information

MDPI

Review

Formal Methods and Validation Techniques for Ensuring Automotive Systems Security

Moez Krichen [1,2]

[1] Department of Information Technology, Faculty of Computer Science and Information Technology, Al-Baha University, Al-Baha 65528, Saudi Arabia; moez.krichen@redcad.org or moez.krichen@ieee.org
[2] ReDCAD Laboratory, University of Sfax, Sfax 3038, Tunisia

Abstract: The increasing complexity and connectivity of automotive systems have raised concerns about their vulnerability to security breaches. As a result, the integration of formal methods and validation techniques has become crucial in ensuring the security of automotive systems. This survey research paper aims to provide a comprehensive overview of the current state-of-the-art formal methods and validation techniques employed in the automotive industry for system security. The paper begins by discussing the challenges associated with automotive system security and the potential consequences of security breaches. Then, it explores various formal methods, such as model checking, theorem proving, and abstract interpretation, which have been widely used to analyze and verify the security properties of automotive systems. Additionally, the survey highlights the validation techniques employed to ensure the effectiveness of security measures, including penetration testing, fault injection, and fuzz testing. Furthermore, the paper examines the integration of formal methods and validation techniques within the automotive development lifecycle, including requirements engineering, design, implementation, and testing phases. It discusses the benefits and limitations of these approaches, considering factors such as scalability, efficiency, and applicability to real-world automotive systems. Through an extensive review of relevant literature and case studies, this survey provides insights into the current research trends, challenges, and open research questions in the field of formal methods and validation techniques for automotive system security. The findings of this survey can serve as a valuable resource for researchers, practitioners, and policymakers involved in the design, development, and evaluation of secure automotive systems.

Keywords: formal methods; validation techniques; automotive systems; security; survey; challenges; integration; research rrends

Citation: Krichen, M. Formal Methods and Validation Techniques for Ensuring Automotive Systems Security . *Information* **2023**, *14*, 666. https://doi.org/10.3390/info14120666

Academic Editors: Giedre Sabaliauskaite, Jeremy Bryans and Farhan Ahmad

Received: 13 November 2023
Revised: 9 December 2023
Accepted: 14 December 2023
Published: 18 December 2023

1. Introduction

The fast advancement of vehicle technology has heralded a new era of extraordinary connectivity and sophistication. Modern vehicles are outfitted with elaborate electronic control units, integrated networks, and a plethora of sensors, enabling advanced functions ranging from self-driving to sophisticated infotainment systems [1–4]. While this boom in technological proficiency has numerous advantages, it also raises an important concern: the vulnerability of automobile systems to security breaches. Vehicles are vulnerable to a variety of cyber risks due to the integration of complicated software and connection elements, which range from remote hacking to illegal access [5,6]. Such security breaches can have a wide range of effects, from compromised personal data to jeopardizing passenger safety [7,8]. As the automotive industry accelerates toward a future of self-driving and connected vehicles, protecting these systems from malicious attacks becomes critical [9,10].

Because of the increasing complexity and interconnection of automobile systems, traditional security methods are no longer enough [11,12]. Traditional techniques, which are primarily focused on perimeter protection, struggle to keep up with cyber adversaries' shifting tactics. This highlights the importance of a more thorough and stringent approach to

automotive system security. Formal methods and validation approaches offer an appealing solution to this growing issue [13–15]. Formal approaches provide a structured framework for expressing and verifying security properties by utilizing mathematical rigor [14,16]. Validation techniques, such as practical assessments and simulated attack scenarios, supplement formal methods by validating security measures in the actual world [17,18].

The primary goal of this research study is to present a thorough overview of the most recent state-of-the-art formal methodologies and validation approaches used in the automotive sector to ensure system security. This includes an in-depth discussion of validation techniques such as penetration testing and fault injection, as well as an exploration of various formal methods spanning from model checking to theorem proving [19,20].

This document covers the complete breadth of automotive system development, from requirements engineering to testing. We hope to emphasize the integration points where formal methods and validation techniques play a critical role in bolstering system security by going into each phase of the development lifecycle. This survey also considers developments in the convergence of formal methods, machine learning, and blockchain technology, providing a view into the future of automotive system security.

This study seeks to be a significant resource for researchers, practitioners, and policymakers interested in the design, development, and evaluation of secure automotive systems by conducting an exhaustive survey of the relevant literature and conducting insightful analysis. The structure of the paper is illustrated in Figure 1.

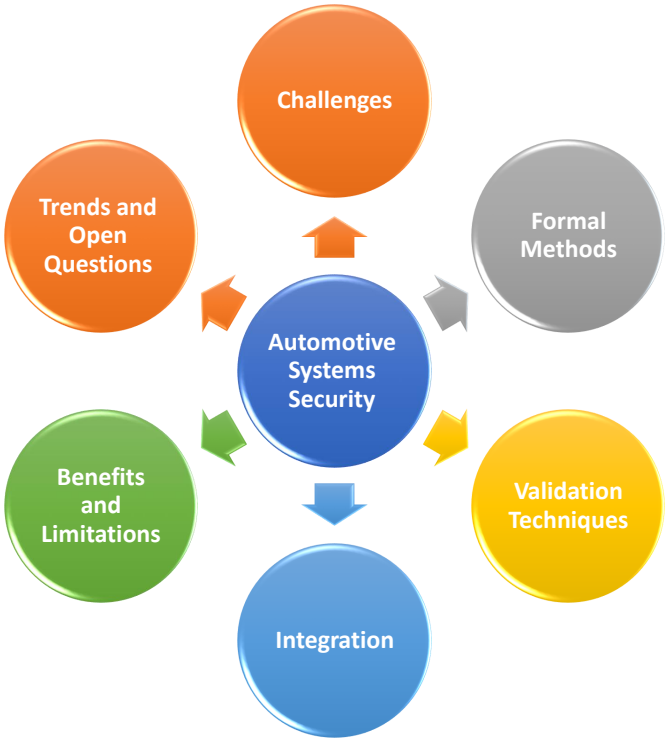

Figure 1. Structure of the survey paper.

Related Surveys:

The work described in [21] provides a comprehensive analysis of existing solutions and key challenges in securing intelligent transportation systems (ITSs). It addresses the security issues associated with the growing connectivity and autonomy of vehicles in ITSs. The work includes a tutorial on security issues and attacks, evaluates existing solutions, and discusses recent trends for enhancing the security of ITSs. Overall, it aims to improve the safety and privacy of ITSs by identifying potential threats and highlighting areas for improvement in securing these systems.

The authors of the paper [22] proposed an overview and comparison of protocols and communication patterns for automotive architectures based on the service-oriented architecture (SOA) paradigm, as opposed to the traditional signal-oriented approaches. The article highlights the need for a transformed architecture design to meet new requirements from customers and manufacturers, which include adding new software functions during the product life cycle. It discusses the challenges and opportunities of SOAs in terms of information security and explores various security countermeasures such as firewalls, intrusion detection systems (IDSs), and identity and access management (IAM) within the automotive context. The work concludes with a discussion on the adaptation of existing security measures and their specific features in a hybrid architecture that combines both signal-oriented and service-oriented approaches.

The primary contribution of the paper [23] is the provision of a comprehensive overview of threat analysis and risk assessment (TARA) as it pertains to connected vehicle cybersecurity in the automotive industry. The paper emphasizes the potential for accidents to result from cybersecurity vulnerabilities in view of the growing interdependence with external networks, thereby highlighting the significance of cybersecurity. The text elucidates that automobile manufacturers are increasing their investments in the development of cybersecurity defense mechanisms and proposes TARA as an effective approach to cost reduction and guarantee efficient defense during the initial phases of vehicle development. A classification of various TARA methodologies is proposed, and existing approaches are compared. An efficacy evaluation of commonly used tools for TARA is also conducted. Furthermore, the article presents the notion of attack–defense mapping, an approach that seeks to synchronize detected vulnerabilities and threats with the most suitable countermeasures. The paper concludes with a discussion of potential future developments of TARA within the automotive industry.

The study [24] contributes to a thorough analysis of over-the-air (OTA) software upgrades in the automotive industry, with a particular emphasis on security issues. The report emphasizes how important over-the-air (OTA) updates are becoming for connected cars, and how they can be used to remotely improve features and fix software faults. The paper attempts to increase awareness in this domain by offering a thorough review of various research areas and techniques in OTA update technologies. It provides a comparative analysis of current methodologies, examines viable and secure OTA update approaches, and talks about the relationship between connected car technologies and OTA update features. Customer happiness, usability characteristics, and car features supporting over-the-air (OTA) upgrades are also examined in the poll. The work offers important insights for the development of OTA updates in cars by pointing out future research directions, especially in the area of security.

A thorough overview of cybersecurity in the context of connected and autonomous vehicles (CAVs) is the primary contribution of the paper [25]. The study recognizes the potential advantages of CAVs, including better mobility options, lower user prices, safer transportation, and the creation of new jobs. It also draws attention to the growing threat of malicious assaults to CAV security as automation and connectivity rise. Based on communication networks and assault objects, the survey divides cybersecurity threats and vulnerabilities into three categories: vehicle-to-everything, in-vehicle, and other attacks. It also addresses defense tactics for safeguarding CAVs and talks about cyber risk as a type of attack in the CAV environment. The available cybersecurity and safety requirements for CAVs are presented in the paper's conclusion, along with some useful advice. Lastly, it highlights issues and unresolved issues that call for more study in the area of CAV cybersecurity.

The construction of an extensive attack taxonomy for the automobile area is the primary contribution of the work [26]. The paper focuses on how the automotive industry has changed over time and how the addition of electronic components to cars has resulted in new attack vectors and vulnerabilities. One area of research and practice that is lacking is a thorough attack taxonomy for the automotive industry. In order to solve

this, the authors carry out a thorough review of the literature, finding and categorizing 48 distinct attacks using the suggested taxonomy of attack methods. By connecting several attack vectors together, the taxonomy can help penetration testers in the automobile industry create more complex attacks. It is a useful tool for penetration testers. Five dimensions—AUTOSAR layers, attack domains, information security principles, attack surfaces, and attacker profiles—are also used to categorize the discovered attack vectors. The findings draw attention to the most often-used attack vectors in the literature, which are primarily directed against the AUTOSAR architecture's application and service layers. These vectors include GPS spoofing, message injection, node impersonation, sybil, and wormhole attacks.

The study outlined in [27] contributes to a thorough assessment investigating trust management within the Internet of Vehicles (IoV) ecosystem. IoV is acknowledged in the article as a way to improve user experience through the provision of advanced services that put comfort and safety first. The authors draw attention to how the architecture of the Internet of Vehicles (IoV) is multifaceted, involving human beings, roadside objects, cars, and intelligent transportation systems (ITSs). They stress that the various forms of communication within the Internet of Vehicles (IoV) create new security needs and increase the ecosystem's attack surface. The study emphasizes the use of trust management as a security technique to improve reliability in the Internet of Vehicles (IoV) environment in order to overcome these issues. Although trust management in the context of vehicular ad hoc networks (VANETs) has been thoroughly explored, the authors contend that techniques created for VANETs must be modified and expanded in order to be useful in the larger Internet of Vehicles environment. They draw attention to how cutting-edge technologies like artificial intelligence, cloud computing, blockchain, and software-defined networking (SDN) may offer more appropriate and practical trust management strategies in the context of the Internet of Vehicles.

The principal contribution of the study [28] is a systematic review and analysis of previously conducted studies on autonomous vehicle defenses and attacks. The authors underscore the significance of intelligent transportation within smart cities and the possible susceptibilities of autonomous vehicles that may have repercussions on human life and security. Through a comprehensive examination of 151 papers published from 2008 to 2019, the survey investigates autonomous attacks and defense mechanisms in depth. The text delineates three distinct categories for attacks and defense mechanisms, emphasizing the significance of machine learning and artificial intelligence in the identification of anomalies. The survey provides significant findings and ramifications for subsequent investigations, with a particular focus on the application of artificial intelligence to tackle the security concerns associated with autonomous vehicles in smart city environments.

The work presented in [29] is a thorough survey that examines the current attacks and defense techniques used for connected and autonomous vehicles (CAVs). The authors emphasize the significant influence of autonomous vehicles on intelligent transportation systems and the potential advantages of CAVs in attaining secure and efficient transportation. Nevertheless, it is important to highlight the considerable security obstacles that connected and autonomous vehicles (CAVs) encounter as a result of their interconnected nature and the susceptibility of their components to potential attacks. The survey examines 189 papers published between 2000 and 2020, specifically focusing on 131 papers that discuss attack models or defense strategies for connected and autonomous vehicles (CAVs). This study offers a comprehensive examination of security attacks and countermeasures for connected and autonomous vehicles (CAVs). It explores various attack models based on targeted components, access requirements, and motives. Additionally, it identifies current research challenges and trends in this field. The survey reveals a gap between academic research and industry implementation of security issues related to connected and autonomous vehicles (CAVs). This underscores the importance of enhancing defenses in future CAV development. The survey enhances the understanding of the current status

and trends in CAV security, offering valuable insights for researchers and engineers aiming to improve CAV security.

The study [30] presents a comprehensive study that investigates the security and privacy concerns associated with vehicular cloud computing (VCC). The prospective advantages of VCC in transforming computing services for intelligent transportation, autonomous driving, and other diverse applications are duly recognized by the authors. Nonetheless, they acknowledge that VCC is constrained by substantial privacy and security concerns. The article presents an up-to-date examination of the architecture, functionalities, and implementations of VCC. The paper executes an exhaustive survey of security and privacy concerns in VCC, encompassing various layers including the physical resource layer, vehicle-to-anything (V2X) network layer, vehicular cloud layer, and the entire system level. It also provides a taxonomy for threat identification. The research emphasizes areas of concern and unresolved matters, providing significant insights that can guide future investigations into the security and privacy challenges associated with VCC. In its entirety, this study contributes to the body of knowledge by furnishing a thorough comprehension of the security and privacy concerns that are unique to VCC. This, in turn, aids in the formulation of efficacious measures to bolster the security and privacy of VCC systems.

The survey presented in this paper distinguishes itself from existing surveys by providing a comprehensive overview of formal methods and validation techniques for automotive system security, covering a wide range of approaches and their integration within the development lifecycle. Unlike other surveys that focus on specific aspects such as protocols, threats, or attacks, our paper takes a holistic approach, examining the entire spectrum of formal methods and validation techniques, their application in different development phases, and identifying current research trends and challenges. This comprehensive perspective makes our survey a valuable resource for researchers, practitioners, and policymakers involved in designing and developing secure automotive systems. Table 1 provides a comprehensive comparison of the 10 survey papers in the field of automotive system security, evaluating their main contributions, limitations, and the year of publication, in relation to the proposed survey presented in this paper.

Table 1. Comparison of survey papers.

Survey Paper	Year	Main Contribution	Limitations
Lamssaggad et al. [21]	2021	Analysis of existing solutions and challenges in securing intelligent transportation systems (ITS)	Limited focus on ITS, may not cover all aspects of automotive system security
Rumez et al. [22]	2020	Overview and comparison of protocols and communication patterns for automotive architectures	Does not explore formal methods or validation techniques for security
Luo et al. [23]	2021	Comprehensive overview of threat analysis and risk assessment (TARA) for connected vehicles	Primarily focuses on threat analysis, does not cover formal methods extensively
Halder et al. [24]	2020	Thorough analysis of over-the-air (OTA) software upgrades in the automotive industry	Limited to OTA software upgrades, does not cover other security aspects
Sun et al. [25]	2021	Overview of cybersecurity in the context of connected and autonomous vehicles (CAVs)	Primarily focuses on CAVs, may not cover all aspects of automotive system security
Pekaric et al. [26]	2021	Construction of an extensive attack taxonomy for the automotive industry	Limited to attack taxonomy, does not explore formal methods or validation techniques

Table 1. *Cont.*

Survey Paper	Year	Main Contribution	Limitations
Hbaieb et al. [27]	2022	Assessment of trust management within the Internet of Vehicles (IoV) ecosystem	Focuses on trust management, does not extensively cover formal methods or validation techniques
Kim et al. [28]	2021	Systematic review and analysis of autonomous vehicle defenses and attacks	Primarily focuses on autonomous vehicles, may not cover all aspects of automotive system security
Pham et al. [29]	2021	Examination of attacks and defense techniques for connected and autonomous vehicles (CAVs)	Primarily focuses on CAVs, may not cover all aspects of automotive system security
Masood et al. [30]	2020	Study of security and privacy concerns associated with vehicular cloud computing (VCC)	Limited to vehicular cloud computing, does not cover other security aspects
Our Paper	2023	Comprehensive overview of formal methods and validation techniques for automotive system security	N/A

2. Challenges in Automotive System Security

The field of automotive systems encounters a multitude of obstacles in the realm of security assurance. The growing intricacy and interconnectivity of these systems have given rise to novel vulnerabilities and risks that have the potential to result in security breaches [31,32]. Comprehending and effectively mitigating these problems is of paramount importance for ensuring the protection of automotive systems against potential security breaches [33,34]. This section delves into the primary obstacles linked to security inside automobile systems.

2.1. Types of Networks and Communication Protocols in Automotive Systems

Automotive systems rely on various networks and communication protocols to enable the exchange of information between different components and subsystems. Understanding the types of networks and communication protocols used in automotive systems is crucial for ensuring efficient and secure communication within the vehicle. Here are some common types of networks used in automotive systems:

- Controller Area Network (CAN): CAN is a widely used network in vehicles that facilitates communication between electronic control units (ECUs). It was initially developed in the 1980s and has since become the de facto standard for in-vehicle communication. CAN is a robust, low-cost, and fault-tolerant network that supports real-time applications. It operates on a bus topology, where multiple ECUs are connected to a shared communication bus. CAN allows for reliable and efficient communication between various vehicle systems, such as the engine control unit, transmission control unit, and body control module.
- Local Interconnect Network (LIN): LIN is another network commonly found in automotive systems, primarily used for communication between less critical components. It provides a cost-effective solution for low-speed communication requirements, such as controlling window switches, door locks, and interior lighting. LIN operates on a master–slave architecture, where a master node communicates with multiple slave nodes. Compared with CAN, LIN has a lower bandwidth and is designed for simpler and less time-critical applications.
- Ethernet: Ethernet is increasingly being adopted in modern vehicles due to its high bandwidth capabilities. It enables communication between various ECUs and sup-

ports advanced applications such as infotainment systems, advanced driver-assistance systems (ADAS), and autonomous driving. Automotive ethernet is based on the ethernet standard but includes additional features to meet the specific requirements of automotive applications. It offers higher data rates, improved reliability, and the ability to prioritize different types of traffic.

