(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2022년10월17일
(11) 등록번호 10-2454845
(24) 등록일자 2022년10월11일

(51) 국제특허분류(Int. Cl.)
G06F 21/12 (2013.01) *G06F 21/57* (2013.01)
G06F 21/64 (2013.01) *G06F 8/41* (2018.01)
G06Q 10/10 (2022.01)
(52) CPC특허분류
G06F 21/125 (2013.01)
G06F 21/57 (2013.01)
(21) 출원번호 10-2021-0006233
(22) 출원일자 2021년01월15일
심사청구일자 2021년01월15일
(65) 공개번호 10-2022-0103518
(43) 공개일자 2022년07월22일
(56) 선행기술조사문헌
KR101849912 B1*
KR102008001 B1*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
충남대학교 산학협력단
대전광역시 유성구 대학로 99 (궁동, 충남대학교)
(72) 발명자
조은선
대전광역시 유성구 어은로 57 한빛아파트, 110동 1504호
김지수
대전광역시 유성구 대학로 169 301호
(74) 대리인
이은철, 김재문

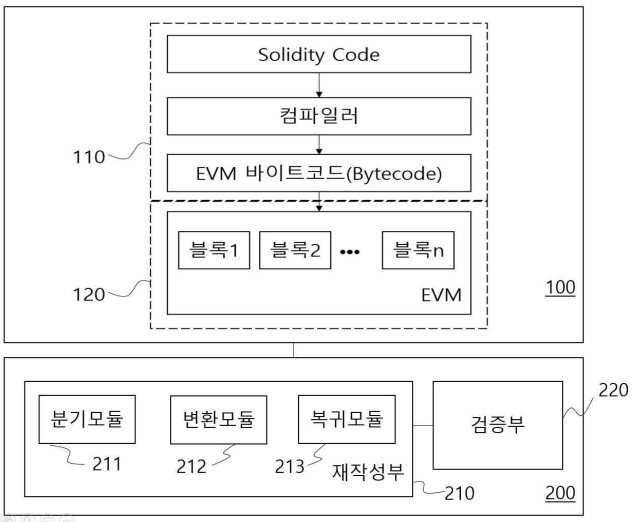
전체 청구항 수 : 총 6 항 심사관 : 문남두

(54) 발명의 명칭 **스마트 컨트랙트 재작성기**

(57) 요 약

본 발명은 스마트 컨트랙트 재작성기에 관한 것으로, 컨트랙트들을 위해 동적으로 실행중 취약점에 대응하기 위한 스마트 컨트랙트 재작성기에 관한 것이다. 본 발명의 일 실시예에 따르면, 일반적으로 시중에 있는 source code level의 취약점 분석도구와는 달리 컴파일된 bytecode를 대상으로 재작성하는 프로그램이다. 따라서 source code의 접근 없이 bytecode를 재작성하여 접근 제어나 취약점을 대비할 수 있는 코드를 주입할 수 있게 한다.

대 표 도 - 도1



(52) CPC특허분류
G06F 21/64 (2013.01)
G06F 8/41 (2013.01)
G06Q 10/10 (2021.08)

이 발명을 지원한 국가연구개발사업

과제고유번호	1711102829
과제번호	2018-0-00251-003
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기획평가원
연구사업명	정보보호핵심원천기술개발(R&D)
연구과제명	스마트 컨트랙트 프라이버시 보호 및 취약점 분석 기술개발
기 여 율	1/1
과제수행기관명	충남대학교 산학협력단
연구기간	2018.04.01 ~ 2020.12.31

공지예외적용 : 있음

명 세 서

청구범위

청구항 1

Solidity 언어를 이더리움 컴파일러에 의해 바이트 코드로 변환한 이후, 복수의 블록으로 구성되는 EVM 환경에서 상기 바이트 코드를 실행하는 스마트 컨트랙트(100);

상기 EVM을 동적 모니터링하는 모니터링부(200);를 포함하며,

상기 모니터링부는,

스마트 컨트랙트가 바이트 코드 형태로 상기 EVM 상에서 실행이 개시되는 시점으로 상기 바이트 코드가 모니터링 가능한 제2 바이트 코드로 변경시키는 재작성부(210);를 포함하며,

상기 재작성부는 바이트코드 실행 개시 시점에 주어진 바이트 코드의 함수 선택자 위치에 코드를 주입할 수 있도록 변경시키는 것을 특징으로 하는 스마트 컨트랙트 재작성기.

