CAVELIER: Automated Security Evaluation for Connected Autonomous Vehicle Applications

Srivalli Boddupalli[1], Venkata Sai Gireesh Chamarthi[1], Chung-Wei Lin[2], and Sandip Ray[1]

***Abstract*— Connected Autonomous Vehicle (CAV) applications have shown the promise of transformative impact on road safety, transportation experience, and sustainability. However, they open large and complex attack surfaces: an adversary can corrupt sensory and communication inputs with catastrophic results. A key challenge in development of security solutions for CAV applications is the lack of effective infrastructure for evaluating such solutions. In this paper, we address the problem by designing an automated, flexible evaluation infrastructure for CAV security solutions. Our tool, CAVELIER, provides an extensible evaluation architecture for CAV security solutions against compromised communication and sensor channels. The tool can be customized for a variety of CAV applications and to target diverse usage models. We illustrate the framework with a number of case studies for security resiliency evaluation in Cooperative Adaptive Cruise Control (CACC).**

I. INTRODUCTION

The automotive industry has been witnessing rapid transformation in recent years, with explosive proliferation of advanced sensors systems and emergence of vehicular communications (V2X). These technologies hold the promise of enabling a variety of connected autonomous vehicle (CAV) applications that facilitate coordination among vehicles to achieve better road safety, fuel efficiency, and utilization of transportation infrastructure. However, one unfortunate consequence of the increasing dependence of vehicular function on sensory and communication inputs is the emergence of a large and complex attack surface: CAV applications are susceptible to adversarial attacks that spoof, jam, or manipulate these input channels. A crucial feature of attacks on these components is that it is not necessary for an attacker to hack into the hardware or software of the victim component: it is possible to create catastrophic impact simply by providing wrong or misleading sensory or communication inputs. Furthermore, the exploitable attack surface is only going to grow in future with the rapidly increasing complexity and sophistication of CAV applications.

There has been significant recent research on addressing this problem through a variety of resiliency solutions for CAV applications [3], [5], [6], [13]. These solutions have used a diversity of technologies, including machine learning, kinematics, control theory, game theory, etc. Obviously, *designing* an effective resiliency solution for CAV applications is non-trivial. However, another critical, non-trivial aspect is *evaluating* such a solution. Note that since CAV applications are safety-critical, the evaluation must be comprehensive and provide high assurance that a purportedly resilient CAV application performs safely under both benign and malicious environments. This is challenging, given the size and complexity of attack surfaces. Unfortunately, — and in spite of the critical need, — we have not found an effective infrastructure for comprehensive evaluation of resiliency of CAV applications. In absence of such infrastructure, every CAV security research has had to resort to developing its own security evaluation infrastructure from scratch. In addition to being tedious, complex, and time-consuming, this makes it extremely difficult to ensure that the evaluation indeed is comprehensive and meets the stringent requirements of safety-critical applications.

In this paper, we address this problem by developing a framework, CAVELIER (CAV Resiliency Evaluator) for comprehensive evaluation of CAV resiliency under compromised sensors and communication. CAVELIER can be easily customized for different CAV applications, adversary settings (*i.e.*, the abilities of the attacker, untrusted input channels), evaluation metrics, etc. Given this information, it can compute and orchestrate a set of attacks that comprehensively explore the attack space defined by the adversary. The output of CAVELIER is an evaluation report, including a rich-set of visualization results reflecting the effectiveness of the resiliency system under various attack scenarios.

The remainder of the paper is organized as follows. In Section II we provide a brief background on CAV resiliency systems and discuss related work. In Section III, we discuss CAV resiliency evaluation challenges. We introduce CAVELIER from a usage perspective in Section IV and the underlying backend architecture in Section V. In Section VI, we demonstrate representative CAVELIER application scenarios. We conclude in Section VII.