- FlexRay: FlexRay is a deterministic, fault-tolerant network that provides high-speed communication in safety-critical automotive systems. It was developed to meet the stringent requirements of advanced driver-assistance systems and x-by-wire applications. FlexRay supports both time-triggered and event-triggered communication, allowing for precise and predictable transmission of data. It offers high bandwidth, fault tolerance, and synchronization capabilities, making it suitable for critical applications that require real-time communication.
- Media-Oriented Systems Transport (MOST): MOST is a network technology primarily used in automotive multimedia and infotainment systems. It enables the transmission of audio, video, and control data between different multimedia devices in the vehicle, such as head units, amplifiers, and displays. MOST supports high-speed data transfer and provides features like synchronous streaming, network management, and fault tolerance.
- Automotive Ethernet: Automotive ethernet is an extension of the ethernet standard specifically designed for automotive applications. It provides high-speed communication and supports the increasing bandwidth requirements of modern vehicles. Automotive ethernet enables the integration of various systems, such as infotainment, ADAS, and vehicle diagnostics, over a single network infrastructure. It utilizes ethernet protocols and technologies, such as Ethernet AVB (audio video bridging) and TSN (time-sensitive networking), to ensure reliable and deterministic communication.
- Wireless Networks: Wireless networks play an essential role in automotive systems, enabling connectivity with external devices and services. For example, Bluetooth is commonly used for hands-free calling, audio streaming, and wireless device connectivity. Wi-Fi can provide in-vehicle internet access, allowing passengers to connect their devices and access online services. Cellular networks, such as 4G LTE and 5G, enable vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication, supporting features like remote diagnostics, over-the-air updates, and connected services.
- CAN-FD: CAN with Flexible Data-Rate (CAN-FD) is an extension of the traditional CAN protocol that allows for higher data rates and increased payload sizes. It addresses the growing demand for higher bandwidth in automotive systems, particularly for applications that require more extensive data transmission, such as high-resolution sensor data from cameras and radars. CAN-FD maintains backward compatibility with existing CAN networks, enabling a smooth transition to higher data rates.
- LIN Sub-bus: LIN Sub-bus is an extension of the LIN protocol that allows for the expansion of LIN networks to accommodate more devices. It enables the connection of additional slave nodes to an existing LIN bus, increasing the overall capacity and flexibility of the network. LIN Sub-bus is commonly used in automotive systems where the number of components exceeds the capacity of a single LIN bus, such as in complex door modules or instrument clusters.

These networks and communication protocols contribute to the diverse communication landscape in automotive systems. The selection and integration of these networks depend on factors such as the specific vehicle architecture, communication requirements, and the desired functionality and features of the vehicle. By utilizing these networks and protocols effectively, automotive systems can achieve efficient and reliable communication, enabling the seamless operation of various subsystems and enhancing the overall driving experience.

2.2. Security Challenges

Contemporary vehicle systems mainly depend on software and network connectivity, rendering them subject to a multitude of vulnerabilities. The presence of these vulnerabilities may stem from various factors, including design deficiencies, faults in implementation, insufficient security protocols, or the utilization of obsolete software components. Addressing these security challenges is crucial to protect against potential cyber threats and safeguard the safety and privacy of vehicle occupants. Here are some of the key security challenges associated with automotive network protocols:

- Insecure Communication Channels [35,36]: Insufficient encryption or authentication procedures implemented in communication protocols possess the potential to compromise the confidentiality of sensitive data and facilitate illegal entry into vehicle functionalities. Attackers may intercept or manipulate communication messages, leading to unauthorized access or unauthorized control of critical vehicle systems.
- Weak Authentication and Authorization [37,38]: Insufficient or inadequately executed authentication and authorization measures have the potential to grant unauthorized access to vital vehicle systems, allowing attackers to assume control without proper authority. Weak authentication mechanisms may enable unauthorized individuals to bypass security barriers and gain unauthorized access to vehicle functions or sensitive data.
- Software Vulnerabilities [39,40]: The exploitation of vulnerabilities in software components, such as operating systems, infotainment systems, or car firmware, has the potential to influence or interrupt the functionality of vehicles. Software vulnerabilities can be exploited by attackers to gain control over critical systems, disrupt vehicle operations, or compromise the safety of vehicle occupants.
- Inadequate Secure Coding Practices [41–43]: The absence of compliance with secure coding principles in the process of software development has the potential to create several vulnerabilities, including but not limited to buffer overflows, SQL injection, and code injection attacks. Inadequate secure coding practices increase the risk of software vulnerabilities, which can be exploited by attackers to gain unauthorized access or execute malicious code within the vehicle's systems.
- Physical Access Exploitation [44,45]: The security of the entire system can be compromised by individuals who have physical access to the vehicle and exploit flaws in diagnostic interfaces, onboard systems, or tamper-proofing features. Attackers with physical access can manipulate or tamper with the vehicle's components, compromising the integrity and functionality of critical systems.
- CAN Bus Security: The widespread use of the controller area network (CAN) in automotive systems makes it an attractive target for cyber attacks. CAN lacks built-in security features, making it vulnerable to various threats, including message spoofing, replay attacks, and unauthorized access. Attackers can manipulate CAN messages to compromise critical vehicle functions, such as braking or steering. Securing the CAN bus requires implementing authentication, encryption, and intrusion detection mechanisms to prevent unauthorized access and ensure the integrity and confidentiality of the communication.
- Ethernet Security: Ethernet is increasingly being adopted in vehicles, especially for advanced applications like infotainment and ADAS. However, ethernet networks face security challenges similar to those in traditional IT networks. These include the risk of unauthorized access, data tampering, and denial-of-service attacks. Securing ethernet in automotive systems involves implementing robust access control mechanisms, encryption protocols, and network segmentation to isolate critical systems from non-critical ones.
- Wireless Network Security: Wireless networks, such as Bluetooth, Wi-Fi, and cellular networks, are susceptible to various security threats. For example, Bluetooth connections can be vulnerable to eavesdropping and unauthorized device pairing. Wi-Fi networks in vehicles may be targeted by attackers attempting to gain unau-

thorized access to the vehicle's systems or steal sensitive data. Cellular networks can be exploited for remote attacks, such as compromising the vehicle's telematics or infotainment systems. Securing wireless networks requires implementing strong authentication, encryption, and intrusion detection mechanisms to protect against unauthorized access and data breaches.

- FlexRay and LIN Security: While FlexRay and LIN are less commonly targeted by external attackers due to their limited external connectivity, they still face security challenges. These protocols may be vulnerable to physical attacks, such as tampering with the communication wires or injecting malicious signals. Securing FlexRay and LIN networks involves implementing physical security measures, such as tamper-resistant wiring and secure connectors, to prevent unauthorized access and tampering.
- Secure Software Updates: Many automotive systems rely on software updates to fix vulnerabilities and introduce new features. However, the process of updating software over the network introduces security risks. Attackers may attempt to exploit vulnerabilities in the update process to inject malicious code or tamper with the software. Secure software update mechanisms, such as code signing, secure boot, and secure update protocols, are essential to ensure the integrity and authenticity of software updates and prevent unauthorized modifications.

Addressing these security challenges requires a comprehensive approach that encompasses secure network design, robust encryption, strong authentication mechanisms, secure software development practices, physical security measures, and secure software update mechanisms. Regular security assessments, vulnerability testing, and continuous monitoring are essential to identify and mitigate potential security risks in automotive network protocols. By addressing these challenges, automotive systems can enhance the security of their communication networks, protect against cyber threats, and ensure the safety and privacy of vehicle occupants.

2.3. Other Possible Risks

The exploitation of these vulnerabilities can result in significant ramifications, such as the illegal manipulation of vehicle functions, compromise of crucial systems, and jeopardization of the safety of both occupants and other individuals on the road. Illustrative instances of probable ramifications encompass:

- Unauthorized Remote Control [46,47]: The unauthorized acquisition of control over various vehicle operations, including steering, braking, and acceleration, by malicious individuals can give rise to substantial safety hazards for both the occupants of the vehicle and other individuals on the road.
- Data Privacy Breaches [48,49]: Security breaches in automobile systems can lead to the unlawful acquisition, theft, or alteration of confidential information, such as personal data, location data, or patterns of vehicle usage.
- Malware Injection [50,51]: The injection of malware by malicious actors into automobile systems has the potential to disrupt operations, undermine safety-critical functions, or facilitate illegal observation and tracking.
- Physical Safety Risks [52,53]: Security breaches have the potential to give rise to physical safety hazards, such the deactivation of safety features, manipulation of airbag deployment, or interference with the anti-lock braking system (ABS), thereby leading to accidents or injuries.

2.4. Limitations of Classical Techniques

The examples listed above underscore the importance of resolving vulnerabilities and guaranteeing the security of automobile systems in order to proactively prevent potential accidents and effectively reduce associated risks. While classical techniques have been employed in the field of automotive system security, they have certain limitations that make them less effective in addressing the evolving threats and vulnerabilities. Here, we

discuss some of the classical techniques commonly used in the automotive industry and their main limitations:

- Firewalls and Intrusion Detection Systems (IDS) [54–57]: Firewalls establish a barrier between internal and external networks, while IDS monitor network traffic for suspicious activities. However, these techniques have limitations in the context of automotive systems. They may not be able to detect attacks that exploit vulnerabilities within the vehicle's internal network, such as compromised components communicating with each other. Additionally, they often struggle to keep up with the increasing complexity and sophistication of attacks, as attackers continuously find new ways to bypass or circumvent traditional network security measures.
- Encryption and Secure Communication Protocols [58–62]: Encryption and secure communication protocols are essential for protecting sensitive data transmitted between different components of an automotive system. These techniques ensure confidentiality and integrity of the communication. However, encryption alone cannot prevent attacks that exploit other vulnerabilities in the system. Moreover, the key management and distribution mechanisms in automotive systems can be challenging to implement securely, and if these mechanisms are compromised, the effectiveness of encryption can be severely undermined.
- Access Control and Authentication Mechanisms [63–66]: Access control and authentication mechanisms are used to restrict access to critical functions and resources within an automotive system. These techniques help prevent unauthorized control and manipulation of the vehicle. However, they rely on the assumption that the authentication mechanisms themselves are secure. Weak or improperly implemented authentication mechanisms can be exploited by attackers to gain unauthorized access. Furthermore, in complex automotive systems with numerous interconnected components, managing and enforcing access control policies can become increasingly challenging.
- Secure Software Development Practices [67–70]: Secure software development practices, such as secure coding guidelines and vulnerability scanning, are crucial for building robust and resilient automotive systems. These practices aim to eliminate common software vulnerabilities and reduce the attack surface. However, they cannot guarantee the absence of all vulnerabilities, especially in complex systems with numerous software components and interactions. Additionally, the incorporation of secure software development practices requires significant effort and expertise, and it can be challenging to enforce them consistently across all stages of the development process.
- Physical Security Measures [71–74]: Physical security measures, such as tamper-proofing mechanisms and secure diagnostic interfaces, are employed to protect automotive systems from physical access attacks. While these measures are important, they may not be sufficient to defend against sophisticated attackers with physical access to the vehicle. Determined attackers can bypass or manipulate physical security measures given enough time and resources. Additionally, physical security measures cannot address vulnerabilities that arise from software or network-based attacks, which are becoming increasingly prevalent in modern automotive systems.

It is important to recognize the limitations of classical techniques and explore more advanced and comprehensive approaches to automotive system security. In the following sections, we will delve into the use of formal methods, validation techniques, and other emerging strategies to address these limitations and enhance the security of automotive systems.

3. Formal Methods for Analyzing Automotive System Security

In order to guarantee the safety and security of automotive systems, formal methods have developed as potent tools for the analysis and verification of their security properties. This section examines many notable formal methodologies employed within the domain of automobile system security [75–77]. As illustrated in Figure 2, they enable system designers to specify the system's behavior and properties using precise mathematical notations like logic formulas and state machines. Formal techniques can then employ

automated tools to examine the system's behavior and attributes, such as consistency, completeness, and correctness.

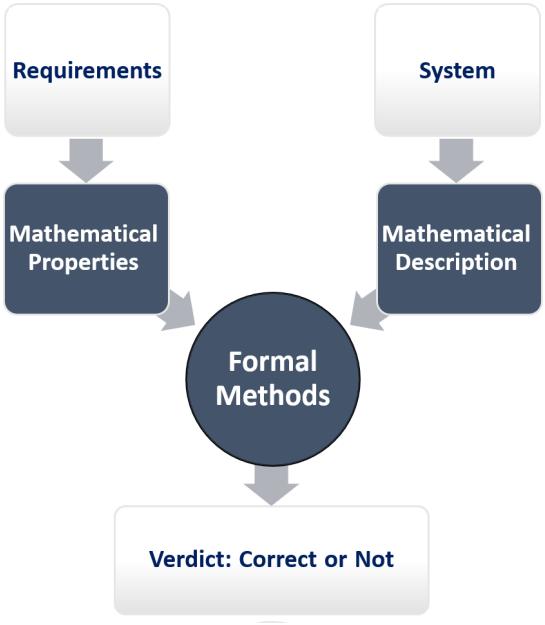

Figure 2. A simplified diagram illustrating how formal methods function.

3.1. Model Checking

Model checking is a rigorous technique that entails systematically examining a finite-state model of a system in order to ascertain its compliance with a specified property or specification [78–81]. In the realm of automotive system security, the application of model checking can prove to be a valuable approach for examining the security-related elements of a given system [82–85].

Formally, let $M = (S, I, T, AP, L)$ be a finite-state model, where:

- S is a finite set of states representing the possible configurations of the system.
- $I \subseteq S$ is the set of initial states from which the system execution begins.
- $T \subseteq S \times S$ is the transition relation that specifies the possible state transitions of the system.
- AP is a set of atomic propositions representing the properties of interest that can hold in a state.
- $L : S \rightarrow 2^{AP}$ is a labeling function that associates each state with the set of atomic propositions satisfied in that state.

The model checking process involves verifying whether a given property φ holds in the model M. The property φ is expressed using a temporal logic formalism, such as linear temporal logic (LTL) or computation tree logic (CTL). The verification is performed by exhaustively exploring the state space of the model and checking whether the property holds in each state and transition.

For example, consider an automotive system that incorporates a cryptographic protocol to ensure secure communication between different components. Let $M = (S, I, T, AP, L)$ represent the finite-state model of the system, where S represents the possible states of the protocol execution, I denotes the initial states, T specifies the transitions between states, AP includes atomic propositions related to security properties, and L is the labeling function that associates states with the satisfied atomic propositions.

To check a security property φ in the model M, we use a temporal logic formula, such as an LTL formula. For instance, φ can express the property "confidentiality holds in all protocol executions", which can be formalized as $\Box(\text{Confidentiality})$. The model checking process then systematically explores all possible protocol executions, checking whether the

formula φ holds in each state and transition. If a violation is found, the model checking process provides a counter example, such as a sequence of protocol messages that leads to a security vulnerability, such as a replay attack or a key compromise.

Through a methodical exploration of all feasible states within a system, the process of model checking can effectively detect vulnerabilities, ascertain potential avenues for attacks, and validate the integrity of security features. This technique proves to be highly advantageous in identifying intricate security vulnerabilities that can emerge as a result of intricate interactions inside a system.

3.2. Theorem Proving

The process of theorem proving is a formal methodology that is grounded in the principles of mathematical logic and proof theory in order to ascertain the accuracy and validity of a given system [86–89]. It involves constructing a formal proof that establishes the truth of a proposition or verifies the validity of a statement based on a set of axioms, rules of inference, and logical reasoning [90–93].

Formally, let Γ be a set of axioms and P be a proposition or statement. The theorem-proving process aims to demonstrate that P holds true based on the axioms and logical deductions from them. This is typically achieved by constructing a formal proof, which is a sequence of logical steps that establish the truth of P using valid inference rules. The proof may involve applying logical rules, such as modus ponens or universal generalization, and making logical deductions based on the axioms and previously proven statements.

In the domain of automobile system security, theorem proving can be applied to examine and analyze the security properties pertaining to different components within the system [94–97]. For example, it can aid in validating the accuracy of security methods, such as authentication procedures or encryption algorithms, employed to safeguard critical data within the automotive system. By formalizing the security attributes as propositions and using automated or interactive theorem provers, it becomes feasible to logically demonstrate the absence of certain vulnerabilities or validate the accuracy of security systems.

For instance, consider an automotive system that utilizes an authentication mechanism to grant access to privileged functionalities. The theorem proving approach can be employed to verify the security properties of the authentication protocol. The axioms in this case could include the properties of a secure authentication scheme, such as uniqueness of user credentials and resistance to impersonation attacks. By applying logical rules and making deductions based on the axioms, a formal proof can be constructed to establish the validity of the authentication protocol, ensuring its compliance with the desired security properties.

The utilization of theorem proving offers a methodical and logical technique for guaranteeing the security of automobile systems. It enables the formal verification of security properties and provides a rigorous approach to verify the accuracy and validity of security mechanisms.

3.3. Abstract Interpretation

Abstract interpretation is a rigorous technique that offers a structured approach to estimating the behavior of a system through the abstraction of its concrete components [98–100]. It provides a framework for analyzing programs or systems by approximating their behaviors using abstract representations [101–103]. These abstract representations capture essential properties of the system while disregarding irrelevant details, enabling scalable analysis and aiding in the identification of security vulnerabilities [104–106].

Formally, let C be the concrete domain representing the actual behaviors of a system, and let A be the abstract domain representing an approximation of the concrete behaviors. The abstract interpretation process involves defining a pair of functions: a sound abstraction function $\alpha : C \rightarrow A$, which maps concrete elements to abstract elements, and a concretization function $\gamma : A \rightarrow C$, which maps abstract elements back to concrete elements. These functions establish a formal connection between the concrete and abstract

domains, ensuring that the abstract representations preserve the essential properties of the concrete behaviors.

In the domain of automotive system security, abstract interpretation can be applied to examine and analyze the security attributes of software components [107–109]. For example, abstract interpretation techniques can be used to analyze software modules and identify vulnerabilities by abstracting their behavior and analyzing hypothetical attacker behavior. By creating abstract models that capture the essential security properties and simulating potential attacks, security weaknesses can be identified and appropriate countermeasures can be developed.

Moreover, abstract interpretation can contribute to the evaluation of the effectiveness of security measures and the assessment of the system's ability to withstand attacks. By abstracting the system's behaviors, security properties, such as confidentiality or integrity, can be defined and verified on the abstract representations. This allows for the estimation of the system's security guarantees and provides insights into the potential impact of attacks.

The utilization of abstract interpretation offers a harmonious combination of accuracy and capacity to handle extensive automotive systems, making it a pragmatic option for analysis purposes. It enables scalable analysis by abstracting the system's behaviors and provides a systematic approach to identifying security vulnerabilities and evaluating the effectiveness of security measures.

For instance, consider an automotive system that incorporates a communication protocol between different components. Abstract interpretation can be applied to abstract the behavior of the protocol and analyze its security properties. The concrete behaviors of the protocol, such as message exchanges and encryption operations, can be abstracted to a higher-level representation. Hypothetical attacker behaviors can also be abstracted and analyzed to identify potential vulnerabilities, such as message spoofing or replay attacks. By reasoning about the abstract representations, security weaknesses can be detected and appropriate security measures can be devised.

3.4. Other Relevant Formal Methods

Besides the techniques of model checking, theorem proving, and abstract interpretation, various other formal methods exist that are pertinent for the analysis of security in automotive systems:

- Static Analysis [110–112]: Static analysis approaches involve the scrutiny of program code or system specifications without their execution, with the objective of identifying potential security vulnerabilities. By analyzing the structure of code, the flow of data, and the dependencies within a system, static analysis techniques can effectively detect common security problems, such as buffer overflows or incorrect data handling procedures. Through the examination of the code structure, static analysis possesses the capability to identify prospective coding faults, such as uninitialized variables or unverified input validation, which possess the potential to give rise to security vulnerabilities. Furthermore, static analysis has the capability to detect insecure data handling methods, such as the insecure storage of sensitive information or insufficient safeguards against information leakage. The analysis of data flows and dependencies enables statical analysis to detect potential security vulnerabilities that may arise from the interaction between various system components. These vulnerabilities include inadequate processing of user input and the transmission of unsafe data across different modules. Static analysis tools frequently utilize advanced algorithms to examine code on a broad scale, rendering them well-suited for intricate codebases in the automotive industry.
- Symbolic Execution [113–117]: Symbolic execution is a methodical methodology that entails the deliberate examination of several paths within a program's code, while considering symbolic inputs. The objective of this approach is to identify vulnerabilities and generate test cases. Symbolic execution is a technique that allows for the exploration of various execution paths and the generation of inputs that can test

different program behaviors. This is achieved by executing a program symbolically. This functionality enables the identification of possible attack pathways and can assist in the creation of targeted testing scenarios. Symbolic execution is capable of detecting inputs that can activate security vulnerabilities, such as path circumstances that result in buffer overflows or inputs that circumvent authentication measures. Symbolic execution is a valuable technique that can be employed to generate comprehensive test cases that encompass various program behaviors, encompassing edge cases and extraordinary scenarios. This approach has the potential to unveil concealed security issues. Nevertheless, the utilization of symbolic execution may encounter the issue of path expansion when confronted with intricate programs, hence presenting significant obstacles in terms of scalability. Different methodologies, including constraint solving and path trimming, are utilized to address these difficulties and enhance the feasibility of symbolic execution in the analysis of automotive systems.

