ScanDal: 안드로이드 악성앱 정적 분석기

윤용호 김진영 이광근
서울대학교 프로그래밍연구실

1. 성능 문제, 원인 살펴보니... '경악'

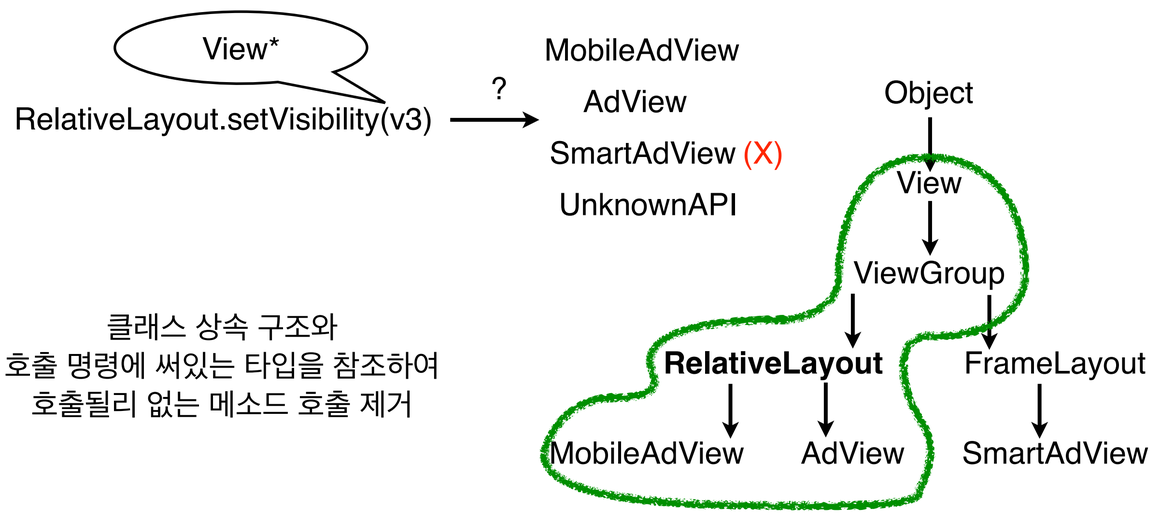
- 분석 비용의 양극화
  - 30분 이내 끝나는 앱 vs 12시간 이상 걸리는 앱
  - 앱의 크기와 무관
- 30분 timeout 기준 갤럭시 S2 기본앱 59/195 분석 실패
- 더 자세한 분석을 위해선 성능 개선이 우선
- 원인은
  - 한 변수에 너무 많은 요약 메모리 주소
  - 호출 될 메소드 선택 : 함수 호출 그래프 폭발

2. 개선 결과

앱 이름	개선 전			개선 후		
	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

3. 호출되지 않을 메소드 걸러내기



클래스 상속 구조와
호출 명령에 써있는 타입을 참조하여
호출될리 없는 메소드 호출 제거

4. 말이 안 되는 타입 걸러내기

```
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
```

타입 정보를 활용하여
각 레지스터에 담긴 주소값 중
타입이 맞지 않는 값을 제거

5. 도달 가능한 메모리만 전달하기

- 가비지 콜렉션의 요약
- 함수 호출 및 리턴 명령에서, 요약 메모리 중 접근 가능한 메모리들만 전달
  - 레지스터들에 담긴 주소와 그로부터 접근 가능한 모든 곳
  - 전역 변수에 담긴 주소와 그로부터 접근 가능한 모든 곳
- Preanalysis가 필요 없고 구현이 쉽지만
- 불필요한 주소들을 많이 보냄

6. 접근하는 메모리만 전달하기?

- Preanalysis가 꼭 필요
  - 빠르고 정확한
  - 포인터 분석을 포함하여
  - 메소드 호출을 포함한 전체 프로그램에 대해
- 기존의 자바 포인터 분석 연구들
  - SOOT 위에서 설계/구현되어 그대로 가져다 쓰기 어려움
  - API의 클래스 파일과 기계 코드까지 모두 가지고 분석
  - 문맥 민감도를 마음대로 조절하기 어려움
- 개별 메소드에 대한 분석만으로 접근할 주소들을 얻는 방법 구상중

7. 성능 개선 요약

- 타입 정보를 활용한
  - 변수의 메모리 주소값 잘라내기
  - 실제 일어나지 않을 메소드 호출 잘라내기
- 도달 가능한 메모리 주소들만 전달하기
- 새로운 preanalysis를 통해 접근하는 메모리만 전달하기 구상중

8. 앞으로 할 일

- 성능 더욱 개선
- ARM 코드를 포함하는 분석
- 일반적인 악성앱 분석기로 확장
- API의 실제 바이트코드와 네이티브 코드를 포함한 분석