(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2021-0145935
(43) 공개일자 2021년12월03일

(51) 국제특허분류(Int. Cl.)
G06F 21/14 (2013.01) *G06F 21/12* (2013.01)
(52) CPC특허분류
G06F 21/14 (2013.01)
G06F 21/126 (2013.01)
(21) 출원번호 10-2020-0062769
(22) 출원일자 2020년05월26일
심사청구일자 없음

(71) 출원인
라인플러스 주식회사
경기도 성남시 분당구 황새울로360번길 42 ,11층(서현동,에이케이플라자분당점)
(72) 발명자
정상민
경기도 성남시 분당구 황새울로360번길 42 ,11층(서현동,에이케이플라자분당점)
최준태
경기도 성남시 분당구 황새울로360번길 42 ,11층(서현동,에이케이플라자분당점)
전상훈
경기도 성남시 분당구 황새울로360번길 42 ,11층(서현동,에이케이플라자분당점)
(74) 대리인
양성보

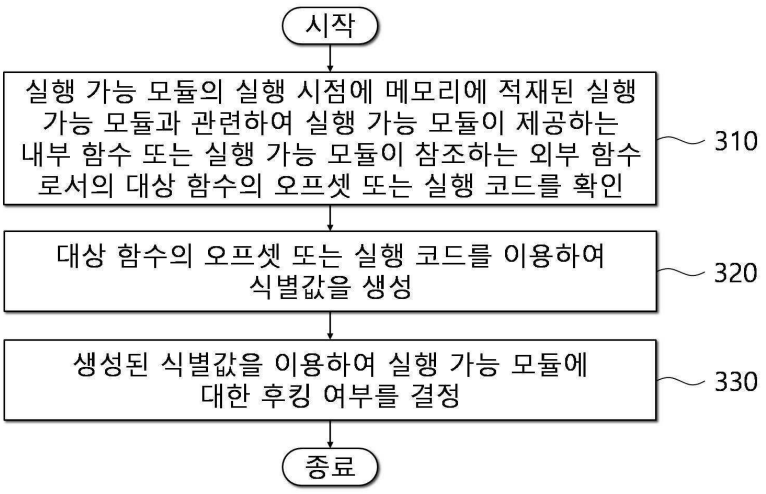
전체 청구항 수 : 총 20 항

(54) 발명의 명칭 **API 정보 군집화를 통한 후킹 판단 방법 및 그 시스템**

(57) 요 약

실행 가능한 모듈이 외부로부터 참조하거나 외부로 제공하는 함수의 위치 정보 또는 실행 코드에 대한 고유한 값을 특정하고, 이러한 고유한 값들을 비교하거나 또는 고유한 값들을 군집화하여 얻어지는 분류를 이용하여 API 후킹을 판단할 수 있는 후킹 판단 방법 및 시스템을 제공한다.

대 표 도 - 도3



명 세 서

청구범위

청구항 1

적어도 하나의 프로세서 및 메모리를 포함하는 컴퓨터 장치의 후킹 탐지 방법에 있어서,

상기 적어도 하나의 프로세서에 의해, 실행 가능 모듈의 실행 시점에 상기 메모리에 적재된 상기 실행 가능 모듈과 관련하여 상기 실행 가능 모듈이 제공하는 내부 함수 또는 상기 실행 가능 모듈이 참조하는 외부 함수로서의 대상 함수의 오프셋 또는 실행 코드를 확인하는 단계;

상기 적어도 하나의 프로세서에 의해, 상기 대상 함수의 오프셋 또는 실행 코드를 이용하여 식별값을 생성하는 단계; 및

상기 적어도 하나의 프로세서에 의해, 상기 생성된 식별값을 이용하여 상기 실행 가능 모듈에 대한 후킹 여부를 결정하는 단계

를 포함하는 후킹 탐지 방법.

청구항 2

제1항에 있어서,

상기 확인하는 단계는,

상기 실행 가능 모듈에서 참조하는 외부 함수 정보를 확인하는 단계;

상기 외부 함수 정보에서 참조하는 외부 모듈의 시작 주소를 획득하는 단계;

상기 외부 함수 정보에서 참조하는 외부 함수의 시작 주소를 획득하는 단계; 및

상기 외부 함수의 시작 주소와 상기 외부 모듈의 시작 주소의 차이를 상기 대상 함수의 오프셋으로 확인하는 단계

를 포함하는 것을 특징으로 하는 후킹 탐지 방법.

청구항 3

제2항에 있어서,

상기 확인하는 단계는,

상기 외부 함수의 시작 주소가 상기 외부 모듈의 시작 주소 및 종료 주소 범위에 포함되지 않는 경우, 상기 실행 가능 모듈 또는 상기 대상 함수에 대한 후킹이 존재하는 것으로 결정하는 단계

를 더 포함하는 것을 특징으로 하는 후킹 탐지 방법.

청구항 4

제1항에 있어서,

상기 확인하는 단계는,

상기 실행 가능 모듈에서 제공하는 내부 함수 정보를 확인하는 단계;

상기 실행 가능 모듈의 시작 주소를 확인하는 단계;

상기 내부 함수 정보에 기초하여 외부에 제공되는 내부 함수의 시작 주소를 확인하는 단계; 및

상기 내부 함수의 시작 주소와 상기 실행 가능 모듈의 시작 주소의 차이를 상기 대상 함수의 오프셋으로 확인하는 단계

를 포함하는 것을 특징으로 하는 후킹 탐지 방법.

청구항 5

제4항에 있어서,

상기 확인하는 단계는,

상기 내부 함수의 시작 주소가 상기 실행 가능 모듈의 시작 주소 및 종료 주소의 범위에 포함되지 않는 경우, 상기 실행 가능 모듈 또는 상기 대상 함수에 대한 후킹이 존재하는 것으로 결정하는 단계

를 더 포함하는 것을 특징으로 하는 후킹 탐지 방법.

청구항 6

제1항에 있어서,

상기 확인하는 단계는,

상기 실행 가능 모듈에서 참조하는 외부 함수 정보를 확인하는 단계;

상기 외부 함수 정보에서 참조하는 외부 함수의 주소를 획득하는 단계; 및

상기 외부 함수의 주소에 저장된 실행 코드를 획득하는 단계

를 포함하는 것을 특징으로 하는 후킹 탐지 방법.

청구항 7

제6항에 있어서,

상기 실행 코드를 획득하는 단계는,

상기 외부 함수의 시작 주소로부터 이동 명령어의 크기에 해당하는 주소까지의 실행 코드를 획득하거나 또는 상기 외부 함수의 시작 주소에서 종료 주소까지의 실행 코드를 획득하는 것을 특징으로 하는 후킹 탐지 방법.

청구항 8

제1항에 있어서,

상기 확인하는 단계는,

상기 실행 가능 모듈에서 제공하는 내부 함수 정보를 확인하는 단계;

상기 내부 함수 정보에 기초하여 외부에 제공되는 내부 함수의 주소를 획득하는 단계; 및

상기 내부 함수의 주소에 저장된 실행 코드를 획득하는 단계

를 포함하는 것을 특징으로 하는 후킹 탐지 방법.

청구항 9

제8항에 있어서,

상기 실행 코드를 획득하는 단계는,

상기 내부 함수의 시작 주소로부터 이동 명령어의 크기에 해당하는 주소까지의 실행 코드를 획득하거나 또는 상기 내부 함수의 시작 주소에서 종료 주소까지의 실행 코드를 획득하는 것을 특징으로 하는 후킹 탐지 방법.

청구항 10

제1항에 있어서,

상기 식별값을 생성하는 단계는,

상기 대상 함수의 오프셋 또는 실행 코드를 상기 식별값으로 생성하여 상기 대상 함수가 포함된 모듈의 식별자 및 상기 대상 함수의 식별자 중 적어도 하나와 연계하는 것을 특징으로 하는 후킹 탐지 방법.

청구항 11

제1항에 있어서,

상기 확인하는 단계는,

상기 실행 가능 모듈이 제공하는 모든 내부 함수 또는 상기 실행 가능 모듈이 참조하는 모든 외부 함수들 각각에 대해 오프셋 또는 실행 코드를 확인하고,

상기 식별값을 생성하는 단계는,

상기 모든 내부 함수 또는 상기 모든 외부 함수에 대해 확인된 오프셋들 또는 실행 코드들에 대한 하나의 해쉬 정보를 상기 식별값으로 생성하여 대응하는 모듈의 식별자와 연계하는 것

을 특징으로 하는 후킹 탐지 방법.

청구항 12

제1항에 있어서,

상기 후킹 여부를 결정하는 단계는,

상기 생성된 식별값을 이전 식별값과 비교하여 상기 실행 가능 모듈에 대한 후킹 여부를 결정하는 것을 특징으로 하는 후킹 탐지 방법.

청구항 13

제12항에 있어서,

상기 이전 식별값은, 상기 실행 가능 모듈을 포함하는 패키지 파일의 빌드 직후에 상기 실행 가능 모듈에 대해 미리 확인된 상기 대상 함수의 오프셋 또는 실행 코드, 상기 실행 가능 모듈의 이전 실행 시점에 상기 실행 가능 모듈에 대해 미리 확인된 상기 대상 함수의 오프셋 또는 실행 코드 또는 일정 주기마다 상기 실행 가능 모듈에 대해 미리 확인된 상기 대상 함수의 오프셋 또는 실행 코드를 포함하는 것을 특징으로 하는 후킹 탐지 방법.

청구항 14

제1항에 있어서,

상기 후킹 여부를 결정하는 단계는,

상기 생성된 식별값을 서버로 전송하고,

상기 서버는 클라이언트들로부터 수집된 식별값들을 연계된 모듈의 식별자 또는 대상 함수의 식별자에 따라 군집화하고, 군집된 식별값들을 비교하여 군집별로 특정 모듈 또는 특정 대상 함수의 후킹 상태를 결정하도록 구현되는 것을 특징으로 하는 후킹 탐지 방법.

청구항 15

제14항에 있어서,

상기 서버는 특정 군집에 포함된 식별값들 중 서로 동일한 식별값들을 정상적인 식별값들로 결정하고, 나머지 식별값들에 대응하는 클라이언트의 특정 모듈 또는 특정 대상 함수를 후킹 상태로 결정하도록 구현되는 것을 특징으로 하는 후킹 탐지 방법.

청구항 16

적어도 하나의 프로세서 및 메모리를 포함하는 컴퓨터 장치의 후킹 탐지 방법에 있어서,

상기 적어도 하나의 프로세서에 의해, 복수의 클라이언트들로부터 식별값들을 수신하는 단계 - 상기 식별값들 각각은 상기 복수의 클라이언트들의 메모리에 적재된 실행 가능 모듈과 관련하여 상기 실행 가능 모듈이 제공하는 내부 함수 또는 상기 실행 가능 모듈이 참조하는 외부 함수로서의 대상 함수의 오프셋 또는 실행 코드에 기반하여 생성됨 -;

상기 적어도 하나의 프로세서에 의해, 상기 수신된 식별값들을 대응하는 모듈의 식별자 또는 대응하는 대상 함수의 식별자에 따라 군집화하는 단계; 및

상기 적어도 하나의 프로세서에 의해, 임의의 군집에 대한 식별값들을 비교하여 상기 임의의 군집에 대응하는 모듈 또는 대상 함수의 후킹 상태를 결정하는 단계

를 포함하는 후킹 탐지 방법.

