Automating Function-Level TARA for Automotive Full-Lifecycle Security

Yuqiao Yang#
UESTC
yyq_0xdq@163.com

Yongzhao Zhang#
UESTC
zhangyongzhao@uestc.edu.cn

Wenhao Liu
GoGoByte Technology
lwh.scu@gmail.com

Jun Li
GoGoByte Technology
lijun_research@gogobyte.com

Pengtao Shi
GoGoByte Technology
shipengtao@gogobyte.com

DingYu Zhong
UESTC
18166789795@163.com

Jie Yang*
UESTC
jie.yang@uestc.edu.cn

Ting Chen*
UESTC
chenting19870201@163.com

Sheng Cao
UESTC
caosheng@uestc.edu.cn

Yuntao Ren
Chengdu Anheng Information
Technology Co., LTD
atao_uestc@163.com

Yongyue Wu
Anheng Vision(Chengdu) Information
Technology Co., LTD
wuyongyue@isecvision.com

Xiaosong Zhang
UESTC
johnsonzxs@uestc.edu.cn

Abstract—As modern vehicles evolve into intelligent and connected systems, their growing complexity introduces significant cybersecurity risks. Threat Analysis and Risk Assessment (TARA) has therefore become essential for managing these risks under mandatory regulations. However, existing TARA automation methods rely on static threat libraries, limiting their utility in the detailed, function-level analyses demanded by industry. This paper introduces DefenseWeaver, the first system that automates function-level TARA using component-specific details and large language models (LLMs). DefenseWeaver dynamically generates attack trees and risk evaluations from system configurations described in an extended OpenXSAM++ format, then employs a multi-agent framework to coordinate specialized LLM roles for more robust analysis. To further adapt to evolving threats and diverse standards, DefenseWeaver incorporates Low-Rank Adaptation (LoRA) fine-tuning and Retrieval-Augmented Generation (RAG) with expert-curated TARA reports. We validated DefenseWeaver through deployment in four automotive security projects, where it identified 11 critical attack paths, verified through penetration testing, and subsequently reported and remediated by the relevant automakers and suppliers. Additionally, DefenseWeaver demonstrated cross-domain adaptability, successfully applying to unmanned aerial vehicles (UAVs) and marine navigation systems. In comparison to human experts, DefenseWeaver outperformed manual attack tree generation across six assessment scenarios. Integrated into commercial cybersecurity platforms such as UAES and Xiaomi, DefenseWeaver has generated over 8,200 attack trees. These results highlight its ability to significantly reduce processing time, and its scalability and transformative impact on cybersecurity across industries.

I. INTRODUCTION

The automotive industry is rapidly advancing toward intelligent, networked vehicles, integrating technologies like autonomous driving [22], Over-The-Air updates [17], and Advanced Driver Assistance Systems [14]. While these innovations enhance functionality and user experience, they also increase the number of Electronic Control Units (ECUs) and the complexity of topologies and interconnectivity within the In-Vehicle Network (IVN), significantly expanding the potential attack surface of modern vehicles. By 2030, an estimated 95% of new vehicles will be connected, creating a vast cyber threat landscape [1]. Another growing concern is the risk of supply chain safety, where vulnerabilities in third-party components can compromise overall security. As component interconnectivity increases, so too does the number of potential attack vectors, highlighting the need to assess both the entire vehicle and its individual components.

In response to the growing attack surface, TARA has become a cornerstone of automotive cybersecurity, systematically identifying, analyzing, and prioritizing security risks. At its core, TARA involves generating attack trees and assessing risk levels, which together provide a structured approach for understanding and mitigating potential threats throughout the vehicle lifecycle. TARA is also a mandatory regulatory requirement for automotive OEMs and suppliers, in compliance with standards such as WP29 R155e [20] and ISO/SAE 21434 [56]. Despite its critical importance, TARA is still largely conducted manually, making it labor-intensive, time-consuming, and difficult to scale. Security analysts must repeatedly perform TARA for multiple threat scenarios, an approach that becomes increasingly impractical as vehicle systems grow in complexity and interconnectivity. This inefficiency, combined with the rise of supply chain vulnerabilities, highlights the urgent need for scalable, automated solutions.

Existing datalog-based approaches [27], [51] to automate TARA for improved efficiency primarily focus on vehicle-level assessments, leaving a critical gap in addressing function-level TARA, as required by WP29 R155e and ISO/SAE 21434. Vehicle-level TARA identifies overarching threats that affect the entire vehicle but often overlooks the specific implementation details of individual components. In contrast, function-level TARA examines detailed functions or com-
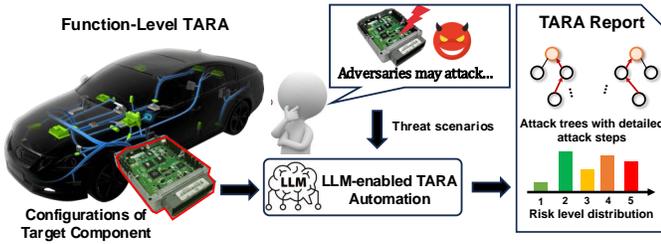
Fig. 1: DefenseWeaver is capable of automating function-level TARA by leveraging the power of LLMs for components with detailed attributes.

ponents, such as battery management systems or individual ECUs, considering their interactions, hardware configurations, software versions, communication channels, and interfaces. This granularity is also crucial in the context of supply chain risks, where vulnerabilities in third-party components can compromise vehicle security. Thus, TARA at the function level offers deeper insight into vulnerabilities and attack paths, making it essential for comprehensive risk management.

Moreover, existing datalog-based approaches [27], [51] rely on predefined threat libraries, which pose major limitations when extending to function-level TARA. These libraries lack the granularity needed to address component-specific threats in function-level analyses and are difficult to maintain amid a rapidly evolving cybersecurity landscape. This raises a critical question: *Can TARA be automated to enable detailed and adaptive function-level assessments?*

Our Approach: We introduce **DefenseWeaver**, a novel system that automates the function-level TARA by leveraging component-specific details and the LLMs. By incorporating detailed, component-specific information, DefenseWeaver dynamically generates attack trees and evaluates associated risk levels. As shown in Fig. 1, users only need to define relevant threat scenarios—DefenseWeaver then conducts the TARA process with minimal manual input, significantly reducing the workload on security analysts. Importantly, the system produces both attack trees and risk assessments, two foundational components of TARA. Attack trees provide structured visualizations of potential threat paths, while risk assessments categorize their severity, enabling prioritized and informed mitigation. When developing DefenseWeaver, we address the following key issues.

Representing Complex Automotive Configurations. Function-level automotive configurations, created during development phases, are diagrammatic models that detail components (e.g., hardware, software), channels, interfaces, and their associated attributes within the IVN. A suitable representation must balance comprehensiveness, machine-readability, and conciseness to ensure efficient processing by LLMs. To address this, we propose the following designs: (i) OpenXSAM++ Format: a structured format to systematically represent automotive configurations, capturing detailed attributes while preserving the logical and visual topology of IVNs. (ii) Logical Path Extraction: abstraction of connectivity

between units for each threat scenario, omitting specific attack techniques or procedures to reduce unnecessary complexity. (iii) Atom Segmentation: decomposition of logical paths into atomic structures, the minimal units that preserve essential topology, enabling efficient and accurate analysis.

Building Attack Trees and Assessing Risk Levels with LLMs. Generating comprehensive attack trees for function-level TARA requires detailed component-level reasoning and the ability to infer attack methods without relying on static threat libraries. To achieve this, we designed a multi-agent framework built on LLMs, with each agent fulfilling a specialized role: (i) *Sub-Tree Constructor*: Creates sub-trees from atomic structures, embedding detailed attack methods for granular analysis. (ii) *Attack-Tree Assembler*: Integrates sub-trees into complete attack trees, ensuring logical consistency and coherence between consecutive nodes. (iii) *Risk Assessor*: Analyzes the feasibility of attack methods and computes the overall risk level for threat scenarios, providing actionable insights into potential vulnerabilities. Together, these agents ensure comprehensive attack trees and rational risk assessments aligned with standard TARA requirements.

Adapting to Evolving Threat Landscapes and Diverse Standards. For full-lifecycle security, the TARA process must adapt to evolving threats and various evaluation standards across regions and stakeholders. To address this, we incorporate Low-Rank Adaptation (LoRA) fine-tuning and Retrieval-Augmented Generation (RAG) to dynamically provide relevant examples and tailored prompts for LLM-based agents. This ensures DefenseWeaver accommodates diverse requirements while maintaining compliance and practicality, enhancing its adaptability and robustness.

We evaluated DefenseWeaver across multiple dimensions to assess its effectiveness, adaptability, and real-world applicability in automating function-level TARA. Deployed across four real automotive security projects, DefenseWeaver successfully identified 11 realistic attack paths, which were validated through penetration testing and subsequently confirmed and patched by the corresponding automakers and suppliers. This demonstrated its practical value in identifying and validating critical attack paths. Beyond automotive applications, DefenseWeaver was tested in non-automotive sectors, including unmanned aerial vehicles (UAVs) and marine navigation systems. Its successful deployment in these safety-critical industries highlights the system's adaptability and robustness.

In comparison to human experts, DefenseWeaver consistently outperformed manual attack tree generation across six assessment scenarios, including these four automotive security projects, UAV, and marine navigation systems. This performance was driven by its ability to avoid common human limitations, including 1) struggled to adapt to new system configurations and overlooked unconventional attacks; 2) inclusion of incorrect elements due to subjective assumptions; 3) overlooking system-specific differences. As a result, the system excelled in *novelty* (+105.00%) and *configuration alignment* (+43.68%), offering a more comprehensive and high-quality risk assessment. Though there was a slight increase in *redun-*

dancy (-0.90%), it reflects the system's comprehensiveness in mining attack paths.

Integrated into cybersecurity management systems used by leading OEMs and suppliers like United Automotive Electronic Systems (UAES) and Xiaomi, DefenseWeaver has generated over 8,200 attack trees, showcasing its scalability and operational efficiency. Notably, the system has enabled enterprises to reduce processing time, greatly improving operational efficiency and supporting compliance with CSMS certification under WP29 R155 regulations. In summary, this paper makes the following contributions:

- We present DefenseWeaver, the first system to automate function-level TARA using component-specific details and LLMs, significantly enhancing efficiency, accuracy, and scalability while reducing reliance on expert input.
- We propose OpenXSAM++, a structured, machine-readable format that enables LLMs to effectively interpret detailed automotive configurations.
- We design a multi-agent LLM framework that automates TARA, enhanced with LoRA and RAG for adaptability and robustness.
- DefenseWeaver has been validated in four automotive projects and demonstrated its adaptability to UAVs and marine systems. It is also integrated into industry systems used by UAES and Xiaomi.

II. BASICS OF AUTOMOTIVE CYBER SECURITY

A. TARA in Automotive Industries

In the automotive industry, TARA should be conducted at different scopes, including vehicle-level TARA and function-level TARA, as shown in Figure 2. Function-level TARA addresses cybersecurity threats specific to individual components (e.g., the BCM component) or groups of peripheral components (e.g., IVI, Gateway, OBD, and TPMS) that perform critical functions (Figure 2b), while vehicle-level TARA operates at a higher level and assumes attackers cannot directly access internal elements (Figure 2a). This distinction leads to three main differences: (i) *Function-level TARA can account for attack entries that vehicle-level TARA may overlook.* For example, a JTAG interface connecting to the MCU, as shown in Figure 2b, requires detailed information about the hardware and software of internal elements (e.g., the MCU, radio module, and SPI channel) to evaluate potential vulnerabilities effectively. (ii) *It provides more specific attack scenarios tailored to different vehicle types, even with similar IVN topologies.* For instance, while logical attack paths like IVI-GW-BCM may remain consistent across different vehicle models, the risk levels of these paths can vary significantly due to differences in OEM implementations (e.g., hardware, software, and suppliers). (iii) *It enables full-lifecycle TARA with dynamic risk assessment.* Automotive systems often undergo updates, including OTA updates, hardware replacements, and software patches. Function-level TARA offers the flexibility to analyze these changes in detail, ensuring threat analyses and risk databases are updated to reflect the latest system state. This

adaptability is essential for maintaining cybersecurity across the vehicle's lifecycle, accounting for evolving vulnerabilities and system configurations.

However, today's TARA activities still rely heavily on human analysts, and this manual approach presents two critical limitations. First, cognitive biases, subjective assumptions, and incomplete attention to component-specific details can produce inconsistent or incomplete results—an issue that becomes more acute at the function level, where far finer-grained information must be considered. Second, because vehicle architectures evolve rapidly, threat analyses and risk databases require continual updates, which demand scarce specialist expertise and ongoing training. Together, these challenges underscore the need for more efficient, automated techniques that can scale function-level TARA and sustain robust cybersecurity throughout the vehicle lifecycle.

B. TARA Pipeline.

TARA, as outlined in ISO/SAE 21434, systematically identifies cybersecurity threats, evaluates associated risks, and implements countermeasures to enhance vehicle security and ensure regulatory compliance. It begins with **Item Definition**, where the system's components and interfaces are modeled, providing a basis for identifying **Assets**—critical elements such as ECUs, communication interfaces, or sensitive data—evaluated according to confidentiality, integrity, and availability. Potential **Threat Scenarios** describe how attackers might compromise these assets, followed by **Attack Path Analysis**, often visualized with attack trees or graphs to represent all possible routes. Next, a **Feasibility Rating** estimates the effort, expertise, and resources needed for a successful attack, while an **Impact Rating** quantifies possible consequences (e.g., safety, financial, operational, privacy). These ratings combine to yield a **Risk Level**, which informs **Risk Treatment Decisions**—such as avoidance, mitigation, sharing, or acceptance. Since threats evolve over time, TARA must be continuously updated throughout the vehicle lifecycle to ensure risk assessments remain accurate and comprehensive.

