APLAS 2014

2014. 11. 17 - 19 @ 싱가포르

서울대학교 이우석

1.개요

올해 APLAS는 싱가포르 국립대학에서 개최되었다. 발표된 논문 수는 24편으로 적은 편이긴 하지만 매우 폭넓은 분야를 다루었다. 우리 연구실에서는 5명의 인원이 4편의 포스터를 발표하기 위해 참석했다.

2.포스터 발표

A Progress Bar for Static Analyzers

Woosuk Lee, Hakjoo Oh, and Kwangkeun Yi

얼마전 SAS에서 발표한 논문을 포스터로 발표하였다. 정적 분석기의 진행율을 예측하는 연구인데 감사하게도 많은 사람들이 흥미있어했다. 학회발표 때 보다

더 날카로운 질문들도 받을 수 있었다. 포스터는 소수의 인원에게 그들이 이해하는 보폭에 맞춰 내용을 전달할 수 있는 장점이 있어서 상대에게 더 많이 이해시킬 수 있고, 덕분에 간혹 더 깊이있는 피드백을 받을 때도 있는 것 같다. 앞으로 포스터 발표 기회가 있을 때 마다 더 심혈을 기울여 준비해야 겠다는 생각을 했다. 덤으로, 투표로 결정된 전체 포스터 2등으로 상을 받아서(부상으로 프레젠터를 받았음) 더욱 감사했다.

3.인상깊은 논문들

A Method for Scalable and Precise Bug Finding Using Program Analysis and Model Checking

Manuel Valdiviezo, Cristina Cifuentes, and Padmanabhan Krishnan

호주 오라클 연구소에서 제작된 분석기에 관한 논문이다. 이들의 목표는 허위경보율 20% 미만의 정확도로 백만라인 이상의 프로그램을 너무 길지 않은 시간에 분석하는 것이다. 기본적인 방침은 모델체킹을 하는 것이다. 성능향상을 위해 오류와 관련없는 프로그램 부분들을 먼저 쳐내고, 남은 부분들에 대해서 진행한다. 이 때 모델체킹이 될 부분을 줄이기 위해서 아주 단순한 형태의 인터벌, 포인터 분석결과를 사용하고, 소프트웨어 모델체킹에서 가장 많은 시간을 잡아먹는 것으로 알려져있는 모델 정교화(abstraction refinement)를 생략한다. 이는 정확도를 어느 정도 포기하면서 성능을 얻는 선택이다. 상향식(bottom-up)으로 함수 간 분석을 한다. 즉, 함수 서머리를 만들어서 분석하는데, 서머리로 만들기 어려운 함수는 분석을 못하고, 이 때문에 안전성을 놓치는 부분이 있다.

결과적으로 백만줄 프로그램 (C++ OpenJDK)을 1시간 20분에 분석하고, 허위경보율은 15%라고 한다. 기술적으로 새로운 것은 없지만 잘 알려진 것들을 잘 버무려서 실용적인 결과를 만드는 데 성공한 것으로 보인다. 이러한 경험들이

잘 축적되고 전달되면 산업현장에서 미탐율과 오탐율 사이에서 줄타기를 하기 위한 디자인 결정들을 내리는데 큰 도움이 될 것이다.

Model-Checking for Android Malware Detection

Fu Song and Tayssir Touili

안드로이드 악성행동을 로직(CTL, LTL 등)으로 명세하고, 모델체킹을 하는 식으로 안드로이드 악성앱을 검출한 연구이다. 로직으로 악성행동을 기술할 수 있기 때문에, 개인정보 누출 이상의 악성행동도 검출할 수 있다는 강점이 있다.

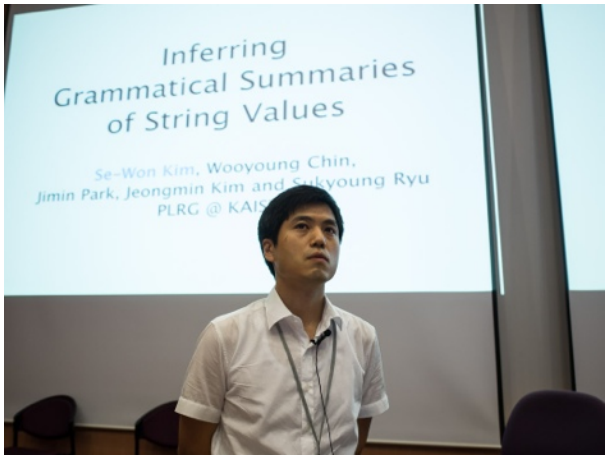
성능이 빠르다. 총 1331개의 앱(악성 1260, 올바른 71)에 대해서 수행된 모델체킹은 수초~수분이내에 끝나고 메모리도 수십MB밖에 쓰지 않는다. 대상 악성행동은 IMEI, IMSI, 전화번호, 주소록 등 개인정보의 누출뿐 아니라 현재 설치된 앱 정보 읽음, 네이티브 코드 실행, 사용자 몰래 영상 촬영 등을 포함한다.

