Qakbot Returns

Published: 2024-01-04 · Archived: 2026-04-05 15:28:33 UTC

The Qakbot malware has reappeared just four months after law enforcement disrupted its distribution in the “Duck Hunt” operation. Lately, various security companies have noticed the malware spreading through phishing emails. Microsoft, which discovered this, described it as a small-scale campaign starting on December 11, 2023, specifically targeting the hospitality industry. Although the number of these emails is currently low, given Qakbot’s past persistence, it’s anticipated that the volume will rise soon. We got our hands on one such sample by this tweet.

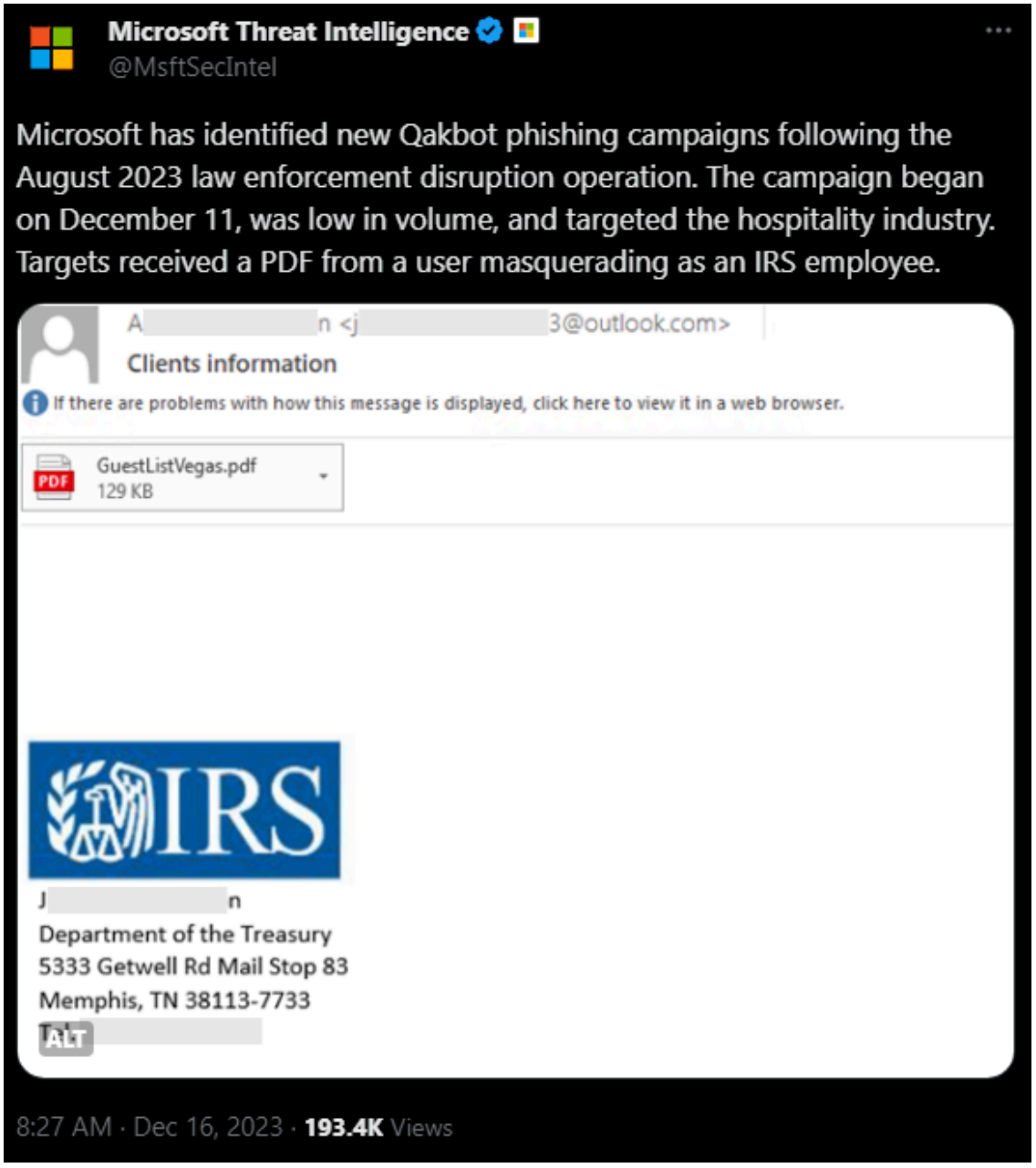

Figure 1: Microsoft discovery of Qakbot resurface

Binary Analysis

As per Microsoft's tweet, in the recent campaign, an MSI file is being downloaded to the user's machine from the malicious PDFs which were spread through phishing mails.