C. Mandatory Regulations

In 2021, the United Nations Economic Commission for Europe (UNECE) introduced WP29 R155e, the first mandatory automotive cybersecurity regulation. It established a two-tier certification system for cybersecurity compliance: the Cyber Security Management System (CSMS) for OEMs and the Vehicle Type Approval (VTA) for individual vehicle types. The CSMS focuses on manufacturers' organizational processes for managing cybersecurity risks across the vehicle lifecycle, mandating that all OEMs in UNECE member countries hold a certified CSMS. Meanwhile, the VTA ensures each vehicle type meets specific regulatory standards through technical tests, verifying that vehicles are developed under a certified CSMS and can detect and respond to cyberattacks. These certifications impose rigorous responsibilities on OEMs and suppliers, requiring comprehensive, lifecycle-spanning cyber-

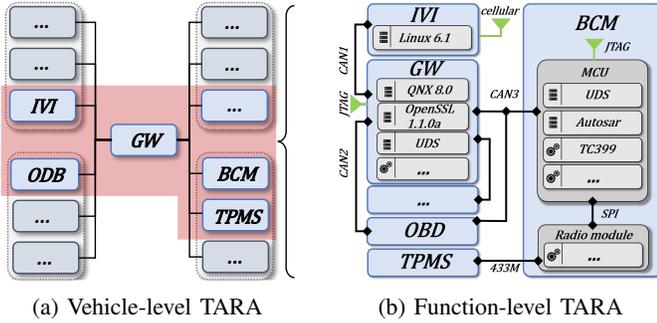(a) Vehicle-level TARA   (b) Function-level TARA

Fig. 2: Comparison of (a) vehicle-level TARA and (b) function-level TARA. Function-level TARA needs to consider extra dimensions such as hardware configurations (e.g., TC399), software versions (e.g., OpenSSL 1.1.0a), communication channels (e.g., CAN bus), interfaces (e.g., JTAG), internal connections (e.g., radio module and MCU) to comprehensively evaluate potential vulnerabilities.

security assessments to uphold industry growth and regulatory standards.

III. RELATED WORK

Automating Tools for TARA: Traditional TARA tools, such as SAHARA [38], EVITA [50], HEAVENS [26][33], and TVRA [24][25], provide systematic frameworks for identifying and assessing threats in automotive systems, relying on methodologies like attack trees[52] or STRIDE. However, these tools are not automated, depending heavily on manual effort and expert knowledge—an increasing challenge in modern complex systems with numerous components and potential attack paths. By contrast, datalog-based tools like MulVal [44][45][46][47] automate parts of TARA by using the logic programming language to encode vulnerabilities, threats, and reasoning rules into a library, which then generates possible attack paths. Building on MulVal, Saulaiman et al.[51] and CarVal[27] tailored it for the automotive domain, with CarVal incorporating expert interviews to manually establish a more comprehensive threat database.

However, these datalog-based solutions remain unsuitable for function-level TARA, which demands detailed, component-specific assessments and faces high system variability. Maintaining granular, dynamic threat libraries is labor-intensive and difficult to scale. Moreover, TARA applies to a range of other systems—such as aircraft, ships, extended reality (XR), and Space Information Networks (SIN)—where constructing reusable, cross-domain threat databases remains a major obstacle.

Attack Tree Generation: Attack trees are the core artifact produced during TARA. By iteratively decomposing high-level threats into concrete attack steps, they provide both a systematic analysis framework and an intuitive medium for communicating risk among engineers and regulators [52], [31]. Several studies have sought to automate their construction in the automotive domain [28], [29], [16], but similarly, they are rule-based and only work at the vehicle-level.

Success of Large Language Models (LLMs): LLMs have profoundly advanced natural language processing and machine learning, sparking transformative changes across diverse fields. Since the introduction of the transformer architecture [58] in 2017, models like BERT [18] (2018) and GPT-4 [13] (2023) have demonstrated remarkable capabilities, owing to billions or even trillions of parameters and training on massive, varied datasets. They excel at generalizing across tasks and adapting to new challenges. In cybersecurity, LLMs have proven effective for vulnerability detection [66][48][65], code fuzzing [41][62], phishing detection [34][30][35][37], and content moderation [32], leveraging fine-tuning or prompting to tailor solutions. Inspired by these advances, we ask: Can we replace static threat libraries in traditional TARA tools with the vast knowledge base of LLMs and thereby automate the TARA process?

IV. DEFENSEWEAVER: APPROACH

In this section, we present the design of DefenseWeaver, an LLM-based tool for function-level TARA automation.

A. System Overview

DefenseWeaver automates function-level TARA by leveraging component-specific details and the capabilities of LLMs. Unlike approaches relying on static threat libraries, DefenseWeaver dynamically infers attack methods and evaluates risk levels using detailed component-specific information. This scalable and adaptive system overcomes the limitations of manual processes and static libraries, enhancing efficiency, accuracy, and responsiveness to evolving cybersecurity challenges. Its architecture consists of five key components, each contributing to its overall functionality and adaptability.

Automotive Configurations and Threat Scenarios (Input): Automotive configurations detail component attributes, including hardware setups, software versions, communication channels, interfaces, and sub-component interactions. Threat scenarios specify attack objectives, the endpoint, and entry points, providing essential context for precise function-level TARA and ensuring comprehensive vehicle configuration analysis.

Atomic Structure Representation: This component decomposes complex configurations into manageable units, thereby improving TARA efficiency and accuracy. Based on the structured OpenXSAM++ format, it constructs a directed graph to identify logical paths within the IVN and segments these paths into atomic structures. Each atomic structure retains essential topological and functional information, facilitating subsequent LLM-based analysis.

LLM Agent-Based Attack Methods Inference: DefenseWeaver employs a multi-agent framework to dynamically infer attack methods, assigning specialized roles to LLMs such as Sub-Tree Constructor, Attack Tree Assembler, and Risk Assessor. For example, the Assembler links sub-trees and may request the Constructor to regenerate methods if inconsistencies arise.
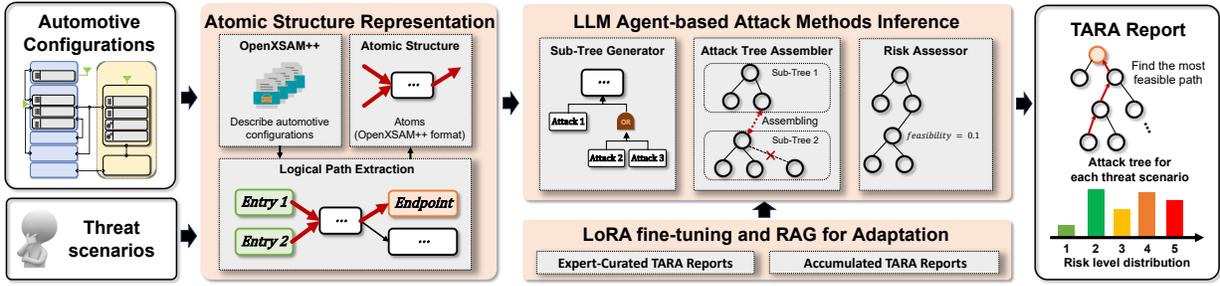
Fig. 3: Framework of DefenseWeaver. Given a vehicle configuration and threat scenarios, DefenseWeaver will automatically convert the visual diagrams into proper representations (atomic structures), generate specific attack methods for each node (sub-trees) before assembling them into attack trees, and evaluate the risk level (from 1 to 5) according to the most feasible attack path for each threat scenario.
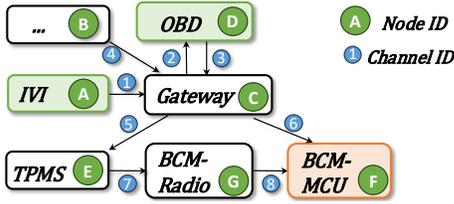

Fig. 4: A simplified IVN topology with one unique attack endpoint (BCM-MCU) and two entrypoints (IVI and OBD) according to given threat scenarios (e.g., disrupt the availability of BCM-MCU). The nodes are connected with channels.

Fine-tuning and RAG for Adaptation: To adapt to dynamic threats and diverse standards, DefenseWeaver integrates Low-Rank Adaptation (LoRA) fine-tuning and Retrieval-Augmented Generation (RAG), which learns from expert-curated TARA reports and relevant contextual information to refine analysis and tailor it to specific requirements. This integration ensures adaptability across regions and organizational standards by leveraging real-time and domain-specific knowledge.

TARA Report (Output): The output is a function-level TARA report that consolidates identified attack methods, risk levels, and analysis results into an actionable document. It provides detailed insights into vulnerabilities, attack paths, and recommended mitigations, serving as a critical tool for automotive OEMs and suppliers to ensure regulatory compliance and maintain robust cybersecurity throughout the lifecycle.

Compared to datalog-based approaches, DefenseWeaver is able to: (i) automatically identify various attack paths/methods even with identical logical paths by considering component-specific details, (ii) discover new attack surfaces, thereby adapting to evolving threat landscapes and enabling full-lifecycle assessment, and (iii) easily deploy the pipeline to other electronic systems (e.g., UAVs and ships, etc.) for cross-domain applicability and peripheral devices of vehicles (e.g., cloud services and smartphones, etc.).

B. Atomic Structure Representation

In this section, we discuss how to efficiently describe vehicle configurations using atomic representations.

1) Comprehensive Description of Configurations: Vehicle configurations, as illustrated in Fig.4, are often represented by visual diagrams that depict various components and the connections among them. These diagrams capture key design details but are difficult to interpret automatically, which is a significant challenge for LLM-based parsing (see Sec.IV-C) and dataset construction (see Sec. IV-D).

To overcome this issue, we convert these visual diagrams into a structured format called OpenXSAM++, an extension of OpenXSAM [3] (***Open X**ml **S**ecure **A**nalysis **M**odel). OpenXSAM is an XML-based framework designed for information exchange in automotive cybersecurity and risk management. It uses standardized, machine-readable documentation to describe assets, threats, risks, and mitigation measures. However, its original specifications do not include several essential elements and attributes needed to cover automotive configurations thoroughly.

To fulfill this gap, we add a `Software` attribute that specifies each component's operating system, software bill of materials, or active network services. We also introduce a `Hardware` attribute to outline hardware modules, chips and debugging capabilities. Furthermore, we incorporate additional elements, such as `Channel` and `Interface`, to capture the breadth of automotive components and their interconnections. While these enhancements focus on vehicle systems, they are also applicable to other electronic or electrical systems. **Therefore, DefenseWeaver relies on OpenXSAM++ format for configuration representation and database construction.**

2) Logical Path Extraction and Atom Construction: Vehicle configurations, with their many interconnected components, often lead to very long OpenXSAM++ descriptions that capture comprehensive details. These extensive descriptions may overwhelm LLMs when generating attack trees, as they introduce substantial irrelevant or distracting information [61]. Therefore, and as illustrated in Fig. 5, **we refine the OpenXSAM++ description before fed into LLMs based on the threat scenarios in two main steps**: (i) logical path
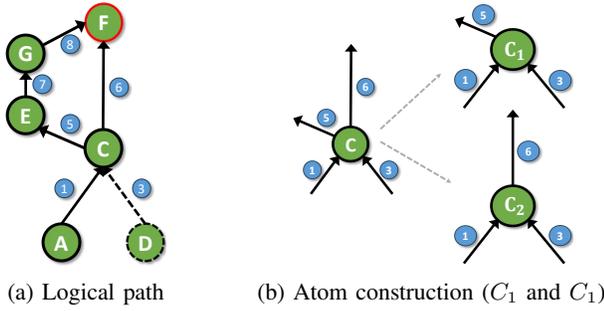
(a) Logical path   (b) Atom construction (C_1 and C_1)

Fig. 5: (a) Logical paths without detailed attack methods in each node, where irrelevant and redundant components (e.g., B) are removed. (b) The segmented node (C) and constructed atomic structures (C_1 and C_2) are derived based on the exit channels (e.g, channel 5 and 6) for sub-tree generation.

extraction to remove redundant or irrelevant components, and (ii) atom construction to split the description into several minimal units (atoms) while preserving topology.

Logical Path Extraction. Because each threat scenario defines a unique attack endpoint (e.g., BCM-MCU (F)) as well as several possible entry points (e.g., IVI (A) and OBD (D)), our goal is to identify every path from each entrypoint to the attack endpoint (for example, $A \rightarrow C \rightarrow F$ in Fig. 5a). By doing so, we discard any irrelevant or redundant elements (e.g., component B and channels 2 and 4) and thus focus on components and channels that genuinely matter to the threat scenarios. Note that these logical paths do not include detailed attack methods per node; such methods are the core of TARA analysis and are inferred by LLMs in later steps.

To systematically generate these paths, we build a directed graph from the OpenXSAM++ description and apply a depth-first search (DFS) to find all acyclic routes connecting each entrypoint to the attack endpoint. We remove cyclic paths (for example, $D \rightarrow C \rightarrow D$), as revisiting a previously compromised component provides no further insights for practical TARA. If there are multiple possible entry points, we simply find each path independently and merge any shared segments.

Atom Construction. Directly supplying all nodes and channels from the extracted logical paths to LLMs may still degrade analysis quality by overloading the model with too much information and complex interactions. To mitigate this, we break each logical path into smaller, more manageable structures (i.e., atoms), while preserving the relationships among nodes. Specifically, an atom is defined as a single node plus all its directly connected edges (i.e., channels). The channel attributes store references to any other connected nodes. For instance, in Fig. 5b, node C has two incoming channels and two outgoing channels. Parsing node C with these channels allows the LLM to infer the previous and next nodes from the channel attributes. Since C links to node F through channel 6 and to node E through channel 5, we split it into two atoms, C_1 and C_2. This ensures each atom has exactly one local attack objective (e.g., propagating the attack to F via channel 6, or to E via channel 5), thereby simplifying

the subsequent inference.

C. LLM-based Attack Methods Inference

Inferring potential attack methods for each node is central to TARA analysis, accounting for the majority of the required time and effort. In this section, we explain how to leverage LLMs to automate the inference of attack methods.

1) Multi-Agent Roles for Automated TARA: Component-specific details—such as software versions, hardware configurations, communication channels, and interfaces—significantly increase the complexity of attack method inference because the number of potential attacks can grow exponentially. Additionally, generating attack trees must factor in the interactions between components and channels while assessing the feasibility of each attack method. Consequently, it is not feasible to generate complete attack trees using simple Q&A approaches.

Recent advances in multi-agent systems have greatly enhanced the capabilities of LLMs in handling complex tasks, including software development [23][49][19][59], game simulation [64], scene simulation [60], and multi-robot systems [39]. The key idea of multi-agent systems is to **mimic human teamwork by dividing a complex task into several simpler subtasks, each handled by an agent with specialized skills**. Through collaboration, these specialized agents reduce the likelihood of errors, especially in tasks with high complexity. Drawing on this approach, we divide TARA analysis into three subtasks, each handled by a separate LLM-based agent. Their roles are as follows, while detailed processes are described in Sec.IV-C2 and Sec.IV-C3:

● **Sub-Tree Constructor:** Generate sub-trees for each atom. Each sub-tree focuses on a single local attack objective for a specific node and includes comprehensive attack methods with corresponding logical relationships (e.g., AND/OR) among the summarized attack methods to achieve the objective.

● **Attack-Tree Assembler:** Merges the sub-trees produced by the *Constructor* into a complete attack tree for each threat scenario. The *Assembler* also collaborates with the *Constructor* iteratively to improve the coherence of connected sub-trees (which are generated independently from different atoms) and to remove leaf nodes that violate user-defined constraints.

