(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2014-0087768
(43) 공개일자 2014년07월09일

(51) 국제특허분류(Int. Cl.)
G06F 11/36 (2006.01) *G06F 11/07* (2006.01)
(21) 출원번호 10-2012-0158397
(22) 출원일자 2012년12월31일
심사청구일자 2012년12월31일

(71) 출원인
현대자동차주식회사
서울특별시 서초구 헌릉로 12 (양재동)
이화여자대학교 산학협력단
서울특별시 서대문구 이화여대길 52 (대현동, 이화여자대학교)
기아자동차주식회사
서울특별시 서초구 헌릉로 12 (양재동)
(72) 발명자
최병주
서울 강남구 압구정로 201, 74동 1305호 (압구정동, 현대아파트)
서주영
서울 송파구 중대로 24, 105동 304호 (문정동, 올림픽훼밀리타운)
(뒷면에 계속)
(74) 대리인
특허법인태평양

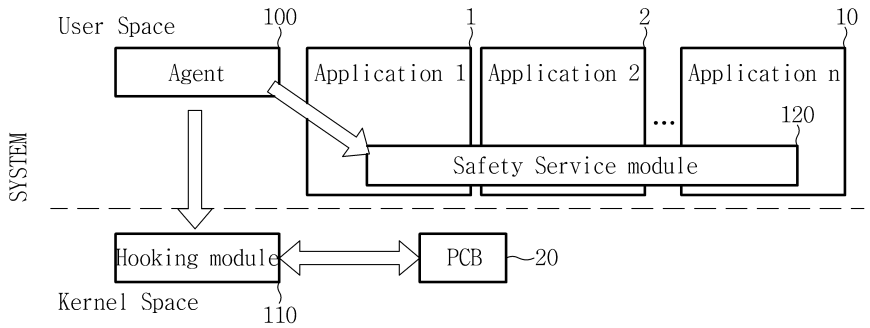
전체 청구항 수 : 총 17 항

(54) 발명의 명칭 **소프트웨어 검사 방법 및 시스템**

(57) 요 약

본 발명은 소프트웨어 검사 방법 및 시스템에 관한 것으로, 본 발명에 따른 시스템은 시스템 부팅 시 실행되어 커널 상에서 각 프로세스에 대응하는 PCB(Process Control Block) 정보를 수집하는 후킹 모듈, 및 상기 수집된 PCB 정보를 토대로 상기 프로세스의 메모리 영역에 삽입되어 상기 프로세스의 결함을 탐지하고 방어하는 세이프티 서비스(Safety Service) 모듈을 포함한다.

대 표 도 - 도1



(72) 발명자

장승연

경기 화성시 남양성지로 203-9, C-204 (남양동, 선주하이츠빌)

오정훈

경기 용인시 수지구 성복2로76번길 31, 105동 801호 (성복동, 대우푸르지오)

오정석

경기 부천시 원미구 상일로 71, 1815동 702호 (상동, 반달마을아파트)

노석영

경기 안양시 동안구 학의로 20, 131동 1101호 (비산동, 관악청구아파트)

양승완

경기 군포시 산본로386번길 21, 1137동 2202호 (산본동, 삼성장미아파트)

특허청구의 범위

청구항 1

커널 상에서 프로세스에 대응하는 PCB(Process Control Block)를 후킹하는 단계;

상기 PCB로부터 상기 프로세스의 주소값에 대한 실행정보를 획득하는 단계;

유효 주소값을 갖는 메모리 영역에 세이프티 서비스(Safety Service) 모듈을 삽입하는 단계; 및

상기 프로세스 실행 중 상기 세이프티 서비스 모듈이 삽입된 메모리 영역이 호출되는 경우에 해당 메모리 영역에 삽입된 세이프티 서비스 모듈에 의해 상기 프로세스의 결함을 탐지하는 단계를 포함하는 것을 특징으로 하는 소프트웨어 검사 방법.

청구항 2

청구항 1에 있어서,

상기 PCB는,

커널 내부에서, 상기 프로세스의 이름, ID, 우선순위 및 주소값 중 적어도 하나에 대한 프로세스 정보, 및 힙 프로세서, 공유객체, 파일 및 뮤텍스 중 적어도 하나에 대한 런타임 자원 정보를 실시간으로 관리하는 것을 특징으로 하는 소프트웨어 검사 방법.

청구항 3

청구항 1에 있어서,

상기 세이프티 서비스를 삽입하는 단계는,

상기 세이프티 서비스 데이터 및 정보태그에 대한 저장 공간을 할당하는 단계; 및

상기 정보태그의 저장 공간에 상기 할당된 저장 공간의 크기정보를 저장하는 단계를 포함하는 것을 특징으로 하는 소프트웨어 검사 방법.

청구항 4

청구항 3에 있어서,

상기 세이프티 서비스 데이터가 할당된 저장 공간의 주소정보를 실행 어플리케이션에 제공하는 단계를 포함하는 것을 특징으로 하는 소프트웨어 검사 방법.

청구항 5

청구항 3에 있어서,

상기 프로세스의 결함을 탐지하는 단계는,

상기 할당된 저장 공간에 접근 이벤트 발생 시, 상기 정보태그의 저장 공간을 검사하는 단계; 및

상기 접근 이벤트의 접근 범위가 상기 정보태그에 저장된 상기 저장 공간의 크기정보에 대해 유효한 범위인지 확인하는 단계를 포함하는 것을 특징으로 하는 소프트웨어 검사 방법.

청구항 6

청구항 5에 있어서,

상기 유효한 범위인지 확인하는 단계의 확인 결과 유효한 범위가 아닌 경우에 상기 접근 이벤트의 접근을 무시하는 단계를 포함하는 것을 특징으로 하는 소프트웨어 검사 방법.

청구항 7

청구항 5에 있어서,

상기 유효한 범위인지 확인하는 단계의 확인 결과 유효한 범위가 아닌 경우에 상기 접근 이벤트의 접근 범위를 유효한 범위로 조절하는 단계를 포함하는 것을 특징으로 하는 소프트웨어 검사 방법.

