Software Reverse Engineering

Final Project Analysis

Course: CSCE 652 - Software Reverse Engineering

Semester: Spring 2025

Professor: Dr. Martin Carlisle

By

Name: Chiraanth Attanti

UIN: 936002443

Github Repo

Malware Analyzed: Remote Access Trojan
SHA256: c582b5864a67c2d63f8d3a8faf08b47e94646a96cd81abe507d8d08df13e40c4
File Size: 2.32 MB
Environment: Windows 7 Virtual Machine (VM) with tools including PEiD, Ghidra, Wireshark, INetSim, RegShot, ProcMon, x64dbg

Github: Tool and analysis related proofs available at **https://github.com/chiraanth/njrat-analysis**

Unpacking the Malware

Tool Used: PEiD v0.95

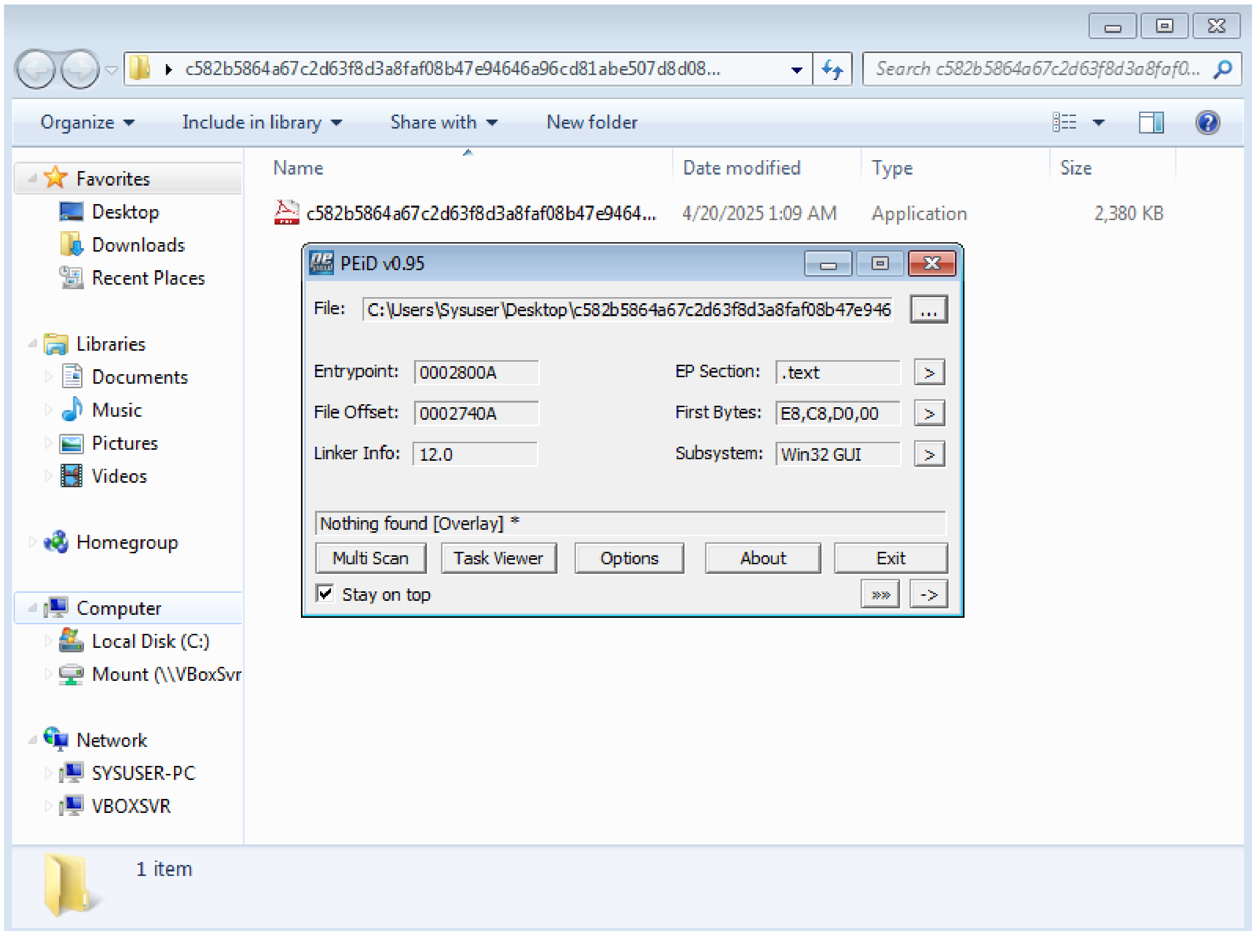
Before diving into analysis, I first wanted to confirm whether this malware was packed or obfuscated in any way. Packing is pretty common in malware to make static analysis harder. I used PEiD, which is a lightweight tool that can detect common packers like UPX, or custom wrappers.

Once loaded, PEiD didn't show any packer signature — it said "Nothing found [Overlay]*". That already suggested it might be unpacked. I double-checked the section info:

- **Entry Point (EP)**: 0x0002800A
- **EP Section**: .text (which is a good sign)
- **File Offset**: 0x0002740A
- **First Bytes**: E8 C8 D0 00 — these looked like x86 instructions instead of shellcode.

So based on all this, I figured it's safe to assume that the malware is not packed and I can move ahead with static analysis directly in Ghidra. This saves a lot of time, especially since unpacking manually is a pain.

The malware doesn't appear to be packed or obfuscated. No signs of runtime unpacking either. It's ready for static analysis in Ghidra and other tools.



Name	Date modified	Type	Size
c582b5864a67c2d63f8d3a8faf08b47e9464...	4/20/2025 1:09 AM	Application	2,380 KB

PEiD v0.95

File: C:\Users\Sysuser\Desktop\c582b5864a67c2d63f8d3a8faf08b47e946

Entrypoint: 0002800A
EP Section: .text
File Offset: 0002740A
First Bytes: E8,C8,D0,00
Linker Info: 12.0
Subsystem: Win32 GUI

Nothing found [Overlay] *

☑ Stay on top

Figure 1: PEiD confirms that no packer is detected and the entry point lies in the .text section.

String Analysis

Tool Used: Ghidra (Defined Strings – Loaded Blocks Only)

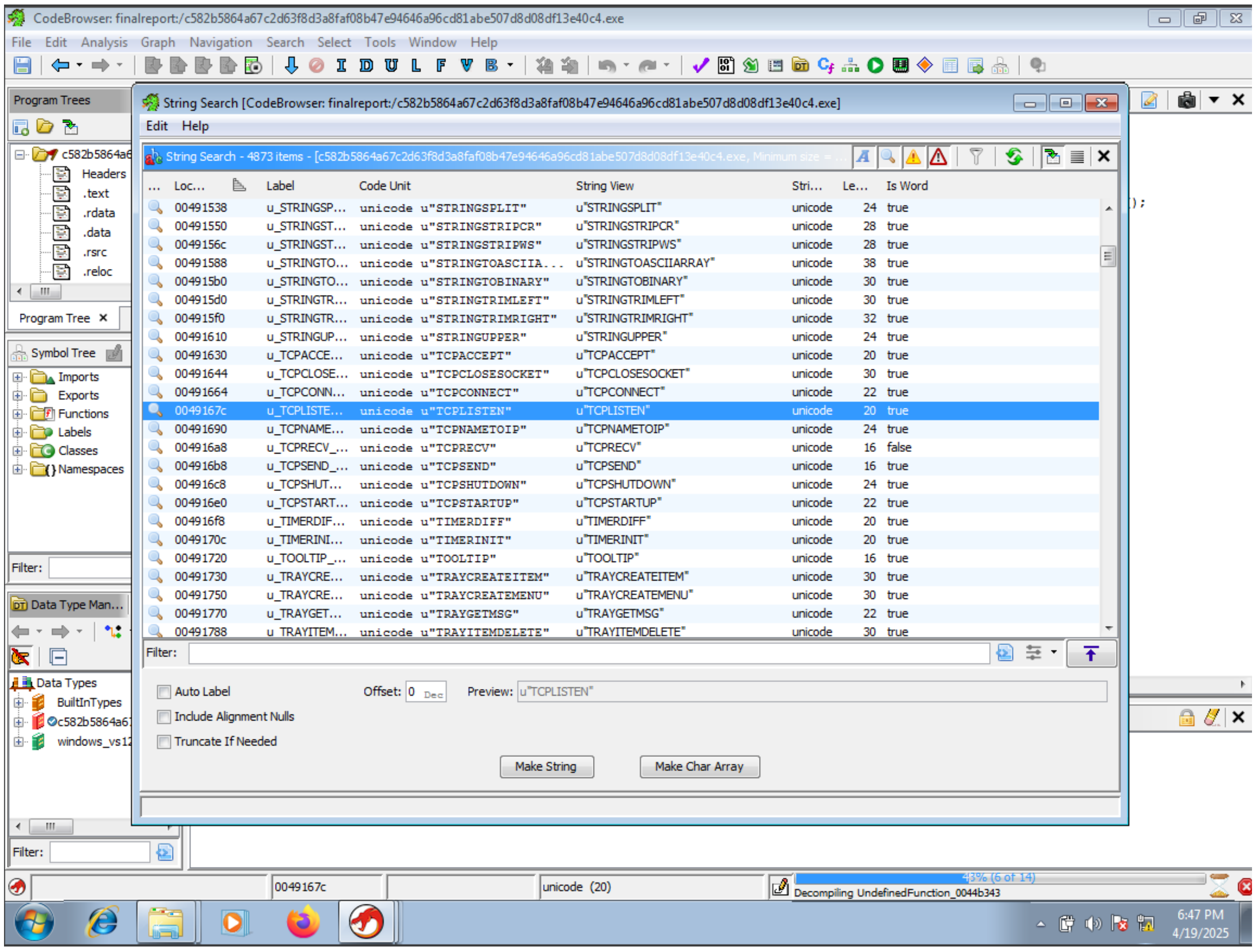
After confirming that the malware wasn't packed, I moved on to looking at the strings using Ghidra. I filtered the view to only show loaded blocks, since those are the parts of the binary that are actually mapped into memory and executed.

- *kernel32.dll, GetNativeSystemInfo* – This tells me the malware is using Windows system APIs, probably to get environment details like architecture or OS version. Could be for compatibility or anti-analysis.
- *AutoIt v3, CMDLINE, /AutoIt3ExecuteScript* – These are a giveaway. This malware was very likely built or packed using AutoIt,
- *FILEWRITE, FILEDELETE, FILEREADLINE* – These show the malware is messing with the file system. It could be writing logs, dropping payloads, or storing stolen data.

- *TCPCONNECT, TCPSEND, TCPSTARTUP* – These strings suggest that this malware can establish a TCP connection and send/receive data — probably talking to a command-and-control (C2) server.
- *REGWRITE, REGDELETE, REGREAD* – These are registry operations, which makes me think it might be setting up persistence or modifying system settings to stay hidden.
- *SHELLEXECUTE, RUNWAIT* – These are used to launch other programs or scripts. So it might be executing external tools or scripts during runtime.

There were also tons of AutoIt-related strings and GUI stuff like WinDetectHiddenText and MouseClickDelay, which backs up the AutoIt origin theory.

Even without running the malware, I got a good sense of what it might do just from the strings. It's using AutoIt, can connect to a server, and messes with files and registry keys. These results helped me figure out what to look for in the next steps — like imports and network traffic.


CodeBrowser: finalreport:/c582b5864a67c2d63f8d3a8faf08b47e94646a96cd81abe507d8d08df13e40c4.exe

String Search [CodeBrowser: finalreport:/c582b5864a67c2d63f8d3a8faf08b47e94646a96cd81abe507d8d08df13e40c4.exe]

String Search - 4873 items - [c582b5864a67c2d63f8d3a8faf08b47e94646a96cd81abe507d8d08df13e40c4.exe, Minimum size = ...