악성행동 명세는 주로 API호출에 관련된 것들이다. 주로 어떤 API호출의 결과물이 어떤 API호출의 인자로 흘러들어가는지로 기술되어있다. API호출들만 살펴보고 값 분석을 안해서 성능이 빠른 것 같다.

그런면에서 악성행동 탐지에는 분명 한계가 있는 방법인 듯 하다. 예컨대 리플렉션을 쓰는 악성앱을 찾기 위해서는 문자열 분석이 필요하다. 하지만 이 방식으로는 그러한 정교한 값 분석을 요하는 악성 앱 탐지가 어려워보인다. 그렇지만 악성 행동을 기술할 수 있는 유연한 틀을 만들려고 시도했다는 점은 의미가 분명 있다. 연구실에 진영이가 이러한 시도를 하고 있는데, Scandal을 이용해서 이 연구보다 분명 더 잘할 수 있을 것 같다.

Inferring Grammatical Summaries of String Values

Se-Won Kim, Wooyoung Chin, Jimin Park, Jeongmin Kim, and Sukyoung Ryu



카이스트 김세원 박사님의 연구결과이다. 레퍼런스 문법이 주어졌을 때, 프로그램 실행 중 생성되는 문자열들을 문법의 비말단(non-terminal)과 말단기

호들로 표현되게끔 요약한다. 레퍼런스 문법은 임의의 문맥자유문법(context free grammar)일 수 있다. 실험은 작은 자바스크립트 프로그램들에 대해서 수행되었고, 1분이내에 사용자가 이해하기 쉬운 형태의 문자열 요약본들이 생성된다.

연구를 진행하면서 부딪힌 문제점과 해결책을 순서대로 설명하는 방식의 발표가 좋았다. 다만 내 배경지식의 부족때문에 기존 연구들과의 관계를 이해하기 어려웠다(LR(k)언어를 대상으로 한 요약 파싱, 문맥자유언어를 대상으로 한 김세원 박사님의 이전 VMCAI'12 논문 등). 후일의 연구를 통해 서로 다른 파싱 알고리즘에 기반한 기존 여러 연구결과들을 일관되게 설명할 수 있는 문자열 분석 틀이 제시된다면 문자열 분석에 한 획을 그을 좋은 연구결과가 되지 않을까 감히 생각해본다.

Syntax-Directed Divide-and-Conquer Data-Flow Analysis

Shigeyuki Sato and Akimasa Morihata

값 흐름 분석(data flow analysis)는 보통 컴파일 시 최적화를 위한 정보를 수집하기 위해 수행되는 요약해석을 말한다. 이 논문은 값 흐름 분석을 병렬화 하는 것에 대한 내용이다. 일반적으로 실행흐름그래프(CFG)로 표현되는 프로그램에 대한 분석은 병렬화 하기가 어렵다. 독립적인 일을 하는 여러부분으로 나누기 어렵기 때문이다. 그에 반해 AST로 표현된 프로그램에 대해서는 분석을 병렬화 하기가 쉽다. 예컨대 if cond e1 e2 에서 e1, e2를 동시에 할 수 있다. 이미 AST에 대해 수행되는 값 흐름 분석을 병렬화 한 시도는 있었는데, goto 구문을 잘 처리하지 못했다. 이 연구는 이를 효과적으로 잘 처리할 수 있다.
도달가능한 변수선언(Reaching definition)을 모으는 간단한 분석을 병렬화 한 결과, 쓰레드 수에 거의 선형으로 비례해서 속도가 빨라지는 것을 확인했다고 한다.
일전에 스패로우를 병렬화 하려는 시도를 한 적이 있어서 관심이 많이 갔으나 디테일에 매몰된 발표방식 탓에 많이 이해할 수는 없었다. 핵심을 잘 캐치해서 스패로우 병렬화에 쓸 수 있으면 좋겠다는 희망을 가져본다.

4.마치며

좋은 자극을 받을 수 있게 지원을 아끼지 않으시는 이광근 교수님과 행정을 도와 주시는 소프트웨어 무결점 연구센터 행정실분들께 매우 감사드립니다.