(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2022-0096589
(43) 공개일자 2022년07월07일

(51) 국제특허분류(Int. Cl.)
G06F 21/12 (2013.01) *G06F 21/14* (2013.01)
G06F 40/151 (2020.01) *G06F 8/41* (2018.01)
(52) CPC특허분류
G06F 21/12 (2013.01)
G06F 21/14 (2013.01)
(21) 출원번호 10-2020-0189169
(22) 출원일자 2020년12월31일
심사청구일자 2020년12월31일

(71) 출원인
충남대학교산학협력단
대전광역시 유성구 대학로 99 (궁동, 충남대학교)
(72) 발명자
강서연
충청북도 청주시 흥덕구 대신로 74 금호어울림 201동 1002호
김유민
대전광역시 유성구 문지로 300 효성해링턴플레이스 105동 205호
(뒷면에 계속)
(74) 대리인
이은철, 김재문

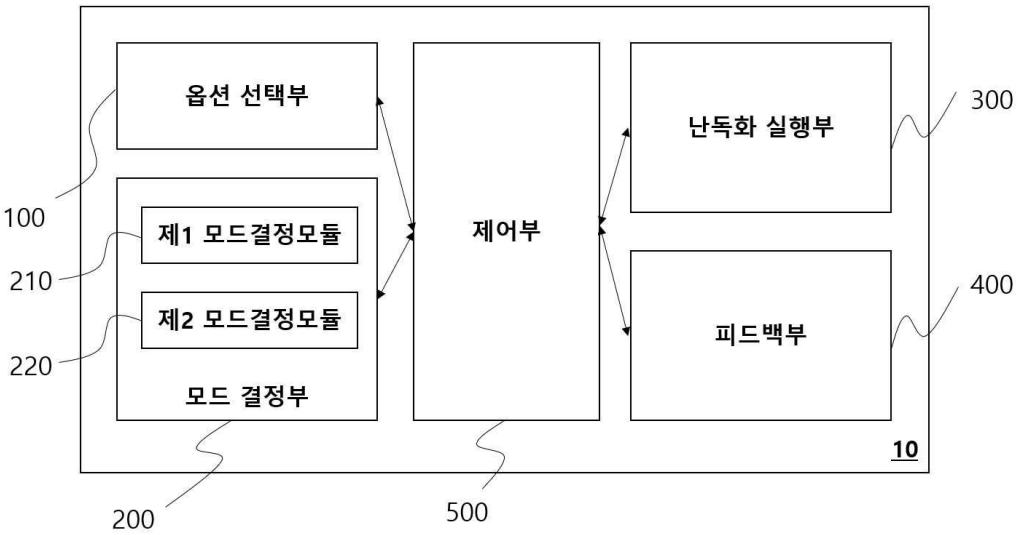
전체 청구항 수 : 총 7 항

(54) 발명의 명칭 **중간언어를 활용한 바이너리 프로그램 난독화 시스템 및 그 방법**

(57) 요 약

본 발명은 중간언어를 활용한 바이너리 프로그램 난독화 시스템에 관한 것으로써, 소프트웨어를 보호하기 위해 중간언어를 활용한 바이너리 프로그램 난독화 시스템 및 그 방법에 관한 것이다. 본 발명의 일 실시예에 따른 중간언어를 활용한 바이너리 프로그램 난독화 시스템에 의하면, 입력받는 언어 프로그램에 따라 해당 언어 프로그램의 난독화를 수행할 수 있어 소프트웨어를 더욱 효과적으로 보호할 수 있는 효과가 있다.

대 표 도 - 도1



(52) CPC특허분류
G06F 40/151 (2020.01)
G06F 8/31 (2013.01)
G06F 8/41 (2013.01)
G06F 2221/2125 (2013.01)
(72) 발명자
이수정
경상북도 봉화군 봉화읍 교촌길 51 삼영주택 A동 508호

조은선
대전광역시 유성구 어은로 57 한빛아파트, 110동 1504호

이 발명을 지원한 국가연구개발사업

과제고유번호	1711102798
과제번호	2015-0-00930-006
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기획평가원
연구사업명	정보통신창의인재양성
연구과제명	SW중심대학(충남대)
기 여 율	1/1
과제수행기관명	충남대학교 산학협력단
연구기간	2020.01.01 ~ 2021.02.28

명 세 서

청구범위

청구항 1

난독화 옵션을 선택하는 옵션 선택부(100);

상기 선택된 난독화 옵션으로 난독화할 원본을 입력받고, 상기 원본의 프로그래밍 언어 레벨에 따라 난독화 모드를 결정하는 모드 결정부(200); 및

상기 모드 결정부를 통해 결정되는 레벨이 저급 언어인 경우, 상기 원본의 저급 언어 프로그램을 상기 난독화 옵션을 수행한 뒤 난독화된 저급 언어 프로그램을 출력하고, 고급 언어인 경우, 상기 원본의 고급 언어 프로그램을 상기 난독화 옵션을 수행하여 난독화된 고급 언어 프로그램을 출력하는 난독화 실행부(300);를 포함하는 것을 특징으로 하는

중간언어를 활용한 바이너리 프로그램 난독화 시스템.

청구항 2

제1항에 있어서,

상기 옵션 선택부는 적어도 어느 하나 이상의 난독화 옵션을 선택받으며,

상기 선택된 난독화 옵션 횟수에 따라 상기 난독화 실행부의 출력을 순차적으로 난독화 실행부의 입력에 피드백 전달하여 난독화가 중첩되게 적용하도록 하는 피드백부(400);를 더 포함하는 것을 특징으로 하는

중간언어를 활용한 바이너리 프로그램 난독화 시스템.

청구항 3

제1항에 있어서,

상기 옵션 선택부는

난독화 옵션 중 입력된 대상에 적용하고자 하는 난독화 기술을 적어도 한 가지 이상 선택하며, 입력한 키워드의 종류와 순서에 따라 난독화가 실행되도록 입력되는 키워드를 통해 선택하는 것을 특징으로 하는

중간언어를 활용한 바이너리 프로그램 난독화 시스템.

청구항 4

제1항에 있어서,

상기 모드 결정부는

상기 난독화할 대상의 프로그래밍 언어가 기계어 또는 어셈블리어인 경우, 저급 언어 난독화 모드로 결정하는 제1 모드 결정모듈; 및

상기 난독화할 대상의 프로그래밍 언어가 기계어 또는 어셈블리어가 아닌 경우, 고급 언어 난독화 모드로 결정하는 제2 모드 결정모듈;을 포함하는 것을 특징으로 하는

중간언어를 활용한 바이너리 프로그램 난독화 시스템.

청구항 5

제1항에 있어서,

상기 난독화 실행부는

상기 모드 결정부를 통해 결정되는 레벨이 저급 언어인 경우, 상기 원본의 저급 언어 레벨을 중간 언어 레벨로 변환하고, 선택한 난독화 옵션에 따라 난독화를 적용한 후 난독화된 중간 언어 레벨의 프로그램을 컴파일하여

저급 언어 프로그램으로 변환하는 것을 특징으로 하는

중간언어를 활용한 바이너리 프로그램 난독화 시스템.

청구항 6

제1항에 있어서,

상기 옵션 선택부는 Opaque predicate 난독화, MBA 난독화, 식별자 난독화, Type change 난독화, Inline Assembly를 활용한 난독화, 소스코드 수준에서 가독성을 떨어뜨리는 Uglyfier 난독화 중 어느 하나 이상을 난독화 옵션으로 선택하는 것을 특징으로 하는 중간언어를 활용한 바이너리 프로그램 난독화 시스템.

청구항 7

(a) 제어부가 난독화 옵션을 선택하는 단계;

(b) 상기 제어부가 선택된 난독화 옵션으로 난독화할 원본을 입력받고, 상기 원본의 프로그래밍 언어 레벨에 따라 난독화 모드를 결정하는 단계; 및

(c) 상기 (b) 단계의 결정결과, 결정되는 레벨이 저급 언어인 경우, 상기 원본의 저급 언어 프로그램을 상기 난독화 옵션을 수행한 뒤 난독화된 저급 언어 프로그램을 출력하고, 고급 언어인 경우, 상기 원본의 고급 언어 프로그램을 상기 난독화 옵션을 수행하여 난독화된 고급 언어 프로그램을 출력하는 단계;를 포함하는 것을 특징으로 하는

중간언어를 활용한 바이너리 프로그램 난독화 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 중간언어를 활용한 바이너리 프로그램 난독화 시스템에 관한 것으로써, 더욱 상세하게는 소프트웨어를 보호하기 위해 중간언어를 활용한 바이너리 프로그램 난독화 시스템 및 그 방법에 관한 것이다.

