제어 흐름 그래프 기반 스마트 컨트랙트 취약성 탐지 연구

정유영[1], 최라연[1], 임동혁[2]
[1]광운대학교 인공지능응용학과
[2]광운대학교 정보융합학부
yycheong@kw.ac.kr, chlfkdus123@kw.ac.kr, dhim@kw.ac.kr

Smart Contract Vulnerability Detection Study Based on Control Flow Graphs

Yoo-Young Cheong[1], La Yeon Choi[1], Dong-Hyuk Im[2]
[1]Dept. of Artificial Intelligence Applications, KwangWoon University
[2]School of Information Convergence, KwangWoon University

요 약

스마트 컨트랙트는 블록체인 상에서 실행되는 프로그램으로 복잡한 비즈니스 논리를 처리할 수 있다. 그러나 블록체인의 무결성과 조건에 따라 실행되는 특성을 이용한 악의적 사용으로 인하여 블록체인 보안에서 시급한 문제가 되고있다. 따라서 스마트 컨트랙트 취약성 탐지문제는 최근 많은 연구가 이루어지고 있다. 그러나 기존 연구의 대부분이 단일 유형의 취약성 여부에 대한 탐지에만 초점이 맞춰져 있어 여러 유형의 취약성에 대한 동시 식별이 어렵다. 이 문제를 해결하고자 본 연구에서는 스마트 컨트랙트 소스코드 제어 흐름 그래프를 기반으로 그래프의 forward edge와 backward edge를 고려한 신경망으로 그래프 구조를 학습한 후 그래프 multi-label classification을 진행하여 다중 취약성을 탐지할 수 있는 모델을 제안한다.

1. 서론

블록체인은 분산 및 공유 트랜잭션 원장이며, 합의 프로토콜에 따라 블록체인 네트워크가 이루어진다[1][2]. 비탈릭 부테린이 개발한 이더리움[3]은 블록체인 기반의 오픈 소스 분산 컴퓨팅 플랫폼으로 이더리움 상에서 실행되는 프로그램인 스마트 컨트랙트가 특징이다. 스마트 컨트랙트는 계약 조건을 소스 코드로 작성하여 자산을 관리하기 위한 규칙을 구현할 수 있다. 블록체인 네트워크에 배포된 스마트 컨트랙트는 블록체인 무결성에 따라 변경이 불가능하다. 따라서 프로그래밍적 결함이 있더라도 수정이 불가능하므로 안전하지 않은 스마트 컨트랙트는 악용되어 심각한 손실을 줄 수 있다[4]. 이러한 스마트 컨트랙트 보안을 위하여 개발자는 일반적으로 스마트 컨트랙트를 실행하기 전 취약성 여부에 대한 테스트를 진행한다. 그러나 수동적인 코드 분석은 코드 수가 많고 취약성의 종류가 다양하고 복잡하여 효율성이 떨어진다. 따라서 본 연구에서는 소스 코드 제어 흐름 그래프(Control Flow Graph)를 사용하여 다중 취약성을 정확하고 효과적으로 동식 식별하여 블록체인 플랫폼의 보안을 강화하고자 한다.

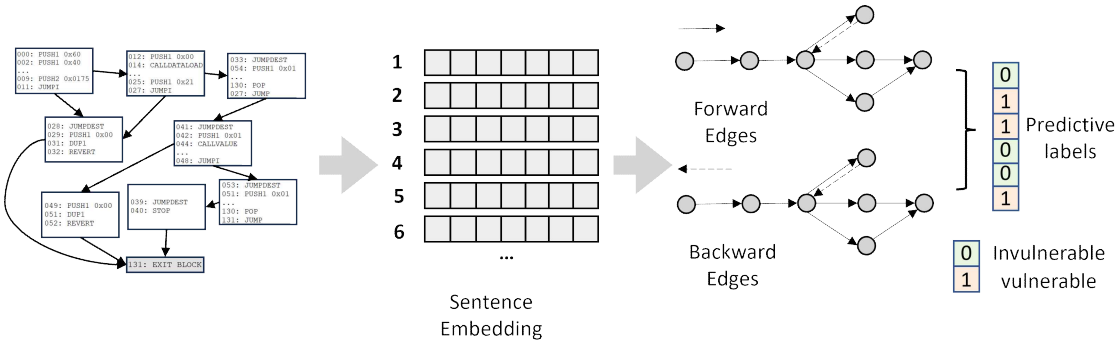
2. 연구 배경

스마트 컨트랙트를 구현하기 위하여 사용하는 Solidity는 객체 지향 고급 언어이다[5]. (그림 1)은 Solidity 언어로 작성한 bank를 구현하기 위한 예시이다. 이러한 Solidity로 작성한 스마트 컨트랙트를 이더리움 블록체인에서 실행하기 위하여 EVM(Ethereum Virtual Machine)에서 실행 가능한 이더리움 바이트코드로 소스코드를 컴파일해야 한다. 이더리움 바이트코드는 16진수 시퀀스로 표현되며 EVM이 실행할 수 있는 최소 명령어인 opcode로 파싱할 수 있다[6]. EtherSolve[6]는 EVM 바이트 코

```
pragma solidity ^0.8.0;

contract SimpleBank {
    mapping(address => uint256) private balances;
    function deposit() public payable {
        require(msg.value > 0, "Check Deposit amount");
        balances[msg.sender] += msg.value;
    }
    function withdraw(uint256 amount) public {
        require(balances[msg.sender] >= amount, "Check balance");
        balances[msg.sender] -= amount;
        payable(msg.sender).transfer(amount);
    }
    function getBalance() public view returns (uint256) {
        return balances[msg.sender];
    }
}
```

(그림 1) 스마트 컨트랙트 예시



(그림 2) 스마트 컨트랙트 취약성 탐지 모델 구조

드를 Opcode 시퀀스로 파싱하여 심볼릭 실행을 기반으로 jump 대상을 해결하여 정확한 CFG 추출을 목적으로 한다. 본 연구에서는 스마트 컨트랙트 EVM 바이트코드 CFG추출을 위해 EtherSolve를 사용한다.

