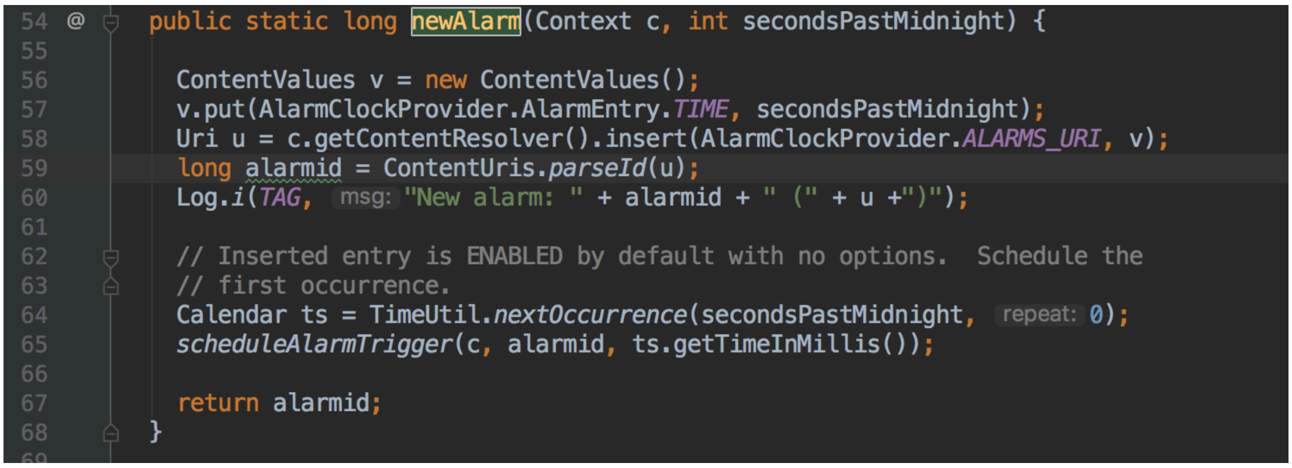
Id : ds-69

Result : False Positive

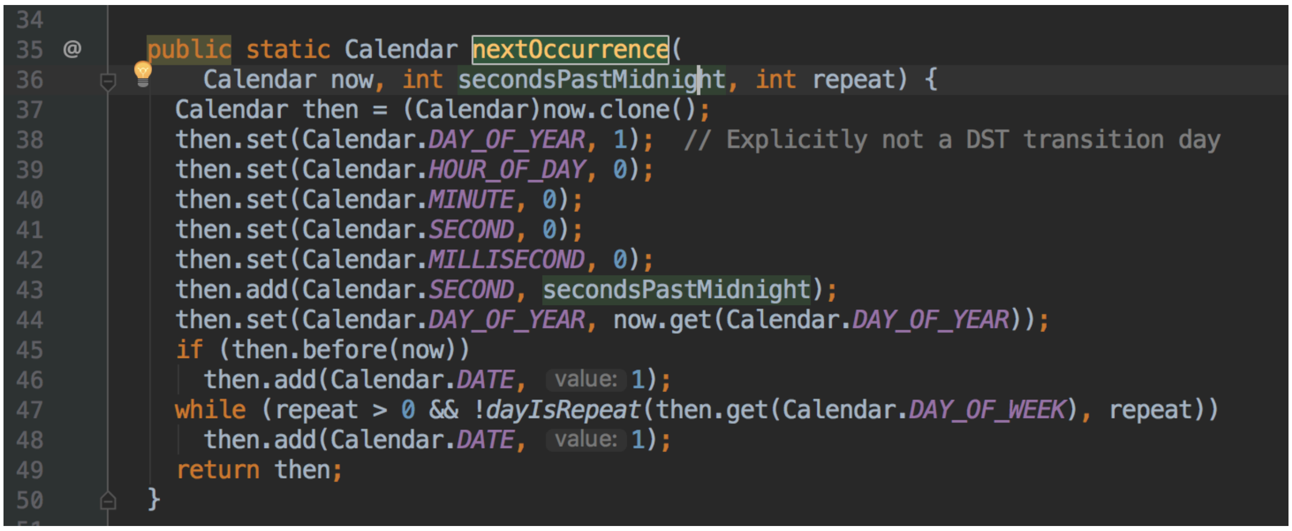
Sink statement : Line 60 in below screenshot is the sink statement.



```
54 @    public static long newAlarm(Context c, int secondsPastMidnight) {
55
56         ContentValues v = new ContentValues();
57         v.put(AlarmClockProvider.AlarmEntry.TIME, secondsPastMidnight);
58         Uri u = c.getContentResolver().insert(AlarmClockProvider.ALARMS_URI, v);
59         long alarmid = ContentUris.parseId(u);
60         Log.i(TAG,  msg: "New alarm: " + alarmid + " (" + u +")");
61
62         // Inserted entry is ENABLED by default with no options.  Schedule the
63         // first occurrence.
64         Calendar ts = TimeUtil.nextOccurrence(secondsPastMidnight,  repeat: 0);
65         scheduleAlarmTrigger(c, alarmid, ts.getTimeInMillis());
66
67         return alarmid;
68     }
```

Both u and alarmid variables to the sink statement are generated at line 58 and 59 respectively and hence are not coming from the given source statement.

Source Statement : Line 44,47 in following screenshot



```
34
35 @    public static Calendar nextOccurrence(
36             Calendar now, int secondsPastMidnight, int repeat) {
37         Calendar then = (Calendar)now.clone();
38         then.set(Calendar.DAY_OF_YEAR, 1);  // Explicitly not a DST transition day
39         then.set(Calendar.HOUR_OF_DAY, 0);
40         then.set(Calendar.MINUTE, 0);
41         then.set(Calendar.SECOND, 0);
42         then.set(Calendar.MILLISECOND, 0);
43         then.add(Calendar.SECOND, secondsPastMidnight);
44         then.set(Calendar.DAY_OF_YEAR, now.get(Calendar.DAY_OF_YEAR));
45         if (then.before(now))
46             then.add(Calendar.DATE,  value: 1);
47         while (repeat > 0 && !dayIsRepeat(then.get(Calendar.DAY_OF_WEEK), repeat))
48             then.add(Calendar.DATE,  value: 1);
49         return then;
50     }
```