



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2019-0078546
(43) 공개일자 2019년07월04일

(51) 국제특허분류(Int. Cl.)
G06F 21/57 (2013.01) G06F 21/56 (2013.01)
(52) CPC특허분류
G06F 21/577 (2013.01)
G06F 21/562 (2013.01)
(21) 출원번호 10-2018-0169834
(22) 출원일자 2018년12월26일
심사청구일자 2018년12월26일
(30) 우선권주장
1020170179651 2017년12월26일 대한민국(KR)

(71) 출원인
충남대학교산학협력단
대전광역시 유성구 대학로 99 (궁동, 충남대학교)
(72) 발명자
조은선
대전광역시 유성구 어은로 57, 110동 1504호
목성균
대전광역시 서구 가장로 131-2, 403호
전현구
경기도 성남시 분당구 판교역로241번길 22, 634
(74) 대리인
홍성욱, 심경식

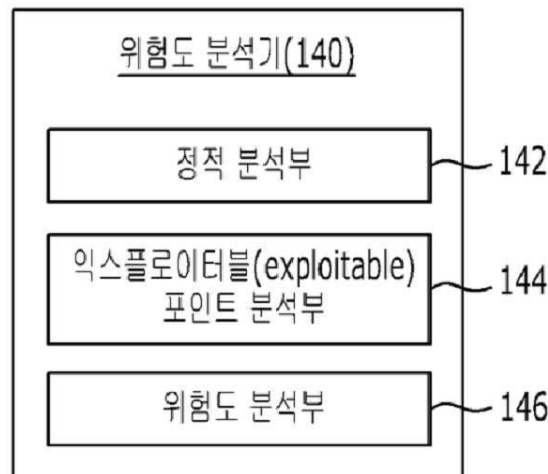
전체 청구항 수 : 총 14 항

(54) 발명의 명칭 프로그램의 크래시 위험도 분석 장치 및 그 방법

(57) 요약

본 발명은 프로그램의 크래시 위험도 분석 장치 및 그 방법에 관한 것으로, 크래시 위험도 분석 장치는 분석 대상인 바이너리(binary)의 실행을 통한 테인트 분석(taint analysis)을 수행하여, 상기 바이너리에 대하여 크래시(crash)를 발생시킨 명령어들 각각에 의해 영향을 받는 모든 명령어를 정적 분석하여 해당 크래시의 위험도를 분석하는 위험도 분석기를 포함하되, 상기 위험도 분석기는, 상기 크래시들을 발생시킨 명령어들 각각에 대하여 상기 명령어들의 도달지점들과 사용가능성(available)을 식별하고, 상기 도달지점들과 사용가능성에 기초하여 프로그램의 제어권을 옮길 수 있는 명령어를 식별하며, 상기 식별된 명령어가 저장(store) 계열 명령어이거나, 분기(branch) 계열 명령어인 경우 공격 가능성을 분석하고, 그 분석 결과에 기초하여 위험도의 단계로 분류할 수 있다.

대표도 - 도2



이 발명을 지원한 국가연구개발사업

과제고유번호 1711070437

부처명 과학기술정보통신부

연구관리전문기관 정보통신기술진흥센터

연구사업명 대학ICT연구센터육성지원사업

연구과제명 지능정보기술을활용한 금융서비스 보호 기술 연구

기 여 율 1/1

주관기관 충남대학교 산학협력단

연구기간 2018.01.01 ~ 2018.12.31

명세서

청구범위

청구항 1

분석 대상인 바이너리(binary)의 실행을 통한 테인트 분석(taint analysis)을 수행하여, 상기 바이너리에 대하여 크래시(crash)를 발생시킨 명령어들 각각에 의해 영향을 받는 모든 명령어를 정적 분석하여 해당 크래시의 위험도를 분석하는 위험도 분석기를 포함하되,

상기 위험도 분석기는,

상기 크래시들을 발생시킨 명령어들 각각에 대하여 상기 명령어들의 도달지점들과 사용가능성(available)을 식별하고, 상기 도달지점들과 사용가능성에 기초하여 프로그램의 제어권을 옮길 수 있는 명령어를 식별하며, 상기 식별된 명령어가 저장(store) 계열 명령어이거나, 분기(branch) 계열 명령어인 경우 공격 가능성을 분석하고, 그 분석 결과에 기초하여 위험도의 단계로 분류하는 것을 특징으로 하는 크래시 위험도 분석 장치.

청구항 2

제1항에 있어서,

상기 도달지점들 식별은 아래 수학적식을 이용하여, 미트 연산자(meet operator)로 합집합을 이용하여 In을 획득하는 것을 특징으로 하는 크래시 위험도 분석 장치.

[수학적식]

$$\begin{aligned} out(B) &= F_B(in(B)) \\ in(B) &= \bigcup_{Q \in \text{Predecessor}(B)} out(Q) \end{aligned}$$

여기서, B는 basic block, $out(B)$ 는 집합 out, $F_B(x)$ 는 전달함수로, $F_B(x) = gen_B \cup (x - kill_B)$ 를 의미하고, $in(B)$ 는 집합 in를 의미함.

청구항 3

제1항에 있어서,

상기 사용가능성은 아래 수학적식을 이용하여 식별하되, 미트 연산자(meet operator)로 교집합($\cap$)을 이용하여 In을 획득하는 것을 특징으로 하는 크래시 위험도 분석 장치.

[수학적식]

$$out(B) = F_B(in(B)), \quad in(B) = \bigcap_{Q \in \text{Pred}(B)} out(Q)$$

청구항 4

제1항에 있어서,

상기 위험도의 단계는,

SE(Strongly Exploitable), E(Exploitable), PE(Probably Exploitable), NE(Not Exploitable)를 포함하는 것을 특징으로 하는 크래시 위험도 분석 장치.

청구항 5

제4항에 있어서,

상기 위험도 분석기는,

상기 프로그램의 제어권을 옮길 수 있는 명령어가 분기(branch) 계열 명령어이고, 상기 분기 계열 명령어의 목적 주소(Target Address)가 상기 크래시를 발생시킨 명령어로부터 영향을 받은 경우, 해당 크래시를 공격 가능한 것으로 결정하되,

해당 명령어가 도달지점들에 기초하여 식별된 분기 계열 명령어인 경우, 위험도를 E(exploitable) 단계로 분류하고, 해당 명령어가 사용가능성에 기초하여 식별된 분기 계열 명령어인 경우 위험도를 SE(Strongly Exploitable) 단계로 분류하는 것을 특징으로 하는 크래시 위험도 분석 장치.

청구항 6

제4항에 있어서,

상기 위험도 분석기는,

상기 프로그램의 제어권을 옮길 수 있는 명령어가 저장 계열 명령어이고, 상기 저장 계열 명령어의 저장할 데이터를 가진 레지스터 또는 저장할 메모리 위치가 상기 크래시를 발생시킨 명령어로부터 영향을 받은 경우, 해당 크래시를 공격 가능한 것으로 결정하되,

해당 명령어가 도달지점들에 기초하여 식별된 저장 계열 명령어인 경우, 위험도를 E(exploitable) 또는 PE(probably exploitable)의 단계로 분류하고, 해당 명령어가 사용가능성에 기초하여 식별된 저장 계열 명령어인 경우 위험도를 SE(Strongly Exploitable) 단계로 분류하는 것을 특징으로 하는 크래시 위험도 분석 장치.

