(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2025-0085479
(43) 공개일자 2025년06월12일

(51) 국제특허분류(Int. Cl.)
G06F 11/36 (2025.01)
(52) CPC특허분류
G06F 11/3636 (2013.01)
G06F 11/3608 (2023.01)
(21) 출원번호 10-2023-0174829
(22) 출원일자 2023년12월05일
심사청구일자 2023년12월05일

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

전체 청구항 수 : 총 9 항

(54) 발명의 명칭 **함수 포인터 감시 장치, 함수 포인터 감시 방법, 및 컴퓨터 프로그램**

(57) 요 약

본 개시의 실시예들에 따르면, 프로그램 코드 내의 함수 포인터 타입의 데이터를 포인터 호출 코드로 검출하고, 포인터 호출 코드를 감시하는 코드를 추가하여 프로그램 실행 중에 포인터 호출 코드에 대한 어뮤징을 감시할 수 있다.

대 표 도 - 도1



(52) CPC특허분류
G06F 11/3648 (2013.01)
G06F 11/3688 (2013.01)
G06F 11/3692 (2013.01)
G06F 11/3696 (2013.01)

이 발명을 지원한 국가연구개발사업

과제고유번호	1711193910
과제번호	2021-0-01817-003
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	2023.01.01 ~ 2023.12.31

명 세 서

청구범위

청구항 1

함수 포인터 감시 장치가, 프로그램 코드에서 함수 포인터 타입의 변수를 포인터 호출 코드로 검출하는 단계;

상기 함수 포인터 감시 장치가, 상기 프로그램 코드를 정적 분석하여 상기 포인터 호출 코드에 대한 유효값 리스트를 생성하는 단계;

상기 함수 포인터 감시 장치가, 상기 포인터 호출 코드에 대한 유효값 리스트를 이용하여 감시 코드를 생성하고, 상기 감시 코드를 프로그램 코드에 삽입하는 단계;

상기 함수 포인터 감시 장치가, 상기 감시 코드가 추가된 프로그램 코드를 실행 파일로 변환하는 단계; 및

상기 함수 포인터 감시 장치가, 상기 실행 파일이 실행되는 과정에, 상기 감시 코드에 의한 포인터 호출 코드를 모니터링하는 단계를 포함하는, 함수 포인터 감시 방법.

청구항 2

제1항에 있어서,

상기 포인터 호출 코드로 검출하는 단계는,

함수 포인터의 변수가 사용되는 함수의 개수, 또는 함수 포인터가 가리키는 메모리 영역을 고려하여, 함수 포인터의 변수를 포인터 호출 코드로 검출하는,

청구항 3

제1항에 있어서,

상기 유효값 리스트를 생성하는 단계는,

상기 프로그램 코드를 정적 분석하여 프로그램 할당 데이터를 생성하여, 상기 프로그램 할당 데이터를 이용하여 포인터 호출 코드에 할당되는 메모리 영역을 검색하고, 상기 메모리 영역에 할당되는 함수에 대한 정보를 획득하여, 상기 함수에 대한 정보를 상기 포인터 호출 코드에 대한 유효값 리스트를 생성하는,

청구항 4

제1항에 있어서,

상기 유효값 리스트를 생성하는 단계는,

상기 프로그램 코드를 정적 분석하여 포인터 호출 코드에 할당되는 함수에 대한 값의 흐름에 대한 데이터를 획득하고, 함수에 대한 값의 흐름에 대한 데이터를 이용하여 상기 포인터 호출 코드에 대한 유효값 리스트를 생성하는,

청구항 5

통신부, 메모리 및 프로세서를 포함하고,

상기 프로세서가,

프로그램 코드에서 함수 포인터 타입의 변수를 포인터 호출 코드로 검출하며,

상기 프로그램 코드를 정적 분석하여 상기 포인터 호출 코드에 대한 유효값 리스트를 생성하고,

상기 포인터 호출 코드에 대한 유효값 리스트를 이용하여 감시 코드를 생성하고, 상기 감시 코드를 프로그램 코드에 삽입하며,

상기 감시 코드가 추가된 프로그램 코드를 실행 파일로 변환하고,

상기 실행 파일이 실행되는 과정에, 상기 감시 코드에 의한 포인터 호출 코드를 모니터링하는 명령어들을 실행시키는, 함수 포인터 감시 장치.

청구항 6

제5항에 있어서,

상기 프로세서가,

함수 포인터의 변수가 사용되는 함수의 개수, 또는 함수 포인터가 가리키는 메모리 영역을 고려하여, 함수 포인터의 변수를 포인터 호출 코드로 검출하는, 함수 포인터 감시 장치.

청구항 7

제5항에 있어서,

상기 프로세서가,

상기 프로그램 코드를 정적 분석하여 프로그램 할당 데이터를 생성하여, 상기 프로그램 할당 데이터를 이용하여 포인터 호출 코드에 할당되는 메모리 영역을 검색하고, 상기 메모리 영역에 할당되는 함수에 대한 정보를 획득하여, 상기 함수에 대한 정보를 상기 포인터 호출 코드에 대한 유효값 리스트를 생성하는, 함수 포인터 감시 장치.

청구항 8

제5항에 있어서,

상기 프로세서가,

상기 프로그램 코드를 정적 분석하여 포인터 호출 코드에 할당되는 함수에 대한 값의 흐름에 대한 데이터를 획득하고, 함수에 대한 값의 흐름에 대한 데이터를 이용하여 상기 포인터 호출 코드에 대한 유효값 리스트를 생성하는, 함수 포인터 감시 장치.

청구항 9

컴퓨터를 이용하여 제1항의 방법을 실행시키기 위하여 컴퓨터 판독 가능한 저장 매체에 저장된 컴퓨터 프로그램.