청구항 2

제1항에 있어서,

상기 모니터링부는,

상기 EVM에서 실행 전 첨부된 메타데이터를 가지고 상기 제2 바이트 코드와 원본 바이트 코드에 내재되어 있는 정보의 무결성을 검증하는 검증부;를 더 포함하는 것을 특징으로 하는 스마트 컨트랙트 재작성기.

청구항 3

제2항에 있어서,

상기 검증부는 코드 해시 검증시 변경된 바이트 코드가 아닌 이전 바이트 코드에 대해 검증하는 것을 특징으로 하는 스마트 컨트랙트 재작성기.

청구항 4

제1항에 있어서,

상기 모니터링부는 의도된 모니터링 지점이 수행되는 시점에 접근 제어, 조건 검사, 로깅, 정지 중 어느 하나 이상을 수행하는 것을 특징으로 하는 스마트 컨트랙트 재작성기.

청구항 5

삭제

청구항 6

제1항에 있어서,

상기 재작성부는 바이트 코드의 함수 선택자 위치인 분기 대상 주소를 조작하여 다른 위치로 분기하게 하는 분기모듈(211);

상기 분기모듈을 통한 분기 대상에 주입될 코드가 바이트코드로 변환되는 변환모듈(212); 및

코드 주입 수행을 마치게 되면 원래 위치로 복귀시키는 복귀모듈(213);을 포함하는 것을 특징으로 하는 스마트 컨트랙트 재작성기.

청구항 7

스마트 컨트랙트 재작성 방법에 있어서,

(a)스마트 컨트랙트의 기본 실행을 위한 Ethereum virtual machine(EVM)에 주입되어 동작하는 단계;

(b)상기 EVM에 본 발명이 주입되어 실행중에 스마트 컨트랙트의 bytecode를 재작성하는 단계;

(c)의도된 모니터링 지점이 수행되는 시점으로서, 접근제어나 기타 조건 검사 및 로깅이나 정지 중 어느 하나 이상의 기능이 추가되었는지 판단하는 단계; 및

(d)상기 스마트 컨트랙트의 함수 호출 관계에 있어 블록체인 상에 올라가 있는 스마트 컨트랙트에 대해 상기 (c) 단계의 특정 기능을 하는 코드가 주입된 스마트 컨트랙트의 bytecode를 재작성하는 단계;를 포함하며,

상기 (b)단계는 바이트코드 실행 개시 시점에 주어진 바이트 코드의 함수 선택자 위치에 코드를 주입할 수 있도록 변경시키는 것을 특징으로 하는 스마트 컨트랙트 재작성 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 스마트 컨트랙트 재작성기에 관한 것으로, 보다 상세하게는 컨트랙트들을 위해 동적으로 실행중 취약점에 대응하기 위한 스마트 컨트랙트 재작성기에 관한 것이다.

배 경 기 술

[0002] 스마트 컨트랙트(Smart contract)란 블록체인을 기반으로 금융거래, 부동산 계약, 공증 등 다양한 형태의 계약을 체결하고 이행하는 것을 말하는 것으로 1994년 제안되었다. 이후 2015년에는 스마트 컨트랙트를 구현하기 위한 분산 컴퓨팅 플랫폼이자 운영체제인 이더리움(Ethereum)이 등장했다. 실제 구현된 이더리움에는 제안된 스마트 컨트렉트의 이상과 달리 여러 취약점이 존재한다. 그 때문에 DAO 해킹 사건, Parity 사고 등 금융피해들이 발생했다.

[0003] 스마트 컨트랙트의 취약점은 경제적인 부분과 직접적인 연관이 있으므로 이에 대한 대응은 필수적이다. 이에 따라 이를 위한 Solidity 등의 소스코드 레벨의 취약점 검증기들이 개발되고 있으며, 배포 전후 적용 가능한 바이트 코드 레벨의 취약점 도구도 등장하고 있다.

[0004] 현재까지 이러한 도구들은 주로 개발단계에서 취약점을 검출하는 데에 초점을 두고 있어, 안전하지 않은 코드의 생산을 막는 데에 도움을 준다. 하지만, 이러한 기존 도구들은 이미 생산된 코드에 대한 대응책으로는 부족하다. 특히, 스마트 컨트렉트는 한번 제작되어 체인에 유입된 후에는 소멸될 수 없으므로, 기존의 안전하지 않은 코드의 실행을 막을 수 있는 기능이 필요하다.

선행기술문헌

특허문헌

[0005] (특허문헌 0001) (특허문헌1)공개번호 2020-0072318(2020.06.22.)

발명의 내용

해결하려는 과제

[0006] 본 발명은 상술한 문제를 해결하고자 고안한 것으로, 스마트 컨트랙트의 source code 접근 없이 bytecode를 재작성하도록 하는 스마트 컨트랙트 재작성기를 제공함에 목적이 있다.

