IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 19 November 2022, accepted 17 December 2022, date of publication 30 December 2022, date of current version 10 January 2023.

Digital Object Identifier 10.1109/ACCESS.2022.3233424

RESEARCH ARTICLE

Toward Attack Modeling Technique Addressing Resilience in Self-Driving Car

JUNAID M. QURASHI, KAMAL JAMBI, FATHY E. EASSA, MAHER KHEMAKHEM, FAWAZ ALSOLAMI, AND ABDULLAH BASUHAIL
Department of Computer Science, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia
Corresponding author: Junaid M. Qurashi (qurashi.junaid@gmail.com)

This work was supported by the Deanship of Scientific Research (DSR), King Abdulaziz University, Jeddah, under Grant RG-8-611-40.

ABSTRACT Self-driving cars are going to be the main future mode of transportation. However, such systems like, any other cyber-physical system, are vulnerable to attack vectors and uncertainties. As a response, resilience-based approaches are being developed. However, the approaches lack a sound attack model that recognizes the attack vectors and vulnerabilities such a system would have and that does a proper severity analysis of such attacks. Moreover, the existing attack models are too generic. Currently, the domain lacks such specific work pertaining to self-driving cars. Given the technology and architecture of self-driving cars, the field requires a domain-specific attack model. This paper gives a review of the attack models and proposes a domain-specific attack model for self-driving cars. The proposed attack model, severity-based analytical attack model for resilience (SAAMR), provides attack analysis based on existing models. Also, a domain-based severity score for attacks is calculated. Further, the attacks are classified using the decision-tree method and predictions of the type of attacks are given using long short-term memory network.

INDEX TERMS Attack-model, autonomous vehicles, cyber-attacks, resilience, security, self-driving car.

NOMENCLATURE

AT	Adversarial attack tree.
CAN	Controller area network.
CT	Code modification/injection tree.
CVE	Common vulnerabilities and exposures.
CVSS	Common vulnerability scoring system.
CWE	Common weakness enumeration.
DATMO	Detection and tracking of moving objects.
DDoS	Distributed denial of service.
DoS	Denial of service.
DT	Decision tree.
ECU	Electronic controller unit.
GPS	Global positioning system.
InV	In-vehicle.
ITS	Intelligent transportation system.
JT	Jamming attack tree.
LiDAR	Light detection and ranging.
LINDDUN	Linkability, identifiability, non-repudiation, detectability, disclosure of information, unawareness, non-compliance.
LSTM	Long short-term memory.
MAE	Mean absolute error.
NIST	National institute of standards and technology.
OBD	On-board diagnostics port.
OCTAVE	Operationally critical threat asset and vulnerability evaluation.
ODD	Operational design domain.
PASTA	Process for attack simulation and threat analysis.
PIER	Probability, impact, exposure, and recovery.
RPN	Risk priority number.
RSU	Roadside unit.
RT	Replay/relay attack tree.
SAAMR	Severity based analytical attack model for resilience.
SAE	Society of automotive engineering.
SAVTA	Software asset vulnerability threat and attacker.
SDC	Self-driving car.
SLAM	Simultaneous localization and mapping.
ST	Spoofing attack tree.

The associate editor coordinating the review of this manuscript and approving it for publication was Payman Dehghanian.

STAMP	Systems-theoretic accident model and processes.
STPA	STAMP-based hazard analysis.
STRIDE	Spoofing, identity, tampering with data, repudiation, information disclosure, denial of service, and elevation of privilege.
V2I	Vehicle-to-infrastructure.
V2V	Vehicle-to-vehicle.

I. INTRODUCTION

The popularity of the concept and attempts at achieving autonomous vehicles cannot be realized unless a proper analysis of the threats posed to the related cyber-physical systems has been done. According to the Society of Automotive Engineers (SAE) [1], a self-driving car (SDC) can be defined according to six degrees of automation characterized by the level of attention and intervention needed by the human drivers while driving or being driven by such vehicles. These levels vary from Level 0 (where the autonomy of the car is limited to issuing warnings without continuous sustained automated control), to Level 5 (where no human intervention is needed in any circumstance and ensures full automation even in a fallback scenario). For more details on the levels of automation in SDC, one may refer to the report by SAE in [1]. SDC, likely a popular mode of transportation in the near future, is still posed with hurdles that should be overcome. Currently, considerable efforts have been made towards realizing Level 5 SDC; the goal, however, is still far-fetched owing to the reliability and safety required in such systems. According to experts, it took 80 years to develop the SDC technology currently available, which represents 80% of the work. Perhaps it will take another 40 years to achieve the remaining 20% of the work, mostly owing to the challenges in maintaining the desirable functionality while ensuring safety and reliability under uncertain circumstances [2]. To achieve Level 5 SDC, it is imperative to identify the vulnerabilities using specific attack models before designing a resilient architecture that withstands malicious attacks and hence the uncertainties associated with it. Inherently, the architecture and design of the SDC pose several threats, as several sensors are meant to collect data from external surroundings, which determines the course of action of the actuators through connected controllers. The result is a physical response through attached actuators. Further, SDC vendors are increasingly employing deep-learning algorithms for obstacle recognition and avoidance. Most of these algorithms work in black box settings, leaving much scope for uncertainty. These challenges have restricted the deployment of SDC to the Operational Design Domain (ODD) [1], that is, the vehicle can operate only under a controlled environment restricted by predefined constraints.

Thus, in SDC architecture, the information generated through the interaction of several components is meant to result in an action that is safe, reliable, and devoid of any uncertainty. Building such architecture would mean a thorough understanding and anticipation of threats being considered while designing such systems. Several works have been published in trying to identify and demonstrate the attacks that could be devised against autonomous systems in general and, hence, the vulnerabilities. Although a good amount of research is dedicated to exploring vulnerabilities in autonomous vehicles, the studies lack a specific focus on attack modeling of SDC. Many attack models have been developed; however, not all of them are comprehensive, and their focus is more on a wide variety of subjects with organizational, people, and system orientations mostly aimed at giving general holistic solutions [3].

The main contributions of this paper are identifying the attack and accidental factors in SDC architecture and proposing an attack model that gives a severity analysis of the attacks in the context of SDC architecture, and hence aids in developing resilient architecture. This paper contributes by:

1) Identifying attacks, attack surfaces, and vulnerabilities in SDC architecture and detailing all the steps under the attack modeling process.
2) Providing an accident analysis of the SDC architecture.
3) Giving a severity analysis of the attacks identified.
4) Outlining a classification and prediction of the attacks identified using decision-tree algorithm and long short-term memory network.

The comprehensive overview of the paper is represented through the illustration in the Fig. 1. The structure of the paper can be described as follows: Section II gives a brief review of the literature on the threat models and discusses the reviewed literature. Section III gives a detailed description of the proposed severity-based analytical attack model for resilience (SAAMR) process. Section IV discusses the proposed model, its limitations, and future directions. The discussion also includes a descriptive analysis of the attacks. Section VI concludes the paper.