발명의 설명

기 술 분 야

[0001] 본 개시의 실시예들은 함수 포인터 감시 장치, 함수 포인터 감시 방법, 및 컴퓨터 프로그램에 관한 것으로, 보다 구체적으로는 프로그램 코드 내의 함수 포인터 타입의 데이터를 포인터 호출 코드로 검출하고, 포인터 호출 코드를 감시하는 코드를 추가하여 프로그램 실행 중에 포인터 호출 코드에 대한 어뮤징을 감시하는 점을 특징으로 한다.

배 경 기 술

[0002] 프로그래밍 코드에서 함수를 호출하는 방법에는, 정의된 이름으로 함수를 호출하는 방법, 표준 함수 호출, 메서드인 경우에는, 점 표기법으로 호출하는 방법, 함수 포인터로 호출하는 방법, 콜백(callback) 함수의 형태로 호출하는 방법, 이벤트 기반 프로그래밍에서 함수를 이벤트와 연관하여 정의하여 이벤트가 발생되는 경우 함수를 호출하는 방법(event-handler) 등이 있다.

[0003] 이중에서, 함수 포인터로 호출하는 방법은 함수를 정의한 이름으로 호출하지 않고 함수를 포인터 타입의 변수로 할당하여 변수를 통해 함수를 호출하는 방법을 말한다.

[0004] 함수 포인터로 함수를 호출하는 경우, 함수 포인터는 할당되는 데이터의 타입에 구애 받지 않고 함수가 호출될 수 있게 된다.

[0005] 이런 경우, 함수의 입력 변수와 출력 변수의 데이터 타입과 다른 변수로 함수가 호출되어 프로그램이 정상적으로 실행되지 않는 문제가 있을 수 있다.

[0006] 따라서, 함수 포인터 타입의 변수를 통해 함수가 호출되는 코드에 대해서 감시를 할 필요성이 대두되고 있다.

발명의 내용

해결하려는 과제

[0007] 본 명세서에서 개시되는 실시예들은, 프로그램 코드 내의 함수 포인터 타입의 데이터를 포인터 호출 코드로 검출하고, 포인터 호출 코드를 감시하는 코드를 추가하여 프로그램 실행 중에 포인터 호출 코드에 대한 어뮤징을 감시하는 장치, 방법을 제공하는데 목적이 있다.

과제의 해결 수단

[0008] 본 개시의 실시예들에 따른, 방법은, 함수 포인터 감시 장치가, 프로그램 코드에서 함수 포인터 타입의 변수를 포인터 호출 코드로 검출하는 단계; 상기 함수 포인터 감시 장치가, 상기 프로그램 코드를 정적 분석하여 상기 포인터 호출 코드에 대한 유효값 리스트를 생성하는 단계; 상기 함수 포인터 감시 장치가, 상기 포인터 호출 코드에 대한 유효값 리스트를 이용하여 감시 코드를 생성하고, 상기 감시 코드를 프로그램 코드에 삽입하는 단계; 상기 함수 포인터 감시 장치가, 상기 감시 코드가 추가된 프로그램 코드를 실행 파일로 변환하는 단계; 및 상기 함수 포인터 감시 장치가, 상기 실행 파일이 실행되는 과정에, 상기 감시 코드에 의한 포인터 호출 코드를 모니터링하는 단계를 포함할 수 있다.

[0009] 상기 포인터 호출 코드로 검출하는 단계는, 함수 포인터의 변수가 사용되는 함수의 개수, 또는 함수 포인터가 가리키는 메모리 영역을 고려하여, 함수 포인터의 변수를 포인터 호출 코드로 검출할 수 있다.

[0010] 상기 유효값 리스트를 생성하는 단계는, 상기 프로그램 코드를 정적 분석하여 프로그램 할당 데이터를 생성하여, 상기 프로그램 할당 데이터를 이용하여 포인터 호출 코드에 할당되는 메모리 영역을 검색하고, 상기 메모리 영역에 할당되는 함수에 대한 정보를 획득하여, 상기 함수에 대한 정보를 상기 포인터 호출 코드에 대한 유효값 리스트를 생성할 수 있다.

[0011] 상기 유효값 리스트를 생성하는 단계는, 상기 프로그램 코드를 정적 분석하여 포인터 호출 코드에 할당되는 함수에 대한 값의 흐름에 대한 데이터를 획득하고, 함수에 대한 값의 흐름에 대한 데이터를 이용하여 상기 포인터 호출 코드에 대한 유효값 리스트를 생성할 수 있다.

[0012] 본 개시의 실시예들에 따른 장치가, 통신부, 메모리 및 프로세서를 포함하고, 상기 프로세서가, 프로그램 코드에서 함수 포인터 타입의 변수를 포인터 호출 코드로 검출하며, 상기 프로그램 코드를 정적 분석하여 상기 포인터 호출 코드에 대한 유효값 리스트를 생성하고, 상기 포인터 호출 코드에 대한 유효값 리스트를 이용하여 감시 코드를 생성하고, 상기 감시 코드를 프로그램 코드에 삽입하며, 상기 감시 코드가 추가된 프로그램 코드를 실행 파일로 변환하고, 상기 실행 파일이 실행되는 과정에, 상기 감시 코드에 의한 포인터 호출 코드를 모니터링하는 명령어들을 실행시킬 수 있다.

[0013] 본 발명의 실시예에 따른 컴퓨터 프로그램은 컴퓨터를 이용하여 본 발명의 실시예에 따른 방법 중 어느 하나의 방법을 실행시키기 위하여 매체에 저장될 수 있다.

[0014] 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.

[0015] 전술한 것 외의 다른 측면, 특징, 이점이 이하의 도면, 특허청구범위 및 발명의 상세한 설명으로부터 명확해 질 것이다.

발명의 효과

[0016] 전술한 과제 해결 수단 중 어느 하나에 의하면, 프로그램 코드 내의 함수 포인터 타입의 데이터를 포인터 호출 코드로 검출하고, 포인터 호출 코드를 감시하는 코드를 추가하여 프로그램 실행 중에 포인터 호출 코드에 대한 어뮤징을 감시할 수 있다.

