ComFASE: A Tool for Evaluating the Effects of V2V Communication Faults and Attacks on Automated Vehicles

Mateen Malik*, Mehdi Maleki*, Peter Folkesson*, Behrooz Sangchoolie*, Johan Karlsson†
*Dependable Transport Systems, RISE Research Institutes of Sweden, Borås, Sweden
{mateen.malik, mehdi.maleki, peter.folkesson, behrooz.sangchoolie}@ri.se
† Department of Computer Science and Engineering, Chalmers University of Technology, Göteborg, Sweden
johan@chalmers.se

***Abstract*—This paper presents ComFASE, a communication fault and attack simulation engine. ComFASE is used to identify and evaluate potentially dangerous behaviours of interconnected automated vehicles in the presence of faults and attacks in wireless vehicular networks. ComFASE is built on top of OMNET++ (a network simulator) and integrates SUMO (a traffic simulator) and Veins (a vehicular network simulator). The tool is flexible in modelling different types of faults and attacks and can be effectively used to study the interplay between safety and cybersecurity attributes by injecting cybersecurity attacks and evaluating their safety implications. To demonstrate the tool, we present results from a series of simulation experiments, where we injected delay and denial-of-service attacks on wireless messages exchanged between vehicles in a platooning application. The results show how different variants of attacks influence the platooning system in terms of collision incidents.**

***Index Terms*—attack injection, fault injection, simulation-based system, V2V communication, platooning, cybersecurity attack**

I. Introduction

Connectivity of safety-critical cyber physical systems [1] has increased drastically leading to the delivery of functions and features with greater efficiency. One such example of these systems is where vehicles communicate with each other to alert drivers of upcoming hazards such as slippery road or obstruction [2]. These systems should be built complying with a comprehensive set of safety and security requirements. To this end, they are equipped with fault and intrusion handling mechanisms to protect the system from hardware and software faults and cybersecurity attacks.

Testing and verification of a system's capability of handling faults and attacks is a complex task. It requires a comprehensive set of expertise and appropriate tools to discover the weaknesses and vulnerabilities in the system. *Fault and attack injection* is one of the effective testing techniques that is used to evaluate the system in the presence of faults and attacks.

Faults and attacks can be injected into a system at different stages of the product development life cycle, e.g., at early design stages or close to a production. Injections performed at initial stages [3]–[6] facilitate early discovery of system weaknesses. This could result in saving time and cost since late discovery of a weakness could trigger a costly redesign.

Fault and attack injection testing could be performed in *real-world* or in *simulation-based* test environments. Simulation-based testing is comparatively low-cost, reproducible and offers high test coverage. Legal bodies such as UNECE (United Nations Economic Commission for Europe) [7] and automotive OEMs (Original Equipment Manufacturers) are moving towards increasing the use of *simulation-based* testing and verification of automated driving system (ADS) [8] together with the *real-world* testing for validation and certifications.

In this paper, we present ComFASE, an open source fault and attack injection tool for evaluating the impact of vehicle-to-vehicle (V2V) communication faults and attacks on automated vehicles in a simulation environment. ComFASE incorporates two open source simulators: SUMO (a traffic simulator) [9] and OMNeT++ (a network simulator) [10]. In addition, it requires a vehicular network simulator based on OMNeT++, such as Veins [11]. ComFASE is capable of evaluating the impact of communication faults and attacks on the target vehicle as well as the surrounding traffic. This is important as previous studies show that a faulty vehicle could significantly influence the behaviour of surrounding vehicles [12], [13].

To demonstrate the capabilities of ComFASE, we present the results of experiments where we have injected two types of cybersecurity attacks, namely *delay* and *denial-of-service (DoS)*, into the wireless channel between vehicles, modelled in the Veins simulator. The target system in these experiments is a platooning system implemented in the Plexe-veins [14].

We classify the results of the injections based on two parameters: *deceleration* profiles of all vehicles in the platoon, and *collision* incidents. These parameters allow us to measure implications of the attacks on the system safety through identification of cases where emergency braking or collision incidents are reported by the target vehicle or the surrounding traffic. After classification of the results, we analyse the experiment outcomes with respect to attack injection parameters, such as the attack initiation time and duration as well as its value.

II. Background

A. Simulation-based Fault and Attack Injection

Simulation-based fault and attack injection is a test method that can be used to evaluate safety and cybersecurity attributes of automated vehicles and is useful for early system design evaluation when only a model or application prototype is available. Using this test method, it is possible to perform extreme test cases which are otherwise difficult to setup in the real world, costly to perform and can have severe consequences in terms of property damages or human injuries. SUFI [12], [13], SAE++ [15] and NETA [16] are some examples of the simulation tools that implement this test method.

B. IEEE Standards for WAVE Communication

There are many wireless vehicular communication standards available such as IEEE 802.11 [17], IEEE 802.11p [18], IEEE 802.11p/ITS-5G [19], 4G [20] and 5G [21]. These standards provide different communication ranges and have applications within different problem domains.

ComFASE injects faults and attacks in the wireless channel model of the V2V communication, which is based on realistic models of the IEEE 802.11p and the IEEE 1609.4 WAVE communication protocols implemented in the Veins (see §II-C). The wireless channel model represents an analog medium where the message is transmitted after it is encoded and modulated in the physical layer. The need for these standards arises from the fact that vehicles move with high velocity and therefore are subject to short and unstable connections [11] [22].

