(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2011년12월20일
(11) 등록번호 10-1095071
(24) 등록일자 2011년12월09일

(51) Int. Cl.
G06F 9/06 (2006.01) *G06F 9/32* (2006.01)
G06F 9/42 (2006.01)
(21) 출원번호 10-2010-0019549
(22) 출원일자 2010년03월04일
심사청구일자 2010년03월04일
(65) 공개번호 10-2011-0100508
(43) 공개일자 2011년09월14일
(56) 선행기술조사문헌
US20070152854 A1*
KR100330437 B1
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
고려대학교 산학협력단

(72) 발명자
이희조

정구현

(74) 대리인
특허법인엠에이피에스

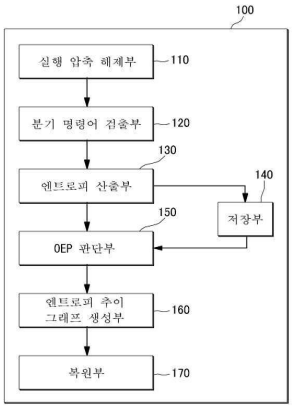
전체 청구항 수 : 총 13 항 심사관 : 복진요

(54) 엔트로피 분석을 이용한 실행 압축 해제 장치 및 그 방법

(57) 요 약

실행 압축을 해제하는 장치로서, 실행 압축 파일을 상기 실행 압축 파일에 포함된 압축 해제 모듈을 이용하여 압축 해제시키는 실행 압축 해제부, 상기 실행 압축 해제 과정에서 분기 명령어의 실행 여부를 검출하는 분기 명령어 검출부, 상기 분기 명령어의 검출 시 상기 실행 압축 해제된 데이터가 탑재된 메모리 영역의 엔트로피 값을 산출하는 엔트로피 산출부 및 상기 산출된 엔트로피 값의 변화 추이를 이용하여 오리지널 엔트리 포인트(Original Entry Point, OEP)를 판단하는 OEP 판단부를 포함한다.

대 표 도 - 도1



특허청구의 범위

청구항 1

실행 압축을 해제하는 장치로서,

실행 압축 파일을 상기 실행 압축 파일에 포함된 압축 해제 모듈을 이용하여 압축 해제시키는 실행 압축 해제부;

상기 실행 압축 해제 과정에서 분기 명령어의 실행 여부를 검출하는 분기 명령어 검출부;

상기 분기 명령어의 검출 시 상기 실행 압축 해제된 데이터가 탑재된 메모리 영역의 엔트로피 값을 산출하는 엔트로피 산출부; 및

상기 산출된 엔트로피 값의 변화 추이를 이용하여 오리지널 엔트리 포인트(Original Entry Point, OEP) 를 판단하는 OEP 판단부를 포함하되,

상기 엔트로피 산출부는 상기 실행 압축 해제 과정에서 상기 실행 압축 해제된 데이터가 탑재된 메모리 영역을 동적으로 판단하는 것인 실행 압축 해제 장치.

청구항 2

제 1 항에 있어서,

상기 분기 명령어 검출부는 상기 실행 압축 해제 과정에서 실행되는 명령어가 API 명령어인 경우 검출 대상에서 해당 명령어를 제외시키는 실행 압축 해제 장치.

청구항 3

제 1 항에 있어서,

상기 분기 명령어는 JMP 또는 CALL 인 실행 압축 해제 장치.

청구항 4

제 1 항에 있어서,

상기 분기 명령어 검출부는 검출된 가장 최근의 n개의 분기 명령어에 대해 캐시 메모리를 이용하는 것인 실행 압축 해제 장치.

청구항 5

제 1 항에 있어서,

상기 엔트로피 산출부는 하기의 수학식 2에 의해 수행되되,

하기의 수학식 2 의 p(xi)는 xi가 발생할 확률이고, I는 이상 확률 변수 X의 자기 정보량(Self-information) 을 의미하는 실행 압축 해제 장치.

(수학식 2)

$$H(X) = \sum_{i=1}^{n} p(x_i) I(x_i) = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i)$$

청구항 6

제 1 항에 있어서,

상기 OEP 판단부는 덤프한 메모리 값이 더미 데이터인 경우 그 값을 삭제하는 실행 압축 해제 장치.

청구항 7

제 1 항에 있어서,

상기 분기 명령어 검출에 따른 엔트로피 값의 변화 추이를 도시한 엔트로피 추이 그래프를 생성하는 엔트로피 추이 그래프 생성부를 더 포함하는 실행 압축 해제 장치.

청구항 8

제 1 항에 있어서,

상기 OEP 판단부에 의해 판단된 오리지널 엔트리 포인트 이후의 메모리 분석을 통해 원본 파일을 복원하는 복원부를 더 포함하는 실행 압축 해제 장치.

청구항 9

제 1 항에 있어서,

상기 OEP 판단부는 상기 엔트로피 산출부에 의해 산출된 엔트로피 값이 미리 설정된 엔트로피 최소값보다 작거나, 미리 설정된 엔트로피 최대값보다 큰 경우 산출된 엔트로피 값을 이용하여 판단된 오리지널 엔트리 포인트는 제외시키는 실행 압축 해제 장치.

청구항 10

실행 압축을 해제하는 방법으로서,

실행 압축 파일을 상기 실행 압축 파일에 포함된 압축 해제 모듈을 이용하여 압축 해제시키는 단계;

상기 실행 압축 해제 과정에서 분기 명령어의 실행 여부를 검출하는 단계;

상기 분기 명령어의 검출 시 상기 실행 압축 해제된 데이터가 탑재된 메모리 영역의 엔트로피 값을 산출하는 단계; 및

상기 산출된 엔트로피 값이 수렴된 이후 실행 흐름이 옮겨진 주소를 오리지널 엔트리 포인트로 판단하는 단계를 포함하되,

상기 엔트로피 값을 산출하는 단계는 상기 실행 압축 해제 과정에서 상기 실행 압축 해제된 데이터가 탑재된 메모리 영역을 동적으로 판단하는 것인 실행 압축 해제 방법.

청구항 11

제 10 항에 있어서,

상기 분기 명령어 검출 단계는 상기 실행 압축 해제 과정에서 실행되는 명령어가 API 명령어인 경우 검출 대상에서 해당 명령어를 제외시키는 실행 압축 해제 방법.

청구항 12

제 10 항에 있어서,

상기 오리지널 엔트리 포인트 판단 단계에 의해 판단된 오리지널 엔트리 포인트 이후의 메모리 분석을 통해 원본 파일을 복원하는 단계를 더 포함하는 실행 압축 해제 방법.

청구항 13

제 10 항에 있어서,

상기 오리지널 엔트리 포인트 판단 단계는 상기 엔트로피 산출 단계에 의해 산출된 엔트로피 값이 미리 설정된 엔트로피 최소값보다 작거나, 미리 설정된 엔트로피 최대값보다 큰 경우 산출된 엔트로피 값을 이용하여 판단된 오리지널 엔트리 포인트는 제외시키는 것을 특징으로 하는 실행 압축 해제 방법.

명 세 서

기 술 분 야

[0001] 본 발명은 실행 압축 해제 장치 및 그 방법에 관한 것으로, 보다 상세하게는, 악성코드 분석 및 제거 등에 사용될 수 있는 실행 압축 해제 장치 및 그 방법에 관한 것이다.

