(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2020년02월19일
(11) 등록번호 10-2079377
(24) 등록일자 2020년02월13일

(51) 국제특허분류(Int. Cl.)
G06F 21/56 (2013.01)
(52) CPC특허분류
G06F 21/56 (2013.01)
(21) 출원번호 10-2018-0064227
(22) 출원일자 2018년06월04일
심사청구일자 2018년06월04일
(65) 공개번호 10-2019-0138093
(43) 공개일자 2019년12월12일
(56) 선행기술조사문헌
KR1020140072749 A
KR101860546 B1
KR101754720 B1
CN106372507 A

(73) 특허권자
고려대학교 산학협력단

(72) 발명자
이희조

한지연

(74) 대리인
특허법인엠에이피에스

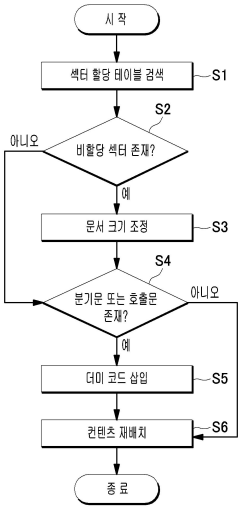
전체 청구항 수 : 총 14 항 심사관 : 윤혜숙

(54) 발명의 명칭 **문서 파일의 악성 코드 무력화 서비스 제공 방법 및 장치**

(57) 요 약

본 발명의 일 측면에 따른 문서 파일의 악성 코드 무력화 서비스 제공 방법 및 그 장치는, 문서 파일에 포함된 악성 코드의 공격을 방지하는 악성 코드 무력화 서비스 제공 장치에 의해 수행되는 문서 파일의 악성 코드 무력화 서비스 제공 방법에 있어서, 상기 문서 파일에 분기문 또는 호출문이 있는지를 탐색하고, 상기 분기문 또는 호출문이 있는 경우, 상기 분기문 또는 호출문이 탐색된 지점에 더미 코드를 삽입하는 단계를 포함하되, 상기 더미 코드는, 상기 분기문 또는 호출문이 존재하는 섹터와 상기 분기문 또는 호출문의 분기 명령 또는 호출 명령에 의한 이동 주소에 해당하는 섹터가 분리되도록 하는 것이다.

대 표 도 - 도3



이 발명을 지원한 국가연구개발사업
과제고유번호 20150005650031001
부처명 과학기술정보통신부
연구관리전문기관 정보통신기술진흥센터
연구사업명 정보보호핵심원천기술개발사업
연구과제명 IoT 소프트웨어 보안 취약점 자동 분석 기술 개발
기 여 율 1/1
주관기관 고려대학교 산학협력단
연구기간 2017.06.01 ~ 2018.05.31

명 세 서

청구범위

청구항 1

문서 파일에 포함된 악성 코드의 공격을 방지하는 악성 코드 무력화 서비스 제공 장치에 의해 수행되는 문서 파일의 악성 코드 무력화 서비스 제공 방법에 있어서,

헤더와 기설정된 수의 섹터로 이루어진 문서 파일에 분기문 또는 호출문이 있는지를 탐색하고, 상기 분기문 또는 호출문이 있는 경우, 상기 분기문 또는 호출문이 탐색된 제1 섹터에 더미 코드를 삽입하는 단계를 포함하되,

상기 더미 코드는,

상기 분기문 또는 호출문이 존재하는 제1 섹터에 삽입되어 상기 분기문 또는 호출문이 존재하는 제1 주소를 상기 제1 섹터에 포함시키고, 상기 분기문의 분기 명령 또는 호출문의 호출 명령에 의한 이동 주소인 제2 주소를 상기 제1 섹터의 다음 섹터인 제2 섹터에 포함시켜 상기 제1 주소와 제2 주소가 분리되도록 하는 것인, 문서 파일의 악성 코드 무력화 서비스 제공 방법.

청구항 2

제 1 항에 있어서,

상기 문서 파일의 섹터 상태를 표시하는 섹터 할당 테이블(Sector Allocation Table)을 참조하여 상기 문서 파일의 섹터 할당 테이블에 할당되지 않는 비할당 섹터(Free Sector)가 있는 지를 탐색하고, 상기 비할당 섹터가 있는 경우, 해당 비할당 섹터를 제거하여 상기 섹터 할당 테이블에 기초한 상기 문서 파일의 실제 파일 크기가 되도록 문서 크기를 조정하는 단계를 더 포함하는 것인, 문서 파일의 악성 코드 무력화 서비스 제공 방법.

청구항 3

제 2 항에 있어서,

상기 문서 크기를 조정하는 단계는,

상기 섹터 할당 테이블을 이용하여 할당된 문서의 섹터의 수(C(u))를 카운트하는 단계; 및

상기 섹터 할당 테이블에 포함되지 않은 헤더 섹터를 추가하여 파일 크기(S'(f))를 계산하는 단계; 및

상기 계산된 파일 크기(S'(f))가 상기 문서 파일의 실제 파일 크기(S(f))보다 클 경우, 상기 비할당 섹터가 있다고 판단하는 단계를 포함하는 것인, 문서 파일의 악성 코드 무력화 서비스 제공 방법.

청구항 4

제 1 항에 있어서,

상기 더미 코드를 삽입하는 단계는,

상기 악성 코드에서 사용되는 함수의 시작(Prologue) 부분이 검색되면, 상기 시작 부분에 해당하는 주소에서 상기 시작 부분을 해체하여 사전 처리를 수행하는 단계를 더 포함하는 것인, 문서 파일의 악성 코드 무력화 서비스 제공 방법.

청구항 5

제1항에 있어서,

상기 더미 코드를 삽입하는 단계는,

상기 분기문 또는 호출문을 검색하고, 상기 검색된 분기문 또는 호출문의 수에 따라 상기 문서 파일에 삽입할 더미 코드의 개수를 계산하는 단계;

상기 분기문 또는 호출문의 피연산자 값에 따라 각 더미 코드의 크기와 삽입 위치를 계산하는 단계; 및

참조값이 설정되지 않은 레코드를 사용하여 더미 코드를 생성하는 단계를 포함하는 것인, 문서 파일의 악성 코드 무력화 서비스 제공 방법.

청구항 6

제 5 항에 있어서,

상기 분기문 또는 호출문을 검색하고, 상기 검색된 분기문 또는 호출문의 수에 따라 상기 문서 파일에 삽입할 더미 코드의 개수를 계산하는 단계 는,

문서 파일의 섹터 상태를 표시하는 섹터 할당 테이블(Sector Allocation Table)의 할당 섹터와 헤더 섹터를 제외한 섹터 영역에 대해 삽입할 더미 코드의 개수를 계산하는 것인, 문서 파일의 악성 코드 무력화 서비스 제공 방법.

청구항 7

제 5 항에 있어서,

상기 분기문 또는 호출문의 피연산자 값에 따라 각 더미 코드의 크기와 삽입 위치를 계산하는 단계는,

상기 분기문 또는 호출문의 이동 주소가 계산 가능한 경우, 상기 분기문 또는 호출문이 존재하는 주소인 제1 주소, 상기 분기문 또는 호출문의 이동 주소인 제2 주소, 섹터 크기(D)를 이용하여 섹터와 섹터의 경계주소인 제3 주소를 계산하고, 상기 제3 주소와 제2 주소의 차이값을 더미 코드의 크기로 설정하며, 상기 설정된 크기를 가지는 더미 코드를 상기 분기문 또는 호출문이 검색된 섹터에 삽입하는 것인, 문서 파일의 악성 코드 무력화 서비스 제공 방법.

청구항 8

제 5 항에 있어서,

상기 분기문 또는 호출문의 피연산자 값에 따라 각 더미 코드의 크기와 삽입 위치를 계산하는 단계는,

상기 분기문 또는 호출문의 이동 주소가 계산 불가능한 경우, 상기 분기문 또는 호출문이 존재하는 주소인 제1 주소, 다음 섹터의 시작 주소인 제3 주소를 이용하여 제3 주소와 제1 주소의 차이값을 더미 코드의 크기로 설정하며, 상기 설정된 크기를 가지는 더미 코드를 상기 분기문 또는 호출문이 검색된 섹터에 삽입하는 것인, 문서 파일의 악성 코드 무력화 서비스 제공 방법.