배 경 기 술

[0002] 소프트웨어 불법 복제란, 소프트웨어 저작권자의 명백한 동의 없이 불법적으로 소프트웨어의 내용을 복제하는 것을 말한다. 이러한 소프트웨어 불법 복제는 소프트웨어 개발 산업에 막대한 손실을 불러오게 된다.

[0003] 이렇게 소프트웨어 산업에 피해를 주는 소프트웨어 불법 복제 행위는, 역공학(Reverse Engineering) 기술을 이용해 소프트웨어를 복제한다. 소프트웨어의 코드를 분석해 프로그램이 어떠한 구조로 이루어져 있고, 어떤 알고리즘, 로직을 이용하는지를 파악하는 것이다. 공격자는 역공학을 이용해 소프트웨어의 취약점을 알아내어 악용할 수 있다. 이를 방지하기 위한 기술로 소프트웨어 난독화 기술이 있다.

[0004] 소프트웨어 난독화 기술이란 프로그램을 읽기 어렵게 만드는 기술이다. 난독화 기술이 적용되면 가독성이 떨어질 뿐만 아니라 프로그램의 흐름을 알기 힘들어 분석이 어렵게 된다.

[0005] 소프트웨어 보안을 위한 다양한 난독화 기술 개발이 이루어지고 있다. 하지만 난독화 기술이 개발되면 그에 대항하는 역난독화 기술이 개발되는 순환이 형성되어 있다. 따라서 소프트웨어를 보호하기 위해서는 지속적인 난독화 연구가 필요하다.

선행기술문헌

특허문헌

[0006] (특허문헌 0001) 공개특허 제10-2017-0049388호

발명의 내용

해결하려는 과제

[0007] 본 발명은 상술한 문제를 해결하고자 고안한 것으로, 입력받는 언어 프로그램에 따라 해당 언어 프로그램의 난독화를 수행하기 위해 중간언어를 활용한 바이너리 프로그램 난독화 시스템 및 그 방법을 제공함에 목적이 있다.

과제의 해결 수단

[0008] 본 발명의 일 측면에 따른 중간언어를 활용한 바이너리 프로그램 난독화 시스템은 난독화 옵션을 선택하는 옵션 선택부(100); 상기 선택된 난독화 옵션으로 난독화할 원본을 입력받고, 상기 원본의 프로그래밍 언어 레벨에 따라 난독화 모드를 결정하는 모드 결정부(200); 및 상기 모드 결정부를 통해 결정되는 레벨이 저급 언어인 경우, 상기 원본의 저급 언어 프로그램을 상기 난독화 옵션을 수행한 뒤 난독화된 저급 언어 프로그램을 출력하고, 고급 언어인 경우, 상기 원본의 고급 언어 프로그램을 상기 난독화 옵션을 수행하여 난독화된 고급 언어 프로그램을 출력하는 난독화 실행부(300);를 포함한다.

[0009] 바람직하게 상기 옵션 선택부는 적어도 어느 하나 이상의 난독화 옵션을 선택받으며, 상기 선택된 난독화 옵션 횟수에 따라 상기 난독화 실행부의 출력을 순차적으로 난독화 실행부의 입력으로 피드백 전달하여 난독화가 중첩되게 적용하도록 하는 피드백부(400);를 더 포함한다.

[0010] 바람직하게 옵션 선택부는 난독화 옵션 중 입력된 대상에 적용하고자 하는 난독화 기술을 적어도 한 가지 이상 선택하며, 입력한 키워드의 종류와 순서에 따라 난독화가 실행되도록 입력되는 키워드를 통해 선택한다.

[0011] 바람직하게 모드 결정부(200)는 상기 난독화할 대상의 프로그래밍 언어가 기계어 또는 어셈블리어인 경우, 저급 언어 난독화 모드로 결정하는 제1 모드 결정모듈(210); 및 상기 난독화할 대상의 프로그래밍 언어가 기계어 또는 어셈블리어가 아닌 경우, 고급 언어 난독화 모드로 결정하는 제2 모드 결정모듈(220);을 포함한다.

[0012] 바람직하게 난독화 실행부는 상기 모드 결정부를 통해 결정되는 레벨이 저급 언어인 경우, 상기 원본의 저급 언어 레벨을 중간 언어 레벨로 변환하고, 선택한 난독화 옵션에 따라 난독화를 적용한 후 난독화된 중간 언어 레벨의 프로그램을 컴파일하여 저급 언어 프로그램으로 변환한다.

[0013] 바람직하게 옵션 선택부는 Opaque predicate 난독화, MBA 난독화, 식별자 난독화, Type change 난독화, Inline Assembly를 활용한 난독화, 소스코드 수준에서 가독성을 떨어뜨리는 Uglyfier 난독화 중 어느 하나 이상을 난독화 옵션으로 선택한다.

[0014] 한편, 옵션 선택부(100), 모드 결정부(200), 난독화 실행부(300), 피드백부(400), 제어부(500)를 포함하는 중간언어를 활용한 바이너리 프로그램 난독화 시스템에 있어서, 중간언어를 활용한 바이너리 프로그램 난독화 방법은 (a) 제어부가 난독화 옵션을 선택하는 단계; (b) 상기 제어부가 선택된 난독화 옵션으로 난독화할 원본을 입력받고, 상기 원본의 프로그래밍 언어 레벨에 따라 난독화 모드를 결정하는 단계; 및 (c) 상기 (b) 단계의 결정 결과, 결정되는 레벨이 저급 언어인 경우, 상기 원본의 저급 언어 프로그램을 상기 난독화 옵션을 수행한 뒤 난독화된 저급 언어 프로그램을 출력하고, 고급 언어인 경우, 상기 원본의 고급 언어 프로그램을 상기 난독화 옵션을 수행하여 난독화된 고급 언어 프로그램을 출력하는 단계;를 포함한다.

발명의 효과

[0015] 본 발명의 일 실시예에 따른 중간언어를 활용한 바이너리 프로그램 난독화 시스템에 의하면, 입력받는 언어 프로그램에 따라 해당 언어 프로그램의 난독화를 수행할 수 있어 소프트웨어를 더욱 효과적으로 보호할 수 있는 효과가 있다.

도면의 간단한 설명

[0016] 도 1은 본 발명의 일 실시예에 따른 중간언어를 활용한 바이너리 프로그램 난독화 시스템의 구성을 나타낸 도면이다.

도 2는 본 발명의 일 실시예에 따른 중간언어를 활용한 바이너리 프로그램 난독화 시스템의 고급 언어 난독화 동작 과정을 나타낸 도면이다.

도 3은 본 발명의 일 실시예에 따른 중간언어를 활용한 바이너리 프로그램 난독화 시스템의 저급 언어 난독화 동작 과정을 설명하기 위한 도면이다.

도 4는 난독화 옵션 키워드의 입력선택을 설명하기 위한 도면이다.

도 5는 저급 언어 난독화 수행을 설명하기 위한 도면이다.

도 6은 난독화 전 코드를 나타낸 도면이다.

도 7은 Var-MBA 옵션 적용 후 결과를 나타낸 도면이다.

도 8은 난독화 전 코드(도 6)를 compile한 어셈블리 코드이다.

도 9는 MBA 난독화를 적용한 결과 파일을 compile해 생성된 어셈블리 코드이다.

도 10은 Dynamic, MBA 옵션 적용 후를 나타낸 도면이다.

도 11은 Dynamic-MBA-Var-Ugly 옵션 적용 후를 나타낸 도면이다.

도 12는 도 10의 코드를 컴파일(compile)한 어셈블리 코드의 일부이다.

도 13은 원본 코드를 컴파일한 난독화 전 코드를 나타낸 도면이다.

도 14는 변환한 LLVM 중간 언어에 MBA 난독화를 두 번 적용했을 때의 모습이다.

도 15는 Obfus의 web 기반 프로그램 중 고급 언어 난독화 페이지이다.

도 16은 Obfus의 web 기반 프로그램 중 저급 언어 난독화 페이지이다.

도 17은 난독화 수행 결과를 나타낸 도면이다.

도 18은 Invariant Opaque Predicates을 설명한 그림이다.

도 19는 Contextual Opaque Predicates을 설명하기 위한 도면이다.

