(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2024-0011475
(43) 공개일자 2024년01월26일

(51) 국제특허분류(Int. Cl.)
G06F 21/57 (2013.01) *G06F 11/36* (2006.01)
G16Y 30/10 (2020.01)
(52) CPC특허분류
G06F 21/577 (2013.01)
G06F 11/3688 (2013.01)
(21) 출원번호 10-2022-0088957
(22) 출원일자 2022년07월19일
심사청구일자 2022년07월19일

(71) 출원인
세종대학교산학협력단
서울특별시 광진구 능동로 209 (군자동, 세종대학교)
(72) 발명자
윤주범
서울특별시 송파구 충민로4길 19, 704동 401호(장지동, 송파파인타운7단지)
유지현
서울특별시 광진구 광나루로28길 18, 302호(화양동)
(뒷면에 계속)
(74) 대리인
특허법인티앤씨

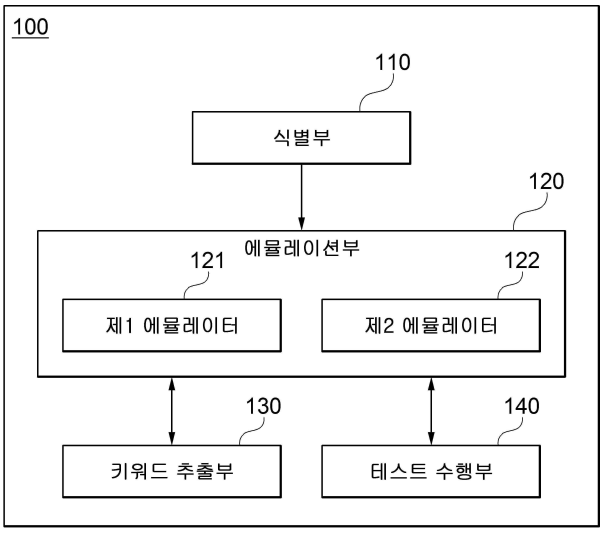
전체 청구항 수 : 총 20 항

(54) 발명의 명칭 **적응형 에뮬레이션 기법을 이용한 IoT 펌웨어 취약점 점검 방법 및 장치**

(57) 요 약

일 실시예에 따른 IoT 펌웨어 취약점 점검 방법은, IoT(Internet of Things) 장치의 펌웨어에 대한 펌웨어 아키텍처를 식별하는 단계; 상기 펌웨어 아키텍처에 기초하여 제1 에뮬레이터 및 제2 에뮬레이터 중 하나를 선택하는 단계; 상기 선택된 에뮬레이터를 이용하여 상기 펌웨어를 에뮬레이션하는 단계; 상기 에뮬레이션된 펌웨어의 실행 경로를 분석하여 하나 이상의 키워드를 추출하는 단계; 상기 하나 이상의 키워드에 기초하여 하나 이상의 테스트 케이스를 생성하는 단계; 및 상기 하나 이상의 테스트 케이스를 이용하여 상기 에뮬레이션된 펌웨어에 대한 퍼징 테스트를 수행하는 단계를 포함한다.

대 표 도 - 도1



(52) CPC특허분류
G06F 11/3696 (2013.01)
G06F 21/572 (2013.01)
G06F 9/4411 (2013.01)
G06F 9/45508 (2013.01)
G16Y 30/10 (2020.01)
(72) 발명자
김주환
서울특별시 광진구 군자로 175-2, 304호(군자동)

이영우
경기도 구리시 경춘로288번길 39, 마동 410호(수택동)

이 발명을 지원한 국가연구개발사업

과제고유번호	1711159975
과제번호	2018-0-01423-005
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기획평가원
연구사업명	정보통신방송혁신인재양성
연구과제명	지능형 비행로봇 융합기술 연구
기 여 율	1/1
과제수행기관명	세종대학교 산학협력단
연구기간	2018.06.01 ~ 2023.12.31

명 세 서

청구범위

청구항 1

IoT(Internet of Things) 장치의 펌웨어에 대한 펌웨어 아키텍처를 식별하는 단계;

상기 펌웨어 아키텍처에 기초하여 제1 에뮬레이터 및 제2 에뮬레이터 중 하나를 선택하는 단계;

상기 선택된 에뮬레이터를 이용하여 상기 펌웨어를 에뮬레이션하는 단계;

상기 에뮬레이션된 펌웨어의 실행 경로를 분석하여 하나 이상의 키워드를 추출하는 단계;

상기 하나 이상의 키워드에 기초하여 하나 이상의 테스트 케이스를 생성하는 단계; 및

상기 하나 이상의 테스트 케이스를 이용하여 상기 에뮬레이션된 펌웨어에 대한 퍼징 테스트를 수행하는 단계를 포함하는, IoT 펌웨어 취약점 점검 방법.

청구항 2

청구항 1에 있어서,

상기 제1 에뮬레이터는, 제1 아키텍처 그룹에 포함된 하나 이상의 펌웨어 아키텍처에 대한 에뮬레이션 환경을 제공하는 에뮬레이터이고,

상기 제2 에뮬레이터는, 제2 아키텍처 그룹에 포함된 하나 이상의 펌웨어 아키텍처에 대한 에뮬레이션 환경을 제공하는 에뮬레이터인, IoT 펌웨어 취약점 점검 방법.

청구항 3

청구항 1에 있어서,

상기 제1 에뮬레이터는, 경험적 근거에 기반한 중재 기법이 적용된 시스템 모드 에뮬레이션을 제공하는, IoT 펌웨어 취약점 점검 방법.

청구항 4

청구항 3에 있어서,

상기 제1 에뮬레이터는, 상기 시스템 모드 에뮬레이션 및 사용자 모드 에뮬레이션을 제공하는, IoT 펌웨어 취약점 점검 방법.

청구항 5

청구항 1에 있어서,

상기 퍼징 테스트를 수행하는 단계는, 상기 제2 에뮬레이터에 의해 에뮬레이션된 상기 펌웨어에 대한 상기 퍼징 테스트 수행 시 상기 펌웨어의 실행 상태에 기초하여 상기 퍼징 테스트 중 발생한 오류를 탐지하고 상기 탐지된 오류에 대한 메시지를 제공하는 단계를 더 포함하는, IoT 펌웨어 취약점 점검 방법.

청구항 6

청구항 1에 있어서,

상기 제2 에뮬레이터는, 사전 구비된 자동화 구성 스크립트에 기초하여 상기 펌웨어를 에뮬레이션하는, IoT 펌웨어 취약점 점검 방법.

청구항 7

청구항 1에 있어서,

상기 하나 이상의 키워드를 추출하는 단계는, 상기 에뮬레이션된 펌웨어에 대한 동적 기호 실행(concolic execution)을 수행하는 단계; 및

상기 동적 기호 실행 결과에 기초하여 상기 하나 이상의 키워드를 추출하는 단계를 포함하는, IoT 펌웨어 취약점 점검 방법.

