Project Report

Amit Prakhar Pandey

Title

Anti-Virus for Detecting Malicious Applications

Abstract

Non-consensual retrieval of user data is becoming an ever-growing concern with every passing year. Despite all the attention towards it, a huge number of people fall victim to these attacks. Many people do not consider this to be a big problem claiming: If you’re doing no wrong, there should be nothing to hide. That however, is incorrect as the collected data can be used in a number of ways to harm an individual, society or a nation without it being inappropriate in nature. This paper discusses the details of an application developed to help protect against a malware of that sort. The application user can be notified of a potential threat before having to use it to remove the malware before it can act.

Introduction

We are all familiar with the pop-up that appears at the top of the screen when we download a new app and it requests certain permissions. These could be to access your camera, microphone, GPS, contacts, gallery etc. And sometimes, it’s necessary to allow these apps, for them to get their work done. How would you post a picture onto Instagram if didn’t have access to your photos, right? But there can be a more sinister side to these permissions: Some apps don't bother asking for your consent at all, turning your device into a pocket spy, loaded with cameras and microphones at the ready.

Back in 2018, for example, over 250 apps across the App Store and Google Play market were listening in for background audio through smartphone microphones, allowing the apps to figure out what you watch or listen to in order to serve up better targeted advertisements. And then, of course, there's the long-standing conspiracy theory that our smartphones are actively eavesdropping on us.

So, for our project we opted to implement a solution to this problem by developing an application that would notify the user of apps with malicious intent. A proper well-built app could benefit a lot of people. Parents would feel much safer giving their children tablets or phones to play with. In fact, all people would find an app like this to be quite a life-saver as this is a serious privacy concern.

Project Details

Our application has the following main functionalities:

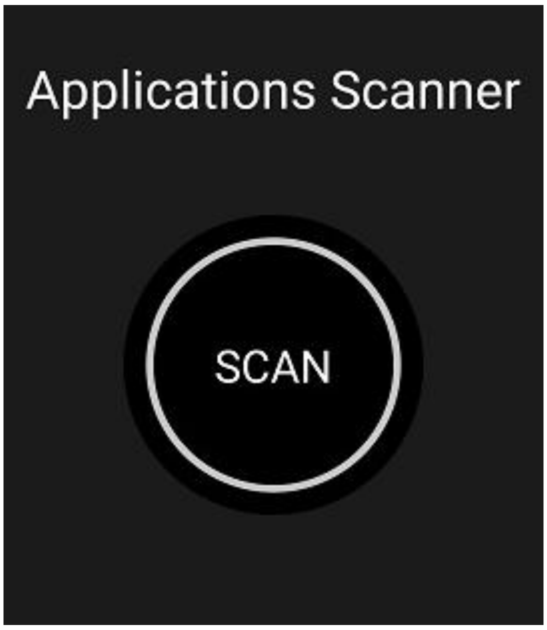
- It will scan through all other apps and identify any other applications that have been downloaded from sources that can not be trusted (APK’s).
- It will also allow the user to see which apps are using camera and microphone in a way that could potentially be harmful.

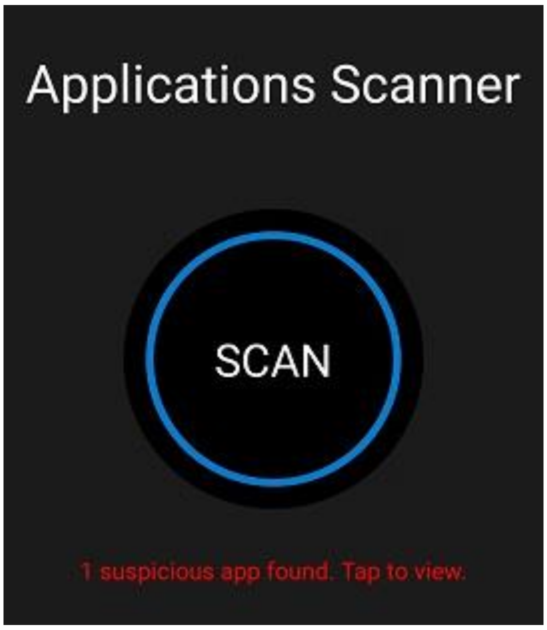
Implementation

The application has a very basic interface. It implements the above two functionalities and the procedure looks something like this:

