ADAssure: Debugging Methodology for Autonomous Driving Control Algorithms

Andrew Roberts[◇][*], Mohammad Reza Heidari Iman[◇][†], Mauro Bellone[*], Tara Ghasempouri[†],
Jaan Raik[†], Olaf Maennel[‡], Mohammad Hamad[§], Sebastian Steinhorst[§]
[*] FinEst Centre for Smart Cities, Tallinn University of Technology
[†] Department of Computer Systems, Tallinn University of Technology
[‡]School of Computer and Mathematical Sciences, The University of Adelaide
[§]Department of Computer Engineering, Technical University of Munich

***Abstract*—Autonomous driving (AD) system designers need methods to efficiently debug vulnerabilities found in control algorithms. Existing methods lack alignment to the requirements of AD control designers to provide an analysis of the parameters of the AD system and how they are affected by cyber-attacks. We introduce ADAssure, a methodology for debugging AD control system algorithms that incorporates automated mechanisms which support generation of assertions to guide the AD system designer to identify vulnerabilities in the system. Our evaluation of ADAssure on a real-world AD vehicular system using diverse cyber-attacks developed a set of assertions that identified weaknesses in the OpenPlanner 2.5 AD planning algorithm and its constituent planning functions. Working with an AD control system designer and safety validation engineer, the results of ADAssure identified remediation of the AD control system, which can support the implementation of a redundant observer for data integrity checking and improvements to the planning algorithm. The adoption of ADAssure improves autonomous system design by providing a systematic approach to enhance safety and reliability through the identification and mitigation of vulnerabilities from corner cases.**

***Index Terms*—Security, Autonomous Driving**

I. Introduction

Autonomous driving (AD) vehicles are increasingly being utilised for transportation on public roads. Waymo and Cruise offer AD ride-hailing services in San Francisco, Apollo Baidu in China, and numerous such services are operating in Europe. Central to the wider-adoption of AD vehicles on public roads is the security and safety of their control algorithms that enable self-driving technology. AD control algorithms comprise a complex code-base of interconnected modules that perform tasks and sub-tasks that enable a vehicle to sense, perceive, localise, and navigate in a driving environment. With the increase in diversity of AD use-cases from valet parking to public transportation in public traffic, the code base of AD control algorithms will reputedly grow from 100-200 million to billions of lines of code [1].

Within this complex environment, debugging the code for logical errors arising from unexpected control behaviour is a fundamental challenge [2]. AD system designers need to pinpoint where in the control software weaknesses are, in order to focus debugging efforts in an efficient manner. Existing studies attempt to rectify unexpected AD control behaviour at run-time through smoothing trajectories utilising neural networks [3] [4] [5]. The applicability of these studies in real-world AD programs are limited due to the highly dynamic environment of autonomous driving and the probabilistic nature of the algorithms for planning.

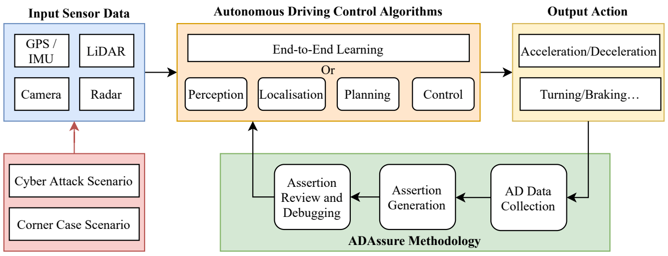

Fig. 1: Comprehensive ADAssure methodology overview that illustrates each step of the process, from data collection to assertion creation, review of assertions, and debugging.

Furthermore, in these studies, the analysis lacks the expertise from the algorithm designer and safety engineer to inform on the nature of the behaviour of vehicle dynamics, whether noise identified as irregular could be considered for a control engineer within normal constraints, whether AD behaviour could be considered a legitimate safety response to an unexpected event and whether the parameters for which the run-time solution is designed are appropriate for differing class of vehicles with different dynamic profiles. We consider the design phase to offer the most promising area of initial investigation to improve the robustness of control algorithms, which can be translated to real-world AD systems.

