악성 프로그램 대응 기술

System Software Research Lab

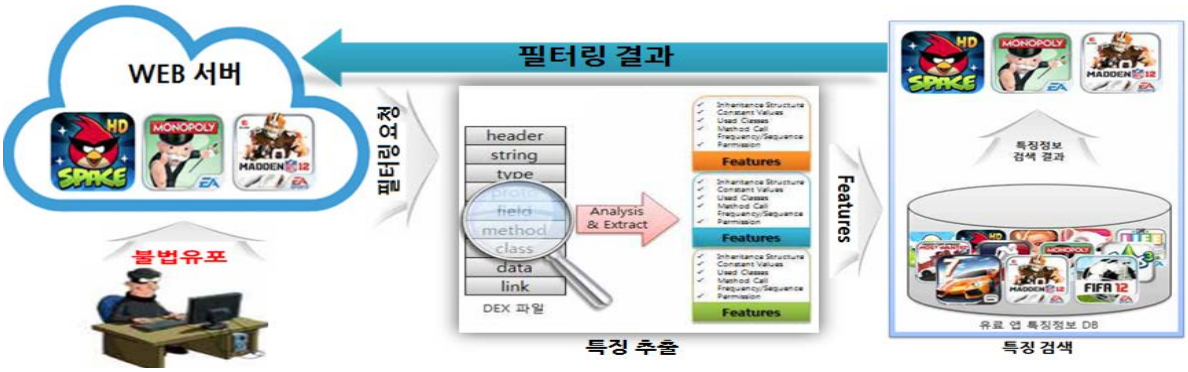
- **주요 연구분야**
  - **빅 데이터** 분석을 통한 **정보 보안 및 APP 불법 복제 방지** 관련 **연구 집중**
  - 빅 데이터 분석 기반 랜섬웨어 패턴 분석을 토대로 **미끼 파일을 이용한 악성 프로그램 차단 기술, 백업 파일 악성 프로그램 감염 차단 기술** 및 화이트리스트 구축 기술 **등 개발**

1. 악성 프로그램 차단 기술

- 빅 데이터 분석을 통해 랜섬웨어(Ransomware)와 같은 **악성 프로그램의 침입 및 감염 패턴**을 분석
- **미끼 파일(Decoy)을 이용하여 랜섬웨어를 효과적으로 탐지**하고 시스템 감염 확인이 가능
- **화이트리스트**를 이용하여 악성 코드 검출에 소요되는 **시간을 획기적으로 단축**

2. 스마트폰 앱 불법 복제 방지 기술

- 안드로이드 앱 실행 환경에 최적화된 워터마킹, 포렌식 마킹 기술을 기반으로 스마트폰 내의 불법 앱을 탐지하고, 특징 정보 기반 불법 앱 필터링



[특징정보 기반 불법 APP 필터링]

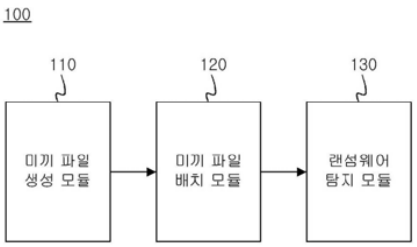
특허 정보

- **악성 프로그램 감염을 차단하는 파일 백업 장치 및 그 방법과 랜섬웨어 탐지 방법, 이를 수행하기 위한 기록매체 및 랜섬웨어 탐지 시스템과 악성코드 검출을 위한 화이트리스트 구축 방법 및 이를 수행하기 이한 기록매체(출원 제 10-2018-0066884 호, 출원 제 10-2017-0182897 호, 출원 제 10-2017-0182896호)**

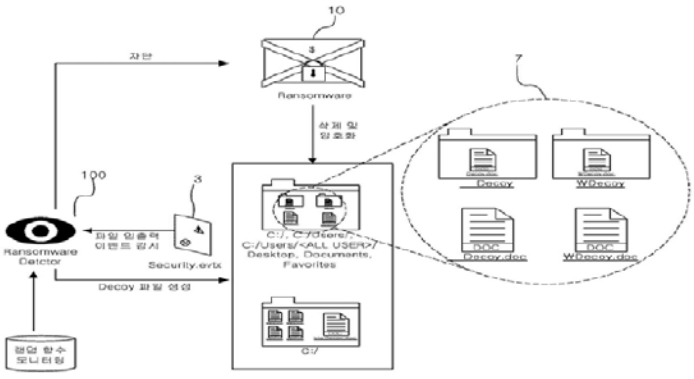
기술 개요

- **본 기술은 미끼 파일(Decoy)을 이용하여 악성 프로그램을 침입을 감지하고, 효율적으로 차단하기 위한 소프트웨어 기술**

- 랜섬웨어의 침입 패턴을 분석한 결과를 토대로 미끼 파일을 생성하고 배치하여 효율적으로 랜섬웨어 침입 탐지

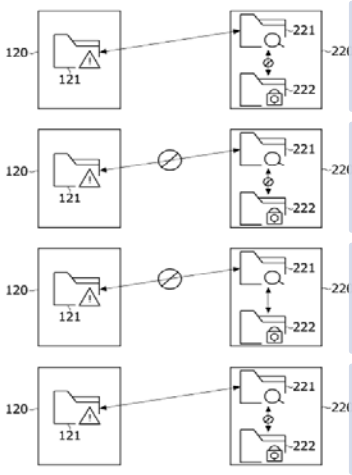


[랜섬웨어 탐지 시스템]

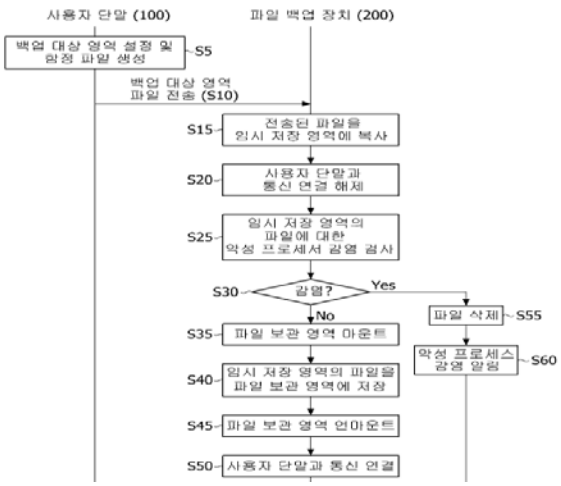


[랜섬웨어 탐지 동작 흐름도]

- **미끼 파일을 이용하여 백업 파일의 악성 프로그램 감염 확인 후 파일 백업 결정**



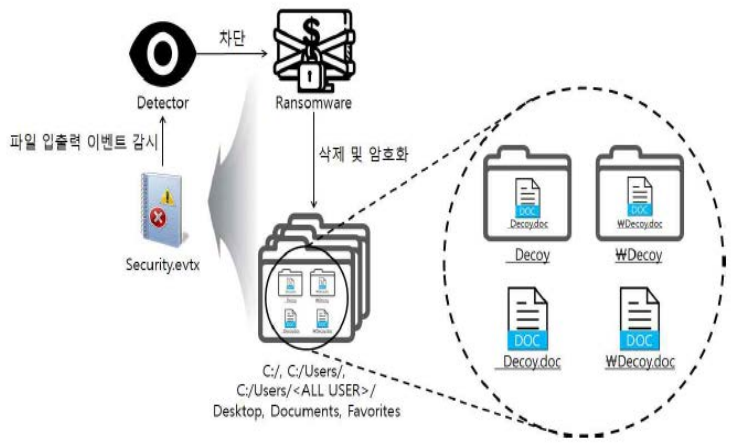
- 백업 대상 영역과 임시 저장 영역 연결
- 백업 대상 영역과 임시 저장 영역 연결 해제
- 악성 프로세스 감염 없는 경우, 임시 저장 영역과 파일 보관 영역 연결
- 임시 저장 영역과 파일 보관 영역 연결 해제



기술 특장점

▪ 기존 랜섬웨이 및 신종 랜섬웨이 탐지를 위한 효율적인 미끼 파일 생성 및 배치

- Windows-1252 순서 또는 역순으로 파일을 탐색하는 랜섬웨어 패턴을 이용하여 미끼 파일을 생성, 배치하여 효율적으로 랜섬웨어 탐지



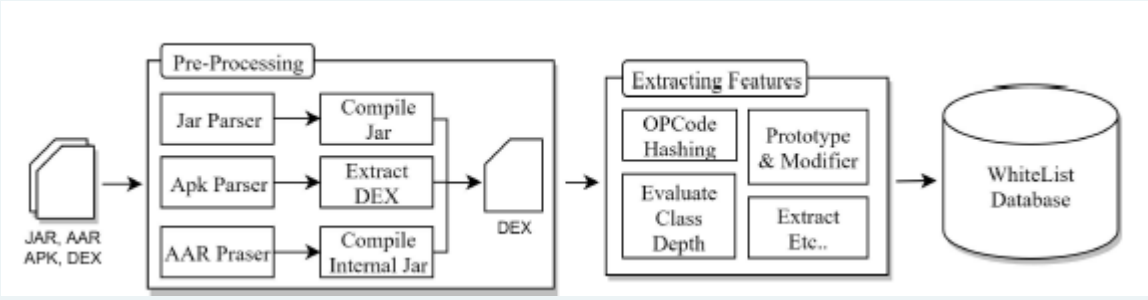
[랜섬웨어 탐지를 위한 미끼 파일]

```
1: procedure MakeDecoyFiles(smallDecoy, bigDecoy, n)
2:    userNames = GetSystemUserName(ALLUSER)
3:    sw = GetStartOfWord();
4:    ew = GetEndOfWord();
5:    startFolder = {"C:/", "C:/Users/"}
6:    for each name in userNames do
7:       startFolder+="C:/Users/"+name+"/Desktop/"
8:       startFolder+="C:/Users/"+name+"/Documents/"
9:       startFolder+="C:/Users/"+name+"/Favorites/"
10:
11:   for each path in startFolder do
12:      MakeFolder(path+sw)
13:      MakeFolder(path+ew)
14:
15:      for i = 1 to n do
16:         MakeDecoyFile(path+sw+i+".doc",smallDecoy)
17:         MakeDecoyFile(path+ew+i+".doc",smallDecoy)
18:         MakeDecoyFile(path+sw+"/"+sw+i+".doc",smallDecoy)
19:         MakeDecoyFile(path+ew+"/"+ew+i+".doc",smallDecoy)
20:
21:      MakeDecoyFile(path+sw+"big"+i+".doc", bigDecoy)
22:
23:   MakeDecoyFile("C:/"+sw+"/Recent.doc", bigDecoy)
24:   new Thread(RegularlyUpdateDecoyFile,
25:      "C:/"+sw+"/Recent.doc")
26:
27:   new Thread(RandomizeMonitoring)
```

[미끼 파일 배치 의사코드]

▪ 3rd-party 라이브러리에 대한 화이트리스트로 악성 코드 검출 시간 단축

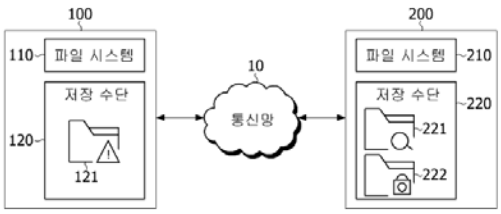
- 클래스 난독화 기법을 적용한 후 클래스의 깊이 정보, 접근 제어자(access flag), 메소드 프로토타입의 인수 개수, 연산 코드의 해시값 등 변경 불가능한 정보를 특징정보로 추출, 화이트리스트 생성



[화이트리스트 구축 과정]

▪ 백업 시스템의 악성 프로그램 감염 방지

- 백업 시스템 내의 임시 저장 영역에 악성 프로세스 감염이 없을 때만 임시 저장 영역(221)과 파일 보관 영역(222)를 연결함으로써 백업 시스템에 악성 프로세스 감염을 방지

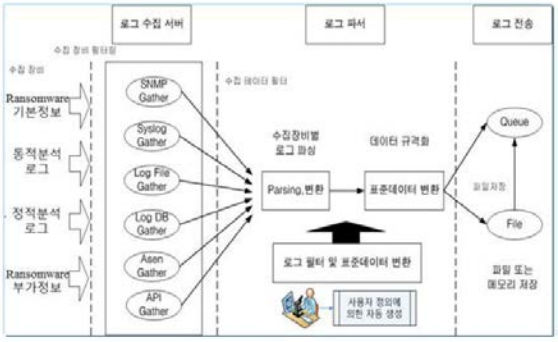


[파일 백업 시스템 구성도]

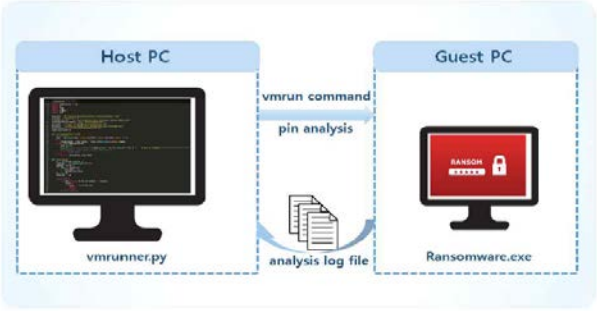
기술개발 현황

▪ 빅데이터를 활용한 랜섬웨어 분석 모듈 개발

- 빅데이터 시스템에 정량화된 랜섬웨어 특징정보를 자동으로 추출하여 수집하고, Cuckoo Sandbox를 활용하여 정적/동적 분석



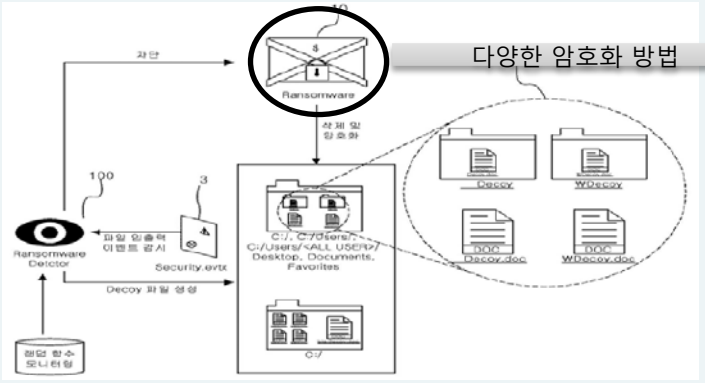
[랜섬웨어 특징 정보 수집 구성도]

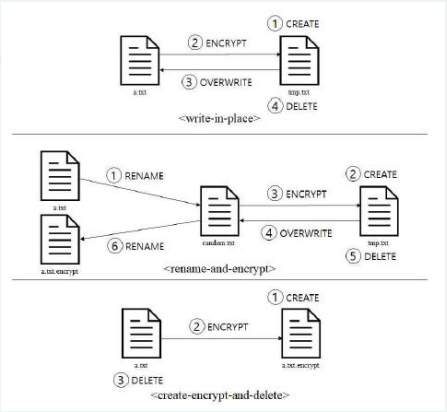


[랜섬웨어 특징정보 추출 자동화 모듈 동작도]

▪ 미끼 파일 감시 기반 랜섬웨어 감시 모듈 개발

- 랜섬웨어 패턴 분석을 통한 랜섬웨어 탐지를 위한 미끼 파일을 생성, 특정 프로세스가 미끼 파일을 변경 및 삭제시 랜섬웨어로 간주

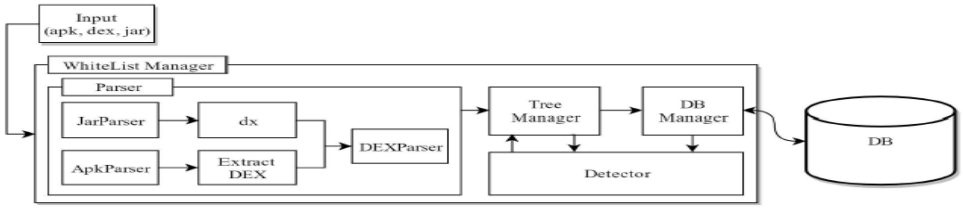




[랜섬웨어 감시 모듈 동작도]

▪ 화이트리스트 분석 GUI 도구 구현

- 라이브러리 정보, 패키지 정보, 패키지 내의 클래스 정보, 메소드 정보를 기반으로 화이트리스트 DB를 구축하고, 화이트리스트 DB를 이용하여 화이트리스트 분석



[화이트리스트 매니저 구조]

종래 기술 대비 우수성

종래기술

- 루트 경로를 탐색하지 않은 랜섬웨어 탐지 불가능
- 신종 랜섬웨어 탐지 불가능
- API 분석 과정에 상당한 시간이 소비
- 클라우드 자체의 랜섬웨어 감염으로 인한 파일 감염에 무방비

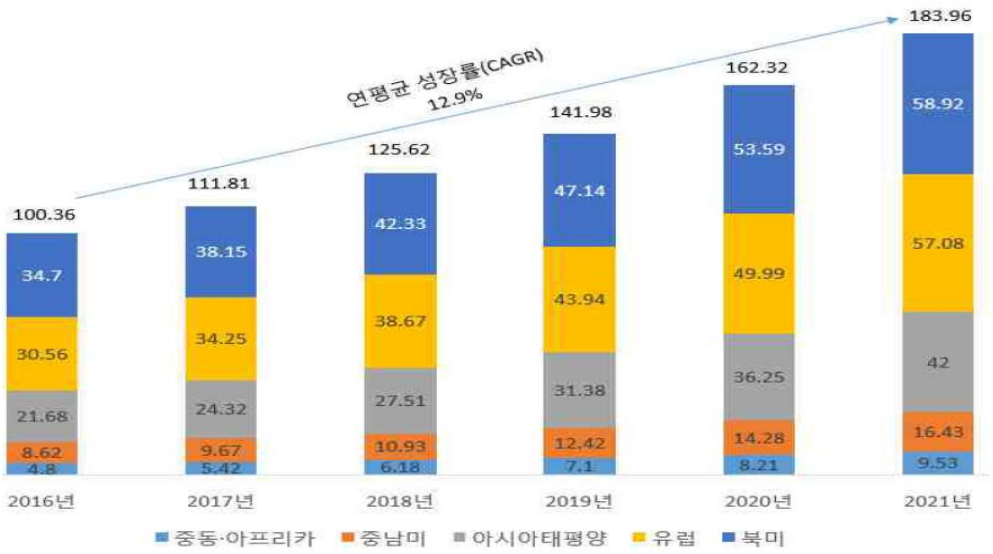
개발기술

- 다양한 공격 패턴을 가지는 여러 종류의 랜섬웨어 완벽 차단➔ 기존 랜섬웨어 및 신종 랜섬웨어 종류에 상관없이 차단 가능
- 클라우드 백업 파일에 대한 랜섬웨어 감염 원천 차단 가능
- 신속한 랜섬웨어 탐지 가능 ➔ 랜섬웨어 탐지까지 걸리는 시간 최소화

기술 활용 전망

- **전 세계 사이버보안 시장 향후 5년간 연평균 12.9%의 높은 성장세 전망**

- IT 분야 전문 시장조사기관 Technavio에 따르면, 전 세계 사이버보안 시장은 2016년 1,003억 6천만 달러 규모에서 5년간 연평균 12.9%로 성장해 2021년에는 1,839억 6천만 달러 규모에 이를 전망



자료: Technavio(2017.09)

[전 세계 권역별 사이버 보안 시장 성장 전망]

보유 특허

No	국가	출원(등록)번호	명칭
1	KR	10-2017-0182896	악성코드 검출을 위한 화이트리스트 구축 방법 및 이를 수행하기 위한 기록매체 및 장치
2	KR	10-2017-0182897	랜섬웨어 탐지 방법, 이를 수행하기 위한 기록매체 및 랜섬웨어 탐지 시스템
3	KR	10-2018-0066884	악성 프로세스 감염을 차단하는 파일 백업 장치 및 그 방법
4	KR	10-2017-0179507	사물인터넷 디바이스 모니터링 프레임워크, 이를 탑재한 포크 서버 및 포크 컴퓨팅 시스템
5	KR	10-2017-153525	모니터링, 분석 및 제어 플랫폼, 이를 탑재한 포크 서버 및 이를 포함하는 포크 컴퓨팅 시스템
6	KR	10-1718739	이기종 하둡을 위한 동적 데이터 복제 시스템 및 방법
7	KR	10-1515707	소프트웨어의 특징정보 기반 스마트폰 앱 불법 복제 및 표절 탐지 방법
8	KR	10-1554393	프로그램의 유사도 비교를 위한 데이터 처리 방법, 장치 및 컴퓨터 프로그램 제품
9	KR	10-1707601	가상 머신 모니터 및 가상 머신 모니터의 스케쥴링 방법

수행 연구 과제

No	과제명	수행년도	주관기관
1	랜섬웨어 대응 기술 개발	2018	정보통신기술진흥센터
2	대용량 데이터 학습기반 동작인식 알고리즘의 최적화 기법 연구	2018	한국전자통신연구원
3	안드로이드 폰 내부 생성 로그 데이터 분석 기술 연구	2018	ETRI부설국가보안기술연구소
4	도커 기반 통합 프레임워크의 효율적인 자동화 관리 방안 연구	2018	전자부품연구원
5	차세대정보컴퓨팅기술개발 사업의 효과성 분석 및 발전방안 연구	2018	한국연구재단
6	가상화 기반 오픈 게이트웨이 플랫폼 오케스트레이션 보안 서비스 프레임워크 설계 구현	2018	한국연구재단
7	가상화 플랫폼기반 통합선박항해지원시스템(INS) 개발	2018	기업체
8	익명 통신 및 봇넷 통신 구조 연구	2017	ETRI부설국가보안기술연구소

(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2019년07월15일
(11) 등록번호 10-2000369
(24) 등록일자 2019년07월09일

(51) 국제특허분류(Int. Cl.)
G06F 21/56 (2013.01)
(52) CPC특허분류
G06F 21/56 (2013.01)
(21) 출원번호 10-2017-0182897
(22) 출원일자 2017년12월28일
심사청구일자 2017년12월28일
(65) 공개번호 10-2019-0080446
(43) 공개일자 2019년07월08일
(56) 선행기술조사문헌
KR101685014 B1*
KR101710928 B1*
JP2016033690 A
JP3947110 B2
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
숭실대학교산학협력단
서울특별시 동작구 상도로 369 (상도동)
(72) 발명자
홍지만
서울특별시 서초구 나루터로4길 49, 332동 701호
이정환
서울특별시 구로구 경인로3길 92, 501호
이진우
서울특별시 동작구 상도로58길 12, 201호 (삼미빌라)
(74) 대리인
윤귀상

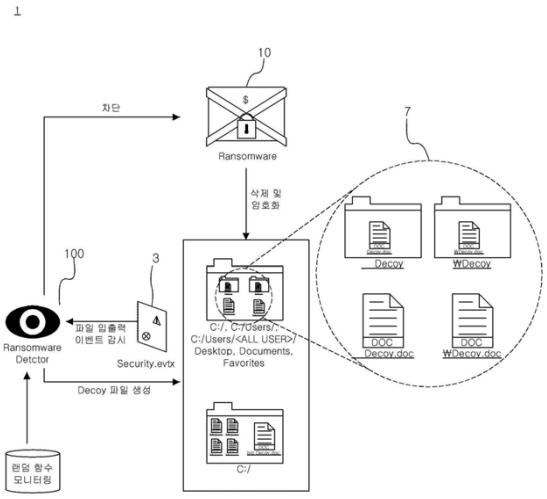
전체 청구항 수 : 총 12 항 심사관 : 윤혜숙

(54) 발명의 명칭 **랜섬웨어 탐지 방법, 이를 수행하기 위한 기록매체 및 램섬웨어 탐지 시스템**

(57) 요 약

랜섬웨어 탐지 방법, 이를 수행하기 위한 기록매체 및 램섬웨어 탐지 시스템이 개시된다. 랜섬웨어 탐지 방법은, 파일명, 파일 크기 및 파일 접근 시간 중 적어도 하나에 따른 랜섬웨어의 파일 탐색 패턴을 반영하여 미끼 파일을 생성하고, 상기 미끼 파일을 랜섬웨어의 모든 탐색 시작 디렉터리에 배치하며, 컴퓨터 시스템 운영체제의 이벤트 로그를 이용하여 상기 미끼 파일을 암호화하는 프로세스를 랜섬웨어로 탐지한다.

