Transactions of KSAE, Vol. 34, No. 7, pp.785-805 (July, 2026)

Check for updates

Copyright © 2026 KSAE / 248-05
pISSN 1225-6382 / eISSN 2234-0149
DOI http://dx.doi.org/10.7467/KSAE.2026.34.7.785

Physical AI 보안을 위한 TARA 프레임워크 개선 연구: ADS 적용 사례

우 지 훈[1] · 박 강 현[2] · 이 준 형[*2]

세종대학교 컴퓨터공학과[1] · (재)한국화학융합시험연구원 인공지능융합센터[2]

Enhancing the TARA Framework for Physical AI Security: Application to ADS

Jihun Woo[1] · Kanghyun Park[2] · Junhyeong Lee[*2]

[1]Department of Computer Engineering, Sejong University, Seoul 05006, Korea
[2]AI Convergence Center, Korea Testing & Research Institute, 20 Pangyo-ro, 289 beon-gil, Bundang-gu, Seongnam-si, Gyeonggi 13448, Korea

(Received 10 December 2025 / Revised 29 January 2026 / Accepted 6 February 2026)

Abstract : Advances in automated driving systems (ADS) have significantly improved safety and convenience, but they have also introduced new security threats stemming from platforms built on physical artificial intelligence (physical AI). In line with this, Conventional Threat Analysis and Risk Assessment (TARA) methods have been proven effective at identifying traditional software and hardware vulnerabilities; however, they failed to fully capture AI-specific threats. Motivated by this gap, this study proposed an improved TARA framework tailored to physical AI security and validated it through an ADS case study.

Methodologically, we conducted TARA using the publicly available software system architecture of the NVIDIA Jetson Nano and the ADS architecture defined in ISO/TS 5083. In terms of threat identification, we applied threat modeling while concurrently incorporating AI-specific threat catalogs by referencing MITRE ATLAS and the OWASP Top 10 for LLM Applications. These approaches yielded a comprehensive analysis method that encompassed conventional and AI-specific threats that may arise in physical AI environments.

Our analysis resulted in a comprehensive checklist for security considerations, which derived security requirements, offering concrete, practical guidance applicable across the full life cycle of ADS. Particularly, the proposed method enables a more exhaustive and systematic threat analysis by identifying AI-driven attack vectors that conventional TARA approaches may overlook.

The contribution of this work is to equip industry and research organizations in the expanding autonomous-driving sector with an enhanced TARA framework for executing threat analysis and risk assessment. Accordingly, this is expected to strengthen the cybersecurity of physical AI systems like ADS and support the dissemination of safe and trustworthy autonomous-driving technologies. The case study leveraging the Jetson Nano and the ISO/TS 5083 architecture demonstrates the effectiveness of the proposed method and suggests its utility as a reference model for future ADS cybersecurity assessments.

Key words : Physical AI(물리적 AI), TARA(위협 분석 및 위험 평가), ADS(자율주행시스템), Threat modeling(위협모델링), CRA(사이버복원력법), ISO/SAE 21434(차량 사이버보안 엔지니어링)

---

[*]A part of this paper was presented at the KSAE 2025 Fall Conference and Exhibition

[*]Corresponding author, E-mail: jh0302@ktr.or.kr

[*]This is an Open-Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License(http://creativecommons.org/licenses/by-nc/3.0) which permits unrestricted non-commercial use, distribution, and reproduction in any medium provided the original work is properly cited.

1. 서 론

최근 Physical AI 보안이 학계 · 산업계에서 급부상하고 있다. 물리 세계의 미세한 신호 · 환경 조작이 AI 지각 · 의사결정에 직접적인 안전 위험으로 증폭되는 사례가 빠르게 축적되고 있기 때문이다. 예컨대 레이저로 유도한 음향 간섭이 MEMS(Micro Electro Mechanical Systems) 자이로를 교란해 카메라 안정화를 무너뜨리면, 컴퓨터비전 기반 차량 · 드론의 객체검출 성능(mAP)이 평균 41 % 수준까지 급락하는 등 물리-사이버 연계 공격의 실효성이 실험적으로 확인되고 있다.[1] 이러한 경향은 ADS에 특히 치명적이다. ADS는 카메라 · 레이더 등 복수 센서, 고성능 엣지 컴퓨팅, AI 알고리즘, 차량과 사물 연결을 통합하는 전형적 CPS(Cyber-Physical System) 구조를 이룬다.[41,47] 따라서 물리 환경 교란이 지각-계획-제어 루프로 전파된다.[2,5] 더불어 현대 차량은 Wi-Fi · GPS · 카메라 스트리밍 · 레이더 · LiDAR 등 외부 매체 의존성이 급증해 Attack Surface가 확대되고 있어, 생애주기 초기에 체계적 위협모델링과 위험평가(Threat analysis and Risk assessment, TARA)를 반복 적용해야 한다는 필요성이 지속적으로 제기된다.[3,17] 실제로 최신 자율주행 보안 문헌은 LiDAR 블라인딩 · 가짜 객체 투사 같은 환경 교란형 물리 공격과 V2X(Vehicle to Everything) 조작 · 서비스거부 등 네트워크 기반 위협이 현실적 위험임을 재확인하고 있다.[2,4,14-16] 이는 'Physical AI 보안'을 ADS 맥락에서 우선순위를 높게 다룰 필요성을 뒷받침한다.

기존의 위협 모델링 기법은 주로 소프트웨어 및 IT 시스템을 대상으로 발전해 왔다.[2,44,45] 이러한 전통적인 접근법은 CPS의 Security-By-Design 요구와 안전 목표를 충분히 반영하지 못한다.[2,44,45] CPS는 AI 알고리즘이 실시간으로 판단을 수행하고, 센서와 액추에이터가 물리적 환경과 지속적으로 상호작용하는 복잡한 구조를 가진다.[2,44] 본 연구에서는 이처럼 AI가 물리적 상호작용과 제어의 주체가 되는 고도화된 CPS를 'Physical AI 시스템'으로 통칭한다. 이러한 Physical AI 시스템의 보안을 위해서는 기존 IT 중심 위협 모델링 분석 범위를 물리적 상호작용 영역까지 확장하여 적용할 필요가 있다.[2,6-8,44,45]

본 연구는 이러한 문제의식 위에서 ISO/SAE 21434 표준 TARA 프로세스를 기반으로 Physical AI 시스템의 특성을 반영할 수 있도록 확장된 위협 분석 절차를 제안한다.[50] Fig. 1은 본 연구에서 제안하는 분석 절차의 전체 흐름을 보여준다. 이 절차는 ISO/SAE 21434의 TARA 절차를 기준으로 하여, 전통 위협모델(STRIDE)과 더불어 MITRE ATLAS 및 OWASP LLM(Large Language Model) Top 10을 병렬 적용하여 AI 특화 위협을 포괄적으로 반영한다.[6-8] 이를 토대로 보안 요구사항과 체크리스트를 도출하고자 한다. 이를 위해 2장 선행연구에서는 ISO/SAE 21434에서 명시하는 TARA에 대해서 소개 한다. 3장 본론에서는 앞서 제안한 확장된 위협 분석 절차를 적용하여 ADS에 대한 TARA를 진행한다. 4장 검증에서는 본론에서 도출된 체크리스트와 글로벌 차량 사이버보안 법규 간 비교 분석을 통해 연구 산출물의 사회적 기여 여부를 확인한다. 마지막으로 5장 결론에서는 연구의 내용을 요약하고 향후 연구방향을 제시한다.

2. 배경 및 선행연구

2.1 위협 모델링 확장 및 보완 연구

기존 사이버보안 위협 분석 기법들을 새로운 맥락에 통합하거나 확대 적용하려는 연구가 활발히 이루어지고 있다. S. Ghosh 등은 자율 주행 자동차의 인지 시스템을 대상으로 TARA 절차 기반 STPA-Sec과 STRIDE를 통합하는 방법을 제안하였다.[2] 이 연구에서는 STPA-Sec 기반으로 안전 관점의 인과 시나리오를 도출하고 보안 관점에서 공격 가능 지점을 식별한다.[2] 또한 STPA-Sec의 결과를 STRIDE의 공격 분류 체계로 맵핑하여 ISO/SAE 21434의 위협 분석 단계와 연계하였다.[2]

F. V. Jedrzejewski 등은 LLM이 통합된 응용 시스템

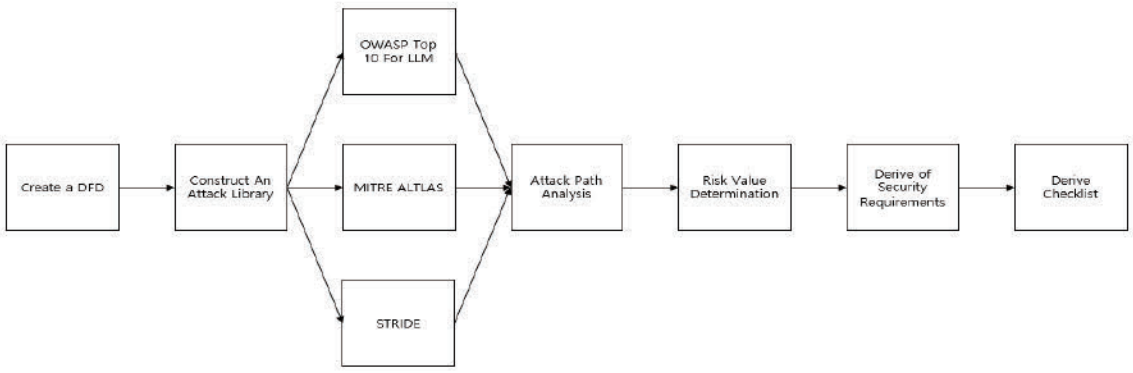


Fig. 1 Threat Analysis and Risk Assessment(TARA) framework process for Physical AI

(Large Language Model Integrated Applications, LIA)의 공격 표면이 기존 소프트웨어보다 크게 확장됨을 지적한다.[6,7] 특히 기존 위협 모델링 도구들이 수작업 위주이고 LLM 특유 위협을 충분히 반영하지 못한다고 분석한다.[6,7] 이들은 이를 극복하기 위해 OWASP Top 10 for LLM, MITRE ATLAS 등 AI 전용 보안 지식베이스의 활용을 제안한다.[6,7] 또한 기존 위협 모델과 시스템 아키텍처 정보를 통합하여 지속적으로 위협 모델을 생성·업데이트하는 RAG(Retrieval-Augmented Generation) 기반 도구를 제안한다.[6,7] 이 연구는 위협모델링 과정을 보안 전문가 개입을 줄이고 시간과 비용을 절감하는 것을 목표로 한다.[6,7]

또한 S. M. Khalil 등은 소프트웨어 개발 프로젝트에서 널리 사용되는 위협 모델링이 CPS분야에서는 아직 체계화되지 않았음을 지적하였다.[45] 사이버 공간과 물리 공간의 상호작용으로 인해 CPS 환경의 위협 식별이 복잡하기 때문이다.[45] 이에 따라 해당 연구에서는 STRIDE 기법을 CPS에 적용하기 위한 세부 절차와 단계별 가이드를 제안하였다.[45] 사례 연구를 통해 CPS 분야에서도 체계적인 위협 분석이 가능함을 보였다.[45] 다만 이 접근법은 AI 알고리즘이 포함된 Physical AI 시스템의 위협 요인은 고려하지 않았다.

2.2 UN Regulation No. 155

UN Regulation No. 155(UN R 155)는 UN/WP.29(World Forum for Harmonization of Vehicle Regulations)에서 제정한 차량 사이버보안 법규이다.[48] 해당 법규는 완성차 제조사가 차량 형식 승인(Vehicle Type Approval, VTA)을 획득하기 위해 개발부터 양산 후 단계까지의 사이버보안 관리 체계(CyberSecurity Management System, CSMS) 구축을 의무화하고 있다.[17,48]

UN R 155는 이러한 보안 활동의 기술적 가이드라인으로 ISO/SAE 21434를 지정하고 있다.[48] 규정 5.3.1항 (a)에서는 승인 당국 및 기술 서비스 기관이 갖추어야 할 역량 요건으로 "자동차 위험 평가 지식"을 규정하며, 각주로 ISO/SAE 21434를 명시하였다.[48] 이는 ISO/SAE 21434 기반의 TARA수행 및 보안 설계가 실제 법규 인증 심사에서 요구되는 핵심 기준임을 명시한다.

