(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2021년09월23일
(11) 등록번호 10-2304861
(24) 등록일자 2021년09월15일

(51) 국제특허분류(Int. Cl.)
G06F 11/36 (2006.01) *G06F 9/455* (2018.01)
(52) CPC특허분류
G06F 11/3664 (2013.01)
G06F 11/3636 (2013.01)
(21) 출원번호 10-2021-0040963
(22) 출원일자 2021년03월30일
심사청구일자 2021년03월30일
(56) 선행기술조사문헌
KR102209676 B1*
(뒷면에 계속)

(73) 특허권자
세종대학교산학협력단
서울특별시 광진구 능동로 209 (군자동, 세종대학교)
(72) 발명자
윤주범
서울특별시 송파구 충민로4길 19, 704동 401호(장지동, 송파파인타운7단지)
김현욱
서울특별시 관악구 서원10길 25, 3층(신림동)
김주환
서울특별시 광진구 군자로 175-2, 304호(군자동)
(74) 대리인
두호특허법인

전체 청구항 수 : 총 16 항 심사관 : 김계준

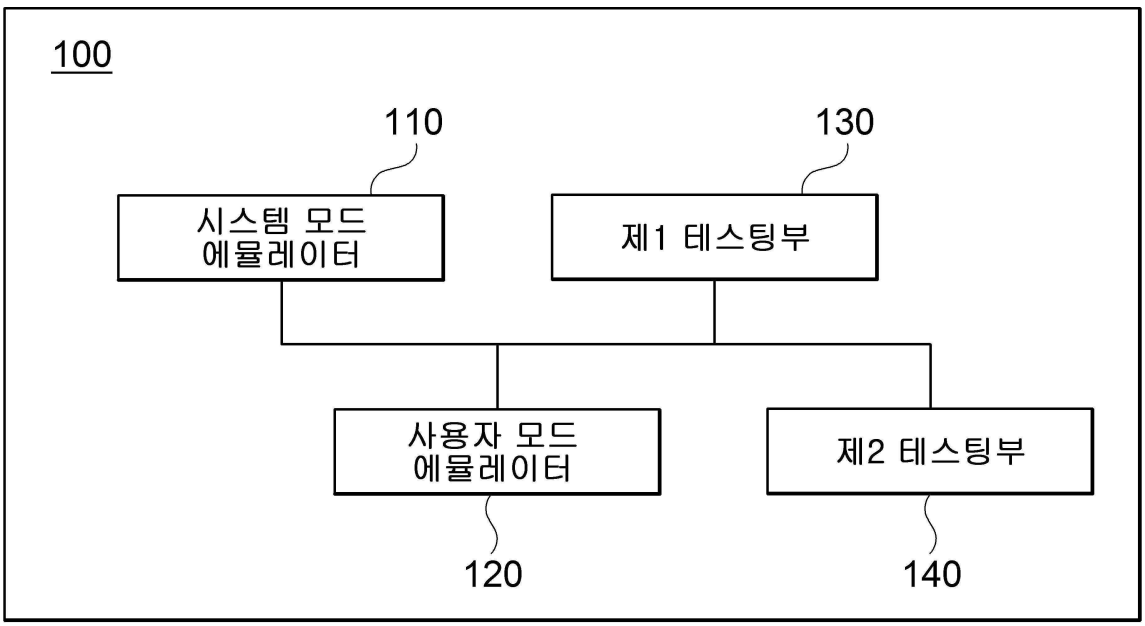
(54) 발명의 명칭 **하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치 및 방법**

(57) 요 약

하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치 및 방법이 개시된다. 일 실시예에 따르면 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치는 임의의 사물 인터넷(IoT; Internet of Things) 기기의 펌웨어에 시스템 모드 에뮬레이션 환경을 제공하는 시스템 모드 에뮬레이터; 상기 시스템 모드 에뮬레이션 환경에서 실행 중인 상기 펌웨

(뒷면에 계속)

대 표 도 - 도1



어의 하나 이상의 프로세스 중 대상 프로세스에 사용자 모드 에뮬레이션 환경을 제공하는 사용자 모드 에뮬레이터; 상기 사용자 모드 에뮬레이션 환경에서 상기 대상 프로세스에 변이 기반 퍼징(Fuzzing)을 실행하는 제1 테스팅부; 및 상기 퍼징의 실행 중 기 설정된 이벤트가 발생된 경우, 상기 시스템 모드 에뮬레이션 환경에서 상기 이벤트의 발생 당시 상기 대상 프로세스의 상태에 기초하여 발생된 이벤트를 해결하기 위한 동작을 수행하는, 제2 테스팅부를 포함하되, 상기 제1 테스팅부는, 상기 기 설정된 이벤트가 발생한 경우, 상기 퍼징의 실행을 중단하고, 기 동작의 결과에 기초하여 상기 퍼징의 실행을 재개하여 상기 펌웨어의 취약점을 검출한다.

(52) CPC특허분류
G06F 11/3684 (2013.01)
G06F 11/3688 (2013.01)
G06F 9/45508 (2013.01)
G06F 9/45558 (2013.01)
G06F 2009/45575 (2019.08)
G06F 2009/45583 (2019.08)

(56) 선행기술조사문헌
KR101559651 B1
KR101963752 B1
KR1020200080541 A
US20190114436 A1
*는 심사관에 의하여 인용된 문헌

이 발명을 지원한 국가연구개발사업
과제고유번호 1711075702
과제번호 2018-0-01423-004
부처명 과학기술정보통신부
과제관리(전문)기관명 정보통신기술진흥센터
연구사업명 정보통신기술인력양성
연구과제명 지능형 비행로봇 융합기술 연구
기 여 율 1/2
과제수행기관명 세종대학교 산학협력단
연구기간 2021.01.01 ~ 2021.12.31
이 발명을 지원한 국가연구개발사업
과제고유번호 1711120086
과제번호 2020-0-01602
부처명 과학기술정보통신부
과제관리(전문)기관명 정보통신기획평가원
연구사업명 대학ICT연구센터지원사업
연구과제명 지능형 사이버 위협 대응 기술 개발 및 인력양성
기 여 율 1/2
과제수행기관명 숭실대학교 산학협력단
연구기간 2021.01.01 ~ 2021.12.31

명 세 서

청구범위

청구항 1

임의의 사물 인터넷(IoT; Internet of Things) 기기의 펌웨어에 시스템 모드 에뮬레이션 환경을 제공하는 시스템 모드 에뮬레이터;

상기 시스템 모드 에뮬레이션 환경에서 실행 중인 상기 펌웨어의 하나 이상의 프로세스 중 대상 프로세스에 사용자 모드 에뮬레이션 환경을 제공하는 사용자 모드 에뮬레이터;

상기 사용자 모드 에뮬레이션 환경에서 상기 대상 프로세스에 변이 기반 퍼징(Fuzzing)을 실행하는 제1 테스팅부;

상기 퍼징의 실행 중 기 설정된 이벤트가 발생된 경우, 상기 시스템 모드 에뮬레이션 환경에서 상기 이벤트의 발생 당시 상기 대상 프로세스의 상태에 기초하여 발생된 이벤트를 해결하기 위한 동작을 수행하는, 제2 테스팅부; 및

상기 이벤트가 발생할 당시 상기 대상 프로세스의 메모리 상태 및 상기 퍼징에 대응하는 제1 테스트 케이스 중 적어도 하나를 상기 퍼징의 실행 결과로서 저장하고, 상기 이벤트를 해결하고자 상기 실행 결과를 상기 제2 테스팅부에 제공하고, 상기 제2 테스팅부의 상기 동작의 결과를 상기 제1 테스팅부에 제공하는 제어부를 포함하되,

상기 제1 테스팅부는,

상기 기 설정된 이벤트가 발생한 경우, 상기 퍼징의 실행을 중단하고, 상기 동작의 결과에 기초하여 상기 퍼징의 실행을 재개하여 상기 펌웨어의 취약점을 검출하는, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치.

