Simutack - An Attack Simulation Framework for Connected and Autonomous Vehicles

Andreas Finkenzeller, Anshu Mathur, Jan Lauinger, Mohammad Hamad, and Sebastian Steinhorst
Department of Computer Engineering
Technical University of Munich, Germany
Email: firstname.lastname@tum.de

***Abstract*—With the ongoing efforts toward autonomous driving, modern vehicles become increasingly digital and smart. Hence, the vehicle architecture including smart sensors, ECUs, and in-vehicle communication also faces new challenges to satisfy the ever-changing safety and security requirements. The complexity of the system naturally exposes many attack surfaces that demand for sound security solutions to protect the vehicle from potential intrusions. State-of-the-art approaches such as intrusion detection and intrusion response systems require lots of training and testing against various attack scenarios. However, implementing such attacks in real environments is difficult, expensive, and involves many legal and safety considerations. With Simutack, we present an open-source attack simulation framework that is capable of generating realistic attack scenarios for comprehensive security testing in the automotive development process. The framework integrates several classes of attacks, for instance, smart sensor attacks, V2X attacks, and attacks targeting the in-vehicle networks, which are all among the most commonly exploited attack vectors. We evaluate three common attack scenarios that showcase the applicability and capabilities of our work. In each scenario, the generated attack data is processed and returned to the simulation to visualize the attack's effect on the vehicle and its environment. Furthermore, a custom autopilot application demonstrates the attack's impact on autonomous driving systems.**

***Index Terms*—Security Framework, Attack Generation, Simulation, V2X, Connected and Autonomous Vehicle Security**

I. Introduction

In 1986, the German researcher Ernst Dickmanns and his team presented the *VaMoRs* project, a first approach to autonomous driving based on live-captured sensor data and real-time data processing [1]. This data-driven approach has proven to be very effective and is, thus, still in use today albeit in a much more sophisticated and complex fashion. Many recently deployed algorithms include Machine Learning (ML) techniques which cope with the large amount of generated data and the high complexity of current automotive systems. However, the necessary training of those algorithms requires lots of training data which is usually quite costly to acquire in real environments. Furthermore, many adversarial attacks which were published over the last decade, such as [2] and [3], have shown that security is also becoming an increasingly important concern for Original Equipment Manufacturers (OEMs). This observation implies that nowadays also multiple attack scenarios must be considered and tested in the development process in order to build resilient and secure vehicles. The implementation of such attacks, however, is very expensive and time-consuming. Most importantly, it also requires a special test environment to account for any possible safety, legal, and financial risks. Hence, we have to rely on sophisticated simulation environments instead. Although available automotive simulators like *CARLA* [4] and the *LGSVL* simulator [5] are quite comprehensive regarding sensor support, these frameworks were not designed to simulate adversarial attacks against autonomous driving systems. Connected and Autonomous Vehicles (CAVs), nonetheless, are very complex systems which comprise various components such as sensors, Electronic Control Units (ECUs), and communication interfaces. These components all expose many attack surfaces and are, thus, potential attack targets which need to be properly secured. Consequently, there is a great need for appropriate testing environments which safely generate realistic attack data for all required components.

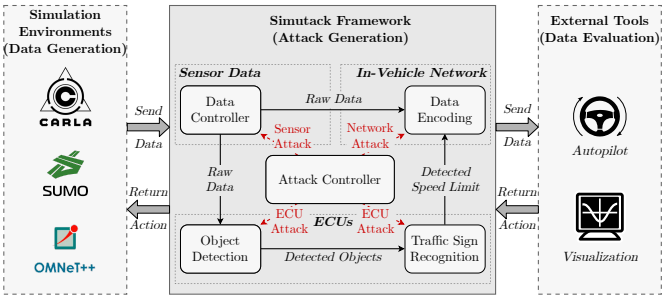


Fig. 1: System overview of our Simutack framework. The data generation is based on existing simulation environments that are integrated into our toolkit. The core parts are the subsequent steps to post-process the data in ECUs, encode it for network transmission, and apply attacks at all intermediate stages. Finally, the generated data can be used within external applications.

