(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2016-0100887
(43) 공개일자 2016년08월24일

(51) 국제특허분류(Int. Cl.)
G06F 21/56 (2013.01)
(52) CPC특허분류
G06F 21/563 (2013.01)
(21) 출원번호 10-2016-0102941(분할)
(22) 출원일자 2016년08월12일
심사청구일자 2016년08월12일
(62) 원출원 특허 10-2014-0192799
원출원일자 2014년12월30일
심사청구일자 2014년12월30일

(71) 출원인
충남대학교산학협력단
대전광역시 유성구 대학로 99 (궁동, 충남대학교)
(72) 발명자
김형식
대전광역시 서구 둔산로 155 크로바아파트 113동 806호
조제경
대전광역시 유성구 대학로 99 충남대학교 공대5호관 630호
(뒷면에 계속)
(74) 대리인
권혁수, 송윤호

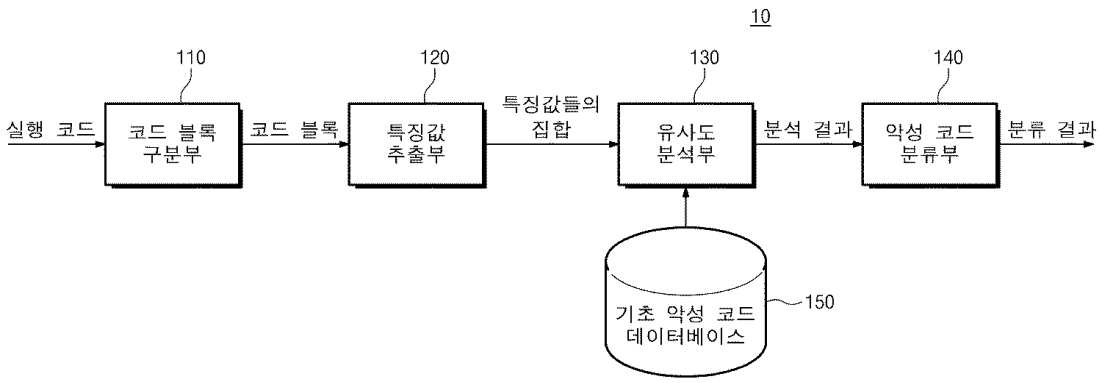
전체 청구항 수 : 총 4 항

(54) 발명의 명칭 **코드 블록 비교를 통한 악성 코드 탐지 방법**

(57) 요 약

본 발명은 악성 코드 탐지 방법 및 장치에 관한 것으로, 보다 구체적으로 코드 블록 비교를 통하여 악성 코드를 탐지하는 방법 및 장치에 관한 것이다. 본 발명의 일 실시 예에 따른 악성 코드 탐지 방법은, 정보 처리 장치가 데이터베이스에 저장된 기초 악성 코드에 관한 정보를 이용하여 악성 코드를 탐지하는 방법으로서, 실행 코드를 적어도 하나의 코드 블록으로 구분하는 단계; 상기 코드 블록으로부터 특징값을 추출하는 단계; 상기 특징값을 기반으로 상기 기초 악성 코드에 대한 상기 실행 코드의 유사도를 분석하는 단계; 및 상기 유사도에 따라 상기 실행 코드를 상기 기초 악성 코드에 속하는 악성 코드로 분류하는 단계;를 포함할 수 있다.

대 표 도



(72) 발명자

류재철

대전광역시 유성구 어은로 57 한빛아파트 132-801

박해룡

인천광역시 연수구 청능대로 175 연수우성1차아파트 112동 1701호

이 발명을 지원한 국가연구개발사업

과제고유번호 1711021897

부처명 미래창조과학부

연구관리전문기관 한국산업기술평가관리원

연구사업명 방송통신연구개발사업

연구과제명 사이버 공격의 사전 사후 대응을 위한 사이버 블랙박스 및 통합 사이버보안 상황분석 기술 개발

기 여 율 1/1

주관기관 한국인터넷진흥원

연구기간 2014.04.01 ~ 2017.02.28

명 세 서

청구범위

청구항 1

정보 처리 장치가 데이터베이스에 저장된 기초 악성 코드에 관한 정보를 이용하여 어셈블리 언어로 작성된 악성 코드를 탐지하는 방법에 있어서,

실행 코드를 복수 개의 코드 블록으로 구분하는 단계;

상기 코드 블록들 각각으로부터 코드를 추출하여, 추출된 코드의 변수 중에서 레지스터에 관한 변수만을 남겨두고 나머지 변수들을 삭제한 뒤 해당 코드의 해쉬값을 코드 블록의 해쉬값으로 획득하며, 각 코드 블록으로부터 얻은 해쉬값을 논리 연산하여 상기 실행 코드의 특징값을 획득하는 단계;

상기 코드 블록들 각각으로부터 추출된 특징값들의 집합을 기반으로 상기 기초 악성 코드에 대한 상기 실행 코드의 유사도를 분석하는 단계; 및

상기 유사도에 따라 상기 실행 코드를 상기 기초 악성 코드에 속하는 악성 코드로 분류하는 단계;

를 포함하고,

상기 코드 블록들 각각으로부터 추출된 특징값들의 집합을 기반으로 상기 기초 악성 코드에 대한 상기 실행 코드의 유사도를 분석하는 단계는:

상기 특징값들의 집합이 상기 기초 악성 코드에 대한 특징값에 대해 일치하는 정도를 기반으로 유사도를 계산하는 단계를 포함하고,

상기 유사도에 따라 상기 실행 코드를 상기 기초 악성 코드에 속하는 악성 코드로 분류하는 단계는:

상기 유사도가 기 설정된 기준치 이상인 경우, 해당 실행 코드를 상기 기초 악성 코드에 속하는 악성 코드로 결정하는 단계를 포함하는 악성 코드 탐지 방법.

