(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2022-0043023
(43) 공개일자 2022년04월05일

(51) 국제특허분류(Int. Cl.)
G06F 21/55 (2013.01) *G06F 21/57* (2013.01)
G06N 3/08 (2006.01)
(52) CPC특허분류
G06F 21/55 (2013.01)
G06F 21/577 (2013.01)
(21) 출원번호 10-2021-0111132
(22) 출원일자 2021년08월23일
심사청구일자 2021년08월23일
(30) 우선권주장
1020200126036 2020년09월28일 대한민국(KR)

(71) 출원인
고려대학교 산학협력단
서울특별시 성북구 안암로 145, 고려대학교 (안암동5가)
(72) 발명자
정호용
경기도 고양시 덕양구 동산2로 18, 2101동 1602호
류도현
서울특별시 은평구 연서로34길 10-7, 101호
허준범
경기도 용인시 기흥구 보정로 30, 119동 201호(보정동,동아솔레시티아파트)
(74) 대리인
김홍석

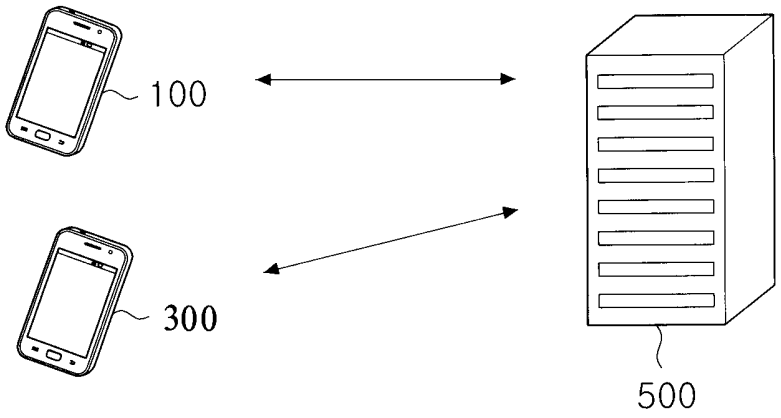
전체 청구항 수 : 총 6 항

(54) 발명의 명칭 **멜트다운 취약점을 이용한 인공신경망 추출 방법**

(57) 요 약

인공신경망 추출 방법이 개시된다. 상기 인공신경망 추출 방법은, MLaaS(Machine-Learning-as-a-Service)를 제공하는 서버와 통신가능하고 적어도 프로세서를 포함하는 컴퓨팅 장치에 의해 수행되는 인공신경망 추출 방법으로써, 공격 대상 프로세스의 페이지 테이블을 획득하는 단계, 상기 페이지 테이블에 기초하여 상기 공격 대상 프로세스의 힙 영역 데이터를 획득하는 단계, 상기 힙 영역 데이터에 기초하여 상기 공격 대상 프로세스의 인공신경망 인스턴스를 획득하는 단계, 및 상기 인공신경망 인스턴스에 기초하여 인공신경망 모델의 구조를 추출하는 단계를 포함한다.

대 표 도 - 도14



10

(52) CPC특허분류
G06N 3/08 (2013.01)

이 발명을 지원한 국가연구개발사업
과제고유번호 1711102924
과제번호 2019-0-00533-002
부처명 과학기술정보통신부
과제관리(전문)기관명 정보통신기획평가원
연구사업명 정보보호핵심원천기술개발(R&D)
연구과제명 컴퓨터 프로세서의 구조적 보안 취약점 검증 및 공격 탐지대응
기 여 율 1/2
과제수행기관명 고려대학교산학협력단
연구기간 2020.01.01 ~ 2020.12.31
이 발명을 지원한 국가연구개발사업
과제고유번호 1711120246
과제번호 2020-0-01819-001
부처명 과학기술정보통신부
과제관리(전문)기관명 정보통신기획평가원
연구사업명 정보통신방송혁신인재양성(R&D)
연구과제명 ICT명품인재양성(고려대학교)
기 여 율 1/2
과제수행기관명 고려대학교산학협력단
연구기간 2020.07.01 ~ 2020.12.31

명 세 서

청구범위

청구항 1

MLaaS(Machine-Learning-as-a-Service)를 제공하는 서버와 통신가능하고, 적어도 프로세서를 포함하는 컴퓨팅 장치에 의해 수행되는 인공신경망 추출 방법에 있어서,

공격 대상 프로세스의 페이지 테이블을 획득하는 단계;

상기 페이지 테이블에 기초하여 상기 공격 대상 프로세스의 힙 영역 데이터를 획득하는 단계;

상기 힙 영역 데이터에 기초하여 상기 공격 대상 프로세스의 인공신경망 인스턴스를 획득하는 단계; 및

상기 인공신경망 인스턴스에 기초하여 인공신경망 모델의 구조를 추출하는 단계를 포함하는 인공신경망 추출 방법.

청구항 2

제1항에 있어서,

상기 페이지 테이블을 획득하는 단계는,

상기 서버의 메모리 내에서 프로세스 정보가 저장된 연결리스트의 헤드 위치에 대해 멜트다운 공격(Meltdown attack)을 수행하여 상기 연결리스트를 추출하는 단계;

상기 연결리스트에 포함된 다음 노드에 대한 포인터에 기초하여 상기 다음 노드에 대한 멜트다운 공격을 적어도 일회 수행함으로써 상기 공격 대상 프로세스의 노드를 획득하는 단계; 및

상기 공격 대상 프로세스의 노드 내에 포함된 상기 페이지 테이블의 위치에 대해 멜트다운 공격을 수행하여 상기 페이지 테이블을 획득하는 단계를 포함하는,

인공신경망 추출 방법.

청구항 3

제1항에 있어서,

상기 힙 영역 데이터를 획득하는 단계는,

상기 페이지 테이블을 이용하여 상기 공격 대상 프로세스가 매핑된 물리주소(physical address)를 획득하는 단계; 및

상기 물리주소에 대해 멜트다운 공격을 수행하여 상기 힙 영역 데이터를 획득하는 단계를 포함하는,

인공신경망 추출 방법.

청구항 4

제1항에 있어서,

상기 인공신경망 인스턴스를 추출하는 단계는,

상기 힙 영역 데이터로부터 딕셔너리 인스턴스를 획득하는 단계;

상기 딕셔너리 인스턴스로부터 전역 심볼테이블(global symbol table)을 획득하는 단계; 및

상기 전역 심볼테이블에 포함된 상기 인공신경망 인스턴스의 주소에 대해 멜트다운 공격을 수행하여 상기 인공신경망 인스턴스를 추출하는 단계를 포함하는,

인공신경망 추출 방법.

청구항 5

제4항에 있어서,

상기 인공신경망 모델의 구조를 추출하는 단계는,

상기 인공신경망 인스턴스로부터 인스턴스 심볼테이블을 획득하는 단계;

상기 인스턴스 심볼테이블에 포함된 상기 인공신경망의 레이어 정보에 기초하여 상기 인공신경망의 구조를 추출하는 단계를 포함하는,

인공신경망 추출 방법.

청구항 6

제5항에 있어서,

상기 인공신경망 모델의 가중치를 추출하는 단계를 더 포함하고,

상기 가중치를 추출하는 단계는,

상기 인스턴스 심볼테이블에 포함된 각 은닉층 인스턴스의 포인터를 획득하는 단계;

각 은닉층 인스턴스의 심볼테이블을 획득하는 단계; 및

각 은닉층 인스턴스의 심볼테이블로부터 각 은닉층의 가중치를 획득하는 단계를 포함하는,

인공신경망 추출 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 멀티-테넌시(multi-tenancy) 환경에서 멜트다운 취약점을 이용하여 딥러닝 모델의 내부 정보를 추출할 수 있는 방법 및 장치에 관한 것이다.