대 표 도 - 도1



이 발명을 지원한 국가연구개발사업
과제고유번호 2017-0-00388
부처명 과학기술정보통신부
연구관리전문기관 정보통신기술진흥센터
연구사업명 정보통신 방송 연구개발 1단계 1차년도
연구과제명 랜섬웨어 대응 기술 개발
기 여 율 1/1
주관기관 한양대학교 산학협력단
연구기간 2017.04.01 ~ 2018.12.31
공지예외적용 : 있음

명 세 서

청구범위

청구항 1

컴퓨터 시스템 운영체제에서 특정 파일을 탐색하여 암호화하는 악성 프로세스인 랜섬웨어 탐지 방법에 있어서,

파일명, 파일 크기 및 파일 접근 시간 중 적어도 하나에 따른 랜섬웨어의 파일 탐색 패턴을 반영하여 미끼 파일을 생성하고,

상기 미끼 파일을 랜섬웨어의 모든 탐색 시작 디렉터리에 배치하며,

상기 컴퓨터 시스템 운영체제의 이벤트 로그를 이용하여 상기 미끼 파일을 암호화하는 프로세스를 랜섬웨어로 탐지하고,

파일명, 파일 크기 및 파일 접근 시간 중 적어도 하나에 따른 랜섬웨어의 파일 탐색 패턴을 반영하여 미끼 파일을 생성하는 것은,

표준 서양 언어 키보드(Windows-1252) 배열 순으로 파일명을 탐색하여 암호화할 특정 파일을 탐색하는 상기 랜섬웨어의 파일 탐색 패턴을 반영하여, 표준 서양 언어 키보드 배열의 첫 번째 또는 마지막 배열의 문자로 시작하는 파일명을 갖는 상기 미끼 파일을 생성하는 것을 포함하는 랜섬웨어 탐지 방법.

청구항 2

제1항에 있어서,

상기 컴퓨터 시스템 운영체제의 프로세스 중 랜덤 함수를 이용하며, 파일 탐색 및 입출력 속도가 미리 설정되는 일반 프로세스의 파일 탐색 및 입출력 속도보다 빠른 프로세스를 랜섬웨어로 탐지하는 것을 더 포함하는 랜섬웨어 탐지 방법.

청구항 3

삭제

청구항 4

제1항에 있어서,

파일명, 파일 크기 및 파일 접근 시간 중 적어도 하나에 따른 랜섬웨어의 파일 탐색 패턴을 반영하여 미끼 파일을 생성하는 것은,

파일 크기 순으로 암호화할 특정 파일을 탐색하는 상기 랜섬웨어의 파일 탐색 패턴을 반영하여 미리 설정되는 일반 파일의 크기보다 작은 크기를 갖는 복수 개의 미끼 파일을 생성하고, 상기 미리 설정되는 일반 파일의 크기보다 큰 크기를 갖는 하나의 미끼 파일을 생성하는 것을 포함하는 랜섬웨어 탐지 방법.

청구항 5

제1항에 있어서,

파일명, 파일 크기 및 파일 접근 시간 중 적어도 하나에 따른 랜섬웨어의 파일 탐색 패턴을 반영하여 미끼 파일을 생성하는 것은,

파일 접근 시간 순으로 암호화할 특정 파일을 탐색하는 상기 랜섬웨어의 파일 탐색 패턴을 반영하여 파일 접근 시간이 주기적으로 갱신되는 상기 미끼 파일을 생성하는 것을 포함하는 랜섬웨어 탐지 방법.

청구항 6

제1항에 있어서,

상기 미끼 파일을 랜섬웨어의 모든 탐색 시작 디렉터리에 배치하는 것은,

상기 컴퓨터 시스템 운영체제의 루트 디렉터리에 배치하는 것을 포함하는 랜섬웨어 탐지 방법.

청구항 7

제6항에 있어서,

상기 미끼 파일을 랜섬웨어의 모든 탐색 시작 디렉터리에 배치하는 것은,

상기 루트 디렉터리의 하위 디렉터리인 Favorites, Documents 및 Desktop 디렉터리에 배치하는 것을 포함하는 랜섬웨어 탐지 방법.

청구항 8

제1항, 제2항, 제4항 내지 제7항 중 어느 하나의 항에 따른 랜섬웨어 탐지 방법을 수행하기 위한, 컴퓨터 프로그램이 기록된 컴퓨터로 판독 가능한 기록매체.

청구항 9

컴퓨터 시스템 운영체제에서 특정 파일을 탐색하여 암호화하는 악성 프로세스인 랜섬웨어 탐지 시스템에 있어서,

파일명, 파일 크기 및 파일 접근 시간 중 적어도 하나에 따른 랜섬웨어의 파일 탐색 패턴을 반영하여 미끼 파일을 생성하는 미끼 파일 생성 모듈;

상기 미끼 파일을 랜섬웨어의 모든 탐색 시작 디렉터리에 배치하는 미끼 파일 배치 모듈; 및

상기 컴퓨터 시스템 운영체제의 이벤트 로그를 이용하여 상기 미끼 파일을 암호화하는 프로세스를 랜섬웨어로 탐지하는 랜섬웨어 탐지 모듈을 포함하고,

상기 미끼 파일 생성 모듈은,

파일 크기 순으로 암호화할 특정 파일을 탐색하는 상기 랜섬웨어의 파일 탐색 패턴을 반영하여 미리 설정되는 일반 파일의 크기보다 작은 크기를 갖는 복수 개의 미끼 파일을 생성하고, 상기 미리 설정되는 일반 파일의 크기보다 큰 크기를 갖는 하나의 미끼 파일을 생성하는 것을 포함하는 랜섬웨어 탐지 시스템.

청구항 10

제9항에 있어서,

상기 랜섬웨어 탐지 모듈은,

상기 컴퓨터 시스템 운영체제의 프로세스 중 랜덤 함수를 이용하며, 파일 탐색 및 입출력 속도가 미리 설정되는 일반 프로세스의 파일 탐색 및 입출력 속도보다 빠른 프로세스를 랜섬웨어로 탐지하는 것을 더 포함하는 랜섬웨어 탐지 시스템.

청구항 11

제9항에 있어서,

상기 미끼 파일 생성 모듈은,

표준 서양 언어 키보드(Windows-1252) 배열 순으로 파일명을 탐색하여 암호화할 특정 파일을 탐색하는 상기 랜섬웨어의 파일 탐색 패턴을 반영하여, 표준 서양 언어 키보드 배열의 첫 번째 또는 마지막 배열의 문자로 시작하는 파일명을 갖는 상기 미끼 파일을 생성하는 것을 포함하는 랜섬웨어 탐지 시스템.

청구항 12

삭제

청구항 13

제9항에 있어서,

상기 미끼 파일 생성 모듈은,

파일 접근 시간 순으로 암호화할 특정 파일을 탐색하는 상기 랜섬웨어의 파일 탐색 패턴을 반영하여 파일 접근 시간이 주기적으로 갱신되는 상기 미끼 파일을 생성하는 것을 포함하는 랜섬웨어 탐지 시스템.

청구항 14

제9항에 있어서,

상기 랜섬웨어 탐지 모듈은,

상기 컴퓨터 시스템 운영체제의 이벤트 로그를 이용하여 상기 미끼 파일에 대해 읽기, 쓰기, 삭제 및 파일명 변경 중 어느 하나의 작업을 수행하는 프로세스을 랜섬웨어로 탐지하는 랜섬웨어 탐지 시스템.

발명의 설명

기 술 분 야

[0001] 본 발명은 랜섬웨어 탐지 방법, 이를 수행하기 위한 기록매체 및 랜섬웨어 탐지 시스템에 관한 것으로, 보다 상세하게는 미끼(Decoy) 파일을 이용한 랜섬웨어 탐지 방법, 이를 수행하기 위한 기록매체 및 랜섬웨어 탐지 시스템에 관한 것이다.

배 경 기 술

[0002] 랜섬웨어(Ransomware)는 몸값을 의미하는 "ransom" 및 프로그램을 의미하는 "ware"이 합쳐진 단어로, 사용자의 파일을 암호화하고, 암호화한 파일을 인질로 잡아 사용자에게 금전을 요구하는 악성코드의 한 종류이다. 일반적으로 랜섬웨어에 감염되면 해커에게 돈을 지급하여 손상된 파일의 복구를 시도하는데, 해커에게 돈을 지급하더라도 파일을 완전히 복구할 수 없는 경우도 많아 피해가 증가하고 있다.

[0003] 랜섬웨어의 감염은 대부분 사용자의 부주의에 의해 발생한다. 예를 들어, 사용자가 랜섬웨어가 내제된 실행 이미지를 내려 받아 직접 실행하거나, 플래시 취약점을 이용하는 랜섬웨어가 포함된 특정 웹 사이트에 접속하는 경우 발생한다. 하지만 최근에는 윈도우의 SMB(Server Message Block) 취약점을 이용한 WannaCryptor 랜섬웨어와 같이 인터넷에 연결되어 있기만 해도 랜섬웨어에 감염될 수 있다. 이처럼, 랜섬웨어 감염은 사용자가 주의하더라도 막을 수 있는 것이 아니므로, 랜섬웨어의 탐지 및 차단을 위한 다양한 기술들이 연구되고있다.

[0004] 예를 들어, 파일 확장자 변경 모니터링을 통한 랜섬웨어 탐지 기법이 제안된바 있다. 랜섬웨어가 파일을 암호화하면서 파일 확장자도 변경시키는 점에서 착안한 기법으로, 기존에 발견된 랜섬웨어 확장자(.aaa, .locky, .crypt 등)를 블랙리스트로 등록하고, 파일의 확장자가 블랙리스트에 등록된 확장자로 변경되는지를 모니터링하며, 파일의 확장자를 변경하는 프로세스를 랜섬웨어로 탐지한다. 또는, 사람이 변경할 수 없는 짧은 시간에 다수의 파일의 확장자를 변경하는 프로세스를 탐지하여, 해당 프로세스를 랜섬웨어로 간주하는 방식이 있다. 그러나, 이와 같은 파일 확장자 변경 모니터링 기반의 랜섬웨어 탐지 기법은, 파일 확장자를 변경하지 않는 랜섬웨어의 탐지는 불가능하다.

[0005] 또한, 미끼 파일을 이용한 랜섬웨어 탐지 기법이 제안된바 있다. 미끼 폴더를 각 드라이브의 루트(root) 경로에 생성하여, 미끼 폴더 내의 파일들을 암호화 또는 파일명 변경 등을 시도하는 프로세스를 탐지한다. 그러나, 이와 같은 기법은 루트 경로를 나중에 탐색하는 경우 랜섬웨어를 빠르게 탐지하지 못하며, 드라이브의 루트 경로를 탐색하지 않는 랜섬웨어의 탐지는 불가능하다.

[0006] 또한, 랜섬웨어가 파일을 암호화할 때 발생하는 I/O event 패턴을 이용한 랜섬웨어 탐지 기법이 제안된바 있다. 랜섬웨어가 파일을 암호화할 때 각 파일에 대해 동일한 I/O event 가 발생한다는 점에서 착안한 기법으로, 랜섬웨어의 행위를 분석하여 랜섬웨어가 파일을 암호화할 때 발생하는 I/O event 패턴을 추출하고, 파일 시스템 모니터링을 이용하여 특정 프로세스가 추출한 I/O event 패턴을 발생시키면 해당 프로세스를 랜섬웨어로 간주한다. 그러나, 이와 같은 기법은 파일 암호화 시, 기존의 랜섬웨어의 I/O event 패턴과 다른 패턴을 갖는 신종 랜섬웨어의 탐지는 불가능하다.

발명의 내용

해결하려는 과제

[0007] 본 발명의 일측면은 파일명, 파일 크기 및 파일 접근 시간 중 적어도 하나에 따른 랜섬웨어의 파일 탐색 패턴을 반영하여 미끼 파일을 생성하고, 이를 이용한 랜섬웨어 탐지 방법을 제공한다.

[0008] 본 발명의 다른 측면은 랜섬웨어의 파일 탐색 패턴을 반영하여 미끼 파일을 생성하고, 이를 이용한 랜섬웨어 탐지 방법을 수행하기 위한, 컴퓨터 프로그램이 기록된 컴퓨터로 판독 가능한 기록매체를 제공한다.

[0009] 본 발명의 또 다른 측면은 파일명, 파일 크기 및 파일 접근 시간 중 적어도 하나에 따른 랜섬웨어의 파일 탐색 패턴을 반영하여 미끼 파일을 생성하고, 미끼 파일을 컴퓨터 시스템 운영체제에 배치한 뒤 이를 모니터링하여 랜섬웨어를 탐지하는 랜섬웨어 탐지 시스템을 제공한다.

과제의 해결 수단

[0010] 상기 과제를 해결하기 위한 본 발명의 일측면에 따른 랜섬웨어 탐지 방법은, 파일명, 파일 크기 및 파일 접근 시간 중 적어도 하나에 따른 랜섬웨어의 파일 탐색 패턴을 반영하여 미끼 파일을 생성하고, 상기 미끼 파일을 랜섬웨어의 모든 탐색 시작 디렉터리에 배치하며, 컴퓨터 시스템 운영체제의 이벤트 로그를 이용하여 상기 미끼 파일을 암호화하는 프로세스를 랜섬웨어로 탐지한다.

[0011] 한편, 상기 컴퓨터 시스템 운영체제의 프로세스 중 랜덤 함수를 이용하며, 파일 탐색 및 입출력 속도가 미리 설정되는 일반 프로세스의 파일 탐색 및 입출력 속도보다 빠른 프로세스를 랜섬웨어로 탐지하는 것을 더 포함할 수 있다.

[0012] 또한, 파일명, 파일 크기 및 파일 접근 시간 중 적어도 하나에 따른 랜섬웨어의 파일 탐색 패턴을 반영하여 미끼 파일을 생성하는 것은, 표준 서양 언어 키보드(Windows-1252) 배열 순으로 파일명을 탐색하여 암호화할 특정 파일을 탐색하는 상기 랜섬웨어의 파일 탐색 패턴을 반영하여, 표준 서양 언어 키보드 배열의 첫 번째 또는 마지막 배열의 문자로 시작하는 파일명을 갖는 상기 미끼 파일을 생성하는 것을 포함할 수 있다.

[0013] 또한, 파일명, 파일 크기 및 파일 접근 시간 중 적어도 하나에 따른 랜섬웨어의 파일 탐색 패턴을 반영하여 미끼 파일을 생성하는 것은, 파일 크기 순으로 암호화할 특정 파일을 탐색하는 상기 랜섬웨어의 파일 탐색 패턴을 반영하여 미리 설정되는 일반 파일의 크기보다 작은 크기를 갖는 복수 개의 미끼 파일을 생성하고, 상기 미리 설정되는 일반 파일의 크기보다 큰 크기를 갖는 하나의 미끼 파일을 생성하는 것을 포함할 수 있다.

[0014] 또한, 파일명, 파일 크기 및 파일 접근 시간 중 적어도 하나에 따른 랜섬웨어의 파일 탐색 패턴을 반영하여 미끼 파일을 생성하는 것은, 파일 접근 시간 순으로 암호화할 특정 파일을 탐색하는 상기 랜섬웨어의 파일 탐색 패턴을 반영하여 파일 접근 시간이 주기적으로 갱신되는 상기 미끼 파일을 생성하는 것을 포함할 수 있다.

[0015] 또한, 상기 미끼 파일을 랜섬웨어의 모든 탐색 시작 디렉터리에 배치하는 것은, 상기 컴퓨터 시스템 운영체제의 루트 디렉터리에 배치하는 것을 포함할 수 있다.

[0016] 또한, 상기 미끼 파일을 랜섬웨어의 모든 탐색 시작 디렉터리에 배치하는 것은, 상기 루트 디렉터리의 하위 디렉터리인 Favorites, Documents 및 Desktop 디렉터리에 배치하는 것을 포함할 수 있다.

[0017] 한편, 상기 랜섬웨어 탐지 방법을 수행하기 위한, 컴퓨터 프로그램이 기록된 컴퓨터로 판독 가능한 기록매체일 수 있다.

[0018] 한편, 상기 과제를 해결하기 위한 본 발명의 다른 측면에 따른 랜섬웨어 탐지 시스템은, 파일명, 파일 크기 및 파일 접근 시간 중 적어도 하나에 따른 랜섬웨어의 파일 탐색 패턴을 반영하여 미끼 파일을 생성하는 미끼 파일 생성 모듈, 상기 미끼 파일을 랜섬웨어의 모든 탐색 시작 디렉터리에 배치하는 미끼 파일 배치 모듈 및 컴퓨터 시스템 운영체제의 이벤트 로그를 이용하여 상기 미끼 파일을 암호화하는 프로세스를 랜섬웨어로 탐지하는 랜섬웨어 탐지 모듈을 포함한다.

[0019] 한편, 상기 랜섬웨어 탐지 모듈은, 상기 컴퓨터 시스템 운영체제의 프로세스 중 랜덤 함수를 이용하며, 파일 탐색 및 입출력 속도가 미리 설정되는 일반 프로세스의 파일 탐색 및 입출력 속도보다 빠른 프로세스를 랜섬웨어로 탐지하는 것을 더 포함할 수 있다.

[0020] 또한, 상기 미끼 파일 생성 모듈은, 표준 서양 언어 키보드(Windows-1252) 배열 순으로 파일명을 탐색하여 암호

화할 특정 파일을 탐색하는 상기 랜섬웨어의 파일 탐색 패턴을 반영하여, 표준 서양 언어 키보드 배열의 첫 번째 또는 마지막 배열의 문자로 시작하는 파일명을 갖는 상기 미끼 파일을 생성하는 것을 포함할 수 있다.

[0021] 또한, 상기 미끼 파일 생성 모듈은, 파일 크기 순으로 암호화할 특정 파일을 탐색하는 상기 랜섬웨어의 파일 탐색 패턴을 반영하여 미리 설정되는 일반 파일의 크기보다 작은 크기를 갖는 복수 개의 미끼 파일을 생성하고, 상기 미리 설정되는 일반 파일의 크기보다 큰 크기를 갖는 하나의 미끼 파일을 생성하는 것을 포함할 수 있다.

[0022] 또한, 상기 미끼 파일 생성 모듈은, 파일 접근 시간 순으로 암호화할 특정 파일을 탐색하는 상기 랜섬웨어의 파일 탐색 패턴을 반영하여 파일 접근 시간이 주기적으로 갱신되는 상기 미끼 파일을 생성하는 것을 포함할 수 있다.

[0023] 또한, 상기 랜섬웨어 탐지 모듈은, 상기 컴퓨터 시스템 운영체제의 이벤트 로그를 이용하여 상기 미끼 파일에 대해 읽기, 쓰기, 삭제 및 파일명 변경 중 어느 하나의 작업을 수행하는 프로세스을 랜섬웨어로 탐지할 수 있다.

발명의 효과

[0024] 본 발명에 따르면, 랜섬웨어의 파일 탐색 패턴을 반영하여 미끼 파일을 생성하고, 이를 이용하여 랜섬웨어를 탐지함으로써, 미리 추출되는 패턴에 부합하는 암호화 행위를 하는 기존 랜섬웨어뿐만 아니라 신종 랜섬웨어의 탐지가 가능하다.

[0025] 아울러, 랜섬웨어를 탐지하여 사용자의 파일이 피해를 받기 이전에 랜섬웨어를 차단할 수 있어 금전적 손해를 방지할 수 있다.

도면의 간단한 설명

[0026] 도 1은 본 발명의 일 실시예에 따른 랜섬웨어 탐지 시스템에서의 랜섬웨어 탐지를 위한 동작 흐름을 나타낸 개념도이다.

도 2는 본 발명의 일 실시예에 따른 랜섬웨어 탐지 시스템의 제어 블록도이다.

도 3은 도 2에 도시된 미끼 파일 생성 모듈 및 미끼 파일 배치 모듈의 동작을 구현하기 위한 소스 코드의 일 예이다.

도 4는 윈도우 운영체제에서 제공하는 이벤트 뷰어의 일 예이다.

도 5 내지 도 10은 랜섬웨어의 특정 파일 암호화 방법을 설명하기 위한 도면이다.

도 11은 본 발명의 일 실시예에 따른 랜섬웨어 탐지 방법의 순서도이다.

발명을 실시하기 위한 구체적인 내용

[0027] 후술하는 본 발명에 대한 상세한 설명은, 본 발명이 실시될 수 있는 특정 실시예를 예시로서 도시하는 첨부 도면을 참조한다. 이들 실시예는 당업자가 본 발명을 실시할 수 있기에 충분하도록 상세히 설명된다. 본 발명의 다양한 실시예는 서로 다르지만 상호 배타적일 필요는 없음이 이해되어야 한다. 예를 들어, 여기에 기재되어 있는 특정 형상, 구조 및 특성은 일 실시예와 관련하여 본 발명의 정신 및 범위를 벗어나지 않으면서 다른 실시예로 구현될 수 있다. 또한, 각각의 개시된 실시예 내의 개별 구성요소의 위치 또는 배치는 본 발명의 정신 및 범위를 벗어나지 않으면서 변경될 수 있음이 이해되어야 한다. 따라서, 후술하는 상세한 설명은 한정적인 의미로서 취하려는 것이 아니며, 본 발명의 범위는, 적절하게 설명된다면, 그 청구항들이 주장하는 것과 균등한 모든 범위와 더불어 첨부된 청구항에 의해서만 한정된다. 도면에서 유사한 참조부호는 여러 측면에 걸쳐서 동일하거나 유사한 기능을 지칭한다.

[0028] 이하, 도면들을 참조하여 본 발명의 바람직한 실시예들을 보다 상세하게 설명하기로 한다.

[0029] 도 1은 본 발명의 일 실시예에 따른 랜섬웨어 탐지 시스템에서의 랜섬웨어 탐지를 위한 동작 흐름을 나타낸 개념도이다.

[0030] 도 1을 참조하면, 본 발명의 일 실시예에 따른 랜섬웨어 탐지 시스템(100)은 컴퓨터 시스템 운영체제(1)에 나타나는 랜섬웨어(10)를 탐지할 수 있다. 랜섬웨어(10)는 사용자의 파일을 암호화하고, 암호화한 파일을 인질로 삼아 사용자에게 금전을 요구하는 악성코드의 한 종류이다. 본 발명의 일 실시에에 따른 랜섬웨어 탐지 시스템

(100)은 다양한 컴퓨터 시스템 운영체제(1)에 범용적으로 적용될 수 있다.

[0031] 본 발명의 일 실시예에 따른 랜섬웨어 탐지 시스템(100)은 랜섬웨어(10)의 탐지를 위해 미끼(decoy) 파일(7)을 생성하여 디렉터리에 배치할 수 있다. 특히, 랜섬웨어 탐지 시스템(100)은 파일명, 파일 크기 및 파일 접근 시간 중 적어도 하나에 따른 랜섬웨어(10)의 파일 탐색 패턴을 반영하여 미끼 파일(7)을 생성할 수 있으며, 생성한 미끼 파일(7)을 랜섬웨어(10)의 일반적인 탐색 시작 디렉터리에 배치할 수 있다.

[0032] 또한, 랜섬웨어 탐지 시스템(100)은 컴퓨터 시스템 운영체제(1)에서 제공하는 이벤트 로그를 이용하여 미끼 파일(7)에 대한 작업을 모니터링할 수 있다. 이벤트 로그는 컴퓨터 시스템 운영체제(1)에서 발생하는 모든 작업, 예를 들어, 로그인 시도, 레지스트리 변경, 네트워크 이용, 파일 생성, 파일 접근, 파일 삭제 등이 실시간으로 저장될 수 있다. 이러한 이벤트 로그는 BXML(Binaly XML) 형태로 저장될 수 있으며, 랜섬웨어 탐지 시스템(100)은 이벤트 로그 파일(Security.evtx)(3)을 분석하여 미끼 파일(7)에 대한 암호화 작업을 하는 프로세스를 랜섬웨어(10)로 탐지할 수 있다.

[0033] 또한, 랜섬웨어 탐지 시스템(100)은 컴퓨터 시스템 운영체제(1)에서 사용되는 랜덤 함수를 모니터링할 수 있다. 컴퓨터 시스템 운영체제(1)의 일반적인 프로세스는 랜덤 함수를 이용하지 않는 반면, 랜섬웨어(10)는 랜덤 함수를 이용하여 암호화할 파일을 탐색할 수 있다. 따라서, 랜섬웨어 탐지 시스템(100)은 컴퓨터 시스템 운영체제(1)에서 사용되는 랜덤 함수를 모니터링하여 랜덤 함수를 이용하는 프로세스를 랜섬웨어(10)로 탐지할 수 있다.

[0034] 이와 같이, 본 발명의 일 실시예에 따른 랜섬웨어 탐지 시스템(100)은 랜섬웨어(10)의 파일 암호화 행위 패턴이 아닌 파일 탐색 자체를 기준으로 한 패턴에 따라 랜섬웨어(10)의 접근을 유도하는 미끼 파일(7)을 생성함으로써, 종래의 행위 패턴이 알려진 랜섬웨어(10)뿐만 아니라 새로운 행위 패턴을 갖는 신종 랜섬웨어(10)의 탐지가 가능하다.

[0035] 도 2는 본 발명의 일 실시예에 따른 랜섬웨어 탐지 시스템의 제어 블록도이다.

[0036] 도 2를 참조하면, 랜섬웨어 탐지 시스템(100)은 미끼 파일 생성 모듈(110), 미끼 파일 배치 모듈(120) 및 랜섬웨어 탐지 모듈(130)을 포함할 수 있다. 이하, 랜섬웨어 탐지 시스템(100)의 각 구성요소에 대하여 구체적으로 설명한다.