도면의 간단한 설명

[0017] 도 1은, 본 개시의 실시예들에 따른 네트워크 환경(1)을 설명하는 도면이다.

도 2는, 본 개시의 실시예들에 따른 함수 포인터 감시 장치(100)의 블록도이다.

도 3는 메모리(120)의 블록도이다.

도 4는, 프로그램 감시부(121)의 블록도이다.

도 5는 본 개시의 실시예들에 따른 포인터 호출 코드를 감시하는 방법의 흐름도이다.

도 6는 포인터 호출 코드에 대한 예시 도면이다.

도 7은 포인터 호출 코드가 사용되는 형식에 대한 예시 도면이다.

발명을 실시하기 위한 구체적인 내용

[0018] 이하 첨부된 도면들에 도시된 본 발명에 관한 실시 예를 참조하여 본 발명의 구성 및 작용을 상세히 설명한다.

[0019] 본 발명은 다양한 변환을 가할 수 있고 여러 가지 실시 예를 가질 수 있는 바, 특정 실시 예들을 도면에 예시하고 상세한 설명에 상세하게 설명하고자 한다. 본 발명의 효과 및 특징, 그리고 그것들을 달성하는 방법은 도면과 함께 상세하게 후술되어 있는 실시 예들을 참조하면 명확해질 것이다. 그러나 본 발명은 이하에서 개시되는 실시 예들에 한정되는 것이 아니라 다양한 형태로 구현될 수 있다.

[0020] 이하, 첨부된 도면을 참조하여 본 발명의 실시 예들을 상세히 설명하기로 하며, 도면을 참조하여 설명할 때 동일하거나 대응하는 구성 요소는 동일한 도면부호를 부여하고 이에 대한 중복되는 설명은 생략하기로 한다.

[0021] 이하에서, “상부” 나 “상”이라고 기재된 것은 접촉하여 바로 위에 있는 것 뿐만 아니라 비접촉으로 위에 있는 것도 포함할 수 있다.

[0022] 이하의 실시 예에서, 제1, 제2 등의 용어는 한정적인 의미가 아니라 하나의 구성 요소를 다른 구성 요소와 구별하는 목적으로 사용되었다.

[0023] 이하의 실시 예에서, 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다.

[0024] 이하의 실시 예에서, 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.

[0025] 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.

[0026] 또한, 본 명세서에 기재된 “…부”, “…영역” 등의 용어는 적어도 하나의 기능이나 동작을 처리하는 단위를 의미할 수 있다.

[0027] 도 1은, 본 개시의 실시예들에 따른 네트워크 환경(1)을 설명하는 도면이다.

[0028] 제1 전자 장치(10)는, 프로세서, 통신부, 메모리를 포함하는 컴퓨팅 장치를 말한다. 제1 전자 장치(10)는, 네트워크 망에 접속하여 제2 전자 장치와 통신할 수 있다. 제1 전자 장치(10)는, 함수 포인터 감시 모듈이 설치되어 설치된 프로그램에 대한 함수 포인터 감시를 실행시킬 수 있다.

[0029] 제1 전자 장치(10)는, 함수 포인터 감시 모듈의 동작에 의해 제1 프로그램을 정적 분석 도구를 이용하여 정적 분석하고, 함수 포인터 타입의 변수를 통해 함수가 호출되는 포인터 호출 코드를 검출할 수 있다.

[0030] 제1 전자 장치(10)는, 정적 분석 도구를 이용하여, 검출된 포인터 호출 코드에 대해서, 입력 변수와 출력 변수에 대한 유효값 리스트를 생성하여 저장할 수 있다. 제1 전자 장치(10)는, 유효값 리스트를 기초로 포인터 호출 코드에 대한 감시 코드를 생성하여 제1 프로그램 내에 삽입할 수 있다. 감시 코드는, 포인터 타입의 변수를 이용하여 함수가 호출되는 포인터 호출 코드가 호출되는 시점에 대응하여 추가되는 코드로서, 포인터 호출 코드가 실행되는 경우 유효값 리스트에 포함되는 값(value, data)로 호출되는 것인지 여부를 감시하는 코드일 수 있다. 예를 들어, 감시 코드는 제1 포인터 호출 코드가 제1 포인터 호출 코드에 대응하는 제1 유효값 리스트에 대응하는 값으로 호출되지 않는 경우, 제1 포인터 호출 코드의 시점에서 프로그램 실행을 종료 시키는 동작을 실행하는 명령어들의 집합 일 수 있다.

[0031] 제2 전자 장치(20)는, 통신부, 메모리 및 프로세서를 포함하는 컴퓨팅 장치로서, 네트워크 망을 통해 제1 전자 장치(10)로 접근하여 제1 전자 장치(10)를 제어하는 제어 신호를 전송하는 장치일 수 있다. 제2 전자 장치(20)는, 제1 전자 장치(10)에서 실행되는 프로그램을 이용하여 별개의 함수, 또는 모듈을 실행시키려고 할 수 있다. 제2 전자 장치(20)는, 제1 전자 장치(10)에서 함수 포인터 타입의 변수로 함수를 호출하는 프로그램 코드를 이용하여 제1 전자 장치(10)와 무관한 함수 또는 모듈이 실행되도록 할 수 있다. 제2 전자 장치(20)는, 포인터 호출 코드를 이용하여 제1 전자 장치(10)에 대한 공격, 어뮤징 등을 실행시킬 수 있다.

[0032] 이런 경우, 제1 전자 장치(10)의 감시 코드는, 제2 전자 장치(20)로부터 함수 포인터 타입의 변수인 포인터 호출 코드를 통한 접근 신호, 제어 신호, 실행 신호 등을 검출하고 이러한 검출에 대응하여 해당 프로그램을 강제 종료 시킬 수 있다.