청구항 9

제 5 항에 있어서,

상기 참조값이 설정되지 않은 레코드를 사용하여 더미 코드를 생성하는 단계는,

상기 문서 파일이 정상 문서일 경우에 문서의 의미를 유지하고, 상기 문서 파일이 악성 파일일 경우에 프로그램 종료가 수행되도록 종료 인터럽트 코드를 이용해 더미 코드의 값을 설정하는 것인, 문서 파일의 악성 코드 무력화 서비스 제공 방법.

청구항 10

제 1 항에 있어서,

상기 분기문 또는 호출문이 탐색된 제1 섹터에 더미 코드를 삽입하는 단계 이후에,

상기 문서 파일의 섹터들을 기 설정된 섹터 배치 조건을 만족하는 무작위화 알고리즘(Raddom Algorithm)을 사용하여 상기 제1 섹터와 제2 섹터를 포함한 모든 섹터들의 초기 위치 변경을 위해 섹터들을 이동시키는 콘텐츠의 재배치를 수행하여 상기 문서 파일 내 악성 코드가 분리되도록 하는 단계를 더 포함하는 것인, 문서 파일의 악성 코드 무력화 서비스 제공 방법.

청구항 11

제 10 항에 있어서,

상기 기 설정된 섹터 배치 조건은

상기 섹터를 초기 위치로 재배치 하지 않는 제1 조건 및 상기 무작위화 알고리즘을 사용하기 이전에 인접한 섹터들은 다시 인접되지 않도록 배치하는 제2 조건을 포함하는 것인, 문서 파일의 악성 코드 무력화 서비스 제공 방법.

청구항 12

문서 파일에 포함된 악성 코드의 공격을 방지하는 문서 파일의 악성 코드 무력화 서비스 제공 장치에 있어서,

상기 문서 파일의 악성 코드 무력화 서비스 제공을 위한 프로그램이 기록된 메모리; 및 상기 프로그램을 실행하기 위한 프로세서를 포함하며,

상기 프로세서는, 상기 프로그램의 실행에 의해, 헤더와 기설정된 수의 섹터로 이루어진 문서 파일에 분기문 또는 호출문이 있는지를 탐색하고, 상기 분기문 또는 호출문이 있는 경우, 상기 분기문 또는 호출문이 탐색된 제1 섹터에 더미 코드를 삽입하되,

상기 더미 코드는, 상기 분기문 또는 호출문이 존재하는 제1 섹터에 삽입되어 상기 분기문 또는 호출문이 존재하는 제1 주소를 상기 제1 섹터에 포함시키고, 상기 분기문의 분기 명령 또는 호출문의 호출 명령에 의한 이동 주소인 제2 주소를 상기 제1 섹터의 다음 섹터인 제2 섹터에 포함시켜 상기 제1 주소와 제2 주소가 분리되도록 하는 것인, 문서 파일의 악성 코드 무력화 서비스 제공 장치.

청구항 13

제12항에 있어서,

상기 프로세서는, 상기 문서 파일의 섹터 상태를 표시하는 섹터 할당 테이블(Sector Allocation Table)을 참조하여 상기 문서 파일의 섹터 할당 테이블에 할당되지 않는 비할당 섹터(Free Sector)가 있는 지를 탐색하고, 상기 비할당 섹터가 있는 경우, 해당 비할당 섹터를 제거하여 상기 섹터 할당 테이블에 기초한 상기 문서 파일의 실제 파일 크기가 되도록 문서 크기를 조정하는 것인 수행하는 것인, 문서 파일의 악성 코드 무력화 서비스 제공 장치.

청구항 14

제12항에 있어서,

상기 프로세서는, 상기 분기문 또는 호출문이 탐색된 제1 섹터에 더미 코드를 삽입한 이후에, 상기 문서 파일의 섹터들을 기 설정된 섹터 배치 조건을 만족하는 무작위화 알고리즘(Random Algorithm)을 사용하여 상기 제1 섹터와 제2 섹터를 포함한 모든 섹터들의 초기 위치 변경을 위해 섹터들을 이동시키는 콘텐츠의 재배치를 수행하여 상기 문서 파일 내에 악성 코드가 분리되도록 하는 것인, 문서 파일의 악성 코드 무력화 서비스 제공 장치.

발명의 설명

기 술 분 야

[0001] 본 발명은 문서 크기 조정, 더미 코드 삽입 또는 콘텐츠의 재배치를 통해 문서 내의 악성 코드를 효과적으로 방어 할 수 있는 문서 파일의 악성 코드 무력화 서비스 제공 방법 및 장치에 관한 것이다.

배 경 기 술

[0002] 익스플로잇(exploit)이란 컴퓨터의 소프트웨어나 하드웨어 및 컴퓨터 관련 전자 제품의 버그, 보안 취약점 등 설계상 결함을 이용해 공격자의 의도된 동작을 수행하도록 만들어진 절차나 일련의 명령, 스크립트, 프로그램 또는 특정한 데이터 조각을 말하며, 이러한 것들을 사용한 공격 행위를 이르기도 한다.

[0003] 익스플로잇의 목적은 주로 대상 시스템의 제어 권한 획득하는 것이며, 보안취약점을 발생시키는 트리거 코드, 보안기술을 회피하는 코드, 실제 공격을 수행하는 쉘코드의 세 부분으로 구성된다.

[0004] 이 중 트리거 코드와 쉘코드는 모든 익스플로잇에 공통적으로 존재하며, 공격 대상이 사용하고 있는 보안 기술에 따라 그것을 회피하기 위한 보안기술 회피 코드가 별도로 사용된다. 보안기술 회피코드로는 최근 보안기술 회피용 코드인 힙스프레이(heap spraying), 데이터 실행 방지 기술 회피용 코드인 ROP(Return Oriented

Programming) 또는 그 둘을 조합한 것이 주로 사용된다.

[0005] 쉘코드는 작은 크기의 코드로 소프트웨어 취약점을 이용하기 위해 사용된다. 페이로드(payload)라고 불리기도 하며, 전통적으로 쉘코드가 수행이 되었을 때 명령 쉘을 시작시켜 그로부터 공격자가 공격 대상 컴퓨터를 제어하는 방식으로 이용된다. 쉘코드는 특정 시스템에서 실행 가능한 형태의 기계어 코드로 구성되어 있고, 공격 수행시 메모리에 직접 적재되어 실행된다. 쉘코드는 일반적으로 어셈블리어로 작성되고 기계어로 변경된다.

[0006] 악성 문서를 이용한 공격은 끊임없이 증가하고 있다. 시만텍TM의 인터넷 보안 위협 보고서(Internet Security Threat Report) 2017에 따르면 전자 메일은 맬웨어 전파의 주요 벡터이며 악성 코드는 종종 첨부 파일로 사용된다.

[0007] 이러한 악성 문서에는 주로 쉘 코드 또는 악성 실행 파일을 다운로드하는 악성 매크로를 포함하고 있다. 악성 코드인 셸 코드 또는 매크로를 정적 분석, 동적 분석 및 기계 학습으로 탐지하기위한 많은 시도가 수행되었지만 여전히 악성 코드를 탐지하는데 한계가 있으며, 악성 문서는 기존의 악성코드 탐지 기술을 우회하여 끊임없이 진화하고 있다.

[0008] 기존의 익스플로잇 탐지 방법은 크게 정적 탐지 방법 및 동적 탐지 방법으로 구분된다. 동적 탐지 방법은 익스플로잇을 가상화 공간(또는 가상 머신)에서 실제로 실행시켜 악성공격이 수행되는지 여부를 모니터링을 함으로써 익스플로잇을 탐지하는 방법이다. 이러한 익스플로잇 탐지 방법은 익스플로잇을 실행하기 위한 제반 조건(예를 들어, 익스플로잇이 동작될 수 있는 시스템 아키텍쳐, 운영체제, 소프트웨어 버전, 또는 언어 등)이 충족되었을 때에만 실행이 되기 때문에, 그러한 조건을 충족하지 못해 익스플로잇이 탐지 시스템에서 정상적으로 실행되지 않는 경우 탐지에 실패하는 문제점이 있다.