배 경 기 술

[0002] 딥러닝 기술의 발달에 따라 인공지능 기술은 실생활의 여러 분야에서 활발히 사용되고 있으며 점점 그 중요성이 증가하고 있다. 특히 기업의 경우 경제적, 서비스적 개선을 위해 앞다투어 인공지능 솔루션을 도입하는 추세이며 이 과정에서 클라우드 서비스를 이용하는 경우가 많다. 인공지능 개발 및 배포를 위해 직접 인프라를 구축하기보다, 빠르고 안정적이며 비교적 저렴하게 인공지능 솔루션을 도입할 수 있는 클라우드 서비스에 대한 관심이 급증하고 있다. 이러한 서비스는 대부분 멀티-테넌시 환경에서 동작한다. 클라우드 서비스는 가상화 기술을 통해 각 사용자에게 논리적으로 독립된 컴퓨팅 공간을 마련하여 멀티-테넌시 환경을 제공한다.

[0003] 클라우드 의존도의 증가에 따라 새로운 보안 위협과 도전 과제들이 생겨났다. 방화벽과 경계 보호가 주를 이루는 로컬 환경과는 달리, 클라우드에는 여러 사용자의 데이터가 동일 컴퓨터 상에 상주할 뿐만 아니라 연산까지 동일 컴퓨터 상에서 이루어지기도 한다. 이는 클라우드의 멀티-테넌시 구조에서 기인하고, 이러한 멀티-테넌시 구조를 악용하여 다른 사용자의 데이터에 접근이 가능할 수 있다.

[0004] 클라우드의 딥러닝 모델 내부 구조가 공격에 의해 유출되는 경우, 지적재산권 침해로 인해 경제적 손실뿐 아니라, 유출된 모델을 통해 적대적 공격(Adversarial attack), 학습 데이터 추출 공격(Inversion Attack) 등 추가적인 피해를 야기할 수 있기 때문에 기계학습 모델을 보호하는 것은 클라우드 환경에서 매우 중요한 문제이다. 이러한 이유로 딥러닝 모델 추출 공격에 대한 연구가 활발하게 이루어지고 있다. 본 발명에서는 기존 연구의 한계점을 극복하고 현실적인 공격이 가능한 수준의 기법을 제안한다. 제안하는 기법의 중요성은 다음과 같다.

[0005] 1. 멜트다운을 이용하여 멀티-테넌시 환경에서 동작하는 딥러닝 서비스의 내부 정보를 추출하는 공격 시나리오를 제시한다. 기존이 딥러닝 복원 공격들이 인공신경망 구조를 유추하는데 그치는 것에 비해 제안하는 공격은 해당 딥러닝 프로세스의 메모리 영역에 접근해 인공신경망 관련 정보를 전부 추출할 수 있다.

[0006] 2. 멜트다운과 멜트다운 변종 공격들은 데이터를 선별하여 추출할 수 없고 추출된 데이터를 후처리를 통해 파싱하기도 어렵다는 한계점이 존재한다. 본 발명에서는 이를 해결하기 위해 메모리 내에서 목표 데이터가 매핑된

위치를 파악하고 이 위치에 선별적으로 추출 공격을 수행하는 기법을 제시한다.

[0007] 3. 현존하는 멜트다운 변종 및 부채널 공격 연구는 공격자가 데이터의 위치를 이미 알고 있다는 가정 하에 정해진 위치에 대한 공격을 시연한다. 하지만 TensorFlow나 PyTorch 등 많은 딥러닝 라이브러리는 메모리가 런타임에 동적으로 할당되기 때문에 사전에 목표 데이터 위치를 특정하는 것이 불가능하다. 본 발명에서는 동적으로 할당되는 변수의 위치를 추적하는 방법을 통해 기존 연구에서의 실험환경 제약을 해결한다.

선행기술문헌

비특허문헌

[0008] (비특허문헌 0001) Lipp, M., et al. "Meltdown: Reading kernel memory from user space," 27th USENIX Security Symposium, pp. 973-990, Aug. 2018.

(비특허문헌 0002) Yarom, Y. and Falkner, K. "FLUSH+RELOAD: a high resolution, low noise, L3 cache side-channel attack," 23rd USENIX Security Symposium, pp. 719-732, 2014.

발명의 내용

해결하려는 과제

[0009] 본 발명이 이루고자 하는 기술적인 과제는 멜트다운 취약점을 이용하여 인공신경망을 추출할 수 있는 방법 및 장치를 제공하는 것이다.

과제의 해결 수단

[0010] 본 발명의 일 실시예에 따른 인공신경망 추출 방법은, MLaaS(Machine-Learning-as-a-Service)를 제공하는 서버와 통신가능하고 적어도 프로세서를 포함하는 컴퓨팅 장치에 의해 수행되는 인공신경망 추출 방법으로써, 공격 대상 프로세스의 페이지 테이블을 획득하는 단계, 상기 페이지 테이블에 기초하여 상기 공격 대상 프로세스의 힙 영역 데이터를 획득하는 단계, 상기 힙 영역 데이터에 기초하여 상기 공격 대상 프로세스의 인공신경망 인스턴스를 획득하는 단계, 및 상기 인공신경망 인스턴스에 기초하여 인공신경망 모델의 구조를 추출하는 단계를 포함한다.

발명의 효과

[0011] 본 발명의 실시예에 따른 인공신경망 추출 방법에 의할 경우, 원격 서버에서 동작하는 딥러닝 서비스에 대한 추출 방법을 제공할 수 있다.

[0012] 또한, 본 발명에 의할 경우, 원격 부채널을 통해 목표 딥러닝 프로세스 메모리에 직접 접근하여 인공신경망의 모든 데이터를 추출할 수 있다.

도면의 간단한 설명

[0013] 본 발명의 상세한 설명에서 인용되는 도면을 보다 충분히 이해하기 위하여 각 도면의 상세한 설명이 제공된다.

도 1은 멜트다운 공격의 핵심인 x86 어셈블리 코드이다.

도 2는 *PyDictEntry*에 저장된 데이터의 모습니다.

도 3은 Python 3.5.2 빌드에서의 각 자료형 클래스가 존재하는 메모리 위치를 나타내는 표를 도시한다.

도 4는 전역 심볼테이블(global symbol table)을 도시한다.

도 5는 모델 인스턴스의 심볼테이블을 도시한다.

도 6은 은닉층 인스턴스의 심볼테이블을 도시한다.

도 7은 *tf.Variable*의 구조를 도시한다.

도 8은 추출 대상 모델의 내부 정보를 도시한다.

도 9는 모델 인스턴스 심볼데이터의 '*network_nodes*' 요소에 대한 멜트다운 공격으로 배열의 각 아이템을 추출한 결과이다.

도 10은 인공신경망의 가중치 정보를 성공적으로 추출한 결과를 도시한다.

도 11은 IEEE 754 싱글 프리시전 디코딩(single precision decoding)을 도시한다.

도 12는 VGG16 이미지 인식 딥러닝 모델에 대해 공격을 수행하여 추출한 입력 데이터이다.

도 13은 추출 가능한 정보 측면에서 기존 연구들과의 차이점을 정리한 표를 도시한다.

도 14는 본 발명의 일 실시예에 따른 시스템을 도시한다.

도 15는 도 14에 도시된 추출 장치의 기능 블럭도이다.

발명을 실시하기 위한 구체적인 내용

[0014] 본 명세서에 개시되어 있는 본 발명의 개념에 따른 실시예들에 대해서 특정한 구조적 또는 기능적 설명들은 단지 본 발명의 개념에 따른 실시예들을 설명하기 위한 목적으로 예시된 것으로서, 본 발명의 개념에 따른 실시예들은 다양한 형태들로 실시될 수 있으며 본 명세서에 설명된 실시예들에 한정되지 않는다.