청구항 2

제 1 항에 있어서,

상기 실행 코드를 코드 블록으로 구분하는 단계는:

상기 실행 코드에서 시작점 및 끝점을 지정하는 단계; 및

상기 시작점 및 상기 끝점을 경계로 상기 코드 블록을 정의하는 단계;

를 포함하는 악성 코드 탐지 방법.

청구항 3

제 2 항에 있어서,

상기 실행 코드에서 시작점 및 끝점을 지정하는 단계는:

상기 실행 코드 내 분기 명령어를 검출하는 단계; 및

상기 분기 명령어가 가리키는 지점을 상기 시작점으로 선언하는 단계;

를 더 포함하는 악성 코드 탐지 방법.

청구항 4

제 2 항에 있어서,

상기 시작점 및 상기 끝점을 경계로 상기 코드 블록을 정의하는 단계는:

상기 시작점부터 상기 끝점까지의 코드로 구성된 코드 블록을, 상기 시작점부터 상기 분기 명령어가 나타난 지

점까지의 코드로 구성된 코드 블록과, 상기 분기 명령어가 나타난 지점부터 상기 끝점까지의 코드로 구성된 코드 블록으로 분할하는 단계를 포함하는 악성 코드 탐지 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 악성 코드 탐지 방법에 관한 것으로, 보다 구체적으로 코드 블록 비교를 통하여 악성 코드를 탐지하는 방법에 관한 것이다.

배 경 기 술

[0002] 최근 사용자 시스템에 침투해 정보를 유출하거나 정상적인 동작을 지연 또는 방해하는 등 악의적인 목적을 위해 작성된 악성 코드가 해커나 악성 코드 제작자들에 의해 대량으로 빠르게 생성 및 유포되고 있다. 이들은 기존의 악성 코드를 수정하거나 모듈을 재사용하는 등 다양한 방법을 통해 악성 코드의 변종들을 제작하여 유포하고 있으며, 이에 따라 인터넷에서 발견되는 악성 코드의 개체 수는 지속적으로 증가하고 있는 추세이다.

[0003] 기존의 안티-바이러스 소프트웨어는 악성 코드 샘플 분석을 통해 획득한 시그니처를 탐지 엔진에 제공함으로써 변종 악성 코드도 탐지한다. 그러나, 악성 코드를 분석하여 안티-바이러스 엔진에 필요한 시그니처를 적시에 제공하기 위해서는 복잡한 단계를 거쳐야 한다.

[0004] 대부분의 악성 코드 분석 전문가가 다양한 분석 도구를 이용하여 악성 코드 샘플에 대한 리버스 엔지니어링을 통해 정적 분석을 수행하고, 악성 코드 내에 포함되어 있는 특정 문자열이나 패턴을 찾아서 해당 악성 코드를 탐지하기 위한 시그니처를 생성한다. 그러나, 이러한 정적 분석 과정은 악성 코드에 포함된 코드 블록을 명령어 수준에서 상세하게 분석해야 하기 때문에 분석에 많은 시간이 소요된다.

발명의 내용

해결하려는 과제

[0005] 본 발명의 실시 예는 기존 코드를 재사용하여 생성된 변종 악성 코드를 정확하게 탐지하고 분류할 수 있는 악성 코드 탐지 방법 및 장치를 제공하는 것을 목적으로 한다.

과제의 해결 수단

[0006] 상기 과제를 해결하기 위한 본 발명의 일 실시 예에 따른 악성 코드 탐지 방법은, 정보 처리 장치가 데이터베이스에 저장된 기초 악성 코드에 관한 정보를 이용하여 어셈블리 언어로 작성된 악성 코드를 탐지하도록 제공될 수 있다.

[0007] 상기 악성 코드 탐지 방법은, 실행 코드를 복수 개의 코드 블록으로 구분하는 단계; 상기 코드 블록들 각각으로부터 코드를 추출하여, 추출된 코드의 변수 중에서 레지스터에 관한 변수만을 남겨두고 나머지 변수들을 삭제한 뒤 해당 코드의 해쉬값을 코드 블록의 해쉬값으로 획득하며, 각 코드 블록으로부터 얻은 해쉬값을 논리 연산하여 상기 실행 코드의 특징값을 획득하는 단계; 상기 코드 블록들 각각으로부터 추출된 특징값들의 집합을 기반으로 상기 기초 악성 코드에 대한 상기 실행 코드의 유사도를 분석하는 단계; 및 상기 유사도에 따라 상기 실행 코드를 상기 기초 악성 코드에 속하는 악성 코드로 분류하는 단계를 포함할 수 있다.

[0008] 상기 코드 블록들 각각으로부터 추출된 특징값들의 집합을 기반으로 상기 기초 악성 코드에 대한 상기 실행 코드의 유사도를 분석하는 단계는: 상기 특징값들의 집합이 상기 기초 악성 코드에 대한 특징값에 대해 일치하는 정도를 기반으로 유사도를 계산하는 단계를 포함하고, 상기 유사도에 따라 상기 실행 코드를 상기 기초 악성 코드에 속하는 악성 코드로 분류하는 단계는: 상기 유사도가 기 설정된 기준치 이상인 경우, 해당 실행 코드를 상기 기초 악성 코드에 속하는 악성 코드로 결정하는 단계를 포함할 수 있다.

[0009] 상기 실행 코드를 코드 블록으로 구분하는 단계는: 상기 실행 코드에서 시작점 및 끝점을 지정하는 단계; 및 상기 시작점 및 상기 끝점을 경계로 상기 코드 블록을 정의하는 단계를 포함할 수 있다.

[0010] 상기 실행 코드에서 시작점 및 끝점을 지정하는 단계는: 상기 실행 코드 내 분기 명령어를 검출하는 단계; 및 상기 분기 명령어가 가리키는 지점을 상기 시작점으로 선언하는 단계를 더 포함할 수 있다.

