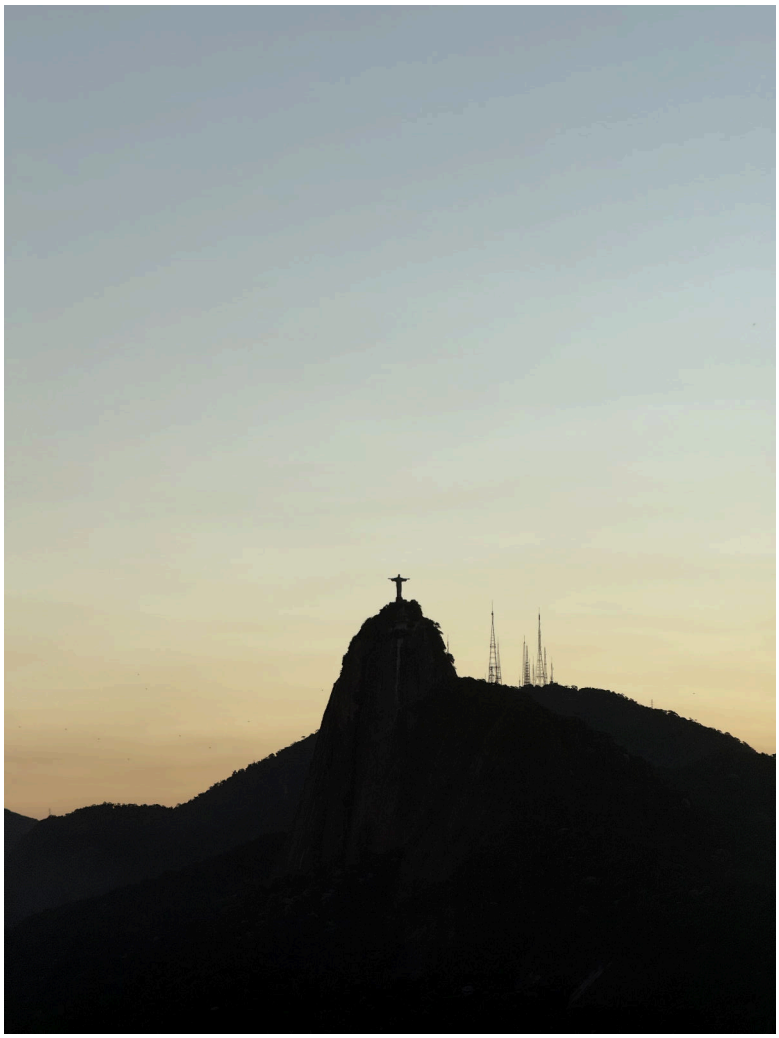
Trip to ICSE 2026

Rio de Janeiro, Brazil



석사과정 허준용

2026.04.13 – 2026.04.20

Software Analysis Lab, Korea Univ.

Contents

들어가며...

개인적으로 굉장히 재밌었고 얻은 것도 많은 여행이었습니다. 보통 잘 가기 힘든 국가를 경험하고 왔고, 연구에 대한 동기부여도 많이 받고 왔습니다. 브라질로 가는 여정은 Road to Brazil 섹션에, 한국으로 오는 여정은 Way Back Home 섹션에 담았고, 학회에 참석한 날들은 학회에서 본 연구에 대한 내용을 담은 At the Conference 섹션, 학회장 밖에서 경험한 것들을 담은 Beyond the Venue 섹션으로 나눠서 작성하였습니다. 좋은 경험을 하게 해주신 오학주 교수님께 감사드립니다.

1 Road to Brazil

평소에 데드리프트나 달리기를 좀 해서 그런가 장시간 비행은 견딜만 했다. 14시간의 비행 후 도착한 애틀란타, 브라질행 비행기까지 4시간 정도를 기다려야 했는데 컴파일러 과제 채점을 하다보니 시간이 금방 갔다. 다만... 이륙이 자꾸만 지연되다가 비행기에 탔는데 불이 몇번 꺼졌다 켜지더니 기체 이상으로 다음날 오전에 출발한다고 호텔 바우처를 줄테니 자고 오란다. 비행기에 탔다가 내렸는데 같은 장소인 경험은 정말 처음이었다. 나중에 듣길 불이 꺼졌다 켜지는게 시동이 꺼져서 비상 전원이 켜지는 거라고 한다. 당시가 새벽 2시쯤이었고 호텔에 도착하니 나같은 사람들이 체크인을 하려고 줄을 서 있었다. 이게 줄까지 설 일인가 했는데 호텔 직원이 거의 10분당 한 팀씩 처리해서 거의 1시간 동안 줄 서 있었다. 비행기에서 주토피아를 봤는데 거기 나오는 나무늘보 공무원 수준이었다... 그렇게 들어간 방... 복도에 담배 냄새가 진하게 나서 설마 했는데 역시 방에서도 은은하게 담배 냄새가 났다 (금연 호텔인데!!!). 2시간 정도 잠을 자고 나가니 공항에 가려는 사람들이 호텔 입구에서 두리번 거리고 있었다. 뭐 거의 예비군 훈련장 가는 것 마냥 4인 우버팟이 급조되어 공항으로 향했다. 우버 돈 내주신 이름 모를 아저씨... 감사합니다.

애틀란타 공항 주변 Motel 6는 거르자!!!

비행기를 다시 탔는데 불이 또 몇번 꺼졌다 켜졌다. 그러다 결국 출발을 한댄다. 무슨 경운기 시동 걸듯이 걸릴 때까지 땡기는 것 같다는 느낌을 받았다. 뭐 그렇게 9시간 별일없이 날아서 브라질에 잘 내렸다. 우버를 타고 숙소에 도착한 후, 마트에 들러 맥주와 소세지를 사서 맛있게 먹었다. 소세지가 아주 맛있었다. 그 때가 밤이었는데, 숙소 주변엔 다행히 강도가 없었다. 브라질에서의 경험이 아주 즐거웠는데, 가는 길에 액땜을 왕창 해서 그랬나 보다.

2 First Day of ICSE

리우 해변의 이국적인 풍경이 반겨준 아침... 낯이 익다 했더니 GTA 모바일 "갱스터 리오: 성자의 도시"에서 질리도록 봤던 풍경이다.

숙소에서 바라본 해변

2.1 At the Conference

학회장이 아주 잘 꾸며져 있었다. 명찰을 받았는데 울퉁불퉁한 종이에 인쇄되어 있어서 '퀄리티가 뭐이래...'라고 생각했는데 당근 씨앗으로 만들어서 땅에 심고 물주면 당근이 자란단다. 이렇게 환경을 생각하는데 AI for Software Engineering 세션이 29개... 하긴 GPU 많이 쓰니 종이랑 플라스틱이라도 덜 써야지. 안그래도 숙소에서 신을 슬리퍼가 필요했는데 기념품으로 쪼리까지 줘서 잔뜩 치솟은 호감도와 함께 개막식장에 들어섰다.

