(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2022년11월03일
(11) 등록번호 10-2461532
(24) 등록일자 2022년10월27일

(51) 국제특허분류(Int. Cl.)
G06F 8/75 (2018.01) *G06F 8/34* (2018.01)
G06F 8/41 (2018.01) *G06F 9/30* (2018.01)
(52) CPC특허분류
G06F 8/75 (2013.01)
G06F 8/34 (2013.01)
(21) 출원번호 10-2020-0180756
(22) 출원일자 2020년12월22일
심사청구일자 2020년12월22일
(65) 공개번호 10-2021-0081280
(43) 공개일자 2021년07월01일
(30) 우선권주장
1020190173098 2019년12월23일 대한민국(KR)
(56) 선행기술조사문헌
KR101051600 B1*
KR1020010075871 A*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
충남대학교 산학협력단
대전광역시 유성구 대학로 99 (궁동, 충남대학교)
(72) 발명자
조은선
대전광역시 유성구 어은로 57 한빛아파트, 110동 1504호
목성균
대전광역시 서구 가장로 131-2, 403호
(74) 대리인
이은철, 김재문

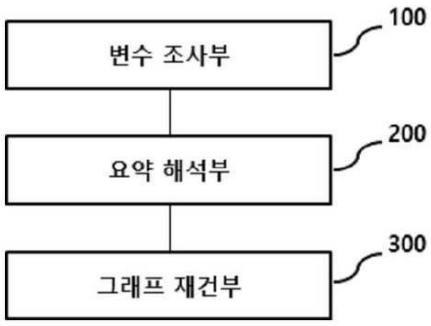
전체 청구항 수 : 총 1 항 심사관 : 지정훈

(54) 발명의 명칭 **프로그램 제어 흐름 그래프 재건 시스템 및 방법**

(57) 요 약

본 기술은 프로그램 제어 흐름 그래프 재건 시스템 및 방법에 관한 것이다. 본 기술의 구현 예에 따르면, 요약 해석을 사용하여 제어 흐름 그래프를 재건함으로써 오버헤드가 줄어든 프로그램 제어 흐름 그래프 재건 시스템 및 방법을 제공할 수 있는 이점이 있다.

대 표 도 - 도1



(52) CPC특허분류
G06F 8/433 (2013.01)
G06F 9/3005 (2013.01)

이 발명을 지원한 국가연구개발사업

과제고유번호	1711103337
과제번호	2019-0-01343-002
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기획평가원
연구사업명	융합보안핵심인재양성사업
연구과제명	융합보안석사과정
기 여 율	1/1
과제수행기관명	한국인터넷진흥원
연구기간	2020.04.01 ~ 2020.12.31

명 세 서

청구범위

청구항 1

메모리 위치 분석을 통해 프로그램의 각 변수들이 가질 수 있는 값을 조사하는 변수 조사부;

상기 프로그램의 명령어의 피연산자의 값을 요약 해석하여 정의하는 요약 해석부; 및

상기 변수 조사부에서 조사된 값과 상기 요약 해석부에서 정의된 예외 발생 값의 요약 해석에 따라 제어 흐름 그래프를 재건하는 그래프 재건부;를 포함하고,

상기 요약 해석부는,

상기 변수 조사부에서 조사된 값을 이용하여 피연산자의 값의 범위를 식별하고, 식별된 피연산자 값의 범위에 예외 발생 값이 존재하는지를 기준으로 피연산자의 값을 요약 해석하여 정의하고,

상기 그래프 재건부는,

상기 요약 해석부에서 정의된 피연산자의 요약 해석 중 예외 발생 값을 포함하는 것을 구분하고, 예외를 발생시키는 피연산자에 대한 그래프 간선을 생성하여,

예외를 포함하는 제어 흐름 그래프를 재건하고, 예외를 발생시키는 피연산자에 대한 그래프 간선을 생성 시 예외 발생 값을 포함하는 피연산자의 요약 해석에 대한 경로만을 고려하여 시간적 오버헤드를 줄일 수 있는 것을 특징으로 하는 프로그램 제어 흐름 그래프 재건 시스템.

청구항 2

삭제

발명의 설명

기 술 분 야

[0001] 프로그램 제어 흐름 그래프 재건 시스템 및 방법에 관한 것으로, 더욱 상세하게는 예외 처리를 포함하여 제어 흐름 그래프를 재건하는 프로그램 제어 흐름 그래프 재건 시스템 및 방법에 관한 것이다.

배 경 기 술

[0003] 바이너리 프로그램의 분석은 우선적으로 프로그램의 제어 흐름 그래프를 재건한 뒤에 본격적인 분석이 이루어진다. 제어흐름 그래프가 얼마나 정확하게 재건되었는지에 따라 분석의 정확도 또한 좌우된다.

