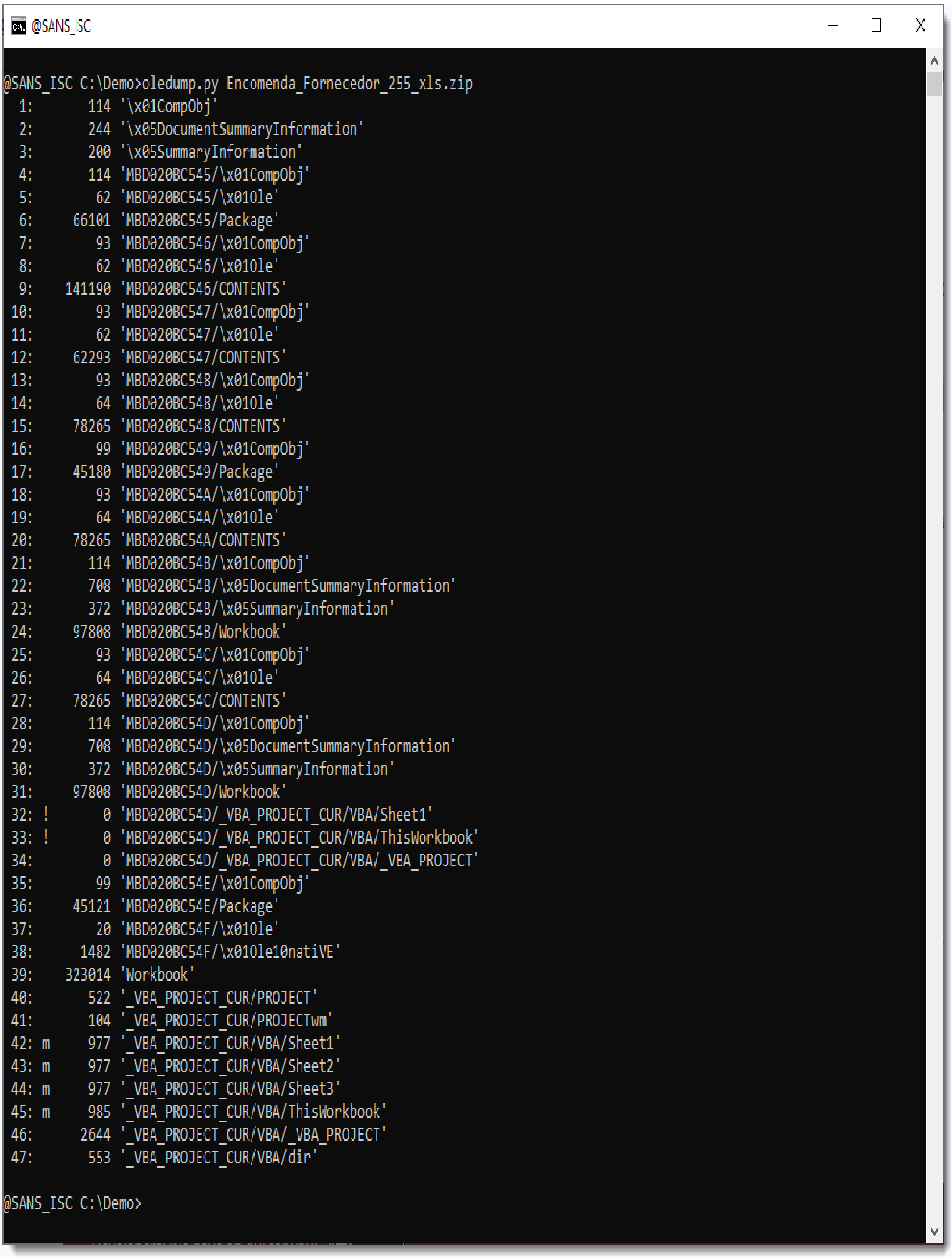
Extracting Multiple Streams From OLE Files

Published: 2023-03-29
Last Updated: 2023-03-29 19:35:43 UTC
by Didier Stevens (Version: 1)

0 comment(s)

Reader Martin asks us for some help extracting embedded content from a submitted malicious document.

These are the streams inside the document, listed with oledump.py:



