

★ Get unlimited access to the best of Medium for less than \$1/week. [Become a member](#)



Decrypting AgentTesla strings and config



Mario Henkel · [Follow](#)

5 min read · Sep 3, 2020

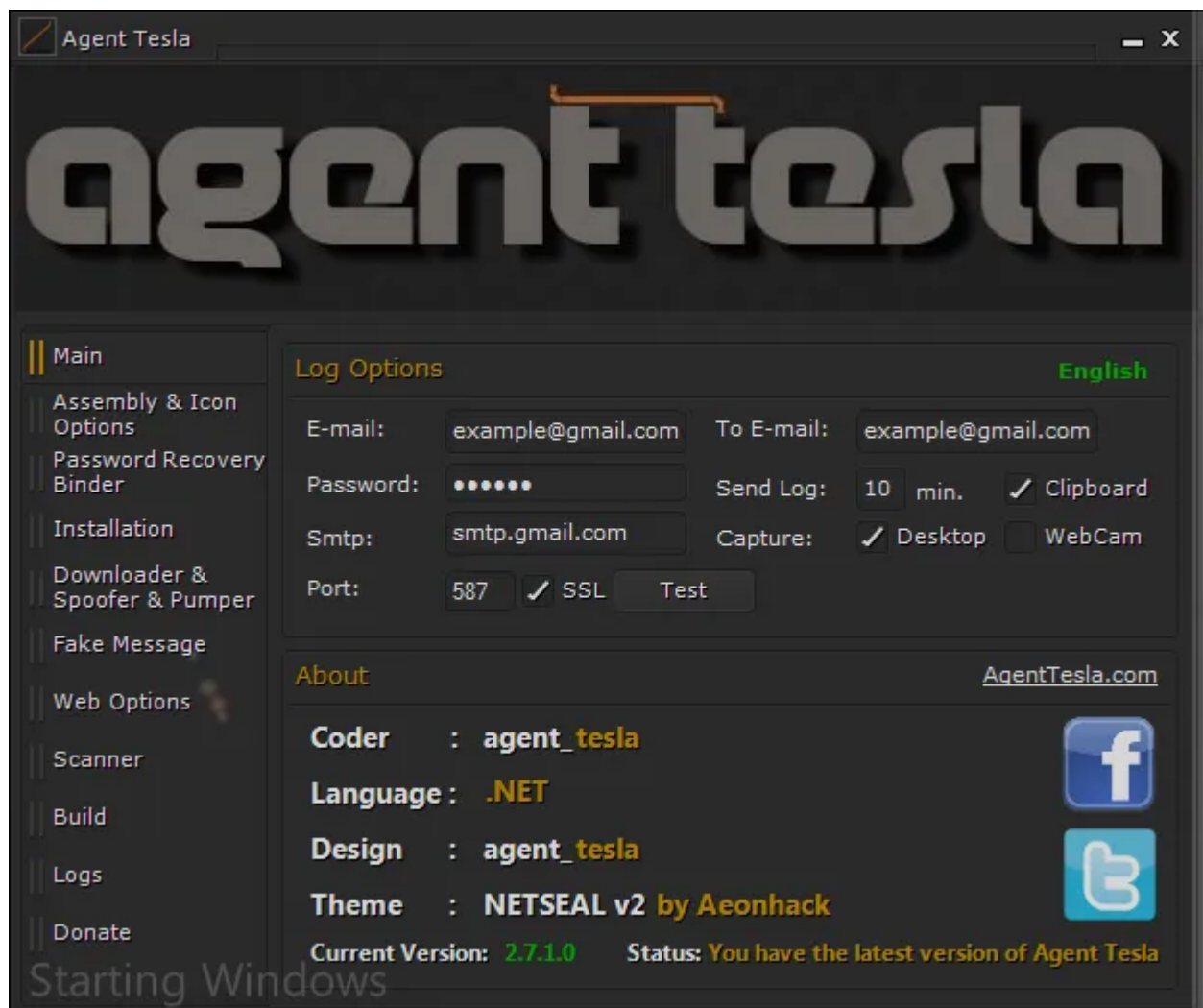


Listen



Share

... More



The nefarious malware AgentTesla has been around for some years and remains a serious threat

Working in cyber security is all about being faster than your adversary to limit or prevent damage to the systems and users you are about to protect.

This naturally leads to the need to automate as much as possible to be able to react quickly to new threats. As you can see from the stats of [Any.run](#) AgentTesla is still one of the most popular malware families when it comes to commodity malware:



ANY.RUN @anyrun_app · 31. Aug.

