QuasarRAT's Dual DLL Sideloading Technique

By Tejaswini Sandapolla

Published: 2023-10-20 · Archived: 2026-04-05 23:16:57 UTC

Coauthored by Karthickkumar Kathiresan of Uptycs Threat Research Team

In a sophisticated twist to the traditional sideloading tactics, the Quasar RAT introduces a novel dual DLL sideloading technique, ingeniously utilizing two commonly trusted Microsoft files: "ctfmon.exe" and "calc.exe." Such a method not only leverages the inherent trust these files enjoy within the Windows ecosystem but also presents an increased challenge to threat detection mechanisms. This article dives deep into the meticulous design and execution of these sideloading techniques, illustrating how they stealthily introduce, deploy, and run malicious payloads under the radar.

QuasarRAT, also known as CinaRAT or Yggdrasil, is a lightweight remote administration tool written in C#. This tool is openly accessible as a GitHub project. This tool is capable of various functions such as gathering system data, running applications, transferring files, recording keystrokes, taking screenshots or camera captures, recovering system passwords, and overseeing operations like File Manager, Startup Manager, Remote Desktop, and executing shell commands.

Windows users, system administrators, and cybersecurity professionals need to be on high alert. The use of legitimate processes to cloak malicious activities helps them bypass traditional security measures. Hence, the need for advanced threat detection and response mechanisms becomes paramount.

QuasarRAT sideloading execution: a closer look at the technique

Given the prevalence of sideloading techniques in malware campaigns, it's vital to understand their mechanisms to defend against them effectively. The case of QuasarRAT provides an insightful example.

Historical context:

In 2022, we detected the Qbot malware employing a DLL sideloading attack using "calc.exe." Such tactics are not new but seeing them evolve and get adopted by other malware strains shows the adaptability of threat actors. Now, in 2023, a strikingly parallel method in two phases has been observed with the QuasarRAT malware.

Step-by-step breakdown:

1. Initial contact and execution:

The threat actor begins by employing DLL side-loading techniques. Interestingly, they opted for two distinct Microsoft files for their attack: "ctfmon.exe" and "calc.exe."

- In the initial phase, the attacker harnesses "ctfmon.exe," which is an authentic Microsoft file. By doing so, they load a malicious DLL which, to the untrained eye, would seem benign because of its disguised name.

- Upon execution of the "ctfmon.exe" binary, the stage is set as the attacker acquires a 'stage 1' payload. This initial payload is crucial, acting as the gateway for the subsequent malicious actions.

2. Payload release:

This 'stage 1' payload plays a dual role. It is responsible for releasing both the legitimate "calc.exe" file and the malevolent DLL into the system.

3. Second phase of attack:

- At this juncture, the threat actor brings into play the "calc.exe" file, which in this context, isn't just a simple calculator application. Alongside "calc.exe," the malicious DLL is also set into motion.
- On executing "calc.exe," the malicious DLL is triggered. This action culminates in the infiltration of the "QuasarRAT" payload into the computer's memory, reflecting the attacker's adeptness at circumventing security mechanisms.

4. Process hollowing:

With the "QuasarRAT" payload now residing in the computer's memory, the payload employs a technique known as 'process hollowing.' Here, it embeds itself into a legitimate system process, further camouflaging its malicious intentions and making detection more challenging.

Figure 1 depicts the QuasarRAT workflow.

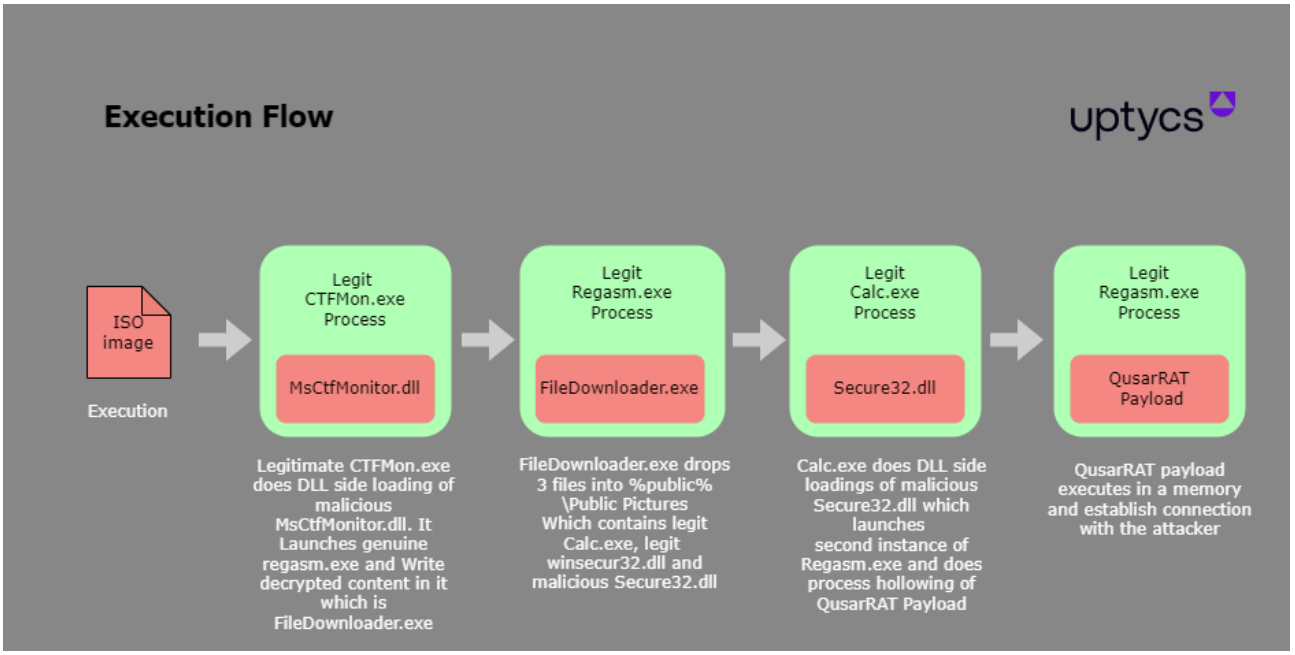

Figure 1 – QuasarRAT execution flow

Technical analysis

We first verified the ISO file and after a successful extraction, we obtained three separate files:

1. eBill-997358806.exe - Legitimate windows file, actual name is CTFMON.EXE

2. monitor.ini - Legitimate windows file, actual name is (MsCtfMonitor.DLL)
3. MsCtfMonitor.dll - Malicious Dll

Figure 2 depicts the Process flow of the new QuasarRAT.

