| Oral Session IX : Big Data, Smart Energy ICT, Smart Information

좌장 : 신춘성 (전남대)

Smart Contract Auditing을 위한 코드 복잡도 추출

박찬솔, 박보경, 문소영, 김영철
홍익대학교 소프트웨어융합학과
e-mail : c2193102@g.hongik.ac.kr, parkse@cue.ac.kr, whit2@hongik.ac.kr, bob@hongik.ac.kr

Extracting Code Complexity for Auditing A Smart Contract

Chansol Park, Bo Kyung Park, So Young Moon, R. Young Chul Kim
SE Lab, Dept. of Software and Communications Eng, Hongik University

요 약

블록체인 네트워크를 이용해 동작하는 DApp에 대한 관심과 수요가 많아지고 있으며 동시에 Smart Contract에 대한 Audit도 증가하고 있다. 하지만 현재 Audit 작업은 수동으로 진행하거나 혹은 부분적으로 자동화 하여 진행하고 있으며 이는 많은 시간과 비용을 유발한다. 본 논문에서는 기존 연구실의 Audit 작업에 대한 자동화를 위한 소프트웨어 아키텍처 가시화 기법을 제안 하며 이를 통해 Audit 작업 속도 단축과 기존 보다 더 풍부한 관점에서의 분석이 가능할 것이라 기대한다.

키워드: 역공학, 블록체인, 스마트 컨트렉트, 소프트웨어 가시화, 소프트웨어 복잡도

1. 서 론

블록체인을 이용한 Smart Contract를 통해 다양한 서비스를 제공하고 있다. 이때 Smart Contract의 안전성을 보장하기 위한 Audit을 진행한다. 하지만 현재 Audit을 발급하기 위해 수동으로 코드를 분석하고 있다. 본 논문에서는 Audit 자동화를 위해 소프트웨어 아키텍처 가시화 기법을 통한 복잡도 측정 방법을 제시한다. Audit을 하는데 효율성이 증가하고, 시간 및 비용을 감소시킬 수 있다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 관련 연구들을 언급하고, 3장에서는 본 논문의 주제인 Smart Contract Audit을 위한 복잡도 추출 연구에 대해 언급한다. 마지막으로 4장에서 결론 및 추후 연구에 대해 언급한다.

2. 관련 연구

2.1 블록체인과 Smart Contract

블록체인 기술은 P2P 네트워크 간 통신을 통해 중앙 서버 없이 데이터를 관리하는 기술이다. 네트워크의 모든 구성원이 같은 정보를 공유하며 트랜잭션을 전파하여 데이터를 수정하고, 이러한 트랜잭션을 모아 데이터의 변동사항을 기록한다. Bitcoin 암호 화폐를 거래할 수 있는 비트코인 네트워크가 가장 처음으로 구현되었으며 대표적인 1세대 블록체인이다.

탈중앙화 애플리케이션 DApp은 블록체인 네트워크를 이용해 서비스 하는 애플리케이션을 말한다. DApp을 구현할 수 있는 블록체인 네트워크를 2세대 블록체인으로 구분하며, 선발주자인 Ethereum과 그 후 개발 된 EOS, TRON등이 포함된다.

Smart Contract는 DApp의 구성요소 중 하나로서 Ethereum에서 실행되는 소프트웨어를 말하며 Ethereum의 World State에 저장되어 있는 스크립트이다. DApp의 사용자가 공통으로 공유하는 데이터를 Ethereum 상에서 조회, 수정, 저장하는 용도로 사용된다.[2] Ethereum에서 Smart Contract를 구현하는 언어 중 가장 활발하게 사용되고, 유지 관리되는 언어는 Solidity와 Viper가 있다. 그 외에도 중간 언어인 Yul 등을 통해 구현한다. 이러한 언어들은 공통적으로 OPCODE로 컴파일 된다. 본 논문에서는 그 중 가장 커뮤니티 활동이 활발하고 점유율이 높은 Solidity를 사용한다.