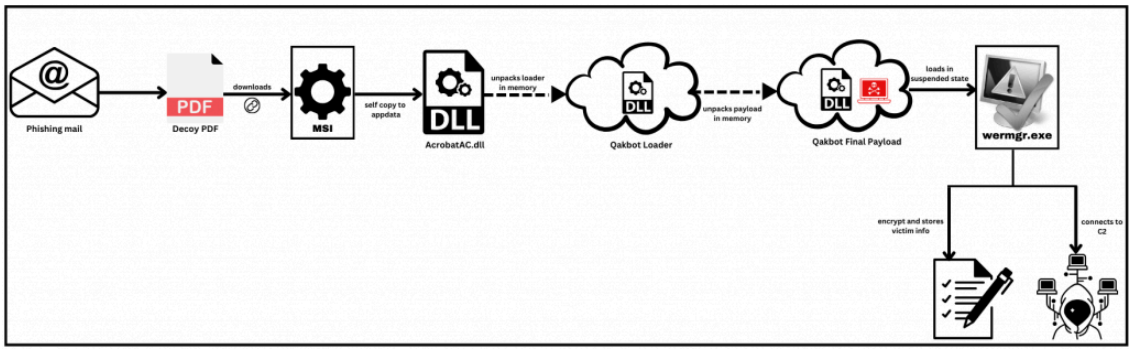

Figure 2: Execution Flow

On analysing the MSI file, we found that the suspicious DLL compressed inside was a patched IDM (Internet Download Manager) DLL with Qakbot inside.

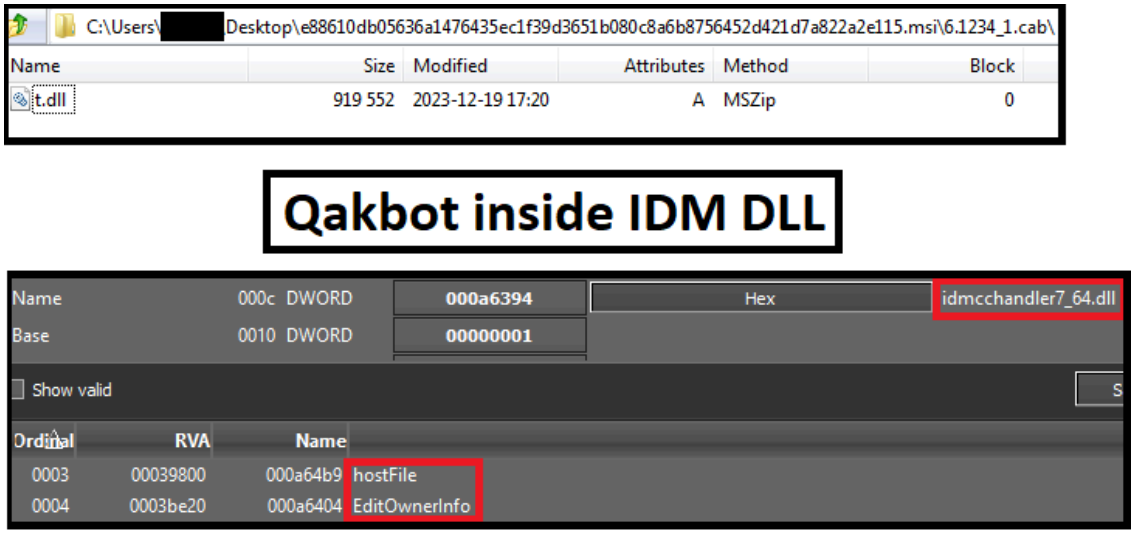

Figure 3: Qakbot inside IDM DLL

We found that this DLL was packed with a custom packer. Usually unpacking the Qakbot DLL is quite simple. It uses **VirtualAlloc()** to allocate memory to unpacked code and **VirtualProtect()** to change the protection on a memory region. We set breakpoints on both of those APIs to unpack. We first got the dump of the PE file without the MZ header. Later, we found that it was the Qakbot second stage loader by manually adding the MZ header. The threat actor employs this method to avoid detection by EDR, as it scans memory regions for MZ headers to identify potential process injection methods.