TOP10 last week's threats by uploads

↓	#Emotet	1615 (1861)	🔥
↑	#NjRAT	150 (120)	
↑	#AgentTesla	113 (110)	
↑	#Remcos	103 (50)	
↓	#FormBook	84 (88)	
↑	#AsyncRAT	84 (41)	
↑	#NanoCore	65 (51)	
↑	#CobaltStrike	65 (48)	
↓	#LokiBot	60 (70)	
↑	#Qbot	59 (38)	



Malware Trends Tracker | ANY.RUN

Explore dynamic articles about various malware types. Look at latest analyzes and IOCs in real-time, track ...

[any.run](#)



1



24



40



Spenden

AgentTesla remains a threat

Due to the number of samples to analyze I created a little tool called [Edison](#) which is able to decrypt all strings used by AgentTesla with the fact in mind that the exfiltration methods are also present in the strings. Equipped with this knowledge let's dive in!

The sample

My preferred site to get interesting samples is [bazaar.abuse.ch](#) so you may also use this site to get the following sample to follow along with these steps — in a secured VM of course (We are about to execute the malware)

[d29da0500ff7aecab3d24397cb745554f399dce5ab59f4ed7a95f6f959b62584](#)

Required tools

As already explained you **need a VM** to run our little experiment since we will execute this sample. If you aren't already using it, this is a good moment to start using FlareVM which is maintained by FireEye.

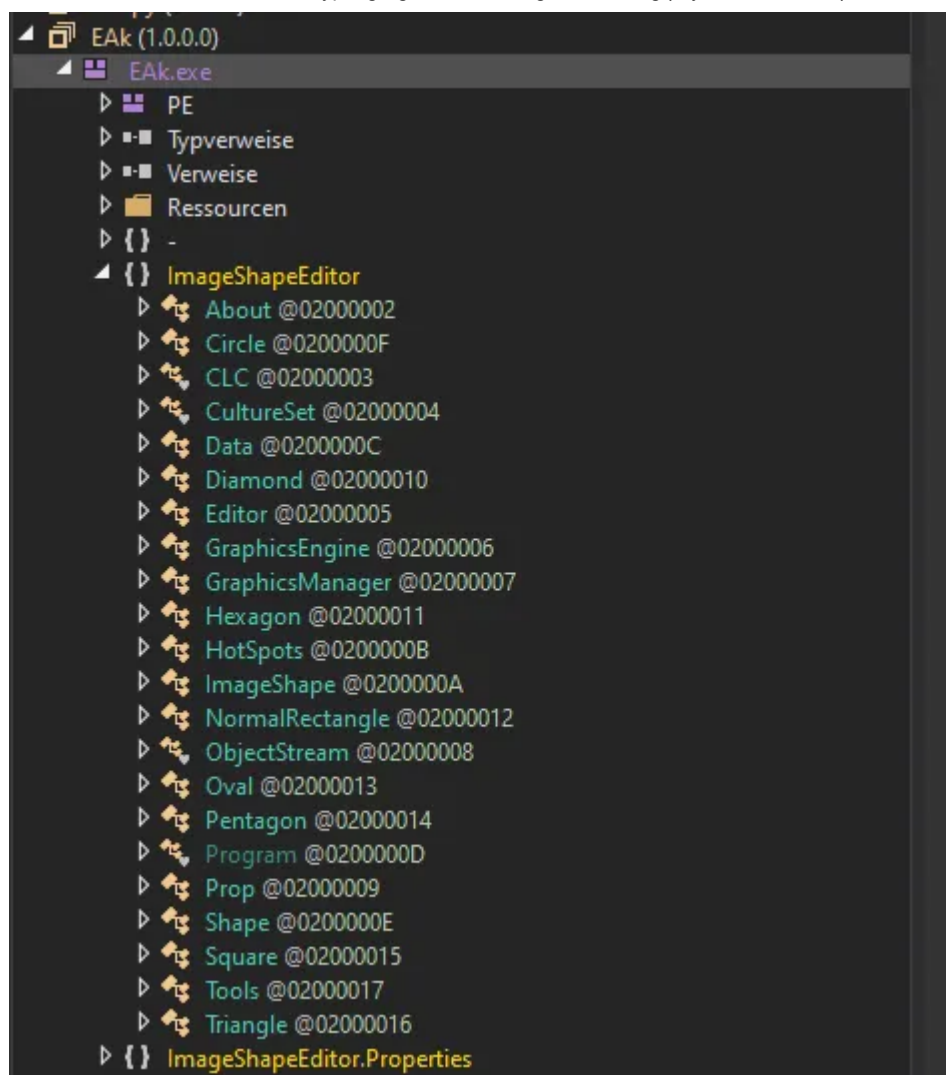
Once you got your VM up and running it's time to get Hollows Hunter. The reason for this is, that most if not all malware comes "packed". This means that the true intention of the malware is not visible at first glance. Only if the packer deems your system worthy to get infected (No VM, only specific regions etc.) it will unpack the malware and pass control to it. Going in too much details is out of the scope of this post so we chose the automatic way with Hollows Hunter. Once the malware is unpacked, Hollows Hunter will dump it to disk, fix some things and you are good to go.