청구항 17

제16항에 있어서,

상기 후킹 상태를 결정하는 단계는,

상기 임의의 군집에 포함된 식별값들 중 서로 동일한 식별값들을 정상적인 식별값들로 결정하는 단계; 및

상기 임의의 군집에 포함된 식별값들 중 정상적인 식별값들로 결정된 식별값들을 제외한 나머지 식별값에 대응하는 클라이언트의 모듈 또는 대상 함수를 후킹 상태로 결정하는 단계

를 포함하는 것을 특징으로 하는 후킹 탐지 방법.

청구항 18

컴퓨터 장치와 결합되어 제1항 내지 제17항 중 어느 한 항의 방법을 컴퓨터 장치에 실행시키기 위해 컴퓨터 판독 가능한 기록매체에 저장된 컴퓨터 프로그램.

청구항 19

제1항 내지 제17항 중 어느 한 항의 방법을 컴퓨터 장치에 실행시키기 위한 컴퓨터 프로그램이 기록되어 있는 컴퓨터 판독 가능한 기록매체.

청구항 20

컴퓨터에서 판독 가능한 명령을 실행하도록 구현되는 적어도 하나의 프로세서

를 포함하고,

상기 적어도 하나의 프로세서에 의해,

복수의 클라이언트들로부터 식별값들을 수신하고 - 상기 식별값들 각각은 상기 복수의 클라이언트들의 메모리에 적재된 실행 가능 모듈과 관련하여 상기 실행 가능 모듈이 제공하는 내부 함수 또는 상기 실행 가능 모듈이 참조하는 외부 함수로서의 대상 함수의 오프셋 또는 실행 코드를 포함함 -,

상기 수신된 식별값들을 대응하는 모듈의 식별자 또는 대응하는 대상 함수의 식별자에 따라 군집화하고,

임의의 군집에 대한 식별값들을 비교하여 상기 임의의 군집에 대응하는 모듈 또는 대상 함수의 후킹 상태를 결정하는 것

을 특징으로 하는 컴퓨터 장치.

발명의 설명

기 술 분 야

[0001] 아래의 설명은 API 정보 군집화를 통한 후킹 판단 방법 및 그 시스템에 관한 것이다.

배 경 기 술

[0002] 임의의 프로그램(바이너리 혹은 소스코드)이 주어졌을 때, 해당 프로그램의 동작 방식을 분석하고 이해하는 것을 '역공학(Reverse Engineering)'이라고 부른다. 완성된 제품(바이너리)을 판매하는 기업의 입장에서는 그 제품에 해당 기업의 기술이 그대로 담겨 있기 때문에 제품의 역공학이 어렵게 되는 것을 원한다. 역공학을 어렵게 하기 위한 일례로, 프로그램의 코드에 대한 다양한 난독화 기법들이 존재한다. 일례로, 프로그램의 소스 코드 수준에서 의미 없는 로직이나 코드들을 소스 코드에 추가하거나, 소스 코드가 포함하는 문자열과 같은 데이

터들을 암호화 또는 인코딩하는 방식으로 난독화가 이루어질 수 있다. 다른 예로, 최종적으로 빌드가 완료된 실행 파일에 대하여 코드 영역을 해석하기 어렵도록 실행 파일의 형식 또는 어셈블리 코드를 수정하는 방식으로 난독화가 이루어질 수 있다. 한국등록특허 제10-1328012호는 애플리케이션 코드 난독화 장치 및 그 방법에 관한 것으로, 자바 가상 머신에서 동작하는 애플리케이션을 대상으로, 애플리케이션을 구성하는 코드 구조를 변경함으로써 난독화를 적용함을 개시하고 있다. 다시 말해 상술한 종래기술은 빌드가 완료된 애플리케이션의 어셈블리 코드를 수정하는 방식으로 난독화가 이루어지는 예를 개시하고 있다.

[0003] 한편, 이러한 역공학의 기법으로서, 어플리케이션의 동작 상태를 파악하거나 동작을 변경하기 위해 API(Application Programming Interface) 후킹이 많이 사용되고 있다. API는 사전에 약속된 형태로 인자 값을 받고 그 결과를 반환해야 한다. 이에 메모리상에서 해당 API의 호출을 가로채서 호출 여부를 파악하거나 동작 방식을 변경해 개발자가 의도하지 않은 형태로 동작하도록 어뷰징(abusing)이 가능해진다. 이러한 API 후킹을 탐지하기 위해, 종래기술에서는 API 후킹을 시도할 수 있는 디버거(Debugger) 또는 디스어셈블러(Disassembler)의 실행 여부나 설치 여부를 확인하는 방식, 리버싱(Reversing)을 수행할 수 있는 프레임워크(Framework)의 실행 여부나 설치 여부를 확인하는 방식 또는 사용중인 API 함수의 주소값을 확인하는 방식 등이 활용되었다.

[0004] 그러나, 상술한 종래기술의 경우, API 후킹 시도가 가능한 프로그램의 실행 여부나 설치 여부를 통해서 간접적으로 API 후킹 위험을 판단하는 것이기 때문에 오탐의 가능성이 있으며, 또한 시스템의 상황에 따라서 메모리에 적재되는 모듈의 위치가 가변적일 경우에는 API 함수의 주소로 후킹 여부를 탐지하기 어렵다는 문제점이 있다. 또한, 새로운 형태의 후킹 프로그램이 등장하거나 동작을 숨긴 경우, 또는 메모리에 적재된 모듈의 위치가 변경되는 경우에는 API 후킹에 대해 적시에 대응할 수 없는 문제점이 있다.

발명의 내용

해결하려는 과제

[0005] 실행 가능한 모듈이 외부로부터 참조하거나 외부로 제공하는 함수의 위치 정보 또는 실행 코드에 대한 고유한 값을 특정하고, 이러한 고유한 값들을 비교하거나 또는 고유한 값들을 군집화하여 얻어지는 분류를 이용하여 API 후킹을 판단할 수 있는 후킹 판단 방법 및 시스템을 제공한다.

과제의 해결 수단

[0006] 적어도 하나의 프로세서 및 메모리를 포함하는 컴퓨터 장치의 후킹 탐지 방법에 있어서, 상기 적어도 하나의 프로세서에 의해, 실행 가능 모듈의 실행 시점에 상기 메모리에 적재된 상기 실행 가능 모듈과 관련하여 상기 실행 가능 모듈이 제공하는 내부 함수 또는 상기 실행 가능 모듈이 참조하는 외부 함수로서의 대상 함수의 오프셋 또는 실행 코드를 확인하는 단계; 상기 적어도 하나의 프로세서에 의해, 상기 대상 함수의 오프셋 또는 실행 코드를 이용하여 식별값을 생성하는 단계; 및 상기 적어도 하나의 프로세서에 의해, 상기 생성된 식별값을 이용하여 상기 실행 가능 모듈에 대한 후킹 여부를 결정하는 단계를 포함하는 후킹 탐지 방법을 제공한다.

[0007] 일측에 따르면, 상기 확인하는 단계는, 상기 실행 가능 모듈에서 참조하는 외부 함수 정보를 확인하는 단계; 상기 외부 함수 정보에서 참조하는 외부 모듈의 시작 주소를 획득하는 단계; 상기 외부 함수 정보에서 참조하는 외부 함수의 시작 주소를 획득하는 단계; 및 상기 외부 함수의 시작 주소와 상기 외부 모듈의 시작 주소의 차이를 상기 대상 함수의 오프셋으로 확인하는 단계를 포함하는 것을 특징으로 할 수 있다.

[0008] 다른 측면에 따르면, 상기 확인하는 단계는, 상기 외부 함수의 시작 주소가 상기 외부 모듈의 시작 주소 및 종료 주소 범위에 포함되지 않는 경우, 상기 실행 가능 모듈 또는 상기 대상 함수에 대한 후킹이 존재하는 것으로 결정하는 단계를 더 포함하는 것을 특징으로 할 수 있다.

[0009] 또 다른 측면에 따르면, 상기 확인하는 단계는, 상기 실행 가능 모듈에서 제공하는 내부 함수 정보를 확인하는 단계; 상기 실행 가능 모듈의 시작 주소를 확인하는 단계; 상기 내부 함수 정보에 기초하여 외부에 제공되는 내부 함수의 시작 주소를 확인하는 단계; 및 상기 내부 함수의 시작 주소와 상기 실행 가능 모듈의 시작 주소의 차이를 상기 대상 함수의 오프셋으로 확인하는 단계를 포함하는 것을 특징으로 할 수 있다.

[0010] 또 다른 측면에 따르면, 상기 확인하는 단계는, 상기 내부 함수의 시작 주소가 상기 실행 가능 모듈의 시작 주소 및 종료 주소의 범위에 포함되지 않는 경우, 상기 실행 가능 모듈 또는 상기 대상 함수에 대한 후킹이 존재하는 것으로 결정하는 단계를 더 포함하는 것을 특징으로 할 수 있다.

[0011] 또 다른 측면에 따르면, 상기 확인하는 단계는, 상기 실행 가능 모듈에서 참조하는 외부 함수 정보를 확인하는

단계; 상기 외부 함수 정보에서 참조하는 외부 함수의 주소를 획득하는 단계; 및 상기 외부 함수의 주소에 저장된 실행 코드를 획득하는 단계를 포함하는 것을 특징으로 할 수 있다.

[0012] 또 다른 측면에 따르면, 상기 실행 코드를 획득하는 단계는, 상기 외부 함수의 시작 주소로부터 이동 명령어의 크기에 해당하는 주소까지의 실행 코드를 획득하거나 또는 상기 외부 함수의 시작 주소에서 종료 주소까지의 실행 코드를 획득하는 것을 특징으로 할 수 있다.

[0013] 또 다른 측면에 따르면, 상기 확인하는 단계는, 상기 실행 가능 모듈에서 제공하는 내부 함수 정보를 확인하는 단계; 상기 내부 함수 정보에 기초하여 외부에 제공되는 내부 함수의 주소를 획득하는 단계; 및 상기 내부 함수의 주소에 저장된 실행 코드를 획득하는 단계를 포함하는 것을 특징으로 할 수 있다.

[0014] 또 다른 측면에 따르면, 상기 실행 코드를 획득하는 단계는, 상기 내부 함수의 시작 주소로부터 이동 명령어의 크기에 해당하는 주소까지의 실행 코드를 획득하거나 또는 상기 내부 함수의 시작 주소에서 종료 주소까지의 실행 코드를 획득하는 것을 특징으로 할 수 있다.

