Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.36, NO.4, Aug. 2026 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2026.36.4.1227

eSPADE-XR: XR 아키텍처 상 런타임 실행 의도 추적 및 권한 정합성 검증 프레임워크*

주 효 중,[1†] 이 승 수[2‡]
[1,2]인천대학교(학생, 교수)

eSPADE-XR: A Framework for Tracing Intent-to-Use and Verifying Permission Consistency on XR Architectures*

Hyojoong Ju,[1†] Seungsoo Lee[2‡]
[1,2]Incheon National University (Undergraduate student, Professor)

요 약

안드로이드 기반 XR(eXtended Reality) 앱은 민감한 데이터를 처리하나, Binder IPC(Inter-Process Communication) 간접 통신 구조와 Unity/IL2CPP(Intermediate Language To C++), SDK(Software Development Kit) 보일러플레이트로 실제 실행 여부와 제어 의미 판단이 어렵다. 본 논문은 이를 완화하기 위해 eSPADE-XR을 제안한다. eSPADE-XR은 정적 결과를 후보화하고, eBPF(extended Berkeley Packet Filter)로 Target App - XR Broker 간 metadata를 수집하며 AIDL(Android Interface Definition Language) mapping으로 transaction code를 control method로 복원한다. Android XR 앱 31종에서 204개 중 51개를 선별해 33개 Strong, 16개 Weak, 2개 Invalid로 분류해 정적 후보와 Broker-level semantic evidence의 정합성을 보였다.

ABSTRACT

Android XR(eXtended Reality) devices process privacy-sensitive continuous sensing data, but runtime verification is difficult because XR apps communicate with XR Brokers through Binder IPC(Inter-Process Communication), while Unity/IL2CPP(Intermediate Language To C++) and SDK(Software Development Kit) boilerplate limit static analysis. We propose eSPADE-XR, which treats static findings as runtime candidates, collects Target App - XR Broker Binder IPC metadata using eBPF(extended Berkeley Packet Filter), and decodes transaction codes into Broker-level control methods via AIDL(Android Interface Definition Language) mapping. In 31 Android XR apps, eSPADE-XR selected 51 runtime candidates from 204 static candidates and classified them as 33 Strong, 16 Weak, and 2 Invalid. The results show consistency verification between static XR candidates and Broker-level semantic evidence without confirming sensor payload access or malicious behavior.

Keywords: Extended Reality, Permission, Binder IPC, eBPF, Security and Privacy

Received(03. 04. 2026), Modified(06. 08. 2026), Accepted(06. 29. 2026)
* 본 논문은 2026년도 한국정보보호학회 동계학술대회에 발표한 우수논문을 개선 및 확장한 것임.

* 이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No.RS-2025-16069415).
† 주저자, alex0888@inu.ac.kr
‡ 교신저자, seungsoo@inu.ac.kr(Corresponding author)

I. 서 론

최근 고성능 XR(eXtended Reality) 기술이 산업 전반으로 확산되고 있다. 스마트폰과 달리 XR 기기는 사용자 경험을 극대화하기 위해 시선, 손동작, 공간 맵핑 정보 등을 실시간으로 수집하는 연속 센싱(Continuous Sensing) 기능이 요구된다.[1, 2] 하지만 이러한 고해상도 민감 데이터 상시 수집은 개인정보 침해 위협[3-5]을 동반하여, 애플리케이션과 시스템 서비스 전반의 공격 표면을 확대시킬 수 있다.[6-8] 특히 안드로이드 기반 XR 플랫폼은 센서 데이터 수집 및 처리 경로가 스마트폰에 비해 복잡하다. 앱이 센서 API를 직접 호출하는 대신 추상화된 런타임과 시스템 서비스를 경유하는 간접 접근 방식을 취하기 때문에, Manifest에 선언된 권한과 실제 코드상의 동작 사이의 정합성을 확인하기 어렵다[9,10,11].

이러한 한계를 고려하여, 본 연구는 권한 선언과 런타임 사이의 간극을 커널 수준에서 실증하는 다중 계층 증거 기반 검증 프레임워크를 제안한다. 본 프레임워크는 정적 분석을 동적 관측을 위한 후보 생성 도구로 재정의하여 기존 정적 분석의 모호성을 동적 관측을 통해 보완하며, 앱 계층의 기능 요청이 XR Broker 경유 제어 경로로 이어지는지를 추적한다. 최종적으로, 수집된 시스템 신호의 결정성에 기반한 객관적인 평가 지표를 도입하여, XR 서비스 경유 경로의 정합성을 명확한 데이터 기반으로 제시한다.

본 논문이 기여하는 바는 다음과 같다.

- XR 앱의 런타임 구조를 반영하여, 정적 분석을 잠재적 위협의 후보 생성기로 활용하고, 이에 eBPF 기반 동적 관측의 엄밀성을 결합한 다단계 검증 아키텍처를 정립하였다.
- 애플리케이션과 XR Runtime을 수정하지 않고 eBPF 기반 Binder IPC[12] 관측을 통해 대상 앱과 XR Broker 사이의 런타임 제어 경로를 수집한다.
- AIDL(Android Interface Definition Language)[13] Transaction mapping을 통해 Binder transaction code를 XR Broker 제어 메서드 수준으로 복원한다.
- 실제 안드로이드 앱 대상 정적 분석으로 도출된 의심 코드가 런타임에서 Broker를 통해 통신하여, XR 관련 제어 메서드를 호출하려 하는지를 분석하고, 그 결과를 정량화한다.

II. 배 경

2.1 XR 아키텍처 및 Binder IPC 통신 구조

전통적인 Android 환경[14]에서 애플리케이션은 프레임워크가 제공하는 SensorManager 등의 표준 API를 통해 센서 이벤트를 요청, 실제 하드웨어 접근은 안드로이드 프레임워크와 시스템 서비스 계층을 거쳐 처리된다. 그러나 Android 기반 XR 플랫폼은 시선, 안면, 손 동작 및 주변 공간 등 고도화된 연속 센싱 데이터를 실시간으로 처리하기 위해 일반 모바일 기기보다 복잡한 런타임 구조를 가진다. 이러한 보안 설계의 일환으로, 개별 앱이 커널 공간의 하드웨어 디바이스 노드에 직접 접근하는 것은 차단된다. 대신 앱은 샌드박스 내에 격리되며, XR 관련 기능 요청은 런타임, XR Broker, 시스템 서비스와 같은 중간 계층을 통해 처리된다[15].

해당 구조에서 XR 런타임 계층은 앱과 하위 시스템 서비스 간 제어 요청을 중계하는 핵심 지점으로 동작한다. Fig. 1과 같이, 앱이 사용자 상호작용을 위해 연속적인 센싱 기능을 초기화하거나 가상 공간의 노드를 생성하려 할 때, 앱과 XR Runtime 간에는 일반적으로 Binder 기반의 프로세스 간 통신(IPC, Inter-Process Communication)이 발생한다(Fig. 1-①,②). 이 때 Binder IPC는 센서 데이터 자체가 아니라, XR 기능을 초기화하거나 관련 자원을 요청하기 위한 제어 경로로 해석할 수 있다.