II. RELATED WORKS

This section gives a brief review of the attack modeling definition and related works. Attack modeling incorporates several key concepts related to attacks and may include concepts like a threat, attack, attack vector, attacker, and attack-surface. All these terms are defined according to the context for which the attack model is being designed. The definitions do overlap across related domains depending on the context and the assumptions made about the system or organization [4]. Threats in the context can be defined as any possible opportunity for an adversary that affects the operations of an SDC gained through intercepting information, false injection, etc. [5]. An attack is an attempt to realize threats for adversarial motives and is defined by the National Institute of Standards and Technology (NIST) [5] as ‘‘any kind of malicious activity that attempts to collect, disrupt, deny, degrade, or destroy information system resources or the information itself’’. The attack model shows how the attacker attains their goal by using different methods to launch a specific attack and, hence, reveals the threat and vulnerability of the

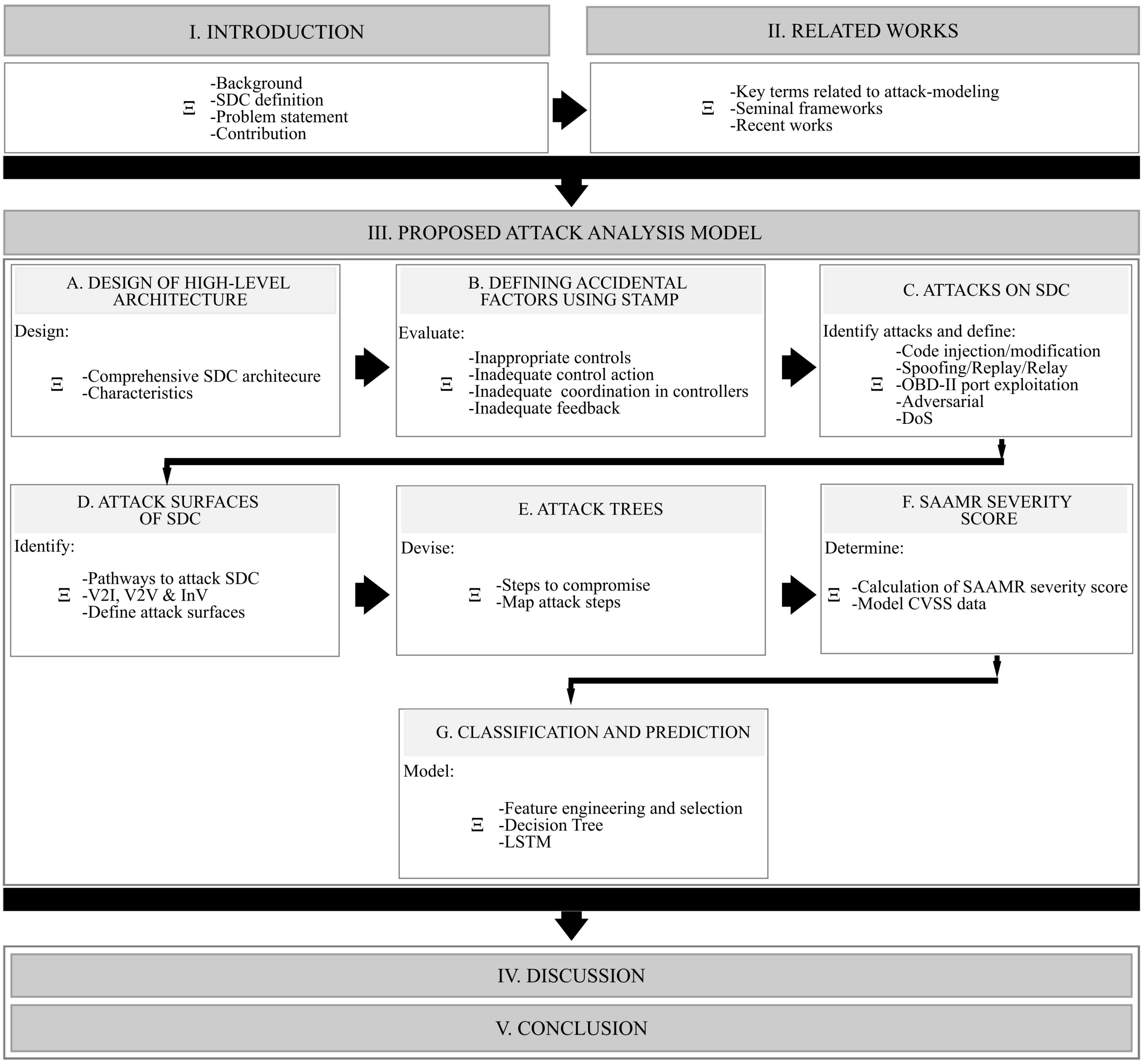


FIGURE 1. Roadmap of the paper.

system [6]. In this purview, several definitions of attack modeling also have been proposed. Uzunov and Fernandez [7] defined attack modeling as a process, "that can be used to analyze potential attacks or threats and can also be supported by threat libraries or attack taxonomies". Dahbul et al. [8] defined attack modeling as a process to, "analyze the security and vulnerabilities of an application or network services". The authors in [9] and [10] have defined attack modeling as an activity such that, "the architecture of the system is represented and analyzed, potential security threats are identified, and appropriate mitigation techniques are selected". For this study, the authors adhere to the definition of [7] and [10] and define attack modeling as a process, *"that can be used to analyze potential attacks through supported attack taxonomies"*.

In the context, some of the popular models that map attacks or faults include Spoofing identity, Tampering with data, Repudiation, Information disclosure, Denial of service, and Elevation of privilege (STRIDE) [11]; Systems-Theoretic Accident Model and Processes (STAMP) [12]; Process for Attack Simulation and Threat Analysis (PASTA) [13]; Linkability, Identifiability, Non-Repudiation, Detectability, Disclosure of Information, Unawareness, Non-Compliance (LINDDUN) [14]; Common Vulnerability Scoring System (CVSS) [15]; Trike [16]; Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) [17]; and Attack

Trees [18]. Brief detail and limitations of the aforementioned models are discussed in the following paragraph.

STRIDE [11], a mnemonic, was one of the earliest methods developed by Microsoft. It has matured over time and has laid the groundwork for the development of better attack analysis tools. Tuned to evaluating a system design in detail, STRIDE mostly relies on manual processes of creating dataflow diagrams. Limitations of the 'vulnerability explosion' are known, meaning that the number of threats can grow quickly as the complexity of the system increases. Nevertheless, several modified versions of STRIDE have been proposed [19], [20]. PASTA [13] is defined by seven stages that start from objectives related to business through to the technical requirements. Several stages define the exhaustiveness of the approach in aligning the organization's goals to the technical requirements while addressing the issues related to security, giving detailed steps on threat analysis, vulnerability analysis, Attack Trees defining attack modeling, and finally risk assessment and mitigation strategies. The limitations have been attributed to being too laborious of a task and its limited focus on attack modeling. LINDDUN [14] is focused on privacy threats and concerns. Similarly, CVSS [15] gives a severity-based score to the vulnerability. However, although there is an online calculator provided to get the scores based on the attack surface being targeted, the limitation of the approach is that there is no transparency on how the scoring system works, which makes it a bit ambiguous. Initially developed by NIST [5], it is now maintained by the Forum of Incident Response and Security [21]. Attack Trees [18] is one of the widely used techniques to model attacks on cyber-physical systems; however, existing studies have failed to utilize Attack Trees in detailing the attacks, as the focus is on general domains, which defeats the purpose of Attack Trees [22].

Recently, several modifications and improvements of the above discussed seminal models have been proposed. The authors in [23] proposed a template for attack modeling in the smart grid domain and highlighted the limitations of STRIDE in not being able to identify the attack severity. In [24] the authors tried to address the generality of the attack modeling. Their approach involved combining the aspects of the model-driven approaches [25] to that of general approaches [4], which would render the proposed model flexible enough to be adopted in different domains, including aerospace and automotive. However, a limitation of this approach is too many layers to adopt from, leading to a tedious task of synchronizing it to any specific domain. In [26], the authors proposed a hybrid model called SAVTA, which stands for Software, Asset, Vulnerability, Threat, and Attacker. They derived that method from the existing models, especially Attack Trees and STRIDE. The model proposed was adopted as a use case in autonomous vehicles; however, the study lacked a thorough review of the attacks demonstrated in the domain and arbitrarily mapped the attacks to SDC. The limitation was left for future work. A similar approach was followed in modelling cyber-attacks in [27]. The study demonstrated how coordinated attacks can result in a cascading effect of compromising several assets. In [18], the authors proposed an "aag" method based on Attack Trees and attack graphs, and they empirically established its efficacy against Attack Trees. However, the model was evaluated on perception, and no use-case was adopted in their work and no severity-based mechanisms were used. In [28], the authors did a survey of hardware attacks in general and gave an attack model for triggering and evaluating the attacks. In [29] they proposed a risk assessment model for connected and autonomous vehicles. The framework was based on four risk criteria, namely, probability, impact, exposure, and recovery (referred to as PIER). However, the limitation of the approach was that the framework was dependent on variables that may not be possible to evaluate, for example, the skill level of the attacker. Instead, variables should be factored on data that is accessible and widely available for the practical use of a framework. In a similar approach, the authors in [30] proposed a security framework that highlights the vulnerabilities in a smart car using Attack Tree. However, the framework merely discussed and defined assets in a smart car. The work was limited to assessing vulnerabilities under three levels of severity defined by safety, privacy, and operation. The approach was comprehensive but inconclusive. The vulnerabilities were not modelled using Attack Trees, but rather they were merely defined theoretically. So far, all the studies discussed in here lack a thorough analysis of attacks against SDC.