청구항 2

삭제

청구항 3

청구항 1에 있어서,

상기 제2 테스팅부는,

상기 퍼징이 실행되는 도중 시스템 호출(syscall)이 발생한 경우, 상기 시스템 모드 에뮬레이션 환경에서 상기 시스템 호출을 해결하는, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치.

청구항 4

청구항 3에 있어서,

상기 제어부는,

상기 시스템 호출의 해결 당시 상기 대상 프로세스의 메모리 상태를 저장하고, 상기 시스템 호출이 해결될 당시 상기 대상 프로세스의 메모리 상태를 상기 제1 테스팅부에 제공하는, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치.

청구항 5

청구항 4에 있어서,

상기 제1 테스팅부는,

상기 동작의 결과로서 상기 시스템 호출이 해결될 당시 상기 대상 프로세스의 메모리 상태에 기초하여 상기 퍼징의 실행을 재개하는, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치.

청구항 6

청구항 1에 있어서,

상기 제2 테스팅부는,

상기 퍼징이 실행되는 도중 기 설정된 시간 동안 제약 조건을 해결하지 못한 경우, 상기 시스템 모드 에뮬레이션 환경에서 동적 기호 실행(Concolic Execution)을 수행하여 상기 제약 조건을 해결하는, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치.

청구항 7

청구항 6항에 있어서,

상기 동적 기호 실행을 수행하여 제약 조건이 해결되는 경우, 상기 동적 기호 실행에 대응하는 제2 테스트 케이스를 생성하는 테스트 케이스 생성부를 더 포함하는, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치.

청구항 8

청구항 7에 있어서,

상기 제어부는,

상기 제2 테스트 케이스를 저장하고, 상기 제2 테스트 케이스를 상기 제1 테스팅부에 제공하는, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치.

청구항 9

청구항 8에 있어서,

상기 제1 테스팅부는,

상기 제2 테스트 케이스에 기초하여 상기 퍼징을 재개하는, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치.

청구항 10

임의의 사물 인터넷(IoT; Internet of Things) 기기의 펌웨어에 시스템 모드 에뮬레이션 환경을 제공하는 단계;

상기 시스템 모드 에뮬레이션 환경에서 실행 중인 상기 펌웨어의 하나 이상의 프로세스 중 대상 프로세스에 사용자 모드 에뮬레이션 환경을 제공하는 단계;

상기 사용자 모드 에뮬레이션 환경에서 상기 대상 프로세스에 변이 기반 퍼징(Fuzzing)을 실행하는 단계;

상기 퍼징의 실행 중 기 설정된 이벤트가 발생한 경우, 상기 퍼징의 실행을 중단하는 단계;

상기 이벤트가 발생할 당시 상기 대상 프로세스의 메모리 상태 및 상기 퍼징에 대응하는 제1 테스트 케이스 중 적어도 하나를 상기 퍼징의 실행 결과로서 저장하는 단계;

상기 시스템 모드 에뮬레이션 환경에서 상기 이벤트의 발생 당시 상기 대상 프로세스의 상태에 기초하여 발생된 이벤트를 해결하기 위한 동작을 수행하는 단계;

상기 동작의 결과에 기초하여 상기 퍼징의 실행을 재개하는 단계; 및

상기 퍼징이 재개된 결과에 기초하여 상기 펌웨어의 취약점을 검출하는 단계를 포함하고,

상기 수행하는 단계는, 상기 이벤트를 해결하고자 상기 퍼징의 실행 결과에 기초하여 상기 이벤트를 해결하기 위한 동작을 수행하고,

상기 재개하는 단계는, 상기 퍼징을 통해 상기 펌웨어의 취약점을 검출하고자 상기 동작의 결과에 기초하여 상기 퍼징의 실행을 재개하는, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 방법.

청구항 11

삭제

청구항 12

청구항 10에 있어서,

상기 수행하는 단계는,

상기 퍼징이 실행되는 도중 시스템 호출(syscall)이 발생하는 경우, 상기 시스템 모드 에뮬레이션 환경에서 상기 시스템 호출을 해결하는, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 방법.

청구항 13

청구항 12에 있어서,

상기 시스템 호출이 해결된 경우, 상기 대상 프로세스의 메모리 상태를 저장하는 단계를 더 포함하는, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 방법.

청구항 14

청구항 13에 있어서,

상기 재개하는 단계는,

상기 시스템 호출을 해결한 상기 대상 프로세스의 메모리 상태에 기초하여 상기 퍼징을 재개하는, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 방법.

청구항 15

청구항 10에 있어서,

상기 수행하는 단계는,

상기 퍼징이 실행되는 도중 기 설정된 시간 동안 제약 조건을 해결하지 못한 경우, 상기 시스템 모드 에뮬레이션 환경에서 동적 기호 실행(Concolic Execution)을 수행하여 상기 제약 조건을 해결하는, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 방법.

청구항 16

청구항 15에 있어서,

상기 동적 기호 실행을 수행하여 제약 조건이 해결되는 경우, 상기 동적 기호 실행에 대응하는 제2 테스트 케이스를 생성하는 단계를 더 포함하는, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 방법.

청구항 17

청구항 16에 있어서,

상기 동적 기호 실행을 수행하여 제약 조건이 해결되는 경우, 상기 제2 테스트 케이스를 저장하는 단계를 더 포함하는, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 방법.

청구항 18

청구항 17에 있어서,

상기 수행하는 단계는,

상기 제2 테스트 케이스에 기초하여 상기 퍼징을 재개하는, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 방법.

발명의 설명

기 술 분 야

[0001] 개시되는 실시예들은 하이브리드 퍼징(Fuzzing)을 기반으로 펌웨어의 취약점을 검출하는 기술과 관련된다.

배 경 기 술

[0002] IoT(IoT; Internet of Things) 장비의 취약점을 찾는 것은 IoT가 상용화되고 있는 현대 사회에서 상당히 중요한 부분이다. 그러나, IoT 장비를 제조하는 제조업체는 수없이 많고, 제조사 별로 펌웨어를 구성하는 방식마저 다르기 때문에 보안 전문가들이 수동으로 IoT 장비의 펌웨어 취약점을 분석하기는 물리적으로 불가능에 가깝다고 볼 수 있다.

[0003] 이에, 최근 연구들은 IoT 장비의 취약점을 검출하는 방법에 있어서 퍼징(Fuzzing)을 사용하여 빠르고 높은 효율로 IoT 장비 펌웨어의 취약점을 찾고 있다. 초기 IoT 장비 펌웨어 퍼징 연구는 IoT 장비에 어떤 방법으로 퍼징을 진행할 것인가에 대한 연구가 많이 진행되었다. 현재까지도 펌웨어 퍼징 방식인 에뮬레이션에 대한 연구가 많이 진행되고 있다. 그러나, 최근 연구 또한 펌웨어의 에뮬레이션에 대한 연구만 이루어지고 있어 IoT 장비 펌웨어 퍼징은 일반 소프트웨어 퍼징에 비해 성능이 좋지 않다는 문제점이 있다.

[0004] 특히, 제약 조건 값이 길거나 어려우면, 쉽게 해결하지 못하고 같은 위치에서 제약 조건 값을 해결하기 위해 테스트 케이스 생성과정을 반복하여 오랜 시간 자원을 낭비하며 테스팅이 진행된다.

[0005] 그러나, 기호 실행은 이러한 문제점을 제약조건 해결기(Constraint Solver) 엔진을 사용하여 빠르게 해결할 수 있어, 퍼징을 보다 빠른 시간 안에 제약 조건을 해결하여 특정 부분의 코드 커버리지를 넓힐 수 있다.

[0006] 이렇듯 기호 실행 기법은 제약 조건이 풀리기 어려울 때, 퍼징을 보다 쉽게 제약 조건을 해결할 수 있지만, 그만큼 PC 자원을 많이 사용하고, 계속 기호 실행 기법만으로 프로그램을 탐색하면 자원 고갈 문제가 존재하는 한계가 존재한다.

