XLSB: Analyzing a Microsoft Excel Binary Spreadsheet

clickallthethings.wordpress.com/2021/02/02/xlsb-analyzing-a-microsoft-excel-binary-spreadsheet/

View all posts by Jamie February 2, 2021

The @InQuest crew has been putting some unusual documents out on the Twitters and I thought I'd take a closer look at one of them. And as ALWAYS, new documents are never that straight forward to analyze. Attackers always put a twist on what has already been done.

In this case, we've got an .xlsb file, XLM code, hidden sheets, protected sheets, and the ever-so-sneaky-hiding-in-plain-sight white font.

Here's the doc: **https://app.any.run/tasks/47e1c347-664c-4ada-9655-1724387e859e/#**

Microsoft Excel Binary Spreadsheets (.xlsb)

I can't say that I've ever seen these until now. They open and function like any other spreadsheet. The difference is under the hood. They store the spreadsheet using a binary format (BIFF12) rather than the typical .xlsx or .xls. This means that .xlsb files are usually bigger as they're not compressed. Extremely complicated spreadsheets (ones with lots of formulas, charts, and shapes) can benefit from this file format as they may save and load much faster.

But this format means that some of our normal analysis tools do not work. For example, oledump can't find any OLE objects in the file.

```
C:\Users\REM\Desktop\sample>oledump.py xlsb_file.xlsb
Warning: no OLE file was found inside this ZIP container (OPC)
```