III. PROPOSED ATTACK ANALYSIS MODEL

The proposed model, Severity-based Analytical Attack Model for Resilience, or SAAMR, was built upon the ideas introduced in STRIDE, STAMP, PASTA, and CVSS. STAMP was adopted for its inherent capability to model feedback mechanisms and to define accidental disturbances. This study evaluates accidents as well as deliberate attempts to disrupt the system, and the adopted STAMP maps the events well, with exception of taking malicious attempts into consideration. This limitation was addressed by incorporating and modifying STRIDE, which accounts for malicious attacks; however, the definition of attacks was updated and improvised for SDC scenarios. STRIDE was modified to enlist attacks that are more relevant to SDC. The attacks were deduced from the literature review on vulnerabilities against SDC. Only those attacks that had a proof-of-concept available were included in this study. PASTA was adapted for its descriptive steps required to identify the threats or attacks using Attack Trees. CVSS was adapted for its widely available data. Thus, the proposed model includes 6 steps that are depicted in Fig. 2, which include:

1) Design out a high-level architecture of SDC.
2) Define control flow and accidental factors using STAMP.
3) Identify attacks.
4) Identify attack surfaces.
5) Create Attack Trees of the identified attacks.

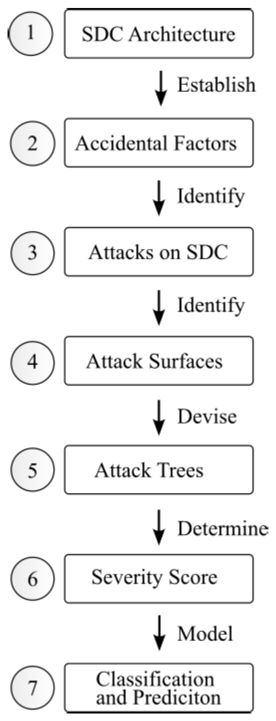


FIGURE 2. **SAAMR process.**

6) Use a SAAMR analysis of the attacks.
7) Model classification and prediction of attacks.

A. DESIGN OF HIGH-LEVEL ARCHITECTURE

SAE has defined six levels of autonomy for SDC, ranging from no autonomy to fully autonomous. The architecture that defines the autonomy of SDC can be differentiated into three layers, viz. perception layer, decision-control layer, and action layer, based upon the components and the tasks assigned to each layer. The architecture is summarized and depicted in Fig. 3. In summary, the perception layer is responsible for collecting the data from required sensors and creating state awareness through SLAM [31] techniques and aiding in object detection and tracking through DATMO [32] techniques. The perception layer can be subdivided into two components: sensor-interface and sensor-fusion module. The sensors deployed in an SDC are meant to collect the data of the environment surrounding the vehicle and the internal mechanics, like orientation, location, speed of the vehicle, etc. All the collected information is then combined through sensor-fusion algorithms and input into the decision-making modules or controllers. The controllers eventually translate the fused data into actual physical actions on the SDC through actuators. All the communication takes place through fast connecting communication paths, such as a controller area network (CAN) or ethernet. As a result, in self-driving cars, any malfunctioning of sensors, controllers, or communication links, or any surrender to malicious attack, could result in unfavorable results. An extended summary on the layers and workings of the modules can be seen in [33] and [34]. In the context of SDC, the architecture should provide the following critical services: availability, reliability, safety, and resilience.

1) AVAILABILITY

Availability in the context of SDC design and architecture means the availability of the perception and control modules for the desired communication required for actions to take place through actuators.

2) RELIABILITY

This study defines reliability for SDC in terms of the architecture that ensures the completion of its goal for the intended purpose.

3) SAFETY

This study defines safety as the assurance that any adverse effects would not cause any harm or jeopardize the well-being of its stakeholders or the system. A system could be regarded as a component of the SDC or SDC itself, while stakeholders may include any human that is taking the services of SDC.

4) RESILIENCE

Resilience is defined in this study as the ability of the SDC to function to the desired level even if the components of its architecture are partially damaged or compromised.

B. DEFINING ACCIDENTAL FACTORS USING STAMP

The requirements of safety and reliability in safety critical systems powered by software are not a new concern. Plenty of evidence from existing systems points towards catastrophic outcomes owing to system failures [35], hence the advent of the STAMP model. According to STAMP, any system can be modeled as a control process that is either automated or human-driven.

Thus, this study identifies the control process required in SDC based on the architecture developed in the previous step and creates system failure examples that can lead to system fallout. Although STAMP primarily considers socio-technical systems, SDC Level 5 is more or less supposed to be automated and independent of human intervention. Therefore, the proposed model restricts the inclusion to automated feedback control.

STAMP views the systems as interrelated components that are kept in dynamic equilibrium by feedback loops of information and control. This very much resonates with the SDC architecture, as SDC workflow inherently is a control process that operates in a loop. It becomes imperative to model the SDC architecture as a control process to identify failures, hence the inclusion of the STAMP step in the proposed model. Further, the author in [12] argued that STAMP leads to a comprehensive understanding of accidents and gave a classification of accident factors in the control process. STAMP views the cause of accidents as the result of a lack of constraints imposed on the system design and operations. As SDCs are prone to errors, including transient errors, including methods that help to indicate such failures is

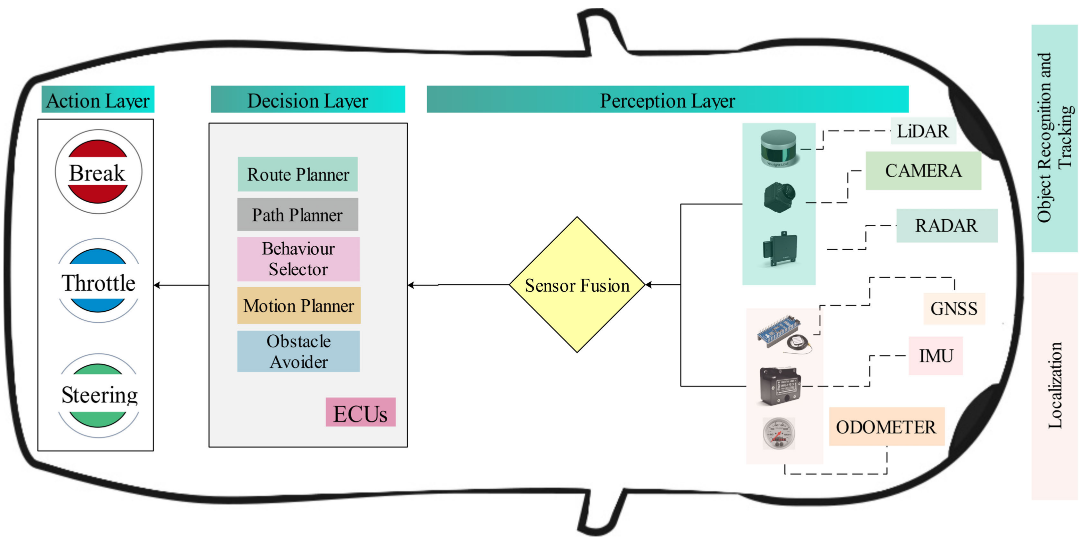


FIGURE 3. SDC high-level architecture.