In this work, we propose ADAssure, a methodology for debugging control algorithms during the design-time phase of AD control software development (Fig. 1). ADAssure is built upon the idea that the data of vehicle dynamics and sensing of AD systems can be analysed for anomalous control behaviour, which can then be transformed into assertions on the AD control. We use association rules that enable us to mine datasets of varying scales and fingerprint the pattern of anomalous activity. These rules can be used to guide AD system designers to focus on the debugging of the control algorithms. To evaluate ADAssure, we focus on a control system algorithm used in a real-world AD vehicular system providing ride-hailing services. To summarise, the paper makes the following contributions:

- We propose ADAssure, a methodology designed for debugging AD control algorithms during the autonomous system development's design phase (see § II).
- To demonstrate our methodology's feasibility, we applied it to diverse datasets, revealing three new vulnerabilities in `Openplanner 2.5` AD, used in real-world vehicles. (see § IV).
- We provide to the community our artifacts to enable reproducibility and assist with developing efforts to im-

[◇] These authors contributed equally to this work.

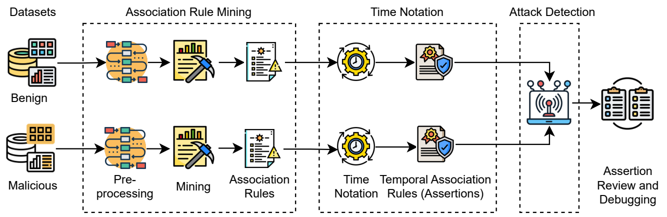

Fig. 2: Phases for Assertion Generation

prove AD control system design. These artifacts include simulation datasets and real-world AD system data (*ADAssure Datasets*).

II. ADAssure: Methodology

The development of ADAssure has three main motivations. First, it aims to provide AD system designers with a methodology to identify and fix vulnerabilities that align with the design of AD algorithms. Second, given the ever-changing nature of the autonomous vehicle system, it strives to establish a structured methodology that allows for consistent, flexible, and repeatable testing. Third, it aims to support unit testing, allowing testing of individual components of the autonomous system in isolation from other dynamic factors affecting autonomous control.

The foundations of the ADAssure methodology are based on the analysis of the vehicle dynamics and sensing data to guide the creation of assertions of the vulnerability of the AD control algorithms. The analysis consists of a sensitivity analysis of vehicle dynamics data (e.g., velocity, yaw, and steering angle), sensor data (e.g., lateral and longitudinal movement), and visualisation of the trajectory of the AD system. This helps identify key parameters to build assertions of the AD control algorithms. The AD control system designers can use the assertions to identify and locate the vulnerabilities of the control model and develop mechanisms to test and fix the errors. The ADAssure methodology comprises three main phases: AD Data Collection, Association Rule Generation, and Assertion Review and Debugging. Next, we will explore each phase in more depth.

A. Autonomous Driving Data Collection

This phase consists of generating data from the real-world system or simulation environment. The benefit of a simulation environment is that driving scenarios can be automated or designed to test a specific condition, such as a cyber-attack or a corner case. The data output is structured according to established metrics. These can be vehicle dynamics parameters (yaw angle, velocity, etc.), sensing data (position co-variance, point-cloud, etc.), and safety parameters (distance-to-collision, etc.). The AD data is outputted in a format that can be interpreted by analytical tools, in our use-case, .csv format.

B. Association Rule Generation Phase

The goal of this phase is to process the data generated from the previous phase and produce a set of association rules that can be translated into assertions in the Assertion Review and Debugging phase. This phase is comprised of three primary steps (as shown in Fig. 2): a) Association Rule Mining, b) Time Notation, and c) Attack Detection. The association rule mining is applied to both benign and malicious datasets, resulting in two distinct sets of association rules.

Algorithm 1: Association rule mining & time notation

```
1  Input: $\mathcal{N}$, $\mathcal{D}$
2  Output: $next[\mathcal{N}] = antecedent \rightarrow next[\mathcal{N}]consequent$,
     $before[\mathcal{N}] = antecedent \rightarrow before[\mathcal{N}]consequent$
   /* Initialization and Preprocessing */
3  $\mathcal{R}$ = $antecedent \rightarrow consequent$
4  forall $f \in \mathcal{D}$ do
5      $\mathcal{D}'$ = MoveUp($f(\mathcal{N})$)
   /* Mining */
6  $\mathcal{R} \leftarrow$ apriori($\mathcal{D}'$)
   /* Time Notation */
7  if ($\mathcal{R}.antecedent$ == ($t \in \mathcal{D}'$)) and ($\mathcal{R}.consequent$ ==
     ($f \in \mathcal{D}'$)) then
8      $next[\mathcal{N}] \leftarrow$ label($\mathcal{R}$)
9  if ($\mathcal{R}.antecedent$ == ($f \in \mathcal{D}'$)) and ($\mathcal{R}.consequent$ ==
     ($t \in \mathcal{D}'$)) then
10     $before[\mathcal{N}] \leftarrow$ label($\mathcal{R}$)
```

