



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2025-0017613
(43) 공개일자 2025년02월04일

(51) 국제특허분류(Int. Cl.)
G06F 11/36 (2025.01) G06F 11/22 (2017.01)
G06F 11/263 (2006.01) G06F 11/34 (2006.01)
G06N 3/092 (2023.01)
(52) CPC특허분류
G06F 11/3652 (2013.01)
G06F 11/2263 (2013.01)
(21) 출원번호 10-2023-0154421
(22) 출원일자 2023년11월09일
심사청구일자 2023년11월09일
(30) 우선권주장
1020230098189 2023년07월27일 대한민국(KR)

(71) 출원인
세종대학교산학협력단
서울특별시 광진구 능동로 209 (군자동, 세종대학교)
(72) 발명자
윤주범
서울특별시 송파구 송파로4길 19, 704동 401호(장지동, 송파파인타운7단지)
유지현
서울특별시 광진구 광나루로28길 18, 302호(화양동)
(74) 대리인
특허법인인

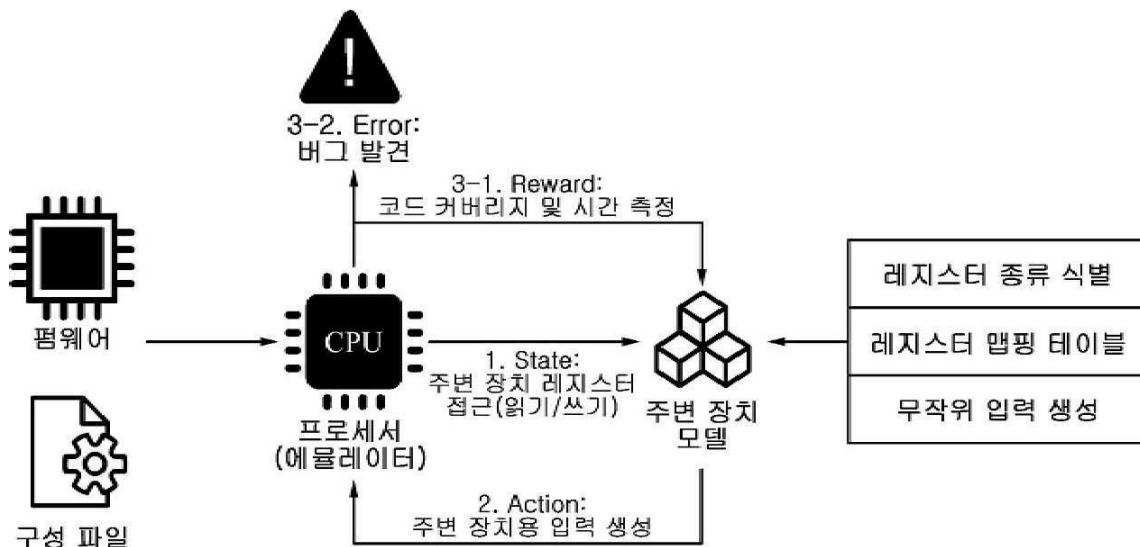
전체 청구항 수 : 총 15 항

(54) 발명의 명칭 강화 학습 기반 주변 장치 모델링을 통한 임베디드 퍼징 시스템 및 방법

(57) 요약

강화 학습 기반 주변 장치 모델링을 통한 임베디드 퍼징 시스템 및 방법이 개시된다. 본 발명의 일 실시예에 따른 임베디드 퍼징 시스템은, 펌웨어 및 펌웨어의 내부 메모리 맵을 포함하는 구성 파일을 수신하면, 구성 파일을 기초로 펌웨어를 실행하기 위한 에뮬레이터 모듈; 주변 장치 레지스터의 접근을 인지하고, 인지된 주변 장치 레지스터의 종류를 식별하기 위한 주변 장치 감지 모듈; 및 식별된 주변 장치 레지스터의 종류의 따라, 무작위 입력력을 발생시켜 예외 핸들러의 발생 여부를 검사하여 버그 및 취약점을 탐지하거나, 또는 복수 개의 추론을 통한 강화 학습을 이용하여 주변 장치 모델링을 수행하는 입력 처리 모듈을 포함한다.

대표도



(52) CPC특허분류

G06F 11/263 (2013.01)
G06F 11/3447 (2013.01)
G06F 11/3684 (2013.01)
G06F 11/3688 (2013.01)
G06F 11/3692 (2013.01)
G06F 11/3698 (2025.01)
G06N 3/092 (2023.01)

윤여훈

서울특별시 광진구 광나루로13길 4, 102호(군자동)

(72) 발명자

김주환

서울특별시 광진구 군자로 175-2, 304호(군자동)

이 발명을 지원한 국가연구개발사업

과제고유번호	1711193689
과제번호	2018-0-01423-006
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기획평가원
연구사업명	대학 ICT 연구센터육성지원사업
연구과제명	지능형 비행로봇 융합기술 연구
기 여 율	1/2
과제수행기관명	세종대학교 산학협력단
연구기간	2023.01.01 ~ 2023.12.31

이 발명을 지원한 국가연구개발사업

과제고유번호	1711193570
과제번호	2020-0-01602-004
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기획평가원
연구사업명	대학 ICT 연구센터지원사업
연구과제명	지능형 사이버 위협 대응 기술 개발 및 인력양성
기 여 율	1/2
과제수행기관명	숭실대학교 산학협력단
연구기간	2023.01.01 ~ 2023.12.31

명세서

청구범위

청구항 1

펌웨어 및 상기 펌웨어의 내부 메모리 맵을 포함하는 구성 파일을 수신하면, 상기 구성 파일을 기초로 상기 펌웨어를 실행하기 위한 에뮬레이터 모듈;

주변 장치 레지스터의 접근을 인지하고, 인지된 상기 주변 장치 레지스터의 종류를 식별하기 위한 주변 장치 감지 모듈; 및

식별된 상기 주변 장치 레지스터의 종류의 따라, 무작위 입력을 발생시켜 예외 핸들러의 발생 여부를 검사하여 버그 및 취약점을 탐지하거나, 또는 복수 개의 추론을 통한 강화 학습을 이용하여 주변 장치 모델링을 수행하는 입력 처리 모듈을 포함하는, 임베디드 퍼징 시스템.

청구항 2

청구항 1에 있어서,

상기 주변 장치 레지스터는, 제어 레지스터 및 상태 레지스터를 포함하고,

상기 입력 처리 모듈은,

상기 주변 장치 레지스터의 종류가 상기 제어 레지스터 또는 상기 상태 레지스터인 경우, 적어도 둘 이상의 추론 값을 생성하여 해당 레지스터에 대한 강화 학습을 수행하는, 임베디드 퍼징 시스템.

청구항 3

청구항 2에 있어서,

상기 입력 처리 모듈은,

생성된 상기 적어도 둘 이상의 추론 값 중 적어도 하나 이상에 대한 강화 학습을 통해 상기 주변 장치 레지스터에 최적값을 추론하여 응답하는, 임베디드 퍼징 시스템.

청구항 4

청구항 3에 있어서,

상기 입력 처리 모듈은,

상기 해당 레지스터에 대한 강화 학습을 수행할 때, 상기 해당 레지스터에 특정 값이 할당됨에 따라 발생된 모든 경로의 수 및 각 경로에 대한 탐색에 따른 총 소요시간을 기초로 보상값을 산출하는, 임베디드 퍼징 시스템.

청구항 5

청구항 1에 있어서,

상기 주변 장치 레지스터는, 데이터 레지스터를 포함하고,

상기 입력 처리 모듈은,

상기 주변 장치 레지스터의 종류가 상기 데이터 레지스터인 경우, 임의의 무작위 값을 생성하여 피드하는, 임베디드 퍼징 시스템.

청구항 6

청구항 5에 있어서,

상기 입력 처리 모듈은,

상기 임의의 무작위 값을 상기 펌웨어에 입력하고, 상기 펌웨어의 상태를 체크하여 상기 예외 핸들러가 호출되었는지 여부를 확인하는, 임베디드 퍼징 시스템.

청구항 7

청구항 5에 있어서,

상기 입력 처리 모듈은,

상기 임의의 무작위 값을 상기 펌웨어에 입력하고, 상기 펌웨어의 상태를 체크하여 에뮬레이션된 펌웨어의 재부팅을 비롯한 에러 발생 여부를 확인하는, 임베디드 퍼징 시스템.

