

RomCom RAT: Not Your Typical Love Story

Published: 2023-09-08 · Archived: 2026-04-05 15:53:58 UTC

Remote Access Trojan (RAT) is a type of malware that, as the name suggests, can remotely access a victims' system after successful infection. This blog is about one such RAT, RomCom RAT which can take complete control of a compromised system by spoofing and deploying fake versions of legitimate applications on the victims' system to gain initial trust.

Let us get into the details of one of the samples which drops a malicious RomCom RAT binary. This sample was digitally signed by Noray Consulting Ltd. On further analysis, we observed that Noray Consulting Ltd had a dummy LinkedIn page and a dubious website to deceive the victims.

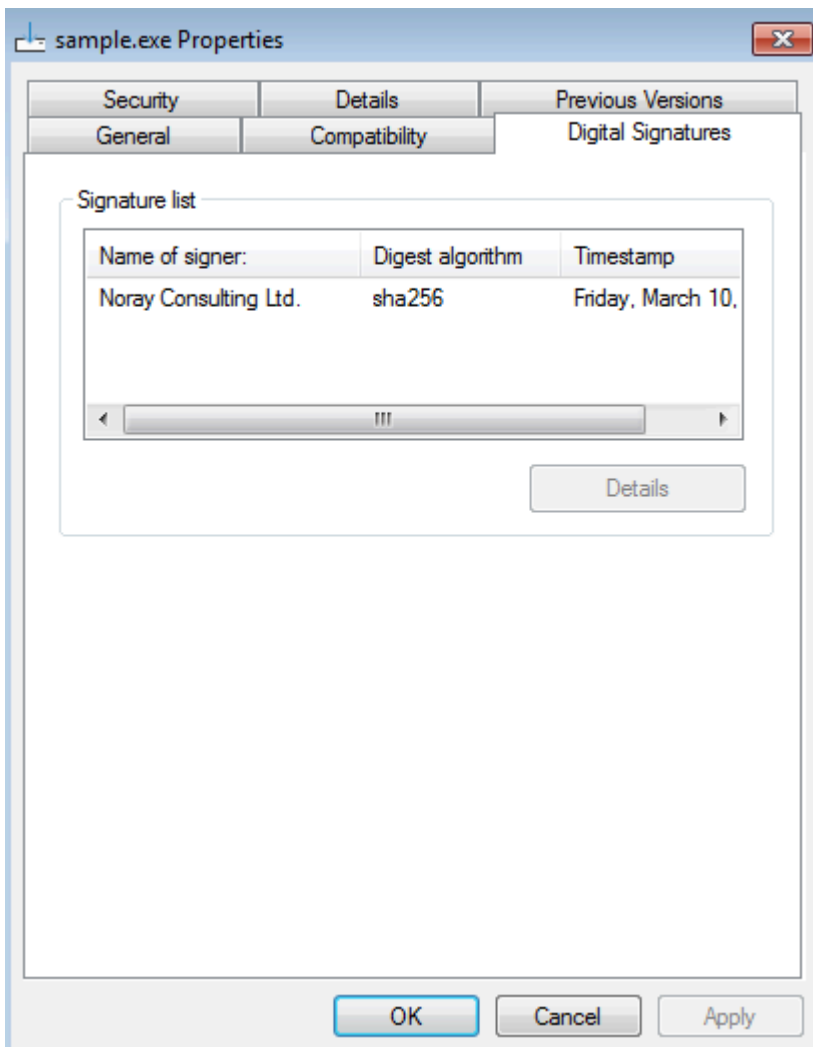


Figure 1-Digital signature of the sample

Figure 1 shows the digital sign and the name of the signer.

desktop.ini	7/14/2009 10:24 AM	Configuration sett...	1 KB
Installer.RemoteDesktopManager.2022.3....	8/9/2023 2:04 PM	Application	1,633 KB
netid718000532.dll0	8/9/2023 2:04 PM	DLL0 File	2,634 KB
pnxmys718000532.dll	8/9/2023 2:04 PM	Application extens...	2,599 KB
Recorded TV	7/14/2009 10:24 AM	Library	1 KB
update.conf	8/9/2023 2:05 PM	CONF File	1 KB

Figure 2-File payloads of our sample

It was observed that the setup file drops RomCom files in C:\Users\Public\Libraries. We observed that all DLLs dropped were VMProtect'ed files.

