



(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2011년05월11일
(11) 등록번호 10-1033191
(24) 등록일자 2011년04월28일

(51) Int. Cl.

G06F 11/30 (2006.01) G06F 11/07 (2006.01)

G06F 21/22 (2006.01)

(21) 출원번호 10-2010-0014953

(22) 출원일자 2010년02월19일

심사청구일자 2010년02월19일

(56) 선행기술조사문헌

KR100586500 B1

KR100645983 B1

논문_001

(73) 특허권자

고려대학교 산학협력단

서울 성북구 안암동5가 1

(72) 발명자

문중섭

서울특별시 강동구 고덕동 고덕시영아파트 7동 404호

이영훈

경기도 광주시 오포읍 양벌리 대주1차아파트 110동 410호

(74) 대리인

특허법인 엠에이피에스

전체 청구항 수 : 총 4 항

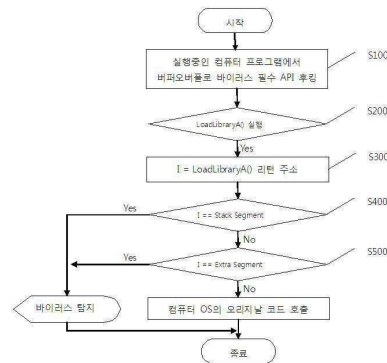
심사관 : 정현수

(54) 메모리 실행영역 추적을 사용한 버퍼오버플로 악성코드 탐지방법

(57) 요약

본 발명은 메모리 실행영역을 추적하여 버퍼오버플로 유형의 바이러스를 실시간 탐지하는 기법으로 안티바이러스 프로그램에 적용 가능한 탐지 방법이다. 즉 Microsoft(주)의 Windows운영체제에서 사용되는 수많은 프로그램 중에서 정상 프로그램과 악성 프로그램을 구별하는 방법으로 본 발명은 프로그램 실행 시 메모리 실행 영역(Stack과 Heap)을 추적 및 분석하여 버퍼오버플로 유형의 바이러스를 탐지하는 방법이다. 또한, 이 방법은 기존 안티바이러스 프로그램에서 사용하는 시그니처 탐지방법이 아니라 프로그램 실행과정에서 탐지하는 것으로, 알려지지 않은 버퍼오버플로 바이러스를 탐지하여 준다.

대표도 - 도1



특허청구의 범위

청구항 1

버퍼오버플로우 바이러스의 탐지방법에 있어서,
실행중인 컴퓨터 프로그램에서 미리 설정된 API(Application Program Interface)의 실행 여부를 검출하는 단계,
상기 미리 설정된 API가 실행되는 것으로 검출된 경우, 상기 API를 후킹하는 단계 및
상기 API의 메모리 주소를 분석하여 버퍼오버플로우 바이러스를 탐지하는 단계
를 포함하는 버퍼오버플로우 바이러스의 탐지 방법.

청구항 2

제 1 항에 있어서,
상기 미리 설정된 API는 LoadLibraryA()인 것인 버퍼오버플로우 바이러스의 탐지 방법.

청구항 3

제 1 항에 있어서,
상기 버퍼오버플로우 바이러스를 탐지하는 단계는,
상기 API의 메모리 주소가 스택 세크먼트에 해당하는 경우, 상기 버퍼오버플로우 바이러스를 탐지하는 버퍼오버플로우 바이러스의 탐지 방법.

청구항 4

제 1 항에 있어서,
상기 버퍼오버플로우 바이러스를 탐지하는 단계는,
상기 API의 메모리 주소가 엑스트라 세크먼트에 해당하는 경우, 상기 버퍼오버플로우 바이러스를 탐지하는 버퍼오버플로우 바이러스의 탐지 방법.

명세서

기술분야

[0001] 본 발명은 개인용 PC에서 사용하는 프로그램 중에서 정상 프로그램과 버퍼오버플로 악성 프로그램을 구별하기 위해 메모리 실행 영역을 추적하여 분석하는 방법이다. 따라서 관련된 기술 분야는 안티바이러스 프로그램 구현 및 시스템 보안 분야가 관련 기술분야이다.