[0011] 상기 시작점 및 상기 끝점을 경계로 상기 코드 블록을 정의하는 단계는: 상기 시작점부터 상기 끝점까지의 코드로 구성된 코드 블록을, 상기 시작점부터 상기 분기 명령어가 나타난 지점까지의 코드로 구성된 코드 블록과, 상기 분기 명령어가 나타난 지점부터 상기 끝점까지의 코드로 구성된 코드 블록으로 분할하는 단계를 포함할 수 있다.

발명의 효과

[0012] 본 발명의 실시 예에 따르면, 기존 코드를 재사용하여 생성된 변종 악성 코드를 정확하게 탐지하고 분류할 수 있다.

도면의 간단한 설명

[0013] 도 1은 본 발명의 일 실시 예에 따른 악성 코드 탐지 장치의 예시적인 블록도이다.

도 2는 본 발명의 일 실시 예에 따른 악성 코드 탐지 방법의 예시적인 흐름도이다.

도 3은 본 발명의 일 실시 예에 따라 실행 코드를 코드 블록으로 구분하는 과정을 설명하기 위한 예시적인 도면이다.

도 4는 본 발명의 일 실시 예에 따라 유사도를 분석하여 악성 코드를 탐지하는 과정을 설명하기 위한 예시적인 도면이다.

발명을 실시하기 위한 구체적인 내용

[0014] 본 발명의 실시예는 데이터베이스에 저장된 기초 악성 코드에 관한 정보를 이용하여 기초 악성 코드와 검사 대상인 실행 코드 간의 유사도를 분석함으로써 실행 코드가 기초 악성 코드에 속하는 변종 악성 코드인지 여부를 검사한다. 이 때, 본 발명의 실시예는 단순히 실행 코드로부터 함수를 추출하여 함수를 기반으로 기초 악성 코드와의 유사도를 분석하는 대신, 실행 코드를 적어도 하나의 코드 블록으로 구분한 뒤, 이 코드 블록 단위로 기초 악성 코드와 유사도를 분석하여 악성 코드를 탐지할 수 있다.

[0015] 그 결과, 본 발명의 실시예에 따르면, 기존의 함수 기반 악성 코드 탐지 프로그램이 제대로 탐지하지 못하던 코드, 예컨대 함수 형태가 아닌 어셈블리어로 직접 작성된 코드 또는 함수 형태지만 호출되지 않는 부분을 갖는 코드에 대해서도 악성 코드 여부를 정확하게 판별할 수 있다.

[0016] 이하, 본 명세서에 첨부된 도면을 참조하여 본 발명의 실시 예를 상세하게 설명한다.

[0017] 도 1은 본 발명의 일 실시 예에 따른 악성 코드 탐지 장치(10)의 예시적인 블록도이다.

[0018] 본 발명의 일 실시 예에 따른 악성 코드 탐지 장치(10)는 악성 코드를 탐지하기 위한 프로그램을 실행하여 검사 대상인 실행 코드가 악성 코드에 해당하는지 여부를 판별할 수 있다. 상기 악성 코드 탐지 장치(10)는 컴퓨터와 같은 정보 처리 장치로서, HDD, SSD 등과 같은 저장 장치로부터 악성 코드 탐지를 위한 프로그램을 불러와 실행할 수 있는 프로세서, 예컨대 CPU 등을 포함할 수 있다.

[0019] 도 1에 도시된 바와 같이, 상기 악성 코드 탐지 장치(10)는 코드 블록 구분부(110), 특징값 추출부(120), 유사도 분석부(130) 및 악성 코드 분류부(140)를 포함할 수 있다.

[0020] 상기 코드 블록 구분부(110)는 실행 코드를 적어도 하나의 코드 블록으로 구분할 수 있다. 상기 특징값 추출부(120)는 상기 코드 블록으로부터 특징값을 추출할 수 있다. 상기 유사도 분석부(130)는 상기 특징값을 기반으로 기초 악성 코드에 대한 상기 실행 코드의 유사도를 분석할 수 있다. 상기 악성 코드 분류부(140)는 분석 결과인 상기 유사도에 따라 상기 실행 코드를 상기 기초 악성 코드에 속하는 악성 코드로 분류할 수 있다.

[0021] 상기 유사도 분석부(130)는 유사도 분석 과정에서 데이터베이스(150)에 저장되어 있는 기초 악성 코드에 관한 정보를 불러와 해당 정보를 이용하여 상기 실행 코드와의 유사도를 분석할 수 있다.

[0022] 도 2는 본 발명의 일 실시예에 따른 악성 코드 탐지 방법(20)의 예시적인 흐름도이다.

[0023] 상기 악성 코드 탐지 방법(20)은 본 발명의 실시예에 따른 악성 코드 탐지 장치(10)에 의해 수행될 수 있다.

[0024] 도 2에 도시된 바와 같이, 상기 악성 코드 탐지 방법(20)은 정보 처리 장치가 데이터베이스(150)에 저장된 기초 악성 코드에 관한 정보를 이용하여 악성 코드를 탐지하는 방법으로서, 실행 코드를 적어도 하나의 코드 블록으로 구분하는 단계(S210), 상기 코드 블록으로부터 특징값을 추출하는 단계(S220), 상기 특징값을 기반으로 상기

기초 악성 코드에 대한 상기 실행 코드의 유사도를 분석하는 단계(S230), 및 상기 유사도에 따라 상기 실행 코드를 상기 기초 악성 코드에 속하는 악성 코드로 분류하는 단계(S240)를 포함할 수 있다.

[0025] 일 실시예에 따르면, 상기 실행 코드는 어셈블리 언어로 작성된 코드일 수 있다. 예를 들어, 상기 실행 코드는 Windows의 x86 플랫폼으로 작성된 어셈블리 코드일 수 있으나, 상기 실행 코드의 포맷은 이에 제한되지 않는다.