이후 XR Runtime은 요청의 성격에 따라 하위 계층으로의 접근을 두 가지 경로로 분리하여 처리한다. 노드 생성이나 센서 초기화 등의 일반적인 제어 명령은 IPC(AIDL/HwBinder)를 통해 하드웨어 추상화 계층(HAL, Hardware Abstraction Layer)[16] 또는 관련 시스템 서비스로 전달되며(Fig. 1-③,④), 커널 드라이버와 상호작용한다(Fig. 1-

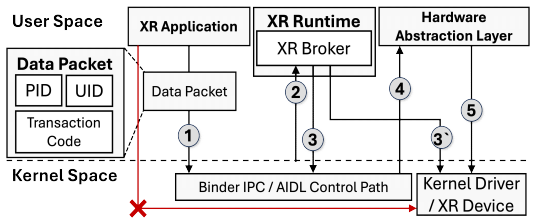

Fig. 1. Android XR runtime architecture and control/data path separation

⑤). 반면, 멀미 방지를 위해 초저지연 처리가 필수적인 연속 센싱 데이터 수신이나 하드웨어 가속 연산의 경우, 매번 Binder IPC를 통해 데이터를 전달하기보다 공유 메모리[17]나 고속 메시지 큐(FMQ, Fast Message Queue)[18]와 같은 별도 데이터 경로가 활용될 수 있다. 이 경우 Binder는 데이터 전송 자체보다 데이터 경로를 설정하거나 핸들을 전달하는 초기 제어 경로로 사용될 수 있으며, 이후 런타임은 메모리 매핑된 영역을 통해 센싱 데이터를 처리[17]한다. (Fig. 1-③).

즉, 런타임 상에서 센싱 데이터 자체는 성능 최적화를 위해 Binder 외의 경로를 통해 흐를 수 있지만, 시스템 아키텍처상 이를 개시하고 하드웨어 자원을 할당받기 위한 최초의 관문 요청은 XR Broker 또는 런타임 계층을 경유하는 Binder IPC로 관측될 수 있다. 이러한 구조적 특징으로 인해, Binder IPC는 앱이 XR 제어 경로에 진입했는지를 확인하기 위한 런타임 증거로 활용될 수 있다.

2.2 Android Interface Definition Language

Android 시스템 내에서 서로 다른 프로세스가 데이터를 주고받을 때에는 주로 Binder IPC 메커니즘이 사용되며, 이때 프로세스 간 호출 인터페이스를 정의하기 위해 AIDL(Android Interface Definition Language)[13]이 활용된다. AIDL은 클라이언트와 서비스가 공유하는 인터페이스를 정의하고, 이를 기반으로 프로세스 간 메서드 호출을 Binder transaction으로 변환한다.

일반적인 Android 프레임워크의 AIDL 인터페이스는 AOSP를 통해 공개되어 있어 인터페이스 정의와 transaction code의 대응관계를 비교적 명확하게 확인할 수 있으나, XR 런타임 서비스의 경우 내부 제어 메서드가 캡슐화되어 있다. 커널 수준에서 Binder IPC를 관측할 경우, 메서드명 대신 sender PID/TGID, receiver PID, transaction code와 같은 메타데이터만 확인할 수 있다. 이러한 현상은 추상화된 통신계층이 만들어내는 의미론적 단절이며, XR 권한 후보를 런타임에서 동적으로 검증하기 위해 해소해야 할 문제로 작용한다.

2.3 extended Berkeley Packet Filter

eBPF(extended Berkeley Packet Filter)[19]는 운영체제 커널의 소스 코드를 수정하거나 커널 모듈을 추가하지 않고도, 커널 공간에서 안전하게 사용자 정의 관측 프로그램을 실행할 수 있도록 지원하는 메커니즘이다.[20] 안드로이드 시스템에서는 이를 통해 커널 및 사용자 공간에서 발생하는 다양한 이벤트를 낮은 시스템 오버헤드로 추적할 수 있다.
최신 Android 환경은 기본적으로 CONFIG_BPF 및 관련 옵션들이 활성화되어 있어, 커널 내 사전 정의된 정적 탐지점인 Tracepoint 등을 활용한 이벤트 캡처에 매우 적합하다. 이를 통해 사용자 공간의 XR 앱 실행 흐름을 간섭하지 않으면서도, 앞서 언급한 Binder 통신 내역이나 시스템 콜 등의 저수준 커널 이벤트를 실시간으로 수집할 수 있는 안정적인 런타임 관측 환경 구성이 가능하다.

III. 문제 상황

확장 현실 환경의 연속 센싱 데이터[21]는 한 번 유출되면 변경이 불가능한 고위험[3] 특성을 띠므로, 엄격한 권한 통제가 필수적이다. 그러나 현재의 Android XR 플랫폼 생태계는 이러한 치명적인 데이터를 다루고 있음에도 불구하고, 권한 검증에 있어 다음과 같은 세 가지 구조적 한계를 지닌다.

3.1 P1: 정적 분석 기반 XR 권한 검증의 한계

기존의 정적 분석 방식[22]은 주로 앱이 선언한 권한(Manifest)과 코드 내부의 키워드를 단순 비교하는 수준에 머물렀다. 하지만 XR 앱은 3D 환경 구현을 위해 Unity/IL2CPP, 여러 SDK, 런타임 라이브러리 등 복잡한 구성 요소를 포함하는 경우가 많아 정적으로 발견된 XR 관련 흔적만으로는 실제 런타임 실행 여부를 판단하기 어렵다[23,24]. 또한 Manifest에 민감 XR 권한이 선언되어 있더라도 해

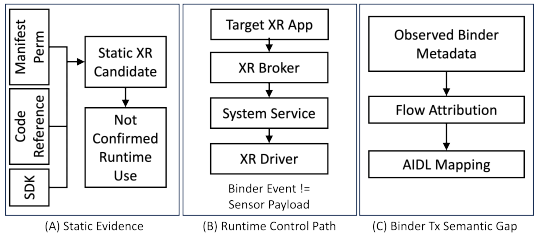

Fig. 2. Structural challenges in Android XR permission verification

당 권한이 실제 실행 중 XR Broker 경유 제어 경로로 이어졌는지는 확인할 수 없다. 따라서 Fig. 2와 같이 정적 분석 결과는 실제 XR 자원 사용이나 위협의 확정 증거가 아니라, 후속 런타임 검증을 위한 후보로 해석되어야 한다.