배경기술

[0002] 본 발명이 목적으로 하는 "힙 스프레이(Heap Spray)" 방식의 악성코드를 찾는 방법은 공개된 경우가 없다. 다만, 다른 종류의 악성코드를 찾는다든지, 일반적으로 악성 코드를 찾는 방법을 제시한 여러 가지 특허와 그 기반 기술에 대한 논문은 아래와 같다.

[0003] 악성코드 탐색에 대한 일반적인 방법에 대한 특허.

[0004] 특허 문헌:

- [0005] 1. 공개번호: 10-2009-0109154, "악성코드 차단장치, 시스템 및 방법", 출원인: 한국전자통신연구원
- [0006] 2. 공개번호: 10-2009-0079625, "악성코드 진단 및 치료장치", 출원인: 주식회사 안철수 연구소
- [0007] 3. 출원번호: 10-2009-0050662, "가상환경을 통한 악성코드탐지방법", 출원인: 이글루시큐리티(주)
- [0008] 일반 논문들
- [0009] (메모리 추적기법에 관한 논문들)
- [0010] 4. H. M. Sun, Y. H. Lin, "API Monitoring System for defeating Worms and Exploits in MS-Windows System," ACISP 2006, pp 3-8, July 2006.
- [0011] 5. O. Ruwase and M. S. Lam Lam, "A Practical Synamic Buffer Overflow Detector," Proceedings of the 11th Annual Network and Distributed System Security Symposium, pp 3-5, Feb 2004.
- [0012] (동적 탐지 기법에 관한 논문들)
- [0013] 6. 김재현, 강신현, "네트워크 트래픽 특성을 이용한 스캐닝 웹 탐지기법," 정보보호학회논문지, 17(1), pp59-64, 2007년2월.
- [0014] 7. D. Mohanty, "Anti-Virus Evasion Technoques and Countermeasures," http://www.infosecwriters.com/text_resources/pdf/AV_Evasion.pdf, pp 14-11.

발명의 내용

해결하려는 과제

- [0015] 개인용 PC에서 바이러스 감염을 방지하기 위해 안티바이러스 프로그램을 사용한다. 그런데 기존의 거의 모든 안티바이러스 프로그램들은 바이러스 탐지를 위해 바이러스 파일의 특징을 데이터베이스화하여 단순 비교하여 정상 프로그램인지 바이러스인지 판별하는 시그니처 탐지기법을 사용한다. 그러나 이 방법의 경우 알려지지 않은 새로운 바이러스 또는 안티바이러스 우회 기법을 적용한 변종 바이러스를 탐지하지 못하는 문제점이 있다. 이를 해결하기 위해 안티바이러스 프로그램은 새로운 바이러스의 특징을 데이터베이스화하여 탐지 규칙을 갱신(위 특허 문서 1, 2 3)하고 있으나 현재의 안티바이러스 프로그램이 새로운 바이러스를 탐지하는 근본적인 방법은 없다. 따라서, 새로운 바이러스 프로그램을 탐지하기 위해서는 프로그램 실행과정에서 정상 프로그램인지 바이러스 프로그램인지 구별하여 이를 탐지하는 기술이 필요하다.
- [0016] 위 배경기술에서 적시 되어 있는 특허 문헌들은 어떤 파일(주로 이메일에 첨부되어 오는)이 악성코드라고 알고 있을 때, 쉽게 분석하기 위한 장치(문헌 1, 2)이거나, 가상환경에서 해당 파일을 실행시켜(문헌 3), 악성코드처럼 작동하는지 확인하는 방법이다. 또한, 논문 5에서는 API의 주소를 검색하는 기법을 사용하였으나, 이 주소를 변경시키는 악성코드는 탐지 못 한다. 문헌 6도 5와 마찬가지로 오브젝트를 변경시키는 코드를 찾는 방법을 제시했으나, 마찬가지로 최근의 악성코드는 이 방법을 피해간다.
- [0017] 따라서, 현재에 발표된 악성코드 탐지를 위한 여러 방법은, 근래에 새로 나타난, 프로그램 메모리의 힙 영역에 숨겨진 작동 코드를 찾아내는 것은 불가능하다.

