1) Executive Summary

The file enumerates processes (could be injecting code by selection), drops some files and copy itself. Malware behaviours;

- stealing and logging data and sending to clients

- FTP enumeration
- Mail communications
- Windows credentials
- Http requests
- Tcp connections

Name:	Trvqn.exe
Sha256 hash:	8553df1ffeba525257607fddac37d402062fb8b724ebe1f1921e6a0226b37634
Sha1 hash:	78951e958059d39f0d8e8afedccd3eaa130640e9
Md5 hash:	c3e87b51d4d3e523f234a07e32eeae27
Filetype:	.exe
Delivery:	Web download
Tag:	Agent Tesla exe

Name:	Xsvusodttbdseujhgoytwow.dll
Sha256 hash:	8553df1ffeba525257607fddac37d402062fb8b724ebe1f1921e6a0226b37634
Sha1 hash:	78951e958059d39f0d8e8afedccd3eaa130640e9
Md5 hash:	c3e87b51d4d3e523f234a07e32eeae27
filetype	.exe

2) Background: Agent Tesla

Agent Tesla, defined with Trojan-PSW.MSIL.Agens, is a pretty old malware that kills confidential information and sends it to aggressive operators. First of all, it is after credentials stored in different programs: browsers, email clients, FTP/SCP clients, databases, remote management tools, VPN applications and various instant messaging programs. However, Agent Tesla can play clipboard data, record keystrokes and take a screenshot. Agent Tesla sends all the information collected to the attackers via email. However, some modifications of malware can also transfer data using Telegram. It can also upload to a website or FTP server.

3) Analysis

3.1) Stage 1

It is estimated from the entropy rate and the import table that the malware is **unpacked** or hidden. If the import table is examined, 3 flags and **mscore.dll** library are seen. Three imports stand out. These are

- SeccurityPermissionAttribute,
- UnverifiableCodeAttribute
- HttpClient

SeccurityPermissionAttribute: It serves to request file I/O permission for a specific predetermined file

UnverifiableCodeAttribute: Indicates that the module contains code that cannot be verified.

HttpClient: Provides a class for sending HTTP requests and receiving HTTP responses from a resource identified by a URI.

first-bytes-text	M Z .. .. @ .. ..
file-size	7168 bytes
entropy	4.694

Figure 1 - entropy

imports (97)	namespace (13)	flag (3)	group (4)	type (10)	ordinal (0)	library (1)
SecurityPermissionAttribute	System.Security.Permissions	x	security	TypeRef	-	mscoree.dll
UnverifiableCodeAttribute	System.Security	x	security	TypeRef	-	mscoree.dll
HttpClient	System.Net.Http	x	network	TypeRef	-	mscoree.dll
Task`1	System.Threading.Tasks	-	execution	TypeRef	-	mscoree.dll
DebuggableAttribute	System.Diagnostics	-	diagnostic	TypeRef	-	mscoree.dll
Object	System	-	-	TypeRef	-	mscoree.dll

Figure 2 - flags

To be sure of these suspicions, it is necessary to examine the malware using a debugger.

when we scan in debbuger, we can see the entry point of the malware. We also see that the malware provides an http connection. It is understood that malware is making http request and communicating.

Figure 3 – http request

We run the debugger in the main module of the malware (Trvqn.exe) until the source is decoded, we get a module called **Tfghtwfqbh.dat**. We open dump on debbuger and see **Xsvusodttbdseujhgoytwow.dll.**

If we look at code about this dll file, we see that it is encrypted with the base64 encryption technique below.

```
// Token: 0x06000D1F RID: 3359 RVA: 0x00033824 File Offset: 0x00031A24
internal static T smethod_0<T>(string string_0, string string_1)
{
    string_1 = Encoding.UTF8.GetString(Convert.FromBase64String
      (string_1.Replace(Class174.smethod_0(Class321.smethod_0(-1678684154),
      61943), null)));
    return (T)((object)Marshal.GetDelegateForFunctionPointer
      (Class158.GetProcAddress_1(Class158.LoadLibraryA(ref string_0), ref
      string_1), typeof(T)));
```

Figure 4 -base64 encryption

After decrypting the encrypted file, we examine it on the debugger. We open it on the debugger and many **obfuscated classes** and a **dynamic library (dll)** are seen.