Not necessary but really helpful in our analysis is dnSpy. If you are using FlareVM, it should already be installed.

Last but not least you'll need Edison to automatically extract the strings from the sample.

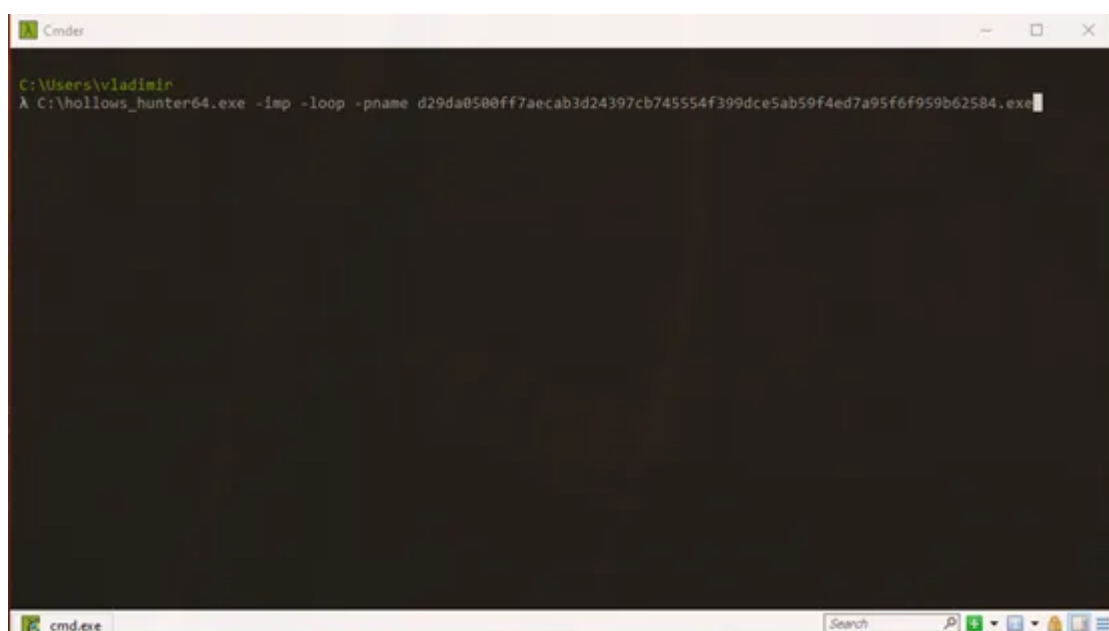
Let's infect ourselves

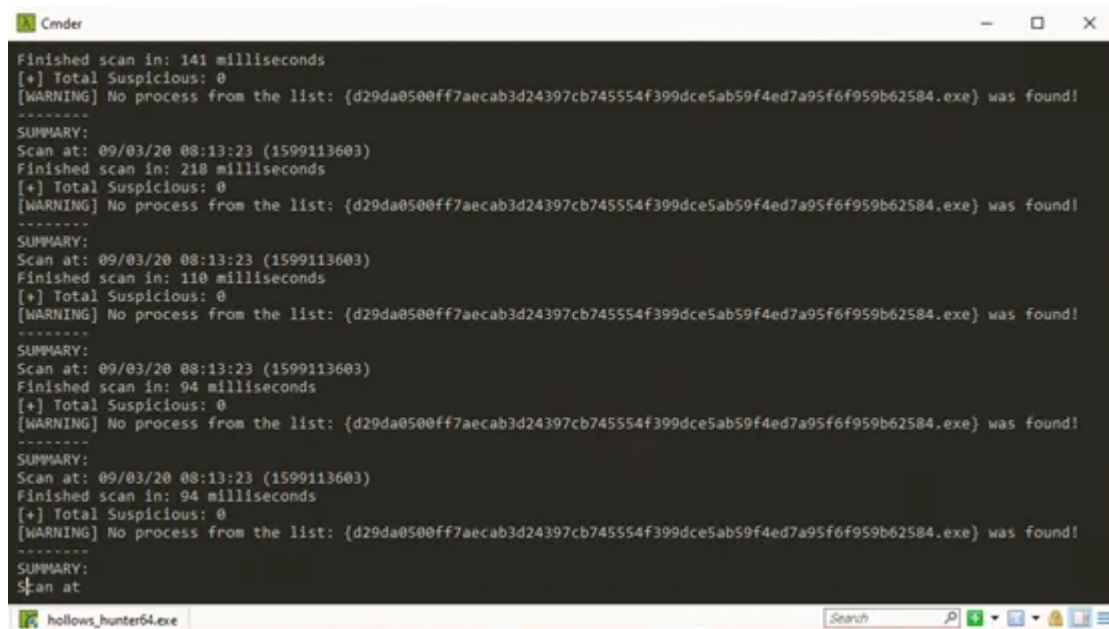
Once you downloaded the sample, we open it up in dnSpy:



That doesn't really look like AgentTesla to be honest

As we can see when we go through the methods it doesn't look like AgentTesla so at this point we assume that it's packed. We now use Hollows Hunter to watch for injections and automatically dump them to disk:





```

Cmder
Finished scan in: 141 milliseconds
[+] Total Suspicious: 0
[WARNING] No process from the list: {d29da0500ff7aecab3d24397cb745554f399dce5ab59f4ed7a95f6f959b62584.exe} was found!
-----
SUMMARY:
Scan at: 09/03/20 08:13:23 (1599113603)
Finished scan in: 218 milliseconds
[+] Total Suspicious: 0
[WARNING] No process from the list: {d29da0500ff7aecab3d24397cb745554f399dce5ab59f4ed7a95f6f959b62584.exe} was found!
-----
SUMMARY:
Scan at: 09/03/20 08:13:23 (1599113603)
Finished scan in: 110 milliseconds
[+] Total Suspicious: 0
[WARNING] No process from the list: {d29da0500ff7aecab3d24397cb745554f399dce5ab59f4ed7a95f6f959b62584.exe} was found!
-----
SUMMARY:
Scan at: 09/03/20 08:13:23 (1599113603)
Finished scan in: 94 milliseconds
[+] Total Suspicious: 0
[WARNING] No process from the list: {d29da0500ff7aecab3d24397cb745554f399dce5ab59f4ed7a95f6f959b62584.exe} was found!
-----
SUMMARY:
Scan at: 09/03/20 08:13:23 (1599113603)
Finished scan in: 94 milliseconds
[+] Total Suspicious: 0
[WARNING] No process from the list: {d29da0500ff7aecab3d24397cb745554f399dce5ab59f4ed7a95f6f959b62584.exe} was found!
-----
SUMMARY:
Scan at:

```

If Hollows Hunter is working, you should see output similar to this

After we got Hollows Hunter in place, we start the malware itself and wait until a new process is spawned. As expected Hollows Hunter catches this:



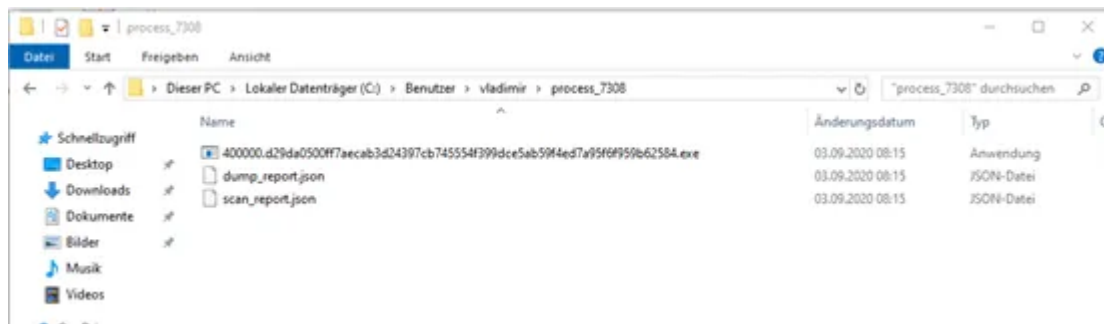
```

[+] List of suspicious:
[0]: PID: 7308, Name: d29da0500ff7aecab3d24397cb745554f399dce5ab59f4ed7a95f6f959b62584.exe
>> Scanning PID: 7308 : d29da0500ff7aecab3d24397cb745554f399dce5ab59f4ed7a95f6f959b62584.exe : 32b
[*] This is a .NET payload and may require Entry Point correction. Current EP: 49e3e
[*] Found possible Entry Point: 49e3e
[*] This is a .NET payload and may require Entry Point correction. Current EP: 49e3e
[*] Found possible Entry Point: 49e3e
>> Detected PID: 7308 (1.00%)
-----
SUMMARY:
Scan at: 09/03/20 08:14:47 (1599113687)
Finished scan in: 687 milliseconds
[+] Total Suspicious: 1
[+] List of suspicious:
[0]: PID: 7308, Name: d29da0500ff7aecab3d24397cb745554f399dce5ab59f4ed7a95f6f959b62584.exe