[0009] 정적 탐지 방법의 경우, 악성 문서의 셸 코드 또는 악의적인 자바 스크립트를 탐지하는데 널리 사용되는데, 최근의 익스플로잇들은 대부분 폴리몰픽 기법을 사용하거나 난독화되어 있는 경우가 많아 일반적인 정적 탐지 방법으로는 익스플로잇을 탐지하기 어렵다는 문제점이 있다.

[0010] 한편, 동적 탐지 방법의 경우, 정적 탐지 방법의 문제를 해결하기 위해 API 함수를 후킹하거나 운영 체제에서 호출한 이벤트 알림 루틴을 모니터링하여 악성 동작을 분석한다. 그러나 시한 폭탄 셸 코드 또는 가상 인식 셸 코드와 같은 많은 악성 쉘 코드가 동적 탐지 기술을 우회할 수 있다.

[0011] 최근 몇 년 동안 기계 학습 기반 탐지 기술이 정적 및 동적 분석의 단점을 극복하기 위해 등장하였지만, 공격자들이 기계학습 기반의 탐지 시스템을 우회할 수 있는 새로운 유형의 쉘 코드를 개발하고 있다.

[0012] 상기한 정적 분석, 동적 분석, 기계 학습 기반의 탐지 기술을 이용하여 악성 문서를 탐지하는 것 외에도 악성 코드의 회피 기술을 방지하기 위해 여러 가지 접근 방법이 사용되고 있다. 응용 프로그램 수준에서 악의적인 공격을 위해 가장 많이 사용되는 Microsoft OfficeTM 문서는 신뢰할 수있는 게시자를 확인하는 등의 악성 코드 실행을 방지하는 여러 가지 보안 옵션을 지원하며, 기본값은 매크로 실행을 비활성화한다. 이러한 옵션은 문서 파일의 악용 가능성을 줄일 수 있지만 악성 코드의 공격 자체를 막을 수는 없다.

선행기술문헌

[0013] 대한민국 등록특허 제 10-1731022호(발명의 명칭: 익스플로잇 탐지 방법 및 장치)

발명의 내용

해결하려는 과제

[0014] 본 발명의 일 실시예는 전술한 종래 기술의 문제점을 해결하기 위한 것으로서, 문서의 일반 요소로 수행되고, 악성 코드의 실행을 방어할 수 있도록 더미 코드를 삽입하여 악의적인 쉘 코드를 분리할 수 있는 문서 파일의 악성 코드 무력화 서비스 제공 방법 및 장치를 제공하고자 한다.

[0015] 또한, 본 발명의 일 실시예는 더미 코드를 삽입하기 전에 악성 코드가 삽입될 가능성이 높은 비할당 섹터를 제거하여 문서 크기를 조정하거나, 쉘 코드의 분리를 위해 콘텐츠 재배치 과정을 수행하여 악성 코드로부터의 실행을 사전에 차단할 수 있는 문서 파일의 악성 코드 무력화 서비스 제공 방법 및 장치를 제공하고자 한다.

[0016] 다만, 본 실시예가 이루고자 하는 기술적 과제는 상기된 바와 같은 기술적 과제로 한정되지 않으며, 또 다른 기술적 과제들이 존재할 수 있다.

과제의 해결 수단

[0017] 상술한 기술적 과제를 달성하기 위한 기술적 수단으로서, 본 발명의 일 측면에 따른 문서 파일의 악성 코드 무력화 서비스 제공 방법은, 문서 파일에 포함된 악성 코드의 공격을 방지하는 악성 코드 무력화 서비스 제공 장치에 의해 수행되는 문서 파일의 악성 코드 무력화 서비스 제공 방법에 있어서, 상기 문서 파일에 분기문 또는 호출문이 있는지를 탐색하고, 상기 분기문 또는 호출문이 있는 경우, 상기 분기문 또는 호출문이 탐색된 지점에 더미 코드를 삽입하는 단계를 포함하되, 상기 더미 코드는, 상기 분기문 또는 호출문이 존재하는 섹터와 상기 분기문 또는 호출문의 분기 명령 또는 호출 명령에 의한 이동 주소에 해당하는 섹터가 분리되도록 하는 것이다.

[0018] 본 발명의 다른 측면에 따른 문서의 악성 코드 무력화 서비스 제공 장치는, 문서 파일에 포함된 악성 코드의 공격을 방지하는 문서 파일의 악성 코드 무력화 서비스 제공 장치에 있어서, 상기 문서 파일의 악성 코드 무력화 서비스 제공을 위한 프로그램이 기록된 메모리; 및 상기 프로그램을 실행하기 위한 프로세서를 포함하며, 상기 프로세서는, 상기 프로그램의 실행에 의해, 상기 문서 파일에 분기문 또는 호출문이 있는지를 탐색하고, 상기 분기문 또는 호출문이 있는 경우, 상기 분기문 또는 호출문이 탐색된 지점에 더미 코드를 삽입하되, 상기 더미 코드는, 상기 분기문 또는 호출문이 존재하는 섹터와 상기 분기문 또는 호출문의 분기 명령 또는 호출 명령에 의한 이동 주소에 해당하는 섹터가 분리되도록 하는 것이다.

발명의 효과

[0019] 전술한 본 발명의 과제 해결 수단 중 어느 하나에 의하면, 셸 코드의 크기에 상관없이 더미 코드 삽입을 통해 셸 코드를 사용하는 익스플로잇을 효과적으로 방어 할 수 있고, 그로 인해 새로운 유형의 셸 코드, 난독화된 셸 코드 등의 모든 형태의 셸 코드를 방어할 수 있는 효과가 있다.

[0020] 또한, 본 발명은 더미 코드를 삽입하기 전에 악성 코드가 삽입될 가능성이 높은 비할당 섹터를 제거하여 문서 크기를 조정하거나, 셸 코드의 분리를 위해 콘텐츠 재배치 과정을 수행하여 악성 코드로부터의 실행을 사전에 차단할 수 있어 악성 코드로부터의 방어율을 더욱 향상시킬 수 있다.

도면의 간단한 설명

[0021] 도 1은 일반적인 복합 파일 섹터 체인을 설명하는 예시도이다.

도 2는 본 발명의 일 실시예에 따른 문서의 악성 코드 무력화 서비스 제공 장치의 구성을 나타낸 도면이다.

도 3은 본 발명의 일 실시예에 따른 문서 파일의 악성 코드 무력화 서비스 제공 방법을 설명하는 순서도이다.

도 4는 도 3의 문서 크기 조정 단계의 처리 과정을 설명하는 도면이다.

도 5는 도 3의 더미 코드 삽입 단계에서 분기문 또는 호출문의 이동 주소가 계산 가능한 경우의 더미 코드 삽입 과정을 설명하는 도면이다.

도 6은 도 3의 더미 코드 삽입 단계에서 분기문 또는 호출문의 이동 주소가 계산 불가능한 경우의 더미 코드 삽입 과정을 설명하는 도면이다

도 7은 더미 코드로 사용되는 종료의 인터럽트 코드에 대한 일례를 설명하는 도면이다.

도 8은 도 3의 콘텐츠 재배치 단계에서 섹터 배치 조건 중 제1 조건을 만족하는 재정렬 과정을 설명하는 도면이다.

도 9는 도 3의 콘텐츠 재배치 단계에서 섹터 배치 조건 중 제2 조건을 만족하는 재정렬 과정을 설명하는 도면이다.

도 10은 본 발명의 일 실시예에 따른 문서의 악성 코드 무력화 서비스 제공 방법이 악성 문서에 가짜 셸 코드와 진짜 셸 코드를 삽입한 경우에 셸 코드의 실행을 차단하는 방어 동작을 설명하는 도면이다.