[0015] 본 발명의 개념에 따른 실시예들은 다양한 변경들을 가할 수 있고 여러 가지 형태들을 가질 수 있으므로 실시예들을 도면에 예시하고 본 명세서에서 상세하게 설명하고자 한다. 그러나, 이는 본 발명의 개념에 따른 실시예들을 특정한 개시 형태들에 대해 한정하려는 것이 아니며, 본 발명의 사상 및 기술 범위에 포함되는 모든 변경, 균등물, 또는 대체물을 포함한다.

[0016] 제1 또는 제2 등의 용어는 다양한 구성 요소들을 설명하는데 사용될 수 있지만, 상기 구성 요소들은 상기 용어들에 의해 한정되어서는 안 된다. 상기 용어들은 하나의 구성 요소를 다른 구성 요소로부터 구별하는 목적으로만, 예컨대 본 발명의 개념에 따른 권리 범위로부터 벗어나지 않은 채, 제1 구성 요소는 제2 구성 요소로 명명될 수 있고 유사하게 제2 구성 요소는 제1 구성 요소로도 명명될 수 있다.

[0017] 어떤 구성 요소가 다른 구성 요소에 "연결되어" 있다거나 "접속되어" 있다고 언급된 때에는, 그 다른 구성 요소에 직접적으로 연결되어 있거나 또는 접속되어 있을 수도 있지만, 중간에 다른 구성 요소가 존재할 수도 있다고 이해되어야 할 것이다. 반면에, 어떤 구성 요소가 다른 구성 요소에 "직접 연결되어" 있다거나 "직접 접속되어" 있다고 언급된 때에는 중간에 다른 구성 요소가 존재하지 않는 것으로 이해되어야 할 것이다. 구성 요소들 간의 관계를 설명하는 다른 표현들, 즉 "~사이에"와 "바로 ~사이에" 또는 "~에 이웃하는"과 "~에 직접 이웃하는" 등도 마찬가지로 해석되어야 한다.

[0018] 본 명세서에서 사용한 용어는 단지 특정한 실시예를 설명하기 위해 사용된 것으로서, 본 발명을 한정하려는 의도가 아니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 명세서에서, "포함하다" 또는 "가지다" 등의 용어는 본 명세서에 기재된 특징, 숫자, 단계, 동작, 구성 요소, 부분품 또는 이들을 조합한 것이 존재함을 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성 요소, 부분품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.

[0019] 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 가진다. 일반적으로 사용되는 사전에 정의되어 있는 것과 같은 용어들은 관련 기술의 문맥상 가지는 의미와 일치하는 의미를 갖는 것으로 해석되어야 하며, 본 명세서에서 명백하게 정의하지 않는 한, 이상적이거나 과도하게 형식적인 의미로 해석되지 않는다.

[0020] 이하, 본 명세서에 첨부된 도면들을 참조하여 본 발명의 실시예들을 상세히 설명한다. 그러나, 특허출원의 범위가 이러한 실시예들에 의해 제한되거나 한정되는 것은 아니다. 각 도면에 제시된 동일한 참조 부호는 동일한 부재를 나타낸다.

[0021] 이하에서는, 기존의 인공신경망 추출 연구를 소개하고 이들의 한계점을 명시한다. 기존 연구는 다음과 같이 분류될 수 있다.

[0022] 1. 딥러닝 서비스에 대해 많은 수의 입출력을 발생시키고 이를 분석하는 oracle 기반 공격

[0023] 2. 딥러닝 서비스가 탑재된 디바이스를 물리적으로 확보한 뒤 전파/전력 부채널 공격을 수행하는 하드웨어 부채널 기반 공격

[0024] 3. 클라우드 등 원격 접근이 가능한 딥러닝 서비스에 대해 소프트웨어적으로 부채널 공격을 수행하는 소프트웨어 부채널 기반 공격

[0025] oracle 기반 공격의 경우 공격 대상 딥러닝 서비스에 지속적으로 쿼리를 보낸 후 이에 대한 응답을 분석하는 과정을 거친다. 이들은 주로 원본 모델을 본떠 대체 모델(substitute network)을 새로 구축하는 방식을 사용한다. 대체 모델은 원본과 비슷한 기능을 수행할 수 있지만 내부 정보를 정확히 알아내는 것은 불가능하다. 또한 추출 가능한 정보의 해상도가 낮아 딥러닝 모델이 복잡해지면 대체 모델을 구성하는 것이 무의미해지기도 한다.

[0026] 하드웨어 부채널 기반 공격은 딥러닝 서비스가 탑재된 디바이스에 직접 전력 분석이나 전자기파 공격 등을 수행하는 방식이다. 따라서 이들의 공격 범위는 매우 한정적인 반면 본 발명은 클라우드 서비스에 탑재된 딥러닝 모델을 공격하는 기법을 제안한다.

[0027] 본 발명을 포함한 최근의 연구들은 원격 서버에서 동작하는 딥러닝 서비스에 대한 추출 공격을 제안한다. 이는 원격으로 통제 가능한 소프트웨어 부채널을 찾고 이를 활용하여 데이터를 유출하는 방식이다. 덕분에 공격범위가 크게 확장되지만, 기존 소프트웨어 부채널 기법들은 전부 인공신경망의 대략적인 구조만을 파악하는데 집중한다. 반면, 본 발명에서 제안하는 기법은 원격 부채널을 통해 목표 딥러닝 프로세스 메모리에 직접 접근하여 인공신경망의 모든 데이터를 추출한다.

[0028] 이하에서는, 본 발명의 배경지식을 설명한다.

[0029] 1. 딥러닝

[0030] 인공신경망(artificial neural network)은 퍼셉트론(perceptron), 가중치(weight), 편향(bias), 활성함수(activation function) 등으로 이루어진 알고리즘으로써 입력값과 목표 출력값을 통해 가중치가 반복적으로 조정되어 최종적으로 입력과 출력 간의 관계가 학습되는 구조이다. 딥러닝은 인공신경망의 발전된 형태로, 여러 은닉층(hidden layer)의 조합으로 구성된다.

[0031] 하나의 은닉층은 여러 퍼셉트론으로 구성되어 있으며 각 퍼셉트론에서 내부연산을 거친 출력값은 다음 은닉층의 입력으로 주어진다. 단일 퍼셉트론 내부에서는 입력과 가중치의 가중합에 편향을 더한 후 활성함수를 적용하여 다음 은닉층으로 넘겨준다.

[0032] 최근 딥러닝 서비스의 신속하고 편리한 운용을 위해 클라우드가 주목을 받고 있다. 인공신경망 학습을 위한 대용량 연산 환경과 서비스 배포에 적합한 유동적인 API 환경을 제공하기에 클라우드는 직접 서버를 구축하는 것보다 저렴하다. Infrastructure-as-a-Service(IaaS), Platform-as-a-Service(PaaS), Machine-Learning-as-a-Service(MLaaS), Software-as-a-Service(Saas) 등 여러 형태의 클라우드 서비스가 존재하고 이들은 대부분 멀티-테넌시 환경에서 동작한다. 각 기업은 수요에 따라 서비스 형태를 선택하고 그 위에서 사용자에게 딥러닝 서비스를 제공한다.

[0033] 2. 멜트다운(Meltdown)

[0034] 멜트다운은 프로세서의 비순차 실행(out-of-order execution)을 악용하여 접근 권한이 없는 메모리를 읽을 수 있는 하드웨어 취약점이다. 2018년 초 공개 당시 대부분의 Intel 프로세서와 ARM Cortex-A75 기반 시스템이 이 취약점에 영향을 받는 것으로 알려졌다. 비순차 실행은 프로세서의 파이프라인을 효과적으로 사용하기 위한 최적화 기술로, 명령어의 순서에 관계없이 실행 가능한 명령어를 우선적으로 실행하는 기술이다. 프로세서는 비순차적으로 실행된 명령어가 커밋(commit)되기 전에 권한 검사를 진행한다. 권한이 없는 메모리 영역에 접근한 것이 확인되면 rollback interrupt를 통해 해당 명령어의 실행을 정지하고, 파이프라인을 초기화한다. 따라서 일반적인 경우 사용자는 해당 명령어의 최종 수행 결과를 알 수 없다. 하지만 해당 명령어가 비순차적으로 실행되는 동안 참조한 메모리 내용이 캐시에 적재되고, 이 데이터는 일반적인 방법으로 접근이 불가능하지만 Flush+Reload와 같은 캐시 부채널 공격을 수행하여 해당 데이터를 추출할 수 있다.