[0015] 또 다른 측면에 따르면, 상기 식별값을 생성하는 단계는, 상기 대상 함수의 오프셋 또는 실행 코드를 상기 대상 함수가 포함된 모듈의 식별자 및 상기 대상 함수의 식별자 중 적어도 하나와 연계하여 상기 식별값을 생성하는 것을 특징으로 할 수 있다.

[0016] 또 다른 측면에 따르면, 상기 확인하는 단계는, 상기 실행 가능 모듈이 제공하는 모든 내부 함수 또는 상기 실행 가능 모듈이 참조하는 모든 외부 함수들 각각에 대해 오프셋 또는 실행 코드를 확인하고, 상기 식별값을 생성하는 단계는, 상기 모든 내부 함수 또는 상기 모든 외부 함수에 대해 확인된 오프셋들 또는 실행 코드들에 대한 하나의 해쉬 정보를 대응하는 모듈의 식별자와 연계하여 상기 식별값을 생성하는 것을 특징으로 할 수 있다.

[0017] 또 다른 측면에 따르면, 상기 후킹 여부를 결정하는 단계는, 상기 생성된 식별값을 이전 식별값과 비교하여 상기 실행 가능 모듈에 대한 후킹 여부를 결정하는 것을 특징으로 할 수 있다.

[0018] 또 다른 측면에 따르면, 상기 이전 식별값은, 상기 실행 가능 모듈을 포함하는 패키지 파일의 빌드 직후에 상기 실행 가능 모듈에 대해 미리 확인된 상기 대상 함수의 오프셋 또는 실행 코드, 상기 실행 가능 모듈의 이전 실행 시점에 상기 실행 가능 모듈에 대해 미리 확인된 상기 대상 함수의 오프셋 또는 실행 코드 또는 일정 주기마다 상기 실행 가능 모듈에 대해 미리 확인된 상기 대상 함수의 오프셋 또는 실행 코드를 포함하는 것을 특징으로 할 수 있다.

[0019] 또 다른 측면에 따르면, 상기 후킹 여부를 결정하는 단계는, 상기 생성된 식별값을 서버로 전송하고, 상기 서버는 클라이언트들로부터 수집된 식별값들을 연계된 모듈의 식별자 또는 대상 함수의 식별자에 따라 군집화하고, 군집된 식별값들을 비교하여 군집별로 특정 모듈 또는 특정 대상 함수의 후킹 상태를 결정하도록 구현되는 것을 특징으로 할 수 있다.

[0020] 또 다른 측면에 따르면, 상기 서버는 특정 군집에 포함된 식별값들 중 서로 동일한 식별값들을 정상적인 식별값들로 결정하고, 나머지 식별값들에 대응하는 클라이언트의 특정 모듈 또는 특정 대상 함수를 후킹 상태로 결정하도록 구현되는 것을 특징으로 할 수 있다.

[0021] 적어도 하나의 프로세서 및 메모리를 포함하는 컴퓨터 장치의 후킹 탐지 방법에 있어서, 상기 적어도 하나의 프로세서에 의해, 복수의 클라이언트들로부터 식별값들을 수신하는 단계 - 상기 식별값들 각각은 상기 복수의 클라이언트들의 메모리에 적재된 실행 가능 모듈과 관련하여 상기 실행 가능 모듈이 제공하는 내부 함수 또는 상기 실행 가능 모듈이 참조하는 외부 함수로서의 대상 함수의 오프셋 또는 실행 코드에 기반하여 생성됨 -; 상기 적어도 하나의 프로세서에 의해, 상기 수신된 식별값들을 대응하는 모듈의 식별자 또는 대응하는 대상 함수의 식별자에 따라 군집화하는 단계; 및 상기 적어도 하나의 프로세서에 의해, 임의의 군집에 대한 식별값들을 비교하여 상기 임의의 군집에 대응하는 모듈 또는 대상 함수의 후킹 상태를 결정하는 단계를 포함하는 후킹 탐지 방법을 제공한다.

[0022] 컴퓨터 장치와 결합되어 상기 방법을 컴퓨터 장치에 실행시키기 위해 컴퓨터 판독 가능한 기록매체에 저장된 컴퓨터 프로그램을 제공한다.

[0023] 상기 방법을 컴퓨터 장치에 실행시키기 위한 프로그램이 기록되어 있는 컴퓨터 판독 가능한 기록매체를 제공한다.

[0024] 컴퓨터에서 판독 가능한 명령을 실행하도록 구현되는 적어도 하나의 프로세서를 포함하고, 상기 적어도 하나의

프로세서에 의해, 복수의 클라이언트들로부터 식별값들을 수신하고 - 상기 식별값들 각각은 상기 복수의 클라이언트들의 메모리에 적재된 실행 가능 모듈과 관련하여 상기 실행 가능 모듈이 제공하는 내부 함수 또는 상기 실행 가능 모듈이 참조하는 외부 함수로서의 대상 함수의 오프셋 또는 실행 코드에 기반하여 생성됨 -, 상기 수신된 식별값들을 대응하는 모듈의 식별자 또는 대응하는 대상 함수의 식별자에 따라 군집화하고, 임의의 군집에 대한 식별값들을 비교하여 상기 임의의 군집에 대응하는 모듈 또는 대상 함수의 후킹 상태를 결정하는 것을 특징으로 하는 컴퓨터 장치를 제공한다.

발명의 효과

[0025] 실행 가능한 모듈이 외부로부터 참조하거나 외부로 제공하는 함수의 위치 정보 또는 실행 코드에 대한 고유한 값을 특정하고, 이러한 고유한 값들을 비교하거나 또는 고유한 값들을 군집화하여 얻어지는 분류를 이용하여 API 후킹을 판단할 수 있다.

도면의 간단한 설명

[0026] 도 1은 본 발명의 일실시예에 따른 네트워크 환경의 예를 도시한 도면이다.

도 2는 본 발명의 일실시예에 따른 컴퓨터 장치의 예를 도시한 블록도이다.

도 3은 본 발명의 일실시예에 따른 후킹 탐지 방법의 예를 도시한 흐름도이다.

도 4는 본 발명의 일실시예에 있어서, 외부 함수의 오프셋을 확인하는 과정의 예를 도시한 흐름도이다.

도 5는 본 발명의 일실시예에 있어서, 내부 함수의 오프셋을 확인하는 과정의 예를 도시한 흐름도이다.

도 6은 본 발명의 일실시예에 있어서, 외부 함수의 실행 코드를 확인하는 과정의 예를 도시한 흐름도이다.

도 7은 본 발명의 일실시예에 있어서, 내부 함수의 실행코드를 확인하는 과정의 예를 도시한 흐름도이다.

도 8 내지 도 9는 본 발명의 실시예들에 따라 오프셋 또는 실행 코드를 획득하는 예들을 도시한 도면들이다.

도 10은 본 발명의 일실시예에 있어서, 서버의 후킹 탐지 방법의 예를 도시한 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0027] 이하, 실시예를 첨부한 도면을 참조하여 상세히 설명한다.

[0028] 본 발명의 실시예들에 따른 후킹 판단 시스템은 적어도 하나의 컴퓨터 장치에 의해 구현될 수 있으며, 본 발명의 실시예들에 따른 후킹 판단 방법은 후킹 판단 시스템에 포함되는 적어도 하나의 컴퓨터 장치를 통해 수행될 수 있다. 컴퓨터 장치에는 본 발명의 일실시예에 따른 컴퓨터 프로그램이 설치 및 구동될 수 있고, 컴퓨터 장치는 구동된 컴퓨터 프로그램의 제어에 따라 본 발명의 실시예들에 따른 후킹 판단 방법을 수행할 수 있다. 상술한 컴퓨터 프로그램은 컴퓨터 장치와 결합되어 후킹 판단 방법을 컴퓨터 장치에 실행시키기 위해 컴퓨터 판독 가능한 기록매체에 저장될 수 있다.

[0029] 도 1은 본 발명의 일실시예에 따른 네트워크 환경의 예를 도시한 도면이다. 도 1의 네트워크 환경은 복수의 전자 기기들(110, 120, 130, 140), 복수의 서버들(150, 160) 및 네트워크(170)를 포함하는 예를 나타내고 있다. 이러한 도 1은 발명의 설명을 위한 일례로 전자 기기의 수나 서버의 수가 도 1과 같이 한정되는 것은 아니다. 또한, 도 1의 네트워크 환경은 본 실시예들에 적용 가능한 환경들 중 하나의 예를 설명하는 것일 뿐, 본 실시예들에 적용 가능한 환경이 도 1의 네트워크 환경으로 한정되는 것은 아니다.

[0030] 복수의 전자 기기들(110, 120, 130, 140)은 컴퓨터 장치로 구현되는 고정형 단말이거나 이동형 단말일 수 있다. 복수의 전자 기기들(110, 120, 130, 140)의 예를 들면, 스마트폰(smart phone), 휴대폰, 네비게이션, 컴퓨터, 노트북, 디지털방송용 단말, PDA(Personal Digital Assistants), PMP(Portable Multimedia Player), 태블릿 PC 등이 있다. 일례로 도 1에서는 전자 기기(110)의 예로 스마트폰의 형상을 나타내고 있으나, 본 발명의 실시예들에서 전자 기기(110)는 실질적으로 무선 또는 유선 통신 방식을 이용하여 네트워크(170)를 통해 다른 전자 기기들(120, 130, 140) 및/또는 서버(150, 160)와 통신할 수 있는 다양한 물리적인 컴퓨터 장치들 중 하나를 의미할 수 있다.

[0031] 통신 방식은 제한되지 않으며, 네트워크(170)가 포함할 수 있는 통신망(일례로, 이동통신망, 유선 인터넷, 무선 인터넷, 방송망)을 활용하는 통신 방식뿐만 아니라 기기들간의 근거리 무선 통신 역시 포함될 수 있다. 예를

들어, 네트워크(170)는, PAN(personal area network), LAN(local area network), CAN(campus area network), MAN(metropolitan area network), WAN(wide area network), BBN(broadband network), 인터넷 등의 네트워크 중 하나 이상의 임의의 네트워크를 포함할 수 있다. 또한, 네트워크(170)는 버스 네트워크, 스타 네트워크, 링 네트워크, 메쉬 네트워크, 스타-버스 네트워크, 트리 또는 계층적(hierarchical) 네트워크 등을 포함하는 네트워크 토폴로지 중 임의의 하나 이상을 포함할 수 있으나, 이에 제한되지 않는다.

[0032] 서버(150, 160) 각각은 복수의 전자 기기들(110, 120, 130, 140)과 네트워크(170)를 통해 통신하여 명령, 코드, 파일, 컨텐츠, 서비스 등을 제공하는 컴퓨터 장치 또는 복수의 컴퓨터 장치들로 구현될 수 있다. 예를 들어, 서버(150)는 네트워크(170)를 통해 접속한 복수의 전자 기기들(110, 120, 130, 140)로 서비스(일례로, 파일 배포 서비스, 지도 서비스, 컨텐츠 제공 서비스, 그룹 통화 서비스(또는 음성 컨퍼런스 서비스), 메시징 서비스, 메일 서비스, 소셜 네트워크 서비스, 지도 서비스, 번역 서비스, 금융 서비스, 결제 서비스, 검색 서비스 등)를 제공하는 시스템일 수 있다.

