



Practical Malware Analysis & Triage

Malware Analysis Report

njRAT Remote Access Trojan

Aug 2022 | Zo | v1.0



Table of Contents

Table of Contents	2
Executive Summary	3
High-Level Technical Summary	4
Malware Composition	5
njRAT.exe:.....	5
njq8.exe:.....	5
windows.exe:.....	5
Basic Static Analysis	6
Basic Dynamic Analysis	9
Advanced Static Analysis	12
Advanced Dynamic Analysis.....	13
Indicators of Compromise.....	14
Network Indicators.....	14
Host-based Indicators	15
Appendices.....	16
A. njRAT Yara Rule.....	16
B. Callback URLs	16
C. Registry keys	16
D. Files.....	16



Executive Summary

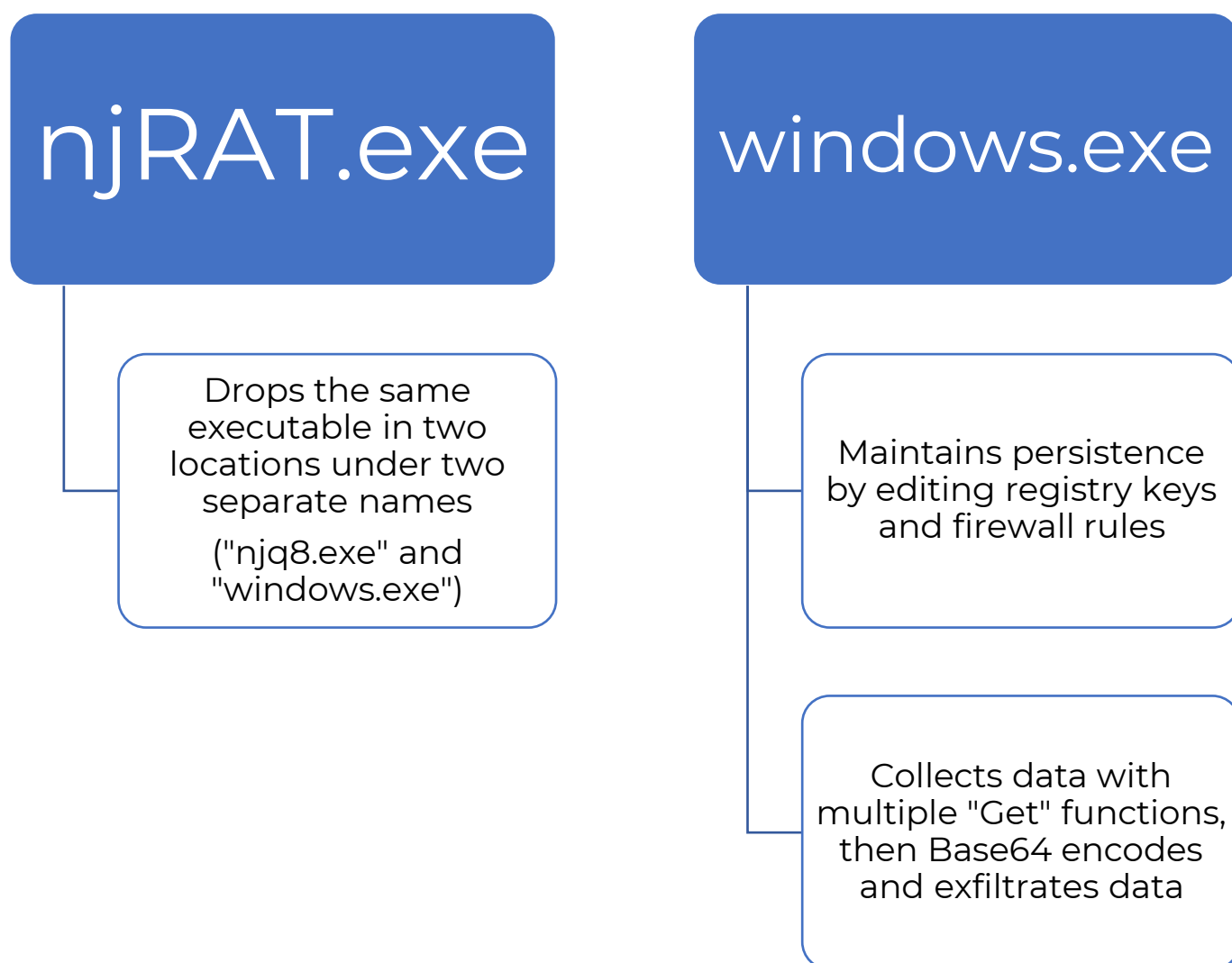
Sample Name	njRAT.exe
SHA256 hash	FD624AA205517580E83FAD7A4CE4D64863E95F62B34AC72647B1974A52822199
VirusTotal Detections	Flagged by 62/69 vendors as malicious: https://www.virustotal.com/gui/file/fd624aa205517580e83fad7a4ce4d64863e95f62b34ac72647b1974a52822199
Source	https://github.com/ytisf/theZoo/tree/master/malware/Binaries/njRAT-v0.6.4
Language	C# (.NET Framework)
Architecture	32 bit