청구항 7

제6항에 있어서,

상기 위험도 분석기는,

상기 명령어가 도달지점들에 기초하여 식별된 저장 계열 명령어이고,

상기 저장 계열 명령어의 저장할 데이터를 가진 레지스터만이 상기 크래시를 발생시킨 명령어로부터 영향을 받은 경우 또는 상기 저장 계열 명령어의 저장할 메모리 위치만이 상기 크래시를 발생시킨 명령어로부터 영향을 받은 경우, 위험도를 PE(probably exploitable)의 단계로 분류하고,

상기 저장 계열 명령어의 저장할 데이터를 가진 레지스터와 저장할 메모리 위치가 상기 크래시를 발생시킨 명령어로부터 영향을 받은 경우 위험도를 E(exploitable) 단계로 분류하는 것을 특징으로 하는 크래시 위험도 분석 장치.

청구항 8

제6항에 있어서,

상기 위험도 분석기는,

상기 명령어가 사용가능성에 기초하여 식별된 저장 계열 명령어이고,

상기 저장 계열 명령어의 저장할 데이터를 가진 레지스터와 저장할 메모리 위치가 상기 크래시를 발생시킨 명령어로부터 영향을 받은 경우 위험도를 SE(Strongly Exploitable) 단계로 분류하는 것을 특징으로 하는 크래시 위험도 분석 장치.

청구항 9

분석 대상인 바이너리(binary)의 실행을 통한 테인트 분석(taint analysis)을 수행하여, 상기 바이너리에 대하여 크래시(crash)를 발생시킨 명령어들 각각에 의해 영향을 받는 모든 명령어를 정적 분석하여, 상기 크래시들을 발생시킨 명령어들 각각에 대하여 상기 명령어들의 도달지점들과 사용가능성(available)을 식별하는 정적 분석부;

상기 식별된 도달지점들과 사용가능성에 기초하여 프로그램의 제어권을 옮길 수 있는 명령어를 식별하는 익스플로이터블 포인트 분석부; 및

상기 식별된 명령어가 저장(store) 계열 명령어이거나, 분기(branch) 계열 명령어인 경우 공격 가능성을 분석하고, 그 분석 결과에 기초하여 위험도의 단계로 분류하는 위험도 분석부를 포함하는 크래시 위험도 분석 장치.

청구항 10

제9항에 있어서,

상기 크래시들을 발생시킨 명령어들의 도달지점들을 식별하는 리칭-데프(Reaching Definition) 분석모듈;

크래시들을 발생시킨 명령어들의 사용가능성(available)을 식별하는 사용 가능성-데프(Available Definition) 분석모듈; 및

메모리를 레지스터처럼 보고 kill하는 메모리 사용가능성-데프 분석모듈을 포함하는 것을 특징으로 하는 크래시 위험도 분석 장치.

청구항 11

제10항에 있어서,

상기 리칭-데프 분석모듈은,

아래 수학적식을 이용하여 도달지점들을 식별하되, 미트 연산자(meet operator)로 합집합을 이용하여 In을 획득하는 것을 특징으로 하는 크래시 위험도 분석 장치.

[수학적식]

$$\begin{aligned} out(B) &= F_B(in(B)) \\ in(B) &= \bigcup_{Q \in \text{Predecessor}(B)} out(Q) \end{aligned}$$

여기서, B는 basic block, $out(B)$ 는 집합 out, $F_B(x)$ 는 전달함수로, $F_B(x) = gen_B \cup (x - kill_B)$ 를 의미하고, $in(B)$ 는 집합 in를 의미함.

청구항 12

제10항에 있어서,

상기 사용 가능성-데프 분석모듈은,

아래 수학적식을 이용하여 상기 사용가능성을 식별하되, 미트 연산자(meet operator)로 교집합($\cap$)을 이용하여 In을 획득하는 것을 특징으로 하는 크래시 위험도 분석 장치.

[수학적식]

$$out(B) = F_B(in(B)), \quad in(B) = \bigcap_{Q \in \text{Pred}(B)} out(Q)$$

청구항 13

제10항에 있어서,

상기 위험도 분석부는,

SE(Strongly Exploitable), E(Exploitable), PE(Probably Exploitable), NE(Not Exploitable) 중 적어도 하나로 위험도의 단계를 분류하는 것을 특징으로 하는 크래시 위험도 분석 장치.

청구항 14

크래시 위험도 분석 장치가 크래시 위험도를 분석하는 방법에 있어서,

분석 대상인 바이너리(binary)의 실행을 통한 테인트 분석(taint analysis)을 수행하여, 상기 바이너리에 대하여 크래시들을 발생시킨 명령어들 각각에 대하여 도달지점들과 사용가능성(available)을 식별하는 단계;

상기 도달지점들과 사용가능성에 기초하여 프로그램의 제어권을 옮길 수 있는 명령어를 식별하는 단계;

상기 식별된 명령어가 저장(store) 계열 명령어이거나, 분기(branch) 계열 명령어인 경우 공격 가능성을 분석하고, 그 분석 결과에 기초하여 위험도의 단계로 분류하는 단계

를 포함하는 크래시 위험도 분석 방법.

발명의 설명

기술 분야

- [0001] 본 발명은 프로그램의 크래시 위험도 분석 장치 및 그 방법에 관한 것으로, 특히 단계별 정적 분석을 통해 보다 정교하게 크래시 위험도를 분류하는 프로그램의 크래시 위험도 분석 장치 및 그 방법에 관한 것이다.

배경 기술

- [0003] 바이너리 프로그램 분석은 소스 코드에 의존하지 않고 프로그램의 구조와 실행 흐름 등을 파악하는데 매우 중요하다. 하지만 다른 모든 분석들과 마찬가지로, 정확한 프로그램 분석에는 프로그램의 크기에 비례하여 많은 시간이 걸리게 되고, 시간이 적게 소요되는 간단한 분석 방법으로는 그 적용 범위가 매우 작거나 잘못된 분석 결과를 도출할 확률이 높아진다. 따라서 바이너리 프로그램을 효율적으로 분석하려는 연구가 필요하다.
- [0004] 바이너리 정적 분석은 취약점 분석, 악성코드 분석, 표절 탐지 등에서 사용되고 있다. 이러한 바이너리 정적 분석 시, 크래시가 발생할 수 있다. 크래시란 프로그램을 실행할 때 예외처리가 되지 않아 프로그램이 종료되는 예외 상황을 의미하며, 이는 개발자가 의도치 않은 실행 흐름으로 이를 악성 코드를 실행시키는 등 악용할 소지가 있어 반드시 수정해야 한다.
- [0005] 이때 크래시를 수동으로 분석하는 것은 매우 시간이 오래 걸리고 힘들기 때문에 자동화된 도구를 사용하는 것이 불가피하다.
- [0006] 이에, 바이너리 프로그램의 크래시를 자동으로 분석할 수 있는 기술 개발이 요구되고 있다.

선행기술문헌

특허문헌

- [0008] (특허문헌 0001) 한국 등록특허공보 제10-1482073호(2006.04.24.)

발명의 내용

해결하려는 과제

- [0009] 본 발명이 해결하고자 하는 과제는 바이너리 파일에 대한 크래시 위험도를 자동으로 단계별로 분석할 수 있는 프로그램의 크래시 위험도 분석 장치 및 그 방법을 제공하는 것이다.
- [0010] 본 발명이 해결하고자 하는 과제는 이상에서 언급한 과제(들)로 제한되지 않으며, 언급되지 않은 또 다른 과제(들)은 아래의 기재로부터 당업자에게 명확하게 이해될 수 있을 것이다.