- Security Protocol Analysis [118–123]: The main goal of security protocol analysis is to assess the cryptographic protocols utilized in automotive systems, in order to provide secure communication and data transmission. This methodology enables the identification of potential vulnerabilities in protocols, such as replay attacks or deficiencies in key exchange systems, thus helping with the improvement of communication security in the automotive system. The process of security protocol analysis entails the formal modeling of protocols and submitting them to rigorous analysis methodologies, such as formal verification or protocol-specific analysis. Academic verification approaches, such as model checking or theorem proving, can be utilized to ascertain the accuracy of protocol implementations and guarantee compliance with required security features. Protocol-specific analysis techniques primarily concentrate on the identification of vulnerabilities that are specific to cryptographic protocols. These techniques aim to find weaknesses in areas such as key lengths and cipher modes that may pose security risks. Through the process of analysing protocols, security protocol analysis has the capability to detect vulnerabilities that have the potential to result in unauthorized access, violations of data integrity, or breaches of privacy. This analysis has the potential to provide guidance for the development and execution of secure communication protocols that are specifically designed to meet the unique demands of automotive systems.

The aforementioned formal methods, in addition to other undisclosed techniques, constitute a comprehensive array of tools for evaluating the security of automobile systems. The full evaluation of a system's security posture can be achieved by selecting, applying, or combining several formal approaches, depending on the specific requirements and characteristics of the system being analyzed. The utilization of a comprehensive strategy that capitalizes on the advantages of diverse formal approaches has the potential to augment the efficacy of security analysis and facilitate the detection of a broad spectrum of vulnerabilities. Nevertheless, it is imperative to acknowledge that the utilization of formal methods necessitates a certain level of skill and meticulous evaluation of the intricacies inherent in the system. The intricate nature of automotive systems and the imperative to encompass all pertinent security factors present obstacles that necessitate meticulous consideration when utilizing formal approaches for the examination of automotive system security.

4. Validation Techniques for Ensuring Automotive System Security

To ensure the security of automotive systems, various validation techniques are employed. This section discusses some prominent techniques commonly used in the industry, dedicating a subsection to each type.

4.1. Penetration Testing

Penetration testing, alternatively referred to as ethical hacking, encompasses the practice of emulating authentic attacks on a system with the aim of detecting vulnerabilities and evaluating the efficacy of security measures [124–129]. Proficient security practition-

ers endeavor to exploit flaws within the system's defensive measures, such as software susceptibilities or misconfigurations, with the intention of attaining unauthorized entry or executing illegal activities. The evaluation of potential security vulnerabilities in automotive systems can be accomplished by the implementation of penetration tests, which enable the identification of weaknesses. Subsequently, appropriate measures can be undertaken to mitigate these vulnerabilities.

4.1.1. Mathematical Definitions

Penetration testing involves various methodologies and techniques that can be mathematically defined and applied during the testing process. These include:

1. Vulnerability Assessment [130–132]: The process of identifying and quantifying vulnerabilities within a system or network. It involves analyzing the system's configuration, code, and infrastructure to uncover potential weaknesses.
2. Exploitation [133–135]: The act of leveraging a vulnerability or weakness in the system to gain unauthorized access or perform unauthorized actions. This may involve executing malicious code, manipulating data, or bypassing security controls.
3. Privilege Escalation [136]: The process of elevating user privileges within a system or network. It involves exploiting vulnerabilities to gain higher levels of access and control over the target system.

4.1.2. Examples of Penetration Testing Techniques

Penetration testing employs a variety of techniques to identify vulnerabilities and assess the security posture of automotive systems. Some common techniques include:

1. Network Scanning [137,138]: This technique involves scanning the target network to identify active hosts, open ports, and services running on those ports. Network scanning helps in identifying potential entry points and vulnerable systems. An example of network scanning code in Python is shown in Listing 1.

Listing 1. Network scanning code in Python.

```
import nmap

def scan_network(target):
nm = nmap.PortScanner()
nm.scan(target, arguments='-p 1-65535 -sV')

for host in nm.all_hosts():
print(f"Host: {host}")
for port in nm[host]['tcp']:
print(f"Port: {port}, Service: {nm[host]['tcp'][port]['name']}")
```

2. Password Cracking [139,140]: This technique involves attempting to crack passwords to gain unauthorized access to user accounts or systems. It can be performed using various methods such as brute-force attacks, dictionary attacks, or rainbow table attacks. An example of brute-force password cracking script in Python is shown in Listing 2.

Listing 2. Brute-force password cracking script in Python.

```
import itertools
import string
import hashlib

def crack_password(password_hash):
salt = password_hash[:2]
password_hash = password_hash[2:]

for password_length in range(1, 9):
for password in itertools.product(
string.ascii_letters + string.digits, repeat=password_length
):
password = ''.join(password)
hashed_password = hashlib.md5(
(salt + password).encode()
).hexdigest()
```

```
    if hashed_password == password_hash:
    return~password

    return None
```

3. SQL Injection [141,142]: This technique exploits vulnerabilities in web applications that use improper input validation. By injecting malicious SQL queries, an attacker can retrieve sensitive information or manipulate the database. An example of SQL injection attack in a web application is shown in Listing 3.

Listing 3. SQL injection attack in a web application.

```
import~requests

def exploit_sql_injection(url, payload):
payload = f"' OR {payload} --"
response = requests.get(url + "?username=" + payload)

if "Welcome back" in response.text:
return True
else:
return False
```

These are just a few examples of the techniques used in penetration testing. The field of penetration testing is vast and constantly evolving, with new techniques and tools emerging regularly to uncover and address security vulnerabilities in automotive systems.

4.2. Fault Injection

Fault injection is a method employed to assess the robustness of automotive systems by intentionally introducing defects or mistakes [143,144]. By simulating various fault scenarios, such as hardware problems, software errors, or network faults, the system's ability to handle unforeseen or atypical circumstances can be evaluated [145,146]. Fault injection serves as a valuable technique for identifying vulnerabilities, evaluating the impact of faults on system security, and enhancing the system's robustness [147,148].

4.2.1. Types of Fault Injection

There are different types of fault injection techniques that can be utilized in automotive system testing:

1. Hardware Fault Injection [149]: This technique involves introducing faults directly into the hardware components of the system, such as microcontrollers, sensors, or communication interfaces. For example, injecting voltage spikes or electromagnetic interference can simulate faulty hardware conditions and assess the system's resilience.
2. Software Fault Injection [150]: By injecting faults into the software components of the system, such as the operating system, middleware, or application software, the impact of software errors on system behavior can be evaluated. Examples include injecting random errors in data processing or triggering specific software vulnerabilities to test the system's response.
3. Network Fault Injection [151,151]: This technique focuses on injecting faults at the network level to evaluate the system's behavior under various network conditions. For instance, introducing packet loss, latency, or network congestion can assess the system's ability to handle communication failures or adverse network conditions.
4. Timing Fault Injection [152,153]: Timing faults involve injecting errors related to timing and synchronization within the system. This technique aims to evaluate the system's behavior when faced with timing violations or synchronization failures. For example, injecting delays or altering the timing of critical events can assess the system's response and resilience to timing-related faults.

4.2.2. Example of Fault Injection Scenario

Consider the example shown in Listing 4 where a fault injection technique is applied to evaluate the response of an autonomous vehicle's collision avoidance system. The goal is to assess the system's behavior when faced with a sensor fault scenario.

Listing 4. Example fault injection code for sensor fault.

```
import~random

def inject_sensor_fault(sensor_data):
if random.random() < 0.1:  # 10% chance of fault injection
sensor_data *= random.uniform(0.5, 0.9)  # Reduce sensor data by 50-90%

return sensor_data
```

In this example, a Python function 'inject_sensor_fault' is provided to simulate a sensor fault. The function takes the sensor data as input and randomly injects a fault with a 10% chance. If a fault is injected, the sensor data is reduced by a random factor between 0.5 and 0.9, simulating a faulty sensor reading.

By integrating this fault injection code into a simulation environment or a real-world test setup, the collision avoidance system's response to the faulty sensor data can be evaluated. This assessment helps identify potential vulnerabilities or weaknesses in the system's ability to handle sensor faults and improve its robustness.

Fault injection techniques enable automotive system developers and security practitioners to proactively test and enhance the system's resilience against various fault scenarios. By identifying and addressing vulnerabilities early in the development lifecycle, the overall security and reliability of automotive systems can be significantly improved.

4.3. Fuzz Testing

Fuzz testing, also known as fuzzing, is a technique that involves providing a system with a large amount of invalid, unexpected, or random data inputs to uncover vulnerabilities or software bugs [154–157]. By subjecting the system to such inputs, fuzz testing aims to identify potential security weaknesses, such as buffer overflows or input validation errors [158–161]. Fuzz testing can be automated and significantly enhances the security of automotive systems by identifying and fixing software vulnerabilities [162–165].

4.3.1. How Fuzz Testing Works

Fuzz testing follows a simple yet effective process:

1. Input Generation: Fuzz testing involves generating a variety of input data that can potentially trigger unexpected behavior in the system. This can include malformed data, random inputs, or edge cases that are outside the normal range of valid inputs.
2. Input Mutation: The generated inputs are then mutated or modified to create additional variations. This helps explore different paths and uncover vulnerabilities that may be sensitive to specific input patterns.
3. Input Injection: The mutated inputs are injected into the system under test. This can be achieved by feeding the inputs directly to the system's interfaces, such as APIs, command-line interfaces, or file parsers.
4. Monitoring and Analysis: During the execution of the system with the fuzzed inputs, monitoring and analysis tools are employed to detect any anomalies, crashes, or unexpected behavior. This information is then used to identify potential vulnerabilities or bugs.
5. Bug Reporting and Fixing: When a vulnerability or bug is discovered through fuzz testing, it is reported to the developers or security team responsible for the system. They can then investigate the issue, reproduce it, and apply appropriate fixes to enhance the system's security and stability.

4.3.2. Example of Fuzz Testing Scenario

Let us consider an example where fuzz testing is applied to a file parser module in an automotive system. The goal is to identify potential vulnerabilities or crashes when parsing malformed input files. The example is shown in Listing 5.

Listing 5. Example of fuzz testing code for file parser.

```
import~random

def fuzz_test_file_parser(file_parser, num_tests):
for _ in range(num_tests):
fuzzed_input = generate_fuzzed_input()
try:
file_parser.parse(fuzzed_input)
except Exception as e:
print("Crash␣detected:", e)
```

In this example, a Python function 'fuzz_test_file_parser' is provided to perform fuzz testing on a file parser module. The function takes the file parser object and the number of fuzz tests to be performed as the input. Within the function, 'generate_fuzzed_input' is a function that generates a fuzzed input specific to the file parser's expected input format. During the fuzz testing, the function injects the fuzzed inputs into the file parser's 'parse' method. If an exception or crash occurs during the parsing process, it is caught and printed as an indication of a potential vulnerability or issue. By running this fuzz testing function with a significant number of tests, different types of fuzzed inputs can be provided to the file parser, increasing the chances of discovering vulnerabilities or crashes related to input parsing. Fuzz testing allows automotive system developers and security researchers to proactively identify and address potential security weaknesses or bugs in software components. By systematically subjecting the system to unexpected inputs, fuzz testing helps improve the overall security and reliability of automotive systems.

4.4. Security Code Review

Security code review involves the manual or automated inspection of the source code to identify security weaknesses, such as insecure coding practices or vulnerabilities [166,167]. By carefully examining the codebase, security experts can identify potential security flaws, including incorrect use of cryptographic algorithms, lack of input validation, or inadequate protection against common attack vectors [168,169]. Conducting thorough security code reviews is essential for ensuring the robustness of automotive systems [170,171].

4.4.1. Benefits of Security Code Review

Security code review offers several benefits in the development and maintenance of automotive systems [172]:

- Vulnerability Detection: By reviewing the source code, security experts can identify vulnerabilities and weaknesses that may not be easily detectable through other testing techniques. This allows for the early identification and mitigation of security risks before they can be exploited.
- Identification of Security Best Practices: Code reviews provide an opportunity to ensure that the codebase adheres to industry-standard security best practices. This includes verifying the proper use of cryptographic algorithms, secure input validation, and protection against common security vulnerabilities, such as injection attacks or XSS (cross-site scripting) vulnerabilities.
- Compliance and Regulatory Requirements: Automotive systems are often subject to regulatory requirements and industry standards related to security. Security code reviews help ensure compliance with these requirements, reducing the risk of penalties and legal consequences.
- Knowledge Sharing and Team Collaboration: Code reviews promote knowledge sharing and collaboration among development teams. They provide an opportunity

for security experts and developers to exchange insights, address potential security concerns, and enhance their understanding of secure coding practices.

- Continuous Improvement: Conducting security code reviews as part of the development process facilitates a culture of continuous improvement. By actively seeking and addressing security weaknesses, the development team can enhance the security posture of the automotive system over time.

4.4.2. Approaches to Security Code Review

Security code reviews can be conducted using various approaches, including [173,174]:

- Manual Code Review: In a manual code review, security experts carefully examine the source code, line by line, to identify security weaknesses. This approach requires expertise in secure coding practices and an understanding of potential vulnerabilities specific to automotive systems.
- Automated Code Analysis: Automated tools and scanners can be used to perform static code analysis and identify potential security flaws. These tools can quickly scan the codebase, check for common vulnerabilities, and provide a list of potential issues. However, they may also generate false positives or miss certain vulnerabilities that require human judgment.
- Combination of Manual and Automated Approaches: A combination of manual and automated code review approaches is often employed to maximize the effectiveness of the review process. Automated tools can quickly identify common vulnerabilities, while manual review allows for a deeper analysis and identification of complex security issues.

4.4.3. Best Practices for Security Code Review

To ensure effective security code reviews, the following best practices should be considered [175,176]:

- Establish Review Guidelines: Define clear guidelines and criteria for security code reviews to ensure consistency and focus. These guidelines can include secure coding practices, industry standards, and regulatory requirements.
- Involve Security Experts: Engage security experts with expertise in secure coding and automotive systems to perform or guide the code review process. Their knowledge and experience can greatly enhance the effectiveness of the review.
- Encourage Collaboration: Foster collaboration between security experts and developers during the code review process. This promotes knowledge sharing, enables discussions on potential vulnerabilities, and ensures that security concerns are understood and addressed by the development team.
- Prioritize High-Risk Areas: Focus on high-risk areas of the codebase, such as input validation, authentication mechanisms, and cryptographic implementations. These areas are more likely to contain vulnerabilities that can have severe consequences if exploited.
- Document and Track Findings: Document the findings of the code review process, including identified vulnerabilities, recommended fixes, and any discussions or decisions made. This documentation serves as a reference for future development and maintenance activities.
- Follow Up on Findings: Ensure that identified security weaknesses are appropriately addressed and fixed. Regularly follow up on the progress of remediation efforts to mitigate the identified vulnerabilities effectively.

By incorporating security code review practices into the development lifecycle of automotive systems, organizations can proactively identify and address potential security weaknesses. This helps enhance the security posture of the systems and reduces the risk of security incidents or breaches.

4.5. Security Architecture Review

A security architecture review involves a thorough examination of the system's architecture to identify potential security risks and ensure that security measures are properly integrated into the design [177,178]. This review assesses the system's overall security posture, including the implementation of security controls, access management mechanisms, communication protocols, and data protection measures. By conducting security architecture reviews, potential vulnerabilities and design flaws can be addressed early in the development process [179,180].

4.5.1. Importance of Security Architecture Review

Security architecture reviews play a critical role in ensuring the security and resilience of automotive systems. Here are some key reasons security architecture reviews are important [181,182]:

- Identifying Design Flaws: By examining the system's architecture, security experts can identify design flaws or weaknesses that could be exploited by attackers. Detecting and addressing these flaws early in the development process helps prevent potential security breaches and reduces the cost of remediation in later stages.
- Ensuring Proper Integration of Security Controls: A security architecture review ensures that security controls and measures are properly integrated into the system's design. This includes access controls, authentication mechanisms, encryption protocols, and secure communication channels. Verifying the correct implementation of these controls helps protect sensitive data and mitigate security risks.
- Assessing Compliance with Standards and Regulations: Automotive systems are often subject to industry-specific security standards and regulatory requirements. Security architecture reviews help assess the system's compliance with these standards, reducing the risk of non-compliance penalties and ensuring a higher level of security.
- Evaluating Resilience and Threat Mitigation: A comprehensive security architecture review evaluates the system's resilience against various threats and potential attack vectors. It helps identify potential weaknesses in the architecture that could allow unauthorized access, data breaches, or service disruptions. By proactively addressing these vulnerabilities, the system's overall security and availability can be improved.
- Aligning Security with Business Goals: Security architecture reviews ensure that security measures align with the business goals and objectives of the automotive system. By considering the specific requirements and risk appetite of the organization, the review helps strike a balance between security and usability, enabling secure and efficient operations.

4.5.2. Elements of a Security Architecture Review

A thorough security architecture review involves the examination of various elements within the system's architecture. Here are some key aspects that should be considered [183]:

- System Components and Interactions: Analyze the different components of the system and the interactions between them. This includes examining the data flow, communication channels, and interfaces exposed by the system.
- Access Control Mechanisms: Evaluate the design and implementation of access control mechanisms, including authentication, authorization, and user management processes. Verify that appropriate access controls are in place to protect sensitive resources and functionalities.
- Secure Communication Protocols: Review the communication protocols used within the system, including network protocols, API communication, and data exchange mechanisms. Verify that secure protocols and encryption are employed to protect data during transit.

- Threat Modeling and Risk Assessment: Conduct a threat modeling exercise to identify potential threats and attack vectors specific to the automotive system. Perform a risk assessment to prioritize the identified risks and allocate appropriate security measures.
- Resilience and Disaster Recovery: Assess the resilience of the architecture against potential disruptions, including natural disasters, system failures, or cyber attacks. Verify the presence of backup mechanisms, disaster recovery plans, and incident response procedures.

By performing thorough security architecture reviews, automotive system developers can identify and mitigate potential security risks, enhance the overall security posture, and build resilient systems that protect sensitive data and functionalities.

4.6. Threat Modeling

Threat modeling is a systematic approach to identify and prioritize potential threats to the system, assess their impact, and devise appropriate countermeasures [184,185]. It involves analyzing the system's components, interactions, and potential attack vectors to understand the security risks [186,187]. By identifying and prioritizing threats, developers and security professionals can allocate resources effectively to address the most critical vulnerabilities and enhance the overall security of automotive systems [188,189].

In the context of automotive systems, threat modeling plays a crucial role in ensuring the safety and security of vehicles and their associated technologies [190,191]. As technology continues to advance, vehicles are becoming increasingly interconnected and reliant on software, making them more susceptible to cyber threats [192,193]. Threat modeling helps organizations proactively identify and mitigate potential risks, minimizing the likelihood of successful attacks and their potential impact.

When conducting threat modeling for automotive systems, several important considerations come into play. First and foremost, it is essential to understand the system's architecture, including its components, interfaces, and data flows. This understanding enables the identification of potential attack vectors and weak points within the system.