[0026] 본 발명의 실시예는 실행 코드가 악성 코드에 해당하는지 여부를 검사하기 위한 사전 단계로 상기 실행 코드를 적어도 하나의 코드 블록으로 구분할 수 있다.

[0027] 도 3은 본 발명의 일 실시 예에 따라 실행 코드를 코드 블록으로 구분하는 과정을 설명하기 위한 예시적인 도면이다.

[0028] 본 발명의 일 실시예에 따르면, 상기 실행 코드를 적어도 하나의 코드 블록으로 구분하는 단계(S210)는, 상기 실행 코드에서 시작점 및 끝점을 지정하는 단계, 및 상기 시작점 및 상기 끝점을 경계로 상기 코드 블록을 정의하는 단계를 포함할 수 있다.

[0029] 예를 들어, 도 3을 참조하면, 상기 실행 코드에서 시작점 및 끝점을 지정하는 단계는, 상기 실행 코드 내에서 "push" 명령어 및 "move" 명령어 중 적어도 하나를 검출하여 시작점(①, ③, ⑥)으로 선언하는 단계, 및 상기 실행 코드 내에서 "ret" 명령어 및 "align" 명령어 중 적어도 하나를 검출하여 끝점(②, ⑤, ⑧)으로 선언하는 단계를 포함할 수 있다.

[0030] 구체적으로, 상기 실행 코드에서 시작점으로 선언되는 지점은 상기 실행 코드 내 "push ebp" 또는 "move ebp, esp"가 표시되는 라인일 수 있으며, "push ebp"와 "move ebp, esp"가 연속으로 표시되는 경우에는 두 라인들 중 먼저 나타나는 라인이 시작점으로 선언될 수 있다.

[0031] 마찬가지로, 상기 실행 코드에서 끝점으로 선언되는 지점은 상기 실행 코드 내 "ret" 또는 "align"이 표시되는 라인일 수 있으며, "ret"와 "align"이 연속으로 표시되는 경우에는 두 라인들 중 먼저 나타나는 라인이 끝점으로 선언될 수 있다.

[0032] 추가적으로, 상기 실행 코드에서 시작점 및 끝점을 지정하는 단계는, 상기 실행 코드 내에서 분기 명령어를 검출하는 단계, 및 상기 분기 명령어가 가리키는 지점을 상기 시작점으로 선언하는 단계를 더 포함할 수 있다.

[0033] 이 실시예에 따르면, 상기 분기 명령어는 실행 코드 내 다른 영역의 코드를 호출하기 위한 명령어로서, 예를 들어 "call", "jmp", "jnz", "jz" 및 "jne" 중 적어도 하나를 포함할 수 있다.

[0034] 이 경우, 상기 분기 명령어가 가리키는 지점, 즉 분기 명령어 다음에 표시되는 어드레스 값에 해당하는 지점이 또 다른 시작점으로 선언될 수 있다.

[0035] 예를 들어, 도 3과 같이 실행 코드의 B 부분에 "call C"라는 명령어가 포함되어 있는 경우, "call"에 의해 호출되는 C 부분이 또 다른 시작점으로 선언될 수 있다. 마찬가지로, 도 3과 같이 실행 코드의 D 부분에 "jmp E"라는 명령어가 포함되어 있는 경우, "jmp"에 의해 지정되는 E 부분이 또 다른 시작점으로 선언될 수 있다.

[0036] 본 발명의 실시예에 따르면, 상기 실행 코드를 코드 블록으로 구분하는 단계(S210)는, 상기 실행 코드에서 시작점과 끝점 사이에 분기 명령어가 포함된 경우, 상기 시작점부터 상기 끝점까지의 코드로 구성된 코드 블록을 상기 분기 명령어가 나타난 지점을 경계로 분할하는 단계를 더 포함할 수 있다.

[0037] 예를 들어, 도 3에 도시된 바와 같이, 상기 실행 코드의 B 부분에 포함된 시작점인 "push ebp"(③)와 끝점인 "ret"(⑤) 사이에 분기 명령어인 "call C"(④)가 포함된 경우, 상기 시작점(③)부터 상기 끝점(⑤)까지의 코드로 구성된 코드 블록(즉, 실행 코드의 B 부분)을 상기 분기 명령어가 나타난 지점(④)을 경계로 소분할할 수 있다.

[0038] 구체적으로, 상기 코드 블록을 분할하는 단계는, 상기 시작점(③)부터 상기 끝점(⑤)까지의 코드로 구성된 코드 블록(실행 코드의 B 부분)을, 상기 시작점(③)부터 상기 분기 명령어가 나타난 지점(④)까지의 코드로 구성된 코드 블록 2-1과, 상기 분기 명령어가 나타난 지점(④)부터 상기 끝점(⑤)까지의 코드로 구성된 코드 블록 2-2으로 분할하는 단계를 포함할 수 있다.

[0039] 마찬가지로, 도 3과 같이 상기 실행 코드의 D 부분에 포함된 시작점인 "push ebp"(⑥)와 끝점인 "align"(⑧) 사이에 분기 명령어인 "jmp E"(⑦)가 포함된 경우, 상기 시작점(⑥)부터 상기 끝점(⑧)까지의 코드로 구성된 코드 블록(실행 코드의 D 부분)을, 상기 시작점(⑥)부터 상기 분기 명령어가 나타난 지점(⑦)까지의 코드로 구성된

코드 블록 4-1과, 상기 분기 명령어가 나타난 지점(⑦)부터 상기 끝점(⑧)까지의 코드로 구성된 코드 블록 4-2로 분할하는 단계를 포함할 수 있다.

[0040] 위와 같이 코드 블록 구분부(110)가 실행 코드를 코드 블록으로 구분하면, 특징값 추출부(120)는 해당 코드 블록의 특징값을 추출할 수 있다.

