RC: Scaling Static Analyses at Facebook

김정현

Abstract

페이스북 (Facebook)은 사내 정적 분석 (Static Analysis) 도구인 Infer와 Zoncolan를 개발하였다. Infer는 안드로이드 (Android) 기반 페이스북 모바일 앱 (Mobile App)을, Zoncolan은 Hack 언어 기반의 페이스북 백앤드 (Back-end)를 대상으로 소스 코드 (Source Code)에서의 취약점 발견을 목표하였다. 이들이 발견한 취약점은 어떠한 목록으로 제공되지 않는 대신, 개발자가 커밋 (Commit)을 했을 때 야기되는 변경내역 대기시간 (Diff Time)에 맞추어 즉각적으로 취약점을 보고한다. 페이스북은 이러한 작업 흐름 (Workflow) 통합형 정적 분석 도구를 통하여 70%에 달하는 취약점 수정율 (Fix Rate)을 획득할 수 있었으며, 2014년부터 Infer는 100,000개가 넘는 취약점을 발견하였고, 6개월 간 Zoncolan은 버그 헌팅 (Bug Hunting)을 통한 취약점 제보 수집 및 사내 분석 조직에 의한 정적 분석에 비하여 약 1.4배에 달하는 취약점을 발견하였다.

"사람이 먼저다." 이 표현은 요즘도 하나의 밈 (Meme)으로 회자되곤 하는, 어느 대선주자의 선거 구호이다. 개발, 특히 페이스북과 같은 거대 공룡 기업에서의 개발력은 곧 '인력'이다. 제 아무리 기술적으로 훌륭한 도구나 프레임워크 (Framework)를 구비한다 한들, 규모가 수백만 라인 (Line)에 달하는 소프트웨어 (Software) 개발은 결국 잘게 분업화된 '인간의 손'을 거쳐 이루어진다. Infer (그리고 Zoncolan)가 훌륭한 정적 분석 도구가 될 수 있었던 것은, 물론 견고한 구성 추론 (Compositional Reasoning)도 한 몫을 했겠지만, 기존의 개발 과정 속 작업 흐름에 '녹아드는' 접근을 취했기 때문이다. 누구나 자신에게 추가적인 일이 주어지면, 애사심이 극도로 높은 이가 아닌 이상, 자연히 본업에 비한다면 더 적은 흥미를 갖기 마련이다. 더군다나 아직까지도 보안의 중요성을 깊이 인지하지 못하는 개발자들이 적지 않은 실정이니, 개발자에게 '추가분의 일'을 그것도 보안 관련 사유로 별도 지급하는 방식은 그 효율성이 높을 리 만무하다. 이러한 측면에서 코드 리뷰 (Code Review)를 변경내역 대기시간에 통합시키는 것은 훌륭하다. 컴퓨터 연산장치 (CPU)와 마찬가지로 사람 역시 문맥 변환 (Context Switch)에 대한 피로를 느끼기 마련이고, 또한 즉각적인 피드백 (Feedback)은 피드백 자체로서 하나의 '흥미 요소'가 될 수 있다. 사람은 자신의 행동에 대한 즉각적인 자극 (보상)에 끌리기 마련이니, 혹은 마인드셋 (Mindset)이 훌륭한 개발자라면 이러한 피드백을 자기 자신의 발전을 위한 '자양분'으로 여기며, 일종의 티칭 시스템 (Teaching System)으로 이용할지도 모를 일이다.

