

머신 러닝을 활용한 스마트 컨트랙트 취약점 탐지 연구 동향

이 의 찬*, 방 지 원*, 최 미 정°

A Survey on Vulnerability Detection in Smart Contract Using Machine Learning Techniques

UI-Chan Lee*, Jiwon Bang*, Mi-Jung Choi°

요 약

최근 블록체인 기술이 다양한 산업(금융, 의료)에서 활발히 활용되며, 블록체인의 핵심 기술 중 하나인 스마트 컨트랙트(Smart Contract) 또한 큰 관심을 받고 있다. 스마트 컨트랙트는 제 3자의 개입 없이 탈중앙화된 환경에서 신뢰성 있는 거래 기능을 제공한다. 하지만 스마트 컨트랙트의 사용이 늘어남에 따라 여러 취약점이 노출될 가능성 역시 증가하고 있다. 실제 스마트 컨트랙트의 여러 보안 취약점으로 인한 금전적인 피해 사례들이 보고되고 있다. 이러한 문제를 방지하기 위해 스마트 컨트랙트 코드 내의 취약점을 사전에 탐지하는 연구가 활발히 진행되고 있다. 최근에는 다양한 분야에서 뛰어난 성능을 보이는 머신 러닝 기법을 도입한 스마트 컨트랙트 취약점 탐지도 활발히 연구되고 있다. 본 논문에서는 스마트 컨트랙트의 주요 취약점들과 머신 러닝 기법을 활용한 스마트 컨트랙트 취약점 탐지 연구를 소개한다. 또한 선정한 연구를 입력 데이터 별로 분류하고 각 연구의 특징을 조사한다.

Key Words : Blockchain, Ethereum, Smart Contract, Vulnerability, Machine Learning, Detection

ABSTRACT

Blockchain technology has been actively utilized in various industries, such as finance and healthcare, with growing interest in smart contracts, one of the core technologies of blockchain. Smart contracts enable reliable transactions in a decentralized environment without the involvement of third parties. However, as the use of smart contracts increases, the potential for various vulnerabilities to be exposed also rises. Indeed, there have been reported cases of financial losses caused by security vulnerabilities in smart contracts. To address these issues, research on detecting vulnerabilities in smart contract code in advance has been actively conducted. Recently, machine learning techniques, which have demonstrated excellent performance in various fields, have been applied to the detection of smart contract vulnerabilities. This paper introduces the key vulnerabilities of smart contracts and explores studies that utilize machine learning techniques for vulnerability detection. Additionally, it categorizes the selected studies based on input data and investigates the characteristics of each study.

※본 연구는 보건복지부의 재원으로 한국보건산업진흥원의 보건의료기술연구개발사업지원에 의하여 이루어진 것임(과제 번호: HI23C0733)

♦ First Author : kangwon National University Department of Computer Science and Engineering, jungkls3@kangwon.ac.kr

° Corresponding Author : Kangwon National University Department of Computer Science and Engineering, IGP. in Medical Bigdata Convergence, and Department of Data Science, mjchoi@kangwon.ac.kr

* kangwon National University IGP. in Medical Bigdata Convergence, and Department of Data Science, jiwonbang@kangwon.ac.kr

논문번호 : KNOM2024-02-03, Received November 18, 2024; Revised December 13, 2024; Accepted December 20, 2024

I. 서 론

블록체인 기술은 2008년 사토시 나카모토의 Bitcoin: A Peer-to-Peer Electronic Cash System[1]을 통해 처음 공개되었다. 블록체인은 중앙 기관 없이도 거래의 투명성과 무결성을 확보할 수 있는 기술로, 금융, 의료, 물류 등 다양한 산업에서 활용되고 있다. 블록체인 기술이 주목받기 시작하면서 스마트 컨트랙트(Smart Contract)에 대한 관심 또한 증가하였다. 스마트 컨트랙트는 2015년 비탈릭 부테린이 이더리움(Ethereum)을 발표하면서, 조건에 따라 자동으로 계약이 실행되는 시스템이 등장했다. 스마트 컨트랙트는 특정 조건이 충족될 때만 계약이 이행되므로, 거래 당사자 간에 신뢰할 수 있는 제 3자의 개입 없이도 안전한 거래가 가능하다.

하지만 스마트 컨트랙트에는 다양한 보안 취약점이 존재하고, 이로 인한 실제 피해 사례들[2, 3]이 꾸준히 보고 되고 있다. 블록체인에 배포된 스마트 컨트랙트는 수정이 불가능한 특징이 있다. 이로 인해, 스마트 컨트랙트 내에 보안 취약점이 발생했을 때 그 피해가 크다.

실제로, 2016년에 설립된 The DAO는 스마트 컨트랙트의 취약점으로 인해 금전적인 피해를 입었다. DAO(Decentralized Autonomous Organization)는 전통적인 조직과 달리 중앙 관리자 없이 스마트 컨트랙트를 통해 규칙과 절차가 자동으로 실행되는 탈중앙화된 자율 조직이다. 익명의 공격자가 The DAO 컨트랙트에 존재하는 재진입(reentrancy) 취약점을 악용하여 대략 360만 ETH를 탈취했으며, 이는 당시 시세로 약 640억원 상당 금액이다[4]. 다른 사건으로 2021년 Cream Finance에서 발생한 해킹 사건이 있다. Cream Finance는 탈중앙화 금융(DeFi) 플랫폼으로, 사용자가 자산을 빌리고 빌려주는 기능을 제공한다. 그러나 스마트 컨트랙트 코드에 존재하는 플래시 론(flash loan) 취약점을 악용하여 약 1600억원의 손실이 발생하였다[5].

이러한 피해를 방지하기 위해 스마트 컨트랙트를 배포하기 전에 취약점을 탐지하는 연구가 활발히 이루어지고 있다. 정적 분석(static analysis), 기호 실행(symbolic execution)과 같은 고전적인 탐지 방법뿐만 아니라 최근에는 머신 러닝을 활용한 방법이 연구되고 있다. 또한 스마트 컨트랙트

트 취약점 탐지 연구에 대한 연구 동향 논문 또한 지속적으로 발표되고 있다[6, 7, 8, 9].

기존의 동향 논문은 주로 머신 러닝 모델에 초점을 맞추어 모델을 중심으로 각 연구의 특징을 제시했다. 하지만 이러한 접근 방식은 머신 러닝 기법에서 핵심적인 요소인 입력 데이터에 대한 정보 전달이 부족하다. 머신 러닝을 활용한 많은 연구가 진행되는 상황에서 입력 데이터에 대한 체계적인 분류와 분석이 필요하다.

본 논문은 스마트 컨트랙트 취약점 탐지 연구에서 입력 데이터 유형의 중요성을 강조하며, 이를 기반으로 연구를 입력 데이터 유형(바이트코드, 오피코드, 소스 코드)별로 분류하고 분석한다. 또한 스마트 컨트랙트에 존재 가능한 주요 취약점을 정리하고, 선정한 연구의 주요 공헌도를 설명한다. 이를 통해 연구자들이 각 데이터 유형별 연구 동향과 특징을 파악하고, 데이터 선택 및 모델 설계 전략을 수립하는 데 가이드를 제공한다.

본 논문의 구성은 다음과 같다. 제 2장에선 블록체인과 스마트 컨트랙트 같은 배경 지식을 소개하고 제 3장에선 스마트 컨트랙트의 주요 취약점들에 대해 소개한다. 제 4장에선 머신 러닝을 활용한 취약점 탐지 연구를 분류하고 특징을 설명한다. 마지막 제 5장에서는 연구 결과를 요약하고 향후 연구에 대해 논의한다.

II. 배경

2.1 블록체인

블록체인은 데이터를 중앙 서버가 아닌 여러 노드에 분산 저장하는 방식으로 탈중앙성, 보안성, 투명성을 제공하는 분산 원장(distributed ledger) 기술이다. 블록체인의 각 노드는 블록이라고 불리는 데이터를 저장하고, 블록들은 체인 형태로 연결되어 불변성을 보장한다. 블록의 구조는 거래 기록, 이전 블록의 해시값, 현재 블록의 해시값 등의 정보를 포함하고 있으며, 블록체인마다 구조는 다를 수 있다.

블록체인 기술은 익명의 개발자, 사토시 나카모토에 의해 처음 공개되었다. 전자화폐 시스템에서 발생하는 이중 지출(double-spending) 문제를 작업 증명(proof of work)이란 합의 알고리즘을 통해 해결하였다. 작업 증명 방식은 SHA 해시함수를 활용하여 특정 해시값을 찾는 계산 작업을 통

해 새로운 블록을 생성한다. 이 과정에서 거래 내역을 조작하기 위해선 많은 연산 자원이 필요하므로 이중 지불 문제를 해결할 수 있다. 합의 알고리즘을 통해 네트워크 내의 모든 노드가 동일한 거래 내역을 공유하고, 새로운 거래를 추가할 때 이를 검증하여 네트워크의 무결성을 유지할 수 있다.

블록체인의 특징은 다음과 같다. 블록체인은 중앙 관리자 없이도 네트워크의 모든 참여자가 동일한 원장을 유지한다. 이를 통해 중개자 없이도 신뢰할 수 있는 탈중앙화된 시스템을 구축할 수 있다. 블록체인 네트워크에 참여하는 누구나 원장의 내용을 열람할 수 있으므로 거래의 투명성을 보장한다. 한번 블록체인에 기록된 데이터는 변경할 수 없다. 모든 거래는 이전 블록과 연결되어 중간에 거래 내용을 수정하려면 전체 체인을 변경해야 하는데 이는 사실상 불가능하다.

