RIPOSTE: A Collaborative Cyber Attack Response Framework for Automotive Systems

Rodi Jolak*†‡, Thomas Rosenstatter†§, Saif Aldaghistani†, Riccardo Scandariato¶
*Gothenburg University, Sweden, {firstname.lastname}@cse.gu.se
†Chalmers University of Technology, Sweden, {firstname.lastname}@chalmers.se
‡Volvo Car Corporation, Sweden, {firstname.lastname}@volvocars.com
§RISE Research Institutes of Sweden, Sweden, {firstname.lastname}@ri.se
¶Hamburg University of Technology, Germany, {firstname.lastname}@tuhh.de

***Abstract*—The automotive domain has got its own share of advancements in information and communication technology, providing more services and leading to more connectivity. However, more connectivity and openness raise cyber security and safety concerns. Indeed, services that depend on online connectivity can serve as entry points for attacks on different assets of the vehicle. This study explores collaborative ways of selecting response techniques to counter real-time cyber attacks on automotive systems. The aim is to mitigate the attacks more quickly than a single vehicle would be able to do, and increase the survivability chances of the collaborating vehicles. To achieve that, the design science research methodology is employed. As a result, we present RIPOSTE, a framework for collaborative real-time evaluation and selection of suitable response techniques when an attack is in progress. We evaluate the framework from a safety perspective by conducting a qualitative study involving domain experts. The proposed framework is deemed slightly unsafe, and insights into how to improve the overall safety of the framework are provided.**

***Index Terms*—automotive security, response techniques, connected car, vehicle collaboration, resilient systems**

I. Introduction

A vehicle is a network of nodes, so-called Electronic Control Units (ECUs), that delivers different driving functionality and services. With the introduction of 5G cellular technology, automotive systems will have an increased level of automation, including tighter integration with other vehicles, traffic infrastructure, and different cloud services [1]. For example, Vehicle-to-Vehicle (V2V) communication paves the way for new opportunities, e.g., a more efficient and safer crossing of intersections. However, the wide connectivity through Internet services raises cyber security concerns, as online vehicle interfaces provide attackers with entry points that can be used to perform attacks on vehicles. To deal with this issue, a great effort has been spent to establish guidelines and standards for securing automotive systems. In 2017, the European Union Agency for Network and Information Security (ENISA) published cyber security guidelines for the automotive domain [2]. Additionally, the ISO standard ISO/SAE 21434 [3] for cyber security engineering for road vehicles was released in 2021.

To comply with these guidelines and standards, different resilience techniques are proposed for cyber attack *detection*, *analysis*, and *response*. Stojanović et al. [4] define cyber attack detection as the ability for a system to recognize unauthorized activity or access that happens in a network-based environment. Ochieng [5] describes cyber attack analysis as *"the process of assessing the cyber activities and capabilities of unknown intelligent entities or criminals"*; this is to measure the impact of a cyber attack and its feasible path within the system. As for cyber attack response, it is the process of blocking, quarantining, or dealing with a threat that is identified by the detection system, preferably with a minimum amount of side effects i.e., losses caused to the system.

Current automotive resilience approaches are mostly about making the single vehicle more resilient. Moreover, there exists little work investigating vehicle-to-vehicle collaboration, yet it focuses on attack detection rather than attack response.

In this study, we explore vehicle-to-vehicle collaboration for real-time attack response. We first explore available response techniques for cyber security attacks against automotive systems. Second, we investigate collaborative ways for evaluating these attack response techniques and finding which technique is most suitable for a specific ongoing attack.

The following research questions are addressed:

RQ1: *How can we support online collaborative cyber attack response between vehicles to select the most efficient and effective response technique when an attack is in progress?*

This research question aims to investigate the design and implementation of collaboration between vehicles against a cyber attack by evaluating different suitable cyber attack response techniques when an attack is in progress. The goal of this collaboration is to counter attacks more quickly and increase the survivability chances of the collaborating vehicles.

RQ2: *What is the perception of experts on the safety of the proposed collaborative attack-response framework?*

Safety is one of the top priorities of vehicle manufacturers since vehicles are safety-critical systems. Thus, this research question aims to look for safety concerns and other dependability related considerations (e.g., security and availability) when implementing the proposed solution by conducting a qualitative study with security and safety experts.

Section II presents the background and related work. Section III describes the employed methodology. Section IV presents the RIPOSTE framework. Section V describes the implementation and testing of the framework. Section VI reports the evaluation. Section VII presents concluding remarks.

II. Background and Related Work

Nowadays, software is gaining central importance in the process of vehicular applications and services development.

It is distributed and executed on multiple ECUs that reside inside the car. These ECUs regulate certain vehicle' functions by processing information from sensors and actuators and by communicating with other ECUs through a network. These functions can range from simple to sophisticated automated operations, such as controlling the windshield wipers, monitoring driver alertness, and automated parking system.

Modern vehicles can connect not only to the Internet but also to their surroundings. Vehicle connectivity is often referred to as Vehicle-to-Everything (V2X) communication which is divided into multiple subcategories. Related to our work are Vehicle-to-Vehicle (V2V) and Vehicle-to-Cloud (V2C) communication.

Vehicle-to-Vehicle (V2V) communication allows vehicles to exchange information in real-time and share data, such as speed, destination, and location. When a vehicle starts V2V communication, it becomes a node inside a mesh network in which it can capture, send, and forward packets.