In this work, we present Simutack, an attack simulation framework for CAVs that is able to provide such a safe environment for comprehensive security tests including the generation of realistic vehicular data under various attack conditions as depicted in Fig. 1. In particular, we:

- introduce a modular and open-source framework[1] that provides a safe and comprehensive simulation environment to test adversarial attacks against CAVs (Sec. II),
- cover many automotive attack targets and provide sample

[1]Access code and demo videos here: https://github.com/tum-esi/simutack

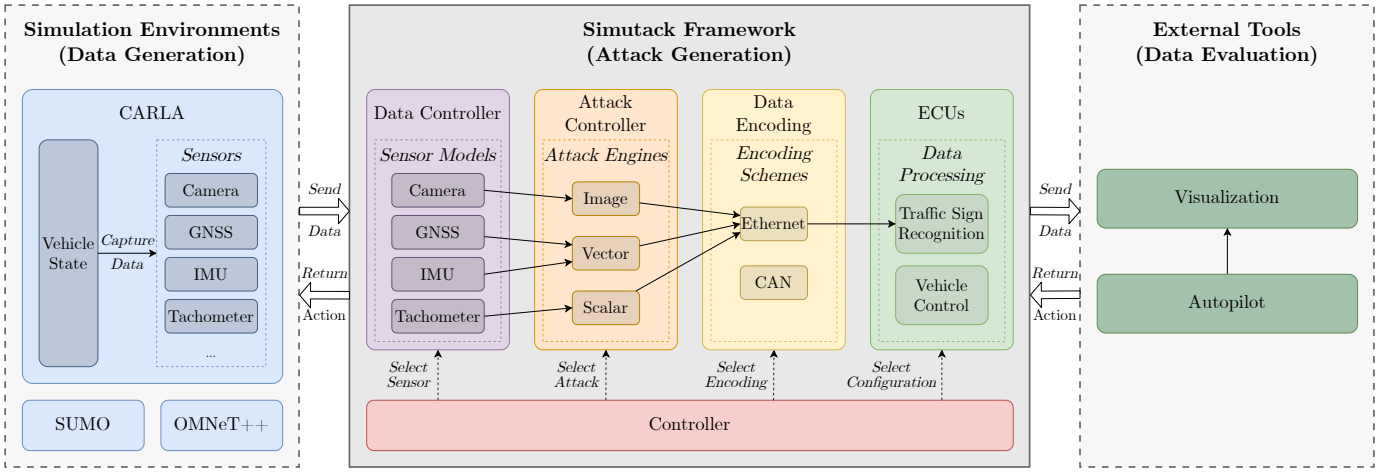


Fig. 2: Detailed system architecture of Simutack. The simulated sensor data is sent from CARLA to Simutack, where it traverses all processing stages before it can be used in external applications. Vehicle control inputs may be returned to the simulation which allows for closed-loop test scenarios.

implementations of well-known attack vectors including sensor, network, and Vehicle-to-Everything (V2X) attacks (Sec. III),
- provide a scenario-based evaluation of our framework and validate the generated simulation data by means of three realistic use cases that were selected due to their high relevance in current automotive systems (Sec. IV),
- show the advantages of Simutack's attack capabilities in comparison with other existing simulators (Sec. V & VI)

II. System Architecture

In the following paragraphs, we introduce the system architecture and Simutack's main components in more detail as shown in Fig. 2.

A. Data Generation

Because Simutack should serve as comprehensive security framework, it was designed to integrate multiple different data sources including sources for sensor and communication data. An important requirement for the simulated data, thereby, is its real-world viability. This is particularly vital for the generated sensor data. Therefore, we need to ensure that some basic constraints are met by all data samples, e.g., proper physical models and time causality. Additionally, it is important to have a single simulation source for all involved sensors to preserve realism and validity of the data. When there is an object on the road, for example, its presence must be reflected in camera, Lidar, and Radar recordings at the same time. Otherwise, common processing steps like sensor fusion approaches cannot be supported. The *CARLA* simulator [4], which is based on *Unreal Engine 4*, is a comprehensive simulation environment that was specifically designed for that purpose. It is widely used in research and supports all common automotive sensors, for example, camera, Inertial Measurement Unit (IMU), Global Navigation Satellite Systems (GNSS), Radar, and Lidar. *CARLA* is not directly integrated in Simutack to maintain the flexibility of leveraging also other data sources in the future. Therefore, Simutack rather uses a general abstract sensor model with an additional *CARLA*-specific interface implementation. Another benefit of this abstraction is a possible categorization of all sensors based on their output data type, i.e., whether the sensor produces scalar values, vector outputs, or some more complex data structures. This makes the framework very generic, especially regarding the addition of new sensors and attacks.

