(12) 특허협력조약에 의하여 공개된 국제출원

(19) 세계지식재산권기구
국제사무국

(43) 국제공개일
2011 년 1 월 6 일 (06.01.2011)

PCT

(10) 국제공개번호
WO 2011/002146 A2

(51) 국제특허분류:
G06F 21/00 (2006.01) *G06F 11/00* (2006.01)

(21) 국제출원번호: PCT/KR2010/002375

(22) 국제출원일: 2010 년 4 월 16 일 (16.04.2010)

(25) 출원언어: 한국어

(26) 공개언어: 한국어

(30) 우선권정보:
10-2009-0058960 2009 년 6 월 30 일 (30.06.2009) KR

(71) 출원인 (US 을(를) 제외한 모든 지정국에 대하여): **(주)잉카인터넷 (INCA INTERNET CO., LTD.)** [KR/KR]; 서울특별시 구로구 구로 3 동 235-2 에이스하이엔드타워 1201 호, 152-740 Seoul (KR).

(72) 발명자; 겸
(75) 발명자/출원인 (US 에 한하여): **김윤동 (KIM, Yun Dong)** [KR/KR]; 인천광역시 부평구 갈산동 갈산주공 1 단지 108 동 303 호, 403-080 Incheon (KR). **서성원 (SEO, Seong Won)** [KR/KR]; 경기도 고양시 덕양구 성사 2 동 신원당 8 단지아파트 802 동 407 호, 412-714 Gyeonggi-do (KR). **연 성 호 (YEON, Seong Ho)** [KR/KR]; 서울특별시 구로구 구로 3 동 792-107 대영아파트 402 호, 152-875 Seoul (KR). **이지남 (LEE, Ji Nam)** [KR/KR]; 서울특별시 관악구 신림 7 동 673-43, 151-015 Seoul (KR). **정영석 (JUNG, Young Suk)** [KR/KR]; 경기도 안양시 만안구 박달 2 동 대림한숲타운 105 동 102 호, 430-702 Gyeonggi-do (KR). **한명호 (HAN, Myeong Ho)** [KR/KR]; 서울특별시 동대문구 휘경 2 동 동성빌라 30 동 302 호, 130-092 Seoul (KR). **최재영 (CHOE, Jae Young)** [KR/KR]; 대구광역시 동구 신암동 603-154, 701-010 Daegu (KR). **이 재 홍 (LEE, Jae Hong)** [KR/KR]; 서울특별시 구로구 구로 3 동 1130-26 503 호, 152-880 Seoul (KR).

(74) 대리인: **전영일 (JEON, Young Il)**; 서울특별시 강남구 삼성동 무역센터 트레이드타워 1306 호 코엑스국제특허법률사무소, 135-729 Seoul (KR).

(81) 지정국 (별도의 표시가 없는 한, 가능한 모든 종류의 국내 권리의 보호를 위하여): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

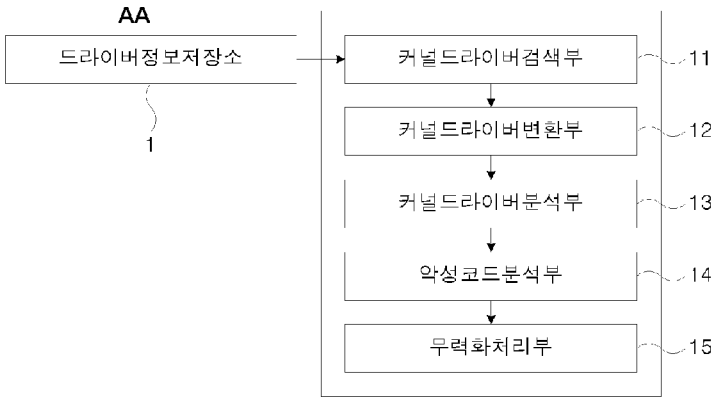
(84) 지정국 (별도의 표시가 없는 한, 가능한 모든 종류의 역내 권리의 보호를 위하여): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM,

[다음 쪽 계속]

(54) Title: SYSTEM AND METHOD FOR DETECTING MALICIOUS CODE

(54) 발명의 명칭 : 악성코드 탐지시스템 및 방법

[Fig. 1]



AA ... Driver information storage
11 ... Kernel driver searcher
12 ... Kernel driver converter
13 ... Kernel driver Analyzer
14 ... Malicious code analyzer
15 ... Incapacitation processor

(57) Abstract: The present invention relates to a system and method which accurately detect whether or not an arbitrary application program includes malicious code by applying a heuristic technique. A system for detecting malicious code according to the present invention includes a kernel driver searcher that selects a search target driver, a kernel driver converter that checks API functions used by the search target driver, a kernel driver analyzer that analyzes whether or not a suspected malicious code API is included in the API functions used by the search target driver, and a malicious code analyzer that disassembles the suspected malicious code API used by the search target driver and a parameter value to analyze whether or not the suspected malicious code API is a malicious code API.

(57) 요약서: 이 발명은 경험적(heuristic) 기법을 적용하여 임의의 응용프로그램이 악성코드를 포함하는지 여부를 정확하게 탐지하는 시스템 및 방법에 관한 것이다. 이 발명에 따른 악성코드 탐지시스템은, 검사대상드라이버를 선택하는 커널드라이버검색부와, 상기 검사대상드라이버가 사용하고 있는 API 함수들을 확인하는 커널드라이버변환부와, 상기 검사대상드라이버가 사용하고 있는 API 함수들 중 악성코드의심 API 가 포함되어 있는지를 분석하는 커널드라이버분석부와, 상기 검사대상드라이버가 사용하고 있는 악성코드의심 API 및 파라미터값을 디어셈블링하여 상기 악성코드의심 API 가 악성코드 API 인지를 분석하는 악성코드분석부를 포함한다.

