

Qbot is Back.Connect

--

By: Joshua Platt, Jason Reaves and Jonathan McCay

QBot is a modular information stealer also known as Qakbot or Pinkslipbot. It has been active since around 2007. It has historically been known as a banking Trojan, meaning that it steals financial data from infected systems, and a loader using C2 (Command and Control) servers for payload targeting and execution.

On May 30th, 2024 Law Enforcement action[1] was taken against the Qbot operators in a coordinated effort to disrupt their activities. But like most things, while the actions taken did disrupt the activity, new signs are showing off a re-emergence of the operators.

But before we get to the interesting connection, research recently emerged from ZScaler on the addition of DNS tunneling to Zloader[2]. Their analysis highlighted a sample that upon further review contained some curious overlaps:

SHA256: 22c5858ff8c7815c34b4386c3b4c83f2b8bb23502d153f5d8fb9f55bd784e764URL: vector123[.]xyz/PixelSignal.dllIP: 80.66.89.100

The IP contained a more interesting delivery with two overlaps:

hxxp://146.19.128.138/pack.dathxxps://80.66.89.100/pack.dat

A pivot into relations shows a ZIP file named 'pack.dat':

c8bddb338404a289ac3a9d6781d139314fab575eb0e6dd3f8e8c37410987e4de

Taking a look inside the ZIP:

Date	Time	Attr	Size	Compressed	Name	-----	----
-----	-----	-----	-----	-----	-----	2024-08-22 09:52:30
1096192	545458	winhttp64.dll	2023-11-01 17:17:28	6083072		
1930818	libcrypto-3-x64.dll	2023-11-01 17:17:28	776704	272981		
libssl-3-x64.dll	2024-11-13 03:28:05	4209176	1699649			
OneDriveStandaloneUpdater.exe	2024-11-19 13:53:12	1366528	1366738			
settingsbackup.dat	2024-11-13 12:01:31	1239040	283066	winhttp.dll	---	
-----	-----	-----	-----	-----	-----	2024-11-19	
13:53:12	14770712	6098710	6 files				

A few things can be quickly ascertained here; 3 of the files have newer dates than the others. Two of them have the same date while a .dat file has a later date. Assuming the DLL file we looked at earlier which is the 'winhttp.dll' file from this ZIP is loaded by the OneDriveStandaloneUpdater.exe then perhaps it will use the .dat file later. Also considering the PDB path of the DLL as something possibly intended to be side loaded.

OneDriveStandaloneUpdater.exe

F:\dbs\sh\odct\1021_111212\client\onedrive\Product\StandaloneUpdater\exe\obj\amd64\One

This file will in fact load the winhttp.dll file:

Z:\j\projects\dll_side\dll_side\x64\DebugDllOneDriveUpdater\dll_side.pdb

This DLL will load and decrypt the previously seen .dat file before deploying some sort of test message:

```
mov     [rbp+110h+var_10C], 0
mov     [rbp+110h+var_E8], 0
lea     rcx, aCalculator ; "calculator"
call    sub_180070F72
lea     rdx, [rbp+110h+var_10C]
lea     rcx, aSettingsbackup ; "settingsbackup.dat"
call    j_Decode_and_load_180075A70
xor     r9d, r9d          ; uType
lea     r8, Caption       ; "hi?"
lea     rdx, Text         ; "Hi, I am ok?"
xor     ecx, ecx          ; hWnd
call    cs:MessageBoxA
mov     cs:dword_18018F34C, 1
```

The 'Decode_and_load' function will open and read in the .dat file:

```
mov     [rbp+280h+var_27C], 0
mov     [rbp+280h+hFile], 0
mov     [rsp+2C0h+hTemplateFile], 0 ; hTemplate
mov     [rsp+2C0h+dwFlagsAndAttributes], 0 ; dw
mov     [rsp+2C0h+dwCreationDisposition], 3 ; d
xor     r9d, r9d          ; lpSecurityAttributes
mov     r8d, 1            ; dwShareMode
mov     edx, 80000000h     ; dwDesiredAccess
mov     rcx, [rbp+280h+lpFileName] ; lpFileName
call    cs:CreateFileW
mov     [rbp+280h+hFile], rax
```

Before RC4 decrypting it using a hardcoded 0x80 byte key:

```

loc_180075C4D:
lea     r8, [rbp+280h+var_1F0]
mov     edx, 80h
lea     rcx, Rc4_key_18015A000
call    j_rc4_init_180077A00
lea     r8, [rbp+280h+var_1F0]
mov     edx, [rbp+280h+var_27C]
mov     rcx, cs:qword_18018F340
call    j_RC4_decrypt_180077800
mov     r8d, [rbp+280h+var_27C]
mov     rdx, [rbp+280h+lpFileName]
lea     rcx, aReadpayloadf_3 ; "ReadPayloadFromDisc(): file '%S' loaded"...