[0033] 도 2는, 본 개시의 실시예들에 따른 함수 포인터 감시 장치(100)의 블록도이다.

[0034] 함수 포인터 감시 장치(100)는 함수 포인터 감시부가 설치되는 컴퓨팅 장치로서, 프로세서(110), 메모리(120), 통신부(130), 및 입출력부(140)를 포함할 수 있다. 함수 포인터 감시 장치(100)는, 도 1에 도시된 제1 전자 장치(10)와 대체 가능한 장치일 수 있다. 제1 전자 장치(10)는, 함수 포인터 감시 장치(100)와 동일한 구조와 기능을 가지는 장치일 수 있다.

[0035] 본 개시의 일 실시 예에 따르면, 프로세서(110)는 통상적으로, 함수 포인터 감시 장치(100)의 전반적인 동작을 제어한다. 예를 들어, 프로세서(110)는 함수 포인터 감시 장치(100)에 저장된 프로그램을 실행함으로써, 함수 포인터 감시 장치(100)가 포함하는 구성요소들을 전반적으로 제어할 수 있다.

[0036] 프로세서(110)는 메모리에 저장된 명령어들을 실행시켜, 프로그램에서 포인터 호출 코드를 검출하고, 포인터 호출 코드에 대응하는 지점에 감시 코드를 삽입할 수 있다. 프로세서(110)는, 감시 코드를 통해서 포인터 호출 코드가 어뮤징 호출되지 않도록 감시할 수 있다. 프로세서(110)는, 감시 코드를 포함하는 프로그램을 통해 포인터 호출 코드를 이용한 사이버 공격, 악성 코드, 랜섬웨어, 디도스, 해킹, 손상, 오용 등의 공격(attack)을 감지할 수 있다. 특히, 프로세서(110)는, 포인터 호출 코드를 이용하여 제어 흐름을 임의로 변경하려는 공격을 감시할 수 있다.

[0037] 본 개시의 일 실시 예에 따르면, 메모리(120)는 프로세서(110)의 처리 및 제어를 위한 프로그램을 저장할 수 있고, 함수 포인터 감시 장치(100)로 입력되거나 함수 포인터 감시 장치(100)로부터 출력되는 데이터를 저장할 수도 있다. 메모리(120)는 자율주행 프로그램을 검증하기 위한 프로그램을 저장할 수 있다. 메모리(120)는 상기의 정보들을 저장한 데이터베이스를 포함할 수 있다.

[0038] 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.

[0039] 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 자율주행 프로그램이 저장된 전자 장치에 접근하여 자율주행 프로그램을 수신할 수 있다.

[0040] 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, 함수 포인터 감시 장치(100)의 프로세서(110)는 메모리(120)에 로딩되거나, 저장된 컴퓨터 프로그램의 명령을 처리함에 있어서 서비스 화면이나 컨텐츠가 입출력부(140)를 통해 디스플레이에 표시될 수 있다.

[0041] 또한, 다른 실시 예들에서, 함수 포인터 감시 장치(100)는 도 2의 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.

[0042] 도 3는 메모리(120)의 블록도이다.

[0043] 메모리(120)는, 프로그램 감시부(121) 및 프로그램 코드(122)를 포함할 수 있다. 프로그램 감시부(121)는 메모리(120)에 저장된 프로그램 코드(122)를 분석하여 포인터 호출 코드를 검출하고, 함수 포인터 호출 코드에 대응하는 지점에 감시 코드를 삽입하고 감시 코드에 의해 함수 포인터 호출 코드에 대한 실행 상태를 모니터링할 수 있다. 프로그램 감시부(121)는, 감시 코드에 의해 포인터 호출 코드에 의해 정당한 사용이 아닌 실행이 검출되면 해당 프로그램을 강제 종료 시킬 수 있다.

[0044] 프로그램 감시부(121)는, 보안 프로그램에 내재되어 실행되도록 구현될 수 있다. 예를 들어, 프로그램 감시부(1211)는, 안티 바이러스 프로그램, 시스템 최적화 프로그램, 운영체제, 브라우저 등에 내재되어 함수 포인터 호출 코드를 검출하여 이에 대한 공격 행위를 모니터링할 수 있다.

[0045] 도 4는, 프로그램 감시부(121)의 블록도이다.

[0046] 프로그램 감시부(121)는, 장치(100) 내에 프로그램 코드에 대해서 감시 코드를 삽입하여 프로그램 실행시 함수 포인터 호출 코드를 통한 공격 행위를 감시하는 것으로, 소프트웨어 적으로 구현된 것일 수 있다. 프로그램 감시부(121)는, 기능을 구현하는 명령어들의 집합 또는 이러한 명령어들을 컴파일하여 생성된 프로그램의 형태일 수 있다. 프로그램 감시부(121)는, 정적 분석부(1211), 함수 포인터 검출부(1212), 감시 코드 삽입부(1213), 및/또는 모니터링부(1214)를 포함할 수 있다. 정적 분석부(1211), 함수 포인터 검출부(1212), 감시 코드 삽입부(1213), 및/또는 모니터링부(1214) 중 적어도 하나는, 소정의 코드로 작성된 명령어들의 집합 또는, 이러한 명령어들을 컴파일하여 생성된 프로그램의 형태일 수 있다.

[0047] 정적 분석부(1211)는 장치(100)내에 설치된 하나 이상의 프로그램 및/또는 프로그램 코드를 정적 분석할 수 있다. 정적 분석 도구를 이용하여 정적 분석할 수 있다. 정적 분석 도구로는, OOXML, OLE, point-to-analysis 등을 이용할 수 있으나, 이에 한정되지 않는다.