청구항 8

청구항 7에 있어서,

상기 하나 이상의 키워드를 추출하는 단계는, 상기 실행 경로 상의 하나 이상의 분기에 대한 경로 제약 조건(path constraint)을 해결하는 하나 이상의 키워드를 추출하는, IoT 펌웨어 취약점 점검 방법.

청구항 9

청구항 1에 있어서,

상기 하나 이상의 테스트 케이스를 생성하는 단계는, 상기 하나 이상의 키워드를 참조하여 시드 데이터(seed data)를 변형함으로써 상기 하나 이상의 테스트 케이스를 생성하는, IoT 펌웨어 취약점 점검 방법.

청구항 10

청구항 9에 있어서,

상기 하나 이상의 테스트 케이스를 생성하는 단계는, 상기 시드 데이터를 변형하여 상기 하나 이상의 테스트 케이스를 생성하되, 상기 하나 이상의 테스트 케이스 각각이 상기 하나 이상의 키워드 중 적어도 하나를 포함하도록 상기 시드 데이터를 변형하는, IoT 펌웨어 취약점 점검 방법.

청구항 11

IoT(Internet of Things) 장치의 펌웨어에 대한 펌웨어 아키텍처를 식별하는 식별부;

상기 펌웨어 아키텍처에 기초하여 제1 에뮬레이터 및 제2 에뮬레이터 중 하나를 선택하고, 상기 선택된 에뮬레이터를 이용하여 상기 펌웨어를 에뮬레이션하는 에뮬레이션부;

상기 에뮬레이션된 펌웨어의 실행 경로를 분석하여 하나 이상의 키워드를 추출하는 키워드 추출부; 및

상기 하나 이상의 키워드에 기초하여 하나 이상의 테스트 케이스를 생성하고, 상기 하나 이상의 테스트 케이스를 이용하여 상기 에뮬레이션된 펌웨어에 대한 퍼징 테스트를 수행하는 테스트 수행부를 포함하는, IoT 펌웨어 취약점 점검 장치.

청구항 12

청구항 11에 있어서,

상기 제1 에뮬레이터는, 제1 아키텍처 그룹에 포함된 하나 이상의 펌웨어 아키텍처에 대한 에뮬레이션 환경을 제공하는 에뮬레이터이고,

상기 제2 에뮬레이터는, 제2 아키텍처 그룹에 포함된 하나 이상의 펌웨어 아키텍처에 대한 에뮬레이션 환경을 제공하는 에뮬레이터인, IoT 펌웨어 취약점 점검 장치.

청구항 13

청구항 11에 있어서,

상기 제1 에뮬레이터는, 경험적 근거에 기반한 중재 기법이 적용된 시스템 모드 에뮬레이션을 제공하는, IoT 펌웨어 취약점 점검 장치.

청구항 14

청구항 13에 있어서,

상기 제1 에뮬레이터는, 상기 시스템 모드 에뮬레이션 및 사용자 모드 에뮬레이션을 제공하는, IoT 펌웨어 취약점 점검 장치.

청구항 15

청구항 11에 있어서,

상기 에뮬레이션부는, 상기 제2 에뮬레이터에 의해 에뮬레이션된 상기 펌웨어에 대한 상기 퍼징 테스트 수행 시 상기 펌웨어의 실행 상태에 기초하여 상기 퍼징 테스트 수행 중 발생한 오류를 탐지하고 상기 탐지된 오류에 대한 메시지를 상기 테스트 수행부로 제공하는 상호 연결 모듈을 포함하는, IoT 펌웨어 취약점 점검 장치.

청구항 16

청구항 11에 있어서,

상기 제2 에뮬레이터는, 사전 구비된 자동화 스크립트에 기초하여 상기 펌웨어를 에뮬레이션하는, IoT 펌웨어 취약점 점검 장치.

청구항 17

청구항 11에 있어서,

상기 키워드 추출부는, 상기 에뮬레이션된 펌웨어에 대한 동적 기호 실행(concolic execution)을 수행하고, 상기 동적 기호 실행 결과에 기초하여 상기 하나 이상의 키워드를 추출하는, IoT 펌웨어 취약점 점검 장치.

청구항 18

청구항 17에 있어서,

상기 키워드 추출부는, 상기 실행 경로 상의 하나 이상의 분기에 대한 경로 제약 조건(path constraint)을 해결하는 하나 이상의 키워드를 추출하는, IoT 펌웨어 취약점 점검 장치.

청구항 19

청구항 11에 있어서,

상기 테스트 수행부는, 상기 하나 이상의 키워드를 참조하여 시드 데이터(seed data)를 변형함으로써 상기 하나 이상의 테스트 케이스를 생성하는, IoT 펌웨어 취약점 점검 장치.

청구항 20

청구항 19에 있어서,

상기 테스트 수행부는, 상기 시드 데이터를 변형하여 상기 하나 이상의 테스트 케이스를 생성하되, 상기 하나 이상의 테스트 케이스 각각이 상기 하나 이상의 키워드 중 적어도 하나를 포함하도록 상기 시드 데이터를 변형하는, IoT 펌웨어 취약점 점검 장치.

발명의 설명

기 술 분 야

[0001] 본 발명의 실시예들은 펌웨어에 대한 취약점 탐지 기술과 관련된다.

배 경 기 술

[0002] IoT(Internet of Things) 장치의 수요는 해가 갈수록 증가하고 있다. 그에 따라, 성능 최적화, 비용, 크기 등의 문제로 장치의 구성 또한 나날이 다양해지고 있다. 하지만 IoT 장치의 보안 수준은 제조업체들의 장치 개발 및 출시일 경쟁에 밀려 낮은 수준에 머물러 있는데, 공격자에게 있어 보안 수준은 낮으나 다양하고 수가 많은 임베디드 장치 분야는 최적의 공격 대상이다.

[0003] 이에 대응하기 위해, 최근 연구들은 IoT 장치의 펌웨어를 자동으로 테스트하는 시스템에 초점을 맞추고 있으며, 특히 이러한 연구들 중 일부는 가상 에뮬레이션 환경에서 펌웨어를 실행한 후 분석 기법을 적용하는 방법을 제안하고 있다. 이 방법은 실제 장치 없이도 적은 자원으로 다양한 펌웨어를 구동할 수 있어 대규모의 펌웨어 분석을 가능하게 한다. 또한, 많은 펌웨어가 기존 데스크탑 소프트웨어와 유사한 코드를 내포하고 있기 때문에 에뮬레이션 환경 내의 펌웨어에 기존 소프트웨어 테스트 기술을 적용하는 것이 용이하다.

[0004] 그러나, 기존 IoT 장치 펌웨어 점검을 위한 에뮬레이션 시스템에는 크게 3가지 문제점이 존재한다.

[0005] 첫째, IoT 장치는 성능 최적화와 비용상의 이유로 장치 구성 및 아키텍처가 점점 다양해지는 데 반해, 기존 에뮬레이션 연구들은 MIPS, ARM 등과 같이 주류적인 펌웨어 아키텍쳐에만 초점이 맞추어져 있다. 기존 연구 중 QEMU의 경우 몇몇 비주류 펌웨어 에뮬레이션을 지원하지만 매뉴얼일 뿐, 자동화된 시스템 형태로 제공하지 않는다.