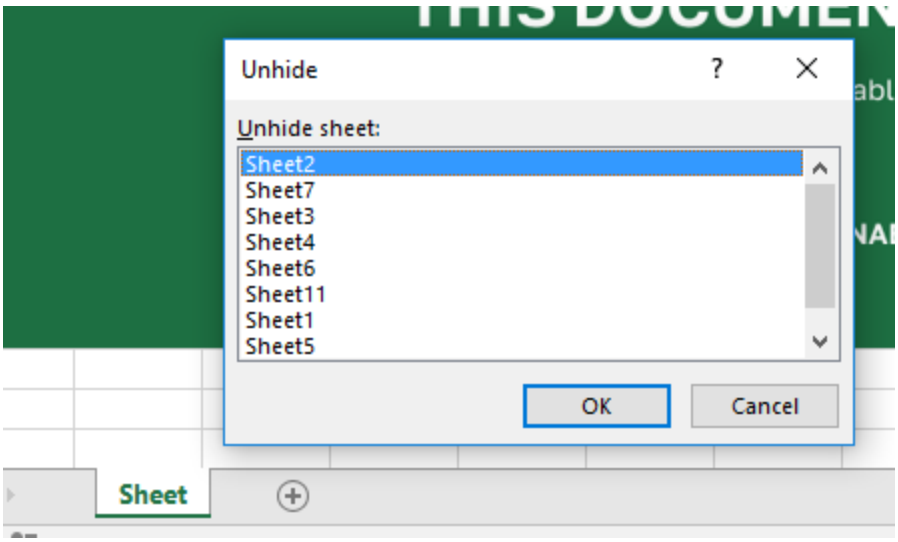
OfficeMalScanner will expand the document if you use the *inflate* option. However, the output doesn't show the normal vbaproject.bin. There's a ton of other files that are NOT vbaproject.bin. (*Of course*, there's no vbaproject.bin. Oledump.py didn't find any macros, right?)

```
C:\Users\REM\Desktop\sample>OfficeMalScanner xlsb_file.xlsb inflate

+----------------------------------------+
|           OfficeMalScanner v0.62       |
|  Frank Boldewin / www.reconstructer.org  |
+----------------------------------------+

[*] INFLATE mode selected
[*] Opening file xlsb_file.xlsb
[*] Filesize is 86772 (0x152f4) Bytes
[*] Microsoft Office Open XML Format document detected.

Found 49 files in this archive

[CONTENT_TYPES].XML ----- 3819 Bytes ----- at Offset 0x00000000
_RELS/.RELS ----- 732 Bytes ----- at Offset 0x0000049d
XL/_RELS/WORKBOOK.BIN.RELS ----- 2096 Bytes ----- at Offset 0x000007d6
XL/WORKBOOK.BIN ----- 978 Bytes ----- at Offset 0x00000a90
DOCPROPS/THUMBNAIL.WMF ----- 114532 Bytes ----- at Offset 0x00000ca1
XL/STYLES.BIN ----- 929 Bytes ----- at Offset 0x0000506d
XL/THEME/THEME1.XML ----- 8390 Bytes ----- at Offset 0x0000521f
XL/WORKSHEETS/_RELS/SHEET4.BIN.RELS ----- 284 Bytes ----- at Offset 0x000059d1
XL/WORKSHEETS/SHEET9.BIN ----- 6434 Bytes ----- at Offset 0x00005ad3
XL/SHAREDSTRINGS.BIN ----- 22052 Bytes ----- at Offset 0x0000608e
XL/DRAWINGS/DRAWING1.XML ----- 1368 Bytes ----- at Offset 0x00007f5a
XL/MEDIA/IMAGE1.PNG ----- 33923 Bytes ----- at Offset 0x00008219
XL/WORKSHEETS/_RELS/SHEET1.BIN.RELS ----- 426 Bytes ----- at Offset 0x000106cd
XL/WORKSHEETS/_RELS/SHEET2.BIN.RELS ----- 692 Bytes ----- at Offset 0x000107f2
XL/WORKSHEETS/_RELS/SHEET3.BIN.RELS ----- 284 Bytes ----- at Offset 0x0001092c
XL/WORKSHEETS/SHEET1.BIN ----- 544 Bytes ----- at Offset 0x00010a2e
XL/WORKSHEETS/_RELS/SHEET5.BIN.RELS ----- 284 Bytes ----- at Offset 0x00010b9b
XL/WORKSHEETS/SHEET8.BIN ----- 2727 Bytes ----- at Offset 0x00010c9d
XL/WORKSHEETS/_RELS/SHEET6.BIN.RELS ----- 284 Bytes ----- at Offset 0x00011083
XL/WORKSHEETS/_RELS/SHEET9.BIN.RELS ----- 449 Bytes ----- at Offset 0x00011185
XL/WORKSHEETS/SHEET4.BIN ----- 1079 Bytes ----- at Offset 0x000112ae
XL/DRAWINGS/_RELS/DRAWING1.XML.RELS ----- 292 Bytes ----- at Offset 0x0001140c
XL/WORKSHEETS/SHEET3.BIN ----- 2143 Bytes ----- at Offset 0x0001150b
XL/WORKSHEETS/SHEET5.BIN ----- 743 Bytes ----- at Offset 0x00011703
XL/WORKSHEETS/_RELS/SHEET8.BIN.RELS ----- 449 Bytes ----- at Offset 0x0001183b
XL/WORKSHEETS/SHEET6.BIN ----- 743 Bytes ----- at Offset 0x00011964
XL/WORKSHEETS/SHEET7.BIN ----- 5971 Bytes ----- at Offset 0x00011aa0
XL/WORKSHEETS/SHEET2.BIN ----- 3140 Bytes ----- at Offset 0x0001213d
XL/MACROSHEETS/_RELS/SHEET1.BIN.RELS ----- 284 Bytes ----- at Offset 0x00012453
XL/WORKSHEETS/_RELS/SHEET7.BIN.RELS ----- 284 Bytes ----- at Offset 0x00012556
XL/MACROSHEETS/SHEET1.BIN ----- 2612 Bytes ----- at Offset 0x00012658
XL/WORKSHEETS/BINARYINDEX2.BIN ----- 345 Bytes ----- at Offset 0x00012a35
DOCPROPS/APP.XML ----- 1132 Bytes ----- at Offset 0x00012b23
XL/WORKSHEETS/BINARYINDEX1.BIN ----- 29 Bytes ----- at Offset 0x00012e1b
XL/TABLES/TABLE1.BIN ----- 1520 Bytes ----- at Offset 0x00012e6b
XL/CALCCHAIN.BIN ----- 3051 Bytes ----- at Offset 0x000130d3
XL/MACROSHEETS/BINARYINDEX1.BIN ----- 194 Bytes ----- at Offset 0x00013366
XL/WORKSHEETS/BINARYINDEX6.BIN ----- 85 Bytes ----- at Offset 0x00013412
XL/WORKSHEETS/BINARYINDEX5.BIN ----- 85 Bytes ----- at Offset 0x00013486
XL/WORKSHEETS/BINARYINDEX4.BIN ----- 121 Bytes ----- at Offset 0x000134fa
XL/WORKSHEETS/BINARYINDEX3.BIN ----- 315 Bytes ----- at Offset 0x0001357c
XL/WORKSHEETS/BINARYINDEX7.BIN ----- 230 Bytes ----- at Offset 0x00013659
XL/WORKSHEETS/BINARYINDEX8.BIN ----- 133 Bytes ----- at Offset 0x00013722
```

