컴파일러 퍼저에서 머신러닝의 적용에 대한 연구

박재열[1], 박성환[2], 권동현†
[1]부산대학교 정보컴퓨터공학부 학부생
[2]부산대학교 정보융합공학과 박사과정
†부산대학교 정보컴퓨터공학부 교수
woduf0628@pusan.ac.kr, starjara@pusan.ac.kr, kwondh@pusan.ac.kr

Research on the Application of Compiler Fuzzing in Machine Learning

Jae-Yeol Park[1], Seong-Hwan Park[2], Dong-Hyeon Kwon†
[1]Dept. of Computer Science and Engineering, Pusan National University
[2]Dept. of Computer Engineering, Pusan National University
†Dept. of Computer Science, Pusan National University

요 약

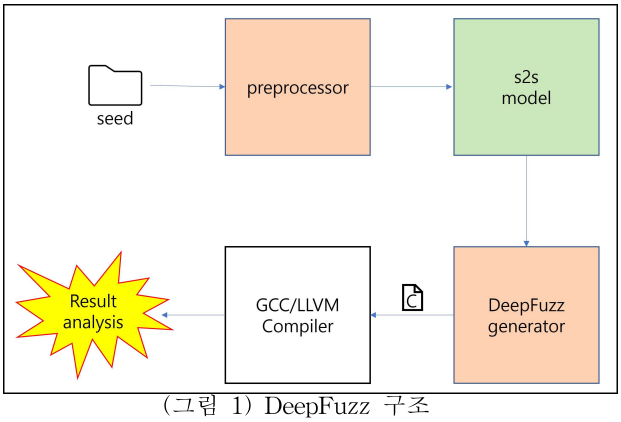
브라우저, 컴파일러 등과 같이 규모가 큰 소프트웨어에 존재하는 버그 및 취약점을 찾기 위해 퍼징은 자주 사용되는 방법 중 하나이다. 특히 컴파일러에 존재하는 버그를 찾기 위해 다양한 퍼징 방법이 연구되었으며 컴파일러의 문법 검사를 통과하여 컴파일러 내부 깊은 곳에 존재하는 버그를 찾기 위한 연구도 진행되었다. 최근 머신러닝을 활용하여 특정 언어의 문법을 학습 시킨 모델을 활용해 퍼징을 하는 연구가 활발히 진행되고 있다. 이에 본 논문에서는 컴파일러를 퍼저에 머신러닝을 적용한 연구에 대하여 정리했다.

1. 서론

브라우저, 컴파일러, 서버와 같은 큰 규모의 소프트웨어에서 효율적으로 버그를 찾기 위해 퍼징이 애용된다. 그중 컴파일러에 존재하는 버그를 찾고자 다양한 방법이 논의되었으며, 특히 입력 파일의 문법 검사를 통과 할 수 있는 방법에 대한 다양한 연구가 있었다.[1] 최근 머신러닝을 활용하여 문법을 학습한 모델을 이용해 입력 파일을 생성하고 퍼징을 수행하는 방법이 제시되었다. 이에 본 논문에서는 머신러닝을 활용한 컴파일러 퍼저를 조사하고 비교하였다.

2. DeepFuzz

sequence to seqeunce 모델을 사용하여 개발된 컴파일러 퍼저이다.[2] gcc 컴파일러를 대상으로 개발되었으며, 프로그램 생성, 컴파일러 테스트 크게 두 단계로 구성되어 있다.[2] 이중 프로그램 생성 단계에 컴파일러의 문법 검사를 통과하며 생성 전략이 적용된 코드를 만드는 퍼저의 핵심 기능이 구현되어 있다. 프로그램 생성 단계는 크게 세 단계로 나뉘어 있는데 학습에 방해가 되는 주석, 공백, 매크로를 없애거나 적절한 값으로 대체하는 전처리 과정, 원본 프로그램과 유사한 형태의 코드만 만들지 않도록 코드의 형태를 다양화 하는 샘플링 변형 과정, 퍼저의 코드 변형 전략이 적용되어 샘플링된 코드를 바꾸는 과정이 있다.