>> Scanning PID: 7308 : d29da0500ff7aecab3d24397cb745554f399dce5ab59f4ed7a95f6f959b62584.exe : 32b
[*] This is a .NET payload and may require Entry Point correction. Current EP: 49e3e
[*] Found possible Entry Point: 49e3e
[*] This is a .NET payload and may require Entry Point correction. Current EP: 49e3e
[*] Found possible Entry Point: 49e3e
>> Detected PID: 7308 (1.00%)
-----
SUMMARY:
Scan at: 09/03/20 08:14:48 (1599113688)
Finished scan in: 734 milliseconds
[+] Total Suspicious: 1
[+] List of suspicious:
[0]: PID: 7308, Name: d29da0500ff7aecab3d24397cb745554f399dce5ab59f4ed7a95f6f959b62584.exe
>> Scanning PID: 7308 : d29da0500ff7aecab3d24397cb745554f399dce5ab59f4ed7a95f6f959b62584.exe : 32b

```

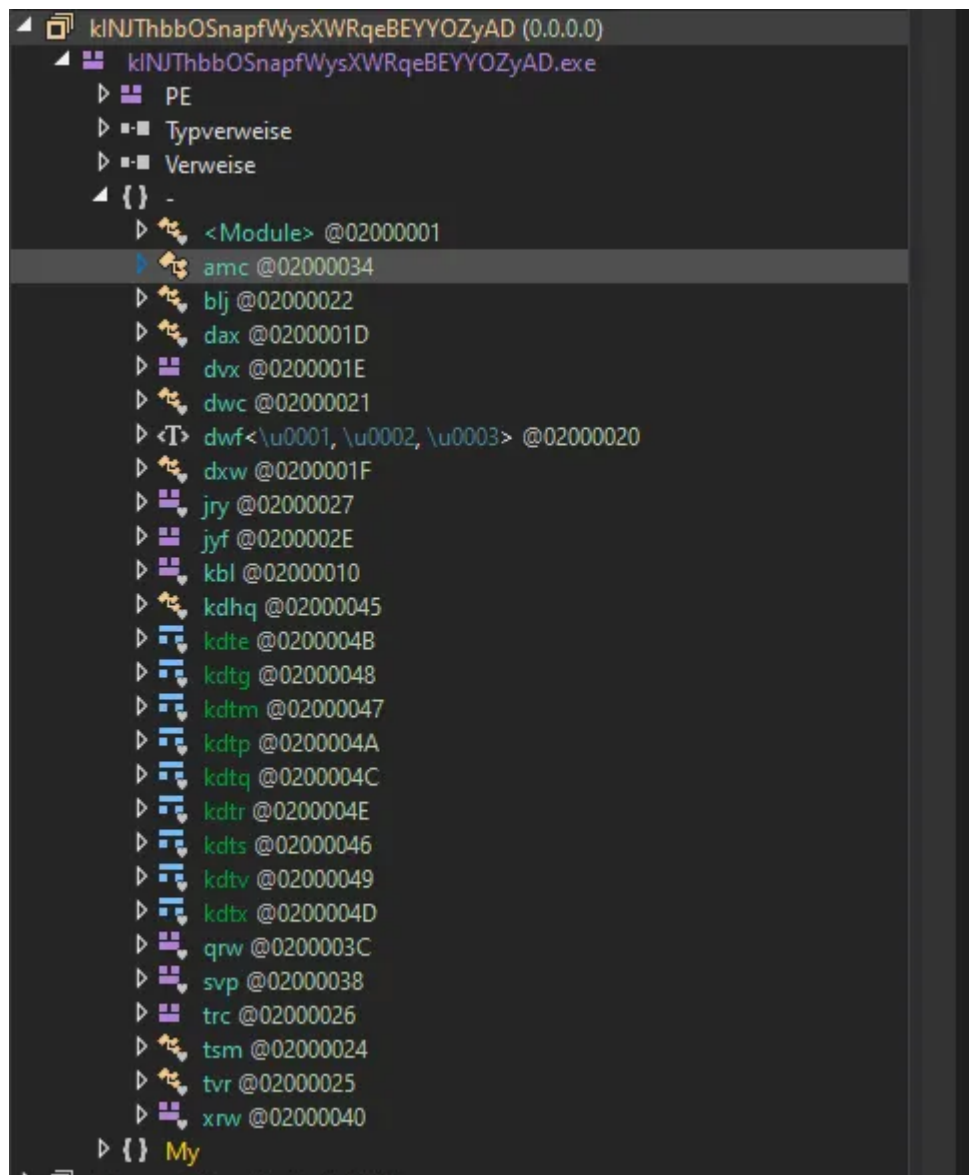
Gotcha!

