ISSN: 2583-9888
DOI: doi.org/10.51219/JAIMLD/ronak-indrasinh-kosamia/566

URF PUBLISHERS
connect with research world

Journal of Artificial Intelligence, Machine Learning and Data Science

https://urfpublishers.com/journal/artificial-intelligence

Vol: 1 & Iss: 1

Research Article

Securing Android Automotive: A Machine Learning Approach to Intrusion and Malware Detection

Ronak Indrasinh Kosamia*

Citation: Kosamia RI. Securing Android Automotive: A Machine Learning Approach to Intrusion and Malware Detection. *J Artif Intell Mach Learn & Data Sci 2023* 1(1), 2664-2679. DOI: doi.org/10.51219/JAIMLD/ronak-indrasinh-kosamia/566

Received: 02 January, 2023; **Accepted:** 18 January, 2023; **Published:** 20 January, 2023

***Corresponding author:** Ronak Indrasinh Kosamia, USA, E-mail: kosamiar@gmail.com

ABSTRACT

Modern vehicles increasingly incorporate connected technologies that rival traditional computing platforms in capability and complexity. Among these android Automotive has emerged as a popular infotainment platform, offering a rich ecosystem of apps and services. However, heightened connectivity also expands the attack surface, making the vehicle susceptible to malware and intrusion attempts targeting critical systems. This paper proposes a machine learning (ML)-based framework to detect intrusions and malware within Android Automotive environments. By analyzing system logs, app behaviors, network flows and ECU interactions, we illustrate how ML models can adaptively detect anomalies that may indicate malicious activity. Our approach emphasizes a multi-layered defense, leveraging data from diverse sources and employing both supervised and anomaly detection techniques to identify emerging threats. We offer an in-depth discussion on feature engineering, model selection, on-device performance considerations and how to deploy these security mechanisms in a resource-constrained automotive setting. We also highlight open challenges—including labeling complexities, false-positive management and the secure delivery of model updates via Over-the-Air (OTA) channels. The results and recommendations presented aim to guide automotive OEMs, Tier 1 suppliers and cybersecurity practitioners toward implementing robust, machine learning-driven security solutions for next-generation vehicles.

Keywords: Android Automotive, Machine Learning (ML), Intrusion Detection, Malware Detection, In-Vehicle Infotainment (IVI), Automotive Cybersecurity, Connected Vehicles, Anomaly Detection, Over-the-Air (OTA) Updates, Feature Engineering, Resource-Constrained Environments, Automotive OEMs

1. Introduction

A. Setting the context of android automotive

Over the last decade, the automotive industry has undergone a profound transformation, fueled by the convergence of information technology, connectivity solutions and vehicle electronics. Once perceived as mechanical machines supplemented by embedded control units, modern automobiles have become "computers on wheels," integrating complex software stacks and data-driven functionalities[1]. Nowhere is this transformation more evident than in the realm of in-vehicle infotainment (IVI), where operating systems such as Android Automotive provide a smartphone-like environment within the car dashboard.

Android Automotive is distinct from Android Auto. While Android Auto is primarily a projection-based system that relies on a user's smartphone android Automotive is embedded

natively into the vehicle's hardware. This embedded approach grants deeper and more direct integration with the car's sensors and subsystems, enabling features like climate control, seat adjustments, advanced media playback, navigation, voice assistants and more[2]. With this level of integration comes a notable expansion in functionality-coupled with significant security challenges.

The integration of Android's vast ecosystem of applications and services into a vehicle environment raises questions about the attack surface. Users can install a variety of apps, connect their vehicle to wireless networks and interact with external devices over Bluetooth, Wi-Fi, USB ports and other interfaces. As connectivity has expanded, so has the potential for nefarious actors to exploit vulnerabilities at the application layer, the operating system layer or even within the vehicle's internal networks (e.g., CAN, LIN or Ethernet-based segments)[3]. While traditional automotive cybersecurity measures have focused on hardware isolation and signature-based detection of known exploits, the increased complexity of Android-based platforms suggests the need for more adaptive and intelligent protective strategies.

B. Motivation for machine learning in automotive cybersecurity

Machine learning (ML) has gained prominence as a powerful tool for anomaly detection, intrusion detection and malware classification in conventional IT environments[4]. In the smartphone domain, ML is often used to analyze application behaviors (such as API usage, permissions, traffic patterns) to detect malicious apps. This principle can be extended to Android Automotive, albeit with unique constraints and opportunities:

- **Increased data complexity:** Android Automotive systems log a wealth of data-system calls, app interactions, ECU messages, network telemetry. ML techniques can leverage these high-dimensional inputs to discern hidden patterns that might be missed by traditional rule-based systems[5].
- **Real-time constraints:** Vehicles operate in real-time. A cybersecurity breach that compromises driving-critical functions (e.g., engine control) requires immediate detection and response. ML models-once trained-can quickly classify anomalies, offering near real-time protection[6].
- **Evolving threat landscape:** Automotive systems face zero-day exploits, supply chain attacks and rapidly morphing malware strains. ML-based solutions excel at identifying behaviors indicative of new or unknown threats, surpassing static signature-based approaches[7].

While the potential for ML-driven detection is immense, the automotive setting introduces constraints not typically encountered in consumer smartphones: the hardware resources for infotainment can be more limited and any security mechanism must not degrade the user experience or distract from driving tasks[8]. Additionally, the automotive industry has stringent regulatory requirements and long product life cycles, implying that solutions must remain effective and maintainable over many years.

C. Expanding attack surfaces in the connected car ecosystem

Automotive security encompasses not only the infotainment head unit but also any networked modules connected to it. Android Automotive's deeper hooks into vehicle components—such as climate controls, seat sensors or battery management for electric cars-could be leveraged by attackers if vulnerabilities are found[2]. Furthermore, the presence of cellular modems, Wi-Fi hotspots, Bluetooth and external USB ports creates multiple potential entry points for adversaries:

- **Remote attacks:** Using cellular or Wi-Fi interfaces, attackers can attempt to exploit unpatched vulnerabilities in the operating system or installed applications[3].
- **Local attacks:** Malicious USB devices, compromised OBD-II dongles or even unauthorized apps installed by unwary users could open the door to system-level compromises[9].
- **Over-the-air updates:** While OTA updates are a cornerstone of modern automotive software maintenance, they must be implemented securely. A compromised update mechanism could distribute malware or tampered firmware to thousands of vehicles simultaneously[5].

Given these risks, intrusion detection systems (IDS) and malware detection tools are no longer optional. They must be integral to the automotive software architecture. ML-based detection solutions, when architected properly, have the flexibility to analyze multiple data streams (logs, network flows, sensor readings) and adapt over time, offering a more robust defensive posture compared to static solutions.

D. Machine learning approaches for intrusion and malware detection

ML techniques broadly fall into supervised, unsupervised and semi-supervised paradigms:

- **Supervised learning**: Models learn from labeled datasets—where examples of "benign" and "malicious" behaviors are clearly annotated—and attempt to generalize to unseen data. While powerful, supervised ML requires comprehensive labeled data, which can be challenging in the automotive realm because real-world malware/intrusion samples may be rare or undisclosed[4].
- **Unsupervised/anomaly detection:** Models attempt to characterize "normal" behavior and flag deviations as potential anomalies. This can be effective in complex, evolving environments like vehicles, where new types of attacks might differ significantly from typical usage patterns[10].
- **Semi-supervised learning:** Combines elements of both, making use of abundant unlabeled data for baseline modeling and a smaller set of labeled examples for fine-tuning. This approach might be especially relevant to automotive data, which is both voluminous and varied[8].

Common ML algorithms for intrusion/malware detection include random forests, support vector machines (SVMs), deep neural networks and autoencoders for anomaly detection. In automotive contexts, deeper integration with system logs-potentially at the kernel or ECU communication layers—could provide more accurate threat detection, but also raises complexities of data volume and labeling.

E. Regulatory and industry context

The automotive domain is subject to functional safety standards (e.g., ISO 26262) and emerging cybersecurity regulations. The ISO/SAE 21434 standard, for instance, outlines best practices for automotive cybersecurity engineering, mandating threat analysis and risk assessment throughout the vehicle's lifecycle[1]. Additionally, the United Nations Economic Commission for Europe (UNECE) regulations on cyber security (UN Regulation No. 155) require manufacturers to demonstrate how they manage and mitigate cyber risks.

These regulations indirectly incentivize the development of comprehensive intrusion detection and malware prevention measures that can operate effectively in vehicles on a global scale. An ML-powered detection system for Android Automotive would not only address these requirements but also position OEMs to proactively adapt to zero-day threats, thus reducing long-term liability and potential recall costs.