2.2 스마트 컨트랙트

스마트 컨트랙트는 1994년 닉 제보(Nick Szabo)가 최초로 제안한 개념이다[10]. 스마트 컨트랙트는 계약 실행에 필요한 조건을 명시하고, 해당 조건이 만족되면 계약이 자동으로 실행되는 프로그램이다. 제 3자의 개입 없이도 신뢰할 수 있는 계약 이행이 가능하도록 설계되었다. 스마트 컨트랙트는 데이터 위변조 문제와 기술적 한계로 인해 실제 구현이 어려웠으나, 2015년 비탈릭 부테린은 스마트 컨트랙트를 블록체인에 도입한 이더리움(Ethereum)을 발표하면서 기술적 구현이 가능해졌다.

스마트 컨트랙트는 블록체인에 배포된 이후 수정이 불가능하다는 특징이 있다. 이러한 불변성은 스마트 컨트랙트의 신뢰성을 보장하는 중요한 요소지만, 동시에 치명적인 보안 취약점이 될 수 있다. 스마트 컨트랙트 코드를 배포한 후 발견되는 오류나 보안 문제는 해결하기 어렵기 때문에, 스마트 컨트랙트의 보안 검증은 매우 중요하다.

스마트 컨트랙트는 실행 과정에서 크게 3가지 형태를 거친다. 그림 1은 스마트 컨트랙트가 실행되는 과정에서 형태의 변화를 나타낸다. 소스 코드(source code)는 사람이 이해할 수 있는 형태로, 솔리디티(Solidity)[11], 바이퍼(Vyper)[12]와 같은 다양한 스마트 컨트랙트 전용 프로그래밍 언어로 작성된다. 개발자는 이를 통해 함수, 상태 변수 등을 설정하여 스마트 컨트랙트의 로직을 담을 수 있다. 바이트코드(bytecode)는 고수준

언어로 작성된 스마트 컨트랙트 소스 코드를 컴파일러가 변환한 결과물이다. 바이트코드는 16진수 형태로 나타나며, 이더리움의 독립적인 실행 환경인 이더리움 가상 머신(Ethereum Virtual Machine, EVM)을 통해 실행된다. 오퍼코드(opcode)는 컴파일된 바이트코드를 구성하는 기본 명령어 단위로, EVM에서 이해하고 실행할 수 있는 최소 실행 명령어이다. 오퍼코드에는 여러 명령어(ADD, MUL, PUSH)가 포함된다. 결론적으로 솔리디티와 같은 고수준 언어로 작성된 스마트 컨트랙트는 컴파일 과정을 거쳐 바이트코드로 변환되며, 이 바이트코드는 오퍼코드로 구성되어 EVM에서 실행된다.

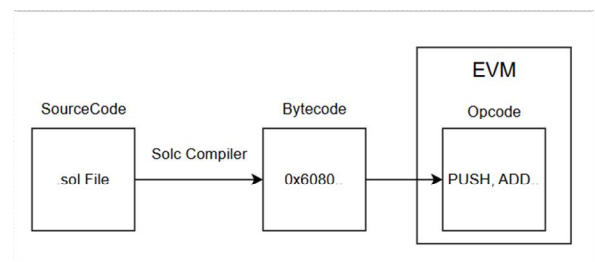


그림 1. 스마트 컨트랙트 실행 과정

Fig. 1. The process of executing a smart contract

Ⅲ. 스마트 컨트랙트 취약점

스마트 컨트랙트는 블록체인 환경에서 자동으로 실행되는 코드로, 한 번 배포되면 수정이 어려워 코드 작성 과정에서 보안이 매우 중요하다. 그러나 스마트 컨트랙트 코드 작성 시 개발자의 실수, 코드 내 버그, 잘못된 API 호출 등으로 인해 다양한 보안 취약점이 발생할 수 있다. 본 논문에서는 영국의 NCC Group에서 공개한 DASP(Decentralized Application Security Project)[13]의 9가지 주요 취약점을 기반으로 스마트 컨트랙트 취약점을 설명한다:

- RE(Reentrancy) : The DAO 사건에서 발생한 주요 취약점인 재진입 공격은 외부 컨트랙트를 호출할 때 원래 컨트랙트로 반복해서 재진입할 수 있는 구조에서 발생한다. 이로 인해 자금의 중복 인출 같은 문제가 발생할 수 있다.
- AC(Access Control) : AC는 모든 프로그램에서 흔하게 발생하는 취약점으로 권한 제어가 제대로 설정되지 않는 것을 의미한다. 이에 악의적인 사용자가 스마트 컨트랙트의 중요한

기능에 접근할 수 있게 되어 자산의 탈취, 관리 기능의 조작 등 다양한 보안 문제가 발생할 수 있다.

- AE(Arithmetic Error) : 정수 오버플로(overflow) 및 언더플로(underflow)로도 알려진 취약점으로, 계산이 예상 범위를 벗어날 때 발생하여 잘못된 결과를 초래할 수 있다.
- UE(Unhandled Exception) : 저수준(low level) 함수 호출(call, send) 후 return 값에 대한 처리를 하지 않으면 저수준 함수 실행이 실패해도 계약이 계속 진행될 수 있다. 이에 자금 손실 또는 의도하지 않은 문제가 발생할 수 있다.
- DoS(Denial of Service) : 스마트 컨트랙트에 존재하는 특정 기능의 과도한 실행을 유도하거나, 자원을 독점해 스마트 컨트랙트가 정상적으로 작동하지 못하게 할 수 있다. 이로 인해 자원 낭비와 기능의 비정상적 동작 등의 문제를 초래할 수 있다.
- BR(Bad Randomness) : 스마트 컨트랙트에서 난수를 예측 가능한 방식으로 생성하면 공격자가 결과를 조작할 수 있다. 난수 생성에 문제가 있으면 무작위성이 중요한 서비스(게임, 복권 등)에서 부정확한 결과가 발생할 수 있다.
- FR(Front-Running) : 공격자가 네트워크상의 트랜잭션을 미리 감지하고, 더 높은 가스비를 지불하여 자신의 트랜잭션을 먼저 처리되도록 함으로써 이득을 취할 수 있다. race condition, transaction ordering dependence로도 알려져 있다. 이를 통해 다른 사용자의 거래를 방해하거나 유리한 조건을 선점하여 부당한 이익을 취할 수 있다.
- TD(Timestamp Dependence) : 블록 타임스탬프 취약점은 스마트 컨트랙트에서 블록의 타임스탬프를 사용하여 특정 조건을 제어하는 경우 발생한다. 악의적인 채굴자가 타임스탬프를 조작할 수 있다. 타임스탬프 조작으로 인해 시간에 따라 실행 조건이 달라질 수 있는 스마트 컨트랙트가 영향을 받을 수 있다.
- SA(Short Address Attack) : 해당 취약점은 트랜잭션 데이터의 길이가 올바르게 확인되지 않을 때 발생한다. 악의적인 공격자가 의도적으로 일부분을 잘라낸 주소를 제공할 경우, 스마트 컨트랙트가 예상하지 못한 방식으로 입력을 해석하게 만들 수 있다.

IV. 머신 러닝을 활용한 취약점 탐지 연구

머신 러닝이 다양한 분야에서 뛰어난 성능을 보여, 스마트 컨트랙트 취약점 탐지에도 이를 활용한 방법이 연구되고 있다. 본 논문에서는 최신 연구 동향 분석을 위해 2021년 이후에 발표된 머신 러닝을 활용한 연구를 선정했다. 머신 러닝을 활용한 연구에서 주로 사용되는 입력 데이터는 바이트코드, 오피코드, 소스 코드이다. 입력 데이터 유형별 연구 동향을 분석하기 위해, 이를 분류기준으로 선정했다. 선정된 논문을 주요 입력 데이터를 기준으로 분류하고, 각 연구의 특징을 소개했다. 또한 주요 공헌도와 탐지 성능을 표로 정리하여 명시했다. 표의 Model 필드는 연구에서 사용된 모델을 나타낸다. 여러 모델이 연구된 경우 "Many"로 표시했다. Vulnerabilities 필드는 해당 연구에서 탐지한 주요 취약점 목록을, Contribution 필드는 연구의 핵심 공헌도를 나타낸다. 성능 지표의 경우 F1-score와 정밀도(precision)의 평균 값을 명시했다.

4.1 바이트코드 기반 연구

바이트코드는 스마트 컨트랙트 소스 코드가 컴파일 과정을 통해 변환된 형태를 의미한다. 바이트코드 기반 연구의 가장 큰 특징은 소스 코드가 공개되지 않은 경우에도 취약점을 탐지할 수 있다는 점이다. 표 1은 본 논문에서 조사한 바이트코드 기반 연구들을 연도별로 오름차순으로 정리하고, 각 연구의 주요 공헌과 탐지 가능한 취약점 목록을 명시하였다.

ESCORT[14]는 DNN(Deep Neural Network)과 전이 학습(transfer learning)을 활용한 프레임워크이다. ESCORT는 특징 추출기를 통해 컨트랙트의 의미론적 특징을 추출하고, 다중 출력(multi output) 신경망 구조를 채택하여 각 취약점을 탐지한다. 또한 새로운 취약점 유형에 대한 경량 전이 학습을 지원하여 최소한의 수정으로 확장이 가능하다. 또한 앞서 소개한 주요 취약점 이외에도 CD(Callstack Depth)와 AV(Assert Violation)을 탐지한다. CD는 EVM의 호출 스택 깊이 제한을 악용하여 컨트랙트가 추가 호출을 처리하지 못하도록 방해할 수 있다. AV는 assert문이 false가 되도록 유도하여,

컨트랙트를 오류 상태로 만들 수 있는 취약점이다. 저자들은 스마트 컨트랙트를 수집하고 기존 도구를 통한 라벨링으로 데이터셋을 구축했다.