도 20은 Dynamic Opaque Predicates을 설명하기 위한 도면이다.

도 21의 왼쪽과 같이 간단하게 간파될 수 있는 식에 대해, 도 21의 오른쪽과 같이 논리연산을 섞어 읽기 어렵게 하는 것이다.

도 22의 오른쪽은 도 22의 왼쪽과 같은 코드에 대해 메서드 이름과 매개변수 이름에 대해 난독화한 것이다.

도 23은 간단한 어셈블리어 프로그램이다.

도 24는 간단한 C 언어 프로그램이다.

도 25는 본 발명의 일 실시예에 따른 중간언어를 활용한 바이너리 프로그램 난독화 방법을 나타낸 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0017] 본 발명의 실시예에서 제시되는 특정한 구조 내지 기능적 설명들은 단지 본 발명의 개념에 따른 실시예를 설명하기 위한 목적으로 예시된 것으로, 본 발명의 개념에 따른 실시예들은 다양한 형태로 실시될 수 있다. 또한, 본 명세서에 설명된 실시예들에 한정되는 것으로 해석되어서는 아니 되며, 본 발명의 사상 및 기술 범위에 포함되는 모든 변경물, 균등물 내지 대체물을 포함하는 것으로 이해되어야 한다.

[0018] 한편, 본 발명에서 제1 및/또는 제2 등의 용어는 다양한 구성요소들을 설명하는데 사용될 수 있지만, 상기 구성요소들은 상기 용어들에 한정되지는 않는다. 상기 용어들은 하나의 구성요소를 다른 구성요소들과 구별하는 목적으로만, 예컨대 본 발명의 개념에 따른 권리 범위로부터 벗어나지 않는 범위 내에서, 제1 구성요소는 제2 구성요소로 명명될 수 있고, 유사하게 제2 구성요소는 제1 구성요소로도 명명될 수 있다.

[0019] 이하 첨부된 도면을 참조하여 본 발명의 실시예를 설명한다. 본 발명의 실시예를 설명함에 있어서, 관련된 공지 기능 혹은 구성에 대한 설명이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우 그 설명을 생략하였다.

[0020] 배포한 소프트웨어는 역공학을 통해서 불법 복제되거나 관리를 위한 기능과 같이 공개하기 어려운 부분에 대한 취약점이 노출될 수 있다. 역공학으로 소프트웨어를 해석하여 프로그램의 구조, 알고리즘 그리고 로직을 파악하고 시리얼 키를 알아낼 수 있으므로 보안에 주의가 필요하다. 이러한 위험에서 소프트웨어를 보호할 수 있는 기술이 프로그램의 구조와 코드를 읽기 어렵게 만드는 난독화 기술이다. 해당 발명에서는 소프트웨어를 안전하게 보호하기 위한 바이너리 프로그램 난독화 도구를 제안한다. 바이너리 프로그램의 플랫폼에 독립적이고, 프로그

램의 복잡도를 높일 수 있는 난독화 도구를 제안한다.

[0021] 먼저, 디컴파일러를 이용하여 바이너리 프로그램을 LLVM 중간언어로 변환한다. 변환된 중간언어 레벨의 프로그램을 난독화 한다. 중간언어를 난독화할 때에 난독화의 복잡도를 높이기 위해서 피드백 방식으로 난독화를 중첩되게 적용하는데, 한 번 난독화된 프로그램을 다시 난독화해서 중첩된 난독화를 적용한다. 난독화된 중간언어 레벨의 프로그램을 다시 컴파일하여 바이너리 프로그램으로 변환한다. 이 과정을 통해 난독화된 바이너리 프로그램을 생성한다.

[0022] 본 발명은 바이너리 프로그램에 대하여 역공학을 예방하고, 보안 취약점을 줄일 수 있다. 프로그램의 산술연산 등을 난독화하기 때문에 프로그램의 주요한 구조를 이해하기 어려워진다. 또한, 바이너리 프로그램을 중간언어로 디컴파일할 때에 아키텍처나 컴파일러에 제약이 없으므로 다양한 프로그램에 활용할 수 있다. 중간언어 레벨에서 난독화할 때에 피드백 방식을 사용하여 프로그램을 좀 더 복잡하고 분석하기 어렵게 하기 때문에 프로그램을 안전하게 보호하는 효과를 기대할 수 있다.

[0023] 도 1은 본 발명의 일 실시예에 따른 중간언어를 활용한 바이너리 프로그램 난독화 시스템의 구성을 나타낸 도면이다. 도 1에 도시된 바와 같이, 중간언어를 활용한 바이너리 프로그램 난독화 시스템(10)은 옵션 선택부(100), 모드 결정부(200), 난독화 실행부(300), 피드백부(400), 제어부(500)를 포함한다.

[0024] 옵션 선택부(100)는 난독화 옵션을 선택하는 구성이다. 이러한 옵션 선택부는 적어도 어느 하나 이상의 난독화 옵션을 선택받을 수 있다. 즉, 난독화 옵션 중 입력된 대상에 적용하고자 하는 난독화 기술을 적어도 한 가지 이상 선택하며, 입력한 키워드의 종류와 순서에 따라 난독화가 실행되도록 입력되는 키워드를 통해 선택한다. 난독화 옵션으로는 Opaque predicate 난독화, MBA 난독화, 식별자 난독화, Type change 난독화, Inline Assembly를 활용한 난독화, 소스코드 수준에서 가독성을 떨어뜨리는 Uglyfier 난독화 중 어느 하나 이상을 난독화 옵션으로 선택한다.

[0025] 모드 결정부(200)는 선택된 난독화 옵션으로 난독화할 원본을 입력받고, 상기 원본의 프로그래밍 언어 레벨에 따라 난독화 모드를 결정한다. 이러한 기능을 수행하기 위한 모드 결정부는 제1 모드 결정모듈(210), 제2 모드 결정모듈(220)을 포함한다.

[0026] 제1 모드 결정모듈(210)은 난독화할 대상의 프로그래밍 언어가 기계어 또는 어셈블리어인 경우, 저급 언어 난독화 모드로 결정한다. 제2 모드 결정모듈(220)은 난독화할 대상의 프로그래밍 언어가 기계어 또는 어셈블리어가 아닌 경우, 고급 언어 난독화 모드로 결정한다.

[0027] 난독화 실행부(300)는 모드 결정부를 통해 결정되는 레벨이 저급 언어인 경우, 상기 원본의 저급 언어 프로그램을 상기 난독화 옵션을 수행한 뒤 난독화된 저급 언어 프로그램을 출력하고, 고급 언어인 경우, 상기 원본의 고급 언어 프로그램을 상기 난독화 옵션을 수행하여 난독화된 고급 언어 프로그램을 출력한다.

[0028] 본 실시예에 따른 난독화 실행부는 모드 결정부를 통해 결정되는 레벨이 저급 언어인 경우, 원본의 저급 언어 레벨을 중간 언어 레벨로 변환하고, 선택한 난독화 옵션에 따라 난독화를 적용한 후 난독화된 중간 언어 레벨의 프로그램을 컴파일하여 저급 언어 프로그램으로 변환하여 출력한다.

[0029] 피드백부(400)는 선택된 난독화 옵션 횟수에 따라 난독화 실행부의 출력을 순차적으로 난독화 실행부의 입력으로 피드백 전달하여 난독화가 중첩되게 적용하도록 한다.

[0030] 제어부(500)는 옵션 선택부(100), 모드 결정부(200), 난독화 실행부(300), 피드백부(400)의 기능을 제어한다.

[0031] 이러한 제어부 구성으로 중간언어를 활용한 바이너리 프로그램 난독화 방법을 도 25를 참조하여 설명하면 다음과 같다. 도 25는 본 발명의 일 실시예에 따른 중간언어를 활용한 바이너리 프로그램 난독화 방법을 나타낸 흐름도이다.

[0032] (a) 제어부가 난독화 옵션을 선택하는 단계, (b) 제어부가 선택된 난독화 옵션으로 난독화할 원본을 입력받고, 상기 원본의 프로그래밍 언어 레벨에 따라 난독화 모드를 결정하는 단계, (c) 상기 (b) 단계의 결정결과, 결정되는 레벨이 저급 언어인 경우, 상기 원본의 저급 언어 프로그램을 상기 난독화 옵션을 수행한 뒤 난독화된 저급 언어 프로그램을 출력하고, 고급 언어인 경우, 원본의 고급 언어 프로그램을 난독화 옵션을 수행하여 난독화된 고급 언어 프로그램을 출력하는 단계를 포함한다.