청구항 8

임베디드 퍼징 시스템에 의해 수행되는 임베디드 퍼징 방법에 있어서,

펌웨어 및 상기 펌웨어의 내부 메모리 맵을 포함하는 구성 파일을 수신하는 단계;

상기 구성 파일을 기초로 상기 펌웨어를 실행하는 단계;

주변 장치 레지스터의 접근을 인지하고, 인지된 상기 주변 장치 레지스터의 종류를 식별하는 단계;

식별된 상기 주변 장치 레지스터의 종류의 따라, 무작위 입력을 발생시켜 예외 핸들러의 발생 여부를 검사하여 버그 및 취약점을 탐지하거나, 또는 복수 개의 추론을 통한 강화 학습을 이용하여 주변 장치 모델링을 수행하는 단계를 포함하는, 임베디드 퍼징 방법.

청구항 9

청구항 8에 있어서,

상기 주변 장치 레지스터는, 제어 레지스터 및 상태 레지스터를 포함하고,

상기 주변 장치 모델을 수행하는 단계에서,

상기 주변 장치 레지스터의 종류가 상기 제어 레지스터 또는 상기 상태 레지스터인 경우, 적어도 둘 이상의 추론 값을 생성하여 해당 레지스터에 대한 강화 학습을 수행하는, 임베디드 퍼징 방법.

청구항 10

청구항 9에 있어서,

상기 주변 장치 모델을 수행하는 단계에서,

생성된 상기 적어도 둘 이상의 추론 값 중 적어도 하나 이상에 대한 강화 학습을 통해 상기 주변 장치 레지스터에 최적값을 추론하여 응답하는, 임베디드 퍼징 방법.

청구항 11

청구항 10에 있어서,

상기 주변 장치 모델을 수행하는 단계에서,

상기 해당 레지스터에 대한 강화 학습을 수행할 때, 상기 해당 레지스터에 특정 값이 할당됨에 따라 발생된 모든 경로의 수 및 각 경로에 대한 탐색에 따른 총 소요시간을 기초로 보상값을 산출하는, 임베디드 퍼징 방법.

청구항 12

청구항 8에 있어서,

상기 주변 장치 레지스터는, 데이터 레지스터를 포함하고,

상기 주변 장치 모델을 수행하는 단계에서,

상기 주변 장치 레지스터의 종류가 상기 데이터 레지스터인 경우, 임의의 무작위 값을 생성하여 피드하는, 임베디드 퍼징 방법.

청구항 13

청구항 12에 있어서,

상기 주변 장치 모델을 수행하는 단계에서,

상기 임의의 무작위 값을 상기 펌웨어에 입력하고, 상기 펌웨어의 상태를 체크하여 상기 예외 핸들러가 호출되었는지 여부를 확인하는, 임베디드 퍼징 방법.

청구항 14

청구항 12에 있어서,

상기 주변 장치 모델을 수행하는 단계에서,

상기 임의의 무작위 값을 상기 펌웨어에 입력하고, 상기 펌웨어의 상태를 체크하여 예물레이션된 펌웨어의 재부팅을 비롯한 에러 발생 여부를 확인하는, 임베디드 퍼징 방법.

청구항 15

청구항 8 내지 14 중 어느 한 항의 방법을 실행시키기 위한 프로그램을 기록한 컴퓨터로 판독 가능한 기록매체.

발명의 설명

기술 분야

[0001] 개시되는 실시예들은 강화 학습 기반 주변 장치 모델링을 통한 임베디드 퍼징 시스템 및 방법과 관련된다.

배경 기술

[0002] 임베디드 장치는 가전 제품, 공유기, 휴대폰, 드론, 산업 및 의료 시스템 등을 포함할 수 있다. 이러한 임베디드 장치의 소프트웨어 스택인 임베디드 펌웨어는 보안 위협 등의 위험에 노출될 수 있기 때문에, 펌웨어의 취약점을 검출하기 위해 다양한 방안이 제안되고 있는 실정이다. 예를 들어, 임베디드 펌웨어를 테스트하는 방법으로, 취약점 정보와 펌웨어의 함수 유사도 비교를 통해 취약한 펌웨어를 찾아내는 방법이 제안되고 있다.

[0003] 운용자는 상술한 방법 이외에도 임베디드 펌웨어의 취약점을 보다 체계적으로 검출할 수 있는 방안을 모색하게 되었다.

선행기술문헌

특허문헌

[0005] (특허문헌 0001) 대한민국 공개특허공보 제10-2023-0016369호 (2023. 02. 02.)

발명의 내용

해결하려는 과제

[0006] 개시된 실시예들은 임베디드 장치의 MCU(microcontroller unit) 기반 임베디드 장치의 펌웨어를 대상으로 버그 취약점을 탐색하기 위한 강화 학습 기반 주변 장치 모델링을 통한 임베디드 퍼징 시스템 및 방법을 제공하고자 한다.

과제의 해결 수단

[0008] 일 실시예에 따른 임베디드 퍼징 시스템은, 펌웨어 및 상기 펌웨어의 내부 메모리 맵을 포함하는 구성 파일을 수신하면, 상기 구성 파일을 기초로 상기 펌웨어를 실행하기 위한 에뮬레이터 모듈; 주변 장치 레지스터의 접근을 인지하고, 인지된 상기 주변 장치 레지스터의 종류를 식별하기 위한 주변 장치 감지 모듈; 및 식별된 상기 주변 장치 레지스터의 종류의 따라, 무작위 입력을 발생시켜 예외 핸들러의 발생 여부를 검사하여 버그 및 취약점을 탐지하거나, 또는 복수 개의 추론을 통한 강화 학습을 이용하여 주변 장치 모델링을 수행하는 입력 처리 모듈을 포함한다.

[0009] 상기 주변 장치 레지스터는, 제어 레지스터 및 상태 레지스터를 포함하고, 상기 입력 처리 모듈은, 상기 주변 장치 레지스터의 종류가 상기 제어 레지스터 또는 상기 상태 레지스터인 경우, 적어도 둘 이상의 추론 값을 생성하여 해당 레지스터에 대한 강화 학습을 수행할 수 있다.

[0010] 상기 입력 처리 모듈은, 생성된 상기 적어도 둘 이상의 추론 값 중 적어도 하나 이상에 대한 강화 학습을 통해 주변 장치 레지스터에 최적값을 추론하여 응답할 수 있다.

[0011] 상기 입력 처리 모듈은, 상기 해당 레지스터에 대한 강화 학습을 수행할 때, 상기 해당 레지스터에 특정 값이 할당됨에 따라 발생된 모든 경로의 수 및 각 경로에 대한 탐색에 따른 총 소요시간을 기초로 보상값을 산출할 수 있다.

[0012] 상기 주변 장치 레지스터는, 데이터 레지스터를 포함하고, 상기 입력 처리 모듈은, 상기 주변 장치 레지스터의 종류가 상기 데이터 레지스터인 경우, 임의의 무작위 값을 생성하여 피드할 수 있다.

[0013] 상기 입력 처리 모듈은, 상기 임의의 무작위 값을 상기 펌웨어에 입력하고, 상기 펌웨어의 상태를 체크하여 상기 예외 핸들러가 호출되었는지 여부를 확인할 수 있다.

[0014] 상기 입력 처리 모듈은, 상기 임의의 무작위 값을 상기 펌웨어에 입력하고, 상기 펌웨어의 상태를 체크하여 에뮬레이션된 펌웨어의 재부팅을 비롯한 에러 발생 여부를 확인할 수 있다.

[0015] 다른 실시예에 따른 임베디드 퍼징 방법은, 임베디드 퍼징 시스템에 의해 수행되는 임베디드 퍼징 방법에 있어서, 펌웨어 및 상기 펌웨어의 내부 메모리 맵을 포함하는 구성 파일을 수신하는 단계; 상기 구성 파일을 기초로 상기 펌웨어를 실행하는 단계; 주변 장치 레지스터의 접근을 인지하고, 인지된 상기 주변 장치 레지스터의 종류를 식별하는 단계; 식별된 상기 주변 장치 레지스터의 종류의 따라, 무작위 입력을 발생시켜 예외 핸들러의 발생 여부를 검사하여 버그 및 취약점을 탐지하거나, 또는 복수 개의 추론을 통한 강화 학습을 이용하여 주변 장치 모델링을 수행하는 단계를 포함한다.

