ANALYZING MALICIOUS DOCUMENTS

This cheat sheet outlines tips and tools for reverse-engineering malicious documents, such as Microsoft Office (DOC, XLS, PPT) and Adobe Acrobat (PDF) files.

General Approach

1. Locate potentially malicious embedded code, such as shellcode, VBA macros or JavaScript.
2. Extract suspicious code from the file.
3. If relevant, disassemble and/or debug shellcode.
4. If relevant, deobfuscate and examine JavaScript, ActionScript, or VB macro code.
5. Understand next steps in the infection chain.

Microsoft Office Binary File Format Notes

Structured Storage (OLE SS) defines a file system inside the binary Microsoft Office file.

Data can be "storage" (folder) and "stream" (file).

Excel stores data inside the "workbook" stream.

PowerPoint stores data inside the "PowerPoint Document" stream.

Word stores data inside various streams.

Tools for Analyzing Microsoft Office Files

OfficeMalScanner locates shellcode and VBA macros from MS Office (DOC, XLS, and PPT) files.

MalHost-Setup extracts shellcode from a given offset in an MS Office file and embeds it an EXE file for further analysis. (Part of OfficeMalScanner)

Offvis shows raw contents and structure of an MS Office file, and identifies some common exploits.

Hachoir-urwid can navigate through the structure of binary Office files and view stream contents.

Office Binary Translator converts DOC, PPT, and XLS files into Open XML files (includes BiffView tool).

pyOLEScanner.py can examine and decode some aspects of malicious binary Office files.

FileHex (not free) and FileInsight hex editors can parse and edit OLE structures.

Useful MS Office Analysis Commands

`OfficeMalScanner` *`file.doc`* `scan brute`	Locate shellcode, OLE data, PE files in *file.doc*
`OfficeMalScanner` *`file.doc`* `info`	Locate VB macro code in *file.doc* (no XML files)
`OfficeMalScanner` *`file.docx`* `inflate`	Decompress *file.docx* to locate VB code (XML files)
`MalHost-Setup` *`file.doc out.exe 0x4500`*	Extract shellcode from *file.doc*'s offset 0x4500 and create it as *out.exe*

Adobe PDF File Format Notes

A PDF File is comprised of header, objects, cross-reference table (to locate objects), and trailer.

"/OpenAction" and "/AA" (Additional Action) specifies the script or action to run automatically.

"/Names", "/AcroForm", "/Action" can also specify and launch scripts or actions.

"/JavaScript" specifies JavaScript to run.

"/GoTo*" changes the view to a specified destination within the PDF or in another PDF file.

"/Launch" launches a program or opens a document.

"/URI" accesses a resource by its URL.

"/SubmitForm" and "/GoToR" can send data to URL.

"/RichMedia" can be used to embed Flash in PDF.

"/ObjStm" can hide objects inside an Object Stream.

Be mindful of obfuscation with hex codes, such as "/JavaScript" vs. "/J#61vaScript". (See examples)

Tools for Analyzing Adobe PDF Files

PDFiD identifies PDFs that contain strings associated with scripts and actions.

PDF-parser and Origami's pdfwalker examine the structure and decode contents of PDF files.

Origami's pdfextract and Jsunpackn's pdf.py extract JavaScript from PDF files.

PDF Stream Dumper combines many PDF analysis tools under a single graphical user interface.

Peepdf and Origami's pdfsh offer an interactive command-line shell for examining PDF files.

PDF X-RAY Lite creates an HTML report containing decoded PDF file structure and contents.

SWF mastah extracts SWF objects from PDF files.

Pyew includes commands for examining and decoding structure and content of PDF files.

Useful PDF Analysis Commands

`pdfid.py` *`file.pdf`*	Locate script and action-related strings in *file.pdf*
`pdf-parser.py` *`file.pdf`*	Show *file.pdf*'s structure to identify suspect elements
`pdf-parser.py --object` *`id file.pdf`*	Display contents of object *id* in *file.pdf*. Add "--filter --raw" to decode the object's stream.
`pdfextract` *`file.pdf`*	Extract JavaScript embedded in *file.pdf* and save it to *file.dump*.
`pdf.py` *`file.pdf`*	Extract JavaScript embedded in *file.pdf* and save it to *file.pdf.out*.
`swf_mastah.py –f` *`file.pdf –o out`*	Extract PDF objects from *file.pdf* into the *out* directory.

Additional PDF Analysis Tools

Malzilla and SpiderMonkey can help deobfuscate JavaScript embedded in malicious PDF files.

Wepawet, Jsunpack, VirusTotal and sandbox tools can analyze some aspects of malicious PDF files.

ExeFilter can filter scripts from Office and PDF files.

References

Adobe Portable Document Format (PDF) Reference

Physical and Logical Structure of PDF Files

Analyzing Targeted Attacks with Office Docs (video)

MSOffice Malware with OfficeMalScanner (follow-up)

PDF Security Analysis and Malware Threats

Reverse-Engineering Malware cheat sheet

REMnux Linux distribution for malware analysis.

Authored by Lenny Zeltser with feedback and contributions from Pedro Bueno, Frank Boldewin, and Didier Stevens.

This and other malware analysis topics are covered in Lenny's Reverse-Engineering Malware (REM) course, which he teaches at SANS Institute—for details visit LearnREM.com.