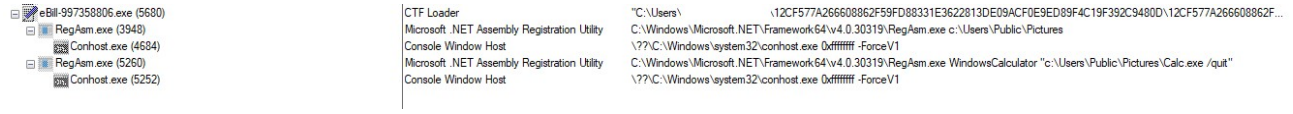

Figure 2 – QuasarRAT process tree

When the binary file "eBill-997358806.exe" is run, it initiates the loading of a file titled "MsCtfMonitor.dll" (name masqueraded) via dll side loading technique, within which malicious code is concealed.

Within the "MsCtfMonitor.dll" file, there exists a resource section containing encrypted data.
The resource section (RCDATA:400) has encrypted data of size 5AC00 hex bytes.

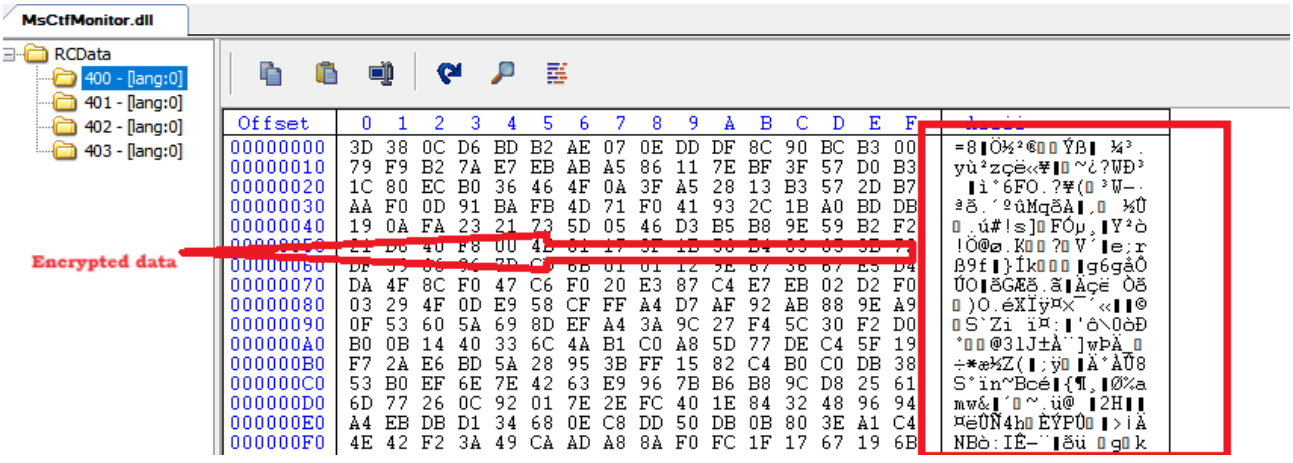

Figure 3 – Resource section of MsCtfMonitor.dll containing encrypted data

This data is accessed via a sequence of APIs as seen in Figure 4.

```
call qword ptr ds:[<&GetModuleHandleA>]
inc si
mov r8d,A
movzx edx,si
mov rcx,rax
mov rdi,rax
call qword ptr ds:[<&FindResourceW>]
mov rdx,rax
mov rcx,rdi
mov rbx,rax
call qword ptr ds:[<&SizeofResource>]
mov rdx,rbx
mov rcx,rdi
mov esi,eax
call qword ptr ds:[<&LoadResource>]
mov rcx,rax
call qword ptr ds:[<&LockResource>]
mov r8d,r14d
```

Figure 4 – Loading of resources

To decrypt the data, the key size is F2 hex bytes and **Systemfunction032** API is used to decrypt the encrypted data. This is an undocumented API that indirectly calls **BCryptGeneratesymmetrickey, CryptEncrypt, and**

CryptDestroyKey to decrypt data by RC4 where the key is also stored in .rsrc section (RCDATA: 401)