2.3 ISO/SAE 21434 TARA

ISO/SAE 21434는 도로 차량 사이버보안 위협 관리 및 요구사항에 대한 표준을 정의한다. ISO/SAE 21434는 위협 식별 및 위험을 평가를 위해 TARA를 수행해야 한다고 명시한다. 또한 TARA 수행을 위한 7가지 구성요소를 명시한다.[5] 7가지의 구성 요소 중 attack feasibility rating(공격 가능성 평가)과 risk value determination(위험 값 결정)은 조직에 따라 평가 기준이 다른 항목으로, 본 연구에서는 다루지 않는다. 본 장에서는 7가지의 구성 요소 중 2가지 구성 요소를 제외한 5가지 구성 요소에 대해서 소개한다.

2.3.1 Asset identification (자산 식별)

자산을 식별하는 단계이다. 자산은 하나 이상의 정보 보안 속성을 가진다. 정보 보안 속성은 기밀성, 무결성, 가용성으로 구성된다. 자산은 위협 시나리오가 발생할 경우 잠재적으로 피해를 초래할 수 있다. 피해가 발생하는 시나리오에 대해서 Damage Scenario(피해 시나리오)로 정의한다.[5]

2.3.2 Threat Scenario identification (위협 시나리오 식별)

위협 시나리오를 식별하는 단계이다. 자산 식별 단계에서 식별한 자산을 토대로 위협을 식별한다. 식별한 위협을 근거로 위협 시나리오를 작성한다. 표준에 따르면 위협 식별을 위해서 EVITA, TVRA, PASTA, STRIDE 등과 같은 위협모델링 프레임워크를 사용할 수 있다.[5]

2.3.3 Impact Rating (영향 평가)

피해 시나리오에 대해서 분류 및 도로 이용자에게 미치는 영향을 평가한다. 분류 카테고리는 안전(Safety, S), 재정(Financial, F), 운영(Operational, O), 개인정보(Privacy, P)이다. 영향 평가는 Severe(심각), Major(주요), Moderate(보통), Negligible(경미)의 단계로 평가를 진행한다. ISO/SAE 21434의 Annex F에서 예시 피해 영향 평가표를 제공하지만 각 조직이 자체적으로 평가 기준을 추가해야한다고 권고한다.[5] 따라서 본 연구에서는 영향 평가는 다루지 않는다.

2.3.4 Attack Path analysis (공격 경로 분석)

공격 경로 분석은 위협 시나리오를 바탕으로 실제로 공격이 수행될 수 있는 경로를 식별하는 단계이다. ISO/SAE 21434에서 권장하는 절차는 Attack tree/graph을 이용한 Top-down 방식과 발견된 취약점으로부터 경로를 구성하는 Bottom-up 방식이 있다. 분석한 공격 경로는 실현 할 수 있는 위협시나리오와 매핑 해야 한다.[5]

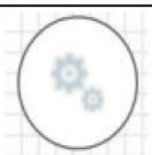
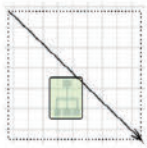
2.3.5 Risk Treatment Decision (위험 처리 결정)

위험을 어떻게 처리할지를 결정하는 단계이다. 각 위협 시나리오에 대해서 조직에 따라 평가한 위험 값을 고려하여, Table 1에 제시되는 옵션 중 하나 이상을 결정해야한다.[5] 본 연구에서는 위험 값 결정 단계를 진행하지 않기 때문에 모든 위협 시나리오에 대해서 보안 요구사항을 도출한다.

Table 1 Risk treatment options

Treatment	Description
Avoiding the risk	Eliminate the source of risk by discontinuing the hazardous activity or decommissioning the associated function.
Reducing the risk	Specify one or more security objectives and apply corresponding security requirements to reduce the likelihood or impact of a successful attack.
Sharing the risk	Share the risk with a third party through contractual agreements or insurance.
Retaining the risk	Risk acceptance due to infeasibility or lack of cost-effectiveness of control implementation.

Table 2 Description of data flow diagram

Element	Shape	Description
External entity		External entities representing the sources and destinations of data outside the system boundary.
Process		Internal system functions representing actions implemented through software code.
Data flow		Data communication representing how information moves between processes and data stores.
Data store		Data stores that hold and access information on a temporary or persistent basis.

3. 본 론 : 확장된 TARA 절차의 ADS 적용

3.1 Create a DFD(Data Flow Diagram)

ISO/SAE 21434에서 제시하는 TARA의 첫 단계인 자산 식별을 위해 Microsoft의 Threat modeling tool을 이용하여 DFD를 작성하였다.[5] 작성된 DFD의 주요 구성요소들은 모두 자산이다. DFD의 주요 구성요소는 Table 2와 같다. DFD는 시스템이나 프로세스에 대해 점진적으로 더 많은 세부 정보를 표시하기 위해 정도에 따라 Level을 구분한다. Level 0은 Context diagram이라 부르는 레벨로, 전체 시스템을 단일 프로세스로 시각화한 가장 간단하고 기본적인 레벨이다. Level 1은 Level 0의 단일 프로세스들의 기능 및 Data flow 경로에 대한 자세한 정보를 제공하고, 하위 프로세스로 세분화한다. Level 2는 새로운 하위 프로세스와 Data flow 및 Data store와 상호 작용 및 관계를 추가하여 세부 정보를 제공하고, 프로세스 내부 작업에 대한 복잡한 보기를 제공한다.[10]

3.1.1 ISO/TS 5083 기반 ADS 기능 분석

ADS의 DFD를 작성하기 위하여 ISO/TS 5083에서 제시하는 기능 아키텍처를 분석하였다. ISO/TS 5083의 Fig. 2에 제시된 기능적 계층 구조는 지각(Perception), 계획(Planning), 제어(Control)의 3단계로 구성된다.[9] 지각(Perception) 단계는 LiDAR, 카메라, 레이더 등 다양한 센서로부터 획득한 데이터를 융합하는 Sensor fusion 기능과 자차의 위치를 추정하는 Localization 기능을 포함한다. 이러한 정보로부터 특징 추출, 객체 검출 및 분할, 차선 및 신호 인식 등의 고수준 인지 기능이 수행된다. 계획(Planning) 단계에서는 지각 결과를 기반으로 주행 환경과 주변 객체의 상태를 예측하고, 주행 상황을 해석하여 경로 및 속도 계획, 행동 결정을 수행한다. 마지막으로 제어(Control) 단계는 계획된 행동에 따라 차량 Actuator (조향, 구동, 제동 등)를 제어하기 위한 명령을 생성하여 실제 차량의 동작을 수행한다. 이러한 기능 구조를 실제

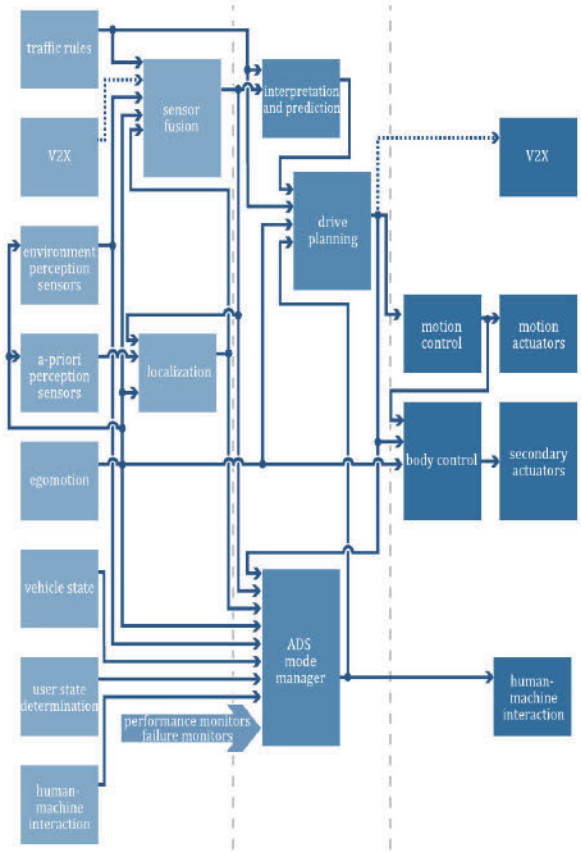


Fig. 2 Example architecture of the intended functionality, including monitors[9]

구현 수준에서 구체화하기 위하여 NVIDIA Jetson Nano 플랫폼을 기반으로 DFD를 설계하였다. NVIDIA Jetson Nano 플랫폼은 공개된 소프트웨어 스택(Fig. 3)을 제공한다. NVIDIA Jetson Nano 플랫폼에서 ADAS(Advanced Driver Assistance System) 기능을 구현한 오픈소스 및 연구 사례는 딥러닝 기반 인지, 센서 데이터 처리, 실시간 제어 알고리즘 구현 가능성을 입증한다.[42-43] 이는 ISO/TS 5083에서 정의한 ADS 기능 구조를 실질적으로 재현·확장하기에 적합한 플랫폼으로 평가된다. 따라서 본 연구에서는 ISO/TS 5083의 기능 아키텍처를 상위 참조 모델로, NVIDIA Jetson Nano 플랫폼의 소프트웨어 스택을 하위 참조 구조로 설정하여 ADS의 DFD를 작성하였다.

3.1.2 DFD 작성 결과

Figs. 4 ~ 6은 각각 Level 0, 1, 2에 해당하며, NVIDIA Jetson Nano 플랫폼 위에서 동작하는 ADS의 DFD이다. Fig. 4는 Level 0로써 NVIDIA Jetson Nano 플랫폼 위에서 External Entity와 상호작용 하는 Process들만 표현하였다. Fig. 5는 Level 1로써 지각(Perception), 계획(Planning), 제어(Control), 보조 기능들을 세분화 하여 표현 하였다. Fig. 6은 Level 2로써 각 기능들에 대해 더욱 세분화 하여 Data flow 및 Data store와 상호 작용에 대한 세부 정보를 제공하였다. Level 2에서 식별한 자산은 10개의 Process, 7개의 External entity, 5개의 Data store, 30개의 Data flow 이다. 전체 DFD 그림은 공개 저장소를 통해 확인할 수 있다.[49]

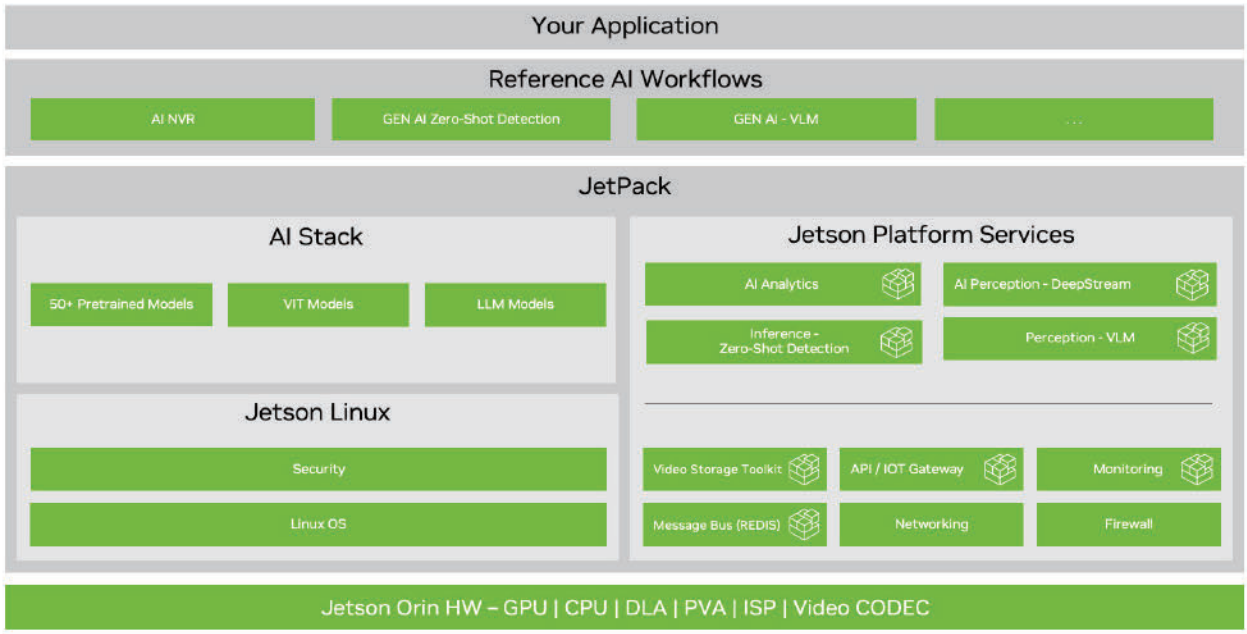


Fig. 3 Nvidia Jetson Nano software stack[46]