[0006] 둘째, 기존 에뮬레이션 시스템들은 완벽한 재연을 목적으로 하기 때문에 에뮬레이션 성공률이 낮다. 펌웨어의 설정 및 버전은 각 장치마다 다르고 이를 모두 수용하기 위해서는 각 장치의 모든 물리적 기능까지 프로그램상으로 재연되어야 한다, 이러한 작업은 많은 인력이 필요하며, 대규모 분석 시스템에서는 사실상 불가능에 가깝다.

[0007] 셋째, 대부분의 IoT 장치는 입출력에 형식이 존재하기 때문에 동적 분석이 어렵다. IoT 장치는 네트워크를 통해 패킷과 같은 입출력 형식으로 호스트와 통신한다. 실행 중인 장치를 면밀히 점검하기 위해서는 대상 프로그램의 모든 경로에 접근하는 것이 이상적인데, 입력이 특정 형식을 가진다면 매우 얕은 경로에서 장시간 필터링될 가능성이 높다.

[0008] 이를 해결하기 위해 IoT 장치에 대한 비주류 펌웨어 아키텍처 지원, 높은 가용성, 자동화된 입력 형식 구성으로 분석의 효율성을 높이는 시스템이 필요한 상황이다.

선행기술문헌

특허문헌

[0009] (특허문헌 0001) 대한민국 공개특허공보 제10-2019-0114158호 (2019. 11. 10. 공개)

발명의 내용

해결하려는 과제

[0010] 본 발명의 실시예들은 펌웨어 취약점 탐지를 위한 IoT 펌웨어 취약점 점검 방법 및 장치를 제공하기 위한 것이다.

과제의 해결 수단

[0011] 일 실시예에 따른 IoT 펌웨어 취약점 점검 방법은, IoT 장치의 펌웨어에 대한 펌웨어 아키텍처를 식별하는 단계; 상기 펌웨어 아키텍처에 기초하여 제1 에뮬레이터 및 제2 에뮬레이터 중 하나를 선택하는 단계; 상기 선택된 에뮬레이터를 이용하여 상기 펌웨어를 에뮬레이션하는 단계; 상기 에뮬레이션된 펌웨어의 실행 경로를 분석하여 하나 이상의 키워드를 추출하는 단계; 상기 하나 이상의 키워드에 기초하여 하나 이상의 테스트 케이스를 생성하는 단계; 및 상기 하나 이상의 테스트 케이스를 이용하여 상기 에뮬레이션된 펌웨어에 대한 퍼징 테스트를 수행하는 단계를 포함한다.

[0012] 상기 제1 에뮬레이터는, 제1 아키텍처 그룹에 포함된 하나 이상의 펌웨어 아키텍처에 대한 에뮬레이션 환경을 제공하는 에뮬레이터이고, 상기 제2 에뮬레이터는, 제2 아키텍처 그룹에 포함된 하나 이상의 펌웨어 아키텍처에 대한 에뮬레이션 환경을 제공하는 에뮬레이터일 수 있다.

[0013] 상기 제1 에뮬레이터는, 경험적 근거에 기반한 중재 기법이 적용된 시스템 모드 에뮬레이션을 제공할 수 있다.

[0014] 상기 제1 에뮬레이터는, 상기 시스템 모드 에뮬레이션 및 사용자 모드 에뮬레이션을 제공할 수 있다.

[0015] 상기 퍼징 테스트를 수행하는 단계는, 상기 제2 에뮬레이터에 의해 에뮬레이션된 상기 펌웨어에 대한 상기 퍼징 테스트 수행 시 상기 펌웨어의 실행 상태에 기초하여 상기 퍼징 테스트 중 발생한 오류를 탐지하고 상기 탐지된 오류에 대한 메시지를 제공하는 단계를 더 포함할 수 있다.

[0016] 상기 제2 에뮬레이터는, 사전 구비된 자동화 구성 스크립트에 기초하여 상기 펌웨어를 에뮬레이션할 수 있다.

[0017] 상기 하나 이상의 키워드를 추출하는 단계는, 상기 에뮬레이션된 펌웨어에 대한 동적 기호 실행(concolic execution)을 수행하는 단계; 및 상기 동적 기호 실행 결과에 기초하여 상기 하나 이상의 키워드를 추출하는 단계를 포함할 수 있다.

[0018] 상기 하나 이상의 키워드를 추출하는 단계는, 상기 실행 경로 상의 하나 이상의 분기에 대한 경로 제약 조건(path constraint)을 해결하는 하나 이상의 키워드를 추출할 수 있다.

[0019] 상기 하나 이상의 테스트 케이스를 생성하는 단계는, 상기 하나 이상의 키워드를 참조하여 시드 데이터(seed data)를 변형함으로써 상기 하나 이상의 테스트 케이스를 생성할 수 있다.

[0020] 상기 하나 이상의 테스트 케이스를 생성하는 단계는, 상기 시드 데이터를 변형하여 상기 하나 이상의 테스트 케이스를 생성하되, 상기 하나 이상의 테스트 케이스 각각이 상기 하나 이상의 키워드 중 적어도 하나를 포함하도록 상기 시드 데이터를 변형할 수 있다.

[0021] 일 실시예에 따른 IoT 펌웨어 취약점 점검 장치는, IoT(Internet of Things) 장치의 펌웨어에 대한 펌웨어 아키텍처를 식별하는 식별부; 상기 펌웨어 아키텍처에 기초하여 제1 에뮬레이터 및 제2 에뮬레이터 중 하나를 선택하고, 상기 선택된 에뮬레이터를 이용하여 상기 펌웨어를 에뮬레이션하는 에뮬레이션부; 상기 에뮬레이션된 펌웨어의 실행 경로를 분석하여 하나 이상의 키워드를 추출하는 키워드 추출부; 및 상기 하나 이상의 키워드에 기초하여 하나 이상의 테스트 케이스를 생성하고, 상기 하나 이상의 테스트 케이스를 이용하여 상기 에뮬레이션된 펌웨어에 대한 퍼징 테스트를 수행하는 테스트 수행부를 포함한다.

[0022] 상기 제1 에뮬레이터는, 제1 아키텍처 그룹에 포함된 하나 이상의 펌웨어 아키텍처에 대한 에뮬레이션 환경을 제공하는 에뮬레이터이고, 상기 제2 에뮬레이터는, 제2 아키텍처 그룹에 포함된 하나 이상의 펌웨어 아키텍처에 대한 에뮬레이션 환경을 제공하는 에뮬레이터일 수 있다.

[0023] 상기 제1 에뮬레이터는, 경험적 근거에 기반한 중재 기법이 적용된 시스템 모드 에뮬레이션을 제공할 수 있다.