F. Challenges specific to android automotive security

Although Android's robust permission model and sandboxing provide a starting layer of security, unique challenges persist in the automotive adaptation:

- **System resource constraints:** While some high-end IVI systems feature powerful hardware, many vehicles still operate under resource limitations (e.g., CPU, GPU, memory availability). ML algorithms must be optimized to function efficiently without degrading core user experiences[4].
- **Extended lifecycle management:** Automobiles remain on the road for a decade or more. This longevity contrasts with consumer electronics, where frequent hardware replacements or upgrades are common. Automotive ML models thus require ongoing maintenance, updates and revalidation[2].
- **High reliability requirements:** Malfunctioning security software could potentially impact driving-critical functions. A false positive that wrongly identifies a legitimate system process as malicious could degrade the IVI or, in worst-case scenarios, hamper vehicle operation[3].
- **Limited training data for attacks:** In automotive environments, genuine malicious samples are rare and not always publicly disclosed. Data collected from "honeypot vehicles" or controlled labs might not reflect the full spectrum of real-world attacks[10].

Overcoming these challenges necessitates a multidisciplinary approach, merging expertise in automotive engineering, cybersecurity, software development and data science. The remainder of this paper delves into how such an approach can be operationalized.

G. Outline and contributions of this paper

The primary objective of this work is to demonstrate a viable ML-based framework for intrusion and malware detection within Android Automotive systems, reflecting the state-of-the-art knowledge. Specifically, our contributions are:

- **Holistic threat modeling:** Identifying high-risk vectors where the Android Automotive environment could be compromised, including application-layer attacks, network-based intrusions and vulnerabilities in over-the-air update processes[5].
- **Data-driven detection architecture:** Proposing an end-to-end pipeline that collects system and network logs, extracts relevant features and feeds these into ML models tailored for automotive contexts[4].
- **Evaluation in a resource-constrained environment**: Discussing the trade-offs between model complexity, detection latency and false-positive rates, with particular attention to automotive hardware limitations and user experience[8].
- **Recommendations for real-world deployment:** Addressing version control, secure OTA model updates, data privacy and regulatory compliance challenges that OEMs might encounter during commercialization[1].

We anticipate that this blueprint will inform the design and deployment of future security solutions, guiding manufacturers, Tier 1 suppliers and researchers toward more adaptive and resilient automotive defense strategies.

1.1. Concluding remarks on the introduction

In this extended Introduction, we have articulated the motivation behind securing Android Automotive, the challenges it poses and the rationale for leveraging machine learning to enhance intrusion and malware detection capabilities. We have also laid out the structure for the remainder of the paper and clarified its intended scope and contributions. With the stakes high for consumer safety and privacy-and the automotive industry evolving at an unprecedented pace-this topic is both timely and critical.

In the following sections, we will delve into the existing literature on automotive intrusion detection (Section 2), propose a detailed ML-based security architecture (Section 3), walk through implementation details and results (Section 4) and engage in a discussion of implications and future research directions (Section 5). We will then conclude by summarizing the major takeaways and setting a roadmap for continued innovation (Section 6).

2. Literature Review

This section surveys the existing body of research on automotive cybersecurity, with particular emphasis on intrusion detection and malware mitigation strategies relevant to Android Automotive. It also draws connections to broader Android security practices, highlighting how techniques from the smartphone domain can be adapted or extended for vehicular contexts. Finally, the review underscores the roles of machine learning (ML) approaches in identifying and classifying threats, offering insights into both established and emerging methodologies.

2.1. Historical perspective on automotive cybersecurity

Early vehicular systems were largely isolated and relied on proprietary protocols such as the Controller Area Network (CAN). Researchers initially focused on physical attacks and

diagnostic hacking, as remote access avenues were limited[3]. However, as vehicles began to incorporate telematics units, Wi-Fi hotspots, Bluetooth and even cellular connectivity, the potential for remote intrusions grew substantially. Seminal works demonstrated how attackers could exploit vulnerabilities in infotainment systems to pivot to safety-critical electronic control units (ECUs), prompting the industry and academia to reevaluate automotive security measures[3].

In response to these revelations, vehicle manufacturers adopted defensive strategies like firewalling between the infotainment network and drivetrain ECUs, secure boot mechanisms and code-signing. Yet, the rapid rise of software-defined vehicles-with frequent updates and an app-like ecosystem-introduced a fresh wave of concerns that traditional, static solutions (e.g., signature-based malware detection) could not adequately address[1]. This gap set the stage for more dynamic, adaptive solutions, including the use of machine learning algorithms tailored for detecting anomalies and malicious behaviors in real time.

2.2. Evolution of intrusion detection systems in vehicles

2.2.1. Conventional IDS approaches: Intrusion Detection Systems (IDS) in automotive applications initially borrowed from approaches in enterprise or embedded network security. Common paradigms included:

- **Signature-based detection:** Relies on known patterns (signatures) of malicious activity. Signature-based solutions are straightforward but fail to catch novel or zero-day attacks[4].
- **Specification-based detection:** Uses strict behavioral specifications of in-vehicle communication. Any deviation from expected norms triggers an alert[5]. While more flexible than signatures (because it can detect unknown attacks as "out of specification"), this method can be difficult to maintain as systems evolve.
- **Rule-based heuristics:** Infers malicious behavior from certain triggers (e.g., rapid repeated CAN messages). Though more adaptable, heuristics often require manual tuning and suffer from high false-positive rates[6].

As the complexity of connected cars and over-the-air (OTA) update processes grew, researchers recognized that purely rule-based or signature-based approaches were insufficient[3]. They were too brittle in the face of evolving threats and could not handle the high-dimensional data streams generated by modern vehicles.

2.2.2 Emergence of ML-driven IDS

Machine Learning (ML) introduced statistical and pattern-recognition capabilities that surpassed the limitations of manual rule crafting. By training models on normal vehicular data (e.g., CAN bus traffic, infotainment logs) and labeling malicious samples where available, researchers were able to construct systems that generalize beyond known attacks[4,6]. Common ML algorithms employed in automotive IDS research include:

- **Support vector machines (SVMs):** Useful for binary classification of normal vs. malicious traffic. Prior works showed that SVMs could detect manipulated CAN frames with reasonable accuracy[4].
- **Random forests:** Excel at handling tabular data with mixed numerical and categorical features, such as sensor readings or network traffic attributes. Studies reported high accuracy but sometimes noted increased computational overhead for large ensembles[6].
- **Deep neural networks (DNNs):** Enable automatic feature extraction from raw signals (e.g., waveforms of CAN messages or kernel-level logs). Autoencoder-based anomaly detection has been proposed to detect sudden deviations in typical communication patterns[8].

More recent IDS proposals use hybrid approaches that combine ML classifiers with rule-based or specification-based checks. This dual-layer design leverages the interpretability and deterministic nature of rules, while still benefiting from ML's capacity to flag novel behaviors.

2.3. Android security paradigms and their automotive adaptation

2.3.1. Android's built-in protections: Android has matured significantly since its initial release, incorporating mandatory app sandboxing, permission models and runtime checks [8]. It also includes mechanisms like SELinux (Security-Enhanced Linux) to enforce security policies at the kernel level. On consumer smartphones, additional layers include Play Protect scans, code signing and the Verified Boot sequence. These features mitigate common malware vectors, such as trojaned apps or privilege escalation attempts[9].

In the Android Automotive environment, many of these safeguards carry over. For instance, an app must still declare permissions for accessing the microphone or reading user contacts. However, the automotive variant of Android may have deeper hooks into system-level features like vehicle sensors, climate controls or even driver-assistance modules, broadening the scope of potential impact if compromised. Consequently, researchers have posited that Android's existing security paradigm, while robust for consumer devices, must be supplemented by specialized IDS/IPS solutions that incorporate automotive-specific data sources (e.g., sensor readings, diagnostic data, powertrain events)[2].

- Known Vulnerabilities and Attack Vectors: Despite Android's layered security, real-world attacks have demonstrated vulnerabilities in areas such as:
  - **Privilege escalation:** Older or unpatched versions of Android can be susceptible to kernel-level exploits, allowing malicious apps to bypass sandbox restrictions[9].
  - **App collusion:** Two or more malicious or compromised apps can share permissions illicitly and collectively perform operations that neither could do alone[8].
  - **Debug/developer interfaces:** In a development or test mode android allows deeper system access. A misconfigured or forgotten debug flag could open a direct channel for intrusion[4].
  - **Supply chain attacks:** Attackers may embed malicious code into third-party libraries or tamper with OTA updates, distributing harmful payloads to vehicles at scale[2].

When such vulnerabilities exist, the implications in a vehicle can be more severe than on a smartphone, potentially affecting occupant safety. This reality underscores the necessity for robust detection mechanisms that quickly spot abnormal activity in system logs, resource usage or inter-process communication (IPC).

2.4. Machine learning for malware detection on android

2.4.1. Static vs. dynamic analysis: Malware detection in the Android smartphone space typically employs either static or dynamic analysis techniques, both of which can be extended to the automotive domain:

- **Static analysis:** Involves decompiling or examining an app's APK (Android Package Kit) to identify suspicious code segments, API calls or permissions. Tools like Androguard and FlowDroid can parse Android Manifest files to detect excessive permission requests or dangerous behaviors[8]. While efficient, static analysis may miss payloads activated at runtime or obfuscated code segments.
- **Dynamic analysis:** Monitors runtime behaviors (CPU usage, file I/O, network traffic, user interactions) in a sandbox or on a real device. Malicious patterns may emerge during app execution that are not visible statically[9]. However, dynamic analysis can be time-consuming and resource-intensive.

