Securing Future In-Vehicle Networks: Monitoring and Control for Ethernet Backbones*

Timo Häckel, Philipp Meyer, Franz Korf, and Thomas C. Schmidt

Dept. Computer Science, Hamburg University of Applied Sciences (HAW), Germany
{timo.haeckel,philipp.meyer,franz.korf,t.schmidt}@haw-hamburg.de

Abstract. Vehicular systems are increasingly controlled by distributed software components that demand a reliable and secure communication infrastructure. Ethernet and Time-Sensitive Networking (TSN) offer a promising foundation for future high-bandwidth, real-time capable In-Vehicle Networks (IVNs). Current IVNs, however, are vulnerable to safety-compromising attacks, calling for a multifaceted security hardening. In this paper, we present a control and monitoring framework that guards safety and security throughout an attack-resistant real-time network architecture with effective misbehavior detection. Software-Defined Networking (SDN) control of TSN networks enables the enforcement of security policies and the dynamic adaptation to changing conditions. A combination of network monitoring techniques, traffic analysis, observable metrics, and detection algorithms tailored for automotive deployment enable the detection of anomalies and attacks. This network-centric approach enhances the security of existing automotive protocols and Electronic Control Units (ECUs) of limited computing power and remains extensible for future demands. Evaluation results from simulations, a hardware test bed, and a real prototypic vehicle demonstrate the feasibility, effectiveness, and remaining challenges of our approach.

Keywords: In-Vehicle Network · Time-Sensitive Networking · Software-Defined Networking · Network Security · Network Monitoring · Anomaly Detection

1 Introduction

Future software-defined vehicle systems [30] require the In-Vehicle Network (IVN) to support high-bandwidth cross-domain communication, *e.g.,* for Advanced Driver Assistance Systems (ADASs). To meet these demands, network and compute resources are centralized and shared among services. A real-time Ethernet backbone [50] with Time-Sensitive Networking (TSN) [22] ensures efficient and reliable communication.

* This work was supported in part by the German Federal Ministry of Education and Research (BMBF) within the SecVI project.

Integrating vehicles into global communication (Vehicle-to-X (V2X)) increases the attack surface, which has been shown in real-world attacks [38], necessitating cybersecurity measures. Industry standards (*e.g.,* ISO/SAE 21434 [24]) and legislation (*e.g.,* the European Cyber Resilience Act [13]) demand security measures throughout the entire vehicle system and lifetime.

Securing vehicle access interfaces, performing plausibility checks, and encrypting critical data are important, but cannot rule out attacks reaching the IVN itself [44]. The IVN is the backbone of the vehicle system, as it connects all Electronic Control Units (ECUs) and enables communication. Manipulation of the IVN can impair the safety of critical driving functions and even harm passengers and other road users. Moreover, attacks can spread throughout the network, necessitating zero-trust in network nodes to prevent their propagation.

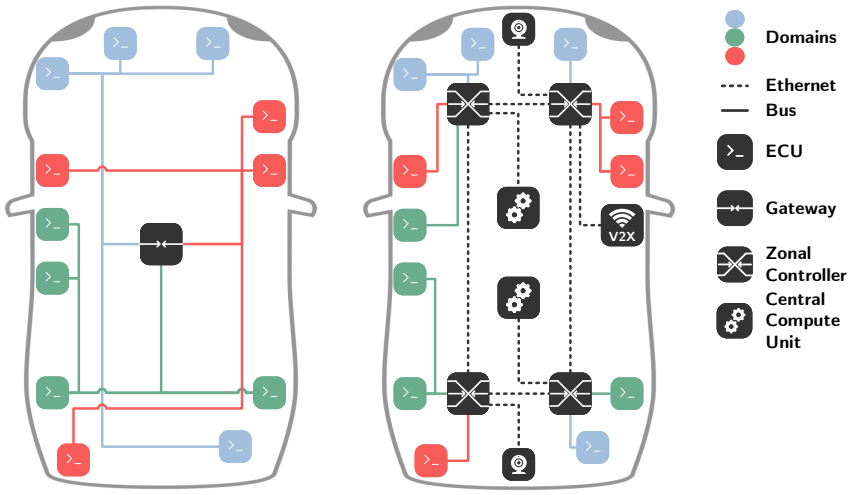
A multilayered defense strategy is crucial to secure the IVN against misbehavior [51]. A network-centric approach ensures security for resource-constrained ECUs and allows secure reuse of legacy devices. Precise control and isolation of network flows are required to minimize the attack surface and prevent attacks from spreading [51]. Effective network monitoring can detect misbehavior and enable a fast response. All security measures must meet the real-time demands of the IVN and remain compatible with existing protocols and devices.

In this work, we propose a monitoring and control architecture enhancing security in future IVN. Our approach encompasses traffic control and network monitoring, which we investigated in various previous work [14–18, 35, 36, 47]. Implementing an attack-resistant real-time network architecture, we ensure both safety and security. To effectively detect misbehavior, we employ a combination of monitoring techniques, network traffic analysis, and observed metrics tailored for automotive deployment.

The remainder of this work is structured as follows: Sec. 2 reviews background and related work. In Sec. 3 we present our control and observe architecture for IVN security. Sec. 4 presents our evaluations in simulation, test beds, and a prototype vehicle, demonstrating the practical application of our approach. Finally, Sec. 5 concludes this work and outlines future work.

2 Background and Related Work

The race towards highly autonomous vehicles requires a new platform for their Electrical/Electronic (E/E) architecture, and the automotive industry is moving towards a software-defined vehicle architecture [30]. Multiple teams create components in a highly modular fashion with increasing complexity for integration and testing, and faster development cycles with frequent updates. The integration of such components requires contracts between components, formal timing specifications, and computation and interaction models [8]. This shift poses new challenges for the design of a secure IVN that can keep up with the increasing dynamics.



(a) Central Gateway Topology (b) Zonal Topology

Fig. 1: IVN topology evolution from separate domains connected via a central gateway (a), to a zonal topology (b) centralizing network and compute resources.

2.1 In-Vehicle Networks

Traditionally, vehicle control applications run on specialized ECUs that are connected via heterogeneous networks, such as Controller Area Network (CAN), Local Interconnect Network (LIN), Media Oriented System Transport (MOST), FlexRay, and Ethernet. The current automotive E/E architecture (*cf.,* Fig. 1a) is divided into separate domains that are interconnected via a central gateway [50]. Examples for these domains include the infotainment, drivetrain or passenger comfort. The increasing number of specialized ECUs required for new functions results in higher hardware overhead, leading to increased vehicle production costs. Additionally, the growing complexity and strict domain separation hinder flexibility.

Future applications require increased communication bandwidth and computational resources, *e.g.,* through high-resolution sensor fusion and ADAS for automated vehicles [10]. To cope with these demands central compute units interconnected via high-bandwidth Ethernet are proposed [6]. In a zonal topology (*cf.,* Fig. 1b), ECUs are grouped into zones according to their placement in the vehicle, *e.g.,* front, rear, and right, left side. Zonal controllers placed at the center of each zone can combine functions of gateways, switches, and applications.