Once we made sure that we got a dumped sample we can kill the malware itself and Hollows Hunter:



The dumped file

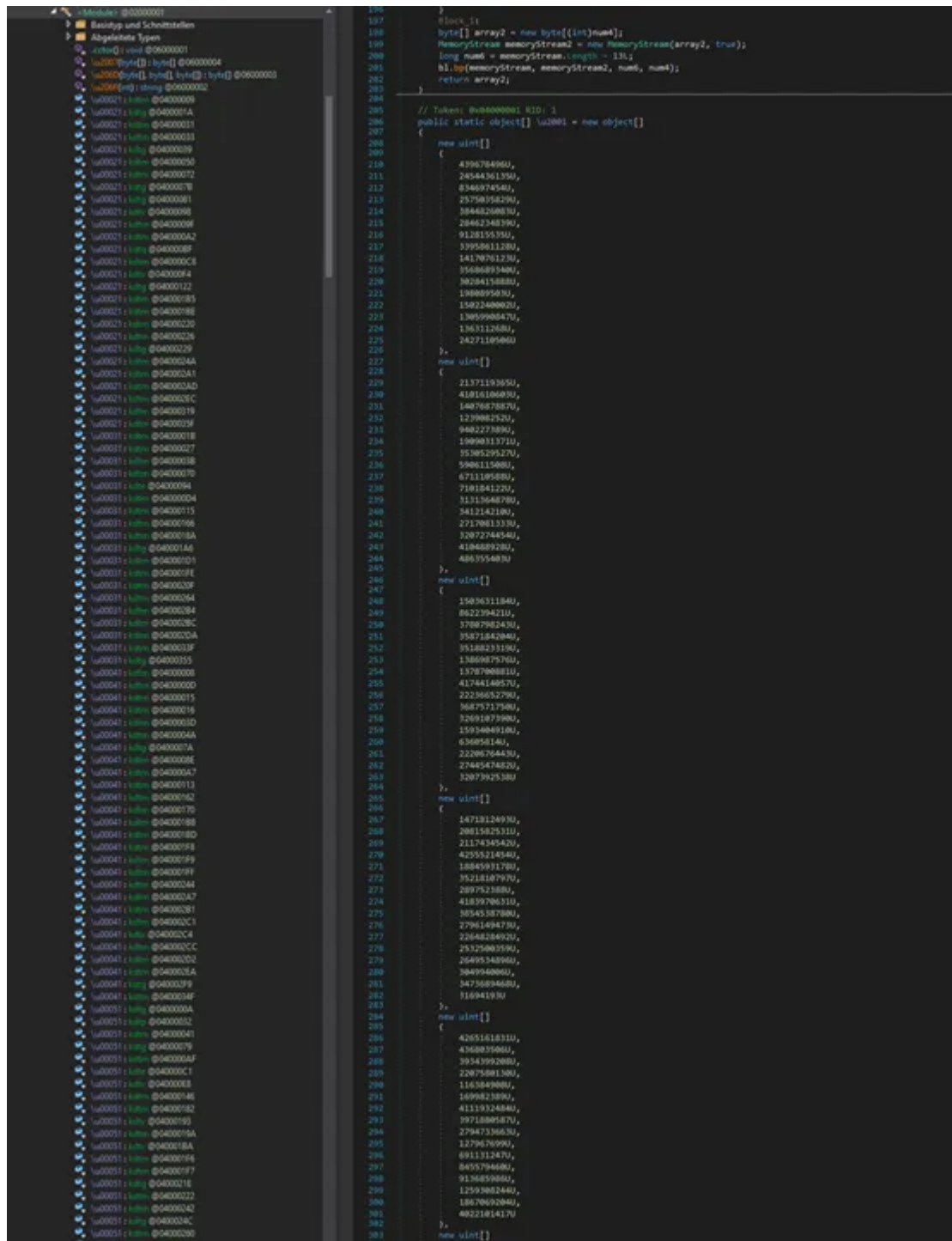
To confirm our assumptions we open this file in dnSpy again:



Hello, AgentTesla!

One characteristic thing about AgentTesla is a huge object array of uint[] which turn out to be the encrypted strings:



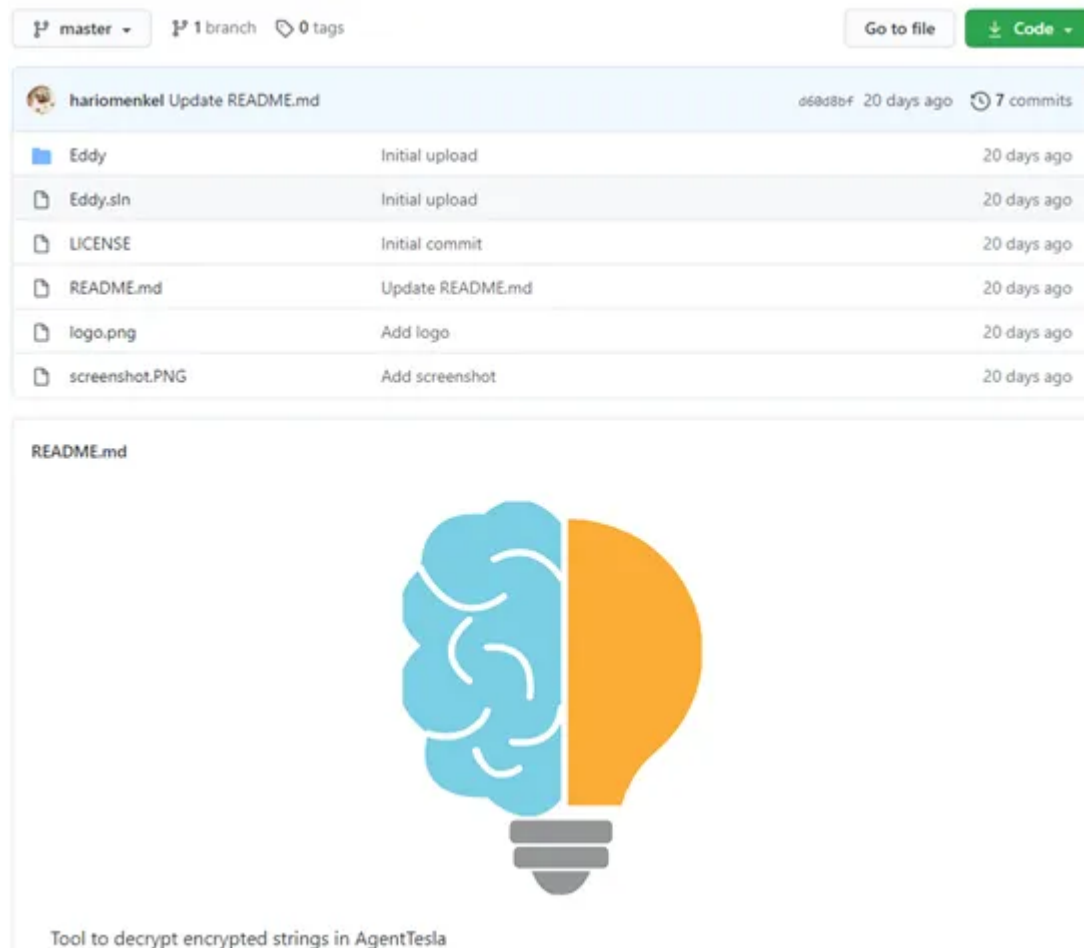


A good training for your fingers if you decide to scroll through the list



Key? IV? CreateDecryptor? Looks like we found the juicy parts!

