



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2024-0097586
(43) 공개일자 2024년06월27일

(51) 국제특허분류(Int. Cl.)

G06F 3/06 (2006.01)

(52) CPC특허분류

G06F 3/0659 (2013.01)

G06F 3/0614 (2013.01)

(21) 출원번호 10-2022-0179795

(22) 출원일자 2022년12월20일

심사청구일자 2022년12월20일

(71) 출원인

울산과학기술원

울산광역시 울주군 언양읍 유니스트길 50

(72) 발명자

전유석

울산광역시 울주군 언양읍 유니스트길 50

(74) 대리인

리앤목특허법인

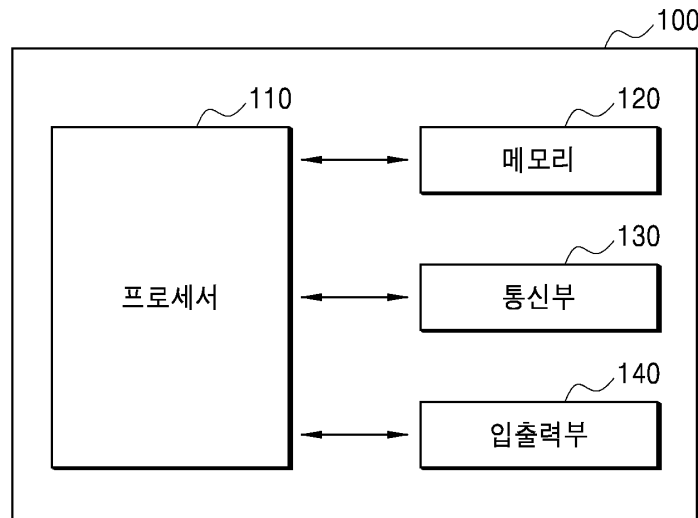
전체 청구항 수 : 총 11 항

(54) 발명의 명칭 메모리 안정성 판단 장치, 비 전형적인 메모리 할당 코드를 탐지하여 메모리 할당 코드의 안정성을 판단하는 방법, 및 컴퓨터 프로그램

(57) 요약

본 개시의 실시예들에 따르면, 메모리 안정성 판단 장치가, 분류기에 프로그램을 입력하여 커스텀 메모리 할당 코드를 탐지하는 단계; 상기 메모리 안정성 판단 장치가, 상기 커스텀 메모리 할당 코드의 실행 시점에 할당되어야 하는 메모리 청크를 정해진 크기 만큼 증가시켜 할당하는 제1 함수 및 증가된 영역을 소정의 가비지 값으로 오염시키는 제2 함수를 삽입하는 단계; 상기 메모리 안정성 판단 장치가, 상기 제1 및 제2 함수가 삽입된 프로그램을 런타임 실행시켜 가비지 값으로 오염된 오염 영역에 대한 접근 여부를 검출하는 단계; 및 상기 메모리 안정성 판단 장치가, 상기 오염 영역에 대한 접근 여부가 참(1, true)인 경우, 상기 커스텀 메모리 할당 코드의 보안성이 취약한 것으로 판단하는 단계;를 포함하는, 메모리 할당 코드의 안정성을 판단하는 방법을 개시한다.

대표도 - 도1



(52) CPC특허분류

G06F 2212/1032 (2013.01)

이 발명을 지원한 국가연구개발사업

과제고유번호	1711152517
과제번호	2021-0-01817-002
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기획평가원
연구사업명	정보통신방송혁신인재양성
연구과제명	하이퍼-컴포저블 데이터센터를 위한 차세대 컴퓨팅 기술 개발
기 여 율	1/1
과제수행기관명	울산과학기술원
연구기간	2022.01.01 ~ 2022.12.31

명세서

청구범위

청구항 1

메모리 안정성 판단 장치가, 분류기에 프로그램을 입력하여 커스텀 메모리 할당 코드를 탐지하는 단계;

상기 메모리 안정성 판단 장치가, 상기 커스텀 메모리 할당 코드의 선행 시점에 할당되어야 하는 메모리 청크를 정해진 크기 만큼 증가시켜 할당하는 제1 함수 및 증가된 영역을 소정의 가비지 값으로 오염시키는 제2 함수를 삽입하는 단계;

상기 메모리 안정성 판단 장치가, 상기 제1 및 제2 함수가 삽입된 프로그램을 런타임 실행시켜 가비지 값으로 오염된 오염 영역에 대한 접근 여부를 검출하는 단계; 및

상기 메모리 안정성 판단 장치가, 상기 오염 영역에 대한 접근 여부가 참(1, true)인 경우, 상기 커스텀 메모리 할당 코드의 보안성이 취약한 것으로 판단하는 단계;를 포함하는, 메모리 할당 코드의 안정성을 판단하는 방법.

청구항 2

제1항에 있어서,

상기 분류기는,

커스텀 메모리 할당 코드들로 학습된 인공 신경망을 이용하여 코드가 커스텀 메모리 할당 코드에 해당하는지 여부를 출력하는, 메모리 할당 코드의 안정성을 판단하는 방법.

청구항 3

제1항에 있어서,

상기 제1 함수가 실행되면, 상기 커스텀 메모리 할당 코드에 의해 할당되어야 하는 제1 크기와, 정해진 제2 크기를 더한 제1 및 제2 크기의 합한 값을 출력하는, 메모리 할당 코드의 안정성을 판단하는 방법.

청구항 4

제3항에 있어서,

상기 제2 함수가 실행되면, 상기 제1 함수의 반환 값 만큼의 메모리 청크를 상기 커스텀 메모리 할당 코드에 의해 할당하며, 상기 제2 크기의 영역을 소정의 가비지 값으로 기록하는, 메모리 할당 코드의 안정성을 판단하는 방법.

청구항 5

제1항에 있어서,

상기 메모리 안정성 판단 장치가, 취약한 보안성을 가지는 할당 코드에 대한 정보를 출력 데이터로 생성하는 단계;를 더 포함하는, 메모리 할당 코드의 안정성을 판단하는 방법.

청구항 6

프로세서 및 비휘발성 메모리를 포함하고,

상기 프로세서가, 상기 메모리에 저장된 명령어들을 실행하고,

분류기에 프로그램을 입력하여 커스텀 메모리 할당 코드를 탐지하고,

상기 커스텀 메모리 할당 코드의 선행 시점에 할당되어야 하는 메모리 청크를 정해진 크기 만큼 증가시켜 할당하는 제1 함수 및 증가된 영역을 소정의 가비지 값으로 오염시키는 제2 함수를 삽입하며,

상기 제1 및 제2 함수가 삽입된 프로그램을 런타임 실행시켜 가비지 값으로 오염된 오염 영역에 대한 접근 여부

를 검출하고,

상기 오염 영역에 대한 접근 여부가 참(1, true)인 경우, 상기 커스텀 메모리 할당 코드의 보안성이 취약한 것으로 판단하는, 메모리 안정성 판단 장치.

청구항 7

제6항에 있어서,

상기 분류기는,

커스텀 메모리 할당 코드들로 학습된 인공 신경망을 이용하여 코드가 커스텀 메모리 할당 코드에 해당하는지 여부를 출력하는, 메모리 안정성 판단 장치.