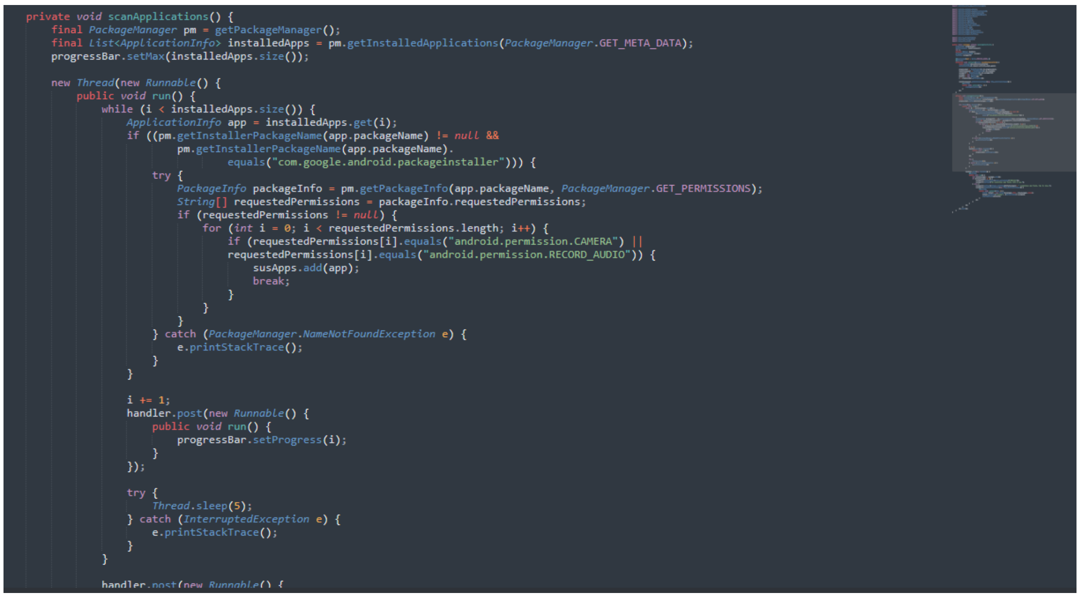
Step 1:

The user clicks on scan and the app tests all other applications to show user which of them are downloaded from unreliable sources.





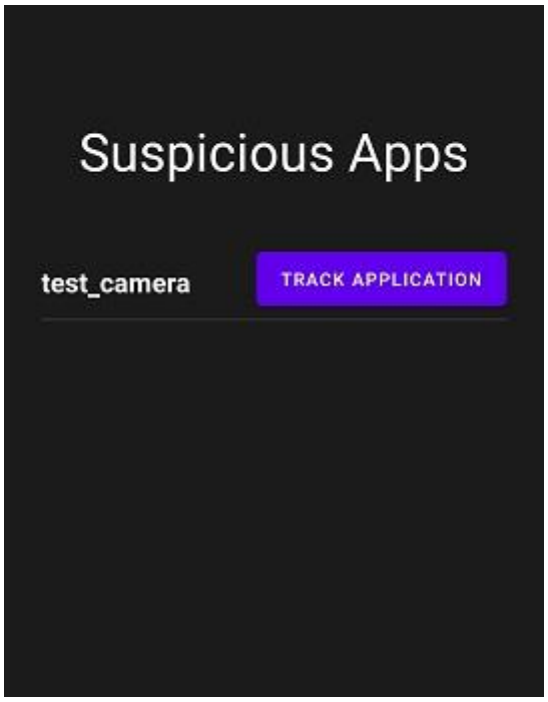
The scan application function in the file “scanApps.java” implements this functionality. The UI is managed seperatel in the .xml files.

