(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2025-0104097
(43) 공개일자 2025년07월08일

(51) 국제특허분류(Int. Cl.)
G06F 21/57 (2013.01) *G06F 12/14* (2006.01)
G06F 3/06 (2006.01)
(52) CPC특허분류
G06F 21/577 (2013.01)
G06F 12/14 (2013.01)
(21) 출원번호 10-2023-0196124
(22) 출원일자 2023년12월29일
심사청구일자 2023년12월29일

(71) 출원인
울산과학기술원
울산광역시 울주군 언양읍 유니스트길 50
(72) 발명자
전유석
울산광역시 울주군 언양읍 유니스트길 50
(74) 대리인
특허법인더웨이브

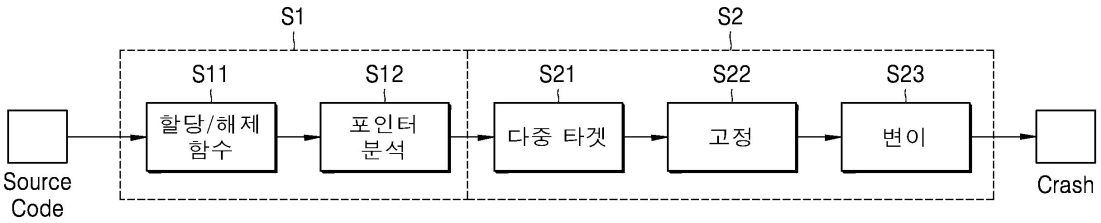
전체 청구항 수 : 총 16 항

(54) 발명의 명칭 **메모리 취약점 탐지 방법 및 장치**

(57) 요 약

적어도 하나의 프로세서와, 상기 프로세서에 의해 실행되는 적어도 하나의 프로그램을 저장하는 메모리와, 상기 메모리의 취약점을 탐지하기 위해 복수의 메모리 객체에 대한 복수의 할당 및 해제 함수들은 선택하는 설정부와, 상기 선택한 함수들에 기초하여 상기 메모리 취약점을 탐지하는 탐지부를 포함하는, 메모리 취약점 탐지 장치 및 탐지 방법에 관한 것이다.

대 표 도



(52) CPC특허분류
G06F 3/062 (2013.01)
G06F 2212/1052 (2013.01)

이 발명을 지원한 국가연구개발사업

과제고유번호	1711193687
과제번호	2022-0-00745-002
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기획평가원
연구사업명	정보보호핵심원천기술개발(R&D)
연구과제명	랜섬웨어 공격 근원지 식별 및 분석 기술 개발
기 여 율	1/1
과제수행기관명	(주)에스투더블유
연구기간	2023.01.01 ~ 2023.12.31

명 세 서

청구범위

청구항 1

적어도 하나의 프로세서;

상기 프로세서에 의해 실행되는 적어도 하나의 프로그램을 저장하는 메모리; 및

상기 메모리의 취약점을 탐지하기 위해 복수의 메모리 객체에 대한 복수의 할당 및 해제 함수들은 선택하는 설정부; 및

상기 선택한 함수들에 기초하여 상기 메모리 취약점을 탐지하는 탐지부를 포함하는, 메모리 취약점 탐지 장치.

청구항 2

제1항에 있어서,

상기 설정부는,

정적 분석을 통해, 상기 할당 및 해제의 함수들을 상기 각 객체 별로 분류하도록 구비된 메모리 취약점 탐지 장치.

청구항 3

제2항에 있어서,

상기 설정부는,

상기 각 객체 별 할당과 해제 함수의 쌍에 대하여 포인터를 생성하고; 및

상기 포인터를 분류하도록 구비된 메모리 취약점 탐지 장치.

청구항 4

제1항에 있어서,

상기 탐지부는,

상기 객체 별로 분류된 할당 및 해제의 조합에 대하여 타겟으로 설정하도록 구비된 메모리 취약점 탐지 장치.

청구항 5

제4항에 있어서,

상기 탐지부는,

제1 타겟을 지나게 하는 제1 입력값 변이를 고정하고; 및

상기 제1 타겟과 다른 제2 타겟을 지나게 하는 제2 입력값 변이를 고정하도록 구비된 메모리 취약점 탐지 장치.

청구항 6

제5항에 있어서,

상기 제1 입력값 변이의 고정 및 제2 입력값 변이의 고정 중 적어도 하나는 바이트 단위의 고정인 메모리 취약점 탐지 장치.

청구항 7

제1항에 있어서,

상기 탐지부는,

복수의 해제 함수가 하나의 할당 함수와 같은 집합에 있는 경우 상기 집합 단위로 커버리지를 계산하도록 구비된 메모리 취약점 탐지 장치.

청구항 8

제1항에 있어서,

상기 프로그램의 실행 전에 정적 분석으로 상기 메모리 취약점에 안전한 제1 할당 및 해제 함수들을 탐지하는 정적 분석부를 더 포함하고,

상기 제1 할당 및 해제 함수들은 상기 탐지부를 실행하지 않도록 하는 메모리 취약점 탐지 장치.

청구항 9

프로세서에 의해 실행되는 적어도 하나의 프로그램을 저장하는 메모리의 취약점을 탐지하는 메모리 취약점 탐지 장치의 메모리 취약점 탐지 방법에 있어서,

메모리 취약점을 탐지하기 위해 복수의 메모리 객체에 대한 복수의 할당 및 해제 함수들을 선택하는 단계;

상기 선택한 코드들에 기초하여 메모리 취약점을 탐지하는 단계를 포함하는, 메모리 취약점 탐지 방법.

청구항 10

제9항에 있어서,

상기 할당 및 해제 함수들을 선택하는 단계는,

정적 분석을 통해, 상기 할당 및 해제의 함수들을 상기 각 객체 별로 분류하는 단계를 포함하는 메모리 취약점 탐지 방법.

청구항 11

제10항에 있어서,

상기 분류하는 단계는, 상기 각 객체 별 할당과 해제 함수에 대하여 포인터를 생성하는 단계; 및

상기 포인터를 분류하는 단계를 포함하는 메모리 취약점 탐지 방법.