IEEE 802.11p is an amendment of the IEEE 802.11 standard to support WAVE for V2V communications. While IEEE 802.11p WAVE only covers the physical and lower layer of MAC (media access control), IEEE 1609.4 covers upper MAC layer of the vehicle communication system (see Fig. 1). IEEE 1609.4 [23] represents various components of the WAVE such as multi-channel communication for radio operations, quality of services, channel switching and routing.

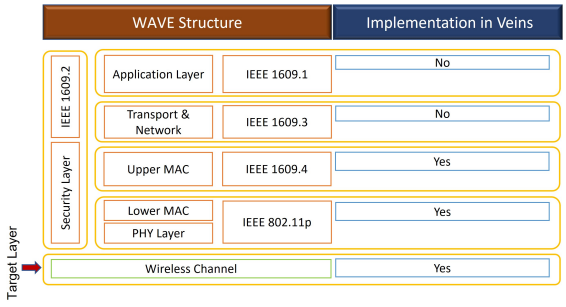

Fig. 1: WAVE communication models and wireless channel models implemented in Veins.

C. Simulation Environment

In this paper, ComFASE is configured to run experiments with four simulators: OMNet++ *v. 5.6.2* (a network simulator) [10], SUMO *v. 1.8.0* (a traffic simulator) [9], Veins *v. 5.1* (a vehicular network simulator) [11] and Plexe-veins *v. 2.1* (a platooning extension for Veins) [14]. Veins is built upon OMNeT++ and implements simulation models of the communication standards discussed in §II-B to allow testing and analysis of vehicular networks and systems. Note, however, that ComFASE can be configured to run with other vehicular network simulators that are build upon OMNeT++. The model of the platooning system (used as the system under test in this paper), including vehicle dynamics and a cruise control model, is implemented in Plexe-veins, which is based on Veins and SUMO. All these simulators are open source and are combined to provide a complete simulation environment (see Fig. 2). Moreover, ComFASE logs detailed traffic simulation data from SUMO and vehicular network simulation data from Veins. Examples of data logged are vehicle speed, acceleration/deceleration and position, as well as various other data, which are used to describe collision incidents.

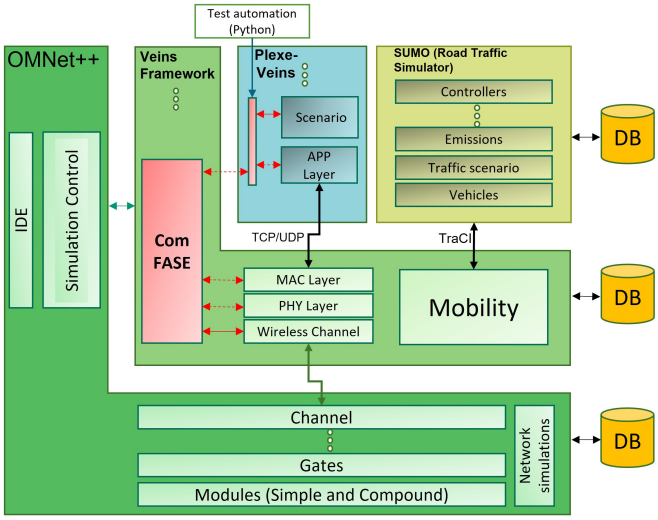

Fig. 2: ComFASE simulation environment.

D. Related Work

Previous studies that investigated the impact of network attacks on platooning applications include Heijden et al. [24], Boeira et al. [25], and Iorio et al. [26].

Heijden et al. [24] studied the impact on vehicle behavior by injecting jamming attacks in the application layer of the vehicular communication system. They investigated the resilience of different vehicle controllers to these attacks. Boeira et al. [25], however, focused on sybil attacks (falsification of multiple identities), message falsification and radio jamming attacks in the application and network layers. The behaviour of the targeted vehicle was observed in SUMO.

Iorio et al. [26] injected falsification of parameters (such as position, speed and acceleration) attacks to study a cooperative adaptive cruise control (CACC) algorithm in a platoon scenario. The target for the attacks was the platoon leader and predecessor vehicle. They analysed the attack injection results by looking into the distance between the platoon vehicles.

Similar to the work presented by Heijden et al. [24], Iorio et al. [26] compared the resiliency of different controllers.

Similar to ComFASE, the authors of [24], [25] and [26] have developed tools to inject attacks in a platooning system modelled in Plexe-veins [14]. However, unlike these tools, ComFASE is capable of injecting attacks in wireless channel models used for WAVE vehicle communication.

Lastinec et al. [27] uses an attack injection tool that is similar to Heijden et al. in terms of message exploitation. The attacker vehicle exploits an emergency vehicle's (e.g., an ambulance) information to gain privileges for driving faster.

SAE++ [15] and NETA (NETwork Attacks) [16] are OMNeT++/INET based attack simulation tools that have been developed in the past. Unlike ComFASE, these tools are not connected to a traffic simulator, and can therefore not assess the impact of successful attacks on a traffic environment.

III. ComFASE: A Communication Fault and Attack Simulation Engine

ComFASE[1] is capable of injecting faults and attacks in the vehicular communication system modelled in the Veins simulator (see Fig. 1). In this paper, we demonstrate the capabilities of ComFASE to inject two types of cybersecurity attacks on a wireless channel. We denote these attacks as *delay* and *denial-of-service* attacks. The tool can be extended with other types of faults and attacks to be injected in other layers of the vehicular communication system such as the physical and application layers. In order to evaluate various cybersecurity attributes of a system, ComFASE allows to integrate different traffic scenarios such as platooning and teleopearation. In the simulated traffic scenario, ComFASE targets the vehicles transmission and reception capabilities by modifying the models built for vehicle communication in Veins. Moreover, the tool provides an opportunity to analyse the impact of faults and attacks on the target vehicle and the surrounding traffic.