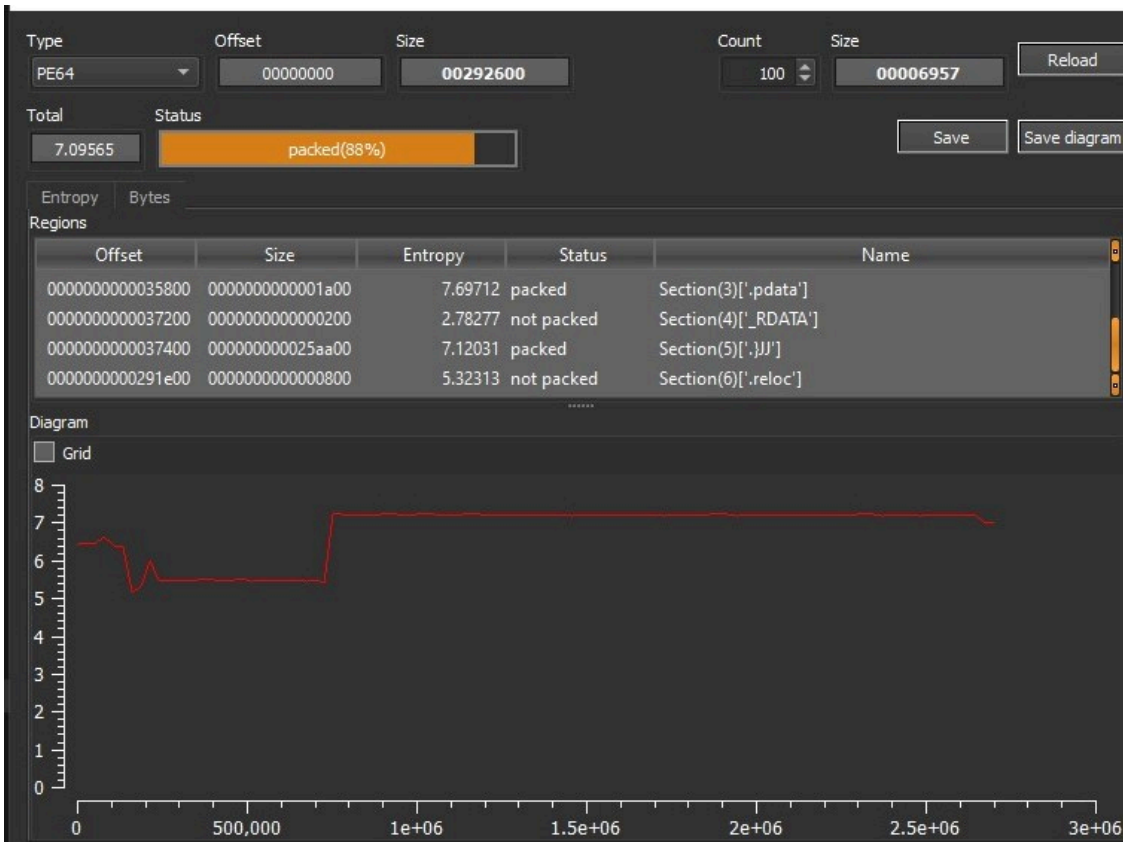


Figure 3-VMProtect packed payload file

From Figure 3, we can see that the dropped file netid7*.dll0 has high entropy and is a VMProtect'ed file.

```
lea rdx, [rbp+130h+var_50]
mov [rcx], ax
xorps xmm0, xmm0
xorps xmm1, xmm1
mov r8d, 0Ah
movups cs:xmmword_180036D40, xmm0
movups cs:xmmword_180036D50, xmm0
movups cs:xmmword_180036D60, xmm0
movups cs:xmmword_180036D70, xmm0
mov rax, gs:60h
movups [rbp+130h+var_50], xmm1
movups [rbp+130h+var_40], xmm1
movups [rbp+130h+var_30], xmm1
movups [rbp+130h+var_20], xmm1
mov ecx, [rax+120h]
call sub_1800093D8
lea rdi, xmmword_180036D40
lea rcx, [rdi-1]
```

