(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2013년02월06일
(11) 등록번호 10-1230271
(24) 등록일자 2013년01월31일

(51) 국제특허분류(Int. Cl.)
G06F 21/20 (2006.01) *G06F 17/40* (2006.01)
G06F 17/30 (2006.01) *G06F 17/10* (2006.01)
(21) 출원번호 10-2010-0134992
(22) 출원일자 2010년12월24일
심사청구일자 2010년12월24일
(65) 공개번호 10-2012-0073018
(43) 공개일자 2012년07월04일
(56) 선행기술조사문헌
정보보호학회 논문지 제16권 제3호
KR1020090065277 A
KR1020100084488 A
정보보호학회논문지 제21권 제2호

(73) 특허권자
고려대학교 산학협력단
한국인터넷진흥원
(72) 발명자
정현철
임채태

(뒷면에 계속)

(74) 대리인
특허법인다울

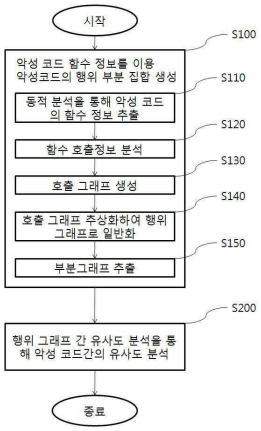
전체 청구항 수 : 총 10 항 심사관 : 엄인권

(54) 발명의 명칭 **악성 코드 탐지를 위한 시스템 및 방법**

(57) 요 약

본 발명은 봇넷 탐지 정보의 분석 시스템 및 방법에 관한 것으로, 트래픽 수집 시스템으로 부터 수신된 정보를 이용하여 봇넷을 탐지하고 행위를 분석하는 봇넷 탐지 정보의 분석 시스템에 있어서, 상기 트래픽 수집 시스템으로부터 전송된 그룹데이터들에 대해 봇넷 그룹을 판정하는 봇넷 탐지 엔진과, 상기 트래픽 수집 시스템과 봇넷 탐지 엔진의 다양한 탐지 정보를 하나의 봇넷 탐지 정보로 출력하는 통합 분석 엔진을 포함하는 것을 특징으로 하는 봇넷 탐지 정보의 분석 시스템 및 이의 분석 방법을 제공한다.

대 표 도 - 도1



(72) 발명자 **권종훈**

지승구

이주석

오주형

강동완 **이제현**

이희조 **김태범**

이 발명을 지원한 국가연구개발사업

과제고유번호	KI001863
부처명	지식경제부
연구사업명	IT산업원천 기술개발사업
연구과제명	신종 봇넷 능동형 탐지 및 대응 기술 개발
주관기관	한국인터넷진흥원
연구기간	2010.03.01 ~ 2011.02.28

특허청구의 범위

청구항 1

신, 변종 악성코드를 효과적으로 분석하고 분류할 수 있는 악성 코드 탐지를 위한 시스템에 있어서,

적어도 하나 이상의 악성 코드의 동적 정보 분석을 통해 행위 부분 집합을 생성하는 행위 감지 모듈; 및

상기 행위 감지 모듈에 의한 악성 코드의 행위 유사성을 판단하는 유사도 판별 모듈을 포함하고,

상기 행위 감지 모듈은,

악성 코드의 함수 호출 정보를 추출하기 위한 호출 정보 추출 모듈;

상기 추출된 함수 호출 정보를 분석하여 행위 노드, 호출 순서와 호출 함수 집합을 갖는 호출 그래프를 생성하는 호출 그래프 생성 모듈;

함수들의 목적에 따라 추상화된 행위 노드에 의해 호출 그래프를 행위 그래프로 일반화하는 행위 그래프 생성 모듈; 및

n개의 노드를 갖는 악성 코드 M에 대하여 행위 그래프에서 노드를 차례로 줄여가며 가능한 모든 부분 그래프를 추출하는 부분 그래프 추출 모듈을 포함하는 것을 특징으로 하는 악성 코드 탐지를 위한 시스템.

청구항 2

삭제

청구항 3

청구항 1에 있어서,

악성 코드의 함수 호출 정보로 API를 사용하는 것을 특징으로 하는 악성 코드 탐지를 위한 시스템.

청구항 4

청구항 3에 있어서,

상기 추상화된 행위 노드는 수많은 API들을 32개의 카테고리로 분류하고, 이들을 각기 다시 4개의 행위로 분류하여 총 128개의 행위 노드로 추상화하는 것을 특징으로 하는 악성 코드 탐지를 위한 시스템.

청구항 5

청구항 1에 있어서,

상기 유사도 판별 모듈은 서로 다른 악성 코드들에서 추출한 부분 그래프에 대한 유사도를 판별하는 것을 특징으로 하는 악성 코드 탐지를 위한 시스템.

청구항 6

신, 변종 악성코드를 효과적으로 분석하고 분류할 수 있는 악성 코드 탐지 방법에 있어서,

악성 코드 함수 정보를 이용하여 행위 그래프를 포함하는 악성 코드의 행위 부분 집합을 생성하는 단계; 및

상기 생성된 행위 그래프 간의 유사도 분석을 통해 악성 코드간의 유사도를 분석하는 단계를 포함하고,

상기 악성 코드의 행위 부분 집합을 생성하는 단계는,

API 호출 정보를 이용하여 호출 그래프를 생성하는 단계;

상기 호출 그래프의 추상화를 통한 행위 그래프로 변환하는 단계; 및

악성 코드의 모듈별 행위 특징을 분석하기 위해 상기 행위 그래프에서 부분 그래프를 추출하는 단계를 포함하는 것을 특징으로 하는 악성 코드 탐지 방법.

청구항 7

삭제

청구항 8

청구항 6에 있어서, 상기 호출 그래프를 생성하는 단계는,

동적 분석을 통해 악성 코드의 API를 추출하는 단계;

API 정보를 분석하는 단계; 및

상기 API 정보를 이용하여 행위 노드, 호출 순서와 호출 함수 집합을 갖는 호출 그래프를 생성하는 단계를 포함하는 것을 특징으로 하는 악성 코드 탐지 방법.

청구항 9

청구항 6에 있어서,

상기 호출 그래프의 추상화는, 상기 API를 32개의 카테고리로 분류하고, 이들을 각기 다시 4개의 행위로 분류하여 총 128개의 행위 노드로 추상화하는 것을 특징으로 하는 악성 코드 탐지 방법.

청구항 10

청구항 6에 있어서, 상기 부분 그래프 추출 단계는,

n개의 노드를 가진 악성 코드 M에 대한 행위 그래프에서 노드를 차례로 감소시켜 추출하는 것을 특징으로 하는 악성 코드 탐지 방법.

청구항 11

청구항 10에 있어서,

상기 n의 최소값은 3이고, n개의 노드를 가진 악성 코드 M에 대한 최대 부분 그래프(SG(Mn)i)의 개수는 하기 수학식과 같은 것을 특징으로 하는 악성 코드 탐지 방법.

수학식

$$|SG(M_n)_i| = \sum_{k=1}^{n-2}(k)$$

청구항 12

청구항 6에 있어서,

상기 유사도 분석에 의한 유사도 지수Sim은 변종 악성 코드를 각기 M, M'로 하고, 부분 그래프를 SG(M)i, SG(M')로 할 경우 하기 수학식과 같은 것을 특징으로 하는 악성 코드 탐지 방법.

수학식

$$Sim(M, M') = \frac{\sqrt{\sum [SG(M, M')_k]^2}}{n(M')}$$

명 세 서

기 술 분 야

[0001] 본 발명은 악성 코드 탐지를 위한 시스템 및 방법에 관한 것으로, 행위 그래프 분석을 통한 악성코드 모듈별 유사도 분석 기법을 통해 악성코드 변종 및 신종 악성코드를 빠르게 탐지할 수 있는 악성 코드 탐지를 위한 시스템 및 방법을 제공한다.

배 경 기 술

[0002] 네트워크 및 컴퓨터의 발전에 따라 악성코드 역시 폭발적인 증가 추이를 보이고 있으며, 새로운 악성코드의 출현과 더불어 기존의 악성코드를 이용한 변종 역시 큰 몫을 차지하고 있다. 특히 실행압축 기술과 코드 난독화를 이용한 변종들은 제작이 쉬울 뿐만 아니라, 자신의 시그너쳐 혹은 구문적 특징를 변조할 수 있어, 악성코드 제작자들이 널리 사용하는 기술이다.