Vehicle-to-Cloud (V2C) communication provides data exchange with the cloud through broadband cellular mobile networks. Some applications that leverage this communication include Over-the-Air (OTA) updates; a way to remotely update the software of the vehicle.

Rosenstatter et al. [6] discuss four types of assets within a vehicle that can be compromised by an attacker through spoofing, tampering, repudiation, information disclosure, denial of service, or elevation of privilege [7]. These assets are: hardware, software, network/communication, and data storage.

Hardware can be divided into *ECUs*, *sensors*, and *actuators*. *Attack example:* Tampering existing hardware inside the vehicle. This can act as a mediator and enable the possibility to gain complete control over the vehicle [6].

Software can exist in different states: *in-transit*, *at-rest* or *running*. In-transit can be related to software provisioning systems, like OTA. The running and at-rest categories can be related to software installation processes or software that runs in ECUs. *Attack example:* Software vulnerabilities could be exploited via a privilege escalation attack, which could lead to reprogramming an ECU and include adding a backdoor [6].

Network/Communication can be divided into *in-vehicle* communication (e.g., CAN, FlexRay, MOST, and LIN) and *inter-vehicle* communication (e.g., Wi-Fi, Bluetooth, and V2X). *Attack example:* Denial-of-service attack [6].

Data Storage can store sensitive data, e.g., system information and cryptographic keys. *Attack example:* Secret keys can be exploited in order to disable a firewall in the vehicle [6].

To ensure the security and safety of the aforementioned assets from malicious attacks, cyber security principles should be applied to the various components of the vehicle.

A. Related Work

This section presents work related to response techniques, vehicle collaboration in cyber security, and continuous experimentation which helps to evaluate attack response techniques.

1) Response Techniques: For identifying cyber attack response techniques, Rosenstatter et al. [6] present a systematic literature review that proposes a framework "REMIND" to support the design of resilient automotive systems. The study provides a taxonomy of state-of-the-art techniques for cyber attack detection, mitigation, recovery, and endurance. It also discusses the trade-offs when using certain cyber attack response techniques.

Ratasich et al. [8] provide an overview of the state-of-the-art mechanisms for resilience of IoT devices that require monitoring and controlling from a distance. It summarizes the state-of-the-art techniques on cyber attack detection, diagnosis, and recovery/mitigation by mainly focusing on non-intrusive methods. This study also states the challenges when applying these techniques in the IoT and describes a road map on how to achieve resilience for these devices.

2) Vehicle Collaboration in Cyber Security: While there is almost no literature regarding collaborative cyber attack response of vehicles, there exist few studies concerned with vehicle collaboration in other fields of cyber security.

Mousavinejad et al. [9] propose a distributed attack detection and recovery mechanism to address the problem of detecting cyber attacks that target the vehicle platooning system. The proposed solution is mainly divided into two parts: attack *detection* and *recovery*. The attack detection focuses on attacks that compromise sensor measurements and/or control command data. The recovery mechanisms provided by the study depend on reliable modifications of signals of the vehicles that are attacked.

Other related work is based on collaborative Intrusion Detection Systems (IDS). Nandy et al. [10] propose a trust-based collaborative IDS in which each vehicle keeps a score table of other vehicles in order to identify their previous patterns of the network behavior. The neighboring vehicles would then share their score tables with each other using the Vehicular Ad-hoc network (VANET).

Another work regarding collaborative IDS is conducted by Raja et al. [11]. It proposes the use of a distributed machine learning (DML) model that is based on the alternating direction method of multipliers. This to leverage the V2V collaboration in the learning processes in order to improve the accuracy, scalability, and storage efficiency. The work also targets privacy risks associated with the DML-based collaborative IDS.

3) Continuous Experimentation in the automotive domain: While continuous experimentation is mainly used for software-intensive web-based applications, it also found its way into cyber-physical systems, as they grow to become more software-dependent. A study by Giaimo et al. [12] is conducted to introduce continuous experimentation to the field of cyber-physical systems, on the example of the automotive domain. The study demonstrates and evaluates a prototype infrastructure that is implemented on a distributed computational system in a commercial truck that is used daily on public roads. The system contains units and sensors, and the software deployment and data retrieval processes are done remotely via a mobile data connection. The study shows that the development team was able to apply software deployments based on real-world data that is collected during the experiment. Hence, proving the applicability of continuous experimentation in the automotive domain.

III. Research Methodology

This study aims to provide a framework for evaluating cyber attack response techniques and selecting the most suitable response for an ongoing attack on vehicles. The purpose is to counter the attacks more quickly and increase the survivability chances of the collaborating vehicles. To achieve that, the design science research methodology [13] is employed since it helps to understand and improve a human-made design (e.g., framework) in an area of practice through problem conceptualization, solution design, and solution evaluation [14].

Problem Conceptualization seeks to understand the problem space by using data collection methods. For this study, the main taxonomies of cyber attack response techniques are elicited from the "REMIND" research paper [6], as it is the most recent systematic literature review available in the literature regarding this matter.

Solution Design aims to create a potential solution for the problem using the knowledge gained from the previous step. Here, brainstorming and an investigation of few technical topics and mechanisms are conducted in order to support the decisions of designing the proposed solution.