@SANS_ISC

```
@SANS_ISC C:\Demo>oledump.py Encomenda_Fornecedor_255_xls.zip
  1:       114 '\x01CompObj'
  2:       244 '\x05DocumentSummaryInformation'
  3:       200 '\x05SummaryInformation'
  4:       114 'MBD020BC545/\x01CompObj'
  5:        62 'MBD020BC545/\x01Ole'
  6:     66101 'MBD020BC545/Package'
  7:        93 'MBD020BC546/\x01CompObj'
  8:        62 'MBD020BC546/\x01Ole'
  9:    141190 'MBD020BC546/CONTENTS'
 10:        93 'MBD020BC547/\x01CompObj'
 11:        62 'MBD020BC547/\x01Ole'
 12:     62293 'MBD020BC547/CONTENTS'
 13:        93 'MBD020BC548/\x01CompObj'
 14:        64 'MBD020BC548/\x01Ole'
 15:     78265 'MBD020BC548/CONTENTS'
 16:        99 'MBD020BC549/\x01CompObj'
 17:     45180 'MBD020BC549/Package'
 18:        93 'MBD020BC54A/\x01CompObj'
 19:        64 'MBD020BC54A/\x01Ole'
 20:     78265 'MBD020BC54A/CONTENTS'
 21:       114 'MBD020BC54B/\x01CompObj'
 22:       708 'MBD020BC54B/\x05DocumentSummaryInformation'
 23:       372 'MBD020BC54B/\x05SummaryInformation'
 24:     97808 'MBD020BC54B/Workbook'
 25:        93 'MBD020BC54C/\x01CompObj'
 26:        64 'MBD020BC54C/\x01Ole'
 27:     78265 'MBD020BC54C/CONTENTS'
 28:       114 'MBD020BC54D/\x01CompObj'
