ISSN 2383-630X(Print) / ISSN 2383-6296(Online)
Journal of KIISE, Vol. 43, No. 2, pp. 212-221, 2016. 2
http://dx.doi.org/10.5626/JOK.2016.43.2.212

# 강건한 안드로이드 어플리케이션 개발을 위한 실행시간 인텐트 명세 검사 기법

## (A Runtime Inspection Technique with Intent Specification for Developing Robust Android Apps)

고 명 필 [†] 최 광 훈 [††] 창 병 모 [†††]
(Myungpil-Ko) (Kwanghoon Choi) (Byeong-Mo Chang)

**요 약** 안드로이드 앱의 액티비티, 서비스, 브로드캐스트 리시버와 같은 컴포넌트에 부적절한 인텐트를 전달하면 비정상적으로 종료되는 인텐트 취약점 문제가 빈번하게 발생한다. 이 논문은 안드로이드 컴포넌트가 기대하는 인텐트 명세를 개발자가 직접 기술하고, 실행시간에 이 컴포넌트에 전달된 인텐트를 검사하여 인텐트 취약점을 방지하는 방법을 제안한다. 인텐트 유효성을 검사하는 여러 조건문이 실수로 누락되거나 다른 코드와 섞여 유지 보수하기 어려운 점을 이 논문에서 제안하는 인텐트 명세를 선언함으로써 해결할 수 있다. 7개의 안드로이드 앱에 대해 제안한 방법을 적용한 실험 결과 인텐트 명세 기반 실행시간 검사 방법을 통해 앱의 비정상 종료를 방지할 수 있었다.

**키워드:** 안드로이드, 취약점, 강건성, 인텐트 명세, 실행시간 검사

***Abstract*** Android apps suffer from intent vulnerabilities in that they abnormally stop execution when Android components such as, activity, service, and broadcast receiver, take malformed intents. This paper proposes a method to prevent intent vulnerabilities by allowing programmers to write a specification on intents that a component expects to have, and by checking intents against the specification in runtime. By declaring intent specifications, we can solve the problem that one may miss writing conditional statements, which check the validity of intents, or one may mix those statements with another regular code, so making it difficult to maintain them. We perform an experiment by applying the proposed method to 7 Android apps, and confirm that many of abnormal termination of the apps because of malformed intents can be avoided by the intent specification based runtime assertion.

**Keywords:** android, vulnerability, robustness, intent specification, runtime assertion

· 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 기초연구연구사업임(No. NRF-2014R1A1A2053446)

† 학생회원 : 연세대학교 컴퓨터정보통신공학부
myungpil.ko@yonsei.ac.kr
†† 정 회 원 : 연세대학교 컴퓨터정보통신공학부 교수(Yonsei Univ.)
kwanghoon.choi@yonsei.ac.kr
(Corresponding author임)
††† 종신회원 : 숙명여자대학교 컴퓨터과학과 교수
chang@sookmyung.ac.kr

논문접수 : 2015년 10월 14일
(Received 14 October 2015)
논문수정 : 2015년 11월 25일
(Revised 25 November 2015)
심사완료 : 2015년 11월 26일
(Accepted 26 November 2015)


정보과학회논문지 제43권 제2호(2016. 2)

## 1. 서 론

안드로이드[1]는 모바일 디바이스를 위한 구글의 오픈 소스 플랫폼이다. 안드로이드 프로그램은 안드로이드 플랫폼 API를 사용하는 자바 프로그램으로, 액티비티, 서비스, 브로드캐스트 리시버, 컨텐트 프로바이더 컴포넌트로 구성된다. 인텐트는 안드로이드 플랫폼의 IPC 메시지다. 컴포넌트를 호출할 때, 호출할 컴포넌트를 지정하고 이 컴포넌트에서 요구하는 정보를 넣어 인텐트를 보낸다.

최근 연구 결과에 따르면, 잘못된 인텐트(malformed intent, 예를 들어, 특정 필드가 없는 인텐트)를 컴포넌트에게 전달하여 안드로이드 앱이 비정상 종료되는 취약점이 많다고 보고된바 있다[2]. 인텐트 취약점(intent vulnerability)를 테스트하는 도구, 인텐트 퍼저(intent fuzzer)[2-7]를 사용하여 무작위로 생성한 인텐트를 컴포넌트에게 전달해 총 332개의 액티비티 중 28(8.7%)개의 액티비티에서 오류가 발생하였다[2].

지금까지는 개발자가 인텐트 유효성을 검사하는 조건문을 직접 작성해서 인텐트 취약점에 대응해왔다. 하지만 기존 방법을 사용하는 경우, 인텐트에 여러 필드를 전달하면서 조건문이 복잡해져 누락 가능성이 있고 다른 코드와 섞여서 새로 필드를 추가하거나 기존 필드를 수정할 때 유지 보수가 어려운 문제가 있다.

본 논문은 인텐트 명세를 선언하여 인텐트 취약점에 대응하는 실행시간 인텐트 검사 방법을 제안한다. 즉, 컴포넌트에 전달된 인텐트를 사용하기 전에 잘못되었는지를 컴포넌트 입구에서 모든 필드에 대해 쉽게 검사하는 방법을 안드로이드 개발자에게 제공한다.

본 논문이 기여한 바는 다음과 같다. 첫째, 컴포넌트가 요구하는 인텐트를 기술할 수 있는 문법을 설계한다. 둘째, 제안된 인텐트 명세 문법으로 기술된 인텐트 정보와 전달 받은 인텐트를 실행시간에 검사하는 방법을 제안한다. 셋째, 잘못된 인텐트를 복구하는 방법을 개발자가 자유롭게 지정할 수 있다. 넷째, 실행시간 인텐트 오류 검사로 인해 비정상 종료되는 취약점을 개선하여 앱의 강건성을 높이고, 실행시간과 실행파일 크기에 대한 오버헤드가 낮음을 실험을 통해서 보인다.

앞으로의 논문 구성은 다음과 같다. 2장에서 배경 및 관련연구에 대해서 설명하고, 3장에서는 인텐트 명세를 제안한다. 4장에서 실험 결과를 설명하고, 5장에서 기타 사항을 논의한 후, 6장에서 결론을 맺는다.