과제의 해결 수단

- [0012] 상기한 과제를 해결하기 위하여, 본 발명의 일 실시예에 따른 크래시 위험도 분석 장치는, 분석 대상인 바이너리(binary)의 실행을 통한 테인트 분석(taint analysis)을 수행하여, 상기 바이너리에 대하여 크래시(crash)를

발생시킨 명령어들 각각에 의해 영향을 받는 모든 명령어를 정적 분석하여 해당 크래시의 위험도를 분석하는 위험도 분석기를 포함하되, 상기 위험도 분석기는, 상기 크래시들을 발생시킨 명령어들 각각에 대하여 상기 명령어들의 도달지점들과 사용가능성(available)을 식별하고, 상기 도달지점들과 사용가능성에 기초하여 프로그램의 제어권을 옮길 수 있는 명령어를 식별하며, 상기 식별된 명령어가 저장(store) 계열 명령어이거나, 분기(branch) 계열 명령어인 경우 공격 가능성을 분석하고, 그 분석 결과에 기초하여 위험도의 단계로 분류하는 것을 특징으로 한다.

[0013] 바람직하게는, 상기 도달지점들 식별은 아래 수학적식을 이용하되, 미트 연산자(meet operator)로 합집합을 이용하여 In을 획득할 수 있다.

[0014] [수학적식]

$$\begin{aligned} out(B) &= F_B(in(B)) \\ in(B) &= \bigcup_{Q \in \text{Predecessor}(B)} out(Q) \end{aligned}$$

[0015]

[0016] 여기서, B는 basic block, $out(B)$ 는 집합 out, $F_B(x)$ 는 전달함수로, $F_B^*(x) = gen_B \cup (x - kill_B)$ 를 의미하고, $in(B)$ 는 집합 in를 의미함.

[0017] 바람직하게는, 상기 사용가능성은 아래 수학적식을 이용하여 식별하되, 미트 연산자(meet operator)로 교집합($\cap$)을 이용하여 In을 획득할 수 있다.

[0018] [수학적식]

$$out(B) = F_B(in(B)), \quad in(B) = \bigcap_{Q \in \text{Pred}(B)} out(Q)$$

[0019]

[0020] 바람직하게는, 상기 위험도의 단계는, SE(Strongly Exploitable), E(Exploitable), PE(Probably Exploitable), NE(Not Exploitable)를 포함할 수 있다.

[0021] 바람직하게는, 상기 크래시 위험도 분석기는, 상기 프로그램의 제어권을 옮길 수 있는 명령어가 분기(branch) 계열 명령어이고, 상기 분기 계열 명령어의 목적 주소(Target Address)가 상기 크래시를 발생시킨 명령어로부터 영향을 받은 경우, 해당 크래시를 공격 가능한 것으로 결정하되, 해당 명령어가 도달지점들에 기초하여 식별된 분기 계열 명령어인 경우, 위험도를 E(exploitable) 단계로 분류하고, 해당 명령어가 사용가능성에 기초하여 식별된 분기 계열 명령어인 경우 위험도를 SE(Strongly Exploitable) 단계로 분류할 수 있다.

[0022] 바람직하게는, 상기 위험도 분석기는, 상기 프로그램의 제어권을 옮길 수 있는 명령어가 저장 계열 명령어이고, 상기 저장 계열 명령어의 저장할 데이터를 가진 레지스터 또는 저장할 메모리 위치가 상기 크래시를 발생시킨 명령어로부터 영향을 받은 경우, 해당 크래시를 공격 가능한 것으로 결정하되, 해당 명령어가 도달지점들에 기초하여 식별된 저장 계열 명령어인 경우, 위험도를 E(exploitable) 또는 PE(probably exploitable)의 단계로 분류하고, 해당 명령어가 사용가능성에 기초하여 식별된 저장 계열 명령어인 경우 위험도를 SE(Strongly Exploitable) 단계로 분류할 수 있다.

[0023] 바람직하게는, 상기 위험도 분석기는, 상기 명령어가 도달지점들에 기초하여 식별된 저장 계열 명령어이고, 상기 저장 계열 명령어의 저장할 데이터를 가진 레지스터만이 상기 크래시를 발생시킨 명령어로부터 영향을 받은 경우 또는 상기 저장 계열 명령어의 저장할 메모리 위치만이 상기 크래시를 발생시킨 명령어로부터 영향을 받은 경우, 위험도를 PE(probably exploitable)의 단계로 분류하고, 상기 저장 계열 명령어의 저장할 데이터를 가진 레지스터와 저장할 메모리 위치가 상기 크래시를 발생시킨 명령어로부터 영향을 받은 경우 위험도를 E(exploitable) 단계로 분류할 수 있다.

[0024] 바람직하게는, 상기 위험도 분석기는, 상기 명령어가 사용가능성에 기초하여 식별된 저장 계열 명령어이고, 상기 저장 계열 명령어의 저장할 데이터를 가진 레지스터와 저장할 메모리 위치가 상기 크래시를 발생시킨 명령어로부터 영향을 받은 경우 위험도를 SE(Strongly Exploitable) 단계로 분류할 수 있다.

[0025] 상기한 과제를 해결하기 위하여, 본 발명의 다른 실시예에 따른 분석 대상인 바이너리(binary)의 실행을 통한 테인트 분석(taint analysis)을 수행하여, 상기 바이너리에 대하여 크래시(crash)를 발생시킨 명령어들 각각에

의해 영향을 받는 모든 명령어를 정적 분석하여, 상기 크래시들을 발생시킨 명령어들 각각에 대하여 상기 명령어들의 도달지점들과 사용가능성(available)을 식별하는 정적 분석부, 상기 식별된 도달지점들과 사용가능성에 기초하여 프로그램의 제어권을 옮길 수 있는 명령어를 식별하는 익스플로이터블 포인트 분석부, 상기 식별된 명령어가 저장(store) 계열 명령어이거나, 분기(branch) 계열 명령어인 경우 공격 가능성을 분석하고, 그 분석 결과에 기초하여 위험도의 단계로 분류하는 위험도 분석부를 포함한다.

[0026] 바람직하게는, 상기 크래시들을 발생시킨 명령어들의 도달지점들을 식별하는 리칭-데프(Reaching Definition) 분석모듈, 크래시들을 발생시킨 명령어들의 사용가능성(available)을 식별하는 사용 가능성-데프(Available Definition) 분석모듈, 메모리를 레지스터처럼 보고 kill하는 메모리 사용가능성-데프 분석모듈을 포함할 수 있다.

[0027] 바람직하게는, 상기 리칭-데프 분석모듈은, 아래 수학적식을 이용하여 도달지점들을 식별하되, 미트 연산자(meet operator)로 합집합을 이용하여 In을 획득할 수 있다.

[0028] [수학식]

$$\begin{aligned} out(B) &= F_B(in(B)) \\ in(B) &= \bigcup_{Q \in \text{Predecessor}(B)} out(Q) \end{aligned}$$

[0029]

[0030] 여기서, B는 basic block, $out(B)$ 는 집합 out, $F_B(x)$ 는 전달함수로, $F_B^*(x) = gen_B \cup (x - kill_B)$ 를 의미하고, $in(B)$ 는 집합 in를 의미함.

[0031] 바람직하게는, 상기 사용 가능성-데프 분석모듈은, 아래 수학적식을 이용하여 상기 사용가능성을 식별하되, 미트 연산자(meet operator)로 교집합($\cap$)을 이용하여 In을 획득할 수 있다.