[0048] 함수 포인터 검출부(1212)는 프로그램 및/또는 프로그램 코드를 분석한 결과를 이용하여 포인터 호출 코드를 검출할 수 있다. 함수 포인터 검출부(1212)는 포인터 호출 코드가 가질 수 있는 하나 이상의 값들을 역방향 분석(backward analysis)을 통해 계산할 수 있다. 보다 구체적으로, 함수 포인터 검출부(1212)는 포인터 호출 코드에서 함수 호출의 변수에 대한 연산, 예를 들어, multiply와 add로 산출된 값을 이용하여 함수 포인터 타입의 데이터가 포인터 호출 코드인지 여부를 판단할 수 있다.

[0049] 함수 포인터 검출부(1212)는 프로그램 코드 안에서 함수 포인터 타입의 데이터를 검색하고 함수 포인터 타입의 데이터 중에서 포인터 호출 코드를 검출할 수 있다.

[0050] 함수 포인터 검출부(1212)는 프로그램의 포인터 변수가 가리키는 메모리 위치(주소)를 식별하여 분석할 수 있다. 함수 포인터 검출부(1212)는, 포인터 호출 코드에 대한 유효값 리스트를 획득할 수 있다. 함수 포인터 검출부(1212)는, 정적 분석 도구를 이용하여 유효값 리스트를 생성할 수 있다. 함수 포인터 검출부(1212)는, 역방향 분석을 이용하여 유효값 리스트를 생성할 수 있다.

[0051] 보다 구체적으로, 함수 포인터 검출부(1212)는, 프로그램 코드를 정적 분석하여, 포인터 호출 코드에 할당되는 메모리 영역을 검색하고, 메모리 영역에 할당되는 함수 등에 대한 정보를 획득하고, 포인터 호출 코드에 대한 유효값 리스트를 생성할 수 있다. 함수 포인터 검출부(1212)는, 프로그램 할당 데이터를 이용하여 함수 포인터에 의해 호출되는 함수를 시계열의 순서로 획득할 수 있다. 프로그램 할당 데이터는, PAG(program allocation graph)일 수 있으나, 이에 한정되지 않는다. 메모리 영역에 할당되는 함수를 모니터링하여 함수 포인터에 대한 유효값 리스트를 생성할 수 있다.

[0052] 다른 실시예에서, 함수 포인터 검출부(1212)는, 프로그램 코드를 정적 분석하여, 포인터 호출 코드에 할당되는 함수에 대한 값의 흐름에 대한 데이터를 획득하고, 함수에 대한 값의 흐름에 대한 데이터를 이용하여 포인터 호출 코드에 대한 유효값 리스트를 생성할 수 있다. 함수 포인터 검출부(1212)는, 정적 변수 흐름 데이터를 이용하여 포인터 호출 코드에 할당되는 함수의 입력 변수, 출력 변수 등이 어떻게 변화하는지에 대한 데이터를 획득할 수 있다. 정적 변수 흐름 데이터는, 노드, 엔터티, 흐름 에지, 유형 정보 등을 포함할 수 있다. 정적 변수 흐름 데이터는 프로그램의 실행 과정에서 변수에 대한 값이 어떻게 변화하는지에 대한 데이터, 하나의 변수에 대한 값을 변화시키는 코드들 사이의 관계를 포함할 수 있다.

[0053] 함수 포인터 검출부(1212)는, 데이터 타입이 함수 포인터인 변수라고 하더라도, 포인터 호출 코드에 해당하는지 여부를 판단하는 과정을 더 실행할 수 있다. 예를 들어, 함수 포인터 검출부(1212)는, 제1 함수 포인터가 1개의 함수에 대해서만 사용되는 경우, 제1 함수 포인터를 포인터 호출 코드의 함수 포인터로 검출하지 않을 수 있다. 함수 포인터 검출부(122)는, 함수 포인터가 사용되는 함수의 개수를 고려하여 함수 포인터가 포인터 호출 코드인지 여부를 결정할 수 있다. 함수 포인터 검출부(1212)는, 함수 포인터가 가리키는 메모리 영역(위치)를 고려하여 함수 포인터를 포인터 호출 코드로 넣을지 여부를 결정할 수 있다. 예를 들어, 제2 함수 포인터가 가리키는 메모리 영역이 프로그램의 실행 도중에 사용되지 않는 영역이라면, 제2 함수 포인터를 포인터 호출 코드에서 제외시킬 수 있다. 제3 함수 포인터가 가리키는 메모리 영역이 다른 변수에 의해 사용되는 영역이라면, 제3 함수 포인터를 포인터 호출 코드에 추가할 수 있다. 함수 포인터 검출부(1212)는, 함수 포인터의 택일적 사용 여부를 고려하여 함수 포인터를 포인터 호출 코드으로 넣을지 여부를 결정할 수 있다. 제4 함수 포인터가 조건문에 의해 사용되거나 사용되지 않는 등의 택일적으로 사용되는 경우, 제4 함수 포인터를 포인터 호출 코드로 추가할 수 있다.

[0054] 감시 코드 삽입부(1213)는 포인터 호출 코드의 입력 변수, 출력 변수가 유효값 리스트에 포함되는지 여부를 판단하고, 유효값 리스트에 포함되지 않는 경우, 해당 프로그램의 실행을 강제 종료 시키는 감시 코드를 프로그램 코드에 추가할 수 있다.

[0055] 감시 코드 삽입부(1213)는, 프로그램 코드를 입력 받고, 프로그램 코드에 대하여 감시 코드를 삽입하고, 감시 코드가 삽입된 프로그램 코드를 실행 파일로 생성하여 출력할 수 있다. 하나의 프로그램 코드에 복수의 감시 코드들이 삽입될 수 있다. 복수의 프로그램 코드에 복수의 감시 코드들이 삽입될 수 있다.

