Analyzing VSTO Office Files

blog.nviso.eu/2022/04/29/analyzing-vsto-office-files

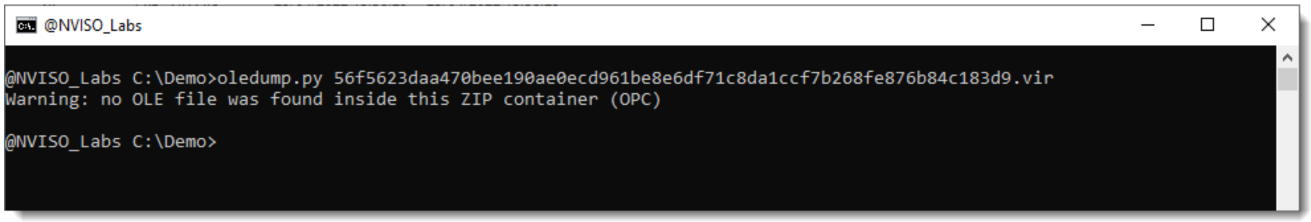
April 29, 2022

didiernviso maldoc, malware April 29, 2022 3 Minutes
VSTO Office files are Office document files linked to a Visual Studio Office File application. When opened, they launch a custom .NET application. There are various ways to achieve this, including methods to serve the VSTO files via an external web server.

An article was recently published on the creation of these document files for phishing purposes, and since then we have observed some VSTO Office files on VirusTotal.

Analysis Method (OOXML)

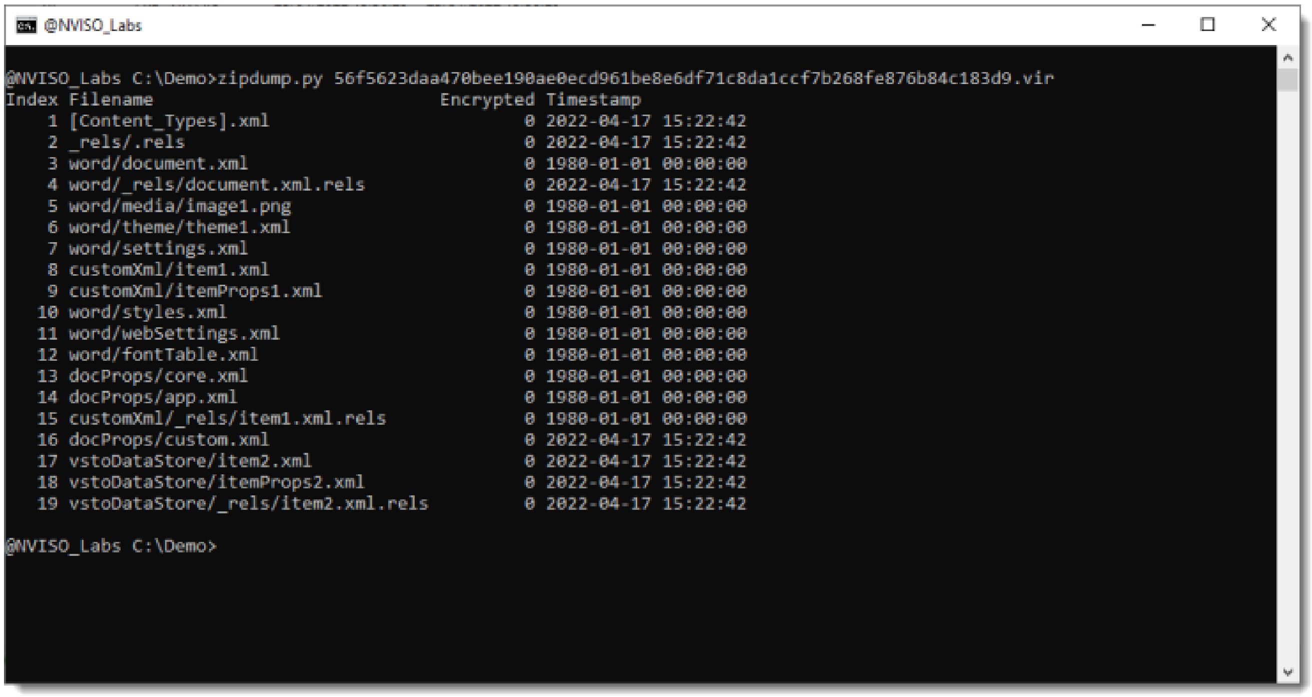
Sample Trusted Updater.docx (0/60 detections) appeared first on VirusTotal on 20/04/2022, 6 days after the publication of said article. It is a .docx file, and as can be expected, it does not contain VBA macros (per definition, .docm files contain VBA macros, .docx files do not):



```
@NVISO_Labs C:\Demo>oledump.py 56f5623daa470bee190ae0ecd961be8e6df71c8da1ccf7b268fe876b84c183d9.vir
Warning: no OLE file was found inside this ZIP container (OPC)

@NVISO_Labs C:\Demo>
```

