

## 151 - Layers of Secrecy

### Team Information

Team Name : HSPACE

Team Member : Jinung Lee, Beomjun Park, DoHyeon Kim, Soyoung Cho

Email Address : [hspacedigitalforensicslab@gmail.com](mailto:hspacedigitalforensicslab@gmail.com)

Teams must:

- Provide a detailed, step-by-step description of their problem-solving approach to ensure reproducibility by another examiner.
- List all tools used to arrive at their conclusions.

### Tools used:

Name:	Hxd	Publisher:	Maël Hörz
Version:	2.5.0.0		
URL:	URL: <a href="https://mh-nexus.de/">https://mh-nexus.de/</a>		

Name:	Visual studio code	Publisher:	Microsoft
Version:	1.104.2		
URL:	<a href="https://code.visualstudio.com/download">https://code.visualstudio.com/download</a>		

Name:	Sublime text	Publisher:	Sublime HQ Pty Ltd
Version:	3.2		
URL:	<a href="https://www.sublimetext.com/3">https://www.sublimetext.com/3</a>		

## Step-by-step methodology:

문제 풀이에 앞서, dfchallenge.org에 공지된 문제 해시와 다운로드 받은 문제 해시를 비교함으로써 분석 대상이 동일한 파일임을 증명한다.

### Hash Value (MD5)

- Lecture.pptx : 1425d412169f6b3b63f6ec6c7a911665
- \_con : feaad600c2b33ba7780fab65c9afaddc

Figure 1. dfchallenge.org에 명시되어 있는 MD5 값

lecture.pptx 속성	
이름	해시 값
CRC32	2F4325E0
MD5	1425D412169F6B3B63F6EC6C7A911665
SHA-1	B6AB0ADE71E076C274051DFCF7B6587B39A36D39
SHA-256	20A7E81FAD347C7D0140284DF4BB057E1B7F9CA9531DC26.

Figure 2. lecture.pptx MD5값

_con 속성	
이름	해시 값
CRC32	056C99CB
MD5	FEAAD600C2B33BA7780FAB65C9AFADDc
SHA-1	9BC7B9A563FF8516B6162692C2CEAEA415011698
SHA-256	57065A4FF265991FAD980EE34A6EB20EACE716BD822C057..

Figure 3. \_con 파일의 MD5값

두 파일의 내용을 확인해보면, 다음과 같다. lecture.pptx 파일의 경우 Powerpoint로 열었을 때, 699개의 슬라이드를 확인할 수 있다.

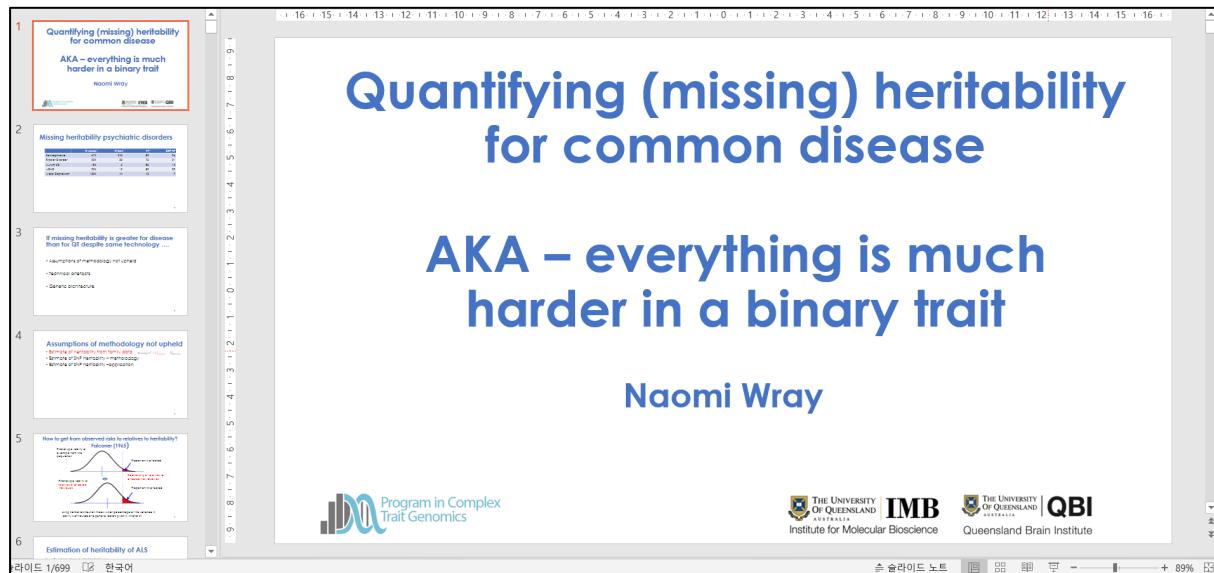


Figure 4. pptx 파일 내용 중 일부

\_con 파일의 경우 파일 자체로는 열리지 않으며, HxD로 확인해 본 결과 다음과 같이 암호화되어 있음을 알 수 있다.

Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded text
00000000	31 CA D6 B0 95 91 4A 4D 6B EB 30 38 82 4D 5C 67	ÍÉÖ°• 'JMkë08,M\g
00000010	1A 7C B3 60 D3 36 CA F6 E6 83 27 63 A0 43 AA C6	. ³Ó6Éöæf'c C²E
00000020	4C 9D 45 E2 23 90 FC 53 5B 0F F5 88 28 67 0F 77	L.Eâ#.#üS[.õ^ (g.w
00000030	4F 52 6A C7 B7 2E 3A 36 AC EA 29 A6 C5 CF 2F 89	ORjÇ..:6-ê)  ÅÍ/‰
00000040	AC 88 E5 74 6F 43 92 4C D2 30 C3 C5 90 92 69 2B	-^åtoC'LÒOÅÅ.'i+
00000050	22 71 27 BB 31 43 22 C2 A7 75 14 D1 4E 63 BA 98	"q'»lC"ÅSu.ÑNc°~
00000060	7C C4 87 BC C3 1B B9 D6 21 92 BD 5F 00 64 E1 71	Ä+uÃ.ºÖ!'%_.dáq
00000070	B8 B1 61 1C DC FA F6 59 31 66 B3 24 A3 B1 85 A8	,fa.ÜúöYlf'§£±.."
00000080	52 D7 1C E9 49 95 08 DE 23 84 46 0D DC 35 E7 5C	R×.éI..ß#.F.Ü5ç\
00000090	33 DB C8 C2 9C CA AF 34 10 31 B4 AA 5A F0 16 27	3ÜÉÀœÈ~4.1~Zð.'
000000A0	DF 95 59 09 DC 3C 5D 42 02 4A 4A 95 E3 A9 16 6B	B•Y.Ü<]B.JJ•ä@.k
000000B0	35 4A DC 4F F6 F8 71 7D 63 8C 92 9A 46 FA 7A 1B	5JÜOöøq}c@'šFúz.
000000C0	C1 8F C6 D4 9E 9B 26 FC 5E 75 B4 CD 90 01 7D 75	Á.EÖž>&ü^u'í..}u
000000D0	AF BF 30 78 DE DA 45 95 74 40 41 F3 31 CC 75 57	~ç0xþÚE•t@AólíuW
000000E0	65 5B CD 76 3C 84 F0 A4 8B 03 7F 94 30 40 61 F0	e[ív<,ð¤<.."0@að
000000F0	AF 54 32 3B 57 43 63 BA EF 92 24 44 89 1D C7 AA	T2;WCc°i'\$D‰.Ç²
00000100	88 C5 29 2C FF 83 6E A6 C7 7A 47 9E 3B D8 A0 F7	~À),ýfn;ÇzGž;Ø ÷
00000110	9E 92 E4 BE 54 23 BE 05 82 B8 FC A5 C3 29 42 F4	ž'å¤T#%,.,ü¥Å)Bô
00000120	20 39 7F 9D D9 12 46 78 54 24 D6 65 0E 1C 84 8D	9..Ù.FxT\$Öe....

Figure 5. \_con 파일 내용 중 일부

## 1. 용의자가 유출하고자 했던 파일의 MD5 해시 값은 무엇인가?

lecture.pptx의 확장자를 .zip으로 바꿔 압축을 풀면, 다음과 같이 파일 내부 구성 요소들을 확인할 수 있다.

