Qakbot Series: Process Injection

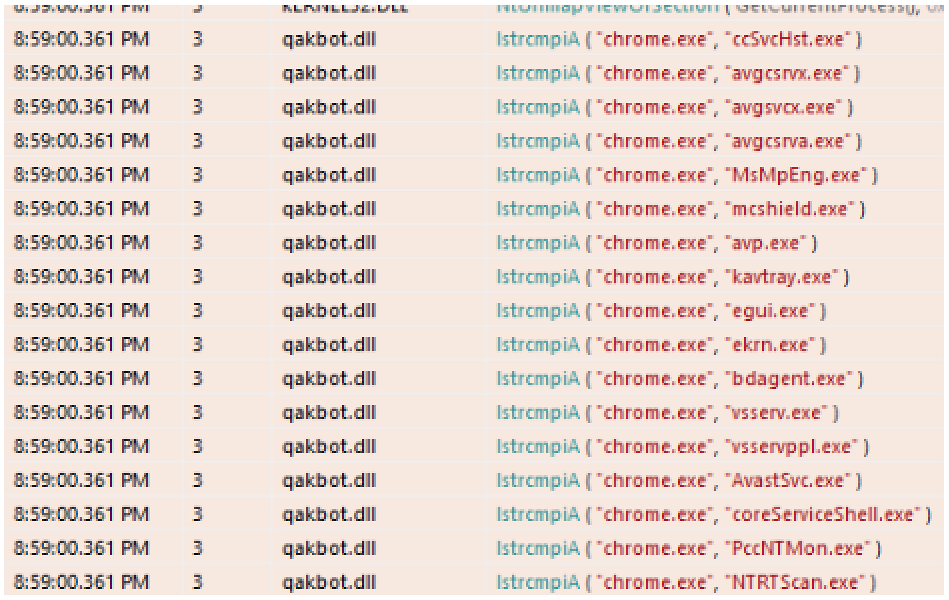
malwarology.com/2022/04/qakbot-series-process-injection/

April 16, 2022

2022-04-16

Malware Analysis , Qakbot

In late March 2022, I was requested to analyze a software artifact. It was an instance of Qakbot, a modular information stealer known since 2007. Differently to other analyses I do as part of my daily job, in this particular case I can disclose wide parts of it with you readers. I'm addressing them in a post series. Here, I'll discuss about the Qakbot process injection techinque based on this specific sample.



```
8:59:00.361 PM   3   KERNEL32.DLL   NtUnmapViewOfSection ( GetCurrentProcess(), 0x
8:59:00.361 PM   3   qakbot.dll     lstrcmpiA ( "chrome.exe", "ccSvcHst.exe" )
8:59:00.361 PM   3   qakbot.dll     lstrcmpiA ( "chrome.exe", "avgcsrvx.exe" )
8:59:00.361 PM   3   qakbot.dll     lstrcmpiA ( "chrome.exe", "avgsvcx.exe" )
8:59:00.361 PM   3   qakbot.dll     lstrcmpiA ( "chrome.exe", "avgcsrva.exe" )
8:59:00.361 PM   3   qakbot.dll     lstrcmpiA ( "chrome.exe", "MsMpEng.exe" )
8:59:00.361 PM   3   qakbot.dll     lstrcmpiA ( "chrome.exe", "mcshield.exe" )
8:59:00.361 PM   3   qakbot.dll     lstrcmpiA ( "chrome.exe", "avp.exe" )
8:59:00.361 PM   3   qakbot.dll     lstrcmpiA ( "chrome.exe", "kavtray.exe" )
8:59:00.361 PM   3   qakbot.dll     lstrcmpiA ( "chrome.exe", "egui.exe" )
8:59:00.361 PM   3   qakbot.dll     lstrcmpiA ( "chrome.exe", "ekrn.exe" )
8:59:00.361 PM   3   qakbot.dll     lstrcmpiA ( "chrome.exe", "bdagent.exe" )
8:59:00.361 PM   3   qakbot.dll     lstrcmpiA ( "chrome.exe", "vsserv.exe" )
8:59:00.361 PM   3   qakbot.dll     lstrcmpiA ( "chrome.exe", "vsservppl.exe" )
8:59:00.361 PM   3   qakbot.dll     lstrcmpiA ( "chrome.exe", "AvastSvc.exe" )
8:59:00.361 PM   3   qakbot.dll     lstrcmpiA ( "chrome.exe", "coreServiceShell.exe" )
8:59:00.361 PM   3   qakbot.dll     lstrcmpiA ( "chrome.exe", "PccNTMon.exe" )
8:59:00.361 PM   3   qakbot.dll     lstrcmpiA ( "chrome.exe", "NTRTScan.exe" )
```