njRAT is a remote access trojan that was first identified on January 1st, 2013. It consists of two payloads that are executed in succession, beginning with the initial dropper, and ending with “windows.exe” as the primary spyware and stealer payload. Symptoms of infection from this sample include continuous beaconing and exfiltration to the C2 address, “hxxp://zaaptoo.zapto.org”, a Microsoft .NET Framework “Unhandled exception” popup, and the executable “windows.exe” appearing in the %TEMP% directory. Since njRAT is currently one of the most widely used RATs in the world, callback URL’s and file names may vary depending on the threat actor and target.

A YARA signature rule and a list of IOCs are attached in Appendix A.

High-Level Technical Summary

njRAT consists of two parts: a dropper and an unpacked stage 2 spyware and stealer executable. It first unpacks its stage 2 payload, then begins stealing host information and user input. If the C2 domain, `hxxp://zaaptoo.zapto[.]org`, is reachable at the time of detonation it will begin exfiltrating immediately. Otherwise, it automatically detects if the domain comes back online and will exfiltrate data once a DNS query is successful.



Malware Composition

njRAT consists of the following components:

File Name	SHA256 Hash
njRAT.exe	FD624AA205517580E83FAD7A4CE4D64863E95F62B34AC72647B1974A52822199
njq8.exe	CE6421107031175F39E61D3BCC5A98D1D94190E250034E27CDBEBBADCBA084A4
windows.exe	CE6421107031175F39E61D3BCC5A98D1D94190E250034E27CDBEBBADCBA084A4

njRAT.exe:

The initial executable, typically sent in a phishing attachment.

njq8.exe:

A dropped file with the same hash as “windows.exe”, located at C:\njq8[.]exe. Appears to be a backup of the primary payload and is named after the Trojan’s original author (twitter handle “@njq8” appears in the strings output).

windows.exe:

Spyware that gathers system information such as computer name, browser passwords, camera and microphone access, and keystrokes, among other data. The file is located at C:\Users\<User>\AppData\Local\Temp\windows[.]exe. Copies user activity into a “.tmp” file, as shown below:

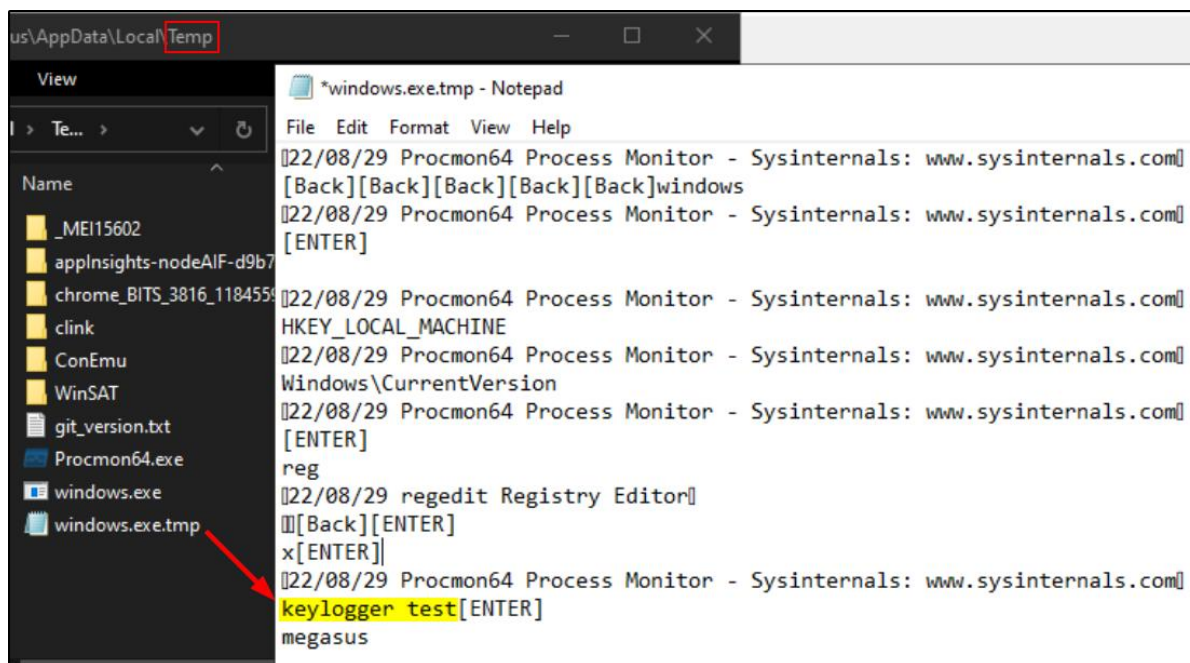


Fig 1: Keylogger output from windows.exe



Basic Static Analysis

Tools used: *floss.exe*, *PEView*, *PEStudio*, *PEiD*

After initial detonation of the malware, the only immediately visible indicator was a .NET Framework error:

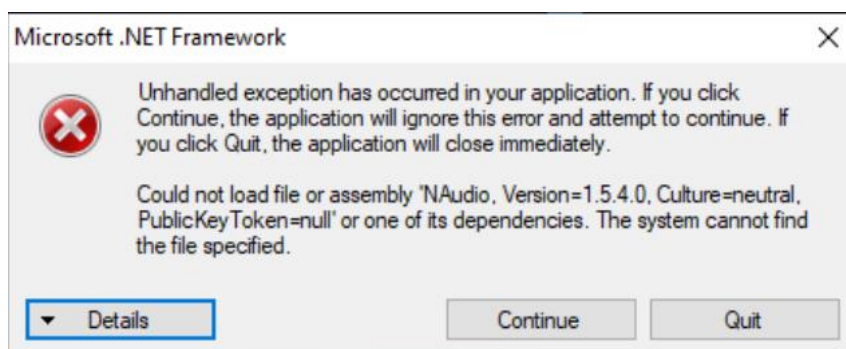


Fig 2: Initial error message

The first step after that was to look at the strings from the malware. I used “floss.exe” view strings and listed the most significant ones found in the table below:

Malware version, modules used, and link to author's twitter (likely indicates commercial malware)	Project : njRat Verison : 0.6.4 Coded By : njq8 FireFox Stealer : DarkSel Paltalk Stealer : protofag Chrome Stealer : RockingWithTheBest Opera Stealer : Black-Blood, KingCobra Icon Changer : Miharbi Thnx To : MaSad ,CoBrAXx Twitter : https://twitter.com/njq8
Executables referenced	ClassLibrary1.exe EnKSaR.HaCKeR.exe njq8.exe
Domains	zaaptoo.zapto[.]org
Interesting commands	netsh firewall add allowedprogram "
“mscorlib” referenced - an indicator that the malware was written in C#	FLOSS static ASCII strings !This program cannot be run in DOS mode. `.sdata @.reloc !System.Resources.ResourceReader, mscorlib , PADPADPF !This program cannot be run in DOS mode.

Analyzing the binary in PEStudio provided the sample's file hash, architecture, and compilation time for the malware, which was **"Fri Sep 27 08:00:20 2013"**. It also listed that the original filename of "windows.exe" as "ClassLibrary1.exe" and "njRAT.exe" as "EnKSaR.HaCKeR.exe", which were both referenced in the strings.

resources (6)	InternalName	EnKSaR.HaCKeR.exe
strings (size)	LegalCopyright	Copyright © njq8 2013
debug (Sep.2013)	LegalTrademarks	njRAT
manifest (administrator)	OriginalFilename	EnKSaR.HaCKeR.exe
version (EnKSaR.HaCKeR.exe)	ProductName	njRAT
overlay (n/a)	ProductVersion	0.6.4.0

Fig 3: PEStudio file analysis

The malware is unlikely to be packed since PEview showed no significant difference between the malware's "Size of Raw Data" and "Virtual Size", and since PEiD was unable to detect a known packer for this executable (output lists "Nothing found").