[0016] 상기 주변 장치 레지스터는, 제어 레지스터 및 상태 레지스터를 포함하고, 상기 임베디드 퍼징 방법은, 상기 주변 장치 모델을 수행하는 단계에서, 상기 주변 장치 레지스터의 종류가 상기 제어 레지스터 또는 상기 상태 레지스터인 경우, 적어도 둘 이상의 추론 값을 생성하여 해당 레지스터에 대한 강화 학습을 수행할 수 있다.

- [0017] 상기 임베디드 퍼징 방법은, 상기 주변 장치 모델을 수행하는 단계에서, 생성된 상기 적어도 둘 이상의 추론 값 중 적어도 하나 이상에 대한 강화 학습을 통해 주변 장치 레지스터에 최적값을 추론하여 응답할 수 있다.
- [0018] 상기 임베디드 퍼징 방법은, 상기 주변 장치 모델을 수행하는 단계에서, 상기 해당 레지스터에 대한 강화 학습을 수행할 때, 상기 해당 레지스터에 특정 값이 할당됨에 따라 발생된 모든 경로의 수 및 각 경로에 대한 탐색에 따른 총 소요시간을 기초로 보상값을 산출할 수 있다.
- [0019] 상기 주변 장치 레지스터는, 데이터 레지스터를 포함하고, 상기 임베디드 퍼징 방법은, 상기 주변 장치 모델을 수행하는 단계에서, 상기 주변 장치 레지스터의 종류가 상기 데이터 레지스터인 경우, 임의의 무작위 값을 생성하여 피드할 수 있다.
- [0020] 상기 임베디드 퍼징 방법은, 상기 주변 장치 모델을 수행하는 단계에서, 상기 임의의 무작위 값을 상기 펌웨어에 입력하고, 상기 펌웨어의 상태를 체크하여 상기 예외 핸들러가 호출되었는지 여부를 확인할 수 있다.
- [0021] 상기 임베디드 퍼징 방법은, 상기 주변 장치 모델을 수행하는 단계에서, 상기 임의의 무작위 값을 상기 펌웨어에 입력하고, 상기 펌웨어의 상태를 체크하여 에뮬레이션된 펌웨어의 재부팅을 비롯한 에러 발생 여부를 확인할 수 있다.
- [0022] 이 외에도, 개시되는 실시예를 구현하기 위한 방법을 실행하기 위한 컴퓨터 프로그램을 기록하는 컴퓨터 판독 가능한 기록 매체가 더 제공될 수 있다.

발명의 효과

- [0024] 개시되는 실시예들에 따르면, 임베디드 장치의 보안 위협을 완화하기 위해 펌웨어의 보안 취약점을 자동으로 탐색할 수 있다는 효과를 기대할 수 있다.
- [0025] 또한, 개시되는 실시예들에 따르면, 펌웨어 분석을 위해 필수적인 주변 장치 모델링에 강화 학습을 적용하여 펌웨어 내부를 효율적으로 탐색할 수 있다.

도면의 간단한 설명

- [0027] 도 1은 일 실시예에 따른 임베디드 퍼징 시스템의 동작을 개략적으로 설명하기 위한 도면
- 도 2는 일 실시예에 따른 임베디드 퍼징 시스템을 설명하기 위한 블록도
- 도 3a 내지 도 9는 일 실시예에 따른 임베디드 퍼징 방법을 설명하기 위한 예시도
- 도 10은 일 실시예에 따른 임베디드 퍼징 방법을 설명하기 위한 흐름도
- 도 11은 일 실시예에 따른 컴퓨팅 장치를 포함하는 컴퓨팅 환경을 예시하여 설명하기 위한 블록도

발명을 실시하기 위한 구체적인 내용

- [0028] 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.
- [0029] 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.

- [0030] 이하에서 개시하는 주변 장치는 MCU(microcontroller unit)를 채택하는 다양한 임베디드 장치들을 의미할 수 있다. 개시되는 실시예에서는 임베디드 장치들을 대상으로 보안 위협을 효율적으로 탐지하기 위한 강화 학습 기반 주변 장치 모델링을 통한 임베디드 퍼징 방법을 개시하기로 한다.
- [0032] 도 1은 일 실시예에 따른 임베디드 퍼징 시스템의 동작을 개략적으로 설명하기 위한 도면이고, 도 2는 일 실시예에 따른 임베디드 퍼징 시스템을 설명하기 위한 블록도이다.
- [0033] 이하에서는, 일 실시예에 따른 임베디드 퍼징 방법을 설명하기 위한 예시도인 도 3a 내지 도 9를 참고하여 설명하기로 한다.
- [0034] 도 1을 참고하면, 임베디드 퍼징 시스템(100)은 MCU 기반 임베디드 장치(이하, 주변 장치라 하기로 함)의 전체 소프트웨어 스택인 펌웨어(MCU 펌웨어)를 대상으로 취약점을 점검할 수 있다. 예를 들어, 주변 장치는 가전 제품, 공유기, 휴대폰, 드론, 산업 및 의료 시스템 등을 포함할 수 있으며, 이에 한정되지 않는다. 이러한 주변 장치의 소프트웨어 스택인 펌웨어는 프로그래밍으로 지정된 작업을 수행하고 주변 장치를 제어할 수 있다. 예를 들어, 펌웨어는 외부 버튼 1번이 선택됨에 따라, LED를 ON시킬 수 있다.
- [0035] 상술한 펌웨어의 취약점은 하위 레벨에 상주하므로 전통적인 소프트웨어 레벨의 취약점보다 더 심각한 영향을 미칠 수 있다. 개시되는 실시예의 임베디드 퍼징 시스템(100)은 무작위로 변이된 입력을 대상에 반복적으로 공급하고, 대상의 상태를 모니터링하는 소프트웨어 테스트 자동화 기법인 퍼징(fuzzing) 기술을 적용할 수 있다.
- [0036] 상술한 퍼징 기술은 펌웨어에 적용하기에 다소 용이하지 않을 수 있다. 첫째, 일반적인 펌웨어(MCU 기반 펌웨어)는 실제 장치 작동에 필요한 모든 코드를 포함하는 싱글 바이너리 블록(single binary blob) 형태로 배포되므로, 근본적인 주변 장치(peripherals)와 직접 상호 작용할 수 있다. 이때, 직접 상호 작용은 주변 장치 레지스터에 접근하는 행위를 의미할 수 있다. 이에, 펌웨어는 실제 하드웨어 없이도 주변 장치에 대한 접근(예를 들어, 읽기, 쓰기)을 적절하게 처리할 수 있는 주변 장치 모델링이 필요할 수 있다. 또한, 전통적인 소프트웨어는 표준(standard) 또는 파일 I/O(Input/Output)를 통해 입력을 읽는 반면, 펌웨어는 주변 장치를 통해 입력을 읽을 수 있다. 즉, 펌웨어에 대한 취약점을 발견하기 위해서는 주변 장치의 퍼징 인터페이스를 식별해야 한다. 주변 장치의 레지스터는 상태 레지스터, 제어 레지스터 및 데이터 데이터로 분류되는 3개의 레지스터를 포함할 수 있다. 상기 데이터 레지스터는 원시 데이터(raw data)가 주변 장치에서 펌웨어로 전송되는 메인 채널이므로 이상적인 퍼징 인터페이스로 볼 수 있다. 이에, 개시되는 임베디드 퍼징 시스템(100)은 상술한 데이터 레지스터를 인지하여 퍼징 절차를 수행할 수 있다.
- [0037] 임베디드 퍼징 시스템(100)은 주변 장치 모델링을 위해 머신 러닝의 학습 모델인 강화 학습(reinforcement learning)을 적용할 수 있다. 상기 강화 학습은 기계 학습(machine learning)의 일종으로, 기계가 취하는 행동에 서로 다른 보상(reward)을 제시하여, 보상을 가장 많이 받을 수 있는 방식이 무언인지를 스스로 학습하도록 하는 방식을 의미할 수 있다. 이에, 개시된 실시예의 주변 장치 모델은 현재 상태(state)에서 어떤 행동(active)을 하면 가장 많은 보상(reward)을 받을지 스스로 학습할 수 있다.
- [0038] 도 3a는 이더넷 주변 장치(주변 장치)에 접근하는 펌웨어 내부 코드의 일 예를 나타내는 것으로, 도 3a의 코드에서 조건이 참이면 주변 장치를 활성화함을 의미할 수 있다. 이러한 펌웨어 내부 코드는 주변 장치로 접근을 가능하게 하는 코드일 수 있다.
- [0039] 임베디드 퍼징 시스템(100)은 도 3b의 코드 예제와 같이, srflags의 값에 따라 두 가지 경로로 펌웨어를 실행할 수 있으므로, 가장 효율적인 코드를 계속 동적으로 측정하면서 탐색할 수 있다.
- [0040] 일반적인 펌웨어 취약점 탐색 방법은 도 3b의 예제 코드를 기초로 펌웨어를 실행하여 도 5와 같이 두 개의 경로가 분기되는 경우, 하나의 경로만을 실행할 수 있다. 이때, 선택된 경로가 진입과 동시에 비정상적인 실행을 하여 에러메시지 재부팅 등의 오류가 검출되는 경우, 해당 경로를 폐기하고 분기된 나머지의 경로를 다시 재 실행하지만, 도 3b의 예제 코드의 경우에는 두 조건문에 UART 장치를 통한 데이터 송신 함수 및 UART 장치를 통한 데이터 수신 함수 모두 정상적인 실행을 수행하므로, 일반적인 펌웨어 취약점 탐색 방법을 적용하는 경우, 하나의 경로만을 실행할 수 있다. 이에, 일반적인 펌웨어의 취약점 탐색 방법은 다양한 코드 커버리지를 탐색하지 못하고, 이로 인해 잠재적인 버그 역시 발견하는 것이 다소 어려울 수 있다. 개시되는 실시예의 임베디드 퍼징 시스템(100)은 강화 학습 기반 주변 장치 모델링을 통해 하나의 경로에 국한되지 않고, 상술한 바와 같이 다양한 경로(예를 들어, 도 5의 UART 장치를 통한 데이터 송신 함수 및 UART 장치를 통한 데이터 수신 함수의 모든

