New Variant of QakBot Being Spread by HTML File Attached to Phishing Emails

fortinet.com/blog/threat-research/new-variant-of-qakbot-spread-by-phishing-emails

July 19, 2022

Fortinet's FortiGuard Labs captured a phishing email as part of a phishing campaign spreading a new variant of QakBot. Also known as QBot, QuackBot, or Pinkslipbot, QakBot is an information stealer and banking Trojan that has been captured and analyzed by security researchers since 2007.

I performed a deep analysis on this phishing campaign and the new QakBot variant using the captured email. In this analysis, you will learn how the attached HTML file leads to downloading and executing the new QakBot variant, what actions it takes on the victim's device, and how it sends the collected data from the victim's device to its C2 server.

Affected platforms: Microsoft Windows
Impacted parties: Microsoft Windows Users
Impact: Controls victim's device and collects sensitive information
Severity level: Critical

Phishing Email and the Attached HTML File

Figure 1.1 shows the phishing email used by hackers to lure the recipient into opening the attached HTML file (ScannedDocs_1586212494.html). This phishing email has been marked as SPAM by Fortinet's FortiMail.

Figure 1.1 – Display of the captured phishing email

The HTML file contains a piece of javascript code that is automatically executed once it is opened in a web browser by the recipient. It decodes a base64 string held by a local variable. It then calls a built-in function, navigator.msSaveOrOpenBlob(), to save the base64 decoded data (a ZIP archive) to a local file named "ScannedDocs_1586212494.zip". Figure 1.2 shows the defined variables with the base64 string and the ZIP file name.

Figure 1.2 – The javascript snippet code inside the HTML file

Figure 1.3 is a screenshot of a Microsoft Edge browser opening the HTML file. As you can see, the ZIP archive has been automatically saved onto the victim's device.

Figure 1.3 – The HTML file opened in the Microsoft Edge browser

Downloading and executing QakBot

Next, we'll look at what's inside the downloaded ZIP archive. It's a Windows shortcut file – "ScannedDocs_1586212494.lnk". As you may know, a Windows shortcut file can execute commands by putting them into the Target field. Figure 2.1 shows a screenshot of this shortcut file and its properties.

Figure 2.1 – The Windows shortcut file and properties

The shortcut is disguised with a Microsoft Write icon to trick the victim into thinking it's a safe text file so they will open it. As for its properties, a group of commands in the target field will be executed by "cmd.exe". When the victim double clicks the file, the commands get executed.

According to the commands found in Figure 2.1, it mainly runs "cURL" (Client URL) to download a file from URL 194[.]36[.]191[.]227/%random%.dat into local file "%ProgramData%\Flop\Tres.dod". cURL is a popular Linux tool, but it has also been part of Windows as a default program since Windows 10.

The downloaded file (“Tres.dod”) is a DLL file. By my analysis, it is a sort of QakBot’s loader program. In this case, “regsvr32” is in charge of executing it using the command “regsvr32 %ProgramData%\Flop\Tres.dod”.

Figure 2.2 – A view of the QakBot Loader Module’s Resource section

The QakBot Loader Module (Tres.dod) that runs in “regsvr32.exe” loads a binary block from its Resource section with the name “AAA”, as shown in Figure 2.2. It proceeds to decrypt the binary block to get a fileless PE file and a piece of dynamic code that is a kind of self-deployment function. It is then called by the Loader Module to deploy the fileless PE file, which is the core module of QakBot, inside the “regsvr32” process. After the core module of QakBot is deployed, the last task of the self-deployment function is to call its entry point. Figure 2.3 explains how the self-deployment function calls the entry point of the QakBot core module.

Figure 2.3 – Self-deployment function about to call the entry point of QakBot

Process Hollowing

Malware usually performs process hollowing to inject malicious code or modules into another process. It does this to evade being detected.