A. ComFASE Execution Flow

The execution of ComFASE is presented in four steps as a pseudo code in Algo. 1. In this section, we present these steps and explain how a test campaign is configured (Step-1: lines 1-4) and executed (Step-2 and Step-3: lines 5-15) and how the campaign results are classified (Step-4: lines 16-18).

Step-1 Test-Configuration: The workflow begins with configuration of the traffic scenario (*TrafficScenario*), communication model (*CommModel*), and attack injection campaign (*AttackCampaignSetup*).

The parameters required to set-up the traffic scenario (line 2 in Algo. 1) are:

- *roadFeatures*, defining the road properties e.g., number of lanes, length, width and speed limit.
- *vehicleFeatures*, defining the software and hardware properties of the vehicle e.g., length of the vehicle, maximum speed, acceleration/deceleration ability and controller such as CACC (cooperative adaptive cruise control).
- *nrVehicles*, setting the number of the vehicles in traffic.
- *scenarioManeuver*, defining the vehicles driving pattern, such as acceleration and braking.
- *totalSimTime*, setting the total simulation time.

The traffic scenario can either be selected from already existing scenarios or can be created and configured based on the user needs.

In order to configure the communication model (line 3 in Algo. 1), the parameters that need to be set are:

- *commProtocol*, specifying the communication protocol which are valid for vehicle communication such as IEEE 802.11p, and IEEE 1609.4.
- *wirelessModel*, specifying the model to use for the environmental effects in the wireless communication such as 'free path loss model' and 'two-ray interference model'.
- *packetSize*, defining the size of the message that is to be transferred through the wireless channel.
- *beaconingTime*, defining the beaconing frequency which every vehicle uses to broadcast the messages.

The inputs required for setting the attack campaign (line 4 in Algo. 1) are:

- *attackModel* selection of a predefined attack model, e.g, *delay* or *denial-of-service (DoS)* attack.
- *targetVehicles*, specifying the vehicles under attack.
- *attackValuesVector*, containing a set of attack model parameter values to be injected during a campaign.
- *attackStartVector*, containing a set of attack initiation times when the attack starts in an experiment.
- *attackEndVector*, containing a set of attack end times when the attack ends in an experiment.

Step-2 Golden Run: In this step, we execute a golden run (line 6 in Algo. 1) where the system is not exposed to any faults or attacks. The inputs required to perform the golden run are *TrafficScenario* and *CommModel*, which we configured in Step-1. The golden run is simulated until the simulation time ends (i.e., *totalSimTime*). The output is recorded as *GoldenRunLog*.

Step-3 Attack Injection Campaign: This step presents the attack injection procedure (lines 7-15 in Algo. 1), where inputs are *TrafficScenario*, *CommModel* and *attackCampaignSetup*. The set of attacks to be injected are determined by the *attackStartVector*, *attackValueVector* and the *attackEndVector*. To this end, ComFASE runs in three nested loops, where the simulation is run until *attackStartTime* with *SimUntil* using the communication model initialised in Step-1 (i.e., *CommModel*). Then, the simulation is run until *attackEndTime* using the updated communication model (i.e., *UpdatedCommModel*), and finally the simulation is run until simulation ends (i.e., equal to *totalSimTime*) with the communication model initialised in Step-1.

Algo. 1 also shows that the data observed from the simulation for each experiment is stored in *AttackCampaignLog* and that the *expNr* keeps track of the total number of experiments.

[1] *https://github.com/RISE-Dependable-Transport-Systems/ComFASE*

Algorithm 1 Pseudo code of ComFASE Execution Flow

```
1: //Step-1: Test-Configuration
2: TrafficScenario= setScenario(roadFeatures, vehicleFeatures, nrVehicles, scenarioManeuver, totalSimTime)
3: CommModel= setCommunication(commProtocol, wirelessModel, packetSize, beaconingTime)
4: AttackCampaignSetup = setCampaign(attackModel, targetVehicles, attackStartVector, attackValuesVector, attackEndVector)
5: //Step-2: Golden-Run (attack-free simulation run)
6: GoldenRunLog= SimUntil(TrafficScenario, CommModel, totalSimTime)
7: //Step-3: Attack-Injection-Campaign (campaign is a set of attack experiments)
8: for each attackStartTime in attackStartVector do
9:     for each attackValue in attackValuesVector do
10:        for each attackEndTime in attackEndVector do
11:            UpdatedCommModel= CommModelEditor(CommModel, attackValues, targetVehicles)
12:            AttackCampaignLog[expNr] += SimUntil(TrafficScenario, CommModel, attackStartTime)
13:            AttackCampaignLog[expNr] += SimUntil(TrafficScenario, UpdatedCommModel, attackEndTime)
14:            AttackCampaignLog[expNr] += SimUntil(TrafficScenario, CommModel, totalSimTime)
15:            expNr ++
16: //Step-4: Classification (classify each experiment)
17: for each exp in Attack-Injection-Campaign do
18:     Classification = Compare(GoldenRunLog, AttackCampaignLog[exp], classificationParameters)
```

Step-4 Classification: The inputs required for classification of the results (lines 16-18 in Algo. 1) are *GoldenRunLog*, *AttackCampaignLog*, and *classificationParameters* (e.g., *deceleration profiles* and *collision incidents*). The results obtained for each of the attack injection experiments are compared with those obtained for the golden run and classified into one of the result classification categories discussed in §IV-B. For instance an experiment is classified as severe if a collision incident is reported.