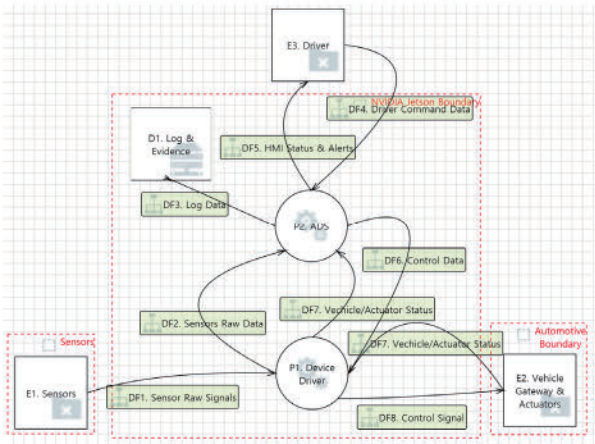


Fig. 4 Level 0 for automated driving systems

3.2 Construct An Attack Library

Attack Library는 분석 대상 시스템에서 발생 가능한 보안 위협을 체계적으로 수집 · 정규화한 지식 저장소이며, 이후 요소별 위협 식별과 시나리오 도출의 근거로 활용된다.

3.2.1 수집 범위와 절차

일반적으로 Attack Library는 CVE(Common Vulnerabilities and Exposures), CWE(Common Weakness Enumeration), Paper, Technical Report 등을 바탕으로 수집하며 DFD에서 식별한 자산 · Data flow의 키워드들을 조합하여 검색하였다. 본 연구에서는 “Sensor”, “Sensor Fusion”, “Localization”, “LLM”, “NVIDIA Jetson”, “Jetson Driver”, “Linux Kernel” 등과 같은 키워드를 조합하여 42개의 CWE, 76개의 CVE, 47개의 Paper, 3개의 Technical report, 2개의 Standard를 수집하였다.

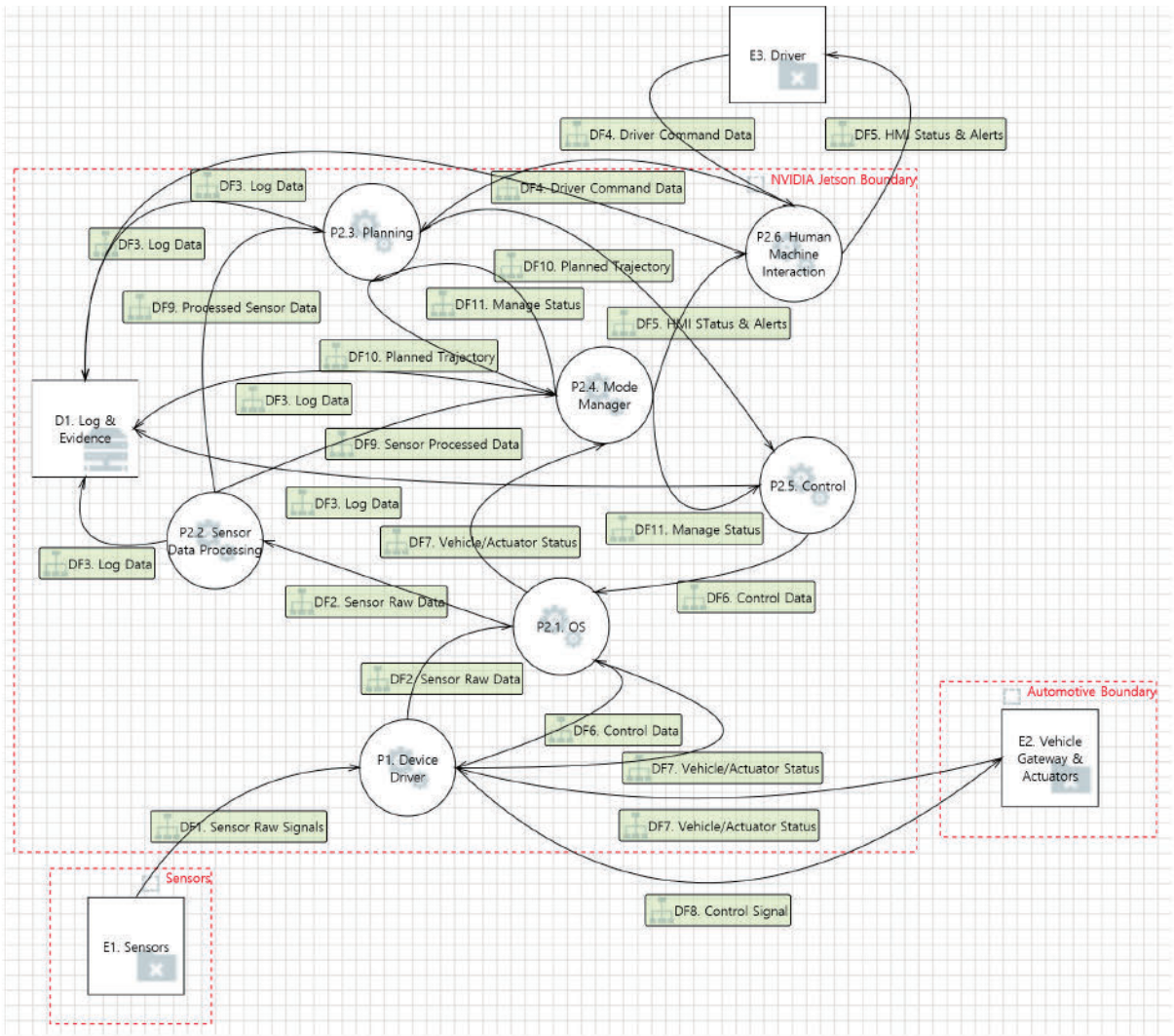


Fig. 5 Level 1 for automated driving systems

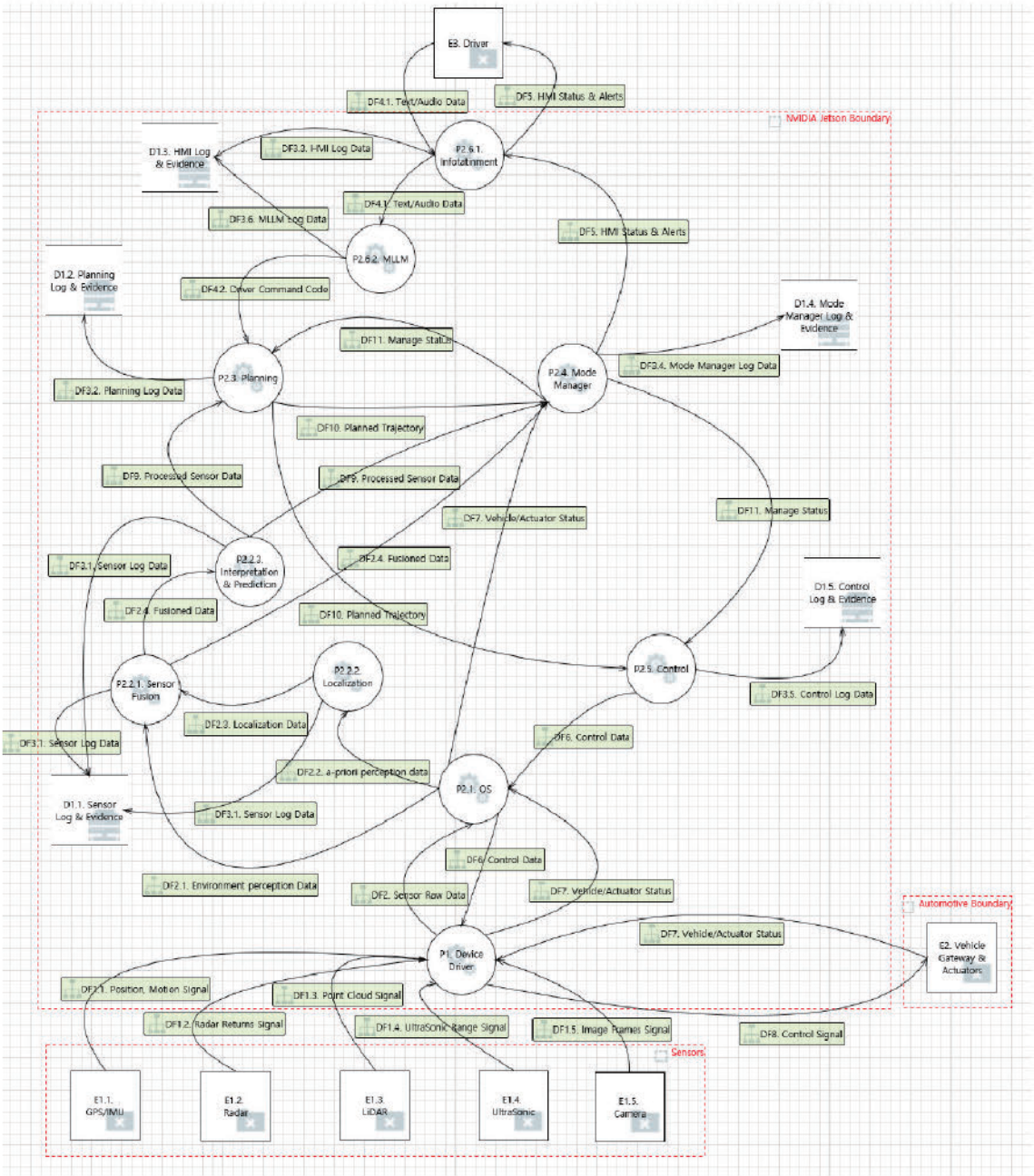


Fig. 6 Level 2 for automated driving systems

3.2.2 정규화 · 추상화

수집된 CVE, CWE, Paper, Technical Report, Standard는 문헌 기반 1차 공격 근거 집합이다. Shostack(2014)은 Threat Modeling에서 이러한 문헌 검토를 “Attack library의 출발점”으로 간주하며, 공격 간의 유사성과 차이를 식별하고 이를 추상화 하는 과정이 Attack library로 발전하기 위한 핵심이라고 언급하였다.[18] 이에 본 연구는 수집된 문헌을 분석하여 중복 · 연계 공격을 정규화하고, 공통 패턴을 기준으로 추상화하여 재사용 가능한 Attack Library를 구축하였다.