II. BACKGROUND AND RELATED WORK

A. Resiliency of Cooperative CAV Applications

Emergent autonomous vehicles are equipped with a variety of sensory and communication inputs to facilitate perception of their driving environments. The automotive sensor suite comprising radar, lidar, ultrasonic transceivers, cameras, etc. provides accurate situational awareness for the vehicles. V2X technologies such as DSRC enable communication among vehicles. CAVs exploit V2X to collaborate with other CAVs and transportation infrastructure to achieve improved safety

[1]Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611, USA. Email: bodsrivalli12@ufl.edu, vchamarthi@ufl.edu, sandip@ece.ufl.edu.

[2]Department of Computer Science and Information Engineering, National Taiwan University, Taipei 10617, Taiwan. Email: cwlin@csie.ntu.edu.tw.

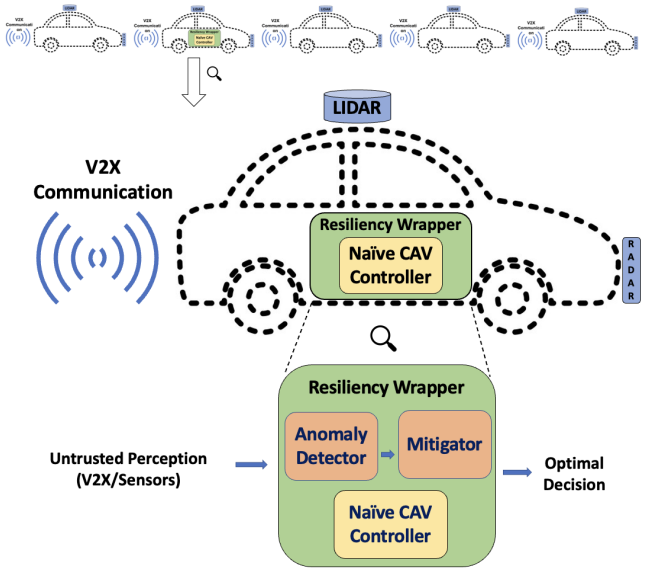


Fig. 1: Resiliency-augmented CAVs

and efficiency. Some popular CAV applications include Cooperative Adaptive Cruise Control (CACC) [2], multi-vehicle platooning [9], on-ramp merging [14], collision avoidance [15], cooperative lane changing [7], etc. Fig. 1 shows an abstracted view of a CAV application. Each participating CAV includes a controller that is responsible for computing real-time driving decisions during the course of the application engagement. The CAV "state" is captured with the help of variables such as *velocity, position, acceleration,* etc. and the controller action determines the state transitions. A *resilient* CAV application augments the controller with additional resiliency components. Typically, this entails real-time anomaly detection and mitigation systems. The detection system is responsible for vetting the untrusted perception inputs fed to the controller and the mitigation system overrides the naive controller output with an alternate decision if anomaly is detected.

B. Related Work

Research on platforms for exploration and analysis of CAV applications has primarily focused on various automotive simulators. SUMO [8], CARLA [4], VENTOS [1], and Veins [12] are some of the popular open-source desktop simulators. CARLA enables development, training, and validation of autonomous driving systems by supporting flexible specification of sensor suites, environmental conditions, and driving environments. SUMO is "microscopic" simulator: each vehicle is modelled explicitly to have its own route and move individually through the network. Various predefined car-following models can be used for each vehicle to adapt the speed based on the vehicles moving in front of them. VENTOS is built on top of SUMO and enables analysis of vehicular traffic flows, collaborative driving, and interactions between vehicles and infrastructure. Veins is also built on SUMO and provides a comprehensive suite of inter-vehicle communication models that can serve as a modular framework for simulating applications. It supports extensions such as PLEXE [11] for platoon management analysis. In addition to desktop simulators, there is a plethora of physical driving simulators. One such physical simulator is RDS1000® [10]. It enables detailed simulation and analysis of vehicular trajectory data in diverse driving environments. It provides a realistic physical interface for a human driver to control the vehicle in simulation and can also be operated in autonomous mode. Physical simulators have been traditionally used to study human interaction with the vehicle and driving environment. The fundamental limitation of most existing simulators in resiliency evaluation is the lack of well-defined feature set to support simulation of security attack scenarios. Veins supports some attack analysis, but it may not scale well to support comprehensive evaluation under a broad attack spectrum: the user would have to manually define each individual attack scenario under a desired adversary model.

