(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2023-0103275
(43) 공개일자 2023년07월07일

(51) 국제특허분류(Int. Cl.)
G06F 21/56 (2013.01) *G06F 40/40* (2020.01)
(52) CPC특허분류
G06F 21/563 (2013.01)
G06F 40/40 (2020.01)
(21) 출원번호 10-2021-0194045
(22) 출원일자 2021년12월31일
심사청구일자 없음

(71) 출원인
주식회사 샌즈랩
서울특별시 강남구 선릉로 577, 4층(역삼동, 조선내화빌딩)
(72) 발명자
김기홍
서울특별시 영등포구 시흥대로 595, 101동 1002호(대림동, H HOUSE 대림 뉴스테이)
(74) 대리인
지관영

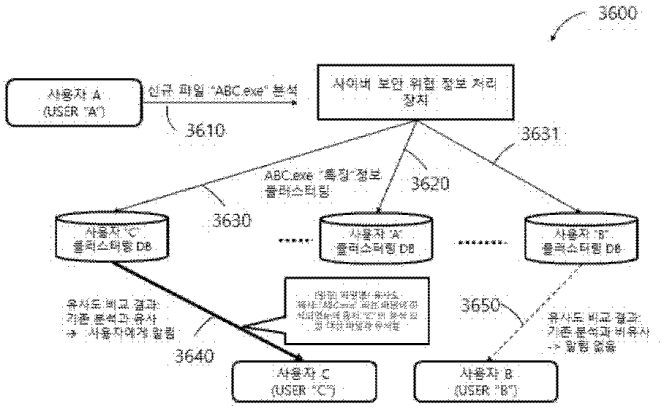
전체 청구항 수 : 총 15 항

(54) 발명의 명칭 **사이버 보안 위협 정보 처리 장치, 사이버 보안 위협 정보 처리 방법 및 사이버 보안 위협 정보 처리하는 프로그램을 저장하는 저장매체**

(57) 요 약

개시하는 실시 예는 사이버 보안 위협 정보 처리 장치, 사이버 보안 위협 정보 처리 방법 및 사이버 보안 위협 정보 처리하는 프로그램을 저장하는 저장매체를 제공한다.

대 표 도 - 도36



이 발명을 지원한 국가연구개발사업

과제고유번호	1711134514
과제번호	2021-0-00624-001
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기획평가원
연구사업명	정보보호핵심원천기술개발(R&D,정보화)
연구과제명	C-ITS 인프라 보안성 강화를 위한 복합 위협헌팅 모델 기반 지능형 사이버 공격/방어 분석 프레임워크 기술 개발
기 여 율	1/1
과제수행기관명	주식회사 샌즈랩
연구기간	2021.04.01 ~ 2021.12.31

명 세 서

청구범위

청구항 1

실행 파일 분석을 위한 제 1 사용자 실행 파일 분석 요청 신호를 입력 받는 단계;

상기 제 1 사용자 실행 파일 분석 요청 신호에 따라 상기 실행 파일을 디스어셈블링(disassembling)하여 디스어셈블된 코드를 얻고 상기 디스어셈블된 코드를 재구성하여 재구성된 디스어셈블드 코드를 얻는 단계;

상기 재구성된 디스어셈블드 코드를 처리하여 해쉬 함수로 변환하고 상기 해쉬 함수를 N 그램(N-gram, N은 자연수) 데이터로 변환하는 단계;

상기 변환된 N 그램(N-gram) 데이터의 블록 단위의 코드에 대해 앙상블 머신 러닝을 수행하여 상기 블록 단위의 코드를 상기 블록 단위의 코드가 수행하는 공격 기법의 식별자 및 상기 블록 단위의 코드를 생성한 공격자의 식별자로 프로파일링하여 실행 파일 분석 결과를 생성하는 단계;

상기 실행 파일 분석 결과를 기반으로 제 1 사용자 클러스터링 데이터를 생성하고, 상기 제 1 사용자와 다른 제 2 사용자의 클러스터링 데이터와 상기 실행 파일 분석 결과간의 유사도를 비교하는 단계; 및

상기 유사도가 기설정된 값보다 높은 경우, 상기 제 2 사용자에게 상기 유사도와 관련된 정보를 제공하는 단계를 포함하는 사이버 보안 위협 정보 처리 방법.

청구항 2

제 1항에 있어서,

상기 유사도와 관련된 정보는 상기 파일 분석 결과 및 상기 유사도를 포함하는, 사이버 보안 위협 정보 처리 방법.

청구항 3

제 1항에 있어서,

상기 제 1 사용자 클러스터링 데이터는 상기 실행 파일 분석 결과에 상기 제 1 사용자를 나타내는 정보를 부여하는 라벨링을 수행하여 생성되는, 사이버 보안 위협 정보 처리 방법.

청구항 4

제 1항에 있어서,

상기 디스어셈블된 코드는, 상기 실행 파일에 포함된 함수에 대응하는 OP-CODE와 상기 함수의 피연산자인 어셈블리 코드를 포함하는 사이버 보안 위협 정보 처리 방법.

청구항 5

제 1항에 있어서

상기 프로파일링하는 단계는,

상기 블록 단위의 코드와 저장된 악성 코드의 유사 패턴을 찾는 단계; 및

상기 유사 패턴인 블록 단위의 코드에 대해 적어도 하나의 노드를 가지는 디시전 트리를 이용하여 상기 공격 기법의 식별자와 공격자의 식별자를 분류하는 단계;를 포함하는 사이버 보안 위협 정보 처리 방법.

청구항 6

제 1항에 있어서,

상기 해쉬 함수를 N 그램(N-gram) 데이터로 변환하는 단계는,

상기 해쉬 함수를 바이트 데이터로 변환하는 단계; 및

상기 바이트 데이터를 2-gram 데이터로 변환하는 단계;를 포함하는 사이버 보안 위협 정보 처리 방법.

청구항 7

제 1항에 있어서,

상기 변환된 N 그램(N-gram) 데이터의 블록 단위의 코드에 대해 상기 블록 단위의 코드가 사이버 공격 행위가 포함된 코드인지 판단할 경우,

상기 블록 단위의 코드를 자연어 처리 방식에 따라 저장된 악성 코드와 유사도를 판단하는 단계;를 포함하는 사이버 보안 위협 정보 처리 방법.

청구항 8

사용자 별 클러스터링 데이터를 저장하는 하나 또는 그 이상의 사용자 별 데이터 베이스들;

실행 파일 분석을 위한 제 1 사용자 실행 파일 분석 요청 신호를 입력받는 입력부; 및

상기 실행 파일을 처리하는 프로세서를 포함하는 사이버 보안 위협 정보 처리 장치로서,

상기 프로세서는 상기 제 1 사용자 실행 파일 분석 요청 신호에 따라 응용 프로그램 인터페이스(Application Programming Interface; API)를 통해 상기 실행 파일을 디스어셈블링(disassmebling)하여 디스어셈블된 코드를 얻고 상기 디스어셈블된 코드를 재구성하여 재구성된 디스어셈블드 코드를 얻는 디스어셈블링 모듈을 수행하고,

상기 재구성된 디스어셈블드 코드를 처리하여 해쉬 함수로 변환하고 상기 해쉬 함수를 N 그램(N-gram,N은 자연수) 데이터로 변환하는 데이터 변환 모듈을 수행하고,

상기 변환된 N 그램(N-gram) 데이터의 블록 단위의 코드에 대해 앙상블 머신 러닝을 수행하여 상기 블록 단위의 코드를 상기 블록 단위의 코드가 수행하는 공격 기법의 식별자 및 상기 블록 단위의 코드를 생성한 공격자의 식별자로 프로파일링하여 실행 파일 분석 결과를 생성하는 프로파일링 모듈을 수행하고,

상기 실행 파일 분석 결과를 기반으로 제 1 사용자 클러스터링 데이터를 생성하고, 상기 제 1 사용자와 다른 제 2 사용자의 클러스터링 데이터와 상기 실행 파일 분석 결과 간의 유사도를 비교하는 유사도 비교 모듈을 수행하고,

상기 유사도가 기설정된 값보다 높은 경우, 상기 제 2 사용자에게 상기 유사도와 관련된 정보를 제공하는, 사이버 보안 위협 정보 처리 장치.

청구항 9

제 8항에 있어서,

상기 유사도와 관련된 정보는 상기 파일 분석 결과 및 상기 유사도를 포함하는, 사이버 보안 위협 정보 처리 장치.

청구항 10

제 8항에 있어서,

상기 제 1 사용자 클러스터링 데이터는 상기 실행 파일 분석 결과에 상기 제 1 사용자를 나타내는 정보를 부여하는 라벨링을 수행하여 생성되는, 사이버 보안 위협 정보 처리 장치.

청구항 11

제 8항에 있어서,

상기 디스어셈블된 코드는, 상기 실행 파일에 포함된 함수에 대응하는 OP-CODE와 상기 함수의 피연산자인 어셈블리 코드를 포함하는 사이버 보안 위협 정보 처리 장치.

청구항 12

제 8항에 있어서,

상기 프로파일링 모듈은,

상기 블록 단위의 코드와 저장된 악성 코드의 유사 패턴을 찾고,

상기 유사 패턴인 블록 단위의 코드에 대해 적어도 하나의 노드를 가지는 디시전 트리를 이용하여 상기 공격 기법의 식별자와 공격자의 식별자를 분류하는 사이버 보안 위협 정보 처리 장치.

청구항 13

제 8항에 있어서,

상기 데이터 변환 모듈은;

상기 해쉬 함수를 바이트 데이터로 변환하고 상기 바이트 데이터를 2-gram 데이터로 변환하는 사이버 보안 위협 정보 처리 장치.

청구항 14

제 8항에 있어서,

상기 프로파일링 모듈은, 상기 변환된 N 그램(N-gram) 데이터의 블록 단위의 코드에 대해 상기 블록 단위의 코드가 사이버 공격 행위가 포함된 코드인지 판단할 경우, 상기 블록 단위의 코드의 자연어 처리 방식에 따라 저장된 악성 코드와 유사도를 판단하는 사이버 보안 위협 정보 처리 장치.

청구항 15

실행 파일 분석을 위한 제 1 사용자 실행 파일 분석 요청 신호에 따라 상기 실행 파일을 디스어셈블링(disassembling)하여 디스어셈블된 코드를 얻고 상기 디스어셈블된 코드를 재구성하여 재구성된 디스어셈블드 코드를 얻고;

상기 재구성된 디스어셈블드 코드를 처리하여 해쉬 함수로 변환하고 상기 해쉬 함수를 N 그램(N-gram, N은 자연수) 데이터로 변환하고;

상기 변환된 N 그램(N-gram) 데이터의 블록 단위의 코드에 대해 앙상블 머신 러닝을 수행하여 상기 블록 단위의 코드를 상기 블록 단위의 코드가 수행하는 공격 기법의 식별자 및 상기 블록 단위의 코드를 생성한 공격자의 식별자로 프로파일링하여 실행 파일 분석 결과를 생성하고;

상기 실행 파일 분석 결과를 기반으로 제 1 사용자 클러스터링 데이터를 생성하고, 상기 제 1 사용자와 다른 제 2 사용자의 클러스터링 데이터와 상기 실행 파일 분석 결과간의 유사도를 비교하고; 및

상기 유사도가 기설정된 값보다 높은 경우, 상기 제 2 사용자에게 상기 유사도와 관련된 정보를 제공하는; 사이버 보안 위협 정보를 처리하는 프로그램을 저장하는 저장 매체.

발명의 설명

기 술 분 야

[0001] 개시하는 실시 예들은 사이버 보안 위협 정보 처리 장치, 사이버 보안 위협 정보 처리 방법 및 사이버 보안 위협 정보 처리하는 프로그램을 저장하는 저장매체에 관한 것이다.

배 경 기 술

[0002] 신종 또는 변종 등의 악성코드를 중심으로 점차 고도화 되고 있는 사이버 보안 위협의 피해가 커지고 있다. 이러한 피해를 조금이라도 줄이고 조기에 대응하기 위해서 다차원의 패턴 구성 및 각종 복합 분석 등을 통해서 대응 기술에 대한 고도화를 병행해 나가고 있다. 그러나, 최근의 사이버 공격은 제어 범위 내에 적절하게 대응되기 보다는 오히려 나날이 위협이 증가하고 있는 추세이다. 이러한 사이버 공격은 기존 ICT (Information and Communication Technology) 기반 시설을 넘어서 우리 삶에 직접적으로 영향을 끼치는 금융, 교통, 환경, 건강 등에 까지 위협을 가하고 있다.

[0003] 현존하는 대부분의 사이버 보안 위협을 탐지하고 대응하는 기반 기술 중에 하나는 사이버 공격 또는 악성 코드에 대한 패턴을 데이터베이스를 사전에 생성하고 데이터 흐름이 필요한 곳에 적절한 모니터링 기술을 활용한다. 기존의 기술은 모니터링된 패턴과 일치하는 데이터 흐름 또는 코드가 탐지되면 위협을 식별하여 대응하는 방식을 바탕으로 발전되어 왔다. 이와 같은 종래의 기술은 사전에 확보된 패턴과 일치하면 빠르고 정확하게 탐지할 수 있다는 장점이 있지만, 패턴이 확보되지 않거나 우회하는 신종, 변종 위협의 경우 탐지 자체가 불가능하거나 분석하는데 매우 시간이 오래 소요되는 문제점이 있었다.

[0004] 종래의 기술은 인공지능 분석을 활용하더라도 악성코드 자체를 탐지하고 분석하는 기술을 고도화하는 방법에 초점이 맞춰져 있다. 그러나 근본적으로 사이버 보안 위협을 대응하기 위한 원천적인 기술은 존재하지 않아 이러한 방법만으로 신종 악성코드나 그 악성코드의 변종에 대응하기 힘들며 한계가 있다는 문제점이 있다.

[0005] 예를 들면 이미 발견된 악성 코드 자체를 탐지하고 분석하는 기술만으로는 그 탐지나 분석 시스템을 속이기 위한 디코이(decoy) 정보나 가짜 정보에 대응하지 못하고 혼선이 발생하는 문제점이 있다.

[0006] 학습할 데이터가 충분히 있는 대량 생산의 악성코드의 경우는 그 특징 정보를 충분히 확보할 수 있기 때문에 악성 여부 및 악성코드 종류를 구분할 수 있다. 그러나, 상대적으로 수량이 작게 만들어져 정교하게 공격하는 APT (Advanced Persistent Threat) 공격의 경우는 학습 데이터와 일치하지 않는 경우가 많고 타겟팅(targeting)된 공격이 대다수를 이루고 있기 때문에 기존 기술은 고도화하더라도 한계점이 존재한다.

[0007] 또한 종래에는 악성 코드, 공격 코드 또는 사이버 위협에 대한 설명을 하는 방법과 표현 기법이 분석가의 입장이나 분석 시각에 따라 달랐다. 예를 들면 악성 코드와 공격 행위를 기술하는 방식은 전세계적으로 표준이 되지 않아 같은 사건, 같은 악성코드를 탐지하여도 해당 분야의 전문가의 설명이 달라 혼동이 되는 문제점이 있었다. 심지어 악성코드 탐지 명 또한 통일이 되지 않아 같은 악성 파일임에도 불구하고 어떤 공격이 정확하게 수행되었는지 식별되지 못하거나 다르게 정리되었다. 따라서 식별된 공격 기법을 정규화되고 표준화된 방식으로 설명하지 못하는 문제점이 있었다.

[0008] 종래의 악성 코드 탐지 및 분석 방법은 악성코드 자체의 탐지를 중시하여 매우 유사한 악성 행위를 수행하는 악성 코드의 경우 생성하는 공격자가 다른 경우 공격자들을 식별하지 못하는 문제점이 있었다.

[0009] 위와 같은 문제점들과 연결되어 종래의 방식은 이러한 개별적인 케이스 집중된 탐지 방법에 의해 추후 가까운 미래에 어떤 사이버 위협 공격이 있을지 예측하기 어려운 문제점이 있었다.

발명의 내용

해결하려는 과제

[0010] 이하에서 개시하는 실시 예의 목적은, 인공 지능으로 학습된 데이터와 정확하게 일치하지 않는 악성 코드라도 탐지하고 대응할 수 있고 악성 코드의 변종에 대응할 수 있는 사이버 보안 위협 정보 처리 장치, 사이버 보안 위협 정보 처리 방법 및 사이버 보안 위협 정보 처리하는 프로그램을 저장하는 저장매체를 제공하는 것이다.

[0011] 실시 예의 다른 목적은 악성 코드의 변종이라도 매우 빠른 시간 내에 악성 코드, 공격 기법, 공격자와 공격 예측 방법을 식별할 수 있는 사이버 보안 위협 정보 처리 장치, 사이버 보안 위협 정보 처리 방법 및 사이버 보안 위협 정보 처리하는 프로그램을 저장하는 저장매체를 제공하는 것이다.

[0012] 실시 예의 다른 목적은 악성코드 탐지 명 등이 통일되지 않거나 사이버 공격 기법이 정확하게 기술되지 못하는 악성 코드의 정보를 정규화되고 표준화된 방식으로 제공할 수 있는 사이버 보안 위협 정보 처리 장치, 사이버 보안 위협 정보 처리 방법 및 사이버 보안 위협 정보 처리하는 프로그램을 저장하는 저장매체를 제공하는 것이다.

[0013] 실시 예의 다른 목적은 매우 유사한 악성 행위를 수행하는 악성 코드를 생성하는 다른 공격자들을 식별하고 미래에 어떤 사이버 위협 공격이 있을지 예측이 가능한 사이버 보안 위협 정보 처리 장치, 사이버 보안 위협 정보 처리 방법 및 사이버 보안 위협 정보 처리하는 프로그램을 저장하는 저장매체를 제공하는 것이다.

과제의 해결 수단

[0014] 따라서 실시예들에 따른 사이버 보안 위협 정보 처리 방법은 실행 파일 분석을 위한 제 1 사용자 실행 파일 분석 요청 신호를 입력 받는 단계, 제 1 사용자 실행 파일 분석 요청 신호에 따라 실행 파일을 디스어셈블링(disassembling)하여 디스어셈블된 코드를 얻고 디스어셈블된 코드를 재구성하여 재구성된 디스어셈블드 코드를

얻는 단계, 재구성된 디스어셈블드 코드를 처리하여 해쉬 함수로 변환하고 해쉬 함수를 N 그램(N-gram, N은 자연수) 데이터로 변환하는 단계, 변환된 N 그램(N-gram) 데이터의 블록 단위의 코드에 대해 앙상블 머신 러닝을 수행하여 블록 단위의 코드를 블록 단위의 코드가 수행하는 공격 기법의 식별자 및 블록 단위의 코드를 생성한 공격자의 식별자로 프로파일링하여 실행 파일 분석 결과를 생성하는 단계, 실행 파일 분석 결과를 기반으로 제 1 사용자 클러스터링 데이터를 생성하고, 제 1 사용자와 다른 제 2 사용자의 클러스터링 데이터와 상기 실행 파일 분석 결과간의 유사도를 비교하는 단계 및 유사도가 기설정된 값보다 높은 경우, 제 2 사용자에게 유사도와 관련된 정보를 제공하는 단계를 포함한다.

[0015] 실시예들에 따른 사이버 보안 위협 처리 장치는 사용자 별 클러스터링 데이터를 저장하는 하나 또는 그 이상의 사용자 별 데이터 베이스들, 실행 파일 분석을 위한 제 1 사용자 실행 파일 분석 요청 신호를 입력받는 입력부 및 실행 파일을 처리하는 프로세서를 포함한다. 실시예들에 따른 프로세서는 제 1 사용자 실행 파일 분석 요청 신호에 따라 응용 프로그램 인터페이스(Application Programming Interface; API)를 통해 실행 파일을 디스어셈블링(disassmebling)하여 디스어셈블된 코드를 얻고 디스어셈블된 코드를 재구성하여 재구성된 디스어셈블드 코드를 얻는 디스어셈블링 모듈을 수행하고, 재구성된 디스어셈블드 코드를 처리하여 해쉬 함수로 변환하고 해쉬 함수를 N 그램(N-gram,N은 자연수) 데이터로 변환하는 데이터 변환 모듈을 수행하고, 변환된 N 그램(N-gram) 데이터의 블록 단위의 코드에 대해 앙상블 머신 러닝을 수행하여 블록 단위의 코드를 블록 단위의 코드가 수행하는 공격 기법의 식별자 및 블록 단위의 코드를 생성한 공격자의 식별자로 프로파일링하여 실행 파일 분석 결과를 생성하는 프로파일링 모듈을 수행하고, 실행 파일 분석 결과를 기반으로 제 1 사용자 클러스터링 데이터를 생성하고, 제 1 사용자와 다른 제 2 사용자의 클러스터링 데이터와 실행 파일 분석 결과 간의 유사도를 비교하는 유사도 비교 모듈을 수행하고, 유사도가 기설정된 값보다 높은 경우, 제 2 사용자에게 유사도와 관련된 정보를 제공한다.

[0016] 실시예들에 따른 사이버 보안 위협 정보를 처리하는 프로그램을 저장하는 저장 매체는 사이버 보안 위협 처리 실행 파일 분석을 위한 제 1 사용자 실행 파일 분석 요청 신호에 따라 실행 파일을 디스어셈블링(disassembling)하여 디스어셈블된 코드를 얻고 디스어셈블된 코드를 재구성하여 재구성된 디스어셈블드 코드를 얻고, 재구성된 디스어셈블드 코드를 처리하여 해쉬 함수로 변환하고 해쉬 함수를 N 그램(N-gram, N은 자연수) 데이터로 변환하고, 변환된 N 그램(N-gram) 데이터의 블록 단위의 코드에 대해 앙상블 머신 러닝을 수행하여 블록 단위의 코드를 블록 단위의 코드가 수행하는 공격 기법의 식별자 및 블록 단위의 코드를 생성한 공격자의 식별자로 프로파일링하여 실행 파일 분석 결과를 생성하고, 실행 파일 분석 결과를 기반으로 제 1 사용자 클러스터링 데이터를 생성하고, 제 1 사용자와 다른 제 2 사용자의 클러스터링 데이터와 실행 파일 분석 결과간의 유사도를 비교하고, 및 유사도가 기설정된 값보다 높은 경우, 제 2 사용자에게 유사도와 관련된 정보를 제공한다.

발명의 효과

[0017] 이하에서 개시하는 실시예에 따르면 머신 러닝으로 학습된 데이터와 정확하게 일치하지 않는 악성 코드라도 탐지하고 대응할 수 있고 악성 코드의 변종에 대응할 수 있다.

[0018] 실시예에 따르면 악성 코드의 변종이라도 매우 빠른 시간 내에 악성 코드, 공격 기법 및 공격자를 식별할 수 있고 나아가 추후의 특정 공격자의 공격 기법을 예측할 수 있다.

[0019] 실시예에 따르면 이러한 악성 코드 여부, 공격 기법, 공격 식별자 및 공격자를 기반으로 사이버 공격 구현 방식을 정확히 식별하고 이를 표준화된 모델로 제공할 수 있다. 실시예에 따르면 악성코드 탐지 명 등이 통일되지 않거나 사이버 공격 기법이 정확하게 기술되지 못하는 악성 코드의 정보를 정규화되고 표준화된 방식으로 제공할 수 있다.

[0020] 또한 기존에 알려지지 않은 악성 코드를 생성 가능성과 이를 개발할 수 있는 공격자들을 예측하고 미래에 어떤 사이버 위협 공격이 있을지 예측 가능한 수단을 제공할 수 있다.

도면의 간단한 설명

[0021] 도 1은 사이버 보안 위협 정보 처리 방법의 일 실시 예를 예시한 도면

도 2는 개시하는 실시 예에 따라 분석 정보 생성하는 과정에서 정적 분석 정보를 얻는 예를 개시한 도면

도 3은 개시하는 실시 예에 따라 분석 정보 생성하는 과정에서 동적 분석 정보를 얻는 예를 개시한 도면

도 4은 개시하는 실시 예에 따라 분석 정보 생성하는 과정에서 심층 분석 정보를 얻는 예를 개시한 도면

도 5는 심층 분석의 일 예로서 악성 코드를 디스어셈블링하여 악성 행위가 포함된 파일임을 판단하는 예를 개시한 도면

도 6은 개시하는 실시 예에 따라 분석 정보 생성하는 과정에서 연관관계 분석 정보를 산출하는 일 예를 개시한 도면

도 7은 개시한 실시 예에 따라 연관관계 분석 정보를 얻는 과정의 일 예를 개시한 도면

도 8은 실시 예에 따라 사이버 위협 정보의 예측 정보 생성하는 일 예를 개시한 도면

도 9는 실시 예에 따라 사이버 위협 정보를 제공하기 위한 악성 코드 질의들의 예를 개시한 도면

도 10은 사이버 보안 위협 정보 처리 장치의 일 실시 예를 개시한 도면

도 11은 개시하는 실시 예에 따라 분석 프레임 워크 중 정적 분석 모듈의 기능을 상세히 설명하기 위한 일 예를 나타낸 도면

도 12는 개시하는 실시 예에 따라 분석 프레임 워크 중 동적분석 모듈의 기능을 상세히 설명하기 위한 일 예를 나타낸 도면

도 13은 개시하는 실시 예에 따라 분석 프레임 워크 중 심층분석 모듈의 기능을 상세히 설명하기 위한 일 예를 나타낸 도면

도 14은 개시하는 실시 예에 따라 분석 프레임 워크 중 연관관계분석 모듈의 기능을 상세히 설명하기 위한 일 예를 나타낸 도면

도 15는 개시하는 실시 예에 따라 예측 프레임 워크의 예측정보생성 모듈의 기능을 상세히 설명하기 위한 일 예를 나타낸 도면

도 16은 개시하는 실시 예에 따라 정적 분석을 수행하는 일 예를 나타낸 도면

도 17은 개시하는 실시 예에 따라 동적 분석을 수행하는 일 예를 나타낸 도면

도 18은 개시하는 실시 예에 따라 심층 분석을 수행하는 일 예를 나타낸 도면

도 19는 개시하는 실시 예에 따라 바이너리 코드에서 추출된 코드들로 공격 기법을 매칭하는 일 예를 나타낸 도면

도 20은 개시하는 실시 예에 따라 OP-CODE를 포함하는 코드 세트와 공격 기법을 매칭하는 일 예를 나타낸 도면

도 21은 개시하는 실시 예에 따라 사이버 위협 정보를 처리하는 흐름을 예시한 도면

도 22는 개시하는 실시 예에 따라 OP-CODE 및 ASM-CODE를 정규화된 코드로 변환한 값을 예시한 도면

도 23은 개시하는 실시 예에 따라 OP-CODE 및 ASM-CODE의 벡터화된 값을 예시한 도면

도 24는 개시하는 실시 예에 따라 코드의 블록 단위를 해쉬 값으로 변환하는 예를 개시한 도면

도 25는 개시하는 실시 예에 따른 앙상블 머신 러닝 모델의 일 예를 나타낸 도면

도 26은 개시하는 실시 예에 따라 머신 러닝으로 데이터를 학습하고 분류하는 흐름을 예시한 도면

도 27은 개시하는 실시 예에 따라 학습 데이터로 공격 식별자와 공격자를 식별하여 라벨링을 수행한 예를 나타낸 도면

도 28은 실시 예에 따라 공격 식별자를 식별한 결과를 나타낸 도면

도 29는 실시 예에 따라 공격 식별자에 따른 그램 데이터 패턴을 예시한 도면

도 30은 개시한 사이버 위협 정보를 처리하는 실시 예의 성능를 예시한 도면

도 31은 사이버 위협 정보의 탐지하는 엔진들의 탐지 엔진들을 탐지 명을 제공하는 예를 나타낸 도면

도 32는 실시 예에 따라 새로운 악성 코드와 공격 방식을 예시하는 일 예를 나타낸 도면

도 33은 사이버 보안 위협 정보 처리 방법의 다른 일 실시 예를 예시한 도면

도 34는 사이버 보안 위협 정보 처리 장치의 다른 일 실시 예를 예시한 도면

도 35는 실시예들에 따른 사이버 보안 위협 정보 처리 장치의 동작을 나타낸 블록도

도 36은 실시예들에 따른 사이버 보안 위협 정보 처리 장치의 동작을 나타낸 블록도

도 37은 실시예들에 따른 사이버 보안 위협 정보 처리 방법을 나타내는 플로우 다이어그램

도 38은 실시예들에 따른 사이버 보안 위협 정보 처리 장치의 예시

도 39는 실시예들에 따른 사이버 보안 위협 정보 처리 방법의 플로우 다이어그램

발명을 실시하기 위한 구체적인 내용

[0022] 이하에서는 첨부한 도면을 참조하여 실시 예를 예시하여 상세히 기술하도록 한다. 실시 예에서 프레임워크, 모듈, 응용 프로그램 인터페이스 등은 물리 장치 결합된 장치로 구현할 수도 있고 소프트웨어로 구현할 수도 있다.

[0023] 실시 예가 소프트웨어로 구현될 경우 저장매체에 저장되고 컴퓨터 등에 설치되어 프로세서에 의해 실행될 수 있다.

[0024] 사이버 보안 위협 정보 처리 장치 및 사이버 보안 위협 정보 처리 방법의 실시 예들을 상세히 개시하면 다음과 같다.

[0025] 도 1은 사이버 보안 위협 정보 처리 방법의 일 실시 예를 예시한 도면이다. 사이버 보안 위협 정보 처리 방법의 일 실시 예를 설명하면 다음과 같다.

[0026] 사이버 보안 위협 정보 처리 장치로 입력된 파일의 전처리를 수행한다(S1000).

[0027] 파일의 전처리를 통해 파일을 식별할 수 있는 식별 정보를 얻을 수 있다. 파일의 전처리 수행의 일 예는 다음과 같다.

[0028] 수신한 파일로부터 파일의 출처 정보, 파일을 얻은 수집 정보, 파일의 사용자 정보 등을 포함한 여러 가지 메타 정보를 얻을 수 있다. 예를 들어 파일이 URL (uniform resource locator)을 포함하거나 또는 전자메일에 포함된 경우 파일에 대한 수집 정보를 얻을 수 있다. 사용자 정보는 파일의 생성, 업로드 또는 최종 저장한 사용자 정보 등을 포함할 수 있다. 전처리 과정에서 파일의 메타 정보로서 IP(internet protocol) 정보, 이에 기반한 국가 정보, API(Application Programming Interface) key 정보, 예를 들면 분석을 의뢰한 사용자의 API 정보 등을 얻을 수 있다.

[0029] 전처리 과정에서 파일의 해쉬(Hash) 값을 추출할 수도 있다. 해쉬 값이 이미 사이버 보안 위협 정보 처리 장치에 알려진 것이라면 이를 기반으로 파일의 종류나 위험 정도를 식별할 수 있다.

[0030] 만약 이미 알려진 파일이 아니라면 기 저장된 정보 또는 필요한 경우 외부의 레퍼런스 웹 사이트(reference website)에 해쉬 값과 파일 정보를 조회하여 파일 종류 식별을 위한 분석 정보를 얻을 수 있다. 예를 들어 외부의 레퍼런스 웹 사이트로서 한국인터넷진흥원에서 운영하는 C-TAS(Cyber Threats Analysis System), CTA(Cyber Threat Alliance)의 운영시스템, VitusTotal 등의 사이트로부터 파일 종류에 따른 정보를 얻을 수 있다.

[0031] 예를 들면, 파일의 MD5 (Message-Digest algorithm 5), SHA1 (Secure Hash Algorithm 1), SHA 256 등의 해쉬 함수의 해쉬 값을 이용하여 해당 사이트에서 파일을 검색할 수 있다. 그리고 검색 결과를 이용해 상기 파일을 식별할 수 있다.

[0032] 파일을 분석을 수행하는 일 예로서, 입력된 파일이 모바일 네트워크를 통해 전송될 경우 네트워크 트래픽을 통해 전송되는 패킷은 네트워크 전송 패킷의 재조합 기술 등을 사용하여 입력된 파일이 모바일 악성 의심 코드인 경우 이를 저장할 수 있다. 패킷의 재조합 기술은 수집된 네트워크 트래픽에서 하나의 실행 코드에 해당하는 일련의 패킷들을 재 조합하며, 재 조합된 패킷들에 의해 전송되는 파일이 모바일 악성 의심 코드인 경우 이 파일이 저장된다.

[0033] 만약 이 단계에서 전송 파일 내에 모바일 악성 의심 코드 추출이 되지 않은 경우 파일 내에 다운로드 URL에 직접 접속하여 모바일 악성 의심 코드를 다운로드하여 저장할 수도 있다.

[0034] 상기 입력된 파일과 관련된 악성 행위(malicious activity) 분석 정보 생성한다(S2000).

[0035] 입력된 파일과 관련된 악성 행위의 분석 정보는 파일 자체에 대한 정보를 분석하는 정적 분석 정보나 입력된 파일로부터 얻은 정보를 실행하여 악성 행위 여부를 판별할 수 있는 동적 분석 정보를 포함할 수 있다.

[0036] 이 단계의 분석 정보는 입력된 파일과 관련된 실행 파일로부터 가공된 정보를 이용하거나 파일과 관련된 메모리 분석을 수행하는 심층 분석 정보를 포함할 수 있다.

[0037] 심층 분석은 악성 행위를 정확하게 식별할 수 있도록 인공 지능 분석을 포함할 수 있다.

[0038] 이 단계의 분석 정보는 또한 파일과 관련하여 이미 저장된 분석 정보나 또는 생성된 분석 정보를 서로 연관시켜 공격 행위나 공격자에 대한 연관 관계를 추정할 수 있는 연관관계 분석 정보를 포함할 수 있다.

[0039] 이 단계에서 다수의 분석 정보는 전체 분석 결과로 제공되기 위해 취합될 수 있다.

[0040] 예를 들어 하나의 파일에 대한 정적 분석 정보, 동적 분석 정보, 심층 분석 정보, 연관관계 분석 정보 등은 정확한 공격 기법과 공격자 식별을 위해 통합 분석될 수 있다. 통합 분석은 분석 정보 사이의 중복된 부분을 제거하고 분석 정보 간 공통의 정보는 정확도를 높이는데 사용될 수 있다.

[0041] 예를 들어 여러 분석과 경로를 통해 수집된 사이버 위협 침해 정보(indicator of compromise, IoC)들은 정보들 사이에 노멀라이징(normalizing)하거나 인리치먼트(enrichment) 수행을 통해 표준화 작업을 수행할 수 있다.

[0042] 분석 정보의 획득하는 실시 예에서 반드시 위의 기술된 모든 분석 정보를 순서에 따라 산출할 필요는 없다. 예를 들어 정적 분석 정보 획득과 동적 분석 정보 획득은 어느 하나만 진행될 수도 있으며 정적 분석 정보 보다 동적 분석 정보를 먼저 수행할 수도 있다.

[0043] 심층 분석 정보는 반드시 정적 분석 또는 동적 분석을 수행한 후 진행될 필요가 없으며, 연관 관계 분석도 심층 분석 정보 없이 수행될 수도 있다.

[0044] 따라서 위 분석 정보를 획득하는 처리 순서는 변경될 수도 있으며 선택적으로 이루어질 수도 있다. 또한 위에 기술한 분석 정보의 획득 과정과 예측 정보의 생성 과정은 파일로부터 획득한 정보에 기초하여 병렬적으로 수행될 수 있다. 예를 들면 동적 분석이 수행이 완료되지 않더라도 연관관계 분석 정보를 생성할 수도 있다. 마찬가지로 동적 분석 수행이나 심층 분석 수행이 동시에 진행될 수 있다.

[0045] 이러한 경우 위에서 예시한 전처리 과정(S1000)은 파일의 정보를 얻거나 식별하기 위한 것이므로 정적 분석, 동적 분석, 심층 분석 또는 연관 분석이 개별적이나 병렬적으로 수행될 경우 각 분석 단계에 일부로서 각각 수행될 수 있다.

[0046] 이 단계에 대한 상세한 실시 예는 아래에서 후술한다.

[0047] 상기 입력된 파일과 관련된 악성 행위의 예측 정보를 생성할 수 있다(S3000).

[0048] 분석 정확도를 높이기 위해 위의 분석된 여러 가지 정보의 데이터 세트를 이용하여 악성 행위의 발생 여부, 공격 기법, 공격자 그룹 등에 대한 예측 정보를 생성할 수 있다.

[0049] 예측 정보의 생성은 이미 분석된 데이터 세트에 대한 인공지능 분석을 통해 수행될 수 있다. 예측 정보의 생성은 필수적인 단계가 아니며 인공지능 분석을 위해 적절하게 분석된 데이터 세트가 마련되어 조건이 만족될 경우 추후 악성 공격 행위에 대한 예측 정보를 생성할 수 있다.

[0050] 실시 예는 여러 가지 분석 정보들을 기반으로 인공 지능 기반의 머신 러닝을 수행한다. 실시 예는 분석된 정보에 대한 데이터 세트를 기반으로 예측 정보를 생성할 수 있다. 예를 들면 인공 지능으로 학습된 데이터를 바탕으로 추가적인 분석 정보를 생성하고 다시 생성된 분석 정보는 다시 새로운 학습 데이터로서 인공 지능의 입력 데이터로 이용될 수 있다.

[0051] 여기서 예측 정보는 악성 코드 제작자 정보, 악성 코드 공격 방법 정보, 악성 코드 공격 그룹 예측, 악성 코드 유사도 예측 정보, 및 악성 코드 확산도 예측 정보 등을 포함할 수 있다.

[0052] 생성된 예측 정보는 악성 코드 자체의 위험도를 예측한 제 1 예측 정보와 악성 코드의 공격자, 공격 그룹, 유사도, 확산도 등을 예측한 제 2 예측 정보 등을 포함할 수 있다.

[0053] 이러한 제 1 예측 정보와 제 2 예측 정보를 포함하는 예측 분석 정보는 서버나 데이터 베이스에 저장될 수 있다.

[0054] 이에 대한 상세한 실시 예는 이하에서 후술한다.

[0055] 상기의 분석 정보 또는 예측 정보에 대한 후처리 후 상기 입력된 파일과 관련된 사이버 위협 정보를 제공한다(S4000).

[0056] 실시 예는 분석 정보 또는 예측 정보에 기초하여 악성 코드 종류 및 악성 코드의 위험도를 결정한다. 그리고 실시 예는 악성 코드에 대한 프로파일링 정보를 생성한다. 따라서 파일 분석을 통해 파일에 대한 자체 분석을 수행한 결과나 추가 및 예측 분석을 수행한 결과를 저장할 수 있다. 생성되는 프로파일링 정보는 악성 코드에 대한 공격 기법이나 공격자에 대한 라벨링을 포함한다.

[0057] 사이버 위협 정보는 위의 전처리가 수행된 정보, 생성되거나 식별된 분석 정보, 생성된 예측 정보 또는 이 정보들의 취합 정보나 이 정보들을 기반으로 결정된 정보를 포함할 수 있다.

[0058] 제공되는 사이버 위협 정보에는 입력된 파일과 관련하여 데이터 베이스에 저장된 분석 정보를 이용하거나 위에서 분석되거나 예측된 정보가 포함될 수 있다.

[0059] 실시 예에 따르면 사용자가 입력된 파일에 대한 악성 행위뿐만 아니라 이미 저장된 파일이나 악성 행위에 대해 사이버 위협 정보를 조회할 경우 이에 대한 정보를 제공할 수 있다.

[0060] 이러한 통합 분석 정보는 해당 파일에 대응하여 서버나 데이터 베이스에 표준화된 포맷으로 저장될 수 있다. 이러한 통합 분석 정보는 표준화된 포맷으로 저장되어 사이버 위협 정보를 검색 또는 조회에 사용될 수 있다.

[0061] 사용자의 사이버 위협 정보의 조회에 대항 추가적인 예시는 이하에서 상세히 후술한다.

[0063] 도 2는 개시하는 실시 예에 따라 분석 정보 생성하는 과정에서 정적 분석 정보를 얻는 예를 개시한다.

[0064] 개시하는 실시 예에 따른 정적 분석 정보를 획득하는 단계는, 입력된 파일의 구조 정보를 얻고 분석하는 단계를 포함할 수 있다(S2110).