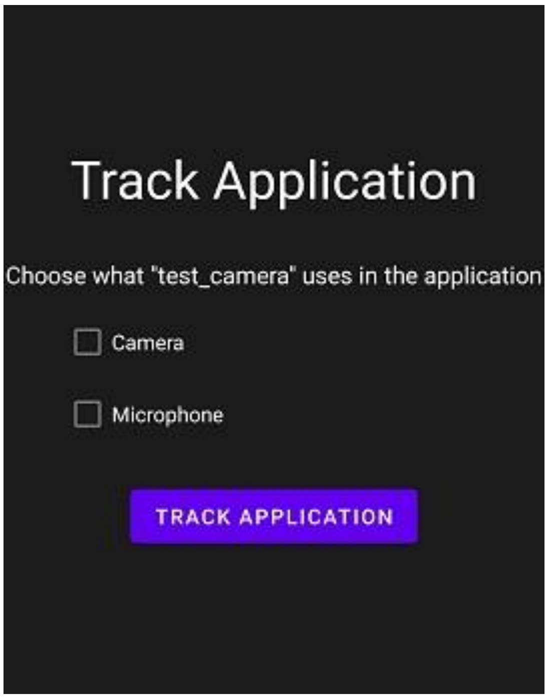


```java
private void scanApplications() {
    final PackageManager pm = getPackageManager();
    final List<ApplicationInfo> installedApps = pm.getInstalledApplications(PackageManager.GET_META_DATA);
    progressBar.setMax(installedApps.size());

    new Thread(new Runnable() {
        public void run() {
            while (i < installedApps.size()) {
                ApplicationInfo app = installedApps.get(i);
                if ((pm.getInstallerPackageName(app.packageName) != null &&
                        pm.getInstallerPackageName(app.packageName).
                                equals("com.google.android.packageinstaller"))) {
                    try {
                        PackageInfo packageInfo = pm.getPackageInfo(app.packageName, PackageManager.GET_PERMISSIONS);
                        String[] requestedPermissions = packageInfo.requestedPermissions;
                        if (requestedPermissions != null) {
                            for (int i = 0; i < requestedPermissions.length; i++) {
                                if (requestedPermissions[i].equals("android.permission.CAMERA") ||
                                requestedPermissions[i].equals("android.permission.RECORD_AUDIO")) {
                                    susApps.add(app);
                                    break;
                                }
                            }
                        }
                    } catch (PackageManager.NameNotFoundException e) {
                        e.printStackTrace();
                    }
                }

                i += 1;
                handler.post(new Runnable() {
                    public void run() {
                        progressBar.setProgress(i);
                    }
                });

                try {
                    Thread.sleep(5);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }

            handler.post(new Runnable() {
```