After the result classification, we conduct separate analyses to investigate correlations between attack parameters (i.e., *attackStartTimes* and *attackValues*) and the severity of the attacks. The severity is then graded based on the magnitude of vehicle decelerations and collision incidents.

B. Attack Model implementation

ComFASE can target different parameters of the communication simulation environment to model different faults and attacks. Examples of such simulation parameters are vehicle information sent through the wireless channel (such as position, acceleration, speed) and the channel properties (such as propagation delay, signal power, interference, and noise). Propagation delay (PD) is a Veins simulation parameter that is used to introduce the natural communication delay between the vehicle communication. The propagation delay in Veins is calculated based on the distance between vehicle and speed of light. We use the 'propagation delay' parameter to model *delay* and *denial-of-service* attacks by modifying the Veins PD value with the attack value when the attack is active. Table I describes the parameters for modelling the attacks in Veins and their connection to real world attack types. Note that, the values in Table I are acceptable value ranges that can be set in the simulation. The specific values that we have used for the demonstration of the tool are detailed in §IV.

C. ComFASE Limitations

ComFASE can only inject faults and attacks on the vehicular communication system and is limited to the simulation environment that are built upon the OMNeT++ simulator. It is also worth noting that the usefulness of the results obtained using any simulation-based testing tool is tied to the accuracy and representativeness of the simulation environment including the communication and vehicle models when it comes to the evaluation of features of automated vehicles. For example, no security mechanisms are implemented inside the Veins communication model, which is why no such mechanisms are evaluated in this paper.

IV. ComFASE Experiments

To demonstrate the ComFASE capabilities, we injected cybersecurity attacks on the wireless communication channel used in a platooning system using the execution flow presented in §III-A. Here, we present the experimental setup and results of the injections.

A. Experimental Setup

1) Traffic Scenario: In this study we have used an existing platooning scenario (see Fig. 3) implemented in Plexe-veins [14] to demonstrate the use of ComFASE. In this platooning scenario, the road consists of 4 lanes with maximum speed limit of $90m/s$, length of $9400m$, and width of $3.2m$ per lane. There are 4 identical vehicles in the scenario with these features: $4m$ length, $9m/s^2$ deceleration ability, $2.5m/s^2$ acceleration ability, $50m/s$ maximum speed, and CACC (cooperative adaptive cruise control) [30] as a controller. The vehicles accelerates and decelerates in a sinusoidal fashion (see Fig. 4) in the platoon.

We limit the total simulation time to $60s$. Note that, the above-mentioned features come with the default scenario setup

TABLE I: Attack types and simulation parameters for modelling the attacks.

Real world attack types and examples		Simulation parameters for modelling the attacks				
Attack type	**Examples**	**Target parameter**	**Default value (unit)**	**Acceptable valueRange**	**Acceptable attackStartTimes**	**Acceptable attackEndTimes**
Delay	Catching the messages between vehicles, which are blocked from reaching the receiver (e.g., using reactive jamming technique [28]), and re-transmit them at a later time.	Propagation delay (PD)	0.0 (s)	0 to totalSimTime	0 to totalSimTime	attackStartTime to totalSimTime
DoS	Disabling the ability of a vehicle to communicate with other vehicles in a traffic by jamming the communication [29].					totalSimTime

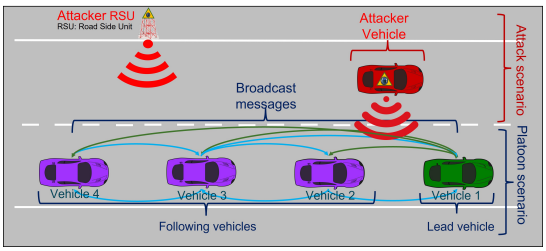

Fig. 3: An example of attacks in a platooning scenario.

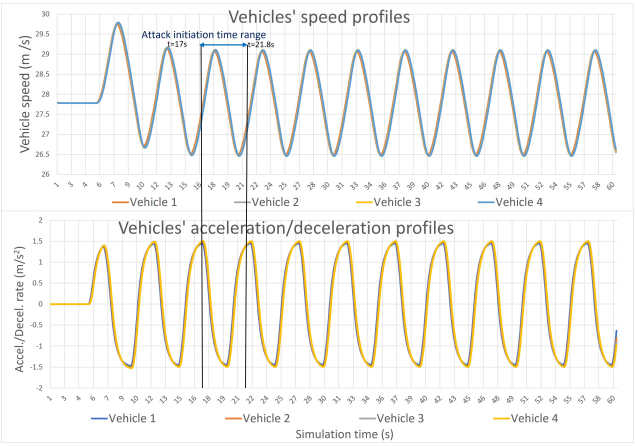

Fig. 4: Speed and acceleration/deceleration profiles of the four vehicles in platoon shown in Fig. 3.

in Plexe-veins and could be easily changed according to the user needs. Also, we use sinusoidal maneuvers to increase the visibility of the behavioral changes of the vehicle under attack. Furthermore, we use the CACC as it uses inter-vehicle communication data to calculate the acceleration and speed of the vehicles in the platoon allowing us to evaluate the effects of V2V communication attacks on the target vehicles.