[0004] 기존의 제어 흐름 그래프 재건 방법은 프로그램의 예외처리 코드를 포함시기지 않아 이에 대한 분석이 누락되어 분석의 부정확성을 초래할 수 있다.

[0005] 그러나 악성코드는 자신들의 코드 분석을 어렵게 하기 위하여 예외처리 코드에 핵심 코드를 넣는 경우가 있어 이러한 악성코드를 분석하기 위하여 반드시 제어 흐름 그래프 재건 시 예외처리를 포함할 필요가 있다.

[0006] 최근 심볼릭 실행을 이용하여 예외처리를 포함하는 제어 흐름 그래프 재건 방법이 제안되었으나, 모든 대상 명령어에 대하여 심볼릭 실행이 이루어져야 하기 때문에 시간적 오버헤드가 발생하고 심볼릭 실행의 단점을 그대로 가지고 있다.

선행기술문헌

비특허문헌

[0008] (비특허문헌 0001) 1. Babak Yadegari, Jon Stephens and Saumya Debray, “Analysis of Exception-Based

Control Transfers," Procedding CODASPY'17 Proceedings of the 7th ACM on Conference on Data and Application Security and Privacy, 2017, pp. 205-216.

발명의 내용

해결하려는 과제

[0009] 본 발명은 상기와 같은 문제를 해결하기 위한 것으로서, 예외 처리를 분석에 포함함으로써 핵심 코드를 예외처리에 포함한 악성코드의 분석이 가능해지고 시간적 오버헤드가 적은 프로그램 제어 흐름 그래프 재건 시스템 및 방법을 제공하는데 그 목적이 있다.

과제의 해결 수단

[0010] 상기와 같은 목적을 달성하기 위한 본 발명은 메모리 위치 분석을 통해 프로그램의 각 변수들이 가질 수 있는 값을 조사하는 변수 조사부; 상기 프로그램의 명령어의 피연산자의 값을 요약 해석하여 정의하는 요약 해석부; 및 상기 변수 조사부에서 조사된 값과 상기 요약 해석부에서 정의된 요약 해석에 따라 제어 흐름 그래프를 재건하는 그래프 재건부;를 포함하는 것을 특징으로 한다.

[0011] 또한 본 발명의 다른 실시예 따른 프로그램 제어 흐름 그래프 재건 방법은 메모리 위치 분석을 통해 프로그램의 각 변수들이 가질 수 있는 값을 조사하는 단계; 상기 프로그램의 명령어의 피연산자의 값을 요약 해석하여 정의하는 단계; 및 상기 변수 조사부에서 조사된 값과 상기 요약 해석부에서 정의된 요약 해석에 따라 제어 흐름 그래프를 재건하는 단계를 포함하는 것을 특징으로 한다.

발명의 효과

[0013] 전술한 바와 같은 본 발명에 따르면, 요약 해석을 사용하여 제어 흐름 그래프를 재건함으로써 핵심 코드를 예외처리에 포함한 악성코드의 분석이 가능해지고 시간적 오버헤드가 적은 프로그램 제어 흐름 그래프 재건 시스템 및 방법을 제공할 수 있는 이점이 있다.

도면의 간단한 설명

[0015] 도 1은 본 발명에 따른 프로그램 제어 흐름 그래프 재건 시스템의 블록도를 도시한다.

도 2는 프로그램 예외 처리 코드의 한 예를 보여준다.

도 3은 도 2의 코드에 식을 대입한 코드의 한 예를 보여준다.

도 4는 도 3 프로그램의 요약 해석에 따른 추상화 결과를 도시한다.

도 5는 예외처리가 고려되지 않은 기존의 제어 흐름 그래프를 도시한다.

도 6는 예외처리를 고려한 제어 흐름 그래프를 도시한다.

도 7는 본 발명에 따른 프로그램 제어 흐름 그래프 재건 방법의 흐름도를 도시한다.

발명을 실시하기 위한 구체적인 내용

[0016] 본 발명의 이점 및 특징, 그리고 그것들을 달성하는 방법은 첨부되는 도면과 함께 후술되어 있는 실시예들을 참조하면 본 발명의 이점 및 특징, 그리고 그것들을 달성하는 방법은 첨부되는 도면과 함께 후술되어 있는 실시예들을 참조하면 명확해질 것이다. 이에 앞서 본 발명에 관련된 공지 기능 및 그 구성에 대한 구체적인 설명이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우에는 그 구체적인 설명을 생략하였음에 유의해야 할 것이다.