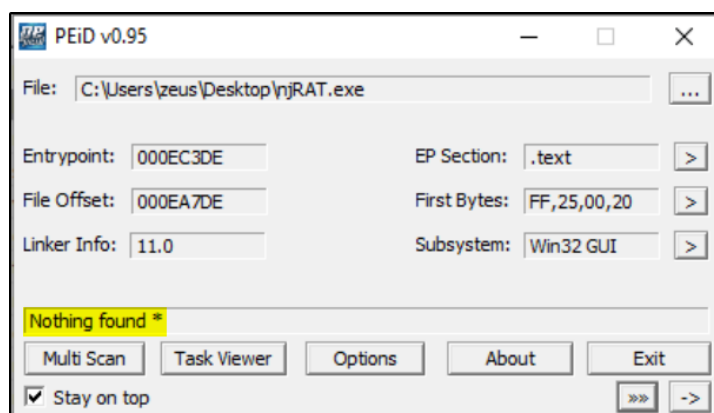


Fig 4: No malware packer identified



Multiple imported API functions from the stage 2 executable “windows.exe” were potential spyware indicators. A few are highlighted in the example below:

Evasion ?	Spying ?
CreateFileMappingA	AttachThreadInput
DeleteFileA	CallNextHookEx
GetModuleHandleA	GetAsyncKeyState
GetProcAddress	GetClipboardData
LoadLibraryA	GetDC
LoadLibraryExA	GetDCEX
LoadResource	GetForegroundWindow
SetEnvironmentVariableA	GetKeyboardState

Fig 5: Suspicious functions mapped, source: malapi.io

Basic Dynamic Analysis

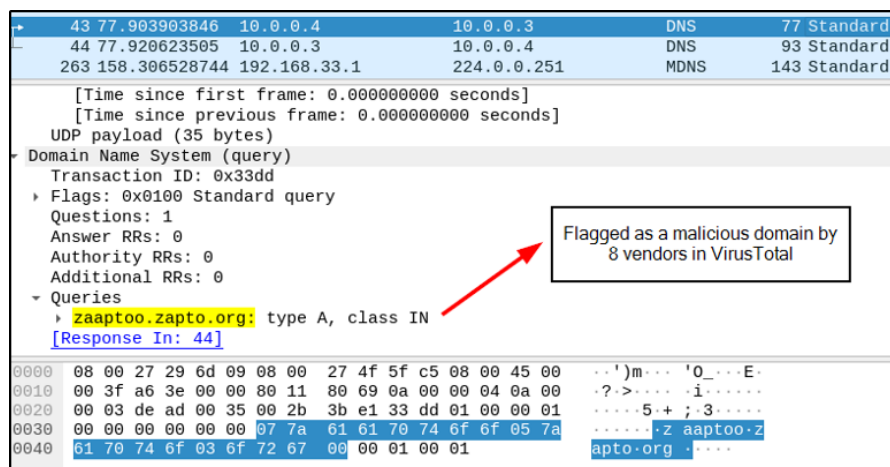
Tools used: TCPView, inetsim, Wireshark, netcat, procmon (Process Monitor)

The malware's beaoning activity was visible in TCPView, where it would create a new TCP SYN connection attempt every few seconds to remote port 1177.

Process Name	Process ID	Protocol	State	Local Address	Local Port	Remote Address	Remote Port
svchost.exe	800	TCP	Listen	0.0.0.0	135	0.0.0.0	0
System	4	TCP	Listen	10.0.0.4	139	0.0.0.0	0
System	4	TCP	Listen	169.254.176.82	139	0.0.0.0	0
windows.exe	1236	TCP	Syn Sent	10.0.0.4	1422	10.0.0.3	1177

Fig 6: Beaoning from windows.exe in TCPView

These connections could only start after a successful DNS query to the C2 domain “zaaptoo.zapto[.]org”. The successful connection was simulated by inetsim and can be seen in the Wireshark packet capture below.



No.	Time	Source	Destination	Protocol	Length	Info
43	77.903903846	10.0.0.4	10.0.0.3	DNS	77	Standard
44	77.920623505	10.0.0.3	10.0.0.4	DNS	93	Standard
263	158.306528744	192.168.33.1	224.0.0.251	MDNS	143	Standard

[Time since first frame: 0.000000000 seconds]
[Time since previous frame: 0.000000000 seconds]
UDP payload (35 bytes)
Domain Name System (query)
Transaction ID: 0x33dd
Flags: 0x0100 Standard query
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
Queries
zaaptoo.zapto.org: type A, class IN
[Response In: 44]