2.1.1 Jan Bosch Keynote: Towards an Awesome AI-driven Future for Software Engineering

AI가 대세인 요즘 답게, 앞으로 AI가 바꾸어갈 소프트웨어 엔지니어링의 미래에 대한 키노트로 학회가 시작되었다. 내용을 요약하면,

익씨의 개막식. 생각보다 본격적이어서 놀랐다.

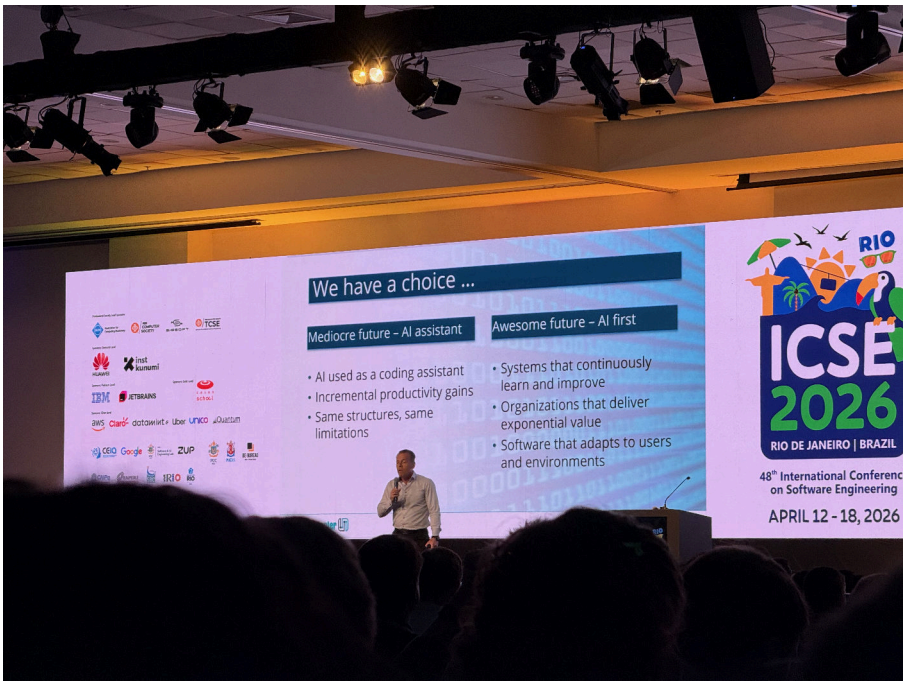
이래 봬도 당근 씨앗이다.

- 사람이 솔루션 구현 → 사람이 의도를 명시하면 시스템이 솔루션 구현
- 사람이 테스팅 → 시스템이 셀프 테스팅
- 사람이 배포 → 시스템이 셀프 배포
- 사후 대응식 유지보수 → 스스로 계속 발전하는 시스템
- 사람이 시스템을 운영 → 시스템이 스스로 운영
- 코드가 상품 → 스스로 학습해서 위와 같이 동작하는 시스템 (이하 Learning System)이 상품

이렇게 변화할 것이고 이에 따른 연구 도전 과제는

- Learning System을 어떻게 만들 것인가?
- 사람은 Learning System과 어떻게 협력할 것인가?
- Learning System이 잘 동작하다는 보장을 어떻게 얻을 것인가?

라고 한다. LLM에게 목적에 맞는 툴을 쥐여주는 에이전트 연구가 활발한 걸로 보아 이미 저 방향으로 나아가고 있는 것 같다. 테스팅, 리페어, 정적 분석 알람 검증 등 여러 목적의 에이전트가 나오고 있고 이러한 에이전트들로 Learning System이 구성되지 않을까. 뻔한 내용이지만 웬 동기부여 영상에 나올 것 같은 근육질 아저씨가 강렬하게 발표를 해서 집중해서 들었다.



Jan Bosch. 파워풀한 발표가 인상 깊었다.

2.1.2 BFix: Automated Safe Memory-Leak Fixing for Binary Code

꽤 오랫동안 메모리 오류를 연구했어서 메모리 릭에 눈길이 가 발표를 들었다. 바이너리 레벨에서 메모리 릭을 고치는 최초의 연구다. 기존 기술들은 모두 소스코드 레벨에서 동작한다. 코드가 공개되지 않은 경우, 툴체인이 오래되어 컴파일이 불가능한 경우, 당장 배포해야 해서 다시 빌드할 시간이 없는 경우 등

소스코드를 이용할 수 없는 경우를 위한 연구라고 한다. 바이너리 코드를 전처리해서 콜 그래프와 CFG를 얻고 메모리 릭이 발생하는 경로를 찾아 `free`를 삽입하는 방식으로, 기존 소스코드 레벨 연구들과 흐름이 유사하다. 다만 바이너리 코드 특성상 다음과 같은 문제들이 생긴다.

- 삽입된 `free`가 기존의 레지스터 값을 바꿔버릴 수 있음
- `free`를 해줘야 할 포인터의 정보가 사라짐
- 조건에 따라 동적 할당이 되는 경우,
  삽입된 `free`가 할당이 일어나지 않은 경로에서 오류를 일으킬 수 있음

이에 대한 해결책으로, 삽입된 `free`호출 이전의 레지스터 값을 스택에 저장 후 호출이 끝나면 복원하고, 사라질 수 있는 포인터 정보도 스택에 저장해 놓으며, 할당 함수(`malloc` 등)에 의해 반환된 포인터만 `free`하도록 한다. 평가 결과 기존의 소스코드 레벨 연구보다 더 많은 메모리 릭을 고쳤고, 오버헤드도 작았다고 한다. 기존 기술 중에 우리 연구실에서 만든 SAVER도 있어서 반가웠다. 사실 솔루션이 학부 과제 수준으로 뻔하다고 생각했다. 최초인 게 의미가 있는 것 같다.

2.1.3 On the Flakiness of LLM-generated Tests for Industrial and Open-Source Database Management Systems

요즘 연구에 LLM을 사용하고 있는지라 LLM이 얼마나 잘하는지 궁금해서 발표를 들었다. LLM이 작성한 데이터베이스 시스템에 대한 테스트 중에 flaky 테스트 비율을 조사한 연구다. Flaky 테스트란 테스트 대상 코드와 테스트 코드가 바뀌지 않았음에도, 반복 실행 시 통과와 실패가 비결정적으로 모두 나타나는 테스트를 말한다. 즉, flaky 테스트가 많다는 것은 테스트를 신뢰할 수 없다는 것을 의미한다. 연구 결과는 flaky 테스트 비율이 1% 미만으로 개발자가 작성한 테스트 중 flaky 테스트 비율과 비슷하다고 한다. 이제 LLM이 충분히 잘하는 것 같다.

2.1.4 Let the Trial Begin: A Mock-Court Approach to Vulnerability Detection using LLM-Based Agents

LLM 기반 에이전트 연구에 관심이 있어서 발표를 들었다. 여러 에이전트를 이용해 법정 재판을 모사함으로써 취약점을 탐지하는 도구인 VULTRIAL을 제안한다. 사용된 에이전트는 다음과 같다.