먼저, 수집된 개별 문헌(r_i)에서 공격의 본질을 결정짓는 3가지 핵심 속성을 추출하여 속성 벡터 $Attr(r_i)$로 정의하였다. 이는 공격 수행 매체 및 기법을 나타내는 공격 벡터($Vector_i$), 공격 대상이 되는 자산을 의미하는 대상($Target_i$), 공격이 초래하는 시스템의 상태 변화인 결과($Consequence_i$)로 구성된다.

$$Attr(r_i) = < Vector_i, Target_i, Consequence_i > \quad (1)$$

정규화 단계에서는 추출된 속성 벡터 간의 중복을 제거하고 대표 공격 시나리오를 도출하였다. 임의의 두 문헌 r_i와 r_j에 대하여, 공격 대상과 초래되는 결과가 동일하고, 공격 벡터가 기술적으로 유사한 범주에 속할 경우, 이를 동일한 공격 유형으로 간주하여 하나의 정규화된 시나리오 A_k로 추상화하였다.

$$\begin{aligned} A_k = Merge(r_i, r_j) \Leftrightarrow & (Vector_i \approx Vector_j) \\ & \wedge (Target_i \equiv Target_j) \\ & \wedge (Consequence_i \equiv Consequence_j) \end{aligned} \quad (2)$$

구체적인 예로, 본 연구에서 분석한 Robust Physical-World Attacks on Deep Learning Visual Classification[22]와 Physical Adversarial Examples for Object Detectors[23]의 경우, 제목과 실험 환경은 상이하나 공격의 매커니즘을 분석한 결과 두 사례 모두 물리적 패턴 부착이라는 공격 벡터를 사용하며, Computer vision 모델을 대상으로 표지판 오분류라는 동일한 결과를 가진다. 이에 따라 해당 문헌들은 별개의 항목이 아닌 하나의 공격 시나리오로 정규화하였다. 반면 유사하게 오분류를 유발하더라도 Adversarial Sensor Attack on LiDAR-based Perception in Autonomous Driving[16]과 같이 공격 대상이 상이한 경우에는 별도의 공격 시나리오로 식별하였다.

3.3 STRIDE · MITRE ATLAS · OWASP LLM Top 10

수집 및 정규화한 공격 항목을 보다 포괄적이고 체계적으로 분류하기 위하여 STRIDE, MITRE ATLAS, OWASP LLM Top 10의 세 가지 위협 프레임워크를 병렬적으로 적용하였다. STRIDE는 전통적인 보안 속성 위반 유형을 기반으로 위협을 체계적으로 식별할 수 있는 대표적 모델이다. 그러나 전통적인 위협모델링 프레임워크는 일반적인 소프트웨어 구조를 전제로 설계되었다.[2,44,45] 데이터 · 모델 공급망 공격(Supply chain attack), 학습 데이터 오염(Data poisoning), 프롬프트 인젝션(Prompt injection) 등 AI 및 Physical AI 환경의 특수한 위협 요소를 충분히 설명하기에는 어려움이 존재한다.[2,6-8),44,45]

3.3.1 분류 및 맵핑

모든 공격 항목에 대해 STRIDE 모델을 기반으로 위협의 기본 속성을 식별한다. 위조(Spoofing), 변조(Tampering), 부인(Repudiation), 정보유출(Information disclosure), 서비스 거부(Denial of service), 권한상승(Elevation of Privilege)의 여섯 범주로 분류한다. 공격 대상이 Physical AI 환경의 자산에 영향을 미치는 경우, 해당 항목을 MITRE ATLAS Techniques의 세부 전술 · 기법 항목에 따라 추가적으로 맵핑한다. LLM(Large Language Model) 자산에 대해서는 LLM 특유의 위협을 반영하기 위하여 OWASP LLM Top 10의 항목을 분류 체계로 적용하여 추가적인 맵핑을 수행한다. Table 3은 본 연구에서 구축한 Attack Library의 예시를 제시한다. 전체 Attack Library는 공개 저장소를 통해 확인할 수 있다.[49] 동일한 공격 항목이 STRIDE, MITRE ATLAS, OWASP LLM Top 10의 세 가지 위협 프레임워크 상에서 어떻게 상호 연관되어 분류 · 맵핑되는지를 보여준다.

Table 3 Attack library example

Attack	Reference	STRIDE	ATLAS	OWASP
Spoofing vehicle-to-vehicle (V2V) safety messages to trigger false collision warnings.	GPS spoofing attack detection on intersection movement assist using one-class classification[14]	S	N/A	N/A
Recording genuine GNSS signals and replaying them with delay or amplification to deceive the receiver into interpreting forged signals as legitimate.	GNSS spoofing and detection[19]	S	N/A	N/A
Injecting laser points to manipulate LiDAR point clouds, generating non-existent obstacles and misleading DNN-based object detectors.	Adversarial sensor attack on LiDAR-based perception in autonomous driving[16]	S	Physical environment access	N/A
Injecting randomly fabricated LiDAR points to bias SLAM-based localization and mapping.	Random spoofing attack against LiDAR-based scan matching SLAM[20]	S	Physical environment access	N/A

Attack	Reference	STRIDE	ATLAS	OWASP
Injecting continuous or high-intensity ultrasonic signals into the receiver to produce fake echoes (ghost obstacles).	Analyzing and enhancing the security of ultrasonic sensors for autonomous vehicles[21]	S	Physical environment access	N/A
Applying graffiti or stickers to a STOP sign to perform a physical adversarial attack that causes computer vision systems to misclassify the traffic sign.	Robust physical-world attacks on deep learning visual classification[22] Physical adversarial examples for object detectors[23]	T	Physical environment access	N/A
Projecting laser patterns to disrupt camera sensors and distort visual input through specific light interference.	An integrated approach of threat analysis for autonomous vehicles perception system[2]	T	Physical environment access	N/A
Inducing DNN misclassification through subtle perturbations such as FGSM or PGD adversarial examples.	Adversarial attacks and defenses against deep neural networks: A survey[24]	T	Evade AI model	N/A
Injecting backdoors into DNNs by poisoning training data with trigger patterns, causing targeted misclassification.	Adversarial machine learning - A taxonomy of attacks and mitigations[25]	T	Poison training data	N/A
Degrading model performance through training data poisoning (e.g., large-scale label flipping) as an availability attack.	Adversarial machine learning - A taxonomy of attacks and mitigations[25]	T	Poison training data	N/A
Conducting model extraction attacks by leveraging inference APIs to reconstruct model parameters via query-response interactions.	Adversarial machine learning - A taxonomy of attacks and mitigations[25] MITRE ATLAS: Exfiltration via AI inference API[26]	I	Exfiltration via AI inference API	N/A
Performing membership inference to leak training data by determining whether specific samples were included in the model's training set.	Adversarial machine learning - A taxonomy of attacks and mitigations[25]	I	Exfiltration via AI inference API	N/A
Executing indirect prompt injection by embedding hidden instructions in external content to manipulate the behavior of an LLM.	Not what you've signed up for: Compromising Real-World LLM-integrated applications with indirect prompt injection[27]	T	LLM prompt injection	Prompt injection
Persistent prompt worm attack – replicating malicious instructions in conversation memory to cause long-term session contamination.	Not what you've signed up for: Compromising Real-World LLM-integrated applications with indirect prompt injection[27]	T	LLM prompt injection	Prompt injection
Exploiting excessive privilege delegation in LLM agents – deceiving the agent into performing unsafe or unauthorized actions.	OWASP Top 10 for LLMs - LLM06 excessive agency[13] Exploiting programmatic behavior of LLMs: Dual-use through standard security attacks[28]	E	Command and scripting interpreter	Excessive agency
System prompt leakage attack – extracting hidden or proprietary system-level instructions from the model.	OWASP Top 10 for LLMs - LLM07 system prompt leakage[13] MITRE ATLAS: Extract LLM system prompt[26]	I	Extract LLM system prompt	System prompt leakage
Knowledge conflict injection – confusing the LLM by introducing contradictory or misleading external information in a RAG (Retrieval-Augmented Generation) pipeline.	ASTUTE RAG: Overcoming imperfect retrieval augmentation and knowledge conflicts for large language models[29]	T	Erode AI model integrity	Vector and embedding weaknesses
LLM training data backdoor injection – poisoning the instruction-tuning dataset with malicious samples to implant hidden triggers.	OWASP Top 10 for LLMs - LLM04 data and model poisoning[13]poisoning language models during instruction tuning[30]	T	Poison training data	Data and model poisoning
Fine-tuning to remove LLM guardrails – conducting small-scale additional training to override or neutralize RLHF-based safety alignment.	Removing RLHF protections in GPT-4 via fine-tuning[31]	T	Poison training data	Data and model poisoning

Attack	Reference	STRIDE	ATLAS	OWASP
Embedding inversion attack – reconstructing original text from vector embeddings to recover sensitive or private content.	Sentence Embedding Leaks More Information than You Expect: Generative Embedding Inversion Attack to Recover the Whole Sentence[32]Information Leakage in Embedding Models[33]	I	Exfiltration via AI inference API	Vector and embedding weaknesses
CAN bus flooding attack – causing denial of service by monopolizing the bus using high-priority message IDs.	Evaluation of DoS Attacks on Vehicle CAN Bus System[34]	D	N/A	N/A
Forged CAN message injection – remotely controlling vehicle functions via compromised telematics or gateway ECUs by injecting spoofed CAN frames.	CVE-2015-5611[35], CVE-2016-9337[35]	S	N/A	N/A
Unauthorized ECU firmware update attack – injecting unsigned software into ECUs bypassing signature verification mechanisms.	CVE-2018-1170[35], CVE-2018-9322[35]	E	N/A	N/A
Voice synthesis authentication bypass – using AI-based voice cloning to impersonate legitimate users and bypass smart assistant authentication.	Hello me, meet the real me: Voice synthesis attacks on voice assistants[36]	S	Evade AI model	N/A
Inaudible ultrasonic command injection – sending ultrasonic signals to issue hidden commands and manipulate voice assistants.	DolphinAttack: Inaudible Voice Commands[37]	S	Physical environment access	N/A
Laser-based remote audio injection (Light Commands) – modulating laser light to transmit inaudible voice commands to microphone sensors.	Light Commands: Laser-Based Audio Injection attacks on voice-controllable systems[38]	S	Physical environment access	N/A
Phantom attack using digital signage or projectors – projecting fake traffic signs or lane markings to deceive perception systems.	Phantom of the ADAS: Securing Advanced Driver-Assistance Systems from Split-Second Phantom Attacks[15]	S	Physical environment access	N/A
DeepBillboard attack – manipulating billboard textures to bias end-to-end steering models in autonomous driving.	DeepBillboard: systematic physical-world testing of autonomous driving systems[39]	T	Physical environment access	N/A
LiDAR detection backdoor attack – embedding trigger patterns to induce false positives or false negatives in LiDAR-based object detection.	Towards Backdoor Attacks against LiDAR Object Detection in AD[40]	T	Poison training data	N/A

3.3.2 STRIDE 기반 위협 시나리오 도출

STRIDE는 Microsoft(1999)가 제안한 위협 모델링 프레임워크로, DFD의 구성요소별로 위협을 여섯 범주로 분류・식별한다.[11] Table 4는 DFD각 구성요소에 적용 가능한 STRIDE 위협 범주를 나타낸다.[11]

Attack Library에서 STRIDE로 분류된 공격들을 기반으로, Table 4의 규칙에 따라 각 DFD 자산별 적용 가능한 위협을 선별한다. 이를 기반으로 위협 시나리오를 도출한다. 예를 들어, External entity 구성요소는 Spoofing 및 Repudiation 위협에 취약하다. 이에 따라 Attack library에서 Spoofing(S) 또는 Repudiation(R)으로 분류된 공격 중

Table 4 Threats affecting elements [11]

Threat	Data flows	Data stores	Processes	External entities
Spoofing			✓	✓
Tampering	✓	✓	✓	
Repudiation			✓	✓
Information disclosure	✓	✓	✓	
Denial of service	✓	✓	✓	
Elevation of privilege			✓	

