151 - Layers of Secrecy

Team Information

Team Name : HSPACE

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

Email Address : 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: https://mh-nexus.de/		

Name:	Visual studio code	Publisher:	Microsoft
Version:	1.104.2		
URL:	https://code.visualstudio.com/download		

Name:	Sublime text	Publisher:	Sublime HQ Pty Ltd
Version:	3.2		
URL:	https://www.sublimetext.com/3		

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

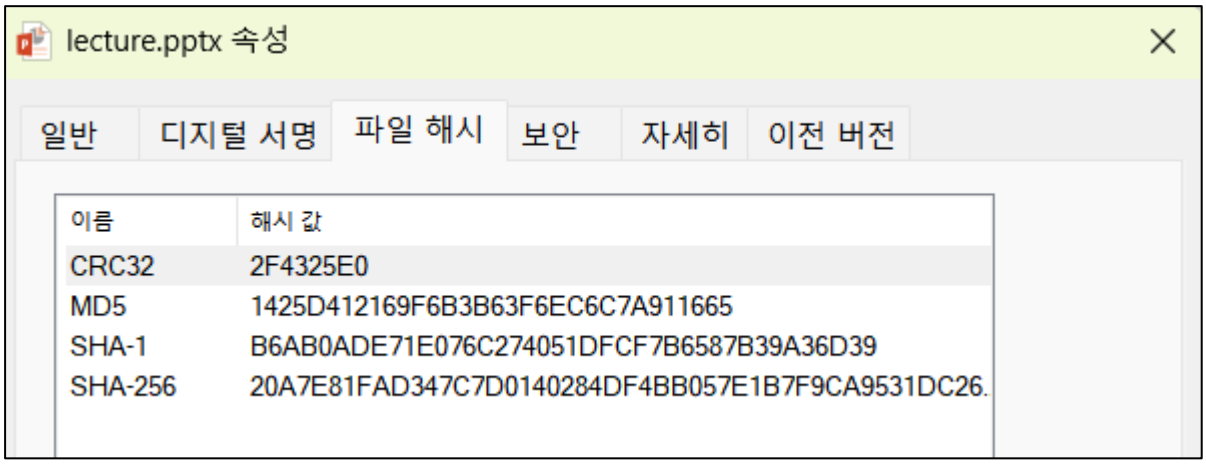

Figure 2. lecture.pptx MD5값

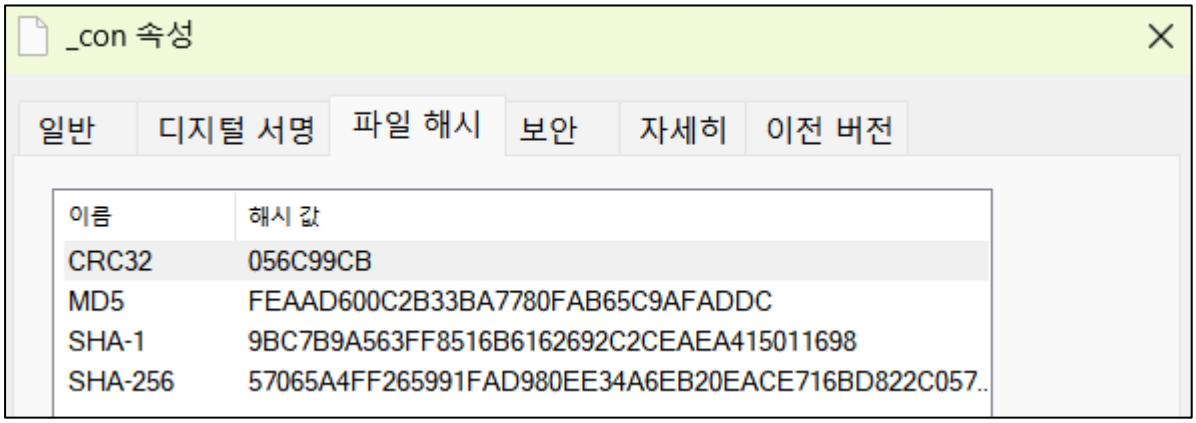

Figure 3. _con 파일의 MD5값

두 파일의 내용을 확인해보면, 다음과 같다. leture.pptx 파일의 경우 Powerpoint 로 열었을 때, 699 개의 슬라이드를 확인할 수 있다.