배 경 기 술

[0002] 실행 압축은 일반적으로 많이 알려져 있는 zip, rar 과 같은 방식으로 압축 및 암호화 알고리즘을 이용하여 파일의 크기를 압축하거나 역공학에 의한 프로그램 보호 등의 용도로 개발되었다. 다만, 실행 압축은 데이터가 아닌 실행할 수 있는 파일을 압축한 것이다. 그러나, 최근 들어 악성코드 제작자에 의해 악성코드의 변종을 제작하기 위하여 악용되고 있다. 이는 실행 압축 기법을 이용한 악성 코드들이 파일 크기가 작아 전파 속도가 빠르며 원래 코드를 변형시킴으로써 악성 코드의 분석을 어렵게 하기 때문이다. 이에 따라 실행 압축 해제 기술에 관한 연구도 많이 이루어져왔다.

[0003] 실행 압축 해제 기술에 관한 연구 방법은 인력을 이용하여 수동으로 직접 악성코드를 분석하는 방법, 특정 실행 압축 기법에 대해 작동하는 알고리즘을 개발하는 방법, 또는 범용적으로 사용할 수 있는 실행 압축 해제 기술을 개발하는 방법 등 크게 세 가지로 분류할 수 있다.

[0004] 수동으로 직접 악성코드를 분석하는 방법의 경우 실행 프로그램의 수많은 명령어들을 일일이 분석하여야 하므로 너무 많은 시간을 소모하는 문제점이 있었다. 또한 특정 실행 압축 기법에 대해 작동하는 알고리즘 개발 방법은 각 실행 압축 기법에 맞는 별도의 알고리즘을 개발해야 하므로 새로운 악성 코드나 기존의 실행 압축 기법이 일부 변형된 코드의 경우 기민하게 대처하기 어렵다는 문제가 있었다.

[0005] 따라서 최근에는 특정 실행 압축 기법에 의존하지 않는 범용적인 실행 압축 해제에 관한 연구가 진행되어, OmniUnpack, Renovo 등 기법이 개발되었지만 이들은 오리지널 엔트리 포인트가 정확히 어디인지 알 수 없어 한계가 있었다. 오리지널 엔트리 포인트는 실행 압축을 한 원 프로그램의 진입점을 말하는 것이다. 오리지널 엔트리 포인트 이후에 실행되는 코드가 실행 압축을 하기 이전의 원래의 프로그램이 되기 때문에, 오리지널 엔트리 포인트를 찾는 것이 실행 압축 해제 기술의 가장 큰 핵심이다.

발명의 내용

해결하려는 과제

[0006] 본 발명은 상기한 바와 같이 선행 기술에 내재되었던 문제점을 해결하기 위해 창작된 것으로, 본 발명의 목적은

실행 압축 해제 과정이 진행될 때, 실행 압축 해제된 데이터가 탑재된 메모리 상태의 엔트로피 값의 변화를 관찰하여 오리지널 엔트리 포인트를 찾아낼 수 있는 실행 압축 해제 장치 및 그 방법을 제공하는데 있다.

과제의 해결 수단

[0007] 상술한 기술적 과제를 달성하기 위한 기술적 수단으로서, 본 발명의 제 1 측면은(일 실시예는) 실행 압축을 해제하는 장치로서, 실행 압축 파일을 상기 실행 압축 파일에 포함된 압축 해제 모듈을 이용하여 압축 해제시키는 실행 압축 해제부, 상기 실행 압축 해제 과정에서 분기 명령어의 실행 여부를 검출하는 분기 명령어 검출부, 상기 분기 명령어의 검출 시 상기 실행 압축 해제된 데이터가 탑재된 메모리 영역의 엔트로피 값을 산출하는 엔트로피 산출부 및 상기 산출된 엔트로피 값의 변화 추이를 이용하여 오리지널 엔트리 포인트(Original Entry Point, OEP) 를 판단하는 OEP 판단부를 포함하는 실행 압축 해제 장치이다.

[0008] 또한, 본 발명의 제 2 측면은(다른 실시예는) 실행 압축을 해제하는 방법으로서, 실행 압축 파일을 상기 실행 압축 파일에 포함된 압축 해제 모듈을 이용하여 압축 해제시키는 단계, 상기 실행 압축 해제 과정에서 분기 명령어의 실행 여부를 검출하는 단계, 상기 분기 명령어의 검출 시 상기 실행 압축 해제된 데이터가 탑재된 메모리 영역의 엔트로피 값을 산출하는 단계 및 상기 산출된 엔트로피 값이 수렴된 이후 실행 흐름이 옮겨진 주소를 오리지널 엔트리 포인트로 판단하는 단계를 포함하는 실행 압축 해제 방법이다.

발명의 효과

[0009] 전술한 본 발명의 과제 해결 수단에 의하면, 실행 압축 해제된 데이터가 탑재된 메모리 상태의 엔트로피 값의 변화를 관찰하여 오리지널 엔트리 포인트를 찾아낸다.

[0010] 또한, 전술한 본 발명의 과제 해결 수단에 의하면, 특정 실행 압축 기법에 의존하지 않고 오리지널 엔트리 포인트를 찾을 수 있다는 장점이 있으며, 또한 실제 사용되는 메모리 영역을 선택적으로 분석하게 됨에 따라 높은 오리지널 엔트리 포인트 탐지 성공률을 얻을 수 있다.

도면의 간단한 설명

[0011] 도 1은 본 발명의 일실시예에 따른 엔트로피 분석을 이용한 실행 압축 해제 장치를 나타낸 블록도.

도 2는 본 발명의 일실시예에 따른 실행 압축 파일의 구조 변화를 담은 도면.

도 3은 본 발명의 일실시예에 따른 분기 명령어 검출 방법을 나타낸 동작 흐름도.

도 4는 본 발명의 일실시예에 따른 엔트로피 산출 방법을 나타낸 동작 흐름도.

도 5는 본 발명의 일실시예에 따른 실행 압축 해제 알고리즘을 담은 도면.

도 6은 본 발명의 일실시예에 따른 실행 압축 해제 패턴을 담은 도면.

발명을 실시하기 위한 구체적인 내용

[0012] 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. 그리고 도면에서 본 발명을 명확하게 설명하기 위해서 설명과 관계없는 부분은 생략하였으며, 명세서 전체를 통하여 유사한 부분에 대해서는 유사한 도면 부호를 붙였다.

[0013] 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 의미한다.

[0014] 도 1은 본 발명의 일실시예에 따른 실행 압축 해제 장치를 도시한 도면이다.

[0015] 도 1에 도시된 바와 같이, 실행 압축 해제 장치(100)는 실행 압축 해제부(110), 분기 명령어 검출부(120), 엔트로피 산출부(130), 저장부(140), OEP 판단부(150), 엔트로피 추이 그래프 생성부(160) 및 복원부(170)를 포함한다.

[0016] 실행 압축 해제부(110)에서는 실행 압축 해제 대상을 확정하고 대상 파일을 실행한다. 실행 압축 해제가 진행

됨에 따라 대상 파일은 내부적인 변화를 겪게 되는데 이를 이해하기 위해서는 먼저 실행 압축 파일의 구조를 파악할 필요가 있다.

[0017] 도 2는 본 발명의 일실시예에 따른 실행 압축 파일의 구조 변화를 담은 도면이다.

[0018] 도 2에 도시된 바와 같이, 초기상태의 경우 실행 압축 파일은 압축해제 모듈(210)과 압축된 코드(220)를 포함한다. 압축된 코드(220)는 원본 실행 파일이 실행 압축 프로그램에 의해 압축된 후 저장된 데이터 부분이고, 압축해제 모듈(210)은 실행 압축 프로그램에 의해 생성된 것으로 원본 실행 파일을 복원하는데 사용된다. 다시 말해, 실행 압축 프로그램을 실행시켜 원본 실행 파일을 압축한다면 압축된 코드(220)부분과 압축해제 모듈(210)부분이 생성되는 것이다. 대표적인 실행 압축 프로그램으로 UPX, ASPack, FSG, Telock, PECompact, WWPack32, EZip, Pex, JDPack, DoomPack, Mew 가 있다.