[0032] [수학식]

$$out(B) = F_B(in(B)), \quad in(B) = \bigcap_{Q \in \text{Pred}(B)} out(Q)$$

[0033]

[0034] 바람직하게는, 상기 위험도 분석부는, SE(Strongly Exploitable), E(Exploitable), PE(Probably Exploitable), NE(Not Exploitable) 중 적어도 하나로 위험도의 단계를 분류할 수 있다.

[0035] 상기한 과제를 해결하기 위하여, 본 발명의 또 다른 실시예에 따른 크래시 위험도 분석 장치가 크래시 위험도를 분석하는 방법에 있어서, 분석 대상인 바이너리(binary)의 실행을 통한 테인트 분석(taint analysis)을 수행하여, 상기 바이너리에 대하여 크래시들을 발생시킨 명령어들 각각에 대하여 도달지점들과 사용가능성(available)을 식별하는 단계, 상기 도달지점들과 사용가능성에 기초하여 프로그램의 제어권을 옮길 수 있는 명령어를 식별하는 단계, 상기 식별된 명령어가 저장(store) 계열 명령어이거나, 분기(branch) 계열 명령어인 경우 공격 가능성을 분석하고, 그 분석 결과에 기초하여 위험도의 단계로 분류하는 단계를 포함한다.

발명의 효과

[0037] 본 발명에 따르면, 바이너리 파일과 크래시 정보로 정적 분석을 실시하고 크래시의 위험도를 자동으로 판단하여 사용자에게 보고함으로써, 비교적 더 심각한 문제를 초래할 수 있는 크래시를 우선적으로 보완하거나 공격가능성이 없는 크래시는 수정범위에서 제외할 수 있을 뿐만 아니라 공격가능성에 대한 정보로 보완에 도움을 줄 수 있다.

[0038] 또한, 본 발명에 따르면, 크래시의 위험도를 5가지로 분류함으로써 사용자가 크래시의 위험도를 보다 정밀하게 알 수 있다.

[0039] 또한, 본 발명에 따르면, 크래시를 3단계의 정적 분석을 함으로써, 크래시의 위험도를 여러 단계로 나누어 사용자에게 알려줄 수 있고, 이는 사용자가 보다 확실하게 크래시의 위험도를 알 수 있게 하고 프로그램의 취약점의 위험도를 알게 함으로써 패치의 순서를 알 수 있다.

[0040] 본 발명의 효과들은 이상에서 언급한 효과들로 제한되지 않으며, 언급되지 않은 또 다른 효과들은 아래의 기재로부터 통상의 기술자에게 명확하게 이해될 수 있을 것이다.

도면의 간단한 설명

[0042] 도 1은 본 발명의 일 실시 예에 따른 크래시 위험도 분석 장치에 대한 개략적인 블록도이다.
 도 2는 본 발명의 일 실시예에 따른 위험도 분석기를 설명하기 위한 블록도이다.
 도 3은 도 2에 도시된 정적 분석부의 세부 구성을 설명하기 위한 블록도이다.
 도 4는 본 발명의 일 실시예에 따른 Reaching Definition와 Avaliable Definition를 설명하기 위한 예시도이다.
 도 5는 본 발명의 일 실시예에 따른 Reaching Definition, Avaliable Definition, Menory Avaliable Definition의 특징을 설명하기 위한 도면이다.
 도 6은 본 발명의 일 실시예에 따른 위험도 분류의 기준을 설명하기 위한 도면이다.
 도 7은 본 발명의 일 실시 예에 따른 크래시 위험도 분석 방법에 대한 개략적인 처리 흐름도이다.
 도 8은 본 발명의 일 실시예에 따른 위험도의 단계를 판단하는 방법을 설명하기 위한 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0043] 본 발명은 다양한 변경을 가할 수 있고 여러 가지 실시예를 가질 수 있는 바, 특정 실시예들을 도면에 예시하고 상세하게 설명하고자 한다. 그러나 이는 본 발명을 특정한 실시 형태에 한정하려는 것이 아니며, 본 발명의 사상 및 기술 범위에 포함되는 모든 변경, 균등물 내지 대체물을 포함하는 것으로 이해되어야 한다. 각 도면을 설명하면서 유사한 참조부호를 유사한 구성요소에 대해 사용하였다.

[0044] 제1, 제2, A, B 등의 용어는 다양한 구성요소들을 설명하는데 사용될 수 있지만, 상기 구성요소들은 상기 용어들에 의해 한정되어서는 안 된다. 상기 용어들은 하나의 구성요소를 다른 구성요소로부터 구별하는 목적으로만 사용된다. 예를 들어, 본 발명의 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.

[0045] 어떤 구성요소가 다른 구성요소에 "연결되어" 있다거나 "접속되어" 있다고 언급된 때에는, 그 다른 구성요소에 직접적으로 연결되어 있거나 또는 접속되어 있을 수도 있지만, 중간에 다른 구성요소가 존재할 수도 있다고 이해되어야 할 것이다. 반면에, 어떤 구성요소가 다른 구성요소에 "직접 연결되어" 있다거나 "직접 접속되어" 있다고 언급된 때에는, 중간에 다른 구성요소가 존재하지 않는 것으로 이해되어야 할 것이다.

[0046] 본 출원에서 사용한 용어는 단지 특정한 실시예를 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.

[0047] 다르게 정의되지 않는 한, 기술적이거나 과학적인 용어를 포함해서 여기서 사용되는 모든 용어들은 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가지고 있다. 일반적으로 사용되는 사전에 정의되어 있는 것과 같은 용어들은 관련 기술의 문맥상 가지는 의미와 일치하는 의미를 가지는 것으로 해석되어야 하며, 본 출원에서 명백하게 정의하지 않는 한, 이상적이거나 과도하게 형식적인 의미로 해석되지 않는다.

[0048] 이하에서는 본 발명에 따른 바람직한 실시예를 첨부된 도면을 참조하여 상세하게 설명한다.

[0049] 도 1은 본 발명의 일 실시 예에 따른 크래시 위험도 분석 장치에 대한 개략적인 블록도이다.

[0050] 도 1을 참조하면, 본 발명의 일 실시 예에 따른 크래시 위험도 분석 장치(100)는 디스어셈블러(110), 중간 언어 변환기(120), 크래시 발생기(130) 및 크래시 위험도 분석기(140)를 포함한다.

