CS7038 - Malware Analysis - Wk09.1
Analysis of PDF Documents

Coleman Kane
kaneca@mail.uc.edu

March 7, 2017

PDF Document Overview

We looked at PDF documents briefly during Week 2.

PDF documents are intended to be a print-representation (or “final copy”) of a document prepared for digital viewing. However, the goal is that the document displayed on-screen is a 100% exact approximation of the visual document you will see if printed. It has roots in the earlier-developed *PostScript* language, but isn’t a fully-compatible reimplementation.

Some interesting features in (most) PDF readers:

- JavaScript (PDFjs, ECMA) interpreter
- Forms UI support (XFA, FDF, XFDF)
- U3D/PRC 3d-model embedded support
- Inline HTML
- Numerous embedded image formats
- PDF-within-PDF
- Encoded/encrypted stream data

PDF Document Structure

PDF documents more or less follows the below structure:

%PDF-N.N	... header data ...	... unused ...
X Y obj	object data	endobj
W Z obj	object data	endobj
...	more object data	...
xref	... xref table ...	... unused ...
trailer	... trailer data ...	startxref NNNN
%%EOF		

Each entity inside of the document is located within one of the indirect objects identified above with the "X Y obj", "Z W obj", etc... declarations.

One of these objects is traditionally the “catalog”, or “root object”.

The *xref table* contains an index of the offsets for each of the indirect objects, from beginning of file.

The *trailer* contains a pointer to the *xref table* as well as a dictionary that defines the catalog, the count of objects in the cross-reference table, and other information that may be specific to the viewer.

PDF Objects

Object data is defined by beginning with the following text (where X and Y are integers):
`X Y obj`

The PDF specification defines a number of data types:

- Boolean values (representing True or False)
- Numbers
- Strings, enclosed with parentheses: `(this is a string)`
- Names, character data beginning with a slash: `/NameVal1`
- Arrays, ordered data enclosed with square brackets:
  `[(Object) (Data) (in) (a) (list)]`
- Dictionaries, name-indexed data, defined with << and >>:
  `<</Val1(This is a string) /Val2[(List) (data)]>>`
- Streams, large blobs of arbitrary data, embedded between `stream` and `endstream` keywords
- Null content

PDF Parser

The `pdf-parser.py` tool in Remnux can be helpful in navigating the PDF document structure.

- Search for data in object: `pdf-parser.py -s mytext file.pdf`
- Search for data in stream: `pdf-parser.py -searchstream=mytext file.pdf`
- List objects and their hashes: `pdf-parser.py -H file.pdf`
- Extract object: `pdf-parser.py -o 1 -d stream.dat file.pdf`
- Extract filtered object: `pdf-parser.py -f -o 1 -d stream.dat file.pdf`
- Parse, extract malformed: `pdf-parser.py -v -x malformed.dat file.pdf`
- Integrate with *yara*: `pdf-parser.py -y, -yarastrings`
- Python code generation: `pdf-parser.py -g example.pdf > example.py`