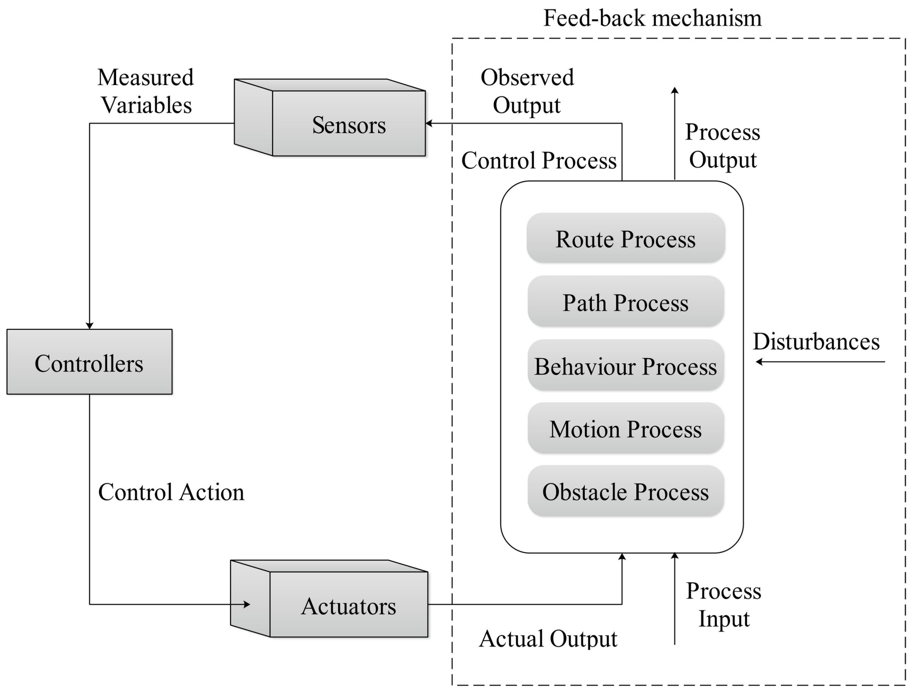


FIGURE 4. SDC STAMP process.

imperative. Thus, this study models the dataflow of the SDC architecture in a control loop structure, where the dataflow is from sensors-to-controllers, controller-to-actuator, and then a feedback mechanism follows from actuators-to-sensors. This is depicted in Fig. 4. The measured output on the actuators is compared to the observed output of the actuators. The feedback given to the control process from the observed output should be corrected accordingly. It is to be noted that the sensors as well as the control process are prone to disturbances that alter the actual measurements. Without a mechanism to counter for the loss or gain, the observed output will have erroneous input to the controllers and is therefore a misleading action for the actuators. The control process in Level 5 SDC is completely automated, and any error-laced decisions pertaining to whether to change lanes, route selection, change in speed, or applying the brakes will have serious consequences. A proper analysis of such events is needed. This study also took inspiration from the STAMP-based Hazard Analysis (STPA) in [36] and [37] and followed that methodology. STPA involves four primary steps: 1) defining the purpose of the analysis, 2) modelling the control structure, 3) identifying unsafe control actions, and 4) identifying loss scenarios. Step 1 of STPA points out the weaknesses in the architecture of the SDC described in the first subsection of Section III, that is, design of high-level architecture. Steps 2, 3, and 4 are condensed with STAMP and an analysis of accidental factors is done. Hence, with the application of STAMP and STPA, this study highlights the accident factors in SDC and defines it in the context of the SDC workflow. Thus, the point-of-failures in the control loop of SDC are identified using STAMP and STPA [36], [37], [38] according to the following factors:

1) Inadequate or inappropriate controls.
2) Inadequate execution of the control action.
3) Inadequate coordination among controllers.
4) Missing or inadequate feedback.

Interestingly, it should be noted that these factors can also very likely be triggered through adversarial action. The accident factors in this context can be summarized as:

1) INADEQUATE CONTROL ALGORITHMS

Sensors are prone to noise. Usually, in SDC, sensor-fusion mechanisms are employed to cancel out the noise or create better estimation using optimization techniques based on algorithms like Kalman Filter or deep learning algorithms. However, the techniques may not always suffice. SDC may deviate to act upon the prescribed path or route owing to faults and transient errors that optimization algorithms fail to correct. Inadequate information flow from the sensors to the controllers can also result in inadequate control action. Without any stability analysis technique on the controllers [39], [40] the outcome would be faulty and laced with errors. Recent examples of Tesla [41] and Uber [42] SDCs failing to appropriate decisions upon encountering an obstacle resulted in loss of life in two separate events. Similarly, failure to correctly perceive speed limit sign boards or emergency notifications/billboards on the road can result in fatal accidents. Another example is of GPS sensors losing signal under tunnels. Similarly, obstructions that hinder any sensors from receiving data in a timely manner may prevent the SDC from working optimally, resulting in accidents.

Constraints to correct the estimates of the optimization algorithms need to be in place, and these need to be tested for their reliability and safety using different standardized scenario-based safety checks.

2) INADEQUATE EXECUTION OF THE CONTROL ACTION

One of the critical functionalities of SDC is to tackle situations that require a change of behavior, which may include: 1) lane changes, 2) applying brakes, 3) increasing speed, and others. These control actions would be taken on actuators like steering, brake, and throttle respectively. Actuators are also prone to faults that may fail in the execution of control actions. Necessary measures to counter the failure should be employed. Redundancy is one of the plausible measures to overcome the issue; however, redundancy needs to be aided with efficient detection and switching mechanism that reduce the recovery time and mitigate the risk associated with the failure. Constraints to check for any failures in the execution of control action on the actuator need to be in place.

3) INADEQUATE COORDINATION AMONG CONTROLLERS

Multiple controllers employed in SDC for different designated control processes need to be coordinated and synchronized to lead SDC in a stable safe state. Any inconsistency or failure in one of the controllers would have a cascading effect on the rest of the controllers. Hence, a mechanism for identifying the deviation from the stable state of the SDC would have to be in place. Different process models govern the overall state of the system in SDC. This makes the control process in SDC prone to disturbances and failures. An example could be that of an error-prone route selection that, irrespective of the other processes in synchronization, would lead to the wrong target destination. Again, this could be attributed to the failure of route control process. No redundancy in the control processes to fall back to the correct estimation could have hazardous results.

4) INADEQUATE OR MISSING FEEDBACK

In SDC, sensors collect data from the external environment and the internal mechanics of the SDC, which is fused through sensor-fusion algorithms, to have the controllers act on the actuators. However, owing to the disturbances and noise, the measured output and the actual output are inconsistent. Although estimation and optimization techniques are deployed to cancel out the errors, more robust and resilient techniques are required to eliminate uncertainties regarding the measurements. The feedback mechanism needs to be aided with resilient techniques to counter any fallouts. Similar scenarios to inadequate control algorithms could lay the groundwork for SDC accidents.

C. ATTACKS ON SDC

This section highlights some of the attacks demonstrated in autonomous cyber-physical systems in general and explains them in the context of SDC architecture. Several survey papers have been published that highlight the devised attacks against SDCs [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54], [55]. However, the attacks considered in the present study are from the literature where proofs of concept of such attacks are available and have been demonstrated experimentally, either in a real-world setting or a lab setting. The references for such attacks are summarized in Table 1. Hence, the proposed model takes Spoofing, DoS, Replay, Relay, Jamming, Code-injection, Code-modification, and Adversarial Attacks into consideration. The attacks identified against SDC can be described as:

1) SPOOFING

Spoofing has been demonstrated to make vehicles vulnerable through the injection of counterfeit signals. The target could be sensors like GPS, which could be spoofed for adversarial

TABLE 1. Attack and references.

Attack	References
Spoofing	[56], [57], [58], [59], [60] [61], [62], [63], [64], [65], [66]
DoS	[67], [62]
Replay	[61], [68]
Relay	[61]
Jamming	[69], [60], [61], [62], [70], [71], [70], [72]
Code injection/modification	[73], [74] [75] [76], [77], [73], [78],
Adversarial	[79], [80], [81], [82], [83], [84], [85]

location [56], [57]. Similarly, LiDARs and cameras could be spoofed for false detection and tracking of objects [86]. Inexpensive equipment can be used to spoof the sensors and eventually carry different types of attacks, including replay and relay attacks [61], [87].

2) DENIAL OF SERVICES (DoS)

SDCs could be easy targets for DoS attacks and distributed DoS. The attacks could easily cripple SDC, as the communication links would be hogged with illegitimate requests and thereby deny any access to the legitimate requests sent to the essential SDC components. Sensor fusion and control units basically could be rendered useless, as legitimate access to these are hampered. Sensor fusion modules could simply be flooded with LiDAR or camera frames that are not of the current driving environment, keeping legitimate frames from being processed.