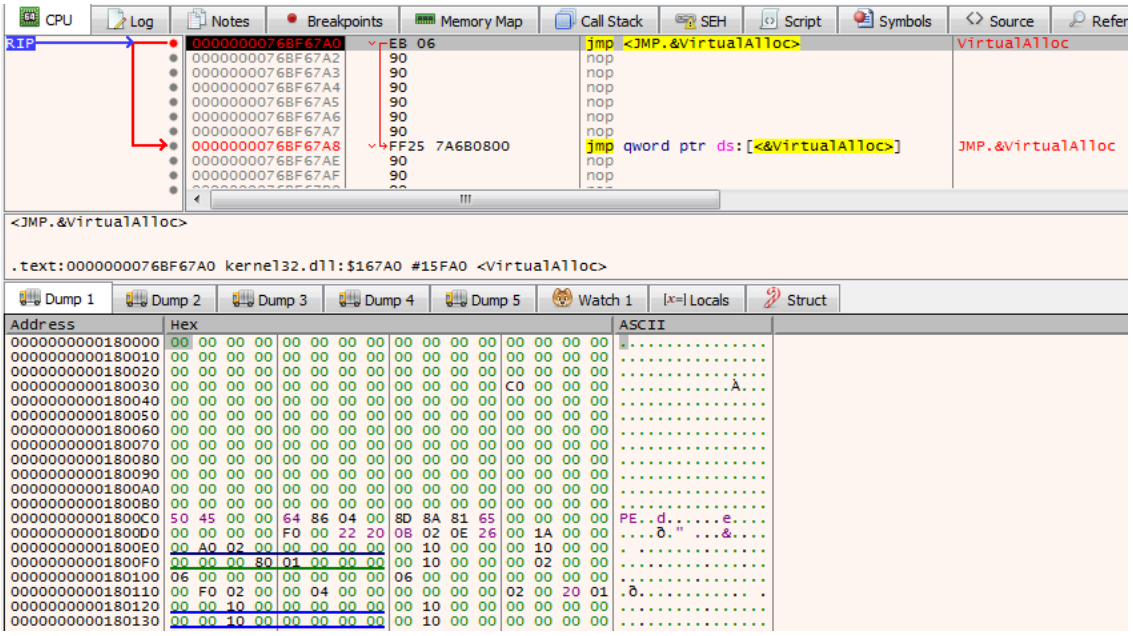

Figure 4: Unpacking Qakbot

On further unpacking, we got the final Qakbot payload which loads from memory while executing. Some security researchers found that in the new campaign, Qakbot uses AES encryption to encrypt and store victim information but the final payload we got was the usual Qakbot payload with the same RC4 encryption.