[0035] 도 1은 멜트다운 공격의 핵심인 x86 어셈블리 코드이다. 공격자는 목표 메모리 주소 rcx의 값 X를 알아내려 한다. line 4 명령은 X를 rax 레지스터에 저장한다. 이는 권한이 없는 접근이기에 커밋되기 전에 rollback이 이루어지지만, rollback interrupt가 발생하기 전에 비순차 실행으로 인해 line 5~7 명령이 실행된다. 이 중 line 7 명령은 rbx로부터 X번째 페이지의 내용에 접근하여 해당 페이지를 캐시에 적재한다. 이후 해당 페이지에 다시 접근할 때는 cache hit이 발생하여 다른 페이지에 비해 소요시간이 짧게 관측될 것이다 이러한 접근 속도 차이

로 인해 공격자는 X의 값을 알아낼 수 있다. 공격자는 자신의 프로세서의 모든 페이지에 하나하나 접근하면서 소요시간을 측정하고, 그 중 접근 속도가 빠른 페이지가 line 7에서 접근했던 X번째 페이지임을 알 수 있다.

[0036] 멜트다운 이전의 캐시 부채널 공격들은 L1 캐시의 부채널을 활용했기 때문에 공격자와 공격 대상 프로세스(이하 victim)가 같은 코어 내에 할당되어야 하는 강한 조건이 필요하다. 하지만 멜트다운의 경우 Last Level Cache(LLC)의 부채널을 활용하기 때문에 동일 CPU 배치만이 전제된다. 클라우드 환경에서는 여러 사용자가 한정된 숫자의 하드웨어를 공유하기 때문에 동일 CPU에 배치되는 경우가 많다. 이러한 사실을 악용하여 공격자는 클라우드 환경에서 멜트다운 공격을 통해 다른 사용자의 데이터에 접근할 수 있다.

[0037] 3. 동적 메모리 할당

[0038] 컴파일 단계에서 자료형과 크기가 정해지는 정적 할당(static allocation)과 달리 동적 할당(dynamic allocation)은 런타임에서야 그 크기가 결정되고 이에 맞춰 공간 배치가 이루어진다. 따라서 동적 할당은 필연적으로 메모리 영역을 할당하고 해당 영역에 대한 포인터를 저장하는 단계를 거친다.

[0039] 본 발명에서는 Python의 동적 메모리 할당에 집중한다. Python은 런타임에 동적 할당된 객체들을 추적하기 위해 딕셔너리 형태의 심볼테이블(symbol table)을 사용하고, 이는 *PyDict_Type* 인스턴스로 저장된다.

[0040] *PyDict_Type* 인스턴스에는 심볼테이블에 관한 여러 정보가 저장된다. 딕셔너리의 직접적인 데이터는 외부 위치(*PyDictEntry*)에 저장되고, *PyDict_Type* 인스턴스에는 해당 *PyDictEntry*로의 포인터가 기록되어 있다. *PyDictEntry*의 각 요소는 해시(hash), 키(key) 포인터, 값(value) 포인터로 구성된다. 키와 값 정보는 직접 기록되지 않고 해당 데이터로의 포인터를 갖는다. 키 포인터와 값 포인터는 각각 변수의 이름에 해당하는 문자열 인스턴스와 해당 변수의 실제 데이터에 대당하는 인스턴스를 가리킨다. 도 2는 *PyDictEntry*에 저장된 데이터의 모습니다.

[0041] 이하에서는 다른 유저의 딥러닝 프로세스에 접근해 인공신경망을 추출하는 기법(방법)을 설명한다. 추출 공격은 총 6단계로 구성되며 이 중 일부를 수정하여 딥러닝 서비스로의 입력 쿼리를 추출하는 공격 또한 가능함을 보인다.

[0042] 1단계: 사전 작업

[0043] 사전 작업에서는 공격 가능 환경 여부 파악과 victim과 같은 CPU 사용 여부를 파악하기 위한 co-location test 등이 포함된다. 공격 가능 환경 여부는 시스템 버전을 확인하는 것 만으로 수행 가능하다. co-location 여부 또한 victim 딥러닝 서비스에 입력 쿼리를 보내고 Flush+Reload 등 캐시 부채널을 이용하여 함수 사용을 모니터링하는 것으로 간단하게 파악할 수 있다.

[0044] 2단계: victim의 페이지 테이블 추출

[0045] 리눅스 커널은 프로세스 관리를 위해 모든 프로세스 정보를 연결리스트 형태로 저장한다. 해당 연결리스트의 헤드(head)는 *init_task* 구조체에 저장되어 있고, 이는 커널별로 고정된 위치에 존재한다. 따라서 공개된 리눅스 커널 빌드를 참조하면 프로세스 정보가 저장되어 있는 연결리스트의 헤드 위치를 파악할 수 있다. 이 위치에 대해 멜트다운 공격을 진행한다. 연결리스트의 다음 노드에 대한 포인터를 참조하여 victim의 노드까지 반복적으로 새로운 노드에 대해 멜트다운 공격을 진행한다. 이때 노드 내에 저장되어 있는 프로세스 이름 등을 이용해 해당 노드가 victim의 것인지 판단할 수 있다. 즉, 미리 알려진 victim의 프로세스의 이름 등을 이용하여 victim의 노드를 식별할 수 있다.

[0046] 각 노드에는 해당 프로세스의 메모리 매핑 정보, PID, 이름 등의 정보 뿐만 아니라 페이지 테이블(page table)의 위치도 기록되어 있다. 이를 이용하여 victim의 페이지 테이블 위치를 특정하고 해당 위치에 대해 멜트다운 공격을 진행하여 페이지 테이블을 확보한다.

[0047] 3단계: victim 힙 영역 추출

[0048] 프로세스 내부 데이터의 위치를 식별하기 위해서는 이러한 정보가 저장되어 있는 심볼테이블을 확보해야 한다. 앞서 언급되었듯이 Python의 심볼테이블은 *PyDict_Type*라는 딕셔너리 인스턴스로 관리된다. 딕셔너리의 모든 데이터는 힙 영역에 할당되기 때문에 victim의 힙 영역 데이터를 먼저 확보해야 한다.

[0049] 2단계에서 추출한 페이지 테이블을 이용하여 victim이 매핑된 물리주소(physical address)를 파악하고, 해당 위치에 대해 멜트다운 공격을 진행하여 victim 메모리의 힙 영역을 추출한다.

[0050] 4단계: victim 인공신경망 인스턴스 추출

[0051] 목표지향적인 공격을 위해 유의미한 데이터의 위치를 파악해야 한다. 따라서 인공신경망(이하 모델)을 추출하기에 앞서 프로세스 메모리의 어느 부분에 인공신경망이 매핑되어 있는지 파악하는 과정이 선행된다.

[0052] 모든 Python 인스턴스는 자신의 자료형을 지칭하는 클래스에 대한 포인터를 갖는다. 예를 들어, 모든 유니코드 문자열 데이터는 *PyUnicode_Type* 클래스를 참조한다. 이러한 기본 자료형 클래스의 위치는 Python 빌드별로 고정적이기 때문에 이 위치 값을 검색하여 특정 자료형의 인스턴스를 전부 포착할 수 있다.

