(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2023년11월06일
(11) 등록번호 10-2598691
(24) 등록일자 2023년11월01일

(51) 국제특허분류(Int. Cl.)
G06F 11/36 (2006.01) *G06F 21/54* (2013.01)
G06F 21/56 (2013.01) *G06F 21/57* (2013.01)
G06N 20/00 (2019.01)
(52) CPC특허분류
G06F 11/3636 (2013.01)
G06F 11/3604 (2013.01)
(21) 출원번호 10-2023-0007409
(22) 출원일자 2023년01월18일
심사청구일자 2023년01월18일
(30) 우선권주장
1020220176168 2022년12월15일 대한민국(KR)
(56) 선행기술조사문헌
WO2018101575 A1*
(뒷면에 계속)

(73) 특허권자
이화여자대학교 산학협력단
서울특별시 서대문구 이화여대길 52 (대현동, 이화여자대학교)
(72) 발명자
최병주
서울특별시 강남구 압구정로 201, 74동 1305호(압구정동, 현대아파트)
박지현
경기도 고양시 일산동구 산두로 228, 502동 301호(정발산동, 밤가시마을5단지건영빌라)
(74) 대리인
특허법인(유한)아이시스

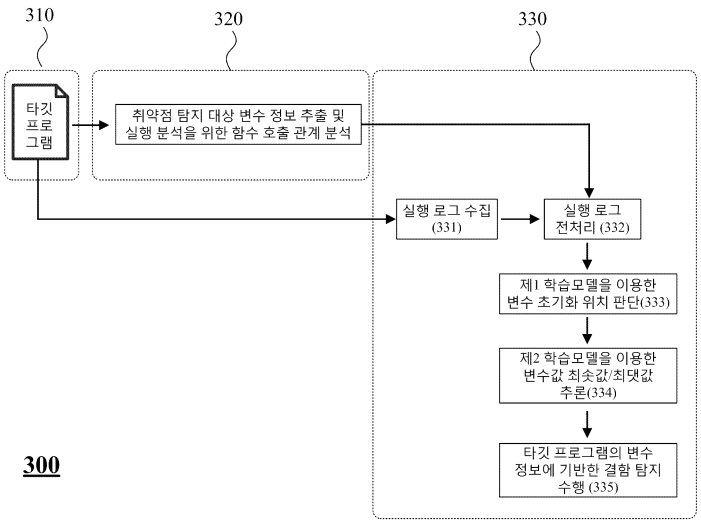
전체 청구항 수 : 총 5 항 심사관 : 김계준

(54) 발명의 명칭 **기계학습모델을 이용한 소프트웨어의 변수 취약점을 탐지하는 방법**

(57) 요 약

기계학습모델을 이용한 소프트웨어의 변수 취약점을 탐지하는 방법은 분석장치가 분석 대상인 프로그램의 소스 코드를 입력받는 단계, 상기 분석장치가 상기 소스 코드를 실행하면서 실행 로그를 추출하는 단계, 상기 분석장치가 상기 실행 로그 중 호출 함수, 변수명 및 콜스택의 함수를 각각 일정한 식별자로 대체하여 상기 실행 로그를 전처리하는 단계, 상기 분석장치가 사전 학습된 제1 학습모델을 이용하여 상기 전처리된 실행 로그를 분석하여 전역 변수와 호출 함수의 쌍들 각각이 초기화 위치인지 여부를 분류하는 단계, 상기 분석장치가 사전 학습된 제2 학습모델을 이용하여 상기 전처리된 실행 로그를 분석하여 상기 전역 변수의 변수값을 예측하는 단계 및 상기 분석장치가 상기 제1 학습모델이 출력하는 정보 및 상기 제2 학습모델이 출력하는 정보로 변수 정보를 생성하고, 상기 변수 정보를 기준으로 상기 전역 변수가 취약한지 여부를 판단하는 단계를 포함한다.

대 표 도 - 도5



(52) CPC특허분류
G06F 11/3676 (2013.01)
G06F 11/3688 (2013.01)
G06F 11/3692 (2013.01)
G06F 21/54 (2013.01)
G06F 21/563 (2013.01)
G06F 21/566 (2013.01)
G06F 21/577 (2013.01)
G06N 20/00 (2021.08)

(56) 선행기술조사문헌
KR102420514 B1*
KR101507469 B1*
KR1020060030393 A*
KR101963756 B1
*는 심사관에 의하여 인용된 문헌

이 발명을 지원한 국가연구개발사업

과제고유번호	1711174090
과제번호	RS-2022-00155966
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기획평가원
연구사업명	인공지능융합혁신인재양성
연구과제명	인공지능융합혁신인재양성(이화여자대학교)
기 여 율	1/1
과제수행기관명	이화여자대학교산학협력단
연구기간	2022.07.01 ~ 2025.12.31

명 세 서

청구범위

청구항 1

분석장치가 분석 대상인 프로그램의 소스 코드를 입력받는 단계;

상기 분석장치가 상기 소스 코드를 실행하면서 실행 로그를 추출하는 단계;

상기 분석장치가 상기 실행 로그 중 호출 함수, 변수명 및 콜스택의 함수를 각각 일정한 식별자로 대체하여 상기 실행 로그를 전처리하는 단계;

상기 분석장치가 사전 학습된 제1 학습모델을 이용하여 상기 전처리된 실행 로그를 분석하여 전역 변수와 호출 함수의 쌍들 각각이 초기화 위치인지 여부를 분류하는 단계;

상기 분석장치가 사전 학습된 제2 학습모델을 이용하여 상기 전처리된 실행 로그를 분석하여 상기 전역 변수의 변수값의 최대값과 최소값을 예측하는 단계; 및

상기 분석장치가 상기 제1 학습모델이 출력하는 정보 및 상기 제2 학습모델이 출력하는 정보로 변수 정보를 생성하고, 상기 변수 정보를 기준으로 상기 전역 변수가 취약한지 여부를 판단하는 단계를 포함하되,

상기 변수 정보는 상기 전역 변수의 초기화 위치, 변수의 사용 여부, 변수값의 변화 추세 및 변수값의 증감 패턴을 포함하고,

상기 분석장치는 상기 변수 정보를 기준으로 상기 전역 변수의 취약점을 (i)변수의 사용 도중 초기화, (ii) 허용 범위를 벗어나는 변수값 및 (iii) 초기화되지 않은 변수 사용 중 어느 하나로 판단하는 기계학습모델을 이용한 소프트웨어의 변수 취약점을 탐지하는 방법.

청구항 2

제1항에 있어서,

상기 분석장치는 상기 실행 로그의 전처리 전에 상기 소스 코드에 대한 정적 분석을 수행하여 변수들의 초기값, 함수 목록 및 함수 간 호출 관계를 추출하는 단계를 더 포함하는 기계학습모델을 이용한 소프트웨어의 변수 취약점을 탐지하는 방법.

청구항 3

제1항에 있어서,

상기 분석장치는 상기 전역 변수의 초기값이 존재하는데 실행 도중 초기값이 변경되고, 상기 초기값을 변경한 함수가 초기화 함수가 아닌 경우 상기 전역 변수의 사용 도중 초기화로 결함으로 판단하는 기계학습모델을 이용한 소프트웨어의 변수 취약점을 탐지하는 방법.

청구항 4

제1항에 있어서,

상기 분석장치는 실행 중 상기 전역 변수의 변수값이 상기 전역 변수의 자료형의 최댓값 또는 최솟값을 벗어나는 경우 결함으로 판단하는 기계학습모델을 이용한 소프트웨어의 변수 취약점을 탐지하는 방법.

청구항 5

제1항에 있어서,

상기 분석장치는 상기 전역 변수가 초기화되지 않은 상태에서 초기화함수가 아닌 다른 함수에 의해 사용되는 경우 결함으로 판단하는 기계학습모델을 이용한 소프트웨어의 변수 취약점을 탐지하는 방법.

발명의 설명

기 술 분 야

[0001] 이하 설명하는 기술은 소프트웨어의 변수 취약점을 탐지하는 기법에 관한 것이다.