[0037] 미끼 파일 생성 모듈(110)은 랜섬웨어(10)의 접근을 유도하는 미끼 파일을 생성할 수 있다. 랜섬웨어는 컴퓨터 시스템 운영체제(1)에서 특정 확장자를 갖는 파일을 탐색하여 암호화할 수 있다. 예를 들어, 랜섬웨어는 컴퓨터 시스템 운영체제(1)에서 ".txt", ".doc", ".docx", ".xls", ".xlsx", ".ppt", ".pptx", ".odt", ".jpg", ".png", ".csv", ".sql", ".mdb", ".sln", ".php", ".asp", ".aspx", ".html", ".xml", ".psd" 등의 확장자를 갖는 파일을 탐색할 수 있다. 따라서, 미끼 파일 생성 모듈(110)은 상술한 확장자를 갖는 미끼 파일을 생성할 수 있다. 이때, 랜섬웨어마다 파일 탐색 순서가 다르지만, 일반적으로 랜섬웨어의 파일 탐색 순서는 파일명, 파일 크기 및 파일 접근 시간 중 적어도 하나에 따라 정해질 수 있다. 따라서, 미끼 파일 생성 모듈(110)은 파일명, 파일 크기 및 파일 접근 시간 중 적어도 하나에 따른 랜섬웨어의 파일 탐색 패턴을 반영하여 미끼 파일을 생성할 수 있다.

[0038] 구체적으로, 미끼 파일 생성 모듈(110)은 파일명에 따른 랜섬웨어의 파일 탐색 패턴을 반영하여 미끼 파일을 생성할 수 있다. 컴퓨터 시스템 운영체제(1)가 일반적으로 널리 사용되는 윈도우 운영체제인 경우, 랜섬웨어의 파일 탐색 패턴을 분석해보면, 랜섬웨어는 윈도우 운영체제의 표준 서양 언어 키보드(Window-1252) 배열 순서에 따라 파일명을 탐색하여 암호화할 특정 파일을 탐색할 수 있다. 또는, 랜섬웨어는 윈도우 운영체제의 표준 서양 언어 키보드 배열의 역순으로 파일명을 탐색하여, 암호화할 특정 파일을 탐색할 수 있다. 따라서, 미끼 파일 생성 모듈(110)은 표준 서양 언어 키보드 배열의 첫 번째 문자(" ", 0x0020)로 시작하는 파일명을 갖는 미끼 파일을 생성할 수 있다. 또는, 미끼 파일 생성 모듈(110)은 표준 서양 언어 키보드 배열의 첫 번재 문자("₩", 0xA3DC)로 시작하는 파일명을 갖는 미끼 파일을 생성할 수 있다.

[0039] 이와 같은 경우, 랜섬웨어가 표준 서양 언어 키보드 배열 순 또는 역순으로 파일명을 탐색하면, 표준 서양 언어 키보드 배열의 첫 번째 또는 마지막 문자로 시작하는 파일명을 갖는 미끼 파일이 가장 먼저 탐색되어 암호화될 수 있다.

[0040] 또한, 미끼 파일 생성 모듈(110)은 파일 크기에 따른 랜섬웨어의 파일 탐색 패턴을 반영하여 미끼 파일을 생성할 수 있다. 랜섬웨어의 파일 탐색 패턴을 분석해보면, 랜섬웨어는 파일 크기가 큰 순으로 암호화할 특정 파일을 탐색할 수 있다. 또는, 랜섬웨어는 파일 크기가 작은 순으로 암호화할 특정 파일을 탐색할 수 있다. 따라서, 미끼 파일 생성 모듈(110)은 미리 설정되는 일반 파일의 크기보다 작은 크기를 갖는 미끼 파일을 복수 개 생성

하고, 일반 파일의 크기보다 큰 크기를 갖는 하나의 미끼 파일을 생성할 수 있다.

[0041] 이와 같은 경우, 랜섬웨어가 파일 크기가 큰 순으로 특정 파일을 탐색하면, 일반 파일의 크기보다 큰 미끼 파일이 가장 먼저 탐색되어 암호화될 수 있다. 또는, 랜섬웨어가 파일 크기가 작은 순으로 특정 파일을 탐색하면, 일반 파일의 크기보다 작은 미끼 파일이 가장 먼저 탐색되어 암호화될 수 있다. 이때, 일반 파일의 크기보다 작은 미끼 파일은 랜섬웨어에 의해 암호화되는데에 걸리는 시간이 일반 파일에 비해 짧아, 랜섬웨어 탐지 시스템(100)에서 랜섬웨어의 탐지 또는 차단 전에 해당 미끼파일의 암호화가 완료될 수 있다. 따라서, 미끼 파일 생성 모듈(110)은 일반 파일의 크기보다 작은 크기를 갖는 미끼 파일을 복수 개 생성하는데, 랜섬웨어가 복수 개의 미끼 파일의 암호화를 완료하고, 일반 파일을 암호화하기 전에 해당 랜섬웨어의 탐지 또는 차단이 보장되며, 컴퓨터 시스템 운영체제(1)의 용량을 많이 차지하지 않도록, 그 개수를 설정할 수 있다.

[0042] 또한, 미끼 파일 생성 모듈(110)은 파일 접근 시간에 따른 랜섬웨어의 파일 탐색 패턴을 반영하여 미끼 파일을 생성할 수 있다. 랜섬웨어의 파일 탐색 패턴을 분석해보면, 랜섬웨어는 파일 접근 시간이 최근인 순으로 암호화할 특정 파일을 탐색할 수 있다. 파일 접근 시간이 최근일수록 사용자에게 중요한 파일일 가능성이 높기 때문이다. 따라서, 미끼 파일 생성 모듈(110)은 파일 접근 시간이 주기적으로 갱신되는 미끼 파일을 생성할 수 있다.

[0043] 미끼 파일 배치 모듈(120)은 미끼 파일 생성 모듈(110)에서 생성하는 적어도 하나의 미끼 파일을 랜섬웨어의 모든 탐색 시작 디렉터리에 배치할 수 있다. 랜섬웨어가 특정 확장자를 갖는 파일을 탐색하기 위해, 탐색을 시작하는 디렉터리는 다양하다. 컴퓨터 시스템 운영체제(1)가 일반적으로 널리 사용되는 윈도우 운영체제인 경우, 랜섬웨어는 윈도우 운영체제의 루트 디렉터리인 C:/ 부터 탐색을 시작하는 것이 대부분이며, 일부 랜섬웨어는 그 하위 디렉터리인 C:/User/ 부터 탐색을 시작한다. 또는, 일부 랜섬웨어는 그 하위 디렉터리인 C:/Users/<ALL USER>/Favorites, C:/Users/<ALL USER>/Documents, C:/Users/ <ALL USER>/Desktop 등부터 탐색을 시작한다. 따라서, 미끼 파일 배치 모듈(120)은 C:/, C:/User/, C:/Users/<ALL USER>/Favorites, C:/Users/<ALL USER>/Documents, C:/Users/ <ALL USER>/Desktop 디렉터리에 미끼 파일을 배치함으로써, 컴퓨터 시스템 운영체제(1)의 저장 공간의 낭비를 줄일 수 있다.

[0044] 이와 같은, 미끼 파일 생성 모듈(110) 및 미끼 파일 배치 모듈(120)에서의 랜섬웨어의 파일 탐색 패턴을 반영한 미끼 파일 생성 및 배치 과정은 도 3에 자세히 개시되어있다.

[0045] 도 3은 도 2에 도시된 미끼 파일 생성 모듈 및 미끼 파일 배치 모듈의 동작을 구현하기 위한 소스 코드의 일 예이다.

[0046] 도 3에 도시된 소스 코드에 대하여 간단히 설명하면, 미끼 파일 배치 모듈(120)은 윈도우 운영체제의 사용자 계정 이름을 획득할 수 있다. 미끼 파일 배치 모듈(120)은 사용자 계정 이름으로부터 완성되는 랜섬웨어의 탐색 시작 디렉터리에 미끼 파일을 배치할 수 있다. 미끼 파일 배치 모듈(120)은 디렉터리에 미끼 폴더를 생성할 수 있으며, 그 안에 미끼 파일 생성 모듈(110)에서 생성하는 미끼 파일을 배치할 수 있다. 미끼 파일 생성 모듈(110)은 ".doc"의 확장자를 갖는 미끼 파일을 생성할 수 있다. 미끼 파일 생성 모듈(110)은 상술한 것처럼, 파일명, 파일 크기 및 파일 접근 시간에 따른 랜섬웨어의 파일 탐색 패턴을 반영하여 미끼 파일을 생성할 수 있다.

[0047] 한편, 랜섬웨어 탐지 모듈(130)은 미끼 파일을 변경하는 프로세스를 랜섬웨어로 탐지할 수 있다. 이를 위해, 랜섬웨어 탐지 모듈(130)은 컴퓨터 시스템 운영체제(1)의 이벤트 로그를 이용하여 미끼 파일을 암호화하는 프로세스를 모니터링할 수 있다. 컴퓨터 시스템 운영체제(1)의 이벤트 로그는 컴퓨터 시스템에서 발생하는 모든 작업이 저장될 수 있다. 예를 들어, 이벤트 로그에는 로그인 시도, 레지스트리 변경, 네트워크 이용, 파일 생성, 접근, 삭제뿐만 아니라, 이벤트 ID, 프로세스 ID, 프로세스 이름, 개체 이름, 개체에 대한 행위 등의 정보가 실시간으로 저장될 수 있다. 이러한 이벤트 로그는 BXML(Binaly XML) 형태로 저장될 수 있다. 컴퓨터 시스템 운영체제(1)가 일반적으로 널리 사용되는 윈도우 운영체제인 경우, 이러한 이벤트 로그에 저장되는 정보를 분석하여 이벤트 뷰어로 제공할 수 있다.

[0048] 도 4는 윈도우 운영체제에서 제공하는 이벤트 뷰어의 일 예이다.

[0049] 윈도우 운영체제는 도 4와 같은 이벤트 뷰어를 제공할 수 있다. 도 4를 참조하면, 파일 삭제 이벤트를 분석하여 제공하고 있는데, 프로세스 ID가 0x4f78인 프로세스 file_create_test.exe 가 aaaaa.txt 파일을 삭제하는 것이 확인된다.

[0050] 랜섬웨어 탐지 모듈(130)은 컴퓨터 시스템 운영체제(1)의 이벤트 로그를 분석하여 미끼 파일에 대해 읽기, 쓰기, 삭제, 파일명 변경 등의 작업을 수행하는 프로세스를 랜섬웨어로 탐지할 수 있다. 이와 관련하여 도 5 내

지 도 11을 참조하여 설명한다.

[0051] 도 5 내지 도 10은 랜섬웨어의 특정 파일 암호화 방법을 설명하기 위한 도면이다.

[0052] 먼저, 도 5를 참조하면, 랜섬웨어는 "write-in-place" 방법으로 파일을 암호화할 수 있다. 랜섬웨어는 임시 파일(tmp.txt)을 생성하고, 목표 파일(a.txt)을 암호화하고 그 데이터를 임시 파일(tmp.txt)에 저장할 수 있다. 그리고, 랜섬웨어는 임시 파일(tmp.txt)에 있는 내용을 목표 파일(a.txt)에 덮어쓸 수 있다. 이와 같은 "write-in-place"암호화 방식의 단계는 도 6에 자세히 개시되어 있다. 이와 같은 방식으로 파일 암호화를 수행하는 랜섬웨어를 탐지하기 위해서는, 이벤트 로그의 읽기 및 쓰기를 감시하는 것이 바람직하다.

[0053] 또한, 도 7을 참조하면, 랜섬웨어는 "rename-and-encrypt" 방법으로 파일을 암호화할 수 있다. 랜섬웨어는 목표 파일(a.txt)의 이름을 다른 이름으로 변경한 파일(random.txt)을 생성하고, 그 파일(random.txt)을 상술한 "write-in-place" 방법으로 암호화한 뒤, 다시 이름을 목표 파일(a.txt)의 이름으로 변경할 수 있다. 이와 같은 "rename-and-encrypt" 암호화 방식의 단계는 도 8에 자세히 개시되어 있다. 이와 같은 방식으로 파일 암호화를 수행하는 랜섬웨어를 탐지하기 위해서는, 이벤트 로그의 읽기 및 쓰기뿐만 아니라, 파일 이름 변경을 감시하는 것이 바람직하다.

[0054] 또한, 도 9를 참조하면, 랜섬웨어는 "create-encrypt-and-delete" 방법으로 파일을 암호화할 수 있다. 랜섬웨어는 새로운 파일(a.txt.encrypt)을 생성하고, 암호화할 파일(a.txt)을 정해진 크기만큼 읽은 뒤 암호화하여 새로운 파일(a.txt.encrypt)에 저장하고, 읽은 파일을 지울 수 있다. 이와 같은 "create-encrypt-and-delete"암호화 방식의 단계는 도 10에 자세히 개시되어 있다. 이와 같은 방식으로 파일 암호화를 수행하는 랜섬웨어를 탐지하기 위해서는, 이벤트 로그의 읽기, 쓰기, 삭제 및 암호화 등을 감시하는 것이 바람직하다.

[0055] 이와 같이, 랜섬웨어는 목표 파일에 읽기, 쓰기, 삭제, 파일명 변경 등의 작업을 수행하여 목표 파일을 변경할 수 있다. 따라서, 랜섬웨어 탐지 모듈(130)은 컴퓨터 시스템 운영체제(1)의 이벤트 로그를 분석하여 미끼 파일에 대해 읽기, 쓰기, 삭제, 파일명 변경 등의 작업을 수행하는 프로세스를 랜섬웨어로 간주할 수 있다.

[0056] 이에 더하여, 랜섬웨어 탐지 모듈(130)은 컴퓨터 시스템 운영체제(1)의 프로세스 중 랜덤 함수를 이용하며, 파일 탐색 및 입출력 속도가 미리 설정되는 일반 프로세스의 파일 탐색 및 입출력 속도보다 빠른 프로세스를 랜섬웨어로 탐지할 수 있다. 컴퓨터 시스템 운영체제(1)의 일반 프로세스는 랜덤 함수를 사용하지 않는다. 반면, 랜섬웨어는 랜덤 함수를 이용하여 암호화할 파일을 탐색할 수 있다. 또한, 컴퓨터 시스템 운영체제(1)의 일반 프로세스 중 랜섬웨어보다 빠른 속도로 파일 탐색 및 입출력을 수행하는 프로세스는 거의 없다. 따라서, 랜섬웨어 탐지 모듈(130)은 컴퓨터 시스템 운영체제(1)의 랜덤 함수를 모니터링하여 랜덤 함수를 이용하는 프로세스를 검색할 수 있으며, 해당 프로세스 중 일반 프로세스의 파일 탐색 및 입출력 속도보다 빠른 프로세스를 랜섬웨어로 탐지할 수 있다.

[0057] 이하에서는, 본 발명의 일 실시예에 따른 랜섬웨어 탐지 방법에 대하여 설명한다. 본 발명의 일 실시예에 다른 랜섬웨어 탐지 방법은 도 2에 도시된 랜섬웨어 탐지 시스템(100)과 실질적으로 동일한 구성에서 진행될 수 있다. 따라서, 도 2의 랜섬웨어 탐지 시스템(100)과 동일한 구성요소는 동일한 도면부호를 부여하고, 반복되는 설명은 생략하기로 한다.

[0058] 도 11은 본 발명의 일 실시예에 따른 랜섬웨어 탐지 방법의 순서도이다.

[0059] 도 11을 참조하면, 미끼 파일 생성 모듈(110)은 랜섬웨어의 파일 탐색 패턴을 반영한 미끼 파일을 생성할 수 있다(300). 미끼 파일 생성 모듈(110)은 파일명, 파일 크기 및 파일 접근 시간 중 적어도 하나에 따른 랜섬웨어의 파일 탐색 패턴을 반영하여 미끼 파일을 생성할 수 있다. 구체적으로, 미끼 파일 생성 모듈(110)은 윈도우 운영체제의 표준 서양 언어 키보드 배열의 첫 번째 또는 마지막 배열의 문자로 시작하는 파일명을 갖는 미끼 파일을 생성할 수 있다. 또는, 미끼 파일 생성 모듈(110)은 미리 설정되는 일반 파일의 크기보다 작은 크기를 갖는 복수 개의 미끼 파일을 생성하고, 미리 설정되는 일반 파일의 크기보다 큰 크기를 갖는 하나의 미끼 파일을 생성할 수 있다. 또는, 미끼 파일 생성 모듈(110)은 파일 접근 시간이 주기적으로 갱신되는 미끼 파일을 생성할 수 있다.

[0060] 또한, 미끼 파일 배치 모듈(120)은 랜섬웨어의 탐색 시작 디렉터리에 미끼 파일을 배치할 수 있다(310). 미끼 파일 배치 모듈(120)은 C:/, C:/User/, C:/Users/<ALL USER>/Favorites, C:/Users/<ALL USER>/Documents, C:/Users/ <ALL USER>/Desktop 디렉터리에 미끼 파일을 배치할 수 있다.

[0061] 또한, 랜섬웨어 탐지 모듈(130)은 컴퓨터 시스템 운영체제(1)의 이벤트 로그를 분석하여(320), 미끼 파일을 암

호화하는 프로세스가 탐지되면(330), 해당 프로세스를 랜섬웨어로 탐지할 수 있다(340). 랜섬웨어 탐지 모듈(130)은 컴퓨터 시스템 운영체제(1)의 이벤트 로그에서, 미끼 파일에 대해 읽기, 쓰기, 삭제, 파일명 변경 등의 작업을 수행하는 프로세스를 미끼 파일을 암호화하는 프로세스로 간주하여 랜섬웨어로 탐지할 수 있다. 또한, 랜섬웨어 탐지 모듈(130)은 컴퓨터 시스템 운영체제(1)의 프로세스 중 랜덤 함수를 이용하며, 파일 탐색 및 입출력 속도가 미리 설정되는 일반 프로세스의 파일 탐색 및 입출력 속도보다 빠른 프로세스를 미끼 파일을 암호화하는 프로세스로 간주하여 랜섬웨어로 탐지할 수 있다.

[0062] 이와 같은, 랜섬웨어 탐지 방법은 애플리케이션으로 구현되거나 다양한 컴퓨터 구성요소를 통하여 수행될 수 있는 프로그램 명령어의 형태로 구현되어 컴퓨터 판독 가능한 기록 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능한 기록 매체는 프로그램 명령어, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다.

[0063] 상기 컴퓨터 판독 가능한 기록 매체에 기록되는 프로그램 명령어는 본 발명을 위하여 특별히 설계되고 구성된 것들이거니와 컴퓨터 소프트웨어 분야의 당업자에게 공지되어 사용 가능한 것일 수도 있다.

[0064] 컴퓨터 판독 가능한 기록 매체의 예에는, 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체, CD-ROM, DVD 와 같은 광기록 매체, 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 ROM, RAM, 플래시 메모리 등과 같은 프로그램 명령어를 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다.

[0065] 프로그램 명령어의 예에는, 컴파일러에 의해 만들어지는 것과 같은 기계어 코드 뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드도 포함된다. 상기 하드웨어 장치는 본 발명에 따른 처리를 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

[0066] 이상에서는 실시예들을 참조하여 설명하였지만, 해당 기술 분야의 숙련된 당업자는 하기의 특허 청구범위에 기재된 본 발명의 사상 및 영역으로부터 벗어나지 않는 범위 내에서 본 발명을 다양하게 수정 및 변경시킬 수 있음을 이해할 수 있을 것이다.

부호의 설명

[0067] 1: 컴퓨터 시스템 운영체제

3: 이벤트 로그 파일

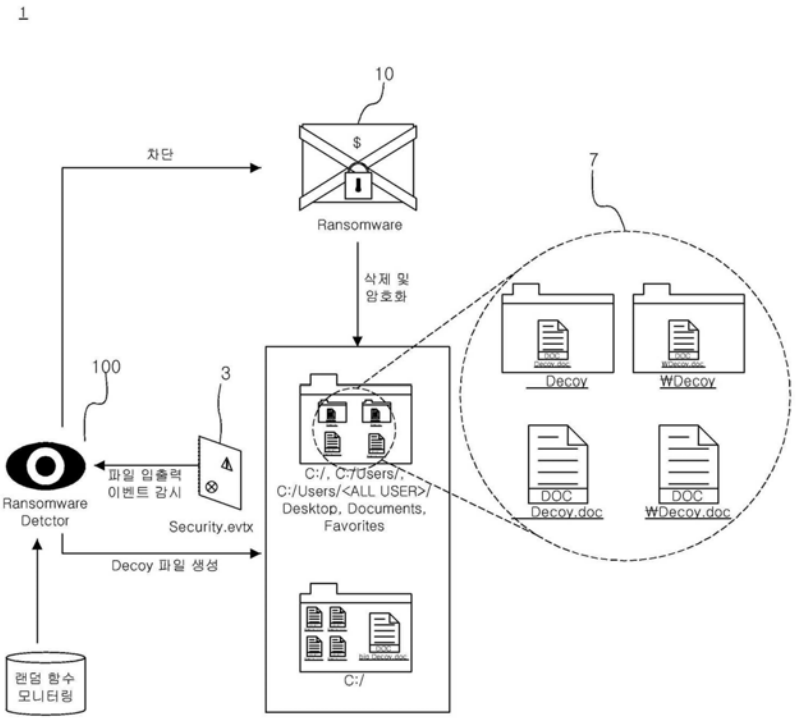
7: 미끼 파일

10: 랜섬웨어

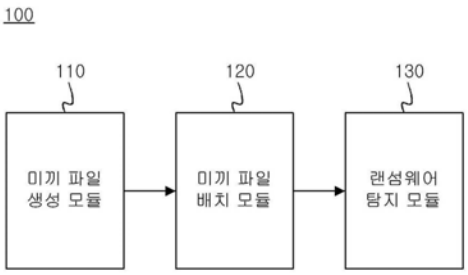
100: 랜섬웨어 탐지 시스템

도면

도면1


1
10
$
Ransomware
차단
삭제 및
암호화
7
100
3
Ransomware
Detctor
파일 입출력
이벤트 감시
Security.evtx
Decoy 파일 생성
랜덤 함수
모니터링
C:/, C:/Users/,
C:/Users/<ALL USER>/
Desktop, Documents,
Favorites
C:/
Decoy
₩Decoy
DOC
Decoy.doc
DOC
₩Decoy.doc

도면2


100
110
120
130
미끼 파일
생성 모듈
미끼 파일
배치 모듈
랜섬웨어
탐지 모듈

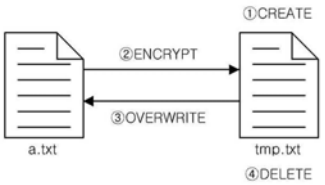
도면3

```
1: procedure MakeDecoyFiles(smallDecoy, bigDecoy, n)
2:     useNames = GetSystemUserName(ALLUSER)
3:     sw = GetStartOfWord();
4:     ew = GetEndOfWord();
5:     startFolder = {"C:/", "C:/Users/"}
6:     for each name in userNames do
7:         startFolder+="C:/Users/"+name+"/Desktop/"
8:         startFolder+="C:/Users/"+name+"/Documents/"
9:         startFolder+="C:/Users/"+name+"/Favorites/"
10:
11:    for each path in startFolder do
12:        MakeFolder(path+sw)
13:        MakeFolder(path+ew)
14:
15:    for i = 1 to n do
16:        MakeDecoyFile(path+sw+i+".doc", smallDecoy)
17:        MakeDecoyFile(path+ew+i+".doc", smallDecoy)
18:        MakeDecoyFile(path+sw+"/"+sw+i+".doc", smallDecoy)
19:        MakeDecoyFile(path+ew+"/"+ew+i+".doc", smallDecoy)
20:
21:        MakeDecoyFile(path+sw+"big"+i+".doc", bigDecoy)
22:
23:    MakeDecoyFile("C:/"+sw+"/Recent.doc", bigDecoy)
24:    new Thread(RegularlyUpdateDecoyFile,
25:        "C:/"+sw+"/Recent.doc")
26:
27:    new Thead(RandomizeMonitoring)
```

도면4

```
개체에 액서스하려고 했습니다.

주체:
        보안 ID:        DESKTOP-QDHQNU0₩User
        계정 이름:      User
        계정 도메인:    DESKTOP-QDHQNU0
        로그인 ID:      0x6408C58

개체:
        개체 서버:      Security
        개체 유형:      File
        개체 이름:      D:₩Documents₩Visula Studio 2017₩Project₩file_create_test₩aaaa.txt
        핸들 ID:        0xf0
        리소스 특성:    S:AI

프로세스 정보:
        프로세스 ID:    0x4f78
        프로세스 이름:  D:₩Documents₩Visula Studio 2017₩Project₩file_create_test₩Debug₩file_create_test.exe

액세스 요청 정보:
        액세스:         DELETE

        액세스 마스크:  0x10000
```