- Security Researcher : 검사 역할. 주어진 코드를 보고 취약점 후보를 근거, 예상되는 위험과 함께 Code Author에게 보고함
- Code Author : 변호사 역할. Security Researcher에게 보고 받은 취약점 후보가 타당한지 판단하고 그 결과를 근거와 함께 Moderator에게 보고함
- Moderator : 판사 역할. Security Researcher와 Code Author의 보고를 요약함
- Review Board : 배심원 역할. 위 세 에이전트의 보고 및 요약을 보고 취약점 후보의 타당성에 대한 최종 판단을 내림

Security Researcher → Code Author → Moderator 순서로 한 바퀴 도는 걸 1 round라고 하자. VULTRIAL은 n round 이후 Review Board가 최종 판단을 내리며, 두번째 round부터는 Security Researcher도 다른 에이전트의 보고 및 요약을 볼 수 있다. 평가는 (vulnerable function, patched function) 쌍으로 이루어진

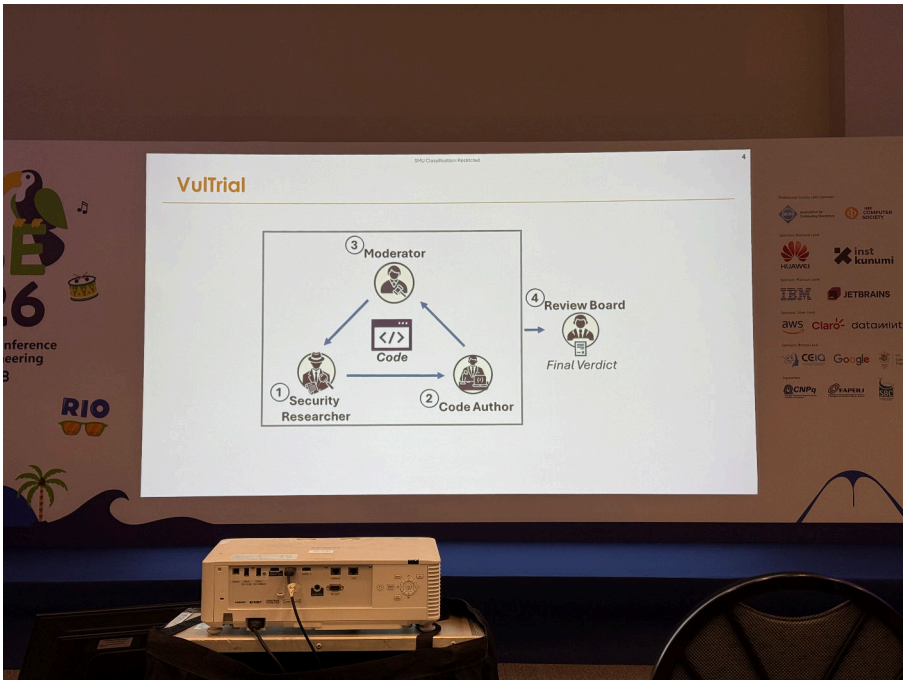


Figure 1: VulTrial의 파이프라인

PrimeVul 데이터셋을 이용해 이루어졌다. 평가 메트릭은 Pair-Correct로 함수 쌍의 vulnerable version, patched version 둘 다에 대해 정확한 판단을 내려야만 정답으로 인정된다. 평가 결과는 기존의 딥러닝 기반, LLM 기반 기술들을 모두 압도했다. 평가 결과에서 Code Author, Moderator에 대한 ablation study 결과도 확인할 수 있는데 둘 다 유의미하게 성능에 기여한다. 주목할 만한 점은 다른 에이전트의 보고를 요약하기만 하는 Moderator가 없으면 성능이 절반 미만으로 떨어진다는 것이다. 판단의 재료가 되는 내용의 핵심만 남기는게 LLM의 정확도에 도움이 되는 걸까. 또한, 1 round만 돌렸을 때가 성능이 가장 좋고 round가 많아질수록 성능이 떨어졌다. Round가 반복될수록 Security Researcher 또는 Code Auther가 설득 당해 에이전트들의 판단이 한쪽으로 치우쳐 오판이 늘어난다고 한다. 재밌기도 하고 LLM을 잘 쓰는 방법에 대한 인사이트를 얻을 수 있어서 유익했다. 다만, 이 연구에서 말하는 에이전트는 프롬프트로만 동작한다. 즉, 요즘 우리 연구실에서 관심을 가지는 툴을 장착한 에이전트가 아니다.

2.1.5 TestWeaver: Execution-aware, Feedback-driven Regression Testing Generation with Large Language Models

테스팅 분야에선 LLM을 어떻게 사용하고 있는지 궁금해서 발표를 들었다. 한 프로젝트에서 기존의 테스트가 커버하지 못하는 라인(이하 타겟 라인)들을 커버하기 위해 LLM에게 테스트 작성을 시키는 연구다. 코드베이스에는 타겟 라인과 관련 없는 부분이 많고, LLM이 코드를 보고 테스트를 만들 때 해당 부분 때문에 혼란에 빠져 성능이 떨어진다는 관찰에 기반한다. TestWeaver는 이 문제를 해결하기 위해 타겟 라인과 관련 없는 부분을 잘라버리는 프로그램 슬라이싱을 수행한다. LLM은 슬라이싱된 코드 덕분에 타겟 라인에 집중하면서 테스트를 작성할 수 있다 (Test Generation). 이렇게 작성된 테스트가 실행되고 난 후, 여전히 커버되지 않은 라인들을 커버하기 위한 또 다른 어프로치가 적용된다. 타겟 라인과 가장 가까이 간 테스트를 찾고 그 테스트의 각 statement 옆에 변수들이 런타임에 가졌던 값을 주석으로 표시해 준다. LLM은 주석이 달린 테스트를 보고 테스트를 다시 작성한다 (Test Re-Generation). 기존 테스트의