III. Resiliency Evaluation Challenges and CAVeliEr Approach

Our design of CAVeliEr is heavily motivated by the challenges in CAV resiliency evaluation and inadequacy of current approaches. We briefly recount the challenges below. Note that the evaluation is carried out today by human security experts conceiving different ways of exercising various adversarial scenarios. Experts define various attack parameters such as specification of untrusted channels, duration, and mode of attack (*e.g.*, a mutation attack on V2X with a constant or sinusoidal bias, jamming attack on a sensory input, etc.), together with settings defining driving and environmental parameters for the targeted CAV application. This manual approach can lead to errors and incomplete understanding of the robustness of the resiliency solution under consideration.

1) Navigating large attack space: The number of possible attack instances on a CAV application can quickly become unmanageable. Consider a simple adversary corrupting a single V2X communication channel of a CAV. Possible attacks include (1) information mutation, (2) fabrication of fake information, or (3) prevention of information delivery (*e.g.*, through a jamming attack). The corruption can take place at different attack frequencies, the interference noise and the extent of mutation can take different forms and magnitudes. Evaluation must account for this large and complex space, which is difficult to achieve in manually designed resiliency approaches.

2) Evaluation under unknown attacks: Attack mechanisms are expected to get increasingly sophisticated with passage of time as CAVs become mainstream. On the other hand, it is not possible to simply develop new resiliency solutions to account for each newly discovered attack, particularly after deployment. A viable resiliency solution must account for a wide spectrum of attacks, whether known at the time of deployment or discovered later in field. This presents a vexing challenge to evaluation: how to ensure that

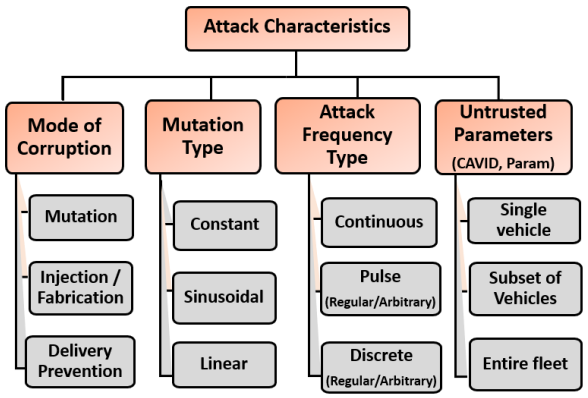


Fig. 2: Taxonomy of Attack Characteristics for Adversaries Against Vehicular Perception

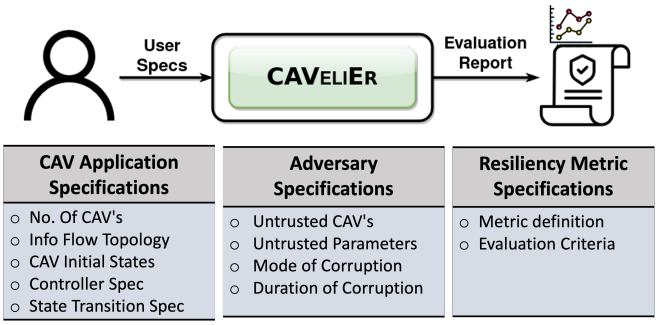


Fig. 3: CAVELIER Usage Model and User Specifications

a resiliency solution is indeed robust against as yet unknown attacks (in addition to the known ones).

3) Inflexibility of simulation platforms: As discussed in Section II-B, existing automotive simulators are not targeted towards resiliency evaluation. In particular, they do not provide the required configurability to easily create different attack scenarios and switch between them during analysis.