 29:       708 'MBD020BC54D/\x05DocumentSummaryInformation'
 30:       372 'MBD020BC54D/\x05SummaryInformation'
 31:     97808 'MBD020BC54D/Workbook'
 32: !       0 'MBD020BC54D/_VBA_PROJECT_CUR/VBA/Sheet1'
 33: !       0 'MBD020BC54D/_VBA_PROJECT_CUR/VBA/ThisWorkbook'
 34:         0 'MBD020BC54D/_VBA_PROJECT_CUR/VBA/_VBA_PROJECT'
 35:        99 'MBD020BC54E/\x01CompObj'
 36:     45121 'MBD020BC54E/Package'
 37:        20 'MBD020BC54F/\x01Ole'
 38:      1482 'MBD020BC54F/\x01Ole10natiVE'
 39:    323014 'Workbook'
 40:       522 '_VBA_PROJECT_CUR/PROJECT'
 41:       104 '_VBA_PROJECT_CUR/PROJECTwm'
 42: m     977 '_VBA_PROJECT_CUR/VBA/Sheet1'
 43: m     977 '_VBA_PROJECT_CUR/VBA/Sheet2'
 44: m     977 '_VBA_PROJECT_CUR/VBA/Sheet3'
 45: m     985 '_VBA_PROJECT_CUR/VBA/ThisWorkbook'
 46:      2644 '_VBA_PROJECT_CUR/VBA/_VBA_PROJECT'
 47:       553 '_VBA_PROJECT_CUR/VBA/dir'

@SANS_ISC C:\Demo>
```

The streams to extract are those where the stream name includes Package, CONTENTS, ...

This can be done with oledump as follows: oledump.py -s 6 -d sample.vir > extracted.vir

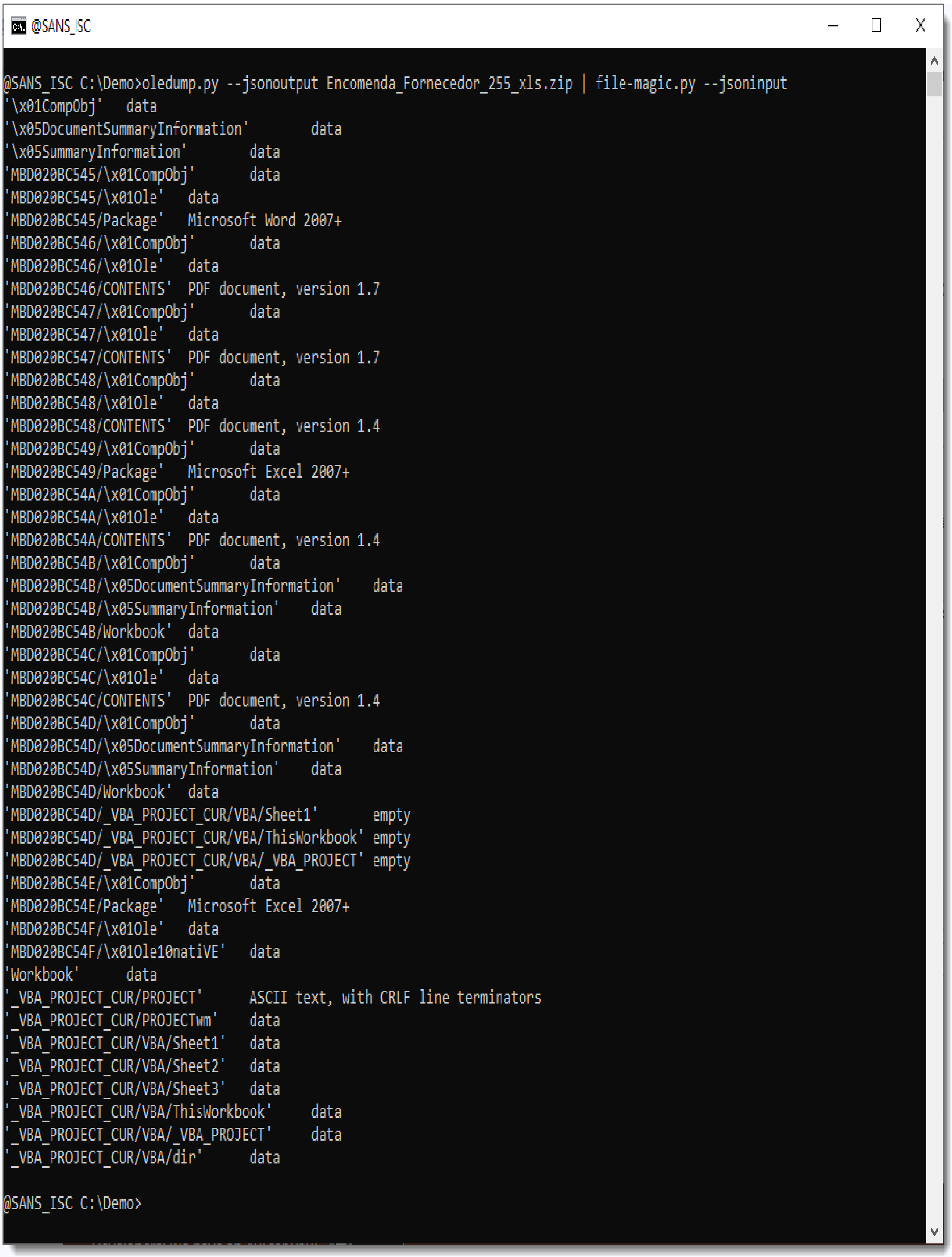
-s 6 selects stream 6, and -d produces a binary dump which is written to a file via file redirection (>).