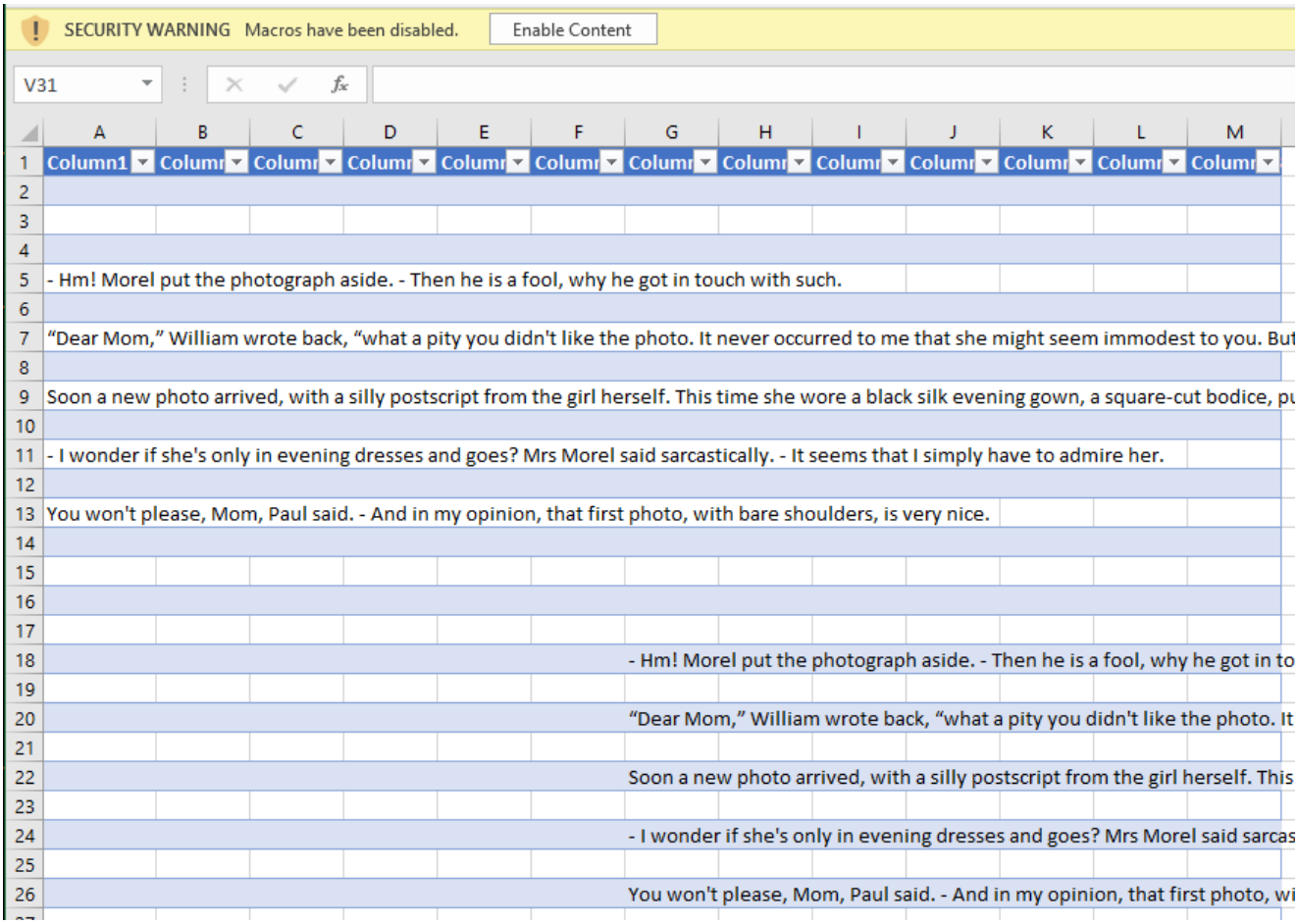
In hindsight, all of these .bin files make sense as this is an .xlsb file. It certainly didn't make sense the first time I cracked it open, though.