- [0051] 디스어셈블러(Disassembler)(110)는 입력 데이터를 기계어로 변환한다. 특히, 디스어셈블러(110)는 크래시 위험도 분석 대상인 바이너리 프로그램(binary program)(예컨대, ARM 바이너리 프로그램)을 디스어셈블링(disassembling)하여 기계어로 변환하고, 상기 기계어로부터 획득된 제어 흐름 정보(Control Flow Graph)를 출력한다.
- [0052] 중간 언어 변환기(Abstract Interpretation Framework)(120)는 디스어셈블러(110)에서 출력되는 기계어를 중간 언어(Intermediate Language)로 변환한다. 예를 들어, 중간 언어 변환기(120)는 상기 기계어를 REIL(Reverse Engineering Intermediate Language)라는 중간 언어로 변환한다. 이때, 기계어를 중간 언어로 변환하는 것은 과다하게 많은 명령어의 수를 줄여서 정적 분석의 편의를 제공하기 위함이다. 예를 들어, 약 300개의 ARM 명령어가 입력된 경우 이를 REIL 이라는 중간 언어로 변환하여 사용할 경우 17개로 줄일 수 있다.
- [0053] 크래시 발생기(130)는 고의로 크래시를 유발시킨다. 이를 위해, 크래시 발생기(130)는 다수의 크래시를 고의로 발생시키기 위해 사전에 제작된 크래시 유발 프로그램을 내장하고, 그 크래시 유발 프로그램(예컨대, Fuzzer 프로그램)에 의해 다수의 크래시들을 발생시킨다. 이는 일반적으로 해커들이 퍼저(Fuzzer)와 같은 크래시 유발 프로그램을 이용하여 대상 프로그램에 많은 크래시(crash)를 발생시키고, 그 크래시들을 직접 하나씩 분석한 후 공격 가능성이 있는 경우에 해당 취약점에 맞는 공격 코드를 작성해서 공격하는 공격 패턴을 역 이용하기 위한 것이다. 즉, 상기와 같이 고의로 발생된 크래시들 각각에 대한 공격 가능성(즉, 위험도)을 자동으로 분석하기 위함이다.
- [0054] 크래시 위험도 분석기(140)는 중간언어 변환기(120)에서 중간 언어로 변환된 명령어들과, 크래시 발생기(130)에서 발생된 크래시 정보에 기초하여 해당 크래시의 위험도를 분석한다. 즉, 크래시 위험도 분석기(140)는 중간 언어로 변환된 명령어들 중 다수의 크래시들을 발생시킨 명령어들 각각에 의해 영향을 받는 모든 명령어를 정적으로 분석하여 해당 크래시의 위험도를 분석한다. 이를 위해, 위험도 분석기(140)는 다수의 크래시들을 발생시킨 명령어들 각각에 대하여 상기 명령어들의 도달지점(Reaching Definition)들을 식별하고, 그 도달지점들 중 해당 명령어가 실제로 사용되는 지점(Def-Use Chaining)을 찾아낸 후, 그 결과를 사용 지점 그래프로 도출한다. 또한, 위험도 분석기(140)는 다수의 크래시들을 발생시킨 명령어들 각각에 대하여 상기 명령어들의 사용가능성(Avaliable Definition)을 식별하고, 그 사용가능지점들 중 해당 명령어가 실제로 사용되는 지점(Def-Use Chaining)을 찾아낸 후, 그 결과를 사용 지점 그래프로 도출한다.
- [0055] 그런 후, 위험도 분석기(140)는 상기 도달지점들과 사용가능성에 기초하여 즉, 사용 지점 그래프로부터 프로그램의 제어권을 옮길 수 있는 명령어를 식별하고, 그 명령어에 대한 공격 가능성을 분석한다. 이때, 위험도 분석기(140)는 크래시를 발생시킨 명령어가 저장(store) 계열 명령어이거나, 분기(branch) 계열 명령어인 경우 공격 가능성을 분석하고, 그 분석 결과에 기초하여 위험도의 단계로 분류한다. 이때, 위험도 분석기는 SE(Strongly Exploitable), E(Exploitable), PE(Probably Exploitable), NE(Not Exploitable)로 위험도의 단계를 분류한다.
- [0056] 이러한 위험도 분석기(140)에 대한 상세한 설명은 도 2를 참조하기로 한다.
- [0057] 도 2는 본 발명의 일 실시예에 따른 위험도 분석기를 설명하기 위한 블록도, 도 3은 도 2에 도시된 정적 분석부의 세부 구성을 설명하기 위한 블록도, 도 4는 본 발명의 일 실시예에 따른 Reaching Definition와 Avaliable Definition를 설명하기 위한 예시도, 도 5는 본 발명의 일 실시예에 따른 Reaching Definition, Avaliable Definition, Memory Avaliable Definition의 특징을 설명하기 위한 도면, 도 6은 본 발명의 일 실시예에 따른 위험도 분류의 기준을 설명하기 위한 도면이다.
- [0058] 도 2를 참조하면, 본 발명의 일 실시예에 따른 위험도 분석기는 정적 분석부(142), 익스플로이터블 포인트(Exploitable Point) 분석부(144), 위험도 분석부(146)를 포함한다.
- [0059] 정적 분석부(142)는 분석 대상인 바이너리(binary)의 실행을 통한 테인트 분석(taint analysis)을 수행하여, 상기 바이너리에 대하여 크래시(crash)를 발생시킨 명령어들 각각에 의해 영향을 받는 모든 명령어를 정적 분석한다. 이때, 정적 분석부(142)는 바이너리 프로그램을 Reaching Definition, Avaliable Definition, Memory Avaliable Definition의 3단계의 정적 분석을 수행한다.
- [0060] 따라서, 정적 분석부(142)는 도 3에 도시된 바와 같이 리칭-데프(Reaching Definition) 분석모듈(142a), 사용가능성-데프(Avaliable Definition) 분석모듈(142b), 메모리 사용가능성-데프(Memory Avaliable Definition) 분석모듈(142c)을 포함한다.
- [0061] 리칭-데프 분석모듈(142a)은 크래시들을 발생시킨 명령어들의 도달지점들을 식별한다.

[0062] Reaching-Definition 분석은 변수를 정의하는 명령어가 어디까지 영향을 줄 수 있는가, 즉 어디까지 도달하는가를 알아내는 것이 목적이다. Reaching-Definition 분석의 도메인은 레지스터를 새롭게 갱신하는 명령어의 집합이며 이 분석을 수행하기 위해서는 두 개의 집합 gen과 kill이 필요하다. 집합 gen은 해당 basic block에서 레지스터를 새롭게 정의하는 명령어의 모임이며, 집합 kill은 해당 basic block에서 정의된 레지스터를 정의하는 다른 모든 명령어의 모임이며 이때 같은 basic block내에서 동일한 레지스터를 정의하는 명령어가 있으면 수행 순서에 따라 하위명령어를 기준으로 계산한다. 집합 in은 레지스터를 갱신하는 명령어 중에서 해당 basic block의 명령어들을 수행하기 시작하는 시점까지 다른 명령어에 의해 영향을 받지 않고 도달한 명령어의 모임이며, 집합 out은 해당 basic block의 명령어들을 모두 수행한 시점까지 도달한 갱신 명령어의 모임이다. 집합 in과 out을 구하기 위해서는 각 basic block의 부모 basic block의 집합 out을 으로 초기화하고 아래 수학적 1을 반복적으로 적용한다.

[0063] [수학적 1]

[0064]

$$\text{Transfer function: } F_B(x) = \text{gen}_B \cup (x - \text{kill}_B)$$

$$\text{Equation} \begin{cases} \text{out}(B) = F_B(\text{in}(B)) \\ \text{in}(B) = \bigcup_{Q \in \text{Predecessor}(B)} \text{out}(Q) \end{cases}$$

[0065] 여기서, B는 basic block, $\text{out}(B)$ 는 집합 out, $F_B(x)$ 는 전달함수로, $F_B(x) = \text{gen}_B \cup (x - \text{kill}_B)$ 를 의미하고, $\text{in}(B)$ 는 집합 in를 의미할 수 있다.

[0066] 수학적 1과 같이 Reaching Definition는 전달함수와 방정식으로 기반으로 만들어진 분석 알고리즘일 수 있다. 여기서 kill은 각 명령어마다 고정되는 상수집합이다. 그래서 수학적 1의 분석을 시작하기 전에 각 명령어에 대한 집합 kill이 계산되어 있어야 한다. 하지만 바이너리를 대상으로 이 집합 kill을 구할 때에는 메모리에 관하여 고려해야 할 사항이 있다. 바이너리에서는 메모리를 접근할 때 레지스터를 가지고 간접적으로 접근하던지, 상수값을 가지고 직접적으로 접근한다.