Figure 1: typical VSTO document does not contain VBA code

Taking a look at the ZIP container (a .docx file is an OOXML file, i.e. a ZIP container containing XML files and other file types), there are some aspects that we don't usually see in "classic" .docx files:



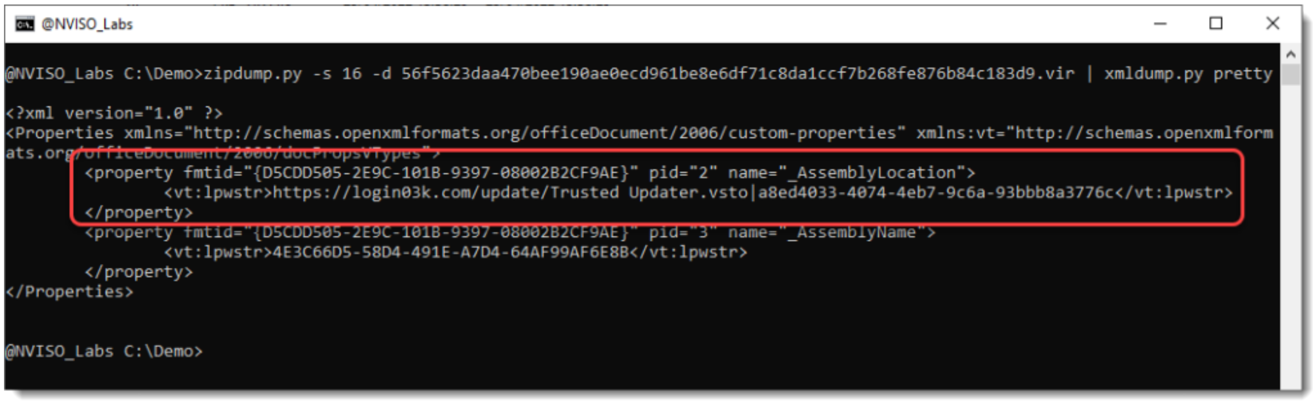
```
@NVISO_Labs C:\Demo>zipdump.py 56f5623daa470bee190ae0ecd961be8e6df71c8da1ccf7b268fe876b84c183d9.vir
Index Filename                              Encrypted Timestamp
    1 [Content_Types].xml                           0 2022-04-17 15:22:42
    2 _rels/.rels                                   0 2022-04-17 15:22:42
    3 word/document.xml                             0 1980-01-01 00:00:00
    4 word/_rels/document.xml.rels                  0 2022-04-17 15:22:42
    5 word/media/image1.png                         0 1980-01-01 00:00:00
    6 word/theme/theme1.xml                         0 1980-01-01 00:00:00
    7 word/settings.xml                             0 1980-01-01 00:00:00
    8 customXml/item1.xml                           0 1980-01-01 00:00:00
    9 customXml/itemProps1.xml                      0 1980-01-01 00:00:00
   10 word/styles.xml                               0 1980-01-01 00:00:00
   11 word/webSettings.xml                          0 1980-01-01 00:00:00
   12 word/fontTable.xml                            0 1980-01-01 00:00:00
   13 docProps/core.xml                             0 1980-01-01 00:00:00
   14 docProps/app.xml                              0 1980-01-01 00:00:00
   15 customXml/_rels/item1.xml.rels                0 1980-01-01 00:00:00
   16 docProps/custom.xml                           0 2022-04-17 15:22:42
   17 vstoDataStore/item2.xml                       0 2022-04-17 15:22:42
   18 vstoDataStore/itemProps2.xml                  0 2022-04-17 15:22:42
   19 vstoDataStore/_rels/item2.xml.rels            0 2022-04-17 15:22:42

@NVISO_Labs C:\Demo>
```