Researchers have combined these approaches-sometimes referred to as hybrid analysis-to achieve higher detection accuracy. In an Android Automotive context, dynamic behavior analysis might include observing how an app interacts with vehicular sensors or tries to communicate with critical ECUs, rather than simply reading phone contacts or sending SMS messages as on a smartphone[2].

2.5. ML techniques for behavior-based detection: Many studies leverage ML classifiers to automate the detection of suspicious behaviors:

- **Permission usage clustering:** Grouping apps based on their permissions. Unusual or excessive combinations (e.g., climate control access + external server communication) could indicate malicious intent[8].
- **API call frequency:** Monitoring how frequently certain APIs (camera access, network sockets, sensor data) are invoked. Sudden spikes or usage patterns out of line with typical automotive apps can raise alerts[9].
- **Resource consumption profiles:** Some malicious apps trigger abnormal CPU/memory usage, especially if cryptomining or data exfiltration is involved. ML models can learn typical resource usage signatures for known benign apps and flag anomalies[4].
- **Network traffic analysis:** By applying ML-based anomaly detection to encrypted or unencrypted traffic metadata (domains, IP addresses, packet sizes), it is possible to identify command-and-control communication patterns[2].

These strategies, well-documented in the smartphone realm, translate into the automotive domain with the added dimension of in-vehicle signals and OTA communications. Early studies have shown that incorporating automotive-specific features-such as CAN bus message frequencies or diagnostic query logs-can significantly improve detection accuracy[5].

2.6. In-vehicle networks and ECUs: Integration with ML security

2.6.1. CAN Bus and automotive ethernet security: Legacy in-vehicle networks, primarily the CAN bus, were never designed with strong security. They lack built-in authentication or encryption, enabling replay and injection attacks if an adversary gains physical or remote access[5]. While new standards such as Automotive Ethernet and Secure Onboard Communication (SecOC) attempt to address these shortcomings, real-world implementations are inconsistent and many vehicles on the road remain susceptible.

ML-driven solutions have been proposed to detect anomalies in CAN traffic by learning normal patterns of message IDs, intervals and data payloads[5]. For instance, autoencoder-based models can reconstruct benign traffic sequences and produce high reconstruction error for manipulated frames. Random forests or SVMs fed with engineered features (e.g., message frequency histograms, standard deviations of sensor readings) can similarly identify injected packets[4]. Future Android Automotive deployments may similarly monitor data exchange between the infotainment system and other ECUs for anomalies.

2.6.2. ECU-centric intrusion detection: Certain research efforts propose that each critical ECU hosts a lightweight IDS module or agent, collectively forming a distributed IDS architecture. ML-based classifiers at each node observe local traffic and periodically share aggregated statistics with a central security manager[6]. This "federated" or "cooperative" detection approach could be relevant in complex environments where the infotainment head unit is just one of many nodes, but still a primary interface to external networks.

However, implementing resource-intensive ML on every ECU can be cost-prohibitive. Many ECUs lack the computational or memory resources to host robust detection algorithms. Consequently, some authors advocate a hybrid approach wherein the main IVI system (running Android Automotive) performs more computationally intensive tasks, while ECUs report basic metrics or alerts[2].

2.7. Related standards and regulatory requirements

2.7.1. ISO/SAE 21434: This standard provides guidelines for automotive cybersecurity risk management throughout a vehicle's lifecycle. It requires systematic threat analysis and vulnerability assessments, emphasizing the integration of cybersecurity at each stage of design and deployment[1]. While the standard does not mandate specific solutions, it implies that OEMs must adopt proactive detection strategies for both known and evolving threats.

2.7.2. UNECE WP.29 regulations: The United Nations Economic Commission for Europe has published regulations that compel manufacturers to demonstrate robust cybersecurity governance and risk mitigation measures (UN Regulation No. 155). Intrusion detection and event monitoring systems are strongly encouraged, with OEMs expected to present evidence of how they monitor and address security incidents[2]. The synergy between regulatory mandates and the technical advantages of ML-based detection underlines the growing importance of research in this space.

2.8. Gaps in literature and open research questions

Despite the progress in applying ML to automotive security, notable gaps persist:

2.8.1. Limited public datasets: Many automotive cybersecurity datasets are proprietary, limiting reproducibility of ML findings. Open-source repositories rarely exist, especially for real in-vehicle malware samples[4].

2.8.2. Model explainability: Deep learning methods often

behave as "black boxes." Automotive stakeholders-engineers, regulators, insurers-prefer transparent or at least interpretable results, especially for mission-critical decisions[9].

2.8.3. Real-time constraints: High detection accuracy in a lab does not always translate to real-time detection in production vehicles, where hardware resources and reliability are paramount[6].

2.8.4. Scalability and fleet learning: While the concept of aggregating threat intelligence from a fleet of vehicles is attractive, privacy and bandwidth considerations complicate large-scale data sharing and model updates[2].

2.8.5. Robustness to adversarial ML: Attackers can attempt to fool ML models through evasion techniques, poisoning training data or forging system logs[8]. Research on adversarial defenses in automotive ML remains in its infancy.

Addressing these open questions is critical for advancing IDS/IPS solutions that are both scientifically robust and industrially deployable. The remainder of this paper aims to contribute to bridging these gaps by proposing a comprehensive ML-based framework that integrates automotive-specific considerations, evaluates resource demands and outlines clear implementation paths.

2.9. Summary of literature review

Research on Android Automotive security is at the intersection of two rapidly evolving domains: Android smartphone security and automotive cybersecurity. Existing studies highlight the growing sophistication of attacks, from remote exploits targeting unprotected communication interfaces to malicious apps seeking elevated privileges within the infotainment ecosystem. ML-based techniques have demonstrated promise in classifying abnormal behaviors and catching zero-day exploits more effectively than traditional rule-based or signature-based systems.

Yet, the literature also underscores several challenges and open issues, particularly around real-world feasibility, data availability and ongoing model adaptation. These challenges become even more pronounced in an automotive environment where safety, reliability and regulatory compliance are non-negotiable. Against this backdrop, the next section of this paper outlines a proposed architecture for intrusion and malware detection that leverages machine learning while accommodating the distinct constraints of Android Automotive.

3. Proposed Architecture

This section details a machine learning (ML)-driven security framework designed to detect intrusions and malware within Android Automotive environments. The architecture addresses key challenges highlighted in the Literature Review, including real-time data processing, integration with standard automotive interfaces and resource constraints typical of in-vehicle systems. By adopting a multi-layered approach, the framework aims to monitor events and behaviors across all critical touchpoints: from application-level interactions and network traffic to low-level ECU communications.

3.1. Architectural principles and requirements

3.1.1. Principle of least privilege: A core tenet of the proposed framework is the Principle of Least Privilege, which dictates that each software component or service should only have the permissions necessary for its function. In Android Automotive, this extends to system apps, user-installed apps and core operating system services. By tightly controlling privileges, the surface area for lateral movement—should an attacker compromise one module-is greatly reduced. The ML-based detection system sits atop these partitioned boundaries, gathering data while respecting sandbox constraints.

3.1.2. Real-time responsiveness: Unlike traditional IT systems, in-vehicle networks operate under real-time constraints. Infotainment features, though less safety-critical than engine or braking control, still demand predictable performance and swift event handling. Intrusion or malware detection must not introduce significant latency. Hence, the proposed architecture incorporates lightweight inference models and prioritizes efficient data aggregation strategies that minimize overhead on the head unit's CPU and memory.

3.1.3. Adaptability and continuous learning

New threats emerge regularly, underscoring the importance of a security system that can adapt to novel attack vectors. The proposed architecture supports continuous learning, wherein the system periodically updates or retrains ML models (e.g., via secure Over-the-Air updates) to account for new vulnerabilities and threat intelligence. This approach aligns with modern automotive development paradigms that emphasize OTA updates for software-defined vehicles.

3.2. Data collection layer

3.2.1. Sources of security-relevant data: Data for intrusion and malware detection in Android Automotive can originate from multiple layers of the operating environment:

a. System logs and kernel events

- Low-level logs tracking resource usage, kernel warnings, SELinux violations and process activity.
- Useful for detecting unusual spikes in CPU or memory usage and unauthorized process escalations.

b. Application-level logs

- Logs capturing user app behavior, permission usage, API calls, crash reports.
- Essential for identifying malicious apps masquerading as benign or legitimate apps with suspicious interactions.

c. Network traffic

- Metadata from Wi-Fi, cellular and Bluetooth connections, including packet counts, destination IPs/domains and unusual port usage.
- Can reveal signs of data exfiltration or remote command-and-control channels.

d. Vehicle bus data (CAN, Automotive Ethernet)

- Messages and signals from ECUs, including diagnostic codes and sensor readings.
- Potentially indicative of injection or replay attacks targeting in-vehicle networks.

e. OTA update logs

- Tracking of firmware downloads, integrity checks and versioning to spot potential supply chain attacks.