발명을 실시하기 위한 구체적인 내용

[0022] 아래에서는 첨부한 도면을 참조하여 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자가 용이하게 실시할 수 있도록 본 발명의 실시예를 상세히 설명한다. 그러나 본 발명은 여러 가지 상이한 형태로 구현될 수 있으며 여기에서 설명하는 실시예에 한정되지 않는다. 본 발명을 명확하게 설명하기 위해 도면에서 설명과 관계없는 부분은 생략하였으며, 명세서 전체를 통하여 유사한 부분에 대해서는 유사한 도면 부호를 붙였다. 또한, 도면

을 참고하여 설명하면서, 같은 명칭으로 나타낸 구성일지라도 도면에 따라 도면 번호가 달라질 수 있고, 도면 번호는 설명의 편의를 위해 기재된 것에 불과하고 해당 도면 번호에 의해 각 구성의 개념, 특징, 기능 또는 효과가 제한 해석되는 것은 아니다.

[0023] 명세서 전체에서, 어떤 부분이 다른 부분과 "연결"되어 있다고 할 때, 이는 "직접적으로 연결"되어 있는 경우뿐 아니라, 그 중간에 다른 소자를 사이에 두고 "전기적으로 연결"되어 있는 경우도 포함한다. 또한, 어떤 부분이 어떤 구성요소를 "포함"한다고 할 때, 이는 특별히 반대되는 기재가 없는 한 다른 구성요소를 제외하는 것이 아니라 다른 구성요소를 더 포함할 수 있는 것을 의미하며, 하나 또는 그 이상의 다른 특징이나 숫자, 단계, 동작, 구성요소, 부분품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.

[0024] 본 명세서에 있어서 '부(部)' 또는 '모듈'이란, 하드웨어 또는 소프트웨어에 의해 실현되는 유닛(unit), 양방을 이용하여 실현되는 유닛을 포함하며, 하나의 유닛이 둘 이상의 하드웨어를 이용하여 실현되어도 되고, 둘 이상의 유닛이 하나의 하드웨어에 의해 실현되어도 된다.

[0025] 도 1은 일반적인 복합 파일 섹터 체인을 설명하는 예시도이다.

[0026] 도 1을 참고하면, 복합 파일 이진 형식은 마이크로소프트에 의해 개발된 문서 구조로서, 개체 연결 및 임베디드 복합 파일(OLECF)로 알려져 있고, 디렉토리, 스트림과 같은 스토리지를 포함하는 FAT 와 유사하는 구조를 갖는다. 스트림은 일정한 길이의 섹터 단위들로 분할되고, 섹터들은 단일 섹터 체인에 의해 연결되고, 연결된 리스트 구조를 갖는다. 섹터 체인은 섹터#0에서 시작하고, 0번 섹터의 다음 섹터는 섹터#2이며, 2번 섹터의 다음 섹터는 섹터 체인의 끝에 있는 섹터#4이다. 또 다른 섹터는 섹터#1에서 시작해서 섹터#3으로 끝난다. 이러한 섹터의 정보는 섹터 할당 테이블에 설명되어 있다.

[0027] 문서 파일은 하나의 헤더와 다수의 섹터로 이루어지고, 헤더에는 문서 파일의 전반적인 설정(섹터 사이즈, SAT 위치 등)의 값이 저장되어 있고, 각 섹터에는 데이터가 저장되어 있다.

[0028] 하나의 섹터 사이즈를 512바이트라고 할 경우, 저장해야 할 데이터의 크기가 해당 바이트의 크기보다 작을 경우에는 빈 공간이 있을지라도 하나의 섹터에 데이터가 모두 저장되어 데이터를 참고할 때 해당 섹터 번호를 참조하면 된다.

[0029] 그러나, 데이터의 크기가 512바이트보다 큰 경우에 여러 개의 섹터에 분할하여 데이터를 저장해야 한다. 이렇게 분할된 데이터의 참조를 위해서 SAT(Sector Allocation Table)를 사용해서 파일의 전체 상태를 표시하고 있다. 데이터가 이어질 경우에, 섹터 다음으로 이어지는 섹터 번호를 SAT에 표기하도록 하고, 데이터의 마지막을 알려주기 위해 -2를 표기하도록 한다. 이러한 방식으로 SAT를 통해 연결된 섹터의 연계성을 확인할 수 있고, SAT 를 통해 특정 섹터의 상태를 확인할 수 있다. 예를 들어, 섹터가 사용되지 않고 있음을 나타내기 위해서는 -1 이 사용되고, SAT가 존재하는 섹터를 표기하기 위해서는 -3이 사용될 수 있다.

[0030] 도 2는 본 발명의 일 실시예에 따른 문서의 악성 코드 무력화 서비스 제공 장치의 구성을 나타낸 도면이다.

[0031] 도 2를 참조하면, 문서의 악성 코드 무력화 서비스 제공 장치(100)는 통신 모듈(110), 메모리(120), 프로세서(130) 및 데이터베이스 (140)를 포함한다.

[0032] 통신 모듈(110)은 통신망과 연동하여 사용자 단말에 통신 인터페이스를 제공하는데, 사용자 단말로부터 전송되는 데이터 요청을 수신하고, 이에 대한 응답으로서 사용자 단말에 데이터를 송신하는 역할을 수행할 수 있다. 여기서, 통신 모듈(110)은 다른 네트워크 장치와 유무선 연결을 통해 제어 신호 또는 데이터 신호와 같은 신호를 송수신하기 위해 필요한 하드웨어 및 소프트웨어를 포함하는 장치일 수 있다.

[0033] 메모리(120)는 섹터 할당 테이블을 저장하고, 문서 파일의 악성 코드 무력화 서비스 제공 방법을 수행하기 위한 프로그램이 기록된다. 또한, 메모리(120)는 프로세서(130)가 처리하는 데이터를 일시적 또는 영구적으로 저장하는 기능을 수행한다. 여기서, 메모리(120)는 휘발성 저장 매체(volatile storage media) 또는 비휘발성 저장 매체(non-volatile storage media)를 포함할 수 있으나, 본 발명의 범위가 이에 한정되는 것은 아니다.

[0034] 프로세서(130)는 일종의 문서 파일의 악성 코드 무력화 서비스 제공 방법을 제공하는 전체 과정을 제어한다. 프로세서(130)가 수행하는 각 단계에 대해서는 도 3을 참조하여 후술하기로 한다.

[0035] 여기서, 프로세서(130)는 프로세서(processor)와 같이 데이터를 처리할 수 있는 모든 종류의 장치를 포함할 수 있다. 여기서, '프로세서(processor)'는, 예를 들어 프로그램 내에 포함된 코드 또는 명령으로 표현된 기능을

수행하기 위해 물리적으로 구조화된 회로를 갖는, 하드웨어에 내장된 데이터 처리 장치를 의미할 수 있다. 이와 같이 하드웨어에 내장된 데이터 처리 장치의 일 예로써, 마이크로프로세서(microprocessor), 중앙처리장치(central processing unit: CPU), 프로세서 코어(processor core), 멀티프로세서(multiprocessor), ASIC(application-specific integrated circuit), FPGA(field programmable gate array) 등의 처리 장치를 망라할 수 있으나, 본 발명의 범위가 이에 한정되는 것은 아니다.

[0036] 도 3은 본 발명의 일 실시예에 따른 문서 파일의 악성 코드 무력화 서비스 제공 방법을 설명하는 순서도이고, 도 4는 도 3의 문서 크기 조정 단계의 처리 과정을 설명하는 도면이다.

[0037] 도 3을 참고하면, 문서 파일의 악성 코드 무력화 서비스 제공 방법은, 크게 문서 크기 조정 단계, 더미 코드 삽입 단계 및 콘텐츠 재배치 단계로 이루어진다. 문서 크기 변경 단계는 문서 파일의 끝에 실행 파일이나 익스플로잇 코드가 포함된 비할당 섹터를 검사하고 제거하기 위한 것이고, 더미 코드 삽입 단계는 섹터와 섹터의 경계에 소형 크기의 쉘 코드를 위치시키기 위한 것이며, 콘텐츠 재배치 단계는 콘텐츠의 순서를 변경하여 쉘코드를 완전히 분할하기 위한 것이다.