Figure 2: content of sample file

Worth noting is the following:

1. The presence of files in a folder called vstoDataStore. These files contain metadata for the execution of the VSTO file.
2. The timestamp of some of the files is not 1980-01-01, as it should be with documents created with Microsoft Office applications like Word.
3. The presence of a docsProp/custom.xml file.

Checking the content of the custom document properties file, we find 2 VSTO related properties: _AssemblyLocation and _AssemblyName:



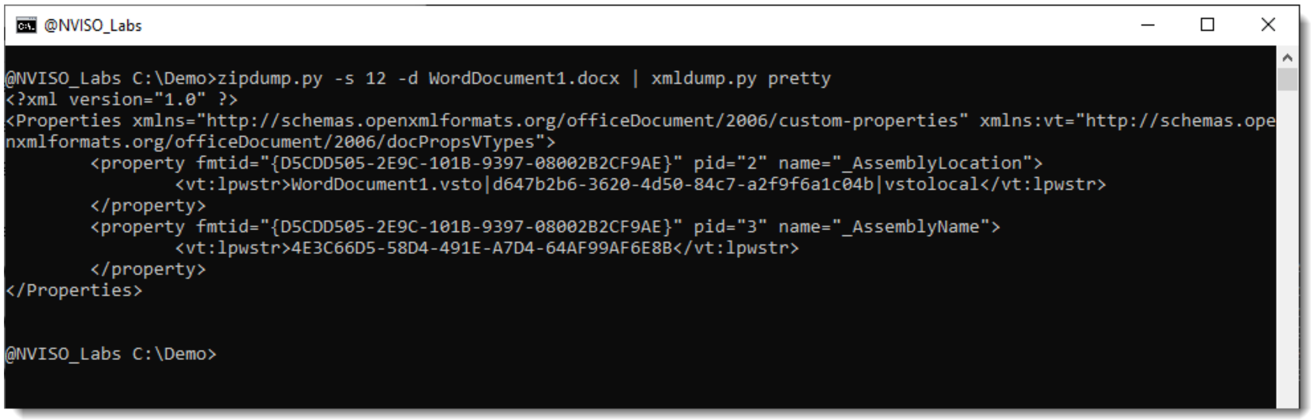
```
@NVISO_Labs C:\Demo>zipdump.py -s 16 -d 56f5623daa470bee190ae0ecd961be8e6df71c8da1ccf7b268fe876b84c183d9.vir | xmldump.py pretty

<?xml version="1.0" ?>
<Properties xmlns="http://schemas.openxmlformats.org/officeDocument/2006/custom-properties" xmlns:vt="http://schemas.openxmlformats.org/officeDocument/2006/docPropsVTypes">
        <property fmtid="{D5CDD505-2E9C-101B-9397-08002B2CF9AE}" pid="2" name="_AssemblyLocation">
                <vt:lpwstr>https://login03k.com/update/Trusted Updater.vsto|a8ed4033-4074-4eb7-9c6a-93bbb8a3776c</vt:lpwstr>
        </property>
        <property fmtid="{D5CDD505-2E9C-101B-9397-08002B2CF9AE}" pid="3" name="_AssemblyName">
                <vt:lpwstr>4E3C66D5-58D4-491E-A7D4-64AF99AF6E8B</vt:lpwstr>
        </property>
</Properties>


@NVISO_Labs C:\Demo>
```

Figure 3: custom properties _AssemblyLocation and _AssemblyName

The _AssemblyLocation in this sample is a URL to download a VSTO file from the Internet. We were not able to download the VSTO file, and neither was VirusTotal at the time of scanning. Thus we can not determine if this sample is a PoC, part of a red team engagement or truly malicious. It is a fact though, that this technique is known and used by red teams like ours, prior to the publication of said article.

