하이퍼레저 패브릭 체인코드의 비결정적 실행 탐지 방안

김범석, 신운섭, 황동엽, 김기형
아주대학교

[godbyul, gcloud, bc8c, kkim86]@ajou.ac.kr

Solution for Hyperledger Fabric Chaincode Non-Deterministic Execution Detection

Beomseok Kim, WoonSeop Shin, Dongyeop Hwang, Ki-Hyung Kim
Ajou Univ.

요 약

최근 사회 여러 도메인에서 블록체인 기반의 스마트 컨트랙트를 작성하는 DApp(Decentralized Application) 개발이 활발히 이루어지고 있다. 특히 하이퍼레저 패브릭은 오픈소스 프로젝트로 퍼미션 블록체인 플랫폼으로 각광을 받고 있다. 그러나 DAO 공격처럼 스마트 컨트랙트의 취약점으로 인해 큰 피해가 발생할 수 있다. 하이퍼레저 패브릭의 스마트 컨트랙트인 체인코드의 잠재적 위협 중 하나는 비결정적 실행이다. 비결정적 실행은 체인코드 뿐만 아니라 다른 스마트 컨트랙트에서도 악영향을 끼친다. 체인코드의 비결정적 실행과 관련한 연구가 진행되고 있으나, Go 로 작성된 체인코드의 비결정적 실행 만 탐지할 수 있다. 이 논문은 지역변수 추적을 통해 체인코드의 비결정적 실행을 탐지하는 알고리즘을 제안한다.

Ⅰ. 서 론

최근 블록체인 기반의 암호화폐 열풍으로 인해 많은 사람들이 블록체인 기술에 주목하게 되었다. 특히 블록체인 기반의 스마트 컨트랙트 기술은 금융, 물류, 의료 등의 분야에 적용되어 큰 혁신을 일으키고 있으며, 대표적인 플랫폼으로 하이퍼레저 패브릭과 이더리움이 있다. 스마트 컨트랙트는 예민한 정보, 높은 가치의 자산을 다루므로, 악의적인 공격자에 의해 악용 될 경우 큰 피해가 발생할 수 있다. 지난 2016 년 6 월 DAO 사태는 510 억원의 피해가 발생했다. DAO 는 이더리움 기반으로 개발되었으며, 스마트 컨트랙트의 취약점에 의해 발생했다[1]. 이더리움 뿐만 아니라 하이퍼레저 패브릭에서도 취약점이나 잠재적 위협이 존재 할 수 있으며, 관련된 연구가 진행되고 있다. 하이퍼레저 패브릭의 스마트 컨트랙트(하이퍼레저 패브릭에서는 스마트 컨트랙트를 체인코드로 정의함)는 비결정적 실행으로 인한 잠재적 위협이 존재한다. 이 문제를 해결하기 위해 사전에 체인코드를 분석해 비결정적 실행을 탐지하는 연구가 진행되고 있다[2,3,4]. 그러나 체인코드는 Go, 자바, 자바스크립트 3 가지 프로그래밍 언어로 작성되는데 기존의 연구들은 Go 로 작성된 체인코드만 탐지할 수 있었다. 이 논문에서는 지역변수 추적을 통해 프로그래밍 언어에 상관없이 체인코드의 비결정적 실행을 탐지할 수 있는 방안에 대해 제안한다. 이 논문은 2 장 본론, 3 장 결론으로 구성된다.

Ⅱ. 본론

2.1 체인코드의 비결정적 실행

- 전역변수 : 하이퍼레저 패브릭은 보증정책에 의해 모든 피어들이 체인코드를 실행하지 않고 일부 피어들이 체인코드를 실행할 수 있다. 이런 경우 각 피어들 마다 전역변수의 값이 달라 질 수 있다.
- 난수 생성 : 각각의 피어가 체인코드를 실행하면서 난수를 생성한다면, 각각의 피어들이 같은 난수를 생성한다는 보장이 없다.
- 타임스탬프 : 각각의 피어는 독립적인 환경에서 실행되므로, 동시에 실행된다는 보장이 없다. 그러므로 타임스탬프를 출력하면 모든 피어들이 같은 타임스탬프를 사용한다는 보장이 없다.
- Map 자료구조 : Go 에서는 Map 자료구조를 지원한다. 그러나 Go 의 특성 상 Map 을 반복문으로 출력할 때, 키 값의 순서가 일정하지 않다. 자바의 HashedMap 도 키 값의 순서가 일정하지 않게 출력된다.
- 프로그램의 동시성 : 체인코드를 작성할 수 있는 Go, 자바, 자바스크립트는 쓰레드 사용이 가능하다. 그러나 쓰레드 사용은 경쟁 상태를 유발할 수 있으며, 이는 각 피어 간의 상태가 같음을 보장할 수 없다.
- 구조체 내 변수선언 : 체인코드를 작성하면서 체인코드 인터페이스를 구현하는 구조체를 작성한다. 이 구조체 내에 변수를 선언하면 이 변수 역시 전역변수가 되며 비결정적 실행을 유발할 수 있다.
- 외부 명령 실행 : 프로그래밍 과정에서 외부 명령을 실행할 수 있다(Go 의 os/exec 등). 그러나 체인코드가 실행되는 환경마다 동일한 실행결과는 보장할 수 없다.

