

# 안드로이드 앱의 랜덤 인텐트 테스트에서 동일한 에러 로그를 자동으로 그룹화하는 방법<sup>i</sup>

김현순, 윤성빈, 최지선, 고명필, 최광훈  
연세대학교 원주캠퍼스 컴퓨터정보통신공학부  
{coli, biassb, giftchoi, myungpil.ko, kwanghoon.choi}@yonsei.ac.kr

## An Automatic Method for Grouping Identical Error Logs in Random Intent Testing on Android Apps

Hyunsoon Kim, Sungbin Yoon, Jisun Choi, Myungpil Ko, Kwanghoon Choi  
Computer&Telecommunications Engineering Division, Yonsei University, Wonju

### 요 약

안드로이드 앱의 인텐트 취약점을 테스트하는 인텐트 퍼저에서 에러 확인 방법을 효율적으로 개선한 새로운 아이디어를 제안한다. 인텐트 퍼저는 랜덤 인텐트를 생성하여 앱을 실행한 다음 앱이 비정상 종료되는지 확인하는 테스트 도구이다. 이 논문에서 동일한 에러로 인해 발생한 다수의 비정상 종료 로그들을 하나의 그룹으로 만드는 자동 분류 방법을 제안한다. 테스터는 각 그룹의 대표 로그만 확인하면 된다. 최장 공통 부분 수열을 구하는 알고리즘을 응용하여 이 방법을 설계하였고, 이 방법을 상용 안드로이드 앱 10 개에 적용해 실험하였다. 모든 로그를 분석하는 대신 대표 로그를 분석하는 것으로 대체할 수 있음을 확인하였다. 그 결과 분석 대상 로그의 수가 크게 줄었다.

### 1. 서론 및 배경

안드로이드 앱은 안드로이드 플랫폼 API 를 사용하는 Java 로 작성한다. 액티비티, 서비스, 방송 수신자와 같은 종류의 컴포넌트들로 앱이 구성되어 있고, 이 컴포넌트들은 인텐트(Intent)라고 부르는 메시지를 서로 전달하며 모바일 서비스를 제공한다.

[그림 1]은 안드로이드 앱 예제로 Note 액티비티의 코드이다. 이 액티비티를 띄울 때 사용하는 인텐트의 action 필드 값으로 EDIT 나 INSERT 를 지정해야 하고, EDIT 액션의 경우 “title”과 “content” 키에 적절한 메모 제목과 내용의 문자열(String 타입)을 extra 인자로 지정해야 한다.

이 예제에서 여러 인텐트 취약점이 발생할 수 있다. 먼저, 인텐트의 action 을 지정하지 않거나 (NULL 로 지정하거나) 기대하는 EDIT 나 INSERT 가 아닌 다른 액션을 지정하는 경우에 널 참조 예외가 발생한다. EDIT 액션으로 지정했으나 “title”과 “content” 키에 String 타입의 문자열을 지정하지 않거나 이 타입이 아닌 Integer 타입의 값을 지정하는 경우 런타임 예외가 발생한다.

이러한 인텐트 취약점을 검출하기 위해 인텐트 퍼저(Intent Fuzzer)에 대한 기존 연구가 있었다 [1,2,3,4,5,6,7]. 인텐트 퍼저를 통해 각자의 방법으로 임의의 인텐트를 생성해서 안드로이드 앱이 실행 중에 비정상적으로 종료하지 않는지 테스트한다. 이 도구를 통해 많은 인텐트 취약점을 찾을 수 있었다.

```
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
    }
}
```

그림 1 인텐트를 사용하는 액티비티

하지만 인텐트 퍼저에 대한 기존 연구에서 비정상 종료 시 얻은 로그를 분석하여 유사한 에러를 하나의 그룹으로 모아 카운팅하는 방법에 대해서 특별한 언급이 없었다. 앱을 테스트할 인텐트가 다른 경우라도 동일한 에러로 귀결될 수 있다. 이러한 경우 하나의 에러로 카운팅해야 한다. 퍼즈 테스트(fuzz testing)의 속성상 테스트할 인텐트의 수가 많기 때문에 수작업으로 유사한 에러 로그들을 분류하기에는 시간이 너무 많이 소요될 것이다. 특히 안드로이드 마켓에 업

로드된 수십만 개의 앱들을 인테트 퍼저로 테스트하는 경우에 수동 에러 로그를 분류하기는 거의 불가능하다.