[0019] 실행 압축 해제는 압축해제 모듈(210)의 실행으로 시작된다. 실행 압축 해제가 진행됨에 따라 압축해제된 코드(230)가 메모리 영역에 쓰여진다. 이 때, 압축해제된 코드(230)는 압축된 코드(220)가 있는 메모리 영역과는 다른 영역에 쓰여지게 되는데, 이것을 컨트롤하는 부분이 압축해제 모듈(210)이다. 실행 흐름(Execution flow)이 압축해제 모듈(210)의 마지막 부분에 이른다면, 압축된 코드(220)로부터 압축해제된 코드(232)가 모두 메모리 영역에 쓰여지게 되고 이로써 실행 압축 해제 과정은 완료된다. 이 때, 압축해제 모듈(210)의 마지막 부분에 이른 실행 흐름은 압축된 코드(220)로 점프하는 것이 아니라 압축해제된 코드(232)의 가장 첫 부분으로 점프하게 되며, 이 지점이 오리지널 엔트리 포인트이다.

[0020] 다시 도 1을 참조하면, 분기 명령어 검출부(120)는 실행 압축 해제부(110)에 의해 실행 압축 과정이 진행되면서 실행되는 명령어 중 분기 명령어를 검출한다. 프로그램이 실행되면 순차적으로 주소값을 쓰거나 해당 주소값에 데이터를 저장하는 등 명령어 단위로 프로세스가 진행된다. 따라서 새로운 분기문을 작성하거나 실행문을 삽입할 때에도 명령어 단위로 하게 된다. 엔트로피 값도 타겟 프로세스에 대해 명령어 단위를 기준으로 측정 해야 하는데, 만약 명령어가 실행되는 매 순간 엔트로피 값을 측정한다면 비효율적일 것이다. 따라서 실행 명령어에 대한 선별장치가 필요하다.

[0021] 실행 명령어에 대한 선별기준을 설정함에 있어서 실행 압축 해제 과정을 상기할 필요가 있다. 실행 흐름(Excution flow)에 따라 오리지널 엔트리 포인트 주소가 결국 분기 명령문 이 후에 호출되므로 분기 명령어인지를 기준으로 선별하는 것이 바람직하다. 예를 들면 JMP 또는 CALL 등의 분기 명령어가 검출된 경우 엔트로피 값을 측정한다.

[0022] 분기 명령어 검출부(120)에서 검출한 분기 명령어들은 엔트로피 검출 시점을 결정하는 중요한 역할을 한다. 하지만 분기 명령어들은 반복되는 루프나 분기점에서 실행되는 경우가 많아 실행 압축 해제를 지연시키는 문제가 있다. 이를 해결하기 위해 캐시 메모리(cache memory)를 이용할 수 있다.

[0023] 하드디스크의 속도는 램보다 매우 느리다. 매번 프로그램을 실행시킬 때마다 디스크를 읽어야 하므로 속도가 느릴 수밖에 없다. 따라서 램과 디스크 사이에 일정량의 임시메모리를 만들고 처음 프로그램을 실행할 때 램으로 들어오는 내용을 그 임시메모리에도 보관한다. 그런 다음 프로그램을 실행시키면 하드디스크가 아닌 임시메모리에서 읽어오게 되기 때문에 읽어오는 시간이 매우 빨라지게 된다. 이 임시메모리를 캐시 메모리라고 한다.

[0024] 가장 최근에 실행된 n개의 분기 명령어를 캐시 메모리에 저장해 놓는다면 반복되는 호출 루프에서 실행속도가 빨라지게 되므로, 오리지널 엔트리 포인트 판단 속도도 빨라지게 된다.

[0025] 엔트로피 산출부(130)는 분기 명령어 검출부(120)에서 분기 명령어가 검출된 경우 실행 압축 해제된 데이터가 탑재된 메모리 영역의 엔트로피 값을 산출하는 장치이다. 분기 명령어가 검출된 시점이 엔트로피 값을 산출하는 시점이 되므로 분기 명령어 검출부(120)가 엔트로피 산출 시점을 결정하게 된다.

[0026] 엔트로피란 일반적으로 열역학적 계의 상태 함수 중의 하나로 통계적인 무질서도를 나타내지만 Shannon 은 “정보 엔트로피” 라는 개념을 통하여 정보의 양을 수치화하여 다음과 같은 수학식 1 로 정보 엔트로피 H 를 정리하였다.

[0027] (수학식 1)

[0028]
$$H(X) = \sum_{i=1}^{n} p(x_i) I(x_i) = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i)$$

[0029] $p(x_i)$는 x_i가 발생할 확률이고, I는 이상 확률 변수 X의 자기 정보량(Self-information) 을 의미한다. log 의 밑 b 값의 대표적인 일예로 2, 오일러 수 e, 10 가 있다. 일반적으로 정보 이론에서의 엔트로피는 메시지 압축에 관한 분야에 대해 연구할 때 많이 사용되며, 엔트로피가 높은 데이터일수록 나타날 수 있는 모든 비트들이 고루 존재함을 의미하므로 어떤 압축 파일의 엔트로피 수치가 높을수록 압축률이 높다. 예를 들어 '100100100111111' 란 문자열(string) 코드가 있다. 만약 이 문자열 코드가 3 비트 단위로 되어있다면, 연속된 문자열 개수와 그 문자열을 차례로 배열함으로써 압축할 수 있다. 예시된 문자열 코드를 압축하면 '011100010111' 이 된다. 3(011)개의 연속된 100 코드와 2(010)개의 연속된 111 코드로 이루어졌기 때문이다. 예시된 코드와 압축된 코드의 엔트로피를 구하기 위해 log 의 밑 b 값을 2라 하고 수학식 1에 따라 엔트로피를 구한다. 그 결과 압축된 코드 '011100010111'의 엔트로피 수치는 약 1.5 로, 압축되기 전 코드 '100100100111111' 의 엔트로피 수치 약 1.0306 보다 크다.

[0030] 만약 엔트로피 값 산출 대상을 전체 가상 메모리 영역으로 설정한다면 그 변화 정도가 미미하여 엔트로피의 변화 추이를 분석하기 어려울 수 있다.

[0031] 따라서 바람직하게는 특정 메모리 영역에 대해서 엔트로피 변화 추이를 분석하도록 설정할 수 있다. 예를 들면, 고정된 메모리 영역을 통해 엔트로피 변화 추이를 분석하도록 설정할 수 있다. 원본 코드가 쓰이는 위치는 실행 압축 프로세스의 첫 번째 섹션에 해당하기 때문에 실행 압축 프로세스의 첫 번째 섹션을 대상 메모리 영역으로 한정할 수 있다. 실행 압축 해제되면서 원본 코드가 새로 쓰이지 않는 부분은 엔트로피 값을 산출해도 변화가 없기 때문이다. 고정된 메모리 영역을 엔트로피 산출 대상으로 한정 한다면 측정 알고리즘을 작성할 때 간편하고 실행 루트도 간단하여 실행 시간이 적게 걸린다는 장점이 있다. 그러나 실행 압축 프로그램에 따라 실행 압축 프로세스의 첫 번째 섹션에 원본 코드를 쓰지 않는 경우도 있을 수 있고, 이 경우 오리지널 엔트리 포인트를 판단할 수 없거나 잘못 판단할 확률이 높다. 더욱이 빠르게 진화하는 악성코드 검출을 위해서는 상기와 같은 고정된 영역은 한계가 있다.