선행기술문헌

특허문헌

[0007] (특허문헌 0001) 대한민국 등록특허공보 제10-1981028호(2019.05.16. 등록)

발명의 내용

해결하려는 과제

[0008] 개시되는 실시예들은 하이브리드 퍼징(Fuzzing) 기반의 펌웨어 취약점 검출 장치 및 방법을 제공하기 위한 것이

다.

과제의 해결 수단

[0009] 일 실시예에 따른 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치는, 임의의 사물 인터넷(IoT; Internet of Things) 기기의 펌웨어에 시스템 모드 에뮬레이션 환경을 제공하는 시스템 모드 에뮬레이터; 상기 시스템 모드 에뮬레이션 환경에서 실행 중인 상기 펌웨어의 하나 이상의 프로세스 중 대상 프로세스에 사용자 모드 에뮬레이션 환경을 제공하는 사용자 모드 에뮬레이터; 상기 사용자 모드 에뮬레이션 환경에서 상기 대상 프로세스에 변이 기반 징(Fuzzing)을 실행하는 제1 테스팅부; 및 상기 퍼징의 실행 중 기 설정된 이벤트가 발생된 경우, 상기 시스템 모드 에뮬레이션 환경에서 상기 이벤트의 발생 당시 상기 대상 프로세스의 메모리 상태에 기초하여 발생된 이벤트를 해결하기 위한 동작을 수행하는, 제2 테스팅부를 포함하되, 상기 제1 테스팅부는, 상기 기 설정된 이벤트가 발생한 경우, 상기 퍼징의 실행을 중단하고, 기 동작의 결과에 기초하여 상기 퍼징의 실행을 재개하여 상기 펌웨어의 취약점을 검출한다.

[0010] 일 실시예에 따른 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치는, 상기 이벤트가 발생할 당시 상기 대상 프로세스의 메모리 상태 및 상기 퍼징에 대응하는 제1 테스트 케이스 중 적어도 하나를 상기 퍼징의 실행 결과로서 저장하고, 상기 이벤트를 해결하고자 상기 실행 결과를 상기 제2 테스팅부에 제공하고, 상기 제2 테스팅부의 상기 동작의 결과를 상기 제1 테스팅부에 제공하는, 제어부를 더 포함할 수 있다.

[0011] 상기 제2 테스팅부는, 상기 퍼징이 실행되는 도중 시스템 호출(syscall)이 발생한 경우, 상기 시스템 모드 에뮬레이션 환경에서 상기 시스템 호출을 해결할 수 있다.

[0012] 상기 제어부는, 상기 시스템 호출의 해결 당시 상기 대상 프로세스의 메모리 상태를 저장하고, 상기 시스템 호출이 해결될 당시 상기 대상 프로세스의 메모리 상태를 상기 제1 테스팅부에 제공할 수 있다.

[0013] 상기 제1 테스팅부는, 상기 동작의 결과로서 상기 시스템 호출이 해결될 당시 상기 대상 프로세스의 메모리 상태에 기초하여 상기 퍼징의 실행을 재개할 수 있다.

[0014] 상기 제2 테스팅부는, 상기 퍼징이 실행되는 도중 기 설정된 시간 동안 제약 조건을 해결하지 못한 경우, 상기 시스템 모드 에뮬레이션 환경에서 동적 기호 실행(Concolic Execution)을 수행하여 상기 제약 조건을 해결할 수 있다.

[0015] 일 실시예에 따른 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치는, 상기 동적 기호 실행을 수행하여 제약 조건이 해결되는 경우, 상기 동적 기호 실행에 대응하는 제2 테스트 케이스를 생성하는 테스트 케이스 생성부를 더 포함할 수 있다.

[0016] 상기 제어부는, 상기 제2 테스트 케이스를 저장하고, 상기 제2 테스트 케이스를 상기 제1 테스팅부에 제공할 수 있다.

[0017] 상기 제1 테스팅부는, 상기 제2 테스트 케이스에 기초하여 상기 퍼징을 재개할 수 있다.

[0018] 일 실시예에 따른 하이브리드 퍼징 기반의 펌웨어 취약점 검출 방법은, 임의의 사물 인터넷(IoT; Internet of Things) 기기의 펌웨어에 시스템 모드 에뮬레이션 환경을 제공하는 단계; 상기 시스템 모드 에뮬레이션 환경에서 실행 중인 상기 펌웨어의 하나 이상의 프로세스 중 대상 프로세스에 사용자 모드 에뮬레이션 환경을 제공하는 단계; 상기 사용자 모드 에뮬레이션 환경에서 상기 대상 프로세스에 변이 기반 퍼징(Fuzzing)을 실행하는 단계; 상기 퍼징의 실행 중 기 설정된 이벤트가 발생한 경우, 상기 시스템 모드 에뮬레이션 환경에서 상기 이벤트의 발생 당시 상기 대상 프로세스의 메모리 상태에 기초하여 발생된 이벤트를 해결하기 위한 동작을 수행하는 단계; 및 상기 기 설정된 이벤트가 발생한 경우, 상기 퍼징의 실행을 중단하는 단계; 상기 기 설정된 이벤트가 발생한 경우, 동작의 결과에 기초하여 상기 퍼징의 실행을 재개하는 단계; 및 상기 퍼징이 재개된 결과에 기초하여 상기 펌웨어의 취약점을 검출하는 단계를 포함한다.

[0019] 일 실시예에 따른 하이브리드 퍼징 기반의 펌웨어 취약점 검출 방법은, 상기 이벤트가 발생할 당시 상기 대상 프로세스의 메모리 상태 및 상기 퍼징에 대응하는 제1 테스트 케이스 중 적어도 하나를 상기 퍼징의 실행 결과로서 저장하는 단계를 더 포함할 수 있다.

[0020] 상기 수행하는 단계는, 상기 이벤트를 해결하고자 상기 퍼징의 실행 결과에 기초하여 상기 이벤트를 해결하기 위한 동작을 수행하는 단계; 및 상기 퍼징을 통해 상기 펌웨어의 취약점을 검출하고자 상기 동작의 결과에 기초하여 상기 퍼징의 실행을 재개하는 단계를 포함할 수 있다.

[0021] 상기 수행하는 단계는, 상기 퍼징이 실행되는 도중 시스템 호출(syscall)이 발생하는 경우, 상기 시스템 모드 에뮬레이션 환경에서 상기 시스템 호출을 해결할 수 있다.

[0022] 일 실시예에 따른 하이브리드 퍼징 기반의 펌웨어 취약점 검출 방법은, 상기 시스템 호출이 해결된 경우, 상기 대상 프로세스의 메모리 상태를 저장하는 단계를 더 포함할 수 있다.

[0023] 상기 재개하는 단계는, 상기 시스템 호출을 해결한 상기 대상 프로세스의 메모리 상태에 기초하여 상기 퍼징을 재개할 수 있다.

[0024] 상기 수행하는 단계는, 상기 퍼징이 실행되는 도중 기 설정된 시간 동안 제약 조건을 해결하지 못한 경우, 상기 시스템 모드 에뮬레이션 환경에서 동적 기호 실행(Concolic Execution)을 수행하여 제약 조건을 할 수 있다.

[0025] 일 실시예에 따른 하이브리드 퍼징 기반의 펌웨어 취약점 검출 방법은, 상기 동적 기호 실행을 수행하여 제약 조건이 해결되는 경우, 상기 동적 기호 실행에 대응하는 제2 테스트 케이스를 생성하는 단계를 더 포함할 수 있다.

[0026] 일 실시예에 따른 하이브리드 퍼징 기반의 펌웨어 취약점 검출 방법은, 상기 동적 기호 실행을 수행하여 제약 조건이 해결되는 경우, 상기 제2 테스트 케이스를 저장하는 단계를 더 포함할 수있다.

[0027] 상기 수행하는 단계는, 상기 제2 테스트 케이스에 기초하여 상기 퍼징을 재개할 수 있다.

발명의 효과

[0028] 개시되는 실시예들에 따르면, 펌웨어에 대해 퍼징(Fuzzing)과 동적 기호 실행(Concolic Execution)을 수행함으로써, 펌웨어의 코드 커버리지를 넓힘과 동시에 넓은 코드 범위에서 펌웨어의 취약점을 검출할 수 있다.