[0033] 이러한 (c) 단계 이후, 제어부가 옵션 선택부는 적어도 어느 하나 이상의 난독화 옵션을 선택받으며, 선택된 난독화 옵션 횟수에 따라 난독화 실행부의 출력을 순차적으로 난독화 실행부의 입력에 피드백 전달하여 난독화가

중첩되게 적용하도록 하는 단계를 포함한다.

[0034] 본 실시예에서 (a) 단계에서, 제어부가 난독화 옵션 중 입력된 대상에 적용하고자 하는 난독화 기술을 적어도 한 가지 이상 선택하며, 입력한 키워드의 종류와 순서에 따라 난독화가 실행되도록 입력되는 키워드를 통해 선택한다. 이러한 (a) 단계는 제어부가 옵션 선택부에서 Opaque predicate 난독화, MBA 난독화, 식별자 난독화, Type change 난독화, Inline Assembly를 활용한 난독화, 소스코드 수준에서 가독성을 떨어뜨리는 Uglyfier 난독화 중 어느 하나 이상을 난독화 옵션으로 선택입력하도록 한다.

[0035] 또한 (b) 단계에서, 제어부가 난독화할 대상의 프로그래밍 언어가 기계어 또는 어셈블리어인 경우, 저급 언어 난독화 모드로 결정하고, 난독화할 대상의 프로그래밍 언어가 기계어 또는 어셈블리어가 아닌 경우, 고급 언어 난독화 모드로 결정한다.

[0036] (c) 단계는 제어부가 모드 결정부를 통해 결정되는 레벨이 저급 언어인 경우, 원본의 저급 언어 레벨을 중간 언어 레벨로 변환하고, 선택한 난독화 옵션에 따라 난독화를 적용한 후 난독화된 중간 언어 레벨의 프로그램을 컴파일하여 저급 언어 프로그램으로 변환하도록 한다.

[0037] 본 발명의 일 실시예에 따른 중간언어를 활용한 바이너리 프로그램 난독화 시스템은 더 발전된 난독화 기술을 제공하기 위해 제안하는 난독화 도구 Obfus를 설명하기로 한다.

[0038] Obfus는 고급 레벨 난독화와 저급 레벨 난독화를 모두 제공한다. Obfus에 저급 언어 프로그램을 입력하면 난독화된 저급 언어 프로그램을 반환하고, 고급 언어 프로그램을 입력하면 난독화된 고급 언어 프로그램을 반환한다. 프로그래밍 언어의 레벨에 따라서 난독화에 사용된 도구와 난독화를 적용하는 방식이 다른데, 자세한 내용을 설명하면 다음과 같다.

[0039] -고급 언어 난독화

[0040] 본 실시예에서, 고급 언어의 난독화 기능은 입력된 고급 언어 프로그램을 난독화하여 난독화된 고급 언어 프로그램을 제공한다. 총 8가지의 난독화 기술을 제공하고 있어 사용자는 입력된 고급 언어 프로그램에 적용하고자 하는 난독화 기술을 선택하여 적용할 수 있다. 또한 Obfus는 난독화 시 이미 난독화를 거친 프로그램에 대해 다시 그 위에 난독화를 적용하는 방법을 사용했다. 이는 다른 난독화 기술과는 차별되는 점으로, 이를 통해 더욱 심화된 난독화를 수행할 수 있게 된다.

[0041] Obfus 고급 언어 난독화 기능은 8가지의 난독화 기술을 제공한다. 이러한 8가지 난독화 기술은 기존의 연구와 본 발명에서 새롭게 고안한 기술이 포함된다. 기존의 난독화 기술로는 Opaque predicate 난독화, MBA 난독화, 식별자 난독화(Rename Identifier)가 있고, 여기에 추가로 제안하는 난독화 기술로는 Type change 난독화와 Inline Assembly를 활용한 난독화, 소스코드 수준에서 가독성을 떨어뜨리는 Uglyfier가 있다.

[0042] Type change 난독화는 변수의 자료형을 변환하여 가독성을 떨어뜨리는 난독화 옵션이다. Inline Assembly를 활용한 난독화는 C 코드에 Inline Assembly를 삽입하는 난독화 옵션이다. Uglyfier는 띄어쓰기 및 엔터, 주석을 지워 소스코드를 한 눈에 들어오지 못하게 하는 난독화 옵션이다.

[0043] (표 1)

Option	Obfuscation function
MBA	MBA Expression
Invariant	Invariant Opaque Predicates
Contextual	Contextual Opaque Predicates
Dynamic	Dynamic Opaque Predicates
ASM	Inline Assembly
CTYPE	Type Change
Var	Rename Identifier
Ugly	Code Uglyfier

[0044]

[0045] 옵션의 키워드와 간략한 설명은 표 1과 같으며 더 확장해나갈 계획이다.

[0046] 표 1과 같이 각 난독화 방법들에 매치되는 키워드를 제공하고 있다. 사용자는 키워드를 통해 난독화 옵션 중 입력된 고급 언어 프로그램에 적용하고자 하는 난독화 기술을 한 가지 이상 선택하여 적용할 수 있다. 이때, 같은 기술을 여러 번 선택하는 것도 가능하다.

[0047] 입력한 키워드의 종류와 순서에 따라 [키워드에 따른 Listener 선택, 난독화 적용, 결과 프로그램 출력]의 과정이 반복적으로 진행된다. 난독화가 적용된 결과 프로그램은 다시 난독화가 적용될 대상이 된다. 이러한 피드백 과정이 통해 입력한 모든 키워드에 대해 반복적으로 이루어지며 난독화가 중첩된다.

[0048] 도 2는 본 발명의 일 실시예에 따른 중간언어를 활용한 바이너리 프로그램 난독화 시스템의 고급 언어 난독화 동작 과정을 나타낸 도면이다.

[0049] 도 2에 도시된 바와 같이, P(Original Program)는 사용자가 입력한 난독화를 적용하고자 하는 프로그램에 해당한다. P'(Obfuscated Program)는 난독화가 적용된 결과 프로그램이다. 사용자가 선택한 옵션에 해당하는 기술로 P에 대한 난독화가 이루어진다. 도 2는 여러 개의 옵션 중 옵션 1과 옵션 3을 선택한 경우를 나타내고 있다. 이 경우 난독화 시나리오는 다음과 같다. 입력된 원본 프로그램을 옵션 1에 해당하는 난독화 기술로 한번 난독화를 적용한다. 그 결과는 다시 입력으로 들어와서 옵션 3에 해당하는 기술로 두 번째 난독화를 적용한다. 난독화를 마치면 최종 결과 프로그램인 P'가 출력된다. 이때 출력되는 결과 프로그램 P'는 고급 언어 프로그램이다.

[0050] 이와 같이 Obfuscation Strategy에서 output을 다시 input으로 전달하고 난독화를 적용하는 것을 반복하며 전체적인 난독화를 중첩??강화하는 동작이 본 발명에서 개발한 시스템의 핵심 기능이다.

[0051] -저급 언어 난독화

[0052] 본 실시예에서, 저급 언어의 난독화 기능은 입력된 저급 언어 프로그램을 LLVM(Low Level Virtual Machine) 중간 언어로 변환한 뒤 난독화하고 다시 컴파일(compile)하여 난독화된 저급 언어 프로그램을 제공한다.

[0053] 여기서, LLVM은 컴파일러의 기반 구조이다. 프로그램을 컴파일 타임, 링크 타임, 런타임 상황에서 프로그램의 작성 언어에 상관없이 최적화를 구현할 수 있도록 한다. LLVM의 핵심은 중간 표현(IR)으로, 어셈블리어와 비슷한 저급 프로그래밍 언어이다.

[0054] 저급 언어난독화 기능에서 제공하는 난독화 기술로는 MBA 난독화가 있다.

[0055] Obfus 저급 언어 난독화 기능의 동작 과정은 다음과 같다. 먼저, decompiler RetDec을 이용하여 입력한 저급 언어 프로그램을 LLVM 중간 언어로 변환한다. 고급 언어 난독화 기능에서와 마찬가지로, 저급 언어 난독화 기능에서도 중간 언어를 난독화할 때에 난독화의 복잡도를 높이기 위해서 출력이 다시 입력으로 들어가는 피드백 방식으로 난독화를 중첩되게 적용한다. 입력한 옵션에 따라 난독화를 적용한 후 난독화된 중간 언어 레벨의 프로그램을 다시 compile하여 저급 언어 프로그램으로 변환한다.