배 경 기 술

[0002] 4차 산업 시대 도래와 함께 다양한 산업 분야에서 소프트웨어의 중요성이 대두되고 있다. 소프트웨어는 설계 또는 구현 과정에서 결함이 발생할 수 있다. CWE(Common Weakness Enumeration)는 소프트웨어에서 발생할 수 있는 취약점을 식별하는데 사용된다. 소프트웨어의 취약점은 변수와 관련된 취약점이 많이 있다.

[0003] 소프트웨어에서 잘못된 변수 사용은 오동작을 유발할 수 있다. 지역변수는 변수의 값을 초기화하지 않고 사용할 경우 컴파일 단계에서 결함이 검출된다. 그러나 전역변수는 초기화 코드가 없더라도 컴파일러에서 초기화시키기 때문에 결함으로 검출되지 않는다. 문제 상황은 특정한 초기값이 있음에도 불구하고 컴파일러에서 잘못된 값으로 초기화하거나, 프로그램 실행 도중 초기화가 되거나 되지 않아야 하는데 반대로 동작하는 경우이다. CWE에 따르면 이러한 변수로 인한 취약점 목록이 계속 증가하고 있다.

선행기술문헌

특허문헌

[0004] (특허문헌 0001) 한국등록특허 제10-1689795호

발명의 내용

해결하려는 과제

[0005] 종래 소프트웨어의 취약점 탐지 도구는 소스 코드에 대한 정적 분석을 수행한다. 정적 분석으로 탐지되는 런타임 결함은 실제 실행에 기반하여 탐지되는 것이 아니므로, 정적 분석은 실제 결함이 아닌데 결함으로 탐지하는 경우가 많다. 소프트웨어의 결함 탐지의 정확성을 높이기 위해서는 실제 실행에 기반한 결함 탐지 및 동적 검증이 필요하다. 그러나 동적 검증은 결함 탐지를 위한 정보 수집이나 실행 추적으로 인해 오버헤드가 크다. 특히 변수로 인한 결함을 탐지하는 경우 해당 변수가 사용되는 위치를 모두 모니터링하여야 하므로 실행 오버헤드가 매우 크다. 나아가, 종래 기법은 결함이 존재하더라고 실행이 되지 않는다면 결함으로 탐지되기 어렵다는 근본적인 문제점이 있다.

[0006] 이하 설명하는 기술은 정적 분석과 동적 검증을 혼합하여 소프트웨어 결함을 탐지하는 기법을 제공하고자 한다. 특히 이하 설명하는 기술은 기계학습 기법을 이용하여 잘못된 변수로 인한 결함을 탐지하는 기법을 제공하고자 한다.

과제의 해결 수단

[0007] 기계학습모델을 이용한 소프트웨어의 변수 취약점을 탐지하는 방법은 분석장치가 분석 대상인 프로그램의 소스 코드를 입력받는 단계, 상기 분석장치가 상기 소스 코드를 실행하면서 실행 로그를 추출하는 단계, 상기 분석장치가 상기 실행 로그 중 호출 함수, 변수명 및 콜스택의 함수를 각각 일정한 식별자로 대체하여 상기 실행 로그를 전처리하는 단계, 상기 분석장치가 사전 학습된 제1 학습모델을 이용하여 상기 전처리된 실행 로그를 분석하여 전역 변수와 호출 함수의 쌍들 각각이 초기화 위치인지 여부를 분류하는 단계, 상기 분석장치가 사전 학습된 제2 학습모델을 이용하여 상기 전처리된 실행 로그를 분석하여 상기 전역 변수의 변수값의 최대값과 최소값을 예측하는 단계 및 상기 분석장치가 상기 제1 학습모델이 출력하는 정보 및 상기 제2 학습모델이 출력하는 정보로 변수 정보를 생성하고, 상기 변수 정보를 기준으로 상기 전역 변수가 취약한지 여부를 판단하는 단계를 포함한다.

발명의 효과

[0008] 이하 설명하는 기술은 잘못된 변수로 인한 결함을 정확하게 탐지할 수 있다.

도면의 간단한 설명

[0009] 도 1은 소프트웨어의 변수 취약점을 탐지하는 시스템에 대한 예이다.

도 2는 분석 모델의 학습 과정에 대한 예이다.

도 3은 변수의 초기화 위치를 추론하는 제1 학습모델의 성능을 평가한 결과이다.

도 4는 변수의 최댓값 및 최솟값을 추론하는 제2 학습모델의 성능을 평가한 결과이다.

도 5는 소프트웨어의 변수 취약점을 탐지하는 추론 과정에 대한 예이다.

도 6은 분석 장치에서 분석 모델을 이용하여 결함을 분석할 때 사용하는 변수에 대한 정보의 예이다.

도 7은 타깃 프로그램의 변수 정보를 이용하여 취약점으로 인한 결함을 판단하는 과정의 예이다.

도 8은 소프트웨어의 변수 취약점을 탐지하는 분석장치에 대한 예이다.

발명을 실시하기 위한 구체적인 내용

[0010] 이하 설명하는 기술은 다양한 변경을 가할 수 있고 여러 가지 실시예를 가질 수 있는 바, 특정 실시예들을 도면에 예시하고 상세하게 설명하고자 한다. 그러나, 이는 이하 설명하는 기술을 특정한 실시 형태에 대해 한정하려는 것이 아니며, 이하 설명하는 기술의 사상 및 기술 범위에 포함되는 모든 변경, 균등물 내지 대체물을 포함하는 것으로 이해되어야 한다.

[0011] 제1, 제2, A, B 등의 용어는 다양한 구성요소들을 설명하는데 사용될 수 있지만, 해당 구성요소들은 상기 용어들에 의해 한정되지는 않으며, 단지 하나의 구성요소를 다른 구성요소로부터 구별하는 목적으로만 사용된다. 예를 들어, 이하 설명하는 기술의 권리 범위를 벗어나지 않으면서 제1 구성요소는 제2 구성요소로 명명될 수 있고, 유사하게 제2 구성요소도 제1 구성요소로 명명될 수 있다. 및/또는 이라는 용어는 복수의 관련된 기재된 항목들의 조합 또는 복수의 관련된 기재된 항목들 중의 어느 항목을 포함한다.

[0012] 본 명세서에서 사용되는 용어에서 단수의 표현은 문맥상 명백하게 다르게 해석되지 않는 한 복수의 표현을 포함하는 것으로 이해되어야 하고, "포함한다" 등의 용어는 설명된 특징, 개수, 단계, 동작, 구성요소, 부분품 또는 이들을 조합한 것이 존재함을 의미하는 것이지, 하나 또는 그 이상의 다른 특징들이나 개수, 단계 동작 구성요소, 부분품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 배제하지 않는 것으로 이해되어야 한다.

[0013] 도면에 대한 상세한 설명을 하기에 앞서, 본 명세서에서의 구성부들에 대한 구분은 각 구성부가 담당하는 주기능 별로 구분한 것에 불과함을 명확히 하고자 한다. 즉, 이하에서 설명할 2개 이상의 구성부가 하나의 구성부로 합쳐지거나 또는 하나의 구성부가 보다 세분화된 기능별로 2개 이상으로 분화되어 구비될 수도 있다. 그리고 이하에서 설명할 구성부 각각은 자신이 담당하는 주기능 이외에도 다른 구성부가 담당하는 기능 중 일부 또는 전부의 기능을 추가적으로 수행할 수도 있으며, 구성부 각각이 담당하는 주기능 중 일부 기능이 다른 구성부에 의해 전담되어 수행될 수도 있음은 물론이다.

[0014] 또, 방법 또는 동작 방법을 수행함에 있어서, 상기 방법을 이루는 각 과정들은 문맥상 명백하게 특정 순서를 기재하지 않은 이상 명기된 순서와 다르게 일어날 수 있다. 즉, 각 과정들은 명기된 순서와 동일하게 일어날 수도 있고 실질적으로 동시에 수행될 수도 있으며 반대의 순서대로 수행될 수도 있다.

[0015] 이하 설명하는 기술은 소프트웨어에서 변수 값에 따른 결함 가능성 내지 취약점을 탐지하는 기법이다.