3.2 P2: XR 아키텍처의 기술적 복합성과 런타임 제어 경로 관측 문제

Android XR 환경에서는 앱이 하드웨어 센서나 XR 자원에 직접 접근하지 않고, XR Broker와 같은 중간 계층을 경유하는 복잡한 아키텍처를 지닌다. 이러한 간접 접근 구조는 정적 코드 흔적과 실제 런타임 제어 경로를 관측하기 어렵게 만든다. 또한 실제 센서 데이터는 공유 메모리, FMQ, HAL 경로 등을 통해 전달될 수 있으므로, Binder IPC는 센서 데이터 자체가 아닌, XR 자원 사용을 초기화하거나 제어 경로에 진입했음을 보이는 런타임 증거로 해석되어야 한다[16-18]. 따라서 Fig. 2와 같이 앱의 XR 관련 요청이 XR Broker를 경유하는지 확인하기 위해서는, 앱이나 XR 런타임을 직접 수정하지 않고 커널 수준에서 Target App과 XR Broker 사이의 Binder IPC를 관측할 수 있는 방식이 필요하다.

3.3 P3: Binder IPC Transaction의 XR 의미 해석 한계

커널 수준에서 Binder IPC를 관측하더라도, 수집되는 정보는 sender PID/TGID, receiver PID, transaction code와 같은 저수준 메타데이터에 제한된다[25,26]. 특히 XR Broker는 여러 앱과 시스템 구성요소의 요청을 처리할 수 있으므로, 단순히 Broker 관련 Binder event가 관측되었다는 사실만으로 특정 앱의 XR 요청이라고 판단하기는 어렵다. 또한 raw transaction code만으로는 해당 호출이 어떤 XR 제어 메서드에 대응하는지 알 수 없으므로, Fig. 2와 같이 Target App에서 XR Broker로 향하는 흐름을 PID/TGID와 Broker PID를 기준으로 구분하고, transaction code를 AIDL mapping table과 대조하여 XR Broker 제어 메서드 수준으로 복원하는 과정이 필요하다.

IV. 관련 연구

Android 앱 보안 연구에서는 Manifest 권한과 코드 내부 API 호출을 분석하여 민감 자원 접근 가능성을 추정하는 정적 분석 기법이 널리 활용되어 왔다[9-11]. 이러한 방식은 대규모 앱 분석에 적합하지만, SDK, 난독화 코드, 네이티브 라이브러리, 런타임 프레임워크가 포함된 앱에서는 정적으로 발견된 흔적이 실제 런타임 실행으로 이어지는지 판단하기 어렵다[22-24]. 특히 Unity/IL2CPP 기반 XR 앱은 Java/Kotlin 계층뿐 아니라 native binary와 metadata에 XR 관련 참조가 분산될 수 있어 단순 문자열 또는 API 참조만으로 권한 사용을 확정하기 어렵다.

XR 프라이버시 연구는 시선, 손동작, 얼굴, 공간 맵핑 등 연속 센싱 데이터의 민감성과 권한 통제 필요성을 지적해왔다[3-5]. 그러나 기존 연구는 주로 데이터 민감도, 사용자 인지, 권한 모델에 초점을 두었으며, Android XR 앱의 정적 권한 후보가 실제 런타임 제어 경로로 연결되는지를 시스템 수준에서 검증하는 문제는 충분히 다루지 않았다[6-8,21,27].

동적 분석과 자동화 테스트는 정적 분석의 한계를 보완하지만, XR 앱의 세션 초기화, 공간 진입, 제스처 기반 상호작용 조건을 안정적으로 재현하기 어렵다[28,29]. 또한 Binder IPC에서 관측되는 정보는 PID/TGID와 transaction code 등 저수준 메타데이터에 제한되므로[12,25], AIDL mapping 없이는 XR Broker 제어 메서드의 의미를 해석하기 어렵다[13,26]. 본 연구는 정적 분석을 후보 생성으로 재정의하고, 표적 실행, eBPF 기반 Binder 관측, AIDL 해독을 결합하여 Broker-level semantic evidence를 검증한다.

V. 시스템 디자인

5.1 디자인 고려사항

본 연구는 앞서 정의한 Android XR 플랫폼의 권한 검증 한계에 대응하기 위해, 새로운 하이브리드 분석 프레임워크가 충족해야 할 세 가지 핵심 설계 요구사항을 다음과 같이 도출한다.

5.1.1 후보 생성으로의 정적 분석 재정의

기존의 안드로이드 권한 분석 기법은 주로 정적 분석의 결과물을 최종적인 위협 탐지 지표로 활용해 왔다. 그러나 XR 환경에서는 Unity와 같은 3D 런타임 엔진, 제조사 제공 SDK, 런타임 라이브러리, 보일러플레이트 코드가 앱 패키지에 함께 포함될 수 있어 정적 분석만으로는 실제 실행 여부를 판단하기 어렵다. 따라서 본 프레임워크는 정적 분석의 역할을 "최종 탐지"가 아닌, 동적 분석을 위한 "표적 추출" 및 "후보 생성" 단계로 역할을 제한하여 재정의한다. 앱의 Manifest에 선언된 권한과 디컴파일된 코드에서 발견된 XR 관련 API, 참조, 메서드를 비교하고, 이를 앱-권한-컴포넌트 단위의 런타임 검증 후보로 구조화함으로써, 이어지는 동적 분석에서 관측할 대상을 결정하도록 설계하였다.

5.1.2 커널 레벨 IPC 관측을 통한 제어 경로 확인

XR 앱은 하드웨어 센서 자원에 직접 접근하기보다 XR 런타임, XR Broker, System Service와 같은 중간 계층을 경유하여 기능 요청을 수행한다. 이러한 간접 접근 구조에서는 앱 내부 코드만으로 실제 런타임 제어 경로를 확인하기 어렵고, 전통적인 사용자 공간 기반 후킹 도구는 이러한 폐쇄적인 통신 구조에서 실행 흐름에 영향을 줄 수 있다. 또한, Binder IPC를 관측하더라도 수집된 데이터는 저수준 메타데이터에 제한되므로, 해당 호출의 실제 목적을 파악할 수 없는 의미론적 단절이 발생한다. 이를 극복하기 위해 본 시스템은 eBPF를 이용해 리눅스 커널의 tracepoint에서 Binder transaction metadata를 수집하는 구조를 채택하였다. 이를 통해 앱과 XR 런타임을 직접 수정하지 않고, Target App에서 XR Broker로 향하는 IPC 요청을 관측한다.