[0038] 512바이트 이하의 쉘 코드는 무조건적으로 분리될 수 없는 것을 의미하지 않는다. 512 바이트보다 더 작은 소형 크기의 쉘 코드는 쉘 코드가 섹터들 사이에 위치할 때 랜덤 콘텐츠로 분리될 수 있다. 문서에서 쉘코드의 사이즈를 α라고 하고, α의 범위는 1부터 512까지라고 한다. 이 쉘 코드가 섹터안에 포함될 확률 γ은 섹터 내에 쉘 코드를 제외한 나머지 영역에 의해 점유된 영역의 비율이다.

[0039] 범용 쉘코드가 한 섹터 내에 포함될 수 있는 크기 범위인 α, α에 따른 확률 γ를 하기한 수학식1에서 찾을 수 있다.

[0040] [수학식 1]

[0041] $$\gamma = 1 - \frac{\alpha - 1}{512} \quad for \quad 0 < \alpha \leq 512.$$

[0042] 악성(Exploit) 코드 또는 실행 파일은 새로 삽입되기 때문에 섹터 할당 테이블에 할당되지 않은 비할당 섹터로 구성될 수 있다. 따라서 프로세서(130)는 문서 파일의 섹터 할당 테이블에 할당되지 않는 비할당 섹터(Free Sector)가 있는 지를 탐색하여 비할당 섹터가 있는 경우에, 파일 끝에 위치한 비할당 섹터를 제거하기 위해 문서 크기 조정을 수행한다(S1, S2, S3).

[0043] 프로세서(130)는 비할당 섹터를 제외하고 섹터 할당 테이블에 할당된 섹터 수를 카운트한다. 이때, f를 파일이라고 하고, S는 실제 파일 크기를 계산하는 함수라고 하며, S'을 파일 크기 계산 함수라고 할 경우, 파일의 크기는 다음 수학식 2를 만족해야 한다.

[0044] [수학식 2]

[0045] $$S(f) \leq S'(f).$$

[0046] u를 문서 단위 섹터, u_f를 자유 섹터, t 를 섹터 할당 테이블이라고 할 경우, 문서의 섹터 수를 나타내는 카운트 함수 C를 정의한다. 하나의 섹터 할당 테이블에 할당할 수 있는 섹터의 최대 개수가 128이라는 사실을 감안할 때, C(u)는 아래 수학식 3을 통해 도출할 수 있다.

[0047] [수학식 3]

[0048] $$C(u) = (C(t) * 128) - C(u_f).$$

[0049] 각 섹터의 크기는 512 바이트이고, 헤더 섹터는 섹터 할당 테이블에 포함되어 있지 않다. 따라서, 헤더 섹터를 추가함으로써 S'(f)를 다음 수학식 4와 같이 정의 할 수 있다.

[0050] [수학식 4]

[0051] $$S'(f) = (C(u) + 1) * S(u).$$

[0052] 수학식 4에 의하면, S(f)가 S'(f)보다 큰 경우에 비할당 섹터가 있음을 의미하므로, 프로세서(130)는 도 4에 도시된 바와 같이 비할당 섹터(E, F)를 제거한다.

[0053] 한편, 프로세서(130)는 문서 파일에 분기문 또는 호출문이 있는지를 탐색하고, 분기문 또는 호출문이 있는 경우, 분기문 또는 호출문이 지시하는 분기 명령 또는 호출 명령을 실행한 이후에 후속되는 명령이 이전의 분기 명령 또는 호출 명령과 분리되도록 더미 코드를 삽입한다 (S4, S5). 즉, 소형 크기의 쉘 코드의 실행을 막기 위해 쉘 코드에 더미 코드를 삽입한다.

[0054] 프로세서(130)는 쉘 코드 기능을 사용하여 더미 코드를 삽입 할 위치를 검색한 후, 해당 위치에 삽입될 더미 코드의 크기를 계산한 후 문서의 속성을 사용하여 더미 콘텐츠를 생성한다.

[0055] 쉘 코드의 일반적인 목적은 원하는 작업을 수행하기 위해 대상 시스템의 권한을 획득하는 것이다. 시스템 제어를 얻기 위해, OS와의 상호 작용이 필요하며 Windows에 존재하는 Windows API가 이 프로세스에 호출된다. 공격자는 호출로 API를 내보내는 kernel32.dll의 메모리 주소를 가져온다. OS 환경에 상관없이 동작하는 것을 목적으로 하는 범용 쉘 코드의 경우에 Kernel32.dll의 주소를 하드 코딩할 수 없다.

[0056] 프로세스 환경 블록 (PEB), 구조화된 예외 처리(SEH) 및 쓰레드 환경 블록 (TEB)과 같은 다양한 방법들이 kernel32.dll의 기본 주소를 얻는 데 사용된다. 공격자는 이러한 방법을 사용하여 kernel32.dll의 기본 주소를 얻고, 해당 주소로 점프하거나 원하는 API를 호출하는 셸 코드를 작성한다. 즉, 공격자는 결국 악의적인 작업을 수행하기 위해 호출 명령 또는 분기 명령을 사용한다. 하기한 표 1은 분기문 및 호출문에 해당하는 명령어 세트를 보여준다.

[0057] [표 1]

[0058]

Call instruction	call ret
Branch instruction	je jne jz jnz ja jae jna jnae jb jbe jnb jnbe jc jnc jnp/jpo jp/jpe jecxz jg jge jng jnge jl jle jnl jnle jo/jno js/jns

[0059] 표 1을 사용하여 사용하여 문서 파일에서 호출문 또는 분기문을 검색하고, 호출문 또는 분기문이 검색된 지점에 더미 코드를 삽입한다. 그러나, 문서와 어셈블리 코드의 바이트 범위가 정확히 같기 때문에 호출문 또는 분기문을 잘못 찾을 수 있다. 따라서 더미 코드를 삽입할 위치를 검색하기 전에 사전 처리를 수행한다. 즉, 쉘 코드에서 자주 사용되는 함수의 프롤로그(Prologue) 부분을 검색하고, 프롤로그가 발견되면 해당 주소에서 프롤로그를 해체한 후 분기문 또는 호출문이 존재하는지 확인한다.

[0060] 문서 파일에 호출문 또는 분기문이 존재하는 경우, 삽입될 더미 코드의 개수가 계산된다. 함수의 프롤로그의 바이트 범위가 문서의 바이트 범위와 동일하기 때문에 검색된 프롤로그, 분기문 및 호출문의 수가 매우 많다. 따라서 헤더나 섹터 할당 테이블 같이 쉘 코드를 삽입할 수 없는 영역은 제외한다. 공격자가 헤더나 섹터 할당 테이블과 같은 영역에 쉘 코드를 삽입하면 문서 파일 자체가 성공적으로 열리지 않으므로 공격자가 이 영역에 쉘 코드를 삽입할 가능성은 거의 없다. 또한 더 정확한 더미 코드 삽입을 위해 기존의 쉘 코드를 모델링하고 100 바이트 당 호출문 또는 분기문의 수를 측정한다. 139 개의 범용 쉘 코드 샘플을 사용하여 100 바이트 당 분기문 또는 호출문의 수를 측정하면 평균 4.125가 산출되므로, 100 바이트 당 호출문 또는 분기문은 4~5개를 기준으로 검색된다고 할 수 있다.

[0061] 이와 같이, 프로세서(130)는 삽입할 더미 코드의 개수가 계산되면, 더미 코드의 크기를 계산한다. 즉, 호출문 또는 분기문의 피연산자 값에 따라 삽입할 더미 코드의 크기를 계산한 후에 분기문 또는 호출문이 이동하는 주소 사이에 더미 코드를 삽입한다. 연속하여, 분기문 또는 호출문의 후속되는 명령은 섹터와 섹터의 경계 주소에 쉘 코드를 위치시키는 데 사용된다. 즉, 더미 코드는 분기문 또는 호출문이 존재하는 섹터와 분기문 또는 호출문의 분기 명령 또는 호출 명령에 의한 이동 주소에 해당하는 섹터가 분리되도록 함으로써 섹터와 섹터의 경계 주소에 쉘 코드를 위치시킬 수 있다.