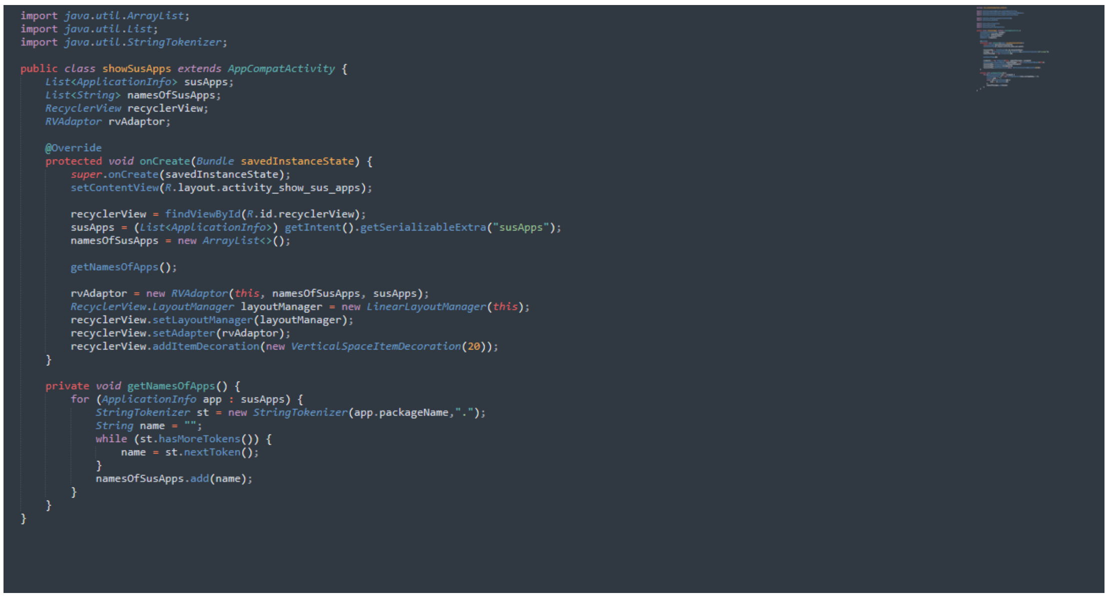
Step 2:

The user is then given the option to select among those suspicious apps, which one to track. The next screen asks the user to select the specific activity the app actually needs for proper functioning. All the ones that remain unticked will be checked.





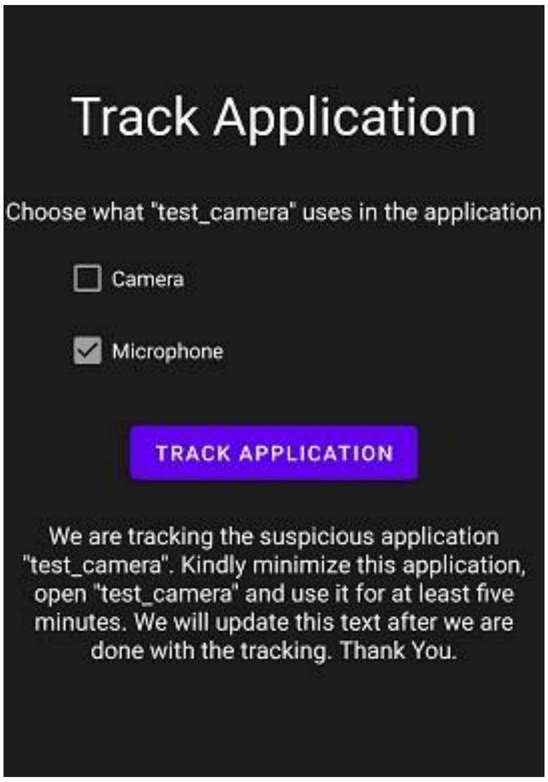
The showSusApps class stores and displays all applications downloaded from sources that cannot be trusted.



```java
import java.util.ArrayList;
import java.util.List;
import java.util.StringTokenizer;

public class showSusApps extends AppCompatActivity {
    List<ApplicationInfo> susApps;
    List<String> namesOfSusApps;
    RecyclerView recyclerView;
    RVAdaptor rvAdaptor;

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

        recyclerView = findViewById(R.id.recyclerView);
        susApps = (List<ApplicationInfo>) getIntent().getSerializableExtra("susApps");
        namesOfSusApps = new ArrayList<>();

        getNamesOfApps();

        rvAdaptor = new RVAdaptor(this, namesOfSusApps, susApps);
        RecyclerView.LayoutManager layoutManager = new LinearLayoutManager(this);
        recyclerView.setLayoutManager(layoutManager);
        recyclerView.setAdapter(rvAdaptor);
        recyclerView.addItemDecoration(new VerticalSpaceItemDecoration(20));
    }

    private void getNamesOfApps() {
        for (ApplicationInfo app : susApps) {
            StringTokenizer st = new StringTokenizer(app.packageName,".");
            String name = "";
            while (st.hasMoreTokens()) {
                name = st.nextToken();
            }
            namesOfSusApps.add(name);
        }
    }
}
```

Step 3:

After selecting and clicking “Track Application”, the user is given a set of guidelines to follow and within five minutes they are notified of the threat (if any).