## 2. 배경 및 관련 연구

이 절에서 안드로이드 앱의 컴포넌트 구조와 컴포넌트 사이에 전달하는 인텐트에 대해 설명하고, 인텐트 취약점 문제 및 기존 연구 결과를 살펴본다.

### 2.1 안드로이드 앱의 컴포넌트 기반 구조

안드로이드 앱은 액티비티(Activity), 서비스(Service), 브로드캐스트 리시버(Broadcast Receiver), 컨텐트 프로바이더(Content provider) 네 가지 유형의 컴포넌트들로 구성되어 있다. 그림 1은 컴포넌트의 관점에서 본 안드로이드 앱의 구조이다.

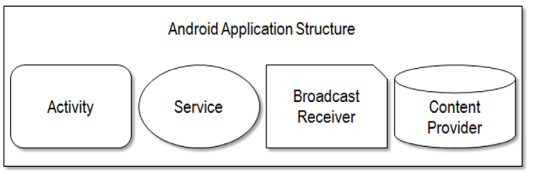


그림 1 안드로이드 앱의 컴포넌트 구조
Fig. 1 The Component Structure of Android Apps

액티비티(Activity)는 하나의 화면을 담당한다. 액티비티 내에서 사용자로부터 입력을 받거나 화면에 내용을 출력한다. 서비스(Service)는 백그라운드 작업의 인터페이스를 담당하는 컴포넌트이다. 액티비티와 달리 사용자 인터페이스가 없고 백그라운드에서 특정한 기능을 수행한다. 브로드캐스트 리시버(Broadcast Receiver)는 시스템 전체에 보내는 메시지를 수신하는 컴포넌트이다. 예를 들어, 시스템 전체로 보내지는 메시지 중 하나인 디바이스의 배터리 정보를 수신하여 일정 이하의 배터리 용량이 되면 사용자에게 알리고 절전상태로 전환하는 기능을 제공한다. 컨텐트 프로바이더(Contentprovider)는 서로 다른 안드로이드 앱이 데이터를 공유하기 위해 사용한다.

### 2.2 인텐트

안드로이드 시스템은 인텐트(intent) 메시지를 이용해 안드로이드 컴포넌트 간 통신을 지원한다. 그림 2는 인텐트로 컴포넌트를 호출하고 그 인텐트를 전달하는 과정이다.

안드로이드 시스템에서는 컴포넌트를 호출하고 이 컴포넌트가 요구하는 데이터를 전달하기 위해서 인텐트를 사용한다. 인텐트에 호출할 컴포넌트의 이름과 컴포넌트

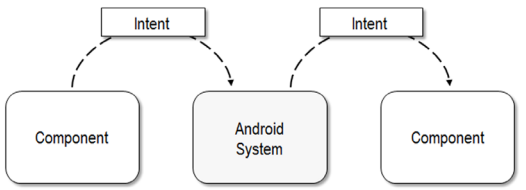


그림 2 컴포넌트간 인텐트 전달
Fig. 2 Intent Passing among Android Components

에게 서비스 받을 기능을 지정하고 해당 기능에 필요한 데이터를 추가로 지정한다. 이 인텐트를 안드로이드 시스템에 전달하면 안드로이드 시스템은 해당 컴포넌트를 호출하고 인텐트를 전달해 해당 컴포넌트의 서비스를 실행한다.

인텐트는 다음과 같은 필드로 구성된다[1].

- action : 서비스 기능을 구분
- data : 각 서비스에 필요한 데이터 URI
- type : 데이터의 MIME 타입
- extras : 추가 데이터
- component : 서비스를 제공하는 타켓 컴포넌트 이름
- 기타 category와 flags

예를 들어, 그림 3은 예제로 Note 액티비티의 클래스와 시작 메소드 onCreate의 코드를 일부 보여준다. Note는 인텐트를 통해 메모 제목과 내용을 전달받아 화면에 보여주고 편집하는 서비스를 제공하는 액티비티이다. 새로 메모를 생성하는 서비스와 기존 메모를 편집하는 서비스를 action INSERT와 EDIT로 구분한다. EDIT 액션의 경우 인텐트에 title과 content를 키로 문자열을 지정하되, INSERT 액션은 별도의 추가 데이터를 지정하지 않는다.

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

그림 3 액티비티 예(Note)
Fig. 3 An Activity Example (Note)

### 2.3 인텐트 취약점에 관한 연구

인텐트 취약점은 안드로이드 컴포넌트에 전달된 인텐트가 기대하는 필드를 적절히 갖추고 있는지 제대로 검사하지 않아 발생할 수 있는 문제다. 그림 3의 예제에서 이러한 부적절한 인텐트를 받으면 앱이 비정상 종료된다.

- 첫째, action이 EDIT일 때, 인텐트의 정보 중 extras인 title과 content를 null값으로 전달하면 나중에 이를 디스플레이할 때 NullPointerException이 발생한다.
- 둘째, action이 EDIT일 때, 인텐트의 정보 중 extras인 title과 content를 문자열이 아닌 숫자를 지정해 보냈다면 역시 NullPointerException이 발생한다.
- 세째, action이 EDIT나 INSERT가 아닌 경우 이를 처리하는 조건문이 없으므로 title, content가 모두 null이되어 동일한 문제가 발생한다.

일반적으로, 안드로이드 앱을 컴파일 할 때 인텐트 취약점을 미리 검사할 수 없다. 만일 안드로이드 컴포넌트 구조와 인텐트를 통한 컴포넌트 호출 과정을 Java 메소드로 매핑해서 구현했다면 메소드 호출에서 인자가 빠지거나 타입이 다른 인자가 주어지는 것에 대해 타입 검사 과정에서 미리 에러를 파악할 수 있었을 것이다. 하지만 인코딩된 방법으로 안드로이드 컴포넌트 구조와 인텐트를 만들었기 때문에 앞에서 설명한 인텐트 취약점을 Java 컴파일러가 미리 검사하지 못한다.

인텐트 퍼즈 테스팅 도구에 관한 기존 연구[2-7]에서 상용 안드로이드 앱의 인텐트 취약점이 심각함을 보고한 바 있다. 예를 들어, [3,4]에서 인텐트에 랜덤으로 생성한 정보를 지정해 안드로이드 컴포넌트에게 전달하여 실험했는데, 그 결과 332개의 액티비티 중 29(8.7%)가 액티비티가 NullPointerException, ClassNotFoundException, IllegalArgumentException의 오류가 발생해 비정상적으로 종료되었다. 심지어 OS가 재부팅까지 되는 사례도 있었다.