과제의 해결 수단

[0007] 본 발명의 일측면에 따른 스마트 컨트랙트 재작성기는 Solidity 언어를 이더리움 컴파일러에 의해 바이트 코드로 변환하는 컴파일러부(110); 및 복수의 블록으로 구성되는 EVM 환경에서 상기 바이트 코드가 배포되어 실행되는 EVM(120)을 동적 모니터링하는 모니터링부(200);를 포함하며, 상기 모니터링부는 스마트 컨트랙트가 바이트 코드 형태로 상기 EVM 상에서 실행이 개시되는 시점으로 상기 바이트 코드가 모니터링 가능한 제2 바이트 코드로 변경시키는 재작성부(210);를 포함한다.

[0008] 바람직하게 모니터링부는 EVM에서 실행 전 첨부된 메타데이터를 가지고 상기 제2 바이트 코드와 원본 바이트 코드에 내재되어 있는 정보의 무결성을 검증하는 검증부;를 더 포함한다.

[0009] 바람직하게 상기 검증부는 코드 해시 검증시 변경된 바이트 코드가 아닌 이전 바이트 코드에 대해 검증한다.

[0010] 바람직하게 상기 모니터링부는 의도된 모니터링 지점이 수행되는 시점에 접근 제어, 조건 검사, 로깅, 정지 중 어느 하나 이상을 수행한다.

[0011] 바람직하게 상기 재작성부는 바이트코드 실행 개시 시점에 주어진 바이트 코드의 함수 선택자 위치에 코드를 주입할 수 있도록 변경시킨다.

[0012] 바람직하게 상기 재작성부는 바이트 코드의 함수 선택자 위치인 분기 대상 주소를 조작하여 다른 위치로 분기하게 하는 분기모듈(211); 상기 분기모듈을 통한 분기 대상에 주입될 코드가 바이트코드로 변환되는 변환모듈(212); 및 상기 코드 주입 수행을 마치게 되면 원래 위치로 복귀시키는 복귀모듈(213);을 포함한다.

[0013] 한편, 본 발명의 일 측면에 따른 스마트 컨트랙트 재작성 방법은 (a)스마트 컨트랙트의 기본 실행을 위한 Ethereum virtual machine(EVM)에 주입되어 동작하는 단계; (b)상기 EVM에 본 발명이 주입되어 실행중에 스마트 컨트랙트의 bytecode를 재작성하는 단계; (c)의도된 모니터링 지점이 수행되는 시점으로서, 접근제어나 기타 조건 검사 및 로깅이나 정지 중 어느 하나 이상의 기능이 추가되었는지 판단하는 단계; 및 (d)상기 스마트 컨트랙트의 함수 호출 관계에 있어 블록체인 상에 올라가 있는 스마트 컨트랙트에 대해 상기 (c) 단계의 특정 기능을 하는 코드가 주입된 스마트 컨트랙트의 bytecode를 재작성하는 단계;를 포함한다.

발명의 효과

[0014] 본 발명의 일 실시예에 따르면, 일반적으로 시중에 있는 source code level의 취약점 분석도구와는 달리 컴파일된 bytecode를 대상으로 재작성하는 프로그램이다. 따라서 source code의 접근 없이 bytecode를 재작성하여 접근 제어나 취약점을 대비할 수 있는 코드를 주입할 수 있게 한다.

도면의 간단한 설명

[0015] 도 1은 본 발명의 일 실시예에 따른 스마트 컨트랙트 재작성기 구성을 나타낸 도면이다.

도 2는 스마트 컨트랙트의 동작 구조를 나타낸 도면이다.

도 3은 본 발명의 일 실시예에 따른 스마트 컨트랙트 재작성기의 구성을 나타낸 도면이다.

도 4는 배포된 컨트랙트가 함수 호출에 대해 정상적으로 동작하는 모습을 나타낸 것이다.

도 5는 코드 주입과 코드 주입 이후의 시간측정을 나타낸 것이다.

도 6은 본 발명의 일 실시예에 따른 스마트 컨트랙트 재작성 방법의 동작을 설명하기 위한 도면이다.

발명을 실시하기 위한 구체적인 내용

[0016] 본 발명의 실시예에서 제시되는 특정한 구조 내지 기능적 설명들은 단지 본 발명의 개념에 따른 실시예를 설명하기 위한 목적으로 예시된 것으로, 본 발명의 개념에 따른 실시예들은 다양한 형태로 실시될 수 있다. 또한, 본 명세서에 설명된 실시예들에 한정되는 것으로 해석되어서는 아니 되며, 본 발명의 사상 및 기술 범위에 포함되는 모든 변경물, 균등물 내지 대체물을 포함하는 것으로 이해되어야 한다.