3) REPLAY ATTACKS

In SDC, sensors like camera, LiDAR would be collecting real time data for perception and similarly, sensors like RADAR, GPS, IMU, would be collecting data for SLAM. If an adversary finds a way to intercept the data, lets us say camera frame where an obstacle has been detected, and later same adversarial frame is replayed for false object identification would result in fatal consequences. Similar methods were adopted to compromise a LiDAR system to perceive a wall that in reality a meter away, was made to perceive it being about 20-30 meters away [61].

4) RELAY ATTACKS

Relay attacks are similar to replay attacks in that a false sense of perception could be created. The authors in [61] demonstrated a way that the original signal from the LiDAR was made to relay from different positions, creating fake echoes. Similar methods could be adopted to relay camera frames of an SDC to produce malicious output for object identification and tracking.

5) JAMMING ATTACKS

Bombing the sensors with noise interference and jamming the sensors from receiving intended signals would essentially leave an SDC blind to any perception and localization. Although an inexpensive method, proximity is required for this attack, thus lowering the probability of such attacks. However, successful jamming attacks have been demonstrated against the Tesla S model [60]. Similarly, LiDARs are vulnerable to jamming attacks. In a similar setting, cameras could be blinded by high-brightness LEDs [61].

6) CODE INJECTION AND MODIFICATION ATTACKS

ECUs and OBD-Ports have been demonstrated to be exploited for software code modification, as these modules consist of millions of lines of code that could be manipulated for adversarial motives. Similarly, CANs could be exploited to inject malicious code into the control units for adversarial action to take place [73], [77], [78].

7) ADVERSARIAL ATTACKS ON DEEP LEARNING/MACHINE-LEARNING ALGORITHMS

Adversarial attacks have become one of the most recognized attacks on deep learning algorithms. Deep learning algorithms have been extensively used for object recognition in SDCs. However, deep learning models are prone to perturbations [88]. A carefully crafted perturbation, imperceptible to the human eye, results in erroneous or adversarial classification. The algorithms or the training data used to aid in the perception of an SDC can be easily fooled and can result in misclassification. Further, the attack could be implemented as black box and white box settings [84].

Thus, based on the review done on attacks against intelligent transportation in general and SDC in particular, this study identifies the most feasible and applicable attacks per the references and proofs-of-concept available in the case of an SDC on the road.

D. ATTACKS SURFACES OF SDC

Even though Level 5 SDCs are supposed to be self-sustaining, they will be communicating with road-side units (RSUs) and other vehicles on the road for better services and experiences as the technology matures, which would expose these to further attack surfaces and vulnerabilities. Also, whether the attack can be conducted remotely or in proximity needs to be considered. The likelihood of an attack with remote access increases as ease of accessibility increases. Similarly, also based on the infrastructure being targeted, the in-vehicle communication, vehicle-to-vehicle (V2V) communication, or vehicle-to-infrastructure (V2I) is being targeted. The attack surfaces could be illustrated in Fig. 5. The summary of the attack surfaces is given in Table 2. The attacks identified in the previous section can be further described in the context of attack surfaces as follows:

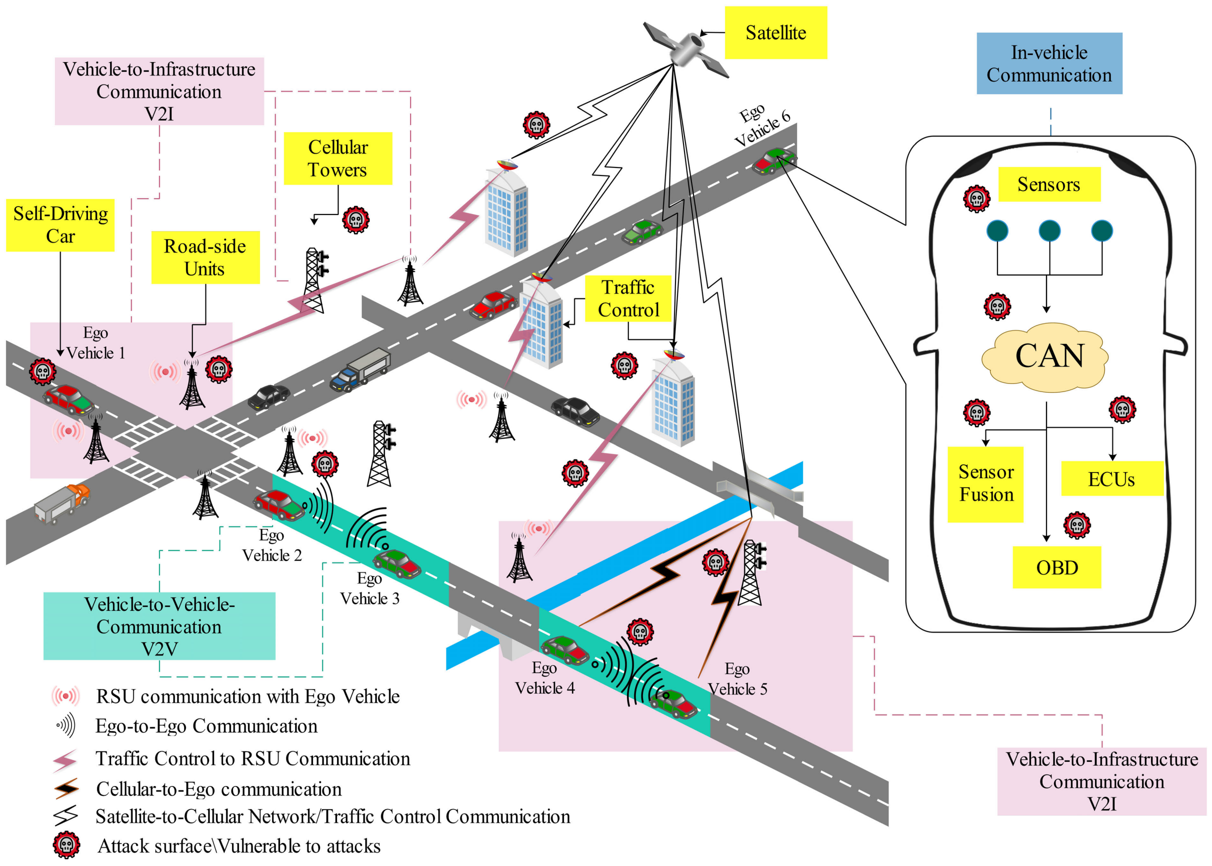

Satellite
In-vehicle Communication
Vehicle-to-Infrastructure Communication V2I
Cellular Towers
Self-Driving Car
Road-side Units
Traffic Control
Ego Vehicle 6
Ego Vehicle 1
Ego Vehicle 2
Ego Vehicle 3
Ego Vehicle 4
Ego Vehicle 5
Vehicle-to-Vehicle-Communication V2V
Sensors
CAN
Sensor Fusion
ECUs
OBD
Vehicle-to-Infrastructure Communication V2I
RSU communication with Ego Vehicle
Ego-to-Ego Communication
Traffic Control to RSU Communication
Cellular-to-Ego communication
Satellite-to-Cellular Network/Traffic Control Communication
Attack surface\Vulnerable to attacks

FIGURE 5. Attack surfaces.

TABLE 2. Attack surfaces.