```

Decrypting it ourselves shows it is a PE file:

```

>>> data = open('settingsbackup.dat', 'rb').read()>>> from Crypto.Cipher import
ARC4>>> rc4 = ARC4.new(key)>>> t = rc4.decrypt(data)>>>
t[:100]b'MZ\x90\x00\x03\x00\x00\x00\x04\x00\x00\x00\xff\xff\x00\x00\xb8\x00\x00\x00\x00
program cannot be'

```

Decoded PE file:

Z:\j\projects\bc_ssl\x64\DebugDLL\bc_ssl_client.pdb

New BackConnect

The general overview of the startup of the BC module client thread is to hook low level createprocess and exitprocess functions before heading to the main functionality:

```

lea     rcx, unk_1801C53A1
call    j_threadcheck_18008F070
call    HookExitProcess_180078CB6
call    HookCreateProcess_180077B22
call    j_main_180081260

```

Inside the main working function the module will look for running copies of itself:

```

rep stosu
lea rcx, unk_1801C53A1
call j_threadcheck_18008F070
call sub_18007A48F
call CheckRunningCopy_180078518
call sub_1800785A9
mov edx, 1Eh
lea rcx, aOut_connection ; "OUT_CONNECTION"
call sub_18007A598

```

```

loc_1800812AC:
xor eax, eax
cmp eax, 1
jz loc_1800813CB

```

```

mov [rbp+190h+var_18C], 0
lea rdx, aSoftwareTitanp ; "Software\\TitanPlus"
mov rcx, 0FFFFFFFF80000001h
call j_CheckRegKey_18008D4B0

```

```

loc_1800813CB:
xor eax, eax
mov edi, eax

```

Then begins a sleep loop that will check for a hardcoded registry key of 'Software\\TitanPlus':

```

loc_1800812AC:
xor eax, eax
cmp eax, 1
jz loc_1800813CB

```

```

mov [rbp+190h+var_18C], 0
lea rdx, aSoftwareTitanp ; "Software\\TitanPlus"
mov rcx, 0FFFFFFFF80000001h
call j_CheckRegKey_18008D4B0
mov [rbp+190h+var_168], rax
cmp [rbp+190h+var_168], 0
jnz short loc_180081307

```

```

loc_1800813CB:
xor eax, eax
mov edi, eax
lea rcx, [rbp+190h+var_180] ; Esp
lea rdx, stru_1801713D0 ; Fd
call j_RTC_CheckStackVars
mov eax, edi
lea rsp, [rbp+188h]
pop rdi
pop rbp
retn
main_180081260 endp

```

```

lea rdx, aSoftwareTitanp ; "Software\\TitanPlus"
mov rcx, 0FFFFFFFF80000002h
call j_CheckRegKey_18008D4B0
mov [rbp+190h+var_168], rax
cmp [rbp+190h+var_168], 0
jnz short loc_180081307

```

```

307:
, [rbp+190h+var_18C]
, 3Bh

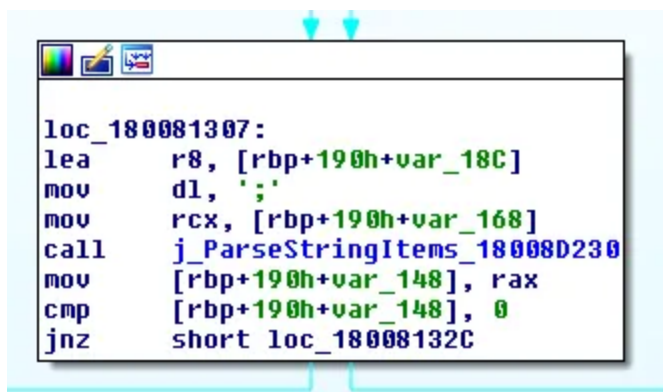
```

```

mov ecx, 1388h ; dwMilliSecon
call cs:Sleep
jmp short loc_1800812AC

```

This value will be sent to a function labeled for parsing a string by a semicolon character:



```
loc_180081307:
lea     r8, [rbp+190h+var_18C]
mov     dl, ';'
mov     rcx, [rbp+190h+var_168]
call    j_ParseStringItems_18008D230
mov     [rbp+190h+var_148], rax
cmp     [rbp+190h+var_148], 0
jnz     short loc_18008132C
```

