Attack Data Generation Framework for Autonomous Vehicle Sensors

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

***Abstract*—Driving scenarios of autonomous vehicles combine many data sources with new networking requirements in highly dynamic system setups. To keep security mechanisms applicable to new application fields in the automotive domain, our work introduces a security framework to generate, attack, and validate realistic data sets at rest and in transit. Concerning realistic data sets, our framework leverages autonomous driving simulators as well as static data sets of vehicle sensors. A configurable networking setup enables flexible data encapsulation to perform and validate networking attacks on data in transit. We validate our results with intrusion detection algorithms and simulation environments. Generated data sets and configurations are reproducible, portable, storable, and support iterative security testing of scenarios.**

***Index Terms*—Security Framework, Data Generation, Intrusion Detection, Autonomous Vehicles**

I. Introduction

The emerging reality of self-driving cars relies heavily on data provided by the massive number of integrated sensors within these vehicles. Autonomous vehicles are built with many control algorithms which process sensor data and induce appropriate driving actions. Recently, many control algorithms have been implemented based on machine learning techniques. This makes the existence of data sets for training such algorithms a crucial requirement. Similarly, realistic data sets containing nominal sensor data and attack data are essential to develop and validate security solutions which detect attacks against a system.

In recent years, many sensor data sets for autonomous driving have been made public [1]. However, the number of data sets containing attack data remains limited due to many restrictions such as confidentiality rights, vulnerability disclosure, hardware costs, etc. Furthermore, generating realistic attack data by exploiting actual vulnerabilities of autonomous vehicles in an operational environment is a very challenging process, not only due to technical difficulties. Regulatory and legal requirements (e.g. authorization of public transportation authorities, car manufacturers, etc) hinder vehicle attacks even further [2]. Additionally, validation of generated attack data requires specific detection algorithms which are typically part of Intrusion Detection Systems (IDSs). Hence, falling back on realistic sensor and attack data generation is a necessity and objective of this work.

Contribution: To solve the issue of missing sensor and sensor-specific attack data, we introduce an attack data generation framework with the additional capability of validating realistic sensor-specific attack data (see Fig. 1). The modular architecture of the framework has three major domains, namely (i) the data source domain which reads in realistic data from real autonomous vehicle data sets (e.g. Argoverse [3]) as well as synthetic data, generated by a simulation environment (e.g. *CARLA* [4]), (ii) the attack generation domain to configure different types of attacks on networking and application layers, and (iii) the attack validation domain which measures the impact of attacks. To summarize, we:

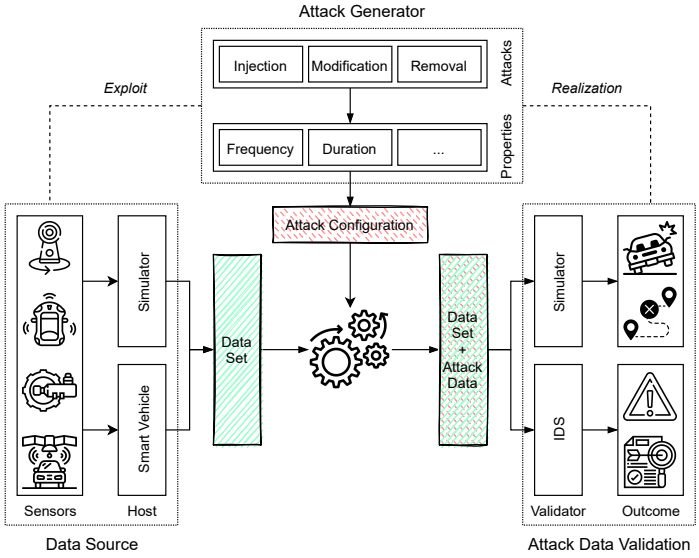


Fig. 1: High-level overview of the attack data generation framework. The workflow is based on configurations of data sources and attacks. Outgoing attack data sets are vaildated by IDS algorithms and a simulation environment.

- Design and develop a scalable attack generation framework[1] for autonomous vehicles.
- Introduce a scenario-based configuration setup providing reproducibility and portability of security tests.
- Present and validate multiple attack types to demonstrate the flexibility of the framework.

II. System Model

We describe our system logic shown in Fig. 1 on a higher level to abstract the scope of parameters into categories which allow mappings between data sources, attacks, and validation techniques.

