PinkDroid: A System Level Contents Blocker

Wonsup Yoon Bilgehan Bingol Nak Hyun Choi

2018. 6. 10.

1 Development Environment

We used two machines. One for compilation and the other one for execution.

1.1 Compilation Server

- OS: Ubuntu 16.04 LTS (In a LXC container)
- Java: javac 1.8.0_162 (OpenJDK)

We followed this link[2], but we do not set a limit on ccache.

1.2 Execution Laptop

- OS: macOS High Sierra 10.13.5
- Java: javac 1.8.0_172 (OracleJDK, but not used)

We did same setup on compilation server in a case-sensitive disk image[2][3].

2 AOSP version

We used AOSP `android-7.1.1_r22` on both sides.

3 Lunch option

We used `aosp_x86-eng` on both sides.

4 Execution method

We used an emulator on the execution laptop. First, we compiled aosp at the server side (Before building, you should have to run `make update-api`.) and downloaded compiled aosp image to the laptop.

```
bash-3.2$ scp -r <our-server>:~/aosp/out/target/product/generic_x86/* \
 /Volumes/android/out/target/product/generic_x86/
```

Then, we started emeulator by following command.

```
bash-3.2$ source build/envsetup.sh
bash-3.2$ lunch aosp_x86-eng
bash-3.2$ export QEMU_AUDIO_DRV=none
bash-3.2$ emulator -memory 4096 -verbose -show-kernel -gpu host -partition-size 1024\
  -skindir ~/Library/Android/sdk/platforms/android-27/skins/ -skin WVGA800
```

5 How to Use PinkDroid

5.1 API Key Issue

First, you have to issue a API key for Google Cloud Vision API. Detailed instruction is on [1].

5.2 API Key Setup

In our PinkVision library (`graphics/PinkVision.java`), you have to setup issued API key.

```
public class PinkVision {

    private static boolean DEBUG = false;

    private static final String API_KEY = "AIzaSyBvgtvwYFqQmy_lSqpqPkMZKaMkIEqx244";

    ...
```

5.3 Dictionary Setup

You can add some words to be filtered in PinkDictionary library (`graphics/PinkDictionary.java`).

```
public class PinkDictionary {

    private static final String[] BAD_WORDS = { "kaist", "KAIST", "Wonsup", "Nak", "Bilgehan" };

    ...
```

5.4 Configuration Option

In each code, there are static variables in there. Some of them are used to configure PinkVision library.

5.4.1 PinkVision

- `DEBUG`: Send all requests to debug server (mockup Cloud Vision Server, codes are attached below).
- `MAX_DIMENSION`: Scale down a bitmap to this value before sending.
- `FILTER_LEVEL`: Minimum Cloud Vision filter level.
- `LOADING_IMAGE_BASE64`: Base64 encoded loading image.
- `PINKDROID_BASE64`: Base64 encoded PinkDroid image.

If you want to use debug server, you should set a hard coded server URL.

```
    private HttpURLConnection getConnection_debug() throws IOException {
        Log.i("PinkVision", "new DEBUG getConnection");
        URL url = new URL("<YOUR-DEBUG-SERVER-URL>");

        ...
```

5.4.2 Canvas

- `CV_MINIMUM`: If image is larger than this value, it is sent to Cloud Vision server.
- `mEnablePinkVision`: Let it `false`. If this value is true, all Canvas instances try to send images to Cloud Vision server whether it is UI canvas or not.
- `mEnableBlocking`: Let it `false`. If this value is true, PinkDroid use UI blocked filtering.
- `mTrust`: Let it `true`. If this value is false, PinkDroid do not use trust model.

5.5 Permissions

Our `patch` is `git diff` of `framework/base`. However, we not only edited `framework/base`. To give `INTERNET` permission we also edited prebuilt app source codes. So, you have to set `INTERNET` permission on `SystemUI` and `Launcher`. If you are going to use another app, you should have to add `INTERNET` permission on it.

5.5.1 Laucher2

Path: `<aosp-root>/packages/apps/Launcher2`

```
pusnow@blueberry:~/aosp/packages/apps/Launcher2$ git diff
diff --git a/AndroidManifest.xml b/AndroidManifest.xml
index 4260781..d246ddb 100644
--- a/AndroidManifest.xml
+++ b/AndroidManifest.xml
@@ -22,7 +22,7 @@
     package="com.android.launcher">

     <original-package android:name="com.android.launcher2" />
-
+    <uses-permission android:name="android.permission.INTERNET" />
     <permission
         android:name="com.android.launcher.permission.PRELOAD_WORKSPACE"
         android:permissionGroup="android.permission-group.SYSTEM_TOOLS"
```

5.5.2 Launcher3

Path: `<aosp-root>/packages/apps/Launcher3`

```
pusnow@blueberry:~/aosp/packages/apps/Launcher3$ git diff
diff --git a/AndroidManifest.xml b/AndroidManifest.xml
index 6c5990d..81e4f01 100644
--- a/AndroidManifest.xml
+++ b/AndroidManifest.xml
@@ -25,12 +25,13 @@
     Manifest entries specific to Launcher3. This is merged with AndroidManifest-common.xml.
     Refer comments around specific entries on how to extend individual components.
     -->
```

```
-
+
     <!--
     Permissions required for read/write access to the workspace data. These permission name
     should not conflict with that defined in other apps, as such an app should embed its package
     name in the permissions. eq com.mypackage.permission.READ_SETTINGS
     -->
+    <uses-permission android:name="android.permission.INTERNET" />
     <permission
         android:name="com.android.launcher3.permission.READ_SETTINGS"
         android:permissionGroup="android.permission-group.SYSTEM_TOOLS"
```

5.5.3 SystemUI

This package resides in `framework/base`.

5.6 Mockup Cloud Vision Server

Mockup server is developed by Python 3.6 and flask. You can use it simply by following command.

```
bash-3.2$ python3 run.py
```

`run.py` is like below.

```
from flask import Flask, request
from flask import jsonify
import base64
app = Flask(__name__)


counter = 0

@app.route('/', methods=["GET", "POST"])
def hello_world():
    global counter

    result = {
        "responses": [{
            "safeSearchAnnotation": {
                "adult": "VERY_LIKELY",
                "spoof": "VERY_LIKELY",
                "medical": "UNLIKELY",
                "violence": "UNLIKELY",
                "racy": "VERY_UNLIKELY",
            }
        }]
    }

    rq = request.get_json()
    for r in rq['requests']:
        counter+=1
        with open(str(counter) + ".png", "wb") as img:
            img.write(base64.b64decode(r['image']['content']))
```

```
    return jsonify(result)


if __name__ == '__main__':
    app.run(host= '0.0.0.0')
```

6 Note

Because we did many trials and auto-format our source codes, diff version of source codes can be unreadable(`git diff` of `framework/base`). So, we also attached file version of our source codes.

References

[1] Google cloud vision api documentation: *https://cloud.google.com/vision/docs/*.

[2] How to build android roms on ubuntu 16.04: 2018. *https://www.digitalocean.com/community/tutorials/how-to-build-android-roms-on-ubuntu-16-04*.

[3] Setting up a mac os build environment: 2018. *https://source.android.com/setup/build/initializing#setting-up-a-mac-os-x-build-environment*.