[0033] 도 2는 본 발명의 일실시예에 따른 컴퓨터 장치의 예를 도시한 블록도이다. 앞서 설명한 복수의 전자 기기들(110, 120, 130, 140) 각각이나 서버들(150, 160) 각각은 도 2를 통해 도시된 컴퓨터 장치(200)에 의해 구현될 수 있다.

[0034] 이러한 컴퓨터 장치(200)는 도 2에 도시된 바와 같이, 메모리(210), 프로세서(220), 통신 인터페이스(230) 그리고 입출력 인터페이스(240)를 포함할 수 있다. 메모리(210)는 컴퓨터에서 판독 가능한 기록매체로서, RAM(random access memory), ROM(read only memory) 및 디스크 드라이브와 같은 비소멸성 대용량 기록장치(permanent mass storage device)를 포함할 수 있다. 여기서 ROM과 디스크 드라이브와 같은 비소멸성 대용량 기록장치는 메모리(210)와는 구분되는 별도의 영구 저장 장치로서 컴퓨터 장치(200)에 포함될 수도 있다. 또한, 메모리(210)에는 운영체제와 적어도 하나의 프로그램 코드가 저장될 수 있다. 이러한 소프트웨어 구성요소들은 메모리(210)와는 별도의 컴퓨터에서 판독 가능한 기록매체로부터 메모리(210)로 로딩될 수 있다. 이러한 별도의 컴퓨터에서 판독 가능한 기록매체는 플로피 드라이브, 디스크, 테이프, DVD/CD-ROM 드라이브, 메모리 카드 등의 컴퓨터에서 판독 가능한 기록매체를 포함할 수 있다. 다른 실시예에서 소프트웨어 구성요소들은 컴퓨터에서 판독 가능한 기록매체가 아닌 통신 인터페이스(230)를 통해 메모리(210)에 로딩될 수도 있다. 예를 들어, 소프트웨어 구성요소들은 네트워크(170)를 통해 수신되는 파일들에 의해 설치되는 컴퓨터 프로그램에 기반하여 컴퓨터 장치(200)의 메모리(210)에 로딩될 수 있다.

[0035] 프로세서(220)는 기본적인 산술, 로직 및 입출력 연산을 수행함으로써, 컴퓨터 프로그램의 명령을 처리하도록 구성될 수 있다. 명령은 메모리(210) 또는 통신 인터페이스(230)에 의해 프로세서(220)로 제공될 수 있다. 예를 들어 프로세서(220)는 메모리(210)와 같은 기록 장치에 저장된 프로그램 코드에 따라 수신되는 명령을 실행하도록 구성될 수 있다.

[0036] 통신 인터페이스(230)는 네트워크(170)를 통해 컴퓨터 장치(200)가 다른 장치(일례로, 앞서 설명한 저장 장치들)와 서로 통신하기 위한 기능을 제공할 수 있다. 일례로, 컴퓨터 장치(200)의 프로세서(220)가 메모리(210)와 같은 기록 장치에 저장된 프로그램 코드에 따라 생성한 요청이나 명령, 데이터, 파일 등이 통신 인터페이스(230)의 제어에 따라 네트워크(170)를 통해 다른 장치들로 전달될 수 있다. 역으로, 다른 장치로부터의 신호나 명령, 데이터, 파일 등이 네트워크(170)를 거쳐 컴퓨터 장치(200)의 통신 인터페이스(230)를 통해 컴퓨터 장치(200)로 수신될 수 있다. 통신 인터페이스(230)를 통해 수신된 신호나 명령, 데이터 등은 프로세서(220)나 메모리(210)로 전달될 수 있고, 파일 등은 컴퓨터 장치(200)가 더 포함할 수 있는 저장 매체(상술한 영구 저장 장치)로 저장될 수 있다.

[0037] 입출력 인터페이스(240)는 입출력 장치(250)와의 인터페이스를 위한 수단일 수 있다. 예를 들어, 입력 장치는 마이크, 키보드 또는 마우스 등의 장치를, 그리고 출력 장치는 디스플레이, 스피커와 같은 장치를 포함할 수 있다. 다른 예로 입출력 인터페이스(240)는 터치스크린과 같이 입력과 출력을 위한 기능이 하나로 통합된 장치와의 인터페이스를 위한 수단일 수도 있다. 입출력 장치(250)는 컴퓨터 장치(200)와 하나의 장치로 구성될 수도 있다.

[0038] 또한, 다른 실시예들에서 컴퓨터 장치(200)는 도 2의 구성요소들보다 더 적은 혹은 더 많은 구성요소들을 포함할 수도 있다. 그러나, 대부분의 종래기술적 구성요소들을 명확하게 도시할 필요성은 없다. 예를 들어, 컴퓨터 장치(200)는 상술한 입출력 장치(250) 중 적어도 일부를 포함하도록 구현되거나 또는 트랜시버(transceiver), 데이터베이스 등과 같은 다른 구성요소들을 더 포함할 수도 있다.

[0039] 도 3은 본 발명의 일실시예에 따른 클라이언트의 후킹 탐지 방법의 예를 도시한 흐름도이다. 본 실시예에 따른 후킹 탐지 방법은 후킹 탐지의 대상이 되는 실행 가능 모듈을 적어도 활용하여 컴퓨터 프로그램을 실행하는 클라이언트의 컴퓨터 장치(200)에 의해 수행될 수 있다. 이때, 컴퓨터 장치(200)의 프로세서(220)는 메모리(210)가 포함하는 운영체제의 코드나 적어도 하나의 컴퓨터 프로그램의 코드에 따른 제어 명령(instruction)을 실행하도록 구현될 수 있다. 여기서, 프로세서(220)는 컴퓨터 장치(200)에 저장된 코드가 제공하는 제어 명령에 따라 컴퓨터 장치(200)가 도 3의 방법이 포함하는 단계들(310 내지 330)을 수행하도록 컴퓨터 장치(200)를 제어할 수 있다.

[0040] 단계(310)에서 컴퓨터 장치(200)는 실행 가능 모듈의 실행 시점에 메모리(210)에 적재된 실행 가능 모듈과 관련하여 실행 가능 모듈이 제공하는 내부 함수 또는 실행 가능 모듈이 참조하는 외부 함수로서의 대상 함수의 오프셋 또는 실행 코드를 확인할 수 있다.

[0041] 하나의 실행 가능 모듈은 API 호출에 기반하여 외부의 다른 모듈(해당 실행 가능 모듈의 관점에서의 외부 모듈, 이하 '외부 모듈')이 포함하는 함수(해당 실행 가능 모듈의 관점에서의 외부 함수, 이하 '외부 함수')를 참조할 수도 있고, 외부 모듈로부터의 API 호출을 통해 자신의 함수(해당 실행 가능 모듈의 관점에서의 내부 함수, 이하 '내부 함수')를 제공할 수도 있다. 이때, 컴퓨터 장치(200)는 외부 함수 또는 내부 함수로서의 대상 함수의 오프셋 또는 실행 코드를 확인할 수 있다. 오프셋은 대상 함수의 위치에 대한 고유한 값으로서 활용될 수 있으며, 실행 코드는 대상 함수의 내용에 대한 고유한 값으로서 활용될 수 있다. 이러한 고유한 값들은 이후 단계(320)에서 식별값을 생성하는데 활용될 수 있다. 이때, 외부 함수의 오프셋 및 외부 함수의 실행 코드 중 적어도 하나가 활용되거나 및/또는 내부 함수의 오프셋 및 내부 함수의 실행 코드 중 적어도 하나가 식별값을 생성하는데 활용될 수 있으며, 함수의 오프셋이나 실행 코드를 획득하기 위한 방법에 대해서는 이후 도 4 내지 도 7을 통해 더욱 자세히 설명한다.

[0042] 단계(320)에서 컴퓨터 장치(200)는 대상 함수의 오프셋 또는 실행 코드를 이용하여 식별값을 생성할 수 있다. 일실시예에서, 컴퓨터 장치(200)는 대상 함수의 오프셋 또는 실행 코드를 대상 함수가 포함된 모듈의 식별자 및 대상 함수의 식별자 중 적어도 하나와 연계하여 식별값을 생성할 수 있다. 여기서, 대상 함수가 포함된 모듈은 외부 함수에 대해서는 외부 모듈일 수 있으며, 내부 함수에 대해서는 실행 가능 모듈일 수 있다. 또한, 모듈의 식별자는 일례로 모듈명일 수 있으며 함수의 식별자는 함수명일 수 있으나, 특정 모듈이나 함수를 식별할 수 있는 값이라면 한정되지 않고 활용될 수 있다. 다른 실시예에서, 컴퓨터 장치(200)는 단계(310)에서 실행 가능 모듈이 제공하는 모든 내부 함수 또는 실행 가능 모듈이 참조하는 모든 외부 함수들 각각에 대해 오프셋 또는 실행 코드를 확인할 수 있다. 이 경우, 컴퓨터 장치(200)는 단계(320)에서 모든 내부 함수 또는 모든 외부 함수에 대해 확인된 오프셋들 또는 실행 코드들에 대한 하나의 해쉬 정보를 대응하는 모듈의 식별자와 연계하여 식별값을 생성할 수 있다.

[0043] 단계(330)에서 컴퓨터 장치(200)는 생성된 식별값을 이용하여 실행 가능 모듈에 대한 후킹 여부를 결정할 수 있다. 만약, API 후킹이 발생하여 대상 함수가 후킹됨에 따라 대상 함수가 아닌 다른 함수가 실행되는 경우, 다른 함수의 오프셋 및/또는 실행 코드는 대상 함수의 오프셋 및/또는 실행 코드와 달라지게 된다. 이는 동일한 모듈 및/또는 함수에 대해 얻어지는 식별값이 API 후킹의 발생 여부에 따라 서로 달라지게 됨을 의미할 수 있다. 따라서, 컴퓨터 장치(200)는 생성된 식별값을 이용하여 실행 가능 모듈에 대한 후킹 여부를 결정할 수 있게 된다. 보다 구체적인 예로, 컴퓨터 장치(200)는 생성된 식별값을 이전 식별값과 비교하여 실행 가능 모듈에 대한 후킹 여부를 결정할 수 있다. 이때, 컴퓨터 장치(200)는 생성된 식별값과 이전 식별값이 서로 동일하지 않으면, 실행 중인 실행 가능 모듈 및/또는 대상 함수에 대해 후킹이 발생하였음을 확인할 수 있게 된다. 여기서, 이전 식별값은 (1) 실행 가능 모듈을 포함하는 패키지 파일의 빌드 직후에 실행 가능 모듈에 대해 미리 확인된 대상 함수의 오프셋 또는 실행 코드, (2) 실행 가능 모듈의 이전 실행 시점에 실행 가능 모듈에 대해 미리 확인된 대상 함수의 오프셋 또는 실행 코드 및/또는 (3) 일정 주기마다 상기 실행 가능 모듈에 대해 미리 확인된 상기 대상 함수의 오프셋 또는 실행 코드를 포함할 수 있다.