Track Application

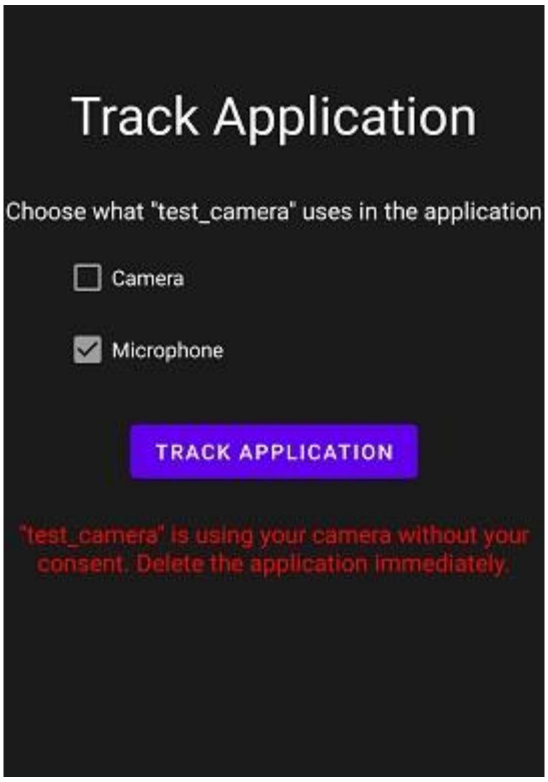
Choose what "test_camera" uses in the application

☐ Camera

☑ Microphone

TRACK APPLICATION

We are tracking the suspicious application "test_camera". Kindly minimize this application, open "test_camera" and use it for at least five minutes. We will update this text after we are done with the tracking. Thank You.