[0017] 한편, 본 발명에서 제1 및/또는 제2 등의 용어는 다양한 구성요소들을 설명하는데 사용될 수 있지만, 상기 구성요소들은 상기 용어들에 한정되지는 않는다. 상기 용어들은 하나의 구성요소를 다른 구성요소들과 구별하는 목적으로만, 예컨대 본 발명의 개념에 따른 권리 범위로부터 벗어나지 않는 범위 내에서, 제1 구성요소는 제2 구

성요소로 명명될 수 있고, 유사하게 제2 구성요소는 제1 구성요소로도 명명될 수 있다.

[0018] 이하 첨부된 도면을 참조하여 본 발명의 실시예를 설명한다. 본 발명의 실시예를 설명함에 있어서, 관련된 공지 기능 혹은 구성에 대한 설명이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우 그 설명을 생략하였다.

[0019] 도 1은 본 발명의 일 실시예에 따른 스마트 컨트랙트 재작성기(10)를 나타낸 구성도이다. 도 1에 도시된 바와 같이, 컴파일러부(110), EVM(120)를 구성하는 스마트 컨트랙트(100)와 모니터링부(200)를 포함한다.

[0020] 컴파일러부(110)는 Solidity 언어를 이더리움 컴파일러에 의해 바이트 코드로 변환하는 구성이다. EVM(120)은 복수의 블록으로 구성되는 EVM 환경에서 상기 바이트 코드가 배포되어 실행되는 구성이다.

[0021] 모니터링부(200)는 EVM을 실시간 동적 모니터링하는 구성이다. 모니터링부는 의도된 모니터링 지점이 수행되는 시점에 접근 제어, 조건 검사, 로깅, 정지 중 어느 하나 이상을 수행한다.

[0022] 먼저, 본 실시예에 따른 스마트 컨트랙트 재작성기의 모니터링부는 스마트 컨트랙트가 바이트 코드 형태로 EVM 상에서 실행이 개시되는 시점에서, 바이트 코드를 모니터링 가능한 제2 바이트 코드로 변경시키는 재작성부(210)를 포함한다.

[0023] 재작성부는 바이트코드 실행 개시 시점에 주어진 바이트 코드의 함수 선택자 위치에 코드를 주입할 수 있도록 변경한다. 이러한 재작성부는 바이트 코드의 함수 선택자 위치인 분기 대상 주소를 조작하여 다른 위치로 분기하게 하는 분기모듈(211), 분기모듈을 통한 분기 대상에 주입될 코드가 바이트코드로 변환되는 변환모듈(212), 코드 주입 수행을 마치게 되면 원래 위치로 복귀시키는 복귀모듈(213)을 포함한다.

[0024] 여기에 본 실시예에 따른 스마트 컨트랙트 재작성기는 EVM에서 실행 전 첨부된 메타데이터를 가지고 상기 제2 바이트 코드와 원본 바이트 코드에 내재되어 있는 정보의 무결성을 검증하는 검증부(220)를 포함한다. 이때, 검증부는 코드 해시 검증시 변경된 바이트 코드가 아닌 이전 바이트 코드에 대해 검증한다.

[0025] 이러한 모니터링부는 아래에서 EVM+2.0에 해당하는 구성으로 아래에서 구체적으로 설명하기로 한다.

[0026] 본 발명의 일 실시예에 따른 스마트 컨트랙트 재작성기는 스마트 컨트랙트의 기본 실행을 위한 Ethereum virtual machine(EVM)에 주입되어 동작한다. EVM에 본 발명이 주입되어 실행중에 스마트 컨트랙트의 bytecode를 재작성한다.

[0027] 대부분의 취약점들은 함수 호출 간에 발생하고 취약점 분석 도구들 또한 함수 호출 부분에 초점을 맞춰 분석한다. 따라서 본 발명 또한 스마트 컨트랙트의 함수 호출 관계에 있어 bytecode를 재작성한다.

[0028] 스마트 컨트랙트의 동작 중 bytecode를 scan하여 함수 호출 부분을 스마트 컨트랙트의 function selector를 이용하여 detecting 한다. detecting 한 함수 호출 부분을 토대로 함수호출 관계에서 함수의 실행지점을 원래의 함수 몸체부분이 아닌 bytecode의 뒷 부분에 주입한 취약점 탐지부분 코드로 옮긴다. 그 후 다시 원래의 함수의 몸체 부분으로 돌아가도록 실행지점을 옮긴다. 이러한 동작이 가능하도록 프로그램의 실행 중 스마트 컨트랙트의 bytecode를 재작성한다.