Eventually these values will be passed off to the main worker function for 'nattun_client_loop':

```
mov     [rbp+0AA0h+var_A9C], 0FFFFFFFFh
lea     rcx, aNattun_clien_9 ; "nattun_client_loop(): start work!"
call    sub_18007A598
mov     rcx, [rbp+0AA0h+arg_0]
call    sub_18007853B
test    eax, eax
jge     short loc_180081CF7
```

When the client connects in it will send in information about the infected system:



```
push    rax
sub     rsp, 1B8h
lea     rbp, [rsp+30h]
lea     rcx, unk_1801C5B98
call    j_threadcheck_18008F070
mov     [rbp+190h+var_188], 0
call    GetDnsServers_180078EC8
mov     [rbp+190h+var_168], rax
call    GetPrimaryDomainName_1800794FE
mov     [rbp+190h+var_148], rax
call    GetLogonServer_1800796E8
mov     [rbp+190h+var_128], rax
call    GetNetworkInterfaces_18007742E
mov     [rbp+190h+var_108], rax
cmp     [rbp+190h+var_148], 0
jz      short loc_18008BE7E
```

While pivoting on the PDB information we managed to find a number of interesting files with references to Qbot.

4b4398f64e574cfdb8de05d388d97ed255e888045f0316808311f51f63212efbZ:\j\projects\qbot4\te
7215d9421e0a6d1a7cfde3f6d742670550fed009585ab35b53cbb845f63c5f74Z:\j\projects\qbot4\Re

The references on this qd_x86 file:

hxxps://upd5[.pro/update/qd_x86.exe

This file looks like it might be some sort of a debugging tool:

```
Usage: %S <-t | -T | -s | -i | -c cmd | -h | -n base_random_name | -l seclog_file brn
| -L seclog_blz_crypted_file dst_file passphrase> | -dm file | - du file>      -t
send shutdown command to bot          -T          stop bot, and clean config      -s
run as test server                    -c cmd       send server command 'cmd'       -n
[base_random_name]                    generate nick for local host or from base_random_name      -l
seclog_file [base_random_name]        decrypt seclog to stdout                  -L seclog_file
decrypt blzipped seclog to stdout      -i          decrypt current config to stdout
-b          check Update running        -dm file    drop saved main dll to disk      -du
file      drop update dll to disk      -h          print this help
```

A certain string related to a command can be traced back to the law enforcement takedown of Qbot.[3]

QPCMD_BOT_SHUTDOWN sent ok.

In addition to the new backConnect malware developed by Qbot operators, research has emerged tying zloader[4] activity to that of the BlackBasta ransomware operation. It is highly likely this new side loading backConnect malware has been or is going to be utilized to further ransomware attacks. We have released a yara rule in order to better identify the samples and help with detections.

IOCs

```
SHA256: 22c5858ff8c7815c34b4386c3b4c83f2b8bb23502d153f5d8fb9f55bd784e764
SHA256: 98d38282563c1fd09444724eacf5283626aeef36bcb3efa9d7a667db7314d81f
SHA256: c8bddb338404a289ac3a9d6781d139314fab575eb0e6dd3f8e8c37410987e4de
SHA256: bf861f5bd384707e23148716240822208ceeba50c132fb172b784a6575e5e555
SHA256: 9cdef45dc9f7c667a54effa9b8187ef128d64ea49c97bdae4e9567d866c63f5a
SHA256: 651e49a45b573bb39e21746cb99fcd5d17679e87e04201f4cc6ca10ff2d166e4
SHA256: 4cad17ef867f03081eb690b1c16d7f4d5c937c3f20726af0442d7274413e3620
SHA256: a197804c6ae915f59add068e862945b79916c92a508c0287a97db718e72280a3
vector123[.]xyz/PixelSignal.dll
upd5[.]pro
146.19.128[.]138/pack.dat
80.66.89[.]100/pack.dat
80.66.89[.]100
146.19.128[.]138
```

Registry:Software\\TitanPlus

YARA

```
rule new_bc{strings:$a1 =
{4a6869736864694932556873766f6339346b65696f6a6e376e7331396d30646f}condition:all of
them}
```

References

1: <https://operation-endgame.com/>

2: <https://www.zscaler.com/blogs/security-research/inside-zloader-s-latest-trick-dns-tunneling>

3: <https://www.secureworks.com/blog/law-enforcement-takes-down-qakbot>

4: <https://www.rapid7.com/blog/post/2024/12/04/black-basta-ransomware-campaign-drops-zbot-darkgate-and-custom-malware/>