[0032] 다른 방법으로 실행 압축 해제된 데이터가 탑재되는 메모리 영역을 기초로 엔트로피 변화 추이를 분석하도록 설정할 수 있다. 이것은 고정된 영역이 아니라 가변적인 영역으로서, 실행 압축 해제 과정이 진행되면서 원본 코드를 쓸 때 압축 해제된 데이터가 탑재되는 메모리 영역을 동적으로 판단할 수 있다. 기계어 명령어 수준에서, 메모리에 읽고 쓰는 작업은 메모리의 주소 값을 레지스터에 저장하는 명령어와 저장된 주소 값으로 데이터를 옮기는 명령어의 조합으로 이루어진다. 따라서 상기 명령어의 조합을 이용한다면 데이터가 탑재되는 메모리 주소를 알 수 있다. 메모리의 주소 값을 레지스터에 저장하는 명령어의 일예로 LEA 명령어가 있으며, 저장된 주소 값으로 데이터를 옮기는 명령어의 일예로는 MOV 명령어가 있다.

[0033] 고정된 영역이 아닌 가변적 영역을 대상으로 엔트로피 분석을 하게 된다면 더미 데이터가 배제된 측정값을 얻을 수 있다. 또한 엔트로피 분석 영역이 명확히 한정됨에 따라 그 변화 추이도 보다 명확하게 되어 오리지널 엔트리 포인트 검출 확률이 높아진다. 실행 압축 프로그램에 상관없이 실행 압축 해제 과정을 거치면서 원본 데이터를 필연적으로 쓸 것이므로, 현존하는 모든 실행 압축 프로그램뿐 아니라 앞으로 개발될 압축 프로그램에 의한 실행 압축 파일에 대해서도 분석할 수 있다.

[0034] 저장부(140)는 산출된 엔트로피 값을 저장 매체에 저장한다. 분기 명령어가 검출될 때마다 엔트로피 값이 산출되는데, 이 값을 누적적으로 저장할 수 있다. 누적된 엔트로피 값은 이 후 OEP 판단부(150)에서 오리지널 엔트리 포인트를 판단하거나, 실행 압축 해제 과정이 완료되었는지 판단할 때 이용된다. 저장 매체의 대표적인 일예로, HDD, CD-ROM 드라이브가 있다.

[0035] OEP 판단부(150)는 산출된 엔트로피 값의 변화 추이를 이용하여 오리지널 엔트리 포인트를 판단한다. 저장부(140)에 저장된 엔트로피 값을 통해 분석한다.

[0036] 실행 압축된 파일이 일반 실행 파일에 비해 높은 엔트로피 값을 가지므로 압축 해제가 진행되는 초기 단계에는 실행 압축된 코드로 인해 해당 프로세스 메모리의 엔트로피 값이 커진다. 그러나 해제하는 과정을 거치면서 점차 메모리의 엔트로피 값이 일정하게 안정된다. 그러나 특정 영역에 대한 메모리 부분을 대상으로 엔트로피 분석을 수행할 경우, 압축 해제된 코드가 쓰여지는 부분에 엔트로피가 높은 다른 코드가 존재한다면 점차 엔트로피 값이 커지면서 일정하게 안정될 수도 있다.

[0037] 어떠한 경우든 실행 압축 해제 과정이 완료되면서 엔트로피가 일정하게 수렴된다. 산출된 엔트로피 값이 일정하게 수렴되기 시작하는 시점이 곧 압축 해제 과정이 완료되는 시점이며, 완료 이후 실행 흐름이 옮겨진 주소가 오리지널 엔트리 포인트가 된다.

[0038] 산출된 엔트로피 값을 미리 설정한 엔트로피 최소값(E_{min}) 및 최대값(E_{max})과 비교함으로써 실행 압축 해제가 완료되었는지 알 수 있다. 산출된 엔트로피 값이 미리 설정한 엔트로피 최소값과 최대값 사이라면 실행 압축 해제가 완료된 것으로 판단할 수 있다. 엔트로피 최소값과 최대값은, 여러 종류의 실행 압축 프로그램으로 압축한 실행 압축 파일을 실행 압축 해제함으로써 얻어진 다수의 수렴된 엔트로피 샘플 값으로부터 오차를 고려하여 정할 수 있다. 실행 압축 프로그램의 대표적인 예로 UPX, ASPack, FSG, Telock 가 있다.

[0039] 또한 일정하게 수렴된 엔트로피 값은 OEP 판단부(150)가 판단한 오리지널 엔트리 포인트가 참인지 알려준다. 만약 일정하게 수렴된 엔트로피 값이 미리 설정된 엔트로피 최소값과 최대값 사이라면, 실제 오리지널 엔트리 포인트를 찾은 것으로 간주할 수 있다.

[0040] 실행 압축 해제 과정은 자동 또는 수동으로 진행될 수 있다. 분기 명령어 검출, 엔트로피 산출 및 오리지널 엔트리 포인트 판단 모두 자동화되어, 프로세스의 중지나 외부 입력 없이 압축 해제 완료까지 진행될 수 있다. 그러나 사용자의 의도에 따라 실행 압축 해제 과정의 전체 또는 일부를 수동적으로 진행시킬 수도 있다.

[0041] 엔트로피 추이 그래프 생성부(160)는 분기 명령어 검출에 따른 엔트로피 값의 변화 추이를 도시한 엔트로피 추이 그래프를 생성한다.

[0042] 실행 압축 해제가 시작되면 명령어 단위로 호출된다. 분기 명령어 검출부(120)에서 호출되는 명령어 중 분기 명령어를 검출하면, 엔트로피 산출부(130)에서 특정 메모리 영역에 대해 엔트로피 값을 산출한다. 저장부(140)에서는 산출된 엔트로피 값을 저장매체에 누적적으로 저장한다. 엔트로피 추이 그래프 생성부(160)는 저장매체에 누적적으로 저장된 엔트로피 값을 이용하여 그래프를 생성한다.

[0043] 일실시예로, X축은 분기 명령어 검출 분기 명령어 검출 시점이 될 수 있고, Y축은 특정 메모리 영역에 대해 산출된 엔트로피 값이 될 수 있다. 다른 예로, X축을 분기 명령어 중 JMP 명령어를 검출한 시점으로 할 수 있다.

[0044] 그래프는 각각의 단위 정보를 연결하여 구조화시킨 것으로, 대표적인 예로 막대 그래프, 꺽은선 그래프, 띠그래프 등이 있다. 상기 그래프는 저장매체에 저장된 분기 명령어 검출 시점 정보와 엔트로피 값 정보를 연결하여 구조화시킨 것이라면 어떤 형태이든 무방하다.

[0045] 그래프를 생성하는 엔트로피 추이 그래프 생성부(160)는 실행 압축 해제 과정의 자동화 및 편의를 위해 실행 압축 해제 장치(100) 내에 포함되는 것이 바람직하다. 그러나, 사용자의 의도에 따라 그래프를 생성할 수 있는 장치를 외부에 연결하거나 수동적으로 이용할 수도 있다.

[0046] 복원부(170)는 실행 압축된 원본 실행 파일을 복원한다.

[0047] 실행 파일은 특정한 형식에 따라 구성된다. 예를 들어, WINDOWS 의 경우 PE 형식이고, LINUX 의 경우 ELF 형식이다. 특정한 형식에 따라 구성된 실행 파일을 실행시키면 운영 체제는 실행 파일에 저장되어 있는 정보를 바탕으로 코드, 데이터 등을 메모리 상에 탑재한다. 그러므로 메모리 또는 레지스터를 분석한다면 특정 형식에 맞는 실행 파일을 재구성할 수 있다.