도면5

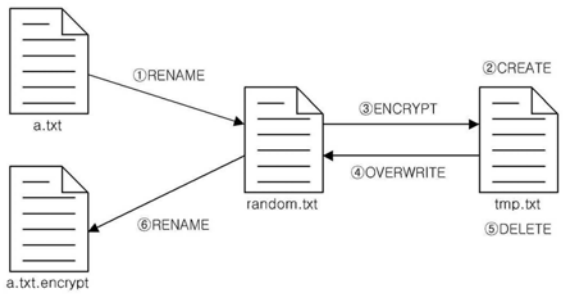


<write-in-place>

도면6

```
1: procedure WriteInPlace(filePath)
2:     tmpFile = CreateTmpFile()
3:     encryptFile = Open(filePath)
4:
5:     while true do
6:         data = Read(encryptFile)
7:         if data == null then
8:             break
9:         encryptedData = Encrypt(data)
10:        Write(tmpFile, encryptedData);
11:
12:    while true do
13:        data = Read(tmpFile)
14:        if data == null then
15:            beark
16:        Write(encryFile, data);
17:
18:    Close(tmpFile)
19:    Close(encryptFile)
```

도면7

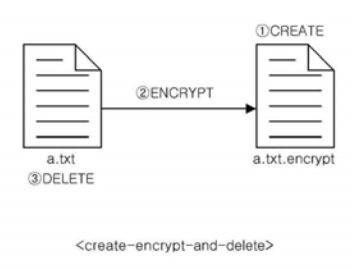


<rename-and-encrypt>

도면8

```
1: procedure RenameAndEncrypt(filePath)
2:     tmpFilePath = GetTmpFileName()
3:     RenameFile(filePath, tmpFilePath)
4:     tmpFile = CreateTmpFile()
5:     encryptFile = Open(tmpFilePath)
6:
7:     while true do
8:         data = Read(encryptFile)
9:         if data == null then
10:            beark
11:        encryptData = encrypt(data)
12:        Write(tmpFile, encryptData)
13:
14:    while true do
15:        data = Read(tmpFile)
16:        if data == null then
17:            break
18:        Write(encryFile, data)
19:
20:    Close(tmpFile)
21:    Close(encryptFile)
22:    RenameFile(tmpFilePath, filePath + ransomwareExtnesion)
```

도면9

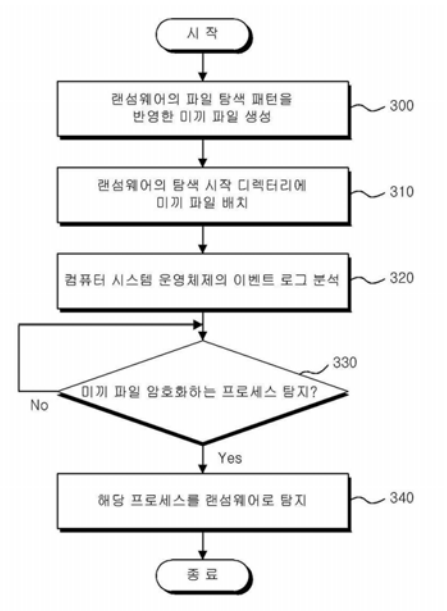


<create-encrypt-and-delete>

도면10

```
1: procedure CreateEncryptAndDelete(filePath)
2:     newFile = CraeteTmpFile(filePath + ransomwareExtension)
3:     originalFile = Open(FilePath)
4:
5:     while true do
6:         data = Read(originalFile)
7:         if data == null then
8:             BREAK
9:         encryptdData = Encrypt(data)
10:        Write(newFile, encryptedData)
11:
12:    Close(newFile)
13:    Close(originalFile)
```

도면11



(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2020년03월03일
(11) 등록번호 10-2084183
(24) 등록일자 2020년02월26일

(51) 국제특허분류(Int. Cl.)
G06F 21/56 (2013.01) *G06F 11/14* (2006.01)
(52) CPC특허분류
G06F 21/56 (2013.01)
G06F 11/1448 (2013.01)
(21) 출원번호 10-2018-0066884
(22) 출원일자 2018년06월11일
심사청구일자 2018년06월11일
(65) 공개번호 10-2019-0140285
(43) 공개일자 2019년12월19일
(56) 선행기술조사문헌
US20170206353 A1*
KR1020170137534 A
KR1020090089584 A
KR1020150057980 A*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
숭실대학교산학협력단
서울특별시 동작구 상도로 369 (상도동)
(72) 발명자
홍지만
서울특별시 서초구 나루터로4길 49, 332동 701호 (잠원동, 신반포17차아파트)
손명준
서울특별시 관악구 인헌길 48 102동 203호(봉천동)
(뒷면에 계속)
(74) 대리인
특허법인다나

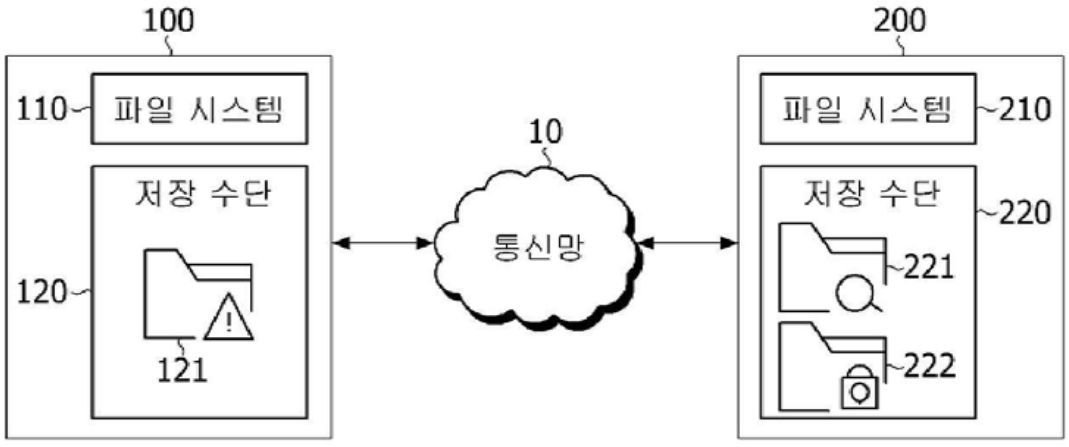
전체 청구항 수 : 총 4 항 심사관 : 윤혜숙

(54) 발명의 명칭 **악성 프로세스 감염을 차단하는 파일 백업 장치 및 그 방법**

(57) 요 약

본 발명의 실시예에 따른 파일 백업 장치는 일정 주기로 사용자 단말의 저장 수단 내 백업 대상 영역에 저장된 파일을 파일 백업 장치의 저장 수단 내 임시 저장 영역으로 복사하는 파일 복사부, 상기 임시 저장 영역에 복사된 파일의 악성 프로세스 감염 여부를 검사하는 감염 검사부, 그리고 검사 결과 상기 임시 저장 영역에 복사된 파일이 악성 프로세스에 감염되지 않았다고 판단되면, 상기 임시 저장 영역에 복사된 파일을 상기 파일 백업 장치의 저장 수단 내 파일 보관 영역에 저장하는 파일 보관부를 포함한다.

대 표 도 - 도1



(72) 발명자

이진우

서울특별시 동작구 상도로58길 12 삼미빌라 201호

김용민

서울특별시 동작구 상도로54길 14 104호(상도동)

이 발명을 지원한 국가연구개발사업

과제고유번호	2017-0-00388
부처명	과학기술정보통신부
연구관리전문기관	정보통신기술진흥센터
연구사업명	정보보호핵심원천기술개발사업
연구과제명	랜섬웨어 대응 기술 개발
기 여 율	1/1
주관기관	한양대학교 산학협력단
연구기간	2017.04.01 ~ 2018.12.31

명 세 서

청구범위

청구항 1

일정 주기로 사용자 단말의 저장 수단 내 백업 대상 영역에 저장된 파일을 파일 백업 장치의 저장 수단 내 임시 저장 영역으로 복사하는 파일 복사부,

상기 임시 저장 영역에 복사된 파일의 악성 프로세스 감염 여부를 검사하는 감염 검사부, 그리고

검사 결과 상기 임시 저장 영역에 복사된 파일이 악성 프로세스에 감염되지 않았다고 판단되면, 상기 임시 저장 영역에 복사된 파일을 상기 파일 백업 장치의 저장 수단 내 파일 보관 영역에 저장하는 파일 보관부, 그리고

상기 임시 저장 영역으로 복사가 완료되면 상기 백업 대상 영역을 언마운트(unmount)하여 상기 백업 대상 영역과 상기 임시 저장 영역의 연결을 해제하고, 검사 결과 상기 임시 저장 영역에 복사된 파일이 악성 프로세스에 감염되지 않았다고 판단되면 언마운트 상태의 상기 파일 보관 영역을 마운트(mount)하여 상기 임시 저장 영역에 연결시키며, 임시 저장 영역에 복사된 파일이 상기 파일 보관 영역에 저장되면, 상기 파일 보관 영역을 언마운트하여 상기 임시 저장 영역과 상기 파일 보관 영역의 연결을 해제하고, 상기 백업 대상 영역을 마운트하여 상기 백업 대상 영역과 상기 임시 저장 영역을 연결하는 연결 제어부를 포함하는 파일 백업 장치.

청구항 2

제1항에 있어서,

상기 백업 대상 영역에 저장된 파일은,

상기 악성 프로세스의 감염 대상이 되는 함정파일을 포함하며,

상기 감염 검사부는,

상기 함정파일이 상기 악성 프로세스에 의해 암호화되었는지 여부를 통해 상기 악성 프로세스 감염 여부를 검사하는 파일 백업 장치.

청구항 3

삭제

청구항 4

삭제

청구항 5

삭제

청구항 6

파일 백업 장치를 이용한 파일 백업 방법에 있어서,

상기 파일 백업 장치가 일정 주기로 사용자 단말의 저장 수단 내 백업 대상 영역에 저장된 파일을 상기 파일 백업 장치의 저장 수단 내 임시 저장 영역으로 복사하는 단계,

상기 임시 저장 영역으로 복사가 완료되면, 상기 파일 백업 장치가 백업 대상 영역을 언마운트(unmount)하여 상기 백업 대상 영역과 상기 임시 저장 영역의 연결을 해제하는 단계,

상기 파일 백업 장치가 상기 임시 저장 영역에 복사된 파일의 악성 프로세스 감염 여부를 검사하는 단계,

검사 결과 상기 임시 저장 영역에 복사된 파일이 악성 프로세스에 감염되지 않았다고 판단되면, 상기 파일 백업 장치가 언마운트 상태의 파일 보관 영역을 마운트(mount)하여 상기 임시 저장 영역에 연결시키고, 상기 파일 백업 장치가 상기 임시 저장 영역에 복사된 파일을 상기 파일 백업 장치의 저장 수단 내 파일 보관 영역에 저장하

는 단계, 그리고

상기 임시 저장 영역에 복사된 파일이 상기 파일 보관 영역에 저장되면, 상기 파일 백업 장치가 상기 파일 보관 영역을 언마운트(unmount)하여 상기 임시 저장 영역과 상기 파일 보관 영역의 연결을 해제하고, 상기 백업 대상 영역을 마운트하여 상기 백업 대상 영역과 상기 임시 저장 영역을 연결하는 단계를 포함하는 파일 백업 방법.

청구항 7

제6항에 있어서,

상기 백업 대상 영역에 저장된 파일은,

상기 악성 프로세스의 감염 대상이 되는 함정파일을 포함하며,

상기 악성 프로세스 감염 여부를 검사하는 단계는,

상기 함정파일이 암호화되었는지 여부를 통해 상기 악성 프로세스 감염 여부를 검사하는 파일 백업 방법.

청구항 8

삭제

청구항 9

삭제

청구항 10

삭제

발명의 설명

기 술 분 야

[0001] 실시 예는 파일 백업 시스템 및 그 방법에 관한 것이다.

배 경 기 술

[0002] 랜섬웨어는 사용자 디바이스 또는 네트워크 스토리지 디바이스의 저장소에 존재하는 파일들을 암호화하여 인질로 삼는 멀웨어(Malware)의 일종이다. 암호화된 파일을 원상태로 되돌리기 위해서는 복호화 키가 필요하며, 랜섬웨어 개발자는 이 복호화 키를 통해 암호화 된 파일들에 대해서 비용을 요구한다. 일반적으로 비용은 추적이 어렵다고 알려진 암호화폐(비트코인 등)를 이용한 전자 결제 방식을 통해서 지불하게 된다. 랜섬웨어는 대부분 이메일에 정상적인 파일로 가장된 상태로 첨부되어 배포되며, 첨부 파일을 열어보는 순간 랜섬웨어 프로그램이 활성화 된다. 랜섬웨어는 운영체제의 취약점이나 일반 사용자들이 사용하는 프로그램인 애플리케이션의 취약점을 이용하여 암호화를 수행하는 것이 일반적이다.

[0003] 일부 컴퓨터 운영체제의 경우 자체적으로 램섬웨어와 같은 악성 프로세스로부터 데이터를 보호하는 자체 보안 기능을 가진다. 자체 보안 기능이 없는 운영체제의 경우에는 함정파일을 이용하는 별도의 백신 프로그램을 사용하여 랜섬웨어를 방지한다. 최근 다년간의 성장을 통해 큰 규모로 성장한 클라우드 컴퓨팅을 통해서 랜섬웨어를 대응 하는 기술들도 등장하고 있다.

[0004] 하지만, 함정 파일을 두는 경우에는 그것들을 배치하는 경로가 랜섬웨어에 따라 가변적일 수 있고 어떤 함정 파일을 두느냐에 따라 랜섬웨어 탐지는 가능하지만 방지는 불가능 할 수 있다. 또한 클라우드 컴퓨팅을 통한 랜섬웨어 방지 솔루션은 그 클라우드 자체가 랜섬웨어에 감염된다면 기존 백업 파일 역시 랜섬웨어에 감염될 수 있는 문제가 발생한다.

발명의 내용

해결하려는 과제

[0005] 실시 예는 랜섬웨어를 포함하는 악성 프로세스가 백업 파일에 감염되는 것을 차단하는 파일 백업 장치 및 그 방

법을 제공하기 위한 것이다.

[0006] 실시 예에서 해결하고자 하는 과제는 이에 한정되는 것은 아니며, 아래에서 설명하는 과제의 해결수단이나 실시 형태로부터 파악될 수 있는 목적이나 효과도 포함된다고 할 것이다.

과제의 해결 수단

[0007] 본 발명의 실시예에 따른 파일 백업 장치는 일정 주기로 사용자 단말의 저장 수단 내 백업 대상 영역에 저장된 파일을 파일 백업 장치의 저장 수단 내 임시 저장 영역으로 복사하는 파일 복사부, 상기 임시 저장 영역에 복사된 파일의 악성 프로세스 감염 여부를 검사하는 감염 검사부, 그리고 검사 결과 상기 임시 저장 영역에 복사된 파일이 악성 프로세스에 감염되지 않았다고 판단되면, 상기 임시 저장 영역에 복사된 파일을 상기 파일 백업 장치의 저장 수단 내 파일 보관 영역에 저장하는 파일 보관부를 포함한다.

[0008] 상기 백업 대상 영역에 저장된 파일은, 상기 악성 프로세스의 감염 대상이 되는 함정파일을 포함하며, 상기 감염 검사부는, 상기 함정파일이 상기 악성 프로세스에 의해 암호화되었는지 여부를 통해 상기 악성 프로세스 감염 여부를 검사할 수 있다.

[0009] 상기 임시 저장 영역으로 복사가 완료되면, 상기 백업 대상 영역을 언마운트(unmount)하여 상기 백업 대상 영역과 상기 임시 저장 영역의 연결을 해제하는 연결 제어부를 더 포함할 수 있다.

[0010] 상기 연결 제어부는, 검사 결과 상기 임시 저장 영역에 복사된 파일이 악성 프로세스에 감염되지 않았다고 판단되면, 상기 파일 보관 영역을 마운트(mount)하여 상기 임시 저장 영역에 연결시킬 수 있다.

[0011] 상기 연결 제어부는, 상기 임시 저장 영역에 복사된 파일이 상기 파일 보관 영역에 저장되면, 상기 파일 보관 영역을 언마운트(unmount)하여 상기 임시 저장 영역과 상기 파일 보관 영역의 연결을 해제하고, 상기 백업 대상 영역을 마운트하여 상기 백업 대상 영역과 상기 임시 저장 영역을 연결할 수 있다.

[0012] 본 발명의 실시예에 따른 파일 백업 장치를 이용한 파일 백업 방법에 있어서, 파일 백업 방법은 상기 파일 백업 장치가 일정 주기로 사용자 단말의 저장 수단 내 백업 대상 영역에 저장된 파일을 상기 파일 백업 장치의 저장 수단 내 임시 저장 영역으로 복사하는 단계, 상기 파일 백업 장치가 상기 임시 저장 영역에 복사된 파일의 악성 프로세스 감염 여부를 검사하는 단계, 검사 결과 상기 임시 저장 영역에 복사된 파일이 악성 프로세스에 감염되지 않았다고 판단되면, 상기 파일 백업 장치가 상기 임시 저장 영역에 복사된 파일을 상기 파일 백업 장치의 저장 수단 내 파일 보관 영역에 저장하는 단계를 포함한다.

[0013] 상기 백업 대상 영역에 저장된 파일은, 상기 악성 프로세스의 감염 대상이 되는 함정파일을 포함하며, 상기 악성 프로세스 감염 여부를 검사하는 단계는, 상기 함정파일이 암호화되었는지 여부를 통해 상기 악성 프로세스 감염 여부를 검사할 수 있다.

[0014] 상기 임시 저장 영역으로 복사가 완료되면, 상기 파일 백업 장치가 상기 백업 대상 영역을 언마운트(unmount)하여 상기 백업 대상 영역과 상기 임시 저장 영역의 연결을 해제하는 단계를 더 포함할 수 있다.

[0015] 검사 결과 상기 임시 저장 영역에 복사된 파일이 악성 프로세스에 감염되지 않았다고 판단되면, 상기 파일 백업 장치가 상기 파일 보관 영역을 마운트(mount)하여 상기 임시 저장 영역에 연결시키는 단계를 더 포함할 수 있다.

[0016] 상기 임시 저장 영역에 복사된 파일이 상기 파일 보관 영역에 저장되면, 상기 파일 백업 장치가 상기 파일 보관 영역을 언마운트(unmount)하여 상기 임시 저장 영역과 상기 파일 보관 영역의 연결을 해제하고, 상기 백업 대상 영역을 마운트하여 상기 백업 대상 영역과 상기 임시 저장 영역을 연결할 수 있다.

발명의 효과

[0017] 본 발명의 실시예에 따르면, 파일의 악성 프로세스 감염 검사 및 파일 복사 저장 단계마다 보조 기억 장치 사이의 연결이나 보조 기억 장치 내 저장 영역의 연결을 유지하거나 해제함으로써 백업 시스템에 악성 프로세스가 감염되는 상황을 미연에 방지할 수 있다. 이에 따라, 사용자는 중요한 데이터를 보다 안전하게 보관할 수 있는 장점이 있다.

[0018] 본 발명의 다양하면서도 유익한 장점과 효과는 상술한 내용에 한정되지 않으며, 본 발명의 구체적인 실시형태를 설명하는 과정에서 보다 쉽게 이해될 수 있을 것이다.

도면의 간단한 설명

[0019] 도 1은 본 발명의 실시예에 따른 파일 백업 시스템의 구성도이다.

도 2는 본 발명의 실시예에 따른 파일 백업 장치의 구성도이다.

도 3은 본 발명의 실시예에 따른 파일 백업 방법의 순서도이다.

도 4는 본 발명의 실시예에 따른 연결 제어부가 각 영역의 연결을 제어하는 과정을 설명하기 위한 도면이다.

발명을 실시하기 위한 구체적인 내용

[0020] 본 발명은 다양한 변경을 가할 수 있고 여러 가지 실시예를 가질 수 있는 바, 특정 실시예들을 도면에 예시하고 설명하고자 한다. 그러나, 이는 본 발명을 특정한 실시 형태에 대해 한정하려는 것이 아니며, 본 발명의 사상 및 기술 범위에 포함되는 모든 변경, 균등물 내지 대체물을 포함하는 것으로 이해되어야 한다.

[0021] 제2, 제1 등과 같이 서수를 포함하는 용어는 다양한 구성요소들을 설명하는데 사용될 수 있지만, 상기 구성요소들은 상기 용어들에 의해 한정되지는 않는다. 상기 용어들은 하나의 구성요소를 다른 구성요소로부터 구별하는 목적으로만 사용된다. 예를 들어, 본 발명의 권리 범위를 벗어나지 않으면서 제2 구성요소는 제1 구성요소로 명명될 수 있고, 유사하게 제1 구성요소도 제2 구성요소로 명명될 수 있다. 및/또는 이라는 용어는 복수의 관련된 기재된 항목들의 조합 또는 복수의 관련된 기재된 항목들 중의 어느 항목을 포함한다.

[0022] 어떤 구성요소가 다른 구성요소에 "연결되어" 있다거나 "접속되어" 있다고 언급된 때에는, 그 다른 구성요소에 직접적으로 연결되어 있거나 또는 접속되어 있을 수도 있지만, 중간에 다른 구성요소가 존재할 수도 있다고 이해되어야 할 것이다. 반면에, 어떤 구성요소가 다른 구성요소에 "직접 연결되어" 있다거나 "직접 접속되어" 있다고 언급된 때에는, 중간에 다른 구성요소가 존재하지 않는 것으로 이해되어야 할 것이다.

[0023] 본 출원에서 사용한 용어는 단지 특정한 실시예를 설명하기 위해 사용된 것으로, 본 발명을 한정하려는 의도가 아니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 출원에서, "포함하다" 또는 "가지다" 등의 용어는 명세서상에 기재된 특징, 숫자, 단계, 동작, 구성요소, 부품 또는 이들을 조합한 것이 존재함을 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성요소, 부품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.

[0024] 다르게 정의되지 않는 한, 기술적이거나 과학적인 용어를 포함해서 여기서 사용되는 모든 용어들은 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가지고 있다. 일반적으로 사용되는 사전에 정의되어 있는 것과 같은 용어들은 관련 기술의 문맥 상 가지는 의미와 일치하는 의미를 가지는 것으로 해석되어야 하며, 본 출원에서 명백하게 정의하지 않는 한, 이상적이거나 과도하게 형식적인 의미로 해석되지 않는다.

[0025] 이하, 첨부된 도면을 참조하여 실시예를 상세히 설명하되, 도면 부호에 관계없이 동일하거나 대응하는 구성 요소는 동일한 참조 번호를 부여하고 이에 대한 중복되는 설명은 생략하기로 한다.

[0026] 도 1은 본 발명의 실시예에 따른 파일 백업 시스템의 구성도이다.

[0027] 도 1에 도시된 바와 같이, 본 발명의 실시예에 따른 파일 백업 시스템은 사용자 단말(100)과 파일 백업 장치(200)를 포함한다.

[0028] 본 발명의 실시예에 따른 파일 백업 시스템은 사용자 단말(100)에 저장된 파일을 파일 백업 장치(200)에 저장할 때 랜섬웨어(ransomware)와 같은 악성 프로세스(malware)의 감염이 없는 파일만을 선택적으로 보관할 수 있다. 이를 통해, 본 발명의 실시예에 따른 파일 백업 시스템은 파일 보관의 안전성을 높일 수 있다.

[0029] 그러면, 본 발명의 실시예에 따른 파일 백업 시스템의 각 구성에 대해 구체적으로 살펴보도록 한다.

[0030] 우선, 사용자 단말(100)은 사용자가 데이터 처리 시스템과 교신하기 위한 입출력 장치를 의미한다. 사용자 단말은 휴대폰(mobile phone), 스마트 폰(smart phone), 노트북 컴퓨터(notebook computer), 데스크탑 컴퓨터(desktop computer), PDA(personal digital assistants), PMP(portable multimedia player), 태블릿 PC(tablet PC)와 같은 장치를 통해 구현될 수 있다.

[0031] 사용자 단말(100)은 파일 시스템(110) 및 저장 수단(120)을 포함한다.

[0032] 파일 시스템(file system)은 운영 체제에서 보조 기억 장치와 그 안에 저장되는 파일을 관리하는 시스템을 의미한다. 본 발명의 실시예에서, 사용자 단말(100)의 파일 시스템(110)은 사용자 단말(100)의 운영 체제에서 저장 수단(120) 및 저장 수단(120)에 저장되는 파일을 관리하는 시스템을 의미할 수 있다.

[0033] 저장 수단(120)은 파일을 저장하는 보조 기억 장치를 의미한다. 보조 기억 장치는 중앙처리장치가 아닌 외부에서 프로그램이나 데이터를 보관하기 위한 기억장치를 의미한다. 따라서, 저장 수단은 HDD(Hard Disk Drive), SSD(Solid State Disk), USB(Universal Serial Bus), 자기 테이프, 자기 디스크, 레이저 디스크, 광자기디스크, 플로피 디스크, CD-ROM 등을 포함할 수 있다.