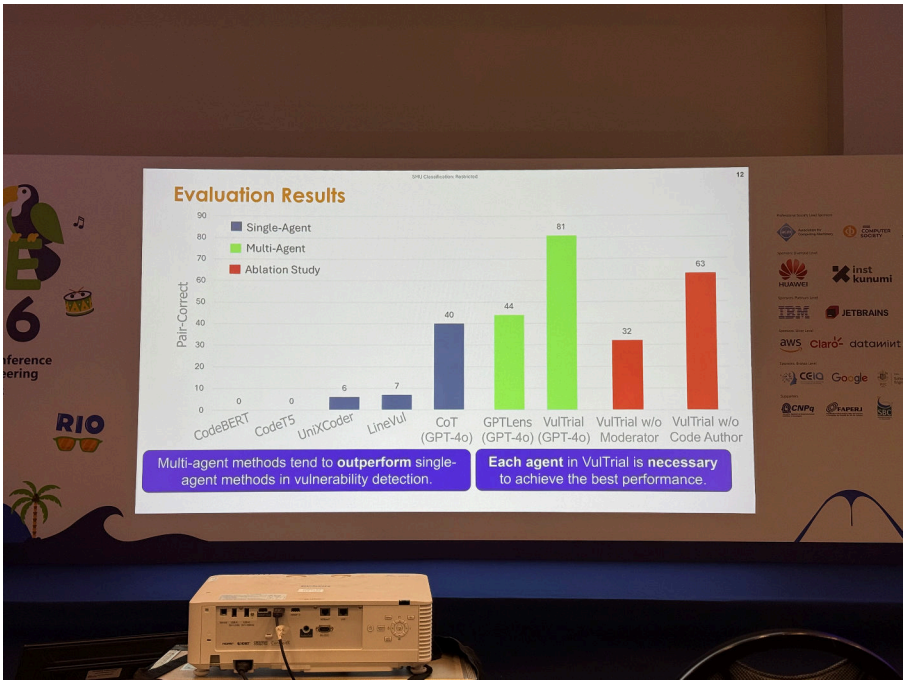


Figure 2: VulTrial의 평가 결과

변수값들을 보고 LLM은 어디를 바꿔야 타겟 라인을 커버할 수 있을지 파악할 수 있다. TestWeaver는 커버되지 않은 라인들을 커버하기 위해 Test Generation → 실행 → Test Re-Generation → 실행을 반복한다. 평가 결과는 기존의 LLM 기반 테스트 작성 연구보다 더 높은 커버리지를 달성했다. VulTrial과 마찬가지로 LLM을 잘 쓰는 방법에 대한 인사이트를 얻을 수 있었다. VulTrial에서도 LLM에게 들어가는 인풋을 요약하는 게 성능에 크게 기여했고, 이 연구에서도 인풋을 프로그램 슬라이싱으로 요약함으로써 좋은 결과를 얻었다. 다만, 귀국 후 연구실에서 원석이가 TestWeaver가 안돌아간다고 불평하는 소리를 듣긴 했다...

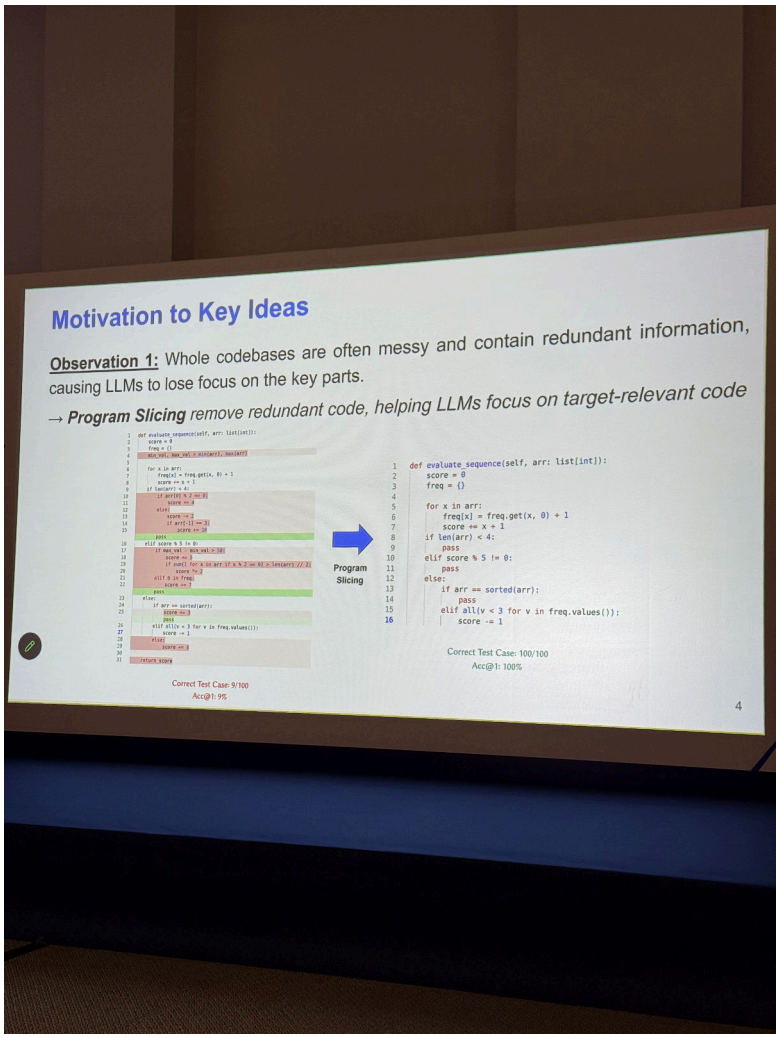


Figure 3: TestWeaver의 프로그램 슬라이싱. 관련 없는 부분을 pass로 치환한다.

2.2 Beyond the Venue

학회를 어느정도 본 후, 우버를 타고 리우 데 자네이루 대성당으로 향했다. 거대한 원뿔형 건물이었는데 시멘트로 지어져서 외관은 투박했지만 그 크기만으로 랜드마크라 할만 했다. 건물 내부엔 사방에 스테인드글라스가 높은 내벽을 타고 장식돼 있었는데, 스케일은 경이로웠지만 시멘트 뼈대가 그물망처럼 쳐져 있어서 살짝 아쉬웠다.

리우 대성당의 단상. 실제로 미사가 열린다고 한다.

성당을 둘러보고 셀라론 계단으로 향했다. 타일을 붙여 장식한 예술 계단이라고 한다. 원래 우버를 타는게 맞지만 가까운 거리고 가는 길이 대로변이라 걸어가기로 했다. 가는 길에 다리가 있었는데 나중에 듣길 노숙자가 많아 위험한 곳이라고 한다. 대낮에도 강도를 당한다고 하는데... 뭐, 살았으니 좋은 전략이

리우 대성당 천장 뷰. 스테인드글라스가 십자가 모양이다.

위험한 다리. 리우에선 door-to-door로 우버를 타는 게 권장된다.

었다. 그렇게 도착한 셀라론 계단엔 사람이 아주 많았고 형형색색의 타일들로 잘 꾸며져 있었다. 무엇보다 내 눈길을 끈 건 기타를 치면서 노래하고 있는 사람들이었는데, 역시 삼바의 나라 답게 기타 리듬이 남달랐다. 거기에 노래까지... 나도 기타를 치는데 더 정진해야겠다. 계단을 끝까지 올랐다가 내려올 때 맥주를 한 잔 마셨는데 크... 정말 최고였다.

예술 계단의 예술가. 팬서비스가 좋다.

다음 목적지는 Sugarloaf Mountain, 한국인에겐 빵산이라 불리는 바위산이다. 케이블카를 타고 올라갈 수 있고 리우의 전경을 볼 수 있다. 우리는 야경을 보기 위해 일부러 일몰 시간에 맞춰서 갔다. 올라갈 때도 그렇고 내려갈 때도 그렇고 줄을 오래 서야 했지만 리우를 한 눈에 담을 수 있어서 만족스러웠다. 리우가 반짝이는 만큼 밤하늘도 별들로 반짝이고 있었다. 서울에선 이제 볼 수 없는 풍경이었다.

초록과 어우러져 반짝이는 리우

빛나는 예수상. 예수님의 가호 아래 있는 도시인데 범죄율이 높다니 아이러니하다.

빵산에서 내려온 후 슈하스코를 맛있게 먹고 숙소로 가서 쉬었다.

3 Second Day of ICSE

아침에 일어나 창 밖을 보니 ICSE 사람들이 뛰고 있었다.

브라질에서도 ICSE는 달린다...