과제의 해결 수단

- [0018] 1. 버퍼오버플로 바이러스에서 사용되는 필수 API 선정한다. 즉, 악성코드들이 주로 사용하는 API가 무엇인지 파악하고, 2. 필수 API 추적을 위한 API 후킹 기법을 사용하여, 해당 API를 파악하고, 3. 스택 영역 메모리 실시간 분석 기법을 사용하여, 프로그램 메모리의 스택에 숨겨있는 코드와, 4. 힙 영역 메모리 실시간 분석 기법을 통하여, 프로그램 메모리의 힙에 숨겨 있는 악성코드를 탐지한다.

발명의 효과

- [0019] 본 발명에서 제시하는 방법을 사용하면, 개인용 PC에서 사용하는 프로그램을 실시간 감시하여 바이러스 여부를

탐지할 수 있다. 따라서 효과는

[0020]

[0021]

1. 개인용 PC에서 알려지지 않은 버퍼오버플로 바이러스를 차단할 수 있다.

[0022]

2. 안티바이러스 프로그램에서 이 발명을 적용할 경우, 새로 나타나는 버퍼오버플로 바이러스를 규칙을 새로 추가하지 않고, 탐지할 수 있다.

[0023]

[0024]

따라서 이 방법을 안티바이러스 프로그램의 탐지규칙 갱신 미실시로 인한 버퍼오버플로 바이러스를 감염을 방지할 수 있어 개인용 PC의 보안을 한층 강화할 수 있다.

도면의 간단한 설명

[0025]

도 1. 버퍼오버플로 바이러스 탐지를 위해 프로그램의 메모리 실행 영역 분석 기법을 설명하는 것으로서 두 단계의 방법으로 구성되어 있다.

도 2. LoadLibraryA()의 리턴주소 I가 Stack Segment에 포함되는지 비교하는 루틴(S400)을 상세 설명하는 것이다.

도 3. LoadLibraryA()의 리턴주소 I가 Extra Segment에 포함되는지 비교하는 루틴(S500)을 상세 설명하는 것이다.

발명을 실시하기 위한 구체적인 내용

[0026]

개인용 PC에서 버퍼오버플로 바이러스 탐지를 위해 프로그램이 실행될 때 메모리의 실행영역을 분석하여 정상 프로그램과 구별하여야 한다. 정상 프로그램에서 Application Program Interface(API) 호출이 발생할 경우 현재 쓰레드(프로세스)의 Code Segment영역에서 실행된다. 그러나 버퍼오버플로 공격코드에서는 스택(데이터 영역)과 같은 비정상 영역에서 API 호출이 발생한다. 또한, Heap Spray기법의 공격코드는 힙 영역 할당에 사용되는 Extra Segment(추가 데이터 영역)에서 API호출이 발생한다.

[0027]

본 발명에서, 사용하는 방법은 간단히 설명하면, 개인용 PC환경(Microsoft(주) Windows)에서, 먼저, 버퍼오버플로 바이러스에 사용되는 필수적인 API를 선정하고 선정된 API를 실시간 감시한다. 다음으로, 선정된 API의 메모리 영역을 분석하여 스택 또는 힙 영역 할당에 사용되는 Extra Segment영역인지를 판단하여 버퍼오버플로 바이러스 여부를 판단한다. 이 방법은 도 1에 잘 설명되어 있다. 즉, Microsoft(주) Windows를 사용하는 컴퓨터에서, 가장 먼저 실행중인 컴퓨터 프로그램에서 버퍼오버플로 바이러스에서 필수적으로 사용되는 LoadLibraryA() API를 후킹한다(S100). 만일 실행중인 프로그램에서 LoadLibraryA() API가 사용되면 LoadLibraryA()의 리턴주소를 변수 I에 저장하고(S300), I의 주소가 Stack Segment에 속하거나 (S400), 또는 이 프로그램이 사용하는 Extra Segment(S500)에 포함될 경우 버퍼오버플로 바이러스로 탐지한다.