[0029] 스마트 컨트랙트의 취약점을 방지하는 연구는 주로 소스코드나 바이트코드 레벨의 취약점을 검증하는 방법으로 제공된다. 따라서 주로 개발단계에서 취약점을 검출하는데에 초점을 두고 있어, 이미 체인에 유입된 취약한 컨트렉트에 대해 실행 단계의 제어에는 부족한 면이 있다. 이를 해결하기 위해 본 발명의 일 실시예에 따른 스마트 컨트랙트 재작성기에서는 이더리움 가상머신인 EVM을 개선한 EVM+2.0을 소개한다.

[0030] 본 발명의 일 실시예에서는 기존 이더리움 스마트 컨트랙트의 실행 환경인 EVM을 동적 모니터링이 가능하도록 확장한 EVM+2.0을 소개한다. EVM+2.0은 실제 컨트랙트를 수행해서 검증하는 노드들에서 기존의 EVM 대신 사용함으로써, 안전한 실행을 가능하게 한다. EVM+2.0은 동적 모니터링 기능을 지원하고 있으며 검증 노드에서 사용될 때 안전한 실행을 가능하게 한다.

[0031] 도 2는 스마트 컨트랙트의 동작 구조를 나타낸 도면이다. 도 2에 도시된 바와 같이, 스마트 컨트랙트는 Solidity라는 프로그래밍 언어로 구현되어 있다. Solidity 프로그램은 이더리움 컴파일러 (Solc)에 의해 EVM 바이트코드로 변환된다. 생성된 바이트코드는 배포(deploy)되어 블록에 올라가고 각 노드에서 가상머신인 EVM 환경에서 실행된다.

[0032] EVM 바이트코드는 바이트의 시퀀스로서 크게 3부분으로 나눌 수 있다. 컨트랙트를 생성하는 부분, 실제 컨트랙

트가 실행되는 런타임 부분, 런타임 부문의 끝에 위치하는 메타데이터 해시이다. 메타데이터 해시는 소스코드, 컴파일 방법 등 계약에 필요한 정보를 담고 있다.

[0033] EVM 바이트코드는 Solidity와 달리 함수가 식별 가능하지 않고 함수 선택자(selector)라는 것으로 대신한다. 함수 선택자는 스마트 컨트랙트의 정적인 코드 내에 실행 코드 부분의 앞쪽에 존재하는 코드로서 분기 명령을 통해 함수 호출을 진행하고 있다.

[0034] 기존의 스마트 컨트랙트 취약점을 방지하기 위한 도구로는 소스 코드 레벨의 검증, 바이트 코드 레벨의 검증 등이 주류를 이루고 있다. 이러한 검증 기법들은 요약 해석이나 심볼릭 실행 등의 프로그램 분석 기법을 사용하거나 논리 증명기를 통해 의미적인 일관성을 유지하는지를 추론에 의해 검토하게 된다.

[0035] 그러나 이러한 방법들은 정해진 취약점을 내포하는지에 대해 실행 전에 검증을 하는 것이 목적이기 때문에 실시간성과 실제 위험성 등을 고려할 필요가 없다는 측면이 있다. 그리고 문제를 검출하였을 때의 행동은 사용자에게 보고하는 것으로 가정되며, EVM 자체를 개선하여 실행 도중의 안전성을 보장하는 것과는 거리가 있다.

[0036] 안전하지 않은 행위를 실행 중 감지하는 방법은 Java나 웹의 접근 제어, 운영체제나 안티 바이러스 제품의 실시간 모니터링 등의 예가 있다. 이러한 도구들의 주된 방법은 프로그램 실행 중 특정 행위가 발생할 때 정지나 로깅 등의 작업을 상황에 맞게 주입할 수 있도록 하는 것이다. 이러한 기능들은 가상환경이나 프로토콜 또는 운영체제가 지원하는 방법을 사용하여 이루어진다.

[0037] 그러나 EVM은 이러한 기능을 제공하지 않으므로 실행 중 프로그램의 행위를 모니터링하는데에 어려움이 있다. 따라서 스마트 컨트랙트가 기존과 동일한 방식으로 실행되면서 실시간 모니터링이 가능하도록 EVM을 개선하는 작업이 필요하다.

