인공지능 기반 퍼징 기술 연구 동향

김세중 [1], 오현영 [2*]
[1] 가천대학교 AI·소프트웨어학과 석사과정
[2*]가천대학교 AI·소프트웨어학과 교수

humming1106@gachon.ac.kr, hyoh@ gachon.ac.kr

Research Trends in AI-based Fuzzing Technic

Se-Jung Kim, Hyun-Young Oh*
Dept. of AI·Software, Gachon University

요 약

IT 기술의 발전으로 소프트웨어의 고도화가 진행되면서 잠재적인 취약점이나 버그를 찾아 내는 테스트가 중요한 부분을 차지하고 있다. 그러나 점점 복잡해진 소프트웨어는 전통적인 퍼징으로 취약점을 찾기 위한 깊은 탐색을 하지 못하는 어려움을 겪고 있다. 해당 문제를 해결하기 위해서 퍼징 테스트에 인공지능을 접목시키는 연구가 활발히 이루어 지고 있다. 본 논문에서는 인공지능 기반 퍼징 기법들을 분석하고, 인공지능 기반 퍼징의 한계점과 향후 연구 방향성을 살펴본다.

1. 서론

최근 소프트웨어의 신뢰성과 보안을 확보하기 위해 퍼징(fuzzing)[1] 기술이 널리 활용되고 있다. 퍼징은 프로그램에 대량의 입력을 자동으로 투입해 예기치 않은 동작이나 취약점을 발견하는 기법으로, 수많은 실제 소프트웨어 버그와 보안 취약점을 찾아내며 효과를 입증해왔다. 특히 coverage-guided 퍼징[2]은 실행 경로 커버리지를 높이는 방향으로 입력을 변이(mutation)시켜가며 높은 효율을 보여왔다. 하지만 소프트웨어의 복잡성이 지속적으로 증가함에 따라, 전통적인 퍼저(fuzzer)들은 무작위 변이와 휴리스틱에 의존하기 때문에 유효한 테스트 케이스를 만들어내기 어려운 한계를 지닌다. 또한 테스트 케이스의 품질은 프로그래머의 경험과 지식에 의해 결정된다[3]. 이와 같은 상황에서, 이미지 인식, 자연어 처리 등 다양한 분야에서 인공지능(AI)기술이 혁신적인 성과를 보임에 따라 소프트웨어 테스트 분야에도 인공지능을 퍼징에 접목하려는 연구가 활발하게 이루어지고 있다[4-6]. 인공지능 기반 퍼징은 머신러닝을 활용하는 모델을 시작으로 점차적으로 딥러닝 모델로 발전해 왔으며, 최근 들어서 대규모 언어모델(LLM)을 사용하는 연구가 활발히 이루어 지고 있다. 따라서 본 논문에서는 이러한 연구 사례들에 대해 알아본다.

2. 배경지식

2.1 퍼징

퍼징은 프로그램에 무작위나 유도된 입력 값들을 반복적으로 제공하여 오류를 유발함으로써 잠재적인 버그를 찾아내는 동적 테스트 기법이다. 기본적으로 퍼저는 프로그램의 충돌이나 예외가 발생하면 이를 잠재적 취약점으로 기록하며, 이러한 과정을 자동 반복함으로써 사람이 찾아내기 어려운 코너케이스까지 광범위하게 탐색한다. 블랙박스 퍼징은 프로그램 내부를 고려하지 않고 입력만 바꿔가며 테스트하며, 그레이박스 퍼징은 실행 중 얻는 커버리지 정보를 피드백으로 활용하여 새로운 입력을 생성한다. 대표적인 그레이박스 퍼저인 AFL[7]은 신뢰성이 입증되어 많은 연구들이 등장했으며 효율적인 변이 전략으로 다양한 소프트웨어에서 높은 커버리지를 달성해왔다. 그러나 퍼징 대상 입력의 형식이 매우 복잡하거나, 입력 값들 간에 제약 관계가 있는 경우, 랜덤 변이나 단순 휴리스틱으로는 유효한 입력을 얻기 어려워 깊은 실행 경로를 탐색하지 못하는 한계가 있다. 이에 따라 입력 형식을 이해하거나 보다 지능적으로 입력을 변이할 수 있는 기법에 대한 필요성이 대두되었다.