Besides *CARLA*, which focuses on sensor data generation, Simutack also integrates further simulation environments to enrich its capabilities. The *SUMO* simulation package [6] serves as plugin for *CARLA* to facilitate enhanced traffic control. This leads to more advanced application scenarios, such as platooning, which can be modeled with our framework, and also to more realistic sensor outputs. Furthermore, we leverage the *OMNeT++* toolkit [7] to simulate a V2X communication interface. *OMNeT++* allows to precisely model the propagation of wireless data packets considering attenuation, propagation delay, and other channel properties.

B. Attack Generation

Although the aforementioned simulation environments are readily available for normal sensor data generation, none takes potential adversarial attacks into account. Besides the integration of existing simulators, the introduction of an additional attack generation layer is, therefore, a major contribution and a unique characteristic of Simutack. For this purpose, we thoroughly analyzed both the attacks and the simulation platforms in order to successfully integrate the available building blocks into a unified framework.

Modern vehicles expose many different attack surfaces due to the heterogeneity and complexity of all comprising components and the entire system itself. It is, thus, desirable but also challenging to find an efficient approach which simulates as many different attack vectors as possible to develop appropriate countermeasures. Potential attacks may arise form smart sensors, ECUs, and communication networks alike. From a

data perspective, nevertheless, many attacks have the same impact on generated vehicular data which enables a respective clustering. To further elucidate this statement, we look at the familiar scenario of automotive navigation. The GNSS sensor captures the vehicle's current position and sends it via an internal communication bus to the navigation device. For the navigation device, it thereby makes no difference whether the sensor perception or the communication path was targeted by the adversary. Based on the received data, the device cannot distinguish where the data was modified. Ultimately, the ECU simply obtains erroneous position data and, hence, the navigation fails. With this observation, we define several attack classes based on the smart sensor's data type, i.e., scalar, vector, or complex outputs and the attack's effect on the captured data, which include among others, modification, injection, and deletion of the data frames. Afterwards, we just need to implement the finite set of defined classes and map the variety of desired attack vectors accordingly. With this approach, Simutack can simulate many smart sensor, ECU, and network attacks in a consistent and flexible manner that also allows for straightforward extensions with new classes and attacks. Further details on the actual attack implementations are discussed in Sec. III.

C. Data Encoding

Inside the vehicle, there exist multiple different communication busses to exchange data among sensors and ECUs. The most prominent representatives are the Controller Area Network (CAN), FlexRay, and Automotive Ethernet. Each bus system has its individual characteristics and requires a certain data encoding based on the respective standards. In order to achieve compatibility with other systems including physical Hardware-in-the-Loop (HIL) applications, Simutack offers the possibility to encode the simulated data accordingly. Thereby, the generated raw data is simply wrapped or encoded with the appropriate protocol headers.

Besides the mentioned in-vehicle communication, our framework further supports the simulation of V2X data to additionally cover external communication scenarios. In particular, we use the IEEE 802.11p wireless communication protocol that is readily available in *OMNeT++*.

D. ECUs

Another important part in CAV architectures are ECUs and the respective tasks they perform. ECUs are essentially embedded computers which have defined Input/Output (I/O) interfaces and run dedicated software to add certain driving or comfort features to the vehicle. Because they are in general interesting attack targets and are required for realistic in-vehicle communication, Simutack also contains an ECU simulation approach. We use OS-level virtualization techniques to implement isolated services that can be interconnected via predefined network interfaces which simulate the in-vehicle communication. The system architecture can be preconfigured with a provided system description to launch the desired containers and links for each scenario automatically.

III. Attack Implementation

After the general introduction of Simutack's attack generation approach in Sec. II-B, we now discuss different attack surfaces and the simulation of related practical attacks in more detail. The presented attacks cover relevant and well-known examples from the literature to showcase Simutack's capabilities. However, the selection shall not constitute an exhaustive list of what attacks can be implemented with our framework.