These rules are then processed through the Time Notation step to incorporate temporal information, yielding temporal association rules (assertions) in the form of $next[\mathcal{N}]$ and $before[\mathcal{N}]$ patterns. We define $next[\mathcal{N}]$ type of rule in the general form of $\mathcal{X} \rightarrow next[\mathcal{N}]\mathcal{Y}$. This rule indicates that when $\mathcal{X}$ occurs, after $\mathcal{N}$ time instants, $\mathcal{Y}$ will occur. $\mathcal{N}$ is a positive integer value. Moreover, we define $before[\mathcal{N}]$ rule in the general form of $\mathcal{X} \rightarrow before[\mathcal{N}]\mathcal{Y}$. This rule demonstrates that whenever $\mathcal{X}$ happens, $\mathcal{Y}$ should have occurred $\mathcal{N}$ time instants before that. The "Attack Detection" step compares these temporal association rules, ultimately detecting attacks and anomalies within the datasets. Subsequent sections provide a more in-depth discussion of each step.

a) Association Rule Mining: This step primarily serves two objectives: pre-processing the datasets and subsequently mining association rules from the preprocessed data. To mine the association rules, apriori algorithm [6] was adopted and enhanced to mine temporal rules capable of detecting attacks at various time instances during autonomous vehicle (AV) operation. Algorithm 1 presents the details of the Association Rule Mining and Time Notation steps. In this algorithm, $\mathcal{D}$ denotes the dataset and $\mathcal{D}'$ is the preprocessed dataset, while f and t represent the dataset's features and target values. To prepare the dataset for mining the $next[\mathcal{N}]$ and $before[\mathcal{N}]$ temporal patterns, all the features of the dataset are moved $\mathcal{N}$ records above its original position (Line 5). However, the target of the dataset remains as it is. Afterwards, the apriori algorithm is applied to the preprocessed dataset to mine a set of association rules. The output of this phase is a set of association rules in the general form of $antecedent \rightarrow consequent$ that are ready to be forwarded to the Time Notation step.

b) Time Notation: In this step, the method integrates the concept of time into the association rules generated in the association rule mining step, leading to a set of temporal association rules. The method determines to which temporal pattern ($next[\mathcal{N}]$ or $before[\mathcal{N}]$) each extracted rule belongs and subsequently assigns the corresponding time label to the rule. If the antecedent value matches a target value in the dataset, and the consequent value has already been moved to another record in the dataset, the rule is labelled as a $next$ temporal association rule (Line 8). Otherwise, if the antecedent of a rule mined in the association rule mining step matches a dataset feature that has already been moved to another record and the consequent of the rule matches the target value of the dataset, we label this rule as a $before$ temporal association rule (Line 10). The mined rules are in the forms of $antecedent \rightarrow next[\mathcal{N}]consequent$, and $antecedent \rightarrow before[\mathcal{N}]consequent$, serving as assertions for debugging the AD system.

c) Attack Detection: This step aims to identify rules indicating attacks on the AV. We assume that the sets of mined rules from the benign and malicious datasets should be similar under normal conditions, without any AV attacks. Any deviation between these rule sets signifies an anomaly in the autonomous vehicle. Per this assumption, the temporal association rules (assertions) mined during the time notation phase are classified into two sets. The first category comprises rules exclusively mined from the malicious dataset, lacking counterparts in the benign dataset. Any rule extracted solely from the malicious dataset, without a corresponding counterpart in the benign dataset, signifies an attack. These rules reveal abnormal behaviour in the malicious dataset, contrasting with different behaviour observed in the corresponding time instance of the benign dataset. Consequently, we classify these as attacks. The second category comprises similar rules mined from both benign and malicious datasets, but with different *minimum support* (`min_supp`) and *minimum confidence* (`min_conf`) values. The variations in these values indicate that, while the mined rules are similar, abnormal behaviours and anomalies exist between the datasets. The apriori algorithm employs these two metrics (i.e., `min_supp` and `min_conf`). The `min_supp` value is the threshold and a minimum value that is chosen by the expert to decide whether a rule occurs frequently in the dataset or not [7], [8]. The `min_conf` is the minimum value that is chosen by the expert and is an indication of how often a rule has been found to be true [6], [9]. Increasing the `min_supp` value results in fewer association rules that describe more general behaviour of the autonomous vehicle, while decreasing the `min_supp` value leads to rules covering rare behaviours (corner cases). Similarly, raising the `min_conf` value produces fewer but more valid rules. Valid rules refer to association rules that will not be violated with different attack scenarios like corner cases. These values in the ADAssure facilitate an effective attack detection process. The second category of rules aids the ADAssure in effectively identifying corner cases and the attacks that rarely occur on the AV. These rare attacks exhibit behaviour very similar to normal vehicle operation but are malicious and can lead to AV failure.