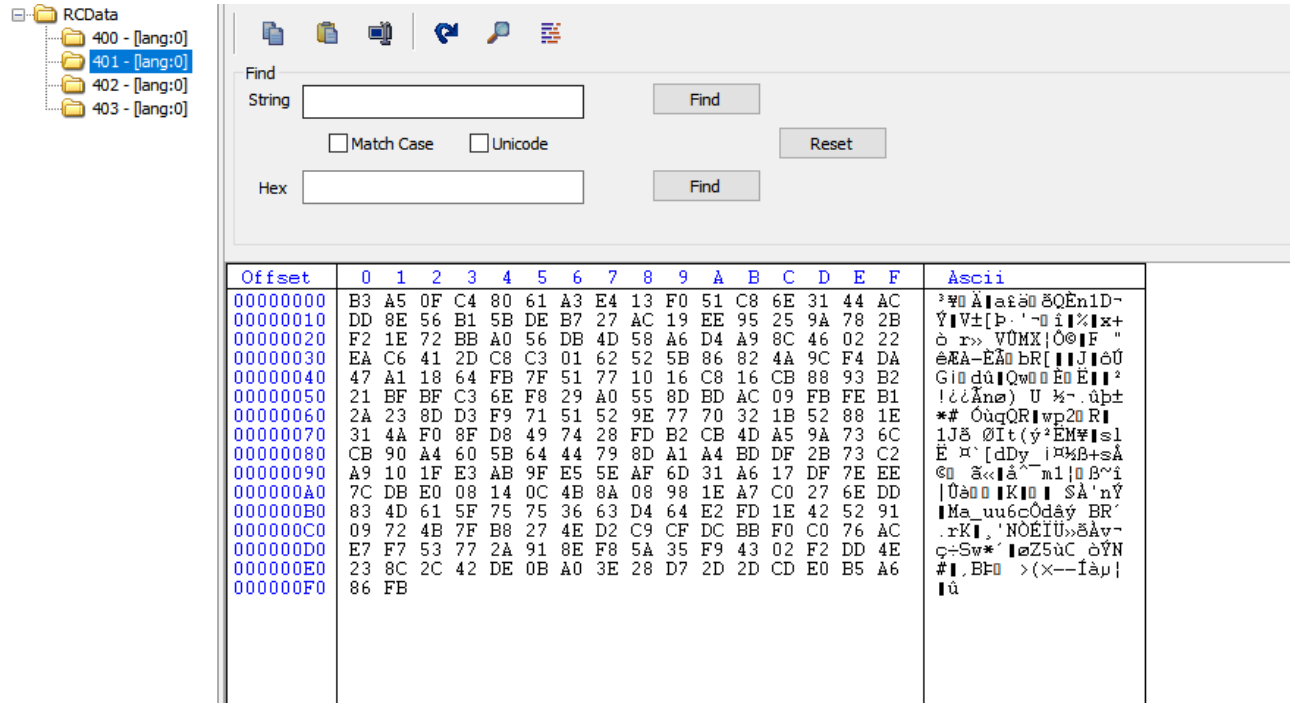

Figure 5– Resource section of MsCtfMonitor.dll containing decryption key

After decryption, the resource data is decrypted and give's PE file which is stage 1 "FileDownloader.exe."

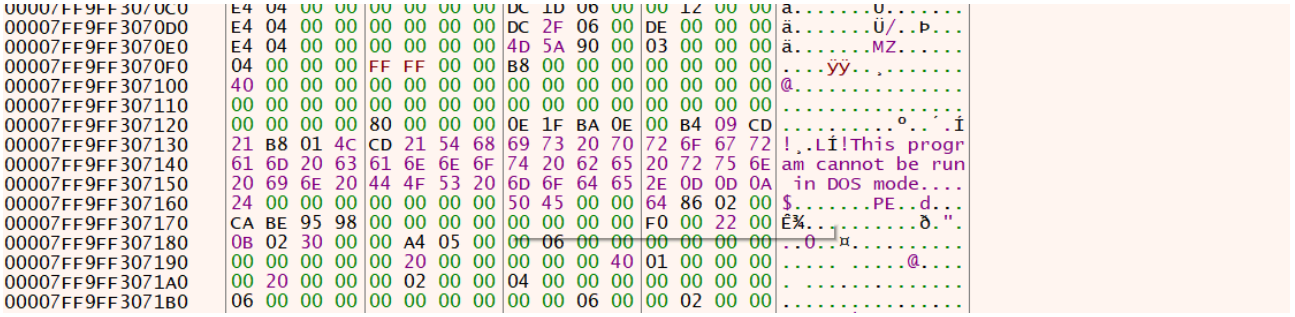

Figure 6– Decrypted data

This PE File is then injected into Regasm.exe by the following sequence of API : CreateProcess, GetThreadContext, ReadProcessMemory, VirtualAllocEx and WriteProcessMemory, GetThreadContext, SetThreadContext and ResumeThread.

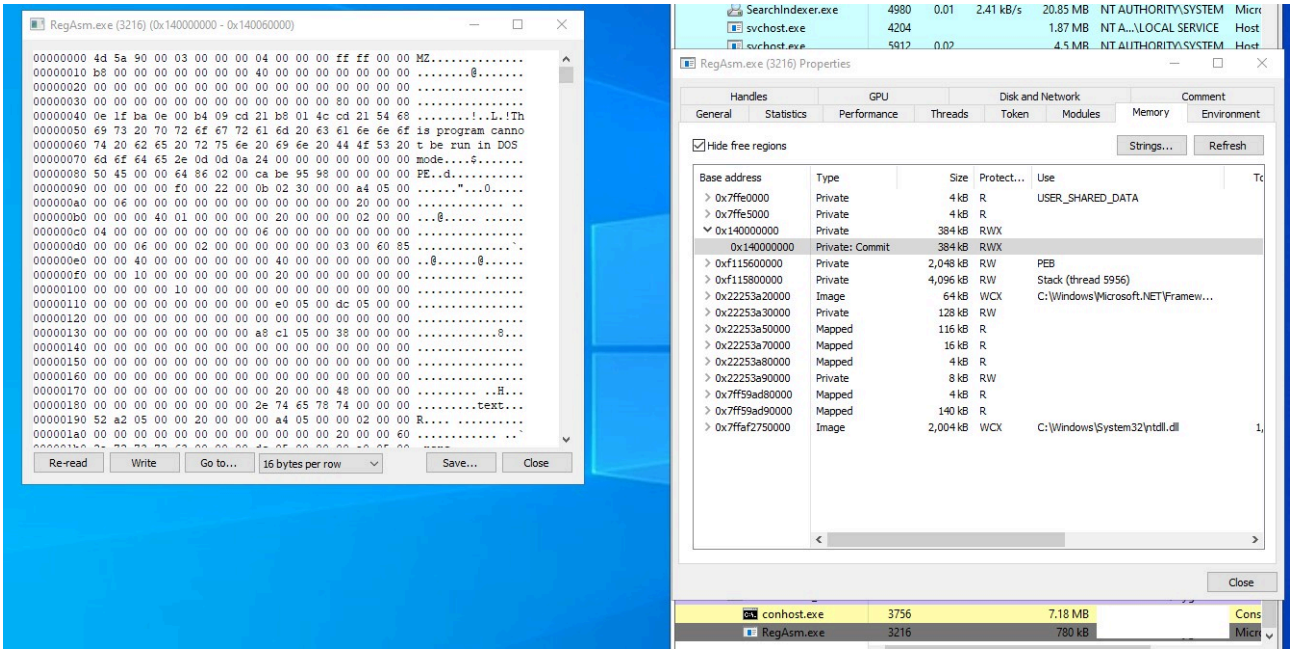

Figure 7 – Stage 1 payload in the memory of RegAsm

In Figure 7 we can observe the presence of the stage 1 payload within the memory of the RegAsm process.