[0003] 악성코드는 사용자가 알지 못하는 사이 컴퓨터 시스템에 침입, 설치되어 시스템이나 네트워크에 피해를 주고, 불법적으로 정보를 취득하도록 설계된 소프트웨어를 의미한다. 악성코드는 그 목적이나 행위 특성에 따라 트로잔, 웜, 바이러스, 봇 등으로 분류할 수 있으며, 분산 서비스 거부 공격, 스팸 메일 발송, 피싱 사이트 유도, 개인 정보 탈취 등 다양한 형태의 공격을 통해 불법적인 금전적 취득에 이용되고 있다. 이러한 악성코드의 위협에 대응하기 위해, 현재 다양한 악성코드 분석 및 탐지 연구가 활발하게 진행되고 있지만, 날이 갈수록 지능화되고 정교해지는 악성코드들에 대응하기에는 많은 한계가 따르는 것이 현실이다.

[0004] 악성코드 대응에 있어 가장 현실적인 어려움은 악성코드 종류의 폭발적인 증가이다. 2010년도 시만텍 사의 보고서에 따르면, 2006년부터 2008년 사이 새로운 악성코드의 지수적 증가는 해마다 두 배에 이르렀으며, 2009년 한 해에만 약 3백만 개의 새로운 악성코드가 발견되었다고 보고하였다. 이러한 악성코드의 폭발적인 증가는 코드 난독화 및 실행 압축 기술 등을 이용한 변종 제작과 밀접한 관계가 있다.

[0005] 코드 난독화란, 해당 코드의 기능적, 의미론적 특징은 유지한 채, 외형적 구조를 변경하는 기술이다. 따라서 행위적 본질은 유지한 채, 외형에 변화를 가져올 수 있다는 점에서 악성코드 개발자들이 널리 사용하는 기술이다. 또한 실행압축 기술 역시 코드 난독화의 일종으로, 실행 과정 중 압축 해제를 통해 외형적 변화를 달성할 수 있다. 이러한 기술들은 간단한 툴을 이용하여 쉽게 적용 가능하며, 새로운 악성코드 제작보다 적은 노력과 비용으로 다수의 변종을 쉽게 생성할 수 있어, 악성코드 제작자들이 널리 사용하고 있다. 최근 연구에 따르면, 약 80% 이상의 악성코드에서 코드 난독화 및 실행압축 기술이 사용되고 있다고 보고되었으며, 이는 안티바이러스 업체들에게 많은 부담으로 작용하고 있다.

[0006] 현재 악성코드 탐지 및 대응을 위해 안티바이러스 업체에서 가장 일반적으로 사용하고 있는 방법은 시그너쳐 기반의 탐지 기법이다. 시그너쳐 탐지 기법이란, 악성코드가 가지고 있는 고유한 바이너리 형태를 시그너쳐로 등록하고, 특정 바이너리 내에 해당 시그너쳐가 존재하는 지를 검사하여 탐지하는 기법이다. 따라서 악성코드의 지수적 증가는 곧 악성코드 시그너쳐의 증가로 이어지며, 안티바이러스 업체들에게 많은 인적, 금전적 노력을 새로운 시그너쳐 생성에 강요하게 되어 부담으로 이어질 수 밖에 없다. 또한 새로운 형태의 악성코드 등장과 새로운 시그너쳐 등록까지 많은 시간이 소요되므로, 그 간의 피해는 감수할 수 밖에 없게된다.

[0007] 현재까지도 안티바이러스 업체들이 보편적으로 사용하는 방법은 시그너쳐 기반의 악성코드 탐지기법이다. 하지만 코드 난독화 기술 등이 널리 이용되면서 악성코드들은 이러한 시그너쳐 기반의 탐지를 쉽게 우회할 수 있게 되었다. 이러한 문제점을 타개하기 위해, 관련 연구자들 사이에서 행위 기반의 악성코드 탐지는 하나의 새로운 이슈로 떠오르게 되었다. 대표적으로 나이브 베이즈 방법(Naive Bayes method[]), SVM(Support vector machine), 그리고 디씨젼 트리 클래시파이어(Decision Tree classifiers)와 같은 데이터 마이닝과 기계 학습기법을 이용한 탐지 기법이 발표되었다.

[0008] 악성코드를 분석하는 방법은 크게 정적 분석과 동적 분석 두 가지로 분류될 수 있다. 정적 분석은 악성코드를 실행시키지 않고 분석하는 방법으로, 현재 상업용 안티바이러스 업체를 포함한 악성코드 분석가들 사이에서 가장 널리 사용하는 방법이다. 바이너리 패턴 매칭, 데이터 플로우와 코드 플로우 분석 등이 대표적인 정적 분석 기법의 하나이다. 이러한 정적 분석 기법은 악성코드의 실행을 배재하기 때문에 안전하고 빠른 분석이 용이하다는 장점을 가지고 있다. 하지만 실행 압축을 이용한 코드 난독화를 수행하는 악성코드의 경우, 정확한 분석이 쉽지 않은 단점을 갖고 있다.

[0009] 이러한 정적 분석의 단점을 극복하기 위해 다양한 연구가 진행되어왔다. 특히 코드 난독화를 이용한 악성코드로부터 원래 형태의 코드를 추출하기 위한 연구들이 발표되었으며, 흔히 코드 일반화 기술이라고 불리우고 있다.

[0010] 대표적인 코드 일반화 연구로는 치스토도레스쿠(Chistodorescu), 와렌스테인(Walenstein) 등이 발표한 연구가 있다. 치스토도레스쿠(Chistodorescu)의 연구는 난독화된 실행파일로부터 일반화된 코드 원형을 추출하므로서 탐지 성능을 향상시켰다. 또한 와렌스테인(Walenstein) 연구팀은 명령 수행 순서를 유한집합(finite set)을 이용하여 치환하므로서 난독화된 코드를 비실행 상태에서 일반화를 시켰다. 이들의 연구 성과는 코드 일반화 기술

에 대한 긍정적 가능성을 널리 알렸지만, 특정 코드 난독화 기술에 한정되어 있다는 점에서 한계가 존재한다.

[0011] 프레다(Preda)등은 의미론적 분석을 통해 난독화된 악성코드 분석을 수행하였다. 이 연구들은 특정 API 호출 감시를 이용한 악성코드의 의미론적 모델을 추출할 수 있었으며, 높은 탐지 성능을 보였다. 하지만 안타깝게도, 위 연구들은 특정 API의 발생 빈도에 의존적이어서 레드 해링 시스템 콜(red herring system call)과 같은 새로운 형태의 코드 난독화 기술에 취약한 단점이 있다.

[0012] 위 연구들과 같은 다양한 노력에도 불구하고, 정적 분석 기법에는 분석 정확도 측면에서 여전히 많은 어려움이 존재하고 있다. 이러한 어려움을 극복하기 위해 제안된 새로운 형태의 분석 접근법이 동적 분석 기법이다. 동적 분석은 가상 머신과 같은 제어 가능한 환경 속에서 악성코드를 동작시켜 그 행위를 분석하는 기법으로, 실행압축과 같은 코드 난독화와 무관하게 정확한 실제 행위를 볼 수 있다는 장점을 가지고 있다. 윌리엄스(Williams) 연구팀이 보인 CWSandbox와 TTAnalyze들이 대표적인 동적 분석 기법을 이용한 분석 연구로서, 현재까지도 많은 연구자들이 활용하고 있다. 물론 동적 분석 기법에도 단점은 존재한다. 그것은 실제 악성코드 실행에 따르는 실험환경의 오염 가능성과 행위 관찰을 위해 많은 시간이 소요된다는 사실이다. 하지만 이러한 단점은, 보다 정확한 악성코드 분석을 위한 트레이드 오프(Trade-off)로서 간주할 수 있으며, 본 연구에서도 보다 정확한 분석을 위해 동적 분석기법을 기반으로 하고 있다.

발명의 내용

해결하려는 과제

[0013] 기존의 악성코드 탐지 기법들은 일반적으로 악성코드의 문법적 구조를 기반으로 이루어져 있기 때문에 코드 난독화와 같은 우회기법을 이용하여 회피가 가능하였다.

