 있어서 디코딩을 하였다.



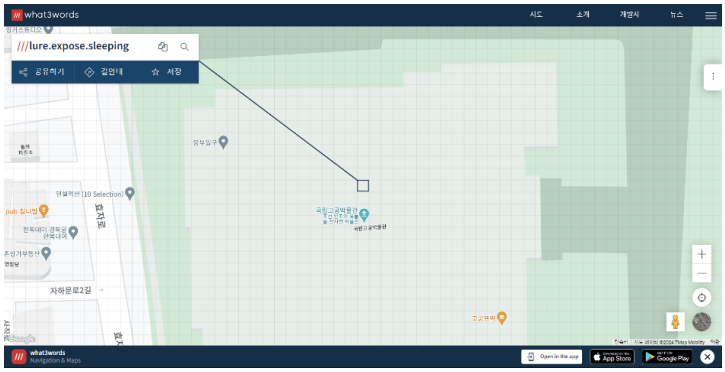
[사진 12] png 이미지 파일로 복호화 된 모습

Cyberchef(https://gchq.github.io/CyberChef/)에서 ket, iv, 그리고 위의 hex 코드를 가지고 png 이미지 파일로 복호화가 되었다.

[사진 13] 복호화해서 나온 png 파일

[사진 14] 이미지 속 숨겨진 문자열

[사진 14]를 보고 스테가노그래피 문제인 것 같아 stegSolve 를 사용하여 숨겨진 문자열을 찾았다.



[사진 15] 공격이 예상된 장소

그리고 해당 문자열을 그대로 크롬에서 검색해 보았더니 지리 코드라는 것을 알았다.

따라서 what3words 사이트

(https://what3words.com/%ED%91%9C%ED%98%84.%EB%A9%B4%ED%97%88%EC%A6%9D.%EA%B0%80%EB%B2%BC%EC%9A%B4)에서 코드를 그대로 입력했고 공격이 예상된 장소는 '국립고궁박물관 입구'인 것을 알 수 있었다.

5. Flag

1) 2023-05-20 20:30:00 UTC+9

2) Key: c3psUmRTV3JDeGZtV3dvZEoxQ3dZWk9uY2tWYVI3

IV: UUhvbENUVHVMNXlzVTVUNzAyVEJ2Uzl1RA==

3) 국립고궁박물관 입구

2) 별도 첨부

```python
import numpy as np  # type: ignore
import os

def save_frame_on_subtitle_change(video_path, subtitle_area, output_folder):
    cap = cv2.VideoCapture(video_path)
    prev_frame = None
    frame_count = 0
    saved_frame_count = 0

    os.makedirs(output_folder, exist_ok=True)

    while cap.isOpened():
        ret, frame  = cap.read()
        if not ret:
            break

        subtitle_frame = frame[subtitle_area[1]:subtitle_area[3], subtitle_area[0]:subtitle_area[2]]
        subtitle_gray = cv2.cvtColor(subtitle_frame, cv2.COLOR_BGR2GRAY)

        if prev_frame is None:
            prev_frame = subtitle_gray
            continue

        difference = cv2.absdiff(prev_frame, subtitle_gray)
        mean_diff = np.mean(difference)

        if mean_diff > 5:
            frame_filename = os.path.join(output_folder, f'frame_{frame_count:04d}.jpg')
            cv2.imwrite(frame_filename, frame)
            print(f'Saved {frame_filename}')
            saved_frame_count += 1

        prev_frame = subtitle_gray
        frame_count += 1

    cap.release()
    cv2.destroyAllWindows()
    print(f'Total frames saved: {saved_frame_count}')

video_path = 'c3.mp4'
subtitle_area = (262, 437, 686, 509)
output_folder = 'jpg2'

save_frame_on_subtitle_change(video_path, subtitle_area, output_folder)
```

3) Reference

- https://blog.naver.com/sampoo00/30179559424
- https://mj-thump-thump-story.tistory.com/entry/OCR-Tesseract-Windows-%ED%99%98%EA%B2%BD%EC%97%90-%EC%85%8B%EC%97%85#google_vignette