Malware Analysis (CS6038)

Week 05.1 Document Analysis

Scott Nusbaum
nusbausa@ucmail.uc.edu

February 12, 2019

University of CINCINNATI

Overview

- Homework
- Walkthrough solutions for the in-class problems
- Document Analysis
  - Adobe PDF
  - Microsoft Office Documents
  - Container Documents

University of CINCINNATI

Homework

- Homework 1:
  - Graded and Submitted to Blackboard
- Homework 2:
  - Due Feb 19, 2019
  - See here for the assignment
  - Hint: The problems from Thursday will solve 95% of the homework
- Homework 3:
  - Assign Feb 14, 2019
  - Due Feb 28, 2019
  - Covers Document Analysis and Windows Artifacts

University of CINCINNATI

In-Class Problem 1

- Demo 1
  - file
  - strings
  - Load in disassembler: IDA, Hopper
    - Find Main Function
      - Use known strings to help
      - Also show where mingw puts.
        - Take the EntryPoint -> jump -> Look for call to exit
  - Load into x32dbg
    - Add break point on Main.
      - Step through

University of CINCINNATI

In-Class Problem 2

- Demo 2
  - file
  - strings
  - Load in disassembler: IDA, Hopper
    - Find Main Function
      - Use known strings to help
      - Also show where mingw puts.
        - Take the EntryPoint -> jump -> Look for call to exit
  - Load into x32dbg
    - Add break point on Main.
      - Step through

University of CINCINNATI

In-Class Problem 3

- Demo 3
  - file
  - Open python file
  - Execute python file
    - Review the similarities and differences of the output

In-Class Problem 4

- Demo 4 -- d4.out
  - file
  - strings
  - Open in editor
    - See the "=" at the end of alphanumeric character string
  - Decode
    - This one is more confusing as I didn't give a hint as to what to do next.
  - Start over with file, strings, etc.
  - Still nothing. Lets try a bruteforce XOR

University of CINCINNATI

In-Class -- Shellcode

- It was discussed on how shellcode can be analyzed by creating a c program that passes execution to the character buffer holding the shellcode.
- Loading this compiled executable into a debugger.
- This saves a lot of time. **USE IT**

University of CINCINNATI

Document Analysis

- Analysis of Word, and PDF documents
  - https://zeltser.com/analyzing-malicious-documents/

University of CINCINNATI

PDF Analysis

- 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 Analysis

- 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.

University of CINCINNATI

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

University of CINCINNATI

PDF-Parser

- The pdf-parser.py tool 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

University of CINCINNATI

Office Documents

- We will focus our efforts on the Microsoft suite of software, though it is notable that the space is diverse, and any one of these can be its own intrusion vector.

University of CINCINNATI

Microsoft Office File Formats

- Generally, there are two data file formats that are of interest to MS Office document malware analysis:
  - Office Open XML (OOXML) Files
    - Basically PKZIP archives with a specially-defined layout. Most office documents since about 2007 are dsitributed using this format (XLSX, DOCX, PPTX, etc.)
  - Compound File Binary (CFB)
    - A binary file specification defined by Microsoft. Older Microsoft Office documents were built up from this format (DOC, XLS, PPT).
    - Since 2007, it is still frequently used to embed Microsoft-specific binary data structures within documents and applications.
- Latest CFB file specification
- Latest Office Open XML specifications (ISO/IEC 29500-1:2016, 29500-2:2012, 29500-3:2015, 29500-4:2008)

University of CINCINNATI

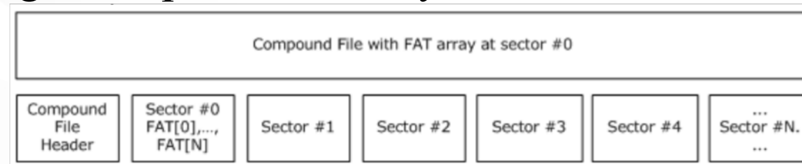
CFB File Format

- The CFB file format is a chunked data format,
  - The file is divided into sectors,
  - There exist file allocation tables that each define an array of pointers to other file locations that map blocks in the file to their ordering within a data stream.
- This organizational model creates a file structure where whole data streams (such as images, sub-documents, videos, content, embedded fonts, macros, etc...) are not guaranteed to exist contiguous within the file.
- There exist a number of utilities that are useful for navigating this structure:
  - https://www.decalage.info/python/oletools
  - https://github.com/unixfreak0037/officeparser
  - https://poi.apache.org/ - Java API for Office Documents