[0048] 메모리에 탑재된 프로그램은 엔트리 포인트 주소 값에 있는 명령어부터 수행한다. 실행 압축 파일의 경우 먼저 압축된 원본 프로그램을 메모리 상에 해제한다. 이때 실행 압축 해제가 완료되는 시점을 알 수 있으므로 원본 프로그램이 시작되는 순간, 메모리와 레지스터에 어떤 값들이 저장되어 있는지 알 수 있다. 따라서 메모리 또는 레지스터에 저장된 값을 분석하여 실행 파일을 복원할 수 있다. 실행 파일을 복원한다면 본 발명의 기술분야인 악성코드 분석을 효과적으로 달성할 수 있으며, 실행 압축 기법에 관계없이 복원할 수 있어 바람직하다.

[0049] 원본 파일의 복원은 자동 또는 수동으로 진행될 수 있다. 실행 압축 해제 과정에 이어 원본 파일의 복원 과정도 자동화되어, 프로세스의 중지나 외부 입력 없이 복원될 수 있다. 그러나 사용자의 의도에 따라 실행 압축 해제 과정이 끝나고 프로세스를 중지 시키거나 또는 수동적 입력에 따라 복원 과정을 진행시킬 수도 있다.

[0050] 도 3은 본 발명의 일실시예에 따른 분기 명령어 검출 방법을 나타낸 동작 흐름도이다.

[0051] 먼저, 명령어를 검사한다(S310). 압축 해제 프로세스가 진행되면 명령어 단위로 호출되므로 호출된 명령어들을 감지하여 검사한다.

[0052] 다음으로, 해당 명령어가 분기 명령어인지 체크한다(S320). 그 결과 만약 분기 명령어가 아니라면 해당 명령어를 제외하고(S330), 다음 명령어 검사를 수행한다(S310). 분기 명령어의 대표적인 예로 JMP 또는 CALL 이 있다.

[0053] 다음으로, API(Application Program Interface)의 분기 명령어 인지 체크한다(S340). 만약 검출된 분기 명령어가 API로부터 호출된 것이라면 해당 명령어를 제외하고(S330), 다음 명령어 검사를 수행한다(S310). 오리지널 엔트리 포인트는 실행 압축 프로그램에 의해 정해지고 실행 압축 해제 과정이 완료된 이후 실행 흐름이 옮겨진 주소이므로, API로부터 호출된 분기 명령어들은 제외시키는 것이 바람직하기 때문이다. API로부터 호출된 분기 명령어들을 제외시키는 단계를 포함함으로써, 오리지널 엔트리 포인트 검출 속도가 빨라진다.

[0054] 다음으로 엔트로피를 산출한다(S400). 앞서 설명한 바와 같이 미리 설정된 기준에 따라 한정한 메모리 영역에 대하여 수학식 1을 이용해 엔트로피를 산출할 수 있다. 엔트로피 산출과정의 상세 내용은 도면을 통해 살펴보기로 한다.

[0055] 도 4는 본 발명의 일실시예에 따른 엔트로피 산출 방법을 나타낸 동작 흐름도이다.

[0056] 먼저, 분기 명령어 검출 단계를 수행한다(S300). 이 과정에서 API로부터 호출된 분기 명령어는 제외된다.

[0057] 다음으로 프로세스를 중지한다(S410). 프로세스가 중단됨으로써 새로운 분기 명령어 검출을 방지할 수 있다.

[0058] 다음으로 메모리를 덤프한다(S420). 메모리 덤프란 메모리에 있는 데이터를 받아오는 것이다. 이 과정은 다음 단계인 더미 데이터인지 여부나 엔트로피 값 검출을 위해 필요하다.

[0059] 다음으로 더미 데이터인지 판단한다(S430). 만약 더미 데이터라면 해당 메모리를 엔트로피 값 산출 대상에서 제외시키고(S440), 더미 데이터가 아니라면 엔트로피 값 산출 단계로 넘어간다(S450).

[0060] 더미 데이터란 무의미한 데이터 또는 무의미한 데이터의 연속으로, 연속된 0x00 값이 나타나는 경우가 대표적이다. 덤프한 메모리 값이 더미 데이터라면 상기 메모리에 대한 엔트로피 값을 산출하는 것은 무의미하다. 엔트로피 산출 대상이 잘못 지정되었기 때문이다. 실행 압축 해제된 데이터가 탑재되는 메모리 영역에 한 해 엔트로피 값을 산출한다면 잘못된 메모리를 선택할 경우가 거의 없을 것이나, 더미 데이터 판단 단계는 예상치 못한 경우에 대비하기 위한 것으로, 선택에 따라 수행하지 않을 수 있다.

[0061] 다음으로 엔트로피 값을 산출한다(S450). 엔트로피 값은 상기 수학식 1에 따라 산출한다.

[0062] 다음으로, 저장 매체에 저장한다(S460). 저장 매체의 대표적인 일예로, HDD, CD-ROM 드라이브가 있다.

[0063] 다음으로, OEP 판단단계를 수행한다(S500). 저장 매체에 저장된 엔트로피 값의 변화 추이, 또는 미리 설정한 범위에 해당하는지를 판단하여 오리지널 엔트리 포인트를 결정한다.

[0064] 도 5는 본 발명의 일실시예에 따른 실행 압축 해제 알고리즘을 담은 도면이다.

[0065] 먼저 초기화가 필요하다. 만약 고정된 메모리 영역에 대해 엔트로피 변화 추이를 분석하도록 설정한다면, 고정된 메모리 영역에 대한 설정이 필요하다.

[0066] 다음으로, 엔트로피 산출하는 시점을 결정한다. 일예로 JMP 명령어라면 해당 메모리의 엔트로피 값을 산출하도록 한다.

[0067] 다음으로, 실행 압축 해제가 완료되었는지 검사한다. 산출된 엔트로피 값을 미리 설정한 엔트로피 최소값(E_{min}) 및 최대값(E_{max})과 비교함으로써 실행 압축 해제가 완료되었는지 알 수 있다.

[0068] 실행 압축 해제가 완료되면 엔트로피 값은 일정하게 유지되는데, 이 값은 실행 압축 프로그램의 종류와 실행 압축 파일의 특징에 따라 조금씩 달라진다. 그러나 이 값들은 일정한 범위 내에 분포하며, 다수의 샘플로부터 오차를 고려하여 엔트로피 최소값과 최대값을 정할 수 있다.

[0069] 정해진 최소값과 최대값은 실행 압축 해제가 완료되었는지 알려줄 뿐만 아니라, OEP 판단부(150)에서 판단한 오리지널 엔트리 포인트가 참인지 알려주는 플래그(flag)로 기능할 수 있다. 만약 엔트로피 산출부(130)에서 산출된 엔트로피 값이 미리 설정된 엔트로피 최소값보다 작거나 미리 설정된 엔트로피 최대값보다 크다면, 산출된 엔트로피 값을 이용하여 판단된 오리지널 엔트리 포인트는 실제 오리지널 엔트리 포인트가 아닐 확률이 높으므로 제외시킨다. 만약 엔트로피 산출부(130)에서 산출된 엔트로피 값이 미리 설정된 엔트로피 최소값과 최대값 사이라면, 실제 오리지널 엔트리 포인트를 찾은 것으로 간주되고 동시에 실행 압축 해제가 완료되었음을 의미한다. 이 단계를 통해 오리지널 엔트리 포인트 검출 확률을 높이고, 실행 압축 해제 장치가 실제 오리지널 엔트리 포인트를 검출했는지 확인할 수 있다.