널 인텐트 퍼저(null intent fuzzer)[3]는 인텐트에 오직 널 값을 설정해서 컴포넌트를 테스트하고, 자자빙크스(JarJarBinks)[2]는 랜덤 값이나 일부 유효하지 않는 값을 생성해 설정해 테스트한다. 드로이드퍼저(DroidFuzzer)[5]는 인텐트의 data 필드에 비정상 오디오, 비디오 파일을 가리키도록 설정하고 컴포넌트를 테스트한 결과를 보고하였다. [4,6]은 각각 정적 분석과 동적 분석을 사용하여 인텐트의 모양을 최대한 유추해서 이 정보를 가지고 인텐트를 만들어 테스트하였다.

이와 같이 인텐트 취약점 테스트[2-6,8,9], 안드로이드 플랫폼에서의 인텐트 모니터링[10], 암시적 인텐트의 보안 취약점[11], 안드로이드 앱 시큐어 코딩 가이드[12]에 대한 연구가 진행되었으나, 본 논문에서는 제안한, 컴포넌트에서 기대하는 인텐트 명세를 개발자가 직접 기술하여 실행시간에 컴포넌트로 전달된 인텐트가 잘못된 것인지 검사하는 방법은 아직 다른 연구에서 시도된 바가 없다.

## 3. 실행시간 인텐트 명세 검사

인텐트 명세 기반 실행시간 검사 방법은 컴포넌트에 전달된 인텐트를 미리 선언한 인텐트 명세에 부합하는

지 실행시간에 검사하는 인텐트 취약점 방지 방법이다.

### 3.1 인텐트 명세 문법

그림 4는 인텐트 명세 언어에 대한 문법이다. 주요한 내용은 다음과 같다.

- 하나의 intent 모양을 { ... } 안에서 표현한다. 하나 이상의 인텐트를 기대하는 경우, ||로 구분해 나열한다.
- action은 값이 문자열이다. 그림 4에서 EDIT의 문자열을 act=android.intent.action.EDIT로 작성한다.
- data는 Uri값의 유무를 지정한다. 만약 data의 값이 content://contacts/people/1이라면 dat=non-null으로 기술한다. data 필드를 정규식 등으로 확장 정의하면 더욱 상세하게 인텐트 명세를 지정할 수 있을 것이다.
- type 필드 값이 존재해야 한다면 typ=non-null과 같이 작성한다.
- extras는 키와 값들의 리스트로 이뤄진다. 예를 들어, title 키에 String 타입의 값이 오고, content키와 String 타입의 값이 와야 한다면 extras는 [title=String, context=String]으로 기술한다. 배열은 타입이름과 []를 붙여 기술하는데, int 배열이라면 int[]와 같이 작성한다.
- component는 앱 패키지명과 앱 클래스 명을 /로 구분해서 기술한다. 예를 들어, Note 컴포넌트 이름은 cmp= kr.ac.yonsei.mobilesw.assertonintent/.Note라고 작성한다.
- category는 단일 항목의 경우 아래와 같이 작성한다. 여러 항목의 경우 콤마로 구분해서 나열한다.
  - cat=[android.intent.category.BROWSABLE]
- flag는 안드로이드 시스템에서 정의한 인텐트 플래그 정수를 지정한다.

아래의 인텐트 명세 예제는 그림 3의 Note 액티비티에서 요구하는 인텐트의 형태를 인텐트 명세 언어 문법에 따라 작성한 것이다.

```
{ cmp=com.example.android/.Note
    act=android.intent.action.EDIT
    [ title=String, content=String ]      }
|| { cmp=com.example.android/.Note
     act=android.intent.action.INSERT }
```

그림 4의 인텐트 명세 언어 문법에서 시작 기호는 INTENT로, 인텐트의 한가지 모양을 { FIELDS }로 기술하고 여러 모양을 ||로 구분한다. 문법에 의하면, FIELDS 기호로부터 액션, 추가 데이터 등의 0개 이상의 인텐트 필드들을 생성할 수 있다. 즉, Note 액티비티 컴포넌트의 이름을 필드 cmp=com.example.android/.Note로 지정하고, 액션을 필드 act=android.intent.action.EDIT로 기술하며, 추가 데이터를 리스트 형태로 [ ] 사이에 키와 값의 타입을 나열한다. 즉, title 키와 content 키에 String 타입의 값을 각각 설정해야하는 요구사항을 [title=String, content=String]으로 표현한다.

```
INTENT ::= { FIELDS } || INTENT
         | { FIELDS }
FIELDS ::= ACTION FIELDS
         | CATEGORY FIELDS
         | DATA FIELDS
         | TYPE FIELDS
         | COMPONENT FIELDS
         | EXTRA FIELDS
         | FLAG FIELDS
         | ε
ACTION ::= act=ID
CATEGORY ::= cat=[ID CATEGORYSUB]
CATEGORYSUB ::= , ID CATEGORYSUB | ε
DATA ::= dat=non-null
TYPE ::= typ=non-null
COMPONENT ::= cmp= ID / (ID | .ID)+
EXTRA ::= [ID = ID ARR EXTRASUB ]
EXTRASUB ::= , ID=ID ARR EXTRASUB | ε
FLAG ::= flg
ID ::= LETTER (A - Z | a - z | 0 - 9 | _ | .)*
LETTER ::= (A - Z | a - z)+
ARR ::= ([])*
```

그림 4 인텐트 명세 언어의 문법

Fig. 4 A Grammar for an Intent Specification Language

이 논문의 연구에서 그림 4의 인텐트 명세 언어 문법에 따라 파서 및 실행시간 검사 라이브러리를 구현했다. 이 구현 결과물을 아래의 웹 사이트에서 다운로드 받을 수 있다.

- https://github.com/nzzzn/AssertOnIntent

### 3.2 인텐트 명세를 사용한 실행시간 검사

인텐트 취약점으로 인해 안드로이드 앱이 비정상 종료되는 문제를 해결하기 위해서 앱의 각 컴포넌트에서 요구하는 인텐트 형태를 명세로 선언하고, 실행시간에 전달된 인텐트가 이 명세에 부합하는지 판단하고, 부적절한 인텐트를 발견했을 때 정상적인 프로그램 상태로 복구할 수 있는 기회를 제공하고자 한다.