[0041] 상기 특징값 추출부(120)는 우선 상기 추출된 코드의 변수중에서 레지스터에 관한 변수만을 남겨두고 나머지 변수들을 삭제한 뒤, 해당 코드의 해쉬값을 획득한다. 여기서, 상기 레지스터에 관한 변수는 "eax", "ecx", "ebp", "ebx, "edi"와 같은 레지스터를 나타낼 수 있다.

[0042] 상기 특징값 추출부(120)는 각각의 코드 블록으로부터 얻은 해쉬값을 논리 연산하여 상기 실행 코드의 특징값을 획득할 수 있다. 즉, 상기 특징값 추출부(120)는 상기 실행 코드에 포함된 각 코드 블록의 해쉬값을 논리 연산함으로써 최종적으로 상기 실행 코드로부터 하나의 특징값을 추출할 수 있다.

[0043] 각 코드 블록으로부터 특징값을 추출하면, 상기 유사도 분석부(130)는 상기 특징값들의 집합을 상기 데이터베이스(150)에 저장된 기초 악성 코드에 대한 특징값과 비교하여 그 일치 여부를 판별함으로써 기초 악성 코드에 대한 실행 코드의 유사도를 분석할 수 있다.

[0044] 이때 상기 유사도 분석부(130)는 특징값들의 집합에 속하는 특징값들의 일부만 상기 데이터베이스(150)에 저장된 기초 악성 코드에 대한 특징값과 일치하더라도 상기 기초 악성 코드와 상기 실행 코드가 유사하다고 판단할 수 있다.

[0045] 상기 유사도 분석부(130)는 주어진 특징값들의 집합에 대하여 비교 대상 기초 악성 코드에 대한 특징값과 일치하는 특징값들의 비율을 계산하고, 계산된 비율을 미리 정해진 임계값과 비교할 수 있다.

[0046] 상기 데이터베이스(150)는 기초 악성 코드에 관한 정보, 예컨대 특징값을 저장하고 있으며, 상기 유사도 분석부(130)는 상기 데이터베이스(150)로부터 상기 기초 악성 코드의 특징값을 불러와 상기 실행 코드의 유사도 분석에 사용할 수 있다.

[0047] 그러고 나서, 상기 악성 코드 분류부(140)는 상기 유사도에 따라 실행 코드를 기초 악성 코드에 속하는 악성 코드로 분류할 수 있다. 예를 들어, 상기 악성 코드 분류부(140)는 실행 코드의 상기 기초 악성 코드에 대한 유사도가 기 설정된 기준치 이상인 경우, 상기 실행 코드를 해당 기초 악성 코드에 속하는 변종 악성 코드로 결정할 수 있다.

[0048] 도 4는 본 발명의 일 실시예에 따라 유사도를 분석하여 악성 코드를 탐지하는 과정을 설명하기 위한 예시적인 도면이다.

[0049] 도 4를 참조하면, 실행 코드 C와 비교되는 기초 악성 코드는 악성 코드 A와 악성 코드 B이며, 악성 코드 A로부터 얻은 코드 블록 테이블(tbl1)에는 복수의 코드 블록들(A-1, A-2, A-3, A-4)이 포함되어 있고, 악성 코드 B로부터 얻은 코드 블록 테이블(tbl2)에는 복수의 코드 블록들(B-1, B-2, B-3, B-4)이 포함되어 있다.

[0050] 그리고, 데이터베이스에는 악성 코드 A와 악성 코드 B로부터 얻은 각 코드 블록의 특징값이 저장되어 있다.

[0051] 본 발명의 실시예에 따르면, 상기 실행 코드 C가 악성 코드인지 여부를 검사하기 위해, 먼저 상기 코드 블록 구분부(110)가 상기 실행 코드 C를 적어도 하나의 코드 블록으로 구분한다. 그러고 나서, 상기 특징값 추출부(120)는 상기 실행 코드 C의 각 코드 블록으로부터 특징값(특징값 C-1, 특징값 C-2, 특징값 C-3, 특징값 C-4)을 획득할 수 있다.

[0052] 그러고 나서, 상기 유사도 분석부(130)는 추출된 특징값과 데이터베이스(150)에 저장된 기초 악성 코드들의 특징값을 비교하여 상기 실행 코드 C의 코드 블록과 유사한 기초 코드 블록을 결정할 수 있다.

[0053] 도 4에 도시된 분석 테이블(tbl4)을 참조하면, 기초 악성 코드 A의 특징값 A-3와 실행 코드 C의 특징값 C-1이 유사하고, 기초 악성 코드 A의 특징값 A-4와 실행 코드 C의 특징값 C-2가 유사하다. 또한, 기초 악성 코드 B의 특징값 B-2와 실행 코드 C의 특징값 C-4가 유사하다. 실행 코드 C의 특징값 C-3와 유사한 기초 악성 코드의 특징값은 발견되지 않았다. 그 결과, 상기 유사도 분석부(130)는 상기 실행 코드 C에 포함된 코드 블록과 유사한 기초 악성 코드의 코드 블록으로 코드 블록 A-3, A-4 그리고 B-2를 선택할 수 있다.

[0054] 그러고 나서, 상기 악성 코드 분류부(140)는 상기 유사도 분석부(130)가 분석하여 얻은 상기 실행 코드의 상기 기초 악성 코드들에 대한 유사도를 기반으로, 상기 기초 악성 코드들 중에서 상기 실행 코드에 대응하는 기초

악성 코드를 선택하여 상기 실행 코드를 해당 기초 악성 코드에 속하는 변종 악성 코드로 분류할 수 있다.

[0055] 일 실시예에 따르면, 상기 악성 코드 분류부(140)는 상기 실행 코드의 코드 블록과 유사한 상기 기초 악성 코드의 코드 블록 개수에 따라 상기 실행 코드에 대응하는 기초 악성 코드를 선택할 수 있다.