3.1 At the Conference

Distinguished Artifact 상을 받는 멋있는 원석이와 함께 하루를 시작했다.

3.1.1 Panel - SE.next: Research in the Agentic Era

에이전트 시대에서의 연구에 대해 교수님들이 이야기를 나누는 세션이었다. 교수의 역할이 어떻게 변할지에 대한 의견을 먼저 나누었는데, 정리하면 AI로 인해 지식이 democratized됐기 때문에 지식을 수직적으로 전수하고 명령하는 supervisor 역할에서 학생들이 잘 나아갈 수 있도록 조언해주는 advisor 역할로 변한다는 것이다. Advisor가 하는 일을 debugging people이라고 표현하는게 인상깊었다. 되게 뻔하고 누구나 다 아는 소리를 한다고 생각했는데 저번 겨울 SIGPL이었나, 오학주 교수님께서 자기는 학생들이 뭔가를 해오면 피드백을 주는 agentic research를 한다고 하신 적이 있다. 사실 저게 교수님의 agentic research 맥락과 같고, 이미 그걸 겪고 있어서 그렇게 느꼈나보다. 관객과 소통하는 시간도 있었는데, 한 학생이 AI의 발전으로 세상이 너무 빨리 변해서 본인이 풀고있는 문제가 사라질 수도 있다는 생각에 불안하다는 이야기를 했다.


Wonseok Oh, Hakjoo Oh
A Replication Package for TypeCare: Boosting Py
Inference Models via Context-Aware Re-Ranking
Augmentation

부러운 쉑


SE.NEXT: Research in the Agentic Era
Andreas Zeller
Abhik Roychoudhury
Carolyn Seaman
Daniel Russo
Patanamon Thongtanunam
Lingming Zhang
RIO
CSE
2026
DE JANEIRO | BRAZIL
International Conference
Software Engineering
RIL 12 - 18, 2026

SE 권위자들

- 트렌드를 읽고 미래에 문제가 될 만한 걸 풀어라.
- Long-term direction을 가지고 나아가라. 그리고 아이디어를 내든 방향을 제시하든 기여를 명확히 분류하는게 중요한데, 기여엔 기술적 기여와 개념적 기여가 있다. 기술적 기여만 있고 개념적 기여는 없는 거엔 시간을 많이 투자하지 마라.
- 문제가 없어지더라도 연구해온 결과에서 얻을 건 분명히 있을 것. 새로운 게 계속 나오더라도 지속하라.
- 용기를 가져라. AI가 진입 장벽을 치워줘서 아이디어가 있으면 뭐든 연구할 수 있다. Exciting한 문제를 상상하라.

이러한 답변들이 나왔는데, 개인적으론 새로운 아이디어의 기여를 명확히 하고 개념적으로 기여하는 거에 집중하는 전략이 되게 좋다고 생각한다. 마지막 답변은 사진의 맨 오른쪽 링밍 장 교수님이 하신 것인데 처음에 교수의 역할이 연구실의 학생들이 사랑을 느끼고 서로 유대하게 만드는 거라고도 하시고... 참 낭만파다.

3.1.2 StorFuzz: Using Data Diversity to Overcome Fuzzing Plateaus

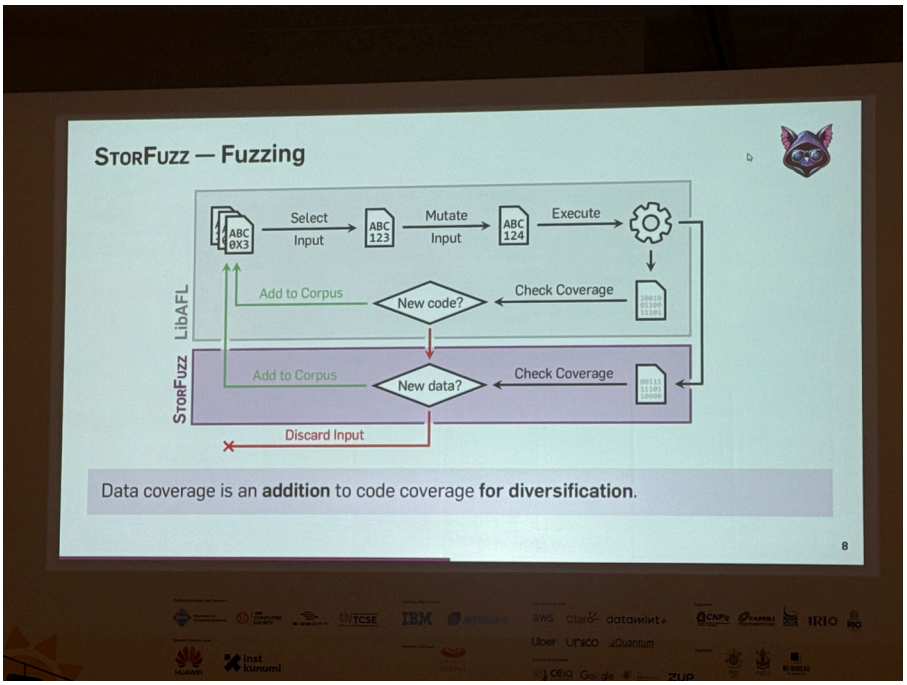


Figure 4: StorFuzz의 파이프라인

이번 ICSE에서 fuzzing이란 걸 머릿속에 잘 정리할 수 있었는데, 이 연구가 그 계기가 된 연구이다. Fuzzing은 테스트되지 않은 영역이 있음에도 커버리지가 더이상 증가하지 않는 plateau에 도달하기 마련이다. 이 연구는 *데이터 커버리지*라는 새로운 개념을 도입해 plateau를 극복하고 더 많은 영역을 테스트하고자 한다. 아이디어는 간단하다. Store 인스트럭션마다 런타임에 저장되는 값을 알 수 있는 계측을 하고, 저장되는 값의 다양한 정도를 데이터 커버리지로 정의한다. 파이프라인에 나와 있듯이 StorFuzz는 기존의 코드 커버리지를 베이스로 동작한다. 코드 커버리지가 증가시킨 시드를 시드풀에 넣는다. 다만, 코드 커버

리지가 증가하지 않았을 때 데이터 커버리지의 증가 여부를 확인하고 증가했으면 시드풀에 넣고 그렇지 않으면 버리는 추가적인 과정을 가진다. 평과 결과는 기존 기술보다 더 빨리 plateau를 뚫어버리고 더 많은 코드를 커버한다.

간단한 아이디어임에도 임팩트가 세다. 이런 게 패널 토크에서 나온 개념적 기여의 극이 아닌가 싶다. 다음은 학회 동안 fuzzing에 대한 생각을 정리한 내용이다. Fuzzing은 커버리지, 뮤테이션, directed fuzzing이라면 타겟과 시드가 얼마나 가까운지를 의미하는 점수 (혹은 distance), 이렇게 세 컴포넌트로 이루어져 있다. 이 연구는 fuzzing의 컴포넌트 중 커버리지에 관한다. 커버리지란 무작위 실행 도중 다양하게 얻고 싶은 대상이다. 예를 들어, 브랜치 커버리지는 다양한 브랜치를 커버하는 걸 목표로 하기 때문에 커버한 브랜치의 수로 정의되고, 데이터 커버리지는 다양한 데이터를 얻는 게 목표이기 때문에 변수와 해당 변수에 저장된 데이터 집합의 매핑으로 정의된다. 물론, 커버리지는 버그를 찾는데 도움이 되어야 한다. 이렇게 생각을 정리하면서 앞으로 하고 싶은 연구가 많이 떠올랐고 fuzzing에 대한 자신감도 붙었다. 뮤테이션에 관한 내용은 후술하겠다.