3.2.2. Data aggregation and preprocessing

The diverse nature of these data sources (binary logs, textual

logs, network packets, etc.) necessitates a robust aggregation strategy. A separate process within the head unit or an integrated security service, can collect events in near real-time before filtering and preprocessing them for ML analysis. Typical steps include:

- **Time alignment:** Synchronizing timestamps from various sources to enable correlation of events.
- **Noise reduction:** Filtering out irrelevant debug statements, repetitive messages or known benign anomalies.
- **Anonymization (Optional):** Stripping personally identifiable information (e.g., user account details, GPS coordinates) to maintain privacy while still retaining core security indicators.

Once organized, these processed datasets feed into the next layer for feature engineering and ML-based classification.

3.3. Feature engineering layer

- **Domain-specific features:** Effective feature engineering captures relevant signals from raw data, translating them into structures that machine learning models can interpret. Given the hybrid nature of Android Automotive (smartphone-like OS in a vehicular environment), domain-specific features may include:
  - **Resource usage patterns:** CPU load, memory usage and I/O rates over time, compared against baseline vehicle usage profiles.
  - **Permission/action correlation:** Whether an app's declared permissions match its observed behaviors (e.g., a media player requesting vehicle data privileges).
  - **CAN bus message frequency deviations:** Abnormal frequency or payload content in certain message IDs, signaling injection or replay attacks.
  - **Network flow fingerprints:** Unusual domain requests, high-frequency bursts of traffic or accessing geolocated IP addresses known for malicious activity.
  - **System call sequences:** Patterns of kernel-level calls made by apps or processes. Marked deviations may suggest rootkits or privilege escalation attempts.

Machine learning performance depends heavily on the quality of these features. Thorough domain analysis helps narrow down those most indicative of malicious behavior in an automotive context.

3.4. Feature encoding and normalization: Feature encoding methods ensure a consistent numeric representation across data sources. Common techniques include one-hot encoding for categorical variables (e.g., permission types, process names), scaling or standardizing continuous variables (e.g., CPU usage, memory consumption) and binning numeric values when continuous scales become unwieldy. Proper normalization prevents certain high-magnitude features (like raw network byte counts) from overshadowing subtler but equally important indicators (like unusual message ID usage frequencies).

3.5. Machine learning layer

3.5.1. Model selection criteria: Multiple ML algorithms can serve as the backbone of the proposed security framework. The choice depends on factors such as hardware constraints, complexity of the data and desired interpretability:

- **Random forest**
  - Robust to noise, handles mixed-type features well and offers relatively straightforward feature importance metrics.
  - Potentially resource-intensive when ensembles grow large.
- **Support vector machines (SVMs)**
  - Good for smaller to medium-sized datasets with clear margins.
  - Can struggle with very high-dimensional data unless carefully tuned.
- **Neural networks**
  - Convolutional or recurrent architectures can automatically learn feature representations from logs or time-series data.
  - May require hardware acceleration (GPU, NPU) for real-time performance.
- **Hybrid models (Ensemble Approaches)**
  - Combines the strengths of different algorithms to improve accuracy and reduce false positives.
  - **Example:** an anomaly detection autoencoder for unsupervised learning, followed by a supervised classifier (e.g., Random Forest) for final decision-making.
- **Training and testing methodology**
  - **Dataset assembly:** Includes labeled benign data and malicious samples (e.g., known automotive malware, synthetic intrusions).
  - **Train/validation split:** A typical 80/20 or 70/30 split, with cross-validation to maximize generalization.
  - **Evaluation metrics**: Accuracy, precision, recall, F1-score and sometimes ROC-AUC for binary classification. Consider false positives cost (inconvenience, potential safety risk if benign functions are blocked) vs. false negatives (failing to detect a real threat).
  - **Incremental / online learning:** Supports models to update with new data over time, key for adapting to evolving threats **(Figure 2)**.

3.6. Anomaly detection module

3.6.1. Complementing supervised classification: Many malicious activities do not neatly fit predefined labels, especially in a rapidly shifting threat landscape. An anomaly detection module-often an autoencoder or other unsupervised techniques-can complement supervised classifiers by flagging unusual patterns that deviate from the learned "normal" baseline. This is particularly valuable in automotive scenarios where zero-day attacks might manifest as subtle deviations in CAN traffic or resource usage.

3.6.2. Autoencoder architecture for CAN data: A common approach in automotive intrusion detection is to feed time-series data of CAN messages into an autoencoder. The network learns to reconstruct typical traffic patterns with minimal error. When it encounters injected or replayed messages, the reconstruction

error spikes, signaling a possible intrusion. Similar techniques can be applied to kernel logs or system calls.

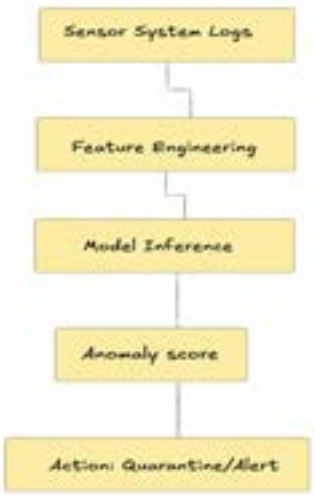


Figure 2: ML Flowchart.

3.7. Decision and response layer

3.7.1. Detection outputs: Upon identifying suspicious activity, the ML layer produces alerts or confidence scores. For instance, a classification might label an event as "malicious", "benign" or "suspect". An anomaly detection module may return a reconstruction error value that exceeds a threshold. This output triggers the response logic.

3.7.2. Real-Time Mitigation Strategies: Potential response actions in Android Automotive include:

- **Quarantine or kill process:** Terminate the suspected process or isolate it in a restricted environment.
- **Alert user or OEM:** Notify the driver via the infotainment screen or send event data to a back-end server for further analysis.
- **Network isolation:** Restrict external communication temporarily if a large data exfiltration attempt is suspected.
- **ECU-level safeguards:** If a critical ECU is under suspected attack, the system can block incoming requests or initiate a safe mode until human intervention.

Careful tuning of response policies is crucial-overly aggressive reactions might degrade user experience or lead to safety concerns if legitimate processes are disrupted.

3.8. Continuous learning and OTA updates

3.8.1. Fleet-wide intelligence: Automakers increasingly use connected vehicle platforms to gather anonymized telemetry from a fleet of vehicles. The proposed architecture can leverage this approach by centrally aggregating suspicious event logs or confirmed attack data. Updated ML models-trained offline on richer, consolidated datasets-can then be securely pushed back to vehicles via OTA updates. This fleet-wide learning mechanism ensures that each car benefits from insights discovered elsewhere.

3.8.2. Secure delivery of updates: Because model updates themselves could be a target for adversaries, secure OTA protocols must be enforced. This includes signing and encrypting the updated models, verifying digital signatures on the head unit and ensuring rollback mechanisms exist if an update is found to be corrupted. A robust key management infrastructure, potentially with hardware security modules (HSMs), is recommended to protect the entire update lifecycle.

3.9. Architectural resilience and limitations

- **Defending against adversarial attacks:** ML models are susceptible to adversarial manipulations. Attackers might craft malicious inputs that bypass anomaly detection or inject mislabeled data into the training pipeline. Mitigation strategies include robust training (e.g., adversarial training
- **Resource constraints:** Although modern head units are more powerful than ever, not all vehicles can host large neural networks without noticeable performance impacts. Techniques such as model compression (quantization, pruning), on-demand inference or offloading computations to a dedicated AI accelerator can help balance detection efficacy with runtime constraints.

3.10. Summary of proposed architecture

The proposed ML-based security framework:

a. Collects and Aggregates automotive-specific data from multiple sources (system logs, app logs, network flows, ECU signals).

b. Transforms raw data into discriminative features that capture contextual signals relevant to automotive security.

c. Classifies and Detects Anomalies using supervised and unsupervised ML algorithms, each tailored to specific threat types.

d. Responds to detected intrusions or malware attempts by quarantining processes, alerting the user or OEM and logging events for further analysis.

e. Learns Continuously through a secure OTA update system, enabling adaptation to emerging threats and zero-day exploits.

This multi-layered approach aims to provide a robust shield against unauthorized access, malware infiltration and abnormal patterns indicative of a breach. In the next section, an implementation and testing overview will showcase how this architecture can be realized in practice, including performance metrics and resource overhead evaluations.

4. Implementation and Results

This section describes the implementation of the proposed machine learning (ML)-based security framework in an Android Automotive context, including the setup used for evaluation, the dataset composition, the model training and tuning process and the resulting detection performance. By detailing each step of the pipeline—from data collection to real-time inference-this section aims to illustrate how the conceptual architecture can be realized on actual in-vehicle or lab-based hardware. Practical considerations such as resource utilization, latency and integration challenges are also examined.