Smart Contract의 기능 중에는 토큰을 생성 및 거래하거나, Ether를 거래하는 기능이다. 대체가능한 토큰, 대체불가능한 토큰 그리고 Ether는 현재 가치가 있다고 여겨지는 만큼, 사소한 문제라도 치명적으로 작용될 수 있다. 따라서 DApp 서비스 제공자는 DApp의 사용자로 하여금 서비스를 안심하고 이용할 수 있도록 Audit을 발급받는다.

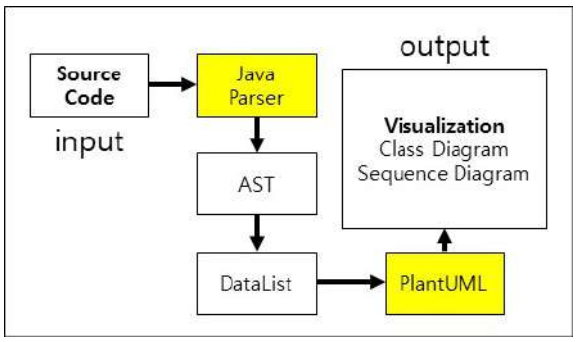
Smart Contract Audit은 Solidity 코드에 대해 수동 혹은 부분적으로 자동화하여 분석을 진행하고 문서화하는 일련의 과정이다. 2022년 현재 다양한 업체에서 Auditing 서비스를 제공하고 있으며, 일관적인 양식과 Audit 보고서 내용에 대한 표준은 없지만 공통적으로 취약점에 대한 분석을 제공하고 몇몇 업체는 여기에 부가적으로 DApp과 관련된 토큰의 가격 혹은 SNS 여론분석 등의 추가적인 분석을 제공하고 있다.

2.2 소프트웨어 아키텍처 가시화[3]

소프트웨어는 비가시적 특성과 소프트웨어가 점차 다양

함 기능을 함유함에 있어 복잡하지고 있기 때문에 품질 관리에 어려움을 겪는다. 이러한 문제점에 대한 해결책으로 우리는 기존에 정적분석을 통한 소프트웨어 가시화 기법을 제시하였다. 개발된 도구를 이용해 설계 및 복잡도에 대한 점수를 도출한다.

다음의 그림은 기존 Java에 대한 소프트웨어 가시화 툴체인 구조이다. 소소코드로부터 Abstract Syntax Tree(AST)를 추출하여 DataList 구조로 저장 한 후 DataList의 정보를 가공하여 Class Diagram 및 Sequence Diagram 형태로 가시화 한다.

