AFL, 무작위성도 꿰어야 보배

Tae Eun Kim

2021.10.07

Abstract

Fuzzing은 사람이 예측하지 못한 무작위 입력을 통해 프로그램을 테스트하는 기법이다. 하지만 무작위성만 갖춘 단순한 Fuzzing은 명백한 한계가 존재한다. 이를 보완하기 위해 Mutation과 Greybox의 개념을 탑재한 AFL이라는 도구가 등장하였다. AFL은 놀라운 성능을 보여주었고 수많은 후속 연구들이 이를 기반으로 다양한 Fuzzer를 제안하였다. 무작위성엔 강력한 힘이 있지만 AFL과 같이 적절히 통제하였을 때 더 유용한 도구가 된다.

말 그대로 태풍속에서 태어난 Fuzzing은 일반적이지 않은 무작위성 입력을 통해 프로그램을 테스트 하는 기법이다. 사람이 만든 테스트는 프로그램의 일반적인 실행에 상식적인 예외사항이 더해져서 만들어지지만, Fuzzzing은 그보다 훨씬 넓은 범위의 입력을 테스트로 던지게 된다. 이를 통해 개발자가 예상하지 못했던 프로그램의 결함을 더 많이 찾을 수 있다.

하지만 순전히 무작위성에 기대서는 프로그램을 깊이 관통하는 입력을 생성할 수 없다. 프로그램에 들어갈 수 있는 입력은 거의 무한하지만 실제로 프로그램이 처리하는 입력은 제한되어 있기 때문이다. 따라서 올바른 입력을 변형하여 새로운 입력을 생성하는 Mutation이라는 개념이 도입되었다. 하지만 아무리 좋은 입력을 생성해도 실행 결과만 가지고는 Fuzzing 과정이 잘 되고 있는지 확인할 방법이 없다. 이에 최소한의 프로그램 실행 정보, 즉 커버리지 달성률을 통해 Fuzzing을 보조하는 Greybox Fuzzing이라는 개념도 등장하였다.

AFL은 상술한 두 개념에 기반하여 탄생한 Fuzzing 도구이다. AFL은 효과적인 커버리지 달성을 통해수많은 버그를 발견하였으며, JPEG 파일을 입력으로 받는 프로그램의 경우, Fuzzing을 하며 6시간만에 "hello"라는 문자열에서 JPEG 파일을 생성해 내기도 하였다. 이러한 성능에 매료된 수많은 후속 연구들이 이를 기반으로 한 Fuzzer를 제안하였으며, AFL은 마치 하나의 플랫폼처럼 자리잡았다.

AFL은 무작위성과 컴퓨팅 파워가 있는 그대로도 강력한 도구지만 이를 적절히 통제한다면 비약적인 성능 향상을 이룰 수 있는 것을 보여주었다. 이처럼 주어진 자원을 효과적으로 잘 사용하는 것은 공학의 기초이고, 연구자가 항상 고민해야 될 문제일 것이다.