Malicious DLL Technical Analysis Report

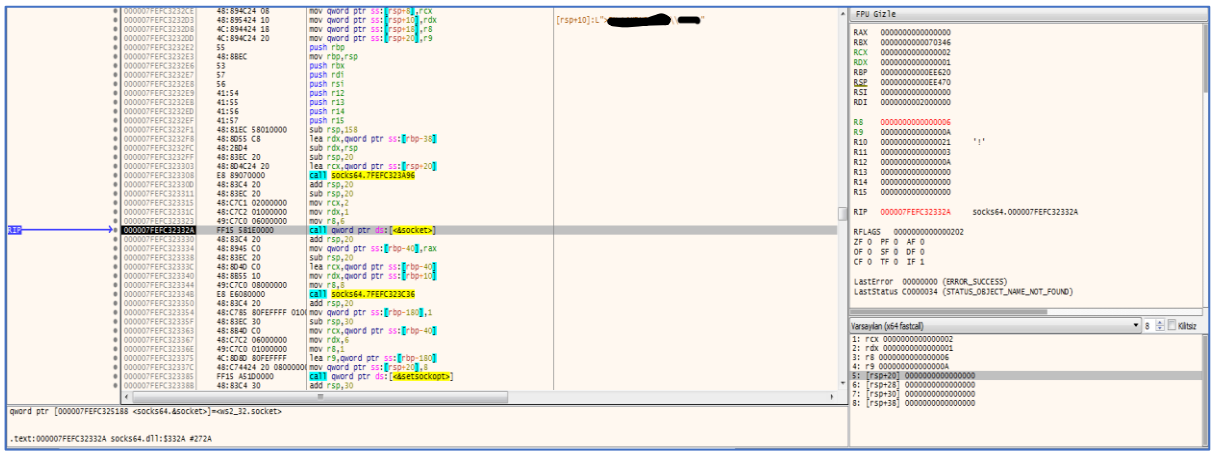
Table of Contents

File Name	socks64.dll
MD5	7269EABD886ECAEA06F8A0F462DBAF85
SHA1	E1A4DA4B1C193594897040C80D878BEA872FA506
SHA256	4D3F4C91EB570415E25ED33F59C91B352167D25ED1B94DC315456DCA80E728D9

Dynamic Analysis

Socket Operations

When the malicious DLL was examined, it was seen that it sent data to a command and control server by performing socket operations and also received data. It has been observed that it creates sockets for the imported data and the data to be sent.