[0028]

(스택 영역 침범 악성코드 탐지 방법)

[0029]

여기에서, 어떤 코드가 스택 세그먼트에서 실행되는가 확인하기 위해서는 다음과 같은 상세한 방법을 사용한다. 이는 도 2에 플로어차트로 보여준다. 도 2는

[0030]

LoadLibraryA()의 리턴주소 I가 Stack Segment에 포함되는지 비교하는 루틴(S400)을 상세 설명하는 것으로 먼저 현재 쓰레드가 생성한 스택 영역을 얻기 위해 fs 레지스터가 가리키는 메모리의 주소로부터 오프셋 값 0x4위치에 저장된 값(4 바이트)을 레지스터 eax에 저장하고(S410), 이를 stack top에 저장한다(S420). 또한, fs 레지스터 가리키는 메모리의 주소로부터 오프셋 값 0x8위치에 저장된 값(4 바이트)을 eax에 저장하고(S430), 이를 stack bottom에 저장한다(S440). 이후 LoadLibraryA()의 리턴주소값 I(S300)의 주소가 Stack bottom과 stack top사이에 위치할 경우 버퍼오버플로 바이러스로 인식한다.

[0031]

여기서, 레지스터란 컴퓨터의 CPU 내에 존재하는 소수의 메모리들인데, Intel의 Pentium은 Eax, FS, 등등의 이름을 가지고, 이 메모리는 4 바이트를 가지고, 내부에 데이터를 가진다. 이때, 도 2에, FS:[04] 같은 표현은, FS가 가진 값을 메모리의 베이스(시작) 주소로 하여, 4 바이트 떨어져 있는 메모리라는 표현방법이다. 예를 들어, FS에 123456 들어 있을 때, FS:[04]는, 메모리 주소가 123460(10진법)번지인 곳을 의미한다.

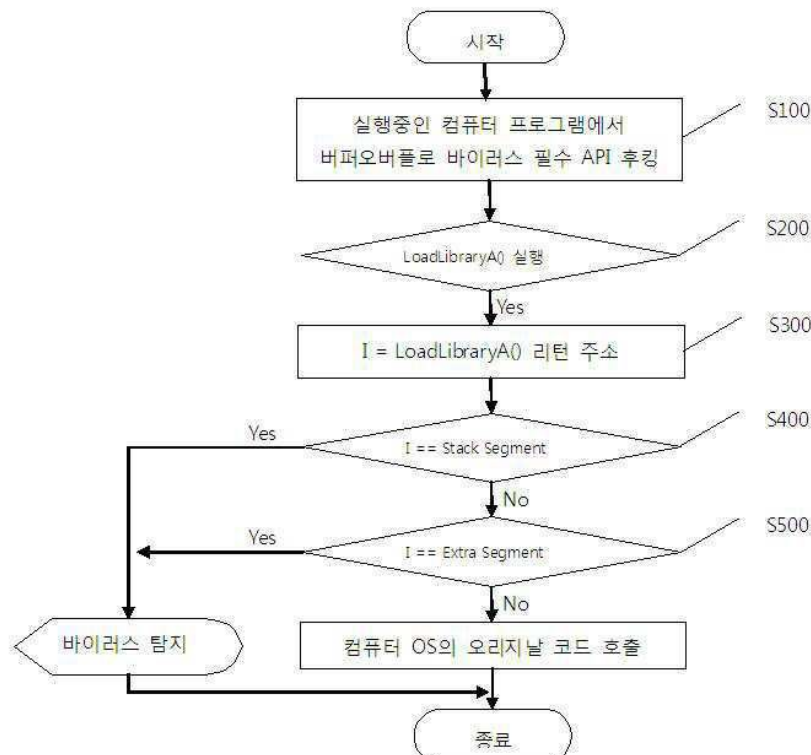
[0032]

(힙 영역 탐지 방법)