[0053] 도 3은 Python 3.5.2 빌드에서의 각 자료형 클래스가 존재하는 메모리 위치를 나타내는 표를 도시한다. 도 3을 참조하면, *PyDict_Type* 클래스는 *0xa3ab20*에 고정적으로 매핑됨을 알 수 있고, 모든 딕셔너리 인스턴스는 이 주소값을 저장하고 있음을 유추할 수 있다. 따라서 3단계에서 추출한 힙 영영 데이터에서 이 주소값을 포함하는 인스턴스를 검색하여 딕셔너리 인스턴스를 모두 포착할 수 있다.

[0054] 도 4의 전역 심볼테이블은 포착된 딕셔너리 중 전역 심볼(global symbol)을 포함하고 있는 딕셔너리이다. 전역 심볼테이블만이 포함하는 요소들의 존재 여부로 여러 딕셔너리 중 전역 심볼테이블을 특정해 낼 수 있다. 해당 딕셔너리의 요소 중 모델 인스턴스의 주소에 대해 멜트다운 공격을 수행하여 목표 인스턴스를 추출한다.

[0055] 5단계: 모델 구조 추출

[0056] 모델의 인스턴스 안에는 해당 인스턴스 내부에서 사용하는 심볼테이블이 존재한다. 4단계에서 전역 심볼테이블을 선별한 것과 같은 방식으로 인스턴스 심볼테이블을 포착한다. 즉, 심볼테이블은 *PyDict_Type* 클래스이고, *Pydict_Type* 인스턴스는 0xa3ab20 주소값을 저장하고 있다. 따라서, 모델의 인스턴스에서 해당 주소값을 포함하는 인스턴스를 검색함으로써 심볼테이블을 획득할 수 있다.

[0057] 포착된 모델의 인스턴스는 도 5의 형식을 띈다. 이 중 '*network_nodes*'에 인공 신경망 레이어 정보가 배열 형태로 존재하는 것을 확인할 수 있다. 따라서 이 요소를 추출하는 것으로 모델의 구조를 파악할 수 있다.

[0058] 6단계: 모델의 가중치 추출

[0059] 이제 모델의 가중치를 추출하기 위해 은닉층 정보가 어디에 저장되어 있는지 알아낸다. 모델 인스턴스의 심볼테이블에서 해당하는 요소를 찾고, 이를 분석하여 각 은닉층의 위치를 파악한다. 도 5의 '_layers' 요소의 값은 각 은닉층 인스턴스의 포인터로 구성된 리스트이므로 이를 활용한다.

[0060] 각 은닉층 인스턴스에 접근하여 동일한 방식으로 해당 인스턴스의 심볼테이블을 포착하고, 이를 활용하여 '_trainable_weights' 요소의 값에 접근한다. 즉, 심볼테이블은 *PyDict_Type* 클래스이고, *Pydict_Type* 인스턴스는 0xa3ab20 주소값을 저장하고 있다. 따라서, 은닉층 인스턴스에서 해당 주소값을 포함하는 인스턴스를 검색함으로써 심볼테이블을 획득할 수 있다.

[0061] 도 6의 '*_trainable_weights*' 요소는 두가지 *tf.Variable*의 배열로 구성되어 있음을 확인할 수 있다. 이는 각각 가중치와 편향 값에 해당하는 행렬이다. *tf.Variable*은 텐서 데이터를 위해 TensorFlow 내부적으로 사용하는 자료형이다.

[0062] *tf.Variable*의 구조는 도 7과 같다. 행렬을 row-major 형태의 배열로 저장하고 stride와 dimension 등 헤더 정보를 이용하여 배열 데이터를 행렬로 해석한다. 도 7의 메모리 블록은 3x3 2차원 행렬을 나타낸 것으로 데이터 하나당 4바이트, 한 행은 12바이트임을 알 수 있다.

[0063] 각 *tf.Variable*에 대한 멜트다운 공격을 통해 배열과 헤더 정보를 추출하고 이를 해석하여 원본 가중치 및 편향 행렬로 복원한다.

[0064] 이하에서는, 본 발명에 의한 인공신경망 추출 방법을 적용하여 실험한 결과에 대해 설명한다.

[0065] 1. 실험 환경

[0066] 실험은 로컬 리눅스 서버에서 진행되었으며 커널에서 기본적으로 제공하는 멀티 유저 환경을 사용했다. victim과 공격자는 각기 다른 유저로 해당 시스템에 접근하며 커널에 의해 격리된 독립적인 환경을 부여받는다. victim으로는 TensorFlow MNIST 손글씨 인식 딥러닝 서비스를 사용하였으며 구체적인 환경 설정은 다음과 같다.

[0067] · CPU: Intel(R) Core(TM) i5-7500

[0068] · kernel: Linux version4.12.0

[0069] · OS: Ubuntu 16.04.6 LTS

[0070] 도 8은 victim 딥러닝 모델의 정보이고 공격자의 목표는 이를 추출하는 것이다. 실험의 편의를 위해 관리자 권한으로 공격의 3단계를 수행한 후 권한없이 나머지 단계를 진행하였다.

[0071] 2. 실험 결과

[0072] 도 9는 모델 인스턴스 심볼데이터의 '*network_nodes*' 요소에 대한 멜트다운 공격으로 배열의 각 아이템을 추출한 결과이다. 각 은닉층의 이름이 아스키 코드로 저장되어 있는 것을 볼 수 있다. 이를 참고하여 모델 구조를 재현한다.

[0073] 도 10은 victim 인공신경망의 가중치 정보를 성공적으로 추출한 결과이다. 각 가중치 값은 little endian의 부동소수점 형태로 변환되어 저장되기 때문에 도 11을 참조하여 첫 번째 가중치를 정확히 추출했음을 확인할 수 있으며 첫 번째 가중치에 이어 다른 가중치 정보도 연속적으로 저장되어 있음을 볼 수 있다.

[0074] 또한 딥러닝 서비스로의 쿼리 입력을 목표로 하도록 공격의 4단계를 수정하여 딥러닝 모델의 입력을 추출할 수 있다. 이는 사용자 얼굴 인식이나 음성 인식 등 민감한 데이터가 입력으로 사용되는 경우 큰 위협을 야기한다. 도 12는 VGG16 이미지 인식 딥러닝 모델에 대해 공격을 수행하여 추출한 입력 데이터이다. 이는 단일 공격으로 추출된 결과이며 반복적인 공격으로 노이즈를 보정하여 추출 정확도를 향상시킬 수 있다.

[0075] 본 발명 이전에도 인공신경망 복원 공격 기법에 대한 다양한 연구가 진행되었다. 하지만 기존의 인공신경망 복원 공격들은 인공신경망의 구조를 복원하는데 그치기에 가중치나 활성함수 종류 등 기타 내부 정보는 복원할 수 없다. 또한 함수사용 모니터링이나 소요시간 만으로 구조를 복원하는 많은 기법들은 인공신경망 구조가 복잡해질수록 정확도가 급격히 감소하는 반면 본 발명에서 제안하는 공격 기법은 인공신경망의 복잡도에 영향을 받지 않는다.

[0076] 앞서 진행한 실험은 인공신경망의 구조, 가중치, 입력 이미지 정보를 추출해오는 것만을 보여주지만, 동일한 방법으로 인공신경망과 관련된 다른 모든 정보를 추출할 수 있다. 도 13은 추출 가능한 정보 측면에서 기존 연구들과의 차이점을 정리한 표를 도시한다.

[0077] 본 발명에서 제시하는 공격의 성능을 평가하기 위해, victim 인공신경망의 가중치 값과 추출해낸 가중치 값을 비교했다. 이를 통해 바이트 수준에서의 추출 정확도와 소요 시간을 측정했다. 해당 실험에서는 멜트다운의 기저로 Flush+Reload를 사용했며, 앞서 기재한 실험 환경에서 추출 공격을 분석한 결과 92.875%의 정확도와 1.325kB/s의 추출 속도를 보였다. 이러한 성능은 CPU 및 환경 별로 조금씩 차이를 보이며 다음과 같은 방법으로 최적화할 수 있다.