Stage 1: FileDownloader.exe

The stage 1 payload is a 64-bit MSIL binary file, which includes a resource section containing three binaries stored in a zip archive format.

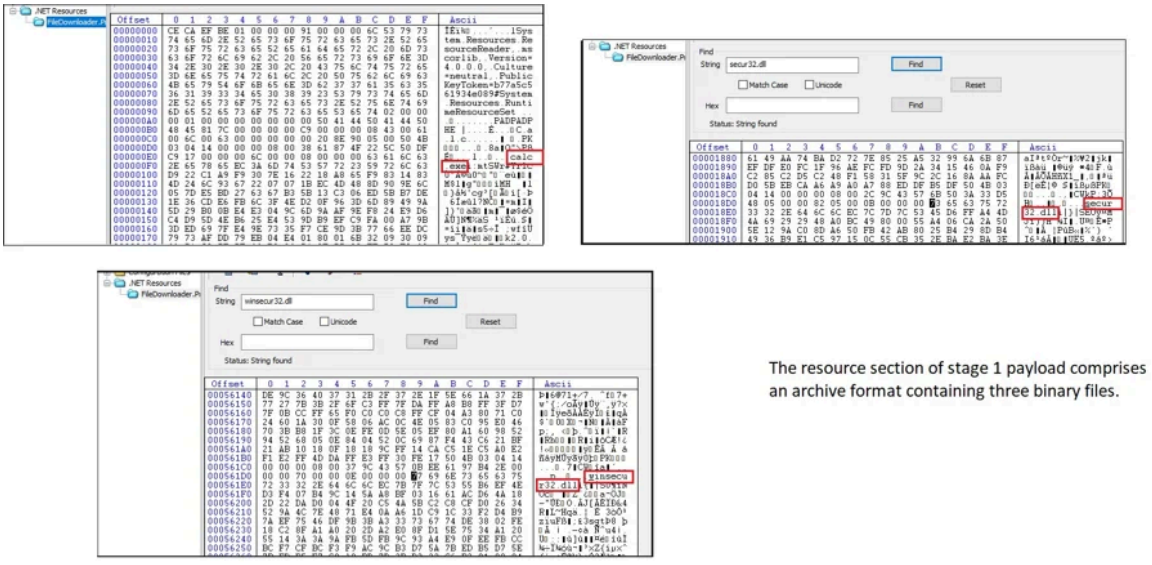

Figure 8 – Resource section of stage 1

The stage 1 payload is equipped with code to unzip this archive successfully, depositing all the files into the Public Pictures folder.

The following files are placed in the Public Pictures folder:

1. Calc.exe - Legitimate windows file
2. Secure32.dll - Malicious DLL
3. Winsecu32.dll - Legitimate windows file

Stage 2: Calc.exe

Next, "Calc.exe" was run using the command "c:\Users\Public\Pictures\Calc.exe /quit."

"/quit" serves as an argument or parameter for the Calculator executable, instructing it to open and promptly close upon launch.

When you run calc.exe, it loads a malicious DLL named "Secure32.dll." again via Dll-side loading technique. This DLL contains an encrypted resource section.

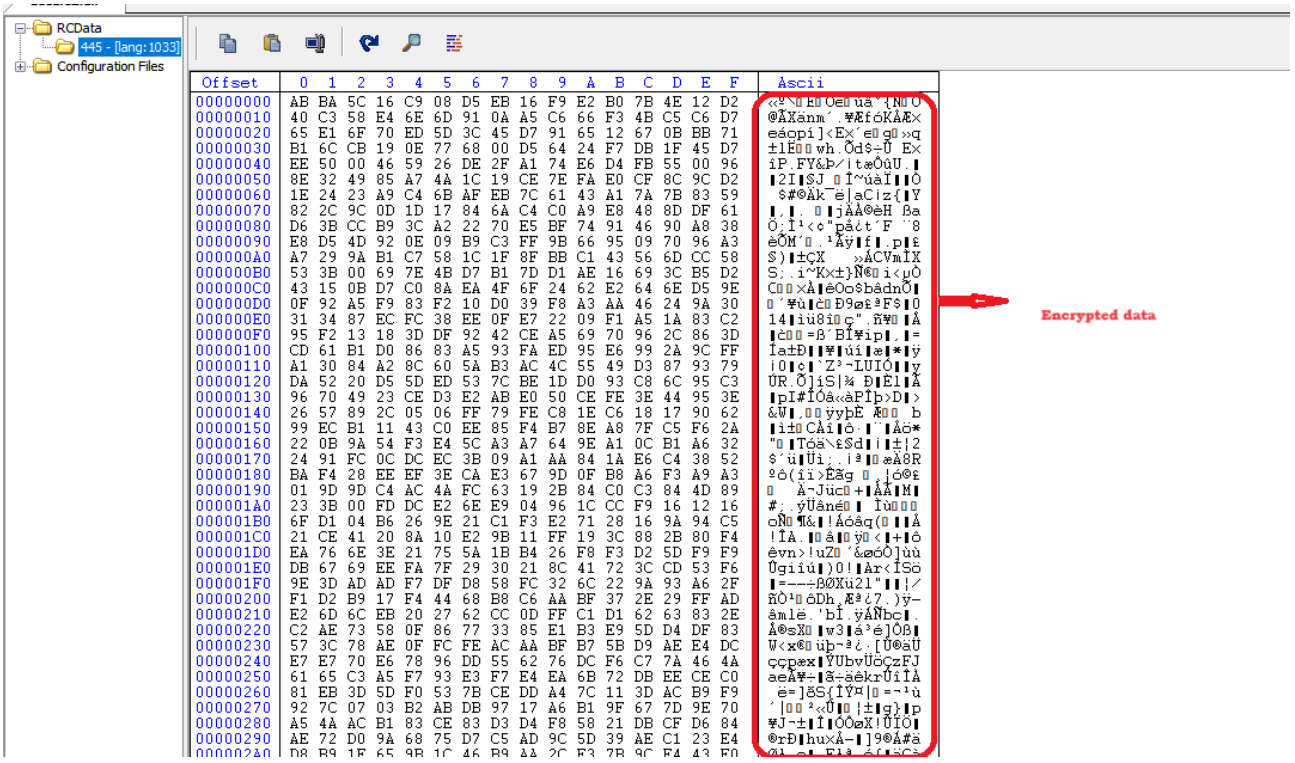

Figure 9 - Resource section of Secure32.dll containing encrypted data

This resource data is accessed via a sequence of APIs like previous method mentioned as in the stage 1 (Figure 4)

To decrypt the data, Key size is 2F hex bytes and Systemfunction32 API is used to decrypt the encrypted data same as in the first case which again gives a PE file.