[0067] 사용 가능성-테프 분석모듈(142b)은 크래시들을 발생시킨 명령어들의 사용가능성(available)을 식별한다.

[0068] Available Definition 분석은 Reaching definition과 유사하나, 집합 in과 out을 구하기 위해서는 각 basic block의 부모 basic block의 집합 out을 Universe로 초기화하고, 아래 수학적 2를 반복적으로 적용한다.

[0069] [수학적 2]

[0070]

$$\text{out}(B) = F_B(\text{in}(B)), \text{ in}(B) = \bigcap_{Q \in \text{Pred}(B)} \text{out}(Q)$$

[0071] Available Definition 분석은 Reaching definition 분석에서 미트 연산자(meet operator)로 사용한 합집합 대신, 교집합($\cap$)을 이용하여 In을 얻게 된다.

[0072] 이러한 Available Definition 분석에 따라 전반적인 분석 결과로 오염된 데이터가 프로그램의 경로에 관계없이 확실하게 전파되는지 확인할 수 있다.

[0073] Reaching-Definition과 Available Definition에 대해 도 4의 예시를 참조하여 설명하기로 한다.

[0074] 도 4의 (a)의 코드를 CFG로 바꾸면 (b)와 같을 수 있다. (b)에 대해 Reaching Definition을 수행하면 (c)와 같을 수 있다. (c)를 참조하면, 3번 베이직 블록에서 변수 b가 null 로 새로운 값이 정의되지만, Exit 베이직 블록에서는 2와 3의 베이직 블록의 out이 합집합된 결과가 In으로 들어오게 된다. 즉, 1에서 정의된 b의 정의가 Exit 베이직 블록에서도 유효하다고 보게 된다.

[0075] 따라서, Reaching-Definition 분석은 프로그램의 path에 따라 유효하지 않은 정의가 도달 가능하다고 분석한 결과가 나올 수 있다.

[0076] (b)에 대해 Available definition을 수행하면 (d)와 같을 수 있다. Reaching definition과는 반대로 Exit 베이직 블록에서 2와 3의 베이직 블록의 out을 교집합으로 In을 얻게 된다. 따라서 path에 상관없이 명확한

(explicit)한 데이터의 전파만을 보기 때문에 한 단계 더 높은 수준의 경고를 할 수 있게 함을 알 수 있다.

[0077] 메모리 사용가능성-테프 분석모듈(142c)은 메모리를 레지스터처럼 보고 kill한다. 리칭-테프 분석모듈(142a)과 사용 가능성-테프 분석모듈(142b)은 메모리에 대해서 kill하지 않으므로, 메모리를 kill할 필요가 있다. 이에 메모리 사용가능성-테프 분석모듈(142c)은 메모리를 레지스터처럼 보고 kill한다.

[0078] Available Definition은 레지스터에 대해서만 kill을 하게 된다. 두 개의 레지스터가 하나의 메모리 공간을 가르켜도 다른 레지스터이기 때문에 서로 kill 하지 않는다. 이렇게 kill 되지 않은 레지스터는 후에 분석의 부정확함을 초래할 수 있다. 하지만 Memory Available Definition은 레지스터가 메모리를 가르킬 경우 같은 메모리를 가르키는 레지스터 또한 kill 한다. 앞서 언급한 예처럼 두 개의 레지스터가 하나의 메모리 공간을 가르키고 하나의 레지스터를 통해 그 메모리 공간에 새로운 값을 쓸 경우, 다른 레지스터 또한 kill 하게 된다.

[0079] 상술한 Reaching-Definition, Available Definition, Memory Available Definition 각각의 특징은 도 5에 도시된 바와 같을 수 있다.

[0080] 다시 도 2를 참조하면, 익스플로이터블 포인터 분석부(144)는 정적 분석부(142)에서 식별된 도달지점들과 사용가능성에 기초하여 프로그램의 제어권을 옮길 수 있는 명령어를 식별한다. 즉, 익스플로이터블 포인터 분석부(144)는 도달지점들 및 사용가능성들 중 당 명령어가 실제로 사용되는 지점을 찾아낸 후, 그 결과를 사용지점 그래프로 도출한다. 그런 후, 익스플로이터블 포인터 분석부(144)는 사용지점 그래프로부터 프로그램의 제어권을 옮길 수 있는 명령어를 식별한다.

[0081] 위험도 분석부(146)는 익스플로이터블 포인터 분석부(144)에서 식별된 크래시를 발생시킨 명령어가 저장(store) 계열 명령어이거나, 분기(branch) 계열 명령어인 경우 공격 가능성을 분석하고, 그 분석 결과에 기초하여 위험도의 단계로 분류한다. 이때, 위험도 분석부(146)는 도 6에 도시된 기준에 따라 SE(Strongly Exploitable), E(Exploitable), PE(Probably Exploitable), NE(Not Exploitable)로 위험도의 단계를 분류한다. 분기 계열의 명령어는 말 그대로 실행할 명령어를 가리키고 있는 레지스터의 값을 변경하여 실행 흐름을 변경할 수 있기 때문에 제어권을 옮길 수 있다. 만약 이때 제어권을 옮기려는 목적 주소(Target Address)가 Crash Point로부터 영향을 받았다면 해당 Crash는 공격 가능하다고 할 수 있다. 여기서, 크래시 포인트는 Crash를 유발시킨 명령어를 의미할 수 있다. 그리고 저장 계열 명령어는 직접 실행할 명령어를 가리키고 있는 레지스터의 값을 변경할 수는 없지만, 함수의 Return 주소나 IAT(Import Address Table), EAT(Export Address Table)에 있는 함수의 주소 등을 변경하여 제어권을 옮길 수 있다. 이러한 공격 가능성을 가진 명령어는 아래 표 1과 같을 수 있다.

[0082] [표 1]

제어권 이동가능 명령어 계열	제어권 이동가능 명령어 종류
Branch 계열	함수콜
	분기
	조건분기
Store 계열	스토어

[0083]

[0085] 구체적으로, 위험도 분석부(146)는 프로그램의 제어권을 옮길 수 있는 명령어가 분기(branch) 계열 명령어이고, 분기 계열 명령어의 목적 주소(Target Address)가 크래시를 발생시킨 명령어로부터 영향을 받은 경우, 해당 크래시를 공격 가능한 것으로 결정한다. 이때, 위험도 분석부(146)는 해당 명령어가 도달지점들에 기초하여 식별된 분기 계열 명령어인 경우, 위험도를 E(exploitable) 단계로 분류하고, 해당 명령어가 사용가능성에 기초하여 식별된 분기 계열 명령어인 경우 위험도를 SE(Strongly Exploitable) 단계로 분류한다.

[0086] 또한, 위험도 분석부(146)는 프로그램의 제어권을 옮길 수 있는 명령어가 저장 계열 명령어이고, 저장 계열 명령어의 저장할 데이터를 가진 레지스터 또는 저장할 메모리 위치가 크래시를 발생시킨 명령어로부터 영향을 받은 경우, 해당 크래시를 공격 가능한 것으로 결정한다. 이때, 위험도 분석부(146)는 해당 명령어가 도달지점들에 기초하여 식별된 저장 계열 명령어인 경우, 위험도를 E(exploitable) 또는 PE(probably exploitable)의 단계로 분류하고, 해당 명령어가 사용가능성에 기초하여 식별된 저장 계열 명령어인 경우 위험도를 SE(Strongly

Exploitable) 단계로 분류한다. 즉, 위험도 분석부(146)는 명령어가 도달지점들에 기초하여 식별된 저장 계열 명령어이고, 저장 계열 명령어의 저장할 데이터를 가진 레지스터만이 크래시를 발생시킨 명령어로부터 영향을 받은 경우 또는 저장 계열 명령어의 저장할 메모리 위치만이 크래시를 발생시킨 명령어로부터 영향을 받은 경우, 위험도를 PE(probably exploitable)의 단계로 분류하고, 저장 계열 명령어의 저장할 데이터를 가진 레지스터와 저장할 메모리 위치가 크래시를 발생시킨 명령어로부터 영향을 받은 경우 위험도를 E(exploitable) 단계로 분류한다.