청구항 12

제9항에 있어서,

상기 메모리 취약점을 탐지하는 단계는,

상기 객체 별로 분류된 할당 및 해제의 조합에 대하여 타겟으로 설정하는 단계를 포함하는 메모리 취약점 탐지 방법.

청구항 13

제12항에 있어서,

상기 메모리 취약점을 탐지하는 단계는,

제1 타겟을 지나게 하는 제1 입력값 변이를 고정하는 단계; 및

상기 제1 타겟과 다른 제2 타겟을 지나게 하는 제2 입력값 변이를 고정하는 단계;를 포함하는 메모리 취약점 탐지 방법.

청구항 14

제13항에 있어서,

상기 제1 입력값 변이의 고정 및 제2 입력값 변이의 고정 중 적어도 하나는 바이트 단위의 고정인 메모리 취약점 탐지 방법.

청구항 15

제9항에 있어서,

상기 메모리 취약점을 탐지하는 단계는,

복수의 해제 함수가 하나의 할당 함수와 같은 집합에 있는 경우 상기 집합 단위로 커버리지를 계산하는 단계를 포함하는 메모리 취약점 탐지 방법.

청구항 16

제9항에 있어서,

상기 프로그램의 실행 전에 정적 분석으로 상기 메모리 취약점에 안전한 제1 할당 및 해제 함수들을 탐지하는 단계를 더 포함하고,

상기 제1 할당 및 해제 함수들은 상기 메모리 취약점을 탐지하는 단계를 실행하지 않도록 하는 메모리 취약점 탐지 방법.

발명의 설명

기 술 분 야

[0001] 본 발명의 실시예들은 메모리 취약점을 탐지할 수 있는 방법 및 장치에 대한 것으로, 더욱 상세하게는 에너지 효율성이 높게 메모리 취약점을 탐지할 수 있는 취약점 탐지 방법 및 장치에 대한 것이다.

배 경 기 술

[0002] 최근 Google의 OSS-FUZZ, AFL 등 실제 프로그램의 취약점을 탐지하기 위한 소위 퍼징(puzzling) 기술들이 많이 제시되고 사용되고 있다.

[0003] 이러한 취약점 탐지 방법은, Coverage를 기반으로 입력 값에 랜덤한 데이터를 대입함으로서 프로그램의 취약점을 유도하며, 프로그램 내 많은 코드를 탐색해, 개발자가 실수로 놓치는 취약점을 자동으로 탐지해주는 장점을 가지고 있다.

[0004] 한편 UAF(Use After Free)와 같은 메모리 취약점은 메모리를 할당하고 해제했을 때, 해제된 메모리에 접근하면 발생하는 취약점을 가지고 있다. UAF가 발생하기 위해서는, 한 객체에 대한 할당과 해제, 참조가 필요한 데, 기존의 coverage 기반의 취약점 탐지 기술의 경우, UAF가 발생하는 코드에 도달하기까지 coverage를 늘이기 위해 여러 단계의 코드(할당, 해제, 참조)를 지나야 하기 때문에 버그를 유도하기 어려운 문제가 있다.

[0005] 특히 Concurrency UAF와 같이 UAF가 다른 종류의 취약점과 결합하여 동시에 일어나면 이를 발견하기가 더 어려워진다. 또한, 메모리의 할당과 해제 사이의 시간이 길고, NVD(National Vulnerability Database)의 위험도 수치가 7-10을 차지할 정도로 익스플로잇(exploit)했을 경우의 위험도가 높아 2023년 CWE(Common Weakness Enumeration) 위험한 취약점 4위를 차지했다. 이 외에도 Microsoft에서 패치한 취약점 543개 중 99개가 UAF 취약점으로 18%의 비중을 차지하였고, Google이 발견한 취약점에서는 36%의 비중을 차지했다.

[0006] 이러한 UAF의 문제 때문에 UAF만을 탐지하기 위한 다양한 취약점 탐지 도구들도 개발되고 있다.

[0007] 그러나 종래의 취약점 탐지 방법들의 경우, 그 성능을 보장하기 어려운 경우가 많고, 또, 불필요한 코드 테스트로 인한 에너지 소비가 많은 한계가 있다.

발명의 내용

해결하려는 과제

[0009] 본 발명이 이루고자 하는 일 기술적 과제는, 불필요한 코드 테스트로 인한 에너지 소비를 줄이고 버그 발생이 가능한 코드를 중점적으로 테스트할 수 있는 효과적인 메모리 취약점 탐지 장치 및 방법을 제시하는 것이다.

[0010] 본 발명이 이루고자 하는 다른 일 기술적 과제는, 버그가 실현되기 위해 필요한 제약들인, 객체의 할당, 해제, 참조 코드를 효과적으로 테스트할 수 있는 메모리 취약점 탐지 방법 및 장치를 제시하는 것이다.

과제의 해결 수단

[0011] 본 발명이 이루고자 하는 기술적 과제를 달성하기 위하여, 적어도 하나의 프로세서와, 상기 프로세서에 의해 실행되는 적어도 하나의 프로그램을 저장하는 메모리와, 상기 메모리의 취약점을 탐지하기 위해 복수의 메모리 객체에 대한 복수의 할당 및 해제 함수들은 선택하는 설정부와, 상기 선택한 함수들에 기초하여 상기 메모리 취약점을 탐지하는 탐지부를 포함하는, 메모리 취약점 탐지 장치를 제공한다.

[0012] 상기 설정부는, 정적 분석을 통해, 상기 할당 및 해제의 함수들을 상기 각 객체 별로 분류하도록 구비될 수 있다.

[0013] 상기 설정부는, 상기 각 객체 별 할당과 해제 함수의 쌍에 대하여 포인터를 생성하고, 상기 포인터를 분류하도록 구비될 수 있다.

[0014] 상기 탐지부는, 상기 객체 별로 분류된 할당 및 해제의 조합에 대하여 타겟으로 설정하도록 구비될 수 있다.

[0015] 상기 탐지부는, 제1 타겟을 지나게 하는 제1 입력값 변이를 고정하고, 상기 제1 타겟과 다른 제2 타겟을 지나게 하는 제2 입력값 변이를 고정하도록 구비될 수 있다.

[0016] 상기 제1 입력값 변이의 고정 및 제2 입력값 변이의 고정 중 적어도 하나는 바이트 단위의 고정일 수 있다.