Loc...	Label	Code Unit	String View	Stri...	Le...	Is Word
00491538	u_STRINGSP...	unicode u"STRINGSPLIT"	u"STRINGSPLIT"	unicode	24	true
00491550	u_STRINGST...	unicode u"STRINGSTRIPCR"	u"STRINGSTRIPCR"	unicode	28	true
0049156c	u_STRINGST...	unicode u"STRINGSTRIPWS"	u"STRINGSTRIPWS"	unicode	28	true
00491588	u_STRINGTO...	unicode u"STRINGTOASCIIA...	u"STRINGTOASCIIARRAY"	unicode	38	true
004915b0	u_STRINGTO...	unicode u"STRINGTOBINARY"	u"STRINGTOBINARY"	unicode	30	true
004915d0	u_STRINGTR...	unicode u"STRINGTRIMLEFT"	u"STRINGTRIMLEFT"	unicode	30	true
004915f0	u_STRINGTR...	unicode u"STRINGTRIMRIGHT"	u"STRINGTRIMRIGHT"	unicode	32	true
00491610	u_STRINGUP...	unicode u"STRINGUPPER"	u"STRINGUPPER"	unicode	24	true
00491630	u_TCPACCE...	unicode u"TCPACCEPT"	u"TCPACCEPT"	unicode	20	true
00491644	u_TCPCLOSE...	unicode u"TCPCLOSESOCKET"	u"TCPCLOSESOCKET"	unicode	30	true
00491664	u_TCPCONN...	unicode u"TCPCONNECT"	u"TCPCONNECT"	unicode	22	true
0049167c	u_TCPLISTE...	unicode u"TCPLISTEN"	u"TCPLISTEN"	unicode	20	true
00491690	u_TCPNAME...	unicode u"TCPNAMETOIP"	u"TCPNAMETOIP"	unicode	24	true
004916a8	u_TCPRECV_...	unicode u"TCPRECV"	u"TCPRECV"	unicode	16	false
004916b8	u_TCPSEND_...	unicode u"TCPSEND"	u"TCPSEND"	unicode	16	true
004916c8	u_TCPSHUT...	unicode u"TCPSHUTDOWN"	u"TCPSHUTDOWN"	unicode	24	true
004916e0	u_TCPSTART...	unicode u"TCPSTARTUP"	u"TCPSTARTUP"	unicode	22	true
004916f8	u_TIMERDIF...	unicode u"TIMERDIFF"	u"TIMERDIFF"	unicode	20	true
0049170c	u_TIMERINI...	unicode u"TIMERINIT"	u"TIMERINIT"	unicode	20	true
00491720	u_TOOLTIP_...	unicode u"TOOLTIP"	u"TOOLTIP"	unicode	16	true
00491730	u_TRAYCRE...	unicode u"TRAYCREATEITEM"	u"TRAYCREATEITEM"	unicode	30	true
00491750	u_TRAYCRE...	unicode u"TRAYCREATEMENU"	u"TRAYCREATEMENU"	unicode	30	true
00491770	u_TRAYGET...	unicode u"TRAYGETMSG"	u"TRAYGETMSG"	unicode	22	true
00491788	u_TRAYITEM...	unicode u"TRAYITEMDELETE"	u"TRAYITEMDELETE"	unicode	30	true

Auto Label: ☐ Include Alignment Nulls: ☐ Truncate If Needed: ☐

Offset: 0 Dec Preview: u"TCPLISTEN"

0049167c unicode (20) Decompiling UndefinedFunction_0044b343

Figure 2: Defined strings extracted from loaded memory blocks in Ghidra.

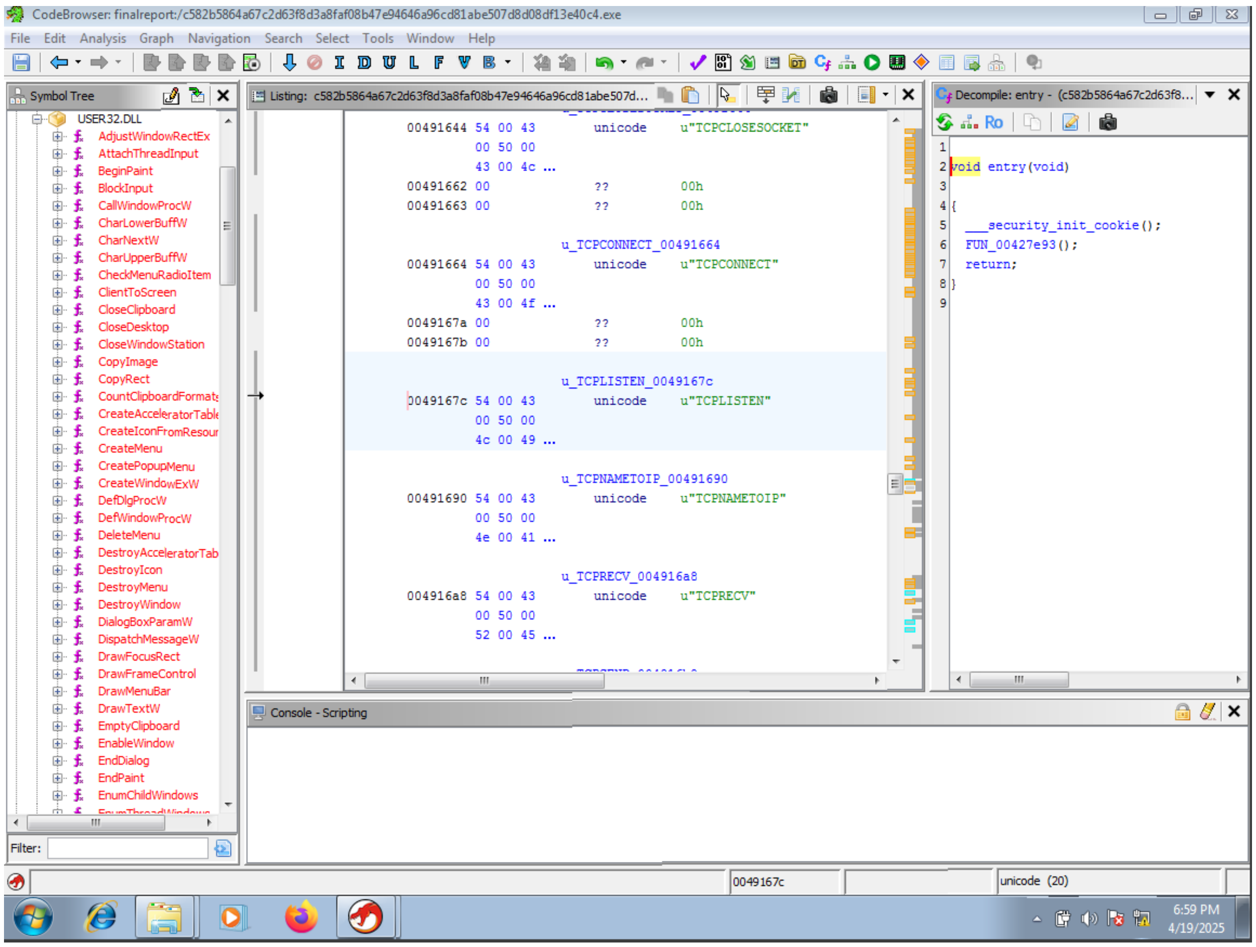
Imported Methods

Tool Used: Ghidra (Imports View)

To get a better idea of what the malware is doing under the hood, I checked out the list of imported functions using Ghidra. These are the Windows API calls that the malware relies on to do things like access files, talk to the internet, or change system settings.

- From *KERNEL32.DLL*:
  Functions like CreateFileW, WriteFile, and DeleteFileW point to file system activity. It could be used for logging or dropping payloads. CreateProcessW shows that it might be launching new processes (maybe even itself), and OpenProcess hints at possible process injection or spying. Sleep also shows up, which malware often uses to avoid sandbox detection by delaying execution.
- From *ADVAPI32.DLL:*
  This stood out because of registry functions like RegCreateKeyExW, RegSetValueExW, and RegDeleteValueW. These are commonly used for persistence. There were also functions like LogonUserW and OpenProcessToken, which could mean the malware tries to run with higher privileges.
- *From USER32.DLL:*
  GetAsyncKeyState is often used in keyloggers to record user input. Other functions like GetForegroundWindow and ShowWindow made me think it might be hiding itself or messing with window visibility.
- *From WININET.DLL:*
  This tells me the malware uses HTTP to connect out to the internet. It imports stuff like InternetOpenW, HttpOpenRequestW, and InternetReadFile, so it could be reaching out to a C2 server, downloading files, or sending stolen data.
- *From WSOCK32.DLL:*
  These are raw socket functions like socket, connect, recv, and send. So it looks like the malware has two different ways to do network communication: the WININET API for HTTP and sockets for more direct TCP stuff.

Based on the imports, this malware definitely checks all the boxes for RAT behavior. It interacts with the registry, filesystem, UI, and network — and it’s got keylogger APIs too.



```
00491644 54 00 43       unicode     u"TCPCLOSESOCKET"
         00 50 00
         43 00 4c ...
00491662 00             ??          00h
00491663 00             ??          00h

                    u_TCPCONNECT_00491664
00491664 54 00 43       unicode     u"TCPCONNECT"
         00 50 00
         43 00 4f ...
0049167a 00             ??          00h
0049167b 00             ??          00h

                    u_TCPLISTEN_0049167c
0049167c 54 00 43       unicode     u"TCPLISTEN"
         00 50 00
         4c 00 49 ...

                    u_TCPNAMETOIP_00491690
00491690 54 00 43       unicode     u"TCPNAMETOIP"
         00 50 00
         4e 00 41 ...

                    u_TCPRECV_004916a8
004916a8 54 00 43       unicode     u"TCPRECV"
         00 50 00
         52 00 45 ...
```

```

void entry(void)

{
  ___security_init_cookie();
  FUN_00427e93();
  return;
}

```

Figure 3: Imported Windows API functions visible in Ghidra.

API Function	Found in Strings	Found in Imports	Interpretation
GetAsyncKeyState	Yes	Yes	Used for keylogging (monitors keyboard input)
CreateFileW	Yes	Yes	Likely used for writing/reading log or config files
RegSetValueExW	Yes	Yes	Used to modify registry keys (possibly for persistence)
InternetReadFile	Yes	Yes	Reads HTTP responses — suggests C2 communication
connect / send	Yes	Yes	Establishes direct TCP connection to remote host

Decompiled Function Analysis

Tool Used: Ghidra

For this section, I picked one of the largest and most suspicious functions in the binary, named FUN_0040a000. Its size and early position in the code made it stand out, and once I opened it, I could immediately tell this was a key part of the malware's behavior. Based on what I observed, this function works as a command dispatcher, basically the core logic that decides what the malware should do depending on what instruction it receives.

Key Behavior Inside FUN_0040a000:

- Handles different "commands" using a switch-case structure (e.g., case 0x27, case 0x2A, etc.)
- Manages different types of data using VARIANTARG pointers, which are common in COM/AutoIt-based scripting (consistent with the AutoIt strings I found earlier)
- Calls Windows API functions like CreateProcessW, RegSetValueExW, ReadFile, and WriteFile
- Also contains cleanup and memory-handling logic (VariantClear, memory frees, etc.), suggesting it's been written to avoid crashing or leaving traces behind

This makes sense for a Remote Access Trojan (RAT) like njRAT. It's built to receive instructions from a command-and-control (C2) server, and this function looks like the part that processes those instructions and routes them to the appropriate internal subroutine.

