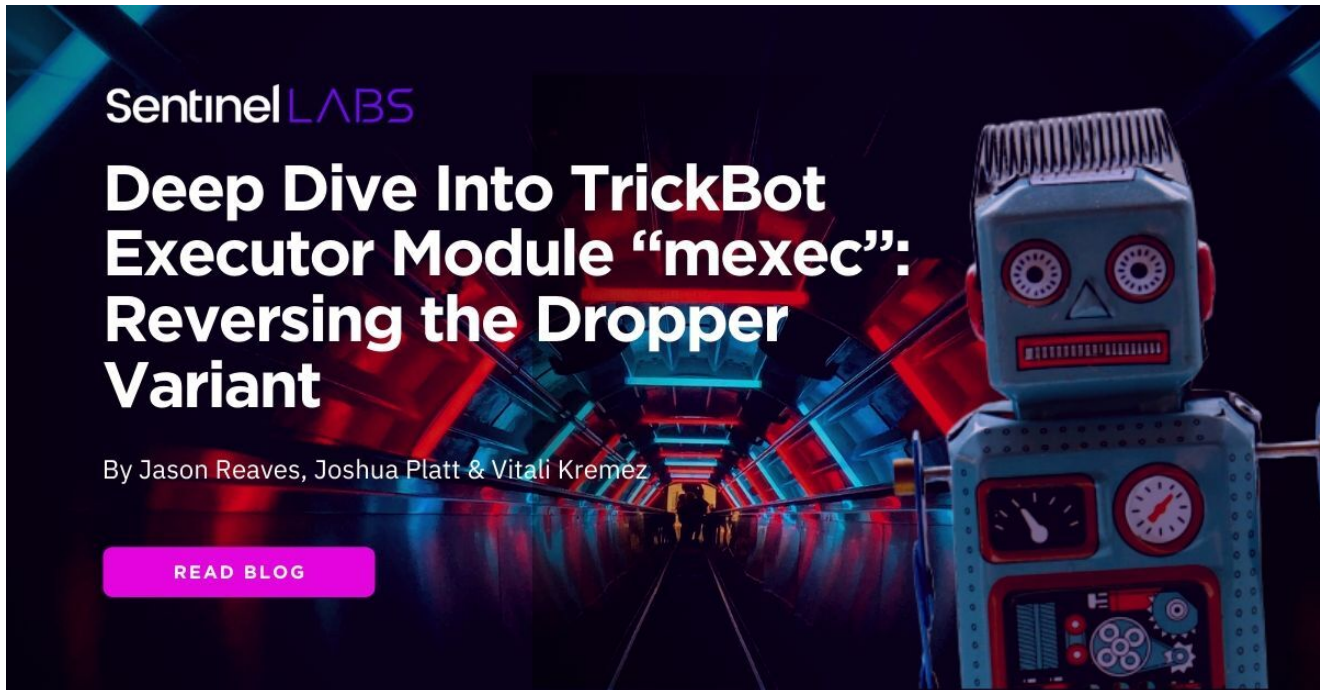


Deep Dive Into TrickBot Executor Module “mexec”: Reversing the Dropper Variant

 labs.sentinelone.com/deep-dive-into-trickbot-executor-module-mexec-reversing-the-dropper-variant/

Jason Reaves



In a recent post [1], we disclosed details of a new Trickbot executor module “mexec” and analyzed the downloader variant of this module. In this follow up post, we provide the first analysis the dropper version of the mexec module.

See the report for the full list of IOCs and further details on the TrickBot executor module “mexec”.

[Read the Full Report](#)

Background

TrickBot is the successor of Dyre [2,3], and at first was primarily focused on banking fraud and utilized injection systems in the same manner. Over the years, TrickBot has shifted focus to enterprise environments to incorporate everything from network profiling and mass data collection to lateral traversal exploits. This focus shift is also prevalent in their incorporation of malware and techniques in their tertiary deliveries that are targeting enterprise environments. Such behavior is similar to a company where the focus will shift depending on what generates the best revenue.

Research Insights

The dropper version of mexec is very similar to the downloader except that the payload is carried onboard, which makes the module substantially larger than its downloader counterpart. The dropper variant is also very similar to TrickBot's loader but was written to be able to accommodate more generic deliveries.