[0017] 상기 탐지부는, 복수의 해제 함수가 하나의 할당 함수와 같은 집합에 있는 경우 상기 집합 단위로 커버리지를 계산하도록 구비될 수 있다.

[0018] 상기 프로그램의 실행 전에 정적 분석으로 상기 메모리 취약점에 안전한 제1 할당 및 해제 함수들을 탐지하는 정적 분석부를 더 포함하고, 상기 제1 할당 및 해제 함수들은 상기 탐지부를 실행하지 않도록 할 수 있다.

[0019] 본 발명이 이루고자 하는 기술적 과제를 달성하기 위하여, 프로세서에 의해 실행되는 적어도 하나의 프로그램을 저장하는 메모리의 취약점을 탐지하는 메모리 취약점 탐지 장치의 메모리 취약점 탐지 방법에 있어서, 메모리 취약점을 탐지하기 위해 복수의 메모리 객체에 대한 복수의 할당 및 해제 함수들을 선택하는 단계와, 상기 선택한 코드들에 기초하여 메모리 취약점을 탐지하는 단계를 포함하는, 메모리 취약점 탐지 방법을 제공한다.

[0020] 상기 할당 및 해제 함수들을 선택하는 단계는, 정적 분석을 통해, 상기 할당 및 해제의 함수들을 상기 각 객체 별로 분류하는 단계를 포함할 수 있다.

[0021] 상기 분류하는 단계는, 상기 각 객체 별 할당과 해제 함수에 대하여 포인터를 생성하는 단계와, 상기 포인터를 분류하는 단계를 포함할 수 있다.

[0022] 상기 메모리 취약점을 탐지하는 단계는, 상기 객체 별로 분류된 할당 및 해제의 조합에 대하여 타겟으로 설정하는 단계를 포함할 수 있다.

[0023] 상기 메모리 취약점을 탐지하는 단계는, 제1 타겟을 지나게 하는 제1 입력값 변이를 고정하는 단계와, 상기 제1 타겟과 다른 제2 타겟을 지나게 하는 제2 입력값 변이를 고정하는 단계를 포함할 수 있다.

[0024] 상기 제1 입력값 변이의 고정 및 제2 입력값 변이의 고정 중 적어도 하나는 바이트 단위의 고정일 수 있다.

[0025] 상기 메모리 취약점을 탐지하는 단계는, 복수의 해제 함수가 하나의 할당 함수와 같은 집합에 있는 경우 상기 집합 단위로 커버리지를 계산하는 단계를 포함할 수 있다.

[0026] 상기 프로그램의 실행 전에 정적 분석으로 상기 메모리 취약점에 안전한 제1 할당 및 해제 함수들을 탐지하는 단계를 더 포함하고, 상기 제1 할당 및 해제 함수들은 상기 메모리 취약점을 탐지하는 단계를 실행하지 않도록 할 수 있다.

발명의 효과

[0027] 본 발명의 적어도 하나의 실시예들을 활용하면, UAF와 같은 메모리 취약점을 탐지함에 있어서 불필요한 코드 영역 탐색으로 인한 오버헤드를 줄일 수 있다.

[0028] 본 발명의 적어도 하나의 실시예들을 활용하면, 보다 쉽게 메모리 취약점을 탐지할 수 있고, 에너지 효율성을

더욱 높일 수 있다.

도면의 간단한 설명

[0029] 도 1은 일 실시예에 따른 메모리 취약점 탐지 장치의 구성도이다.

도 2는 다른 일 실시예에 따른 메모리 취약점 탐지 방법을 구현하기 위한 설계도이다.

도 3은 또 다른 일 실시예에 따른 메모리 취약점 탐지 장치의 구성도이다.

도 4는 또 다른 일 실시예에 따른 메모리 취약점 탐지 방법을 구현하기 위한 설계도이다.

발명을 실시하기 위한 구체적인 내용

[0030] 본 발명의 이점 및 특징, 그리고 그것들을 달성하는 방법은 첨부되는 도면과 함께 상세하게 설명되는 실시 예들을 참조하면 명확해질 것이다. 그러나 본 발명은 아래에서 제시되는 실시 예들로 한정되는 것이 아니라, 서로 다른 다양한 형태로 구현될 수 있고, 본 발명의 사상 및 기술 범위에 포함되는 모든 변환, 균등물 내지 대체물을 포함하는 것으로 이해되어야 한다. 아래에 제시되는 실시 예들은 본 발명의 개시가 완전하도록 하며, 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자에게 발명의 범주를 완전하게 알려주기 위해 제공되는 것이다. 본 발명을 설명함에 있어서 관련된 공지 기술에 대한 구체적인 설명이 본 발명의 요지를 흐릴 수 있다고 판단되는 경우 그 상세한 설명을 생략한다.

[0031] 본 출원에서 사용한 용어는 단지 특정한 실시 예를 설명하기 위해 사용된 것으로, 본 발명을 한정하려는 의도가 아니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 출원에서, "포함하다" 또는 "가지다" 등의 용어는 명세서상에 기재된 특징, 숫자, 단계, 동작, 구성요소, 부품 또는 이들을 조합한 것이 존재함을 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성요소, 부품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다. 제1, 제2 등의 용어는 다양한 구성요소들을 설명하는데 사용될 수 있지만, 구성요소들은 상기 용어들에 의해 한정되어서는 안 된다. 상기 용어들은 하나의 구성요소를 다른 구성요소로부터 구별하는 목적으로만 사용된다.

[0032] 또한, 본 출원서에서, "부"는 프로세서 또는 회로와 같은 하드웨어 구성(hardware component), 및/또는 프로세서와 같은 하드웨어 구성에 의해 실행되는 소프트웨어 구성(software component)일 수 있다.

[0033] 이하, 본 발명에 따른 실시 예들을 첨부된 도면을 참조하여 상세히 설명하기로 하며, 첨부 도면을 참조하여 설명함에 있어, 동일하거나 대응하는 구성 요소는 동일한 도면번호를 부여하고 이에 대한 중복되는 설명은 생략하기로 한다.