After designing the artifact, a simulation is conducted to test it. Veanble et al. [15] propose two testing methods, naturalistic (e.g., observation) and artificial (e.g., simulation or experiment). Naturalistic can be complex due to the nature of reality and the number of variables included. Thus, it can include the risk of misinterpretation. Artificial testing, on the other hand, can limit the risk of misinterpretation, but on the cost of applicability in a real scenario. The choice of an artificial testing is made instead of naturalistic due to the limited resources available when conducting the study.

Solution Evaluation helps to evaluate the proposed design. For this study, the proposed solution is evaluated with domain experts through a qualitative study

The qualitative study is conducted through the use of a video demonstration and questionnaire to collect viewpoints in a structured way [16]. The video demonstration is an eight minutes long video providing background information about the problem and proposed solution[1]. The questionnaire includes a combination of closed and open-ended questions in order to gather qualitative data and an indication of intensity of the answers. Security and/or safety experts from different organizations are invited to participate with the purpose of evaluating the safety of the proposed solution based on perceptions. In particular, emails are sent to experts including: (i) the *video demonstration*, (ii) the slides used in the video demonstration for reference, and (iii) a link to the *questionnaire*. The evaluation material are provided online[1].

[1]Evaluation Material: https://doi.org/10.5281/zenodo.6617458

Thematic analysis is applied to analyze qualitative data elicited from the questionnaire. Thematic analysis is a method that is used to identify, analyze, and report patterns (i.e., themes) within the data [17]. First, high-order themes are identified based on the topic/subject of the open-ended questions in the questionnaire. These high-order themes are: (i) automated experiments without human interaction, (ii) qualification of workshop cars, (iii) trustworthiness of results, and (iv) framework design (see Figure 3). After that, the participants' responses to the questionnaire are thoroughly analyzed and a coding process is performed to assign the coded data to the high-order themes or to a newly emerged theme during the process.

A. Participants

The participants of this study are selected following convenience sampling, which is a non-probabilistic type of sampling where the target population meets certain practical criteria [18]. The participants are chosen through the following criteria: knowledge as well as experience in the security or safety of automotive systems, and their willingness to participate. Table I lists the details of the participants who took a part in the evaluation of RIPOSTE.

TABLE I
Participants in the Evaluation of RIPOSTE

Expert	Edu. Degree	Domain of the Degree	Occupation	Experience in SSS*	Expertise in SSS*
A	B.Sc.	Computer Science and Engineering	Principal engineer in cybersecurity at an automotive company	84 months	Very High
B	M.Sc.	Computer Science and Engineering	Ph.D. student in networks and systems working in automotive	45 months	High
C	Ph.D.	Computer Science and Engineering	Research engineer in verification and validation of safety and security	100 months	High
D	Ph.D.	Vehicular Communication Security	Technical Specialist Automotive Cyber Security	96 months	High

***SSS**: System Security and Safety

IV. RIPOSTE

This section describes the design of RIPOSTE, a framework that helps in deciding which attack response technique to deploy against an ongoing attack by collaborating with other vehicles. We first identify attack response techniques in Section IV-A. We discuss the design rationale for RIPOSTE in Section IV-B. In Sections IV-C and IV-D we further detail the structure and behavior of the framework.

A. Response Techniques

Each asset type in automotive systems (see Section II) requires different response techniques to counter attacks and mitigate their effects. In RIPOSTE, we focus on identifying the best suited response techniques for attacks on the *software*

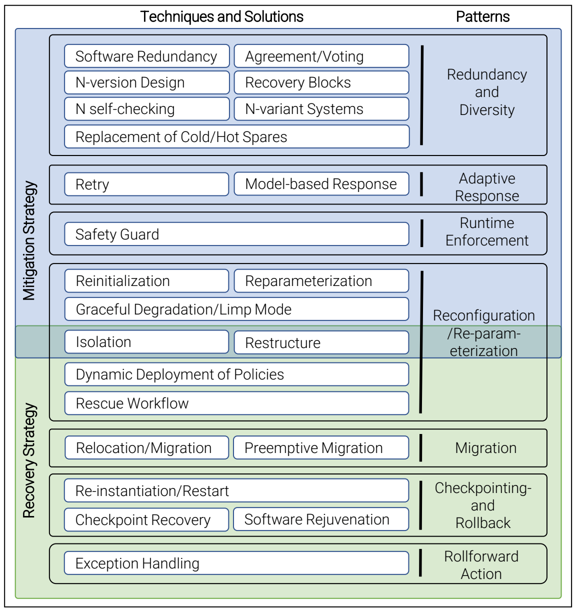

Fig. 1. Response techniques for software assets [6].

asset. Thus, we focus on the *Mitigation* and *Recovery* strategies presented in REMIND [6] and illustrated in Figure 1.

The *mitigation strategy* aims at triggering techniques when anomalies are detected and located. These techniques will keep the system operational, but could result in a non-optimal state. The different patterns for the mitigation strategy are *Redundancy*, *Diversity*, *Adaptive Response*, *Runtime Enforcement*, and *Reconfiguration/Re-parameterization*. The last pattern overlaps with the recovery strategy as reconfiguration can be used for both, threat mitigation and recovery. The main goal of *recovery* is to transition the system back to the desired state. It is also broken down into different patterns: *Reconfiguration/Re-parameterization*, *Migration*, *Checkpointing and Rollback*, and *Rollforward Actions*.