[0016] 이하 소프트웨어의 변수 취약점을 탐지하는 장치를 분석장치라고 명명한다. 분석장치는 입력되는 소프트웨어 코드를 분석하여 결함을 탐지하는 장치로서 다양한 형태로 구현될 수 있다. 예컨대, 분석장치는 서버, PC, 스마트기기, 취약점 탐지를 위한 프로그램이 임베딩된 칩셋 등 다양한 형태일 수 있다.

[0017] 분석 장치는 기계학습모델을 이용하여 변수 취약점을 탐지한다. 기계 학습은 인공 지능의 한 분야로, 컴퓨터가 학습할 수 있도록 알고리즘을 개발하는 분야를 의미한다. 기계학습모델은 결정 트리, 랜덤 포레스트(random forest), KNN(K-nearest neighbor), 나이브 베이즈(Naive Bayes), SVM(support vector machine), 인공신경망(artificial neural network) 등 다양한 유형의 모델이 있다.

[0018] 이하 소프트웨어의 변수 취약점을 탐지하는 기계학습모델을 분석 모델이라고 명명한다. 분석 모델은 다양한 기계학습 방법론 중 어느 하나로 구현될 수 있다.

[0020] 도 1은 소프트웨어의 변수 취약점을 탐지하는 시스템(100)에 대한 예이다. 도 1에서 분석장치(130, 150)가 변수

취약점을 탐지한다. 도 1에서 분석장치는 서버(130) 및 컴퓨터 단말(150) 형태로 도시하였다.

[0021] 소스 코드 제공 장치(110)는 개발자가 코딩한 특정 프로그램의 소스 코드를 제공한다. 소스 코드 제공 장치(110)는 개발자 단말 또는 소스 코드를 저장하는 데이터베이스 내지 서버와 같은 장치일 수 있다.

[0022] 실행 로그 제공 장치(120)는 프로그램 소스코드를 전달받아 해당 프로그램을 실행하면서 실행 로그를 생성한다. 실행 로그 제공 장치(120)는 해당 프로그램이 동작하는 환경을 제공하여 실행 시간에 따라 실행 로그를 생성한다.

[0023] 분석장치(130, 150)는 소스 코드 제공 장치(110)로부터 분석 대상인 프로그램의 소스 코드를 입력받는다. 분석장치(130, 150)는 실행 로그 제공 장치(120)로부터 분석 대상인 프로그램의 실행 로그를 입력받는다.

[0024] 분석장치(130, 150)는 소스 코드 및 실행 로그를 분석하여 변수 취약점을 탐지한다. 분석장치(130, 150)는 분석 모델을 활용하여 일정한 변수 정보를 추출한다. 변수 정보는 변수 초기화 위치, 실행 과정 중 변수의 최댓값 및 최솟값 등을 포함한다. 분석장치(130, 150)는 추출한 변수 정보를 기준으로 분석 대상인 프로그램에서 변수 취약점 내지 결함 가능성을 도출한다. 분석장치(130, 150)는 분석한 변수 취약점을 출력하거나, 다른 단말에 전달할 수 있다.

[0026] 이하 연구자가 변수 취약점을 탐지하는 분석모델을 구축한 과정을 설명한다.

[0027] 연구자는 잘못된 변수 값 사용으로 인한 취약점을 분류하여 결함 유형을 아래 표 1과 같이 정의하였다.

표 1

[0029]

결함 ID	설명	관련된 CWE-IDs
F01 UINIT_VAR	초기화되지 않은 변수의 사용	CWE-457
F02 INVALID_INIT	변수의 사용 도중 초기화	CWE-481
F03 OUT_OF_BOUND	변수값이 허용 범위 벗어남	CWE-125, 128, 130, 131, 197, 805, 839, 843

[0030] F01 UINIT_VAR는 초기화되지 않은 변수를 사용하여 발생하는 결함이다. 초기화되지 않은 지역 변수는 대부분 컴파일러에서 탐지가 되고, 전역 변수는 컴파일러에서 초기화된다. 따라서, UINIT_VAR 결함은 지역 변수나 전역 변수가 올바른 값으로 초기화되지 않은 경우의 결함이다. 또한 UINIT_VAR 결함은 전역 변수의 초기화 코드가 존재하더라도 함수 호출 순서가 잘못되는 것과 같이 초기화되기 전에 사용되는 경우의 결함도 포함한다.

[0031] F02 INVALID_INIT은 변수의 초기화 함수가 아닌 다른함수에서 변수를 사용하던 도중 변수의 값이 초기값으로 변경되는 결함이다. F03 OUT_OF_BOUND는 변수의 값이 허용된 범위를 벗어나는 결함이다. 이 때의 허용 범위는 변수의 자료형에 따른 범위뿐만 아니라 소스 코드 분석 시 추출되는 변수의 허용 범위를 포함한다.

[0033] 분석 모델의 구축 과정을 설명한다. 분석 모델 학습은 학습 장치가 수행한다고 설명한다. 학습 장치는 데이터 처리 및 기계학습모델의 학습 과정 수행이 가능한 컴퓨터 장치를 의미한다.

[0034] 연구자는 분석 모델의 학습을 위하여 학습 데이터를 마련하였다. 학습 데이터는 특정한 동일 프로그램에 대하여 마련될 수 있다. 즉, 분석 모델은 특정 프로그램에 특이적으로 마련될 수 있다. 학습 데이터는 프로그램에 대한 소스 코드 및 실행로그를 포함할 수 있다.

[0035] 취약점을 탐지하고자 하는 대상은 프로그램에서 사용되는 지역변수와 전역변수를 모두 포함한다. 표 1에서 정의한 결함을 탐지하기 위하여 각 변수에 대하여 (i) 변수의 최댓값 및 최솟값, (ii) 초기값 및 (ii) 초기값으로 변경하도록 하는 초기화 함수가 정의되어야 한다.

[0037] 도 2는 분석 모델의 학습 과정(200)에 대한 예이다.

[0038] 학습 장치는 학습을 위한 프로그램에 대하여 정적 분석을 수행한다(210). 학습 장치는 프로그램의 소스 코드를 정적 분석하여 변수에 대한 정보를 추출한다.

[0039] 학습 장치는 학습을 위한 프로그램의 소스 코드를 정적 분석하여 변수 목록, 각 변수의 초기값, 최댓값, 최솟값을 변수 정보로 추출한다. 또한 학습 장치는 실행 분석을 위해 변수를 사용하는 함수 목록과 함수 간 호출 관계를 추출한다. 정적 분석으로 정의되지 않은 변수의 최댓값, 최소값 및 초기화 함수 목록은 실행 로그를 학습할 때 정의된다.

[0040] 이후 학습 장치는 프로그램을 실행하며 수집된 실행 로그를 이용한 동적 검증을 수행한다(220). 동적 검증은 실행 로그를 기준으로 변수 취약점을 탐지하는 분석 모델을 구축하는 과정이다. 분석 모델은 취약점 탐지 대상인 초기화 함수의 위치를 판정하는 모델과 함수에서 사용되는 변수값을 추론하는 모델을 포함한다.

[0041] 학습 장치는 정적 분석 후에는 프로그램을 실행하며 실행 로그를 수집한다(221). 학습 장치는 프로그램이 실행되는 동안 모니터링 대상 변수에 접근하여 호출 함수 위치, 변수명, 접근 전/후 변수값, 콜스택을 추출하여 실행 로그로 수집한다.

[0042] 실행 로그는 학습에 직접 이용하기 어려운 로데이터(raw date)이므로, 일정한 전처리가 필요하다. 학습 장치는 수집한 실행 로그를 학습에 활용할 수 있는 데이터로 전처리(일반화)한다(222). 전처리 과정에서 학습 장치는 초기 실행 로그에서 호출 함수, 변수명 및 콜스택의 함수를 각각 함수 ID 및 변수 ID로 대체 한다.