The dropper version of mexec functionally overlaps with how TrickLoader works:

- Custom Base64 alphabet string encoding
- Usage of MiniLZO
- Function obfuscation using a table of offsets

This version of mexec could easily be misidentified as TrickLoader, but mexec is not used to deliver TrickBot and is simply setup as a dropper piece that will write and execute an onboard hidden executable. It's also a DLL and designed to be executed in memory to perform its task or to deliver another piece of malware.

While the strings are encoded in the same manner you would expect in a Trickbot sample, there are noticeably fewer of them.

```
uu      0
a7eyx3jocpne49d db '7eyx3JoCpNE49dhTPm+Srsu0/MHKbQVgzAqBaLl20FvI1X6wcDkiZjtFU8RY5WGn',0
              align 10h
aM2szq0n1bfz6m0 db 'M2sZQ0NLbFz6M0AL',0
aZsSas99ip6mod1 db '+Zs+Sas99ip6MoD1',0
aRzajs3ci9q8ako db 'rZAJ33ci9q8aKoc',0
              align 4
aada-
```

Decoding them is the same process as you would find in a TrickBot sample.

```
def decode_s(str):
    b64 = "7eyx3JoCpNE49dhTPm+Srsu0/MHKbQVgzAqBaLl20FvI1X6wcDkiZjtFU8RY5WGn"
    std = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"
    str = str.translate(string.maketrans(b64,std))
    return base64.b64decode(str+'='*(len(str)%4))
```

```
Python>decode_s('M2sZQ0NLbFz6M0AL')
futuresx.exe
Python>decode_s('+Zs+Sas99ip6MoD1')
KERNEL32.dll
Python>decode_s('rZAJ33ci9q8aKoc')
SHELL32.dll
```

The first thing the sample does is spin up multiple threads, but the interesting one is the second one.

```

mov     dword_1001F000, 0
lea    eax, [ebp+ThreadId]
push   eax                ; lpThreadId
push   0                  ; dwCreationFlags
push   0                  ; lpParameter
push   offset StartAddress ; lpStartAddress
push   0                  ; dwStackSize
push   0                  ; lpThreadAttributes
call   ds:CreateThread
lea    eax, [ebp+ThreadId]
push   eax                ; lpThreadId
push   0                  ; dwCreationFlags
push   offset dword_1001F00C ; lpParameter
push   offset loc_10001160 ; lpStartAddress
push   0                  ; dwStackSize
push   0                  ; lpThreadAttributes
call   ds:CreateThread

```

In the main thread that is spun up as the second one, it performs the normal deobfuscation routine that you find in a Trickbot loader sample. If you are unfamiliar with this process of deobfuscation, you can find more details in our report on [TrickLoader Deobfuscation](#).

```

loc_10001160:                                ; DATA XREF: Control+5E1fo
mov     ebp, esp
call   loc_100011AF
; -----
db  0BCh
db  0
db  3Dh ; =
db  0
db  0
db  3
db  70h ; p
db  0
db  0B0h
db  0
db  40h
db  2
db  50h ; P
db  0
db  50h
dw  0
dd  70005001h, 60003000h, 80004000h, 30009000h, 30005000h
dd  70004000h, 6003C000h, 0C001D001h, 4000D001h, 70008000h
dd  50006001h, 0F0005000h, 600030FFh
db  3 dup(0)
; -----
loc_100011AF:                                ; CODE XREF: .text:10001162fp
pop     edi
pop     eax
pop     esi
xor     eax, eax
push   31Eh
pop     ecx
loc_10001400:                                ; CODE XREF: .text:10001400f

```

The above code is the start of the process of building out the function address table; it jumps over a table of offsets to kick off the process.

```
loc_100011BA:                                ; CODE XREF: .t
        push    eax
        loop   loc_100011BA
        mov     eax, edi
        push   edi
        mov     ebp, esp
        add     eax, 1C049h
        mov     [ebp+4], eax
        mov     eax, esi
        mov     [ebp+20h], eax
        push   0FFF0h
        pop    ecx
        mov     esi, edi
        mov     edx, edi
        cld

loc_100011DA:                                ; CODE XREF: .t
        mov     eax, ecx
        lodsw
        test    eax, eax
        jz     short loc_100011FE
        cmp     eax, ecx
        jb     short loc_100011F9
        sub     eax, ecx
        push   ecx
        mov     ecx, edi
        add     ecx, 1C051h
        shl     eax, 2
        add     ecx, eax
        mov     eax, [ecx]
        pop    ecx
```