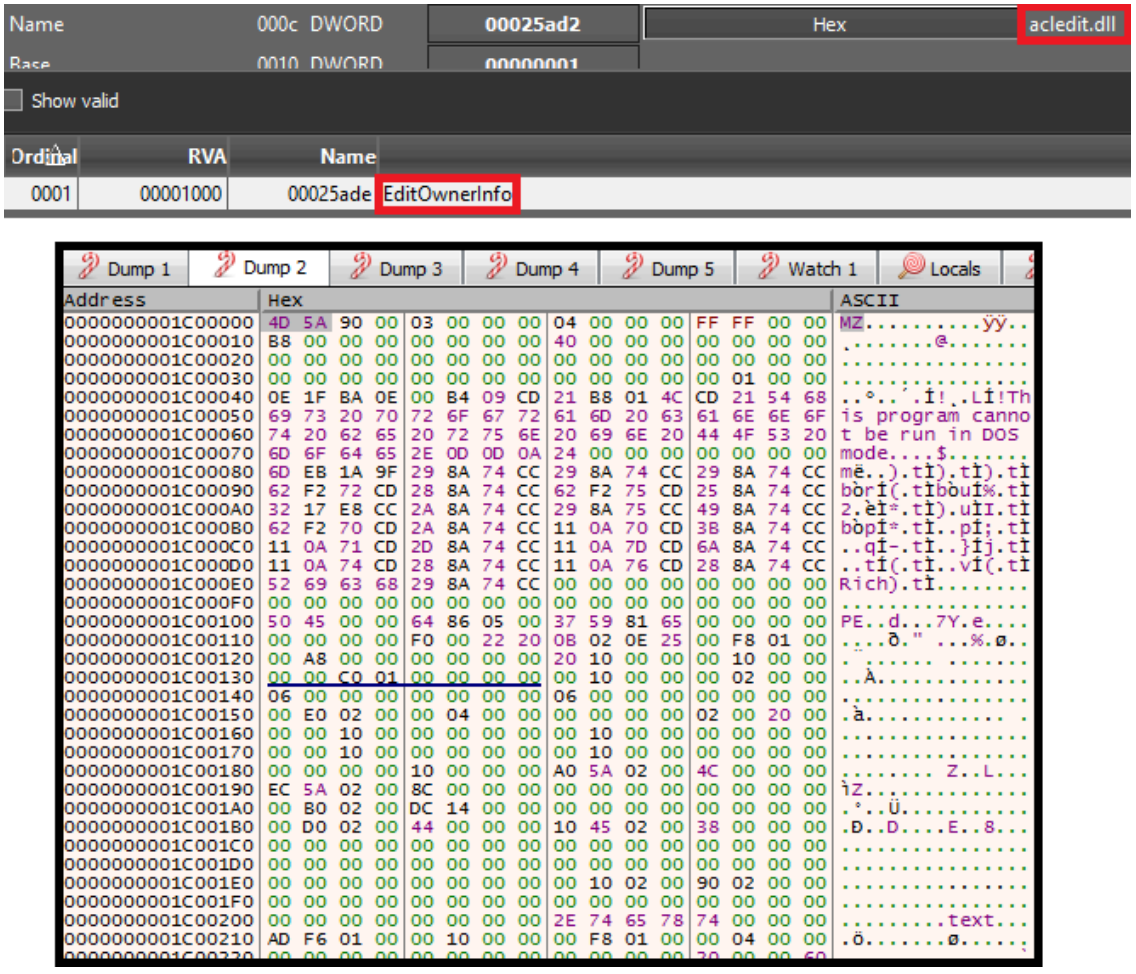

Figure 5: Final Qakbot payload

On dynamic analysis, the MSI drops an installer temp file which passes the command line to invoke rundll32.exe and hides the window to run in background.

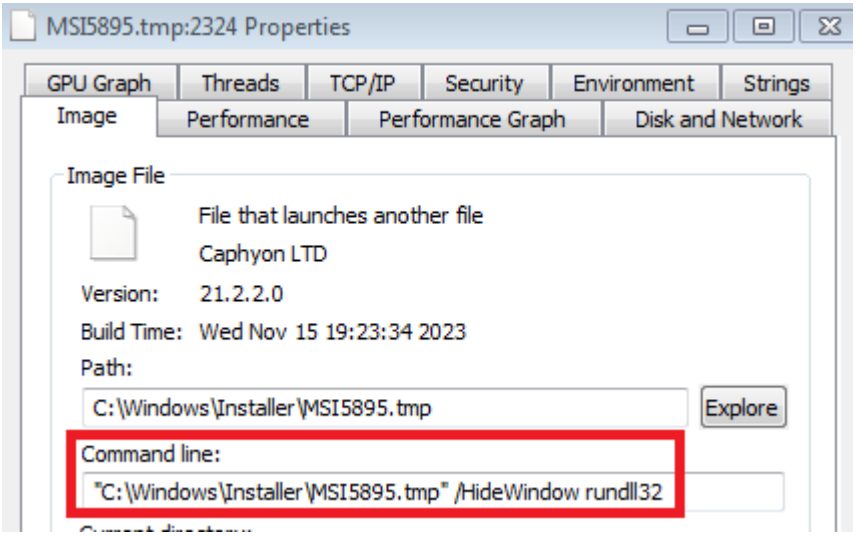

Figure 6: MSI installer

Since the threat actor uses PDF in their kill chain, the malicious DLL self copies itself in the name of **AcrobatAC.dll** and passes the command line arguments to execute the DLL with Qakbot export function **EditOwnerInfo**.