Figure 1

-

The API trace shows that Qakbot scans the process names on the infected system

The API logs for the sample show an interesting pattern. The malware seems to enumerate all the processes running on the infected system and compare their name with the name of some security processes. Figure 15 reports an excerpt from the API log that I gathered by running the malware in a controlled environment. You can see that the Chrome process name is compared with the name of several security processes.

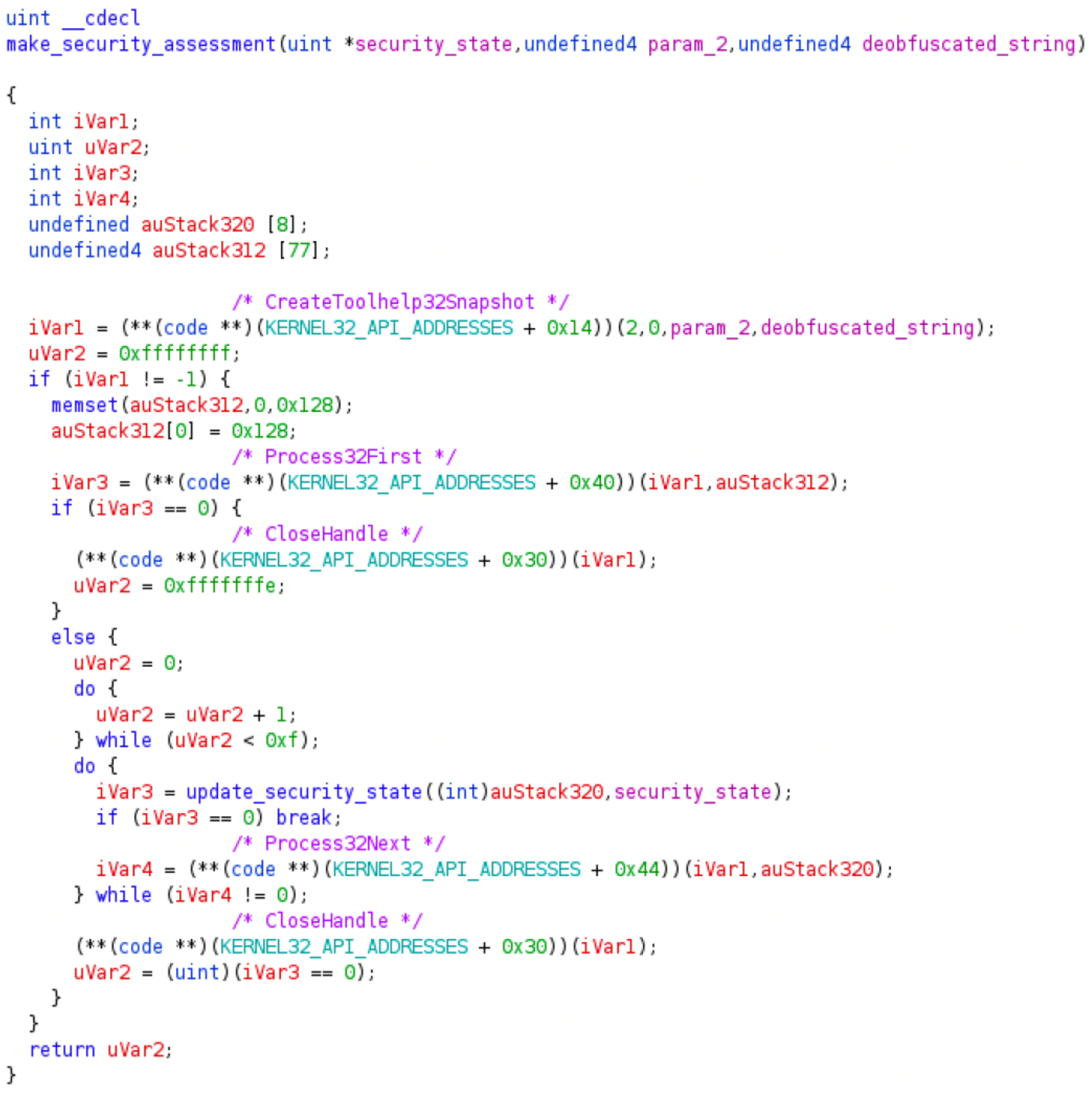
I investigated the malware to find a motivation for that observed behavior and I realized that it is doing a security assessment of the just infected system. The aim of such an assessment consists in understanding if there are security products running in the system and what are those products. As I will discuss in a while, the malware decides what process to inject based on the outcome of this security assessment.