[0065] 실시 예는 파일이 실행되지 않는 환경에서 먼저 식별된 파일 기본적인 구조 정보를 분석할 수 있다. 이 단계에서는 예를 들어 파일의 종류가 ELF(Executable and Linkable Format), PE(Portable Executable), APK(Android Application Package) 등에 파일 종류가 다르더라도 파일의 위 파일 구조나 그 구조로부터 추출할 수 있는 정보를 획득하거나 분석한다.

[0066] 참고로 예시하는 정적 분석에서 파일의 식별은 개시한 전처리 단계에서 수행될 수도 있는데 이러한 경우 S210 단계의 분석 단계는 전처리 단계와 함께 수행될 수 있다.

[0067] 그리고 입력된 파일의 패턴 분석을 수행할 수 있다(S2120).

[0068] 여기서는 식별된 파일에 대해 파일 패턴을 분석하는 경우로서 파일에 어떤 조치를 취하지 않고 파일 자체를 오픈하여 추출할 수 있는 여러 스트링(string) 등을 확인하여 파일의 패턴을 얻을 수 있다.

[0069] 입력된 파일이 제작과 관련된 정보를 얻고 분석할 수 있다(S2130).

[0070] 실시 예는 파일이 가지고 있는 고유 정보나 메타 정보, 예를 들면 파일 제작자 정보, 실행 파일인 경우 코드사이닝(codesigning) 정보 등을 얻을 수 있다.

[0071] 그리고 입력된 파일의 환경 정보를 분석할 수 있다(S2140).

[0072] 여기서는 대상 파일이 갖추어야 할 시스템 환경적 구성 요소 정보 등에 정보를 얻을 수 있다.

[0073] 그리고 입력된 파일과 관련된 여러 가지 기타 정보들을 분석하고 저장한다(S2150). 이러한 파일의 수행 없이 파일 자체의 정적 정보를 특정 파일 포맷, 예를 들어 JSON (JavaScript Object Notation)과 같은 데이터 포맷으로 저장할 수 있다.

[0074] 정적 분석의 예는 파일 자체를 분석하는 것으로서 코딩 기반의 취약 항목 존재 여부, 인터페이스 또는 함수의 호출 구조 문제, 또는 파일의 바이너리 구조 등을 얻을 수 있다.

[0075] 위에서 개시한 정적 정보를 분석하는 일 예를 편의상 플로우 차트로 나타내었으나, 위 단계들은 반드시 위에서 기술되거나 도면에서 표시된 순서로 수행될 필요가 없다. 또한 파일에 따라 이 도면에서 개시한 모든 단계를 수행할 필요도 없으며 정적 분석 정보를 얻기 위해 일부 단계, 예를 들면 구조 정부 분석, 제작 관련 정보 분석

및 환경 정보 분석을 선택적으로 수행할 수도 있다. 즉 이에 대한 실시 순서와 실시 단계의 선택의 당업자의 선택에 따라 달라질 수 있다.

[0076] 개시된 실시 예에 따라 정적 분석 정보를 획득하는 예들을 간략하게 설명하면 다음과 같다.

[0077] 정적 분석을 수행하는 일 예로서, 전처리 과정에서 입력된 파일의 해쉬(Hash) 값을 추출할 경우 추출된 파일의 해쉬 값과, 악성코드에 대해 이미 저장된 해쉬 값과 비교하여 상기 입력된 파일이 악성코드 여부를 분석할 수 있다. 분석된 기반으로 파일 내에 악성 코드가 있는지 탐지할 수 있다.

[0078] 만약, 입력 파일이 모바일 데이터 인 경우 입력된 파일로부터 모바일 악성 의심 코드의 코드 정보를 추출한다. 여기서, 코드 정보란 모바일 악성 의심 코드를 실행하지 않고 코드 자체로부터 추출할 수 있는 정보를 의미하는 것으로, 예를 들어, 해쉬(Hash) 정보, 코드 크기 정보, 파일 헤더 정보, 코드 내에 포함되어 있는 식별 가능한 문자열 정보 및 동작 플랫폼 정보 등을 포함할 수 있다.

[0079] 설명한 바와 같이 이와 같이 획득된 정적 분석 정보는 해당 파일에 대응하여 저장될 수 있다.

[0081] 도 3은 개시하는 실시 예에 따라 분석 정보 생성하는 과정에서 동적 분석 정보를 얻는 예를 개시한다.

[0082] 전처리로부터 식별된 파일 정보 또는 정적 분석 정보 중 적어도 하나에 기반하여 식별된 파일의 실행 환경에서 실행된 결과 데이터에 따른 동적 분석 정보를 획득할 수 있다

[0083] 개시하는 실시 예에 따른 동적 분석 정보를 획득하는 단계는 파일이 실행 중인 환경에서 다양한 입출력 데이터를 분석하거나 또는 파일 실행 시 실행 환경과 상호작용의 변화를 분석하여 취약하거나 위험한 이상현상을 탐지하는 단계이다. 일반적으로 가상화 환경에서 파일을 직접적으로 실행하여 이상 여부를 분석한다.

[0084] 동적 분석을 수행하기 위해 실시 예는 입력 파일을 실행하기 위한 동적 분석 환경을 생성하고 준비한다(S2210). 입력된 파일의 타입을 식별한 경우 각각의 파일의 타입에 따라 어떤 실행 환경이 필요한지 알 수 있다. 예를 들면 파일에 따라 윈도우 운영체제, 리눅스 운영체제, 모바일 기기 운영체제에서 실행되는 파일인지 식별할 수 있다.

[0085] 준비된 분석 환경에서 악성 코드 여부를 판별하기 위해 획득된 파일을 실행한다(S2220).

[0086] 동적 분석 정보를 획득하기 위해 이러한 실행 환경에서 파일을 실행하여 해당 시스템에서 발생하는 이벤트를 수집할 수 있다(S2230). 예를 파일 자체, 프로세스, 메모리, 레지스트리, 네트워크의 시스템에 대한 이벤트 또는 각 시스템의 설정을 변경시키는 이벤트를 수집할 수 있다. 그리고, 수집된 이벤트들을 개별적으로 또는 취합하여 분석한다.

[0087] 수집된 결과를 취합한 후 동적 분석을 위한 환경을 다시 복구한다(S2240).

[0088] 이와 같이 획득된 결과는 해당 파일에 대응된 동적 분석 정보로 저장될 수 있다.

[0090] 이하에서 이와 같은 동적 분석 정보를 획득하는 실시 예에 따라 동적 분석 정보를 수집하고 분석하는 예를 간략하게 개시한다.

[0091] 동적 분석의 일 실시 예로서, 입력된 파일이 모바일 기기 운영 체제에서 동작하는 파일로 식별된 경우, 파일을 모바일 단말 또는 모바일 단말 환경과 동일하게 구성된 에뮬레이터나 가상화 환경에서 직접 실행한다. 그리고 파일 내에 모바일 악성 의심 코드가 실행된 후에 단말에 발생하는 모든 변화, 즉 행위 정보를 추출하고 기록한다. 행위 정보는 단말의 운영체제(OS) 환경에 따라 상이하나, 통상적으로 프로세스, 파일, 메모리 및 네트워크 정보 등의 이벤트 정보를 포함할 수 있다.

[0092] 동적 분석의 다른 실시 예로서 전처리 과정에서 입력된 파일의 해쉬(Hash) 값을 추출되지 않고 사용자 단말에서 해쉬 값이 추출된 경우라도, 단말에서 추출된 파일의 해쉬 값을 인텔리전스 플랫폼을 통해 수신할 수 있다.

[0093] 데이터베이스에 해당 파일의 해쉬 값이 이미 저장되지 않는 경우 수신된 파일을 가상 또는 실제의 운영체제에서 실행시키고, 실행 시에 발생되는 행위를 실시간으로 수집하고 수집된 동적분석 정보를 데이터베이스에 이미 저장된 정보와 비교할 수 있다.

[0094] 상기 비교 결과 이미 정의된 위험도를 초과하는 경우 입력된 파일이 악성 코드를 포함하고 있다고 판단할 수 있

고, 해당 파일의 해쉬 값을 데이터 베이스에 저장하여 추후 정적 분석 등에 이용할 수 있다.

[0095] 악성 코드에 따라 행위 주체가 되는 제 1 프로세스가 시스템에 위험한 행위를 발생하는 경우도 있다. 그러나, 경우에 따라 상기 제 1 의 프로세스의 행위가 추가적으로 자식 프로세스인 제 2 프로세스를 추가로 생성하고 상기 제 2 프로세스가 시스템에 악성 행위를 수행하는 경우도 있다.

[0096] 이러한 경우, 동적 분석의 일 실시 예는 최초의 제 1 의 프로세스의 행위가 실행 시스템에 발생시키는 이벤트들을 저장하고, 추가적으로 제 1 프로세스의 자식 프로세스인 제 2 프로세스를 추출 또는 확인하여 상기 제 2 프로세스에 따른 악성 행위의 이벤트를 저장할 수도 있다. 이와 같이 이 예에서 동적 분석은 최초의 제 1 프로세스와 그와 연결될 제 2, 3의 프로세스의 이벤트 정보도 종합적으로 분석하여 식별된 파일이 악성 코드를 포함하는지 판단할 수 있다.

[0097] 입력된 파일의 실행 결과에 따라 알려지지 않은 악성 코드의 특성이 없는 경우는 악성 코드의 특성을 가지고 있더라도 탐지하기 어려운 경우 있다. 이러한 경우 동적 분석의 또 다른 실시 예는 식별된 파일이 실행 시에 외부와 통신하는 네트워크 프로세스를 모니터링하고 분석하여 상기 실행 프로세스의 악성 행위를 탐지할 수 있다.

[0098] 예를 들면 식별된 파일을 실행한 경우 외부와 통신하는 네트워크 이벤트를 모니터링할 수 있다. 파일 실행에 따라 로컬 어드레스 오브젝트(local address object)를 생성한 프로세스 아이디(Process IDentifier, PID)를 저장한다. 그리고, 상기 파일 실행과 관련된 네트워크 이벤트가 발생될 경우 해당 네트워크 이벤트의 IRP(Interior Router Protocol) 정보로부터 로컬 어드레스 오브젝트 정보들을 추출할 수 있다.

[0099] 상기 프로세스 아이디가 생성한 로컬 어드레스 오브젝트와 상기 네트워크 이벤트와 관련된 로컬 어드레스 오브젝트들을 비교하여 악성 행위를 판단하는 동적 분석을 수행할 수 있다. 예를 들면 상기 네트워크 이벤트에 따라 송수신되는 패킷의 패턴이나 또는 패킷 전송을 유발하는 C&C (Control and Command) 서버를 확인하여 악성 행위 여부를 판단할 수 있다.

[0100] 동적 분석의 또 다른 실시 예로서, 주소 결정 프로토콜(Address Resolution Protocol, ARP) 스푸핑 (spoofing) 공격을 방지하기 위해 ARP 정보를 모니터닝할 수도 있다. 일반적으로 로컬 영역 네트워크에서 장비의 IP(internet protocol) 주소와 MAC (media access control) 주소간의 대응은 ARP 이나 Neighbor Discovery Protocol (NDP) 이 사용될 수 있다.

[0101] ARP 스푸핑 공격은 공격자가 IP 패킷을 전송할 경우 수신 네트워크 장비의 MAC 주소가 아닌 자신의 MAC 주소에 대응하는 ARP 메시지를 전송하여 이루어진다. 전송된 메시지를 수신한 네트워크 장비는 전송 패킷을 정상적인 IP 주소가 아닌 공격자로 전송하도록 한다.

[0102] 실시 예는 이러한 공격에 대응하기 위하여 네트워크 장비들로부터 직접 수집된 ARP 정보와, 가상 네트워크에 포함된 네트워크 장비들의 SNMP (Simple Network Management Protocol) 정보 내의 ARP 정보를 비교함으로써 ARP 스푸핑 공격 발생 여부를 판단할 수 있다.

[0103] 즉, 동적 분석의 일 실시 예는, 호스트가 네트워크에 연결된 장비들에 ARP 정보 요청 메시지를 전송하여 회신된 ARP 응답 메시지에 포함된 제 1 ARP 정보와, 가상 네트워크에 접속된 장비들의 SNMP 정보 내에 포함된 제 2 ARP 정보를 비교하여 제 1 ARP 정보와 제 2 ARP 정보가 다른 경우 ARP 스푸닝 공격이 발생했다고 판단할 수 있다.

[0104] 이 실시 예는 이러한 동적 분석의 방식을 이용하여 ARP 스푸핑 공격을 탐지하고 호스트 장비에 저장될 기밀 정보 유출을 방지할 수 있다.

[0105] 동적 분석 방식의 또 다른 실시에는 가상 환경을 회피하도록 하는 악성 코드를 분석할 수 있는 방법이다. 여기서 관리 서버와 네트워크를 통해 연결된 단말은 관리 서버에 저장된 제 1 OS (operating system) 이미지를 이용해 부팅을 수행할 수 있다. 단말이 부팅된 후 상기 제 1 OS에 기초하여 악성 코드를 분석한 후, 상기 단말은 관리 서버로부터 제 2 OS 이미지를 수신하고, 수신된 제 2 OS 이미지를 이용해 초기화를 수행한다. 그리고 상기 단말이 악성 코드가 분석 종료된 시그니처를 상기 관리 서버로 전송하도록 한다. 따라서, 제 1 OS에 기초하여 악성 코드를 분석 후에 발행된 악성 행위가 있더라도 상기 관리 서버는 단말이 제 1 OS을 단말에서 삭제하도록 하고 원본 OS 이미지와 동일한 제 2 OS를 기초로 단말이 부팅하도록 함으로써 단말에 악성 행위 발생을 방지하도록 할 수 있다.

[0106] 악성 코드는 외부의 서버와 통신하며 추가적인 명령을 발생시키고 파일을 수신하도록 할 수 있다.

[0107] 그런데 동적 분석을 수행할 수 있는 서버가 중지된 경우는 이러한 동적 분석에 매우 오랜 시간이 소요될 수 있고 해당 행위가 사전 차단된 경우에도 동적 분석을 수행할 수 없는 경우가 있다.

[0108] 동적 분석을 통해 네트워크 행위를 분석하기 위해서는 악성 코드가 사용하는 명령 제어 서버(C&C 서버), 추가적인 악성 코드를 다운로드하기 위한 다운로드 서버 또는 악성 코드들끼리 정보를 주고 받거나 해커와 정보를 주고 받는 커뮤니케이션 패킷 등의 정보를 추출하여 분석해야 한다. 그러나, 이와 같이 관련 서버가 작동하지 않는 경우에는 그러한 정보의 추출할 수 없다.

[0109] 여기서 개시하는 동적 분석 방법의 또 다른 실시 예는 서버가 동작 중지된 경우에도 동적 분석을 수행하도록 할 수 있다.

[0110] 예를 들어 네트워크 접속 유도 장치가 악성 코드에 감염된 클라이언트 단말과 관리 서버에 사이에서 단말의 접속 요청을 처리하도록 하여 동적 분석을 진행하도록 할 수도 있다. 네트워크 접속 유도 장치는 단말로부터 접속 요청을 수신하고 이를 악성 코드 행위를 유발시키는 C&C 서버로 전달하도록 할 수 있다. 그리고, 만약 상기 네트워크 접속 유도 장치가 일정 시간 내에 C&C 서버로부터 응답 패킷을 수신하지 못하면, 상기 네트워크 접속 유도 장치는 별도의 가상의 응답 패킷과 접속 요청을 함께 상기 단말에 전송하도록 한다.

[0111] 이후에 상기 단말로부터 수신된 악성 코드 분석에 관련된 데이터를 추출할 수 있다.

[0112] 가상의 응답 패킷을 이용하는 예는 가상의 응답 패킷 TCP 세션을 생성하기 위한 패킷 형식이면 충분하다. 악성 코드가 사용하는 일반적인 TCP (Transmission Control Protocol) 프로토콜은 TCP 세션만 생성하도록 상기 클라이언트 단말이 전송하는 데이터 패킷을 생성할 수 있다. 그리고 상기 데이터 패킷으로부터 악성 코드의 동적 분석에 필요한 중요 정보들을 추출할 수 있다. 이와 같이 하면 관리 서버가 동작하지 않더라도 네트워크 접속 유도 장치의 동작을 이용하여 동적 분석을 수행할 수 있다.

[0113] 이와 같이 실시 예는 수신된 파일을 실행하여 발행하는 이벤트를 분석할 수 하고 동적 분석 정보를 데이터베이스에 저장할 수 있다.

[0115] 도 4는 개시하는 실시 예에 따라 분석 정보 생성하는 과정에서 심층 분석 정보를 얻는 예를 개시한다.

[0116] 개시하는 실시 예에 따른 심적 분석 정보를 획득하는 단계는 수신된 파일 포함하는 실행 가능한 파일 디스어셈블링(disassembling)하여 기계 언어 레벨에서 분석하여 악성 행위를 유발하는 공격 기법이나 공격자를 식별하는 특징을 포함한다.

[0117] 심층 분석 정보는 기술한 정적 분석이나 동적 분석의 결과를 이용하여 얻을 수도 있고, 분석자의 해석 기준에 따라 실행 가능한 파일을 악성 행위를 유발하는 파일로 분석할 수 있다.

[0118] 또한 심층 분석 정보는 파일 자체의 분석 정보나 또는 파일을 여러 번 가공한 정보를 포함할 수 있고 이미 저장된 정보를 기반으로 수행될 수 있다.

[0119] 심층 분석은 디스어셈블링(disassembling), 디스어셈블된 기계언어레벨의 코드추출, 공격행위(TTP)식별, 공격자 식별, 테인트분석(taint analysis)을 수행하는 단계를 포함할 수 있다.

[0120] 도면을 참조하여 상세히 예시하면 다음과 같다.

[0121] 입력된 파일이 실행 가능한 파일을 포함할 경우 심층 분석은 실행 가능한 파일을 디스어셈블(disassemble)한다(S2410).

[0122] 디스어셈블(disassemble)된 어셈블리 코드(assembly code)들은 OP-CODE(operation code)와 피연산자(operand)를 포함할 수 있다. OP-CODE(operation code)는 명령어 코드로 호칭할 수는 기계 언어 명령어를 나타내고, 피연산자(operand)는 실행 동작에 필요한 정보, 즉 기계 언어 명령어의 대상 데이터나 메모리 위치를 나타낸다.

[0123] 이하에서는 편의상 디스어셈블(disassemble)된 어셈블리 코드(assembly code)들 중 OP-CODE를 제외한 부분을 ASM-CODE로 호칭하도록 한다. 따라서, 이하에서 ASM-CODE 는 피연산자(operand) 부분을 포함할 수 있다.

[0124] 디스어셈블링(disassembling)을 통해 오브젝트 코드 형식의 실행 가능한 파일은 특정 형식, 예를 들면 어셈블러 언어 형식의 코드 또는 디스어셈블된 코드로 변환된다. 이러한 디스어셈블된 코드로부터 일정 형식을 가진 OP-CODE (operation code) 와 ASM-CODE를 추출할 수 있다 (S2420).

[0125] 추출된 디스어셈블드 코드를 일정 형식의 데이터 포맷을 변환할 수 있다. 일정 형식의 데이터 포맷의 변환 예시는 아래에서 개시한다.

[0126] 심층 분석은 추출된 디스어셈블된 코드나 상기 일정 형식으로 변환된 데이터 포맷을 기반으로 공격행위를 식별할 수 있다(S2430).

[0127] 디스어셈블된 코드 내에 OP-CODE는 수행될 연산을 특정하는 기계 언어 명령어의 일부인데, 사이버 보안 상 공격 행위 또는 공격 기법(Terrorist Tactics, Techniques, and Procedures, 이하 TTP)을 유발하는 OP-CODE는 해당 공격 행위 별로 매우 유사한 값이나 포맷을 가질 수 있다. 따라서, 이러한 OP-CODE와 ASM-CODE 를 분석하면 특정 공격 행위를 구별할 수 있다.

[0128] 실행 가능한 파일로부터 디스어셈블된 코드들을 추출하고 추출된 디스어셈블된 코드들은 실행 함수에 따라 분리될 수 있다.

[0129] 예를 들면 디스어셈블된 코드로부터 추출된 OP-CODE와 ASM-CODE 또는 상기 디스어셈블된 코드의 재조합된 코드는 퍼지 해쉬(Fuzzy Hashing) 방식 또는 CTPH (context triggered piecewise hashes) 방식 등의 해쉬 값이나 이를 일정 형식의 코드로 변환할 수 있다.

[0130] 실시 예는 실행 가능한 파일의 디스어셈블된 코드를 일정 형식으로 변환하고 사이버 보안 전문가 집단들이 공통적으로 인정하는 공격 행위 세부 요소들로 매칭하도록 하여 그 공격행위를 식별할 수 있다.

[0131] 그리고 이미 추출된 디스어셈블된 코드들과 공격행위(TTP) 별 매칭 관계를 저장한 데이터베이스에 기반하여 공격행위(TTP)를 식별하도록 할 수 있다. 이 경우 추출된 디스어셈블된 코드들의 CTPH 알고리즘에 따른 퍼지 해쉬 값이나 이를 일정 형식으로 변환한 데이터와 공격 행위(TTP) 별 매칭 유사도를 고속으로 수행할 수 있다.

[0132] 이러한 보안 전문가 집단의 공격 행위를 저장한 데이터 베이스의 일 예로서 MITRE ATT&CK 등의 정보를 저장한 데이터베이스를 예로 들 수 있다. MITRE ATT&CK은 실제 보안 공격 기법이나 행위에 대한 데이터 베이스의 하나로서, 특정 보안 공격 기법이나 행위들을 매트릭스 형식의 구성 요소들로 표시함으로써, 공격 기법과 행위들을 일정한 데이터 세트 형식으로 식별할 수 있도록 한다.

[0133] MITRE ATT&CK는 해커 또는 악성 코드의 공격 기법에 대한 내용을 공격의 단계 별로 분류하여 CVE 코드(Common Vulnerabilities and Exposures Code)의 매트릭스로 표현한다.

[0134] 실시 예는 디스어셈블된 코드를 분석함으로써 여러 가지 공격 행위들 중 특정 공격 행위를 식별하되, 식별된 타입의 공격 행위가 전문가 단체들이 인정하는 실제 수행되는 공격 코드들에 매칭되도록 함으로써 공격 행위 식별이 전문적이면서 공통으로 인식되는 요소들로 표현되도록 할 수 있다.

[0135] 디스어셈블된 코드 내에 OP-CODE는 특정 행위를 유발시키는 기계 언어 명령어이므로, 동일한 공격 행위를 유발하는 파일의 OP-CODE 는 매우 유사할 수 있다. 그러나 동일 공격 행위와 이를 유발하는 파일에 포함된 OP-CODE가 정확하게 완전히 동일한 것은 아니므로, 실시 예는 OP-CODE를 포함하는 디스어셈블링된 코드에 대해 인공 지능 기반의 머신 러닝을 수행하도록 할 수 있다. 머신 러닝이 수행되면 임계 치 이상의 유사도를 가진 공격 코드의 포함 여부와 공격 코드의 공격 기법이 식별될 수 있다.

[0136] 따라서, 동일한 악성 행위를 유발시키는 파일들의 디스어셈블링된 코드들이 완전히 동일하지 않더라도 디스어셈블링된 코드기반으로 악성 행위를 수행하는 파일을 식별할 수 있다.

[0137] 머신 러닝 알고리즘으로 Perceptron, Logistic Regression, Support Vector Machines, Multilayer Perceptron 등의 알고리즘이 사용될 수 있다.

[0138] 디스어셈블된 코드들의 퍼지 해쉬 값들의 유사도를 AI(Artificial Intelligence; 이하 AI) 알고리즘을 이용하여 기존에 학습된 MITRE ATT&CK과 같은 공격 기법의 공격 코드들로 매칭하여 최종적으로 악성 코드임을 탐지할 수 있다

[0139] 그리고 실시 예는 인공 지능 머신 러닝의 결과를 이용하면 보다 정확성을 가지고 신속하게 디스어셈블된 코드에 대응되는 공격 행위 또는 공격 행위의 취약 요소들을 식별할 수 있다.

[0140] 이에 대한 구체적인 실시 예들은 이하에서 도면을 참고하여 상세히 개시한다.

[0141] 심층 분석의 실시 예는 디스어셈블된 코드와 인공 지능 기반의 머신 러닝 결과를 이용해 유사 공격 행위를 유발하는 공격자도 식별하는 단계를 포함할 수도 있다(S2440). 마찬가지로 공격자 식별에 대한 구체적인 예는 후술

한다

[0142] 그리고 심층 분석의 실시 예는 파일이 없는(fileless) 악성 코드의 경우도 특정 시점에서 시스템의 메모리 분석을 통해 공격 행위가 있는지 여부에 대해 판단할 수 있는 테인트분석(taint analysis)을 포함할 수 있다(S2450).

[0143] 심층 분석은 실행 파일의 디스어셈블링된 코드를 처리하는 것에 기반하며 이에 따른 공격 기법이나 공격자의 식별, 또는 테인트 분석은 선택적으로 수행될 수도 있다.

[0144] 이와 같이 수행된 최종 심층 분석 정보는 해당 파일에 대응되는 심층 분석 정보로 데이터베이스에 저장할 수 있다.

[0146] 도 5는 심층 분석의 일 예로서 악성 코드를 디스어셈블링하여 악성 행위가 포함된 파일임을 판단하는 예를 개시한다.

[0147] 기술한 바와 같이 실행 가능한 파일을 디스어셈블링을 수행하면 어셈블리 언어 형식의 코드의 형식인 OP-CODE와 ASM-CODE를 얻을 수 있다.

[0148] 예를 들어 EXE 실행 파일 내에 특정 함수 A는 디스어셈블러(disassembler)를 거치면 OP-CODE를 포함하는 디스어셈블링된 코드 또는 디스어셈블드 코드(disassembled cocde)로 변환될 수 있다.

[0149] 만약 EXE 실행 파일이 악성 행위를 유발하는 악성 코드인 경우, 이러한 행위를 유발하는 함수나 코드 부분을 디스어셈블링하면 악성 행위를 유발하는 디스어셈블드 코드 세트를 얻을 수 있다.

[0150] 디스어셈블드 코드 세트는 상기 악성 행위 또는 악성 코드에 대응되는 OP-CODE 세트 또는 OP-CODE 와 ASM-CODE가 조합된 세트를 포함할 수 있다.

[0151] 악성 행위가 동일하더라도 이를 수행하도록 하는 악성 코드의 알고리즘이나 실행 파일의 디스어셈블링 결과가 정확하게 같지 않기 때문에 인공 지능 기반의 유사도 분석을 통해 입력된 악성 코드가 특정 디스어셈블드 코드 세트와 대응되는지를 식별할 수 있다.

[0152] 이렇게 특정 디스어셈블드 코드 세트와 대응되는 악성 행위를, MITRE ATT&CK와 같은 전문적이고 공용의 공격 방식 또는 공격 기법에 대응시켜 공격 기법 (TTP)를 식별하는데 사용할 수 있다.

[0153] 또는 특정 디스어셈블드 코드 내 OP-CODE 세트 또는 OP-CODE 와 ASM-CODE가 조합된 세트를 MITRE ATT&CK에서 정의한 공격 기법 요소들과 대응시켜 공격 기법을 판단하는데 사용할 수 있다.

[0154] 이 도면은 실행 파일, 해당 실행 파일의 디스어셈블드 코드 세트와 MITRE ATT&CK에서 공격 기법 요소들에 대응되는 공격 기법을 대응한 예를 나타낸다.

[0157] 도 6은 개시하는 실시 예에 따라 분석 정보 생성하는 과정에서 연관관계 분석 정보를 산출하는 일 예를 개시한다.

[0158] 상기 얻은 여러 가지 분석 정보들은 사이버 위협 침해 정보로 이용될 수 있는데, 사이버 위협 침해 정보에 기반해 공격자 또는 공격 기법의 연관관계를 나타내는 연관관계 분석 정보를 생성한다.

[0159] 사이버 위협 침해 정보(indicator of compromise, IoC)는 시스템이나 네트워크 상에 발생하는 실제 또는 잠재적인 사이버 보안 위협 행위, 공격 행위 또는 악성 행위를 식별하는 여러 가지 정보들을 지칭한다. 예를 들면, 사이버 위협 침해 정보(IoC)는 이러한 행위들을 지칭하는 파일, 로그 정보 상에 나타나는 여러 흔적들, 파일 자체, 경로 등 또는 이런 행위를 추론하도록 하는 정보들을 나타낸다.

[0160] 이미 분석된 정적, 동적, 심층 분석 정보 등과 식별된 파일을 이용하여, 분석 정보와 공격 행위 사이의 IP 정보의 연관관계(S2510), 이메일에 포함되거나 웹사이트의 호스트네임의 연관관계(S2520), URL의 연관관계(S2530), 파일의 코드사인(codesign)의 연관 관계들(S2540)을 얻을 수 있다.

[0161] 여기서 예시하는 연관관계 분석 정보를 획득하는 과정은 일 예로서 반드시 예시한 순서를 따르거나 모든 연관관

계가 분석되어야 하는 것은 아니다. 예를 들어 분석 정보와 공격행위 사이의 IP 와 URL의 연관관계만 이용해도 관련 파일에 대한 연관관계를 얻어낼 수 있다. 이러한 연관관계 분석 정보는 정확하게 공격기법 또는 공격자를 추론하는데 사용될 수 있다.

[0162] 정적 분석, 동적 분석, 심층 분석 등으로 공격 행위나 공격자가 식별되지 않더라도 분석된 정보들 간의 연관관계를 이용하면 공격 행위와 공격자를 추정할 수 있는 정보를 얻을 수 있다. 이에 대한 상세한 설명은 이하에서 도면을 참조하여 설명한다.

[0163] 이러한 연관 관계 분석 정보는 수신되는 파일에 대해 지속적이고 누적적으로 저장하고 추후 새로운 파일을 수신할 때마다 저장된 연관관계 분석 정보는 다시 업데이트할 수 있다.

[0164] 위에서 분석한 여러 가지 분석 정보를 기반으로 사이버 위협 침해 정보를 얻는다.

[0165] 그리고 사이버 위협 침해 정보(IoC)를 이용해 공격 행위나 공격자를 식별할 수 있는 여러 가지 연관관계 정보를 얻을 수 있다(S2550).

[0166] 이러한 사이버 위협 침해 정보(IoC)는 추후에 공격 기법을 추론하는 연관관계 분석 정보를 얻는데 이용될 수 있다. 연관 관계 분석과 이를 이용하여 공격자를 추적 또는 공격 행위를 추론할 수 있는 예는 이하에서 상세히 설명한다.

[0167] 그리고 획득된 연관관계 분석 정보는 해당 파일에 대응하여 다시 서버나 데이터 베이스에 저장될 수 있다.

[0168] 설명한 바와 같이 위와 같이 분석된 정보들은 취합되어 중복 제거, 표준화, 인리치먼트 (enrichment) 과정을 통해 표준화될 수 있다. 예를 들면 정적 분석 정보, 동적분석 정보, 심층분석 정보, 연관관계분석 정보들은 사용자에게 제공되거나 추후 사이버 위협 정보를 갱신 또는 재생산하기 위해 표준화된 포맷으로 저장될 수 있다.

[0169] 여기서 각 분석 정보들의 중복되거나 공통된 분석 정보는 중복된 부분을 제거하고, 부족한 부분의 데이터의 인리치먼트(enrichment) 작업 등을 수행할 수 있다.

[0170] 그리고 사용자의 조회 질의에 따라 또는 서비스 정책에 따라 사이버 위협 정보로 제공될 수 있다. 사이버 위협 정보로 제공에 대해서도 이하에서 상세히 설명한다.

[0171] 이러한 사이버 위협 정보는 사용자에게 직접 제공될 수도 있고 아래에서 설명하는 사이버 위협 예측 정보로 생성된 후 사용자의 요청이나 서비스에 따라 제공될 수도 있다.

[0173] 도 7은 개시한 실시 예에 따라 연관관계 분석 정보를 얻는 과정의 일 예를 개시한 도면이다.

[0174] 이 도면에서 파일 A-1 (10), A-2 (20), B-1 (30)은 악성 행위를 유발할 수 있는 파일을 지칭하고, 서버 (가)(110), 서버 (나)(120)는 악성 행위를 유발시키는 C&C 서버를 나타낸다.

[0175] 개시한 실시 예에 따라 파일 A-1(10)의 파일을 수신하여 동적 분석을 수행한 경우, 파일 A-1 (10) 실행 시에 서버 (가) (110) 를 접속하는 것을 확인하였다고 가정한다.

[0176] 실시 예는 악성 코드에 대한 여러 가지 분석 정보를 저장하는 데이터 베이스로부터 파일 A-1 (10)과 유사한 파일 A-2 (20)의 저장된 분석 정보를 얻을 수 있다. 파일 A-2 (20)의 분석 정보로부터 동일한 서버인 서버 (가) (110) 가 파일 A-1 (10) 과 파일 A-2 (20)을 활용한다는 것을 파악할 수 있고 이러한 정보로부터 서버 (가) (110) 는 동일 공격 기법 또는 동일 서버를 이용하는 해커임을 추정할 수 잇다.

[0177] 실시 예에 따라 이미 분석된 파일인 파일 A-2 (20) 이 서버 (가) (110)뿐만 아니라 서버 (나) (120) 도 접속하는 경우 파일 A-2 (20) 의 연관 관계로서 서버 (나) (120)의 정보를 저장할 수 있다.

[0178] 만약 파일 A-1(10) 과 파일 A-2(20) 과는 전혀 다른 파일이지만 파일 B-1 (30) 의 분석 정보가 서버 (나) (120)를 접속한 기록을 저장했다면 파일 형식이 다르지만 서버 (가) (110) 와 서버 (나) (120) 는 동일한 공격자 그룹 또는 동일한 기법을 이용하는 공격자 그룹일 수 있다.

[0179] 따라서, 이와 같이 파일과 관련된 여러 가지 분석 정보에 대해 연관관계를 분석하면 악성 행위를 유발하는 공격자, 공격 기법 등에 대한 그룹핑 정보를 얻을 수 있고, 이러한 연관관계 분석 정보는 공격자나 공격자 그룹을 식별하는데 활용될 수 있다.

[0181] 이하에서는 사이버 위협 예측 정보를 설명하는 예를 개시한다.

[0182] 파일의 식별 정보와 얻은 분석 정보들 중 적어도 하나 이상의 정보를 이용하거나 취합한 데이터 세트에 기초하여 사이버 위협 예측 정보를 생성할 수 있다

[0184] 도 8은 실시 예에 따라 사이버 위협 정보의 예측 정보 생성하는 일 예를 개시한다. 도면을 참조하여 사이버 위협 정보의 예측 정보를 생성하는 예를 설명하면 다음과 같다.

[0185] 분석 정보에 대한 데이터 세트가 확보되면 그 데이터 세트를 기초로 추후에 발생할 공격 행위와 관련된 예측 정보 생성이 가능하다.

[0186] 위와 같이 추출된 분석 정보에 따른 데이터 세트를 인공 지능 기반의 학습 데이터 세트로 가공하고, 가공된 학습 데이터 세트를 기초로 인공 지능 분석을 수행하면 공격 행위와 관련된 여러 가지 예측 정보 생성이 가능하다.

[0187] 이렇게 생성된 예측 정보의 데이터 세트는 다시 새로운 학습 데이터 세트로 반복적으로 생성 또는 가공할 수 있다.

[0188] 이 도면의 실시 예는 위의 분석 정보의 데이터 세트를 인공 지능 학습을 통해 악성 코드 제작자의 예측 정보(S3110), 악성 코드 공격 방법의 예측 정보(S3120), 악성 코드 공격 그룹의 예측 정보(S3130), 악성 코드 유사도 예측 정보(S3140), 악성 코드 확산도 예측 정보(S3150) 등을 생성하는 예를 개시한다.

[0189] 여기서 예측 정보의 순서는 일 예로서 예측 정보 획득의 순서의 변경이 가능하다. 예를 들면 악성 코드 유사도 예측 정보(S3140)와 악성 코드 확산도 예측 정보(S3150)의 순서는 변경될 수 있으며 나머지 예측 정보의 생성도 반드시 예시된 순서에 따를 필요가 없다.

[0190] 또한 예시한 유사도 예측 정보 이외에 사이버 위협 정보와 관련된 추가적인 예측 정보 생성도 가능하다.

[0191] 이렇게 생성한 악성 코드의 예측 정보는 자체 위험도를 예측하는 위험도 예측 정보와 공격자, 공격 그룹, 유사도, 확산도 등을 각각 예측하는 예측 정보 또는 그 예측 정보를 종합적으로 표시하는 악성 코드의 종합 예측 정보로 나뉘어 데이터베이스에 저장될 수 있다.

[0192] 위와 같은 사이버 위협 정보의 분석 정보와 예측 정보를 이용하면 입력된 파일과 관련된 악성 코드의 종류를 식별하고 이에 대한 위험도를 결정할 수 있다.

[0193] 또한 입력된 파일과 관련된 악성 코드의 기록을 포함한 프로파일링 정보를 생성하여 저장될 수 있는데, 저장된 악성 코드와 관련된 분석 정보, 예측 정보, 위험도 또는 프로파일링 정보는 사용자가 이를 쉽게 조회할 수 있도록 추가로 가공될 수 있다.

[0194] 사용자에게 사이버 위협 정보를 제공하는 일 예를 개시하면 다음과 같다.

[0195] 특정 파일을 기준으로 여러 가지 연관 관계 분석 정보가 발생될 수 있어서 사이버 위협 침해 정보(IoC)를 매우 많은 데이터 통신량이 필요할 수 있다. 실시 예는 사이버 보안의 위협에 신속하게 대처하기 위해서는 이러한 정보를 빠른 시간 내에 공유, 저장, 조회, 및 업데이트할 수 있다.

[0196] 위와 같은 분석 정보들에 기초하여 실시 예는 보안 이벤트가 발생하면 발생된 보안 이벤트에 관련된 사이버 위협 침해 정보(IoC)를 암호화 소켓 통신을 통해 사이버 위협 침해 정보(IoC) 저장 서버나 다른 사용자 단말기들에 P2P 소켓 통신을 이용해 조회를 요청할 수 있다. 그리고 사이버 위협 침해 정보(IoC) 저장 서버나 다른 사용자 단말기들 중 사이버 위협 침해 정보(IoC)를 빨리 수신하는 정보를 사이버 위협 침해 정보(IoC)로 이용할 수 있다.

[0197] 또 다른 예로서, 사이버 위협 정보를 제공하는 또 다른 예로서 사용자가 사용하는 단말에서 상기와 같이 분석된 악성 코드에 대한 정보를 조회할 경우 조회된 정보를 다음과 같이 제공할 수 있다.

[0198] 예를 들어 사용자가 사용하는 단말이 파일의 해쉬 값을 산출한 경우, 산출된 해쉬 값에 대해 텍스트 형식으로 악성 코드 여부의 조회하는 질의를 서버로 전송할 수 있다. 해쉬 값과 질의를 수신한 서버가 위와 같이 악성 코드 정보가 저장된 데이터 베이스에 상기 해쉬 값을 전달하고 이에 대한 조회 결과를 수신한다. 조회 결과를 수신한 서버는 그 결과를 상기 해쉬 값에 대응되는 텍스트 값으로 사용자 단말에 다시 리턴할 수 있다.

[0199] 저장된 악성 코드에 대한 정보를 기반으로 사용자의 요청에 따라 사이버 위협 정보를 제공하는 다른 예를 도면을 참조하여 설명하면 다음과 같다.

[0201] 도 9는 실시 예에 따라 사이버 위협 정보를 제공하기 위한 악성 코드 질의들의 예를 개시한다.

[0202] 사이버 보안 위협 정보 처리에 대한 실시 예는 위와 같이 산출한 분석 정보와 예측 정보를 기초로 식별한 악성 코드를 여러 가지 메타 정보와 함께 저장할 수 있다.

[0203] 위에서 설명한 바와 같이 사용자는 악성 코드 정보가 저장된 데이터 베이스에 예시한 바와 같은 조회를 요청할 수 있다.