You could now dynamically analyze the malware, set breakpoints, work your way to the obfuscated control flow OR we fire up Edison to just decrypt all strings for us:



If not already downloaded, this is your chance!

Once downloaded you can start it without arguments to see the usage:

```
C:\Users\vladimir> cd C:\Users\vladimir\Downloads\Edison-master\Eddy\bin\Debug
C:\Users\vladimir\Downloads\Edison-master\Eddy\bin\Debug> Eddy.exe
> Edison
>> Automatically extract encrypted strings from AgentTesla samples
>> @hariomenkel / https://github.com/hariomenkel/Edison
Usage: Eddy.exe <Input File> <Output File>
Example: Eddy.exe AgentTesla.exe DecryptedStrings.txt

C:\Users\vladimir> Eddy.exe
λ |
```

No rocket science. Really.

We do as advised:



```
1 .lnk
2 .lnk
3 .lnk
4 WScript.Shell
5 CreateShortcut
6 .lnk
7 TargetPath
8 cmd.exe
9 WorkingDirectory
10 Arguments
11 /c start
12 &start
13 & exit
14 IconLocation
15 Save
16 .lnk
17 WScript.Shell
18 CreateShortcut
19 .lnk
20 TargetPath
21 cmd.exe
22 WorkingDirectory
23 Arguments
24 /c start
25 &explorer /root,"%CD%
26 " & exit
27 IconLocation
28 %SystemRoot%\system32\SHELL32.dll,3
29 Save
30 Software\Classes\
31 OpenSubKey
32 OpenSubKey
33 GetValue
34 \DefaultIcon\
35 GetValue
```

```
170 password.  
171 Application:  
172 .html  
173 <html>  
174 </html>  
175 chuk5anderson@yandex.ru  
176 🌐📧🔗  
177 smtp.yandex.com  
178 chuk5anderson@yandex.ru  
179 chuk5anderson@yandex.ru  
180 text/html  
181 .html  
182 .html
```

Teach me master!

You can browse the repository to see the complete source code but for reference purposes you can see the interesting part here:

```
byte[] array_Key = new byte[32];
byte[] array_IV = new byte[16];
int keyLength = 32;
int ivLength = 16;

Assembly a = null;
try
{
    a = Assembly.LoadFile(input);
}
catch (Exception ex)
{
    Console.WriteLine("Error while loading the file: " + ex.Message);
    return;
}

Module[] modules = a.GetModules();
var fields = modules[0].GetFields();
foreach (var field in fields)
{
    var objArr = field.GetValue(null);
    var values = (object[])objArr;

    for (int i = 0; i < values.Length; i++)
    {
        try
        {
            uint[] encryptedValue = (uint[])values[i];
            byte[] arrEncryptedValue = new byte[encryptedValue.Length * 4];
            Buffer.BlockCopy(encryptedValue, 0, arrEncryptedValue, 0, encryptedValue.Length * 4);
            byte[] arrPayload = arrEncryptedValue;
            int offsetKeyAndIV = arrPayload.Length - (keyLength + ivLength);
            byte[] array_EncryptedValue = new byte[offsetKeyAndIV];
            Buffer.BlockCopy(arrPayload, 0, array_Key, 0, keyLength);
            Buffer.BlockCopy(arrPayload, keyLength, array_IV, 0, ivLength);
            Buffer.BlockCopy(arrPayload, keyLength + ivLength, array_EncryptedValue, 0, offsetKeyAndIV);

            using (System.IO.StreamWriter file = new System.IO.StreamWriter(output, true))
            {
                Console.WriteLine(":: Success :: " + Encoding.UTF8.GetString(Decrypt(array_EncryptedValue, array_Key, array_IV)));
                file.WriteLine(Encoding.UTF8.GetString(Decrypt(array_EncryptedValue, array_Key, array_IV)));
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine("!! Error !! " + ex.Message);
        }
    }
}
```

Open in app ↗



Search



fields of the first module which turn out to be the uint[] elements.

Through reversing the sample I found out that the Key and IV for every uint[]/string differs and is found at the end of every uint[]. Since we know the size of the key and

the IV we can use these values to pass those values to the decrypt function we saw earlier.

Agenttesla

Malware

Reverse Engineering



Follow

Written by Mario Henkel

24 Followers

More from Mario Henkel



Mario Henkel

Using CAPE Sandbox and FOG to analyze malware on physical machines