최근 퍼징 기법들과 발전에 관한 연구

전소희*, 이영한*, 김현준*, 백윤흥*
*서울대학교 전기 · 정보공학부, 반도체공동연구소

shjun@sor.snu.ac.kr, yhlee@sor.snu.ac.kr, hjkim@sor.snu.ac.kr, ypaek@snu.ac.kr

A Study of fuzzing techniques and their development

So-Hee Jun*, Young-Han Lee*, Hyun-Jun Kim*, and Yun-Heung Paek*
*Dept. of Electrical and Computer Engineering and Inter-University Semiconductor Research Center (ISRC),

Seoul National University

요 약

최근 컴퓨터 프로그램의 크기가 증가하고 목적이 다양해지면서 프로그램의 취약점에 대한 위험이 증가하고 있다. 공격자 보다 먼저 프로그램 취약점을 찾아내기 위한 여러 기법들이 있다. 그 중 프로그램의 취약점을 보다 효율적으로 찾아내기 위한 기법 중 하나인 퍼징 (Fuzzing) 은 프로그램에 무작위로 입력 데이터를 입력하여 프로그램의 정의되지 않은 영역을 검증하는 기법이다. 이러한 입력 데이터를 최대한 적은 시간과 자원을 소모하여 생성하기 위해 인공지능과 퍼징을 결합하는 연구가 활발히 진행 중이다. 본 논문에서는 퍼징의 개념 및 종류에 대해 설명하고 퍼징과 인공지능이 결합된 최신 연구에 대해 서술한다.

1. 서론

최근 다양한 목적을 가진 여러 종류의 프로그램들이 요구되고 개발되면서, 프로그램의 크기 및 복잡도가 매우 증가하고 있다. 그에 따라, 프로그램의 취약점 또한 증가하게 되며 이런 프로그램의 취약점은 공격자에게 공격 대상이 될 수 있기 때문에 반드시 공격자보다 먼저 알아내어 보완되어야 한다. 마이크로소프트, 페이스북 등과 같은 IT 대기업 회사에서는 자사 프로그램의 취약점을 찾아낸 사람에게 포상금을 지급하는 버그바운티 (Bug bounty) 제도가 있을 정도로 프로그램의 취약점을 찾는 것은 중요한 작업이다. 하지만 개발자가 프로그램을 개발하면서 직접 취약점을 찾아내는 것은 사실상 불가능에 가까우며, 전문가를 통한 역공학 (Reverse engineering) 기법, 동적 테스팅과 같은 기법들이 사용되고 있지만 많은 인력, 시간과 자원이 소모된다. 이를 보완하기 위해 취약점 자동화 테스트 기법들이 활발히 연구되고 있으며 대표적으로 퍼징 (Fuzzing)기법이 있다. 본 논문은 이러한 퍼징 기법에 대해 알아보며 최근 기존 퍼징 기법의 단점을 보완하기위해 퍼징과 인공지능을 결합하는 연구에 대해 서술한다.

2. 퍼징이란?

퍼징 (Fuzzing)이란, 소프트웨어 테스트 기법으로 프로그램에 대해 무수한 여러 데이터를 입력하여 프로그램의 충돌이 발생하는 프로그램 취약점의 위치를 찾아내는 기법이며 주로 소프트웨어나 컴퓨터 시스템들의 보안 취약점을 파악하고 정의되지 않은 영역을 검증하기 위해 사용된다. 이러한 퍼징 기법의 기원은 매우 우연적이다. 장마철 전자기 간섭으로 인해 프로그램에 임의의 값들이 무작위로 입력되어 프로그램의 충돌이 발생하였고 이를 통해 무작위로 생성된 입력이 프로그램의 취약점을 발현 시킬 수 있다는 것이 발견되었다 [1]. 퍼징은 주어진 시간 동안 최대한 많은 프로그램의 취약점을 찾아내는 입력 데이터를 찾아내는 최적화 문제이지만 취약점이 프로그램 내에 드문드문하게 위치하기 때문에 퍼징의 성능은 주로 프로그램의 코드를 커버한 정도로 평가된다.

2.1 퍼징 기법의 분류

퍼징 기법은 프로그램의 정보를 활용하는 정도에 따라 블랙박스 (black-box) 퍼징, 화이트박스 (white-box) 퍼징, 그레이박스 (gray-box) 퍼징으로 분류할 수 있다. 블랙박스 퍼징은 대상 프로그램의 내부 정보를 사용하지 않고 대상 프로그램의 입력과 출력 데이터

만 사용하는 퍼징 기법이다. 대표적으로 SPIKE [2], BFF (Basic Fuzzing Framework) [3], FOE (Failure Observation Engine) [4] 등이 있다. 화이트박스 퍼징은 대상 프로그램의 내부 구조와 실행 중 발생하는 정보들을 사용하는 퍼징 기법이다. 대표적으로, 그레이박스 퍼징은 블랙박스와 화이트박스의 중간적인 특성을 가지며 대상 프로그램의 내부 정보와 실행 중 발생하는 정보 일부를 사용하는 퍼징 기법이다. 대표적으로, AFL (America Fuzzy Loop) [5], VUzzer [6] 등이 있다.