[0034] 이하의 실시 예에서, 제1, 제2 등의 용어는 한정적인 의미가 아니라 하나의 구성 요소를 다른 구성 요소와 구별하는 목적으로 사용되었다

[0035] 이하의 실시 예에서, 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다.

[0036] 이하의 실시 예에서, 포함하다 또는 가지다 등의 용어는 명세서상에 기재된 특징, 또는 구성요소가 존재함을 의미하는 것이고, 하나 이상의 다른 특징을 또는 구성요소가 부가될 가능성을 미리 배제하는 것은 아니다.

[0037] 어떤 실시 예가 달리 구현 가능한 경우에 특정한 공정 순서는 설명되는 순서와 다르게 수행될 수도 있다. 예를 들어, 연속하여 설명되는 두 공정이 실질적으로 동시에 수행될 수도 있고, 설명되는 순서와 반대의 순서로 진행될 수 있다.

[0039] 이하, 도면을 참고하면서 더욱 더 상세하게 설명한다. 당업자에게 자명한 사실은 상세한 설명을 생략한다.

[0041] 도 1은 일 실시예에 따른 메모리 취약점 탐지 장치의 구성도이다.

[0042] 도 1을 참조하면, 일 실시예에 따른 메모리 취약점 탐지 장치(100)는 설정부(110), 탐지부(120), 프로세서(140) 및 메모리(150)를 포함할 수 있다.

[0043] 상기 설정부(110)는 상기 메모리의 취약점을 탐지하기 위해 복수의 메모리 객체에 대한 복수의 할당 및 해제 함

수들은 선택할 수 있다.

[0044] 이러한 설정부(110)는 정적 분석을 통해, 상기 할당과 해제의 함수들을 각 객체 별로 분류하도록 구비될 수 있다.

[0045] 일 실시예에 따르면, 상기 설정부(110)는, 상기 각 객체 별 할당과 해제 함수의 쌍에 대하여 포인터를 생성하고, 상기 포인터를 분류하도록 구비될 수 있다.

[0046] 상기 탐지부(120)는, 상기 객체 별로 분류된 할당 및 해제의 쌍에 대하여 목표로 설정하도록 구비될 수 있다.

[0047] 일 실시예에 따르면, 상기 탐지부(120)는, 제1 목표를 지나게 하는 제1 입력값 변이를 고정하고, 상기 제1 목표와 다른 제2 목표를 지나게 하는 제2 입력값 변이를 고정하도록 구비될 수 있다. 이 때, 상기 제1 입력값 변이의 고정 및 제2 입력값 변이의 고정 중 적어도 하나는 바이트 단위의 고정일 수 있다.

[0048] 일 실시예에 따르면, 상기 탐지부는, 복수의 해제 함수가 하나의 할당 함수와 같은 집합에 있는 경우 상기 집합 단위로 커버리지를 계산하도록 구비될 수 있다.

[0049] 프로세서(140)는 메모리 취약점 탐지 장치(100)의 동작을 제어할 수 있다. 실시예에서, 프로세서(140)는 예를 들어 프로그램 내에 포함된 코드 또는 명령으로 표현된 기능을 수행하기 위해 물리적으로 구조화된 회로를 갖는, 하드웨어에 내장된 데이터 처리 장치를 의미할 수 있다. 이와 같이 하드웨어에 내장된 데이터 처리 장치의 일 예로써, 마이크로프로세서(microprocessor), 중앙처리장치(central processing unit: CPU), 프로세서 코어(processor core), 멀티프로세서(multiprocessor), ASIC(application-specific integrated circuit), FPGA(field programmable gate array) 등의 처리 장치를 망라할 수 있으나, 본 발명의 범위가 이에 한정되는 것은 아니다.

[0050] 메모리(150)는 프로세서(140)와 동작 가능하게 연결되고, 프로세서(140)에 의해 수행되는 일련의 처리 과정에 대한 적어도 하나 이상의 프로그램이 탑재될 수 있다. 메모리(150)는 프로세서(140)에서 수행되는 적어도 하나의 코드를 저장할 수 있다. 메모리(150)는 프로세서(140)를 통해 메모리 취약점 탐지 방법이 실행될 때 프로세서(140)로 하여금, 메모리 취약점 탐지 방법의 실행에 관련하여, 할당 및/또는 해제의 함수들과 관련된 코드를 저장할 수 있다.

[0051] 실시예에서, 메모리(150)는 자기 저장 매체(magnetic storage media) 또는 플래시 저장 매체(flash storage media)를 포함할 수 있으나, 본 발명의 범위가 이에 한정되는 것은 아니다. 이러한 메모리(150)는 내장 메모리 및/또는 외장 메모리를 포함할 수 있으며, DRAM, SRAM, 또는 SDRAM 등과 같은 휘발성 메모리, OTPROM(one time programmable ROM), PROM, EPROM, EEPROM, mask ROM, flash ROM, NAND 플래시 메모리, 또는 NOR 플래시 메모리 등과 같은 비휘발성 메모리, SSD. CF(compact flash) 카드, SD 카드, Micro-SD 카드, Mini-SD 카드, Xd 카드, 또는 메모리 스틱(memory stick) 등과 같은 플래시 드라이브, 또는 HDD와 같은 저장 장치를 포함할 수 있다.

[0052] 실시예에서, 메모리 취약점 탐지 장치(100)는 예를 들어, 전자 장치(도시하지 않음)에 포함될 수 있으며, 전자 장치에서 발생할 수 있는 악성행위를 탐지하여 출력할 수 있다. 여기서, 전자 장치는 컴퓨팅 장치의 기능을 수행할 수 있는 통신 단말을 포함할 수 있으며, 사용자가 조작하는 데스크 탑 컴퓨터, 스마트폰 노트북 이외에, 태블릿 PC, 스마트 TV, 휴대폰, PDA(personal digital assistant), 미디어 플레이어, 마이크로 서버, GPS(global positioning system) 장치, 전자책 단말, 디지털방송용 단말, 네비게이션, 키오스크, MP3 플레이어, 디지털 카메라, 가전기기 및 기타 모바일 또는 비모바일 컴퓨팅 장치일 수 있으나, 이에 제한되지 않는다.

