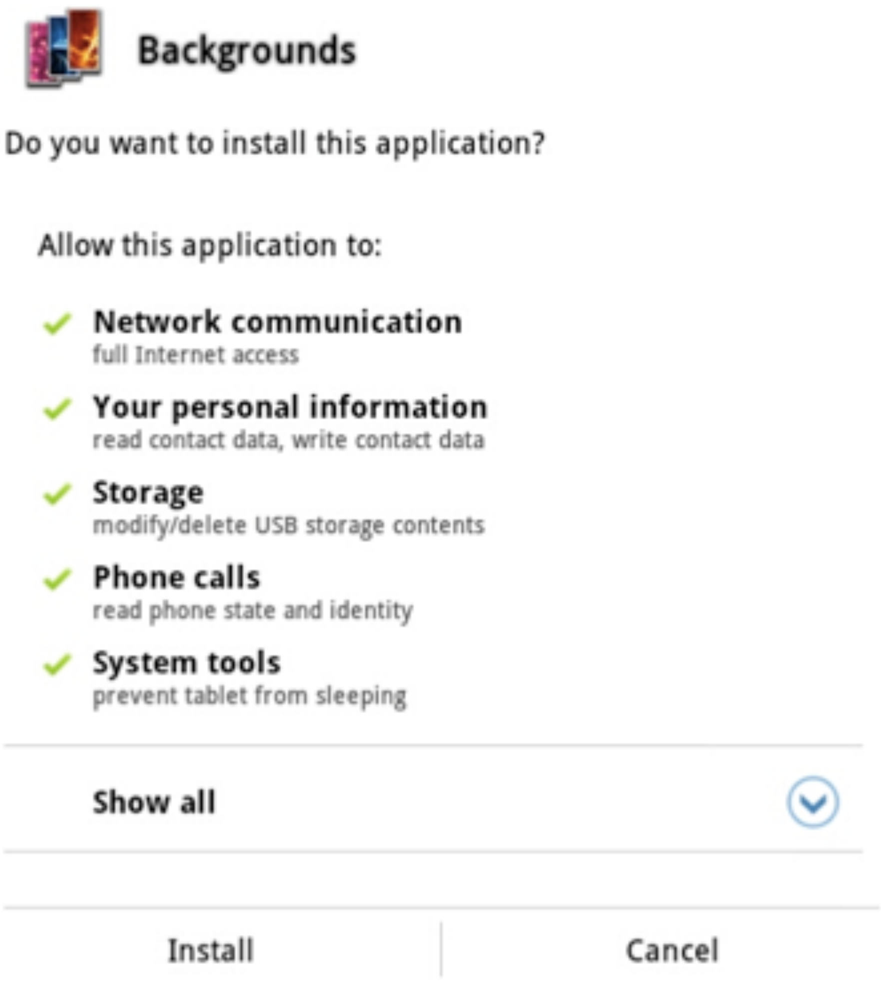
ScanDal: Static Analyzer for Detecting Privacy Leaks in Android Application

Jinyung Kim, Yongho Yoon, Kwangkeun Yi
Seoul National University
scandal@ropas.snu.ac.kr

Student talk, POPL 2013, Rome, Italy

Security Model of Android

- Check access controls on installation time
- App can use any service if it declared permission
- Cannot control leaks



Backgrounds

Do you want to install this application?

Allow this application to:

✓ **Network communication**
full Internet access

✓ **Your personal information**
read contact data, write contact data

✓ **Storage**
modify/delete USB storage contents

✓ **Phone calls**
read phone state and identity

✓ **System tools**
prevent tablet from sleeping

Malwares in Android

- Privacy leak
- Botnet
- Exploit (get root permission)
- Steal SMS
- Send SMS without user's approval ...

Yajin Zhou and Xuxian Jiang. Dissecting android malware: Characterization and evolution.
Security and Privacy, IEEE Symposium on, 0:95–109, 2012.