[0014] 따라서, 본 발명은 상기의 제반 문제를 해결하기 위하여 창출된 것으로, 동적 분석을 통한 의미론적 행위 모델 분석 기법으로 호출 그래프 생성, 추상화를 통한 행위 그래프로의 변환, 부분 그래프 추출 및 행위 유사도 분석을 통해 악성코드를 보다 효율적이고 정확하게 탐지할 수 있는 악성 코드 탐지를 위한 시스템 및 방법을 제공한다.

과제의 해결 수단

[0015] 본 발명에 따른 신, 변종 악성코드를 효과적으로 분석하고 분류할 수 있는 악성 코드 탐지를 위한 시스템에 있어서, 적어도 하나 이상의 악성 코드의 동적 정보 분석을 통해 행위 부분 집합을 생성하는 행위 감지 모듈 및 상기 행위 감지 모듈에 의한 악성 코드의 행위 유사성을 판단하는 유사도 판별 모듈을 포함하는 것을 특징으로 하는 악성 코드 탐지를 위한 시스템을 제공한다.

[0016] 상기 행위 감지 모듈은, 악성 코드의 함수 호출 정보를 추출하기 위한 호출 정보 추출 모듈과, 상기 추출된 함수 호출 정보를 분석하여 행위 노드, 호출 순서와 호출 함수 집합을 갖는 호출 그래프를 생성하는 호출 그래프 생성 모듈과, 함수들의 목적에 따라 추상화된 행위 노드에 의해 호출 그래프를 행위 그래프로 일반화하는 행위 그래프 생성 모듈 및 n개의 노드를 갖는 악성 코드 M에 대하여 행위 그래프에서 노드를 차례로 줄여가며 가능한 모든 부분 그래프를 추출하는 부분 그래프 추출 모듈을 포함한다.

[0017] 악성 코드의 함수 호출 정보로 API를 사용한다.

[0018] 상기 추상화된 행위 노드는 수많은 API들을 32개의 카테고리로 분류하고, 이들을 각기 다시 4개의 행위로 분류하여 총 128개의 행위 노드로 추상화한다.

[0019] 상기 유사도 판별 모듈은 서로 다른 악성 코드들에서 추출한 부분 그래프에 대한 유사도를 판별한다.

[0020] 또한, 본 발명에 따른 신, 변종 악성코드를 효과적으로 분석하고 분류할 수 있는 악성 코드 탐지 방법에 있어서, 악성 코드 함수 정보를 이용하여 행위 그래프를 포함하는 악성 코드의 행위 부분 집합을 생성하는 단계 및 상기 생성된 행위 그래프 간의 유사도 분석을 통해 악성 코드간의 유사도를 분석하는 단계를 포함하는 것을 특징으로 하는 악성 코드 탐지 방법을 제공한다.

[0021] 상기 악성 코드의 행위 부분 집합을 생성하는 단계는, API 호출 정보를 이용하여 호출 그래프를 생성하는 단계와, 상기 호출 그래프의 추상화를 통한 행위 그래프로 변환하는 단계 및 악성 코드의 모듈별 행위 특징을 분석하기 위해 상기 행위 그래프에서 부분 그래프를 추출하는 단계를 포함한다.

[0022] 상기 호출 그래프를 생성하는 단계는, 동적 분석을 통해 악성 코드의 API를 추출하는 단계와, API 정보를 분석하는 단계 및 상기 API 정보를 이용하여 행위 노드, 호출 순서와 호출 함수 집합을 갖는 호출 그래프를 생성하는 단계를 포함한다.

[0023] 상기 호출 그래프의 추상화는, 상기 API를 32개의 카테고리로 분류하고, 이들을 각기 다시 4개의 행위로 분류하여 총 128개의 행위 노드로 추상화한다.

[0024] 상기 부분 그래프 추출 단계는, n개의 노드를 가진 악성 코드 M에 대한 행위 그래프에서 노드를 차례로 감소시켜 추출한다.

[0025] 상기 n의 최소값은 3이고, n개의 노드를 가진 악성 코드 M에 대한 최대 부분 그래프(SG(Mn)i)의 개수는 하기 수학식과 같은 것을 특징으로 한다.

[0026] 수학식

[0027] $$|SG(M_n)_i| = \sum_{k=1}^{n-2}(k)$$

[0028] 상기 유사도 분석에 의한 유사도 지수Sim은 변종 악성 코드를 각기 M, M'로 하고, 부분 그래프를 SG(M)i, SG(M')로 할 경우 하기 수학식과 같은 것을 특징으로 한다.

[0029] 수학식

[0030] $$Sim(M, M') = \frac{\sqrt{\sum [SG(M, M')_k]^2}}{n(M')}$$

발명의 효과

[0031] 상술한 바와 같이 본 발명은 신, 변종 악성코드를 효과적으로 분석하고 분류할 수 있는 새로운 알고리즘을 제안하였다.

[0032] 그리고, 악성코드의 API 호출 순서와 행위 추상화를 이용하여, 악성코드의 고유한 행위 모델을 추출하여 코드 난독화를 이용한 변종 악성코드의 효율적인 분석 및 분류를 할 수 있다. 악성코드의 폭발적인 증가에 따른 시그너쳐의 증가의 문제점 역시, 고유한 행위 시그너쳐 생성을 이용하여 해결할 수 있다. 또한 부분 행위 특징 분석을 이용하여, 악성코드 간 공유되는 부분 행위를 추적하고, 새로운 악성코드 분류 기준을 제시할 수 있다.

도면의 간단한 설명

[0033] 도 1은 본 발명의 일 실시예에 따른 악성 코드 탐지 방법을 설명하기 위한 흐름도.

도 2 일 실시예에 따른 호출 그래프의 도면.

도 3은 도 2의 호출 그래프를 추상화한 행위 그래프.

도 4는 일 실시예에 따른 악성 코드의 의미론적 행위 그래프 분포도.

도 5는 일실시예에 따른 Trojan.Downloader.Win32.Multdl 변종들의 의미론적 행위 그래프.

도 6은 일 실시예에 따른 윔에서 나타나는 공통 행위 부분 그래프.

도 7은 본 발명의 일 실시예에 따른 악성 코드 탐지를 위한 시스템의 개념도.

발명을 실시하기 위한 구체적인 내용

[0034] 이하, 첨부된 도면을 참조하여 본 발명의 실시예를 더욱 상세히 설명하기로 한다. 그러나 본 발명은 이하에서 개시되는 실시예에 한정되는 것이 아니라 서로 다른 다양한 형태로 구현될 것이며, 단지 본 실시예들은 본 발명의 개시가 완전하도록 하며, 통상의 지식을 가진 자에게 발명의 범주를 완전하게 알려주기 위해 제공되는 것이다. 도면상에서 동일 부호는 동일한 요소를 지칭한다.

[0035] 앞서 언급한 바와 같이 악성 코드는 코드 난독화나 실행 압축 기술을 이용하더라도 본질적으로 유지되는 특성이 있다. 이는 악성 코드가 목적을 달성하기 위해 수행하는 행위들이 있다. 따라서, 본 실시예에서는 악성코드의 의미론적 행위 모델을 이용한 탐지 기법을 사용한다. 이때, 의미론적 행위 모델이란 상기와 같이 악성 코드가 목적을 달성하기 위해 수행하는 행위를 의미한다.

[0036] 본 실시예에서는 먼저 악성 코드의 행위 모델을 추상화하였고, 이를 동적 분석을 실시하여 행위 그래프를 재구성하였다. 이와 같이 추상화된 행위 그래프는 각각의 악성코드가 가지는 의미론적 행위 모델을 표현하게 된다. 또한, 이 행위 그래프를 가능한 모든 부분 그래프 단위로 분할하고, 서로 다른 악성코드에서 추출된 부분 그래프들을 상호비교하여, 악성코드 간 행위 유사 정도는 물론, 악성코드 간 공유되고 있는 행위 모델들을 분석하였다.

[0037] 도 1은 본 발명의 일 실시예에 따른 악성 코드 탐지 방법을 설명하기 위한 흐름도이다. 도 2 일 실시예에 따른 호출 그래프의 도면이고, 도 3은 도 2의 호출 그래프를 추상화한 행위 그래프이다.