A. Sensor Attacks

Sensor attacks aim to affect the sensor's perception of the physical world in a malicious way. Typical examples are jamming and spoofing which either prevent any perception at all or generate bogus sensor readings, respectively [8]. The actual implementation of such attacks can be quite costly and problematic in practice as discussed earlier. On the other hand, applying the characteristic effect of a certain sensor attack on the captured output data in a postprocessing step is easy to perform once the effect is known. Therefore, we analyze the correlation of known sensor attacks, such as [9] and [10], and the respective changes in the generated output data. The previously mentioned categorization based on the data type allows for reusing obtained insights with multiple sensors. A sensor jam, for example, results in either no output at all or a default value outside the reasonable measurement range dependent on the sensor. Constant offsets to the data can simulate systematic measurement errors that may arise from additive noise. Spoofed input signals result in arbitrary sensor readings within the defined perception scope. In addition to these rather general relationships, Simutack supports also the simulation of more specific attacks, for instance, the camera blinding attack discussed in [3]. By pointing some light source toward the camera, the auto exposure mechanism is disturbed and too much light is perceived by the camera chip, appearing as white pixels in the final image. Similar results can be achieved in simulation with appropriate image postprocessing steps. An important aspect, however, is to also consider the inter-frame dependency of this attack. When the vehicle is moving, two consecutive images result from different camera angles, while the position of the light source might be fixed in the scene. This problem can be solved with the use of basic projection geometry to compute the exact position and size of the affected pixels in the image to account for the moving camera.

B. Network Attacks

Once the data is captured, the sensors forward the records to ECUs via dedicated bus systems. From an attacker's perspective, this reduces to a simple network packet transmission, regardless which data or which bus system is utilized. Common network attacks comprise Denial-of-Service (DoS), message falsification, and message spoofing attacks. The first attack category essentially has similar repercussions as sensor jamming attacks, i.e., no (meaningful) data reaches the destination due to packet drops. The second category enables arbitrary message modifications including transmission-related bit errors.

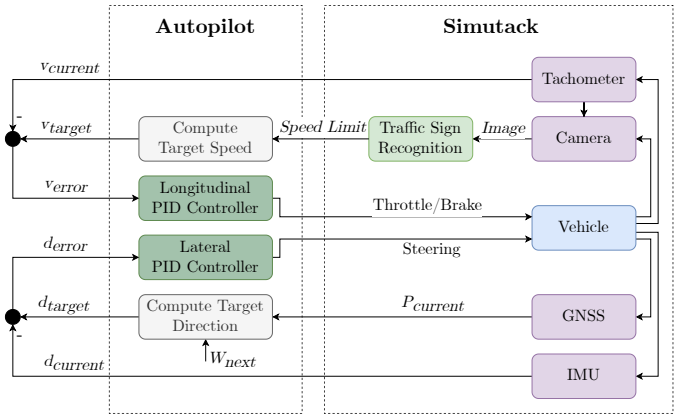


Fig. 3: The autopilot's internal control logic. We use the algorithm to evaluate the validity of our simulated attack data. The captured sensor data is used to calculate vehicle control inputs which is looped back to the simulation to have a closed-loop control system. In case of an attack, the effect is directly visible in the simulation by the captured sensor data.

When spoofing messages, the attacker injects bogus packets with forged origins to trick the recipient into erroneous actions or decisions. The mentioned attacks, thereby, apply equally to in-vehicle communication and external V2X communication, respectively. Due to the high similarities and the comparable effect on the output data, in most cases, sensor and network attacks can be jointly implemented, especially when the data is used as input to test ECUs and related algorithms.

C. ECU Attacks

A third way of attacking CAVs is by targeting the ECUs. An attacker can exploit a simple software bug or some other flaw to take over the unit and control the hosted tasks. Additionally, a single ECU can serve as starting point to pivot to other components via the in-vehicle networks. Despite the large attack potential, the available implementations and functionalities of ECUs are so diverse that it seems impossible to find a general model that covers all the relevant internals and details and is, thus, beyond the scope of Simutack. Nevertheless, if we consider an ECU as a black box device and only focus on its data flow while neglecting the underlying complexity, we are still able to model the impact of many ECU attacks on other components. Furthermore, we can use it to validate the effect of other attacks based on the observed output. For example, if the attacker can compromise the ECU that detects approaching traffic signs, which is relevant for autonomous driving and other control algorithms, this traffic sign information becomes unreliable. Hence, this attack poses a threat to other systems inside the vehicle which rely on traffic sign detection and can be used for a security analysis of the affected components.

IV. Evaluation

In order to validate our framework and the generated attack data, we provide a scenario-based evaluation to show the real-world viability and applicability of Simutack. In the following paragraphs, we present three selected scenarios that cover a variety of relevant attack vectors against CAVs which are also widely discussed in the literature.