그림 5는 그림 3의 Note 액티비티 컴포넌트에 인텐트 명세 기반 실행시간 검사 방법을 적용하여 구현한 예이다. 이 논문에서, Java 언어[2]의 assert문의 아이디어를 활용하여

- assertOnIntent(검사할 인텐트, 인텐트 명세, 핸들러)

형태로 라이브러리를 설계했다. 기술한 인텐트 명세에 검사 대상 인텐트가 부합하는지 이 메소드에서 검사하고 예외 발생 시 개발자가 지정한 핸들러에서 처리한다.

그림 5에서 사용한 인텐트 명세는 ||로 묶어진 두 가지 인텐트 유형을 나타낸다. 이 인텐트의 action은 EDIT이거나 INSERT이다. action이 EDIT인 경우 data가 null

```
public class Note extends Activity {
 void onCreate(Bundle savedInstanceState){
  new AssertOnIntent().assertOnIntent(
   // (1) The intent to inspect
   getIntent(),
   // (2) An intent specification
   "{ act=android.intent.action.EDIT     //action
      cmp=com.example.android/.Note
                                        //component
      [ title=String, content=String ] } //extras
   ||
    { act=android.intent.action.INSERT //action
      cmp=com.example.android/.Note }",
                                        //component
   // (3) A handler
   new MalformedIntentHandler() {
    public void handle (Intent intent,
                MalformedIntentException m) {
     switch(m.getNumber()) {
      case EXTRA_FIELD_MISSING:
       // Initialize the intent with default extras
       break;
      default: // Finish itself
       break;
     }
    }
   });  // End of assertOnIntet()

  // The same code as onCreate() in Fig.1
  Intent intent = getIntent();
  …
 }
 …
}
```

그림 5 실행시간 인텐트 검사
Fig. 5 Runtime Assertion on Intent

이 아니고, component는 com.example.android/.Note이며, extras로 title과 context 키에 String 타입의 문자열이 있어야 한다. action이 INSERT인 경우 component가 com.example.android/.Note이고, 그 외의 제약은 없다.

컴포넌트에서 전달받은 인텐트를 인텐트 명세와 비교하면 세 가지 경우가 발생할 수 있다. 인텐트 명세에 나열된 인텐트 모양 중 하나와 정확하게 일치하면 정상으로 판단한다. 명세에 기술된 것보다 더 많은 필드를 포함하고 있는 경우도 (예를 들어, EDIT action의 경우 extras에 date키와 값을 추가로 포함해도) 일종의 서브타이핑을 고려해 정상이라 판단한다. 만일 인텐트 명세에 기술한 필드를 가지고 있지 않는, 더 적은 수의 필드를 가진 인텐트를 받았을 경우에 비정상으로 판단한다.

주어진 인텐트가 명세에 부합하지 않는 인텐트 취약점이 발생하면 각 개별 상황별로 개발자가 대응할 수 있다. 그림 5의 assertOnIntent 메소드 호출에서 첫 번째 인자로 받은 인텐트와 두 번째 인자로 받은 인텐트 명세를 비교해서 부합하면 정상 리턴해서 다음 문장을 실행하지만, 그렇지 않으면 세 번째 인자로 지정한 MalformedIntentHandler 인터페이스 타입의 핸들러 객체에서 정의한 handle 메소드를 호출하여 인텐트 취약점 발생 상황을 알린다. Malformed IntentException 예외 클래스를 설계하여 이 상황의 오류 코드를 전달한다. 예를 들어, 그림 5에서 오류 코드 EXTRA_FIELD_MISSING은 인텐트 명세에 기술된 extras 필드 중 일부가 주어진 인텐트에 지정되어 있지 않아서 실패했음을 나타낸다. 이 경우 개발자는 전달받은 인텐트의 extras에 디폴트 값을 설정하면 프로그램을 정상 상태로 복구할 수 있을 것이다. 그 이외의 오류 코드에 대해서는 Note 액티비티를 정상적으로 종료한다.

인텐트 명세에서 사용하는 예외코드는, 인텐트 명세의 구문 오류, action, data, category, type, component, extras, flag에서 명세와 일치하지 않는 오류, 기타 오류, 34가지로 분류해서 설계하였다. 오류 코드의 목록과 자세한 설명을 [13]에서 확인할 수 있다.

### 3.3 Java 주석을 활용한 인텐트 명세 선언 및 실행시간 검사 코드로의 컴파일 변환

3.2절에서 설명한 바와 같이 assertOnIntent 메소드를 포함한 라이브러리를 개발자가 직접 호출하는 방식으로 사용할 수도 있지만, Java의 주석(Annotation)으로 인텐트 명세를 선언하여 인텐트 검사를 할 수 있도록 구현하였다.

그림 6은 예제로 사용한 Note 액티비티의 onCreate 메소드 앞에 인텐트 명세를 Java 주석으로 선언한 코드 예를 보여준다. 이 논문에서 개발한 인텐트 명세 주석 처리기를 이 예제에 적용하면 개발자가 직접 라이브러리를 사용한 그림 5의 코드로 자동 컴파일 변환된다.

실행시간 인텐트 검사를 할 메소드 앞에 주석 표시(@)와 주석 이름 IntentSpec으로 선언을 시작한다. 이 주석의 요소 spec에 인텐트 명세를 문자열로 기술한다. 옵션으로 요소 exception에 인텐트 명세를 통과하지 못한 경우에 대한 처리 코드를 하나 이상의 @IntentSpecException에 지정할 수 있다. IntentSpecException은 에러 코드와 이 에러를 처리하는 코드를 각각 code와 process에 문자열로 기술한다.