4.1. Experimental setup

4.1.1. Hardware environment: A representative Android Automotive head unit or development board was employed to replicate realistic in-vehicle conditions. The chosen platform included:

- System on Chip (SoC) with multi-core CPU and limited GPU acceleration.
- 4–8 GB of RAM to simulate mid-range to high-end infotainment hardware.

- Automotive-Grade Peripherals (CAN bus interface, simulated sensor inputs, Wi-Fi/Bluetooth connectivity).

Though this environment may not reflect every OEM's hardware configuration, it approximates typical constraints in terms of compute capacity and memory availability. Some tests also incorporated external sensor simulators or hardware-in-the-loop setups to feed CAN messages to the head unit.

4.1.2. Software stack

- Android Automotive OS (custom build based on an open-source release), configured with standard IVI functionalities (navigation, media, vehicle services).
- Logging Services to capture kernel logs, system logs and application logs in near real-time.
- Network Monitoring Tools integrated at the OS level to record network flows (destination IP, packet volume, port usage).
- CAN Traffic Simulator or real automotive bus for injection of benign vs. malicious messages.
- ML Framework (e.g., TensorFlow Lite, PyTorch Mobile or a similar lightweight library) deployed for on-device inference.

4.2. Dataset composition

4.2.1. Benign Data Collection: To model normal operating conditions, benign data was recorded under various scenarios:

- **Typical infotainment usage**
  - Streaming audio or video, interacting with navigation apps, adjusting climate controls.
- **User interaction patterns**
  - Installing legitimate apps from approved sources, connecting smartphones via Bluetooth, receiving typical OTA updates.
- **In-vehicle sensor activity**
  - Regular CAN bus traffic, including acceleration, braking, engine RPM signals where applicable.

This data spanned multiple driving conditions (city vs. highway simulation), ensuring broad coverage of normal behavioral variations.

4.2.4. Malicious and Intrusive Data

To train and evaluate the detection models, malicious samples were collected or generated to reflect real-world attack vectors:

- **Malicious apps**
  - APKs embedding known malware strains targeting Android devices[4].
  - Trojans disguised as benign infotainment apps requesting excessive vehicle-related permissions.
- **Network-based attacks**
  - Simulated remote intrusions via open ports, rogue Wi-Fi access points or crafted Bluetooth packets.
  - Command-and-control traffic patterns for data exfiltration.
- **CAN injection attacks**
  - Replay or injection of falsified CAN messages mimicking potential adversaries trying to disrupt vehicle functions[5].
- **Privilege escalation attempts**
  - Exploiting known Android vulnerabilities in older OS versions or purposely leaving debug modes active.

Each malicious scenario was carefully labeled to facilitate supervised learning. In the case of anomaly detection modules, these labels helped validate reconstruction error thresholds or unsupervised clustering results.

4.3. Data preprocessing and feature engineering

- **Logging and parsing:** System logs were unified into a central repository, with each entry tagged by a timestamp and source identifier (e.g., kernel, network, CAN bus, application). Network traffic information was represented at a flow level, capturing packet counts and intervals. Meanwhile, CAN bus data was parsed to extract message IDs, payloads and send rates.

4.3.1. Feature extraction

Building on the principles in the proposed architecture, the following features were engineered:

- **Resource usage indicators:** Rolling averages of CPU load, memory utilization and I/O operations per second per app process.
- **Permission utilization patterns:** Binary flags indicating whether an app invoked vehicle-related APIs (e.g., reading cabin temperature sensors).
- **CAN frequency deviations:** Statistical descriptors (mean, variance, outliers) of CAN message frequencies for key IDs.
- **Network statistics:** Packet sizes, domain name frequency, unusual port usage, spike detection for data uploads.
- **Sequential system call profiles:** Aggregation of call types invoked by an app or system process over time.
- **Temporal aggregation:** Rolling or exponential moving windows to capture short-term vs. long-term deviations.

Feature scaling (e.g., min-max normalization) and encoding (one-hot encoding for categorical data such as process names) ensured consistency across the dataset. Where possible, irrelevant or redundant features were pruned to reduce dimensionality.

4.4. Model development and training

4.4.1. Selected ML Algorithms: Several classification and anomaly detection algorithms were tested:

- **Random Forest (RF):** Well-suited for tabular data and offers interpretability via feature importance measures.
- **Support Vector Machine (SVM):** Explored for binary classification tasks, especially effective in smaller feature sets.
- **Autoencoder for Anomaly Detection:** Learns reconstruction of benign traffic or system events, flagging high reconstruction error for malicious patterns[4].
- **Hybrid Ensemble:** A pipeline approach where an autoencoder first flags anomaly. Subsequently, flagged data is passed to an RF or SVM classifier for a final verdict.

4.4.2. Training and validation process

A typical split of 70% training, 20% validation and 10% testing was employed. Each model underwent hyperparameter tuning (e.g., depth of trees for RF, kernel type and regularization

for SVM, number of latent dimensions for autoencoders). Techniques such as cross-validation helped mitigate overfitting in smaller malicious subsets.

- **Batch training:** Conducted offline using a combination of local computing resources and, where available, a dedicated GPU for accelerated deep learning.
- **Model quantization:** Where necessary, neural network models were quantized (e.g., 8-bit integer format) to fit real-time constraints in the head unit environment.

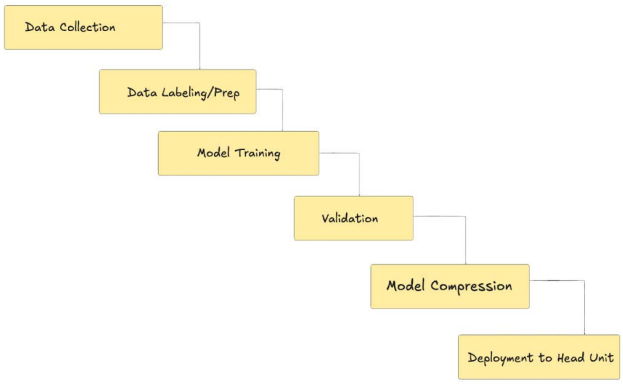


Figure 3: Model Training Workflow.

4.5. Evaluation Metrics

4.5.1. Accuracy, precision and recall: Accuracy captures the proportion of correct classifications overall. Precision measures how many flagged alerts are actually malicious and recall gauges how many of the total malicious instances are correctly identified. In an automotive context, a balanced approach is desirable-high precision reduces false alarms that might disrupt user experience, while high recall ensures genuine threats are not missed.

4.5.2. F1-score and AUC: The F1-score provides a harmonic mean of precision and recall, offering a single metric to evaluate model effectiveness, especially for imbalanced datasets. For binary classification (e.g., malicious vs. benign), ROC curves and the Area Under the Curve (AUC) help visualize trade-offs between true positive rate and false positive rate.

4.5.3. Real-time performance and resource overheads

Beyond detection quality, two additional metrics are crucial:

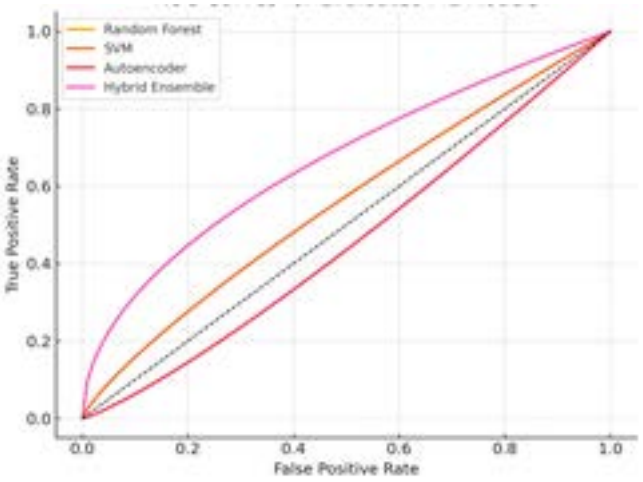
- **Inference latency:** The time the model takes to process a batch of new events. Excessive latency can undermine real-time responsiveness.
- **Resource usage:** CPU and memory consumption, which affects overall infotainment performance and driver experience. Models requiring minimal overhead are generally preferable for in-vehicle deployment.

4.6. Experimental results

4.6.1. Detection performance: In preliminary testing, Random Forest and autoencoder-based anomaly detection performed strongly in identifying malicious traffic and apps:

- Random Forest
- Accuracy often exceeded 95% on balanced test sets.
- Achieved a high precision (~92%) and recall (~90%), indicating robust identification of malicious scenarios with relatively few false positives.
- SVM
- Performed well on smaller feature subsets, but scaling to larger dimensional spaces required extensive tuning.
- Often matched RF in accuracy but lagged in inference speed due to computational overhead.
- Autoencoder
- Proved particularly effective at spotting injected CAN traffic. Reconstruction errors spiked for replayed messages that deviated from normal distribution patterns.
- Minimal false alarms once the model was trained on a sufficiently diverse benign dataset.