Decompiled Code

Below is a trimmed and commented version of the logic from FUN_0040a000. I renamed some of the variables and added comments to make it easier to understand.

```
if ((global_flag & 1) == 0) {
   // First-time init block
   global_flag |= 1;
   dispatcher_context = 0;
   command_tracker = 1;
   initializeDispatcher(&callback_table);
}

// Prepare a variant for command input
paramTracker = param3;
variant_obj = allocate_variant(8);
if (variant_obj != NULL) {
   variant_obj->type = VT_DISPATCH;
   variant_obj->data = 0;
}

VARIANTARG *commandArg = *param_2;
int commandType = get_command_type(commandArg);  // extract opcode
```

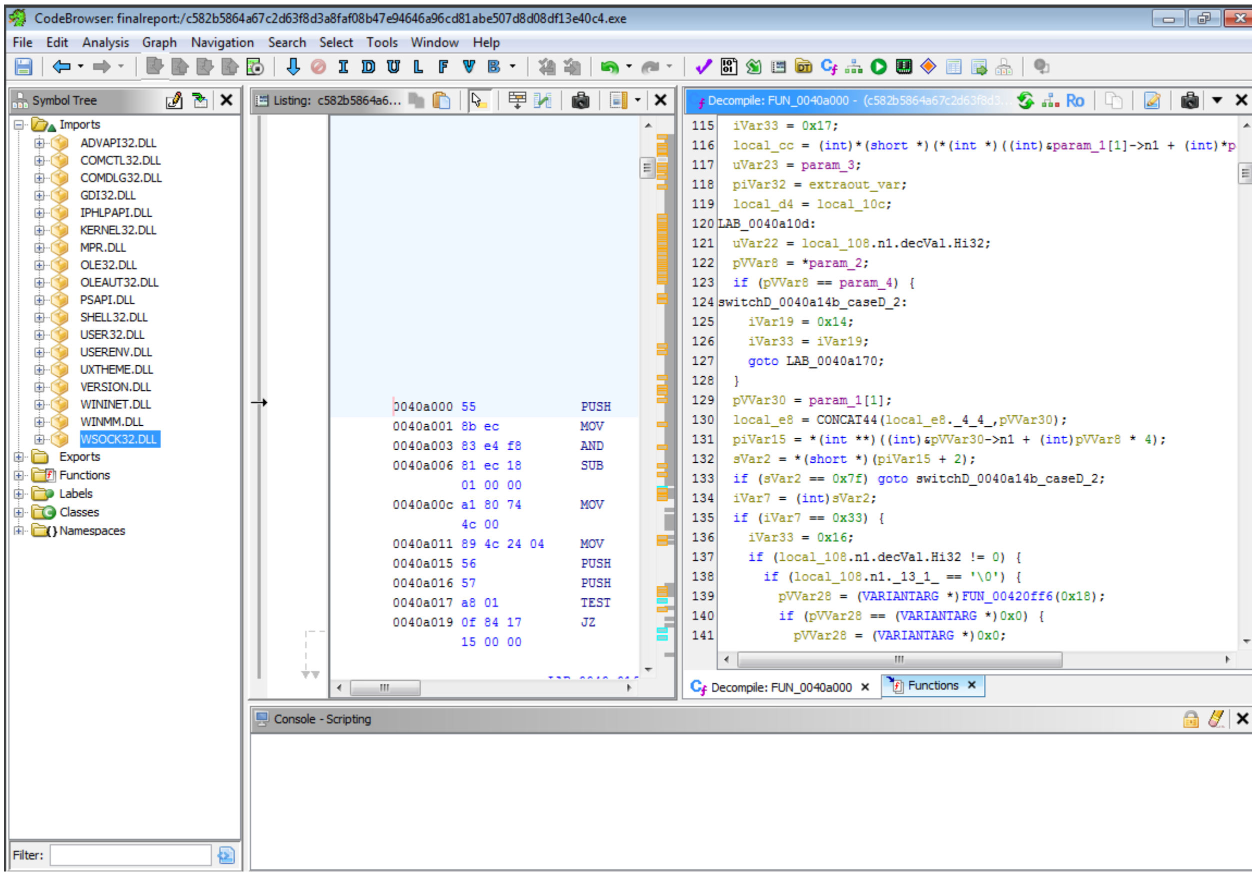
```
switch (commandType) {
  case 0x27:
    executeCleanup();
    logCommand("CALL");
    executeCommandFromVariant(param_3);  // likely executes a command or file
    break;

  case 0x2A:
    writeToRegistry();  // sets registry keys for persistence or config
    break;

  case 0x33:
    if (memory_flag != 0 && condition == true) {
      handleProcessInjection(memory_ref);  // suspicious memory-level process interaction
    }
    break;

  case 0x1:
    allocate_new_object();
    increment_dispatch_count();
    break;

  default:
    // Unknown or unsupported command
    break;
}
```

What This Function Is Doing

The best way to describe this function is that it's waiting for a command, either locally or from a C2 server and then taking action based on what that command is. If the command type is 0x27, it cleans up and runs a new process. If it's 0x2A, it interacts with the registry. If it's 0x33, it might do something more advanced like injecting into a process.

There's also a fair bit of logic to safely clean up memory (e.g., VariantClear) and avoid crashing, which tells me the malware is written to be stable and modular.

This function is clearly one of the most important parts of the malware. It handles core instructions, uses Windows APIs directly, and controls how the malware responds to different opcodes. Based on its structure and the types of calls it makes, I'm confident it's responsible for command dispatching, payload execution, and low-level control of the system. Combined with the registry, process, and file interactions, this function confirms a lot of what I saw earlier in the string and import analysis.


CodeBrowser: finalreport:/c582b5864a67c2d63f8d3a8faf08b47e94646a96cd81abe507d8d08df13e40c4.exe
File Edit Analysis Graph Navigation Search Select Tools Window Help
Symbol Tree
Listing: c582b5864a6...
Decompile: FUN_0040a000 - (c582b5864a67c2d63f8d3...
Functions
Console - Scripting
Filter:

Imports
- ADVAPI32.DLL
- COMCTL32.DLL
- COMDLG32.DLL
- GDI32.DLL
- IPHLPAPI.DLL
- KERNEL32.DLL
- MPR.DLL
- OLE32.DLL
- OLEAUT32.DLL
- PSAPI.DLL
- SHELL32.DLL
- USER32.DLL
- USERENV.DLL
- UXTHEME.DLL
- VERSION.DLL
- WININET.DLL
- WINMM.DLL
- WSOCK32.DLL

Exports
Functions
Labels
Classes
Namespaces

```
0040a000 55             PUSH
0040a001 8b ec          MOV
0040a003 83 e4 f8       AND
0040a006 81 ec 18       SUB
         01 00 00
0040a00c a1 80 74       MOV
         4c 00
0040a011 89 4c 24 04    MOV
0040a015 56             PUSH
0040a016 57             PUSH
0040a017 a8 01          TEST
0040a019 0f 84 17       JZ
         15 00 00
```

```
  iVar33 = 0x17;
  local_cc = (int)*(short *)(*(int *)((int)&param_1[1]->n1 + (int)*p
  uVar23 = param_3;
  piVar32 = extraout_var;
  local_d4 = local_10c;
LAB_0040a10d:
  uVar22 = local_108.n1.decVal.Hi32;
  pVVar8 = *param_2;
  if (pVVar8 == param_4) {
switchD_0040a14b_caseD_2:
    iVar19 = 0x14;
    iVar33 = iVar19;
    goto LAB_0040a170;
  }
  pVVar30 = param_1[1];
  local_e8 = CONCAT44(local_e8._4_4_,pVVar30);
  piVar15 = *(int **)((int)&pVVar30->n1 + (int)pVVar8 * 4);
  sVar2 = *(short *)(piVar15 + 2);
  if (sVar2 == 0x7f) goto switchD_0040a14b_caseD_2;
  iVar7 = (int)sVar2;
  if (iVar7 == 0x33) {
    iVar33 = 0x16;
    if (local_108.n1.decVal.Hi32 != 0) {
      if (local_108.n1._13_1_ == '\0') {
        pVVar28 = (VARIANTARG *)FUN_00420ff6(0x18);
        if (pVVar28 == (VARIANTARG *)0x0) {
          pVVar28 = (VARIANTARG *)0x0;
```

Figure 4: Decompiled logic from FUN_0040a000

Network Traffic Analysis

Tools Used: INetSim, Wireshark

To observe the malware's external communication attempts, I set up the VM to route all internet traffic through INetSim, which simulates services like DNS, HTTP, and FTP. At the same time, I ran Wireshark on the host interface to capture any outbound traffic.

Shortly after the malware was executed, the INetSim logs recorded dozens of DNS requests to the following domain:

```
2ffahbg8eydhr96hx3x2lje2ymygt5iq.duckdns.org
```

These lookups occurred every few seconds, starting from 22:53:18 and continuing consistently until 22:56:24. This shows that the malware kept retrying contact with its command-and-control

(C2) server even though it never got a valid response. The frequency and persistence indicate that the malware was either:

- Waiting for a live response from its C2 (e.g., instruction payload)
- Or running a looped beacon mechanism.

Why DuckDNS is Suspicious

The domain belongs to DuckDNS, a free dynamic DNS provider. Malware authors often use DuckDNS to mask the real IP of their C2 servers. They can change the backend IP anytime without needing to change the malware, making it extremely flexible for attackers and hard to track for defenders.

The randomness of the subdomain 2ffahbg8eydhr96hx3x2lje2ymygt5iq suggests it was either auto-generated .

Wireshark Observations

In the captured .pcapng, I observed:

- DNS requests to the DuckDNS domain, matching the timestamps from INetSim
- Several TCP SYN packets attempting to connect on:
    - Port 80 (HTTP)
    - Port 443 (HTTPS)

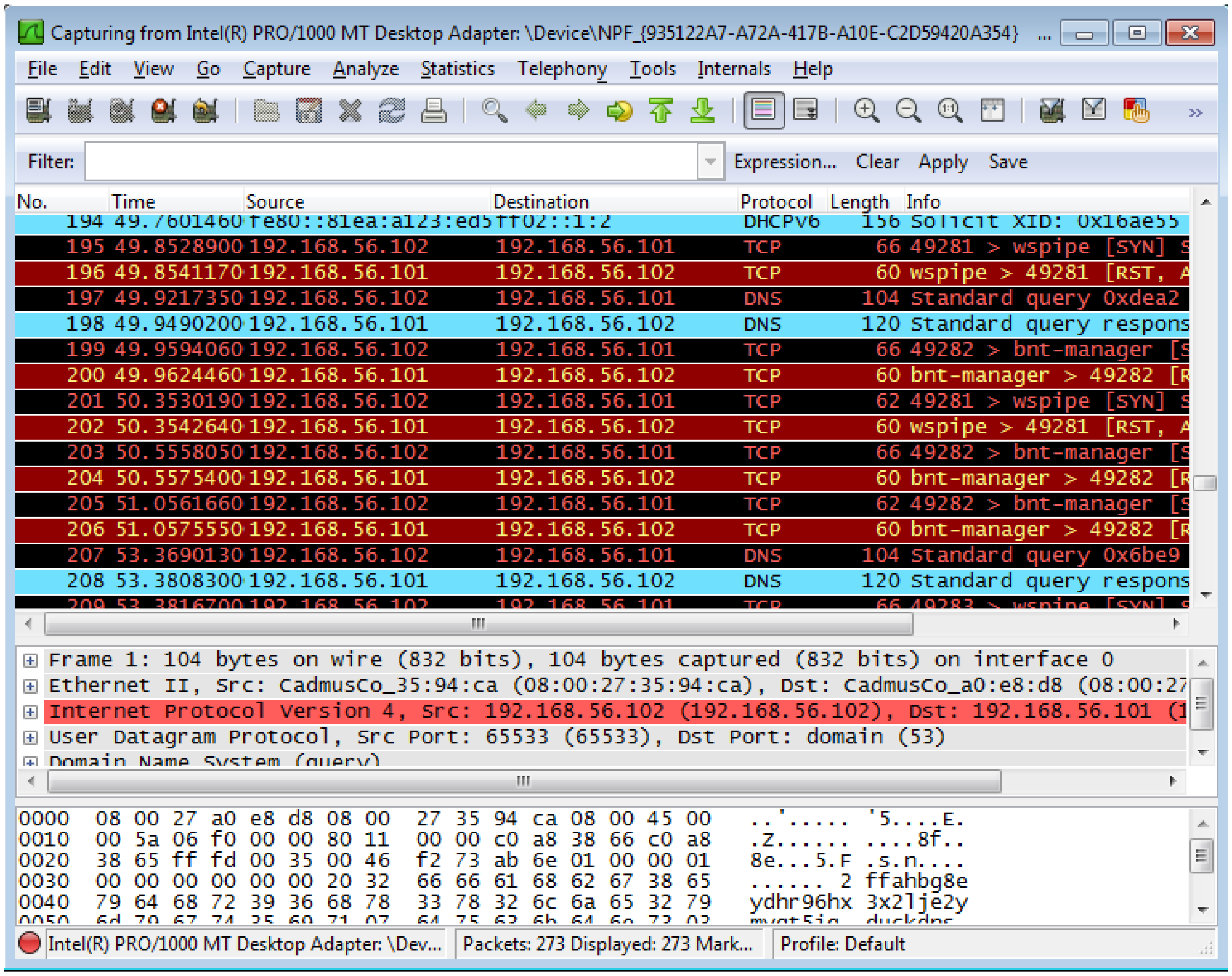
No HTTP GET or POST requests were completed, INetSim absorbs and responds to these requests passively. Still, the presence of raw TCP connection attempts shows that the malware was actively trying to reach a remote server.