또한, 퍼징 기법은 입력 데이터를 생성하는 법에 따라 두가지로 구분할 수 있다. 생성 기반 (Generation-based) 퍼징은 데이터의 구조 및 프로토콜을 이해하여 프로그램에 적합한 입력 데이터를 생성하는 기법이다. 생성될 입력 데이터의 구조 및 프로토콜을 이해하기 때문에 유효한 입력 데이터를 잘 구성할 수 있지만 많은 시간이 소요 될 수 있다. 변이 기반 (Mutation-based) 퍼징은 입력 데이터를 특정하여 그 입력 데이터에 대해 조금씩 변이를 주어 새로운 입력 데이터를 생성하는 기법으로 여기서 특정된 입력 데이터는 주로 씨드 (seed) 데이터라 통칭된다. 변이 기반 기법은 씨드 데이터에 무작위적으로 변형을 주는 기법이기 때문에 많은 시간이 소요되지 않지만 유효하지 않은 입력 데이터가 생성되는 경우가 많아 멍청한 (dumb) 퍼징이라 불리기도 한다.

3. 인공지능을 활용한 퍼징

퍼징 기법의 시초는 데이터를 무작위로 생성하는 것이었지만, 임의의 값을 무작위로 생성하는 것은 많은 시간과 비용을 소모하며 유효하지 않은 입력 데이터를 만들 가능성이 매우 높아 비효율적이다. 그렇기에 다수의 퍼징 기법은 진화 알고리즘 (Evolutionary algorithm)을 사용한다. 진화 알고리즘은 세대에 걸쳐 입력 데이터를 생성하는 방법으로 이전 세대에서 유용하게 사용되었던 입력을 골라 다음 세대에서 재사용하여 무작위 생성 방법보다 효율성을 높일 수 있고 시간도 절약할 수 있다.

최근에는 퍼징 기법에 인공지능을 결합하여 보다 효율적으로 프로그램 취약점을 검증하는 연구가 활발히 진행되고 있다. 기존 진화 알고리즘 (Evolutionary Algorithm)을 사용하는 방법은 시간적 이점을 가지지만 무작위적으로 변형을 진행하면서 유효하지 않은 입력 데이터를 다수 생성할 가능성이 높다는 단점을 가진다. 이러한 단점을 보완하기위해, 데이터의 패턴을 학습할 수 있는 인공지능 기술과 퍼징을 결합하는 연구가 많이 진행되고 있다. 그 중 인공 신경망을 통해 입력 데이터와 출력 데이터 간의 관계를 파악하는 Neuzz [7]와 퍼징에 효과적인 입력 데이터를 생성하기 위해 인공 신경망을 사용하는 Learn & Fuzz [8]과 강화학습 (Reinforcement learning)을 퍼징에 활용하는 Deep Reinforcement Fuzzing [9]에 대해 서술한다.

3.1 Neuzz

Neuuz 는 프로그램의 엣지 커버리지 (Edge coverage) 데이터를 사용하는 그레이박스 퍼징 기법으로 신경망 (Neural Network)을 통해 출력 데이터 간의 관계를 이해하여 입력 데이터가 출력 데이터에 끼치는 영향력을 분석하여 입력 데이터에서 출력 데이터에 높은 영향력을 끼치는 부분을 변이하여 보다 효과적으로 프로그램의 취약점을 찾는다. 신경망은 대상 프로그램의 브랜치 행동 (Branch behavior)과 관련된 입력 데이터간의 관계를 비선형 함수로 근사화하여 입력 데이터에 따른 대상 프로그램의 컨트롤 플로우 엣지 (Control flow edge)를 예측한다. 학습된 신경망을 활용하여 입력 데이터에서 되면 출력 데이터에 큰 변화를 줄 수 있는 부분을 변이하여 취약점 탐지를 위한 입력 데이터를 생성한다. 또한 생성된 새로운 입력 데이터로 다시 신경망을 학습시켜 신경망이 기존 입력 데이터와 생성된 입력 데이터에 대한 학습을 증진시켜 높은 성능을 보였다.