5.1.3 IPC 흐름 식별 및 AIDL 기반 Transaction 의미 복원

XR Broker는 여러 앱과 시스템 구성요소의 요청을 동시에 처리할 수 있으므로, 단순히 Broker 관련 Binder 이벤트가 관측되었다는 사실만으로는 특정 앱의 XR 제어 요청이라고 판단하기 어렵다. 따라서 Target App의 PID/TGID와 XR Broker PID를 기준으로 앱에서 Broker 방향의 Binder transaction을 식별하고, Broker 이후의 하위 서비스 호출 등의 접근은 보조적인 흐름 증거로 분리한다. 또한 커널 수준에서 수집되는 Binder transaction은 메서드명이 아닌 정수형 transaction code로 표현되므로, 이를 XR AIDL 매핑 테이블과 대조하여 XR Broker 제어 메서드 수준으로 복원한다. 이 과정을 통해 raw IPC metadata를 사람이 해석 가능한 의미론적 증거로 변환하고, 정적 후보가 런타임에서 XR Broker 제어 경로와 연결되는지 평가한다.

5.2 전체 아키텍처

본 연구에서 제안하는 하이브리드 프레임워크인 eSPADE-XR은 Fig. 3와 같이, 디컴파일된 대상 앱들의 최상위 디렉터리 경로(Base Directory Path)를 입력(Input)으로 받아 정적 후보 생성, 런타임 제어 경로 관측, 증거 통합 및 평가로 이어지는 파이프라인을 수행한다. 전체 시스템은 정적 분석을 통해 검증 대상을 구조화하는 후보 생성(Candidate Generation) 단계, 후보 컴포넌트 실행 구간에서 커널 수준 Binder IPC를 관측하는 런타임 관측(Runtime Observation) 단계, 그리고 수집된 다계층 데이터를 종합하여 최종 위험도를 산출하는 교차 검증 및 평가(Cross-Verification & Evaluation) 단계로 구성되며, 각 단계는 세분화된 내부 컴

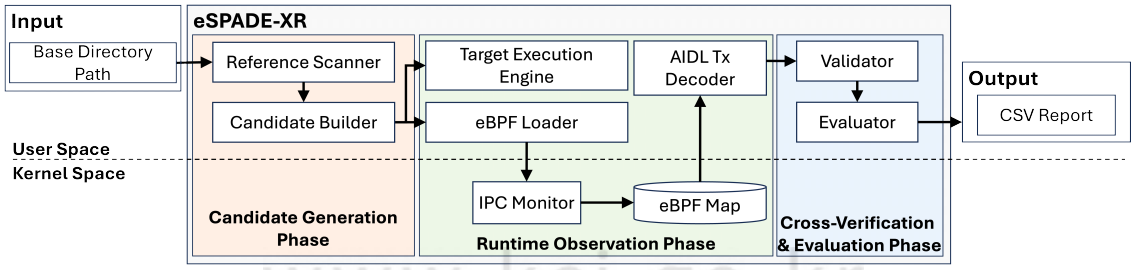

Fig. 3. Overall Architecture of eSPADE-XR

포넌트들을 통해 데이터의 흐름을 제어한다.

첫 번째 후보 생성 단계는 디컴파일러에 의해 해체된 대상 앱들의 최상위 디렉터리 경로를 입력으로 받는다. 먼저 앱의 매니페스트(AndroidManifest.xml)를 파싱하여, 패키지명 기준으로 타겟 디렉터리를 논리적으로 그룹화한다. 이후, Reference Scanner는 그룹화된 디렉터리 내 3D 런타임 엔진의 메타데이터 및 Smali 코드 계층을 스캔하여 XR 특화 권한이 참조되는 구역을 탐색한다. 이때 정적 분석 결과는 실제 권한 사용의 확정 증거가 아니라 런타임 검증 후보로 해석된다. Candidate Builder는 탐색된 정적 evidence를 앱-권한-컴포넌트 단위로 구조화하고, 동적 관측 대상으로 사용할 primary candidate를 선별한다. 이 과정을 거쳐 생성된 정적 후보는 01_static_candidates.jsonl 형태로 저장되어 다음 파이프라인에 전달된다.

두 번째 런타임 관측 단계는 정적 분석에서 도출된 JSONL을 바탕으로 동적 커널 통신을 감시한다. Target Execution Engine은 후보 컴포넌트를 실행하여 관측 가능한 런타임 구간을 만들고, 실제 대상 앱 PID를 확보한 뒤 eBPF Collector를 해당 PID 기준으로 부착한다. 이후 IPC Monitor는 Binder 관련 tracepoint에서 Target App과 XR Broker 사이의 transaction metadata를 필터링, 수집하여 eBPF Map에 실시간으로 기록한다. 수집된 transaction code는 사전 구축된 AIDL mapping table과 대조되어 제어 메서드 수준의 의미 정보로 복원될 수 있는지 확인된다.

마지막 교차 검증 및 평가 단계에서는 앞선 모듈들의 산출물을 통합 분석한다. Evaluator는 정적 분석에서 생성된 candidate와 런타임 관측 결과를 연결하고, 동일 후보에 대해 여러 컴포넌트 실행 attempt가 존재할 경우 가장 강한 관측 증거를 기준으로 candidate-level 결과를 산출한다. 이때 최종 등급은 실제 센서 데이터 접근이나 악성 행위 여부가 아니라, 정적 후보가 런타임에서 XR Broker 제어 경로 및 AIDL 기반 제어 메서드 증거로 연결되는지를 나타내는 evidence level로 정의된다. 최종 결과는 CSV 형태로 산출되며, Evaluator는 이 검증 결과를 바탕으로 각 앱의 실질적인 XR 권한 오용 위험도를 2단계(Strong/Weak)로 정량화하여 구조화된 형태의 평가 리포트를 도출한다.

VI. 시스템 세부사항

6.1 다계층 정적 스캔 및 구조적 노이즈 정제

6.1.1 Manifest 파싱 및 코드 레벨 XR 증거 추출

앞서 제기된 정적 분석의 한계(P1)에 대응하고 런타임 관측 대상을 구조화하기 위해, 본 프레임워크에서는 먼저 입력된 디렉터리들을 필터링하는 전처리 과정을 거친다. 다수의 디컴파일된 파일 경로에서 AndroidManifest.xml을 파싱하여 패키지(Package)명을 식별하고, 이를 통해 Split APK 등으로 분산된 디렉터리들을 단일 앱 기준으로 그룹화한다. 동시에 앱이 명시적으로 요구한 XR 특화 권한의 기준선을 확립한다. 이후 Unity/IL2CPP 기반 XR 앱의 구조적 불투명성을 완화하기 위해 다계층 스캔을 수행한다. 구체적으로는 libil2cpp.so 바이너리와 global-metadata.dat의 존재 여부를 확인하고, IL2CPPDumper를 통해 dump.cs 생성을 시도한 뒤, 실패 시 문자열 기반 탐색으로 폴백한다. 이렇게 확보된 심볼과 Java/Kotlin의 Smali 코드 계층을 대상으로 공식 문서 기반 XR 키워드 세트를 이용한 문자열 및 정규식 기반 매칭을 수행하여 하드웨어 센서나 공간 정보 접근을 시사하는 1차적인 코드 레벨 단서를 추출한다.