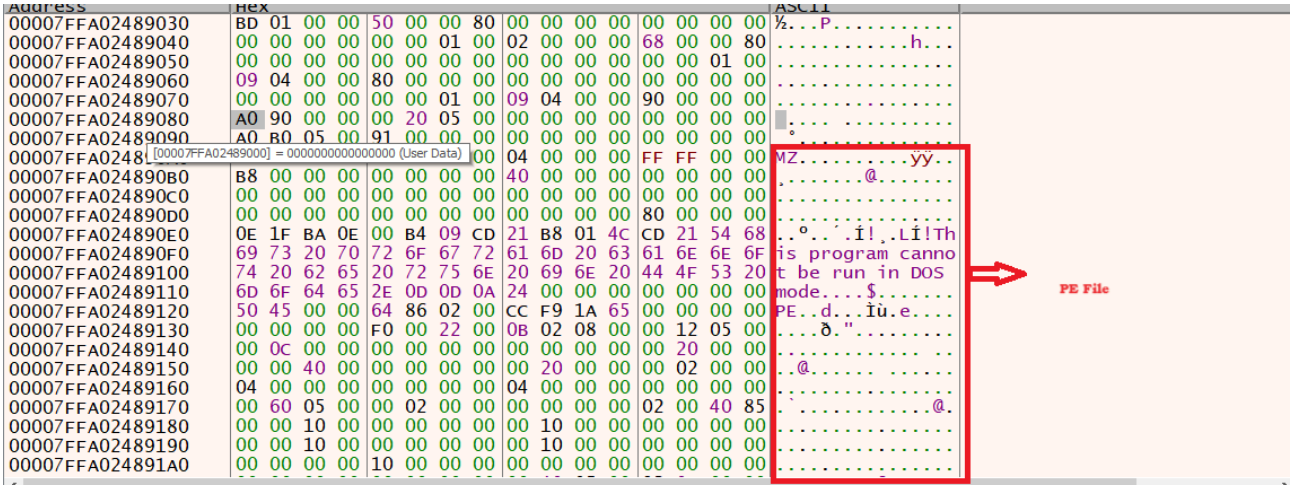

Figure 10 - Decrypted data

This PE File is now injected into memory space Regasm.exe via process hollowing. The API sequence followed is the same as in stage 1. But here, while calling VirtualAllocEx, the default memory address of regasm.exe (0x400000) is explicitly passed to hollow the regasm.exe and replace it with the malicious PE file.

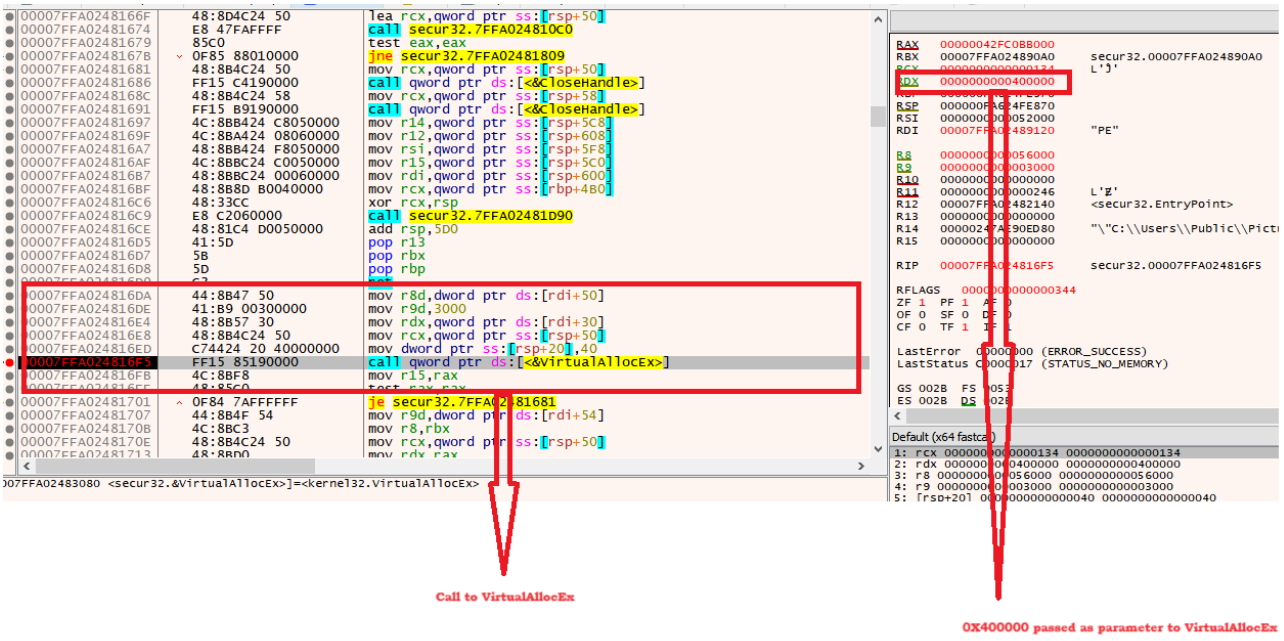

Figure 11 - Call to VirtualAllocEx where RDX = 0X400000 is passed

Final payload

By getting the dump of the above region(Figure 11), we can analyze that this PE file is an MSIL executable obfuscated by Smart assembly.

Looking at the copyright of the file as "Copyright © MaxXor 2020", it looks like it might be inspired by open-source Quasar RAT by MaxXor.

After deobfuscating, we can see commands executed in the function names which include keylogging, file transfer, shell execute, etc.