청구항 8

제6항에 있어서,

상기 제1 함수가 실행되면, 상기 커스텀 메모리 할당 코드에 의해 할당되어야 하는 제1 크기와, 정해진 제2 크기를 더한 제1 및 제2 크기의 합한 값을 출력하는, 메모리 안정성 판단 장치.

청구항 9

제8항에 있어서,

상기 제2 함수가 실행되면, 상기 제1 함수의 반환 값 만큼의 메모리 청크를 상기 커스텀 메모리 할당 코드에 의해 할당하며, 상기 제2 크기의 영역을 소정의 가비지 값으로 기록하는, 메모리 안정성 판단 장치.

청구항 10

제6항에 있어서,

상기 프로세서가,

취약한 보안성을 가지는 할당 코드에 대한 정보를 출력 데이터로 생성하는, 메모리 안정성 판단 장치.

청구항 11

컴퓨터를 이용하여 제1항 내지 제5항 중 어느 한 항의 방법을 실행시키기 위하여 컴퓨터 판독 가능한 저장 매체에 저장된 컴퓨터 프로그램.

발명의 설명

기술 분야

[0001] 본 개시의 실시예들은 메모리 안정성 판단 장치, 비 전형적인 메모리 할당 코드를 탐지하여 메모리 할당 코드의 안정성을 판단하는 방법, 및 컴퓨터 프로그램에 관한 것으로, 보다 구체적으로는 기계학습된 분류기를 통해 획득된 메모리 할당 코드에 해당하는지 여부를 이용하여 프로그램에 포함된 비전형적인 메모리 할당 코드를 탐지하고, 메모리 할당 코드의 안정성을 판단할 수 있다.

배경 기술

[0002] C, C++는 빠른 성능과 높은 자유도로 운영체제와 펌웨어 등 시스템 소프트웨어에 가장 광범위하게 사용되고 있는 언어이다. 그러나, C, C++는 완벽한 보안성을 제공하지 못하고 버퍼오버플로우, Use After Free 등 메모리 취약점이 존재한다. 어드레스 새니타이저(Address Sanitizer)는 비교적 낮은 오버헤드와 높은 탐지율로 이러한 메모리 취약점을 탐지하는 도구로 가장 많이 사용되고 있는 도구 중 하나이다.

[0003] 다양한 응용 프로그램에서는 메모리를 효율적으로 사용하기 위해서 커스텀 메모리 할당 코드(Custom Memory Allocator, CMA)를 사용하는데, Address Sanitizer가 커스텀 메모리 할당 코드로부터 생성된 오브젝트 관련 메모리 안전 위반 취약점을 탐지하고 못하고 있는 실정이다.

[0004] 이러한 커스텀 메모리 할당 코드로부터 파생된 버그를 잡아내기 위한 도구인 H0Tracer가 도입되었다. 이러한, H0Tracer는 커스텀 메모리 할당 코드를 휴리스틱 기반의 행동 패턴을 도출하고 이를 기반으로 커스텀 메모리 할당 코드를 탐지하여 할당자에서 파생된 포인터와 접근 범위를 추적하여 힙 버퍼오버플로우를 찾아내도록 고안되었다. 그러나, 이러한 휴리스틱 기반 탐지 방법을 사용하면 커스텀 메모리 할당 코드의 구현 방식에 따라 탐지하지 못하는 할당자가 존재하는 문제가 있었다.

[0005] 따라서, 커스텀 메모리 할당 코드에 존재하는 메모리 취약점을 탐지하는 도구에 대한 필요성이 증대되고 있다.

발명의 내용

해결하려는 과제

[0006] 본 명세서에서 개시되는 실시예들은, 프로그램의 각각의 코드를 기계학습된 분류기에 입력하여 출력된 메모리 할당 코드에 해당하는지 여부를 이용하여 프로그램에 포함된 비전형적인 메모리 할당 코드를 탐지하고, 메모리 할당 코드의 안정성을 판단하는 장치, 방법, 및 컴퓨터 프로그램을 제시하는데 목적이 있다.

과제의 해결 수단

[0007] 본 개시의 실시예들에 따른 방법은, 메모리 안정성 판단 장치가, 분류기에 프로그램을 입력하여 커스텀 메모리 할당 코드를 탐지하는 단계; 상기 메모리 안정성 판단 장치가, 상기 커스텀 메모리 할당 코드의 선행 시점에 할당되어야 하는 메모리 청크를 정해진 크기 만큼 증가시켜 할당하는 제1 함수 및 증가된 영역을 소정의 가비지 값으로 오염시키는 제2 함수를 삽입하는 단계; 상기 메모리 안정성 판단 장치가, 상기 제1 및 제2 함수가 삽입된 프로그램을 런타임 실행시켜 가비지 값으로 오염된 오염 영역에 대한 접근 여부를 검출하는 단계; 및 상기 메모리 안정성 판단 장치가, 상기 오염 영역에 대한 접근 여부가 참(1, true)인 경우, 상기 커스텀 메모리 할당 코드의 보안성이 취약한 것으로 판단하는 단계;를 포함할 수 있다.

[0008] 상기 분류기는, 커스텀 메모리 할당 코드들로 학습된 인공 신경망을 이용하여 코드가 커스텀 메모리 할당 코드에 해당하는지 여부를 출력할 수 있다.

[0009] 본 개시의 실시예들에 따르면, 상기 제1 함수가 실행되면, 상기 커스텀 메모리 할당 코드에 의해 할당되어야 하는 제1 크기와, 정해진 제2 크기를 더한 제1 및 제2 크기의 합한 값을 출력할 수 있다.

[0010] 본 개시의 실시예들에 따르면, 상기 제2 함수가 실행되면, 상기 제1 함수의 반환 값 만큼의 메모리 청크를 상기 커스텀 메모리 할당 코드에 의해 할당하며, 상기 제2 크기의 영역을 소정의 가비지 값으로 기록할 수 있다.

[0011] 본 개시의 실시예들에 따르면, 상기 메모리 안정성 판단 장치가, 취약한 보안성을 가지는 할당 코드에 대한 정보를 출력 데이터로 생성하는 단계;를 더 포함할 수 있다.

[0012] 본 발명의 실시예에 따른 컴퓨터 프로그램은 컴퓨터를 이용하여 본 발명의 실시예에 따른 방법 중 어느 하나의 방법을 실행시키기 위하여 매체에 저장될 수 있다.

[0013] 이 외에도, 본 발명을 구현하기 위한 다른 방법, 다른 시스템 및 상기 방법을 실행하기 위한 컴퓨터 프로그램을 기록하는 컴퓨터 판독 가능한 기록 매체가 더 제공된다.

[0014] 전술한 것 외의 다른 측면, 특징, 이점이 이하의 도면, 특허청구범위 및 발명의 상세한 설명으로부터 명확해 질 것이다.

발명의 효과

[0015] 전술한 과제 해결 수단 중 어느 하나에 의하면, 프로그램의 각각의 코드를 기계학습된 분류기에 입력하여 출력된 메모리 할당 코드에 해당하는지 여부를 이용하여 프로그램에 포함된 비전형적인 메모리 할당 코드를 탐지하고, 메모리 할당 코드의 안정성을 판단할 수 있다.

도면의 간단한 설명

[0016] 도 1은, 본 개시의 실시예들에 따른 메모리 안정성 판단 장치의 블록도이다.