Figure 4-Malicious binary accessing PEB

Here we can observe that the malware accesses the Process Environment Block (PEB) using gs:60h, after getting access to which, it checks for the OS Build number using [rax+120h](#)

```
loc_180008B96:
mov rax, [rbp+508h]
mov [rbp+500h+ContextRecord._Rip], rax
lea rax, [rbp+508h]
add rax, 8
mov [rsp+600h+var_590], esi
mov [rbp+500h+ContextRecord._Rsp], rax
mov rax, [rbp+508h]
mov [rbp+500h+var_580], rax
mov [rsp+600h+var_58C], edi
call cs:IsDebuggerPresent
xor ecx, ecx ; lpTopLevelExceptionFilter
mov edi, eax
call cs:SetUnhandledExceptionFilter
lea rcx, [rsp+600h+ExceptionInfo] ; ExceptionInfo
call cs:UnhandledExceptionFilter
```

Figure 5-Is the process being debugged check

Then the current process is checked if it is being run under a debugger.

```
lea    rbp, [rsp-4C0h]
sub    rsp, 5C0h
mov    ebx, ecx
mov    ecx, 17h
call   cs:IsProcessorFeaturePresent
test   eax, eax
jz     short loc_180002D62
```

```
loc_180002D62:
mov    ecx, 3
call   sub_180002D30
xor    edx, edx
lea    rcx, [rbp+4C0h+ContextRecord]
mov    r8d, 4D0h
call   sub_180004A70
lea    rcx, [rbp+4C0h+ContextRecord]
call   cs:RtlCaptureContext
mov    rbx, [rbp+4C0h+ContextRecord._Rip]
```

```
mov    ecx, ebx
int    29h
```

Figure 6-IsProcessorFeaturePresent

It then uses IsProcessorFeaturePresent, the argument 0x17 is passed to check if the __fastfail option is available or not.

```
and    [rbp+arg_8], 0
lea    rcx, [rbp+arg_8]
call   cs:GetSystemTimeAsFileTime
mov    rax, [rbp+arg_8]
mov    [rbp+arg_0], rax
call   cs:GetCurrentThreadId
mov    eax, eax
xor    [rbp+arg_0], rax
call   cs:GetCurrentProcessId
mov    eax, eax
lea    rcx, [rbp+arg_10]
xor    [rbp+arg_0], rax
call   cs:QueryPerformanceCounter
mov    eax, dword ptr [rbp+arg_10]
lea    rcx, [rbp+arg_0]
shl    rax, 20h
xor    rax, [rbp+arg_10]
xor    rax, [rbp+arg_0]
xor    rax, rcx
mov    rcx, 0FFFFFFFFFh
and    rax, rcx
```

Figure 7-Anti Debug Check

Here, QueryPerformanceCounter is being used for anti-debug techniques. When a process is being debugged there is a delay between instruction and execution. By using QueryPerformanceCounter we can measure the delay taken to run each instruction.

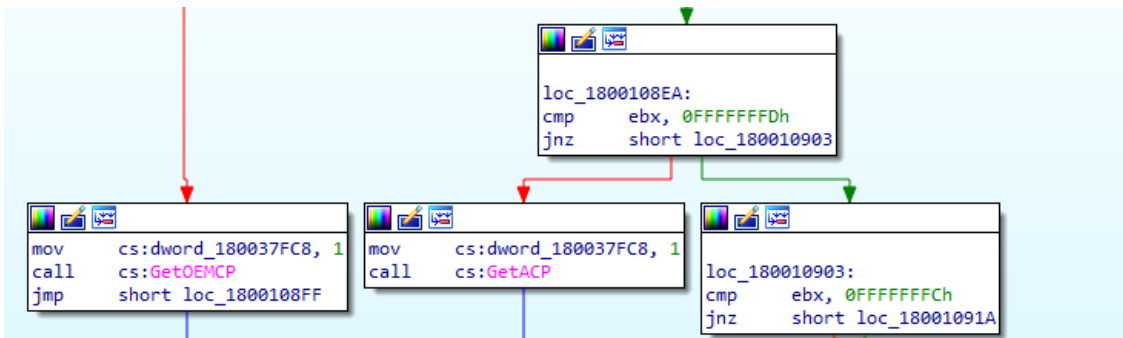


Figure 8-Calling GetOEMCP

In the above figure, we can see that [GetOEMCP](#) is used, which returns the OEM code page identifier of the Operating System.