After rebuilding the address table, the function responsible for decoding functions is called.

```

decode_10002AC0 proc near
    push    ebx
    push    edi
    push    esi
    push    ebp
    pop     eax
    shl     ecx, 2
    sub     eax, ecx
    push    eax
    mov     edi, [eax]
    pop     ecx
    dec     ecx
    dec     ecx
    dec     ecx
    dec     ecx
    mov     eax, [ecx]
    sub     eax, edi
    push    eax
    mov     ecx, eax
    cld
    push    edi
    mov     eax, [ebp+4]
    add     eax, 0Ch
    mov     esi, eax
    push    10h
    pop     ebx
    push    ebx

loc_10002AE7:
    push    ecx
    mov     eax, esi
    inc     esi
    mov     edi, esi

```

In the exact same way that we go over deobfuscating the TrickLoader binary, we can do the same with this sample. This will then leave us with all of the decoded functions.

```

Release
sub_10001560
sub_100015D0
sub_10001680
sub_100018C0
sub_10001910
sub_10001960
sub_10001A60
sub_10001AB0
sub_10001B20
sub_10001B50
sub_10001BB0
sub_10001BF0
sub_10001C70
sub_10001D00
sub_10001D30
sub_10001D80
sub_10001DB0
sub_10001DF0
sub_10001E60
sub_10002220
sub_10002380
sub_10002550
sub_10002710
sub_100027E0
sub_10002820
sub_100028A0
sub_10002A10
sub_10002A70

