Practical static analysis of context leaks in Android applications

Authors: Flavio Toffalini[1], Jun Sun[1], Martín Ochoa[2]

[1]Singapore University of Technology and Design, Singapore
[2]Department of Applied Mathematics and Computer Science, Universidad del Rosario, Colombia

Presented by: Dilver Huertas Guerrero
Master's student in systems and computer engineering
Universidad Nacional de Colombia

UNSECURELAB

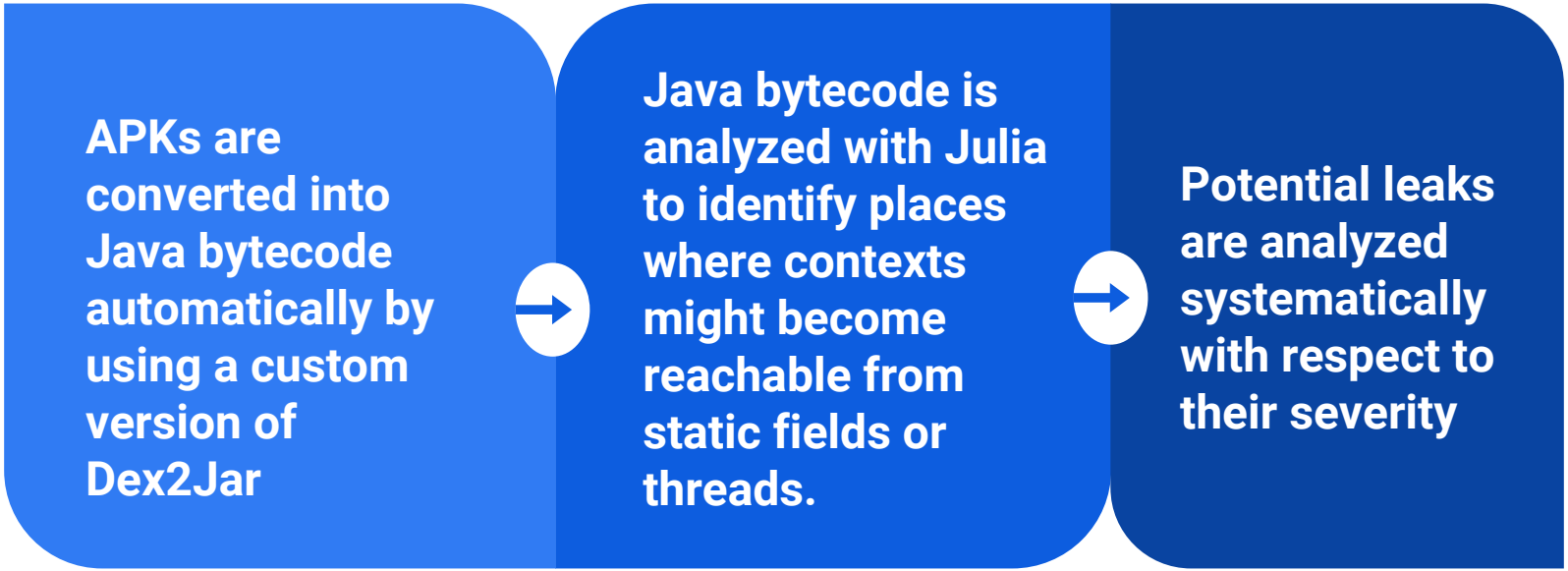
Research

Work focus

This work is focus on context leaks in Android native applications written in Java, shipped in APK format containing Dalvik bytecode. They develop, implement, and experiment with a static analysis method that identifies context leaks by analyzing the bytecode of the APK.