On the software layer this shift to more centralized resources is accompanied by a shift to a dynamic Service-Oriented Architecture (SOA) [27]. Services utilizing virtualized resources in containerized applications are considered for improving the flexibility of the software architecture [27].

Fig. 2 shows a future protocol stack for service-oriented communication via Ethernet [33]. *Scalable service-Oriented MiddlewarE over IP* (SOME/IP) [2] is the most widely deployed middleware for such an automotive SOA. Other candidates such as Data Distribution Service (DDS) [42] are also considered, and available in the AUTomotive Open System ARchitecture (AUTOSAR). Both utilize UDP/TCP/IP for adaptive communication. TSN [22] enhances Ethernet to meet the real-time requirements of the IVN. A header adds virtual LANs and a Priority Code Point (PCP) from zero to seven. TSN devices (*cf.,* Fig. 3) have gates at input and output ports. A closed input gate drops incoming frames and a closed output gate queues frames until it is open. Periodic Gate Control Lists (GCLs) determine opening and closing times of gates. Ingress meters and egress shapers complement the GCL functionality. It is important to note that while the ingress can operate on a per-stream basis, the egress operates on per-priority queuing using the frame PCP and shapes all streams of the same priority together. The switching fabric forwards frames that pass ingress control to the egress control of an outgoing port. Precise time synchronization via IEEE 802.1AS [23] is necessary to synchronize packet transmission and coordinate time between applications, ECUs, and network devices. We add fully programmable options for TSN flows (*cf.,* Sec. 3.1) and utilize the Per-Stream Filtering and Policing (PSFP) for network anomaly detection (*cf.,* Sec. 3.5).

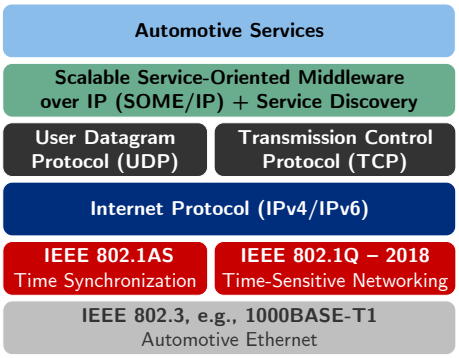


Fig. 2: The automotive protocol stack.

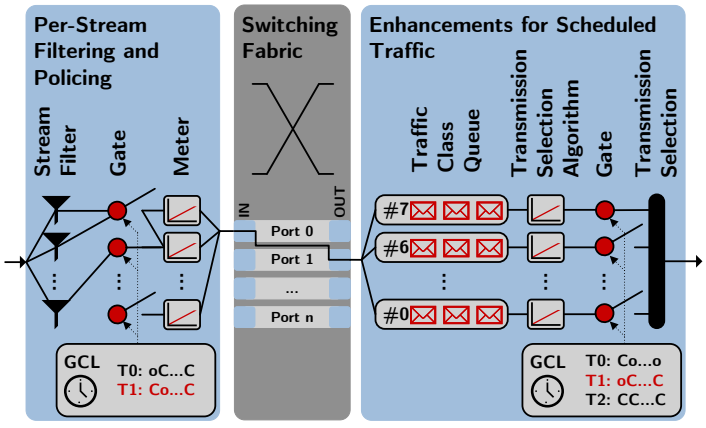


Fig. 3: TSN modules for ingress control and egress shaping.

2.2 In-Vehicle Network Security

Modern cars integrate with global communication, *e.g.,* for V2X communication, which in turn provides a multitude of new interfaces. Current vehicles are vulnerable to manipulation by third parties, which has been demonstrated in the field [38]. A fundamental analysis of the automotive attack surface showing how a variety of interfaces can be used to gain malicious access to in-car devices is provided by Checkoway *et al.* [7]. Manipulation of ECUs and the IVN can compromise the safety of the vehicle, putting passengers at risk.

Attacks on IVNs include Denial of Service (DoS), replay, spoofing, and falsified-information attacks [11, 51]. They can be classified as *alter* attacks (modifying data), *listen* attacks (monitoring data), *disable* attacks (denying services), and *forge* attacks (inserting incorrect data) [39]. Defensive measures can be divided into attack prevention, detection, and mitigation [51]. The key security objectives for IVNs are *availability* (resources are accessible and deadlines are met), *integrity* (accuracy and completeness of data), and *authenticity* (verifiability of data sources and sinks) [11, 39].

Integrity and authenticity can be ensured with authentication and encryption methods suitable for IVNs [21,40]. Encryption, however, cannot protect data flows from all network attacks, *e.g.,* DoS. Traditional in-vehicle ECUs have limited computing power which restricts resources for security features [40]. In addition, established and verified ECUs are used over many vehicle generations. Our network-centric approach establishes an attack-resistant network (*cf.,* Sec. 3.2) and enables intrusion detection (*cf.,* Sec. 3.4 and 3.5) that helps to secure ECUs with limited computing power and allows secure reuse of legacy devices.

2.3 Attack-Resistant Network Architectures

Current multiple-access bus systems such as the CAN bus are missing flow control capabilities and thus do not allow for safety and security measures on the network level. Gateways can only filter messages between different sections of the IVN [49]. Firewalls and access control mechanisms can prevent attacks with stateful inspection, rate limiting, and filtering [45]. Flexible security solutions, such as software-defined security elements, will be beneficial to cope with the growing dynamics of the IVN [11, 48].

As software dynamics increase, they extend to the network level, where flows must be established during runtime. Software-Defined Networking (SDN) [34] is a promising solution to address the challenges posed by the growing dynamics of IVN [14, 15, 19]. In SDN, the control plane is separated from the data plane. While traditional networks focus on individual end devices, SDN takes a more holistic approach forwarding traffic based on flow rules that match packet headers from Layer 2 to Layer 4. The control plane is logically centralized in the SDN controller. This fundamental distinction allows SDN to offer greater flexibility, adaptability, robustness, and security compared to traditional networks [53].

The logically centralized control plane can be physically distributed for hot-standby and failover to avoid a single point of failure [20]. Additionally, security

elements may be vulnerable, especially in software functions. A compromised SDN controller can be used to manipulate the network and bypass security measures. Protection mechanisms have been investigated in the past [20] and are out of scope of this work.

This work explores the potentials and challenges of a software-defined IVN in design, simulation (*cf.,* Sec. 4.1), and practical implementation in a prototype vehicle (*cf.,* Sec. 4.3). The discussed potentials include enforcing real-time guarantees even for dynamic flows (*cf.,* Sec. 3.1), protecting in-vehicle control flows from malicious actors (*cf.,* Sec. 3.2), incident response, and supporting specific automotive protocols within the network (*cf.,* Sec. 3.3).

2.4 Network Intrusion Detection

Network intrusion detection is a crucial component of any cybersecurity strategy [52]. It enables reactive measures to be taken to prevent extended damage. A multitude of approaches, datasets and challenges for network intrusion detection have been proposed [4,28]. In the context of in-vehicle Ethernet networks, the detection of malicious activities is particularly challenging due to domain specific requirements, protocols, and missing large-scale deployments. Thus, matching datasets for validation are rare and because there are just a handful of prototypes with future Ethernet-based IVNs out in the wild concrete attacks are rarely seen so far.