● **Risk Assessor:** After constructing complete attack trees for each threat scenario, the *Assessor* first evaluates the feasibility of each attack method (i.e., step feasibility). It then assesses the cumulative feasibility of the entire tree (i.e., focusing on the most feasible attack path) and the potential impact of the threat scenario. Based on these assessments, the *Assessor* determines the risk level for each scenario according to the ISO/SAE 21434 standard.

In practical functional-level TARA processes, there may be tens or even hundreds of threat scenarios, necessitating repeated application of the above steps for each scenario. Ultimately, the TARA reports provide a comprehensive risk distribution that summarizes the risk levels of all potential threat scenarios for the analyzed target.

2) Attack Tree Generation: Attack tree generation aims to construct comprehensive and coherent attack trees for given

threat scenarios. Note that, we need to generate an attack sub-tree for each atom before assembling them into a complete attack tree. Fig. 6 provides an example of an attack tree that consists of sub-trees for nodes A, C_1, and F, under the threat scenario "Disrupt the availability of BCM-MCU". The attack methods for each node are derived based on component-specific details, which are hardly included in predefined static libraries.

a) Sub-Tree Construction: To generate sub-trees, the *Constructor* needs to summarize specific attack methods for the input atoms, relating them to the threat scenario and node attributes. However, some attributes of the node may only have simple descriptions; for example, the software attribute of the IVI might be as simple as "Linux 6.1", which may cause LLMs to overlook some attributes, resulting in incomplete analysis. Additionally, understanding the logical connections among the generated attack methods for each node is crucial for improving the readability and quality of the entire attack tree, but it is often challenging for LLMs to consider so many details simultaneously.

To make the attack methods comprehensive and tightly connected to the threat scenario, **we adopt the concept of Chain-of-Thought (CoT) to guide the construction of sub-trees** through the following steps: (1) attack surface inference, (2) threat scenario analysis and local attack objective understanding, and (3) attack sub-tree generation. First, inferring the potential attack surfaces of the given node ensures comprehensive identification of the vulnerabilities of each component. Second, the *Constructor* combines the threat scenario (e.g., "Disrupt the availability of BCM-MCU") and attack surfaces to infer the local attack objective (e.g., "Make gateway send incorrect data..." for the gateway node and "Make IVI send erroneous lighting commands..." for the IVI node). After that, the *Constructor* formulates a series of specific attack methods closely related to the local objective, combining key information from the node, the attack surface, and its extensive cross-domain knowledge. Note that each attack method contains only one operation to better support feasibility rating in Sec. IV-C3. In other words, there might be several attack methods required to launch an attack against one attack surface.

As a result, the logical relationships among attack methods are also important to demonstrate the practical attack path. If several attack methods rely on the "Linux 6.1" attribute and must be executed in sequence to launch an attack, the process might involve obtaining the Linux system firmware from the IVI, reverse-engineering the firmware to identify vulnerabilities, and exploiting a known Linux vulnerability (e.g., CVE-2023-0179) to gain control of the IVI. These sequential attack methods should be connected with a logical AND. Conversely, if completing any one of the attack methods is sufficient to achieve the attack objective, they can be connected with a logical OR. Therefore, based on the attack objectives, the *Constructor* further analyzes the logical relationships between the attack methods and connects them together with logical nodes (e.g., AND and OR) to form the final attack sub-tree. For
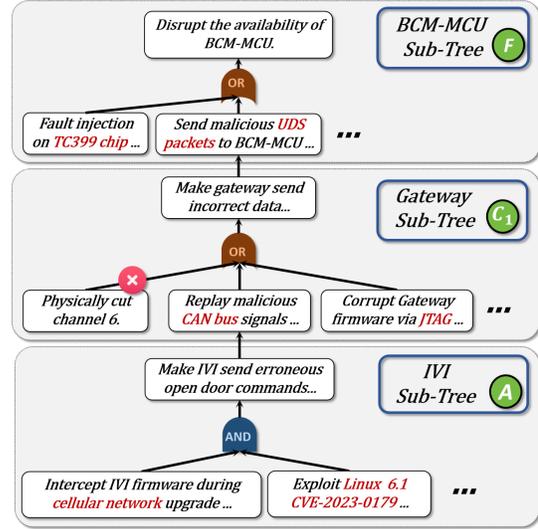

Fig. 6: Simplified attack tree assembled from sub-trees for IVI (A), Gateway (C_1), and BCM-MCU (F). The realizations of attack methods for each node are derived based on component-specific details, which hardly be included by predefined static libraries.

example, as shown in Fig. 6, if the attack objective is "Make the gateway send incorrect data to the BCM-MCU," and accomplishing either one is sufficient to achieve the objective (i.e., Accessing the gateway via JTAG to corrupt the BCM-MCU firmware or Replaying malicious CAN bus signals on channel 6 to the BCM-MCU), they can be connected with an "OR" node.

b) Attack Tree Assembling: The *Assembler* is responsible for assembling the attack sub-trees generated by the *Constructor* into a complete attack tree. Therefore, the *Assembler* needs to determine how to connect two sub-trees with multiple attack methods. The key observation is that **only attack methods related to the channels can propagate the attack outcomes of one component to another**, which can be easily distinguished from the attack surfaces. For example, an attacker can exploit the IVI to replay malicious CAN bus signals over Channel 1, conducting a Denial-of-Service (DoS) attack on the gateway. Consequently, the *Assembler* connects the sub-tree generated for the IVI to the corresponding attack method related to the communication channel in the sub-tree of the gateway (e.g., "Replay malicious CAN bus signals on channel 6 to BCM-MCU.").

In addition to simply connecting sub-trees, **another important task of the *Assembler* is to validate the quality of the *Constructor*'s generation**. For example, since these sub-trees are generated independently, the attack methods related to the channels (shared by two nodes) may lack coherence due to the lack of a global perspective of both nodes. In such cases, the *Assembler* can request the *Constructor* to improve the coherence of these attack methods by providing essential information about the two sub-trees (e.g., the local

Fig. 7: Risk distribution for more than 100 threat scenarios of the IVN configuration shown in Fig. 4. For each threat scenario, the attack tree and risk evaluation will be generated independently.

attack objective of the previous node and the attack method of the input channel of the next node). Furthermore, the *Assembler* can be easily customized to adapt to users' different requirements by providing explicitly defined constraints for attack methods, such as excluding social engineering attacks (e.g., stealing key fobs or passwords), physical destruction of components (e.g., damaging signal transceivers, cutting hardware, or chip replacement), and physical attacks (e.g., side-channel analysis, fault injection, or chip decapping), etc. Therefore, the attack method "Physically cut channel 6" will be removed.

The main difference between the *Constructor* and the *Assembler* is that the *Constructor* focuses on generation and inference, while the *Assembler* focuses on validation. Therefore, the *Assembler* can double-check the generated sub-trees and request the *Constructor* to regenerate some of them if necessary.

3) Risk Assessment of Threat Scenarios: In an automotive system—or even within a single component—there can be tens or hundreds of threat scenarios. Therefore, it is required to build attack trees independently for each threat scenario in a TARA report. The next challenge is **how to assess the risk level of each threat scenario to provide an overall analysis result to security analysts and prioritize countermeasures**. Since it is usually impossible to address all threats simultaneously and manual analysis of all attack trees requires a huge amount of human effort (as currently done in industry TARA analysis), efficient risk assessment is crucial.

As suggested by the ISO/SAE 21434 standard, the risk level of threat scenarios can be divided into two parts: overall attack feasibility and potential impact of the threat. After determining the scores of these two factors (i.e., high, medium, low, and very low for attack feasibility, and severe, major, moderate, and negligible for potential impact), the risk levels (from 1 to 5) can be summarized in a risk distribution table. An example is shown in Fig. 7. For example, according to the standard, a threat with *High* Feasibility and *Serious* Impact is assigned the highest risk level 5 and decrement the risk level as feasibility and impact decrease. Notably, threats with a risk level greater than risk level 3 (i.e., *Moderate* Impact and *High* Attack Feasibility; *Major* Impact and *Medium* Attack Feasibility; *Serious* Impact and *Low* Attack Feasibility) have

practical inspection value and need to be verified in subsequent penetration testing. **The *Risk Assessor* determines the attack feasibility and potential impact of each threat.**

a) Attack Feasibility Assessing: It is often unreliable to directly analyze the entire attack tree to determine the feasibility of a threat scenario. Therefore, our high-level strategy is to analyze the step feasibility of each individual attack method before evaluating the cumulative feasibility score. Furthermore, for each attack method, we score it across the following five dimensions according to the ISO/SAE 21434 standard: elapsed time (ET), specialist expertise (SE), knowledge of the item or component (KoIC), window of opportunity (WoO), and equipment (Eq). For each dimension, a lower score means higher feasibility. To compute the step feasibility for each node, the *Risk Assessor* is first required to choose a score from a given range for each dimension, before providing a brief explanation to ensure reasonable scores. Note that the scoring standards may vary across different regions, companies, and even products, which can be customized by users as discussed in Sec. IV-D.

The attack feasibility of a specific threat scenario is determined by the most feasible attack path (i.e., the one with the lowest overall cumulative score), although there are usually multiple attack paths in an attack tree from leaf nodes to the root node. Moreover, the cumulative feasibility of one attack path is determined by the hardest attack method (i.e., the one with the highest score). Therefore, the principles for computing the cumulative feasibility (divided into five dimensions) are as follows: (1) For sequential nodes, the cumulative scores of the current node are the maximum of the cumulative scores of its child node and its own step-feasibility scores; (2) For logical AND nodes, the cumulative scores are the highest scores among their child nodes; (3) For logical OR nodes, the cumulative scores are assigned the scores of the child node with the lowest overall score. To summarize, the cumulative feasibility is computed in a bottom-up approach, and the cumulative feasibility of each first leaf node equals its step-feasibility score. Therefore, the cumulative scores at the root node represent the scores of the most feasible attack path, and the sum of the cumulative scores is the overall attack feasibility of the threat scenario.

b) Potential Impact Assessing: Assessing attack feasibility is more challenging since it involves technical details of the system (i.e., it is product-specific). In contrast, assessing potential impact only requires a high-level understanding of threat scenarios and system usage. In practice, impact assessment is conducted by evaluating potential consequences across four dimensions, including Safety, Financial, Operational, and Privacy. Since impact assessment is not product-specific, DefenseWeaver scores each of the four dimensions separately, before calculating the potential impact based on the ISO 21434 standard.

D. Fine-Tuning and RAG for Adaptation

To cope with evolving threat landscapes and diverse standards, DefenseWeaver adopts a differentiated adapta-

tion strategy: using Retrieval-Augmented Generation (RAG) for the Risk Assessor agent and Low-Rank Adaptation (LoRA) fine-tuning for the Sub-Tree Constructor agent. The Assessor (which generates step-feasibility scores and impact scores) benefits from real-time retrieval of authoritative references and up-to-date enterprise guidelines. In contrast, the Constructor (which builds attack sub-trees) is fine-tuned via LoRA to internalize expert attack logic. We found that applying RAG to the Constructor may introduce irrelevant or incompatible examples (e.g., pulling QNX-specific nodes into a Linux analysis), thereby degrading generation quality. LoRA fine-tuning avoids this by embedding the correct patterns in the model's parameters instead of relying on potentially noisy external retrieval.

Specifically, we leverage the OpenXSAM++ format (Section IV-B1) with three critical fields added to each analyzed node: "Sub-Tree" (the expert-annotated attack subtree), "Step-Feasibility" and "Impact" (the expert-assessed scores). These fields support our two adaptation processes. For the Constructor, we construct a supervised training set using each node's system attributes (e.g., hardware configurations, software versions) as the input and the corresponding expert "Sub-Tree" as the output. We then apply LoRA fine-tuning to the base LLM using this dataset, teaching the model to generate attack sub-trees in line with expert logic. This fine-tuning updates only a small portion of model weights (preserving over 95% of the original parameters), thus retaining the model's general language capabilities while infusing domain-specific patterns. We also apply regularization (dropout 0.3) and early stopping (halting training if validation F1 stagnates for 3 epochs) to prevent overfitting. For the Assessor, we implement RAG-based prompt augmentation by retrieving similar prior attack methods and their scores (i.e., feasibility and impact) from our knowledge base. An embedding-based similarity search [15], [43] finds the most relevant historical cases (from both expert and enterprise data), and the top matches are inserted into the Assessor's input prompt [36]. This gives the LLM concrete reference points for scoring, ensuring its assessments are grounded in authoritative examples and can dynamically adapt to the latest enterprise context via real-time retrieval.

Our adaptation strategy is grounded in **two primary databases** with domain knowledge, which feed the LoRA training and RAG retrieval components: (i) Expert-Curated TARA Reports: A corpus of 116 vetted automotive threat scenarios (from an industry reference library) provides over 1,000 standardized attack sub-trees and impact scores, and about 5,000 step-feasibility entries. The sub-trees serve as high-quality training targets for LoRA fine-tuning, while the score ratings populate the Assessor's RAG reference library. (ii) Enterprise-Specific TARA Reports: DefenseWeaver also ingests feedback from enterprise-specific assessments collected via a GUI (Section V-A). When users adjust an attack tree or scores in practice, the corrected sub-tree is added as incremental training data to further refine the LoRA-based Constructor, and the updated scoring data is immediately incorporated into the RAG retrieval library.

V. EVALUATION

In this section, we apply DefenseWeaver to real-world scenarios and compare its attack trees with those crafted by human experts. We demonstrate that DefenseWeaver can identify practical attack paths (validated via penetration tests) and produce higher-quality attack trees.

A. Implementation

Setup: We developed an interactive web application for DefenseWeaver with a user-centered design. Users can draw or import system models and specify a threat scenario (defining the attack entry point(s) and endpoint). Based on this input, DefenseWeaver automatically generates a detailed attack tree and evaluates the feasibility of each attack path. The interface allows users to interactively refine results: for example, they can modify or add attack methods and logic nodes, adjust feasibility ratings, or reuse portions of attack trees when system components are updated. Once an attack tree is finalized, the tool serializes it in the extended OpenXSAM++ format (including fields such as "Sub-Tree", "Step-Feasibilit" and "Impact"). These results are stored as the training set for LoRA fine-tuning and update the RAG dataset for future analyses. The base model in our implementation is ChatGPT-4 [13].