이 논문에서 인테트 퍼저를 안드로이드 앱에 적용해 얻은 비정상 로그들을 동일한 에러를 의미하는 그룹으로 자동으로 묶는 방법을 제시한다. 이 문제를 풀기 위해 최장 공통 부분 수열(Longest Common Subsequence, LCS)을 구하는 문제를 푸는 전통적인 알고리즘[8]을 사용했다. 두 개의 비정상 로그의 LCS와 두 로그 중 길이가 더 긴 로그의 길이 비율로 유사도를 정의한다. 이 기준으로 동일한 프로그램 위치에서 동일한 예외로 발생한 비정상 로그들을 동일한 에러로 판단할 수 있다. 또한 매번 실행할 때마다 다른 스레드 식별번호와 같은 약간의 차이가 있어도 동일하게 판단할 수 있다.

2 절에서 인테트 퍼저를 안드로이드 앱에 적용해 얻은 비정상 로그들 중 동일한 에러로 자동 분류하는 방법을 설명한다. 3 절에서 저자들이 구현한 인테트 퍼저를 상용 안드로이드 앱 10 개에 적용하여 실험한 결과를 설명하고, 4 절에서 결론을 맺는다.

## 2. 최장 공통 부분 수열 알고리즘을 활용한 동일 에러 로그 자동 그룹화 방법

이 논문에서 사용한 최장 공통 부분 수열 알고리즘에 대해 간략히 설명한다. 이 알고리즘은 두 수열의 가장 긴 공통 부분 수열을 찾는다. 단, 두 수열과 공통 부분 수열이 동일한 순서로 이루어져 있어야 한다. 그러나 꼭 연속적일 필요는 없다. 예를 들어, 수열  $X=<ABCBDAB>$ 와  $Y=<BDCABA>$ 에 대해 길이가 3인 공통 부분 수열은  $<BCA>$ 와  $<BDA>$ 가 있다. 길이가 4인 경우는  $<BCBA>$ 와  $<BDAB>$ 가 있고 길이가 5 이상인 공통 부분 수열은 존재하지 않으므로 최장 공통 부분 수열은 길이가 4인 경우로 결론 낸다.

두 수열의 공통 부분 수열의 길이가 길수록 두 수열이 유사하고, 또한 두 수열 중간에 각각 포함된 사소한 차이에 영향 받지 않는 점에서 안드로이드 비정상 로그들을 공통 에러로 그룹화하는데 적절하다.

안드로이드 앱이 비정상 종료하는 경우 얻은 로그에서 Error 로그와 Warning 로그를 순서대로 추출하여 하나의 비정상 로그 문자열을 만든다.

최장 공통 부분 수열 알고리즘을 활용한 동일 에러 로그 자동 그룹화 방법은 다음과 같다.

- 입력: 안드로이드 앱의 비정상 로그 N 개
- 출력: 동일 에러 그룹 M 개 ( $1 \leq M \leq N$ )

1. 모든 서로 다른 비정상 로그 문자열 두 개를 선택하여 다음 과정을 수행
2. 최장 공통 부분 수열 알고리즘을 적용하여 LCS(최장 공통 부분 수열)을 계산
3. LCS의 길이를 두 비정상 로그 문자열 중 더 긴 길이로 나눈 비율(유사도)를 계산

4. 계산한 유사도가 테스터가 정한 기준 이상이면 동일 그룹에 포함시킴
5. S1과 S2가 이미 같은 그룹에 속하고, S2와 S3의 유사도가 기준 이상이면 S1, S2, S3를 모두 같은 그룹으로 묶음

즉, 제안한 방법으로 얻은 동일 에러 로그 그룹은 추이 관계(Transitive relation)에 의해 기준 이상의 유사도를 갖는 모든 비정상 로그 문자열을 포함한다.