[0044] 예를 들어, 식별값이 대상 함수의 오프셋을 포함하는 경우, 대상 함수에 대해 생성된 식별값이 해당 대상 함수에 대해 이전에 생성된 이전 식별값과 동일하지 않다는 것은 대상 함수에 대해 API 후킹이 발생하였음을 의미할 수 있다. 다른 예로, 식별값이 실행 가능 모듈이 제공하는 모든 내부 함수의 오프셋들 또는 실행 코드들에 대한 해쉬 정보를 포함하는 경우, 해당 식별값이 이전 식별값과 동일하지 않다는 것은 해당 실행 가능 모듈의 내부 함수들 중 적어도 하나에 대해 API 후킹이 발생하였음을 의미할 수 있다. 또 다른 예로, 식별값이 실행 가능 모듈이 제공하는 모든 외부 함수의 오프셋들 또는 실행 코드들에 대한 해쉬 정보를 포함하는 경우, 해당 식별값이 이전 식별값과 동일하지 않다는 것은 특정 외부 모듈(특히, 해당 외부 모듈이 참조하는 외부 함수들 중

적어도 하나)에 대해 API 후킹이 발생하였음을 의미할 수 있다.

[0045] 이처럼 본 실시예에 따른 후킹 탐지 방법에서는 실행 가능 함수와 관련된 API 함수에 대한 위치 정보나 내용에 대한 고유한 값(이전 식별값)을 미리 특정해놓고, 추후 실행 가능 함수의 실행 시점에 생성되는 식별값을 이전 식별값과 비교하는 방식으로 후킹의 발생 여부를 탐지할 수 있다.

[0046] 도 4는 본 발명의 일실시예에 있어서, 외부 함수의 오프셋을 확인하는 과정의 예를 도시한 흐름도이다. 도 4에 나타난 단계들(410 내지 450)은 도 3의 단계(310)에 포함되어 컴퓨터 장치(200)에 의해 수행될 수 있다.

[0047] 단계(410)에서 컴퓨터 장치(200)는 실행 가능 모듈에서 참조하는 외부 함수 정보를 확인할 수 있다. 일례로, 외부 함수 정보는 실행 가능 모듈에 대해 정의되는 IAT(Import Address Table)에 대응할 수 있다.

[0048] 단계(420)에서 컴퓨터 장치(200)는 외부 함수 정보에서 참조하는 외부 모듈의 시작 주소를 획득할 수 있다.

[0049] 단계(430)에서 컴퓨터 장치(200)는 외부 함수 정보에서 참조하는 외부 함수의 시작 주소를 획득할 수 있다.

[0050] 단계(440)에서 컴퓨터 장치(200)는 외부 함수의 시작 주소가 외부 모듈의 시작 주소 및 종료 주소 범위에 포함되지 않는 경우, 실행 가능 모듈 또는 대상 함수에 대한 후킹이 존재하는 것으로 결정할 수 있다. 외부 모듈이 포함하는 외부 함수의 시작 주소는 적어도 외부 모듈의 시작 주소 및 종료 주소의 범위에 포함되어야 한다. 만약, 외부 함수가 후킹되어 다른 함수가 실행된다면, 외부 함수의 시작 주소로서 다른 함수의 시작 주소가 외부 함수 정보에 기록되며, 다른 함수의 시작 주소는 외부 모듈의 시작 주소 및 종료 주소의 범위를 벗어나게 된다. 따라서, 이 경우에는 오프셋을 식별값으로 활용하지 않고도 바로 후킹 여부를 판단할 수 있다.

[0051] 단계(450)에서 컴퓨터 장치(200)는 외부 함수의 시작 주소와 외부 모듈의 시작 주소의 차이를 대상 함수의 오프셋으로 확인할 수 있다. 이러한 오프셋은 외부 함수의 외부 모듈의 시작 주소에 대한 상대 주소를 의미할 수 있다. 외부 모듈의 시작 주소가 실행 시점마다 변경된다 하더라도 외부 함수의 오프셋은 변경되지 않기 때문에, 이러한 오프셋을 포함하는 식별값이 이전 식별값과 다르다면, 해당 외부 함수에 대한 후킹이 발생하여 다른 함수가 수행되고 있음을 의미할 수 있다.

[0052] 도 5는 본 발명의 일실시예에 있어서, 내부 함수의 오프셋을 확인하는 과정의 예를 도시한 흐름도이다. 도 5에 나타난 단계들(510 내지 550)은 도 3의 단계(310)에 포함되어 컴퓨터 장치(200)에 의해 수행될 수 있다.

[0053] 단계(510)에서 컴퓨터 장치(200)는 실행 가능 모듈에서 제공하는 내부 함수 정보를 확인할 수 있다. 일례로, 내부 함수 정보는 실행 가능 모듈에 대해 정의되는 EAT(Export Address Table)에 대응할 수 있다.

[0054] 단계(520)에서 컴퓨터 장치(200)는 실행 가능 모듈의 시작 주소를 확인할 수 있다.

[0055] 단계(530)에서 컴퓨터 장치(200)는 내부 함수 정보에 기초하여 외부에 제공되는 내부 함수의 시작 주소를 확인할 수 있다.

[0056] 단계(540)에서 컴퓨터 장치(200)는 내부 함수의 시작 주소가 실행 가능 모듈의 시작 주소 및 종료 주소의 범위에 포함되지 않는 경우, 실행 가능 모듈 또는 대상 함수에 대한 후킹이 존재하는 것으로 결정할 수 있다. 실행 가능 모듈이 포함하는 내부 함수의 시작 주소는 실행 가능 모듈의 시작 주소 및 종료 주소의 범위에 포함되어야 한다. 만약, 내부 함수가 후킹되어 다른 함수가 실행된다면, 내부 함수의 시작 주소로서 다른 함수의 시작 주소가 내부 함수 정보에 기록되며, 다른 함수의 시작 주소는 실행 가능 모듈의 시작 주소 및 종료 주소의 범위를 벗어나게 된다. 따라서, 이 경우에는 오프셋을 식별값으로 활용하지 않고도 바로 후킹 여부를 판단할 수 있다.

[0057] 단계(550)에서 컴퓨터 장치(200)는 내부 함수의 시작 주소와 실행 가능 모듈의 시작 주소의 차이를 대상 함수의 오프셋으로 확인할 수 있다. 이러한 오프셋은 내부 함수의 실행 가능 모듈의 시작 주소에 대한 상대 주소를 의미할 수 있다. 실행 가능 모듈의 시작 주소가 실행 시점마다 변경된다 하더라도 내부 함수의 오프셋은 변경되지 않기 때문에, 이러한 오프셋을 포함하는 식별값이 이전 식별값과 다르다면, 해당 내부 함수에 대한 후킹이 발생하여 다른 함수가 수행되고 있음을 의미할 수 있다.

[0058] 도 6은 본 발명의 일실시예에 있어서, 외부 함수의 실행 코드를 확인하는 과정의 예를 도시한 흐름도이다. 도 6에 나타난 단계들(610 내지 630)은 도 3의 단계(310)에 포함되어 컴퓨터 장치(200)에 의해 수행될 수 있다.

[0059] 단계(610)에서 컴퓨터 장치(200)는 실행 가능 모듈에서 참조하는 외부 함수 정보를 확인할 수 있다. 이미 설명한 바와 같이, 외부 함수 정보는 실행 가능 모듈에 대해 정의되는 IAT(Import Address Table)에 대응할 수

있다.

[0060] 단계(620)에서 컴퓨터 장치(200)는 외부 함수 정보에서 참조하는 외부 함수의 주소를 획득할 수 있다. 일례로, 외부 함수의 주소는 외부 함수 정보에 기록된 외부 함수의 시작 주소를 포함할 수 있다.

[0061] 단계(630)에서 컴퓨터 장치(200)는 외부 함수의 주소에 저장된 실행 코드를 획득할 수 있다.

[0062] 일실시예로, 컴퓨터 장치(200)는 외부 함수의 시작 주소로부터 이동 명령어의 크기에 해당하는 주소까지의 실행 코드를 획득할 수 있다. 여기서, 이동 명령어는 후킹 명령어에 대응할 수 있다. 예를 들어, 외부 함수의 시작 주소에 후킹 명령어가 삽입되는 경우, 외부 함수의 시작 주소로부터 이동 명령어의 크기에 해당하는 주소까지의 실행 코드는 후킹 명령어에 대응할 수 있다. 다시 말해, 후킹이 발생하기 전과 후에 획득되는 실행 코드가 변경될 수 있다. 이러한 실행 코드를 포함하는 식별값이 이전 식별값과 다르다면, 해당 외부 함수에 대한 후킹이 발생하여 다른 함수가 수행되고 있음을 의미할 수 있다.

[0063] 다른 실시예로, 컴퓨터 장치(200)는 외부 함수의 시작 주소에서 종료 주소까지의 실행 코드를 획득할 수 있다. 이는 외부 함수에 대한 전체 실행 코드를 획득하는 것을 의미할 수 있다. 이 경우, 외부 함수에 후킹 명령어가 삽입된다면, 실행 코드가 변경될 수 있다. 이러한 실행 코드를 포함하는 식별값이 이전 식별값과 다르다면, 해당 외부 함수에 대한 후킹이 발생하여 다른 함수가 수행되고 있음을 의미할 수 있다.

[0064] 도 7은 본 발명의 일실시예에 있어서, 내부 함수의 실행코드를 확인하는 과정의 예를 도시한 흐름도이다. 도 7에 나타난 단계들(710 내지 730)은 도 3의 단계(310)에 포함되어 컴퓨터 장치(200)에 의해 수행될 수 있다.

[0065] 단계(710)에서 컴퓨터 장치(200)는 실행 가능 모듈에서 제공하는 내부 함수 정보를 확인할 수 있다. 이미 설명한 바와 같이 내부 함수 정보는 실행 가능 모듈에 대해 정의되는 EAT(Export Address Table)에 대응할 수 있다.

[0066] 단계(720)에서 컴퓨터 장치(200)는 내부 함수 정보에 기초하여 외부에 제공되는 내부 함수의 주소를 획득할 수 있다. 여기서 내부 함수의 주소는 내부 함수의 시작 주소를 포함할 수 있다.

[0067] 단계(730)에서 컴퓨터 장치(200)는 내부 함수의 주소에 저장된 실행 코드를 획득할 수 있다. 이 경우에도 컴퓨터 장치(200)는 내부 함수의 시작 주소로부터 이동 명령어의 크기에 해당하는 주소까지의 실행 코드를 획득하거나 또는 내부 함수의 시작 주소에서 종료 주소까지의 실행 코드를 획득할 수 있다. 후킹 명령어가 내부 함수에 삽입되어 실행 코드가 변경되는 경우, 실행 코드를 포함하는 식별값이 이전 식별값과 달라지게 될 것이며, 따라서 내부 함수에 대한 후킹의 발생을 탐지할 수 있게 된다.