Threat modeling typically involves the following steps:

1. Identifying Assets: Begin by identifying the valuable assets within the automotive system. This includes not only the vehicle itself, but also the data it generates and processes, such as personal information, navigation data, and vehicle telemetry.
2. Creating a System Overview: Develop a comprehensive understanding of the system's architecture, including hardware, software, and network components. This step involves mapping out the system's various elements, their relationships, and the flow of information between them.
3. Identifying Threats: Once the system's architecture is understood, systematically identify potential threats and vulnerabilities. This can be achieved by brainstorming potential attack scenarios, analyzing historical attack patterns, and leveraging industry best practices and security guidelines.
4. Assessing Impact: Evaluate the potential impact of each identified threat on the system and its assets. Consider factors such as the likelihood of the threat being exploited; the potential consequences of a successful attack; and the associated risks to safety, privacy, and financial aspects.
5. Prioritizing Countermeasures: Prioritize the identified threats based on their potential impact and likelihood of occurrence. This step helps allocate resources effectively, ensuring that the most critical vulnerabilities are addressed first. It is important to involve relevant stakeholders, including developers, engineers, and security professionals, in this process to gain diverse perspectives and expertise.
6. Developing Countermeasures: Once threats are prioritized, devise appropriate countermeasures to mitigate the identified risks. This may include implementing security controls, applying secure coding practices, conducting penetration testing, and establishing incident response plans. It is crucial to consider both technical and procedural countermeasures to ensure a holistic approach to security.

7. Continuous Monitoring and Improvement: Threat modeling is not a one-time process but rather an iterative and continuous effort. As new threats emerge and the system evolves, it is important to regularly reassess and update the threat model. This ensures that the system remains resilient against evolving security threats throughout its lifecycle.

By incorporating threat modeling into the development and maintenance of automotive systems, organizations can proactively address security vulnerabilities and enhance the overall resilience of their products. This approach helps minimize the potential for successful attacks, safeguarding the safety and privacy of vehicle occupants and protecting critical data and infrastructure from unauthorized access and misuse.

4.7. Security Testing Frameworks

The utilization of specialized frameworks that provide tools and methodologies for conducting comprehensive security testing is crucial [194–196]. These frameworks encompass a range of techniques, including vulnerability scanning, code analysis, and security assessment [196–198]. They assist in automating security testing processes, identifying security weaknesses, and ensuring adherence to established security standards [199–201]. Security testing frameworks provide a structured and systematic approach to evaluate the security posture of automotive systems [202–205].

4.7.1. Importance of Security Testing Frameworks

Security testing frameworks play a critical role in assessing the security of automotive systems. Here are some key reasons security testing frameworks are important:

- Comprehensive Testing: Security testing frameworks provide a comprehensive set of tools and techniques to assess the security of automotive systems. These frameworks cover a wide range of security aspects, including vulnerability scanning, penetration testing, code analysis, and security assessment. By utilizing these frameworks, organizations can conduct thorough security tests and identify potential vulnerabilities and weaknesses in the system.
- Automation and Efficiency: Security testing frameworks automate various security testing processes, enabling organizations to conduct tests more efficiently and effectively. These frameworks provide automated tools for vulnerability scanning, code analysis, and other security testing activities. Automation helps reduce manual effort, speeds up the testing process, and improves the accuracy of security assessments.
- Standard Compliance: Security testing frameworks often incorporate established security standards and best practices. They provide guidelines and checks to ensure adherence to these standards, such as the ISO 27001, NIST Cybersecurity Framework, or industry-specific security requirements. By using these frameworks, organizations can evaluate their compliance with relevant security standards and demonstrate their commitment to security.
- Risk Mitigation: Security testing frameworks help identify vulnerabilities and weaknesses in automotive systems, allowing organizations to proactively address them. By conducting regular security tests using these frameworks, organizations can identify and mitigate potential risks before they are exploited by attackers. This helps reduce the likelihood and impact of security breaches, protecting sensitive data and maintaining the overall integrity of the system.
- Continuous Improvement: Security testing frameworks facilitate a continuous improvement approach to security. They provide organizations with a structured and systematic way to evaluate the security posture of their automotive systems on an ongoing basis. By regularly utilizing these frameworks, organizations can identify emerging threats, address evolving vulnerabilities, and enhance the overall security of their systems over time.

4.7.2. Common Security Testing Frameworks

There are several widely used security testing frameworks available for evaluating the security of automotive systems. Here are some examples:

- OWASP Testing Guide [206,207]: The Open Web Application Security Project (OWASP) provides a comprehensive testing guide that covers various aspects of application security. It includes methodologies, tools, and techniques for testing web applications, APIs, and other software components.
- NIST SP 800-115 [208]: The National Institute of Standards and Technology (NIST) Special Publication 800-115 provides guidance on information security testing and assessment. It covers topics such as penetration testing, vulnerability scanning, and security assessment methodologies.
- OSSTMM [209]: The Open Source Security Testing Methodology Manual (OSSTMM) is a framework that provides guidelines and methodologies for security testing. It covers areas such as network security, physical security, and operational security.
- PTES [210]: The Penetration Testing Execution Standard (PTES) is a framework that provides a standardized approach to conducting penetration testing. It includes a methodology and guidelines for performing thorough penetration tests on various systems and applications.
- ISSAF [211]: The Information Systems Security Assessment Framework (ISSAF) is a framework that provides guidance on security assessment and testing. It covers areas such as risk assessment, vulnerability assessment, and security auditing.

These frameworks offer a range of tools, methodologies, and guidelines that organizations can leverage to assess the security of their automotive systems effectively.

4.7.3. Integration with Development Lifecycle

To maximize the effectiveness of security testing frameworks, it is essential to integrate security testing throughout the development lifecycle of automotive systems. By incorporating security testing in each phase, from requirements gathering to deployment, organizations can identify and address security issues early on. Integrating security testing frameworks with development methodologies such as DevSecOps helps ensure that security is considered at every stage of the system's lifecycle.

By utilizing security testing frameworks, organizations can enhance the security of their automotive systems, identify vulnerabilities, and establish a robust security posture that protects against potential threats.

4.8. Blockchain Techniques

A blockchain is a growing chain of data blocks linked together, as shown in Figure 3. Spread ledgers are spread over peer-to-peer networks like this network of data blocks. Digital data are synced, copied, distributed, and shared across a peer-to-peer network in a distributed ledger. Each network partner has the same copy of the shared ledger as each device has the newest version. The database can only be expanded by adding blocks to the chain, and the ledger is safe. Changes to chain-registered records are computationally impossible. Decentralization is a major benefit of the distributed ledger. No central authority controls the ledger, but each node updates its ledger when a new block is added to the blockchain via a joint consensus procedure. Blockchain techniques have gained significant attention in recent years due to their potential for enhancing security and trust in various domains [212–217]. In the context of automotive system security, blockchain can be leveraged to provide tamper-proof and transparent transaction records, decentralized identity management, and secure communication channels [218–224].

4.8.1. Integrity and Traceability of Software Updates

One application of blockchain in automotive system security is ensuring the integrity and traceability of software updates. By utilizing a distributed ledger, automotive manu-

facturers can securely record and verify the authenticity of software updates, preventing unauthorized modifications or tampering. This enhances the security of the automotive system by ensuring that only authorized and verified software updates are applied, reducing the risk of malicious code injections or unauthorized changes.

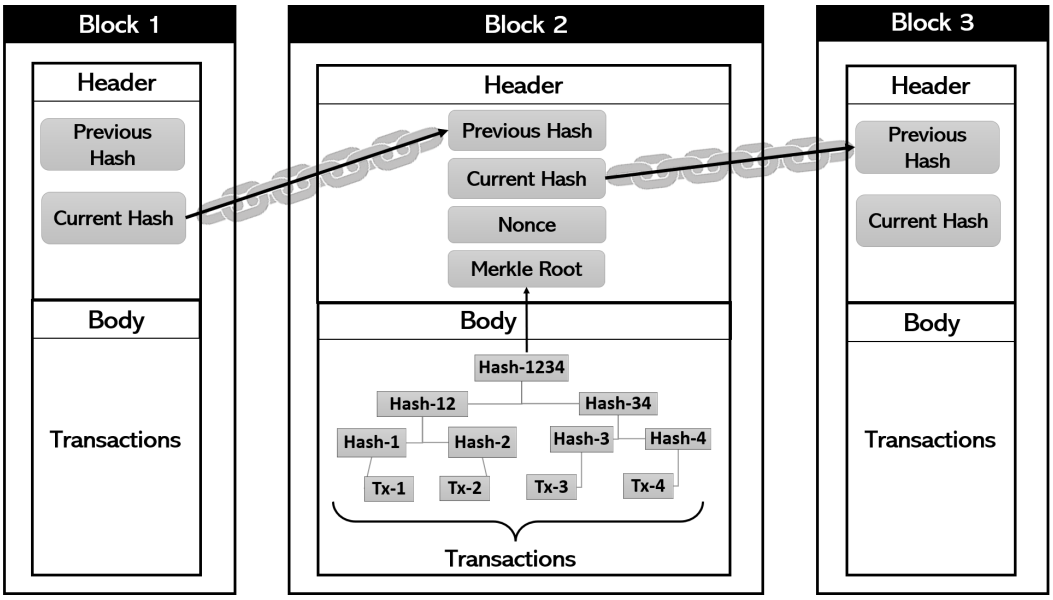

Figure 3. Blockchain general architecture.

4.8.2. Secure Agreements with Smart Contracts

Blockchain-based smart contracts can be employed to establish secure and automated agreements between different entities in the automotive ecosystem, such as suppliers, manufacturers, and service providers [225–227]. Smart contracts are self-executing contracts with the terms of the agreement directly written into code. By leveraging blockchain's decentralized and tamper-proof nature, smart contracts can facilitate secure and transparent transactions, automate payment processes, and enforce compliance with predefined rules and conditions [228–230]. This helps streamline interactions, reduce reliance on intermediaries, and enhance the overall security and efficiency of business operations within the automotive industry [231–233].

4.8.3. Secure and Privacy-Preserving Data Sharing

Blockchain technology can also be utilized for secure and privacy-preserving data sharing among vehicles and infrastructure components. Techniques like zero-knowledge proofs and private transactions can be employed to ensure that sensitive data, such as location information or vehicle diagnostics, can be shared securely without compromising privacy. Zero-knowledge proofs enable the verification of a statement without revealing the underlying data, while private transactions ensure that transaction details are only visible to authorized parties. By leveraging blockchain for secure data sharing, automotive systems can benefit from improved collaboration, enhanced situational awareness, and efficient data-driven services, while maintaining data privacy and security.

4.9. Machine Learning Techniques

Software engineering involves manually writing computer instructions. Automation of rule writing is added by machine learning. In other words, software developers use their brains to solve a problem and create a computer program. The data scientists who install machine learning systems do not write their own programs. They collect input data and goal values. They direct a computer to find software that computes outputs for each input value. In Figure 4, we compare machine learning (ML) with classical programming to illustrate the difference in the approach to software development. The term "program" in the

figure refers to the traditional process of software engineering, where software developers manually write computer instructions to solve a problem. In this approach, developers use their expertise and knowledge to design algorithms and logic that govern the behavior of the program. They carefully craft the code to ensure it performs the desired computations and produces the expected outputs. On the other hand, machine learning introduces a different paradigm. Data scientists who work with machine learning systems do not typically write explicit programs in the traditional sense. Instead, they focus on collecting input data and corresponding goal values. Then, they direct a computer to automatically learn patterns and relationships from the data, enabling it to generate software that can compute outputs for new input values. In this context, the "program" label in the figure represents the automated software generated by the machine learning system based on the collected data and specified goals. By contrasting classical programming with machine learning, Figure 4 aims to highlight the shift from manual rule writing to automated learning and pattern recognition. It illustrates how machine learning leverages data-driven approaches to generate software that can adapt and make predictions or decisions based on new input data. The classical machine learning cycle is shown in Figure 5.

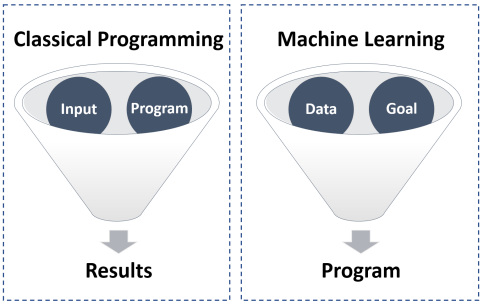

Figure 4. The difference between classical programming and machine learning.

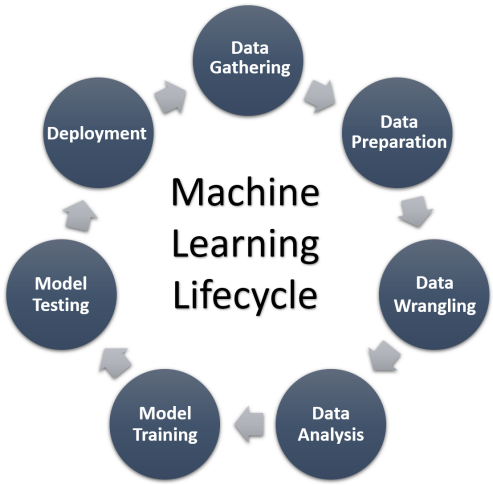

Figure 5. Machine learning lifecycle.

Machine learning techniques have been increasingly applied in automotive system security to detect and mitigate security threats in real-time [234–237]. Machine learning algorithms can analyze vast amounts of data collected from various sensors, network traffic, and system logs to identify anomalous behavior and potential security breaches [238–241].

In the context of automotive cybersecurity, machine learning can be employed for intrusion detection and prevention systems [242–244]. By training models on a combination of normal and malicious activities, machine learning algorithms can learn patterns and identify deviations that may indicate an ongoing attack [245–247]. These algorithms can

continuously monitor the network, detect suspicious activities, and trigger appropriate security measures [248–250].

Additionally, machine learning techniques can be utilized for anomaly detection in vehicle behavior [251,252]. By establishing the baselines of normal vehicle operation, machine learning models can identify deviations that may indicate unauthorized access or malicious control attempts [253,254]. This can help protect vehicles from cyber-physical attacks, such as remote hijacking or manipulation of critical systems [255,256].

Moreover, machine learning techniques can play a vital role in securing connected and autonomous vehicles by enabling predictive maintenance and vulnerability assessment. By analyzing sensor data, machine learning algorithms can identify patterns and correlations that indicate potential vulnerabilities or impending failures in vehicle components. This proactive approach allows manufacturers and service providers to address these issues before they can be exploited by malicious actors [257,258].

Furthermore, machine learning can be leveraged for secure firmware and software updates in vehicles. By employing anomaly detection algorithms, machine learning models can verify the integrity and authenticity of software updates, ensuring that only authorized and tamper-free updates are installed. This eliminates the risk of compromised or malicious software being introduced into the vehicle's systems, thereby maintaining the security and reliability of the vehicle's software stack [259].

Machine learning can also enhance the effectiveness of intrusion response systems in automotive cybersecurity. By continuously analyzing and learning from security incidents, machine learning models can improve their ability to detect and respond to emerging threats. These models can adapt and evolve over time, incorporating new knowledge and techniques to enhance the overall security posture of the vehicle [260].

Overall, machine learning techniques offer significant potential for enhancing automotive system security. They enable real-time threat detection, anomaly detection, predictive maintenance, secure updates, and improved intrusion response. As the automotive industry continues to embrace connectivity and autonomous technologies, the integration of machine learning into cybersecurity practices will become increasingly crucial to ensure the safety and security of vehicles and their occupants.

5. Integration of Formal Methods and Validation Techniques

Ensuring automotive system security requires a comprehensive approach that integrates formal methods and validation techniques throughout the development lifecycle. This section discusses how these methodologies can be effectively integrated into key phases of automotive system development.

5.1. Requirements Engineering Phase

In the requirements engineering phase, the foundation for system security is meticulously established. This phase serves as a critical juncture where the fundamental security objectives and constraints are delineated. Formal methods emerge as indispensable tools in this process, imparting a structured and rigorous approach to specifying security requirements.

Formal specification languages, such as Z notation and Alloy, along with model-based notations like UMLsec, stand at the forefront of this endeavor. These tools provide a systematic framework for expressing security properties, ensuring that they are articulated with precision and clarity. By leveraging formal methods, ambiguities and potential misinterpretations that may arise in natural language specifications are mitigated. This fosters a shared understanding among stakeholders, from developers to security experts, laying a robust foundation for subsequent phases.

Furthermore, formal methods aid in the identification and representation of critical security properties. These may encompass confidentiality, integrity, availability, and other vital attributes. Through the application of formal techniques, the intricacies of security requirements are dissected, allowing for a comprehensive and unambiguous definition of the system's security posture.

Validation techniques complement formal methods during this phase, enriching the process with a practical, risk-oriented perspective. Activities like threat modeling and risk analysis come to the forefront. Threat modeling systematically evaluates potential threats and vulnerabilities that the system may encounter. By examining attack vectors and potential exploit scenarios, the development team gains invaluable insights into the security landscape of the automotive system.

The findings from threat modeling and risk analysis synergize with the formal specification process. They serve as a wellspring of contextual information, enabling a more precise definition of security requirements. This symbiotic relationship between validation techniques and formal methods culminates in a refined and comprehensive set of security specifications.

Ultimately, the requirements engineering phase embodies a harmonious integration of formal methods and validation techniques. Formal methods provide the backbone of structured, unambiguous security specifications, while validation techniques infuse a practical understanding of real-world security risks. Together, they lay the cornerstone for a resilient and well-informed approach to automotive system security.

5.2. Design Phase

The design phase represents a pivotal stage in the development of automotive systems, where the blueprint for the system's architecture is crafted. Formal methods continue to play a central role in this phase, offering a structured approach to refining the security architecture.

Model checking, a cornerstone of formal verification, assumes particular significance during the design phase. This technique allows for an exhaustive exploration of system behavior against specified security properties. By subjecting the design to a battery of meticulously crafted scenarios, model checking unveils potential design flaws or inconsistencies. These may include issues related to access control, data flow, or other critical security aspects. Identifying these vulnerabilities at this early stage is paramount, as it enables preemptive corrective measures to be implemented before the system progresses further.

In tandem with formal methods, validation techniques step forward to provide a practical, hands-on assessment of the design's security robustness. Security architecture review, a meticulous examination of the system's architectural components, serves as a linchpin in this evaluation process. This activity scrutinizes key elements such as network configurations, cryptographic protocols, and access controls. By dissecting the architecture from a security perspective, potential weaknesses or oversights are unearthed, contributing valuable insights for refinement.

Additionally, threat modeling continues to be a vital tool during the design phase. This structured process involves the identification and evaluation of potential threats and vulnerabilities in the system's design. By simulating attack scenarios and considering potential threat agents, developers gain a comprehensive understanding of the security landscape. The insights garnered from threat modeling are instrumental in fine-tuning the security architecture, ensuring that it remains resilient against a spectrum of potential threats.

The interplay between formal methods and validation techniques during the design phase culminates in a robust and meticulously crafted security architecture. Formal methods uncover subtle design intricacies, while validation techniques provide a practical reality check against potential real-world threats. Together, they forge a design that not only meets functional requirements, but also exemplifies a steadfast commitment to security.

5.3. Implementation Phase

The implementation phase marks a pivotal transition in the development process, where the abstract designs and specifications are concretely realized in executable code. Formal methods step into the spotlight during this phase, ensuring that the meticulously defined security requirements are faithfully translated into code that upholds the specified security properties.

Theorem proving stands as a formidable technique in this phase, providing a rigorous mathematical verification process. It subjects the codebase to a battery of formal proofs, meticulously scrutinizing each line for compliance with the specified security properties. This process offers a high degree of confidence that the implemented code aligns with the intended security posture, mitigating the risk of inadvertent vulnerabilities.

