analyzing-PDF-metadata

February 12, 2021

1 Analyzing PDF's Metadata with Python

Author: samsar4

1.0.1 A little bit about the internal structure of PDF

- **Objects**: A PDF document is a data structure composed from a small set of basic types of data objects. Sub-clause 7.2, "Lexical Conventions," describes the character set used to write objects and other syntactic elements. Sub-clause 7.3, "Objects," describes the syntax and essential properties of the objects. Sub-clause 7.3.8, "Stream Objects," provides complete details of the most complex data type, the stream object.
- **File structure**: The PDF file structure determines how objects are stored in a PDF file, how they are accessed, and how they are updated. This structure is independent of the semantics of the objects. Sub- clause 7.5, "File Structure," describes the file structure. Sub-clause 7.6, "Encryption," describes a file-level mechanism for protecting a document's contents from unauthorized access.
- **Document structure**: The PDF document structure specifies how the basic object types are used to represent components of a PDF document: pages, fonts, annotations, and so forth. Sub-clause 7.7, "Document Structure," describes the overall document structure; later clauses address the detailed semantics of the components.
- **Content streams**: A PDF content stream contains a sequence of instructions describing the appearance of a page or other graphical entity. These instructions, while also represented as objects, are conceptually distinct from the objects that represent the document structure and are described separately. Sub-clause 7.8, "Content Streams and Resources," discusses PDF content streams and their associated resources.

1.0.2 PyPDF2 module

A Pure-Python library built as a PDF toolkit. - Features - Extracting document information (title, author, …) - Splitting documents page by page - Merging documents page by page - Cropping pages - Merging multiple pages into a single page - Encrypting and decrypting PDF files

Docs: https://pythonhosted.org/PyPDF2/

[7]:
```
import pprint
import PyPDF2
```

`getDocumentInfo()` and `PdfFileReader()` methods:

- Retrieves the PDF file's document information **dictionary**, if it exists.
- Some PDF files use metadata streams instead of docinfo dictionaries, and these metadata streams will not be accessed by this function.

These 2 methods together yields a function that can be used to extract document information dictionary metadata from PDFs!

[8]:
```
def get_doc_info(file_name):
    pp = pprint.PrettyPrinter(indent=2)
    pdf_file = PyPDF2.PdfFileReader(file_name, 'rb')
    docInfo = pdf_file.getDocumentInfo()
    pp.pprint(docInfo)
```

[9]:
```
get_doc_info("teste.pdf")
```

```
{ '/CreationDate': "D:20200330103628-05'00'",
  '/Creator': 'Adobe InDesign 15.0 (Macintosh)',
  '/GTS_PDFXVersion': 'PDF/X-4',
  '/ModDate': "D:20200416192011-07'00'",
  '/Producer': 'Adobe PDF Library 15.0',
  '/Title': 'SEC573_Course-Flyer.indd',
  '/Trapped': '/False'}
```

1.1 Getting XMP metadata

Adobe's **Extensible Metadata Platform (XMP)** is a file labeling technology that lets you embed metadata into files themselves during the content creation process. [+]

`getXmpMetadata()` method:

- This method retrieves XMP (Extensible Metadata Platform) data from the PDF document root.
- The `xmpMethods` list uses the full collection of methods provided by PyPDF2 documentation, that will be used to access the extracted metadata.

[4]:
```
def get_xmp_info(file_name):
    pp = pprint.PrettyPrinter(indent=2)
    pdf_file = PyPDF2.PdfFileReader(file_name, 'rb')
    pdf_xmp = pdf_file.getXmpMetadata()
    xmpMethods = ['custom_properties', 'dc_contributor',
                  'dc_coveragedc_creator', 'dc_date', 'dc_description',
                  'dc_format', 'dc_identifier', 'dc_languagedc_publisher',
                  'dc_relation', 'dc_rights', 'dc_source', 'dc_subject',
                  'dc_title', 'dc_type', 'pdf_keywords', 'pdf_pdfversion',
                  'pdf_producer"', 'xmp_createDate', 'xmp_creatorTool',
                  'xmp_metadataDate', 'xmp_modifyDate', 'xmpmm_documentId',
                  'xmpmm_instanceId']
    xmp = {}
    for i in xmpMethods:
        try:
            xmp[i] = getattr(pdf_xmp,i)
        except:
            xmp[i] = ''
    pp.pprint(xmp)
```

```
[5]: get_xmp_info("teste.pdf")
```

```
{ 'custom_properties': {'GTS_PDFXVersion': 'PDF/X-4'},
  'dc_contributor': [],
  'dc_coveragedc_creator': '',
  'dc_date': [],
  'dc_description': {},
  'dc_format': 'application/pdf',
  'dc_identifier': None,
  'dc_languagedc_publisher': '',
  'dc_relation': [],
  'dc_rights': {},
  'dc_source': None,
  'dc_subject': [],
  'dc_title': {'x-default': 'SEC573_Course-Flyer.indd'},
  'dc_type': [],
  'pdf_keywords': None,
  'pdf_pdfversion': None,
  'pdf_producer"': '',
  'xmp_createDate': datetime.datetime(2020, 3, 30, 15, 36, 28),
  'xmp_creatorTool': 'Adobe InDesign 15.0 (Macintosh)',
  'xmp_metadataDate': datetime.datetime(2020, 4, 17, 2, 20, 11),
  'xmp_modifyDate': datetime.datetime(2020, 4, 17, 2, 20, 11),
  'xmpmm_documentId': 'xmp.id:192f14a8-04ee-4d61-8cbc-58d8b3e24033',
  'xmpmm_instanceId': 'uuid:a432719a-38f6-4541-b5a5-0141a783857c'}
```

1.2 PyPDF2 problems

After running several hundred PDF files through this PyPDF2 based application, it was discovered serious problems with the module that could be useless for serious forensic investigations, leading to inability to extract multiple metadata objects and reporting false negatives.

1.3 Other solutions

1.3.1 Python regex

An alternative low-level technique of carving the PDF binary directly with Python, using the re module

1.3.2 PDF Metadata

Is an automated utility that extracts metadata from PDF file. The utility is used by passing arguments to the script from the command line. As this utility is intended for digital forensics use case, in distinction from PDF parsing libraries (e.g. PyPDF2), which extract only a structured subset of available metadata, PDF Forensics aims to extract all relevant file system and application metadata in their native format. These extracted data are then collected into a structured HTML-formatted report.

PDF Metadata repository: https://gitlab.com/nxl4/pdf-metadata/-/tree/master

1.3.3 References:

- https://www.sans.org/reading-room/whitepapers/forensics/pdf-metadata-extraction-python-38800
- https://www.sans.org/reading-room/whitepapers/privacy/document-metadata-the-silent-killer–32974
- https://pythonhosted.org/PyPDF2/
- https://www.adobe.com/content/dam/acom/en/devnet/pdf/pdfs/pdf_reference_archives/PDFReference.p
- https://gitlab.com/nxl4/pdf-metadata/-/tree/master
- https://resources.infosecinstitute.com/topic/pdf-file-format-basic-structure/