(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2024-0107991
(43) 공개일자 2024년07월09일

(51) 국제특허분류(Int. Cl.)
G06F 21/85 (2013.01) *G06F 21/44* (2013.01)
G06F 21/57 (2013.01)
(52) CPC특허분류
G06F 21/85 (2013.01)
G06F 21/44 (2013.01)
(21) 출원번호 10-2022-0191150
(22) 출원일자 2022년12월30일
심사청구일자 2022년12월30일

(71) 출원인
충남대학교산학협력단
대전광역시 유성구 대학로 99 (궁동, 충남대학교)
(72) 발명자
장진수
대전광역시 유성구 대학로 99, 충남대학교 공대5호관 620호
한승균
대전광역시 유성구 은구비남로 55, 열매마을7단지 707-103
(74) 대리인
이은철, 김재문

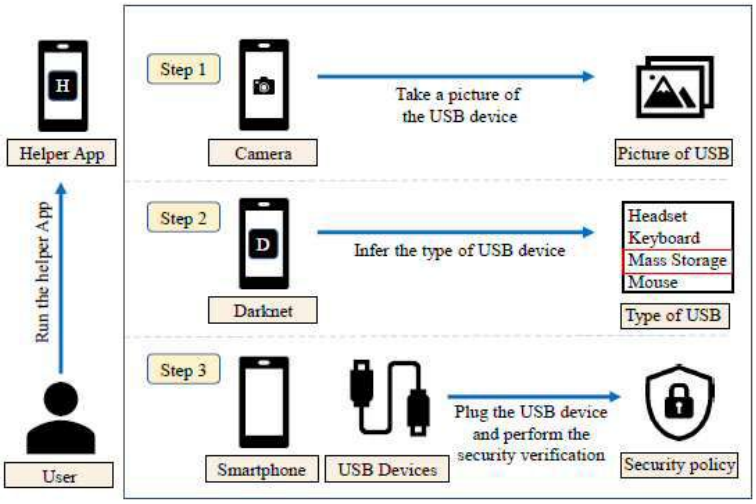
전체 청구항 수 : 총 6 항

(54) 발명의 명칭 **모바일 기기의 USB 장치 인증 시스템 및 방법**

(57) 요 약

본 발명은 모바일 기기의 USB 장치 인증 시스템에 관한 것으로써, 모바일 기기에 대한 USB 악성 장치를 활용한 공격을 방어하기 위한 모바일 기기의 USB 장치 인증 시스템 및 방법에 관한 것이다. 본 발명에 따른 모바일 기기의 USB 장치 인증 시스템은 인공지능 기반으로 USB 장치의 유형을 판단하는 판단부, 판단결과에 따라 각 USB 장치 유형에 대한 운영체제 커널에 요청된 USB기기의 인터페이스 활성화 여부를 확인하는 인터페이스부 및 인터페이스의 활성화 여부에 따라 현재 연결의 안전성을 결정하는 제어부를 포함하여 USB 장치의 외관과는 다른 작업을 수행하는 USB 위장 공격을 방어할 수 있는 효과가 있다.

대 표 도 - 도1



(52) CPC특허분류
G06F 21/57 (2013.01)
G06F 2212/2146 (2013.01)

이 발명을 지원한 국가연구개발사업

과제고유번호	1711160654
과제번호	2022-0-01200-001
부처명	과학기술정보통신부
과제관리(전문)기관명	정보통신기획평가원
연구사업명	정보통신방송혁신인재양성
연구과제명	융합보안대학원(충남대학교)
기 여 율	1/1
과제수행기관명	충남대학교 산학협력단
연구기간	2022.01.01 ~ 2022.12.31

명 세 서

청구범위

청구항 1

USB 장치의 유형을 판단하는 판단부,

상기 판단부의 결과에 따라 각 USB 장치 유형에 대한 운영체제 커널에 요청된 USB기기의 인터페이스 활성화 여부를 확인하는 인터페이스부 및

상기 인터페이스 활성화 여부에 따라 현재 연결의 안전성을 결정하는 제어부를 포함하는 것을 특징으로 하는 모바일 기기의 USB 장치 인증 시스템.

청구항 2

제1항에 있어서,

상기 인터페이스부는 상기 USB 장치 유형별로 요구되는 보안수준에 따른 인공지능 기반 또는 유저기반 정책 설정 방식을 선택하도록 하는 선택모듈을 더 포함하는 것을 특징으로 하는 모바일 기기의 USB 장치 인증 시스템.

청구항 3

제1항에 있어서,

상기 제어부는 장치가 악성이 아닌 것으로 판단되면 모든 인터페이스에 해당하는 드라이버를 로드하는 것을 특징으로 하는 모바일 기기의 USB 장치 인증 시스템.

청구항 4

제1항에 있어서,

상기 제어부는 USB 기기가 호스트에 연결될 때마다 모든 인터페이스를 팝업하는 엄격 모드를 실행시키는 것을 특징으로 하는 모바일 기기의 USB 장치 인증 시스템.

청구항 5

제4항에 있어서,

상기 제어부는 상기 엄격 모드에서 사용자가 인터페이스의 일부를 선택적으로 활성화할 수 있도록 인터페이스 목록이 표시되도록 하는 것을 특징으로 하는 모바일 기기의 USB 장치 인증 시스템.

청구항 6

모바일 기기의 USB 장치 인증 시스템을 이용한 방법에 있어서,

(a) 상기 USB 장치 인증 시스템이 USB 장치의 유형을 판단하는 단계,

(b) 상기 USB 장치 인증 시스템이 상기 (a) 판단결과에 따라 각 USB 장치 유형에 대한 운영체제 커널에 요청된 USB기기의 인터페이스 활성화 여부를 확인하는 단계 및

(c) 상기 USB 장치 인증 시스템이 상기 인터페이스 활성화 여부에 따라 현재 연결의 안전성을 결정하는 단계를 포함하는 것을 특징으로 하는 모바일 기기의 USB 장치 인증 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 모바일 기기의 USB 장치 인증 시스템에 관한 것으로써, 보다 상세하게는 모바일 기기에 대한 USB 악성 장치를 활용한 공격을 방어하기 위한 모바일 기기의 USB 장치 인증 시스템 및 방법에 관한 것이다.

배 경 기 술

[0002] USB(Universal Serial Bus)가 발명된 이후 USB 프로토콜은 통신 속도와 범용성 확장 측면에서 지속적으로 개선되었다. 특히, 편리성과 사용성으로 주변기기를 바로 사용할 수 있게 해주는 PnP는 USB의 대중화와 지속적인 발전에 기여하는 주요 기능 중 하나이다. 그러나 USB는 불특정 다수의 장치를 맹목적으로 신뢰하는 방식으로 구현되어 호스트 공격에 악용되어 왔다. USB를 이용한 다양한 공격에는 악성코드 주입, 키 로깅, 키 주입, 도청, 데이터 유출 등이 있다. 이를 가능하게 하는 도구가 제시되었다. USB 공격 방법은 다음과 같이 세 가지 유형으로 분류할 수 있다.

[0003] 소프트웨어 취약점 악용: 이 공격은 USB 장치 드라이버 및 USB Core의 취약점을 악용하여 다른 커널 구성 요소를 공격하는 원시 공격으로 사용된다.

[0004] Protocol masquerading: 악의적인 USB 장치는 사용자에 의해 예기치 않게 동작한다. 예를 들어 USB 저장 장치처럼 보이지만 호스트 시스템에서 USB 키보드로 인식되어 키 입력을 주입한다.

[0005] 사회 공학 공격자는 사용자가 악성 USB 장치를 호스트 시스템에 연결하도록 유도한다. 그러면 사용자가 USB 장치에서 전달하는 악성코드를 직접 실행하거나 시스템의 자동 실행 기능(autorun.inf)이 자동으로 악성코드를 실행한다.

[0006] 사회 공학 공격은 (1)과 (2)의 기본 단계에서 구현될 수 있다. 즉, 호스트에 예기치 않은 동작을 수행하거나 소프트웨어 취약점을 악용하는 공격 페이로드를 전송하는 USB 장치를 사용자가 호스트에 연결하도록 한다. 이를 방어하기 위해 다음과 같은 몇 가지 조치가 제안된 바 있다.

[0007] 사용자에게 사회 공학 공격의 위험에 대해 교육, AutoRun 기능의 남용을 방지하기 위한 보안 설정, 무단 차단을 위한 내장 Windows 기능 실행 파일, USB 장치의 악성 코드를 탐지하는 바이러스 스캐너, 악성 코드를 차단하는 바이러스 백신 시스템 등이 있다. 또한 공격이 USB 소프트웨어 스택의 취약점을 악용하는 것을 방지하기 위해 일반적인 소프트웨어 보안 기술이 채택되었다. 예를 들어 퍼징과 가상화 기반 격리를 각각 재검토하여 USB 소프트웨어의 취약한 부분을 찾거나 격리할 수 있다.

[0008] 이 작업에서 다루는 주요 대상인 위장 공격은 방어하기 어려운 문제가 있다. 예를 들어, Rubber Ducky는 USB 저장 장치로 가장하여 키 입력을 주입할 수 있는 악성 USB 장치이다. 저렴한 가격에 공개구매가 가능하고, 키스트로크 주입을 위해 작성해야 하는 스크립트를 스크립트 키드가 쉽게 작성할 수 있다. 이러한 매스커레이드(위장) 공격은 사용자가 예상한 것과 다른 행동만을 할 뿐, 행동 자체가 커널에 정의된 표준 인터페이스를 기반으로 동작하기 때문에 소프트웨어 퍼징, 바이러스 스캐너 등의 대응책 활용이 불가능하다.

[0009] 화이트리스트를 사용하는 USB 인증 솔루션은 USB 프로토콜 위장 공격을 방어하는 데 널리 사용된다. Port-Blocker는 먼저 USB 포트를 잠근 다음 화이트리스트에 미리 등록된 장치만 허용한다. USB FILTER는 제품 및 제조업체와 같은 USB 장치 정보를 기반으로 호스트로 전송되는 USB 패킷의 필터링을 활성화한다. Cinch와 USBSec은 호스트와 장치 간의 통신 암호화를 통해 인증을 수행했다.

[0010] 반면, Mohammadmoradi et al. 상위 7개 USB 기능을 사용하여 화이트리스트 작성을 위한 고유 식별자를 생성할 것을 제안했다. DeviceVeil은 하드웨어 PUF(Physical Unclonable Function)를 사용하여 호스트 시스템에 USB 장치를 등록하고 인증하는 방법을 제안했다. 이러한 방어 기술은 다음과 같은 이유로 덜 실용적이다.

[0011] 기존 및 새로 출시된 USB 장치를 관리하기 위해 화이트리스트를 유지하려면 상당한 비용이 필요하다. 특히, 솔루션에 따라 화이트리스트 및 접근 제어 정책 설정을 위해 호스트에 먼저 USB 장치를 연결해야 한다.

[0012] 암호화 및 PUF 기반 인증의 경우 암호화 기능을 활성화하고, 하드웨어에서 강력한 식별자를 도출하기 위해 추가 하드웨어를 확보해야 한다.