_rels	파일 폴더
docProps	파일 폴더
ppt	파일 폴더
[Content_Types].xml	Microsoft Edge HTML Do... 7KB 아니요 222KB

Figure 6. lecture.ppt 파일 확장자 변경 이후 확인한 모습

"ppt\\slides" 폴더를 확인해보면, slide1.xml부터 slide700.xml까지 총 700개의 슬라이드 파일이 존재하는 것을 확인할 수 있다. 그러나 PowerPoint로 직접 열었을 때는 699장으로 표시되는 것으로 보아, 1개의 슬라이드가 삭제되었거나 숨겨진 상태임을 알 수 있다.

slide1.xml	Microsoft Edge HTML Do...	2KB	아니요	5KB	68%	1899-12-30 오전 9:00
slide2.xml	Microsoft Edge HTML Do...	2KB	아니요	11KB	85%	1899-12-30 오전 9:00
slide3.xml	Microsoft Edge HTML Do...	2KB	아니요	4KB	70%	1899-12-30 오전 9:00
slide4.xml	Microsoft Edge HTML Do...	2KB	아니요	15KB	86%	1899-12-30 오전 9:00
slide694.xml	Microsoft Edge HTML Do...	1KB	아니요	2KB	60%	1899-12-30 오전 9:00
slide695.xml	Microsoft Edge HTML Do...	1KB	아니요	2KB	64%	1899-12-30 오전 9:00
slide696.xml	Microsoft Edge HTML Do...	1KB	아니요	2KB	52%	1899-12-30 오전 9:00
slide697.xml	Microsoft Edge HTML Do...	1KB	아니요	2KB	56%	1899-12-30 오전 9:00
slide698.xml	Microsoft Edge HTML Do...	1KB	아니요	5KB	79%	1899-12-30 오전 9:00
slide699.xml	Microsoft Edge HTML Do...	1KB	아니요	2KB	56%	1899-12-30 오전 9:00
slide700.xml	Microsoft Edge HTML Do...	1KB	아니요	2KB	56%	1899-12-30 오전 9:00

Figure 7. ppt 폴더 내부 slides 값 확인

presentation.xml의 슬라이드 참조 목록(sldIdLst)을 확인한 결과, rId535가 누락되어 해당 슬라이드가 로드되지 않는 것으로 확인되었다.

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <p:presentation xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main"
3  <p:sldId id="1573" r:id="^Id534"/><p:sldId id="1575" r:id="rId536"/>
4  <p:sldId id="1576" r:id="^Id537"/><p:sldId id="1577" r:id="rId538"/>
5  <p:sldId id="1578" r:id="^Id539"/><p:sldId id="1579" r:id="rId540"/>
6  <p:sldId id="1580" r:id="^Id541"/><p:sldId id="1581" r:id="rId542"/><p:sldId id=
```

Figure 8. presentation.xml 내부 rId535가 누락된 모습

따라서, 다음 파이썬 코드를 통해 누락되어 있는 페이지를 복구하였다.

```
import zipfile
import os
import xml.etree.ElementTree as ET
from xml.dom import minidom

def fix_powerpoint_slides(input_pptx, output_pptx):
    """
    PowerPoint 파일에서 누락된 rId535 슬라이드 참조를 복구
    """

    # 임시 디렉토리 생성
    temp_dir = "temp_pptx_extract"
    if os.path.exists(temp_dir):
        import shutil
        shutil.rmtree(temp_dir)
    os.makedirs(temp_dir)

    try:
        # PPTX 파일을 ZIP 으로 해제
        with zipfile.ZipFile(input_pptx, 'r') as zip_ref:
            zip_ref.extractall(temp_dir)

        # presentation.xml 파일 경로
        presentation_xml_path = os.path.join(temp_dir, 'ppt',
                                              'presentation.xml')

        # presentation.xml 파일 읽기
        with open(presentation_xml_path, 'r', encoding='utf-8') as f:
            content = f.read()

        # XML 파싱
        root = ET.fromstring(content)

        # 네임스페이스 정의
        namespaces = {
            'p': 'http://schemas.openxmlformats.org/presentationml/2006/main',
            'r': 'http://schemas.openxmlformats.org/officeDocument/2006/relationships'
        }

        # sldIdLst 찾기
        sld_id_lst = root.find('.//p:sldIdLst', namespaces)

        if sld_id_lst is None:
            print("sldIdLst 를 찾을 수 없습니다.")

