FaS
OOXML 데이터 은닉

정보보안암호수학과 20182209 김수빈

OOXML 파일에 정보 은닉

1. [Content_Type].xml에 외부 파일(zip) 은닉

- [Content_Type].xml 파트에는 패키지 내에 존재하는 파트들의 확장자가 명시되어 있다
- 은닉하고자 하는 파일이 zip 확장자를 갖는다면, Fig. 2에 표시 된 부분과 같이 확장자를 넣어 준다.
- 은닉된 파일에 대해 관계 파트에 관계 Id를 할당하고 타깃으로 지정한다.
- 그러면 OOXML은 은닉된 데이터가 있는 문서를 실행해도 정상적인 파일로 인식하게 된다

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Types xmlns="http://schemas.openxmlformats.org/package/2006/content-types">
    <Default Extension="bin"
        ContentType="application/vnd.openxmlformats-officedocument.spreadsheetml.printerSettings"/>
    <Default Extension="png" ContentType="image/png"/>
    <Default Extension="rels" ContentType="application/vnd.openxmlformats-package.relationships+xml"/>
    <Default Extension="xml" ContentType="application/xml"/>

    <Default Extension="zip" ContentType="application/zip"/>

    <Override PartName="/xl/workbook.xml"
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
    <Relationship Id="rId3" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/extended-properties" Target="docProps/app.xml"/>
    <Relationship Id="rId2" Type="http://schemas.openxmlformats.org/package/2006/relationships/metadata/core-properties" Target="docProps/core.xml"/>
    <Relationship Id="rId1" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/officeDocument" Target="xl/workbook.xml"/>
    <Relationship Id="rId4" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/custom-properties" Target="docProps/custom.xml"/>