[0053] 실시예에서 설정부(110) 및 탐지부(120)가 메모리(150)와 분리된 별도의 구성요소로서 나타내었으나, 반드시 이에 한정되는 것은 아니고, 상기 설정부(110) 및/또는 탐지부(120)는 메모리(150)에 저장된 프로그램의 일 구획이 될 수 있다.

[0054] 상기와 같은 메모리 취약점 탐지 장치(100)는 도 2에 도시된 바와 같은 다른 일 실시예에 따른 메모리 취약점 탐지 방법을 구현하기 위한 설계도와 같이 구현될 수 있다.

[0055] 일 실시예에 따른 메모리 취약점 탐지 방법을 구현하기 위한 프로그램 툴은 인스투르멘테이션(instrumentation) 영역(S1)과 퍼징(fuzzing) 영역(S2)으로 나뉘어 구성될 수 있다.

[0056] 인스투르멘테이션 영역(S1)은 전술한 설정부(110)와 같이, 입력된 소스 코드에 대하여 할당과 해제 함수들을 추출하고(S11) 이를 선택할 수 있다. 실시예에 따르면, 상기 할당과 해제 함수들은 예컨대 LLVM을 이용하여 추출

할 수 있다.

[0057] 추출된 함수들에 대하여 포인터 분석(S12)을 통해 각 할당 및 해제의 함수들을 객체에 따라 분류할 수 있다.

[0058] 포인터 분석은 정적 분석 방법에 의해 포인터의 데이터 흐름을 분석하는 것으로, 프로그램 실행 전에 포인터가 어떤 변수 또는 저장 장소를 가리키는지 알려주고, 포인터 사이의 관계를 나타내 준다. 즉, 객체의 할당 시 반환 받는 값과 해제해야 할 대상이 같으면 할당 함수와 해제 함수의 객체가 같다는 것을 알 수 있다. UAF와 같은 메모리 취약점이 발생하기 위해서는 하나의 객체에 대하여 할당, 해제, 참조가 발생해야 하는 데, 포인터 분석을 통해 객체에 따른 포인터를 구분함으로써, 서로 다른 객체에서 발생한 할당과 해제가 올바르게 연결되었는지 확인할 수 있고, 이를 통해 오버헤드를 줄일 수 있다.

[0059] 한편, 함수, 즉 할당 및/또는 해제 함수들이 반환하는 파라미터를 통해서만 분류하게 되면, 할당 함수 하나에 여러 해제 함수들이 생길 수 있다. 예를 들어 객체 A를 할당한 후 해제 함수가 여러 개의 조건문에 포함된 경우, 각 조건문에 포함된 모든 해제 함수들을 확인해야 한다. 이 경우 여러 개의 해제 함수와 하나의 할당 함수를 하나의 집합으로 묶어 진행할 경우, 퍼징 영역(S2)에서 하나의 집합에서 너무 오랜 시간을 투자하는 것을 방지할 수 있다.

[0060] 퍼징 영역(S2)에서는, 메모리 취약점을 탐지하기에 최적화된 퍼징 도구들이 설계되었다.

[0061] 일 실시예에 따르면, 상기 퍼징 영역(S2)은 전술한 탐지부(120)와 같이, 인스투르멘테이션 영역(S1)에서 선택된 함수들에 기초하여 메모리 취약점을 탐지할 수 있다.

[0062] 일 실시예에 따르면, 상기 퍼징 영역(S2)은 예컨대 다중 타겟 방향성 퍼징을 실시할 수 있다.

[0063] 방향성 퍼징은, 메모리를 참조하는 코드를 타겟으로 설정하여, 할당과 해제 함수를 지나고 참조 코드에 도달하는 입력 값들을 모으는 식으로 취약점을 탐지한다. 그러나 기존의 방향성 퍼징 도구들은, 타겟에 도달하기까지의 거리 외의 요소들은 고려하지 않는 한계가 있다. 즉 UAF와 같은 메모리 취약점은 할당, 해제, 참조의 순서로 이루어지는 데, 참조를 타겟으로 하면 메모리 취약점에 관련된 할당과 해제는 퍼징 시 지나칠 수 있다.

[0064] 일 실시예에 따르면, 객체의 할당과 해제를 다중 타겟으로 설정하는(S21) 다중 타겟 방향성 퍼징을 실시할 수 있다. UAF와 같은 메모리 취약점이 발생되기 위해서는 객체의 할당과 해제가 필연적이므로, 이들을 다중 타겟으로 할 경우 메모리 취약점과 관련된 코드를 지나치는 것이 없게 된다. 더욱이 일 실시예에 따르면, 상기 다중 타겟 방향성 퍼징은 객체의 할당과 해제의 조합에 대하여 타겟으로 하기 때문에 더욱 에너지 효율적으로 퍼징이 이루어질 수 있다.

[0065] 일 실시예에 따르면, 상기 퍼징 영역(S2)은 다중 입력값 고정을 실시할 수 있고(S22), 선택적으로 이를 바이트 단위로 고정시킬 수 있다. 이후 입력값 변이를 실시하고(S23), 다시 고정하는 과정을 반복할 수 있다.

[0066] 퍼징은 입력 값에 랜덤한 데이터를 대입하여 버그가 발생하도록 유도하기 때문에 입력 값 변형 전략에 따라 얼마나 많은 버그를 찾을 수 있는지와 버그를 발견하는 시간이 크게 차이가 날 수 있다. 예컨대 Parmesan과 같은 툴의 경우, 기존에 테스트하지 않았던 코드 영역을 발견했을 때, 입력 값을 Byte단위로 살펴 새로운 코드 영역에 도달하기 위해 필요한 Byte를 고정시킨다.

[0067] 전술한 바와 같이 다중 타겟으로 퍼징을 진행할 때에, 타겟을 지나게 하는 입력값 변이를 고정한 후, 다음 타겟으로 고정한 변이 외의 다른 변이들로 방향성 퍼징을 진행한다(S22, S23). 즉, 일 실시예에 따르면, 제1 타겟을 지나게 하는 제1 입력값 변이를 고정한 후, 상기 제1 타겟과 다른 제2 타겟을 지나게 하는 제2 입력값 변이를 고정한다. 선택적으로 상기 제1 입력값 변이의 고정 및/또는 제2 입력값 변이의 고정은 바이트(Byte) 단위로 실시할 수 있다.