[0062] 도 5는 도 3의 더미 코드 삽입 단계에서 분기문 또는 호출문의 이동 주소가 계산 가능한 경우의 더미 코드 삽

입 과정을 설명하는 도면이고, 도 6은 도 3의 더미 코드 삽입 단계에서 분기문 또는 호출문의 이동 주소가 계산 불가능한 경우의 더미 코드 삽입 과정을 설명하는 도면이다. 이때, 도 5 및 도 6에서 분기문 또는 호출문이 존재하는 주소인 제1 주소(A), 분기문 또는 호출문의 이동 주소인 제2 주소(B), 섹터와 섹터의 경계주소인 제3 주소(C)라고 한다.

[0063] 도 5에 도시된 바와 같이, 'jmp' 라는 어셈블리 명령어가 문서 스트림 내에 발견되었으므로 해당 문서 파일에는 분기문이 존재하고, 해당 분기 명령은 '0xce6' 이라는 주소로 이동을 지시하고 있다.

[0064] 이때, 프로세서(130)는 각 섹터의 크기가 512바이트이므로 ceil 함수를 이용하여 제3 주소를 계산할 수 있다. 즉, 제3 주소는 (ceil(B/512)+1)×512가 된다.

[0065] 따라서, 프로세서(130)는 C-B 의 크기를 가지는 더미 코드를 분기문이 존재하는 섹터에 삽입하여 분기문이 존재하는 주소와 분기문의 이동 주소 사이를 섹터의 경계에 위치하게 할 수 있고, 그로 인해 쉘 코드의 분리가 가능하게 된다.

[0066] 분기문의 이동 주소가 하드 코딩되거나 분기 명령에 대해 상대적이라면, 삽입될 더미 코드의 크기는 도 5에 도시된 바와 같이 계산 될 수 있지만, 분기문의 이동 주소가 레지스터에 저장되고 레지스터에 저장된 주소로 이동하면 주소를 계산할 수 없다.

[0067] 따라서, 도 6에 도시된 바와 같이, 'jmp' 라는 어셈블리 명령어가 문서 스트림 내에 발견되었으므로 해당 문서 파일에는 분기문이 존재하고, 해당 분기 명령은 'QWORD [RDX]' 로 이동을 지시하고 있다. 이 경우, 정적 분석을 진행시 'RDX' 의 주소값을 알 수 없지만, 제1 주소와 제3 주소를 알고 있으므로 C-A의 크기를 가지는 더미 코드를 분기문이 존재하는 섹터에 삽입한다. 따라서, 분기문 다음의 명령어인 'ADD [RDX], CH' 를 분기문과 분리시킬 수 있어 쉘 코드의 무력화가 가능해진다.

[0068] 프로세서는 더미 코드를 섹터에 삽입할 때 몇 가지 조건을 만족해야 한다. 즉, 더미 코드는 문서 파일에 영향을 미치치 않아야 한다. 악성 문서를 검출하지 않고 정상 파일이나 악성 파일에 더미 코드를 삽입한다. 따라서 더미 코드는 정상적인 문서에 삽입될 때 문서 파일에 영향을 미치지 않으면서 악성 문서에 삽입될 때 프로그램을 종료하도록 해야 한다. 그러므로, 문서 스트림의 구성 요소인 레코드를 사용하여 더미 코드를 생성하고, 이 레코드는 주로 문서의 속성들과 관련된 값으로 구성된다.

[0069] [표 2]

[0070]

Type	Size	Value
49(font)	10	00 00 00 00 00 00 00 00 00 00

[0071] 표 2는 레코드 형식의 일례로서, 유형에서 문서 속성을 지정할 수 있다. 예제 유형에서 글꼴 속성이 지정된다. 레코드를 참조하는 값을 수정하여 문서에 적용되거나 적용되지 않도록 할 수 있다.

[0072] 따라서, 프로세서(130)는 더미코드로 레코드를 삽입하고 레코드의 참조 값을 설정하지 않았으므로 문서 파일에 영향을 미치지 않는다. 이때, 무의미한 코드로 더미 코드의 값을 채운다면, 그것을 Nop-sled로 사용할 수 있다. 따라서 더미 코드의 콘텐츠에서 프로그램을 종료하기 위해 INT 21H 인터럽트 코드를 더미 코드의 값으로 삽입한다.

[0073] 도 7은 더미 코드로 사용되는 종료의 인터럽트 코드에 대한 일례를 설명하는 도면으로서, 프로세서는 더미 코드에 종료의 인터럽트 코드를 삽입한다. 이때, INT 21H 인터럽트 코드는 Windows XP에서만 실행되고, DOS 내부 함수를 호출하는 함수이며, AH 레지스터 값에 의해 호출되는 함수를 결정하며, 0은 종료를 의미한다.

[0074] INT 21H인터럽트 코드는16 진 코드 내용으로 분기문과 호출문 사이에 레코드를 삽입한다. 이러한 종료의 인터럽트 코드를 더미 코드에 삽입함으로써 다음 주소가 더미 코드의 주소를 참조하는 경우에도 프로그램을 종료할 수 있고, 레코드가 참조되지 않으므로 문서의 의미를 유지할 수 있다.

[0075] 다시 도 3을 설명하면, 프로세서(130)는 문서 파일의 섹터들을 기 설정된 섹터 배치 조건을 만족하는 무작위화 알고리즘(Raddom Algorithm)을 사용하여 문서 파일 내 악성 코드가 분리되도록 콘텐츠의 재배치를 수행한다(S6).

[0076] 기존의 콘텐츠 무작위화 알고리즘은 임의의 경우에 따라 제한이 있다. 예를 들어, 무작위로 적용 되더라도 섹터

가 원래 위치로 우연히 재배치 될 수 있다.

[0077] 따라서, 프로세서(130)는 악성 코드로부터 문서 파일의 방어 효과를 극대화하기 위해 섹터 배치 조건을 만족하는 무작위화 알고리즘을 수행한다. 섹터 배치 조건은 섹터를 초기 위치로 재배치 하지 않는 제1 조건 및 무작위화 알고리즘을 사용하기 이전에 인접한 섹터들은 다시 인접되지 않도록 배치하는 제2 조건을 포함한다.

[0078] 도 8은 도 3의 콘텐츠 재배치 단계에서 섹터 배치 조건중 제1 조건을 만족하는 재정렬 과정을 설명하는 도면이고, 도 9는 도 3의 콘텐츠 재배치 단계에서 섹터 배치 조건 중 제2 조건을 만족하는 재정렬 과정을 설명하는 도면이다.

[0079] 제1 조건은 섹터를 원래 위치로 다시 배치하지 않는 것이다. 콘텐츠 무작위화는 임의의 경우에 따라 섹터의 위치를 변경하지 않을 수 있고, 이는 악성 코드의 방지율을 낮추는 요인이다. 섹터를 관리 할 때, 섹터를 포지셔닝되지 않은 위치로 이동한다. 예를 들어, 도 8에 도시된 바와 같이, 섹터 # 1의 A는 섹터 # 6으로 위치로 이동하고, 섹터 # 2의 B는 섹터 # 1로 위치로 이동한다.

[0080] 제2 조건은 콘텐츠 무작위화 이전에 인접한 섹터는 다시 인접될 수 없도록 하는 것이다. 콘텐츠 무작위화 알고리즘이 적용 되더라도 인접한 섹터가 다시 인접해 있으면 쉘 코드가 분리되지 않을 가능성이 높다. 따라서, 서로 인접한 섹터는 무작위화 알고리즘을 적용하여 원래 인접한 섹터에서 멀리 이동되어야 한다.

[0081] 예를 들어,도 9에 도시된 바와 같이, A 섹터와 B 섹터들은 콘텐츠 무작위화 이 전에 인접해 있다. 따라서 섹터 # 1의 A와 섹터 # 2의 B는 무작위화 알고리즘 적용시 다시 인접해 있을 수 없도록 섹터 # 1의 위치에 A, 섹터 # 4의 위치에 B 가 위치되도록 하여 서로 분리한다.