Java 주석 형식으로 선언한 인텐트 명세를 컴파일 변환하는 방법은 Spoon 소스 코드 변환기[14]에서 제공하는 Java 주석 처리 방법을 활용하여 구현하였다.

```
public class Note extends Activity {
 @IntentSpec(
  spec=
  "{ act=android.intent.action.EDIT    //action
       cmp=com.example.android/.Note
                                        //component
       [ title=String, content=String ] } //extras
   ||
    { act=android.intent.action.INSERT //action
       cmp=com.example.android/.Note }",
  exception={
   @IntentSpecException(
     code="EXTRA_FIELD_MISSING",
     process=
        "// Initialize the intent with default extras"),
   @IntentSpecException(
     code="default",
     process="// Finish itself")})
 protected void onCreate(Bundle savedInstanceState)
 {
 ...
 }
 ...
}
```

그림 6 Java 주석을 사용한 인텐트 명세 선언 예
Fig. 6 An Example of Intent Specification Declaration using Java Annotations

## 4. 실험 결과

실행시간 인텐트 명세 검사 방법을 7가지 안드로이드 앱에 적용하여 인텐트 취약점 방지에 대한 실험을 수행하였다. 안드로이드 SDK에 포함된 샘플 앱 BluetoothChat과 NotesList, 학부생 프로젝트 Cafe, 나머지 앱은 고급 안드로이드 프로그래밍 서적[15,16]에서 참조하였다. 실험 환경으로 안드로이드 4.4.2버전의 LG-F220K 모바일 기기를 사용하였다.

### 4.1 인텐트 취약점 방지

표 1은 인텐트 명세 기반 실행시간 인텐트 검사를 7개의 안드로이드 앱에 적용해 취약점 방지를 분석한 결과이다. 안드로이드 앱의 소스 코드를 직접 들여다보고 각 컴포넌트에서 기대하는 인텐트를 분석한 다음 이 논문에서 제안한 assertOnInsert 문에 명세를 작성하여 원래 소스 코드에 추가하였다.

표 1에서 인텐트 취약점(intent vulnerabilities) 개수는 잘못된 인텐트를 안드로이드 앱에 전달하여 비정상 종료되거나 컴포넌트가 잘못된 정보를 검사하지 않고 사용한 수를 의미한다. 소스 코드 리뷰를 통해 이러한 인텐트 취약점을 30개 발견하였다. 잘못된 인텐트 정보

표 1 실행시간 인텐트 취약점 검사 결과(단위: 개수)
Table 1 A Result of Runtime Checking Intent Vulnerabilities

| Apps | Intent Vulner-abilities | malformed Intents Detected | abnormal Execution | defending Abnormal Execution |
|---|---|---|---|---|
| Android Security | 1 | 1 | 0 | 0 |
| Bluetooth Chat | 4 | 4 | 1 | 1 |
| Cafe | 1 | 1 | 0 | 0 |
| Contacts Activity | 1 | 1 | 0 | 0 |
| NotesList | 17 | 15 | 6 | 4 |
| MediaPlayer | 4 | 4 | 0 | 0 |
| EarthQuake | 2 | 2 | 0 | 0 |
| Total | 30 | 28 | 7 | 5 |

를 검사하지 않는 취약점을 모든 앱에서 확인할 수 있었다.

실행시간 인텐트 검사 방법으로 검출한 잘못된 인텐트(Malformed Intents Detected)의 개수는 두 번째 열에서 정리한 인텐트 취약점을 유발할 수 있는 잘못된 인텐트 30개를 각 해당 앱에 전달하여 측정한 것이다. 총 30개의 잘못된 인텐트 중 28(93.3%)개를 제안한 방법으로 검출할 수 있었다. 검출하지 못한 두 개의 인텐트는 data 필드의 값이 올바르지 않은 위치(Uri)를 가르키는 경우이다. 이 논문에서 제안한 인텐트 명세의 표현력의 범위에서 data의 Uri 유무만을 작성하기 때문에 이 두 가지 경우를 통과시켰지만 향후 연구로 패턴 매칭이나 접근 가능한 데이터인지 확인하는 방법을 추가한다면 개선할 수 있을 것이다.

인텐트 취약점으로 인해 안드로이드 앱이 비정상 종료(Abnormal Execution)되는 심각한 경우는 BluetoothChat와 NotesList이다. 이 두 가지 어플리케이션에서 비정상 종료되는 인텐트 취약점의 원인은 다음과 같이 파악할 수 있었다.

- BluetoothChat 앱의 경우에는 주변 블루투스 장치를 찾을 경우 그 장치의 주소를 담은 인텐트를 전달하는데 만일 이를 누락하고 인텐트를 보내도록 조작하면이 장치의 주소를 읽는 코드에서 NullPointerException 에러가 발생한다.
- NotesList 앱의 경우는 data 필드가 누락되거나 잘못된 위치를 담고 있는 경우 NullPointerException 에러가 발생하거나 IllegalArgumentException 에러가 발생하며 비정상 종료된다.

비정상 종료되는 경우 중에서 제안한 실행시간 인텐트 검사 방법으로 방지한(Defending Abnormal Execution) 개수를 조사해보니, 비정상 종료되는 경우가 관찰

표 2 비정상 종료 에러 분류(단위: 개수)
Table 2 Classification of Exceptions in the Abnormal Terminations

| Apps | NullPointer Exception | IllegalArgument Exception |
|---|---|---|
| BluetoothChat | 1 | 0 |
| NotesList | 4 | 2 |
| Total | 5 | 2 |

된 두 가지 앱 BluetoothChat과 NotesList에서 비정상 종료되는 총 7개의 취약점 사례 중 5(71.4%)개의 취약점을 방지하였다. 하지만, NotesList에서 data 필드에 잘못된 위치를 기술한 인텐트를 전달하는 경우 IllegalArgumentException 예외가 발생하였다. 이러한 형태의 잘못된 인텐트는 해당 위치의 데이터에 존재 여부를 판단해야만 방지할 수 있는데 제안한 인텐트 명세 언어의 표현력의 범위를 벗어난다.

표 2는 실험에서 수집한 모든 비정상 종료 사례에서 발생한 예외의 종류를 나타낸다. BluetoothChat의 경우 인텐트 취약점으로 인해 모두 NullPointerException이 발생했고, NotesList의 경우에는 IllegalArgumentException도 함께 발생했다.

실행시간 인텐트 검사에 대한 실험 결과를 요약하면, 전체 7개의 안드로이드 앱에서 인텐트를 검사하지 않고 사용하는 취약점을 모두 보유하고 있고, 이 중 2개의 앱에서는 비정상 종료되는 심각한 취약점을 발견하였다. 제안한 인텐트 명세 검사방법으로 인텐트 오류를 확인해 전체 인텐트 취약점 30개 중 28(93.3%)개를 방지할 수 있었고 특히 비정상 종료되는 심각한 인텐트 취약점 7개 중 5(71.4%)개를 방지함으로써 제안한 방법이 유용함을 보였다.