청구항 8

청구항 3에 있어서,

상기 프로세스의 결함을 탐지하는 단계는,

상기 할당된 저장 공간에 해제 이벤트 발생 시, 상기 정보태그의 저장 공간을 검사하는 단계; 및

상기 정보태그에 저장된 정보에 근거하여 상기 해제 이벤트가 발생한 저장 공간이 해제 가능한 유효 주소 공간인지 확인하는 단계를 포함하는 것을 특징으로 하는 소프트웨어 검사 방법.

청구항 9

청구항 8에 있어서,

상기 유효 주소 공간인지 확인하는 단계의 확인 결과 유효 주소 공간인 경우에 해당 저장 공간에 대한 해제 이벤트를 수행하는 단계를 포함하는 것을 특징으로 하는 소프트웨어 검사 방법.

청구항 10

청구항 9에 있어서,

상기 해제 이벤트를 수행하는 단계 이후에 해당 주소 공간에 할당된 변수를 초기화하는 단계를 포함하는 것을 특징으로 하는 소프트웨어 검사 방법.

청구항 11

청구항 1에 있어서,

상기 프로세스의 결함을 탐지하는 단계에서 탐지된 결함에 대응하는 방어 액션을 수행하는 단계를 더 포함하는 것을 특징으로 하는 소프트웨어 검사 방법.

청구항 12

청구항 11에 있어서,

상기 방어 액션을 수행하는 단계는,

무시 액션, 지속 액션, 경고 액션, 반복 액션 및 종료 액션 중 상기 탐지된 결함 유형에 대응하는 방어 액션을 수행하는 것을 특징으로 하는 소프트웨어 검사 방법.

청구항 13

시스템 부팅 시 실행되어 커널 상에서 각 프로세스에 대응하는 PCB(Process Control Block) 정보를 수집하는 후킹 모듈; 및

상기 수집된 PCB 정보를 토대로 상기 프로세스의 메모리 영역에 삽입되어 상기 프로세스의 결함을 탐지하고 방어하는 세이프티 서비스(Safety Service) 모듈을 포함하는 것을 특징으로 하는 소프트웨어 검사 시스템.

청구항 14

청구항 13에 있어서,

상기 PCB는,

커널 내부에서, 상기 프로세스의 이름, ID, 우선순위 및 주소값 중 적어도 하나에 대한 프로세스 정보, 및 힙 프로세서, 공유객체, 파일 및 뮤텍스 중 적어도 하나에 대한 런타임 자원 정보를 실시간으로 관리하는 것을 특징으로 하는 소프트웨어 검사 시스템.

청구항 15

청구항 13에 있어서,

상기 세이프티 서비스(Safety Service) 모듈은,

결함 탐지 루틴 및 능동 방어 루틴을 포함하는 것을 특징으로 하는 소프트웨어 검사 시스템.

청구항 16

청구항 15에 있어서,

상기 결함 탐지 루틴은,

상기 메모리 영역에 할당된 정보태그를 이용하여 유효 범위 또는 유효 주소값을 확인하고, 확인 결과에 따라 입력 이벤트에 대한 상기 프로세스의 결함을 탐지하는 것을 특징으로 하는 소프트웨어 검사 시스템.

청구항 17

청구항 15에 있어서,

상기 능동 방어 루틴은,

무시 액션, 지속 액션, 경고 액션, 반복 액션 및 종료 액션 중 적어도 하나에 대한 방어 액션을 정의하고, 상기 정의된 방어 액션 중 상기 결함 탐지 루틴에 의해 탐지된 결함 유형에 대응하는 방어 액션을 수행하는 것을 특징으로 하는 소프트웨어 검사 시스템.

명 세 서

기 술 분 야

[0001] 본 발명은 소프트웨어 검사 방법 및 시스템에 관한 것으로, 프로세서의 특정 메모리 영역을 세이프티 서비스(Safety Service) 모듈로 바꿔치기하여 해당되는 결함 탐지 액션 및 능동 방어 액션이 수행되도록 하는 기술에 관한 것이다.

배 경 기 술

[0002] 능동 방어란 공격이 오는 것을 예측하여 해당 공격을 무력화시키는 방안으로 일찍이 국방무기체계시스템에서 시작된 연구이다. IT 분야에서는 웹과 네트워크 도메인에서 악성코드 공격에 대한 시스템 보안 유지 방안으로 능동 방어에 대한 연구가 활발하게 이루어지고 있다. 즉, 네트워크 방화벽처럼 서로 신뢰수준이 다른 네트워크를 지나는 데이터를 검열하여 바이러스나 DDoS 공격과 같이 시스템 보안에 위협이 될 요소를 탐지하고 이를 거부하는 활동이 능동방어의 대표적 사례라 할 수 있다.

[0003] 대부분의 능동 방어 연구는 서로 기능적으로 독립된 시스템들 사이의 공격과 방어의 문제를 다루고 있다. 즉 시스템 위협 요소는 시스템 외부에 존재한다고 가정하며, 신뢰할 수 없는 외부 시스템으로부터의 공격을 예측하거나 탐지하고, 이를 방어함으로 내부 시스템의 안전을 유지하려 한다.

[0004] 한편, 시스템 내의 프로그램은 언제든 문제가 있을 수 있고, 변경이 생길 수 있으므로, 이러한 문제를 방지하기 위해 예외 처리 및 안전 코드로 설계하는 것이 필요하다. 그러나, 일반적인 소프트웨어는 철저한 예외 처리가 오히려 시스템 성능에 부담을 줄 수도 있기 때문에, 서로 대립되는 요소 사이의 균형을 고려할 수 밖에 없다. 또한, 시스템 전체의 입장에서도 각각의 어플리케이션이 모든 예외에 대한 처리 코드를 개별적으로 보유하는 방식은 많은 오버헤드가 따른다.