[0024] 상기 제1 에뮬레이터는, 상기 시스템 모드 에뮬레이션 및 사용자 모드 에뮬레이션을 제공할 수 있다.

[0025] 상기 에뮬레이션부는, 상기 제2 에뮬레이터에 의해 에뮬레이션된 상기 펌웨어에 대한 상기 퍼징 테스트 수행 시 상기 펌웨어의 실행 상태에 기초하여 상기 퍼징 테스트 수행 중 발생한 오류를 탐지하고 상기 탐지된 오류에 대한 메시지를 상기 테스트 수행부로 제공하는 상호 연결 모듈을 포함할 수 있다.

[0026] 상기 제2 에뮬레이터는, 사전 구비된 자동화 스크립트에 기초하여 상기 펌웨어를 에뮬레이션할 수 있다.

[0027] 상기 키워드 추출부는, 상기 에뮬레이션된 펌웨어에 대한 동적 기호 실행(concolic execution)을 수행하고, 상기 동적 기호 실행 결과에 기초하여 상기 하나 이상의 키워드를 추출할 수 있다.

[0028] 상기 키워드 추출부는, 상기 실행 경로 상의 하나 이상의 분기에 대한 경로 제약 조건(path constraint)을 해결하는 하나 이상의 키워드를 추출할 수 있다.

[0029] 상기 테스트 수행부는, 상기 하나 이상의 키워드를 참조하여 시드 데이터(seed data)를 변형함으로써 상기 하나 이상의 테스트 케이스를 생성할 수 있다.

[0030] 상기 테스트 수행부는, 상기 시드 데이터를 변형하여 상기 하나 이상의 테스트 케이스를 생성하되, 상기 하나 이상의 테스트 케이스 각각이 상기 하나 이상의 키워드 중 적어도 하나를 포함하도록 상기 시드 데이터를 변형할 수 있다.

발명의 효과

[0031] 개시되는 실시예들에 따르면, 펌웨어 아키텍처에 기초하여 펌웨어 아키텍처에 적합한 에뮬레이터를 선택하여 펌웨어를 에뮬레이션함으로써 펌웨어에 대한 에뮬레이션 성공률을 높일 수 있으며, 펌웨어의 실행 경로 분석을 통해 추출된 키워드를 기반으로 퍼징 테스트를 위한 테스트 케이스을 구성함으로써 퍼징 테스트 시 깊은 경로까지 정확하게 취약점을 점검할 수 있게 된다.

도면의 간단한 설명

[0032] 도 1은 일 실시예에 따른 IoT 펌웨어 취약점 점검 장치의 구성도

도 2는 일 실시예에 따른 테스트 케이스의 일 예를 나타낸 도면

도 3은 일 실시예에 따른 상호 연결부의 동작을 설명하기 위한 도면

도 4는 일 실시예에 따른 IoT 펌웨어 취약점 점검 방법의 순서도

도 5는 예시적인 실시예들에서 사용되기에 적합한 컴퓨팅 장치를 포함하는 컴퓨팅 환경을 예시하여 설명하기 위한 블록도

발명을 실시하기 위한 구체적인 내용

[0033] 이하, 도면을 참조하여 본 발명의 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.

[0034] 본 발명의 실시예들을 설명함에 있어서, 본 발명과 관련된 공지기술에 대한 구체적인 설명이 본 발명의 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 본 발명의 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 특성들, 숫자들, 단계들, 동작들, 요소들, 이들의 일부 또는 조합을 가리키기 위한 것이며, 기술된 것 이외에 하나 또는 그 이상의 다른 특성, 숫자, 단계, 동작, 요소, 이들의 일부 또는 조합의 존재 또는 가능성을 배제하도록 해석되어서는 안 된다.

[0035] 도 1은 일 실시예에 따른 IoT 펌웨어 취약점 점검 장치의 구성도이다.

[0036] 도 1을 참조하면, 일 실시예에 따른 IoT 펌웨어 취약점 점검 장치(100)는 식별부(110), 에뮬레이션부(120), 키

워드 추출부(130) 및 테스트 수행부(150)를 포함한다.

[0037] 식별부(110)는 IoT(Internet of Things) 장치의 펌웨어(이하, 대상 펌웨어)에 대한 펌웨어 아키텍처를 식별한다.

[0038] 이때, IoT 장치는 예를 들어, IP 카메라, 공유기, 자동차, TV, 등과 같이 IoT 환경 내에서 인터넷을 통해 연결된 다양한 형태의 장치들을 포함할 수 있으며, 반드시 특정한 형태의 장치로 한정되는 것은 아니다.

[0039] 또한, 펌웨어 아키텍처는 예를 들어, x86, MIPS(Microprocessor without Interlocked Pipeline Stages), ARM(Advanced RISC Machine), ARMhf(ARM hard float), Xtensa, PowerPC, RISC-V, ColdFire 등과 같이 대상 펌웨어가 실행되는 마이크로프로세서의 아키텍처를 의미할 수 있다.

[0040] 일 실시예에 따르면, 선택부(110)는 예를 들어, Binwalk, FMK(Firmware Mod Kit), readelf, Ghidra와 같은 공지된 다양한 펌웨어 분석 도구를 이용하여 대상 펌웨어의 펌웨어 아키텍처를 식별할 수 있다.

[0041] 에뮬레이션부(120)는 제1 에뮬레이터(121) 및 제2 에뮬레이터(122)를 포함한다.

[0042] 구체적으로, 제1 에뮬레이터(121) 및 제2 에뮬레이터(122)는 에뮬레이터 선택부(110)의 선택에 따라 대상 펌웨어를 에뮬레이션하기 위한 에뮬레이터이며, 각각 상이한 펌웨어 아키텍처에 대한 에뮬레이션 환경을 제공할 수 있다.

[0043] 일 실시예에 따르면, 제1 에뮬레이터(121)는 제1 아키텍처 그룹에 포함된 하나 이상의 펌웨어 아키텍처에 대한 에뮬레이션 환경을 제공하는 에뮬레이터일 수 있다.

[0044] 이때, 제1 아키텍처 그룹에 포함된 하나 이상의 펌웨어 아키텍처는 예를 들어, x86, MIPS(Microprocessor without Interlocked Pipeline Stages), ARM(Advanced RISC Machine), ARMhf(ARM hard float) 등과 같은 시장 지배적인 주류 펌웨어 아키텍처일 수 있다. 한편, 제1 아키텍처 그룹에 포함된 펌웨어 아키텍처는 사전 설정될 수 있으며, 반드시 상술한 예에 한정되는 것은 아니다.

[0045] 한편, 일 실시예에 따르면, 제1 에뮬레이터(121)는 시스템 모드 에뮬레이션 및 사용자 모드 에뮬레이션을 제공할 수 있고, 시스템 모드 에뮬레이션에는 경험적 근거에 기초한 중재 기법이 적용될 수 있다.