3.2 Learn & Fuzz

Learn & Fuzz 는 신경망과 샘플 입력 데이터를 사용해 문법 기반 (Grammar-based) 퍼징을 위한 입력 데이터를 생성하는 기법이다. Learn & Fuzz 의 목표는 대상 프로그램의 코드 커버리지를 최대화하기 위한 입력 데이터를 생성하는 것과 대상 프로그램의 정의되지 않은 영역을 검증하기 위한 입력 데이터를 생성하는 것이다. 이를 위해 Learn & Fuzz 는 새로운 입력 데이터를 생성하기 위해 다음에 올 값을 예측할 수 있는 신경망의 한 종류인 RNN (Recurrent Neural Network)을 사용하며 대상 프로그램과 입력 데이터는 PDF (Portable Documents Format)가 대상이다. Learn & Fuzz 는 세가지 방법을 통해 새로운 PDF 데이터를 생성한다. 첫번째 방법은 뒤에 이어질 가장 높은 확률의 문자를 선택하는 것이고, 두번째 방법은 뒤에 이어질 문자를 확률적으로 선택하는 것이고, 세번째 방법은 위의 두 방법을 합친 것으로 앞의 문자가 공백으로 끝날 때는 두번째 방법을 사용하고 아닐 경우에는 첫번째 방법을 사용하는 것이다. 이러한 Learn & Fuzz 는 처음으로 신경망 기반 확률적 학습 기법을 사용하여 문법 기반의 퍼징을 위한 프로그램 입력 데이터를 생성하였으며 학습된 신경망 모델을 통해 유효하며 프로그램의 커버리지를 높일 수 있는 입력 데이터를 생성하였다.

3.2 Deep Reinforcement Fuzzing

Deep Reinforcement Fuzzing 은 강화 학습을 퍼징에 사용한 방법으로 강화 학습은 환경 (Environment), 행위 (Actions), 보상 (Reward)를 기본 요소로 가지며, 환경과 소통하며 행위를 하여 보상을 얻는 에이전트 (Agent)가 존재한다. 에이전트의 목적은 최대한 많은 보상을 얻는 것이다. 이러한 에이전트가 입력 데이터에 대해 행위를 수행하여 새로운 입력 데이터를 생성하고 생성된 데이터가 프로그램의 취약점을 많이 찾아내거나 프로그램의 코드 커버리지를 증가시킬수록 높은 보상을 받는 과정을 따른다. 이 과정을 통해 에이전트는 최대한 많은 보상을 얻기 위해 유효하고 퍼징 성능이 좋은 입력 데이터를 생성하게 된다. 강화 학습은 인공지능 분야 중에서도 최근 활발하게 연구되고 있는 분야로 앞으로 더욱 발전하여 퍼징과 결합된다면 높은 성능을 보일 수 있을 것으로 기대된다.

4. 결론

우리의 삶 가까운 곳에서 함께하고 있는 컴퓨터 프로그램의 목적과 기능이 다양해지면서 그에 따른 보안의 필요성 또한 크게 증가하고 있다. 이로 인해, 프로그램의 취약점을 검증하기 위한 기법들에 대한 연구가 진행 중이며 특히 자동화 소프트웨어 테스팅 기법인 퍼징에 대한 연구가 활발히 진행되고 있다. 최근에는 전통적인 퍼징 기법의 한계점을 인공지능 기술을 통해 보완하는 기법이 연구되고 있으며 높은 효율성과 좋은 성능을 보이고 있어 큰 발전이 기대되고 있다.

5. ACKNOWLEDGEMENT

이 논문은 2020 년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원 (NRF-2017R1A2A1A17069478), 2020 년도 두뇌한국 21 플러스 사업에 의하여 지원되었고 2020 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2018-0-00230, (IoT 총괄/1 세부) IoT 디바이스 자율 신뢰보장 기술 및 글로벌 표준 기반 IoT 통합보안 오픈 플랫폼 기술개발 [TrusThingz 프로젝트])

참고문헌

[1] Barton P. Miller, Louis Fredriksen, Bryan So, “An Empirical Study of the Reliability of UNIX Utilities.” Communications of the ACM, 33(12):33–44, December 1990.

[2] D. Aitel, “An introduction to SPIKE, the fuzzer creation kit,” in Proceedings of the Black Hat USA, 2001.

[3] CERT, “Basic Fuzzing Framework,” https://www.cert.org/vulnerability-analysis/tools/bff.cfm.

[4] “Failure Observation Engine,” https://www.cert.org/vulnerability-analysis/tools/foe.cfm

[5] M. Zalewski, “American Fuzzy Lop,” http://lcamtuf.coredump.cx/afl/.

[6] S. Rawat, V. Jain, A. Kumar, L. Cojocar, C. Giuffrida, and H. Bos, “VUzzer: Application-aware evolutionary fuzzing,” in Proceedings of the Network and Distributed System Security Symposium, 2017.

[7] Dongdong She, Kexin Pei, Dave Epstein, Junfeng Yang, Baishakhi Ray, Suman Jana. “Neuzz: Efficient fuzzing with neural program learning.” In 2019 IEEE Symposium on Security and Privacy (SP). IEEE, 2019.

[8] P. Godefroid, H. Peleg, and R. Singh, “Learn&fuzz: Machine learning for input fuzzing,” CoRR, vol. abs/1701.07232, 2017.

[9] Konstantin Bottinger, Patrice Godefroid, Rishabh Singh. “Deep reinforcement fuzzing.” 2018 IEEE Security and Privacy Workshops, SP Workshops 2018, San Francisco, CA, USA, May 24, 2018, pages 116–122, 2018.