[0013] 인증에 사용되는 장치 정보(예: 벤더 ID(VID) 및 제품 ID(PID))는 쉽게 변조될 수 있다. 이는 화이트리스트를 위한 식별자 정보가 악성 USB 장치 자체에서 제공되기 때문이다. 예를 들어 공격자는 호스트에 제공된 USB 설명자를 조작하여 장치가 특정 HID 장치인 것처럼 주장할 수 있다. 이러한 악성 장치가 호스트에 연결되면 장치에 내장된 악성 스크립트가 PnP에 의해 즉시 실행된다.

선행기술문헌

특허문헌

[0014] (특허문헌 0001) 공개특허 10-2019-0103801 (2019.09.05)

발명의 내용

해결하려는 과제

[0015] 본 발명은 모바일 기기에 대한 USB 악성 장치를 활용한 공격을 방어하기 위한 모바일 기기의 USB 장치 인증 시스템 및 방법을 제공함에 목적이 있다.

과제의 해결 수단

[0016] 상기 목적을 달성하기 위하여 본 발명은, 인공지능 기반으로 USB 장치의 유형을 판단하는 판단부, 상기 판단부의 결과에 따라 각 USB 장치 유형에 대한 운영체제 커널에 요청된 USB기기의 인터페이스 활성화 여부를 확인하는 인터페이스부 및 상기 인터페이스부의 활성화 여부에 따라 현재 연결의 안전성을 결정하는 제어부를 포함한다.

[0017] 한편, 모바일 기기의 USB 장치 인증 시스템을 이용한 방법에 있어서, (a) 상기 USB 장치 인증 시스템이 USB 장치의 유형을 판단하는 단계, (b) 상기 USB 장치 인증 시스템이 상기 (a) 판단결과에 따라 각 USB 장치 유형에 대한 운영체제 커널에 요청된 USB기기의 인터페이스 활성화 여부를 확인하는 단계 및 (c) 상기 USB 장치 인증 시스템이 상기 인터페이스 활성화 여부에 따라 현재 연결의 안전성을 결정하는 단계를 포함한다.

발명의 효과

[0018] 본 발명에 따르면, 사용자를 속여 USB 장치의 외관과는 다른 작업을 수행하는 공격(USB 위장 공격)을 방어할 수 있는 효과가 있다. 가령, 알려진 USB 공격 툴인 Rubber Ducky와 같이 USB 저장소가 키입력을 수행하는 공격을 방어할 수 있는 효과가 있다.

도면의 간단한 설명

[0019] 도 1은 USB Eye 개요의 사용 모델을 나타낸 것이다.

도 2는 USB Eye와 USB 장치 연결을 설명하기 위한 도면이다.

도 3은 폴링 기반 시스템 설계를 나타낸 것이다.

도 4는 USB 장치 연결 시간의 분석을 설명한다.

도 5는 본 발명의 일 실시예에 따른 모바일 기기의 USB 장치 인증 시스템의 구성을 나타낸 구성 블록도이다.

도 6은 본 발명의 일 실시예에 따른 모바일 기기의 USB 장치 인증 방법을 나타낸 개략적인 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0020] 이하, 첨부된 도면들에 기재된 내용들을 참조하여 본 발명을 상세히 설명한다. 다만, 본 발명이 예시적 실시 예들에 의해 제한되거나 한정되는 것은 아니다. 각 도면에 제시된 동일 참조부호는 실질적으로 동일한 기능을 수행하는 부재를 나타낸다.

[0021] 본 발명의 목적 및 효과는 하기의 설명에 의해서 자연스럽게 이해되거나 보다 분명해질 수 있으며, 하기의 기재만으로 본 발명의 목적 및 효과가 제한되는 것은 아니다. 또한, 본 발명을 설명함에 있어서 본 발명과 관련된 공지 기술에 대한 구체적인 설명이, 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우에는 그 상세한 설명을 생략하기로 한다.

[0022] 본 발명의 일 실시예에 따른 모바일 기기의 USB 장치 인증 시스템은 모바일 기기에 대한 USB 악성 장치를 활용한 공격을 방어하기 위한 USB 기기 인증을 위한 시스템이다.

[0023] 범용 직렬 버스(USB)의 보안은 편의성보다 우선 순위가 낮다. 예를 들어 USB 장치 연결은 플러그 앤 플레이(PnP)를 활성화하기 위해 맹목적으로 신뢰된다. 그러나 장치의 부적절한 인증은 특히, USB 장치를 가장하는 공

격을 유발할 수 있다. 이 문제를 해결하기 위해 화이트리스트 기반 USB 장치 인증 방식이 제안되었다. 그러나 적법한 장치 목록을 유지하는 데 상당한 노력과 비용이 든다는 점에서 실용성이 부족하다. 또한 잠재적으로 악의적인 장치가 인증에 필요한 정보를 제공하기 때문에 접근 방식을 쉽게 우회하여 이러한 접근 방식의 한계를 강조한다.

[0024] 이 문제를 완화하기 위해 CNN(Convolutional Neural Network) 지원 USB 장치인 USB Eye를 인증 솔루션으로 제안한다. 특히, CNN에서 추론한 장치 유형을 기반으로 정의된 USB 인터페이스의 적법성을 검증하여 가장한 USB 장치를 감지한다. 예를 들어 장치가 USB 저장소로 분류되지만 키보드 인터페이스를 정의하는 경우, USB Eye는 현재 호스트에 연결된 USB 장치가 악성이라고 판단한다.

[0025] 본 실시예에서, 이미지 분류를 수행하는 Android 애플리케이션과 악성 USB 장치 연결을 차단하는 Linux 커널의 USB 필터를 구현하여 접근 방식의 타당성을 입증했다. Rubber Duck을 사용한 보안 분석에서는 USB Eye가 프로토콜 위장 공격을 성공적으로 탐지하고 차단하는 것으로 나타났다.

[0026] 본 실시예에서는 화이트리스트 유지, 펌웨어 검증, 추가 하드웨어 도입 없이도 USB 프로토콜 매스커레이드 공격을 방어할 수 있는 USB Eye라는 솔루션을 제안한다. 모바일 장치에 연결된 USB 장치의 악의적이고 예기치 않은 인터페이스를 탐지하고 차단할 수 있다. 이를 위해 USB Eye는 CNN(Convolutional Neural Network)을 통해 USB 장치의 유형을 유추한다. 그런 다음 각 장치 유형에 대한 보안 정책에서 허용된 인터페이스에 대해 장치가 요청한 인터페이스를 확인하여 장치에 대한 현재 연결의 안전성을 결정한다.

[0027] 장치가 악성이 아닌 것으로 판단되면 USB Eye는 모든 인터페이스에 해당하는 드라이버를 로드한다. 안전하지 않은 경우 USB Eye는 모든 인터페이스 바인딩을 취소한다. 강력한 사용자 중심 검증을 위해 USB Eye는 새 USB 장치가 호스트에 연결될 때마다 항상 모든 인터페이스를 팝업하는 엄격한 모드를 제공한다.

[0028] 엄격 모드에서는 사용자가 인터페이스의 일부를 선택적으로 활성화할 수 있도록 인터페이스 목록이 표시된다. 장치가 안전하지 않은 것으로 판단되면 모든 인터페이스 정보와 함께 보안 경고가 팝업된다.

[0029] 본 실시예에서는 Cortex-A53 코어가 장착된 Juno 보드에 USB Eye를 구현하였다. Android 및 Linux 커널 버전은 각각 9.0.0 릴리스 33 및 4.14.59이다.

[0030] USB Eye는 USB 유형을 추론하기 위해 CNN을 실행하는 Android 애플리케이션, USB HID 이벤트 및 USB 장치의 인터페이스를 확인하기 위한 HID 및 USB Core에 대한 패치, 애플리케이션이 패치된 USB 코어와 통신하도록 돕는 Android 서비스로 구성된다.

[0031] Rubber Ducky를 사용하여 USB Eye의 보안성을 평가하였다. USB Eye는 Rubber Ducky의 숨겨진 키보드 인터페이스를 노출하여 마스커레이드 공격을 성공적으로 저지한다. 성능 분석에서 실험 결과는 우리의 보안 시스템이 USB 장치 연결에 대한 101%의 오버헤드와 USB 이벤트 처리에 대한 2%의 허용 가능한 성능 오버헤드를 발생시킨다는 것을 나타낸다. 요약하면 본 실시예는 다음과 같은 기여를 한다.

[0032] 화이트리스트나 추가 하드웨어 없이 실제 환경에서 적용하기 어려운 USB 프로토콜 위장을 효과적으로 차단하는 솔루션인 USB Eye를 제안한다.

[0033] USB 장치 유형을 추론하기 위한 신경망은 장치 연결에 대한 보안 결정을 자동화하여 빈번한 보안 경고의 단점을 최소화하는 데 활용된다.

[0034] USB Eye의 PoC가 Android를 실행하는 Juno ARM 참조 보드에서 구현 및 평가되었지만 기본 설계는 아키텍처 및 OS에 구애받지 않는다. 따라서 서버 및 PC 환경으로도 확장될 것이다.

[0035] USB 장치 설명자(USB Device Descriptor)에 대해 설명하면 다음과 같다.

[0036] USB 장치에는 구성, 인터페이스 및 엔드포인트와 같은 단일 장치 설명자와 하위 설명자가 포함된다. 특히 구성은 USB 장치의 논리적 기능인 인터페이스의 조합을 정의한다. 사용자가 마이크, 버튼 및 스피커에 대한 인터페이스를 제공하는 스피커 및 버튼 인터페이스로만 USB 헤드셋을 구성하면 해당 헤드셋에서 마이크가 비활성화된다. USB 장치가 호스트 장치 시스템에 연결되면 USB 코어는 구성의 각 인터페이스를 적절한 장치 드라이버에 바인딩한다.

[0037] 특히 각 드라이버는 USB 장치의 기능을 처리한다. BInterfaceClass, BInterfaceSubClass, BInterfaceProtocol 등의 정보가 포함된 인터페이스 디스크립터는 디바이스 디스크립터의 idVendor 및 idProduct와 함께 어떤 드라이버를 바인딩할지 결정하기 위해 참조된다.

[0038] Linux USB 시스템 및 USB 장치 등록에 대해 설명하면 다음과 같다.

[0039] Linux 시스템에서 USB Core는 Linux 호스트 시스템에서 USB 장치를 사용하기 위해 USB 하위 시스템 초기화, USB 버스 생성 및 등록, USB 데이터 구조 할당 및 관리, 설명자 관리, USB 연결 및 드라이버 바인딩을 수행한다. USB Core는 Hub 개념을 통해 모든 USB 장치를 관리한다. USB 장치는 물리적으로 연결될 때마다 시스템의 허브에 논리적으로 연결되어 관리되고 사용된다.

[0040] 특히, USB 장치가 연결되면 허브는 USB 장치를 버스에 등록하고 새 장치에 대한 USB 개체를 생성하여 허브 포트의 연결 이벤트를 처리한다. USB Core는 연결된 USB 장치를 쿼리하여 인터페이스 정보를 얻는다. 그러면 USB Core는 USB 장치에 정의된 기능을 수행할 수 있도록 USB 장치의 인터페이스를 각 드라이버에 바인딩하는 일련의 과정을 수행한다.