2) Communication Model: We used DSRC/WAVE communication protocols (see Fig. 1) as *commProtocol* and chose 'free space path loss model' as *wirelessModel* for our experimentation. This model represents the data propagation in a wireless channel between the vehicles. It models a situation where the distance between the vehicles are minimized and is free of obstacles such as in a platooning scenario. To set up the model, we have chosen 200 bits as the *packetSize* and $0.1s$ as the *beaconingTime*.

TABLE II: Parameters values used in experiments.

Attack type	Selected PD valueRange	Selected attackStartTimes	Selected attackEndTimes
Delay	$0.0s$ to $3.0s$ with $0.2s$ step	$17.0s$ to $21.8s$ with $0.2s$ step	attackStartTime $+(1s$ to $30s)$ with $1s$ step
DoS	$60.0s$		$60.0s$

3) Attack Campaign Setup: In this paper, we consider two types of attacks on the wireless channel. One of them represents blocking the reception and transmission of messages of a specific vehicle for a limited time. This type of attack primarily causes a delay in the exchange of messages among the interacting vehicles, and hence we denote such attacks as *delay* attacks. In the other attack model, the reception and transmission of messages of a specific vehicle is blocked from the attack initiation time until the simulation ends. We denote this type of attacks as *denial-of-service* attacks.

Our models assume that the attacker has access to advanced jamming techniques, such as reactive jamming described in [28], to enable blocking of messages. In the simulations, the attacks are injected in the sender & receiver modules of the inter-vehicle communication model in Veins. More specifically, we use the *propagation delay* parameter in this model to implement the attacks.

We injected attacks only on messages sent and received by Vehicle 2, which is the one directly behind the lead vehicle in Fig. 3. (In general, to support various levels of attack complexity, ComFASE can target any number of vehicles in a scenario for experimentation and analysis.)

Table II presents the parameters used in the simulations. We select *attackStartTimes* from time $17s$ to $21.8s$ which is one complete platooning cycle as illustrated in Fig. 4. Within this cycle, we chose start times that are $0.2s$ apart resulting in a total of 25 start time points. Attack duration time specifies the total time during which an attack is active i.e, from the *attackStartTime* to the *attackEndTime*. The total simulation time of each experiment for *delay* and *DoS* attacks was selected as $60s$. For the *delay attacks*, we chose the attack duration from $1.0s$ to $30s$ with a step of $1.0s$ resulting in a total of 30 *attackEndTimes*. The maximum attack duration of $30s$ was selected to make sure that we observe the impact of all attacks before the end of the simulation run. For the *DoS* attacks however, the attack duration is from the attack start until the end of the simulation.

For the delay attacks, we chose to delay the data between $0.2s$ to $3.0s$ with propagation delay (PD) values that are $0.2s$ apart resulting in a total of 15 PD values. Note that, we have also conducted experiments with values higher than $3.0s$ and observed no significant differences in the results obtained compared to those reported for when $3.0s$ was chosen.

We set the propagation delay to $60s$ for the DoS attacks, which implies that all the messages are blocked until the end of the simulation.

B. Result Classification

We use *deceleration profiles* and *collision events* of the vehicles to classify the results in five categories detailed in this section. The deceleration profiles are chosen based on previous studies of rear-end accidents [31].

- **Non-effective:** The injected attack has no effects on the behaviour of the vehicles (i.e., identical speed profiles as in the golden run) and the simulation ends with no indication of failures.
- **Negligible:** The injected attack has modified the behaviour of at least one of the vehicles. The change of behaviour is negligible as the recorded maximum deceleration is less than or equal to $1.53m/s^2$, which is the maximum deceleration recorded in the golden run.
- **Benign:** The injected attack has modified the behaviour of at least one of the vehicles, leading to a deceleration value greater than $1.53m/s^2$ and less than or equal to $5m/s^2$. The safety implications of the change is considered to be benign as it does not lead to a deceleration value greater than the maximum comfortable braking rate, which is $5m/s^2$.
- **Severe:** We classify an experiment as severe if (i) a collision occurs between the vehicles or (ii) when a vehicle performs an emergency braking. The latter case refers to when the injected attack has modified the behaviour of at least one of the vehicles, leading to a deceleration value greater than $5m/s^2$ and less than or equal to $8m/s^2$.

C. Experimental Results

1) Analyses of the Delay Attack Results: We performed 11250 experiments, modelling the delay attacks (25 *attackStartTimes**15 *attackValues**30 *attackEndTimes*=11250). This resulted in 5923, 4941, and 386 experiments classified as severe, benign, and negligible, respectively. The total time it took to simulate the 11250 experiments and parse the logged data was about 7 hours on a computer setup with AMD Ryzen 7 5800X 8-Core processor and 96GiB system memory.

Fig. 5 shows the classification of results based on the duration to which the system under test was exposed to attacks. The figure shows that attacks with duration times greater than $5s$ are always classified as either benign or severe. Moreover, the figure shows that exposing the target system to attacks for a period longer than $4.0s$ affects the total number of severe cases insignificantly. Note that, none of the experiments are classified as non-effective. Moreover, all the severe cases caused are as a result of vehicle collisions.