VSCL[15]은 DNN과 메트릭 학습(metric learning)을 결합한 프레임워크이다. VSCL은 스마트 컨트랙트의 바이트코드를 CFG(Control Flow Graph)로 변환하여 실행 경로와 흐름을 파악한다. 각 CFG의 노드는 컨트랙트의 실행 가능한 경로를 나타내며 CFG 시퀀스는 n-그램과 TF-IDF(Term Frequency-Inverse Document Frequency)를 통해 특징 벡터로 변환된다. 저자들은 메트릭 학습을 통해 클래스 내 분산을 줄이고, 클래스 간 분리를 향상 시켜 성능을 높였다. 또한 3가지 도구(Oyente, Mythril, Vandal) 중 2가지 이상이 탐지한 경우를 취약점으로 고려한 데이터셋과 3가지 도구 중 하나라도 탐지한 경우를 취약점으로 고려한 데이터셋, 총 두 가지를 사용했다. RE, UE 같은 주요 취약점 이외에도 CD와 같은 취약점을 탐지하며, 다각도로 성능 평가를 진행했다.

J. Huang et al.[16]의 저자들은 바이트코드 매칭을 위해 그래프 임베딩(graph embedding)을 도입했다. 그래프를 수치화된 벡터로 변환해 유사성을 측정하고 취약점을 탐지했다. 이 과정에서 다양한 컴파일러 버전으로 인한 바이트코드의 불일치와 불필요한 코드로 인한 간섭 문제를 해결하기 위해, 정규화 및 슬라이싱 기법을 제안했다. 스마트 컨트랙트의 데이터 및 제어 흐름을 추적하여, 취약점과 관련 없는 코드를 제거함으로써 탐지 정확도의 향상을 보였다. 성능 평가를 위해 Mist[17], Etherscan[18]에서 컨트랙트를 수집하고, 데이터셋을 구성했다.

Attention-based[19]의 저자들은 CNN(Convolutional Neural Network)과 셀프 어텐션(self-attention) 메커니즘을 결합한 모델인 ABCNN(Attention-Based CNN)을 제안했다. 오피코드의 특징을 추출하고 셀프 어텐션 기반 임베딩을 통해 스마트 컨트랙트 명령어 간의 연관성을 파악한다. 또한 전처리 과정을 통해 취약점을 나타내는 패턴을 효과적으로 검출하도록 설계했다. ABCNN은 기존의 정적 분석 도구인 Oyente와 Mythril 대비 높은 성능을 보였다. 성능 평가를 위해 Etherscan에서 컨트랙트를 수집하여 데이터셋을 구성했다.

SmartDagger[20]는 두 개 이상의 스마트

컨트랙트 간의 호출 경로에서 발생하는 크로스-컨트랙트 취약점 탐지를 위한 도구이다. 서로 다른 스마트 컨트랙트 간의 상호작용 중 나타나는 문맥 정보, 제어 흐름 의존성, 데이터 흐름 의존성 등을 분석한다. SmartDagger는 바이트코드에서 호출 경로 내의 누락된 컨트랙트 속성을 복원하기 위해 NMT(Neural Machine Translation)를 사용한다. NMT를 통해 스마트 컨트랙트의 의미론적 정보를 되찾고 분석을 수행한다. 또한 크로스-컨트랙트 취약점과 관련된 함수만을 선택적으로 분석하고, 이전의 데이터 흐름 결과를 재사용하여 성능향상을 보였다. 저자들은 성능 평가를 위해 Etherscan에서 컨트랙트 수집하여 데이터셋을 구성했다.

S. J. Qin et al.[21]는 중요 경로 조합(Critical Combination Path, CCP)과 딥러닝을 활용한 취약점 탐지 방법을 제안했다. 저자들은 CFG(Control Flow Graph)와 상태 변수 종속성 그래프를 생성하고, 취약점과 관련된 경로만을 필터링하여 CCP를 구성했다. 또한, 오피코드에서 불필요한 연산자를 제거함으로써 데이터의 노이즈를 줄였다. 정규화된 CCP를 SimHash를 사용해 벡터화하고, 이를 그레이스케일 이미지로 변환하여 ResNet50의 입력으로 활용했다. 성능 평가를 위해 여러 논문과 Awesome-Buggy-ERC20-Tokens[22]에서 컨트랙트를 수집하고 데이터셋을 구성했다.

P. Qian et al.[23]의 저자들은 교사(teacher)-학생(student) 네트워크로 구성된 크로스-모달리티(cross-modality) 상호 학습 프레임워크를 제안했다. 교사 네트워크는 소스 코드와 바이트코드 모두를 사용하여 취약점 정보를 학습하고, 학생 네트워크는 바이트코드 정보만을 학습한다. 학습 중 두 네트워크가 서로 정보를 교환하여, 학생 네트워크가 바이트코드만으로 소스 코드에서 파생된 정보(제어 흐름, 데이터 흐름)를 예측하도록 훈련한다. 이를 통해 탐지 단계에서 바이트코드만으로 RE, TD, AE, DC(DelegateCall)와 같은 여러 취약점을 탐지한다. DC는 호출자의 실행 환경에서 피호출자의 코드를 실행하므로, 환경 차이에 의해 의도치 않은 결과가 발생 가능한 취약점이다. 저자들은 성능 평가를 위해 여러 출처(Github, Ethereum 네트워크)에서 컨트랙트를 수집하고, 데이터셋을 구성했다.

VulHunter[24]는 다중 인스턴스 학습

표 1. 바이트코드 기반 스마트 컨트랙트 취약점 탐지 연구

Table 1. Research on Bytecode-Based Smart Contract Vulnerability Detection

Research	Model	Vulnerabilities	Contribution
ESCORT[14]	DNN	RE, DoS, FR, AC, Callstack Depth, Assert Violation	- 경량 전이 학습을 지원하는 최초의 DNN 기반 프레임워크 - 최소한의 수정과 재학습으로 확장 가능 - 성능 평가: 정밀도-97%, F1-score-95%
VSCL[15]	CNN	RE, UE, FR, TD, AC, Callstack Depth	- CFG 기반 특징 벡터화와 Metric Learning을 적용한 DNN 모델을 개발 - 성능 평가: 정밀도-87%, F1-score-76%
J. Huang et al.[16]	Graph Embedding	RE, AE, BR, AE, UE	- 바이트코드 매칭을 위해 그래프 임베딩 기법을 최초로 도입 - 성능 평가: 정밀도-91.95%
Attention-based[19]	Hybrid Model	RE, AE, TD	- CNN과 self-attention을 결합한 모델을 제안 - 성능 평가: 정밀도-83%, F1-score-88%
SmartDagger[20]	Hybrid Model	RE, TD, DoS, AC	- 크로스-컨트랙트 취약점 탐지를 위한 최초의 도구 - 바이트코드에서 누락된 컨트랙트 속성을 복원, 분석 - 성능 평가: 정밀도-88%
S. J. Qin et al.[21]	CNN	RE, AC, AE, TD	- CCP 기법을 사용한 코드 필터링을 통해 노이즈를 감소 - 정규화된 CCP를 SimHash를 사용해 벡터화한 후 분류 - 성능 평가: 정밀도-81.20%, F1-score-82.89%
P. Qian et al.[23]	Hybrid Model	RE, TD, AE, Delegatecall	- 최초로 상호 학습을 활용한 연구 - 두 개의 네트워크가 상호 학습을 통해 취약점을 탐지하는 새로운 접근 방식을 제안 - 성능 평가: 정밀도 87%, F1-score-84%
VulHunter[24]	Hybrid Model	RE, TD, AE 외 26가지 취약점	- 다중 인스턴스 학습 기반 프레임워크 - 분류 결과와 함께 결함 위치와 해당 인스턴스 식별 - 성능 평가: F1-score-86%
COBRA[25]	Hybrid Model	RE, AE, UE, FR, TD	- 시맨틱 컨텍스트에서 함수 입력 매개변수를 추출하기 위해 seq2seq 구조를 활용 - 성능 평가: F1-score-95%
L. Prifti et al.[27]	CNN, RNN	RE, AE, UE, AC, Other	- 여러 모델(ResNet, Inception, LSTM)과 RNN, CNN 모델의 성능을 비교 - 4가지 취약점을 제외한 기타 취약점을 Other로 표시 - 성능 평가: Micro F1-score-89%

(multi-instance learning) 기반 프레임워크이다. 저자들은 스마트 컨트랙트의 실행 흐름을 분석하기 위해 CFG를 구축하고, 각 경로를 개별 실행 인스턴스 단위로 나눴다. 이를 Bi-LSTM(Long Short-Term Memory)을 활용한 다중 인스턴스 학습을 통해 바이트코드 시퀀스의 앞뒤 맥락을 모두 학습하도록 했다. 또한 셀프 어텐션(self-attention) 메커니즘을 도입하여 중요한 부분에 높은 가중치를 부여함으로써, 모델이 취약한 경로를 집중적으로 학습하도록 설계했다. 저자들은 EtherScan, GitHub 및 다른 공개 데이터셋을 활용하여 데이터셋을 구성했다. 이를 통해 여러 취약점(RE, TD, Tx-origin, Send-transfer)을 탐지하고 VulHunter의 성능을 평가했다.

COBRA[25]의 저자들은 의미적 맥락과 함수 인터페이스 정보를 결합한 SRIF(Signatures Reverse Inference from Functions) 모델

을 제안했다. 시퀀스-투-시퀀스(seq2seq)구조를 사용한 인코더(encoder) 모델을 채택하여 스마트 컨트랙트의 의미적 맥락을 파악하고, SRIF를 통해 의미적 맥락과 함수의 고유한 식별자 역할을 하는 함수 시그니처 간의 매핑을 수행한다. 이를 통해 ABI(Application Binary Interface)가 없는 경우에도 사용된 함수 정보를 복원하여 컨트랙트 간 상호작용에 포함된 취약점을 탐지 가능하도록 설계했다. XBlock_Eth dataset[26]을 활용해 데이터셋을 구성했다.