[0204] Query (A)를 참고하면, 사용자는 실시 예에 따른 사이버 위협 정보가 저장된 데이터베이스에 Query (A)와 같이 악성 코드와 관련된 기간, 특정 악성 코드의 수량, 탐지명, 파일 타입, 유포지, 코드사인 및 파일 크기 등의 카테고리로 악성 코드를 질의할 수 있다.

[0205] 그러면 사이버 위협 정보가 저장된 데이터 베이스는 서버를 통해 Query 에 대응되는 사이버 위협 정보나 악성 코드 정보를 리턴한다.

[0206] 다른 예로 사용자는 이 도면의 Query (B)에서 예시한 바와 같이 악성 코드와 관련된 특정일, 특정 악성 코드의 수량, 파일 타입, 유포지 여부, 자식 프로세스의 생성 여부 등을 질의할 수 있다.

[0207] Query (C)에서 예시하는 바와 같이 사용자는 악성 코드와 관련된 기간, 특정 악성 코드의 수량, 파일 타입, 유포지 정보, 파일 명 정보, 악성 코드 수행에 따른 공격 행위, 파일 크기에 정보를 이용하여 악성 코드에 대한 정보를 질의할 수 있다.

[0208] Query (D)의 예는 악성 코드와 관련된 기간, 특정 악성 코드의 수량, 파일 타입, 유포지 주소 및 악성 코드의 통계 정보를 이용하여 악성 코드에 대한 정보를 질의할 수 있다.

[0209] 설명한 바와 같이 사이버 보안 위협 정보 처리 방법의 실시 예는 분석 정보, 예측 정보는 사용자의 조회 문의에 대해 대응되는 악성 코드 정보를 제공하기 위해 악성 코드에 위와 같은 조건에 맞는 정보를 데이터베이스에 함께 저장한다.

[0210] 따라서, 서버는 해당 질의 조건과 일치하는 악성 코드에 대한 정보를 데이터베이스부터 얻어 사용자에게 전송할 수 있다.

[0211] 예시한 바와 같이 사용자는 파일의 여러 가지 메타 정보를 이용해 악성 코드 정보를 조회할 수 있다. 사용자는 보호해야 하는 정보나 시스템이 악성 코드에 의해 피해나 위협이 될 수 있는 정보를 미리 얻을 수 있다.

[0213] 도 10은 사이버 보안 위협 정보 처리 장치의 일 실시 예를 개시한 도면이다. 이 도면의 실시 예는 사이버 보안 위협 정보 처리 장치를 개념적으로 예시하는데 이 도면을 참조하여 사이버 보안 위협 정보 처리 장치의 실시 예를 설명하면 다음과 같다.

[0214] 개시하는 사이버 보안 위협 정보 처리 장치는 물리장치(2000)인 데이터베이스 및 서버(2100) 및 데이터베이스(2200)와 상기 물리장치(2000) 상에서 구동되는 응용 프로그래밍 인터페이스 Application Programming Interface, API) 포함하는 플랫폼 (10000)을 포함한다. 이하에서 플랫폼(10000)은 사이버 위협 인텔리전스 플랫폼(cyber threat intelligence platform; CTIP) 또는 간략하게 인텔리전스 플랫폼(10000)으로 호칭한다.

[0215] 서버(2100)는 중앙연산장치(central processing unit, CPU) 나 프로세서와 같은 연산장치를 포함하고 데이터베이스(2200)에 데이터를 저장하거나 읽을 수 있다.

[0216] 서버(2100)는 입력되는 보안 관련 데이터를 연산 및 처리하며 파일을 실행하여 여러 가지 보안 이벤트를 발생시키고 관련된 데이터를 처리하도록 한다. 그리고 서버(2100)는 여러 가지 사이버 보안 관련 데이터의 입출력을 제어하고 인텔리전스 플랫폼(10000)에서 처리된 데이터를 데이터베이스(2200)에 저장할 수 있다.

[0217] 서버(2100)는 데이터 입력을 위한 네트워크 장치나 네트워크의 보안 장치를 포함할 수 있다. 서버(2100)의 중앙처리장치, 프로세서 또는 연산장치는 이하의 도면에서 예시하는 프레임워크나 해당 프레임 워크 내의 모듈을 수행할 수 있다.

[0218] 실시 예에 따른 인텔리전스 플랫폼(10000)은 사이버 위협 정보의 처리를 위한 응용 프로그래밍 인터페이스(API)를 제공한다. 예를 들어 인텔리전스 플랫폼(10000)은, 네트워크와 연결된 네트워크 보안 장치나 악성 행위를 스캔 및 감지하는 사이버 악성 행위 방지 프로그래밍 소프트웨어로부터 파일이나 데이터를 입력받을 수 있다.

[0219] 예를 들어 실시 예에 따른 인텔리전스 플랫폼(10000)은 보안 이벤트를 제공하는 SIEM (Security Information and Event Management) API, 실행 환경에 대한 데이터를 제공하는 EDR (Environmental Data Retrieval) API, 네트워크 트래픽을 정의된 보안 정책에 따라 모니터하고 제어하는 파이어월(firewall) API 등의 기능을 제공할 수 있다. 또한 인텔리전스 플랫폼(10000)은 내부와 외부 네트워크 사이에 방화벽과 유사한 역할을 수행하는 IPS (Intrusion Prevention Systems)의 API의 역할도 제공할 수 있다.

[0220] 실시 예에 따른 인텔리전스 플랫폼(10000)의 응용 프로그래밍 인터페이스(API)(1100)는 사이버 보안의 공격 행위를 수행하는 악성 코드를 포함하는 파일들을 여러 클라이언트 기기들 (1010, 1020, 1030) 로부터 수신할 수 있다.

[0221] 실시 예에 따른 인텔리전스 플랫폼(10000)은 전처리부(미도시), 분석 프레임 워크(1210)와 예측 프레임 워크(1220) 및 AI 엔진 (1230) 및 후처리부(미도시)을 포함할 수 있다.

[0222] 인텔리전스 플랫폼(10000)의 전처리부는 클라이언트 기기들(1010, 1020, 1030)로부터 수신된 여러 가지 파일들에 대한 사이버 위협 정보를 분석할 수 있도록 전처리를 수행한다.

[0223] 예를 들면 전처리부는 수신된 파일을 처리하여 그 파일로부터 파일의 출처 정보, 파일을 얻은 수집 정보, 파일의 사용자 정보 등을 포함한 여러 가지 메타 정보를 얻을 수 있다. 예를 들어 파일이 URL (uniform resource locator)을 포함하거나 또는 전자메일에 포함된 경우 파일에 대한 수집 정보를 얻을 수 있다. 사용자 정보는 파일의 생성, 업로드 또는 최종 저장한 사용자 정보 등을 포함할 수 있다. 전처리 과정에서 파일의 메타 정보로서 IP(internet protocol) 정보, 이에 기반한 국가 정보, API(Application Programming Interface) key 정보 등을 얻을 수 있다.

[0224] 인텔리전스 플랫폼(10000)의 전처리부(미도시)는 입력된 파일의 해쉬(Hash) 값을 추출할 수 있다. 해쉬 값이 이미 사이버 보안 위협 정보 처리 장치에 알려진 것이라면 이를 기반으로 파일의 종류를 식별할 수 있다.

[0225] 만약 이미 알려진 파일이 아니라면 운영하는 C-TAS(Cyber Threats Analysis System), CTA(Cyber Threat Alliance)의 운영시스템, VitusTotal 등의 사이버 위협 정보의 레퍼런스 인터넷 사이트에 해쉬 값과 파일 정보를 조회하여 파일 종류 식별을 위한 분석 정보를 얻을 수 있다.

[0226] 설명한 바와 같이 입력된 파일의 해쉬 값은 MD5 (Message-Digest algorithm 5), SHA1 (Secure Hash Algorithm 1), SHA 256 등의 해쉬 함수의 해쉬 값이 될 수 있다.

[0227] 분석 프레임 워크(1210)는 입력된 파일로부터 악성 코드에 대한 분석 정보를 생성할 수 있다.

[0228] 분석 프레임 워크(1210)는 정적 분석 모듈(1211), 동적분석 모듈(1213), 심층분석 모듈(1215) 및 연관관계분석 모듈(1217) 등 여러 가지 분석 방식에 따른 분석 모듈을 포함할 수 있다.

[0229] 정적 분석 모듈(1211)은 입력된 파일과 관련된 악성 행위의 분석 정보는 파일 자체에 대한 악성 코드 관련 정보를 분석할 수 있다.

[0230] 동적분석 모듈(1213)은 입력된 파일로부터 얻은 여러 가지 정보들을 기반으로 여러 행위를 수행함으로써 악성 코드 관련 정보를 분석할 수 있다.

[0231] 심층분석 모듈(1215)은 입력된 파일과 관련된 실행 가능한 파일을 가공한 정보를 이용하거나 실행 가능한 파일과 관련된 메모리 분석을 수행하여 악성 코드 관련 정보를 분석할 수 있다. 심층분석 모듈(1215)은 악성 행위를 정확하게 식별할 수 있도록 인공 지능 분석을 포함할 수 있다.

[0232] 연관관계분석 모듈(1217)은 입력된 파일과 관련하여 이미 저장된 분석 정보들이나 또는 생성된 분석 정보들을 서로 연관시켜 공격 행위나 공격자에 대한 연관 관계를 추정할 수 있는 연관관계 분석 정보를 포함할 수 있다.

[0233] 분석 프레임 워크(1210)는 정적 분석 모듈(1211), 동적분석 모듈(1213), 심층분석 모듈(1215) 및 연관관계분석 모듈(1217)로부터 분석된 정보들을 악성 코드의 특성과 행위에 대한 분석 결과들을 서로 결합하고, 결합된 최종 정보를 사용자에게 제공할 수 있다.

[0234] 예를 들어 분석 프레임 워크(1210)는 하나의 파일에 대한 정적 분석 정보, 동적 분석 정보, 심층 분석 정보, 연

관관계 분석 정보 등은 정확한 공격 기법과 공격자 식별을 위해 통합 분석할 수 있다. 분석 프레임 워크(1210)는 분석 정보들 사이에 중복된 부분을 제거하고 분석 정보들 사이에 공통의 정보는 정확도를 높이는데 사용한다.

[0235] 분석 프레임 워크(1210)는 제공하는 정보를 표준화할 수 있는데, 예를 들면 여러 분석과 경로를 통해 수집된 사이버 위협 침해 정보(indicator of compromise, IoC)들을 노멀라이징(normalizing)하거나 인리치먼트(enrichment) 작업한다. 그리고 최종 표준화된 악성 코드 또는 악성 행위에 대한 분석 정보를 생성할 수 있다.

[0236] 분석 프레임 워크(1210)의 정적 분석 모듈(1211), 동적분석 모듈(1213), 심층분석 모듈(1215) 및 연관관계분석 모듈(1217)은 분석되는 데이터의 정확성을 높이기 위해 분석 대상 데이터에 인공지능 분석에 따른 머신 러닝이나 딥 러닝 기법을 수행할 수 있다.

[0237] AI 엔진(1230)은 분석 프레임 워크(1210)의 분석 정보 생성을 위해 인공지능 분석 알고리즘을 수행할 수 있다.

[0238] 이러한 정보는 데이터 베이스(2200)에 저장될 수 있고 서버(2100)는 사용자나 클라이언트 요청에 따라 데이터베이스(2200)에 저장된 악성 코드 또는 악성 행위에 대한 분석 정보를 사이버 위협 인텔리전스 정보로 제공할 수 있다.

[0239] 예측 프레임 워크(1220)은 제1예측정보생성모듈(1221), 제2예측정보생성모듈 (1223) 등 예측 정보에 따라 다수의 예측정보생성모듈들을 포함할 수 있다. 예측 프레임 워크(1220)은 분석 정확도를 높이기 위해 위의 분석된 여러 가지 정보의 데이터 세트를 이용하여 악성 행위의 발생 여부, 공격 기법, 공격자 그룹 등에 대한 예측 정보를 생성할 수 있다.

[0240] 예측 프레임 워크(1220)는 분석 프레임 워크(1210)가 분석한 분석 정보에 대한 데이터 세트를 기반으로 AI 엔진(1230)을 이용하여 인공지능 분석 알고리즘을 수행하여 입력된 파일과 관련된 악성 행위에 대한 예측 정보를 생성할 수 있다.

[0241] AI 엔진(1230)은 분석 정보에 대한 데이터 세트에 대해 인공 지능 기반의 머신 러닝으로 학습하여 추가적인 분석 정보를 생성하고, 추가 생성된 분석 정보는 다시 새로운 학습 데이터로서 인공 지능의 입력 데이터로 이용될 수 있다.

[0242] 예측 프레임 워크(1220)가 생성하는 예측 정보는 악성 코드 제작자 정보, 악성 코드 공격 방법 정보, 악성 코드 공격 그룹 예측, 악성 코드 유사도 예측 정보, 및 악성 코드 확산도 예측 정보 등을 포함할 수 있다.

[0243] 위와 같이 여러 가지 악성 코드나 공격 행위 등에 관련된 예측 정보를 생성한 예측 프레임 워크(1220)는 생성한 예측 정보들을 데이터베이스(2200)에 저장할 수 있다. 그리고 사용자의 요청에 따라 또는 공격 징후에 따라 생성한 예측정보를 사용자에게 제공할 수 있다.

[0244] 서버(2100)는 설명한 바와 같이 데이터 베이스(2200)에 저장된 분석 정보 또는 예측 정보에 대한 후처리 후 상기 입력된 파일과 관련된 사이버 위협 정보를 제공할 수 있다.

[0245] 서버(2100)의 프로세서는 생성된 분석 정보 또는 예측 정보에 기초하여 악성 코드 종류 및 악성 코드의 위험도를 결정하는 작업을 수행한다.

[0246] 서버(2100)의 프로세서는 악성 코드에 대한 프로파일링 정보를 생성할 수 있다. 데이터베이스(2200)는 파일 분석을 통해 파일에 대한 자체 분석을 수행한 결과나 추가 및 예측 분석을 수행한 결과를 저장할 수 있다.

[0247] 서버(2100)에 의해 사용자에게 제공되는 사이버 위협 정보는, 기술된 전처리가 수행된 정보, 생성되거나 식별된 분석 정보, 생성된 예측 정보 또는 이 정보들의 취합 정보나 이 정보들을 기반으로 결정된 정보를 포함할 수 있다.

[0248] 제공되는 사이버 위협 정보에는 입력된 파일과 관련하여 데이터 베이스에 저장된 분석 정보를 이용하거나 위에서 분석되거나 예측된 정보가 포함될 수 있다.

[0249] 실시 예에 따르면 사용자가 입력된 파일에 대한 악성 행위뿐만 아니라 이미 저장된 파일이나 악성 행위에 대해 사이버 위협 정보를 조회할 경우 이에 대한 정보를 제공할 수 있다.

[0250] 이러한 통합 분석 정보는 해당 파일에 대응하여 서버나 데이터 베이스에 표준화된 포맷으로 저장될 수 있다. 이러한 통합 분석 정보는 표준화된 포맷으로 저장되어 사이버 위협 정보를 검색 또는 조회하는데 사용될 수 있다.

[0252] 도 11은 개시하는 실시 예에 따라 분석 프레임 워크 중 정적 분석 모듈의 기능을 상세히 설명하기 위한 일 예를 나타낸다. 이 도면을 참조하여 정적 분석 모듈의 수행 과정을 예시하면 다음과 같다.

[0253] 개시한 바와 같이 인텔리전스 플랫폼(100)의 분석 프레임 워크(15000)는 정적분석 모듈(15100)을 포함할 수 있다.

[0254] 정적분석 모듈(15100)은 파일 자체를 분석할 수 있는데, 파일 또는 파일의 메타 정보 등에 기초하여 코딩 기반의 취약 항목 존재 여부, 인터페이스 또는 함수의 호출 구조 문제, 또는 파일의 바이너리 구조 등 파일과 관련하여 악성 행위에 연관될 수 있는 정보를 얻을 수 있다.

[0255] 정적분석 모듈(15100)은 파일구조분석 모듈(15101), 파일패턴분석 모듈(15103), 파일제작정보분석 모듈(15105), 파일환경분석 모듈(15107), 및 파일관련분석 모듈(15109)를 포함할 수 있다.

[0256] 정적분석 모듈(15100) 중 파일구조분석 모듈(15101)는 파일이 실행되지 않는 환경에서 식별된 파일의 기본적인 구조 정보를 분석할 수 있다.

[0257] 파일구조분석 모듈(15101)는 예를 들어 파일의 종류가 ELF(Executable and Linkable Format), PE(Portable Executable), APK(Android Application Package) 등에 파일 종류가 다르더라도 파일의 위 파일 구조나 그 구조로부터 추출할 수 있는 정보를 획득하거나 분석한다.

[0258] 파일패턴분석 모듈(15103)은 파일의 패턴 분석을 수행할 수 있는데, 식별된 파일에 어떤 조치를 취하지 않고 파일 자체를 오픈하여 추출할 수 있는 여러 스트링(string) 등을 확인하여 파일의 패턴을 얻을 수 있다.

[0259] 파일제작정보분석 모듈(15105)은 입력된 파일이 제작과 관련된 정보를 얻고 분석할 수 있다. 파일제작정보분석 모듈(15105)은 파일이 가지고 있는 고유 정보나 메타 정보, 예를 들면 파일 제작자 정보, 실행 파일인 경우 코드사이닝(codesigning) 정보 등을 얻을 수 있다.

[0260] 파일환경분석 모듈(15107)은 입력된 파일의 환경 정보를 분석할 수 있다. 파일환경분석 모듈(15107)은 대상 파일이 갖추어야 할 시스템 환경적 구성 요소 정보 등에 정보를 얻을 수 있다.

[0261] 파일관련분석 모듈(15109)은 그리고 입력된 파일과 관련된 여러 가지 기타 메타 정보들을 분석할 수 있다.

[0262] 정적분석 모듈(15100)은 입력된 파일의 수행 없이 개시한 바와 같이 얻고 분석된 파일 자체의 정적 정보를 JSON(JavaScript Object Notation)과 같은 데이터 포맷으로 변환하여 데이터베이스(2200)에 저장할 수 있다.

[0263] 서버(2100)는 데이터베이스(2200)에 저장된 파일에 대한 정적 분석 정보를 사용자에 제공할 수 있다.

[0264] 분석프레임워크(15000)의 정적분석 모듈(15100)은 입력된 파일의 해쉬(Hash) 값과, 데이터베이스(2200)에 악성코드에 대해 이미 저장된 해쉬 값을 비교하여 상기 입력된 파일이 악성코드 여부를 분석할 수 있다. 그리고 입력 파일의 악성 코드에 대해 분석된 정보는 데이터베이스(2200)에 저장할 수 있다.

[0265] 분석프레임워크(15000)의 정적분석 모듈(15100)은 입력 파일이 모바일 데이터 인 경우 입력된 파일로부터 모바일 악성 의심 코드의 코드 정보를 추출할 수 있다. 악성 의심 코드의 코드 정보는 해쉬(Hash) 정보, 코드 크기 정보, 파일 헤더 정보, 코드 내에 포함되어 있는 식별 가능한 문자열 정보 및 동작 플랫폼 정보 등을 포함할 수 있다.

[0266] 분석프레임워크(15000)의 정적분석 모듈(15100)은 분석한 분석정보를 기반으로 파일 내에 악성 코드가 있는지 탐지할 수 있다. 그리고 탐지된 악성 코드와 관련된 정적 분석 정보를 데이터베이스(2200)에 저장할 수 있다.

[0268] 도 12는 개시하는 실시 예에 따라 분석 프레임 워크 중 동적분석 모듈의 기능을 상세히 설명하기 위한 일 예를 나타낸다. 이 도면을 참조하여 동적분석 모듈의 수행 과정을 예시하면 다음과 같다.

[0269] 예시한 인텔리전스 플랫폼(10000)의 분석 프레임 워크(15000)는 동적분석 모듈(15200)을 포함할 수 있다. 동적분석 모듈(15200)은 전처리된 파일 정보 또는 정적 분석 정보 중 적어도 하나에 기반하여 식별된 파일의 실행 환경에서 실행된 결과 데이터에 따른 동적 분석 정보를 획득할 수 있다.

[0270] 동적분석 모듈(15200)은 파일이 실행 중인 환경에서 다양한 입출력 데이터를 분석하거나 또는 파일 실행 시 실행 환경과 상호작용의 변화를 분석하여 취약하거나 위험한 이상현상을 탐지할 수 있다. 동적분석 모듈(15200)은

가상화 환경 등을 생성하고 생성된 가상화 환경에서 파일을 직접적으로 실행하여 이상 여부를 분석할 수 있다.

[0271] 분석 프레임 워크(15000)의 동적분석 모듈(15200)은 환경준비 모듈(15201), 파일실행 모듈(15203), 행위수집 모듈(15205), 분석결과취합 모듈(15207), 및 분석환경복구 모듈(15209)를 포함할 수 있다.

[0272] 환경준비 모듈(15201)은 입력 파일과 관련된 실행 파일을 실행하기 위한 동적 분석 환경을 생성하고 준비한다. 환경준비 모듈(15201)은 실행 파일의 타입을 식별한 경우 각각의 파일의 타입에 따라 어떤 실행 환경이 필요한지 식별할 수 있다. 예를 들면 파일에 따라 윈도우 운영체제, 리눅스 운영체제, 모바일 기기 운영체제에서 실행되는 파일인지 식별할 수 있다. 환경준비 모듈(15201)은 실행 파일을 실행하기 위해 식별된 환경을 준비할 수 있다.

[0273] 파일실행 모듈(15203)은 환경준비 모듈(15201)이 준비한 분석 환경에서 실행 파일이 악성 코드 포함하고 있는지 여부를 판별하기 위해 파일을 실행한다.

[0274] 행위수집 모듈(15205)은 동적 분석 정보를 획득하기 위해 실행 환경에서 실행된 파일의 실행 중에 시스템에서 발생하는 이벤트를 수집할 수 있다. 예를 들어 행위수집 모듈(15205)은 파일 자체, 프로세스, 메모리, 레지스트리, 네트워크의 시스템에 대한 이벤트 또는 각 시스템의 설정을 변경시키는 이벤트를 수집할 수 있다.

[0275] 분석결과취합 모듈(15207)은 행위수집 모듈(15205)이 수집한 이벤트들을 개별적으로 또는 취합하여 분석한다.

[0276] 분석환경복구 모듈(15209)은 수집된 결과를 취합한 후 동적 분석을 위한 환경을 다시 복구한다.

[0277] 동적분석 모듈(15200)은 이와 같이 획득된 결과를 해당 파일 또는 파일의 악성 코드에 대응된 동적 분석 정보로 데이터베이스(2200)에 저장할 수 있다.

[0278] 동적분석 모듈(15200)이 위 실시 예에 따라 동적 분석 정보를 수집하고 분석하는 예를 간략하게 개시하면 다음과 같다.

[0280] 동적 분석의 일 실시 예로서, 동적분석 모듈(15200)은 입력된 파일이 모바일 기기 운영 체제에서 동작하는 파일로 식별된 경우, 파일을 모바일 단말 또는 모바일 단말 환경과 동일하게 구성된 에뮬레이터나 가상화 환경을 생성할 수 있다. 그리고 동적분석 모듈(15200)은 생성한 에뮬레이터나 가상화 환경에서 상기 파일을 직접 실행할 수 있다. 동적분석 모듈(15200)은 파일 내에 모바일 악성 의심 코드가 실행된 후에 단말에 발생하는 모든 변화, 즉 행위 정보를 추출하고 기록할 수 있다. 행위 정보는 단말의 운영체제(OS) 환경이 다른 경우라도 프로세스, 파일, 메모리 및 네트워크 정보 등의 이벤트 정보를 포함할 수 있다.

[0281] 동적 분석의 다른 실시 예로서 동적분석 모듈(15200)은 전처리 과정에서 입력된 파일의 해쉬(Hash) 값을 추출되지 않고 사용자 단말에서 추출된 경우라도 단말에서 추출된 파일의 해쉬 값을 인텔리전스 플랫폼(10000)을 통해 수신할 수 있다.

[0282] 데이터베이스(2200)에 해당 파일의 해쉬 값이 이미 저장되지 않는 경우 동적분석 모듈(15200)은 수신된 파일을 가상 또는 실제의 운영체제에서 실행시키고, 실행 시에 발생되는 행위를 실시간으로 수집하고 수집된 동적분석 정보를 데이터베이스(2200)에 이미 저장된 정보와 비교할 수 있다.

[0283] 상기 비교 결과 이미 정의된 위험도를 초과하는 경우 입력된 파일이 악성 코드를 포함하고 있다고 판단할 수 있고, 동적분석 모듈(15200)은 악성 코드에 대응되는 파일의 해쉬 값을 데이터베이스(2200)에 저장할 수 있다. 저장된 악성 해쉬 값은 추후 정적 분석 등에 이용할 수 있다.

[0284] 악성 코드는 외부의 서버와 통신하며 추가적인 명령을 발생시키고 파일을 수신하도록 할 수 있다.

[0285] 그런데 동적 분석을 수행할 수 있는 플랫폼과 서버가 중지된 경우는 이러한 동적 분석에 매우 오랜 시간이 소요될 수 있고 해당 행위가 사전 차단된 경우에도 동적 분석을 수행할 수 없는 경우가 있다.

[0286] 실시 예에 따른 동적분석 모듈(15200)은 네트워크 행위를 분석할 경우, 악성 코드가 사용하는 명령 제어 서버(C&C 서버), 추가적인 악성 코드를 다운로드하기 위한 다운로드 서버 또는 악성 코드들끼리 정보를 주고 받거나 해커와 정보를 주고 받는 커뮤니케이션 패킷 등의 정보를 추출하여 분석할 수 있다.

[0287] 여기서 개시하는 동적분석 모듈(15200)은 서버(2100)가 동작 중지된 경우에도 동적 분석을 수행하도록 할 수 있다.

[0288] 예를 들어 네트워크 접속 유도 장치(미도시)가 악성 코드에 감염된 클라이언트 단말과 인텔리전스 플랫폼(10000) 또는 서버(2100)에 사이에서 단말의 접속 요청을 처리하도록 하여 동적 분석을 진행하도록 할 수도 있다.

[0289] 네트워크 접속 유도 장치(미도시)는 단말로부터 접속 요청을 수신하고 이를 악성 코드 행위를 유발시키는 C&C 서버로 전달하도록 할 수 있다.

[0290] 그리고, 만약 상기 네트워크 접속 유도 장치가 일정 시간 내에 C&C 서버로부터 응답 패킷을 수신하지 못하면, 상기 네트워크 접속 유도 장치는 별도의 가상의 응답 패킷과 접속 요청을 함께 상기 단말에 전송하도록 한다.

[0291] 이후에 상기 단말로부터 수신된 악성 코드 분석에 관련된 데이터를 추출할 수 있다.

[0292] 가상의 응답 패킷을 이용하는 예는 가상의 응답 패킷 TCP 세션을 생성하기 위한 패킷 형식이면 충분하다. 악성 코드가 사용하는 일반적인 TCP (Transmission Control Protocol) 프로토콜은 TCP 세션만 생성하도록 상기 클라이언트 단말이 전송하는 데이터 패킷을 생성할 수 있다. 그리고 상기 데이터 패킷으로부터 악성 코드의 동적 분석에 필요한 중요 정보들을 추출할 수 있다. 이와 같이 하면 관리 서버가 동작하지 않더라도 네트워크 접속 유도 장치의 동작을 이용하여 동적 분석을 수행할 수 있다.

[0294] 도 13은 개시하는 실시 예에 따라 분석 프레임 워크 중 심층분석 모듈의 기능을 상세히 설명하기 위한 일 예를 나타낸다. 이 도면을 참조하여 심층분석 모듈의 수행 과정을 예시하면 다음과 같다.

[0295] 인텔리전스 플랫폼(10000)의 분석 프레임 워크(15000)는 심층분석 모듈(15300)을 포함할 수 있다. 심층분석 모듈(15300)은 수신된 파일 포함하는 실행 가능한 파일 디스어셈블링하여 기계 언어 레벨에서 분석하여 악성 행위를 유발하는 공격 기법이나 공격자를 식별할 수 있다.

[0296] 심층분석 모듈(15300)은 기술한 정적 분석이나 동적 분석의 기반으로 심층 분석 정보를 얻을 수도 있고, 분석자의 해석 기준에 따라 실행 가능한 파일을 악성 행위를 유발하는 파일을 이용하여 분석할 수도 있다.

[0297] 심층분석 모듈(15300)은 파일 자체의 분석 정보나 또는 파일을 여러 번 가공한 정보를 포함할 수 있고 이미 저장된 정보를 기반으로 심층 분석 정보를 생성할 수 있다

[0298] 심층분석 모듈(15300)은 또한, 심층 분석은 디스어셈블링(disassembling) 모듈(15301), 기계언어코드추출 모듈(15303), 공격행위(TTP)식별 모듈(15305), 공격자식별 모듈(15307), 테인트분석(taint analysis)모듈(15309)를 포함할 수 있다.

[0299] 분석 프레임 워크(15000)는 심층분석 모듈(15300)은 AI 엔진(1230)을 이용하여 인공 지능 기반의 머신 러닝 알고리즘을 수행하고, 그 결과로 심층분석 정보를 얻을 수 있다.

[0300] 디스어셈블링(disassembling) 모듈(15301)은 입력된 파일이 실행 가능한 파일을 포함할 경우 실행 가능한 파일을 디스어셈블(disassemble)한다.

[0301] 실행 가능한 파일이 디스어셈블링(disassembling)되면 오브젝트 코드 형식의 특정 형식, 예를 들면 어셈블러 언어 형식의 코드로 변환된다.

[0302] 기계언어코드추출모듈(15303)은 일정 형식을 가진 OP-CODE (operation code)와 ASM-CODE를 포함하는 디스어셈블드 코드를 추출할 수 있다. 일정 형식을 가진 OP-CODE (operation code)는 악성 코드와 관련된 OP-CODE 부분을 의미하는 것으로 추출된 OP-CODE를 포함하는 디스어셈블드 코드는 악성 코드 또는 악성 행위와 관련된 부분을 지칭한다.

[0303] 기계언어코드추출모듈(15303)은 디스어셈블드 코드를 일정 형식의 데이터 포맷을 변환할 수 있다. 일정 형식의 데이터 포맷의 변환 예시는 아래에서 개시한다.

[0304] 실행 가능한 파일의 디스어셈블드 코드를 사이버 보안 전문가 집단들이 공통적으로 인정하는 공격 행위 세부 요소들로 매칭하도록 하여 그 공격행위를 식별할 수 있다.

[0305] 공격행위(TTP)식별 모듈(15305)은 추출된 디스어셈블드 코드나 일정 형식으로 변환된 포맷의 데이터를 기반으로 공격행위, 공격기법 및 공격 프로세스를 식별할 수 있다.

[0306] 공격행위(TTP)식별 모듈(15305)은 실행 가능한 파일의 디스어셈블드 코드를 기반의 퍼지 해쉬 값을 사이버 보안

전문가 집단들이 공통적으로 인정하는 공격 행위 세부 요소들로 매칭하도록 하여 그 공격행위를 식별할 수 있다.

[0307] 공격행위(TTP)식별 모듈(15305)은 이미 추출된 디스어셈블드 코드들과 공격행위(TTP) 별 매칭 관계를 저장한 데이터베이스(2200) 또는 외부 레퍼런스 데이터베이스에 기반하여 공격행위(TTP)를 식별하도록 할 수 있다. 공격행위(TTP)식별 모듈(15305)은 AI 엔진(1230)의 머신 러닝을 이용하여 추출된 디스어셈블드 코드들의 CTPH 알고리즘 등의 퍼지 해쉬 값과 공격행위(TTP) 별 매칭 유사도를 고속으로 수행하여 공격행위 또는 공격기법을 분류할 수 있다..

[0308] 디스어셈블드 코드 내 OP-CODE는 수행될 연산을 특정하는 기계 언어 명령어의 일부인데, 사이버 보안 상 공격기법 또는 공격행위(Terrorist Tactics, Techniques, and Procedures, 이하 TTP)를 유발하는 OP-CODE 를 포함하는 디스어셈블드 코드는 해당 공격 행위 별로 매우 유사한 값이나 포맷을 가질 수 있다. 따라서, 이러한 OP-CODE와 ASM-CODE의 조합인 디스어셈블드 코드를 분석하면 특정 타입의 공격 행위를 구별할 수 있다.

[0309] 예를 들면 공격행위(TTP)식별 모듈(15305)는 실행 가능한 파일로부터 추출된 디스어셈블드 코드를 퍼지 해쉬(Fuzzy Hashing) 방식 또는 CTPH (context triggered piecewise hashes) 방식의 해쉬 값으로 변환할 수 있다.

[0310] 공격행위(TTP)식별 모듈(15305)과 함께 수행되는 AI 엔진(1230)의 머신 러닝 알고리즘으로 Perceptron, Logistic Regression, Support Vector Machines, Multilayer Perceptron 등의 알고리즘이 사용될 수 있다. 또한 AI 엔진(1230)으로 앙상블 머신 러닝 알고리즘이나 자연어 처리 알고리즘도 사용할 수 있다. 이에 대한 예는 이하에서 상세히 개시한다.

[0311] 보안 전문가 집단의 공격 행위를 저장한 데이터 베이스의 일 예로서 MITRE ATT&CK은 실제 보안 공격 기법이나 행위에 대한 데이터 베이스인데 공격행위(TTP)식별 모듈(15305)은 추출한 OP-CODE을 포함하는 디스어셈블드 코드이 변환된 해쉬 값을 MITRE ATT&CK의 데이터베이스 상의 일정한 데이터 세트 형식 또는 식별자로 식별할 수 있도록 한다.

[0312] MITRE ATT&CK는 해커 또는 악성 코드의 공격 기법에 대한 취약 요소들을 CVE 코드(Common Vulnerabilities and Exposures Code)의 매트릭스로 표현한다.

[0313] 실시 예는 디스어셈블드 코드를 분석함으로써 여러 가지 공격 행위들 중 특정 공격 행위를 식별하되, 식별된 타입의 공격 행위가 전문가 단체들이 인정하는 공격 행위의 요소들로 매칭되도록 함으로써 공격 행위 식별이 전문적이면서 공통으로 인식되는 요소들로 표현되도록 할 수 있다.

[0314] 설명한 바와 같이 OP-CODE는 특정 행위를 유발시키는 기계 언어 명령어이므로, 동일한 공격 행위를 유발하는 파일의 디스어셈블드 코드는 매우 유사할 수 있다. 그러나 공격 행위와 이를 유발하는 파일의 디스어셈블드 코드가 정확하게 매칭되는 것은 아니므로 코드 상 일부 차이가 있을 수 있다.

[0315] 공격행위(TTP)식별 모듈(15305)은 추출한 디스어셈블드 코드를 일정 형식으로 변환한 코드에 대해 AI 엔진(1230)의 머신 러닝 수행하도록 한다. 따라서, 동일한 악성 행위를 유발시키는 파일들의 OP-CODE들이 완전히 동일하지 않더라도 공격행위(TTP)식별 모듈(15305)은 머신 러닝과 추출된 OP-CODE 기반의 퍼지 해쉬 값과 그에 대응하는 공격 요소를 매칭하여 공격 행위 등을 식별할 수 있다.

[0316] 공격행위(TTP)식별 모듈(15305)은 디스어셈블드 코드들의 유사도를 AI 알고리즘을 이용하여 MITRE ATT&CK과 같은 공격 기법에 매칭하여 최종적으로 해당 파일이 악성 코드임을 탐지할 수 있다.

[0317] 이에 대한 구체적인 예는 후술 한다.

[0318] 공격자식별 모듈(15307)은 추출된 디스어셈블드 코드와 인공 지능 기반의 머신 러닝 결과를 이용해 유사 공격행위를 유발하는 공격자도 식별하는 단계를 포함할 수도 있다. 마찬가지로 공격자 식별에 대한 구체적인 예는 후술한다

[0319] 테인트분석(taint analysis)모듈(15309)은 파일이 없는(fileless) 악성 코드의 경우도 특정 시점에서 시스템의 메모리 분석을 통해 공격 행위가 있는지 여부에 대해 판단할 수 있다.

[0320] 심층분석 모듈(15300)은 해당 파일이나 그 파일로부터 식별된 악성 코드에 대응되는 심층 분석 정보를 데이터베이스(2200)에 저장할 수 있다.

[0322] 도 14은 개시하는 실시 예에 따라 분석 프레임 워크 중 연관관계분석 모듈의 기능을 상세히 설명하기 위한 일 예를 나타낸다. 이 도면을 참조하여 연관관계분석 모듈의 수행 과정을 예시하면 다음과 같다.

[0323] 인텔리전스 플랫폼(10000)의 분석 프레임 워크(15000)는 연관관계분석 모듈(15400)을 포함할 수 있다. 연관관계분석 모듈(15400)은 분석 프레임 워크(15000)가 분석하는 여러 가지 분석 정보들을, 사이버 위협 침해 정보(IoC)에 기반하여 공격자 또는 공격 기법 사이에 연관관계로 표현되도록 연관관계 분석 정보를 생성한다.

[0324] 연관관계분석 모듈(15400)은 분석 정보와 공격 행위 사이의 IP 정보의 연관관계를 분석하는 제 1 연관관계분석 모듈(15401), 이메일에 포함되거나 웹사이트 등에 포함된 호스트네임의 연관관계를 분석하는 제 2 연관관계분석 모듈 (15403), URL의 연관관계를 분석하는 제 3 연관관계분석 모듈 (15405), 파일의 코드사인(codesign)의 연관관계를 분석하는 제 4 연관관계분석 모듈 (15407), 공격 기법들 사이의 연관관계를 분석하는 제 5 연관관계분석 모듈 (15407) 등을 포함할 수 있다.

[0325] 이 도면에 표시된 모듈들은 예시에 불과하며, 이 도면에 표시되지 않더라도 연관관계분석 모듈(15400)은 공격 기법과 공격자를 판단하기 위해 분석된 정보들 사이에 여러 가지 연관관계들을 분석할 수 있는 모듈들을 포함할 수 있다. 예를 들면 연관관계분석 모듈(15400)은 생성한 연관관계 정보들을 취합하거나 통합하는 통합 분석 모듈을 포함할 수도 있다.

[0326] 연관관계분석 모듈(15400)은 정확하게 공격기법 또는 공격자를 추론하는데 사용되는 연관관계 분석 정보를 생성할 수 있다.

[0327] 연관관계분석 모듈(15400)은 수신되는 파일이나 악성 코드에 대해 지속적이고 누적적으로 분석 정보들을 저장하고 추후 새로운 파일이나 악성 코드가 분석될 때마다 관련된 연관관계 분석 정보를 다시 업데이트하여 데이터베이스(2220)에 저장한다.

[0328] 연관관계분석 모듈(15400)은 위에서 분석한 여러 가지 분석 정보(정적분석정보, 동적분석정보, 심층분석정보 등)를 기반으로 사이버 위협 침해 정보를 얻을 수 있다.

[0329] 연관관계분석 모듈(15400)은 사이버 위협 침해 정보(IoC)를 이용해 공격 행위나 공격자를 식별할 수 있는 여러 가지 연관관계 정보를 얻을 수 있으며 이와 같이 분석된 연관관계 분석 정보를 데이터베이스(2200)에 저장할 수 있다.

[0330] 위에서 개시한 바와 같이 인텔리전스 플랫폼(10000)의 분석 프레임 워크(15000)는 분석된 정보들을 종합하여 중복 제거, 표준화, 인리치먼트 과정을 통해 표준화된 정보를 데이터베이스(2220)에 저장할 수 있다.

[0331] 인텔리전스 플랫폼(10000)는 정적 분석 정보, 동적분석 정보, 심층분석 정보, 연관관계분석 정보들을 사이버 위협 정보를 갱신 또는 재생산하기 위해 표준화된 포맷으로 데이터베이스(2200)에 저장할 수 있다.

[0332] 여기서 인텔리전스 플랫폼(10000)는 각 분석 정보들의 중복되거나 공통된 분석 정보의 중복된 부분을 제거하고, 부족한 부분의 데이터의 인리치먼트(enrichment) 작업 등을 수행할 수 있다.