Depending on the affected machine’s platform (32-bit or 64-bit) and installed anti-virus software, QakBot will select a system process from a process list as the target process for performing process hollowing. This list includes OneDriveSetup.exe, explorer.exe, mobsync.exe, msra.exe, and iexplore.exe for this variant.

In my testing environment, it picked “OneDriveSetup.exe”. QakBot then calls the API CreateProcessW() to start a new process using the creation flag CREATE_SUSPENDED so it gets suspended at start. It can then modify its memory data, like carrying the QakBot core module onto the newly-created “OneDriveSetup.exe” process by calling API WriteProcessMemory(). Next, it modifies the code at the entry point of the new process to jump to the injected core module. It eventually calls the API ResumeThread() to resume the new process, and QakBot is then executed in the target process.

Figure 3.1 shows a process tree with all relevant processes from downloading QakBot Loader (“curl.exe”) to “OneDriveSetup.exe”.

Figure 3.1 – Overview of the process tree for relevant processes

Anti-analysis technique

Before analyzing QakBot’s core module, let’s go through some of the anti-analysis techniques that QakBot uses to prevent itself from being easily analyzed.

Constant strings are encrypted

Constant strings are useful information for researchers to analyze code. QakBot holds encrypted constant strings, which are only decrypted by a particular function before using. Figure 4.1 is an example of obtaining a constant string, "Mozilla/5.0 (Windows NT 6.1; rv:77.0) Gecko/20100101 Firefox/77.0", through the function at 609DD8 by string index 0xA8.

Figure 4.1 – Example of one decrypted constant string by a function

Dynamically obtaining key Windows APIs

Most Windows APIs are obtained during QakBot run-time. It is hard to guess which API is called until executing the instruction. Below is an instance of calling the API CreateThread(), where dword_61F818 is a dynamically loaded function table, whose offset +74H is the function of CreateThread().

```
xor ecx, ecx
lea eax, [ebp+var_4]
push eax
mov eax, dword_61F818 ; Function table of Kernel32.dll
push ecx
push ecx
push offset thread_fun
push ecx
push ecx
mov [ebp+var_4], ecx
call dword ptr [eax+74h] ; =>; CreateThread
mov dword_61F83C, eax
test eax, eax
```

Detecting Analysis Tools

QakBot has a thread function that checks once per second to see if any analysis tool is running on the affected machine. To do this, it predefines a process name list of some analysis tools, which of course, is a decrypted constant string. Once any of them matches one of the running processes, it will affect QakBot's workflow (say, never connecting to a C2 server).

Here is the predefined process name list:

frida-winjector-helper-32.exe, frida-winjector-helper-64.exe, tcpdump.exe,windump.exe, ethereal.exe, wireshark.exe, ettercap.exe;rtsniff.exe, packetcapture.exe, capturenet.exe, qak_proxy;dumpcap.exe, CFF Explorer.exe, not_rundll32.exe, ProcessHacker.exe, tcpview.exe, filemon.exe,

procmon.exe;idaq64.exe, PETools.exe, ImportREC.exe, LordPE.exe, SysInspector.exe, proc_analyzer.exe, sysAnalyzer.exe, sniff_hit.exe, joeboxcontrol.exe, joeboxserver.exe, ResourceHacker.exe, x64dbg.exe, Fiddler.exe, sniff_hit.exe, sysAnalyzer.exe

According to the above process list, I determined that the analysis tools include, but are not limited to:

Joe Sandbox, TcpDump, WinPcap, Wireshark, Ettercap, PacketCapture, CaptureNet, CFF Explorer, ProcessHacker, TcpView, FileMon, ProcMon, IDA pro, PETools, ImportREC, LordPE, SysInspector, SysAnalyzer, ResourceHacker, x64dbg, and Fiddler.

QakBot's Core Module Connects to C2 Server

As long as the QakBot core module is resumed in the target process (such as "OneDriveSetup.exe"), it starts using another entry function other than the one in regsvr32.exe.