경로)를 실행할 수 있다. 이를 통해, 개시되는 실시예는 하나의 경로에 국한되지 않고 다양한 경로를 탐색하면서 잠재적인 버그를 발견할 수 있다. 이에 대한 상세 설명은 후술하기로 한다.

- [0042] 도 2를 참고하면, 임베디드 퍼징 시스템(100)은 에뮬레이터 모듈(110), 주변 장치 감지 모듈(130) 및 입력 처리 모듈(150)을 포함한다. 도 1에 도시된 구성요소들은 본 개시에 따른 임베디드 퍼징 시스템(100)을 구현하는데 있어서 필수적인 것은 아니어서, 본 명세서 상에서 설명되는 임베디드 퍼징 시스템(100)은 위에서 열거된 구성요소들 보다 많거나, 또는 적은 구성요소들을 가질 수 있다.
- [0043] 도 2에 도시된 구성요소들은 통신 네트워크(미도시)를 통해 서로 간에 통신 가능하게 연결될 수 있다. 몇몇 실시예들에서, 통신 네트워크는 인터넷, 하나 이상의 로컬 영역 네트워크(local area networks), 광역 네트워크(wire area networks), 셀룰러 네트워크, 모바일 네트워크, 그 밖에 다른 종류의 네트워크들, 또는 이러한 네트워크들의 조합을 포함할 수 있다.
- [0044] 에뮬레이터 모듈(110)은 펌웨어 및 펌웨어의 내부 메모리 맵을 포함하는 구성 파일을 수신하면, 구성 파일을 기초로 펌웨어를 실행하기 위한 구성일 수 있다. 상기 펌웨어 내부 메모리 맵은 주변 장치 주소 범위 및 데이터 레지스터의 주소를 포함할 수 있다.
- [0045] 펌웨어는 실제 장치 작동에 필요한 모든 코드를 포함하므로, 주변 장치와 직접 상호 작용할 수 있다. 펌웨어는 일반적인 소프트웨어와는 달리 주변 장치 레지스터를 통해서만 입력받을 수 있다. 즉, 펌웨어는 주변 장치 레지스터에 접근할 수 있다. 이때, 이더넷 주변 장치(주변 장치)에 접근하는 펌웨어 내부 코드는 도 3a와 같을 수 있다. 도 3a에서, 조건이 참이면 주변 장치를 활성화함을 의미할 수 있다.
- [0046] 개시되는 실시예의 임베디드 퍼징 시스템(100)은 실제 하드웨어 없이도 주변 장치에 접근(예를 들어, 주변 장치 레지스터의 읽기 또는 쓰기)할 수 있도록 주변 장치 모델을 구축할 수 있다. 이때, 주변 장치 모델 구축 및 후술하는 업데이터는 입력 처리 모듈(150)을 통해 이루어질 수 있으며, 이에 한정되지 않고 별도의 구성(예를 들어, 미도시)을 통해 구현할 수 있다.
- [0047] 도 3b는 UART(Universal Asynchronus Reciver Transmitter) 주변 장치(이하, 주변 장치라 하기로 함)의 핵심 코드의 일 예시로, 전송 모드 및 수신 모드의 핵심코드를 예로 나타내는 것일 수 있다.
- [0048] 도 3b는 펌웨어 코드 내 주변 장치와 관련된 코드로, 에뮬레이터 모듈(110)은 이러한 코드를 기초로 특정 값에 따라 주변 장치(도 3의 UART 장치)를 전송 모드 또는 수신 모드로 실행시킬 수 있다.
- [0049] 주변 장치 감지 모듈(130)은 주변 장치 레지스터의 접근을 인지하고, 인지된 주변 장치 레지스터의 종류를 식별하기 위한 구성일 수 있다.
- [0050] 구체적으로, 주변 장치 감지 모듈(130)은 도 3a 및 도 3b와 같은 펌웨어 코드 내 주변 장치 레지스터에 접근 시에 이를 자동적으로 가로챌 수 있다.
- [0051] 이때, 주변 장치 레지스터의 접근은 주변 장치 레지스터의 읽기 및 쓰기를 의미할 수 있다.
- [0052] 상기 주변 장치 레지스터는 제어 레지스터, 상태 레지스터 및 데이터 레지스터를 포함할 수 있다. 이때, 제어 레지스터 및 상태 레지스터는 펌웨어 실행의 흐름에 영향을 미치는 레지스터를 의미할 수 있다. 상기 데이터 레지스터는 로우 데이터(raw data)가 주변 장치에서 펌웨어로 전송되는 메인 채널이므로, 주변감지 장치 모듈(130)은 데이터 레지스터를 퍼징 인터페이스로 식별할 수 있다.
- [0053] 도 3b를 참고하면, 주변 장치 감지 모듈(130)은 에뮬레이션된 펌웨어가 주변 장치 레지스터에 접근하는 경우, 주변 장치 모델을 통해 접근된 레지스터의 종류를 먼저 식별할 수 있다.
- [0054] 입력 처리 모듈(150)은 주변 장치 레지스터의 종류의 따라, 무작위 입력을 발생시켜 예외 핸들러(exception handler)의 발생 여부를 검사하여 버그 및 취약점을 탐지하거나, 또는 복수 개의 추론을 통한 강화 학습을 이용하여 주변 장치 모델링을 수행하는 구성일 수 있다.
- [0055] 구체적으로, 입력 처리 모듈(150)은 주변 장치 레지스터가 데이터 레지스터인 경우, 퍼징을 통해 생성된 무작위 값을 주변 장치 레지스터와 일대일 매핑할 수 있다. 이때, 입력 처리 모듈(150)은 무작위 입력을 주변 장치 레지스터에 일대일 맵핑하는 것은 버그 및 취약점 발견이 목표일 수 있다.
- [0056] 입력 처리 모듈(150)은 주변 장치 레지스터가 제어 레지스터 또는 상태 레지스터인 경우, 기호 실행을 통해 추

론된 특정 값을 레지스터와 일대일 매핑할 수 있다. 이때, 주변 장치 레지스터에 특정 값으로 응답(일대일 매핑)하는 것은 주변 장치 모델링으로 명명할 수 있다.