[0056] 도 3은 본 발명의 일 실시예에 따른 중간언어를 활용한 바이너리 프로그램 난독화 시스템의 저급 언어 난독화 동작 과정을 설명하기 위한 도면이다.

[0057] 도 3에 도시된 바와 같이, 저급 언어 난독화 기능의 동작 과정을 살펴보면, P(Original Program)는 사용자가 입력한 난독화를 적용하고자 하는 프로그램에 해당한다. P'(Obfuscated Program)는 난독화가 적용된 결과 프로그램이다. 그리고 P와 P'는 모두 Binary Code로 이루어져 있다.

[0058] 도 3과 같이, P가 입력되면 이 프로그램은 decompile되어 LLVM 중간언어로 lifting 된다. 그리고 이 LLVM 중간언어가 input이 되어 여러 난독화 method가 수행되고, 난독화된 LLVM 중간언어가 output으로 반환된다. 난독화된 LLVM 중간언어는 다시 compile되어 Binary code로 변환된다. 이 과정을 반복하여 더 복잡한 결과물을 얻을 수 있으며, 최종 Binary code는 난독화가 적용된 결과 프로그램인 P'(Obfuscated Program)가 된다.

[0059] 다음으로, 본 발명의 일 실시예에 따른 중간언어를 활용한 바이너리 프로그램 난독화 시스템의 난독화 구현을 고급 언어와 저급 언어로 설명하기로 한다.

[0060] 먼저, 고급 언어의 난독화는 C 언어로 작성된 소스코드를 ANTLR 4(ANTLR (ANother Tool for Language Recognition), http://www.antlr.org)를 이용해서 파싱하고, Listener로 파스 트리를 순회하며 난독화를 진행하였다. 기존의 난독화 아이디어와 본 발명에서 고안한 난독화 아이디어를 입력 소스코드에 적용하는 기능을 Listener를 확장한 별도의 Listener 클래스로 제작하였다. 다양한 난독화 기법이 있으므로, 난독화 기법마다 Listener 클래스를 다르게 제작하였다. 사용자가 선택하는 옵션에 따라 그에 맞는 Listener를 동작하여 난독화를 적용하는 구조로 구현하였고, jar 파일로 만들어 도구를 실행할 수 있게 하였다.

[0061] 도 4는 본 발명의 일 실시예에 따른 중간언어를 활용한 바이너리 프로그램 난독화 시스템에서, 난독화 옵션 키워드의 입력선택을 설명하기 위한 도면이다. 도 4와 같이, Command Line Arguments를 통해 사용하고자 하는 옵션의 키워드를 입력함으로써 옵션을 선택할 수 있도록 구현하였다. 이를 통해 사용자는 난독화 기법의 선택과

그 적용 순서를 제어할 수 있다. C 언어에서 사용하는 Inline Assembly를 활용하여 고급 언어로 표현된 코드를 저급 언어로 변환하는 난독화 기법을 사용하는데, 이때 Assembly의 해독을 어렵게 만들기 위해 Opaque Predicates를 접목한 방법으로 구현하였다.

[0062] 다음으로, 저급 언어의 난독화는 입력된 바이너리 프로그램을 decompiler인 RetDec(RetDec (Retargetable Decompiler), https://retdec.com/)을 이용하여 중간 언어 레벨로 변환한 뒤 옵션에 대한 난독화를 진행하고, 다시 compile하여 결과로 보낸다. 입력한 중간 언어 프로그램을 난독화 해 주는 부분이 LLVM Pass로 구현되어 있다.

[0063] 중간 언어 프로그램을 해당 Pass를 적용하여 compile하면 a.out이라는 이름의 난독화된 프로그램이 나오게 되고, 사용자는 이 난독화된 Binary code program을 제공받는다.

[0064] 도 5는 본 발명의 일 실시예에 따른 중간언어를 활용한 바이너리 프로그램 난독화 시스템의 저급 언어 난독화 수행을 설명하기 위한 도면이다. 도 5와 같이, 저급 언어 난독화를 수행하는 Shell Script를 통해 입력한 옵션의 개수만큼 반복하여 난독화를 적용할 수 있게 구현하였다. 입력한 옵션의 개수를 count로 받고 그만큼 반복하며 옵션에 대한 난독화를 적용한다. 먼저 RetDec을 이용해 tartget 바이너리 프로그램을 target.ll파일로 decompile한다. 변화된 target.ll에 난독화를 수행하는 LLVM Pass인 libSkeletonPass.so를 통과시켜 난독화를 수행한다. LLVMIR 레벨에서 난독화되고 다시 compile된 결과물인 a.out 파일이 나오는데, 이를 target으로 다시 이동시켜 중첩 난독화가 가능하게 한다.

[0065] 다음은 본 실시예에 따른 중간언어를 활용한 바이너리 프로그램 난독화 시스템의 구현결과를 설명하기로 한다. 프로그램의 언어 레벨에 따라서 각기 다른 실험을 진행하였다. 먼저 고급 언어에 대한 난독화 실험을 수행 후 저급 언어에 대한 난독화를 진행하였다.

[0066] 도 6은 본 발명의 일 실시예에 따른 중간언어를 활용한 바이너리 프로그램 난독화 시스템의 난독화 전 코드를 나타낸 도면이다.

[0067] -고급 언어 난독화 구현결과

[0068] 제안한 도구를 통해 난독화된 결과를 비교해 보기 위해 두 가지 방법으로 실험을 진행했다. 도 6의 코드를 입력으로 넣고 각 실험에 대해 옵션을 다르게 하며 각각 어떤 결과가 나오는지를 볼 수 있다.

[0069] 1)Command Line Arguments : Var, MBA

[0070] 표 1의 Var, MBA 순서로 옵션을 설정하고 실행하였다. Var(식별자 난독화, Rename Identifier) 난독화를 적용한 코드에 MBA(Mixed Boolean Atrithmetic, 산술연산에 대해 논리 연산을 함께하는 표현) 옵션에 해당하는 MBA Expression을 이용한 난독화를 적용한 최종 출력 코드이다. 이미 변수를 식별하기 어려운 입력에 MBA Expression 난독화를 적용하여 더욱 의미를 이해하기 어렵다. 도 7은 Var-MBA 옵션 적용 후 결과를 나타낸 도면이다.

[0071] 도 8은 난독화 전 어셈블리 코드를 나타낸 도면이고, 도 9는 MBA 옵션 적용 후 어셈블리 코드를 나타낸 도면이다.

[0072] 도 8은 난독화 전 코드(도 6)를 compile한 어셈블리 코드이고 도 9는 MBA 난독화를 적용한 결과 파일을 compile해 생성된 어셈블리 코드이다. 모든 어셈블리 코드는 gcc의 o옵션을 사용하여 compile하였다. 두 그림 모두 도 6의 4번째 줄과 같은 실행을 하지만, 도 8의 코드는 간단하게 동작을 하는 것에 반해 도 9는 더하거나 빼는 등 다른 연산이 추가되어 도 8의 연산보다 이해하기 어렵게 난독화된 것을 확인할 수 있다.

[0073] 2)Command Line Arguments : Dynamic, MBA, Var, Ugly

[0074] 표 1의 Dynamic, MBA, Var, Ugly 순서로 옵션을 설정하고 실행하였다.

[0075] 도 10은 Dynamic, MBA 옵션 적용 후를 나타낸 도면이다. 도 10에 도시된 바와 같이, 도 6의 난독화 전 코드에 첫 번째 옵션인 Dynamic Opaque Predicates를 이용한 난독화와 두 번째 옵션인 MBA 옵션에 해당하는 MBA Expression을 이용한 난독화를 적용한 코드이다. 첫 번째 옵션의 난독화를 적용한 코드에 MBA Expression 이용한 난독화가 중첩 적용되었음을 볼 수 있다.

[0076] 도 11은 Dynamic-MBA-Var-Ugly 옵션 적용 후를 나타낸 도면이다. 도 11에서 난독화를 2회 적용한 도 10의 코드에 Var 옵션과 Ugly 옵션에 해당하는 난독화를 적용한 코드이다. 도 10의 코드에 식별자 난독화(Var)를 중첩 적

용하고, 코드의 여백을 삭제함으로써, 앞서 난독화를 적용한 코드가 더욱 식별하기 어려워진다.