[0046] 구체적으로, 시스템 모드 에뮬레이션은 IoT 장치의 프로세서와 주변 기기들을 포함하는 IoT 장치와 관련된 전체 시스템에 대한 에뮬레이션 환경을 제공하는 에뮬레이션 모드를 의미할 수 있다. IoT 장치와 관련된 전체 시스템에 대한 에뮬레이션을 위해서는 NVRAM(Non-Volatile Random Access Memory), 부트 로더(boot loader), 네트워크와 같은 구성들이 정확히 정의되어야 하므로 다양한 펌웨어에 대해 전체 시스템을 완벽히 에뮬레이션하는 것은 매우 어려운 문제이며, 이는 에뮬레이션 성공률 저하의 원인이 된다. 이에, 일 실시예에 따른 시스템 모드 에뮬레이션은 중재 기법을 통해, 각 에뮬레이션 단계에서 에뮬레이션 실패의 주요 원인을 분석하고 경험적 근거에 기반한 수정 및 추가를 통해 펌웨어 실행을 휴리스틱(heuristic)하게 수정함으로써 에뮬레이션 성공률을 향상시킬 수 있다. 구체적으로, 펌웨어 에뮬레이션에서 발생하는 주요 실패 원인과 중재 기법을 이용한 수정 방법의 일 예는 아래와 같다.

[0047] 부트(boot): 펌웨어가 에뮬레이션 환경에서 실행되기 시작할 때, 부적절한 부팅 시퀀스 및 파일 시스템 인식 불능과 같은 다양한 문제들이 발생하여 커널 패닉(kernel panic)이 발생할 수 있다. 이러한 문제들은 예를 들어, 추가적인 초기화 파일 및 부트 파일 식별을 활성화하여 해결될 수 있다.

[0048] 네트워크: 부팅이 정상적으로 완료되면, 네트워크가 구성된다. 네트워크는 분석 시스템과 에뮬레이터 사이에서 통신을 가능하게 하므로, 네트워크는 퍼징 테스트를 위해 에뮬레이션해야 하는 중요한 구성요소이다. 그러나, 네트워크 정보의 부족, 잘못된 IP 별칭(IP alias), 다중 네트워크 인터페이스 등과 같이 네트워크 구성에 있어 몇 가지 문제점이 존재할 수 있다. 이러한 문제들은 예를 들어, 네트워크를 강제로 구성하고 하나의 이더넷 인터페이스만을 설정함으로써 해결될 수 있다.

[0049] NVRAM: NVRAM은 IoT 장치의 구성 데이터(configuration data)를 저장하기 위한 주변 장치로서 널리 사용되는 플래시 메모리이다. "no NVRAM default files", "only custom default files" 등과 같은 NVRAM 관련 문제들은 예를 들어, NVRAM 파일의 형식(format)인 키-값(key-value) 형식의 파일을 찾고, 마치 NVRAM 파일이 존재하는 것처럼 인위적인 응답을 생성함으로써 해결될 수 있다.

[0050] 커널(Kernel): 일반적인 OS와 같이, 펌웨어는 입출력 제어(ioctl) 명령을 이용하여 주변 장치를 제어하기 위해 주변 장치들과 연결된 커널을 가지고 있다. 그러나, 커널은 펌웨어만큼 다양하므로 지원되지 않는 커널 및 버전

호환성 문제가 발생할 수 있다. 이러한 문제들은 예를 들어, 인위적인 ioctl 명령을 생성하고, 새로운 커널을 컴파일하고 컴파일 옵션들을 추가함으로써 해결될 수 있다.

[0051] 기타: 다른 문제로는 실행되지 않은 웹서버, 에뮬레이팅 타임아웃, 에뮬레이션에 부적절한 프로그램 등이 있으며, 이러한 문제들은 예를 들어, 대응하는 구성 파일로 웹서버를 찾은 후 강제로 에뮬레이션을 실행하고, 타임아웃을 최적화하고, 에뮬레이터에 최신 버전의 비지박스(busybox)를 추가함으로써 해결될 수 있다.

[0052] 한편, 사용자 모드 에뮬레이션은 특정 프로세서에 대해 컴파일된 프로세스를 다른 CPU에서 수행하도록 하는 에뮬레이션 모드를 의미할 수 있다.

[0053] 일 실시예에 따르면, 제1 에뮬레이터(121)는 기본적으로 사용자 모드 에뮬레이션을 통해 펌웨어를 에뮬레이션하며, 사용자 모드 에뮬레이션에서 처리할 수 없는 시스템 호출이 발생한 경우, 시스템 모드 에뮬레이션으로 스위칭하여 시스템 호출을 처리한 후 사용자 모드 에뮬레이션으로 복귀할 수 있다.

[0054] 한편, 일 실시예에 따르면, 제2 에뮬레이터(122)는 제2 아키텍처 그룹에 포함된 하나 이상의 펌웨어 아키텍처에 대한 에뮬레이션 환경을 제공하는 에뮬레이터일 수 있다.

[0055] 이때, 제2 아키텍처 그룹에 포함된 하나 이상의 펌웨어 아키텍처는 예를 들어, Xtensa, PowerPC, RISC-V, ColdFire 등과 같은 비주류 펌웨어 아키텍처일 수 있다. 한편, 제2 아키텍처 그룹에 포함된 펌웨어 아키텍처는 사전 설정될 수 있으며, 반드시 상술한 예에 한정되는 것은 아니다.

[0056] 일 실시예에 따르면, 제2 에뮬레이터(122)는 사전 구비된 자동화 구성 스크립트에 기초하여 대상 펌웨어를 에뮬레이션할 수 있다. 이때, 자동화 구성 스크립트는 제2 에뮬레이터(122)에 의해 제공되는 에뮬레이션 환경에서 대상 펌웨어에 대한 퍼징 테스트를 수행하기 위해 필요한 다양한 정보들을 포함할 수 있다. 구체적으로, 자동화 구성 스크립트는 예를 들어, 데이터 레지스터 식별 정보, 메모리 레이아웃 및 퍼징 시작 주소 등과 같이 퍼징 테스트에 의한 분석을 시작할 주소와 입력 파라미터를 명시하기 위한 정보를 포함할 수 있다.

[0057] 한편, 에뮬레이션부(120)는 식별부(110)에 의해 식별된 펌웨어 아키텍처에 기초하여 제1 에뮬레이터(121) 및 제2 에뮬레이터(122) 중 하나를 선택하고, 선택된 에뮬레이터를 이용하여 대상 펌웨어를 에뮬레이션한다.

[0058] 구체적으로, 일 실시예에 따르면, 에뮬레이션부(120)는 식별된 펌웨어 아키텍처가 제1 아키텍처 그룹에 속하는 펌웨어 아키텍처인 경우, 제1 에뮬레이터(121)를 대상 펌웨어를 에뮬레이션할 에뮬레이터로 선택하고, 식별된 펌웨어 아키텍처가 제2 아키텍처 그룹에 속하는 펌웨어 아키텍처인 경우, 제2 에뮬레이터(122)를 대상 펌웨어를 에뮬레이션할 에뮬레이터로 선택할 수 있다.