발명의 내용

해결하려는 과제

[0005] 본 발명의 목적은, 프로세서의 특정 메모리 영역을 세이프티 서비스 모듈로 바꿔치기함으로써 후킹과 정보태깅 기술을 활용하여 해당되는 결함 탐지 액션 및 능동 방어 액션을 수행함에 따라 시스템의 기본 행위가 방해되는

것을 최소화하면서 결함 발생을 탐지하도록 하는 소프트웨어 검사 방법 및 시스템을 제공함에 있다.

[0006] 또한, 본 발명의 다른 목적은, 결함이 탐지되어도 시스템 본연의 기능을 보존하면서 결함을 예방할 수 있도록 결함 유형에 따라 정의된 다양한 능동 방어 액션을 구현하는 소프트웨어 검사 방법 및 시스템을 제공함에 있다.

[0007] 또한, 본 발명의 또 다른 목적은, 어플리케이션 각각에 대한 개별 수준이 아닌 시스템을 관리하는 커널 레벨에서 런-타임 결함들에 대한 능동 방어 액션를 지원하여 성능적으로 효율적이면서도 방어적 설계가 가능한 소프트웨어 검사 방법 및 시스템을 제공함에 있다.

과제의 해결 수단

[0008] 상기의 목적을 달성하기 위한 본 발명에 따른 소프트웨어 검사 방법은, 커널 상에서 프로세스에 대응하는 PCB(Process Control Block)를 후킹하는 단계, 상기 PCB로부터 상기 프로세스의 주소값에 대한 실행정보를 획득하는 단계, 유효 주소값을 갖는 메모리 영역에 세이프티 서비스(Safety Service) 모듈을 삽입하는 단계, 및 상기 프로세스 실행 중 상기 세이프티 서비스 모듈이 삽입된 메모리 영역이 호출되는 경우에 해당 메모리 영역에 삽입된 세이프티 서비스 모듈에 의해 상기 프로세스의 결함을 탐지하는 단계를 포함하는 것을 특징으로 한다.

[0009] 상기 PCB는, 커널 내부에서, 상기 프로세스의 이름, ID, 우선순위 및 주소값 중 적어도 하나에 대한 프로세스 정보, 및 힘 프로세서, 공유객체, 파일 및 뮤텍스 중 적어도 하나에 대한 런타임 자원 정보를 실시간으로 관리하는 것을 특징으로 한다.

[0010] 상기 세이프티 서비스를 삽입하는 단계는, 상기 세이프티 서비스 데이터 및 정보태그에 대한 저장 공간을 할당하는 단계 및 상기 정보태그의 저장 공간에 상기 할당된 저장 공간의 크기정보를 저장하는 단계를 포함하는 것을 특징으로 한다.

[0011] 또한, 본 발명에 따른 소프트웨어 검사 방법은, 상기 세이프티 서비스 데이터가 할당된 저장 공간의 주소정보를 실행 어플리케이션에 제공하는 단계를 더 포함하는 것을 특징으로 한다.

[0012] 상기 프로세스의 결함을 탐지하는 단계는, 상기 할당된 저장 공간에 접근 이벤트 발생 시, 상기 정보태그의 저장 공간을 검사하는 단계, 및 상기 접근 이벤트의 접근 범위가 상기 정보태그에 저장된 상기 저장 공간의 크기정보에 대해 유효한 범위인지 확인하는 단계를 포함하는 것을 특징으로 한다.

[0013] 또한, 본 발명에 따른 소프트웨어 검사 방법은, 상기 유효한 범위인지 확인하는 단계의 확인 결과 유효한 범위가 아닌 경우에 상기 접근 이벤트의 접근을 무시하는 단계를 더 포함하는 것을 특징으로 한다.

[0014] 또한, 본 발명에 따른 소프트웨어 검사 방법은, 상기 유효한 범위인지 확인하는 단계의 확인 결과 유효한 범위가 아닌 경우에 상기 접근 이벤트의 접근 범위를 유효한 범위로 조절하는 단계를 더 포함하는 것을 특징으로 한다.

[0015] 상기 프로세스의 결함을 탐지하는 단계는, 상기 할당된 저장 공간에 해제 이벤트 발생 시, 상기 정보태그의 저장 공간을 검사하는 단계, 및 상기 정보태그에 저장된 정보에 근거하여 상기 해제 이벤트가 발생한 저장 공간이 해제 가능한 유효 주소 공간인지 확인하는 단계를 포함하는 것을 특징으로 한다.

[0016] 또한, 본 발명에 따른 소프트웨어 검사 방법은, 상기 유효 주소 공간인지 확인하는 단계의 확인 결과 유효 주소 공간인 경우에 해당 저장 공간에 대한 해제 이벤트를 수행하는 더 단계를 포함하는 것을 특징으로 한다.

[0017] 또한, 본 발명에 따른 소프트웨어 검사 방법은, 상기 해제 이벤트를 수행하는 단계 이후에 해당 주소 공간에 할당된 변수를 초기화하는 단계를 더 포함하는 것을 특징으로 한다.

[0018] 또한, 본 발명에 따른 소프트웨어 검사 방법은, 상기 프로세스의 결함을 탐지하는 단계에서 탐지된 결함에 대응하는 방어 액션을 수행하는 단계를 더 포함하는 것을 특징으로 한다.

[0019] 상기 방어 액션을 수행하는 단계는, 무시 액션, 지속 액션, 경고 액션, 반복 액션 및 종료 액션 중 상기 탐지된 결함 유형에 대응하는 방어 액션을 수행하는 것을 특징으로 한다.

[0020] 한편, 상기의 목적을 달성하기 위한 본 발명에 따른 소프트웨어 검사 시스템은, 시스템 부팅 시 실행되어 커널 상에서 각 프로세스에 대응하는 PCB(Process Control Block) 정보를 수집하는 후킹 모듈, 및 상기 수집된 PCB 정보를 토대로 상기 프로세스의 메모리 영역에 삽입되어 상기 프로세스의 결함을 탐지하고 방어하는 세이프티 서비스(Safety Service) 모듈을 포함하는 것을 특징으로 한다.