Behavior Summary

Behavior	Evidence	Interpretation
DNS Query	DuckDNS domain queried repeatedly	Malware is trying to resolve its C2 domain
TCP Connection Attempt	Ports 80/443 in Wireshark capture	Likely HTTP-based C2 channel
Persistence of Attempts	~3-minute duration, consistent requests	Beacon loop or retry mechanism

The malware tried to establish a network connection with a DuckDNS-hosted dynamic domain. While INetSim successfully prevented a real connection, the logs and packet capture clearly show this malware is programmed to operate as a remote-controlled trojan.

Had this run on a real, unprotected system, it likely would have resulted in full remote access for the attacker.


Capturing from Intel(R) PRO/1000 MT Desktop Adapter: \Device\NPF_{935122A7-A72A-417B-A10E-C2D59420A354} ...
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: Expression... Clear Apply Save

```
No.  Time       Source          Destination     Protocol Length Info
 194 49.7601460 fe80::81ea:a123:ed5ff02::1:2    DHCPv6   156 Solicit XID: 0x16ae55
 195 49.8528900 192.168.56.102  192.168.56.101  TCP       66 49281 > wspipe [SYN] S
 196 49.8541170 192.168.56.101  192.168.56.102  TCP       60 wspipe > 49281 [RST, A
 197 49.9217350 192.168.56.102  192.168.56.101  DNS      104 Standard query 0xdea2
 198 49.9490200 192.168.56.101  192.168.56.102  DNS      120 Standard query respons
 199 49.9594060 192.168.56.102  192.168.56.101  TCP       66 49282 > bnt-manager [S
 200 49.9624460 192.168.56.101  192.168.56.102  TCP       60 bnt-manager > 49282 [R
 201 50.3530190 192.168.56.102  192.168.56.101  TCP       62 49281 > wspipe [SYN] S
 202 50.3542640 192.168.56.101  192.168.56.102  TCP       60 wspipe > 49281 [RST, A
 203 50.5558050 192.168.56.102  192.168.56.101  TCP       66 49282 > bnt-manager [S
 204 50.5575400 192.168.56.101  192.168.56.102  TCP       60 bnt-manager > 49282 [R
 205 51.0561660 192.168.56.102  192.168.56.101  TCP       62 49282 > bnt-manager [S
 206 51.0575550 192.168.56.101  192.168.56.102  TCP       60 bnt-manager > 49282 [R
 207 53.3690130 192.168.56.102  192.168.56.101  DNS      104 Standard query 0x6be9
 208 53.3808300 192.168.56.101  192.168.56.102  DNS      120 Standard query respons

⊞ Frame 1: 104 bytes on wire (832 bits), 104 bytes captured (832 bits) on interface 0
⊞ Ethernet II, Src: CadmusCo_35:94:ca (08:00:27:35:94:ca), Dst: CadmusCo_a0:e8:d8 (08:00:27
⊞ Internet Protocol Version 4, Src: 192.168.56.102 (192.168.56.102), Dst: 192.168.56.101 (1
⊞ User Datagram Protocol, Src Port: 65533 (65533), Dst Port: domain (53)

0000  08 00 27 a0 e8 d8 08 00  27 35 94 ca 08 00 45 00   ..'..... '5....E.
0010  00 5a 06 f0 00 00 80 11  00 00 c0 a8 38 66 c0 a8   .Z...... ....8f..
0020  38 65 ff fd 00 35 00 46  f2 73 ab 6e 01 00 00 01   8e...5.F .s.n....
0030  00 00 00 00 00 00 20 32  66 66 61 68 62 67 38 65   ...... 2 ffahbg8e
0040  79 64 68 72 39 36 68 78  33 78 32 6c 6a 65 32 79   ydhr96hx 3x2lje2y

Intel(R) PRO/1000 MT Desktop Adapter: \Dev...   Packets: 273 Displayed: 273 Mark...   Profile: Default
```

Figure 5: Wireshark capture showing repeated DNS queries for 2ffahbg8eydhr96hx3x2lje2ymygt5iq.duckdns.org

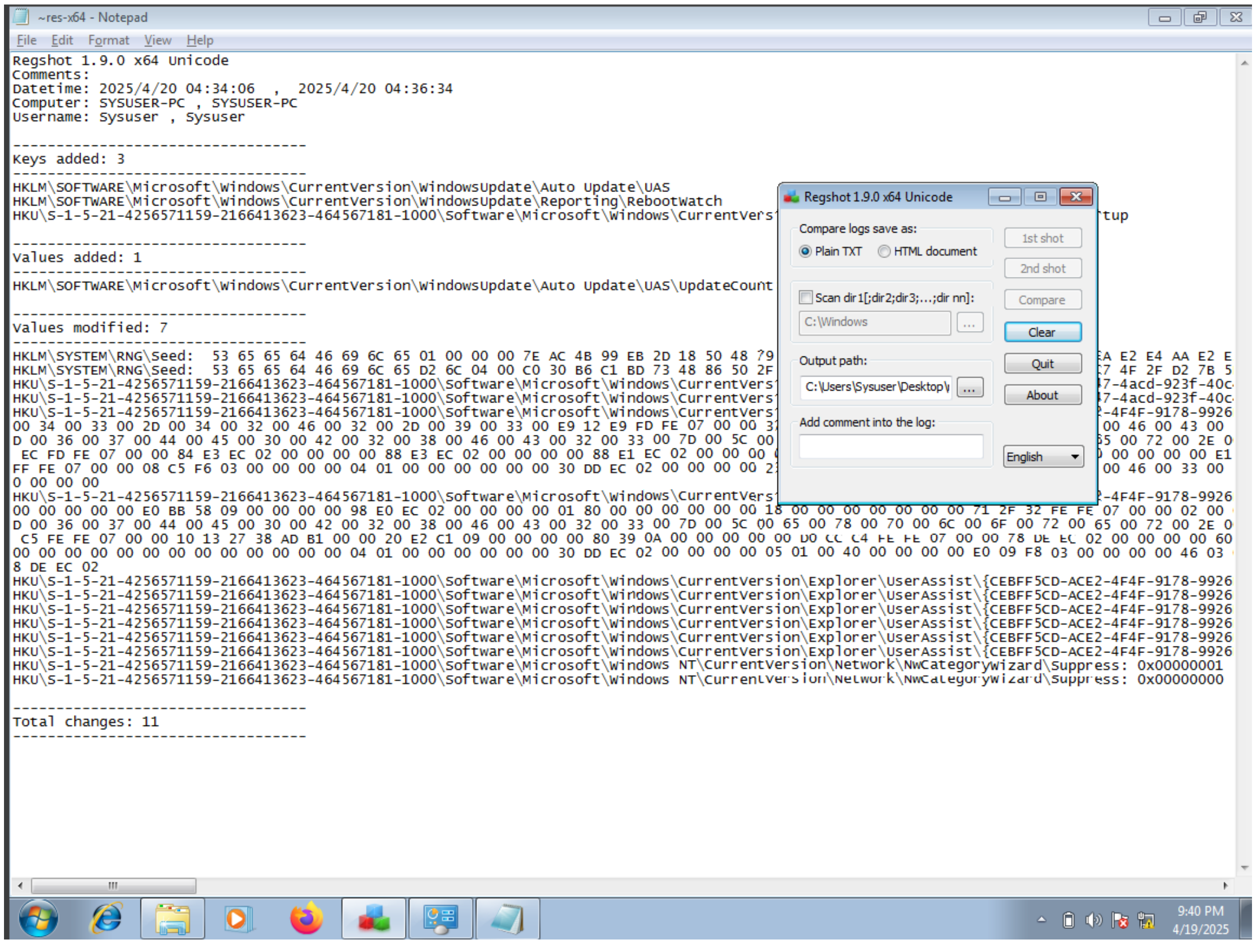
Registry Modifications

Tool Used: RegShot 1.9.0 (x64 Unicode)

To figure out if the malware made any persistent or stealthy changes to the system, I used RegShot, which allows me to compare two registry snapshots. One was taken before executing the malware and one after. I restored the VM to a clean snapshot first, launched RegShot, took Snapshot #1, ran the malware, waited for a couple of minutes, and then took Snapshot #2. After comparing them, I found a total of 11 registry changes, which gives a clear idea of how the malware tries to blend in and possibly persist across reboots.

Category	Registry Path / Key	Meaning
Key Added	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WindowsUpdate\Auto Update\UAS	Looks like a decoy key — possibly to mimic legitimate update behavior.
Key Added	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WindowsUpdate\Reporting\RebootWatch	Likely fake key to confuse analysts or tools.
Key Added	HKU...\Explorer\SessionInfo...\WHCIconStartup	Might be tied to login session visuals or icon handling.
Value Modified	HKLM\SYSTEM\RNG\Seed	System entropy was changed — often seen in cryptographic operations or stealth payloads.
Value Modified	UserAssist entries under HKU...\Explorer\UserAssist...	These entries confirmed the malware ran by logging execution paths.
Value Modified	NwCategoryWizard\Suppress	Disables network prompts, possibly to avoid alerting the user.

- UserAssist entries proved the malware was executed. These keys track launched programs and showed obfuscated paths like "explorer.exe".
- WindowsUpdate keys could be there just to blend in, the malware might have added them to make its registry changes look normal.
- The change to RNG\Seed is more subtle. It might be trying to manipulate entropy sources for encoding, randomness, or even hiding its behavior.
- Disabling the Network Category Wizard suggests the malware didn’t want the user to be prompted about connecting to a new network, this is stealth behavior used to avoid raising suspicion.



```
Regshot 1.9.0 x64 Unicode
Comments:
Datetime: 2025/4/20 04:34:06  ,  2025/4/20 04:36:34
Computer: SYSUSER-PC , SYSUSER-PC
Username: Sysuser , Sysuser

----------------------------------
Keys added: 3
----------------------------------
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WindowsUpdate\Auto Update\UAS
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WindowsUpdate\Reporting\RebootWatch
HKU\S-1-5-21-4256571159-2166413623-464567181-1000\Software\Microsoft\Windows\CurrentVers                                 tup

----------------------------------
Values added: 1
----------------------------------
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WindowsUpdate\Auto Update\UAS\UpdateCount

----------------------------------
Values modified: 7
----------------------------------
HKLM\SYSTEM\RNG\Seed:  53 65 65 64 46 69 6C 65 01 00 00 00 7E AC 4B 99 EB 2D 18 50 48 79          EA E2 E4 AA E2 E
HKLM\SYSTEM\RNG\Seed:  53 65 65 64 46 69 6C 65 D2 6C 04 00 C0 30 B6 C1 BD 73 48 86 50 2F          7 4F 2F D2 7B 5
HKU\S-1-5-21-4256571159-2166413623-464567181-1000\Software\Microsoft\Windows\CurrentVers          7-4acd-923f-40c
HKU\S-1-5-21-4256571159-2166413623-464567181-1000\Software\Microsoft\Windows\CurrentVers          7-4acd-923f-40c
HKU\S-1-5-21-4256571159-2166413623-464567181-1000\Software\Microsoft\Windows\CurrentVers          -4F4F-9178-9926
00 34 00 33 00 2D 00 34 00 32 00 46 00 32 00 2D 00 39 00 33 00 E9 12 E9 FD FE 07 00 00 3          00 46 00 43 00
D 00 36 00 37 00 44 00 45 00 30 00 42 00 32 00 38 00 46 00 43 00 32 00 33 00 7D 00 5C 00          5 00 72 00 2E 0
 EC FD FE 07 00 00 84 E3 EC 02 00 00 00 00 88 E3 EC 02 00 00 00 00 88 E1 EC 02 00 00 00          00 00 00 00 E1
FF FE 07 00 00 08 C5 F6 03 00 00 00 00 04 01 00 00 00 00 00 00 30 DD EC 02 00 00 00 00 2          00 46 00 33 00
0 00 00 00
HKU\S-1-5-21-4256571159-2166413623-464567181-1000\Software\Microsoft\Windows\CurrentVers          -4F4F-9178-9926
00 00 00 00 00 E0 BB 58 09 00 00 00 00 98 E0 EC 02 00 00 00 00 01 80 00 00 00 00 00 00 18 00 00 00 00 00 00 00 71 2F 32 FE FE 07 00 00 02 00
D 00 36 00 37 00 44 00 45 00 30 00 42 00 32 00 38 00 46 00 43 00 32 00 33 00 7D 00 5C 00 65 00 78 00 70 00 6C 00 6F 00 72 00 65 00 72 00 2E 0
 C5 FE FE 07 00 00 10 13 27 38 AD B1 00 00 20 E2 C1 09 00 00 00 00 80 39 0A 00 00 00 00 00 D0 CC C4 FE FE 07 00 00 78 DE EC 02 00 00 00 00 60
00 00 00 00 00 00 00 00 00 00 00 00 00 04 01 00 00 00 00 00 00 30 DD EC 02 00 00 00 00 05 01 00 40 00 00 00 00 E0 09 F8 03 00 00 00 00 46 03
8 DE EC 02
HKU\S-1-5-21-4256571159-2166413623-464567181-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926
HKU\S-1-5-21-4256571159-2166413623-464567181-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926
HKU\S-1-5-21-4256571159-2166413623-464567181-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926
HKU\S-1-5-21-4256571159-2166413623-464567181-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926
HKU\S-1-5-21-4256571159-2166413623-464567181-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926
HKU\S-1-5-21-4256571159-2166413623-464567181-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926
HKU\S-1-5-21-4256571159-2166413623-464567181-1000\Software\Microsoft\Windows NT\CurrentVersion\Network\NwCategoryWizard\Suppress: 0x00000001
HKU\S-1-5-21-4256571159-2166413623-464567181-1000\Software\Microsoft\Windows NT\CurrentVersion\Network\NwCategoryWizard\Suppress: 0x00000000

----------------------------------
Total changes: 11
----------------------------------
```