### 4.2 실행시간 인텐트 검사 방법의 오버헤드

인텐트 명세 기반 실행시간 인텐트 검사 방법은 실행시간이 늘어나고 실행파일 크기가 증가하는 오버헤드를 수반한다. 표 3은 이 오버헤드를 측정한 결과이다.

바이너리 크기(Binary size)는 수정하기 전 안드로이드 바이너리 파일의 크기를 나타내고 괄호 안의 숫자는 제안한 방법을 앱에 적용할 때 라이브러리 포함으로 인해 증가한 바이너리 크기를 나타낸다. 실행시간 인텐트 검사방법을 앱에 적용하기 위해 추가되는 인텐트 명세 파서 코드와 잘못된 인텐트를 검사하는 코드, 핸들링하는 코드로 인해 각 앱이 평균 10,439(3.4%)byte가 증가함을 확인할 수 있다. 앱 크기가 작은 앱인 BluetoothChat의 경우에는 10,811byte가 증가하여 39.2%가 증가되기는 했지만 제안한 방법을 적용한 후의 앱의 크기는 37Kbyte에 불과하다.

표 3 실행시간 인텐트 검사 적용의 오버헤드 측정
Table 3 Measuring Overhead Owing to Runtime Checking Intent Vulnerabilities

| Apps | Binary Size (bytes) | Startup Time(ms) | LOC |
|---|---|---|---|
| Android Security | 321,704(+9,009) | 421(+10) | 2 |
| Bluetooth Chat | 27,545(+10,811) | 450(+10) | 4 |
| Cafe | 1,473,874(+10,057) | 431(+9) | 3 |
| Contacts Activity | 36,116(+10,616) | 440(+10) | 2 |
| NotesList | 60,405(+11,038) | 421(+10) | 10 |
| MediaPlayer | 154,282(+11,120) | 450(+10) | 8 |
| Earthquake | 51,117(+10,439) | 431(+9) | 2 |
| Average | 303,577(+10,439) | 434(+9.7) | 4.4 |

시작 시간(Startup time)은 변경 전 앱의 시작 액티비티 컴포넌트를 호출해서 onCreate 메소드를 실행 완료하기까지의 스타트업 시간을 나타내고, 괄호 안의 숫자는 제안한 방법을 적용할 때 증가한 시간을 나타낸다. 어플리케이션이 시작되면서 선언된 인텐트 명세를 파싱하고 전달된 인텐트와 검사하는 과정이 추가되어 각 앱이 평균 9.7(2.2%)ms의 시간이 증가했다. 스타트업 시간이 증가하기는 했지만 사용자가 적용 전과 후의 스타트업 시간을 구분할 수 있을 정도의 지연 시간은 아니다.

라인 수(LOC)는 제안한 실행시간 인텐트 명세 검사 방법을 앱에 적용하면서 소스 코드에 추가 작성한 코드의 라인 수이다. 인텐트를 사용하는 부분이 많을수록, 사용하는 형태가 다양할수록 추가할 코드의 라인 수는 늘어난다. 라인 수가 가장 많이 증가한 NotesList의 경우 4개의 액티비티에서 인텐트를 사용하여 10줄의 라인이 증가한다. 각 어플리케이션에 증가하는 평균 라인 수는 4.4줄 정도로 많지 않다. 인텐트 명세 검사 예외 처리는 라이브러리에서 디폴트 옵션으로 제공하는 피호출 컴포넌트 종료를 지정해 실험하였다.

### 4.3 인텐트 퍼즈 테스팅 결과

인텐트 퍼즈 테스팅(intent fuzz testing)[2-7]은 유효한 인텐트 뿐만 아니라 필드가 빠지거나 타입이 다른 값으로 설정된 잘못된 인텐트를 만들어서 안드로이드 컴포넌트에 전달해 비정상 종료되는지 여부를 확인하는 방법이다. 안드로이드 앱의 인텐트 취약점을 검사하는 효과적인 방법으로 알려져 있다.

표 4는 인텐트 명세 기반 실행시간 검사 방법을 적용한 안드로이드 앱을 인텐트 퍼즈 테스트 환경에서 측정한 실험 결과이다. 이 실험을 통해 인텐트 퍼즈 테스팅 환경에서도 제안한 실행시간 인텐트 검사 방법이 효과적인지 확인하려 한다.

표 4 인텐트 퍼즈 테스팅 실험 결과 (단위: 개수)
Table 4 An Experiment Result in Intent Fuzz Testing

| Apps | Intents matched with the spec. | | Intents unmatched with the spec. | |
|---|---|---|---|---|
| | Normal Exec. | Abnormal Exec. | Normal Exec. | Abnormal Exec. |
| Android Security | 33 | 0 | 42 | 0 |
| Bluetooth Chat | 35 | 0 | 40 | 0 |
| Cafe | 32 | 0 | 43 | 0 |
| Contacts Activity | 37 | 0 | 38 | 0 |
| NotesList | 193 | 13 | 234 | 10 |
| MediaPlayer | 135 | 0 | 301 | 14 |
| Earthquake | 211 | 0 | 89 | 0 |
| Total | 676 | 13 | 787 | 24 |

실험에서 사용한 인텐트 퍼즈 테스팅 도구[7]는 총 1,500개의 임의의 인텐트를 생성해서 7개의 앱에 적용하여 비정상 종료 여부를 확인한다. 각각의 인텐트를 인텐트 명세에 부합하는 것과 그렇지 않은 것으로 먼저 분류하였다. 부합하는 인텐트는 실행시간 인텐트 검사를 통과한다. 또한 각 인텐트 분류에서 앱이 정상 실행되는지 여부를 조사하였다. 이 절의 실험을 위하여, 실행시간 인텐트 검사 라이브러리를 일부 수정하여 인텐트 검사를 해서 명세와 부합하는지 여부를 기록하되 무조건 통과시켜 앱의 실행을 모니터링 하였다.