[0038] 먼저, 도 1에 도시된 바와 같이 악성 코드의 함수 호출 정보를 이용하여 악성 코드의 행위 부분 집합을 생성한다(S100). 즉, 본 실시예에서는 동적 분석을 통해 악성 코드들의 API 호출 순서를 추상화된 그래프로 변환한다. 이어서, 추상화된 그래프를 이용하여 부분 그래프를 추출함으로써 행위 부분 집합을 정리하였다.

[0039] 이를 위해 본 실시예에서는 먼저, 호출 그래프를 생성한다. 동적 분석을 통해 악성 코드의 함수 호출 정보를 추출하고(S110), 이를 바탕으로 함수 호출 정보를 분석한다(S120). 그리고, 관찰된 호출 정보를 이용하여 도 2에서와 같이 호출 그래프를 생성한다(S130). 즉, 악성코드의 악성행위 분석 기법에는 다양한 방법이 존재한다. 그 중 가장 널리 사용되는 방법은 악성코드 발생시키는 함수호출 정보를 분석하는 것이다. 특히 악성 코드의 API 함수 호출 정보를 관찰 분석하였다.

[0040] 본 출원인은 악성코드의 함수 호출 정보를 추출하기 위해 가상머신을 이용한 동적 분석을 수행하였다. 우선 가상 머신을 통해 제어 가능한 환경을 구축하고,시스템 와이드 윈도우즈 후크(System-wide Windows Hook)을 이용하여 악성코드의 API 호출을 관찰하였다. 이를 위해 IAT(Import Address Table)에 기재되어 있는 API들의 주소를 출원인이 작성한 훅 DLL(Dynamic Link Library) 내의 주소로 덮어 쓴다. 이를 통해 악성코드에서 해당 API를 호출할 때, 이렇게 변경된 IAT는 악성코드의 제어권을 훅 DLL로 넘기게 된다. 훅 DLL은 호출된 API의 정보와 호출 시간, 관련 라이브러리들의 정보를 기록하고, 실제 시스템 API를 호출함으로써 제어권을 다시 악성코드로 전달한다.

[0041] 상술한 호출 그래프란 방향성이 존재하는 그래프로, 이는 분석하고자 하는 악성코드의 특징을 표현하는 중요한 기준이 된다. 따라서, 관찰된 API들의 호출 정보를 이용하여 호출 그래프 G를 생성하였다.

[0042] 호출 그래프 G는 하기 수학식 1과 같다.

수학식 1

[0043] $$G = (V, E)$$

[0044] 여기서, V는 그래프를 구성하는 노드들의 집합이다. 즉, 호출 그래프 G에서 V는 호출된 API 함수들의 집합을 의미한다.

[0045] E는 API들의 호출 순서를 의미하며, 선행 API vi와 호출되는 API vj의 호출 순서 관계를 나타낸다. E는 하기 수학식 2와 같이 표현된다.

수학식 2

[0046] $$E = \{(v_i, v_j) \mid v_i, v_j \in V\}$$

[0047] 도 2에서는 Win32.Worm.Allaple.Gen의 호출 그래프로 약 90개의 노드가 존재한다.

[0048] 이어서, 상술한 호출 그래프를 함수들의 목적에 따라 추상화하여 행위 그래프로 일반화한다(S140).

[0049] 이를 위해 함수들을 사전에 분리된 행위 노드로 추상화한다. 이어서, 같은 그룹의 노드를 하나로 합치는 작업을 수행한다. 이후, 각 노드들을 도 3에서와 같이 행위 그래프로 표현한다.

[0050] 이에 관해 상세히 설명하면 다음과 같다.

[0051] 상술한 함수호출 정보는 정확도 측면에서 매우 높게 평가되지만, 분석에 사용되는 함수들이 수백, 수천에 이르기 때문에 이를 분석하기 위해서는 상당한 노력과 시간이 수반되어야 한다. 또한 같은 행위를 달성하기 위해 작성된 코드라도 목적 시스템, 라이브러리 등에 따라 다양한 함수가 사용될 가능성이 존재한다. 따라서 악성코드마다 다른 노드로 표현된 호출 그래프가 생성되며, 이는 악성코드 간 분석에 일관성을 해치는 요인으로 작용할 수 있다.

[0052] 이에 본 출원인은 다양한 함수들을 그들의 목적에 따라 추상화하여 호출 그래프를 행위 그래프라는 형태로 일반화하였다. 앞서 설명한 바와 같이, 호출 그래프의 노드들은 악성코드에 의해 호출된 API함수들을 의미하고, 연결선은 해당 함수들의 호출 순서를 의미한다.

[0053] 본 실시예에서는 수 많은 API들을 32개의 카테고리로 분류하고, 다시 각각을 4개의 행위로 분류하여, 총 128개의 행위 노드로 추상화하였다.

[0054] 즉, 32개의 카테고리는 MSDN(MicroSoft Developer Network)를 참고하여, 각각 API들의 목적에 따라 프로세스(process), 메모리(memory; registry), 파일(file), 그리고 소켓(socket) 등으로 분류하였으며, 4개의 행위 분류는 오픈(open), 클로우스(close), 리드(read), 그리고 라이트(write)로 정의하였다.

[0055] 따라서 서로 다른 형태의 API들도 최소 128개중 하나의 노드로 표현 가능하게 되었다. 예를 들어 CloseSocket() 함수의 경우, 소켓-클로우스(socket-close) 노드로 표현되며, OpenProcess()의 경우 프로세서-오픈(process-open) 노드로, 그리고 RegSaveKey()의 경우 레지스트리-라이트(registry-write) 노드로 추상화된다.

[0056] 노드 추상화가 완료된 이후에는, 같은 그룹으로 이루어진 노드들을 하나로 합치는 작업이 수행된다. 결과적으로 호출 그래프 추상화를 통해, 악성코드들마다 고유한 행위 특성을 최대 128개의 고정된 개수의 노드로 표현할 수 있고, 이를 도 3에서와 같이 행위 그래프로 표현할 수 있다. 도 3은 Win32.Worm.Allaple.Gen의 행위 그래프로 약 29개의 노드가 존재한다.

[0057] 이어서, 추상화를 통한 행위 그래프 즉, 행위 특징을 분석하기 위해 부분 그래프를 추출한다(S150). 즉, n개의 노드를 가진 악성코드 M에 대한 행위 그래프에서 노드를 차례로 줄여가며 가능한 모든 부분 그래프를 추출하였다.

[0058] 이는, 과거 악성코드들은 그 행위 목적을 실현하기 위해 하나의 바이너리 형태로 구현되었다. 하지만 점차 지능화된 악성코드들은, 한 가지 목적만을 수행하도록 구현된 모듈단위로 이루어져 있으며, 이러한 모듈들이 하나의 그룹 형태로 상호 협력하여 악성행위를 이루도록 점차 변해가고 있다. 이러한 형태의 악성코드 모듈 집합을 악성코드 패밀리(Malware Family)라고 부른다. 한 예로, 최근 이슈가 되고있는 쿱페이스(Koobface) 봇의 경우, 본체에 해당하는 로더(Loader) 외에 블랙리스트 체크를 위한 지채크(GCheck), 가짜 구글(Google) 계정 생성을 수행하는 블로그스팟(Blogspot), 피싱 사이트 수행을 위한 웹 서버(Web server), 그리고 캡차 브레이커(Captcha breaker) 모듈 등, 다 수의 모듈단위로 구현되어 동작한다. 또한 이렇게 모듈화된 악성코드들은 악성코드 개발자들 사이에서 공유되어, 새로운 형태의 악성코드를 보다 쉽게 제작할 수 있도록 돕기 때문에 간과할 수 없는 문제이다.

[0059] 따라서, 악성코드의 모듈별 행위 특징을 분석하기 위해 부분 그래프 추출을 수행하였다. n개의 노드를 가진 악성코드 M에 대한 행위 그래프 G(Mn)에서 노드를 차례로 줄여가며 가능한 모든 부분 그래프 SG(Mn)i를 추출하였다.

[0060] 이때 n은 최소값은 임의로 3으로 정의하였으며, 이는 최소 3단계의 함수 호출이 의미론적 행위에 해당한다. 이는 경험적 논리에 바탕을 두었다. 따라서 n개의 노드를 가진 악성코드 M에 대한 최대 부분그래프의 개수는 하기 수학식 3과 같다.