C. Assertion Review and Debugging

Within this phase, the association rules generated from the association rule mining are reviewed in conjunction with an analysis of the control behaviour and individual data parameters to develop assertions. Trajectory maps of the AD system and graphs, which demonstrate the sensitivity of the data parameters during benign and cyber-attack scenarios, are compared to the anomalous behavioural patterns detected by the association rule mining tool. Using expertise from the algorithm designer and safety validation engineer assists in understanding which parameters can uniquely demonstrate a vulnerability of an algorithm within the system. From developing an assertion on the system's vulnerability, the debugging effort focuses on a control flow analysis. As the assertion assists in pinpointing the specific module, the static analysis can focus on the control flow of the substituent functions within the module. As an example of the importance of this pinpointing, a local-planning module could have 15 diverse algorithms, and within these, each could have multiple different methods or functions. As the code of AD algorithms are differential equations, debugging can suggest optimisations that enable mitigation mechanisms against the identified vulnerabilities.

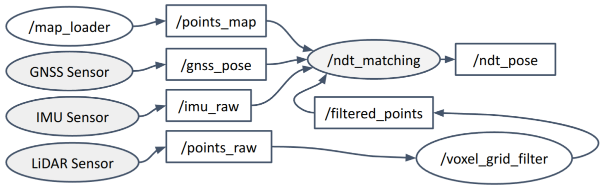

Fig. 3: Localisation Algorithm Flow within AD System.

III. Autonomous Driving Control Algorithm

To evaluate the methodology, we focus on an AD control algorithm used in a real-world AD ride-hailing service. Within the AD pipeline, there are four key modules: localisation, perception, planning, and control. Within our study, we focus on the localisation and planning modules.

A. Localisation Module

This module provides accurate information regarding the position and orientation of the vehicle. Using a Normal Distributions Transform (NDT) matching search algorithm, it identifies the best matching position based on sensor perception. It uses input from the Inertial measurement unit (IMU) and the point cloud generated by the LiDAR. Then, it attempts to match the points from our current scan to a grid of probability functions derived from the map. NDT matching algorithms can also benefit from the GNSS sensor, which provides initial rough estimates of localization on geo-referenced maps, thereby avoiding any sudden errors in localization calculations that may result in failures. Fig. 3 displays the flow of the localisation algorithm within the AD system.

B. Planning Module

For the AD system to plan a mission, firstly, a global planner generates a global reference path using a vector (road network) map. The function of the global planner is to stipulate a route between the starting position and goal position of the mission on the road map. The local-planner generates smooth and obstacle-free trajectories in the operational local domain following the global route. The local-planner consists of several modules (see Fig. 4); trajectory generation, trajectory evaluation, intention and trajectory estimator, object-tracker and behavior selection (decision making) [10]. The trajectory generation module generates alternative tracks parallel to the main path defined by the global planner. These tracks are named roll-outs. The trajectory evaluation module assesses all possible roll-outs and the data input from sensed-data of the AV and makes a cost estimation. The behaviour selector will lead the AV to motion on a roll-out based on the least-cost.

IV. Experimentation and Results

To evaluate the impact of corner cases on AD system behaviour using the ADAssure methodology, we use datasets of corner cases from simulation and real-world driving from

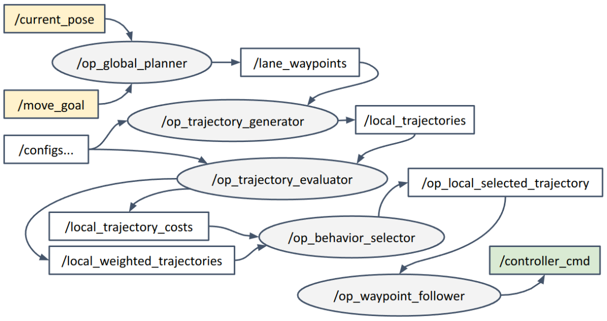

Fig. 4: Abstract Local Planning Algorithm Flow within AD System.

the target AD system. The 1st corner case scenario dataset is of three diverse cyber-security attacks on the AD system conducted in a simulation environment. As our focus is the planning and localisation algorithms, we used a low-fidelity simulation provided by `Autoware.AI` and the `OpenPlanner 2.5` planning algorithm. The 2nd corner case scenario dataset is of a Global Positioning System (GPS) spoofing event that occurred on the AD system during its operation on the roads of a capital city.