[0070] 도 6은 본 발명의 일실시예에 따른 실행 압축 해제 패턴을 담은 도면이다.

[0071] 실행 압축 프로그램의 종류에 따라 실행 압축 해제는 각각 다른 패턴을 갖는다. 만약 고정된 메모리 영역에 대해 엔트로피 변화 추이를 분석하도록 설정한다면, 고정된 메모리 영역에 실행 압축 해제된 코드를 쓰는 패턴이 실행 압축 프로그램에 따라 다를 수 있으므로 엔트로피 변화 추이도 달라질 수 있다.

[0072] 도 6은 6개의 그래프를 도시한다. 각 그래프마다 실행 압축 프로그램의 종류가 다르다. 실행 압축 프로그램으로 nSpack, upxn, RLPack, nPack, mpress, aspack 이 사용되었다. 각각의 실행 압축 프로그램은 서로 크기가 다른 6개의 실행 파일에 대해 실행 압축 하였다. 실행 파일로 Calc, freecell, mshearts, msiexec, notepad, telnet 가 사용되었다. X축은 분기 명령어 중 JMP 명령어를 나타내고, Y축은 고정된 메모리 영역에 대해 산출된 엔트로피 값을 나타낸다.

[0073] 예를 들어, 첫 번째 그래프(왼쪽 상단)에서 실선은 nSpack 실행 압축 프로그램으로 압축한 calc.exe 파일의 실행 압축 해제에 관한 것이다. 프로세스가 진행되어 JMP 명령어를 검출할 때마다, 고정된 영역에 대해 엔트로피 값을 산출한다. 초기에 엔트로피 값이 급격하게 증가하다가, JMP 명령어가 약 50000번 검출된 시점부터 증가폭이 둔화되기 시작하고, 약 200000번 검출된 시점부터는 엔트로피 값이 거의 변화하지 않고 안정되기 시작한 것을 알 수 있다.

[0074] 실행 압축 프로그램에 따라 엔트로피 변화 추이가 구별된다. TYPE Ⅰ-ⅰ 는 엔트로피 값이 지속적으로 증가하나, TYPE Ⅰ-ⅱ 는 초기에 엔트로피 값이 급격히 감소된 후 증가한다. TYPE Ⅱ-ⅰ 는 초기에 지속적으로 감소하다가, 이 후 안정된다. TYPE Ⅱ-ⅱ 는 초기에 소폭으로 감소하다가, 이 후 급격하게 감소한다. TYPE Ⅱ-ⅲ 는 엔트로피 값이 일정하게 유지되다가 특정 시점에 순식간에 감소된 후 다시 일정하게 유지된다.

[0075] 실행 압축 프로그램에 따라 엔트로피 값의 변화 추이는 일정한 패턴을 가질 수 있다. 따라서 임의의 실행 압축 파일에 대해 실행 압축 해제를 하더라도, 엔트로피 변화 추이를 분석한다면 어떤 실행 압축 프로그램으로 압축한 것인지 판단할 수 있다.

[0076] 그래프 분석을 통해, 엔트로피 값이 수렴한 시점을 찾을 수 있다. 엔트로피 변화 추이는 실행 압축 프로그램에 따라 다르지만, 엔트로피 값이 수렴한 시점은 모두 유일하다. 이 시점이 실행 압축 해제 완료 시점이 된다. 따라서 임의의 실행 압축 파일에 대해 실행 압축 해제를 하더라도, 엔트로피 값이 수렴한 시점을 찾는다면 실행 압축 해제 완료 시점을 판단할 수 있다.

[0077] 참고로, 본 발명의 실시예에 따른 도 1에 도시된 구성 요소들은 소프트웨어 또는 FPGA(Field Programmable Gate Array) 또는 ASIC(Application Specific Integrated Circuit)와 같은 하드웨어 구성 요소를 의미하며, 소정의 역할들을 수행한다.

[0078] 그렇지만 '구성 요소들'은 소프트웨어 또는 하드웨어에 한정되는 의미는 아니며, 각 구성 요소는 어드레싱할 수 있는 저장 매체에 있도록 구성될 수도 있고 하나 또는 그 이상의 프로세서들을 재생시키도록 구성될 수도 있다.

[0079] 따라서, 일 예로서 구성 요소는 소프트웨어 구성 요소들, 객체지향 소프트웨어 구성 요소들, 클래스 구성 요소들 및 태스크 구성 요소들과 같은 구성 요소들과, 프로세스들, 함수들, 속성들, 프로시저들, 서브루틴들, 프로그램 코드의 세그먼트들, 드라이버들, 펌웨어, 마이크로 코드, 회로, 데이터, 데이터베이스, 데이터 구조들, 테이블들, 어레이들 및 변수들을 포함한다.

[0080] 구성 요소들과 해당 구성 요소들 안에서 제공되는 기능은 더 작은 수의 구성 요소들로 결합되거나 추가적인 구성 요소들로 더 분리될 수 있다.

[0081] 전술한 본 발명의 설명은 예시를 위한 것이며, 본 발명이 속하는 기술분야의 통상의 지식을 가진 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.

[0082] 본 발명의 범위는 상기 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

[0083] 100: 실행 압축 해제 장치

110: 실행 압축 해제부

120: 분기 명령어 검출부

130: 엔트로피 산출부

140: 저장부

150: OEP 판단부

160: 엔트로피 추이 그래프 생성부

170: 복원부

210: 압축해제 모듈

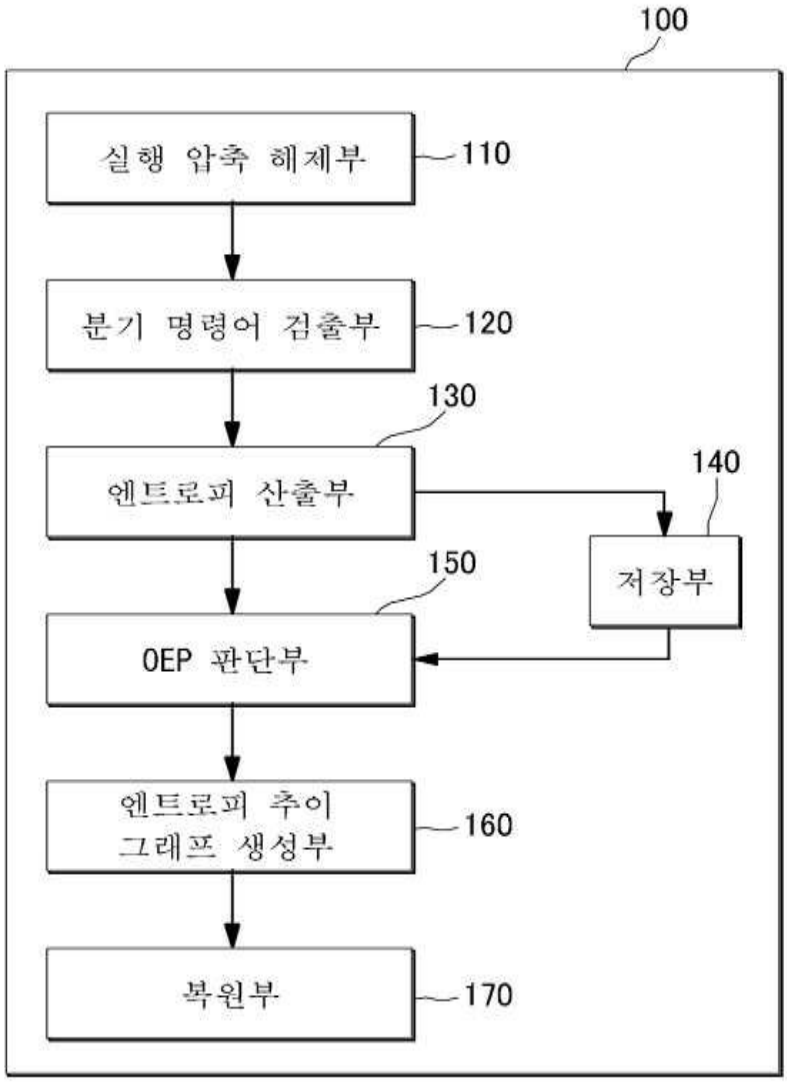
220: 압축된 코드

230: 압축해제된 코드(진행 상태)

