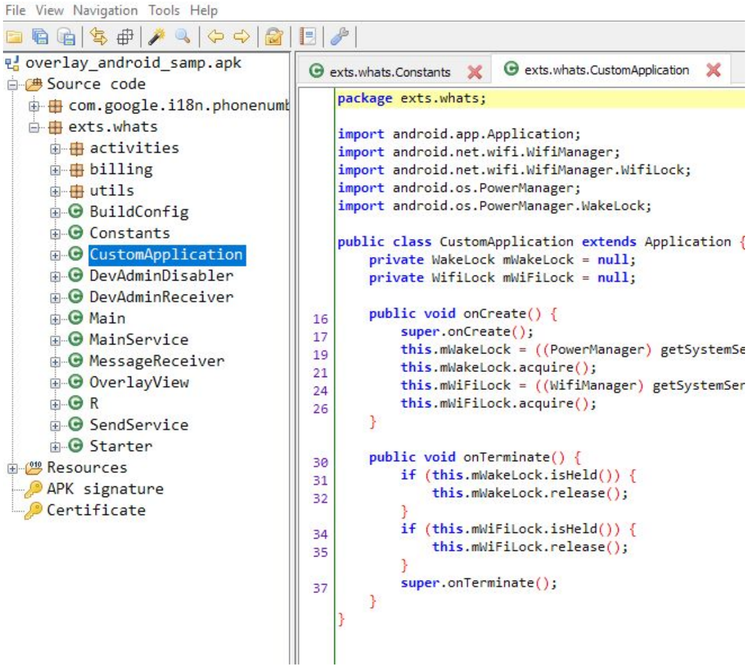
Documentation Task 2

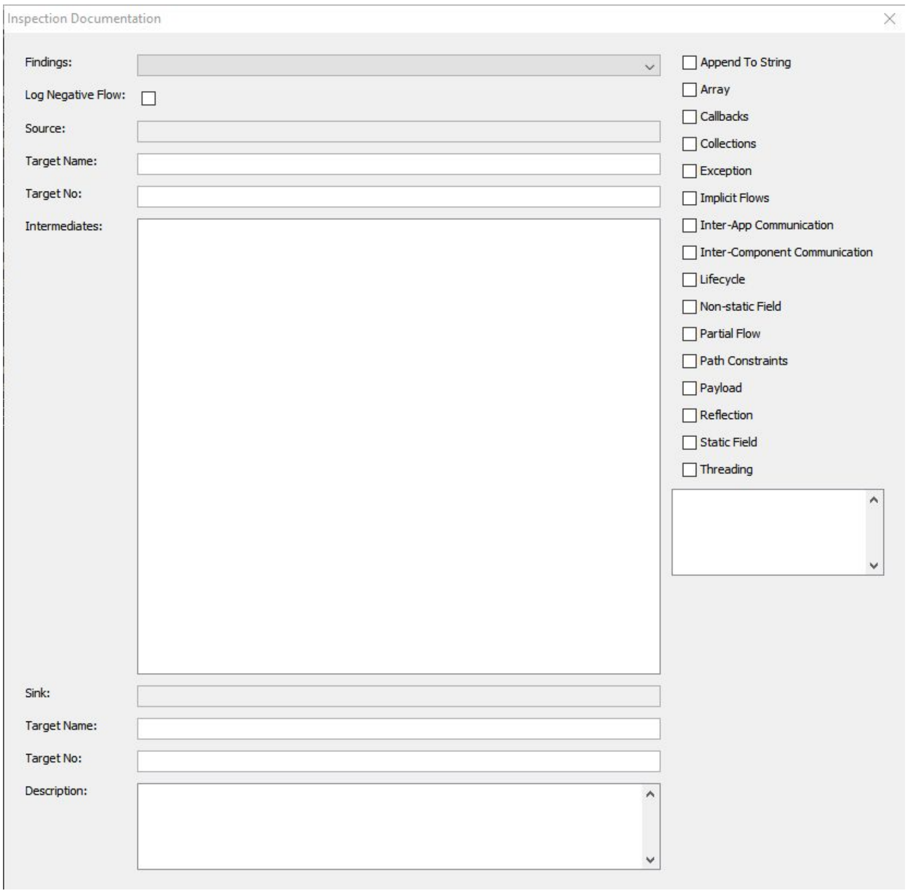
Imagine you are an expert in taint analysis and you want to create a new benchmark suite for evaluating your Android taint analysis tool. For this purpose, you have to document taint flows in an Android apk. Because we don’t have so much time in a user study for you to discover taint flows by yourself, you will be given taint flows found by us.