6.1.2 증거 모델링을 통한 정적 후보 구조화

추출된 1차 단서들은 서드파티 라이브러리나 3D 엔진(예: Unity XR Interaction Toolkit) 등이 포함되면서 발생하는 정적 코드 흔적을 포함할 수 있다. 이러한 구조적 모호성(P1)을 고려하여, 수집된 단서들에 증거 모델링 기법을 적용한다. 단순히 권한 참조의 존재 여부만을 판별하는 것이 아니라, Manifest에 선언된 XR 권한과 코드 레벨 evidence의 존재 여부를 비교하여 MATCH, OVER, NONE으로 분류하고, 이 중 Manifest에 민감 XR 권한이 선언된 MATCH 및 OVER 후보만을 후속 런타임 관측 대상으로 구조화한다.

6.1.3 표적 실행을 위한 타겟 컴포넌트 구조화

최종적으로, 고가중치 정적 증거로 선별된 정적 후보가 앱 내의 어떤 구체적인 컴포넌트(Activity

또는 Service)와 연결되는지 구조화한다. 이때 코드 증거에서 추정된 class 정보와 Manifest component 정보를 대조하여 후속 실행 대상 컴포넌트를 선정한다. 산출된 후보는 컴포넌트명, 대상 권한, 정적 label 등을 포함한 JSONL(JSON Lines) 포맷으로 저장되며, 이는 후속 파이프라인에서 무작위 탐색이 아닌 표적 실행(Targeted Execution)의 핵심 입력 데이터로 활용된다.

6.2 표적 실행 및 커널 통신 모니터링

6.2.1 표적 실행 기반 런타임 관측 구간 생성

정적 분석에서 도출된 후보 컴포넌트가 런타임에서 XR Broker 경유 제어 경로와 연결되는지 관측하기 위해, 본 프레임워크에서는 표적 실행을 통해 동적 관측 구간을 생성한다. 일반적인 XR 앱은 특정한 가상 공간 진입이나 사용자의 제스처가 발생해야만 관련 코드가 활성화되는 경우가 많아, 기존의 무작위 UI 탐색 도구(예: Android Monkey)로는 XR 실행 경로에 도달하기 어렵다. 이러한 한계를 완화하고자 본 프레임워크는 Fig. 4의 Targeted Execution 단계와 같이 앞선 모듈에서 전달받은 JSONL 형식의 후보 리포트를 파싱하여 권한별 Activity 및 Service 컴포넌트를 추출한다. 이후 am start -n 명령어를 통해 해당 컴포넌트를 직접 실행하여 관측 가능한 런타임 구간을 구성한다. 이 과정은 정상 사용자 상호작용 전체를 재현하거나 XR 세션 초기화를 보장하기 위한 것이 아니라, 후보 컴포넌트 실행 중 Target App과 XR Broker 사이의 Binder IPC evidence가 관측되는지 확인하기 위한 절차이다.

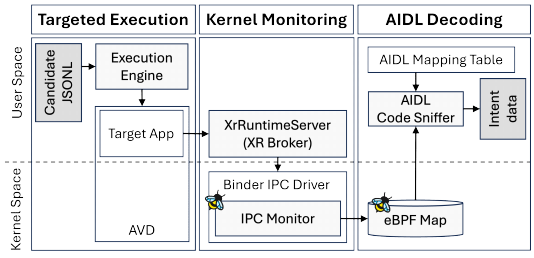

Fig. 4. Target execution and kernel communication monitoring

6.2.2 eBPF 기반 Binder IPC 메타데이터 수집

표적 실행된 앱이 XR Broker 경유 제어 경로에 진입하는지 관측하기 위해, 커널 영역에서 앱과 시스템 브로커 간의 IPC 메타데이터를 수집한다. C언어로 작성된 eBPF 프로그램을 리눅스 커널의 binder_transaction 트레이스포인트에 부착하여, 안드로이드 시스템에서 발생하는 Binder 메시지 중 사전 식별된 XR Broker PID에 연관된 트랜잭션을 필터링한다. 구체적인 필터링 로직은 Fig. 5의 의사코드(Pseudocode)와 같이 동작한다. eBPF 훅(hook)은 매 트랜잭션 이벤트마다 송신자 PID/TGID(s), 수신자 PID(r), 그리고 트랜잭션 코드(c)를 추출한다(Line 2-4). 이후 송신자가 타겟 앱(target_pid)이고 수신자가 XR 브로커(broker_pid)인 경우, 이를 Target App에서 XR Broker로 향하는 1st-hop 제어 경로 evidence로 기록하고, 해당 transaction code를 eBPF map에 저장한다(Line 6-12). 또한 XR 브로커가 송신자인 상황에서 하위 시스템 서비스로 향하는 트랜잭션이 관측될 경우, 이를

```
Input   : binder_transaction event t
Output  : eBPF Map Components
1.  hook binder_transaction(t)
2.    s ⟨- t.sender_tgid
3.    r ⟨- t.recv_pid
4.    c ⟨- t.code
5.
6.    if s = target_pid then
7.      update Map_TargetToRecv ⟨- r
8.      if r = broker_pid then
9.        update Map_TargetToBrokerCode ⟨- c
10.       update Map_PrimaryEvidence ⟨- (s, r, c)
11.     end if
12.   end if
13.
14.   if s = broker_pid then
15.     update Map_BrokerToRecv ⟨- r
16.     update Map_BrokerToCode ⟨- c
17.     if r != target_pid then
18.       update Map_BrokerCallback ⟨- c
19.     else
20.       update Map_BrokerToService ⟨- (r, c)
21.     end if
22.   end if
23.
24.   return
```

Fig. 5. Pseudocode for Binder IPC Metadata Collection

Broker 이후의 보조 흐름 evidence로 별도 기록한다(Line 14-22). 이렇게 수집된 트랜잭션 데이터는 Fig. 4의 Kernel Monitoring 흐름과 같이 eBPF Map에 기록되며, 후속 단계에서 Target App과 XR Broker 사이의 제어 경로 관측 여부 및 AIDL 기반 transaction code 해석에 활용된다. 이러한 커널 레벨 관측 방식은 앱과 XR Runtime을 직접 수정하지 않고 런타임 통신 흐름을 수집할 수 있다는 점에서, 사용자 공간 후킹 방식보다 실행 흐름에 대한 간섭을 줄일 수 있다.

Binder transaction 외에도, 본 프레임워크는 동일한 관측 구간에서 XR Broker가 Target App으로 수행하는 callback 여부와 특정 XR 관련 서비스로 향하는 접근 여부를 보조 런타임 증거로 함께 기록한다. 이러한 정보는 Target App → XR Broker 방향의 primary evidence를 대체하지 않으며, 런타임 제어 경로 주변에서 발생한 부가적인 흐름을 해석하기 위한 supporting evidence로 사용된다.