[0044] 변수의 초기화와 관련된 취약점을 판정하기 위한 기준은 변수의 초기화 여부, 변수의 값 변경 및 호출 함수와 관련있다. 연구자는 변수를 사용중인 함수(호출 함수)가 변수의 값을 초기화시키는 함수(초기화 함수)인지 여부를 판단하기 위해 기계학습모델(제1 학습모델)을 이용하였다. 제1 학습 모델은 {전역 변수, 호출 함수}의 쌍이 초기화 함수인지 아닌지를 분류한다.

[0045] 학습 장치는 전처리된 실행 로그를 이용하여 변수의 초기화와 관련된 취약점 판단을 위한 초기화 함수 여부를 추론하는 제1 학습모델의 학습을 수행한다(223). 제1 학습모델은 전처리된 실행 로그를 입력받아 {전역 변수, 호출 함수}의 쌍이 초기화 함수인지 여부를 분류한다. 이를 통해 제1 학습 모델은 변수의 초기화 위치를 추론한다.

[0046] 연구자는 제1 학습 모델로 결정 트리 기법을 사용하였다. 결정 트리 기법은 ID3, C4.5, CART, 앙상블 알고리즘 등 다양한 알고리즘들이 있다.

[0047] 연구자는 초기화 함수의 예측과 비슷한 데이터셋인 NASA Nearest Object 데이터셋을 이용하여 제1 학습 모델의 성능을 평가하였다. 연구자는 다양한 유형의 결정 트리를 구축하여 성능을 평가하였다.

[0048] 도 3은 변수의 초기화 위치를 추론하는 제1 학습모델의 성능을 평가한 결과이다. 도 3과 같이 연구자가 구축한 모델 중 RF(RandomForest) 기법이 가장 높은 성능을 보였다. 이에 연구자는 제1 학습모델로 RF를 사용하였다. 다만, 제1 학습모델은 결정 트리 외에 다른 유형의 학습모델이 활용될 수도 있다.

[0050] 또한 연구자는 변수값이 적절한가를 판정하기 위하여 변수값 허용 범위 여부를 판정하는 별도의 기계학습모델(제2 학습모델)을 구축하였다. 제2 학습모델은 특정 변수의 최댓값 및 최솟값을 도출한다.

[0051] 학습 데이터는 실행 로그에서 변수값의 변화 정보이다. 학습 장치는 전처리된 실행 로그를 이용하여 제2 학습모델이 특정 변수의 변화 추이를 추론하여 해당 변수의 최댓값 및 최솟값을 도출하도록 학습 과정을 수행한다(224). 연구자는 제2 학습모델로 회귀분석 기법을 이용하였다. 회귀 기반 학습 기법은 선형 회귀(linear), 능형 회귀(ridge). 로지스틱 회귀(logistic), 다항 회귀(polynomial)와 같은 다양한 기법이 있다.

[0052] 연구자는 다양한 회귀 분석 모델을 구축하고 성능을 평가하였다. 도 4는 변수의 최댓값 및 최솟값을 추론하는 제2 학습모델의 성능을 평가한 결과이다. 도 4와 같이 다항 회귀 분석의 성능이 가장 높았다. 연구자는 제2 학습모델로 다항 회귀 분석을 사용하였다. 연구자는 scikit-learn의 polynomialFeatures 클래스로 데이터 변환 후 LinearRegression을 적용하여 제2 학습모델을 구축하였다.

[0054] 도 5는 소프트웨어의 변수 취약점을 탐지하는 추론 과정(300)에 대한 예이다.

[0055] 분석장치는 분석 대상인 타깃 프로그램의 소스 코드를 입력받는다(310).

[0056] 분석장치는 타깃 프로그램에 대하여 정적 분석을 수행할 수 있다(310). 분석장치는 타깃 프로그램의 소스 코드를 정적 분석하여 변수 목록을 추출할 수 있다. 또한, 분석장치는 실행 분석을 위해 변수를 사용하는 함수 목록과 함수 간 호출 관계를 추출할 수 있다. 한편, 타깃 프로그램이 학습 과정에서 사용된 프로그램과 동일한 프로그램이라면 학습 과정에서 도출한 정적 분석 결과를 활용할 수도 있다. 이 경우 분석장치가 별도로 정적 분석을 수행하지 않을 수 있다. 즉, 분석장치가 타깃 프로그램의 소스 코드와 함께 변수 목록, 함수 목록 및 함수 간 호출 관계를 입력받을 수도 있다.

[0057] 분석장치는 프로그램을 실행하며 수집된 실행 로그를 이용한 동적 검증을 수행한다(330). 분석장치는 프로그램을 실행하며 실행 로그를 수집한다(331). 분석장치는 프로그램이 실행되는 동안 모니터링 대상 변수에 접근하여 호출 함수 위치, 변수명, 접근 전/후 변수값, 콜스택을 추출하여 실행 로그로 수집한다. 분석장치는 수집한 실

행 로그를 학습에 활용할 수 있는 데이터로 전처리(일반화)한다(332). 전처리 과정에서 학습 장치는 초기 실행 로그에서 호출 함수, 변수명 및 콜스택의 함수를 각각 함수 ID 및 변수 ID로 대체 한다.

[0058] 분석장치는 사전에 학습된 제1 학습 모델을 이용하여 전처리된 실행 로그를 분석하여 변수가 올바른 위치에서 초기화 되었는지를 판단한다(333).

[0059] 분석장치는 사전에 학습된 제2 학습 모델을 이용하여 실행 과정에서 변수의 값이 변수의 최대값과 최소값 이내의 범위에 해당하는지를 판단한다(334).

[0060] 분석장치는 변수 초기화 위치, 초기화 정보, 변수 사용 정보, 변수값 변화 정보, 변수의 허용 범위 등을 기준으로 특정 변수의 취약점 여부를 분석한다. 이를 통해 분석장치는 타깃 프로그램에서 결함 탐지를 수행한다(335). 구체적인 결함 타지 예는 후술한다.

[0062] 전술한 과정을 통해 분석장치는 타깃 프로그램의 소스 코드 및 실행 로그를 분석하여 변수에 대한 정보를 도출한다. 도 6은 분석 장치에서 분석 모델을 이용하여 결함을 분석할 때 사용하는 변수에 대한 정보의 예이다.

[0064] 분석 모델을 이용하여 도출하는 변수에 대한 정보에 대한 예이다. 도 6은 다수의 플래그(flag)로 구성된 변수 정보에 대한 예이다. 변수 정보는 (i) 초기화 정보, (ii) 변수 사용 정보, (iii) 변수값 변화 추세 정보 및 (iv) 변수값 증감 패턴 정보를 포함한다. 초기화 정보는 컴파일러에서 초기화 여부, 초기화 함수에서 초기화 여부 및 변수 선언 시 초기값 존재 여부 등을 포함할 수 있다. 변수 사용 정보는 변수의 사용 여부, 사용되는 변수의 사용 목적(read, write 또는 read & write) 등을 포함할 수 있다. 변수값 변화 추세 정보는 변수값의 변화 여부, 변수값의 증감 여부 등을 포함할 수 있다. 변수값 증감 패턴 정보는 패턴의 변화 여부, 변수값 계속 증가, 변수값 계속 감소, 일정 구간에서 변수값의 파동(fluctuation) 변화 등을 포함할 수 있다.

[0065] 분석장치는 타깃 프로그램에서 타깃 프로그램의 소스 코드를 분석한 결과 및 타깃 프로그램의 실행 로그를 이용하여 추출한 변수 정보를 이용하여 변수 취약점을 탐지한다(335). 분석장치는 타깃 프로그램의 실행 시작부터 종료까지 변수가 실행될 때 남긴 실행 로그로부터 계속 변수 정보를 추출하여 이를 기준으로 변수 취약점을 탐지할 수 있다.

[0067] 도 7은 타깃 프로그램의 변수 정보를 이용하여 취약점으로 인한 결함을 판단하는 과정의 예이다. 분석장치가 판단하는 변수 취약점 유형은 (i)변수의 사용 도중 초기화, (ii) 허용 범위를 벗어나는 변수값 및 (iii) 초기화되지 않은 변수 사용 중 어느 하나이다.