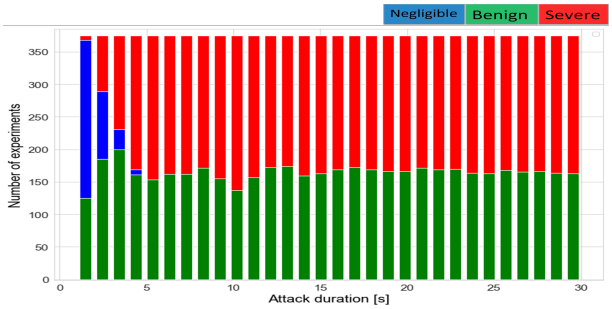

Fig. 5: Classification of results w.r.t. the *duration* in which the propagation delay attack is active.

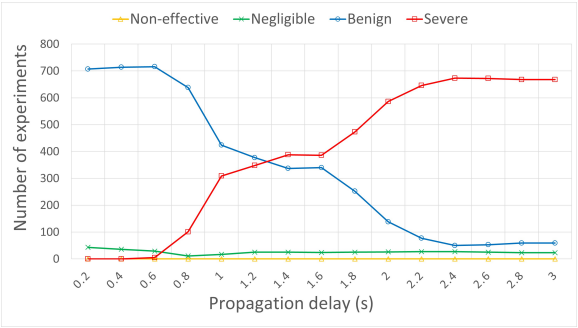

Fig. 6: Classification of results w.r.t. the *propagation delay* values i.e., the length to which data is delayed.

We also analysed the injection results with respect to the different propagation delay (PD) values used in the experiments (see Fig. 6).

For each PD value, a total of 750 experiments were conducted (25 *attackStartTimes**30 *attackEndTimes*=750). The figure shows that, experiments with higher PD values result in a higher number of severe cases. However, no significant differences are observed between the number of severe cases caused for PD values larger than $2.2s$. In fact, we also conducted a set of experiments with PD values up to $5.0s$ and observed the same behaviour. The results of the experiments are also analysed with respect to the *attackStartTimes*. For each *attackStartTimes*, a total of 450 experiments were conducted (15 *attackValues**30 *attackEndTimes*=450). The results are presented in Fig. 7. The figure shows that the majority of experiments are classified as severe when the initiation time is lower than $17.2s$ or higher than $20.6s$. The figure also shows that, the number of severe cases changes drastically in between attack start times of $19.4s$ and $20.2s$.

In order to understand the results presented in Fig. 7, we investigate the behaviour of vehicles driving in a sinusoidal pattern illustrated in Fig. 4. By comparing these figures, we observe that initiating attacks in periods where the lead vehicle has a high acceleration rate could result in higher number of severe cases. This is due to the fact that when the lead vehicle decelerates, the deceleration data is delivered with a delay to the other vehicles in the platoon causing vehicle collisions. The other vehicles in the platoon still believe that the lead vehicle

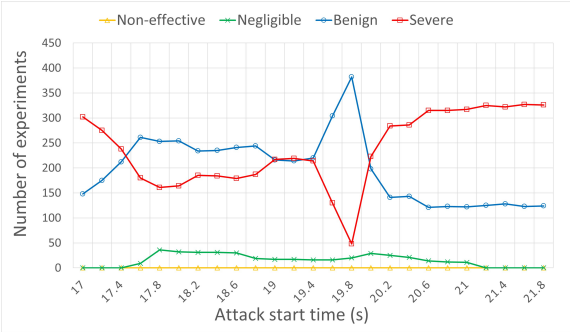

Fig. 7: Classification of results w.r.t. the *attack start time* i.e., the point in time where the first attack is injected.

has a high acceleration rate, while the lead vehicle has already started to decelerate. The same line of justification could be used to explain the low number of severe cases caused for experiments whose start times are between $19.4s$ and $20.2s$. The acceleration rate of the vehicles in this period is as low as $0m/s^2$ causing a high number of benign cases.

Finally, we analysed the impact of the attacks on the surrounding traffic. To this end, we focused on the severe cases, where the injected attacks cause collisions, and studied the vehicle that is responsible for the collision, also known as collider [32]. The result analysis shows that out of 5923 severe cases, the target vehicle (Vehicle 2) is responsible for 65.4% of the incidents, while Vehicle 3 and Vehicle 4 are responsible for another 18.1% and 16.5% of the incidents, respectively. This confirms the previous studies that it is important to analyse impact of the attack injections on surrounding vehicle when only one vehicle is targeted.

2) Analyses of the DoS Attack Results: We perform 25 DoS attack experiments, starting the attacks from $17.0s$ to $21.8s$ where the start time of each experiment is $0.2s$ apart. The results of the experiments show that all the experiments are classified as severe leading to vehicle collisions. The results are also analysed with respect to the vehicle that caused the collision (collider vehicle). The vehicle under attack (Vehicle 2) caused 48% of the collision incidents, however, Vehicle 3 and 4 caused 40% and 12% of the incidents, respectively. After investigating the attack start times for the collision experiments, we learned that if the attack gets activated in times between $17.6s$ - $19.4s$ or $19.6s$ - $20s$, it is Vehicle 3 and Vehicle 4 that are responsible for the collision, respectively. Moreover, for the remaining *attackStartTimes*, it is Vehicle 2 that is responsible for the collision.

3) Discussion: From a tester's point of view, the results presented in this section could be used when designing future experiments. For example, the results obtained for experiments with either shorter duration or propagation delay (PD) values could be used to estimate the number of severe cases for experiments with larger duration and PD values, respectively. As with higher PD values all experiment results saturates to only collision incidents.