As per QakBot tradition, it uses many threads to perform its tasks. One is to collect information about the affected device and send it to its C2 server.

The core module has two binary data blocks in its Resource section, named "102" and "103". The data of Resource "103" is an RC4 encrypted configuration. After decryption, it is the string "10=obama189\r\n3=1655107308\r\n". "obama189" is a QakBot ID of this variant, and "1655107308" is a Unix Epoch time.

The "102" Resource data is an RC4 encrypted C2 server list. "\System32\WindowsPowerShel1\v1.0\powershel1.exe" is a constant string that generates an Rc4 key buffer to decrypt the C2 server data.

Figure 5.1 – Screenshot of partially decrypted "102" Resource

The top section of Figure 5.1 shows it was about to call the API FindResourceW with the Resource name "102", while the bottom section is a partial list of the decrypted binary IP and Port of the C2 server. There are 123 IP and Port pairs inside this variant.

QakBot goes through all listed C2 servers, one by one, until a connection is established. It then sends the victim registry packet (the first packet) to that C2 server to register the victim. The plain text of the registry packet is:

"{\"2\":\"hrzpxm292261\",\"8\":9,\"1\":18}"

The keys are string numbers, like “2”, “8”, and “1”. The value of “2” is “hrzpxm292261” (the victim’s ID) that was generated using hardware information, the value of key “8” specifies the packet type (it’s 9 for this packet), and “1”’s value is 18, which is the QakBot version.

The packet is RC4 encrypted and then encoded as a string using a base64 algorithm. All the packets between QakBot and the C2 server are sealed in a JSON structure.

It then sends the data to its C2 server using the HTTP Post method with URL “/t4” and the base64 encoded registry data as the body and being transported over SSL protocol. Figure 5.2 shows a screenshot of an analysis tool with the sent packet on the left and response data on the right.

Figure 5.2 – The view of the registry packet in an analysis tool

It takes the reverse path to restore the response data to plain text, which is base64 decoding and RC4 decryption.

“{\”8\”:5,\”16\”:3257495567,\”39\”:\”vLLO\”,\”38\”:1}” is the plain text for this case, which will set or update the value of some local variables.

Sending Sensitive Data to the C2 Server

QakBot collects sensitive data from the victim’s device and sends it to its C2 server. Similarly, the hacker could transfer corresponding sub-modules to the QakBot client to be executed on the victim’s device.

QakBot leverages Windows APIs, Windows commands, and WMI Query Language (WQL) to obtain the information. Below are the details.

Window APIs

API Function	Description
GetVersionEx()	Windows edition information, including build number, such as “10.0.1.19043.0.0.0100” for my testing system running Windows 10.
GetComputerNameW()	Computer name, like “DESKTOP-P952NC4”.
GetSystemMetrics()	Obtain screen size (width and height).
NetGetJoinInformation()	Retrieve the AD Domain, like “WORKGROUP”.

LookupAccountSidW()	The User name.
GetSystemInfo()	Processor Architecture.
CreateToolhelp32Snapshot(), Process32FirstW(), Process32NextW()	Obtain running process information.
GetModuleFileNameW()	The full path of QakBot and the full path of the target process.
CreatProcessW()	Execute Windows commands.

WMI Object Query

Query String	Description
SELECT * FROM Win32_OperatingSystem	OS information.
SELECT * FROM AntiVirusProduct	Obtain the installed AntiVirus software, like Microsoft Defender or FortiClient.
SELECT * FROM Win32_Processor	CPU processor information.
SELECT * FROM Win32_ComputerSystem	System environment information, like Model, Domain, Manufacturer, etc.
SELECT * FROM Win32_Bios	Device’s BIOS information.
SELECT * FROM Win32_DiskDrive	Hard disk information, like Partitions, Size, and Model.
SELECT * FROM Win32_PhysicalMemory	Physical RAM sticks’ detailed information, such as capacity, clock speed, and channel.