A comprehensive list of algorithm types can be used for network anomaly detection [46]. Many of them are based on Machine Learning (ML) techniques that require validation and verification similar to safety-critical applications for future cars [9]. The real-time pattern of TSN-based Ethernet communication can be beneficial for definition and learning of normal network behavior which can significantly improve performance compared to non-real-time Ethernet environments. It is promising that in-car traffic patterns effectively describe normal behavior and that network anomaly detection methods can be used efficiently to detect malicious activities and even zero-day attacks.

In this work, we explore two potential network anomaly detection architectures for in-vehicle Ethernet networks. Our dedicated network anomaly detection (*cf.,* Sec. 3.4) can leverage advanced ML methods, while network-integrated anomaly detection based on TSN ingress control (*cf.,* Sec. 3.5) exploits the already existing precise configuration of real-time patterns.

3 Control and Observe In-Vehicle Ethernet Backbones

Our control and observe architecture for in-vehicle Ethernet backbones is based on two key concepts: (*i*) Software-Defined Networking (SDN) enables dynamic reconfiguration for precisely separated traffic flows and builds an attack resistant network architecture. (*ii*) Network Anomaly Detection Systems (NADSs) observe network flows, extract key traffic metrics, and detect misbehavior. Fig. 4 illustrates the interaction between these components and the IVN.

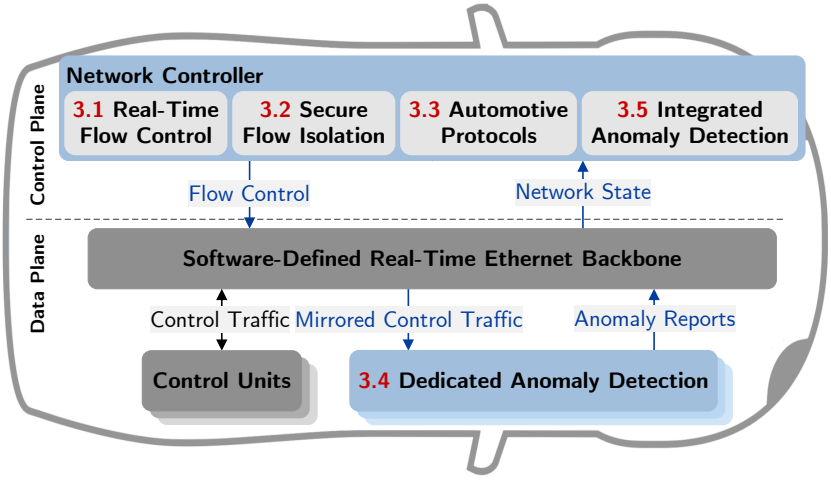


Fig. 4: Overview of the network observation and management architecture.

Deploying SDN in cars logically separates the *control plane* from the *data plane*, introducing a controller with global network knowledge. The control plane is responsible for situational awareness. Specialized control applications program the data plane based on the topology, traffic flows, Quality-of-Service (QoS) requirements and security policies of the IVN. This improves flexibility, with the ability to update and add services, and adapt to changes in communication. The data plane handles the actual data transmission and is subject to real-time requirements that demand deterministic and predictable behavior. It consists of a time-sensitive Ethernet backbone connecting vehicle control units via switches. We demonstrate the applicability of SDN in cars, including central control for real-time flows (*cf.,* Sec. 3.1), secure isolation of vehicle control flows (*cf.,* Sec. 3.2), and integration with existing automotive protocols (*cf.,* Sec. 3.3).

Anomaly detection systems are a subtype of Intrusion Detection Systems (IDSs) designed to identify deviations from normal behavior, which could indicate malicious activity [4]. NADSs monitor the data traffic to detect misbehavior. Defining or learning normal behavior presents a significant challenge. We present two approaches to fingerprint normal traffic and detect anomalies in IVNs: A dedicated NADS based on ML that allows evaluation of varying inputs, metrics, and algorithms (*cf.,* Sec. 3.4). An integrated NADS (*cf.,* Sec. 3.5) that exploits predefined TSN configurations as definition of normal behavior, TSN switch statistics as metrics, and a controller application as the detector.

The control and observe architecture can isolate traffic flows and their paths in the network, while at the same time detect anomalies in allowed traffic flows. Anomaly reports can be used along with network state information for dynamic reconfiguration to mitigate the impact of attacks and ensure that driver, passengers, and manufacturer can be informed [17].

3.1 Software-Defined Real-Time Flow Control

We proposed Time-Sensitive Software-Defined Networking (TSSDN) [14, 15], which combines TSN (*cf.,* Fig. 3) and SDN for dynamic real-time communication. On the data plane, TSN endpoints are connected by switches that integrate SDN forwarding with TSN real-time control: (*i*) PSFP applies filters and time checks to incoming frames, (*ii*) passing frames undergo flow table lookup following SDN forwarding, and (*iii*) TSN egress control shapes outbound traffic.

Control plane functions are offloaded to a central controller that performs SDN address learning, routing, and TSN Stream Reservation (SR) and scheduling. Data plane devices forward packets that do not match any flow rule to the controller, where network applications decide whether to drop the packet, reply directly, or forward it. For forwarding, the application determines a route and installs flow rules on the data plane, enabling independent packet forwarding.

Reliable communication is crucial for safety-critical vehicle traffic. In the TSN configuration, priority queues can be partitioned into static and dynamic queues [32]. A static configuration with a separate flow table ensures correctness and serves as a fail-safe in case of controller failure or security incidents. The SDN controller handles additional dynamic traffic by identifying senders and receivers, verifying permissions, and creating precise flows. Therefore, the control plane requires adaptations for real-time traffic.

Asynchronous real-time flows in TSN use the Stream Reservation Protocol (SRP) to dynamically announce resource requirements across the network. Talkers announce streams with bandwidth demands, and interested listeners subscribe. Devices along the path reserve bandwidth and shape the stream accordingly in a transmission selection algorithm, *e.g.,* Credit Based Shaping (CBS).

We map the centralized SR model (802.1Qcc), which aligns well with SDN, to the OpenFlow protocol [14]. Talkers and listeners still use SRP for announcements, and network devices forward them to the SDN controller. A network application checks available bandwidth updates stream tables as listeners join or leave, programs the switches with flow entries and allocates bandwidth. This ensures correct identification, forwarding, and bandwidth control for the stream [14].

In simulation studies (*cf.,* Sec. 4.1), we found that the actual real-time data transfer performs equally well as with TSN [14]. The dynamic SR, which is not subject to real-time requirements in TSN, experiences a delay from traversing the controller. This delay is generally small compared to the latency introduced by cross-traffic and network schedules [15]. Since the SR is done in advance, real-time flows are already installed in the TSSDN switches, avoiding further controller inspection delays.

Synchronous real-time flows require coordination across multiple transmitters and links. A periodic GCL (*cf.,* Fig. 3) controls priority gates to schedule time slots for transmitters. Offline calculations determine fixed Time Division Multiple Access (TDMA) schedules that cannot accommodate changes in communication as services are activated, shutdown, or updated. In TSSDN, the controller

can modify the GCL schedule and flow paths when synchronous traffic changes. Time slots can be added, removed, or shifted within the period, and complex operations such as rerouting can combine these actions.