This has to be repeated for every stream that could be interesting.

But I also have another method, that involves less repeated commands.

First, we let oledump.py analyze the file, and produce JSON output. This JSON output contains all the streams (id, name and content) and can be consumed by other tools I make, like file-magic.py, a tool to identify files based on their content.

Like this:



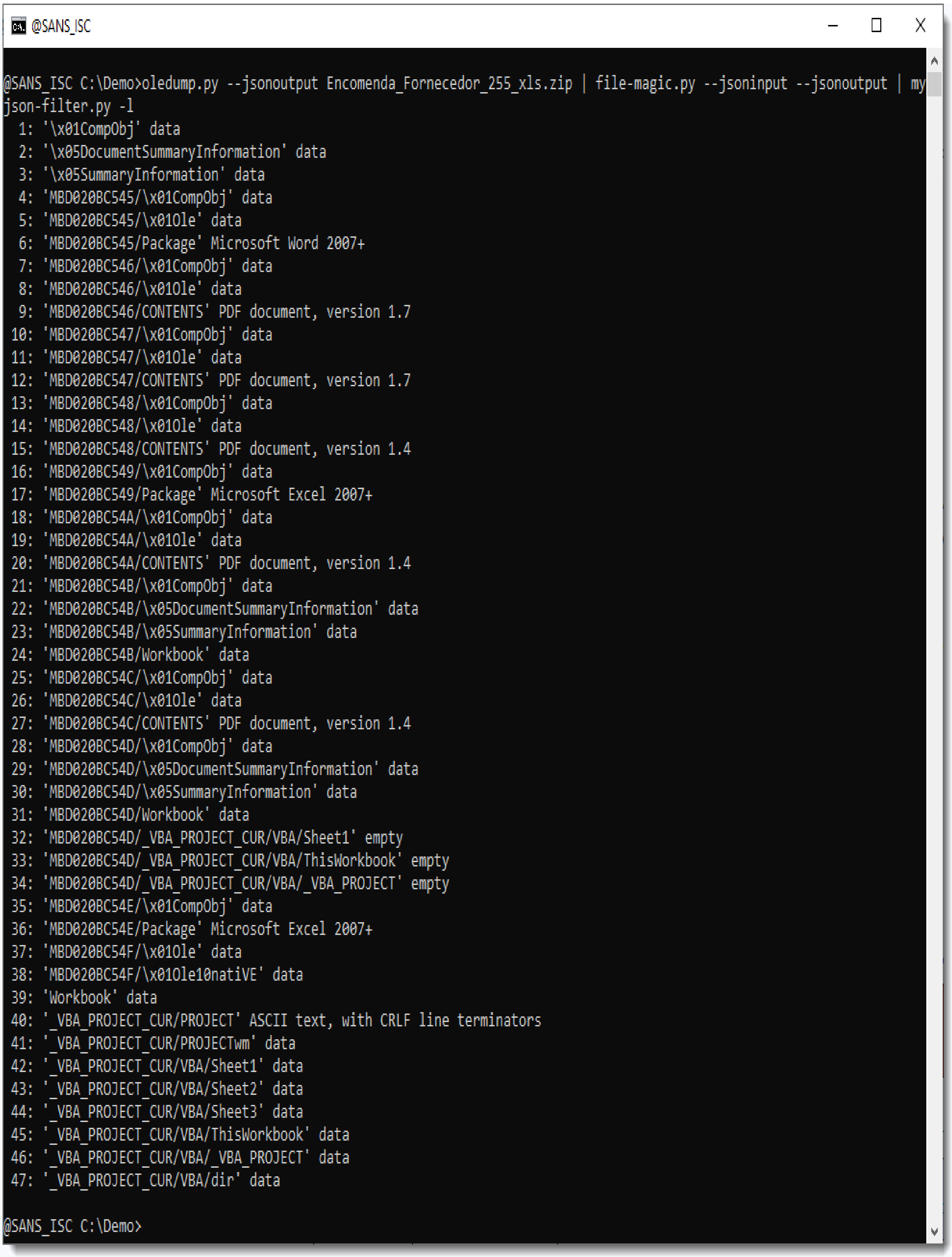
```
@SANS_ISC

@SANS_ISC C:\Demo>oledump.py --jsonoutput Encomenda_Fornecedor_255_xls.zip | file-magic.py --jsoninput
'\x01CompObj'   data
'\x05DocumentSummaryInformation'        data
'\x05SummaryInformation'        data
'MBD020BC545/\x01CompObj'       data
'MBD020BC545/\x01Ole'   data
'MBD020BC545/Package'   Microsoft Word 2007+
'MBD020BC546/\x01CompObj'       data
'MBD020BC546/\x01Ole'   data
'MBD020BC546/CONTENTS'  PDF document, version 1.7
'MBD020BC547/\x01CompObj'       data
'MBD020BC547/\x01Ole'   data
'MBD020BC547/CONTENTS'  PDF document, version 1.7
'MBD020BC548/\x01CompObj'       data
'MBD020BC548/\x01Ole'   data
'MBD020BC548/CONTENTS'  PDF document, version 1.4
'MBD020BC549/\x01CompObj'       data
'MBD020BC549/Package'   Microsoft Excel 2007+
'MBD020BC54A/\x01CompObj'       data
'MBD020BC54A/\x01Ole'   data
'MBD020BC54A/CONTENTS'  PDF document, version 1.4