[0038] 본 실시예에서는 실시간 모니터링 기능을 지원하도록 확장된 EVM+2.0에 대해 설명하기로 한다. EVM+2.0은 3가지 시점에서 기존 EVM과 다르게 동작한다. 첫째, 스마트 컨트랙트가 바이트 코드 형태로 EVM 상에서 실행이 개시되는 시점으로, 바이트 코드는 모니터링 가능한 바이트 코드로 변경된다. 둘째, EVM에서 실행 전 검증을 하는 부분들로서 단일 시점이 아닌 여러 부분에서 실시되고 있으며, 대부분 첨부된 메타데이터를 가지고 바이트 코드에 내재되어 있는 정보의 무결성을 검증하고 있다. EVM+2.0은 재작성된 바이트 코드와 원본 바이트 코드를 모두 관리하여 적절한 검증이 효과적으로 이루어지게 한다. 셋째, 의도된 모니터링 지점이 수행되는 시점으로서, 접근 제어나 기타 조건 검사 및 로깅이나 정지 등의 새로운 기능이 추가된다.

[0039] 바이트코드 변경 및 검증과 관련하여 제안하는 EVM+2.0은 바이트 코드 실행 개시 시점에, 주어진 바이트 코드의 일부 부분에 코드를 주입할 수 있도로 바꾼다. 코드 주입은 코드 전체의 행위를 바꿀 수도 있는 사안이므로 주입 위치와 주입 내용에 대해 제한이 필요하다. 예를 들어 Java에 코드 주입을 지원하는 AspectJ는 메소드 정의마다 코드를 주입할 수 있도록 하고 있다. 제안하는 EVM+2.0도 이와 유사하게 함수 위치를 중심으로 코드 주입을 지원한다. 그러나 코드 바이트코드로 바뀐 Solidity 컨트랙트는 함수 호출이 명확히 식별되지 않는다는 문제가 있다. 본 EVM+2.0에서는 함수 선택자 위치에 코드를 주입할 수 있게 하였다.

[0040] 이를 위해 EVM+2.0.은 함수 선택자의 분기 대상 주소를 조작하여 다른 위치로 분기하게 한다. 이때, 새로운 분기 대상에는 주입될 코드가 바이트 코드로 변환되어 존재하며, 이 코드의 수행을 마치게 되면 원래 위치로 복귀한다.

[0041] 예를 들어 도 3과 같이 원래의 JUMPI의 대상위치인 PUSH2의 피연산자 값(도 2의 0x10)에 해당 바이트 코드 내의 다른 위치(주로 가장 마지막 위치)를 저장한다. 바이트 코드의 마지막 위치에는 각 함수마다 JUMPDEST를 삽입하고 이 위치에 코드주입이 가능하도록 한 후, 원래의 JUMPI의 대상 위치인 PUSH2 명령과 기존 피연산자를 추가한다. 이러한 과정을 각 함수에 대해 실행한다.

[0042] 위에서 구현한 내용이 실제로 이루어지기 위해서는 EVM에서 기존에 이루어지고 있었던 검증 과정도 개선되어야 한다. 예를 들어 기존의 메타데이터로 새로운 바이트코드를 검증하거나 기존 코드에 대해 JUMPDEST 위치 등의 분석 결과를 확보하여 새로운 코드에 적용한다면 검증 실패로 인해 실행이 진행될 수 없다.

[0043] 제안하는 EVM+2.0은 기존 EVM의 해당 부분을 수정하여 기존 코드와 새로운 바이트코드를 동시에 보유하고 관리하도록 개선하였다. 즉, 코드 해시 검증을 하는 부분에서는 새로 변경된 코드가 아닌 예전 코드에 대해 검증하도록 관리하며, 이외에도 추가된 코드의 정렬(alignment) 등 여러 군데 분포되어 있는 검증에 대해 체계적으로 관리해야한다. 현재는 일부에 대해 구동중이며 Solidity 0.6x 이상의 언어로 기술된 컨트랙트에 대해서는 정상 작동됨을 확인하였다.

[0044] 코드 주입 내용은 컨트랙트마다 JSON 파일을 두어 명시하도록 하는데, 인자에 대해 조건검사를 하게 되며 이에 따른 정지 명령을 내릴 수 있다. 현재는 제한적으로 구현되어 있는데, 추후 동적인 성질을 표현하는데 제약이 적은 상위 언어로 확장할 수 있다.

[0045] EVM+2.0의 구현을 위해 기존의 EVM의 소스가 필요하며 재작성기, 개선된 검증기, 실행 중 관리기, 주입 코드 변환기 등이 필요하다 이러한 코드들은 Go 언어로 구현되었으며 Ubuntu에서 실험하였다. 실험 환경으로 EVM 1.9.10과 Geth 1.9.10, Solidity 0.6x, Go 1.13을 사용하였다.