제시한 알고리즘을 예를 통해 설명한다. [그림 2]에서 인테트 퍼저를 특정 안드로이드 앱에 적용하여 비정상 종료하는 경우에 얻은 두 가지 로그이다. 이 두 로그에서 추출한 두 비정상 로그 문자열에 최장 공통 부분 수열 알고리즘을 적용해서 얻은 LCS는 [그림 3]에서 보여준다. LCS의 길이는 271이고 두 문자열은 모두 길이가 275이므로 유사도는 99%(271/275)이다. [그림 4]는 또 다른 예의 비정상 종료 로그이다. [그림 2]의 두 가지 비정상 종료 로그들과 비교해서 각각 유사도를 구해보면 66%(180/275)와 65%(179/275)이다. 따라서 테스터의 기준이 99%이면, 세 가지 비정상 로그를 입력 받아 {1, 2}, {3}으로 그룹화해서 출력한다.

Log 1	
E	dalvikvm >>>> com.CouponChart [ userId:0   appId:10292 ]
W	ContextImpl Failed to ensure directory: /storage/extSdCard/Android/data/com.CouponChart/cache
E	AndroidRuntime Process: com.CouponChart, PID: 14721
W	ActivityManager Force finishing activity com.CouponChart/.SplashActivity
Log 2	
E	dalvikvm >>>> com.CouponChart [ userId:0   appId:10292 ]
W	ContextImpl Failed to ensure directory: /storage/extSdCard/Android/data/com.CouponChart/cache
E	AndroidRuntime Process: com.CouponChart, PID: 15668
W	ActivityManager Force finishing activity com.CouponChart/.SplashActivity

그림 2 두 개의 안드로이드 에러 로그 Log1, 2

```
Edalvikvm>>>>> com.CouponChart [ userId:0 |
appId:10292 ]WContextImplFailed to ensure directory:
/storage/extSdCard/Android/data/com.CouponChart/cacheEAndroidRuntimeProcess: com.CouponChart, PID:
1WActivityManager Force finishing activity
com.CouponChart/.SplashActivity
```

그림 3 Log1과 Log2의 최장 공통 부분

Log 3	
E	dalvikvm >>>> com.CouponChart [ userId:0   appId:10292 ]
E	AndroidRuntime Process: com.CouponChart, PID: 16247
W	ActivityManager Force finishing activity com.CouponChart/.SplashActivity

그림 4 안드로이드 에러 로그 Log3

```
Edalvikvm>>>>> com.CouponChart [ userId:0 |
appId:10292 ]EAndroidRuntimeProcess:
com.CouponChart, PID: 147WActivityManager Force
finishing activity com.CouponChart/.SplashActivity
```

```
Edalvikvm>>>>> com.CouponChart [ userId:0 |
appld:10292 ]EAndroidRuntimeProcess:
com.CouponChart, PID: 16WActivityManager Force
finishing activity com.CouponChart/.SplashActivity
```

그림 5 Log1,2 와 Log3 사이의 최장 공통 부분

동일 에러 로그 자동 그룹화 방법을 구현하기 위해서 공간 절약형 최장 공통 부분 수열 알고리즘[8]을 사용하였다. 그 이유는 단순 알고리즘의 경우 지나치게 많은 공간을 요구하기 때문이다. 두 수열의 길이가  $m$  과  $n$  일 때 단순 알고리즘은  $O(m*n)$  공간이 필요하다. 실험에서 사용한 비정상 로그 문자열의 길이가 5596 과 8742 인 예에서 195.7M bytes( $5596*8742*4$ )가 필요했다. 반면 공간 절약형 알고리즘은  $O(m+n)$  공간만 사용하여 69.9K bytes( $2*8743*4$ )가 필요하다.