[0017] 바이너리 프로그램의 제어 흐름 그래프를 만들기 위해서는 점프 명령문을 분석하여야 한다. 그런데 예를 들면, 리눅스에서 C 언어로 작성된 이진 프로그램 등에서 예외는 신호를 통해 처리되기 때문에 예외를 위한 명시적인 점프 문장이 없어 암묵적인 제어 흐름이 생성된다.

[0018] 따라서 간단한 리버스 엔지니어링으로 제어 흐름 그래프를 예외를 포함하여 재건할 수 없어 예외 처리를 이용하

는 악성코드 프로그램을 분석하는 것은 쉽지 않다.

[0019] 이와 같이 예외는 프로그램의 취약점으로 나타날 수 있어 테스트되어야 할 필요가 있으나, 예외를 고려하지 않는 제어 흐름 그래프는 예외를 포함한 완전한 코드 커버리지가 이루어지지 않아 악성코드를 제대로 분석하지 못하는 문제가 있다.

[0020] 도 1은 본 발명에 따른 프로그램 제어 흐름 그래프 재건 시스템의 블록도를 도시한다.

[0021] 본 발명에 따른 프로그램 제어 흐름 그래프 재건 시스템은 변수 조사부(100), 요약 해석부(200) 및 그래프 재건부(300)을 포함한다.

[0022] 변수 조사부(100)는 메모리 위치 분석을 통해 프로그램의 각 변수들이 가질 수 있는 값을 조사한다. 요약 해석부(200)는 프로그램의 명령어의 피연산자의 값을 요약 해석하여 정의한다. 그래프 재건부(300)는 변수 조사부(100)에서 조사된 값과 요약 해석부(200)에서 정의된 요약 해석에 따라 제어 흐름 그래프를 재건한다.

[0023] 요약 해석의 예를 들면, 피연산자 값에 있어서 임의의 양수를 “PLUS”로 정의할 수 있고, 0을 “ZERO”로 정의할 수 있으며, 임의의 음수를 “MINUS”로 정의할 수 있다.

[0024] 도 2는 프로그램 예외 처리 코드의 한 예를 보여준다. 도 2를 참조하면 코드 5행에 예외 상태에 따른 함수가 기재되어 있다. 이 코드에서 SIGFPE는 0으로 나누기와 같은 치명적인 산술 오류를 가리키는 매크로 신호이다. 도 2의 코드 5행에 기재된 함수 “fpe_handler()”는 예외가 발생하면 실행된다.

[0025] 예를 들면, 0으로 나누는 명령이 기재된 도 2의 라인 9, 11 또는 13에서 “fpe_handler()”가 실행될 것이다.

[0026] 본 발명은 심볼릭 실행의 제약을 극복하기 위해 요약 해석을 이용하는 전처리를 수행한다.

[0027] 요약 해석을 이용함으로써 완전한 코드 커버리지와 경로의 수에 무관하게 한 번의 실행으로 결과를 찾을 수 있다.

[0028] 본 발명에 따른 요약 해석은 예외를 발생시키는 명령어의 피연산자 값의 범위를 식별한다. 그 다음 피연산자 값의 범위에 예외를 발생시키는 값이 포함되어 있는지를 확인한다.

[0029] 도 3은 도 2의 코드에 식을 대입한 코드의 한 예를 보여준다. 도 3을 참조하면, 코드의 4, 6, 8행은 나눗셈을 수행한다. 기존의 예외를 고려한 프로그램 제어 흐름 그래프 재건은 도 3에 도시된 코드의 4, 6, 8행에 대해 0으로 나누는 것인지 심볼릭 실행을 수행한다.

[0030] 도 4는 도 3 프로그램의 요약 해석에 따른 추상화 결과를 도시한다. 본 발명은 도 4와 같은 요약 해석을 통해 피연산자의 값을 추상화한다. 도 4를 참조하면, 정사각형의 넓이를 구하는 공식은 양수이므로 도 3에 도시된 코드 4행의 추상화된 피연산자의 값은 “PLUS”이다.

[0031] 그러나 도 3에 도시된 코드의 6행의 식에서는 양수를 뺀 결과를 알 수 없으므로 6 행의 피연산자 값 추상화의 결과는 “PLUS”, “MINUS” 또는 “ZERO”가 된다. 이러한 결과로 도 3에 도시된 코드의 6행에서 제수를 0으로 만드는 입력 값을 찾아낼 수 있다.

[0032] 도 5는 예외처리가 고려되지 않은 기존의 제어 흐름 그래프를 도시하고, 도 6는 예외처리를 고려한 제어 흐름 그래프를 도시한다.