Flagged as a malicious domain by 8 vendors in VirusTotal

Fig 7: Initial beacon to command-and-control URL

After reaching the C2 URL, it began encoding and exfiltrating data to this address. I captured the data in transit by editing the hosts file to point “zaaptoo.zapto[.]org” to the virtual machine’s loopback address (127.0.0.1) and then set up a netcat listener on port 1177.

```
C:\Users\zeus
λ netcat -nlvp 1177
Ncat: Version 5.9BETA1 ( http://nmap.org/netcat )
Ncat: Listening on 0.0.0.0:1177
Ncat: Connection from 127.0.0.1:2509.
lv|'|SGFjS2VkXzI4QjU5OUY0|'|DESKTOP-DPGS83P|'|zeus|'|2022-08-29|'|'|Win 10 Enterprise B
f)act|'|[eof]act|'|Q21kZXI=[eof]act|'|U2V0dGluZ3M=[eof]act|'|Q21kZXI=[eof]C:\Users\zeus\
Decoded from Base64 = "Cmder" (application currently in use)
```



To verify if any other changes were made on the system, I checked the process tree in procmon for a basic overview njRAT's activity.

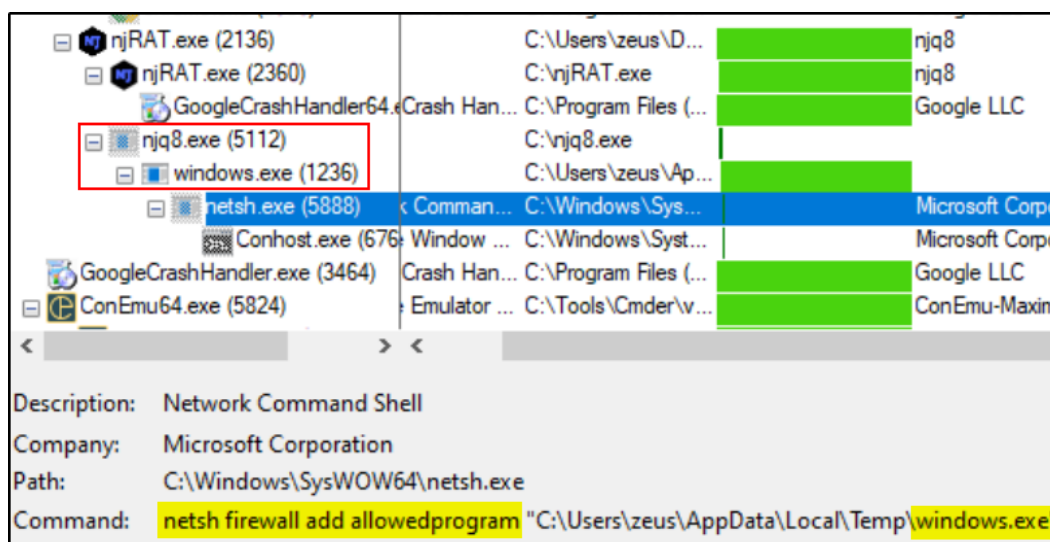


Fig 8: Process tree for njRAT in procmon

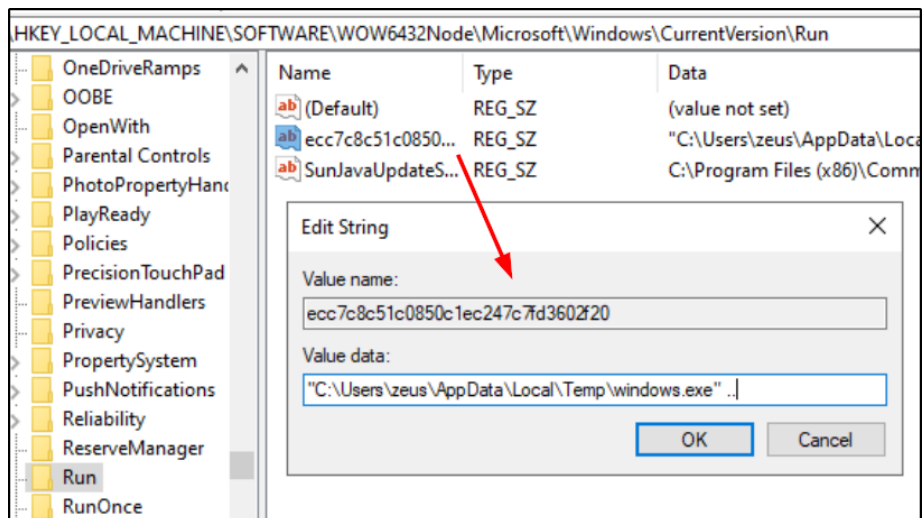
Here we can see that njRAT drops two executables. The primary focus is the stage 2 payload, windows.exe, which is seen whitelisting itself in Defender.