[0056] 예를 들어, 도 4를 참조하면, 기초 악성 코드 A 및 B 중에서 실행 코드 C의 코드 블록들(C-1, C-2, C-3, C-4)과 유사한 코드 블록을 가장 많이 포함하고 있는 코드는 기초 악성 코드 A이며, 상기 악성 코드 분류부(140)는 상기 기초 악성 코드 A 및 B 중에서 상기 실행 코드 C의 코드 블록과 유사한 코드 블록을 가장 많이 포함하고 있는 기초 악성 코드 A를 상기 실행 코드에 대응하는 기초 악성 코드로 선택할 수 있다.

[0057] 그 결과, 도 4의 코드 그룹 A와 같이, 상기 악성 코드 분류부(140)는 상기 실행 코드 C를 상기 기초 악성 코드 A에 속하는 악성 코드로 분류할 수 있다.

[0058] 실시예에 따라, 상기 악성 코드 분류부(140)는 상기 실행 코드의 코드 블록 내 코드에 의한 동작이 기 결정된 주의 동작에 해당하는지 여부를 판별하고, 판별 결과에 따라 상기 실행 코드의 악성 코드 여부를 결정할 수 있다.

[0059] 구체적으로, 상기 악성 코드 분류부(140)는 상기 실행 코드의 코드 블록 내 코드에 의한 동작이 상기 주의 동작에 해당하는 경우 상기 실행 코드를 악성 코드로 분류하고, 상기 주의 동작에 해당하지 않는 경우에는 상기 실행 코드를 악성 코드로 분류하지 않을 수 있다.

[0060] 이 실시예에서, 상기 주의 동작은 시스템의 성능에 영향을 미칠 수 있는 동작으로서, 파일에 접근하는 동작, 레지스트리에 접근하는 동작, 및 네트워크에 접근하는 동작 중 적어도 하나를 포함할 수 있다.

[0061] 상기 파일에 접근하는 동작은 시스템에 저장된 파일을 열거나, 수정하거나, 삭제하는 행위로서, 상기 실행 코드의 코드 블록에 "open", "delete" 등의 명령어가 포함된 경우, 해당 코드 블록 내 코드는 파일에 접근하는 동작을 수행하는 것으로 판별될 수 있다.

[0062] 상기 레지스트리에 접근하는 동작은 시스템의 레지스트리에 저장된 정보를 열람하거나, 수정하거나, 삭제하는 행위로서, 상기 실행 코드의 코드 블록에 "registry key open" 등의 명령어가 포함된 경우, 해당 코드 블록 내 코드는 레지스트리에 접근하는 동작을 수행하는 것으로 판별될 수 있다.

[0063] 상기 네트워크에 접근하는 동작은 시스템에 연결된 네트워크를 통해 트래픽을 유발하는 행위로서, 상기 실행 코드의 코드 블록에 "recv", "send", "send to" 등의 명령어가 포함된 경우, 해당 코드 블록 내 코드는 네트워크에 접근하는 동작을 수행하는 것으로 판별될 수 있다.

[0064] 이와 같이, 본 발명의 실시 예는 실행 코드의 코드 블록으로부터 추출된 특징값을 기초 악성 코드의 코드 블록으로부터 추출된 특징값과 비교하여 기초 악성 코드에 대한 실행 코드의 유사도를 분석하는 과정 외에, 실행 코드의 코드 블록 내 코드에 의한 동작을 더 분석함으로써 악성 코드 탐지 정확도를 보다 향상시킬 수 있다.

[0065] 이상에서 실시 예를 통해 본 발명을 설명하였으나, 위 실시 예는 단지 본 발명의 사상을 설명하기 위한 것으로 이에 한정되지 않는다. 통상의 기술자는 전술한 실시예에 다양한 변형이 가해질 수 있음을 이해할 것이다. 본 발명의 범위는 첨부된 특허청구범위의 해석을 통해서만 정해진다.