[0059] 키워드 추출부(130)는 제1 에뮬레이터(121) 및 제2 에뮬레이터(122) 중 선택부(110)에 의해 선택된 에뮬레이터에 의해 제공되는 에뮬레이션 환경에서 대상 펌웨어의 실행 경로를 분석하여 하나 이상의 키워드를 추출한다.

[0060] 일 실시예에 따르면, 키워드 추출부(130)는 선택된 에뮬레이터에 의해 에뮬레이션된 대상 펌웨어에 대해 동적 기호 실행(concolic execution)을 수행하고, 동적 기호 실행 결과에 기초하여 대상 펌웨어로부터 하나 이상의 키워드를 추출할 수 있다.

[0061] 구체적으로, 일 실시예에 따르면, 키워드 추출부(130)는 동적 기호 실행 결과에 기초하여 대상 펌웨어의 실행 경로 상의 하나 이상의 분기(branch)에 대한 경로 제약 조건(path constraint)를 해결하는 하나 이상의 키워드를 추출할 수 있다. 보다 구체적으로, 키워드 추출부(130)는 대상 펌웨어 펌웨어의 실행 경로 상에 있는 각 분기에 접근하여 통과하기 위해 필요한 헤더 및 값 중 적어도 하나를 키워드로 추출할 수 있다.

[0062] 테스트 수행부(140)는 키워드 추출부(130)에 의해 추출된 하나 이상의 키워드에 기초하여 하나 이상의 테스트 케이스를 생성한다. 또한, 테스트 수행부(140)는 생성된 하나 이상의 테스트 케이스를 이용하여, 선택부(110)에 의해 선택된 에뮬레이터에 의해 에뮬레이션된 대상 펌웨어에 대한 퍼징 테스트를 수행한다.

[0063] 일 실시예에 따르면, 테스트 수행부(140)는 추출된 하나 이상의 키워드를 참조하여 시드 데이터(seed data)를 변형함으로써 하나 이상의 테스트 케이스를 생성할 수 있다. 구체적으로, 테스트 수행부(140)는 시드 데이터를 변형하여 하나 이상의 테스트 케이스를 생성하되, 각 테스트 케이스가 추출된 하나 이상의 키워드 중 적어도 하나를 포함하도록 시드 데이터를 변형할 수 있다.

[0064] 도 2는 일 실시예에 따른 테스트 케이스의 일 예를 나타낸 도면이다.

[0065] 도 2에 도시된 예에서, 사각형 박스로 표시된 부분은 각각 키워드 추출에 의해 추출된 키워드를 나타내며, 테스

트 수행부(140)는 추출된 키워드를 제외한 나머지 부분을 변형함으로써 테스트 케이스를 생성할 수 있다.

[0066] 한편, 테스트 수행부(140)는 생성된 테스트 케이스를 이용하여 퍼징 테스트를 수행하고, 퍼징 테스트 수행 결과 경로 커버리지(path coverage)가 증가한 경우, 해당 시드 데이터를 추가적으로 변형하여 새로운 테스트 케이스를 생성하는 방식으로 퍼징 테스트를 반복 수행할 수 있다.

[0067] 한편, 일 실시예에 따르면, 제2 에뮬레이터(122)에 의해 에뮬레이션되는 대상 펌웨어는 펌웨어 실행 중 메모리 충돌(memory corruption), 스틱(stuck) 상태 등과 같은 오류 발생 시 이에 대한 응답 또는 로깅(logging) 기능을 구비하지 않은 펌웨어일 수 있다. 이 경우, 테스트 수행부(140)에 의한 퍼징 테스트 수행 시 테스트 수행부(140)가 대상 펌웨어에서 발생한 충돌 또는 오류를 파악하지 못하는 경우가 발생할 수 있다. 이러한 문제를 해결하기 위해, 에뮬레이션부(120)는 제2 에뮬레이터(122)에 의해 에뮬레이션된 대상 펌웨어에 대한 퍼징 테스트 수행 시 대상 펌웨어의 실행 상태에 기초하여 퍼징 테스트 중 발생한 오류를 탐지하고, 탐지된 오류에 대한 정보를 테스트 수행부(140)로 제공하는 상호 연결부(123)를 더 포함할 수 있다.

[0068] 구체적으로, 도 3은 일 실시예에 따른 상호 연결부의 동작을 설명하기 위한 도면이다.

[0069] 도 3을 참조하면, 테스트 수행부(140)는 퍼징 테스트를 위한 테스트 케이스를 생성한 후 상호 연결부(123)로 생성한 테스트 케이스를 제공하고, 상호 연결부(123)는 해당 테스트 케이스를 제2 에뮬레이터(122)로 전달할 수 있다.

[0070] 테스트 케이스를 수신한 제2 에뮬레이터(122)는 수신된 테스트 케이스를 이용하여 대상 펌웨어를 에뮬레이션할 수 있다. 이때, 제2 에뮬레이터(122)는 상술한 자동화 구성 스크립트를 참고하여 대상 펌웨어를 에뮬레이션할 수 있다.

[0071] 한편, 제2 에뮬레이터(122)는 테스트 케이스에 의한 경로 커버리지(path coverage)를 측정하여 테스트 수행부(140)로 제공할 수 있다. 예를 들어, 제2 에뮬레이터(122)는 테스트 케이스에 의해 실행된 펌웨어의 기본 블록 주소를 측정하고, 측정된 주소를 테스트 수행부(140)로 제공할 수 있다.

[0072] 또한, 제2 에뮬레이터(122)는 테스트 케이스에 의한 대상 펌웨어의 실행 상태를 상호 연결부(123)로 제공하고, 상호 연결부(123)는 대상 펌웨어의 실행 상태를 분석하여 예를 들어, 메모리 충돌과 같은 오류를 탐지할 수 있다. 오류가 탐지된 경우, 상호 연결부(123)는 테스트 수행부(140)로 탐지된 오류에 대한 보고 메시지를 제공할 수 있다. 이때, 상호 연결부(123)에 의해 탐지되는 오류는 사전에 탐지 대상인 취약점으로 명시된 오류일 수 있다.

[0073] 도 4는 일 실시예에 따른 IoT 펌웨어 취약점 점검 방법의 순서도이다.

[0074] 도 4에 도시된 방법은 예를 들어, IoT 펌웨어 취약점 점검 장치(100)에 의해 수행될 수 있다.

[0075] 도 4를 참조하면, IoT 펌웨어 취약점 점검 장치(100)는 대상 펌웨어에 대한 펌웨어 아키텍처를 식별한다(410).

[0076] 이후, IoT 펌웨어 취약점 점검 장치(100)는 식별된 펌웨어 아키텍처에 기초하여 제1 에뮬레이터 및 제2 에뮬레이터 중 하나를 선택한다(420).

[0077] 이후, IoT 펌웨어 취약점 점검 장치(100)는 선택된 에뮬레이터를 이용하여 대상 펌웨어를 에뮬레이션한다(430).