3.1.3 Is Call Graph Pruning Really Effective?

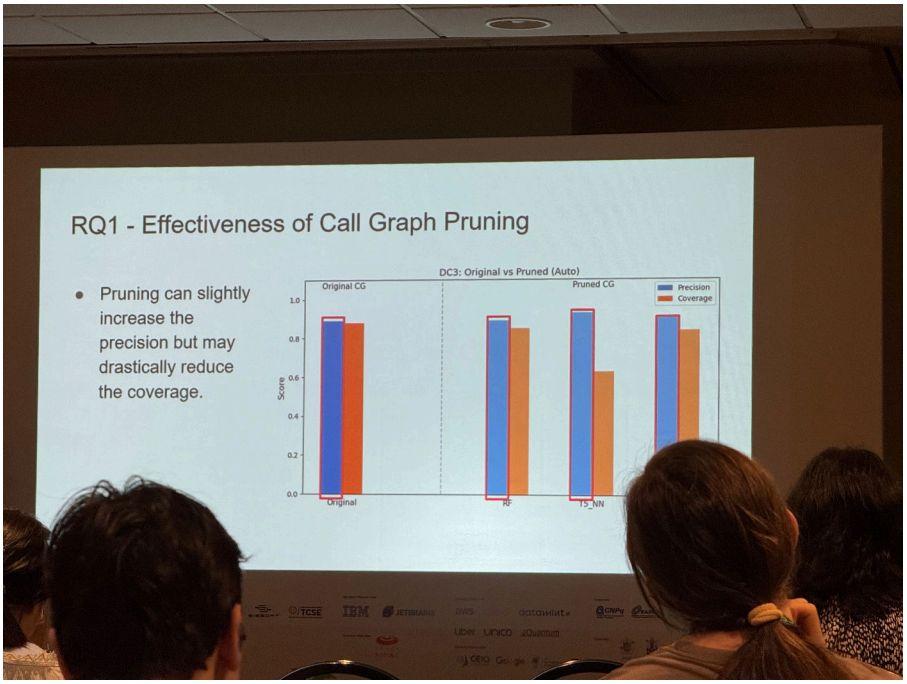


Figure 5: 기존 콜 그래프 프루닝 기술들을 제대로 평가한 결과. 오히려 떨어진 지표도 있다.

평소 콜 그래프를 쓰면서 over-approximation으로 많이 고통 받는지라 콜 그래프 프루닝이란 용어에 꽂혀서 발표를 들었다. 이 연구는 정적으로 구한 콜 그래프에서 머신러닝을 이용해 false positive를 잘라내는 기술들이 정말로 효과적인가를 반문한다. 해당 기술들에서 가장 문제가 되는 점은 사용한 데이터셋의 ground truth (GT)가 동적으로 구한 콜 그래프라는 것이다. 다시 말해, 동적 실행에서 관측된 edge만 true라고 취급한다. NJR1이라는 데이터셋의 경우 true edge가 5.7%이고 나머지는 모두 false edge인데, 저자들은 그 중에서 의미있는 엣지만 남기고 통계적으로 유의미하게 일부를 샘플링 해 매뉴얼하게 라벨링을 다시 했다. 그 결과 동적으로 구한 true edge + 매뉴얼하게 라벨링 한 true edge가 65%였고, false edge가 1.1%, 나머지는 unknown edge로 남긴 데이터셋이 완성됐다. 기존의 데이터셋과 다시 라벨링 한

데이터셋의 true edge 비율 차이가 크다. 애초에 라벨에 문제가 있는 데이터셋을 가지고 학습하고 평가를 했기 때문에 기존의 머신러닝 기반 콜 그래프 프루닝 기술들이 효과적일 수 없다고 주장하는 것이다. 그 증거로 저자들이 바로 잡은 데이터셋에 기존 기술들을 적용했을 때 성능이 크게 오르지 않고 개선의 여지가 많이 남아있음을 보였다. 내 연구와는 크게 관련이 없었지만 내용이 흥미로웠고 기존 기술들의 문제점을 지적하는 것도 연구가 될 수 있다는 걸 알게 되어서 유익했다.

3.2 Beyond the Venue

이 날 저녁엔 뱅퀫에 참석했다. 다른 학교 분들이 음식도 맛없고 재미도 없어서 금방 나왔다고 해서 갈지 말지 고민했지만 원석이가 익씨 뱅퀫은 한 번쯤 가볼만 하다고 해서 가기로 했다. 학회장에서 대절한 버스를 타고 30분 정도 가니 한 클럽에 도착했다. DJ가 음악을 틀고 있고 뷔페와 바가 있었다. 바에서 보드카와 베리로 만든 칵테일을 받고 뷔페를 먹었는데 확실히 음식은 맛없었다. 뱅퀫장을 둘러보다가 발표를 인상 깊게 들었던 StorFuzz 저자를 발견해서 말을 걸지 말지 고민하다가 결국 가서 말을 걸었다. 이름이 Leon 이었다. 통성명을 하고 연구를 칭찬하면서 내가 발표를 듣고 떠올렸던 아이디어를 말해봤다. StorFuzz는 변수에 저장되는 데이터를 다양화 함으로써 더 높은 커버리지를 달성한다. 그럼 다양화 할 데이터의 종류를 제한하면 fuzzing을 특정 버그에 특화시킬 수 있지 않을까? 버퍼 오버플로우를 잡고 싶으면 인덱스 변수와 관련된 데이터의 다양성만 고려하고, 타입 오류를 잡고 싶으면 타입 데이터의 다양성만 고려하고... 이걸 열심히 영어로 설명했지만 잘 못알아들은 것 같다. Leon과 같이 있던 무리 속에서 지금 하고 있는 연구 얘기, 연구실 생활, 브라질에 대한 잡담을 하다가 조금 지쳐서 빠져나왔다. 정말 영어 회화를 공부해야겠단 동기 부여를 가득 받은 것 같다.

뱅퀫이 열린 클럽 입구

처음엔 DJ 앞 스테이지에 아무도 없었는데 사람들이 나와서 춤을 추고 있었다. 학회의 주축이 되는 분들이 나이가 있어서 그런가 살짝 올드한 팝송들이 나왔다. 익씨는 전통적으로 이런 파티 분위기의 뱅퀫을 한다고 한다. 학회에선 진지하게 학문에 몰두하다가 파티에선 신나게 춤 추고 있는 모습이 되게 보기

좋았다. 여기 있는 사람 대부분이 박사 또는 박사과정이라고 생각하니까 새삼 신기했다. 세상에서 가장 고학력인 파티가 아니었을까... 좀 있으니까 초청된 삼바팀이 내려와서 삼바 공연을 하고 갔는데 나도 이때 한바탕 흔들고 숙소에 돌아가서 골아떨어졌다.