'MBD020BC54B/\x01CompObj'       data
'MBD020BC54B/\x05DocumentSummaryInformation'    data
'MBD020BC54B/\x05SummaryInformation'    data
'MBD020BC54B/Workbook'  data
'MBD020BC54C/\x01CompObj'       data
'MBD020BC54C/\x01Ole'   data
'MBD020BC54C/CONTENTS'  PDF document, version 1.4
'MBD020BC54D/\x01CompObj'       data
'MBD020BC54D/\x05DocumentSummaryInformation'    data
'MBD020BC54D/\x05SummaryInformation'    data
'MBD020BC54D/Workbook'  data
'MBD020BC54D/_VBA_PROJECT_CUR/VBA/Sheet1'       empty
'MBD020BC54D/_VBA_PROJECT_CUR/VBA/ThisWorkbook' empty
'MBD020BC54D/_VBA_PROJECT_CUR/VBA/_VBA_PROJECT' empty
'MBD020BC54E/\x01CompObj'       data
'MBD020BC54E/Package'   Microsoft Excel 2007+
'MBD020BC54F/\x01Ole'   data
'MBD020BC54F/\x01Ole10natiVE'   data
'Workbook'      data
'_VBA_PROJECT_CUR/PROJECT'      ASCII text, with CRLF line terminators
'_VBA_PROJECT_CUR/PROJECTwm'    data
'_VBA_PROJECT_CUR/VBA/Sheet1'   data
'_VBA_PROJECT_CUR/VBA/Sheet2'   data
'_VBA_PROJECT_CUR/VBA/Sheet3'   data
'_VBA_PROJECT_CUR/VBA/ThisWorkbook'     data
'_VBA_PROJECT_CUR/VBA/_VBA_PROJECT'     data
'_VBA_PROJECT_CUR/VBA/dir'      data

@SANS_ISC C:\Demo>
```

file-magic.py identified the content of each stream: data, Word, PDF, ...

We can now let file-magic.py produce JSON output, that can then be filtered by another tool: myjson-filter.py:

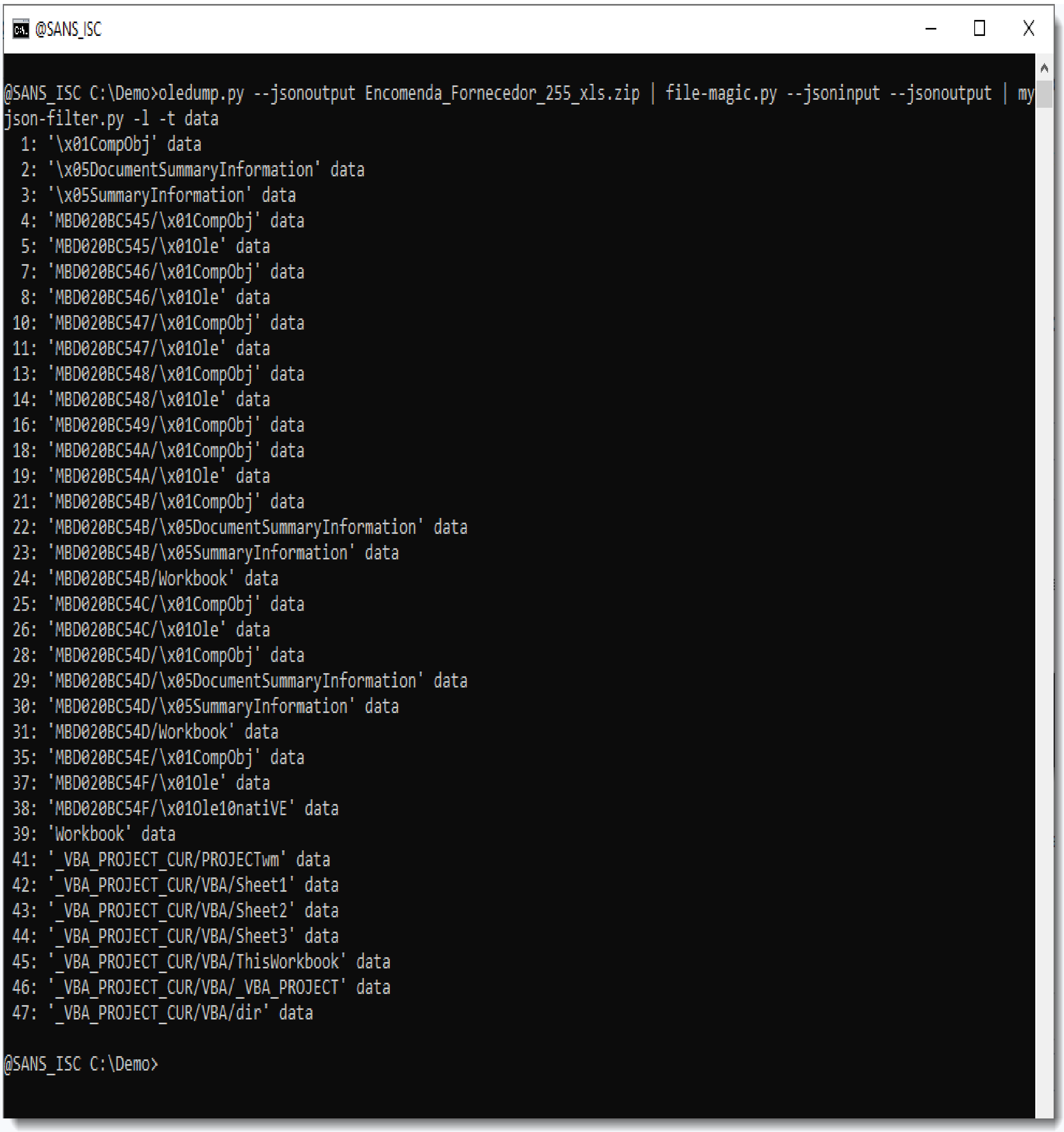

@SANS_ISC

```
@SANS_ISC C:\Demo>oledump.py --jsonoutput Encomenda_Fornecedor_255_xls.zip | file-magic.py --jsoninput --jsonoutput | my
json-filter.py -l
  1: '\x01CompObj' data
  2: '\x05DocumentSummaryInformation' data
  3: '\x05SummaryInformation' data
  4: 'MBD020BC545/\x01CompObj' data
  5: 'MBD020BC545/\x01Ole' data
  6: 'MBD020BC545/Package' Microsoft Word 2007+
  7: 'MBD020BC546/\x01CompObj' data
  8: 'MBD020BC546/\x01Ole' data
  9: 'MBD020BC546/CONTENTS' PDF document, version 1.7
 10: 'MBD020BC547/\x01CompObj' data
 11: 'MBD020BC547/\x01Ole' data
 12: 'MBD020BC547/CONTENTS' PDF document, version 1.7
 13: 'MBD020BC548/\x01CompObj' data
 14: 'MBD020BC548/\x01Ole' data
 15: 'MBD020BC548/CONTENTS' PDF document, version 1.4
 16: 'MBD020BC549/\x01CompObj' data
 17: 'MBD020BC549/Package' Microsoft Excel 2007+
 18: 'MBD020BC54A/\x01CompObj' data
 19: 'MBD020BC54A/\x01Ole' data
 20: 'MBD020BC54A/CONTENTS' PDF document, version 1.4
 21: 'MBD020BC54B/\x01CompObj' data
 22: 'MBD020BC54B/\x05DocumentSummaryInformation' data
 23: 'MBD020BC54B/\x05SummaryInformation' data
 24: 'MBD020BC54B/Workbook' data
 25: 'MBD020BC54C/\x01CompObj' data
 26: 'MBD020BC54C/\x01Ole' data
 27: 'MBD020BC54C/CONTENTS' PDF document, version 1.4
 28: 'MBD020BC54D/\x01CompObj' data
 29: 'MBD020BC54D/\x05DocumentSummaryInformation' data
 30: 'MBD020BC54D/\x05SummaryInformation' data
 31: 'MBD020BC54D/Workbook' data
 32: 'MBD020BC54D/_VBA_PROJECT_CUR/VBA/Sheet1' empty
 33: 'MBD020BC54D/_VBA_PROJECT_CUR/VBA/ThisWorkbook' empty
 34: 'MBD020BC54D/_VBA_PROJECT_CUR/VBA/_VBA_PROJECT' empty
 35: 'MBD020BC54E/\x01CompObj' data
 36: 'MBD020BC54E/Package' Microsoft Excel 2007+
 37: 'MBD020BC54F/\x01Ole' data
 38: 'MBD020BC54F/\x01Ole10natiVE' data
 39: 'Workbook' data
 40: '_VBA_PROJECT_CUR/PROJECT' ASCII text, with CRLF line terminators
 41: '_VBA_PROJECT_CUR/PROJECTwm' data
 42: '_VBA_PROJECT_CUR/VBA/Sheet1' data
 43: '_VBA_PROJECT_CUR/VBA/Sheet2' data
 44: '_VBA_PROJECT_CUR/VBA/Sheet3' data
 45: '_VBA_PROJECT_CUR/VBA/ThisWorkbook' data
 46: '_VBA_PROJECT_CUR/VBA/_VBA_PROJECT' data
 47: '_VBA_PROJECT_CUR/VBA/dir' data

@SANS_ISC C:\Demo>
```