두 번째 열에서 인텐트 명세에 부합하는 인텐트는 실행시간 인텐트 검사 방법을 통과한 경우이다. 676개는 실행시간 검사를 통과하고 정상 실행되었으나 13가지 인텐트의 경우 NotesList 앱이 비정상 종료하였다. 이때 발생한 예외를 살펴보면, NullPointerException 3개, SecurityException 5개, IllegalArgumentException 5개로 분류할 수 있다. 이 예외는 모두 NotesList에서 data 필드를 사용할 때 발생한 것으로 분석되었다. 예를 들어, data 필드에 e4S/2bdzIT9와 같은 부적절한 Uri 값을 담은 인텐트를 NotesList에 전달하면 e4S/2bdzIT9의 위치에는 데이터가 존재하지 않으므로 NullPointerException이 발생했다. NotesList가 보유하고 있지 않은 접근 권한을 요구하는 content://contacts/people/1을 data 필드 값으로 지정한 인텐트를 전달 받으면 SecurityException이 발생했다. 마지막으로 인텐트의 data필드에 tel:123과 같은 Uri 값을 지정해 NotesList에 전달하면 잘못된 Uri로 인식하고 IllegalArgumentException이 발생했다. 모두 4.1절에서 확인한 문제와 동일함을 확인할 수 있었다.

세 번째 열에서 인텐트 명세에 만족하지 않는 인텐트는 제안한 실행시간 인텐트 검사 방법으로 모두 걸러낼 수 있는 경우이다. NotesList와 MediaPlayer의 경우 비정상 종료되는 심각한 인텐트 취약점이 발생할 수 있는데 이를 방지할 수 있음을 확인하였다.

## 5. 논의

### 5.1 인텐트 명세 선언의 장점

안드로이드 컴포넌트에 인텐트 명세를 선언함으로써 기존의 복잡하고 유지보수하기 어려운 인텐트 유효성 검사 코드를 대체할 수 있는 장점을 지닌다.

컴포넌트에서 기대하는 인텐트 액션의 수가 늘어나고 추가 데이터(extras)가 복잡하면 개발자가 작성해야할 유효성 검사 코드도 복잡해진다. 그림 3의 Note 액티비티 예제에서는 이 액티비티를 시작할 때 처음 실행되는 메소드(onCreate)에서 인텐트로 전달받은 title과 content 문자열 인자를 추출하였다. 하지만, 실제 안드로이드 앱 소스 코드를 살펴보면 한 두 단계 이상 메소드를 추가로 호출하여 도달한 곳에서 데이터를 추출하는 경우도 있고, 또는 사용자가 화면 버튼을 누르면 비로소 (버튼 리스너) 메소드 안에서 인텐트에 전달된 값을 추출하는 경우도 많다.

이런 사례에서와 같이 인텐트의 유효성을 검사하는 조건문이 앱 소스 코드의 여러 부분에 흩어져 있기 때문에 실수 등의 이유로 유효성 검사를 누락할 가능성이 있고, 누락된 사실도 앱을 실행해서 (예를 들어, 사용자가 버튼을 누른 다음에) 비로소 확인할 수 있다. 그리고, 새로운 필드를 인텐트에 도입하거나 기존 필드를 변경하기 위해 인텐트 유효성 검사 코드를 재 작성할 때 유지 보수하기가 어렵다.

반면에 이 논문에서 제안한 방법은 인텐트에서 값을 추출하는 코드 위치와 무관하게 컴포넌트 시작 메소드에서 인텐트 명세를 선언한다. 따라서 인텐트 유효 검사 코드를 그 이외의 정규 코드와 명확히 구분할 수 있고, 인텐트에 새로 필드를 추가하거나 필드 타입을 변경하기도 매우 용이하다. 인텐트 명세에 유효한 인텐트가 갖춰야 할 조건을 모두 기술하므로 유효성 검사에서 특정 조건을 누락할 가능성이 매우 낮다.

### 5.2 인텐트 필터를 이용한 인텐트 명세 자동 작성

인텐트 명세 기반 실행시간 인텐트 검사 방법의 단점은 개발자가 인텐트 명세를 작성해야하는 점이다. 이를 보완하기 위해, 안드로이드 앱의 인텐트 필터 정보를 이용해서 인텐트 명세를 부분적으로 자동 생성할 수 있고, 이를 개발자가 보완하는 방법을 고려할 수 있다. 인텐트 필터를 활용하여 테스트할 인텐트의 일부 정보를 알아내는 방법을 인텐트 취약점 테스트 분야의 연구에서 활용한 바 있다[5,7].

안드로이드 앱의 컴포넌트 설정 파일인 매니페스트 파일(AndroidManifest.xml)에서 컴포넌트 별 인텐트 필터를 선언한다. 이것은 타겟 컴포넌트를 지정하지 않는 암시적 인텐트(implicit intent)를 받을 컴포넌트를 결정하기 위해 안드로이드 시스템에서 사용한다[1].

컴포넌트가 요구하는 인텐트의 형태를 선언하는 방법이라는 점에서 이 논문에서 제안한 인텐트 명세와 유사한 점이 있지만, 인텐트 필터는 action과 data의 타입, category 필드만 선언하는 제한적인 방법이다.

예를 들어, Note 액티비티에 대한 인텐트 필터는 다음과 같이 선언되어 있다.

```
<activity android:name="com.example.android.Note">
 <Intent-filter>
 <action android:name="android.intent.action.EDIT"/>
 <action android:name="android.intent.action.INSERT"/>
 ...
 </intent-filter>
</activity>
```

컴포넌트 이름을 지정하지 않은 암시적 인텐트를 만들고 액션 필드에 EDIT나 INSERT를 설정해서 이에 매치되는 액티비티를 찾아 실행할 것을 안드로이드 플랫폼에서 요청하면 이 인텐트 필터를 찾고 Note 액티비티를 실행한다.

이 인텐트 필터로부터 인텐트 명세를 자동으로 만든다면 다음과 같은 결과를 얻을 것이다.

```
{ cmp=com.example.android/.Note
    act=android.intent.action.EDIT }
||
{ cmp=com.example.android/.Note
  act=android.intent.action.INSERT }
```

비록 인텐트의 추가 데이터 필드를 유추할 수는 없었지만 개발자가 인텐트 명세를 작성하는 작업을 부분적으로 자동화 할 수 있다.