Integration in Industry: DefenseWeaver has been integrated into leading automotive manufacturers including Xiaomi Auto and United Automotive Electronic Systems (UAES) as a core component of their cybersecurity solutions. According to system operation statistics, it has consumed around 300 million tokens in 3 months and generated more than 8,200 attack trees (around 36,500 tokens per attack tree on average). In daily operation, the system evaluates 90+ production threat scenarios, giving engineers continuous, fine-grained risk visibility. Participants from these enterprise deployments highlight DefenseWeaver's real-world impact: The UAES security manager commented, "We rely on DefenseWeaver for TARA analysis and attack tree generation, which has been instrumental in achieving R155 compliance. Its efficiency makes it indispensable for our operations." Similarly, Xiaomi TARA manager noted, "DefenseWeaver's automated attack tree generation effectively solves critical issues and greatly improves our workflow efficiency." These large-scale deployments prove that DefenseWeaver's effectiveness meets the cybersecurity demands of modern automotive development.

B. Open Science and Ethics Considerations

Since DefenseWeaver has been integrated into commercial automotive cybersecurity management platforms, we cannot fully open-source the source code of DefenseWeaver due to proprietary licensing constraints. However, in accordance with open science principles, we will release a community version of our web application and open-source the database built from expert-curated TARA reports in [4], thereby enabling scientific research in this field. All vulnerabilities discovered in real vehicles have been reported to the responsible parties and have since been resolved. We have a demo video in [5].

C. Experimental Analysis on Real Vehicles

1) Vehicles Under Examination: To evaluate DefenseWeaver's effectiveness in function-level TARA, we deployed it in four real automotive security assessment projects. These involved four distinct in-vehicle components: two Body Control Modules (BCM) in different vehicles (Car A and Car B), one Cockpit Domain Controller (CDC, Car C), and one Passive Keyless Entry and Start system (PKES, Car D). The OEMs provided detailed configuration models for each component (see Fig. 8a-8d), which we imported into DefenseWeaver. For each case, we defined a representative threat scenario and generated the corresponding attack tree with feasibility and risk evaluations. In accordance with industry practice (WP.29 R155 and ISO 21434), we treated any path rated with at least "Medium" feasibility as a candidate for penetration testing, since such paths warrant deeper security analysis. Guided by DefenseWeaver's output, we conducted targeted security tests on Cars A–D and ultimately confirmed 11 practical attack paths (each with risk level ≥ 3) via proof-of-concept (POC) exploits. (Appendix A provides the complete attack trees and detailed attack steps for these scenarios.)

For the two BCM cases (Car A and Car B), we chose an identical threat scenario aiming to disable the BCM's door control function (i.e., compromise its availability). This scenario has the same high-level attack entry point (the IVI infotainment unit) and endpoint (the BCM's microcontroller) in both vehicles, following a logical path IVI→Gateway→BCM. For Car C (CDC), the threat scenario targeted the OTA update mechanism, aiming to install unauthorized firmware on the CDC. For Car D (PKES), the scenario focused on unlocking the vehicle's doors without the owner's authorization by exploiting the wireless key system. All four components have complex setups of hardware, software, and network interfaces, providing a rigorous testbed for DefenseWeaver.

After DefenseWeaver generated the initial attack trees and risk assessments for each scenario, we examined the output to identify high-feasibility attack paths for validation. All attack paths discussed below were rated at least "Medium" feasibility by the system, and thus merited real-world testing. We next describe the results for each scenario, including the proof-of-concept (PoC) attacks we performed and key insights derived from each case.

2) BCM: Identical Logical Paths but Different Attack Methods Due to Configuration Variations: Modern vehicle E/E architectures are increasingly centralized, often placing a BCM behind a gateway that mediates inbound and outbound communications (e.g., software updates via Unified Diagnostic Services, UDS). Thus, similar threat scenarios against different BCMs may share the same high-level logical chain (IVI→Gateway→BCM), yet yield distinct attack paths due to differences in hardware/software configurations. To illustrate, we analyzed the BCM of two vehicles (Car A and Car B, see Fig. 8a and Fig. 8b) under the same scenario of disabling the door-unlock function. DefenseWeaver produced

similar logical attack trees for both, but with divergent specific exploits reflecting each vehicle's nuances. For example, Car A's gateway runs OpenSSL 1.1.0a, which is vulnerable to a known buffer overflow (CVE-2016-6309 [11]), whereas Car B's gateway uses a newer OpenSSL 3.3 (not affected by that CVE) but exposes a UART debug interface absent in Car A's gateway. DefenseWeaver accordingly identified different feasible attack vectors: exploiting the OpenSSL CVE in Car A for remote code execution, versus leveraging the UART interface and a Linux kernel vulnerability in Car B.

PoC attack. We validated DefenseWeaver's suggested attack paths on both vehicles. **Car A**: We first obtained physical access to the BCM's microcontroller via its JTAG debug port and dumped the firmware. From this, we reverse-engineered the BCM's seed2key authentication algorithm for the UDS SecurityAccess service [57], giving us the ability to bypass its security handshake. We then compromised the IVI unit (e.g., via an existing weakness) and sent crafted TCP packets to the gateway, exploiting the OpenSSL CVE-2016-6309 vulnerability to gain a remote shell on the gateway. Using this shell access, we injected CAN messages to the BCM, ultimately reprogramming the BCM's firmware once the UDS authentication was bypassed. This multi-step attack was rated *Medium* feasibility, with *Major* potential impact, yielding an overall risk level 3. **Car B**: Here, DefenseWeaver also guides us to reverse-engineer the algorithm for the UDS SecurityAccess service in the BCM's firmware. Using this knowledge, we reconstructed the BCM's unlock authentication and then connected to the gateway via its UART interface. Through the UART, we exploited a privilege escalation vulnerability in the gateway's Linux OS (CVE-2023-0179 in Linux 6.1) to fully compromise the gateway. From the gateway, we were able to send UDS commands to overwrite the BCM's firmware. This Car B attack path was assessed as *High* feasibility, *Major* impact, and risk level 4.

Insights. Despite an identical high-level attack chain, Cars A and B demanded different exploits—highlighting two broader risk factors. (i) UDS access: Automotive security analysts often assume that UDS-based attacks require direct physical access (e.g., via OBD-II ports); however, our results show that network-facing vulnerabilities can enable remote exploitation of UDS services. (ii) Outdated software: Legacy software versions (e.g., running an obsolete OpenSSL library) can expose a vehicle to N-day vulnerabilities like CVE-2016-6309, which attackers can remotely leverage. **These findings demonstrate DefenseWeaver's strength in generating distinct, practical attack paths by accounting for each vehicle's specific configuration.** In contrast, static, datalog-based TARA tools [27], [51] that rely on predefined threat libraries would treat these two BCM scenarios similarly and likely miss such configuration-dependent attack vectors.

3) CDC: New Attack Vectors from Cloud Services: Modern intelligent vehicles often include a Central Domain Controller (CDC) with Over-The-Air (OTA) capabilities, allowing manufacturers to remotely push software updates from the vehicle cloud. In addition, various external systems—such as cloud
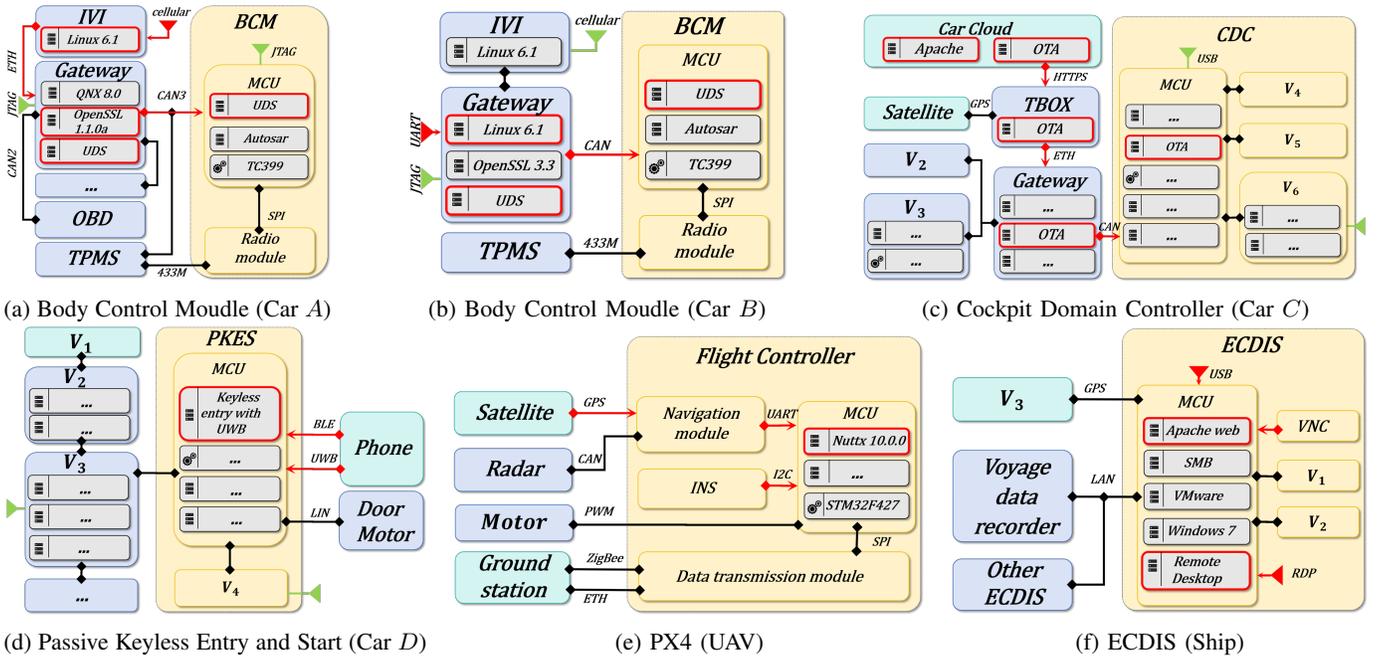
Fig. 8: Configurations of 6 systems. (a) and (b) are BCMs of Car A and Car B with the same logical path, demonstrating DefenseWeaver's ability to generate distinct attack paths by considering the component-specific details. (c) CDC of Car C. DefenseWeaver can comprehensively cover peripheral devices and cloud services in its anaylsis. (d) PKES of Car D. DefenseWeaver successfully discovers recently emerged attack surfaces. (e) PX4 architecture of UAV and (f) ECDIS of Ship show that DefenseWeaver can be applied to various electrical and electronic systems.

servers and mobile apps—interact with the CDC without a direct physical connection to the car, thereby introducing new attack surfaces in automotive TARA. Attackers can leverage vulnerabilities in the cloud infrastructure or in the CDC's firmware-verification logic to construct malicious update packages and bypass integrity checks, ultimately tampering with critical components. We used DefenseWeaver to examine the OTA update process of the CDC in Car C (Fig. 8c), generating an attack tree that highlights multiple threat vectors (Fig. 12 in Appendix A). For instance, one path involves exploiting an Apache remote-code-execution vulnerability (such as Log4j2 [10]) on the server side, or obtaining SSL certificates and keys to perform a man-in-the-middle (MITM [2]) attack on the HTTPS channel between the vehicle cloud and the TBOX. Then we can tamper with the firmware update and push it via the TBOX and gateway to the CDC-MCU.

PoC Attack. We validated this attack on Car C by discovering hardcoded CA certificates[6] and private keys in the TBOX, which allowed us to bypass TLS/SSL verification and execute a MITM attack on the connection between the cloud and the TBOX. As a result, we could inject malicious code into the OTA firmware. Furthermore, reverse-engineering the firmware's verification process revealed a logical flaw: the CDC would proceed with an upgrade even if the firmware's hash check failed. Exploiting this flaw, we successfully installed our modified firmware onto the CDC-MCU. This attack path has an overall feasibility of $Medium$, a potential impact classified as $Serious$, and a final risk level of Level 4.

Insights. This attack path demonstrates how OTA updates delivered through the vehicle cloud can serve as a conduit for injecting malicious code into a modern vehicle's core systems. Additional external components—such as smartphones (see Fig. 8d), satellites, or ground stations—also act as potential off-vehicle entry points for functional-level TARA, expanding the overall attack surface beyond in-vehicle networks. **The results show that DefenseWeaver can comprehensively cover peripheral devices and cloud services in its analysis, providing a more holistic view of potential attack vectors in modern vehicles.** However, it is difficult for the datalog-based approaches [27], [51] to capture all of these off-vehicle attack surfaces due to their static library-based design.

4) PKES: Discovering Unforeseen Attack Surfaces: Passive Keyless Entry and Start (PKES) systems provide convenient vehicle access by using radio-frequency signals to unlock and start the car. Nevertheless, they are susceptible to relay attacks, wherein attackers trick the system into believing the key fob is nearby, enabling unauthorized entry or ignition. To mitigate such threats, the latest PKES implementations (Fig. 8d) integrate Ultra-Wideband (UWB) technology according to the CCC protocol. UWB offers a "secure ranging" feature that effectively counters standard relay attacks [21]. Even so, advanced PKES technology can still present unforeseen attack surfaces. Using DefenseWeaver, we simply added a software annotation indicating that "the latest PKES system adds UWB modules to prevent relay attacks." We then configured a threat scenario (attack entry at the user's phone and attack endpoint

at the door motor) under the objective "illegally open the car door." The resulting attack tree (Fig. 13 in Appendix A) reveals that bypassing PKES involves two core steps: first, intercepting UWB signals and injecting malicious ones to disrupt the normal UWB ranging process; second, using a relay attack against the system's Bluetooth Low Energy (BLE) channel. Together, these actions deceive the PKES into maintaining outdated distance data, enabling an unauthorized door unlock once the authorized user moves away.

PoC Attack. We validated DefenseWeaver's results on Car D. After sniffing both the UWB and BLE signals, we identified system parameters like connection intervals and window offsets on BLE. By continually interfering with UWB ranging while the legitimate user was present, the PKES retained the user's proximity data even after the user left. We then relayed the BLE signals to complete the unlock procedure. In this scenario, the system relied on outdated UWB measurements, thus erroneously concluding that the user was still nearby. This attack path has an overall feasibility of $High$, a potential impact classified as $Serious$, and a final risk level of Level 5.

Insights. This attack path illustrates how UWB—though designed to bolster PKES security—can itself become an attack surface when coupled with relay attacks on other channels. Moreover, **DefenseWeaver demonstrates its ability to synthesize various methods (BLE, UWB manipulation, and relay attacks) into coherent, novel attack paths.** Such comprehensive analyses underscore the importance of integrating both traditional and emerging communication standards in function-level TARA. In comparison, the datalog-based approaches [27], [51] can only reason the known attack surfaces based on predefined threat libraries.