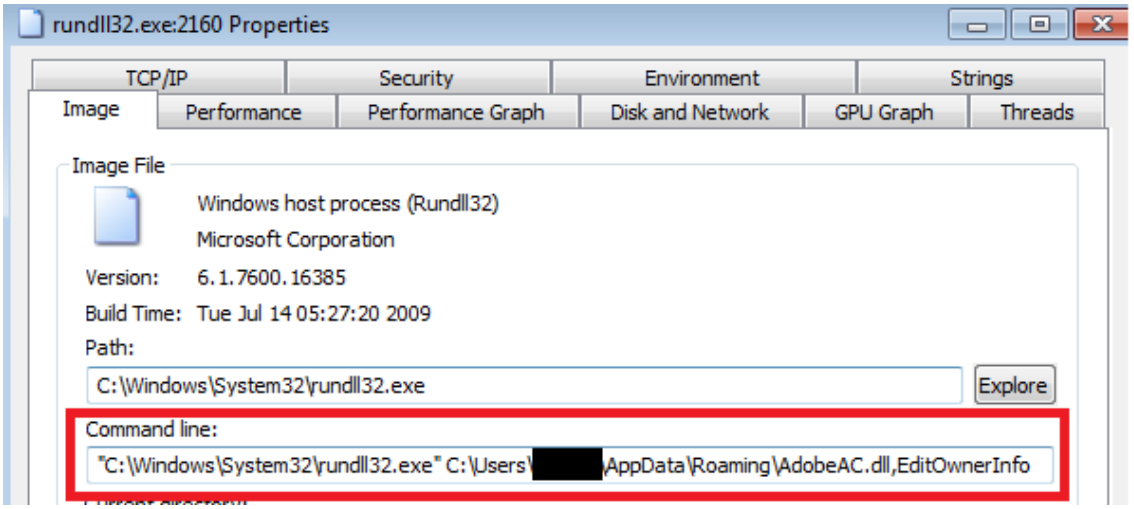

Figure 7: Malicious DLL running on background

It showed the dummy Acrobat window and fake error window as a decoy. Further we found that the malicious DLL invokes the *wermgr.exe* – Windows Error Manager in suspended state to pursue its kill chain.