When it comes to the start time of an injection, the results of the delay experiments show that the likelihood of an attack to cause a collision is higher when the acceleration rate of the vehicles in the platoon is high. Note that, we only initiated the attacks on one driving cycle of a platoon ($17.0s$-$22.0s$) since the driving patterns (e.g., acceleration and speed) of the vehicles in all successor cycles were the same (see Fig. 4). This way, the results obtained could be used to estimate the severity of attacks within the succeeding cycles. This also shows the importance of scenario-based testing as the evaluation results are tied to the scenario under investigation.

When it comes to the impact of an attack on the entire traffic, the results revealed that injections of attacks into the Vehicle 2 could cause collisions of the Vehicle 3 and 4, thus, an attacker could significantly influence the behavior of the surrounding traffic by targeting only one vehicle.

The high sensitivity of the target platooning system to the two attack types modeled using ComFASE is due to the fact that our target communication model is not equipped with any safety/security mechanisms. However, ComFASE is capable of evaluating these systems regardless of whether or not they are equipped with any mechanisms. The high sensitivity of the target system also shows the importance of having redundant component in these systems. This could e.g., be provided through introduction of sensor models in our simulation environment that monitors the distance between vehicles, which in fact is one of our future research directions.

V. Conclusion and Future Work

In this paper, we presented ComFASE, a tool that facilitates injection of faults and attacks in the wireless channel of vehicular communication systems. ComFASE is built upon OMNeT++, SUMO and Veins simulators, which allows us to study the safety implications of the injected faults and attacks on the target vehicles as well as the surrounding traffic. In ComFASE, fault and attack models are implemented in separate scripts, facilitating addition of new models. Moreover, the tool runs batches of experiments automatically.

To evaluate ComFASE, we have injected *delay* and *denial-of-service* attacks in a platooning scenario and showed the usefulness of the tool. To this end, we conducted *11275* experiments where *5948* of them resulted in collision incidents. The results obtained also revealed the importance of attack's initiation time and duration as well as the vehicles' driving pattern in causing collision incidents.

As part of our future work, we plan to extend ComFASE by modelling faults and attacks in the physical and application layers as well as conducting jamming attacks in the wireless channel of vehicular communication systems. Moreover, we plan to integrate ComFASE with the INET simulator [33] [34] which offers other communication protocols such as 4G and 5G to be able to evaluate scenarios other than platooning such as, a teleoperation scenario. Artery and Vanetza are also other potential simulators to be integrated into ComFASE. Artery enables V2X simulations based on ETSI ITS-G5 protocols and Vanetza has implemented the ETSI C-ITS protocol suite.

ACKNOWLEDGMENT

This work was supported by VALU3S project, which has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 876852. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Austria, Czech Republic, Germany, Ireland, Italy, Portugal, Spain, Sweden, Turkey.

REFERENCES

[1] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: The next computing revolution," in *Design Automation Conference*, 2010, pp. 731–736.

[2] Volvo models warn each other of slippery roads and hazards. https://www.media.volvocars.com/global/en-gb/media/pressreleases/251381/. (accessed: 05.09.2021).

[3] R. Svenningsson, J. Vinter, H. Eriksson, and M. Törngren, "MODIFI: A MODel-Implemented Fault Injection Tool," in *Computer Safety, Reliability, and Security*, E. Schoitsch, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 210–222.

[4] B. Sangchoolie, P. Folkesson, and J. Vinter, "A study of the interplay between safety and security using model-implemented fault injection," in *2018 14th European Dependable Computing Conference (EDCC)*. IEEE, 2018, pp. 41–48.

[5] S. Rösch, S. Ulewicz, J. Provost, and B. Vogel-Heuser, "Review of model-based testing approaches in production automation and adjacent domains-current challenges and research gaps," *Journal of Software Engineering and Applications*, 2015.

[6] R. Rana, M. Staron, C. Berger, J. Hansson, M. Nilsson, and F. Törner, "Increasing efficiency of iso 26262 verification and validation by combining fault injection and mutation testing with model based development," in *International Conference on Software Engineering and Applications*, vol. 2. SCITEPRESS, 2013, pp. 251–257.

[7] United Nations Economic Commission for Europe. https://unece.org/. (accessed: 08.12.2021).

[8] United Nations Economic Commission for Europe. https://wiki.unece.org/pages/viewpage.action?pageId=63310525. (accessed: 08.12.2021).

[9] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic Traffic Simulation using SUMO," in *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. [Online]. Available: https://elib.dlr.de/124092/

[10] OMNeT++ Simulation Models and Tools. https://omnetpp.org/download/models-and-tools. (accessed: 14.07.2021).

[11] D. Eckhoff and C. Sommer, "A Multi-channel IEEE 1609.4 and 802.11p EDCA model for the Veins framework," in *Proceedings of 5th ACM/ICST international conference on simulation tools and techniques for communications, networks and systems: 5th ACM/ICST international workshop on OMNet++.(Desenzano, Italy, 19-23 March, 2012). OMNeT*, 2012.

[12] M. Maleki and B. Sangchoolie, "SUFI: A Simulation-based Fault Injection Tool for Safety Evaluation of Advanced Driver Assistance Systems Modelled in SUMO," in *2021 17th European Dependable Computing Conference (EDCC)*, 2021, pp. 45–52.