WO 2011/002146 A2

ZW), 유라시아 (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), 유럽 (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

공개:

— 국제조사보고서 없이 공개하며 보고서 접수 후 이를 별도 공개함 (규칙 48.2(g))

명세서

발명의 명칭: 악성코드 탐지시스템 및 방법

기술분야

[1] 이 발명은 악성코드 탐지시스템 및 방법에 관한 것으로서, 보다 상세하게는 경험적(heuristic) 기법을 적용하여 임의의 응용프로그램이 악성코드를 포함하는지 여부를 정확하게 탐지하는 시스템 및 방법에 관한 것이다.

배경기술

[2] 최근 정보보호의 위협은 악성코드(exploit code, malicious code)에 집중되고 있으며, 이러한 악성코드는 비밀성, 무결성, 가용성 등으로 집약되는 정보보호의 목적에 반하여 전반적인 문제를 일으키고 있다. 악성코드의 실질적 정의는 다른 사람에게 심리적, 실질적으로 피해를 주기 위하여 제작된 모든 프로그램과 실행가능한 부분을 말한다. 크래커들의 악성코드 제작 기술력이 증가함에 따라 보안 위협도 증가하고 있다.

[3] 이러한 악성코드를 분석하기 위한 방법에는 지문(signature) 검사법, CRC(Cyclic Redundancy Check) 검사법 및 경험적(heuristic) 검사법이 있다.

[4] 지문 검사법은 사람을 구별할 때 지문을 보듯이 보안프로그램이 악성코드를 진단하는 방법 중의 한 가지이다. 즉, 악성코드가 가지고 있는 독특한 문자열(패턴)을 수집하여 이를 데이터베이스에 저장하고, 보안프로그램이 패턴을 매칭하는 방법을 이용하여 악성코드를 분석한다. 이러한 지문 검사법에는 순차적 문자열 검사법과 특정 문자열 검사법이 있는데, 순차적 문자열 검사법은 속도는 빠르지만 악성코드 탐지율이 떨어지는 단점이 있고, 특정 문자열 검사법은 악성코드탐지율은 높으나 속도가 느리다는 단점이 있다.

[5] 이 지문 검사법은 동일한 문자열 패턴 비교를 통해 악성코드를 탐지하기 때문에, 악성코드의 패턴 중 일부가 수정될 경우에는 해당 악성코드를 탐지할 수 없는 문제점이 있다. 이 때문에, 악성코드 탐지율을 높이려면, 보안업체는 악성코드의 패턴이 수정될 때마다 매번 다시 분석하여 수정된 악성코드의 패턴을 데이터베이스에 저장해야 하는 작업을 반복해야 하는 문제점이 있다.

[6] CRC 검사법은 시리얼 전송에서 데이터의 신뢰성을 검증하기 위한 에러검출방법의 일종으로 오진율이 낮다는 장점이 있으나, 데이터가 1 바이트라도 변형되면 악성코드를 진단할 수 없는 단점이 있다.

[7] 따라서, 최근에는 악성코드 분석방법으로서 지문 검사법의 기능을 향상시킨 경험적(heuristic) 기법이 주로 사용되는데, 이는 악성코드의 행동을 분석하거나 방식을 분석하여 자체적으로 학습하는 학습기반 분석법 중 하나이다. 즉, 악성 바이러스프로그램의 경우 독특한 조합의 API 명령을 사용하는 경우가 많은데, 경험적 기법은 이와 같이 독특한 API 명령의 조합을 학습하여 API 명령을 기반으로 악성코드 여부를 판단한다.

[8] 그러나, 통상적으로 악성코드와 보안프로그램이 유사한 API 명령어 조합을 사용하는 경우가 많으며, 이로 인해 임의의 응용프로그램의 악성코드 포함 여부를 경험적 기법으로 판단할 경우 보안프로그램을 악성코드로 오탐지하게 될 문제점이 있다.

[9]

발명의 상세한 설명

기술적 과제

[10] 상술한 문제점을 해결하기 위하여 안출된 이 발명의 목적은, 컴퓨터시스템에서 구동되는 임의의 응용프로그램이 사용하는 API 명령어 패턴을 커널레벨 영역에서 검출하고 디어셈블링하여 해당 응용프로그램의 행동패턴을 파악함으로써, 해당 응용프로그램이 악성코드를 포함하는지 여부를 정확하게 탐지하는 시스템 및 방법을 제공하기 위한 것이다.

과제 해결 수단

[11] 상기한 목적을 달성하기 위한 이 발명의 한 실시예에 따른 악성코드 탐지시스템은, 검사대상드라이버를 선택하는 커널드라이버검색부와, 상기 검사대상드라이버가 사용하고 있는 API 함수들을 확인하는 커널드라이버변환부와, 상기 검사대상드라이버가 사용하고 있는 API 함수들 중 악성코드의심API가 포함되어 있는지를 분석하는 커널드라이버분석부와, 상기 검사대상드라이버가 사용하고 있는 악성코드의심API 및 파라미터값을 디어셈블링하여 상기 악성코드의심API가 악성코드API인지를 분석하는 악성코드분석부를 포함한 것을 특징으로 한다.