Flag	Process(es)
0x1	ccSvcHst.exe
0x2	avgcsrvx.exe, avgsvcx.exe, avgcsrva.exe
0x4	MsMpEng.exe
0x8	mcshield.exe
0x10	avp.exe, kavtray.exe
0x20	egui.exe, ekrn.exe
0x40	bdagent.exe, vsserv.exe, vsservppl.exe
0x80	AvastSvc.exe
0x100	coreServiceShell.exe, PccNTMon.exe, NTRTScan.exe
0x200	SAVAdminService.exe, SavService.exe
0x400	fshoster32.exe
0x800	WRSA.exe
0x1000	vkise.exe, isesrv.exe, cmdagent.exe
0x2000	ByteFence.exe
0x4000	MBAMService.exe, mbamgui.exe
0x8000	fmon.exe
0x10000	dwengine.exe, dwarkdaemon.exe, dwwatcher.exe

Table 1

-
Mapping between security processes and boolean flags used during the security assessment of the infected system

The sample groups security processes mostly by vendor. As an example, I observed that the malware developers defined a group containing some processes related to the Dr.Web vendor: *dwengine.exe*, *dwarkdaemon.exe* and *dwwatcher.exe*. This group has been recently included since it isn't documented in a detailed analysis of a similar specimen published in 2021 (Trung Kien - 2021). A flag is assigned to each group. The flag for a given group is true if and only if at least one of the processes belonging to that group has been found on the infected system. The security state of a system is defined by the disgiunction of all the flags. What i call security assessment is implemented in a function located at *0xb2f3a9*. This function defines the mapping between groups and flags. As you may notice from **Table 1**, reporting the mapping between groups and flags, a group is represented as a string composed of comma-separated process names. However, those strings are obfuscated as discussed in the first post of the Qakbot series.



```
uint __cdecl
make_security_assessment(uint *security_state,undefined4 param_2,undefined4 deobfuscated_string)

{
  int iVar1;
  uint uVar2;
  int iVar3;
  int iVar4;
  undefined auStack320 [8];
  undefined4 auStack312 [77];

                    /* CreateToolhelp32Snapshot */
  iVar1 = (**(code **)(KERNEL32_API_ADDRESSES + 0x14))(2,0,param_2,deobfuscated_string);
  uVar2 = 0xffffffff;
  if (iVar1 != -1) {
    memset(auStack312,0,0x128);
    auStack312[0] = 0x128;
                    /* Process32First */
    iVar3 = (**(code **)(KERNEL32_API_ADDRESSES + 0x40))(iVar1,auStack312);
    if (iVar3 == 0) {
                    /* CloseHandle */
      (**(code **)(KERNEL32_API_ADDRESSES + 0x30))(iVar1);
      uVar2 = 0xfffffffe;
    }
    else {
      uVar2 = 0;
      do {
        uVar2 = uVar2 + 1;
      } while (uVar2 < 0xf);
      do {
        iVar3 = update_security_state((int)auStack320,security_state);
        if (iVar3 == 0) break;
                    /* Process32Next */
        iVar4 = (**(code **)(KERNEL32_API_ADDRESSES + 0x44))(iVar1,auStack320);
      } while (iVar4 != 0);
                    /* CloseHandle */
      (**(code **)(KERNEL32_API_ADDRESSES + 0x30))(iVar1);
      uVar2 = (uint)(iVar3 == 0);
    }
  }
  return uVar2;
}
```

