LYHINS' LAB

LSCP Responsible Disclosure Lab

How White-Box hacking works: “Ok, Google, I wanna pwn this app….”

July 18, 2020 | No Comments

Mobile applications should not trust other applications on the device.

The new generation likes it when an organization wants them to use lightweight mobile devices instead of PCs. In this case, employees don't spend time and efforts walking around the room to find that single machine with Windows XP in the left corner of the room and to wait until it boots, unlags, and proceeds… People are use to carrying phones or tablets and send messages to each other all the time. Finally, a tablet usually is significantly less expensive, than a PC or a laptop, that increases efficiency.

However, who and how can control those devices? Particularly, a new tablet would have dozens of applications installed during the next months. Thus, dozens of different groups of people can legally execute code on your devices in your network and **might sell this right legally.** How many attack vectors can you count if an organization has hundreds of devices, and each application can get to any device including the router within the network? How would you protect your people and your networks against these attacks? Finally, what if you **do not own** all the devices in your network, **how would you ask** your employees to protect themselves and the perimeter? Hard questions, solvable in hours of consultations, but the basic advice is **do not trust applications**. This also works if you have your own application, you have a large base of users and plan to have more, you work with PII, and do not want to violate privacy laws like CCPA or GDPR.

Typical exploit case

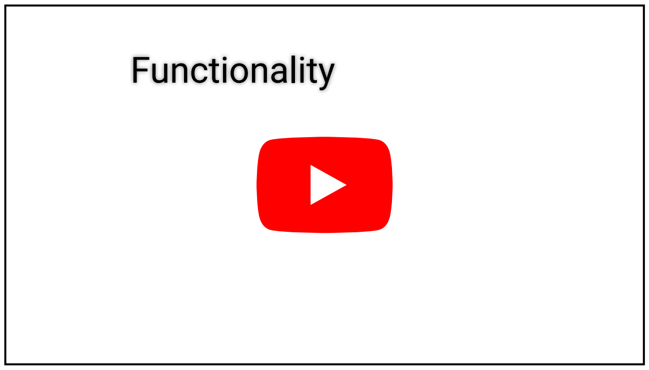
To better understand the threat, we highly recommend to repeat the next attack manually on a device.

Target posture

“Astrid Tasks” is an open-source application, that flourished before Yahoo bought the app and was discontinued in 2013. Then, Alex Baker found open-source parts of the app and decided to resurrect it. As for July 2020, the application is completely free with no ads, has more than 1000 stars on Github and more than 5000 active users. Thus, Lyhin's Lab decided to slightly improve the security of the “Tasks” application. Current app version is 9.7.3.

Bug discovery

Even though the “Tasks” application returns to life at a very early stage, this application has a couple of interesting possibilities – particularly, the integration with “Ok, Google” service.



Unfortunately, the “VoiceCommandActivity” application component allows arbitrary applications on a device to add tasks with no restrictions.

```
 dz> run app.activity.start --component org.tasks.debug org.tasks.voice.VoiceCommandActivity --
action=com.google.android.gm.action.AUTO_SEND --extra string android.intent.extra.TEXT "Super
Important Task: visit my website https://lyhinslab.org"
```

Activity representation:

app/src/main/AndroidManifest.xml

```
 <activity
       android:name=".voice.VoiceCommandActivity"
```

Search...
Search

Recent Posts

Temporary LLab suspension

How White-Box hacking works: InvoicePlane – A Lot Of XSS And A Couple Of BAC Vulnerabilities

Lifehacks for hackers: what certification next?

How White-Box hacking works: XSS + CSRF in Arunna

Lifehacks for hackers: The value of “No”.

Recent Comments

Archives

October 2022
January 2022
December 2021
November 2021
October 2021
September 2021
August 2021
July 2021
June 2021
May 2021
April 2021
March 2021
February 2021
January 2021
December 2020
November 2020
October 2020
September 2020
August 2020
July 2020
June 2020
May 2020
April 2020
March 2020

Categories

Uncategorized

Meta

Log in

```
      android:theme="@style/TranslucentDialog">
      <intent-filter>
        <action android:name="com.google.android.gm.action.AUTO_SEND"/>

        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
      </intent-filter>
    </activity>
```