Track Application

Choose what "test_camera" uses in the application

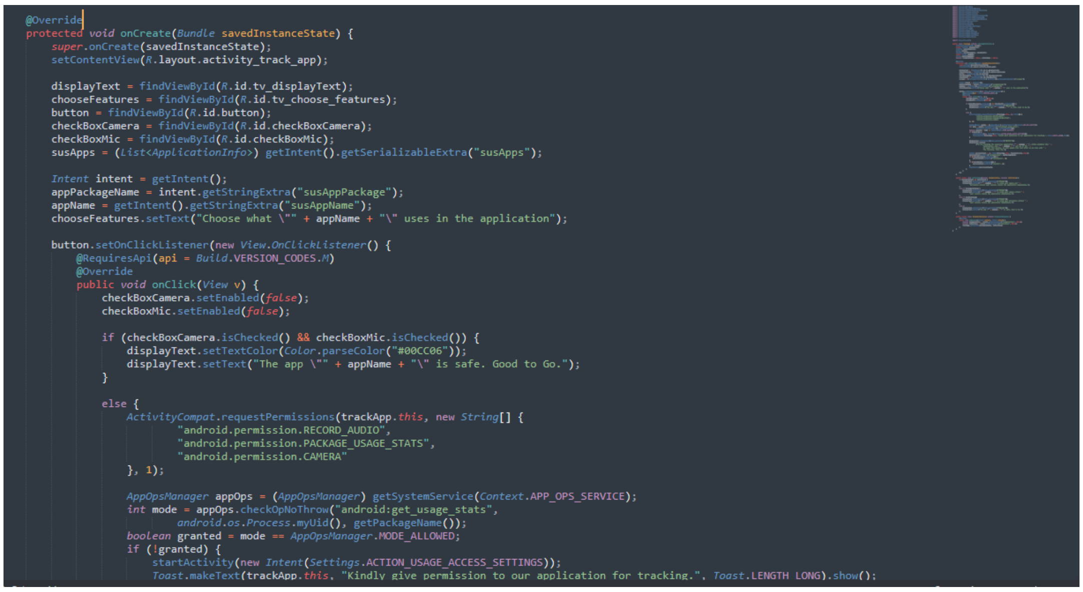
☐ Camera

☑ Microphone

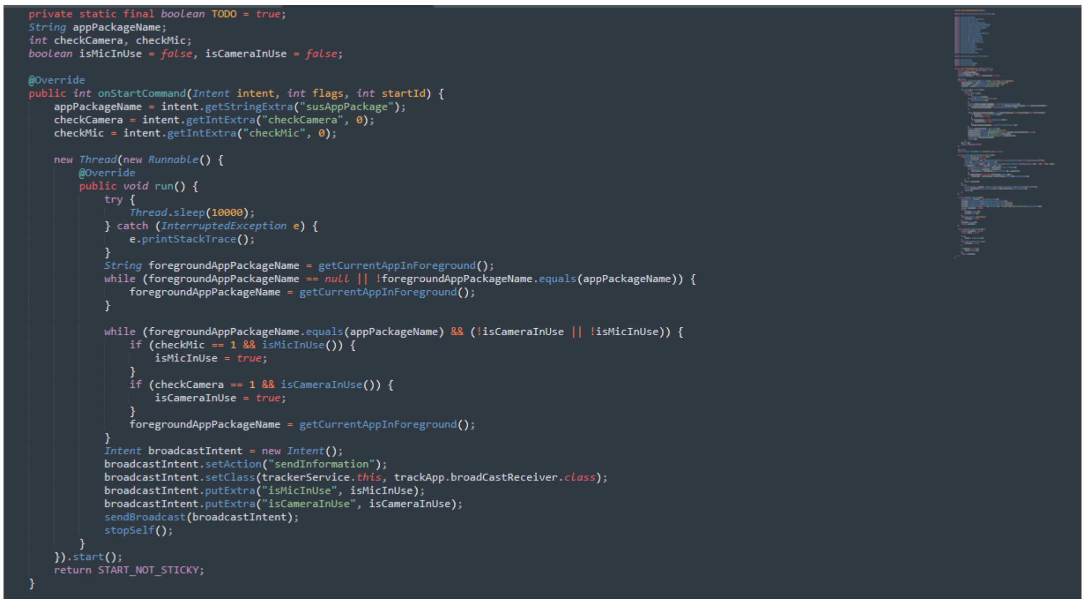
TRACK APPLICATION

"test_camera" is using your camera without your consent. Delete the application immediately.

This functionality has been implemented in two classes: “trackApp” and “trackServices”.



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

    displayText = findViewById(R.id.tv_displayText);
    chooseFeatures = findViewById(R.id.tv_choose_features);
    button = findViewById(R.id.button);
    checkBoxCamera = findViewById(R.id.checkBoxCamera);
    checkBoxMic = findViewById(R.id.checkBoxMic);
    susApps = (List<ApplicationInfo>) getIntent().getSerializableExtra("susApps");

    Intent intent = getIntent();
    appPackageName = intent.getStringExtra("susAppPackage");
    appName = getIntent().getStringExtra("susAppName");
    chooseFeatures.setText("Choose what \"" + appName + "\" uses in the application");

    button.setOnClickListener(new View.OnClickListener() {
        @RequiresApi(api = Build.VERSION_CODES.M)
        @Override
        public void onClick(View v) {
            checkBoxCamera.setEnabled(false);
            checkBoxMic.setEnabled(false);

            if (checkBoxCamera.isChecked() && checkBoxMic.isChecked()) {
                displayText.setTextColor(Color.parseColor("#00CC06"));
                displayText.setText("The app \"" + appName + "\" is safe. Good to Go.");
            }

            else {
                ActivityCompat.requestPermissions(trackApp.this, new String[] {
                        "android.permission.RECORD_AUDIO",
                        "android.permission.PACKAGE_USAGE_STATS",
                        "android.permission.CAMERA"
                }, 1);