CAVELIER addresses these challenges as follows. To achieve comprehensive attack space coverage, it adopts a unique approach to characterize attacks with a curated set of features that account for the "effects" of an attack rather than the attack mechanism. The key observation is that irrespective of the attack mechanism, the effects can be captured through a small number of characteristics. For instance, if the adversary is confined to V2X messages, the attack mechanisms may be diverse and complicated (*e.g.*, signal jamming, spoofing, electronic interference, manipulation of delivery software, etc.), but in terms of effect the possibilities are (1) mutation of an existing message, (2) fabrication of a new message, and (3) prevention of the delivery of a message [3]. Note that this observation is independent of the underlying CAV application and only based on the definition of the untrusted input channel. Fig. 2 shows a comprehensive taxonomy of attacks defined based on this observation. It enables CAVELIER to use the threat model of the application for automatically and systematically generating a comprehensive set of representative attacks covering the adversarial space. Furthermore, CAVELIER provides significant flexibility and configurability to the user in specifying application parameters and adversary settings as well as the granularity of the evaluation as described in the subsequent sections.

IV. CAVELIER FROM USER VIEW

From a usage perspective, CAVELIER is a platform that enables comprehensive evaluation of cooperative CAV applications under a given perception adversary. The user specifies the CAV application and the adversary model through a standard template provided by the tool. Fig. 3 shows the usage model and the list of user specifications provided through the standard template. Additionally, the user can specify a desired resiliency metric (*e.g.*, safety or efficiency metric in terms of time-headway between vehicles, etc). CAVELIER generates attack environments specifying how to corrupt the perception information shared within the CAV fleet. It analyzes the resiliency of all the participating CAVs by modelling the application behavior and computing the resiliency metric under each attack environment. Ultimately, it generates a detailed evaluation report together with a set of visualizations.

CAVELIER provides two unique features for effective resiliency evaluation.

1) Support for arbitrary applications: CAVELIER lets the user specify CAV applications with: (i) arbitrary number of participating CAVs, (ii) both homogeneous (all CAVs having the same controller) and heterogeneous (each CAV may having a different controller) settings, (iii) different information flow topologies (or the communication scheme), and (iv) different types of applications (car-following, route management, lane changing, etc). The backend of the tool is agnostic to the internal working of the controller.

2) Support for arbitrary perception adversaries: CAVELIER supports adversaries of varying scope, *i.e.*, the user can provide a coarse-grained description of an adversary with a broad scope, or a highly specific adversary model that only accounts for a single attack instance (or anywhere in between). In case of a coarsely defined adversary, the user can choose to leave various attack characteristics unspecified. The tool generates all environments within the scope by taking into account the entire range of possible alternatives for the unspecified attack characteristics. On the other hand, the user can also select a very specific list of attack characteristics to be included under the adversary and the tool generates environments within those constraints.

CAVELIER frontend includes an interface template for various application and adversary parameters which can be populated by the user. The template comprises both essential and non-essential (optional) parameters. Essential parameters for describing the application include: (i) the number of participating CAVs, (ii) the controller behavior and state transition methods for each CAV, and (iii) the information flow topology (the communication scheme that describes