B. Design Rationale for RIPOSTE

For the network architecture we have considered two alternatives: (i) Vehicular Ad-hoc networks (VANETs) and (ii) a centralized, i.e., client-server, architecture.

In VANETs, the vehicles organize themselves in an overlay network to share information and resources, e.g., by using IEEE 802.11-2016 OCB. Client-server architectures are based on the concept of services provided to the clients through servers connected to Internet. The servers are also responsible for authentication, authorization, and resource management.

For RIPOSTE, a centralized approach is more suitable for different reasons: (i) security is managed by a central authority; (ii) the connection is more stable considering the ephemeral nature of VANETs; and (iii) infrastructure for providing services that communicate directly with the vehicle, such as over-the-air updates and remote diagnostics, are already in place and could be expanded to include RIPOSTE.

Our framework is inspired by continuous experimentation for updating and applying the best suited response technique. Schermann et al. [19] identified two techniques for implementing experimentation: *code-level* techniques, where multiple versions of the same code exist within the same code base (known as *feature toggles*), and *deployment-based* techniques, where multiple instances of a service with different software versions are running in parallel (known as *traffic routing*).

Feature toggles are a code-level experimentation technique [19], [20] allowing the modification of a system behavior without changing code [20]. In the simplest form, they can be conditional statements that decide which code to execute next [19]. However, the simplicity comes with a cost, mainly technical debt caused by dead code and additional maintenance [21]. Another challenge with feature toggles is to synchronize the experimentation state and setup and to make sure that multiple instances can be toggled to new versions simultaneously [19].

Traffic routing deploys multiple versions of an application that run in parallel (e.g., in containers or multiple cloud instances). Depending on the filter criteria that are applied to user requests, dynamically configured components (e.g., network-level proxies) decide which version of the software should be forwarded. The main advantage is that this technique is non-intrusive on the code level, avoiding technical debt. However, deploying multiple instances can be costly (e.g., CPU and bandwidth usage). Moreover, the intermediate components that decide the routing path (e.g., proxies) introduce overhead that needs to be taken into account [19].

For RIPOSTE, feature toggles are better suited for two reasons: (i) the implementation of various response techniques in the codebase cannot be considered as overhead as each technique can be used to protect against a particular type of attack; (ii) the vehicle's response to an attack can be updated immediately (e.g., via a command), while traffic routing could require the system to be updated, which could take minutes until the specific version of the software is downloaded, installed, and executed.

C. RIPOSTE: Architectural Components

The deployment diagram in Figure 2 shows the main components and devices comprising RIPOSTE. The *onRoadCar* device is the customer car that requests an evaluation of response techniques from the server when it detects an attack. The *server* device receives this request and looks for *workshopCars* that could run the experiments. The *workshopCar* device has identical parameters to the *onRoadCar* and runs the experiments to assess different response techniques. Once all required experiments are finished, the *workshopCar* sends the results to the server for evaluation.

The main component, which handles the response inside the vehicle, is the *Response Component*. It applies a response technique once an attack is detected and communicates through the *Communication Module* to the *Collaboration Management Component* to request the best suited response technique. The *Response Assessment Component* applies different techniques and logs metrics, such as CPU utilization and/or memory

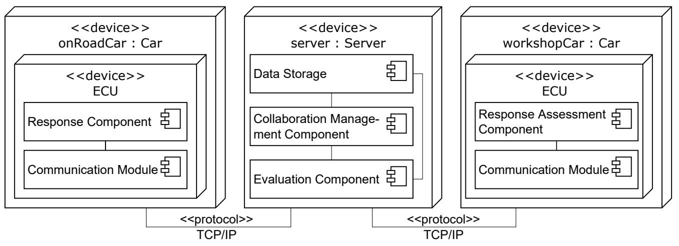

Fig. 2. Deployment diagram of the RIPOSTE framework.

usage. These logs are sent back to the *Collaboration Management Component* and saved in the *Data Storage*. The *Evaluation Component* compares the results to find the most efficient and effective response technique. Finally, the *Collaboration Management Component* manages all coordination and collaboration between these components.

D. RIPOSTE: Behavioral Description

The behavior of RIPOSTE is described by a sequence diagram[2]. The diagram shows an *onRoadCar*, a server, and *n workshopCars*. The *onRoadCar* and *workshopCars* first enroll to the server, pass their software version, hardware specifications, and the collaboration status. If the car is collaborative, it acts as a test environment to conduct experiments. If the *onRoadCar* gets attacked, it sends a request evaluation message to the server informing it about the attack and which response technique it is planning to apply. The server checks its database to look for previously assessed response techniques matching the software version and the hardware specifications of the *onRoadCar*. If it finds that all response techniques were previously assessed and evaluated, it checks whether the *onRoadCar* is using the best suited response technique (i.e., most efficient and effective technique). If the *onRoadCar* is already using the best technique, it sends an acknowledgment message. Otherwise, the server sends the best response technique it previously identified.

In case a response technique has not been assessed yet, the server starts looking for available *workshopCars* to assess these techniques. The server updates the available *workshopCars* with the appropriate response technique and runs an attack simulation to activate it. After the *workshopCar* is done with the experiment, it reverts back to its original state. The operation of running experiments (i.e. finding the *workshopCar*, updating them, and running attack simulation) is done in parallel, to ensure a fast response to the *onRoadCar*.