    <Relationship Id="rId100" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/custom-properties/a" Target="xl/media/hidden.zip"/>
```

Fig. 2. Modified '[Content_Type].xml' part(upper) and modified '.rels' part(lower)

OOXML 파일에 정보 은닉

1. [Content_Type].xml에 외부 파일(zip) 은닉 - 탐지 방법 1번

- 은닉 된 파일에 대해서는 각 파트들이 관계 파트 내에서 타깃으로 존재하는지 확인한다.
- 만약 파트가 타깃으로 존재하지 않는다면, 은닉된 데이터로 판단한다.
- 그 다음으로, 타깃의 타입이 OOXML 표준에 정의 되어 있지 않다면, 이를 은닉된 데이터로 판단한다.

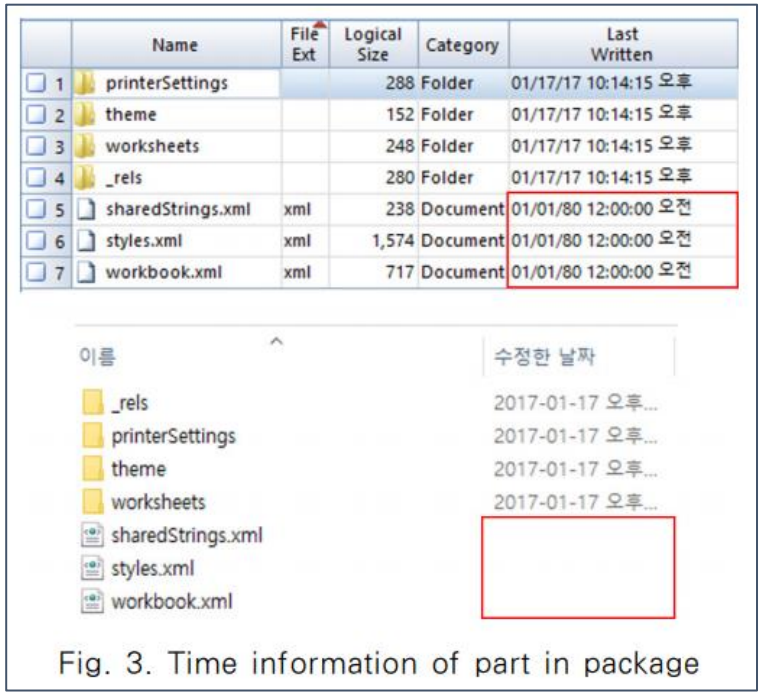
✓ 한계점

- 기밀문서의 확장자를 '.xml'로 변경하여 OOXML 내에 삽입하고,
- 관계 파트에 이를 타깃으로 하는 엘리먼트를 추가하였다면 제시한 탐지 알고리즘을 회피할 수 있다.

OOXML 파일에 정보 은닉

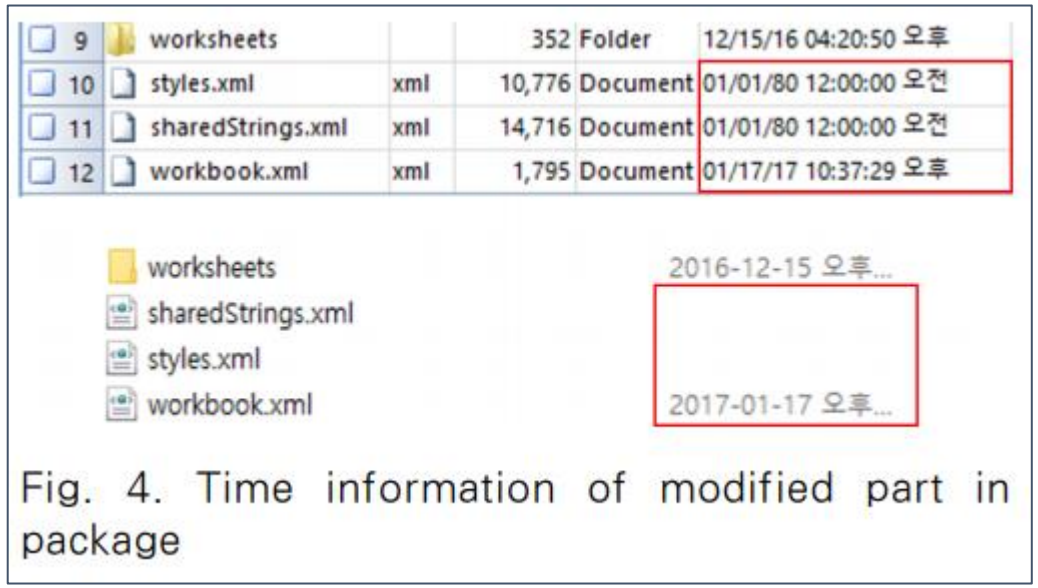
1. [Content_Type].xml에 외부 파일(zip) 은닉 - 탐지 방법 2번

- EnCase로 본 파트의 시간 정보는 '1980년 1월 1일 오전 12시'가 기본 값, 윈도우 탐색기 상에서 확인할 수 있는 시간 정보는 비어있다.
- 하지만 조작 된 파트는 수정한 날짜가 보입니다.



	Name	File Ext	Logical Size	Category	Last Written
1	printerSettings		288	Folder	01/17/17 10:14:15 오후
2	theme		152	Folder	01/17/17 10:14:15 오후
3	worksheets		248	Folder	01/17/17 10:14:15 오후
4	_rels		280	Folder	01/17/17 10:14:15 오후
5	sharedStrings.xml	xml	238	Document	01/01/80 12:00:00 오전
6	styles.xml	xml	1,574	Document	01/01/80 12:00:00 오전
7	workbook.xml	xml	717	Document	01/01/80 12:00:00 오전

Fig. 3. Time information of part in package



9	worksheets		352	Folder	12/15/16 04:20:50 오후