A. Data Source

In order to make our framework compatible with a large number of different sensors, we categorize the sensors based on their data output. Most of the available sensors simply produce a scalar value such as a temperature or a pressure value. More sophisticated sensors, such as cameras, require a vector or even more complex data structures like matrices or custom data types to describe the captured data. With this classification, it is easy to add new sensors to our framework by simply putting them into one of the existing categories. Concerning network data, we consider OSI layers of link, network, transport, and application.

[1] https://github.com/tum-esi/attack_generation_framework

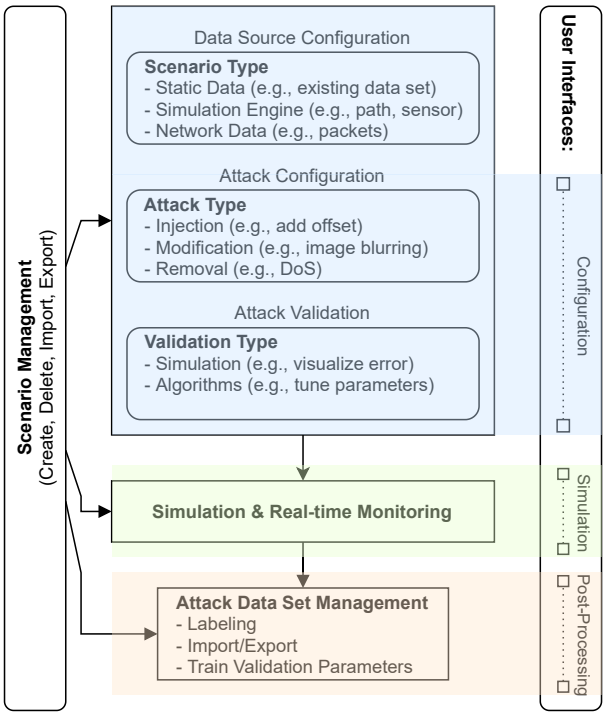


Fig. 2: Framework flowchart with stages of scenario definition and configuration, simulation and monitoring, and attack data set management.

B. Attack Generator

For the attacks, we use a similar approach where we group the attacks based on their actual effect on the data value and map these attacks to every sensor based on the known vulnerabilities of that sensor. Currently, our framework supports three main attack categories: injection, modification, and removal. To implement every attack, different parameters need to be configured and scheduled. These parameters include attack frequency, attack duration, value ranges, etc. It is important to note that the actual implementation of attacks can vary drastically and depends on vulnerabilities and resources of the system, thus, our work does not explore the space of possible attacks but considers techniques to produce verifiable attack data.

C. Attack Data Validation

The framework supports multiple validation approaches to validate the generated attack data. These approaches include feeding the generated data set, which includes attack data, to a simulation environment to demonstrate the attack's impact visually. Another strategy is to use the generated data set as an input for statistical models, which, in turn, allow attack validation. To support validation techniques, our framework allows automatic as well as manual labeling of generated attack data sets.

III. Attack Generation Framework

To connect the domains of the system model from Sec. II, we define a scenario management approach which allows generation of attack data sets with reproducible configurations. The following subsections describe the workflow to generate these data sets.

A. Workflow

The high-level workflow of the attack data generation framework, as illustrated in Fig. 2, can be separated into three workflow phases with (i) the *configuration* phase to incorporate configuration settings of the data source, attack, and validation domain into individual scenarios, with (ii) the *simulation* phase for running and monitoring the scenario which has been set up, and with (iii) the *post-processing* phase for post-processing actions on generated data sets. We divide scenarios into different types and elaborate on scenario structures in Sec. III-B. During scenario execution in the *simulation* phase, users directly see the effects of the configurations and may return to the *configuration* phase if output data is not reflecting desired goals. The *post-processing* phase focuses on the final result of the generation, allowing adjustments or addition of labels. If possible, analytical techniques can be applied for validation. From the prototype perspective, all three phases are supported with individual User Interface (UI) pages.