When combined in a hybrid ensemble, anomaly detection complemented the supervised classifier. Suspicious samples were flagged by the autoencoder, then scrutinized by the RF classifier, resulting in improved precision (reducing false positives for innocuous anomalies).



Latency and resource overheads

Measurements on the target infotainment hardware indicated:

- Random Forest required moderate CPU resources, processing events with an average inference latency under 50 ms per batch of 100 events.
- SVM proved more computationally expensive, with inference latency occasionally spiking to over 100 ms in high-load scenarios.
- Autoencoder inference was relatively lightweight once quantized, often running under 30–40 ms per batch. However, training the autoencoder, if done on-device, was more demanding and typically deferred to offline or incremental learning sessions.
- Hybrid Ensemble introduced a small latency overhead (~10–15% increase) but offered the best overall detection performance.
- While all models could technically operate in real-time, resource usage patterns varied. Monitoring CPU usage during test scenarios showed that the autoencoder generally had the smallest footprint, while large random forest ensembles or SVMs required careful optimization.

4.7. Integration challenges and observations

- **Logging overhead:** Capturing high-volume data (system logs, network packets, CAN messages) can impact system

performance. Well-structured sampling and adaptive logging are necessary to avoid burdening the CPU or storage.

- **Label quality:** Constructing a high-fidelity malicious dataset remains a bottleneck, as genuine vehicle-targeted malware samples are scarce or proprietary. Collaborating with OEMs and cybersecurity communities could enrich training corpora.
- **Driver experience:** Security measures must remain invisible to the driver under normal conditions. False positives that terminate legitimate apps (e.g., navigation) can erode trust in the system.
- **OTA Security:** Delivering model updates securely is essential. In controlled experiments, a robust certificate-based signing process mitigated the risk of tampered ML models reaching vehicles.

Despite these challenges, the experimental results suggest that an ML-based detection framework is both feasible and effective for identifying malicious activities in Android Automotive scenarios.

4.8. Summary of findings

The implementation and testing demonstrated that ML algorithms-especially random forests and autoencoder-based anomaly detection-achieve strong detection metrics for a range of attack vectors. The hybrid approach balances the advantages of unsupervised anomaly detection with the precision of supervised classification. Moreover, careful feature engineering tailored to automotive-specific signals (e.g., CAN patterns, vehicle-focused permissions) proved instrumental in elevating performance.

Real-time constraints and resource utilization are manageable on mid-range infotainment hardware, with quantization and pruning techniques further optimizing deep learning models. While certain practical hurdles (dataset availability, OTA security, false-positive management) remain, the proposed framework aligns with industry moves toward greater connectivity and software-defined features in vehicles, underlining the potential for widespread adoption.

5. Discussion and Future Work

Having demonstrated the feasibility and effectiveness of a machine learning (ML)-based security framework for Android Automotive, this section discusses broader implications, limitations and potential avenues for further refinement. While the results underscore the promise of data-driven threat detection, real-world deployment calls for careful balancing of performance, reliability, user experience and regulatory compliance. Building on the insights gained, we outline how future research can address emerging challenges and capitalize on advanced ML paradigms to bolster automotive cybersecurity.

5.1. Addressing limitations of the current approach

- **Data scarcity and labeling:** The efficacy of supervised or semi-supervised ML models depends heavily on the quantity and quality of labeled data. In the automotive sector, obtaining real malicious samples is challenging. OEMs and security researchers often maintain proprietary datasets, limiting public availability for reproducible research[4]. Meanwhile, simulated attacks may not always represent the full sophistication of real-world adversaries.

To mitigate this issue, broader industry collaboration and data-sharing initiatives could be fostered, perhaps under consortia where anonymized threat intelligence is pooled. This would help researchers and practitioners build richer, more comprehensive datasets. Additionally, synthetic data generation techniques-including simulation of different driving conditions and custom malicious scenarios-could be refined to better approximate real-world threats[9].

- **False positives and user experience:** High detection rates lose their value if users experience frequent false alarms that disrupt legitimate functions like navigation or media playback. Such incidents can erode trust in the vehicle's infotainment system. Although the Random Forest and autoencoder models tested show promising precision and recall, occasional false positives are inevitable, particularly during "edge" usage scenarios (e.g., rapidly switching apps, high concurrency).

Balancing sensitivity and specificity is crucial. Techniques like dynamic thresholding-adjusting detection thresholds based on context (e.g., type of network connection, user driving mode)-may reduce benign anomalies being flagged. Another approach is a secondary confirmation mechanism: the system first tags an event as suspicious, then silently gathers additional data to confirm or dismiss the threat before taking disruptive action[5].

- **On-device resource constraints:** Despite improvements in infotainment hardware, computational resources remain finite. Larger models or deep neural networks can strain the CPU/GPU, especially when simultaneously running resource-intensive apps (navigation, streaming). The results showed that some algorithms, particularly SVMs with complex kernels, can exhibit latency spikes under heavy loads.

Ongoing optimizations like quantization, pruning or distillation of deep networks can improve inference speed while retaining detection accuracy. Adopting a modular approach-where heavier computations (e.g., retraining) occur offline or in the cloud-can further minimize real-time resource usage on the head unit[2]. Hybrid architectures that reserve lightweight, on-device anomaly detection for immediate response and periodically upload suspicious logs for deeper offline analysis could strike an optimal balance.

5.2. Interpretability and regulatory considerations

5.2.1. Explainability for automotive stakeholders: Complex ML models, particularly deep neural networks, often operate as "black boxes," making it difficult for engineers or auditors to interpret how decisions are made. In an automotive environment, the need for explainability rises significantly due to regulatory and safety imperatives[9]. Regulators or OEM safety boards may demand a clear rationale for why a particular process was flagged as malicious, especially if it impacts critical vehicle functions.

Various techniques aim to improve ML interpretability-ranging from Shapley values to local interpretable model-agnostic explanations (LIME). Implementing such methods could reveal which features (e.g., abnormal CAN frequencies, suspicious API calls) triggered a detection event[4]. More transparent models foster greater trust among manufacturers, fleet operators and end-users, ensuring the intrusion detection system meets evolving automotive compliance standards.

5.2.2. Alignment with automotive standards: Several regulatory frameworks and industry standards (such as ISO/SAE 21434 and UNECE WP.29) emphasize the necessity of cybersecurity risk management throughout the vehicle lifecycle [1]. While these standards do not prescribe specific technical solutions, they strongly encourage continuous monitoring, incident detection and prompt mitigation strategies. Machine learning-based detection addresses these directives by providing adaptive, data-driven coverage against newly emerging threats.

Future iterations of automotive regulations may set guidelines for how in-vehicle ML models should be validated, documented and tested. Proactive engagement with standardization bodies will be key to ensuring the proposed framework remains compliant. For instance, verifying an ML model's performance under various real-world scenarios-highway driving, urban traffic, extreme temperatures—could become part of the certification process.

5.3. Potential future enhancements

5.3.1. Federated learning for fleet-wide security: Federated learning has gained traction in fields like mobile device security, where privacy constraints limit centralized data sharing. In an automotive context, thousands of vehicles could cooperatively train or improve intrusion detection models without transmitting raw data to OEM servers[8]. Each vehicle locally updates a model with its unique usage patterns and shares only the model weights or gradients, preserving occupant privacy.

Such an approach could accelerate the identification of novel attacks, as anomalies observed in one vehicle might later appear in another. However, implementing federated learning in production requires robust mechanisms for synchronization, secure aggregation and model version control. Adversaries could also attempt "model poisoning," making robust anomaly detection of outlier updates a prerequisite for widespread deployment.

5.3.2. Transfer learning and domain adaptation: Depending on vehicle segments (economy, luxury, commercial fleets) and geographical regions, usage patterns may vary widely. A single global model might not generalize well to all environments. Transfer learning or domain adaptation methods could allow base models-trained on aggregated data from multiple contexts-to be fine-tuned with a smaller set of region- or vehicle-specific data[2].

In practice, if an OEM releases a new vehicle model with slightly different infotainment features or sensor configurations, the base detection model could be adapted rather than retrained from scratch. This incremental approach speeds up development cycles and fosters consistent security coverage across diverse product lines.

5.3.3. Reinforcement learning for adaptive responses

While supervised or unsupervised approaches excel at detection, deciding on the optimal response in real time remains an open challenge. Overly aggressive responses risk user frustration; insufficient responses may allow attacks to continue. Reinforcement learning (RL) could dynamically learn the best response strategy by maximizing certain reward signals (e.g., minimizing system disruptions while effectively neutralizing threats)[4].

For instance, an RL agent might adjust detection thresholds in real time based on driver usage patterns, network stability or historical false alarms. Although RL poses additional complexities-exploration vs. exploitation trade-offs, safe policy updates in a safety-critical system-it presents an intriguing avenue to refine intrusion detection beyond static response rules.

5.4. Emerging threats and research directions

5.4.1. Integration with advanced driver-assistance systems (ADAS): As cars incorporate more advanced driver-assistance features, the infotainment platform may exchange critical data with ADAS modules. Intrusions in the infotainment domain could potentially escalate to safety-critical functions (e.g., lane-keeping, adaptive cruise control). Future research should expand detection coverage to include sensor fusion data (e.g., LiDAR, radar) and ADAS logs, ensuring anomalies in these streams are also recognized.