5) Responsible Disclosure and Summary: All penetration tests were conducted jointly with the vehicle manufacturers under ethical guidelines. In total, we identified 11 distinct vulnerabilities/attack paths across Cars A–D, each of which was promptly reported to the responsible OEM or supplier and has since been patched. Due to confidentiality agreements, we omit specific manufacturer names and certain low-level details. These four cases cover a wide range of automotive technologies (multiple ECUs, wireless interfaces, OS software, etc.), and in each case DefenseWeaver discovered component-specific attack methods beyond the scope of existing threat libraries. Traditional approaches require analysts to manually select likely vulnerabilities from a database and write custom reasoning rules for each scenario, whereas DefenseWeaver automates the end-to-end process of attack tree generation and feasibility evaluation. The real-world results above confirm that DefenseWeaver can drastically reduce the manual effort while uncovering critical, non-intuitive attack paths in complex vehicle systems.

D. Case Studies on Other Electronic Systems

Although our primary evaluation is in the automotive domain, DefenseWeaver's methodology is general. We performed two case studies on non-automotive cyber-physical systems to demonstrate cross-domain applicability.

1) Systems Under Examination: Leveraging its broad knowledge base, DefenseWeaver can generate attack trees for diverse embedded systems (without any code modifications). We applied it to two distinct platforms: (i) a PX4-based unmanned aerial vehicle (UAV) system with an inertial measurement unit (IMU) and GPS sensors, and (ii) an Electronic Chart Display and Information System (ECDIS) used in ship navigation (responsible for map display, route planning, etc.). In these case studies, we did not perform live penetration tests; instead, we built system models from publicly available documentation [40], [9] and prior research [63], [53], [54]. We then compared DefenseWeaver's generated attack paths to known vulnerabilities reported in the literature. The complete attack trees for the UAV and ECDIS are provided in Appendix B (Fig. 14 and Fig. 15, respectively). **These experiments demonstrate that DefenseWeaver's approach to function-level TARA can be easily extended to complex, safety-critical environments beyond automotive.**

2) Unmanned Aerial Vehicle (UAV): Drones require thorough threat assessments, as mandated by national and international safety standards (e.g., GB 42590-2023 [8]). These standards span the entire drone lifecycle, from data-link protection to electromagnetic compatibility. A recent work [63] manually identified 2 novel multi-round attack paths to degrade drone's sensor reliability over time. By applying DefenseWeaver to a PX4-based UAV (Fig. 8e), we identified not only these two reported 2 multi-round attack paths in [63], but also 1 additional attack path against drone clusters. (i) Electromagnetic Interference on the IMU. Broadcasting interference signals at frequencies matching the MPU6000 hardware chip can distort gyroscope outputs. The resulting unstable flight dynamics cause blurred images that lead to target misclassification. (ii) Forged GPS Signals. Emitting counterfeit GPS data disrupts accurate positioning, again producing blurred or misplaced images and undermining flight autonomy. (iii) Swarm-Level Manipulation for Drone Clusters. Exploiting the communication channels between drones in a swarm allows attackers to tamper with the collective positioning signals. This can delay or misroute multiple drones, potentially compromising time-critical missions like search and rescue. Though this attack path is not covered and validated in [63], it is well discussed in [55]. The above 3 diverse paths highlight how DefenseWeaver takes hardware, software, and operational context into account—ultimately revealing new drone-specific vulnerabilities (e.g., swarm communication) not covered in earlier automotive studies.

3) ECDIS of Ships: Ships also face stringent cybersecurity requirements. For instance, the UR E27 Rev.1 standard [12] explicitly mandates risk assessments for on-board systems and equipment. Existing works [53], [54] show that ECDIS platforms often contain high-risk vulnerabilities such as (i) outdated services (e.g., SMB or RDP) lacking authentication and (ii) third-party software (e.g., web servers) prone to N-day exploits such as Log4j2 or CVE-2021-41773. By applying
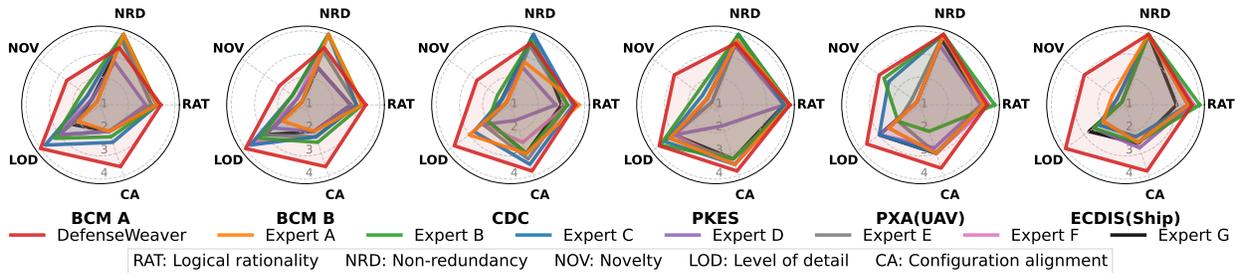
Fig. 9: Comparison of DefenseWeaver and seven human experts in different dimensions.

DefenseWeaver to a typical ECDIS system (Fig. 8f), we also identified the 2 known threats and 1 additional risk factor: (i) Exploiting Remote Desktop Services. Vulnerabilities like CVE-2019-0708 enable unauthorized remote access, potentially leading to arbitrary code execution. (ii) Leveraging Third-Party Software Flaws. Known Apache server exploits allow attackers to exfiltrate data or upload malicious code. (iii) Malicious ENC Files and USB Interfaces. By injecting counterfeit electronic navigational chart (ENC) files through the system's USB update port, an attacker can alter critical route data. This tactic not only exploits a physical interface but also poses immediate risks to maritime navigation safety. Although this attack path is not covered and validated in [53], [54], it is involved in the ECDIS cybersecurity guidelines [7].

E. Comparison with Human Experts

While the above experiments show DefenseWeaver guiding the discovery of real attack paths, we also quantitatively evaluated the quality of its attack trees against those designed by human experts. We assembled a seven-member review team of security professionals (with backgrounds in automotive TARA, penetration testing, and cybersecurity management; see Appendix C, Table II) to score attack trees generated by DefenseWeaver versus those created by human experts. Each human expert was required to spend enough time to get familiar with system configurations for the six scenarios we considered (the four automotive components and the two additional case-study systems) before they independently created attack trees. The review team assessed all attack trees on five key dimensions of quality – logical rationality, non-redundancy, novelty, level of detail, and configuration alignment – following ISO/SAE 21434 guidelines and industry best practices. Each dimension was rated on a 1–4 scale (higher is better). DefenseWeaver's trees were rated by all seven reviewers, whereas each expert-designed tree was scored only by the other six reviewers (preventing authors from rating their own work). Scores for each dimension were averaged across reviewers, and we visualized overall performance using five-dimensional radar charts (Fig. 9). Please refer to Appendix C for further details on the expert recruitment, scoring criteria, and evaluation process.

Results and Analysis: We identify three major limitations in the human-crafted attack trees that DefenseWeaver was able to overcome: (i) Human experts struggled to adapt to new system configurations, often failing to consider unconventional attacks outside their experience. (ii) Some expert-designed trees introduced superfluous or incorrect elements (e.g., non-existent components) based on subjective assumptions. (3) Experts sometimes overlooked critical yet subtle system-specific differences (for instance, they treated the two BCM cases too similarly, missing Car B's UART vector). By avoiding these issues, DefenseWeaver's trees exhibited significantly higher novelty and system alignment. On average, as shown in Table I, the DefenseWeaver-generated trees scored +105.00% higher in novelty (identifying many more unconventional attack paths) and +43.68% higher in system alignment (strictly mapping to actual system components and interfaces) than the human experts. DefenseWeaver's trees also contained far more detail in attack steps (+41.46% on the detail dimension), providing granular step-by-step paths. The logical rationality of DefenseWeaver's attack trees was on par with experts (a slight +5.79% gain in rationality score). Notably, DefenseWeaver's comprehensive approach led to only a 0.90% increase in redundancy, indicating it did not suffer from excessive duplicate paths despite its thoroughness. Overall, in all six scenarios, DefenseWeaver's automatically generated attack trees achieved equal or higher scores than the human-crafted trees in every dimension. Notably, the time consuming of DefenseWeaver for each case is only 0.43min, which is significantly reduced by 98.8% compared with that of human experts. **These results indicate that DefenseWeaver can produce attack trees of substantially higher quality than traditional expert-driven methods, offering more complete coverage of potential threats without sacrificing coherence or correctness.**

VI. LIMITATIONS AND FUTURE WORK

While DefenseWeaver already automates the generation and assessment of attack paths, it still relies on user-provided threat

TABLE I: Overall Comparison with Human Experts (Average)

	RAT[1]	NRD[2]	NOV[3]	LOD[4]	CA[5]	Time
DW[6]	3.62	3.67	2.73	3.88	3.71	0.43m
Expert	3.43	3.70	1.33	2.74	2.58	36.12m

[1] RAT: Logical rationality [2] NRD: Non-redundancy [3] NOV: Novelty
[4] LOD: Level of detail [5] CA: Configuration alignment [6] DW: DefenseWeaver

scenarios, which are high-level concepts that do not account for component-specific details. Therefore, we envision integrating Microsoft's STRIDE model [42]—covering Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege—to automatically derive more comprehensive sets of threat scenarios. This enhancement would further reduce manual intervention and streamline end-to-end TARA processes.

VII. CONCLUSION

We introduced DefenseWeaver, a novel system that automates function-level TARA by leveraging LLMs. Unlike existing methods bound by static threat libraries, DefenseWeaver adapts to evolving vulnerabilities due to the adoption of LLMs, offering flexibility across diverse standards and platforms. In extensive evaluations on four real automotive security projects, DefenseWeaver uncovered 11 practical attack paths, each validated via penetration testing and responsibly disclosed. We also deploy DefenseWeaver in UAV and ECDIS systems to demonstrate its cross-domain applicability, revealing new attack surfaces beyond traditional automotive contexts. Integrated into commercial cybersecurity management platforms, DefenseWeaver has produced more than 8,200 attack trees in the industry to date. Compared with human experts, DefenseWeaver significantly reduced time consumption of TARA process by 98.8%. Overall, DefenseWeaver provides a robust, adaptive approach to TARA, significantly advancing the state of practice in automotive cybersecurity and beyond.

REFERENCES

[1] ABDELKADER, G., ELGAZZAR, K., AND KHAMIS, A. Connected vehicles: Technology review, state of the art, challenges and opportunities. *Sensors 21*, 22 (2021), 7712.

[2] ACCESSED DEC, O. L. Man-in-the-middle attack, https://en.wikipedia.org/wiki/Man-in-the-middle_attack, 2024.

[3] ACCESSED JAN, O. L. OpenXSAM, https://github.com/ASRG/openXSAM, 2023.

[4] ACCESSED JAN, O. L. DefenseWeave, https://github.com/0xYYQ/DefenseWeaver, 2025.

[5] ACCESSED JAN, O. L. DefenceWeaver TARA tool demo, https://youtu.be/aEAuLp2BaIU, 2025.

[6] ACCESSED JAN, O. L. Certificate authority, https://en.wikipedia.org/wiki/Certificate_authority, 2025.

[7] ACCESSED JUN, O. L. IHO ENC & ECDIS Cyber Security Guideline, https://iho.int/uploads/user/Services%20and%20Standards/ENCWG/ENCWG7/ENCWG7-4.5_2022_EN_ECDIS%20cyber%20security%20guideline%20draft.pdf, 2017.

[8] ACCESSED MAY, O. L. Gb 42590-2023, 2023.

[9] ACCESSED NOV, O. L. PX4 Architectural Overview, https://docs.px4.io/main/en/concept/architecture.html, 2024.

[10] ACCESSED OCT, O. L. CApache Log4j Security Vulnerabilities, https://logging-log4j.staged.apache.org/log4j/2.x/security.html, 2023.

[11] ACCESSED SEP, O. L. cve-2016-6309, https://nvd.nist.gov/vuln/detail/cve-2016-6309, 2016.

[12] ACCESSED SEP, O. L. Ur e27 rev.1, 2023.

[13] ACHIAM, J., ADLER, S., AGARWAL, S., AHMAD, L., AKKAYA, I., ALEMAN, F. L., ALMEIDA, D., ALTENSCHMIDT, J., ALTMAN, S., ANADKAT, S., ET AL. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[14] ANTONY, M. M., AND WHENISH, R. Advanced driver assistance systems (adas). In *Automotive Embedded Systems: Key Technologies, Innovations, and Applications*. Springer, 2021, pp. 165–181.

[15] CHEN, J., XIAO, S., ZHANG, P., LUO, K., LIAN, D., AND LIU, Z. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation, 2024.

[16] CHLUP, S., CHRISTL, K., SCHMITTNER, C., SHAABAN, A. M., SCHAUER, S., AND LATZENHOFER, M. Threatget: Towards automated attack tree analysis for automotive cybersecurity. *Information 14*, 1 (2023).

[17] CHOWDHURY, T., LESIUTA, E., RIKLEY, K., LIN, C.-W., KANG, E., KIM, B., SHIRAISHI, S., LAWFORD, M., AND WASSYNG, A. Safe and secure automotive over-the-air updates. In *Computer Safety, Reliability, and Security: 37th International Conference, SAFECOMP 2018, Västerås, Sweden, September 19-21, 2018, Proceedings 37* (2018), Springer, pp. 172–187.

[18] DEVLIN, J., CHANG, M.-W., LEE, K., AND TOUTANOVA, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (Minneapolis, Minnesota, June 2019), J. Burstein, C. Doran, and T. Solorio, Eds., Association for Computational Linguistics, pp. 4171–4186.

[19] DONG, Y., JIANG, X., JIN, Z., AND LI, G. Self-collaboration code generation via chatgpt, 2024.

[20] FOR EUROPE, U. N. E. C. Un regulation no. 155 - cyber security and cyber security management system. https://unece.org/transport/documents/2021/03/standards/un-regulation-no-155-cyber-security-and-cyber-security, Accessed 2021-03-04.

[21] FRANCILLON, A., DANEV, B., AND CAPKUN, S. Relay attacks on passive keyless entry and start systems in modern cars. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)* (2011), Eidgenössische Technische Hochschule Zürich, Department of Computer Science.

[22] GAO, C., WANG, G., SHI, W., WANG, Z., AND CHEN, Y. Autonomous driving security: State of the art and challenges. *IEEE Internet of Things Journal 9*, 10 (2022), 7572–7595.

[23] HONG, S., ZHUGE, M., CHEN, J., ZHENG, X., CHENG, Y., ZHANG, C., WANG, J., WANG, Z., YAU, S. K. S., LIN, Z., ZHOU, L., RAN, C., XIAO, L., WU, C., AND SCHMIDHUBER, J. Metagpt: Meta programming for a multi-agent collaborative framework, 2024.