Table 5 Threat scenario examples by STRIDE

Asset	Threat scenario	STRIDE
E1.1. GPS/ IMU	Contamination of Position, Velocity, and Time (PVT) data via GNSS Spoofing.	S
	Acceptance of malicious data due to the absence of authentication mechanisms for critical functions.	S
	Ingestion of falsified GNSS measurements resulting from insufficient data authenticity verification.	S
	Sensor data injection facilitated by inadequate endpoint constraints on communication channels.	S
	Failure in post-incident actor identification and non-repudiation due to the lack of logging for GNSS/IMU anomalies.	R
P2.1 OS	Creation of unauthorized processes, services, or nodes stemming from a lack of mutual authentication.	S
	Session hijacking and process impersonation caused by improper key management or cryptographic misconfiguration.	S
	Modification of Kernel/OS memory space via exploitation of Out-of-bounds or Integer Overflow vulnerabilities.	T
	Alteration of system configurations or execution of arbitrary commands via OS/Command Injection vectors.	T
	Modification of configuration, log, or module files through Path Traversal vulnerabilities.	T
	Corruption of kernel data and memory structures by exploiting vulnerabilities in the Linux CAN Broadcast Manager.	T
	Exposure of sensitive data resulting from transmission in plaintext or the use of weak cryptographic algorithms.	I
	Unauthorized elevation or misuse of privileges due to inadequate privilege management controls.	E

해당 자산에 적용 가능한 공격을 활용하여 위협 시나리오를 구성한다. 이러한 절차를 통해 각 자산별로 위협 시나리오를 체계적으로 도출하였으며, Table 5는 STRIDE 맵핑을 통해 도출된 위협 시나리오의 예시이다. 전체 위협 시나리오는 공개 저장소를 통해 확인할 수 있다.[49]

3.3.3 AI 자산 위협 시나리오 도출

인공지능(AI) 관련 공격 항목을 대상으로, MITRE ATLAS 및 OWASP LLM Top 10을 병렬 적용하여 AI 자산에 대한 위협 시나리오를 도출하였다. OWASP LLM Top 10은 LLM 응용에 특화된 위협 프레임워크로, 프롬프트 인젝션, 시스템 프롬프트 유출, 모델 오염 등 LLM의 상호작용・출력・학습 단계에서 발생 가능한 취약점을 중점적으로 다룬다.[12] 한편 MITRE ATLAS는 AI 시스템 전반에 걸친 공격자 전술・기법(Tactics & Techniques)을 체계화한 프레임워크로, 데이터 오염(Poison training data), 모델 추론 정보 탈취(Exfiltration via AI Inference API), 물리 환경 교란(Physical environment access) 등 AI와 관련된 여러 자산 계층의 위협까지 포괄한다. Table 6은 P2.2.3. InterPretation & Prediction 자산에 대해서, Table 7은 P2.3. Planning 자산에 대해서, Table 8은 P2.6.2. MLLM 자산에 대한 위협시나리오 예시이다. 전체 위협 시나리오는 공개 저장소를 통해 확인할 수 있다.[49]

Table 6 P2.2.3.threat scenario examples

Asset	Threat scenario	ATLAS
P2.2.3. InterPretation & Prediction	Distortion of object detection outputs via LiDAR point cloud injection, inducing the perception of non-existent (phantom) objects.	Physical environment access
	Induction of bias in SLAM pose estimation through the insertion of randomized, fabricated LiDAR point data.	Physical environment access
	Generation of spurious echoes (ghost obstacles) via continuous wave signal injection targeting ultrasonic sensors.	Physical environment access
	Masking of legitimate echo signals via high-intensity acoustic pressure on ultrasonic receivers, resulting in obstacle detection failure.	Physical environment access
	Induction of traffic sign recognition errors through physical perturbations such as stickers or graffiti.	Physical environment access
	Disruption of camera sensor pixels via laser projection (blinding), causing object recognition failures.	Physical environment access
	Model misclassification precipitated by the injection of digital adversarial examples (e.g., FGSM, PGD).	Evade AI Model
	Degradation of predictive accuracy in recognition models due to training data label manipulation (Label Flipping).	Poison training data

Table 7 P2.3.threat scenario examples

Asset	Threat scenario	ATLAS
P2.3. Planning	Induction of maladaptive behaviors (e.g., abrupt braking, lane deviation) in path planning algorithms via training data poisoning.	Poison training data
	Replication of the internal planning model and reverse engineering of vulnerabilities via exploitation of the model inference API.	Exfiltration via AI inference API
	Induction of erroneous decision-making in the AI planning system via manipulation of external visual semantics	Physical environment access
	Fabrication of object presence via activation of latent backdoors within the LiDAR detection module.	Physical environment access

Table 8 P2.6.2.threat scenario examples

Asset	Threat scenario	OWASP
P2.6.2. MLLM	Execution of Indirect Prompt Injection via the ingestion of untrusted external content (e.g., web data, RAG corpora).	Prompt injection
	Persistent compromise of the model's context window caused by the residency of Prompt Worms within session memory.	Prompt injection
	Exposure of the System Prompt (internal constraints and instructions), enabling adversaries to reverse-engineer and circumvent safety policies.	System prompt leakage
	Ablation of RLHF-based safety guardrails via malicious Fine-tuning, aimed at eliciting restricted or harmful outputs.	Data and model poisoning
	Autonomous execution of high-risk commands resulting from the exploitation of Excessive Agency and insufficient scope limitation.	Excessive agency
	Reconstruction of sensitive source data or text segments via Embedding Inversion attacks.	Vector and embedding weaknesses
	Induction of erroneous inference or hallucinations triggered by Knowledge Conflict within the Retrieval-Augmented Generation (RAG) pipeline.	Vector and embedding weaknesses

Table 9 Damage scenario example

Asset	Damage scenario	CIA	SFOP
P2.1. OS	Critical compromise of driving control integrity driven by malicious processes manipulating control and sensor data streams.	I/A	S
	Exposure of sensitive data resulting in privacy violations, security breaches, and regulatory non-compliance.	C	P
P2.3. Planning	Imminent risk of collision or pedestrian injury caused by erratic maneuvers (e.g., inappropriate braking, abrupt steering, lane departure).	I/A	S
	Compromise of user privacy facilitating unauthorized surveillance, tracking, and targeted social engineering attacks.	C	P
P2.6.2. MLLM	Induction of malformed vehicle control messages or invalid authorization codes via the generation of hazardous commands by the MLLM.	I	S
	Execution of restricted operations resulting from the circumvention of safety guardrails and misuse of privileges.	I	O

3.3.4 피해 시나리오 도출

피해 시나리오는 차량 또는 차량 기능과 관련되어 도로 이용자에게 영향을 미치는 부정적인 결과이다.[5] 앞서 진행된 결과인 위협시나리오와 사이버보안 속성인 C/I/A 기반으로 자산 별 피해 시나리오를 도출 하였다. 피해 시나리오는 안전(Safe, S), 재무(Financial, F), 운영(Operational, O) 및 개인정보(Privacy, P) 카테고리로 분류할 수 있으며, 조직에 따라 Severe(심각), Major(주요), Moderate (보통), Negligible(경미) 등급으로 평가 할 수 있다.[5] 본 연구의 경우 피해 시나리오에 대한 분류는 진행하나, 조직에 따라 달라질 수 있는 영향 평가는 진행하지 않는다. Table 9는 본 연구에서 각 자산별 도출된 피해 시나리오

와 사이버보안 속성 및 S/F/O/P에 대한 분류 예시를 제시한다. 전체 위협 시나리오는 공개 저장소를 통해 확인할 수 있다.[49]

3.4 Attack Path Analysis

공격 경로는 맵핑된 위협 시나리오를 실현하기 위한 일련의 의도적인 행동들이다.[5] 본 연구에서 앞서 도출된 위협 시나리오와 피해 시나리오를 양 끝단으로 설정하고, 그 사이를 연결하는 구체적인 공격 행위를 Attack library에 근거하여 상세화하는 방식으로 공격 경로를 도출하였다. 구체적으로, 공격 경로의 구성은 자산의 특성, 공격자의 의도, 그리고 공격을 수행하기 위한 기술적 수단을 종합적으로 고려하여 수행되었다. 우선, 각 자산에 맵핑된 위협 시나리오를 실현하기 위해 공격자가 수행해야 하는 선행 조건과 진입 지점을 식별하였다. 이후, 해당 위협 시나리오 도출의 근거가 되었던 Attack library의 구체적인 공격 기법을 경로상의 핵심 실행 단계로 배치함으로써 공격의 기술적 실현 가능성을 확보하였다. 마지막으로 이러한 공격 행위가 시스템에 미치는 기능적 영향을 분석하여 최종적인 피해 시나리오로 연결되는 인과 관계를 완성하였다. Table 10은 이러한 분석 과정을 통해 도출된 위협 시나리오별 공격 경로의 예시를 보여준다. 전체 공격 경로는 공개 저장소를 통해 확인할 수 있다.[49]

3.5 보안 요구사항 도출

ISO/SAE 21434의 TARA 프로세스에서는 위협 시나리오별 위험 처리 결정을 수행한다. 이 단계는 조직별 자체 평가기준에 따라 위험을 어떻게 처리할지 결정하는 단계이다. 해당 단계에서 Reducing the risk가 선택된 경우, 위험 감소를 위한 기술적·조직적 통제가 필요하기 때문에 보안 요구사항을 도출한다. 이 보안 요구사항은 후속 단계에서 정의되는 Cybersecurity goals의 기반 요소이다. 그러나 본 연구에서는 조직별 자체 평가기준에 기반한 위험 처리 결정 단계를 수행하지 않는다. 따라서 모든 위협 시나리오에 대해 일관적으로 보안 요구사항을 도출한다. 보안 요구사항은 각 자산에 대해 도출된 위협 시나리오, 피해 시나리오, 공격 경로, Attack library를 근거하여 도출 한다. 동일한 보호 조치를 필요로 하는 위협 시나리오들은 하나의 보안 요구사항으로 군집화하여 맵핑한다. 이러한 통합 절차는 보안 요구사항 중복을 방지한다. 이후 체크리스트 구성을 위한 기본 산출물을 제공한다. Table 11은 도출된 보안 요구사항의 예시이다. 전체 보안 요구사항은 공개 저장소를 통해 확인할 수 있다.[49]

Table 10 Attack path examples by threat scenario

Asset	Damage scenario	Threat scenario	Attack path
P2.1. OS	Collapse of security boundaries via privilege escalation or isolation bypass, resulting in persistent manipulation, data exfiltration, and the neutralization of safety functions, ultimately causing hazardous driving or collisions.	Inadequate Privilege and Permission Management. (Includes insecure default settings, mismanagement of privileged accounts, and incorrect permission assignments.)	i. Reconnaissance: The attacker analyzes the system's privilege management policies and default configurations. ii. Vulnerability Discovery: Excessive default privileges or incorrect permission assignments are identified. iii. Exploitation (Escalation): The attacker utilizes privilege escalation techniques from a low-integrity process to acquire administrative (root) privileges [AL-29]. iv. Action on Objectives: With elevated privileges, the attacker bypasses all security boundaries, neutralizes safety systems, and establishes remote control over the vehicle.
P2.3. Planning	Risk of collision or pedestrian injury caused by maladaptive maneuvers. (e.g., inappropriate braking, abrupt steering, lane deviation.)	Induction of erroneous decision-making in the AI planning system via manipulation of external visual semantics. (e.g., altering billboards or infrastructure.)	i. The attacker modifies the physical environment with adversarial stickers or paint. ii. These visual changes alter the semantic meaning perceived by the AI [AL-36]. iii. The planner generates a trajectory that follows the manipulated semantics (e.g., driving into a wall).
P2.6.2. MLLM	Induction of malformed vehicle control messages or invalid authorization codes via the generation of hazardous commands by the MLLM.	Ablation of RLHF-based safety guardrails via malicious Fine-tuning, aimed at eliciting restricted or harmful outputs.	i. The attacker gains access to the model's fine-tuning pipeline. ii. A malicious dataset designed to reverse safety alignment is used for fine-tuning [AL-49]. iii. The deployed model unlearns its safety constraints and executes previously restricted commands.