[0056] 감시 코드는, 함수 포인터에 의해 함수가 호출되었다는 것을 메시지로 출력하는 명령어와 함수 포인터에 할당되는 데이터가 유효값 리스트에 있는지 여부를 판단하고, 유효값 리스트에 포함되지 않는 경우, 해당 프로그램을 강제 종료 시키는 명령어를 포함할 수 있다. 감시 코드는, 프로그램 코드와 동일한 프로그래밍 언어로 구현될 수 있다.

[0057] 모니터링부(1214)는, 감시 코드가 삽입된 프로그램 코드를 컴파일하여 실행 파일을 생성할 수 있다.

[0058] 모니터링부(1214)는 프로그램 코드에 삽입된 감시 코드의 실행을 모니터링할 수 있다. 모니터링부(1214)는, 감시 코드에 의해 출력되는 메시지들을 프로그램 별, 및/또는 함수 별로 관리할 수 있다.

[0059] 모니터링부(1214)는, 제1 프로그램에 삽입된 제1 감시 코드 및/또는 제2 감시 코드가 실행됨에 따라 출력되는 메시지들을 파일로 생성하여 저장할 수 있다. 모니터링부(1214)는, 제1 감시 코드 및/또는 제2 감시 코드에 의해 제1 또는 제2 원시 포인터에 유효하지 않은 함수 할당, 함수 호출이 실행되는 것으로 검출된 경우, 제1 감시 코드 및/또는 제2 감시 코드에 의해 해당 프로그램의 실행이 강제 종료 될 수 있다.

[0060] 본 개시의 실시예들에 따른 과정은, 프로그램 코드를 실행 파일로 변환시키는 과정 중에 실행될 수 있다. 예를 들어, 프로그램 코드를 실행 파일로 변환하는 입력에 대응하여, 프로그램 코드에서 포인터 호출 코드를 검출하고, 그에 대한 감시 코드를 생성하여 삽입한 후, 실행 파일로 변환할 수 있다.

[0061] 이를 통해, 프로그램 코드를 실행 파일로 변환하는 과정을 통해서 함수 포인터에 대한 감시 코드가 추가된 프로그램이 생성될 수 있다.

[0062] 도 5는 본 개시의 실시예들에 따른 포인터 호출 코드를 감시하는 방법의 흐름도이다.

[0063] S110에서는, 함수 포인터 감시 장치(100)는, 프로그램 코드에서 함수 포인터 타입의변수를 포인터 호출 코드로 검출할 수 있다. 함수 포인터 감시 장치(100)는, 포인터 호출 코드가 가질 수 있는 하나 이상의 값들을 역방향 분석(backward analysis)을 통해 계산할 수 있다. 보다 구체적으로, 함수 포인터 감시 장치(100)는, 포인터 호출 코드에서 함수 호출의 변수에 대한 연산, 예를 들어, multiply와 add로 산출된 값을 이용하여 함수 포인터 타입의 데이터가 포인터 호출 코드인지 여부를 판단할 수 있다.

[0064] 함수 포인터 감시 장치(100)는, 프로그램 코드 안에서 함수 포인터 타입의 데이터를 검색하고 함수 포인터 타입의 데이터 중에서 포인터 호출 코드를 검출할 수 있다. 함수 포인터 감시 장치(100)는, 데이터 타입이 함수 포인터인 변수라고 하더라도, 포인터 호출 코드에 해당하는지 여부를 판단하는 과정을 더 실행할 수 있다. 예를 들어, 함수 포인터 감시 장치(100)는, 제1 함수 포인터가 1개의 함수에 대해서만 사용되는 경우, 제1 함수 포인터를 포인터 호출 코드의 함수 포인터로 검출하지 않을 수 있다. 함수 포인터 감시 장치(100)는, 함수 포인터가 가리키는 메모리 영역(위치)를 고려하여 함수 포인터를 감시 대상으로 넣을지 여부를 결정할 수 있다. 예를 들

어, 제2 함수 포인터가 가리키는 메모리 영역이 프로그램의 실행 도중에 사용되지 않는 영역이라면, 제2 함수 포인터를 포인터 호출 코드에서 제외시킬 수 있다. 제3 함수 포인터가 가리키는 메모리 영역이 다른 변수에 의해 사용되는 영역이라면, 제3 함수 포인터를 포인터 호출 코드에 추가할 수 있다. 함수 포인터 감시 장치(100)는, 함수 포인터의 택일적 사용 여부를 고려하여 함수 포인터를 포인터 호출 코드으로 넣을지 여부를 결정할 수 있다. 제4 함수 포인터가 조건문에 의해 사용되거나 사용되지 않는 등의 택일적으로 사용되는 경우, 제4 함수 포인터를 포인터 호출 코드로 추가할 수 있다.

[0065] S120에서는, 함수 포인터 감시 장치(100)는, 프로그램의 포인터 변수가 가리키는 메모리 위치(주소)를 식별하여 분석할 수 있다. 함수 포인터 감시 장치(100)는, 포인터 호출 코드에 대한 유효값 리스트를 획득할 수 있다. 함수 포인터 감시 장치(100)는, 정적 분석 도구를 이용하여 유효값 리스트를 생성할 수 있다 함수 포인터 감시 장치(100)는, 역방향 분석을 이용하여 유효값 리스트를 생성할 수 있다.

[0066] 보다 구체적으로, 함수 포인터 감시 장치(100)는, 프로그램 코드를 정적 분석하여, 포인터 호출 코드에 할당되는 메모리 영역을 검색하고, 메모리 영역에 할당되는 함수 등에 대한 정보를 획득하고, 포인터 호출 코드에 대한 유효값 리스트를 생성할 수 있다. 함수 포인터 감시 장치(100)는, 프로그램 할당 데이터를 이용하여 함수 포인터에 의해 호출되는 함수를 시계열의 순서로 획득할 수 있다. 프로그램 할당 데이터는, PAG(program allocation graph)일 수 있으나, 이에 한정되지 않는다. 메모리 영역에 할당되는 함수를 모니터링하여 함수 포인터에 대한 유효값 리스트를 생성할 수 있다.

