[DFC-2023-302] Write-Up

작성자	허은정
분석 일자	2024.05.24~2024.05.31
작성 일자	2024.06.01
분석 대상	Seoul.MP4
문서 버전	1.0
작성자 E-mail	dmswjs4315@yonsei.ac.kr

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	2/ 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

3. 환경

OS
Window 11 64-bit

4. Write-Up

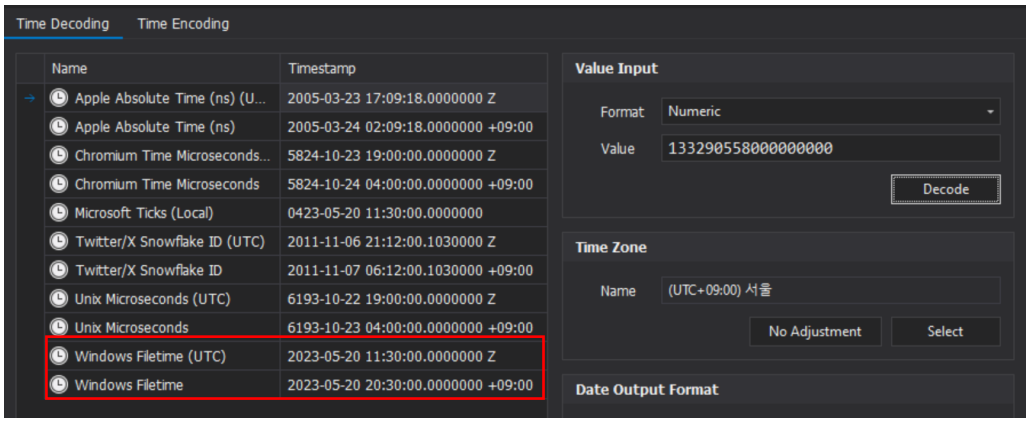
파일명	Seoul.MP4
용량	3.36GB
SHA256	b9424e47b27caffaf747e123690a7c1c85e8c25640a1c743b5cde1868b635040
Timestamp	2024-05-25 14:40:36

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

```
66 3A 53 70 68 65 72 69 63 61 6C 56 69 64 65 6F  f:SphericalVideo
3E 00 C2 20 A0 6D 6F 76 69 00 00 00 31 33 33 32  >.Â  movi...1332
39 30 35 35 38 30 30 30 30 30 30 30 30 30 00 00  90558000000000..
00 33 64 35 66 65 61 30 38 31 30 65 39 35 64 38  .3d5fea0810e95d8
```

[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.

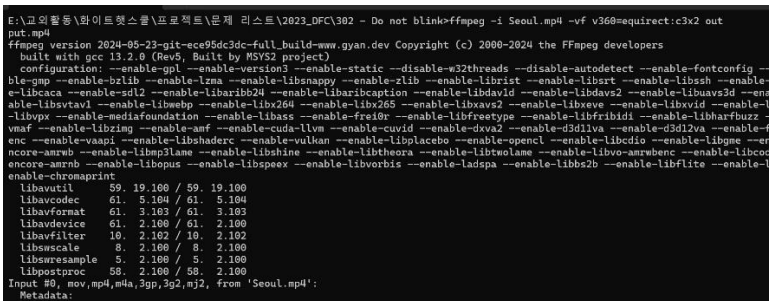
2) **What is the cryptographic key is needed to identify the location of the attack? (125 points)**

[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.

```
Total frames saved: 14554
```

[사진 8] RGB값이 달라진 프레임 개수

따라서 자막이 있는 부분의 RGB 값을 비교하여 자막 값이 달라지는 경우 해당 프레임을 저장하는 *코드를 만들어 실행하였다. 그 결과, 총 14554 개의 프레임이 특정 RGB 값을 벗어난다는 것을 알 수 있다.

*별도 첨부에서 확인

해당 자막 부분들을 Tesseract OCR 을 사용하여 CSV 에 기록하는 *코드를 만들어 실행하려고 하였는데 값이 추출되지 못하였다.

*별도 첨부에서 확인

5. Flag

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

6. 별도 첨부

```python
import cv2
import numpy as np
from google.colab import drive
drive.mount('/content/gdrive')

def save_frame_on_subtitle_change(video_path, subtitle_area, output_folder):
    # Open the video file
    cap = cv2.VideoCapture(video_path)
    prev_frame = None
    frame_count = 0
    saved_frame_count = 0

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

        # Crop the subtitle area from the frame
        subtitle_frame = frame[subtitle_area[1]:subtitle_area[3], subtitle_area[0]:subtitle_area[2]]

        # Convert to grayscale for comparison
        subtitle_gray = cv2.cvtColor(subtitle_frame, cv2.COLOR_BGR2GRAY)

        if prev_frame is None:
            prev_frame = subtitle_gray
            continue

        # Compare the current subtitle frame with the previous one
        difference = cv2.absdiff(prev_frame, subtitle_gray)
        mean_diff = np.mean(difference)

        # If there is a significant difference, save the frame
        if mean_diff > 5:  # Adjust the threshold as needed
            frame_filename = f'{output_folder}/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}')

# Example usage
video_path = '/content/gdrive/MyDrive/output.mp4'
subtitle_area = (100, 500, 1180, 580)  # Example coordinates (x1, y1, x2, y2)
output_folder = '/content/gdrive/MyDrive/DFC2023'  # Folder where frames will be saved

save_frame_on_subtitle_change(video_path, subtitle_area, output_folder)
```

```python
import cv2
import pytesseract
import os
import csv

# 프레임 이미지가 저장된 폴더 경로
frames_folder = 'C:/Users/a0104/Downloads/DFC2023-20240530T025420Z-001/DFC2023'
# 출력 CSV 파일 경로
output_csv = 'output.csv'

# Tesseract 실행 파일 경로 (Windows 경로 예시)
pytesseract.pytesseract.tesseract_cmd = r'E:₩Tesseract-OCR₩tesseract.exe'

# 주어진 이미지에 OCR을 수행하는 함수
def ocr_image(image_path):
    # 이미지를 읽어옵니다
    img = cv2.imread(image_path)
    # 이미지를 RGB로 변환합니다 (OpenCV는 기본적으로 BGR을 사용합니다)
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    # OCR을 수행합니다
    text = pytesseract.image_to_string(img_rgb, lang='eng')
    return text

# OCR 결과를 저장할 리스트
ocr_results = []

# 폴더 내 모든 프레임 이미지를 순회합니다
for frame_file in os.listdir(frames_folder):
    if frame_file.endswith('.jpg'):
        frame_path = os.path.join(frames_folder, frame_file)
        # 프레임에 OCR을 수행합니다
        ocr_text = ocr_image(frame_path)
        # 결과를 리스트에 추가합니다
        ocr_results.append([frame_file, ocr_text])

# OCR 결과를 CSV 파일에 작성합니다
with open(output_csv, mode='w', newline='', encoding='utf-8') as file:
    writer = csv.writer(file)
    writer.writerow(['Frame', 'Text'])
    writer.writerows(ocr_results)

print(f'OCR 결과가 {output_csv} 파일에 저장되었습니다.')
```

7. 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