I filtered procmon to look for any other activity from windows.exe, focusing on file and registry key changes. I found that the binary was continuously recreating two "Run" keys in the registry, both under the name "ecc7c8c51c0850c1ec247c7fd3602f20".

Time ...	Process Name	PID	Operation	Path
11:48:...	windows.exe	4892	RegSetValue	HKCU\\Environment\\SEE_MASK_NOZONECHECKS
11:48:...	windows.exe	4892	RegSetValue	HKCU\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run\\ecc7c8c51c0850c1ec247c7fd3602f20
11:48:...	windows.exe	4892	RegSetValue	HKLM\\SOFTWARE\\WOW6432Node\\Microsoft\\Windows\\CurrentVersion\\Run\\ecc7c8c51c0850c1ec247c7fd3602f20
11:48:...	windows.exe	4892	RegSetValue	HKCU\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run\\ecc7c8c51c0850c1ec247c7fd3602f20
11:48:...	windows.exe	4892	RegSetValue	HKLM\\SOFTWARE\\WOW6432Node\\Microsoft\\Windows\\CurrentVersion\\Run\\ecc7c8c51c0850c1ec247c7fd3602f20
11:48:...	windows.exe	4892	RegSetValue	HKCU\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run\\ecc7c8c51c0850c1ec247c7fd3602f20
11:48:...	windows.exe	4892	RegSetValue	HKLM\\SOFTWARE\\WOW6432Node\\Microsoft\\Windows\\CurrentVersion\\Run\\ecc7c8c51c0850c1ec247c7fd3602f20



Checking these values in Registry Editor reveals that windows.exe is being launched at startup, and that both the registry entry and file itself will respawn if deleted.





Advanced Static Analysis

Tools used: dnSpy, Cutter

Since the malware sample was written in C#, dnSpy was able to recreate what the code potentially looked like. Below is a function that retrieves and stores the computer hostname:

```
197
198 // Token: 0x0600002B RID: 43 RVA: 0x000AAB64 File Offset: 0x000A9D64
199 internal unsafe static string GetComputerName()
200 {
201     Span<char> span = new Span<char>(stackalloc byte[(UIntPtr)32], 16);
202     Span<char> span2 = span;
203     uint length = (uint)span2.Length;
204     if (Interop.Kernel32.GetComputerName(MemoryMarshal.GetReference<char>(span2), ref length) == 0)
205     {
206         return null;
207     }
208     return span2.Slice(0, (int)length).ToString();
209 }
```

Loading the main executable into Cutter rendered more of an insight into the data collected by the binary, including remote desktop and registry manipulation capabilities.

Functions				Strings	
Name	Size	Imp.	Offs	Address	String
entry0	6	0x00	0x00	0x00440711	FromLINKToolStripMenuItem.Image
fcn.004284e9	116	0x00	0x00	0x00440754	FromLinkToolStripMenuItem1.Image
fcn.0046e21f	41	0x00	0x00	0x00440799	GetPasswordsToolStripMenuItem.Image
fcn.0046f107	1017	0x00	0x00	0x004407e4	IMG2.ImageStream
fcn.0047627f	1877	0x00	0x00	0x00440809	KeyloggerToolStripMenuItem.Image
fcn.00491537	1	0x00	0x00	0x0044084e	LargeToolStripMenuItem.Image
fcn.004930d5	1791	0x00	0x00	0x0044088b	ListToolStripMenuItem.Image
fcn.004a4d67	3	0x00	0x00	0x004408c6	MediumToolStripMenuItem.Image
fcn.004a5c4e	2	0x00	0x00	0x00440905	OpenChatToolStripMenuItem.Image
fcn.004a5ccf	48	0x00	0x00	0x00440948	OpenFolderToolStripMenuItem.Image
fcn.004b669b	335	0x00	0x00	0x0044098f	ProcessManagerToolStripMenuItem1.Image
fcn.004c2d52	25	0x00	0x00	0x004409e0	RegistryToolStripMenuItem.Image
fcn.004c2d52	25	0x00	0x00	0x00440a23	RemoteCamToolStripMenuItem1.Image
fcn.004c2d52	25	0x00	0x00	0x00440a6a	RemoteDesktopToolStripMenuItem.Image
fcn.004c2d52	25	0x00	0x00	0x00440ab7	RemoteShellToolStripMenuItem.Image
fcn.004cd3e1	509	0x00	0x00	0x00440b00	RenameToolStripMenuItem.Image
fcn.004cd8bb	97	0x00	0x00	0x00440b3f	RestartToolStripMenuItem.Image
				0x00440b80	RunFileToolStripMenuItem.Image



Advanced Dynamic Analysis

Tools used: x32dbg

Debugging njRAT revealed similar information as found with advanced static analysis, mainly including the vast information-gathering utilities of this malware.