SELECT Caption,Description,Vendor,Version, InstallDate,InstallSource,PackageName FROM Win32_Product	Installed software information.
SELECT Caption,Description,DeviceID, Manufacturer,Name,PNPDeviceID,Service, Status FROM Win32_PnPEntity	Properties of Plug and Play devices, like Keyboard, Mouse, CD-ROM, Network adapter, and more.

Windows Commands

Commands	**Description**
"ipconfig /all"	All TCP/IP network configuration values.
"nslookup -querytype=ALL -timeout=12 _ldap._tcp.dc._msdcs.%s"	Query SRV records for the domain from the main DNS server of the victim's device.
"nltest /domain_trusts /all_trusts"	Enumerating domain trusts.
"net share"	Shared resources and names.
"route print"	Active routes' tables.
"netstat -nao"	Active connections in the victim's device.
"net localgroup"	Local groups information.
"qwinsta"	Active sessions on the victim's device.
"arp -a"	Information about ARP entries.
"net view /all"	Display all the shares on a remote computer.

Once QakBot has collected all the information shown in the above tables, it seals the information inside a packet with packet type “8”:4. Figure 6.1 shows the JSON data of this packet, which was about to call the RC4 encryption function.

Figure 6.1 – Plain text of packet “8”:4 with sensitive information

Although this QakBot successfully established connections with its C2 server, I have not received any sub-modules. I’m still monitoring the communication and will update this analysis if I get something interesting.

Conclusion

According to this analysis, I proved that an attached HTML file is no safer than any other risky files (like MS Word, MS Excel, PDF, and so on). You have to be extra cautious when receiving emails with attachments.

I then explained how the HTML file drops a ZIP archive through a piece of auto-execution JavaScript code. Later, I focused on how a disguised Windows shortcut file downloads the loader module of QakBot.

You also learned what the loader module does to decrypt and deploy the core module of QakBot in a picked target process (“OneDriveSetup.exe” for this time).

And finally, we walked through QakBot starting threads to connect to its C2 server using an IP address and port pair chosen from a C2 server list that had been decrypted from its Resource “102”, as well as what sensitive data it retrieved from the victim’s device and then submitted to its C2 server..

Fortinet Protections

Fortinet customers are already protected from this malware through FortiGuard’s Web Filtering, AntiVirus, FortiMail, FortiClient, and FortiEDR services, as follows:

The phishing email was detected as "**SPAM**" by the FortiMail service.

The URL to download QakBot and its C2 servers has been rated as "**Malicious Websites**" by the FortiGuard Web Filtering service.

The HTML file attached to the phishing email and the downloaded QakBot Loader module are detected as "**JS/Agent.BLOB!tr**" and "**W32/Qbot.D!tr**" and are blocked by the FortiGuard Antivirus service.

FortiEDR detects the involved file as malicious based on its behavior.

In addition to these protections, we suggest that organizations have their end users also go through the FREE NSE training: NSE 1 – Information Security Awareness. It includes a module on Internet threats designed to help end users learn how to identify and protect themselves from phishing attacks.

IOCs

URLs:

194[.]36[.]191[.]227/%random%.dat

Click here for the complete C2 server list

Sample SHA-256 Involved in the Campaign:

[Attached HTML file]

FE1043A63E6F0A6FAA762771FF0C82F253E979E6E3F4ADD1C26A7BD0C4B2E14C

[Loader module of QakBot]

9C3D3CD9B0FCB39117692600A7296B68DDDF2995C6D302BC9D9C8B786780BA19

[ScannedDocs_1586212494.lnk]

F5B6619E92D7C4698733D9514DF62AFACA99883DFAC8B9EE32A07D087F2800BF

Learn more about Fortinet's FortiGuard Labs threat research and intelligence organization and the FortiGuard Security Subscriptions and Services portfolio.