정탐 (True-positive)과 오탐 (False-Positive) 간의 적절한 균형점 (Balance)을 찾는 것 역시 동일한 맥락에서 이해할 수 있다. 정탐을 늘리기 위하여 마냥 과대평가를 한다면 이에 따라 오탐의 빈도 역시 증가한다. 그리고 그 증가된 오탐의 검증은 결국 사람의 손에 달려 있다. 기계적인 자동화를 통하여 오탐 분류 (Triage) 역시 어느 정도까지는 자동화할 수 있겠지만, 인간이 개입하지 않는 완벽한 자동화란 쉬운 일이 아닐 뿐더러, 인간의 손으로 이루어져야 하는 만큼 오탐으로 인한 전체 탐지 수의 증가는 마냥 반갑지 못하다. 결국 오탐 증가는 추가적인 인력 소모로 이어지며, 불필요한 인력 소모라는 관점으로 하여금 다시 본래의 균형점에 대한 의문이 제기된다. 대개 이러한 류의 균형 찾기 문제는 휴리스틱 (Heuristic)한데, 가령 취약점 발견을 예로 든다면 '어떠한' 수준에서 '얼마만큼의' 과대평가를 용인할지는 대개 수학적으로 결정적이지 못하다. 때문에 균형 찾기란 필연적으로 적절한 '선'을 찾아가는 휴리스틱 문제에 속하게 된다. 대상의 시스템을 완벽히 이해하고 현상을 관찰하는 것이 아니라, 시스템 속에서 어떠한 대상을 '발견'하는 과정 속에서 그 현상 역시 발견된다. 이로 인하여 균형의 발견은 곧 '균형을 발견하는 경험'의 발견이 된다.

나는 이러한 휴리스틱 문제, 특히 인간이 조정해내기 까다로운 소모적인 문제는 기계 학습 (Machine-Learning)을 잘 활용해야 한다는 믿음을 갖고 있다. 그리고 이러한 휴리스틱 조정의 자동화 역시, 사람의 '수고로움'을 감소시킨다는 측면에서 본다면 앞에서 이야기한 "사람이 먼저다"라는 관점과 그 맥을 함께 한다. 현재까지 기계 학습은 기나긴 발전을 거듭해 왔으며, 초기의 알파고에서부터 현재의 Github Copilot에 이르기까지 실로 다양한 영역에 적용되어 왔다. 기계 학습의 의의는, 한마디로 간단히 명명할 수 있는 성질의 것은 아니지만, '휴리스틱의 자동 생성'에 있다. 신경망 (Neural Network)이란 인간이 그동안 휴리스틱하게 고안해왔던 수많은 알고리즘을 대신할 수 있는 하나의 '대안'이며, 그 학습 과정 전체를 러프 (Rough)하게 본다면 이는 휴리스틱의 자동 생성으로 이해할 수 있다. 때문에 기계 학습을 적용한다는 것은 곧 인간이 직면해야할 수많은

휴리스틱 문제를 자동화한다는 것이며, 이는 앞에서 이야기한 '균형을 발견하는 경험'으로부터의 완전한 해방과 다름없다. 하지만 신경망은 결코 어떠한 '마법'과 같은 것이 아니다. 신경망에 대한 심도 깊은 이해가 수반되어야 이에 따른 적합한 적용이 가능하다. 기계 학습에 대한 막연한 환상, 그리고 어설픈 이해로 인한 그릇된 실망감은 기계 학습의 본질을 해칠 것이며, 더 나아가서는 인간의 휴리스틱으로부터의 해방 시점을 다소간 늦추게 될 것이다.

기술은 기술의 수준 뿐만 아니라 기술의 '이용자'까지 고양시킬 수 있어야 한다. 기술의 본질은 어디에서 오는가? 바로 기술의 '이용'이다. 기술 수준의 측면에서는 훌륭하나 이를 이용하는 측면에서 그 가치가 절하된다면, 이는 '죽은 기술'과 다름없다. 이 세상 모든 개념은 각기 자신의 존재 가치를 발하는 어떠한 '지점'이 존재한다. 가령 학문은 과학적 성취로서 그 존재 가치를 갖는다. 학문의 관점에서 그 성취의 의의가 뚜렷하다면, 그 자체로 족하다. 하지만 기술은 학문이 아니다. 구성상 훌륭함이 반드시 실제적인 효율성 증대로 연결되지 않는다. 따라서 기술, 특히 소프트웨어 개발은 이용자에게 제공할 수 있는 환경의 훌륭함도 고려해야 한다. 하지만 그렇다고 해서 기술이 꼭 이용 가치의 향상만을 바래서는 안 된다. 기술 본래의 수준이 탄탄해야 이후의 발전 가능성 (확장 가능성)도 높아지는 법이다. 개발의 수준과 개발의 이용 가치 사이 적절한 '균형'을 찾는 것, 참으로 어려워 보인다.