There's little information regarding domain login03k[.]com, except that it appeared last year in a potential phishing domain list, and that VirusTotal tags it as DGA.

If the document uses a local VSTO file, then the _AssemblyLocation is not a URL:

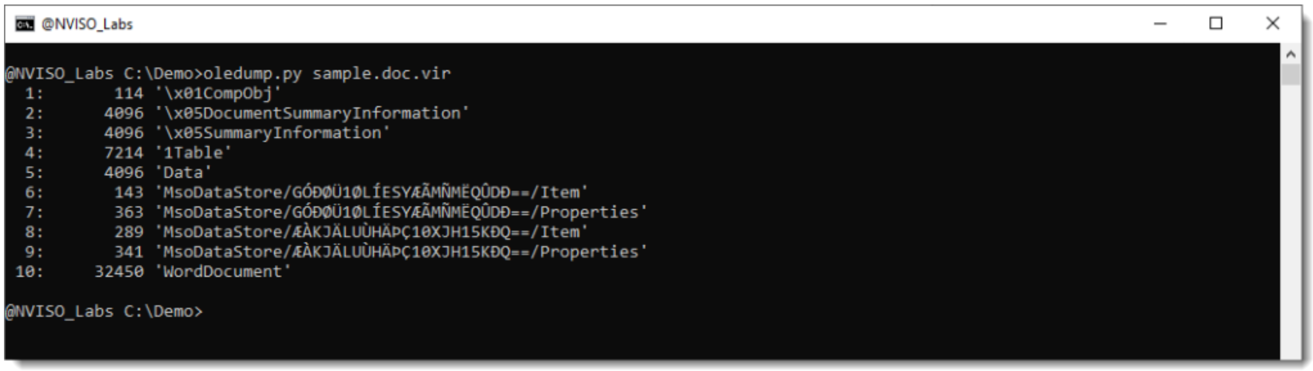


```
@NVISO_Labs C:\Demo>zipdump.py -s 12 -d WordDocument1.docx | xmldump.py pretty
<?xml version="1.0" ?>
<Properties xmlns="http://schemas.openxmlformats.org/officeDocument/2006/custom-properties" xmlns:vt="http://schemas.ope
nxmlformats.org/officeDocument/2006/docPropsVTypes">
        <property fmtid="{D5CDD505-2E9C-101B-9397-08002B2CF9AE}" pid="2" name="_AssemblyLocation">
                <vt:lpwstr>WordDocument1.vsto|d647b2b6-3620-4d50-84c7-a2f9f6a1c04b|vstolocal</vt:lpwstr>
        </property>
        <property fmtid="{D5CDD505-2E9C-101B-9397-08002B2CF9AE}" pid="3" name="_AssemblyName">
                <vt:lpwstr>4E3C66D5-58D4-491E-A7D4-64AF99AF6E8B</vt:lpwstr>
        </property>
</Properties>


@NVISO_Labs C:\Demo>
```

Figure 4: referencing a local VSTO file

Analysis Method (OLE)