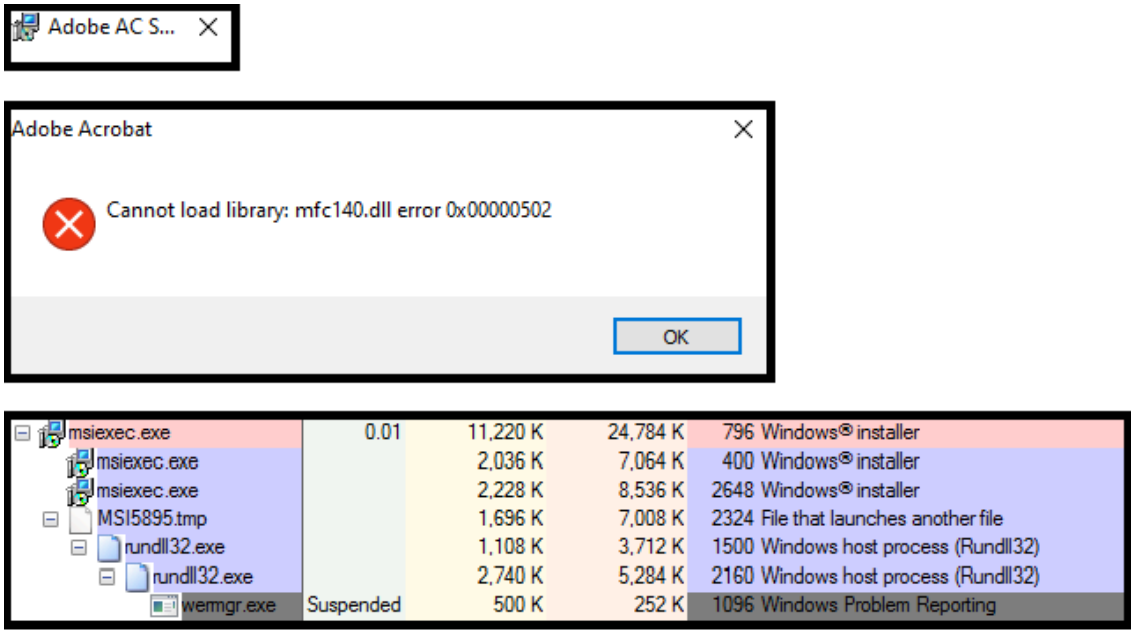

Figure 8: Decoy and invoking wermgr.exe

We dumped the PE file from *wermgr.exe* which was our previously unpacked final Qakbot payload. The threat actor implied **Process Hollowing** technique to inject malicious code into the suspended process of Windows Error Manager.

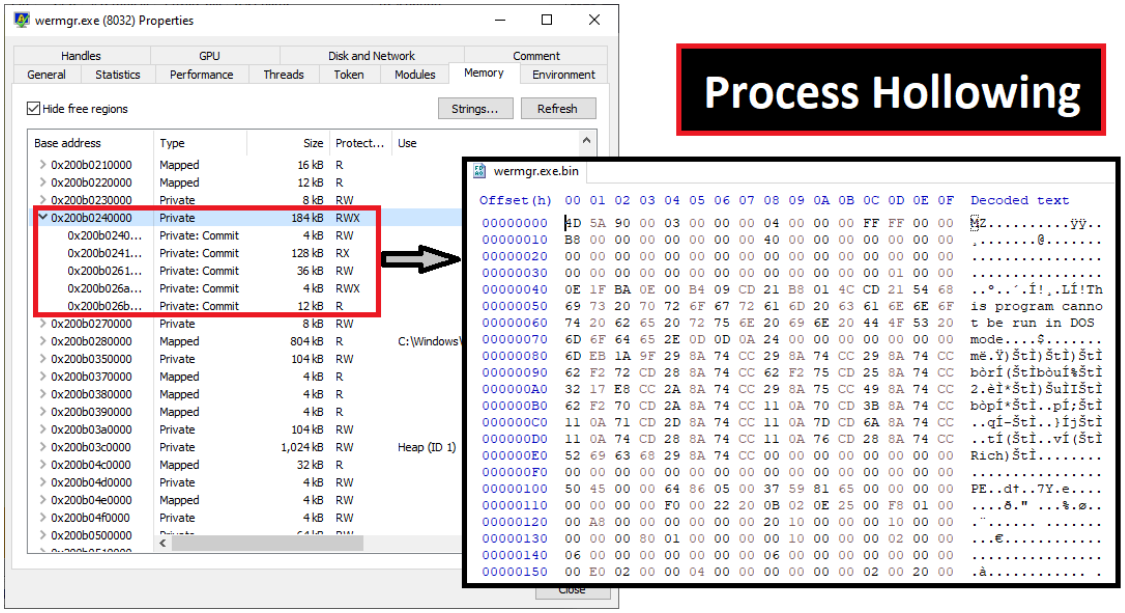

Figure 9: Process Hollowing wermgr.exe

As mentioned earlier, the *wermgr.exe* creates a registry key with RC4 encrypted data of victim system information, timestamp of installation and C2 information which is a usual Qakbot TTP.