도 2는 메모리(120)의 블록도이다.

도 3는 메모리 안정성 탐지부(121)의 블록도이다.

도 4는, 본 개시의 실시예들에 따른 메모리 안정성 판단 장치의 예시 도면이다.

도 5는, 본 개시의 실시예들에 따른 메모리 보안성 판단 방법의 흐름도이다.

도 6a는 Pool 패턴의 메모리 할당에 대한 예시 도면이다.

도 6b는 Freelist 패턴의 메모리 할당에 대한 예시 도면이다.

도 6c는 패딩 패턴의 메모리 할당에 대한 예시 도면이다.

도 7은, 본 개시의 실시 예에 의해 발견된 취약점과, 원래의 방법으로 발견된 취약점을 비교한 테이블이다.

발명을 실시하기 위한 구체적인 내용

- [0017] 이하 첨부된 도면들에 도시된 본 발명에 관한 실시예를 참조하여 본 발명의 구성 및 작용을 상세히 설명한다.
- [0018] 본 발명은 다양한 변환을 가할 수 있고 여러 가지 실시예를 가질 수 있는 바, 특정 실시예들을 도면에 예시하고 상세한 설명에 상세하게 설명하고자 한다. 본 발명의 효과 및 특징, 그리고 그것들을 달성하는 방법은 도면과 함께 상세하게 후술되어 있는 실시예들을 참조하면 명확해질 것이다. 그러나 본 발명은 이하에서 개시되는 실시예들에 한정되는 것이 아니라 다양한 형태로 구현될 수 있다.
- [0019] 이하, 첨부된 도면을 참조하여 본 발명의 실시예들을 상세히 설명하기로 하며, 도면을 참조하여 설명할 때 동일하거나 대응하는 구성 요소는 동일한 도면부호를 부여하고 이에 대한 중복되는 설명은 생략하기로 한다.
- [0020] 본 명세서에서 "학습", "러닝" 등의 용어는 인간의 교육 활동과 같은 정신적 작용을 지칭하도록 의도된 것이 아닌 절차에 따른 컴퓨팅(computing)을 통하여 기계 학습(machine learning)을 수행함을 일컫는 용어로 해석한다.
- [0021] 이하의 실시예에서, 제1, 제2 등의 용어는 한정적인 의미가 아니라 하나의 구성 요소를 다른 구성 요소와 구별하는 목적으로 사용되었다.
- [0022] 이하의 실시예에서, 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다.
- [0023] 이하의 실시예에서, 포함하다 또는 가지다 등의 용어는 명세서상에 기재된 특징, 또는 구성요소가 존재함을 의미하는 것이고, 하나 이상의 다른 특징들 또는 구성요소가 부가될 가능성을 미리 배제하는 것은 아니다.
- [0024] 도면에서는 설명의 편의를 위하여 구성 요소들이 그 크기가 과장 또는 축소될 수 있다. 예컨대, 도면에서 나타난 각 구성의 크기 및 두께는 설명의 편의를 위해 임의로 나타내었으므로, 본 발명이 반드시 도시된 바에 한정되지 않는다.
- [0025] 어떤 실시예가 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.
- [0026] 본 명세서에서, 기계 학습은, 훈련 데이터로부터 하나의 함수를 유추해내기 위한 방법으로, 지도 학습, 비지도 학습, 강화 학습 등의 방법 중 하나를 포함할 수 있다.
- [0027] 본 명세서에서, 인공 신경망은, 기계학습과 인지과학에서 생물학의 신경망에서 영감을 얻은 통계학적 학습 알고리즘으로, 스냅스의 결합으로 네트워크를 형성한 인공 뉴런이 학습을 통해 시냅스의 결합 세기를 변화시켜 문제 해결 능력을 가지도록 하는 모델을 말한다.
- [0028] 본 명세서에서, 메모리 오버플로우 보안 취약점은, 연속된 메모리 공간을 사용하는 프로그램에서 할당된 메모리의 범위를 넘어선 위치에 자료를 읽거나 쓰려고 할 때 발생될 수 있다. 메모리 오버플로우 보안 취약점으로 인해 프로그램의 오동작이 발생되거나 악의적인 코드가 실행될 수 있다. 메모리 오버플로우 보안 취약점으로 분류되는 예시로는, 스택 메모리 버퍼 오버플로우와 힙 메모리 버퍼 오버플로우가 있을 수 있다.
- [0029] 이러한 메모리 오버플로우 보안 취약점을 진단하는 도구들은, 버퍼에 값을 기록하는 경우에 기록되는 값의 크기가 버퍼의 크기보다 큰지 여부를 판단함으로써, 메모리 오버플로우 보안 취약점을 탐지할 수 있다. 구현된 프로그램 안에는 다양한 형식의 메모리 할당 코드가 있을 수 있는데, 메모리 오버플로우 보안 취약점을 진단하는 도구들은 전형적인 메모리 할당 코드 이외의 메모리 할당 코드를 검출하지 못하는 문제가 있었다.
- [0030] 비전형적인 메모리 할당 코드는, 커스텀 메모리 할당 코드(Custom memory allocator)를 말하며, Pool 패턴, Freelist 패턴, 좌측 패딩 패턴, 우측 패딩 패턴 등이 있을 수 있다. 여기서, 커스텀 메모리 할당 코드는, 표준

라이브러리에서 제공하는 것이 아닌 malloc 계열사랑, 메모리 할당자를 사용하여 구현된 코드를 말한다.