[0029] 개시되는 실시예들에 따르면, 기 설정된 이벤트 발생 시에만 동적 기호 실행이 수행될 수 있도록 설정함으로써, 펌웨어 취약점의 검출 속도를 효율적으로 향상시킬 수 있다.

도면의 간단한 설명

[0030] 도 1은 일 실시예에 따른 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치를 설명하기 위한 블록도

도 2는 추가적인 실시예에 따른 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치를 설명하기 위한 블록도

도 3은 일 실시예에 따른 하이브리드 퍼징 기반의 펌웨어 취약점 검출 방법을 설명하기 위한 흐름도

도 4는 추가적인 실시예에 따른 하이브리드 퍼징 기반의 펌웨어 취약점 검출 방법을 설명하기 위한 흐름도

도 5는 일 실시예에 따른 컴퓨팅 장치를 포함하는 컴퓨팅 환경을 예시하여 설명하기 위한 블록도

발명을 실시하기 위한 구체적인 내용

[0031] 이하, 도면을 참조하여 일 실시예의 구체적인 실시형태를 설명하기로 한다. 이하의 상세한 설명은 본 명세서에서 기술된 방법, 장치 및/또는 시스템에 대한 포괄적인 이해를 돕기 위해 제공된다. 그러나 이는 예시에 불과하며 본 발명은 이에 제한되지 않는다.

[0032] 일 실시예들을 설명함에 있어서, 본 발명과 관련된 공지기술에 대한 구체적인 설명이 일 실시예의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우에는 그 상세한 설명을 생략하기로 한다. 그리고, 후술되는 용어들은 본 발명에서의 기능을 고려하여 정의된 용어들로서 이는 사용자, 운용자의 의도 또는 관례 등에 따라 달라질 수 있다. 그러므로 그 정의는 본 명세서 전반에 걸친 내용을 토대로 내려져야 할 것이다. 상세한 설명에서 사용되는 용어는 단지 일 실시예들을 기술하기 위한 것이며, 결코 제한적이어서는 안 된다. 명확하게 달리 사용되지 않는 한, 단수 형태의 표현은 복수 형태의 의미를 포함한다. 본 설명에서, "포함" 또는 "구비"와 같은 표현은 어떤 성분들, 숫자들, 단계들, 동작들, 요소들, 이들의 일부 또는 조합을 가리키기 위한 것이며, 기술된 것 이외에 하나 또는 그 이상의 다른 성분, 숫자, 단계, 동작, 요소, 이들의 일부 또는 조합의 존재 또는 가능성을 배제하도록 해석되어서는 안 된다.

[0033] 도 1은 일 실시예에 따른 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)를 설명하기 위한 블록도이다.

[0034] 도 1을 참조하면, 일 실시예에 따른 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)는 시스템 모드 에뮬레이터(110), 시스템 모드 에뮬레이터(120), 제1 테스팅부(130) 및 제2 테스팅부(140)를 포함한다.

[0035] 시스템 모드 에뮬레이터(110)는 임의의 사물 인터넷(IoT; Internet of Things) 기기의 펌웨어에 시스템 모드 에뮬레이션 환경을 제공한다.시스템 모드 에뮬레이터(120)는 시스템 모드 에뮬레이션 환경에서 실행 중인 펌웨어의 하나 이상의 프로세스 중 대상 프로세스에 사용자 모드 에뮬레이션 환경을 제공한다.

[0036] 이때, 에뮬레이터(emulator)란, 에뮬레이팅(emulating) 하는 장치를 의미한다. 에뮬레이팅이란, 에뮬레이션(emulation) 환경을 제공하는 것으로, 에뮬레이터란, 즉, 에뮬레이션 환경에서 일련의 프로세스를 실행하는 장치를 의미한다. 구체적으로, 에뮬레이터는 일련의 프로세스가 실행되던 원래 시스템을 다른 시스템으로 복제하여 일련의 프로세스를 다른 시스템 상에서 실행할 수 있다.

[0037] 즉, 시스템 모드 에뮬레이터(110)는 임의의 사물 인터넷 기기 전체에 에뮬레이팅(emulating)을 수행하는 시스템 모드 에뮬레이션 환경을 제공할 수 있다.

[0038] 시스템 모드 에뮬레이터(120)는 시스템 모드 에뮬레이터(110)를 통해 실행된 펌웨어의 하나 이상의 프로세스 중 대상 프로세스에 대응되는 메모리 파일에 기초하여 대상 프로세스에 대한 에뮬레이션 환경으로서, 시스템 모드 에뮬레이션 환경을 제공할 수 있다. 이때, 대상 프로세스는, 시스템 모드 에뮬레이터(110)를 통해 실행된 펌웨어의 하나 이상의 프로세스 중 임의의 프로세스로서, 변이 기반 퍼징 수행의 대상이 되는 프로세스를 지칭한다.

[0039] 제1 테스팅부(130)는, 사용자 모드 에뮬레이션 환경에서 대상 프로세스에 변이 기반 퍼징(Fuzzing)을 실행한다. 이때, 변이 기반 퍼징이란, 퍼징 수행 시 기 마련된 시드 파일(seed file)을 변형하여 테스트 케이스를 생성하는 퍼징 기법을 지칭한다.

[0040] 구체적으로, 일 실시예에 따르면, 제1 테스팅부(130)는, 퍼징 수행 시 기 마련된 시드 파일(seed file)을 변형하여 대상 프로세스의 포맷(format)을 만족하는 테스트 케이스를 생성할 수 있다.

[0041] 제2 테스팅부(140)는, 변이 기반 퍼징의 실행 중 기 설정된 이벤트가 발생된 경우, 시스템 모드 에뮬레이션 환경에서 이벤트의 발생 당시 대상 프로세스의 메모리 상태에 기초하여 발생된 이벤트를 해결하기 위한 동작을 수행한다.

[0042] 여기서, 기 설정된 이벤트가 발생한 경우란, 퍼징이 실행되는 도중 시스템 호출(syscall)이 발생하는 경우 및 퍼징이 실행되는 도중 기 설정된 시간 동안 제약 조건을 해결하지 못한 경우를 포함한다. 이때, 시스템 호출이란, 사용자 모드 에뮬레이션 환경에서 실행되는 프로세스 상 처리할 수 없는 호출을 의미한다.

[0043] 예를 들어, 발생한 이벤트가 변이 기반 퍼징이 실행되는 도중 시스템 호출이 발생하는 경우일 때, 구체적으로, 제2 테스팅부(140)는, 시스템 모드 에뮬레이션 환경에서 시스템 호출이 발생하는 당시 대상 프로세스의 메모리 상태에 기초하여 시스템 호출을 해결할 수 있다.

[0044] 다른 예로, 이벤트가 발생한 경우가 변이 기반 퍼징이 실행되는 도중 기 설정된 시간 동안 제약 조건을 해결하지 못한 경우일 때, 제2 테스팅부(140)는, 시스템 모드 에뮬레이션 환경에서 동적 기호 실행(Concolic Execution)을 수행하여 제약 조건을 해결할 수 있다.

[0045] 여기서, 동적 기호 실행이란, 실제 수행(concrete execution)과 기호 실행(symbolic execution)을 모두 이용하여 경로 제약조건(path constraint)을 해결하는 기법으로, 동적 분석 및 정적 분석 기법을 결합하여 테스트 케이스를 자동으로 생성한다.

[0046] 구체적으로, 동적 기호 실행은 테스트 케이스 생성 대상이 되는 프로그램과 초기 테스트 케이스를 제공 받아 초기 테스트 케이스를 실행한다. 이후, 실행된 프로그램의 경로를 분석하여 분기문에서 어떠한 조건이 수행되었는지를 나타내는 경로 제약 조건식을 생성한다. 다만, 프로그램의 모든 실행 경로를 테스트하거나, 사용자가 지정한 종료 조건을 만족하면, 동적 기호 실행은 종료된다. 이때, 동적 기호 실행이 종료될 경우, 생성된 경로 제약 조건식을 분석함으로써 제약 조건을 해결하는 테스트 케이스를 생성한다.