6.3 Binder IPC 의미 해석 및 방향성 흐름 분석

6.3.1 대상 앱, XR Broker, 하위 서비스 간 흐름 식별

정적 분석(6.1)과 런타임 관측(6.2)에서 도출된 다계층 데이터를 통합하기 위해, 본 프레임워크는 먼저 Binder IPC의 요청 주체와 방향성을 식별한다. XR Broker는 여러 앱과 시스템 구성요소의 요청을 동시에 처리할 수 있으므로, 단순히 Broker 관련 트랜잭션이 관측되었다는 사실만으로 이를 대상 앱의 XR 제어 요청으로 해석할 수 없다. 이에 따라 eBPF 관측 단계에서 수집된 sender PID/TGID, receiver PID, Broker PID, transaction code를 기준으로, Target App에서 XR Broker 방향으로 발생한 Binder transaction을 우선적으로 식별한다. 이 흐름은 정적 후보가 런타임에서 XR Broker 경유 제어 경로로 연결되었는지를 판단하기 위한 primary evidence로 사용된다. 반면 XR Broker가 하위 시스템 서비스로 전달한 요청이나 특정 XR 관련 service로 향하는 호출은 동일 실행 관측 구간 내에서 수집된 supporting evidence로 분리하여 기록한다. 이러한 구분을 통해 백그라운드 시스템 이벤트나 다른 프로세스의 통신이 대상 앱의 실행 결과로 잘못 귀속되는 것을 줄이고, 후속 AIDL decoding 단계에서 app-to-broker transaction code만을 의미 해석 대상으로 사용할 수 있도록 한다. 단, 이 과정은 개별 Binder transaction 간의 인과적 연쇄를 복원하는 것이 아니라, 동일 관측 구간 내에서 Target App, XR Broker, 하위 서비스 사이의 방향성 흐름을 구분하는 절차로 한정된다.

6.3.2 AIDL Mapping을 통한 트랜잭션 코드 해독

커널 레벨에서 Binder IPC 메타데이터를 수집하더라도, Binder 통신의 특성상 관측되는 호출 정보는 메서드명이 아닌 정수형 transaction code로 표현되므로 의미론적 단절이 발생한다. Fig. 4의 AIDL Decoding 과정과 같이, 본 프레임워크는 이를 해석하기 위해 사전에 구축한 XR AIDL 제어 메서드 매핑 테이블을 활용한다. 해당 매핑 테이블은 Android XR 환경의 네이티브 라이브러리인 libopenxr_android.so를 분석하여 구성되며, Target App에서 XR Broker 방향으로 관측된 transaction code를 known Broker control method와 대조하는 데 사용된다.

구체적으로, 앞선 흐름 식별 단계에서 app-to-broker 방향의 Binder transaction으로 분류된 code만을 해독 대상으로 사용한다. 이는 XR Broker가 다른 앱이나 시스템 구성요소의 요청도 함께 처리할 수 있기 때문에, 단순히 Broker 관련 transaction이 존재한다는 사실만으로 대상 앱의 실행 흐름으로 해석하는 것을 방지하기 위함이다. 수집된 transaction code가 AIDL mapping table에 존재하는 경우, 해당 code는 createNode와 같은 사람이 해석 가능한 XR Broker 제어 메서드명으로 복원된다. 반대로 mapping table에 존재하지 않는 code는 unknown으로 유지하여, 후속 평가에서 과도한 의미 부여가 발생하지 않도록 한다.

이 과정을 통해 단순한 Binder IPC 관측 결과는 raw metadata 수준을 넘어, XR Broker control method 수준의 semantic evidence로 변환된다. 다만 이 해석은 Target App이 XR Broker의 특정 제어 메서드에 도달했음을 나타내는 것이며, 실제 센서 payload의 반환이나 민감 데이터 접근을 직접 입증하는 것은 아니다. 최종적으로 복원된 method name, transaction code, 대상 permission, 관측 component 정보는 candidate-level 평가 단계로 전달되어, 정적 후보가 런타임에서 Brok

er-level semantic evidence로 연결되는지 판단하는 근거로 활용된다.

VII. 분석 결과 및 평가

7.1 실험 목적 및 방법

본 실험의 목적은 eSPADE-XR이 정적 분석으로 도출된 XR 권한 후보를 런타임 관측과 교차 검증하여, 해당 후보가 Target App과 XR Broker 사이의 제어 경로 및 AIDL 기반 제어 메서드 증거로 연결되는지를 확인하는 것이다. 최종 평가는 실제 센서 데이터 접근 여부가 아니라, 후보 단위에서 XR Broker-level semantic evidence가 관측되는지를 기준으로 수행하였다.

실험에는 Android Studio의 XR Headset Emulator(Android 14 기반)를 사용하였다. 데이터셋은 Google Play Store에서 수집한 Android XR 앱 31종으로 구성되며, 이 중 22종은 'Made for XR', 9종은 'XR Ready'로 분류된다. APK 분해 및 패키지 기준 정리 과정에서 분석 단위는 34개의 app_id로 구성되었다.

정적 분석 단계에서는 Manifest 파싱과 IL2CPP/Smali 다계층 스캔을 통해 6개 민감 XR 권한에 대한 204개의 앱-권한 단위 정적 후보를 생성하고, 이 중 17개 app_id에서 도출된 51개의 primary runtime candidate를 런타임 관측 대상으로 선별하였다. 이후 후보 컴포넌트에 대해 총 239회의 관측 attempt를 수행하고, 실제 대상 앱 PID 기준으로 eBPF Binder IPC 관측기를 부착하여 Target App과 XR Broker 사이의 transaction metadata를 수집하였다. 수집된 transaction code는 AIDL mapping table과 대조한 뒤 후보 단위로 통합하여 Weak 또는 Strong evidence level로 평가하였다.