[12] 또한, 이 발명의 한 실시예에 따른 악성코드 탐지방법은, 검사대상드라이버를 선택하는 커널드라이버검색단계와, 상기 검사대상드라이버가 사용하고 있는 API 함수들을 확인하는 커널드라이버변환단계와, 상기 검사대상드라이버가 사용하고 있는 API 함수들 중 악성코드의심API가 포함되어 있는지를 분석하는 커널드라이버분석단계와, 상기 검사대상드라이버가 사용하고 있는 악성코드의심API 및 파라미터값을 디어셈블링하여 상기 악성코드의심API가 악성코드API인지를 분석하는 악성코드분석단계를 포함한 것을 특징으로 한다.

[13] 또한, 이 발명의 다른 실시예에 따른 악성코드 탐지시스템은, 검사대상드라이버를 선택하는 커널드라이버검색부와, 상기 검사대상드라이버가 사용하고 있는 API 함수들을 확인하는 커널드라이버변환부와, 상기 검사대상드라이버가 사용하고 있는 API 함수들 중 악성코드의심API가 포함되어 있는지를 분석하는 커널드라이버분석부와, 상기 검사대상드라이버가 사용하고 있는 악성코드의심API 및 파라미터값을 디어셈블링하여 상기 악성코드의심API가 악성코드API인지를 분석하는 악성코드분석부와, 상기 악성코드분석부의 분석결과 상기 검사대상드라이버에 상기 악성코드API가 포함되면 상기 검사대상드라이버를 사용하는

유저프로세스를 조사하는 유저프로세스검색부와, 상기 유저프로세스가 상기 악성코드API를 호출하는지 분석하여 상기 검사대상드라이버의 상기 악성코드API가 실제 실행되는지를 탐지하는 유저프로세스분석부를 포함한 것을 특징으로 한다.

[14] 또한, 이 발명의 다른 실시예에 따른 악성코드 탐지방법은, 검사대상드라이버를 선택하는 커널드라이버검색단계와, 상기 검사대상드라이버가 사용하고 있는 API 함수들을 확인하는 커널드라이버변환단계와, 상기 검사대상드라이버가 사용하고 있는 API 함수들 중 악성코드의심API가 포함되어 있는지를 분석하는 커널드라이버분석단계와, 상기 검사대상드라이버가 사용하고 있는 악성코드의심API 및 파라미터값을 디어셈블링하여 상기 악성코드의심API가 악성코드API인지를 분석하는 악성코드분석단계와, 상기 악성코드분석단계의 분석결과 상기 검사대상드라이버에 상기 악성코드API가 포함되면 상기 검사대상드라이버를 사용하는 유저프로세스를 조사하는 유저프로세스검색단계와, 상기 유저프로세스가 상기 악성코드API를 호출하는지 분석하여 상기 검사대상드라이버의 상기 악성코드API가 실제 실행되는지를 탐지하는 유저프로세스분석단계를 포함한 것을 특징으로 한다.

발명의 효과

[15] 이상과 같이 상술한 이 발명에 따른 악성코드 탐지시스템 및 방법은 컴퓨터시스템에서 구동되는 임의의 응용프로그램의 악성코드 포함 여부를 정확하게 탐지할 수 있는 효과가 있다.

도면의 간단한 설명

[16] 도 1은 이 발명의 한 실시예에 따른 악성코드 탐지시스템의 기능 블록도,

[17] 도 2는 이 발명의 한 실시예에 따른 악성코드 탐지방법을 도시한 동작 흐름도,

[18] 도 3은 이 발명의 다른 실시예에 따른 악성코드 탐지시스템의 기능 블록도,

[19] 도 4는 이 발명의 다른 실시예에 따른 악성코드 탐지방법을 도시한 동작 흐름도이다.

[20] < 도면의 주요 부분에 대한 부호의 간단한 설명 >

[21] 1 : 드라이버정보 저장소 2 : 공유메모리

[22] 11, 31 : 커널드라이버검색부 12, 32 : 커널드라이버변환부

[23] 13, 33 : 커널드라이버분석부 14, 34 : 악성코드분석부

[24] 15 : 무력화처리부 35 : 유저프로세스검색부

[25] 36 : 유저프로세스분석부 37 : 무력화처리부

발명의 실시를 위한 최선의 형태

[26] 이하, 첨부된 도면을 참조하여 이 발명의 한 실시예에 따른 악성코드 탐지시스템 및 방법을 보다 상세하게 설명하면 다음과 같다.

[27] 컴퓨터시스템에 응용프로그램들이 실행되면, 해당 컴퓨터시스템의

드라이버정보 저장소에는 그 실행되는 응용프로그램의 커널레벨프로그램(통상적으로 커널드라이버라고 칭함) 이름들과, 각 커널드라이버가 참조하는 메모리 번지값이 저장된다. 이 드라이버정보 저장소에 저장된 드라이버 리스트의 모든 커널드라이버가 이 발명의 검사대상드라이버이다. 드라이버 리스트의 검사대상드라이버를 PE 구조체로 변환하면 해당 커널드라이버가 사용하고 있는 API 함수들과 해당 API 함수가 실제 존재하는 메모리 번지 정보를 추출할 수 있다.

[28] 이 발명에서는 컴퓨터시스템에 구동되는 응용프로그램들 중 악성코드를 포함하는 응용프로그램을 악성코드프로그램이라고 하고, 악성코드에서 사용하는 API 함수들을 악성코드의심API라고 명명한다.

[29] 도 1은 이 발명의 한 실시예에 따른 악성코드 탐지시스템의 구성 블록도이다.