L. Prifti et al.[27]의 저자들은 CNN과 RNN(Recurrent Neural Network) 모델을 사용하여 스마트 컨트랙트 내 취약점을 탐지했다. 입력 데이터를 쌍(pair) 단위로 나누어 EVM 바이트코드의 특징을 더 잘 파악하도록 설계했다. 또한 RNN 모델은 약 80,000개의 파라미터를, CNN 모델은 약 36,000개의 파라미터를 사용하여 상대적으로 단순한 구조로 설계했다. 다른 논문

에서 사용된 데이터셋[28]을 활용해 다른 모델들(ResNet, Inception 등)과 비교하며 성능을 평가했다.

4.2 오피코드 기반 연구

오피코드는 바이트코드가 EVM이 이해할 수 있는 명령어 형태로 변환된 것을 의미한다. 오피코드는 명령어의 시퀀스 정보를 포함하여, 이를 활용한 취약점 탐지가 주로 연구된다. 표 2에서는 표 1과 동일한 방식으로 연구를 정리하였으며, 주요 공헌과 탐지 가능한 취약점 목록을 명시하였다.

TokenCheck[29]는 LSTM모델을 활용한 도구이다. LSTM은 오피코드의 시퀀스(sequence) 데이터를 학습하여 복잡한 ERC-20 토큰 컨트랙트에 특화된 취약점 탐지를 보였다. 성능 최적화를 위해 EVM 오피코드를 전 처리하여 데이터의 길이를 줄였다. 저자들은 Ethereum Public Dataset[30]을 통해 컨트랙트를 수집하고 데이터셋을 구성했다. 또한 취약점을 Suicidal, Produgal, Greedy로 분류하고 탐지했다. Suicidal은 self-destruct 명령어가 보호되지 않아 강제 종료될 수 있는 경우를 말하며, Produgal은 임의의 주소로 이더(Ether)를 보낼 수 있는 취약점이다. Greedy는 이더를 수신할 수 있으나 반환하지 못하는 상황을 말한다.

J. Su et al.[31]의 저자들은 강화학습(reinforcement learning)과 퍼징(fuzzing) 기법을 결합한 RLF(Reinforcement Learning Fuzzer)를 제안했다. RLF는 스마트 컨트랙트 퍼징을 마르코프 결정 프로세스(markov decision process)로 모델링하여 트랜잭션 시퀀스를 생성하는 강화 학습 프레임워크이다. RLF는 DQN(Deep Q-Network)을 사용하여 트랜잭션 시퀀스를 생성하는 과정에서 보상을 최대화하도록 학습한다. 저자들은 취약성 발견에 중점을 둔 보상 외에도 코드 커버리지(coverage)에 관한 보상을 추가하여 복잡한 취약점을 탐지할 수 있도록 유도했다. 또한 XBlock[32]과 VeriSmart[33]에서 컨트랙트를 수집하여 데이터셋을 구성했다. 이를 통해 UE, AC, DC 같은 주요 취약점과 EL(Ether Leaking), EF(Ether Freezing) 등 여러 취약점을 탐지했다. EF는 이더를 수신 가능하지만 출금할 수 없는 구조를 가질 경우 이더가 잠길 수

있다. EL은 잘못된 접근 제어, 설계 오류 등으로 이더가 탈취 가능한 취약점이다.

J. Huang et al.[34]의 저자들은 다중 작업 학습(Multi-Task Learning) 기반 모델을 제안했다. 다중 학습 모델은 하위 레이어와 작업 레이어로 구성된다. 하위 레이어는 입력 데이터를 처리하고, 어텐션 메커니즘을 적용하여 의미 정보를 학습한다. 작업별 레이어는 CNN을 사용하여 분류 네트워크를 구축하고, 하위 레이어에서 학습된 특징 벡터를 사용해 각 작업의 목표를 달성한다. 다중 작업 모델을 통해 여러 작업을 동시에 수행할 수 있어 단일 작업 모델에 비해 시간, 계산 비용을 절감할 수 있다. 저자들은 XBlock에서 컨트랙트를 수집하여 데이터셋을 구성하고 RE, AE, UA(Unknown Address) 취약점을 탐지했다. UA는 컨트랙트 주소로 알려지지 않은 값이 들어올 경우, 예기치 않은 결과가 발생할 수 있다.

S. Gopali et al.[35]의 저자들은 LSTM 및 TCN(Temporal Convolutional Network) 모델을 이용한 기법을 제안했다. TCN을 통해 컨트랙트 간의 종속 관계를 처리하도록 설계했다. 또한 추가적인 컨트랙트를 수집하고 오버 샘플링과 언더샘플링을 적용하여, 기존 연구들에서 사용하던 데이터셋의 불균형 문제를 개선했다. 저자들은 취약점을 Suicidal, Produgal, Greedy로 분류하고 Ethereum Public Dataset을 활용한 데이터셋으로 LSTM과 TCN의 성능을 비교했다.

Z. Wu et al.[36]의 저자들은 딥러닝 기반 방법을 제안했다. unigram과 bigram을 사용하여 스마트 컨트랙트의 오피코드를 분석하고, 특징 벡터를 생성했다. 또한 TF-IDF 방식을 적용하여 드물게 발생하는 잠재적 악성 특징을 강조했다. 저자들은 이더리움 네트워크에서 컨트랙트를 수집하고, SMOTE(Synthetic Minority Oversampling Technique) 기법을 사용해 데이터 불균형 문제를 해결했다. 이를 통해 다양한 모델(CNN, LSTM, CNN-BiLSTM, ResNets)의 성능을 평가했다.

W2V-SA[37]는 여러 모델을 결합한 하이브리드 모델이다. W2V-SA는 CNN과 셀프 어텐션을 결합하여 스마트 컨트랙트의 복잡한 관계를 파악하고 취약점을 탐지한다. 워드 임베딩 레이어에선 스마트 컨트랙트를 word2vec 방식으로 벡터화하고, 이를 학습한다. CNN 레이어는 입력 데이터에

표 2. 오피코드 기반 스마트 컨트랙트 취약점 탐지 연구

Table 2. Research on Opcode-Based Smart Contract Vulnerability Detection

Research	Model	Vulnerabilities	Contribution
TokenCheck[29]	LSTM	Suicidal, Prodigal, Greedy	- LSTM 활용해 보다 복잡한 ERC-20 토큰의 취약점 탐지 방법을 제안 - 취약점을 보다 큰 3가지 범주로 분류 - 성능 평가: 정밀도-93%, F1-score-93%
J. Su et al.[31]	DNN	UE, AC, TD Ether Leaking, Delegatecall, Ether Freezing	- 강화학습과 DQN을 결합하여 새로운 퍼징 기법인 RLF를 제안 - 성능 평가: 정밀도-98.6%
J. Huang et al.[34]	Hybrid Model	RE, AE, Unknown Address	- 다중 작업 학습과 CNN 구조를 도입한 모델을 제안 - 성능 평가: 정밀도-75.55%, F1-score-79.28%
S. Gopali et al.[35]	TCN	Suicidal, Prodigal, Greedy	- 최초로 TCN을 사용한 취약점 탐지 방법을 제안 - 성능 평가: 정밀도-95.95%, F1-score-94.83%
Z. Wu et al.[36]	Many	RE, AE, FR, TD, UE	- ResNet, LSTM, KNN 등 다양한 모델의 성능을 분석 - 성능 평가: ResNets, Macro F1-score-82%
W2V-SA[37]	Hybrid Model	RE, AE, TD, UE, Assert Violation	- 서로 다른 뉴럴 네트워크 모델과 워드 임베딩 방식을 결합한 모델 제안 - 성능 평가: 정밀도-94.85%, F1-score-94%
J. Zhu et al.[38]	LSTM	RE, AC, TD, UE 외 3 가지 취약점	- 오피코드 시퀀스를 활용한 LSTM 기반 모델을 개발 - 시퀀스 길이와 취약성 유형을 조정한 데이터셋을 생성 - Missing the Transfer Event와 같은 취약점을 추가로 탐지 - 성능 평가: 정밀도-80%, F1-score-79.74%
L. Colin et al.[40]	MLP	UE, FR, TD, AC, AE	- 데이터 처리와 관련된 버그 삽입, 전처리 알고리즘 개발 - MLP와 LSTM을 기반으로 하는 다중 입력 모델과 프레임워크를 설계 - 성능 평가를 위한 MythSlith라는 새로운 도구 개발 - 성능 평가: 정밀도-91.28%, F1-score-91.27%
Y. Boxin[42]	Many	RE, UE, TD, AC, AE, Lack of Standard Events	- 동적 실행 중에 발생하는 오피코드 시퀀스를 통한 탐지 방식을 제안 - N-gram 모델과 가중치 케널링 메커니즘을 결합 - 성능 평가: SVM, F1-score-92.55%
X. Feng et al.[43]	XGBoost	RE, TD, AE, FR, Callstack Depth	- SHAP을 사용해 특징이 모델에 미치는 영향을 해석 - 해석 결과를 활용한 적응형 취약점 탐지 방식 제안 - 성능 평가: F1-score-93%

서 주요 패턴이나 특징을 찾아내고, 셀프 어텐션 메커니즘을 활용해 더 중요한 정보에 집중한다. 저자들은 이더리움 공식 페이지에서 컨트랙트를 수집하여 데이터셋을 구성하고, 여러 취약점(RE, AE, AV)을 취약점을 탐지했다.