수학식 3

[0061] $$|SG(M_n)_i| = \sum_{k=1}^{n-2}(k)$$

[0062] 추출된 부분 그래프들은 악성코드가 내포하고 있는 모든 모듈별 행위 특징을 포괄할 수 있다. 따라서 부분 그래프 분석을 이용할 경우, 코드 난독화를 이용한 변종 악성코드들 뿐만 아니라, 동일 모듈을 공유하고 있는 이종 악성코드들에 대한 분석, 더 나아가 악성 행위 목적에 따른 공통된 행위 모델 정의에도 용이하다.

[0063] 상술한 바와 같은 프로세스를 통해 악성 코드 행위에 대한 부분 그래프를 확인할 수 있다. 이후, 도 1에서와 같이 부분 그래프들 간의 비교 분석을 통해 해당 악성 코드들의 유사 유무를 판단한다.

[0064] 즉, 행위 그래프들 간의 유사도 분석을 통해 악성 코드 간의 유사도를 판별한다(S200).

[0065] 즉, 행위 그래프 유사도 분석은 두 악성코드의 행위가 얼마나 유사한지를 판단하기 위한 것으로, 코드 난독화 등을 이용한 변종 악성코드 간의 행위 유사도를 판별하는 중요한 척도이다. 더불어 앞서 추출한 부분 그래프를 이용하여 유사도 분석을 행할 경우, 이종 악성코드 간 공유되는 모듈 행위나 공통 행위 특징을 찾는 것이 가능하다. 물론 이종 그래프 간의 동일성을 판단하는 문제(Graph Isomorphism)는 비결정 완전(NP-complete) 복잡도를 가진 것으로 잘 알려져 있다. 하지만, 본 실시예에서 분석하고자 하는 행위 그래프는 최대 노드 수 128개라는 일관성있는 그래프로서, 일대일 대응을 통한 분석이 정해진 시간 내에 가능하다.

[0066] 이때, 앞서와 같이 악성 코드 간의 유사도를 판별하여 변종 악성코드 M, M'에서 추출한 모든 부분 그래프 SG(M)i 와 SG(M')j에 대해 일치(또는 완전 일치)하는 부분 그래프 그룹 SG(M, M')k를 추출하였다. 여기서, 일치는 80% 이상의 일치를 지칭한다. 물론 완전 일치로 100%의 일치를 추출하는 것이 효과적이다. 이와 같이 추출한 일치 부분 그래프 그룹은 분석 대상이 되는 악성코드 M'을 기준으로 보았을 때, 전체 그래프 G(M')에서 일치 부분 그래프의 비율로 계산된다. 수학식 4는 이러한 유사도 지수 Sim를 수식화한 것이다.

수학식 4

[0067] $$Sim(M, M') = \frac{\sqrt{\sum [SG(M, M')_k]^2}}{n(M')}$$

[0068] 상술한 바와 같은 알고리즘을 통해 변종 악성 코드를 오탐없이 탐지할 수 있고, 서로 다른 악성 코드들 간에 공유되는 행위 모델 또한 분석할 수 있다.

[0069] 하기에서는 상술한 부분 그래프 기반의 행위 분석을 통한 변종 악성 코드 탐지 방법의 실험예에 관해 설명한다.

[0070] 상술한 바와 같이 제안된 알고리즘의 성능 평가를 위해 몇 가지 실험을 통한 검증을 수행하였다.

[0071] 첫째로 악성코드의 고유한 의미론적 특징 모델을 추출하였고, 두번째로 코드 난독화를 실행한 악성코드들에 대한 성능 평가와 변종 구분에 대한 성능 평가하였고, 그리고 마지막으로 악성코드들 간에 공유되고 있는 모듈 행위를 분석하여 향 후, 새로운 악성코드 탐지를 위한 시그너쳐 활용 가능성을 분석하였다.

[0072] 이를 위해 실험 데이터는 트로잔, 웜, 바이러스, 봇을 포함한 101개(변종 악성코드 15개, 고유 악성코드 86개)의 실제 악성코드와 코드 난독화를 적용시킨 172개(코드 난독화 2종 * 고유 악성코드 86개)의 악성코드를 포함하여, 총273개의 악성코드가 이용되었다.

[0073] 실험에 사용된 실제 악성코드는 Offensive-Computing, VX Heavens, 그리고 VX Chaos과 같은 악성코드 제공 사이트를 통해 수집하였다. 실험은 Intel Core2Duo 2.66Ghz CPU와 4GB 주메모리를 탑제한 PC에서 수행되었으며, 어떠한 보안 업데이트도 하지 않은 Windows 운영체제를 기반으로 한 가상머신을 이용하였다.

[0074] 첫 실험에서는, 제안된 알고리즘을 통해 획득한 의미론적 행위 그래프가 얼마나 고유한 분포 형태를 보이는지 평가하였다.

[0075] 이는 신, 변종 악성코드 분류의 기준이 되는 의미론적 행위 그래프가 각 악성코드들의 다양한 특징을 고유한 행위 시그너쳐 형태로 포괄하고 있어야 하기 때문이다. 만약 의미론적 행위 그래프가 악성코드의 행위 특징을 잘 포함하지 못한다면, 이는 제안된 알고리즘의 성능 하향에 지대한 영향을 보일 것이다.

[0076] 도 4는 일 실시예에 따른 악성 코드의 의미론적 행위 그래프 분포도이다.

[0077] 평가를 위해 수집한 101개의 고유한 악성코드로부터 각각 의미론적 행위 그래프를 추출하였다. 모든 그래프에 대해 2개씩 짝을 지어 교차 유사도 분석을 수행하였다. 총 10100개의 분석 결과를 추출하였다.

[0078] 도 4는 10100개의 유사도 분석 결과를 정렬한 결과이다. 결과 중 오직 3%인 342개가 유사도 1을 보였으며, 대다수는 19개의 변종 악성코드에 의해 발생한 것으로 분석되었다. 또한 0.8이상의 높은 유사도를 보인 경우는, 총 45개로 0.4%정도를 차지하였다. 따라서 변종에 의한 높은 유사도 부분을 재외하면, 악성코드들 간에 고유한 행위 그래프 분포를 보이는 것으로 해석할 수 있다.

[0079] 앞선 실험을 통해 각 악성코드들의 고유한 의미론적 행위 그래프를 추출하였다. 하지만 만일 이러한 의미론적 행위 그래프가 코드 난독화를 이용한 변종 악성코드들에서 서로 다른 형태를 보인다면, 기존의 시그너쳐 기반의 악성코드 분석과 전혀 다를 바 없다. 따라서 난독화된 악성코드 간에도 동일한 의미론적 행위 그래프를 보이는지 검증할 필요성이 있다. 이를 위해, 우리는 실제 악성코드와 코드 난독화를 실현한 변종 악성코드간의 비교 검증을 수행하였다. 실험은 수집된 실제 악성코드 중 고유한 형태를 가진 86개에, 인터넷에서 쉽게 구할 수 있는 코드 난독화 기술 두 가지를 접목시켜, 총 258개의 바이너리를 이용하여 실행하였다. 또한 비교 검증을 위해 3가지 상용화된 안티바이러스 제품과 동일한 바이너리에 대해 검증을 실행하였다. 위 실험의 결과는 표 1. 과 같다.

표 1

[0080]

Scanner	Original Malwares	Obfuscated Malwares 1	Obfuscated Malwares 2
Our Mechanism	-	100%	100%
'A' Scanner	98%	78%	74%
'E' Scanner	62%	5%	55%
'K' Scanner	99%	11%	96%

[0081] 위 실험결과에서 볼 수 있듯이 일반적인 시그너쳐 기반의 안티바이러스 제품은 난독화된 악성코드에 대해 낮은 인식률을 기록하였다. 반면 본 실시예에서 제안한 알고리즘의 경우, 난독화된 악성코드들에 대해 100%의 인식률을 보였다. 위 결과는, 비록 난독화된 악성코드라도 의미론적 행위 특징은 그대로 유지된다는 것을 의미한다.

[0082] 현재 새로운 악성코드들의 약 50%가 기존의 악성코드를 변조해 재활용하고 있으며, 이러한 현상은 더욱 가속화될 것으로 예상된다. 따라서 악성코드의 시그너쳐 증가는 매우 중요한 문제이다. 제안된 분석 기법은 이러한 문제를 해결하는데 매우 효과적이다.