Validation techniques, in parallel, constitute a linchpin in the implementation phase. Among these, security code review assumes paramount importance. This systematic review process involves a comprehensive examination of the codebase for potential security vulnerabilities. Developers meticulously inspect the code, identifying potential weak points or oversights that may expose the system to security risks. By rectifying these issues at this stage, developers preemptively fortify the system against potential threats, minimizing the likelihood of vulnerabilities manifesting in the deployed environment.

Furthermore, security testing frameworks, integrated within the implementation phase, play a vital role in validating the security robustness of the implemented code. These frameworks subject the codebase to a battery of simulated attack scenarios, gauging its resilience against potential threats.

The synergy between formal methods and validation techniques during the implementation phase represents a formidable bulwark against security vulnerabilities. Formal methods instill mathematical rigor, ensuring the code aligns with specified security properties. Concurrently, validation techniques, with a focus on security code review, provide a practical safeguard against potential vulnerabilities. Together, they forge an implementation that not only adheres to functional requirements, but also exemplifies a robust commitment to security.

5.4. Testing Phase

The testing phase stands as a critical juncture where the efficacy of security measures is rigorously assessed. This phase represents the culmination of the integration of formal methods and validation techniques, aiming to affirm the robustness of the automotive system's security posture.

Formal methods continue to wield their analytical prowess during this phase. Techniques such as model checking and abstract interpretation are leveraged to systematically scrutinize the system's behavior against specified security properties. Model checking, in particular, allows for an exhaustive exploration of potential states and transitions, providing assurance that critical security properties hold under various conditions. Abstract interpretation complements this by providing a broader analysis of the system's behavior, enabling the identification of potential vulnerabilities that may have evaded formal verification alone.

In parallel, validation techniques emerge as practical enforcers of security resilience. Penetration testing, a cornerstone of security assessment, involves simulated attacks on the system to uncover potential vulnerabilities. By replicating real-world attack scenarios, penetration testing offers a crucial reality check, uncovering weaknesses that may not have been evident through formal verification processes alone. Fault injection, another powerful technique, involves the deliberate introduction of faults or errors into the system to assess its resilience. This method simulates unforeseen circumstances, offering insights into how the system behaves under adverse conditions. Additionally, fuzz testing introduces unexpected inputs to the system, probing for vulnerabilities that may arise from unforeseen data.

The integration of these validation techniques in the testing phase significantly enhances the comprehensiveness of security assessment. They inject a practical dimension, subjecting the system to real-world attack scenarios, thereby uncovering potential vulnerabilities that formal methods alone may not detect.

Through the seamless integration of formal methods and validation techniques, the testing phase serves as the ultimate crucible for automotive system security. Formal methods provide a structured and analytical approach to security verification, while validation techniques offer a practical validation of security measures under real-world

conditions. Together, they ensure that the automotive system emerges from this phase with a robust and validated security posture.

By integrating formal methods and validation techniques across these key phases, automotive systems can achieve a higher level of security robustness. This holistic approach ensures that security considerations are woven into the fabric of the development process, resulting in more resilient systems that are better prepared to withstand evolving cyber threats.

Table 2 provides a concise summary of the integration of formal methods and validation techniques in each phase of automotive system development, while maintaining the requested line separation.

Table 2. Integration of formal methods and validation techniques in automotive system development.

Phase	Summary
Requirements Engineering	Integration of formal methods (such as Z notation, Alloy, and UMLsec) for specifying security requirements. Validation techniques like threat modeling and risk analysis complement formal methods to refine security specifications.
Design	Formal methods (e.g., model checking) used for exhaustive exploration of system behavior against security properties. Validation techniques (e.g., security architecture review, threat modeling) identify weaknesses and refine the security architecture.
Implementation	Formal methods (e.g., theorem proving) ensure code compliance with specified security properties. Validation techniques (e.g., security code review and security testing frameworks) fortify the code against vulnerabilities.
Testing	Formal methods (e.g., model checking, abstract interpretation) analyze system behavior and verify security properties. Validation techniques (e.g., penetration testing, fault injection, and fuzz testing) simulate real-world attacks and assess system resilience.

5.5. Distinctive Aspects of Formal Methods and Validation Techniques in Enhancing Automotive System Security

Formal methods and validation techniques play a crucial role in enhancing the security of various systems, including automotive systems, IoT devices, software applications, and embedded systems. While there are commonalities in the application of these methods across different domains, there are also distinctive aspects specific to the use of formal methods and validation techniques for automotive system security:

1. Complex and Safety-Critical Nature of Automotive Systems: Automotive systems are characterized by their complexity and safety-critical nature. They involve intricate interactions between various components, including sensors, actuators, control units, and communication networks. Formal methods and validation techniques need to address the unique challenges posed by the complexity of automotive systems, such as modeling the behavior of interconnected components, verifying safety properties, and ensuring the reliability and robustness of the system under different operating conditions.
2. Real-Time Constraints and Performance Requirements: Automotive systems operate in real-time environments, where timely and accurate responses are essential for ensuring safety and security. Formal methods and validation techniques for automotive systems must consider real-time constraints, including response times, latency, and timing requirements. Analyzing and verifying the timing behavior of automotive systems is crucial to prevent potential security vulnerabilities and ensure the system's reliable operation.

3. Integration of Safety and Security Considerations: Unlike other systems, automotive systems require the integration of both safety and security considerations. While safety focuses on preventing accidents and minimizing harm to occupants and pedestrians, security addresses the protection of the system against malicious attacks and unauthorized access. Formal methods and validation techniques in the automotive domain need to encompass both safety and security aspects, ensuring that the system is resilient to both accidental failures and intentional attacks.
4. Automotive-Specific Threat Landscape: The automotive domain presents a unique threat landscape compared with other systems. Automotive systems are susceptible to a wide range of security threats, including remote exploits, unauthorized access to the vehicle's network, tampering with electronic control units (ECUs), and compromising the integrity of sensor data. Formal methods and validation techniques for automotive system security must address these specific threats and vulnerabilities, considering the potential impact on safety, privacy, and the overall functionality of the vehicle.
5. Compliance with Industry Standards and Regulations: The automotive industry is subject to stringent safety and security standards and regulations. Formal methods and validation techniques must align with these industry-specific standards, such as ISO 26262 for functional safety and ISO/SAE 21434 for automotive cybersecurity. Adhering to these standards ensures that the application of formal methods and validation techniques in automotive systems meets the necessary requirements and guidelines for safety and security.

In summary, while there are commonalities in the application of formal methods and validation techniques across various domains, the use of these methods in enhancing automotive system security involves distinctive aspects. The complex and safety-critical nature of automotive systems, real-time constraints, integration of safety and security considerations, automotive-specific threat landscape, and compliance with industry standards all contribute to the unique challenges and considerations in securing automotive systems. Addressing these distinctive aspects is crucial for effectively applying formal methods and validation techniques to enhance the security of automotive systems.

6. Benefits and Limitations of the Approaches

The integration of formal methods and validation techniques brings a wealth of advantages to the domain of automotive system security. However, it is important to acknowledge that these approaches are not without their own set of considerations and constraints.

6.1. Scalability Considerations

Benefits: Scalability is a crucial factor in the assessment of security methodologies, especially in the context of complex automotive systems. Formal methods, with their mathematical foundation, excel in handling complexity. They offer the ability to analyze intricate systems with precision, ensuring that security properties are verified comprehensively. This proves invaluable for identifying potential vulnerabilities in large-scale automotive systems.

Limitations: However, as systems grow in complexity, the computational resources required for formal verification can escalate significantly. Model checking, for instance, may face scalability challenges when dealing with exceptionally large state spaces. This necessitates careful consideration of resource allocation and the exploration of specialized techniques to address scalability concerns.

6.2. Efficiency Considerations

Benefits: Formal methods provide a level of assurance that is hard to match with purely empirical testing. They offer a systematic and exhaustive approach to security verification. By leveraging mathematical rigor, formal methods can identify vulnerabilities with a high degree of confidence, reducing the likelihood of false negatives.

Limitations: Yet, formal methods can be computationally intensive, potentially leading to prolonged verification times. Theorem proving, for instance, may require significant computational resources to establish the correctness of complex code segments. Striking a balance between precision and efficiency is paramount to ensure that the verification process remains practical within the constraints of real-world development cycles.

6.3. Applicability to Real-World Automotive Systems

Benefits: The effectiveness of formal methods and validation techniques in the context of real-world automotive systems is evident in their ability to uncover subtle security vulnerabilities. By subjecting systems to rigorous analysis and real-world attack scenarios, these approaches provide a robust defense against potential threats. This proactive stance towards security aligns with the evolving landscape of automotive technology, where cyber threats continue to evolve.

Limitations: However, the applicability of formal methods and validation techniques may vary depending on the specific characteristics of the automotive system. Highly specialized or proprietary systems may present unique challenges in terms of integration and suitability for certain formal verification techniques. Additionally, the availability of skilled practitioners proficient in formal methods may influence the feasibility of their widespread adoption within the automotive industry.

In summary (See Table 3), the integration of formal methods and validation techniques offers a powerful arsenal in the pursuit of automotive system security. While they provide unparalleled precision in identifying vulnerabilities, considerations of scalability, efficiency, and applicability to real-world systems must be taken into account. Striking a balance between these factors is crucial for harnessing the full potential of these methodologies.

Table 3. Benefits and Limitations.

Consideration	Benefits	Limitations
Scalability	Formal methods excel in handling complexity, allowing for comprehensive verification of security properties in large-scale automotive systems.	They may face scalability challenges with exceptionally large state spaces.
Efficiency	Formal methods offer a systematic and exhaustive approach to security verification, reducing the likelihood of false negatives.	They can be computationally intensive, requiring careful balancing of precision and efficiency.
Applicability to Real-World Automotive Systems	Formal methods and validation techniques effectively uncover security vulnerabilities in real-world automotive systems.	Their applicability may vary depending on system characteristics and the availability of skilled practitioners.

7. Current Research Trends and Open Research Questions

The field of formal methods and validation techniques for automotive system security is in a state of constant evolution, driven by ongoing research and technological advancements. This section provides an in-depth exploration of the current research trends and outlines crucial open questions that warrant further investigation.

7.1. Literature Review

A comprehensive literature review reveals a dynamic landscape marked by a surge of interest in the integration of formal methods and validation techniques for automotive system security. Researchers have undertaken a diverse range of studies, spanning from the development of novel formal models to the exploration of innovative validation methodologies. Noteworthy contributions have addressed critical challenges, including

the scalability of verification processes, the application of advanced mathematical techniques, and the proposal of frameworks for seamless integration within the automotive development lifecycle.

Moreover, researchers have made significant strides in bridging the gap between theoretical formal methods and practical implementation, exemplified by case studies showcasing successful applications in real-world automotive systems. These studies serve as compelling testaments to the efficacy of these methodologies, offering valuable insights into the practical challenges and solutions in ensuring automotive system security.

7.2. Emerging Trends

The ever-evolving landscape of formal methods and validation techniques for automotive system security is marked by the emergence of innovative trends poised to shape the future of the field. Notably, researchers are exploring the integration of machine learning algorithms as a means to augment security assessments. Machine learning techniques hold the potential to enhance automation and precision in identifying security vulnerabilities, representing a paradigm shift in how automotive systems are evaluated for security.

Additionally, the incorporation of blockchain technology has garnered substantial attention. Blockchain's inherent characteristics, such as tamper-resistance and decentralized consensus mechanisms, present opportunities to ensure the integrity and traceability of software updates. Furthermore, the use of smart contracts within blockchain ecosystems holds promise for establishing secure agreements in automotive system interactions.

7.3. Open Research Questions

While significant strides have been taken, critical open research questions persist within the domain of formal methods and validation techniques for automotive system security. One pressing concern revolves around the development of techniques capable of effectively handling the escalating scalability demands posed by increasingly intricate automotive systems. As vehicles become more connected and autonomous, the complexity of verifying their security properties becomes paramount.

Furthermore, a pressing need exists for methodologies that can seamlessly adapt to the evolving threat landscape. Ensuring resilience against sophisticated cyber attacks remains a formidable challenge, necessitating innovative approaches that go beyond conventional security paradigms.

A promising area for future exploration lies at the intersection of formal methods, machine learning, and blockchain technology. Investigating how these synergistic technologies can be harnessed to bolster security measures presents an exciting frontier with the potential to revolutionize automotive system security.

In summary, the current research landscape in formal methods and validation techniques for automotive system security is marked by a dynamic interplay of literature reviews, insightful case studies, the exploration of emerging trends, and the pursuit of answers to open research questions. This vibrant field holds the promise of significantly shaping the future of secure automotive systems, fortifying them against an ever-evolving array of cyber threats.

8. Conclusions

In the pursuit of automotive system security, the integration of formal methods and validation techniques emerges as a formidable approach. This survey has explored the landscape of these methodologies, shedding light on their application, benefits, and limitations. In this concluding section, we summarize the key findings and discuss their contributions and implications.

The survey has elucidated the pivotal role played by formal methods and validation techniques in ensuring the security of automotive systems. From the early stages of requirements engineering to the testing phase, these methodologies provide a structured and systematic means of identifying and mitigating security vulnerabilities. Formal methods,

including model checking and theorem proving, offer mathematical rigor in verifying security properties, while validation techniques such as penetration testing and fault injection provide a practical validation of security measures.

The integration of these methodologies within the automotive development lifecycle fosters a holistic approach to security, embedding it in the very fabric of system design and implementation. Case studies and emerging trends further underscore the practical applicability and evolving nature of these techniques, highlighting their instrumental role in safeguarding automotive systems.

The contributions of this survey lie in providing a comprehensive overview of the current state-of-the-art formal methods and validation techniques for automotive system security. By synthesizing research findings, case studies, and emerging trends, this survey serves as a valuable resource for researchers, practitioners, and policymakers involved in the design, development, and evaluation of secure automotive systems.

Furthermore, the implications of this survey extend to the broader landscape of cybersecurity. The methodologies discussed here offer valuable insights and principles that are transferable to other domains with similar security concerns. The emphasis on proactive security measures, rigorous verification, and real-world validation scenarios provides a blueprint for fortifying systems against an ever-evolving threat landscape.

In conclusion, the integration of formal methods and validation techniques represents a crucial paradigm in automotive system security. This survey underscores their significance, offering a roadmap for fortifying automotive systems against a spectrum of potential threats. As the automotive industry continues to evolve, these methodologies will remain pivotal in ensuring the security and integrity of the vehicles of tomorrow.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Rahim, M.A.; Rahman, M.A.; Rahman, M.M.; Asyhari, A.T.; Bhuiyan, M.Z.A.; Ramasamy, D. Evolution of IoT-enabled connectivity and applications in automotive industry: A review. *Veh. Commun.* **2021**, *27*, 100285. [CrossRef]
2. Sadaf, M.; Iqbal, Z.; Javed, A.R.; Saba, I.; Krichen, M.; Majeed, S.; Raza, A. Connected and Automated Vehicles: Infrastructure, Applications, Security, Critical Challenges, and Future Aspects. *Technologies* **2023**, *11*, 117. [CrossRef]
3. Gohoungodji, P.; N'Dri, A.B.; Latulippe, J.M.; Matos, A.L.B. What is stopping the automotive industry from going green? A systematic review of barriers to green innovation in the automotive industry. *J. Clean. Prod.* **2020**, *277*, 123524. [CrossRef]
4. Sarfraz, M.S.; Hong, H.; Kim, S.S. Recent developments in the manufacturing technologies of composite components and their cost-effectiveness in the automotive industry: A review study. *Compos. Struct.* **2021**, *266*, 113864. [CrossRef]
5. Almeaibed, S.; Al-Rubaye, S.; Tsourdos, A.; Avdelidis, N.P. Digital twin analysis to promote safety and security in autonomous vehicles. *IEEE Commun. Stand. Mag.* **2021**, *5*, 40–46. [CrossRef]
6. Schmittner, C.; Macher, G. Automotive cybersecurity standards-relation and overview. In Proceedings of the Computer Safety, Reliability, and Security: SAFECOMP 2019 Workshops, ASSURE, DECSoS, SASSUR, STRIVE, and WAISE, Turku, Finland, 10 September 2019; Proceedings 38. pp. 153–165.
7. Sommer, F.; Dürrwang, J.; Kriesten, R. Survey and classification of automotive security attacks. *Information* **2019**, *10*, 148. [CrossRef]
8. Breuing, H.; Heil, L.; Vierling, B. It security for the entire automotive ecosystem. *ATZelectronics Worldw.* **2019**, *14*, 60–63. [CrossRef]
9. Young, C.; Zambreno, J.; Olufowobi, H.; Bloom, G. Survey of automotive controller area network intrusion detection systems. *IEEE Des. Test* **2019**, *36*, 48–55. [CrossRef]
10. Dobaj, J.; Macher, G.; Ekert, D.; Riel, A.; Messnarz, R. Towards a security-driven automotive development lifecycle. *J. Softw. Evol. Process* **2023**, *35*, e2407. [CrossRef]
11. Huang, J.; Zhao, M.; Zhou, Y.; Xing, C.C. In-vehicle networking: Protocols, challenges, and solutions. *IEEE Netw.* **2018**, *33*, 92–98. [CrossRef]
12. Yu, Z.; Khan, S.A.R.; Umar, M. Circular economy practices and industry 4.0 technologies: A strategic move of automobile industry. *Bus. Strategy Environ.* **2022**, *31*, 796–809. [CrossRef]
13. Krichen, M. A Survey on Formal Verification and Validation Techniques for Internet of Things. *Appl. Sci.* **2023**, *13*, 8122. [CrossRef]