<pre>push 1 lea ecx,dword ptr ss:[esp+C] call sechost.7678FE30 push sechost.767A3DA8 lea ecx,dword ptr ss:[esp+8] call sechost.7678F860 lea eax,dword ptr ss:[esp+4] mov ecx,esi push eax call sechost.7678E7E0 push 33 push sechost.767A3D40 mov ecx,eax call sechost.7678E69F push 0 push 1 lea ecx,dword ptr ss:[esp+C] call sechost.7678FE30 push sechost.767A3D0C lea ecx,dword ptr ss:[esp+8] call sechost.7678F860 lea eax,dword ptr ss:[esp+4] mov ecx,esi push eax call sechost.7678E7E0</pre>	<pre>[esp+C]:"<L\$T3İè#N" 767A3DA8:L"devicecapabilitymicrophone" eax:"MOCà\x01" 767A3D40:L"S-1-15-3-787448254-1207972858-3" eax:"MOCà\x01" [esp+C]:"<L\$T3İè#N" 767A3D0C:L"devicecapabilitycamera" eax:"MOCà\x01"</pre>
---	---



Indicators of Compromise

The full list of IOCs can be found in the Appendices.

Network Indicators

Process Name	Process ID	Protocol	State	Local Address	Local Port	Remote Address	Remote Port
svchost.exe	800	TCP	Listen	0.0.0.0	135	0.0.0.0	0
System	4	TCP	Listen	10.0.0.4	139	0.0.0.0	0
System	4	TCP	Listen	169.254.176.82	139	0.0.0.0	0
Windows.exe	1236	TCP	Syn Sent	10.0.0.4	1422	10.0.0.3	1177

Fig 9: Windows.exe exfiltration attempts to C2 domain (remote address is inetsim)

```
Domain Name System (query)
  Transaction ID: 0x33dd
  Flags: 0x0100 Standard query
  Questions: 1
  Answer RRs: 0
  Authority RRs: 0
  Additional RRs: 0
  Queries
    zaaptoo.zapto.org: type A, class IN
    [Response In: 44]
```

Fig 10: WireShark Packet Capture of initial beacon check-in

964	427.549118944	10.0.0.4	10.0.0.3	TCP	66 [TCP Retransmission] [TCP Port numbers reused]
965	427.549175862	10.0.0.3	10.0.0.4	TCP	54 1177 → 1474 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
966	428.080836600	10.0.0.4	10.0.0.3	TCP	66 [TCP Retransmission] [TCP Port numbers reused]
967	428.080864206	10.0.0.3	10.0.0.4	TCP	54 1177 → 1474 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
968	428.476414513	192.168.33.1	239.255.255.250	SSDP	143 M-SEARCH * HTTP/1.1
969	428.595350561	10.0.0.4	10.0.0.3	TCP	66 [TCP Retransmission] [TCP Port numbers reused]
970	428.595385528	10.0.0.3	10.0.0.4	TCP	54 1177 → 1474 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
971	429.096830356	10.0.0.4	10.0.0.3	TCP	66 [TCP Retransmission] [TCP Port numbers reused]
972	429.096867146	10.0.0.3	10.0.0.4	TCP	54 1177 → 1474 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
Transmission Control Protocol, Src Port: 1474, Dst Port: 1177, Seq: 0, Len: 0					
Source Port: 1474					
Destination Port: 1177					
[Stream index: 62]					
[Conversation completeness: Incomplete (37)]					
[TCP Segment Len: 0]					
Sequence Number: 0 (relative sequence number)					
Sequence Number (raw): 2699807054					
[Next Sequence Number: 1 (relative sequence number)]					
Acknowledgment Number: 0					
Acknowledgment number (raw): 0					
1000 = Header Length: 32 bytes (8)					
Flags: 0x002 (SYN)					

Fig 11: WireShark packet capture of spurious retransmissions during exfiltration attempts