10	styles.xml	xml	10,776	Document	01/01/80 12:00:00 오전
11	sharedStrings.xml	xml	14,716	Document	01/01/80 12:00:00 오전
12	workbook.xml	xml	1,795	Document	01/17/17 10:37:29 오후

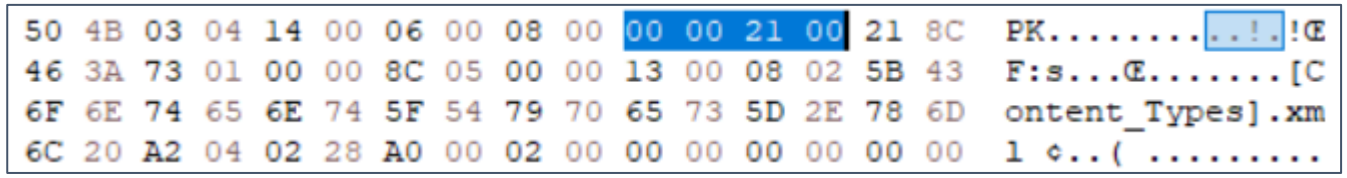
Fig. 4. Time information of modified part in package

OOXML 파일에 정보 은닉

1. [Content_Type].xml에 외부 파일(zip) 은닉 - 탐지 방법 2번

✓ 한계점

- ZIP 아카이브에 대한 구조는 공개되어 있고, HxD를 활용한다면 시간 정보를 초기화할 수 있다.
- DOSTIME과 DOSDATE 각각 두 바이트가 시간 정보를 갖는 영역이다.
- DOSTIME은 초기값인 '00 00'으로, DOSDATE는 초기값인 '21 00'으로 바꾸어 저장하면 파트의 시간 정보를 초기화시킬 수 있다.



OOXML 파일에 정보 은닉

2. 주석 이용하여 은닉

- '.xml'을 확장자로 갖는 파트는 주석을 이용하여 문자열 데이터를 은닉할 수 있다.

2. 주석 이용하여 은닉 - 탐지 방법

- OOXML 내에 기본적으로 생성되는 XML 문서에는 주석이 없다.
- 따라서 '<!—주석 내용-->'이 있는 경우 사용자가 고의적으로 데이터를 삽입한 것으로 판단한다.

OOXML 파일에 정보 은닉

3. 외부 파일의 확장자를 .xml로 변경하여 은닉

- 엑셀(xlsx), 파워포인트(pptx), 워드 (docx) 모두 가능하다
- [Content_Type].xml 파트를 수정 하지 않고 확장자만 변경하여 파일을 삽입하더라도 문서를 여는 데에 아무런 지장이 없다
- 기존의 탐지 방법을 회피하도록 관계 파트에 타깃으로 지정하여도 역시 문제가 발생하지 않는다

OOXML 파일에 정보 은닉

3. 외부 파일의 확장자를 .xml로 변경하여 은닉 – 탐지 방법

- XML 파일의 시작 문자열과 비교해 구분할 수 있다.
- '<?xml version="1.0" encoding="UTF-8" standalone="yes"?>'는 XML 파일의 시작 문자열로, 보통 파일의 헤더와 구분되는 점을 활용하여 외부 파일의 확장자를 수정하여 삽입하였는지 판단하게 된다.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Types  xmlns="http://schemas.openxmlformats.org/package/2006/content-types">
```