Figure 9-Locale based exclusion

The malware then checks if the code page identifier is one of zh-CN, zh-TW, ko-KR, ja-JP. This is done to check if there is any clipboard data related to Chinese, Japanese or Korean language. If it is in one of these locales, the malware process throws an exception and terminates.

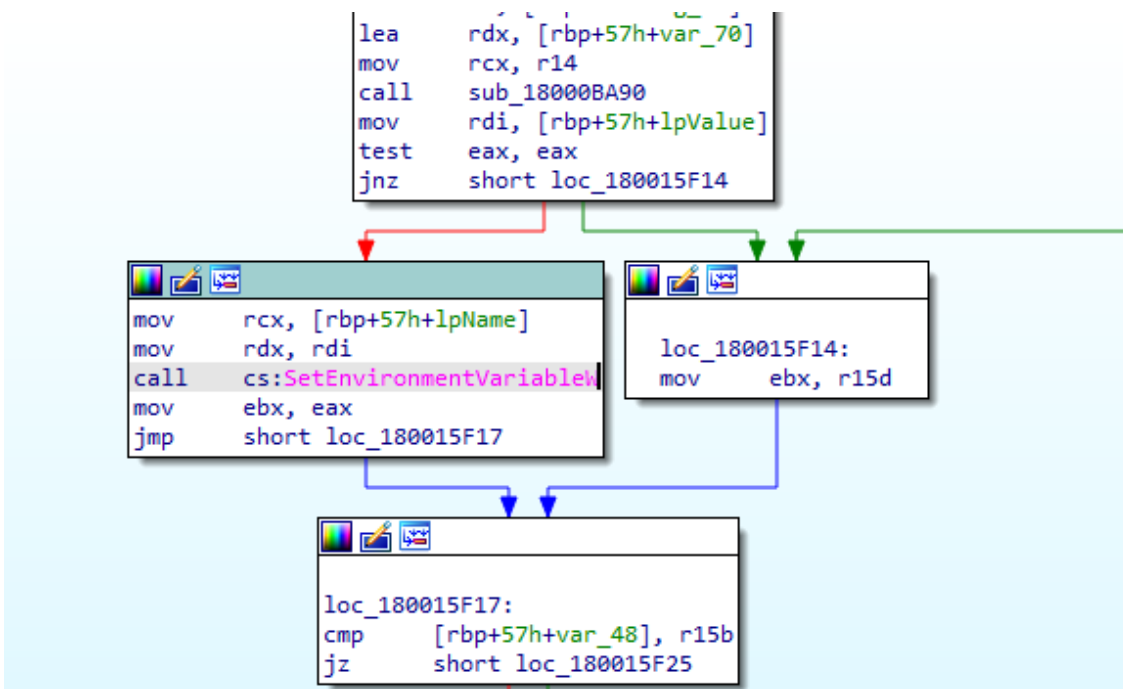


Figure 10-Environment variable

The sample under consideration has the ability to set an Environment Variable, it has been observed that malware tends to bypass the normal order of loading a DLL and loading it from another location.