2.2 인공지능

인공지능은 기계가 인간처럼 학습, 추론, 문제 해

* 교신저자

결 등의 인지적 기능을 수행할 수 있도록 하는 기술이다. 인공지능은 크게 머신러닝, 딥러닝, 대규모 언어모델 기반의 생성 AI 가 있다. 머신러닝과 딥러닝 알고리즘은 대량의 데이터를 학습하여 복잡한 패턴을 모델링할 수 있으므로, 퍼징에 활용하면 입력 공간에 대한 확률 모델을 획득하거나 취약점이 존재할 수 있는 패턴을 학습할 수 있다. 예를 들어 RNN[8]이나 GAN[9] 같은 생성 모델은 실제 프로그램이 처리하는 입력들의 분포를 학습한 뒤, 이를 기반으로 변형된 새로운 테스트 입력들을 생성할 수 있다. 이는 단순 무작위보다 의미 있는 실행을 이끌어 낼 확률을 높여준다.

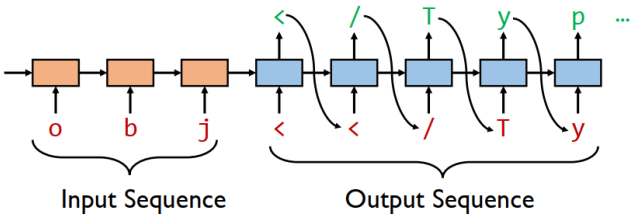
또한 2020 년대 들어 등장한 GPT-3[10] 같은 대규모 언어 모델(LLM)[11]은 코드와 자연어를 다룰 수 있을 만큼 방대한 지식을 내포하고 있다. LLM 은 사람이 작성한 것 같은 논리적이고 복잡한 입력이나 코드 조각을 생성 할 수 있어 복잡한 API 호출 시나리오를 생성하는 데 응용되고 있다.[12] 예를 들어, LLM 모델에 특정 API 에 대한 간단한 프롬프트를 주면 그 API 를 사용하는 작은 프로그램 코드를 생성해준다. 이러한 방식으로 사람의 노력 없이 다양한 조합의 API 호출을 테스트할 수 있다. 요약하면 인공지능 모델을 활용하여 복잡한 입력 구조나 사용 패턴을 학습하여 효율적으로 탐색하거나 과거 데이터로부터 더 나은 입력 값을 만드는 것이 핵심이다.

3. 연구사례

인공지능 기반 퍼징에 관한 학술 연구는 몇 가지 유형으로 나누어 볼 수 있다. (1) 딥러닝 기반 생성 모델을 활용하여 높은 커버리지를 달성할 수 있는 테스트 입력을 만들어 직접 만들어내는 접근, (2) 학습 모델로부터 생성한 입력 값을 기존 AFL 의 시드 값으로 제공하여 효과적인 변이를 달성하는 접근 (3) 대규모 언어 모델을 통해 고차원적인 입력(예: 코드 또는 시나리오) 생성에 도전하는 접근 등이 대표적이다. 아래에서는 이러한 방법론을 대표하는 주요 연구 사례들을 소개한다.

3.1 Learn&Fuzz[4]

Learn&Fuzz 는 인공지능과 퍼징을 결합한 초창기의 선구적인 연구로서, 신경망 기반 언어 모델을 이용해 입력 포맷의 문법을 학습하고 이를 퍼징에 활용한 연구이다. (그림 1)에서 대상 프로그램이 처리하는 입력 샘플들을 다수 수집한 후, 문자 단위 RNN(char-RNN) 모델을 학습시켜 입력의 확률 분포와 문법을 생성하는 과정을 보여준다. 이렇게 학습된 입력 생성 모델을 통해 현실적인 입력을 생성하되, 퍼징 목적에