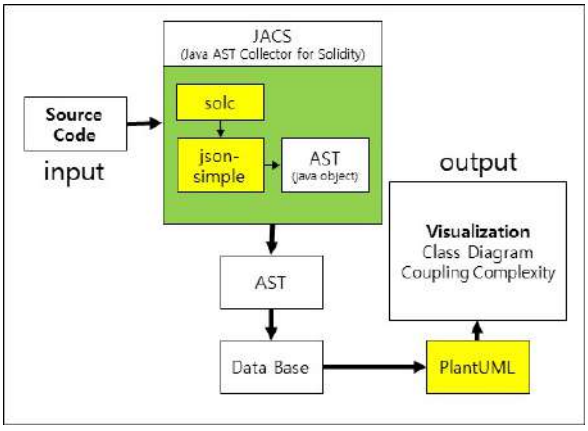


(그림 1) Java에 대한 소프트웨어 아키텍처 가시화 기법

3. Smart Contract Auditing을 위한 코드 복잡도 추출

3.1 Solidity 아키텍처 가시화 도구 설계

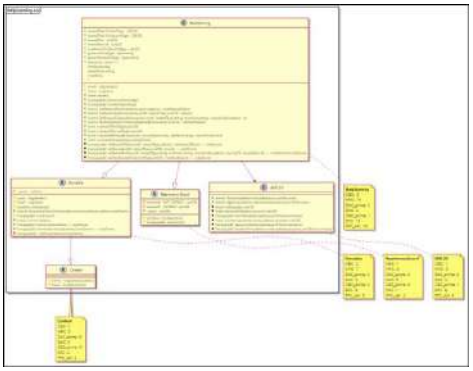
Etehreum은 Solidity를 계약 지향 언어라고 명시한다. 여기서 계약 지향 언어란 함수와 상태변수를 포함하고 있는 계약을 선언 하고, 이러한 계약 간 관계를 통해 스마트 컨트랙트를 구성한다. 이때 계약의 구조는 객체 지향 언어에서의 객체의 구조를 보인다. 따라서 기존의 객체 지향 언어에 대한 소프트웨어 가시화 기법을 기반으로 Solidity에 대한 아키텍처 가시화 도구를 설계하였다.



(그림 2) Solidity에 대한 소프트웨어 가시화 공정

Solidity에 대한 소프트웨어 가시화 공정은 다음과 같다. 우선 Java AST Collector for Solidity(JACS)도구를 이용하여 Java의 객체 구조에 AST 노드들을 저장한다. 이후 AST 구조에서 필요한 정보들을 추출하여 데이터베이스에 저장한다. 이후 가시화 도구에서 데이터베이스를 이용해 필요한 정보를 통해 소프트웨어 복잡도를 측정한 후 설계와 함께 표시한다. 이번 연구에서는 기존 객체 지향 소프트웨어 품질지표 중 Coupling 복잡도를 Solidity에 적용하여 복잡도 측정 및 가시화를 진행하였다.

3.2 Solidity 아키텍처 가시화 도구 구현



(그림 3) Solidity에 대한 소프트웨어 아키텍처 가시화 예시

그림 3는 Solidity 아키텍처 가시화 도구를 샘플 Solidity 코드에 적용한 결과이다. Class Diagram으로 설계를 추출한 후 CBO, RFC, MPC, DAC와 그의 몇몇 변종들[3][4][5]을 분석 후 각 클래스별로 복잡도를 측정하여 Class Diagram에 표기하였다.

4. 결론

본 논문에서는 Smart Contract에 대한 Audit을 위해 기존 객체 지향 프로그램에 대한 설계 추출 및 복잡도 그리고 기존 연구실의 툴체인 프로세스에 대해 적용해 보았다. 추후 연구를 통해 이러한 설계 추출과 복잡도의 Solidity에 대한 유효성을 검증하고자 한다. 이를 통해 Smart Contract에 대한 Audit 과정을 자동화 하는 것을 기대한다.

ACKNOWLEDGMENT

이 논문은 교육부 및 한국연구재단의 4단계 두뇌한국21 사업의 지원(F21YY8102068)과 2022년도 정부(교육부)의 재원으로 한국연구재단의 지원(No. 2021R1I1A305040711, No. 2021R1I1A1A01044060)을 받아 수행된 연구임.

참 고 문 헌

[1] Wood, Gavin, "Ethereum: A secure decentralised generalised transaction ledger.", Ethereum project yellow paper, 2014, pp.1-32.

[2] Jung, Se Jun, et al, "Automatic UML Design Extraction with Software Visualization based on Reverse Engineering.", International journal of advanced smart convergence 10.3, 2021, pp.89-96.

[3] S. R. Chidamber, Shyam R., and C. F. Kemerer, "Towards a metrics suite for object oriented design.", Conference proceedings on Object-oriented programming systems, languages, and applications, 1991, pp.197-211.

[4] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design,", in IEEE Transactions on Software Engineering, vol. 20, no. 6, June 1994, pp. 476-493.

[5] W. Li and S. Henry, “Object-Oriented Metrics that Predict Maintainability,”, Journal of Systems and Software, Vol. 23, No. 2, 1993, pp. 111-122.