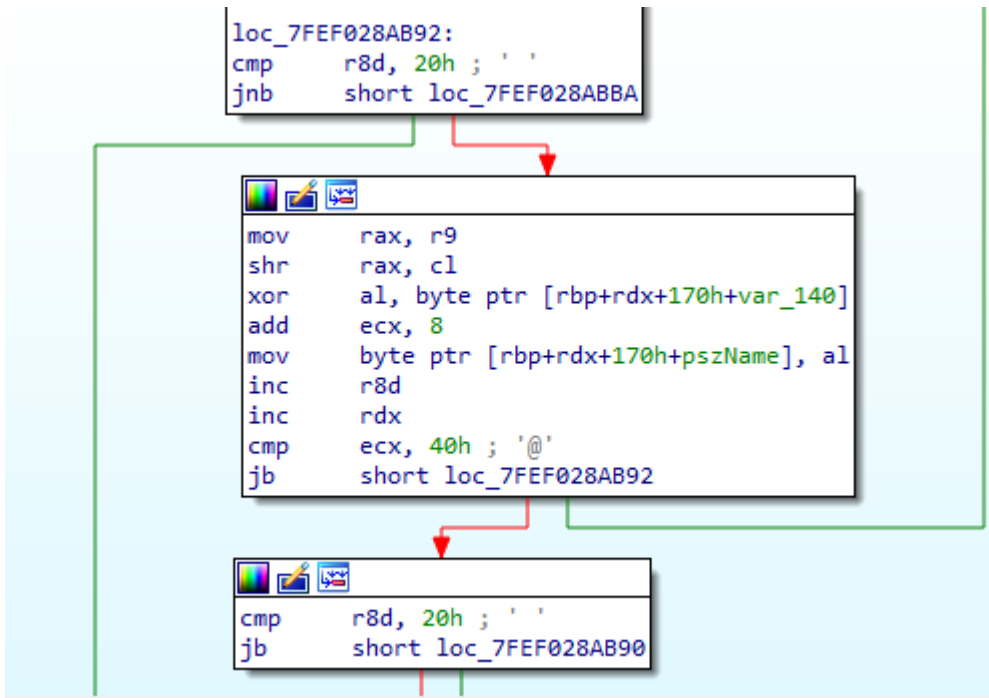


Figure 11-Decrypting Function

The malware keeps all the static data like url, client header, registry value, filename encrypted, which is decrypted as shown in Figure 11.

For example

Encrypted filename –

3B 20 54 18 7E 4F 8C 7F 70 20 5D 18 22 4F 86 7F 70 20 41 18 0C 4F

Decryption key –

08 20 24 18 0C 4F E3 7F

The decrypted filename –

3proxy.exe

Encrypted Rundll32 path –

1B 5A 05 49 6F 95 80 90 2F 13 05 4D 7F 88 90 9A 35 53 6B 42 74 8E 8A 9B 34 0C 6A 2C 28 9E 9C 9A 78

Decryption key –

58 60 59 1E 06 FB E4 FF

Decrypted Rundll32 path –

C:\Windows\System32\rundll32.exe

Encrypted string –

7B 20 2C 16 67 FF 92 7F 7C 20 34 16 63 FF 81 7F 6F 20 2D 16 63 FF CE 7F 66 20 3D 16 72 FF E0 7F

Decryption key-

08 20 58 16 06 FF E0 7F

Decrypted string – s.t.a.r.t.l.e.a.g.u.e...n.e.t

Basic XOR encryption/decryption is at play here.

```

mov [rbp+170h+var_190], rdx
mov [rbp+170h+var_198], rcx
mov rax, [rbp+170h+arg_20]
mov [rbp+170h+var_188], rax
xor r14d, r14d
mov [rbp+170h+dwNumberOfBytesAvailable], r14d
mov [rbp+170h+var_144], 800h
lea rcx, [rbp+170h+pProxyConfig]
call cs:WinHttpGetIEProxyConfigForCurrentUser
mov dword ptr [rsp+270h+pswzServerName], 153D205Fh
mov dword ptr [rsp+270h+pswzServerName+4], 7FA87F68h
mov dword ptr [rsp+270h+pswzServerName+8], 1500205Ch
mov dword ptr [rsp+270h+pswzServerName+0Ch], 7FC07F56h
mov dword ptr [rsp+270h+var_220], 152C204Dh
mov dword ptr [rsp+270h+var_220+4], 7F8D7F67h
mov dword ptr [rsp+270h+var_220+8], 15382078h
mov dword ptr [rsp+270h+var_220+0Ch], 7FCF7F63h
mov [rsp+270h+var_210], 157A2039h
mov [rsp+270h+var_20C], 7FE07F36h
mov r9, 7FE07F0615542008h
mov [rsp+270h+var_208], r9
mov r8d, r14d
mov edx, r14d

```

Figure 12-Obtaining proxy configuration if any

WinHttpGetIEProxyConfigForCurrentUser API is used to get the Internet Explorer proxy configuration for the current user. Then using this the malware can exfiltrate data gathered from the victim.

The screenshot shows a debugger window with assembly code on the left and a memory dump on the right. A red box highlights a `call` instruction: `call qword ptr ds:[&getProcessHeap]`. A comment points to this instruction: "startleauge.net is website related to RomCom RAT". Below the assembly, the memory dump shows the ASCII string "s.t.a.r.t.l.e.a.g.u.e...n.e.t" in a red box, which is the decrypted string from Figure 11.