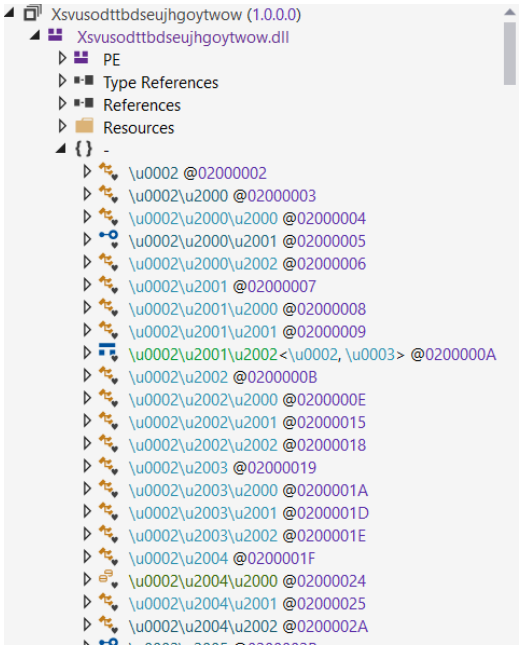

Figure 5 - Obfuscated classes

3.2) Stage 2

In stage 2, after deobfuscate the hidden classes, we can examine the dll downloaded by the malware.

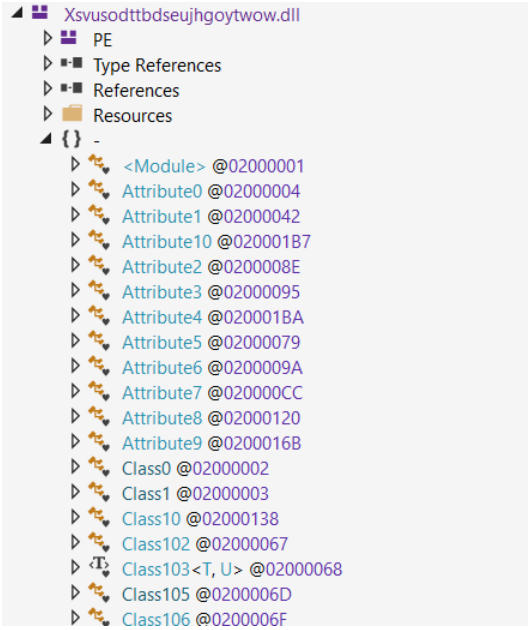

Figure 6 - Deobfuscated classes

```
Process.Start(new ProcessStartInfo
{
    FileName = Class174.smethod_0(Class321.smethod_0(-1678682163), 60880),
    Arguments = Class174.smethod_0(Class321.smethod_0(-1678682160), 59733) + Class121.smethod_0().FullName +
      Class174.smethod_0(Class321.smethod_0(-1678682189), 59310),
    WindowStyle = ProcessWindowStyle.Hidden,
    Verb = Class174.smethod_0(Class321.smethod_0(-1678682232), 60538),
    UseShellExecute = true
```

Figure 8 – executes malicious code with hidden powershell

The use of the Process.Start() method with hidden window style and shell execution potentially be used to execute code without the user's knowledge or consent. When the method_0 function used in the code where powershell is running is examined in method_0, it is seen that **XOR** encryption is done. In other words, this function also provides additional encryption.

```
// Token: 0x06000799 RID: 1945 RVA: 0x00025A20 File Offset: 0x00023C20
public string method_0(string string_0, int int_0)
{
    int num = string_0.Length;
    char[] array = string_0.ToCharArray();
    while (--num >= 0)
    {
        array[num] = (char)((int)array[num] ^ ((int)this.byte_0[int_0 & 15] | int_0));
    }
    return new string(array);
}
```

Figure 9 – XOR encryption with method_0() function

3.3) Stage 3

After static analysis, we examine the traffic on the network when executable malware is run. Many dns queries are seen. In stage 1, it is seen that Trvqn.exe communicates with the address we see while examining, as well as other domains to which it transmits information.

This malware injects Tfghtwfqbh.dat, which is creates 2nd file containing Xsvusodttbdseujhgoytwow.dll from mesiadomnesca.ro shown in figure 10, via powershell. Eventually, dll encrypted with base64 technique. Encrypted dll file is logging many data and copy itself as DKztLTI.exe.

It constantly creates new processes and continues to execute by copying itself under **Appdata**. It forwards data to client while execute.

WIN-5PV2N3JITBN	12
dns.msftncsi.com	1
mail.paradis-bijuterii.ro	1
api.ipify.org	1
mosiadomneasca.ro	1
processhacker.sourceforge.net	1

Figure 10 – unique DNS queries

3.3) Conclusion

In summary, to stealing sensitive data, Agent Tesla can be used to execute other malicious activities, such as downloading additional malware, conducting DDoS attacks, and encrypting files for ransom.

To protect yourself from Agent Tesla and other malware, it is important to practice safe browsing habits and be cautious when opening email attachments or clicking on links from unknown sources. It is also important to keep your operating system and software up to date with the latest security patches and to use antivirus software to detect and remove any potential threats.