발명의 효과

[0021] 본 발명에 따르면, 프로세서의 특정 메모리 영역을 세이프티 서비스 모듈로 바꿔치기함으로써 후킹과 정보태깅 기술을 활용하여 해당되는 결함 탐지 액션 및 능동 방어 액션을 수행함에 따라 시스템의 기본 행위가 방해되는 것을 최소화하면서 결함 발생을 탐지할 수 있는 이점이 있다.

[0022] 또한, 본 발명은, 결함 유형에 따라 다양한 능동 방어 액션을 정의함으로써 결함이 탐지되어도 시스템 본연의 기능을 보존하면서 결함을 예방할 수 있는 이점이 있다.

[0023] 또한, 본 발명은, 어플리케이션 각각에 대한 개별 수준이 아닌 시스템을 관리하는 커널 레벨에서 런-타임 결함들에 대한 능동 방어 액션를 지원하여 성능적으로 효율적이면서도 방어적 설계가 가능한 이점이 있으며, 이로 인해 시스템 내의 모든 어플리케이션들에게 동등한 레벨의 신뢰도를 지원할 수 있는 이점이 있다.

도면의 간단한 설명

[0024] 도 1은 본 발명에 따른 소프트웨어 검사 시스템 구성을 도시한 도이다.

도 2는 본 발명에 따른 소프트웨어 검사 시스템의 개략적인 동작을 나타낸 도이다.

도 3은 본 발명에 따른 소프트웨어 검사 방법에 대한 동작 흐름을 도시한 순서도이다.

도 4는 본 발명에 적용되는 PCB 구조를 도시한 도이다.

도 5는 본 발명에 따른 세이프티 서비스 모듈이 할당된 저장 공간의 구조를 도시한 예시도이다.

도 6은 본 발명에 따른 세이프티 서비스 모듈의 실행 코드를 도시한 예시도이다.

도 7a 내지 도 7d는 본 발명에 적용되는 코드를 도시한 예시도이다.

도 8은 본 발명에 따른 소프트웨어 검사 시스템의 능동 방어 동작을 설명하는데 참조되는 예시도이다.

도 9a 내지 도 9c는 본 발명에 따른 소프트웨어 검사 시스템의 능동 방어 유형별 코드를 도시한 예시도이다.

발명을 실시하기 위한 구체적인 내용

[0025] 이하, 첨부된 도면을 참조하여 본 발명의 실시예를 설명하도록 한다.

[0026] 도 1은 본 발명에 따른 소프트웨어 검사 시스템 구성을 도시한 도이고 도 2는 소프트웨어 검사 시스템의 개략적인 동작을 나타낸 도이다. 도 1 및 도 2를 참조하면, 본 발명에 따른 소프트웨어 검사 시스템은 시스템 부팅 시 실행되어 각 어플리케이션(1~10)의 프로세스에 대응하는 PCB(Process Control Block)(20)에 대한 정보를 수집하는 후킹 모듈(110) 및 수집된 PCB(20)의 정보를 토대로 프로세스의 특정 메모리 영역에 삽입되어 시스템 내에서 프로세스의 결함을 탐지하고 방어하는 세이프티 서비스(Safety Service) 모듈(120)을 포함한다.

[0027] 여기서, 본 발명에 따른 소프트웨어 검사 시스템의 에이전트(agent)(100), 즉, ROPHE AD 에이전트는 후킹 모듈(110) 및 세이프티 서비스 모듈(120)을 관리한다. 여기서, ROPHE AD는 'RemOte run-time Protection for Highrisk Error - Active Defensor'의 약자로서, 임베디드 리눅스 플랫폼에서 동작하는 자동화 도구이다.

[0028] 한편, 후킹 모듈(110)은 커널 상에 존재하는 모듈로서, 커널 상에 존재하는 PCB(20)를 후킹하여 프로세스의 메모리 영역에 대한 실행정보를 획득한다. 본 발명에 적용되는 후킹 기법은 실행 경로를 가로채는 대표적인 기술로 시스템의 소프트웨어 실행 상황을 런타임으로 파악하기에 유용한 방식이다. 이에, 본 발명은 후킹 기술을 적용함으로써 시스템 기본 행위가 방해되는 것을 최소화하면서 결함이 발생하는 상황을 감시할 수 있게 된다.

[0029] 이때, 후킹 모듈(110)은 도 2에서의 (1)과 같이, 획득한 정보를 소프트웨어 검사 시스템의 에이전트(20)로 제공한다.

[0030] 세이프티 서비스 모듈(120)은 각 어플리케이션(1~10)의 각 프로세스에서 결함이 발생한 만한 메모리 영역에 삽입되어, 프로세스 실행 시 해당 메모리 영역에서 세이프티 서비스 루틴으로 바꿔치기하여 실행하도록 한다.

[0031] 다시 말해, 에이전트(100)는 도 2에서의 (2)와 같이 후킹 모듈(110)에 의해 후킹된 PCB 정보에 근거하여 세이프티 서비스 모듈(120)을 각 어플리케이션(1~10)에 삽입(injection)하고, 도 2에서의 (3)과 같이 각 어플리케이션(1~10)에 삽입된 세이프티 서비스 모듈(120)을 통해 프로세스에 대한 공격을 가로채기(interception) 하여 능동

방어를 수행하도록 한다.

[0032] 이때, 각 어플리케이션(1~10)에 삽입되는 세이프티 서비스 모듈(120)은 프로세스의 결함 발생을 예측하는 결함 탐지 루틴 및 결함 유형별로 방어 기능을 수행하는 능동방어 루틴을 포함한다. 여기서, 결함 탐지 루틴은 입력받은 포인터 변수가 유효한 메모리 주소값 인지를 판단하고, 능동방어 루틴은 유효하지 않은 주소값인 경우 안전한 NULL값으로 초기화를 해 준 후 기능을 지속하도록 함으로써 결함이 발생하는 것을 막는다.

[0033] 따라서, 세이프티 서비스 모듈(120)은 에이전트(100)로부터 제공된 PCB 정보를 활용하여 결함 탐지 루틴을 실행하고, 결함 탐지 루틴의 실행 결과에 따라 능동방어 루틴을 실행하게 된다.