- [0031] 도 1은, 본 개시의 실시예들에 따른 메모리 안정성 판단 장치의 블록도이다.
- [0032] 본 개시의 실시예들에 따른 메모리 안정성 판단 장치(100)는 기계학습된 분류기를 이용하여 코드가 메모리 할당 코드에 해당하는지 여부를 판단할 수 있다. 프로그램 안에 존재하는 메모리 할당 코드는 분류기를 이용하여 분류될 수 있다. 메모리 안정성 판단 장치(100)는, 분류기로 검출된 메모리 할당 코드의 보안 취약점 여부를 판단할 수 있다.
- [0033] 좌측 패딩 패턴, 우측 패딩 패턴 등의 패딩은 코드에 의해 정해진 크기보다 큰 크기의 메모리 청크를 사용하는 것으로, 패딩은, 표준 할당자로부터 패딩 영역을 할당 받고 패딩을 제외한 영역을 리턴하게(return, 할당하지 않게) 된다. 즉, 좌측 패딩 패턴, 우측 패딩 패턴 등의 코드는 패딩을 제외한 영역을 리턴하여 할당된 영역이 아닌 할당되지 않은 영역을 리턴함으로 인해서, 패딩 영역에 대한 오버플로우가 탐지되지 않는 문제점이 있었다.
- [0034] Pool 패턴은 하나의 메모리 청크(chunk)로 판단되어 Pool 내부에 하나 이상의 메모리가 할당된 경우, Pool 내부의 할당되지 않은 영역에 침범하는 오버플로우를 탐지할 수 없었다. 도 6a를 참조하면, Pool 패턴의 메모리에는, (a), (b), (c), (d), (e) 등과 같이 5개의 데이터가 할당될 수 있다. 할당된 (a), (b), (c), (d), (e)의 사이에는 할당되지 않은 영역들이 존재하는데 할당되지 않은 영역들까지 메모리 오버플로우가 발생하는 것은 데이터에 할당되지 않은 영역의 메모리가 할당되기 때문에 검출되지 않는 문제가 있었다.
- [0035] Freelist 패턴은 처음 메모리 청크가 할당된 뒤 사용자가 해제를 요청하였을 때, 해당 메모리 청크를 리스트에 추가하여 이후 할당에 재활용하는 방식이며, 이를 위해 최초 적합(first-fit), 최적적합(best-fit), 최악 적합(worst-fit) 알고리즘을 사용할 수 있다. 도 6b를 참조하면, Freelist 패턴의 메모리 역시 할당되지 않은 영역을 포함하는 (g)와 같은 경우에는, 데이터가 할당되지 않은 영역을 사용하더라도 오버플로우로 탐지되지 않는 문제가 있었다.
- [0036] 본 개시의 실시예들에 따른 메모리 안정성 판단 장치(100)는 분류기를 이용하여 전형적인 메모리 할당 코드가 아닌 Pool 패턴, Freelist 패턴, 좌측 패딩 패턴, 우측 패딩 패턴 등에 해당하는 메모리 할당 코드를 탐지하고, 탐지된 메모리 할당 코드에 대해서 메모리 오버플로우 취약점을 검출할 수 있다.
- [0037] 본 개시의 실시예들에 따른 메모리 안정성 판단 장치(100)는 코드에서 요청한 크기의 메모리 청크를 할당하고, 메모리 청크를 정해진 레드존 크기만큼 증가하여 할당하고, 레드존 크기의 영역을 가비지 값으로 오염시켜 오염 영역에 대한 접근 여부로 오버플로우 취약점을 탐지할 수 있다.
- [0038] 본 개시의 실시예들에 따른 메모리 안정성 판단 장치(100)는 메모리 청크 크기 인자를 정해진 레드존 크기만큼 증가시키는 런타임 함수를 삽입하고, 상기 런타임 함수 이후에 증가된 레드존 크기의 영역을 오염시키는 런타임 함수를 삽입할 수 있다. 메모리 안정성 판단 장치(100)는 삽입된 오염 영역의 위치 정보를 별도로 기억하고 이러한 위치 정보를 이용하여 메모리 할당에 의해 삽입된 오염 영역에 대한 접근 여부를 검출할 수 있다. 오염 영역에 대해서 접근이 확인되면, 메모리 할당 코드에 의해 할당되는 메모리의 안정성이 취약한 것으로 취약점을 판단할 수 있다.
- [0039] 메모리 안정성 판단 장치(100)는 프로세서(110), 메모리(120), 통신부(130), 입출력부(140)를 포함할 수 있다.
- [0040] 본 개시의 일 실시 예에 따르면, 프로세서(110)는 통상적으로, 메모리 안정성 판단 장치(100)의 전반적인 동작을 제어한다. 예를 들어, 프로세서(110)는 메모리 안정성 판단 장치(100)에 저장된 프로그램을 실행함으로써, 메모리 안정성 판단 장치(100)가 포함하는 구성요소들을 전반적으로 제어할 수 있다.
- [0041] 프로세서(110)는 분류기를 이용하여 전형적인 메모리 할당 코드가 아닌 Pool 패턴, Freelist 패턴, 좌측 패딩 패턴, 우측 패딩 패턴 등에 해당하는 메모리 할당 코드를 탐지하고, 탐지된 메모리 할당 코드에 대해서 메모리 오버플로우 취약점을 검출할 수 있다. 여기서, 분류기는, 프로그램에서 커스텀 메모리 할당 코드를 탐지하는 구성요소로, 소프트웨어 또는 하드웨어 적으로 구현될 수 있다. 분류기는, 프로그램을 입력으로 하고 프로그램 내의 커스텀 메모리 할당 코드에 대한 정보(파일명, 라인수 등)를 출력으로 하는 모듈을 말할 수 있다. 분류기는, 학습된 인공 신경망을 이용할 수 있다. 분류기는, 미리 정해진 규칙에 해당하는지 여부로 커스텀 메모리 할당 코드에 해당하는지 여부를 판단할 수 있다.
- [0042] 프로세서(110)는 코드에서 요청한 크기의 메모리 청크를 할당하고, 메모리 청크를 레드존 크기 만큼 증가하여 요청하고 확장된 메모리 청크 중에서 레드존 크기의 영역을 정해진 가비지 값, 예를 들어, 0xff로 오염시켜 오

버플로우 취약점을 탐지할 수 있다.