[0041] 안드로이드 서비스(Android Services)에 대해 설명하기로 한다.

[0042] Android 서비스는 사용자의 서비스 요구 사항에 따라 애플리케이션 개발에 필요한 중요한 API를 제공하는 데몬이다. 이러한 서비스는 JAVA 서비스와 C 또는 C++로 구현된 네이티브 서비스로 분류된다. NDK(Native Development Kit)는 개발자가 Android에서 C 및 C++ 코드를 개발하는 데 도움이 되는 도구 세트이며 JNI(Java Native Interface)는 JAVA와 네이티브 바이너리 간의 통신을 용이하게 하는 인터페이스이다. Android는 개발자가 Java에서 처리하기 어려운 시스템 호출 및 하드웨어 제어와 같은 기본 서비스를 쉽게 개발할 수 있도록 NDK 및 JNI를 제공한다.

[0043] USB에 대한 공격(Attacks Against USB)에 대해 설명하면 다음과 같다.

[0044] 여기에서는 USB 장치를 사용하는 공격 벡터에 대해 자세히 설명한다.

[0045] 소프트웨어 취약성, 프로토콜 가장, 사회 공학의 세 가지 유형으로 분류한다.

[0046] 소프트웨어 취약점 악용. 공격자는 공격 페이로드를 전달하는 악성 USB 장치를 사용하여 호스트의 USB 장치 드라이버 취약성을 악용할 수 있다. 장치 드라이버는 일반적으로 OS와 동일한 권한을 가지므로 장치 드라이버에 대한 공격이 성공하면 전체 시스템이 손상될 수 있다.

[0047] 프로토콜 위장. 예상치 못한 HID 인터페이스를 숨겨 악의적인 작업을 수행한다. USB 저장 장치의 펌웨어 변경을 가능하게 하는 BadUSB 및 Rubber Ducky는 마스커레이드 공격을 수행하는 도구의 예이다. 사용자는 외관상 정상적인 USB 저장 장치로 인식하지만 예상치 못한 인터페이스가 정의되어 있다. 마우스와 키보드로 HID 기능을 수행하여 호스트에 연결되면 악의적으로 행동할 수 있다. 이러한 숨겨진 HID 인터페이스는 표준 USB 프로토콜을 사용하여 작동하기 때문에 해당 작업은 호스트 시스템에서 합법적인 것으로 간주된다.

[0048] 사회 공학 공격. 이동식 미디어가 호스트에 연결되면 특정 작업을 자동으로 수행하는 AutoRun 기능에 사용할 수 있는 Autorun.inf가 악용될 수 있다.

[0049] 또한 사용자는 USB 장치에서 악성 소프트웨어를 실행하도록 만들 수 있다. 위에서 설명한 두 가지 공격을 실행하려면 악성 USB가 피해자의 호스트 장치에 물리적으로 연결되어 있어야 한다.

[0050] 공격자는 두 가지 공격의 전 단계로 사회 공학 기술을 악용하여 사용자가 USB 장치를 호스트 장치에 연결하도록 유도할 수 있다. 사회 공학 공격은 목표를 달성하기 위해 사람들 사이의 신뢰를 악용한다.

[0051] 다크넷(Darknet)에 대해 설명하면 다음과 같다.

[0052] Darknet은 C 및 CUDA로 작성된 오픈 소스 신경망 프레임워크이다. 사용자는 입력 이미지의 배치, 세분화, 너비 및 높이, 네트워크 아키텍처와 같은 변수를 정의하는 cfg 파일을 작성하여 모델 교육을 구성할 수 있다. 또한 데이터 파일을 사용하여 학습 데이터를 구성할 수 있다.

[0053] 훈련 결과 훈련된 모델을 정의하는 weight-file이 생성된다. 여기서 채택된 CNN 외에도 RNN(Recurrent Neural Network)과 GAN(Generative Adversarial Network)을 이용한 다양한 도구가 있다. 또한 가벼운 크기로 인해 프레임워크는 에지 장치에서 모델 보안을 강화하기 위해 신뢰할 수 있는 실행 환경(TEE)에서 호스팅되었다.

[0054] **본 실시예에 따른 모바일 기기의 USB 장치 인증 시스템의 위협모델을 설명하면 다음과 같다.**

[0055] 공격자는 사회 공학 공격을 수행하여 사용자가 악의적인 USB 장치를 연결하도록 유도할 수 있다. 악성 장치는 마스커레이드 공격을 수행하는 BadUSB 및 Rubber Ducky와 같은 도구를 말한다. 따라서 악의적인 도구는 예기치

않은 작업을 트리거하지만 표준 USB 프로토콜에 따라 수행되기 때문에 커널의 관점에서 작업은 여전히 정상이다. 이를 악용하여 공격자는 모바일 장치에서 민감한 데이터를 빼내거나 맬웨어 다운로드와 같은 추가 악의적인 작업을 수행하려고 시도한다. 그러나 본 실시예에서는 공격자가 피해자의 장치를 물리적으로 획득할 수 없다고 가정한다.

[0056] 즉, 공격자는 신호 도청, 전력 분석과 같이 디바이스에 대한 물리적 접근이 필요한 USB 버스의 신호 주입 공격을 수행할 수 없다. 또한 악성 USB 장치에서 악성 바이너리(스크립트)를 직접 실행하고 호스트 USB 소프트웨어 스택의 취약점을 악용하는 것은 여기에서 고려되지 않는다.

[0057] 도 1은 USB Eye 개요의 사용 모델을 나타낸 것이다.

[0058] 본 실시예에 따른 사용모델(USAGE MODEL)을 설명하면 다음과 같다.

[0059] 본 실시예에서는 먼저 제안하는 시스템의 초기화 방법을 설명한다. 그런 다음 USB Eye의 감독하에 USB 장치를 사용하는 절차를 사용자 관점에서 설명하기로 한다.

[0060] 초기화 단계. 신경망 모델(Darknet의 weight-file)은 대용량 저장소, 키보드, 마우스, 헤드셋 등 4가지 유형의 USB 장치 이미지를 사용하여 훈련하고 USB Eye가 있을 때 원격 서버에서 사용자에게 배포되는 USB 장치 유형을 유추한다.

[0061] 설치. 또한 helper_app이 실행될 때 비동기적으로 업데이트된다. 현재 보안 관리자가 모델을 생성했다고 가정한다. 그러나 연합 학습은 프라이버시 보호 모델 훈련에 채택될 수 있다.

[0062] 런타임 단계. USB Eye가 배포되고 활성화되면 본 실시예에 따른 시스템은 도 1과 같이 USB 연결 절차를 따라야 한다. 본 실시예에 따른 시스템은 먼저 helper_app를 실행하고 다음 단계를 진행한다.

[0063] (1) 연결할 USB 장치의 사진을 찍는다.

[0064] (2) Darknet이 백그라운드에서 시작된다. 사진을 분류하여 USB 장치 유형을 유추한다.

[0065] (3) 2단계의 텍스트 상자에서 Darknet에 의해 분류된 USB 유형을 확인하고 호스트에 USB 장치를 연결한다. 이후 USB 장치가 보안 정책에 대해 악성이 아닌 것으로 판단되면 사용자는 즉시 사용할 수 있다.

[0066] 안전하지 않은 경우(사용자가 인식하지 못하는 위장된 인터페이스 정의) USB 장치가 차단된다.

[0067] 오분류를 고려하여 USB Eye는 기본 및 엄격의 두 가지 작동 모드를 제공하며 이는 helper_app를 구성하여 전환할 수 있다. 기본 모드에서 USB Eye는 앞서 언급한 대로 자동 연결 및 차단으로 작동한다. 그러나 엄격 모드에서는 USB Eye가 무조건 USB 인터페이스 정보를 팝업하여 사용자가 USB 장치에 정의된 모든 인터페이스를 인식하고 원하는 인터페이스를 선택적으로 활성화할 수 있도록 한다.

[0068] 신경망에 의한 오분류 가능성에도 불구하고 잦은 팝업 창의 단점을 완화하기 때문에 여전히 사용하는 것이 유리하다. 특히 빈번한 팝업은 피로를 유발하고 사용자가 때때로 대화를 무시하도록 유도할 수 있다. 예를 들어, 팝업 창에 짜증이 난 사용자는 경고를 무시하고 "모두 연결" 버튼을 누르거나 팝업 창에 익숙해져 무심코 "모두 연결" 버튼을 누르는 경우가 있다.

[0069] 본 실시예에 따른 시스템 설계에 대해 설명하면 다음과 같다. 여기에서는 USB 장치가 어떻게 연결되어 있는지, USB 장치를 검증하기 위해 보안 정책이 어떻게 정의되고 참조되는지 등 USB Eye의 세부 사항에 대해 설명하기로 한다.

[0070] 본 실시예에 따른 USB Eye 개요를 설명하면 다음과 같다.

[0071] 본 실시예에서는 BadUSB 및 Rubber Ducky에 의해 쉽게 수행될 수 있는 마스커레이드 공격을 방어하는 USB Eye라는 보안 시스템을 제안한다.

[0072] 앞에서 설명한 바와 같이, USB 장치의 기능은 USB 프로토콜에서 인터페이스로 표현되며, 그 기능을 다루기 위해서는 장치 드라이버가 인터페이스에 올바르게 바인딩되어야 한다. 특히 가장 공격이 악의적인 작업을 수행하려면 각 드라이버가 프로토콜 가장에 대해 정의된 인터페이스에 바인딩되어야 한다. 따라서 Masquerade 공격을 차단하고 예상치 못한 인터페이스를 감지하는 가장 적절한 인스턴스는 드라이버를 바인딩하기 직전이다.

[0073] 본 실시예에 따른 USB Eye는 현재 연결을 확인하고 CNN에서 결정한 USB 장치 유형에 따라 악성 인터페이스를 감지한다. 검증은 관리자가 정의한 보안 정책을 특정 장치에 적합한 최소한의 일반 인터페이스로 설정하고 시행하

는 방식으로 이루어진다. 검증 결과와 선택한 모드에 따라 USB Eye는 연결된 장치를 자동으로 연결 또는 거부(기본 모드)하거나 사용자가 원하는 인터페이스만 연결할 수 있도록 모든 인터페이스 정보를 사용자에게 알려준다(strict 모드).

[0074] 또한 엄격 모드에서 USB 장치가 안전하지 않은 경우, 인터페이스와 함께 팝업에 경고가 포함된다. 인터페이스를 선택적으로 활성화하면 개인 정보에 민감한 USB IO 장치(마이크, 캠 등)의 일부 기능을 커널 수준에서 차단하여 악성 소프트웨어가 바인딩된 인터페이스를 악용하는 것을 방지할 수 있다. 사용자는 기본 모드에서 팝업 창을 줄이는 이점 또는 엄격 모드에서 선택적 연결 인터페이스의 이점을 적절하게 선택할 수 있다.