Entries feed

Comments feed

WordPress.org

Activity implementation:

org/tasks/voice/VoiceCommandActivity.java

```java
public class VoiceCommandActivity extends InjectingAppCompatActivity {

  @Inject TaskCreator taskCreator;
  @Inject @ForApplication Context context;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    Intent intent = getIntent();
      if ("com.google.android.gm.action.AUTO_SEND".equals(intent.getAction())) {
      final String text = intent.getStringExtra(Intent.EXTRA_TEXT);
      if (!isNullOrEmpty(text)) {
        taskCreator.basicQuickAddTask(text);
        Toast.makeText(context, getString(R.string.voice_command_added_task), Toast.LENGTH_LONG)
            .show();
      }
      finish();
    }
  }

  @Override
  public void inject(ActivityComponent component) {
    component.inject(this);
  }
}
```

How to repair

In the Android world, components like activities may require permissions. Thus, we could find a permission that a trusted application has, but an untrusted application cannot have. Notice, that sometimes it's not possible to find such permission, and, generally speaking, android components cannot easily get the actual sender of the intent in runtime in a secure way.

The activity representation states that the activity action is:

```
com.google.android.gm.action.AUTO_SEND
```

As for today, an official guide on developer.android.com says nothing about integration with "Ok, Google" functionality on tablets. But the Internet has traces of the next signature level permission:

```
com.google.android.gm.permission.AUTO_SEND
```

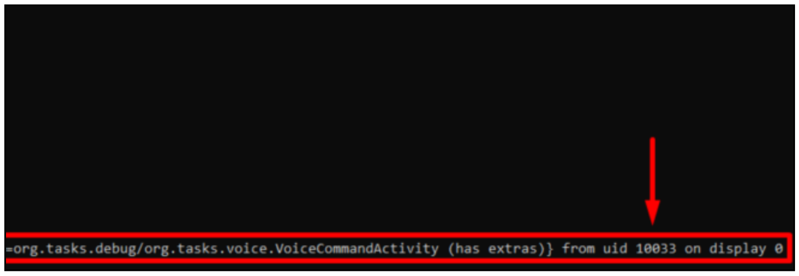
Because of the signature level permissions design, com.google.android.gm.permission.AUTO_SEND permission could be obtained by those applications only that were signed by the same signature Gmail application was signed. If the legal "Ok, Google" intent sender has this permission, VoiceCommandActivity should require com.google.android.gm.permission.AUTO_SEND permission to be secure. Follow the next steps to detect the package that sends intents and to ensure that this application has "com.google.android.gm.permission.AUTO_SEND" permission.

1. Clear logcat, and start log inspecting

```
adb shell
su
logcat -c
logcat
```

2. Add a new task with "Ok, Google" functionality manually

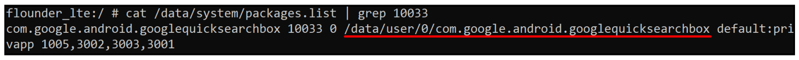
3. Find the right place in logs. Some application sends an intent to VoiceCommandActivity



```
=org.tasks.debug/org.tasks.voice.VoiceCommandActivity (has extras)} from uid 10033 on display 0
```

4. Find the package with the right UID

```
cat /data/system/packages.list | grep 10033
```



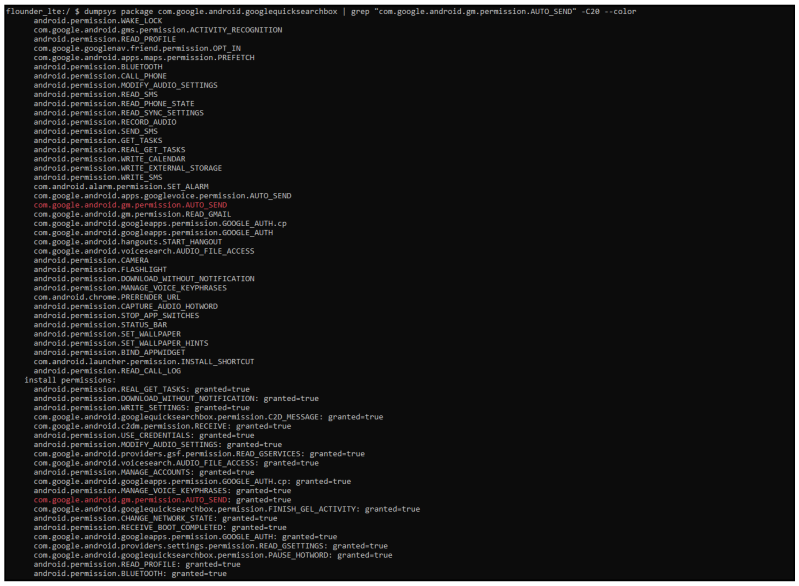
```
flounder_lte:/ # cat /data/system/packages.list | grep 10033
com.google.android.googlequicksearchbox 10033 0 /data/user/0/com.google.android.googlequicksearchbox default:pri
vapp 1005,3002,3003,3001
```