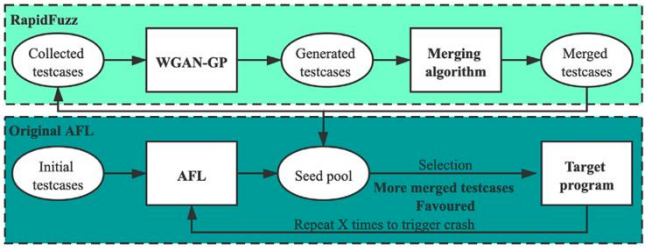
(그림 1) char-RNN 에서의 PDF objects 생성[4]

맞게 일부 위치를 확률적으로 변이 시켜 비정상적인 입력을 만들어낸다. PDF 파서를 퍼징하는 경우, 정상 PDF 파일들의 패턴을 RNN 이 학습하지만, 테스트 생성 시에는 패턴 일부를 변이 시켜 문법에 맞지만 살짝 어긋난 입력들을 만들어 내도록 한다. 연구에서는 PDF 파일 포맷과 Edge 브라우저의 PDF 파서를 사례로 실험하여, 학습된 모델이 잘 형성된 입력 구조를 따르면서도 무작위 변이를 통해 새로운 경로를 탐색하도록 조율할 수 있음을 보여주었다.

하지만 인공지능의 학습은 잘 형성된 데이터 구조를 따르는 경향을 지적하면서, 학습된 확률분포를 기반으로 무작위 변이 지점을 선택하는 SampleFuzz 알고리즘을 제안하였다. 이 알고리즘은 RNN 모델이 예측한 값에 high-confidence 를 보일 경우 일부러 낮은 확률의 문자를 삽입하여 퍼징의 효과를 높였다. Learn&Fuzz 는 인공지능을 통해 입력 언어의 이해도를 높여 퍼징 효율을 높인 초기 사례로, 이후 다양한 학습 기반의 퍼징 연구의 토대가 되었다.

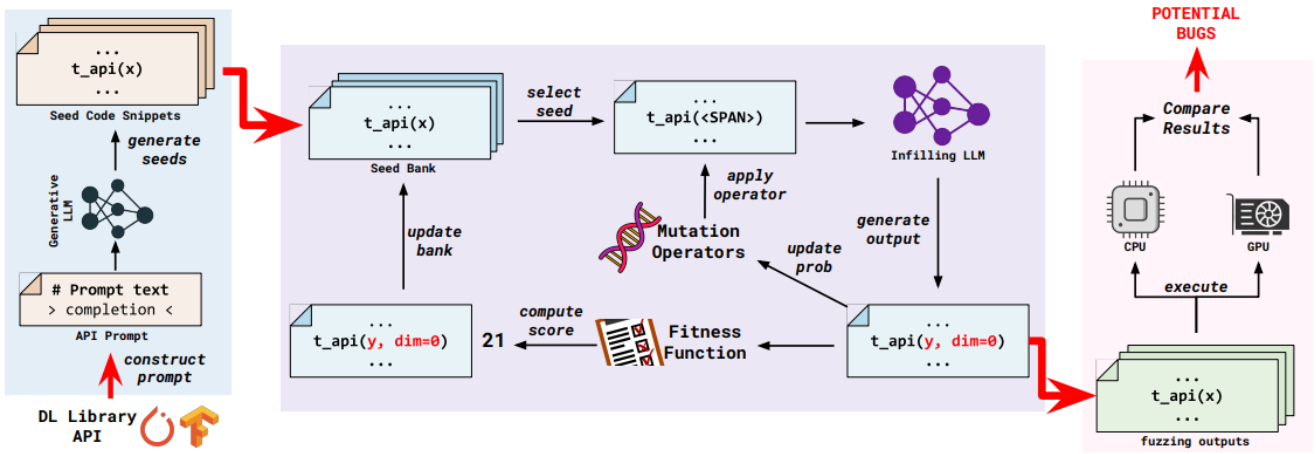
3.2 RapidFuzz[5]