When all response techniques are assessed and logs from the *workshopCars* are received, an evaluation is initiated. The server uses the logs to identify the best suited response technique. Once found, it updates its database and pushes the update to the *onRoadCar*. The *onRoadCar* then applies the best response technique received from the server.

Another scenario is that the *onRoadCar* first applies its default response technique, such as rebooting, and then requests an evaluation of the technique it has used. This way, the *onRoadCar* can instantly respond to an attack without having to wait for the server's evaluation, and still gets informed about the best response technique for this particular attack. Yet, the first time the *onRoadCar* uses its default response technique may not the best response technique, therefore a trade-off analysis between instant response to an attack and the required efficiency of the response needs to be performed.

V. Implementation and Testing

A testbed is developed to evaluate RIPOSTE. The setup consists of three Raspberry Pi 4 devices running *Raspberry Pi OS* and a computer (i.e., server) with *Raspberry Pi Desktop*. All devices are connected via Ethernet. One Raspberry Pi device acts as *onRoadCar* which sends evaluation requests to the server when it detects an attack. The other two Raspberry Pi devices act as *workshopCar1* and *workshopCar2* and perform the assessments of requested response techniques. To test the feasibility of RIPOSTE, we implemented a *privilege escalation* attack and three response techniques, namely *graceful degradation/limp mode, reparameterization* and *re-instantiation/restart*. The latter represents an edge case where the car has to disconnect, reboot, and reconnect.

The test scenario starts by manually attacking the *onRoadCar* to trigger RIPOSTE, causing the *onRoadCar* to request an evaluation from the server including information about the response technique it plans to use. The server processes the request and instructs the response technique to be used by the *onRoadCar* based on two criteria: (i) it is an effective technique and (ii) it is the most efficient technique.

A response technique can either be *effective* (i.e., it prevents the attack) or *not effective*. This is measured by running the attack simulation on *workshopCars* a second time after applying the response technique. If the attack does not succeed, then the technique is effective. However, if the attack is successful, then the technique is not effective. As for efficiency, time needed for the deployment of each response technique is measured. The response technique with the shortest deployment duration is considered as most efficient.

RIPOSTE first looks for all effective techniques and subsequently communicates the technique with the shortest deployment time to the *onRoadCar*. The implementation and testing details of the RIPOSTE proof of concept are available online[3].

Next, we describe the implemented attack and deployed response techniques in Section V-A, and describe the scenarios used to test RIPOSTE in Section V-B.

A. Attack and Response Techniques

The attack implemented for testing the framework is a simple representation of a privilege escalation attack. The *onRoadCar* monitors the system's shadow file (`/etc/shadow`) which can only be read by root. Any alert about the modification of the attributes of this file will trigger RIPOSTE.

We implemented three response techniques, namely graceful degradation, reparameterization and re-instantiation/restart.

[2]RIPOSTE Sequence Diagram: https://doi.org/10.5281/zenodo.6617399

[3]Implementation and Testing: https://doi.org/10.5281/zenodo.6617376

Graceful Degradation/Limp Mode: It allows a system to prevent a catastrophic failure when being attacked without stopping to function. In this setup the number of services running on the system is set to the minimum required to provide the core functionalities. Specifically, services allowing remote access to the system (*SSH*, *Bluetooth*) are disabled to reduce the attack surface. Furthermore, the mail transfer agent *exim4*, *nfs-common*, and *cron* are disabled to reduce the possibilities of information leakage and distribution of malicious scripts.

Reparameterization: It is used to dynamically adjust the system based on the situation. The results of applying this technique can lead to a non-optimal state, which is similar to *Graceful Degradation/Limp mode*. Therefore, the implementation focused on making the shadow file immutable; preventing any further attribute modifications on this file even by root.

Re-instantiation/Restart: It is used to revert to the initial configuration and remove possible temporary access from the attacker. The system saves the analysis log before it reboots, reconnects back to the server after the reboot is completed, and sends the log file with the analysis data. This technique is picked to demonstrate functionality in edge cases, e.g., when the car loses its connection to the server during an evaluation, it needs to be able to reconnect and submit the analysis log.

B. Test Scenarios

At first the server starts listening on the interface's IP address (192.168.1.253:12321) and waits for connection requests. The three clients (*onRoadCar*, *workshopCar1* and *workshopCar2*) send their technical specification and start monitoring their systems, and the server in turn sends an acknowledgment to the clients. After establishing a successful connection with the server, we run four scenarios to verify the correct behavior of the RIPOSTE proof of concept.

Scenario 1. An attack is manually triggered which causes the *onRoadCar* to send an evaluation request to the server. Next, the server initiates the assessment of the response techniques in collaboration with the workshop cars. After the assessment the server sends the best suited (i.e., most efficient and effective) response technique to the *onRoadCar*. The *onRoadCar* receives the update message and applies the response technique suggested by the server which is reparameterization in the chosen use case.

Scenario 2. The attack is executed again in order to verify that the suggested response technique is applied and working.

Scenario 3. The state of the *onRoadCar* is manually reverted to bring the vehicle back to the desired state. Once the attack is triggered again, the client sends an evaluation request which includes the information that the *onRoadCar* wants to apply the reparameterization technique. The server acknowledges that it is the best suited technique since the server has assessed and evaluated it previously.

Scenario 4. *WorkshopCar1* is used to assess the restart response technique. After the restart, a script to connect back to the server and report the evaluation results is executed.