A. AD Control System Datasets

a) Cyber-security Corner Case Dataset: Within this dataset, three attacks were conducted on the target AD vehicular system, which is attempting an overtaking manoeuvre. The three attacks are classified as: 1) *Lateral Position Offset Attack* 2) *Longitudinal Position Offset Attack* 3) *Message Time-Delay*. In the lateral and longitudinal position offset attack, an attacker injects malicious data input into the lateral or longitudinal pose whilst the AD vehicular system is in the process of the overtaking manoeuvre (Fig. 5). This attack could be conducted through GPS spoofing or interception and manipulation of the localisation sensor data. The attacker introduces a delay into the `current_pose` (lateral and longitudinal) sensor messages reaching the AD control pipeline for the message time-delay. The malicious data is injected at around the 21 m mark of the AV journey (travelled distanced) to the 67 m. Each attack was conducted 300 times, accommodating a variation of different attack parameters. The lateral and longitudinal attacks introduced a deviation ranging from 0.16 % to 1.0 %, which equates to around 20 cm to 1 m. The message time-delay introduced delays of 0.3 %, 0.6 %, 1.0 % second, as a message is transmitted every 20 ms, this range represents a delay of 15 to 50 messages. In total, the dataset comprises over 1500 scenario runs of attacks and benign safety cases.

b) GPS Spoofing Real-World AV Dataset: The AD ride-hailing service transmits its sensor data via a logging node to an edge server, which stores the AD System data in a database. During its operations near the port area of the city, the AD vehicle encountered a loss of localisation from a GPS spoofing event which also affected other GPS-enabled platforms. This GPS spoofing continued intermittently throughout the preceding months. The dataset used in this study is from the logging system of AD ride-hailing service.

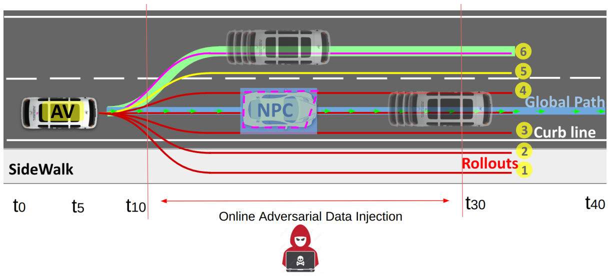

Fig. 5: The threat model used for conducting the attack cases.

TABLE I: AD System Data.

AD Data Type	Description
AV_X	Longitudinal Position of the AD System as to the HD Map
AV_Y	Lateral Position of the AD System as to the HD Map
AV_Steer	Steering Angle of the AD System
AV_Vel	Velocity of the AD System
AV_Yaw	Orientation of the AD System based on its centre of gravity
Roll-out_Num	Current Lane according to the lane selector of the AD Control Algorithm
DTC	Distance to collision of the AD vehicular system to the overtaking vehicle.
Position Co-variance	GPS position co-variance
Altitude	Altitude derived from the GPS

c) AD System Data: The simulation and real-world datasets were structured to output data as shown in Table I.

B. Experimental Results

To evaluate the ADAssure methodology, we chose six attack types and their corresponding safety (benign) scenarios. These attack types included each of the aforementioned attacks with differing levels of noise (lateral and longitudinal position offset, delay message).

a) Automated Analysis: Utilising the ADAssure methodology on the three types of attacks yields three distinct set of assertions corresponding to each attack type. The results of the assertion generation phase are presented in Table II. The threshold for minimum support (`min_supp`) is set at 0.01 , while the minimum confidence (`min_conf`) threshold is 1 Notably, the method exhibits a swift execution time. Within the 3 attacks of the cybersecurity corner case dataset, the assertions identify two patterns of anomalous AD behaviour. Firstly, extreme steering angles of 20° and -20° and sudden lane transition. Secondly, multiple lane-transitions combined with the extreme steering angle and sudden changes in vehicular velocity. This behaviour can be seen to be the effect of cyber activity on the smoothness of the initiation of the overtaking manoeuvre which results in turbulent movements and in some cases, a collision event. The assertions generated from the GNSS spoofing dataset identified the changes to the altitude and position co-variance. These were consistent with dramatic change in the values of the GPS coordinates and the resultant change in altitude.

b) Assertion Review and Debugging: The patterns identified in the association rules enables us to extrapolate that the Yaw angle and angular velocity are good reference point to show the effect of cyber-attacks. During the injection of the position offset attacks, the vehicle's orientation demonstrates dramatic action; in some circumstances, the vehicle can be

TABLE II: ADAssure Assertion Generation phase results.