- [0043] 프로세서(110)는 메모리 체크 크기 인자를 정해진 레드존 크기만큼 증가시키는 런타임 함수를 삽입하고, 상기 런타임 함수 이후에 증가된 영역을 오염시키는 런타임 함수를 삽입할 수 있다. 프로세서(110)는 삽입된 오염 영역의 위치 정보를 별도로 기억하여 메모리 할당에 의해 삽입된 오염 영역에 대한 접근 여부를 이용하여 메모리 할당 코드의 취약점을 판단할 수 있다.
- [0044] 본 개시의 일 실시 예에 따르면, 메모리(120)는 프로세서(110)의 처리 및 제어를 위한 프로그램을 저장할 수 있고, 메모리 안정성 판단 장치(100)로 입력되거나 메모리 안정성 판단 장치(100)로부터 출력되는 데이터를 저장할 수도 있다. 본 개시의 일 실시 예에 따르면, 메모리(120)는 메모리 안전성 판단에 필요한 구성요소를 저장할 수도 있고, 메모리 안전성 판단에 대한 정보를 저장할 수도 있다. 메모리(120)는 메모리 안전성 판단이 필요한 프로그램을 저장할 수 있고, 프로그램을 정적 분석 및/또는 동적 분석하는 구성, 프로그램의 실행을 모니터링하는 구성 등을 저장할 수 있다. 메모리(120)는 상기의 정보들을 저장한 데이터베이스를 포함할 수 있다.
- [0045] 본 개시의 일 실시 예에 따르면, 메모리(120)는 플래시 메모리 타입(flash memory type), 하드디스크 타입(hard disk type), 멀티미디어 카드 마이크로 타입(multimedia card micro type), 카드 타입의 메모리(예를 들어 SD 또는 XD 메모리 등), 램(RAM, Random Access Memory) SRAM(Static Random Access Memory), 롬(ROM, Read-Only Memory), EEPROM(Electrically Erasable Programmable Read-Only Memory), PROM(Programmable Read-Only Memory), 자기 메모리, 자기 디스크, 광디스크 중 적어도 하나의 타입의 저장매체를 포함할 수 있다. 또한 본 개시의 일 실시 예에 따르면, 메모리(120)에 저장된 프로그램들은 그 기능에 따라 복수 개의 모듈들로 분류할 수 있다.
- [0046] 본 개시의 일 실시 예에 따르면, 통신부(130)는 프로세서(110)의 외부 장치와 통신을 수행할 수 있다. 예를 들어, 통신부(130)는 프로세서(110)의 제어에 의해 보안이 필요한 프로그램이 저장된 전자 장치와 통신을 수행할 수 있다. 또한, 통신부(130)는 외부 인터페이스와의 통신을 통해 보안이 필요한 프로그램이 저장된 전자 장치에 접근하여 프로그램을 분석하고 프로그램에 대한 보안 정책을 설정하도록 통신할 수 있다.
- [0047] 입출력부(140)는 입출력 장치와의 인터페이스를 위한 수단일 수 있다. 예를 들어, 입력 장치는 키보드 또는 마우스 등의 장치를, 그리고 출력 장치는 어플리케이션의 통신 세션을 표시하기 위한 디스플레이와 같은 장치를 포함할 수 있다. 다른 예로 입출력부(140)는 터치스크린과 같이 입력과 출력을 위한 기능이 하나로 통합된 장치와의 인터페이스를 위한 수단일 수도 있다. 보다 구체적인 예로, 메모리 안정성 판단 장치(100)의 프로세서(110)는 메모리(120)에 로딩되거나, 저장된 컴퓨터 프로그램의 명령을 처리함에 있어서 서비스 화면이나 콘텐츠가 입출력부(140)를 통해 디스플레이에 표시될 수 있다.
- [0048] 또한, 다른 실시 예들에서 메모리 안정성 판단 장치(100)는 도 1의 구성요소들보다 더 많은 구성요소들을 포함할 수도 있다. 예를 들어, 상술한 입출력 장치 중 적어도 일부를 포함하도록 구현되거나 또는 내부 구성요소들에 전력을 공급하는 배터리 및 충전 장치, 각종 센서, 데이터베이스 등과 같은 다른 구성요소들을 더 포함할 수도 있다.
- [0049] 도 2는 메모리(120)의 블록도이다.
- [0050] 메모리(120)는, 메모리 안정성 탐지부(121)와 메모리 안정성을 탐지하고자 하는 대상 프로그램(122)를 포함할 수 있다.
- [0051] 메모리 안정성 탐지부(121)는 사용자의 지시 또는 프로그램의 지시에 따라 대상 프로그램(122)를 판독하여 대상 프로그램(122)에서 커스텀 메모리 할당 코드를 검출하고 커스텀 메모리 할당 코드가 메모리 오버플로우 취약점을 가지는지 여부를 탐지할 수 있다. 메모리 안정성 탐지부(121)는 탐지된 결과 데이터를 생성하여 제공할 수 있다.
- [0052] 메모리 안정성 탐지부(121)는, 분류기를 이용하여 전형적인 메모리 할당 코드가 아닌 Pool 패턴, Freelist 패턴, 좌측 패딩 패턴, 우측 패딩 패턴 등에 해당하는 메모리 할당 코드를 탐지하고, 탐지된 메모리 할당 코드에 대해서 메모리 오버플로우 취약점을 검출할 수 있다. 여기서, 분류기는, 커스텀 메모리 할당 코드들로 학습된 모델로, 학습된 인공 신경망을 이용하여 코드가 커스텀 메모리 할당 코드에 해당하는지 여부를 출력할 수 있다.
- [0053] 분류기에 입력되는 학습 데이터는, 일부 불필요한 코드가 제거된 IR 코드일 수 있다. 분류기는 IR 코드를 기반으로 각 명령을 임베딩하여 학습된 RNN 신경망을 이용할 수 있다. IR 코드는, 저레벨 중간 표현의 코드를 말한

다. 프로그래밍 언어로 구현된 프로그램은 컴파일러를 통해서 고레벨 중간 표현의 코드를 거쳐, 저레벨 중간 표현의 코드로 변환될 수 있다.

- [0054] 메모리 안정성 탐지부(121)는, 코드에서 요청한 크기의 메모리 청크를 할당하고, 메모리 청크를 레드존 크기만큼 증가하고 증가된 메모리 청크의 영역을 미리 정해진 가비지 값으로 오염시키고, 메모리 할당 코드의 사용 이후에 상기 메모리 청크의 영역의 값들을 확인하여 메모리 오버플로우 취약점을 탐지할 수 있다.
- [0055] 보다 구체적으로, 메모리 안정성 탐지부(121)는, 메모리 할당 코드에서 할당하는 메모리 영역 이외에 메모리 청크 크기 인자를 정해진 영역인 레드존만큼 증가시키는 런타임 함수를 삽입할 수 있다. 런타임 함수는, 커스텀 메모리 할당 코드에 의해 할당되어야 하는 제1 크기와, 정해진 제2 크기를 더한 제1 및 제2 크기의 합한 값을 출력할 수 있다.
- [0056] 또한, 메모리 안정성 탐지부(121)는 상기 런타임 함수 이후에 증가된 영역인 레드존을 오염시키는(poison) 런타임 함수를 삽입할 수 있다. 여기서, 증가된 영역인 레드존은, 제2 크기의 영역을 말한다. 런타임 함수에 의해 오염된 레드존은 오염 영역으로 명명할 수 있다. 메모리 안정성 탐지부(121)는 해제 함수의 호출 코드 앞에는 추가된 오염 영역인 레드존을 해제하는 런타임 함수를 삽입할 수 있다. 메모리 안정성 탐지부(121)는, 삽입된 오염 영역의 위치 정보를 별도로 기억하여 메모리 할당에 의해 삽입된 오염 영역에 대한 접근 여부를 이용하여 메모리 할당 코드의 취약점을 판단할 수 있다.
- [0057] 도 3는 메모리 안정성 탐지부(121)의 블록도이다.
- [0058] 메모리 안정성 탐지부(121)는, 분석부(1211), 코드 추출부(1212), 탐지부(1213), 출력 데이터 생성부(1214)를 포함할 수 있다.
- [0059] 분석부(1211)는 메모리 오버플로우 취약점을 탐지하고자 하는 대상 프로그램을 런타임으로 실행시킬 수 있다.
- [0060] 코드 추출부(1212)는 대상 프로그램의 명령어들을 관독하여 커스텀 메모리 할당 코드를 검출할 수 있다. 코드 추출부(1212)는, 커스텀 메모리 할당 코드를 할당 코드와 해제 코드로 분류할 수 있다. 코드 추출부(1212)는, 분류기를 이용하여 커스텀 메모리 할당 코드를 할당 코드, 해제 코드, 전형적 메모리 할당 코드 중 하나로 분류할 수 있다.
- [0061] 탐지부(1213)는 코드 추출부(1212)를 통해 검출된 커스텀 메모리 할당 코드에 취약점 탐지 함수를 추가할 수 있다. 탐지부(1213)는 추가된 함수들을 이용하여 커스텀 메모리 할당 코드의 취약점을 탐지할 수 있다.
- [0062] 탐지부(1213)는, 커스텀 메모리 할당 코드 중에서, 할당 코드의 취약점을 탐지하기 위해서, 상기의 할당 코드에서 할당하는 메모리 영역 이외에 메모리 청크 크기 인자를 정해진 크기의 레드존만큼 증가시키는 제1 런타임 함수를 삽입할 수 있다. 제1 런타임 함수는 커스텀 메모리 할당 코드에 의해 할당되어야 하는 제1 크기와, 정해진 제2 크기를 더한 제1 및 제2 크기의 합한 값을 출력할 수 있다.
- [0063] 또한, 탐지부(1213)는 상기 런타임 함수 이후에 증가된 영역인 레드존을 오염시키는 제2 런타임 함수를 삽입할 수 있다. 제2 런타임 함수에 의해 메모리 청크의 우측에 오염 영역이 삽입될 수 있다. 제2 런타임 함수는, 제1 함수의 반환 값 만큼의 메모리 청크를 상기 커스텀 메모리 할당 코드에 의해 할당하며, 상기 제2 크기의 영역을 소정의 가비지 값으로 기록할 수 있다. 탐지부(1213)는 해제 함수의 호출되는 코드 앞에는 레드존만큼 증가되어 할당된 영역을 해제하는 제3 런타임 함수를 삽입할 수 있다.
- [0064] 도 6a를 참조하면, Pool 패턴의 메모리 할당은, M1과 같이 하나의 메모리 Pool에 (a), (b), (c), (d), (e)를 할당하는 것을 말할 수 있다.
- [0065] Pool 패턴의 메모리 할당 코드로 (a), (b), (c), (d), (e)의 영역이 할당되는 경우, 탐지부(1213)는, 제1, 2 런타임 함수를 통해서 (a), (b), (c), (d), (e) 영역에 R11, R12, R13, R14, R15의 오염 영역을 (a), (b), (c), (d), (e) 영역과 인접하여 삽입할 수 있다(M12). 이러한 과정을 통해서 (b), (c), (d), (e)의 우측에 오염 영역이 삽입될 수 있다. R11, R12, R13, R14, R15의 오염 영역은 (a), (b), (c), (d), (e) 사이에 배치되도록 구현될 수 있다. 탐지부(1213)는, 오염 영역(R11, R12, R13, R14, R15)에 대한 접근 여부를 통해서 메모리 할당 코드의 우측 오버플로우의 탐지 여부를 검출할 수 있다.
- [0066] 메모리 청크의 우측 영역은, 메모리 청크의 주소값들 보다 큰 주소값을 가지는 영역일 수 있다. 우측 방향이 주소값이 증가하는 경우, 메모리 청크의 우측 영역은 메모리 청크보다 더 큰 주소값을 가지는 영역을 말한다. 오버플로우가 나게 되면, 할당된 영역을 초과하는 영역에 까지 데이터가 저장되는 것이므로, 우측 영역은, 더 큰