University of CINCINNATI

CFB Sectors

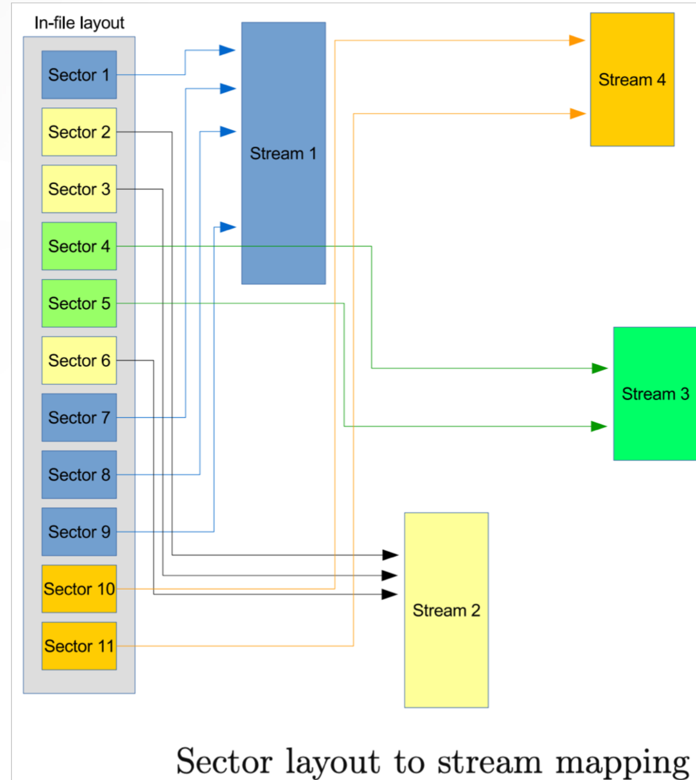
- The first sector of the file contains the CFB header, which is where all of the information defining the top-level file layout



- Almost exclusively, sector sizes are defined to be 512 bytes (0x200 hex), which is consistent with most common OS filesystems as well.

University of CINCINNATI

CFB Streams



Sector layout to stream mapping

University of CINCINNATI

OLE Tools

- Documentation on the following site:
  - olebrowse: A GUI browser enabling you to navigate, view and extract streams. Very basic.
  - oledir: Dump the stream directory of the document
  - olemap: Dump the sector mappings (allocation) of a file
  - olemeta: Dump metadata about the document
  - olevba: Dump VBA macros from files

University of CINCINNATI

OOXML Layout

Can be extracted through the unzip utilities.

```
Archive:  test-doc.docx
    testing: _rels/.rels                   OK
    testing: word/document.xml             OK
    testing: word/styles.xml               OK
    testing: word/_rels/document.xml.rels    OK
    testing: word/settings.xml             OK
    testing: word/media/image1.jpeg        OK
    testing: word/fontTable.xml            OK
    testing: docProps/app.xml              OK
    testing: docProps/core.xml             OK
    testing: [Content_Types].xml           OK
```

University of CINCINNATI

Macros

- Microsoft Office supports executable scripts embedded within documents.
  - A common language used for this is Visual Basic for Applications (VBA).
  - Similar to PDFjs that we discussed earlier, this language is a derivative of Visual Basic
  - Has special hooks into the Office environment and the current (and linked) documents.
- An example macro is available here
- Macros can be used to execute arbitrary code, without relying upon exploits that intend to break parsing of the document. Some examples:
  - http://blog.fortinet.com/2017/03/08/microsoft-excel-files-increasingly-used-to-spread-malware
  - https://blogs.sophos.com/2015/09/28/why-word-malware-is-basic/
  - http://www.kahusecurity.com/2015/malicious-word-macro-caught-using-sneaky-trick/

University of CINCINNATI

Container Documents

- A file that contains other files
  - Zip
  - GunZip
  - Microsoft OOXML format (docx)
  - Androids Apk
  - Java Jar

University of CINCINNATI