More details on the these test scenarios are provided in the implementation and testing material online (See footnote [3]).

VI. Evaluation

The *safety* of RIPOSTE is evaluated by domain experts. The responses to the evaluation questionnaire are analyzed using thematic analysis following the steps recommended by Cruzes and Dybå [17]. Figure 3 shows the thematic model created using the analysis. The overall safety of the framework is rated as *slightly unsafe* with a score of 2 out of 5 (scale from 1/extremely unsafe to 5/very safe). In the following subsections, the safety concerns and considerations shown in the thematic model are used to explain the obtained safety rating of the developed framework.

A. Automated experiments without human interaction

One of the considerations is the attack complexity when conducting automated experiments. Being able to replicate exact attacks from a source may require human intervention, including consultations with safety and security experts. That being said, the implementation of response techniques that target complex attacks can be hard to perform, especially in an automated manner. For that, one of the participants in the evaluation suggested the use of emulators within simulators to overcome environment replication challenges and get as close as possible to the conditions that the *onRoadCar* is in.

Configurations of the continuous experiments are another consideration. Participants mentioned that environmental conditions and human interactions should be considered when evaluating response techniques. For instance, the environment for a moving vehicle is quite different to a stationary vehicle, which may lead to varying results from successful to devastating. Moreover, the attributes under focus can play an important role when judging the safety of automated experiments. A participant mentioned that continuous experiments should be kept inside a variability of ranges of the possible configurations to avoid any unconsidered state that might be left unnoticed or not anticipated in the design analysis.

Another comment addresses the implementation of response techniques. Standards and certifications should be followed when developing these techniques. Thereafter, the response techniques should be tested and evaluated in terms of safety by running an entire attack scenario along with the environmental input to achieve optimal results.

B. Qualification of workshop cars

The qualification of *workshop cars* is another safety concern since it is rated *severe* with a score of 5 out of 5 (scale from 1/insignificant to 5/severe). The participants focused on how well the *workshop car* represents the *onRoadCar* that is under attack. This included the representation of inputs, environment conditions, and attack vectors. The participants also pointed out that technical specifications (e.g., hardware and software) and component setup should be identical in order to get optimal evaluation results. Furthermore, strengthening of the security of the framework should be also considered, since

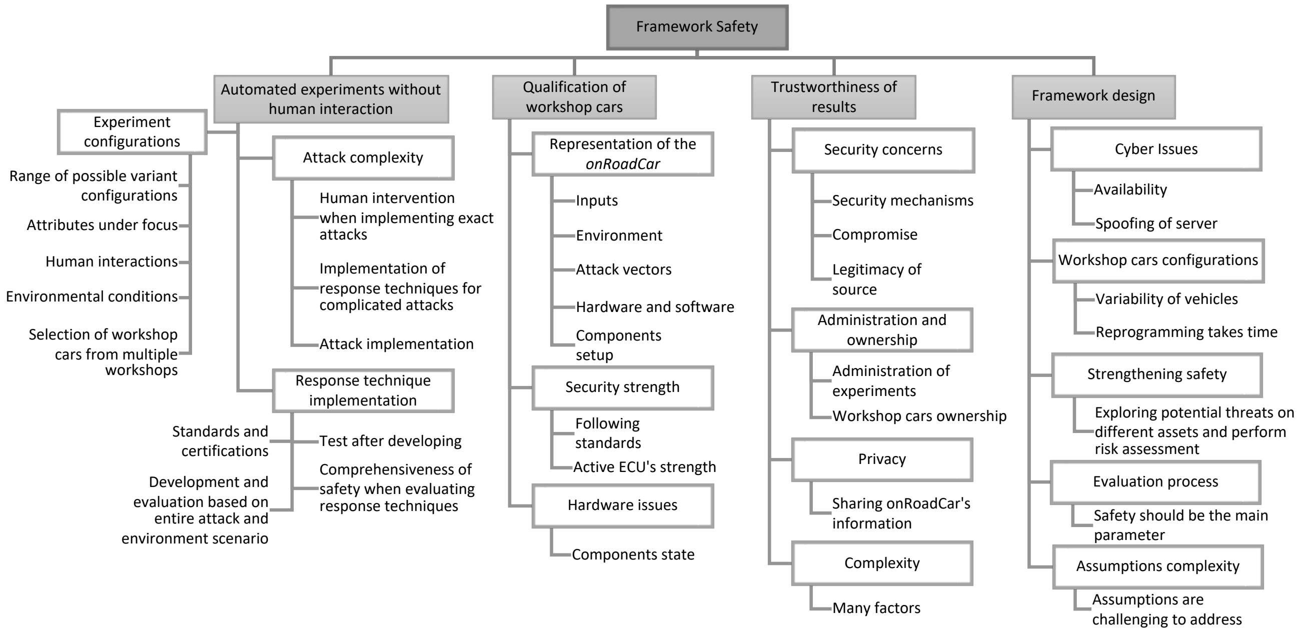
Framework Safety
Automated experiments without human interaction
Experiment configurations
Range of possible variant configurations
Attributes under focus
Human interactions
Environmental conditions
Selection of workshop cars from multiple workshops
Attack complexity
Human intervention when implementing exact attacks
Implementation of response techniques for complicated attacks
Attack implementation
Response technique implementation
Standards and certifications
Development and evaluation based on entire attack and environment scenario
Test after developing
Comprehensiveness of safety when evaluating response techniques
Qualification of workshop cars
Representation of the onRoadCar
Inputs
Environment
Attack vectors
Hardware and software
Components setup
Security strength
Following standards
Active ECU's strength
Hardware issues
Components state
Trustworthiness of results
Security concerns
Security mechanisms
Compromise
Legitimacy of source
Administration and ownership
Administration of experiments
Workshop cars ownership
Privacy
Sharing onRoadCar's information
Complexity
Many factors
Framework design
Cyber Issues
Availability
Spoofing of server
Workshop cars configurations
Variability of vehicles
Reprogramming takes time
Strengthening safety
Exploring potential threats on different assets and perform risk assessment
Evaluation process
Safety should be the main parameter
Assumptions complexity
Assumptions are challenging to address