Table 1. Dataset and Analysis Units

Item	Count
Collected Android XR apps	31
"Made for XR" apps	22
"XR Ready" apps	9
Analyzed app IDs	34

7.2 정적-동적 교차검증 및 Broker 증거 분류

정적 분석으로 선별된 51개의 primary runtime candidate를 대상으로 후보 컴포넌트 기반 런타임 관측을 수행한 결과, 총 239회의 실행 attempt 중 215회에서 유효한 관측 결과를 확보하였다. 이를 후보 단위로 통합한 결과, 33개 후보는 Strong, 16개 후보는 Weak으로 분류되었고, 2개 후보는 collector output 누락으로 평가에서 제외되었다. Strong은 Target App에서 XR Broker 방향의 Binder transaction이 관측되고, 해당 transaction code가 AIDL mapping을 통해 알려진 XR Broker 제어 메서드로 복원된 경우를 의미한다. 본 실험에서 Strong으로 분류된 후보들의 transaction code는 주로 createNode와 같은 Broker-level control method로 복원되었다. 반면 Weak은 정적

Table 2. Candidate-level evidence classification results

Level	Count	Ratio
Strong	33	64.7%
Weak	16	31.4%
Invalid	2	3.9%
Total	51	100%

- Invalid candidates are excluded from Strong/Weak grading

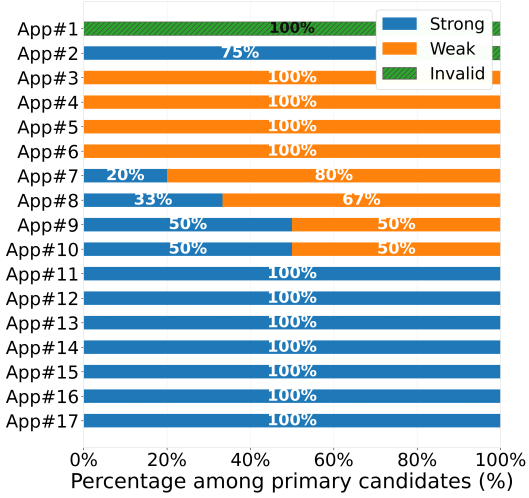

Fig. 6. App-wise Strong/Weak/Invalid distribution for primary runtime candidates

후보는 존재하지만, 테스트 실행 조건에서 app-to-broker transaction과 AIDL 기반 메서드 증거가 함께 확인되지 않은 경우이다. 따라서 Weak은 기능 미사용이나 보일러플레이트 확정을 의미하지 않으며, 현재 관측 조건에서 Broker-level semantic evidence가 확인되지 않았음을 나타낸다. Fig. 6은 primary runtime candidate가 존재하는 app_id만을 대상으로 Strong, Weak, Invalid 분포를 나타낸다. Primary runtime candidate가 없는 app_id는 제외되었으며, 해당 결과는 앱별 위험도 순위가 아니라 평가 대상 후보의 evidence 분포를 보여주는 보조 결과이다.

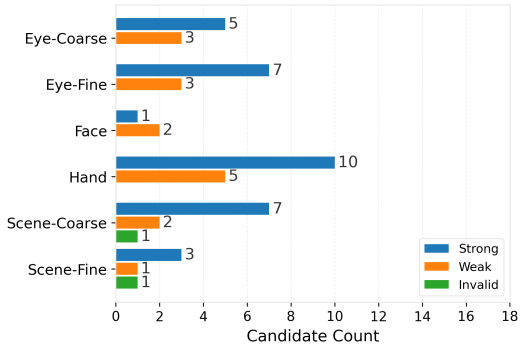

Fig. 7. Permission-wise distribution of candidate-level evidence

7.3 권한 범주별 Candidate Evidence 분포

권한 범주별로 candidate-level evidence를 분석한 결과, Strong evidence는 단일 권한에만 집중되지 않고 HAND_TRACKING, EYE_TRACKING_FINE, SCENE_UNDERSTANDING_COARSE 등 복수의 XR permission category에서 관측되었다. 구체적으로 HAND_TRACKING은 15개 후보 중 10개가 Strong, 5개가 Weak으로 분류되었고, EYE_TRACKING_FINE은 10개 후보 중 7개가 Strong, 3개가 Weak으로 분류되었다. SCENE_UNDERSTANDING_COARSE는 10개 후보 중 7개가 Strong, 2개가 Weak, 1개가 Invalid로 나타났다. 반면 FACE_TRACKING은 전체 후보 수가 3개로 제한적이므로, 해당 결과를 권한 범주 간 위험도 차이로 해석하기는 어렵다. Fig. 7은 이러한 권한별 분포를 나타내며, 이는 권한별 위험도 순위가 아니라 각 XR permission category에서 정적 후보가 Broker-level semantic evidence로 연결된 정도를 보여주는 보조 결과이다.

VIII. 고 찰

eSPADE-XR이 검증하는 대상은 실제 sensor payload의 반환 여부나 악성 행위 자체가 아니라, 정적으로 도출된 XR 권한 후보가 런타임에서 Target App과 XR Broker 사이의 제어 경로 및 AIDL 기반 제어 메서드 증거로 연결되는지 여부이다. 따라서 Strong evidence는 앱이 XR Broker의 알려진 control method에 도달했음을 의미하지만, 실제 물리 센서 데이터 접근이나 개인정보 유출을 확정하지 않는다. 또한 본 실험은 Android XR Headset Emulator 환경에서 수행되어 실제 물리 센서, OEM HAL, kernel driver, shared memory 또는 FMQ 기반 데이터 스트림까지 관측하지 못했다. AIDL transaction mapping 역시 Android Runtime, SDK/API 버전, OEM별 Broker 구현 차이에 따라 달라질 수 있으며, 표적 실행은 정상 사용자 플로우 전체를 재현하지 못한다. 향후 실제 XR 기기와 더 큰 앱 집합을 대상으로 반복 검증을 수행하고, OEM별 Broker 및 AIDL mapping 차이를 보완할 필요가 있다.

IX. 결 론

본 논문은 Android XR 플랫폼에서 정적 권한 선언과 실제 런타임 동작 사이의 정합성을 검증하기 어렵다는 문제와, XR Broker 기반 IPC 구조로 인해 동적 관측 결과의 의미 해석이 제한되는 문제를 다루었다. 이를 위해 eSPADE-XR은 정적 분석 결과를 최종 판단이 아닌 런타임 검증 후보로 재정의하고, eBPF 기반 Binder IPC 관측과 AIDL transaction mapping을 결합하여 Target App과 XR Broker 사이의 제어 경로 및 Broker-level semantic evidence를 분석하였다. 실험 결과, 정적 후보 중 일부가 런타임에서 AIDL 기반 제어 메서드 증거로 연결됨을 확인하였으며, 이를 통해 XR 권한 검증에서 정적 분석과 동적 관측을 결합하는 접근의 가능성을 보였다.

다만 본 연구는 Android XR Headset Emulator 환경에서 수행되었으므로 실제 물리 센서, HA

L, shared memory 또는 FMQ 기반 데이터 경로까지 확인하지는 못했다. 향후에는 실제 XR 디바이스 및 AR 글래스 환경에서 재검증을 수행하고, 하위 데이터 경로와 Broker 구현 차이를 함께 분석할 계획이다. 또한 의미론적 권한 모델과 eBPF/LSM 기반 런타임 정책 집행을 결합하여, XR 권한 사용을 보다 세밀하게 관리하는 방향으로 확장하고자 한다.

References

[1] R. Acheampong, T.C. Balan, D.M. Popovici, E. Tuyishime, A. Rekeraho, and G.D. Voinea, "Balancing usability, user experience, security and privacy in XR systems: a multidimensional approach," International Journal of Information Security, vol. 24, no. 3, pp. 112 Apr. 2025.