[0333] 인텔리전스 플랫폼(10000)는 후 처리를 통해 표준화된 정보를 사이버 공격들의 방지하기 위해 고안된 표준인 STIX 이나 TAXII 등의 포맷으로 저장할 수 있다.

[0334] 서버 (2100)는 사용자의 조회 질의에 따라 또는 서비스 정책에 따라 분석 프레임 워크(15000)가 생성한 분석 정보 등을 표준화된 사이버 위협 정보로 제공할 수 있다. 사이버 위협 정보로 제공 방법에 대해서도 이하에서 상세히 후술한다.

[0335] 이러한 사이버 위협 정보는 사용자의 요청이나 서비스에 따라 제공할 수도 있다.

[0337] 도 15는 개시하는 실시 예에 따라 예측 프레임 워크의 예측정보생성 모듈의 기능을 상세히 설명하기 위한 일 예를 나타낸다. 이 도면을 참조하여 예측 프레임 워크의 수행 과정을 예시하면 다음과 같다.

[0338] 예시한 인텔리전스 플랫폼(10000)의 예측 프레임 워크(17000)는 예측정보생성모듈(17100)을 포함할 수 있다. 예측정보생성모듈(17100)은 생성하는 예측정보에 따라 다수의 정보예측모듈들을 포함할 수 있다. 이 예에서는 예측정보생성모듈(17100)이 제1정보예측모듈(1711), 제2정보예측모듈(1713), 제3정보예측모듈(1715), 제4정보예측모듈(1717), 및 제5정보예측모듈(1719)을 포함하는 예를 나타낸다.

[0339] 예측 프레임 워크(17000)는 이전에 예시한 분석 프레임 워크(미도시)가 생성한 분석정보들을 이용할 수 있다. 예측 프레임 워크(17000)는 여러 가지 분석 정보들에 따른 데이터 세트를 인공 지능 기반의 학습 데이터 세트로 가공하고, AI 엔진(1230)은 가공된 학습 데이터 세트를 기초로 인공 지능 분석을 수행할 수 있다.

[0340] 예측 프레임 워크(17000)과 AI 엔진(1230)의 수행을 통해 공격 행위와 관련된 여러 가지 예측 정보 생성할 수 있다.

[0341] 이 예에서는 제1정보예측모듈(1711)는 인공 지능 학습을 통해 악성 코드 제작자의 예측 정보를 생성할 수 있다. 제2정보예측모듈(1713)는 악성 코드 공격 방법의 예측 정보를 생성하고 제3정보예측모듈(1715)는 악성 코드 공격 그룹의 예측 정보를 생성할 수 있다. 그리고 제4정보예측모듈(1717)는 악성 코드 유사도 예측 정보를 생성하고, 제5정보예측모듈(1719)는 악성 코드 확산도 예측 정보를 생성하는 예를 나타낸다.

[0342] 구체적인 예측 정보의 생성의 예는 이하에서 후술한다.

[0343] 예측 프레임 워크(17000)는 생성한 예측 정보를 데이터베이스(2200)에 저장할 수 있다.

[0344] 예를 들면 예측 프레임 워크(17000)는 특정 악성 코드에 대해 그 악성코드의 위험 자체를 예측한 악성코드 위험도 예측 정보를 생성하여 데이터베이스(2200)에 저장할 수 있다.

[0345] 그리고 예측 프레임 워크(17000)는 특정 악성 코드에 대해 예측한 제작자, 공격방법, 공격 그룹, 유사도, 확산도의 예측 정보를 데이터베이스(2200)에 저장할 수 있다.

[0346] 개시한 바와 같이 인텔리전스 플랫폼(1000)은 분석 정보 또는 예측 정보에 기초하여 악성 코드 종류 및 악성 코드의 위험도를 생성할 수 있다. 그리고 인텔리전스 플랫폼(10000)은 악성 코드에 대한 프로파일링 정보를 생성할 수 있다.

[0347] 인텔리전스 플랫폼(10000)은 파일 분석을 통해 파일에 대한 자체 분석을 수행한 결과나 추가 및 예측 분석을 수행한 결과를 데이터베이스(2200)에 저장할 수 있다.

[0348] 인텔리전스 플랫폼(10000)이 제공하는 사이버 위협 정보는, 위의 전처리를 수행한 정보, 생성한 분석 정보, 생성한 예측 정보 또는 이 정보들의 취합 정보나 이 정보들을 기반으로 추가 후 처리된 정보를 포함할 수 있다.

[0349] 따라서 제공되는 사이버 위협 정보에는 입력된 파일과 관련하여 통합 분석

[0350] 이러한 예시한 인텔리전스 플랫폼(10000)에 의해 제공되는 통합 분석 정보는, 입력된 파일에 대응하여 서버(2100)에 의해 데이터베이스(2200)에 표준화된 포맷으로 저장될 수 있다. 이러한 통합 분석 정보는 표준화된 포맷으로 저장되어 사이버 위협 정보를 검색 또는 조회에 사용될 수 있다.

[0352] 이하에서는 각 처리 단계 또는 모듈에 따른 상세한 실시 예들을 개시한다.

[0353] 도 16은 개시하는 실시 예에 따라 정적 분석을 수행하는 일 예를 나타낸다. 도면을 참조하여 실시 예에 따른 정적 분석 방법의 일 예를 설명하며 다음과 같다.

[0354] 설명한 바와 같이 정적 분석을 수행하기 이전에 전처리 단계나 정적 분석의 초기 단계에서 파일의 종류를 식별할 수 있다. 이 도면은 파일의 종류로서 편의상 ELF, EXE, ARK 파일이 식별된 경우를 예시하지만 실시예의 적용은 이에 국한되지 않는다.

[0355] 악성코드의 정적 분석 또는 탐지는 위와 같은 파일 자체가 가지고 있는 성격과 기존에 확인된 패턴 데이터베이스와 비교 하는 과정을 기반으로 동작할 수 있다.

[0356] 정적 정보 추출기는 입력된 파일의 구조를 파싱하여 구조 정보를 얻을 수 있다.

[0357] 파싱된 파일의 구조 상 패턴(pattern)은 데이터베이스(DB)(2200)에 이미 저장된 악성 코드의 패턴과 비교될 수 있다.

[0358] 파싱된 파일의 구조 특징과 패턴은 상기 파싱된 파일의 메타 정보가 될 수 있다.

[0359] 위에 개시된 예에서는 표시하지 않았으나 개시하는 실시예의 정적 분석에서도 머신 러닝 엔진이 사용될 수 있다. 데이터베이스(2200)는 이미 저장된 악성 코드의 학습된 특징들을 포함하는 데이터 세트를 저장할 수 있다.

[0360] AI 엔진은 위와 같이 파상된 파일로부터 얻은 메타 정보를 머신 러닝을 통해 학습하고, 데이터베이스(2200)에 이미 저장된 데이터 세트를 비교하여 악성코드 여부를 판단할 수 있다.

[0361] 정적 분석을 통해 악성 코드로 분석된 파일은 파일의 구조적 특징은 악성 코드와 관련된 데이터 세트로 다시 저장될 수 있다.

[0363] 도 17은 개시하는 실시 예에 따라 동적 분석을 수행하는 일 예를 나타낸다. 도면을 참조하여 실시 예에 따른 동적 분석 방법의 일 예를 설명하며 다음과 같다.

[0364] 설명한 바와 같이 동적 분석을 수행하기 이전에 전처리 단계나 동적 분석의 초기 단계에서 파일의 종류를 식별할 수 있다. 마찬가지로 이 예시에서 파일의 종류로서 편의상 ELF, EXE, ARK 파일이 식별된 경우를 예시한다.

[0365] 전처리를 통해 동적 분석 대상이 되는 파일 종류를 식별할 수 있다. 식별된 파일은 각 파일의 종류와 타입에 따라 가상 환경에서 실행될 수 있다.

[0366] 예를 들어 식별된 파일이 ELF 파일인 경우 대기 큐(Que)를 거쳐 리눅스 가상 환경(Virtual Machine, VM)의 운영체제에서 실행될 수 있다.

[0367] ELF 파일이 실행될 경우 발생하는 이벤트는 행위 로그(log)에 기록될 수 있다.

[0368] 이와 같이 각각의 식별 파일의 종류 별로 윈도우, 리눅스, 모바일 운영체제 시스템을 가상으로 구축한 후 가상 시스템의 실행 이벤트를 기록한다.

[0369] 그리고 데이터베이스(2200)에 이미 저장된 악성 코드의 실행 이벤트들과 기록한 실행 이벤트들을 비교할 수 있다. 위에서 예시하지 않았으나 동적 분석의 경우에도 머신 러닝을 통해 기록한 실행 이벤트들을 학습하고, 학습된 데이터가 이미 저장된 악성 코드의 실행 이벤트들과 유사한지 판단할 수 있다.

[0370] 동적 분석의 경우 파일에 따라 가상 환경을 구축해야 하고 이에 따라 분석 및 탐지 시스템의 규모가 커질 수 있다.

[0372] 도 18은 개시하는 실시 예에 따라 심층 분석을 수행하는 일 예를 나타낸다. 도면을 참조하여 실시 예에 따른 심층 분석 방법의 일 예를 설명하며 다음과 같다.

[0373] 설명한 바와 같이 심층 분석을 수행하기 이전에 전처리 단계나 심층 분석의 초기 단계에서 파일의 종류를 식별할 수 있다. 개시된 예는 식별된 파일이 ELF, EXE, ARK 의 실행 가능한 바이너리 파일을 예시한다.

[0374] 실행 가능한 바이너리 파일을 디스어셈블(disassemble)을 수행하면 CPU(Central Processing Unit)의 명령어 집합 중 함수들의 구조를 분석할 수 있다.

[0375] 심층 분석은 동적 분석과 다르게 바이너리 파일을 디스어셈블하여 추출된 코드를 기반으로 동작하기 때문에 상대적으로 시스템 규모가 간단하게 분석이 가능하다. 그리고 심층 분석은 별도의 엔진 없이 추출된 코드들을 정규화 하는 과정을 통해 만들어진 데이터를 기초로 인공지능 분석을 수행할 수 있다.

[0376] 이 도면에서 디스어셈블드 코드는 OP-CODE와 ASM-CODE의 결합으로 표현된다.

[0377] 실시 예는 OP-CODE 와 ASM-CODE를 기반으로 두 가지 코드를 조합하고, 조합된 코드 중 의미가 있는 코드 블록(Code Block)을 추출할 수 있다.

[0378] OP-CODE 와 ASM-CODE을 포함하는 디스어셈블된 코드의 코드 블록(Code Block)은 일정한 형식을 변환하여 해당 코드가 악성 코드와 관련되었는지, 어떤 악성 코드이지 또는 어떤 공격자가 개발했는지를 식별할 수 있다.

[0379] 이를 판단하기 위한 코드 블록(Code Block)의 데이터 변환 방식을 여러 가지 과정이 있다. 디스어셈블된 코드의 데이터 변환 과정은 데이터의 처리 속도와 정확도에 따라 선택적으로 적용될 수 있으나 이 도면에서는 정규화 과정과 벡터화 과정만을 표기하였다.

[0380] OP-CODE와 ASM-CODE의 결합 코드의 추출된 코드 블록(Code Block)을 정규화 과정과 벡터화 과정을 수행할 수 있다.

[0381] 즉 바이너리 코드의 OP-CODE 와 ASM-CODE 조합으로 코드 블록(Code Block)을 추출하고 이 코드 블록(Code

Block)의 특징 정보를 벡터화시킨 후 다양한 특징 정보를 통해 학습된 데이터와 비교하여 공격 행위 등을 식별하도록 한다.

[0382] 동일한 실행 파일이라도 이와 같이 추출된 코드 블록(Code Block)이 모두 다를 수 있기 때문에 실시 예는 추출된 코드 블록(Code Block)를 악성 코드로 판단하고 분류하는 방식으로 머신 러닝 또는 인공 지능(AI) 방식을 이용할 수 있다.

[0383] 그리고 실시 예는 정규화 및 벡터화 과정이 수행된 최종 데이터를 인공 지능을 통해 학습시킨다. 학습된 데이터는 데이터베이스(2200)에 이미 저장된 공격 기법(TTP)과 공격자 또는 공격 그룹의 데이터와 비교되어 악성 코드 여부 등의 정보를 얻을 수 있다.

[0384] 실시 예는 악성 코드의 핵심 부분인 구성 요소를 MITRE ATT&CK 모델을 기반으로 분류하고 구분할 수 있다.

[0385] 이에 대한 구체적인 실시 예는 이하에서 더욱 상세하게 개시된다.

[0387] 도 19는 개시하는 실시 예에 따라 바이너리 코드에서 추출된 코드들로 공격 기법을 매칭하는 일 예를 나타낸다. 여기에서는 공격 기법을 매칭하는 일 예로 표준화된 모델을 사용하는 예를 개시한다.

[0388] 여기서 표준화된 모델로 MITRE ATT&CK® Framework를 예시한다.

[0389] 예를 들어 사이버 보안 상 “악성 행위” 라고 하는 것은 분석가에 따라 해석 방식이 다르고 각자가 가지고 있는 식견에 따라서 다르게 해석하는 경우가 많았다.

[0390] 국제적으로 시스템 상에서 발생하는 “악성 행위”를 표준화 하고 모두가 같은 해석을 할 수 있도록 전문가들 사이에 많은 노력을 수행되고 있다. 미국 연방정부의 지원을 받으며 국가안보관련 업무를 수행하던 비영리 연구개발 단체인 MITRE(https://attack.mitre.org)에서 “악성 행위” 에 대한 정의를 연구하였고 그에 따라 ATT&CK® Framework 이라는 것을 만들고 공표하였다. 이 프레임 워크는 사이버 위협 또는 악성코드에 대해 모두가 같은 “악성 행위”를 정의 할 수 있도록 정의하였다.

[0391] MITRE ATT&CK® Framework (이하, MITRE ATT&CK®)는 공격자들의 최신 공격 기술 정보를 정리한 것으로서 Adversarial Tactics, Techniques, and Common Knowledge의 약어이다. MITRE ATT&CK® 은, 실제 사이버 공격 사례를 관찰한 후 공격자의 악의적 행위(Adversary behaviors)에 대해서 공격 방법(Tactics)과 기술(Techniques)을 분석하여 다양한 공격 그룹들의 공격 기법들에 대한 정보들을 분류하고 목록화한 표준적인 데이터이다.

[0392] MITRE ATT&CK® 은 전통적인 사이버 킬체인의 개념과는 약간 관점을 달리하여 지능화된 공격의 탐지를 향상시키기 위해 위협적인 전술과 기술을 체계화(패턴화)한 것이다. 원래 ATT&CK는 MITRE에서 윈도우 운영체제를 사용하는 기업 환경에 사용되는 해킹 공격에 대해서 방법(Tactics), 기술(Techniques), 절차(Procedures) 등 TTP를 문서화하는 것으로 시작되었다. 그 이후 ATT&CK은 공격자로부터 발생한 일관된 공격 행동 패턴에 대한 분석을 기반으로 TTP 정보를 매핑하여 공격자의 행위를 식별해 줄 수 있는 프레임워크로 발전하였다.

[0393] 개시하는 실시 예에서 언급하는 악성 행위는, MITRE ATT&CK® 와 같은 표준화된 모델에 기반하여 악성 코드를 공격 기법에 매칭하여 표현할 수 있는데 표준화된 모델이 어떤 것이든 악성 코드를 요소 별로 식별하고 분류하여 공격 식별자에 매칭할 수 있다.

[0394] 이 도면의 예 어떻게 악성 코드의 악성 행위와 MITRE ATT&CK 모델 기반으로 공격 기법이 매칭되는지를 개념적으로 나타낸다.

[0395] 실행 파일 EXE는 파일 실행 시에 수행되는 여러 가지 함수들(Function A, B, C, D, E,…, N,…, Z)을 포함할 수 있다. 그 함수들 중 적어도 하나의 함수를 포함하는 함수 그룹은 하나의 공격 방법(tactic)을 수행할 수 있다.

[0396] 이 도면의 예에서 함수 A, B, C는 공격 방법(tactic) A에 대응되고, 함수 D, B, F는 공격 방법(tactic) B에 대응되는 예를 개시한다. 유사하게 함수 Z, R, C는 공격 방법(tactic) C에 대응되고, 함수 K 및 F는 공격 방법(tactic) D에 대응된다.

[0397] 실시 예는 각 공격 방법(tactic)에 대응되는 함수들의 집합과 특정 디스어셈블드 코드 의 부분을 대응시킬 수 있다. 데이터베이스는 이미 인공 지능으로 학습된 디스어셈블드 코드들에 대응될 수 있는 의 공격 방법

(Tactics), 기술(Techniques), 절차(Procedures) (TTP)의 공격 식별자 (T-ID)를 저장하고 있다.

[0398] 공격 방법(Tactics), 기술(Techniques), 절차(Procedures) (TTP)의 공격 식별자 (T-ID)는 표준화된 모델을 따르며 여기 도면의 예시는 사이버 위협 정보의 표준화된 모델로 MITRE ATT&CK®를 예시하였다.

[0399] 따라서, 실시 예는 바이너리 파일에서 디스어셈블드 코드로부터 추출한 결과 데이터를 표준화된 공격 식별자로 매칭시킬 수 있다. 공격 식별자를 매칭하는 보다 구체적인 방식은 아래에서 개시한다.

[0401] 도 20은 개시하는 실시 예에 따라 OP-CODE를 포함하는 코드 세트와 공격 기법을 매칭하는 일 예를 나타낸다.

[0402] 대부분의 인공지능 엔진은 악성 코드의 다양한 특징 정보를 바탕으로 학습된 데이터 셋(data set)을 이용해 악성 코드를 판별한다. 그러면 악성 코드의 악성 여부는 판단이 되지만 이러한 방식은 악성 코드가 왜 악성 코드인지에 대한 설명을 하기 힘들었다. 그러나 예시한 바와 같이 표준화된 공격 방법(TTP)의 식별자로 대응시키면 해당 악성 코드가 어떤 위협 요소가 있는지 식별이 가능하다. 따라서, 실시 예는 보안 관리자에게 사이버 위협 정보를 정확하게 전달하도록 하고, 보안 관리자가 사이버 위협 정보를 체계적이고 장기적으로 관리할 수 있도록 할 수 있다.

[0403] 실시 예는 디스어셈블드 코드를 기반으로 공격 방법(TTP)을 식별하기 위한 인공 지능 학습용 데이터 셋을 생성할 때 단순히 공격 방법(TTP)의 식별자 또는 라벨링 만을 구분할 뿐만 아니라 공격 방법(TTP)을 어떻게 구현했는지에 대한 특징을 중요한 요소로 반영할 수 있다.

[0404] 동일한 공격 방법(TTP)을 구현하는 악성 코드라도 개발자에 따라 동일한 코드로 생성하는 것은 불가능하다. 즉, 공격 방법(TTP)의 기술은 인간 구술 언어 형태로 되어 있으나, 개발자에 따라 이를 구현 방식과 코드 작성 방법이 동일하지 않다.

[0405] 이러한 코드 작성의 차이는 개발자의 역량이나 프로그램 로직을 구현하는 방식이나 습관에 따르는데 이러한 차이는 바이너리 코드 또는 이를 디스어셈블된 OP-CODE 와 ASM-CODE의 차이로 나타낸다.

[0406] 그래서 단순히 결과적인 공격 방법(TTP)의 타입에 따라 공격 식별자를 부여하거나 대응시키면 악성 코드를 생성하는 공격자 또는 공격자 그룹까지 정확하게 식별하기 힘들다.

[0407] 반대로 디스어셈블된 OP-CODE 와 ASM-CODE의 특성을 중요한 변수로 반영시켜서 모델링을 수행하면 특정 악성코드나 공격 도구를 개발한 개발자 혹은 자동으로 생성하는 도구 자체까지도 식별이 가능하다.

[0408] 개시하는 실시 예는 디스어셈블된 OP-CODE 와 ASM-CODE 결합 코드의 고유한 특성에 따라 현대의 사이버 전에서 굉장히 중요한 위협 인텔리전스를 생성하도록 할 수 있다. 즉, 이러한 고유 특성에 기초하면 실시 예는 공격 코드 또는 악성 코드를 어떻게 동작을 하는지, 이것을 누가 어떤 의도로 개발했는지에 대한 내용을 함께 식별할 수 있다.

[0409] 그리고 추후에 해당 공격자가 계속해서 공격하는 특징 정보를 바탕으로 취약한 시스템을 보완할 수 있고 사이버 보안 위협에 대한 능동적이고 선제적인 대응이 가능하도록 할 수 있다.

[0410] 이러한 개념 상에서 실시 예는 단순히 OP-CODE 기반으로 공격 결과에 따른 공격 기법을 식별하는 방식과 성능에서 전혀 다른 결과를 제공한다.

[0411] 실시 예는 공격 방법(TTP)를 구현하기 위해 사용된 코딩 기법을 정확하게 식별하여 분류하기 위해 디스어셈블된 OP-CODE 와 ASM-CODE을 조합된 특징에 기초한 디스어셈블드 코드의 데이터 세트를 생성할 수 있다. 이렇게 생성된 데이터 세트로부터 고유한 특성을 식별하도록 모델링하면 공격 방법(TTP)뿐만 아니라 개발자의 특징 정보, 즉 개발자 (또는 자동화된 제작 도구)가 누구인지까지 식별이 가능하다.

[0412] 이 도면은 위에서 설명한 방식으로 모델링된 OP-CODE 데이터 세트를 공격 식별자에 매칭하는 예를 나타낸다.

[0413] 이 예에서 제 1 OP-CODE 세트(OP-CODE set #1)는 공격 기법 식별자 T1011에 매칭되고, 제 2 OP-CODE 세트(OP-CODE set #2)는 공격 기법 식별자 T2013에 매칭됨을 나타낸다. 그리고 제 3 OP-CODE 세트(OP-CODE set #3)는 공격 기법 식별자 T1488에 매칭할 수 있고, 제 N번째 OP-CODE 세트(OP-CODE set #N)는 임의의 공격 기법 식별자 T1XXX에 매칭됨을 나타낸다. 표준화된 모델인 MITRE ATT&CK®은 공격 기법의 식별자를 요소 별로 매트릭스 형식으로 표현하지만, 실시 예는 공격 기법의 식별자 이외에 공격자 또는 공격 도구를 추가로 식별할 수 있다.

[0414] 이 도면은 편의 상 OP-CODE 데이터 세트로 표시하였으나 OP-CODE 와 ASM-CODE을 포함하는 디스어셈블드 코드의

데이터 세트로 공격 기법을 식별하면 OP-CODE 데이터 세트만으로 공격 기법을 식별하는 것보다 더욱 세분화된 공격 기법을 식별할 수 있다.

[0415] 실시 예에 따라 디스어셈블드 코드의 데이터 세트의 조합을 분석하면 공격 기법 식별자 뿐만 아니라 공격자 또는 공격 그룹의 식별할 수도 있다.

[0416] 따라서, 실시 예는 기존의 기술보다 인텔리전스 정보 획득 차원에서 고도화된 기술을 제공할 수 있을 뿐만 아니라 종래의 보안 영역에서 해결하지 못한 문제를 해결할 수 있다.

[0417] 위와 같이 복잡한 환경에서 정확한 인텔리전스 정보를 확보하기 위해 빠른 데이터처리와 알고리즘이 요구된다. 이하에서는 이와 관련된 추가적인 실시 예와 그에 따른 성능에 대해 개시하도록 한다.

[0419] 도 21은 개시하는 실시 예에 따라 사이버 위협 정보를 처리하는 흐름을 예시한 도면이다.

[0420] 이 도면에서 식별된 파일이 ELF, EXE, ARK 의 실행 가능한 바이너리 파일인 경우를 예로 하여 설명한다. 이 단계의 처리 과정은 위에서 개시한 심층 분석과 관련된다.

[0421] 먼저 제 1 단계로서 OP-CODE 코드를 포함하는 디스어셈블드 코드를 추출하는 과정의 일 상세한 예를 설명하면 다음과 같다.

[0422] 소스 코드를 컴파일(complie)하면 실행 파일이 생성된다.

[0423] 원시 소스 코드는 실행 가능한 각 운영체제(OS) 환경에서 컴파일러에 의해 기계의 처리에 적합한 형태의 새로운 데이터로 생성된다. 새롭게 구성된 바이너리 데이터는 사람이 읽기에는 적합하지 않은 형태로 되어 있어 실행 파일 형태로 만들어진 파일을 인간이 해석해서 그 내부 로직을 파악하는 것은 불가능하다.

[0424] 그러나 보안 시스템의 취약점 분석과 다양한 목적을 위해서 그 역과정을 수행하여 기계어의 해석이나 분석을 수행하는데 설명한 바와 같이 디스어셈블 과정이라고 한다. 디스어셈블 과정은 특정 운영체제의 중앙처리장치(CPU)와 처리 비트 수(32비트, 64비트 등) 에 맞춰서 수행될 수 있다.

[0425] 예시한 ELF, EXE, ARK 의 실행 파일을 각각 디스어셈블을 수행하면 디스어셈블된 어셈블리 코드를 획득할 수 있다.

[0426] 디스어셈블된 코드는 OP-CODE 와 ASM-CODE가 조합된 코드를 포함할 수 있다.

[0427] 실시 예는 디스어셈블 도구를 기반으로 실행 파일을 분석하여 실행 파일로부터 OP-CODE 와 ASM-CODE을 추출할 수 있다.

[0428] 개시하는 실시 예는 추출된 OP-CODE 와 ASM-CODE을 그대로 이용하지 않고 각 함수 별로 재구성하여 OP-CODE 배열을 다시 구성한다. OP-CODE 배열을 재정리할 경우 원본 바이너리 데이터도 함께 포함하여 데이터의 해석을 충분히 수행할 수 있도록 데이터를 재구성할 수 있다. 이러한 재배열를 통해 OP-CODE 와 ASM-CODE의 새로운 조합은 공격 기법뿐만 아니라 공격자를 식별할 수 있는 기초 데이터를 제공한다.

[0429] 제 2 단계로 어셈블리 데이터를 처리하는 과정(ASM)을 상세히 설명하면 다음과 같다.

[0430] 어셈블리 데이터 처리 과정은 OP-CODE와 필요한 ASM-CODE 만을 분리한 후 인간 또는 컴퓨터가 읽기 좋은 형태로 재구성된 데이터를 기반으로 유사도를 분석하고 정보를 추출하는 과정이다.

[0431] 이 단계에서 디스어셈블된 어셈블리 데이터는 일정한 데이터 형식으로 변환될 수 있다.

[0432] 이러한 데이터 형식의 변환은 데이터 처리 속도를 높이고 데이터의 정확한 분석을 위해 아래 기술된 변환 방식들은 모두 적용될 필요없이 선택적으로 적용될 수 있다.

[0433] 재배열된 OP-CODE 와 ASM-CODE의 조합의 어셈블리 데이터로부터 여러 가지 함수를 추출할 수 있다.

[0434] 하나의 실행 파일을 디스어셈블하면 프로그램 크기에 따라 다르지만 평균적으로 약, 7,000~12,000개 정도 되는 함수를 포함할 수 있다. 이 함수들은 프로그래머가 필요에 따라 구현한 함수도 있으며 운영체제에서 기본적으로 제공하는 함수들도 있다.

[0435] 실제 ASM-CODE를 분석하면 약 87%~91% 정도의 함수가 운영체제에서 기본적으로 제공하는 함수(OS supported)이고 프로그래머가 프로그램 로직을 위해서 실제 구현한 ASM-CODE는 약 10% 정도이다. 운영체제에서 제공한 함수

는 함수 명과 함께 운영체제 설치 시에 기본적으로 설치되는 각종 DLL, SO 파일 등에 포함되는 함수들(Default function)이다. 이러한 운영체제 제공 함수들은 이미 분석하여 저장하여 분석 대상 데이터로부터 필터링할 수 있다. 이렇게 분석해야 할 코드만 분리하면 이후 처리 속도와 성능을 높일 수 있다.

[0436] 실시 예는 프로그램의 기능적 분석을 정확하게 수행하기 위해서 OP-CODE를 함수 단위로 분리해서 처리할 수 있다. 실시 예는 모든 의미적 분석의 최소 단위를 어셈블리 코드에 포함된 함수를 기반하여 수행할 수 있다.

[0437] 분석 성능과 처리 속도를 높이기 위해 실시 예는 의미가 정확하지 않은 연산자 수준의 함수들은 필터링하고 정보량이 임계 치 보다 작은 함수들 도 분석 대상에서 제거할 수 있다. 함수들의 필터링의 여부와 정도는 실시 예에 따라 다르게 설정할 수 있다.

[0438] 실시 예는 함수에 따라 정리된 OP-CODE 로부터 디스어셈블러가 출력 시 제공하는 주석 데이터를 제거할 수 있다. 그리고 실시 예는 디스어셈블된 코드를 재배열할 수 있다.

[0439] 예를 들면, 디스어셈블러가 출력하는 디스어셈블된 코드는 [ASM-CODE, OP-CODE, 파라미터]의 순서를 가질 수 있다.

[0440] 실시 예는 어셈블리 데이터로부터 파라미터 데이터를 제거하고 위 순서의 디스어셈블된 코드를 [OP-CODE, ASM-CODE] 순서로 재정리 또는 재구성할 수 있다. 이렇게 재정된 디스어셈블된 코드는 정규화 또는 벡터화하여 처리하기 용이하다. 그리고 처리 속도를 현격하게 높일 수 있다.

[0441] 특히 [OP-CODE, ASM-CODE] 의 조합을 가지는 디스어셈블된 코드 중 ASM-CODE 부분은 데이터의 길이가 달라 서로 비교하기 용이하지 않다. 따라서 해당 어셈블리 데이터의 고유성을 확인하기 위해서 데이터를 특정 크기의 데이터 포맷으로 정규화시킬 수 있다. 예를 들면 실시 예는 [OP-CODE, ASM-CODE] 조합의 디스어셈블된 코드의 고유성을 확인하기 위해서 데이터 부분을 정규화하기 용이한 특정 길이의 데이터 세트, 예를 들면 CRC(cyclic redundancy check) 데이터로 변환시킬 수 있다.

[0442] 일 예로서 [OP-CODE, ASM-CODE] 조합의 디스어셈블된 코드에서 OP-CODE 부분은 제 1 길이의 CRC 데이터로, ASM-CODE 부분은 제 2 길이의 CRC 데이터로 각각 변환하는 것도 가능하다.

[0443] OP-CODE와 ASM-CODE 변환된 정규화 데이터는 각각 해당 변환 이전의 각각 코드의 고유성을 유지할 수 있도록 한다. 고유성을 가지고 변환된 정규화 데이터의 유사도 판단 속도를 빠르게 하기 위해 상기 정규화된 데이터를 벡터화(Vectorization)를 수행할 수 있다.

[0444] 설명한 바와 같이 데이터 변환 과정으로서 정규화 또는 벡터화 과정은 데이터 처리 속도를 높이고 데이터의 정확한 분석을 선택적으로 적용될 수도 있다.

[0445] 정규화 과정과 벡터화 과정의 상세한 예는 다시 아래에서 상세히 개시한다.

[0446] 제 3단계로서 디스어셈블드 코드를 분석하는 데이터의 분석과정을 상세히 설명하면 다음과 같다.

[0447] 이 과정에서도 데이터 처리 속도를 높이고 데이터의 정확한 분석을 위해 여러 가지 데이터 형식의 변환이 사용될 수 있는데, 아래 개시하는 기술된 변환 방식들은 모두 적용할 필요없이 그 중 일부를 선택적으로 적용할 수 있다.

[0448] 이러한 변환된 데이터에 기초하여 변환된 디스어셈블드 코드 내의 함수 별 데이터 세트를 기반으로 악성 코드와 유사도를 분석하는 단계이다.

[0449] 실시 예는 코드 간 유사도를 수행하기 위해 벡터화된 OP-CODE 와 ASM-CODE의 데이터 세트들을 바이트 데이터로 다시 변환할 수 있다.

[0450] 재변환된 바이트 데이터를 기반으로 블록 단위의 해쉬 값을 추출하고 블록 단위의 고유 값을 기반으로 전체 데이터의 해쉬 값을 생성할 수 있다.

[0451] 해쉬 값은 바이트 데이터의 부분인 블록 단위의 비교를 효율적으로 수행하기 위해서 각 블록 단위의 고유 값을 추출하도록 지정된 단위의 해쉬 값을 추출하여 비교할 수 있다.

[0452] 이와 같이 지정된 단위의 해쉬 값을 추출하고 2개 이상의 데이터의 유사도를 비교하기 위해 퍼지 해쉬(Fuzzy Hashing) 기법이 사용될 수 있다. 예를 들면 실시 예는 퍼지 해쉬(Fuzzy Hashing) 중 CTPH(Context Triggered Piecewise Hashing) 방식을 사용하여 블록 단위로 추출된 해쉬 값과 기 저장된 악성 코드 중 일부 단위의 해쉬

값을 서로 비교하여 유사도를 판단할 수 있다.

[0453] 정리하면 실시 예는 OP-CODE 및 ASM-CODE의 조합 코드가 특정 기능을 함수 단위로 구현한다는 사실에 기반하여, 각 특정 기능의 고유성을 확인하기 위해서 OP-CODE 와 ASM-CODE의 디스어셈블된 코드의 고유 값을 생성한다. 그리고 이 고유 값을 기반으로 디스어셈블된 코드의 OP-CODE와 ASM-CODE중 블록 단위의 고유 값을 추출하여 유사도 연산을 수행할 수 있다.

[0454] 블록 단위의 해쉬 값을 추출 하는 상세한 예도 아래에서 도면을 참조하여 개시하도록 한다.

[0455] 설명한 바와 같이 실시 예는 유사도 연산을 수행할 경우 블록 단위 해쉬 값을 이용할 수 있다.

[0456] 추출된 블록 단위 해쉬 값은 String Data (Byte Data) 로 구성되어 있고 String Data (Byte Data)는 수치화 값들로 코드 간의 유사도를 비교할 수 있다. 만약 수십억 개의 디스어셈블된 코드 데이터 세트의 바이트 비교를 수행하면 하나의 유사도 결과를 얻는데 엄청난 시간을 소비할 수 있다.

[0457] 따라서 실시 예는 String Data (Byte Data)는 수치화 값으로 변환할 수 있는데 이러한 수치화 값에 기반하면 인공지능 기술을 활용해 유사도 분석을 빠르게 수행할 수 있다.

[0458] 실시 예는 추출된 블록 단위의 해쉬 값의 String Data (Byte Data) 를 N-gram 데이터 기반으로 벡터화시킬 수 있다. 이 도면의 실시 예는 연산 속도를 높이기 위해 블록 단위의 해쉬 값을 2-gram 데이터로 벡터화 수행하는 경우를 예시한다. 그런데 실시 예는 블록 단위의 해쉬 값을 반드시 2-gram 데이터로 변환할 필요는 없으며 3-gram, 4-gram,…, N-gram의 데이터로 벡터화 변환하는 것도 가능하다. N-gram의 데이터에서 N이 증가할수록 데이터의 특성을 정확하게 반영할 수 있지만 데이터의 처리 시간의 속도가 증가한다.

[0459] 기술한 바와 같이 데이터 처리 속도를 높이고 데이터의 정확한 분석을 위해 바이트 변환, 해쉬의 변환 및 아래의 N-gram 변환은 선택적으로 적용할 수 있다.

[0460] 예시한 2-gram 변환 데이터는 최대 65,536 차원을 가진다. 학습 데이터의 차원이 높아질수록, 데이터의 분포가 희박해(sparse)지며, 이에 따라 분류 성능에 악영향을 끼칠 수 있다. 그리고 학습 데이터의 차원이 높아지면 데이터를 학습하기 위한 시간 복잡도와 공간 복잡도가 증가한다.

[0461] 이러한 문제점을 해결하기 위해 실시 예는 다양한 텍스트 표현 기반의 여러 가지 자연어 처리 알고리즘으로 처리할 수 있다. 이 실시 예에서는 이러한 알고리즘으로 TF-IDF(Term Frequency-Inversed Document Frequency) 기법을 예로 하여 설명한다.

[0462] 이 단계의 학습 데이터의 유사도를 처리하기 위한 일 예로서, 고차원 데이터 중에서 공격 식별자 또는 클래스(T-ID)를 판단할 경우 의미 있는 특징(패턴)을 선택하기 위해 TF-IDF(Term Frequency-Inversed Document Frequency) 기법을 사용할 수 있다. 일반적으로, TF-IDF 기법은 검색 엔진에서 유사도가 높은 문서를 찾기 위해 사용되는데 이를 계산하는 수학식들은 다음과 같다.

수학식 1

[0464]
$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

[0465] 여기서 $tf(t,d)$ 는 특정 문서 d 에서 특정 단어 t 의 빈도율을 의미하고 그 단어가 반복적으로 나올수록 높은 값을 갖는다.

수학식 2

[0467] $$idf(t,D)=\log\frac{N}{|\{d\in D: t\in d\}|}$$

[0469] $idf(t,D)$ 는 특정 단어 t 를 포함하는 문서 d 의 비율의 역수 값으로, 단어가 여러 문서에서 흔하게 나타날수록 낮은 값을 갖는다.

수학식 3

[0471] $$tf-idf(t,d,D)=tf(t,d)\times idf(t,D)$$

[0473] $tf-idf(t,d,D)$ 는 $tf(t,d)$ 와 $idf(t,D)$ 를 곱한 값으로, 어떤 단어가 어떤 문서에 더 적합한지 수치화시킬 수 있다.

[0474] TF-IDF 방식은 수학식 1에 의한 단어의 빈도와 수학식 2에 의한 역문서빈도 (문서의 빈도에 특정한 역수)를 이용하여 수학식 3과 같이 문서 단어 행렬 내의 단어의 중요도에 따라 가중치를 반영하는 하는 방식이다.

[0475] 실시 예에서 블록 단위의 코드 상의 단어의 특징 또는 패턴에 기반하여 해당 단어가 포함된 문서를 공격 식별자(T-ID)라고 추론할 수 있다. 따라서, 블록 단위의 코드로부터 추출된 패턴에 대해서 TF-IDF를 계산하면, 특정 공격 식별자(T-ID) 내에서 빈번하게 나타나는 패턴을 추출하거나 또는 특정 공격 식별자(T-ID)와 관련 없는 패턴을 가지는 코드를 제거할 수 있다.

[0476] 예를 들어, 특정 패턴 A는 모든 공격 식별자(T-ID)들에서 발현되는 패턴이라고 했을 때, 특정 패턴 A에 대한 TF-IDF 값은 낮게 측정될 것이다. 그리고 이러한 패턴은 실제 공격 식별자(T-ID)를 구분하기 위해 불필요한 패턴임을 판단할 수 있다. TF-IDF와 같은 자연어의 유사도 판단을 위한 알고리즘은 머신 러닝 알고리즘의 학습을 통해 수행될 수도 있다.

[0477] 실시 예는 이러한 불필요한 패턴을 제거하여 불필요한 연산을 줄이고 추론 시간을 단축시킬 수 있다.

[0478] 상세하게 실시 예는 변환되어 블록 단위 코드의 데이터에 대해, 여러 가지 자연어 처리의 텍스트 표현에 기초한 유사도 알고리즘을 수행할 수 있다. 유사도 알고리즘을 통해 공격 식별자와 관련이 없는 패턴의 코드는 제거하여 아래 수행되는 알고리즘 수행과 머신 러닝에 따른 분류 과정의 수행을 크게 단축시킬 수 있다.

[0479] 실시 예는 블록 단위의 코드 상의 특징 또는 패턴을 기반하여 공격 식별자의 패턴을 분류하기 위해 분류 모델링을 수행할 수 있다. 실시 예는 벡터화된 블록 단위의 코드 특징 또는 패턴이 알려진 공격 식별자의 패턴인지를 학습하고, 이를 정확한 공격 기법이나 구현방식으로 분류할 수 있다. 실시 예는 악성 코드와 유사한 코드 패턴이 있다고 판단된 코드에 대해 정확한 공격 구현 방식, 즉 공격 식별자와 공격자를 분류를 위해 여러 가지 앙상블 머신 러닝 모델들을 이용한다.