[0078] 최적화 1: 멜트다운 성능 향상

[0079] 본 발명에서 제안하는 공격의 성능은 멜트다운의 성능에 의존하므로 멜트다운 샘플링 횟수를 증가시키거나 victim과 동일 코어를 사용하도록 설정하는 것으로 전체 공격 파이프라인의 속도와 정확도를 향상시킬 수 있다.

[0080] 최적화 2: Python 고정 해시 활용

[0081] 심볼테이블 딕셔너리에는 해시와 키 포인터, 값 포인터 정보가 존재한다. 따라서 심볼테이블을 확보하는 것 만으로는 각 요소가 무엇을 뜻하는지 알 수 없고, 멜트다운으로 키 포인터 위치를 추출해야만 해당 요소의 변수명을 파악할 수 있었다. 하지만 Python 3.3 이전 버전의 경우 딕셔너리의 해시 부분에 간단한 해시함수를 사용한다. 따라서 공격자는 키 포인터를 따로 추출할 필요 없이, 자신이 원하는 변수명의 해시를 직접 구한 후 딕셔너리의 해시와 비교해 원하는 변수를 선별할 수 있다. 이 경우 심볼테이블을 해석하는 과정이 단축되어 공격의 성능이 향상된다.

[0082] 도 14는 본 발명의 일 실시예에 따른 시스템을 도시한다. 이하의 설명에서, 앞선 기재와 중복되는 내용에 관하여는 그 구체적인 기재를 생략하기로 한다.

[0083] 인공신경망 추출 시스템, 추출 시스템 등으로 명명될 수 있는 시스템(10)은 사용자 단말(100), 추출 장치(300), 및 서버(500)를 포함한다.

[0084] 서버(500)는 등록된 사용자인 사용자 단말(100) 및/또는 추출 장치(300)에게 클라우드 서비스, 예컨대 IaaS(Infrastructure-as-a-Service), PaaS(Platform-as-a-Service), MLaaS(Machine-Learning-as-a-Service), 및 SaaS(Service-as-a-Service) 중 적어도 하나의 서비스를 제공할 수 있다. 사용자 단말(100)과 서버(500) 그

리고 추출 장치(300)와 서버(500)는 소정의 유무선 통신망을 통하여 통신할 수 있다.

[0085] 또한, 서버(500)는 물리적으로 분리된 단일의 서버를 의미할 수 있다. 다시 말해, 사용자 단말(100)과 추출 장치(300)는 단일의 서버에 의해 소정의 서비스를 제공받을 수 있다.

[0086] 사용자 단말(100)은 서버(500)로부터 소정의 서비스를 제공받는 단말로써, 실시예에 따라 복수의 단말들을 의미할 수 있다. 사용자 단말(100)이 서버(500)로부터 제공받는 서비스는 MLaaS일 수 있으나, 본 발명이 이에 제한되는 것은 아니다.

[0087] 추출 장치(300)는 적어도 프로세서 및/또는 메모리를 포함하는 컴퓨팅 장치로써, 서버(500)가 사용자, 예컨대 사용자 단말(100)에게 제공하는 인공신경망을 추출할 수 있다. 추출 장치(300)는 서버(500)와의 통신을 통해 데이터를 주고 받으며, 수신된 데이터에 기초하여 사용자 단말(100)에게 제공하는 인공신경망 모델의 구조, 인공신경망 모델의 가중치(weights), 인공신경망 모델의 편향(bias), 및 인공신경망 모델의 입력 중 적어도 하나를 추출할 수 있다. 이를 위해, 추출 장치(300)는 서버(500)의 메모리 영역의 소정 위치에 대해 멜트다운 공격을 수행할 수 있다.

[0088] 도 15는 도 14에 도시된 추출 장치의 기능 블럭도이다.

[0089] 도 15를 참조하면, 추출 장치(300)는 페이지 테이블 추출부(310), 힙 영역 추출부(320), 인스턴스 추출부(330), 모델 구조 추출부(340), 및 가중치 추출부(350)를 포함한다. 실시예에 따라, 추출 장치(300)는 저장부(360)를 더 포함할 수 있다.

[0090] 추출 장치(300)는 적어도 프로세서 및/또는 메모리를 포함하는 컴퓨팅 장치로써, 이하에서의 각 구성들의 적어도 일부의 동작은 프로세서의 동작으로 이해될 수 있다. 실시예에 따라, 추출 장치(300)는 서버(500)와 데이터를 주고받을 수 있는데, 이러한 동작은 추출 장치(300)의 프로세서가 추출 장치(300)에 추가적으로 포함될 수 있는 통신 모듈(미도시)을 제어함으로써 소정의 데이터를 서버(500)로 송신하고 이에 대응하는 데이터를 수신받는 동작으로 이해될 수 있다.

[0091] 페이지 테이블 추출부(310)는 공격 대상 프로세스이 페이지 테이블을 추출할 수 있다. 공격 대상 프로세스는 공격 대상 인공신경망 모델, 추출 대상 인공신경망 모델, 빅팀(victim), 공격 대상 모델, 추출 대상 모델 등으로 명명될 수도 있다.

[0092] 이를 위해, 페이지 테이블 추출부(310)는 프로세스 정보가 저장되어 있는 연결리스트의 헤드 위치에 대해 멜트다운 공격을 수행할 수 있다. 연결리스트의 헤드 위치는 공개된 리눅스 커널 빌드를 통해 사전에 파악될 수 있다. 또한, 페이지 테이블 추출부(310)는 연결리스트의 다음 노드에 대한 포인터를 참조하여 빅팀의 노드(추출 대상 노드라 명명될 수 있음)까지 반복적으로 새로운 노드에 대해 멜트다운 공격을 수행할 수 있다. 페이지 테이블 추출부(310)는 노드 내에 저장되어 있는 프로세스 이름, PID 등을 이용해 해당 노드가 빅팀의 것인지 판단할 수 있다. 이를 통해 페이지 테이블 추출부(310)는 빅팀의 노드에 저장되어 있는 데이터를 추출할 수 있다.

[0093] 또한, 페이지 테이블 추출부(310)는 추출된 노드에 기록되어 있는 페이지 테이블(page table)의 위치에 대해 멜트다운 공격을 수행함으로써 빅팀의 페이지 테이블을 획득할 수 있다.

[0094] 힙 영역 추출부(320)는 빅팀이 매핑된 물리주소(physical address)에 대해 멜트다운 공격을 진행하여 빅팀 메모리의 힙 영역(힙 영역 데이터)을 추출할 수 있다. 매핑된 물리주소는 페이지 테이블 추출부(310)에 의해 추출된 페이지 테이블 내에 포함될 수 있다.

[0095] 인스턴스 추출부(330)는 빅팀의 인공신경망 인스턴스를 추출할 수 있다. 이를 위해, 인스턴스 추출부(330)는 힙 영역 추출부(320)에 의해 추출된 힙 영역 데이터로부터 딕셔너리 인스턴스만을 추출할 수 있다. 추가적으로, 인스턴스 추출부(330)는 추출된 딕셔너리 인스터스 중 전역 심볼(global symbol)을 포함하고 있는 전역 심볼테이블을 선택할 수 있다. 선택된 전역 심볼테이블에는 모델 인스턴스의 주소가 기재되어 있기 때문에, 인스턴스 추출부(330)는 모델 인스턴스의 주소에 대해 멜트다운 공격을 수행함으로써 목표 인스턴스, 즉 빅팀의 인공신경망 인스턴스를 추출할 수 있다.