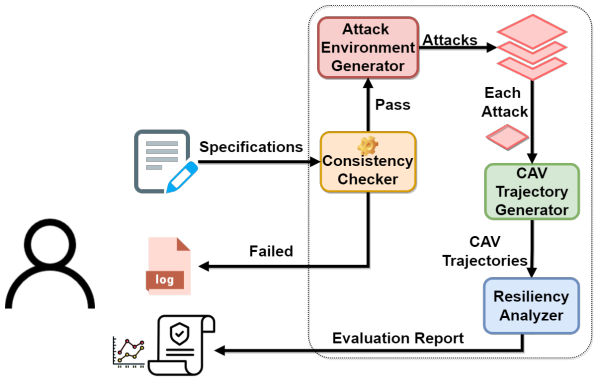


Fig. 4: CAVELIER Backend Architecture

how the communication takes place within the fleet). Essential specifications for adversary include the list of untrusted channels. All the non-essential parameters left unspecified by the user are typically assumed by default as directive to consider evaluation under all possible parameter values, *e.g.*, if mode of corruption is left unspecified the tool must account all the supported modes (mutation, fabrication, and delivery prevention). On the other hand, a user interested in specific attack type can focus on that through use of the non-essential parameters. The tool also caters to users desiring fine-grained control over attack generation: such users can specify low-level configurations of "mutation type" and "frequency type" choices, *e.g.*, width of pulse attacks, duration of continuous attacks, sinusoidal mutation time period, etc.

V. CAVELIER BACKEND ARCHITECTURE

CAVELIER architecture is designed with the goal of flexibility, modularity, and a high degree of configurability as well as feature extension. The tool design includes four modules as shown in Fig. 4: (i) Specification Consistency Checker, (ii) Attack Environment Generator, (iii) CAV Trajectory Generator, and (iv) Resiliency Analyzer.

A. Specification Consistency Checker

Specification Consistency Checker takes the standard interface template populated with user-defined specifications of application, adversary, and evaluation criteria, and performs a series of analyses to determine compatibility of the specification constraints. A key check is whether the adversary is realizable under the application constraints and communication constraints, *e.g.*, an attack involving a corrupt vehicle $\mathcal{C}$ corrupting a parameter p would not result in a viable adversary if p is never communicated to any vehicle.

B. Attack Environment Generator

Attack Environment Generator is responsible for interpreting the user-specified adversary model and generating the representative attack environments. The attack environment defines the characteristics of a specific attack describing how the reported information should be corrupted. Attack environments are generated by traversing the branches of Fig. 2 in a hierarchical manner and creating all possible combinations of attack characteristics constrained by the user-specified adversary model. Attack environments are grouped into different sub-categories based on the "number of corrupt paramters". As per this value, combinations of untrusted parameters are selected to be corrupted under each attack. For each corrupt parameter, the mode of attack indicates if the reported parameter should be (i) mutated from ground truth (by adding any one of the 4 different types of offsets/biases as listed in Fig. 2), (ii) replaced with a fabricated value (a random value independent of the ground truth), or (iii) prevented from being delivered to other CAVs. Under mutation attacks, for any offset type, the value of corruption is selected from a range of offset values corresponding to the untrusted parameter. The tool allows the user to specify a range of offset values and also the count to select as many equally distributed individual values from the continuous range. As the impact of attack is highly dependent on the mutation offset, this flexibility enables the user to select a subset of attacks that are impactful. Attack frequency types indicate how frequently the corruption should take place.

C. CAV Trajectory Generator

CAV Trajectory Generator is responsible for modelling the user specified CAV application and invoking it under each attack environment. It generates the "CAV trajectories" or the state progressions over time, based on the user-defined controller equations for each CAV for the duration of the test. At each time instance, CAV controllers are invoked to compute decisions based on the reported states of other CAVs in the fleet. The reported states reflect the ground truth (or the "actual") states of CAVs under benign conditions. Under attack environments, the reported states deviate from actual and are corrupted as per the attack characteristics (*e.g.*, Reported "Vel" of CAV_0 is mutated by adding a constant offset of $0.5ms^{-1}$ to the ground truth). The communication scheme or the information flow topology of the user-defined application determines which vehicles in the fleet receive the corrupted perception information under each attack. The computed decisions are applied to the CAVs to update their state. The resultant states are appended to the CAV trajectories. After generating CAV trajectories under all the attack environments, it invokes Resiliency Analyzer.