- [0057] 즉, 입력 처리 모듈(150)은 주변 장치 레지스터가 주변 장치 레지스터가 제어 레지스터 또는 상태 레지스터인 경우, 기호 실행을 통해 해당 주변 장치 레지스터가 가질 수 있는, 예를 들어, A, B 경로를 만족하는, 복수 개의 값을 추론할 수 있다. 입력 처리 모듈(150)은 이후 추론 값들을 지속적으로 강화학습하여 실시간으로 학습된 우선순위가 높은 추론 값을 주변 장치 레지스터에 일대일 맵핑할 수 있다. 이때, 입력 처리 모듈(150)의 동작은 완전한 펌웨어 실행과 다수의 경로를 커버링하는 것이 목표일 수 있다.
- [0058] 입력 처리 모듈(150)은 주변 장치 모델링 후 코드 커버리지(code coverage)를 측정할 수 있다. 이때, 코드 커버리지는 소프트웨어 테스트를 진행할 때 코드 자체가 얼마나 실행되었는지 여부를 의미하는 것으로, 테스트가 충분한가를 나타내는 지표일 수 있다.
- [0059] 입력 처리 모듈(150)은 주변 장치 모델링 후 하나의 경로 탐색이 종료될 때까지의 시간을 측정할 수 있다.
- [0060] 입력 처리 모듈(150)은 주변 장치 레지스터가 제어 레지스터 또는 상태 레지스터인 경우, 강화 학습을 통해 주변 장치 모델링을 지속적으로 업데이트 할 수 있다.
- [0061] 입력 처리 모듈(150)은 버그 식별을 위해 예외 핸들러를 감지할 수 있다.
- [0062] 도 3b의 라인 3 내지 5를 참고하면, 입력 처리 모듈(150)은 주변 장치 내 상태 레지스터를 리드하여 srflags 변수에 값을 할당할 수 있다. 예를 들어, srflags 변수 값은 00000000 00000000 00000000 00000000 일 수 있다.
- [0063] 다음, 도 3b의 라인 11 및 20을 참고하면, srflags 값에 따라 두 개의 분기가 발생할 수 있다.
- [0064] 이때, 상태 레지스터로부터 리드한 srflags 값에 TXE라는 플래그가 설정되면, 라인 11의 조건문을 진입할 수 있고, srflags의 값이 RXNE라는 플래그가 설정되면 라인 20의 조건문을 진입할 수 있다. 이때, 도 3 내지 도 5를 참고하면, srflags 값이 TXE라는 플래그가 설정되어 있음은 해당 값이 1인 경우를 의미하고, srflags 값이 RXNE라는 플래그가 설정되어 있음은 해당 값이 1인 경우를 의미할 수 있다.
- [0065] 도 4는 주변 장치(UART 주변 장치) 내 상태 레지스터 값의 의미를 나타내는 것으로, 상태 레지스터의 32개 비트 중 플래그마다 어떤 의미인지를 나타내는 것일 수 있다. 이때, 레지스터는 32 비트일 수 있다. 도 4를 참고하면, TXE 플래그는 오른쪽으로부터 8번째 위치하며, RXNE 플래그는 6번째 위치함을 확인할 수 있다.
- [0066] 도 5는 도 4를 참고하여 도 3b의 코드를 도식화한 것으로, 두 조건문에 진입하기 위해서는 srflags의 값이 다름을 확인할 수 있다.
- [0067] 도 3b의 라인 11을 참고하면, 입력 처리 모듈(150)은 UART_Transmit() 함수(도 5의 UART 장치를 통한 데이터 송신 함수)를 호출하는 조건문에 진입하기 위해, 16진수로 표현된 "0x80"의 값을 srflags에 할당할 수 있다.
- [0068] 도 3b의 라인 20을 참고하면, 입력 처리 모듈(150)은 UART_Receive() 함수(도 5의 UART 장치를 통한 데이터 수신 함수)를 호출하는 조건문에 진입하기 위해, 16진수로 표현된 "0x20"의 값을 srflags에 할당할 수 있다.
- [0069] 이하에서는, 주변 장치 레지스터의 종류에 따른 입력 처리 모듈(150)의 동작을 설명하기로 한다.
- [0070] 일 예로, 입력 처리 모듈(150)은 주변 장치 레지스터의 종류가 제어 레지스터 또는 상태 레지스터인 경우, 적어도 둘 이상의 추론 값을 생성하여 해당 레지스터에 대한 강화 학습을 수행할 수 있다. 이때, 입력 처리 모듈(150)은 기호 실행(symbolic execution) 기술을 이용하여 모든 주변 장치 레지스터를 기호식으로 표현하고 펌웨어 코드의 특정 위치에 도달하기 위해 필요한 제약 조건을 만족하는 추론 값과 만족하지 않는 추론 값을 생성할 수 있다. 즉, 상술한 둘 이상의 추론 값은 필요한 제안 조건을 만족하는 추론 값과 만족하지 않는 추론 값을 포함할 수 있다. 이때, 해당 레지스터는 제어 레지스터 및 상태 레지스터 중 접근이 인식된 레지스터를 의미하는 것일 수 있다.
- [0071] 입력 처리 모듈(150)은 생성된 적어도 둘 이상의 추론 값 중 적어도 하나 이상에 대한 강화 학습을 통해 주변 장치 레지스터에 최적값을 추론하여 응답할 수 있다.
- [0072] 입력 처리 모듈(150)은 해당 레지스터에 대한 강화 학습을 수행할 때, 해당 레지스터에 특정 값이 할당됨에 따라 발생된 모든 경로의 수 및 각 경로에 대한 탐색에 따른 총 소요시간을 기초로 보상값을 산출할 수 있다. 이때 보상값을 산출하는 것은 수학식 1과 같을 수 있다. 이때, 보상값은 보상점수와 동일한 의미일 수 있다.

- [0073] [수학식 1]
- [0074] 보상값(Reward) = 발견된 경로 * 100 - 시간(t)
- [0076] 이하에서는, 도 6 내지 도 9를 참고하여, 보상값을 산출하는 일 예를 설명하기로 한다.
- [0077] 도 6을 참고하면, srflags의 값("0x80"과 "0x20")에 따라 펌웨어 실행이 두 개로 분기될 수 있다.
- [0078] 도 6을 참고하면, 입력 처리 모듈(150)은 첫 번째 라운드에서, srflags 변수에 "0x80"을 할당했을 경우, 후속 노드가 1개이고 총 소요 시간이 10초 이므로, 보상값 90점을 획득할 수 있다.
- [0079] 도 7을 참고하면, 입력 처리 모듈(150)은 두 번째 라운드에서, srflags 변수에 "0x20"을 할당했을 경우, 후속 노드가 2개이고 총 소요 시간이 30 초이므로, 보상값 170점을 획득할 수 있다.
- [0080] 입력 처리 모듈(150)은 주변 장치 모델링을 통해 srflags 변수에 "0x20"을 할당하여 지속적인 탐색을 수행할 수 있다.
- [0081] 한편, 도 8을 참고하면, 데이터를 수신하는 과정에서 펌웨어 실행이 고착 상태가 되는 알 수 없는 에러로 인해 무한루프 상태가 발생할 수 있다.
- [0082] 도 7을 참고하면, 입력 처리 모듈(150)은 세 번째 라운드에서, srflags에 "0x20"을 할당하였을 때 시간이 300 초가 경과하여 보상값 -100점을 획득할 수 있다. 이때, 입력 처리 모듈(150)은 기존 srflags의 값이 "0x80"일 때 보상값이 90점이었으므로 주변 장치 모델링을 통해 네 번째 라운드에서 srflags의 값을 다시 "0x80"으로 할당할 수 있다.
- [0083] 다른 예로, 입력 처리 모듈(150)은 주변 장치 레지스터의 종류가 데이터 레지스터인 경우, 임의의 무작위 값을 생성하여 피드할 수 있다. 이때, 입력 처리 모듈(150)은 주변 장치 모델을 이용하여 피징과 같은 전략으로 임의의 무작위 값을 생성하여 피드할 수 있다. 개시되는 실시예의 입력 처리 모듈(150)은 주변 장치 모델을 이용한 펌웨어의 버그 발견 및 취약점 탐지 성능을 향상시킬 수 있다.
- [0084] 입력 처리 모듈(150)은 임의의 무작위 값을 펌웨어에 입력하고, 펌웨어의 상태를 체크하여 예외 핸들러가 호출되었는지 여부를 확인할 수 있다.
- [0085] 입력 처리 모듈(150)은 임의의 무작위 값을 펌웨어에 입력하고, 펌웨어의 상태를 체크하여 에뮬레이션된 펌웨어의 재부팅을 비롯한 에러 발생 여부를 확인할 수 있다.
- [0086] 구체적으로, 입력 처리 모듈(150)은 버그 및 취약점 탐지를 식별하기 위해 주변 장치 모델링을 수행하면서 데이터 레지스터에 무작위 입력을 입력할 수 있다. 이때, 입력 처리 모듈(150)은 에뮬레이션된 펌웨어가 현재 명령어(instruction)를 중단하고 예외 핸들러(exception handlers)로 분기되었는지 주기적으로 모니터링할 수 있다. 예외 핸들러는 유효하지 않거나 실행할 수 없는 메모리 지역에 접근하거나, 스택 오버플로우 등과 같은 다양한 이유로 트리거될 수 있다. 예외 핸들러는 실행 가능한 코드가 없는 무한 루프 형태이다. 또한, 입력 처리 모듈(150)은 에뮬레이터된 펌웨어가 강제적으로 재부팅되는 현상이 발생하면, 이를 버그로 식별할 수 있다. 상기 스택 오버플로우는 취약점의 한 종류일 수 있다.
- [0087] 에뮬레이터 모듈(110)은 상술한 펌웨어 실행이 정상적으로 종료되면, 에뮬레이션을 깨끗한 상태로 복원하고 다시 구성 파일을 기초로 펌웨어를 실행하는 과정을 수행할 수 있다.
- [0089] 도 10은 일 실시예에 따른 임베디드 피징 방법을 설명하기 위한 흐름도이다. 도 10에 도시된 방법은 예를 들어, 전술한 임베디드 피징 시스템(100)에 의해 수행될 수 있다. 도시된 흐름도에서는 상기 방법을 복수 개의 단계로 나누어 기재하였으나, 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.
- [0090] 1100 단계에서, 임베디드 피징 시스템(100)은 에뮬레이터 모듈(110)을 통해 펌웨어 및 상기 펌웨어의 내부 메모리 맵을 포함하는 구성 파일을 수신할 수 있다.
- [0091] 1200 단계에서, 임베디드 피징 시스템(100)은 에뮬레이터 모듈(110)을 통해 구성 파일을 기초로 펌웨어를 실행