Dataset		Assertion			Execution Time
Name	#Records	Total	$\#Next[\mathcal{N}]$	$\#Before[\mathcal{N}]$	
Longitude	412	5	3	2	1 ns
Latitude	356	7	7	0	1 ns
Delay	417	5	3	2	1 ns
GNSS	16	5	4	1	1 ns

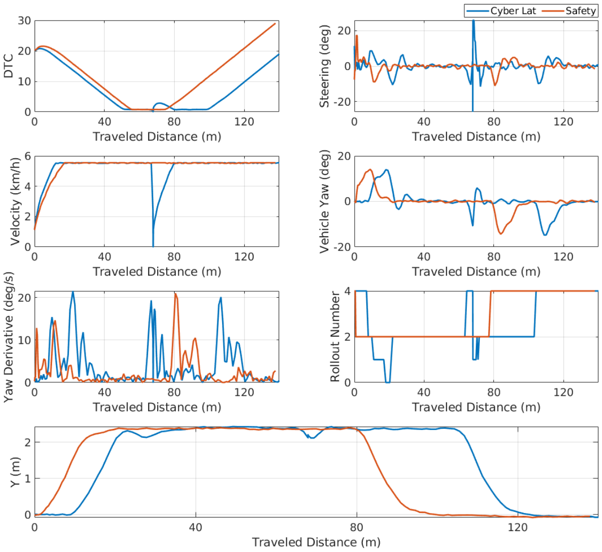

Fig. 6: Lateral position offset attack vehicle parameters.

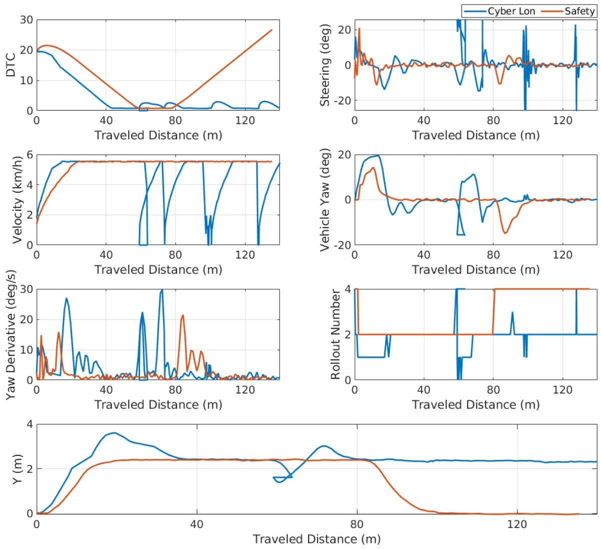

Fig. 7: Longitudinal position offset attack vehicle parameters.

seen to be essentially spinning. As displayed in Fig. 6, the Lateral Position Offset Attack displays the Yaw (angle) of the vehicle making sharp changes, of 15 deg/sec from 15 meters mark of the AV journey. This vehicle dynamic behaviour is a characteristic also seen in both the longitudinal position offset (Fig. 7) and delay message attack (Fig. 8). The results for the velocity parameter demonstrate that it only indicates immediate collision of the vehicle, and it does not support early identification of anomalous vehicle behaviour. Assertion 1 contends that the AD system should not allow movements that challenge the physical limitations of the steering model.

> **Assertion 1:** To determine the vulnerability of the yaw angle and momentum, we can derive the assertion: *AV.displacement_of_yaw_angle* $>$ *max_yaw_angle_threshold && time* $<$ *time_threshold.*

The roll-out transition, steer, and distance-to-collision parameters demonstrate identifiable change during a cyber-attack. The manipulation of the lateral and longitudinal position alters the vehicle position on the map and, therefore, has the effect of inducing greater transitions between roll-outs, which is the effective position of the vehicle on the road. The frequency of transition impacts the smoothness of the steering angle. From the distance-to-collision parameter, it is noted that the effect of the attack is most prominent during the overtaking maneuver and mostly during the cut-in process, when the vehicle cuts-in front of the passing vehicle (NPC). Assertion 2 contends that when the vehicle transitions across multiple roll-outs and displays 180° steering and closes to less than $0.5\,\mathrm{m}$ to the passing vehicle, this represents affected behaviour from the cyber attack.