부호의 설명

[0066] 10: 악성 코드 탐지 장치

110: 코드 블록 구분부

120: 특징값 추출부

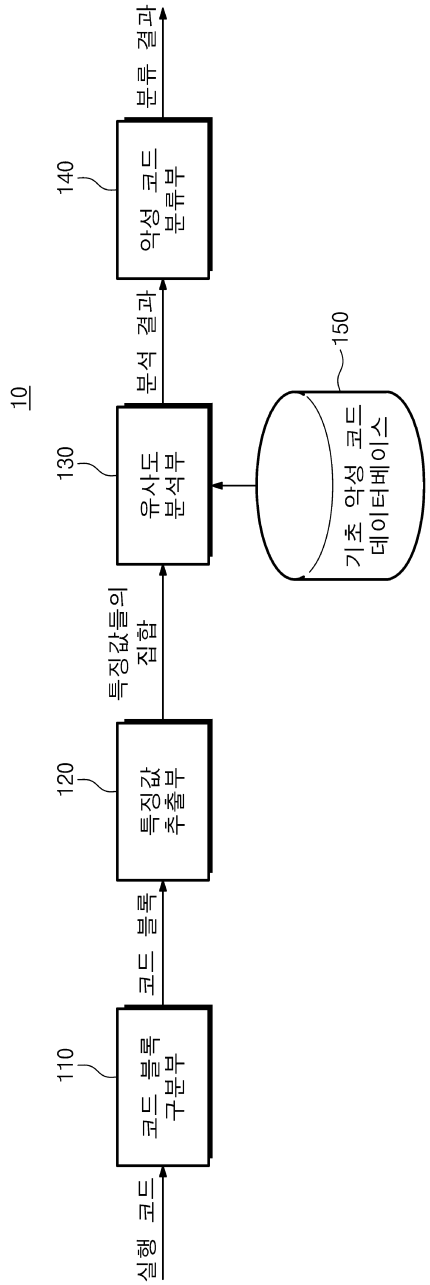
130: 유사도 분석부

140: 악성 코드 분류부

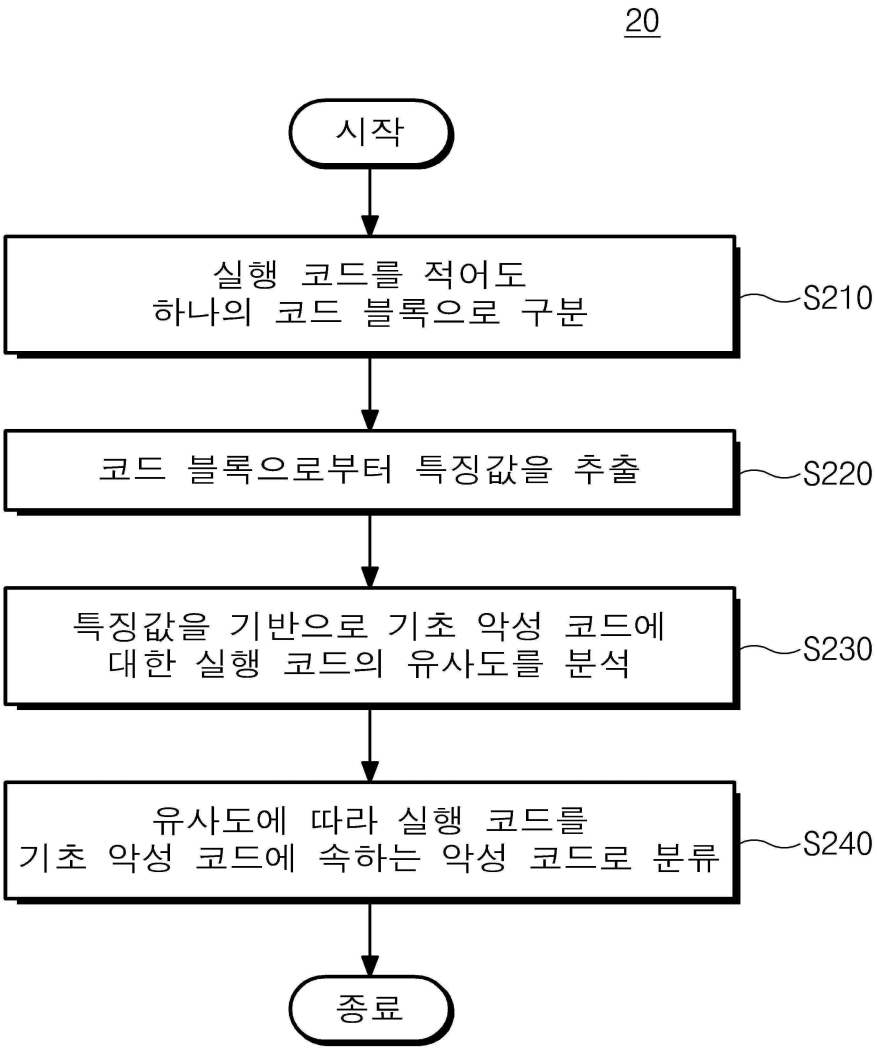
150: 데이터베이스

도면

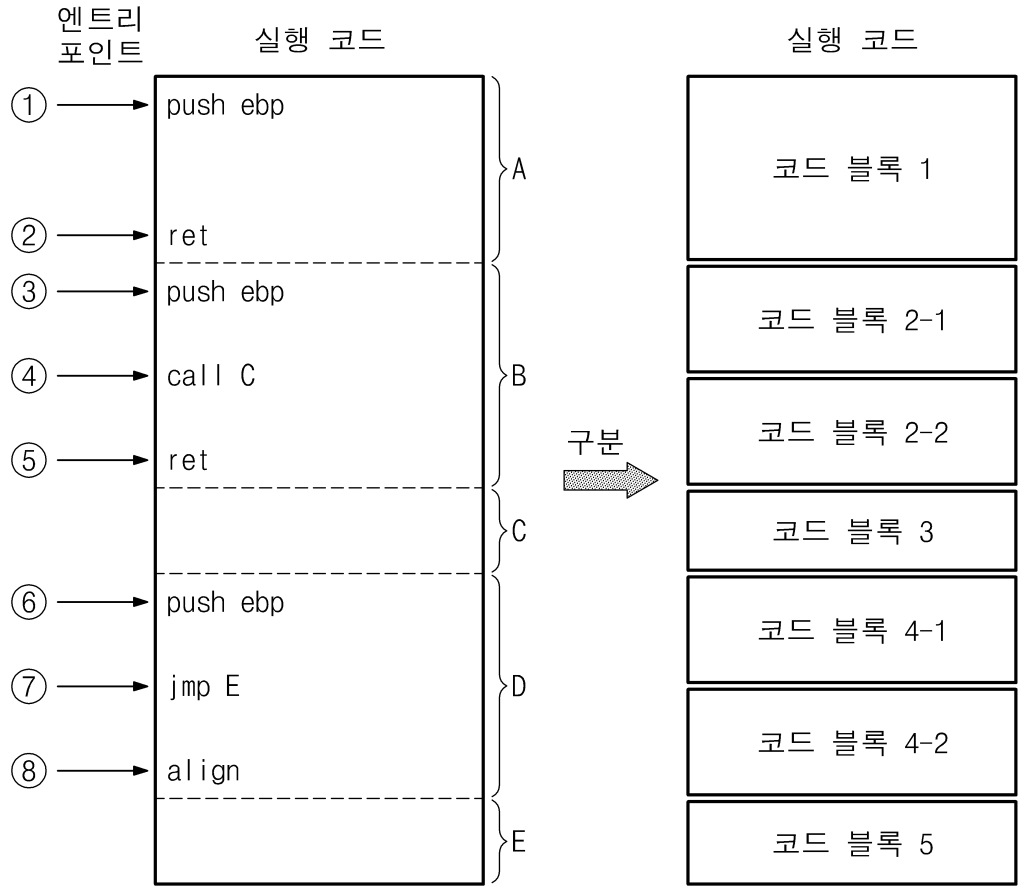
도면1


10
실행 코드
110
코드 블록 구분부
코드 블록
120
특징값 추출부
특징값들의 집합
130
유사도 분석부
분석 결과
140
악성 코드 분류부
분류 결과
150
기초 악성 코드 데이터베이스