주소값을 가지는 영역을 말한다.

- [0067] 도 6b를 참조하면, Freelist 패턴은 메모리 영역에 비어 있는 영역에 (f), (g), (h)를 독립적으로 할당하는 것을 말한다(M21).
- [0068] Freelist 패턴의 메모리 할당 코드로, 메모리 청크 (f), (g), (h)가 존재하는 경우, 메모리 청크 (f)는 최적 적합으로 할당된 메모리 청크이며, (g)는 최초 적합 또는 최악 적합으로 할당된 메모리 청크이고, (h)는 해제된 상태의 메모리 청크이다. 탐지부(1213)는 최적 적합으로 할당된 메모리 청크(g)의 우측 영역에 오염 영역(R2)를 삽입할 수 있고(M22), 오염 영역(R2)에 대한 접근 여부를 통해서 메모리 할당 코드의 취약점을 탐지할 수 있다.
- [0069] 여기서, 최초 적합은, 메모리를 뒤져서 가장 처음에 할당될 수 있는 영역에 데이터를 할당하는 것을 말한다. 최적 적합은, 할당되어야 하는 크기가 가장 비슷한 영역에 할당하는 것을 말한다. 최악 적합은, 가장 큰 비어있는 영역에 할당하는 것을 말한다.
- [0070] 도 6c를 참조하면, 패딩 패턴의 메모리 할당 코드로 메모리 청크(i)의 경우, 탐지부(1213)는, 메모리 청크(i)의 우측에 오염 영역(R3)을 삽입할 수 있다.
- [0071] 탐지부(1213)는, 오염 영역에 대한 접근 여부를 통해서 커스텀 메모리 할당 코드의 취약점을 검출할 수 있다. 제1 할당 코드의 우측에 제1 오염 영역을 삽입하고, 프로그램의 실행 중에 제1 오염 영역의 접근 신호가 검출되는 경우, 제1 할당 코드에 보안 취약점이 존재하는 것으로 판단할 수 있다.
- [0072] 출력 데이터 생성부(1214)는 보안 취약점을 가지는 할당 코드에 대한 정보를 출력 데이터로 생성할 수 있다. 출력 데이터 생성부(1214)는, 할당 코드의 보안성이 취약한 것으로 판단하면, 취약한 보안성을 가지는 할당 코드에 대한 정보를 출력 데이터로 생성할 수 있다. 생성된 출력 데이터는, 출력 장치를 통해서 디스플레이될 수 있다.
- [0073] 도 4는, 본 개시의 실시예들에 따른 메모리 안정성 판단 장치의 예시 도면이다.
- [0074] 본 개시의 실시예들에 따른 메모리 안정성 판단 장치(100)의 M1는, 프로그램을 분류기에 입력하여 커스텀 메모리 할당 코드를 탐지할 수 있다. 메모리 안정성 판단 장치(100)는, Arrow, leveldb, nginx, scipy 등 8개의 프로그램에 포함된 Freelist, Pool 할당자를 대상으로 데이터 셋을 구축하여 분류기를 학습시킬 수 있다. 메모리 안정성 판단 장치(100)는, 코드를 기반으로 각 명령을 임베딩(embedding)하여 RNN 신경망을 학습시킬 수 있다.
- [0075] 메모리 안정성 판단 장치(100)의 M2는, 런타임 실행시킨 경우, 할당 코드(Alloc)와 해제 코드(Free)의 전 후방에 런타임 함수를 추가할 수 있다. 구체적으로, M2는 할당 코드의 선행 시점에 할당 코드의 크기 인자를 증가시키는 함수(ExtendSize)를 삽입할 수 있다. ExtendSize를 통해서 소정의 레드존이 추가될 수 있다. M2는 할당 코드의 후행 시점에 메모리 청크의 우측을 오염시키는 함수(Poison Right)를 삽입할 수 있다. Poison Right는 레드존을 오염시켜 오염 영역으로 만들 수 있다. M2는 추가된 레드존인 오염 영역의 위치 정보를 메타 데이터로 저장하고, 해제 코드(Free)의 포인터 인자로부터 메모리 청크 크기 만큼 떨어진 오염 영역을 비오염 상태로 변경하는 함수(UnPoisonRight)를 구현하여, 해제 함수의 선행 시점에 삽입할 수 있다.
- [0076] 상기의 과정을 통해 검출된 메모리 안정성에 대한 정보는 다른 테스트 프로그램(fuzzer, Test, Real World)에 전달될 수 있다(M3).
- [0077] 도 5는, 본 개시의 실시예들에 따른 메모리 보안성 판단 방법의 흐름도이다.
- [0078] S110에서는, 메모리 안정성 판단 장치(100)는, 분류기에 프로그램을 입력하여 커스텀 메모리 할당 코드를 탐지할 수 있다. 탐지된 커스텀 메모리 할당 코드에 대한 데이터가 생성될 수 있다.
- [0079] S120에서는, 메모리 안정성 판단 장치(100)는, 커스텀 메모리 할당 코드의 선행 시점에 할당되어야 하는 메모리 청크를 정해진 크기 만큼 증가시켜 할당하는 함수와, 증가된 영역을 소정의 가비지 값으로 오염시키는 함수를 삽입할 수 있다. 할당하는 함수에 의해, 증가된 크기만큼의 메모리를 할당할 수 있다. 예를 들어, a크기의 메모리 청크라고 한다면, 정해진 크기b를 증가시킨 a+b 크기의 메모리 청크를 할당할 수 있다. 오염시키는 함수에 의해, 증가된 영역(b)에 대해서 정해진 가비지 값을 기록할 수 있다. 증가된 영역은, 원래의 메모리 청크보다 높은 어드레스로 할당되기 때문에 메모리 청크의 우측에 할당될 수 있다.
- [0080] S130에서는, 메모리 안정성 판단 장치(100)는, 상기와 같이 함수들이 삽입된 프로그램을 런타임 실행시켜, 가비지 값으로 오염된 오염 영역에 대한 접근 여부를 검출할 수 있다.