[0034] 상기와 같이 구성되는 본 발명에 따른 소프트웨어 검사 시스템의 동작 흐름을 보다 상세히 설명하면 다음과 같다.

[0035] 도 3은 본 발명에 따른 소프트웨어 검사 시스템의 소프트웨어 검사 방법에 대한 동작 흐름을 도시한 순서도이다. 도 3을 참조하면, 본 발명에 따른 소프트웨어 검사 시스템은 후킹 모듈을 이용하여 커널 상에서 프로세스에 대응하는 PCB(Process Control Block)를 후킹하여(S100), PCB로부터 해당 프로세스의 주소 공간에 대한 실행정보를 획득한다(S110). 여기서, PCB는 커널 상에 존재하며, 해당 프로세스의 이름, ID, 우선순위 및 주소값 중 적어도 하나에 대한 프로세스 정보, 및 힘 프로세서, 공유객체, 파일 및 뮤텍스 중 적어도 하나에 대한 런타임 자원 정보를 저장하고 실시간으로 관리한다.

[0036] 한편, 소프트웨어 검사 시스템은 'S110' 과정에서 획득한 정보에 근거하여 프로세스의 유효 주소값을 갖는 메모리 영역에 세이프티 서비스 모듈을 삽입한다. 이때, 삽입되는 세이프티 서비스 모듈은 프로세스의 결함 발생을 예측하는 결함 탐지 루틴 및 결함 유형별로 방어 기능을 수행하는 능동방어 루틴을 포함한다.

[0037] 따라서, 프로세스의 유효 주소값을 갖는 메모리 영역에 삽입된 세이프티 서비스 모듈은 프로세스 실행 시 해당 메모리 영역이 호출되면 결함 탐지 루틴을 실행하여 프로세스의 결함을 탐지하고(S130), 결함이 탐지되는 경우에는 능동방어 루틴을 실행하여 프로세스의 결함에 대한 능동 방어를 수행하도록 한다(S140).

[0038] 여기서, 세이프티 서비스 모듈의 결함 탐지 루틴 및 능동방어 루틴은 도 6 내지 도 9c를 참조하여 보다 상세히 설명하도록 한다.

[0039] 도 4는 본 발명에 적용되는 PCB 구조를 도시한 도이다. 도 4에 도시된 바와 같이, 본 발명에 적용되는 PCB는 프로세스 정보 및 런타임 자원 정보가 저장된다.

[0040] 일 예로서, PCB는 대응되는 프로세스와 관련하여, Process ID, Process Handle, Memory Pointer, Base Pointer of EXE Load, Process Name, Program Counter (PC), Export Rable Position, Import Table Position, Resource Table Position,Virtual Base Address of Module, Maximum Stack Size, Nunber of Memory Objects, 및 Priority State 등의 정보를 저장하고, 프로세스의 상태에 따라 저장된 정보를 실시간으로 관리한다.

[0041] 도 5는 본 발명에 따른 세이프티 서비스 모듈이 할당된 메모리 영역의 구조를 도시한 예시도이다. 본 발명에 따른 소프트웨어 검사 시스템의 에이전트가 세이프티 서비스 모듈을 프로세스의 메모리 영역에 삽입하는 경우, 해당 어플리케이션은 유효 주소값의 메모리 영역에서 세이프티 서비스 모듈에 대한 저장 공간(520)을 할당한다. 이때, 세이프티 서비스 모듈 외에 런타임 실행 정보를 함께 저장하기 위한 정보태그의 저장 공간(510)을 추가로 할당한다.

[0042] 정보태그 및 세이프티 서비스 모듈에 대해 할당된 저장 공간(510, 520)은 도 5에 도시된 바와 같다. 이때, 정보태그의 저장 공간(510)에는 세이프티 서비스 모듈에 대해 할당된 저장 공간(520)의 크기 정보가 저장된다. 이때, 세이프티 서비스 모듈의 결함 탐지 루틴은 정보태그에 저장된 저장 공간(520)의 크기 정보를 이용하여 해당 메모리 영역의 주소값이 유효한 주소 영역에 포함되는지를 판단함으로써 해당 메모리 영역의 결함을 탐지하게 된다. 물론, 정보태그의 저장 공간(510)은 결함 유형에 따라 저장 공간을 확장하여 다양한 정보를 저장할 수 있도록 한다.

[0043] 다만, 정보태그 및 세이프티 서비스 모듈이 할당된 저장 공간(510, 520)에 대한 시작 주소값은 세이프티 모듈이 할당된 저장 공간(520)의 시작 주소값을 해당 어플리케이션에 제공하는 것으로 하며, 정보태그의 저장 공간(510)에 대한 정보는 커널 수준에서만 인식할 수 있는 히든 공간인 것으로 한다.

[0044] 도 6은 본 발명에 따른 세이프티 서비스 모듈의 실행 코드를 도시한 예시도이다. 도 6을 참조하면, 각 어플리케이션에 삽입되는 세이프티 서비스 모듈은 프로세스의 결함 발생을 예측하는 결함 탐지 루틴 및 결함 유형별로

방어 기능을 수행하는 능동방어 루틴을 포함한다. 이때, 세이프티 서비스 모듈은 도 6에 도시된 3)의 오리지날 서비스(Original service)의 주소값을 세이프티 서비스의 주소값으로 바꿔치기함으로써, 해당 메모리 영역의 오리지날 서비스 실행이 요청되면 세이프티 서비스가 실행되게 된다.

[0045] 세이프티 서비스가 실행되면, 우선 1)의 결함 감지 액션(Fault detection action)에 대한 실행 코드가 동작하게 되고, 결함 감지 루틴에 의해 결함이 감지되면, 2)의 능동 방어 액션(Active defedse action)에 대한 실행코드가 동작하여 발생된 결함에 대한 방어를 수행한다.

[0046] 만일, 결함 감지 루틴에 의해 결함이 감지되지 않은 경우에는 3)의 오리지날 서비스가 실행되게 된다.