The package is:

```
com.google.android.googlequicksearchbox
```

5. Inspect the package permissions.

```
dumpsys package com.google.android.googlequicksearchbox | grep
"com.google.android.gm.permission.AUTO_SEND" -C20 --color
```



```
flounder_lte:/ $ dumpsys package com.google.android.googlequicksearchbox | grep "com.google.android.gm.permission.AUTO_SEND" -C20 --color
      android.permission.WAKE_LOCK
      com.google.android.gms.permission.ACTIVITY_RECOGNITION
      android.permission.READ_PROFILE
      com.google.googlenav.friend.permission.OPT_IN
      com.google.android.apps.maps.permission.PREFETCH
      android.permission.BLUETOOTH
      android.permission.CALL_PHONE
      android.permission.MODIFY_AUDIO_SETTINGS
      android.permission.READ_SMS
      android.permission.READ_PHONE_STATE
      android.permission.READ_SYNC_SETTINGS
      android.permission.RECORD_AUDIO
      android.permission.SEND_SMS
      android.permission.GET_TASKS
      android.permission.REAL_GET_TASKS
      android.permission.WRITE_CALENDAR
      android.permission.WRITE_EXTERNAL_STORAGE
      android.permission.WRITE_SMS
      com.android.alarm.permission.SET_ALARM
      com.google.android.apps.googlevoice.permission.AUTO_SEND
      com.google.android.gm.permission.AUTO_SEND
      com.google.android.gm.permission.READ_GMAIL
      com.google.android.googleapps.permission.GOOGLE_AUTH.cp
      com.google.android.googleapps.permission.GOOGLE_AUTH
      com.google.android.hangouts.START_HANGOUT
      com.google.android.voicesearch.AUDIO_FILE_ACCESS
      android.permission.CAMERA
      android.permission.FLASHLIGHT
      android.permission.DOWNLOAD_WITHOUT_NOTIFICATION
      android.permission.MANAGE_VOICE_KEYPHRASES
      com.android.chrome.PRERENDER_URL
      android.permission.CAPTURE_AUDIO_HOTWORD
      android.permission.STOP_APP_SWITCHES
      android.permission.STATUS_BAR
      android.permission.SET_WALLPAPER
      android.permission.SET_WALLPAPER_HINTS
      android.permission.BIND_APPWIDGET
      com.android.launcher.permission.INSTALL_SHORTCUT
      android.permission.READ_CALL_LOG
    install permissions:
      android.permission.REAL_GET_TASKS: granted=true
      android.permission.DOWNLOAD_WITHOUT_NOTIFICATION: granted=true
      android.permission.WRITE_SETTINGS: granted=true
      com.google.android.googlequicksearchbox.permission.C2D_MESSAGE: granted=true
      com.google.android.c2dm.permission.RECEIVE: granted=true
      android.permission.USE_CREDENTIALS: granted=true
      android.permission.MODIFY_AUDIO_SETTINGS: granted=true
      com.google.android.providers.gsf.permission.READ_GSERVICES: granted=true
      com.google.android.voicesearch.AUDIO_FILE_ACCESS: granted=true
      android.permission.MANAGE_ACCOUNTS: granted=true
      com.google.android.googleapps.permission.GOOGLE_AUTH.cp: granted=true
      android.permission.MANAGE_VOICE_KEYPHRASES: granted=true
      com.google.android.gm.permission.AUTO_SEND: granted=true
      com.google.android.googlequicksearchbox.permission.FINISH_GEL_ACTIVITY: granted=true
      android.permission.CHANGE_NETWORK_STATE: granted=true
      android.permission.RECEIVE_BOOT_COMPLETED: granted=true
      com.google.android.googleapps.permission.GOOGLE_AUTH: granted=true
      com.google.android.providers.settings.permission.READ_GSETTINGS: granted=true
      com.google.android.googlequicksearchbox.permission.PAUSE_HOTWORD: granted=true
      android.permission.READ_PROFILE: granted=true
      android.permission.BLUETOOTH: granted=true
```

This package has the right permission.