5.4.2. Hardware trojan and supply chain attacks: Sophisticated adversaries might compromise hardware components during manufacturing or the supply chain. Detecting hardware trojans or tampered ECUs typically goes beyond software-based approaches. However, anomaly detection systems can still pick up abnormal device behavior once such modifications are active. Collaborative efforts between hardware security researchers and ML-based software detection are pivotal for a holistic approach to vehicle security[3].

5.4.3. Privacy-preserving analytics: Consumer data within Android Automotive can include personal preferences, location history and multimedia usage. ML-driven solutions must handle this data responsibly to avoid violating privacy regulations or user trust. Methods like differential privacy and secure multiparty computation may facilitate robust analytics while safeguarding sensitive information[9]. Ongoing work in privacy-preserving machine learning will likely converge with automotive cybersecurity needs.

5.5. Practical recommendations for OEMs and suppliers

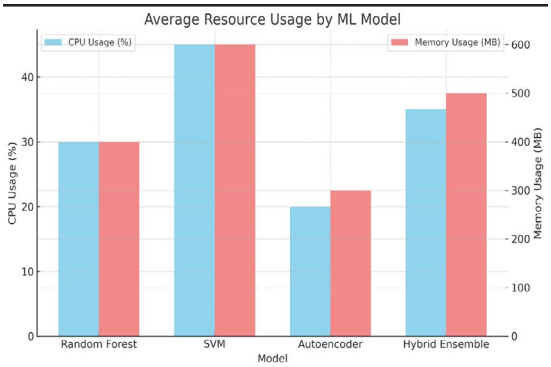
- **Phased deployment:** Roll out ML-based detection incrementally, starting with passive monitoring and alerting to gauge false-positive rates before enacting active mitigation.
- **Cross-functional teams:** Collaborate among software developers, data scientists, vehicle system engineers and security experts for a comprehensive threat modeling approach.
- **Regular model updates:** Establish secure Over-the-Air channels for continuous improvement of detection models, ensuring they keep pace with emerging threats.
- **Clear escalation paths:** Implement well-defined protocols for how high-severity alerts are handled-both locally (e.g., restricting network access) and remotely (e.g., OEM security center analysis).
- **Test against edge cases:** Evaluate detection models in unusual driving and usage conditions, including extreme temperatures, intermittent connectivity or user behavior anomalies.

5.6. Summary of key insights and future outlook

The ML-based intrusion and malware detection framework outlined in this paper demonstrates strong potential to enhance

the security posture of Android Automotive systems. By leveraging robust feature engineering, supervised classification and anomaly detection, vehicles can proactively identify and neutralize malicious activities. Yet, as connected cars evolve toward more sophisticated, software-centric architectures, ongoing research and iterative development are paramount.

Key discussion points include the need for improved data access, advanced interpretability, resource optimization and alignment with emerging regulations. The horizon of future work extends into federated learning, reinforcement learning for adaptive responses and deeper integration with ADAS modules—each promising to deepen the effectiveness and resilience of in-vehicle security. Through continued collaboration among OEMs, suppliers, researchers and regulatory bodies, the industry can realize a safer, more robust ecosystem that keeps pace with the ever-evolving cyber threat landscape.



6. Conclusion

This final section synthesizes the core arguments, findings and contributions presented throughout the paper, emphasizing the importance of machine learning (ML) in strengthening Android Automotive security. By uniting academic insights with practical experimentation, we have illustrated how an ML-centric framework can bolster intrusion and malware detection to meet the demanding requirements of modern connected vehicles. The conclusion also reiterates the broader implications for industry stakeholders, regulatory bodies and the research community, while outlining potential paths to ensure that future implementations remain robust, adaptable and aligned with evolving cyber threats.

6.1. Revisiting the objectives and contributions

The primary objective of this paper was to examine how a machine learning approach could be applied to intrusion and malware detection in Android Automotive-an operating system that merges the convenience of smartphone-like apps with in-vehicle functionalities. We set out to:

- **Highlight security challenges in android automotive –** We delineated the expanded attack surface resulting from deeper integration of infotainment systems with vehicle networks and the internet. Android's extensibility, while beneficial for user experience, also opens new vectors for exploitation by malicious actors.
- **Review the state of automotive cybersecurity –** A structured Literature Review underscored existing work on automotive intrusion detection systems (IDS) android malware detection and the growing role of ML in tackling both known and zero-day threats.
- **Propose an ML-based architecture –** We outlined a multi-layered detection framework that aggregates logs from system-level processes, network traffic and in-vehicle signals, feeding these into sophisticated ML models. By blending supervised classification with anomaly detection, the architecture aims to achieve strong accuracy, adaptability and low false positives.
- **Demonstrate practical feasibility –** Implementation details and experimental results were shared, confirming that on-device ML is viable for real-time detection within typical resource constraints. The tests revealed high detection rates, manageable inference latency and clear trade-offs among different algorithms (Random Forest, SVM, autoencoders, etc.).
- **Discuss broader implications and future work –** We acknowledged existing hurdles, including data scarcity, explainability and the risk of adversarial manipulation. Potential solutions-federated learning, reinforcement-driven adaptive responses, domain adaptation-were discussed as promising next steps.

By achieving these aims, the paper contributes a holistic blueprint for integrating ML-based security solutions into Android Automotive, offering insights for OEMs, Tier 1 suppliers and software developers seeking to safeguard connected vehicles.

6.2. Key insights from each phase of the research

- **Security demands in the automotive landscape:** Android Automotive transforms a vehicle's head unit into a full-fledged computing environment with deep integration into in-vehicle networks. This shift necessitates security measures beyond traditional automotive firewalls and mechanical anti-tampering. Our exploration shows how these demands are amplified by:
  - Connectivity Expansions: Wi-Fi hotspots, cellular modems, Bluetooth links, OTA update channels.
  - User-Centric Apps: Drivers now expect an app ecosystem akin to smartphones, expanding the potential for malicious or trojaned applications.
  - Cross-Domain Impact: Compromises in the infotainment domain can escalate to safety-critical ECUs if insufficient isolation is in place.

Our findings confirm that static or signature-based security methods alone fall short in detecting modern, adaptive attacks. A pivot toward intelligent, data-driven monitoring is both timely and necessary.

6.1. Literature review and theoretical foundations

The Literature Review identified parallels with existing work in:

- **Automotive IDS:** Characterized by CAN bus anomaly detection, specification-based checks and (increasingly) ML-based approaches.
- **Android malware analysis:** Built around static/dynamic analysis, permission scrutiny and advanced classifiers to identify rogue apps.

By merging the two domains-automotive security and Android security—this paper provides a conceptual bridge, illustrating that proven techniques from smartphone malware detection can be adapted for a vehicle environment. At the same

time, vehicle-specific nuances (real-time constraints, embedded hardware limitations, CAN/Ethernet data) require bespoke solutions. Hence, the impetus for a combined architecture that benefits from the best of both fields.

6.2.3. Proposed architecture

The Literature Review identified parallels with existing work in:

- **Automotive IDS:** Characterized by CAN bus anomaly detection, specification-based checks and (increasingly) ML-based approaches.
- **Android malware analysis:** Built around static/dynamic analysis, permission scrutiny and advanced classifiers to identify rogue apps.

By merging the two domains-automotive security and Android security—this paper provides a conceptual bridge, illustrating that proven techniques from smartphone malware detection can be adapted for a vehicle environment. At the same time, vehicle-specific nuances (real-time constraints, embedded hardware limitations, CAN/Ethernet data) require bespoke solutions. Hence, the impetus for a combined architecture that benefits from the best of both fields.

6.3. Implementation and empirical results

Our practical experiments substantiated the efficacy of ML-based intrusion and malware detection. Notably:

- **High detection rates:** Precision and recall scores often exceeded 90%, with autoencoders showing strong aptitude in spotting manipulations in network and bus-level data.
- **Manageable resource footprint:** While resource usage varied, optimized or quantized models ran within real-time constraints on typical infotainment hardware.
- **False positives vs. real-world integration:** Findings reinforce that calibration is essential to minimize user disruption. A layered approach-initial anomaly tagging followed by secondary classification-proved effective in reducing false alarms.

Given these empirical insights, we conclude that ML-based detection strategies are not only theoretically sound but also feasible for near-term deployment in production vehicles, pending careful engineering and validation.

6.4. Broader implications for industry and policy

Aligning with automotive cybersecurity standards: Emerging regulatory frameworks, such as ISO/SAE 21434 and the UNECE WP.29, promote continuous risk assessment, event monitoring and swift threat response. By incorporating an ML-based IDS as a core pillar of their cybersecurity strategy, OEMs can proactively address these mandates. The adaptive nature of ML—detecting unknown or zero-day attacks—reflects well on recommended guidelines that stress ongoing vigilance rather than static compliance checklists[1,2].