At the egress, the GCL is scheduled per priority queue, not per flow. Adding flows to TDMA-scheduled priorities without updating the GCL can cause queue overflow and missed deadlines for critical traffic. This makes the dynamic nature of SDN flow control unsuitable. In contrast, the NETCONF protocol [12] is well-suited as it supports transaction-oriented configurations that ensure isolation and allow verifying modifications before applying changes.

In simulation studies (*cf.,* Sec. 4.1), we showed the need for multi-device transactions to avoid queue overflow and packet loss during schedule reconfiguration [15]. Moving a time slot in the schedule while a packet is in transit can cause the packet to miss its time slot on the next device, causing a delay for subsequent traffic of that priority. Therefore, updates must be executed simultaneously on all network devices or in a specific order based on the schedule modifications.

3.2 Secure Isolation of In-Vehicle Control Flows

Despite the shift from static configuration to dynamic SOA, control flows of automotive services remain well-defined. A control flow consists of related messages with the same unique identifier sent from one origin to one or more receivers via the network. Each control flow belongs to a specific domain (*e.g.,* drivetrain).

The automotive protocol stack (Fig. 2) incorporates a service-oriented middleware (*e.g.,* SOME/IP) that relies on transport, network, and data link layer protocols. Protocol headers include control flow information to indicate the transported information type to network and applications. SDN flows match header fields from layer 2 to 4. The choice of control flow embedding determines whether control flow information remains hidden or exposed to the network.

In the current state of the art, control flows are typically tunneled using an application layer protocol (*e.g.,* SOME/IP), which hides control flow identifiers from the network. This limits separation capabilities and prevents distinguishing between unique control flows in the network. However, control flow embedding can also be fully exposed, with the control flow context embedded in packet header fields used for network forwarding decisions. This enables perfect isolation with one network flow per vehicle control flow. Recently, Nayak *et al.* introduced a P4 [5] programmable data plane that supports SOME/IP [41], making the SOME/IP header information available for forwarding decisions.

In previous work [15, 17, 18, 36], we assessed the impact of control flow embedding on traffic isolation in a realistic software-defined in-vehicle backbone (*cf.,* Sec. 4.3). When devices participate in a hidden application layer tunnel, *e.g.,* for an entire domain, they can receive and send all control flows of this domain. Exposed embedding enhances flow separation and isolation, restricting communication to known and permitted relations and reducing eavesdropping potential. Exposed embedding and SDN-based isolation require attackers to com-

promise the exact sender of each control flow on the Ethernet backbone to issue messages in that channel, which reduces the attack surfaces.

3.3 Network Support for Automotive Protocols

Support for automotive application layer protocols on the SDN control plane optimizes flow control. Intercepting network control protocols is a common practice to enhance networking objectives via SDN, such as improved Address Resolution Protocol (ARP) [1] or IP multicast routing [26]. Bertaux *et al.* [3] propose an initial design for an SDN application that dynamically allocates network resources for DDS services. Such an adaptation is needed for automotive protocols to extend the mentioned advantages of SDN to dynamically discovered services.

SOME/IP is the most widely accepted protocol for automotive SOA. It provides a complementary Service Discovery (SD) to resolve service locations during runtime. Further, it supports two communication schemes: (*i*) publish-subscribe for data required on every change or cyclically, and (*ii*) request-response if a service is needed once or sporadically.

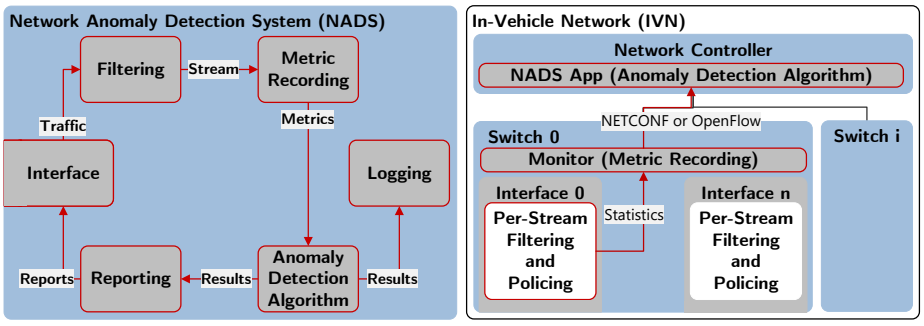
Open challenges and opportunities remain for supporting the SD and communication on the network level. The SOME/IP discovery lacks trust mechanisms, access control, and authentication [25], which could be improved by network intelligence. Lower layer QoS, such as TSN shaping, is not supported by the dynamic SOA. A dedicated group management is needed for multicast groups, which is not supported by regular Ethernet switches. Multiple instances of the same service can exist, *e.g.,* for redundancy or distribution, requiring network-level support for resource reservation and fast handover.

We demonstrated that SDN can support the SOME/IP protocol using the controller as a rendezvous point for SD [16]. This enables in-network caching of service information and automatic path setup for communication, including multicast support. We evaluated the performance of our approach in simulation (*cf.,* Sec. 4.1), comparing its scalability to non-optimized SDN and standard Ethernet switching. The central controller processing all SOME/IP SD messages significantly reduces performance in both SDN approaches, but the delay only affects subscription setup, not the actual data transfer (*cf.,* TSN SR Sec. 3.1). Our approach improved SDN performance by up to 50% compared to the non-optimized SDN solution. This allows for potential extensions of a SOME/IP SD-aware SDN for optimizing SD, service mobility, and reconfiguration mechanisms to achieve improved robustness, QoS support, and security enhancements.

3.4 Dedicated Network Anomaly Detection with mirrored traffic

A dedicated NADS device can be easily added to existing networks. Integration solely requires that forwarding devices can mirror all incoming traffic to an interface with adequate bandwidth. The mirror interface links to a dedicated NADS device (*cf.,* Fig. 4) that captures and processes traffic at line rate.

Fig. 5a shows the architecture of a dedicated NADS instance. Multiple instances operate in parallel to observe multiple streams. Incoming traffic reaches



(a) Dedicated NADS for mirrored traffic. (b) Integrated NADS based on PSFP.

Fig. 5: Different approaches for network anomaly detection in IVNs.

a component that filters out non-observed traffic. Metrics (bandwidth, jitter, average frame size) are extracted from the stream over a certain observation interval and used as input for the anomaly detection component. The anomaly detection component uses ML models (*e.g.,* clustering, outlier detection, autoencoder) to train normal behavior and detect anomalies in the stream based on the chosen metrics. Results are logged and reported notifying the network. In our case, the reports are consumed by the SDN controller, which captures the network-wide threat situation and can initiate countermeasures to mitigate the anomaly, notify the driver, and forward reports to online infrastructure.

The dedicated NADS approach qualifies for straightforward deployment. A modular interface attaches to a simulation environment (*cf.,* Sec. 4.1), a test bed (*cf.,* Sec. 4.2), and a real-world IVN (*cf.,* Sec. 4.3). The different environments enable reproducible scenarios, automatic generation of labeled datasets, manageable real-world conditions, and real driving scenarios.