[0068] 식별값은 (1) 외부 함수의 오프셋, (2) 내부 함수의 오프셋, (3) 외부 함수의 실행 코드 또는 (4) 내부 함수의 실행 코드를 포함하도록 생성될 수 있으나, (1) 내지 (4) 중 둘 이상을 포함하도록 생성될 수도 있으며, 이미 설명한 바와 같이, 모듈의 모든 API 함수들의 오프셋들 또는 실행 코드들을 하나의 해쉬 함수로 변환한 값을 식별값으로 생성할 수도 있다.

[0069] 도 8 및 도 9는 본 발명의 실시예들에 따라 오프셋을 획득하는 예들을 도시한 도면들이다. 도 8 및 도 9는 컴퓨터 장치(200)의 메모리(210)상에 적재된 실행 가능 모듈(810)과 외부 모듈(820)의 예를 나타내고 있다. 실행 가능 모듈(810)은 코드 섹션(Code, 811), 데이터 섹션(Data, 812), IAT 섹션(Import, 813) 및 EAT 섹션(Export, 814)를 포함하고 있으며, 이와 유사하게 외부 모듈(820)은 코드 섹션(Code, 821), 데이터 섹션(Data, 822), EAT 섹션(Export, 823) 및 IAT 섹션(Import, 824)를 포함하고 있다.

[0070] 이때, 도 8은 실행 가능 모듈(810)에서 참조하는 외부 함수(825)에 대해, 외부 모듈(820)의 시작 주소와 외부 함수(825)의 시작 주소가 실행 가능 모듈(810)의 IAT 섹션(Import, 813)에 기록될 수 있으며, 이러한 두 시작 주소들의 차이를 통해 외부 함수(825)의 오프셋이 얻어질 수 있음을 나타내고 있다.

[0071] 도 9는 실행 가능 모듈(810)에서 외부로 제공되는 내부 함수(910)의 시작 주소가 실행 가능 모듈(810)의 EAT 섹션(Export, 814)에 기록되어 외부 모듈(920)로 참조될 수 있으며, 이때, 실행 가능 모듈(810)의 시작 주소와 EAT 섹션(Export, 814)에서 확인될 수 있는 내부 함수(910)의 시작 주소의 차이를 통해 내부 함수(910)의 오프셋이 얻어질 수 있음을 나타내고 있다.

[0072] 실행 코드 역시 이러한 외부 함수(825)나 내부 함수(910)의 시작 주소에 기반하여 얻어질 수 있음을 쉽게 이해할 수 있을 것이다.

[0073] 한편, 이상에서는 컴퓨터 프로그램이 실행되는 클라이언트의 단말로서의 컴퓨터 장치(200)에서 현재 식별값과 이전 식별값을 비교하여 후킹 여부를 판단하는 실시예들을 설명하였다. 다른 실시예에서는 서버에서 식별값들

을 군집화하여 클라이언트의 특정 모듈 또는 특정 함수에 대한 후킹 여부를 판단할 수도 있다.

[0074] 예를 들어, 도 3에서 클라이언트가 구현되는 컴퓨터 장치(200)는 단계(330)에서 생성된 식별값을 서버로 전송할 수 있다. 이 경우, 서버는 클라이언트들로부터 수집된 식별값들을 연계된 모듈의 식별자 또는 대상 함수의 식별자에 따라 군집화하고, 군집된 식별값들을 비교하여 군집별로 특정 모듈 또는 특정 대상 함수의 후킹 상태를 결정하도록 구현될 수 있다. 보다 구체적으로 서버는 특정 군집에 포함된 식별값들 중 서로 동일한 식별값들을 정상적인 식별값들로 결정하고, 나머지 식별값들에 대응하는 클라이언트의 특정 모듈 또는 특정 대상 함수를 후킹 상태로 결정하도록 구현될 수 있다. 보다 구체적인 예로, 모듈 A의 함수 B에 대해 1000 개의 클라이언트가 각각 하나씩 총 1000 개의 식별값들을 서버로 전송하였다고 가정한다. 이때, 997개의 식별값들은 서로 동일한 값을 갖는 반면, 3 개의 식별값들이 상이한 값을 갖는 경우, 서버는 서로 상이한 값을 갖는 3개의 식별값들에 대응하는 클라이언트들에서 모듈 A 및/또는 함수 B에 대한 후킹이 이루어진 것으로 판단할 수 있다. 이미 설명한 바와 같이, 오프셋이나 실행 코드는 실행 가능 모듈이나 외부 모듈이 메모리상의 어느 위치에 적재되었는가와 무관하게 동일하기 때문에, 이러한 오프셋이나 실행 코드를 포함하는 식별값이 다른 클라이언트와 상이하다는 점을 이용하여 소수의 상이한 식별값을 전송한 클라이언트들에 대해 해당 모듈 및/또는 함수에 대해 후킹이 발생하였음을 인지할 수 있게 된다.

[0075] 이를 위해, 서버는 클라이언트들이 전송하는 식별값들을 일정기간 동안 군집화할 수 있다. 이후, 서버는 클라이언트들로부터 전송되는 식별값들을 통해 실시간으로 후킹 여부를 판단할 수 있게 된다. 예를 들어, 최초 일정기간 동안의 군집화를 통해 500개의 동일한 식별자 aaa가 수신되었다고 가정한다. 이후, 서버는 클라이언트 1로부터 식별자 aaa가 수신된다면, 군집화된 식별자 aaa의 집합을 통해 실시간으로 클라이언트 1에서 후킹이 발생하지 않았다고 판단할 수 있다. 반면, 클라이언트 2로부터 식별자 aaa와 다른 식별자 bbb가 수신된다면, 서버는 군집화된 식별자 aaa의 집합을 통해 클라이언트 2에서 후킹이 발생하였다고 실시간으로 판단할 수 있게 된다. 다른 실시예로, 서버는 일정 기간마다 클라이언트들의 후킹 여부를 판단할 수도 있다. 예를 들어, 서버는 일정 기간마다 해당 기간 동안 클라이언트들이 전송하는 식별값들을 군집화하고, 군집화된 식별값들을 비교하여 소수의 상이한 식별값을 전송한 클라이언트들에 대해 해당 모듈 및/또는 함수에 대해 후킹이 발생하였음을 인지할 수 있다.

[0076] 도 10은 본 발명의 일실시예에 있어서, 서버의 후킹 탐지 방법의 예를 도시한 흐름도이다. 본 실시예에 따른 후킹 탐지 방법은 다수의 클라이언트들로부터 식별값들을 수신하는 서버의 컴퓨터 장치(200)에 의해 수행될 수 있다. 이때, 컴퓨터 장치(200)의 프로세서(220)는 메모리(210)가 포함하는 운영체제의 코드나 적어도 하나의 컴퓨터 프로그램의 코드에 따른 제어 명령(instruction)을 실행하도록 구현될 수 있다. 여기서, 프로세서(220)는 컴퓨터 장치(200)에 저장된 코드가 제공하는 제어 명령에 따라 컴퓨터 장치(200)가 도 10의 방법이 포함하는 단계들(1010 내지 1030)을 수행하도록 컴퓨터 장치(200)를 제어할 수 있다.

[0077] 단계(1010)에서 컴퓨터 장치(200)는 복수의 클라이언트들로부터 식별값들을 수신할 수 있다. 이미 설명한 바와 같이, 식별값들 각각은 복수의 클라이언트들의 메모리에 적재된 실행 가능 모듈과 관련하여 실행 가능 모듈이 제공하는 내부 함수 또는 실행 가능 모듈이 참조하는 외부 함수로서의 대상 함수의 오프셋 또는 실행 코드에 기반하여 생성될 수 있다. 식별값은 (1) 외부 함수의 오프셋, (2) 내부 함수의 오프셋, (3) 외부 함수의 실행 코드 또는 (4) 내부 함수의 실행 코드를 포함하도록 생성될 수 있으나, (1) 내지 (4) 중 둘 이상을 포함하도록 생성될 수도 있으며, 이미 설명한 바와 같이, 모듈의 모든 API 함수들의 오프셋들 또는 실행 코드들을 하나의 해쉬 함수로 변환한 값을 식별값으로 생성할 수도 있다.

[0078] 단계(1020)에서 컴퓨터 장치(200)는 수신된 식별값들을 대응하는 모듈의 식별자 또는 대응하는 대상 함수의 식별자에 따라 군집화할 수 있다. 다시 말해, 모듈별 및/또는 함수별로 식별값들이 분류될 수 있다.

[0079] 단계(1030)에서 컴퓨터 장치(200)는 임의의 군집에 대한 식별값들을 비교하여 임의의 군집에 대응하는 모듈 또는 대상 함수의 후킹 상태를 결정할 수 있다. 예를 들어, 컴퓨터 장치(200)는 임의의 군집에 포함된 식별값들 중 서로 동일한 식별값들을 정상적인 식별값들로 결정할 수 있으며, 임의의 군집에 포함된 식별값들 중 정상적인 식별값들로 결정된 식별값들을 제외한 나머지 식별값에 대응하는 클라이언트의 모듈 또는 대상 함수를 후킹 상태로 결정할 수 있다.

[0080] 이와 같이, 본 발명의 실시예들에 따르면, 실행 가능한 모듈이 외부로부터 참조하거나 외부로 제공하는 함수의 위치 정보 또는 실행 코드에 대한 고유한 값을 특정하고, 이러한 고유한 값들을 비교하거나 또는 고유한 값들을 군집화하여 얻어지는 분류를 이용하여 API 후킹을 판단할 수 있다.

[0081] 이상에서 설명된 시스템 또는 장치는 하드웨어 구성요소, 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 어플리케이션을 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.

[0082] 소프트웨어는 컴퓨터 프로그램(computer program), 코드(code), 명령(instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성요소(component), 물리적 장치, 가상 장치(virtual equipment), 컴퓨터 저장 매체 또는 장치에 구체화(embody)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록매체에 저장될 수 있다.