[0077] 도 12는 도 10의 코드를 컴파일(compile)한 어셈블리 코드의 일부이다. Dynamic Opaque predicate을 사용한 난독화를 통해 어셈블리 수준에서 분기 명령이 추가되고 코드의 control flow를 복잡하게 하는 난독화가 적용되었다.

[0078] -저급 언어 난독화 구현결과

[0079] 저급 언어에서 난독화된 결과 또한 확인해보기 위해 다음의 방법으로 실험을 진행해 보았다. 도 13은 원본 코드를 컴파일한 난독화 전 코드를 나타낸 도면이다.

[0080] 도 13에 도시된 바와 같이, 실험에 이용한 원본 코드는 도 6의 코드이고, 해당 코드를 compile한 Binary program(도 13)을 대상으로 실험을 진행하였다. 해당 program에 산술 연산을 난독화하는 표현인 MBA로 난독화한 결과를 비교해 보았다.

[0081] 도 13이 입력으로 들어오면 decompiler를 통해 LLVM 중간 언어로 변환한다. 도 14는 변환한 LLVM 중간 언어에 MBA 난독화를 두 번 적용했을 때의 모습이다. 난독화 전 바이너리 코드는 도 6의 4번째 줄의 더하기 연산을 표현하기 위해서 add가 1회만 수행된다. 하지만 난독화된 도 14의 코드는 그 연산을 이해하기 힘들 정도로 복잡하게 연산이 수행되는 것을 확인할 수 있다.

[0082] 앞서 도 6 내지 도 14에서의 실험을 통해 한번 난독화를 진행하는 것보다 여러 번의 난독화를 중첩시키는 방법이 효과적임을 알 수 있었다. 도 8, 9, 12를 통해 본 도구의 난독화가 저급 레벨에서도 적용됨을 확인할 수 있다. 여러 번의 난독화를 중첩해서 적용할수록 원래 코드와의 유사도가 더 낮아지고, 어떤 옵션을 조합하느냐에 따라 같은 원본 코드에 대해서도 다양한 난독화 결과를 도출할 수 있다.

[0083] 또한 저급 언어에서도 도 13과 도 14를 보면, 도 13에서의 간단했던 binary 연산이 도 14와 같이 이해하기 어렵게 난독화된다. 이를 통해 저급 언어 또한 효과적으로 난독화되는 것을 확인할 수 있다. 실험을 통해 본 도구는 고급 언어와 저급 언어 모두에서 난독화가 가능하고, 한 번 난독화한 프로그램을 다시 난독화하는 중첩 방식으로 난독화한 결과가 더 복잡한 것을 확인하였다. 또한 옵션의 조합에 따라 다양한 난독화가 가능하므로 본 난독화 도구의 활용성이 높음을 알 수 있다.

[0084] 앞선 실험은 모두 Obfus의 web 기반 프로그램을 이용해서 수행되었다. 도 15는 Obfus의 web 기반 프로그램 중 고급 언어 난독화 페이지이다. 도 16은 Obfus의 web 기반 프로그램 중 저급 언어 난독화 페이지이다. 각각의 난독화 페이지는 사용자가 원하는 옵션을 선택하여 난독화를 수행할 수 있다. 도 17과 같이 난독화 수행 결과를 내려받을 수 있다.

[0085] 본 발명의 일 실시예에 따른 중간언어를 활용한 바이너리 프로그램 난독화 시스템은 소프트웨어 저작권 및 취약점 보호를 위한 새로운 난독화 도구의 개발을 목표로 하였다. 개발한 도구는 ANTLR와 LLVM을 사용하여 보호하기를 원하는 프로그램이 소스 코드인지, 바이너리 수준인지와 상관없이 난독화 적용이 가능하다. 또한, 자체 제작한 다양한 난독화 기술을 여러 조합과 횟수로 적용할 수 있다. 이전의 코드에 다음 난독화 기법이 중첩 적용되는 방식으로, 이를 거친 코드는 한 가지의 방법으로 난독화한 코드보다 더 분석하기 복잡하게 난독화 됨을 확인하였다. 이에 더불어 사용의 편의성을 위하여 웹 기반의 플랫폼을 제공하여 손쉽게 코드에 난독화를 적용할 수 있게 하였다. 따라서 본 실시예에서 개발한 도구를 통해 같은 기능을 제공하면서도 소프트웨어를 안전하게 보호할 수 있게 된다.

[0086] 참고로, 본 발명의 일 실시예에 따른 중간언어를 활용한 바이너리 프로그램 난독화 시스템은 고급 언어의 난독화 기능은 입력된 고급 언어 프로그램을 난독화하여 난독화된 고급 언어 프로그램을 제공하는데, 총 8가지의 난독화 기술을 제공하고 있어 입력된 고급 언어 프로그램에 적용하고자 하는 난독화 기술을 옵션으로 선택하여 적용할 수 있다. 또한 이미 난독화를 거친 프로그램에 다시 그 위에 난독화를 적용할 수 있다. 이러한 8가지 난독화 기술 중, 기존의 난독화 기술로는 Opaque predicate 난독화, MBA 난독화, 식별자 난독화(Rename Identifier)를 설명하기로 한다.

[0087] 1) Opaque predicate 난독화를 설명하면, 조건문의 한 방향만 실행되도록 설정하고 실행되지 않는 부분에 더미 코드를 삽입하는 난독화 방법이다. 프로그램 일부를 최적화하는 것을 어렵게 하고, 제어 흐름을 복잡하게 만듦으로써 프로그램 분석을 방해한다. Opaque Predicates는 3가지 유형이 있다.

[0088] 가. Invariant Opaque Predicates

[0089] Invariant Opaque Predicates는 $x^2 \geqq 0$처럼, x 자리에 어떠한 값이 들어오더라도 참 또는 거짓 중에 반드시 하나의 값만을 도출하는 식을 조건으로 작성하는 분기문이다. 실행되는 흐름에 해당하는 부분에 필요한 코드를 작성하고, 실행되지 않는 흐름에 해당하는 부분에 더미코드를 작성하여 난독화한다. 도 18은 Invariant Opaque Predicates을 설명한 그림이다.

[0090] 나. Contextual Opaque Predicates

[0091] Contextual Opaque Predicates는 어떠한 전제 조건(Prior Condition)하에 항상 같은 값을 도출하는 식을 조건으로 작성하는 분기문이다. Invariant Opaque Predicates와 마찬가지로, 실행 코드와 더미 코드를 각각의 부분에 작성하여 난독화한다. 도 19는 Contextual Opaque Predicates을 설명하기 위한 도면이다.

[0092] 다. Dynamic Opaque Predicates

[0093] Dynamic Opaque Predicates는 어떠한 경우이든지 같은 결과를 내는 분기문의 집합이다. 이를 이용하여 실행할 때마다 조건문의 흐름을 타는 방향은 다르지만, 결과는 항상 같게 되도록 하는 구조로 난독화를 적용한다. 도 20은 Dynamic Opaque Predicates을 설명하기 위한 도면이다.

[0094] 2) MBA 난독화를 설명하면, MBA(Mixed Boolean Arithmetic)는 산술 연산에 대하여 논리 연산을 함께하는 표현이다. 도 21은 원본(original)과 MBA 난독화(obfuscated)를 설명하기 위한 도면이다.

[0095] 도 21의 왼쪽과 같이 간단하게 간파될 수 있는 식에 대해, 도 21의 오른쪽과 같이 논리연산을 섞어 읽기 어렵게 하는 것이다.

[0096] 3) 식별자 난독화를 설명하면, 기존에 작성된 식별자를 변경하여 식별자의 의미로부터 제공되는 단서를 가리거나, 가독성을 떨어지게 만들어 사람이 알아보기 어렵게 하는 방법이다. 도 22의 오른쪽은 도 22의 왼쪽과 같은 코드에 대해 메서드 이름과 매개변수 이름에 대해 난독화한 것이다.

[0097] 한편, 저급 프로그래밍 언어와 고급 프로그래밍 언어에 대해 간략히 정리하면 다음과 같다.

[0098] 1) 저급 프로그래밍 언어

[0099] 저급 프로그래밍 언어란 컴퓨터가 이해하기 쉽게 작성된 프로그래밍 언어로, 일반적으로 기계어와 어셈블리어가 있다. 실행속도가 매우 빠르며, 다른 기계마다 다른 코드를 가진다. 저급 언어와 반대되는 것으로 고급 프로그래밍 언어가 있다. 도 23은 간단한 어셈블리어 프로그램이다.