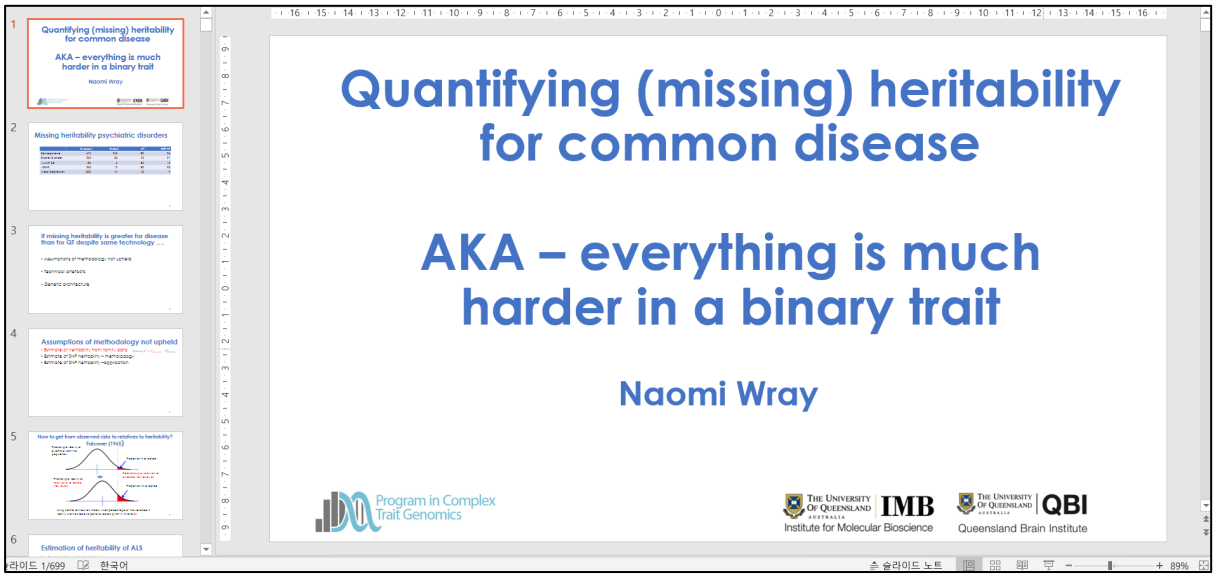

Figure 4. pptx 파일 내용 중 일부

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

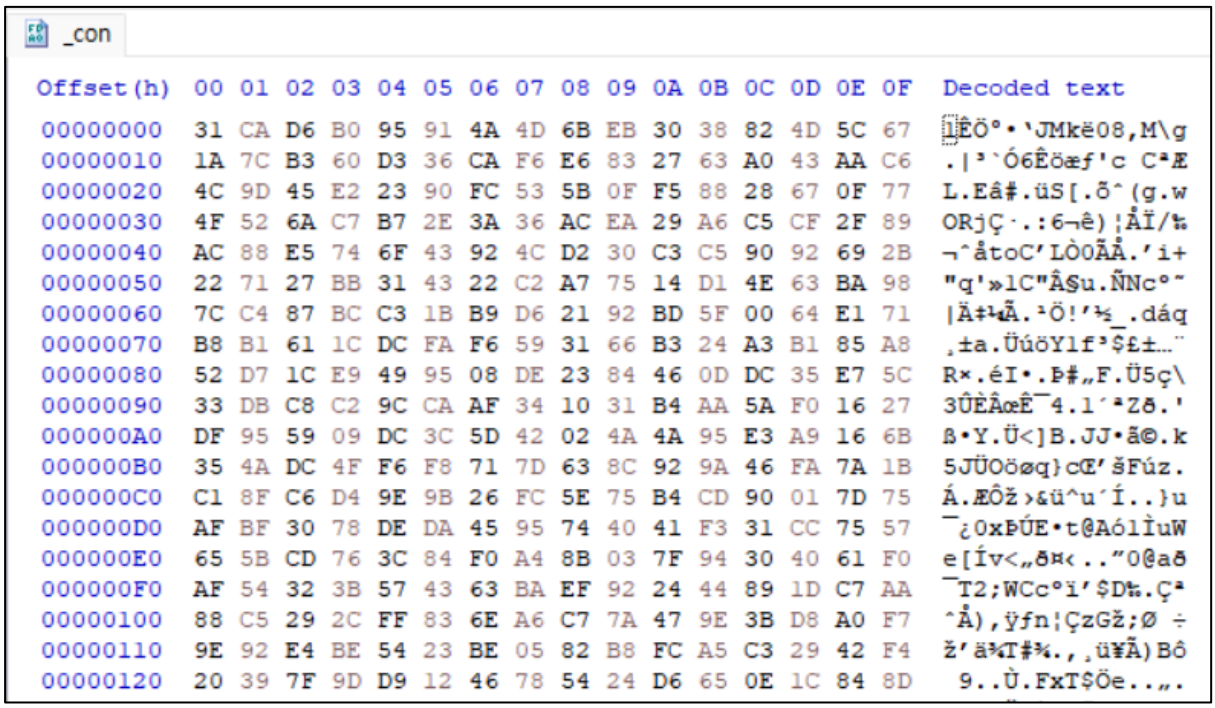

Figure 5. _con 파일 내용 중 일부

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.