J. Zhu et al.[38]의 저자들은 오피코드 시퀀스를 기반으로 취약점을 탐지하는 LSTM 기반 모델을 제안했다. 이를 통해 소스 코드 접근이 어려운 배포된 컨트랙트의 경우에도 오피코드를 통한 취약점 탐지가 가능하도록 설계했다. 저자들은 시퀀스 길이에 집중하여 길이를 1000, 2000, 3000으로 조정하며, 정밀도의 변화를 분석했다. 성능 평가를 위해 이더리움 메인넷에서 오피코드 시퀀스를 수집하고, SODA[39] 플러그인을 통해 라벨링하여 데이터셋을 구축했다. 이를 통해 주요 취약점 이외에 MTE(Missing the Transfer Event), SCB(Strict Check for Balance)를 포함한 7가지 취약점을 탐지했다. MTE는 전

송 후 기록을 관리하는 이벤트를 남기지 않는 경우, SCB는 이더를 전송하기 전 잔액에 대한 검증이 엄격하게 수행되지 않을 경우 발생 가능하다.

L. S. H. Colin et al.[40]의 저자들은 MLP을 포함한 여러 모델(SVM, RF, XCG)을 활용한 취약점 탐지를 수행했다. 오피코드와 CFG를 활용해 실행 패턴과 흐름 정보를 학습하도록 설계했다. 저자들은 Smart Contract Sanctuary[41]에서 컨트랙트를 수집했다. 수집한 데이터에 전처리 방법과 버그 삽입 알고리즘을 적용해 균형 잡힌 데이터셋을 구성했다. 또한 고전적인 탐지 도구(Mythril, Slither)를 결합한 MythSlith를 개발하고, 이를 성능 평가 지표로 활용했다.

Y. Boxin[42]의 저자들은 스마트 컨트랙트의 동적 실행 중 발생하는 오피코드 시퀀스를 분석하여 취약점을 탐지하는 방식을 제안했다. 오피코드 시퀀스를 N-gram을 통해 분석하여, 순차적 특징

을 효과적으로 추출했다. 또한 TF-IDF와 유사한 방식으로 가중치 페널티 메커니즘을 적용해 코드별 중요도를 설정했다. 저자들은 컨트랙트의 실행 상태를 모니터링하여 시퀀스 데이터를 수집하고, 데이터셋을 구성했다. 이를 통해 여러 주요 취약점(RE, UE, TD 등)과 LSE(Lack of Standard Events)를 탐지하고, 여러 모델(SVM, KNN, LR)의 성능을 비교했다. LSE는 주요 상태 변경을 기록하지 않거나 표준 이벤트를 따르지 않을 경우, 투명성, 신뢰성 저하를 가져올 수 있다.

X. Feng et al.[43]의 저자들은 기존의 탐지 연구들과 달리 설명 가능한 스마트 컨트랙트 취약점 탐지 모델을 위해 SHAP(Shapley Additive ExPlanations) 값을 기반으로 한 방식을 제안했다. SHAP 값을 활용해 중요한 특징만을 선택하여 학습에 활용했다. 이러한 적응형 최적화를 통해 모델의 해석 가능성을 높이고, 효율적인 학습을 가능하게 했다. 저자들은 이더리움 네트워크에서 컨트랙트를 수집해 데이터셋을 구성하고 여러 취약점(RE, TD, CD 등)을 탐지했다.

4.3 소스 코드 기반 연구

소스 코드는 사람이 이해할 수 있는 고수준 프로그래밍 언어로 작성된 코드로, 주로 솔리디티와 같은 언어를 통해 작성된다. 소스 코드는 스마트 컨트랙트의 문맥과 의미 정보를 모두 포함하고 있어, 이를 활용한 연구가 활발히 이루어지고 있다. 표 3은 표 1, 2와 같은 방식으로 연구를 정리하였으며, 각 연구의 주요 공헌과 탐지 가능한 취약점 목록을 명시하였다.

Eth2Vec[44]의 저자들은 NLP(Natural Language Processing)를 활용한 기법을 제안했다. EVM Extractor 모듈을 개발해 소스 코드로부터 바이트코드의 구문적, 의미적 정보를 JSON 형태로 변환했다. Eth2Vec은 이를 PV-DM 모델의 입력으로 활용하여, 바이트코드를 구조적으로 분석하고 코드의 의미론적 유사성을 평가한다. 유사성 학습을 통해 소스 코드가 변경되더라도 취약점을 탐지할 수 있다. 저자들은 Etherscan에서 수집한 스마트 컨트랙트를 통해 데이터셋을 구성하고 취약점을 탐지했다. ERC-20 Transfer는 LSE와 유사한 취약점으로, 여러 이벤트 중 transfer 이벤트에 대한 처리를 준수하지 않은 경우이다.

Z. Liu et al.[45]의 저자들은 GNN을 활용한 새로운 접근 방식을 제안했다. 소스 코드를 그래프 형식으로 변환하고 노드 정규화 단계를 추가하여 중요 노드를 강조했다. 특히 새로운 모델인 TMPN(Temporal Message Propagation Network)을 도입해 스마트 컨트랙트 코드의 복잡한 제어 및 데이터 흐름을 학습하고 탐지를 수행했다. 또한 Etherscan과 VNT Chain[46]에서 컨트랙트를 수집하여 데이터셋을 구성하고, 이를 통해 RE, TD, Infinite Loop(IL) 취약점을 탐지했다. IL은 컨트랙트 코드 내에 반복문이 포함되어 있고 종료 조건이 명시되지 않은 경우로 DoS와 같은 문제를 일으킬 수 있다.

DeeSCVHunter[47]는 DNN 기반 프레임워크이다. 저자들은 취약성 후보 슬라이스(Vulnerability Candidate Slice, VCS)라는 새로운 개념을 도입했다. VCS를 통해 코드의 구문적, 의미적 정보를 더 잘 이해하도록 하여 모델 성능을 향상 시켰다. VCS는 구문 및 제어 흐름을 추적하여 중요한 취약성 정보를 효과적으로 학습할 수 있다. 또한 프레임워크 및 사용된 데이터셋을 공개하여, 다른 연구자들이 취약성 탐지 시스템을 구축하고 평가할 수 있도록 했다.

Peculiar[48]는 사전 학습(pre-training)된 GraphCodeBERT를 기반으로 하는 프레임워크이다. Peculiar는 기존의 방식에서 사용하는 복잡한 그래프 구조 대신 중요한 데이터 흐름(crucial data flow)만을 고려하도록 설계됐다. 저자들은 그래프 마스크 어텐션(graph masked attention) 기법을 도입하여 토큰 간 관계를 모델링하고 코드 내 변수들 간의 의존 관계를 효과적으로 학습하도록 설계했다. 또한 SmartBugs Dataset[49]을 활용해 성능을 평가했다. 평가 결과 사전 학습된 모델을 통해 높은 일반화 성능을 보였다.

CodeNet[50]은 CNN을 활용한 탐지 기법을 제안했다. 저자들은 기존 CNN 아키텍처와 달리, 스트라이드를 사용하지 않는 방식을 통해 스마트 컨트랙트 코드의 문맥과 의미를 유지하며 취약점을 탐지하도록 설계했다. 또한 분류 마지막 단계에서 GAP(Global Average Pooling)이 아닌 GMP(Global Max Pooling)을 활용해 이미지의 중요한 특징이 보존되도록 했다. 스트라이드를 사용하지 않아 발생하는 메모리 문제를 Depthwise Separable Convolution을 적

표 3. 소스 코드 기반 스마트 컨트랙트 취약점 탐지 연구

Table 3. Research on Sourcecode-Based Smart Contract Vulnerability Detection

Research	Model	Vulnerabilities	Contribution
Eth2Vec[44]	NLP	RE, AE, TD, DoS, AC, ERC-20 Transfer	- 함수의 문맥을 학습, 코드의 변화에도 유연한 탐지를 제공 - 구문 정보를 신경망에 입력하는 모듈 개발 - 성능 평가: 정밀도-77%, F1-score-57.5%
Z. Liu et al.[45]	GNN	RE, TD, Infinite Loop	- 코드를 그래프 형식으로 변환, GNN을 통해 취약점을 탐지하는 방법을 제안 - 성능 평가: F1-score-85.43%
DeeSCVHunter[47]	DNN	RE, TD	- 최초의 딥러닝 모델 기반 탐지 프레임워크 - 취약성 후보 슬라이스 개념을 도입, 학습과정에서 구문 및 의미적 정보 이해능력 향상 - 성능 평가: Bi-LSTM, F1-score-86.87%
Peculiar[48]	LLM	RE	- 데이터 흐름 정보에 기반하여 컨트랙트를 표현하는 새로운 접근 방식을 제안 - 탐지에 사전 학습 기법을 활용한 최초의 접근 - 성능 평가: F1-score-92.10%
CodeNet[50]	CNN	RE, UE, TD, AC	- 기존 CNN과 달리 Stride 없는 CNN 설계 - 데이터 전처리 과정에서 코드의 지역성을 유지 - 성능 평가: F1-score-98.75%
SPCBIG-EC[51]	Hybrid Model	RE, TD, AE, Infinite Loop, Callstack Depth	- CNN과 RNN을 결합한 모델 SPCBIG-EC 제안 - CNN 구조 개선한 SPCNN를 제안, 코드 시퀀스 정보를 유지하면서 특징을 추출 - 성능 평가: 정밀도-89.09%, F1-score-90.13%
xFuzz[52]	Ensemble learning	RE, AC, Delegatecall	- 세 가지 크로스-컨트랙트 취약점을 체계적으로 정의 - 탐색 공간을 줄이기 위해 ML 모델을 사용 - 성능 평가: 정밀도-95.5%
Gvd-net[53]	Graph Embedding	AE, AC, Asset Frozen	- 변수와 함수 호출 관계를 바탕으로 비유클리드 그래프를 생성, 복잡한 관계를 반영 - Node2Vec 알고리즘을 사용해 노드 간 관계를 벡터로 임베딩 - 성능 평가: 정확도-90.5%
PSCVFinder[54]	LLM	RE, TD	- 최초로 Prompt-Tuning을 활용한 모델 제안 - CSCV라는 코드 슬라이싱 기법을 제안, 정보를 유지하면서 컨트랙트 코드를 간소화 - 성능 평가:F1-score-94%
VulDet[56]	GAT	RE, TD	- 스마트 컨트랙트 그래프와 도메인 지식을 결합 - GAT를 통해 특정 노드에 가중치 부여, 탐지 정확도를 향상 - 성능 평가: 정밀도-74.82%, F1-score-78.69%
J. Li et al.[58]	Hybrid Model	RE, TD, AE, AC	- 워드 임베딩 방법과 딥러닝 기술을 결합한 방식 제안 - 실행 중 중요한 정보를 수집하여 동적 특징을 추출, 정적 특징과 결합하여 탐지에 활용 - 성능 평가: 정밀도-83.61%, F1-score-84.98%
GPTScan[60]	LLM	FR, AC 외 8가지	- GPT와 정적 분석을 결합한 탐지 도구 제안 - Slippage, Risky First Deposit 등 다양한 취약점을 탐지 - 각 취약점 유형을 시나리오와 속성으로 세분화하여 정의, 탐지 성능을 향상 - 성능 평가: F1-score-67.8%
LLM4Fuzz[63]	LLM	RE, FR, Logical Error, Price Manipulation, Invariant Violation	- LLM 활용한 퍼징 우선순위 스케줄링 도입 - 중요한 코드 영역을 자동으로 식별 및 테스트 - 성능 평가: ITYFuzz보다 넓은 커버리지를 보임
Y. Chen et al.[64]	Transformer	RE, TD, AE	- 대조 학습을 적용한 새로운 방식을 제안 - 상관 관계 레이블을 생성하여 훈련에 사용 - 성능 평가: F1-score-94.52%
SCVHunter[65]	GNN	RE, FR, TD, DoS	- SG와 GNN을 활용한 확장 가능한 프레임워크 - 사용자가 CSG의 핵심 노드를 지정 가능 - 성능 평가: F1-score-85.65%
VulnHunt-GPT[67]	LLM	RE, AC, AE, UE, DoS, BR, FR, TD, SA	- GPT 모델을 활용해 자연어 처리 방식으로 분석하는 접근 방식 도입 - GPT 모델을 사용해 높은 확장성을 제공 - 성능 평가: 유의미한 탐지 수, 실행 시간을 보임