14. Moghadasi, N.; Kulkarni, A.; Crayton, D.; Grissom, R.; Lambert, J.H.; Feng, L. Formal Methods in Unmanned Aerial Vehicle Swarm Control for Wildfire Detection and Monitoring. In Proceedings of the 2023 IEEE International Systems Conference (SysCon), Vancouver, BC, Canada, 17–20 April 2023; pp. 1–8.
15. Krichen, M. Contributions to Model-Based Testing of Dynamic and Distributed Real-Time Systems. Ph.D. Thesis, École Nationale d'Ingénieurs de Sfax (Tunisie), Sfax, Tunisia, 2018.
16. Zita, A.; Mohajerani, S.; Fabian, M. Application of formal verification to the lane change module of an autonomous vehicle. In Proceedings of the 2017 13th IEEE Conference on Automation Science and Engineering (CASE), Xi'an, China, 20–23 August 2017; pp. 932–937.
17. Krichen, M.; Alroobaea, R. A New Model-based Framework for Testing Security of IoT Systems in Smart Cities using Attack Trees and Price Timed Automata. In Proceedings of the 14th International Conference on Evaluation of Novel Approaches to Software Engineering—ENASE 2019, Setubal, Portugal, 4–5 May 2019.
18. Krichen, M.; Mechti, S.; Alroobaea, R.; Said, E.; Singh, P.; Khalaf, O.I.; Masud, M. A formal testing model for operating room control system using internet of things. *Comput. Mater. Contin.* **2021**, *66*, 2997–3011. [CrossRef]
19. Maâlej, A.J.; Krichen, M. A Model Based Approach to Combine Load and Functional Tests for Service Oriented Architectures. In Proceedings of the VECoS, Tunis, Tunisia, 6–7 October 2016; pp. 123–140.
20. Krichen, M. A formal framework for black-box conformance testing of distributed real-time systems. *Int. J. Crit. Comput.-Based Syst.* **2012**, *3*, 26–43. [CrossRef]
21. Lamssaggad, A.; Benamar, N.; Hafid, A.S.; Msahli, M. A survey on the current security landscape of intelligent transportation systems. *IEEE Access* **2021**, *9*, 9180–9208. [CrossRef]
22. Rumez, M.; Grimm, D.; Kriesten, R.; Sax, E. An overview of automotive service-oriented architectures and implications for security countermeasures. *IEEE Access* **2020**, *8*, 221852–221870. [CrossRef]
23. Luo, F.; Jiang, Y.; Zhang, Z.; Ren, Y.; Hou, S. Threat analysis and risk assessment for connected vehicles: A survey. *Secur. Commun. Netw.* **2021**, *2021*, 1263820. [CrossRef]
24. Halder, S.; Ghosal, A.; Conti, M. Secure over-the-air software updates in connected vehicles: A survey. *Comput. Netw.* **2020**, *178*, 107343. [CrossRef]
25. Sun, X.; Yu, F.R.; Zhang, P. A survey on cyber-security of connected and autonomous vehicles (CAVs). *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 6240–6259. [CrossRef]
26. Pekaric, I.; Sauerwein, C.; Haselwanter, S.; Felderer, M. A taxonomy of attack mechanisms in the automotive domain. *Comput. Stand. Interfaces* **2021**, *78*, 103539. [CrossRef]
27. Hbaieb, A.; Ayed, S.; Chaari, L. A survey of trust management in the Internet of Vehicles. *Comput. Netw.* **2022**, *203*, 108558. [CrossRef]
28. Kim, K.; Kim, J.S.; Jeong, S.; Park, J.H.; Kim, H.K. Cybersecurity for autonomous vehicles: Review of attacks and defense. *Comput. Secur.* **2021**, *103*, 102150. [CrossRef]
29. Pham, M.; Xiong, K. A survey on security attacks and defense techniques for connected and autonomous vehicles. *Comput. Secur.* **2021**, *109*, 102269. [CrossRef]
30. Masood, A.; Lakew, D.S.; Cho, S. Security and privacy challenges in connected vehicular cloud computing. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 2725–2764. [CrossRef]
31. Bera, S.; Misra, S.; Vasilakos, A.V. Software-defined networking for internet of things: A survey. *IEEE Internet Things J.* **2017**, *4*, 1994–2008. [CrossRef]
32. Said, S.B.H.; Cousin, B.; Lahoud, S. Software Defined Networking (SDN) for reliable user connectivity in 5G Networks. In Proceedings of the 2017 IEEE Conference on Network Softwarization (NetSoft), Bologna, Italy, 3–7 July 2017; pp. 1–5.
33. Studnia, I.; Nicomette, V.; Alata, E.; Deswarte, Y.; Kaâniche, M.; Laarouchi, Y. Survey on security threats and protection mechanisms in embedded automotive networks. In Proceedings of the 2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W), Budapest, Hungary, 24–27 June 2013; pp. 1–12.
34. Miller, C.; Valasek, C. A survey of remote automotive attack surfaces. *Black Hat USA* **2014**, *2014*, 94.
35. Khuwaja, A.A.; Chen, Y.; Zhao, N.; Alouini, M.S.; Dobbins, P. A survey of channel modeling for UAV communications. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 2804–2821. [CrossRef]
36. Zeng, Y.; Zhang, R.; Lim, T.J. Wireless communications with unmanned aerial vehicles: Opportunities and challenges. *IEEE Commun. Mag.* **2016**, *54*, 36–42. [CrossRef]
37. Khan, A.; Ahmad, A.; Ahmed, M.; Sessa, J.; Anisetti, M. Authorization schemes for internet of things: Requirements, weaknesses, future challenges and trends. *Complex Intell. Syst.* **2022**, *8*, 3919–3941. [CrossRef]
38. Trnka, M.; Cerny, T.; Stickney, N. Survey of Authentication and Authorization for the Internet of Things. *Secur. Commun. Netw.* **2018**, *2018*, 4351603. [CrossRef]
39. Hanif, H.; Nasir, M.H.N.M.; Ab Razak, M.F.; Firdaus, A.; Anuar, N.B. The rise of software vulnerability: Taxonomy of software vulnerabilities detection and machine learning approaches. *J. Netw. Comput. Appl.* **2021**, *179*, 103009. [CrossRef]
40. Moiz, A.; Alalfi, M.H. A survey of security vulnerabilities in android automotive apps. In Proceedings of the 3rd International Workshop on Engineering and Cybersecurity of Critical Systems, Pittsburgh, PA, USA, 16 May 2022; pp. 17–24.

41. Lopez, T.; Sharp, H.; Tun, T.; Bandara, A.; Levine, M.; Nuseibeh, B. "Hopefully We Are Mostly Secure": Views on Secure Code in Professional Practice. In Proceedings of the 2019 IEEE/ACM 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), Montreal, QC, Canada, 27 May 2019; pp. 61–68.
42. Gasiba, T.E.; Lechner, U.; Pinto-Albuquerque, M.; Fernandez, D.M. Awareness of Secure Coding Guidelines in the Industry-A first data analysis. In Proceedings of the 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Guangzhou, China, 29 December–1 January 2020; pp. 345–352.
43. Meng, N.; Nagy, S.; Yao, D.; Zhuang, W.; Argoty, G.A. Secure coding practices in java: Challenges and vulnerabilities. In Proceedings of the 40th International Conference on Software Engineering, Gothenburg, Sweden, 27 May–3 June 2018; pp. 372–383.
44. Barrère, M.; Hankin, C.; Nicolaou, N.; Eliades, D.G.; Parisini, T. Measuring cyber-physical security in industrial control systems via minimum-effort attack strategies. *J. Inf. Secur. Appl.* **2020**, *52*, 102471. [CrossRef]
45. Dorbala, S.Y.; Bhadoria, R.S. Analysis for security attacks in cyber-physical systems. In *Cyber-Physical Systems: A Computational Perspective*; Chapman and Hall/CRC: Boca Raton, FL, USA, 2015; pp. 395–414.
46. Subke, P.; Moshref, M.; Vach, A.; Steffelbauer, M. Measures to prevent unauthorized access to the in-vehicle e/e system, due to the security vulnerability of a remote diagnostic tester. *SAE Int. J. Passeng. Cars-Electron. Electr. Syst.* **2017**, *10*, 422–429. [CrossRef]
47. Guerar, M.; Verderame, L.; Merlo, A.; Palmieri, F.; Migliardi, M.; Vallerini, L. CirclePIN: A novel authentication mechanism for smartwatches to prevent unauthorized access to IoT devices. *ACM Trans. Cyber-Phys. Syst.* **2020**, *4*, 1–19. [CrossRef]
48. Onik, M.M.H.; Chul-Soo, K.; Jinhong, Y. Personal data privacy challenges of the fourth industrial revolution. In Proceedings of the 2019 21st International Conference on Advanced Communication Technology (ICACT), PyeongChang, Republic of Korea, 17–20 February 2019; pp. 635–638.
49. Rustad, M.L.; Koenig, T.H. Towards a global data privacy standard. *Fla. L. Rev.* **2019**, *71*, 365.
50. Sun, Y.; Jee, K.; Sivakorn, S.; Li, Z.; Lumezanu, C.; Korts-Parn, L.; Wu, Z.; Rhee, J.; Kim, C.H.; Chiang, M.; et al. Detecting malware injection with program-dns behavior. In Proceedings of the 2020 IEEE European Symposium on Security and Privacy (EuroS&P), Genoa, Italy, 7–11 September 2020; pp. 552–568.
51. Ranjan, I.; Agnihotri, R.B. Ambiguity in cloud security with malware-injection attack. In Proceedings of the 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 12–14 June 2019; pp. 306–310.
52. Carreras Guzman, N.H.; Wied, M.; Kozine, I.; Lundteigen, M.A. Conceptualizing the key features of cyber-physical systems in a multi-layered representation for safety and security analysis. *Syst. Eng.* **2020**, *23*, 189–210. [CrossRef]
53. Zhou, C.; Luo, H.; Fang, W.; Wei, R.; Ding, L. Cyber-physical-system-based safety monitoring for blind hoisting with the internet of things: A case study. *Autom. Constr.* **2019**, *97*, 138–150. [CrossRef]
54. Ganesh, V.; Sharma, M. Intrusion Detection and Prevention Systems: A Review. In *Inventive Communication and Computational Technologies: Proceedings of ICICCT 2020*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 835–844.
55. Lenard, T.; Bolboaca, R. A statefull firewall and intrusion detection system enforced with secure logging for controller area network. In Proceedings of the European Interdisciplinary Cybersecurity Conference, Targu Mures, Romania, 10 November 2021; pp. 39–45.
56. Sayeed, M.A.; Sayeed, M.A.; Saxena, S. Intrusion detection system based on Software Defined Network firewall. In Proceedings of the 2015 1st International Conference on Next Generation Computing Technologies (NGCT), Dehradun, India, 4–5 September 2015; pp. 379–382.
57. Potluri, S.; Diedrich, C. High performance intrusion detection and prevention systems: A survey. In Proceedings of the ECCWS2016—The 15th European Conference on Cyber Warfare and Security, Munich, Germany, 7–8 July 2016; p. 260.
58. Ferretti, L.; Marchetti, M.; Colajanni, M. Fog-based secure communications for low-power IoT devices. *ACM Trans. Internet Technol. (TOIT)* **2019**, *19*, 1–21. [CrossRef]
59. Khan, N.A.; Jhanjhi, N.Z.; Brohi, S.N.; Nayyar, A. Emerging use of UAV's: Secure communication protocol issues and challenges. In *Drones in Smart-Cities*; Elsevier: Amsterdam, The Netherlands, 2020; pp. 37–55.
60. Nguyen, K.T.; Laurent, M.; Oualha, N. Survey on secure communication protocols for the Internet of Things. *Ad Hoc Netw.* **2015**, *32*, 17–31. [CrossRef]
61. Dragomir, D.; Gheorghe, L.; Costea, S.; Radovici, A. A survey on secure communication protocols for IoT systems. In Proceedings of the 2016 international workshop on Secure Internet of Things (SIoT), Heraklion, Greece, 26–30 September 2016; pp. 47–62.
62. Krichen, M.; Adoni, W.Y.H.; Mihoub, A.; Alzahrani, M.Y.; Nahhal, T. Security challenges for drone communications: Possible threats, attacks and countermeasures. In Proceedings of the 2022 2nd International Conference of Smart Systems and Emerging Technologies (SMARTTECH), Riyadh, Saudi Arabia, 9–11 May 2022; pp. 184–189.
63. Alshahrani, M.; Traore, I. Secure mutual authentication and automated access control for IoT smart home using cumulative keyed-hash chain. *J. Inf. Secur. Appl.* **2019**, *45*, 156–175. [CrossRef]
64. Nandy, T.; Idris, M.Y.I.B.; Noor, R.M.; Kiah, L.M.; Lun, L.S.; Juma'at, N.B.A.; Ahmedy, I.; Ghani, N.A.; Bhattacharyya, S. Review on security of internet of things authentication mechanism. *IEEE Access* **2019**, *7*, 151054–151089. [CrossRef]
65. El Sibai, R.; Gemayel, N.; Bou Abdo, J.; Demerjian, J. A survey on access control mechanisms for cloud computing. *Trans. Emerg. Telecommun. Technol.* **2020**, *31*, e3720. [CrossRef]
66. Behrad, S.; Bertin, E.; Tuffin, S.; Crespi, N. A new scalable authentication and access control mechanism for 5G-based IoT. *Future Gener. Comput. Syst.* **2020**, *108*, 46–61. [CrossRef]

67. Moyón, F.; Almeida, P.; Riofrío, D.; Mendez, D.; Kalinowski, M. Security compliance in agile software development: A systematic mapping study. In Proceedings of the 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Portoroz, Slovenia, 26– 28 August 2020; pp. 413–420.
68. Tahaei, M.; Vaniea, K. A survey on developer-centred security. In Proceedings of the 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Stockholm, Sweden, 17–19 June 2019; pp. 129–138.
69. Ansari, M.T.J.; Pandey, D.; Alenezi, M. STORE: Security threat oriented requirements engineering methodology. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 191–203. [CrossRef]
70. Singleton, L.; Zhao, R.; Song, M.; Siy, H. Cryptotutor: Teaching secure coding practices through misuse pattern detection. In Proceedings of the 21st Annual Conference on Information Technology Education, Omaha, NE, USA, 7–9 October 2020; pp. 403–408.
71. Shrivastava, R.K.; Singh, S.P.; Hasan, M.K.; Islam, S.; Abdullah, S.; Aman, A.H.M. Securing Internet of Things devices against code tampering attacks using Return Oriented Programming. *Comput. Commun.* **2022**, *193*, 38–46. [CrossRef]
72. Shrivastava, R.; Singh, S.P.; Hasan, M.K. Code Tamper-Proofing Using Return Oriented Programming in IoT Devices. In *Rising Threats in Expert Applications and Solutions: Proceedings of FICR-TEAS 2022*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 167–174.
73. Sey, C.; Lei, H.; Qian, W.; Li, X.; Fiasam, L.D.; Kodjiku, S.L.; Adjei-Mensah, I.; Agyemang, I.O. VBlock: A Blockchain-Based Tamper-Proofing Data Protection Model for Internet of Vehicle Networks. *Sensors* **2022**, *22*, 8083. [CrossRef] [PubMed]
74. Xu, Y.; Li, X.; Jin, M.; Lu, Y. A Trusted Distribution Mechanism of Tasks for the Internet of Vehicles Based on Blockchain. In Proceedings of the 2021 13th International Conference on Wireless Communications and Signal Processing (WCSP), Changsha, China, 20–22 October 2021; pp. 1–5.
75. Rajabli, N.; Flammini, F.; Nardone, R.; Vittorini, V. Software verification and validation of safe autonomous cars: A systematic literature review. *IEEE Access* **2020**, *9*, 4797–4819. [CrossRef]
76. Fremont, D.J.; Kim, E.; Pant, Y.V.; Seshia, S.A.; Acharya, A.; Bruso, X.; Wells, P.; Lemke, S.; Lu, Q.; Mehta, S. Formal scenario-based testing of autonomous vehicles: From simulation to the real world. In Proceedings of the 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), Rhodes, Greece, 20–23 September 2020; pp. 1–8.
77. Domenici, A.; Fagiolini, A.; Palmieri, M. Integrated simulation and formal verification of a simple autonomous vehicle. In Proceedings of the Software Engineering and Formal Methods: SEFM 2017 Collocated Workshops: DataMod, FAACS, MSE, CoSim-CPS, and FOCLASA, Trento, Italy, 4–5 September 2017; Revised Selected Papers 15; pp. 300–314.
78. Bérard, B.; Bidoit, M.; Finkel, A.; Laroussinie, F.; Petit, A.; Petrucci, L.; Schnoebelen, P. *Systems and Software Verification: Model-Checking Techniques and Tools*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2013.
79. Legay, A.; Delahaye, B.; Bensalem, S. Statistical model checking: An overview. In Proceedings of the International Conference on Runtime Verification, St. Julians, Malta, 1–4 November 2010; pp. 122–135.
80. Clarke, E.M.; Henzinger, T.A.; Veith, H.; Bloem, R. *Handbook of Model Checking*; Springer: Berlin/Heidelberg, Germany, 2018; Volume 10.
81. Choi, Y. Model checking trampoline OS: A case study on safety analysis for automotive software. *Softw. Test. Verif. Reliab.* **2014**, *24*, 38–60. [CrossRef]
82. Yamaguchi, T.; Kaga, T.; Donzé, A.; Seshia, S.A. Combining requirement mining, software model checking and simulation-based verification for industrial automotive systems. In Proceedings of the 2016 Formal Methods in Computer-Aided Design (FMCAD), Mountain View, CA, USA, 3–6 October 2016; pp. 201–204.
83. Marinescu, R. Model-Checking and Model-Based Testing of Automotive Embedded Systems: Starting from the System Architecture. Ph.D. Thesis, Mälardalen University, Västerås, Sweden, 2014.
84. Baouya, A.; Mohamed, O.A.; Ouchani, S.; Bennouar, D. Reliability-driven automotive software deployment based on a parametrizable probabilistic model checking. *Expert Syst. Appl.* **2021**, *174*, 114572. [CrossRef]
85. Mundhenk, P.; Steinhorst, S.; Lukasiewycz, M.; Fahmy, S.A.; Chakraborty, S. Security analysis of automotive architectures using probabilistic model checking. In Proceedings of the 52nd Annual Design Automation Conference, San Francisco, CA, USA, 7–11 June 2015; pp. 1–6.
86. Kaliszyk, C.; Urban, J. Learning-assisted theorem proving with millions of lemmas. *J. Symb. Comput.* **2015**, *69*, 109–128. [CrossRef]
87. Kovács, L.; Voronkov, A. First-order theorem proving and Vampire. In Proceedings of the International Conference on Computer Aided Verification, Saint Petersburg, Russia, 13–19 July 2013; pp. 1–35.
88. Harrison, J. *Theorem Proving with the Real Numbers*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012.
89. Loveland, D.W. *Automated Theorem Proving: A Logical Basis*; Elsevier: Amsterdam, The Netherlands, 2016.
90. Harrison, J.; Urban, J.; Wiedijk, F. History of Interactive Theorem Proving. *Comput. Log.* **2014**, *9*, 135–214.
91. Gogate, V.; Domingos, P. Probabilistic theorem proving. *Commun. ACM* **2016**, *59*, 107–115. [CrossRef]
92. Bibel, W. *Automated Theorem Proving*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2013.
93. Cook, S.A. The complexity of theorem-proving procedures. In *Logic, Automata, and Computational Complexity: The Works of Stephen A. Cook*; ACM, 2023; pp. 143–152.
94. Abed, S.; Rashid, A.; Hasan, O. Formal analysis of unmanned aerial vehicles using higher-order-logic theorem proving. *J. Aerosp. Inf. Syst.* **2020**, *17*, 481–495. [CrossRef]