[0096] 모델 구조 추출부(340)는 모델 구조를 추출할 수 있다. 이를 위해, 모델 구조 추출부(340)는 추출된 목표 인스턴스로부터 인스턴스 심볼테이블을 획득할 수 있다. 모델 구조 추출부(340)는 인스턴스 심볼테이블에 포함된 인공신경망 레이어 정보에 기초하여, 모델의 구조를 파악(추출)할 수 있다.

[0097] 가중치 추출부(350)는 모델에 포함된 가중치 및/또는 편향 값을 추출할 수 있다. 이를 위해, 가중치 추출부

(350)는 모델 인스턴스의 심볼테이블에 포함되어 있는 각 은닉층 인스턴스의 포인터 정보를 이용하여 각 은닉층 인스턴스에 접근할 수 있다. 즉, 가중치 추출부(350)는 각 은닉층 인스턴스의 심볼테이블에 기재된 각 은닉층의 가중치 및/또는 편향 값을 추출할 수 있다.

[0098] 저장부(360)에는 페이지 테이블 추출부(310)에 의해 추출된 페이지 테이블, 페이지 테이블을 추출하기 위해 필요한 데이터나 페이지 테이블을 추출하는 과정에서 일시적으로 또는 비일시적으로 생성되는 데이터, 힙 영역 추출부(320)에 의해 추출된 힙 영역 데이터, 인스턴스 추출부(330)에 의해 추출된 딕셔너리 인스턴스, 전역 심볼테이블, 목표 인스턴스, 모델 구조 추출부(340)에 의해 추출된 모델의 구조, 가중치 추출부(350)에 의해 추출된 각 은닉층의 가중치 및/또는 편향 값 등이 저장될 수 있다.

[0099] 상술한 바와 같이, 추출 장치(300)는 서버(500)의 메모리 내에서 소정의 위치에 대해 멜트다운 공격을 수행한 결과로 획득한 데이터를 이용하여 사용자 단말(100)에 제공되는 인공신경망 구조, 각 은닉층의 가중치, 및 편향 값 중 적어도 하나를 추출할 수 있다.

[0100] 이상에서 설명된 장치는 하드웨어 구성 요소, 소프트웨어 구성 요소, 및/또는 하드웨어 구성 요소 및 소프트웨어 구성 요소의 집합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성 요소는, 예를 들어, 프로세서, 콘트롤러, ALU(Arithmetic Logic Unit), 디지털 신호 프로세서(Digital Signal Processor), 마이크로컴퓨터, FPA(Field Programmable array), PLU(Programmable Logic Unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 하나 이상의 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(Operation System, OS) 및 상기 운영 체제 상에서 수행되는 하나 이상의 소프트웨어 애플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술 분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(Processing Element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 콘트롤러를 포함할 수 있다. 또한, 병렬 프로세서(Parallel Processor)와 같은, 다른 처리 구성(Processing Configuration)도 가능하다.

[0101] 소프트웨어는 컴퓨터 프로그램(Computer Program), 코드(Code), 명령(Instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(Collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성 요소(Component), 물리적 장치, 가상 장치(Virtual Equipment), 컴퓨터 저장 매체 또는 장치, 또는 전송되는 신호 파(Signal Wave)에 영구적으로, 또는 일시적으로 구체화(Embody)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

[0102] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(Magnetic Media), CD-ROM, DVD와 같은 광기록 매체(Optical Media), 플롭티컬 디스크(Floptical Disk)와 같은 자기-광 매체(Magneto-optical Media), 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다. 상기된 하드웨어 장치는 실시예의 동작을 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

[0103] 본 발명은 도면에 도시된 실시예를 참고로 설명되었으나 이는 예시적인 것에 불과하며, 본 기술 분야의 통상의 지식을 가진 자라면 이로부터 다양한 변형 및 균등한 타 실시예가 가능하다는 점을 이해할 것이다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성 요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성 요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다. 따라서, 본 발명의 진정한 기술적 보호 범위는 첨부된 등록청구범위의 기술적 사상에 의해 정해져야 할 것이다.

부호의 설명

[0104] 10 : 시스템

100 : 사용자 단말

300 : 추출 장치

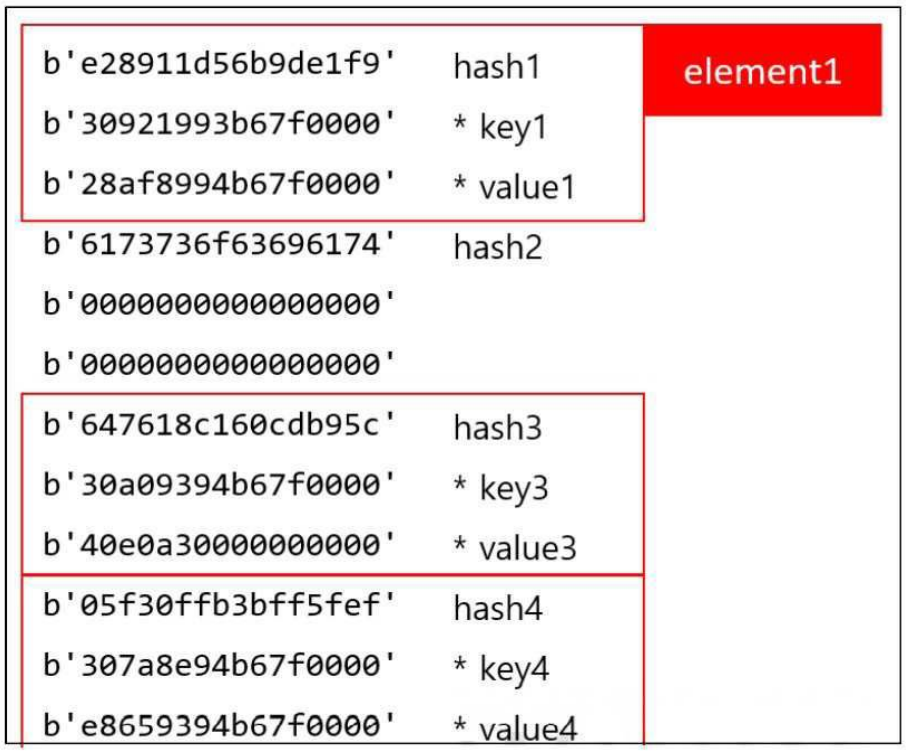
500 : 서버

도면

도면1

```
1 ; rcx = kernel address, rbx = probe array
2 xor rax, rax
3 retry:
4 mov al, byte [rcx]
5 shl rax, 0xc
6 jz retry
7 mov rbx, qword [rbx + rax]
```

도면2



도면3

data type	address(offset)
PyObject_Type	*0x6367f0*
PyLong_Type	*0xa3c900*
PyBool_Type	*0xa3d160*
PyFloat_Type	*0xa3d300*
PyDict_Type	*0xa3ab20*
PyUnicode_Type	*0xa38440*

도면4

```
{'__name__': '__main__',
…,
'__builtins__': <module 'builtins' (built-in)>,
'tensorflow': <module 'tensorflow' from
'/home/hoyong/.local/lib/python3.6/site-
packages/tensorflow/__init__.py'>,
'model': <tensorflow.python.keras.engine.
sequential.Sequential object at 0x7f4abfcee978>}
```