UNSECURELAB

Research method



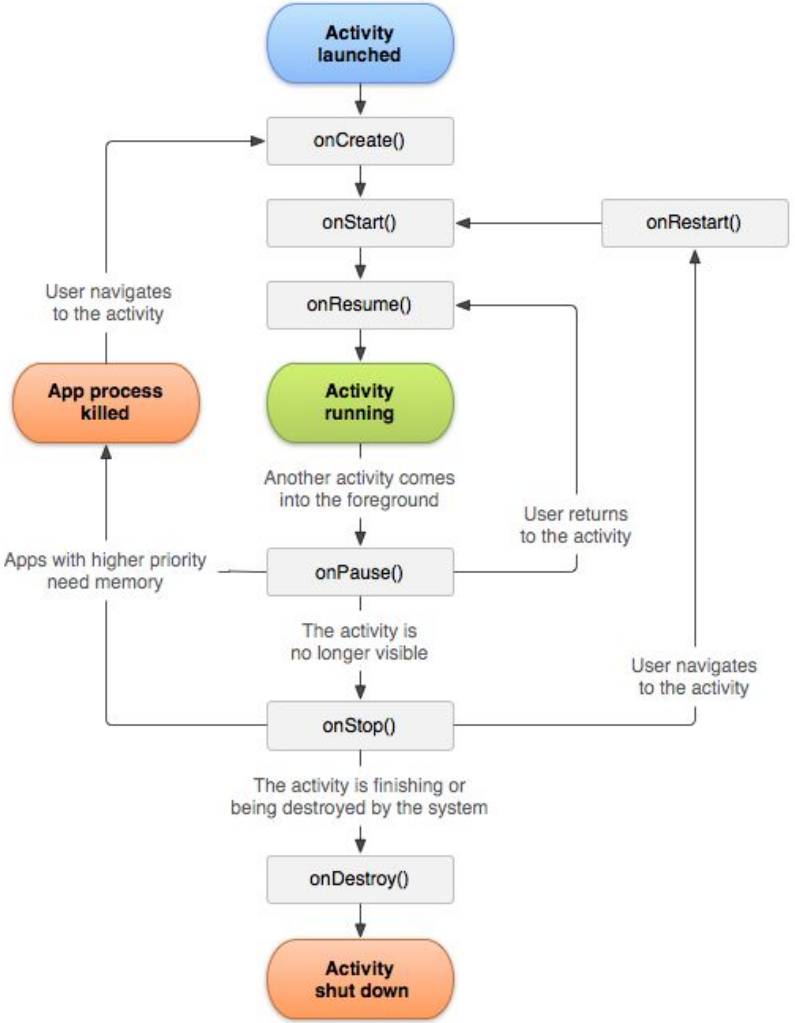
UNSECURELAB

Scope

They conduct extensive experiments by applying our method to 500 third-party widely used APKs from the Android market. The experiment results suggest that context leaks are potentially widespread in real applications, with various severity degrees.

UNSECURELAB

Android



UNSECURELAB

Android Contexts

	Application	Activity	Service	ContentProvider	BroadcastReceiver
Show a Dialog	NO	YES	NO	NO	NO
Start an Activity	NO[1]	YES	NO[1]	NO[1]	NO[1]
Layout Inflation	NO[2]	YES	NO[2]	NO[2]	NO[2]
Start a Service	YES	YES	YES	YES	YES
Bind to a Service	YES	YES	YES	YES	NO
Send a Broadcast	YES	YES	YES	YES	YES
Register BroadcastReceiver	YES	YES	YES	YES	NO[3]
Load Resource Values	YES	YES	YES	YES	YES

Contexts should not be made reachable from static fields or threads, which are roots of nongarbage collectable data.

UNSECURELAB

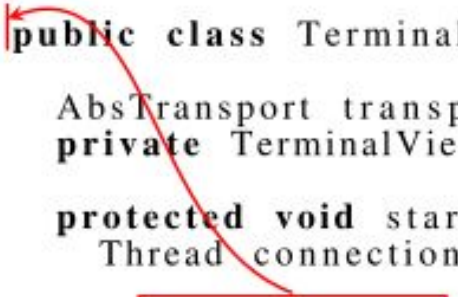
Contexts leaks in Android

A context leak occurs in Android if and only if a context has reached its life cycle end but is still reachable from a running thread or from a static field.
The three typical origins of a context leak in Android are: A thread that reaches a context, a static field that reaches a context, and a system callback that reaches a context.

UNSECURELAB

A thread reaches a context

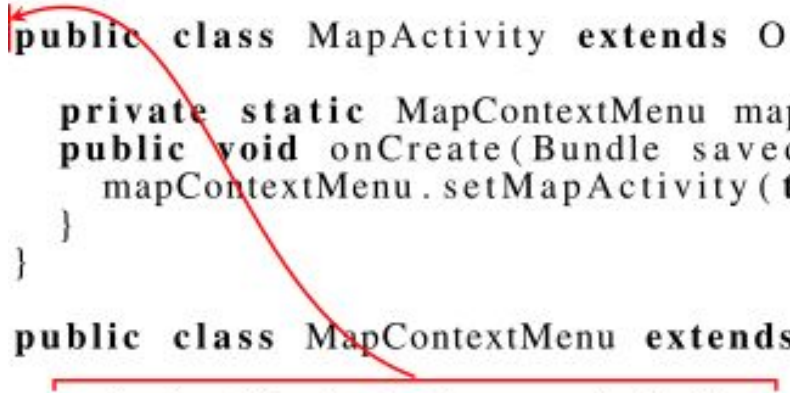
```
public class TerminalBridge implements VDUDisplay {

  AbsTransport transport = ...;
  private TerminalView parent = null;

  protected void startConnection() {
    Thread connectionThread = new Thread(

      new Runnable() {
        public void run() {
          transport.connect();
        }
      });
    connectionThread.start();
  }

  public final synchronized void parentChanged(
                                 TerminalView parent) {
    this.parent = parent;
  }
}
```