A. Autopilot Application

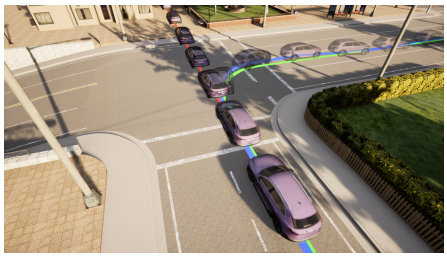
Data validation can be a difficult task because there exists no general definition of valid attack data. In our case, a reasonable criterion could be the similarity between simulation and real data. However, what are good and fair metrics for this comparison and is genuine reference data always available? Since there is no general answer for these questions, we first put the data into a certain context and give a specific meaning to it before we proceed with the actual evaluation. In particular, we developed a basic autopilot application that uses the provided sensor data either directly or after some processing steps to produce vehicle control outputs that are returned to the simulation. This allows to monitor the effect of an applied attack directly in the simulation by means of the vehicle's driving behavior and the continuously captured sensor data.

The autopilot is based on two Proportional-Integral-Derivative (PID) controllers similar to the idea provided in *CARLA*'s `controller.py` module[2]. One controller governs the vehicle's longitudinal movement, while the other handles all lateral motions. One major difference to *CARLA*'s implementation is that our controllers are purely based on live sensor and ECU data. Hence, there is no internal information from the *CARLA* simulation environment needed, and the autopilot can be entirely implemented as external application. Also, instead of random navigation, we provide predefined routes that the vehicle should follow. The computed steering and throttle actions are returned to the simulation to create a closed-loop control scheme as depicted in Fig. 3.

B. Scenario 1: GNSS Attack

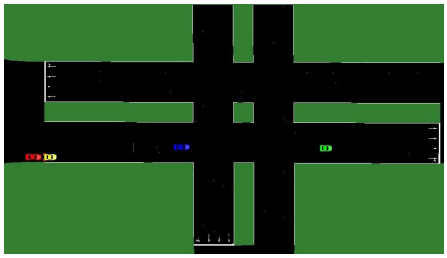
In the first scenario, we perform a DoS attack against the GNSS sensor. The previously introduced autopilot application uses this sensor to obtain information about the vehicle's current position. Based on the current position and the given target position, the controller computes a target vector that represents the ideal driving direction towards the target. The difference of this direction and the vehicle's current orientation, which can be acquired from the IMU sensor, is the error term that the lateral PID controller uses for its steering control. In the default case, the target direction is correctly determined which lets the vehicle properly take the curve to follow the ideal path (green line) as shown in Fig. 4a. In case of the DoS attack, the position updates from the sensor stop. Hence, the target direction cannot be properly computed anymore resulting in incorrect steering behavior visualized by the red line. Fig. 5 depicts the vehicle's orientation over time captured by the GNSS sensor in both the normal and attack case to further elucidate the impact of the attack.

[2]https://github.com/carla-simulator/carla/blob/master/PythonAPI/carla/agents/navigation/controller.py



(a) GNSS Attack.

(b) Camera Blinding Attack.



(c) V2X Attack.

Fig. 4: Visualization of the three presented attack simulation scenarios.

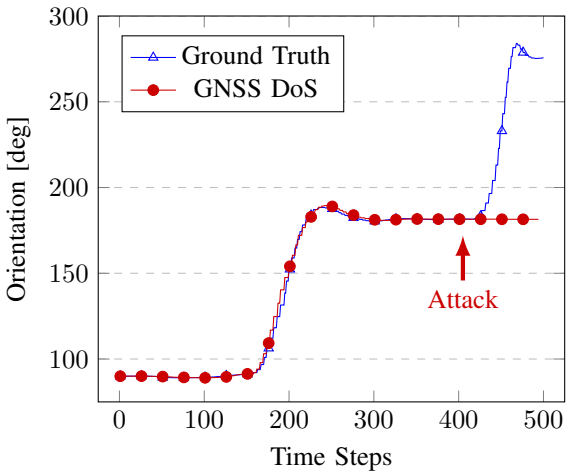


Fig. 5: GNSS DoS Attack Scenario. Due to the missing position updates, the autopilot is not able to properly control the vehicle. Hence, it crashes into the wall (red) instead of driving the curve (blue).