ML based NADSs with appropriate performance require a substantial number of datasets, which often must be labeled. Obtaining such datasets is challenging because there are no future IVN cars in the wild that experience real attacks. Datasets need to be build based on related data like *e.g.,* attacks on current IVNs or Ethernet networks in other domains. To overcome this, we use simulation (*cf.,* Sec. 4.1) to generate datasets with different IVN architectures and a variety of normal and attack scenarios.

False positives are a common issue in anomaly detection systems. They occur when the system identifies a normal behavior as anomalous. False positives can be particularly problematic in safety-critical in-car networks, as they lead to unnecessary alerts, avoidable countermeasures, and decreased trust in the system. Due to the short event cycles (*e.g.,* < 1 second) in network communication, even small false positive rates (*e.g.,* 0,02%) can lead to multiple false positives during one trip with a vehicle (*e.g.,* > 6 false positives in 8 hours) [17].

3.5 Integrated Network Anomaly Detection with PSFP

Cyber-physical systems such as cars are subject to real-time requirements. This property carries over to the IVN and is implemented on the link layer through TSN protocols. Real-time Ethernet traffic shaping enforces patterns in communication behavior (*e.g.,* , timings, gaps, sequences) that provide the opportunity to fingerprint normal behavior [35].

Our integrated NADS approach utilizes the TSN standard Per-Stream Filtering and Policing (PSFP) to define and enforce stream behaviors of incoming traffic at forwarding devices (*cf.,* Fig. 3). PSFP consists of stream filters that match traffic to stream gates and flow meters. Stream gates can be open or closed, either with a static configuration or synchronized and defined in the associated Gate Control List (GCL). Meters utilize algorithms to enforce stream behavior, such as bandwidth, gap, and size. Individual frames that pass through these stages are forwarded, marked, or dropped dependent on gate states and meter results.

Fig. 5b shows the NADS integration in a TSN network with a central controller. Switches monitor statistics of their active features, such as packet counters. In our case, we use PSFP statistics to monitor stream behavior, including the number of dropped frames. These statistics indicate violations of configured thresholds, *e.g.,* for timing, bandwidth, and frame size. When a frame violates configured thresholds in PSFP and is dropped the statistic for the number of dropped frames is increased. The SDN controller collects these statistics and employs an anomaly detection application to detect abnormal behavior in the specific stream. In a non-SDN network, central collection can still be implemented by traditional network management systems, *e.g.,* using SNMP.

The integrated approach is a lightweight solution for NADS as it utilizes existing network components. Nevertheless, it has certain system requirements: (*i*) the deployed switches must support PSFP, which filters incoming traffic and defines normal traffic per stream. (*ii*) The switches need to be manageable or SDN capable to record statistics, serving as the metric recording component of NADS. (*iii*) A central collector of statistics is required that acts as the detection algorithm of the NADS. In IVNs, all those requirements are in discussion for future deployment regardless of network anomaly detection [6, 19]. Still, the availability of hardware and software components is currently limited. Therefore, we implemented the integrated approach in our simulation environment (*cf.,* Sec. 4.1).

PSFP is configured at design time for ensuring safety and security of critical traffic. It regulates normal traffic behavior, and a correctly configured PSFP will not drop frames during design compliant operation. By utilizing such distinct statistics for anomaly detection, the NADS operates without false positives [35], which is a significant advantage of the integrated approach. The absence of false positives is a solid foundation for automated countermeasures. Moreover, the performance of anomaly detection does not depend on training with large datasets but is solely influenced by the quality of the network design and sophistication of the PSFP configuration.

4 Evaluation in Simulation and Real-World Deployments

The evaluation of the proposed concepts necessitates a multifaceted approach. A simulation environment enables the exploration of new concepts, modifications to standards, and novel algorithms within a controlled setting. A test bed facilitates performance assessment in a controlled hardware environment and verifies their applicability in real-world deployments. A prototype based on a production vehicle serves to demonstrate the feasibility of the proposed concepts.

4.1 In-Car Network Simulation for Reproducible Evaluation

Simulation studies are essential for reproducible evaluations of concepts and algorithms in a controlled environment, particularly in the complex setting of the IVN. A key advantage is the ability to compare results by changing a single parameter at a time, such as topology, traffic patterns, shapers, network control algorithms, and anomaly detection algorithms.

Our simulation environment (*cf.,* Fig. 6) is based on OMNeT++, a discrete event simulator [43]. To simulate future IVNs, we use a combination of frameworks. The INET framework [43] is well-established in the networking community and provides networking modules for a full Ethernet-IP-UDP/TCP stack, including network devices, hosts, and traffic generators. Our research group maintains a rich IVN simulation environment[1], including real-time Ethernet, bus systems such as CAN, and gateways to translation [37]. Recent extensions added Service-Oriented Architecture for Communication over Real-Time Ethernet (SOA4CoRE), and Software-Defined Networking for Communication over Real-Time Ethernet (SDN4CoRE) based on the OpenFlowOMNeTSuite [29].

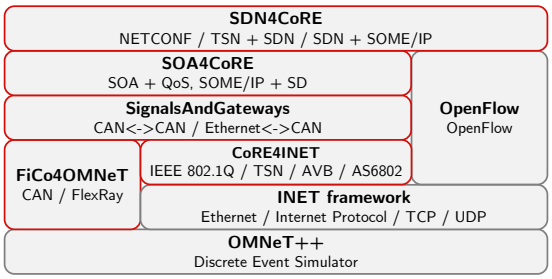


Fig. 6: In-car network simulation environment.

A significant challenge arises from the gap between simulations and real-world implementations. The level of abstraction in the simulation models results in a trade-off between performance and realism. While our simulator accurately represents network communication and control, it does not simulate actual application behavior. To enhance realism, we incorporate traces collected in our prototype vehicle (*cf.,* Sec. 4.3) and model the communication relations of in-car ECUs. Noteworthily, not all simulated features are available in real-world implementations, and the performance of components such as the SDN controller can vary significantly [47]. Further, we observed differences between standards and implementations, *e.g.,* possible combinations of shapers in TSN switches.

[1] CoRE Frameworks for the OMNeT++ Simulator `https://github.com/CoRE-RG`

Simulations provide valuable insights and contribute to the advancement and validation of our research. We modeled our TSSDN architecture (*cf.,* Sec. 3.1), which is not yet available in hardware, and evaluated TSSDN performance for IVN [14, 15]. We successfully implemented protocol extensions, *e.g.,* anomaly detection in the TSN ingress control [35] (*cf.,* Sec. 3.5) and a SOME/IP aware SDN control plane [16] (*cf.,* Sec. 3.3). To support anomaly detection, we collected network traces from our prototype vehicle [15] (*cf.,* Sec. 4.3), which were used for training, testing, and comparing ML algorithms.

4.2 Test Bed for Secure In-Vehicle Communication

Our test bed is a physical setup used to test and evaluate the performance and security of IVN designs. It enables the evaluation of security measures and validates real traffic characteristics for simulations (*cf.,* Section 4.3). Additionally, it serves as preparation for real car deployment (*cf.,* Section 4.3) and indicates hardware performance limitations.