할 수 있다.

- [0092] 1300 단계에서, 임베디드 퍼징 시스템(100)은 주변장치 감지 모듈(130)을 통해 주변 장치 레지스터의 접근을 인지하고, 인지된 주변 장치 레지스터의 종류를 식별할 수 있다.
- [0093] 상기 주변 장치 레지스터는 제어 레지스터, 상태 레지스터 및 데이터 레지스터를 포함할 수 있다.
- [0094] 1400 단계 내지 1800 단계에서, 임베디드 퍼징 시스템(100)은 입력 처리 모듈(150)을 통해 식별된 주변 장치 레지스터의 종류의 따라, 무작위 입력을 발생시켜 예외 핸들러의 발생 여부를 검사하여 버그 및 취약점을 탐지하거나, 또는 복수 개의 추론을 통한 강화 학습을 이용하여 주변 장치 모델링을 수행할 수 있다.
- [0095] 구체적으로, 1400 단계의 확인 결과, 주변 장치 레지스터의 종류가 상기 데이터 레지스터인 경우, 임베디드 퍼징 시스템(100)은 임의의 무작위 값을 생성하여 피드할 수 있다(1500 단계).
- [0096] 일 예로, 임베디드 퍼징 시스템(100)은 임의의 무작위 값을 펌웨어에 입력하고, 펌웨어의 상태를 체크하여 상기 예외 핸들러가 호출되었는지 여부를 확인할 수 있다.
- [0097] 다른 예로, 임베디드 퍼징 시스템(100)은 임의의 무작위 값을 펌웨어에 입력하고, 펌웨어의 상태를 체크하여 에뮬레이션된 펌웨어의 재부팅을 비롯한 에러 발생 여부를 확인할 수 있다.
- [0098] 임베디드 퍼징 시스템(100)은 버그 및 취약점 탐지 등의 퍼징 절차를 수행한 후, 버그 식별결과를 운용자 단말(미도시) 등으로 출력할 수 있다(1600 단계).
- [0099] 한편, 1400 단계의 확인 결과, 주변 장치 레지스터의 종류가 제어 레지스터 또는 상태 레지스터인 경우, 임베디드 퍼징 시스템(100)은 적어도 둘 이상의 추론 값을 생성하여 해당 레지스터에 대한 강화 학습을 수행할 수 있다(1700 단계 및 1800 단계).
- [0100] 임베디드 퍼징 시스템(100)은 생성된 적어도 둘 이상의 추론 값 중 적어도 하나 이상에 대한 강화 학습을 통해 주변 장치 레지스터에 최적값을 추론하여 응답할 수 있다.
- [0101] 이때, 임베디드 퍼징 시스템(100)은 생성된 적어도 둘 이상의 추론 값 중 적어도 하나 이상에 대한 강화 학습을 통해 주변 장치 레지스터에 최적값을 지속적으로 추론하여 응답함으로써 시간이 지남에 따라 펌웨어 코드 내 여러 위치에 도달할 수 있으며 이는 자동적으로 버그 발견 및 취약점 탐지의 가능성을 높일 수 있다.
- [0102] 개시되는 실시예에서, 주변 장치 레지스터에 추론된 값을 응답(일대일 맵핑)한다는 의미는 주변 장치 모델링으로 명명할 수 있다. 이때, 주변 장치 모델링은 주변 장치 레지스터 모델링이라 할 수 있으며, 주변 장치 레지스터에 어떤 특정 값을 응답(일대일 맵핑)하는 것을 의미할 수 있다.
- [0103] 임베디드 퍼징 시스템(100)은 해당 레지스터에 대한 강화 학습을 수행할 때, 해당 레지스터에 특정 값이 할당됨에 따라 발생된 모든 경로의 수 및 각 경로에 대한 탐색에 따른 총 소요시간을 기초로 보상값을 산출할 수 있다.
- [0104] 도시하지 않았지만, 임베디드 퍼징 시스템(100)은 입력 처리 모듈(150)을 통해 상술한 펌웨어 실행이 정상적으로 종료되면, 에뮬레이터 모듈(110)을 통해 에뮬레이션을 깨끗한 상태로 복원하고 다시 구성 파일을 기초로 펌웨어를 실행하는 1200 단계로 진입할 수 있다.
- [0106] 도 11은 일 실시예에 따른 컴퓨팅 장치를 포함하는 컴퓨팅 환경을 예시하여 설명하기 위한 블록도이다. 도시된 실시예에서, 각 컴포넌트들은 이하에 기술된 것 이외에 상이한 기능 및 능력을 가질 수 있고, 이하에 기술된 것 이외에도 추가적인 컴포넌트를 포함할 수 있다.
- [0107] 도시된 컴퓨팅 환경(10)은 컴퓨팅 장치(12)를 포함한다. 컴퓨팅 장치(12)는 일 실시예에 따른 임베디드 퍼징 시스템(100)에 포함된 하나 이상의 컴포넌트일 수 있다.
- [0108] 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 동작

들을 수행하도록 구성될 수 있다.

[0109] 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.

[0110] 통신 버스(18)는 프로세서(14), 컴퓨터 판독 가능 저장 매체(16)를 포함하여 컴퓨팅 장치(12)의 다른 다양한 컴포넌트들을 상호 연결한다.

[0111] 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.

[0112] 개시된 실시예들은 컴퓨터에 의해 실행 가능한 명령어를 저장하는 기록매체의 형태로 구현될 수 있다. 명령어는 프로그램 코드의 형태로 저장될 수 있으며, 프로세서에 의해 실행되었을 때, 프로그램 모듈을 생성하여 개시된 실시예들의 동작을 수행할 수 있다. 기록매체는 컴퓨터로 읽을 수 있는 기록매체로 구현될 수 있다.