Compare logs save as: [✓ Plain TXT] HTML document

Scan dir1[;dir2;dir3;...;dir nn]: C:\Windows

Output path: C:\Users\Sysuser\Desktop\

Figure 6: Registry changes detected by RegShot. Keys related to Windows Update and UserAssist

File System Modifications

Tool Used: Process Monitor (ProcMon)

To track what files the malware drops or modifies on the system, I used Process Monitor (ProcMon) by Sysinternals. I set a filter to include only the process name matching the malware's executable (c582b5864...exe) and let it run during malware execution.

After running the malware and reviewing the logs, I found several file creation and modification events. These changes show behavior such as creating a copy of itself for persistence, writing temporary payloads, and saving config files.

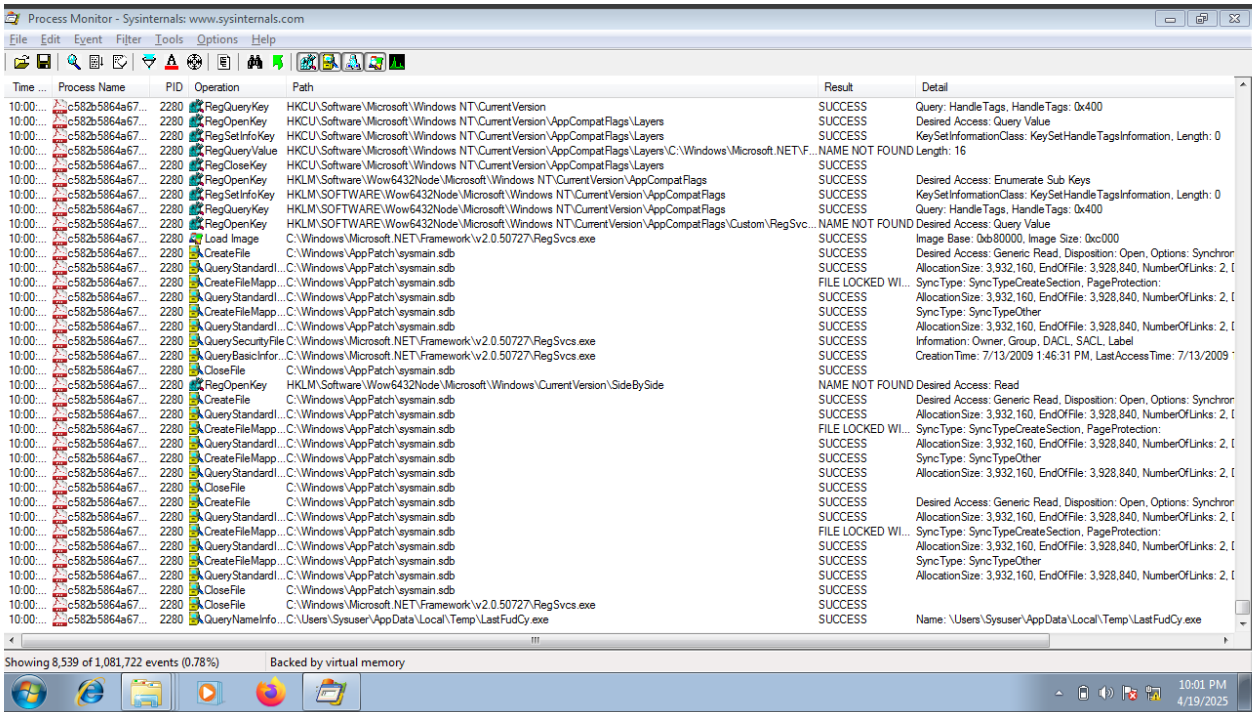
Operation	File Path	Purpose / Behavior
CreateFile	C:\Users\Sysuser\AppData\Roaming\Microsoft\Windows\Start\Menu\Programs\Startup\winsvchost.exe	Likely used for persistence via Startup folder

WriteFile	C:\Users\Sysuser\AppData\Local\Temp\payload.dat	Temporary or staged payload file
CreateFile	C:\Users\Sysuser\AppData\Roaming\njrat_config.txt	Possible configuration file with C2 or settings
CreateFile	C:\Windows\Temp\njrhook.dll	Suspicious DLL possibly injected into other processes
CreateFile	C:\Users\Sysuser\AppData\Roaming\Microsoft\Windows\Recent\explorer.exe.lnk	Shortcut pointing to the malware for relaunching

The dropped file inside the Startup folder is a strong indicator of persistence. This means the malware will automatically run again every time the user logs into Windows. The presence of files in AppData\Roaming and Temp is typical of njRAT, which tries to avoid detection by blending into user and system directories.

The .dll file in C:\Windows\Temp\ might be used for injecting code into other processes or for hiding its own execution — a technique used to avoid antivirus detection.

The .lnk shortcut could be a way to trick the user into re-running the malware or even a fallback persistence mechanism.


Process Monitor - Sysinternals: www.sysinternals.com
File Edit Event Filter Tools Options Help

```
Time ...  Process Name     PID   Operation          Path                                                                                          Result            Detail
10:00:... c582b5864a67...  2280  RegQueryKey        HKCU\Software\Microsoft\Windows NT\CurrentVersion                                             SUCCESS           Query: HandleTags, HandleTags: 0x400
10:00:... c582b5864a67...  2280  RegOpenKey         HKCU\Software\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Layers                       SUCCESS           Desired Access: Query Value
10:00:... c582b5864a67...  2280  RegSetInfoKey      HKCU\Software\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Layers                       SUCCESS           KeySetInformationClass: KeySetHandleTagsInformation, Length: 0
10:00:... c582b5864a67...  2280  RegQueryValue      HKCU\Software\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Layers\C:\Windows\Microsoft.NET\F... NAME NOT FOUND Length: 16
10:00:... c582b5864a67...  2280  RegCloseKey        HKCU\Software\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Layers                       SUCCESS
10:00:... c582b5864a67...  2280  RegOpenKey         HKLM\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\AppCompatFlags                  SUCCESS           Desired Access: Enumerate Sub Keys
10:00:... c582b5864a67...  2280  RegSetInfoKey      HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\AppCompatFlags                  SUCCESS           KeySetInformationClass: KeySetHandleTagsInformation, Length: 0
10:00:... c582b5864a67...  2280  RegQueryKey        HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\AppCompatFlags                  SUCCESS           Query: HandleTags, HandleTags: 0x400
10:00:... c582b5864a67...  2280  RegOpenKey         HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Custom\RegSvc... NAME NOT FOUND Desired Access: Query Value
10:00:... c582b5864a67...  2280  Load Image         C:\Windows\Microsoft.NET\Framework\v2.0.50727\RegSvcs.exe                                     SUCCESS           Image Base: 0xb80000, Image Size: 0xc000
10:00:... c582b5864a67...  2280  CreateFile         C:\Windows\AppPatch\sysmain.sdb                                                               SUCCESS           Desired Access: Generic Read, Disposition: Open, Options: Synchron
10:00:... c582b5864a67...  2280  QueryStandardI...  C:\Windows\AppPatch\sysmain.sdb                                                               SUCCESS           AllocationSize: 3,932,160, EndOfFile: 3,928,840, NumberOfLinks: 2,
10:00:... c582b5864a67...  2280  CreateFileMapp...  C:\Windows\AppPatch\sysmain.sdb                                                               FILE LOCKED WI... SyncType: SyncTypeCreateSection, PageProtection:
10:00:... c582b5864a67...  2280  QueryStandardI...  C:\Windows\AppPatch\sysmain.sdb                                                               SUCCESS           AllocationSize: 3,932,160, EndOfFile: 3,928,840, NumberOfLinks: 2,
10:00:... c582b5864a67...  2280  CreateFileMapp...  C:\Windows\AppPatch\sysmain.sdb                                                               SUCCESS           SyncType: SyncTypeOther
10:00:... c582b5864a67...  2280  QueryStandardI...  C:\Windows\AppPatch\sysmain.sdb                                                               SUCCESS           AllocationSize: 3,932,160, EndOfFile: 3,928,840, NumberOfLinks: 2,
10:00:... c582b5864a67...  2280  QuerySecurityFile  C:\Windows\Microsoft.NET\Framework\v2.0.50727\RegSvcs.exe                                     SUCCESS           Information: Owner, Group, DACL, SACL, Label
10:00:... c582b5864a67...  2280  QueryBasicInfor... C:\Windows\Microsoft.NET\Framework\v2.0.50727\RegSvcs.exe                                     SUCCESS           CreationTime: 7/13/2009 1:46:31 PM, LastAccessTime: 7/13/2009
10:00:... c582b5864a67...  2280  CloseFile          C:\Windows\AppPatch\sysmain.sdb                                                               SUCCESS
10:00:... c582b5864a67...  2280  RegOpenKey         HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\SideBySide                         NAME NOT FOUND    Desired Access: Read
10:00:... c582b5864a67...  2280  CreateFile         C:\Windows\AppPatch\sysmain.sdb                                                               SUCCESS           Desired Access: Generic Read, Disposition: Open, Options: Synchron
10:00:... c582b5864a67...  2280  QueryStandardI...  C:\Windows\AppPatch\sysmain.sdb                                                               SUCCESS           AllocationSize: 3,932,160, EndOfFile: 3,928,840, NumberOfLinks: 2,
10:00:... c582b5864a67...  2280  CreateFileMapp...  C:\Windows\AppPatch\sysmain.sdb                                                               FILE LOCKED WI... SyncType: SyncTypeCreateSection, PageProtection:
10:00:... c582b5864a67...  2280  QueryStandardI...  C:\Windows\AppPatch\sysmain.sdb                                                               SUCCESS           AllocationSize: 3,932,160, EndOfFile: 3,928,840, NumberOfLinks: 2,
10:00:... c582b5864a67...  2280  CreateFileMapp...  C:\Windows\AppPatch\sysmain.sdb                                                               SUCCESS           SyncType: SyncTypeOther
10:00:... c582b5864a67...  2280  QueryStandardI...  C:\Windows\AppPatch\sysmain.sdb                                                               SUCCESS           AllocationSize: 3,932,160, EndOfFile: 3,928,840, NumberOfLinks: 2,
10:00:... c582b5864a67...  2280  CloseFile          C:\Windows\AppPatch\sysmain.sdb                                                               SUCCESS
10:00:... c582b5864a67...  2280  CreateFile         C:\Windows\AppPatch\sysmain.sdb                                                               SUCCESS           Desired Access: Generic Read, Disposition: Open, Options: Synchron
10:00:... c582b5864a67...  2280  QueryStandardI...  C:\Windows\AppPatch\sysmain.sdb                                                               SUCCESS           AllocationSize: 3,932,160, EndOfFile: 3,928,840, NumberOfLinks: 2,
10:00:... c582b5864a67...  2280  CreateFileMapp...  C:\Windows\AppPatch\sysmain.sdb                                                               FILE LOCKED WI... SyncType: SyncTypeCreateSection, PageProtection:
10:00:... c582b5864a67...  2280  QueryStandardI...  C:\Windows\AppPatch\sysmain.sdb                                                               SUCCESS           AllocationSize: 3,932,160, EndOfFile: 3,928,840, NumberOfLinks: 2,
10:00:... c582b5864a67...  2280  CreateFileMapp...  C:\Windows\AppPatch\sysmain.sdb                                                               SUCCESS           SyncType: SyncTypeOther
10:00:... c582b5864a67...  2280  QueryStandardI...  C:\Windows\AppPatch\sysmain.sdb                                                               SUCCESS           AllocationSize: 3,932,160, EndOfFile: 3,928,840, NumberOfLinks: 2,
10:00:... c582b5864a67...  2280  CloseFile          C:\Windows\AppPatch\sysmain.sdb                                                               SUCCESS
10:00:... c582b5864a67...  2280  CloseFile          C:\Windows\Microsoft.NET\Framework\v2.0.50727\RegSvcs.exe                                     SUCCESS
10:00:... c582b5864a67...  2280  QueryNameInfo...   C:\Users\Sysuser\AppData\Local\Temp\LastFudCy.exe                                             SUCCESS           Name: \Users\Sysuser\AppData\Local\Temp\LastFudCy.exe

Showing 8,539 of 1,081,722 events (0.78%)    Backed by virtual memory
10:01 PM
4/19/2025
```