[0046] 이더리움의 Geth는 EVM을 포함하는 이더리움 공식 클라이언트 소프트웨어이다. 실험을 위해 Geth의 기존 EVM 대신 EVM+2.0으로 바꾸어 빌드한 다음, EtherScan 상의 스마트 컨트랙트 3개를 추출하여 Geth console 상에서 가상 네트워크를 대상으로 마이닝을 하며 실제 배포 실행시켰다.

[0047] 도 4는 배포된 컨트랙트가 함수 호출데 대해 정상적으로 동작하는 모습을 나타낸 것이다. 도 5는 코드 주입과 코드 주입 이후의 시간측정을 나타낸 것이다. 도 3은 블록에 올라간 컨트랙트의 함수인 bid()를 실행했을 때 주입된 코드로 분기 후 돌아온 후 정상적으로 작동한 화면이다. 스마트 컨트랙트에 코드를 주입하여 주어진 조건을 만족하지 않으면 컨트랙트를 종료하도록 하였으며, 4개 모두 정상적으로 동작하였다.

[0048] 또한 코드 주입과 코드 주입 이후의 시간측정을 도 4에 도시된 바와 같이 실시하였다.

[0049] 도 6은 본 발명의 일 실시예에 따른 스마트 컨트랙트의 동작을 설명하기 위한 도면이다. 도 6에 도시된 바와 같이, 스마트 컨트랙트의 기본 실행을 위한 Ethereum virtual machine(EVM)에 주입되어 동작한다(601).

[0050] 다음으로 EVM에 본 발명이 주입되어 실행중에 스마트 컨트랙트의 bytecode를 재작성한다(602).

[0051] 의도된 모니터링 지점이 수행되는 시점으로서, 접근제어나 기타 조건 검사 및 로깅이나 정지 등의 새로운 기능이 추가되었는지 판단한다(603). 이때, 대부분의 취약점들은 함수 호출 간에 발생하고 취약점 분석 도구들 또한 함수 호출 부분에 초점을 맞춰 분석한다.

[0052] 다음으로 스마트 컨트랙트의 함수 호출 관계에 있어 블록체인 상에 올라가 있는 스마트 컨트랙트에 대해 특정 기능을 하는 코드를 주입된 bytecode를 재작성한다(604).

[0053] 스마트 컨트랙트의 동작 중 bytecode를 scan하여 함수 호출 부분을 스마트 컨트랙트의 함수 선택자(function selector)를 이용하여 detecting 한다. detecting 한 함수 호출 부분을 토대로 함수호출 관계에서 함수의 실행 지점을 원래의 함수 몸체부분이 아닌 bytecode의 뒷 부분에 주입한 취약점 탐지부분 코드로 옮긴다. 그 후 다시 원래의 함수의 몸체 부분으로 돌아가도록 실행지점을 옮긴다. 이러한 동작이 가능하도록 프로그램의 실행 중 스마트 컨트랙트의 bytecode를 재작성한다.

[0054] 본 실시예에서 소개하는 EVM+2.0은 스마트 컨트랙트의 취약점을 방지하기 위한 EVM의 확장으로서 이미 체인에 유입된 취약한 컨트랙트에 대해 실행 단계의 제어한느 것이 목적이다. 따라서 EVM+2.0은 코드 재작성 방식의 동적 모니터링 기능을 지원하고 있으며 적은 오버헤드로 안전한 실행을 가능하게 한다.

[0055] 본 발명은 일반적으로 시중에 있는 source code level의 취약점 분석도구와는 달리 컴파일된 bytecode를 대상으로 재작성하는 프로그램이다. 따라서 source code의 접근없이 bytecode 재작성을 통해 접근제어 코드를 주입하여 접근 제어나 취약점을 대비할 수 있는 코드를 주입할 수 있게 한다.

[0056] 이상에서 설명한 본 발명은 전술한 실시예 및 첨부된 도면에 의해 한정되는 것이 아니고, 본 발명의 기술적 사상을 벗어나지 않는 범위 내에서 여러 가지 치환, 변형 및 변경이 가능함은 당업자에게 명백할 것이다.