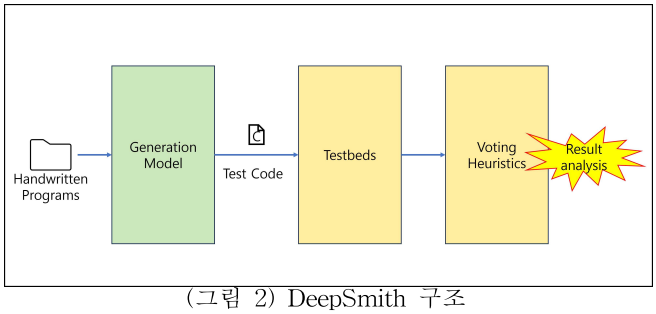


(그림 1) DeepFuzz 구조

3. DeepSmith

생성형 모델을 기반으로 OpenCL 프레임워크 상에서 컴파일러 퍼징을 위하여 개발된 퍼저이다.[3] 생성형 모델, 테스트 하네스, 투표 휴리스틱 3가지 단계로 이뤄져 있다. 10000개의 오픈소스 프로젝트를 조합하여 직접 문법에 맞지 않는 파일을 제외하며 코드 뭉치를 만들고 생성형 모델을 학습한다. 이

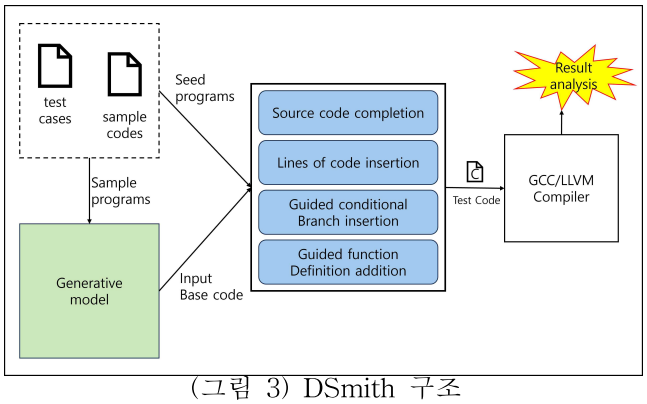
후 모델이 샘플 코드를 생성하면 코드에 맞게 입력 값을 무작위로 생성하며 컴파일러를 테스트한다. 이후 각 테스트 입력의 결과를 차등 테스트 방법론을 통해 분석하여 오류를 유발하는 입력을 찾아낸다.



(그림 2) DeepSmith 구조

4. DSmith

강력한 테스트 케이스 생성을 위해 구문의 장거리 종속성을 고려하는 sequence to sequene 모델 기반 퍼저이다.[4] 프로그램에서 각 토큰의 숨겨진 상태를 기억하고 숨겨진 상태의 상호 작용을 활용하여 토큰 간의 장거리 종속성을 내장한다. 그런 다음 장거리 종속성을 포함하는 인코더-디코더 아키텍처를 기반으로 일반 프로그램의 언어 모델을 구축한다. DSmith는 언어 모델을 사용하여 소스코드 완성, 코드 삽입, 분기 삽입, 함수 삽입 4가지 새로운 생성 전략에 따라 테스트 케이스를 생성한다. 이후 각 테스트 케이스를 직접 실행해보며 결과를 분석하여 버그 여부를 판단한다.



(그림 3) DSmith 구조

5. 비교

DeepFuzz, DeepSmith, DSmith 퍼저 모두 sequence to sequence 모델을 기반으로 한 생성형 모델을 활용하여 컴파일러에 입력으로 들어가는 코드를 생성한다는 공통점이 있으며, 생성된 코드를 어떻게 변화를 주는 지, 컴파일 결과의 분석을 어떻게 하는 지에 차이가 있었다.

모델과 별개로 입력을 수동으로 하며 퍼징을 수행한다는 점, 출력 결과를 테스터가 직접 보고 분석해서 다음 입력에 차이를 줘야 한다는 공통점 또한 존재했다.

6. 결론

머신러닝을 활용하여 컴파일러의 문법 검사를 통과하는 입력을 생성하는 퍼저는 입력에 대한 결과를 수동으로 분석하고 퍼징 수행 시 입력이 자동화되지 않는다는 특징이 있다. 이는 짧은 시간에 보다 많은 테스트를 수행하여 버그를 효율적으로 탐지해내는 것이 주요 목표인 퍼저에서 큰 단점으로 볼 수 있다. 그러므로 입력 및 결과 분석, 피드백, 입력 변형을 자동화하는 새로운 프레임워크의 개발이 필요하다.

"본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학ICT연구센터사업의 연 구결과로 수행되었음" (IITP-2023-RS-2023-00259967)

참고문헌

[1] Ma, Haoyang. "A Survey of Modern Compiler Fuzzing." arXiv preprint arXiv:2306.06884 (2023).

[2] Liu, Xiao, et al. "Deepfuzz: Automatic generation of syntax valid c programs for fuzz testing." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 33. No. 01. 2019.

[3] Cummins, Chris, et al. "Compiler fuzzing through deep learning." Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis. 2018.

[4] Xu, Haoran, et al. "Dsmith: Compiler fuzzing through generative deep learning model with attention." 2020 International Joint Conference on Neural Networks (IJCNN). IEEE, 2020.