D. Resiliency Analyzer

Resiliency Analyzer probes the CAV trajectories under each attack environment and computes the user-defined resiliency metric. It checks if the resiliency metric falls within the ideal range indicating the resiliency objective(s) being met. It generates an evaluation report collating all the information about the attack environments and the corresponding resiliency status. It embeds a visual representation of statistics of the resiliency metric distribution over different sub-categories of attacks under the adversary. In addition to the evaluation report, it also generates a fine-grained set of auxiliary figures showing the attack orchestration, resultant state progressions or trajectories of CAVs under

			SCENARIO 1	SCENARIO 2	SCENARIO 3
USER SPECIFICATIONS [* Essential]	CAV Application Specifications	*CAV's count	2 [IDs: CAV_0, CAV_1]		
		*CAV parameters	["Acc", "Vel", "Pos"]		
		*Information flow topology	[[1,1], [0,1]]		
		*Controller model, state update procedure	As per Ammozadeh et.al.,		
	Adversary Model Specifications	*Untrusted parameters list	["CAV_0_Acc", "CAV_0_Vel", "CAV_0_Pos"]	["CAV_0_Vel", "CAV_0_Pos"]	["CAV_0_Acc"]
		Desired #of corrupt parameters	Unspecified	[1,2]	[1]
		Mode of corruption	Unspecified	["Mutation", "Fabrication"]	["Mutation"]
		Attack frequency types	Unspecified	["Continuous", "Regular Pulse"]	["Continuous"]
		Mutation type	Unspecified	["Constant", "Linear"]	["Constant"]
		Desired mutations count	Unspecified	3	1
	Resiliency Metric Specifications	*Metric	Mean Time Headway (THW)		
		*Evaluation criteria	THW in [0.55, 0.75]		
ATTACKS SUMMARY	Total attacks generated		1,030,730	176	1
	Single corrupt attacks count		330	28	1
	Two corrupt attacks count		30150	148	-
	Three corrupt attacks count		1000250	-	-

Fig. 5: Example Test Scenarios Evaluated by CAVELIER

each attack, in comparison to the benign scenario. While the report shows cumulative resiliency analysis over sub-categories of attacks, these figures give the user a deeper perspective into the resiliency of the controller if desired. The tool lets the user specify whether the fine-grained attack-specific visualizations should be generated during evaluation.

VI. CAVELIER IMPLEMENTATION AND CASE STUDY

We implemented CAVELIER in Python. Each component of the backend is implemented independently in a modular fashion. A central program coordinates the flow of operation. A standard user-specification template is created as a collection of 3 classes encapsulating variables and methods that guide the user to specify the application, adversary, and the resiliency evaluation criteria.

Although CAVELIER is independent of the underlying CAV implementation, it is illuminating to observe its use on a simple, illustrative case study. For this purpose, we use the resilient controller developed in prior work [3] for CACC. We choose this because of our high familiarity with this application, that helps us illustrate various facets of the resiliency evaluation performed by CAVELIER. In CACC, a follower CAV adapts its acceleration in accordance with the relative velocity, gap, and acceleration of the vehicle in front. The resilient controller, RACCON, extends a specific CACC implementation [2] with Machine Learning components to protect against adversaries that corrupt the preceding acceleration data. In Fig. 5, we consider three evaluation scenarios for RACCON, and the attacks generated by CAVELIER. In Scenario 1, the adversary is loosely defined with the user specifying only the essential parameters. Scenarios 2 and 3 progressively constrain the adversary, *e.g.*, Scenario 3 permits a single, specific attack instance. Fig. 6 shows the Consistency Checker log file (for passing and failing checks). Note that if the check fails, the log points to missing parameters and specification inconsistencies as shown in Fig. 6(a).

Evaluation Report: Fig. 7 shows the evaluation report automatically generated by CAVELIER under Scenario 2. The report shows: (i) summary of attacks orchestrated under different categories, (ii) resiliency analysis summary indicating the observed trend in resiliency metric with respect to the user-specified evaluation criteria, and (iii) (paths directing to) the auxiliary visualizations folders. The report is generated after the trajectories under all the attack environments are orchestrated. The resiliency system in this case met the cumulative resiliency criteria under all the attack sub-categories (*i.e.*, mean time headway (THW) is well within the range specified by the user) and has been assigned a "Pass" status by CAVELIER.