[0034] 사용자 단말(100)의 저장 수단(120)은 사용자 단말(100)의 파일 시스템(110)에 의해 관리될 수 있다. 사용자가 사용자 단말(100)을 통해 파일을 생성하면, 파일 시스템(110)은 저장 수단의 특정 영역에 파일을 저장할 수 있다. 여기서, 특정 영역이란 디렉토리나 폴더, 논리 드라이브를 의미할 수 있다. 뿐만 아니라, 사용자 단말(100)의 저장 수단이 복수일 경우, 물리적으로 분리된 하나 이상의 저장 수단일 수도 있다.

[0035] 사용자 단말(100)은 저장 수단(120) 내에 어느 특정 영역을 백업 대상 영역(121)으로 설정할 수 있다. 구체적으로, 사용자 단말(100)의 파일 시스템(110)은 저장 수단(120) 내에 어느 특정 영역을 백업 대상 영역(121)으로 설정할 수 있다. 여기서, 백업 대상 영역(121)이란 본 발명의 실시예에 따른 파일 백업 장치(200)로 전송될 대상 파일이 저장된 공간을 의미한다. 예를 들어, 사용자 단말(100)은 “c:₩Un-safe region₩”이라는 폴더를 백업 대상 영역(121)으로 설정할 수 있으며, “c:₩Un-safe region₩” 폴더 내에 저장된 파일은 파일 백업 장치(200)로 전송될 수 있다. 따라서, 백업 대상 영역(121) 이외의 영역에 저장된 파일은 파일 백업 장치(200)로 전송될 수 없다.

[0036] 사용자 단말(100)은 저장 수단(120) 내 백업 대상 영역(121)에 함정파일(미끼파일)을 생성할 수 있다. 함정파일은 악성 프로세스의 감염 대상이 파일을 의미한다. 따라서, 백업 대상 영역(121)에 저장된 파일이 파일 백업 장치(200)로 전송될 때, 함정파일도 함께 전송될 수 있다.

[0037] 함정 파일은 악성 프로세스의 정책 기초하여 생성될 수 있다. 예를 들어, 랜섬웨어가 사용자 단말의 저장 수단 내 최상위 경로부터 암호화를 진행하는 정책을 가진다고 가정한다. 그러면, 사용자 단말은 파일명이나 폴더명 앞에 특수기호(%, $, !, @ 등)가 배치되는 함정파일(decoy file)을 생성할 수 있다.

[0038] 다음으로, 파일 백업 장치(200)는 사용자 단말(100)과 통신하여 파일을 수신 및 저장할 수 있는 입출력 장치를 의미한다. 본 발명의 실시예에 따르면, 파일 백업 장치(200)는 서버(sever)의 형태로 구현될 수 있으며, 서버의 기능을 구현할 수 있는 장치를 모두 포함할 수 있다. 파일 백업 장치(200)는 클라우드(Cloud), NAS(Network Attached Storage), 데스크탑 컴퓨터, 노트북 컴퓨터, 태블릿 PC 등을 포함할 수 있다.

[0039] 파일 백업 장치(200)는 파일 시스템(210) 및 저장 수단(220)을 포함한다.

[0040] 파일 시스템(210)은 운영 체제에서 보조 기억 장치와 그 안에 저장되는 파일을 관리하는 시스템을 의미한다. 본 발명의 실시예에서, 파일 백업 장치(200)의 파일 시스템(210)은 파일 백업 장치(200)의 운영 체제에서 저장 수단(220) 및 저장 수단(220)에 저장되는 파일을 관리하는 시스템을 의미할 수 있다.

[0041] 저장 수단(220)은 파일을 저장하는 보조 기억 장치를 의미한다. 보조 기억 장치는 중앙처리장치가 아닌 외부에서 프로그램이나 데이터를 보관하기 위한 기억장치를 의미한다. 따라서, 저장 수단(220)은 HDD(Hard Disk Drive), SSD(Solid State Disk), USB(Universal Serial Bus), 자기 테이프, 자기 디스크, 레이저 디스크, 광자기디스크, 플로피 디스크, CD-ROM 등을 포함할 수 있다.

[0042] 파일 백업 장치(200)의 저장 수단(220)은 파일 백업 장치(200)의 파일 시스템(210)에 의해 관리될 수 있다. 파일 백업 장치(200)의 파일 시스템은 사용자 단말(100)로부터 수신한 파일을 저장 수단(220)의 특정 영역에 저장할 수 있다. 여기서, 특정 영역이란 디렉토리나 폴더, 논리 드라이브를 의미할 수 있다. 뿐만 아니라, 파일 백업 장치(200)의 저장 수단(220)이 복수일 경우, 물리적으로 분리된 하나 이상의 저장 수단(220)일 수도 있다.

[0043] 파일 백업 장치(200)는 저장 수단(220) 내에 어느 특정 영역을 임시 저장 영역(221)으로 설정할 수 있다. 예를 들어, 파일 백업 장치(200)의 파일 시스템(210)은 저장 수단(220) 내에 어느 특정 영역을 임시 저장 영역(221)으로 설정할 수 있다. 여기서, 임시 저장 영역(221)이란 사용자 단말(100)의 백업 대상 영역(121)에 저장된 파일을 복사하여 저장하는 영역을 의미한다. 예를 들어, 파일 백업 장치(200)는 “c:₩Middle region₩”이라는 폴더를 임시 저장 영역(221)으로 설정할 수 있으며, 사용자 단말(100)로부터 전송된 파일은 “c:₩Middle region₩” 폴더 내에 저장될 수 있다.

[0044] 또한, 파일 백업 장치(200)는 저장 수단(220) 내에 어느 특정 영역을 파일 보관 영역(222)으로 설정할 수 있다. 예를 들어, 파일 백업 장치(200)의 파일 시스템(210)은 저장 수단(220) 내에 어느 특정 영역을 파일 보관 영역(222)으로 설정할 수 있다. 여기서, 파일 보관 영역(222)이란 임시 저장 영역(221)에 저장된 파일 중 악성 프로세스가 감염되지 않은 파일을 저장하는 영역을 의미한다. 예를 들어, 파일 백업 장치(200)는 “c:₩Safe region₩”이라는 폴더를 파일 보관 영역(222)으로 설정할 수 있으며, 악성 프로세스의 감염이 되지 않았다고 검증된 파일을 “c:₩Safe region₩” 폴더 내에 저장할 수 있다.

[0045] 본 발명의 실시예에 따르면, 사용자 단말과 파일 백업 장치는 통신망(10)을 통해 연결될 수 있으며, SMB(server message block) 프로토콜이나 CIFS(common Internet file system) 프로토콜을 통해 연결될 수 있다. 예를 들어, 파일 백업 장치는 SMB 프로토콜을 이용하여 사용자 단말과 통신 연결할 수 있으며, SMB 프로토콜 및 CIFS 프로토콜 기반의 삼바(samba)를 통해 백업 대상 영역의 파일을 전송받아 임시 저장 영역에 복사할 수 있다. 이는 본 발명의 일 실시예로서, 사용자 단말과 파일 백업 장치의 파일 시스템에 따라 프로토콜은 달라질 수 있다.

[0046] 그러면, 도 2를 참조하여, 본 발명의 실시예에 따른 파일 백업 장치에 대해 살펴보도록 한다.

[0047] 도 2는 본 발명의 실시예에 따른 파일 백업 장치의 구성도이다.

[0048] 도 2에 도시된 바와 같이, 본 발명의 실시예에 따른 파일 백업 장치(200)는 파일 시스템(210)과 저장 수단(220)을 포함하며, 파일 시스템(210)은 파일 복사부(211), 감염 검사부(212), 파일 보관부(213) 및 연결 제어부(214)를 포함한다.

[0049] 도 2에서는 파일 복사부(211), 감염 검사부(212), 파일 보관부(213) 및 연결 제어부(214)가 파일 시스템(210)에 포함된 구성으로 설명하고 있으나, 이에 한정되지 않으며, 파일 시스템(210)과 다른 별도의 시스템으로 구현될 수도 있다. 저장 수단(220)의 경우 도 1을 통해 설명하였는바, 상세한 설명은 생략하도록 하며, 아래에서는 파일 시스템(210)의 구성에 관해 살펴보도록 한다.

[0050] 우선, 파일 복사부(211)는 일정 주기로 사용자 단말의 저장 수단 내 백업 대상 영역에 저장된 파일을 파일 백업 장치(200)의 저장 수단(220) 내 임시 저장 영역으로 복사한다.

[0051] 이때, 파일 전송은 일정 주기마다 진행된다. 예를 들어, 일정 주기가 24시간인 경우, 24시간마다 파일 전송이 진행된다. 일정 주기는 사용자가 사용자 단말(100)을 통해 설정될 수 있다. 사용자가 파일을 생성하는 빈도나 저장되는 파일의 중요도에 따라 전송 주기를 설정할 수 있으므로, 파일 보관의 안전성을 높일 수 있다.

[0052] 다음으로, 감염 검사부(212)는 임시 저장 영역에 복사된 파일의 악성 프로세스 감염 여부를 검사한다. 감염 검사부(212)는 함정파일이 악성 프로세스에 의해 암호화되었는지 여부를 통해 악성 프로세스 감염 여부를 검사할 수 있다.

[0053] 다음으로, 파일 보관부(213)는 검사 결과 임시 저장 영역에 복사된 파일이 악성 프로세스에 감염되지 않았다고 판단되면, 임시 저장 영역에 복사된 파일을 파일 백업 장치(200)의 저장 수단(220) 내 파일 보관 영역에 저장한다.

[0054] 다음으로, 연결 제어부(214)는 임시 저장 영역으로 복사가 완료되면, 사용자 단말과의 연결을 해제한다. 이때, 연결 제어부(214)는 사용자 단말(100)의 파일 시스템과 파일 백업 장치(200)의 파일 시스템 간 연결을 의미할 수 있다. 즉, 연결 제어부(214)는 사용자 단말(100)의 백업 대상 영역을 언마운트하여, 백업 대상 영역과 임시 저장 영역의 연결을 해제할 수 있다.

[0055] 연결 제어부(214)는 검사 결과 임시 저장 영역에 복사된 파일이 악성 프로세스에 감염되지 않았다고 판단되면, 파일 보관 영역을 마운트(mount)하여 임시 저장 영역에 연결시킨다.

[0056] 연결 제어부(214)는 임시 저장 영역에 복사된 파일이 파일 보관 영역에 저장되면, 파일 보관 영역을 언마운트(unmount)하여 임시 저장 영역과 파일 보관 영역의 연결을 해제하고, 사용자 단말과의 통신을 연결한다.

[0057] 그러면, 도 3을 통해 본 발명의 실시예에 따른 파일 백업 방법에 대해 살펴보도록 한다.

[0058] 도 3은 본 발명의 실시예에 따른 파일 백업 방법의 순서도이다.

[0059] 도 3을 참조하면, 우선, 사용자 단말(100)은 사용자 단말(100)의 저장 수단 내 백업 대상 영역을 설정한 후, 백업 대상 영역에 함정파일을 생성한다(S5).

[0060] 그리고, 사용자 단말(100)은 백업 대상 영역에 저장된 파일을 파일 백업 장치(200)로 전송한다(S10). 백업 대상

영역에는 사용자의 사용자 단말(100) 사용에 따라 생성된 파일 뿐만 아니라, S5 단계에서 생성된 함정파일도 포함된다.

[0061] 다음으로, 파일 백업 장치(200)는 전송된 파일, 즉 백업 대상 영역에 저장된 파일을 파일 백업 장치(200)의 저장 수단 내 임시 저장 영역으로 복사한다(S15). 이때, 복사된 파일에는 S5 단계에서 생성된 함정파일도 포함된다.

[0062] 임시 저장 영역으로의 복사가 완료되면, 파일 백업 장치(200)는 사용자 단말(100)과의 연결을 해제한다(S20). S25 단계 등을 수행하는 중 사용자 단말(100)에 악성 프로세스가 감염될 수 있는데, 이때 사용자 단말(100)의 백업 대상 영역과 파일 백업 장치(200)의 임시 저장 영역이 계속 연결되어 있으면 S15 단계 이후에 파일 백업 장치(200)가 악성 프로세스에 감염될 수 있다. 예를 들어, S25 단계에서 악성 프로세스 감염 검사가 완료되어 악성 프로세스의 감염이 없다고 판단된 후 복사된 파일이 악성 프로세스에 감염될 수 있다. 이로 인해, 파일 백업 장치(200)에 저장되어 있던 파일들까지 악성 프로세스에 감염될 수 있는 문제점이 발생한다. 따라서, 본 발명의 실시예에 따른 파일 백업 장치(200)는 S20 단계에서 파일 백업 장치(200)와 사용자 단말(100) 간 연결을 해제함으로써 상기의 문제점을 미연에 차단하여 파일 보관의 안전성을 높인다.

[0063] S20 단계에서, 파일 백업 장치(200)와 사용자 단말(100) 간 통신 연결이 해제되면, 파일 백업 장치(200)는 임시 저장 영역에 복사된 파일의 악성 프로세스 감염 여부를 검사하여(S25, S30).

[0064] 검사 결과, 임시 저장 영역에 복사된 파일에 악성 프로세스가 감염되지 않았다고 판단되면, 파일 백업 장치(200)는 파일 보관 영역을 마운트(mount)하여 임시 저장 영역에 연결한다(S35).

[0065] 즉, 파일 보관 영역에는 임시 저장 영역에 복사된 파일에 악성 프로세스가 감염되지 않았다고 판단될 때까지 접근할 수 없다. 따라서, 임시 저장 영역에 복사된 파일에 악성 프로세스가 감염되었다고 하더라도, 파일 보관 영역에 저장된 파일은 악성 프로세스의 감염을 피할 수 있는 장점이 있다.

[0066] 파일 보관 영역이 마운트되면, 파일 백업 장치(200)는 임시 저장 영역에 복사된 파일을 파일 보관 영역에 저장한다(S40).

[0067] 그리고, 파일 보관 영역에 저장이 완료되면, 파일 백업 장치(200)는 파일 보관 영역을 언마운트(unmount)하여 임시 저장 영역과 파일 보관 영역의 연결을 해제한다(S45).

[0068] 즉, 파일 보관 영역으로의 저장이 완료되면, 악성 프로세스가 접근할 수 없도록 임시 저장 영역과 파일 보관 연결의 연결을 해제하여, 파일 보관의 안전성을 높인다.

[0069] 그리고, 파일 백업 장치(200)는 사용자 단말(100)과 재연결한다(S50). 구체적으로, 파일 백업 장치(200)는 사용자 단말(100)의 백업 대상 영역을 마운트하여 임시 저장 영역과 재연결을 수행한다. 파일 백업 장치(200)와 사용자 단말(100) 사이의 연결은 다음 주기에서 임시 저장 영역에 파일이 복사될 때까지 계속된다.

[0070] 한편, S25 및 S30 단계의 검사 결과, 임시 저장 영역에 복사된 파일에 악성 프로세스가 감염되었다고 판단되면, 파일 백업 장치(200)는 임시 저장 영역에 복사된 파일을 삭제할 수 있다(S55).

[0071] S55 단계에서 파일의 삭제가 완료되면, 파일 백업 장치(200)는 사용자 단말(100)로 백업 대상 영역에 악성 프로세스가 감염되었음을 통지할 수 있다(S60). 이 경우, 예를 들어, 사용자 단말(100)이 백업 대상 영역에 대한 악성 프로세스 삭제나 치료 등의 조치를 취하기 전까지 파일 백업 장치(200)는 사용자 단말(100)의 백업 대상 영역을 마운트하지 않는다.

[0072] S10 내지 S60 단계는 일정 주기로 반복될 수 있다.

[0073] 도 4는 본 발명의 실시예에 따른 연결 제어부가 각 영역의 연결을 제어하는 과정을 설명하기 위한 도면이다.

[0074] 우선, 파일 백업이 수행되지 않는 동안에는, 도 4의 (a)에서와 같이, 사용자 단말(100)의 백업 대상 영역(121)과 파일 백업 장치(200)의 임시 저장 영역(221)은 서로 연결된다. 즉, 파일 백업 장치(200)는 사용자 단말(100)의 백업 대상 영역(121)을 마운트하여 백업 대상 영역(121)과 임시 저장 영역(221)을 연결한다. 하지만, 파일 백업 장치(200) 내 임시 저장 영역(221)과 파일 보관 영역(222)은 서로 연결되지 않는다.

[0075] 다음으로, 도 3의 S15단계가 완료된 후에는, 도 4의 (b)에서와 같이, 사용자 단말(100)의 백업 대상 영역(121)과 파일 백업 장치(200)의 임시 저장 영역(221)은 연결이 해제된다. 즉, 파일 백업 장치(200)는 사용자 단말(100)의 백업 대상 영역(121)을 언마운트하여 백업 대상 영역(121)과 임시 저장 영역(221)을 연결을 해제한다.

이때, 파일 백업 장치(200) 내 임시 저장 영역(221)과 파일 보관 영역(222)은 서로 연결되지 않는다.

[0076] 그리고, 도 3의 S30 단계가 종료되고 악성 프로세스의 감염이 없다고 판단된 후에는 도 4의 (c)에서와 같이, 파일 백업 장치(200) 내 임시 저장 영역(221)과 파일 보관 영역(222)은 서로 연결된다. 이때, 사용자 단말(100)과 파일 백업 장치(200)는 연결이 되지 않는다.

[0077] 다음으로, 도 3의 S40 단계가 종료되면, 도 4의 (d)에서와 같이, 파일 백업 장치(200) 내 임시 저장 영역(221)과 파일 보관 영역(222)의 연결은 해제된다. 그리고, 사용자 단말(100)과 파일 백업 장치(200)는 연결된다. 즉, 도 4의 (a)와 같은 연결 상태가 된다.

[0078] 본 발명의 실시예에 따르면, 파일의 악성 프로세스 감염 검사 및 파일 복사 저장 단계마다 보조 기억 장치 사이의 연결이나 보조 기억 장치 내 저장 영역의 연결을 유지하거나 해제함으로써 백업 시스템에 악성 프로세스가 감염되는 상황을 미연에 방지할 수 있다. 이에 따라, 사용자는 중요한 데이터를 보다 안전하게 보관할 수 있는 장점이 있다.

[0079] 본 실시예에서 사용되는 '~부'라는 용어는 소프트웨어 또는 FPGA(field-programmable gate array) 또는 ASIC과 같은 하드웨어 구성요소를 의미하며, '~부'는 어떤 역할들을 수행한다. 그렇지만 '~부'는 소프트웨어 또는 하드웨어에 한정되는 의미는 아니다. '~부'는 어드레싱할 수 있는 저장 매체에 있도록 구성될 수도 있고 하나 또는 그 이상의 프로세서들을 재생시키도록 구성될 수도 있다. 따라서, 일 예로서 '~부'는 소프트웨어 구성요소들, 객체지향 소프트웨어 구성요소들, 클래스 구성요소들 및 태스크 구성요소들과 같은 구성요소들과, 프로세스들, 함수들, 속성들, 프로시저들, 서브루틴들, 프로그램 코드의 세그먼트들, 드라이버들, 펌웨어, 마이크로코드, 회로, 데이터, 데이터베이스, 데이터 구조들, 테이블들, 어레이들, 및 변수들을 포함한다. 구성요소들과 '~부'들 안에서 제공되는 기능은 더 작은 수의 구성요소들 및 '~부'들로 결합되거나 추가적인 구성요소들과 '~부'들로 더 분리될 수 있다. 뿐만 아니라, 구성요소들 및 '~부'들은 디바이스 또는 보안 멀티미디어카드 내의 하나 또는 그 이상의 CPU들을 재생시키도록 구현될 수도 있다.

[0080] 이상에서 실시예를 중심으로 설명하였으나 이는 단지 예시일 뿐 본 발명을 한정하는 것이 아니며, 본 발명이 속하는 분야의 통상의 지식을 가진 자라면 본 실시예의 본질적인 특성을 벗어나지 않는 범위에서 이상에 예시되지 않은 여러 가지의 변형과 응용이 가능함을 알 수 있을 것이다. 예를 들어, 실시예에 구체적으로 나타난 각 구성요소는 변형하여 실시할 수 있는 것이다. 그리고 이러한 변형과 응용에 관계된 차이점들은 첨부된 청구 범위에서 규정하는 본 발명의 범위에 포함되는 것으로 해석되어야 할 것이다.

부호의 설명

[0081]

100 : 사용자 단말

200 : 파일 백업 장치

210 : 파일 복사부

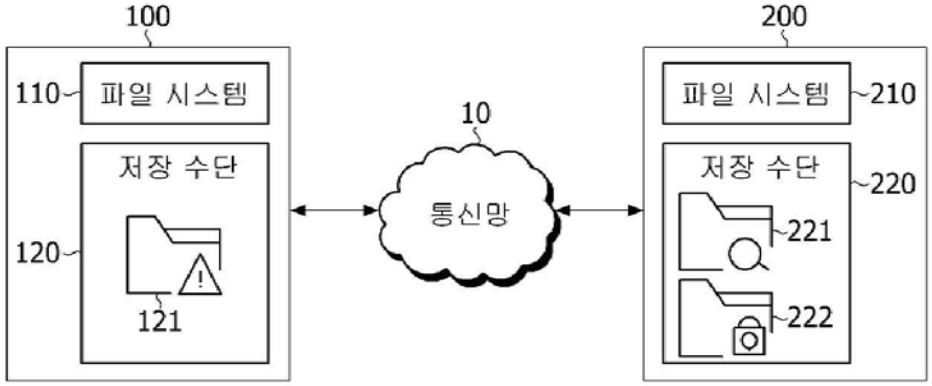
220 : 감염 검사부

230 : 파일 보관부

240 : 제어부

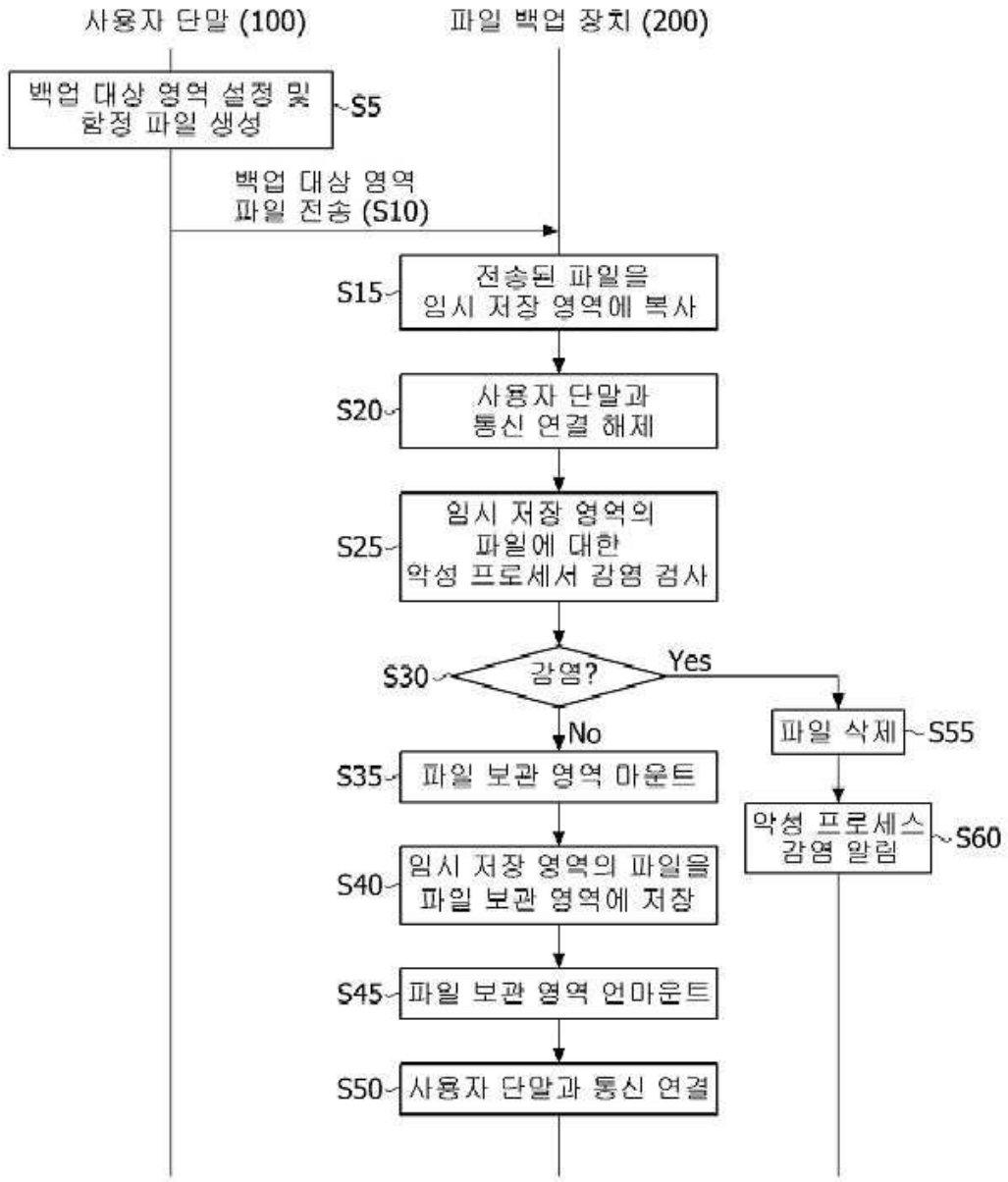
도면

도면1

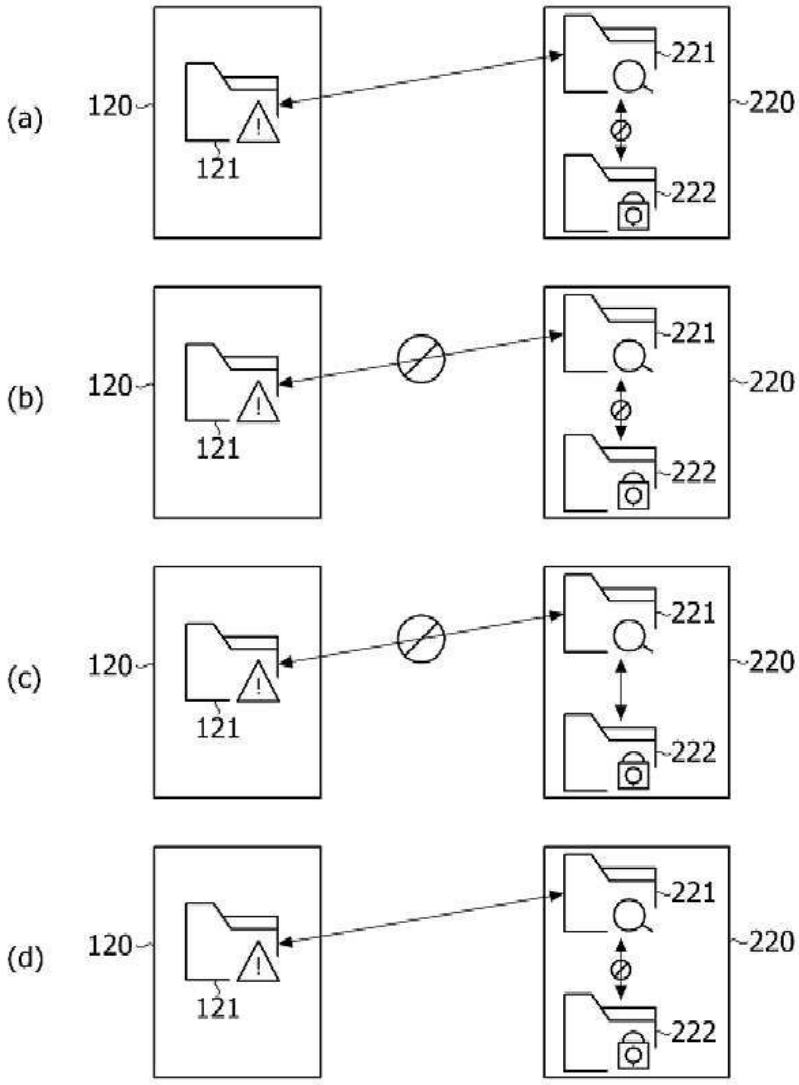


도면2

도면3


사용자 단말 (100)
파일 백업 장치 (200)
백업 대상 영역 설정 및 함정 파일 생성
S5
백업 대상 영역 파일 전송 (S10)
S15
전송된 파일을 임시 저장 영역에 복사
S20
사용자 단말과 통신 연결 해제
S25
임시 저장 영역의 파일에 대한 악성 프로세서 감염 검사
S30
감염?
Yes
No
파일 삭제
S55
악성 프로세스 감염 알림
S60
S35
파일 보관 영역 마운트
S40
임시 저장 영역의 파일을 파일 보관 영역에 저장
S45
파일 보관 영역 언마운트
S50
사용자 단말과 통신 연결