[0047] 일 예로서, 접근 이벤트에 의해 세이프티 서비스 모듈이 할당된 저장 공간을 포함하는 메모리 영역이 호출되는 경우 결함 탐지 루틴이 실행되게 되고, 결함 탐지 루틴은 세이프티 서비스 모듈이 할당된 저장 공간에 대한 크기 정보가 저장된 정보태그의 저장 공간을 먼저 호출하여 우선 검사하는 것으로 한다. 이 경우, 결함 탐지 루틴은 정보태그의 저장 공간에 저장된 저장 공간의 크기 정보를 토대로 접근 이벤트에 의한 접근 범위가 유효한 범위인지를 검사할 수 있다.

[0048] 물론, 접근 이벤트에 의한 접근 범위가 유효한 범위가 아니라면, 능동방어 루틴은 상황에 따라 해당 메모리 영역으로의 접근을 무시하거나 범위를 유효범위로 조정하여 실행을 이어갈 수 있다.

[0049] 다른 예로서, 해제 이벤트에 의해 세이프티 서비스 모듈이 할당된 저장 공간을 포함하는 메모리 영역이 호출되는 경우, 결함 탐지 루틴은 정보태그의 저장 공간을 호출하여 해당 메모리 영역의 주소값이 유효 주소값 인지를 확인할 수 있다. 만일, 해당 메모리 영역의 유효 주소값인 경우, 능동방어 루틴은 정보태그를 포함한 메모리 영역에 대한 해제 이벤트를 수행하며, 해당 변수를 NULL 값으로 초기화 해줌으로써 해제한 메모리 영역의 주소값에 접근하는 오류를 줄일 수 있도록 한다.

[0050] 한편, 해당 메모리 영역의 주소값이 유효 주소값이 아닌 경우, 예를 들어, 이미 해제된 주소값인 경우에, 능동방어 루틴은 중복 해제로 인해 시스템이 다운되지 않도록 해제 이벤트를 무시할 수 있다.

[0051] 도 7a 내지 도 7d는 본 발명에 적용되는 코드를 도시한 예시도이다.

[0052] 먼저, 도 7a는 포인터 변수가 NULL값으로 초기화되어, 포인터에 메모리가 아직 할당되기 전임을 판단할 수 있는 경우를 나타낸 것이다. 도 7b는 포인터 변수를 초기화하지 않아 쓰레기 값(garbage value)을 지니고 있는 경우를 나타낸 것이다.

[0053] 한편, 도 7c는 메모리 결함 발생을 막기 위해 입력 값을 검사하는 코드를 지닌 메모리 해지 코드의 일 실시예를 나타낸 것이다. 도 7c의 메모리 해지 코드가 실행되는 경우, 도 7a에 도시된 실시예는 포인터 변수가 NULL값으로 초기화되어 있어, 초기화된 포인터 변수가 입력이 될 때는 해당 포인터의 주소값이 유효한 번지로 잘못 오해되어 결함을 만들 가능성이 없다. 한편, 도 7b에 도시된 실시예는 도 7c의 메모리 해지 코드가 실행되는 경우, 포인터에 메모리가 할당되어 유효한 값을 갖고있는 지를 판단하기 어렵기 때문에 메모리 결함이 발생할 가능성이 매우 높다.

[0054] 따라서, 도 7d에 도시된 세이프티 서비스의 경우에는, 결함 탐지 루틴이 입력받은 포인터 변수가 유효한 메모리 주소인지를 판단하고, 유효하지 않은 주소인 경우 능동 방어 루틴이 포인터 변수를 안전한 NULL값으로 초기화를 해 준 후 해당 기능을 지속하기 때문에, 도 7c의 메모리 해지 코드가 실행되는 경우에 결함이 발생하는 것을 막을 수 있게 된다.

[0055] 도 8 내지 도 9c는 본 발명에 따른 소프트웨어 검사 시스템의 능동 방어 동작을 설명하는데 참조되는 예시도이다.

[0056] 도 8에 도시된 바와 같이, 능동 방어 루틴은 결함 탐지 루틴에 의해 탐지된 결함 유형에 따라 무시(ignore), 지속(continue), 경고(warning), 반복(repeat), 및 종료 (terminate)와 같은 다섯 개의 방어 유형으로 방어 동작을 수행할 수 있다.

[0057] 시스템에 결함이 발생하는 경우는 크게 입력 데이터가 유효 데이터가 아닌 경우와 시스템 상태가 불안정한 경우를 들 수 있다. 따라서, 능동 방어 루틴은 입력 데이터가 유효 데이터인지 여부 및 실행 결과가 성공인지 혹은 실패인지 여부 등에 따라 대응하는 방어 유형으로 방어를 수행할 수 있다.

[0058] 일 예로서, 능동 방어 루틴은 입력값이 유효 범위내이고, 실행결과가 성공이면 결함이 탐지되지 않은 것으로 판단하여 다음 기능이 수행되도록 한다.

[0059] 한편, 능동 방어 루틴은 입력값이 유효 범위이지만, 실행결과가 성공이 아니면 그 실패원인을 확인한다. 만일, 실패원인이 일시적인 현상으로 인한 것이면, 도 9a와 같은 반복에 해당하는 방어 액션을 수행하도록 한다.

[0060] 여기서, 반복에 해당하는 방어 액션은 프로그램의 입력 값은 유효 범위지만, 시스템의 상태에 따라 오류가 일시적으로 발생한 경우에 수행되는 액션이다. 반복 액션은 시스템의 상태가 정상으로 복귀될 때까지 동일 이벤트를 반복적으로 수행하며, 일정 횟수이상 지속적으로 실패할 경우에는 해당 어플이케이션에 '실패(fail)'를 반환한다.

[0061] 특히, 도 9a는 시스템의 일시적 메모리 부족 현상으로 인해 메모리 할당에 실패한 경우를 나타낸 것이다. 이 경우, 프로그램 입력은 '12345'로 정상이나, 일시적인 시스템 상태로 인해 문제가 발생한 것으로, 반복 액션에 의해 약속된 횟수만큼 해당 기능을 반복적으로 시도하게 된다. 즉, 시스템 상태가 일시적인 현상이었다면, 몇 번의 반복 실행으로 인해 시스템이 안정적인 동작을 보장하게 되며, 그로 인해 '12345'가 그대로 출력되게 된다.