[24] INSTITUTE, E. T. S. Cyber; methods and protocols; part 1: Method and pro forma for threat, vulnerability, risk analysis (tvra). https://standards.globalspec.com/std/10259890/ts-102-165-1, Accessed 2017-10-01.

[25] INSTITUTE, E. T. S. Cyber; methods and protocols; part 1: Method and pro forma for threat, vulnerability, risk analysis (tvra). https://standards.globalspec.com/std/14488631/ts-102-165-1, Accessed 2022-01-01.

[26] ISLAM, M. M., LAUTENBACH, A., SANDBERG, C., AND OLOVSSON, T. A risk assessment framework for automotive embedded systems. In *Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security* (New York, NY, USA, 2016), CPSS '16, Association for Computing Machinery, p. 3–14.

[27] JING, P., CAI, Z., CAO, Y., YU, L., DU, Y., ZHANG, W., QIAN, C., LUO, X., NIE, S., AND WU, S. Revisiting automotive attack surfaces: a practitioners' perspective. In *2024 IEEE Symposium on Security and Privacy (SP)* (2024), pp. 2348–2365.

[28] KARRAY, K., DANGER, J.-L., GUILLEY, S., AND ABDE-LAZIZ ELAABID, M. *Attack Tree Construction and Its Application to the Connected Vehicle*. Springer International Publishing, Cham, 2018, pp. 175–190.

[29] KERN, M., LIU, B., BETANCOURT, V. P., AND BECKER, J. Model-based attack tree generation for cybersecurity risk-assessments in automotive. In *2021 IEEE International Symposium on Systems Engineering (ISSE)* (2021), pp. 1–7.

[30] KOIDE, T., FUKUSHI, N., NAKANO, H., AND CHIBA, D. Chatspamdetector: Leveraging large language models for effective phishing email detection, 2024.

[31] KONG, H.-K., HONG, M. K., AND KIM, T.-S. Security risk assessment framework for smart car using the attack tree analysis. *Journal of Ambient Intelligence and Humanized Computing 9*, 3 (Jun 2018), 531–551.

[32] KUMAR, D., ABUHASHEM, Y., AND DURUMERIC, Z. Watch your language: Investigating content moderation with large language models, 2024.

[33] LAUTENBACH, A., ALMGREN, M., AND OLOVSSON, T. Proposing heavens 2.0 – an automotive risk assessment model. In *Proceedings of the 5th ACM Computer Science in Cars Symposium* (New York, NY, USA, 2021), CSCS '21, Association for Computing Machinery.

[34] LEE, J., TANG, F., YE, P., ABBASI, F., HAY, P., AND DIVAKARAN, D. M. D-fence: A flexible, efficient, and comprehensive phishing email detection system. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)* (2021), pp. 578–597.

[35] LIN, Y., LIU, R., DIVAKARAN, D. M., NG, J. Y., CHAN, Q. Z., LU, Y., SI, Y., ZHANG, F., AND DONG, J. S. Phishpedia: A hybrid deep learning based approach to visually identify phishing webpages. In *30th USENIX Security Symposium (USENIX Security 21)* (Aug. 2021), USENIX Association, pp. 3793–3810.

[36] LIU, P., YUAN, W., FU, J., JIANG, Z., HAYASHI, H., AND NEUBIG, G. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys 55*, 9 (2023), 1–35.

[37] LIU, R., LIN, Y., YANG, X., NG, S. H., DIVAKARAN, D. M., AND DONG, J. S. Inferring phishing intention via webpage appearance and dynamics: A deep vision based approach. In *31st USENIX Security Symposium (USENIX Security 22)* (Boston, MA, Aug. 2022), USENIX Association, pp. 1633–1650.

[38] MACHER, G., SPORER, H., BERLACH, R., ARMENGAUD, E., AND KREINER, C. Sahara: A security-aware hazard and risk analysis method. In *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (2015), pp. 621–624.

[39] MANDI, Z., JAIN, S., AND SONG, S. Roco: Dialectic multi-robot collaboration with large language models, 2023.

[40] MEIER, L., HONEGGER, D., AND POLLEFEYS, M. Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *2015 IEEE International Conference on Robotics and Automation (ICRA)* (2015), pp. 6235–6240.

[41] MENG, R., MIRCHEV, M., BÖHME, M., AND ROYCHOUDHURY, A. Large language model guided protocol fuzzing. *Proceedings 2024 Network and Distributed System Security Symposium* (2024).

[42] MICROSOFT. Microsoft threat modeling tool threats. https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-threats, Accessed 2022-08-25.

[43] MING, X. text2vec: A tool for text to vector, 2022.

[44] NOEL, S., AND JAJODIA, S. Understanding complex network attack graphs through clustered adjacency matrices. In *21st Annual Computer Security Applications Conference (ACSAC'05)* (2005), pp. 10 pp.–169.

[45] OU, X., BOYER, W. F., AND MCQUEEN, M. A. A scalable approach to attack graph generation. In *Proceedings of the 13th ACM Conference on Computer and Communications Security* (New York, NY, USA, 2006), CCS '06, Association for Computing Machinery, p. 336–345.

[46] OU, X., GOVINDAVAJHALA, S., AND APPEL, A. W. MulVAL: A logic-based network security analyzer. In *14th USENIX Security Symposium (USENIX Security 05)* (Baltimore, MD, July 2005), USENIX Association.

[47] OU, X., AND SINGHAL, A. *Attack Graph Techniques.* Springer New York, New York, NY, 2011, pp. 5–8.

[48] PEARCE, H., TAN, B., AHMAD, B., KARRI, R., AND DOLAN-GAVITT, B. Examining zero-shot vulnerability repair with large language models. In *2023 IEEE Symposium on Security and Privacy (SP)* (2023), pp. 2339–2356.

[49] QIAN, C., LIU, W., LIU, H., CHEN, N., DANG, Y., LI, J., YANG, C., CHEN, W., SU, Y., CONG, X., XU, J., LI, D., LIU, Z., AND SUN, M. Chatdev: Communicative agents for software development, 2024.

[50] RUDDLE, A., WEYL, B., IDREES, S., ROUDIER, Y., FRIEDEWALD, M., LEIMBACH, T., FUCHS, A., GÜRGENS, S., HENNINGER, O., RIEKE, R., RITSCHER, M., BROBERG, H., APVRILLE, L., PACALET, R., AND PEDROZA, G. Security requirements for automotive on-board networks based on dark-side scenarios. deliverable d2.3: Evita. e-safety vehicle intrusion protected applications. *Fraunhofer ISI* (01 2009).

[51] SAULAIMAN, M. N.-E., KOZLOVSZKY, M., BANATI, A., AND CSILLING, A. Use cases of attack graph in threat analysis and risk assessment for the automotive domain. In *2022 IEEE 1st International Conference on Cognitive Mobility (CogMob)* (2022), pp. 000085–000092.

[52] SCHNEIER, B. Attack trees. *Dr. Dobb's journal 24*, 12 (1999), 21–29.

[53] SVILICIC, B., BRČIĆ, D., ŽUŠKIN, S., AND KALEBIĆ, D. Raising awareness on cyber security of ecdis. *TransNav: International Journal on Marine Navigation and Safety of Sea Transportation 13*, 1 (2019), 231–236.

[54] SVILICIC, B., KAMAHARA, J., CELIC, J., AND BOLMSTEN, J. Assessing ship cyber risks: A framework and case study of ecdis security. *WMU Journal of Maritime Affairs 18* (2019), 509–520.

[55] TAYYAB, M., MUMTAZ, M., MUZAMMAL, S. M., JHANJHI, N. Z., ET AL. Swarm security: Tackling threats in the age of drone swarms. In *Cybersecurity Issues and Challenges in the Drone Industry.* IGI Global, 2024, pp. 324–342.

[56] THE INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. Iso/sae 21434:2021 road vehicles — cybersecurity engineering. https://www.iso.org/standard/70918.html, Accessed 2021-08.

[57] THOMPSON, M. Uds security access for constrained ecus. Tech. rep., SAE Technical Paper, 2022.

[58] VASWANI, A. Attention is all you need. *Advances in Neural Information Processing Systems* (2017).

[59] WANG, C., ZHANG, J., FENG, Y., LI, T., SUN, W., LIU, Y., AND PENG, X. Teaching code llms to use autocompletion tools in repository-level code generation, 2024.

[60] WEI, Y., WANG, Z., LU, Y., XU, C., LIU, C., ZHAO, H., CHEN, S., AND WANG, Y. Editable scene simulation for autonomous driving via collaborative llm-agents. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2024), pp. 15077–15087.

[61] WU, S., XIE, J., CHEN, J., ZHU, T., ZHANG, K., AND XIAO, Y. How easily do irrelevant inputs skew the responses of large language models?, 2024.

[62] XIA, C. S., PALTENGHI, M., LE TIAN, J., PRADEL, M., AND ZHANG, L. Fuzz4all: Universal fuzzing with large language models. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering* (New York, NY, USA, 2024), ICSE '24, Association for Computing Machinery.

[63] XU, Y., HAN, X., DENG, G., LI, J., LIU, Y., AND ZHANG, T. Sok: Rethinking sensor spoofing attacks against robotic vehicles from a systematic view. In *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)* (2023), IEEE, pp. 1082–1100.

[64] XU, Z., YU, C., FANG, F., WANG, Y., AND WU, Y. Language agents with reinforcement learning for strategic play in the werewolf game, 2024.

[65] ZHANG, Y., RUAN, H., FAN, Z., AND ROYCHOUDHURY, A. Autocoderover: Autonomous program improvement. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis* (New York, NY, USA, 2024), ISSTA 2024, Association for Computing Machinery, p. 1592–1604.

[66] ZHOU, X., ZHANG, T., AND LO, D. Large language model for vulnerability detection: Emerging results and future directions. In *Proceedings of the 2024 ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results* (New York, NY, USA, 2024), ICSE-NIER'24, Association for Computing Machinery, p. 47–51.

Experimental Methodology

Each attack tree comprises a **threat scenario** (root node), **logical nodes** (AND, OR), **attack objectives** ($AO - X$), and **attack methods** ($AM - X$). For each method, DefenseWeaver performs step-by-step feasibility assessments ($Step\_F$) and calculates a cumulative feasibility score ($Cumulative\_F$). The threat scenario denotes the ultimate goal of an attack, while the attack objectives serve as intermediate goals that support it. Logical nodes connect the attack methods necessary for achieving each objective. Using DefenseWeaver, we generated four comprehensive attack trees for the four threat scenarios discussed in Sec V-C (see Fig 10, 11, 12, 13). In these figures, we highlight in red the specific attack paths detailed in Sec V-C, annotated with feasibility indicators ($Step\_F$ and $Cumulative\_F$). Guided by these four trees, each showing at least a "Medium" feasibility rating and a relatively high risk level (at least 3), we identified 11 practical attack paths confirmed through penetration tests. Note that the attack tree serves as a high-level guide and may not exactly mirror the real-world attack paths uncovered in full detail. The 11 verified attack paths are as follows:

Complete Attack Tree of BCM (Car A)

Drawing on Fig 10, we identified three feasible paths. For this attack tree, the threat scenario's overall attack feasibility is $Medium$, the potential impact is $Major$, and the resulting risk level is 3:

Attack Path 1: By dumping the BCM firmware via the MCU's JTAG port, we reverse-engineered the seed-key conversion algorithm for UDS SecurityAccess Service to reconstruct the complete authentication mechanism. After compromising the IVI, we sent malicious TCP packets to the Gateway to trigger the CVE-2016-6309 vulnerability in OpenSSL and obtain a reverse shell. Within this shell, we send malicious CAN messages—ultimately tampering with the BCM firmware once the UDS authentication was bypassed.

Attack Path 2: We demodulated TPMS data using FSK and cracked a non-standard CRC8 algorithm to forge TPMS signals. Replaying these signals triggered a buffer overflow in the BCM-MCU component.

Attack Path 3: We successfully detected the JTAG interface through hardware reverse engineering, including TDI, TDO, TMS, and TCK, and established a connection. However, when attempting to read the firmware further, we discovered that it has built-in read and write protection.s.

Complete Attack Tree of BCM (Car B)

Similarly, based on Fig 11, we discovered three paths. The threat scenario's overall attack feasibility is $High$, the potential impact is $Major$, and the final risk level is 4:

Attack Path 4: As in Car A, we identified hard-coded UDS authentication in the BCM firmware. Through the Gateway's UART interface, we exploited CVE-2023-0179 in Linux 6.1 to compromise the Gateway and then tampered with the BCM firmware via the UDS protocol.

Attack Path 5: We launched a DoS attack by using high-power signals to attack the radio receiver module, thereby damaging its gain module.

Attack Path 6: Like Path 2, we can successfully connect to the JTAG interface, but due to the read protection, we are unable to read the firmware.

Complete Attack Tree of CDC (Car C)

Based on Fig 12, we identified three paths. The threat scenario's overall attack feasibility is $Medium$, the potential impact is $Serious$, and the final risk level is 4:

Attack Path 7: In the T-BOX, we discovered hard-coded CA certificates and private keys, which allowed us to bypass the TLS/SSL verification and conduct a man-in-the-middle attack between the Cloud and the T-BOX. This enabled us to inject malicious code into the firmware upgrade package. Additionally, by reverse-engineering the OTA firmware package verification process in the firmware, we found a logical flaw where the CDC would proceed with the upgrade regardless of whether the firmware package hash was correct. This allowed us to successfully run the modified firmware on the CDC-MCU.

Attack Path 8: The CDC-MCU can be upgraded via a USB drive. By reverse-engineering the firmware, we identified a logical flaw where the CDC would proceed with the upgrade regardless of whether the firmware package hash was correct. We created a malicious firmware package on a USB drive and successfully ran the modified firmware on the CDC-MCU through the USB port.

Attack Path 9: We used a timing side-channel attack to brute-force the ADB access password for the IVI system. We then controlled the IVI to issue an illegal firmware rollback request. Due to a design flaw, the old firmware was updated to the CDC-MCU via the OTA function.

Complete Attack Tree of PKES (Car D)

Lastly, guided by Fig 13, we identified two feasible paths. The overall attack feasibility is $High$, the potential impact is $Serious$, and the final risk level is 5:

Attack Path 10: We first sniffed the normal UWB (Ultra-Wideband) signals to analyze the physical layer structure in use, and sniffed BLE signals to confirm the exchanged information such as MAC, UUID. Subsequently, we continuously transmitted malicious UWB signals to disrupt the UWB ranging process. We discovered that when UWB ranging fails consecutively for a certain period, the car still uses the old distance data from before the interference. Therefore, as long as we interfere with the UWB signals while the car owner is still near the car, and then relay the BLE signals after the owner has moved away, the UWB ranging data will still reflect the old distance data from when the owner was near the car. The PKES still near the car and erroneously unlock the doors automatically.