[30] 이 발명의 악성코드 탐지시스템은 드라이버정보저장소(1)에 저장된 드라이버 리스트 중 임의의 검사대상드라이버를 선택하는 커널드라이버검색부(11)와, 검사대상드라이버를 PE구조체로 변환하여 검사대상드라이버가 사용하고 있는 API 함수들을 확인하는 커널드라이버변환부(12)와, 검사대상드라이버가 사용하고 있는 API함수들 중 악성코드의심API가 포함되어 있는지를 분석하는 커널드라이버분석부(13)와, 검사대상드라이버가 사용하고 있는 악성코드의심API 및 파라미터값을 디어셈블링하여 상기 악성코드의심API가 악성코드API인지를 분석하는 악성코드분석부(14)를 포함한다. 이 발명의 악성코드 탐지시스템은 악성코드분석부(14)의 분석 결과, 악성코드API를 사용하는 검사대상드라이버의 작동을 차단하는 무력화처리부(15)를 더 포함한다.

[31] 커널드라이버검색부(11)는 드라이버정보 저장소(1)에 저장된 드라이버 리스트 중 임의의 검사대상드라이버를 선택한다. 커널드라이버변환부(12)는 검사대상드라이버를 PE 구조체로 변환하는데, 그러면 검사대상드라이버가 사용하고 있는 API 함수들의 리스트 정보가 얻어진다.

[32] 커널드라이버분석부(13)는 검사대상드라이버가 사용하고 있는 API 함수들 중 악성코드의심API가 포함되어 있는지를 분석한다. 여기서, 악성코드의심API는 키로거가 사용하는 키 입출력이나 키보드 포트 접근 관련 API 함수일 수도 있고, USB 입출력함수나 파일을 읽기 또는 저장하는 API 함수일 수도 있으며, 온라인게임핵이 사용하는 키보드나 마우스의 입력과 관련된 API 함수 또는 메모리 조작과 관련된 API 함수일 수도 있다.

[33] 커널드라이버분석부(13)는 검사대상드라이버가 사용하고 있는 API 함수들 중 악성코드의심API가 포함되어 있으면 해당 악성코드의심API가 실제 존재하는 메모리의 번지 정보를 추출한다. 이 악성코드의심API는 악성코드API일 수도 있고 아닐 수도 있으며, 악성코드분석부(14)는 해당 악성코드의심API가 악성코드API인지 아닌지를 분석한다.

[34] 악성코드분석부(14)는 악성코드의심API의 메모리 번지를 주변으로 기설정된

바이트(디어셈블링대상바이트)의 코드값을 추출하여 디어셈블링한다. 이때 디어셈블링하는 코드값에 해당 악성코드의심API와 그 악성코드의심API가 사용하는 파라미터값이 포함되도록 디어셈블링대상바이트를 설정한다. 악성코드분석부(14)는 디어셈블링된 악성코드의심API와 그 파라미터값을 분석하여 해당 악성코드의심API가 보호해야 할 자원에 접근하는지를 검사함으로써, 해당 악성코드의심API가 악성코드API인지 여부를 판단한다.

[35] 예컨대, 하드웨어와 관련된 악성코드의심API는 포트의 주소를 파라미터값으로 갖는다. 키보드보안을 하고자 할 경우 키보드 입출력에 관련된 악성코드의심API(예컨대, ReadPortUChar API)가 파라미터값으로서 60h 또는 64h를 갖는다면, 그 악성코드의심API를 사용하는 검사대상드라이버는 키보드에 접근을 시도하는 악성코드로 판단한다.

[36] 무력화처리부(15)는 악성코드분석부(14)가 악성코드를 포함하는 것으로 판단한 검사대상드라이버를 무력화한다. 드라이버를 무력화하는 기술은 종래 공지된 기술을 단순 적용할 수 있다.

[37] 도 2는 이 발명의 한 실시예에 따른 악성코드 탐지시스템의 악성코드 탐지방법을 도시한 동작 흐름도이다.

[38] 먼저, 드라이버 리스트 중 검사대상드라이버를 선정한다(S21). 검사대상드라이버를 PE구조체로 변환하여 해당 검사대상드라이버가 사용하는 API함수들을 확인한다(S22). 그 API함수들 중 악성코드의심API가 포함되는지 분석하고(S23), 악성코드의심API와 파라미터값을 디어셈블링한다(S24). 그리고, 디어셈블링된 악성코드의심API와 파라미터값을 이용하여 해당 악성코드의심API가 악성코드API인지 분석한다(S25). 마지막으로, 단계 S25의 분석결과 악성코드API를 사용하는 검사대상드라이버를 무력화한다(S26).

[39] 상술한 도 1 및 도 2의 실시예에 따르면 커널드라이버가 악성코드를 포함하고 있으면 모두 무력화하여 그 실행을 차단한다. 그러나, 커널드라이버가 악성코드를 포함하고 있더라도 실제 유저레벨의 프로세스가 이를 실행시키지 않으면 해당 악성코드는 실행되지 않는데, 도 1 및 도 2의 실시예는 이러한 경우까지 모두 차단하는 문제점이 있다. 도 3은 이러한 문제를 해결하기 위한 해결방안이다.

발명의 실시를 위한 형태

[40] 도 3은 이 발명의 다른 실시예에 따른 악성코드 탐지시스템의 구성 블록도이다.

