인텐트 스펙 기반 안드로이드 유닛 테스팅 프레임워크 설계와 구현

고명필 O[1], 최광훈 [1], 창병모 [2]

[1] 연세대학교 컴퓨터정보통신공학부, [2] 숙명여자대학교 컴퓨터과학부

{myungpil.ko, kwanghoon.choi}@yonsei.ac.kr, chang@sookmyung.ac.kr

A Design and Implementation of Android Unit Testing Framework Using Intent Specification

Myungpil Ko O[1], Kwanghoon Choi[1], ByeongMo Chang[2]

[1]Computer&Telecommunications Engineering Division, Yonsei University,

[2]Department of Computer Science, Sookmyung Women's University

요 약

안드로이드 프로그램은 인텐트를 이용해 컴포넌트를 호출하고 서비스를 제공한다. 안드로이드는 인텐트의 정보가 부족하거나 달라도 컴파일 시간에 검사하지 못해 실행시간에 비정상 종료되는 취약점이 있다. 본 논문은 컴포넌트가 요구하는 인텐트 정보를 자유롭게 기술하는 인텐트 스펙을 제안하고, 기술된 인텐트 스펙에 따라 랜덤하게 생성한 인텐트를 전달해 컴포넌트의 취약점을 분석하는 방법을 제안한다.

1. 서 론

인텐트는 안드로이드에서 액티비티, 브로드캐스트 리시버, 서비스와 같은 컴포넌트 간 통신을 위한 메시지다. 컴포넌트에 인텐트를 전달하면 메서드에 인수를 전달한 것과 같이 인텐트를 인수처럼 이용한다.

안드로이드 플랫폼은 컴포넌트에서 요구하는 인텐트 형태가 달라도 컴파일 시간에 검사하지 못해 실행시간에 비정상 종료되는 취약점이 많다[1].

그림 1은 인텐트 정보를 이용해 제목과 내용을 보여주는 Note 액티비티다. 인텐트의 정보로 action은 *EDIT*와 *INSERT*를 기대한다. action이 *EDIT*인 경우는 *title, content*키에 *String* 타입의 extra를 추가로 기대한다.

이 예제는 잘못된 인텐트로 인한 취약점이 있다. 첫째, 인텐트의 action이 *EDIT*, *INSERT*가 아닌 다른 값이거나 *null*이면 *NullPointException*이 발생한다. 둘째, action이 *EDIT*일 때 *title*, *content*키의 타입이 *String*이 아니거나 값이 *null*이면 *NullPointException*이 발생한다.

이러한 취약점을 파악하기 위해 컴포넌트에 인텐트를 전달하는 유닛 테스트 방법을 제안한다. 이 방법으로 인텐트로 인해 발생하는 컴포넌트의 취약점을 파악할 수 있다. 제안한 유닛 테스팅은 자유롭게 인텐트를 기술할 수 있는 인텐트 스펙을 사용하여 원하는 인텐트 형태로 테스트할 수 있는 최초의 방법이다.

본 논문의 구성은 다음과 같다. 2장에서 관련 연구를 간략히 설명하고, 3장에서 인텐트 스펙 기반 안드로이드 유닛 테스팅 프레임워크에 대해 설명한다. 4장에 실험결과를 설명한 다음, 5장에서 결론을 맺고 향후 연구를 논의한다.

```
public class Note extends Activity {
  String title, content;
  void onCreate(Bundle savedInstanceState){
    Intent intent = getIntent();
    String action = intent.getAction();
    if (Intent.ACTION_EDIT.equals(action)) {
      title = intent.getStringExtra("title");
      content = intent.getStringExtra("content"); }
    else if (Intent.ACTION_INSERT.equals(action) {
      title = "new"; content = "memo"; }
    // Display title, content   } // Skip
}
```

그림 1 인텐트를 사용하는 액티비티

2. 관련 연구

랜덤 생성한 인텐트를 전달하는 Intent fuzzer[2]를 확장한 실험[3]은 컴포넌트가 인텐트로 인해 취약하다는 것을 보인다. ComDroid[4]는 바이너리 코드를 정적으로 분석하여 인텐트로 취약점이 발생할 가능성이 있는 부분을 알린다. 우리가 이전에 제안한 인텐트 스펙 검사[5]는 인텐트 스펙을 사용해 실행시간에 전달된 인텐트를 검사하는 방법으로 잘못된 인텐트로 발생하는 비정상 종료를 막고, 복구할 수 있는 기회를 제공한다.

본 논문에서 제안한 테스트 방법은 컴포넌트에서 기대하는 인텐트 정보를 자유롭게 기술할 수 있는 인텐트 스펙을 기반으로 랜덤 생성한 인텐트를 전달해 컴포넌트의 취약점을 분석한다. 이 방법은 다른 연구에서 시도된 적이 없다.