[0047] 즉, 일 실시예에 따르면, 제2 테스팅부(140)는, 변이 기반 퍼징에 기반하여 오랜 시간 동안 시드를 변이하더라도 변이 기반 퍼징이 실행되는 도중 기 설정된 시간 동안 제약 조건을 해결하지 못 했을 당시의 대상 프로세스의 메모리 상태에 기초하여 동적 기호 실행을 수행하여 대상 프로세스의 실행 경로를 분석한다.

[0048] 이후, 제2 테스팅부(140)는, 분기문에서 어떤 조건이 수행되었는지 나타내는 경로 제약 조건식을 생성한 후, 제약 조건식을 분석함으로써 동적 기호 실행이 종료될 때 생성된 경로 제약 조건식을 분석하여 제약 조건을 해결하는 새로운 테스트 케이스를 생성한다.

[0049] 결국, 제1 테스팅부(130)가 변이 기반 퍼징을 실행하는 도중 기 설정된 시간 동안 제약 조건을 해결하지 못한

경우, 제2 테스팅부(140)는 시스템 모드 에뮬레이션 환경에서 기 설정된 시간 동안 제약 조건을 해결하지 못 했을 당시의 대상 프로세스의 메모리 상태에 동적 기호 실행을 수행하여 제약 조건을 해결할 수 있다.

[0050] 이때, 제2 테스팅부(140)는, 임의의 사물 인터넷 기기 전체에 에뮬레이팅하는 시스템 모드 에뮬레이션 환경에서 동적 기호 실행을 수행하는 바, 제2 테스팅부(140)는, 하드웨어 종속성 함수의 실행이 가능하여 높은 정확도로 동적 기호 실행을 수행할 수 있다.

[0051] 다만, 제1 테스팅부(130)는, 기 설정된 이벤트가 발생한 경우, 퍼징의 실행을 중단하되, 제2 테스팅부(140)가 발생된 이벤트를 해결하기 위한 동작 수행을 완료한 경우, 동작의 결과에 기초하여 변이 기반 퍼징의 실행을 재개하여 펌웨어의 취약점을 검출한다.

[0052] 예를 들어, 변이 기반 퍼징이 실행되는 도중 시스템 호출이 발생하는 경우일 때, 제1 테스팅부(130)는, 제2 테스팅부(140)의 동작의 결과로서 상기 동작의 결과가 반영된 시스템 호출이 해결될 당시 대상 프로세스의 메모리 상태에 기초하여 변이 기반 퍼징의 실행을 재개할 수 있다.

[0053] 다른 예로, 변이 기반 퍼징이 실행되는 도중 기 설정된 시간 동안 제약 조건이 해결되지 못한 경우, 제1 테스팅부(130)는, 변이 기반 퍼징이 실행되는 도중 기 설정된 시간 동안 제약 조건을 해결하지 못 했을 당시의 대상 프로세스의 메모리 상태에 기초하여 변이 기반 퍼징의 실행을 재개할 수 있다.

[0054] 결국, 일 실시예에 따른 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)는 제1 테스팅부(130)가 해결하지 못하는 이벤트를 제2 테스팅부(140)로부터 해결하고, 제2 테스팅부(140)의 동작의 결과가 반영된 대상 프로세스의 메모리 상태에 기초하여 다시 제1 테스팅부(130)로부터 변이 기반 퍼징을 재개함으로써 변이 기반 퍼징의 수행 속도를 향상시킴과 동시에 코드 커버리지의 범위를 넓히는 효과를 발휘할 수 있다.

[0055] 도 2는 추가적인 실시예에 따른 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)를 설명하기 위한 블록도이다.

[0056] 도 2를 참조하면, 추가적인 실시예에 따른 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)는 제어부(150) 및 테스트 케이스 생성부(160)를 더 포함한다.

[0057] 한편, 설명의 편의를 위해 이하의 변이 기반 퍼징에 대응하는 테스트 케이스는 제1 테스트 케이스로 지칭하고, 제약 조건을 해결하여 제2 테스팅부(140)가 생성하는 새로운 테스트 케이스는 제2 테스트 케이스라고 지칭한다.

[0058] 일 실시예에 따르면, 제어부(150)는, 제1 테스팅부(130)의 변이 기반 퍼징 결과와 제2 테스팅부(140)의 동작의 결과가 서로 연계되어 사용될 수 있도록 제1 테스팅부(130)와 제2 테스팅부(140)를 하기와 같은 방식으로 제어한다.

[0059] 구체적으로, 제어부(150)는 이벤트가 발생할 당시 변이 기반 퍼징의 실행 결과로서 대상 프로세스의 메모리 상태 및 제1 테스트 케이스 중 적어도 하나를 저장할 수 있다.

[0060] 또한, 제어부(150)는 제1 테스팅부(130)가 해결할 수 없는 이벤트를 해결하고자 제1 테스팅부(130)의 변이 기반 퍼징의 실행 결과를 제2 테스팅부(140)에 제공할 수 있다.

[0061] 또한, 제어부(150)는, 제1 테스팅부(130)가 해결할 수 없는 이벤트를 해결하는 제2 테스팅부(140)의 동작의 결과가 반영된 대상 프로세스의 메모리 상태에 기초하여 변이 기반 퍼징이 제1 테스팅부(130)에서 재개될 수 있도록 제2 테스팅부(140)의 동작의 결과를 제1 테스팅부(130)에 제공할 수 있다.

[0062] 예를 들어, 변이 기반 퍼징이 실행되는 도중 시스템 호출이 발생하는 경우, 제어부(150)는 시스템 호출의 해결 당시 대상 프로세스의 메모리 상태를 저장할 수 있다. 또한, 제어부(150)는 제2 테스팅부(140)의 동작의 결과로서 시스템 호출이 해결될 당시 대상 프로세스의 메모리 상태를 제1 테스팅부(130)에 제공할 수 있다.

[0063] 다른 예로, 변이 기반 퍼징이 실행되는 도중 기 설정된 시간 동안 제1 테스팅부(130)가 제약 조건을 해결하지 못한 경우, 제어부(150)는 기 설정된 시간 동안 제약 조건을 찾지 못 했을 당시 대상 프로세스의 메모리 상태를 저장할 수 있다. 또한, 제어부(150)는, 제2 테스팅부(140)의 동작의 결과로서, 제약 조건을 해결하는 동적 기호 실행의 동작 의 결과를 제1 테스팅부(130)에 제공할 수 있다.

[0064] 이때, 테스트 케이스 생성부(160)는 제1 테스팅부(130)에 제공하기 위한 제2 테스트 케이스를 제약 조건을 해결하는 동적 기호 실행의 동작의 결과에 기초하여 생성할 수 있다.

[0065] 결국, 제어부(150)는 제2 테스트 케이스를 제1 테스팅부(130)에 제공하고자, 제2 테스트 케이스를 저장할 수 있

으며, 저장된 제2 테스트 케이스를 제1 테스팅부(130)에 제공할 수 있다.

[0066] 도 3은 일 실시예에 따른 하이브리드 퍼징 기반의 펌웨어 취약점 검출 방법을 설명하기 위한 흐름도이다.

[0067] 도 3에 도시된 방법은 도 1에서 도시된 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)에 의해 수행될 수 있다.

[0068] 도 3을 참조하면, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)는 임의의 사물 인터넷(IoT; Internet of Things) 기기의 펌웨어에 시스템 모드 에뮬레이션 환경을 제공한다(310).

[0069] 이후, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)는 시스템 모드 에뮬레이션 환경에서 실행 중인 펌웨어의 하나 이상의 프로세스 중 대상 프로세스에 사용자 모드 에뮬레이션 환경을 제공한다(320).

[0070] 이후, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)는 사용자 모드 에뮬레이션 환경에서 대상 프로세스에 변이 기반 퍼징(Fuzzing)을 실행한다(330).

[0071] 이후, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)는 퍼징의 실행 중 기 설정된 이벤트가 발생 여부를 판단한다(340).