[0067] 다른 실시예에서, 함수 포인터 감시 장치(100)는, 프로그램 코드를 정적 분석하여, 포인터 호출 코드에 할당되는 함수에 대한 값의 흐름에 대한 데이터를 획득하고, 포인터 호출 코드에 대한 유효값 리스트를 생성할 수 있다. 함수 포인터 감시 장치(100)는, 정적 변수 흐름 데이터를 이용하여 포인터 호출 코드에 할당되는 함수의 입력 변수, 출력 변수 등이 어떻게 변화하는지에 대한 데이터를 획득할 수 있다. 정적 변수 흐름 데이터는, 노드, 엔터티, 흐름 에지, 유형 정보 등을 포함할 수 있다. 정적 변수 흐름 데이터는 프로그램의 실행 과정에서 변수에 대한 값이 어떻게 변화하는지에 대한 데이터, 하나의 변수에 대한 값을 변화시키는 코드들 사이의 관계를 포함할 수 있다.

[0068] S130에서는, 함수 포인터 감시 장치(100)는, 포인터 호출 코드의 입력 변수, 출력 변수가 유효값 리스트에 포함되는지 여부를 판단하고, 유효값 리스트에 포함되지 않는 경우, 해당 프로그램의 실행을 강제 종료 시키는 감시 코드를 프로그램 코드에 추가할 수 있다.

[0069] 함수 포인터 감시 장치(100)는, 프로그램 코드를 입력 받고, 프로그램 코드에 대하여 감시 코드를 삽입하고, 감시 코드가 삽입된 프로그램 코드를 실행 파일로 생성하여 출력할 수 있다. 하나의 프로그램 코드에 복수의 감시 코드들이 삽입될 수 있다. 복수의 프로그램 코드에 복수의 감시 코드들이 삽입될 수 있다.

[0070] 감시 코드는, 함수 포인터에 의해 함수가 호출되었다는 것을 메시지로 출력하는 명령어와 함수 포인터에 할당되는 데이터가 유효값 리스트에 있는지 여부를 판단하고, 유효값 리스트에 포함되지 않는 경우, 해당 프로그램을 강제 종료 시키는 명령어를 포함할 수 있다. 감시 코드는, 프로그램 코드와 동일한 프로그래밍 언어로 구현될 수 있다.

[0071] S140에서는, 함수 포인터 감시 장치(100)는, 감시 코드가 삽입된 프로그램 코드를 컴파일하여 실행 파일을 생성할 수 있다.

[0072] S150에서는, 함수 포인터 감시 장치(100)는, 프로그램 코드에 삽입된 감시 코드의 실행을 모니터링할 수 있다. 함수 포인터 감시 장치(100)는, 감시 코드에 의해 출력되는 메시지들을 프로그램 별, 및/또는 함수 별로 관리할 수 있다.

[0073] 함수 포인터 감시 장치(100)는, 제1 프로그램에 삽입된 제1 감시 코드 및/또는 제2 감시 코드가 실행됨에 따라 출력되는 메시지들을 파일로 생성하여 저장할 수 있다. 함수 포인터 감시 장치(100)는, 제1 감시 코드 및/또는 제2 감시 코드에 의해 제1 또는 제2 원시 포인터에 유효하지 않은 함수 할당, 함수 호출이 실행되는 것으로 검출된 경우, 제1 감시 코드 및/또는 제2 감시 코드에 의해 해당 프로그램의 실행이 강제 종료 될 수 있다.

[0074] 도 6는 포인터 호출 코드에 대한 예시 도면이다.

[0075] 도 6에 도시된 바와 같이, funcPtr은 void 타입으로 선언된 함수 포인터일 수 있다. ‘funcPtr=hello’를 통해 funcPtr에 hello가 할당되어 funcPtr에 의해 함수 hello가 실행되게 된다.

[0076] 본 개시의 실시예들에 따라서, 이러한 형식을 가지는 코드를 포인터 호출 코드로 검출할 수 있다.

[0077] 도 7은 포인터 호출 코드가 사용되는 형식에 대한 예시 도면이다.

[0078] 도 7에 도시된 바와 같이, 포인터 타입 p는 z에 접근할 수 있다(p:= &z). 포인터 타입인 p에 의해 z 값이 2로 변경될 수 있다(*p := 2).

[0079] 포인터 타입은, 할당된 이후에, 할당된 데이터의 값을 간접적으로 변경하게 된다. 이러한 변경을 통해 프로그램에서 어떠한 데이터 값을 변경하게 할 수 있다.

[0080] 또한, 이상에서는 본 발명의 바람직한 실시예에 대하여 도시하고 설명하였지만, 본 발명은 상술한 특정의 실시예에 한정되지 아니하며, 청구범위에서 청구하는 본 발명의 요지를 벗어남이 없이 당해 발명이 속하는 기술분야에서 통상의 지식을 가진 자에 의해 다양한 변형실시가 가능한 것은 물론이고, 이러한 변형실시들은 본 발명의 기술적 사상이나 전망으로부터 개별적으로 이해되어져서는 안될 것이다.

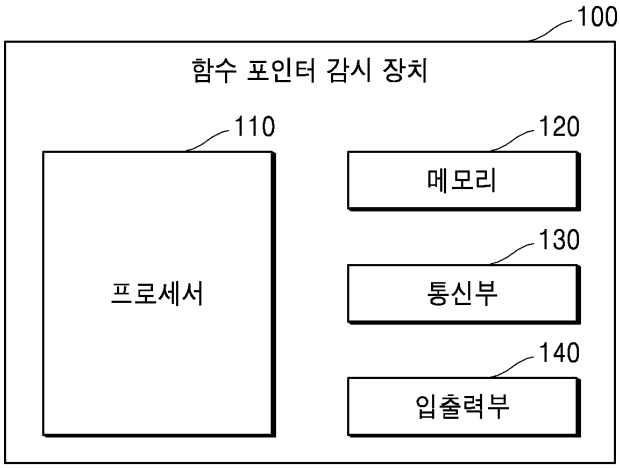
[0081] 따라서, 본 발명의 사상은 앞에서 설명된 실시예들에 국한하여 정해져서는 아니되며, 후술하는 특허청구범위뿐만 아니라 이 특허청구범위와 균등한 또는 이로부터 등가적으로 변경된 모든 범위가 본 발명의 사상의 범주에 속한다고 할 것이다.