[0480] 앙상블 머신 러닝 모델들은 준비된 데이터를 여러 개의 분류 노드들을 생성하고 각 분류 노드의 대한 노드의 예측을 결합하여 정확한 예측을 수행하는 기법이다. 위에서 설명한 바와 같이 블록 단위의 코드 상의 단어의 특징 또는 패턴이 어떤 공격 구현 방식인지, 즉 공격 식별자 또는 공격자인지 분류하는 앙상블 머신 러닝 모델들을 수행한다.

[0481] 앙상블 머신 러닝 모델들을 적용 시에 과탐과 오탐을 방지하기 위해 준비된 데이터의 분류를 위한 임계 값을 설

정할 수 있다. 설정된 탐지 임계 값 이상의 데이터들만 분류하고 설정된 탐지 임계 값에 도달하지 못하는 데이터는 분류 수행을 하지 않을 수 있다.

[0482] 기술 바와 같이 데이터 처리 속도를 높이고 데이터의 정확한 분석을 위해 여러 가지 데이터 형식의 변환이 사용될 수 있다. 위에서 기술한 데이터 변환 방식 중

[0483] 이러한 앙상블 머신 러닝 모델들을 적용하는 대한 구체적인 실시 예는 아래의 이하에서 다시 상세히 설명한다.

[0484] 제 4단계로서 공격 기법(TTP)을 식별하여 라벨링을 부여하는 프로파일링 하는 과정을 설명하면 다음과 같다.

[0485] 이미 분석된 공격 코드 또는 악성 코드에 기반하여 입력된 바이너리 데이터의 OP-CODE와 ASM-CODE를 포함하는 디스어셈블드 코드의 특징 추출을 통해 벡터화시키는 예를 위에서 기술하였다.

[0486] 이렇게 벡터화된 데이터는 머신 러닝 모델링을 통해 학습된 후 특정 공격 기법으로 분류되고 분류된 코드들은 프로파일링 과정에서 상기 분류된 데이터의 라벨링이 수행된다.

[0487] 라벨링은 크게 두 부분에 수행될 수 있는데 하나는 표준화된 모델에서 정의한 공격 식별자에 대한 고유 인덱스를 붙이는 것이고 다른 하나는 공격 코드를 작성한 사용자에 대한 정보를 기입하는 것이다.

[0488] 라벨링은 표준화된 모델, 예를 들면 MITRE ATT&CK에서 반영된 공격 식별자(T-ID)에 따라 부여하도록 하여 추가적인 작업 없이 사용자에게 정확한 정보를 전달할 수 있도록 한다.

[0489] 그리고 라벨링은 공격 식별자뿐만 아니라 해당 공격 식별자를 구현한 공격자를 구별할 수 있도록 부여된다. 따라서 공격 식별자뿐만 아니라 공격자와 그에 따른 구현 방식을 식별할 수 있도록 제공할 수 있다.

[0490] 실시 예는 기존에 분류된 디스어셈블된 코드(OP-CODE, ASM-CODE, 또는 그 조합)의 데이터 세트를 학습한 데이터를 기반으로 고도화된 프로파일링이 가능한다. 실시 예는 위에서 개시한 정적 분석, 동적 분석, 또는 연관 분석의 데이터도 라벨링을 수행하는 참고 데이터로 활용할 수 있다. 따라서 기존에 분석되지 않은 데이터 세트라고 하더라도 정적, 동적, 및 연관 분석의 결과를 함께 고려하면 매우 빠르고 효율적으로 프로파일링 데이터를 확보할 수 있다.

[0491] 위에서 3단계의 악성 코드와 유사한 패턴을 가지는 코드를 학습하고 학습된 데이터가 분류되는 과정과 4단계의 분류된 데이터의 프로파일링 과정은 머신 러닝에 알고리즘에 의해 함께 진행될 수 있다.

[0492] 이에 대한 상세한 예는 아래에서 개시한다. 그리고 프로파일링된 데이터 세트의 실제 예도 아래에서 도면을 참고하여 예시하도록 한다.

[0494] 도 22는 개시하는 실시 예의 데이터 변환의 일 예로서 디스어셈블드 코드의 OP-CODE 및 ASM-CODE를 정규화된 코드로 변환한 값을 예시한 도면이다.

[0495] 설명한 바와 같이 실행 파일의 디스어셈블링을 수행하면 OP-CODE 및 ASM-CODE가 결합된 데이터가 출력된다.

[0496] 실시 예는 디스어셈블링된 데이터로부터 함수 별로 출력되는 주석 데이터를 제거하고 처리가 용이하도록 OP-CODE, ASM-CODE, 및 대응 파라미터의 배치 순서를 변경할 수 있다.

[0497] 재구성된 OP-CODE와 ASM-CODE를 정규화된 코드 데이터로 변경하는데, 이 도면의 예는 정규화된 코드 데이터로 CRC 데이터를 예시한다.

[0498] 일 예로 OP-CODE는 CRC-16로 변환하고 ASM-CODE로 CRC-32로 변환할 수 있다.

[0499] 예시한 표의 첫 번째 행에서 OP-CODE의 push함수를 0x45E9의 CRC-16 데이터로 변경하고, ASM-CODE의 55를 0xC9034AF6의 CRC-32 데이터로 변경한 것을 예시한다.

[0500] 두 번째 행에서는 OP-CODE의 mov함수를 0x10E3의 CRC-16 데이터로 변경하고, ASM-CODE의 8B EC 를 0x3012FD2C의 CRC-32 데이터로 변경하였다. 세 번째 행에서는 OP-CODE의 lea함수를 0xAACE의 CRC-16 데이터로 변경하고, ASM-CODE의 8D 45 0C를 0x9214A6AA의 CRC-32 데이터로 변경하였다.

[0501] 네 번째 행에서 OP-CODE의 push함수를 0x45E9의 CRC-16 데이터로 변경하고, ASM-CODE의 50를 0xB969BE79의 CRC-32 데이터로 변경한 것을 예시한다.

[0502] 이 예와 다르게 CRC 데이터와 다른 다른 정규화 코드 데이터나 길이가 다른 코드 데이터를 사용할 수도 있다.

[0503] 이렇게 디스어셈블링된 코드를 정규화된 코드로 변경하면 각 코드의 고유성을 확보하면서 이후의 연산, 유사도 산출 및 벡터화 수행을 용이하게 빠르게 수행할 수 있다.

[0505] 도 23은 개시하는 실시 예의 데이터 변환의 일 예로서 디스어셈블드 코드의 OP-CODE 및 ASM-CODE의 벡터화된 값을 예시한 도면이다.

[0506] 이 도면에서는 정규화된 OP-CODE 의 코드(위의 예에 따르면 CRC-16)와 정규화된 ASM-CODE (위의 예에 따르면 CRC-32)를 각각 벡터화시킨 결과를 예시한다.

[0507] 정규화된 OP-CODE 의 코드를 벡터화한 값(OP-CODE Vector)와 정규화된 ASM-CODE의 코드를 벡터화한 값(ASM-CODE Vector)을 이 도면에 표 형식으로 나타내었다.

[0508] 이 도면의 각 행의 OP-CODE Vector 값과 ASM-CODE Vector 값은 각각 도 22의 각 행의 OP-CODE의 정규화 값과 ASM- CODE의 정규화 값에 대응된다.

[0509] 예를 들어, 도 22의 표의 네 번째 행의 CRC 데이터 0x45E9와 0xB969BE79의 벡터화 값들은 각각 이 도면의 표의 네 번째 행의 17897와 185 105 121 44이 된다.

[0510] 이렇게 정규화된 데이터에 대해 벡터화를 수행하면 디스어셈블링된 OP-CODE의 함수와 ASM-CODE가 각각 고유 특징을 포함하면서 벡터화 값으로 변화된다.

[0512] 도 24는 개시하는 실시 예의 데이터 변환의 일 예로서 코드의 블록 단위를 해쉬 값으로 변환하는 예를 개시한 도면이다.

[0513] 유사도 분석을 수행하기 위해서 벡터화된 각 OP-CODE 및 ASM-CODE 의 데이터 세트는 바이트 데이터 형태로 재변환이 수행된다. 재변환된 바이트 데이터는 블록 단위의 해쉬 값으로 변환될 수 있다. 그리고 다시 블록 단위의 해쉬 값들에 기반하여 전체 재변환된 바이트 데이터의 해쉬 값을 생성한다.

[0514] 실시 예는 재변환된 해쉬 값을 산출하는데 MD5(Message-Digest algorithm 5), SHA1 (Secure Hash Algorithm 1), SHA 256이 등의 해쉬 값을 사용될 수도 있는데, 데이터 사이의 유사도 판단을 위한 퍼지 해쉬(Fuzzy Hash) 함수를 이용할 수 있다.

[0515] 이 도면의 표에서 첫 번째 행은 데이터에 포함될 수 있는 사람이 가독할 수 있는 character를 나타낸다. 재변환된 바이트 데이터 중 블록 단위에 포함되는 값은 이와 같은 가독성의 character들을 포함할 수 있다.

[0516] 각 character들은 두 번째 행의 아스키 값(ascii val)인 97, 98, 99, 100, …., 48, 49에 대응될 수 있다.

[0517] 첫 번째 행의 character 값들을 포함하는 데이터를 세그먼트하여 아스키 값들의 합산이 가능한 블록으로 분리할 수 있다.

[0518] 표의 세 번째 행은 4개의 character 를 가지는 블록 단위 내에서 각 character 값에 대응되는 아스키 값의 합산 값을 나타낸다.

[0519] 첫 번째 블록의 경우 그 블록 내 character 에 대응되는 아스키 값(ascii val) 97, 98, 99, 100의 합(ascii sum)인 394의 값을 가질 수 있다.

[0520] 그리고 마지막 행은 블록 단위의 아스키 값의 합이 Base 64의 표현으로 변환된 경우를 나타낸다. 문자(letter) K는 첫 번째 블록의 합산이 된다.

[0521] 이러한 방식으로 해당 데이터에 대해 Kaq6KaU라는 시그니처를 얻을 수 있다.

[0522] 이러한 시그니처를 기반으로 두 개의 블록 단위 데이터에 대한 유사도를 산출할 수 있다.

[0523] 이 실시 예는 재변환된 바이트 데이터 중 코드에 포함된 블록 단위들에 대해 유사도 판단을 위한 퍼지 해쉬 함수로 해쉬 값을 산출하고, 산출된 해쉬 값들을 기반으로 유사도를 판단할 수 있다. 유사도 판단을 위한 퍼지 해쉬 함수로 CTPH(Context Triggered Piecewise Hashing)를 예시하였으나 데이터의 유사도를 산출할 수 있는 다른 퍼지 해쉬 함수를 사용하는 것도 가능하다.

[0525] 도 25는 개시하는 실시 예에 따른 앙상블 머신 러닝 모델의 일 예를 나타낸 도면이다.

[0526] 실시 예는 앙상블 머신 러닝 모델을 이용하여 악성 코드로 판단되는 파일의 공격 식별자(T-ID)를 정확하게 분류할 수 있다.

[0527] String Data (Byte Data)로 구성된 블록 단위를 해쉬 값은 N-gram 특징 정보 기반으로 수치화시킨 후 이것이 공격 식별자(T-ID) 또는 분류될 클래스인지를 판단하기 위해 TF-IDF 등의 기법으로 유사도를 계산할 수 있다.

[0528] 불필요한 연산을 줄여 공격 기법 식별의 성능을 높이기 위해 실시 예는 위 해쉬 값 중 유사도를 기반으로 불필요한 패턴을 제거할 수 있다.

[0529] 그리고 불필요한 패턴이 제거된 데이터를 앙상블 머신 러닝을 통해 모델링하여 공격 식별자를 분류할 수 있다.

[0530] 앙상블 머신 러닝 모델의 여러 개의 분류 노드의 학습 결과들을 결합하기는 방식으로 보팅(Voting), 배깅(Bagging), 부스팅(Booting) 등의 방식이 있다 이러한 방식들을 적절히 조합한 앙상블 머신 러닝 모델은 학습 데이터의 분류 정확도를 높이는데 기여할 수 있다.

[0531] 여기서는 일 예로서 배깅 방식의 랜덤 포레스트(Random Forest) 방식을 적용하는 경우를 예를 들어 공격 식별자를 보다 정확하게 분류하는 방법을 설명한다.

[0532] 랜덤 포레스트(Random Forest) 방식은 많은 수의 디시전 트리(Decision Tree) 생성하여 단일 디시전 트리에 의한 분류 오류를 낮추고 일반화된 분류 결과를 얻는 방식이다. 실시 예는 준비된 데이터에 대해 적어도 하나 이상의 디시전 트리(Decision Tree)를 이용한 랜덤 포레스트(Random Forest) 학습 알고리즘을 적용할 수 있다. 여기서 준비된 데이터는 블록 단위의 퍼지 해쉬 값으로부터 불필요한 패턴이 제거된 데이터를 의미한다.

[0533] 블록 단위 해쉬 값의 유사도 판단을 위해 적어도 하나 이상의 노드를 가진 디시전 트리(Decision Tree)모델을 수행한다. 디시전 트리(Decision Tree)의 정보 획득(information gain) 정도에 따라 1개 이상의 클래스(공격 식별자; T-ID)를 구분할 수 있는 특징 값(여기서는 블록 단위 해쉬 값을 기초로 한 분류 패턴의 발현 개수)에 대해 비교 조건을 최적화할 수 있다.

[0534] 이를 위해 도면에서 예시한 바와 같은 디시전 트리(Decision Tree)를 생성할 수 있다.

[0535] 이 도면에서 위 쪽의 사각형(2510, 2520, 2530, 2540)들은 인 터미널 노드로서 클래스를 구분하는 조건을 의미하고 아래 쪽의 사각형 부분(2610, 2620, 2630)은 터미널 노드로 분류되는 클래스를 의미한다.

[0536] 예를 들어 랜덤 포레스트(Random Forest) 모델을 앙상블 머신 러닝 모델로 적용할 경우, 1개 이상의 디시전 트리(Decision Tree)를 이용하여 앙상블 기법을 사용하는 분류 모델이다. 랜덤 포레스트(Random Forest) 모델을 구성하는 디시전 트리(Decision Tree)의 입력 데이터의 특징을 다르게 하여 다양한 디시전 트리(Decision Tree)를 구성한다. 여러 개 생성된 디시전 트리(Decision Tree) 모델에 대해 분류를 수행하고 다수결 투표 기법을 사용하여 최종 분류 클래스를 결정한다. 각 노드의 테스트는 병렬적으로 진행될 수 있어 계산 효율이 높다.

[0537] 클래스를 분류할 경우 과탐과 오탐을 방지하기 위해 임계값을 설정하고 하한 임계값 이하의 값은 버리고, 탐지 임계값 이상의 데이터 대상으로 분류를 수행할 수 있다.

[0539] 도 26은 개시하는 실시 예에 따라 머신 러닝으로 데이터를 학습하고 분류하는 흐름을 예시한 도면이다.

[0540] 입력 데이터의 프로파일링은 분류 단계(S2610)과 학습 단계(S2620)를 포함할 수 있다.

[0541] 실시 예에서 학습 단계(S2620)는 (a) 해쉬 값 추출 과정, (b) N-gram 패턴 추출 과정, (c) 자연어 처리 분석(TF-IDF 분석) 과정, (d) 패턴 선택 과정, (e) 모델 학습 과정 등을 포함할 수 있다.

[0542] 그리고 실시 예에서 분류 단계(S2610)는, (a) 해쉬 값 추출 과정, (b) N-gram 패턴 추출 과정, (f) 패턴 선택 과정, (g) 벡터화에 의한 분류 과정 등을 포함할 수 있다.

[0543] 실시 예에 따른 프로파일링 단계 중 분류 단계(S2620)를 먼저 설명하면 다음과 같다.

[0544] 실행 파일 집합이나 처리된 파일로부터 입력 데이터를 수신한다.

[0545] 데이터베이스에 저장된 실행 파일 집합들로부터 입력 데이터를 수신하거나 또는 위에서 예시한 처리 과정으로부터 전달되는 실행 파일이 포함된 입력 데이터를 수신한다. 입력 데이터는 OP-CODE 와 ASM-CODE 코드를 포함하는

디스어셈블된 코드를 변환시킨 데이터로 벡터화시킨 데이터일 수 있다.

[0546] 입력 데이터인 디스어셈블된 코드로부터 퍼지 해쉬(Fuzzy Hash) 값을 추출(a)하고 특정 함수에 대한 N-gram 패턴 데이터를 추출한다(b). 이때 기존의 의미 패턴 집합 중 악성 코드와 유사하다고 판단한 패턴을 포함한 2-gram 의 패턴 데이터를 선택할 수 있다(f).

[0547] 선택한 패턴의 N-gram 데이터를 벡터화 데이터로 변환하고 벡터화 데이터를 의미가 패턴이 결정된 함수로 분류할 수 있다(g).

[0549] 실시 예에 따른 프로파일링 단계 중 학습 단계(S2610)는 다음과 같이 수행된다.

[0550] 만약 입력된 데이터가 새로운 파일이라면 입력 데이터인 디스어셈블된 코드로부터 퍼지 해쉬(Fuzzy Hash) 값을 추출한다(a).

[0551] 추출된 퍼지 해쉬(Fuzzy Hash) 값을 N-gram 데이터(이 예에서는 2-gram)로 벡터화시킨다(b).

[0552] 추출된 특정 패턴에 대해 TF-IDF 와 같은 자연어 처리 분석을 수행한다(c)

[0553] 기존의 공격 식별자(T-ID)와 관련된 패턴을 가지는 데이터 세트 중 유사도가 높은 데이터 세트를 선택하고 나머지는 필터링한다(d). 이때 기존의 의미 패턴 집합에 저장된 데이터 세트들과 비교하여 공격 식별자(T-ID)와 관련된 패턴을 가지는 데이터 세트의 일부 또는 전부의 특징을 포함한 샘플 데이터 세트들을 선택할 수 있다.

[0554] 추출된 샘플 데이터 세트를 기반으로 벡터화한 N-gram 데이터를 학습시킬 수 있다(e).

[0555] N-gram 의 벡터화 데이터를 분류 모델에 입력하여 공격 식별자(T-ID) 별로 확률을 얻는다. 예를 들어 N-gram 구조의 벡터화 데이터가 특정 공격 식별자(T-ID) T1027일 확률이 A%이고, 공격 식별자 T1055일 확률이 (100-A)%인 확률 등의 확률을 얻을 수 있다.

[0556] 분류 모델은 적어도 하나 이상의 디시전 트리를 포함하는 랜덤 포레스트 등의 앙상블 머신 러닝 모델을 이용할 수 있다.

[0557] (A) 여기서 분류 모델에 기반하여 벡터화한 N-gram 데이터가 어떤 공격 기법 또는 공격자인지 판단할 수 있다.

[0558] 분류 모델(e)의 분류 결과 또는 기존의 저장된 패턴의 선택(f) 결과에 따라 입력 데이터를 분류하여 라벨링을 수행한다(g).

[0559] 최종 라벨링이 수행된 결과는 다음의 도면을 참조하여 예시한다.

[0561] 도 27은 개시하는 실시 예에 따라 입력 데이터를 학습하고 분류하여 공격 식별자와 공격자를 라벨링한 예를 나타낸 도면이다.

[0562] 이 도면은 프로파일리의 결과로서 공격 식별자, 공격자 또는 공격 그룹, 어셈블리 코드에 대응되는 퍼지 해쉬 값, 그에 대응되는 N-gram(여기서는 2-gram 데이터로 기재)를 각각 표 형식으로 나타낸 도면이다.

[0563] 실시 예에 따라 프로파일링이 완료되면 다음과 같은 공격 방법의 구현과 관련하여 분류된 데이터를 얻을 수 있다.

[0564] 실시 예에 의한 프로파일링에 따라 공격 식별자(T-ID)와 공격자 또는 공격자 그룹(Attacker or Group)에 각각 라벨링될 수 있다.

[0565] 여기서 공격 식별자(T-ID)는 설명한 바와 같이 표준화된 모델에 따를 수 있는데 이 예에서는 MITRE ATT&CK®에서 제공하는 공격 식별자(T-ID)를 부여한 결과를 예시한다.

[0566] 위에서 기술한 바와 같이 식별된 공격자 또는 공격자 그룹(Attacker or Group)에도 라벨링이 추가될 수 있다. 이 도면은 공격자 또는 공격자 그룹(Attacker or Group)의 라벨링으로 공격자 TA504를 식별한 예를 나타낸다.

[0567] SHA-256 (size)는 각각의 공격 식별자(T-ID) 또는 공격자 그룹(Attacker or Group)에 대응되는 악성 코드의 퍼지 해쉬 값과 데이터 사이즈을 나타낸다. 설명한 바와 같이 이러한 악성 코드는 OP-CODE 와 ASM-CODE의 재배치와 조합에 대응될 수 있다.

[0568] 그리고 N-gram으로 표시한 섹션의 값은 공격 식별자(T-ID) 또는 공격자 그룹과 악성 코드의 퍼지 해쉬 값에 대응되는 N-gram 패턴 데이터로서, 이 예에서는 2-gram 데이터의 일부로 표시하였다.

[0569] 이 도면에서 예시한 바와 같이 악성 코드(OP-CODE 와 ASM-CODE)의 퍼지 해쉬 값과 N-gram 패턴 데이터에 대응되는 공격 식별자(T-ID) 또는 공격자 그룹이 라벨링되어 저장될 수 있다.

[0570] 예시한 라벨링된 데이터는 앙상블 머신 러닝의 참조 데이터로 이용될 수 있고, 분류 모델의 참조 데이터로 이용될 수도 있다.

[0572] 이하에서 개시한 실시 예들의 성능 결과를 예시한다.

[0573] 도 28은 실시 예에 따라 공격 식별자를 식별한 결과를 나타낸 도면이다.

[0574] 이 도면은 유클리언 디스턴스 매트릭스(Euclidean Distance Matrix)를 예시하는데, 유클리언 디스턴스 매트릭스(Euclidean Distance Matrix)는 두 데이터 세트 사이의 유사도를 나타낼 수 있다.

[0575] 이 도면에서 밝은 부분은 두 데이터 세트의 유사도가 낮은 것을 의미하고 어두운 부분은 두 데이터 세트의 유사도가 높은 것을 의미한다.

[0576] 이 도면에서 T10XX는 공격 식별자(T-ID)를 의미하고 괄호 안에 character T, K, L은 각각 해당 공격 식별자(T-ID)에 따른 공격 기법을 작성한 공격자 그룹을 의미한다.

[0577] 즉, 행과 열은 각각의 공격자 그룹들(T, K, L)이 생성한 공격 식별자(T-ID)들을 의미하며 행과 열은 동일한 의미를 가진다. 예를 들어 T1055(K)는 L 공격자 그룹이 생성한 T1055 공격을 의미하고, T1055(K)는 K 공격자 그룹이 생성한 동일한 공격 방법 T1055를 의미한다.

[0578] 각각의 데이터 세트의 샘플들은 자신의 샘플을 포함하기 때문에 다른 샘플들과의 거리를 각각 계산하면 왼쪽 위에서 오른쪽 아래의 대각선 방향으로 동일성이 높은 분포를 나타낸다.

[0579] 이 도면을 보면 동일한 공격 식별자(T-ID)의 경우 공격자 그룹이 다르더라도 유사한 특징을 나타내는 것을 확인할 수 있다. 예를 들어 T1027의 공격 식별자는 공격 그룹이 T 또는 K라고 하더라도 공격 기법이 유사하면 유사도가 높게 평가될 수 있다.

[0580] 따라서, 위의 실시 예와 같이 추출한 데이터 세트를 기반으로 학습을 진행하면 동일한 공격자가 구현한 같은 공격 기법(T-ID)에 대한 특징은 명확하게 식별되고(가장 어두운 부분), 다른 공격자가 구현한 동일한 공격 기법(T-ID)은 유사도가 높은 것(중간 어두운 부분)을 확인할 수 있다.

[0581] 따라서, 이와 같이 OP-CODE 와 ASM-CODE 의 조합에 기초한 샘플 데이터를 추출하여 적용해 공격 기법을 분류하면 공격자가 다른 경우라고 하더라도 특정의 공격 기법 또는 식별자(T-ID)를 확실하게 분류해 낼 수 있다. 반대로 OP-CODE 와 ASM-CODE 의 조합을 통해 악성 코드 내부에 구현된 특정 코드를 명확하게 식별할 수 있을 뿐만 아니라 공격자, 공격 식별자를 포함함 공격 구현 방식을 식별할 수 있다.

[0583] 도 29는 실시 예에 따라 공격 식별자에 따른 그램 데이터 패턴을 예시한 도면이다.

[0584] 이 도면은 서로 다른 공격 식별자 (T-ID)가 다른 경우 그램 데이터의 패턴을 예시한 도면이다. 예를 들어 공격 식별자 T1027과 T1055를 포함한 각각의 악성 코드를 2-gram의 패턴 데이터로 변환하여 실시예에 따라 분류하면 공격 식별자 (T-ID)가 별로 다른 그램 패턴을 보인다.

[0585] 즉, OP-CODE 와 ASM-CODE 의 조합을 기반으로 악성 코드 내 공격 기법들을 식별하는 실시 예에 따르면 공격 식별자 (T-ID)별로 그램 데이터의 패턴이 나뉠 수 있다.

[0586] 이 결과는 본 실시예에 따르면 공격자가 같더라도 악성 코드 내 숨겨진 여러 가지 공격 식별자 (T-ID)들을 명확하게 식별할 수 있다는 것을 의미한다.

[0588] 도 30은 개시한 사이버 위협 정보를 처리하는 실시 예의 성능을 예시한 도면이다.

[0589] 이 도면은 개시한 실시예의 성능 중 공격 식별자 또는 공격자를 분류하는 연산 속도에 대한 성능을 예시한 것이

다.

[0590] 가로축은 데이터베이스에 저장된 데이터의 양을 나타내고 세로축은 공격 식별자를 분류하는데 소요되는 시간을 나타낸다.

[0591] 데이터베이스에 저장된 퍼지 해쉬 데이터의 데이터의 개수를 증가시키면서, 일반적인 샘플을 각각 N : 1 (N대 1)로 비교하면 데이터의 개수 에 따라 처리 시간이 기하급수적으로 증가할 수 있다. 예를 들어 단순히 해쉬 값이나 퍼지 해쉬 값의 유사도만을 비교하면(ssdeep로 표시) 비교하는 데이터의 양에 따라 소요시간이 매우 증가한다.

[0592] 그러나 실시 예의 앙상블 머신 러닝 모델의 디시전 트리(Decision Tree) 모델을 이용하면 공격 식별자 등을 분류하는 추론 시간이 데이터의 개수가 증가해도 증가하지 않는다.

[0593] 즉 최적화된 비교 트리를 생성하는 디시전 트리(Decision Tree) 모델은 노드를 병렬적으로 처리할 수 있으므로 데이터 개수가 증가해도 계산 속도에 큰 영향을 받지 않는 장점이 있다.

[0595] 도 31은 사이버 위협 정보의 탐지하는 엔진들의 탐지 엔진들을 탐지 명을 제공하는 예를 나타낸 도면이다.

[0596] 악성코드 탐지 분야의 다양한 엔진들이 개발되어 사이버 위협 정보를 탐지 수행이 되고 있다. 인공 지능 분석이 늘어나면서 악성 코드의 탐지 능력이 증가하였다고 하더라도 탐지된 악성 코드를 제대로 설명하고 그 정보를 제공하지 못하면 이러한 탐지 능력의 효용성이 매우 떨어진다.

[0597] 이 도면은 VirusTotal 사이트에서 제공하는 해외 유명의 탐지 엔진들(3210)(왼쪽)과, 각 그 탐지 엔진이 제공하는 동일한 악성 코드의 탐지명(오른편)을 예시한 것이다.

[0598] 동일한 악성 코드의 식별과 전달이 정확하게 이루어지지 않기 때문에 해당 악성 코드가 어떤 이유로 탐지되었는지 식별하기 어렵다. 따라서 보안 담당자가 해당 정보에 기초하여 어떤 오브젝트에 대한 조치를 취해야 하는지 대응책을 찾기 힘들었고 보안 위협에 대한 리스크에 대응하기 힘들었다.

[0599] 그러나 개시하는 실시 예는 표준화된 모델인 MITRE ATT&CK 등에서 제공하는 공격 식별자의 매트릭스 요소와 그 조합으로 사이버 위협 정보를 제공하고 표준화된 식별자(T-ID)로 악성 코드에 대한 정보 제공함으로써 범용성과 효율성을 매우 높일 수 있다.

[0601] 이하에서는 개시한 실시 예에 기반하여 공격자 추적하고 새로운 공격을 예측할 수 있는 예를 부연하여 설명한다.

[0602] 도 32는 실시 예에 따라 새로운 악성 코드와 공격 방식을 예시하는 일 예를 나타낸 도면이다.

[0603] 코드의 개발자는 코드를 생성하는데 본인만의 고유의 습관들, 예를 들어 변수명 선언, 함수 호출 구조, 파라미터 호출 방법 등을 사용하는 경향이 매우 높다. 프로그램의 개발이 논리의 흐름과 경험에 기반해 생성되기 때문에 이러한 습관을 완전히 변경하는 것은 매우 어려운 것이다.

[0604] 이러한 근거에 기반하여 실시 예는 코드 상의 이와 같은 결과물들을 개발자의 핑거 프린팅로 사용하여 공격자를 추적할 수 있다.

[0605] 악성 코드의 공격 식별자(T-ID)를 기반으로 학습 데이터를 구성할 경우 위와 같은 특징 정보를 이용해서 개발자를 특정할 수 있다. 악성 코드의 디스어셈블된 코드는 이러한 개발자의 고유 특성이나 습관을 반영하고 있다.

[0606] 특정 해커가 특정 공격 기법을 구현하기 위해서 본인이 인지하지 못한 본인만의 사용하는 기법을 사용할 수 있으며 그 코드의 복잡도가 증가할수록 특정 개발자를 지정할 수 있는 가능성이 높아진다.

[0607] 또한 각 공격 식별자(T-ID) 별 OP-CODE 와 ASM-CODE 의 코드 블록을 조합하면 아직 알려지지 않은 신종 또는 변종의 악성 코드 탐지에도 사용될 수 있다.

[0608] 이 도면은 아래와 실시 예에 따라 디스어셈블된 OP-CODE 와 ASM-CODE의 조합을 통해 현존하지 않는 새로운 TTP의 조합을 만드는 예를 개시한다.

[0609] 이 예에서 T1044, T1039, T1211,…, T-N은 각각 공격 식별자(T-ID)들을 예시한다.

[0610] 각 공격 식별자에 대응하는 OP-CODE 1 ~ N 세트는 각각의 각 공격 식별자의 악성 코드에 포함되는 코드 세트를 의미한다.

[0611] 여기서 예시한 바와 같이 malware 악성 코드는 기존에 알려진 공격 식별자T1044의 OP-CODE 1, T1039의 OP-CODE2, T1211의 OP-CODE3, 및, T-N의 OP-CODE 1 등을 조합을 포함하는 악성 코드라고 하자. 이러한 OP-CODE의 조합의 세트를 포함하는 malware 악성 코드는 이미 알려진 코드일 수도 있고 알려지지 않은 코드일 수도 있다.

[0612] 유사한 방식으로 T1044의 OP-CODE 3, T1039의 OP-CODEN, T1211의 OP-CODE4 및, T-N의 OP-CODE 2 등을 포함하는 새로운 공격 기법을 찾을 수 있다.

[0613] 또는 T1044의 OP-CODE 4, T1039의 OP-CODE4, T1211의 OP-CODE2 및, T-N의 OP-CODE 3 등을 포함하는 새롭고 알려지지 않은 공격 기법을 찾을 수도 있다.

[0614] 위에서는 편의상 OP-CODE의 조합만으로 공격 기법을 찾는 예를 개시하였으나, OP-CODE와 ASM-CODE를 조합하여 디스어셈블드 코드를 생성하면 공격 기법을 찾을 뿐만 아니라 공격자나 공격 그룹도 식별할 수 있다.

[0615] 유사하게 OP-CODE와 ASM-CODE를 포함하는 디스어셈블드 코드의 재조합을 통해 새로운 코드 세트를 생성할 수 있다. 실행 파일의 함수에 대응되는 OP-CODE 뿐만 아니라 실행 파일의 대상이나 저장 위치를 나타내는 ASM-CODE를 재구성하거나 또는 재조합된 디스어셈블드 코드를 생성할 수 있다.

[0616] 이러한 재구성 디스어셈블드 코드를 머신 러닝을 통해 학습하여 기존에 분석된 악성 코드와 비교하면 세분화된 새로운 방식의 공격 기법과 이를 생성하는 공격자를 식별하는 것을 넘어 추후 공격 예측이 가능하다.

[0617] 이렇게 새로운 TTP 의 조합과 공격 경로의 조합은 지금까지 존재 하지 않았던 새로운 사이버 위협 또는 악성코드의 공격 방법을 만들어 낼 수 있는데, 실시 예는 이렇게 기존의 디스어셈블된 코드 세트를 조합하여 공격 가능한 코드가 생성되는지 확인할 수 있다. 공격 가능한 코드인지 여부는 동적 분석 등의 테스트 등을 통해 확인할 수도 있다.

[0618] 따라서 실시 예는 디스어셈블된 코드 세트의 조합을 통해 향후 있을 보안 위협에 대응할 수 있는 정보를 제공할 수 있어 이에 대한 선제적인 대응이 가능하다.

[0619] 예를 들면 조합된 코드에 기반하여 각 공격 기법(TTP) 별 사용 빈도나 사용 했을 때 성공 가능성 등의 값을 반영한 코드를 생성할 수 있다.

[0620] 또는 인공 지능을 학습을 통해 성공 확률이 높은 새로운 코드 블록 조합의 공격 코드나 악성 코드를 미리 생성할 수 있다. 그리고 이러한 정보를 반영하여 기존 보안 제품들이 대응 할 수 있는 패턴을 생성하거나 내부 시스템의 취약한 부분의 보안성을 강화할 수 있는 정보를 제공할 수 있다.

[0622] 도 33은 사이버 보안 위협 정보 처리 방법의 다른 일 실시 예를 예시한 도면이다.

[0623] 입력된 실행 파일을 디스어셈블링하여 디스어셈블된 코드를 얻고 상기 디스어셈블된 코드를 재구성하여 재구성된 디스어셈블드 코드를 얻는다(S3110).

[0624] 디스어셈블된 코드를 얻고 재구성하는 예는 도 18 및 도 21 등을 참조하여 설명하였다.

[0625] 상기 재구성된 디스어셈블드 코드를 처리하여 해쉬 함수로 변환하고 상기 해쉬 함수를 N 그램(N-gram) 데이터로 변환한다(3120).

[0626] 재구성된 디스어셈블드 코드를 일정한 포맷의 데이터 세트로 변환하는 예는 도 21, 도 24 등에 예시하였다.

[0627] 상기 변환된 N 그램(N-gram) 데이터의 블록 단위의 코드에 대해 앙상블 머신 러닝을 수행하여 상기 블록 단위의 코드를 상기 블록 단위의 코드가 수행하는 공격 기법의 식별자 및 상기 블록 단위의 코드를 생성한 공격자의 식별자로 프로파일링한다(S3130)

[0628] 이 단계의 공격 기법의 식별자와 공격자의 식별자를 프로파일링하는 예는 도 19, 도 20, 도 21, 도 25, 도 26, 도 27 등을 참조하여 설명하였다.

[0630] 도 34는 사이버 보안 위협 정보 처리 장치의 다른 일 실시 예를 예시한 도면이다.

[0631] 사이버 보안 위협 정보 처리 장치의 다른 일 실시예는 프로세서를 포함하는 서버(2100), 데이터베이스(2200), 및 인텔리전스 플랫폼(10000)을 포함할 수 있다.

[0632] 인텔리전스 플랫폼(10000)은 응용 프로그램 인터페이스(Application Programming Interface) (1100), 프레임워크(18000), 여러 가지 알고리즘과 수행 모듈을 실행하는 분석및예측모듈(18100), AI 엔진(1230)을 포함할 수 있다.

[0633] 데이터베이스(2200)는 이미 분류된 악성 코드 또는 악성 코드의 패턴 코드를 저장할 수 있다.

[0634] 서버(2100)의 프로세서는 응용 프로그램 인터페이스(Application Programming Interface) (1100)로부터 수신된 실행 파일을 입력된 실행 파일을 디스어셈블링하여 디스어셈블된 코드를 얻고 상기 디스어셈블된 코드를 재구성하여 재구성된 디스어셈블드 코드를 얻는 제 1 모듈(18101)의 수행할 수 있다.

[0635] 제 1 모듈(18101)의 수행 과정의 예는 도 18 및 도 21 등을 예시하였다.

[0636] 그리고 서버(2100)의 프로세서는 상기 재구성된 디스어셈블드 코드를 처리하여 해쉬 함수로 변환하고 상기 해쉬 함수를 N 그램(N-gram) 데이터로 변환하는 제 2 모듈(18103)을 수행할 수 있다.

[0637] 제 2 모듈(18103)의 수행 과정의 예는 도 21, 도 24 등에 예시하였다.

[0638] 서버(2100)의 프로세서는 상기 변환된 N 그램(N-gram) 데이터의 블록 단위의 코드에 대해 앙상블 머신 러닝을 수행하여 상기 블록 단위의 코드를 상기 블록 단위의 코드가 수행하는 공격 기법의 식별자 및 상기 블록 단위의 코드를 생성한 공격자의 식별자로 프로파일링하는 제 3 모듈(18105)을 수행할 수 있다.

[0639] 제 3 모듈(18105)의 수행 과정의 예는 도 19, 도 20, 도 21, 도 25, 도 26, 도 27 등을 참조하여 설명하였다.

[0641] 따라서 개시한 실시예에 따르면 머신 러닝으로 학습된 데이터와 정확하게 일치하지 않는 악성 코드라도 탐지하고 대응할 수 있고 악성 코드의 변종에 대응할 수 있다.

[0642] 실시예에 따르면 악성 코드의 변종이라도 매우 빠른 시간 내에 악성 코드, 공격 기법 및 공격자를 식별할 수 있고 나아가 추후의 특정 공격자의 공격 기법을 예측할 수 있다.

[0643] 실시예에 따르면 이러한 악성 코드 여부, 공격 기법, 공격 식별자 및 공격자를 기반으로 사이버 공격 구현 방식을 정확히 식별하고 이를 표준화된 모델로 제공할 수 있다. 실시예에 따르면 악성코드 탐지 명 등이 통일되지 않거나 사이버 공격 기법이 정확하게 기술되지 못하는 악성 코드의 정보를 정규화되고 표준화된 방식으로 제공할 수 있다.

[0644] 또한 기존에 알려지지 않은 악성 코드를 생성 가능성과 이를 개발할 수 있는 공격자들을 예측하고 미래에 어떤 사이버 위협 공격이 있을지 예측 가능한 수단을 제공할 수 있다.

[0645] 도 35는 실시예들에 따른 사이버 보안 위협 정보 처리 장치의 동작을 나타낸 블록도이다.

[0646] 도 35의 블록도(3500)는 도 1 내지 도 34에서 설명한 사이버 보안 위협 정보 처리 장치가 사용자의 파일 분석 요청 신호에 따라 파일 특징을 수집하고 분석하여 사용자별 클러스터링을 생성하는 동작의 예시이다.

[0647] 실시예들에 따른 사이버 보안 위협 정보 처리 장치는 사용자 (User A)로부터 파일 분석 요청 신호(3510)를 수신한다. 실시예들에 따른 사용자의 파일 분석 요청 신호(3510)는 네트워크를 통해 전송될 수 있다.

[0648] 실시예들에 따른 사이버 보안 위협 정보 처리 장치는 파일 분석 요청 신호에 따라 실행 파일을 분석한다(3520). 사이버 보안 위협 정보 처리 장치의 파일 분석의 구체적인 동작은 다음과 같다.