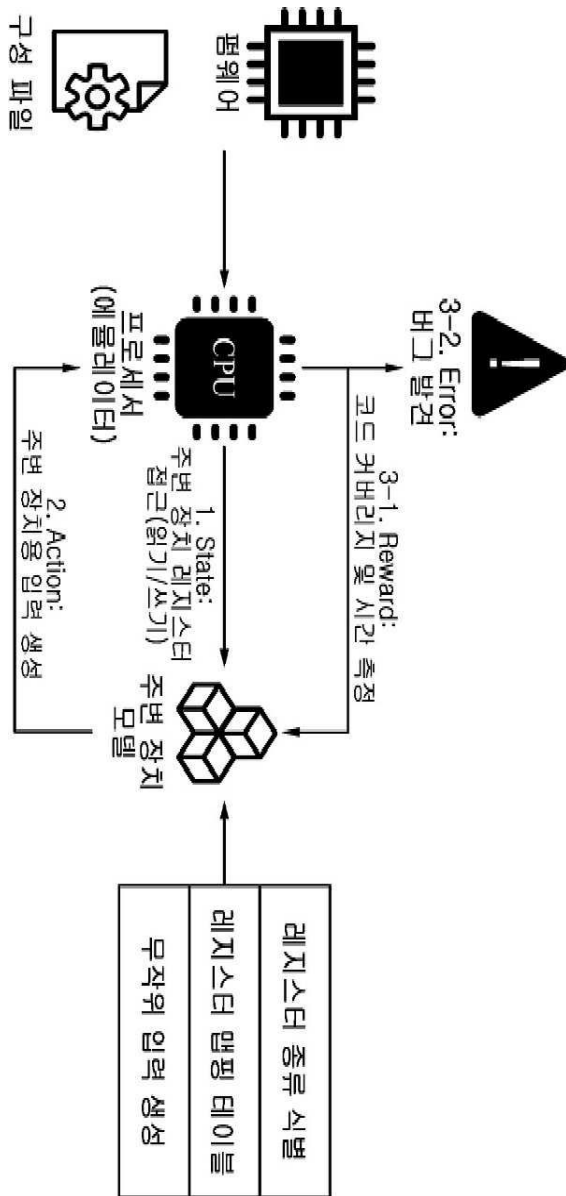
[0113] 이상에서 본 발명의 대표적인 실시예들을 상세하게 설명하였으나, 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자는 상술한 실시예에 대하여 본 발명의 범주에서 벗어나지 않는 한도 내에서 다양한 변형이 가능함을 이해할 것이다. 그러므로 본 발명의 권리범위는 설명된 실시예에 국한되어 정해져서는 안 되며, 후술하는 청구 범위뿐만 아니라 이 청구범위와 균등한 것들에 의해 정해져야 한다.

부호의 설명

- [0114]
- 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: 임베디드 펌웨어 시스템
 - 110: 애플레이터 모듈
 - 130: 주변 장치 감지 모듈
 - 150: 입력 처리 모듈

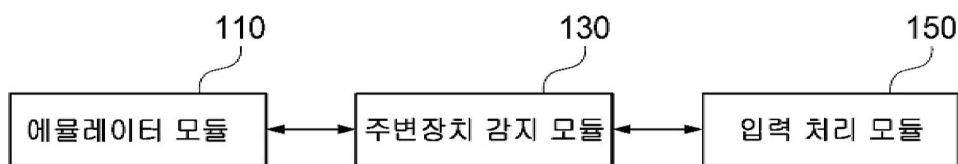
도면

도면1



도면2

100



도면3a

```

1  if (EMAC->CR & EMAC->SR == 0x1E7FF)
2      Enable_Ethernet_interrupt();
3

```

도면3b

```

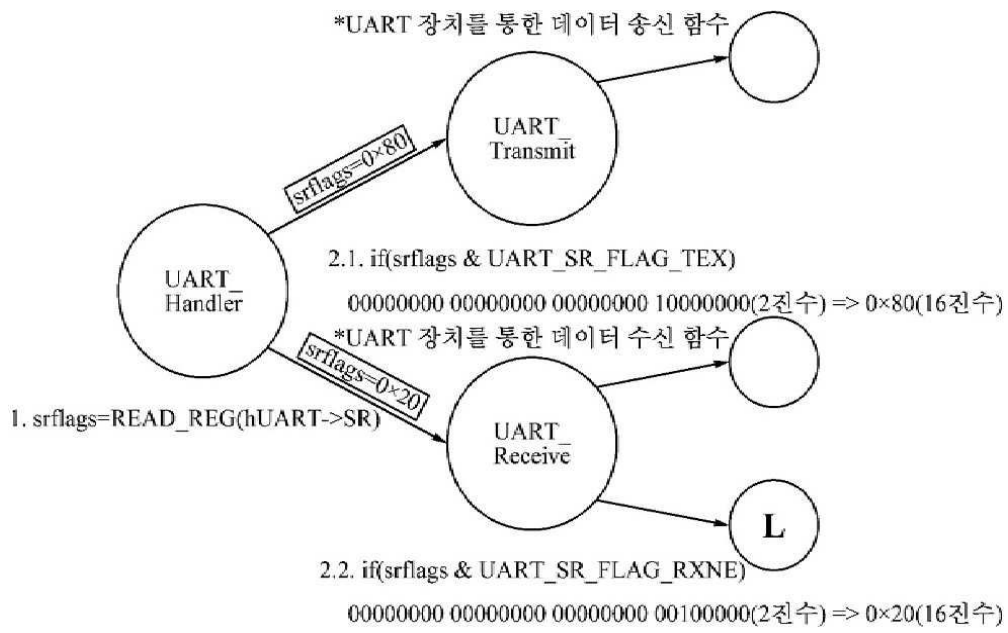
1  int UART_Handler (UART_Handle *hUART)
2  {
3      // UART 장치의 SR 레지스터로부터 (32)비트를 읽어 srflags에 할당함
4      // ex) srflags 값: 00000000 00000000 00000000 00000000
5      uint32_t srflags = READ_REG(hUART->SR);
6      uint32_t data;
7
8      /* [1] UART의 전송 모드*/
9      // SR 레지스터 내 TXE 플래그가 활성화되어있으면 (1)
10     // TXE 플래그: 데이터 송신 시 1로 SET
11     if((srflags & UART_SR_FLAG_TXE)&& ..)
12     {
13         // UART_Transmit을 호출하고 데이터를 송신함
14         UART_Transmit(hUART, hUART->DR);
15     }
16
17     /* [2] UART의 수신 모드*/
18     // SR 레지스터 내 RXNE 플래그가 활성화되어있으면 (1)
19     // RXNE 플래그: 데이터 수신 시 1로 SET
20     if((srflags & UART_SR_FLAG_RXNE)&& ..)
21     {
22         // UART_Receive를 호출하고 데이터를 수신받음
23         data = UART_Receive(hUART);
24     }
25
26     ...
27
28 }

```

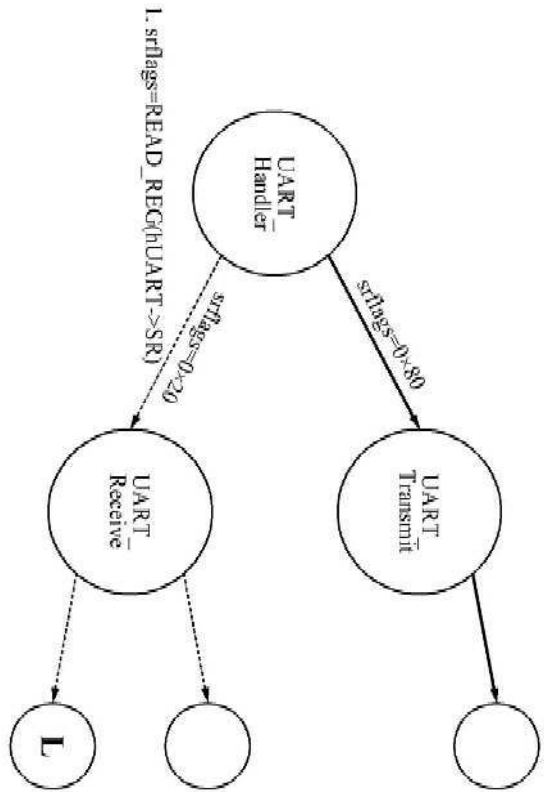
도면4

0x00		Offset
Reset value	RUART_SR	Register
Reserved		31
		30
		29
		28
		27
		26
		25
		24
		23
		22
		21
		20
		19
		18
		17
		16
		15
		14
		13
		12
		11
		10
0	CTS	9
0	LBD	8
1	TXE	7
1	TC	6
0	RXNE	5
0	IDLE	4
0	ORE	3
0	NE	2
0	FE	1
0	PE	0