[0068] 즉, 본 실시예에 따르면, 전술한 Parmesan과 같이 새로운 코드 영역을 탐지했을 때 바이트를 고정하는 것이 아닌, 타겟에 도달했을 때 바이트를 고정함으로써 취약점과 관련이 없는 코드들을 탐지하는 에너지 소모를 줄일 수 있다.

[0069] 한편, 일 실시예에 따르면, 중간 타겟인 할당과 해제의 조합을 선정할 때, 같은 객체를 할당 및 해제할 수 있는 조합을 선택할 수 있다. 이에 따라 동적 분석 상황에서 객체의 주소를 확인할 수 있어 정확한 타겟을 설정할 수 있다.

[0070] 일 실시예에 따르면, 선택적으로 집합 단위의 커버리지 퍼징을 실시할 수 있다.

[0071] 기존의 순서 기반 커버리지(Coverage) 퍼징은, 할당, 해제, 참조가 순서대로 얼마나 많이 진행되었는지를 기준으로 피드백을 받는다. 이 퍼징은 입력 값들을 변화시켰을 때 기존에 테스트하지 않았던 새로운 영역의 코드를 테스트할 경우 변화시킨 입력 값을 리스트에 추가해준다. 즉, 얼마나 많은 코드를 탐색했는지 커버리지라는 단위로 기존에 확인하지 못했던 코드를 테스트하도록 유도하는 입력 값들을 변화시킬 대상으로 우선 선정하는 것이다. 즉, 변화 대상의 우선선정 기준을 새로운 테스트 결과가 아닌, 할당, 해제, 참조에 대한 새로운 실행 순서를 찾았는지 여부로 결정한다. 따라서 메모리 취약점과 관련 없는 코드에 대한 커버리지를 계산하지 않음으로 버그 유도에 집중할 수 있지만, 메모리 취약점과 관련 있는 새로운 코드를 탐지하기가 어렵기 때문에 메모리 취약점 관련 커버리지가 올라가는 속도가 느리고, 커버리지가 올라가는 속도가 느려짐에 따라 버그를 탐지하기까지 오랜 시간이 걸리는 한계가 있다.

[0072] 이에 따라, 본 발명의 실시예에 따르면, 인스트루멘테이션 영역(S1)에서 만들어 낸 여러 개의 해제 함수가 하나의 할당 함수와 같은 집합에 있는 경우 이러한 집합 단위로 커버리지를 계산할 수 있다. 이에 따라 커버리지 계산 시 속도가 저하되는 한계를 해소할 수 있다.

[0073] 도 3은 또 다른 일 실시예에 따른 메모리 취약점 탐지 장치의 구성도이다.

[0074] 도 3에 도시된 실시예에 따르면, 또 다른 일 실시예에 따른 메모리 취약점 탐지 장치(100)는 설정부(110), 탐지부(120), 정적 분석부(130), 프로세서(140) 및 메모리(150)를 포함할 수 있다. 도 1과 동일한 부분에 대한 설명은 생략한다.

[0075] 상기 정적 분석부(130)는, 프로그램의 실행 전에 정적 분석으로 상기 메모리 취약점에 안전한 제1 할당 및 해제 함수들을 탐지하고, 상기 제1 할당 및 해제 함수들은 상기 탐지부를 실행하지 않도록 할 수 있다. 이에 따라 선택한 코드 영역에 대하여 보다 효율적으로 에너지를 사용할 수 있다.

[0076] 상기와 같은 메모리 취약점 탐지 장치(100)는 도 4에 도시된 바와 같은 또 다른 일 실시예에 따른 메모리 취약점 탐지 방법을 구현하기 위한 설계도와 같이 구현될 수 있다.

[0077] 일 실시예에 따른 메모리 취약점 탐지 방법을 구현하기 위한 프로그램 툴은 정적 분석 영역(S3)과 퍼징 피드백 영역(S4)으로 나뉘어 구성될 수 있다.

[0078] 정적 분석 영역(S3)에서는, 선택된 할당 및 해제 함수들의 조합에 대하여 정적 분석(S31)을 통해 메모리 취약점에 안전한지 여부를 분석한다. 이러한 정적 분석(S31)을 통해 메모리 취약점에 안전한 할당 및 해제 함수들은 퍼징 피드백 영역(S4)으로 진행하지 않는다. 이처럼 정적 분석(S31)만으로 분석이 가능한 할당 및 해제 함수들을 퍼징 피드백 영역(S4)에서 분석하지 않도록 함으로써 불필요한 오버헤드를 줄일 수 있다.

[0079] 정적 분석(S31)을 통해 메모리 취약점에 안전한지 여부가 명확하지 않은 할당 및 해제 함수들에 대해서는 전술한 바와 같이 새로이 다중 타겟 설정(S32)을 진행하고, 퍼징 피드백 영역(S4)에서 퍼징을 실행한다(S41).

[0080] 선택적으로 또 다른 일 실시예에 따른 정적 분석 영역(S3)은 전술한 인스트루멘테이션 영역(S1)을 포함할 수 있다. 따라서 정적 분석(S31)의 시행 전에 도 2에 도시된 바와 같이 할당 및 해제 함수들을 추출하고(S11), 추출된 함수들에 대하여 포인터 분석(S12)을 실시할 수 있다.

[0081] 선택적으로 상기 정적 분석(S31)은, 도 2에 도시된 바와 같이 할당 및 해제 함수들을 추출하고(S11), 추출된 함수들에 대하여 포인터 분석(S12)을 포함할 수 있다.

[0082] 선택적으로 도 2에 도시된 실시예에서도, 상기 인스트루멘테이션 영역(S1)에서 정적 분석을 실시하여 선택된 할당 및 해제 함수들의 조합에 대하여 메모리 취약점에 안전한지 여부를 분석한 후, 메모리 취약점에 안전한 할당 및 해제 함수들에 대해서는 퍼징 영역(S2)으로의 진행이 되지 않도록 할 수 있다.