Fix

Repair the VoiceCommandActivity activity representation in manifest. Before:

```
<activity
      android:name=".voice.VoiceCommandActivity"
      android:theme="@style/TranslucentDialog">

      <intent-filter>

        <action android:name="com.google.android.gm.action.AUTO_SEND"/>

        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
      </intent-filter>
    </activity>
```

After:

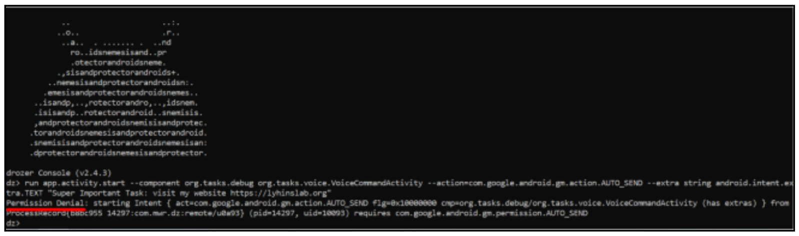
```
<activity
      android:name=".voice.VoiceCommandActivity"
      android:theme="@style/TranslucentDialog"
      android:permission="com.google.android.gm.permission.AUTO_SEND">

      <intent-filter>

        <action android:name="com.google.android.gm.action.AUTO_SEND"/>

        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
      </intent-filter>
    </activity>
```

Result:



```
         ..                    ..:.
        ..o..                  .r..
         ..a..  . ....... .  ..nd
           ro..idsnemesisand..pr
           .otectorandroidsneme.
        .,sisandprotectorandroids+.
      ..nemesisandprotectorandroidsn:.
     .emesisandprotectorandroidsnemes..
   ..isandp,..,rotectorandro,..,idsnem.
   .isisandp..rotectorandroid..snemisis.
   ,andprotectorandroidsnemisisandprotec.
  .torandroidsnemesisandprotectorandroid.
  .snemisisandprotectorandroidsnemesisan:
  .dprotectorandroidsnemesisandprotector.

drozer Console (v2.4.3)
dz> run app.activity.start --component org.tasks.debug org.tasks.voice.VoiceCommandActivity --action=com.google.android.gm.action.AUTO_SEND --extra string android.intent.ex
tra.TEXT "Super Important Task: visit my website https://lyhinslab.org"
Permission Denial: starting Intent { act=com.google.android.gm.action.AUTO_SEND flg=0x10000000 cmp=org.tasks.debug/org.tasks.voice.VoiceCommandActivity (has extras) } from
ProcessRecord{b8bc955 14297:com.mwr.dz:remote/u0a93} (pid=14297, uid=10093) requires com.google.android.gm.permission.AUTO_SEND
dz>
```

Permission denial

Permission denial for Drozer, while the legal functionality works fine. Looks super easy, requires hours of reseach to understand, at least for the regular people like we are.

Conclusion

We recommend to "Tasks" application maintainers to enforce the VoiceCommandActivity to require com.google.android.gm.permission.AUTO_SEND permission. We notified Alex on a vulnerability about two months ago and unfortunately got no reaction, so it's time to disclose the vulnerability now. Code changes are above.

UPD July 23. Changes had been commited to the official repository, so the vulnerability is fixed.

The possible future of well-growing companies brings new people and new devices, and new risks with them. On the other side, how about medium-sized companies today? How many unique small applications can access the internal infrastructure of a regular company? Easy to calculate, hard to mitigate.

Nowadays, a lot of issues discovered during mobile security application assessments contain permission related vulnerabilities, reasonably – in the mobile world, developers should use controls that the platform offers. Assess android permissions respectfully, verify permissions for exported components, stay safe yourself, and protect your assets.

LL advises to all the researchers do not break real applications illegally. This fun leads to broken businesses and lives, and, most likely, will not make an attacker really rich.

← Lifehacks for hackers: When to relax and when to do not
Lifehacks for hackers: the family networking weaknesses, 0-days guaranteed →

Leave a Reply

You must be logged in to post a comment.