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.

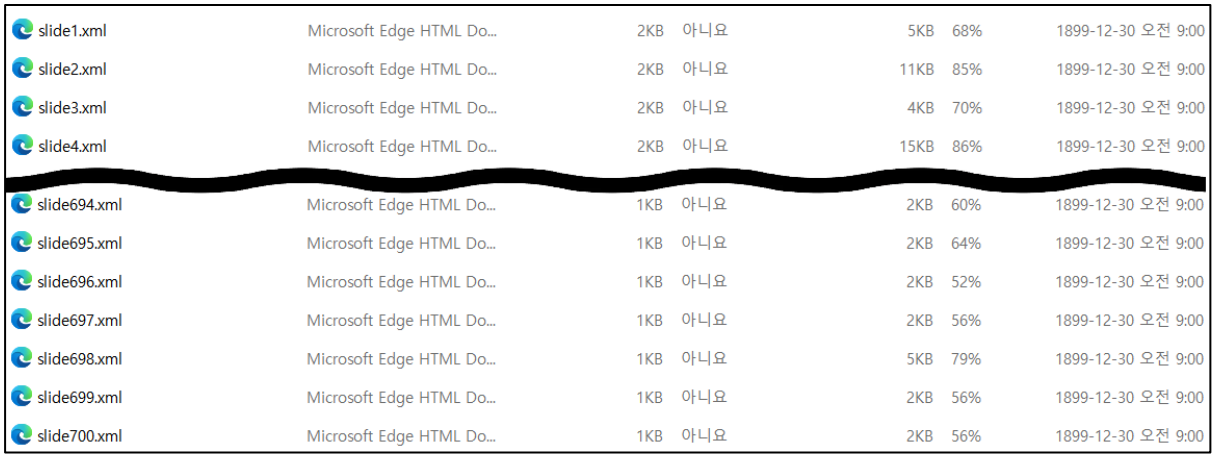

Figure 7. ppt 폴더 내부 slides 값 확인

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

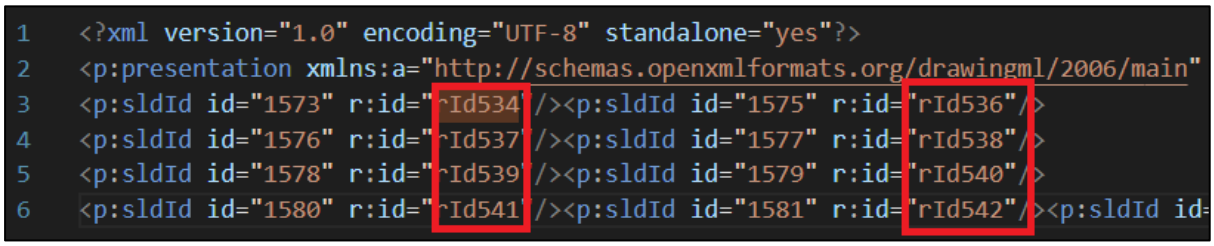

Figure 8. presentation.xml 내부 rid535가 누락된 모습

따라서, 다음 파이썬 코드를 통해 누락되어 있는 페이지를 복구하였다.

```python
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

        # rId534와 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/relations
hips}id')
            existing_ids.append(rid)

        # rId535가 이미 있는지 확인
        if 'rId535' in existing_ids:
            print("rId535가 이미 존재합니다.")
            return False

        # rId534와 rId536의 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/relations
hips}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 rId536를 찾을 수 없습니다.")
            return False

        # rId535 요소 생성
        new_sld_id =