[0068] 분석장치는 변수의 초기값이 존재하는데 전역 변수의 사용 도중 변경된 값이 초기값이고, 호출한 함수가 초기화 함수가 아닌 경우에는 전역 변수의 사용 도중 초기화 결함으로 판단한다.

[0069] 결함 탐지 대상을 추출할 때 전역 변수의 이름과 자료형, 초기값을 추출하는데, 분석장치는 실행 중 변수값이 추출된 자료형의 최댓값/최솟값을 벗어나는 경우에는 변수의 범위를 벗어난 결함으로 판단한다. 또한, 분석장치는 전역 변수의 값이 계속해서 증가 혹은 감소만하면서 제2 학습 모델로부터 도출된 변수의 최대값과 최소값 범위를 벗어나는 경우에도 변수의 범위를 벗어난 결함으로 판단한다.

[0070] 분석장치는 모든 실행이 완료된 이후에도 전역 변수의 변수값 증감 패턴 정보가 계속 증가 또는 감소로 설정되어 있다면, 변수의 사용 도중 초기화가 되지 않는 것으로 볼 수 있으며 장시간 사용 시 변수의 값의 범위를 벗어날 수 있으므로 이를 변수의 범위를 벗어난 결함 후보로 판단한다.

[0071] 분석장치는 변수의 값이 선언과 동시에 초기화가 되지 않은 상태에서, 초기화 함수가 아닌 다른 함수에 의해 사용되었을 때를 초기화하지 않은 변수 사용 결함으로 판단할 수 있다. 분석장치는 최초 사용된 위치가 변수에 대한 값을 읽어가는(read) 경우에는 바로 초기화하지 않은 변수 사용 결함으로 판단한다. 분석장치는 변수에 값을 쓰는(write) 경우에는 해당 변수에 값을 쓴 함수가 초기화 함수로 학습이 될 수 있으므로 초기화하지 않은 변수 사용 결함 후보로 판단한다.

[0073] 연구자는 구축한 분석 모델 및 변수 취약점 탐지의 성능을 검증하였다.

[0074] 검증 실험에 사용된 애플리케이션은 아래 표 2와 같다. 연구자는 리눅스에서 잘 알려진 7개의 어플리케이션을 대상으로 실험하였다. 학습에 사용된 실행 로그는 각 어플리케이션에 대해 리눅스 매뉴얼 페이지의 샘플 스크립트와 SARD(Software Assurance Reference Dataset)에서 제공하는 테스트케이스, Linux Test Project(LTP)의 테스트 케이스를 동작시켜 수집하였다.

표 2

[0076]

Application			학습에 사용된 실행로그 수집 TestCase
이름	설명	사이즈(LOC)	
grep	검색 유틸리티	5317,	linux manual-sample scripts, SARD
gzip	압축 프로그램	7592	linux manual-sample scripts, LTP
bc	쉘 계산기	8803	linux manual-sample scripts
wc	워드 카운터	1126	linux manual-sample scripts, LTP
sort	정렬	4850	linux manual-sample scripts
sed	스트림 에디터	5234	linux manual-sample scripts
jpegtran	jpeg 변환	31839	linux manual-sample scripts

[0078] 연구자는 변수로 인한 취약점을 탐지하기 위해 애플리케이션에 인위적으로 결함을 주입하였다. 연구자는 종래 연구 clang-sfi(Gabor et al, "High-Accuracy Software Fault Injection in Source Code with Clang." In 2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC), pp. 75-7509.)에서 활용된 데이터를 이용하였다. clang-sfi에서 주입 가능한 결함 중 연구자가 제안한 방식과 관련된 변수 취약점에 대한 결함 유형은 아래 표 3과 같다.

표 3

[0080]

결함 유형	Description	주입 예	
		결함 주입 전	결함 주입 후
MVAE	Missing variable assignment with an expression	hash ^= s -> elems[i] . index + s -> elems[i] . constraint;	-
MVAV	Missing variable assignment using a value	options = 0	-
MVIV	Missing variable initialization using a value	int options = 0;	int options;
WVAV_1	Wrong value assigned to variable	state_newline = 0	state_newline = 1894; (clang-sfi에서 리턴된 값)
WVAV_2			state_newline =INT_MIN; state_newline =INT_MAX;
WVAV_3			state_newline = rand()

[0082] 연구자는 표 2의 애플리케이션에서 지역 변수와 전역 변수를 각각 추출하고, 추출된 각 위치에 표 3의 결함유형을 주입하였다. WVAV(Wrong value assigned to variable) 결함 유형은 clang-sfi에서 Boolean 타입인 경우 토글시키고, 그 외의 경우에는 0xFF 값을 오류 값으로 주입한다. 그러나 보다 다양한 경우를 검증하기 위해 연구자는 clang-sfi에서 주입하는 유형을 WVAV_1으로 하고, 해당 변수의 타입에 대한 최댓값과 최솟값을 주입하는 경우를 WVAV_2로 하고, 해당 변수가 가질 수 있는 값에서 랜덤하게 선택한 값으로 주입하는 경우를 WVAV_3으로 정의하였다.

[0084] 연구자는 제안한 분석 모델 기반의 변수 취약점을 탐지 기법(제안 기술)의 성능을 종래 분석 도구의 성능과 비교하였다. 비교에 사용한 종래 분석 도구는 아래 표 4와 같다.

표 4

[0086]

구분		도구명	탐지 가능한 결함 유형	AI/ML 적용 여부
정적	코드 유사도	VUDDY	CWE 취약점	X
		SySeVR	CWE 취약점	O
	패턴 기반	cppcheck	초기화하지 않은 변수 사용, 형 변환 오류, 배열의 범위를 벗어난 접근 등	X
		controlflag	CWE 취약점	O

[0088] 연구자는 특히 변수 취약점을 탐지하는 성능을 비교하였다. 연구자는 코드 유사성을 비교하여 취약점을 탐지하는 기법과 패턴 기반으로 취약점을 탐지하는 기법을 이용한 도구 VUDDY, SySeVR, CppCheck, ControlFlag에 대해 실험을 진행하고 그 결과를 비교하였다. VUDDY와 CppCheck는 AI/ML 기법이 적용되지 않은 도구이고, SySeVR과 ControlFlag는 AI/ML 기반 도구이다. 성능 검증 결과는 아래 표 5와 같이 나타났다.

표 5

[0090]

구분	분석도구	위양성율	정확도	정밀도	변수 취약점 탐지율
코드 유사성 비교	VUDDY	0	9.17	100	9.17
	SySeVR	22.9	18.5	77.08	18.5
패턴 기반	Cppcheck	36.84	77.92	63.16	15.58
	ControlFlag	0	9.5	100	9.5
기타	제안 기술	0	77.55	100	77.55

[0092] 연구자는 결함 유형별로 위양성율, 정확도, 정밀도, 변수 취약점 탐지율을 평가하였다. 제안 기술은 위양성율, 정확도, 정밀도 및 변수 취약점 탐지율이 각각 0%, 77.55%, 100% 및 77.55%로 나타났다. 나머지 도구들은 변수 취약점 탐지율이 6%~18.5%로 제안 기술보다 낮게 나타났다.

[0094] 도 8은 소프트웨어의 변수 취약점을 탐지하는 분석장치(400)에 대한 예이다. 분석장치(400)는 도 1의 분석장치(130 또는 150)에 해당하는 장치이다. 분석장치(400)는 전술한 분석 모델을 이용하여 소프트웨어의 변수 결함을 탐지한다. 분석장치(400)는 물리적으로 다양한 형태로 구현될 수 있다. 예컨대, 분석장치(400)는 PC, 스마트 기기, 네트워크상의 서버, 데이터 처리 전용 칩셋 등의 형태를 가질 수 있다.

[0095] 분석장치(400)는 저장장치(410), 메모리(420), 연산장치(430), 인터페이스 장치(440), 통신장치(450) 및 출력장치(460)를 포함할 수 있다.

[0096] 저장장치(410)는 전술한 분석 모델을 저장할 수 있다. 분석 모델은 변수 초기화 위치를 판정하는 제1 학습모델 및 실행 과정에서 변수값(최댓값 및 최솟값)을 예측하는 제2 학습모델을 포함한다. 분석 모델은 학습 과정은 전술한 바와 같다.