도면4


(a)
120
121
221
220
222
(b)
120
121
221
220
222
(c)
120
121
221
220
222
(d)
120
121
221
220
222

(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2019년08월19일
(11) 등록번호 10-2011725
(24) 등록일자 2019년08월12일

(51) 국제특허분류(Int. Cl.)
G06F 21/56 (2013.01) *G06F 16/00* (2019.01)
(52) CPC특허분류
G06F 21/56 (2013.01)
G06F 16/35 (2019.01)
(21) 출원번호 10-2017-0182896
(22) 출원일자 2017년12월28일
심사청구일자 2017년12월28일
(65) 공개번호 10-2019-0080445
(43) 공개일자 2019년07월08일
(56) 선행기술조사문헌
KR101541526 B1
KR1020150056244 A
KR1020160028724 A
KR1020160009980 A

(73) 특허권자
숭실대학교산학협력단
서울특별시 동작구 상도로 369 (상도동)
(72) 발명자
홍지만
서울특별시 서초구 나루터로4길 49, 332동 701호
김경민
서울특별시 동작구 상도로67길 12-13, 303호
(뒷면에 계속)
(74) 대리인
윤귀상

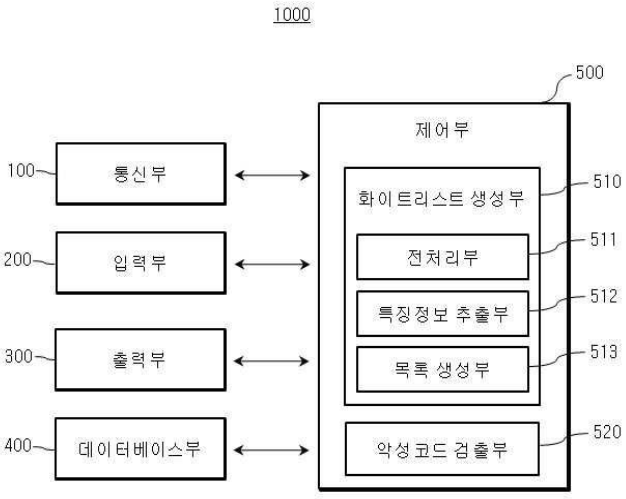
전체 청구항 수 : 총 11 항 심사관 : 윤혜숙

(54) 발명의 명칭 **악성코드 검출을 위한 화이트리스트 구축 방법 및 이를 수행하기 위한 기록매체 및 장치**

(57) 요 약

본 발명에 따른 악성코드 검출을 위한 화이트리스트 구축 방법은, 기준 애플리케이션을 구성하는 써드파티 라이브러리의 확장자를 안드로이드 가상 머신에서 실행되는 확장자로 컴파일하고, 컴파일된 써드파티 라이브러리의 클래스 및 메소드에 대하여, 변경되기 어려운 정보들에 대한 특징정보를 추출하여 화이트리스트를 생성함으로써, 악성코드 분석 대상이 되는 소스코드의 양을 감소시킬 수 있다.

대 표 도 - 도1



(72) 발명자

허만우

서울특별시 성북구 정릉로10길 108, 2동 109호 (정릉동, 영빈빌라)

이한솔

서울특별시 관악구 관악로14길 108-5

이 발명을 지원한 국가연구개발사업

과제고유번호 2017-0-00388

부처명 과학기술정보통신부

연구관리전문기관 정보통신기술진흥센터

연구사업명 정보통신 방송 연구개발

연구과제명 랜섬웨어 대응 기술 개발

기 여 율 1/1

주관기관 한양대학교 산학협력단

연구기간 2017.04.01 ~ 2018.12.31

공지예외적용 : 있음

명 세 서

청구범위

청구항 1

악성 코드 검출 장치에서 수행되는 악성코드 검출을 위한 화이트리스트 구축 방법에 있어서,

화이트리스트 구축을 위한 기준 애플리케이션을 구성하는 써드파티 라이브러리의 확장자를 안드로이드 가상 머신에서 실행되는 확장자로 컴파일하는 단계; 및

컴파일된 써드파티 라이브러리의 클래스 및 메소드에 대하여, 연산 코드의 해시값, 메소드 프로토타입의 인수 개수, 클래스 접근 제어자, 메소드가 포함된 클래스의 깊이정보 및 메소드 접근 제어자 중 적어도 하나를 포함하는 특징정보를 추출하여 화이트리스트를 생성하는 단계를 포함하는, 악성코드 검출을 위한 화이트리스트 구축 방법.

청구항 2

◈청구항 2은(는) 설정등록료 납부시 포기되었습니다.◈

제1항에 있어서,

상기 안드로이드 가상 머신에서 실행되는 파일 형식은,

안드로이드 달빅 가상 머신에서 동작하는 파일 형식인 DEX인 것을 특징으로 하는, 악성코드 검출을 위한 화이트리스트 구축 방법.

청구항 3

제1항에 있어서,

상기 특징정보를 추출하는 것은,

상기 컴파일된 상기 써드파티 라이브러리인 바이트코드에서 피연산자를 제거하여 상기 연산 코드를 추출하고, 상기 연산 코드를 미리 정해진 해시함수를 이용하여 해시값으로 변환하는 것을 특징으로 하는, 악성코드 검출을 위한 화이트리스트 구축 방법.

청구항 4

제1항에 있어서,

상기 특징정보를 추출하는 것은,

상기 컴파일된 상기 써드파티 라이브러리의 메소드의 이름, 반환값의 유형 및 인수를 정의하는 메소드 프로토타입을 분석하여 상기 메소드 프로토타입의 인수 개수를 산출하는 것을 특징으로 하는, 악성코드 검출을 위한 화이트리스트 구축 방법.

청구항 5

제1항에 있어서,

상기 특징정보를 추출하는 것은,

상기 기준 애플리케이션의 생성 시 설정되는 상기 컴파일된 상기 써드파티 라이브러리의 클래스 접근 레벨 및

클래스 멤버의 특성을 정의하는 상기 클래스 접근 제어자를 검색하는 것을 특징으로 하는, 악성코드 검출을 위한 화이트리스트 구축 방법.

청구항 6

제1항에 있어서,

상기 특징정보를 추출하는 것은,

상기 기준 애플리케이션의 생성 시 설정되는 클래스별 계층정보를 기초로 상기 컴파일된 상기 써드파티 라이브러리의 클래스 계층에 대한 상기 깊이정보를 추출하는 것을 특징으로 하는, 악성코드 검출을 위한 화이트리스트 구축 방법.

청구항 7

제1항에 있어서,

상기 화이트리스트를 생성하는 것은,

상기 깊이정보 및 상기 메소드 접근 제어자를 클래스 정보 테이블로 설정하고, 상기 연산 코드의 해시값 및 메소드 프로토타입의 인수 개수를 메소드 정보 테이블로 설정하는 것을 특징으로 하는, 악성코드 검출을 위한 화이트리스트 구축 방법.

청구항 8

제7항에 있어서,

상기 화이트리스트를 생성하는 것은,

악성코드 분석을 위한 검사 대상 애플리케이션을 클래스 정보 테이블 및 상기 메소드 정보 테이블과 비교하는 과정에서 어느 하나의 코드정보가 일치하는 것으로 탐지되면, 상기 코드정보의 라이브러리 버전 및 라이브러리 출시일을 포함하는 추가정보를 제공하는 라이브러리 정보 테이블 및 패키지 정보 테이블을 더 설정하는 것을 특징으로 하는, 악성코드 검출을 위한 화이트리스트 구축 방법.

청구항 9

제1항에 있어서,

검사 대상 애플리케이션의 라이브러리를 상기 화이트리스트에 포함된 특징정보와 비교하는 단계; 및

상기 검사 대상 애플리케이션을 구성하는 복수의 코드정보 중 상기 특징정보와 일치하는 코드정보를 제외한 나머지 코드정보들에 대한 악성코드 분석을 수행하는 단계를 더 포함하는, 악성코드 검출을 위한 화이트리스트 구축 방법.

청구항 10

제9항에 있어서,

상기 특징정보와 비교하는 것은,

상기 검사 대상 애플리케이션의 라이브러리를 구성하는 어느 하나의 코드정보의 깊이정보 및 접근 제어자를 상기 화이트리스트에 저장된 복수의 상기 깊이정보 및 상기 클래스 접근 제어자와 비교하여 일치하는지를 확인하고, 일치하는 클래스가 존재하면 상기 어느 하나의 코드정보의 연산코드의 해시값, 메소드 프로토타입의 인수 개수 및 메소드 접근 제어자를 상기 화이트리스트에 저장된 복수의 상기 연산 코드의 해시값, 상기 메소드 프로

토타입의 인수 개수 및 상기 메소드 접근 제어자와 비교하여 일치하는지를 확인하여 일치하는 메소드가 발견되는 경우, 상기 어느 하나의 코드정보를 상기 특징정보와 일치하는 코드정보인 것으로 판단하는 것을 특징으로 하는, 악성코드 검출을 위한 화이트리스트 구축 방법.

청구항 11

◈청구항 11은(는) 설정등록료 납부시 포기되었습니다.◈

제1항 내지 제10항 중 어느 하나의 항에 따른 악성코드 검출을 위한 화이트리스트 구축 방법을 수행하기 위한, 컴퓨터 프로그램이 기록된 컴퓨터로 판독 가능한 기록매체.

청구항 12

화이트리스트 구축을 위한 기준 애플리케이션을 구성하는 써드파티 라이브러리의 확장자를 안드로이드 가상 머신에서 실행되는 확장자로 컴파일하고, 컴파일된 써드파티 라이브러리의 클래스 및 메소드에 대하여, 연산 코드의 해시값, 메소드 프로토타입의 인수 개수, 클래스 접근 제어자, 메소드가 포함된 클래스의 깊이정보 및 메소드 접근 제어자 중 적어도 하나를 포함하는 특징정보를 추출하여 화이트리스트를 생성하는 화이트리스트 생성부; 및

검사 대상 애플리케이션의 라이브러리를 상기 화이트리스트에 포함된 특징정보와 비교하고, 상기 검사 대상 애플리케이션을 구성하는 복수의 코드정보 중 상기 특징정보와 일치하는 코드정보를 제외한 나머지 코드정보들에 대한 악성코드 분석을 수행하는 악성코드 검출부를 포함하는, 악성 코드 검출 장치.

청구항 13

제12항에 있어서,

상기 화이트리스트는, 라이브러리 정보 테이블, 패키지 정보 테이블, 클래스 정보 테이블 및 메소드 정보 테이블이 계층 구조를 형성하는 것을 특징으로 하고,

상기 화이트리스트 생성부는,

상기 깊이정보 및 상기 메소드 접근 제어자를 상기 클래스 정보 테이블로 설정하고, 상기 연산 코드의 해시값 및 메소드 프로토타입의 인수 개수를 상기 메소드 정보 테이블로 설정하고, 상기 기준 애플리케이션의 이름, 업로드 날짜, 써드파티 라이브러리 버전, 패키지명 및 클래스명에 대한 추가정보를 이용하여 상기 라이브러리 정보 테이블 및 상기 패키지 정보 테이블을 설정하는, 악성 코드 검출 장치.

발명의 설명

기 술 분 야

[0001] 본 발명은 악성코드 검출을 위한 화이트리스트 구축 방법, 이를 수행하기 위한 기록매체 및 장치에 관한 것으로서, 더욱 상세하게는 신뢰성 있는 애플리케이션 코드들을 이용하여 악성코드 검출을 위한 화이트리스트를 구축하는 악성코드 검출을 위한 화이트리스트 구축 방법, 이를 수행하기 위한 기록매체 및 장치에 관한 것이다.

배 경 기 술

[0003] 안드로이드 플랫폼을 사용하는 스마트폰 사용자의 수가 증가하면서, 안드로이드 플랫폼을 대상으로 하는 악성코드 또한 급속하게 증가하고 있다. 이에 따라, 악성 코드를 탐지하기 위한 기술 개발이 활발히 이루어지고 있다.

[0004] 종래의 악성 코드 탐지 기술은 크게 정적 분석 방법과 동적 분석 방법으로 분류된다. 정적 분석 방법은 프로그램 코드를 실행시키기 않고 죽은 코드를 분석하는 방법으로, 일반적으로 실행 파일을 분석하여 프로그램의 구조

와 동작을 분석하는 것이 특징이다. 그리고, 동적 분석 방법은 다양한 분석 환경에서 악성 코드가 있는 프로그램을 실행하여 프로세스 및 시스템의 상태 변화를 분석하는 방법이다.

[0005] 하지만, 종래의 악성 코드 탐지 기술은 분석 대상 애플리케이션의 코드 크기가 클 경우 정확한 분석이 어려울 뿐 아니라 분석 과정에 상당한 시간이 소비된다는 문제점이 있다. 더욱이, 최근 출시되는 애플리케이션의 경우 다양한 기능들을 가진 써드파티(3rd-party) 라이브러리를 대부분 사용하는 경향이 있다. 이러한 써드파티 라이브러리를 사용하는 경우 패키지 이름이나 클래스 이름을 쉽게 위조할 수 있어 악성 코드의 이름을 지정할 때 악성코드를 사용할 수 있다는 문제점이 있다. 이에 따라, 써드파티 라이브러리로 구성된 애플리케이션의 악성 코드를 효율적으로 검출할 수 있는 새로운 기술이 요구되고 있는 실정이다.

선행기술문헌

특허문헌

[0007] (특허문헌 0001) 한국공개특허 제10-2015-0044490호

(특허문헌 0002) 한국등록특허 제10-1246623호

발명의 내용

해결하려는 과제

[0008] 본 발명의 일측면은 안드로이드 애플리케이션 분석 시 미리 구축된 화이트리스트와의 비교를 통해 분석하지 않아도 코드들을 사전에 제거함으로써, 분석 시간을 단축시키면서도 분석 효율은 향상시킬 수 있는 악성코드 검출을 위한 화이트리스트 구축 방법, 이를 수행하기 위한 기록매체 및 장치를 제공한다.

[0009] 본 발명의 기술적 과제는 이상에서 언급한 기술적 과제로 제한되지 않으며, 언급되지 않은 또 다른 기술적 과제들은 아래의 기재로부터 당업자에게 명확하게 이해될 수 있을 것이다.

과제의 해결 수단

[0011] 본 발명의 일 실시예에 따른 악성코드 검출을 위한 화이트리스트 구축 방법은, 기준 애플리케이션을 구성하는 써드파티 라이브러리의 확장자를 안드로이드 가상 머신에서 실행되는 확장자로 컴파일하는 단계 및 컴파일된 써드파티 라이브러리의 클래스 및 메소드에 대하여, 연산 코드의 해시값, 메소드 프로토타입의 인수 개수, 클래스 접근 제어자, 메소드가 포함된 클래스의 깊이정보 중 적어도 하나를 포함하는 특징정보를 추출하여 화이트리스트를 생성하는 단계를 포함한다.

[0012] 상기 안드로이드 가상 머신에서 실행되는 파일 형식은, 안드로이드 달빅 가상 머신에서 동작하는 파일 형식인 DEX인 것을 특징으로 할 수 있다.

[0013] 상기 특징정보를 추출하는 것은, 상기 컴파일된 상기 써드파티 라이브러리에 대한 바이트코드에서 피연산자를 제거하여 상기 연산 코드를 추출하고, 상기 연산 코드를 미리 정해진 해시함수를 이용하여 해시값으로 변환하는 것을 특징으로 할 수 있다.

[0014] 상기 특징정보를 추출하는 것은, 상기 컴파일된 상기 써드파티 라이브러리의 메소드의 이름, 반환값의 유형 및 인수를 정의하는 메소드 프로토타입을 분석하여 상기 메소드 프로토타입의 인수 개수를 산출하는 것을 특징으로 할 수 있다.

[0015] 상기 특징정보를 추출하는 것은, 상기 기준 애플리케이션의 생성 시 설정되는 상기 컴파일된 상기 써드파티 라이브러리의 클래스 접근 레벨 및 클래스 멤버의 특성을 정의하는 상기 클래스 접근 제어자를 검색하는 것을 특징으로 할 수 있다.

[0016] 상기 특징정보를 추출하는 것은, 상기 기준 애플리케이션의 생성 시 설정되는 클래스별 계층정보를 기초로 상기 컴파일된 상기 써드파티 라이브러리의 클래스 계층에 대한 상기 깊이정보를 추출하는 것을 특징으로 할 수 있다.

[0017] 상기 화이트리스트를 생성하는 것은, 상기 깊이정보 및 상기 메소드 접근 제어자를 클래스 정보 테이블로 설정

하고, 상기 연산 코드의 해시값 및 메소드 프로토타입의 인수 개수를 메소드 정보 테이블로 설정하는 것을 특징으로 할 수 있다.

[0018] 상기 화이트리스트를 생성하는 것은, 악성코드 분석을 위한 검사 대상 애플리케이션을 클래스 정보 테이블 및 상기 메소드 정보 테이블과 비교하는 과정에서 어느 하나의 코드정보가 일치하는 것으로 탐지되면, 상기 코드정보의 라이브러리 버전 및 라이브러리 출시일을 포함하는 추가정보를 제공하는 라이브러리 정보 테이블 및 패키지 정보 테이블을 더 설정하는 것을 특징으로 할 수 있다.

[0019] 검사 대상 애플리케이션의 라이브러리를 상기 화이트리스트에 포함된 특징정보와 비교하는 단계 및 상기 검사 대상 애플리케이션을 구성하는 복수의 코드정보 중 상기 특징정보와 일치하는 코드정보를 제외한 나머지 코드정보들에 대한 악성코드 분석을 수행하는 단계를 더 포함할 수 있다.

[0020] 상기 특징정보와 비교하는 것은, 상기 검사 대상 애플리케이션의 라이브러리를 구성하는 어느 하나의 코드정보의 깊이정보 및 접근 제어자를 상기 화이트리스트에 저장된 복수의 상기 깊이정보 및 상기 클래스 접근 제어자와 비교하여 일치하는지를 확인하고, 일치하는 클래스가 존재하면 상기 어느 하나의 코드정보의 연산코드의 해시값, 메소드 프로토타입의 인수 개수를 상기 화이트리스트에 저장된 복수의 상기 연산 코드의 해시값 및 상기 메소드 프로토타입의 인수 개수와 비교하여 일치하는지를 확인하여 일치하는 메소드가 발견되는 경우, 상기 어느 하나의 코드정보를 상기 특징정보와 일치하는 코드정보인 것으로 판단하는 것을 특징으로 할 수 있다.

[0021] 또한, 본 발명에 따른 악성코드 검출을 위한 화이트리스트 구축 방법을 수행하기 위한, 컴퓨터 프로그램이 기록된 컴퓨터로 판독 가능한 기록매체일 수 있다.

[0022] 또한, 본 발명의 일 실시예에 따른 악성코드 검출 장치는, 기준 애플리케이션을 구성하는 써드파티 라이브러리의 확장자를 안드로이드 가상 머신에서 실행되는 확장자로 컴파일하고, 컴파일된 써드파티 라이브러리의 클래스 및 메소드에 대하여, 연산 코드의 해시값, 메소드 프로토타입의 인수 개수, 클래스 접근 제어자, 메소드가 포함된 클래스의 깊이정보 중 적어도 하나를 포함하는 특징정보를 추출하여 화이트리스트를 생성하는 화이트리스트 생성부 및 검사 대상 애플리케이션의 라이브러리를 상기 화이트리스트에 포함된 특징정보와 비교하고, 상기 검사 대상 애플리케이션을 구성하는 복수의 코드정보 중 상기 특징정보와 일치하는 코드정보를 제외한 나머지 코드정보들에 대한 악성코드 분석을 수행하는 악성코드 검출부를 포함한다.

[0023] 상기 화이트리스트는, 라이브러리 정보 테이블, 패키지 정보 테이블, 클래스 정보 테이블 및 메소드 정보 테이블이 계층 구조를 형성하는 것을 특징으로 하고, 상기 화이트리스트 생성부는, 상기 깊이정보 및 상기 메소드 접근 제어자를 상기 클래스 정보 테이블로 설정하고, 상기 연산 코드의 해시값 및 메소드 프로토타입의 인수 개수를 상기 메소드 정보 테이블로 설정하고, 상기 기준 애플리케이션의 이름, 업로드 날짜, 써드파티 라이브러리 버전, 패키지명 및 클래스명에 대한 추가정보를 이용하여 상기 라이브러리 정보 테이블 및 상기 패키지 정보 테이블을 설정할 수 있다.

발명의 효과

[0025] 상술한 본 발명의 일측면에 따르면, 기준 애플리케이션의 특징정보에 따라 미리 구축되는 화이트리스트를 이용하여, 분석 대상이 되는 안드로이드 애플리케이션의 코드 양을 크게 줄여 안드로이드 애플리케이션 분석 시 분석시간이 감소하고 분석 정확도가 향상될 수 있다.

도면의 간단한 설명

[0027] 도 1은 본 발명에 따른 악성코드 검출 장치의 개략적인 구성이 도시된 블록도이다.

도 2는 안드로이드 컴파일 프로세스의 구체적인 일 예가 도시된 도면이다.

도 3은 자바 패키지 구조의 일 예가 도시된 도면이다.

도 4는 자바 언어에서의 메소드 접근제어자와 프로토타입의 일 예가 도시된 도면이다.

도 5는 동일한 라이브러리를 적용한 애플리케이션의 바이트코드의 일 예가 도시된 도면이다.

도 6은 화이트리스트 데이터베이스의 스키마가 도시된 도면이다.

도 7은 악성코드 검출부에서 검사 대상 애플리케이션에서 제외시킬 코드를 탐지하는 알고리즘의 일 예가 도시된 도면이다.