OOXML 파일에 정보 은닉

4. 내부 데이터인 시트, 슬라이드 은닉

- 엑셀과 파워포인트에서만 가능함
- 엑셀 → 은닉할 시트에 대한 정보가 있는 ₩xl₩workbook.xml 파트를 수정하여 저장한다.
- 파워포인트 → ₩ppt₩presentation.xml 파트에서 은닉할 데이터와 관련된 파트를 수정하여 저장한다.

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <workbook xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main"
3      xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships"
4      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" mc:Ignorable="x15"
5      xmlns:x15="http://schemas.microsoft.com/office/spreadsheetml/2010/11/main">
                              ⋮
17     <sheets>
18         <sheet name="Sheet1" sheetId="1" r:id="rId1"/>
19         <sheet name="Sheet2" sheetId="2" r:id="rId2"/>
20     </sheets>
                              ⋮
28 </workbook>
```

Fig. 6. 'workbook.xml' part in MS office Excel

4. 내부 데이터인 시트, 슬라이드 은닉 – 탐지 방법

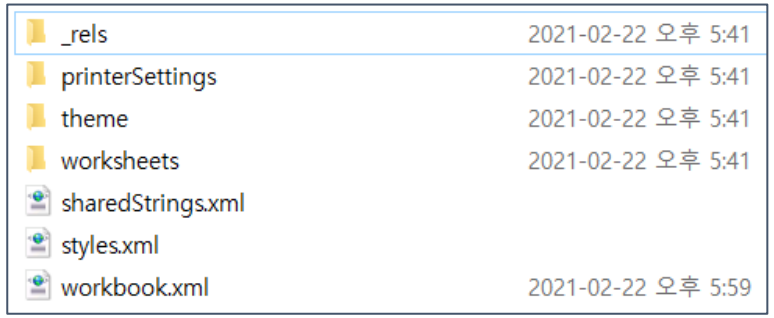
- 엑셀과 파워포인트 각 관계 파트 'workbook.xml.rels', 'presentation.xml.rels' 에서 'target' 을 확인
- 타깃 파트의 수와 실제 파트인 파일의 수가 일치하지 않으면 데이터를 은닉하였다고 판단할 수 있다.

실습 – OOXML 데이터 은닉

1. 시트 숨기기

```
<sheets>
    <sheet  name="년도,논문지" sheetId="1" r:id="rId1"/>
    <sheet  name="논문명" sheetId="2" r:id="rId2"/>
</sheets>
```

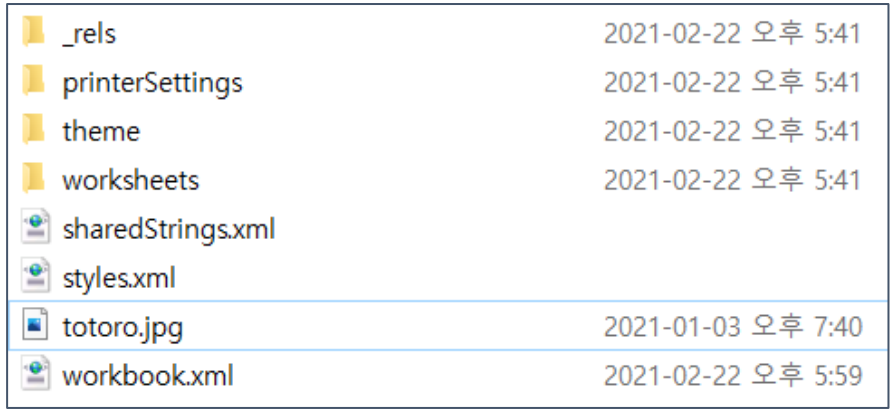
```
<sheets>
    <sheet  name="년도,논문지" sheetId="1" r:id="rId1"/>
</sheets>
```

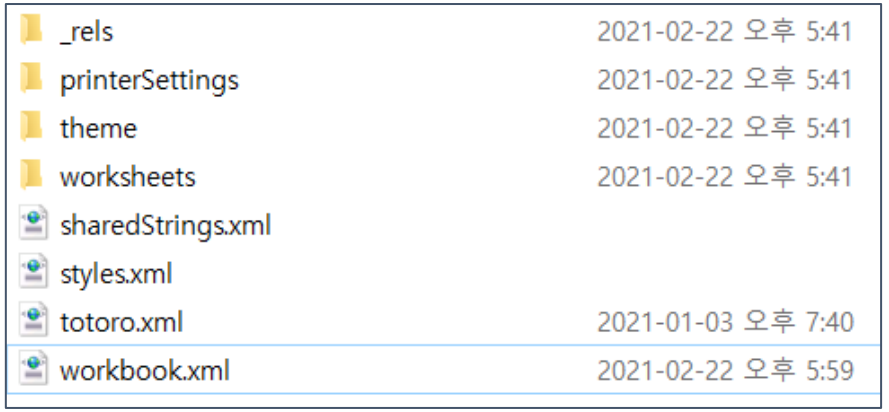


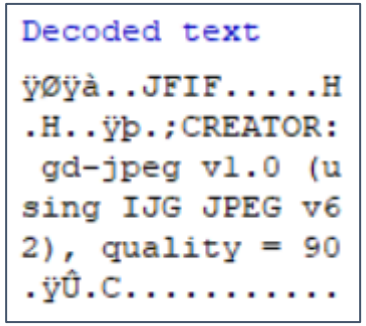
- xl/workbook.xml 파일에서 '논문명' 시트를 참조하는 부분을 삭제 → 수정한 시간이 뜬다.

실습 – OOXML 데이터 은닉

2. 외부 파일 확장자를 .xml로 바꿔 은닉







- totoro.jpg 파일을 totoro.xml 로 확장자를 바꿔 저장
- 원래 xml파일이 아니었기 때문에 '<?xml version="1.0" encoding="UTF-8" standalone="yes"?>'로 시작하지 않는다.

실습 – 파이썬으로 은닉한 데이터 탐지 도구 만들기

1. 수정 시간 이용하여 은닉한 데이터 탐지