You are given an Android apk **overlay_android_samp.apk** and a tool called **Jadx-doc**. **Jadx-doc** is a decompiler which can decompile Android apks and display decompiled source code in a window as shown below:



```
package exts.whats;

import android.app.Application;
import android.net.wifi.WifiManager;
import android.net.wifi.WifiManager.WifiLock;
import android.os.PowerManager;
import android.os.PowerManager.WakeLock;

public class CustomApplication extends Application {
    private WakeLock mWakeLock = null;
    private WifiLock mWiFiLock = null;

    public void onCreate() {
        super.onCreate();
        this.mWakeLock = ((PowerManager) getSystemSe
        this.mWakeLock.acquire();
        this.mWiFiLock = ((WifiManager) getSystemSer
        this.mWiFiLock.acquire();
    }

    public void onTerminate() {
        if (this.mWakeLock.isHeld()) {
            this.mWakeLock.release();
        }
        if (this.mWiFiLock.isHeld()) {
            this.mWiFiLock.release();
        }
        super.onTerminate();
    }
}
```

Besides the main window above, it also shows you another window called inspection documentation. This window allows you to document taint flows.



Findings:

Log Negative Flow: □

Source:

Target Name:

Target No:

Intermediates:

Sink:

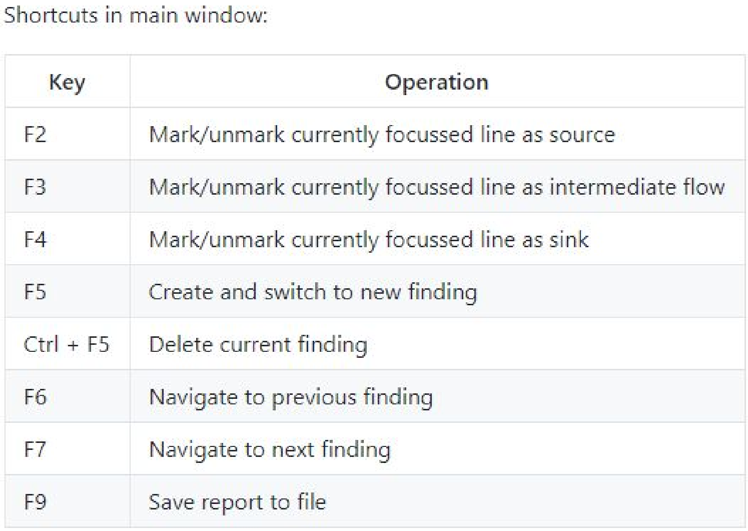
Target Name:

Target No:

Description:

□ Append To String
□ Array
□ Callbacks
□ Collections
□ Exception
□ Implicit Flows
□ Inter-App Communication
□ Inter-Component Communication
□ Lifecycle
□ Non-static Field
□ Partial Flow
□ Path Constraints
□ Payload
□ Reflection
□ Static Field
□ Threading

You can select source code directly in the main window with following shortcut keys:



Shortcuts in main window:

Key	Operation
F2	Mark/unmark currently focussed line as source
F3	Mark/unmark currently focussed line as intermediate flow
F4	Mark/unmark currently focussed line as sink
F5	Create and switch to new finding
Ctrl + F5	Delete current finding
F6	Navigate to previous finding
F7	Navigate to next finding
F9	Save report to file

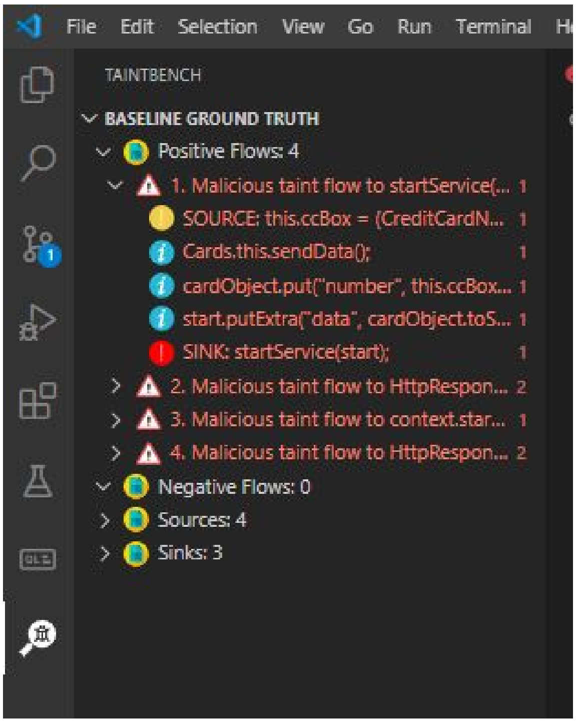
A common documentation process is as follows:

- Open apk file in Jadx-doc
- Add finding (press F5)
- Click a line to set focus
- Mark focussed line as source/intermediate/sink (press F2/F3/F4)
- If you mistakenly selected a source/intermediate/sink, simply unmark it (press F2/F3/F4)
- Click different line for next mark etc ...
- When done with current finding, add next finding with F5
- When done with all findings, save report (F9)

Please watch this tutorial with subtitles on: **https://youtu.be/rTLNIGXELS0**

Your Task:

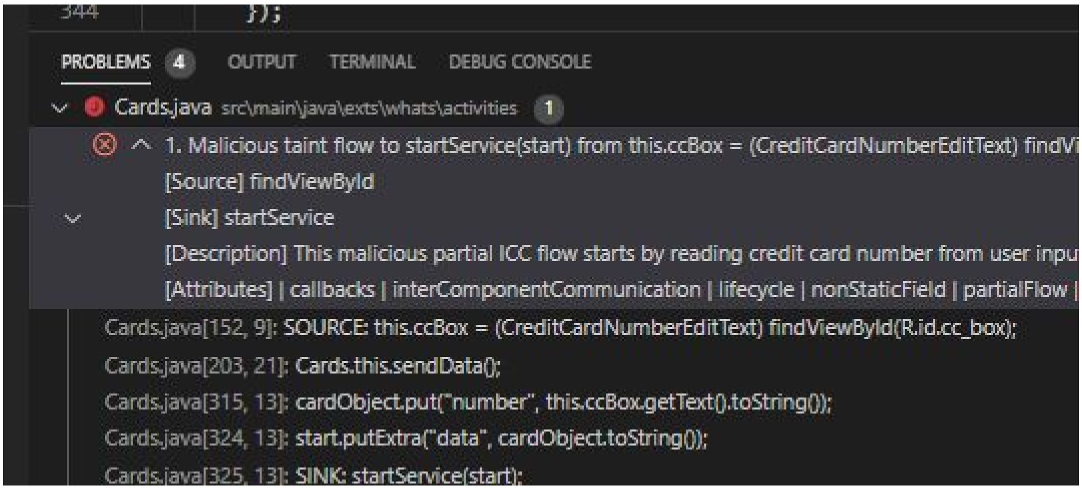
1. Open **overlay_android_samp.apk** in **Jadx-doc**.
2. Start Visual Studio Code, enable the extension TB-Viewer in Visual Studio Code if you have it disabled. Restart Visual Studio Code.
3. In Visual Studio Code, click **File >> Open Workspace >> Select myworkspace.code-workspace** in the folder **overlay_android_samp**
4. Click 🔍 on the side panel, you will be shown 4 taint flows in the following screenshot:



TAINTBENCH

- BASELINE GROUND TRUTH
  - Positive Flows: 4
    - 1. Malicious taint flow to startService(... 1
      - SOURCE: this.ccBox = (CreditCardN... 1
      - Cards.this.sendData(); 1
      - cardObject.put("number", this.ccBox... 1
      - start.putExtra("data", cardObject.toS... 1
      - SINK: startService(start); 1
    - 2. Malicious taint flow to HttpRespon... 2
    - 3. Malicious taint flow to context.star... 1
    - 4. Malicious taint flow to HttpRespon... 2
  - Negative Flows: 0
  - Sources: 4
  - Sinks: 3

5. All information you need about the taint flows can be found in the PROBLEMS view of Visual Studio Code as shown below:



```
Cards.java src\main\java\exts\whats\activities 1
  1. Malicious taint flow to startService(start) from this.ccBox = (CreditCardNumberEditText) findVi
     [Source] findViewById
     [Sink] startService
     [Description] This malicious partial ICC flow starts by reading credit card number from user inpu
     [Attributes] | callbacks | interComponentCommunication | lifecycle | nonStaticField | partialFlow |
  Cards.java[152, 9]: SOURCE: this.ccBox = (CreditCardNumberEditText) findViewById(R.id.cc_box);
  Cards.java[203, 21]: Cards.this.sendData();
  Cards.java[315, 13]: cardObject.put("number", this.ccBox.getText().toString());
  Cards.java[324, 13]: start.putExtra("data", cardObject.toString());
  Cards.java[325, 13]: SINK: startService(start);
```

6. Find the positions of the taint flows with FlowID **1 and 2** in **jadx-doc** and document them with **jadx-doc.**
7. Once you are done, save the report as **jadx-doc.json.**
8. **Tell me when you start!**
9. **Tell me when you finish!**
10. **Send me jadx-doc.json.**