이 논문은 2014년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. NRF-2014R1A1A2053446)

3. 인텐트 스펙 기반 안드로이드 유닛 테스팅

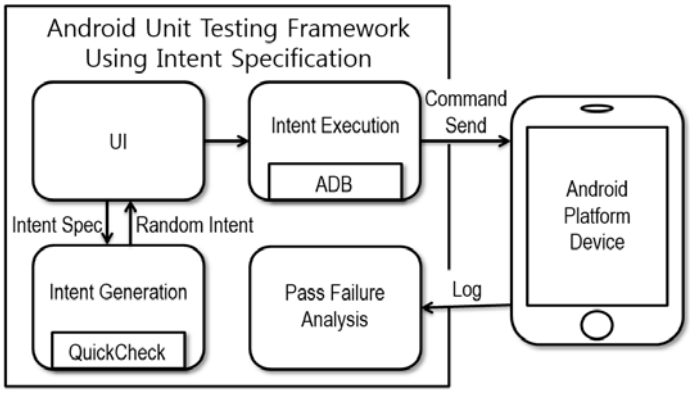

그림 2 인텐트 스펙 기반 안드로이드 유닛 테스팅

테스팅 프레임워크는 그림 2와 같이 네 단계로 구성된다. 첫째, UI에서는 랜덤으로 생성할 인텐트의 정보를 인텐트 스펙으로 입력받는다. 둘째, Intent Generation에서는 인텐트 스펙에 따라 랜덤 인텐트를 생성한다. 이때, 데이터를 랜덤하게 생성하는 QuickCheck[6]를 사용한다. 셋째, Intent Execution에서는 PC와 안드로이드 디바이스를 연결해주는 Android Debug Bridge(adb)[7]를 이용해 인텐트를 전달한다. 넷째, Pass Failure Analysis에서는 디바이스에서 발생하는 로그를 분석하여 취약점을 판단한다.

3.1 인텐트 스펙의 예

```
{ act = android.intent.action.EDIT
  [ title = String "new", content = String "content"] }
|| { act = android.intent.action.INSERT }
```

그림 3 Note 액티비티(그림 1)의 인텐트 정보

Note 액티비티가 기대하는 인텐트 정보는 인텐트 스펙[5]을 확장하여 그림 3과 같이 표현된다. 인텐트 스펙은 action(act)이 *EDIT*이거나 *INSERT*다. action이 *EDIT*인 경우는 추가 정보를 제공하는 extra([…])로 *title*과 *content*키에 *String*타입의 값을 요구한다.

3.2 인텐트 생성

기술한 인텐트 스펙에 따라 랜덤 인텐트를 생성한다. 인텐트 스펙에 부합하는 정도에 따라 다음 세 가지 종류로 구분한다.

- Compatible : 인텐트 스펙의 필드와 값에 맞춰 생성하되 새로운 필드를 추가한다.
- Shape-Compatible : 인텐트 스펙의 모든 필드는 갖추되 필드 값은 랜덤으로 생성한다. 그리고 새로운 필드도 추가한다.
- Random : 인텐트 스펙과 무관하게 랜덤 생성한 인텐트다.

그림 4는 그림 3의 인텐트 스펙으로 생성된 인텐트의 예를 보여준다.

```
▶ Compatible
{ act = android.intent.action.EDIT [ title = String "new",
content = String "content"] dat = qoFXwARtpfV-LNN ... }
▶ Shape-Compatible
{ act = android.intent.action.ADD [ key2 = int[] -1233387,
-72316, dKQn = String "xZQbcCTOW"] typ = video/* ... }
▶ Random
{ dat = tel:123 cat = [ttoIjEWJnpk, vYQEpERvvb, xpWj_Q,
 android.intent.category.APP_CALENDAR] ... }
```

그림 4 인텐트 스펙으로 생성된 랜덤 인텐트

3.3 유닛 테스트 실행

생성된 인텐트를 PC와 안드로이드 디바이스를 연결하는 adb[7]를 이용해 안드로이드 시스템에 전달한다. 안드로이드 시스템은 전달된 인텐트로 컴포넌트를 호출한다. 테스트는 Intent Generation에서 랜덤으로 생성한 인텐트의 개수만큼 반복한다.

3.4 유닛 테스트 결과 분석

유닛 테스트 실행결과 수집한 로그를 분석하여 자동으로 정상 동작과 비정상 판단을 구분한다. 추가로 소스를 확인하여 이때 사용한 테스트용 랜덤 인텐트가 테스트를 적용한 컴포넌트에서 기대하는 인텐트와 부합하는지 여부를 분석한다.

- 부합하는 인텐트 : 컴포넌트가 기대하는 인텐트의 필드가 일치하거나, 컴포넌트가 기대하는 필드를 모두 포함하고 추가로 더 많은 필드가 포함되어 있는 인텐트
- 부합하지 않는 인텐트 : 컴포넌트가 기대하는 필드 중 빠진 필드가 있거나, 컴포넌트가 기대하는 필드의 타입이 다른 인텐트

제안한 인텐트 스펙 기반 유닛 테스트 프레임워크가 구현된 화면은 그림 5와 같다.