RapidFuzz 는 딥러닝 생성 모델 중 생성적 적대 신경망(GAN)을 퍼징에 활용하여 퍼징 속도와 커버리지를 크게 향상시킨 연구이다. (그림 2)에서 볼 수 있듯이 GAN 의 Generator 를 훈련시켜 대상 프로그램의 입력의 데이터 구조 특징을 추출하게 한다. 이를 통해 실제 입력들과 유사하지만 랜덤성과 다양성을 추가하여 새로운 입력들을 생성할 수 있다. RapidFuzz 는 GAN 으로 만들어낸 입력들을 퍼징의 시드(seed)로 활용하여, 기존 커버리지 기반 퍼저(AFL)에 공급함으로써 퍼징을 가속한다.



(그림 2) RapidFuzz 개요[5]

특히 GAN 이 출력한 입력들 중 핫-포인트라 부르는 중요 변이 지점을 식별하는 알고리즘을 제안하였다. 이는 GAN 이 생성한 시드들에서 반복적으로 나타나는



(그림 3) TitanFuzz 모델 개요[6]

변이 패턴을 분석하여 효과적인 변이 위치를 찾는 기법이다. 실험 결과 RapidFuzz는 TIFF 이미지 처리 프로그램(예: tiff2pdf, tiffdump) 등 구조가 복잡한 입력 포맷을 다루는 9개의 프로그램을 대상으로 순수 AFL에 비해 퍼징 성능(코드 커버리지, 탐지한 경로 수 등등)을 크게 향상시켰다. 특히 두 개 사례 프로그램에서 코드 커버리지가 20% 이상 증가했으며, 동일 커버리지를 달성하는 데 걸리는 시간도 크게 단축됨을 보여준다. 또한 AFL이 RapidFuzz가 생성한 시드 파일 중 약 15%~21%를 새로운 경로 발견에 유용한 테스트 케이스를 채택하여 내부 큐에 추가한 것도 관찰되었다. 이는 GAN이 생성한 입력 중 상당수가 기존 퍼저에 유용한 시드로 사용되었음을 의미한다. RapidFuzz는 딥러닝 기반 생성 모델을 활용하여 전통적인 퍼징을 보조함으로써, 적은 시간에 더 많은 커버리지를 탐색할 수 있음을 입증하였다.

3.3 TitanFuzz[6]

TitanFuzz는 최근 주목받는 대규모 언어모델(LLM)을 퍼징에 도입한 연구로, 딥러닝 라이브러리(예: TensorFlow, PyTorch)의 API를 집중적으로 퍼징하기 위해 고안된 기법이다. 논문의 부제인 “Large Language Models are Zero-Shot Fuzzers”에서 알 수 있듯이 추가 학습 없이 거대 언어모델 자체를 제로-샷(Zero-Shot) 퍼저로 활용하는 접근을 보여준다. TitanFuzz의 아이디어는 GPT 계열의 코드 생성 모델들이 방대한 오픈소스 코드로 학습되어 있기 때문에, 여기에는 각종 DL 라이브러리 API를 호출하는 코드 조각들이 다수 포함되어 있을 것이라는 점에서 착안한다. 즉, LLM은 파이썬 문법과 DL API 사용 제약까지 내재적으로 학습하고 있으므로, 이를 활용하면 사람이 일일이 작성하지 않아도 복잡한 API 호출 시나리오를 생성할 수 있음을 보여주고 있다.(그림 3)은 TitanFuzz의 전체적인 개요를 보여준다. TitanFuzz는 생성형 LLM인 Codex를 이용하여 무작위로 다양한 시드 프로그램을 만들어낸다. 생성된 시드는 fitness score를 측정하여, 높은 점수를 가진 시드 프로그램을 선택한다. 선택된 시드는 변이 연산자를 사용하여 특정 코드 부분을 <SPAN> 토큰으로 마스킹한다. 마스킹된 부분을 바탕으로 코드 인필링(infilling) LLM을 활용하여 마스킹된 토큰에 새로운 코드를 작성한다. 이후 반복을 통해 높은 점수를 받은 시드를 우선적으로 퍼징테스트의 입력으로 사용하여 잠재적인 버그를 식별하게 된다. 이렇게 생성된 수많은 입력 프로그램을 DL 라이브러리에 적용하여 퍼징을 수행한 결과, TitanFuzz는 기존 퍼저들 대비 TensorFlow/PyTorch에서 30.38%/50.84% 더 높은 코드 커버리지를 달성하였고, 총 65개의 버그를 발견하였고 이 중 41개의 신규 버그로 확인되는 성과를 거두었다. 이는 전통적인 퍼저로는 다루기 힘든 고차원 입력공간에서도 LLM을 활용하면 효과적인 탐색이 가능함을 보여준다.