Attack Path 11: We used a custom-made RFID relay device to receive the Select AID data transmitted by the car and relay it to another relay device positioned near the car's NFC

card. The NFC card would respond with an encrypted reply, which the relay device would then forward to the vehicle's card reader, thereby unlocking the car door.

APPENDIX B: COMPLETE ATTACK TREES OF CASE STUDIES

We also generated two complete attack trees based on previous research and publicly available information (Fig 14 and Fig 15), highlighting high-feasibility paths detailed in Sec V-D in red. Figure 14 encompasses most UAV-related attack paths from [63], whereas Fig 15 includes all high-risk vulnerabilities referenced in [53].

Complete Attack Tree of UAV

Fig 14 reveals three highly feasible paths, with an overall attack feasibility of $High$, a potential impact of $Major$, and a risk level of 4:

Attack Path 1: By transmitting electromagnetic interference signals at specific frequencies, the normal operation of the MPU60001.0 hardware chip is disrupted, causing distortion in the gyroscope signals it outputs. This leads to drone instability and subsequent image processing errors.

Attack Path 2: By sending spoofed GPS signals to interfere with the drone's positioning system, the drone receives incorrect geographical information, causing its camera to fail to focus or accurately target objects, resulting in blurred images that disrupt the drone's functionality.

Attack Path 3: By tampering with the positioning signals transmitted between drones, the swarm algorithm of the entire drone fleet is disrupted, causing delays in reaching the target location and affecting the timeliness of drone rescue missions.

Complete Attack Tree of ECDIS

Likewise, Fig 15 guides three highly feasible paths, with an overall attack feasibility of $High$, a potential impact of $Major$, and a final risk level of 4:

Attack Path 1: Attackers can exploit vulnerabilities in the RDP service, such as CVE-2019-0708, to gain unauthorized remote desktop access and execute malicious code on the ECDIS system.

Attack Path 2: By exploiting known vulnerabilities in the Apache web server, such as directory traversal or remote code execution vulnerabilities, malicious code can be uploaded to the server.

Attack Path 3: By updating incorrect ENC files through the USB interface of the ECDIS system, the navigation data is tampered with, causing the vessel to deviate from its intended route.

APPENDIX C: COMPARISON WITH HUMAN EXPERTS

1) Study Setup: To scientifically and objectively assess whether the attack trees designed by DefenseWeaver are superior to those designed by human experts, we conducted a three-step scoring process: establishing scoring criteria, designing attack trees by experts, and scoring by experts.

TABLE II: Seven-member review team

ID	Exp[1]	Co.[2]	Position[3]	Task[4]
Expert A	5	C3	TARA, Test	Dsgn, Scr
Expert B	3	C3	TARA, Test	Dsgn, Scr
Expert C	10	C3	TARA, Test	Crt, Dsgn, Scr
Expert D	6	C1	TARA, Manag	Crt, Dsgn, Scr
Expert E	5	C2	TARA, Manag	Crt, Dsgn, Scr
Expert F	12	C4	TARA, Manag, Reg	Crt, Dsgn, Scr
Expert G	5	C4	TARA, Manag, Reg	Crt, Dsgn, Scr

[1] Years of working experience in security;
[2] C1, C2: 1st party vehicle manufacturer,C3: 3rd party supplier; C4: TARA assessment agency.
[3] TARA: Threat Analysis and Risk Assessment; Manag: Project manager; Reg: Regulation-related study; Test: Security testing.
[4] Crt: establishing scoring criteria; Dsgn: designing Attack Trees; Scr: conducting scoring .

Recruitment. To ensure the professionalism and objectivity of the results, we invited experts from the automotive cybersecurity field of first-tier automotive manufacturers (2 persons), third-party suppliers (3 persons), and TARA ssessment agency (2 persons) from multiple countries (China and Germany) to form a seven-member review team, with their information presented in Table II. On average, the team members have about 6 years of experience in the security field, with Expert c and Expert f having over 10 years of experience. Their positions include TARA, security testing, project management, and regulation study. Tasks were assigned based on their positions and interviews (5 persons establishing the scoring criteria, all experts designing the attack trees, and conducting scoring), ensuring that all participants have sufficient experience to competently perform their tasks.

Procedure. To ensure a thorough and unbiased evaluation of the attack trees, we assembled a seven-member review team to conduct a comprehensive scoring analysis. The team assessed the attack trees designed by DefenseWeaver and those created by seven experts (each expert contributed six attack trees targeting four automotive components and two other electronic systems). Experts selected five key scoring dimensions—rationality, non-redundancy, novelty, level of detail, and configuration alignment—based on ISO/SAE 21434 to capture different aspects of attack tree quality. Each dimension was scored on a scale of 1 to 4 points, allowing for a nuanced comparison (Table III). To maintain fairness and minimize potential bias, we implemented a structured scoring process: (i) DefenseWeaver's attack trees were evaluated by all seven reviewers to ensure broad consensus. (ii) The experts' attack trees were scored only by individuals not involved in their creation, preventing author-related biases. For consistency, the score for each dimension was calculated as the average based on the total points and the number of reviewers. To visually represent the overall performance differences, we plotted the five-dimensional scores into radar charts (Fig. 9) and compared their areas. This approach allowed us to objectively quantify the comprehensive superiority of the attack trees, rather than relying on subjective judgments.

TABLE III: Overall Comparison with Human Experts

System	Dimension	DefenseWeaver	Expert	Improv.
	Rationality	3.40	3.10	↑ 9.68%
	Non-redundancy	3.40	3.70	↓ 8.11%
BCM_A	Novelty	2.60	1.30	↑ 100.00%
	Level of detail	4.00	3.00	↑ 33.33%
	System align.	3.60	2.15	↑ 67.44%
	Rationality	3.40	3.05	↑ 11.48%
	Non-redundancy	3.40	3.55	↓ 4.23%
BCM_B	Novelty	2.20	1.25	↑ 76.00%
	Level of detail	4.00	2.90	↑ 37.93%
	System align.	3.60	2.20	↑ 63.64%
	Rationality	3.60	3.30	↑ 9.09%
	Non-redundancy	3.60	3.30	↑ 9.09%
CDC	Novelty	2.60	1.20	↑ 116.67%
	Level of detail	3.80	2.45	↑ 55.10%
	System align.	3.80	2.85	↑ 33.33%
	Rationality	4.00	3.90	↑ 2.56%
	Non-redundancy	3.60	3.75	↓ 4.00%
PKES	Novelty	3.00	1.35	↑ 122.22%
	Level of detail	3.80	3.35	↑ 13.43%
	System align.	3.80	3.05	↑ 24.59%
	Rationality	3.67	3.58	↑ 2.51%
	Non-redundancy	4.00	3.90	↑ 2.56%
UAV	Novelty	3.00	1.70	↑ 76.47%
	Level of detail	3.67	2.50	↑ 46.80%
	System align.	3.67	2.75	↑ 33.45%
	Rationality	3.67	3.62	↑ 1.38%
	Non-redundancy	4.00	4.00	-
Ship	Novelty	3.00	1.20	↑ 150.00%
	Level of detail	4.00	2.25	↑ 77.78%
	System align.	3.80	2.50	↑ 52.00%
	Rationality	3.62	3.43	↑ 5.79%
	Non-redundancy	3.67	3.70	↓ 0.90%
Overall	Novelty	2.73	1.33	↑ 105.00%
	Level of detail	3.88	2.74	↑ 41.46%
	System align	3.71	2.58	↑ 43.68%

TABLE IV: Attack Tree Scoring Criteria

Dimension	Score	Level	Description
Logical rationality	1	Poor	All paths are considered unreasonable
	2	Limited	Only very few paths are reasonable
	3	Good	Most paths are reasonable with some exceptions
	4	Excellent	Nearly all paths are reasonable
Non-redundancy	1	Poor	All reasonable paths are completely repetitive
	2	Limited	Majority of paths are redundant
	3	Good	Few paths are repetitive
	4	Excellent	All paths are unique
Configuration alignment	1	Poor	Uses substantial irrelevant information
	2	Limited	Uses some relevant information mixed with irrelevant data
	3	Good	Uses 60% of system modeling information
	4	Excellent	Uses 80% of system modeling information
Level of detail	1	Poor	Incomprehensible to users
	2	Limited	Specifies target objects (e.g., attacking TBOX)
	3	Good	Specifies objects with attack techniques (e.g., MITM on TBOX's WiFi)
	4	Excellent	Includes objects, techniques, and examples (e.g., MITM on TBOX's WiFi using CVE-XXXX)
Novelty	1	Poor	No new attack methods provided
	2	Limited	1-2 unexpected attack methods
	3	Good	2-3 unexpected attack methods
	4	Excellent	More than 3 unexpected attack methods

2) Scoring Criterion: Experts have selected five core evaluation dimensions (Table IV)—logical rationality, non-redundancy, configuration alignment, level of detail, and novelty—based on ISO/SAE 21434 and industry white papers to comprehensively assess the quality of attack trees(as shown in Table IV). The rationale for each dimension is as follows: (i) **Logical Rationality** : ISO/SAE 21434 requires each attack path should be logically sound and follow security reasoning (no implausible leaps). This includes ensuring that the local attack objectives are clearly defined and realistic, the attack methods are technically feasible and relevant to the context, the logical nodes accurately represent the steps and relationships in the attack process, the feasibility and impact ratings are reasonable, and the overall attack path is consistent with the defined threat scenarios; (ii) **Non-redundancy**: According to TARA optimization principles, the attack tree should avoid duplicate or redundant paths, focusing on unique attack vectors and avoiding analytical redundancy aligns with the minimal attack tree criterion; (iii) **Configuration Alignment**: ISO/SAE 21434 mandate that attack tree nodes strictly correspond to system modeling elements (e.g., ECUs, communication protocols), ensuring TARA's practical applicability; (iv) **Level of Detail**: the attack steps should be described with sufficient technical detail and specificity (e.g., including concrete attack techniques or CVE examples); (v) **Novelty**: While ensuring compliance, the attack tree should include creative attack methods beyond well-known threat templates to enhance the comprehensiveness of threat coverage. Each dimension was scored on a four-point scale from Poor (1) to Excellent (4). The final score for each dimension is the average of all reviewers' ratings for that criterion on a given tree.
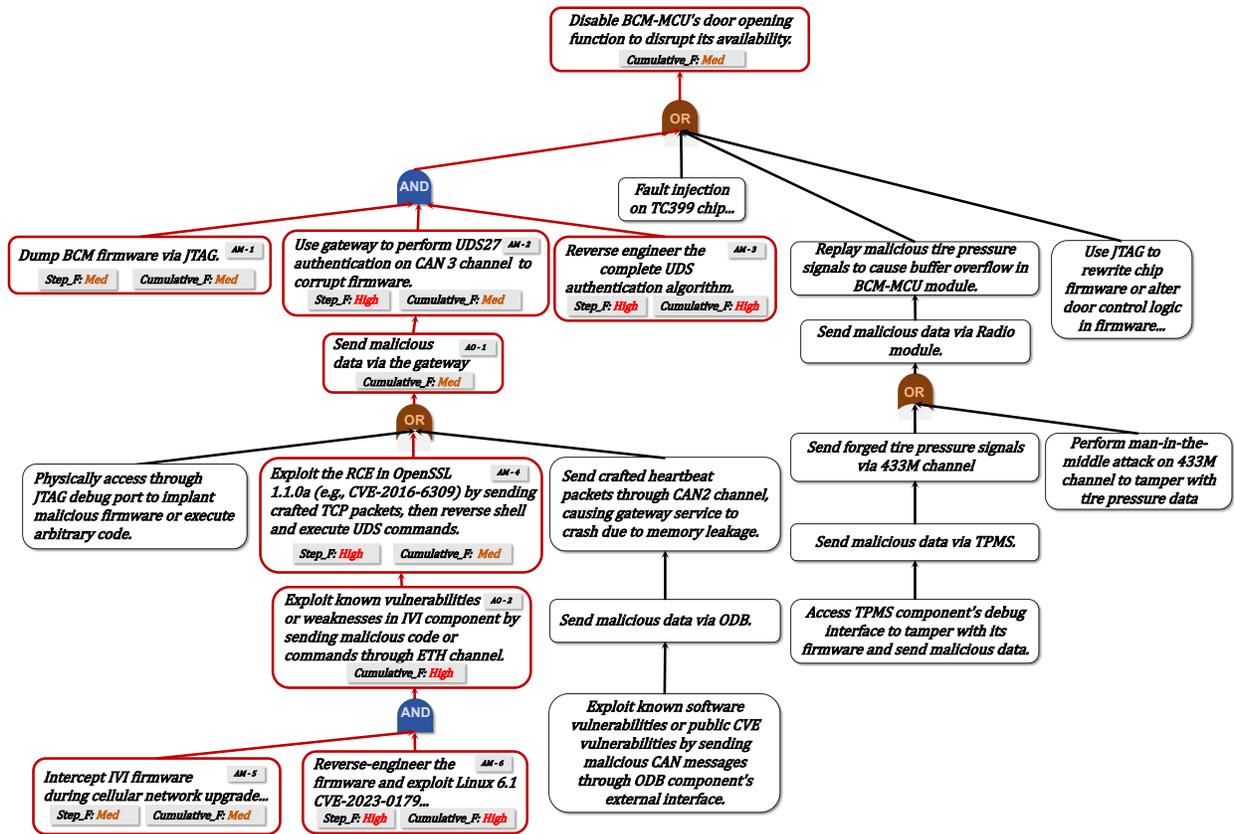
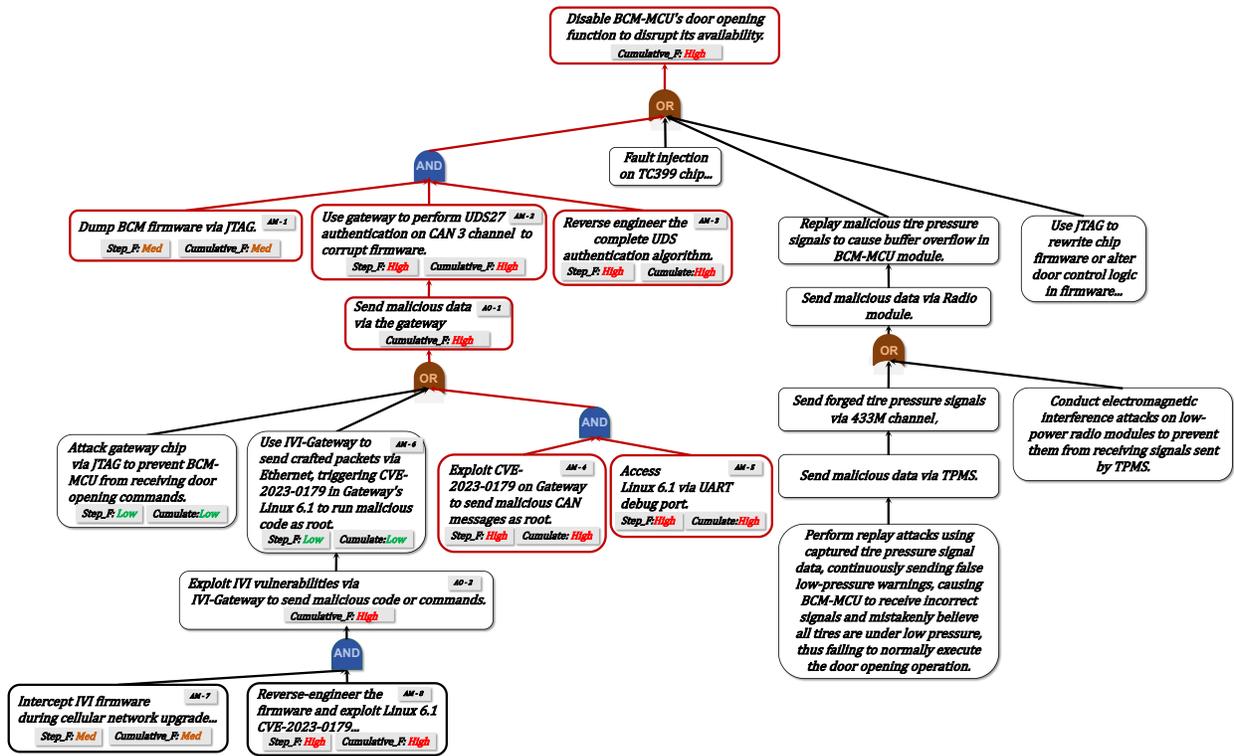Fig. 10: Attack tree BCM (Car A)