[0087] 또한, 위험도 분석부(146)는 명령어가 사용가능성에 기초하여 식별된 저장 계열 명령어이고, 저장 계열 명령어의 저장할 데이터를 가진 레지스터와 저장할 메모리 위치가 크래시를 발생시킨 명령어로부터 영향을 받은 경우 위험도를 SE(Strongly Exploitable) 단계로 분류한다.

[0089] 도 7은 본 발명의 일 실시 예에 따른 크래시 위험도 분석 방법에 대한 개략적인 처리 흐름도이다.

[0090] 도 7을 참조하면, 디스어셈블러(110)가 바이너리 프로그램을 디스어셈블링하여 기계어로 변환한다(S710). 특히, 디스어셈블러(110)는 크래시 위험도 분석 대상인 바이너리 프로그램(binary program)(예컨대, ARM 바이너리 프로그램)을 디스어셈블링(disassembling)하여 기계어로 변환한다.

[0091] S710이 수행되면, 디스어셈블러(110)가 상기 기계어로부터 제어 흐름 정보를 획득한다(S720). 이때, 제어 흐름 정보는 이후에 실시되는 위험도 분석 단계(S500)에서 각 명령어들의 도달 지점이 어디까지인지, 또는 실제 사용 지점이 어느 지점인지를 도출하기 위한 참고 자료로 사용될 수 있다.

[0092] 단계 S720이 수행되면, 중간 언어 변환기(120)가 상기 기계어를 중간언어로 변환한다(S730). 예를 들어, 중간 언어 변환기(120)는 상기 기계어를 REIL(Reverse Engineering Intermediate Language)라는 중간 언어로 변환한다. 이때, 상기와 같이 기계어를 중간 언어로 변환하는 것은 과다하게 많은 명령어의 수를 줄여서 정적 분석의 편의를 제공하기 위함이다. 예를 들어, 약 300개의 ARM 명령어가 입력된 경우 이를 REIL 이라는 중간 언어로 변환하여 사용할 경우 17개로 줄일 수 있다.

[0093] 단계 S730이 수행되면, 크래시 발생기(130)가 고의로 크래시를 발생시킨다(S740). 이를 위해, 크래시 발생기(130)는 다수의 크래시를 고의로 발생시키기 위해 사전에 제작된 크래시 유발 프로그램을 내장하고, 그 크래시 유발 프로그램(예컨대, Fuzzer 프로그램)에 의해 다수의 크래시들을 발생시킨다. 이는 일반적으로 해커들이 퍼저(Fuzzer)와 같은 크래시 유발 프로그램을 이용하여 대상 프로그램에 많은 크래시(crash)를 발생시키고, 그 크래시들을 직접 하나씩 분석한 후 공격 가능성이 있는 경우에 해당 취약점에 맞는 공격 코드를 작성해서 공격하는 공격 패턴을 역 이용하기 위한 것이다. 즉, 상기와 같이 고의로 발생된 크래시들 각각에 대한 공격 가능성(즉, 위험도)을 자동으로 분석하기 위함이다.

[0094] 단계 S740이 수행되면, 크래시 위험도 분석기(140)가 S730 단계에서 중간 언어로 변환된 명령어들과, S740 단계에서 발생된 크래시 정보에 기초하여 해당 크래시의 위험도를 분석한다(S750). 즉, 크래시 위험도 분석기(140)는 중간언어로 변환된 명령어들 중 크래시들을 발생시킨 명령어들 각각에 의해 영향을 받는 모든 명령어를 정적으로 분석하여 해당 크래시의 위험도를 분석한다.

[0095] 위험도 분석기가 크래시의 위험도를 분석하는 방법에 대한 상세한 설명은 도 8을 참조하기로 한다.

[0096] 도 8은 본 발명의 일 실시예에 따른 위험도의 단계를 판단하는 방법을 설명하기 위한 흐름도이다.

[0097] 도 8을 참조하면, 위험도 분석기는 크래시들을 발생시킨 명령어들 각각에 대하여 도달지점들과 사용가능성(available)을 식별하고(S802), 도달지점들과 사용가능성에 기초하여 프로그램의 제어권을 옮길 수 있는 명령어를 식별한다(S804).

[0098] 단계 S804가 수행되면, 위험도 분석기는 식별된 명령어가 저장(store) 계열 명령어인지를 판단한다(S806).

[0099] 단계 S806의 판단결과, 저장 계열 명령어인 경우, 위험도 분석기는 저장 계열 명령어의 저장할 데이터를 가진 레지스터와 저장할 메모리 위치 모두가 크래시를 발생시킨 명령어로부터 영향을 받는지를 판단한다(S808).

[0100] 단계 S808의 판단결과 저장할 데이터를 가진 레지스터와 저장할 메모리 위치 모두에 영향을 받으면, 위험도 분석기는 해당 저장 계열 명령어가 도달 지점에 기초하여 식별된 명령어인지를 판단한다(S810).

[0101] 단계 S810의 판단결과, 해당 저장 계열 명령어가 도달 지점에 기초하여 식별된 명령어인 경우, 위험도 분석기는

위험도를 E(exploitable) 단계로 분류한다(S812).

- [0102] 만약, 단계 S810의 판단결과, 해당 저장 계열 명령어가 도달 지점에 기초하여 식별된 명령어가 아니면, 위험도 분석기는 사용가능성에 기초하여 식별된 명령어인지를 판단한다(S814).
- [0103] 단계 S814의 판단결과, 해당 저장 계열 명령어가 사용가능성에 기초하여 식별된 명령어이면, 위험도 분석기는 위험도를 SE(Strongly Exploitable) 단계로 분류한다(S816).
- [0104] 만약, 단계 S808의 판단결과 저장할 데이터를 가진 레지스터와 저장할 메모리 위치 모두에 영향을 받지 않으면, 위험도 분석기는 둘중 하나에 영향을 받는지를 판단한다(S818).
- [0105] 단계 S818의 판단결과, 저장할 데이터를 가진 레지스터 또는 저장할 메모리 위치 중 하나에 의해 영향을 받으면, 위험도 분석기는 위험도를 PE(probably exploitable)의 단계로 분류하고(S820), 영향을 받지 않으면 위험도를 NE(not exploitable)의 단계로 분류한다(S822).
- [0106] 만약, 단계 S806의 판단결과, 저장 계열 명령어가 아니면, 위험도 분석기는 분기 계열 명령어인지를 판단한다(S824).
- [0107] 단계 S824의 판단결과 분기 계열 명령어이면, 위험도 분석기는 분기 계열 명령어의 목적 주소(Target Address)가 크래시를 발생시킨 명령어로부터 영향을 받는지를 판단한다(S826).
- [0108] 단계 S826의 판단결과, 분기 계열 명령어의 목적 주소가 영향을 받으면, 위험도 분석기는 해당 분기 계열 명령어가 도달 지점에 기초하여 식별된 명령어인지를 판단한다(S828).
- [0109] 단계 S828의 판단결과, 해당 분기 계열 명령어가 도달 지점에 기초하여 식별된 명령어인 경우, 위험도 분석기는 위험도를 E(exploitable) 단계로 분류한다(S830).
- [0110] 만약, 단계 S828의 판단결과, 해당 분기 계열 명령어가 도달 지점에 기초하여 식별된 명령어가 아니면, 위험도 분석기는 해당 분기 계열 명령어가 사용 가능성에 기초하여 식별된 명령어인지를 판단한다(S832).
- [0111] 단계 S832의 판단결과, 사용 가능성에 기초하여 식별된 명령어이면, 위험도 분석기는 위험도를 SE(Strongly Exploitable) 단계로 분류한다(S834).
- [0113] 이제까지 본 발명에 대하여 그 바람직한 실시예들을 중심으로 살펴보았다. 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자는 본 발명이 본 발명의 본질적인 특성에서 벗어나지 않는 범위에서 변형된 형태로 구현될 수 있음을 이해할 수 있을 것이다. 그러므로 개시된 실시예들은 한정적인 관점이 아니라 설명적인 관점에서 고려되어야 한다. 본 발명의 범위는 전술한 설명이 아니라 특허청구범위에 나타나 있으며, 그와 동등한 범위 내에 있는 모든 차이점은 본 발명에 포함된 것으로 해석되어야 할 것이다.