```

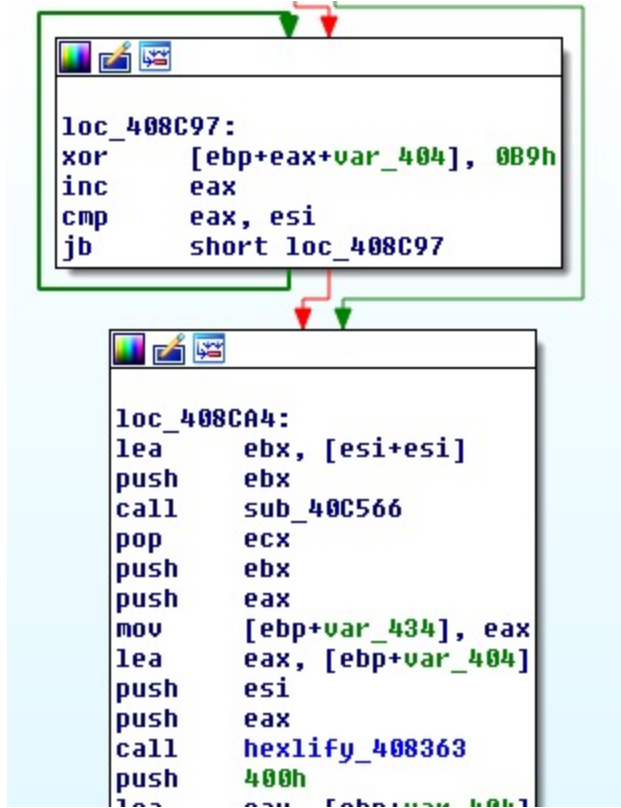
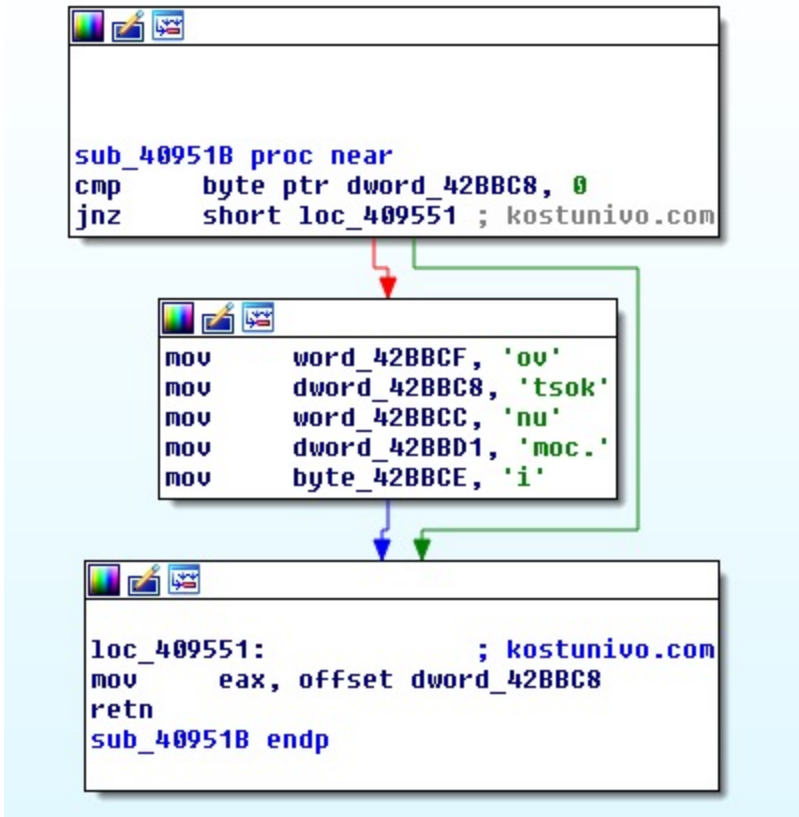
One of the decoded addresses from the table does not appear to be a function; in fact, much like TrickLoader, this is a LZO compressed PE file.

```

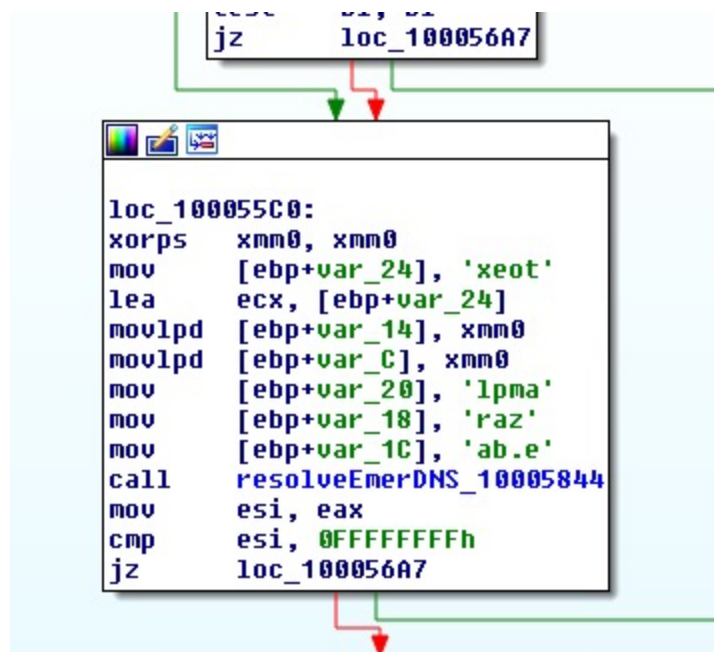
db 1Ah
db 4Dh ; M
db 5Ah ; Z
db 90h
db 0
db 3
db 0
db 0
db 4
db 4Eh ; N
db 0
db 0FFh
db 0FFh

```

After decompressing and analyzing the PE file, we discover this version of mexec is dropping the DNS variant of Anchor TrickBot[4].



Another variant recovered was also dropping the normal variant of Anchor TrickBot.



Pivoting on the decoding string of “futuresx.exe” from above also leads to a sandbox report[4] on an Anchor DNS sample. In VirusTotal, we can see this same hash was also delivered through the downloader variant:

Download URLs

This file has been spotted as the response content of the follo

.....
<http://172.82.152.15/cloudberry.exe>

<http://64.91.251.250/UgaNda73n>

In-the-wild file names

cloudberry.exe

Read the Full Report

See the report for the full list of IOCs and further details on the TrickBot executor module “mexec”.

[Read the Full Report](#)

References

- 1: [Deep Dive Into TrickBot Executor Module “mexec”: Hidden “Anchor” Bot Nexus Operations](#)
- 2: <https://blog.malwarebytes.com/threat-analysis/2016/10/trick-bot-dyrezas-successor/>
- 3: <https://www.fidelissecurity.com/threatgeek/archive/trickbot-we-missed-you-dyre/>
- 4: <https://www.sentinelone.com/wp-content/uploads/the-deadly-planeswalker-how-the-trickbot-group-united-high-tech-crimeware-apt/>