UNSECURELAB

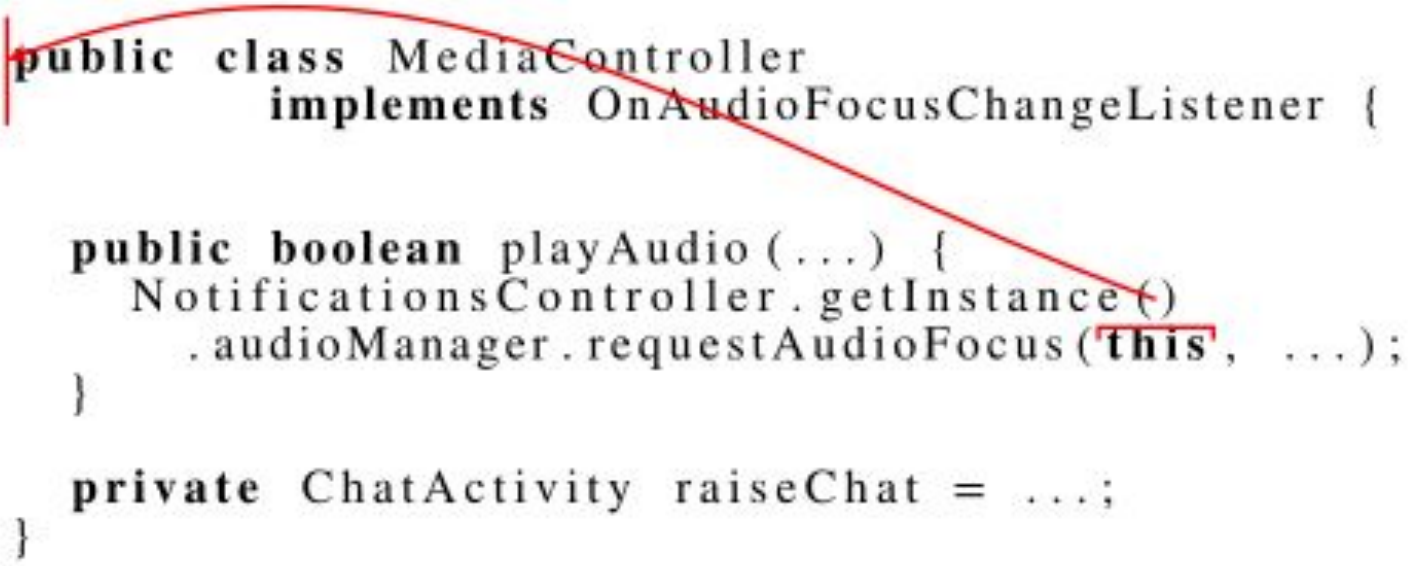
A static field reaches a context

```
public class MapActivity extends OsmandActionBarActivity implements ... {

   private static MapContextMenu mapContextMenu = new MapContextMenu();
   public void onCreate(Bundle savedInstanceState) {
      mapContextMenu.setMapActivity(this);
   }
}

public class MapContextMenu extends MenuTitleController implements ... {

   private MapActivity mapActivity;
   public void setMapActivity(MapActivity mapActivity) {
      this.mapActivity = mapActivity;
   }
}
```



UNSECURELAB

A system callback reaches a context



UNSECURELAB

Warning Classification and experiments

Classification of methods

- Linear methods contain no loops.
- Library interacting methods call library functions.
- Looping methods contain loops or recursion.
- File system methods call file system functions.
- Networking methods call networking functions.