Table 11 Security requirements examples

Asset	No.	Security requirement category	Security requirement	Security requirement description
P2.1. OS	SR-38	Authentication & Access Control	Middleware and network services shall enforce mutualauthentication and bind to specific internal interfaces only	To prevent unauthorized node connections and injection attacks, ROS/DDS middleware must utilize security plugins (e.g., DDS Security) forauthentication, and IPC sockets/services must not bind to wildcard addresses (0.0.0.0).
	SR-40	System Hardening & Platform Security	Sensitive memory regions and execution contexts shall be isolated using hardware-enforced protection domains.	To mitigate side-channel attacks (Spectre), KASLR bypass, and SMM extraction, critical secrets (keys) and privileged code (SMM/TrustZone) must be isolated from user space using memory protection units and side-channel resistant coding practices.
	SR-41	Data Confidentiality & Privacy	Data at rest and in transit shall be protected using strong encryption and authenticity verification.	All stored sensitive files must be encrypted, and network packets must carry MACs to prevent tampering. Session keys must be managed securely to prevent hijacking.
P2.3. Planning	SR-61	Communication Security & Integrity	Middleware communication channels for planning inputs shall be authenticated to prevent unauthorized trajectory injection.	To mitigate the injection of fake trajectories or manipulated status frames (e.g., emergency stop), the planning module must verify the digital signature and source authenticity of all incoming ROS/DDS messages.
	SR-62	AI/Sensor Robustness & Input Validation	Adversarial defense mechanisms and input sanitization shall be implemented to detect semantic perturbations and prompt injections.	The system must employ defenses against physical adversarial attacks (stickers, patches), digital noise in cost maps, and indirect prompt injections (e.g., malicious text on signs) to ensure robust decision-making.
	SR-64	System Hardening & Platform Security	Secure boot and strict input validation shall be implemented to prevent arbitrary code execution and memory corruption.	The system must validate deserialization processes and input parameters to prevent RCE, memory overflows, and privilege escalation. Additionally, Secure Boot must protect the boot chain (CBoot/MB2) from tampering.
P2.6.2. MLLM	SR-86	AI/Sensor Robustness & Input Validation	Multi-modal input sanitization and alignment mechanisms shall be implemented to detect and reject jailbreaks and adversarial injections.	To defend against direct/indirect prompt injection (e.g., "Ignore rules"), visual adversarial patches, and hidden text in images, the system must employ robust input filtering, adversarial training, and safety alignment (e.g., RLHF) to block malicious instructions.
	SR-91	Availability & Resilience	Resource quotas and algorithmic complexity limits shall be enforced to prevent Denial of Service via sponge attacks.	To defend against algorithmic DoS attacks (e.g., sponge examples, token bombs) that exhaust GPU memory or energy, the system must enforce strict limits on input token length, recursion depth, and inference time per request.

3.6 체크리스트 도출

본 절에서는 자산별로 도출된 보안 요구사항에 대해 병합, 정규화 및 중복 제거를 수행하여 기술적 정합성을 확보하고, 이를 기반으로 총 38개의 보안 체크리스트를 최종 도출하였다. Table 12는 도출된 체크리스트이다. 이러한 체크리스트 변환 과정은 추상적인 요구사항을 정량적 평가가 가능한 구체적인 검증 항목으로 구체화하여 구현 단계의 모호성을 제거하기 위함이다. 나아가, 본 체크리스트는 향후 사이버보안 관리 체계(CyberSecurity Management System, CSMS) 감사 및 차량 형식 승인(Vehicle Type Approval, VTA) 시, 실질적인 보안 적용을 입증하는 객관적 증적자료로 활용하기 위함이다.

Table 12 Checklist

No.	Checklist item	Security requirement No.
CL-01	Is data ingested from external assets subject to rigorous validation of integrity and freshness (anti-replay)?(e.g., Implementing HMAC-SHA256 or CMAC for integrity, and verifying Timestamps or Monotonic Counters to prevent replay)	SR-1, SR-3, SR-6, SR-7, SR-61, SR-62, …
CL-02	Is input data accepted as trusted only when originating from registered and authenticated external assets?(e.g., Enforcing mutual authentication via X.509 Certificates (mTLS) or validating Digital Signatures (Ed25519/ECDSA))	SR-38, SR-43, SR-56, SR-61, SR-81, …
CL-03	Is secure channel communication (e.g., end-to-end authentication and encryption) enforced between external assets?	SR-41, SR-53, SR-59, SR-66, SR-71, …
CL-04	Are all events related to external assets logged at a granularity sufficient to ensure non-repudiation and enable post-incident forensic analysis?	SR-5, SR-9, SR-13, SR-17, SR-21, SR-26, …
CL-05	Are unauthorized nodes or devices (those not registered or authenticated) effectively blocked from interaction?(e.g., Using 802.1X NAC for network access control, or applying strict Allow-list based MAC Filtering and Firewall rules)	SR-4, SR-8, SR-12, SR-16, SR-20, SR-23, …
CL-06	Are session tokens and cryptographic keys identifying external assets generated, stored, distributed, and rotated via secure lifecycle management procedures?	SR-24, SR-28, SR-41, SR-140, SR-148, …
CL-07	Are guarantees in place to prevent the compromise or reuse (replay) of session tokens and cryptographic keys identifying external assets?	SR-24, SR-28, SR-41, …
CL-08	Are remote interfaces (IVI, Cellular, Bluetooth, etc.) logically and physically segregated and isolated from the critical control network?	SR-23, SR-33, SR-38, SR-40, SR-47, SR-83, …
CL-09	Are vulnerability mitigation measures (updates, patches, filtering) actively applied to remote interfaces?	SR-23, SR-80, SR-82, SR-194, SR-218, …
CL-10	Are control frames (commands, status messages) and associated metadata subject to integrity/freshness verification and tamper detection upon transmission and reception?	SR-22, SR-68, SR-73, SR-117, SR-199, …
CL-11	Are design and verification mechanisms in place to guarantee memory safety (preventing Buffer Overflow, Integer Overflow, UAF, TOCTOU, etc.)?(e.g., Adopting memory-safe languages like Rust, or applying ASLR, DEP/NX, and Stack Canaries in C/C++ environments)	SR-31, SR-34, SR-36, SR-40, SR-45, SR-47, …
CL-12	Is an appropriate privilege management mechanism enforced to prevent privilege escalation by unauthorized processes?	SR-40, SR-47, SR-64, SR-75, SR-87, …
CL-13	Are relevant security events logged at a level sufficient to ensure non-repudiation and forensic analysis?	SR-35, SR-42, SR-48, SR-54, SR-60, SR-67, …
CL-14	Are guarantees in place to prevent the compromise or reuse of session tokens and cryptographic keys identifying internal assets and processes?	SR-24, SR-38, SR-40, SR-41, SR-81, …
CL-15	Does the system defend against Denial of Service (DoS) caused by memory leaks, deadlocks, UAF, or bus flooding, and provide recovery mechanisms in case of anomalies?	SR-2, SR-34, SR-39, SR-52, SR-58, SR-65, …
CL-16	Are internal system metadata and state information protected from exposure via plaintext transmission, weak encryption, debug outputs, or side-channels?	SR-25, SR-34, SR-40, SR-53, SR-71, SR-83, …
CL-17	Are debug and diagnostic functions activated only under strictly authenticated and authorized conditions?	SR-4, SR-8, SR-25, SR-33, SR-83, SR-97, SR-102, SR-107, …
CL-18	Does the Secure Boot chain enforce signature verification for platform and OS images to prevent the execution of unsigned/tampered code and boot-time privilege escalation?	SR-33, SR-37, SR-46, SR-50, SR-57, SR-63, …

No.	Checklist item	Security requirement No.
CL-19	Are defense functions in place to detect and mitigate spoofing or adversarial patterns at the sensor level (LiDAR, Radar, Camera, Ultrasonic, GNSS)?	SR-1, SR-6, SR-10, SR-14, SR-18, SR-27, …
CL-20	Is signature or integrity verification performed on configuration, map, and sensor parameter files, ensuring that unauthorized modifications are blocked?	SR-33, SR-46, SR-50, SR-57, SR-63, SR-69, SR-74, SR-82, SR-89, SR-220
CL-21	Are protection features applied during input processing to prevent output distortion caused by Command Injection?	SR-31, SR-45, SR-51, SR-62, SR-64, SR-69, …
CL-22	Does the system provide defense mechanisms to maintain computation cycles and minimize critical data drops even under input overload conditions (e.g., random points, massive phantom objects)?	SR-34, SR-39, SR-44, SR-52, SR-55, SR-58, …
CL-23	Do processes adhere to the Principle of Least Privilege and isolation to defend against unauthorized privilege escalation or isolation bypass?	SR-29, SR-40, SR-47, …
CL-24	Are Command Codes (or other AI inputs) generated by MLLMs verified for authenticity and checked for Prompt Injection and Context Contamination?	SR-62, SR-80, SR-86, …
CL-25	Does the system possess mechanisms to detect and mitigate physical and digital adversarial examples (stickers, light sources, lasers, phantom objects, etc.)?	SR-18, SR-55, SR-62, SR-86, SR-126, …
CL-26	Does the system possess mechanisms to detect and mitigate Transfer Attacks?	SR-44, SR-90, SR-159, SR-200
CL-27	Does the system possess mechanisms to detect and mitigate Poisoned Dataset injection?	SR-50, SR-55, SR-57, SR-62, SR-63, SR-88, SR-89
CL-28	Are inputs to the MLLM monitored to detect, verify, and block Prompt Injection, Context Contamination, and Adversarial Inputs?	SR-55, SR-62, SR-80, SR-86, SR-88, SR-194
CL-29	Does the MLLM accept input (voice, text) only from legitimate, authenticated users and processes?	SR-27, SR-79, SR-81, SR-198, SR-222
CL-30	Are Command Codes and input data subject to rigorous verification against tampering and replay attacks?	SR-22, SR-27, SR-49, SR-73, SR-79, …
CL-31	Are integrity verification and anti-deserialization safeguards enforced on models, policies, tokenizers, plugin files, environment variables, and temporary files?	SR-45, SR-46, SR-50, SR-51, SR-57, SR-63, …
CL-32	Are sensitive data elements (user voice, text, intent, command codes, key slots) protected via encryption and strict access control mechanisms?	SR-24, SR-29, SR-38, SR-53, SR-71, SR-75, …
CL-33	Does the system maintain operational stability without computational or memory exhaustion under conditions of excessive token input, long-context ingestion, or alert flooding?	SR-52, SR-58, SR-65, SR-76, SR-84, SR-91, …
CL-34	Does the system transition to a defined "Safe Mode" (fail-safe state) in the event of computational or memory resource exhaustion?	SR-2, SR-70, SR-76, SR-95, SR-100, …
CL-35	Is access to log repositories and collectors restricted exclusively to authenticated and authorized subjects?(e.g., Implementing RBAC (Role-Based Access Control) policies and ensuring OS-level isolation via SELinux or AppArmor)	SR-25, SR-94, SR-98, SR-103, …
CL-36	Are attempts at unauthorized access, modification, or deletion of log repositories blocked, with such attempts being explicitly audited (logged)?	SR-93, SR-94, SR-98, SR-99, SR-103, SR-104, …
CL-37	Is the system hardened against insecure deserialization and arbitrary code execution vulnerabilities?(e.g., Avoiding unsafe serialization formats like Pickle, and enforcing strict schema validation for JSON/Protobuf inputs)	SR-31, SR-33, SR-36, SR-37, SR-40, SR-45, …
CL-38	Is the logging infrastructure resilient against manipulation or service interruption caused by the injection of malicious event records (e.g., Log Injection or Log Smashing)?	SR-80, SR-95, SR-100, SR-104, SR-105, SR-109, …

4. 검 증

본 장에서는 3장에서 수행한 확장된 위협 분석 절차를 통해 도출된 보안 체크리스트의 규제 준수 적합성 및 실효성을 검증한다. 본 연구에서 적용한 분석 방법론은 ISO/SAE 21434 표준 모델을 기반으로 하되, 센서 데이터의 물리적 변조와 AI 모델의 적대적 공격을 포괄적으로 식별할 수 있도록 분석 범위를 확장하여 적용한 것이다. 이를 통해 도출된 총 38개의 체크리스트는 Physical AI 시스템의 고유한 보안 요구사항을 반영하고 있다.