```
for i in range(len(LF_sig_offset)):
    # time
    time_offset = LF_sig_offset[i] + 10
    f.seek(time_offset)
    time_hex = f.read(2)

    # date
    date_offset = LF_sig_offset[i] + 12
    f.seek(date_offset)
    date_hex = f.read(2)
```

```
if time_hex != b'\x00\x00' or date_hex != b'\x21\x00':
    if name_hex[-1] != '/':
        hidden_name.append(name_hex)
```

```
print('hidden file using modification time :', hidden_name)
```

- 원래 zip 파싱 프로그램에서 time과 data 찾는 부분 추가
- time과 data가 00 00 21 00 이 아니면 파일이 사용자에 의해 임의로 수정된 것

```
hidden file using modification time : ['xl/totoro.xml', 'xl/workbook.xml']
```

totoro.xml과 workbook.xml이 임의로 수정된 파일인 것을 알 수 있다.

실습 – 파이썬으로 은닉한 데이터 탐지 도구 만들기

2. xml 파일 중에 시작 문자열 이용하여 은닉된 데이터 탐지

```
def xmldata(dataoffset, datalen):
    f.seek(dataoffset)
    data = f.read(datalen)
    data = zlib.decompress(data, -zlib.MAX_WBITS)
    return data
```

```
for i in range(len(name)):
    # external hidden file
    if name[i][-4:] == '.xml':
        xml_data = xmldata(data_offset[i], dataLen[i])
        if xml_data[:55] != b'<?xml version="1.0" encoding="UTF-8" standalone="yes"?>':
            print('external hidden file :', name[i])
```

- 확장자가 .xml인 파일 데이터 구하기
- 데이터 처음 시작 문자열이 '<?xml version="1.0" encoding="UTF-8" standalone="yes"?>'가 아니면 외부 파일의 확장자를 xml으로 변환해 은닉한 것

```
external hidden file : xl/totoro.xml
```

totoro.xml이 확장자를 xml으로 변환해 은닉한 외부 파일인 것을 알 수 있다.

실습 – 파이썬으로 은닉한 데이터 탐지 도구 만들기

3. workbook.xml에서 sheet를 참조하는 개수를 이용하여 은닉된 데이터 탐지

```
if 'xl/worksheets/sheet' in name_hex:
    sheet_cnt += 1
```

```
# internal sheet hidden
if name[i] == 'xl/workbook.xml':
    wb_data = xmldata(data_offset[i], dataLen[i])
    wb_data = wb_data.decode()
    ws = wb_data.find('<sheets>')
    we = wb_data.find('</sheets>', ws)
    sheets = wb_data[ws+8:we]
    seen_sheet_cnt = sheets.count('<sheet')

    if seen_sheet_cnt != sheet_cnt:
        print('internal sheet hidden', sheet_cnt - seen_sheet_cnt, '개')
```

- xl/worksheets/sheet#.xml 파일 개수 세기
- workbook.xml 파일 데이터 구해서, <sheet ~> 개수 세기
- 개수 비교해서 다르면 sheet가 은닉되어 있는 것 확인

```
internal sheet hidden 1 개
```

sheet가 1개 은닉되어 있는 것을 알 수 있다.