[0083] 한편, 상기 정적 분석(S31) 및/또는 포인터 분석(S12)으로 모든 할당과 해제 함수에 UAF와 같은 메모리 취약점이 발생하는지 여부를 확인하기 어려울 수 있다. 이에 따라 확인이 힘든 할당과 해제 함수에 대하여는 퍼징 피드백 영역(S4)에서 같은 주소를 가지는 객체의 할당과 해제를 타겟으로 선정할 수 있다.

[0084] 상기 퍼징 피드백 영역(S4)은 퍼징 실행(S41)이 되면 1차 방향성 퍼징(S42)을 진행하고, 입력값에 대한 고정 및 변이(S43)를 달리하여 2차 방향성 퍼징(S44)을 진행할 수 있다. 상기 1차 방향성 퍼징(S42) 및 또는 2차 방향성 퍼징(S44)은 전술한 실시예와 같이 다중 타겟 방향성 퍼징일 수 있다. 상기 입력값에 대한 고정 및 변이(S43) 과정은 전술한 다중 입력값 고정 과정을 포함할 수 있다.

[0085] 도 4에서는 1차 및 2차 방향성 퍼징을 나타내었으나, 본 발명은 이에 한정되는 것은 아니고, 입력값에 대한 고

정과 변이를 진행하면서 복수의 방향성 퍼징을 수행할 수 있다.

[0086] 또 다른 일 실시예에 따르면, 이렇게 퍼징 피드백을 진행한 후 및/또는 퍼징 피드백과 동시에 커버리지 퍼징(S45)을 실시할 수 있는 데, 일 실시예에 따르면 전술한 바와 같이 집합 단위의 커버리지 퍼징을 실시할 수 있다. 예컨대, 변이(mutation)를 위한 시드(seed) 데이터로서 할당과 해제 함수의 집합을 큐(queue)에 넣고, 1차 방향성 퍼징(S42), 고정/변이(S43) 및/또는 2차 방향성 퍼징(S44)을 실행해 커버리지를 측정하고 크래쉬(crash) 여부를 확인한다. 실행한 결과가 커버리지를 증가시키면, 해당 실행을 시드로서 큐에 넣고 다음 라운드의 변이의 우선순위에 놓고, 그렇지 않다면 버린다. 크래쉬가 발생한다면 이를 리포트에 추가한다.

[0087] 선택적으로, 할당과 해제의 다중 타겟을 설정해 방향성 퍼징을 구간별로 실행할 수 있다. 즉, 상기 1차 방향성 퍼징(S42)은 할당을 1차 타겟으로 하여 방향성 퍼징을 수행하고, 2차 방향성 퍼징(S44)에서 해제를 2차 타겟으로 하여 방향성 퍼징을 수행할 수 있다.

[0088] 일 실시예에 따르면, 타겟 변경 시 이전 타겟을 통과하기 위한 입력값 변이를 고정한 후 피드백을 진행할 수 있다. 즉, 1차 타겟인 할당에 도달했을 때 일어난 변이를 저장하고, 2차 타겟인 해제로 방향성 퍼징을 진행할 때 기 저장한 1차 타겟을 지나기 위한 입력값 변이를 고정한 후 퍼징을 진행한다.

[0089] 이처럼 본 발명의 실시예들에 따르면, UAF와 같은 메모리 취약점을 탐지함에 있어서 불필요한 코드 영역 탐색으로 인한 오버헤드를 줄일 수 있고, 보다 쉽게 메모리 취약점을 탐지할 수 있으며, 에너지 효율성을 더욱 높일 수 있다.

[0090] 이상 설명된 본 발명에 따른 실시 예는 컴퓨터 상에서 다양한 구성요소를 통하여 실행될 수 있는 컴퓨터 프로그램의 형태로 구현될 수 있으며, 이와 같은 컴퓨터 프로그램은 컴퓨터로 판독 가능한 매체에 기록될 수 있다. 이때, 매체는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체, CD-ROM 및 DVD와 같은 광기록 매체, 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical medium), 및 ROM, RAM, 플래시 메모리 등과 같은, 프로그램 명령어를 저장하고 실행하도록 특별히 구성된 하드웨어 장치를 포함할 수 있다.

[0091] 한편, 상기 컴퓨터 프로그램은 본 발명을 위하여 특별히 설계되고 구성된 것이거나 컴퓨터 소프트웨어 분야의 당업자에게 공지되어 사용 가능한 것일 수 있다. 컴퓨터 프로그램의 예에는, 컴파일러에 의하여 만들어지는 것과 같은 기계어 코드 뿐만 아니라 인터프리터 등을 사용하여 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드도 포함될 수 있다.

[0092] 본 발명의 명세서(특히 특허청구범위에서)에서 "상기"의 용어 및 이와 유사한 지시 용어의 사용은 단수 및 복수 모두에 해당하는 것일 수 있다. 또한, 본 발명에서 범위(range)를 기재한 경우 상기 범위에 속하는 개별적인 값을 적용한 발명을 포함하는 것으로서(이에 반하는 기재가 없다면), 발명의 상세한 설명에 상기 범위를 구성하는 각 개별적인 값을 기재한 것과 같다.

[0093] 본 발명에 따른 방법을 구성하는 단계들에 대하여 명백하게 순서를 기재하거나 반하는 기재가 없다면, 상기 단계들은 적당한 순서로 행해질 수 있다. 반드시 상기 단계들의 기재 순서에 따라 본 발명이 한정되는 것은 아니다. 본 발명에서 모든 예들 또는 예시적인 용어(예들 들어, 등등)의 사용은 단순히 본 발명을 상세히 설명하기 위한 것으로서 특허청구범위에 의해 한정되지 않는 이상 상기 예들 또는 예시적인 용어로 인해 본 발명의 범위가 한정되는 것은 아니다. 또한, 당업자는 다양한 수정, 조합 및 변경이 부가된 특허청구범위 또는 그 균등물의 범주 내에서 설계 조건 및 팩터에 따라 구성될 수 있음을 알 수 있다.

[0094] 따라서, 본 발명의 사상은 상기 설명된 실시예에 국한되어 정해져서는 아니 되며, 후술하는 특허청구범위 뿐만 아니라 이 특허청구범위와 균등한 또는 이로부터 등가적으로 변경된 모든 범위는 본 발명의 사상의 범주에 속한다고 할 것이다.