- [0081] S140에서는, 메모리 안정성 판단 장치(100)는, 오염된 오염 영역에 대한 접근 여부가 참(1, TRUE)인 경우, 커스텀 메모리 할당 코드의 보안성은 취약한 것으로 판단될 수 있다.
- [0082] 도 7은, 본 개시의 실시 예에 의해 발견된 취약점과, 원래의 방법으로 발견된 취약점을 비교한 테이블이다.
- [0083] 테이블에 도시된 바와 같이, 종래 기술에 따르면, Pool 패턴, 패딩 패턴, Freelist 패턴으로 구현된 커스텀 메모리 할당 코드에 대해서 발생된 오버플로우가 탐지되지 않음을 알 수 있다.
- [0084] 그러나, 제안 발명에 따르면, Pool 패턴의 좌측 오버플로우와 우측 오버플로우를 탐지할 수 있다. 또한, Padding 패턴의 우측 오버플로우를 탐지할 수 있다. 또한, Freelist 패턴의 우측 오버플로우를 탐지할 수 있다.
- [0085] 이상에서 설명된 장치는 하드웨어 구성요소, 소프트웨어 구성요소, 및/또는 하드웨어 구성요소 및 소프트웨어 구성요소의 조합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성요소는, 예를 들어, 프로세서, 컨트롤러, ALU(arithmetic logic unit), 디지털 신호 프로세서(digital signal processor), 마이크로컴퓨터, FPGA(field programmable gate array), PLU(programmable logic unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 하나 이상의 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(OS) 및 상기 운영 체제 상에서 수행되는 하나 이상의 소프트웨어 어플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(processing element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 컨트롤러를 포함할 수 있다. 또한, 병렬 프로세서(parallel processor)와 같은, 다른 처리 구성(processing configuration)도 가능하다.
- [0086] 소프트웨어는 컴퓨터 프로그램(computer program), 코드(code), 명령(instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성요소(component), 물리적 장치, 가상 장치(virtual equipment), 컴퓨터 저장 매체 또는 장치, 또는 전송되는 신호 파(signal wave)에 영구적으로, 또는 일시적으로 구체화(embody)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.
- [0087] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(magnetic media), CD-ROM, DVD와 같은 광기록 매체(optical media), 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다. 상기된 하드웨어 장치는 실시예의 동작을 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.
- [0088] 이상과 같이 실시예들이 비록 한정된 실시예와 도면에 의해 설명되었으나, 해당 기술분야에서 통상의 지식을 가진 자라면 상기의 기재로부터 다양한 수정 및 변형이 가능하다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다.
- [0089] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

부호의 설명

[0090]