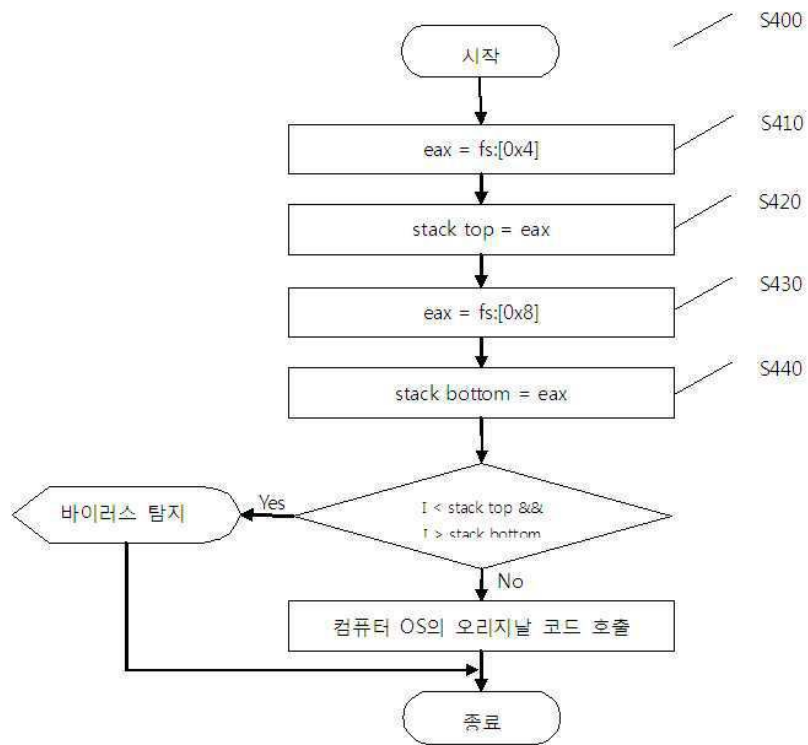
- [0033] 다음으로, Heap spray방식으로 사용하는 악성코드를 탐지하는 방법은 도 3에 플로우차트 표현되어 있다. 즉, 도 3은, 앞서 설명한 LoadLibraryA()의 리턴주소 I가 Extra Segment에 포함되는지 비교하는 루틴(S500)을 상세 설명하는 것이다.
- [0034] 현재 스레드(Thread)가 생성한 힙영역은 Linked List로 구현되어 있는데 이는 Process Environment Block에 저장된 Process Heap값을 이용해 파악할 수 있다.
- [0035] 먼저, fs가 가지고 있는 메모리의 주소 값을 베이스(기본)로 하여, 0x18(16진법 18, 즉 10진 주소로는 24번째) 메모리의 4 바이트 데이터를 레지스터 EAX(이값은 Thread Information Block(TIB)의 주소가 된다)에 보관한다(S510). 그 다음 TIB의 오프셋 값 0x30에 위치한 PEB값(S520)을 얻는다. 이값을 다시, EAX레지스터에 보관한다.
- [0036] 이후 TEB구조체의 오프셋 값 90h값에서 현재 스레드가 생성한 heap block의 Linked List의 시작주소를 process heap에 저장한다 (S550, S560).
- [0037] 그리고 Heap block의 linked list의 개수를 TEB구조체의 오프셋 값 88h의 위치에서 heap block의 개수를 얻어 heap count에 저장한다(S530, S540). 이후 heap count/4 만큼 반복하면서(S580, 4 바이트씩 데이터를 가져온다). 도 1.에서 저장한 LoadLibraryA()의 리턴주소값 I(S300)의 주소가 heap[i] 와 heap[i]+10000 에 포함될 경우(S590) 버퍼오버플로 바이러스로 인식한다.
- [0038] 도 2, 3에서 사용하는 레지스터 이름, EAX, ECX는 Pentium Process에서 사용하는 범용 레지스터로써, EBX, EDX 등, 어느 것을 사용해도 무방하다. 즉, 4가지 범용 레지스터 중에서 임의로 사용해도 상관이 없다.

도면

도면1



도면2



도면3