### 3. 실험 결과 및 논의

앞 절에서 설명한 동일 에러 로그 자동 그룹화 방법에 대한 실험 결과를 제시한다. 상용 안드로이드 앱 10 개를 선정해서 저자가 이전 연구에서 개발한 인텐트 퍼저 [6,7]를 적용하여 각 앱의 비정상 로그들을 준비하였다. 이 논문에서 제시한 방법을 사용하여 동일 에러 로그 그룹들을 자동으로 계산하고, 저자들이 직접 눈으로 확인한 유사 에러 로그 그룹과 비교해서 제안한 방법의 성능을 평가한다.

[표 1]은 실험 결과를 요약한 것이다. 첫번째 컬럼은 안드로이드 앱 이름이고, 두번째 컬럼은 인텐트 퍼저를 적용해 해당 안드로이드 앱이 비정상 종료한 횟수이고, 그 만큼 비정상 종료 로그를 얻을 수 있었다. 세번째 컬럼은 제안한 방법을 적용했을 때 얻은 동일 에러 그룹의 수이고, 네번째 컬럼은 저자가 직접 분류한 동일 에러 그룹에 대한 결과이다. 유사도 기준은 엄격하게 99%로 설정하였다. 100%를 설정할 수 없는 이유는 많은 비정상 종료 로그에 프로세스 ID 나 스레드 ID 등 실행할 때 마다 다른 번호가 빈번하게 나타나서 중요하지 않은 차이를 만들기 때문이다.

<표 1> 실험 결과 표

안드로이드 앱	비정상 종료 로그의 수	자동 분류한 그룹의 수	직접 분류한 그룹의 수
afreecatv	54	9	5
between	54	5	4
camera 365	52	2	2
cgv	271	24	16
clean master	0	0	0
coocha	29	16	3
facebook	0	0	0
gs shop	60	7	6
hoho	288	27	15
instagram	0	0	0

실험 결과를 다음 두 가지 중요한 사항을 확인할 수 있었다.

- 첫째, 비정상 종료 로그의 수에 비해 자동 분류한 동일 에러 그룹의 수가 매우 적다.
- 둘째, 엄격한 유사도 기준 99% 아래 자동 분류한 모든 동일 에러 그룹은 수작업으로 분류한 그룹에 모두 포함된다.
- 셋째, 수작업으로 분류한 동일 에러 그룹의 수와 자동 분류한 것과의 차이가 있는 이유는 동일한 에러 로그를 출력하지만 스레드 실행과 같은 동시 실행으로 인해 그 순서가 다른 경우가 있다.

첫 번째 분석 결과에 의해 테스터가 확인해야 할 비정상 종료 에러의 수를 줄일 수 있음을 확인하였다. 기본적으로 인텐트 취약점으로 인한 비정상 종료의 원인을 확인하기 위해 비정상 종료 로그를 모두 확인해야 한다. 반면 동일한 에러 그룹으로 분류하면 매우 적은 수의 각 그룹의 대표 로그만 확인하면 된다. 예를 들어 cgv 는 247(271-24)개, hoho 는 261(288-27)개 로그를 확인하지 않아도 된다.

두 번째 분석 결과를 통해 제안한 동일 에러 로그 분류 방법이 건전함(Soundness)을 보여준다. 분류된 각 그룹의 대표 로그만 확인하면 미처 확인하지 않고 놓친 에러 로그는 없다는 것이다.

세 번째 분석 결과는 제안한 방법이 개선해야 할 방향을 제시한다. 첫 번째와 두 번째 실험 결과를 통해 최장 공통 부분 수열 알고리즘이 비정상 종료 에러 로그를 분류하는데 효과적임을 확인했지만 더 고려해야 할 패턴의 로그들이 있음을 알 수 있었다. 출력 순서는 최장 공통 부분 수열을 만들 때 매우 중요한 영향을 미친다. 순서가 바뀌는 로그 패턴의 경우 적어도 줄 단위에 있는 부분 수열이 함께 순서가 바뀌는 특성을 이용해서 기존 최장 공통 부분 수열 알고리즘을 확장하는 것을 고려해 볼 수 있다.