ScanDal

- Static analyzer based on abstract interpretation
- Scan Dalvik bytecode
- For now, focus on detecting privacy leak
  - Flow analysis, from source API to sink API
- MoST 2012 (workshop in IEEE S&P)

ScanDal +α

- Ongoing : performance tuning
  - Memory, time, alarms
- To do : extension
  - Cover ARM native code (JNI)
  - Extend to general malware detector

Performance

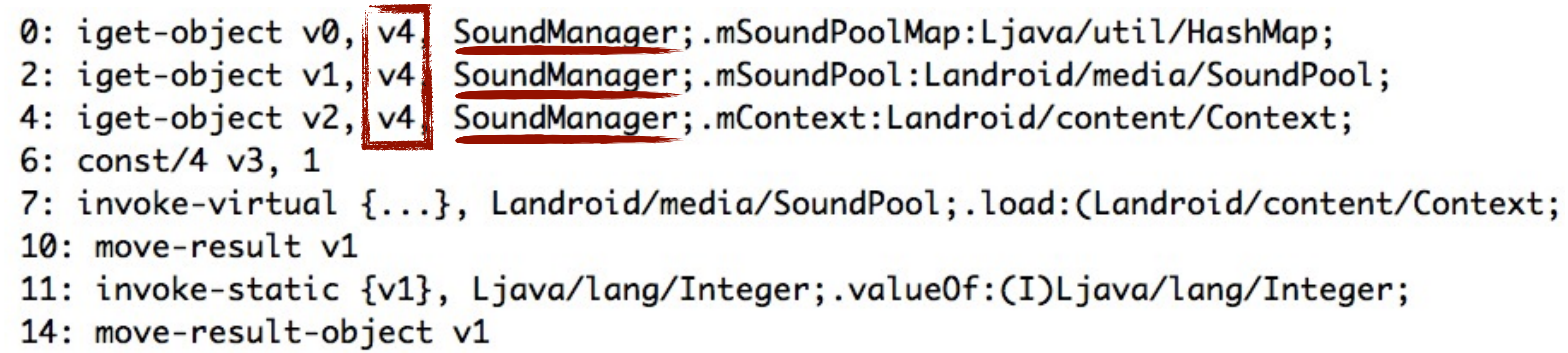
Name	MoST 2012			Recent		
	time(sec)	mem(MB)	#alarm	time(sec)	mem(MB)	#alarm
Kids Preschool Puzzle	1	62	29	1	29	6
Job Search	1	95	7	1	45	3
Kids Shapes	2	137	36	4	77	6
Kids ABC Phonics	3	109	30	1	47	6
Backgrounds HD Wallpapres	4	133	10	1	30	3
Bible Quotes	8	265	3	1	40	2
ES Task Manager	20	424	3	29	302	2
Multi Touch Paint	42	718	53	56	431	38
Adao File Manager	67	1143	14	116	819	1
(D-Day) The Day Before	225	2648	14	84	976	1
Kids Numbers and Math	559	176	29	1	41	6

X 0.3 X 0.5 X 0.3

Localization

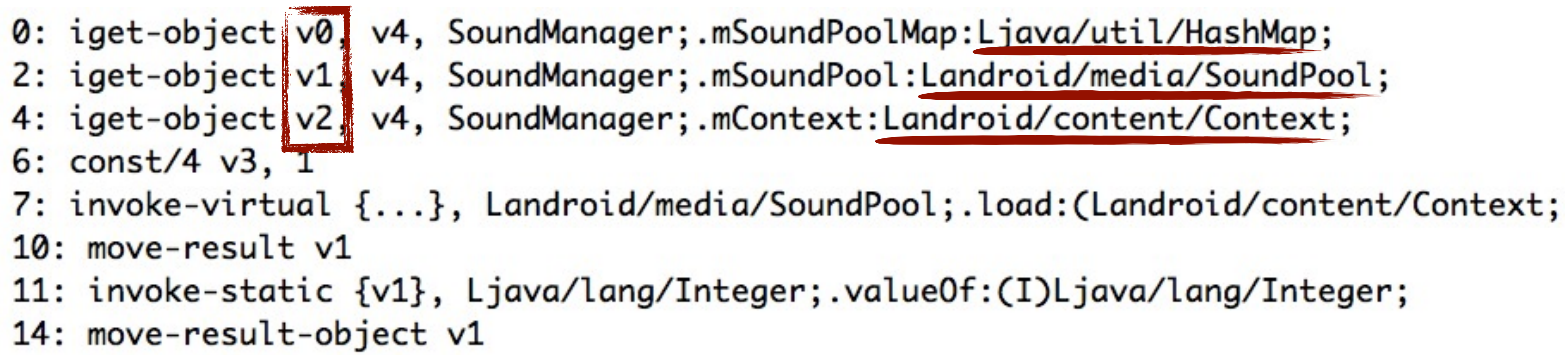
- Reachability-based localization
  - Abstract garbage collection
- Localization without pre-analysis