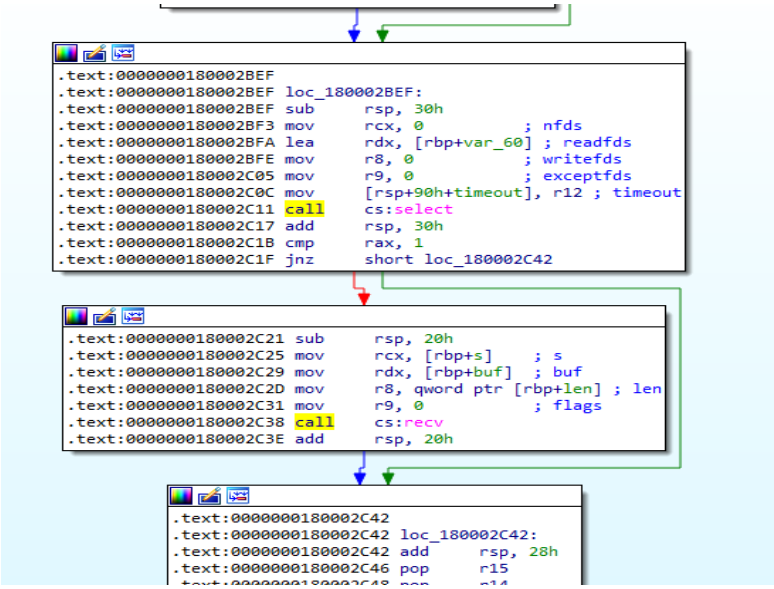
After creating a socket, it tries to **KEEP_ALIVE** connection with the **connect** API to predefined IPs. However, because the servers are not currently standing, the malware is never able to provide this connection.

```
.text:000000018000343D call     cs:htons
.text:0000000180003443 add      rsp, 20h
.text:0000000180003447 mov      word ptr [rbp+name.sa_data], ax
.text:000000018000344B mov      [rbp+name.sa_family], 2
.text:0000000180003451 mov      qword ptr [rbp+optval], 1
.text:000000018000345C sub      rsp, 20h
.text:0000000180003460 mov      rcx, [rbp+s]    ; s
.text:0000000180003464 mov      rdx, 8004667Eh  ; cmd
.text:000000018000346E lea      r8, [rbp+optval] ; argp
.text:0000000180003475 call     cs:ioctlsocket
.text:000000018000347B add      rsp, 20h
.text:000000018000347F sub      rsp, 20h
.text:0000000180003483 mov      rcx, [rbp+s]    ; s
.text:0000000180003487 lea      rdx, [rbp+name] ; name
.text:000000018000348B mov      r8, 10h         ; namelen
.text:0000000180003492 call     cs:connect
.text:0000000180003498 add      rsp, 20h
.text:000000018000349C sub      rsp, 30h
.text:00000001800034A0 mov      rcx, [rbp+s]
.text:00000001800034A4 mov      rdx, 0
.text:00000001800034AB lea      r8, [rbp+var_68]
.text:00000001800034AF mov      r9, [rbp+arg_18]
.text:00000001800034B3 mov      qword ptr [rsp+1C0h+optlen], 0
.text:00000001800034BC call     sub_180002FF6
.text:00000001800034C1 add      rsp, 30h
.text:00000001800034C5 sub      rsp, 30h
.text:00000001800034C9 mov      rcx, 0          ; nfds
.text:00000001800034D0 mov      rdx, 0          ; readfds
.text:00000001800034D7 lea      r8, [rbp+var_68] ; writefds
.text:00000001800034DB mov      r9, 0           ; exceptfds
.text:00000001800034E2 lea      rax, [rbp+timeout]
.text:00000001800034E6 mov      qword ptr [rsp+1C0h+optlen], rax ; timeout
.text:00000001800034EB call     cs:select
.text:00000001800034F1 add      rsp, 30h
.text:00000001800034F5 cmp      rax, 1
.text:00000001800034F9 jnz      short loc_180003534
```

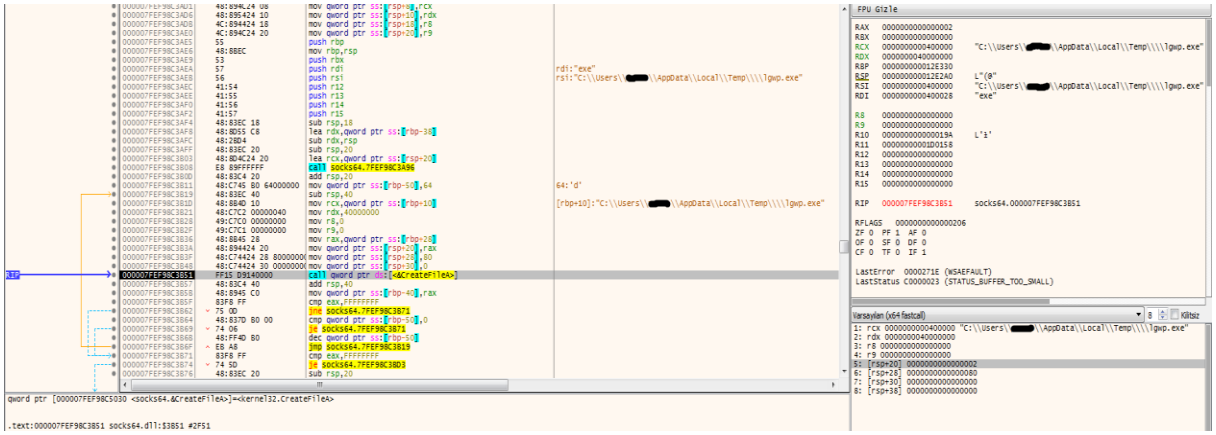
IPs of the malware is trying to connect:

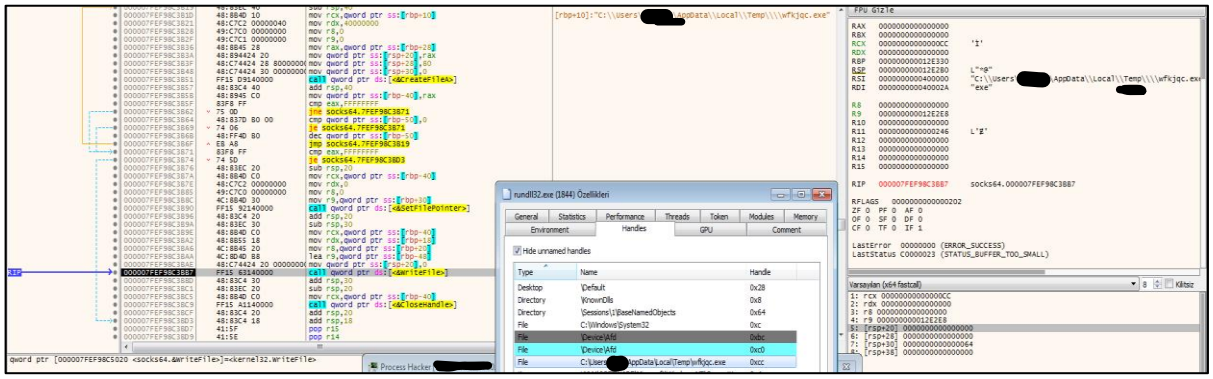
Time of Day	Process Name	PID	Operation	Path	Result	Detail
00:24:13,0232016	rundll32.exe	3712	TCP Reconnect	[redacted]localdomain:49196 -> .214.132.62:https	SUCCESS	Length: 0, seqnum: 0, connid: 0
00:24:19,0340019	rundll32.exe	3712	TCP Reconnect	[redacted]localdomain:49196 -> .214.132.62:https	SUCCESS	Length: 0, seqnum: 0, connid: 0
00:24:39,4246275	rundll32.exe	3712	TCP Reconnect	[redacted]localdomain:49197 -> li979- .members. .com:https	SUCCESS	Length: 0, seqnum: 0, connid: 0
00:24:45,4298466	rundll32.exe	3712	TCP Reconnect	[redacted]localdomain:49197 -> li979- .members. .com:https	SUCCESS	Length: 0, seqnum: 0, connid: 0

Assuming the servers are active, it has been observed that it receives data from this server with **recv** API**.** By assuming because the server not be able to connect, we observe that it receives a payload from this server and then writes it into a file.



However, payload cannot be reached because the servers are closed.