도 8은 도 1의 악성코드 검출 장치의 성능 테스트 결과를 나타내는 그래프이다.

도 9는 본 발명의 일 실시예에 따른 악성코드 검출을 위한 화이트리스트 구축 방법의 개략적인 흐름이 도시된 도면이다.

발명을 실시하기 위한 구체적인 내용

[0028] 후술하는 본 발명에 대한 상세한 설명은, 본 발명이 실시될 수 있는 특정 실시예를 예시로서 도시하는 첨부 도면을 참조한다. 이들 실시예는 당업자가 본 발명을 실시할 수 있기에 충분하도록 상세히 설명된다. 본 발명의 다양한 실시예는 서로 다르지만 상호 배타적일 필요는 없음이 이해되어야 한다. 예를 들어, 여기에 기재되어 있는 특정 형상, 구조 및 특성은 일 실시예와 관련하여 본 발명의 정신 및 범위를 벗어나지 않으면서 다른 실시예로 구현될 수 있다. 또한, 각각의 개시된 실시예 내의 개별 구성요소의 위치 또는 배치는 본 발명의 정신 및 범위를 벗어나지 않으면서 변경될 수 있음이 이해되어야 한다. 따라서, 후술하는 상세한 설명은 한정적인 의미로서 취하려는 것이 아니며, 본 발명의 범위는, 적절하게 설명된다면, 그 청구항들이 주장하는 것과 균등한 모든 범위와 더불어 첨부된 청구항에 의해서만 한정된다. 도면에서 유사한 참조부호는 여러 측면에 걸쳐서 동일하거나 유사한 기능을 지칭한다.

[0029] 이하, 도면들을 참조하여 본 발명의 바람직한 실시예들을 보다 상세하게 설명하기로 한다.

[0030] 도 1은 본 발명의 일 실시예에 따른 악성코드 검출을 위한 화이트리스트 구축 및 화이트리스트를 이용한 악성코드 검출 장치(이하, 악성코드 검출 장치, 100)의 개략적인 구성이 도시된 도면이다.

[0031] 본 발명에 따른 악성코드 검출 장치(1000)는 네트워크에 접속하여 다른 장치들과 통신이 가능하며, 사용자의 요청을 입력받아 특정 작업을 처리하여 결과를 출력할 수 있는 컴퓨터 장치일 수 있다. 하지만, 악성코드 검출 장치(1000)는 이에 한정되는 것은 아니며 스마트폰, 태블릿 PC, 노트북, 웨어러블 장치 등과 같이 이하에서 설명되는 기능들을 수행할 수 있는 다른 전자장치들을 포함할 수 있다. 또한, 본 발명에 따른 악성코드 검출 장치(1000)는 물리적으로 구현되지 않은 응용 프로그램의 형태일 수도 있으며, 이러한 경우 응용프로그램(애플리케이션)의 형태로 다른 장치의 기록매체에 설치되어 동작될 수 있다. 예를 들어, 악성코드 검출 장치(1000)는 달빅 가상 머신(Dalvik Virtual Machine)의 형태로 컴퓨터 장치에 구현될 수 있다.

[0032] 구체적으로, 본 발명의 일 실시예에 따른 악성코드 검출 장치(1000)는 통신부(100), 입력부(200), 출력부(300), 데이터베이스부(400) 및 제어부(500)를 포함한다.

[0033] 이때, 악성코드 검출 장치(1000)가 컴퓨터 장치인 경우, 각각의 구성요소는 물리적인 모듈로 구현될 수 있다. 반면, 악성코드 검출 장치(1000)가 달빅 가상 머신의 형태로 구현되는 경우, 각각의 구성요소는 후술하는 기능들을 수행할 수 있도록 소프트웨어적으로 구현될 수 있다. 이하, 각각의 구성요소에 대하여 상세히 설명하기로 한다.

[0034] 통신부(100)는 네트워크로 연결되어 외부 장치와 악성코드 검출 장치(1000)를 유무선 통신망으로 연결시킬 수 있다. 통신부(100)는 외부 장치와 통신하여 화이트리스트 구축을 위한 기준 애플리케이션 또는 악성코드 분석을 위한 검사 대상 애플리케이션을 수신할 수 있다. 또한, 악성코드 검출 장치(1000)가 달빅 가상 머신의 형태인 경우, 통신부(100)는 달빅 가상 머신이 구현된 컴퓨터 장치로부터 기준 애플리케이션 또는 검사 대상 애플리케이션을 수신할 수 있다.

[0035] 입력부(200)는 사용자의 조작에 의해 악성코드 검출 장치(1000)가 동작되도록 사용자로부터 입력되는 입력신호를 감지할 수 있다. 입력부(200)는 키보드, 마우스, 조이스틱, 터치스크린, 마이크 등을 포함할 수 있다.

[0036] 출력부(300)는 악성코드 검출 장치(1000)에 의해 처리된 결과들을 화상 또는 음성으로 출력할 수 있다. 출력부(330)는 디스플레이 모듈, 스피커 등을 포함할 수 있다.

[0037] 데이터베이스부(400)는 악성코드 검출 장치(1000)로 입력되는 데이터와, 후술하는 제어부(500)에 의해 생성된 화이트리스트를 저장할 수 있다. 즉, 데이터베이스부(400)는 통신부(100)를 통해 수신되는 기준 애플리케이션 및 검사 대상 애플리케이션과, 제어부(500)에 의해 생성된 화이트리스트 및 이와 관련된 메타데이터들을 저장할 수 있다. 또한, 데이터베이스부(400)는 본 발명에 따른 악성코드 검출을 위한 화이트리스트 구축 방법 및 이를 이용하여 악성코드를 탐지하는 방법이 구현된 소프트웨어(응용프로그램)이 설치될 수 있다.

[0038] 제어부(500)는 악성코드 검출 장치(1000)의 전반적인 동작을 제어할 수 있다. 특히, 제어부(500)는 통신부(100)를 통해 수신된 기준 애플리케이션을 이용하여 화이트리스트를 구축하고, 구축된 화이트리스트를 이용하여 악

성코드 분석 대상으로 설정되는 검사 대상 애플리케이션의 악성코드를 검출하는 과정 및 결과를 출력부(300)를 통해 출력되도록 제어할 수 있다. 이를 위해, 제어부(500)는 화이트리스트 생성부(510) 및 악성코드 검출부(520)를 포함한다.

[0039] 화이트리스트 생성부(510)는 검사 대상 애플리케이션의 악성코드 분석 시 애플리케이션을 구성하는 코드들 중 검사 대상에서 제외시킬 코드를 결정하기 위한 화이트리스트를 생성할 수 있다. 즉, 화이트리스트는 종래에 안전한 것으로 알려진 기준 애플리케이션들의 특징들을 목록화한 데이터일 수 있다. 화이트리스트 생성부(510)는 화이트리스트를 구축하기 위한 각 단계를 수행하는 전처리부(511), 특징정보 추출부(512) 및 목록 생성부(513)를 포함한다.

[0040] 전처리부(511)는 기준 애플리케이션을 분석하기 위하여 애플리케이션의 확장자를 통일시키는 전처리 과정을 수행할 수 있다. 상술한 바와 같이, 기준 애플리케이션은 종래에 안전한 것으로 알려진 애플리케이션으로, 본 발명에 따른 화이트리스트를 구성하는 애플리케이션의 파일 확장자는 JAR(Java Archive), AAR(Android Archive), APK(Android Application Package) 및 DEX(Dalvik Executable)를 포함할 수 있다. 이들 중에서, 전처리부(511)는 화이트리스트의 효율적인 구축을 위해, JAR, AAR 및 APK 형태로 생성된 애플리케이션의 라이브러리들을 안드로이드 실행 가능한 파일 형태인 DEX파일 형태로 컴파일 할 수 있다. 이와 관련하여, 도 2를 함께 참조하여 설명하기로 한다.

[0041] 도 2는 안드로이드 컴파일 프로세스의 구체적인 일 예가 도시된 도면이다.

[0042] JAR은 텍스트, 이미지 등과 같이 여러 자바 클래스 파일(a.class, b.class), 관련 메타데이터 및 리소스들을 하나의 파일로 통합하여 배포하는 패키지 파일 형식이다. 자바 클래스 파일(a.class, b.class)은 자바 컴파일러(Javac)를 통해 자바 언어로 작성된 소스 코드(Java Source Code)를 컴파일하는 파일이다. JAR은 자바 가상 머신(Java Virtual Machine, JVM)을 대상으로 하기 때문에, 전처리부(511)는 안드로이드 소프트웨어 개발 키트(Software Development Kit, SDK)에 포함된 dx 도구를 이용하여 자바 바이트코드(bytecode)를 DEX 바이트코드로 변환할 수 있다.

[0043] AAR은 안드로이드 애플리케이션을 개발할 때 사용되는 Android Manifest, JAR 및 리소스 파일(resources)들을 통합하는 패키지 파일 형식이다. 전처리부(511)는 AAR에서 추출한 JAR을 상술한 dx 도구를 사용하여 DEX 바이트코드로 변환할 수 있다.

[0044] APK는 안드로이드 응용 프로그램 패키지 파일의 확장자로, 안드로이드의 소프트웨어와 미들웨어 배포에 사용되는 패키지 파일 형식이다. APK는 구조적으로 AAR과 동일하나, JAR이 포함된 AAR과는 다르게 자바 컴파일러와 dx로 컴파일된 DEX를 이미 포함하고 있다. 즉, 전처리부(511)는 APK 파일을 압축 해제하여 DEX 파일을 추출할 수 있다.

[0045] DEX 파일은 안드로이드 가상 머신인 달빅(Dalvik)이 인식할 수 있도록 클래스(class) 파일이 바이트 코드로 변환된 파일이며, 달빅 명령어로 구성되어 있다. DEX 파일을 달빅 가상 머신(Dalvik Virtual Machine, DVM)의 바이트 코드로 디컴파일하여 클래스 파일을 추출 할 경우, 안드로이드 애플리케이션의 smali 코드 및 java 코드를 추출할 수 있다. 여기서 달빅 가상 머신은 달빅 바이트 코드를 실행할 수 있는 주체를 의미한다.

[0046] 이와 같이, 전처리부(511)는 서로 다른 파일 확장자를 가진 기준 애플리케이션을 용이하게 비교할 수 있도록 하기 위한 전처리 과정의 일환으로 기준 애플리케이션의 확장자를 악성코드 검출 장치(1000), 즉 안드로이드 달빅 가상 머신(Android Dalvik Virtual Machine)상에서 동작되는 확장자인 DEX 파일로 변환시키는 컴파일 과정을 수행할 수 있다.

[0047] 특징정보 추출부(512)는 컴파일된 DEX 바이트코드를 분석하여 적어도 하나의 특징정보를 추출할 수 있다.

[0048] 상술한 바와 같이, 최근 출시되는 애플리케이션의 경우 다양한 기능들을 가진 써드파티(3rd-party) 라이브러리를 대부분 사용하는 경향이 있다. 이러한 써드파티 라이브러리를 사용하는 경우 패키지명이나 클래스명을 쉽게 위조할 수 있어 악성 코드의 이름을 지정할 때 악성코드를 사용할 수 있다는 문제점이 있다. 또한, 패키지명 또는 클래스명을 화이트리스트 목록으로 구축한 후 이를 비교하여 써드파티 라이브러리를 식별하는 종래 기술에 따르면 패키지명 또는 클래스명의 난독화(obfuscation) 시 판별이 매우 어렵고, 써드파티 라이브러리 이름을 이용하여 악성코드를 제작하는 경우 이를 오탐하게 된다는 문제점이 있다.

[0049] 따라서, 본 발명에 따른 특징정보 추출부(512)는 난독화 기법으로 변경하기 어려운 특징들을 추출하여 화이트리스트를 구축할 수 있다. 구체적으로, 특징정보 추출부(512)는 클래스의 깊이정보, 접근 제어자(access flag),

메소드 프로토타입의 인수 개수 및 연산 코드의 해시값을 특징정보로 추출할 수 있다.

[0050] 먼저, 특징정보 추출부(512)는 클래스의 깊이정보를 특징정보로 추출할 수 있다. 이와 관련하여, 도 3을 함께 참조하여 설명하기로 한다.

[0051] 도 3은 자바 패키지 구조의 일 예가 도시된 도면이다. 구체적으로, 도 3a는 난독화 전의 클래스의 구조가 도시된 도면이고, 도 3b는 난독화 후의 클래스명의 구조가 도시된 도면이다.

[0052] 자바는 자바 패키지의 클래스를 효율적으로 관리하기 위해 도시된 바와 같이 계층적 패키지 구조를 사용한다. 예를 들어, support 클래스는 v4 클래스 및 v7 클래스의 상위클래스이고, view.class로 명명된 클래스는 네 번째 하위클래스에 위치하고 있다. 계층적 패키지 구조는 클래스명의 충돌을 사전에 방지하고 프로그래머가 패키지별로 접근 권한을 상이하게 설정할 수 있다는 장점이 있다. 그러나, 이러한 계층적 패키지 구조를 유지해야 하는 자바의 특성 때문에, 클래스 난독화 기법 적용 시 도 3b에 도시된 바와 같이 패키지 구조는 그대로 유지한 채 클래스 파일명에 대해서만 난독화 기법을 적용 하거나, 패키지 전체의 이름에 대한 파일명 난독화를 적용하는 Proguard등의 도구를 사용하고 있다. 이때, 해당 자바 패키지에 난독화 기법이 적용되어 view.class의 이름이 aa.class로 변경되더라도, 도 3b에 도시된 바와 같이 패키지의 구조 자체는 유지될 수 있다. 즉, 특징정보 추출부(512)는 변경이 어려운 클래스 정보의 계층 구조를 나타내는 깊이정보를 특징정보로 추출할 수 있다.

[0053] 다음으로, 특징정보 추출부(512)는 접근 제어자를 특징정보로 추출할 수 있다. 이와 관련하여, 표 1을 함께 참조하여 설명하기로 한다.

[0054] 표 1은 접근 제어자의 일 예가 표시된 도표이다.

표 1

[0055]

Access Flag Name	Value
ACC_PUBLIC	0x1
ACC_PRIVATE	0x2
ACC_PROTECTED	0x4
...	...
ACC_INTERFACE	0x200
...	...

[0056] 안드로이드 애플리케이션은 공개(public), 비공개(private), 최종(final) 및 동기화(synchronized) 등과 같이 클래스의 멤버 특성 및 접근 레벨을 정의하는 접근 제어자(access flag)를 설정할 수 있다. 즉, 접근 제어자는 DEX 파일 내부에 비트 필드(bit-field)로 설정되어 있으며, 클래스와 클래스 멤버의 접근 제한 및 전체적인 속성을 나타내는 정보로 정의될 수 있다. 예를 들어, 특정 클래스의 접근 제어자가 0X201로 설정된 경우, 해당 클래스의 ACC_PUBLIC과 ACC_INTERFACE 속성은 표 1에 도시된 바와 같이 각각 0X4와 0X200의 값을 가질 수 있다. 이러한 접근 제어자의 정보는 변경되기 어렵기 때문에, 특징정보 추출부(512)는 접근 제어자를 화이트리스트를 구축하기 위한 특징정보로 추출할 수 있다. 이러한 접근 제어자는 클래스에 대한 접근 제어자와 메소드에 대한 접근 제어자를 포함할 수 있다.

[0057] 또한, 특징정보 추출부(512)는 메소드 프로토타입의 인수 개수를 특징정보로 추출할 수 있다. 이와 관련하여, 도 4를 함께 참조하여 설명하기로 한다.

[0058] 도 4는 자바 언어에서의 메소드 접근제어자와 프로토타입의 일 예가 도시된 도면이다.

[0059] 도 2를 함께 참조하면, 메소드 프로토타입(method prototype)은 DEX에서 proto_ids로 정의되며, 메소드의 이름, 반환 유형(return type) 및 인수(argument)를 정의하는 정보이다. 도 4에 도시된 실시예에서 인수는 a1, a2, a3이고, 따라서 인수의 개수는 3개이다. 도시되지 않은 다른 실시예로, getString () 메소드의 헤더의 바이트코드가 [38 00 94 05 DA 28 00 00]인 경우, 처음의 두 바이트인 [38 00]은 클래스의 인덱스를 의미하고, 다음의 두 바이트인 [94 05]는 메소드의 프로토타입 인덱스를 의미한다. 이때, 메소드 프로토타입의 인수 개수는 프로토타입 정보, 즉 프로토타입 인덱스를 기초로 산출될 수 있다. 이러한 메소드 프로토타입은 클래스 이름 난독화

기법이 적용되는 경우 변질되기 쉽지만 인수의 개수는 변경되지 않으므로, 특징정보 추출부(512)는 메소드 프로토타입의 인수 개수를 특징정보로 추출할 수 있다. 즉, 특징정보 추출부(512)는 써드파티 라이브러리가 DEX로 컴파일된 바이트코드로부터 메소드 프로토타입의 인수 개수를 산출할 수 있다.

[0060] 마지막으로, 특징정보 추출부(512)는 연산 코드의 해시값을 특징정보로 추출할 수 있다. 이와 관련하여, 도 5를 함께 참조하여 설명하기로 한다.

[0061] 도 5는 동일한 라이브러리를 적용한 애플리케이션의 바이트코드의 일 예가 도시된 도면이다.

[0062] 써드파티 라이브러리는 일반적으로 웹 저장소로 배포 시 달빅 가상 머신(DVM)에서 실행될 때까지 바이트코드가 최적화 되지 않는다. 하지만, 코드에서 참조하는 스트링 객체의 주소값과 호출하려는 메소드(함수)의 메모리 주소값 등은 매 컴파일 시마다 다르게 나타날 수 있다. 도 6은 동일한 소스코드를 가지며, 동일한 써드파티 라이브러리가 적용된 애플리케이션 A와 A'의 바이트코드가 도시되어 있다. 도 5의 (a) 와 도 5의 (b)는 동일한 라이브러리의 메소드를 호출하고 있지만 애플리케이션의 미묘한 차이로 참조하는 주소값이 바뀐 것(D1, D2)을 확인할 수 있다.

[0063] 이러한 문제점을 해결하기 위한 방법으로, 종래에는 baksmali도구를 사용하여 바이트코드들을 Smali코드로 변환하는 방법이 개시되어 있다. 종래 기술에 따른 방법을 이용하여 바이트코드를 Smali코드로 변환하게 되면, 참조하고 있는 스트링의 주소가 나타나는 것이 아닌 스트링 정보가 코드상에 나타나고, 호출하려는 함수의 주소가 나타나는 것이 아닌 호출하려는 함수정보가 코드상에 나타나게 된다. 하지만, 종래 기술에 따르면 매번 수행하는 디컴파일 연산의 오버헤드가 크고, 디컴파일 시 사용하는 레지스터 정보가 매번 변경될 수 있다는 단점이 있다.

[0064] 따라서, 특징정보 추출부(512)는 DEX 바이트코드에서 피연산자를 제거하여 연산 코드(OPCode)를 추출하고, 추출된 연산 코드만을 이용할 수 있다. 이에 따르면, 디컴파일 연산을 생략할 수 있어 종래 기술보다 연산 속도가 향상될 수 있으며, 더욱 정확한 비교가 가능하다. 특징정보 추출부(512)는 안드로이드 달빅 바이트코드의 특수 규칙에 따라 DEX 바이트코드에서 연산 코드를 추출하여 화이트리스트에 저장할 수 있다. 이와 관련하여, 표 2 및 표 3을 함께 참조하여 설명하기로 한다.

표 2

OPCode & Format	Syntax(smali language)
00 10x	nop
01 12x	move vA, vB
...	...
6e~72 35c	invoke-kind {vC, vD, vE, vF, vG}, meth@BBBB 6e: invoke-virtual ... 71: invoke-static
...	...

[0065]

표 3

[0067]

Format	ID	Syntax
ØØ|op	10x	op
B|A|op	12x	op vA, vB
	11n	op vA, #+B
...	...	...
A|G|op BBBB F|E|D|C	35c	[A=5] op {vC, vD, vE, vF, vG}, type@BBBB ... [A=1] op {vC}, kind@BBBB [A=0] op {}, kind@BBBB
...	...	...

[0069] 표 2 및 표 3은 데이터베이스부(400)에 미리 저장된 안드로이드 달빅 바이트코드의 특수 규칙을 도표화한 일 예이다. 구체적으로, 표 2는 연산 코드에 따른 형식 값 및 구문(syntax)을 나타내는 도표이고, 표 3은 달빅에서 실행 가능한 명령어들의 형식을 나타내는 도표이다.

[0070] 표 2의 첫번째 열인 연산 코드 및 형식(OPCode & Format)은 표 3의 두 번째 열의 ID에 해당된다. 또한, 표 2의 두 번째 열에서 smali 언어로 변환된 연산 코드의 일 예가 도표화 되어 있다.

[0071] 따라서, 특징정보 추출부(512)는 데이터베이스부(400)에 미리 저장된 안드로이드 달빅 바이트코드의 특수 규칙을 검색하여 달빅 실행 가능 명령어들의 규칙을 확인할 수 있다. 구체적으로, 특징정보 추출부(512)는 표 2의 첫 번째 열의 형식 값을 표 3의 두 번째 열의 ID 값과 매핑하여 달빅 실행 명령어 형식을 획득할 수 있다. 이때, 표 3의 각 행에 도시된 바와 같이, 데이터베이스부(400)에는 연산 코드뿐 아니라 피연산자의 정보 및 명령어의 크기를 나타내는 정보들이 함께 저장될 수 있다. 특징정보 추출부(512)는 표 3에 나타난 바와 같이 ID의 첫 번째 자리에 2바이트를 곱하여 각 명령어의 크기를 산출할 수 있다.

[0072] 이때, 특징정보 추출부(512)는 화이트리스트의 효율적인 구축을 위해 연산 코드를 미리 정해진 해시(hash)함수를 이용하여 해시값을 변환시킬 수 있다. 이는, 하나의 메소드로부터 추출될 수 있는 연산 코드의 개수에는 제한이 없으므로, 화이트리스트에 불필요하게 많은 연산 코드가 저장되는 것을 방지하기 위하여 특징정보 추출부(512)는 연산 코드를 해시값으로 변환시킬 수 있다. 여기서, 미리 정해진 해시함수는 128비트 암호화 해시 함수인 MD5(Message-Digest algorithm 5)일 수 있으나, 사용 환경 및 사용자의 설정에 따라 다양한 해시함수를 이용하여 연산 코드를 해시값으로 변환시킬 수 있음은 물론이다.

[0073] 예를 들어, 도 5의 (a)에서 5번째 줄 바이트코드인 [71 10 67 40 05 00]은 최소 끝 형식(little-endian format)이므로, 최대 끝 형식(big-endian format)인 [10 71 40 67 00 05]로 표시될 수 있다. 이때, 연산 코드는 항상 첫 번째 단어의 하위 8비트에 위치하므로, 특징정보 추출부(512)는 최대 끝 형식으로 표시된 [71]을 연산 코드로 추출할 수 있다. 연산 코드 [71]은 표 2의 첫 번째 열에서 6e와 72사이에 위치하며, 특징정보 추출부(512)는 첫 번째 열에서 35c를, 두 번째 열에서 invoke-static을 각각 획득할 수 있다. 최종적으로, 특징정보 추출부(512)는 데이터베이스부(400)에 저장된 표 3에 대한 달빅에서 실행 가능한 명령어들의 형식을 기초로, 35c를 ID로 매핑함으로써 첫 번째 열과 해당 구문으로부터 [A|G|op BBBBF|E|D|C]를 획득할 수 있다. 따라서, 특징정보 추출부(512)는 추출된 연산 코드들의 집합을 표 2 및 표 3을 기초로 MD5 암호화 해시함수를 통해 고유한 해시값으로 변환시킬 수 있다.

[0074] 이와 같은 방법으로, 특징정보 추출부(512)는 하나의 기준 애플리케이션에 대하여 클래스의 깊이정보, 접근 제어자, 메소드 프로토타입의 인수 개수 및 연산 코드의 해시값을 특징정보로 추출할 수 있다. 특징정보 추출부(512)는 수신된 모든 기준 애플리케이션 각각에 대한 특징정보를 추출할 수 있다. 이 과정에서, 특징정보 추출부(512)는 DEX로부터 애플리케이션 이름, 업로드 날짜, 써드파티 라이브러리 버전, 패키지명 및 클래스명에 대한 추가정보를 추출할 수 있다. 추가정보는 추후 탐지된 코드가 소속된 클래스와 써드파티 라이브러리를 확인하

는 과정에서 사용될 수 있다.

[0075] 목록 생성부(513)는 특징정보 추출부(512)에 의해 추출된 특징정보들을 이용하여 화이트리스트를 생성할 수 있다. 이와 관련하여, 도 6을 함께 참조하여 설명하기로 한다.

[0076] 도 6은 화이트리스트 데이터베이스의 스키마가 도시된 도면이다.