[0062] 반면, 실패원인이 일시적인 현상이 아니면 종료에 해당하는 방어 액션을 수행하도록 한다. 종료 액션은 프로그램의 입력 값은 유효 범위지만, 시스템의 상태에 따라 오류가 발생하여 지속적으로 유지되는 경우에 수행되는 액션으로, 이벤트의 실행 결과가 시스템에 미치는 영향이 치명적인 경우 해당하는 프로세스를 종료시킨다.

[0063] 또한, 능동 방어 루틴은 입력값이 유효 범위가 아니고, 실패 원인을 예측할 수 없으면 경고에 해당하는 방어 액션을 수행하도록 한다. 경고 액션은 프로그램 입력 값이 유효한 값은 아니지만, 실패 원인을 정확히 유추할 수 없는 경우에 수행되는 액션으로, 사용자에게 해당 이벤트 실행에 문제가 있음을 보고하기 위해 해당 이벤트의 수행은 계속하면서 경고 메시지를 전달한다.

[0064] 반면, 입력값이 유효 범위가 아니고, 실패 원인을 예측할 수 있으면 입력값을 교정하는 것으로 안전한 실행을 보정할 수 있는지 여부를 판별하여 안전한 실행을 보장할 수 있으면 도 9b와 같이 지속 액션을 수행하여 다음 기능을 계속 진행하도록 한다. 여기서, 지속 액션은 해당 이벤트를 실행하지 않고도 프로그램 입력 값만으로도 실패 원인을 판단할 수 있으며 적절한 입력 데이터 값의 교정에 의해 정상적인 실행을 보장할 수 있는 경우에 수행되는 액션이다.

[0065] 특히, 도 9b는 문자열을 복사하는 함수에서 유효한 할당 범위를 넘어서는 복사를 수행하는 경우를 나타낸 것이다. 이 경우, 정보태그를 통해 데이터의 유효 접근 범위를 알 수 있기 때문에 유효 할당 범위만큼만 복사가 되도록 입력 값을 안전한 범위로 조정하여 실행을 지속할 수 있다.

[0066] 한편, 입력값이 유효 범위가 아니고, 실패 원인을 예측할 수 있는 상태에서 입력값을 교정하는 것으로 안전한 실행을 보정할 수 없는 경우에는, 도 9c와 같이 무시에 해당하는 방어 액션을 수행하도록 한다. 여기서, 무시 액션은 해당 이벤트의 실행이 문제를 일으킬 수 있으며, 다음 실행에 영향을 주지 않음을 프로그램 입력 값만으로도 판단할 수 있는 경우에 수행되는 액션으로, 해당하는 이벤트를 무시하고 바로 해당 어플리케이션에 '실패(fail)'를 반환한다.

[0067] 특히, 도 9c는 동적 할당된 포인터 변수가 두 번에 걸쳐 해제 동작이 수행된 경우를 나타낸 것으로, 두 번째 해제 동작에 대해 무시 액션을 취함으로써 정상적인 실행을 보장하게 된다.

[0068] 이상과 같이 본 발명에 의한 소프트웨어 검사 방법 및 시스템은 예시된 도면을 참조로 설명하였으나, 본 명세서에 개시된 실시예와 도면에 의해 본 발명은 한정되지 않고, 기술사상이 보호되는 범위 이내에서 응용될 수 있다.

부호의 설명

[0069] 1~10 : 어플리케이션

20: PCB(Process Control Block)