    
```

```

        return False

    # rId534 와 rId536 사이에 rId535 추가하기
    # 먼저 기존 요소들 찾기
    existing_ids = []
    for sld_id in sld_id_lst.findall('p:sldId', namespaces):
        rid =
sld_id.get('{http://schemas.openxmlformats.org/officeDocument/2006/relationships}id')
        existing_ids.append(rid)

    # rId535 가 이미 있는지 확인
    if 'rId535' in existing_ids:
        print("rId535 가 이미 존재합니다.")
        return False

    # rId534 와 rId536 의 위치 찾기
    rid534_element = None
    rid536_element = None

    for sld_id in sld_id_lst.findall('p:sldId', namespaces):
        rid =
sld_id.get('{http://schemas.openxmlformats.org/officeDocument/2006/relationships}id')
        if rid == 'rId534':
            rid534_element = sld_id
        elif rid == 'rId536':
            rid536_element = sld_id

    if rid534_element is None or rid536_element is None:
        print("rId534 또는 rId536 를 찾을 수 없습니다.")
        return False

    # rId535 요소 생성
    new_sld_id =
ET.Element('{http://schemas.openxmlformats.org/presentationml/2006/main}sldId')
    new_sld_id.set('id', '1574')
    new_sld_id.set('{http://schemas.openxmlformats.org/officeDocument/2006/relationships}id', 'rId535')

    # rId536 요소 바로 앞에 rId535 삽입
    elements = list(sld_id_lst)
    rid536_index = elements.index(rid536_element)
    sld_id_lst.insert(rid536_index, new_sld_id)

    # XML 문서를 문자열로 변환

```

```

xml_str = ET.tostring(root, encoding='unicode')

# minidom 을 사용해서 보기 좋게 포맷팅
dom = minidom.parseString(xml_str)
pretty_xml = dom.toprettyxml(indent="  ")

# XML 선언 중복 제거
lines = pretty_xml.split('\n')
if lines[0].startswith('<?xml') and lines[1].startswith('<?xml'):
    lines = lines[1:]
pretty_xml = '\n'.join(lines)

# 수정된 내용을 파일에 저장
with open(presentation_xml_path, 'w', encoding='utf-8') as f:
    f.write(pretty_xml)

# 수정된 파일들을 새로운 PPTX 로 압축
with zipfile.ZipFile(output_pptx, 'w', zipfile.ZIP_DEFLATED) as zipf:
    for root_dir, dirs, files in os.walk(temp_dir):
        for file in files:
            file_path = os.path.join(root_dir, file)
            arcname = os.path.relpath(file_path, temp_dir)
            zipf.write(file_path, arcname)

    print(f"복구된 PowerPoint 파일이 {output_pptx}로 저장되었습니다.")
    print("rId535 슬라이드 참조가 추가되어 이제 모든 700 개 슬라이드를 볼 수 있습니다.")

return True

except Exception as e:
    print(f"오류 발생: {e}")
    return False

finally:
    # 임시 디렉토리 정리
    if os.path.exists(temp_dir):
        import shutil
        shutil.rmtree(temp_dir)

# 사용 예시
if __name__ == "__main__":
    input_file = "lecture.pptx" # 원본 PPTX 파일명을 입력하세요
    output_file = "result.pptx"

    if os.path.exists(input_file):

```

```

success = fix_powerpoint_slides(input_file, output_file)
if success:
    print("복구 완료!")
else:
    print("복구 실패!")
else:
    print(f"입력 파일 '{input_file}'을 찾을 수 없습니다.")
    print("스크립트와 같은 폴더에 PPTX 파일을 두고 파일명을 확인하세요.")

```

Table 1. ppt 페이지 복구 코드

rId535가 나타내는 실제 슬라이드 번호를 확인하기 위하여 "WlectureWpptW\_relsWpresentation.xml.rels" 를 확인해준 결과 512 슬라이드가 복구된 것을 알 수 있다.