Attack Channel	Attack Surface	Components
Vehicle-to-Infrastructure (V2I)	RSU-to-Ego vehicle, RSU-to-RSU-to-Ego vehicle, Cellular-to-Ego vehicle, Traffic Control-to-Ego vehicle	Satellite, Traffic Control, RSU, Cellular Tower, Ego vehicle
Vehicle-to-Vehicle (V2V)	Ego-to-Ego	SDC
In-Vehicle Communication	Sensors, CAN, Sensor fusion, ECU, OBD-Port Sensors-to-CAN,CAN-to-Sensor fusion, Sensor fusion-to-ECU, OBD-to-ECU.	Sensors, CAN, ECUs, OBD Port, Infotainment services

1) IN-VEHICLE COMMUNICATION

In-vehicle communication is responsible for the vital information that exists within the vehicle. All communication takes place through popular protocols like CAN. The attack surfaces identified in the in-vehicle network are as follows:

1) Sensor-to-sensor fusion: In SDCs, sensors interact with the external variables, which makes these easy targets for attacks like jamming. Further, the data must be made accessible through encryption and secure protocols to sensor fusion algorithms for SDC perception and localization. Any breach in CAN will give adversaries unprecedented access to the internal modules and data of the vehicle. Code modification/injection, man-in-middle attacks, etc. are examples of attacks that can be carried out on modules like the ECU. Further, any local access breach to the onboard diagnostic port (OBD-II) will have similar unprecedented repercussions.
2) Sensor fusion-to-ECU: Any malicious data or code that is transferred to electronic controller unit (ECU) for decision-making will have compromised results and hence adversarial action on the actuators. Multiple ECUs responsible for the different decision-making of the SDC, like route planning, steering, and acceleration, have a chance of being compromised with attacks like code injection/modification, spoofing, replay, relay, and DoS attacks.
3) OBD-Port II-to-ECU: Several vulnerabilities have been explored in securing OBD-Ports from adversarial inputs. Several proofs-of-concept are available, and

numerous instances in common weakness enumeration (CWE) data have exposed OBD-Port vulnerabilities. Access to OBD-Port will essentially lead to the compromise of several internal modules of SDC, including ECUs and CAN.

2) VEHICLE-TO-VEHICLE (V2V)

SDCs, also referred to as an ego vehicle, most likely will be communicating with each other for a better experience on the road and to avoid any collision or other risks pertaining to the road. The communication could take place through a *shared network* when in the vicinity of another ego vehicle. This communication can be exploited for adversarial output and hence acts as an entry point for several attacks like spoofing, replay, relay, and in extreme cases code injection/modification.

3) VEHICLE-TO-INFRASTRUCTURE (V2I)

RSUs, traffic control systems, cellular communications, and satellite communications are important infrastructure resources that SDCs rely on to make a reliable and safe journey. However, communication with these resources very likely will expose SDCs to easier remote accessibility by adversaries once the vulnerabilities in these communication channels are exploited. Several examples and proofs-of-concept are available for such vulnerabilities. Several entries in the common vulnerabilities and exposures (CVE) database also have been made for remote attacks using cellular network communication, including the remote attack on Jeep Cherokees [89]. Please note that any of these attack surfaces included will give the adversary remote accessibility to the SDC. Attack surfaces in V2I network can be described as:

1) RSU-to-ego vehicle: RSUs play an essential role in relaying information about the traffic and road condition to the SDCs. Interception of the communication between ego vehicles and RSUs by the adversary will give way to attacks like spoofing, replay, and relay attacks. Replay or relay attacks will lead to a delayed response by the SDC, which may not be correct at the time of reception. Spoofing will simply counter genuine signals with malicious ones, leading to uncertain behavior of SDCs on the road, with serious repercussions.
2) Cellular-to-ego vehicle: Cellular communication can grant easier access to multiple SDCs on the road at the same time, hence the accessibility and easiness make this attack surface more vulnerable. Enough evidence points to code injection/modification attacks through cellular networks. On-the-air updates to SDCs are more likely to be intercepted through cellular networks for malicious attacks.
3) Traffic control-to-ego vehicle: Traffic controllers may be sending real-time updates to SDCs to manage traffic on various routes. Any compromise to the communication links can cause the SDC to route to an adversarial location.
4) Road signs: The perception module of the SDC is heavily dependent on the camera. The road signs are captured by cameras, which then are input to machine/deep learning algorithms for interpretation. However, these algorithms have been found vulnerable to adversarial attacks that have caused the SDC to interpret road signs to that of adversarial choice or any sign other than the actual sign. This either can be achieved by physically manipulating the road signs or through code injection/modification to the sensor fusion modules.

However, it is also to be mentioned that whether the mode of the attack is remote or in proximity, and irrespective of the infrastructure being targeted in the context of the SDC, the assumption is that the final aim of the adversary would be to compromise the in-vehicle components of the SDC to be able to make viable damage to the SDC itself.

E. ATTACK TREES OF THE IDENTIFIED ATTACKS

The Attack Trees proposed by Schneier [90] are simple yet powerful tools for revealing vulnerabilities. This study adhered to Attack Trees for their simple representation yet explicit annotation for revealing the steps or requirements to achieve an attack in remote settings, and hence the vulnerable components of a system. The Boolean gates *AND* and *OR* represent if all the steps are to be required or there is a choice in steps to be followed to achieve the goal, respectively. For example, in Fig. 6, adversarial attack could be achieved through black box or white box settings, which in turn also could be achieved through local or remote settings. Similarly, other attack scenarios for each of the attacks described in the sub-section above were explained using Attack Trees. It should be noted that each of these trees could be elaborated on for specific technical details that may be required to carry out the specific attack. This work should be treated, however, as a stepping stone for establishing more specific and detailed Attack Trees on each of the attack methods discussed. This work lays a foundation for future work to create resilient frameworks in the autonomous car industry. Descriptions of the abbreviations used in defining Attack Trees are listed in Table 3.

1) ADVERSARIAL ATTACK TREE (*AT*)

In the scenario of an adversarial attack, the attack could be achieved through a black box (AT_B) or white box (AT_W) setting. The white box attack assumes that the attacker has complete knowledge of the deep learning model/algorithm being used and can manipulate the parameters to create an adversarial example. The black box model assumes that the attacker has little or no knowledge of the model being used. Both the types of attacks could be conducted remotely (AT_R) and locally (AT_L) depending upon the resources and access the attacker has. In the local access form of the attack, it is assumed that the attacker has compromised the OBD ($AT_{L.1.1}$) and has gained access to the in-vehicle networks of the SDC ($AT_{L.1.2}$), although achieving this is difficult. Access to the OBD port will essentially give away too many

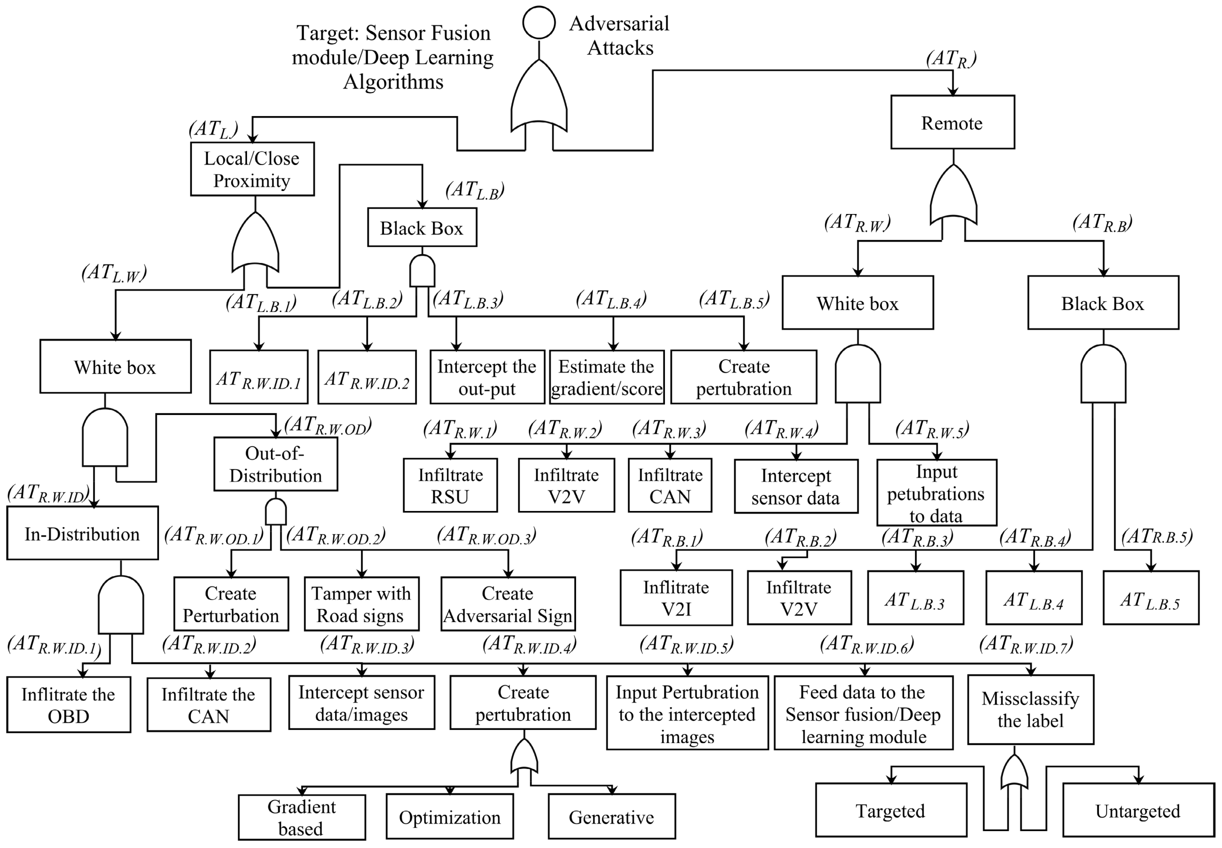


FIGURE 6. Adversarial attack tree.

