**Analysis Malicious Documents Cheat Sheet**

**PDF Analysis**

**Risky PDF Keywords**

|  |  |
| --- | --- |
| /OpenAction and /AA specify the script or action to run automatically | /JavaScript, /JS, /AcroForm, and /XFA can specify JavaScript to run |
| /URI accesses a resource by its URL, perhaps for phishing | /SubmitForm and /GoToR can send data to URL. |
| /RichMedia can be used to embed Flash in a PDF. | /ObjStm can hide objects inside an object stream |
| /XObject can embed an image for phishing. |

**PDF Tool**

|  |  |
| --- | --- |
| **Tool** | **Use it for** |
| pdfid.py *file.pdf* -n | Display risky keywords present in file *file.pdf*. |
| pdf-parser.py *file.pdf* -a | Show stats about keywords. Add “-O” to include object streams. |
| pdf-parser.py sample\_doc.pdf -s Javascript | Display risky keywords present in file *file.pdf*. |
| pdf-parser.py ---object 3 -f -w -d savefile badpdf.pdf | It extracts obj data |
| pdfextract important.pdf | It extracts all file, script data and store folder |
| PDF Stream Dumper | It analysis all stream and shellcode with GUI |
| Peepdf -I -f file.pdf | Display inside pdf as interactive mode |
| Pdf> object id | It displays specific object |
| Pdf> js\_analyse object id > save to file | It decodes js obfuscate code |
| Pdf> stream id > save to file | It decodes and apply filter and save to file |
| Pdf> rawstream 13 > shellcode. bin | It extracts exact byte to analysis bin manual |
| pdfextract | Extracts binary resources of a document (images, scripts, fonts, etc.). |

**Shellcode Tool**

|  |  |
| --- | --- |
| xorsearch -W -d 3 *file.bin* | Locate shellcode patterns inside the binary file *file.bin*. |
| scdbgc /f *file.bin* | Emulate execution of shellcode in file.bin. Use “/off” to specify offset. |
| runsc32 -f *file.bin*-n | Execute shellcode in *file.bin* to observe behavior in an isolated lab. |
| base64dump.py *file.txt* | Convert numbers that represent characters in *file* to a string. |
| numbers-to-string.py *file* | Convert numbers that represent characters in *file* to a string |
| Cscript,spidermonkey tool | help deobfuscate JavaScript that you extract from document files by running as simulation. |

**Analysis Microsoft Docs**

|  |  |
| --- | --- |
| **Tool** | **Use it for** |
| Exiftool file.doc | It show metadata of the file |
| Olevba -c file.doc > file.vba | Extract vba macros code |
| Olevba –deobf –reveal file.vba >file\_deobf.vba | Deobfuscated macro code |
| oleobj | to extract embedded objects from OLE files. |
| zipdump.py -y /index.yar obj3 | It dump zip data hidden and scan with yara |
| zipdump.py -W vir 749048.docm | Dump all |
| Oledump.py -y /index.yar obj3  Oledump.py vba.bin -s index -v | It dump data hidden and scan with yara |
| Vmonkey file.doc | Emulate the vba code to detect malware |
| rtfobj.py *file.rtf* | Extract objects embedded into RTF *file.rtf* |
| xlmdeobfuscator --file *file.xlsm* | Deobfuscate XLM (Excel 4) macros in *file.xlsm* |
| AMSIScriptContentRetrieval.ps1 | observe Microsoft Office execute macros |
| OfficeMalScanner | Analyze office documents Office 2007 (doc vs docx) |
| Media-Extractor | application to extract packed media in Microsoft Office files (e.g. Word, PowerPoint or Excel documents), as well as in common archive files (e.g. zip, 7z, tar) |
| msodde | script to parse MS Office documents (e.g. Word, Excel, RTF, XML), to detect and extract **DDE links** such as **DDEAUTO**, that have been used to run malicious commands to deliver malware |