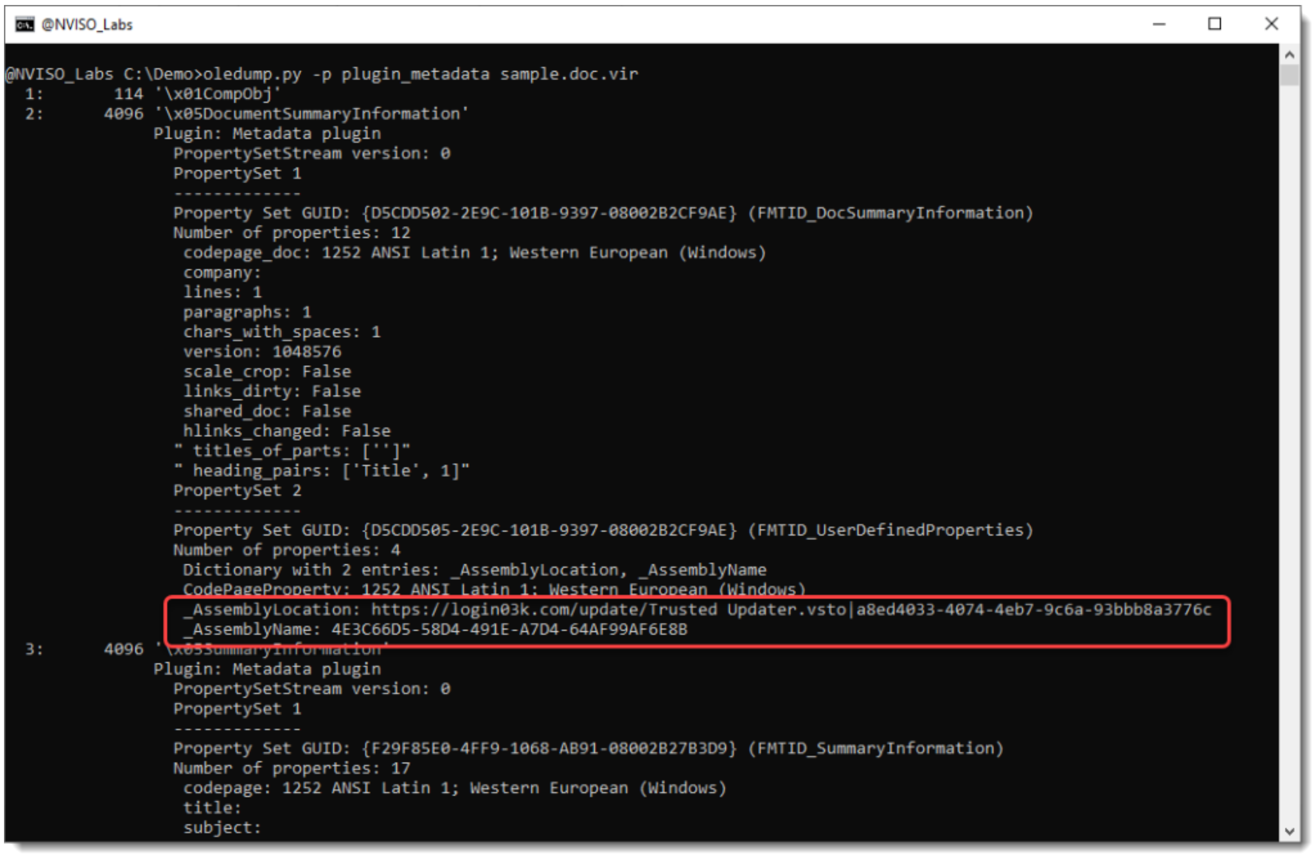
OLE files (the default Office document format prior to Office 2007) can also be associated with VSTO applications. We have found several examples on VirusTotal, but none that are malicious.
Therefore, to illustrate how to analyze such a sample, we converted the .docx maldoc from our first analysis, to a .doc maldoc.



```
@NVISO_Labs C:\Demo>oledump.py sample.doc.vir
  1:       114 '\x01CompObj'
  2:      4096 '\x05DocumentSummaryInformation'
  3:      4096 '\x05SummaryInformation'
  4:      7214 '1Table'
  5:      4096 'Data'
  6:       143 'MsoDataStore/GÓÐØÜ1ØLÍESYÆÃMÑMËQÛDÐ==/Item'
  7:       363 'MsoDataStore/GÓÐØÜ1ØLÍESYÆÃMÑMËQÛDÐ==/Properties'
  8:       289 'MsoDataStore/ÆÀKJÄLUÙHÄÞÇ10XJH15KÐQ==/Item'
  9:       341 'MsoDataStore/ÆÀKJÄLUÙHÄÞÇ10XJH15KÐQ==/Properties'
 10:     32450 'WordDocument'

@NVISO_Labs C:\Demo>
```