그러나 도출된 체크리스트가 학술적 제안을 넘어 실제 산업 현장에서 활용되기 위해서는, 현재 차량 사이버 보안의 국제적 기준이자 강제 법규인 UN R 155와의 정합성을 확보해야 한다. 따라서 본 연구에서는 도출된 체크리스트를 UN R 155의 기술적 요구사항과 비교 분석하여, 해당 체크리스트가 글로벌 법규 준수를 위한 실질적인 가이드라인으로 기능할 수 있는지 검증한다.

4.1 검증 기준 : UN R 155 Annex 5

검증을 위한 비교 기준으로는 UN R 155의 Annex 5를 선정하였다. UN R 155는 제조사가 위험 평가 및 처리를 수행할 때 Annex 5에 나열된 위협과 완화 조치를 필수적으로 고려하도록 명시하고 있다. Annex 5는 크게 위협 및 취약점 목록인 Part A, 차량 내부 완화 조치인 Part B, 차량 외부(백엔드 등) 완화 조치인 Part C로 구성된다. 따라서 본 연구에서 도출할 체크리스트가 Annex 5 Part A에 명시된 다양한 위협 시나리오를 효과적으로 방어할 수 있다면, 이는 체크리스트가 국제 법규 수준의 보안 요구사항을 충족함을 의미한다.

4.2 체크리스트 맵핑 및 정합성 분석

검증은 본 연구의 보안 체크리스트 항목이 UN R 155 Annex 5 Part A에서 정의한 세부(Sub-level) 위협을 커버할 수 있는지 확인하는 방식으로 수행되었다. Annex 5 Part A는 총 7개의 카테고리와 이에 속한 30여 개의 세부 위협 및 취약점(Sub-level description)을 정의하고 있다.

Table 13은 도출된 체크리스트 항목과 UN R 155의 위협 항목을 맵핑한 결과의 일부를 나타낸다. 분석 결과, 본 연구에서 도출한 38개의 체크리스트는 UN R 155 Annex 5 Part A에 제시된 AI 및 센서 관련 위협은 물론, 일반적인 임베디드 보안 위협까지 포괄적으로 대응하고 있음을 확인하였다. 이는 해당 체크리스트가 실제 차량 형식 승인(VTA) 심사 및 CSMS 감사 과정에서 법규 준수 여부를 입증하는 객관적인 증적 자료로 활용 가능함을 시사한다.

Table 13 UN R.155 Annex 5 Part A mapping

UN R 155 Annex 5 Part A	Checklist No.
1. Back-end servers used as a means to attack a vehicle or extract data	CL-01, CL-02, CL-03, CL-05
2. Services from back-end server being disrupted, affecting the operation of a vehicle	CL-08, CL-15, CL-22, CL-33, CL-34
3. Vehicle related data held on back-end servers being lost or compromised (“data breach”)	CL-03, CL-06, CL-07, CL-14, CL-32, CL-35, CL-36
4. Spoofing of messages or data received by the vehicle	CL-01, CL-02, CL-05, CL-10, CL-19, CL-24, CL-25, CL-26, CL-28, CL-29
5. Communication channels used to conduct unauthorized manipulation, deletion or other amendments to vehicle held code/data	CL-01, CL-02, CL-03, CL-09, CL-10, CL-20, CL-21, CL-24, CL-26, CL-27, CL-30, CL-31, CL-37, CL-38
6. Communication channels permit untrusted/unreliable messages to be accepted or are vulnerable to session hijacking/replay attacks	CL-01, CL-02, CL-03, CL-05, CL-07, CL-10, CL-14, CL-21, CL-28, CL-29, CL-30
7. Information can be readily disclosed. For example, through eavesdropping on communications or through allowing unauthorized access to sensitive files or folders	CL-03, CL-04, CL-07, CL-13, CL-14, CL-16, CL-32, CL-35, CL-36
8. Denial of service attacks via communication channels to disrupt vehicle functions	CL-08, CL-15, CL-22, CL-33, CL-34
9. An unprivileged user is able to gain privileged access to vehicle systems	CL-12, CL-17, CL-23
10. Viruses embedded in communication media are able to infect vehicle systems	CL-09, CL-11, CL-31, CL-37
11. Messages received by the vehicle (for example X2V or diagnostic messages), or transmitted within it, contain malicious content	CL-01, CL-10, CL-19, CL-21, CL-24, CL-26, CL-28, CL-30
12. Misuse or compromise of update procedures	CL-03, CL-06, CL-07, CL-09, CL-14, CL-18, CL-27
13. It is possible to deny legitimate updates	CL-08, CL-09, CL-15

UN R 155 Annex 5 Part A	Checklist No.
15. Legitimate actors are able to take actions that would unwittingly facilitate a cyber-attack	CL-05, CL-08, CL-17, CL-23
16. Manipulation of the connectivity of vehicle functions enables a cyber-attack, this can include telematics; systems that permit remote operations; and systems using short range wireless communications	CL-02, CL-03, CL-05, CL-08, CL-09
17. Hosted 3rd party software, e.g. entertainment applications, used as a means to attack vehicle systems	CL-08, CL-09, CL-12, CL-18, CL-23, CL-28, CL-31, CL-37
18. Devices connected to external interfaces e.g. USB ports, OBD port, used as a means to attack vehicle systems	CL-05, CL-08, CL-09, CL-12, CL-17, CL-23
19. Extraction of vehicle data/code	CL-06, CL-07, CL-14, CL-16, CL-31, CL-32, CL-37
20. Manipulation of vehicle data/code	CL-04, CL-13, CL-17, CL-20, CL-21, CL-31, CL-35, CL-36, CL-38
21. Erasure of data/code	CL-04, CL-13, CL-35, CL-36, CL-38
22. Introduction of malware	CL-09, CL-11, CL-18, CL-31, CL-37
23. Introduction of new software or overwrite existing software	CL-11, CL-18, CL-20, CL-31, CL-37
24. Disruption of systems or operations	CL-08, CL-09, CL-15, CL-19, CL-22, CL-25, CL-33, CL-34, CL-38
25. Manipulation of vehicle parameters	CL-20, CL-21
26. Cryptographic technologies can be compromised or are insufficiently applied	CL-06, CL-07, CL-14, CL-16, CL-18, CL-32
27. Parts or supplies could be compromised to permit vehicles to be attacked	CL-18, CL-27, CL-31
28. Software or hardware development permits vulnerabilities	CL-06, CL-11, CL-12, CL-15, CL-16, CL-17, CL-23, CL-37
29. Network design introduces vulnerabilities	CL-08, CL-15, CL-22, CL-23, CL-33
31. Unintended transfer of data can occur	CL-05, CL-17, CL-25
32. Physical manipulation of systems can enable an attack	CL-25, CL-26, CL-27, CL-28

4.3 기존 TARA 기법과의 비교 및 효과 분석

본 연구에서 제안한 Physical AI 특성을 반영한 확장된 위협 분석 절차의 개선 효과를 분석하기 위해 기존의 STRIDE 기반의 위협 분석 결과와 본 연구의 분석 결과를 비교하였다.

우선, 가장 주된 차이점은 위협 식별의 구체성 및 분석 범위에 있다. 기존 STRIDE 모델은 데이터 흐름에 따른 소프트웨어 및 네트워크 취약점을 식별하는 데에는 효과적이나, AI 모델의 학습 및 추론 과정이나 물리적 센서 데이터의 특성과 관련된 위협을 구체화하는 데에는 구조적 한계가 존재한다. 예를 들어, STRIDE의 '변조(Tampering)' 항목은 통신 패킷의 위·변조는 포괄할 수 있으나, '오염된 데이터셋으로 학습된 모델(Poisoned model)'이나 '인간의 인지 영역을 벗어난 노이즈가 주입된 적대적 예제(Adversarial example)'와 같은 AI 특화 공격을 명확히 구분하여 식별하지 못한다. 반면, 본 연구의 확장된 분석 절차는 MITRE ATLAS와 OWASP LLM Top 10을 통합함으로써, 기존 TARA에서는 '기타 오작동'이나 '일반적인 무결성 위협'으로 간주되던 항목들을 기만 공격(Evasion), 모델 역공학(Model inversion), 프롬프트 주입(Prompt injection) 등 Physical AI 고유의 구체적인 위협 시나리오로 세분화하여 도출하였다.

이와 더불어, 정량적인 위협 식별 결과에서도 유의미한 개선 효과가 확인되었다. 동일한 ADS 아키텍처를 대상으로 분석을 수행한 결과, 제안된 방법론은 기존 기법 대비 데이터 무결성 및 AI 모델 신뢰성 측면에서 한층 확장된 체크리스트를 도출하였다. 분석 결과, 전통적인 사이버보안 위협(네트워크 도청, 비인가 접근, 서비스 거부 등)은 동일하게 식별하였으나, 기존 기법에서는 일반적인 무결성 위협 등으로 모호하게 분류되었던 AI 및 센서 데이터 관련 위협(CL-19, CL-20, CL-22, CL-25, CL-26, CL-27)과 LLM 특화 위협(CL-24, CL-28, CL-29, CL-31, CL-32, CL-33)이 본 연구의 분석 절차를 통한 개별 시나리오로 도출되었다. 이는 전체 도출된 38개의 체크리스트 중 약 30 %에 해당하며, Physical AI 환경에서 필수적으로 요구되는 보안 커버리지가 실질적으로 확장되었음을 시사한다.

5. 결 론

ADS는 다중 센서, 고성능 엣지 컴퓨팅, AI 알고리즘, V2X를 통합한 CPS다. 기존 위협모델링은 소프트웨어 중심으로 발전했다. AI 알고리즘이 실시간 판단을 수행하고 물리적 환경과 지속적으로 상호작용하는 CPS 구조에 그대로 적용하기에는 어려움이 존재한다.

본 연구는 이러한 어려움을 극복하고 Physical AI 시스템의 보안 안전성을 확보하기 위해, 기존 ISO/SAE 21434 표준을 기반으로 Physical AI 특성을 반영한 확장된 위협 분석 절차를 제안하였다. 본 연구에서 확장한 위협 분석 절차는 STRIDE · MITRE ATLAS · OWASP LLM Top 10을 병렬 적용하여 공격 시나리오를 다차원적으로 식별하고, 자산 단위의 위협 및 피해 시나리오를 연계 분석할 수 있도록 구성하였다. 위협 및 피해시나리오와 공격 경로 분석 결과를 근거로 보안요구사항과 체크리스트를 도출 하였다.

본 연구에서 제안한 확장된 위협 분석 절차의 실효성은 도출된 보안 체크리스트를 글로벌 차량 사이버보안 법규인 UN R 155의 Annex 5 Part A 위협 항목과 비교 분석하여 검증하였다. 결과적으로 본 연구는 기존의 차량 사이버보안 위협에 더해 Physical AI 고유의 위협까지 포괄하는 확장된 보안 요구사항을 정립함으로써, 제조사가 직면한 차량 형식 승인(VTA) 심사 및 CSMS 인증 절차에 대해 체계적이고 효율적인 대응 방안을 제시 하였다. 아울러, 본 연구 결과의 신뢰성을 높이고 학술적 재현성을 보장하기 위해 연구 과정에서 도출된 전체 데이터셋은 공개 저장소를 통해 공유하였다.[49)]

다만, 본 연구는 Physical AI 환경의 다양한 위협을 누락 없이 포괄적으로 식별하는 데 중점을 두었기에, 조직별로 상이한 위험 수용 기준을 고려하여 Attack Feasibility 및 Risk value 단계는 제외하였다. 이는 실제 산업 현장에 적용 시 한정된 보안 자원을 효율적으로 배분하기 위한 위협 간 우선순위 설정에 추가적인 판단이 필요함을 의미한다. 본 연구에서 제안한 확장된 위협 분석 절차가 차량 내 Physical AI 시스템의 보안 위협 식별 및 인증 대응 활동에 실질적인 도움이 되기를 기대한다. 향후 연구에서는 본 분석 방법론의 적용 대상을 자율주행 차량 플랫폼에서 로봇 및 지능형 모빌리티 등 다양한 Physical AI 시스템 분야로 확대하여 그 범용성과 실효성을 지속적으로 검증할 계획이다. 이는 본 연구가 MITRE ATLAS, OWASP LLM Top 10 등을 통합하여 Physical AI 시스템의 핵심인 AI 모델 및 데이터 특화 위협을 포괄적으로 분석하도록 설계되었기에, 유사한 AI 기반 제어 구조를 가진 타 시스템에도 범용적으로 적용 가능하기 때문이다. 단, 타 시스템에 이를 효과적으로 적용하기 위해서는 대상 시스템의 아키텍처와 물리적 특성을 반영하여 데이터 흐름도(DFD) 기반의 자산을 명확히 재정의 하고, 해당 자산 유형에 최적화된 Attack library를 구축하는 과정이 필요하다.