Fig. 3. Thematic model for the safety of the developed framework.

following the standards and increasing the security of active safety ECUs could potentially reduce unintended errors during the attack simulation process. One participant mentioned that the component's state should be seriously considered. As components age, the *workshop cars* using these components can provide wrong data to the server, making the evaluation process faulty. However, this participant also pointed out that this can be resolved through using majority voting with several cars running the experiments simultaneously.

C. Trustworthiness of results

The trustworthiness of results is addressed by the participants and rated as *severe* with a score of 5 out of 5 (scale from 1/insignificant to 5/severe). The security aspect of the *workshop cars* should be considered by implementing different security mechanisms, such as mutual authentication and secure communication, in order to prevent spoofing and tampering attacks. Additionally, *workshop cars* need to be ensured that they are not compromised, for example, through malware.

One participant raised concerns about the administrative authority performing the experiments, for instance, whether it is a trustful Original Equipment Manufacturer (OEM) or a third party. The participant also pointed out that the ownership of the *workshop cars* should be considered, i.e., whether the *workshop cars* are owned by the OEM or by individuals who agreed to participate in the experimentation.

Privacy concerns when sharing information about the *onRoadCar* with the server and other *workshop cars* are also raised. Users should know e.g., what information related to their car are shared and with who. Lastly, trusting the results is considered as a complex process that could be compromised by many factors, hence more investigations of these factors and their effect on the trusting process are needed.

D. Framework Design

Main topics in the considerations about the framework design are related to cyber issues. One participant pointed out that, besides safety, availability may become a factor. For instance, a compromised infotainment system gets automatically shut down in response to a cyber attack. The unavailability caused by the chosen response technique could cause the driver to become irritated or nervous, which in turn could lead to an accident. This is a specific example which highlights the need to inform the driver when core functionalities of the vehicle are limited due to an attack. Another cyber issue is spoofing the server's identity and suggesting inefficient response techniques. Although the participant points out that this applies to all centralized communications, it is a concern that should be taken into account.

The configuration of *workshop cars* is considered as an important factor in framework design. Indeed, one participant mentioned that the variability between vehicles is a major factor that needs to be factored in since it can be difficult to find two identical cars with the same parameters and configurations. This participant also explained that *workshop cars* are likely going to be reprogrammed, which could delay the process of responding to the *onRoadCar* as well as the consultation of other requests.

One participant highlighted the need for strengthening the safety through looking into threats that could potentially affect different assets of the vehicle and perform a risk assessment. This participant also pointed out that the main goal should be ensuring safety when evaluating different response techniques. Another participant pointed out that the framework relies on a set of assumptions that could be challenging to address, such as attack simulation.

Based on the results of the qualitative evaluation, we summarize the recommendations of the participants for improving the overall safety and security of the RIPOSTE framework:

- The design and implementation of response techniques should follow standards and certifications.
- *Workshop cars* should represent the *onRoadCar* from all aspects e.g., same context, events, and active functions.
- The component's state should be regularly evaluated.
- Security of the framework should be considered.
- The administration of experiments should be performed by a trusted party.
- Privacy of the *onRoadCar* should be considered.
- Cyber issues should be counted for. E.g., availability is improved by adding redundant servers.

Simulating *onRoadCars* to deal with attack complexity and safety concerns in form of digital twins may be an alternative to the use of *workshopCars*. However, digital twins also face challenges, for instance, legacy systems may not be sufficiently documented and thus affect the accuracy of the simulations, and firmware may not be available as source code due to proprietary solutions from third party suppliers [22], [23].

The number of involved participants in the evaluation might be considered as small, thus limiting the generalizability of the results. We do not consider this as a major threat since the aim of the evaluation is to get a preliminary *qualitative* feedback that helps in improving the framework and not to generalize over a population of actors.

VII. Conclusion

Addressing the growing security concerns is necessary due to the broad connectivity of modern vehicles. In this paper, we present RIPOSTE, a framework that supports the collaboration between vehicles to evaluate cyber attack response techniques and select the most efficient and effective response technique against an ongoing attack. To the best of our knowledge, the proposed framework is the first of its kind.

The framework is evaluated in terms of safety by conducting a qualitative study with experts from the automotive security and/or safety domain. The results of the evaluation indicate that there are some safety concerns that need to be addressed such as the qualification of workshop cars, trustworthiness of results, automated experiments without human interaction, and the overall framework design. Moreover, the evaluation shows that the interplay of safety and security should be one of the top priorities when it comes to developing a solution in the automotive domain. The results of the conducted evaluation will be used to improve the framework in the future. Moreover, we plan to further test the framework using more scenarios, and conduct additional evaluations of the security and safety thereof involving more experts.