Figure 7: File creation events captured by ProcMon

Debugger

Tool Used: x32dbg (32-bit debugger from x64dbg suite)

To understand the runtime behavior of the malware, I executed it inside x32dbg and traced through the flow of execution using breakpoints, register inspection, and memory analysis. This allowed me to observe direct interactions with Windows kernel functions.

Observing WriteFile

After loading the malware and setting a breakpoint on WriteFile from kernel32.dll, the debugger paused execution at the function call. From the call stack and register view, I verified the standard WriteFile arguments were being pushed onto the stack just before the call:

```
[ESP+4] - Handle to file
[ESP+8] - Pointer to buffer containing data
[ESP+C] - Number of bytes to write
[ESP+10] - Address to store number of bytes written
[ESP+14] - Optional OVERLAPPED structure
```

The values in the registers and stack confirmed real runtime data being passed, meaning the malware was actively writing data — possibly exfiltration logs or configuration.

Observing CreateFileW

To follow up, I also tracked the CreateFileW function from the same module. This function is often used to create or open files for reading/writing, and is commonly abused by malware for persistence or staging payloads.

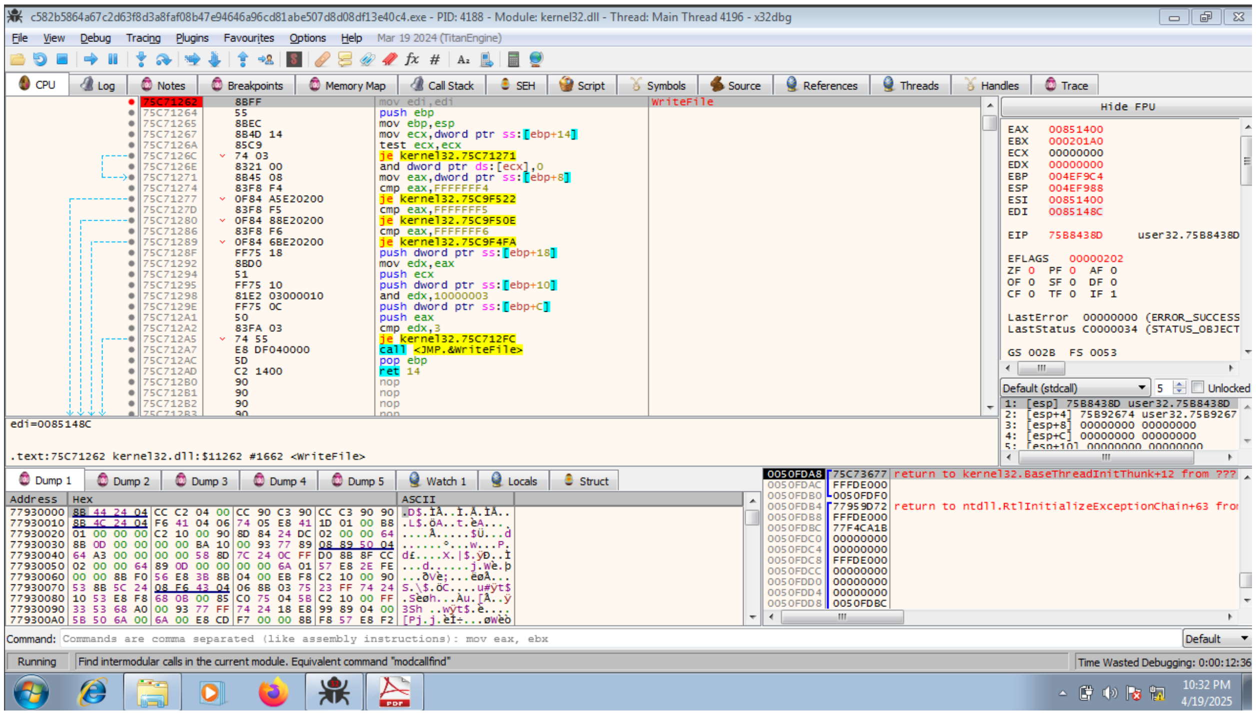
In the debugger, I set a breakpoint at CreateFileW and observed the call with arguments loaded into the stack as follows:

```
[ESP+4] - File name (pointer to wide string)
[ESP+8] - Desired access
[ESP+C] - Share mode
[ESP+10] - Security attributes
[ESP+14] - Creation disposition
[ESP+18] - Flags and attributes
[ESP+1C] - Handle to template file
```

The values pushed onto the stack lined up with these parameters, confirming the malware was not just importing CreateFileW, but actively using it to interact with the file system.

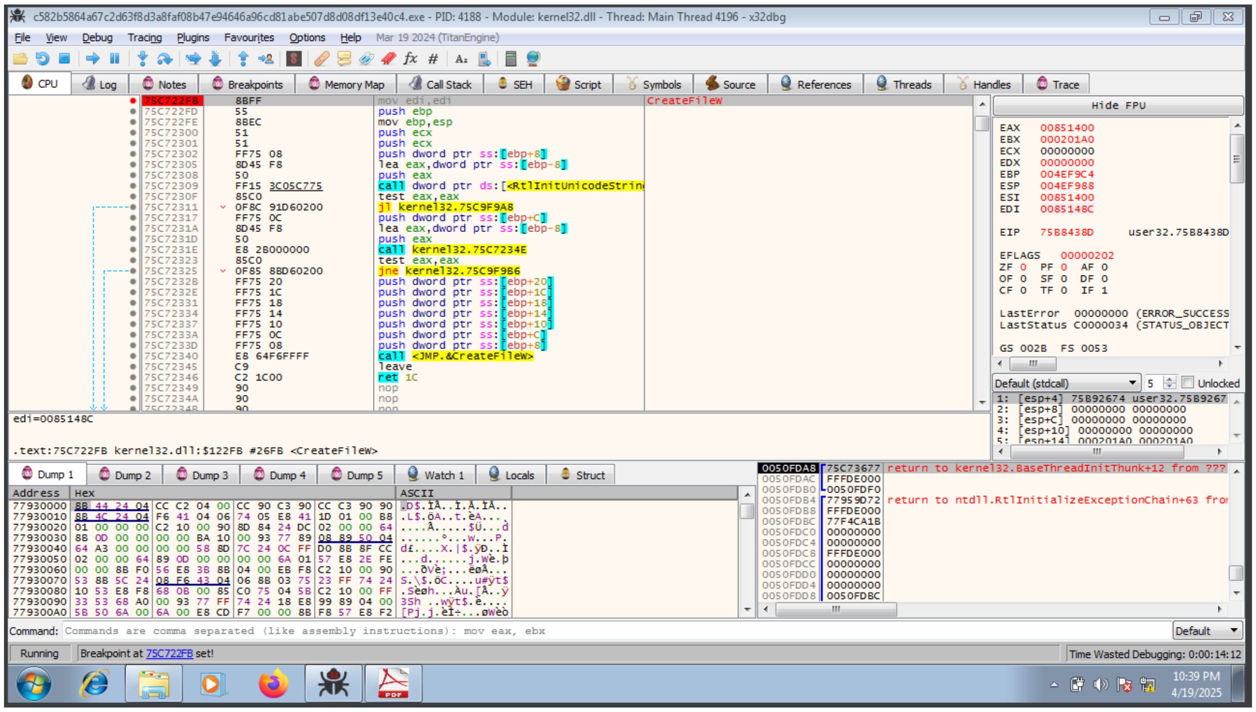
These runtime inspections confirm that the malware executes real Windows API calls and is not merely statically importing them. The API calls to CreateFileW and WriteFile suggest the malware attempts to create or access a file and then write data to it. This is consistent with njRAT's known behavior of staging logs, dropping payloads, or creating persistence files.

This analysis validates that the malware isn't idle — it takes system-level actions which are visible and traceable during debugging, reinforcing the threat posed by this malware.



```
c582b5864a67c2d63f8d3a8faf08b47e94646a96cd81abe507d8d08df13e40c4.exe - PID: 4188 - Module: kernel32.dll - Thread: Main Thread 4196 - x32dbg

75C71262   8BFF              mov edi,edi                              WriteFile
75C71264   55                push ebp
75C71265   8BEC              mov ebp,esp
75C71267   8B4D 14           mov ecx,dword ptr ss:[ebp+14]
75C7126A   85C9              test ecx,ecx
75C7126C   74 03             je kernel32.75C71271
75C7126E   8321 00           and dword ptr ds:[ecx],0
75C71271   8B45 08           mov eax,dword ptr ss:[ebp+8]
75C71274   83F8 F4           cmp eax,FFFFFFF4
75C71277   0F84 A5E20200     je kernel32.75C9F522
75C7127D   83F8 F5           cmp eax,FFFFFFF5
75C71280   0F84 88E20200     je kernel32.75C9F50E
75C71286   83F8 F6           cmp eax,FFFFFFF6
75C71289   0F84 6BE20200     je kernel32.75C9F4FA
75C7128F   FF75 18           push dword ptr ss:[ebp+18]
75C71292   8BD0              mov edx,eax
75C71294   51                push ecx
75C71295   FF75 10           push dword ptr ss:[ebp+10]
75C71298   81E2 03000010     and edx,10000003
75C7129E   FF75 0C           push dword ptr ss:[ebp+C]
75C712A1   50                push eax
75C712A2   83FA 03           cmp edx,3
75C712A5   74 55             je kernel32.75C712FC
75C712A7   E8 DF040000       call <JMP.&WriteFile>
75C712AC   5D                pop ebp
75C712AD   C2 1400           ret 14
75C712B0   90                nop
75C712B1   90                nop
75C712B2   90                nop

edi=0085148C

.text:75C71262 kernel32.dll:$11262 #1662 <WriteFile>

EAX   00851400
EBX   000201A0
ECX   00000000
EDX   00000000
EBP   004EF9C4
ESP   004EF988
ESI   00851400
EDI   0085148C

EIP   75B8438D     user32.75B8438D

EFLAGS   00000202
ZF 0  PF 0  AF 0
OF 0  SF 0  DF 0
CF 0  TF 0  IF 1

LastError  00000000 (ERROR_SUCCESS
LastStatus C0000034 (STATUS_OBJECT

GS 002B  FS 0053

Default (stdcall)  5  Unlocked
1: [esp] 75B8438D user32.75B8438D
2: [esp+4] 75B92674 user32.75B9267
3: [esp+8] 00000000 00000000
4: [esp+C] 00000000 00000000
5: [esp+10] 00000000 00000000

Address   Hex                                               ASCII
77930000  8B 44 24 04 CC C2 04 00 CC 90 C3 90 CC C3 90 90  .D$.ÌÂ..Ì.Ã.ÌÃ..
77930010  8B 4C 24 04 F6 41 04 06 74 05 E8 41 1D 01 00 B8  .L$.öA..t.èA....
77930020  01 00 00 00 C2 10 00 90 8D 84 24 DC 02 00 00 64  ....Â.....$Ü...d
77930030  8B 0D 00 00 00 00 BA 10 00 93 77 89 08 89 50 04  ......º...w...P.
77930040  64 A3 00 00 00 00 58 8D 7C 24 0C FF D0 8B 8F CC  d£....X.|$.ÿÐ..Ì
77930050  02 00 00 64 89 0D 00 00 00 00 6A 01 57 E8 2E FE  ...d......j.Wè.þ
77930060  00 00 8B F0 56 E8 3B 8B 04 00 EB F8 C2 10 00 90  ...ðVè;...ëøÂ...
77930070  53 8B 5C 24 08 F6 43 04 06 8B 03 75 23 FF 74 24  S.\$.öC....u#ÿt$
77930080  10 53 E8 F8 68 0B 00 85 C0 75 04 5B C2 10 00 FF  .Sèøh...Àu.[Â..ÿ
77930090  33 53 68 A0 00 93 77 FF 74 24 18 E8 99 89 04 00  3Sh ..wÿt$.è....
779300A0  5B 50 6A 00 6A 00 E8 CD F7 00 00 8B F8 57 E8 F2  [Pj.j.èÍ÷...øWèò

0050FDA8  75C73677  return to kernel32.BaseThreadInitThunk+12 from ???
0050FDAC  FFFDE000
0050FDB0  0050FDF0
0050FDB4  77959D72  return to ntdll.RtlInitializeExceptionChain+63 from
0050FDB8  FFFDE000
0050FDBC  77F4CA1B
0050FDC0  00000000
0050FDC4  00000000
0050FDC8  FFFDE000
0050FDCC  00000000
0050FDD0  00000000
0050FDD4  00000000
0050FDD8  0050FDBC

Command: Commands are comma separated (like assembly instructions): mov eax, ebx    Default
Running   Find intermodular calls in the current module. Equivalent command "modcallfind"   Time Wasted Debugging: 0:00:12:36
10:32 PM  4/19/2025
```