[0072] 이후, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)는 시스템 모드 에뮬레이션 환경에서 이벤트의 발생 당시 대상 프로세스의 상태에 기초하여 발생된 이벤트를 해결하기 위한 동작을 수행한다(350).

[0073] 이후, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)는 기 설정된 이벤트가 발생한 경우, 퍼징의 실행을 중단한다(360).

[0074] 이후, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)는 기 설정된 이벤트가 발생한 경우, 동작의 결과에 기초하여 퍼징의 실행을 재개한다(370).

[0075] 이후, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)는 퍼징의 실행 결과에 기초하여 펌웨어의 취약점을 검출한다(380).

[0076] 도 4는 추가적인 실시예에 따른 하이브리드 퍼징 기반의 펌웨어 취약점 검출 방법을 설명하기 위한 흐름도이다.

[0077] 도 4에 도시된 방법은 도 1에서 도시된 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)에 의해 수행될 수 있다.

[0078] 도 4를 참조하면, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)는 임의의 사물 인터넷(IoT; Internet of Things) 기기의 펌웨어에 시스템 모드 에뮬레이션 환경을 제공한다(410).

[0079] 이후, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)는 시스템 모드 에뮬레이션 환경에서 실행 중인 펌웨어의 하나 이상의 프로세스 중 대상 프로세스에 사용자 모드 에뮬레이션 환경을 제공한다(420).

[0080] 이후, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)는 사용자 모드 에뮬레이션 환경에서 대상 프로세스에 변이 기반 퍼징(Fuzzing)을 실행한다(430).

[0081] 이후, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)는 퍼징의 실행 도중 시스템 호출이 발생되었는지 여부를 판단한다(440).

[0082] 이후, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)가 시스템 호출이 발생되었다고 판단한 경우, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)는 시스템 모드 에뮬레이션 환경에서 시스템 호출을 해결한다(441).

[0083] 이후, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)는 시스템 호출을 해결한 대상 프로세스의 메모리 상태를 저장한다(442).

[0084] 이후, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)는 시스템 호출을 해결한 대상 프로세스의 메모리 상태에 기초하여 퍼징을 재개한다(443).

[0085] 이후, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)는 퍼징이 재개된 결과에 기초하여 펌웨어의 취약점을 검출한다(460).

[0086] 반면, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)는 시스템 호출이 발생되지 않았다고 판단한 경우, 퍼징 기반의 펌웨어 취약점 검출 장치(100)는 퍼징의 수행 시간이 기 설정된 시간을 초과하였는지 여부를 판단

한다(450).

[0087] 이후, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)가 퍼징의 수행 시간이 기 설정된 시간을 초과하였다고 판단한 경우, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)는 시스템 모드 에뮬레이션 환경에서 동적 기호 실행을 수행하여 제약 조건을 해결한다(451).

[0088] 이후, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)는 동적 기호 실행을 수행하여 제약 조건이 해결되는 경우, 동적 기호 실행에 대응하는 제2 테스트 케이스를 생성한다(452).

[0089] 이후, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)는 동적 기호 실행을 수행하여 제약 조건이 해결되는 경우, 제2 테스트 케이스를 저장한다(453).

[0090] 이후, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)는 제2 테스트 케이스에 기초하여 퍼징을 재개한다(454).

[0091] 이후, 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)는 퍼징이 재개된 결과에 기초하여 펌웨어의 취약점을 검출한다(460).

[0092] 도 3 및 도4에 도시된 실시예는 복수 개의 단계로 나누어 기재되었으나, 적어도 일부의 단계들은 순서를 바꾸어 수행되거나, 다른 단계와 결합되어 함께 수행되거나, 생략되거나, 세부 단계들로 나뉘어 수행되거나, 또는 도시되지 않은 하나 이상의 단계가 부가되어 수행될 수 있다.

[0093] 도 5는 일 실시예에 따른 컴퓨팅 장치를 포함하는 컴퓨팅 환경을 예시하여 설명하기 위한 블록도

[0094] 도 5는 일 실시예에 따르면 컴퓨팅 장치(12)를 포함하는 컴퓨팅 환경(10)을 예시하여 설명하기 위한 블록도이다. 도시된 실시예에서, 각 컴포넌트들은 이하에 기술된 것 이외에 상이한 기능 및 능력을 가질 수 있고, 이하에 기술되지 않은 것 이외에도 추가적인 컴포넌트를 포함할 수 있다.

[0095] 도시된 컴퓨팅 환경(10)은 컴퓨팅 장치(12)를 포함한다. 일 실시예에서, 컴퓨팅 장치(12)는 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치(100)에 포함된 하나 이상의 컴포넌트일 수 있다.

[0096] 컴퓨팅 장치(12)는 적어도 하나의 프로세스(14), 컴퓨터 판독 가능 저장 매체(16) 및 통신 버스(18)를 포함한다. 프로세스(14)는 컴퓨팅 장치(12)로 하여금 앞서 언급된 예시적인 실시예에 따라 동작하도록 할 수 있다. 예컨대, 프로세스(14)는 컴퓨터 판독 가능 저장 매체(16)에 저장된 하나 이상의 프로그램들을 실행할 수 있다. 상기 하나 이상의 프로그램들은 하나 이상의 컴퓨터 실행 가능 명령어를 포함할 수 있으며, 상기 컴퓨터 실행 가능 명령어는 프로세스(14)에 의해 실행되는 경우 컴퓨팅 장치(12)로 하여금 예시적인 실시예에 따른 동작들을 수행하도록 구성될 수 있다.

[0097] 컴퓨터 판독 가능 저장 매체(16)는 컴퓨터 실행 가능 명령어 내지 프로그램 코드, 프로그램 데이터 및/또는 다른 적합한 형태의 정보를 저장하도록 구성된다. 컴퓨터 판독 가능 저장 매체(16)에 저장된 프로그램(20)은 프로세스(14)에 의해 실행 가능한 명령어의 집합을 포함한다. 일 실시예에서, 컴퓨터 판독 가능 저장 매체(16)는 메모리(랜덤 액세스 메모리와 같은 휘발성 메모리, 비휘발성 메모리, 또는 이들의 적절한 조합), 하나 이상의 자기 디스크 저장 디바이스들, 광학 디스크 저장 디바이스들, 플래시 메모리 디바이스들, 그 밖에 컴퓨팅 장치(12)에 의해 액세스되고 원하는 정보를 저장할 수 있는 다른 형태의 저장 매체, 또는 이들의 적합한 조합일 수 있다.

[0098] 통신 버스(18)는 프로세스(14), 컴퓨터 판독 가능 저장 매체(16)를 포함하여 컴퓨팅 장치(12)의 다른 다양한 컴포넌트들을 상호 연결한다.

[0099] 컴퓨팅 장치(12)는 또한 하나 이상의 입출력 장치(24)를 위한 인터페이스를 제공하는 하나 이상의 입출력 인터페이스(22) 및 하나 이상의 네트워크 통신 인터페이스(26)를 포함할 수 있다. 입출력 인터페이스(22) 및 네트워크 통신 인터페이스(26)는 통신 버스(18)에 연결된다. 입출력 장치(24)는 입출력 인터페이스(22)를 통해 컴퓨팅 장치(12)의 다른 컴포넌트들에 연결될 수 있다. 예시적인 입출력 장치(24)는 포인팅 장치(마우스 또는 트랙패드 등), 키보드, 터치 입력 장치(터치패드 또는 터치스크린 등), 음성 또는 소리 입력 장치, 다양한 종류의 센서 장치 및/또는 촬영 장치와 같은 입력 장치, 및/또는 디스플레이 장치, 프린터, 스피커 및/또는 네트워크 카드와 같은 출력 장치를 포함할 수 있다. 예시적인 입출력 장치(24)는 컴퓨팅 장치(12)를 구성하는 일 컴포넌트로서 컴퓨팅 장치(12)의 내부에 포함될 수도 있고, 컴퓨팅 장치(12)와는 구별되는 별개의 장치로 컴퓨팅 장치(12)와 연결될 수도 있다.