By default, myjson-filter.py produces JSON output (filtered), but with option -l (--list), we obtain a list of the items and can easily observe what the effect of our filtering is (for the moment, we have not yet filtered).

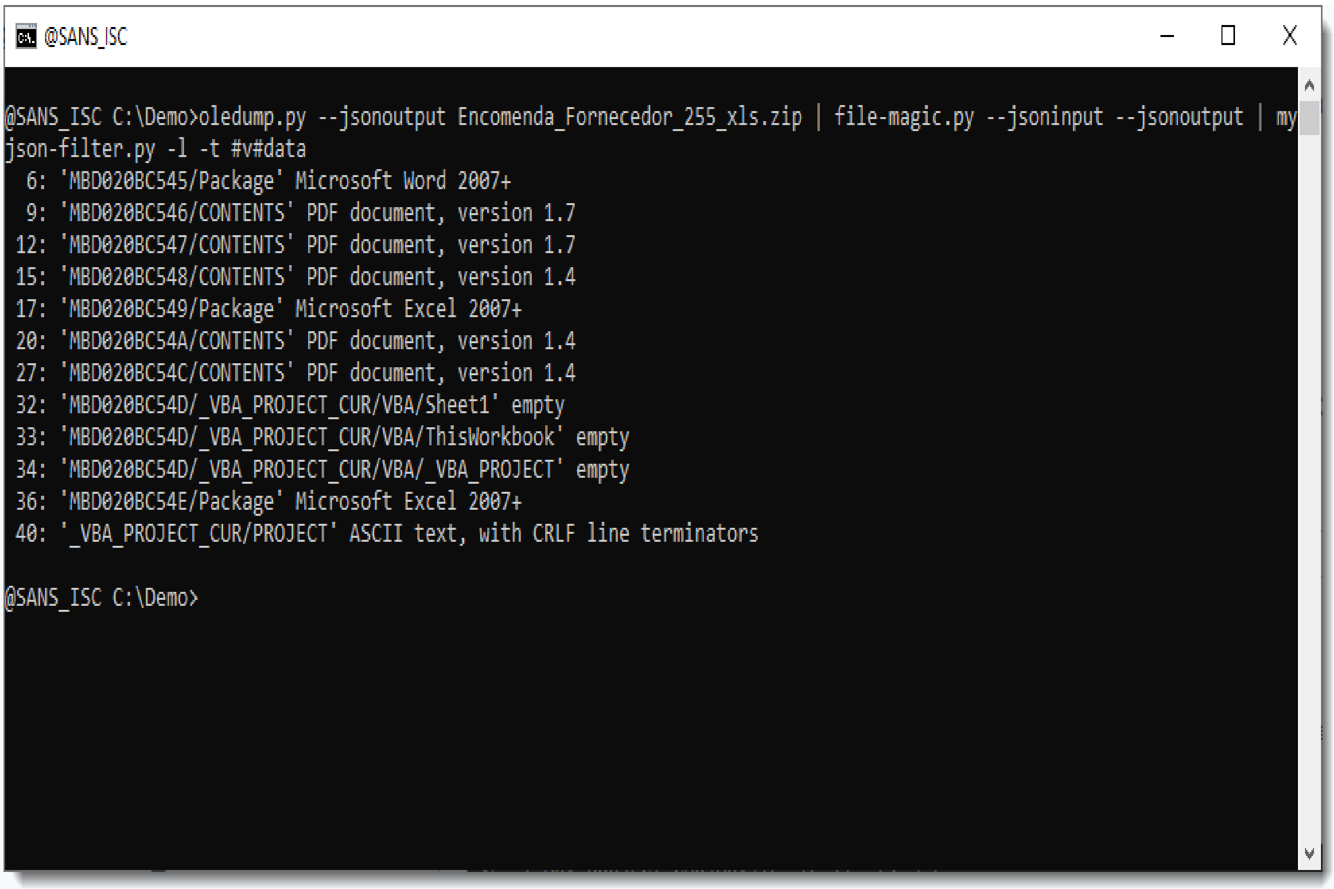
With option -t, we will filter by type (determined by file-magic.py). Option -t takes a regular expression that will be used to select types. Let's go with regular expression data:



```
@SANS_ISC C:\Demo>oledump.py --jsonoutput Encomenda_Fornecedor_255_xls.zip | file-magic.py --jsoninput --jsonoutput | my
json-filter.py -l -t data
  1: '\x01CompObj' data
  2: '\x05DocumentSummaryInformation' data
  3: '\x05SummaryInformation' data
  4: 'MBD020BC545/\x01CompObj' data
  5: 'MBD020BC545/\x01Ole' data
  7: 'MBD020BC546/\x01CompObj' data
  8: 'MBD020BC546/\x01Ole' data
 10: 'MBD020BC547/\x01CompObj' data
 11: 'MBD020BC547/\x01Ole' data
 13: 'MBD020BC548/\x01CompObj' data
 14: 'MBD020BC548/\x01Ole' data
 16: 'MBD020BC549/\x01CompObj' data
 18: 'MBD020BC54A/\x01CompObj' data
 19: 'MBD020BC54A/\x01Ole' data
 21: 'MBD020BC54B/\x01CompObj' data
 22: 'MBD020BC54B/\x05DocumentSummaryInformation' data
 23: 'MBD020BC54B/\x05SummaryInformation' data
 24: 'MBD020BC54B/Workbook' data
 25: 'MBD020BC54C/\x01CompObj' data
 26: 'MBD020BC54C/\x01Ole' data
 28: 'MBD020BC54D/\x01CompObj' data
 29: 'MBD020BC54D/\x05DocumentSummaryInformation' data
 30: 'MBD020BC54D/\x05SummaryInformation' data
 31: 'MBD020BC54D/Workbook' data
 35: 'MBD020BC54E/\x01CompObj' data
 37: 'MBD020BC54F/\x01Ole' data
 38: 'MBD020BC54F/\x01Ole10natiVE' data
 39: 'Workbook' data
 41: '_VBA_PROJECT_CUR/PROJECTwm' data
 42: '_VBA_PROJECT_CUR/VBA/Sheet1' data
 43: '_VBA_PROJECT_CUR/VBA/Sheet2' data
 44: '_VBA_PROJECT_CUR/VBA/Sheet3' data
 45: '_VBA_PROJECT_CUR/VBA/ThisWorkbook' data
 46: '_VBA_PROJECT_CUR/VBA/_VBA_PROJECT' data
 47: '_VBA_PROJECT_CUR/VBA/dir' data

@SANS_ISC C:\Demo>
```