TABLE 3. Attack tree terms.

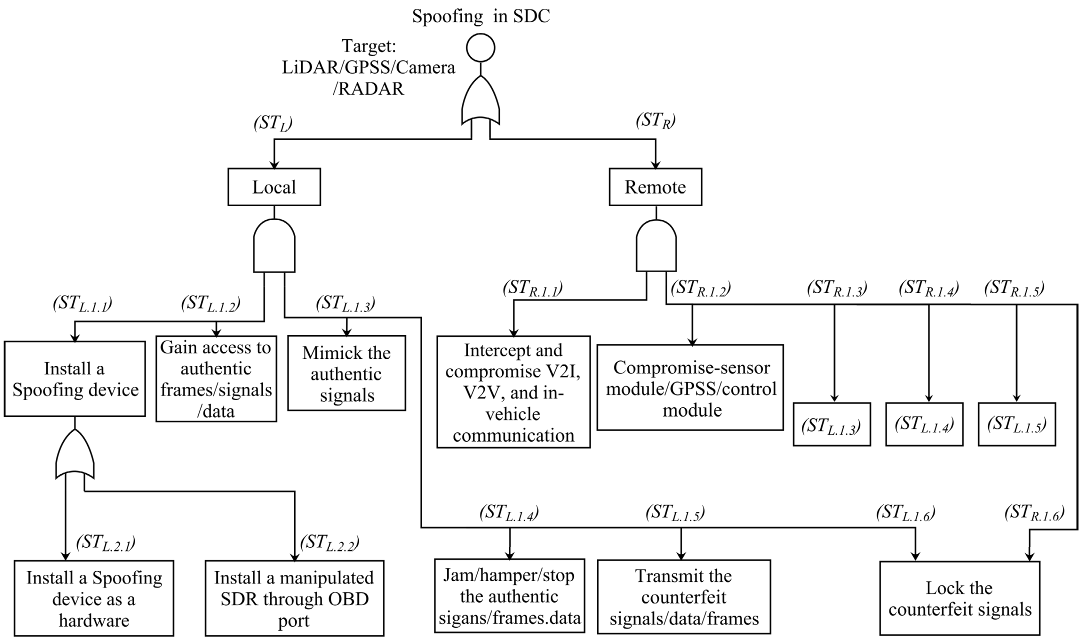
Terms	Description
AT	Adversarial Attack Tree
AT_B	Adversarial attack in a black-box setting
AT_W	Adversarial attack in a white-box setting
AT_R	Adversarial attack in a remote setting
AT_L	Adversarial attack in a local setting
ST	Spoofing Attack Tree
ST_R	Spoofing attack in a remote setting
ST_L	Spoofing attack in a local setting
JT	Jamming Attack Tree
CT	Code injection/modification Tree
CT_L	Code injection/modification Tree in a local setting
CT_R	Code injection/modification in a remote setting
RT	Replay/Relay Attack Tree
RT_L	Replay/Relay attacks in a local setting
RT_R	Replay/Relay attacks in a remote setting

of the SDC resources. However, the primary action would be to overcome any encryption or access control or any such traditional security features of the CAN ($AT_{L.1.3}$). Plenty of literature is also available to manipulate the CAN protocols that would virtually give access to the core modules of the SDC, including the sensor-fusion module, where the sensor data is munched for techniques like SLAM [31], [91] and DATMO [32]. The adversary can either induce the perturbation to the sensor data collected from the sensors ($AT_{L.1.4}$) or simply manipulate the parameters of deep learning algorithms like gradient function ($AT_{L.1.5}$).

Although the local adversarial attack is straightforward, the risk of getting caught is also high, as the adversary would have to be physically available to plug the adversarial device into the OBD port. While a bit complicated, the chances of remotely compromising the SDC for an adversarial attack (AT_R) are high, owing to low-risk factors for the adversary. The steps would involve compromising the cellular network communication between a vehicle and the infrastructure, like RSU ($AT_{R.1.1}$), which, in turn, would compromise the V2V network ($AT_{R.1.2}$). This would allow the adversary to infiltrate the internal networks of the SDC ($AT_{R.1.3}$) and eventually gain access to controller and sensor fusion modules, and the steps would then be the same for ($AT_{L.1.4}$) and ($AT_{L.1.5}$) as in a local adversarial attack. The Attack Tree for the adversarial attacks is depicted in Fig. 6.

2) SPOOFING ATTACK TREE (ST)

Spoofing involves intercepting genuine signals and replacing them with counterfeit signals. The steps would involve mimicking the genuine signals and transmitting the counterfeit signals. Spoofing has been made more feasible because of software-defined Radio (SDR) technology [92]. An attack could be conducted remotely (ST_R) as well as locally (ST_L).



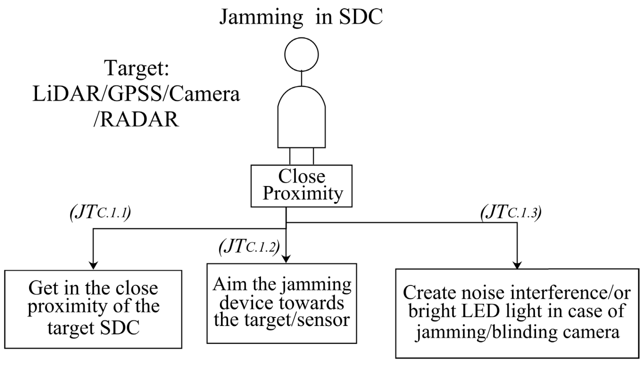
Spoofing in SDC
Target: LiDAR/GPSS/Camera /RADAR
(ST_L)
(ST_R)
Local
Remote
(ST_L.1.1)
(ST_L.1.2)
(ST_L.1.3)
(ST_R.1.1)
(ST_R.1.2)
(ST_R.1.3)
(ST_R.1.4)
(ST_R.1.5)
Install a Spoofing device
Gain access to authentic frames/signals /data
Mimick the authentic signals
Intercept and compromise V2I, V2V, and in-vehicle communication
Compromise-sensor module/GPSS/control module
(ST_L.1.3)
(ST_L.1.4)
(ST_L.1.5)
(ST_L.1.4)
(ST_L.1.5)
(ST_L.1.6)
(ST_R.1.6)
(ST_L.2.1)
(ST_L.2.2)
Install a Spoofing device as a hardware
Install a manipulated SDR through OBD port
Jam/hamper/stop the authentic sigans/frames.data
Transmit the counterfeit signals/data/frames
Lock the counterfeit signals

FIGURE 7. **Spoofing attack tree.**

The local attack could usually be achieved through installing a device that is capable of mimicking and replacing the authentic signals ($ST_{L.1.1}$). This could be achieved through a hardware ($ST_{L.2.1}$) or a software-defined component installed through the OBD port ($ST_{L.2.2}$) or a software-defined radar (SDR) in case of a RADAR signal being spoofed. The same would apply to GPS and other sensors. The spoofed hardware for GPSS is known. Cameras and LiDAR could also be vulnerable to such installations. Once the installation of a malicious software/device has been done, access to the authentic signal ($ST_{L.1.2}$) should be achieved to mimic them ($ST_{L.1.3}$). Simultaneously, jamming authentic signals should be in progress ($ST_{L.1.4}$) while transmitting of the counterfeit signals is being initiated ($ST_{L.1.5}$). Usually, malicious signals deviate little or subtly to avoid triggering any traditional security measures, hence avoiding detection. Finally, the counterfeit signals are locked onto the victim ($ST_{L.1.6}$) until the malicious objective is achieved. For a remote spoofing attack, the initial steps of compromising the cellular networks that connect V2V and V2I communications should be used to install spoofing software or a module that compromises the sensor/camera or GPSS module. Once that is achieved, the rest of the steps ($ST_{R.1.3}$), ($ST_{R.1.4}$), ($ST_{R.1.5}$), and ($ST_{R.1.6}$) are similar to ($ST_{L.1.3}$), ($ST_{L.1.4}$), ($ST_{L.1.5}$), and ($ST_{L.1.6}$), respectively. The Attack Tree for spoofing attacks is depicted in Fig. 7.