### 4. 관련 연구

퍼즈 테스팅(fuzz testing)으로 인텐트 취약점을 효과적으로 찾는 인텐트 퍼저에 대한 연구는 다음과 같다. 널 인텐트 퍼저 [1]는 action 이나 data 필드에 널(NULL)을 설정한 인텐트로 테스트 한다. [2]에서 유효한 인텐트 필드 값을 설정하는 방법을 도입 확장했다. DroidFuzzer [3]는 data 필드에 정상 또는 비정상 멀티 미디어 파일을 가리키는 인텐트를 만들어 테스트한다. [4]와 [5]는 각각 정적 분석과 동적 모니터링을 통해 얻은 인텐트 필드의 키와 값을 이용해 인텐트를 채워 테스트한다

하지만 기존 연구에서 설계한 모든 인텐트 퍼저는 비정상 로그를 효율적으로 분석하는 방법에 대해서는 전혀 고려하지 않았다. 그리고 이 논문에서 제안한 방법은 특정 인텐트 퍼저에 의존적이지 않기 때문에 기존 연구에서 개발한 도구들에 활용할 수 있다.

### 5. 결론 및 향후 연구

안드로이드 앱의 인텐트 취약점을 테스트하는 인텐트 퍼저에서 그 적용 결과로 얻은 비정상 종료 로그

를 동일한 에러 그룹으로 자동 분류하는 방법을 제안하였다. 상용 안드로이드 앱에 제안한 방법을 적용하여 많은 동일한 에러로 분류할 수 있는 비정상 종료 로그들을 자동으로 올바르게 그룹화할 수 있음을 실험을 통해 보였다.

이 방법은 기존 인텐트 퍼저에 적용 가능한 장점이 있다. 또한 수십만 개의 안드로이드 앱이 있고 매일 다수의 앱이 업데이트되는 앱 마켓에서 관리자가 인텐트 퍼저를 통해 인텐트 취약점을 지닌 앱을 자동 필터링할 때 유용하게 사용할 수 있을 것이다.

### 참고문헌

[1] J. Burns. Intent Fuzzer, <https://www.isecpartners.com/tools/mobile-security/intent-fuzzer.aspx>, 2009.  
 [2] A.K.Maji, F.A.Arshad, S.Bagchi, J.S.Rellermeyer. An Empirical Study of the Robustness of Inter-component Communication in Android. In Proc. Of the Int'l Conf. on Dependable Systems and Networks, 2012.  
 [3] F.J.Hui Ye, Shaoyin Cheng, Lanbo Zhang. DroidFuzzer:

Fuzzing the Android Apps with Intent-Filter Tag, In Proc. of Int'l Conf. on Advances in Mobile Computing & Multimedia, pp.68-74, Vienna, Austria, 2013, ACM.  
 [4] R.Sasnauskas, J.Regehr. Intent Fuzzer: Crafting Intents of Death, In Proc. of the Joint Int'l Workshop on Dynamic Analysis and Software and System Performance Testing, Debugging, and Analytics, pp.1-5, San Jose, CA, 2014, ACM.  
 [5] M.P.Roe Hay, Omer Tirpp, Dyanmic Detection of Inter-application Communication Vulnerabilities in Android, In Proc. of the Int'l Symp. on Software Testing and Analysis, pp.118-128, New York, NY, USA, 2015, ACM.  
 [6] 고명필, 최광훈, 창병모, 인텐트 스펙 기반 안드로이드 유닛 테스트 프레임워크 설계와 구현, KCC2015, 제주대학교, 2015년 6월 24-26일.  
 [7] 윤성빈, 최지선, 최광훈, 안드로이드 앱 검수 자동화를 위한 인텐트 스펙 기반 테스트 방법, KCC2015, 제주대학교, 2015년 6월 24-26일.  
 [8] D.S.Hirschberg, A Linear Space Algorithm for Computing Maximal Common Subsequences, Communications of the ACM, 18(6):341-343, 1975.

<sup>i</sup> 본 논문은 정부(산업통상자원부)의 재원으로 한국산업기술진흥원의 바이오제조 GMP 기술인력양성사업의 지원을 받아 수행하였습니다(N0000961).

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