Host-based Indicators

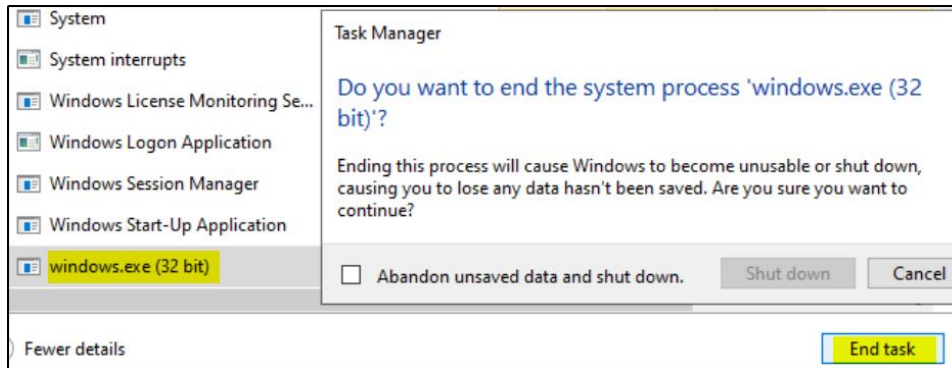


Fig 12: Windows.exe running in Task Manager's Details and Startup tabs

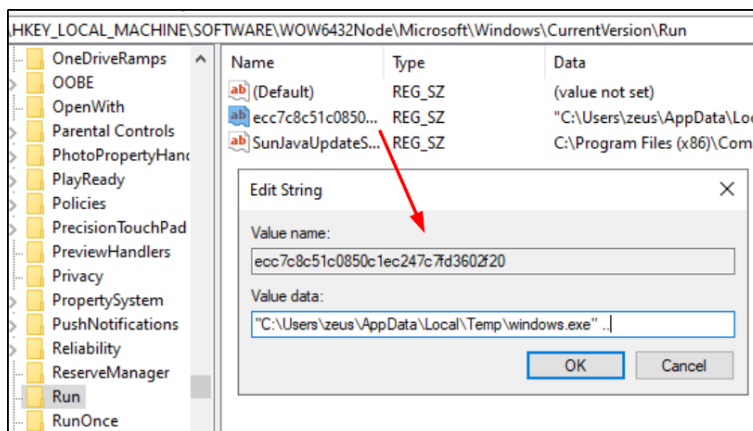


Fig 13: Windows.exe registry entry for persistence

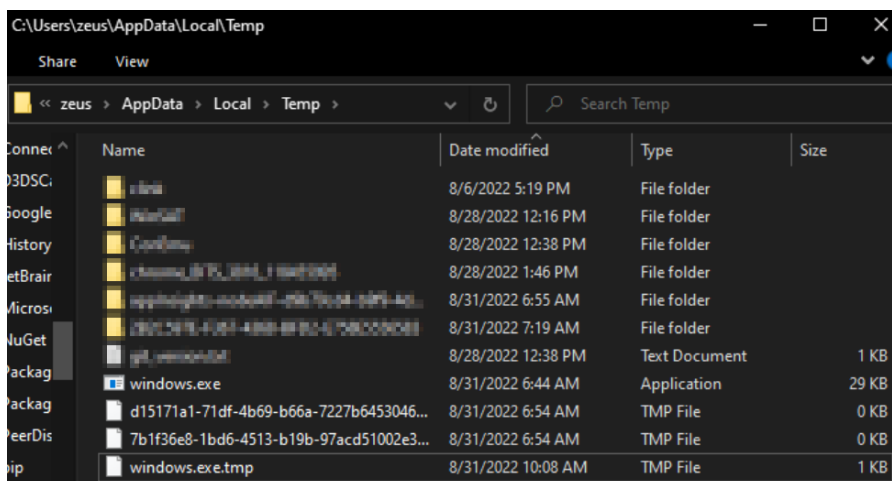


Fig 14: Location of the stage 2 payload and log file



Appendices

A. njRAT Yara Rule

```
rule njRAT {  
  
  meta:  
    last_updated = "2022-08-29"  
    author = "Z"  
    description = "Yara rule for njRAT.exe"  
  
  strings:  
    $string1 = "EnKSaR.HaCKeR"  
    $string2 = "njRAT.exe"  
    $string3 = "https://twitter.com/njq8"  
    $PE_magic_byte = "MZ"  
  
  condition:  
    $PE_magic_byte at 0 and  
    ($string1 and $string2 and $string3)  
}
```

B. Callback URLs

Domain	Port
hxxp://zaaptoo.zapto.org	1177

C. Registry keys

- HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\ecc7c8c51c0850c1ec247c7fd3602f20
- HKLM\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Run\ecc7c8c51c0850c1ec247c7fd3602f20

D. Files

- C:\njRAT[.].exe
- C:\njq8[.].exe
- C:\Users\<User>\AppData\Local\Temp\windows[.].exe