[0100] 이상에서 대표적인 실시예를 통하여 본 발명에 대하여 상세하게 설명하였으나, 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자는 전술한 실시예에 대하여 본 발명의 범주에서 벗어나지 않는 한도 내에서 다양한 변형이 가능함을 이해할 것이다. 그러므로 본 발명의 권리범위는 설명된 실시예에 국한되어 정해져서는 안 되며, 후술하는 청구범위뿐만 아니라 이 청구범위와 균등한 것들에 의해 정해져야 한다.

부호의 설명

[0101] 10: 컴퓨팅 환경

12: 컴퓨팅 장치

14: 프로세스

16: 컴퓨터 판독 가능 저장 매체

18: 통신 버스

20: 프로그램

22: 입출력 인터페이스

24: 입출력 장치

26: 네트워크 통신 인터페이스

100: 하이브리드 퍼징 기반의 펌웨어 취약점 검출 장치

110: 시스템 모드 에뮬레이터

120: 사용자 모드 에뮬레이터

130: 제1 테스팅부

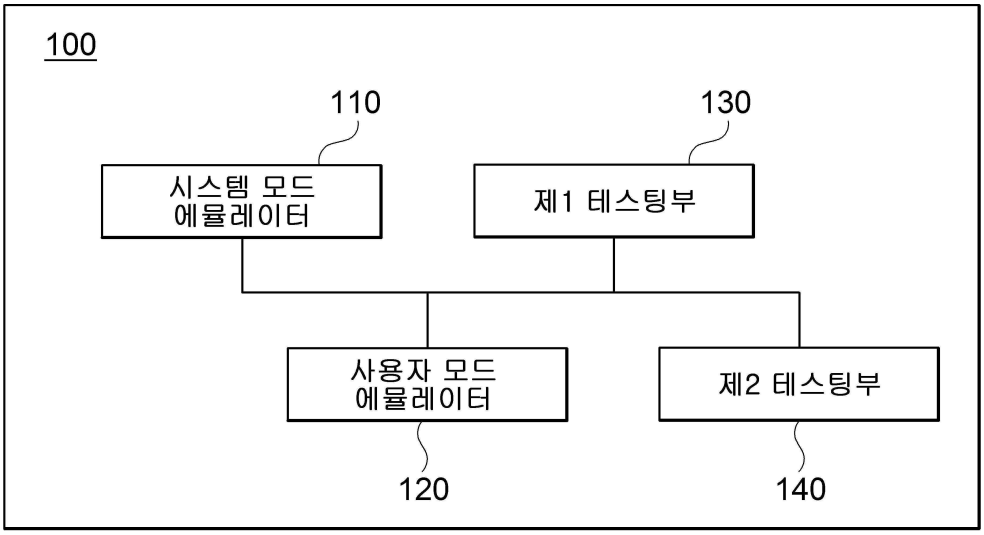
140: 제2 테스팅부

150: 제어부

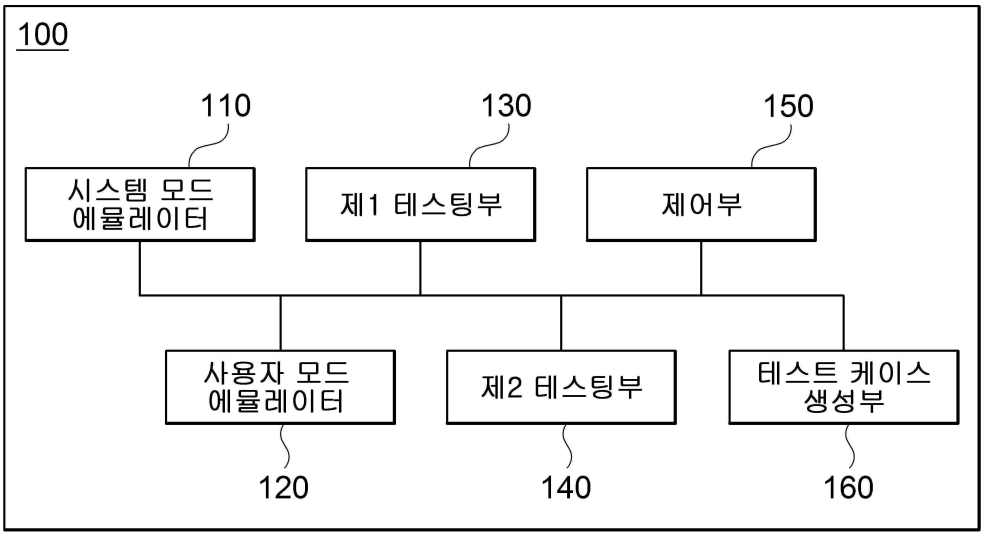
160: 테스트 케이스 생성부

도면

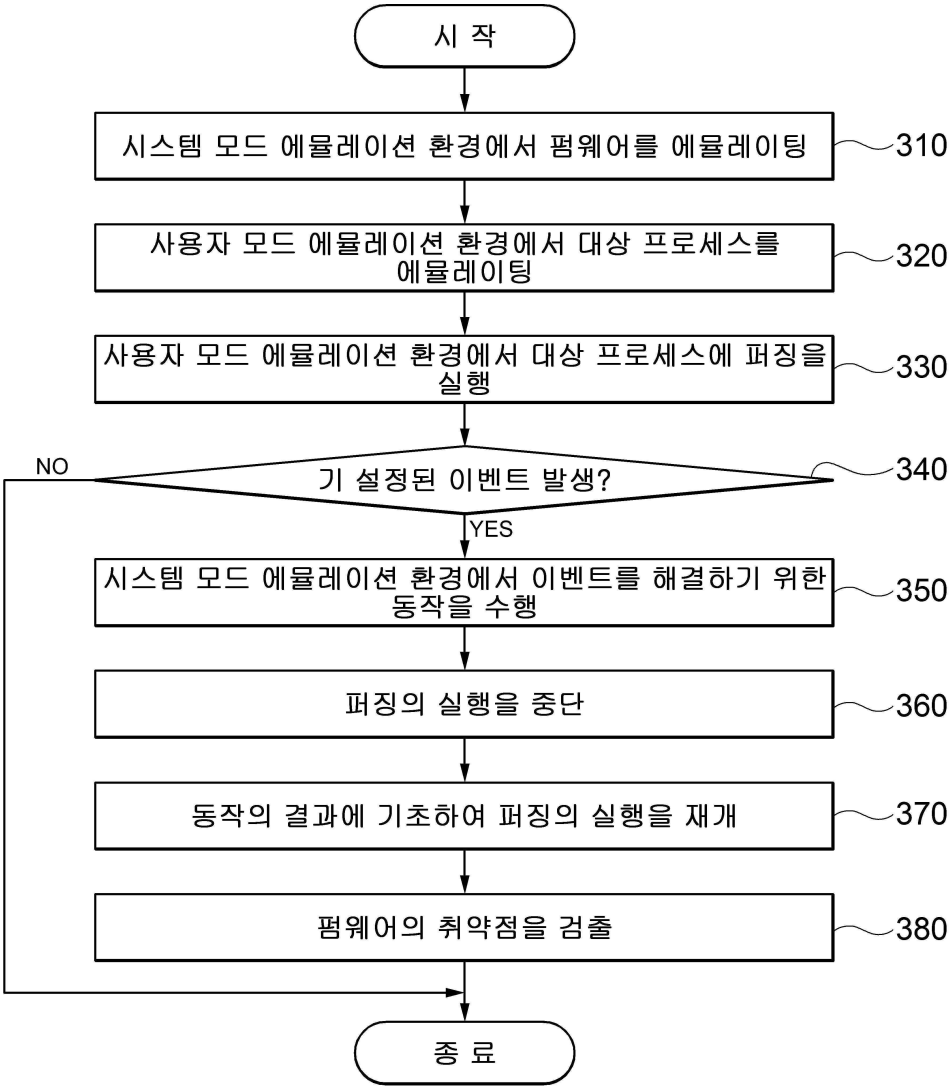
도면1



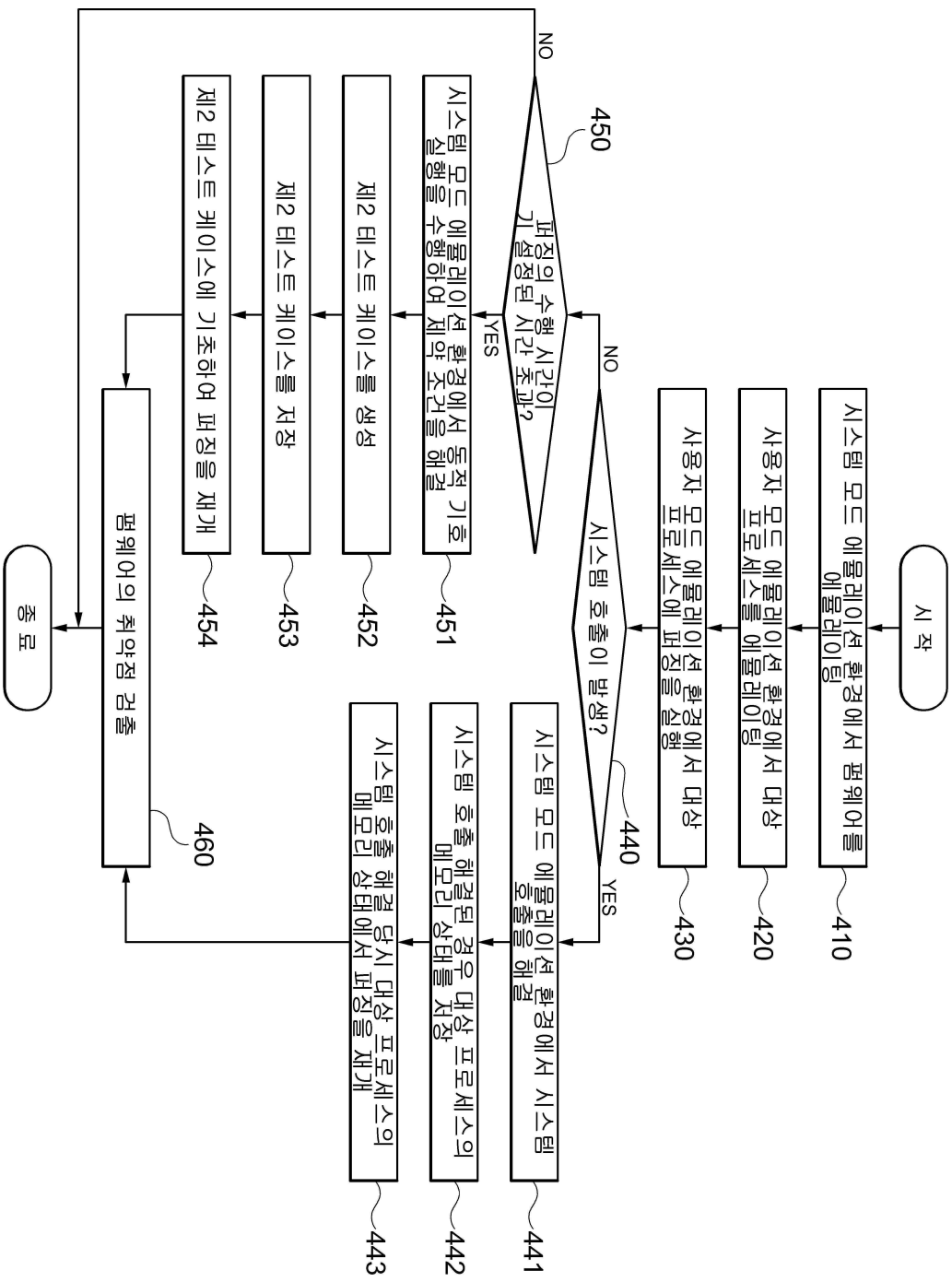
도면2


100
110
130
150
시스템 모드 에뮬레이터
제1 테스팅부
제어부
사용자 모드 에뮬레이터
제2 테스팅부
테스트 케이스 생성부
120
140
160