Pruning by type



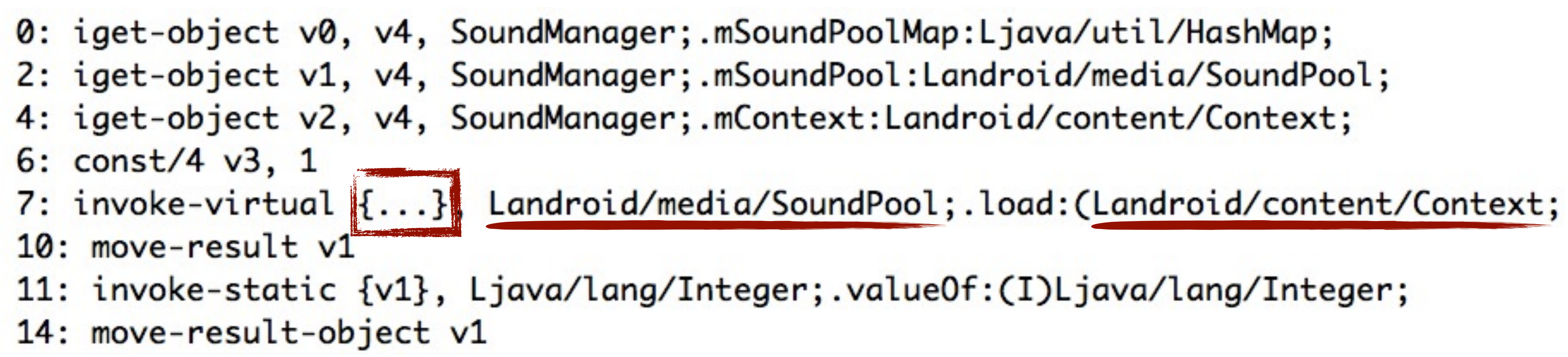
```
0: iget-object v0, v4, SoundManager;.mSoundPoolMap:Ljava/util/HashMap;
2: iget-object v1, v4, SoundManager;.mSoundPool:Landroid/media/SoundPool;
4: iget-object v2, v4, SoundManager;.mContext:Landroid/content/Context;
6: const/4 v3, 1
7: invoke-virtual {...}, Landroid/media/SoundPool;.load:(Landroid/content/Context;
10: move-result v1
11: invoke-static {v1}, Ljava/lang/Integer;.valueOf:(I)Ljava/lang/Integer;
14: move-result-object v1
```

Pruning by type



```
0: iget-object v0, v4, SoundManager;.mSoundPoolMap:Ljava/util/HashMap;
2: iget-object v1, v4, SoundManager;.mSoundPool:Landroid/media/SoundPool;
4: iget-object v2, v4, SoundManager;.mContext:Landroid/content/Context;
6: const/4 v3, 1
7: invoke-virtual {...}, Landroid/media/SoundPool;.load:(Landroid/content/Context;
10: move-result v1
11: invoke-static {v1}, Ljava/lang/Integer;.valueOf:(I)Ljava/lang/Integer;
14: move-result-object v1
```