Analysis from within the document

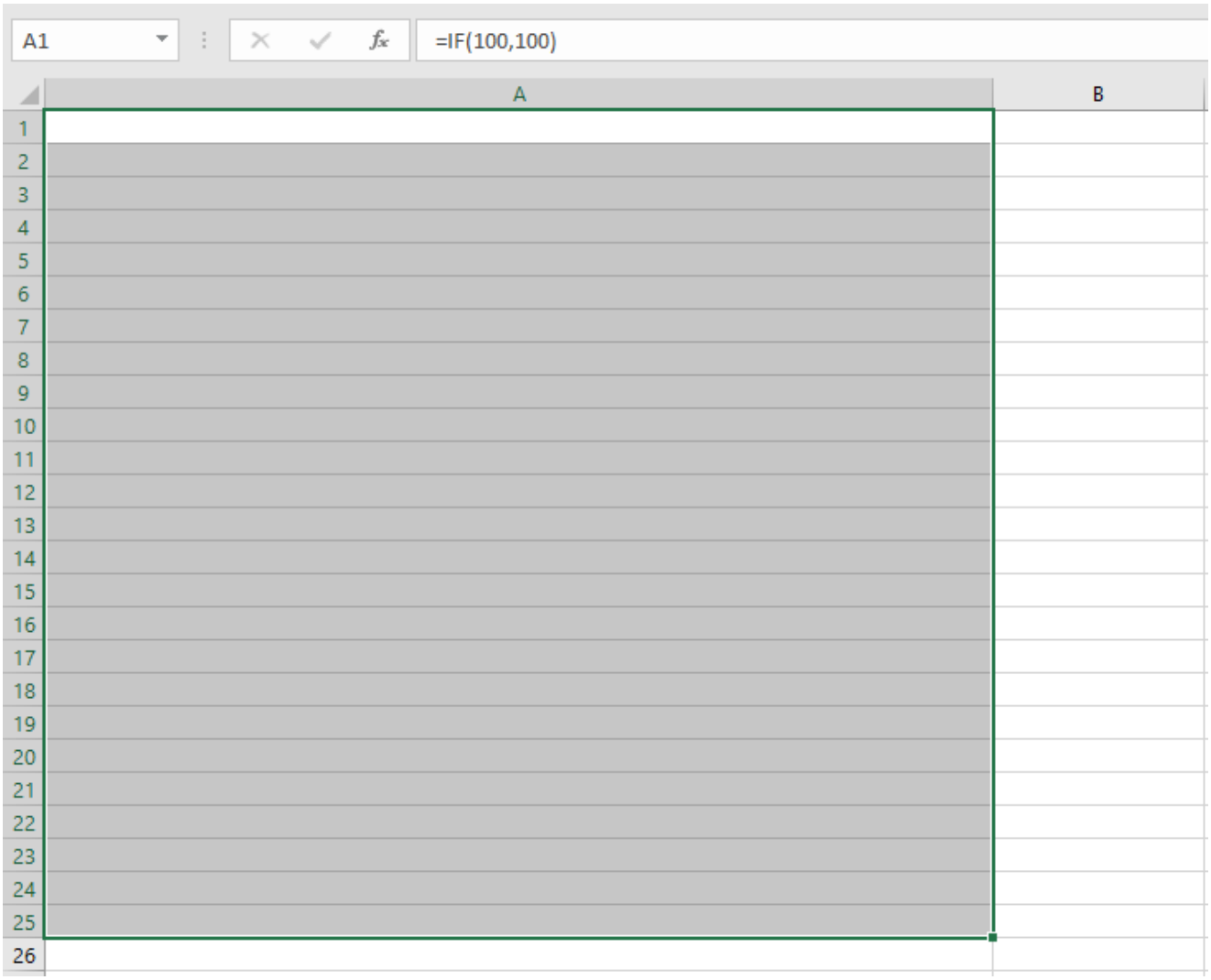
First, there are a bunch of hidden sheets.