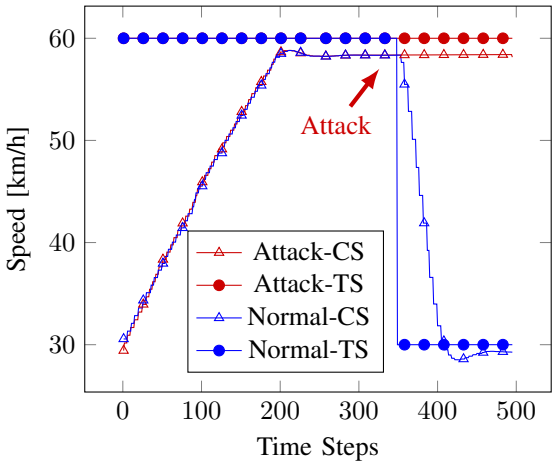


Fig. 6: Camera Blinding Attack Scenario. The autopilot aims to match the vehicle's current speed (CS) to the computed target speed (TS). In case of the camera blinding attack, the appropriate speed limit cannot be extracted from the provided image which results in ignoring the speed limit.

C. Scenario 2: Camera Blinding Attack

In the second scenario, we showcase a more complex attack implementation, namely the camera blinding attack. For this purpose, we make use of the autopilot application from Sec. IV-A but with a focus on the longitudinal controller instead. According to Fig. 3, the vehicle's current speed is adapted to match the last applicable speed limit that was detected by the appropriate ECU. The implemented detection mechanism is a two-fold approach. First, the traffic sign is extracted from a provided camera image using the *Yolov4* object detection framework [11]. In a second step, the sign's text is read by means of the optical character framework *EasyOCR* [12] to obtain the actual speed limit. In the default case, the algorithms can successfully extract the speed limit and the PID controller adapts the vehicle's velocity accordingly. The additional camera blinding attack which is visualized in Fig. 4b, however, prevents the *Yolov4* framework from detecting the traffic sign. Hence, also no speed limit can be extracted from the provided image. Consequently, the autopilot fails to respect the speed limit in force and continues at the higher speed. Fig. 6 visualizes the two stated cases.

D. Scenario 3: V2X Attack

In the last scenario, we illustrate the impact of a message falsification attack on the V2X communication between several vehicles in a platooning application. The attack is implemented on a platoon consisting of three connected vehicles, where the malicious attacker is also part of the platoon. In this constellation, all the vehicles follow the leader's driving instructions, which it transmits using V2X communication based on the IEEE 802.11p wireless standard.

In the reference case, the leader vehicle detects a roadblock sign on the current driving lane and decides to switch to the lane to its left. Consequently, it also communicates the new lane information to all its succeeding vehicles which follow as expected. In the attack case, the leader also shares the information as previously, however, the second vehicle in the platoon turns into the attacker. It alters the received V2X packet to stay in the initial lane before it broadcasts the malicious packet further to the victim, i.e., the last platoon vehicle. On receiving the falsified driving information from the attacker, the victim mistakenly continues to drive on the same lane and eventually collides with the blocking object. Fig. 7

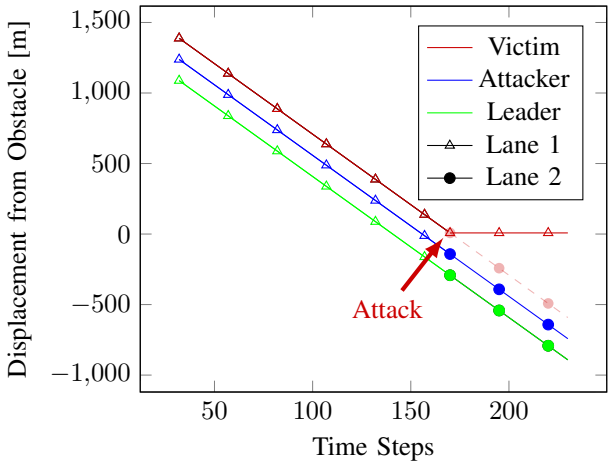


Fig. 7: V2X Attack Scenario. In the attack case, the victim is provided with malicious V2X messages, misses to change to lane 2 (circle) and eventually crashes into the roadblock on lane 1 (triangle).

visualizes the portrayed scenario. In particular, it shows the distance of each vehicle to the roadblock and their respective driving lane information over time.