Figure 13-Establishing C2 connection

Here, we can observe that it's decrypting "startleauge.net" using the decryption function mentioned in Figure 11.

```
mov     r8, rbx
xor     edx, edx
mov     rcx, rax
call   sub_180004A70
mov     [rbp+170h+dwNumberOfBytesRead], r13d
lea    r9, [rbp+170h+dwNumberOfBytesRead]
mov     r8d, [rbp+170h+dwNumberOfBytesAvailable]
mov     rdx, rdi
mov     rcx, rsi
call   cs:WinHttpReadData
test   eax, eax
jz     short loc_18001A539
```

```
mov     ebx, [rbp+170h+dwNumberOfBytesAvailable]
mov     r8d, ebx
movsxd rcx, r14d
add     rcx, r12
mov     rdx, rdi
call   sub_1800043C0
add     r14d, ebx
```

Figure 14-File download

RomCom RAT uses WinHttpReadData API to download any file which is pushed by the C2.

```
mov     [rbp+1480h+pSessionId], r14d
mov     ebx, 300007Fh
call   cs:GetCurrentProcessId
mov     ecx, eax
lea    rdx, [rbp+1480h+pSessionId]
call   cs:ProcessIdToSessionId
test   eax, eax
jz     short loc_18001CB63
```

```
mov     ecx, [rbp+1480h+pSessionId]
shl    ecx, 8
call   cs:htonl
lea    ebx, [rax+300007Fh]
```

Figure 15-RDP connection check

The malware tries to check if any active RDP session is live using the API ProcessIdToSessionId then proceeds to use the retrieved session ID to establish connection.

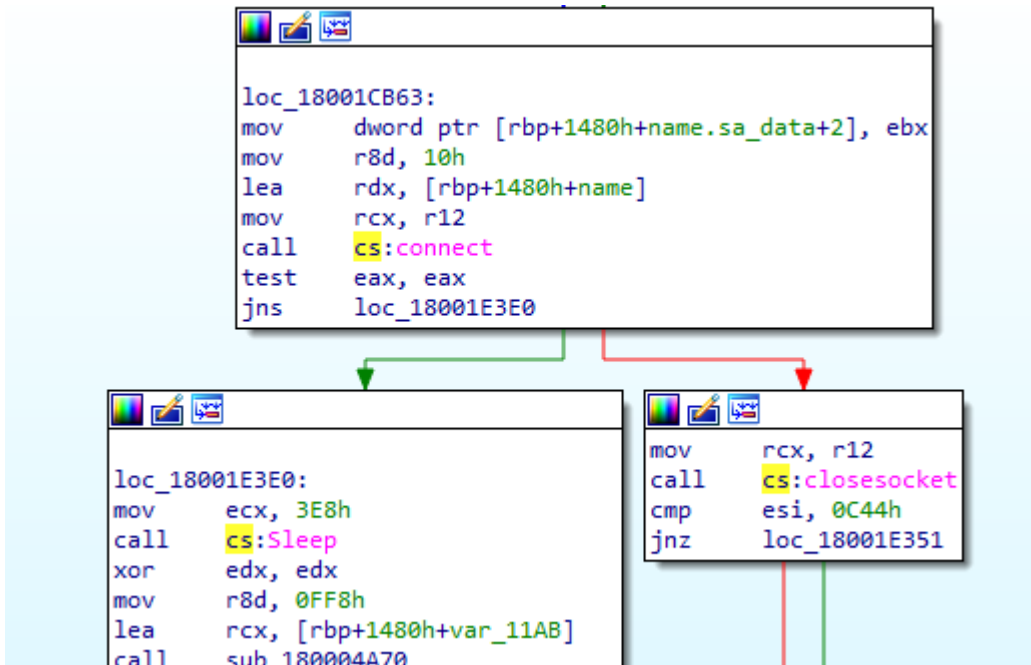


Figure 16-C2 Connection

In Figure 16 we can observe that this RomCom RAT is trying to connect to C2. However, if the connection is not established then there is a sleep time before checking again.