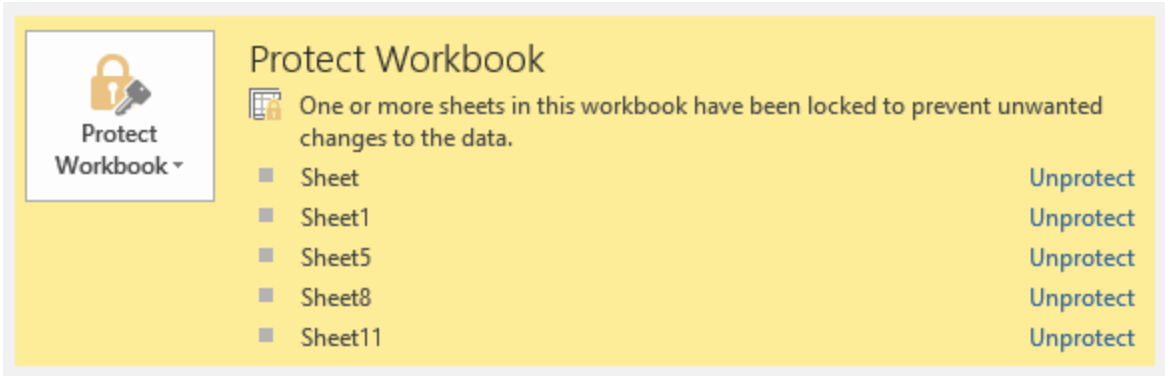
Quite a few of these sheets contain a lot of nonsense.



However, Auto_Open is pointing to Sheet11!A1. If we go there, we can see a empty columns containing XLM in white font.