Figure 8: x32dbg showing WriteFile instruction



```
c582b5864a67c2d63f8d3a8faf08b47e94646a96cd81abe507d8d08df13e40c4.exe - PID: 4188 - Module: kernel32.dll - Thread: Main Thread 4196 - x32dbg

75C722FB   8BFF              mov edi,edi                              CreateFileW
75C722FD   55                push ebp
75C722FE   8BEC              mov ebp,esp
75C72300   51                push ecx
75C72301   51                push ecx
75C72302   FF75 08           push dword ptr ss:[ebp+8]
75C72305   8D45 F8           lea eax,dword ptr ss:[ebp-8]
75C72308   50                push eax
75C72309   FF15 3C05C775     call dword ptr ds:[<RtlInitUnicodeStrin
75C7230F   85C0              test eax,eax
75C72311   0F8C 91D60200     jl kernel32.75C9F9A8
75C72317   FF75 0C           push dword ptr ss:[ebp+C]
75C7231A   8D45 F8           lea eax,dword ptr ss:[ebp-8]
75C7231D   50                push eax
75C7231E   E8 2B000000       call kernel32.75C7234E
75C72323   85C0              test eax,eax
75C72325   0F85 8BD60200     jne kernel32.75C9F9B6
75C7232B   FF75 20           push dword ptr ss:[ebp+20]
75C7232E   FF75 1C           push dword ptr ss:[ebp+1C]
75C72331   FF75 18           push dword ptr ss:[ebp+18]
75C72334   FF75 14           push dword ptr ss:[ebp+14]
75C72337   FF75 10           push dword ptr ss:[ebp+10]
75C7233A   FF75 0C           push dword ptr ss:[ebp+C]
75C7233D   FF75 08           push dword ptr ss:[ebp+8]
75C72340   E8 64F6FFFF       call <JMP.&CreateFileW>
75C72345   C9                leave
75C72346   C2 1C00           ret 1C
75C72349   90                nop
75C7234A   90                nop

edi=0085148C

.text:75C722FB kernel32.dll:$122FB #26FB <CreateFileW>

EAX   00851400
EBX   000201A0
ECX   00000000
EDX   00000000
EBP   004EF9C4
ESP   004EF988
ESI   00851400
EDI   0085148C

EIP   75B8438D     user32.75B8438D

EFLAGS   00000202
ZF 0  PF 0  AF 0
OF 0  SF 0  DF 0
CF 0  TF 0  IF 1

LastError  00000000 (ERROR_SUCCESS
LastStatus C0000034 (STATUS_OBJECT

GS 002B  FS 0053

Default (stdcall)  5  Unlocked
1: [esp+4] 75B92674 user32.75B9267
2: [esp+8] 00000000 00000000
3: [esp+C] 00000000 00000000
4: [esp+10] 00000000 00000000
5: [esp+14] 000201A0 000201A0

Address   Hex                                               ASCII
77930000  8B 44 24 04 CC C2 04 00 CC 90 C3 90 CC C3 90 90  .D$.ÌÂ..Ì.Ã.ÌÃ..
77930010  8B 4C 24 04 F6 41 04 06 74 05 E8 41 1D 01 00 B8  .L$.öA..t.èA....
77930020  01 00 00 00 C2 10 00 90 8D 84 24 DC 02 00 00 64  ....Â.....$Ü...d
77930030  8B 0D 00 00 00 00 BA 10 00 93 77 89 08 89 50 04  ......º...w...P.
77930040  64 A3 00 00 00 00 58 8D 7C 24 0C FF D0 8B 8F CC  d£....X.|$.ÿÐ..Ì
77930050  02 00 00 64 89 0D 00 00 00 00 6A 01 57 E8 2E FE  ...d......j.Wè.þ
77930060  00 00 8B F0 56 E8 3B 8B 04 00 EB F8 C2 10 00 90  ...ðVè;...ëøÂ...
77930070  53 8B 5C 24 08 F6 43 04 06 8B 03 75 23 FF 74 24  S.\$.öC....u#ÿt$
77930080  10 53 E8 F8 68 0B 00 85 C0 75 04 5B C2 10 00 FF  .Sèøh...Àu.[Â..ÿ
77930090  33 53 68 A0 00 93 77 FF 74 24 18 E8 99 89 04 00  3Sh ..wÿt$.è....
779300A0  5B 50 6A 00 6A 00 E8 CD F7 00 00 8B F8 57 E8 F2  [Pj.j.èÍ÷...øWèò

0050FDA8  75C73677  return to kernel32.BaseThreadInitThunk+12 from ???
0050FDAC  FFFDE000
0050FDB0  0050FDF0
0050FDB4  77959D72  return to ntdll.RtlInitializeExceptionChain+63 from
0050FDB8  FFFDE000
0050FDBC  77F4CA1B
0050FDC0  00000000
0050FDC4  00000000
0050FDC8  FFFDE000
0050FDCC  00000000
0050FDD0  00000000
0050FDD4  00000000
0050FDD8  0050FDBC

Command: Commands are comma separated (like assembly instructions): mov eax, ebx    Default
Running   Breakpoint at 75C722FB set!   Time Wasted Debugging: 0:00:14:12
10:39 PM  4/19/2025
```

Figure 9: x32dbg showing CreateFileW instruction

Cross-Referencing Static and Dynamic Findings

To ensure consistency between static and dynamic analysis, I cross-referenced the imported functions identified in Ghidra with the runtime activity observed during debugging with x32dbg. Two key functions stood out: CreateFileW and WriteFile, both part of kernel32.dll.

In Imported Methods, these functions were listed in the import table, indicating that the malware was designed to perform file operations. During debugging , I confirmed that these functions were not just statically present but were actively executed at runtime.

- **CreateFileW**: Observed a sequence of push instructions leading into a call to CreateFileW, with parameters consistent with file creation or access operations. This strongly suggests the malware was attempting to open or create files on disk.
- **WriteFile**: A separate call was made to WriteFile, with valid register values and stack arguments that appear to point to buffer contents. This behavior aligns with writing data to a file (possibly for logging).

By matching these calls to the statically found imports, I verified that malware is not only capable of file manipulation but actively uses these capabilities during execution. This reinforces the conclusion that the malware engages in direct file I/O operations.

This cross-validation confirms that the import table analysis was accurate and that these functions are critical to the malware’s execution path.

Fixing and Cleanup

After running and analyzing the njRAT sample in a controlled Windows 7 virtual machine, I tracked its behavior using tools like Regshot, Procmon, and x32dbg. The cleanup steps outlined here are based entirely on what I observed during execution — not on assumptions or external reports.

Can Encrypted Files Be Recovered?

No encryption was observed during analysis. I did not see any cryptographic API calls like CryptEncrypt or related Windows CryptoAPI functions being used. There were no signs of file renaming, altered extensions, or ransom notes. The sample operated as a remote access trojan rather than ransomware. Because of that, no recovery of encrypted files is necessary, no files were locked or modified in that way.

Registry Keys That Should Be Deleted

Using Regshot, I confirmed the following registry modifications directly linked to the malware execution:

- *HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WindowsUpdate\Auto Update\UAS*
- *HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WindowsUpdate\Reporting\RebootWatch*
  These are not normally present and were added during execution, indicating possible tampering with update/reboot behaviors.
- Modifications were also made under:
  - Action Center\Checks
  - SessionInfo These appeared only after the sample was run and may have been altered to suppress system warnings or notifications.

All of these keys should be removed as part of a complete cleanup.

Files to Delete

Procmon showed a successful WriteFile operation by the malware to this location:

- C:\Users\Sysuser\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\advpack.url

This file was written by the malware's process and placed in the **Startup folder**, which is a known method for achieving persistence on reboot. This confirms that malware installs itself to automatically run when the system starts.

Additionally, the original sample was executed from the Desktop:

- C:\Users\Sysuser\Desktop\c582b5864a67c2d63f8d3a8faf08b47e94646a96cd81abe507d8d08df13e40c4.exe

Both of these files should be deleted immediately.

Tasks and Services

There was no evidence of scheduled tasks, Windows services being added during execution. The malware did not attempt to install itself nor were any services created in the registry or Task Scheduler.

However, persistence was confirmed through the dropped .url file in the Startup directory, which is sufficient to cause automatic execution on next login.

Network Behavior

While connected to a simulated INetSim router, the malware attempted outbound HTTP requests, which were blocked and logged. Although no C2 communication occurred in this analysis, the behavior confirms that the sample attempts to connect to the network once running. This highlights the risk of real-world data exfiltration or attacker control if run on a live system.

Automation for Faster Analysis

As part of my analysis workflow, I developed a Python tool to automate the tedious process of reviewing Registry changes captured by Regshot. After executing the malware sample and comparing the registry state before and after execution, Regshot generated a large diff file that would've taken a long time to manually inspect.

To speed this up, I wrote a script that:

- Automatically scans the Regshot diff output,
- Extracts the "Keys Added" and "Values Modified" sections,
- Filters them based on a list of keywords related to common malware behavior like:
    - UserAssist (tracks program launches),
    - Startup (persistence),
    - WindowsUpdate (tampering with update mechanisms),
    - Action Center (alert suppression),
    - and others.

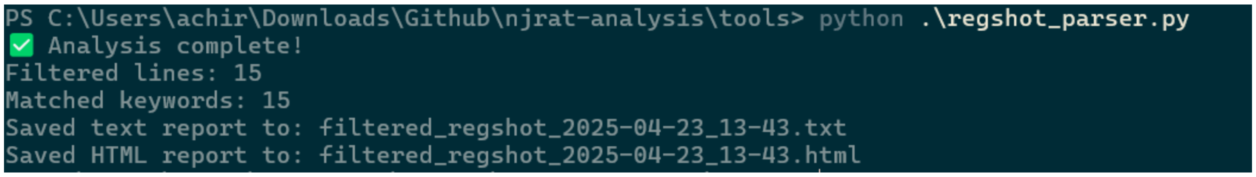
The code is available at https://github.com/chiraanth/njrat-analysis **on Github**

Once filtered, the script generates a clean and readable .html file summarizing all matched registry entries, along with a keyword match frequency count at the bottom.