[0097] 저장장치(410)는 분석 대상인 타깃 프로그램의 소스 코드를 정적 분석하는 프로그램 내지 코드를 저장할 수 있다.

[0098] 저장장치(410)는 소스 코드에 대한 실행 로그를 생성하는 프로그램 내지 코드를 저장할 수 있다.

[0099] 저장장치(410)는 실행 로그를 일정하게 전처리하는 프로그램 내지 코드를 저장할 수 있다.

[0100] 저장장치(410)는 분석 모델의 출력 정보(초기화 위치, 변수값 정보 등)를 기준으로 변수 결함 내지 결함 후보를 판정하는 프로그램 내지 코드를 저장할 수 있다. 이 과정은 도 7에서 설명한 바와 같다.

[0101] 저장장치(410)는 타깃 프로그램의 소스 코드를 저장할 수 있다.

[0102] 저장장치(410)는 타깃 프로그램의 실행 로그를 저장할 수 있다.

[0103] 메모리(420)는 분석장치(400)가 타깃 프로그램의 변수 취약점을 탐지하는 과정에서 생성되는 데이터 및 정보 등을 저장할 수 있다.

[0104] 인터페이스 장치(440)는 외부로부터 일정한 명령 및 데이터를 입력받는 장치이다. 인터페이스 장치(440)는 물리적으로 연결된 입력 장치 또는 외부 저장장치로부터 타깃 프로그램의 소스 코드를 입력받을 수 있다. 또는, 인터페이스 장치(440)는 타깃 프로그램의 실행 로그를 입력받을 수도 있다.

[0105] 통신장치(450)는 유선 또는 무선 네트워크를 통해 일정한 정보를 수신하고 전송하는 구성을 의미한다. 통신장치(450)는 외부 객체로부터 타깃 프로그램의 소스 코드를 수신할 수 있다. 통신장치(450)는 타깃 프로그램의 실행 로그를 수신할 수 있다.

[0106] 통신장치(450)는 타깃 프로그램에 대한 변수 취약점 탐지 결과를 외부 객체로 송신할 수 있다.

[0107] 나아가, 인터페이스 장치(440)는 통신장치(450)로부터 수신하는 데이터를 전달받는 구성일 수도 있다.

[0108] 출력장치(460)는 일정한 정보를 출력하는 장치이다. 출력장치(460)는 데이터 처리 과정에 필요한 인터페이스,

분석 결과 등을 출력할 수 있다. 출력장치(460)는 결함 탐지 결과를 출력할 수 있다.

[0109] 연산 장치(430)는 타깃 프로그램의 소스 코드에 대한 정적 분석을 수행할 수 있다. 이를 통해 연산 장치(430)는 취약점 탐지 대상인 변수 목록, 함수 목록, 함수의 호출 관계를 추출할 수 있다.

[0110] 연산 장치(430)는 타깃 프로그램을 실행하면서 실행 로그를 추출할 수 있다.

[0111] 연산 장치(430)는 타깃 프로그램이 실행되는 동안 모니터링 대상 변수에 접근하여 호출 함수 위치, 변수명, 접근 전/후 변수값, 콜스택을 추출하여 실행 로그로 수집한다.

[0112] 연산 장치(430)는 수집한 실행 로그를 학습에 활용할 수 있는 데이터로 전처리할 수 있다. 전처리는 실행 로그에서 호출 함수, 변수명 및 콜스택의 함수를 각각 함수 ID 및 변수 ID로 대체 하는 과정이다. 결함 탐지 과정의 전처리는 학습 과정의 전처리와 동일하게 진행된다.

[0113] 연산 장치(430)는 제1 학습모델을 이용하여 전처리된 실행 로그를 분석한다. 연산 장치(430)는 제1 학습모델을 통해 변수 초기화 위치를 판단할 수 있다. 제1 학습모델은 특정 변수의 초기화 여부 내지 초기화 위치를 출력할 수 있다. 제1 학습모델은 전처리된 실행 로그를 입력받아 실행 로그 중 {전역 변수, 호출 함수}의 쌍들 각각이 초기화 함수인지 여부를 분류한다.

[0114] 연산 장치(430)는 제2 학습모델을 이용하여 전처리된 실행 로그를 분석한다. 연산 장치(430)는 제2 학습모델을 통해 실행 과정에서 특정 변수의 최댓값 및 최솟값을 추출할 수 있다.

[0115] 제1 학습모델과 제2 학습모델을 이용하여 추출하는 정보는 실행 로그에서 추출한 변수 정보이다. 실행 로그에서 추출한 변수 정보는 (i) 초기화 정보, (ii) 변수 사용 정보, (iii) 변수값 변화 추세 정보 및 (iv) 변수값 증감 패턴 정보를 포함할 수 있다.

[0116] 연산 장치(430)는 실행 로그에서 추출한 변수 정보를 기준으로 변수 취약점을 판단할 수 있다. 판정 과정은 도 7에서 설명한 바와 같다.

[0117] 연산 장치(430)는 데이터를 처리하고, 일정한 연산을 처리하는 프로세서, AP, 프로그램이 임베디드된 칩과 같은 장치일 수 있다.

[0119] 또한, 상술한 바와 같은 프로그램 변수 취약점 내지 결함 탐지 방법은 컴퓨터에서 실행될 수 있는 실행가능한 알고리즘을 포함하는 프로그램(또는 어플리케이션)으로 구현될 수 있다. 상기 프로그램은 일시적 또는 비일시적 판독 가능 매체(non-transitory computer readable medium)에 저장되어 제공될 수 있다.

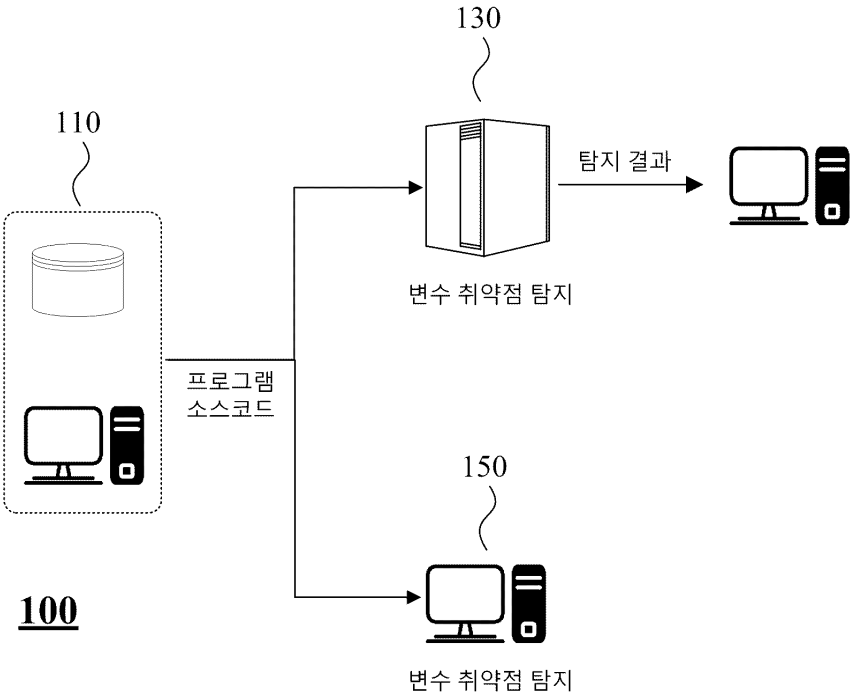
[0120] 비일시적 판독 가능 매체란 레지스터, 캐쉬, 메모리 등과 같이 짧은 순간 동안 데이터를 저장하는 매체가 아니라 반영구적으로 데이터를 저장하며, 기기에 의해 판독(reading)이 가능한 매체를 의미한다. 구체적으로는, 상술한 다양한 어플리케이션 또는 프로그램들은 CD, DVD, 하드 디스크, 블루레이 디스크, USB, 메모리카드, ROM (read-only memory), PROM (programmable read only memory), EPROM(Erasable PROM, EPROM) 또는 EEPROM(Electrically EPROM) 또는 플래시 메모리 등과 같은 비일시적 판독 가능 매체에 저장되어 제공될 수 있다.