Figure 5: analysis of .doc file

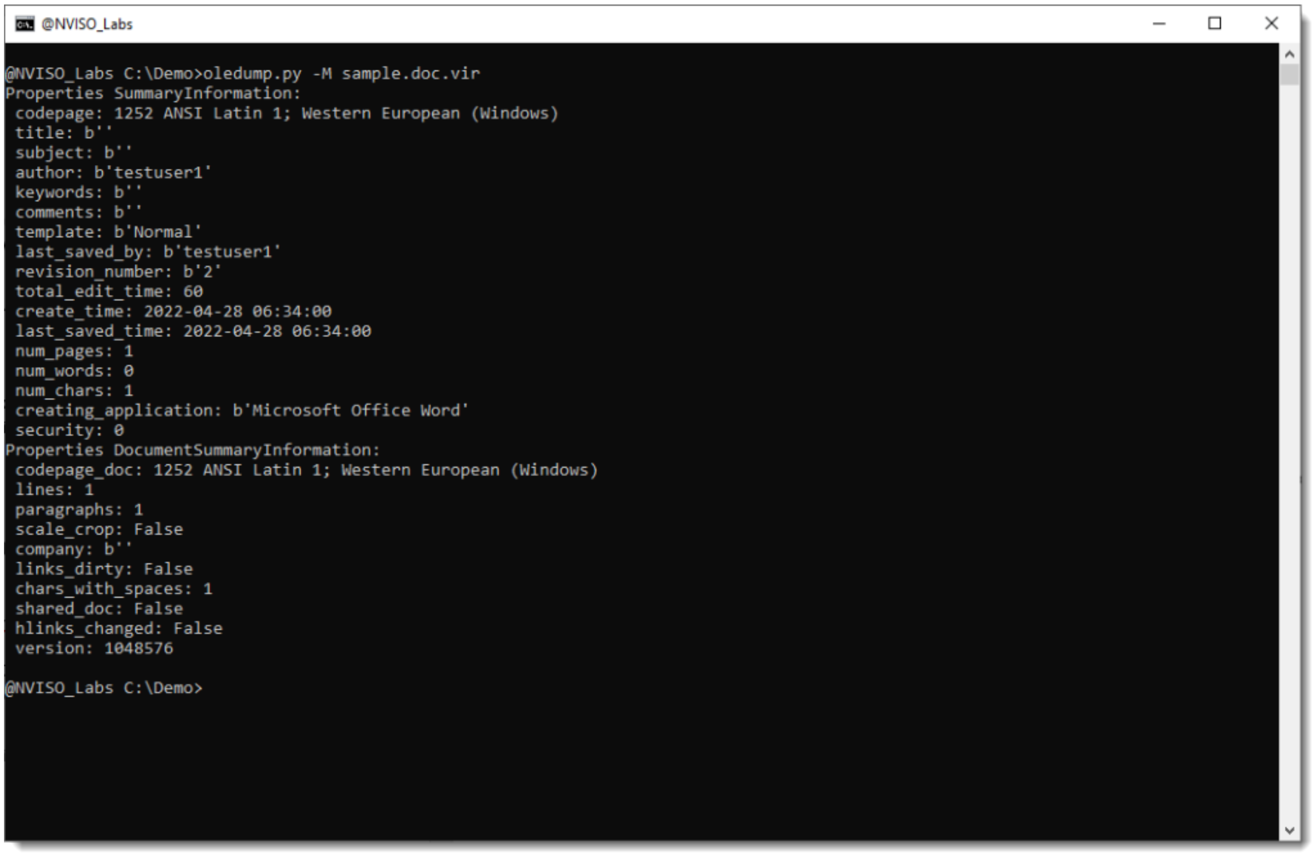
Taking a look at the metadata with oledump's plugin_metadata, we find the _AssemblyLocation and _AssemblyName properties (with the URL):



```
@NVISO_Labs C:\Demo>oledump.py -p plugin_metadata sample.doc.vir
  1:       114 '\x01CompObj'
  2:      4096 '\x05DocumentSummaryInformation'
               Plugin: Metadata plugin
                 PropertySetStream version: 0
                 PropertySet 1
                 -------------
                 Property Set GUID: {D5CDD502-2E9C-101B-9397-08002B2CF9AE} (FMTID_DocSummaryInformation)
                 Number of properties: 12
                  codepage_doc: 1252 ANSI Latin 1; Western European (Windows)
                  company:
                  lines: 1
                  paragraphs: 1
                  chars_with_spaces: 1
                  version: 1048576
                  scale_crop: False
                  links_dirty: False
                  shared_doc: False
                  hlinks_changed: False
                 " titles_of_parts: ['']"
                 " heading_pairs: ['Title', 1]"
                 PropertySet 2
                 -------------
                 Property Set GUID: {D5CDD505-2E9C-101B-9397-08002B2CF9AE} (FMTID_UserDefinedProperties)
                 Number of properties: 4
                  Dictionary with 2 entries: _AssemblyLocation, _AssemblyName
                  CodePageProperty: 1252 ANSI Latin 1; Western European (Windows)
                  _AssemblyLocation: https://login03k.com/update/Trusted Updater.vsto|a8ed4033-4074-4eb7-9c6a-93bbb8a3776c
                  _AssemblyName: 4E3C66D5-58D4-491E-A7D4-64AF99AF6E8B
  3:      4096 '\x05SummaryInformation'
               Plugin: Metadata plugin
                 PropertySetStream version: 0
                 PropertySet 1
                 -------------
                 Property Set GUID: {F29F85E0-4FF9-1068-AB91-08002B27B3D9} (FMTID_SummaryInformation)
                 Number of properties: 17
                  codepage: 1252 ANSI Latin 1; Western European (Windows)
                  title:
                  subject:
```