4. 비교 분석

세 가지 인공지능 기반 퍼징 기법은 각각 다른 인공지능 모델을 활용하여 입력 데이터를 생성하거나 변이하는 전략을 취하고 있다. 이를 통해 보다 구조화된 입력과 정밀한 변이 전략을 가능하게 하였다. <표 1>은 Learn&Fuzz, RapidFuzz, TitanFuzz 세 기법의 대표적인 특성을 보여준다.

	Learn&Fuzz	RapidFuzz	TitanFuzz
사용 모델	char-RNN	GAN	LLM(Codex, InCoer)
퍼징 대상	Edge PDF 파서	tiff2pdf, tiffdump 등	PyTorch/ TensorFlow
퍼징 입력	PDF 파일	이미지 처리	DL 코드
커버리지 향상	+0.6%	+20%	+50.84%/+30.38%

<표 1> 인공지능 기반 퍼징의 비교분석

전통적인 퍼징은 단순 무작위한 입력 값을 변형하는 반면, 세 가지의 인공지능 기반 퍼징은 문법, 코드의 패턴에 대한 학습을 통해 의미 있는 변이 지점을 만들어내고 있다는 점에서 맥을 같이한다. 세 가

지의 퍼저 모두 결과적으로 기존 퍼저 대비 코드 커버리지가 향상되었음을 보고 하고 있다.

그럼에도 불구하고 각각의 퍼저들은 아키텍처 적용 분야의 차이로 인해 여러 구별점이 존재한다. Learn&Fuzz는 학습을 기반으로 생성한 입력 값을 SampleFuzz 알고리즘을 사용하여 입력 값을 새로 생성하는 생성 기반 퍼징을 제시하였고, RapidFuzz는 GAN으로 생성한 입력 값을 기존 AFL에 시드파일로 공급함으로써 커버리지 효율을 높이는 방법을 제시하였다. 이 두 기법은 입력 형태가 바이트 시퀀스인 파일 데이터를 다루는 반면, TitanFuzz는 다른 퍼저들과 다르게 2개의 LLM모델을 사용하여 프로그래밍 언어 코드 기반을 퍼징 입력으로 사용한다. 또한 TitanFuzz를 제외한 두 기법은 학습 데이터가 필요하지만 TitanFuzz는 사전 학습 모델을 활용하여 추가 학습 없이 동작한다는 차이도 있다. 성능적인 측면에서 Learn&Fuzz는 전통적인 퍼징보다 0.6%의 커버리지 향상을 보여주고 RapidFuzz와 TitanFuzz에서는 각각 20%와 30%이상의 커버리지 향상을 보여준다. Learn&Fuzz은 대상 프로그램에 안정적인 퍼징 입력 값을 실행 할 수 있는 안정성을 중점으로 개선하였기 때문에 낮은 성능폭을 보여준다. RapidFuzz는 커버리지와 효율 측면에서 확실한 향상을 입증하였지만 발견한 신규 취약점이 없었다. TitanFuzz는 가장 두드러진 개선을 보였고 실제 다수의 신규 취약점을 찾아냈다.