                AppOpsManager appOps = (AppOpsManager) getSystemService(Context.APP_OPS_SERVICE);
                int mode = appOps.checkOpNoThrow("android:get_usage_stats",
                        android.os.Process.myUid(), getPackageName());
                boolean granted = mode == AppOpsManager.MODE_ALLOWED;
                if (!granted) {
                    startActivity(new Intent(Settings.ACTION_USAGE_ACCESS_SETTINGS));
                    Toast.makeText(trackApp.this, "Kindly give permission to our application for tracking.", Toast.LENGTH_LONG).show();
```



```java
private static final boolean TODO = true;
String appPackageName;
int checkCamera, checkMic;
boolean isMicInUse = false, isCameraInUse = false;

@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    appPackageName = intent.getStringExtra("susAppPackage");
    checkCamera = intent.getIntExtra("checkCamera", 0);
    checkMic = intent.getIntExtra("checkMic", 0);

    new Thread(new Runnable() {
        @Override
        public void run() {
            try {
                Thread.sleep(10000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            String foregroundAppPackageName = getCurrentAppInForeground();
            while (foregroundAppPackageName == null || !foregroundAppPackageName.equals(appPackageName)) {
                foregroundAppPackageName = getCurrentAppInForeground();
            }

            while (foregroundAppPackageName.equals(appPackageName) && (!isCameraInUse || !isMicInUse)) {
                if (checkMic == 1 && isMicInUse()) {
                    isMicInUse = true;
                }
                if (checkCamera == 1 && isCameraInUse()) {
                    isCameraInUse = true;
                }
                foregroundAppPackageName = getCurrentAppInForeground();
            }
            Intent broadcastIntent = new Intent();
            broadcastIntent.setAction("sendInformation");
            broadcastIntent.setClass(trackerService.this, trackApp.broadCastReceiver.class);
            broadcastIntent.putExtra("isMicInUse", isMicInUse);
            broadcastIntent.putExtra("isCameraInUse", isCameraInUse);
            sendBroadcast(broadcastIntent);
            stopSelf();
        }
    }).start();
    return START_NOT_STICKY;
}
```

Security Analysis

There were two main categories for the choice of project, i.e., a malware implementation or a defensive implementation. For our project, we decided to opt for the defensive implementation. The real-world problem that our project addresses is the non-consensual retrieval of user data. This is a very common problem when downloading applications using APK's. Which is why it is advised to download apps from trusted sources. However, there is a huge number of people who fall prey to these types of attacks. There have been many cases of fraudulent applications that obtain and use data without the users having agreed to it. In fact, in most cases people are not even aware that they're microphone or cameras are being used in applications where it's not needed. Some people may even ignore it while knowing about it, not realizing the potential security threat. It is also agreed upon that some nations create applications with these malwares to further some political agenda.

Keeping this in mind, we developed an app that runs a check on all applications on an android device and discovers which ones were downloaded through APK's. After that, it runs another scan when prompted to. The second scan lets the user know whether or not the phone's camera or microphone are being used while the user is doing something else on the phone such as, playing a game for instance. The user is then notified that the app could potentially be malicious and should not be trusted.

While this app is providing a solution to the problem, it is reliant on the user to run it and scan it. Some people may find that to be inconvenient and others may simply forget. A simple sure-fire way to avoid running into this problem is to download applications from trusted sources only. Only allow permissions that are actually needed, instead of ignoring prompts and hitting "Allow" on all, just to get them out of the way. Other than that, the app provided a simple and effective solution to the problem and could be used by anyone to check for this issue.