[0083] 하나의 의미론적 행위 그래프는 다 수의 변종 악성코드에 인식에 이용할 수 있어, 시그너쳐의 수를 대폭 줄일 수 있으며, 새로운 변종 악성코드가 발견되었을 때 빠른 대응이 가능할 것으로 예상된다.

[0084] 상술한 첫 번째 실험에서 우리는 실제 악성코드 101개에 대한 의미론적 행위 그래프를 추출하고 이를 교차분석하였다. 이 중에는 변종 악성코드 19개(원본 4개, 변종 15개)에 대한 유사도 분석 결과가 포함되어 있으며, 결과는 표 2와 같다.

표 2

[0085]

Metamorphic Malware	Variants	Sim
Win32.Worm.Allaple.Gen	5	1.0
Win32.Worm.Vb.NVA	9	1.0
Trojan.Downloader.Win32.Multdl	2	1.0
Trojan.Downloader.Win32.Delf	3	1.0

[0086] 상기 표 2에서 유사도 지수 Sim은 두 개의 행위 그래프가 얼마나 유사한지를 나타내는 지표로서, 0부터 1까지의 수치로 표현되며, 완전히 일치하는 경우 1을 나타낸다. 실험 결과에서 알 수 있듯이, 변종 악성코드 간의 의미론적 행위 특징 역시 높은 유사도를 보였다.

[0087] 도 5는 일실시예에 따른 Trojan.Downloader.Win32.Multdl 변종들의 의미론적 행위 그래프이다.

[0088] 도 5에서와 같이 매우 복잡한 행위 그래프 형태를 보이는 악성코드들도 고유한 의미론적 행위 특징은 변종간 공유하는 것을 알 수 있다. 따라서 앞선 실험 결과를 비추어 볼 때, 의미론적 행위 그래프는 변종 악성코드를 분류하는 하나의 행위 시그너쳐로 작용할 수 있다는 것을 다시 한 번 증명하였다.

[0089] 최근의 악성코드는 탐지를 회피하기 위해 모듈단위로 구현되는 경우가 많다. 이러한 모듈들은 악성코드 제작자들 사이에서 공유되며, 새로운 악성코드를 개발하는데 활용되고 있다. 부분 그래프를 이용한 의미론적 행위 그래프 분석은 두 그래프의 유사도를 측정하는 중요한 척도일 뿐만 아니라, 서로 다른 악성코드에서 공유되는, 혹은 공통적으로 나타나는 행위 특징을 분석하는데 도움을 준다. 하기에서는 이러한 분석과정에서 발견된 공통 행위 특성에 대해 분석한다.

[0090] 도 6은 일 실시예에 따른 웜에서 나타나는 공통 행위 부분 그래프이다.

[0091] 도 6은 앞서 정의한 128개의 행위 노드 중, console-read, file-read, system_information-read, service-read로 구성된 부분 그래프이다.

[0092] 위 행위 노드들은 모두 감염 시스템의 정보 수집을 주로 수행할 때 발견되는 노드들로, 각 노드를 오가며 지속적인 정보 수집 행위가 이어진 것으로 분석된다.

[0093] 부분 그래프는 실험에 사용된 웜 45개 중, 82%인 37개에서 모두 발견되었다. 해당 부분 그래프의 악성 여부를 검증하기 위해, 우리는 Windows 시스템상의 정상 프로그램 20개와 비교분석 하였으며, 모든 정상 프로그램에서 위 부분 그래프와 동일한 행위 패턴은 발견되지 않았다.

[0094] 상술한 부분 그래프를 통해 악성 코드를 분류하는데 중요한 척도로 작용할 행위 모델을 추출할 수 있다.

[0095] 상술한 바와 같이 본 실시예는 신, 변종 악성코드를 효과적으로 분석하고 분류할 수 있는 새로운 알고리즘을 제안하였다.

[0096] 그리고, 악성코드의 API 호출 순서와 행위 추상화를 이용하여, 악성코드의 고유한 행위 모델을 추출하여 코드 난독화를 이용한 변종 악성코드의 효율적인 분석 및 분류를 할 수 있다. 악성코드의 폭발적인 증가에 따른 시그너쳐의 증가의 문제점 역시, 고유한 행위 시그너쳐 생성을 이용하여 해결할 수 있다. 또한 부분 행위 특징 분석을 이용하여, 악성코드 간 공유되는 부분 행위를 추적하고, 새로운 악성코드 분류 기준을 제시할 수 있다.

[0097] 하기에서는 상술한 악성 코드 탐지 방법을 구현하기 위한 시스템에 관해 설명한다. 물론 본 실시예의 악성 코드 탐지 방법을 구현하기 위한 시스템은 후술되는 시스템 이외에 그 동작과 기능을 하는 모듈의 통합 또는 세분화를 통해 다양한 구성으로 제작될 수 있다. 후술되는 설명은 그 일 예를 설명하는 것으로 본 발명은 이에 한정되지 않고, 다양한 변형이 가능하다.

[0098] 도 7은 본 발명의 일 실시예에 따른 악성 코드 탐지를 위한 시스템의 개념도이다.

[0099] 도 7을 참조하면, 본 실시예에 따른 악성 코드 탐지 시스템은 먼저, 악성 코드의 동적 정보 분석을 통해 행위 부분 집합을 생성하는 행위 감지 모듈(100)과, 상기 행위 감지 모듈(100)에 의한 악성 코드의 행위 유사성을 판단하는 유사도 판별 모듈(200)을 구비한다.

[0100] 행위 감지 모듈(100)은 악성 코드의 함수 호출 정보를 추출하기 위한 호출 정보 추출 모듈(110)과, 상기 추출된 함수 호출 정보를 분석하여 행위 노드, 호출 순서와 호출 함수 집합을 갖는 호출 그래프를 생성하는 호출 그래프 생성 모듈(120)과, 함수들의 목적에 따라 추상화된 행위 노드에 의해 호출 그래프를 행위 그래프로 일반화하는 행위 그래프 생성 모듈(130)과, n개의 노드를 갖는 악성 코드 M에 대하여 행위 그래프에서 노드를 차례로 줄여가며 가능한 모든 부분 그래프를 추출하는 부분 그래프 추출 모듈(140)을 포함한다.

[0101] 본 실시예에서는 상기 함수 호출을 악성 코드의 API를 호출한다. 즉, 호출 정보 추출 모듈(110)은 API 정보를

추출하고, 호출 그래프 생성 모듈(120)은 API들이 호출 정보를 이용하여 호출 그래프를 생성한다. 호출 정보 추출 모듈(110)은 가상 머신을 이용한 동적 분석을 실시한다.

[0102] 또한, 본 실시예에서는 수많은 API 들을 32개의 카테고리로 분류하고, 다시 이들을 각기 4개의 행위로 분류하였다. 따라서, 총 128개의 행위 노드로 추상화하였다. 이를 통해 호출 그래프를 행위 그래프로 일반화할 수 있게 되었다.

[0103] 더욱이, 이와 같이 일반화된 행위 그래프에서 부분 그래프를 추출함으로 인해 악성 코드가 내포하는 모듈별 행위 특징을 포괄되고 공통된 행위 모델로 정의할 수 있게 된다.

[0104] 이어서, 유사도 판별 모듈(200)은 행위 그래프 유사도를 분석한다. 이는 서로 다른 악성 코드들에서 추출한 부분 그래프에 대한 상호 비교를 통해 일치하는 부분 그래프 그룹을 추출한다. 이를 통해 악성 코드가 코드 난독화 및 실행압축 기술에 의해 변종 악성 코드화되더라도 이를 빠르게 탐지할 수 있고, 이에 대하여 빠르게 대응할 수 있게 된다.

[0105] 이상, 본 발명에 대하여 전술한 실시예들 및 첨부된 도면을 참조하여 설명하였으나, 본 발명은 이에 한정되지 않으며 후술되는 특허청구범위에 의해 한정된다. 따라서 본 기술분야의 통상의 지식을 가진 자라면 후술되는 특허청구범위의 기술적 사상을 벗어나지 않는 범위 내에서 본 발명이 다양하게 변형 및 수정될 수 있음을 알 수 있을 것이다.