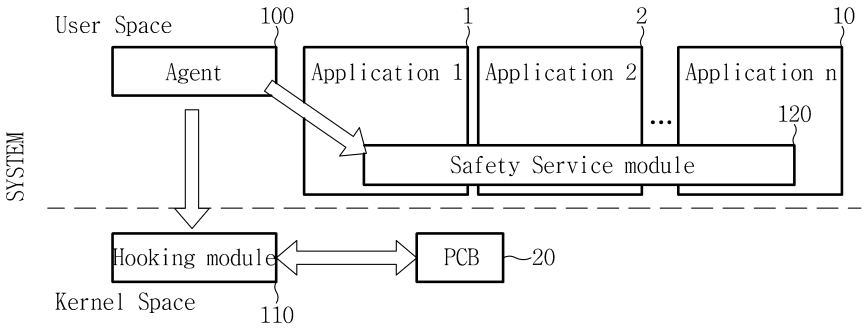
100: 에이전트

110: 후킹 모듈

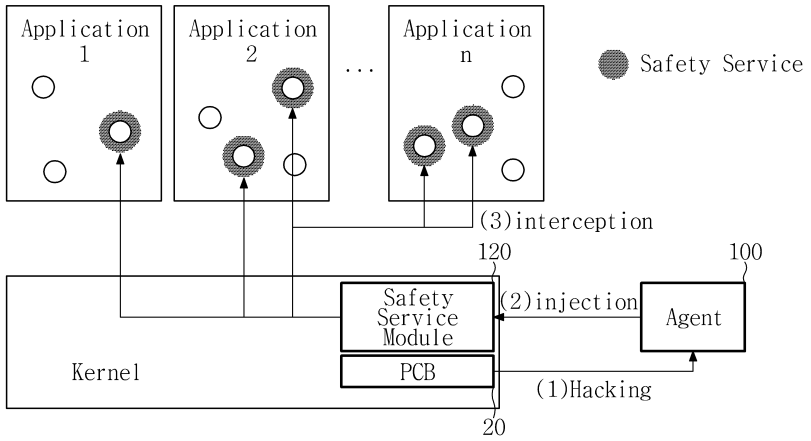
120: 세이프티 서비스 모듈

도면

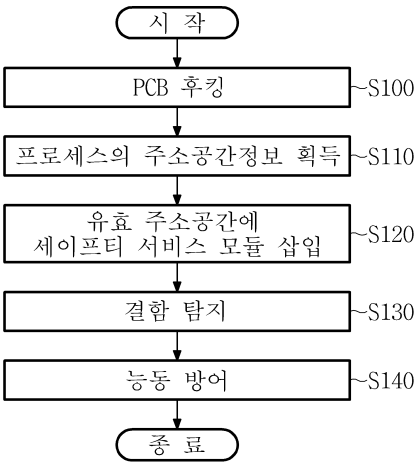
도면1



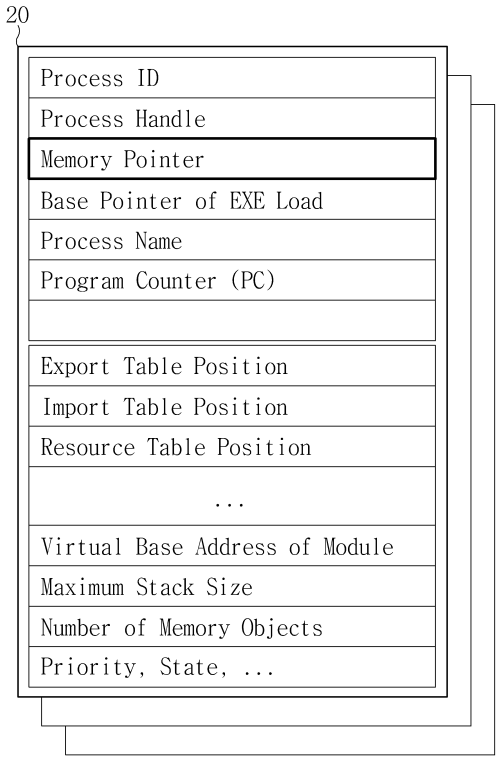
도면2



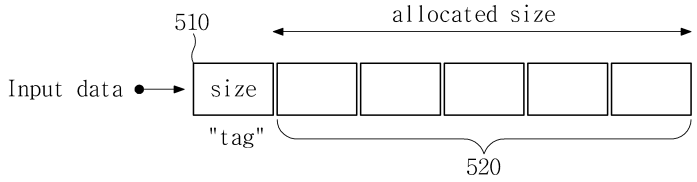
도면3



도면4



도면5



도면6

```
SafetyService ()
{
        // 1) Fault detection action
        if ( fault exist ) {
            // 2) Active defense action
        }
        // 3) Original service
}
```

도면7a

```
int   * iptr = NULL;
MemoryFree( iptr );
```

도면7b

```
int   * iptr;
MemoryFree( iptr );
```

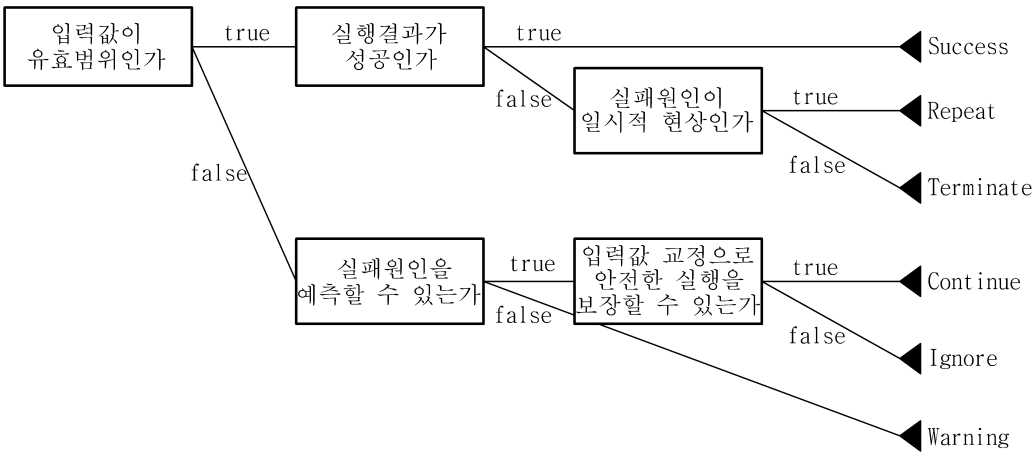
도면7c

```
void MemoryFree(void * ptr)
{
if ( ptr ) {
free ( ptr );
}
}
```

도면7d

```
void SafetyService(void * ptr)
{
bool   isValid = Check if ptr value is valid
if ( isValid )     MemoryFree( ptr);
ptr = NULL;
}
```

도면8



도면9a

char *buf = (char *)malloc(10);
strcpy(buf, "12345");
printf("%s", buf);
free(buf);
	void * SafetyService (int size)
	{
	void *ptr = NULL;
	int repeat = 0;
	do { // ***repeat action***
	ptr = malloc(size);
	} while (ptr == NULL && repeat++ < REPEAT_MAX);
	return ptr;
	}

Expected Result: print out "12345" Actual Result: Segmentation fault Cause: Memory shortage	Actual Result: print out "12345"

도면9b

char *buf = (char *)malloc(10);
strcpy(buf, "12345678901");
printf("%s", buf);
free(buf);
	char * SafetyService (char* dest, const char * src)
	{
	int src_size = Read *infoTag* field of *src*;
	int dest_size = Read *infoTag* field of *dest*;
	
	if (src_size > dest_size) src_size = dest_size; // ***correction***
	strncpy(dest, src, src_size); // ***continue action***
	return dest;
	}

Expected Result: print out "123456789" Actual Result: Aborted (core dumped) Cause: Buffer overflow	Actual Result: print out "123456789"

도면9c

char *buf = (char *)malloc(10);
strcpy(buf, "12345");
free(buf);
free(buf);
printf("%s", buf);
	void SafetyService (void* ptr)
	{
	int size = Read *infoTag* field of *ptr*;
	if (size) free(ptr);
	else ; // ***ignore action***
	ptr = NULL;
	}

Expected Result: print out *NONE* Actual Result: Aborted (core dumped) Cause: Duplicated free	Actual Result: print out *NONE*