[0075] 도 2는 USB Eye와 USB 장치 연결을 설명하기 위한 도면이다. 도 2에 도시된 바와 같이, 다음과 같이 수행된다. (1) 본 실시예에 따른 모바일 기기의 USB 장치 인증 시스템은 USB 장치의 유형을 유추하고 업데이트한다. (2) 다음으로 모바일 기기의 USB 장치 인증 시스템은 USB 장치가 연결되어 있다. (3) 다음으로 모바일 기기의 USB 장치 인증 시스템은 장치 정보를 업데이트한다. (4) 다음으로 모바일 기기의 USB 장치 인증 시스템은 안드로이드 서비스를 실행한다. (5) 다음으로 모바일 기기의 USB 장치 인증 시스템은 장치 및 유추된 유형에 대한 정보를 요청하고 수신한다. (6) 다음으로 모바일 기기의 USB 장치 인증 시스템은 USB 장치를 확인한다. (7) 다음으로 모바일 기기의 USB 장치 인증 시스템은 경고 대화 상자를 만들고 표시한다. 이때, 기본 모드가 선택된 경우 (9)로 건너뜁니다. (8) 다음으로 모바일 기기의 USB 장치 인증 시스템은 외부 입력을 통해 사용자로부터 응답을 받는다. (9) 다음으로 모바일 기기의 USB 장치 인증 시스템은 연결 정보를 업데이트한다. (10) 다음으로 모바일 기기의 USB 장치 인증 시스템은 선택한 인터페이스에 드라이버를 바인딩한다.

[0076] 본 실시예에 따른 모바일 기기의 USB 장치 인증 시스템에서 USB Eye의 핵심 구성 요소에 대해 설명하면 다음과 같다.

[0077] USBEye는 다음 구성 요소로 구성된다.

[0078] ?? 일반적인 Linux 디바이스 드라이버인 USBEye 드라이버는 인터페이스와 같은 USB 디바이스의 데이터 구조를 USB Core와 공유한다.

[0079] ?? Android 서비스는 기본 모드에서 현재 연결의 안전성을 확인하여 선택한 USBEye 드라이버에 연결 옵션(예: Connect All 또는 Refuse All)을 보낸다. 엄격 모드에서 Android 서비스는 USB 인터페이스 정보를 인코딩하여 AlertDialog를 통해 표시하여 사용자에게 전달한다.

[0080] 사용자가 선택한 USB 연결 옵션은 USBEye 드라이버로 전송된다.

[0081] ?? helper_app은 USB 장치의 종류를 분류하고 그 결과를 USBEye 드라이버로 전송한다.

[0082] ?? NDK를 사용하면 네이티브 서비스(C 또는 C++)를 JAVA에서 처리할 수 있다. Android 서비스 및 helper_app의 일부는 NDK로 구현되어 OS 커널과의 통신과 같은 기본 작업을 지원한다. 특히, USB Eye 드라이버와 helpe_app 간의 데이터 공유는 NDK 서비스를 사용하여 수행된다.

[0083] USB 인터페이스 검증에 대해 설명하면 다음과 같다.

[0084] USB Eye의 검증 과정에서 사용자가 helper_app의 1단계와 2단계를 진행함에 따라 USB Eye는 Darknet에서 유추한 USB 유형을 USB Eye 드라이버에 전달한다. 사용자가 이를 건너뛰면 USB 연결을 진행할 수 없다. 사용자가 3단계를 진행하고 USB 장치가 호스트 장치의 USB 허브에 연결되면 USB Eye는 USB Core에서 USB Eye 드라이버로의 인터페이스와 같은 USB 장치 정보를 업데이트한다. 그런 다음 액티비티 관리자(am) 명령을 실행하여 Android 서비스를 실행한다. Android 서비스는 기본(NDK) 기능을 통해 USB Eye 드라이버에서 유추된 유형 및 USB 장치 정보를 가져온다. 안드로이드 서비스는 타입에 따라 디바이스에 정의될 것으로 예상되는 인터페이스와 실제로 정의된 USB 디바이스의 인터페이스를 비교하여 USB 디바이스와의 연결 안전성을 검증한다. 두 정보가 모두 일치하면 USB 장치는 위장된 인터페이스없이 합법적인 인터페이스만 정의하고 Android 서비스는 장치가 연결하기에 안전하다고 판단하고 "모든 인터페이스 연결"에 대한 연결 정보를 업데이트한다. 반대로 불일치는 USB 장치에 가장된 인터페이스가 포함되어 있음을 나타낸다. Android 서비스는 기기가 안전하지 않다고 판단하고 '모든 인터페이스를 거부'하도록 연결 정보를 업데이트한다.

[0085] Strict 모드를 선택하면 Android 서비스는 연결 정보를 직접 업데이트하는 대신 기기 정보로 AlertDialog를 생성하고 AlertDialog를 통해 기기에 정의된 모든 인터페이스를 사용자에게 보여준다. Android 서비스에서 기기가 안전하지 않다고 판단하면 AlertDialog에 경고가 포함된다. 사용자는 경고 메시지를 확인하고 AlertDialog에 응답하여 연결할 인터페이스를 선택하여 연결된 USB 장치의 안전을 확인할 수 있다. 이후 Android 서비스는 모든

인터페이스 연결, 모든 인터페이스 거부 또는 사용자가 선택한 인터페이스만 연결과 같은 연결 옵션을 USBEye 드라이버에 업데이트하고 옵션을 USB Core에서 읽는다. 마지막으로 USB Core는 옵션에 따라 현재 연결을 처리한다.

[0086] 기본적으로 악성 장치는 USB Eye에 의해 자동으로 차단되므로 사용자는 장치가 일부 악성 기능을 정의하고 있음을 인식할 수 있다. 엄격 모드에서 사용자는 USB 장치의 인터페이스 및 경고와 같이 AlertDialog에 표시된 정보를 기반으로 장치를 신뢰할 수 있는지 여부를 결정할 수 있다.

[0087] 또한 사용자가 엄격한 모드를 선택하고 선택적 연결 기능을 사용하면 사용자는 선택한 USB 장치 기능만 안전하게 사용할 수 있다. 예를 들어 사용자가 화상 회의 애플리케이션을 사용할 때 비디오 캠의 카메라 기능을 끄고자 한다고 가정하면 공격자에 의해 손상된 화상 회의 애플리케이션이 의도적으로 카메라를 켜거나 비디오 캠에서 민감한 데이터를 요청할 수 있다.

[0088] 이 경우 사용자가 엄격한 모드를 선택하고 원하는 인터페이스를 선택하면 선택되지 않은 인터페이스는 드라이버에 바인딩되지 않으므로 손상된 애플리케이션이 장치의 기능을 남용하는 것을 방지할 수 있다. 본 실시예에서는 공격 모델에서 OS를 신뢰하기 때문에 악성 애플리케이션이 권한을 상승시킬 수 있는 커널 익스플로잇을 고려하지 않는다.

[0089] AI 도입 효과에 대해 설명하면 다음과 같다.

[0090] 앞에서 USB 인터페이스 검증에 대해 설명한 바와 같이, 디폴트 모드에서 자동접속 또는 거부는 팝업창의 빈도를 줄일 수 있다. 엄격 모드에서는 장치 정보가 항상 팝업되어 사용자 응답을 받는다. 연결 규칙은 다음과 같이 요약할 수 있다.

[0091] R1. 기본 모드를 선택하고 USB 장치의 인터페이스가 보안 정책에 정의된 예상 인터페이스와 일치하면 USBEye에 의해 팝업창을 표시하지 않고 장치가 자동으로 연결된다.

[0092] R2. 기본 모드가 선택되고 USB 장치의 인터페이스가 예상 인터페이스와 일치하지 않으면 USBEye에서 팝업 창을 표시하지 않고 모든 연결이 자동으로 거부된다.

[0093] R3. 엄격 모드가 선택되고 USB 장치의 인터페이스가 예상 인터페이스와 일치하면 USBEye는 모든 인터페이스의 정보가 포함된 팝업창을 표시한다.

[0094] R4. 엄격 모드가 선택되고 USB 장치의 인터페이스가 예상 인터페이스와 일치하지 않으면 USBEye는 경고 메시지와 인터페이스 목록이 포함된 팝업창을 표시한다.

[0095] R1은 USB 장치가 안전하고 정의된 인터페이스 이외의 악의적인 작업을 수행할 수 없도록 한다. 예를 들어 helper_app는 장치 유형을 대용량 저장 장치로 분류하고 보안 정책에서 예상하는 장치의 인터페이스 세트도 대용량 저장 장치 유형(bInterfaceClass: 0x8)이다. 장치가 동일한 유추 인터페이스를 정의하는 경우 이 장치는 키 입력 삽입 및 데이터 저장 이외의 불법 기록과 같은 예기치 않은 동작을 수행하지 않을 수 있다. 따라서 USBEye는 사용자와의 상호 작용 없이 USB 장치를 연결한다. 인터페이스가 유형과 일치하지 않는 R2의 경우 USB 장치가 예기치 않은 동작을 수행할 수 있다. **결과적으로 USBEye는 장치가 잠재적으로 악의적이라고 판단하고 호스트에 대한 연결을 차단한다.**

[0096] R1, R2는 USB 장치 연결에 사용자 개입이 필요 없기 때문에 잦은 팝업에 대한 사용자의 부담을 덜어준다.

[0097] R3는 엄격 모드에서 USB 장치의 유형이 예상 인터페이스와 일치하는 상황을 고려한다. 예를 들어, 관리자는 USB 헤드셋이 4개의 인터페이스를 정의해야 하는 보안 정책을 구성한다. 이는 bInterfaceClass: 0x1(Audio) 및 bInterface-Subclass: 0x1(Control Device)로 표시되는 제어 장치용 인터페이스 1개, 스트리밍 입력용 인터페이스 2개 및 bInterfaceClass: 0x1(Audio) 및 bInterfaceSubclass: 0x2(Streaming)에 의해 제시된 출력 및 bInterface-Class: 0x3(HID), bInterfaceSubclass:0x0(None) 및 bInterfaceProtocol에 의해 제시된 볼륨을 제어하기 위한 마지막 HID 인터페이스: 0x0(없음)가 있다. 마지막 인터페이스는 해당 장치 드라이버가 바인딩된 후 런타임에 HID 이벤트의 유효성을 검사해야 하는 예외적인 경우이다. 이에 대해서는 HID 이벤트 검증에서 더 논의하기로 한다. USB 장치에 의해 정의된 인터페이스는 보안 정책을 충족하기 때문에 USB Eye는 현재 연결이 안전한 것으로 간주하지만 사용자가 인터페이스의 일부를 선택적으로 활성화할 수 있도록 모든 인터페이스 목록이 포함된 팝업을 계속 표시한다. R4에서는 USB 장치가 마스커레이드 공격을 수행하는 것으로 의심되기 때문에 팝업에 경고 메시지가 추가되어 사용자에게 인터페이스를 신중하게 선택하도록 경고한다.

[0098] 요약하면 R3와 R4는 R1과 R2와 달리 USB 연결을 위해 항상 사용자 개입이 필요하다. 잦은 팝업은 보안시스템의 효율성을 떨어뜨린다. 그러나 인터페이스를 선택적으로 활성화하는데 여전히 이점이 있기 때문에 이 옵션을 계속 고려한다. PoC에서 사용자는 장치를 연결하기 전에 helper_app를 통해 기본 모드와 엄격 모드 사이를 전환한다.