Acknowledgment

본 연구는 산업통상자원부 및 한국산업기술기획평가원의 지원으로 수행된 “국제표준기반시험장비기술개발 및 고도화지원사업”(연구과제번호: RS-2025-13732975)의 연구 결과물이다.

References

1) N. Shamsi, K. Chandrasekar, Y. Long, C. Limbach, K. Rebello and K. Fu, “WIP: Threat Modeling Laser-Induced Acoustic Interference in Computer Vision-Assisted Vehicles,” Symposium on Vehicle Security and Privacy (VehicleSec), 2024.
2) S. Ghosh, A. Zaboli, H. Junho and J. Kwon, “An Integrated Approach of Threat Analysis for Autonomous Vehicles Perception System,” IEEE Access, Vol.11, 2023.
3) Z. Abuabed, A. Alsadeh and A. Taweel, “STRIDE Threat Model-Based Framework for Assessing the Vulnerabilities of Modern Vehicles,” Computers & Security, Vol.133, Paper No.103391, 2023.
4) 4. P. Zambare, V. N. Thanikella and Y. Liu, “Securing Agentic AI: Threat Modeling and Risk Analysis for a Network Monitoring Agentic AI System,” arXiv Preprint arXiv:2508.10043, 2025.
5) ISO/SAE International, “Road Vehicles—Cybersecurity Engineering,” ISO/SAE 21434:2021, 2021.
6) F. V. Jedrzejewski, D. Fucci and O. Adamov, “ThreMoLIA: Threat Modeling of Large Language Model-Integrated Applications,” arXiv Preprint arXiv:2504.18369, 2025.
7) F. V. Jedrzejewski, “Threat Modeling of ML-Intensive Systems: Research Proposal,” CAIN 2024: Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering – Software Engineering for AI , pp.264-266, 2024.
8) S. Dellersnyder, “Mind the Gap: STRIDE-AI — Your Clear Path to Understanding AI Vulnerabilities,” Toreon, https://www.toreon.com/stride-ai-your-clear-path-to-understanding-ai-vulnerabilities/, 2025.
9) ISO International, “Road Vehicles—Safety for Automated Driving Systems—Design, Verification and Validation,” ISO/TS 5083:2025, 2025.

10) G. Lindemulder and M. Kosinski, “What Is a Data Flow Diagram (DFD)?,” IBM, https://www.ibm.com/think/topics/data-flow-diagram, 2025.
11) S. Hernan, S. Lambert, T. Ostwald and A. Shostack, “Uncover Security Design Flaws Using the STRIDE Approach,” MSDN Magazine, https://learn.microsoft.com/en-us/archive/msdn-magazine/2006/november/uncover-security-design-flaws-using-the-stride-approach, 2006.
12) OWASP Gen AI Security Project, “OWASP Gen AI Security Project Introduction and Background,” OWASP, https://genai.owasp.org/introduction-genai-security-project, 2023.
13) OWASP Gen AI Security Project, “OWASP Top 10 for LLM Applications 2025,” OWASP, https://genai.owasp.org/llm-top-10/, 2025.
14) J. Ying, Y. Feng, Q. A. Chen and Z. M. Mao, “GPS Spoofing Attack Detection on Intersection Movement Assist Using One-Class Classification,” Symposium on Vehicle Security and Privacy (VehicleSec), 2023.
15) B. Nassi, Y. Mirsky, D. Nassi, R. Ben-Netanel, O. Drokin and Y. Elovici, “Phantom of the ADAS: Securing Advanced Driver-Assistance Systems from Split-Second Phantom Attacks,” CCS '20: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, pp.293-308, 2020.
16) Y. Cao, C. Xiao, B. Cyr, Y. Zhou, W. Park, S. Rampazzi, Q. A. Chen, K. Fu and Z. M. Mao, “Adversarial Sensor Attack on LiDAR-Based Perception in Autonomous Driving,” CCS '19: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp.2267-2281, 2019.
17) S. Lee and S. Cho, “Attack Scenarios Based on UNECE Automotive Cybersecurity Regulation (UN R.155),” Transactions of KSAE, Vol.29, No.8, pp.717-732, 2021.
18) A. Shostack, Threat Modeling: Designing for Security, Wiley, 2014.
19) M. L. Psiaki and T. E. Humphreys, “GNSS Spoofing and Detection,” Proceedings of the IEEE, Vol.104, No.6, pp.1258-1270, 2016.
20) M. Fukunaga and T. Sugawara, “Random Spoofing Attack against Scan Matching Algorithm SLAM (Long),” Symposium on Vehicle Security and Privacy (VehicleSec), 2024.
21) W. Xu, C. Yan, W. Jia, X. Ji and J. Liu, “Analyzing and Enhancing the Security of Ultrasonic Sensors for Autonomous Vehicles,” IEEE Internet of Things Journal, Vol.5, No.6, pp.5015-5029, 2018.
22) K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno and D. Song, “Robust Physical-World Attacks on Deep Learning Visual Classification,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp.1625-1634, 2018.
23) K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, F. Tramèr, A. Prakash, T. Kohno and D. Song, “Physical Adversarial Examples for Object Detectors,” WOOT '18: Proceedings of the 12th USENIX Conference on Offensive Technologies, p.1, 2018.
24) M. Ozdag, “Adversarial Attacks and Defenses Against Deep Neural Networks: A Survey,” Procedia Computer Science, Vol.140, pp.152-161, 2018.
25) A. Vassilev, A. Oprea, A. Fordyce, H. Anderson, X. Davies and M. Hamin, “Adversarial Machine Learning: A Taxonomy and Terminology of Attacks and Mitigations (NIST AI 100-2e2025),” National Institute of Standards and Technology (NIST), 2025.
26) MITRE Corporation, “MITRE ATLAS (Adversarial Threat Landscape for Artificial-Intelligence Systems),” MITRE Corporation, https://atlas.mitre.org/, 2021.
27) K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz and M. Fritz, “Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection,” AISec '23: Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security , pp.79-90, 2023.
28) D. Kang, X. Li, I. Stoica, C. Guestrin, M. Zaharia and T. Hashimoto, “Exploiting Programmatic Behavior of LLMs: Dual-Use Through Standard Security Attacks,” Proceedings of the 2024 IEEE Security and Privacy Workshops (SPW), pp.132-143, 2024.
29) F. Wang, X. Wan, R. Sun, J. Chen and S. O. Arik, “Astute RAG: Overcoming Imperfect Retrieval Augmentation and Knowledge Conflicts for Large Language Models,” Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics, Vol.1, pp.30553-30571, 2025.
30) A. Wan, E. Wallace, S. Shen and D. Klein, “Poisoning Language Models During Instruction Tuning,” ICML 2023: Proceedings of the 40th International Conference on Machine Learning, Vol.202, pp.35413-35425, 2023.
31) Q. Zhan, R. Fang, R. Bindu, A. Gupta, T. Hashimoto

and D. Kang, “Removing RLHF Protections in GPT-4 via Fine-Tuning,” Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol.2, pp.681-687, 2024.

32) H. Li, M. Xu and Y. Song, “Sentence Embedding Leaks More Information than You Expect: Generative Embedding Inversion Attack to Recover the Whole Sentence,” Findings of the Association for Computational Linguistics: ACL 2023, pp.14022-14040, 2023.
33) C. Song and A. Raghunathan, “Information Leakage in Embedding Models,” CCS ’20: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security , pp.377-390, 2020.
34) S.-H. Chen and C.-H. R. Lin, “Evaluation of DoS Attacks on Vehicle CAN Bus System,” IIH-MSP 2018: Proceedings of the International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Vol.110, pp.308-314, 2018.
35) MITRE Corporation, “CVE,” MITRE Corporation, https://cve.mitre.org/cgi-bin/cvename.cgi, 2025.
36) D. Bilika, N. Michopoulou, E. Alepis and C. Patsakis, “Hello Me, Meet the Real Me: Voice Synthesis Attacks on Voice Assistants,” Computers & Security, Vol.137, Paper No.103617, 2024.
37) G. Zhang, C. Yan, X. Ji, T. Zhang, T. Zhang and W. Xu, “DolphinAttack: Inaudible Voice Commands,” CCS ’17: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security , pp.103-117, 2017.
38) T. Sugawara, B. Cyr, S. Rampazzi, D. Genkin and K. Fu, “Light Commands: Laser-Based Audio Injection Attacks on Voice-Controllable Systems,” SEC ’20: Proceedings of the 29th USENIX Security Symposium, pp.2631-2648, 2020.
39) H. Zhou, W. Li, Z. Kong, J. Guo, Y. Zhang, B. Yu, L. Zhang and C. Liu, “DeepBillboard: Systematic Physical-World Testing of Autonomous Driving Systems,” ICSE ’20: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering, pp.347-358, 2020.
40) Y. Zhang, Y. Zhu, Z. Liu, C. Miao, F. Hajiaghajani, L. Su and C. Qiao, “Towards Backdoor Attacks against LiDAR Object Detection in Autonomous Driving,” SenSys '22: Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems, pp.533-547, 2023.
41) A. Chattopadhyay and K.-Y. Lam, “Security of Autonomous Vehicle as a Cyber-Physical System,” Proceedings of the 7th International Symposium on Embedded Computing and System Design (ISED), pp.1-6, 2017.
42) S. K. Dhatrika, D. R. Reddy and N. K. Reddy, “Real-Time Object Recognition for Advanced Driver-Assistance Systems (ADAS) Using Deep Learning on Edge Devices,” Procedia Computer Science, Vol.252, pp.25-42, 2025.
43) B. Bi, “ADAS: Advanced Driving Assistance System Based on Jetson Nano,” GitHub Repository, https://github.com/Bill-Bi/ADAS, 2025.
44) A. Chattopadhyay, K.-Y. Lam and Y. Tavva, “Autonomous Vehicle: Security by Design,” IEEE Transactions on Intelligent Transportation Systems, Vol.22, No.11, pp.7015-7029, 2021.
45) S. M. Khalil, H. Bahsi, H. O. Dola, T. Korõtko, K. McLaughlin and V. Kotkas, “Threat Modeling of Cyber-Physical Systems: A Case Study of a Microgrid System,” Computers & Security, Vol.124, Paper No.102950, 2023.
46) NVIDIA Corporation, “Jetson Platform Services Documentation,” NVIDIA Corporation, https://docs.nvidia.com/jetson/jps/moj-overview.html#software-stack, 2025.
47) A. A. Ali, S. Krayem, B. Chramcov and M. F. Kadi, “Self-Stabilizing Fault Tolerance Distributed Cyber Physical Systems,” Proceedings of the 29th DAAAM International Symposium, pp.1173-1180, 2018.
48) UNECE WP.29 (World Forum for Harmonization of Vehicle Regulations), “UN Regulation No.155 – Cyber Security and Cyber Security Management System,” 2021.
49) J. Woo, “ADS-Threat-Modeling,” GitHub Repository, https://github.com/ddiyoung/ADS-Threat-Modeling, 2026.
50) J. Woo, K. Park and J. Lee, “Enhancing the TARA Framework for Physical AI Security: Application to ADS,” KSAE 2025 Annual Autumn Conference & Exhibition, Busan, 2025.