[2] S. Mansour, V. Winterhalter, F. Alt, and V. Paneva, "A multi-layered privacy permission framework for extended reality," Proceedings of the 2025 New Security Paradigms Workshop, pp. 50-65, Aug. 2025.

[3] V. Nair, W. Guo, J. Mattern, R. Wang, J.F. O'Brien, L. Rosenberg, and D. Song, "Unique identification of 50,000+ virtual reality users from head & hand motion data," 32nd USENIX Security Symposium (USENIX Security 23), pp. 895-910, Aug. 2023.

[4] D. Cayir, A. Acar, R. Lazzeretti, M. Angelini, M. Conti, and S. Uluagac, "Augmenting security and privacy in the virtual realm: an analysis of extended reality devices," IEEE Security & Privacy, vol. 22, no. 1, pp. 10-23, Jan.-Feb. 2024.

[5] R. Acheampong, D.M. Popovici, T.C. Balan, A. Rekeraho, and I.A. Oprea, "A cybersecurity risk assessment for enhanced security in virtual reality," Information, vol. 16, no. 6, pp. 430 Jun. 2025.

[6] V.C. Nair, G. Munilla Garrido, and D. Song, "Going incognito in the metaverse: achieving theoretically optimal privacy-usability tradeoffs in VR," Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, pp. 1-16, Oct. 2023.

[7] A. Giaretta, "Security and privacy in virtual reality: a literature survey," Virtual Reality, vol. 29, no. 1, pp. 10, Dec. 2024.

[8] C. Warin, V. Pak, and D. Reinhardt, "Privacy perceptions across the XR spectrum: an extended reality cross-platform comparative analysis of a virtual house tour," Proceedings on Privacy Enhancing Technologies, vol. 2025, no. 1, pp. 150-168, Jul. 2025.

[9] R. Baalous, A. Althobaiti, D. Alyoubi, R. Alzahrani, and M. Aljohani, "Detecting the inconsistency between Android apps' data collection and Google Play's data safety using static analysis," Cybernetics and Information Technologies, vol. 25, no. 1, pp. 110-125, Mar. 2025.

[10] Z. Wu, H. Lee, and S.U.J. Lee, "An empirical study on the impact of permission smell in Android applications," Journal of The Korea Society of Computer and Information, 26(6), pp. 89-96, Jun. 2021.

[11] C. Yue, K. Chen, Z. Guo, J. Dai, X. Sun, and Y. Yang, "What's done is not what's claimed: detecting and interpreting inconsistencies in app behaviors," Network and Distributed System Security Symposium 2025, Feb. 2025.

[12] Android Open Source Project, "Binder overview," https://source.android.com/docs/core/architecture/ipc/binder-overview, accessed Jun. 2026.

[13] Android Open Source Project, "AIDL o

verview," https://source.android.com/docs/core/architecture/aidl, accessed Jun. 2026.
[14] Android Open Source Project, "Application Sandbox," https://source.android.com/docs/security/app-sandbox, accessed Jun. 2026.
[15] The Khronos OpenXR Working Group, "The OpenXR Specification," version 1.0.34, Sec. 2.5, Feb. 2024, https://registry.khronos.org/OpenXR/specs/1.0-khr/pdf/xrspec.pdf, accessed Jun. 2026.
[16] Android Open Source Project, "Hardware abstraction layer (HAL) overview," https://source.android.com/docs/core/architecture/hal, accessed Jun. 2026.
[17] Android Developers, "Sensor," https://developer.android.com/ndk/reference/group/sensor, accessed Jun. 2026.
[18] Android Open Source Project, "Fast Message Queue with AIDL," https://source.android.com/docs/core/architecture/aidl/fmq, accessed Jun. 2026.
[19] eBPF.io, "What is eBPF? An introduction and deep dive into the eBPF technology," https://ebpf.io/what-is-ebpf/, accessed Jun. 2026.
[20] Android Open Source Project, "Extend the kernel with eBPF," https://source.android.com/docs/core/architecture/kernel/bpf, accessed Jun. 2026.
[21] G. Munilla Garrido, V. Nair, and D. Song, "SoK: data privacy in virtual reality," Proceedings on Privacy Enhancing Technologies, vol. 2024, no. 1, pp. 21-40, Jul. 2024.
[22] H. Guo, H.N. Dai, X. Luo, Z. Zheng, G. Xu, and F. He, "An empirical study on Oculus virtual reality applications: security and privacy perspectives," Proceedings of the IEEE/ACM 46th International Conference on Software Engineering, pp. 1-13, Apr. 2024.
[23] C. Zuo and Z. Lin, "Playing without paying: detecting vulnerable payment verification in native binaries of Unity mobile games," 31st USENIX Security Symposium (USENIX Security 22), pp. 3093-3110, Aug. 2022.
[24] D. Rodriguez, J.A. Calandrino, J.M. Del Alamo, and N. Sadeh, "Privacy settings of third-party libraries in Andriod apps: a study of Facebook SDKs," Proceedings on Privacy Enhancing Technologies, vol. 2025, no. 2, pp. 173-187, Jul. 2025.
[25] Android Developers, "NdkBinder," https://developer.android.com/ndk/reference/group/ndk-binder, accessed Jun. 2026.
[26] Android Open Source Project, "aidl.cpp," https://android.googlesource.com/platform/system/tools/aidl/%2B/02e012eddd8149f9bd48071591ecca5b4b700d92/aidl.cpp, accessed Jun. 2026.
[27] R. Trimananda, H. Le, H. Cui, J.T. Ho, A. Shuba, and A. Markopoulou, "OVRseen: auditing network traffic and privacy policies in Oculus VR," 31st USENIX Security Symposium (USENIX Security 22), pp. 3789-3806, Aug. 2022.
[28] D. Suo, L. Xue, W. Huang, R. Tan, and G. Sun, "Assessing the capability of Android dynamic analysis tools to combat anti-runtime analysis techniques," Journal of Systems and Software, vol. 234, pp. 112747, Apr. 2026.
[29] J.Y. Kim, C. Zuo, Y. Zhao, and Z. Lin, "AUTOVR: automated UI exploration for detecting sensitive data flow exposures in virtual reality apps," 34th USENIX Security Symposium (USENIX Security 25), pp. 7937-7955, Aug. 2025.

〈저자소개〉

주 효 중 (Hyojoong Ju) 학생회원
2021년 3월~현재 : 인천대학교 컴퓨터공학부 학사과정
〈관심분야〉 XR 보안, 클라우드 보안

이 승 수 (Seungsoo Lee) 종신회원
2014년 2월 : 숭실대학교 컴퓨터학부 학사
2016년 2월 : KAIST 정보보호대학원 석사
2020년 8월 : KAIST 정보보호대학원 박사
〈관심분야〉 클라우드 보안, 네트워크 보안