부호의 설명

[0095] 100: 메모리 취약점 탐지 장치

110: 설정부

120: 탐지부

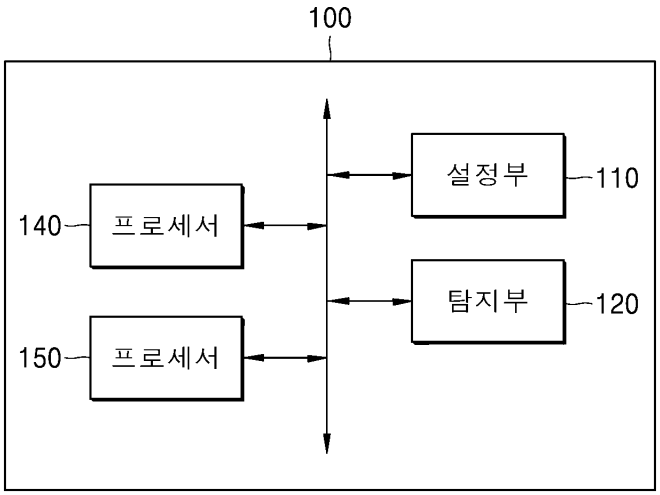
130: 정적 분석부

140: 프로세서

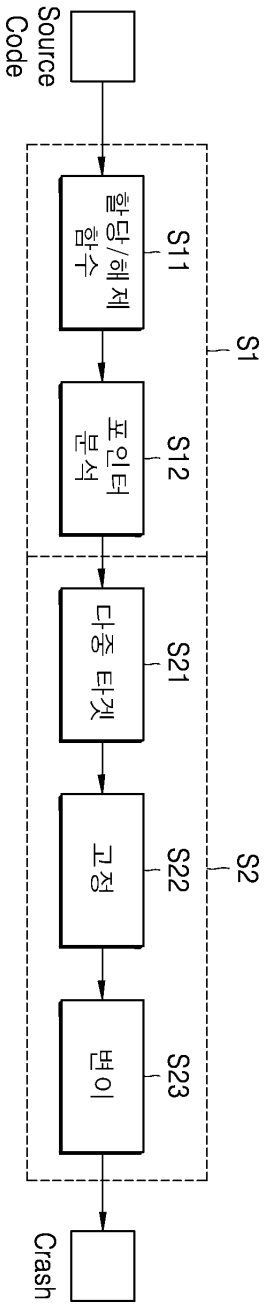
150: 메모리

도면

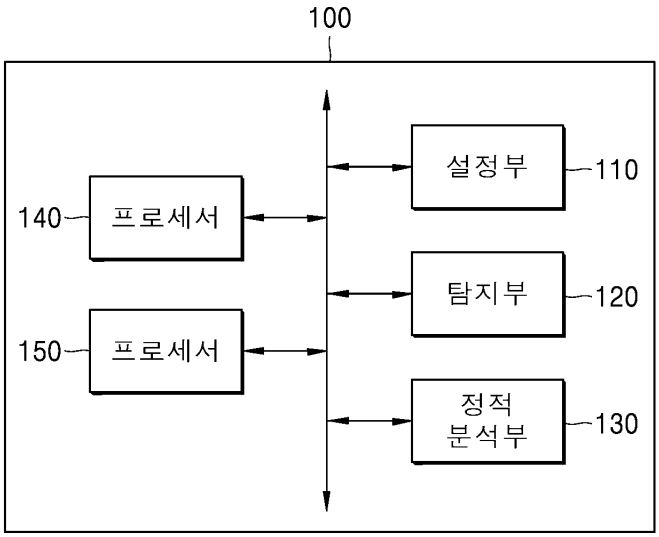
도면1



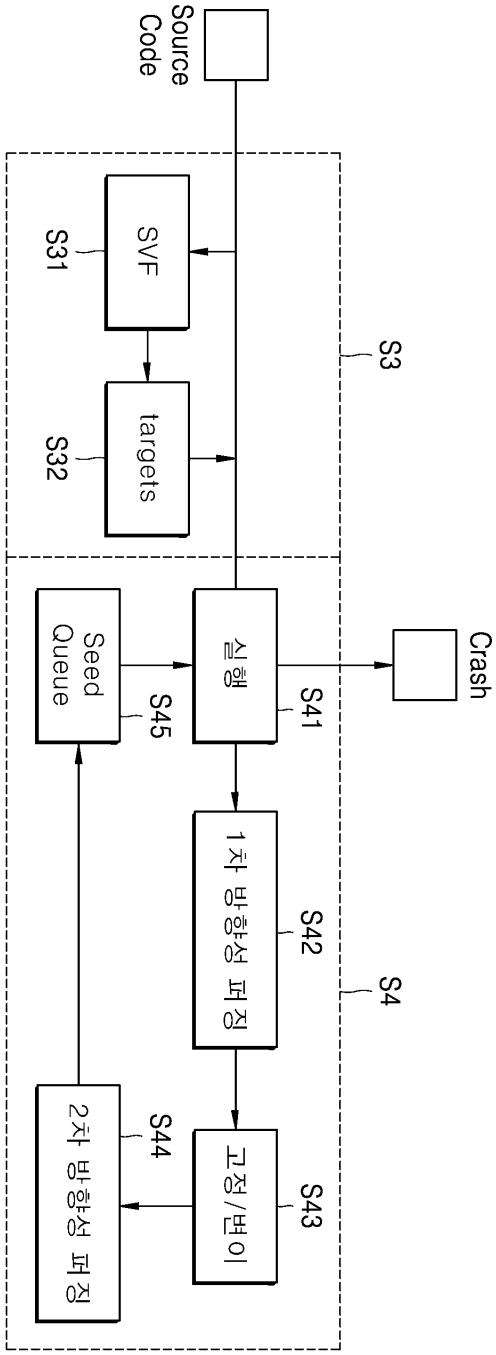
도면2


Source Code
할당/해제 함수
S11
포인터 분석
S12
S1
다중 타겟
S21
고정
S22
변이
S23
S2
Crash

도면3



도면4


Source Code
SVF
S31
targets
S32
S3
Seed Queue
S45
실행
S41
Crash
1차 방향성 퍼징
S42
S4
2차 방향성 퍼징
S44
고정/변이
S43