[41] 이 발명의 악성코드 탐지시스템은 드라이버정보저장소(1)에 저장된 드라이버 리스트 중 임의의 검사대상드라이버를 선택하는 커널드라이버검색부(31)와, 검사대상드라이버를 PE구조체로 변환하여 검사대상드라이버가 사용하고 있는 API 함수들을 확인하는 커널드라이버변환부(32)와, 검사대상드라이버가 사용하고 있는 API함수들 중 악성코드의심API가 포함되어 있는지를 분석하는 커널드라이버분석부(33)와, 검사대상드라이버가 사용하고 있는

악성코드의심API 및 파라미터값을 디어셈블링하여 상기 악성코드의심API가 악성코드API인지를 분석하고 악성코드API를 사용하는 검사대상드라이버의 드라이버핸들 및 악성코드API 정보를 공유메모리(2)에 저장하는 악성코드분석부(34)와, 상기 공유메모리에 저장된 드라이버핸들 정보를 이용하여 검사대상드라이버를 사용하는 유저프로세스 리스트를 조사하는 유저프로세스검색부(35)와, 유저프로세스 리스트의 각 유저프로세스가 상기 악성코드API를 호출하는지를 검사하는 유저프로세스 분석부(36)와, 악성코드API를 호출하는 유저프로세스를 무력화하는 무력화처리부(37)를 포함한다.

[42] 무력화처리부(37)는 검사대상드라이버의 드라이버핸들을 닫거나, 유저프로세스를 종료시킴으로써, 악성코드의 실행을 차단한다.

[43] 상기와 같이 구성된 악성코드 탐지시스템에서 커널드라이버검색부(31)와 커널드라이버변환부(32)와, 커널드라이버분석부(33)의 기능 및 동작은 도 1의 커널드라이버검색부(11)와 커널드라이버변환부(32)와 커널드라이버분석부(33)의 기능 및 동작과 동일한 바, 여기에서는 상세한 설명은 생략한다.

[44] 악성코드분석부(34)는 악성코드의심API의 메모리 번지를 주변으로 디어셈블링대상바이트의 코드값을 추출하여 디어셈블링하고, 디어셈블링된 악성코드의심API와 그 파라미터값을 분석하여 해당 악성코드의심API가 보호해야 할 자원에 접근하는지를 검사함으로써, 해당 악성코드의심API가 악성코드API인지 여부를 판단한다. 그리고, 악성코드분석부(34)는 검사대상드라이버가 악성코드API를 사용하는 것으로 판단되면, 해당 검사대상드라이버의 드라이버핸들 및 사용중인 악성코드API 정보를 공유메모리(2)에 저장한다.

[45] 유저프로세스검색부(35)는 공유메모리에 저장된 드라이버핸들 정보를 이용하여 검사대상드라이버를 사용하고 있는 유저프로세스들을 조사하여 유저프로세스 리스트를 만든다. 커널드라이버와 유저프로세스들은 일대일 매칭 관계가 아니므로, 검사대상드라이버를 사용하는 유저프로세스가 다수 존재하거나 없을 수도 있다.

[46] 유저프로세스분석부(36)는 유저프로세스 리스트의 유저프로세스가 검사대상드라이버에 포함된 악성코드API를 호출하는 지를 분석한다. 유저프로세스가 검사대상드라이버에 포함된 악성코드API를 호출할 경우, 악성코드가 실행되는 것이기 때문에 무력화처리부(37)는 해당 악성코드가 실행되지 못하도록 무력화 처리한다. 유저프로세스가 검사대상드라이버에 포함된 악성코드API를 호출하지 않은 경우에는 악성코드API의 악용 가능성이 없는 것이므로 검사대상드라이버와 유저프로세스가 정상적으로 동작하도록 한다.

[47] 무력화처리부(37)는 검사대상드라이버의 드라이버핸들을 닫거나,

유저프로세스를 종료시킴으로써, 해당 악성코드가 실행되지 못하도록 무력화 처리한다.

[48] 도 4는 이 발명의 다른 실시예에 따른 악성코드 탐지시스템의 악성코드 탐지방법을 도시한 동작 흐름도이다.

[49] 먼저, 드라이버 리스트 중 검사대상드라이버를 선정한다(S41). 검사대상드라이버를 PE구조체로 변환하여 해당 검사대상드라이버가 사용하는 API함수들을 확인한다(S42). 그 API함수들 중 악성코드의심API가 포함되는지 분석하고(S43), 악성코드의심API와 파라미터값을 디어셈블링한다(S44). 그리고, 디어셈블링된 악성코드의심API와 파라미터값을 이용하여 해당 악성코드의심API가 악성코드API인지 분석한다(S45). 단계 S45의 분석결과, 악성코드API를 사용하는 검사대상드라이버의 드라이버의 드라이버핸들 및 사용중인 악성코드API 정보를 공유메모리에 저장한다(S46).

[50] 드라이버핸들 정보를 이용하여 검사대상드라이버를 사용하는 유저프로세스 리스트를 조사하고(S47), 해당 유저프로세스가 악성코드API를 호출하는지를 분석한다(S48). 마지막으로, 단계 S48의 분석결과 유저프로세스가 악성코드API를 호출할 경우에는 해당 악성코드의 실행을 무력화하는데(S49), 이는 해당 드라이버핸들을 닫거나 유저프로세스를 종료시킴으로써 이룰 수 있다.

[51] 이상에서 본 발명에 대한 기술사상을 첨부도면과 함께 서술하였지만, 이는 본 발명의 가장 양호한 실시예를 예시적으로 설명한 것이지 본 발명을 한정하는 것은 아니다. 또한, 이 기술분야의 통상의 지식을 가진 자라면 누구나 본 발명의 기술사상의 범주를 이탈하지 않는 범위 내에서 다양한 변형 및 모방이 가능함은 명백한 사실이다.

[52]

[53]

청구범위

[청구항 1] 검사대상드라이버를 선택하는 커널드라이버검색부와,
상기 검사대상드라이버가 사용하고 있는 API 함수들을 확인하는 커널드라이버변환부와,
상기 검사대상드라이버가 사용하고 있는 API 함수들 중 악성코드의심API가 포함되어 있는지를 분석하는 커널드라이버분석부와,
상기 검사대상드라이버가 사용하고 있는 악성코드의심API 및 파라미터값을 디어셈블링하여 상기 악성코드의심API가 악성코드API인지를 분석하는 악성코드분석부를 포함한 것을 특징으로 하는 악성코드 탐지시스템.