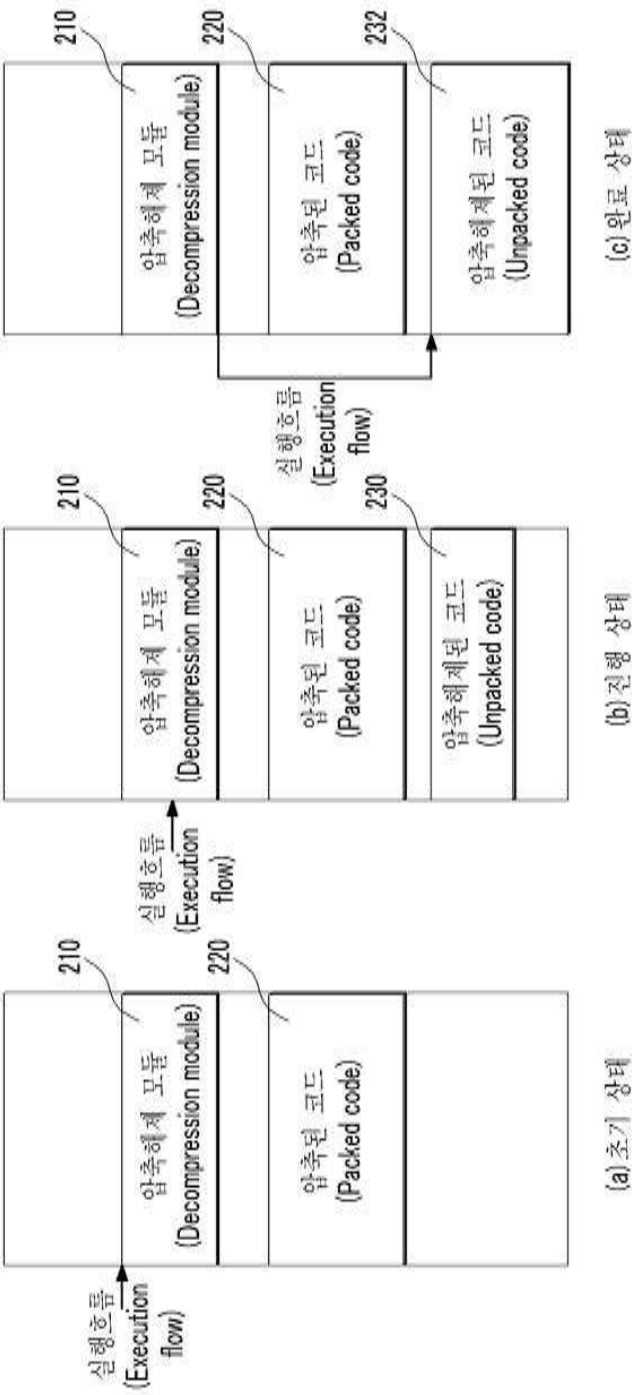
232: 압축해제된 코드(완료 상태)