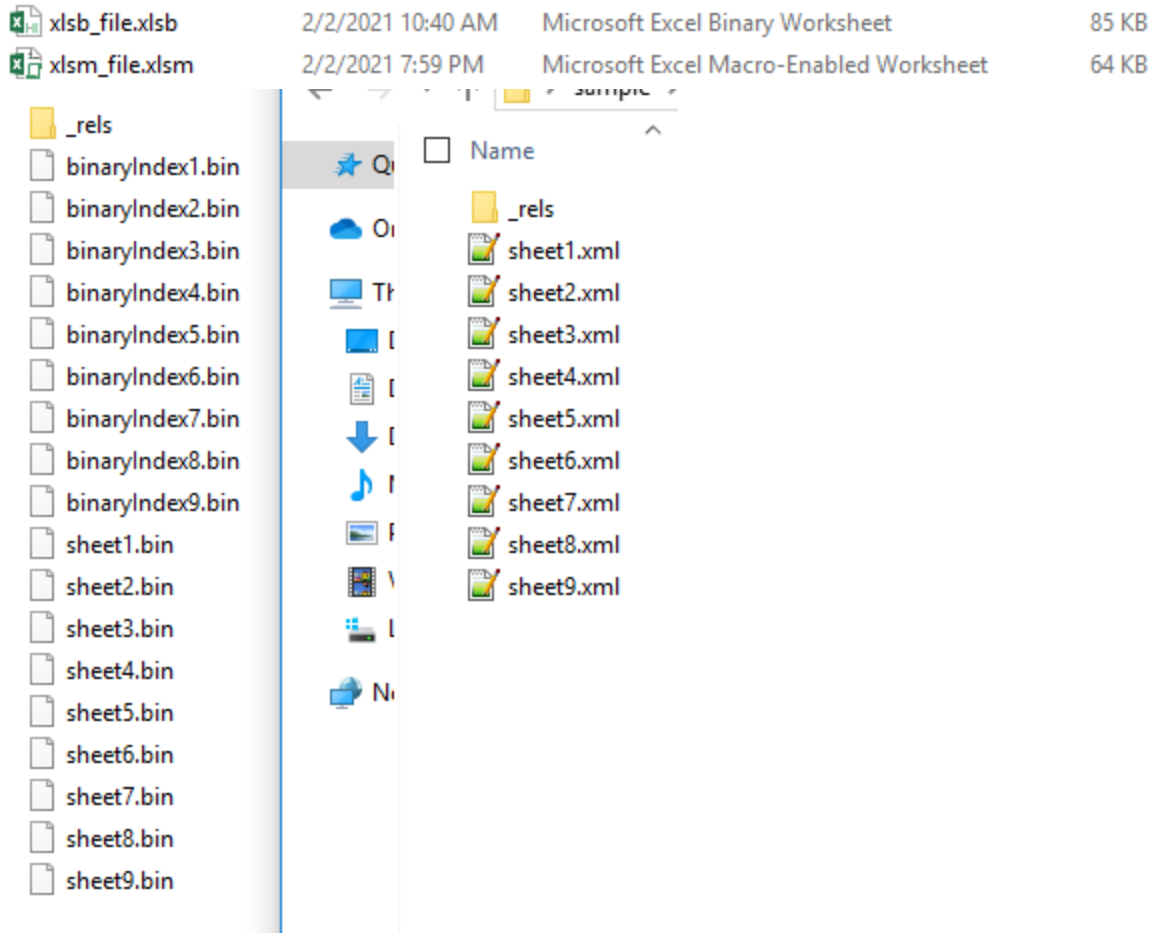
And of COURSE we can't edit the font because a bunch of these sheets are protected AND we don't have the password!



It is possible to enable content and step through the XLM like we've done before. However, the other sheets contain macro code and characters that are spread out all over the place. That's annoying enough to analyze even if it wasn't in white font.

Unprotecting the sheets

All hope is not lost. We can bypass the protected sheets if we save the document in a different macro-enabled format like .xlsm. This will also change the worksheets from .bin files to .xml files. This will be important in a moment.