V. Discussion

The presented scenarios showcase an exemplary usage of Simutack and, most importantly, show that our framework is able to simulate appropriate attack data to create a realistic but safe testing environment for CAVs. The security of individual components can be safely evaluated by generating the desired attack data and analyzing the resulting output and, particularly, its effect in the simulation environment. If the behavior is not aligned with the expectations, the component to be tested is not yet sufficiently secure against the simulated attack. Furthermore, *CARLA* is a well-established tool which is widely used in academia and the industry and is considered to generate trustworthy data. Since Simutack relies on *CARLA* for its sensor data, the generated data is also trustworthy. Nevertheless, while using the simulated sensor data, we additionally propose a comprehensive attack layer in our framework which can simulate highly relevant attacks against CAVs. This extension turns out to be a significant improvement over current automotive test environments such as *CARLA* and comparable simulators regarding the design and testing capabilities of secure autonomous systems.

VI. Related Work

As previously mentioned, many simulators are used in the development of autonomous driving systems to efficiently analyze many driving scenarios within a safe, risk-free, and flexible environment [13]. These simulators can be divided into multiple groups based on the part of the automotive system that they simulate: 1) Simulators that focus on various sensor suites including Lidar, camera, GNSS, etc. (e.g., *CARLA* [4] and *LGSVL* [5]), 2) simulators that target in-vehicle networks (e.g., VEOS [14] and CANoe [15]), and 3) simulators that cover the communication with the outside world (e.g., *Veins* [16], *SUMO* [6], and *OMNeT++* [7]). The simultaneous simulation of multiple system parts requires the integration of many individual tools. However, this integration also involves a considerable amount of time and effort.

Table I: Comparison of Simutack with other systems.

Solution	Systems			Attacks			Open Source
	In-vehicle	Smart Sensors	V2X	In-vehicle	Smart Sensors	V2X	
CARLA [4]	○	●	○	○	○	○	●
LGSVL [5]	○	●	○	○	○	○	●
CANoe [15]	●	○	○	○	○	○	○
[17]	○	●	○	○	●	○	●
[18]	○	●	○	○	●	○	○
[19]	○	○	●	○	○	●	◑[1]
PASS [20]	○	●	○	○	●	○	○
[21]	●	○	○	●	○	○	○
[22]	○	●	●	○	●	●	●
Simutack	●	●	●	●	●	●	●

○ no support, ◑ partial support, ● full support

[1] Data-set was shared only.

Furthermore, all these simulation environments were not designed to cover adversarial attacks against autonomous driving systems. Recently, many researchers have started to tackle this gap by using several platforms to simulate attacks against various components of a smart vehicle. In [17], the authors extended the *CARLA* driving simulator to implement security attacks against GNSS sensors. In [18], Nesti et al. utilized *CARLA* in order to implement and evaluate Adversarial Patch Attacks. Finally, Iqbal et al. proposed a methodology for generating attack data for V2X communication using the Eclipse MOSAIC simulation framework [19]. This framework uses two vehicular simulation tools, namely *OMNeT++* and *SUMO*, to simulate the attack and visualize it in 2D.

In Table I, we compare our proposed work with some of the solutions presented above. Unlike other existing solutions (e.g., [20], [21], [22], etc.), our framework is not limited to one perspective but addresses different aspects of the automotive system including in-vehicle networks, smart sensors, and V2X communication. Additionally, it supports a variety of relevant attacks that can target all these system parts. Finally, our solution is an open-source implementation, so that it can also be used by other researchers.

VII. Conclusion

In this work, we present a comprehensive attack simulation framework for connected and autonomous vehicles. We incorporate multiple attacks against different parts of autonomous

vehicles, namely sensor, ECU, and network attacks into a single open-source toolkit that can serve as a safe test environment to design secure and resilient system components. Our three presented evaluation scenarios confirm Simutack's applicability in realistic driving situations and show the validity and practical relevance of the simulated data.

ACKNOWLEDGMENT

This work has received funding from The Bavarian State Ministry for the Economy, Media, Energy and Technology, within the R&D program "Information and Communication Technology", managed by VDI/VDE Innovation + Technik GmbH.

REFERENCES

[1] W. Thiel, "The VaMoRs Was the World's First Real-Deal Autonomous Car," https://www.web2carz.com/autos/car-tech/6396/the-vamors-was-the-worlds-first-real-deal-autonomous-car, [Accessed: 17-October-2022].

[2] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, no. S 91, 2015.