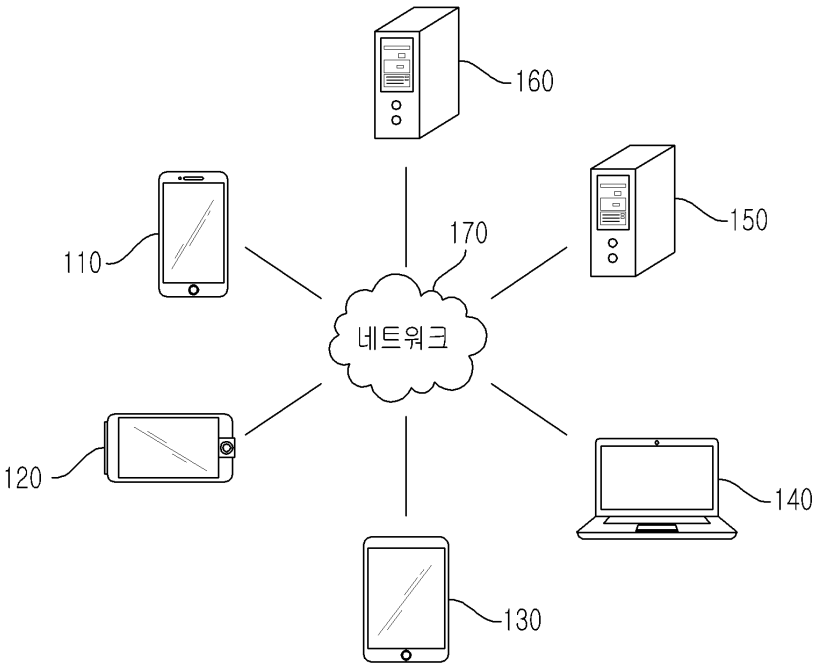
[0083] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 매체는 컴퓨터로 실행 가능한 프로그램을 계속 저장하거나, 실행 또는 다운로드를 위해 임시 저장하는 것일 수도 있다. 또한, 매체는 단일 또는 수개 하드웨어가 결합된 형태의 다양한 기록수단 또는 저장수단일 수 있는데, 어떤 컴퓨터 시스템에 직접 접속되는 매체에 한정되지 않고, 네트워크 상에 분산 존재하는 것일 수도 있다. 매체의 예시로는, 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체, CD-ROM 및 DVD와 같은 광기록 매체, 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical medium), 및 ROM, RAM, 플래시 메모리 등을 포함하여 프로그램 명령어가 저장되도록 구성된 것이 있을 수 있다. 또한, 다른 매체의 예시로, 애플리케이션을 유통하는 앱 스토어나 기타 다양한 소프트웨어를 공급 내지 유통하는 사이트, 서버 등에서 관리하는 기록매체 내지 저장매체도 들 수 있다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다.

[0084] 이상과 같이 실시예들이 비록 한정된 실시예와 도면에 의해 설명되었으나, 해당 기술분야에서 통상의 지식을 가진 자라면 상기의 기재로부터 다양한 수정 및 변형이 가능하다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다.

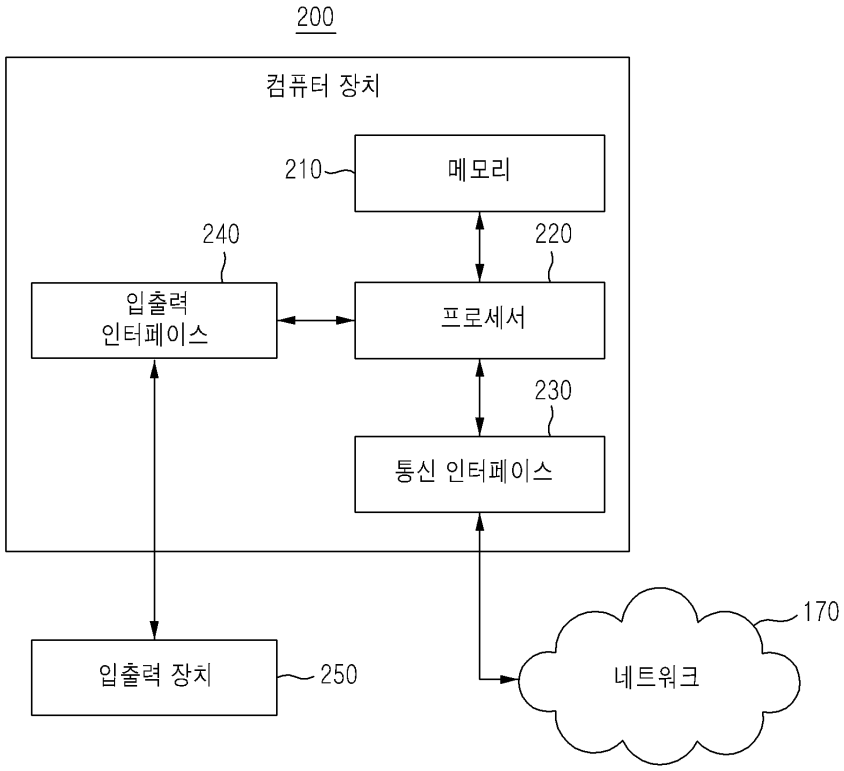
[0085] 그러므로, 다른 구현들, 다른 실시예들 및 청구범위와 균등한 것들도 후술하는 청구범위의 범위에 속한다.

도면

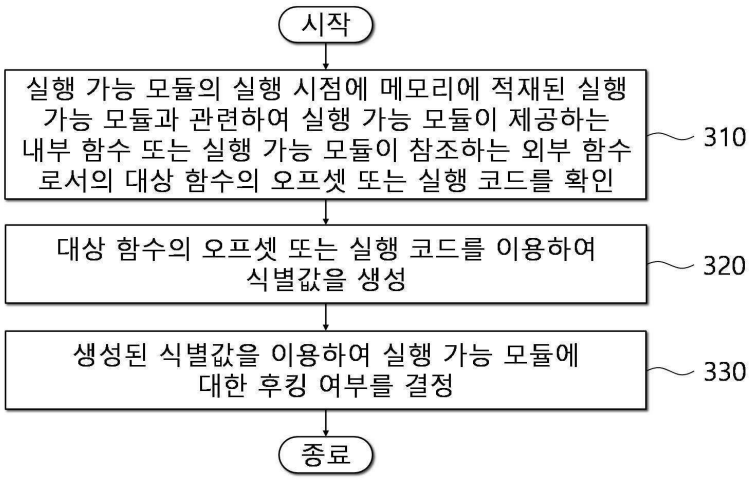
도면1


160
150
110
170
네트워크
120
140
130

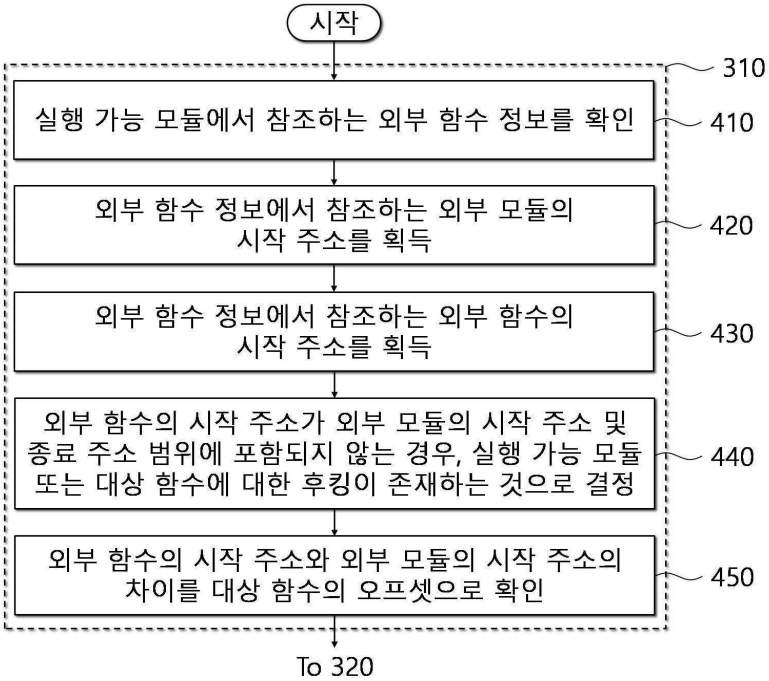
도면2


200
컴퓨터 장치
210
메모리
240
220
입출력
인터페이스
프로세서
230
통신 인터페이스
170
입출력 장치
250
네트워크

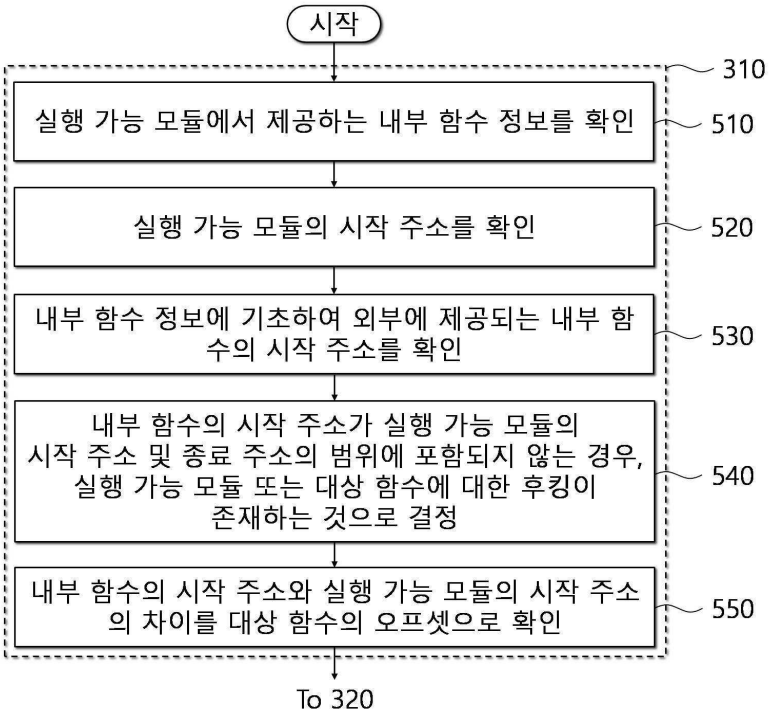
도면3


시작
실행 가능 모듈의 실행 시점에 메모리에 적재된 실행 가능 모듈과 관련하여 실행 가능 모듈이 제공하는 내부 함수 또는 실행 가능 모듈이 참조하는 외부 함수로서의 대상 함수의 오프셋 또는 실행 코드를 확인
310
대상 함수의 오프셋 또는 실행 코드를 이용하여 식별값을 생성
320
생성된 식별값을 이용하여 실행 가능 모듈에 대한 후킹 여부를 결정
330
종료

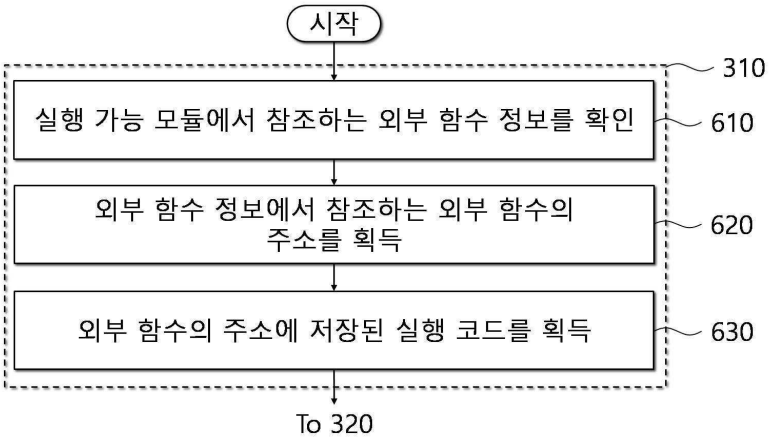
도면4


시작
310
실행 가능 모듈에서 참조하는 외부 함수 정보를 확인
410
외부 함수 정보에서 참조하는 외부 모듈의 시작 주소를 획득
420
외부 함수 정보에서 참조하는 외부 함수의 시작 주소를 획득
430
외부 함수의 시작 주소가 외부 모듈의 시작 주소 및 종료 주소 범위에 포함되지 않는 경우, 실행 가능 모듈 또는 대상 함수에 대한 후킹이 존재하는 것으로 결정
440
외부 함수의 시작 주소와 외부 모듈의 시작 주소의 차이를 대상 함수의 오프셋으로 확인
450
To 320