Pruning by type



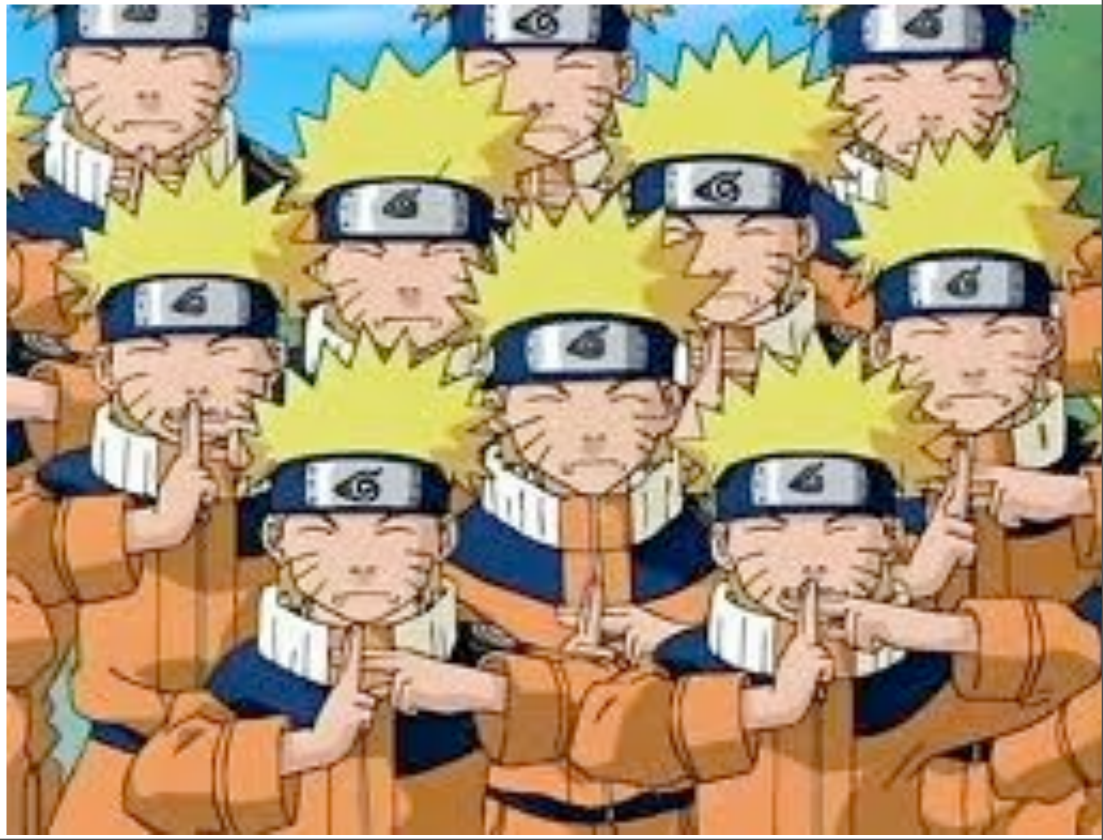
```
0: iget-object v0, v4, SoundManager;.mSoundPoolMap:Ljava/util/HashMap;
2: iget-object v1, v4, SoundManager;.mSoundPool:Landroid/media/SoundPool;
4: iget-object v2, v4, SoundManager;.mContext:Landroid/content/Context;
6: const/4 v3, 1
7: invoke-virtual {...}, Landroid/media/SoundPool;.load:(Landroid/content/Context;
10: move-result v1
11: invoke-static {v1}, Ljava/lang/Integer;.valueOf:(I)Ljava/lang/Integer;
14: move-result-object v1
```

Bottleneck - API

- Too many API classes, methods
  - Java / Android / Device manufacturer
- Handling unknown method
  - Keep Soundness? Give up analysis
  - So assume reasonable, general behavior
  - Preserve type information

Methods with same name

- From Java api class
  - equals:()Z, toString:()LJava/lang/String, run:()V, ...
- Obfuscation
  - a:(I)I, a:()V, a:(II)Z, ...
- virtual call with unknown type location => control explosion



Control Explosion

```
invoke-virtual {v0, v1} Container;.get:(I)Object;
move-result-object v2
invoke-virtual {v2} Object;.toString:()String
```

?

Conclusion

- Performance tuning
  - More localization
  - Improve handling APIs

- Future work : extension
  - Cover ARM native code (JNI)
  - Extend to general malware detector

Thank you