그림 5 테스팅 프레임워크 구현 화면

4. 실 험

제안한 방법을 안드로이드 4.4.2 버전의 LG-F220K 환경에서 실험하고 결과를 표로 나타낸다. 실험은 컴포넌트가 기대하는 인텐트마다 Compatible, Shape-Compatible, Random 유형으로 25개씩 생성했다. 실험한 어플리케이션은 모두 오픈소스다.

표 1 유닛 테스트 실험결과

앱	정상 동작	비정상 종료
AndroidSecurity	75	0
BluetoothChat	75	0
Cafe	75	0
ContactsActivity	75	0
NotesList	427	23
Media_Player	436	14
Earthquake	300	0

표 1에서 첫 번째 열은 인텐트를 전달하여 어플리케이션이 정상 동작한 수를 나타낸다. 두 번째 열은 인텐트를 전달하여 어플리케이션이 비정상 종료된 수를 나타낸다.

NotesList는 23(5.1%)건의 인텐트에서 비정상으로 판단된다. NotesList의 취약점은 data 필드가 위치 형태로 전달되지만 해당 위치에 데이터가 없는 경우로 확인했다. Media_Player는 14(3.1%)건의 인텐트에서 비정상으로 판단된다. Media_Player의 취약점은 extra 필드가 존재하는지 검사하지 않고 사용해 발생한다.

안드로이드 플랫폼은 프로그램 속성을 선언하는 매니페이스에 선언된 action 조차 검사하지 않는 것을 실험을 통해 파악할 수 있다. 예를 들어, 파일을 제거하는 액티비티는 action이 *EDIT*인 인텐트를 전달해도 *DELETE* 기능을 수행하는 문제가 있다.

표 2 유닛 테스트 실험 결과 분석

앱	부합 인텐트		비부합 인텐트	
	정상	비정상	정상	비정상
AndroidSecurity	33	0	42	0
BluetoothChat	35	0	40	0
Cafe	32	0	43	0
ContactsActivity	37	0	38	0
NotesList	193	13	234	10
Media_Player	135	0	301	14
Earthquake	211	0	89	0

표 2에서 첫 번째 열과 두 번째 열은 컴포넌트에서 기대하는 인텐트 필드와 부합하는 랜덤 인텐트를 전달해 정상과 비정상으로 나타낸다. 세 번째 열과 네 번째 열은 컴포넌트에서 기대하는 인텐트 필드와 부합하지 않는 랜덤 인텐트를 전달해 정상과 비정상으로 나타낸다.

실험을 분석한 결과 실험한 모든 어플리케이션은 컴포넌트에 잘못된 인텐트가 전달되도 정상적인 인텐트를 전달한 것과 같이 서비스 하는 문제가 있다.

5. 결론 및 향후 연구

인텐트 스펙 기반으로 랜덤 생성한 인텐트를 전달하여 취약점을 분석하는 방법을 제안했다. 제안한 방법으로 컴포넌트의 취약점을 분석할 수 있음을 실험을 통해 보였다. 또한 컴포넌트에서 기대하는 인텐트 필드와 랜덤 인텐트의 부합여부 판단한 추가 실험으로 공통적으로 가지고 있는 취약점을 분석했다.

향후 연구로는 인텐트의 정보를 기술할 때 타입과 값이 일치하는지 타입체크를 통해 검증할 예정이다.

참고문헌

[1] Amiya K, Maji, Fahad A. Arshad, Saurabh Bagchi, and Jan S. Rellermeyer, “An empirical study of the robustness of Inter-component Communication in Android”, In proceedings of the 2012 42nd Annual IEEE/IFIP International comference on Dependable Systems and Networks (DSN) (DSN’12), IEEE Computer society, Washington, DC, USA, 1-12, 2012

[2] Intent Fuzzer, https://www.isecpartners.com/tools/mobile-security/intent-fuzzer.aspx, iSEC Partners, 2009.

[3] Raimondas Sasnauskas and John Regehr, “Intent Fuzzer: crafting intents of death”, In Proceedings of the 2014 Joint International Workshop on Dynamic Analysis (WODA) and software and System Performance Testing, Debugging, and Analytics (PERTEA)

[4] Erika Chin, Adrienne porter Felt, Kate Greenwood, and David Wagner, “Analyzing inter-application communication in android”, In Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services(MobySys11), pages 239-252, ACM, New York, Ny, Usa, 2011

[5] 고명필, 최광훈, 창병모, “강건한 안드로이드 앱을 위한 실행시간 인텐트 스펙 검사 방법”, 소프트웨어공학 학술대회, 2015

[6] Koen Claessen and John Hughes. QuickCheck: a lightweight tool for random testing of Haskell programs. In Proc. ICFP, pages 268–279, 2000

[7] Android Developers. Android Debug Bridge, 2015. https://developer.android.com/tools/help/adb.html