B. Scenario Management

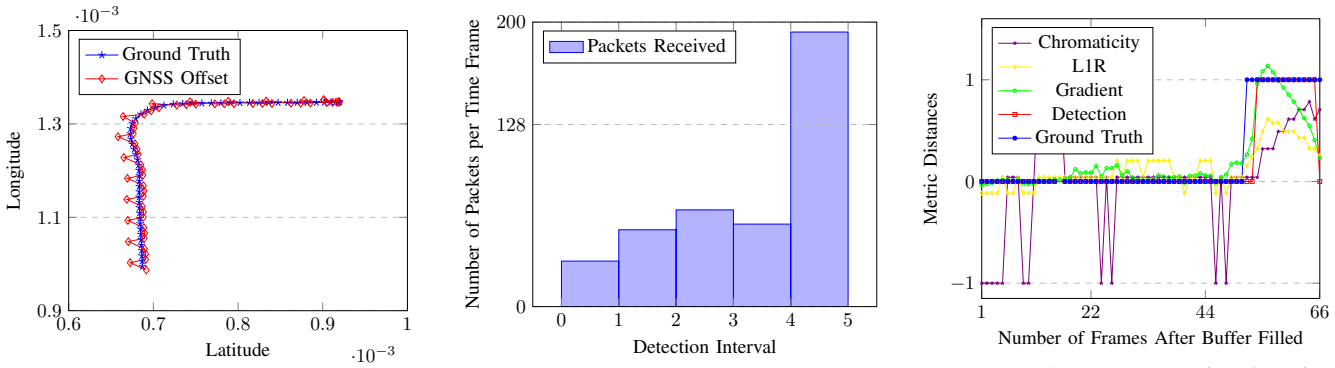
Scenario objects bind configuration logic about data sources, attacks, and validation techniques. It is not possible to generalize all configuration parameters and apply the same attack to different data sources, which is why we abstract scenarios into different types. The definition of scenario types is mainly driven by the data structure of our different data sources. The following itemization outlines differences among data sources and maps possible attacks. In addition, we name software modules which produce configured features but provide a detailed description of these software modules in Sec. III-C.

- The scenario type *driving simulation* makes use of the *simulator bridge* software module to configure autonomous driving scenarios. At the same time, sensors can be configured, added to the vehicle, and attacked to produce realistic attack sensor data.
- If the scenario is of type *networking*, users can create and scale virtual networking setups based on pre-defined networking resources via the *virtual networking* module. It is possible to define containers to attack existing resources of the configured virtual networking setup. Traffic between networking peers encapsulates either freshly generated sensor data or data from real-world data sets.
- The type *static data* lets users select existing data sets and apply modification attacks on a percentage of data items. The validation phase of this scenario type allows the creation of analysis models which store trained settings of parameter tuning. As an example, image frames generated by the *driving simulation* scenario type can be validated with statistical analysis models.

C. Prototype Software Modules

From the software development perspective, we created four main software modules to build the framework logic. The next enumeration elaborates on the individual modules.

1) **Controller:** This module is the core part of the framework and the first point of interaction from a user perspective. Acting as a web server, the *controller* exposes a UI which lets users navigate between workflow phases. Additionally, the *controller* module has compatibility with interfaces of other software modules (*simulator bridge*, *database*, and *virtual networking*) to expose features of other modules via the UI. Hence, the *controller* has the ability to support each workflow phase with features coming from other main software modules.
2) **Simulator Bridge:** The purpose of the *simulator bridge* is to provide connectivity towards existing autonomous driving simulators. Until now, our framework is compatible with



(a) Use case 1: GNSS offset attack. (b) Use case 2: DoS attack. (c) Use case 3: Image tampering detection.

Fig. 3: Validation of the generated attack data based on three use cases: (a) shows constant offset GNSS attack ($0.5m$ longitude, $2m$ latitude) with period $1s$. (b) shows network traffic monitoring and attack detection with pre-determined static threshold of 128. (c) shows real-time short/long-term memory LDA detection.

CARLA and support for the *SVL Simulator* [5] is planned. By defining a driving scenario on the *configuration* page of our framework, driving simulator compatibility provides realistic generation of sensor data including types such as GNSS, camera, IMU, LIDAR, radar, etc. Additionally, the *simulator bridge* supports to pass attack configurations to respective simulation environments.