Fig. 7 shows our test bed consisting of an Ethernet backbone with an OpenFlow-enabled switch, four zonal controllers, and a set of ECUs. The switch is divided into two virtual Open vSwitch instances with programmable flow tables. Packet forwarding is controlled by a central ONOS [2] SDN controller equipped with custom applications following our concepts (*cf.,* Sec. 3.2). Each zonal controller is connected to a CAN bus that emulates messages of a production vehicle filtered for the corresponding zone. All CAN messages are tunneled through the backbone to the corresponding destination ECUs and intermediate zonal controllers. One ECU employs an IoT edge node for cloud connectivity [31], and additional ECUs generate high bandwidth traffic such as video and raw LIDAR streams. Incoming packets are mirrored in the switches to NADS instances for traffic monitoring in dedicated devices (*cf.,* Sec. 3.4). Each NADS employs ML to fingerprint communication behavior, and reports violations to the SDN controller.

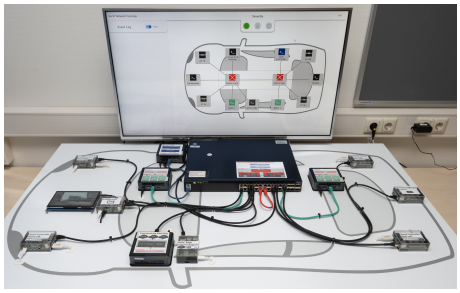


Fig. 7: IVN security test bed.

Working with the test bed poses challenges in emulating real car traffic in various vehicle states. Realistic stimuli involve playing back captured data and creating additional stimuli such as LIDAR and camera streams. Finding suitable hardware, especially switches combining TSN and SDN, was difficult. Even switches that implement either TSN or SDN had limitations in features and stability, restricting evaluations of concepts such as TSSDN and integrated NADS. Furthermore, real pre-compiled automotive ECUs with the AUTOSAR platform

[2] Open Network Operating System (ONOS) by the Open Networking Foundation (ONF): `https://opennetworking.org/onos/`

are impractical for prototyping. Consequently, we chose a reliable SDN-only switch and utilized established prototyping platforms such as the Raspberry Pi.

The test bed allows for experiments and evaluations, including long-term tests that emulate continuous operation without a physical car running 24/7 [17]. This enables comprehensive evaluation of SDN controllers [47], connectivity gateways, and anomaly detection strategies [17]. We performed penetration tests to assess vulnerabilities and potential attacks safely. Variants not supported by conventional automotive software and hardware can be evaluated, for example, security implications of different CAN-Ethernet embeddings [15,18] and gateway strategies that extend the SDN control plane to the gateways.

4.3 A Prototype with Security-Enhanced In-Vehicle Network

In collaboration with our industry partner, we integrated our concepts in a modified production vehicle (*cf.,* Fig. 8). The car includes a security-enhanced network in the trunk showcasing various aspects of the SecVI project[3] [17, 36].

Fig. 8: SecVI prototype with security-enhanced network installed in the trunk.

The installation follows a zone topology with four zonal controllers (left, right, front, rear) serving as gateways between the in-vehicle CAN buses and our Ethernet backbone. CAN messages are assigned to zones based on the position of the sending ECU as defined in the communication matrix of our vehicle. Additional communication along with powerful compute units for tasks such as infotainment and cloud connectivity align the setup with future cars. For monitoring and control of the Ethernet backbone, the prototype utilizes the same setup as the test bed (*cf.,* Sec. 4.2).

The integration of new concepts in the real car required significant effort, making prior evaluations through simulation and test bed essential. Adapting an existing production vehicle for new concepts poses challenges, considering hardware limitations and uncertain future use cases such as ADAS, new sensor data streams, and cloud connectivity. Nonetheless, the prototype demonstrates

[3] German BMBF project SecVI: `https://secvi.inet.haw-hamburg.de/`

concept applicability in real-world deployments while revealing challenges that need to be addressed before integration into production-grade vehicles.

Deploying our network control and monitoring architecture in the prototype vehicle revealed that our robust SDN architecture significantly reduces the attack surface of the IVN by limiting communication flows to the necessary ones while enhancing adaptability [15, 18]. Our ML-based NADS effectively detects attacks in real-world deployments with minimal false positives for control and video traffic [17, 36]. Additionally, our SDN successfully implements countermeasures through network reconfiguration and service mobility to combat attacks [17]. When applying direct countermeasures to anomaly reports, however, future in-car NADS should prioritize minimizing false positives over detection rates to avoid service disruptions in the vehicle.

5 Conclusion and Outlook

Future cars require a robust and secure communication infrastructure. We introduced our control and observe architecture for in-car Ethernet backbones. TSSDN enables real-time flow control, secure isolation of in-vehicle control flows, and network support for automotive protocols. A NADS provides a framework for intrusion detection through machine learning algorithms on dedicated devices, or integrated approaches using TSN ingress control (PSFP), for example.

To ensure accurate evaluations, we use a multifaceted approach: Simulations provide a controlled setting to evaluate new concepts, a test bed facilitates performance assessment in a hardware environment, and a prototype based on a production vehicle serves to demonstrate the feasibility in a real-world deployment. With this we demonstrated the effectiveness of our concepts and algorithms in multiple studies.

Several open issues need to be addressed in future work. Firstly, there is a need to optimize controllers for the vehicle use-case to ensure deterministic behavior in terms of latency and throughput [47]. Further investigation is required to develop protection mechanisms for the controller. Additionally, evaluating TSSDN in a hardware prototype is necessary. Real-time capable NADS algorithms should be developed to enhance intrusion detection, and filtering NADS false positives is important to reduce false alarms. A detailed comparison of different machine learning algorithms for various use cases and traffic types in the vehicle is needed. Furthermore, exploring incident response tailored to IVN is crucial for effectively addressing network security events. Finally, adaptive network configuration and observation are necessary for agile software development in future cars.

References

1. Alharbi, T., Portmann, M.: SProxy ARP - Efficient ARP Handling in SDN. In: 2016 26th International Telecommunication Networks and Applications Conference (ITNAC). IEEE (Dec 2016). https://doi.org/10.1109/atnac.2016.7878805