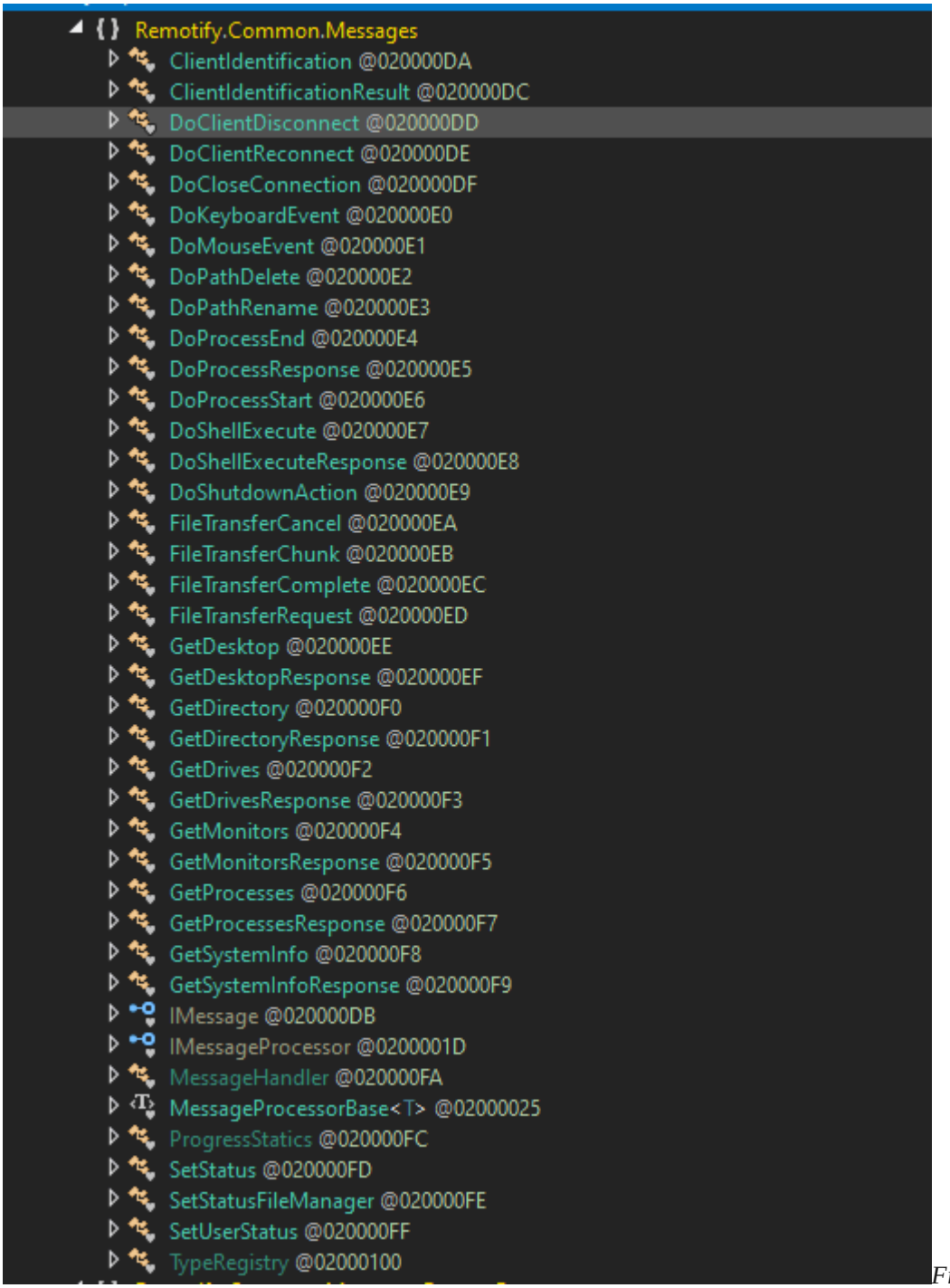

Figure 12 - Quasar RAT commands like keylogging, file transfer, etc.

It also drops a .bat script to create the restart batch file in the %Temp% directory which is executed and runs chcp 65001 && ping -n 10 localhost.

File Edit Format View Help

```
@echo off
chcp 65001
echo DONT CLOSE THIS WINDOW!
ping -n 10 localhost > nul
start "" "C:\Windows\Microsoft.NET\Framework64\v4.0.30319\RegAsm.exe"
del /a /q /f "C:\Users\gambit\AppData\Local\Temp\llbNwERFGo5p.bat"
```

Figure 13 - .bat file

The RAT creates a socket connection to CNC (3[.]94[.]91[.]208 >> ec2-3-94-91-208[.]compute-1[.]amazonaws.com) where it sends the victim's info such as IP, Country code etc.

```
{
    // Token: 0x060000BA RID: 186 RVA: 0x0000A078 File Offset: 0x00008278
    internal Class25 method_0()
    {
        Class25 result;
        try
        {
            HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(Class27.getString_0(107397134));
            httpWebRequest.UserAgent = Class27.getString_0(107397057);
            httpWebRequest.Proxy = null;
            httpWebRequest.Timeout = 10000;
            using (HttpWebResponse httpWebResponse = (HttpWebResponse)httpWebRequest.GetResponse())
            {
                using (Stream responseStream = httpWebResponse.GetResponseStream())
                {
                    GeoResponse geoResponse = Class32.smethod_0<GeoResponse>(responseStream);
                    Class25 @class = new Class25();
                    @class.method_1(geoResponse.Data.Geo.Ip);
                    @class.method_3(geoResponse.Data.Geo.CountryName);
                    @class.method_5(geoResponse.Data.Geo.CountryCode);
                    @class.method_7(geoResponse.Data.Geo.Timezone);
                    @class.method_9(geoResponse.Data.Geo.Asn.ToString());
                    @class.method_11(geoResponse.Data.Geo.Isp);
                    result = @class;
                }
            }
        }
        catch
        {
            result = null;
        }
        return result;
    }
```

Figure 14 - Collects victim's PC information

After deobfuscating more content, we can see strings related to Quasar RAT such as Quasar Server etc.