[0082] 이와 같이, 본 발명의 일 실시예에 따른 문서의 악성 코드 무력화 서비스 제공 방법은 플랫폼에 관계없이 다양한 프로그램에 적용 할 수 있고, 더미 코드의 삽입만 적용되더라도 악성 문서의 악용을 방어 할 수 있다. 그러나, 문서 파일에 삽입할 더미 코드의 크기가 매우 크게 되면 파일의 크기가 커진다.

[0083] 예를 들어, 쉘 코드의 분기문 또는 호출문이 현재 주소에서 멀리 떨어진 주소로 이동하면 쉘 코드의 실행을 막기 위해 대형 더미 코드를 삽입해야 한다. 따라서, 본 발명은 더미 코드의 삽입 후에 콘텐츠 재배치 과정을 적용하여 더 작은 크기의 더미 코드를 삽입할 수 있다. 즉, 쉘 코드가 섹터 경계에 위치 할 수 있을 정도의 더미 코드를 삽입하면 되므로 더미 코드의 크기가 크지 않아도 된다.

[0084] 이때, 더미 코드의 크기는 쉘 코드에서 분기문 또는 호출문 검색하여 계산된다. Metasploit, ExploitDB 및 Shellstorm에서 수집 한 쉘 코드를 기반으로 100 바이트 당 분기문 또는 호출문의 수를 검색한다.

[0085] 콘텐츠 재배치 과정이 적용되면 뷰어 프로그램은 무한 루프에 빠지게 된다. 이는 악성 문서의 공격이 호출문 또는 분기문에 의해 이동 주소로 점프할 때 원래 점프하려는 주소가 아닌 다른 주소로 점프하기 때문에 발생한다. 그러므로, 악성 문서의 경우, 원래 이동 주소가 아닌 다른 주소로 점프를 계속하게 된다. 따라서, 본 발명은 더미 코드에 종료의 인터럽트 코드를 삽입하여 호출문 또는 분기문에 의한 이동 주소로 이동할 수 있도록 한다.

[0086] 도 10은 본 발명의 일 실시예에 따른 문서의 악성 코드 무력화 서비스 제공 방법이 악성 문서에 가짜 쉘 코드와 진짜 쉘 코드를 삽입한 경우에 쉘 코드의 실행을 차단하는 방어 동작을 설명하는 도면이다.

[0087] 도 10에 도시된 바와 같이, 공격자는 문서 파일에 악의적인 행위를 하지않는 가짜 쉘 코드(Fake shellcode)와 악의적인 행위를 수행하는 진짜 셀 코드(Real shellcode)를 삽입 할 수 있다.

[0088] 가짜 쉘 코드에 더미코드를 삽입하더라도 더미코드의 내용이 인터럽트 코드로 구성되어 있지 않을 경우, 또는 인터럽트 코드로 구성되어 있더라도 프로그램 실행 환경이 windows XP 보다 높은 버전일 경우, 해당 더미 코드는 nop-sled로 사용되어 결국은 진짜 쉘 코드가 실행될 수 있다. 그러나, 본 발명의 실시예에서는 모든 호출문이나 분기문에 더미코드를 삽입하기 때문에 가짜 쉘 코드를 사용하여 우회할 수 없게 된다.

[0089] 이상에서 설명한 본 발명의 실시예에 따른 문서의 악성 코드 무력화 서비스 제공 방법은, 컴퓨터에 의해 실행되는 프로그램 모듈과 같은 컴퓨터에 의해 실행 가능한 명령어를 포함하는 기록 매체의 형태로도 구현될 수 있다. 이러한 기록 매체는 컴퓨터 판독 가능 매체를 포함하며, 컴퓨터 판독 가능 매체는 컴퓨터에 의해 액세스될 수 있는 임의의 가용 매체일 수 있고, 휘발성 및 비휘발성 매체, 분리형 및 비분리형 매체를 모두 포함한다. 또한, 컴퓨터 판독가능 매체는 컴퓨터 저장 매체를 포함하며, 컴퓨터 저장 매체는 컴퓨터 판독가능 명령어, 데이터 구조, 프로그램 모듈 또는 기타 데이터와 같은 정보의 저장을 위한 임의의 방법 또는 기술로 구현된 휘발

성 및 비휘발성, 분리형 및 비분리형 매체를 모두 포함한다.

[0090] 전술한 본 발명의 설명은 예시를 위한 것이며, 본 발명이 속하는 기술분야의 통상의 지식을 조사 자는 본 발명의 기술적 사상이나 필수적인 특징을 변경하지 않고서 다른 구체적인 형태로 쉽게 변형이 가능하다는 것을 이해할 수 있을 것이다. 그러므로 이상에서 기술한 실시예들은 모든 면에서 예시적인 것이며 한정적이 아닌 것으로 이해해야만 한다. 예를 들어, 단일형으로 설명되어 있는 각 구성 요소는 분산되어 실시될 수도 있으며, 마찬가지로 분산된 것으로 설명되어 있는 구성 요소들도 결합된 형태로 실시될 수 있다.

[0091] 또한, 본 발명의 방법 및 시스템은 특정 실시예와 관련하여 설명되었지만, 그것들의 구성 요소 또는 동작의 일부 또는 전부는 범용 하드웨어 아키텍쳐를 갖는 컴퓨터 시스템을 사용하여 구현될 수도 있다.

[0092] 본 발명의 범위는 상세한 설명보다는 후술하는 특허청구범위에 의하여 나타내어지며, 특허청구범위의 의미 및 범위 그리고 그 균등 개념으로부터 도출되는 모든 변경 또는 변형된 형태가 본 발명의 범위에 포함되는 것으로 해석되어야 한다.