2.2 관련 연구

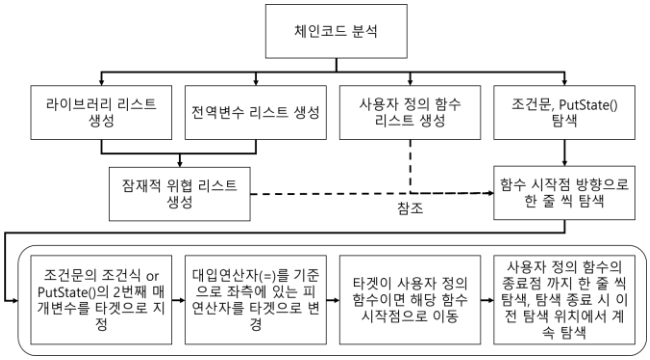
ChainSecurity 는 체인코드 정적 분석 툴인 ChaincodeScanner 를 제공한다. ChaincodeScanner 는 Go 로 작성된 체인코드를 분석할 수 있으며, 체인코드에서 발생할 수 있는 6 가지 잠재적 위협을 제안했다[2,3].

[4]에서는 ChaincodeScanner 가 제안한 6 가지 잠재적 위협 외에 Go 의 특성, 하이퍼레저 패브릭의 특성, 블록체인 외부 요소, State 데이터베이스에 의해 발생하는 잠재적 위협을 분류했다. 그리고 AST(Abstract Syntax Tree)를 생성하고 AST 를 통해 CFG(Control Flow Graph)를 생성해 잠재적 위협을 탐지했다.

2.3 비결정적 실행 탐지 기법 제안

기존의 연구들은 Go 로 작성된 체인코드의 비결정적 실행만 탐지할 수 있다. 그러나 체인코드는 Go 뿐만 아니라 자바, 자바스크립트로도 작성된다. 체인코드는 사용자가 정의한 함수를 호출해 트랜잭션을 실행한다. 함수 내부의 PutState()는 분산원장에 데이터를 입력하거나 변경한다. 비결정적인 데이터가 PutState()의 매개변수가 된다면 각각의 피어의 실행 결과는 다르게 나올 것이다. 뿐만 아니라 if 문과 같은 조건문의 조건식에 비결정적인 수식이 존재한다면 다른 결과로 분기 될 수 있다. 이 논문에서는 이러한 점을 착안해 지역변수 추적을 통해 비결정적 실행을 탐지하며 동작 순서는 다음과 같다.

1. 체인코드를 스캔해 라이브러리 리스트, 전역변수 리스트, 함수 리스트를 생성한다.
2. 라이브러리 리스트에서 잠재적 위협 패턴에 포함되는 라이브러리를 탐색한다.(Time, Random, CMD 등)
3. PutState() 함수를 찾고, 2 번째 파라미터를 타겟으로 지정한다.
4. 타겟이 잠재적 위협 패턴에 포함되는지 확인하고 포함되면 현재 위치가 잠재적 위협 임을 알린다.
5. 잠재적 위협이 아닐 경우 한 줄 씩 위로 올라가며 타겟을 탐색한다.
6. 타겟이 존재하고 타겟이 연산으로 나온 결과라면 대입연산자(=) 왼쪽에 있는 피연산자를 타겟으로 변경한다.
7. 타겟이 사용자 정의 함수라면 해당 함수로 이동한다. 해당 함수를 한 줄씩 스캔해 잠재적 위협 패턴이 존재하는지 확인한다. 함수 내에 위협 패턴이 존재하는 경우 잠재적 위협임을 알린다. 함수의 마지막 지점까지 스캔하면 함수를 빠져나와 해당 지점 윗줄부터 스캔해 타겟을 탐색한다.
8. 위 과정을 해당 함수의 시작점 까지 진행한다.



(그림 1) 비결정적 실행 탐지 프로세스

Ⅲ. 결론

이 논문에서는 체인코드의 비결정적 실행 탐지 기법에 대해 제안하였다. 기존 연구[4]는 Go 의 라이브러리를 사용했으므로, Go 에만 적용 가능하다. 실제 체인코드는 Go 뿐만 아니라 Java, Node.js 도 사용되며 이 로직을 다른 프로그래밍 언어에 적용하기에는 많은 노력이 소모된다. 이 논문에서 제안한 기법은 해당 프로그래밍 언어의 라이브러리를 사용하지 않고 소스코드를 분석해 함수 내의 지역변수를 추적하는 방법이다. 체인코드를 작성할 수 있는 Go, 자바, 자바스크립트에서 대입연산자를 사용하는 방법은 똑같으므로, 각 프로그래밍 언어에 해당되는 블랙리스트 라이브러리만 정의하면 Go 뿐만 아니라 자바, 자바스크립트로 작성된 체인코드의 비결정적 실행을 탐지할 수 있다.

ACKNOWLEDGMENT

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학 ICT 연구센터지원사업(IITP-2020-2018-0-01396)과 2018 년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업(NRF-2018R1D1A1B07048697)과 2020 년도 정부(산업통상자원부)의 재원으로 한국산업기술진흥원의 지원을 받아 수행된 연구임 (P0008703, 2020 년 산업전문인력역량강화사업)

참 고 문 헌

[1] LG CNS. “가상화폐와 블록체인 해킹 사례”, https://blog.lgcns.com/1874

[2] Chain security. “Chaincode Scanner”, https://chaincode.chainsecurity.com/

[3] Tobias Kaiser. “ChainSecurity’s Chaincode Scanner: a powerful Security Checker for Hyperledger Fabric Smart Contracts”, https://medium.com/chainsecurity/release-of-hyperchecker-2dff2ebe30cc

[4] Kazuhiro Yamashita, Yoshihide Nomura, Ence Zhou, Bingfeng Pi, Sun Jun. “Potential Risks of Hyperledger Fabric Smart Contracts”, IWBOSE 2019, 2019.