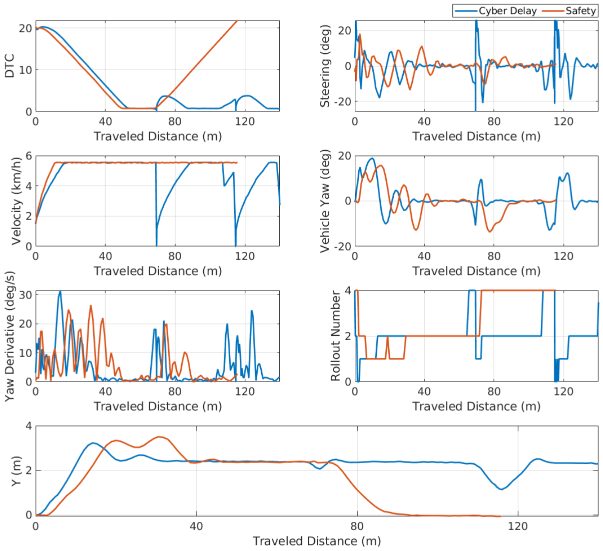

Fig. 8: Delay message attack vehicle parameters.

> **Assertion 2:** To identify vehicle dynamic changes from cyber-attack: *AV.x* $-$ *NPC.x* $<$ *distance_threshold && AV.lane_transition* $>$ *max_transition_number && AV.steer_angle* $\notin$ *[min, max]_steer_angle*

Assertion 3 contends with activity seen in the longitudinal position offset (Fig. 7) where the AV collides with the passing vehicle and then accelerates to the previous set-point.

> **Assertion 3:** To identify collisions we can derive the assertion: $|AV.v_k - AV.v_{k+1}| > threshold.$

Assertion 3 could also be used to detect anomalies in GPS data. The GNSS spoofing attack demonstrates a significant deviation in the altitude and position co-variance parameters. Assuming that velocity data comes from two sources, a wheel sensor measurement and calculated by deriving the position from GPS data, the two results should be close to each other. In the case of a GNSS spoofing attack, the deviation in the position co-variance would generate a spike in the velocity (calculated by deriving the position in GPS data), and thus violating assertion 3.

For our specific AD system, the threshold for assertion 1 is 15° yaw angle displacement within $1\,\mathrm{s}$ duration. Assertion 2 threshold is identified as a distance between AV and passing vehicle as less than $0.5\,\mathrm{m}$, lane transition greater than 1 roll-out and steering angle that is outside the bounds of 20 and $-20°$. It is important to note that these values are valid for a low-speed AV ride-hailing service and for designers of different classes of vehicles, it is required to calculate values consistent with their specific application.

Solvable bugs come from several points in the controller; a simple one is wrong or imprecise saturation values of the control signal, which generates a high acceleration or a high steering angle in the vehicle. This is clearly visible in Fig. 7 where a signal overshoot causes the vehicle to change lane multiple times. Another example, clearly visible in Fig. 6,7& 8 is the lack of a fallback plan. There is a clear indication of a collision as the vehicle speed suddenly drops to $0\,\mathrm{m\,s^{-1}}$ and then quickly accelerates to the reference point, this is a violation of Assertion 3. A robust controller should have a fallback plan for such a case which indicates a bug in the functional design of the controller. In such a case, the vehicle should be aware of the fact that the global trajectory cannot be followed anymore and switch to emergency mode.

The main reason for searching for unexpected behaviours is to debug the controller, with reference to the experimental results, a violation of Assertion 1 can be associated to a bug in the `/ndt_pose` module (see Fig. 3), while a violation of Assertion 2 can be back-propagated to the module `/op_trajectory_evaluator`. A violation of assertion three can be backpropagated to the modules of `/op_trajectory_generator` and `/op_behaviour_selector` (see Fig. 3). To pinpoint the violation of assertion 3 to a specific function, we abstracted from the local_planner algorithm and its substituent lane_rule algorithm, the `getClosestWaypointNumber` method, which selects the next waypoint to follow in the global trajectory and returned an exception to be handled as a different driving behaviour (e.g., there was a crash, emergency mode activated).

In the case of GNSS attack, the NDT localisation algorithm doesn't detect the deviation in position co-variance, and this is due to the normal vector pointing in the same direction. Debugging focuses on optimisation of the NDT localisation using visual odometry for holding the local position at short-distances until the source of the disturbance has been resolved.