도면5



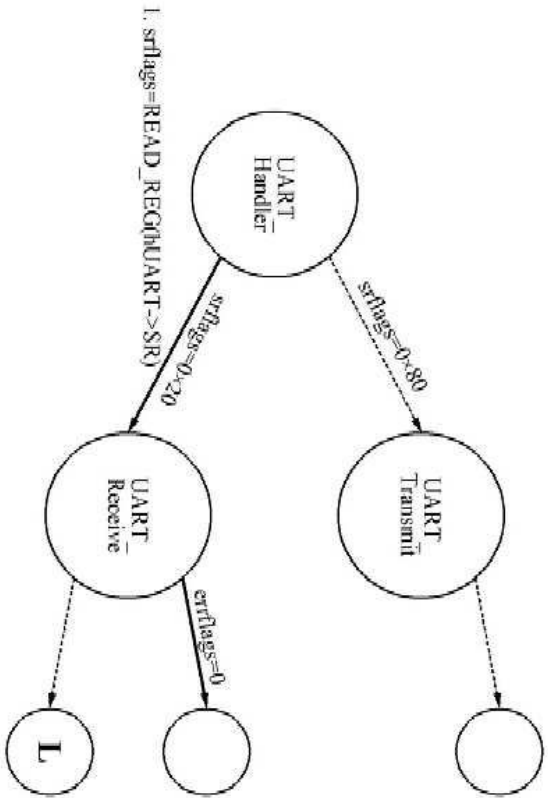
강화 학습 기반 주변 장치 모델링



도면6

Round	Variable(State)	Current Varhue (Action)	Found Path(p)	Consume Time(t)	Reward (p*100-t)
1	sflags	0×80	1	10 sec	90

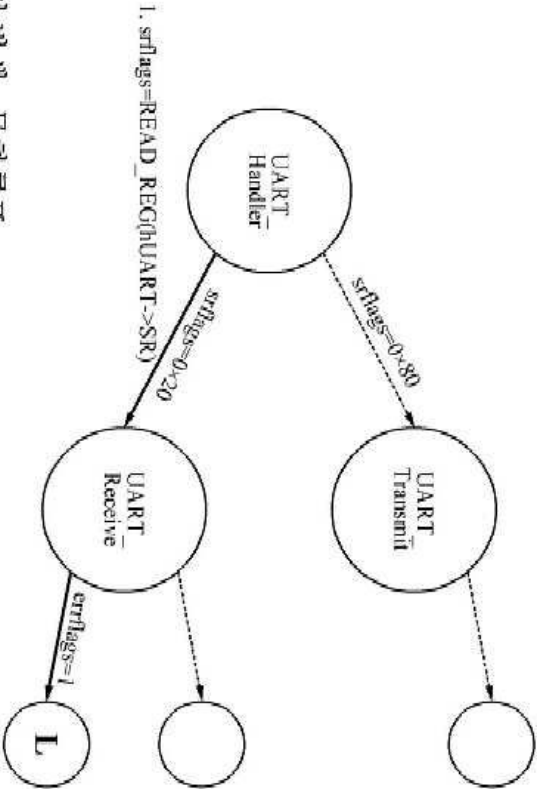
강화 학습 기반 주변 장치 모델링



도면7

Round	Variable(State)	Current Var'lue (Action)	Found Path(p)	Consume Time(t)	Reward (p*100-t)
1	srflags	0×80	1	10 sec	90
2	srflags	0×20	2	30 sec	170

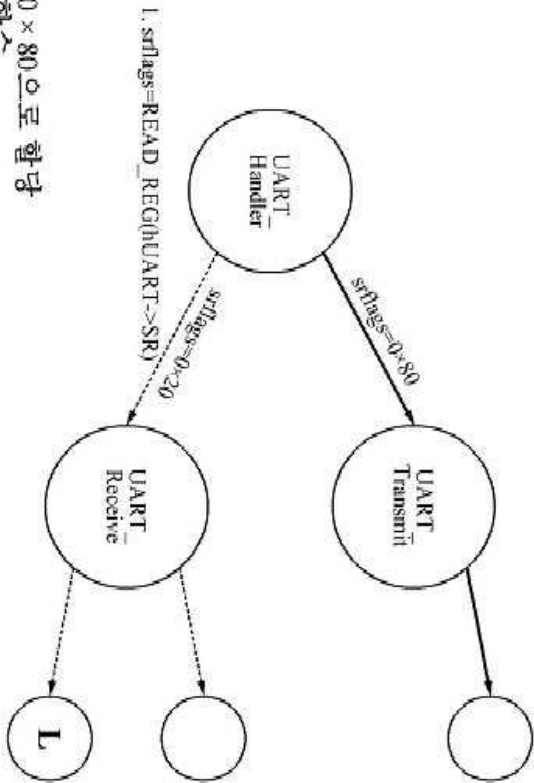
강화 학습 기반 주변 장치 모델링



수신 과정에서 에리 발생 - 무한루프

Round	Variable(State)	Current Varlue (Action)	Found Path(p)	Consume Time(t)	Reward (p≠100-t)
1	srflags	0×80	1	10 sec	90
2	srflags	0×20	2	30 sec	170
3	srflags	0×20	2	300 sec	-100

강화 학습 기반 주변 장치 모델링

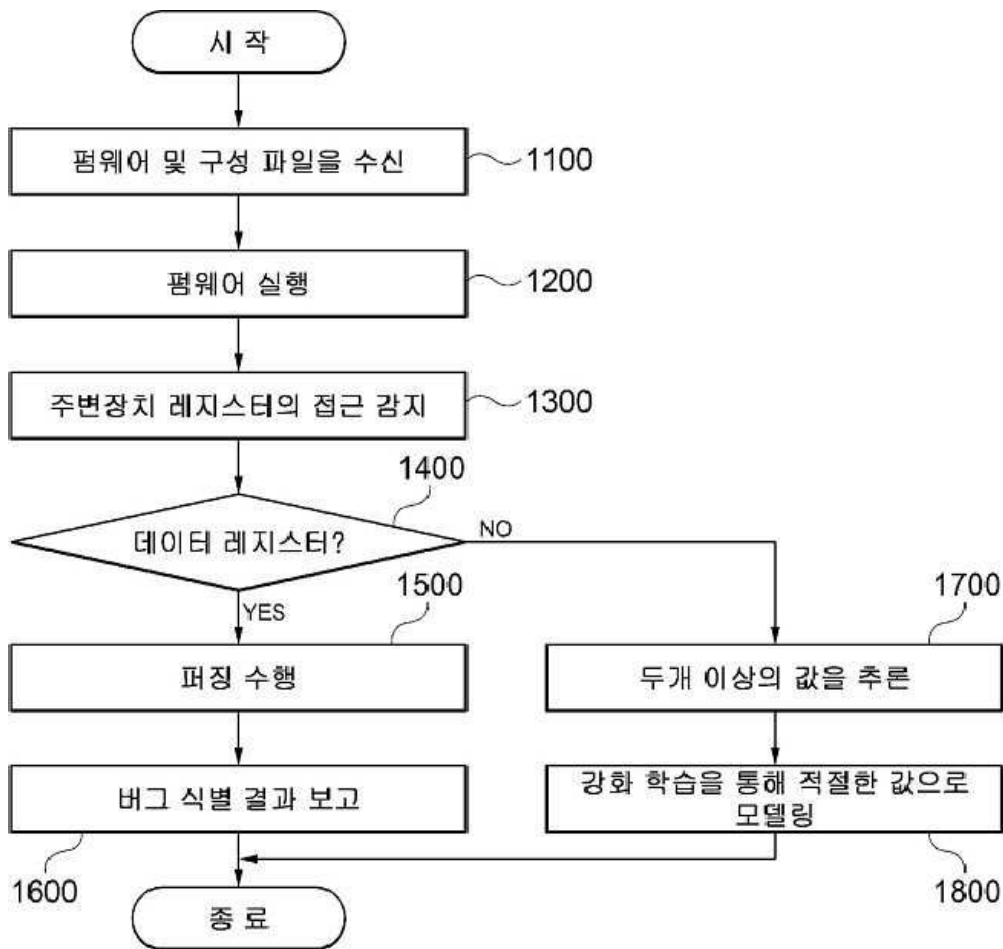


보상에 따라 다시 0×80 으로 할당
더 좋은 보상으로 학습

Round	Variable(State)	Current Varilue (Action)	Found Path(p)	Consume Time(t)	Reward (p*100-t)
1	srFlags	0×80	1	10 sec	90
2	srFlags	0×20	2	30 sec	170
3	srFlags	0×20	2	300 sec	-100
4	srFlags	0×80	1	10 sec	90

도면9

도면10



도면11

10