3) **Virtual Networking:** The *virtual networking* modules supports dynamic configurations of virtual networking environments. Currently, we leverage *Docker* to manage virtual networks which support three types of containers for network traffic generation. *Data generation* containers have the ability to periodically send arbitrary data to a destination container. *Attack generation* containers have the same configuration parameters as data generation containers but can be configured and replicated differently. *Receiver* containers act as destinations and support configurable network traffic collection. This allows collection of network traffic from all layers.
4) **Database:** The *database* module stores scenarios and configurations as objects for reloading and replay purposes. Concerning data set storage, currently supported formats are .cap and .csv.

IV. Use Cases & Evaluation

Since data set generation depends on experimental setups [1], we describe attack data generation with use cases. Due to system requirements introduced by the *CARLA Docker* image, we evaluate our use cases on a computer with an Intel i7-8700K CPU, a Nvidia GTX 1080-Ti graphics card, 32 GB of RAM, and 256 GB SSD storage.

A. Autonomous Driving Sensor Attack and Offset Detection

Description: To attack sensors, our framework allows configurations that inject, modify, or remove specific data items which reflect value ranges of sensors under attack. Timestamps augment data items and replace removed data samples for labeling purposes. In order to detect attacks, simulation scenarios are executed twice using comparable ground truth paths.

Evaluation: To evaluate an exemplary sensor attack, Fig. 3a shows how a constant GNSS offset modification affects the location of the vehicle. The figure shows consecutive longitude and latitude values in periodic intervals. Longitude and latitude value pairs refer to a fix point location in the simulation environment and do not relate to real-world locations. The outcome of the sensor attack causes the vehicle to leave its scheduled path. Thereby, alternating GNSS offset values show the location deviation inside the simulator environment. As a direct consequence, sensor values attached to the simulated vehicle are affected as well (e.g. camera direction changes).

B. Network Traffic Denial of Service (DoS) Attempt and Detection

Description: As already outlined and to model networking traffic and access all networking layers of packets, our framework supports container-driven virtual networking configurations, where further communication protocols will be added in the future. Taking the *networking* scenario type, one sample use case is to produce a camera image stream between a sending and receiving container. We configure an attacking container to perform a DoS attack targeting the receiver node. From the receiver side, we use *Wireshark* as the network analyser tool to access network traffic of the scenario. Fig. 3b shows the data collection of the receiver. Here, the increasing number of detection intervals measures the number of received packets.

Evaluation: To detect and validate networking attacks, our framework supports a ring-buffer memory capturing of packets and compares traffic distributions between different time frames. The attack is scheduled to occur at round 4. As visualized in Fig. 3b, the number of packets increases to 197 packets. Our ring-buffer traffic analysis at the receiver side triggers a detection after 0.238 ms, when using a pre-defined threshold value of 128 packets.

C. Image Tampering and Real-time Threshold Detection

Description: To showcase an attack on static real-world data sets, we perform two types of image tampering attacks, namely image blacking and blurring. To apply these attacks, users are required to configure a percentage of an image array that should be modified. To validate image tampering and expecting an constant input stream of images, we reproduced the real-time image tampering detection algorithm from [6]. This algorithm depends on a Linear Discriminant Analysis (LDA) for threshold determination. Autonomous vehicles typically scan environments before their deployment [7]. This allows data-driven parameter tuning of the image tampering detection threshold which, when set up, predicts new incoming images automatically.

Evaluation: To evaluate a blurring attack on a static data set, we considered the training data set 1 of [3], specifically, the data of the centered front camera. After generating two new data sets using the image blacking and image blurring attacks, training of the short LDA-based threshold provided training parameters as shown in row 1 of Tab. I. Using the trained LDA threshold

Tab. I: Image tampering LDA parameters; Accuracy, True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN)

Attack	Test Phase	Accuracy	TP	FP	TN	FN
Blurred	Training	95.45 %	18	0	1	3
Blurred	Prediction	95.45 %	51	0	3	12
Blacked	Prediction	92.42 %	51	0	5	10

model, Fig. 3c shows real-time image tampering detection after injecting 2 malicious images into the blurred image attack data set. Accuracy results of LDA parameters can be found in row 2 of Tab. I. Detection accuracy decreased slightly to 92.42% in the case of blacked images, as indicated by row 3. Besides the model parameters, Fig. 3c shows the short-term long-term buffer algorithm which compares different image metrics. Here, chromaticity histogram comparison performs a color analysis, the L1R score compares brightness, and gradients compare edges in images. Further information about these metrics can be found in the work [6]. Every line of the chart represents the differences between short-term and long-term buffer values of the described metrics. An attacked image causes the difference to increase above the threshold which allows detection. Every new image that is passed to the algorithm counts as a new frame for comparison, allowing real-time detection of image tampering. The ground truth value is a reference to the detection line which indicates a successful detection after the occurrence of two malicious frames.