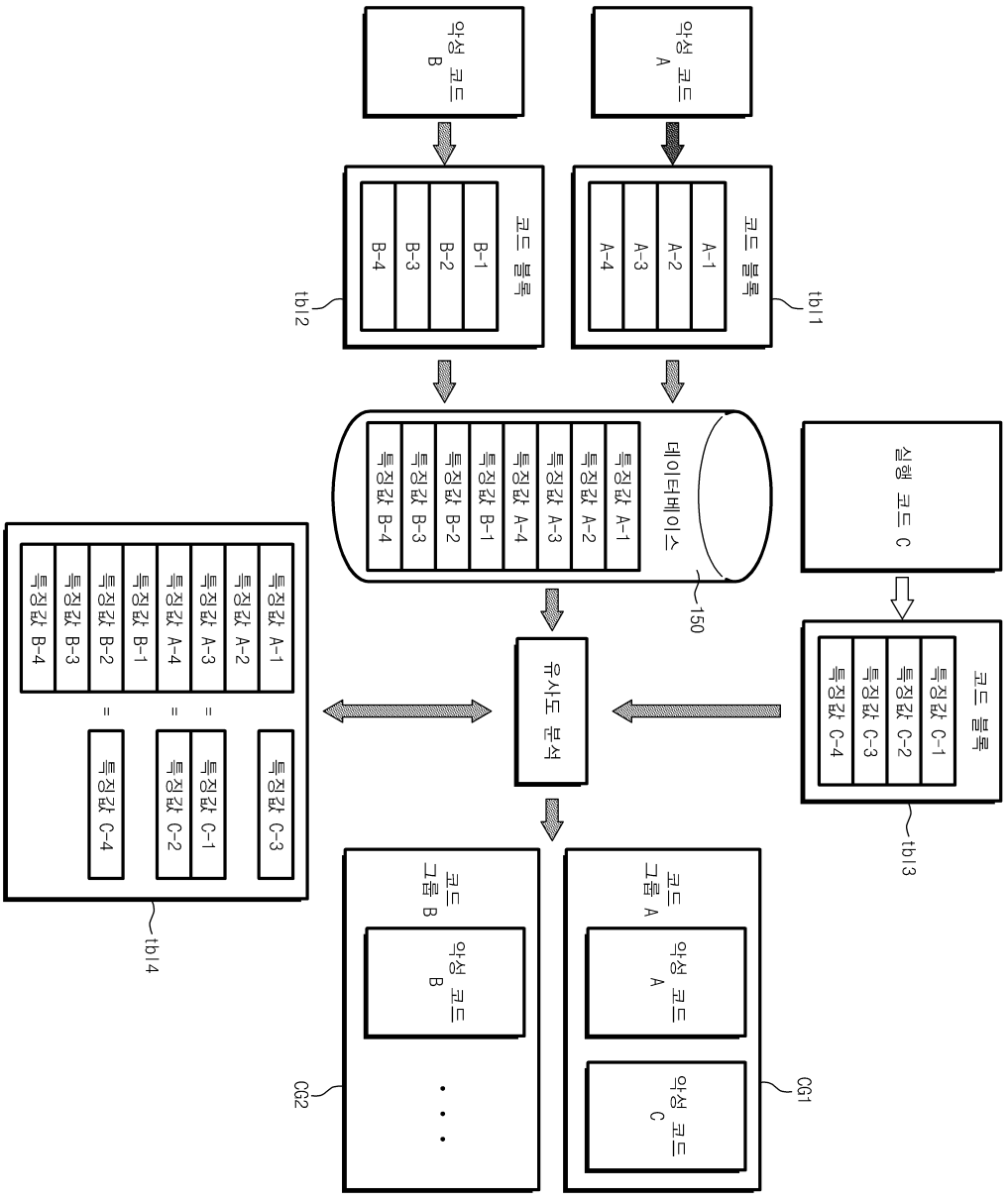
도면2


20
시작
실행 코드를 적어도
하나의 코드 블록으로 구분
S210
코드 블록으로부터 특징값을 추출
S220
특징값을 기반으로 기초 악성 코드에
대한 실행 코드의 유사도를 분석
S230
유사도에 따라 실행 코드를
기초 악성 코드에 속하는 악성 코드로 분류
S240
종료

도면3


엔트리 포인트
실행 코드
실행 코드
①
push ebp
②
ret
③
push ebp
④
call C
⑤
ret
⑥
push ebp
⑦
jmp E
⑧
align
A
B
C
D
E
구분
코드 블록 1
코드 블록 2-1
코드 블록 2-2
코드 블록 3
코드 블록 4-1
코드 블록 4-2
코드 블록 5

도면4


악성 코드 A
악성 코드 B
코드 블록
A-1
A-2
A-3
A-4
tbl1
코드 블록
B-1
B-2
B-3
B-4
tbl2
실행 코드 C
코드 블록
특징값 C-1
특징값 C-2
특징값 C-3
특징값 C-4
tbl3
데이터베이스
특징값 A-1
특징값 A-2
특징값 A-3
특징값 A-4
특징값 B-1
특징값 B-2
특징값 B-3
특징값 B-4
150
유사도 분석
특징값 A-1
특징값 A-2
특징값 A-3
특징값 A-4
특징값 B-1
특징값 B-2
특징값 B-3
특징값 B-4
=
=
=
특징값 C-3
특징값 C-1
특징값 C-2
특징값 C-4
tbl4
코드 그룹 A
악성 코드 A
악성 코드 C
CG1
코드 그룹 B
악성 코드 B
CG2