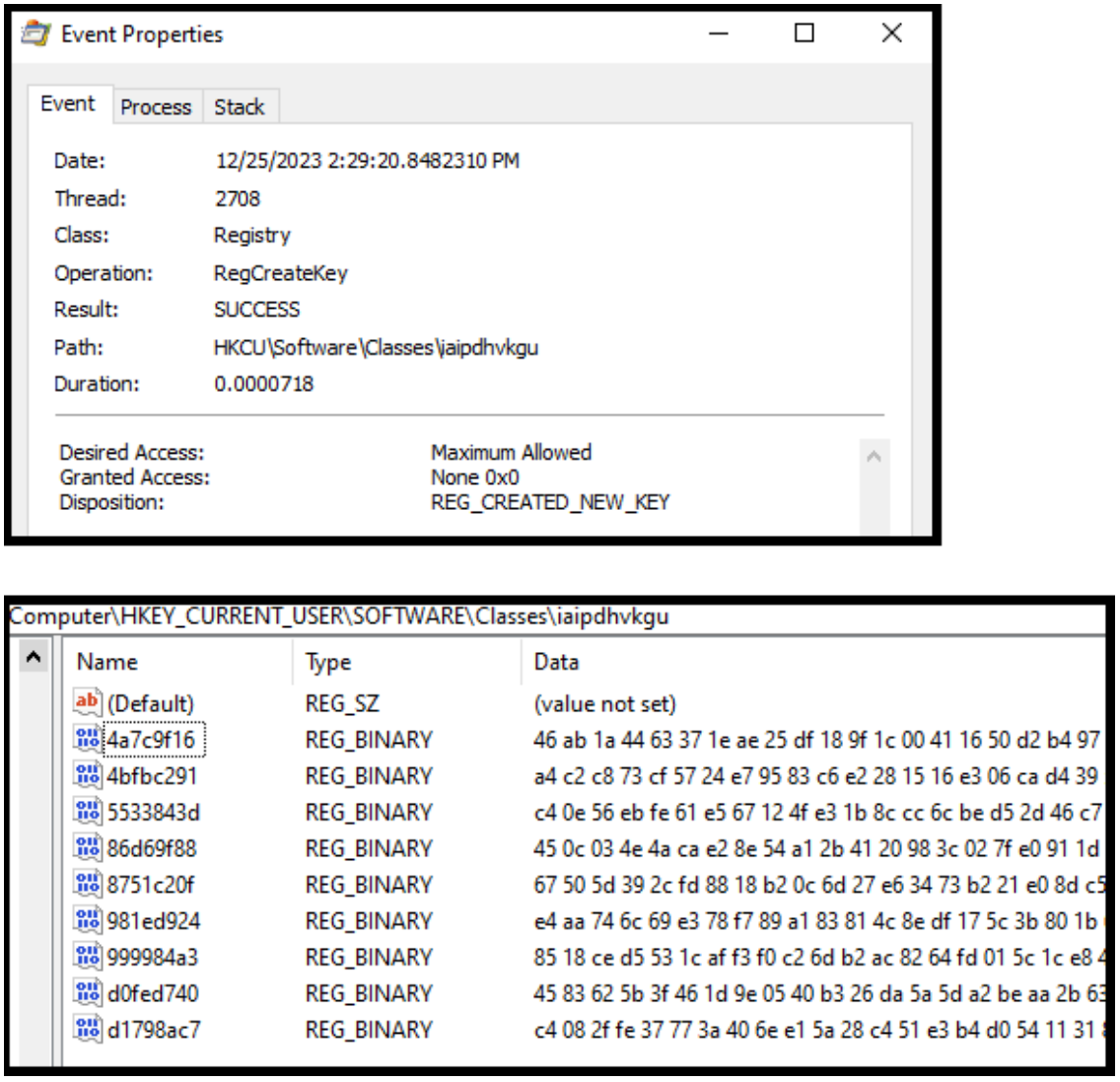

Figure 10: Creates registry key

Qakbot tries to make a C2 connection in the background when the victim believes *wermgr.exe* is running. Since the C2 was down at the time of analysis, it was unable to establish a connection for carrying out any further malicious activity.

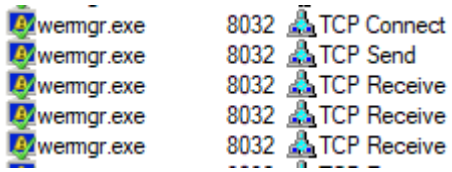

Figure 11: C2 connection by wermgr.exe

We at K7 Labs provide detection against latest threats and also for this newer variant of Qakbot. Users are advised to use a reliable security product such as “K7 Total Security” and keep it up-to-date so as to safeguard their devices.

IoCs

Hash	Detection Name

723DAE8ED3F157E40635681F028328E6	Backdoor (005af9cf1)
88BBF2A743BAAF81F7A312BE61F90D76	Backdoor (005af9cf1)

References

1. https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/decrypting-qakbots-encrypted-registry-keys/
2. https://n1ght-w0lf.github.io/malware%20analysis/qbot-banking-trojan/