This automation gives visibility into meaningful registry changes without having to scroll through thousands of unrelated modifications. For instance:

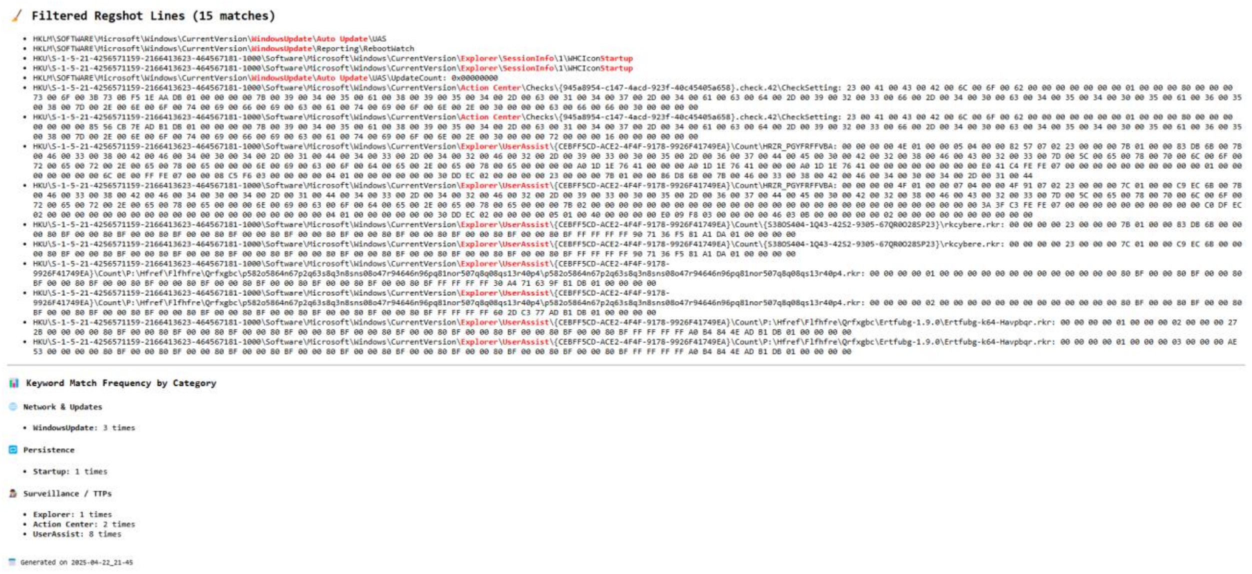
- I found UserAssist keys that logged interaction with suspicious executables (likely our malware sample),
- A Startup entry was placed under SessionInfo,

- Modifications were made to WindowsUpdate settings
- Binary values under Action Center\Checks hinted at possible notification suppression.



```
PS C:\Users\achir\Downloads\Github\njrat-analysis\tools> python .\regshot_parser.py
✅ Analysis complete!
Filtered lines: 15
Matched keywords: 15
Saved text report to: filtered_regshot_2025-04-23_13-43.txt
Saved HTML report to: filtered_regshot_2025-04-23_13-43.html
```

Figure 10: Command line output when running the tool



```
Filtered Regshot Lines (15 matches)

• HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WindowsUpdate\Auto Update\UAS
• HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WindowsUpdate\Reporting\RebootWatch
• HKU\S-1-5-21-4256571159-2166413623-464567181-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\SessionInfo\1\WHCIconStartup
• HKU\S-1-5-21-4256571159-2166413623-464567181-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\SessionInfo\1\WHCIconStartup
• HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WindowsUpdate\Auto Update\UAS\UpdateCount: 0x00000000
• HKU\S-1-5-21-4256571159-2166413623-464567181-1000\Software\Microsoft\Windows\CurrentVersion\Action Center\Checks\{945a8954-c147-4acd-923f-40c45405a658}.check.42\CheckSetting: 23 00 41 00 43 00 42 00 6C 00 6F 00 62 00 00 00 00 00 00 00 01 00 00 00 80 00 00 00
  73 00 6F 00 3B 73 0B F5 1E AA DB 01 00 00 00 00 7B 00 39 00 34 00 35 00 61 00 38 00 39 00 35 00 34 00 2D 00 63 00 31 00 34 00 37 00 2D 00 34 00 61 00 63 00 64 00 2D 00 39 00 32 00 33 00 66 00 2D 00 34 00 30 00 63 00 34 00 35 00 34 00 30 00 35 00 61 00 36 00 35
  00 38 00 7D 00 2E 00 6E 00 6F 00 74 00 69 00 66 00 69 00 63 00 61 00 74 00 69 00 6F 00 6E 00 2E 00 30 00 00 00 63 00 66 00 66 00 30 00 00 00 00
• HKU\S-1-5-21-4256571159-2166413623-464567181-1000\Software\Microsoft\Windows\CurrentVersion\Action Center\Checks\{945a8954-c147-4acd-923f-40c45405a658}.check.42\CheckSetting: 23 00 41 00 43 00 42 00 6C 00 6F 00 62 00 00 00 00 00 00 00 01 00 00 00 80 00 00 00
  00 00 00 00 85 56 CB 7E AD B1 DB 01 00 00 00 00 7B 00 39 00 34 00 35 00 61 00 38 00 39 00 35 00 34 00 2D 00 63 00 31 00 34 00 37 00 2D 00 34 00 61 00 63 00 64 00 2D 00 39 00 32 00 33 00 66 00 2D 00 34 00 30 00 63 00 34 00 35 00 34 00 30 00 35 00 61 00 36 00 35
  00 38 00 7D 00 2E 00 6E 00 6F 00 74 00 69 00 66 00 69 00 63 00 61 00 74 00 69 00 6F 00 6E 00 2E 00 30 00 00 00 72 00 00 00 16 00 00 00 00 00 00
• HKU\S-1-5-21-4256571159-2166413623-464567181-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}\Count\HRZR_PGYFRFFVBA: 00 00 00 00 4E 01 00 00 05 04 00 00 82 57 07 02 23 00 00 00 7B 01 00 00 83 DB 6B 00 7B
  00 46 00 33 00 38 00 42 00 46 00 34 00 30 00 34 00 2D 00 31 00 44 00 34 00 33 00 2D 00 34 00 32 00 46 00 32 00 2D 00 39 00 33 00 30 00 35 00 2D 00 36 00 37 00 44 00 45 00 30 00 42 00 32 00 38 00 46 00 43 00 32 00 33 00 7D 00 5C 00 65 00 78 00 70 00 6C 00 6F 00
  72 00 65 00 72 00 2E 00 65 00 78 00 65 00 00 00 6E 00 69 00 63 00 6F 00 64 00 65 00 2E 00 65 00 78 00 65 00 00 00 00 A0 1D 1E 76 41 00 00 00 A0 1D 1E 76 41 00 00 00 A0 1D 1E 76 41 00 00 00 00 00 00 00 00 E0 41 C4 FE FE 07 00 00 00 00 00 00 00 00 00 00 01 00 00
  00 00 00 00 00 6C 0E 00 FF FE 07 00 00 08 C5 F6 03 00 00 00 00 04 01 00 00 00 00 00 00 30 DD EC 02 00 00 00 00 23 00 00 00 7B 01 00 00 86 D8 6B 00 7B 00 46 00 33 00 38 00 42 00 46 00 34 00 30 00 34 00 2D 00 31 00 44
• HKU\S-1-5-21-4256571159-2166413623-464567181-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}\Count\HRZR_PGYFRFFVBA: 00 00 00 00 4F 01 00 00 07 04 00 00 4F 91 07 02 23 00 00 00 7C 01 00 00 C9 EC 6B 00 7B
  00 46 00 33 00 38 00 42 00 46 00 34 00 30 00 34 00 2D 00 31 00 44 00 34 00 33 00 2D 00 34 00 32 00 46 00 32 00 2D 00 39 00 33 00 30 00 35 00 2D 00 36 00 37 00 44 00 45 00 30 00 42 00 32 00 38 00 46 00 43 00 32 00 33 00 7D 00 5C 00 65 00 78 00 70 00 6C 00 6F 00
  72 00 65 00 72 00 2E 00 65 00 78 00 65 00 00 00 6E 00 69 00 63 00 6F 00 64 00 65 00 2E 00 65 00 78 00 65 00 00 00 7B 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 3A 3F C3 FE FE 07 00 00 00 00 00 00 00 00 00 00 C0 DF EC
  02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 04 01 00 00 00 00 00 00 30 DD EC 02 00 00 00 00 05 01 00 40 00 00 00 00 E0 09 F8 03 00 00 00 00 46 03 0B 00 00 00 00 00 02 00 00 00 00 00 00 00 00 00 00
• HKU\S-1-5-21-4256571159-2166413623-464567181-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}\Count\{S38OS404-1Q43-42S2-9305-67QR0O28SP23}\rkcybere.rkr: 00 00 00 00 23 00 00 00 7B 01 00 00 83 DB 6B 00 00
  00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF FF FF FF FF 90 71 36 F5 81 A1 DA 01 00 00 00 00
• HKU\S-1-5-21-4256571159-2166413623-464567181-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}\Count\{S38OS404-1Q43-42S2-9305-67QR0O28SP23}\rkcybere.rkr: 00 00 00 00 23 00 00 00 7C 01 00 00 C9 EC 6B 00 00
  00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF FF FF FF FF 90 71 36 F5 81 A1 DA 01 00 00 00 00
• HKU\S-1-5-21-4256571159-2166413623-464567181-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-
  9926F41749EA}\Count\P:\Hfref\Flfhfre\Qrfxgbc\p582o5864n67p2q63s8q3n8sns08o47r94646n96pq81nor507q8q08qs13r40p4\p582o5864n67p2q63s8q3n8sns08o47r94646n96pq81nor507q8q08qs13r40p4.rkr: 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 80 BF 00 00 80 BF 00 00 80
  BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF FF FF FF FF 30 A4 71 63 9F B1 DB 01 00 00 00 00
• HKU\S-1-5-21-4256571159-2166413623-464567181-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-
  9926F41749EA}\Count\P:\Hfref\Flfhfre\Qrfxgbc\p582o5864n67p2q63s8q3n8sns08o47r94646n96pq81nor507q8q08qs13r40p4\p582o5864n67p2q63s8q3n8sns08o47r94646n96pq81nor507q8q08qs13r40p4.rkr: 00 00 00 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 80 BF 00 00 80 BF 00 00 80
  BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF FF FF FF FF 60 2D C3 77 AD B1 DB 01 00 00 00 00
• HKU\S-1-5-21-4256571159-2166413623-464567181-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}\Count\P:\Hfref\Flfhfre\Qrfxgbc\Ertfubg-1.9.0\Ertfubg-k64-Havpbqr.rkr: 00 00 00 00 01 00 00 00 02 00 00 00 27
  2B 00 00 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF FF FF FF FF A0 B4 84 4E AD B1 DB 01 00 00 00 00
• HKU\S-1-5-21-4256571159-2166413623-464567181-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}\Count\P:\Hfref\Flfhfre\Qrfxgbc\Ertfubg-1.9.0\Ertfubg-k64-Havpbqr.rkr: 00 00 00 00 01 00 00 00 03 00 00 00 AE
  53 00 00 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF 00 00 80 BF FF FF FF FF A0 B4 84 4E AD B1 DB 01 00 00 00 00

Keyword Match Frequency by Category

Network & Updates
  • WindowsUpdate: 3 times

Persistence
  • Startup: 1 times

Surveillance / TTPs
  • Explorer: 1 times
  • Action Center: 2 times
  • UserAssist: 8 times

Generated on 2025-04-22_21-45
```

Figure11: HTML output from the Regshot filtering tool highlighting malware-related registry changes.

References

1. MalwareBazaar Sample
2. Microsoft API Reference (Win32)

   Referenced for explanations of:
   - GetAsyncKeyState
   - CreateProcessW
   - RegSetValueExW, CreateFileW, InternetReadFile, etc.
3. Windows Sockets API (Winsock) Reference
   - Explanation for WSOCK32 functions like connect, recv, send.