5. 결론

인공지능 기술의 발전에 힘입어 퍼징은 과거와는 차원이 다른 형태로 진화하고 있다. 본 논문에서는 머신러닝과 딥러닝을 통해 입력 구조를 학습하거나 새로운 테스트케이스를 생성하는 접근, 대규모 언어 모델을 활용하여 사람처럼 고차원적인 시나리오를 자동 작성하는 접근까지 여러 방법들을 사용하여 저마다의 장점을 통해 퍼징의 효율과 효과를 높이고 있다. 요약하면, 전통적인 퍼징의 난제였던 “복잡한 입력 공간의 탐색” 문제를 인공지능으로 풀어내는 방향으로 연구 흐름이 진행되고 있으며, 이미 여러 도메인(파일 포맷, 딥러닝 프레임워크 등)에서 유의미한 성과가 보고되고 있다.

물론 인공지능 기반 퍼징에도 몇 가지 도전과제가 남아 있다. 첫째, 학습 데이터나 모델 편향으로 인해 퍼징 탐색이 한쪽으로 치우칠 위험이 있으며, 신규 취약점 탐지를 위해서는 여전히 무작위성과 다양한 퍼징이 중요한다. 둘째, 사전 학습이 필요 없는 대규모 모델을 활용하더라도 비효율적인 경로 탐색을 줄이기 위한 프롬프트 엔지니어링 같은 정교한 기법이 필요하며, 학습과 퍼징 간 시너지를 최대화하는 하이브리드 기법에 대한 연구도 앞으로 필요할 것이다. 따라서 인공지능과 퍼징 간의 트레이드 오프 관계를 고려해야 한다. 그럼에도 불구하고, 퍼징과 인공지능의 결합은 이미 입증된 여러 성공 사례들처럼 향후 소프트웨어 테스팅 분야의 새로운 지평을 열고 있다. 특히 수 많은 생성형 인공지능의 폭발적인 발전은 퍼징에도 큰 기회를 제공하고 있어, 앞으로 인공지능을 통한 퍼징 자동화가 더욱 흥미로운 방향으로 전개될 것으로 기대된다. 지속적인 연구를 통해, 인간 전문가의 개입을 최소화하면서도 더 많은 버그를 찾아내는 인공지능 기반 퍼징 시스템이 현실화되고 있다.

사사문구

이 논문은 2025 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구 결과 임 (No. RS-2024-00337414, SW 공급망 운영 환경에서 역공학 한계를 넘어서는 자동화된 마이크로 보안 패치 기술 개발).

참고문헌

[1] Zhu, Xiaogang, et al. "Fuzzing: a survey for roadmap." ACM Computing Surveys (CSUR) 54.11s (2022): 1-36.
[2] Böhme, et al. "Coverage-based greybox fuzzing as markov chain." Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. 2016.
[3] Zeng, Peng, et al. "Software vulnerability analysis and discovery using deep learning techniques: A survey." IEEE Access 8 (2020): 197158-197172.
[4] Godefroid, Patrice, et al. "Learn&fuzz: Machine learning for input fuzzing." 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2017.
[5] Ye, Aoshuang, et al. "RapidFuzz: Accelerating fuzzing via generative adversarial networks." Neurocomputing 460 (2021): 195-204.
[6] Deng, Yinlin, et al. "Large language models are zero-shot fuzzers: Fuzzing deep-learning libraries via large language models." Proceedings of the 32nd ACM SIGSOFT international symposium on software testing and analysis. 2023.
[7] M. Zalewski, “American fuzzy lop,” 2017. [Online]. Retrieved April 9, 2025 http://lcamtuf.coredump.cx/afl/
[8] Grossberg, Stephen. "Recurrent neural networks." Scholarpedia 8.2 (2013): 1888.
[9] Goodfellow, Ian, et al. "Generative adversarial networks." Communications of the ACM 63.11 (2020): 139-144.
[10] Brown, et al. "Language models are few-shot learners." Advances in neural information processing systems 33 (2020): 1877-1901.
[11] Zhao, Wayne Xin, et al. "A survey of large language models." arXiv preprint arXiv:2303.18223 1.2 (2023).
[12] Huang, Linghan, et al. "Large language models based fuzzing techniques: A survey." arXiv preprint arXiv:2402.00350 (2024).