신바람 n박사

4 Last Day of ICSE

전 날의 여파로 일어나기 힘들었다...

4.1 At the Conference

같이 간 원석, 미령이가 발표하는 날. 원석이를 알아보는 팬도 있었다;

4.1.1 On Interaction Effects in Greybox Fuzzing

이 연구는 fuzzing의 컴포넌트 중 뮤테이션에 관한 연구다. 사실 대부분의 fuzzing 연구에서 AFL 또는 AFL++의 뮤테이션을 그대로 쓰고 있다. 비트 하나 뒤집기, 한 블락을 다른 블락으로 덮어쓰기 등 여러 뮤테이션 방법(이하 뮤테이터)이 있고 각각이 수행될 확률이 부여돼 있다. 개인적으로 fuzzing 연구를 하면서 뮤테이션 자체를 건드는 연구는 잘 없다고 느꼈는데 이 연구에서 참고하는 기존 연구는 MOPT라고 타겟 프로그램을 테스트 할 때 커버리지를 올리는 시드(이하 interesting input)가 더 많이 생성되도록 개별 뮤테이터가 수행될 확률을 적응시키는 기술이다. 이 연구는 여기서 더 나아가 두 뮤테이터가 연달아 실행됐을 때 생성된 interesting input의 수를 측정해서 두 뮤테이터의 상호작용 여부를 알아낸다. 측정

발표 원석 ㅋㅋ

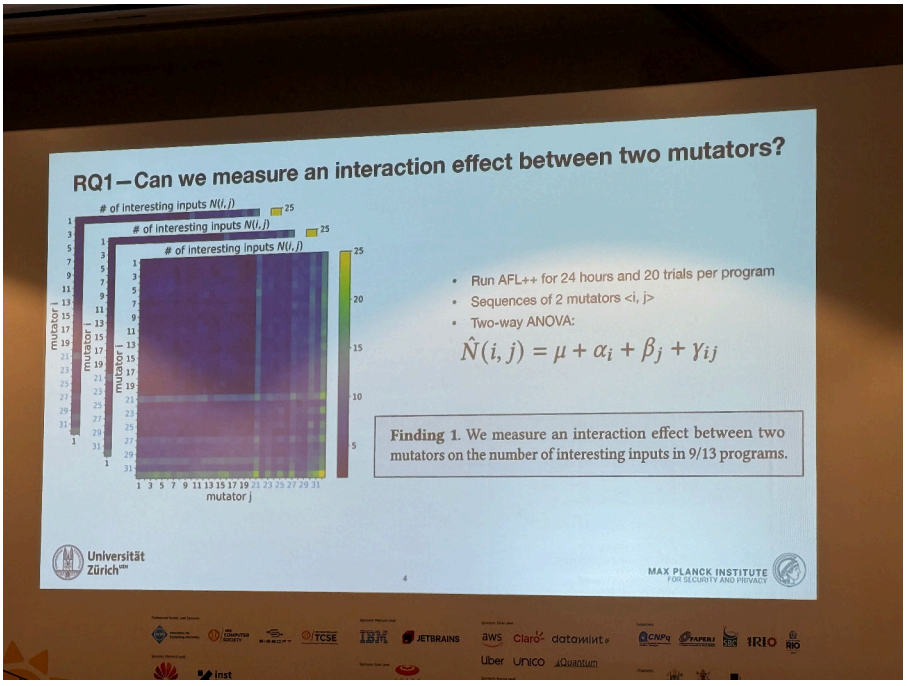


Figure 6: 측정된 두 뮤테이터 간 상호작용의 효과

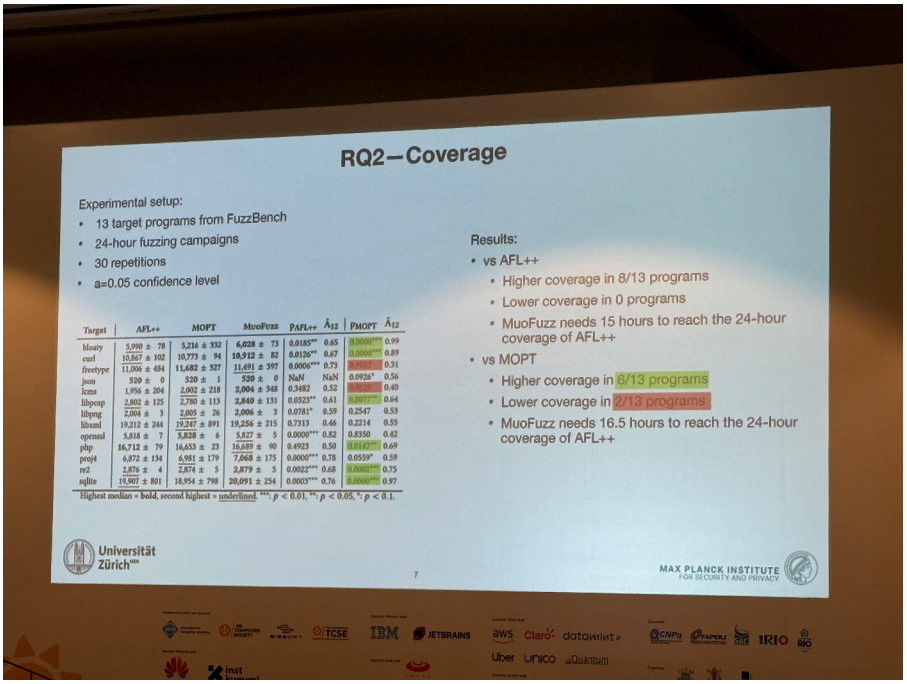


Figure 7: MuoFuzz와 기존 기술의 커버리지 비교

결과, 연달아 실행됐을 때 효과가 좋은 조합이 분명히 관측된다. 또한 이 연구는 관측된 효과적인 뮤테이터 조합을 우선시하는 fuzzer인 MuoFuzz를 제안하는데, 평가 결과 기존 기술인 MOPT보다 6개의 벤치마크 프로그램에서 더 높은 커버리지를 달성했고 2개의 벤치마크 프로그램에서 더 낮은 커버리지를 달성했다. 사실 기존 기술보다 성능이 크게 좋아진 건 아니다. 다만 뮤테이터의 조합을 처음으로 고려했기 때문에 개념적으로 크게 기여한 연구라고 생각한다. 저자 중에 AFLGo를 만든 Marcel Böhme이 있는데, directed fuzzing이라는 breakthrough를 가져왔듯이 이번에도 여러개의 뮤테이터를 조합한다는 breakthrough를 가져와서 여기서 많은 연구가 파생될 수 있을 것 같다.

위에서 말한 fuzzing 이야기를 계속 하자면, 뮤테이션이란 컴포넌트는 커버리지를 직접적으로 올릴 수 있는 수단이다. 하지만 내가 지금 몸을 담고 있는 directed fuzzing 연구판을 보면 커버리지 및 시드 점수에는 집중을 많이 하는 것 같은데 뮤테이션은 아직 AFL++ 수준에 머물러 있는 것 같다. 이 발표를 듣고 일반적인 fuzzing이든 directed fuzzing이든 뮤테이션을 개선시킬 많은 아이디어가 떠올랐고, 앞으로 그걸 실현할 생각에 설레였다.