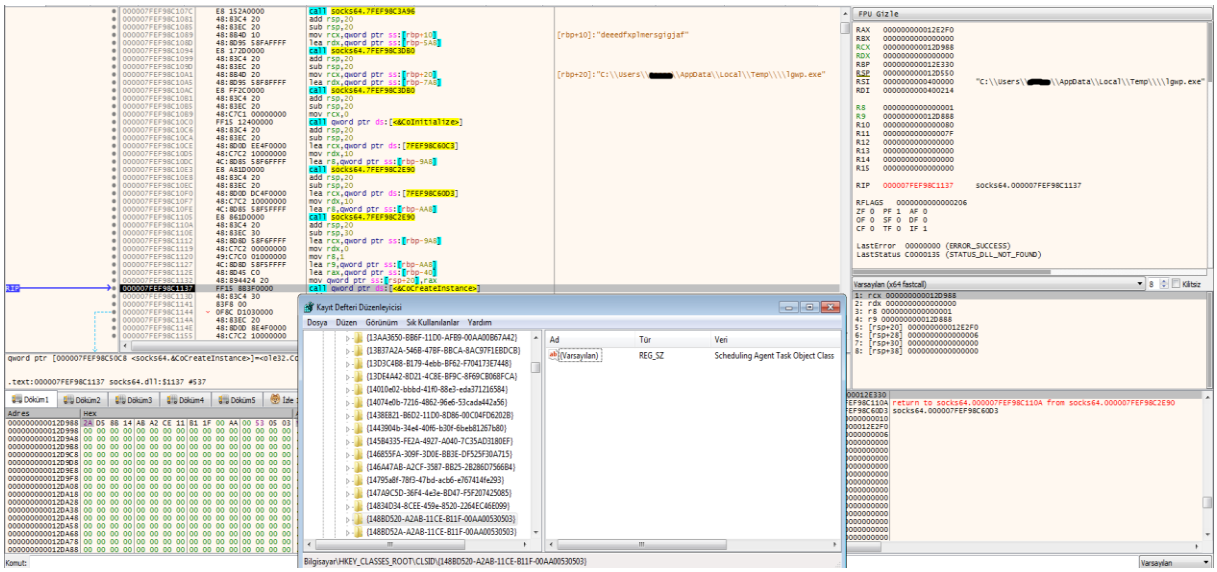




COM Transactions

As is known, COM objects are used by malware; the backdoor is used for purposes such as remote control. The malicious DLL we have uses the Task Schedule feature of Windows to run the payload it receives using COM objects at certain times, using the Windows concept without the need for the user to run this file, and running the malware on the user's device.

For this method to be used, **CoInitialize** and then **CoCreateInstance** should be used first.

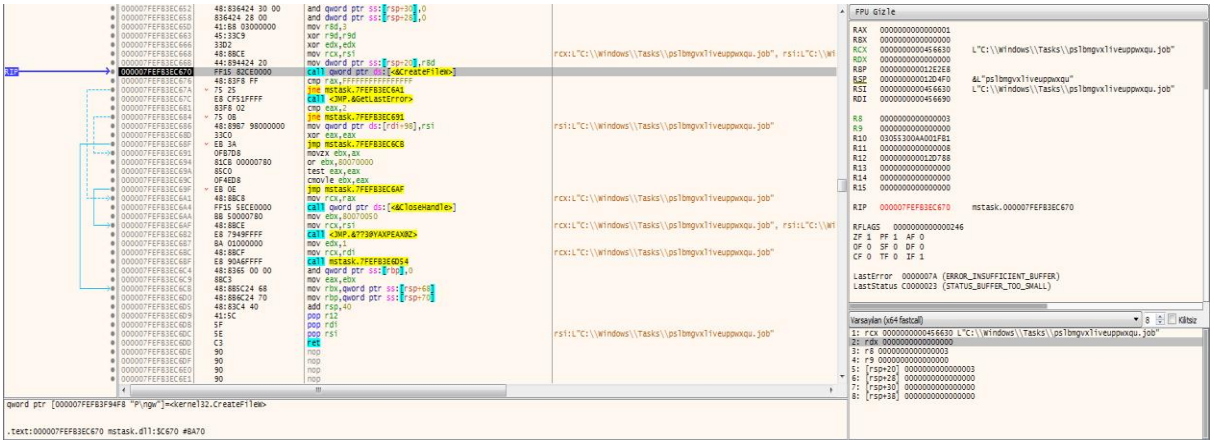


In the 1st parameter of the **CoCreateInstance** API, we can see when we look at the parameter "**rclsid**" and search for "14 8B D5 2A - A2 AB - 11 CE - B1 1F- 00 AA 00 53 05 03" in the sub-index "**HKEY_CLASSES_ROOT-CLSID**" at this address, the value of this key is "**Scheduling Agent Task Object Object**". From here, we observe that our malware creates a COM object from the Windows Task Scheduler class.