[0099] HID 이벤트 검증에 대해 설명하면 다음과 같다.

[0100] 장치 제조업체에서 제공하는 특정 장치 드라이버 외에도 OS에 번들로 제공되는 일반 장치 드라이버도 HID 이벤트 처리에 사용된다. 특히 usb_match_id 함수에서 적절한 장치 드라이버를 일치시키기 위한 충분한 정보를 제공하지 않는 USB HID 인터페이스는 HID 일반 드라이버를 바인딩하여 처리한다.

[0101] HID 제네릭 드라이버는 키보드, 마우스, 터치패드와 같은 일반 HID 장치의 이벤트를 종합적으로 처리하기 위해 흔히 사용된다. HID 제네릭 드라이버가 바인딩된 HID 인터페이스에서 생성된 입력 이벤트는 HID 코어라는 입력 코어를 통해 입력 장치 드라이버로 전달된다. 입력 장치 드라이버는 HID 장치에서 받은 입력 이벤트의 유형을 구문 분석하고 확인한다. 그런 다음 이벤트가 최종적으로 이벤트 드라이버에 전달되고 그에 따라 처리된다.

[0102] HID 제네릭 드라이버를 바인딩해야 하는 인터페이스 중 일부 인터페이스는 HID 장치의 유형을 지정하지 않는다[bInterfaceClass: 0x3(HID), bInterfaceSubclass: 0x0(None) 및 bInterfaceProtocol: 0x0(None)]. 특히 인터페이스에서 사용하는 HID 기능은 드라이버가 열거될 때 지정되지 않는다. 따라서 이 인터페이스에 HID 제네릭 드라이버를 바인딩한 후 드라이버 내에서 어떤 종류의 HID 이벤트가 발생하는지 확인해야 한다. 예를 들어, USB 헤드셋의 경우 오디오 버튼 제어(볼륨 업, 볼륨 다운, 마이크 음소거)를 위한 인터페이스 0x3, 0x0, 0x0을 정의할 수 있다. HID 제네릭 드라이버는 키보드, 마우스와 같은 모든 HID 이벤트를 처리하기 때문에 이 인터페이스가 악의적인 목적으로 사용되는 것은 아닌지 고려해야 한다. 따라서 USB Eye는 바인딩 후 HID 이벤트를 확인하기 위해 HID 코어에 확인 로직을 추가했다. 안전하지 않은 이벤트는 USB 장치의 종류에 따라 차단되며 예제 확인 규칙은 다음과 같다.

[0103] ?? USB 헤드셋 및 USB 스피커와 같은 USB 오디오 장치 유형은 볼륨 높이기, 볼륨 낮추기, 음소거 및 마이크 켜기/끄기의 네 가지 이벤트를 제외한 모든 키보드 이벤트를 거부한다.

[0104] ?? USB 오디오를 제외한 USB가 아닌 키보드 장치 유형은 모든 키보드 이벤트를 거부한다.

[0105] ?? USB 마우스 이외의 모든 장치 유형은 모든 마우스 이벤트를 거부한다.

[0106] ?? 모든 HID 터치패드 장치 이벤트를 거부한다.

[0107] 이 검증 방식은 안정성을 위해 '모두 거부, 일부 허용' 정책을 구현하여 필요한 이벤트만 허용한다. 첫 번째 규칙과 두 번째 규칙을 기반으로 필수 키보드 이벤트 코드(스캔 코드)를 제외한 모든 악성 키보드 이벤트(키 입력)를 차단할 수 있다. 세 번째와 네 번째 규칙은 악의적인 마우스 및 터치패드 이벤트를 차단할 수 있다. 이러한 규칙 세트는 예시이며 관리자는 조직 및 IT 인프라에 적합한 자체 규칙을 설정할 수 있다. (예를 들어, 관리자는 마우스 자체 인터페이스와 키보드 인터페이스를 허용하여 보안 정책을 구성한다.)

[0108] 본 발명의 일 실시예에 따른 모바일 기기의 USB 장치 인증 시스템의 구현에 대해 설명하면 다음과 같다.

[0109] USBEye는 Cortex-A72 및 A53 CPU와 8GB DDR3 RAM 메모리가 있는 ARM Juno r2 개발 보드에서 구현되었다. 소프트웨어 플랫폼으로 Android(9.0.0)와 Linux 커널(4.14.59)을 실행하였다.

[0110] 본 실시예에 따른 모바일 기기의 USB 장치 인증 시스템의 커널 구성 요소에 대해 설명하기로 한다.

[0111] Linux에서 USB 장치 등록. 시스템이 부팅되면 USB 코어는 USB 하위 시스템과 관련된 구성 요소를 초기화하고 USB 버스를 생성한다. Linux에서 허브는 연결된 모든 USB 장치를 논리적으로 관리한다. 새 장치가 호스트에 연결될 때마다 허브 이벤트는 연결된 각 장치에 대한 데이터 구조를 만들고 삭제하는 kthread에 의해 처리된다. 연결 이벤트 처리의 일부로 device_add 함수가 여러 번 호출되어 각각 새 USB 장치를 등록하고 적절한 드라이버를 인터페이스에 바인딩한다. 특히, 등록 후 코어는 디바이스와 통신하여 VID, PID, 인터페이스 등의 정보를 획득한다. 마지막으로 set_configuration의 device_add를 호출하여 장치 드라이버를 인터페이스에 바인딩한다. 이 전체 절차를 USB 열거라고 한다.

[0112] 커널 패치. USB Eye는 악성으로 판단되는 경우, 현재 연결된 USB 장치의 인터페이스 정보를 보여준다. 리눅스 커널에서는 USB 열거(enumeration)시 앞서 언급한 바와 같이, USB core의 set_configuration 함수에서 인터페이스 바인딩을 위한 device_add 함수가 호출되기 전에 디바이스 등록 및 정보 획득이 이루어진다. 따라서

helper_app에 인터페이스 정보를 전달하고 사용자의 응답을 받을 수 있도록 USB 코어 소스(linux/drivers/usb/core/message.c)를 업데이트한다.

[0113] 특히, Android 서비스 실행을 목표로 하는 NDK 바이너리를 시작하기 위해 채택된 call_usermodeheloper API는 적절한 장치 드라이버를 바인딩하기 위해 모든 인터페이스를 열거하는 for 루프 앞에 삽입된다. 개별 인터페이스를 (비)활성화하기 위한 사용자의 결정을 반영하는 논리가 루프에 추가된다. USB 코어 외에도 HID 이벤트의 세밀한 제어를 위해 HID 코어가 강화되었다. 불법 HID 이벤트는 hid-core의 hid_input_field 함수에서 이벤트 코드를 확인하여 차단한다.

[0114] USB아이 드라이버. 커널 패치 외에도 USB 코어가 사용자 수준 구성 요소(예: helper_app, Darknet 및 Android 서비스)와 통신하는 데 도움이 되는 USBEye 커널 드라이버를 구현했다.

[0115] 사용자와 커널 간에 데이터를 공유하기 위한 읽기, 쓰기 및 ioctl 처리기를 제공하는 간단한 문자 장치 드라이버이다. 유추된 USB 유형, 요청된 USB 인터페이스, 사용자의 인터페이스 및 동기화 활성화 결정에 대한 플래그 제공과 같은 데이터 공유를 위해 여러 데이터 구조가 개발되었다.

[0116] 사용자 구성 요소에 대해 설명하면 다음과 같다.

[0117] 안드로이드 서비스. 현재 디스플레이에 AlertDialog와 같은 UI 구성 요소를 오버레이하려면 서비스에 시스템 권한이 필요하다. Android 프레임워크를 빌드할 때 권한 있는 애플리케이션으로 빌드하여 Android 서비스에 시스템 권한을 부여한다.

[0118] 네이티브 개발 키트(NDK). 본 실시예에서는 NDK를 사용하여 Java 코드에서 직접 지원할 수 없는 Android 서비스 및 helper_app의 기본 기능을 수행했다. NDK는 Android 서비스 및 애플리케이션의 일부로 JNI를 통해 기본 서비스를 제공한다. Darknet은 NDK를 사용하여 Android 애플리케이션에서 호출할 수 있는 기본 서비스로 포팅되었다.

[0119] 폴링 기반 디자인(Polling-based Design)에 대해 설명하기로 한다.

[0120] 새로운 USB 장치를 연결하는 이벤트를 처리하기 위해 백그라운드에서 주기적으로 실행되는 Android 시스템 서비스를 사용한 폴링 기반 설계를 고려할 수 있다. 서비스는 UI 관리 및 사용자와의 커뮤니케이션도 담당한다. 현재 설계와 달리 이 접근 방식은 성능 평가에서 관찰된 가장 높은 오버헤드를 발생시키는 USB 포트 이벤트와 동시에 새 프로세스(즉, am 서비스)를 시작할 필요가 없다.

[0121] 또한, 이러한 이벤트 트리거 프로세스 생성이 면제되어 USB 코어를 변경할 필요가 없다. 그러나 전력 소모를 고려할 때 폴링 기반 접근 방식은 모바일 디바이스 환경에 적합하지 않을 수 있다. 또한 성능 평가에서 보듯이 폴링 기반 설계는 제안된 설계보다 추가적인 CPU 사용량과 부하를 발생시켰다.

[0122] 본 실시예에 따른 모바일 기기의 USB 장치 인증 시스템에서 평가를 설명하면 다음과 같다.

[0123] Darknet의 모델 및 데이터 세트(Model and Dataset of Darknet)와 관련하여 설명하기로 한다.

[0124] cfg 파일에서 배치 16, 학습 지연 0.1 및 7개의 컨볼루션 레이어와 3개의 최대 풀링 레이어가 있는 신경망 아키텍처를 설정한다. 컨볼루션 레이어는 다음과 같이 구성된다. Activation is leakey, 크기는 [3, 1, 3, 1, 3, 1, 1], 필터는 [16, 8, 32, 16, 64, 32, 10]이다. 또한 Activation이 선형이고 출력이 4개인 연결 계층을 구성하여 USB 장치를 4가지 유형 중 하나로 분류한다. USB 키보드, USB 마우스, USB 대용량 저장 장치 및 USB 헤드셋의 4가지 유형의 USB 장치에 대해 각각 약 700개의 이미지로 훈련을 위해 3K 이미지가 있는 데이터 세트를 사용한다.

[0125] 도 3은 폴링 기반 시스템 설계를 나타낸 것이다. USB Eye의 각 구성 요소는 다음과 같이 작동한다. (1) 본 실시예에 따른 모바일 기기의 USB 장치 인증 시스템이 USB 장치의 유형을 유추하고 업데이트한다.(이하, 모바일 기기의 USB 장치 인증 시스템에서 수행함) (2) USB 장치가 연결되어 있다. (3) 장치 정보를 업데이트한다. (4) 장치 및 예상 유형에 대한 정보를 요청하고 수신한다. (5) USB 장치를 확인한다. (6) 경고 대화 상자를 만들고 표시한다. 기본 모드가 선택된 경우 (8)로 건너뛴다. (7) 사용자로부터 응답을 받는다. (8) 연결 정보를 업데이트한다. (9) 선택한 인터페이스에 드라이버를 바인딩한다.