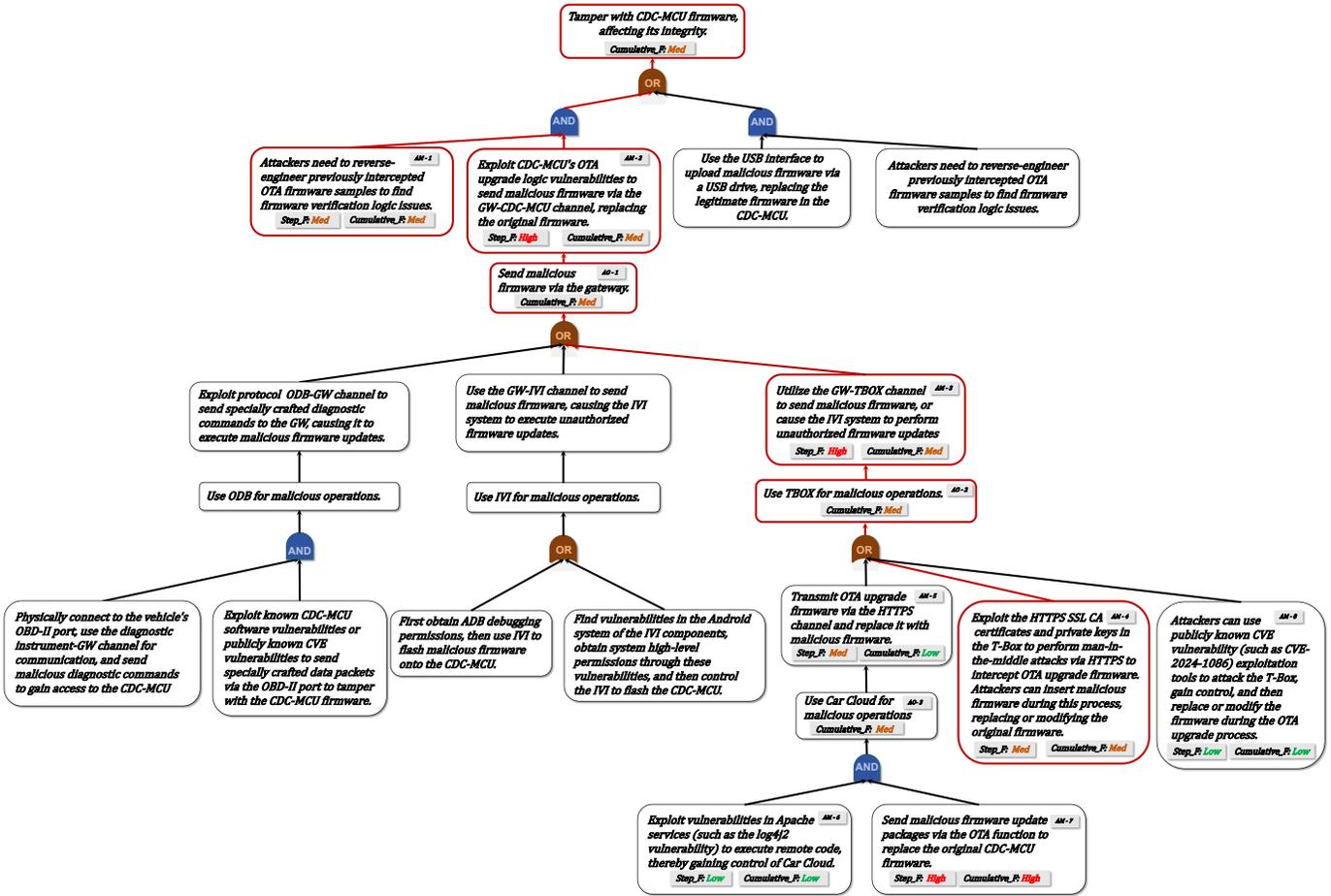Fig. 11: Attack tree BCM (Car B)
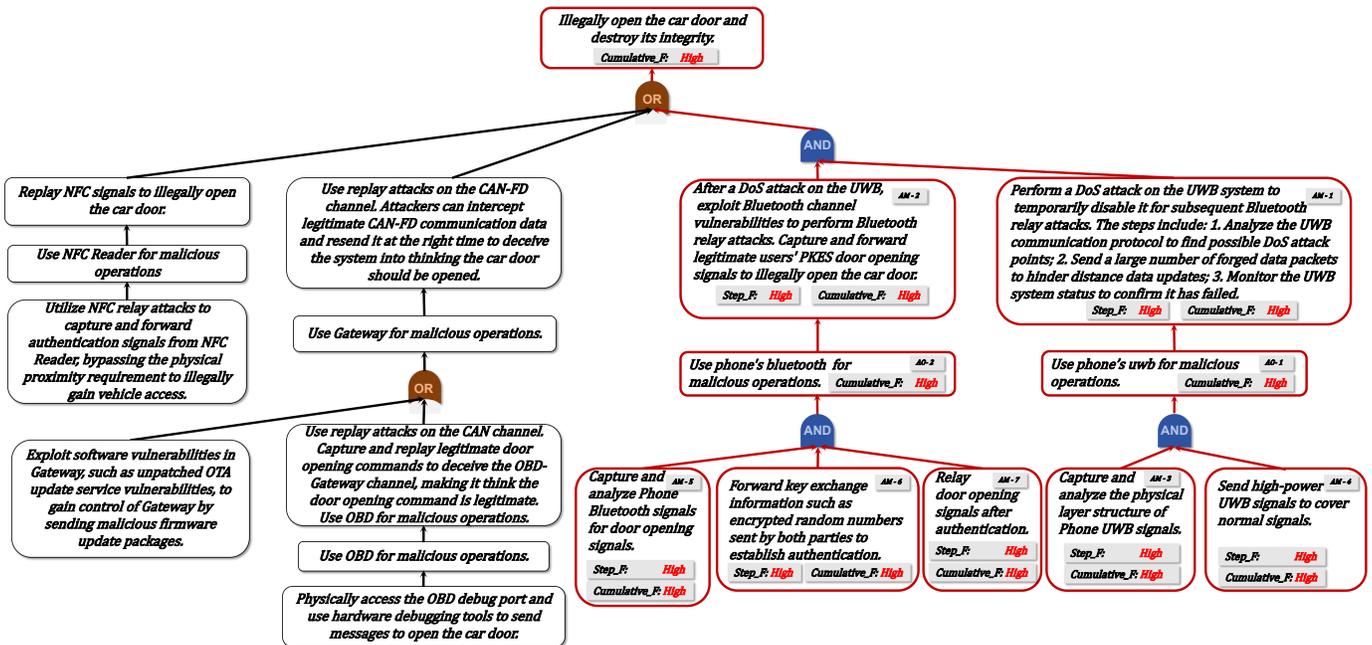
Fig. 12: Attack tree CDC (Car C)
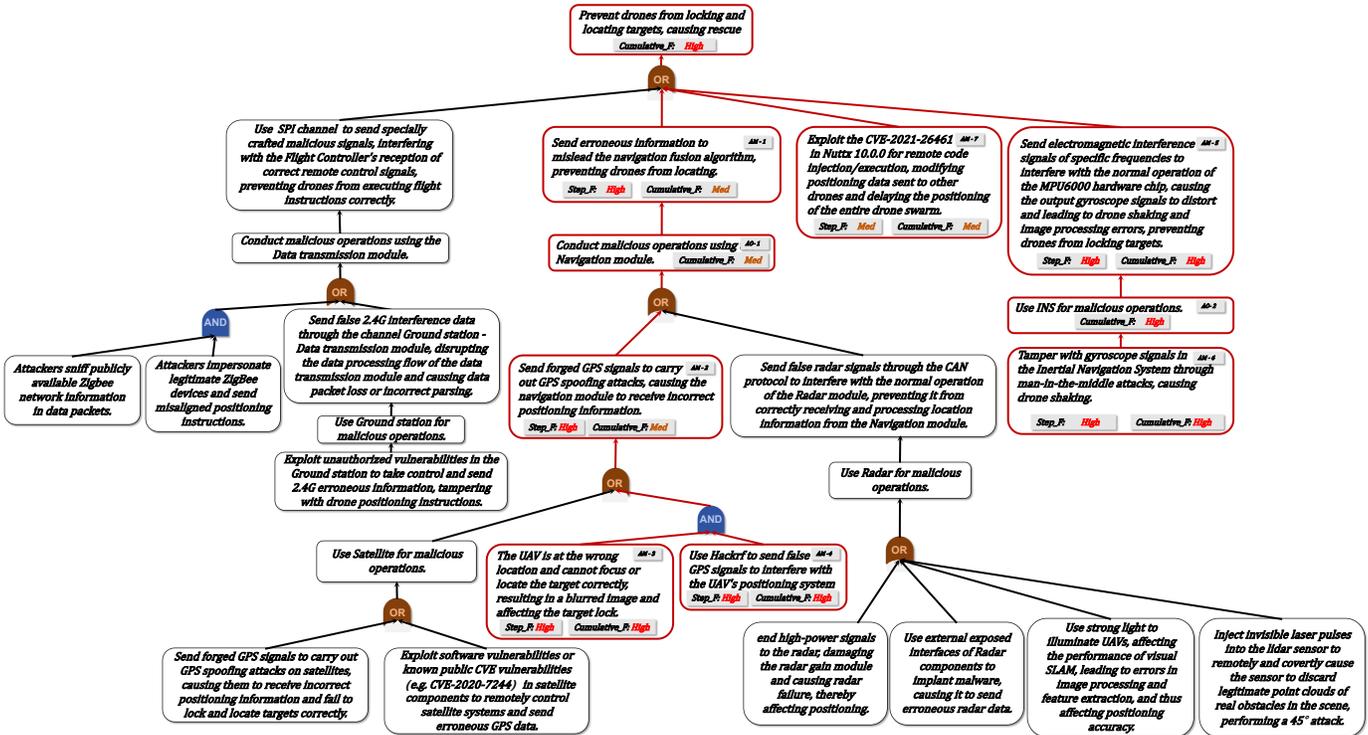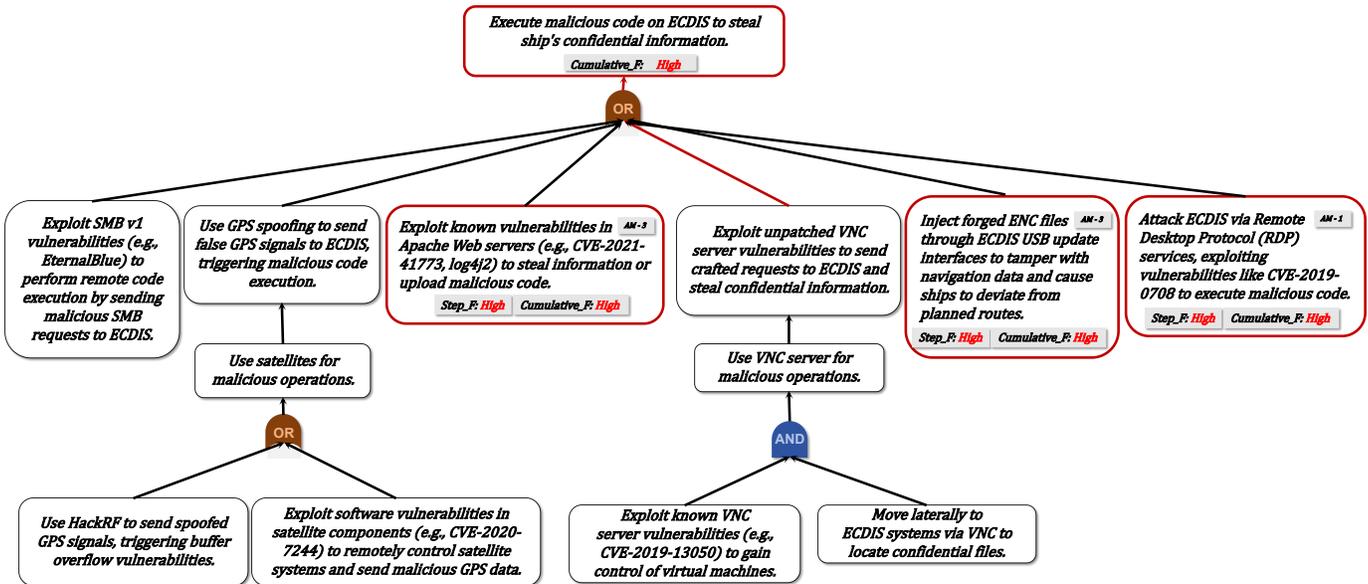

Fig. 13: Attack tree PKES (Car D)

Fig. 14: Attack tree UAV (Uav E)

Fig. 15: Attack tree ECDIS (Ship E)