At first, what is identified as just data, doesn't interest us. So we will reverse the selection (v), to select everything that isn't data, like this:

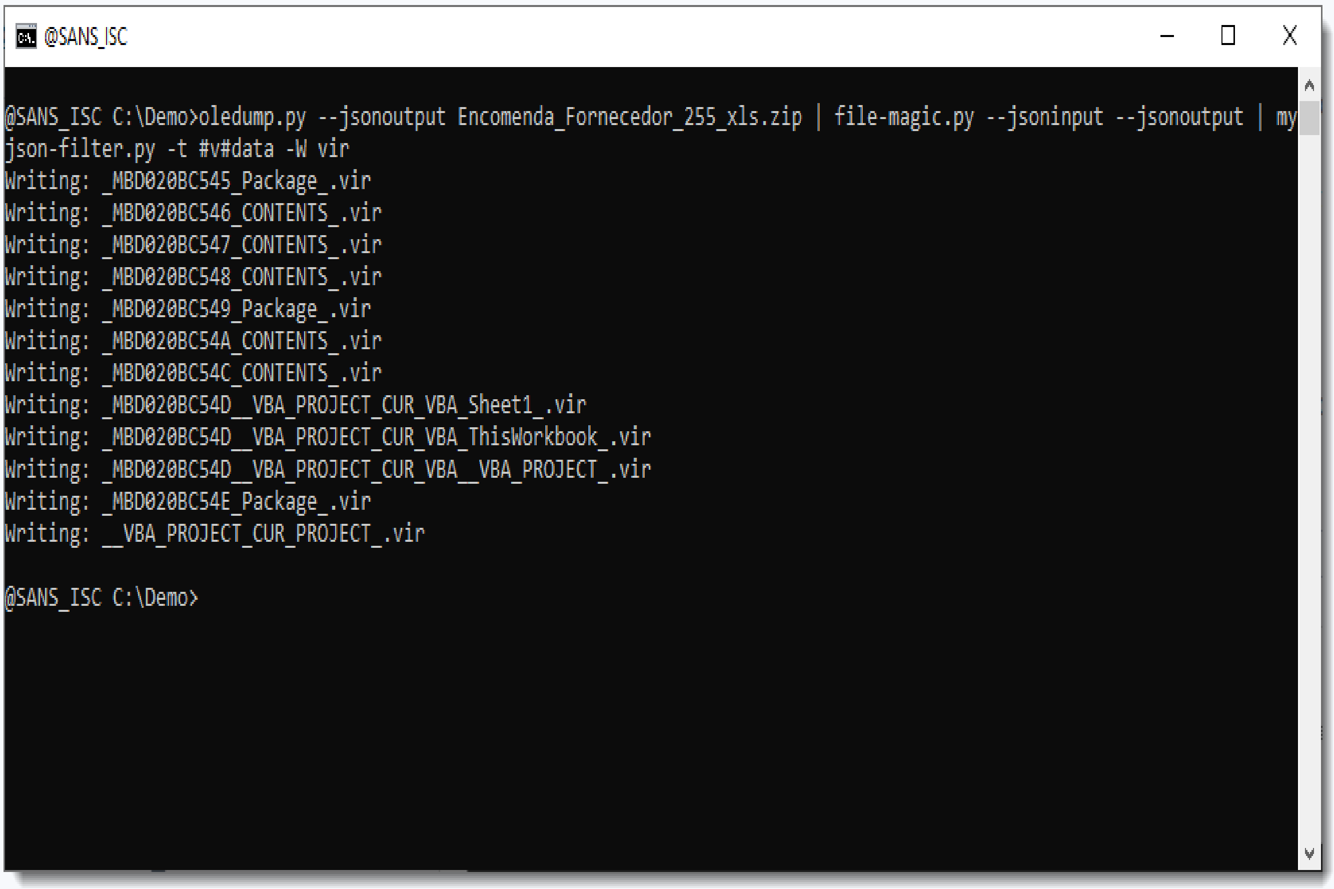


```
@SANS_ISC C:\Demo>oledump.py --jsonoutput Encomenda_Fornecedor_255_xls.zip | file-magic.py --jsoninput --jsonoutput | my
json-filter.py -l -t #v#data
  6: 'MBD020BC545/Package' Microsoft Word 2007+
  9: 'MBD020BC546/CONTENTS' PDF document, version 1.7
 12: 'MBD020BC547/CONTENTS' PDF document, version 1.7
 15: 'MBD020BC548/CONTENTS' PDF document, version 1.4
 17: 'MBD020BC549/Package' Microsoft Excel 2007+
 20: 'MBD020BC54A/CONTENTS' PDF document, version 1.4
 27: 'MBD020BC54C/CONTENTS' PDF document, version 1.4
 32: 'MBD020BC54D/_VBA_PROJECT_CUR/VBA/Sheet1' empty
 33: 'MBD020BC54D/_VBA_PROJECT_CUR/VBA/ThisWorkbook' empty
 34: 'MBD020BC54D/_VBA_PROJECT_CUR/VBA/_VBA_PROJECT' empty
 36: 'MBD020BC54E/Package' Microsoft Excel 2007+
 40: '_VBA_PROJECT_CUR/PROJECT' ASCII text, with CRLF line terminators

@SANS_ISC C:\Demo>
```

I justed added a new option to my myjson-filter.py tool, to easily write all selected items to disk as individual files: option -W (--write).

Option -W requires a value: vir, hash, hashvir or idvir. Value vir instructs my tool to create files with a filename that is the (cleaned) item name and with extension .vir. Like this:



```
@SANS_ISC C:\Demo>oledump.py --jsonoutput Encomenda_Fornecedor_255_xls.zip | file-magic.py --jsoninput --jsonoutput | my
json-filter.py -t #v#data -W vir
Writing: _MBD020BC545_Package_.vir
Writing: _MBD020BC546_CONTENTS_.vir
Writing: _MBD020BC547_CONTENTS_.vir
Writing: _MBD020BC548_CONTENTS_.vir
Writing: _MBD020BC549_Package_.vir
Writing: _MBD020BC54A_CONTENTS_.vir
Writing: _MBD020BC54C_CONTENTS_.vir
Writing: _MBD020BC54D__VBA_PROJECT_CUR_VBA_Sheet1_.vir
Writing: _MBD020BC54D__VBA_PROJECT_CUR_VBA_ThisWorkbook_.vir
Writing: _MBD020BC54D__VBA_PROJECT_CUR_VBA__VBA_PROJECT_.vir
Writing: _MBD020BC54E_Package_.vir
Writing: __VBA_PROJECT_CUR_PROJECT_.vir

@SANS_ISC C:\Demo>
```

So now we have written all streams to disk, that were identified as something else than just plain data.

If you don't find what you are looking for in these files, just use -t data to write all data files to disk, and see if you can find what you are looking for in these files.

For another example of my tools that support JSON, take a look at my blog post "Combining zipdump, file-magic And myjson-filter".

Didier Stevens
Senior handler
Microsoft MVP
blog.DidierStevens.com