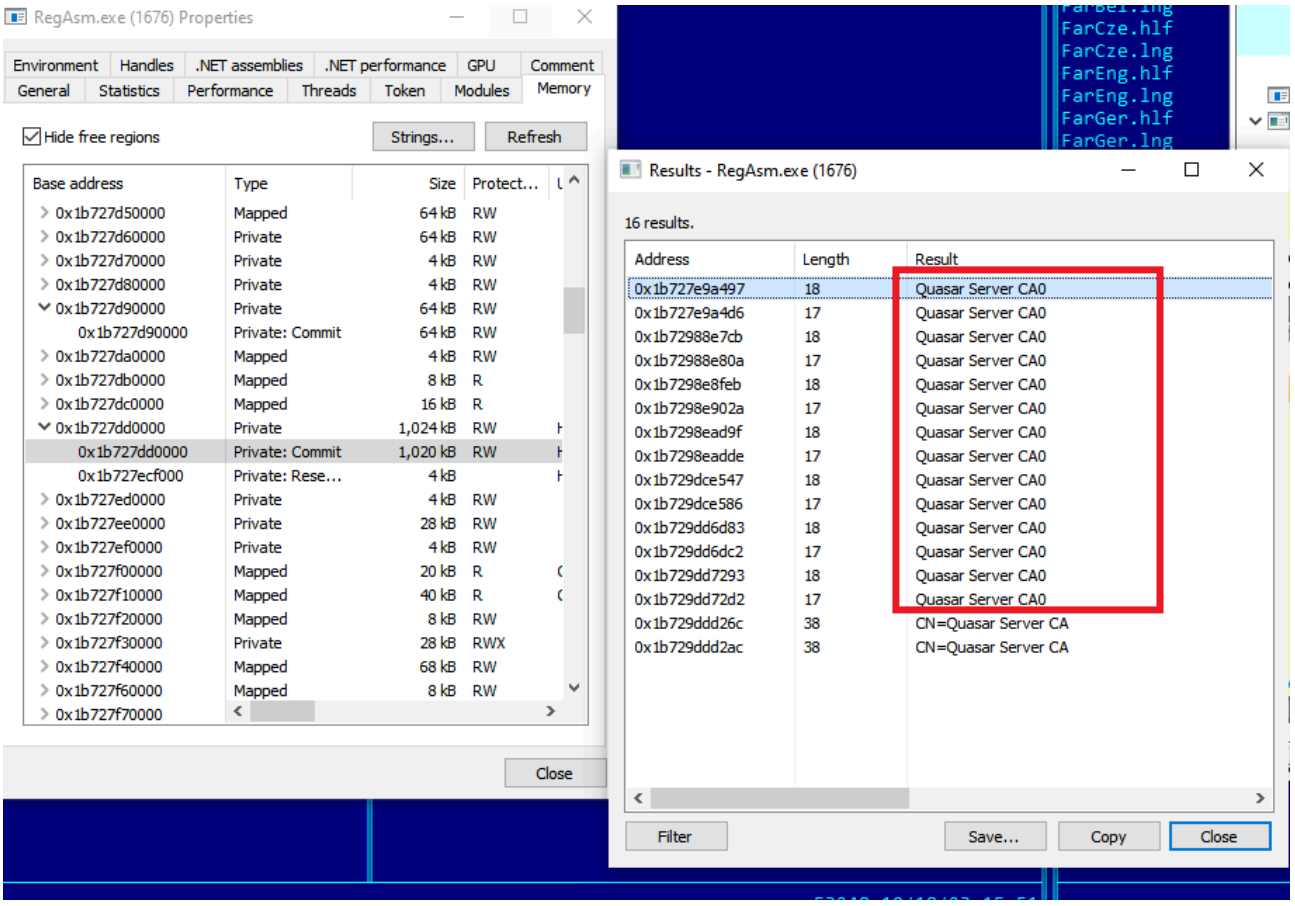

Figure 15 - Quasar RAT related strings inside Regasm.exe memory

We can see a lot of base64 content also in the memory which on decoding gives various strings such as:

SELECT * FROM Win32_OperatingSystem WHERE Primary='true'
SELECT * FROM Win32_BaseBoard
SELECT * FROM FirewallProduct
SELECT * FROM Win32_Processor
SELECT * FROM AntivirusProduct

By looking at the above strings we can understand that then the RAT is querying for the AntiVirusProduct and Firewall WMI class. The Quasar RAT payload also looks for BIOS infrastructure, GPU details, hostname , etc.

Quasar RAT is an open-source remote access trojan (RAT) that has been widely used by threat actors due to its powerful techniques. Quasar RAT capabilities include Keylogging, stealing passwords, taking screenshots, reverse proxy, Downloading and uploading files etc. We can see in the below figure Reverse proxy functionalities inside our final payload.