[0077] 도시된 바와 같이, 목록 생성부(513)는 라이브러리 정보 테이블(LibraryInfo), 패키지 정보 테이블(PackageInfo), 클래스 정보 테이블(ClassInfo) 및 메소드 정보 테이블(MethodInfo)로 구성된 화이트리스트 목록을 구축할 수 있다. 도시된 네 개의 테이블은 자바의 패키지 구조와 동일하게 라이브러리 정보 테이블(LibraryInfo), 패키지 정보 테이블(PackageInfo), 클래스 정보 테이블(ClassInfo), 메소드 정보 테이블(MethodInfo) 순서로 계층을 형성하여 구조화 될 수 있다.

[0078] 목록 생성부(513)는 특징정보를 이용하여 클래스 정보 테이블(ClassInfo) 및 메소드 정보 테이블(MethodInfo)을 생성할 수 있다. 목록 생성부(513) 추출된 특징정보 중 기준 애플리케이션의 연산 코드의 해시값(MethodHash)들과 메소드에 대한 접근 제어자들과, 메소드 프로토타입의 인수 개수에 대한 정보들을 메소드 정보 테이블(MethodInfo)에 포함시킬 수 있다. 그리고, 목록 생성부(513)는 클래스의 깊이정보와, 클래스의 접근 제어자들에 대한 정보들을 클래스 정보 테이블(ClassInfo)에 포함시킬 수 있다. 메소드 정보 테이블(MethodInfo)과 클래스 정보 테이블(ClassInfo)은 검사 대상 애플리케이션에서 화이트리스트에 포함된 코드들을 검출하기 위한 목적으로 사용될 수 있다.

[0079] 목록 생성부(513)는 추가정보를 이용하여 라이브러리 정보 테이블(LibraryInfo) 및 패키지 정보 테이블(PackageInfo)을 구축할 수 있다. 목록 생성부(513)는 써드파티 라이브러리에 대한 추가정보들(이름, 업데이트 일자, 버전 등)을 이용하여 라이브러리 정보 테이블(LibraryInfo)을 구축할 수 있다. 목록 생성부(513)는 패키지에에 대한 추가정보들(패키지명, 라이브러리 ID)을 이용하여 패키지 정보 테이블(PackageInfo)에 포함시킬 수 있다. 라이브러리 정보 테이블(LibraryInfo)과 패키지 정보 테이블(PackageInfo)은 화이트리스트와의 비교를 통해 검출된 코드의 추가적인 정보를 파악하기 위한 목적으로 사용될 수 있다.

[0080] 악성코드 검출부(520)는 검사 대상 애플리케이션을 검사하여 악성코드가 존재하는지 여부를 분석할 수 있다. 이때, 악성코드 검출부(520)는 기준 애플리케이션을 기초로 구축된 화이트리스트와 검사 대상 애플리케이션을 비교하여 분석하지 않아도 되는 코드들을 미리 제거하여 분석 효율을 높일 수 있다. 즉, 악성코드 검출부(520)는 구축된 화이트리스트를 이용하여 검사 대상 애플리케이션의 어떠한 코드가 분석 대상에서 제외되어야 하는지를 판단할 수 있다. 이와 관련하여, 도 7을 함께 참조하여 설명하기로 한다.

[0081] 도 7은 악성코드 검출부(520)에서 검사 대상에서 제외시킬 코드를 탐지하는 알고리즘의 일 예가 도시된 도면이다.

[0082] 악성코드 검출부(520)는 검사 대상 애플리케이션의 탐지된 라이브러리를 출력부(300)를 통해 화상정보로 출력되도록 제어할 수 있다. 먼저, 악성코드 검출부(520)는 검사 대상 애플리케이션의 메소드에 대한 클래스 깊이정보와 클래스 접근 제어자를 화이트리스트의 클래스 정보 테이블과 비교하여 일치하는 클래스가 있는지를 확인할 수 있다. 악성코드 검출부(520)는 검사 대상 애플리케이션의 클래스 깊이정보와 클래스 접근 제어자가 클래스 정보 테이블에 저장된 값들 중 어느 하나와 일치하는 것으로 확인되면, 검사 대상 애플리케이션의 해당 클래스 ID를 저장할 수 있다. 악성코드 검출부(520)는 저장된 클래스 ID를 참조 키(Foreign Key, FK)로 활용할 수 있다.

[0083] 계속해서, 악성코드 검출부(520)는 검사 대상 애플리케이션의 메소드에서 추출된 연산 코드의 해시값, 메소드 프로토타입의 인수 개수 및 메소드 접근 제어자를 메소드 정보 테이블과 비교하여 일치하는 메소드가 있는지를 확인할 수 있다. 악성코드 검출부(520)는 화이트리스트를 구성하는 메소드 정보 테이블의 어느 하나의 메소드와 일치하는 것으로 확인되면, 해당 메소드가 포함된 코드를 악성 코드 검사 대상에서 제외시키고, 검사 대상 애플리케이션의 해당 메소드를 메소드 목록(listMethod)에 저장할 수 있다. 메소드 목록은 메소드가 감지되었는지 여부를 확인하기 위하여 배열로 선언된 변수일 수 있다. 악성코드 검출부(520)는 검사 대상 애플리케이션의 모든 메소드에 대한 비교가 완료될 때까지 상술한 단계를 반복하여 수행할 수 있다.

[0084] 이후, 악성코드 검출부(520)는 화이트리스트로 결정된 응용 프로그램 또는 라이브러리를 업로드하여 목록 생성부(513)에 의해 화이트리스트가 업데이트되도록 제어할 수 있다.

[0085] 도 8은 본 발명에 따른 악성코드 검출 장치(1000)의 성능 테스트 결과를 나타내는 그래프이다.

[0086] 성능 테스트를 위하여, 악성코드 검출 장치(1000)는 윈도우 10 및 MySql 5.6이 화이트리스트 데이터베이스 구축 환경에 사용되었다. 또한, 화이트리스트는 페이스북, 구글, 스퀘어, 트위터 및 Android Arsenal에서 배포된 2,900개의 라이브러리를 기초로 구축하였다.

[0087] 본 발명에 따른 악성코드 검출 장치(1000)에 의해 구축된 화이트리스트의 타당성 및 성능을 평가하기 위하여, 써드파티 안드로이드 마켓인 APPSAPK의 26개의 잘 알려진 안드로이드용 애플리케이션을 비교하였다. 이는 안드로이드 공식 마켓인 플레이스토어를 통하는 것보다 써드파티 애플리케이션 마켓을 통해 악성 코드에 더 많이 감염되기 때문이다.

[0088] 도시된 바와 같이, 26개의 안드로이드 애플리케이션에서 1,150,629개의 메소드 중 780,321개의 메소드가 발견되었다. 화이트리스트에 포함된 라이브러리는 평균 67%이며, 고품질의 월페이퍼를 제공하는 HD Wallpapers Plus Application의 경우에는 95.90%의 메소드가 화이트리스트에 포함된 메소드인 것으로 검색되었다. 즉, 본원 발명에 따른 악성코드 검출 장치(1000)에 의해 구축된 화이트리스트를 이용하는 경우, 써드파티 라이브러리의 약 67%를 악성 코드 검사 대상에서 제외시킴으로써 애플리케이션에서 악성 코드를 검출하는 소스 코드의 크기를 줄일 수 있다.

[0089] 도 9는 본 발명의 일 실시예에 따른 악성코드 검출을 위한 화이트리스트 구축 방법의 개략적인 흐름이 도시된 순서도이다.

[0090] 악성코드 검출 장치(1000)는 화이트리스트 구축을 위해 기준 애플리케이션을 구성하는 써드파티 라이브러리의 확장자를 안드로이드 가상 머신에서 실행되는 확장자로 컴파일하는 전처리과정을 수행할 수 있다(910). 써드파티 라이브러리는 JAR, AAR, APK 및 DEX 형식의 확장자로 구현되며, 악성코드 검출 장치(1000)는 기준 애플리케이션의 확장자를 DEX 파일 형식으로 컴파일할 수 있다.

[0091] 악성코드 검출 장치(1000)는 컴파일된 써드파티 라이브러리를 분석하여, 연산 코드의 해시값, 메소드 프로토타입의 인수 개수, 클래스 접근 제어자, 메소드가 포함된 클래스의 깊이정보를 포함하는 특징정보를 추출할 수 있다(920). 특징정보는 난독화 기법이 적용되더라도 변경되기 어려운 특징들에 대한 정보이며, 특징정보를 추출하는 구체적인 과정은 상술하였으므로 반복되는 설명은 생략하기로 한다.

[0092] 악성코드 검출 장치(1000)는 추출된 특징정보를 이용하여 화이트리스트를 구축할 수 있다(930). 화이트리스트는 라이브러리 정보 테이블, 패키지 정보 테이블, 클래스 정보 테이블 및 메소드 정보 테이블이 계층 구조를 형성하고 있으며, 클래스 정보 테이블 및 메소드 정보 테이블은 검사 대상 애플리케이션의 메소드들과 비교하기 위한 테이블이고, 라이브러리 정보 테이블 및 패키지 정보 테이블은 화이트리스트와 일치하는 것으로 탐지된 검사 대상 애플리케이션의 클래스 또는 라이브러리의 추가적인 정보를 제공하기 위한 테이블로 활용될 수 있다.

[0093] 이후, 악성코드 검출 장치(1000)는 구축된 화이트리스트를 이용하여 검사 대상 애플리케이션을 구성하는 소스코드들 중에서 화이트리스트와 일치하는 메소드를 포함하는 일부 코드를 검사 대상에서 제외시킨 후 나머지 코드들에 대하여 악성코드 검사를 실시할 수 있다.

[0094] 이와 같은, 악성코드 검출을 위한 화이트리스트 구축 방법을 제공하는 기술은 애플리케이션으로 구현되거나 다양한 컴퓨터 구성요소를 통하여 수행될 수 있는 프로그램 명령어의 형태로 구현되어 컴퓨터 판독 가능한 기록 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능한 기록 매체는 프로그램 명령어, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다.

[0095] 상기 컴퓨터 판독 가능한 기록 매체에 기록되는 프로그램 명령어는 본 발명을 위하여 특별히 설계되고 구성된 것들이거나와 컴퓨터 소프트웨어 분야의 당업자에게 공지되어 사용 가능한 것일 수도 있다.

[0096] 컴퓨터 판독 가능한 기록 매체의 예에는, 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체, CD-ROM, DVD 와 같은 광기록 매체, 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 ROM, RAM, 플래시 메모리 등과 같은 프로그램 명령어를 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다.

[0097] 프로그램 명령어의 예에는, 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드도 포함된다. 상기 하드웨어 장치는 본 발명에 따른 처리를 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

[0098] 이상에서는 실시예들을 참조하여 설명하였지만, 해당 기술 분야의 숙련된 당업자는 하기의 특허 청구범위에 기재된 본 발명의 사상 및 영역으로부터 벗어나지 않는 범위 내에서 본 발명을 다양하게 수정 및 변경시킬 수 있

음을 이해할 수 있을 것이다.

부호의 설명

[0100] 1000: 악성코드 검출 장치

100: 통신부

200: 입력부

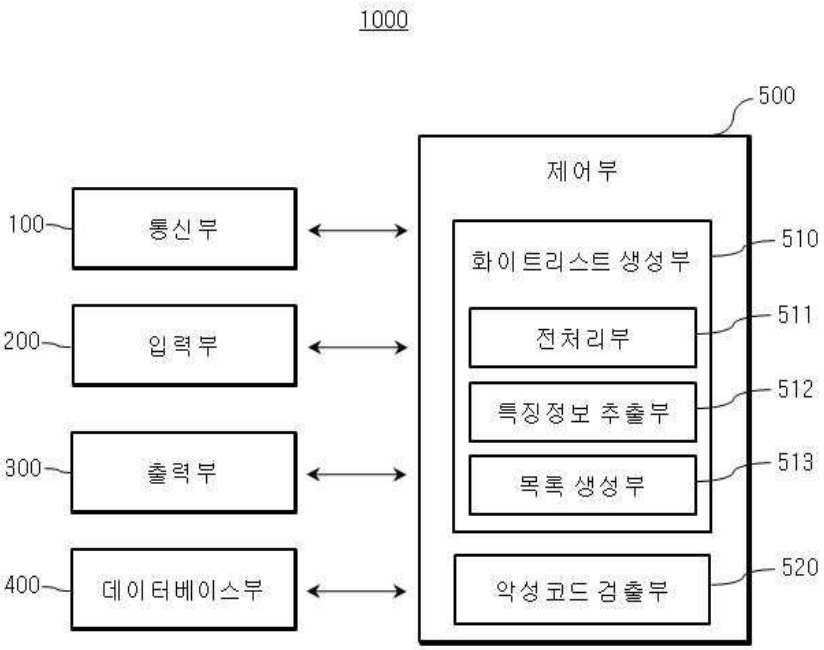
300: 출력부

400: 데이터베이스부

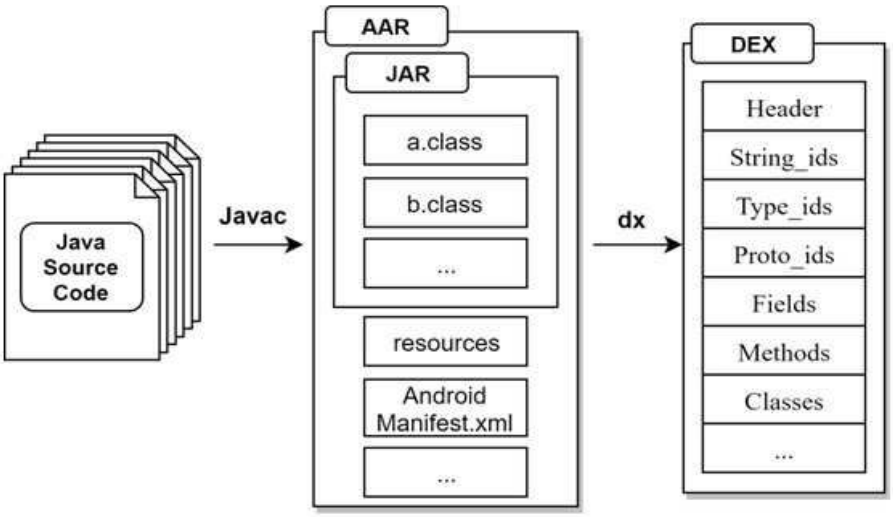
500: 제어부

도면

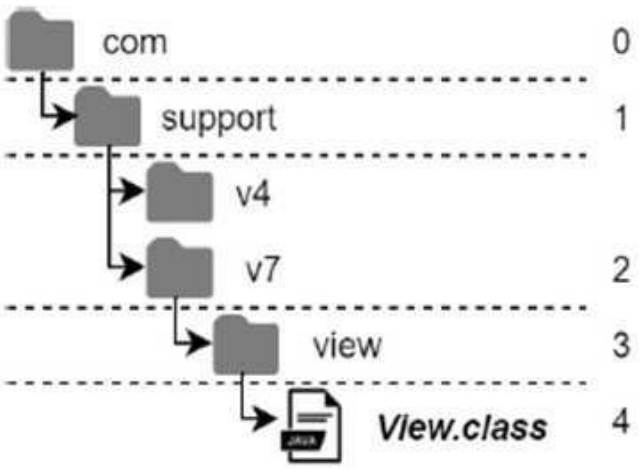
도면1



도면2

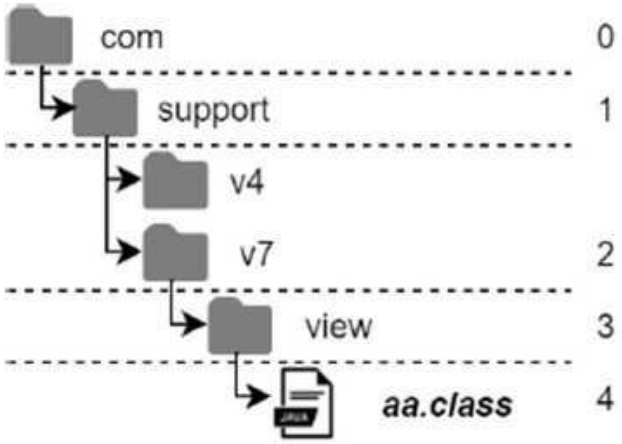


도면3a



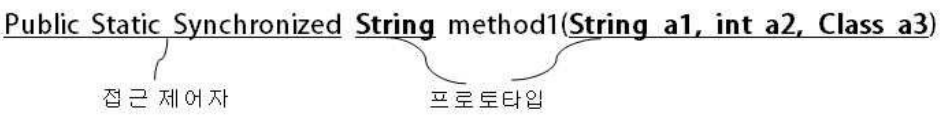
난독화 기법 적용 전의 클래스명

도면3b

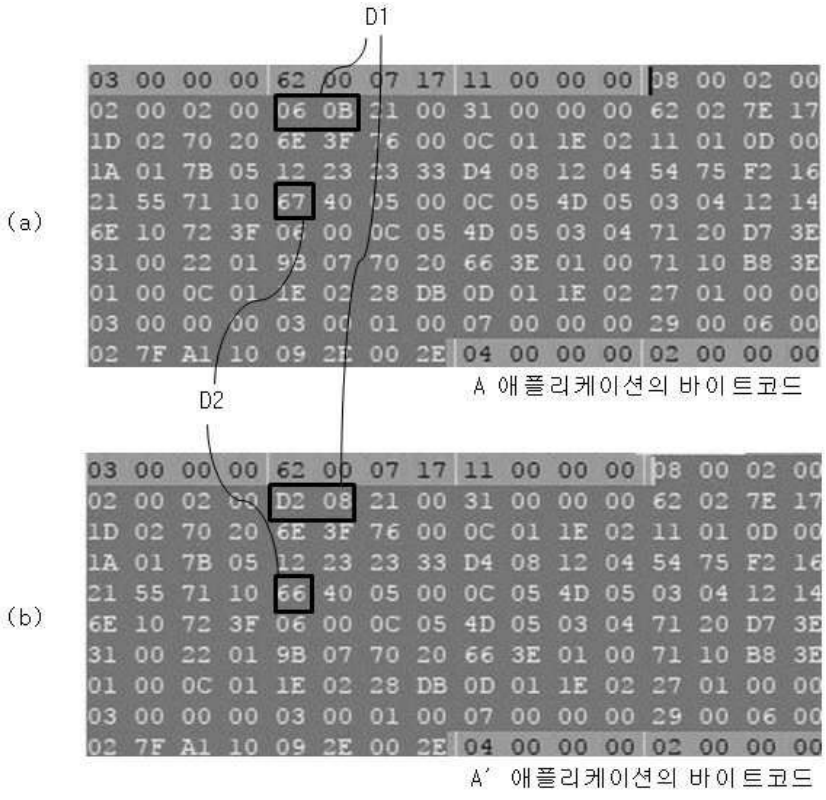


난독화 기법 적용 후의 클래스명

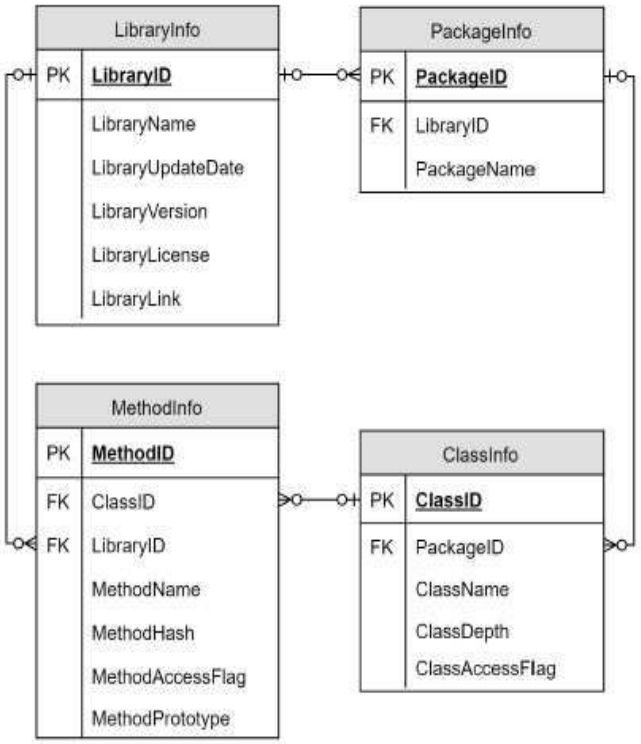
도면4



도면5



도면6

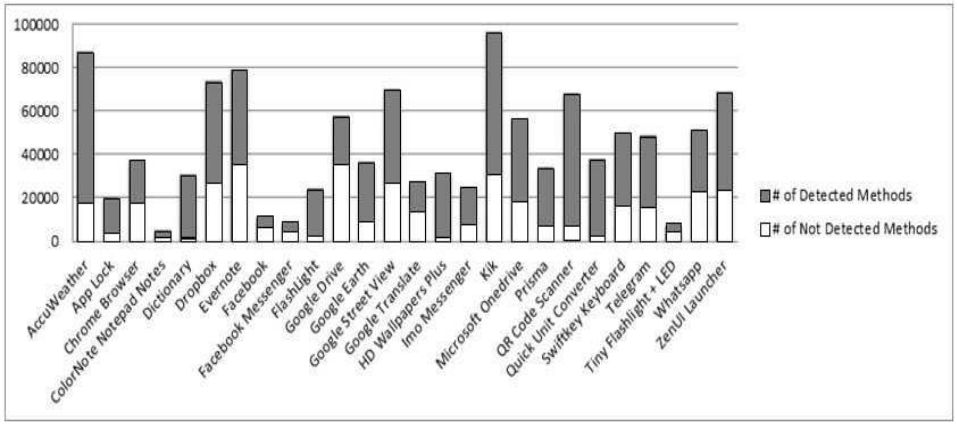


도면7

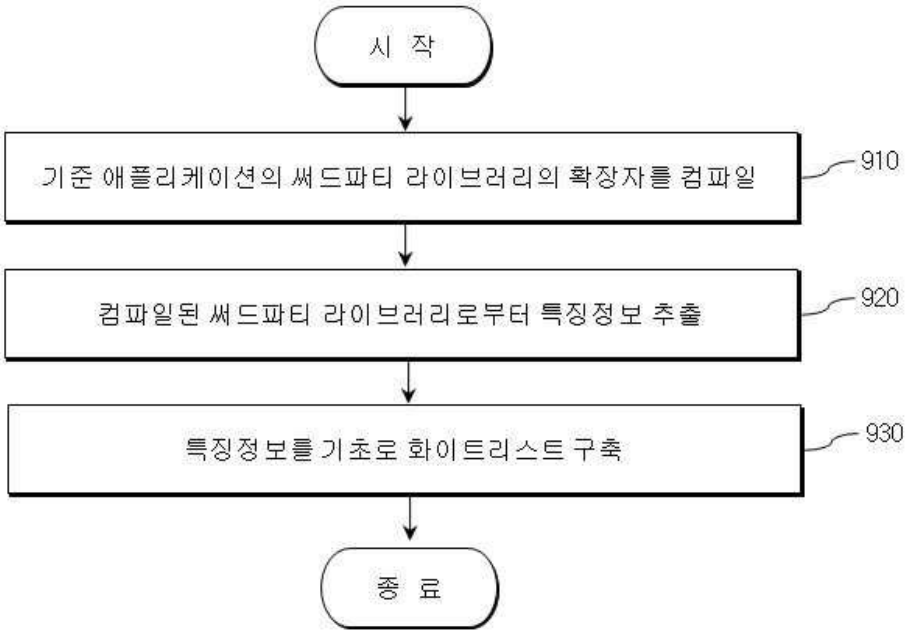
Algorithm 1 Detection algorithm

```
procedure DETECTION PROCEDURE
    while Fetch Method until at the end do
        clsDepth ← depth of Class
        clsFlag ← extract Access Flag of Class
        if (clsDepth = ClassInfo.ClassDepth and
            clsFlag = ClassInfo.ClassAccessFlag) then
                clsId ← Found Class Id
                metHash ← md5 of Method OPCodes
                metProto ← extracst Prototype of Method
                metFlag ← extract Access Flag of Method
            if (clsId = MethodInfo.ClassID and
                metHash = MethodInfo.MethodHash) and
                metProto = MethodInfo.MethodPrototype and
                metFlag = MethodInfo.MethodAccessFlag then
                    listMethod[i] ← found
    end
```

도면8


100000
80000
60000
40000
20000
0
# of Detected Methods
# of Not Detected Methods
AccuWeather
App Lock
Chrome Browser
ColorNote Notepad Notes
Dictionary
Dropbox
Evernote
Facebook
Facebook Messenger
FlashLight
Google Drive
Google Earth
Google Street View
Google Translate
HD Wallpapers Plus
Imo Messenger
Kik
Microsoft Onedrive
Prisma
QR Code Scanner
Quick Unit Converter
Swiftkey Keyboard
Telegram
Tiny Flashlight + LED
Whatsapp
ZenUI Launcher

도면9


시 작
기준 애플리케이션의 써드파티 라이브러리의 확장자를 컴파일
910
컴파일된 써드파티 라이브러리로부터 특징정보 추출
920
특징정보를 기초로 화이트리스트 구축
930
종 료