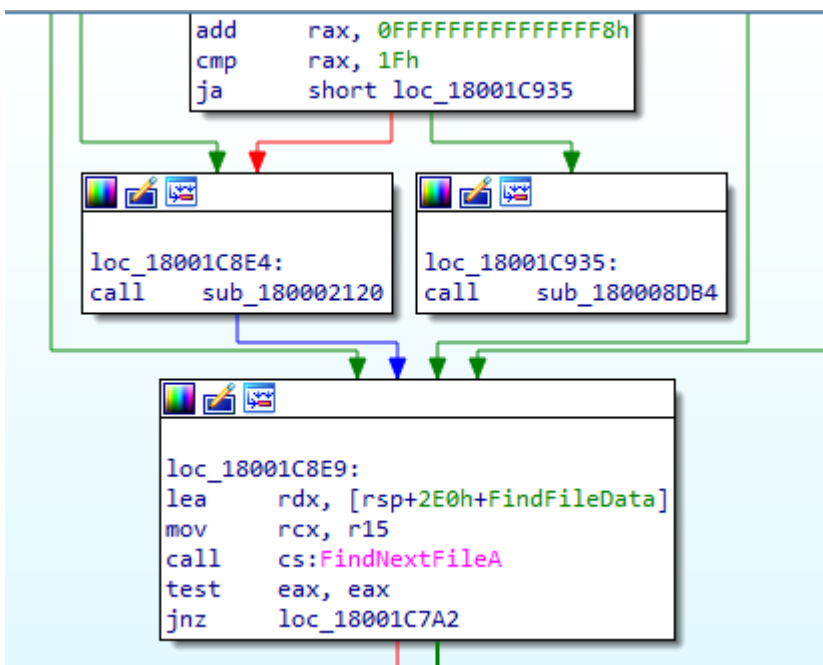


Figure 17-Performing local reconnaissance

It traverses the file system using `FindFirstFileA`, `FindNextFileA` and collects a list of filenames and sends it to C2.

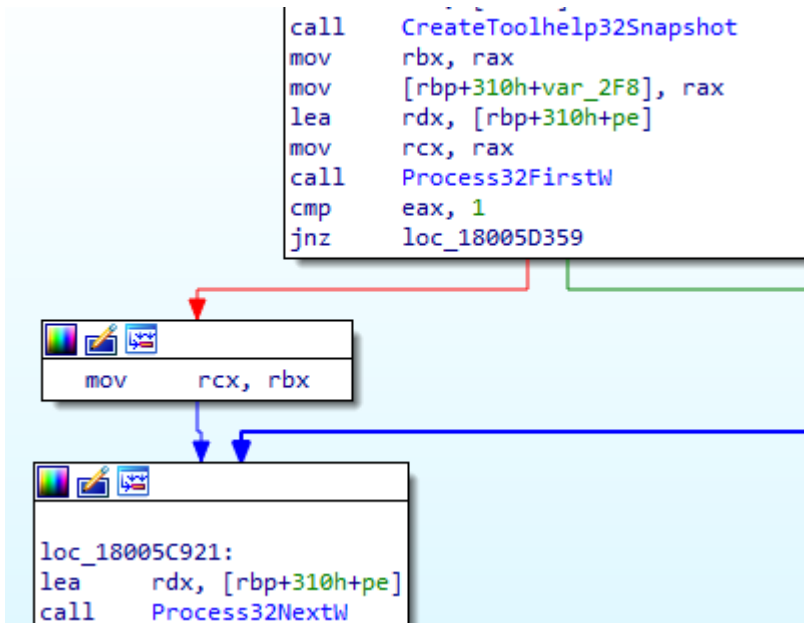


Figure 18-Eyeing list of running processes

It uses CreateToolHelp32Snapshot and then iterates through the process using Process32FirstW and Process32NextW which is used to list all the running processes.

It has been observed that in a number of instances RomCom threat actors have used fake websites and applications to do its malicious activity.

We at K7 Labs provide detection for RomCom RAT and all the latest threats. Users are advised to use a reliable security product such as “K7 Total Security” and keep it up-to-date to safeguard their devices.

Indicators of Compromise (IOCs)

Hash	Detection Name
007A67BFA732084B3F8278B302BEF49E	Trojan (005a54be1)
6F47723E5FC6E96AB5E9F96F6BC585FA	Trojan (00566ad51)
46AC4B26D35F619D8A1415B5E4365A52	Trojan (005a3e761)

C2

startleauge.net

References

https://www.trendmicro.com/en_in/research/23/e/void-rabisu-s-use-of-romcom-backdoor-shows-a-growing-shift-in-th.html

<https://blogs.blackberry.com/en/2023/07/romcom-targets-ukraine-nato-membership-talks-at-nato-summit>

Source: <https://labs.k7computing.com/index.php/romcom-rat-not-your-typical-love-story/>