V. Related Work

Recent publications on anomaly detection in vehicular AD control systems propose the usage of vehicle dynamics as a key detection indicator for cyber-attacks [11] [12] [13]. Studies such as Guo et al. [14] emphasise the effect cyber-attacks have on the trajectory of the AD system and the noise of individual sensors. Mitigation mechanisms focus on two diverse approaches 1) implementation of an observer of AD vehicle state estimation which can inform an emergency action (sensor switching etc.) [14] 2) implementation of trajectory smoothing algorithm to correct unplanned vehicle behaviour [12] [13]. However, these solutions for detection and mitigation are developed based on assumptions of driving environment and algorithm configuration and this limits the scope of their applicability.

VI. Conclusion

Cyber-attacks present new challenges to the design of AD algorithms. Designers need methods to debug vulnerabilities to improve robustness. In this paper, we introduced ADAssure, a methodology for debugging AD algorithms during the design phase. The methodology consisted of three phases; 1) AD Data collection 2) Assertion Rule Generation 3) Assertion Review and Debugging. The concept of the methodology was to develop association rules from mining AD data which can be transformed into assertions on the vulnerability of the system.

Our evaluation of ADAssure on diverse cyber-security datasets from simulation and real-world revealed that the ADAssure method could identify three assertions on the vulnerability of the `OpenPlanner 2.5` AD planning algorithm. These assertions were able to guide an algorithm designer and safety engineer to pinpoint the specific modules in the planning algorithm for debugging.

Acknowledgment

This work has been supported by the European Commission through the European Union's Horizon 2020 Research and Innovation Programme, under grant agreement No 101021727.

References

[1] Bosch, "Facts and figures about electronics and software in vehicles," *Automotive World*, July, 2021.

[2] W. Zeng, M. Wu, P. Chen, Z. Cao, and S. Xie, "Review of shared online hailing and autonomous taxi services," *Transportmetrica B: Transport Dynamics*, vol. 11, no. 1, pp. 486–509, 2023.

[3] K. K.-C. Chang, X. Liu, C.-W. Lin, C. Huang, and Q. Zhu, "A safety-guaranteed framework for neural-network-based planners in connected vehicles under communication disturbance," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2023.

[4] R. Jiao, H. Liang, T. Sato, J. Shen, Q. A. Chen, and Q. Zhu, "End-to-end uncertainty-based mitigation of adversarial attacks to automated lane centering," in *2021 IEEE Intelligent Vehicles Symposium (IV)*, 2021.

[5] X. Liu, R. Jiao, B. Zheng, D. Liang, and Q. Zhu, "Safety-driven interactive planning for neural network-based lane changing," in *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, ser. ASPDAC '23. Association for Computing Machinery, 2023.

[6] J. Han, M. Kamber, and J. Pei, "6 - mining frequent patterns, associations, and correlations: Basic concepts and methods," in *Data Mining (Third Edition)*, ser. The Morgan Kaufmann Series in Data Management Systems. Boston: Morgan Kaufmann, 2012, pp. 243–278.

[7] M. Zaki, "Scalable algorithms for association mining," *IEEE Transactions on Knowledge and Data Engineering*, 2000.

[8] M. R. Heidari Iman, J. Raik, M. Jenihhin, G. Jervan, and T. Ghasempouri, "An automated method for mining high-quality assertion sets," *Microprocessors and Microsystems*, vol. 97, p. 104773, 2023.

[9] M. Shahin, M. R. Heidari Iman, M. Kaushik, R. Sharma, T. Ghasempouri, and D. Draheim, "Exploring factors in a crossroad dataset using cluster-based association rule mining," *Procedia Computer Science*, 2022.

[10] H. Darweesh, E. Takeuchi, and K. Takeda, "Openplanner 2.0: The portable open source planner for autonomous driving applications," in *2021 IEEE Intelligent Vehicles Symposium Workshops*, 2021.

[11] Z. Ju, H. Zhang, X. Li, X. Chen, J. Han, and M. Yang, "A survey on attack detection and resilience for connected and automated vehicles: From vehicle dynamics and control perspective," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 4, pp. 815–837, 2022.

[12] Y. Ma, J. Sharp, R. Wang, E. Fernandes, and X. Zhu, "Sequential attacks on kalman filter-based forward collision warning systems," in *AAAI Conference on Artificial Intelligence*, 2020.

[13] J. Shen, Y. Luo, Z. Wan, and Q. A. Chen, "Lateral-direction localization attack in high-level autonomous driving: Domain-specific defense opportunity via lane detection," 2023.

[14] J. Guo, L. Li, J. Wang, and K. Li, "Cyber-physical system-based path tracking control of autonomous vehicles under cyber-attacks," *IEEE Transactions on Industrial Informatics*, 2023.