용하여 해결했다. 성능 평가를 위해 SmartBugs 데이터셋을 활용했다.

SPCBIG-EC[51]의 저자들은 새로운 하이브리드 모델인 SPCBIG-EC(Serial-Parallel Convolutional Bidirectional Gated Recurrent Network Model incorporating Ensemble Classifiers)를 제안했다. 이 모델은 CNN과 RNN을 결합한 구조로, 다중 변수가 결합된 특징을 효과적으로 추출할 수 있다. 또한 기존 CNN 구조를 개선한 SPCNN (Serial-Parallel CNN)을 제안하여 코드 시퀀스의 위치 정보와 구조적 정보를 효과적으로 유지하면서 특징 추출이 가능하도록 했다. 저자들은 이더리움 네트워크에서 컨트랙트를 수집하여 데이터셋을 구성하고, 여러 취약점(RE, TD, IL, CD)을 탐지했다.

xFuzz[52]는 기존 도구들이 거의 다루지 않는 크로스-컨트랙트 취약점을 탐지하기 위한 도구이다. 세 개 이상의 컨트랙트가 상호작용할 때 발생하는 취약점을 찾아내는 데 특화되어 있다. xFuzz는 머신 러닝 기법을 통해 효율적으로 경로를 필터링하고 우선순위를 부여한다. 잠재적 취약점이 있는 경로를 우선으로 테스트하여 취약점을 효과적으로 탐지할 수 있도록 설계했다. 저자들은 다양한 소스(Etherscan, SmartBugs 등)를 활용해 데이터셋을 구성하고, 여러 취약점(RE, AC, DC)을 탐지했다.

Gvd-net[53]은 스마트 컨트랙트 취약점 탐지에 특화된 그래프 임베딩 기반 모델이다. 기존의 선형 구조 대신 스마트 컨트랙트 코드의 변수 및 함수 호출 관계를 기반으로 비유클리드(non-euclidean) 그래프를 구축했다. 이를 통해 코드와 의미구조 간의 복잡한 관계를 더 잘 표현하고 탐지 성능을 향상 시켰다. node2vec을 사용해 노드로부터 벡터를 생성하고, 유사도 분석을 통해 여러 취약점을 탐지한다. Asset Frozen은 EF과와 같은 취약점으로 이더를 수신할 수 있지만, 외부로 송금할 수 없는 구조로 작성된 경우이다. 저자들은 SmartBugs 데이터셋을 활용해 모델의 성능을 평가했다.

PSCVFinder[54]는 스마트 컨트랙트 취약점 탐지에 프롬프트 튜닝(prompt tuning)을 처음으로 적용한 모델이다. 저자들은 사전 학습된 LLM 모델인 CodeT5를 사용하여 취약점을 탐지했다. CodeT5는 코드에서 중요한 식별자와 토큰

을 이해하는 데 강점을 가진 모델이다. 저자들은 성능 향상을 위해 스마트 컨트랙트 코드 슬라이싱 기법을 제안했다. 취약점 탐지와 관련 없는 불필요한 정보를 제거해 모델 성능을 개선했다. 또한 SmartBugs와 다른 연구[55]에서 사용한 데이터셋을 활용해 성능 평가를 진행했다.

VulDet[56]은 GAN(Graph Attention Network)를 사용한 새로운 모델이다. 스마트 컨트랙트의 소스 코드를 그래프로 변환하고, 모든 변수와 함수 호출을 노드로 처리한다. 저자들은 오픈 소스 데이터셋[57]을 사용해 성능을 평가했다. 그 결과 GAN을 통해 취약점과 밀접하게 연관된 노드에 추가적인 중요도를 부여해, 기존 딥러닝 기반 방식보다 우수한 성능을 보였다.

J. Li et al.[58]의 저자들 스마트 컨트랙트 정적 및 동적 특징을 결합하는 특징 결합(feature fusion) 기반 탐지 방식을 제안했다. 스마트 컨트랙트 오피코드로부터 CFG를 생성하고, 이를 GCN(Graph Convolutional Network)을 통해 학습 시켰다. GCN을 통해 스마트 컨트랙트의 구조적 관계를 학습하고, Bi-LSTM을 통해 시간적 순서를 반영하여 취약점을 예측하는 구조를 설계했다. 저자들은 SmartBugs와 SolidFI[59] 데이터셋을 활용했다. 다양한 모델(RNN, GRU, LSTM)과 비교한 결과, 저자들이 사용한 Bi-LSTM이 가장 높은 성능을 보였다.

GPTScan[60]은 GPT와 정적 분석을 결합한 탐지 도구이다. GTPScan은 취약점 유형을 구체적인 시나리오와 속성으로 분할하고, GPT를 활용해 후보 취약점을 선정한다. 저자들은 GPT 모델의 응답이 불확실성을 띠지 않도록, 동일 질문을 여러 번 묻고 가장 많이 일치한 응답을 채택하는 방식을 사용했다. GTPScan은 최종적으로 분류한 취약점을 정적 분석을 통해 검증한다. 저자들은 Web3Bugs[61], DefiHacks[62] 데이터셋을 활용했다. 이를 통해 Slippage, WCO(Wrong Checkpoint Order)와 같은 10가지 취약점을 탐지했다. Slippage는 자산 스왑(swap) 또는 유동성(liquidity) 추가 시 가격 변동으로 인해 예기치 못한 결과가 발생할 수 있다. WCO는 상태 업데이트와 실제 계산 순서가 올바르지 않을 경우 동기화 문제가 발생할 수 있다.

LLM4Fuzz[63]는 퍼징 스케줄링에 머신 러닝 기법을 도입한 도구이다. LLM4Fuzz는 LLM을 활용한 우선순위 스케줄링을 도입하여,

LLM이 생성한 지표를 바탕으로 테스트를 수행한다. 스케줄링을 통해 복잡도가 높은 함수나 코드 영역을 중점적으로 테스트하도록 설계됐다. 이러한 방식으로 기존 퍼징 도구보다 더 효율적이고 넓은 커버리지를 달성했다. LLM4FUZZ는 탐지 가능한 취약점 목록을 구체적으로 나열하지 않았지만, 논문에서 제시한 사례, 설계 원리를 통해 LE, PM(Price Manipulation), IV(Invariant Violation)와 같은 여러 취약점을 탐지한다. PM은 AMM(Automated Market Maker)에서 유동성 풀(liquidity pool)을 통해 가격을 조작할 수 있다. IV는 컨트랙트 내에서 불변 값이 변경 가능하도록 설정된 경우 토큰의 총량이 변경되는 등 여러 문제가 발생 가능하다.

Y. Chen et al.[64] 저자들은 대조 학습(Contrastive Learning)을 적용한 새로운 모델을 제안했다. 기존의 취약점 탐지 연구에서 잘 고려되지 않던 컨트랙트 간의 상관 관계를 모델링하여 활용했다. 이를 통해 컨트랙트 간의 유사성과 다른 유형 간의 차이를 파악하여 취약성 탐지 성능을 향상 시켰다. 대조 학습 단계에서 컨텍스트 증강(contextual augmentation)을 위해 MLM(Masked Language Model)을 사용했다. MLM은 입력 데이터의 일부 토큰을 무작위로 마스킹하고 주변 문맥을 바탕으로 예측하게 하여 모델이 코드의 구조적, 의미적 특징을 더 잘 이해하도록 한다. 성능 평가를 위해 이전의 연구[23]에서 사용된 데이터셋을 활용했다.