95. Rashid, A.; Hasan, O. Formal analysis of linear control systems using theorem proving. In Proceedings of the Formal Methods and Software Engineering: 19th International Conference on Formal Engineering Methods, ICFEM 2017, Xi'an, China, 13–17 November 2017; pp. 345–361.
96. Rashid, A.; Hasan, O.; Abed, S. Using an Interactive Theorem Prover for Formally Analyzing the Dynamics of the Unmanned Aerial Vehicles. In *Mobile Robot: Motion Control and Path Planning*; Springer: Berlin/Heidelberg, Germany, 2023; pp. 253–282.
97. Lin, Q.; Mitsch, S.; Platzer, A.; Dolan, J.M. Safe and resilient practical waypoint-following for autonomous vehicles. *IEEE Control Syst. Lett.* **2021**, *6*, 1574–1579. [CrossRef]
98. Sousa, M.; Rodríguez, C.; D'Silva, V.; Kroening, D. Abstract interpretation with unfoldings. In Proceedings of the Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, 24–28 July 2017; Proceedings, Part II 30; pp. 197–216.
99. Cousot, P.; Monerau, M. Probabilistic abstract interpretation. In Proceedings of the European Symposium on Programming, Tallinn, Estonia, 28–30 November 2012; pp. 169–193.
100. Fähndrich, M.; Logozzo, F. Static contract checking with abstract interpretation. In Proceedings of the International Conference on Formal Verification of Object-Oriented Software, Paris, France, 28–30 June 2010; pp. 10–30.
101. Cousot, P. *Principles of Abstract Interpretation*; MIT Press: Cambridge, MA, USA, 2021.
102. Cousot, P.; Cousot, R. Abstract interpretation: Past, present and future. In Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), Vienna, Austria, 14–18 July 2014; pp. 1–10.
103. Reps, T.; Thakur, A. Automating abstract interpretation. In Proceedings of the Verification, Model Checking, and Abstract Interpretation: 17th International Conference, VMCAI 2016, St. Petersburg, FL, USA, 17–19 January 2016; pp. 3–40.
104. Brat, G.; Navas, J.A.; Shi, N.; Venet, A. IKOS: A framework for static analysis based on abstract interpretation. In Proceedings of the Software Engineering and Formal Methods: 12th International Conference, SEFM 2014, Grenoble, France, 1–5 September 2014; pp. 271–277.
105. Beckett, R.; Gupta, A.; Mahajan, R.; Walker, D. Abstract interpretation of distributed network control planes. *Proc. ACM Program. Lang.* **2019**, *4*, 1–27. [CrossRef]
106. Giacobazzi, R.; Ranzato, F. History of abstract interpretation. *IEEE Ann. Hist. Comput.* **2021**, *44*, 33–43. [CrossRef]
107. Todorov, V.; Boulanger, F.; Taha, S. Formal verification of automotive embedded software. In Proceedings of the 6th Conference on Formal Methods in Software Engineering, Gothenburg, Sweden, 2 June 2018; pp. 84–87.
108. Quante, J. Use Cases of a Generic Model Interpreter in an Automotive Software Setting. In Proceedings of the 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME), Raleigh, NC, USA, 2–7 October 2016; pp. 539–542.
109. Yamaguchi, T.; Brain, M.; Ryder, C.; Imai, Y.; Kawamura, Y. Application of abstract interpretation to the automotive electronic control system. In Proceedings of the Verification, Model Checking, and Abstract Interpretation: 20th International Conference, VMCAI 2019, Cascais, Portugal, 13–15 January 2019; pp. 425–445.
110. Beller, M.; Bholanath, R.; McIntosh, S.; Zaidman, A. Analyzing the state of static analysis: A large-scale evaluation in open source software. In Proceedings of the 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), Suita, Osaka, Japan, 14–18 March 2016; Volume 1, pp. 470–481.
111. Midtgaard, J.; Møller, A. Quickchecking static analysis properties. *Softw. Test. Verif. Reliab.* **2017**, *27*, e1640. [CrossRef]
112. Kaestner, D.; Schmidt, B.; Schlund, M.; Mauborgne, L.; Wilhelm, S.; Ferdinand, C. Analyze This! Sound Static Analysis for Integration Verification of Large-Scale Automotive Software. Technical Report, SAE Technical Paper. 2019. Available online: https://www.sae.org/publications/technical-papers/content/2019-01-1246/ (accessed on 11 August 2023).
113. Kim, Y.; Lee, D.; Baek, J.; Kim, M. MAESTRO: Automated test generation framework for high test coverage and reduced human effort in automotive industry. *Inf. Softw. Technol.* **2020**, *123*, 106221. [CrossRef]
114. Kurian, E.; Briola, D.; Braione, P.; Denaro, G. Automatically generating test cases for safety-critical software via symbolic execution. *J. Syst. Softw.* **2023**, *199*, 111629. [CrossRef]
115. Ahmed, M.; Safar, M. Symbolic Execution based Verification of Compliance with the ISO 26262 Functional Safety Standard. In Proceedings of the 2019 14th International Conference on Design & Technology of Integrated Systems In Nanoscale Era (DTIS), Mykonos, Greece, 16–18 April 2019; pp. 1–6.
116. Guo, S.; Wu, M.; Wang, C. Symbolic execution of programmable logic controller code. In Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, Paderborn, Germany, 4–8 September 2017; pp. 326–336.
117. Baldoni, R.; Coppa, E.; D'elia, D.C.; Demetrescu, C.; Finocchi, I. A survey of symbolic execution techniques. *ACM Comput. Surv. (CSUR)* **2018**, *51*, 1–39. [CrossRef]
118. Siswantyo, S. Security Analysis and Improvement of Lighweight VANET Authentication Protocol (Case Study: Zhao et al. LVAP). *J. Comput. Netw. Archit. High Perform. Comput.* **2021**, *3*, 135–143. [CrossRef]
119. Zelle, D.; Lauser, T.; Kern, D.; Krauß, C. Analyzing and securing SOME/IP automotive services with formal and practical methods. In Proceedings of the 16th International Conference on Availability, Reliability and Security, Vienna, Austria, 17–20 August 2021; pp. 1–20.
120. Dong, W.; Wang, T.; Zhang, L.; Fan, H. Security protocol analysis based on run modes and Petri net. In Proceedings of the International Conference on Algorithms, Microchips and Network Applications, Zhuhai, China, 18–20 February 2022; Volume 12176; pp. 397–401.

121. Lauser, T.; Zelle, D.; Krauß, C. Security analysis of automotive protocols. In Proceedings of the 4th ACM Computer Science in Cars Symposium, Feldkirchen Germany, 2 December 2020; pp. 1–12.
122. Dhaya, R.; Kanthavel, R.; Venusamy, K. Cloud computing security protocol analysis with parity-based distributed file system. *Ann. Oper. Res.* **2021**, *326* , 1–20. [CrossRef]
123. Cremers, C.; Jacomme, C.; Lukert, P. Subterm-based proof techniques for improving the automation and scope of security protocol analysis. In Proceedings of the 2023 IEEE 36th Computer Security Foundations Symposium (CSF), Dubrovnik, Croatia, 9–13 July 2023; pp. 200–213.
124. Altulaihan, E.A.; Alismail, A.; Frikha, M. A Survey on Web Application Penetration Testing. *Electronics* **2023**, *12*, 1229. [CrossRef]
125. Filiol, E.; Mercaldo, F.; Santone, A. A method for automatic penetration testing and mitigation: A red hat approach. *Procedia Comput. Sci.* **2021**, *192*, 2039–2046. [CrossRef]
126. Johari, R.; Kaur, I.; Tripathi, R.; Gupta, K. Penetration testing in IoT network. In Proceedings of the 2020 5th International Conference on Computing, Communication and Security (ICCCS), Patna, India, 14–16 October 2020; pp. 1–7.
127. Ebert, C.; Ray, R. Penetration Testing for Automotive Cybersecurity. *ATZelectronics Worldw.* **2021**, *16*, 16–22. [CrossRef]
128. Schönhärl, S.; Fuxen, P.; Graf, J.; Schmidt, J.; Hackenberg, R.; Mottok, J. An Automotive Penetration Testing Framework for IT-Security Education. In Proceedings of the Cloud Computing 2022: The Thirteenth International Conference on Cloud Computing, GRIDs, and Virtualization, Special Track FAST-CSP, Barcelona, Spain, 24–28 April 2022; p. 10.
129. Lahami, M.; Fakhfakh, F.; Krichen, M.; Jmaiel, M. Towards a TTCN-3 test system for runtime testing of adaptable and distributed systems. In Proceedings of the Testing Software and Systems: 24th IFIP WG 6.1 International Conference, ICTSS 2012, Aalborg, Denmark, 19–21 November 2012; pp. 71–86.
130. Yurtseven, I.; Bagriyanik, S. A review of penetration testing and vulnerability assessment in cloud environment. In Proceedings of the 2020 Turkish National Software Engineering Symposium (UYMS), Istanbul, Turkey, 7–9 October 2020; pp. 1–6.
131. Khera, Y.; Kumar, D.; Garg, N. Analysis and impact of vulnerability assessment and penetration testing. In Proceedings of the 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), Faridabad, India, 14–16 February 2019; pp. 525–530.
132. McKinnel, D.R.; Dargahi, T.; Dehghantanha, A.; Choo, K.K.R. A systematic literature review and meta-analysis on artificial intelligence in penetration testing and vulnerability assessment. *Comput. Electr. Eng.* **2019**, *75*, 175–188. [CrossRef]
133. Maddala, S.; Patil, S. Agentless automation model for post exploitation penetration testing. In Proceedings of the Intelligent Computing, Information and Control Systems: ICICCS 2019, Madurai, India, 15–17 May 2020; pp. 529–539.
134. Nhu, N.X.; Nghia, T.T.; Quyen, N.H.; Pham, V.H.; Duy, P.T. Leveraging Deep Reinforcement Learning for Automating Penetration Testing in Reconnaissance and Exploitation Phase. In Proceedings of the 2022 RIVF International Conference on Computing and Communication Technologies (RIVF), Ho Chi Minh City, Vietnam, 20–22 December 2022; pp. 41–46.
135. Sweigert, D.; Chowdhury, M.M.; Rifat, N. Exploit Security Vulnerabilities by Penetration Testing. In Proceedings of the 2022 IEEE International Conference on Electro Information Technology (eIT), Mankato, MN, USA, 19–21 May 2022; pp. 527–532.
136. Yi, J.; Liu, X. Deep Reinforcement Learning for Intelligent Penetration Testing Path Design. *Appl. Sci.* **2023**, *13*, 9467. [CrossRef]
137. Zhang, Z.; Towey, D.; Ying, Z.; Zhang, Y.; Zhou, Z.Q. MT4NS: Metamorphic testing for network scanning. In Proceedings of the 2021 IEEE/ACM 6th International Workshop on Metamorphic Testing (MET), Madrid, Spain, 2 June 2021; pp. 17–23.
138. Roy, I.; Sonthalia, S.; Mandal, T.; Kairi, A.; Chakraborty, M. Study on Network Scanning Using Machine Learning-Based Methods. In Proceedings of the International Ethical Hacking Conference 2019: EHaCON 2019, Kolkata, India, 17–25 August 2020; pp. 77–85.
139. Kanta, A.; Coisel, I.; Scanlon, M. A survey exploring open source Intelligence for smarter password cracking. *Forensic Sci. Int. Digit. Investig.* **2020**, *35*, 301075. [CrossRef]
140. Kanta, A.; Coisel, I.; Scanlon, M. PCWQ: A framework for evaluating password cracking wordlist quality. In Proceedings of the International Conference on Digital Forensics and Cyber Crime, Boston, MA, USA, 6–9 December 2021; pp. 159–175.
141. Raman, R.H.A. Enhanced Automated-Scripting Method for Improved Management of SQL Injection Penetration Tests on a Large Scale. In Proceedings of the 2019 IEEE 9th Symposium on Computer Applications & Industrial Electronics (ISCAIE), Malaysia, 27–28 April 2019; pp. 259–266.
142. Liu, M.; Li, K.; Chen, T. Security testing of web applications: A search-based approach for detecting SQL injection vulnerabilities. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, Boston, MA, USA, 9–13 July 2019; pp. 417–418.
143. Bandeira, V.; Rosa, F.; Reis, R.; Ost, L. Non-intrusive fault injection techniques for efficient soft error vulnerability analysis. In Proceedings of the 2019 IFIP/IEEE 27th International Conference on Very Large Scale Integration (VLSI-SoC), Cuzco, Peru, 6–9 October 2019; pp. 123–128.
144. Eslami, M.; Ghavami, B.; Raji, M.; Mahani, A. A survey on fault injection methods of digital integrated circuits. *Integration* **2020**, *71*, 154–163. [CrossRef]
145. Gangolli, A.; Mahmoud, Q.H.; Azim, A. A systematic review of fault injection attacks on iot systems. *Electronics* **2022**, *11*, 2023. [CrossRef]
146. Su, P.; Chen, D. Using fault injection for the training of functions to detect soft errors of dnns in automotive vehicles. In Proceedings of the International Conference on Dependability and Complex Systems, Paris, France, 20–21 September 2022; pp. 308–318.

147. Jha, S.; Banerjee, S.; Tsai, T.; Hari, S.K.; Sullivan, M.B.; Kalbarczyk, Z.T.; Keckler, S.W.; Iyer, R.K. Ml-based fault injection for autonomous vehicles: A case for bayesian fault injection. In Proceedings of the 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Portland, OR, USA, 24–27 June 2019; pp. 112–124.
148. Oakes, B.J.; Moradi, M.; Van Mierlo, S.; Vangheluwe, H.; Denil, J. Machine Learning-Based Fault Injection for Hazard Analysis and Risk Assessment. In Proceedings of the International Conference on Computer Safety, Reliability, and Security, York, UK, 7–10 September 2021; pp. 178–192.
149. Given-Wilson, T.; Jafri, N.; Legay, A. Combined software and hardware fault injection vulnerability detection. *Innov. Syst. Softw. Eng.* **2020**, *16*, 101–120. [CrossRef]
150. Salih, N.K.; Satyanarayana, D.; Alkalbani, A.S.; Gopal, R. A survey on software/hardware fault injection tools and techniques. In Proceedings of the 2022 IEEE Symposium on Industrial Electronics & Applications (ISIEA), Langkawi Island, Malaysia, 16–17 July 2022; pp. 1–7.
151. Cotroneo, D.; De Simone, L.; Natella, R. Thorfi: A novel approach for network fault injection as a service. *J. Netw. Comput. Appl.* **2022**, *201*, 103334. [CrossRef]
152. Shuvo, A.M.; Pundir, N.; Park, J.; Farahmandi, F.; Tehranipoor, M. Ldtfi: Layout-aware timing fault-injection attack assessment against differential fault analysis. In Proceedings of the 2022 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Nicosia, Cyprus, 4–6 July 2022; pp. 134–139.
153. Zhang, M.; Li, H.; Wang, P.; Liu, Q. Parity Check Based Fault Detection against Timing Fault Injection Attacks. *Electronics* **2022**, *11*, 4082. [CrossRef]
154. Liu, X.; Li, X.; Prajapati, R.; Wu, D. Deepfuzz: Automatic generation of syntax valid c programs for fuzz testing. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 1044–1051.
155. Lemieux, C.; Sen, K. Fairfuzz: A targeted mutation strategy for increasing greybox fuzz testing coverage. In Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, Montpellier, France, 3–7 September 2018; pp. 475–485.
156. Liang, J.; Wang, M.; Chen, Y.; Jiang, Y.; Zhang, R. Fuzz testing in practice: Obstacles and solutions. In Proceedings of the 2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER), Campobasso, Italy, 20–23 March 2018; pp. 562–566.
157. Klees, G.; Ruef, A.; Cooper, B.; Wei, S.; Hicks, M. Evaluating fuzz testing. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 2123–2138.
158. Patki, P.; Gotkhindikar, A.; Mane, S. Intelligent fuzz testing framework for finding hidden vulnerabilities in automotive environment. In Proceedings of the 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India, 16–18 August 2018; pp. 1–4.
159. Moukahal, L.J.; Zulkernine, M.; Soukup, M. Vulnerability-oriented fuzz testing for connected autonomous vehicle systems. *IEEE Trans. Reliab.* **2021**, *70*, 1422–1437. [CrossRef]
160. Fowler, D.S.; Bryans, J.; Cheah, M.; Wooderson, P.; Shaikh, S.A. A method for constructing automotive cybersecurity tests, a CAN fuzz testing example. In Proceedings of the 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C), Sofia, Bulgaria, 22–26 July 2019; pp. 1–8.
161. Fowler, D.S.; Bryans, J.; Shaikh, S.A.; Wooderson, P. Fuzz testing for automotive cyber-security. In Proceedings of the 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), Luxembourg, 25–28 June 2018; pp. 239–246.
162. Nyamdelger, T.; Batzorig, M.; Albhelil, E.A.; Koh, Y.; Yim, K. Fuzz Testing and Safe Framework Development for Vehicle Security Analysis. In Proceedings of the International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, Toronto, ON, Canada, 14–17 November 2023; pp. 103–111.
163. Han, J.C.; Zhou, Z.Q. Metamorphic fuzz testing of autonomous vehicles. In Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops, Seoul, Republic of Korea, 27 June–19 July 2020; pp. 380–385.
164. Zhang, H.; Huang, K.; Wang, J.; Liu, Z. CAN-FT: A Fuzz Testing Method for Automotive Controller Area Network Bus. In Proceedings of the 2021 International Conference on Computer Information Science and Artificial Intelligence (CISAI), Kunming, China, 17–19 September 2021; pp. 225–231.
165. Werquin, T.; Hubrechtsen, M.; Thangarajan, A.; Piessens, F.; Mühlberg, J.T. Automated fuzzing of automotive control units. In Proceedings of the 2019 International Workshop on Secure Internet of Things (SIOT), Luxembourg, Luxembourg, 26 September 2019; pp. 1–8.
166. Janičić, M.V.; Plavšić, O.; Brkušanin, M.; Jovanović, P. AUTOCHECK: A Tool For Checking Compliance With Automotive Coding Standards. In Proceedings of the 2021 Zooming Innovation in Consumer Technologies Conference (ZINC), Novi Sad, Serbia, 26–27 May 2021; pp. 150–155.
167. Hicken, A. Mitigate Risk With Leveraging Automotive Development Standards. *ATZelektronik Worldw.* **2018**, *13*, 42–47. [CrossRef]
168. Thompson, C.; Wagner, D. A large-scale study of modern code review and security in open source projects. In Proceedings of the 13th International Conference on Predictive Models and Data Analytics in Software Engineering, Toronto, ON, Canada, 8 November 2017; pp. 83–92.
169. Paul, R. ASTOR: An Approach to Identify Security Code Reviews. In Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering, Rochester, MI, USA, 10 October 2022; pp. 1–3.