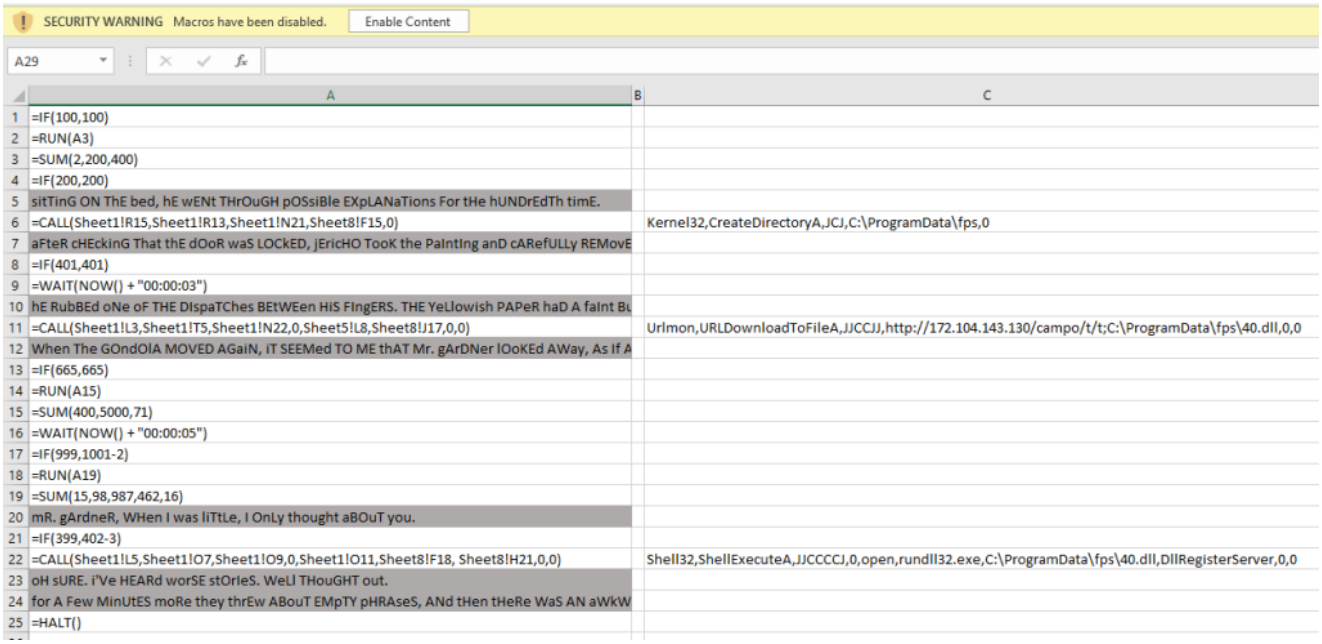
Change your newly created .xlsm file to a .zip and navigate your way into *xl\worksheets* and *xl\macrosheets* to find .xml sheets. We will un-protect these sheets by taking out a section called *sheetProtection*. Delete everything from <*sheetProtection* to its trailing />, including the two < > characters, and then save the .xml. Delete sheet protection wherever you find it.

```
</v></c></row></sheetData><sheetProtection algorithmName="SHA-512" hashValue=
"UsJ9dKUhoi7F1hwdMRJBEe5f7TQrFHG1ToNAXJpE6eXhMtCUzosHDHvRAe5SbVftlm8Xv+QQ3mjO9A7+01MUYg==" saltValue="nuuR1fcA2gWnmTSuIssUJQ==" spinCount
="100000" sheet="1" objects="1" scenarios="1"/><pageMargins left="0.7" right="0.7" top="0.75" bottom="0.75" header="0.3" footer="0.3"/>
</xm:macrosheet>
```

You'll have to drag the various .xml files out of the .zip file, edit them, and then copy them back into their appropriate locations. Finally, change the file from .zip to .xlsm.

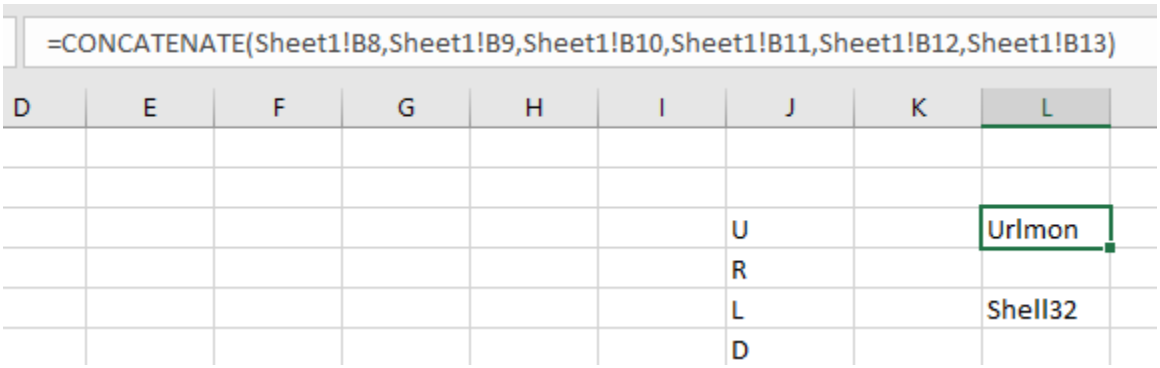
Analyzing the XLM

Now that the sheets are unprotected, we can finally get rid of that white font and see the XLM. Auto_open is pointing to Sheet11!A1. I color-coded the extraneous lines and analyzed the three important =CALL() commands.



Ultimately, it downloads a .dll to C:\ProgramData\fps\ and registers that .dll via rundll32.exe.

As a bonus, Excel does some of the XLM concatenating for us making analysis easier. Here's an example from Sheet1. You can see that cell L3 contains =CONCATENATE(Sheet…), but the cell itself shows the result.



Good times!

Thanks for reading.