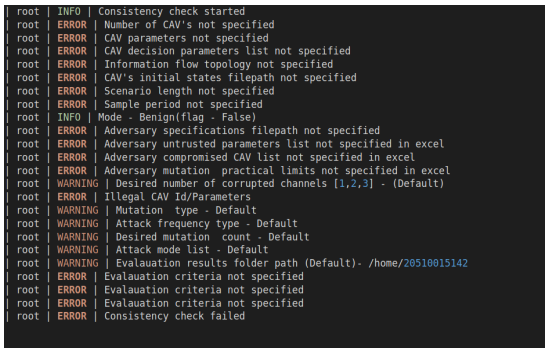
The case study reflects the flexibility of CAVELIER with respect to the adversarial capabilities. Furthermore, note that for a loosely defined adversary specification (as required to ensure comprehensive evaluation of the entire attack space), the number of attack instances can be exorbitant, *e.g.*, for Scenario 1, over 1 million attack scenarios are generated by the tool even for this simple application as shown in Fig. 3. This points to the inadequacy of current manual approaches to perform this evaluation and the critical need for an automated infrastructure like CAVELIER.

VII. CONCLUSION AND FUTURE WORK

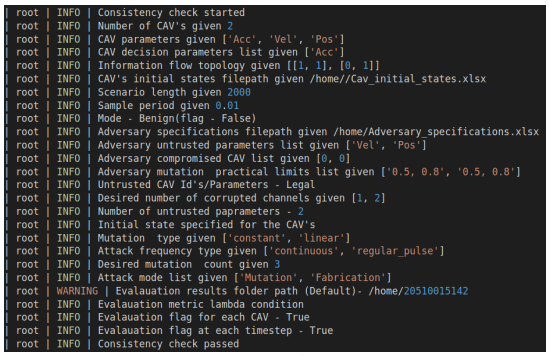
We have presented a tool, CAVELIER, for evaluation of resiliency in CAV applications against adversaries targeting perception inputs such as sensors and communications. CAVELIER can be easily customized to cater to different applications, configure application parameters, explore different adversaries, etc. Consequently, it ameliorates the complexities of validation for the CAV security designer, letting them focus on the creative task of *designing* effective solutions. It also provides a standard infrastructure for certification authorities to evaluate CAV resiliency. We showed case studies to illustrate various facets of CAVELIER.

In future work, we will perform more case studies on CAVELIER, and improve its computational efficiency. Note that in a practical CAV application with a "loose" parameter setting, CAVELIER can generate millions or even billions of attack scenarios. Executing them on a realistic CAV application and developing summary values for metrics can be computationally prohibitive. We will explore design and implementation optimizations to make such analysis viable.

Acknowledgements: This research has been supported in part by the National Science Foundation under Grant No. CNS-1908549, and by Ministry of Education (MOE) and Ministry of Science and Technology (MOST) in Taiwan under Grant Numbers NTU-110V0901 and MOST-111-2636-E-002-018.



(a)



(b)

Fig. 6: Log files showing Consistency Checker in Action (a) Error Messages under Failed Consistency Check (b) Specification Summary under Successful Consistency Check