## 6. 결 론

부적절한 인텐트를 컴포넌트에 전달해 비정상적으로 앱이 종료되는 인텐트 취약점 문제를 인텐트 명세를 기술하고 실행시간에 검사하여 해결하는 방법을 제안하였다. 기존의 안드로이드 프로그래밍 방법에서 인텐트 유효성을 검사하는 여러 조건문이 실수로 누락되거나 다른 코드와 섞여 유지 보수하기 어려운 점을 이 논문에서 제안한 인텐트 명세를 선언함으로써 해결할 수 있다. 안드로이드 앱에 적용하는 실험을 통해, 인텐트 취약점으로 비정상 종료되는 경우를 방지하여 앱의 강건성을 높일 수 있음을 확인하였다.

향후 연구로 개발자가 인텐트 명세를 올바르게 작성했는지 검증하는 정적 분석에 관한 연구를 진행하고자 한다. 이 논문에서 제안한 실행시간 인텐트 검사 방법을 사용할 때 컴포넌트가 기대하는 대로 인텐트 명세를 정확히 작성했음을 가정하였다. 하지만 실제로 복잡한 앱의 경우 인텐트 명세를 잘못 작성할 수 있고, 새로운 기능을 추가하고 유지 보수하는 과정에서 컴포넌트 코드에서 기대하는 바와 불일치하게 될 수도 있다. 정적 분석 도구가 있어서 이러한 상황을 미리 탐지할 수 있다면 이 논문에서 제안한 방법을 사용하는데 도움이 될 것이다.

## References


[1] Android APIs, [Online]. Available: http://developer.android.com.

[2] A. K. Maji, F. A. Arshad, S. Bagchi, and J. S. Rellermeyer, "An Empirical Study of the Robustness of Inter-component Communication in Android," *Proc. of the 2012 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) (DSN'12), IEEE Computer society*, Washington, DC, USA, pp. 1-12, 2012.

[3] J. Burns, Intent Fuzzer, [Online]. Available: https://www.isecpartners.com-/tools/mobile-security/intent-fuzzer.aspx, iSEC Partners, 2009.

[4] R. Sasnauskas and J. Regehr, "Intent Fuzzer: Crafting Intents of Death," *Proc. of the 2014 Joint International Workshop on Dynamic Analysis (WODA) and Software and System Performance Testing, Debugging, and Analytics (PERTEA)*, pp. 1-5, San Jose, CA, 2014, ACM.

[5] F. J. Hui Ye, S. Cheng, and L. Zhang, "DroidFuzzer: Fuzzing the Android Apps with Intent-filter Tag," *Proc. of International Conference on Advances in Mobile Computing & Multimedia (MoMM)*, pp. 68-74, Vienna, Austria, 2013, ACM.

[6] M. P. Roee Hay and O. Tripp, "Dynamic Detection of Inter-application Communication Vulnerabilities in Android," *Proc. of the 2015 International Symposium on Software Testing and Analysis (ISSTA)*, pp. 118-128, New York, NY, USA, 2015, ACM.

[7] M. Ko, K. Choi, and B-M, Chang, "A Flexible Intent Fuzzer with an Automatic Tally of Failures for Detecting Vulnerabilities of Android App," Technical Report TR-SEP-2015-3, Sep. 2015.

[8] J. Gosling, B. Joy, G. L. Steel, G Bracha, and A. Buckley, The JavaTM Language Specification, Java SE 8 Edition. Oracle, 2014.

[9] JaeChul Ryu, Eunseon Jo, Moon-Joo Kim, Jin Kwak, Development of Mobile Software Security Testing Tool for Detecting New Android Vulnerabilities, *Journal of Korea Institute of Information security and Cryptology*, Vol. 25, No. 1, pp. 57-59, Feb. 2015.

[10] Sehwan Yeo, Jin Lee, Jungsun Kim, Blocking Harmful Application by Intent Monitoring in the

Android Platofrm, KIISE Transactions on Computing Practices, Vol. 19, No. 5, pp. 273-277, May 2013.
[11] Min Jae Jo, Ji Sun Shin, Study on Security Vulnerabilities of Implicit Intents in Android, *Journal of the Korea Institute of Information Security & Cryptology*, Vol. 24, No. 6, pp. 1175-1184, Dec. 2014.
[12] Joon-Seok Oh, Miyoung Kang, Jin-Young Choi, "Research on Android App API Coding Guide," *Proc. of Korea Computer Congress 2010, The Korean Institute of Information Scientists and Engineers*, Vol. 37, No. 2(C), pp. 129-132, Nov. 2010.
[13] Myungpil Ko, A Design and Implementation of Intent Specification Language for Robust Android Apps, MS Thesis, Computer Science in Yonsei University, Aug. 2015.
[14] Spoon-Processor for Java annotations, [Online]. Available: http://spoon.gforge.inria.fr/processor_annotations.html
[15] R. Meier, *Professional Android 4 Application Development*, 3rd Ed., Wrox, Hoboken, NJ, USA, 2012.
[16] Z. Mednieks, G. B. Meike, and L. Dornin, *Enterprise Android: Programming Android Database Applications for the Enterprise*, John Wiley & Sons, Somerset, NJ, USA, 2013.

고 명 필

2013년 연세대학교 컴퓨터공학과 졸업(공학사). 2013년~현재 연세대학교 전산학과 석사과정. 관심분야는 모바일 소프트웨어, 프로그래밍언어, 컴파일러, 프로그램 분석, 소프트웨어 검증

최 광 훈

1994년 한국과학기술원 전산학과 졸업(학사). 1996년 한국과학기술원 전산학과 졸업(공학석사). 2003년 한국과학기술원 전산학과 졸업(공학박사). 2003년~2005년 JAIST, Researcher. 2005년~2006년 Tohoku Univ., Researcher. 2006년~2010년 LG전자 Mobile Communication 연구소, 책임연구원. 2010년~2011년 서강대학교 컴퓨터공학과 BK21연구교수. 2011년~현재 연세대학교 컴퓨터정보통신공학부 조교수 관심분야는 모바일 소프트웨어, 프로그래밍언어, 컴파일러, 프로그램 분석

창 병 모

1988년 서울대학교 컴퓨터과학과 졸업(공학사). 1990년 한국과학기술원 전산학과 졸업(공학석사). 1994년 한국과학기술원 전산학과 졸업(공학박사). 1995년~현재 숙명여자대학교 컴퓨터과학부 교수. 관심분야는 프로그래밍언어, 프로그램 분석 및 검증, 소프트웨어 보안