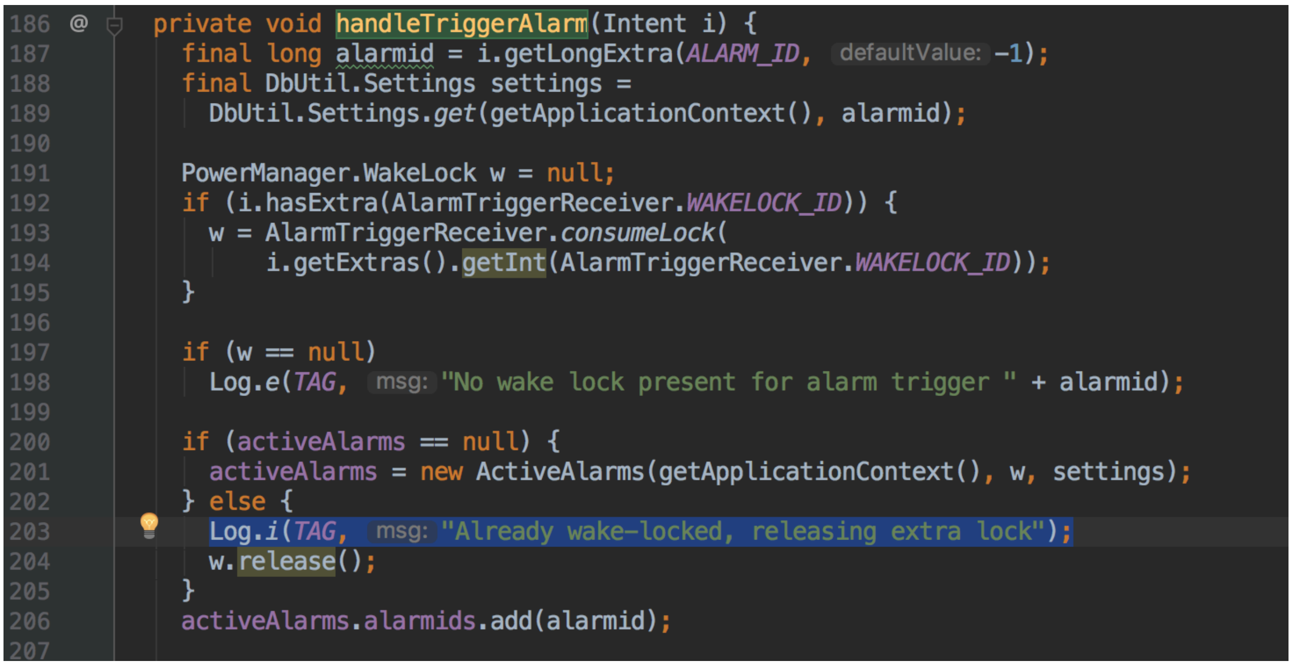
<u>Id : fd-65</u>

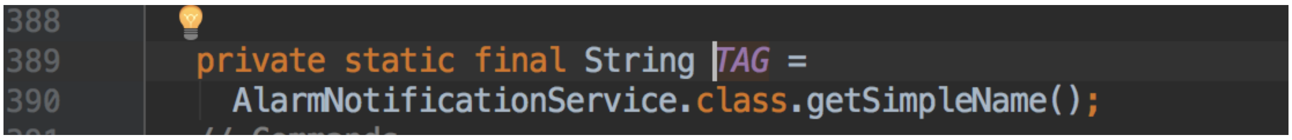
Result : False Positive

Sink statement : Line 203 in below screenshot is the sink statement.



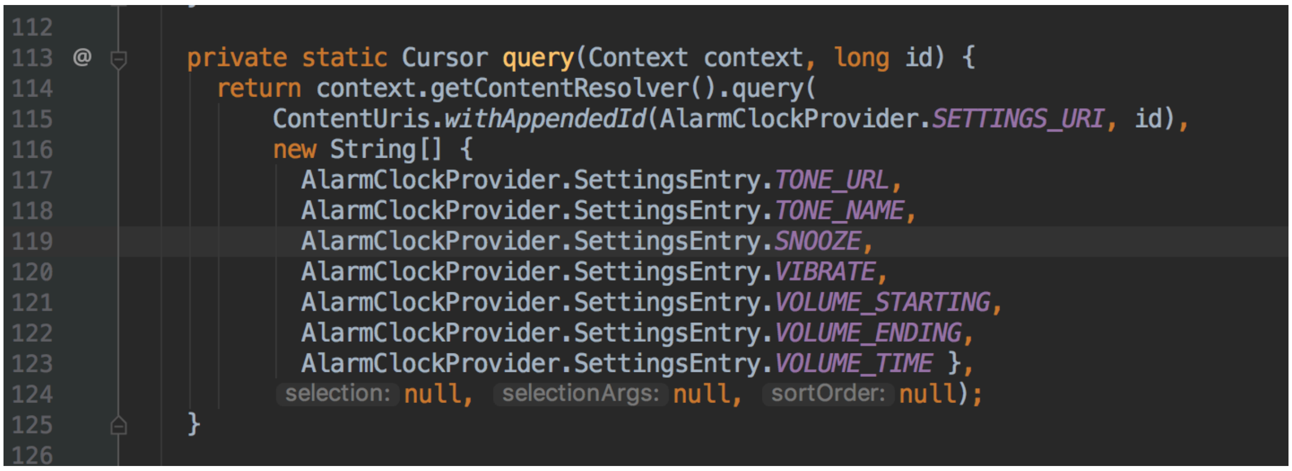
```
186 @   private void handleTriggerAlarm(Intent i) {
187       final long alarmid = i.getLongExtra(ALARM_ID,  defaultValue: -1);
188       final DbUtil.Settings settings =
189         DbUtil.Settings.get(getApplicationContext(), alarmid);
190
191       PowerManager.WakeLock w = null;
192       if (i.hasExtra(AlarmTriggerReceiver.WAKELOCK_ID)) {
193         w = AlarmTriggerReceiver.consumeLock(
194             i.getExtras().getInt(AlarmTriggerReceiver.WAKELOCK_ID));
195       }
196
197       if (w == null)
198         Log.e(TAG,  msg: "No wake lock present for alarm trigger " + alarmid);
199
200       if (activeAlarms == null) {
201         activeAlarms = new ActiveAlarms(getApplicationContext(), w, settings);
202       } else {
203         Log.i(TAG,  msg: "Already wake-locked, releasing extra lock");
204         w.release();
205       }
206       activeAlarms.alarmids.add(alarmid);
207
```

All the variables to the sink statement are constants.



```
388
389   private static final String TAG =
390     AlarmNotificationService.class.getSimpleName();
```

Source Statement : Line 114 in following screenshot



```
112
113 @   private static Cursor query(Context context, long id) {
114       return context.getContentResolver().query(
115           ContentUris.withAppendedId(AlarmClockProvider.SETTINGS_URI, id),
116           new String[] {
117             AlarmClockProvider.SettingsEntry.TONE_URL,
118             AlarmClockProvider.SettingsEntry.TONE_NAME,
119             AlarmClockProvider.SettingsEntry.SNOOZE,
120             AlarmClockProvider.SettingsEntry.VIBRATE,
121             AlarmClockProvider.SettingsEntry.VOLUME_STARTING,
122             AlarmClockProvider.SettingsEntry.VOLUME_ENDING,
123             AlarmClockProvider.SettingsEntry.VOLUME_TIME },
124        selection: null,  selectionArgs: null,  sortOrder: null);
125     }
126
```