[0121] 일시적 판독 가능 매체는 스태틱 램(Static RAM, SRAM), 다이내믹 램(Dynamic RAM, DRAM), 싱크로너스 디램(Synchronous DRAM, SDRAM), 2배속 SDRAM(Double Data Rate SDRAM, DDR SDRAM), 증강형 SDRAM(Enhanced SDRAM, ESDRAM), 동기화 DRAM(Synclink DRAM, SLDRAM) 및 직접 램버스 램(Direct Rambus RAM, DRRAM) 과 같은 다양한 RAM을 의미한다.

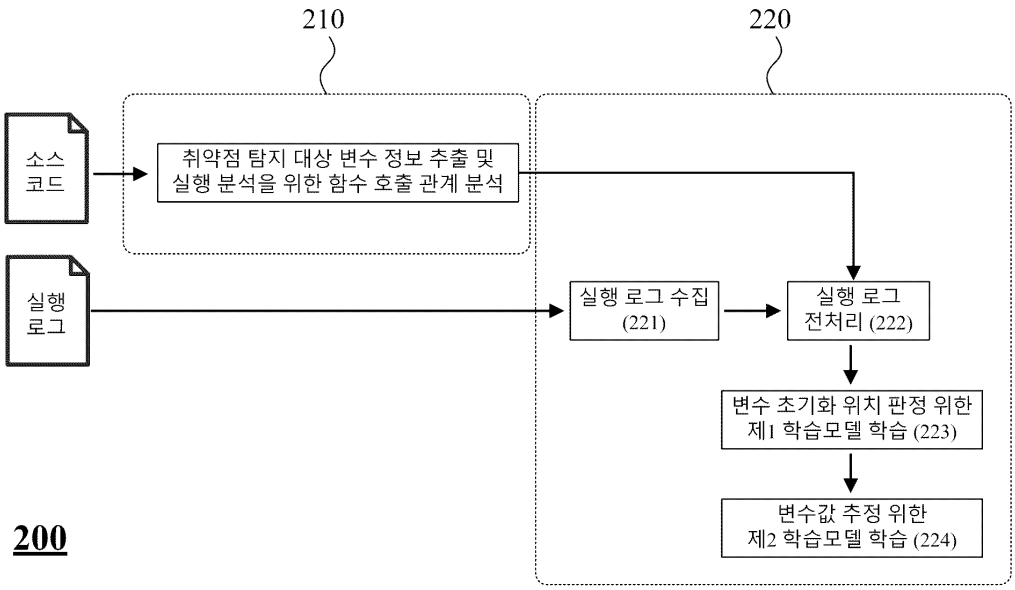
[0122] 본 실시예 및 본 명세서에 첨부된 도면은 전술한 기술에 포함되는 기술적 사상의 일부를 명확하게 나타내고 있는 것에 불과하며, 전술한 기술의 명세서 및 도면에 포함된 기술적 사상의 범위 내에서 당업자가 용이하게 유추할 수 있는 변형 예와 구체적인 실시례는 모두 전술한 기술의 권리범위에 포함되는 것이 자명하다고 할 것이다.

도면

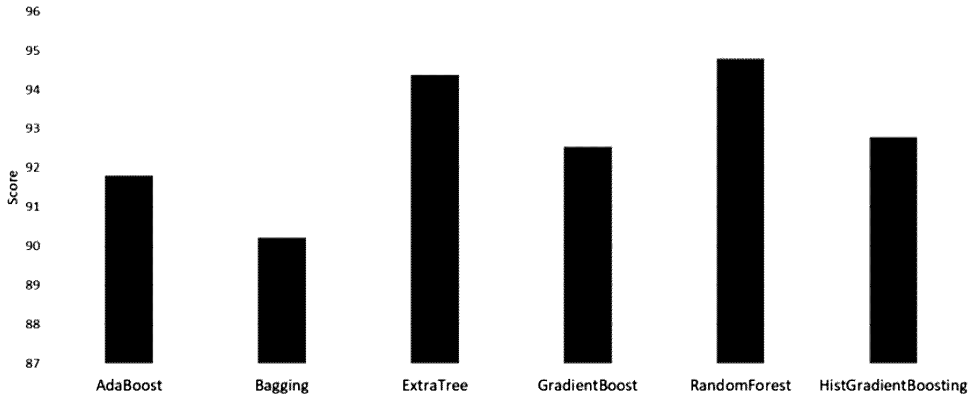
도면1


130
110
탐지 결과
변수 취약점 탐지
프로그램
소스코드
150
100
변수 취약점 탐지

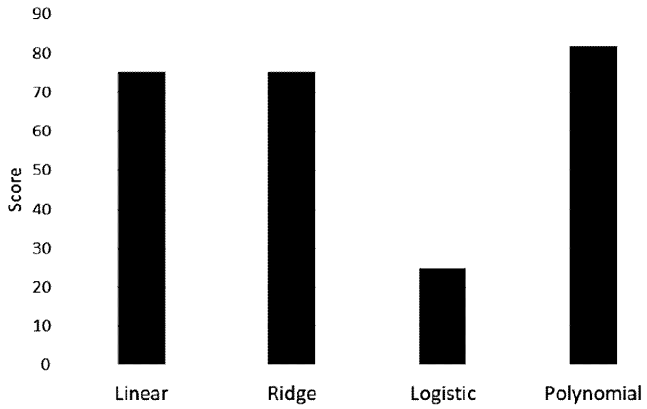
도면2


210
220
소스
코드
취약점 탐지 대상 변수 정보 추출 및
실행 분석을 위한 함수 호출 관계 분석
실행
로그
실행 로그 수집
(221)
실행 로그
전처리 (222)
변수 초기화 위치 판정 위한
제1 학습모델 학습 (223)
변수값 추정 위한
제2 학습모델 학습 (224)
200

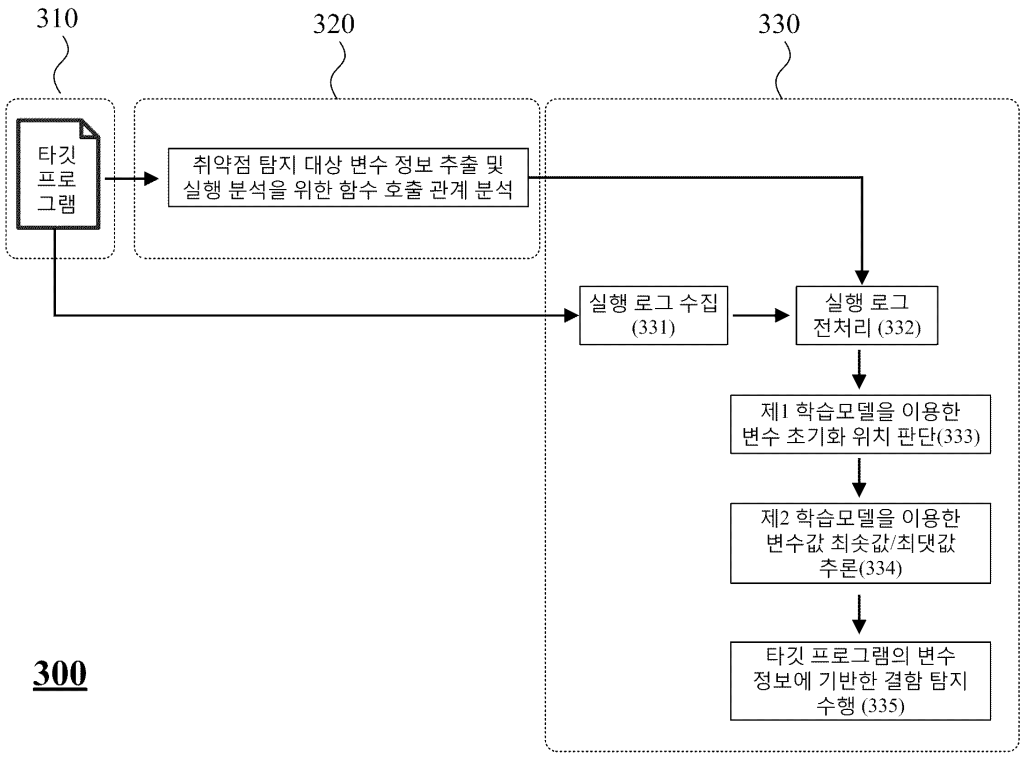
도면3


96
95
94
93
92
91
90
89
88
87
Score
AdaBoost
Bagging
ExtraTree
GradientBoost
RandomForest
HistGradientBoosting