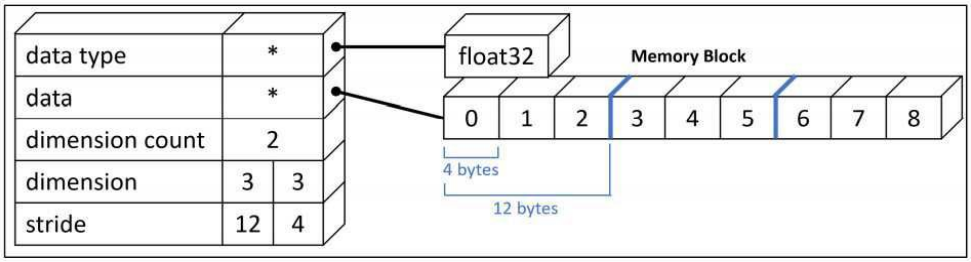
도면5

```
{…,
'_name': 'sequential_6',
…,
'_layers': [<tensorflow.python.keras.engine.input_layer.InputLayer
object at 0x7f5b7e6d5160>, <tensorflow.python.keras.layers.core.Flatten
object at 0x7f5b7e6d1e10>, <tensorflow.python.keras.layers.core.Dense
object at 0x7f5b7e6cf390>,<tensorflow.python.keras.layers.core.Dense
object at 0x7f5b7e6cf400>],
…,
'_network_nodes': {'flatten_input_ib-6', 'flatten_ib-6', 'dense_ib-12',
'dense_1_ib-13'},
…,
'optimizer': <tensorflow.python.keras.optimizer_v2.adam.Adam object at
0x7f5b7e6d3240>,
…,
'loss': 'sparse_categorical_crossentropy',
…}
```

도면6

```
{…,
'_name': 'dense_12',
…,
'_trainable_weights':[
        <tf.Variable 'dense_12/kernel:0' shape=(784, 10) dtype=float32>,
        <tf.Variable 'dense_12/bias:0' shape=(10,) dtype=float32>
],
'_non_trainable_weights': [],
…}
```

도면7



도면8

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_6 (Flatten)          (None, 784)               0
_________________________________________________________________
dense_12 (Dense)             (None, 10)                7850
_________________________________________________________________
dense_13 (Dense)             (None, 10)                110
=================================================================
Total params: 7,960
Trainable params: 7,960
Non-trainable params: 0
_________________________________________________________________

>> layer2
[<tf.Variable 'dense_13/kernel:0' shape=(10, 10) dtype=float32, numpy=
array([[-1.3233812 ,  0.26736608,  1.0216433 ,  0.59734255, -1.0421642 ,
        -0.3907373 , -0.13165668,  1.0037075 , -1.406748  , -2.09728   ],
       [-0.87404656,  0.26599044,  0.17201532, -0.1804208 ,  0.43607864,
        -2.5771687 , -1.6013101 ,  0.25765058, -0.36030576,  0.3341029 ],
       [ 0.60462254,  0.5616927 , -0.3610924 ,  0.46177837, -1.6261365 ,
```

도면9

```
e4 5c a3 00 00 00 00 00 00 00 00 00 00 00 00 | .\.............. |
66 6c 61 74 74 65 6e 5f 36 5f 69 6e 70 75 74 5f | flatten_6_input_ |
69 62 2d 30 00 00 00 00 02 00 00 00 00 00 00 00 | ib-0............ |
…
e4 78 a3 01 00 00 00 00 94 af 00 00 00 00 00 00 | .x.............. |
66 6c 61 74 74 65 6e 5f 36 5f 69 62 2d 30 00 00 | flatten_6_ib-0.. |
00 00 00 00 00 00 00 00 f0 10 ce 74 23 7f d4 8f | ...........t#... |
…
e4 78 a3 00 00 00 00 00 00 00 00 00 00 00 00 00 | .x.............. |
64 65 6e 73 65 5f 31 32 5f 69 62 2d 30 00 00 00 | dense_12_ib-0... |
03 00 00 00 00 00 00 00 40 84 a3 00 00 00 00 00 | ........@....... |
```

도면10

```
4260b0780: 8e 64 a9 bf 35 e4 88 3e 35 c5 82 3f 71 eb 18 3f
4260b0790: 01 65 85 bf b8 0e c8 be 02 d1 06 84 7d 79 80 3f
4260b07a0: 52 10 b4 bf d6 39 06 c0 84 c1 5f bf e6 2f 88 3e
4260b07b0: c9 24 30 3e 3b c0 38 be b3 45 df 3e 55 f0 24 c0
4260b07c0: …
```

도면11

Hex value: 0x8e64a9bf Convert to float

0xbfa9648e (swapped endianness)

b	f	a	9	6	4	8	
1 0 1 1	1 1 1 1	1 0 1 0	1 0 0 1	0 1 1 0	0 1 0 0	1 0 0 0	1 1

1	01111111	01010010110010010001110

sign	exponent	mantissa
-1	127	1.01010010110010010001110 (binary)
-1 *	2^(127 - 127) *	1.3233811855316162
-1 *	1.00000000 *	1.3233811855316162

-1.32338

Float value: -1.3233812 Convert to hex

도면12

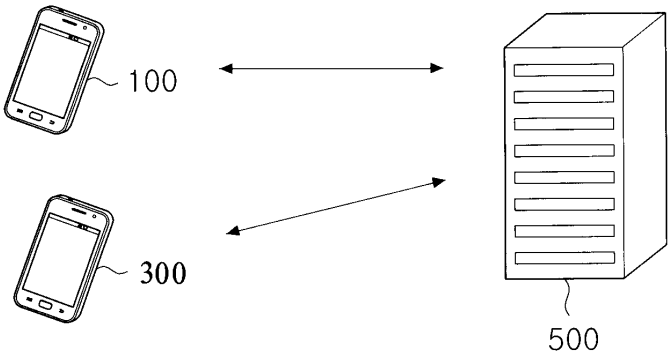


도면13

		structure	learning rate	loss function	weight & bias	model input
oracle	via prediction APIs [10]	△				×
	practical black-box attacks [11]	△				×
	high accuracy and fidelity [12]	△				×
	stealing hyperparam.s in ML [13]	×	×	◐	×	×
hardware side-channel	I know what you see [14]	×	×	×	×	●
	floating-point multiplication [15]	×	×	×	×	●
	CSI NN [16]	●	×	×	◐	×
software side-channel	via timing side-channels [17]	◐	×	×	×	×
	leaky DNN [18]	●	×	×	×	×
	cache telepathy [19]	●	×	×	×	×
	our attack	●	●	●	●	●

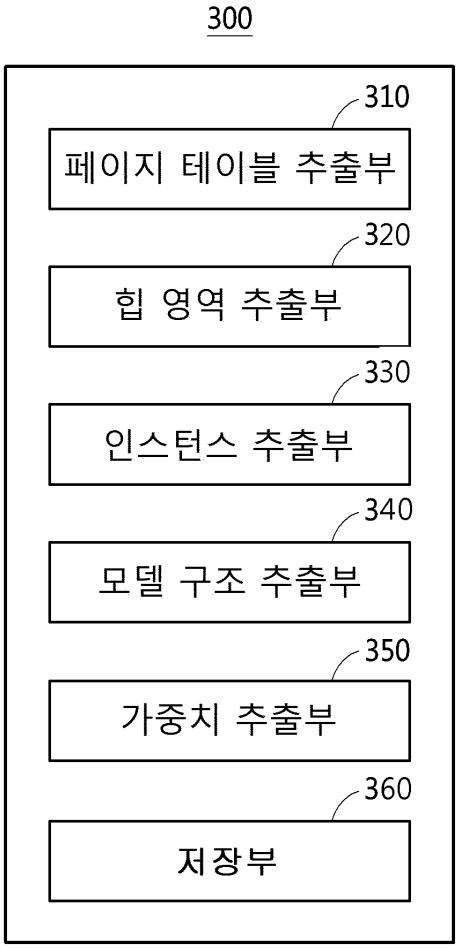
● indicates that the technique can successfully steal the corresponding information.
× indicates that the technique cannot extract corresponding information.
◐ indicates that the extraction is only successful with simple neural networks.
△ indicates that the technique does not steal the corresponding information directly, but constructs a substitute network. The contents of such substitute network are distant from the original victim network; hence it cannot be regarded as an extraction.

도면14



10

도면15


300
310
페이지 테이블 추출부
320
힙 영역 추출부
330
인스턴스 추출부
340
모델 구조 추출부
350
가중치 추출부
360
저장부