[청구항 2] 제 1 항에 있어서, 상기 악성코드분석부의 분석 결과 상기 검사대상드라이버가 상기 악성코드API를 사용하면 상기 검사대상드라이버의 동작을 무력화 처리하는 무력화처리부를 더 포함한 것을 특징으로 하는 악성코드 탐지시스템.

[청구항 3] 제 2 항에 있어서, 상기 무력화처리부는 상기 검사대상드라이버의 드라이버핸들을 닫는 것을 특징으로 하는 악성코드 탐지시스템.

[청구항 4] 검사대상드라이버를 선택하는 커널드라이버검색단계와,
상기 검사대상드라이버가 사용하고 있는 API 함수들을 확인하는 커널드라이버변환단계와,
상기 검사대상드라이버가 사용하고 있는 API 함수들 중 악성코드의심API가 포함되어 있는지를 분석하는 커널드라이버분석단계와,
상기 검사대상드라이버가 사용하고 있는 악성코드의심API 및 파라미터값을 디어셈블링하여 상기 악성코드의심API가 악성코드API인지를 분석하는 악성코드분석단계를 포함한 것을 특징으로 하는 악성코드 탐지방법.

[청구항 5] 제 4 항에 있어서, 상기 악성코드분석단계의 분석 결과 상기 검사대상드라이버가 상기 악성코드API를 사용하면 상기 검사대상드라이버의 동작을 무력화 처리하는 무력화처리단계를 더 포함한 것을 특징으로 하는 악성코드 탐지방법.

[청구항 6] 제 5 항에 있어서, 상기 무력화처리단계는 상기 검사대상드라이버의 드라이버핸들을 닫는 것을 특징으로 하는 악성코드 탐지방법.

[청구항 7] 검사대상드라이버를 선택하는 커널드라이버검색부와,
상기 검사대상드라이버가 사용하고 있는 API 함수들을 확인하는 커널드라이버변환부와,

상기 검사대상드라이버가 사용하고 있는 API 함수들 중
악성코드의심API가 포함되어 있는지를 분석하는
커널드라이버분석부와,
상기 검사대상드라이버가 사용하고 있는 악성코드의심API 및
파라미터값을 디어셈블링하여 상기 악성코드의심API가
악성코드API인지를 분석하는 악성코드분석부와,
상기 악성코드분석부의 분석결과 상기 검사대상드라이버에 상기
악성코드API가 포함되면 상기 검사대상드라이버를 사용하는
유저프로세스를 조사하는 유저프로세스검색부와,
상기 유저프로세스가 상기 악성코드API를 호출하는지 분석하여
상기 검사대상드라이버의 상기 악성코드API가 실제
실행되는지를 탐지하는 유저프로세스분석부를 포함한 것을
특징으로 하는 악성코드 탐지시스템.

[청구항 8] 제 7 항에 있어서, 상기 유저프로세스분석부의 분석 결과 상기
유저프로세스가 상기 악성코드API를 호출하면 상기
검사대상드라이버의 동작을 무력화 처리하는 무력화처리부를 더
포함한 것을 특징으로 하는 악성코드 탐지시스템.

[청구항 9] 제 8 항에 있어서, 상기 무력화처리부는 상기 검사대상드라이버의
드라이버핸들을 닫는 것을 특징으로 하는 악성코드 탐지시스템.

[청구항 10] 제 8 항에 있어서, 상기 무력화처리부는 상기 유저프로세스를
종료시키는 것을 특징으로 하는 악성코드 탐지시스템.

[청구항 11] 제 8 항에 있어서, 상기 악성코드분석부는 상기
검사대상드라이버에 상기 악성코드API가 포함되면 상기
검사대상드라이버의 드라이버핸들과 악성코드API 정보를
공유메모리에 저장하고, 상기 유저프로세스검색부는 상기
검사대상드라이버의 드라이버핸들을 이용하여 상기
검사대상드라이버를 사용하는 유저프로세스를 조사하는 것을
특징으로 하는 악성코드 탐지시스템.

[청구항 12] 검사대상드라이버를 선택하는 커널드라이버검색단계와,
상기 검사대상드라이버가 사용하고 있는 API 함수들을 확인하는
커널드라이버변환단계와,
상기 검사대상드라이버가 사용하고 있는 API 함수들 중
악성코드의심API가 포함되어 있는지를 분석하는
커널드라이버분석단계와,
상기 검사대상드라이버가 사용하고 있는 악성코드의심API 및
파라미터값을 디어셈블링하여 상기 악성코드의심API가
악성코드API인지를 분석하는 악성코드분석단계와,
상기 악성코드분석단계의 분석결과 상기 검사대상드라이버에

상기 악성코드API가 포함되면 상기 검사대상드라이버를 사용하는 유저프로세스를 조사하는 유저프로세스검색단계와, 상기 유저프로세스가 상기 악성코드API를 호출하는지 분석하여 상기 검사대상드라이버의 상기 악성코드API가 실제 실행되는지를 탐지하는 유저프로세스분석단계를 포함한 것을 특징으로 하는 악성코드 탐지방법.