[0649] 실시예들에 따른 사이버 보안 위협 정보 처리 장치는 사용자의 분석 요청 대상인 실행 파일(EXE, EFL, APK 등)을 디스어셈블(Disassemble)한다(3521). 디스어셈블된 어셈블리 코드(assembly code)들은 OP-CODE(operation code)와 피연산자(operand)를 포함할 수 있다. OP-CODE(operation code)는 명령어 코드로 호칭할 수는 기계 언어 명령어를 나타내고, 피연산자(operand)는 실행 동작에 필요한 정보, 즉 기계 언어 명령어의 대상 데이터나 메모리 위치를 나타낸다. 상술한 바와 같이 디스어셈블(disassemble)된 어셈블리 코드(assembly code)들 중 OP-CODE를 제외한 부분은 ASM-CODE로 호칭되며, ASM-CODE 는 피연산자(operand) 부분을 포함할 수 있다. 디스어셈블링(disassembling)을 통해 오브젝트 코드 형식의 실행 가능한 파일은 특정 형식, 예를 들면 어셈블러 언어 형식의 코드 또는 디스어셈블된 코드로 변환된다.

[0650] 실시예들에 따른 사이버 보안 위협 정보 처리 장치는 디스어셈블된 코드로부터 사용자에 따른 일정 형식을 가진 OP-CODE (operation code) 와 ASM-CODE를 추출한다(3522). 실시예들에 따른 사이버 보안 위협 정보 처리 장치는 추출된 OP-CODE 와 ASM-CODE을 그대로 이용하지 않고 각 함수 별로 재구성하여 OP-CODE 배열을 다시 구성할 수 있다. OP-CODE 배열을 재정리할 경우 원본 바이너리 데이터도 함께 포함하여 데이터의 해석을 충분히 수행할 수 있도록 데이터를 재구성할 수 있다. 따라서 실시예들에 따른 사이버 보안 위협 정보 처리 장치는 OP-CODE 와 ASM-CODE의 새로운 조합은 공격 기법뿐만 아니라 공격자를 식별할 수 있는 기초 데이터를 제공할 수 있다. OP-CODE 와 ASM-CODE의 조합 방법은 도 1 내지 도 34에서 설명한 바와 동일하므로 구체적인 설명은 생략한다. 조합된 OP-CODE 와 ASM-CODE는 [OP-CODE, ASM-CODE] 순서로 재정리된다. 이와 같이 재정리된 디스어셈블된 코드는 정규화 또는 벡터화하여 처리하기 용이하여 데이터 처리 속도를 높일 수 있다. [OP-CODE, ASM-CODE] 의 조합을 가지는 디스어셈블된 코드 중 ASM-CODE 부분은 데이터의 길이가 달라 서로 비교하기 용이하지 않다. 따라서 사이버 보안 위협 정보 처리 장치는 해당 어셈블리 데이터의 고유성을 확인하기 위해서 데이터를 특정 크기의 데이터 포맷으로 정규화시킬 수 있다. OP-CODE와 ASM-CODE 변환된 정규화 데이터는 각각 해당 변환 이전의 각각 코드의 고유성을 유지할 수 있다. 사이버 보안 위협 정보 처리 장치는 고유성을 가지고 변환된 정규화 데이터의 유사도 판단 속도를 빠르게 하기 위해 정규화된 데이터를 벡터화(Vectorization)할 수 있다. 실시예들에 따른 정규화 및/또는 벡터화 과정은 데이터 처리 및 분석을 위해 선택적으로 적용될 수 있다. 정규화 및/또는 벡터화 과정은 도 1 내지 도 34에서 설명한 바와 동일하므로 구체적인 설명은 생략한다. 사이버 보안 위협 정보 처리 장치는 벡터화된 OP-CODE 와 ASM-CODE의 데이터 세트들을 바이트 데이터로 다시 변환할 수 있다. 사이버 보안 위협 정보 처리 장치는 재변환된 바이트 데이터를 기반으로 블록 단위의 해쉬 값을 추출하고 블록 단위의 고유 값을 기반으로 전체 데이터의 해쉬 값을 생성할 수 있다. 해쉬 값은 바이트 데이터의 부분인 블록 단위의 비교를 효율적으로 수행하기 위해서 각 블록 단위의 고유 값을 추출하도록 지정된 단위의 값이다. 해쉬 값을 추출하기 위해 퍼지 해쉬(Fuzzy Hashing) 기법이 사용될 수 있다. 해쉬 값 추출 및 퍼지 해쉬 기법에 대해서는 도 1 내지 도 34에서 설명한 바와 동일하다. 사이버 보안 위협 정보 처리 장치는 추출된 해쉬 값과 기 저장된 악성 코드 중 일부 단위의 해쉬 값을 서로 비교하여 유사도를 판단할 수 있다.

[0651] 해쉬 값은 String Data (Byte Data) 로 구성되어 있으므로 수십억 개의 디스어셈블된 코드 데이터 세트의 바이트 비교를 수행하면 하나의 유사도 결과를 얻는데 엄청난 시간을 소비할 수 있다. 따라서 사이버 보안 위협 정보 처리 장치는 연산의 속도를 높이기 위하여 String Data (Byte Data)를 N-gram 데이터 기반으로 벡터화할 수 있다(3523). N-gram 변환의 N 값이 증가하면, 변환된 데이터는 원래 데이터의 특성을 정확하게 반영할 수 있지만, 데이터 변환 처리 시간이 지연될 수 있다. 본 예시에서는 2-gram을 설명하고 있으나, 이는 예시에 불과하며, 3-gram, 4-gram,…, N-gram 데이터 기반의 다양한 벡터화 변환이 적용될 수 있다. N-gram 변환은 도 1 내지 도 34에서 설명한 바와 동일하므로 구체적인 설명은 생략한다.

[0652] 사이버 보안 위협 정보 처리 장치는 변환된 데이터에 대하여 자연어 처리의 텍스트 표현에 기초한 유사도 알고리즘을 수행할 수 있다. 유사도 알고리즘을 통해 공격 식별자와 관련이 없는 패턴의 코드는 제거할 수 있다. 또한 사이버 보안 위협 정보 처리 장치는 블록 단위의 코드 상의 특징 또는 패턴을 기반하여 공격 식별자의 패턴을 분류하기 위해 분류 모델링을 수행할 수 있다. 예를 들어 사이버 보안 위협 정보 처리 장치는 벡터화된 블록 단위의 코드 특징 또는 패턴이 알려진 공격 식별자의 패턴인지를 학습하고, 이를 정확한 공격 기법이나 구현방식으로 분류할 수 있다. 실시예들에 따른 사이버 보안 위협 정보 처리 장치는 악성 코드와 유사한 코드 패턴이 있다고 판단된 코드에 대해 정확한 공격 구현 방식, 즉 공격 식별자와 공격자를 분류를 위해 여러 가지 앙상블 머신 러닝 모델들을 이용할 수 있다. 머신 러닝 모델들에 대한 설명은 도 1 내지 도 34에서 설명한 바와 동일하므로 구체적인 설명은 생략한다.

[0653] 또한 사이버 보안 위협 정보 처리 장치는 변환된 데이터를 특정 공격 기법(TTP)로 식별하고 라벨링하여 파일 분석 결과를 생성한다. 라벨링은 크게 두 부분에 수행될 수 있는데 하나는 표준화된 모델에서 정의한 공격 식별자에 대한 고유 인덱스를 붙이는 것이고 다른 하나는 공격 코드를 작성한 사용자에 대한 정보를 기입하는 것이다. 라벨링은 표준화된 모델, 예를 들면 MITRE ATT&CK에서 반영된 공격 식별자(T-ID)에 따라 부여하도록 하여 추가적인 작업 없이 사용자에게 정확한 정보를 전달할 수 있도록 한다. 라벨링은 공격 식별자뿐만 아니라 해당 공격 식별자를 구현한 공격자를 구별할 수 있도록 부여된다. 따라서 사이버 보안 위협 정보 처리 장치는 공격 식별자뿐만 아니라 공격자와 그에 따른 구현 방식을 식별할 수 있는 정보를 제공할 수 있다. 실시예들에 따른 라벨링은 도 1 내지 도 34에서 설명한 바와 동일하므로 구체적인 설명은 생략한다.

[0654] 또한 사이버 보안 위협 정보 처리 장치는 기존에 분류된 디스어셈블된 코드(OP-CODE, ASM-CODE, 또는 그 조합)의 데이터 세트를 학습한 데이터를 기반으로 고도화된 프로파일링 정보를 생성할 수 있다. 실시예들에 따른 프

로파일링 정보는 라벨링된 공격 식별자, 공격자 또는 공격 그룹, 어셈블리 코드에 대응되는 악성 코드의 퍼지 해쉬 값, 공격 식별자(T-ID) 또는 공격자 그룹과 악성 코드의 퍼지 해쉬 값에 대응되는 N-gram(예를 들면 2-gram 데이터)를 포함할 수 있다. 실시예들에 따른 공격 식별자(T-ID)는 설명한 바와 같이 표준화된 모델(예를 들면 상술한 MITRE ATT&CK®에서 제공하는 공격 식별자(T-ID))에 따른다. 실시예들에 따른 프로파일링 정보 및 생성 방법은 도 1 내지 도 34에서 설명한 바와 동일하므로 구체적인 설명은 생략한다.

[0655] 실시예들에 따른 사이버 보안 위협 정보 처리 장치는 파일 분석 결과(예를 들면 라벨링 결과 정보, 프로파일링 정보 등)를 사용자에게 제공한다(3530). 실시예들에 따른 파일 분석 결과는 텍스트, 이미지 등 다양한 포맷으로 제공될 수 있다. 또한 사이버 보안 위협 정보 처리 장치는 프로파일링 정보를 사용자 별로 클러스터링하여 데이터 베이스에 저장한다. 실시예들에 따른 사용자 별 클러스터링된 데이터는 사용자(예를 들면 사용자 A)가 주로 분석을 요청하는 파일들의 특징을 수집하여 생성되는 데이터 군집이다. 즉, 클러스터링은 사용자(예를 들면 사용자 아이디, 사용자 식별자 등)을 부여하는 라벨링을 통해 생성될 수 있다. 또한 클러스터링된 데이터는 사용자 별 데이터 베이스에 저장 및 관리 될 수 있다. 실시예들에 따른 사이버 보안 위협 정보 처리 장치는 파일 분석 요청 대상 파일에 대한 파일 분석 결과만을 사용자에게 제공할 수 도 있고, 클러스터링된 데이터(또는 클러스터링된 데이터의 결과)를 제공할 수도 있다. 또한 사용자는 클러스터링된 데이터를 저장하는 데이터 베이스에 직접 접근(access)할 수도 있다. 또한 사이버 보안 위협 정보 처리 장치는 사용자의 요청에 따라 또는 요청이 없이도 네트워크를 통해 클러스터링된 데이터를 제공할 수 있다. 도 35에 도시된 블록도는 예시에 불과하며, 사이버 보안 위협 정보 처리 장치의 동작은 본 예시에만 국한되지 않는다. 또한 실시예들에 따른 사이버 보안 위협 정보 처리 장치는 사용자 별 클러스터링 데이터를 저장하는 적어도 하나 이상의 사용자 별 클러스터링 데이터 베이스 및 실행 파일 분석 요청 신호를 입력받는 입력부를 포함할 수 있다.

[0656] 도 36은 실시예들에 따른 사이버 보안 위협 정보 처리 장치의 동작을 나타낸 블록도이다.

[0657] 도 36의 블록도(3600)는 도 1 내지 도 35에서 설명한 사이버 보안 위협 정보 처리 장치가 사용자(예를들면 사용자 A)의 파일 분석 요청 신호에 따라 파일을 분석하고 해당 파일의 특징과 다른 사용자들(예를 들면 사용자 B, 사용자 C)의 클러스터링 데이터와 유사도를 비교하여 사용자 친화적인 사이버 위협 정보 제공 동작을 나타낸다.

[0658] 사이버 보안 위협 정보 처리 장치는 사용자(사용자 A)로부터 실행 파일(신규 파일 “ABC.exe”)분석을 요청하는 실행 파일 분석 요청 신호(또는 파일 분석 요청 신호라고 호칭)를 수신한다(3610). 실시예들에 따른 사이버 보안 위협 정보 처리 장치는 파일을 분석할 수 있다(예를 들면 도 35의 3520). 사이버 보안 위협 정보 처리 장치의 파일 분석 동작은 도 1 내지 도 35에서 설명한 바와 동일하므로 구체적인 설명은 생략한다.

[0659] 실시예들에 따른 사이버 보안 위협 정보 처리 장치는 신규 파일 분석 결과(예를 들면 ABC exe “특징” 정보)를 사용자(예를 들면 사용자 식별자)를 부여하는 라벨링을 통해 클러스터링한다(3620). 기존에 사용자 A의 파일 분석 요청에 따라 생성된 클러스터링된 데이터가 저장된 사용자 A 클러스터링 데이터 베이스가 있는 경우, 사용자 A의 클러스터링된 데이터는 사용자 A 클러스터링 데이터 베이스에 저장 및 관리된다. 따라서 사이버 보안 위협 정보 처리 장치는 파일 분석 결과를 사용자 A의 클러스터링 데이터 베이스에 저장할 수 있다. 만약 사용자 A 클러스터링 데이터 베이스가 없다면 사이버 보안 위협 정보 처리 장치는 생성된 사용자 A의 클러스터링 데이터를 저장 및 관리하기 위한 사용자 A 클러스터링 데이터 베이스를 생성할 수 있다. 실시예들에 따른 파일 분석 결과는 도 1 내지 도 35에서 설명한 특정 공격 기법(TTP)으로 라벨링된 데이터, 프로파일링 정보 중 적어도 어느 하나 이상에 대응할 수 있다. 또한 실시예들에 따른 클러스터링 및 사용자 A의 클러스터링 데이터 베이스의 저장은 순차적으로 또는 동시에 수행될 수 있다. 또한 클러스터링 및 클러스터링 데이터 베이스의 저장은 본 예시에 국한되지 않는다.

[0660] 실시예들에 따른 사이버 보안 위협 정보 처리 장치는 신규 파일 분석 결과(예를 들면 ABC exe “특징”를 기반으로, 신규 파일 분석을 요청하지 않은 사용자에게 신규 파일 분석 결과 정보를 제공할 수 있다. 사이버 보안 위협 정보 처리 장치는 신규 파일 분석 결과와 신규 파일 분석을 요청하지 않은 사용자의 클러스터링 데이터의 유사도를 비교하여 유사한 경우, 해당 사용자의 요청이 없어도 신규 파일 분석 결과 정보 및/또는 유사도와 관련된 정보를 제공할 수 있다. 왜냐하면 유사도가 높은 경우 (또는 유사한 경우), 해당 사용자가 주로 분석을 요청하는 대상 파일들과 신규 분석 대상 파일의 유사도가 높다는 것을 의미하기 때문이다. 따라서 신규 분석 대상 파일을 요청하지 않은 사용자에게도 신규 파일 분석 결과 및 유사도와 관련된 정보는 미래의 사이버 위협을 분석하기 위한 유의미한 정보가 될 수 있다. 즉, 사이버 보안 위협 정보 처리 장치는 기 분석된 사용자별 클러스터링 데이터를 활용하여, 보다 사용자 친화적인 사이버 위협 정보 서비스를 제공할 수 있다.

[0661] 구체적으로, 사이버 보안 위협 정보 처리 장치는 파일 분석 결과(예를 들면 ABC exe “특징” 정보)를 사용자 A

가 아닌 적어도 하나 이상의 다른 사용자들(예를 들면 사용자 B, 사용자 C)의 클러스터링데이터 베이스에 저장된 데이터와 비교할 수 있다(3630, 3631). 상술한 바와 같이 다른 사용자들의 클러스터링된 데이터는 사용자(예를 들면 사용자 아이디, 식별자)를 부여하는 라벨링을 통해 생성되는 데이터로서, 신규 파일 분석 결과와 동일한 포맷, 데이터 종류 등으로 구성된 데이터를 포함한다. 또한 각 사용자별 클러스터링된 데이터는 각 사용자별 클러스터링 데이터 베이스(DB)에 저장 및 관리된다. 사이버 보안 위협 정보 처리 장치는 신규 파일 분석 결과와 사용자별 클러스터링 데이터 베이스 내의 데이터의 유사도를 비교한다. 유사도가 기설정된 유사도보다 높은 경우, 사이버 보안 위협 정보 처리 장치는 신규 파일 분석 결과 및 유사도와 관련된 정보를 해당 사용자 (예를 들면 사용자 C)에게 제공한다(3640. 신규 파일 분석 결과 및 유사도와 관련된 정보는 알람 메시지 등의 형태로 사용자에게 제공될 수 있으며, 제공 방법, 형태, 제공 정보 등은 본 예시에 국한되지 않는다. 또한 신규 파일 분석 결과 및 유사도와 관련된 정보는 파일명 및 유사도를 포함할 수 있다. 실시예들에 따른 사이버 보안 위협 정보 처리 장치는 사용자의 요청에 따라 또는 사용자 요청과 관계없이 신규 파일 분석 결과 및 유사도와 관련된 정보를 해당 사용자의 클러스터링 DB에 저장할 수 있다. 만약 유사도가 기설정된 유사도보다 작은 경우, 사이버 보안 위협 정보 처리 장치는 신규 파일 분석 결과 및 유사도와 관련된 정보를 해당 사용자 (예를 들면 사용자 B)에게 제공하지 않는다(3650). 실시예들에 따른 신규 파일 분석 결과 및 유사도와 관련된 정보는 사용자 요청에 따라 유사도와 관계없이 사용자에게 제공될 수도 있다. 도 36에 도시된 블록도는 예시에 불과하며, 사이버 보안 위협 정보 처리 장치의 동작은 본 예시에만 국한되지 않는다.

[0662] 도 37은 실시예들에 따른 사이버 보안 위협 정보 처리 방법을 나타내는 플로우 다이어그램이다.

[0663] 도 37의 플로우 다이어 그램(3700)은 도 36에서 설명한 사이버 보안 위협 정보 처리 장치의 사이버 보안 위협 정보 처리 방법의 예시이다.

[0664] 실시예들에 따른 사이버 보안 위협 정보 처리 장치는 신규 파일 분석 요청 신호를 수신한다(3710). 상술한 바와 같이 신규 파일 분석 요청 신호는 특정 사용자의 요청 신호로서 네트워크 등을 통해 수신될 수 있다.

[0665] 실시예들에 따른 사이버 보안 위협 정보 처리 장치는 신규 파일 분석 요청 신호에 따라 실행 파일을 분석한다(3720). 사이버 보안 위협 정보 처리 장치는 실행파일을 디스어셈블링하고, OP 코드 및

[0666] ASM 코드를 추출하고 벡터화 한 뒤 2-gram 데이터를 추출하여 공격 식별자 및 해당 공격 식별자를 구현한 공격자 식별 정보를 부여한 파일 분석 결과를 생성할 수 있다. 사이버 보안 위협 정보 처리 장치의 파일 분석 동작은 도 1 내지 도 36에서 설명한 바와 동일하므로 구체적인 설명은 생략한다.

[0667] 실시예들에 따른 사이버 보안 위협 정보 처리 장치는 파일 분석 결과에 대하여 클러스터링을 수행한다(3730). 예를 들어 사이버 보안 위협 정보 처리 장치는 파일 분석 결과에 대하여 파일 분석을 요청한 사용자(예를 들면 사용자 아이디, 사용자 식별자)를 부여하는 라벨링을 통해 클러스터링 데이터를 생성할 수 있다. 클러스터링 데이터 생성 방법은 도 35 내지 도 36에서 설명한 바와 동일하므로 구체적인 설명은 생략한다.

[0668] 실시예들에 따른 사이버 보안 위협 정보 처리 장치는 클러스터링 데이터의 사용자와 기존 사용자 별 클러스터링 데이터를 저장하는 데이터 베이스(DB)의 사용자가 다른지 판단한다(3740). 사용자가 동일한 경우, 클러스터링 데이터는 해당 사용자의 클러스터링 데이터 베이스에 저장된다(3751). 사용자가 다른 경우, 사이버 보안 위협 정보 처리 장치는 클러스터링 데이터와 해당 사용자와 다른 사용자의 클러스터링 데이터 베이스에 저장된 데이터간의 유사도를 비교한다(3752). 실시예들에 따른 유사도 비교동작은 클러스터링 데이터를 해당 사용자의 클러스터링 데이터 베이스에 저장하는 동작과 동시에 또는 순차적으로 수행될 수 있다.

[0669] 실시예들에 따른 사이버 보안 위협 정보 처리 장치는 유사도가 기설정된 값보다 높은지 판단한다(3760). 기설정된 값은 사이버 보안 위협 정보 처리 장치의 관리자 또는 사용자에 의해 설정 및 변경 가능하다.

[0670] 유사도가 기설정된 값보다 높은 경우, 사이버 보안 위협 정보 처리 장치는 비교 대상인 클러스터링 데이터 베이스의 사용자에게 유사도와 관련된 정보를 제공한다(3770). 실시예들에 따른 유사도와 관련된 정보는 신규 파일 분석 결과 및 유사도를 포함하며 알람 메시지 등의 형태로 사용자에게 제공될 수 있다. 또한 유사도와 관련된 정보의 제공 방법, 형태, 제공 정보 등은 본 예시에 국한되지 않는다. 사이버 보안 위협 정보 처리 장치는 사용자의 요청 및/또는 설정, 또는 사용자의 요청 및/또는 설정과 관계없이 해당 사용자의 클러스터링 데이터 베이스에 유사도와 관련된 정보를 저장할 수 있다.

[0671] 유사도가 기설정된 값보다 낮은 경우, 사이버 보안 위협 정보 처리 장치는 비교 대상인 클러스터링 데이터 베이스의 사용자에게 유사도와 관련된 정보를 제공하지 않는다(3780). 실시예들에 따른 사이버 보안 위협 정보 처리 장치는 사용자의 요청 또는 설정에 따라 유사도가 기설정된 값보다 낮은 경우에도 유사도와 관련된 정보를 제공

할 수 있다.

[0672] 도 37에 도시된 플로우 다이어그램은 예시에 불과하며, 플로우 다이어그램을 구성하는 엘레멘트들의 순서등은 본 예시에 국한되지 않는다.

[0673] 도 38은 실시예들에 따른 사이버 보안 위협 정보 처리 장치의 예시이다.

[0674] 도 38의 예시(3800)는 도 1 내지 도 37에서 설명한 사이버 보안 위협 정보 처리 장치의 블록도이다. 실시예들에 따른 사이버 보안 위협 정보 처리 장치는 입력부(3810), 물리 장치인 프로세서(3820)와 데이터베이스(3830) 및 물리장치상에서 구동되는 응용 프로그래밍 인터페이스 Application Programming Interface, API) 포함하는 플랫폼(3840)를 포함할 수 있다. 도 38은 사이버 보안 위협 정보 처리 장치의 예시로서 사이버 보안 위협 정보 처리 장치는 본 예시에 국한되지 않는다 따라서 사이버 보안 위협 정보 처리 장치는 도 1 내지 도 37에서 설명한 동작 및/또는 방법을 수행하기 위하여 도면에 도시되지 않은 하나 또는 그 이상의 엘레멘트들을 더 포함할 수 있다.

[0675] 실시예들에 따른 입력부(3810)는 사용자로부터 입력 신호(예를 들면 파일 분석 요청 신호)를 입력받는다. 실시예들에 따른 프로세서(3820)는 데이터베이스(3830)에 데이터를 저장하거나 읽을 수 있다. 또한 실시예들에 따른 프로세서(3820)는 도 1 내지 도 37에서 설명한 서버(예를 들면 도 10의 서버(2100)) 입력부(3810)를 통해 입력된 파일 분석 요청 신호를 기반으로 플랫폼(3840)에 포함된 하나 또는 그 이상의 모듈들을 수행할 수 있다. 또한 프로세서(3820)는 사용자 입력 신호에 따라, 또는 사용자 입력 신호와 관계없이 사이버 보안 위협 정보 처리 장치에서 생성된 정보(예를 들면 도 1 내지 도 37에서 설명한 파일 분석 결과, 클러스터링된 데이터, 유사도와 관련된 정보 등)를 사용자에게 전달할 수 있다. 상술한 바와 같이 사이버 보안 위협 정보 처리 장치에서 생성된 정보의 포맷, 종류 등은 예시에 국한되지 않는다. 실시예들에 따른 프로세서(3820)는 하나의 블록으로 도시되어 있으나, 중앙연산장치(central processing unit, CPU) 와 같은 연산장치, 데이터 입력을 위한 네트워크 장치나 네트워크의 보안 장치 등을 포함하는 하나 또는 그 이상의 프로세스들을 나타낼 수 있다. 프로세서(3820)에 대한 설명은 도 1 내지 도 37에서 설명한 바와 동일하므로 생략한다.

[0676] 실시예들에 따른 데이터 베이스(3830)는 프로세서(3820)의 제어에 따라 사이버 보안 위협 정보 처리 장치에서 생성된 데이터(예를 들면 도 1 내지 도 37에서 설명한 파일 분석 결과, 클러스터링된 데이터, 유사도와 관련된 정보 등)를 저장할 수 있다. 실시예들에 따른 데이터 베이스(3830)는 하나의 블록으로 도시되어 있으나, 상술한 바와 같이 하나 또는 그 이상의 사용자 별 데이터 베이스들을 나타낼 수 있다.

[0677] 실시예들에 따른 플랫폼(3840)(예를 들면 도 10에서 설명한 인텔리전스 플랫폼(10000))은 사이버 위협 정보의 처리를 위한 응용 프로그래밍 인터페이스(API)를 제공할 수 있다. 실시예들에 따른 플랫폼(3840)은 프로세서(3820)에 의해 수행되는 파일분석 모듈(3841), 클러스터링 생성 모듈(3842) 및 유사도 비교 모듈(3843)을 포함한다. 파일분석 모듈(3841)은 도 35에서 설명한 파일 분석 동작(3520)을 수행할 수 있다. 파일분석 모듈(3841)은 하나의 블록으로 도시되어 있으나 도 1 내지 도 37에서 설명한 다양한 모듈들(예를 들면 전처리부(미도시), 분석 프레임 워크(1210)와 예측 프레임 워크(1220) 및 AI 엔진 (1230) 및 후처리부(미도시)에 포함된 모듈들)을 포함할 수 있다. 클러스터링 생성 모듈(3842)은 파일분석 모듈(3841)에서 생성된 파일 분석 결과에 대하여 클러스터링을 수행하여 클러스터링 데이터를 생성할 수 있다(예를 들면 도 36에서 설명한 클러스터링 생성(3620), 도 37의 클러스터링 수행 (3730)). 클러스터링 데이터를 생성하는 방법은 도 35 내지 도 37에서 설명한 바와 동일하므로 생략한다. 실시예들에 따른 유사도 비교 모듈(3843)은 파일 분석 결과와 파일 분석 결과를 요청한 사용자와 다른 사용자의 클러스터링 DB에 저장된 데이터간의 유사도를 비교한다(예를 들면 도 36에서 설명한 유사도 비교 (3630, 3631), 도 37에서 설명한 유사도 비교(3760)). 유사도 비교 방법은 도 35 내지 도 37에서 설명한 바와 동일하므로 생략한다. 실시예들에 따른 프로세서(3820)는 유사도가 기설정된 값보다 높은 경우, 유사도와 관련된 정보를 생성하고 해당 사용자에게 유사도와 관련된 정보를 제공한다.

[0678] 도 39는 실시예들에 따른 사이버 보안 위협 정보 처리 방법의 플로우 다이어그램이다.

[0679] 도 39의 플로우 다이어그램(3900)은 도 1 내지 도 38에서 설명한 사이버 보안 위협 정보 처리 장치의 사이버 보안 위협 정보 처리 방법 또는 저장매체에 저장된 사이버 보안 위협 정보를 처리하는 프로그램이 수행하는 사이버 보안 위협 정보 처리 방법의 예시를 나타낸다.

[0680] 사이버 보안 위협 정보 처리 장치(예를 들면 도 38의 입력부(3810))는 실행 파일 분석을 위한 제 1 사용자 실행 파일 분석 요청 신호(예를 들면 도 36에서 설명한 사용자 A로부터 입력받은 신규 파일 ABC.exe 분석 요청 신호)를 입력받는다(3910).

[0681] 사이버 보안 위협 정보 처리 장치(예를 들면 도 38의 프로세서(3820) 및 파일분석 모듈(3841))는 제 1 사용자 실행 파일 분석 요청 신호에 따라 실행 파일을 디스어셈블링(disassembling)하여 디스어셈블된 코드를 얻고 상기 디스어셈블된 코드를 재구성하여 재구성된 디스어셈블드 코드를 얻는다(3920). 실시예들에 따른 디스어셈블된 코드는, 실행 파일에 포함된 함수에 대응하는 OP-CODE와 함수의 피연산자인 어셈블리 코드를 포함할 수 있다. 디스어셈블링 및 디스어셈블된 코드에 대한 에 대한 설명은 도 1 내지 도 38에서 설명한 바와 동일하므로 생략한다.

[0682] 사이버 보안 위협 정보 처리 장치(예를 들면 도 38의 프로세서(3820) 및 파일분석 모듈(3841))는 재구성된 디스어셈블드 코드를 처리하여 해쉬 함수로 변환하고 해쉬 함수를 N 그램(N-gram, N은 자연수) 데이터로 변환한다(3930). 사이버 보안 위협 정보 처리 장치(예를 들면 도 38의 프로세서(3820) 및 파일분석 모듈(3841))는 해쉬 함수를 바이트 데이터로 변환하고, 바이트 데이터를 2-gram 데이터로 변환할 수 있다. 실시예들에 따른 N 그램 데이터 변환에 대한 설명은 도 1 내지 도 38에서 설명한 바와 동일하므로 생략한다.

[0683] 사이버 보안 위협 정보 처리 장치(예를 들면 도 38의 프로세서(3820) 및 파일분석 모듈(3841))는 변환된 N 그램(N-gram) 데이터의 블록 단위의 코드에 대해 앙상블 머신 러닝을 수행하여 블록 단위의 코드를 블록 단위의 코드가 수행하는 공격 기법의 식별자 및 상기 블록 단위의 코드를 생성한 공격자의 식별자로 프로파일링하여 실행 파일 분석 결과를 생성한다(3940). 사이버 보안 위협 정보 처리 장치(예를 들면 도 38의 프로세서(3820) 및 파일분석 모듈(3841))는 블록 단위의 코드와 저장된 악성 코드의 유사 패턴을 찾고, 유사 패턴인 블록 단위의 코드에 대해 적어도 하나의 노드를 가지는 디시전 트리를 이용하여 공격 기법의 식별자와 공격자의 식별자를 분류할 수 있다. 실시예들에 따른 프로파일링에 대한 설명은 도 1 내지 도 38에서 설명한 바와 동일하므로 생략한다. 사이버 보안 위협 정보 처리 장치(예를 들면 도 38의 프로세서(3820) 및 파일분석 모듈(3841))는 변환된 N 그램(N-gram) 데이터의 블록 단위의 코드에 대해블록 단위의 코드가 사이버 공격 행위가 포함된 코드인지 판단할 경우, 블록 단위의 코드를 자연어 처리 방식에 따라 저장된 악성 코드와 유사도를 판단할 수 있다. 악성 코드와 유사도를 판단하는 방법에 대한 설명은 도 1 내지 도 38에서 설명한 바와 동일하므로 생략한다.

[0684] 사이버 보안 위협 정보 처리 장치(예를 들면 도 38의 프로세서(3820), 클러스터링 생성 모듈(3842) 및 유사도 비교 모듈(3843))는 실행 파일 분석 결과를 기반으로 제 1 사용자(예를 들면 도 35 내지 도 38에서 설명한 사용자 A) 클러스터링 데이터를 생성하고, 제 1 사용자와 다른 제 2 사용자(예를 들면 도 35 내지 도 38에서 설명한 사용자 B)의 클러스터링 데이터와 실행 파일 분석 결과간의 유사도를 비교한다(3940). 실시예들에 따른 제 1 사용자 클러스터링 데이터는 실행 파일 분석 결과에 제 1 사용자를 나타내는 정보를 부여하는 라벨링을 수행하여 생성될 수 있다. 클러스터링 데이터 생성 방법에 대한 설명은 도 1 내지 도 38에서 설명한 바와 동일하므로 생략한다.

[0685] 사이버 보안 위협 정보 처리 장치(예를 들면 도 38의 프로세서(3820))는 유사도가 기설정된 값보다 높은 경우, 제 2 사용자에게 유사도와 관련된 정보를 제공한다(3950). 실시예들에 따른 유사도와 관련된 정보는 파일 분석 결과 및 유사도를 포함할 수 있다. 사이버 보안 위협 정보 처리 장치(예를 들면 도 38의 프로세서(3820))는 유사도가 기설정된 값보다 낮은 경우, 제 2 사용자에게 유사도와 관련된 정보를 제공하지 않는다. 유사도와 관련된 정보에 대한 설명은 도 1 내지 도 38에서 설명한 바와 동일하므로 생략한다.

[0686] 설명의 편의를 위하여 각 도면을 나누어 설명하였으나, 각 도면에 서술되어 있는 실시예들을 병합하여 새로운 실시예를 구현하도록 설계하는 것도 가능하다. 그리고, 통상의 기술자의 필요에 따라, 이전에 설명된 실시예들을 실행하기 위한 프로그램이 기록되어 있는 컴퓨터에서 판독 가능한 기록 매체를 설계하는 것도 실시예들의 권리범위에 속한다. 실시예들에 따른 장치 및 방법은 상술한 바와 같이 설명된 실시예들의 구성과 방법이 한정되게 적용될 수 있는 것이 아니라, 실시예들은 다양한 변형이 이루어질 수 있도록 각 실시예들의 전부 또는 일부가 선택적으로 조합되어 구성될 수도 있다. 바람직한 예시에 대하여 도시하고 설명하였지만, 실시예들은 상술한 특정의 예시에 한정되지 아니하며, 청구범위에서 청구하는 실시예들의 요지를 벗어남이 없이 당해 발명이 속하는 기술분야에서 통상의 지식을 가진 자에 의해 다양한 변형실시가 가능한 것은 물론이고, 이러한 변형실시들은 실시예들의 기술적 사상이나 전망으로부터 개별적으로 이해돼서는 안 될 것이다. 실시예들에 따른 장치 및 방법에 대한 설명은 서로 보완하여 적용될 수 있다.

[0687] 실시예들에 따른 장치의 다양한 구성요소들은 하드웨어, 소프트웨어, 펌웨어 또는 그것들의 조합에 의해 구성될 수 있다. 실시예들의 다양한 구성요소들은 하나의 칩, 예를 들면 하나의 하드웨어 서킷으로 구현될 수 있다 실시예들에 따라, 실시예들에 따른 구성요소들은 각각 별도의 칩들로 구현될 수 있다. 실시예들에 따른 장치의 구성요소들 중 적어도 하나 이상은 하나 또는 그 이상의 프로그램들을 실행 할 수 있는 하나 또는 그 이상의 프로

세서들로 구성될 수 있으며, 하나 또는 그 이상의 프로그램들은 실시예들에 따른 동작/방법들 중 어느 하나 또는 그 이상의 동작/방법들을 수행시키거나, 수행시키기 위한 인스트럭션들을 포함할 수 있다. 실시예들에 따른 장치의 방법/동작들을 수행하기 위한 실행 가능한 인스트럭션들은 하나 또는 그 이상의 프로세서들에 의해 실행되기 위해 구성된 일시적이지 않은 CRM 또는 다른 컴퓨터 프로그램 제품들에 저장될 수 있거나, 하나 또는 그 이상의 프로세서들에 의해 실행되기 위해 구성된 일시적인 CRM 또는 다른 컴퓨터 프로그램 제품들에 저장될 수 있다. 또한 실시예들에 따른 저장매체(또는 메모리)는 휘발성 메모리(예를 들면 RAM 등)뿐 만 아니라 비휘발성 메모리, 플래쉬 메모리, PROM등을 전부 포함하는 개념으로 사용될 수 있다. 또한, 인터넷을 통한 전송 등과 같은 캐리어 웨이브의 형태로 구현되는 것도 포함될 수 있다. 또한, 프로세서가 읽을 수 있는 기록매체는 네트워크로 연결된 컴퓨터 시스템에 분산되어, 분산방식으로 프로세서가 읽을 수 있는 코드가 저장되고 실행될 수 있다.

[0688] 이 문서에서 “/”와 “,”는 “및/또는”으로 해석된다. 예를 들어, “A/B”는 “A 및/또는 B”로 해석되고, “A, B”는 “A 및/또는 B”로 해석된다. 추가적으로, “A/B/C”는 “A, B 및/또는 C 중 적어도 하나”를 의미한다. 또한, “A, B, C”도 “A, B 및/또는 C 중 적어도 하나”를 의미한다. 추가적으로, 이 문서에서 “또는”는 “및/또는”으로 해석된다. 예를 들어, “A 또는 B”은, 1) “A”만을 의미하거나, 2) “B”만을 의미하거나, 3) “A 및 B”를 의미할 수 있다. 달리 표현하면, 본 문서의 “또는”은 “추가적으로 또는 대체적으로(additionally or alternatively)”를 의미할 수 있다.

[0689] 제1, 제2 등과 같은 용어는 실시예들의 다양한 구성요소들을 설명하기 위해 사용될 수 있다. 하지만 실시예들에 따른 다양한 구성요소들은 위 용어들에 의해 해석이 제한되어서는 안된다. 이러한 용어는 하나의 구성요소를 다른 구성요소와 구별하기 위해 사용되는 것에 불과하다. 것에 불과하다. 예를 들어, 제 1 사용자 인풋 시그널은 제 2 사용자 인풋 시그널로 지칭될 수 있다. 이와 유사하게, 제 2 사용자 인풋 시그널은 제 1사용자 인풋시그널로 지칭될 수 있다. 이러한 용어의 사용은 다양한 실시예들의 범위 내에서 벗어나지 않는 것으로 해석되어야만 한다. 제 1 사용자 인풋 시그널 및 제2 사용자 인풋 시그널은 모두 사용자 인풋 시그널들이지만, 문맥상 명확하게 설명되지 않는 한 동일한 사용자 인풋 시그널들을 의미하지 않는다.

[0690] 실시예들을 설명하기 위해 사용된 용어는 특정 실시예들을 설명하기 위한 목적으로 사용되고, 실시예들을 제한하기 위해서 의도되지 않는다. 실시예들의 설명 및 청구항에서 사용된 바와 같이, 문맥 상 명확하게 지칭하지 않는 한 단수는 복수를 포함하는 것으로 의도된다. 및/또는 표현은 용어 간의 모든 가능한 결합을 포함하는 의미로 사용된다. 포함한다 표현은 특징들, 수들, 단계들, 엘리먼트들, 및/또는 컴포넌트들이 존재하는 것을 설명하고, 추가적인 특징들, 수들, 단계들, 엘리먼트들, 및/또는 컴포넌트들을 포함하지 않는 것을 의미하지 않는다. 실시예들을 설명하기 위해 사용되는, ~인 경우, ~때 등의 조건 표현은 선택적인 경우로만 제한 해석되지 않는다. 특정 조건을 만족하는 때, 특정 조건에 대응하여 관련 동작을 수행하거나, 관련 정의가 해석되도록 의도되었다.