Figure 6: custom properties _AssemblyLocation and _AssemblyName

Notice that this metadata does not appear when you use oledump's option -M:



```
@NVISO_Labs C:\Demo>oledump.py -M sample.doc.vir
Properties SummaryInformation:
 codepage: 1252 ANSI Latin 1; Western European (Windows)
 title: b''
 subject: b''
 author: b'testuser1'
 keywords: b''
 comments: b''
 template: b'Normal'
 last_saved_by: b'testuser1'
 revision_number: b'2'
 total_edit_time: 60
 create_time: 2022-04-28 06:34:00
 last_saved_time: 2022-04-28 06:34:00
 num_pages: 1
 num_words: 0
 num_chars: 1
 creating_application: b'Microsoft Office Word'
 security: 0
Properties DocumentSummaryInformation:
 codepage_doc: 1252 ANSI Latin 1; Western European (Windows)
 lines: 1
 paragraphs: 1
 scale_crop: False
 company: b''
 links_dirty: False
 chars_with_spaces: 1
 shared_doc: False
 hlinks_changed: False
 version: 1048576

@NVISO_Labs C:\Demo>
```

Figure 7: olefile's metadata result

Option -M extracts the metadata using olefile's methods, and this olefile Python module (whereupon oledump relies) does not (yet) parse user defined properties.

Conclusion

To analyze Office documents linked with VSTO apps, search for custom properties **_AssemblyLocation** and **_AssemblyName**.

To detect Office documents like these, we have created some YARA rules for our VirusTotal hunting. You can find them on our Github here. Some of them are rather generic by design, and will generate too many hits for use in a production environment. They are originally designed for hunting on VT.

We will discus these rules in detail in a follow-up blog post, but we already wanted to share these with you.

About the authors

Didier Stevens is a malware expert working for NVISO. Didier is a SANS Internet Storm Center senior handler and Microsoft MVP, and has developed numerous popular tools to assist with malware analysis. You can find Didier on Twitter and LinkedIn.

You can follow NVISO Labs on Twitter to stay up to date on all our future research and publications.

Published by didiernviso

View all posts by didiernviso

Published April 29, 2022