SCVHunter[65]는 스마트 컨트랙트 소스 코드를 CSG(Contract Semantic Graph)로 변환하고, GNN을 활용하여 취약점을 탐지하도록 설계됐다. 이 과정에서 사용자는 간단한 규칙 입력만으로 취약점과 관련된 핵심 노드를 지정할 수 있다. CSG는 핵심 노드, 호출 노드 등으로 구성되어 스마트 컨트랙트의 구조적 의미를 효과적으로 표현한다. SCVHunter는 노드 레벨과 경로 레벨에서 어텐션 메커니즘을 적용하여, 노드 이웃의 중요도를 학습하고 취약점 관련 노드에 더 집중하도록 한다. 저자들은 오픈소스 데이터셋[66]을 활용해 성능을 평가했다.

VulnHunt-GPT[67]의 저자들은 GPT 모델을 활용한 방식을 제안했다. VulnHunt-GPT는 기존의 고전적인 도구와 달리, LLM을 활용하여 취약점 분석을 수행한다. 프롬프트 엔지니어링을 활용하여 GPT 모델의 응답을 조정했다. 주요 취

약점 정보를 사전에 제공해 정확도를 높였다. VulnHunt-GPT는 Python 스크립트를 통해 스마트 컨트랙트 코드를 전처리하고 OpenAI API를 통해 통신한다. 성능 평가 결과 기존 탐지 도구보다 빠른 실행 시간을 보였다. 저자들은 성능 평가를 위해 SmartBugs 데이터셋을 활용했다.

V. 결 론

본 논문은 머신 러닝을 활용한 스마트 컨트랙트 취약점 탐지 연구들에 대해 조사했으며, 스마트 컨트랙트 내에 존재 가능한 대표적인 취약점 9가지에 대해 설명했다. 최근 다양한 분야에서 뛰어난 성능을 보인 머신 러닝 기법을 스마트 컨트랙트 취약점 탐지에 적용한 연구가 진행되고 있다. 본 논문에선 최신 동향 분석을 위해 2021년 이후 발표된 연구를 선정하였다. 머신 러닝을 활용한 연구에서 주로 사용되는 입력 데이터 유형인 바이트코드, 오피코드, 소스 코드를 기준으로 각 연구들을 분류하고, 각 연구의 주요 기여와 특징을 제시하였다.

바이트코드 기반 연구는 소스 코드가 비공개된 경우에도 취약점 탐지가 가능하다는 강점을 가진다. 오피코드 기반 연구는 명령어를 여러 형태(그래프, 이미지)로 변환하거나 명령어 시퀀스를 활용하는 등 다양한 방식으로 연구가 이루어진다. 소스 코드 기반 연구는 자연어 처리와 유사한 방식으로 소스 코드를 분석하여 취약점을 탐지하며, 특히 LLM을 활용한 연구가 활발히 진행되고 있다. 본 논문은 이러한 입력 데이터별 연구 동향과 주요 특징을 조사하여 제시하였다.

향후 연구로는 머신 러닝 기법을 활용한 스마트 컨트랙트 취약점 탐지 모델을 발전시키고자 한다. 스마트 컨트랙트 내에서 발생할 수 있는 새로운 유형의 취약점이 지속적으로 등장하고 있어, 해결하기 위한 탐지 모델 연구가 필수적이다. 기존 연구가 주로 알려진 취약점 탐지에 집중된 한계가 있는 만큼, 새로운 유형의 취약점을 효율적으로 탐지할 수 있는 모델 개발을 목표로 연구를 진행할 계획이다.

또한 현재 연구들에서 사용되는 데이터셋들이 대부분 표준화되지 않아 비교 및 재현성 확보에 어려움이 있다. 이러한 문제를 해결하기 위해 스마트 컨트랙트 데이터셋의 표준화 방안을 함께 연구할 계획이다. 표준화된 데이터셋은 다양한 연구 간 성

능 비교를 용이하게 하고, 연구 결과의 신뢰성을 높이며, 스마트 컨트랙트 취약점 탐지 기술의 발전을 촉진할 수 있을 것으로 기대된다.

References

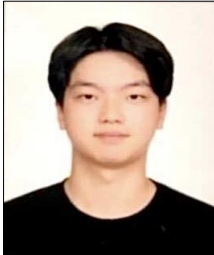
- [1] S. Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System(2009), Retrieved Nov. 14, 2024, from <https://bitcoin.org/bitcoin.pdf>.
- [2] The Parity Wallet Hack Explained, Retrieved Nov. 14, 2024, from <https://blog.openzeppelin.com/on-the-parity-wallet-multisig-hack-405a8c12e8f7>
- [3] Top 17 Smart Contract Hacks in 2021-2022, Retrieved Nov. 14, 2024, from <https://4irelabs.com/articles/top-17-smart-contract-hacks/>
- [4] M. I. Mehar et al., "Understanding a revolutionary and flawed grand experiment in blockchain: the DAO attack," Journal of Cases on Information Technology, Vol. 21, No. 1, pp. 19 - 32, Feb. 2019.
- [5] Cream Finance Exploit, Retrieved Nov. 14, 2024, from <https://www.immunebytes.com/blog/cream-finance-exploit-oct-27-2021-detailed-analysis>.
- [6] R. Kiani and V. S. Sheng, "Ethereum smart contract vulnerability detection and machine learning-driven solutions: a systematic literature review," Electronics, Vol. 13, No. 12, pp. 1 - 23, Jun. 2024.
- [7] C. D. Baets et al., "Vulnerability detection in smart contracts: a comprehensive survey," arXiv preprint arXiv:2407.07922, pp. 1 - 46, Jul. 2024.
- [8] Z. A. Khan and A. S. Namin, "A Survey of Vulnerability Detection Techniques by Smart Contract Tools," IEEE Access, Vol. 12, pp. 70870 - 70910, May, 2024
- [9] T. Jiao et al., "A Survey of Ethereum Smart Contract Security: Attacks and Detection," Distributed Ledger Technologies: Research and Practice, Vol. 3, No. 3, pp. 1 - 28, Sep. 2024.
- [10] N. Szabo, "Formalizing and securing relationships on public networks," First Monday, Vol. 2, No. 9, Sep. 1997.
- [11] Solidity, Retrieved Nov. 14, 2024, from <https://docs.soliditylang.org/en/lates>
- [12] V. Buterin, Vyper Documentation(2018), Retrieved Nov. 14, 2024, from <https://readthedocs.org/projects/vyper/downloads/pdf/v0.1.0-beta.8/>
- [13] DASP - TOP 10, Retrieved Nov. 14, 2024, from <http://dasp.co/>
- [14] O. Lutz et al., "Escort: ethereum smart contracts vulnerability detection using deep neural network and transfer learning," arXiv preprint arXiv:2103.12607, pp. 1 - 17, Mar. 2021.
- [15] F. Mi et al., "VSCL: automating vulnerability detection in smart contracts with deep learning," in Proc. of the 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Sydney, Australia, pp. 1 - 9, May 2021.
- [16] J. Huang et al., "Hunting vulnerable smart contracts via graph embedding based bytecode matching," IEEE Transactions on Information Forensics and Security, Vol. 16, pp. 2144 - 2156, Jan. 2021.
- [17] Mist, Retrieved Dec. 9, 2024, from <https://github.com/ethereum/mist>
- [18] Etherscan, Retrieved Dec. 9, 2024, from <https://etherscan.io/>
- [19] Y. Sun and G. Lize, "Attention-based machine learning model for smart contract vulnerability detection," Journal of physics: conference series, Vol. 1820. No. 1, pp. 1 - 6, 2021.
- [20] Z. Liao et al., "SmartDagger: a bytecode-based static analysis approach for detecting cross-contract vulnerability," in Proc. of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA), South Korea, pp. 752 - 764, Jul. 2022.
- [21] S. J. Qin et al., "Smart contract vulnerability detection based on critical combination path and deep learning," in Proc. of the 2022 12th International Conference on Communication and Network Security, Beijing, China, pp. 219 - 226, Dec. 2022.
- [22] Awesome-Buggy-Erc20-Tokens, Retrieved Dec. 9, 2024, from <https://github.com/sec-bit/awesome-buggy-erc20-tokens>
- [23] P. Qian et al., "Cross-modality mutual learning for enhancing smart contract vulnerability detection on bytecode," in Proc. of the ACM Web Conference 2023, Austin, USA, pp. 2220 - 2229, Apr. 2023.
- [24] Z. Li et al., "VulHunter: Hunting Vulnerable Smart Contracts at EVM bytecode-level via Multiple Instance Learning," IEEE Transactions on Software Engineering, Vol. 49, No. 11, pp. 4886 - 4916, Nov. 2023.
- [25] W. Li et al., "COBRA: Interaction-Aware Bytecode-L