부호의 설명

[0093] 100 : 문서의 악성 코드 무력화 서비스 제공 장치

110 : 통신 모듈

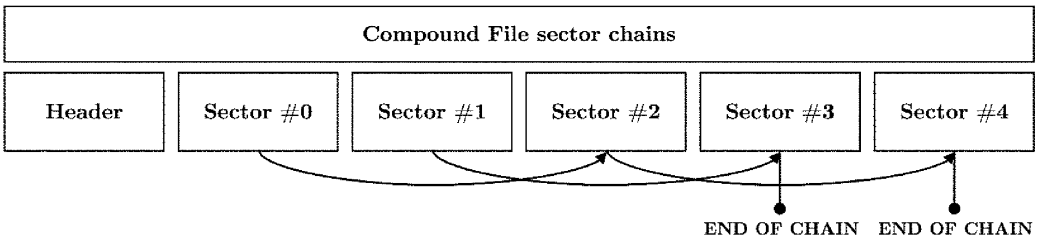
120 : 메모리

130 : 프로세서

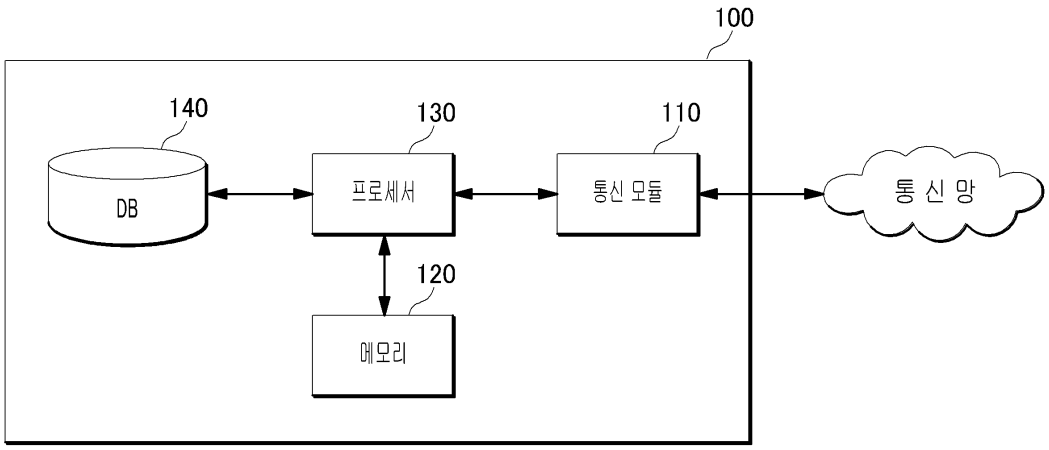
140 : 데이터베이스

도면

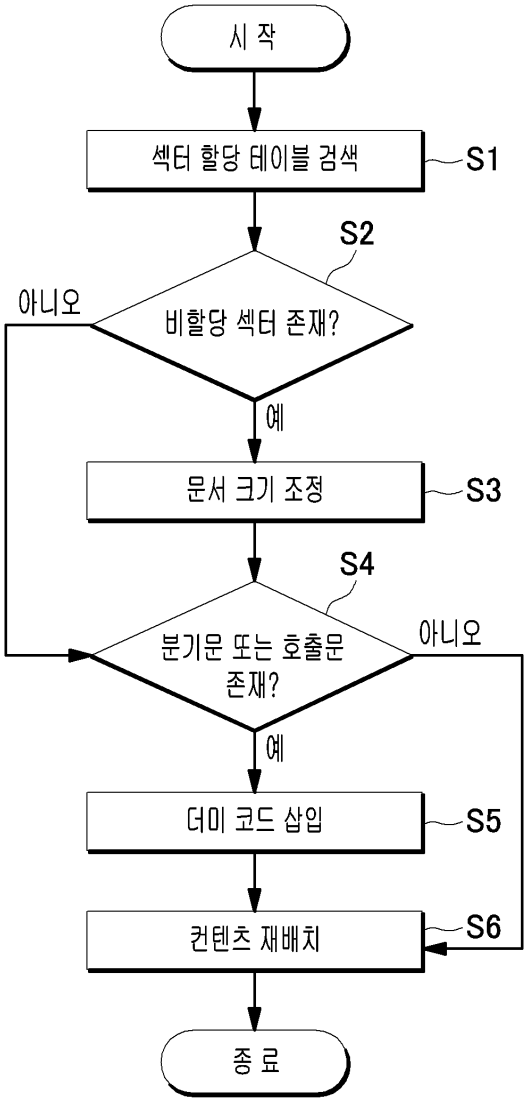
도면1



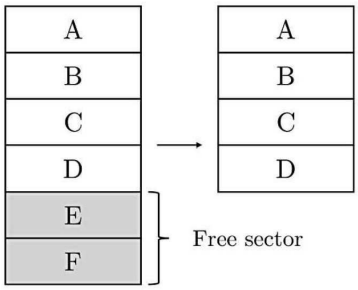
도면2



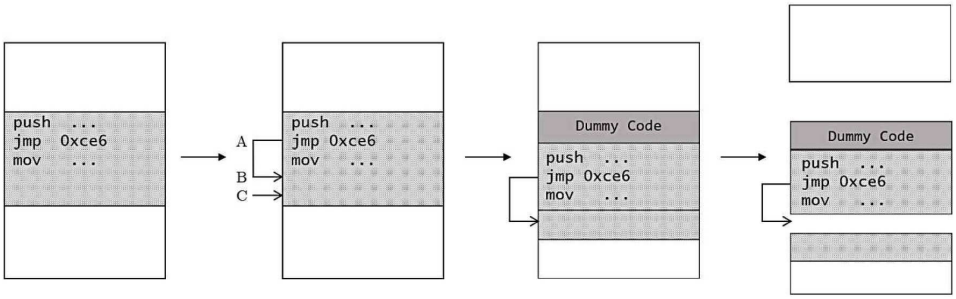
도면3


시 작
섹터 할당 테이블 검색
S1
S2
비할당 섹터 존재?
아니오
예
문서 크기 조정
S3
S4
분기문 또는 호출문 존재?
아니오
예
더미 코드 삽입
S5
컨텐츠 재배치
S6
종 료

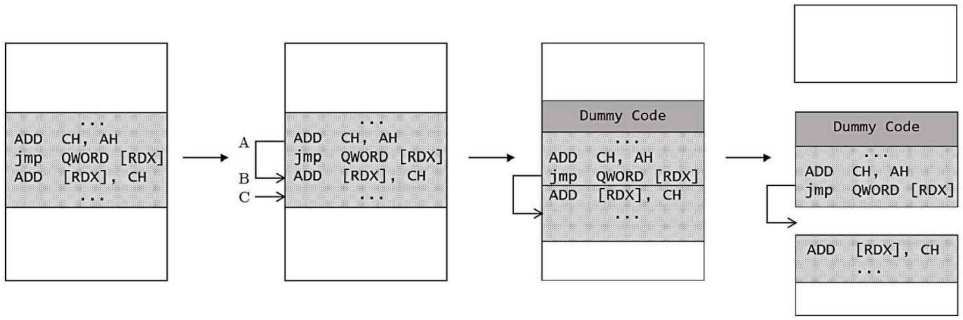
도면4


A
B
C
D
E
F
A
B
C
D
Free sector

도면5



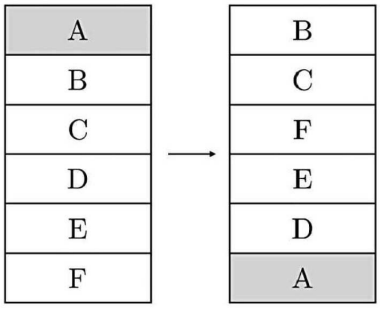
도면6



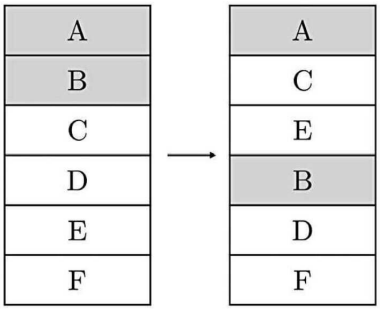
도면7

Assembly code	Hex code
MOV AH, 0B	B4 00
INT 21H	CD 21

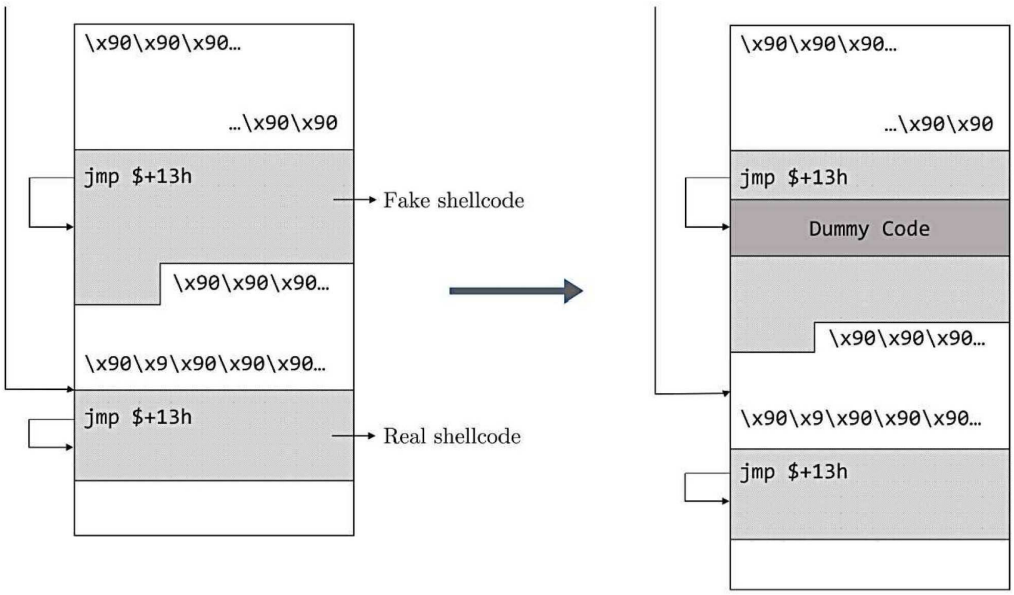
도면8



도면9


A
B
C
D
E
F
A
C
E
B
D
F

도면10


\x90\x90\x90…
…\x90\x90
jmp $+13h
Fake shellcode
\x90\x90\x90…
\x90\x9\x90\x90\x90…
jmp $+13h
Real shellcode
\x90\x90\x90…
…\x90\x90
jmp $+13h
Dummy Code
\x90\x90\x90…
\x90\x9\x90\x90\x90…
jmp $+13h