[0126] 보안 분석에 대해 설명하면 다음과 같다.

[0127] 본 실시예에 따른 모바일 기기의 USB 장치 인증 시스템은 Rubber Ducky를 사용하여 USB Eye의 보안성을 평가하

였다. 성능 평가를 위해 4가지 유형의 USB 장치로 설정한 보안 정책은 다음과 같다.

[0128] (표 1)

USB keyboard (2)	HID_*_Keyboard HID_None_None
USB mouse (2)	HID_*_Mouse HID_None_None
USB mass storage (2)	MassStorage_*_* HID_None_None
USB headset (4)	Audio_ControlDevice_* Audio_Streaming_* (input) Audio_Streaming_* (output) HID_None_None

[0129]

[0130] 표 1은 각 장치에 대해 허용되는 인터페이스를 정의하는 보안 정책의 예를 나타낸 것이다.

[0131] 보안 정책은 표준 USB 프로토콜 설명자, bInterfaceClass, bInterface-SubClass 및 bInterfaceProtocol에 해당하는 값으로 구성되며 인터페이스를 다른 인터페이스와 구별하기 위해 참조할 수 있다. 표 1의 별표는 모든 하위 유형이 허용됨을 나타낸다(와일드카드). 각 유형의 장치에 대한 보안 정책은 최소한의 필수 인터페이스만 허용한다.

[0132] 또한, 이 정책은 모든 유형의 USB 장치에 대해 interface:HID_None_None을 허용한다. 앞서 AI 도입 효과에서 언급했듯이 이러한 유형의 인터페이스는 해당 장치 드라이버를 바인딩한 후 확인해야 한다. 보안 정책에 정의되지 않은 모든 인터페이스는 악의적인 것으로 간주된다. 예를 들어 USBEye는 Rubber Ducky의 유형을 USB 저장 장치로 유추했지만 장치에서 요청한 인터페이스는 키보드이다. 따라서 Rubber Ducky에서 요청한 인터페이스는 정책을 위반한다. 따라서 장치는 악의적이다.

[0133] 결과적으로 선택한 모드에 따라 USBEye는 Rubber Ducky 연결을 거부하거나 경고 메시지와 함께 모든 인터페이스를 전달하는 대화 상자를 팝업한다. 사용자는 의심스러운 인터페이스가 있는지 확인하고 차단할 수 있다.

[0134] (표 2)

	Vendor	Product	Interfaces
USB Keyboard	Logitech	K120	1) HID Keyboard 2) HID None
USB Mouse	Logitech	G25	1) HID Mouse 2) HID None
USB Mass storage	Sandisk	SDCZ73-016G-G46 (Ultra flair USB 3.0 16GB)	1) Mass Storage
USB Headset	Plantronics	PLNAUDIO478	1) Audio Control 2) Audio Streaming IN 3) Audio Streaming OUT 4) HID None
USB Speaker	Cosy	SP1314MK	1) Audio Control 2) Audio Streaming IN 3) Audio Streaming OUT 4) HID None

[0135]

[0136] 표 2는 보안 분석에 사용되는 일반 USB 장치를 나타낸 것이다.

[0137] 또한 HID 일반 드라이버에서 처리하는 HID 이벤트의 검증을 테스트했다. 러버덕키는 펌웨어 수정을 통해 VID와 PID를 조작하여 원하는 드라이버에 바인딩할 수 있다. USB 장치 설명자에서 살펴본 바와 같이, VID와 PID는 어떤 드라이버를 바인딩할지 결정하는 중요한 정보이다. HID 일반 드라이버에 바인딩된 인터페이스를 사용하여 키 입력을 주입하는 악성 헤드셋을 에뮬레이션하기 위해 인터페이스가 일반 드라이버에 바인딩되도록 Rubber Ducky 펌웨어의 VID 및 PID를 변경한다.

[0138] 또한, 본 실시예에서는 Darknet에서 유추한 유형(즉, 저장 공간)을 무시하고 USB 유형을 헤드셋으로 강제 설정한다. 이에 HID 코어의 확인 코드가 공격을 성공적으로 감지했다. USB 헤드셋의 키 입력 이벤트가 차단되었다.

[0139] 표 2의 비악성 USB 장치에 대한 실험도 수행되었다. 장치는 보안 정책을 만족하는 일반적인 인터페이스만 정의하기 때문에 기본 모드에서 호스트에 원활하게 연결되었다. Strict 모드에서는 lsusb 결과와 비교하여 모든 인터페이스가 팝업창에 정상적으로 출력되는 것을 확인하였다. 오디오 제어 인터페이스는 USB Eye가 있는 경우에도 악의적인 USB 장치에 의해 남용될 수 있는 마이크와 같은 일부 하위 기능(즉, wTerminalType)을 정의한다. 아래에서 이 공격 표면에 대해 자세히 설명하기로 한다.

[0140] 본 실시예에 따른 모바일 기기의 USB 장치 인증 시스템의 성능 평가에 대해 설명하면 다음과 같다.

[0141] USB 장치 연결. 팝업 창을 열려면 NDK 바이너리 실행, Android 서비스 실행, 팝업 메시지 생성 및 표시로 인해 오버헤드가 발생한다. USB 장치를 꽂은 시간부터 연결이 완료될 때까지의 경과 시간을 측정한다. 사용자 응답 시간이 변동하여 값을 안정적으로 측정하기 어려운 경우 측정 시간에 포함되지 않는다. 도 4는 USB 장치 연결 시간의 분석을 설명한다. x축은 각 연결 절차를 설명한다. y축은 USBEye 100회 실행의 평균 대기 시간을 나타낸다. 'am startservice' 명령을 호출하여 Android 서비스를 실행하는 시간이 오버헤드의 대부분을 차지했다(도 4). 100%에 가까운 오버헤드가 관찰되긴 했지만, USB 장치를 꽂았을 때 한 번만 발생하기 때문에 사용성을 크게 떨어뜨리지는 않는다.

[0142] 도 4에서, USB 연결 오버헤드. x축의 각 숫자는 다음을 나타냅니다. (1) 본 실시예에 따른 모바일 기기의 USB 장치 인증 시스템은 변수를 초기화한다. (2) 장치 정보를 USBEye 드라이버로 업데이트한다. (3) call_usermodehelper를 호출하여 Android 서비스 실행을 위한 NDK를 실행한다. (4) 액티비티 매니저(am) 명령어를 호출하여 안드로이드 서비스를 시작한다. (5) USB Eye 드라이버에서 정보를 얻습니다. (6) USB 장치를 확인한다. (7) 선택한 모드를 확인한다. (8) 경고 대화 상자를 만들고 표시한다. (9) 연결 정보를 보낸다. (10) USBEye 드라이버에서 연결 정보를 받는다. (11) Android 서비스 프로세스를 종료한다. (12) 선택한 인터페이스에 대한 드라이버 바인딩. (13) 변수를 초기화한다.

[0143] (표 3)

USB connection time(μs)	
Baseline	USBEye
503272	1010153 (2.01×)

[0144]

[0145] 표 3은 USB 장치 연결 오버헤드를 나타낸 것이다.

[0146] (표 4)

Keyboard (μs)		Mouse (μs)	
Baseline	USBEye	Baseline	USBEye
116.538	117.998 (1.01×)	37.761	38.847 (1.02×)

[0147]

[0148] 표 4는 USB 장치 런타임 오버헤드를 나타낸 것이다.

[0149] USB HID 장치. HID 제네릭 드라이버에 바인딩된 인터페이스에 의해 발생한 이벤트는 USBEye(5.5절)에서 검증 후 처리된다. HID 이벤트 생성부터 이벤트 처리까지 걸리는 시간을 측정했다. 표 4는 베이스라인과 USBEye의 키보드 입력 및 마우스 입력 이벤트의 1000배 평균 대기 시간을 설명한다. 오버 헤드는 합리적으로 작았다. 마우스 클릭 이벤트를 처리할 때 최대 2%의 오버헤드가 관찰되었다.

[0150] (표 5)

CPU usage (%)		CPU load (#processes)	
USBEye	Polling	USBEye	Polling
0.9895	1.0220 (1.03×)	0.0887	0.0940 (1.06×)

[0151]

[0152] 표 5는 CPU 사용량은 비유휴 CPU 시간 비율(%)을 나타낸다. CPU 부하는 Linux 가동 시간 유틸리티로 측정된다.

[0153] (표 6)

[0154]

Training time (sec)	
Desktop	Juno Board
178	1184 (6.7×)

[0155] 표 6은 Darknet 훈련 시간. 훈련 결과 USB 장치를 분류하기 위한 매개변수를 정의한 가중치 파일이 생성된다.

[0156] CPU 사용량 및 로드. 우리는 평균 10시간의 CPU 사용 시간(해당 CPU에서 작업을 실행하는 시간)과 CPU 부하(CPU를 사용하고 있는 프로세스 또는 CPU를 사용할 준비가 된 대기열에 있는 프로세스 수)를 제안된 디자인과 폴링 기반 모두에 대해 측정했다. 설계(폴링 기반 디자인). CPU 사용량은 비유휴 CPU 시간의 비율을 나타낸다. CPU 부하를 측정하기 위해 1분, 5분, 15분 간격으로 평균 부하를 산출하는 Linux 가동 시간 유틸리티를 활용했다.

[0157] 여기서, 유휴 시간 비율은 다음과 같이 측정된다. (idle*100) / (user + nice + system + idle + iowait + irq + softirq + steal + guest + guest_nice), 각 항목은 /proc/stat 파일에서 검색된다.

[0158] 특히, 15분 간격으로 측정된 하중을 사용했다. 본 실시예에서는 폴링 기반 설계가 USB Eye에 비해 CPU 사용량과 부하에서 표 5를 통해 각각 3%와 6%의 오버헤드를 유발하는 것을 관찰했다. 이는 폴링 기반 설계가 추가 백그라운드 서비스를 실행하여 주기적으로 USB 연결을 확인하기 때문이다.

[0159] 다크넷 모델 트레이닝 시간(Darknet Model Training Time)에 대해 설명하면 다음과 같다.

[0160] USBEye의 현재 설계에서는 관리자가 CNN 모델을 교육하고 교육된 모델이 포함된 계량 파일을 각 사용자에게 배포해야 한다. 그러나 개인 사용자도 교육을 수행할 수 있다. 또한, 각 사용자의 개인 정보 보호를 고려하여 개별 모바일 장치에서 모델을 관리할 수 있는 연합 학습을 채택할 수 있다. 따라서 데스크톱과 모바일 장치 모두에서 Darknet을 교육하는 시간을 측정했다. 인텔 코어 i7-9700kf CPU 및 16GB DDR4 RAM 메모리가 장착된 데스크탑을 사용하여 USB 장치 유형을 교육했다.