부호의 설명

[0691] 1010, 1020, 1030: 클라이언트

1100: 응용 프로그래밍 인터페이스

1210, 150000: 분석프레임워크

1211, 15100: 정적분석모듈

1213, 15200: 동적분석모듈

1215, 15300: 심층분석모듈

1217,15400: 연관관계분석모듈

1220,17000: 예측프레임워크

1221: 제 1예측정보생성모듈

1223: 제 2예측정보생성모듈

1230: AI 엔진

2000: 물리장치

2200: 데이터베이스

2100: 서버

2510, 2520, 2530,2540, 2610, 2620,2630 디시전 트리의 노드

10000: 인텔리전스 플랫폼

15101: 파일구조분석모듈

15103: 파일패턴분석모듈

15105: 파일제작정보분석모듈

15107: 파일환경분석모듈

15109: 파일관련분석모듈

15201: 환경준비모듈

15203: 파일실행모듈

15205: 행위수집모듈

15207: 분석결과취합모듈

15209: 분석환경복구모듈

15301: 디스어셈블링모듈

15303: 기계언어코드추출모듈

15309: 공격기법식별모듈

15307: 공격자식별모듈

15309: 테인트분석모듈

15401: 제1연관관계모듈

15403: 제2연관관계모듈

15409: 제3연관관계모듈

15407: 제4연관관계모듈

15409: 제5연관관계모듈

17100: 예측정보생성모듈

17101: 제1정보예측모듈

17103: 제2정보예측모듈

17105: 제3정보예측모듈

17107: 제4정보예측모듈

17109: 제5정보예측모듈

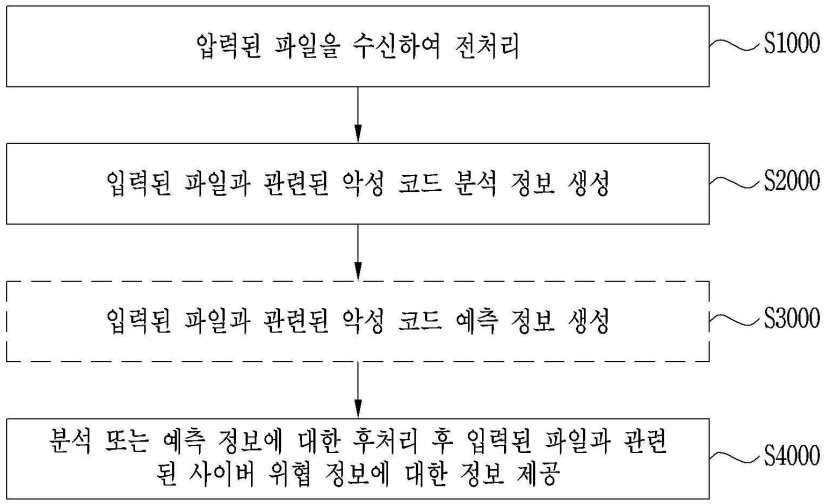
18000: 프레임워크

18100: 분석및예측모듈

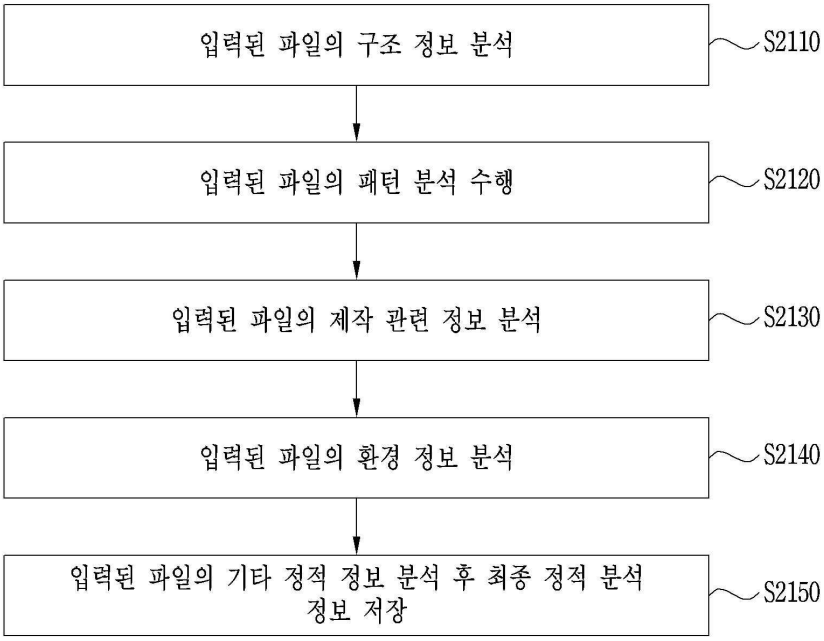
18101, 18103, 18105: 제 1 모듈, 제 2 모듈, 제 3 모듈

도면

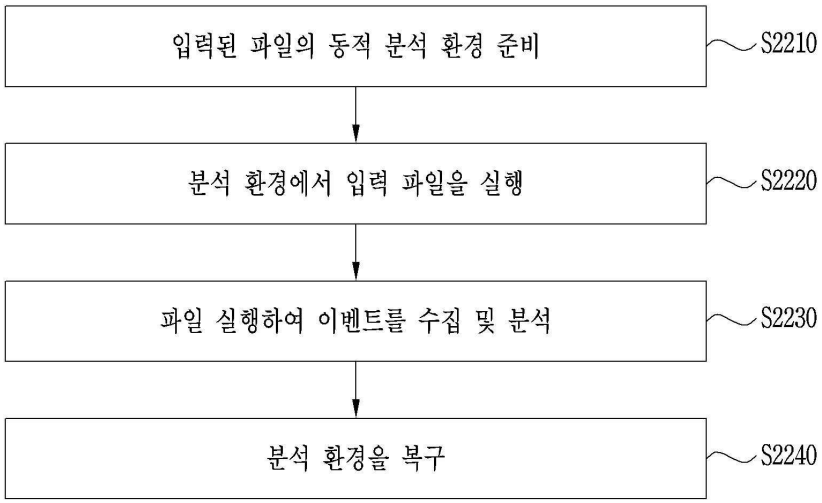
도면1


압력된 파일을 수신하여 전처리
S1000
입력된 파일과 관련된 악성 코드 분석 정보 생성
S2000
입력된 파일과 관련된 악성 코드 예측 정보 생성
S3000
분석 또는 예측 정보에 대한 후처리 후 입력된 파일과 관련된 사이버 위협 정보에 대한 정보 제공
S4000

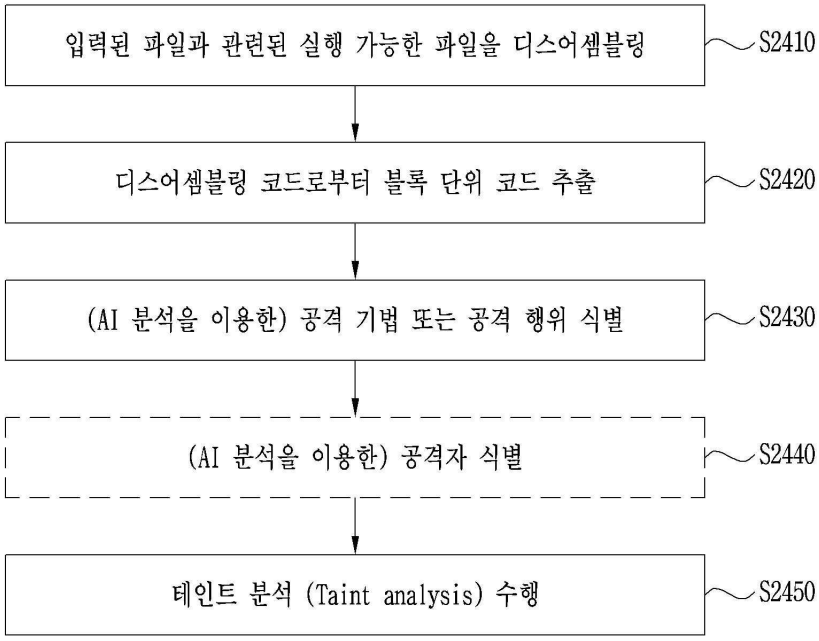
도면2


입력된 파일의 구조 정보 분석
S2110
입력된 파일의 패턴 분석 수행
S2120
입력된 파일의 제작 관련 정보 분석
S2130
입력된 파일의 환경 정보 분석
S2140
입력된 파일의 기타 정적 정보 분석 후 최종 정적 분석 정보 저장
S2150

도면3

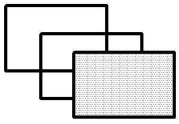

입력된 파일의 동적 분석 환경 준비
S2210
분석 환경에서 입력 파일을 실행
S2220
파일 실행하여 이벤트를 수집 및 분석
S2230
분석 환경을 복구
S2240

도면4


입력된 파일과 관련된 실행 가능한 파일을 디스어셈블링
S2410
디스어셈블링 코드로부터 블록 단위 코드 추출
S2420
(AI 분석을 이용한) 공격 기법 또는 공격 행위 식별
S2430
(AI 분석을 이용한) 공격자 식별
S2440
테인트 분석 (Taint analysis) 수행
S2450

도면5

EXE 파일



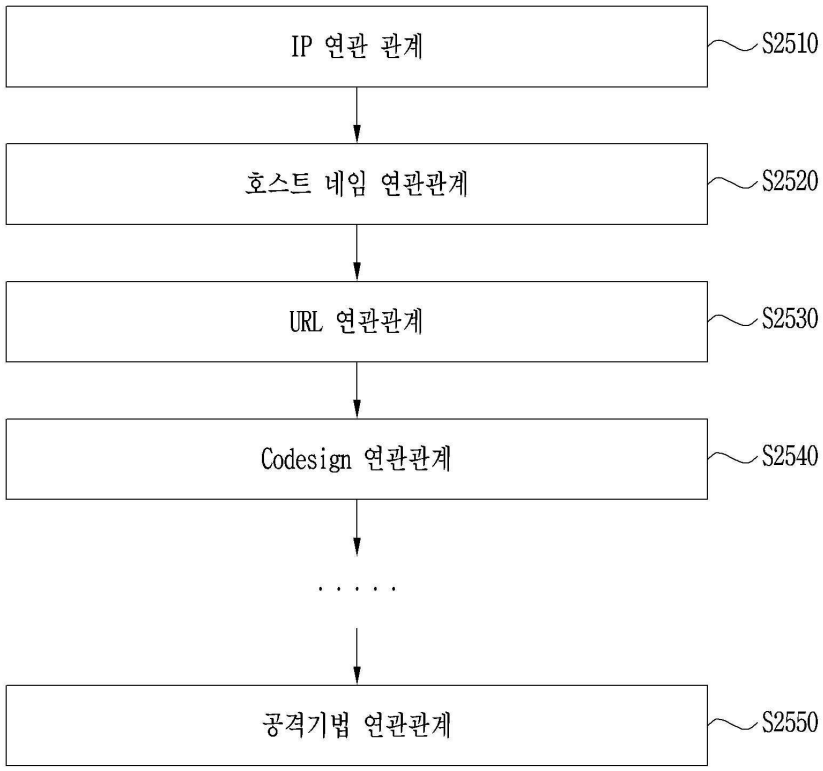
Disassembled code set

파일	OP 코드	T-ID	설명
malware.exe	MOV DWORD PTR SS: [EBP-4] , 1 MOV DWORD PTR SS: [EBP-8] , 2 MOV EDX, DWORD PTR SS: [EBP-8] LEA EAX, DWORD PTR SS: [EBP-4]	1022	시스템 주요 레지스트리 변경

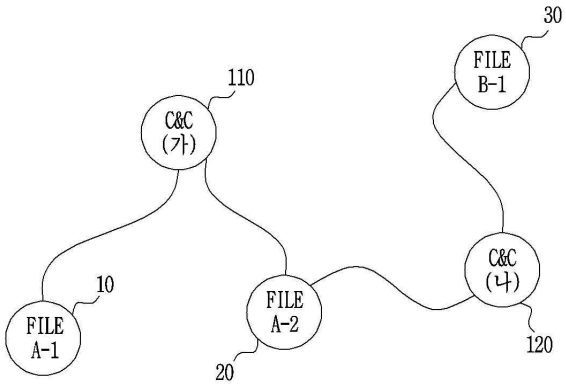
	PUSH EBP
	MOV EBP, ESP
	SUB ESP, 18
	AND ESP, FFFFFFF0
	MOV EAX, 0
		1077	시작 프로그램 등록

	LEA EAX, DWORD PTR SS: [EBP-4] MOV EDX, DWORD PTR SS: [EBP-8] LEA EAX, DWORD PTR SS: [EBP-4] LEA EAX, DWORD PTR SS: [EBP-4]	1034	윈도우 방화벽 해제
	PUSH EBP MOV EBP, ESP MOV EAX, DWORD PTR SS: [EBP+B] ADD EAX, DWORD PTR SS: [EBP+C] POP EBP RETN	1090	신규 유저 추가
	CMP DWORD PTR SS: [EBP-4], 2 JNZ SHORT if. 00401035 PUSH if. 0040C008 CALL if.printf ADD ESP, 4 JMP SHORT if. 00401042	2011	백도어 생성
	CMP DWORD PTR SS: [EBP-B], 1 JE SHORT switch. 00401027 CMP DWORD PTR SS: [EBP-B], 2 JE SHORT switch. 00401036 CMP DWORD PTR SS: [EBP-B], 3 JE SHORT switch. 00401045 JMP SHORT switch. 00401054	3744	보안 프로그램 동작 중지

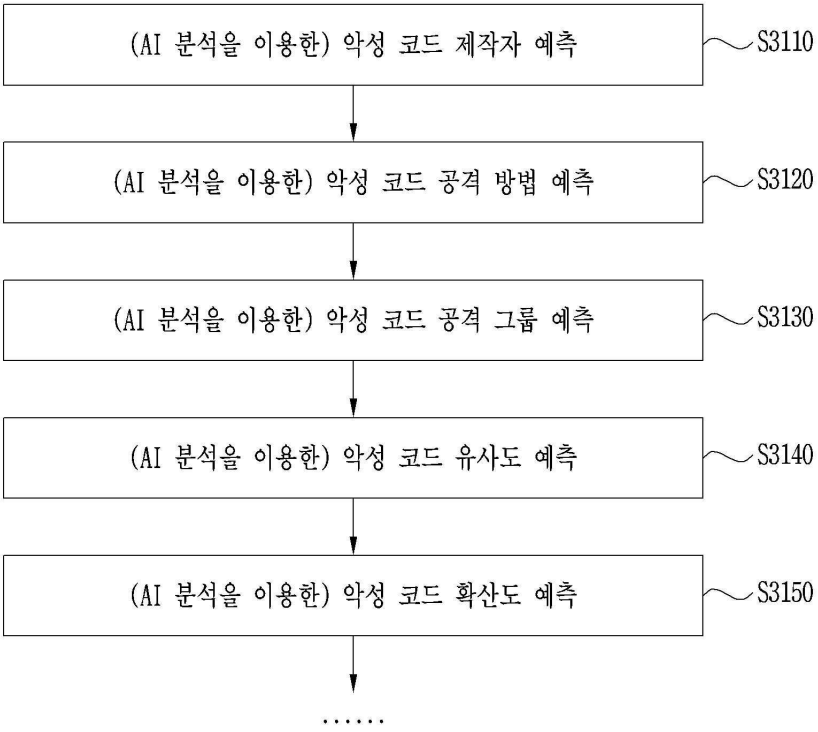
도면6


IP 연관 관계
S2510
호스트 네임 연관관계
S2520
URL 연관관계
S2530
Codesign 연관관계
S2540
.....
공격기법 연관관계
S2550

도면7


30
FILE
B-1
110
C&C
(가)
C&C
(나)
10
FILE
A-1
FILE
A-2
20
120

도면8


(AI 분석을 이용한) 악성 코드 제작자 예측
S3110
(AI 분석을 이용한) 악성 코드 공격 방법 예측
S3120
(AI 분석을 이용한) 악성 코드 공격 그룹 예측
S3130
(AI 분석을 이용한) 악성 코드 유사도 예측
S3140
(AI 분석을 이용한) 악성 코드 확산도 예측
S3150
.....

도면9

Query (A)

"2020년 3월 1일 오후 4시 ~ 2020년 3월 2일 오전 9시 사이에 Anti-Virus 40개 이상 탐지한 악성코드 중에 탐지한 Anti-virus 탐지명 중에 "*ransomware*" 라는 단어가 들어가 있고 EXE 파일 타입이면서 유포지(ITW)가 "*go.kr*"이고 CodeSign 서명 정보가 존재하는 파일 크기 10MB 이하의 파일"

Query (B)

"2020년 3월 1일에 Anti-Virus 10개 이하 탐지한 파일 중에 HWP, DOC, PPT, PDF, XLS 파일 타입이면서 유포지가 존재하고 행위 분석 결과 "powershell.exe" 또는 "vbscript.exe"가 실행되는 파일 크기 10MB 이하의 파일

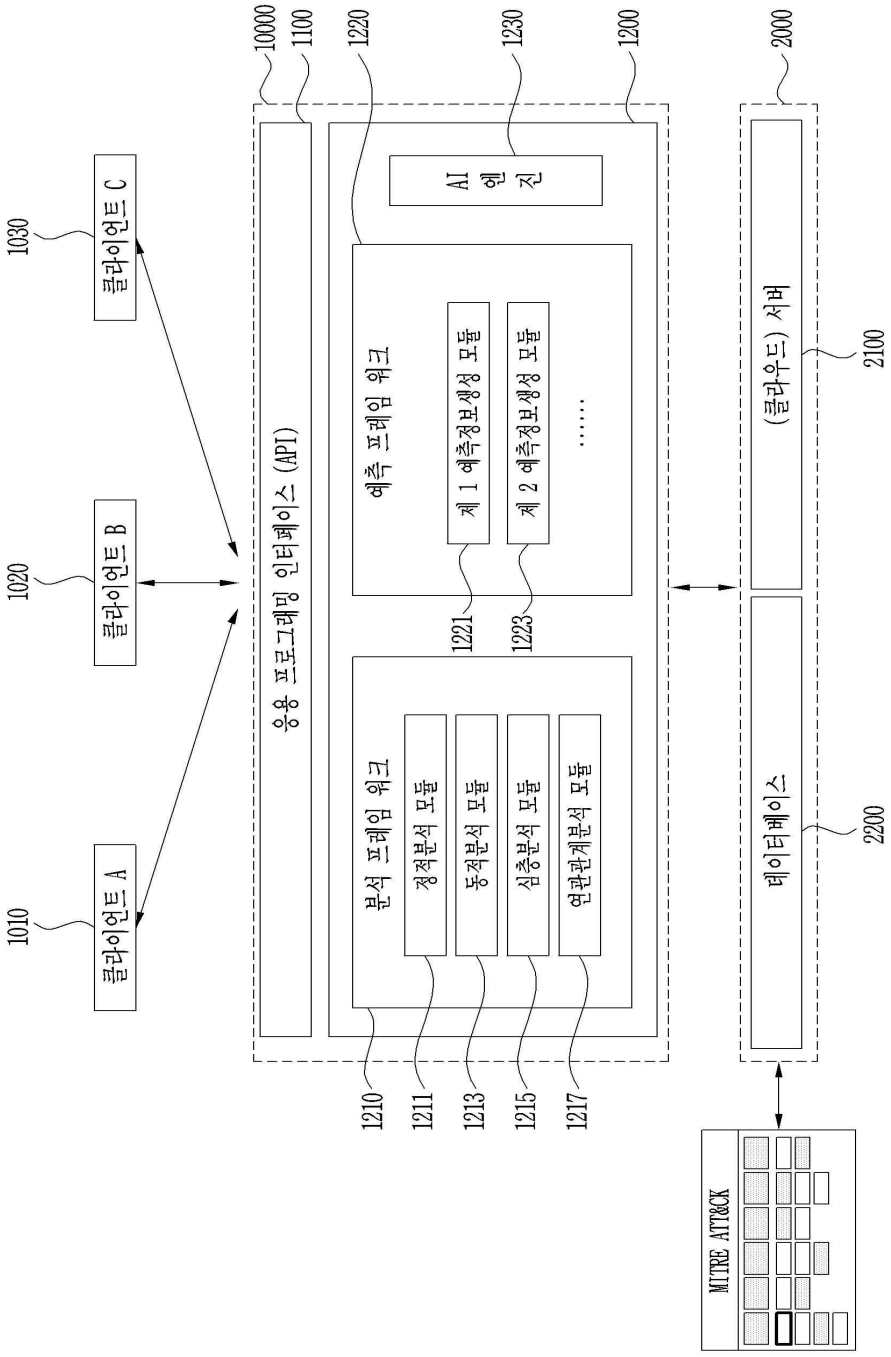
Query (C)

"2019년 1월 ~ 12월에 Anti-Virus 5개 이상 탐지한 파일 중에 EXE_32, EXE_64 파일 타입이면서 유포지가 존재하고 유포지 주소가 "nexon.com"이 아니면서 파일명에 "nexon" 이라는 단어가 들어가며 행위 분석 결과 "방화벽 설정을 수정" 또는 "시작 프로그램에 등록" 행위가 실행되면서 184.55.38.2에 접속 시도를 하는 파일 크기 100KB 이상의 파일

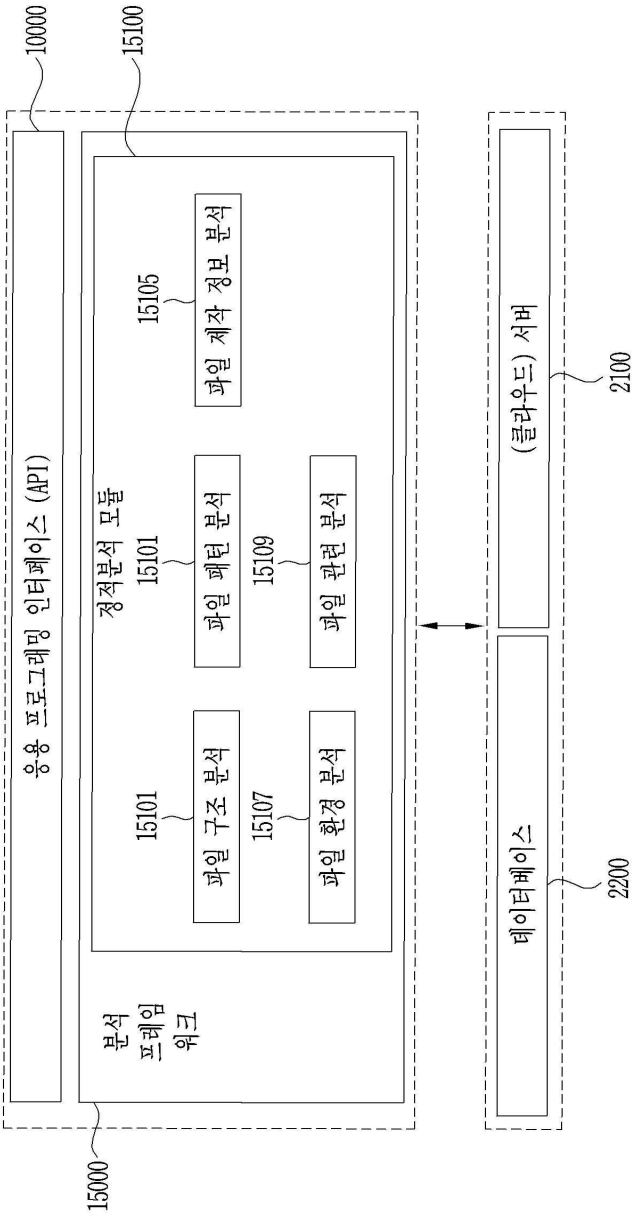
Query (D)

"2019년 1월 ~ 12월에 Anti-Virus 20개 이상 탐지한 파일 중에 문서 (hwp,doc,ppt,xls,pdf)파일 타입이면서 kaspersky의 탐지명에 "not-virus"가 들어가지 않고 유포지 주소가 "*.kr*" 이 들어가는 파일의 매일 통계 정보

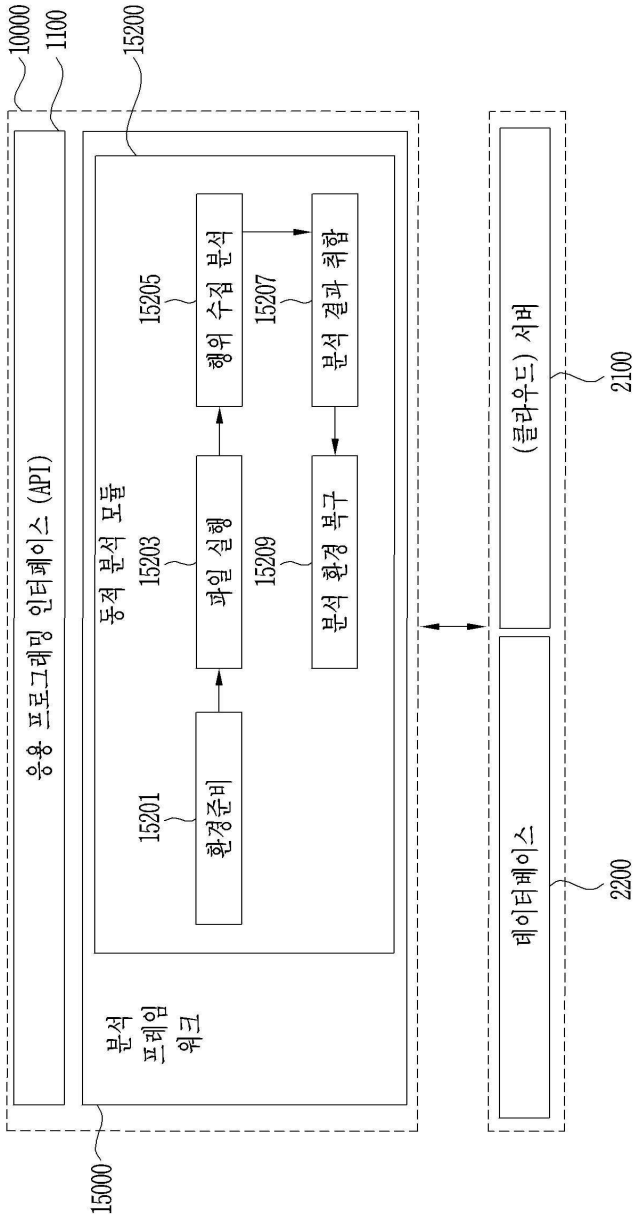
도면10


10000
1100
1220
1230
1200
2000
1030
클라이언트 C
1020
클라이언트 B
1010
클라이언트 A
응용 프로그래밍 인터페이스 (API)
예측 프레임 워크
AI 엔진
제 1 예측정보생성 모듈
제 2 예측정보생성 모듈
.....
1221
1223
분석 프레임 워크
정적분석 모듈
동적분석 모듈
심층분석 모듈
연관관계분석 모듈
1210
1211
1213
1215
1217
(클라우드) 서버
2100
데이터베이스
2200
MITRE ATT&CK

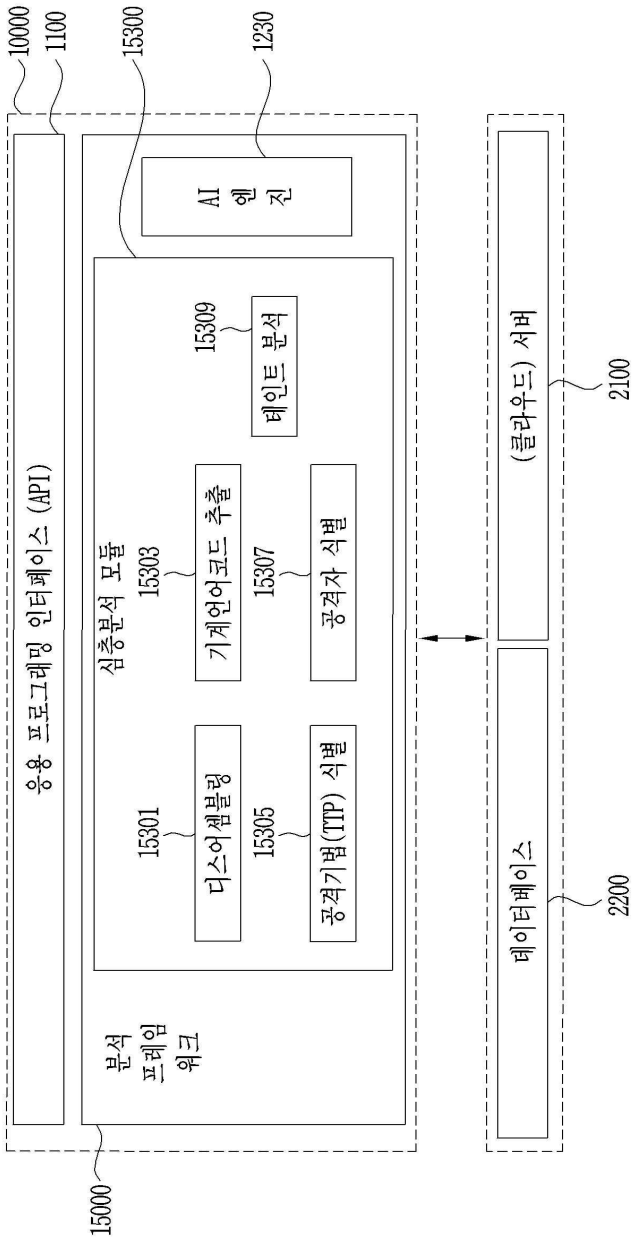
도면11


10000
15100
응용 프로그래밍 인터페이스 (API)
분석 프레임 워크
정적분석 모듈
15101
파일 구조 분석
15101
파일 패턴 분석
15105
파일 제작 정보 분석
15107
파일 환경 분석
15109
파일 관련 분석
15000
(클라우드) 서버
2100
데이터베이스
2200

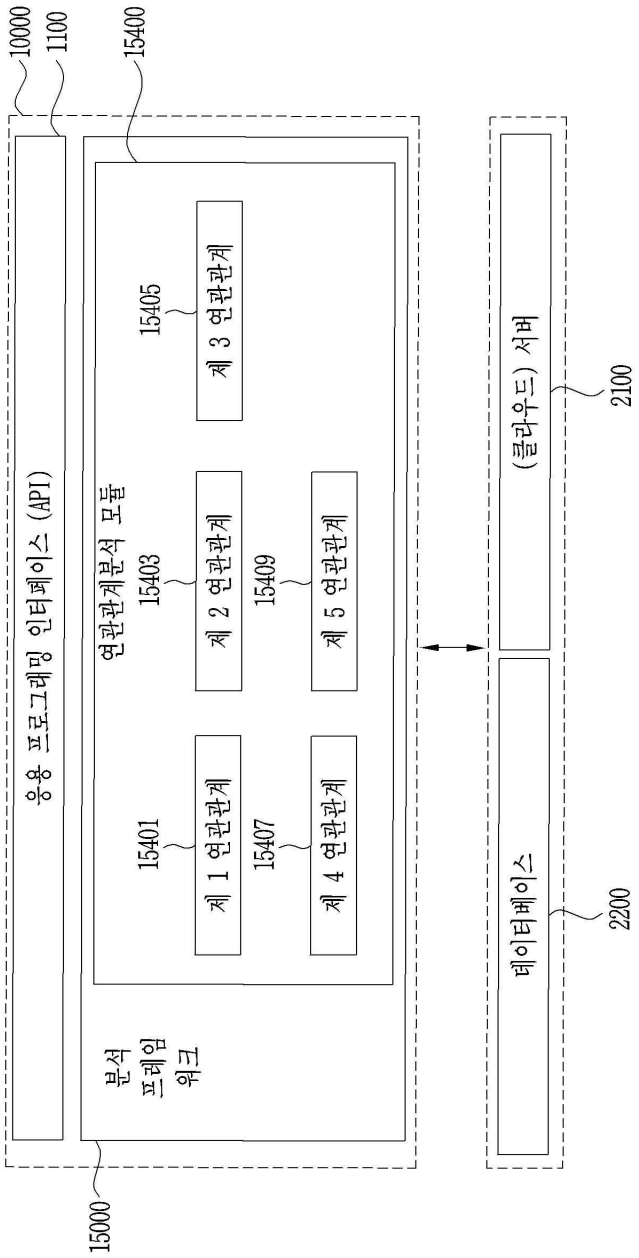
도면12


10000
1100
15200
응용 프로그래밍 인터페이스 (API)
분석 프레임 워크
동적 분석 모듈
15201
환경준비
15203
파일 실행
15205
행위 수집 분석
15207
분석 결과 취합
15209
분석 환경 복구
15000
(클라우드) 서버
2100
데이터베이스
2200

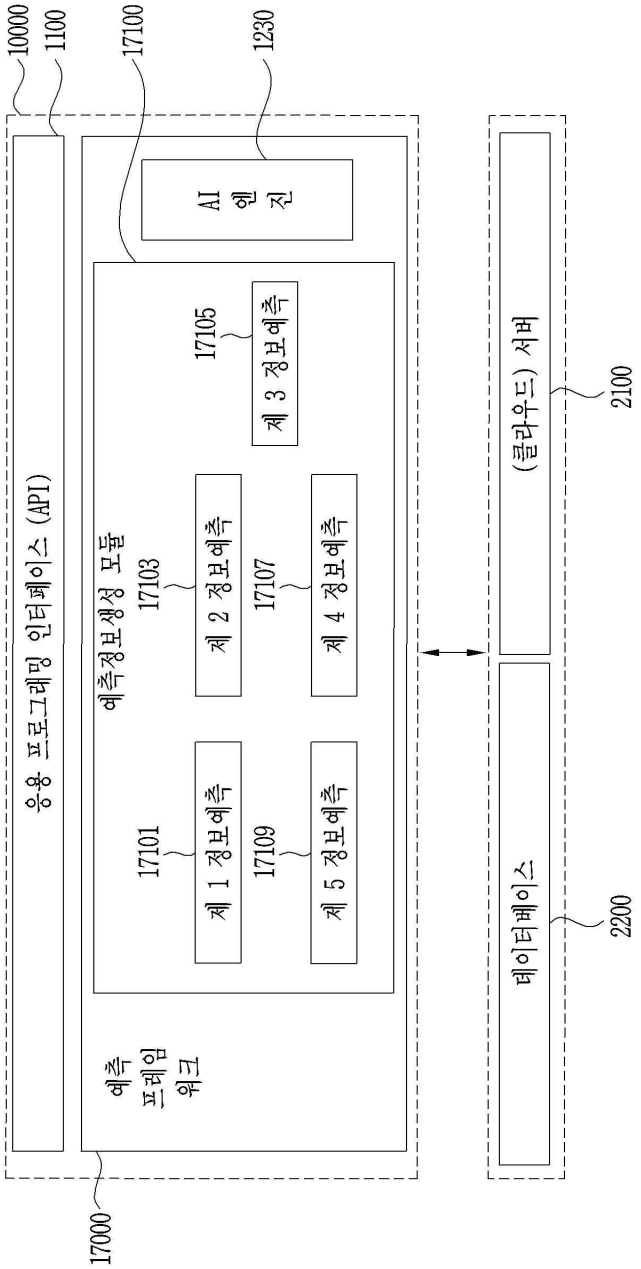
도면13


10000
1100
15300
1230
응용 프로그래밍 인터페이스 (API)
분석 프레임 워크
심층분석 모듈
AI 엔진
15309
테인트 분석
15303
기계언어코드 추출
15307
공격자 식별
15301
디스어셈블링
15305
공격기법(TTP) 식별
15000
(클라우드) 서버
2100
데이터베이스
2200

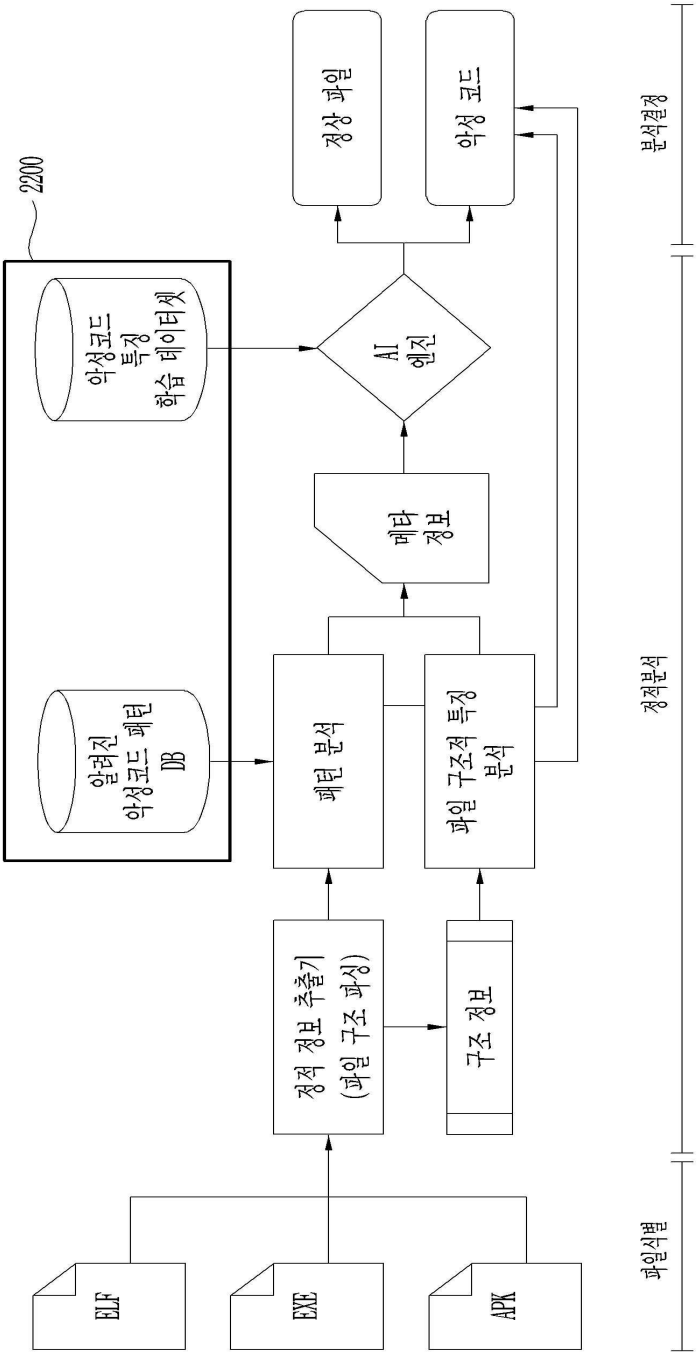
도면14


10000
1100
15400
응용 프로그래밍 인터페이스 (API)
분석 프레임 워크
15000
연관관계분석 모듈
15401
제 1 연관관계
15403
제 2 연관관계
15405
제 3 연관관계
15407
제 4 연관관계
15409
제 5 연관관계
(클라우드) 서버
2100
데이터베이스
2200

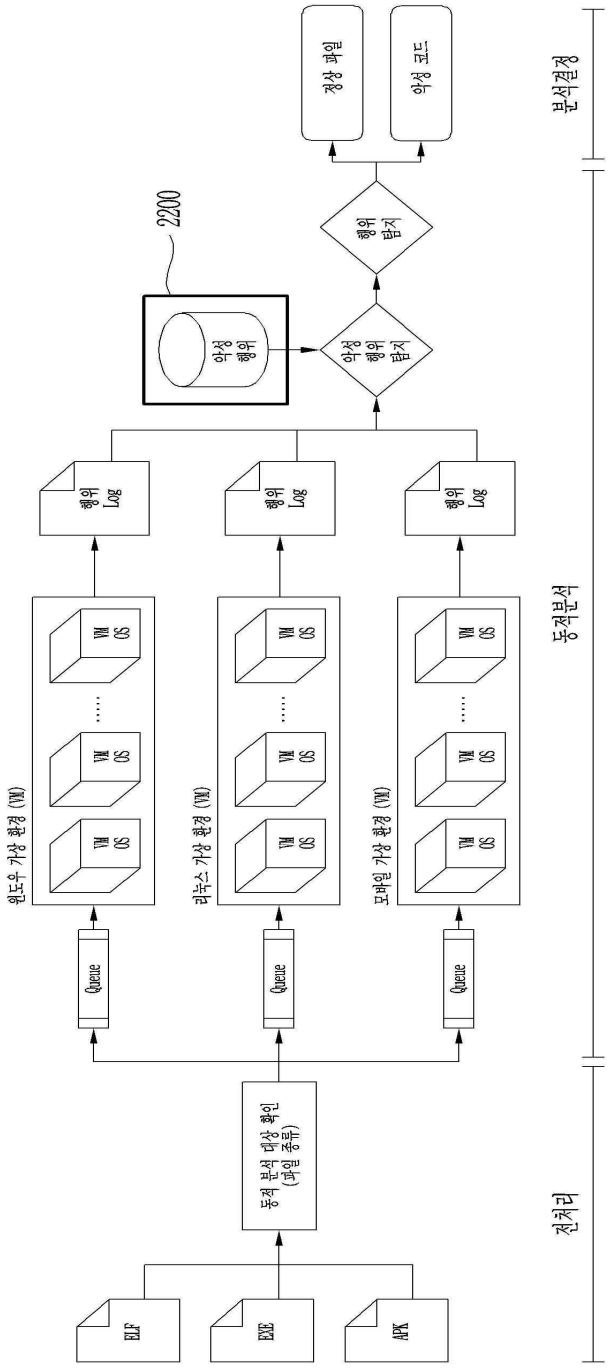
도면15


10000
1100
17100
1230
응용 프로그래밍 인터페이스 (API)
AI 엔진
예측정보생성 모듈
17105
제 3 정보예측
17103
제 2 정보예측
17107
제 4 정보예측
17101
제 1 정보예측
17109
제 5 정보예측
예측 프레임 워크
17000
(클라우드) 서버
2100
데이터베이스
2200