[0078] 이후, IoT 펌웨어 취약점 점검 장치(100)는 에뮬레이션된 대상 펌웨어의 실행 경로를 분석하여 하나 이상의 키워드를 추출한다(440).

[0079] 이후, IoT 펌웨어 취약점 점검 장치(100)는 추출된 하나 이상의 키워드에 기초하여 하나 이상의 테스트 케이스를 생성한다(450).

[0080] 이후, IoT 펌웨어 취약점 점검 장치(100)는 하나 이상의 테스트 케이스를 이용하여 상기 에뮬레이션된 펌웨어에 대한 퍼징 테스트를 수행한다(460).

[0081] 한편, 도 4에 도시된 순서도에서 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.

[0082] 도 5는 예시적인 실시예들에서 사용되기에 적합한 컴퓨팅 장치를 포함하는 컴퓨팅 환경을 예시하여 설명하기 위한 블록도이다.

[0083] 도 5에 도시된 실시예에서, 각 컴포넌트들은 이하에 기술된 것 이외에 상이한 기능 및 능력을 가질 수 있고, 이하에 기술된 것 이외에도 추가적인 컴포넌트를 포함할 수 있다.

[0084] 컴퓨팅 환경(10)은 컴퓨팅 장치(12)를 포함한다. 일 실시예에서, 컴퓨팅 장치(12)는 IoT 펌웨어 취약점 점검 장치(100)에 포함되는 하나 이상의 컴포넌트일 수 있다.

[0085] 컴퓨팅 장치(12)는 적어도 하나의 프로세서(14), 컴퓨터 판독 가능 저장 매체(16) 및 통신 버스(18)를 포함한다. 프로세서(14)는 컴퓨팅 장치(12)로 하여금 앞서 언급된 예시적인 실시예에 따라 동작하도록 할 수 있다. 예컨대, 프로세서(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 프로세서(14)에 의해 실행되는 경우 컴퓨팅 장치(12)로 하여금 예시적인 실시예에 따른 동작들을 수행하도록 구성될 수 있다.

[0086] 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 프로세서(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.

[0087] 통신 버스(18)는 프로세서(14), 컴퓨터 판독 가능 저장 매체(16)를 포함하여 컴퓨팅 장치(12)의 다른 다양한 컴포넌트들을 상호 연결한다.

[0088] 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.

[0089] 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 특허청구범위뿐만 아니라 이 특허청구범위와 균등한 것들에 의해 정해져야 한다.