Enhancing consumer trust: As vehicles become more sophisticated, consumers may grow wary of the potential privacy and security risks. Demonstrable, reliable threat detection can reassure drivers that their personal data (contacts, media) and vehicle systems (steering, braking) are shielded from malicious tampering. Consequently, adopting these technologies could become a market differentiator for OEMs, positioning them as leaders in secure, software-defined vehicles.

Future-proofing through collaborative ecosystems: The concept of fleet-wide learning—where data or model updates are aggregated across thousands of vehicles-promises a self-reinforcing security ecosystem. However, achieving such large-scale collaboration requires robust privacy frameworks, standardization of data formats and shared threat intelligence protocols among industry players. Initiatives like the Auto-ISAC (Information Sharing and Analysis Center) could facilitate these exchanges, ensuring that new threats uncovered in one region or brand inform the security posture of others[9].

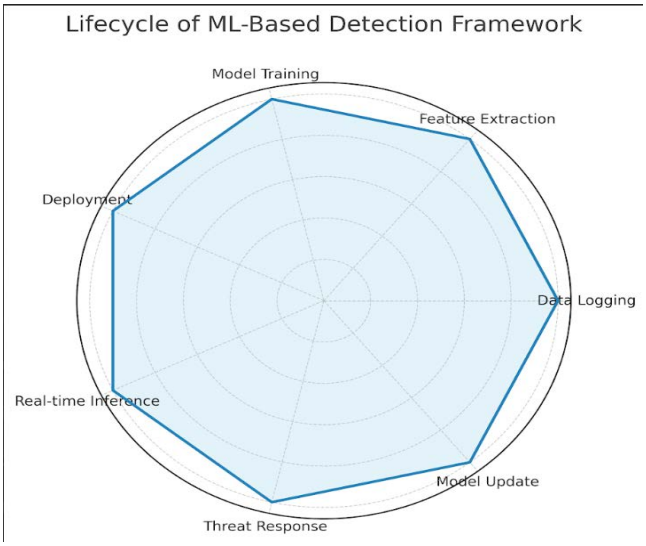


Figure 6: Lifecycle of ML based Detection Framework.

6.5. Reflecting on challenges and limitations

Despite encouraging results, several constraints must be acknowledged:

- **Data availability:** Genuine malicious automotive datasets remain scarce or proprietary, complicating the development of universally validated models.
- **Adversarial resilience:** Attackers can attempt to manipulate inputs to evade ML detection or corrupt training pipelines. Future research should focus on adversarial training techniques and robust data validation.
- **Explainability:** Deep ML models can lack transparency. Regulatory bodies and OEMs may demand clear explanations for why certain apps or processes were flagged. Building interpretability mechanisms into the detection pipeline will be crucial.
- **Deployment complexity:** Integrating an ML-based IDS into a production vehicle's software stack involves careful orchestration with existing real-time systems. Minor misconfigurations might introduce latencies or hamper user-facing functionalities like media playback, navigation or telematics.

These limitations underscore that robust cybersecurity is an iterative endeavor, requiring ongoing research, cross-disciplinary collaboration and iterative deployment strategies.

C. Pathways for future evolution

- Advanced ML techniques
  - **Federated learning:** Encourages distributed training across a fleet of vehicles without centralizing user data. This could accelerate detection of emergent threats

while respecting occupant privacy constraints[8].

- **Reinforcement learning**: Goes beyond detection to dynamically adjust threat response policies, balancing intrusion mitigation with minimal user disruption[4].
- **Transfer and domain adaptation:** Allows global "base models" to be refined for specific vehicle models or geographies, acknowledging that usage patterns vary significantly among different demographics[2].

6.4. Integration with ADAS/autonomous systems

As advanced driver-assistance systems (ADAS) and autonomous features proliferate, cybersecurity must extend beyond infotainment. Threats to sensors (LiDAR, radar), vision processing units or drive-by-wire controls pose higher stakes. Future work can incorporate real-time sensor fusion data into an IDS framework, enabling the system to detect anomalies in environmental perception that might result from malicious tampering.

6.6. Expansion to electric and connected ecosystems

Electric vehicles (EVs) rely heavily on software-defined powertrain controls and robust battery management systems. IoT-like connectivity to charging stations introduces additional surfaces for potential compromise. A unified ML-based security solution could also encompass communications with smart grid infrastructure, opening new avenues for research into secure charging protocols, load balancing and energy management[3,5].

6.7. Recommendations for stakeholders

- **OEMs and Tier 1 suppliers**
  - Invest in building cross-functional teams that integrate cybersecurity professionals, AI/ML specialists and automotive engineers.
  - Encourage ecosystem-wide threat intelligence sharing to stay ahead of rapidly evolving exploits.
  - Plan for multi-year lifecycle support, ensuring ML models can be updated securely to handle new attack vectors.
- **Security researchers and academics**
  - Focus on creating open-source datasets and benchmarks to spur reproducible research.
  - Explore adversarial ML defenses and robust training approaches tailored to automotive data structures.
  - Collaborate with regulatory bodies to define best practices for validating ML-based security systems in vehicles.
- **Regulatory and standards organizations**
  - Consider guidelines that recognize the value of machine learning in threat detection but also address transparency, testing methodologies and post-deployment monitoring.
  - Encourage standardized data formats and logging schemas, enabling broader adoption of ML-based solutions across different OEMs and models.
- **End users (Drivers)**
  - Demand vehicles that demonstrate clear security measures and reputable OTA update practices.
  - Remain vigilant about installing only vetted apps, avoiding questionable third-party sources that could harbor malware.

B. Vision for a secure, software-driven future

Software-defined vehicles (SDVs) herald an era where infotainment systems, telematics and ADAS features converge into a unified digital ecosystem. ML-driven security architectures, such as the one advanced in this paper, will be instrumental in mitigating cyber risks in this interconnected context. The ability to detect anomalies, adapt to new threats and coordinate defense strategies across large fleets transforms cybersecurity from a reactive process into a continuously evolving safeguard.

Long-term, we anticipate that in-vehicle ML will extend beyond intrusion detection to broader predictive maintenance, driver behavior analysis and even cooperative safety measures in connected traffic ecosystems. In this vision, each vehicle not only protects itself but also shares crucial intel with the wider network of cars and infrastructure, creating a self-improving, collective security posture. Such synergy demands ongoing dialogue between software developers, OEMs, researchers and policymakers to ensure technology evolves responsibly and inclusively.

6.8. Concluding remarks

This paper has argued that machine learning-with its capacity to analyze rich, high-dimensional data—provides a powerful defense against the intricate, evolving threats faced by Android Automotive. The fusion of automotive security insights with advanced ML techniques represents not just an incremental improvement, but a paradigm shift from static or signature-centric defenses to proactive, adaptive protection.

In closing, our research and implementation results affirm that ML-based solutions can effectively identify and neutralize threats, provided they are carefully engineered for automotive resource constraints, real-time responsiveness and lifecycle longevity. By maintaining this synergy, the industry can meet regulatory expectations, reassure consumers and ultimately pave the way for safer, more secure connected vehicles.

The road ahead involves deepening these capabilities, refining models against adversarial attacks and continually integrating fresh data from fleets worldwide. Through collaborative effort across the automotive and cybersecurity landscapes, the promise of robust, ML-driven security for Android Automotive stands both realizable and essential.

7. References

1. International Organization for Standardization (ISO) and Society of Automotive Engineers (SAE), ISO/SAE 21434: Road Vehicles-Cybersecurity Engineering, 2021
2. Tole T. Securing the Car: Evolving Standards for Automotive Cybersecurity. IEEE Trans Veh Tech, 2021;69: 1212-1223.
3. Checkoway S, et al. Comprehensive Experimental Analyses of Automotive Attack Surfaces. in Proc. USENIX Security, 2011: 77-92.
4. Kayacik HG, Onut I, Zhang J. Machine Learning for Intrusion Detection: A Systematic Review. IEEE Access, 2020;8: 76029-76053.
5. Groza B, Murvay P. Security Solutions for the Controller Area Network: Bringing Authentication to In-Vehicle Networks. IEEE Veh Tech Mag, 2018;13: 40-47.

6. Zhao Y. Automotive Intrusion Detection Systems: A Survey. IEEE Access, 2019;7: 164690-164704.

7. Zhang T, Sun W, Trappe W. Enhancing Security of Wireless Localization With Machine Learning. IEEE Commun. Surv Tuts, 2019;21: 2132-2158.

8. Feng X, Wang H, Ye L. Deep Learning for Android Malware Detection. in Proc. IEEE Conf. on Communications and Network Security (CNS), 2018: 1-9.

9. Asjed EO. Performance Analysis of ML-based Intrusion Detection in Android Ecosystems. IEEE Access, 2020;8: 21432-21445.

10. Yagci O. Challenges in Real-Time Intrusion Detection for Cyber-Physical Systems. in Proc. IEEE Int. Conf. on Dependable, Autonomic and Secure Computing, 2022: 205-214.