[0161] 모바일 기기에서의 훈련은 앞에서 설명한 Juno 보드를 사용하였다. 두 경우 모두 Darknet의 Makefile에서 GPU=0으로 설정하여 훈련에 GPU를 사용하지 않았다. 그 결과를 표 6에 나타낸다. 교육 시간은 데스크톱에서 178초이지만 Juno 보드에서는 1184초(6.7x)이다. 본 실시예는 초기 교육 시간을 측정하며, 에지 장치에서 모델을 직접 교육하는 것은 네트워크 중단과 같은 극단적인 시나리오를 제외하고 극히 드물다. 연합 학습이 채택되더라도 에지 장치가 모델 매개변수를 조정하고 초기 모델은 여전히 중앙 서버(관리자)에 의해 배포된다.

[0162] 본 발명의 일 실시예에 따른 모바일 기기의 USB 장치 인증 시스템에 대해 논의해보면, 데스크톱 환경을 위한 확장이다. 본 실시예는 USBEye가 최소한의 엔지니어링 노력으로 PC 및 데스크탑 환경으로 쉽게 확장될 것으로 기대된다.

[0163] 예를 들어 안드로이드 서비스 대신에 팝업 창을 관리하기 위한 사용자 데몬을 구현할 수 있다. 또한, USB 장치의 사진을 찍고 이미지 분류를 위해 신경망을 실행하는 사용자 애플리케이션은 helper_app과 유사하게 구현될 수 있다.

[0164] 또한, 대상 호스트가 Linux OS를 실행하는 경우 추가 변경 없이 커널 구성 요소를 채택할 수 있다.

[0165] 필터링 인터페이스의 정확도. 관리자는 보안 정책을 가장 보수적으로 설정해야 한다. 안타깝게도 이로 인해 제조업체가 합법적으로 인터페이스를 정의하더라도 인터페이스가 차단될 수 있다. 기기 이미지를 잘못 분류하면 합법적인 인터페이스가 차단될 수도 있다. 일반적인 USB 장치와 생김새가 다른 새로운 장치를 다룰 경우 분류의 정확도가 떨어질 수 있다. 예를 들어 USB 장치의 크기를 줄여 휴대성을 높이거나 일부 장치는 인체공학적 디자인을 위해 고유한 모양을 가질 수 있다.

[0166] 또한, 일부 장치에는 일반적으로 특정 유형의 장치(예: 마우스, 키보드 및 터치패드 인터페이스가 있는 USB 스마트 컨트롤러)에 내장된 다양한 인터페이스가 있다.

[0167] 인터페이스를 차단하는 이유와 관계없이 USB Eye의 현재 설계는 USB Eye를 엄격 모드로 전환하여 이러한 번거로운 이벤트를 처리할 수 있으며, 이는 사용자가 인터페이스를 선택적으로 활성화하는 방법이다. AI 모델의 정확도를 높이는 것은 직교 문제이며, 정확도가 높은 AI 모델을 채택하고 집중 훈련을 통해 USBEye의 효율성이 높아

질 것으로 기대된다.

[0168] 개인 정보 보호. 신경망의 정확성 외에도 AI 시스템의 개인 정보 보호 문제도 USBEye에 계승된다. 예를 들어 USBEye의 이미지 분류 정확도를 높이려면 사용자가 새 장치의 이미지를 관리자에게 보내야 한다. 이로 인해 잠재적으로 악의적이거나 악의적이지 않지만, 호기심이 많은 관리자에게 각 사용자의 장치가 노출되는 개인 정보 보호 문제가 발생할 수 있다. 이 문제를 해결하기 위해 사용자의 개인 정보를 보호하면서 모델 정확도를 관리하는 연합 학습을 채택하는 것을 고려할 수 있다.

[0169] 사용자 인증을 위한 기계 학습. 물리적 공격은 공격 모델에서 제외되지만 일부 물리적 액세스 기반 USB 공격은 USBEye를 강화하여 여전히 방해할 수 있다. 예를 들어 공격자가 일시적으로 호스트 장치에 대한 짧은 물리적 액세스 권한을 갖고 있다고 가정할 수 있다. 이는 호스트 장치의 소유자에게 알리지 않고 악의적인 USB 장치를 연결할 수 있는 충분한 시간이다. 이 공격을 물리치기 위해 신경망을 훈련시켜 호스트 장치의 소유자를 다른 사람과 구별할 수 있다. AI 기반 사용자 인증을 USBEye에 통합하는 방법을 탐색하는 것은 향후 작업을 위해 따로 설정된다.

[0170] 세분화된 모니터링. 현재 USBEye는 인터페이스의 거친 필터링을 수행하는 것으로 제한된다. 따라서 합법적인 인터페이스의 하위 기능을 남용하는 공격자는 USB 프로토콜을 위장하지 않고 USBEye를 우회할 수 있다. 예를 들어, 오디오 제어 인터페이스에 정의된 마이크는 사용자의 의도 없이 손상된 USB 헤드셋에 의해 은밀하게 활성화될 수 있다.

[0171] 사용자와 손상된 장치 간에 장치 이벤트의 출처를 구별할 수 있는 세분화된 모니터링이 문제를 해결할 것으로 예상된다. 또한 모니터링을 활성화하기 위해 여러 제조업체에서 제공하는 장치 드라이버를 강화하는 효율적인 방법을 모색할 수 있다.

[0172] USB 프로토콜은 장치를 맹목적으로 신뢰하므로 사용자의 추가 설정 없이 주변 장치를 즉시 사용할 수 있다. 따라서 USB Core 또는 USB 드라이버의 취약점을 악용하는 소프트웨어 취약점 공격, 숨겨진 인터페이스를 통해 악의적인 동작을 수행하는 가장 공격, 인간의 신뢰를 악용하여 악의적인 USB 장치를 연결하는 사회 공학 공격이 발생했다. 이러한 공격을 방지하기 위해 다양한 하드웨어나 소프트웨어 도구의 개발과 함께 학계에서 다양한 방어기법이 연구되고 있으나, 추가적인 하드웨어와 비용이 요구되어 실용성이 떨어진다.

[0173] 또한 USB 장치 펌웨어를 조작하여 우회하기는 쉽지만 지속적으로 등장하는 새로운 USB 장치로 유지 관리하기 어려운 화이트리스트에 의존한다. USBEye에 가장 가까운 GoodUSB는 USB 장치의 인터페이스 목록을 팝업하고 사용자가 화이트리스트를 유지하는 대신 인터페이스를 선택적으로 활성화할 수 있도록 한다. 반대로 USBEye는 이미지 분류를 위해 신경망을 채택하여 USB 장치 연결에 대한 보안 검증 및 의사 결정을 자동화한다. 그렇게 함으로써 빈번한 보안 경고의 부작용을 최소화한다.

[0174] 또는 사회공학적 공격의 위험성을 인지하는 보안 교육, AutoRun 기능의 남용을 방지하기 위한 Windows 보안 설정, 승인되지 않은 소프트웨어 실행을 방지하기 위한 Applocker, 바이러스 스캐너 사회 공학 공격을 완화하기 위해 USB 장치 내부에 숨겨진 맬웨어를 탐지하는 것이 제안된다. 소프트웨어 취약점을 악용한 공격을 저지하기 위해 USB 드라이버 퍼징 및 USB 드라이버 격리 기법을 탐색하였다. 이러한 방어는 USBEye를 보완하여 앞서 언급한 다양한 공격 범주의 여러 공격 기술을 결탁하는 보다 복잡한 USB 공격을 방어한다.

[0175] 도 5는 본 발명의 일 실시예에 따른 모바일 기기의 USB 장치 인증 시스템의 구성을 나타낸 구성 블록도이다. 도 5에 도시된 바와 같이, 모바일 기기의 USB 장치 인증 시스템(10)은 판단부(110), 인터페이스부(120), 제어부(130)를 포함한다.

[0176] 판단부(110)는 인공지능 기반으로 USB 장치의 유형을 판단한다. 이러한 판단부는 USB 장치의 외관을 인공지능 기반으로 판단한다.

[0177] 인터페이스부(120)는 판단부의 결과에 따라 각 USB 장치 유형에 대한 운영체제 커널에 요청된 USB기기의 인터페이스 활성화 여부를 확인한다.

[0178] 인터페이스부(120)는 판단되는 기기의 타입을 바탕으로 허용되는 USB장치의 오퍼레이션을 제어한다. 예를 들면, USB 저장소의 경우 키보드 입출력 작업을 수행할 수 없도록 강제한다.

[0179] 이러한 인터페이스부는 USB 장치 유형별로 요구되는 보안수준에 따른 인공지능 기반 또는 유저기반 정책 설정 방식을 선택하도록 하는 선택모듈(121)을 더 포함한다.

[0180] 제어부(130)는 인터페이스부(120)의 인터페이스 활성화 여부에 따라 현재 연결의 안전성을 결정하기 위한 구성이다.

[0181] 본 실시예에 따른 제어부(130)는 장치가 악성이 아닌 것으로 판단되면 모든 인터페이스에 해당하는 드라이버를 로드한다. 또한 제어부는 USB 기기가 호스트에 연결될 때마다 모든 인터페이스를 팝업하는 엄격 모드를 실행시킨다. 이때, 엄격 모드에서 사용자가 인터페이스의 일부를 선택적으로 활성화할 수 있도록 인터페이스 목록이 표시되도록 한다.

[0182] 본 발명에 따르면 사용자를 속여 USB 장치의 외관과는 다른 작업을 수행하는 공격인 USB 위장 공격을 방어할 수 있다. 알려진 USB 공격 툴인 Rubber Ducky와 같이 USB 저장소가 키입력을 수행하는 공격을 방어할 수 있다.

[0183] 본 실시예에 따르면 인공지능 기반으로 USB기기 타입(종류) 판단 결과를 바탕으로 운영체제 커널에 요청된 USB 기기의 인터페이스 활성화 여부를 제어한다. 또한 요구되는 보안 수준에 따른 인공지능 기반 또는 유저기반 정책 설정 방식 선택기능을 더 포함할 수 있다.

[0184] 도 6은 본 발명의 일 실시예에 따른 모바일 기기의 USB 장치 인증 방법을 나타낸 개략적인 흐름도이다. 도 6에 도시된 바와 같이, 모바일 기기의 USB 장치 인증 시스템을 이용한 방법에 있어서, 모바일 기기의 USB 장치 인증 시스템은 인공지능 기반으로 USB 장치의 유형을 판단한다(S2).

[0185] 다음으로 모바일 기기의 USB 장치 인증 시스템은 (S2)의 판단결과에 따라 각 USB 장치 유형에 대한 운영체제 커널에 요청된 USB기기의 인터페이스 활성화 여부를 확인한다(S4).

[0186] 이러한 S4 단계에서, 모바일 기기의 USB 장치 인증 시스템은 인터페이스부를 통해 USB 장치 유형별로 요구되는 보안수준에 따른 인공지능 기반 또는 유저기반 정책 설정 방식을 선택하도록 하는 단계를 더 포함할 수 있다.

[0187] 다음으로 모바일 기기의 USB 장치 인증 시스템은 (S4)의 인터페이스 활성화 여부에 따라 현재 연결의 안전성을 결정한다(S6).