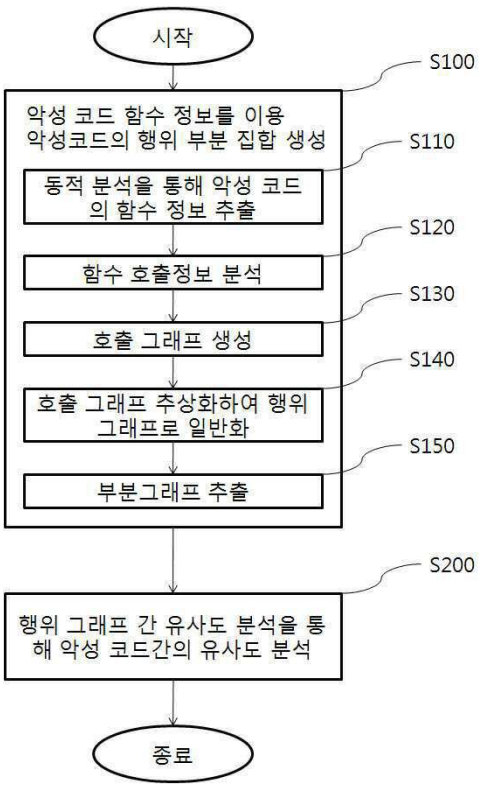
부호의 설명

[0106]

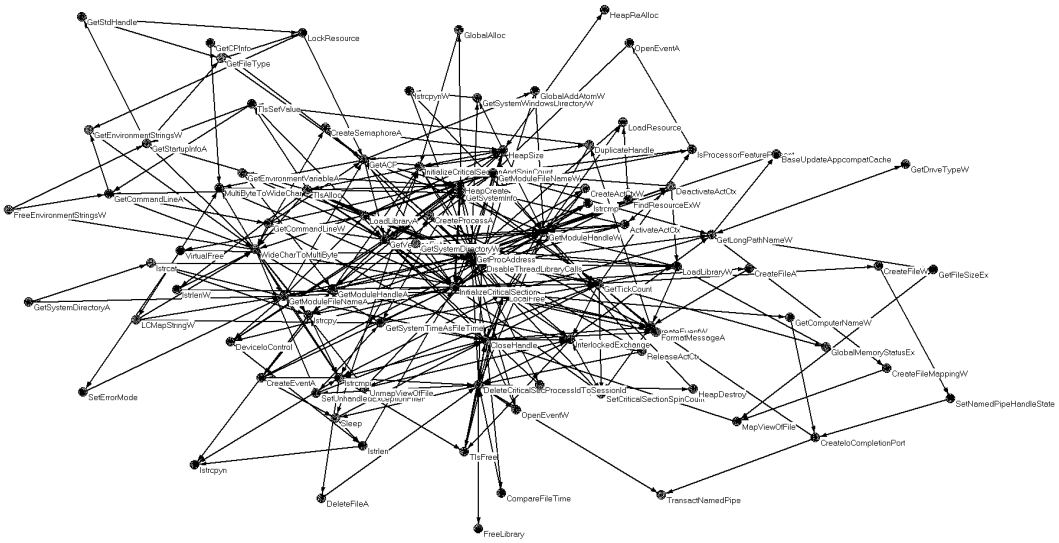
100 : 행위 감지 모듈	110 : 호출 정보 추출 모듈
120 : 호출 그래프 생성 모듈	130 : 행위 그래프 생성 모듈
140 : 부분 그래프 추출 모듈	200 : 유사도 판별 모듈