4.1.2 Locus: Agentic Predicate Synthesis for Directed Fuzzing

이번 학회에서 마지막으로 들은 발표다. 에이전트, directed fuzzing, 내가 관심 있는 두 가지가 제목에 공존해서 무조건 들어야겠다고 생각했다. 다음은 Locus의 파이프라인이다. 이 연구에선 취약점을 일으키는 조건문을 canary라고 하는데, 이미 주어져 있는 canary를 받아서 LLM 에이전트가 프로그램에서 canary가 충족되는데 관련 있는 부분을 뽑고 해당 부분으로 가지 않으면 프로그램을 조기에 종료시키는 predicate을 삽입한다. Canary가 충족될 수 있는데 프로그램이 조기 종료되면 안되기 때문에 canary가 true이면 predicate도 true여야 한다. Locus는 이러한 predicate의 유효성을 기호 실행으로 확인한다. 만약 유효하지 않으면 기호 실행 결과를 이용하여 특정된 canary와 관련 있는 부분을 다듬고 다시 predicate을 합성한다. 실험 결과는 비교 대상 fuzzer들보다 훨씬 빨리 타겟 버그를 찾았다. 사실 발표 내내 에이

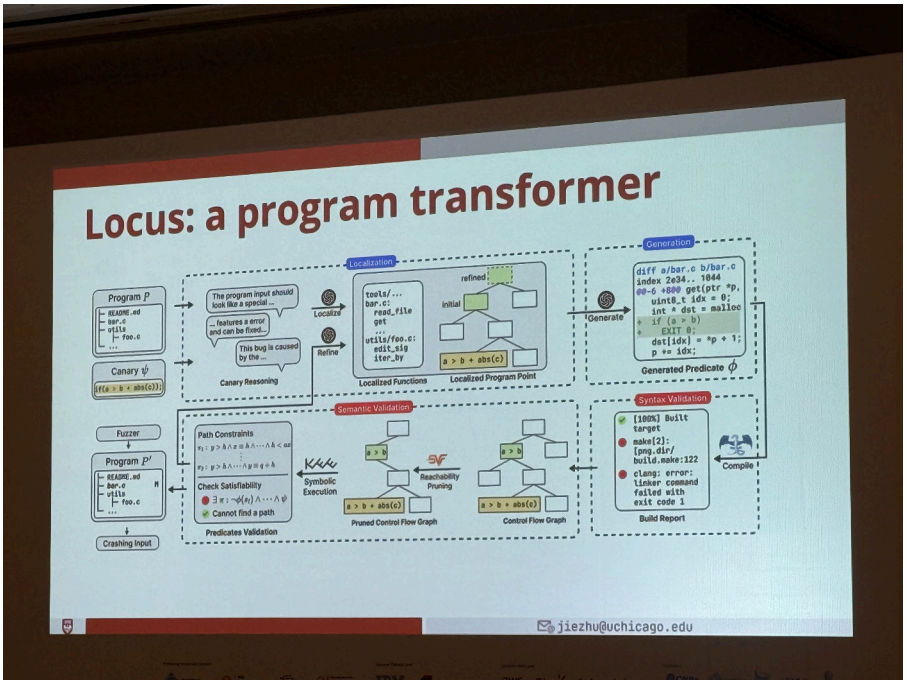


Figure 8: Locus의 파이프라인

전트 얘기는 1도 없어서 실망했으나 나중에 논문을 찾아보니까 정말 툴을 장착한 에이전트를 사용하는게 맞아서 다시 흥미를 느꼈다. 나도 directed fuzzing 연구를 하고 있는데 내가 쓰는 벤치마크엔 canary가 주어지는게 아니라 타겟 소스라인만 주어진다. 인풋으로 받는 정보의 양이 다르기 때문에 비교 대상이라곤 할 수 없다고 생각한다. 다만, fuzzing에 에이전트 다운 에이전트를 사용한 연구라서 후일 fuzzing 에이전트 연구를 하는데 많은 도움이 될 것 같다.

4.2 Beyond the Venue

저녁엔 연구실 선배이신 차수영 교수님이 밥을 사주셨다. 숙소 앞에 있는 레스토랑에서 스테이크를 사주셨는데 아주 맛있게 먹었다. 본인 제자들과 학교 사람들 챙기시기도 바쁘실 텐데, 따로 시간을 내어 신경 써 주셔서 정말 감사했다. 저녁 먹고 숙소에 돌아와 맥주 한 잔 하며 이런저런 이야기를 하다가 잠에 들었다.

차수영 교수님이 사주신 스테이크. 감사합니다!!!

5 Way Back Home

한국으로 돌아가는 날. 점심 즈음에 시내로 나가 짐을 맡긴 뒤 도시를 좀 둘러보기로 계획을 짰다. 호텔에서 묵지 않아 딱히 짐을 맡길 곳이 없어 걱정했는데 Radical Storage라고 식당이나 미용실 같은 곳과 제휴해서 짐을 맡아주는 서비스가 있었다. 짐을 맡기고 이파네마 해변에 갔는데, 영화나 드라마에서만 보던 해변 문화를 직관했다. 주말이라 그런가 사람도 되게 많았고 비치 발리볼, 서핑, 다양한 해변 스포츠를 사람들이 열정적으로 하고 있었다. 특히 파도가 서핑하기에 너무 좋고 해변 오른쪽에서 왼쪽으로 갈수록 점점 세져서 난이도별 코스가 자연적으로 형성돼 있었다. 나도 바다에 뛰어들고 싶었지만 집엔 가야하기 때문에... 아쉬움을 남기고 돌아서서 바닷가 근처 식당으로 갔다. 야외 테이블에 앉아서 병맥주와 함께 세비체를 먹으며 여유를 즐겼다. 브라질에 있으면서 느낀건데 악명과 다르게 사람들이 전반적으로 여유롭고 무던한 것 같다. 한국에선 잘 느끼기 힘든 바이브였다. 삶에 지쳐 쉬고 싶을 때 리우 한달살기 같은 걸 하면 괜찮을 것 같다는 생각이 들었다.

브라질 해변엔 갈매기가 아니라 비둘기가 있다.

밥을 다 먹고, 바닷가 근처 호수에서 야생 카피바라가 나온다길래 걸어서 보러가다가 너무 더워서 쇼핑몰로 피신했다. 바다에 들어가고 싶은 열망이 강해져서 샤워 및 환복을 할 수 있는 장소를 열심히 찾았는데 결국 실패하고, 파르케 라지라는 공원을 둘러보고 쉬다가 공항으로 향했다. 다행히 귀국할 때는 항공편에 문제 없이 잘 돌아왔다. 정말 익사이팅하고 유익한 여행이었다. 그래도... 집으로 가는 공항철도 안에서 느낀건데, 아무래도 역시 한국이 편하다.

연어 세비체와 수제 고구마칩

파르케 라지에 있는 궁전. 저 위에 예수상도 보인다.