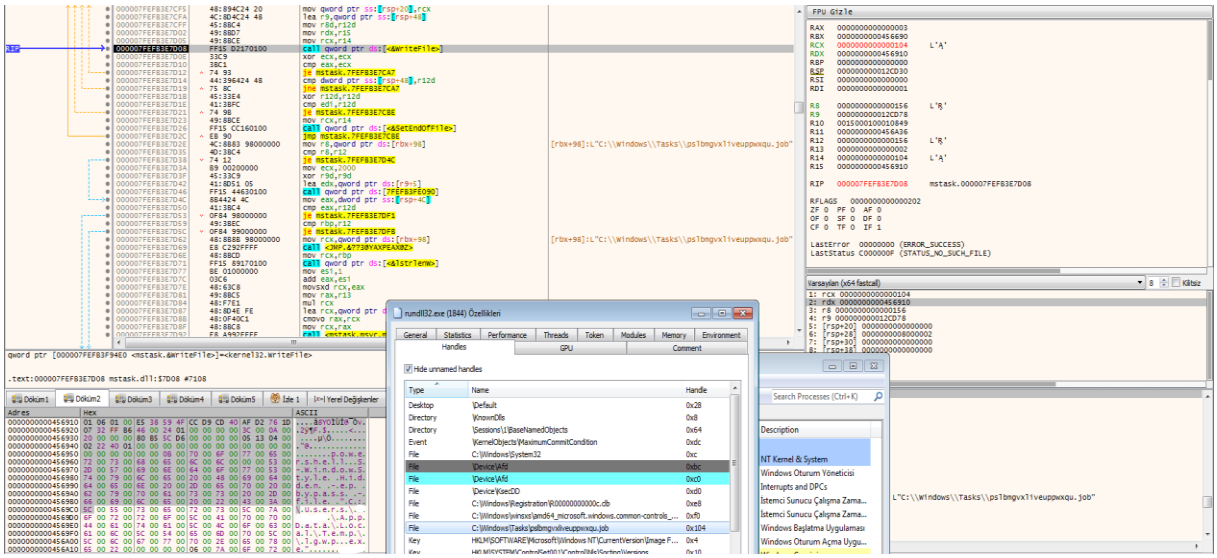
Acces Token Manipulation

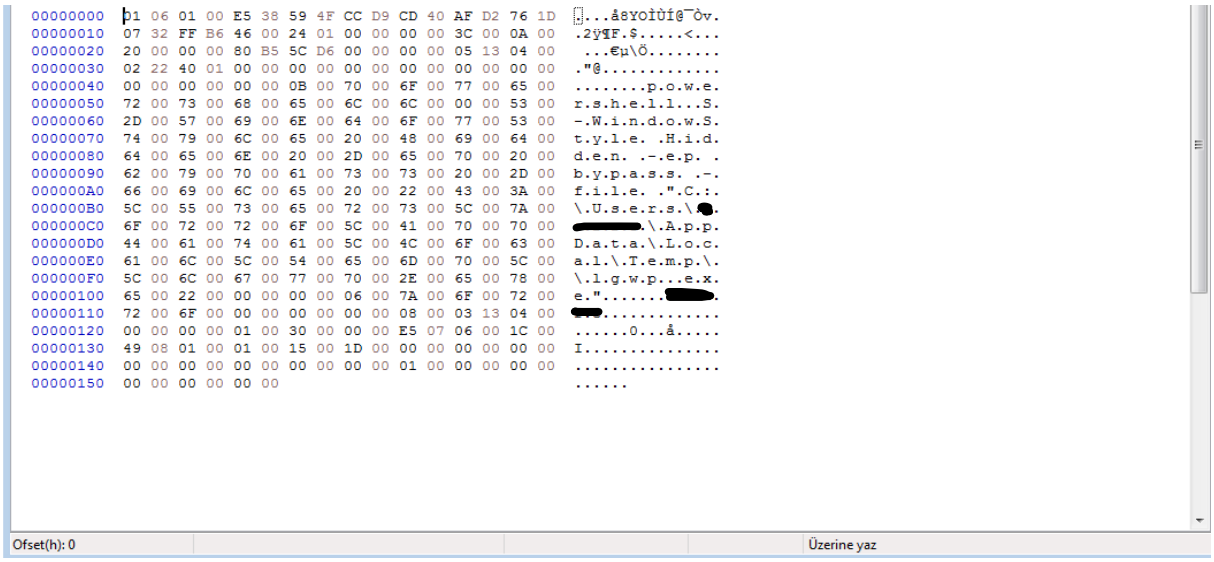
With this technique, which is used to elevate authority, our malware tries to increase the authority because it wants to create a file under the "Windows/Task" folder.