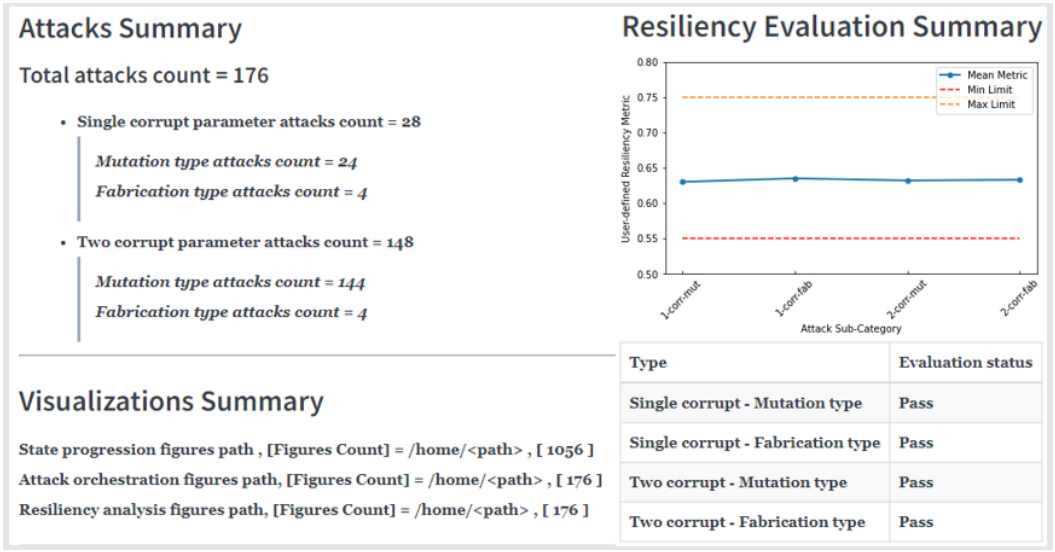


Fig. 7: Automated Resiliency Evaluation Report Generated by CAVELIER for Scenario 2

REFERENCES

[1] M. Amoozadeh, B. Ching, C.-N. Chuah, D. Ghosal, and H. M. Zhang. Ventos: Vehicular network open simulator with hardware-in-the-loop support. *Procedia Computer Science*, 151:61–68, 2019.

[2] M. Amoozadeh et al. Platoon management with cooperative adaptive cruise control enabled by vanet. *Vehicular Communications*, 2015.

[3] S. Boddupalli, A. S. Rao, and S. Ray. Resilient cooperative adaptive cruise control for autonomous vehicles using machine learning. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[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*, pages 1–16, 2017.

[5] R. G. Dutta, F. Yu, T. Zhang, Y. Hu, and Y. Jin. Security for safety: A path toward building trusted autonomous vehicles. In *ICCAD*, 2018.

[6] X. Jin, W. M. Haddad, Z.-P. Jiang, and K. G. Vamvoudakis. Adaptive control for mitigating sensor and actuator attacks in connected autonomous vehicle platoons. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 2810–2815. IEEE, 2018.

[7] U. Khan, P. Basaras, L. Schmidt-Thieme, A. Nanopoulos, and D. Katsaros. Analyzing cooperative lane change models for connected vehicles. In *2014 International Conference on Connected Vehicles and Expo (ICCVE)*, pages 565–570. IEEE, 2014.

[8] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*, pages 2575–2582. IEEE, November 2018.

[9] R. Rajamani, H.-S. Tan, B. K. Law, and W.-B. Zhang. Demonstration of integrated longitudinal and lateral control for the operation of automated vehicles in platoons. *IEEE Transactions on Control Systems Technology*, 8(4):695–708, 2000.

[10] Realtime-Technologies. Physical Automotive Simulator. See URL: `https://www.faac.com/realtime-technologies/products/rds-1000-single-seat-simulator`.

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

[12] C. Sommer, R. German, and F. Dressler. Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. *IEEE Transactions on Mobile Computing (TMC)*, 10(1):3–15, January 2011.

[13] G. Sun, T. Alpcan, B. I. Rubinstein, and S. Camtepe. Strategic mitigation against wireless attacks on autonomous platoons. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 69–84. Springer, 2021.

[14] P. Wang and C.-Y. Chan. Formulation of deep reinforcement learning architecture toward autonomous driving for on-ramp merge. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2017.

[15] J. Wen, C. Wu, R. Zhang, X. Xiao, N. Nv, and Y. Shi. Rear-end collision warning of connected automated vehicles based on a novel stochastic local multivehicle optimal velocity model. *Accident Analysis & Prevention*, 148:105800, 2020.