부호의 설명

[0090] 10: 컴퓨팅 환경

12: 컴퓨팅 장치

14: 프로세서

16: 컴퓨터 판독 가능 저장 매체

18: 통신 버스

20: 프로그램

22: 입출력 인터페이스

24: 입출력 장치

26: 네트워크 통신 인터페이스

100: IoT 펌웨어 취약점 점검 장치

110: 식별부

120: 에뮬레이션부

121: 제1 에뮬레이터

122: 제2 에뮬레이터

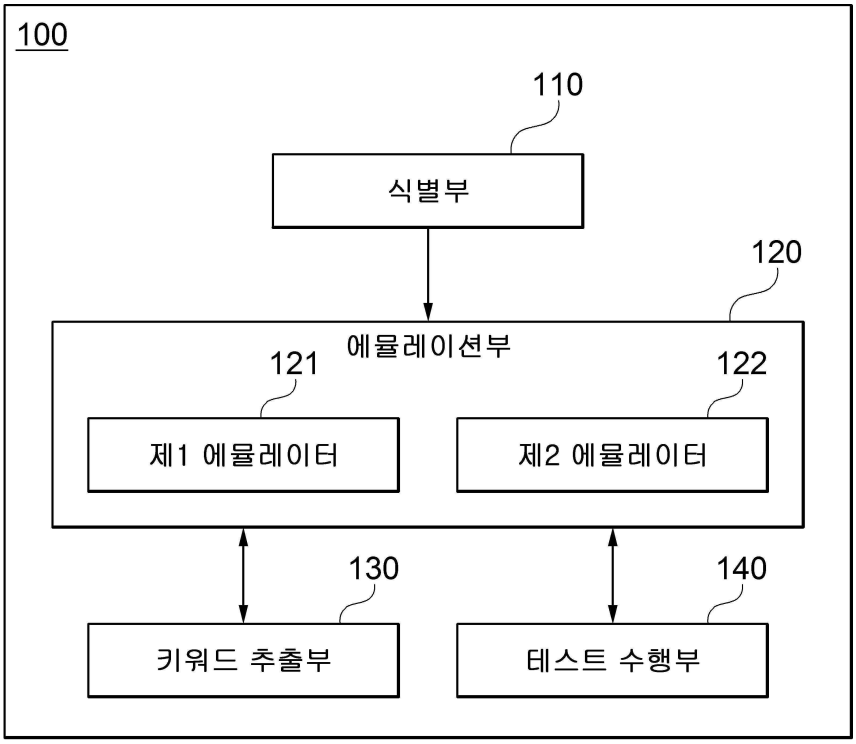
123: 상호 연결부

130: 키워드 추출부

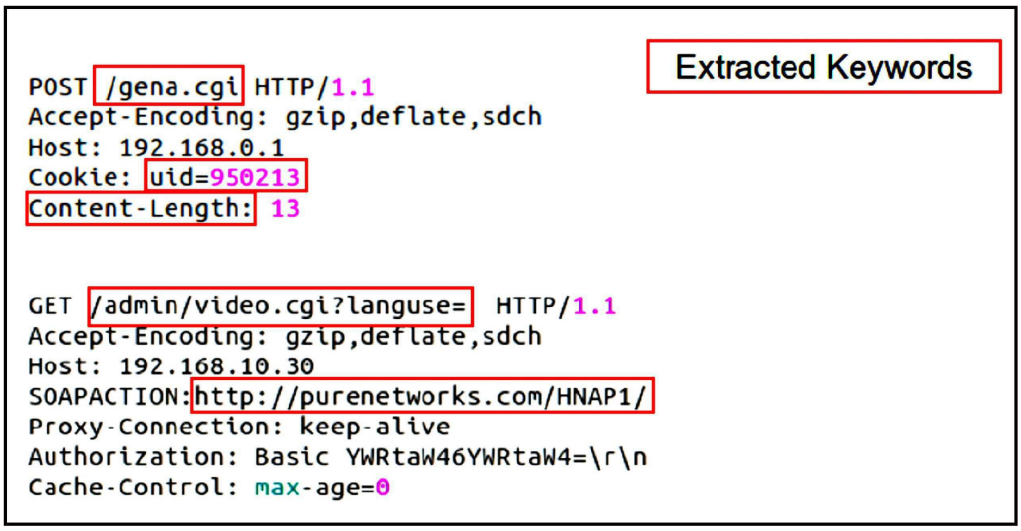
140: 테스트 수행부

도면

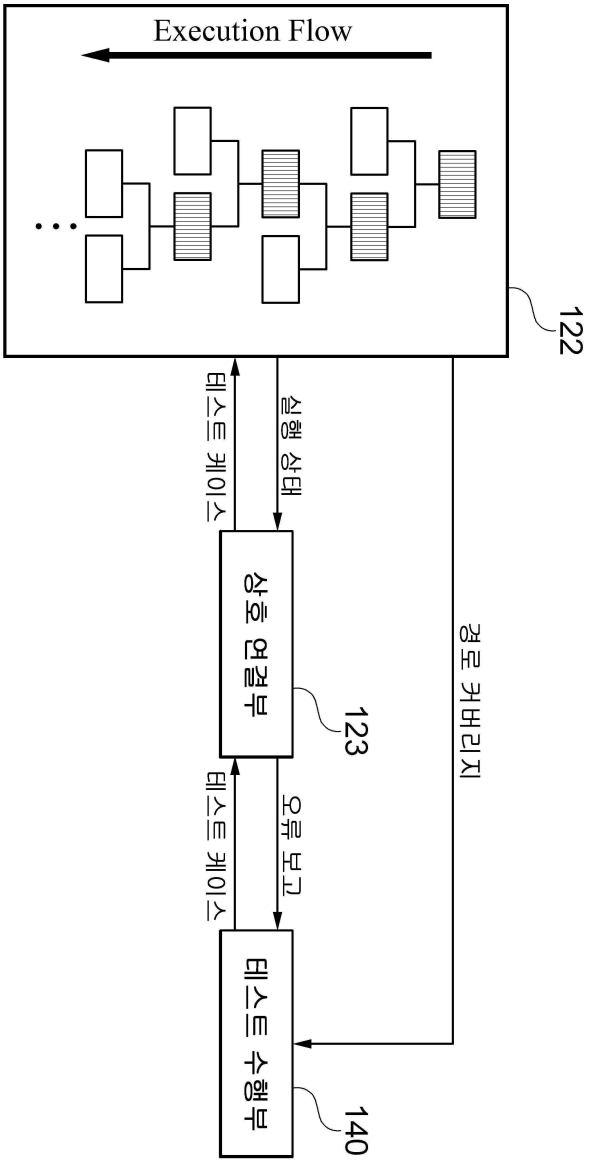
도면1



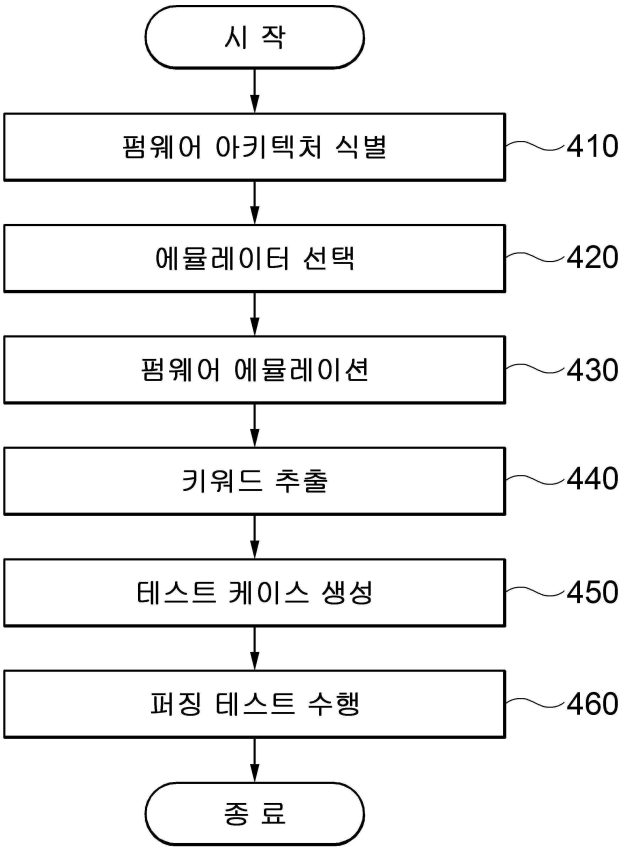
도면2


Extracted Keywords
POST /gena.cgi HTTP/1.1
Accept-Encoding: gzip,deflate,sdch
Host: 192.168.0.1
Cookie: uid=950213
Content-Length: 13
GET /admin/video.cgi?languse= HTTP/1.1
Accept-Encoding: gzip,deflate,sdch
Host: 192.168.10.30
SOAPACTION:http://purenetworks.com/HNAP1/
Proxy-Connection: keep-alive
Authorization: Basic YWRtaW46YWRtaW4=\r\n
Cache-Control: max-age=0

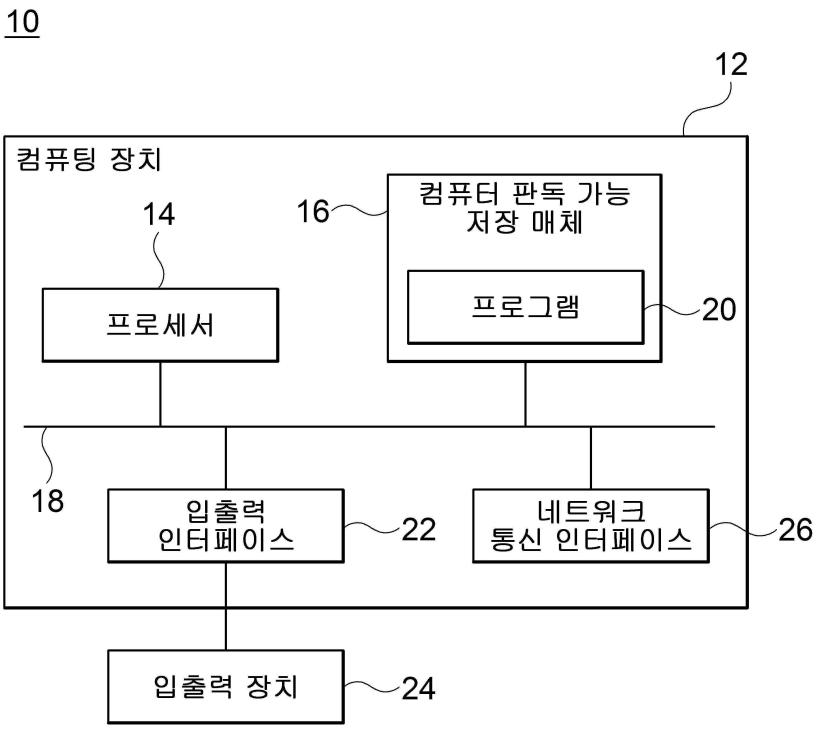
도면3


Execution Flow
122
테스트 케이스
실행 상태
상호 연결부
123
경로 커버리지
테스트 케이스
오류 보고
테스트 수행부
140

도면4


시 작
펌웨어 아키텍처 식별
410
에뮬레이터 선택
420
펌웨어 에뮬레이션
430
키워드 추출
440
테스트 케이스 생성
450
퍼징 테스트 수행
460
종 료

도면5


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