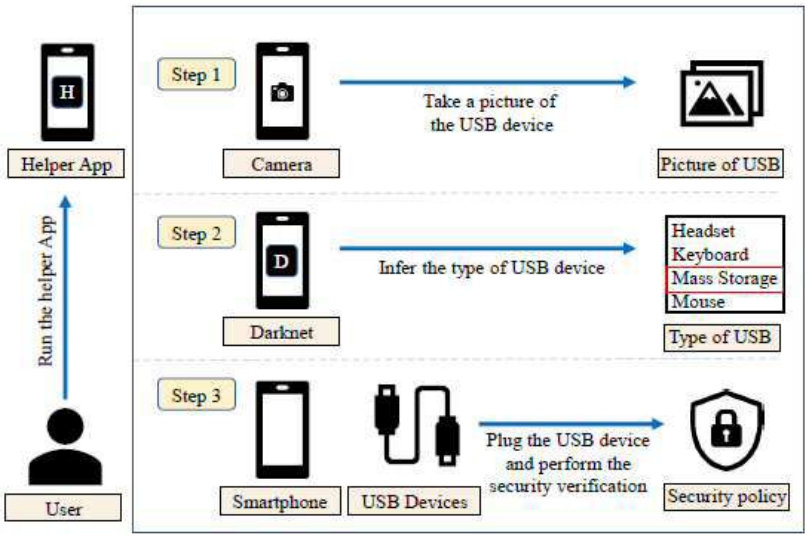
[0188] 이러한 S6 단계에서, 모바일 기기의 USB 장치 인증 시스템의 제어부는 장치가 악성이 아닌 것으로 판단되면 모든 인터페이스에 해당하는 드라이버를 로드하도록 한다. 또한 USB 기기가 호스트에 연결될 때마다 모든 인터페이스를 팝업하는 엄격 모드를 실행시키고, 엄격 모드에서 사용자가 인터페이스의 일부를 선택적으로 활성화할 수 있도록 인터페이스 목록이 표시되도록 한다.

[0189] 본 실시예에 따른 모바일 기기의 USB 장치 인증 시스템은 연결된 USB 장치가 예기치 않은 인터페이스를 정의하는지 여부를 확인하여 가장 공격을 감지하는 USBEye를 설계했다. 특히 USB 장치에 정의된 인터페이스는 보안 정책에 정의된 현재 연결된 장치에 대해 허용된 인터페이스 목록과 비교하여 확인된다. 정책에서 목록을 가져오는 키가 되는 USB 장치의 유형은 CNN에 의해 유추되며, 이는 Darknet을 사용하여 구현되고 Android NDK 서비스로 실행된다.

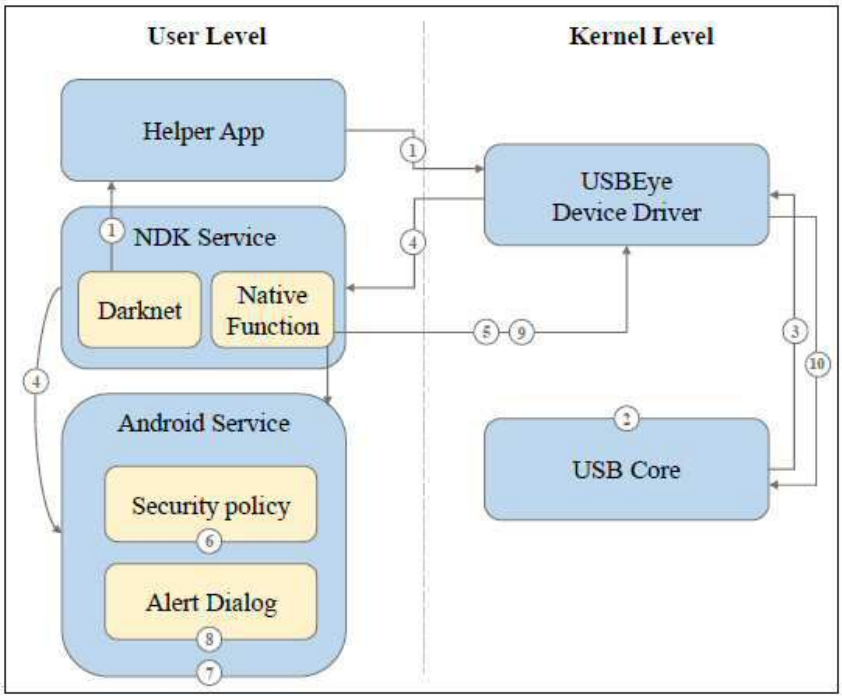
[0190] 보안성 평가에서 USBEye는 악성 키보드 이벤트를 발생시켜 가장 공격을 수행하는 악성 USB 장치(예: Rubber Ducky)를 탐지하고 차단할 수 있음을 보여주었다. 성능 평가에서는 USB 장치 연결 시 101%, HID 장치 작동 시 2%의 오버헤드를 관찰했다. 향후 개인정보 보호 USB 유형 분류를 위해 연합 학습을 도입할 수 있다. 또한 장치 소유자를 다른 소유자와 구별하는 AI 지원 인증 방식을 탐색한다. 따라서 악의적인 USB 장치가 피해자의 장치에 단기적으로 물리적으로 액세스할 수 있는 잠재적 공격자가 USBEye를 우회하는 것을 방지할 수 있다.

도면

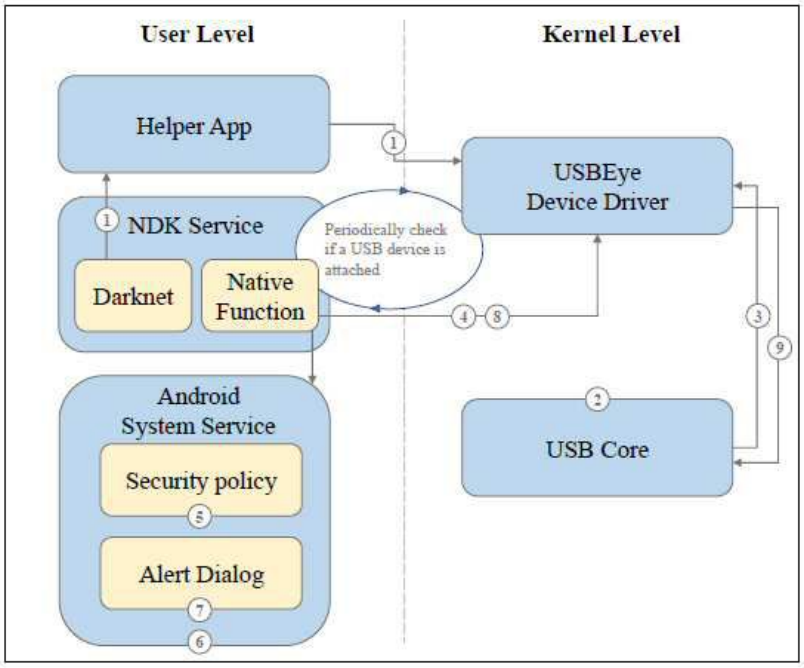
도면1


Helper App
Run the helper App
User
Step 1
Camera
Take a picture of the USB device
Picture of USB
Step 2
Darknet
Infer the type of USB device
Headset
Keyboard
Mass Storage
Mouse
Type of USB
Step 3
Smartphone
USB Devices
Plug the USB device and perform the security verification
Security policy

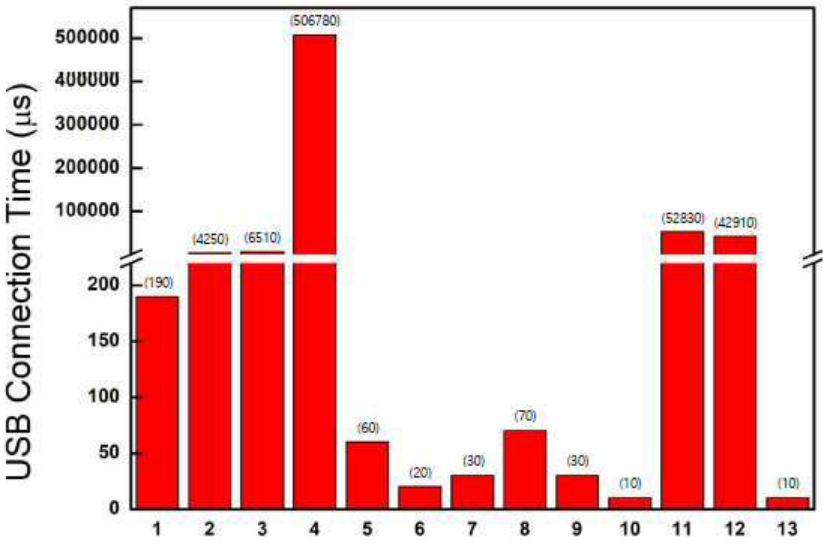
도면2


User Level
Kernel Level
Helper App
NDK Service
Darknet
Native Function
Android Service
Security policy
Alert Dialog
USBEye Device Driver
USB Core

도면3


User Level
Kernel Level
Helper App
USBEye
Device Driver
NDK Service
Periodically check if a USB device is attached
Darknet
Native Function
Android System Service
Security policy
Alert Dialog
USB Core

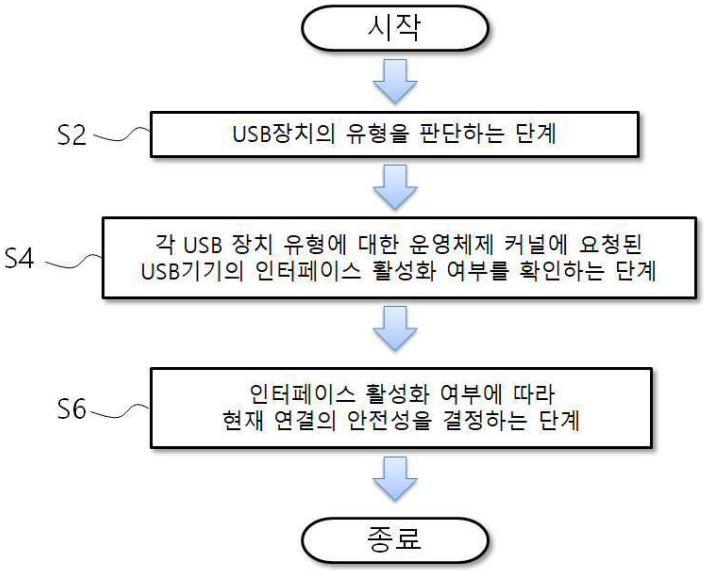
도면4


USB Connection Time (μs)
500000
400000
300000
200000
100000
200
150
100
50
0
(190)
(4250)
(6510)
(506780)
(60)
(20)
(30)
(70)
(30)
(10)
(52830)
(42910)
(10)
1
2
3
4
5
6
7
8
9
10
11
12
13

도면5


판단부
110
선택모듈
121
인터페이스부
120
제어부
130
10

도면6


시작
S2
USB장치의 유형을 판단하는 단계
S4
각 USB 장치 유형에 대한 운영체제 커널에 요청된
USB기기의 인터페이스 활성화 여부를 확인하는 단계
S6
인터페이스 활성화 여부에 따라
현재 연결의 안전성을 결정하는 단계
종료