PDF Analysis and Tools

Analyzing PDF Documents

Look for suspicious keywords

- OpenAction, AA
- JavaScript, JS

Encoded data

Tools and yara rules for exploits

pdfid

Identifies PDF object types and filters

Useful to triage PDF documents

https://blog.didierstevens.com/programs/pdf-tools/

Disadvantages:
It only tells you what is in the document, not where it is

pdfid

- This tool is not a PDF parser, but it will scan a file to look for certain PDF keywords, allowing you to identify PDF documents that contain (for example) JavaScript or execute an action when opened. PDFiD will also handle name obfuscation.
- The idea is to use this tool first to triage PDF documents, and then analyze the suspicious ones with pdf-parser.

pdf-parser

Parses, searches, and extracts data from PDF documents

https://blog.didierstevens.com/programs/pdf-tools/

Options

-s TERM : Search for TERM

-o # : Print object number

-f : Decode data in object

-w : Display raw data from object

You can see the parser in action in this screencast.

```
Usage: pdf-parser.py [options] pdf-file

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -s SEARCH, --search=SEARCH
                        string to search in indirect objects (except streams)
  -f, --filter          pass stream object through filters (FlateDecode only)
  -o OBJECT, --object=OBJECT
                        id of indirect object to select (version independent)
  -r REFERENCE, --reference=REFERENCE
                        id of indirect object being referenced (version
                        independent)
  -w, --raw             raw output for data and filters
  -a, --stats           display stats for pdf document
  -t TYPE, --type=TYPE  type of indirect object to select

  pdf-parser V0.2, use it to parse a PDF document
  Source code put in the public domain by Didier Stevens, no Copyright
  Use at your own risk
  https://DidierStevens.com
```

The stats option display statistics of the objects found in the PDF document. Use this to identify PDF documents with unusual/unexpected objects, or to classify PDF documents. For example, I generated statistics for 2 malicious PDF files, and although they were very different in content and size, the statistics were identical, proving that they used the same attack vector and shared the same origin.

The search option searches for a string in indirect objects (not inside the stream of indirect objects). The search is not case-sensitive, and is susceptible to the obfuscation techniques I documented (as I've yet to encounter these obfuscation techniques in the wild, I decided no to resort to canonicalization).

filter option applies the filter(s) to the stream. For the moment, only FlateDecode is supported (e.g. zlib decompression).

The raw option makes pdf-parser output raw data (e.g. not the printable Python representation).

objects outputs the data of the indirect object which ID was specified. This ID is not version dependent. If more than one object have the same ID (disregarding the version), all these objects will be outputted.

reference allows you to select all objects referencing the specified indirect object. This ID is not version dependent.

type allows you to select all objects of a given type. The type is a Name and as such is case-sensitive and must start with a slash-character (/).

peepdf

Combines multiple tools into one

Finds suspicious objects

Decodes data

JavaScript analysis built-in

http://eternal-todo.com/tools/peepdf-pdf-analysis-tool

Options

-i: Inline mode

-u: Update