도면

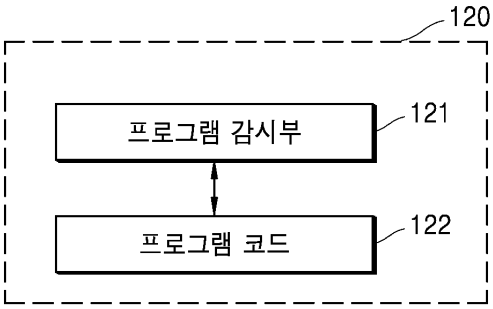
도면1



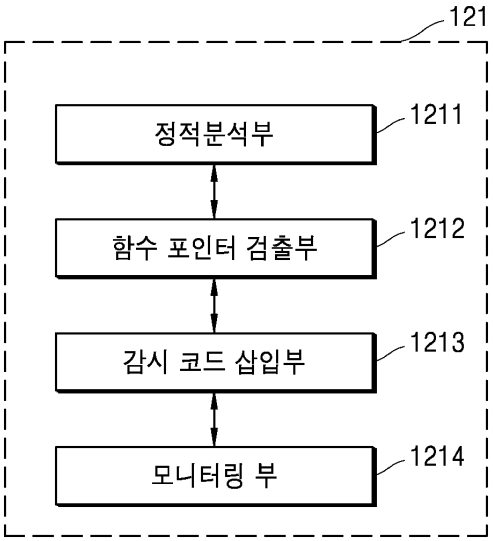
도면2



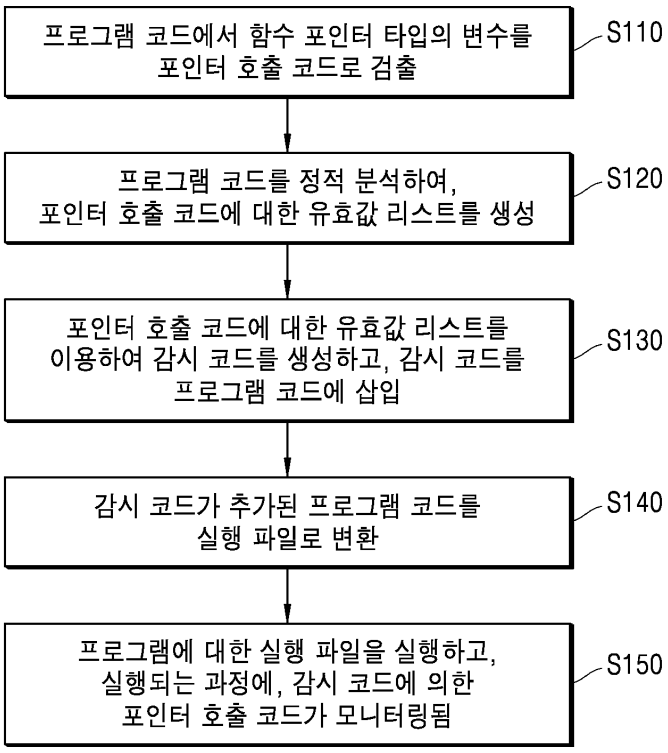
도면3


120
프로그램 감시부
121
프로그램 코드
122

도면4


121
정적분석부
1211
함수 포인터 검출부
1212
감시 코드 삽입부
1213
모니터링 부
1214

도면5


프로그램 코드에서 함수 포인터 타입의 변수를 포인터 호출 코드로 검출
S110
프로그램 코드를 정적 분석하여, 포인터 호출 코드에 대한 유효값 리스트를 생성
S120
포인터 호출 코드에 대한 유효값 리스트를 이용하여 감시 코드를 생성하고, 감시 코드를 프로그램 코드에 삽입
S130
감시 코드가 추가된 프로그램 코드를 실행 파일로 변환
S140
프로그램에 대한 실행 파일을 실행하고, 실행되는 과정에, 감시 코드에 의한 포인터 호출 코드가 모니터링됨
S150

도면6

```
#include <stdio.h>

// 함수 정의
void hello( ) {
    printf("Hello, World!\n");
}

void greet( ) {
    printf("Greetings, Earthling!\n");
}

int main( ) {
    // 함수 포인터 선언 및 초기화
    void (*funcPtr) ( );

    // 함수 포인터를 hello 함수로 설정
    funcPtr = hello;
    funcPtr( );    // Indirect call to hello( )

    // 함수 포인터를 greet 함수로 설정
    funcPtr = greet;
    funcPtr( );    // Indirect call to greet( )

    return 0;
}
```

도면7

$$
\begin{aligned}
1: &\quad z := 1 \\
2: &\quad p := \&z \\
3: &\quad *p := 2 \\
4: &\quad \text{ptint}\, z
\end{aligned}
$$

도면8

```
int main() {
    // Define a function pointer type that takes two ints and returns an
int
    typedef int (*funcPointer)(int, int);

    // Assign the address of the 'add' function to the function pointer
    funcPointer fPointer = add;
    std::cout << "Addition: " << fPointer(5, 3) << std::endl; // Indirect
call to 'add'

    // Finally, reassign the function pointer to the address of the
'multiply' function
    fPointer = multiply;
    std::cout << "Multiplication: " << fPointer(5, 3) << std::endl; //
Indirect call to 'multiply'
```