V. Related Work

The main related work, presented in [8], introduces an Attack Traffic Generation (ATG) toolkit for security testing of the Controller Area Network (CAN) bus message protocol. In their work, Huang et al. analyze the CAN bus attack model and identify injection and modification attacks with configurable timing behavior as main attack scenarios. With capturing, attack configuration, traffic generation, labeling, and formatting stages, the ATG toolkit is able to generate predefined attacks automatically. From the high-level architecture perspective, the ATG framework is similar to our architecture. Similarities are the separation of communication interfaces towards data sources, the Graphical User Interface (GUI), and the file processing module. The ATG task processing unit and bus accessing layer provide similar features as our attack engine and controller loop. Concerning differences, we use an extra sensor manager module to maintain sensor state and configurations, thus relaying connections between the attack engine and the controller unit. Further, our framework focuses on a wide range of Internet of Vehicles (IoV) sensor attacks with less focus on communication protocols. Another point is the abstraction towards data sources which enables our framework to process data sets, simulator data, or virtual networking traffic.

The work in [9] lists seven data sets with attack data that have been collected between 2017 and 2020. However, these data sets contain mainly CAN-related attacks. Some data sets contain attack data related to one type of sensors, such as the one published in [10] which contains camera attacks data, whereas other existing data sets either do not provide domain-specific artifacts, or do not contain automotive-related data and attacks [11]. In contrast, our framework is able to generate automotive-related data sets that contain attack data related to different set of sensors (e.g. GNSS, Camera, etc.).

VI. Conclusion

This work introduces a framework to generate attack data sets to support intrusion detection algorithms and security testing of autonomous vehicles. By configuring application scenarios, attacks, and validation techniques, our framework is able to generate and validate numerous attacks on sensor data, static real-world data, and networking data.

Acknowledgment

This work has received funding by the European Unions Horizon 2020 Research and Innovation Programme through the nIoVe project (https://www.niove.eu/) under grant agreement no. 833742. 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] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.

[2] H. Winner, "Introducing autonomous driving: An overview of safety challenges and market introduction strategies," *at-Automatisierungstechnik*, vol. 66, no. 2, pp. 100–106, 2018.

[3] M.-F. Chang, J. W. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays, "Argoverse: 3d tracking and forecasting with rich maps," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[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*, 2017, pp. 1–16.

[5] G. Rong, B. H. Shin, H. Tabatabaee, Q. Lu, S. Lemke, M. Možeiko, E. Boise, G. Uhm, M. Gerow, S. Mehta *et al.*, "Lgsvl simulator: A high fidelity simulator for autonomous driving," *arXiv preprint arXiv:2005.03778*, 2020.

[6] E. Ribnick, S. Atev, O. Masoud, N. Papanikolopoulos, and R. Voyles, "Real-time detection of camera tampering," in *2006 IEEE International Conference on Video and Signal Based Surveillance*. IEEE, 2006, pp. 10–10.

[7] X. Zhang, A. Liniger, A. Sakai, and F. Borrelli, "Autonomous parking using optimization-based collision avoidance," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 4327–4332.

[8] T. Huang, J. Zhou, and A. Bytes, "Atg: An attack traffic generation tool for security testing of in-vehicle can bus," in *Proceedings of the 13th International Conference on Availability, Reliability and Security*, 2018, pp. 1–6.

[9] M. E. Verma, M. D. Iannacone, R. A. Bridges, S. C. Hollifield, B. Kay, and F. L. Combs, "Road: The real ornl automotive dynamometer controller area network intrusion detection dataset (with a comprehensive can ids dataset survey & guide)," *arXiv preprint arXiv:2012.14600*, 2020.

[10] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: an ensemble of autoencoders for online network intrusion detection," *arXiv preprint arXiv:1802.09089*, 2018.

[11] Q. He, X. Meng, R. Qu, and R. Xi, "Machine learning-based detection for cyber security attacks on connected and autonomous vehicles," *Mathematics*, vol. 8, no. 8, 2020.