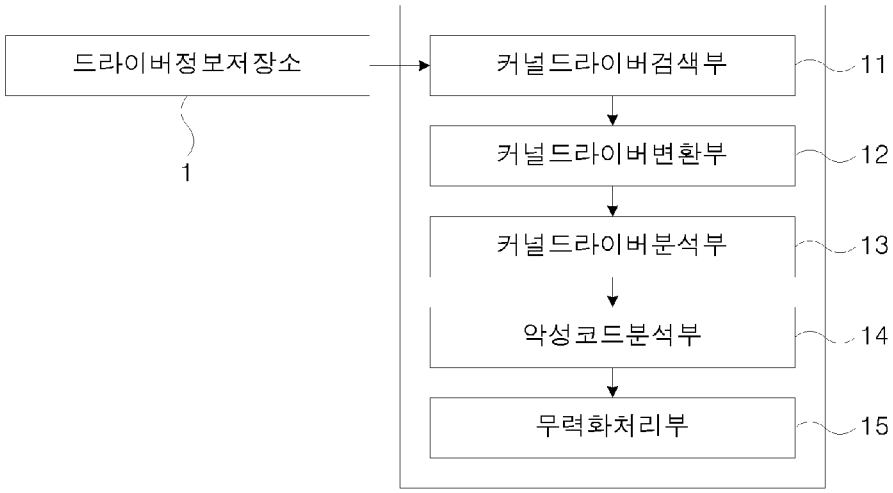
[청구항 13] 제 12 항에 있어서, 상기 유저프로세스분석단계의 분석 결과 상기 유저프로세스가 상기 악성코드API를 호출하면 상기 검사대상드라이버의 동작을 무력화 처리하는 무력화처리단계를 더 포함한 것을 특징으로 하는 악성코드 탐지방법.

[청구항 14] 제 13 항에 있어서, 상기 무력화처리단계는 상기 검사대상드라이버의 드라이버핸들을 닫는 것을 특징으로 하는 악성코드 탐지방법.

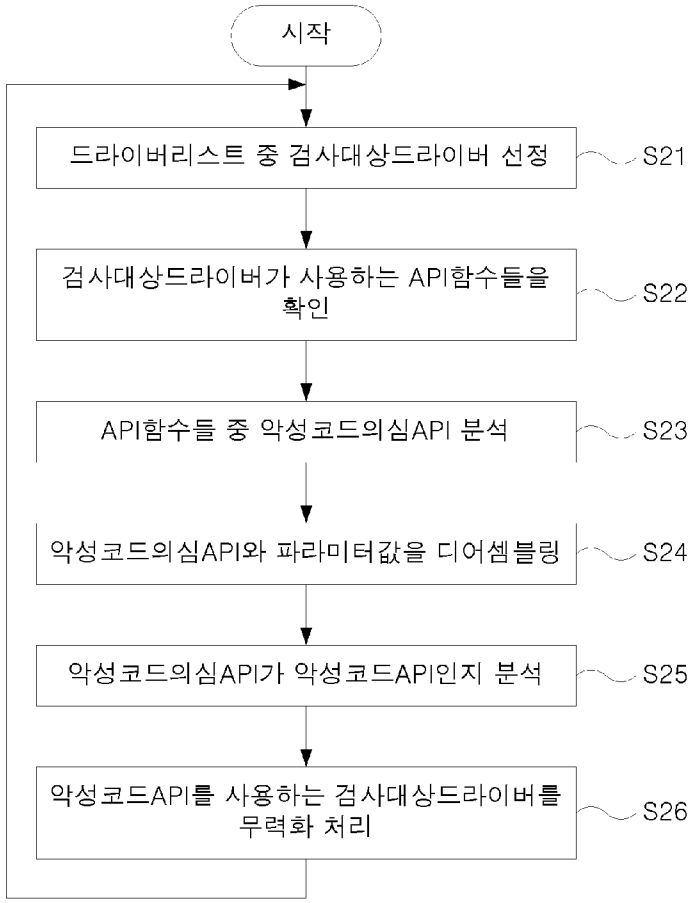
[청구항 15] 제 13 항에 있어서, 상기 무력화처리단계는 상기 유저프로세스를 종료시키는 것을 특징으로 하는 악성코드 탐지방법.

[청구항 16] 제 12 항에 있어서, 상기 악성코드분석단계는 상기 검사대상드라이버에 상기 악성코드API가 포함되면 상기 검사대상드라이버의 드라이버핸들과 악성코드API 정보를 공유메모리에 저장하고, 상기 유저프로세스검색단계는 상기 검사대상드라이버의 드라이버핸들을 이용하여 상기 검사대상드라이버를 사용하는 유저프로세스를 조사하는 것을 특징으로 하는 악성코드 탐지방법.

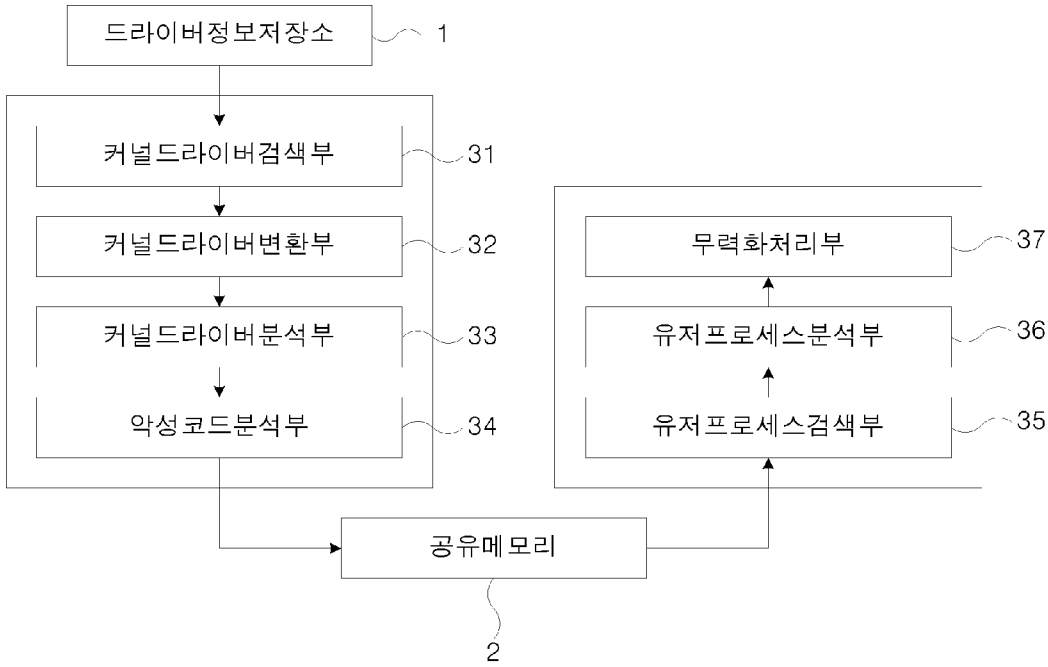
[Fig. 1]