Acknowledgment. This research was supported by the CyReV project (2019-03071) funded by VINNOVA, the Swedish Governmental Agency for Innovation Systems.

References

[1] H. Martin, Z. Ma, C. Schmittner, B. Winkler, M. Krammer, D. Schneider, T. Amorim, G. Macher, and C. Kreiner, "Combined automotive safety and security pattern engineering approach," *Reliability Engineering & System Safety*, vol. 198, p. 106773, 6 2020.

[2] "Cyber Security and Resilience of smart cars," The European Union Agency for Network and Information Security (ENISA), Tech. Rep., 2016. [Online]. Available: https://publications.europa.eu/en/publication-detail/-/publication/13d4bf8d-e9de-11e6-ad7c-01aa75ed71a1

[3] International Organization for Standardization (ISO), "ISO/SAE 21434 Road Vehicles – Cybersecurity Engineering," Standard, 2021.

[4] M. D. Stojanović and S. V. Boštjančič Rakas, Eds., *Cyber Security of Industrial Control Systems in the Future Internet Environment*, ser. Advances in Information Security, Privacy, and Ethics. IGI Global, 2020.

[5] J. Ochieng, "Cyber Threat Analysis - Cyber Experts," available: https://cyberexperts.com/cyber-threat-analysis-a-complete-overview/ (Accessed: 2020-11-09).

[6] T. Rosenstatter, K. Strandberg, R. Jolak, R. Scandariato, and T. Olovsson, "REMIND: A Framework for the Resilient Design of Automotive Systems," in *2020 IEEE Secure Development (SecDev)*. IEEE, 9 2020, pp. 81–95.

[7] M. Howard and S. Lipner, *The security development lifecycle*. Microsoft Press Redmond, 2006, vol. 8.

[8] D. Ratasich, F. Khalid, F. Geissler, R. Grosu, M. Shafique, and E. Bartocci, "A roadmap toward the resilient internet of things for cyber-physical systems," *IEEE Access*, vol. 7, pp. 13 260–13 283, 2019.

[9] E. Mousavinejad, F. Yang, Q.-L. Han, X. Ge, and L. Vlacic, "Distributed cyber attacks detection and recovery mechanism for vehicle platooning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 9, pp. 3821–3834, 9 2020.

[10] T. Nandy, R. M. Noor, M. Yamani Idna Bin Idris, and S. Bhattacharyya, "T-BCIDS: Trust-based collaborative intrusion detection system for VANET," in *2020 National Conference on Emerging Trends on Sustainable Technology and Engineering Applications (NCETSTEA)*. IEEE, 2 2020, pp. 1–5.

[11] G. Raja, S. Anbalagan, G. Vijayaraghavan, S. Theerthagiri, S. V. Suryanarayan, and X.-W. Wu, "SP-CIDS: Secure and private collaborative IDS for VANETs," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–9, 2020.

[12] F. Giaimo and C. Berger, "Continuous experimentation for automotive software on the example of a heavy commercial vehicle in daily operation," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12292 LNCS, pp. 73–88, 3 2020.

[13] R. J. Wieringa, *Design science methodology for information systems and software engineering*. Springer, 2014.

[14] P. Runeson, E. Engström, and M.-A. Storey, "The design science paradigm as a frame for empirical software engineering," in *Contemporary Empirical Methods in Software Engineering*. Springer International Publishing, 2020, pp. 127–147.

[15] J. R. Venable, J. Pries-heje, and R. Baskerville, "Design science research in information systems. advances in theory and practice," vol. 7286, 2012.

[16] B. Gillham, *Developing a Questionnaire*. London, UK: Bloomsbury Publishing Plc, 2008.

[17] D. S. Cruzes and T. Dyba, "Recommended steps for thematic synthesis in software engineering," in *2011 International Symposium on Empirical Software Engineering and Measurement*, no. 7491. IEEE, 9 2011, pp. 275–284.

[18] I. Etikan, "Comparison of convenience sampling and purposive sampling," *American Journal of Theoretical and Applied Statistics*, vol. 5, no. 1, p. 1, 2016.

[19] G. Schermann, J. Cito, and P. Leitner, "Continuous experimentation: Challenges, implementation techniques, and current research," *IEEE Software*, vol. 35, no. 2, pp. 26–31, 3 2018.

[20] P. Hodgson, "Feature toggles (aka feature flags)," 2017, available: https://martinfowler.com/articles/feature-toggles.html (Accessed: 2021-03-17).

[21] M. T. Rahman, L. P. Querel, P. C. Rigby, and B. Adams, "Feature toggles: Practitioner practices and a case study," *Proceedings - 13th Working Conference on Mining Software Repositories, MSR 2016*, pp. 201–211, 2016.

[22] M. Eckhart and A. Ekelhart, *Digital Twins for Cyber-Physical Systems Security: State of the Art and Outlook*. Cham: Springer International Publishing, 2019, pp. 383–412.

[23] D. Holmes, M. Papathanasaki, L. Maglaras, M. A. Ferrag, S. Nepal, and H. Janicke, “Digital twins and cyber security – solution or challenge?” in *2021 6th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)*, 2021, pp. 1–8.