부호의 설명

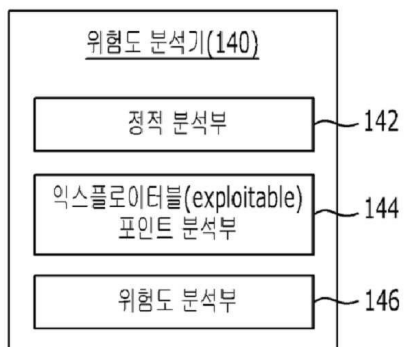
- [0115] 100 : 크래시 위험도 분석 장치
- 110 : 디스어셈블러
- 120 : 중간언어 변환기
- 130 : 크래시 발생기
- 140 : 위험도 분석기
- 142 : 정적 분석부
- 144 : 익스플로이터블 포인트 분석부
- 146 : 위험도 분석부

도면

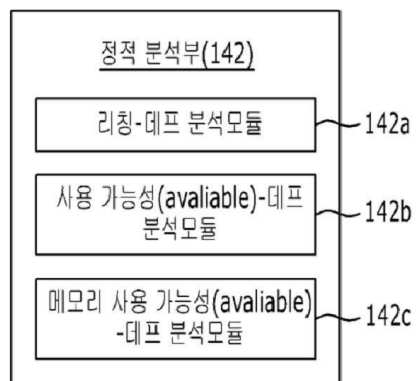
도면1



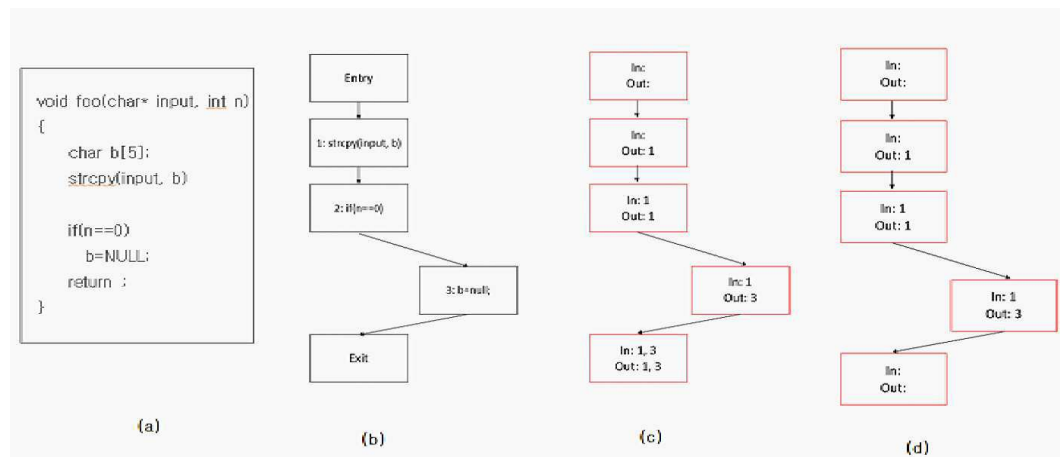
도면2



도면3



도면4



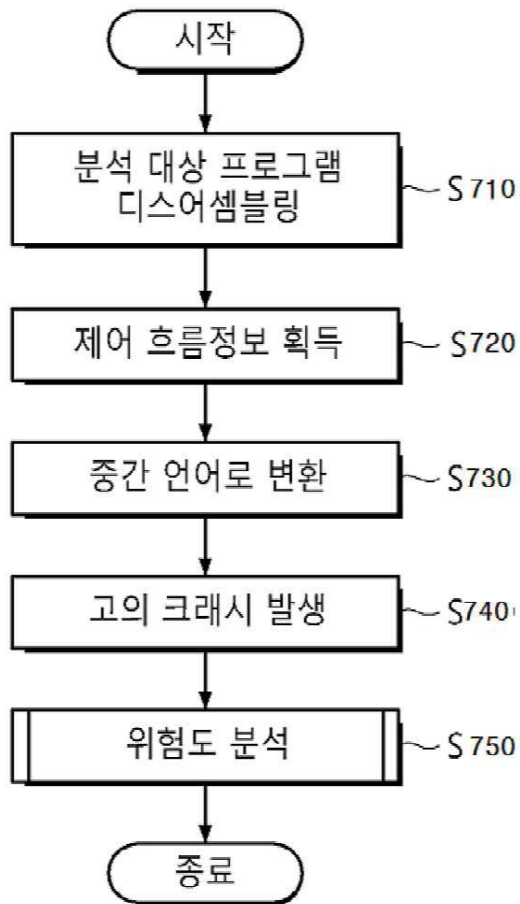
도면5

	Reaching Definition	Available Definition	Memory Available Definition
Domain	Sets of definition	Sets of definition	Sets of definition
Kill	메모리에 대해서 kill 하지 않음	메모리에 대해서 kill 하지 않음	메모리를 레지스터 처럼 보고 kill 함
Direction	Forward	Forward	
Main Logic	$OUT = GEN + (IN - KILL)$	$OUT = GEN + (IN - KILL)$	
Transfer function	$F_B(x) = gen_B \cup (x - kill_B)$	$F_B(x) = gen_B \cup (x - kill_B)$	
Meet	$\cup$	$\cap$	
Initialize	$OUT(X) = \emptyset$	$OUT(X) = U_{universe}$	
Equation	$out(B) = F_B(in(B))$ $in(B) = \bigwedge_{Q \in Pred(B)} out(Q)$	$out(B) = F_B(in(B))$ $in(B) = \bigwedge_{Q \in Pred(B)} out(Q)$	

도면6

타겟 명령어	Condition	Grades
분기 명령어	It can control address of destination, when using available definition	SE (Strongly Exploitable)
	It can control address of destination, when using reaching definition	E (Exploitable)
메모리 저장 명령어	It can control both data and address of destination, when using available definition	SE
	It can control both data and address of destination, when using reaching definition	E
	It can control only data, when using reaching definition	PE (Probably Exploitable)
	It can control only address of destination, when using reaching definition	PE
Not applicable		NE (Not exploitable)

도면7



도면8