[0033] 프로그램 코드에 대하여 요약 해석을 이용하여 피연산자가 예외를 발생시킬 수 있는 값을 가질 수 있는지 판단한 뒤, 그래프 간선을 생성한다.

[0034] 도 7는 본 발명에 따른 프로그램 제어 흐름 그래프 재건 방법의 흐름도를 도시한다.

[0035] 본 발명의 다른 실시예에 따른 프로그램 제어 흐름 그래프 재건 방법은 먼저 메모리 위치 분석을 통해 프로그램의 각 변수들이 가질 수 있는 값을 조사한다(S100). 이어서, 프로그램의 명령어의 피연산자의 값을 요약 해석하여 정의한다(S200). 이어서, S100단계에서 조사된 값과 S200단계에서 정의된 요약 해석에 따라 제어 흐름 그래프를 재건한다(S300).

[0036] 이상에서 설명된 시스템은 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, 콘트롤러, 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 애플리케이션을 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 콘트롤러를 포함할 수 있다. 또한, 병렬 프로세서(parallel processor)와 같은, 다른 처리 구성(processing configuration)도 가능하다.

[0037] 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.

[0038] 이상으로 본 발명의 기술적 사상을 예시하기 위한 바람직한 실시예와 관련하여 설명하고 도시하였지만, 본 발명은 이와 같이 도시되고 설명된 그대로의 구성 및 작용에만 국한되는 것이 아니며, 기술적 사상의 범주를 일탈함이 없이 본 발명에 대해 다수의 변경 및 수정이 가능함을 잘 이해할 수 있을 것이다. 따라서, 그러한 모든 적절한 변경 및 수정과 균등물들도 본 발명의 범위에 속하는 것으로 간주되어야 할 것이다.

부호의 설명

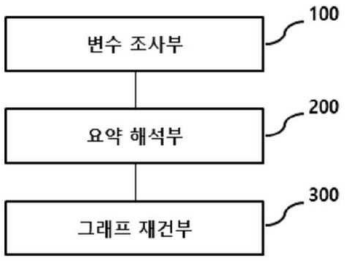
[0040] 100 : 변수 조사부

200 : 요약 해석부

300 : 그래프 재건부

도면

도면1



도면2

```
1: void fpe_handler(){
2:    …
3: }
4: void main(int argc, char* argv[]){
5:    signal(SIGFPE, fpe_handler);
6:    int result0, result1, result2, r0, r1;
7:    …
8:    if( r0 > 0 )
9:       result0=(r0*r0+1)/(r1*r1+1);
10:   else
11:      result1=(r0*r0+1)/(r0*r0-r1*r1);
12:   if( r1 > 0 )
13:      result2=(r0*r0+1)/(r0*r1);
14:}
```

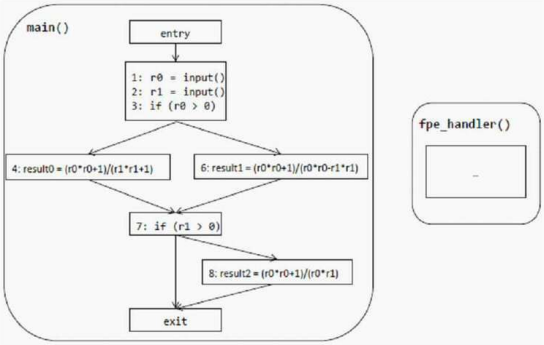
도면3

```
1: r0 = input()
2: r1 = input()
3: if (r0 > 0)
4:    result0 = (r0*r0+1)/(r1*r1+1)
5: else
6:    result1 = (r0*r0+1)/(r0*r0-r1*r1);
7: if (r1 > 0)
8:    result2=(r0*r0+1)/(r0*r1);
```

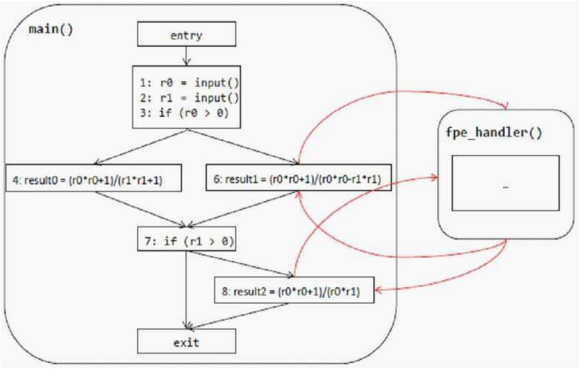
도면4

```
1: r0 = UNKNOWN
2: r1 = UNKNOWN
3: result0 = PLUS/PLUS
4: result1 = PLUS/ALL
5: result2 = PLUS/ALL
```

도면5



도면6



도면7