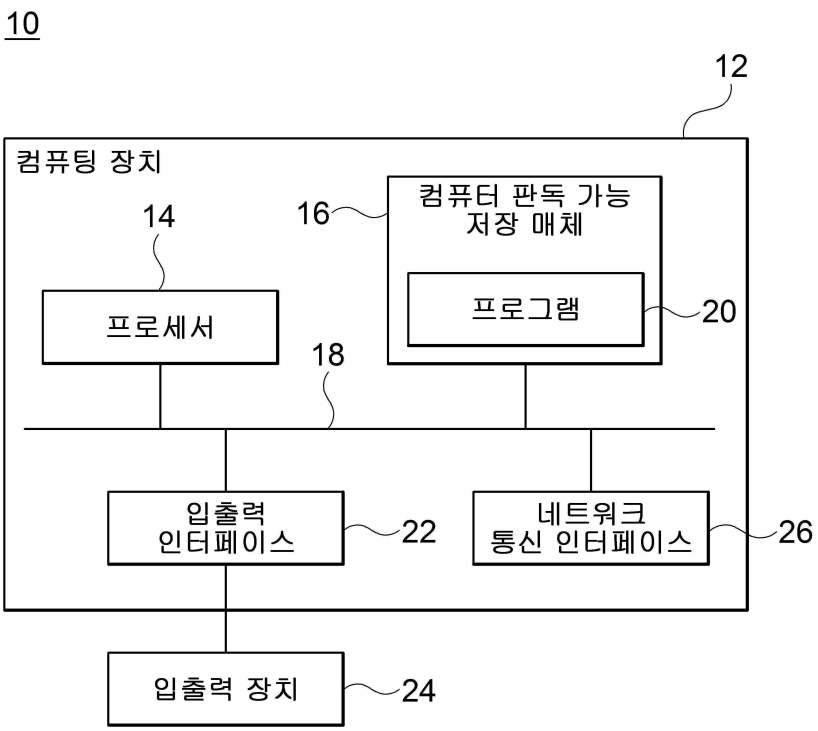
도면3


시 작
시스템 모드 에뮬레이션 환경에서 펌웨어를 에뮬레이팅
310
사용자 모드 에뮬레이션 환경에서 대상 프로세스를 에뮬레이팅
320
사용자 모드 에뮬레이션 환경에서 대상 프로세스에 퍼징을 실행
330
NO
기 설정된 이벤트 발생?
340
YES
시스템 모드 에뮬레이션 환경에서 이벤트를 해결하기 위한 동작을 수행
350
퍼징의 실행을 중단
360
동작의 결과에 기초하여 퍼징의 실행을 재개
370
펌웨어의 취약점을 검출
380
종 료

도면4


시 작
시스템 모드 에뮬레이션 환경에서 펌웨어를 에뮬레이팅
410
사용자 모드 에뮬레이션 환경에서 대상 프로세스를 에뮬레이팅
420
사용자 모드 에뮬레이션 환경에서 대상 프로세스에 퍼징을 실행
430
시스템 호출이 발생?
440
YES
NO
시스템 모드 에뮬레이션 환경에서 시스템 호출을 해결
441
시스템 호출 해결된 경우 대상 프로세스의 메모리 상태를 저장
442
시스템 호출 해결 당시 대상 프로세스의 메모리 상태에서 퍼징을 재개
443
퍼징의 수행 시간이 기 설정된 시간 초과?
450
YES
NO
시스템 모드 에뮬레이션 환경에서 동적 기호 실행을 수행하여 제약 조건을 해결
451
제2 테스트 케이스를 생성
452
제2 테스트 케이스를 저장
453
제2 테스트 케이스에 기초하여 퍼징을 재개
454
펌웨어의 취약점 검출
460
종 료

도면5


10
12
컴퓨팅 장치
14
프로세서
16
컴퓨터 판독 가능
저장 매체
프로그램
20
18
입출력
인터페이스
22
네트워크
통신 인터페이스
26
입출력 장치
24