도면

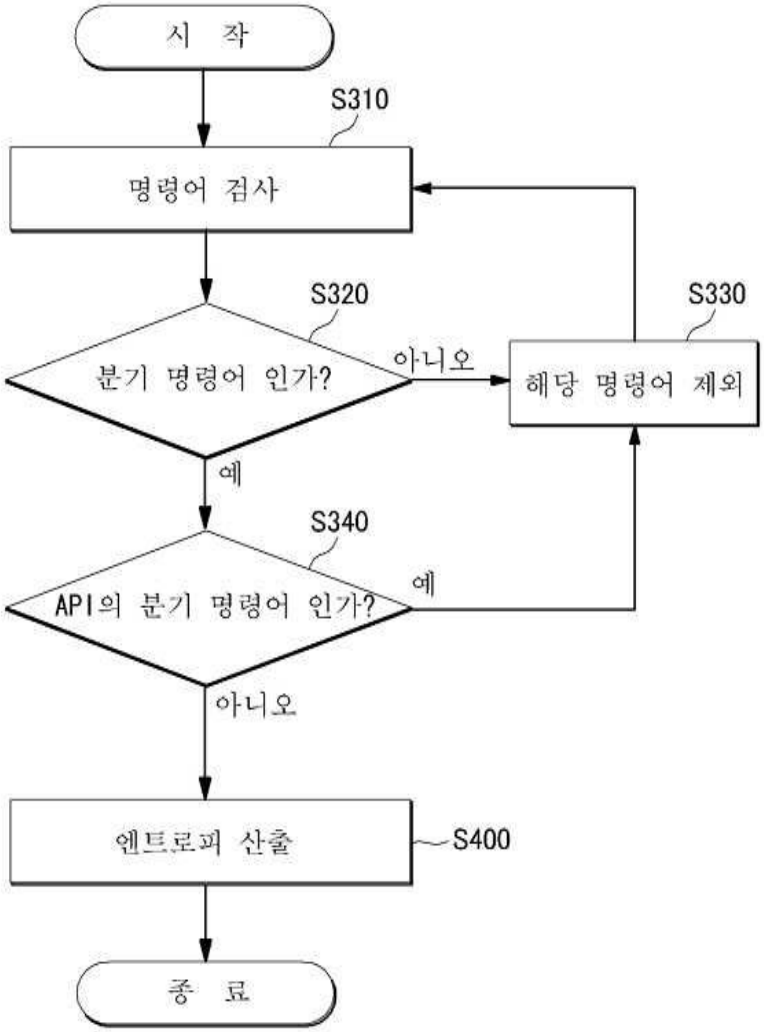
도면1



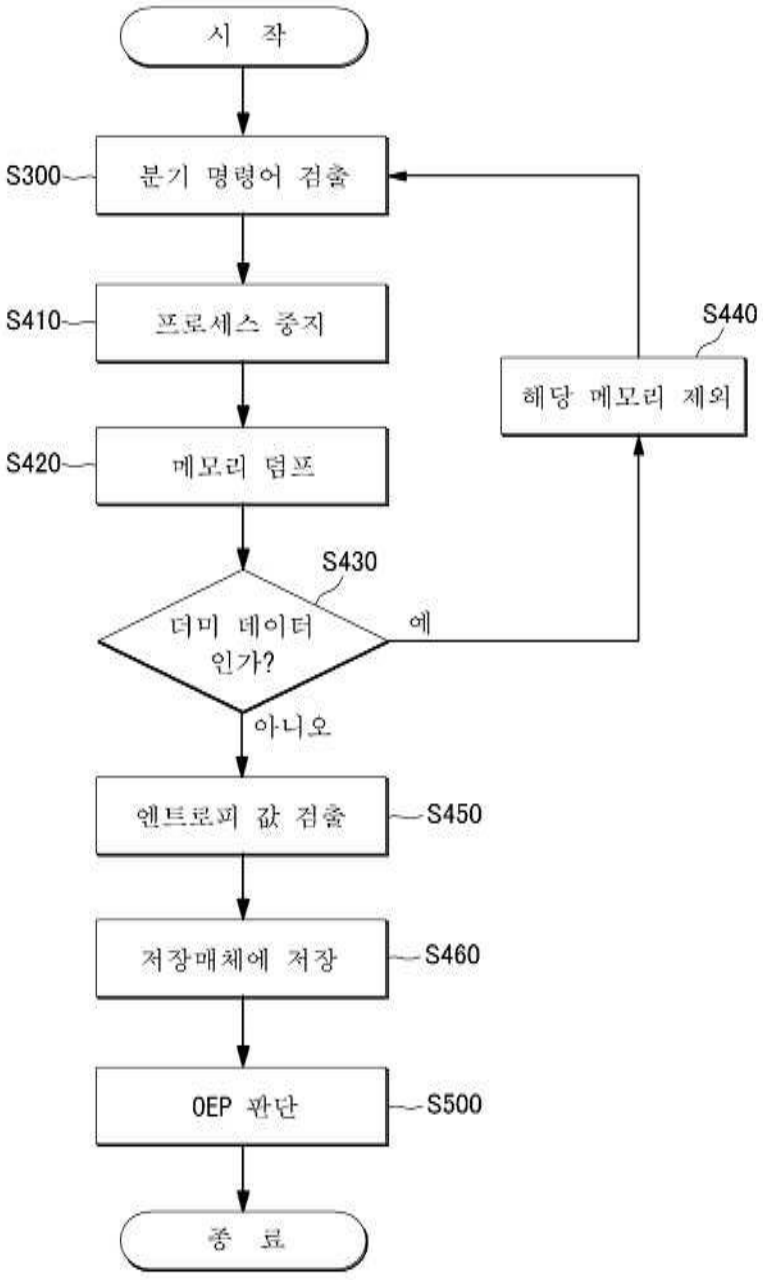
도면2


210
220
232
압축해제 모듈
(Decompression module)
압축된 코드
(Packed code)
압축해제된 코드
(Unpacked code)
(c) 완료 상태
실행흐름
(Execution flow)
210
220
230
압축해제 모듈
(Decompression module)
압축된 코드
(Packed code)
압축해제된 코드
(Unpacked code)
(b) 진행 상태
실행흐름
(Execution flow)
210
220
압축해제 모듈
(Decompression module)
압축된 코드
(Packed code)
(a) 초기 상태
실행흐름
(Execution flow)

도면3


시 작
S310
명령어 검사
S320
분기 명령어 인가?
아니오
S330
해당 명령어 제외
예
S340
API의 분기 명령어 인가?
예
아니오
엔트로피 산출
S400
종 료

도면4


시 작
S300
분기 명령어 검출
S410
프로세스 중지
S420
메모리 덤프
S430
더미 데이터 인가?
예
S440
해당 메모리 제외
아니오
엔트로피 값 검출
S450
저장매체에 저장
S460
OEP 판단
S500
종 료

도면5

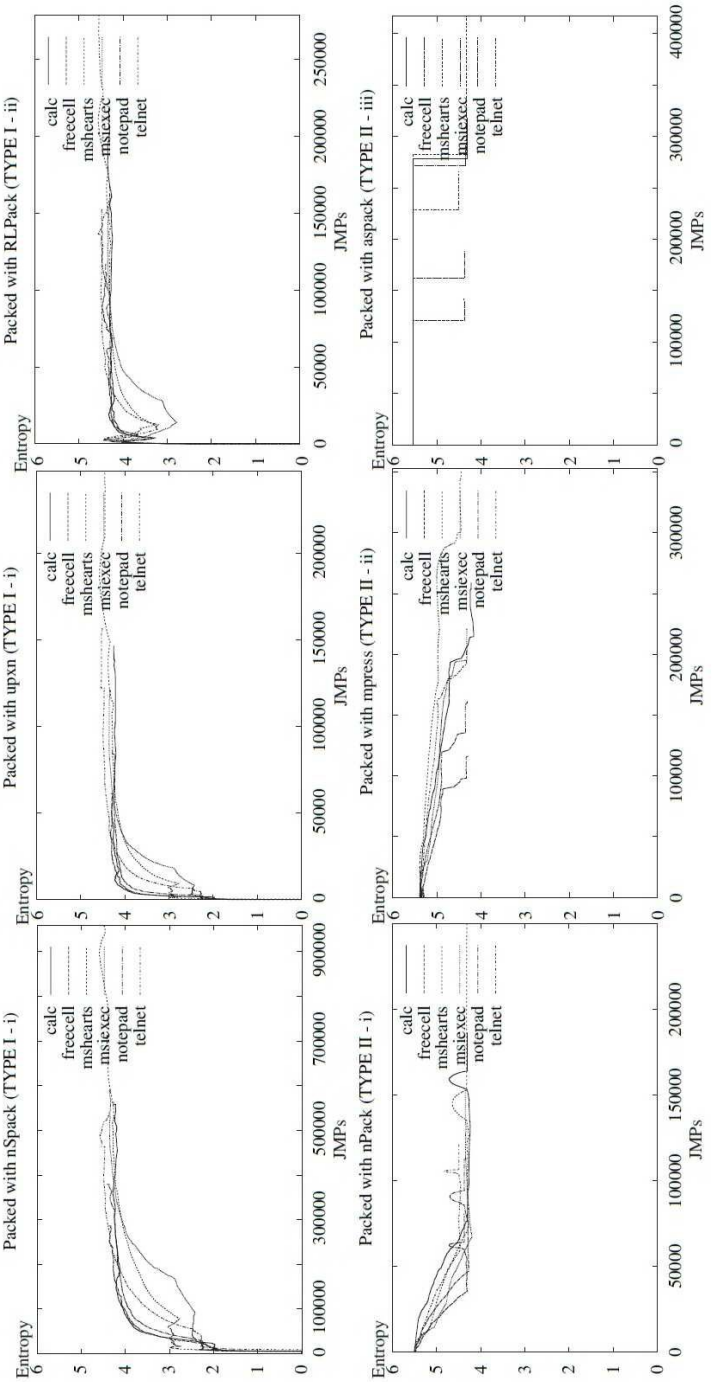
Algorithm 1 An algorithm of our approach

Require: A packed executable to be analysed is required for this algorithm. The packed executable, an instruction pointer, and entropy of unpacked code are written as *P*, *IP* and *E*, respectively, in the following pseudocode. We assume that an executable is categorized as either packed or native.

Ensure: Locate OEP of *P*.

```
/* Initialization */
Find an entry point and the first section of P.
Set a break point to the entry point.
Set R to the range of the first section.
/* Start analysis */
while a PROCESS is not terminated do
    IP ← a current instruction pointer
    /* Measure entropy in only this condition */
    if IP is a JMP instructions then
        Measure entropy of R.
    else
        Continue this loop.
    end if
    /* Check if unpacking is complete */
    if E_min ≤ Measured entropy ≤ E_max then
        /* Check if it jumps onto the unpacked code */
        if Jump into R from outside of R is true then
            OEP ← The next instruction address
            Break this loop.
        else
            Continue this loop.
        end if
        Continue this loop.
    end if
end while
```

도면6


Packed with nSpack (TYPE I - i)
Packed with upxn (TYPE I - i)
Packed with RLPack (TYPE I - ii)
Packed with nPack (TYPE II - i)
Packed with mpress (TYPE II - ii)
Packed with aspack (TYPE II - iii)
Entropy
JMPs
calc
freecell
mshearts
msiexec
notepad
telnet