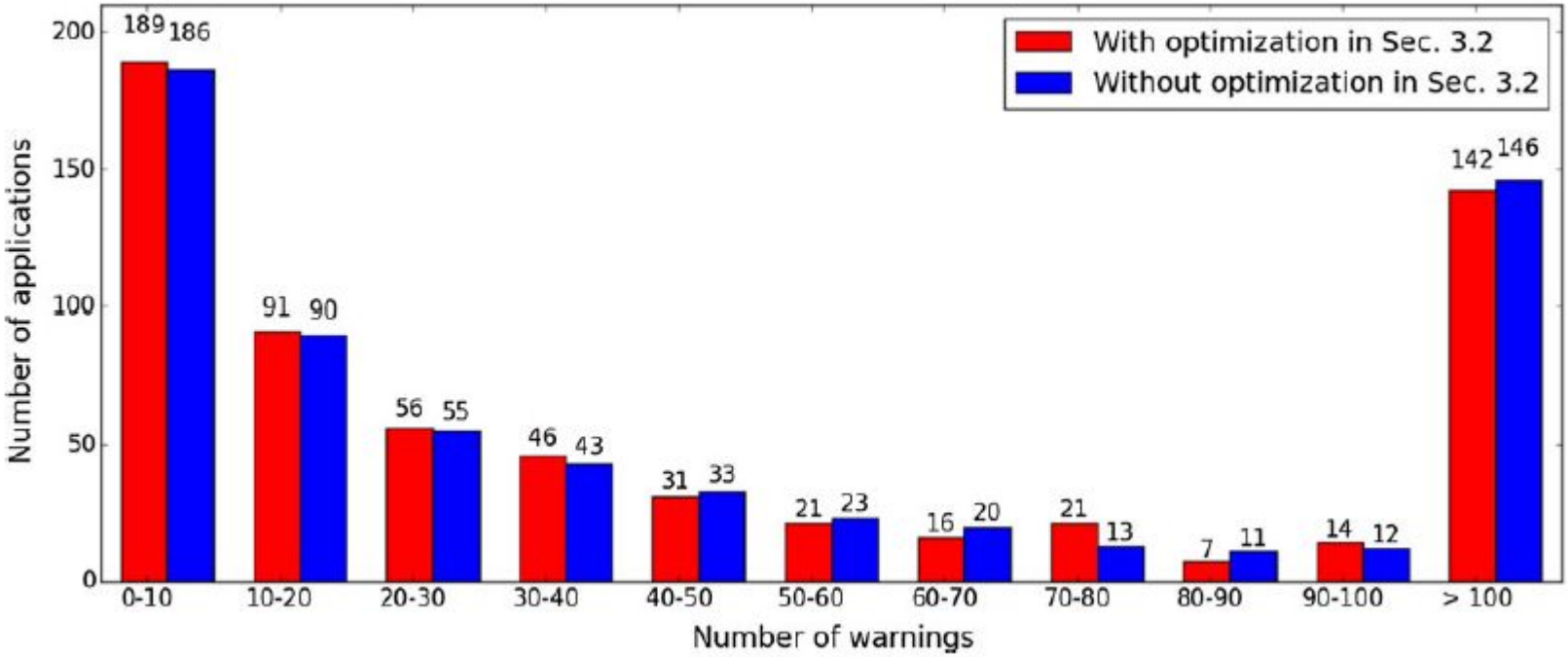
UNSECURELAB

Research questions

- Evaluate whether the target leaks are common in Android applications and whether the method scalable and efficient to identify them in real-world Android applications.
- Evaluate what is the likelihood of the method reporting false positives.
- Evaluate whether the leaks in practice are severe enough so that the effort is justified.

UNSECURELAB

RQ1



UNSECURELAB

RQ2

	Julia						Lint				Infer	
	threads		static		callbacks		threads		static		static	
	true	false	true	false	true	false	true	false	true	false	true	false
webTube	0	0(0)	1	0	0	0(0)	0	0(0)	1	0	0	0
aSQLiteManager	7	0(0)	1	0	0	1(0)	0	0(0)	1	0	0	0
ConnectBot	29	0(0)	0	3	0	3(0)	1	0(0)	0	1	0	0
OwnCloud	24	31(19)	0	6	1	2(0)	2	0(0)	0	1	0	1
Kiwix	7	0(0)	1	0	1	2(1)	1	0(0)	1	0	0	0
Firefox	213	70(12)	16	42	4	26(24)	0	0(0)	6	13	NA	NA
OsmAnd	260	29(2)	13	11	4	13(0)	2	0(0)	0	0	NA	NA
Telegram	237	9(0)	6	12	1	9(5)	0	0(0)	5	0	2	0

	Sum	Min	Max	Mean	SD
webTube	3	3	3	3	NA
aSQLiteManager	10	1	2	1	<1
ConnectBot	87	<1	9	3	3
OwnCloud	97	<1	37	4	8
Kiwix	501	<1	156	56	76
Firefox	668	<1	50	3	8

UNSECURELAB

RQ3

Type of leaked object	#	%
BroadcastReceiver	223	0.47%
Fragment	469	0.98%
Adapter	686	1.44%
ImageView	709	1.49%
Dialog	983	2.06%
Service	1100	2.31%
Context	1762	3.69%
Collections	1799	3.77%
Other interfaces	3574	7.49%
View	6851	14.37%
Android context container	7778	16.31%
Custom context container	10689	22.41%
Activity	11068	23.21%
Total:	47691	

UNSECURELAB

Conclusions

Conclusions

The experiment results show that the method performs better than Lint and Infer, because the technique allows them to identify a larger number of cases than only few syntactic patterns.
Results are more precise because we exclude those objects, which point only to ApplicationContext.
Due to the heuristics they adopt for efficiency and avoiding false positives, there might be leaks that they do not identify.

UNSECURELAB

Thank you!

Questions?

Dilver Huertas Guerrero
@dilverhuertas
djhuertasg@unal.edu.co