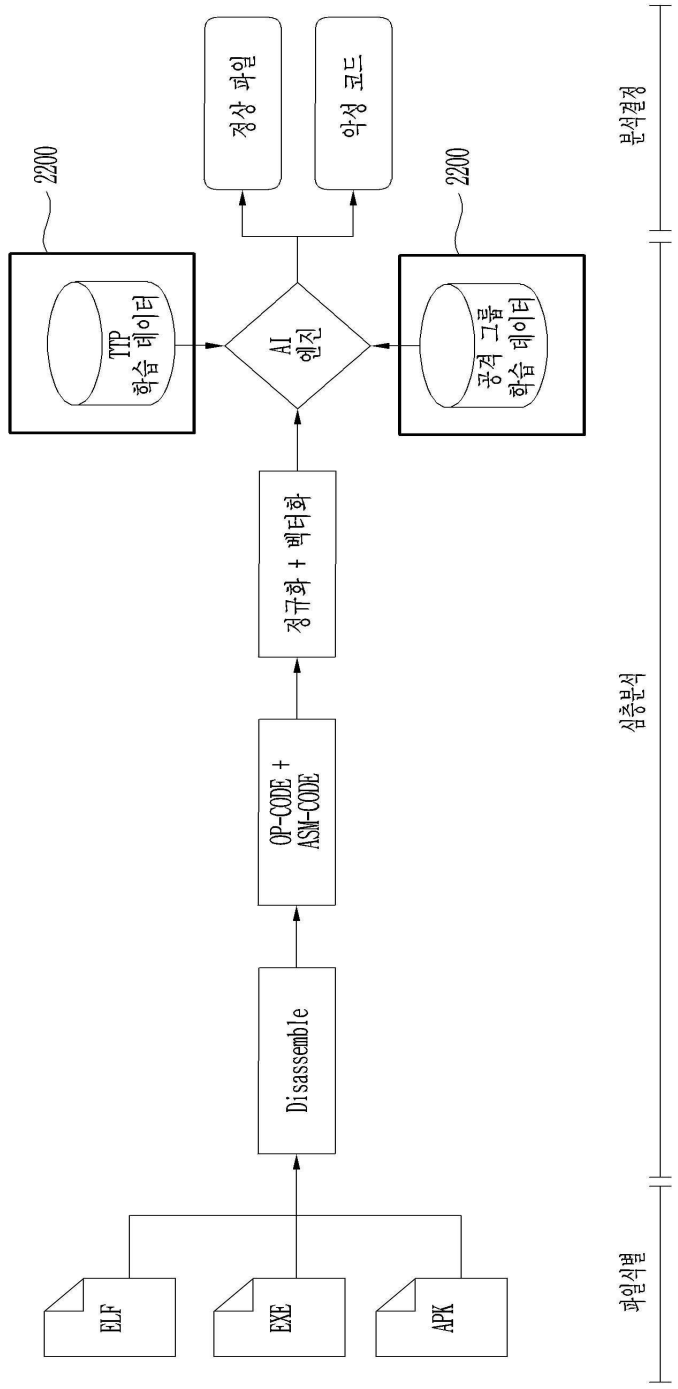
도면16


2200
정상 파일
악성 코드
분석결정
악성코드
특징
학습 데이터셋
AI
엔진
메타
정보
알려진
악성코드 패턴
DB
패턴 분석
파일 구조적 특징
분석
정적분석
정적 정보 추출기
(파일 구조 파싱)
구조 정보
ELF
EXE
APK
파일식별

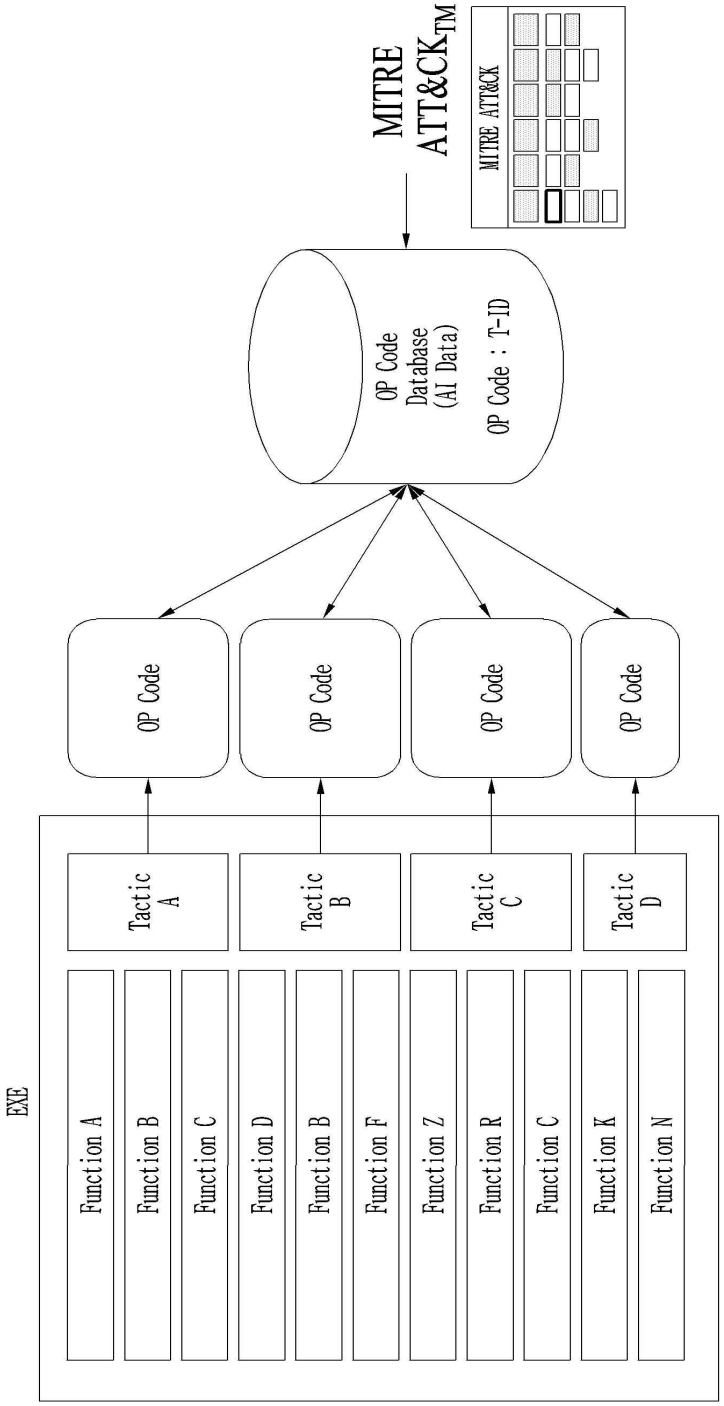
도면17


전처리
동적분석
분석결정
ELF
EXE
APK
동적 분석 대상 확인
(파일 종류)
Queue
Queue
Queue
윈도우 가상 환경 (VM)
리눅스 가상 환경 (VM)
모바일 가상 환경 (VM)
VM
OS
행위
Log
2200
악성
행위
악성
행위
탐지
행위
탐지
정상 파일
악성 코드

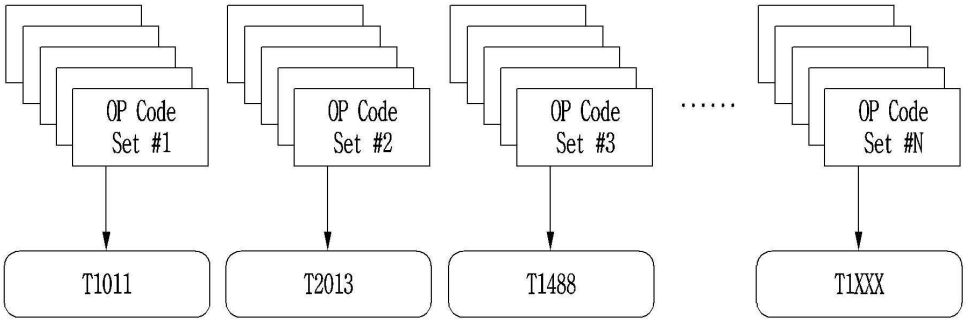
도면18


파일식별
심층분석
분석결정
ELF
EXE
APK
Disassemble
OP-CODE + ASM-CODE
정규화 + 벡터화
2200
TTP 학습 데이터
AI 엔진
2200
공격 그룹 학습 데이터
정상 파일
악성 코드

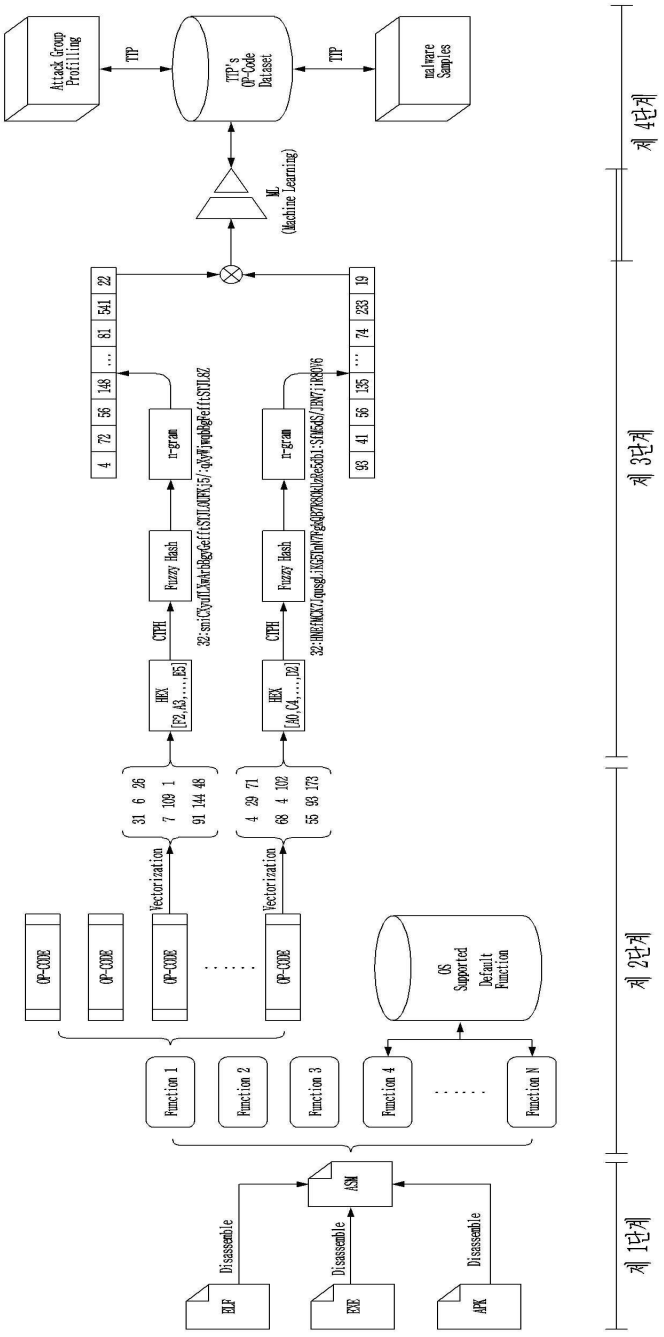
도면19


MITRE ATT&CK™
MITRE ATT&CK
OP Code Database (AI Data)
OP Code : T-ID
OP Code
OP Code
OP Code
OP Code
Tactic A
Tactic B
Tactic C
Tactic D
EXE
Function A
Function B
Function C
Function D
Function B
Function F
Function Z
Function R
Function C
Function K
Function N

도면20


OP Code
Set #1
OP Code
Set #2
OP Code
Set #3
.....
OP Code
Set #N
T1011
T2013
T1488
T1XXX

도면21


제 4단계
제 3단계
제 2단계
제 1단계
Attack Group Profiling
TTP
TTP's OP-Code Dataset
TTP
malware Samples
ML (Machine Learning)
n-gram
Fuzzy Hash
CTPH
HEX
Vectorization
OP-CODE
OS Supported Default Function
Function 1
Function 2
Function 3
Function 4
Function N
ASM
Disassemble
DLP
EXE
APK

도면22

OP-CODE`	OP-CODE → CRC-16	ASM-CODE	ASM-CODE → CRC-32
push	0x45E9	55	0xC9034AF6
mov	0x10E3	8B EC	0x3012FD2C
lea	0xAACE	8D 45 0C	0x9214A6AA
push	0x45E9	50	0xB969BE79
push	0x45E9	6A 00	0xDECB91D2
push	0x45E9	FF 75 08	0x1D4AE87E
push	0x45E9	6A 01	0xA9CCA144
call	0x09F8	FF 15 B0 20 40 00	0x284055E2
pop	0x7117	59	0xC0B506DD
push	0x45E9	50	0xB969BE79
call	0x09F8	E8 AB FF FF FF	0x4452C315
add	0x8B0B	83 C4 10	0xAE137EE5
pop	0x7117	5D	0xC7D8C2C4
retn	0x12B3	C3	0xD06F7C87

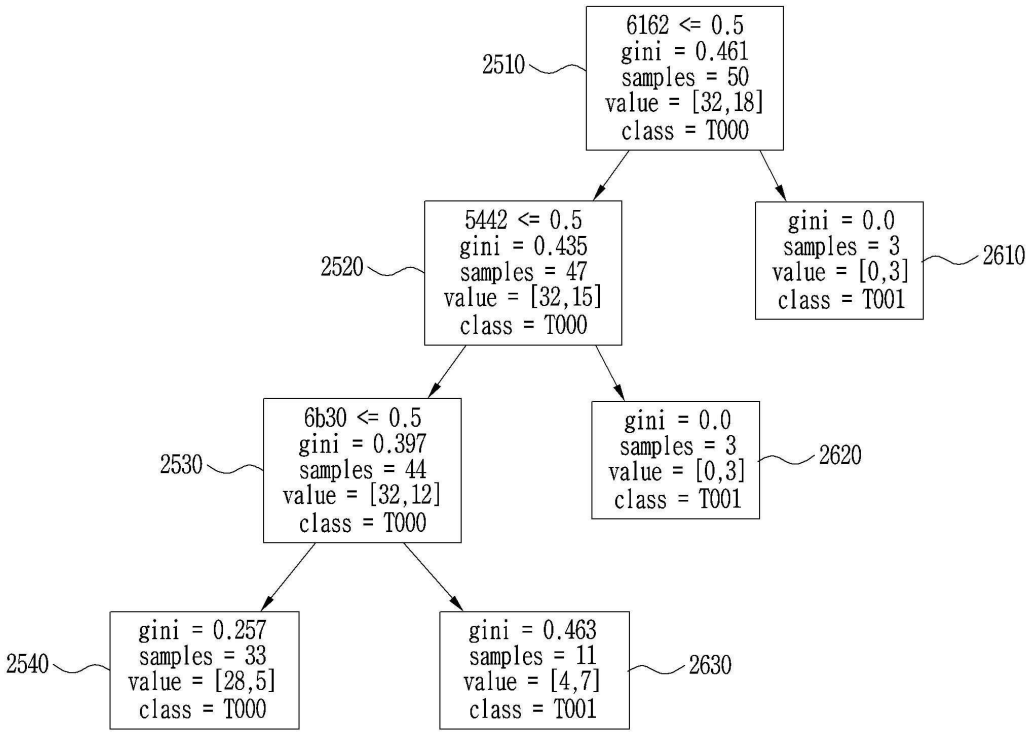
도면23

OP-CODE Vector	ASM-CODE Vector
17897	201 3 74 246
4323	48 18 253 44
43726	146 20 166 170
17897	185 105 121 44
17897	222 203 145 210
17897	29 74 232 126
17897	169 204 161 68
2552	40 64 85 226
28951	192 181 6 221
17897	185 105 190 121
2552	68 82 195 21
35595	174 19 126 229
28951	199 216 194 196
4787	208 111 124 135

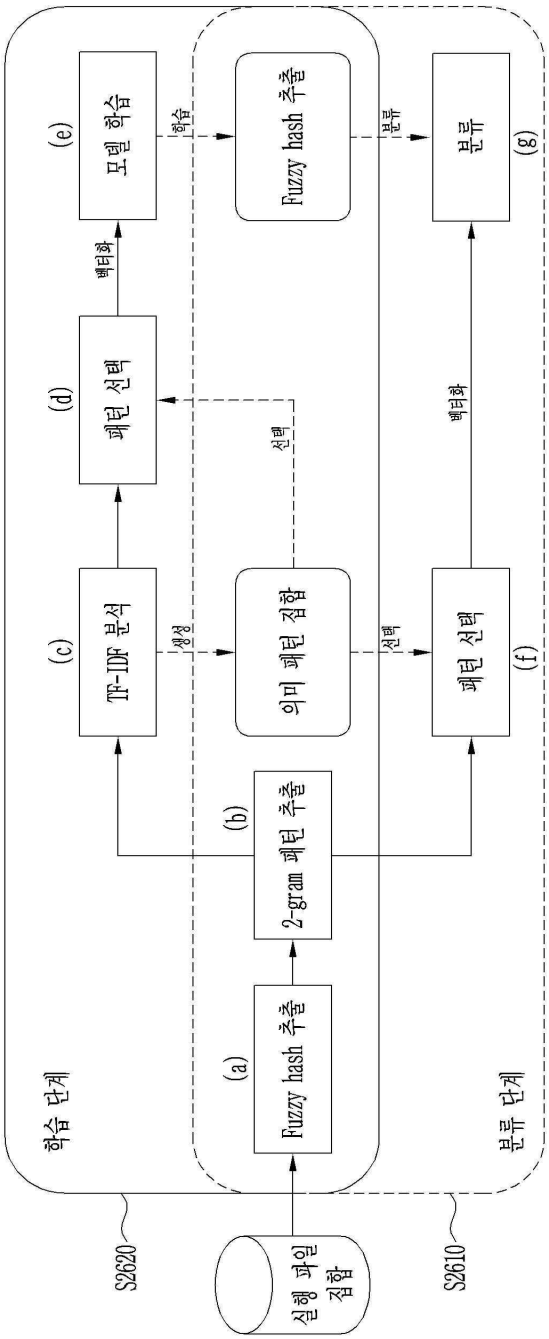
도면24

Fixed block	content	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	0	1
Original	ascii val	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	48	49
	ascii sum	394					410				426				442				458				474				340		
KaqGKaU	b64 char	K					a				q				6				K				a				U		

도면25


6162 <= 0.5
gini = 0.461
samples = 50
value = [32,18]
class = T000
2510
5442 <= 0.5
gini = 0.435
samples = 47
value = [32,15]
class = T000
2520
gini = 0.0
samples = 3
value = [0,3]
class = T001
2610
6b30 <= 0.5
gini = 0.397
samples = 44
value = [32,12]
class = T000
2530
gini = 0.0
samples = 3
value = [0,3]
class = T001
2620
gini = 0.257
samples = 33
value = [28,5]
class = T000
2540
gini = 0.463
samples = 11
value = [4,7]
class = T001
2630

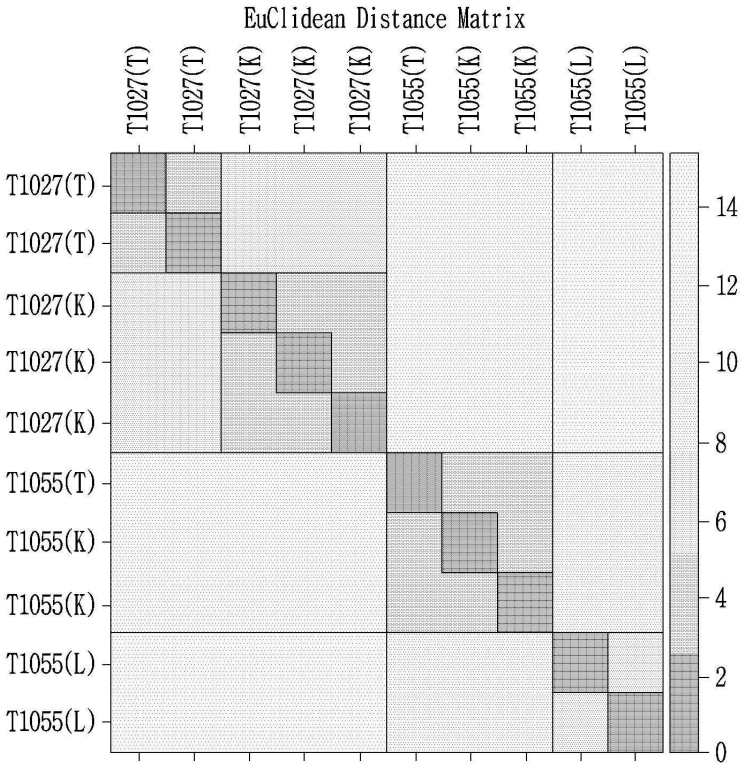
도면26


학습 단계
분류 단계
실행 파일 집합
S2620
S2610
(a)
Fuzzy hash 추출
(b)
2-gram 패턴 추출
(c)
TF-IDF 분석
(d)
패턴 선택
(e)
모델 학습
의미 패턴 집합
패턴 선택
(f)
Fuzzy hash 추출
분류
(g)
생성
선택
벡터화
학습
선택
벡터화
분류

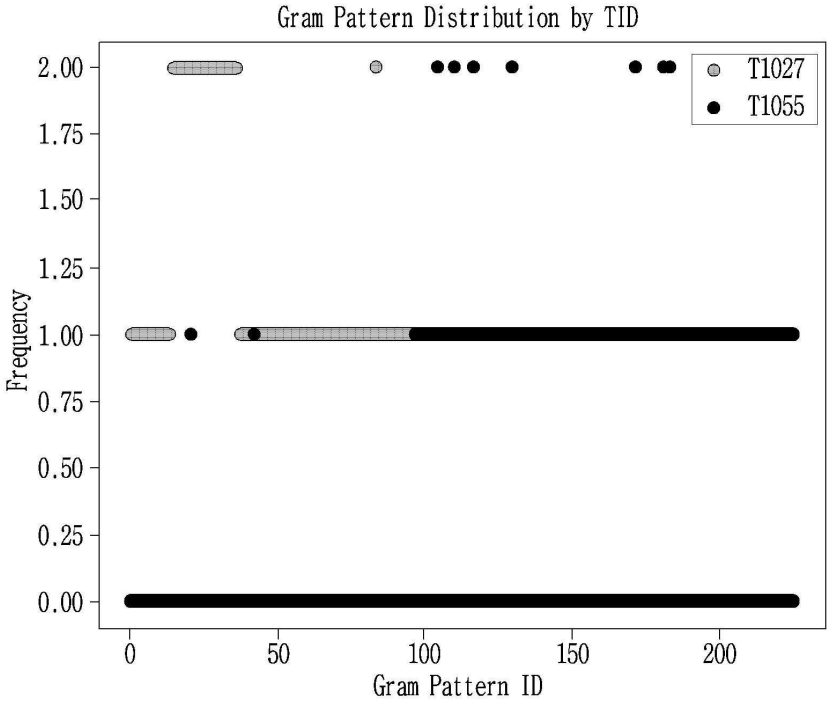
도면27

Label		SHA-256 (size)	N-gram
T-ID	Attacker (or Group)	(OP-CODE + ASM-CODE)'s Fuzzy Hash	
T10XX	TA504	389E03B1A1FD1C5....	{"3736":1,"3645":1, "4563":1, "6344":1, "4472":1,"7255":1 .....
		32:76EcDrURBPQk58t...	
	TA504	E3EC2AA04AFECC6...	{"3736":1,"3645":1, "4563":1, "6344":1, "4472":1,"7255":1, "5552":1,"5242":1, "4250":1.....
		32:76EcDrURBPQk58tGcoHE...	

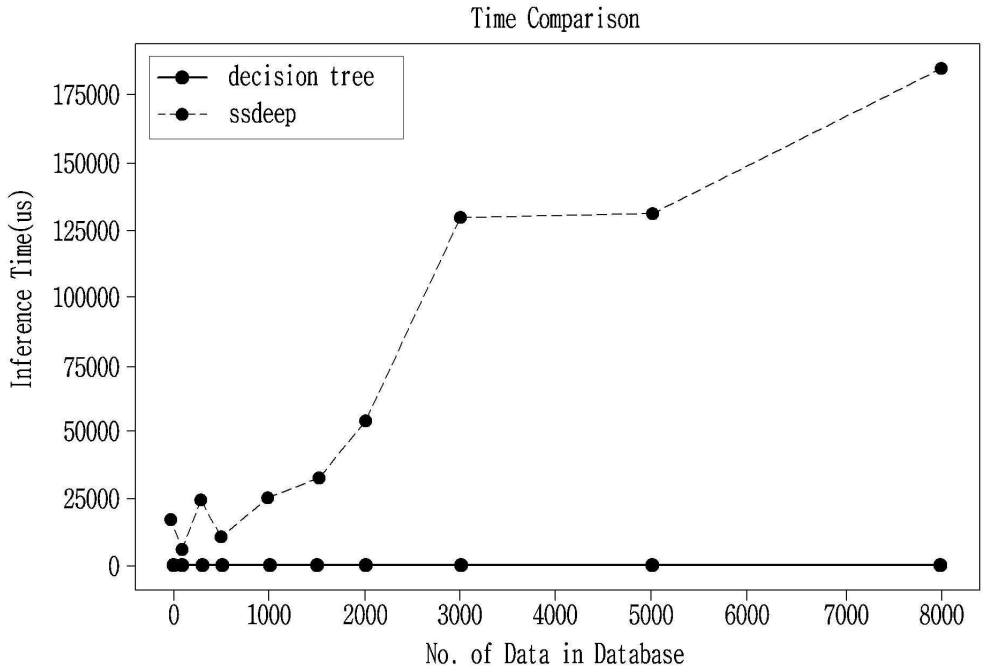
도면28



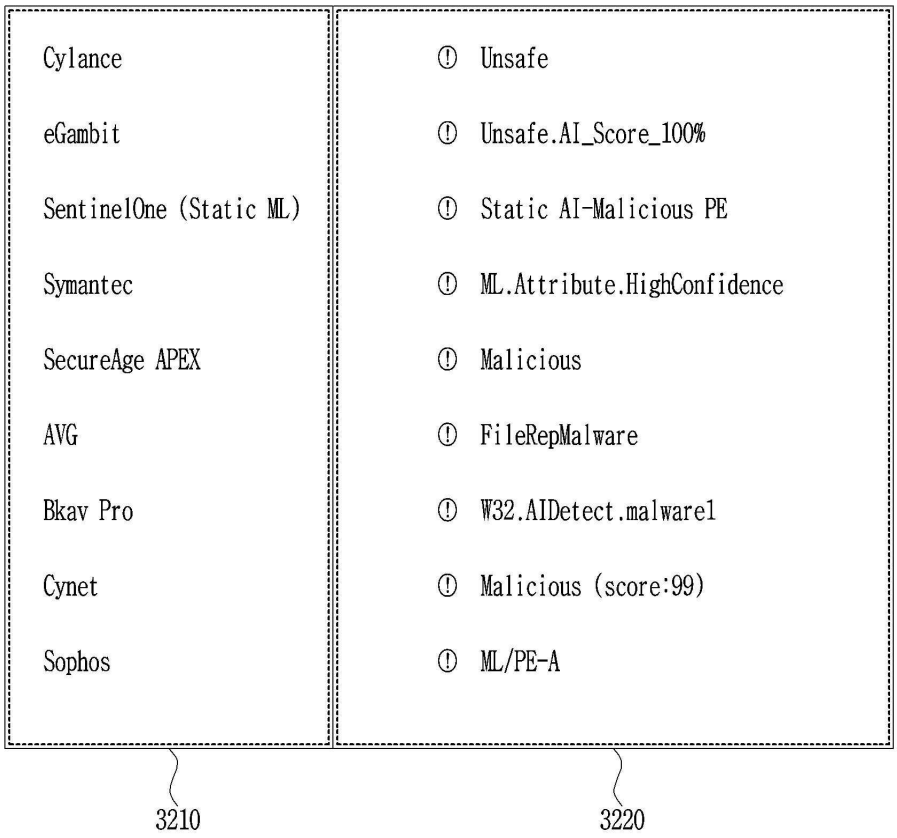
도면29


Gram Pattern Distribution by TID
T1027
T1055
Frequency
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
50
100
150
200
Gram Pattern ID

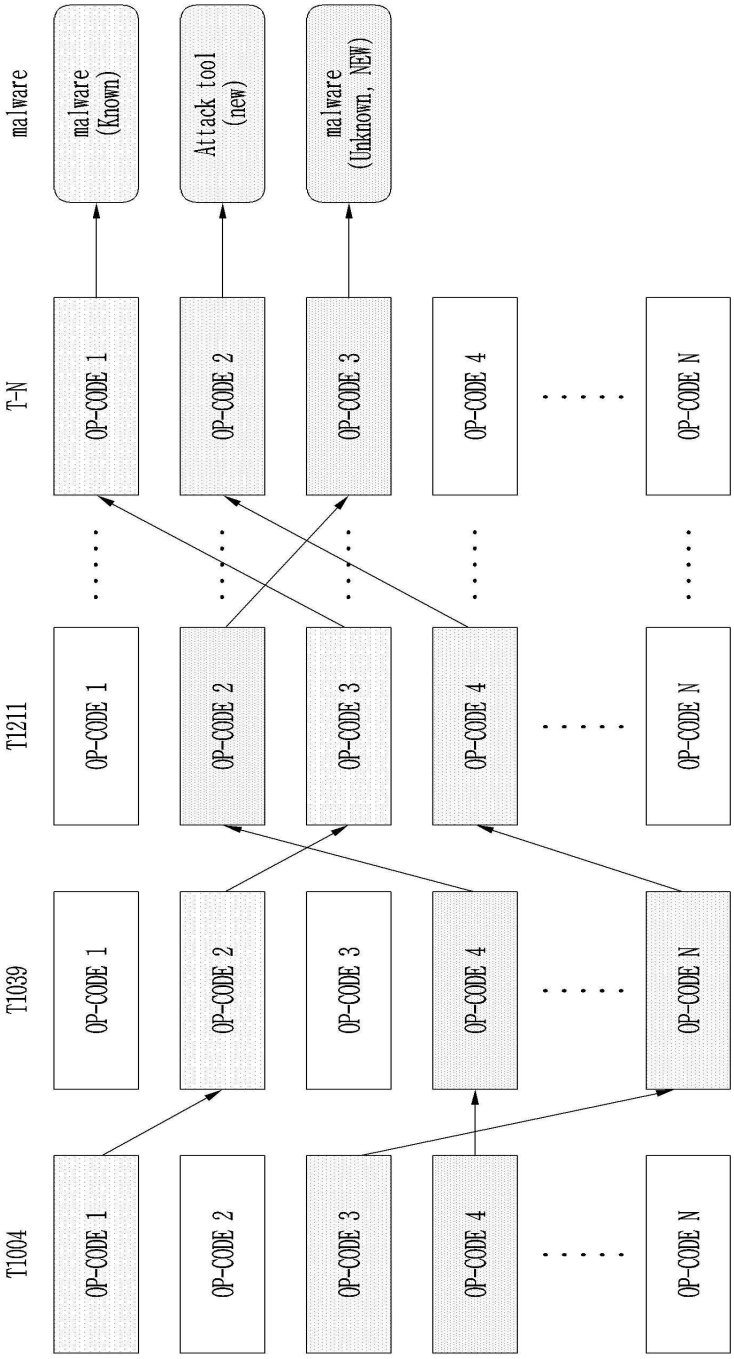
도면30


Time Comparison
decision tree
ssdeep
Inference Time(us)
175000
150000
125000
100000
75000
50000
25000
0
0
1000
2000
3000
4000
5000
6000
7000
8000
No. of Data in Database

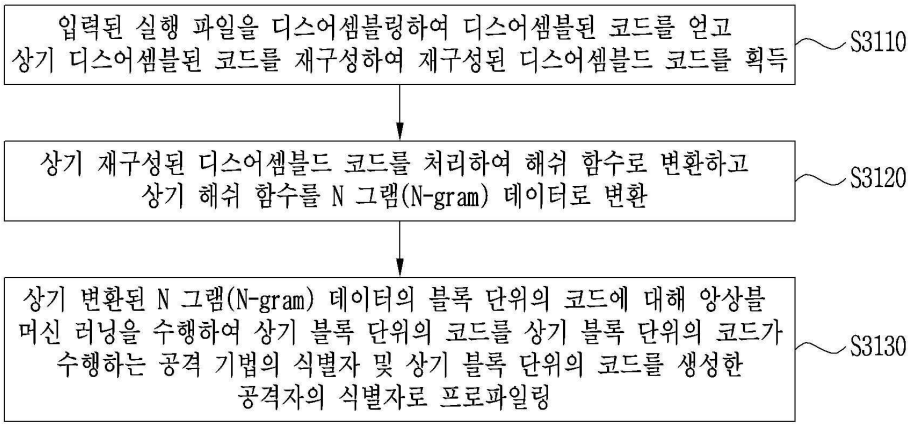
도면31


Cylance
Unsafe
eGambit
Unsafe.AI_Score_100%
SentinelOne (Static ML)
Static AI-Malicious PE
Symantec
ML.Attribute.HighConfidence
SecureAge APEX
Malicious
AVG
FileRepMalware
Bkav Pro
W32.AIDetect.malware1
Cynet
Malicious (score:99)
Sophos
ML/PE-A
3210
3220

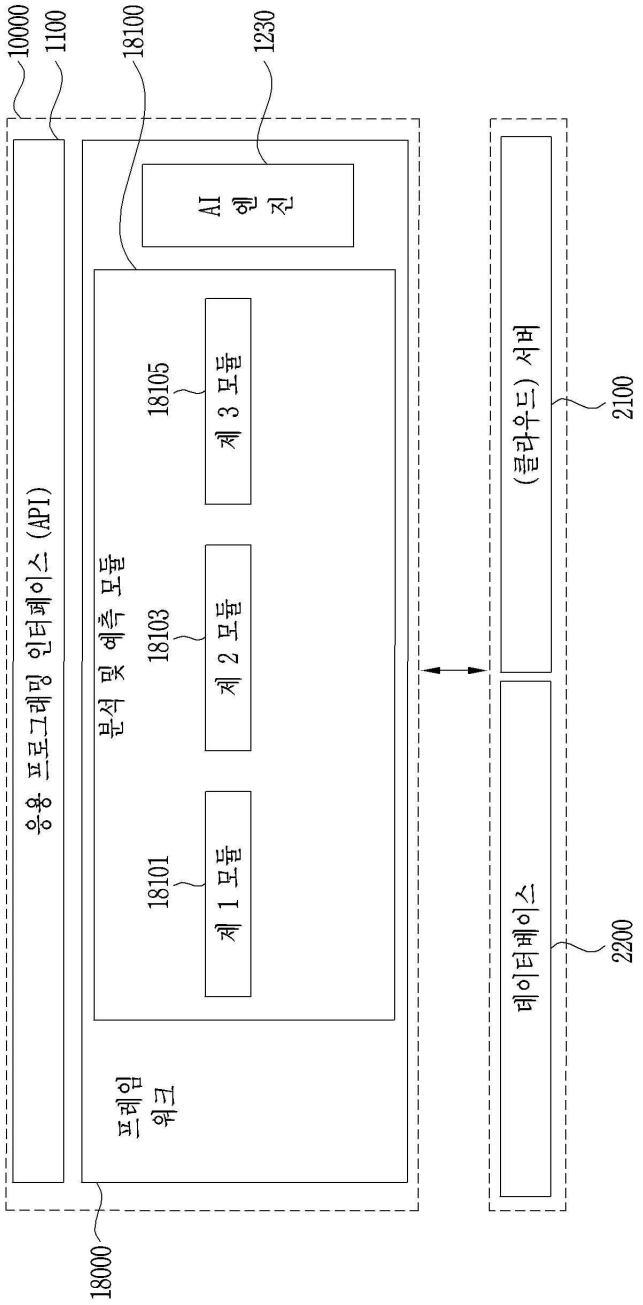
도면32


malware
T-N
T1211
T1039
T1004
malware (Known)
Attack tool (new)
malware (Unknown, NEW)
OP-CODE 1
OP-CODE 2
OP-CODE 3
OP-CODE 4
OP-CODE N

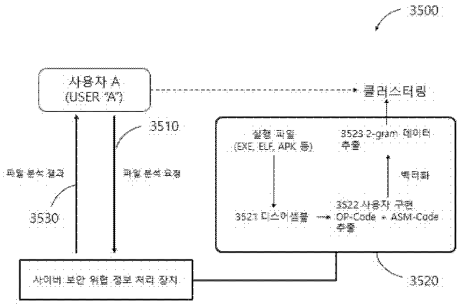
도면33


입력된 실행 파일을 디스어셈블링하여 디스어셈블된 코드를 얻고
상기 디스어셈블된 코드를 재구성하여 재구성된 디스어셈블드 코드를 획득
S3110
상기 재구성된 디스어셈블드 코드를 처리하여 해쉬 함수로 변환하고
상기 해쉬 함수를 N 그램(N-gram) 데이터로 변환
S3120
상기 변환된 N 그램(N-gram) 데이터의 블록 단위의 코드에 대해 앙상블
머신 러닝을 수행하여 상기 블록 단위의 코드를 상기 블록 단위의 코드가
수행하는 공격 기법의 식별자 및 상기 블록 단위의 코드를 생성한
공격자의 식별자로 프로파일링
S3130

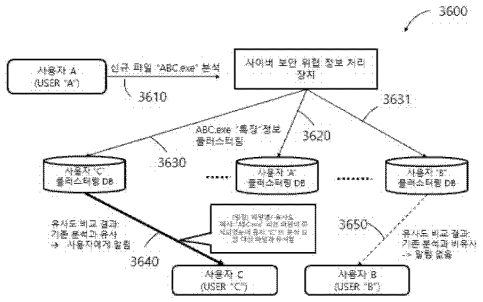
도면34


10000
1100
18100
1230
AI 엔진
18105
제 3 모듈
분석 및 예측 모듈
18103
제 2 모듈
18101
제 1 모듈
프레임 워크
응용 프로그래밍 인터페이스 (API)
18000
(클라우드) 서버
2100
데이터베이스
2200

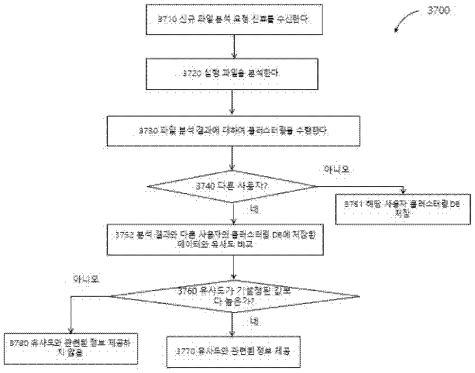
도면35


3500
사용자 A
(USER "A")
클러스터링
3510
파일 분석 결과
파일 분석 요청
3530
실행 파일
(EXE, ELF, APK 등)
3523 2-gram 데이터 추출
벡터화
3521 디스어셈블
3522 사용자 구현 OP-Code + ASM-Code 추출
3520
사이버 보안 위협 정보 처리 장치

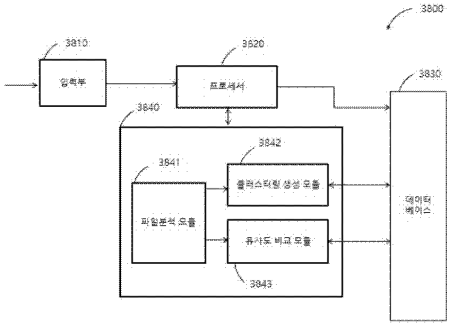
도면36



도면37



도면38



도면39