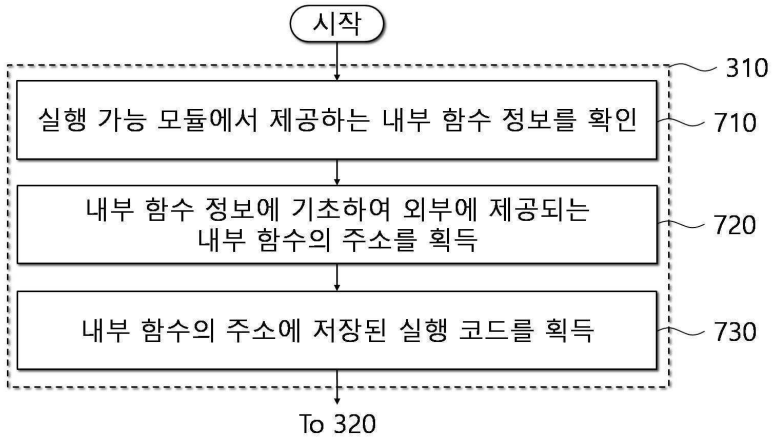
도면5


시작
310
실행 가능 모듈에서 제공하는 내부 함수 정보를 확인
510
실행 가능 모듈의 시작 주소를 확인
520
내부 함수 정보에 기초하여 외부에 제공되는 내부 함수의 시작 주소를 확인
530
내부 함수의 시작 주소가 실행 가능 모듈의 시작 주소 및 종료 주소의 범위에 포함되지 않는 경우, 실행 가능 모듈 또는 대상 함수에 대한 후킹이 존재하는 것으로 결정
540
내부 함수의 시작 주소와 실행 가능 모듈의 시작 주소의 차이를 대상 함수의 오프셋으로 확인
550
To 320

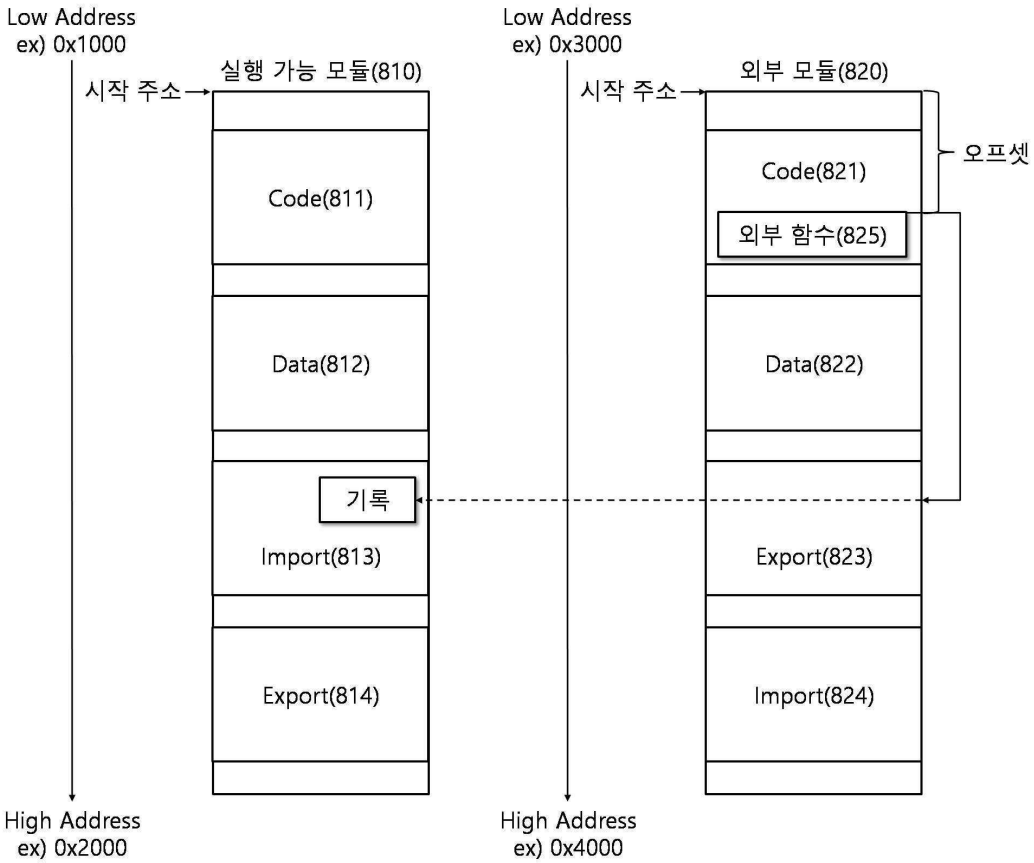
도면6


시작
310
실행 가능 모듈에서 참조하는 외부 함수 정보를 확인
610
외부 함수 정보에서 참조하는 외부 함수의 주소를 획득
620
외부 함수의 주소에 저장된 실행 코드를 획득
630
To 320

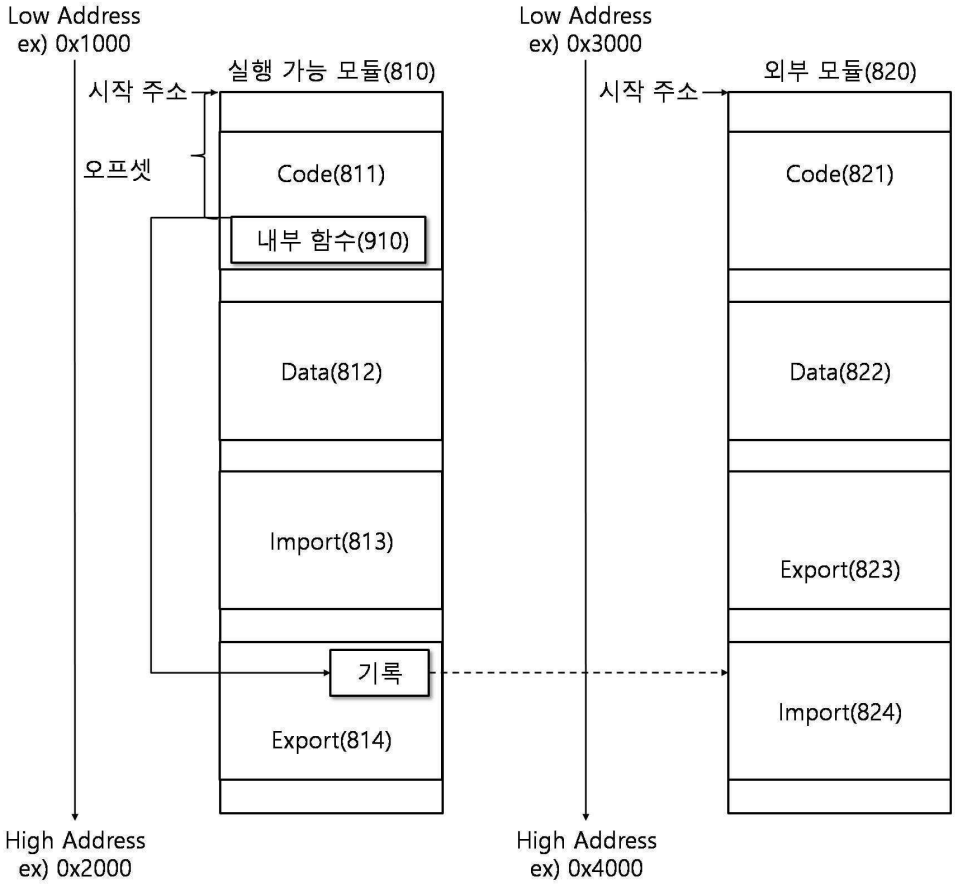
도면7


시작
310
실행 가능 모듈에서 제공하는 내부 함수 정보를 확인
710
내부 함수 정보에 기초하여 외부에 제공되는
내부 함수의 주소를 획득
720
내부 함수의 주소에 저장된 실행 코드를 획득
730
To 320

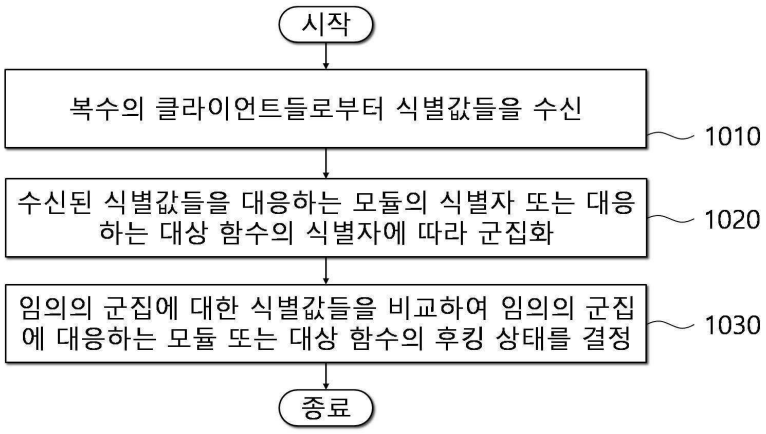
도면8


Low Address
ex) 0x1000
시작 주소
실행 가능 모듈(810)
Code(811)
Data(812)
기록
Import(813)
Export(814)
High Address
ex) 0x2000
Low Address
ex) 0x3000
시작 주소
외부 모듈(820)
Code(821)
오프셋
외부 함수(825)
Data(822)
Export(823)
Import(824)
High Address
ex) 0x4000

도면9


Low Address
ex) 0x1000
실행 가능 모듈(810)
시작 주소
오프셋
Code(811)
내부 함수(910)
Data(812)
Import(813)
기록
Export(814)
High Address
ex) 0x2000
Low Address
ex) 0x3000
외부 모듈(820)
시작 주소
Code(821)
Data(822)
Export(823)
Import(824)
High Address
ex) 0x4000

도면10


시작
복수의 클라이언트들로부터 식별값들을 수신
1010
수신된 식별값들을 대응하는 모듈의 식별자 또는 대응하는 대상 함수의 식별자에 따라 군집화
1020
임의의 군집에 대한 식별값들을 비교하여 임의의 군집에 대응하는 모듈 또는 대상 함수의 후킹 상태를 결정
1030
종료