[13] ——, "Simulation-based Fault Injection in Advanced Driver Assistance Systems Modelled in SUMO," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S)*, 2021, pp. 70–71.

[14] M. Segata, S. Joerer, B. Bloessl, C. Sommer, F. Dressler, and R. L. Cigno, "Plexe: A platooning extension for Veins," in *2014 IEEE Vehicular Networking Conference (VNC)*. IEEE, 2014, pp. 53–60.

[15] M. Tiloca, G. Dini, F. Racciatti, and A. Stagkopoulou, *SEA++: A Framework for Evaluating the Impact of Security Attacks in OMNeT++/INET*. Cham: Springer International Publishing, 2019, pp. 253–278. [Online]. Available: https://doi.org/10.1007/978-3-030-12842-5_7

[16] L. Sánchez-Casado, R. A. Rodríguez-Gómez, R. Magán-Carrión, and G. Maciá-Fernández, "NETA: Evaluating the Effects of NETwork Attacks. MANETs as a Case Study," in *Advances in Security of Information and Communication Networks*, A. I. Awad, A. E. Hassanien, and K. Baba, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 1–10.

[17] B. Mitchell. 802.11 Standards Explained. https://www.lifewire.com/wireless-standards-802-11a-802-11b-g-n-and-802-11ac-816553. (accessed: 23.07.2021).

[18] "IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments," *IEEE Std 802.11p-2010 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, IEEE Std 802.11n-2009, and IEEE Std 802.11w-2009)*, pp. 1–51, 2010.

[19] M. N. Tahir and M. Katz, "Performance evaluation of IEEE 802.11p, LTE and 5G in connected vehicles for cooperative awareness," *Engineering Reports*, vol. n/a, no. n/a, p. e12467. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/eng2.12467

[20] J. Govil and J. Govil, "4G Mobile Communication Systems: Turns, Trends and Transition," in *2007 International Conference on Convergence Information Technology (ICCIT 2007)*, 2007, pp. 13–18.

[21] S. Kumari and B. Singh, "5G standard : The next generation wireless communication system," *Journal of Interdisciplinary Mathematics*, vol. 23, no. 1, pp. 275–283, 2020. [Online]. Available: https://doi.org/10.1080/09720502.2020.1721922

[22] D. Eckhoff, C. Sommer, and F. Dressler, "On the necessity of accurate ieee 802.11p models for ivc protocol simulation," in *2012 IEEE 75th Vehicular Technology Conference (VTC Spring)*, 2012, pp. 1–5.

[23] "IEEE Trial-Use Standard for Wireless Access in Vehicular Environments (WAVE) - Multi-Channel Operation," *IEEE Std 1609.4-2006*, pp. 1–82, 2006.

[24] R. van der Heijden, T. Lukaseder, and F. Kargl, "Analyzing attacks on cooperative adaptive cruise control (CACC)," in *2017 IEEE Vehicular Networking Conference (VNC)*, 2017, pp. 45–52.

[25] F. Boeira, M. P. Barcellos, E. P. de Freitas, A. Vinel, and M. Asplund, "Effects of colluding Sybil nodes in message falsification attacks for vehicular platooning," in *2017 IEEE Vehicular Networking Conference (VNC)*, 2017, pp. 53–60.

[26] M. Iorio, F. Risso, R. Sisto, A. Buttiglieri, and M. Reineri, "Detecting Injection Attacks on Cooperative Adaptive Cruise Control," in *2019 IEEE Vehicular Networking Conference (VNC)*, 2019, pp. 1–8.

[27] J. Lastinec and M. Keszeli, "Analysis of Realistic Attack Scenarios in Vehicle Ad-hoc Networks," in *2019 7th International Symposium on Digital Forensics and Security (ISDFS)*, 2019, pp. 1–6.

[28] D. S. Berger, F. Gringoli, N. Facchi, I. Martinovic, and J. Schmitt, "Gaining insight on friendly jamming in a real-world ieee 802.11 network," in *Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless & Mobile Networks*, ser. WiSec '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 105–116. [Online]. Available: https://doi.org/10.1145/2627393.2627403

[29] M. Lichtman, J. D. Poston, S. Amuru, C. Shahriar, T. C. Clancy, R. M. Buehrer, and J. H. Reed, "A communications jamming taxonomy," *IEEE Security Privacy*, vol. 14, no. 1, pp. 47–54, 2016.

[30] V. Milanés and S. E. Shladover, "Modeling cooperative and autonomous adaptive cruise control dynamic responses using experimental data," *Transportation Research Part C: Emerging Technologies*, vol. 48, pp. 285–300, 2014.

[31] S. E. Lee, E. Llaneras, S. Klauer, and J. Sudweeks, "Analyses of rear-end crashes and near-crashes in the 100-car naturalistic driving study to support rear-signaling countermeasure development," *DOT HS*, vol. 810, p. 846, 2007.

[32] SUMO Collisions Outputs. https://sumo.dlr.de/docs/Simulation/Output/Collisions.html. (accessed: 09.12.2021).

[33] L. Mészáros, A. Varga, and M. Kirsche, *INET Framework*. Cham: Springer International Publishing, 2019, pp. 55–106. [Online]. Available: https://doi.org/10.1007/978-3-030-12842-5_2

[34] Veins_INET Subproject. https://veins.car2x.org/documentation/modules/#veins_inet. (accessed: 14.07.2021).