Figure 2

-

Process enumeration function to update the security state of the system

The core of the security assessment algorithm is implemented in a function located at *0xb2dad3*. As you may notice from the listing of **Figure 2**, that function is responsible for the process scan observed in the API calls logs. The malware invokes CreateToolhelp32Snapshot, Process32First, Process32Next to iterate across the processes running on the system. Those API calls are protected by an API hashing technique I'll discuss about in a dedicated post. The function *update_security_state* is responsible for checking if the name of a process on the infected system is included in some group. If that is the case, then it activates the flag for that specific group. The security state is updated by or-ing itself with the flag of the active group.

Qakbot scans the processes on the infected system to understand if there are security products among them. This assessment is crucial for the malware because it influences the processes chosen as targets for the injection. The function implementing the target selection logic is located at *0xb2d84b* and it always return three targets according to some rules. As an example, consider the following list of processes: coreServiceShell.exe, PccNTMon.exe, NTRTScan.exe, SAVAdminService.exe, SavService.exe, bdagent.exe, vsserv.exe, vsservppl.exe, avp.exe, kavtray.exe, avgcsrvx.exe, avgsvcx.exe, and avgcsrva.exe. If any of those processes is running on the infected system, then the second decision driver is whether the malware is running on an x64 processor (or under the WOW64 Microsoft subsystem). If that is the case, then the target processes are:

- %SystemRoot%\SysWOW64\mobsync.exe
- %SystemRoot%\SysWOW64\explorer.exe
- %ProgramFiles(x86)%\Internet Explorer\iexplore.exe

Otherwise, if at least one of the processes has been found and the malware is running on an x86 processor, then the targets become:

- %SystemRoot%\System32\mobsync.exe
- %SystemRoot%\explorer.exe
- %ProgramFiles%\Internet Explorer\iexplore.exe

The fact that Qakbot targets mobsync and explorer is well known (Trung Kien - 2021). What I discovered with this sample is that now Qakbot may also target msra and OneDriveSetup. That happens, for example, if none of the processes listed before are running on the system and the malware is running on a x86 processor. Indeed, given those conditions, the targets become:

- %SystemRoot%\explorer.exe
- %SystemRoot%\System32\msra.exe
- %SystemRoot%\System32\OneDriveSetup.exe

Once obtained the targets, the sample attempts to inject code into them. The overall injection process is implemented in the function located at *0xb2d6c4*. That function iterates over the designated targets and for each of them tries to spawn a new process in suspended state by using the target path as the application name. If the creation succeeds then the function located at *0xb2d976* is invoked. That function is responsible for the actual injection. The injection prologue is implemented in function *0xb2d446* and it follows the following pattern: NtCreateSection, NtMapViewOfSection on the malware process, NtMapViewOfSection1 on the target process, and NtWriteProcessMemory to copy the Qakbot payload into the newly created memory area. Next, the sample tries to insert a trampoline to the payload at the entry point of the targeted process. To do so, it invokes GetThreadContext, NtProtectVirtualMemory, and NtWriteVirtualMemory. Finally, the malware awakens the process in suspended state.

As always, if you want to share comments or feedbacks (rigorously in broken Italian or broken English) do not esitate to drop me a message at **admin[@]malwarology.com**.