드라이버정보저장소
1
커널드라이버검색부
11
커널드라이버변환부
12
커널드라이버분석부
13
악성코드분석부
14
무력화처리부
15

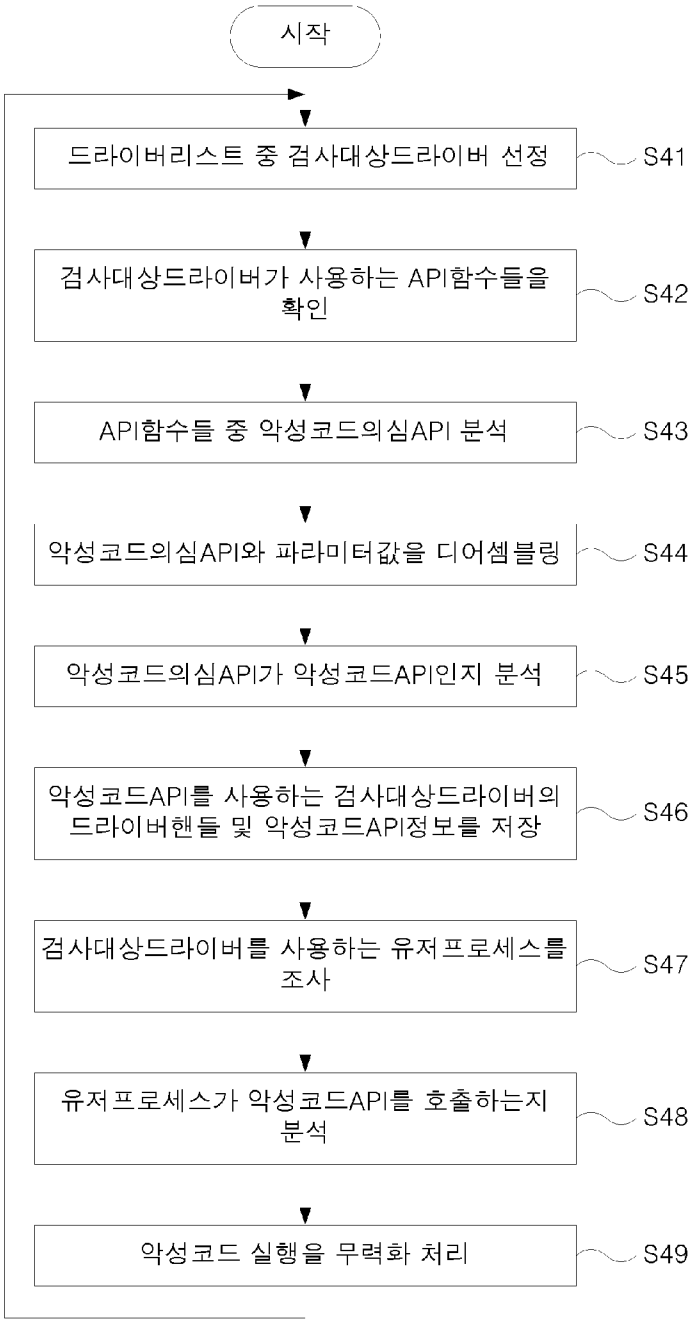
[Fig. 2]


시작
드라이버리스트 중 검사대상드라이버 선정
S21
검사대상드라이버가 사용하는 API함수들을 확인
S22
API함수들 중 악성코드의심API 분석
S23
악성코드의심API와 파라미터값을 디어셈블링
S24
악성코드의심API가 악성코드API인지 분석
S25
악성코드API를 사용하는 검사대상드라이버를 무력화 처리
S26

[Fig. 3]


드라이버정보저장소
1
커널드라이버검색부
31
커널드라이버변환부
32
커널드라이버분석부
33
악성코드분석부
34
무력화처리부
37
유저프로세스분석부
36
유저프로세스검색부
35
공유메모리
2

[Fig. 4]


시작
드라이버리스트 중 검사대상드라이버 선정
S41
검사대상드라이버가 사용하는 API함수들을 확인
S42
API함수들 중 악성코드의심API 분석
S43
악성코드의심API와 파라미터값을 디어셈블링
S44
악성코드의심API가 악성코드API인지 분석
S45
악성코드API를 사용하는 검사대상드라이버의 드라이버핸들 및 악성코드API정보를 저장
S46
검사대상드라이버를 사용하는 유저프로세스를 조사
S47
유저프로세스가 악성코드API를 호출하는지 분석
S48
악성코드 실행을 무력화 처리
S49