도면

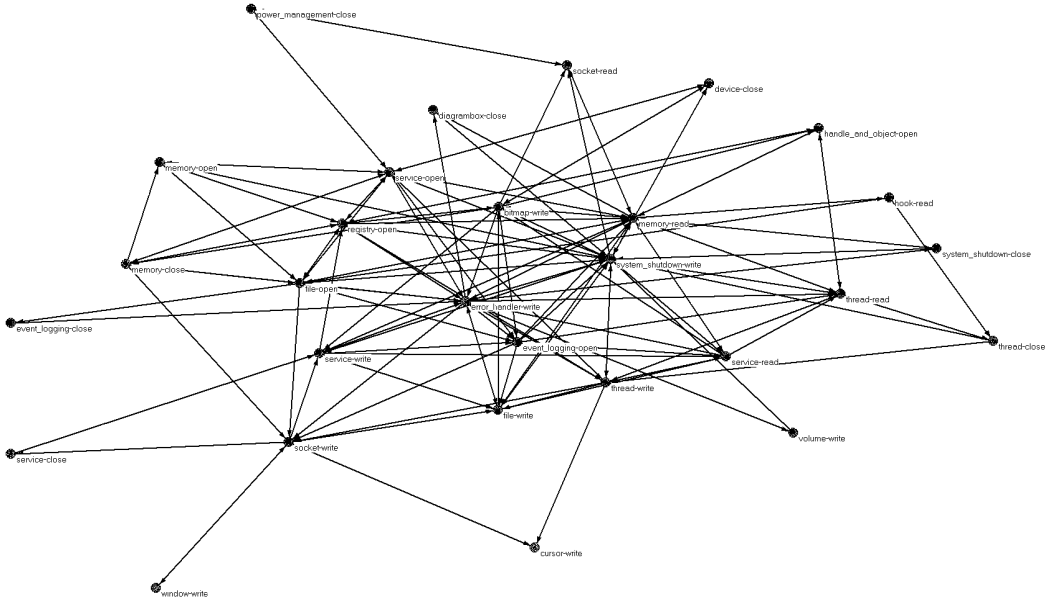
도면1



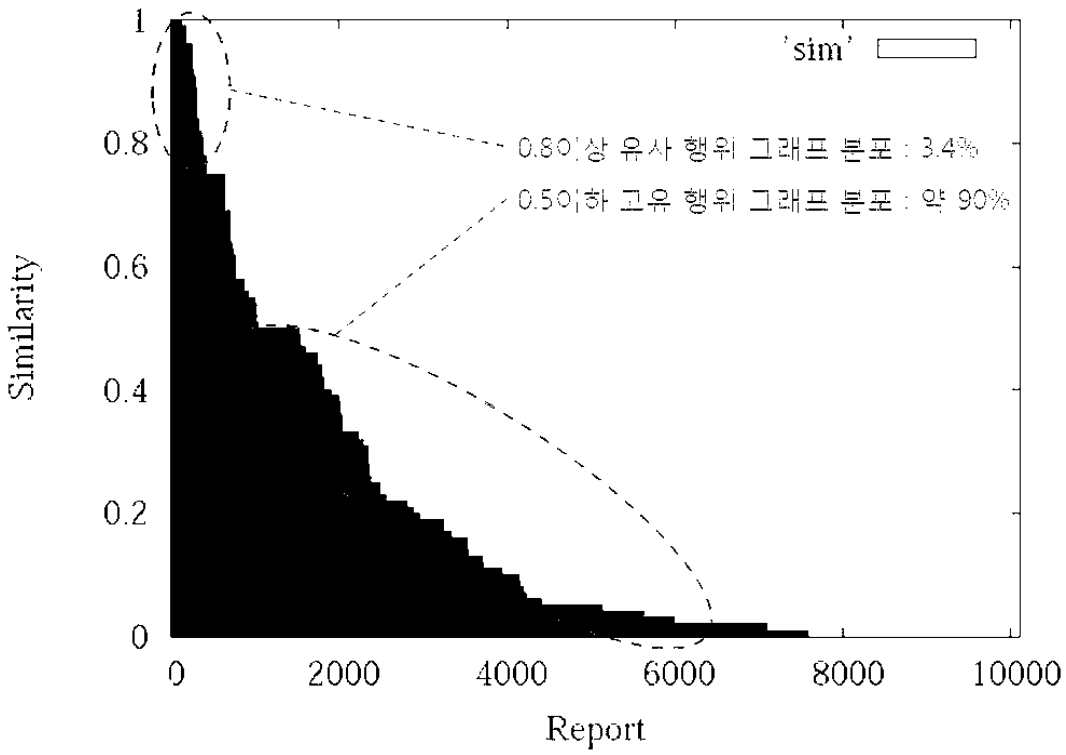
도면2


GetStdHandle
LockResource
GetCPInfo
GetFileType
HeapFileAlloc
GlobalAlloc
OpenEventA
lstrcpynW
GetSystemWindowsDirectoryW
GlobalAddAtomW
LoadResource
GetEnvironmentStringsW
GetStartupInfoA
CreateSemaphoreA
HeapSize
DuplicateHandle
IsProcessorFeaturePresent
BaseUpdateAppcompatCache
GetDriveTypeW
GetCommandLineA
FreeEnvironmentStringsW
GetEnvironmentVariableA
HeapCreate
GetSystemInfo
CreateActCtxW
DeactivateActCtx
FindResourceExW
GetCommandLineW
ActivateActCtx
GetLongPathNameW
VirtualFree
GetProcAddress
GetModuleHandleW
lstrcat
DisableThreadLibraryCalls
LoadLibraryW
CreateFileA
CreateFileW
GetFileSizeEx
GetSystemDirectoryA
GetModuleHandleA
GetTickCount
LCMapStringW
lstrcpy
GetSystemTimeAsFileTime
GetComputerNameW
DeviceIoControl
CloseHandle
InterlockedExchange
FormatMessageA
GlobalMemoryStatusEx
CreateEventA
lstrcmpi
UnmapViewOfFile
DeleteCriticalSection
ProcessIdToSessionId
HeapDestroy
CreateFileMappingW
SetErrorMode
Sleep
OpenEventW
MapViewOfFile
SetNamedPipeHandleState
lstrlen
TlsFree
CreateIoCompletionPort
lstrcpyn
DeleteFileA
CompareFileTime
TransactNamedPipe
FreeLibrary

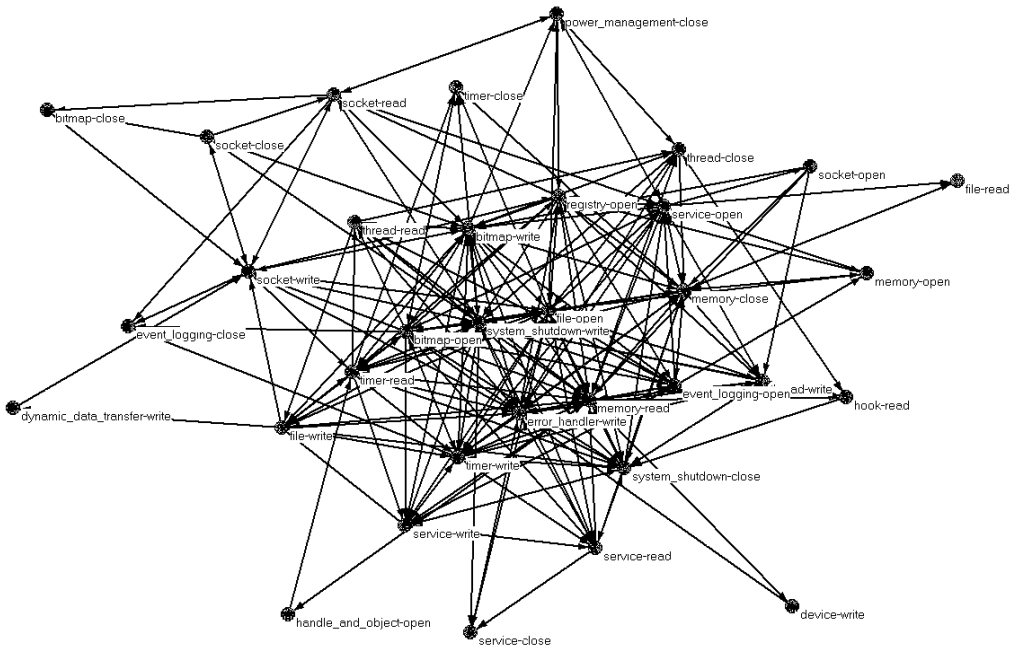
도면3


power_management-close
socket-read
device-close
diagrambox-close
handle_and_object-open
memory-open
service-open
hook-read
mmap-write
memory-read
registry-open
system_shutdown-close
memory-close
system_shutdown-write
file-open
thread-read
error_handler-write
event_logging-close
thread-close
event_logging-open
service-write
service-read
thread-write
file-write
volume-write
service-close
socket-write
cursor-write
window-write

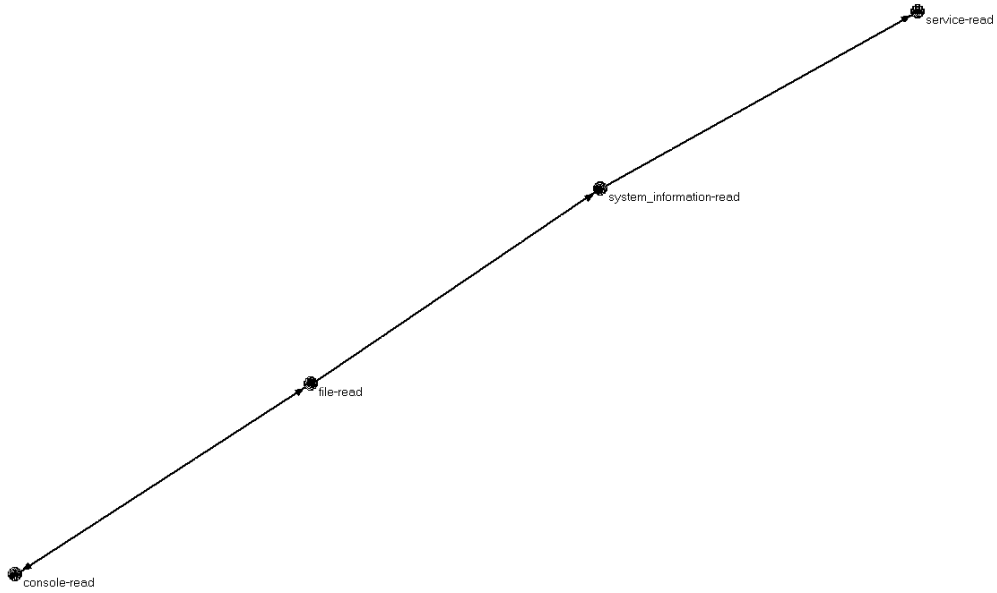
도면4


'sim'
0.80이상 유사 행위 그래프 분포 : 3.4%
0.50이하 고유 행위 그래프 분포 : 약 90%
Similarity
Report
1
0.8
0.6
0.4
0.2
0
0
2000
4000
6000
8000
10000

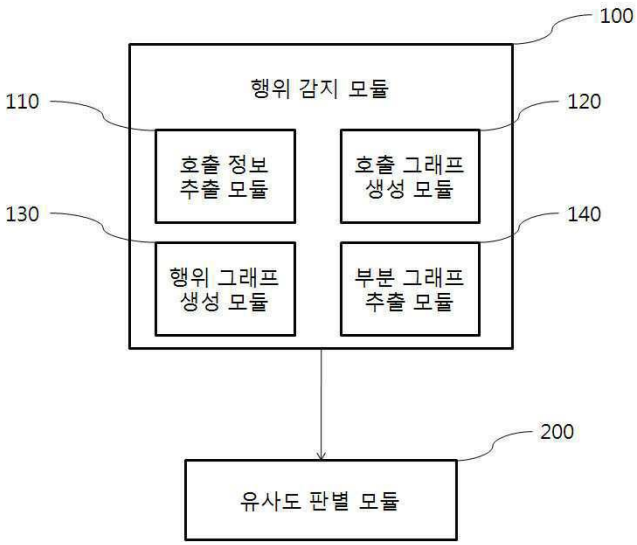
도면5


power_management-close
socket-read
timer-close
bitmap-close
socket-close
thread-close
socket-open
file-read
registry-open
service-open
thread-read
bitmap-write
socket-write
memory-close
memory-open
file-open
system_shutdown-write
event_logging-close
bitmap-open
timer-read
event_logging-open
ad-write
hook-read
dynamic_data_transfer-write
memory-read
error_handler-write
file-write
timer-write
system_shutdown-close
service-write
service-read
device-write
handle_and_object-open
service-close

도면6


service-read
system_information-read
file-read
console-read

도면7


100
행위 감지 모듈
110
120
호출 정보 추출 모듈
호출 그래프 생성 모듈
130
140
행위 그래프 생성 모듈
부분 그래프 추출 모듈
200
유사도 판별 모듈