3. 스마트 컨트랙트 취약성 탐지 모델

제안하는 모델은 (그림 2)와 같이 세 가지 주요 단계로 구성된다. (1) 스마트 컨트랙트 소스코드를 opcode로 파싱한 후 EtherSolve를 사용하여 CFG를 추출한다. (2) CFG의 각 정점은 프로그램 명령어의 시퀀스인 basic block으로 본 연구에서는 이 basic block을 하나의 문장으로 간주하여 처리한다. 각각의 opcode 시퀀스들을 문장 임베딩 기법인 Sent2Vec[7]을 적용하여 고정 길이 벡터로 변환한다. (3) 그래프 구조학습을 위하여 Zhang et al[8]이 제안한 CFGNN모델의 노드 정보 전파 기법을 적용하여 그래프 전역 정보를 융합한다. CFGNN은 CFG을 위한 모델로 본 연구에서는 CFGNN의 전체 방법론 중 일부분인 EXIT 정점으로 향하는 forward edge와 반대 방향인 backward edge를 이용하여 정점의 feature를 전파하는 기법만을 모델에 적용하였다. 세 가지의 단계를 거쳐 그래프의 전역 정보를 집계하여 multi-label classification을 수행한다.

4. 실험

성능 평가 실험을 위하여 Etherscan.io[9]에서 검증된 소스 코드를 가진 이더리움 스마트 컨트랙트를 수집하여 데이터셋을 구축하였다. 컴파일되어 이더리움 네트워크에 배포된 오픈 소스 스마트 컨트랙트 중 무작위로 2000개를 수집하였다. 6개의 취약성에 대한 multi-label classification 성능 실험을 진행하였으며, 레이블은 Oyente[10]를 사용하여 6개의 레이블을 [0 0 0 1 0 1]과 같은 형식으로 구성하였다. 본 연구에서는 Oyente에서 생성된 레이블이 신뢰할 수 있다고 가정한다. 실험 성능 평가 지표로 Accuracy와 Macro-F1, Micro-F1을 선택하였다. <표 1>에서 제안한 모델의 분류 성능을 확인할 수 있다.

<표 1> 스마트 컨트랙트 취약성 탐지 모델 성능 실험 결과

Method	Accuracy	Macro-F1	Micro-F1
Our approach	**0.85**	**0.83**	**0.85**

5. 결론

본 논문에서는 스마트 컨트랙트 소스 코드에서 제어 흐름 그래프를 기반으로 여러 유형의 취약성에 대하여 식별할 수 있는 다중 탐지 모델을 제안하였다. 실험을 통하여 다중 취약성 동식 식별에 대한 성능을 확인하였으며, 향후 작업으로 제어 흐름 그래프의 노드 간 방향 edge의 시퀀스 정보를 좀 더 효과적으로 집계할 수 있는 방향으로 연구하고자 한다.

Acknowledgement

본 연구는 과학기술정보통신부 및 정보통신기술진흥센터의 대학ICT연구센터지원사업의 연구결과로 수행되었음 (IITP-2023-2018-0-01417).

참고문헌

[1] Z. Liu, P. Qian, X. Wang, Y. Zhuang, L. Qiu and X. Wang, "Combining Graph Neural Networks With Expert Knowledge for Smart Contract Vulnerability Detection," in IEEE Transactions on Knowledge and Data Engineering, vol. 35, no. 2,

pp. 1296-1310, 1 Feb. 2023.
[2] Y. Y. Cheong, G. T. Kim, and D. H. Im. "Ethereum Phishing Scam Detection based on Graph Embedding and Semi-Supervised Learning." KIPS Transactions on Computer and Communication Systems, vol. 12, no. 5, pp. 165-170, 2023.
[3] C. Dannen, Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners, 1st ed., Apress, 2017.
[4] Z. A. Khan and A. Siami Namin, "Ethereum Smart Contracts: Vulnerabilities and their Classifications," 2020 IEEE International Conference on Big Data (Big Data), Atlanta, GA, USA, 2020, pp. 1-10.
[5] Ethereum. Solidity documentation. Website. [Accessed: 2023-09-21]. [Online]. Available: https://solidity.readthedocs.io/
[6] F. Contro, M. Crosara, M. Ceccato and M. D. Preda, "EtherSolve: Computing an Accurate Control-Flow Graph from Ethereum Bytecode," 2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC), Madrid, Spain, 2021, pp. 127-137.
[7] M. Pagliardini. P. Gupta, and M. Jaggi. "Unsupervised learning of sentence embeddings using compositional n-gram features," 2017. [Online]. Available: https://arxiv.org/abs/1703.02507.
[8] J. Zhang, X. Wang, H. Zhang, H. Sun, X. Liu, C. Hu, Y. Liu. "Detecting Condition-Related Bugs with Control Flow Graph Neural Network," 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis(ISSTA), Seattle, WA, USA, 2023, pp. 1370 - 1382.
[9] Etherscan.io. Etherscan. [accessed: 2023-09-21]. [Online]. Available: https://etherscan.io/
[10] L. Luu, D. H Chu, Hrishi Olickel, Prateek Saxena, and Aquinas Hobor. 2016. "Making Smart Contracts Smarter," 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS), New York, NY, USA, 2016, pp. 254 - 269.