2. AUTomotive Open System ARchitecture (AUTOSAR) Consortium: SOME/IP Protocol Specification. Tech. Rep. 696, AUTOSAR (Nov 2021). https://www.autosar.org/fileadmin/standards/R22-11/FO/AUTOSAR_PRS_SOMEIPProtocol.pdf
3. Bertaux, L., Hakiri, A., Medjiah, S., Berthou, P., Abdellatif, S.: A DDS/SDN Based Communication System for Efficient Support of Dynamic Distributed Real-Time Applications. In: 2014 IEEE/ACM 18th International Symposium on Distributed Simulation and Real Time Applications. pp. 77–84. IEEE (Oct 2014). https://doi.org/10.1109/ds-rt.2014.18
4. Bhuyan, M.H., Bhattacharyya, D.K., Kalita, J.K.: Network Anomaly Detection: Methods, Systems and Tools. IEEE Communications Surveys & Tutorials **16**(1), 303–336 (Jan 2014). https://doi.org/10.1109/SURV.2013.052213.00046
5. Bosshart, P., Daly, D., Gibb, G., Izzard, M., McKeown, N., Rexford, J., Schlesinger, C., Talayco, D., Vahdat, A., Varghese, G., Walker, D.: P4: Programming Protocol-Independent Packet Processors. SIGCOMM Comput. Commun. Rev. **44**(3), 87–95 (Jul 2014). https://doi.org/10.1145/2656877.2656890
6. Brunner, S., Roder, J., Kucera, M., Waas, T.: Automotive E/E-Architecture Enhancements by Usage of Ethernet TSN. In: 2017 13th Workshop on Intelligent Solutions in Embedded Systems (WISES). pp. 9–13. IEEE (2017). https://doi.org/10.1109/WISES.2017.7986925
7. Checkoway, S., Mccoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., Czeskis, A., Roesner, F., Kohno, T.: Comprehensive Experimental Analyses of Automotive Attack Surfaces. In: Proceedings of the 20th USENIX Security Symposium. vol. 4, pp. 77–92. USENIX Association (Aug 2011). http://www.autosec.org/pubs/cars-usenixsec2011.pdf
8. Damm, W., Ehmen, G., Grüttner, K., Ittershagen, P., Koopmann, B., Poppen, F., Stierand, I.: Multi-Layer Time Coherency in the Development of ADAS/AD Systems: Design Approach and Tooling. In: Proceedings of the Workshop on Design Automation for CPS and IoT (DESTION'19). pp. 20–30. ACM (Apr 2019). https://doi.org/10.1145/3313151.3313167
9. Damm, W., Fränzle, M., Gerwinn, S., Kröger, P.: Perspectives on the Validation and Verification of Machine Learning Systems in the Context of Highly Automated Vehicles. In: 2018 AAAI Spring Symposia. pp. 512–515. AAAI Press (2018).
10. Damm, W., Fränzle, M., Lüdtke, A., Rieger, J.W., Trende, A., Unni, A.: Integrating Neurophysiological Sensors and Driver Models for Safe and Performant Automated Vehicle Control in Mixed Traffic. In: Intelligent Vehicles Symposium (IV 2019). pp. 82–89. IEEE (2019). https://doi.org/10.1109/IVS.2019.8814188
11. Dibaei, M., Zheng, X., Jiang, K., Abbas, R., Liu, S., Zhang, Y., Xiang, Y., Yu, S.: Attacks and defences on intelligent connected vehicles: a survey. Digital Communications and Networks **6**(4), 399–421 (Nov 2020). https://doi.org/10.1016/j.dcan.2020.04.007
12. Enns, R., Bjorklund, M., Schoenwaelder, J., Bierman, A.: Network Configuration Protocol (NETCONF). RFC 6241, IETF (Jun 2011)
13. European Parliament, Council of the European Union: Regulation (EU) 2019/881 of the European Parliament and of the Council of 17 April 2019 on ENISA (the European Union Agency for Cybersecurity) and on information and communications technology cybersecurity certification and repealing Regulation (EU) No 526/2013 (Cybersecurity Act). Official Journal of the European Union (Apr 2019), https://eur-lex.europa.eu/eli/reg/2019/881/oj

14. Häckel, T., Meyer, P., Korf, F., Schmidt, T.C.: Software-Defined Networks Supporting Time-Sensitive In-Vehicular Communication. In: Proc. of the IEEE 89th Vehicular Technology Conference: VTC2019-Spring. pp. 1–5. IEEE (Apr 2019). https://doi.org/10.1109/VTCSpring.2019.8746473
15. Häckel, T., Meyer, P., Korf, F., Schmidt, T.C.: Secure Time-Sensitive Software-Defined Networking in Vehicles. IEEE Transactions on Vehicular Technology **72**(1), 35 – 51 (Jan 2023). https://doi.org/10.1109/TVT.2022.3202368
16. Häckel, T., Meyer, P., Mueller, M., Schmitt-Solbrig, J., Korf, F., Schmidt, T.C.: Dynamic Service-Orientation for Software-Defined In-Vehicle Networks. In: Proc. of the IEEE 97th Vehicular Technology Conference (VTC2023-Spring). IEEE (Jun 2023). https://doi.org/10.1109/VTC2023-Spring57618.2023.10199712
17. Häckel, T., Meyer, P., Stahlbock, L., Langer, F., Eckhardt, S.A., Korf, F., Schmidt, T.C.: A Multilayered Security Infrastructure for Connected Vehicles – First Lessons from the Field. Presented at: 2022 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops). (May 2022). arXiv preprint https://doi.org/10.48550/arXiv.2310.10336
18. Häckel, T., Schmidt, A., Meyer, P., Korf, F., Schmidt, T.C.: Strategies for Integrating Controls Flows in Software-Defined In-Vehicle Networks and Their Impact on Network Security. In: 2020 IEEE Vehicular Networking Conference (VNC). IEEE (Dec 2020). https://doi.org/10.1109/VNC51378.2020.9318372
19. Haeberle, M., Heimgaertner, F., Loehr, H., Nayak, N., Grewe, D., Schildt, S., Menth, M.: Softwarization of Automotive E/E Architectures: A Software-Defined Networking Approach. In: 2020 IEEE Vehicular Networking Conference (VNC). pp. 1–8. IEEE (Dec 2020). https://doi.org/10.1109/VNC51378.2020.9318389
20. Han, T., Jan, S.R.U., Tan, Z., Usman, M., Jan, M.A., Khan, R., Xu, Y.: A comprehensive survey of security threats and their mitigation techniques for next-generation SDN controllers. Concurrency and Computation: Practice and Experience **32**(16), 1–21 (2020). https://doi.org/10.1002/cpe.5300
21. Hu, Q., Luo, F.: Review of Secure Communication Approaches for In-Vehicle Network. International Journal of Automotive Technology **19**(5), 879–894 (Sep 2018). https://doi.org/10.1007/s12239-018-0085-1
22. IEEE 802.1 Working Group: IEEE Standard for Local and Metropolitan Area Network–Bridges and Bridged Networks. Standard Std 802.1Q-2018 (Revision of IEEE Std 802.1Q-2014), IEEE (Jul 2018). https://doi.org/10.1109/IEEESTD.2018.8403927
23. Institute of Electrical and Electronics Engineers: IEEE Standard for Local and Metropolitan Area Networks–Timing and Synchronization for Time-Sensitive Applications. Standard, IEEE (Mar 2020). https://doi.org/10.1109/IEEESTD.2020.9121845
24. International Organization for Standardization: Road vehicles – Cybersecurity engineering. Standard ISO/SAE DIS 21434, ISO, Geneva, CH (2020)
25. Iorio, M., Reineri, M., Risso, F., Sisto, R., Valenza, F.: Securing SOME/IP for In-Vehicle Service Protection. IEEE Transactions on Vehicular Technology **69**(11), 13450–13466 (Nov 2020). https://doi.org/10.1109/tvt.2020.3028880
26. Islam, S., Muslim, N., Atwood, J.W.: A Survey on Multicasting in Software-Defined Networking. IEEE Communications Surveys & Tutorials **20**(1), 355–387 (2018). https://doi.org/10.1109/comst.2017.2776213
27. Kampmann, A., Alrifaee, B., Kohout, M., Wüstenberg, A., Woopen, T., Nolte, M., Eckstein, L., Kowalewski, S.: A Dynamic Service-Oriented Software Architecture for Highly Automated Vehicles. In: 2019 IEEE Intelligent Transportation