[0100] 2) 고급 프로그래밍 언어

[0101] 고급 프로그래밍 언어란 사람이 이해하기 쉽게 작성된 프로그래밍 언어로서, 저급 프로그래밍 언어보다 가독성이 높고 다루기 간단하다는 장점이 있다. 컴파일러나 인터프리터에 의해 저급 프로그래밍 언어로 번역되어 실행된다. C 언어, 자바 등 대부분의 프로그래밍 언어들은 고급 언어에 속한다. 추상화의 정도는 얼마나 프로그래밍 언어가 높은 수준인지를 정의한다. 도 24는 간단한 C 언어 프로그램이다.

[0102] 이상에서 설명한 본 발명은 전술한 실시예 및 첨부된 도면에 의해 한정되는 것이 아니고, 본 발명의 기술적 사상을 벗어나지 않는 범위 내에서 여러 가지 치환, 변형 및 변경이 가능함은 당업자에게 명백할 것이다.

부호의 설명

[0103] 100 : 옵션 선택부

200 : 모드 결정부

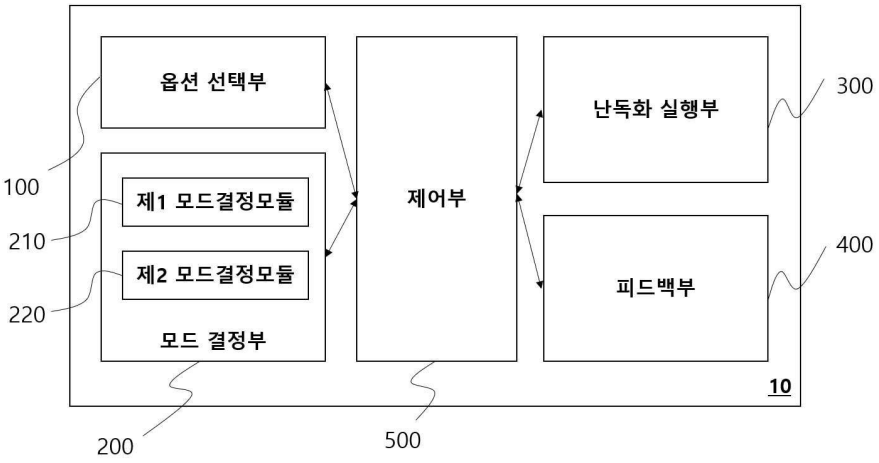
300 : 난독화 실행부

400 : 피드백부

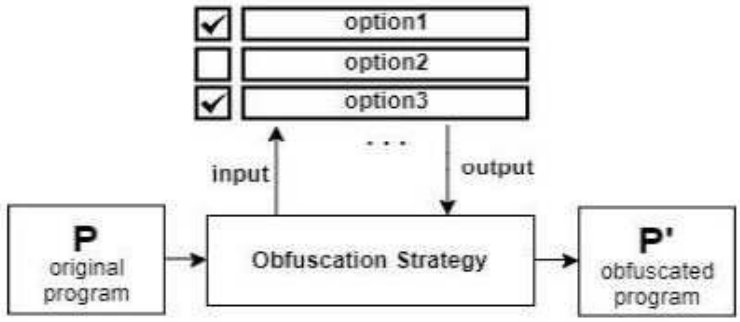
500 : 제어부

도면

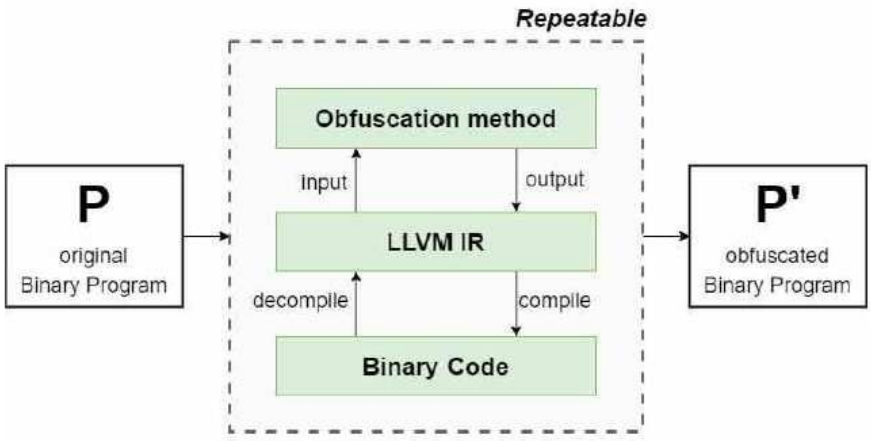
도면1



도면2



도면3



도면4

도면5

```
for((i=0; i<count; i++))
do
        retdec-decompiler.py target
        clang -Xclang -load -Xclang libSkeletonPass.so target.ll
        cp ./a.out target
done
```

도면6

```
1   int adder(int a, int b)
2   {
3       int result = 0;
4       result = a + b;
5       return result;
6   }
```

도면7

```
1  int _(int ___, int _____)  {
2    int ______ = 0;
3     ______ = (___ | _____) + _____ - (~___ & _____);
4    return ______;
5  }
```

도면8

```
00401410  r$ 55            PUSH EBP
00401411  .  89E5          MOV EBP,ESP
00401413  .  83EC 10       SUB ESP,10
00401416  .  C745 FC 00000 MOV DWORD PTR SS:[EBP-4],0
0040141D  .  8B55 08       MOV EDX,DWORD PTR SS:[EBP+8]
00401420  .  8B45 0C       MOV EAX,DWORD PTR SS:[EBP+C]
00401423  .  01D0          ADD EAX,EDX
00401425  .  8945 FC       MOV DWORD PTR SS:[EBP-4],EAX
00401428  .  8B45 FC       MOV EAX,DWORD PTR SS:[EBP-4]
0040142B  .  C9            LEAVE
0040142C  L. C3            RETN
```

도면9

```
0040141D  .  8B45 08       MOV EAX,DWORD PTR SS:[EBP+8]
00401420  .  0B45 0C       OR EAX,DWORD PTR SS:[EBP+C]
00401423  .  89C2          MOV EDX,EAX
00401425  .  8B45 0C       MOV EAX,DWORD PTR SS:[EBP+C]
00401428  .  01C2          ADD EDX,EAX
0040142A  .  8B45 08       MOV EAX,DWORD PTR SS:[EBP+8]
0040142D  .  F7D0          NOT EAX
0040142F  .  2345 0C       AND EAX,DWORD PTR SS:[EBP+C]
00401432  .  29C2          SUB EDX,EAX
00401434  .  89D0          MOV EAX,EDX
00401436  .  8945 FC       MOV DWORD PTR SS:[EBP-4],EAX
00401439  .  8B45 FC       MOV EAX,DWORD PTR SS:[EBP-4]
```

도면10

```
int adder(int a, int b)  {
int result = 0;
 result = (a | b) + b - (~a & b);
 int para = rand() % 9;
 if(para * para * para >= 0){
result = result-1;
}
else {
result = (result | 1) + 1 - (~result & 1);
}
 if(para * para * para < 0){
result = result-1;
}
else {
result = (result | 1) + 1 - (~result & 1);
}
 return result;
}
```

도면11

```
int _(int ___,int _____){int _____=0;_____=
(___|_____)+_____-(~___&_____);int _______=ra
nd()%9;if(_____*_____>=0){_____=_____-1
;}else {_____=(_____|1)+1-(~_____&1);}if(_
_____*_____<0){_____=_____-1;}else {____
__=(_____|1)+1-(~_____&1);}return _____;}
```

도면12

```
00401468  . 8B45 F0       MOV EAX,DWORD PTR SS:[EBP-10]
0040146B  . 0FAFC0        IMUL EAX,EAX
0040146E  . 0FAF45 F0     IMUL EAX,DWORD PTR SS:[EBP-10]
00401472  . 85C0          TEST EAX,EAX
00401474  . 78 06         JS SHORT result_C.0040147C
00401476  . 836D F4 01    SUB DWORD PTR SS:[EBP-C],1
0040147A  . EB 1E         JMP SHORT result_C.0040149A
0040147C  > 8B45 F4       MOV EAX,DWORD PTR SS:[EBP-C]
```