- level Vulnerability Detector for Smart Contracts,” in Proc. of the 39th IEEE/ACM International Conference on Automated Software Engineering, Sacramento, USA, pp. 1358 - 1369, Oct. 2024.
- [26] P. Zheng et al., “Xblock-eth: Extracting and exploring blockchain data from ethereum,” IEEE Open Journal of the Computer Society, Vol. 1, pp. 95 - 106, May 2020.
- [27] L. Prifti et al., “Smart Contract Vulnerability Detection Using Deep Learning Algorithms on EVM bytecode,” in Proc. of the 2024 13th Mediterranean Conference on Embedded Computing (MECO), Budva, Montenegro, pp. 1 - 7, Jun. 2024.
- [28] M. Rossini et al., “Smart contracts vulnerability classification through deep learning,” in Proc. of the 20th ACM Conference on Embedded Networked Sensor Systems, Boston, USA, pp. 1229 - 1230, Nov. 2022.
- [29] S. Goswami et al., “TokenCheck: towards deep learning based security vulnerability detection in ERC-20 tokens,” in Proc. of the 2021 IEEE Region 10 Symposium (TENSYP), Jeju, Korea, pp. 1 - 8, Aug. 2021.
- [30] Ethereum Public Dataset, Retrieved Dec. 9, 2024, from https://bigquery.cloud.google.com/dataset/bigquery-publicdata:ethereum_blockchain
- [31] J. Su et al., “Effectively generating vulnerable transaction sequences in smart contracts with reinforcement learning-guided fuzzing,” in Proc. of the 37th IEEE/ACM International Conference on Automated Software Engineering, Rochester, USA, pp. 1 - 12, Oct. 2022.
- [32] XBlock, Retrieved Dec. 9, 2024, from <https://xblock.pro/#/>
- [33] VeriSmart, Retrieved Dec. 9, 2024, from <https://github.com/kupl/VeriSmart-benchmarks>
- [34] J. Huang et al., “Smart contract vulnerability detection model based on multi-task learning,” Sensors, Vol. 22, No. 5, pp. 1 - 24, Feb. 2022.
- [35] S. Gopali et al., “Vulnerability detection in smart contracts using deep learning,” in Proc. of the 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC), Los Alamitos, USA, pp. 1249 - 1255, Jun. 2022.
- [36] Z. Wu et al., “Detecting Vulnerabilities in Ethereum Smart Contracts with Deep Learning,” in Proc. of the 2022 4th International Conference on Data Intelligence and Security (ICDIS), Shenzhen, China, pp. 55 - 60, Aug. 2022.
- [37] Z. Xu et al., W2V-SA: A Deep Neural Network-based Approach to Smart Contract Vulnerability Detection(2023), Retrieved Nov. 14, 2024, from <https://ceur-ws.org/Vol-3387/paper19.pdf>
- [38] J. Zhu et al., “Opcode Sequences-Based Smart Contract Vulnerabilities Detection Using Deep Learning,” in Proc. of the 2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Exeter, United Kingdom, pp. 284 - 291, Nov. 2023.
- [39] T. Chen et al., “SODA: A Generic Online Detection Framework for Smart Contracts,” in Proc. of Network and Distributed Systems Security (NDSS), San Diego, USA, pp. 1 - 17, Feb. 2020.
- [40] L. S. H. Colin et al., “An Integrated Smart Contract Vulnerability Detection Tool Using Multi-layer Perceptron on Real-time Solidity Smart Contracts,” IEEE Access, Vol. 12, pp. 23549 - 23567, Feb. 2024.
- [41] Smart Contract Sanctuary, Retrieved Dec. 9, 2024, from <https://github.com/tintinweb/smart-contract-sanctuary>
- [42] Y. Boxin, “Research on Dynamic Detection of Vulnerabilities in Smart Contracts Based on Machine Learning,” in Proc. of the 2024 IEEE 3rd International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA), Changchun, China, pp. 219 - 223, Feb. 2024.
- [43] X. Feng et al., “An Interpretable Model for Large-Scale Smart Contract Vulnerability Detection,” Blockchain: Research and Applications, Vol. 5, No. 3, pp. 1 - 10, Sep. 2024.
- [44] N. Ashizawa et al., “Eth2Vec: Learning Contract-Wide Code Representations for Vulnerability Detection on Ethereum Smart Contracts,” in Proc. of the 3rd ACM international symposium on blockchain and secure critical infrastructure, Hong Kong, pp. 47 - 59, Jun. 2021.
- [45] Z. Liu et al., “Combining graph neural networks with expert knowledge for smart contract vulnerability detection,” IEEE Transactions on Knowledge and Data Engineering, Vol. 35, No. 2, pp. 1296 - 1310, Jul. 2021.

- [46] VNT Chain, Retrieved Dec. 9, 2024, from <https://github.com/vntchain/go-vnt>
- [47] X. Yu et al., "Deescvhunter: A deep learning-based framework for smart contract vulnerability detection," in Proc. of the 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, pp. 1 - 8, Jul. 2021.
- [48] H. Wu et al., "Peculiar: Smart contract vulnerability detection based on crucial data flow graph and pre-training techniques," in Proc. of the 2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE), Wuhan, China, pp. 378 - 389, Oct. 2021.
- [49] J. F. Ferreira et al., "Smartbugs: A framework to analyze solidity smart contract," in Proc. of the 35th IEEE/ACM international conference on automated software engineering, New York, pp. 1349 - 1352, Dec. 2020.
- [50] S. J. Hwang et al., "CodeNet: Code-targeted convolutional neural network architecture for smart contract vulnerability detection," IEEE Access, Vol. 10, pp. 32595 - 32607, Mar. 2022.
- [51] L. Zhang et al., "SPCBIG-EC: a robust serial hybrid model for smart contract vulnerability detection," Sensors, Vol. 22, No. 12, pp. 1 - 32, Jun. 2022.
- [52] Y. Xue et al., "xfuzz: Machine learning guided cross-contract fuzzing," IEEE Transactions on Dependable and Secure Computing, Vol. 21, No. 2, pp. 515 - 529, Jun. 2022.
- [53] Z. Wang et al., "Gvd-net: Graph embedding-based machine learning model for smart contract vulnerability detection," in Proc. of the 2022 International Conference on Algorithms, Data Mining, and Information Technology (ADMINT), Xi'an, China, pp. 99 - 103, Sep. 2022.
- [54] L. Yu et al., "Pscvfinder: a prompt-tuning based framework for smart contract vulnerability detection," in Proc. of the 2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE), Florence, Italy, pp. 556 - 567, Oct. 2023.
- [55] Z. Liu et al., "Smart contract vulnerability detection: from pure neural network to interpretable graph feature and expert pattern fusion," arXiv preprint arXiv:2106.09282, pp. 1 - 9, Jun. 2021.
- [56] F. Luo and R. Luo, "VulDet: Smart Contract Vulnerability Detection Based on Graph Attention Networks," in Proc. of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, Zibo, China, pp. 88 - 92, Jun. 2021.
- [57] Y. Zhuang et al., "Smart contract vulnerability detection using graph neural networks," in Proc. of the 2023 2nd International Conference on Artificial Intelligence and Blockchain Technology (AIBT), Yokohama, Japan, pp. 3283 - 3290, Jan. 2021.
- [58] J. Li et al., "A Smart Contract Vulnerability Detection Method Based on Multimodal Feature Fusion and Deep Learning," Mathematics, Vol. 11, No. 23, pp. 1 - 23, Nov. 2023.
- [59] C. S. Yashavant et al., "Scrawld: A dataset of real world ethereum smart contracts labelled with vulnerabilities," arXiv preprint arXiv:2202.11409, pp. 1 - 5, Feb. 2022.
- [60] Y. Sun et al., "Gptscan: Detecting logic vulnerabilities in smart contracts by combining gpt with program analysis," in Proc. of the IEEE/ACM 46th International Conference on Software Engineering, Lisbon, Portugal, pp. 1 - 13, Apr. 2024.
- [61] Web3Bugs, Retrieved Dec. 9, 2024, from <https://github.com/ZhangZhaoSJTU/Web3Bugs>
- [62] DeFi Hacks, Retrieved Dec. 9, 2024, from <https://woted-meter-1d8.notion.site/0e85e02c5ed34df3855ea9f3ca40f53b?v=22e5e2c506ef4caeb40b4f78e23517ee>
- [63] C. Shou et al., "Llm4fuzz: Guided fuzzing of smart contracts with large language models," arXiv preprint arXiv:2401.11108, pp. 1 - 16, Jan. 2024.
- [64] Y. Chen et al., "Improving Smart Contract Security with Contrastive Learning-based Vulnerability Detection," in Proc. of the IEEE/ACM 46th International Conference on Software Engineering, Lisbon, Portugal, pp. 1 - 11, Apr. 2024.
- [65] F. Luo et al., "Scvhunter: Smart contract vulnerability detection based on heterogeneous graph attention network," in Proc. of the IEEE/ACM 46th International Conference on Software Engineering, Lisbon, Portugal, pp. 1 - 13, Apr. 2024.
- [66] T. Durieux et al., "Empirical review of automated analysis tools on 47,587 ethereum smart contracts," in Proc. of the the ACM/IEEE 42nd International conference on software engineering, Seoul, Korea, pp. 530 - 541, Jun. 2020.

- [67] B. Boi et al., “VulnHunt-GPT: a Smart Contract vulnerabilities detector based on OpenAI chatGPT,” in Proc. of the 39th ACM/SIGAPP Symposium on Applied Computing, Avila, Spain, pp. 1517 - 1524, Apr. 2024.

이 의 찬 (Ui-Chan Lee)



2019년 3월~현재 : 강원대학교
컴퓨터정보통신공학 학사

<관심분야> 네트워크 관리, 블록체인, 정보보안

방 지 원 (Jiwon Bang)



2018년 2월 : 강원대학교 컴퓨터과학과 졸업
2020년 2월 : 강원대학교 컴퓨터과학과 석사
2020년 3월~현재 : 강원대학교 빅데이터메디컬융합학과 박사

<관심분야> 네트워크 관리, 블록체인, 정보보안

최 미 정 (Mi-Jung Choi)



1998 : 이화여자대학교 공학사
2000 : 포항공과대학교 공학석사
2004 : 포항공과대학교 공학박사
2004 ~ 2005 : 프랑스 INRIA 연구소 박사후 연구원

2005 ~ 2006 : 캐나다 워털루대학 박사후 연구원
2006 ~ 2008 : 포항공과대학 컴퓨터공학과 연구 조교수
2008 ~ 현재 : 강원대학교 컴퓨터학부 컴퓨터과학전공 교수
<관심분야> 네트워크 관리, 블록체인, 정보보안, 악성코드 연쇄킹