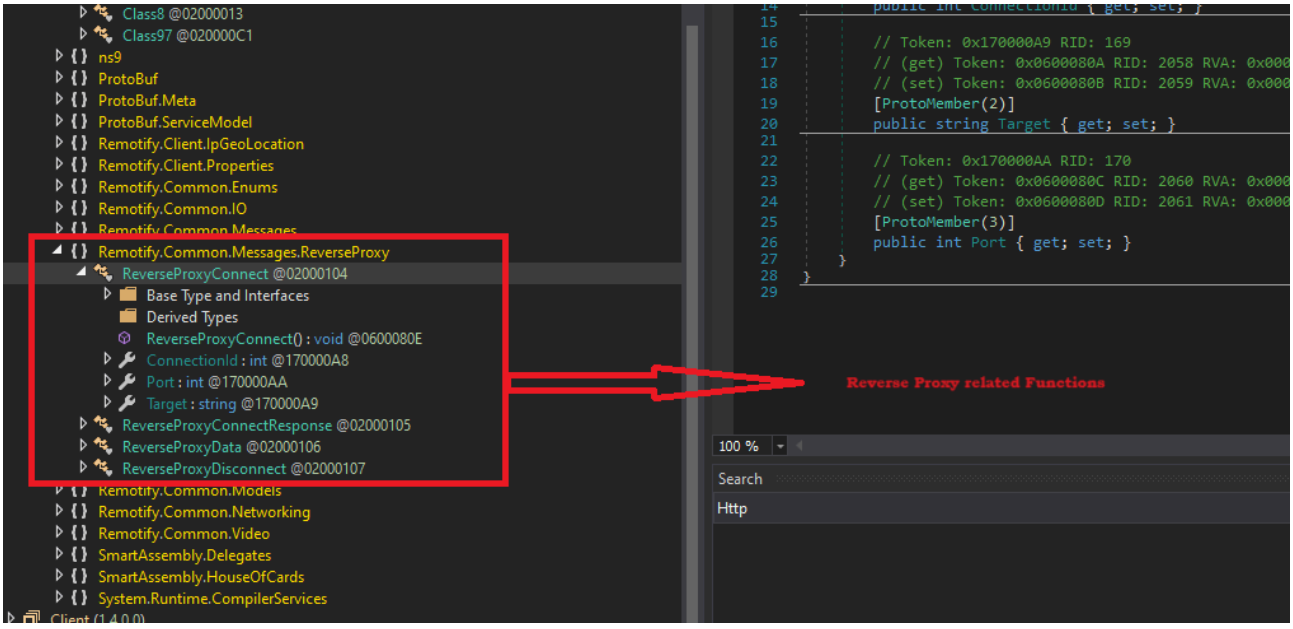

Figure 16 - Reverse proxy functionalities

The malware also establishes a persistent entry within the Windows registry.

HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\WindowsCalculator
“c:\Users\Public\Pictures\Calc.exe /quit”

Uptycs XDR coverage

In addition to having YARA built in and being armed with other advanced detection capabilities, Uptycs XDR users can easily scan for QuasarRAT. XDR contextual detection provides important details about identified malware. Users can navigate to the toolkit data section in the detection screen, and then click a detected item to reveal its profile.

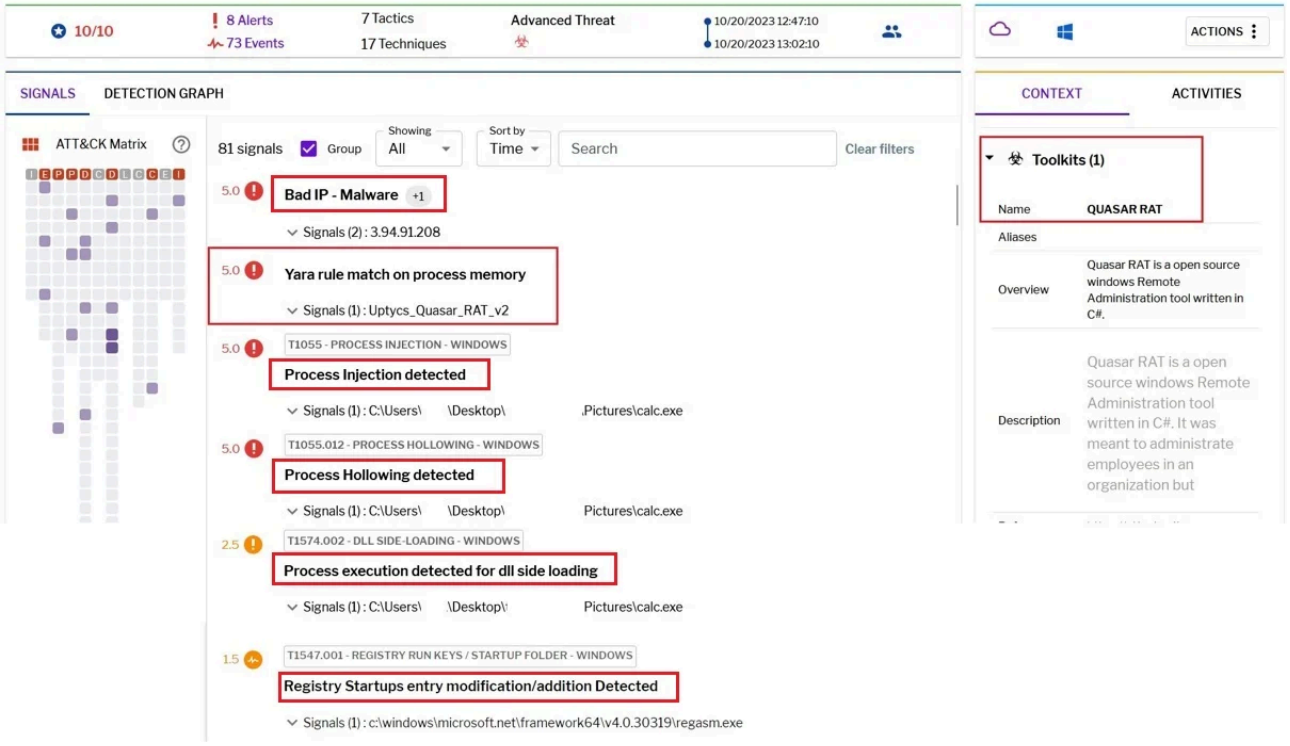

Figure 17 - Uptycs Detection

Precautions

- Keep software and operating systems up to date
- Be wary of dubious emails, links, or attachments. Avoid revealing personal details or engaging with unfamiliar links.
- Implement behavioral analysis tools to identify unusual activities and potential threats.
- Train employees and individuals to recognize suspicious activities and avoid running unfamiliar files or executing unknown commands.
- Develop and enforce strong security policies within your organization.
- Utilize advanced endpoint security solutions to detect and block suspicious activity at the device level.
- Collaborate with cybersecurity experts and share threat information within your industry or community to stay informed about evolving threats.

IOC

File Name	Md5
ISO	e4eb623a0f675960acb002d225c6f1d6

eBill-997358806.exe	B625C18E177D5BEB5A6F6432CCF46FB3
monitor.ini	7074832F0EFB8A2130B1935EAE5A90D6
MsCtfMonitor.dll	B0DB6ADA5B81E42AADB82032CBC5FD60
Stage 1/ FileDownloader.exe	32DE5C2E0BA35CEAC3C515FA767E42BF
Calc.exe	5da8c98136d98dfec4716edd79c7145f
Secure32.dll	d07e4afd8f26f3e2ce4560e08b7278fb
Winsecu32.dll	f11c63cb70a726f1f0b6accd5934e83
Final Payload/Remotify Client	532AF2DB4C10352B2199724D528F535F

URL

3[.]94[.]91[.]208	ec2-3-94-91-208[.]compute-1[.]amazonaws.com

Thanks to the Threat Hunting Team of Uptycs for sharing the IOC.

Source: https://www.uptycs.com/blog/quasar-rat