Systems Conference (ITSC). pp. 2101–2108. IEEE (2019). https://doi.org/10.1109/ITSC.2019.8916841
28. Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J.: Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity **2**(1) (jul 2019). https://doi.org/10.1186/s42400-019-0038-7
29. Klein, D., Jarschel, M.: An OpenFlow extension for the OMNeT++ INET framework. In: Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques. pp. 322–329. SimuTools '13, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Brussels, BEL (2013)
30. Laclau, P., Bonnet, S., Ducourthial, B., Li, X., Lin, T.: Predictive Network Configuration with Hierarchical Spectral Clustering for Software Defined Vehicles. In: Proc. of the IEEE 97th Vehicular Technology Conference (VTC2023-Spring). IEEE (Jun 2023) https://doi.org/10.1109/VTC2023-Spring57618.2023.10199920
31. Langer, F., Schüppel, F., Stahlbock, L.: Establishing an Automotive Cyber Defense Center. In: 17th escar Europe : embedded security in cars (2019). https://doi.org/10.13154/294-6652
32. Leonardi, L., Bello, L.L., Patti, G.: Bandwidth partitioning for Time-Sensitive Networking flows in automotive communications. IEEE Communications Letters pp. 1–1 (2021). https://doi.org/10.1109/lcomm.2021.3103004
33. Matheus, K., Königseder, T.: Automotive Ethernet. Cambridge University Press, Cambridge (2015)
34. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: OpenFlow: Enabling Innovation in Campus Networks. ACM SIGCOMM Computer Communication Review **38**(2), 69–74 (2008). https://doi.org/10.1145/1355734.1355746
35. Meyer, P., Häckel, T., Korf, F., Schmidt, T.C.: Network Anomaly Detection in Cars based on Time-Sensitive Ingress Control. In: Proc. of the IEEE 21th Vehicular Technology Conference: VTC2020-Fall. IEEE (Oct 2020). https://doi.org/10.1109/VTC2020-Fall49728.2020.9348746
36. Meyer, P., Häckel, T., Langer, F., Stahlbock, L., Decker, J., Eckhardt, S.A., Korf, F., Schmidt, T.C., Schüppel, F.: Demo: A security infrastructure for vehicular information using SDN, intrusion detection, and a defense center in the cloud. In: 2020 IEEE Vehicular Networking Conference (VNC). IEEE (Dec 2020). https://doi.org/10.1109/VNC51378.2020.9318351
37. Meyer, P., Korf, F., Steinbach, T., Schmidt, T.C.: Simulation of Mixed Critical Invehicular Networks. In: Virdis, A., Kirsche, M. (eds.) Recent Advances in Network Simulation. EAI/Springer Innovations in Communication and Computing (May 2019), https://link.springer.com/chapter/10.1007/978-3-030-12842-5_10
38. Miller, C., Valasek, C.: Remote Exploitation of an Unaltered Passenger Vehicle. Black Hat USA **2015**, 91 (2015), https://ericberthomier.fr/IMG/pdf/remote_car_hacking.pdf
39. Monteuuis, J.P., Boudguiga, A., Zhang, J., Labiod, H., Servel, A., Urien, P.: SARA: Security Automotive Risk Analysis Method. In: Proceedings of the 4th ACM Workshop on Cyber-Physical System Security. pp. 3–14. CPSS '18, ACM (2018). https://doi.org/10.1145/3198458.3198465
40. Mundhenk, P.: Security for Automotive Electrical/Electronic (E/E) Architectures. Cuvillier, Göttingen (Aug 2017). https://doi.org/10.32657/10220/45957
41. Nayak, N., Ambalavanan, U., Thampan, J.M., Grewe, D., Wagner, M., Schildt, S., Ott, J.: Reimagining Automotive Service-Oriented Communication: A Case Study

on Programmable Data Planes. IEEE Vehicular Technology Magazine pp. 69–79 (Jan 2023). https://doi.org/10.1109/mvt.2022.3225787
42. Object Management Group: Data Distribution Service. Standard DDS 1.4, OMG (Mar 2015), http://www.omg.org/spec/DDS/1.4
43. OpenSim Ltd.: OMNeT++ Discrete Event Simulator and the INET Framework, https://omnetpp.org/
44. Pekaric, I., Sauerwein, C., Haselwanter, S., Felderer, M.: A taxonomy of attack mechanisms in the automotive domain. Computer Standards & Interfaces **78**, 103539 (Oct 2021). https://doi.org/10.1016/j.csi.2021.103539
45. Pesé, M.D., Schmidt, K., Zweck, H.: Hardware/Software Co-Design of an Automotive Embedded Firewall. In: SAE Technical Paper. SAE International (Mar 2017). https://doi.org/10.4271/2017-01-1659
46. Rajbahadur, G.K., Malton, A.J., Walenstein, A., Hassan, A.E.: A Survey of Anomaly Detection for Connected Vehicle Cybersecurity and Safety. In: 2018 IEEE Intelligent Vehicles Symposium (IV). IEEE (Jun 2018). https://doi.org/10.1109/ivs.2018.8500383
47. Rotermund, R., Häckel, T., Meyer, P., Korf, F., Schmidt, T.C.: Requirements Analysis and Performance Evaluation of SDN Controllers for Automotive Use Cases. In: 2020 IEEE Vehicular Networking Conference (VNC). IEEE (Dec 2020). https://doi.org/10.1109/VNC51378.2020.9318378
48. Rumez, M., Grimm, D., Kriesten, R., Sax, E.: An Overview of Automotive Service-Oriented Architectures and Implications for Security Countermeasures. IEEE Access **8**, 221852–221870 (2020). https://doi.org/10.1109/ACCESS.2020.3043070
49. Rumez, M., Duda, A., Grunder, P., Kriesten, R., Sax, E.: Integration of Attribute-based Access Control into Automotive Architectures. In: 2019 IEEE Intelligent Vehicles Symposium (IV). IEEE (Jun 2019). https://doi.org/10.1109/ivs.2019.8814265
50. Steinbach, T.: Ethernet-basierte Fahrzeugnetzwerkarchitekturen für zukünftige Echtzeitsysteme im Automobil. Springer Vieweg, Wiesbaden (Oct 2018). https://doi.org/10.1007/978-3-658-23500-0
51. Thing, V.L.L., Wu, J.: Autonomous Vehicle Security: A Taxonomy of Attacks and Defences. In: 2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). IEEE (Dec 2016). https://doi.org/10.1109/ithings-greencom-cpscom-smartdata.2016.52
52. Waszecki, P., Mundhenk, P., Steinhorst, S., Lukasiewycz, M., Karri, R., Chakraborty, S.: Automotive Electrical and Electronic Architecture Security via Distributed In-Vehicle Traffic Monitoring. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **36**(11), 1790–1803 (Nov 2017). https://doi.org/10.1109/TCAD.2017.2666605
53. Yurekten, O., Demirci, M.: SDN-based cyber defense: A survey. Future Generation Computer Systems **115**, 126–149 (Feb 2021). https://doi.org/10.1016/j.future.2020.09.006