부호의 설명

[0057] 100 : 스마트 컨트랙트

110 : 컴파일러부

120 : EVM

200 : 모니터링부

210 : 재작성부

211 : 분기모듈

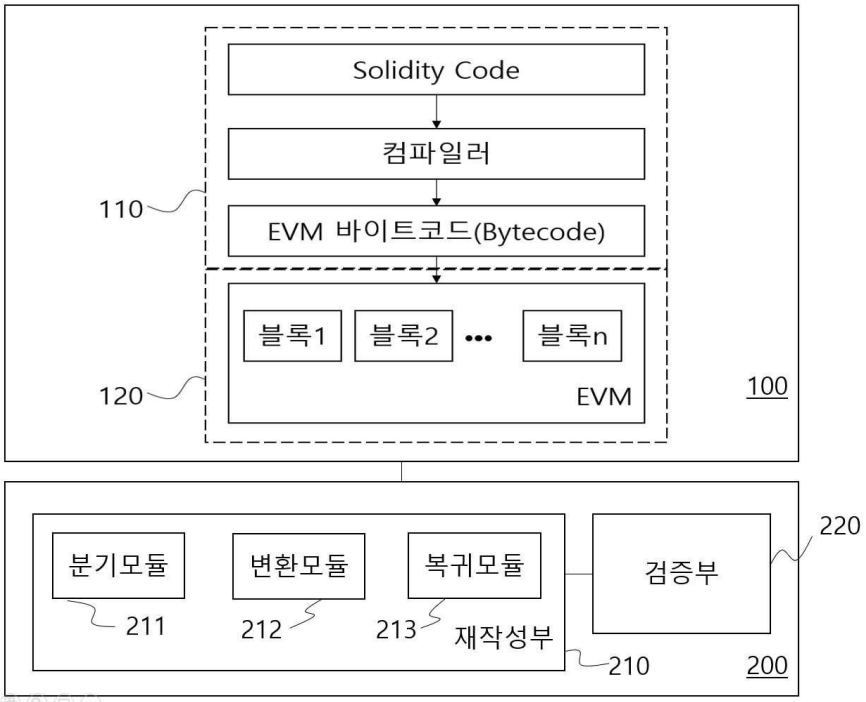
212 : 변환모듈

213 : 복귀모듈

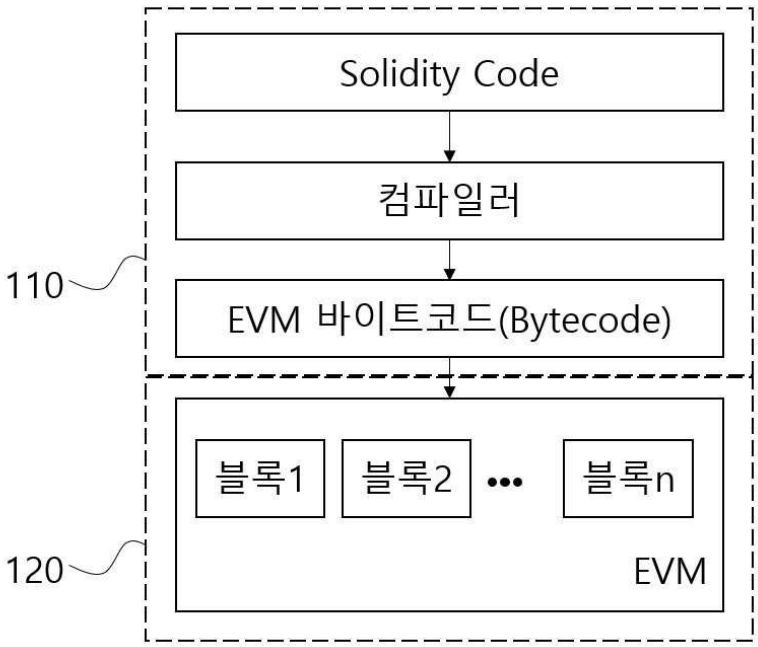
220 : 검증부

도면

도면1



도면2



도면3

```
3   PUSH1 0x80 PUSH1 0x40 MSTORE
4   CALLVALUE DUP1 ISZERO
5   PUSH2 0x10 JUMPI PUSH1 0x0
6   DUP1 REVERT JUMPDEST POP
7   ...
```

도면4

```
> auctionInterface.at(publishedAuctionAddr).bid
({from:eth.accounts[0],gas:1000000})
INFO [06-04|23:08:52.447] Submitted transaction
                    fullhash=0x55e535fd359a0a78
0e75c3a9f6c78735d259582b3e62b1502718adc38198b55
d recipient=0x2a462fE75bd05E561AAe8aD6bb12285Ef
8A52415
"0x55e535fd359a0a780e75c3a9f6c78735d259582b3e62
b1502718adc38198b55d"
```

도면5

```
> ballotInterface.at(publishedBallotAddr).vote.call(1)
evm.go/call function
Elapse Time : 3.902µs []
> ballotInterface.at(publishedBallotRAddr).vote.call(1)
evm.go/call function
Elapse Time : 3.992µs []
```

도면6