It has been observed that it creates files in the specified directory after the elevation of authority.



After the malware creates the file with the extension ".job", it writes a powershell command for run EXE which includes payload received from servers with **WriteFile** API.




00000000 b1 06 01 00 E5 38 59 4F CC D9 CD 40 AF D2 76 1D
00000010 07 32 FF B6 46 00 24 01 00 00 00 00 3C 00 0A 00
00000020 20 00 00 00 80 B5 5C D6 00 00 00 00 05 13 04 00
00000030 02 22 40 01 00 00 00 00 00 00 00 00 00 00 00 00
00000040 00 00 00 00 00 00 0B 00 70 00 6F 00 77 00 65 00
00000050 72 00 73 00 68 00 65 00 6C 00 6C 00 00 00 53 00
00000060 2D 00 57 00 69 00 6E 00 64 00 6F 00 77 00 53 00
00000070 74 00 79 00 6C 00 65 00 20 00 48 00 69 00 64 00
00000080 64 00 65 00 6E 00 20 00 2D 00 65 00 70 00 20 00
00000090 62 00 79 00 70 00 61 00 73 00 73 00 20 00 2D 00
000000A0 66 00 69 00 6C 00 65 00 20 00 22 00 43 00 3A 00
000000B0 5C 00 55 00 73 00 65 00 72 00 73 00 5C 00 7A 00
000000C0 6F 00 72 00 72 00 6F 00 5C 00 41 00 70 00 70 00
000000D0 44 00 61 00 74 00 61 00 5C 00 4C 00 6F 00 63 00
000000E0 61 00 6C 00 5C 00 54 00 65 00 6D 00 70 00 5C 00
000000F0 5C 00 6C 00 67 00 77 00 70 00 2E 00 65 00 78 00
00000100 65 00 22 00 00 00 00 00 06 00 7A 00 6F 00 72 00
00000110 72 00 6F 00 00 00 00 00 00 00 00 00 08 00 03 13 04 00
00000120 00 00 00 00 01 00 30 00 00 00 E5 07 06 00 1C 00
00000130 49 08 01 00 01 00 15 00 1D 00 00 00 00 00 00 00
00000140 00 00 00 00 00 00 00 00 00 00 01 00 00 00 00 00
00000150 00 00 00 00 00 00
Ofset(h): 0
Üzerine yaz

Solution Suggestions

- Use of up-to-date, reliable anti-virus software in systems,

- Careful attention to incoming e-mails, not to open attachments unconsciously without analysis,

- Disregard of spam emails may prevent such malware from infecting the system.

YARA RULE

```
import "hash"

rule FirstFile{
        meta:
                description="socks64.dll"
        strings:
                $str1="CoInitialize"
                $str2="CoCreateInstance"
                $str3="connect"
                $str4="ioctlsocket"
                $str5="recv"
                $str6="send"
                $str7="WSAIoctl"
                $str8=".job"

                condition:
                        hash.md5(0,filesize)=="7269EABD886ECAEA06F8A0F462DBAF85" or all of them
}
```

Fatih YILMAZ

https://www.linkedin.com/in/fatih-yilmaz-f8/