100: 메모리 안정성 판단 장치

110: 프로세서

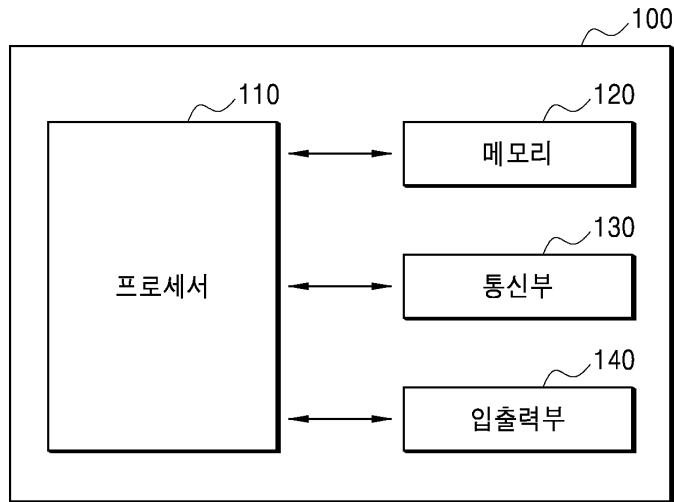
120: 메모리

130: 통신부

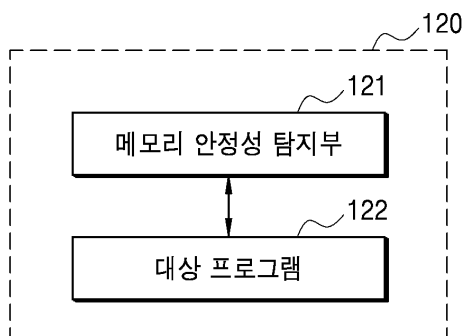
140: 입출력부

도면

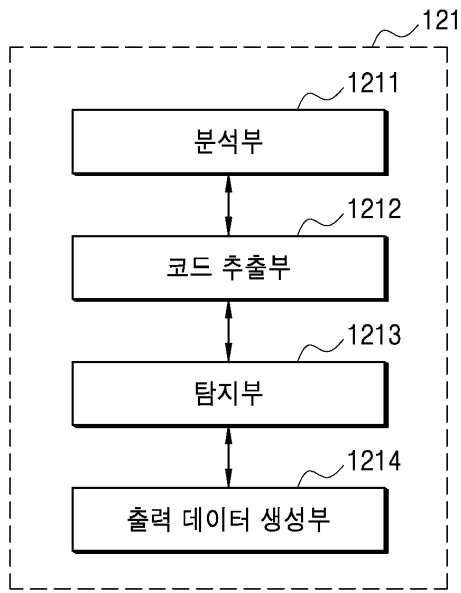
도면1



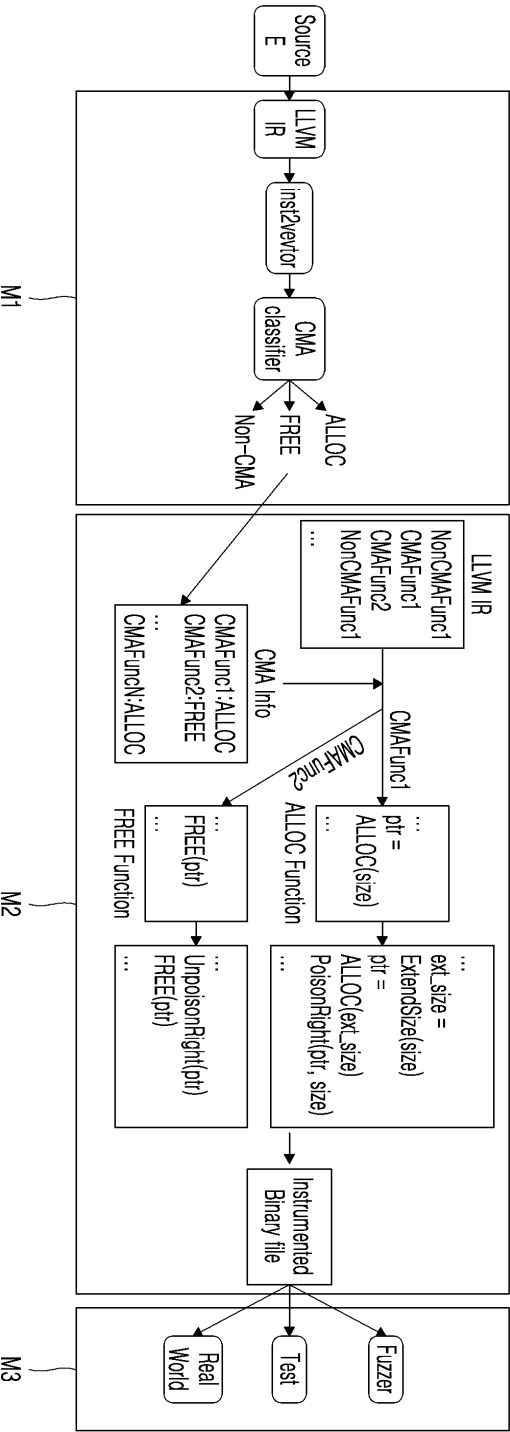
도면2



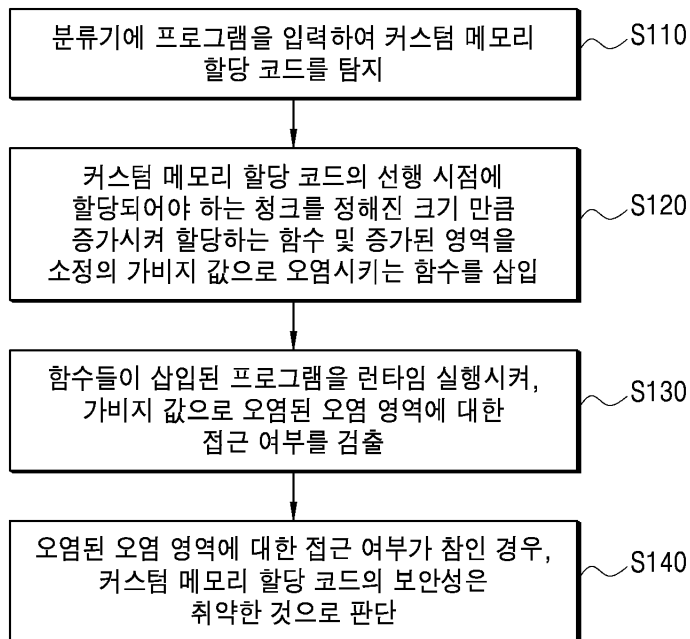
도면3



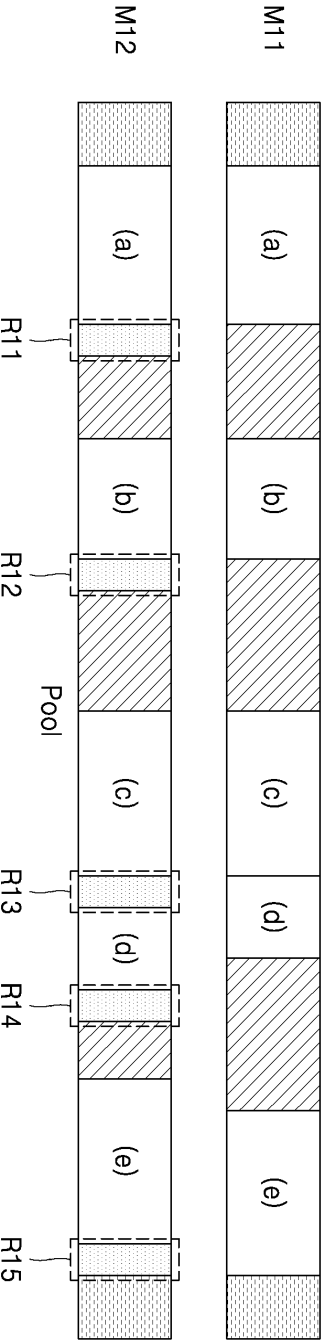
도면4



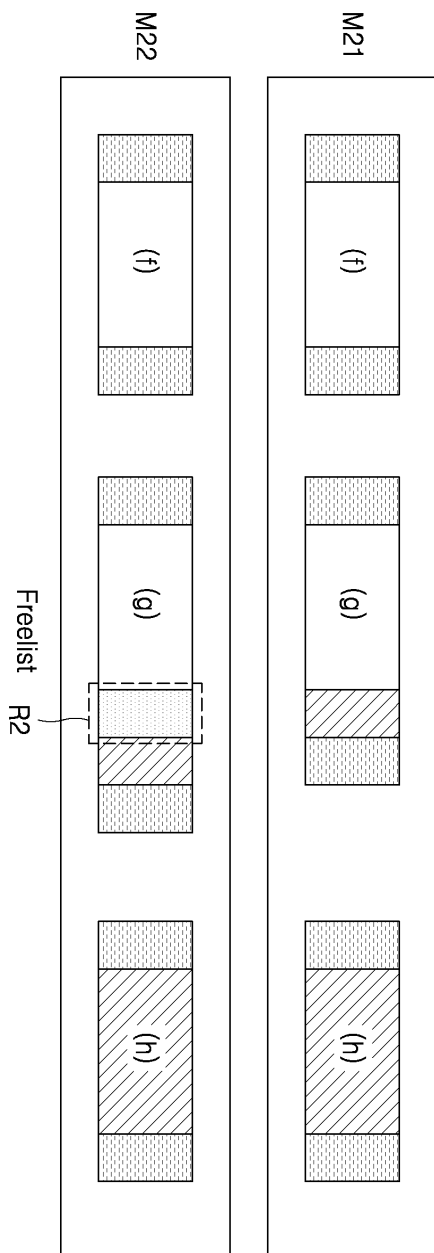
도면5



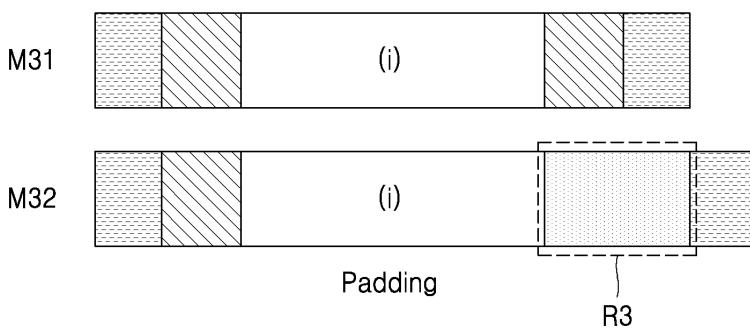
도면6a



도면6b



도면6c



도면7

할당자	패턴	청크 포인터	청크 타입	취약점	종래기술	제안발명
ngx_pool	Pool	p3	(d)	좌측 오버플로우	X	O
ngx_pool	Pool	p3	(d)	우측 오버플로우	X	O
ngx_pool	Pool	p4	(b)	좌측 오버플로우	X	X
Aligned	Padding	p1	(i)	좌측 오버플로우	X	X
Aligned	Padding	p1	(i)	우측 오버플로우	X	O
LowLevel	Freelist	p8	(g)	우측 오버플로우	X	O