```

<Relationship Id="rId535" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/slide" Target="slides/slide512.xml"/>
<Relationship Id="rId174" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/slide" Target="slides/slide151.xml"/>
<Relationship Id="rId381" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/slide" Target="slides/slide358.xml"/>

```

Figure 9. presentation.xml.rels에서 복구된 슬라이드 번호 확인

복구된 512 슬라이드쇼는 다음과 같다.

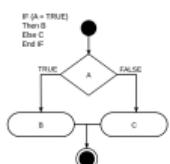
## Worksheet Section II: Does The Claimed Invention Fall Within A Statutory Category (Step 1)?

*Process*

*Machine*

*Manufacture*

*Composition  
of Matter*



Examiners should fill out Section II of the blank worksheet

512

Figure 10. 복구된 512 슬라이드 페이지

해당 슬라이드 페이지에는 여러 개의 그림이 삽입되어 있는데, "그림 서식 > 대체 텍스트"를 확

인하면, 자동자 사진에서 다음과 같이 숨겨져 있는 코드를 확인할 수 있다.

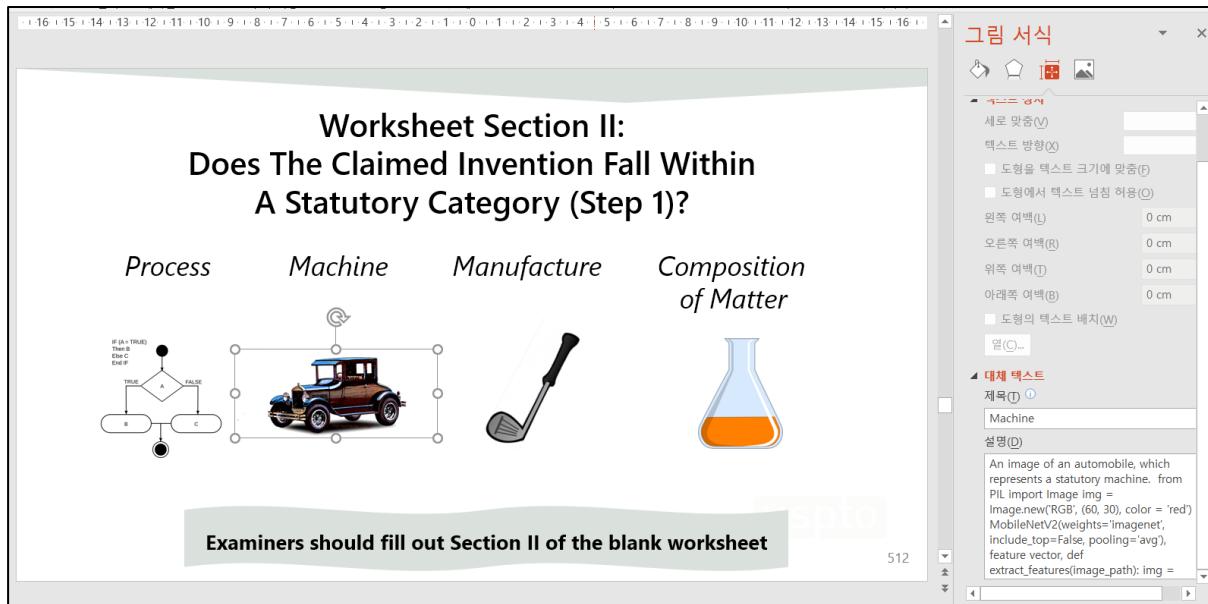


Figure 11. 512 페이지의 숨겨진 코드

숨겨진 코드에는 **이미지 특징 추출 및 암호화** 관련 코드가 포함되어 있다.

```
An image of an automobile, which represents a statutory machine. from PIL import Image
img = Image.new('RGB', (60, 30), color = 'red')
MobileNetV2(weights='imagenet', include_top=False,
pooling='avg'),
feature vector, def extract_features(image_path):
    img =
tf.keras.preprocessing.image.load_img(image_path,
target_size=(224, 224))
img_array =
tf.keras.preprocessing.image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array =
tf.keras.applications.mobilenet_v2.preprocess_input(img_array)
features =
model.predict(img_array)
return features.flatten()
SHA-384, def hash_features(feature_vector):
feature_vector_bytes =
feature_vector.astype(np.float32).tobytes()
sha384_hash =
hashlib.sha384(feature_vector_bytes).digest()
return AES-256 IV=16
```

Table 2. 512 슬라이드 자동차 그림에 숨겨진 코드

해당 코드를 토대로 코드를 작성하였으며, 작성된 코드는 다음과 같다.

```
from PIL import Image
img = Image.new('RGB', (60, 30), color='red')
MobileNetV2(weights='imagenet', include_top=False, pooling='avg')
def extract_features(image_path):
    img = tf.keras.preprocessing.image.load_img(image_path, target_size=(224, 224))
```

```
img_array = tf.keras.preprocessing.image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array = tf.keras.applications.mobilenet_v2.preprocess_input(img_array)
features = model.predict(img_array)
return features.flatten()

def hash_features(feature_vector):
    feature_vector_bytes = feature_vector.astype(np.float32).tobytes()
    sha384_hash = hashlib.sha384(feature_vector_bytes).digest()
    return sha384_hash
```

해당 코드를 사용하여 이미지를 분석한 결과 \_con과 연관된 증거를 찾을 수 없다.