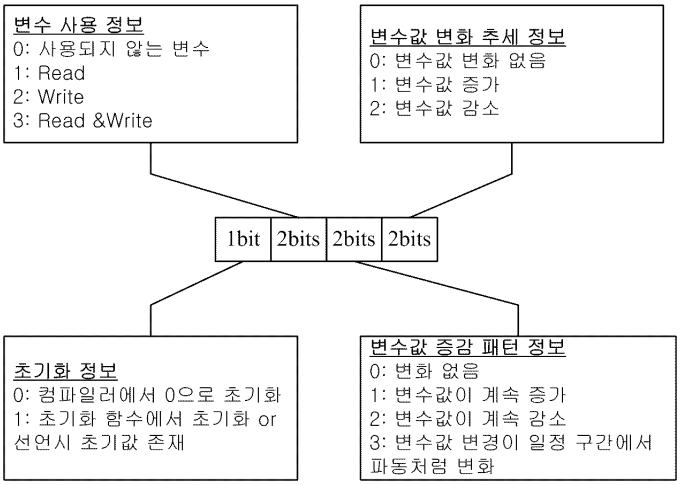
도면4


90
80
70
60
50
40
30
20
10
0
Score
Linear
Ridge
Logistic
Polynomial

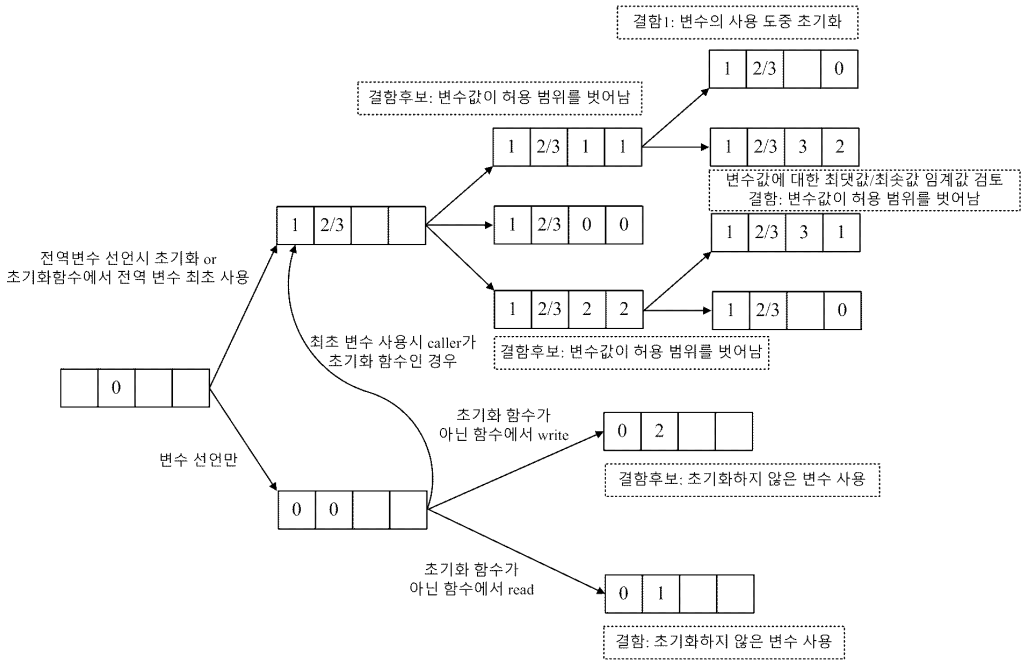
도면5


310
320
330
타깃 프로그램
취약점 탐지 대상 변수 정보 추출 및 실행 분석을 위한 함수 호출 관계 분석
실행 로그 수집 (331)
실행 로그 전처리 (332)
제1 학습모델을 이용한 변수 초기화 위치 판단(333)
제2 학습모델을 이용한 변수값 최솟값/최댓값 추론(334)
타깃 프로그램의 변수 정보에 기반한 결함 탐지 수행 (335)
300

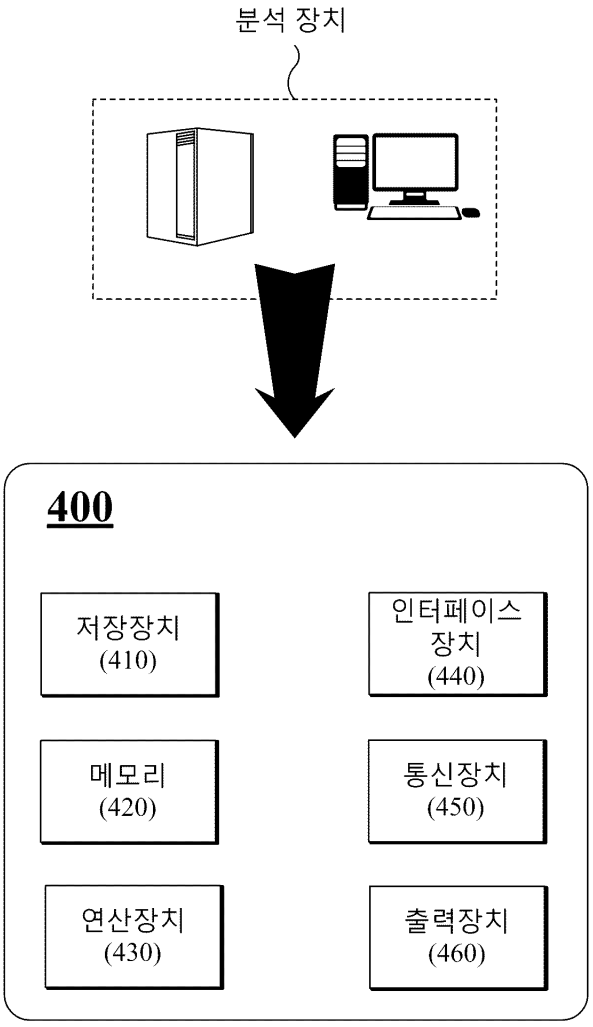
도면6


변수 사용 정보
0: 사용되지 않는 변수
1: Read
2: Write
3: Read &Write
변수값 변화 추세 정보
0: 변수값 변화 없음
1: 변수값 증가
2: 변수값 감소
1bit
2bits
2bits
2bits
초기화 정보
0: 컴파일러에서 0으로 초기화
1: 초기화 함수에서 초기화 or 선언시 초기값 존재
변수값 증감 패턴 정보
0: 변화 없음
1: 변수값이 계속 증가
2: 변수값이 계속 감소
3: 변수값 변경이 일정 구간에서 파동처럼 변화

도면7


결함1: 변수의 사용 도중 초기화
결함후보: 변수값이 허용 범위를 벗어남
변수값에 대한 최댓값/최솟값 임계값 검토
결함: 변수값이 허용 범위를 벗어남
결함후보: 변수값이 허용 범위를 벗어남
전역변수 선언시 초기화 or
초기화함수에서 전역 변수 최초 사용
최초 변수 사용시 caller가
초기화 함수인 경우
초기화 함수가
아닌 함수에서 write
결함후보: 초기화하지 않은 변수 사용
변수 선언만
초기화 함수가
아닌 함수에서 read
결함: 초기화하지 않은 변수 사용

도면8


분석 장치
400
저장장치
(410)
인터페이스
장치
(440)
메모리
(420)
통신장치
(450)
연산장치
(430)
출력장치
(460)