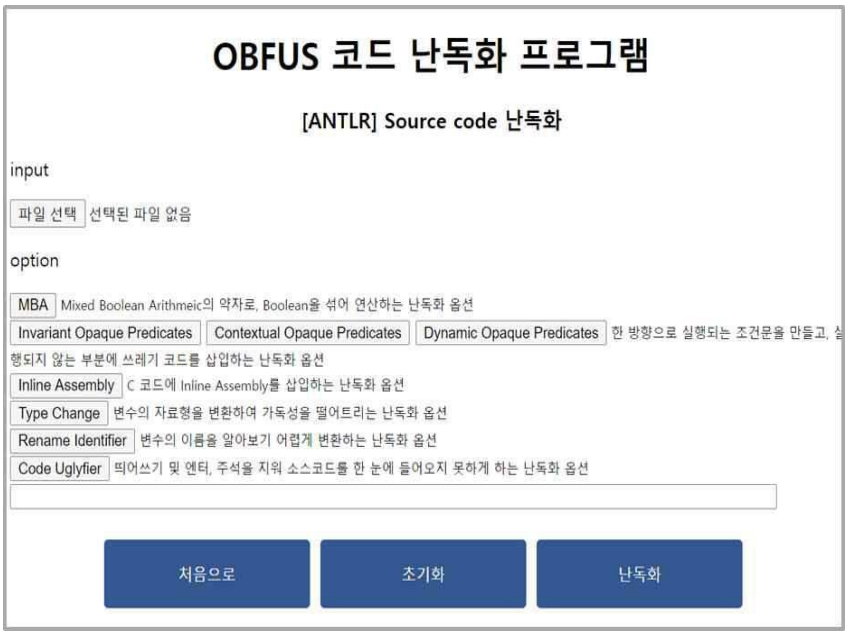
도면13

```
adder:
push    rbp
mov     rbp, rsp
mov     [rbp+var_4], edi
mov     [rbp+var_8], esi
mov     [rbp+var_C], 0
mov     esi, [rbp+var_4]
add     esi, [rbp+var_8]
mov     [rbp+var_C], esi
mov     eax, [rbp+var_C]
pop     rbp
retn
```

도면14

```
adder:
mov     rax, 100000000h
xor     ecx, ecx
mov     edx, ecx
mov     r8, rsi
or      r8, rdi
mov     r9, rsi
xor     r9, 0FFFFFFFFFFFFFFFFh
and     r9, rsi
sub     r8, rdi
add     r8, r9
mov     r9, rdx
sub     r9, rsi
sub     r9, 1
mov     rsi, r9
xor     rsi, 0FFFFFFFFFFFFFFFFh
or      rsi, rdi
xor     r9, 0FFFFFFFFFFFFFFFFh
sub     rsi, r9
mov     r9, r8
or      r9, rdi
add     r9, rdi
xor     r8, 0FFFFFFFFFFFFFFFFh
and     r8, rdi
sub     r9, r8
sub     r9, rsi
mov     r8, r9
xor     r8, rdi
mov     r10, r9
xor     r10, 0FFFFFFFFFFFFFFFFh
and     r10, rdi
add     r8, rdi
sub     r8, r10
sub     rdx, r9
sub     rdx, 1
mov     r9, rdx
xor     r9, 0FFFFFFFFFFFFFFFFh
or      r9, rdi
xor     rdx, 0FFFFFFFFFFFFFFFFh
sub     r9, rdx
mov     rdx, rdi
or      rdx, rax
add     rdx, rax
xor     rdi, 0FFFFFFFFFFFFFFFFh
and     rdi, rax
sub     rdx, rdi
sub     rdx, rsi
sub     rdx, r9
mov     rax, rdx
or      rax, r8
add     rax, r8
xor     rdx, 0FFFFFFFFFFFFFFFFh
and     rdx, r8
sub     rax, rdx
retn
```

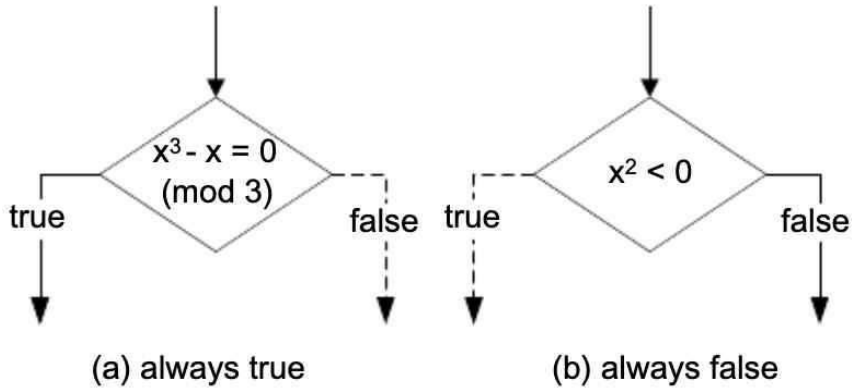
도면15



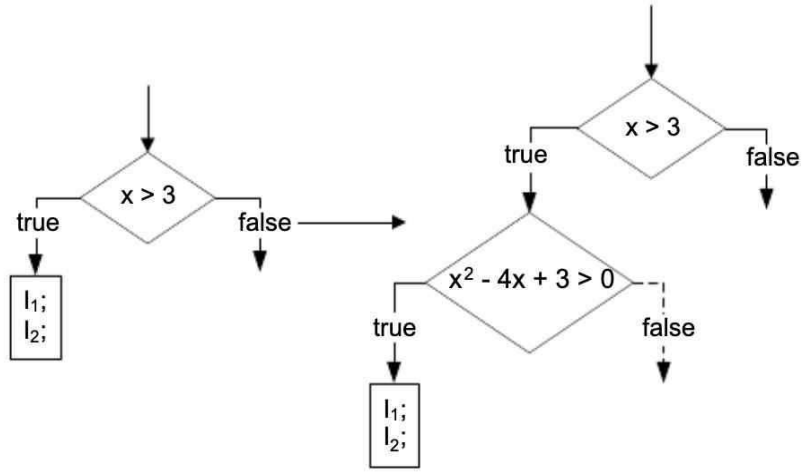
도면16

도면17

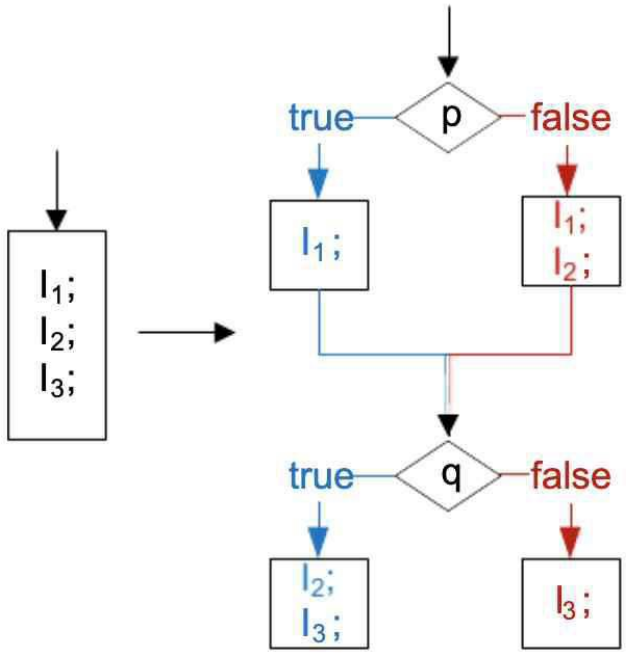
도면18


$x^3 - x = 0$
(mod 3)
true
false
(a) always true
$x^2 < 0$
true
false
(b) always false

도면19


x > 3
true
false
I_1;
I_2;
x > 3
true
false
$x^2 - 4x + 3 > 0$
true
false
I_1;
I_2;

도면20



도면21

<Original> <Obfuscated>

$$(A + B) \oplus A \Rightarrow \begin{array}{l}\Big(((A \vee B) + (A \wedge B)) \wedge A\Big) \\ -(((A \vee B) + (A \wedge B)) \vee A)\end{array}$$

도면22

<Original> <Obfuscated>

```
gcd(a, b){                    ⇒   func(_, __){
  if(b)                             if(__)
    return a;                         return _;
  else                              else
    return gcd(b, a%b);               return func(__, _%__);
}                                 }
```

도면23

```
adosseg
.model small
.stack 100h

.data
hello_message db 'Hello, World!',0dh,0ah,'$'

.code
main proc
      mov    ax, @data
      mov    ds, ax

      mov    ah, 9
      mov    dx, offset hello_message
      int    21h

      mov    ax, 4C00h
      int    21h
main endp
end main
```

도면24

```
#include <stdio.h>
int main ()
{
    printf("Hello world");
}
```

도면25