170. Oka, D.K. *Building Secure Cars: Assuring the Automotive Software Development Lifecycle*; John Wiley & Sons: Hoboken, NJ, USA, 2021.
171. Braz, L.; Bacchelli, A. Software security during modern code review: The developer's perspective. In Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Singapore, 14–18 November 2022; pp. 810–821.
172. di Biase, M.; Bruntink, M.; Bacchelli, A. A security perspective on code review: The case of chromium. In Proceedings of the 2016 IEEE 16th International Working Conference on Source Code Analysis and Manipulation (SCAM), Raleigh, NC, USA, 2–3 October 2016; pp. 21–30.
173. Assal, H. Collaborative security code review. In Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia, Linz, Austria, 30 November–2 December 2015; pp. 439–444.
174. Alfadel, M.; Nagy, N.A.; Costa, D.E.; Abdalkareem, R.; Shihab, E. Empirical analysis of security-related code reviews in npm packages. *J. Syst. Softw.* **2023**, *203*, 111752. [CrossRef]
175. Damanik, V.N.N.; Sunaringtyas, S.U. Secure code recommendation based on code review result using owasp code review guide. In Proceedings of the 2020 International Workshop on Big Data and Information Security (IWBIS), Depok, Indonesia, 17–18 October 2020; pp. 153–158.
176. Buttner, A.; Piazza, R.; Purohit, R.; Summers, A. A Secure Code Review Retrospective. In Proceedings of the 2020 IEEE Secure Development (SecDev), Virtual Conference, 28–30 September 2020; pp. 31–32.
177. Lawless, W.F.; Mittu, R.; Moskowitz, I.S.; Sofge, D.A.; Russell, S. Cyber-(in) security, revisited: Proactive cyber-defenses, interdependence and autonomous human-machine teams (A-HMTs). In *Adversary-Aware Learning Techniques and Trends in Cybersecurity*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 197–224.
178. Fagbemi, D.D.; Wheeler, D.M.; Wheeler, J.C. *The IoT Architect's Guide to Attainable Security and Privacy*; CRC Press: Boca Raton, FL, USA, 2019.
179. Wang, Z.; Guo, G.; Liu, C.; Zhu, W. Research on Railway DevSecOps System Construction Based on "People-Process-Technology". In Proceedings of the 2022 2nd International Signal Processing, Communications and Engineering Management Conference (ISPCEM), Montreal, ON, Canada, 25–27 November 2022; pp. 19–23.
180. Asha, K.; Harshini, V.; Niroopama, K.; Singh, M.; Rajeshwari, R.; Gagan, B.; Suryanarayana, N.; Venkatesha, M. Analysis of Automotive Security Risk using Cyber Security. In Proceedings of the 2023 International Conference on Network, Multimedia and Information Technology (NMITCON), Bengaluru, India, 1–2 September 2023; pp. 01–07.
181. Bokan, B.; Santos, J. Managing cybersecurity risk using threat based methodology for evaluation of cybersecurity architectures. In Proceedings of the 2021 Systems and Information Engineering Design Symposium (SIEDS), Virtual Conference, 29–30 April 2021; pp. 1–6.
182. Oueslati, H.; Rahman, M.M.; ben Othmane, L. Literature review of the challenges of developing secure software using the agile approach. In Proceedings of the 2015 10th International Conference on Availability, Reliability and Security, Toulouse, France, 24–27 August 2015; pp. 540–547.
183. Loft, P.; He, Y.; Yevseyeva, I.; Wagner, I. CAESAR8: An agile enterprise architecture approach to managing information security risks. *Comput. Secur.* **2022**, *122*, 102877. [CrossRef]
184. Xiong, W.; Lagerström, R. Threat modeling–A systematic literature review. *Comput. Secur.* **2019**, *84*, 53–69. [CrossRef]
185. Xiong, W.; Legrand, E.; Åberg, O.; Lagerström, R. Cyber security threat modeling based on the MITRE Enterprise ATT&CK Matrix. *Softw. Syst. Model.* **2022**, *21*, 157–177.
186. Yeboah-Ofori, A.; Islam, S. Cyber security threat modeling for supply chain organizational environments. *Future Internet* **2019**, *11*, 63. [CrossRef]
187. Zografopoulos, I.; Ospina, J.; Liu, X.; Konstantinou, C. Cyber-physical energy systems security: Threat modeling, risk assessment, resources, metrics, and case studies. *IEEE Access* **2021**, *9*, 29775–29818. [CrossRef]
188. Khan, R.; McLaughlin, K.; Laverty, D.; Sezer, S. STRIDE-based threat modeling for cyber-physical systems. In Proceedings of the 2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe), Torino, Italy, 26–29 September 2017; pp. 1–6.
189. Johnson, P.; Lagerström, R.; Ekstedt, M. A meta language for threat modeling and attack simulations. In Proceedings of the 13th International Conference on Availability, Reliability and Security, Hamburg, Germany, 27–30 August 2018; pp. 1–8.
190. Seeam, A.; Ogbeh, O.S.; Guness, S.; Bellekens, X. Threat modeling and security issues for the internet of things. In Proceedings of the 2019 Conference on Next Generation Computing Applications (NextComp), Mauritius, 19–21 September 2019; pp. 1–8.
191. Karahasanovic, A.; Kleberger, P.; Almgren, M. Adapting threat modeling methods for the automotive industry. In Proceedings of the 15th ESCAR Conference, Hamburg, Germany, 15–16 November 2017; pp. 1–10.
192. Hao, J.; Han, G. On the modeling of automotive security: A survey of methods and perspectives. *Future Internet* **2020**, *12*, 198. [CrossRef]
193. Xiong, W.; Krantz, F.; Lagerström, R. Threat modeling and attack simulations of connected vehicles: Proof of concept. In Proceedings of the Information Systems Security and Privacy: 5th International Conference, ICISSP 2019, Prague, Czech Republic, 23–25 February 2019; pp. 272–287.
194. Aydos, M.; Aldan, Ç.; Coşkun, E.; Soydan, A. Security testing of web applications: A systematic mapping of the literature. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 6775–6792. [CrossRef]

195. Peroli, M.; De Meo, F.; Viganò, L.; Guardini, D. MobSTer: A model-based security testing framework for web applications. *Softw. Test. Verif. Reliab.* **2018**, *28*, e1685. [CrossRef]
196. Malik, J.; Pastore, F. An empirical study of vulnerabilities in edge frameworks to support security testing improvement. *Empir. Softw. Eng.* **2023**, *28*, 99. [CrossRef]
197. Jeannotte, B.; Tekeoglu, A. Artorias: IoT security testing framework. In Proceedings of the 2019 26th International Conference on Telecommunications (ICT), Hanoi, Vietnam, 8–10 April 2019; pp. 233–237.
198. Pfrang, S.; Meier, D.; Kautz, V. Towards a modular security testing framework for industrial automation and control systems: Isutest. In Proceedings of the 2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Limassol, Cyprus, 12–15 September 2017; pp. 1–5.
199. Pekaric, I.; Sauerwein, C.; Felderer, M. Applying security testing techniques to automotive engineering. In Proceedings of the 14th International Conference on Availability, Reliability and Security, Vienna, Austria, 30 July–2 August 2019; pp. 1–10.
200. Mahmood, S.; Fouillade, A.; Nguyen, H.N.; Shaikh, S.A. A model-based security testing approach for automotive over-the-air updates. In Proceedings of the 2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), Porto, Portugal, 24–28 October 2020; pp. 6–13.
201. Sommer, F.; Kriesten, R.; Kargl, F. Survey of Model-Based Security Testing Approaches in the Automotive Domain. *IEEE Access* **2023**, *11*, 55474–55514. [CrossRef]
202. Luo, F.; Zhang, X.; Yang, Z.; Jiang, Y.; Wang, J.; Wu, M.; Feng, W. Cybersecurity testing for automotive domain: A survey. *Sensors* **2022**, *22*, 9211. [CrossRef] [PubMed]
203. Kirk, R.; Nguyen, H.N.; Bryans, J.; Shaikh, S.A.; Wartnaby, C. A formal framework for security testing of automotive over-the-air update systems. *J. Log. Algebr. Methods Program.* **2023**, *130*, 100812. [CrossRef]
204. Mahmood, S.; Nguyen, H.N.; Shaikh, S.A. Systematic threat assessment and security testing of automotive over-the-air (OTA) updates. *Veh. Commun.* **2022**, *35*, 100468. [CrossRef]
205. Faschang, T.; Macher, G. An Open Software-Based Framework for Automotive Cybersecurity Testing. In Proceedings of the European Conference on Software Process Improvement, Grenoble, France, 30 August–1 September 2023; pp. 316–328.
206. Mateo Tudela, F.; Bermejo Higuera, J.R.; Bermejo Higuera, J.; Sicilia Montalvo, J.A.; Argyros, M.I. On Combining Static, Dynamic and Interactive Analysis Security Testing Tools to Improve OWASP Top Ten Security Vulnerability Detection in Web Applications. *Appl. Sci.* **2020**, *10*, 9119. [CrossRef]
207. Hagar, J.D. Security OWASP IoT Information Pointer and Logging Events. In *IoT System Testing: An IoT Journey from Devices to Analytics and the Edge*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 209–215.
208. NIST SP 800-115 | NIST. Available online: https://www.nist.gov/privacy-framework/nist-sp-800-115 (accessed on 11 August 2023).
209. The Open Source Security Testing Methodology Manual. 2010. Available online: https://www.isecom.org/OSSTMM.3.pdf (accessed on 11 August 2023).
210. The Penetration Testing Execution Standard. Available online: http://www.pentest-standard.org/index.php/Main_Page (accessed on 11 August 2023).
211. Information System Security Assessment Framework (ISSAF). Available online: https://www.futurelearn.com/info/courses/ethical-hacking-an-introduction/0/steps/71521 (accessed on 11 August 2023).
212. Musa, H.S.; Krichen, M.; Altun, A.A.; Ammi, M. Survey on Blockchain-Based Data Storage Security for Android Mobile Applications. *Sensors* **2023**, *23*, 8749. [CrossRef] [PubMed]
213. Dehshiri, S.J.H.; Emamat, M.S.M.M.; Amiri, M. A novel group BWM approach to evaluate the implementation criteria of blockchain technology in the automotive industry supply chain. *Expert Syst. Appl.* **2022**, *198*, 116826. [CrossRef]
214. Krichen, M.; Ammi, M.; Mihoub, A.; Al-Haija, Q.A. Short Survey on Using Blockchain Technology in Modern Wireless Networks, IoT and Smart Grids. In Proceedings of the International Conference on Cybersecurity, Cybercrimes, and Smart Emerging Technologies. Springer International Publishing Cham, Riyadh, Saudi Arabia, 10–11 May 2022, pp. 163–173.
Blockchain for the Internet of vehicles: How to use blockchain to secure vehicle-to-everything (V2X) communication and payment? *IEEE Sens. J.* **2021**, *21*, 15807–15823.
215. Lopes, E.J.; Kataria, S.; Keshav, S.; Ikram, S.T.; Ghalib, M.R.; Shankar, A.; Krichen, M. Live video streaming service with pay-as-you-use model on Ethereum Blockchain and InterPlanetary file system. *Wirel. Netw.* **2022**, *28*, 3111–3125. [CrossRef]
216. Dorri, A.; Steger, M.; Kanhere, S.S.; Jurdak, R. Blockchain: A distributed solution to automotive security and privacy. *IEEE Commun. Mag.* **2017**, *55*, 119–125. [CrossRef]
217. Lahami, M.; Maâlej, A.J.; Krichen, M.; Hammami, M.A. A Comprehensive Review of Testing Blockchain Oriented Software. *ENASE* **2022**, *182*, 355–362.
218. Jabbar, R. Fetais, N.; Kharbeche, M.; Krichen, M.; Barkaoui, K.; Shinoy, M. Blockchain for the internet of vehicles: How to use blockchain to secure vehicle-to-everything (v2x) communication and payment. *IEEE Sens. J.* **2021**, *21*, 15807–15823. [CrossRef]
219. Sharma, P.K.; Kumar, N.; Park, J.H. Blockchain-based distributed framework for automotive industry in a smart city. *IEEE Trans. Ind. Inform.* **2018**, *15*, 4197–4205. [CrossRef]
220. Jabbar, R.; Krichen, M.; Shinoy, M.; Kharbeche, M.; Fetais, N.; Barkaoui, K. A model-based and resource-aware testing framework for parking system payment using blockchain. In Proceedings of the 2020 International Wireless Communications and Mobile Computing (IWCMC), Limassol, Cyprus, 15–19 June 2020; pp. 1252–1259.

221. Reddy, K.R.K.; Gunasekaran, A.; Kalpana, P.; Sreedharan, V.R.; Kumar, S.A. Developing a blockchain framework for the automotive supply chain: A systematic review. *Comput. Ind. Eng.* **2021**, *157*, 107334. [CrossRef]
222. Jabbar, R.; Krichen, M.; Fetais, N.; Barkaoui, K. Adopting formal verification and model-based testing techniques for validating a blockchain-based healthcare records sharing system. In Proceedings of the 22nd International Conference on Enterprise Information Systems, Online Streaming, 5–7 May 2020; pp. 261–268.
223. Fraga-Lamas, P.; Fernández-Caramés, T.M. A review on blockchain technologies for an advanced and cyber-resilient automotive industry. *IEEE Access* **2019**, *7*, 17578–17598. [CrossRef]
224. Jabbar, R.; Krichen, M.; Kharbeche, M.; Fetais, N.; Barkaoui, K. A formal model-based testing framework for validating an IoT solution for blockchain-based vehicles communication. In Proceedings of the 15th International Conference on Evaluation of Novel Approaches to Software Engineering, Prague, Czech Republic, 5–6 May 2020; pp. 595–602.
225. Mollah, M.B.; Zhao, J.; Niyato, D.; Guan, Y.L.; Yuen, C.; Sun, S.; Lam, K.Y.; Koh, L.H. Blockchain for the internet of vehicles towards intelligent transportation systems: A survey. *IEEE Internet Things J.* **2020**, *8*, 4157–4185. [CrossRef]
226. Krichen, M.; Lahami, M.; Al-Haija, Q.A. Formal methods for the verification of smart contracts: A review. In Proceedings of the 2022 15th International Conference on Security of Information and Networks (SIN), Sousse, Tunisia, 11–13 November 2022; pp. 01–08.
227. Huang, X.; Ye, D.; Yu, R.; Shu, L. Securing parked vehicle assisted fog computing with blockchain and optimal smart contract design. *IEEE/CAA J. Autom. Sin.* **2020**, *7*, 426–441. [CrossRef]
228. Rathee, G.; Sharma, A.; Iqbal, R.; Aloqaily, M.; Jaglan, N.; Kumar, R. A blockchain framework for securing connected and autonomous vehicles. *Sensors* **2019**, *19*, 3165. [CrossRef] [PubMed]
229. Krichen, M. Strengthening the security of smart contracts through the power of artificial intelligence. *Computers* **2023**, *12*, 107. [CrossRef]
230. Su, Z.; Wang, Y.; Xu, Q.; Fei, M.; Tian, Y.C.; Zhang, N. A secure charging scheme for electric vehicles with smart communities in energy blockchain. *IEEE Internet Things J.* **2018**, *6*, 4601–4613. [CrossRef]
231. Javaid, U.; Aman, M.N.; Sikdar, B. DrivMan: Driving trust management and data sharing in VANETS with blockchain and smart contracts. In Proceedings of the 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring), Kuala Lumpur, Malaysia, 28 April – 1 May 2019; pp. 1–5.
232. Liu, H.; Zhang, Y.; Zheng, S.; Li, Y. Electric vehicle power trading mechanism based on blockchain and smart contract in V2G network. *IEEE Access* **2019**, *7*, 160546–160558. [CrossRef]
233. Chen, C.; Xiao, T.; Qiu, T.; Lv, N.; Pei, Q. Smart-contract-based economical platooning in blockchain-enabled urban internet of vehicles. *IEEE Trans. Ind. Inform.* **2019**, *16*, 4122–4133. [CrossRef]
234. Mihoub, A.; Krichen, M.; Alswailim, M.; Mahfoudhi, S.; Bel Hadj Salah, R. Road Scanner: A Road State Scanning Approach Based on Machine Learning Techniques. *Appl. Sci.* **2023**, *13*, 683. [CrossRef]
235. Ali, E.S.; Hasan, M.K.; Hassan, R.; Saeed, R.A.; Hassan, M.B.; Islam, S.; Nafi, N.S.; Bevinakoppa, S. Machine learning technologies for secure vehicular communication in internet of vehicles: Recent advances and applications. *Secur. Commun. Netw.* **2021**, *2021*, 8868355. [CrossRef]
236. Krichen, M. How artificial intelligence can revolutionize software testing techniques. In Proceedings of the International Conference on Innovations in Bio-Inspired Computing and Applications, Online Streaming, 15–17 December 2022; pp. 189–198.
237. Alkhudaydi, O.A.; Krichen, M.; Alghamdi, A.D. A Deep Learning Methodology for Predicting Cybersecurity Attacks on the Internet of Things. *Information* **2023**, *14*, 550. [CrossRef]
238. Aworka, R.; Cedric, L.S.; Adoni, W.Y.H.; Zoueu, J.T.; Mutombo, F.K.; Kimpolo, C.L.M.; Nahhal, T.; Krichen, M. Agricultural decision system based on advanced machine learning models for yield prediction: Case of East African countries. *Smart Agric. Technol.* **2022**, *2*, 100048. [CrossRef]
239. Qayyum, A.; Usama, M.; Qadir, J.; Al-Fuqaha, A. Securing connected & autonomous vehicles: Challenges posed by adversarial machine learning and the way forward. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 998–1026.
240. Cedric, L.S.; Adoni, W.Y.H.; Aworka, R.; Zoueu, J.T.; Mutombo, F.K.; Krichen, M.; Kimpolo, C.L.M. Crops yield prediction based on machine learning models: Case of West African countries. *Smart Agric. Technol.* **2022**, *2*, 100049. [CrossRef]
241. Krichen, M. Convolutional neural networks: A survey. *Computers* **2023**, *12*, 151. [CrossRef]
242. Park, S.; Choi, J.Y. Malware detection in self-driving vehicles using machine learning algorithms. *J. Adv. Transp.* **2020**, *2020*, 1–9. [CrossRef]
243. Mohseni, S.; Pitale, M.; Singh, V.; Wang, Z. Practical solutions for machine learning safety in autonomous vehicles. *arXiv* **2019**, arXiv:1912.09630.
244. Avatefipour, O.; Al-Sumaiti, A.S.; El-Sherbeeny, A.M.; Awwad, E.M.; Elmeligy, M.A.; Mohamed, M.A.; Malik, H. An intelligent secured framework for cyberattack detection in electric vehicles' CAN bus using machine learning. *IEEE Access* **2019**, *7*, 127580–127592. [CrossRef]
245. Berry, H.; Abdel-Malek, M.A.; Ibrahim, A.S. A machine learning approach for combating cyber attacks in self-driving vehicles. In Proceedings of the SoutheastCon 2021, Online Streaming, 10– 14 March 2021; pp. 1–3.
246. Bendiab, G.; Hameurlaine, A.; Germanos, G.; Kolokotronis, N.; Shiaeles, S. Autonomous vehicles security: Challenges and solutions using blockchain and artificial intelligence. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 3614–3637. [CrossRef]

247. Xun, Y.; Liu, J.; Kato, N.; Fang, Y.; Zhang, Y. Automobile driver fingerprinting: A new machine learning based authentication scheme. *IEEE Trans. Ind. Inform.* **2019**, *16*, 1417–1426. [CrossRef]
248. Madhav, A.S.; Mohan, A.; Tyagi, A.K. IMPROVE: Intelligent Machine Learning based Portable, Reliable and Optimal VErification System for Future Vehicles. In Proceedings of the 2023 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 23–25 January 2023; pp. 1–6.
249. Challita, U.; Ferdowsi, A.; Chen, M.; Saad, W. Machine learning for wireless connectivity and security of cellular-connected UAVs. *IEEE Wirel. Commun.* **2019**, *26*, 28–35. [CrossRef]
250. Uprety, A.; Rawat, D.B.; Li, J. Privacy preserving misbehavior detection in IoV using federated machine learning. In Proceedings of the 2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 9–12 January 2021; pp. 1–6.
251. Ferdowsi, A.; Challita, U.; Saad, W.; Mandayam, N.B. Robust deep reinforcement learning for security and safety in autonomous vehicle systems. In Proceedings of the 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 4–7 November 2018; pp. 307–312.
252. Waheed, N.; He, X.; Ikram, M.; Usman, M.; Hashmi, S.S.; Usman, M. Security and privacy in IoT using machine learning and blockchain: Threats and countermeasures. *ACM Comput. Surv. (CSUR)* **2020**, *53*, 1–37. [CrossRef]
253. Gyawali, S.; Qian, Y. Misbehavior detection using machine learning in vehicular communication networks. In Proceedings of the ICC 2019-2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–6.
254. Said, D.; Elloumi, M.; Khoukhi, L. Cyber-attack on P2P energy transaction between connected electric vehicles: A false data injection detection based machine learning model. *IEEE Access* **2022**, *10*, 63640–63647. [CrossRef]
255. Sharmin, S.; Mansor, H. Intrusion detection on the in-vehicle network using machine learning. In Proceedings of the 2021 3rd International Cyber Resilience Conference (CRC), Langkawi Island, Malaysia, 29–31 January 2021; pp. 1–6.
256. So, S.; Sharma, P.; Petit, J. Integrating plausibility checks and machine learning for misbehavior detection in VANET. In Proceedings of the 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Orlando, FL, USA, 17–20 December 2018; pp. 564–571.
257. Abualsauod, E.H. A hybrid blockchain method in internet of things for privacy and security in unmanned aerial vehicles network. *Comput. Electr. Eng.* **2022**, *99*, 107847. [CrossRef]
258. Tang, F.; Kawamoto, Y.; Kato, N.; Liu, J. Future intelligent and secure vehicular network toward 6G: Machine-learning approaches. *Proc. IEEE* **2019**, *108*, 292–307. [CrossRef]
259. Handa, A.; Sharma, A.; Shukla, S.K. Machine learning in cybersecurity: A review. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2019**, *9*, e1306. [CrossRef]
260. Chai, H.; Leng, S.; Chen, Y.; Zhang, K. A hierarchical blockchain-enabled federated learning algorithm for knowledge sharing in internet of vehicles. *IEEE Trans. Intell. Transp. Syst.* **2020**, *22*, 3975–3986. [CrossRef]