ET.Element('{http://schemas.openxmlformats.org/presentationml/2006/main}sldI
d')
        new_sld_id.set('id', '1574')
        new_sld_id.set('{http://schemas.openxmlformats.org/officeDocument/20
06/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')

        # minidom을 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 PPTX로 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 700개 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 "₩lecture₩ppt₩_rels₩presentation.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.rels 에서 복구된 슬라이드 번호 확인

복구된 512 슬라이드쇼는 다음과 같다.

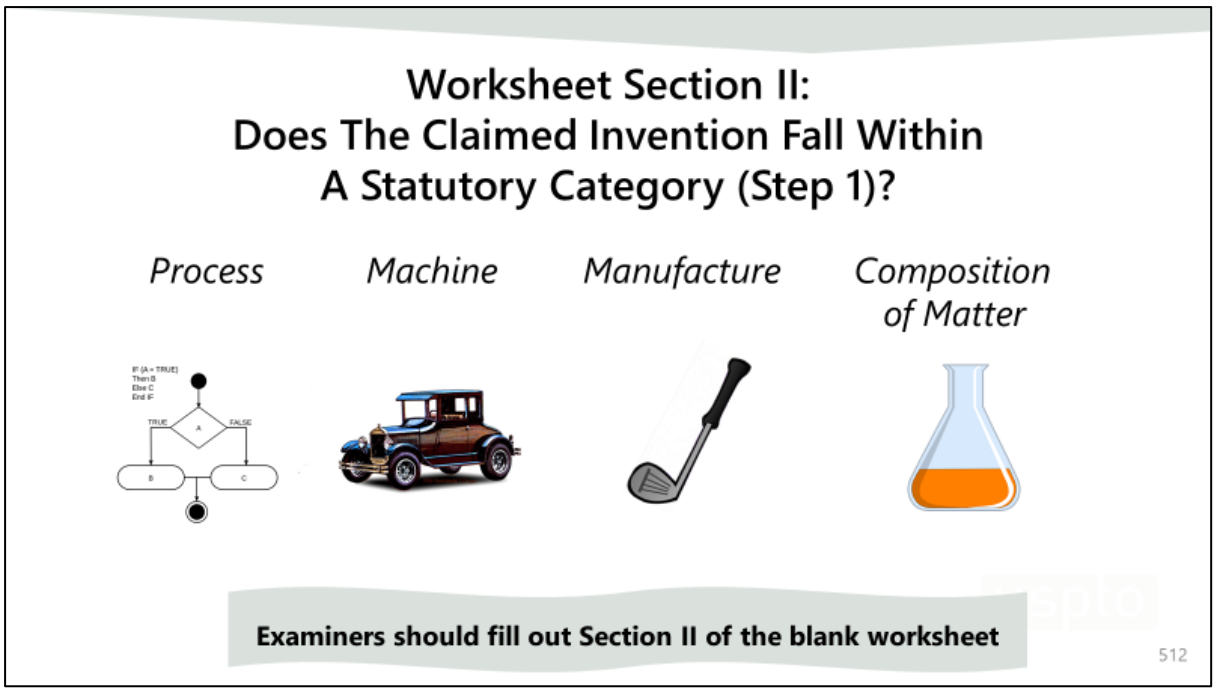

Figure 10. 복구된 512 슬라이드 페이지

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.

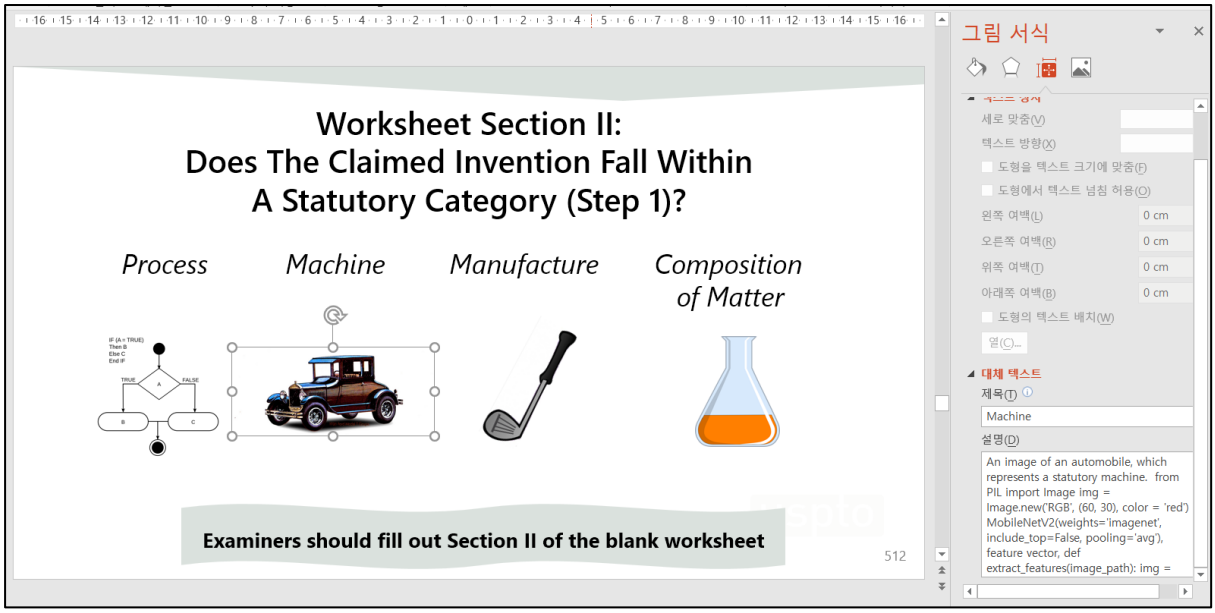

Figure 11. 512 페이지의 숨겨진 코드

숨겨진 코드에는 **이미지 특징 추출 및 암호화 관련 코드**가 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')
model = 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.