[3] J. Petit, B. Stottelaar, M. Feiri, and F. Kargl, "Remote attacks on automated vehicles sensors: Experiments on camera and lidar," *Black Hat Europe*, vol. 11, 2015.

[4] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, S. Levine, V. Vanhoucke, and K. Goldberg, Eds., vol. 78. PMLR, 13–15 Nov 2017, pp. 1–16. [Online]. Available: https://proceedings.mlr.press/v78/dosovitskiy17a.html

[5] G. Rong, B. H. Shin, H. Tabatabaee, Q. Lu, S. Lemke, M. Možeiko, E. Boise, G. Uhm, M. Gerow, S. Mehta, E. Agafonov, T. H. Kim, E. Sterner, K. Ushiroda, M. Reyes, D. Zelenkovsky, and S. Kim, "Lgsvl simulator: A high fidelity simulator for autonomous driving," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE Press, 2020, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ITSC45102.2020.9294422

[6] D. Krajzewicz, G. Hertkorn, C. Rössel, and P. Wagner, "Sumo (simulation of urban mobility)-an open-source traffic simulation," in *Proceedings of the 4th middle East Symposium on Simulation and Modelling (MESM20002)*, 2002, pp. 183–187.

[7] A. Varga and R. Hornig, "An overview of the omnet++ simulation environment," in *1st International ICST Conference on Simulation Tools and Techniques for Communications, Networks and Systems*, 2010.

[8] Z. El-Rewini, K. Sadatsharan, N. Sugunaraj, D. F. Selvaraj, S. J. Plathottam, and P. Ranganathan, "Cybersecurity attacks in vehicular sensors," *IEEE Sensors Journal*, vol. 20, no. 22, pp. 13 752–13 767, 2020.

[9] N. O. Tippenhauer, C. Pöpper, K. B. Rasmussen, and S. Capkun, "On the requirements for successful gps spoofing attacks," in *Proceedings of the 18th ACM conference on Computer and communications security*, 2011, pp. 75–86.

[10] I. Rouf, R. Miller, H. Mustafa, T. Taylor, S. Oh, W. Xu, M. Gruteser, W. Trappe, and I. Seskar, "Security and privacy vulnerabilities of {In-Car} wireless networks: A tire pressure monitoring system case study," in *19th USENIX Security Symposium (USENIX Security 10)*, 2010.

[11] fredotran, "Traffic sign detector using yolov4," https://github.com/fredotran/traffic-sign-detector-yolov4.

[12] JaidedAI, "Easyocr," https://github.com/JaidedAI/EasyOCR.

[13] P. Kaur, S. Taghavi, Z. Tian, and W. Shi, "A survey on simulators for testing self-driving cars," in *2021 Fourth International Conference on Connected and Autonomous Driving (MetroCAD)*, 2021, pp. 62–70.

[14] "dSpace VEOS," https://www.dspace.com/en/pub/home/products/sw/simulation_software/veos.cfm.

[15] "CANoe," https://www.vector.com/int/en/products/products-a-z/software/canoe/.

[16] "The open source vehicular network simulation framework," https://veins.car2x.org/documentation/.

[17] J. Lauinger, A. Finkenzeller, H. Lautebach, M. Hamad, and S. Steinhorst, "Attack data generation framework for autonomous vehicle sensors," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2022, pp. 128–131.

[18] F. Nesti, G. Rossolini, S. Nair, A. Biondi, and G. Buttazzo, "Evaluating the robustness of semantic segmentation for autonomous driving against real-world adversarial patch attacks," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 2280–2289.

[19] S. Iqbal, P. Ball, M. H. Kamarudin, and A. Bradley, "Simulating malicious attacks on vanets for connected and autonomous vehicle cybersecurity: A machine learning dataset," in *2022 13th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, 2022, pp. 332–337.

[20] Z. Hu, J. Shen, S. Guo, X. Zhang, Z. Zhong, Q. A. Chen, and K. Li, "PASS: A system-driven evaluation platform for autonomous driving safety and security," in *NDSS Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*, 2022.

[21] J. Laufenberg, T. Kropf, and O. Bringmann, "A framework for can communication and attack simulation," in *2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring)*, 2022, pp. 1–7.

[22] M. L. Bouchouia, J.-P. Monteuuis, H. Labiod, O. Jelassi, W. B. Jaballah, and J. Petit, "Demo: A simulator for cooperative and automated driving security," in *NDSS Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*, 2022.