3) JAMMING ATTACK TREE (JT)

Jamming is one of the most inexpensive and simple attacks. It requires the adversary to get in proximity of the target vehicle ($JT_{C.1.1}$) and aim the jamming device at the sensor ($JT_{C.1.2}$). In SDCs, this may include a high-intensity LED or LiDAR to blind/jam the camera or LiDAR, respectively. Similar techniques can be employed to jam the other essential sensors like GPS, radar, and ultrasound ($JT_{C.1.3}$). The Attack Tree for jamming attacks is depicted in Fig. 8.



Jamming in SDC
Target: LiDAR/GPSS/Camera /RADAR
Close Proximity
(JTC.1.1)
(JTC.1.2)
(JTC.1.3)
Get in the close proximity of the target SDC
Aim the jamming device towards the target/sensor
Create noise interference/or bright LED light in case of jamming/blinding camera

FIGURE 8. **Jamming attack tree.**

4) CODE INJECTION/MODIFICATION ATTACK TREE (CT)

Code modification attacks can be carried out locally (CT_L) as well remotely (CT_R). Even if the chances of gaining access to the ODB-II port are there, the chances of getting caught or being notified are high also. The goal would be to compromise the access control of the vehicle either through the brute-force method ($CT_{L.1.1}$) or similar other methodologies to gain access to CAN for further manipulation. This can be achieved by introducing malicious packets ($CT_{L.1.2}$). A compromised CAN give access to sensor data or sensor fusion algorithms. This would allow adversaries to inject or modify the code of the fusion modules or even manipulate the

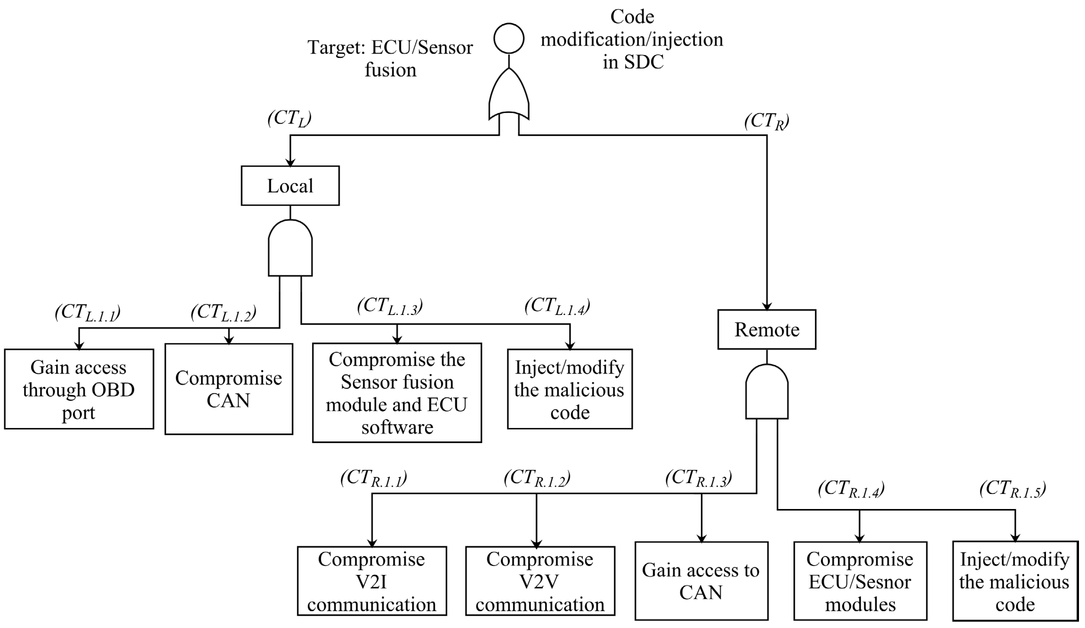

Target: ECU/Sensor fusion
Code modification/injection in SDC
(CT_L)
(CT_R)
Local
Remote
(CT_L.1.1)
(CT_L.1.2)
(CT_L.1.3)
(CT_L.1.4)
Gain access through OBD port
Compromise CAN
Compromise the Sensor fusion module and ECU software
Inject/modify the malicious code
(CT_R.1.1)
(CT_R.1.2)
(CT_R.1.3)
(CT_R.1.4)
(CT_R.1.5)
Compromise V2I communication
Compromise V2V communication
Gain access to CAN
Compromise ECU/Sesnor modules
Inject/modify the malicious code

FIGURE 9. **Code modification/injection attack tree.**

data from sensors ($CT_{L.1.3}$). The attacks could be white box or black box in nature. Then, further, the ECU software or modules could be modified for adversarial action by injecting new libraries to compromise the functionality ($CT_{L.1.4}$). The faulty or malicious control signals to the actuators would result in malicious behavioral modes that include lane changing, steering, or route selection. Code injection/modification is carried out by compromising the external networks that may include V2V or V2I communication channels. Each attack surface chosen in the V2I and V2V communication would have detailed steps to compromise the communication. The breach could happen through breaking the access control encryption or other vulnerabilities of the technologies involved in V2I ($CT_{R.1.1}$) and V2V ($CT_{R.1.2}$) communication. The rest of the steps ($CT_{R.1.3}$), ($CT_{R.1.4}$), and ($CT_{R.1.5}$) are similar to ($CT_{L.1.2}$), ($CT_{L.1.3}$) and ($CT_{L.1.4}$), respectively. The Attack Tree for the code modification/injection attacks is depicted in Fig. 9.

5) REPLAY/RELAY ATTACK TREE (RT)

Replay/relay attacks can be conducted locally (RT_L) or remotely (RT_R). The local attack can be achieved by installing malicious hardware that can intercept the signals/packets ($RT_{L.1.2}$) and record ($RT_{L.1.3}$) these to be replayed at different time interval and location ($RT_{L.1.4}$) later. This can be achieved either by plugging the device through the OBD-II port ($RT_{L.2.1}$) or compromising the CAN bus for a replay/relay attack. Another way to achieve it is to corrupt the near-field network as Bluetooth or wi-fi that is used for infotainment services ($RT_{L.2.2}$). An example of a relay attack is recorded frames of the camera being replayed at a different place or location ($RT_{L.1.4}$) to give a false perception to the SDC ($RT_{L.1.5}$). Similar steps would be followed in a remote attack with the extra steps of ($RT_{R.1.1}$) and ($RT_{R.1.2}$) for compromising the V2I, V2V communication and gaining access to the CAN bus ($RT_{R.1.3}$). Relay attacks also have the same steps as that of replay attacks, except that the goal is to relay signals to a different vehicle or device for malicious reasons. Finally, remote attacks would have the same steps, except for the initial steps of gaining access through V2I, V2V communications to the CAN bus, i.e., ($RT_{R.1.1}$), ($RT_{R.1.2}$), and ($RT_{R.1.3}$). The rest of the steps ($RT_{R.1.4}$), ($RT_{R.1.5}$), ($RT_{R.1.6}$), ($RT_{R.1.7}$) remain the same as those of ($RT_{L.1.2}$), ($RT_{L.1.3}$) ($RT_{L.1.4}$), and ($RT_{L.1.5}$). The Attack Tree for code replay/relay attacks is depicted in Fig. 10.

F. SAAMR SEVERITY ANALYSIS OF THE ATTACKS

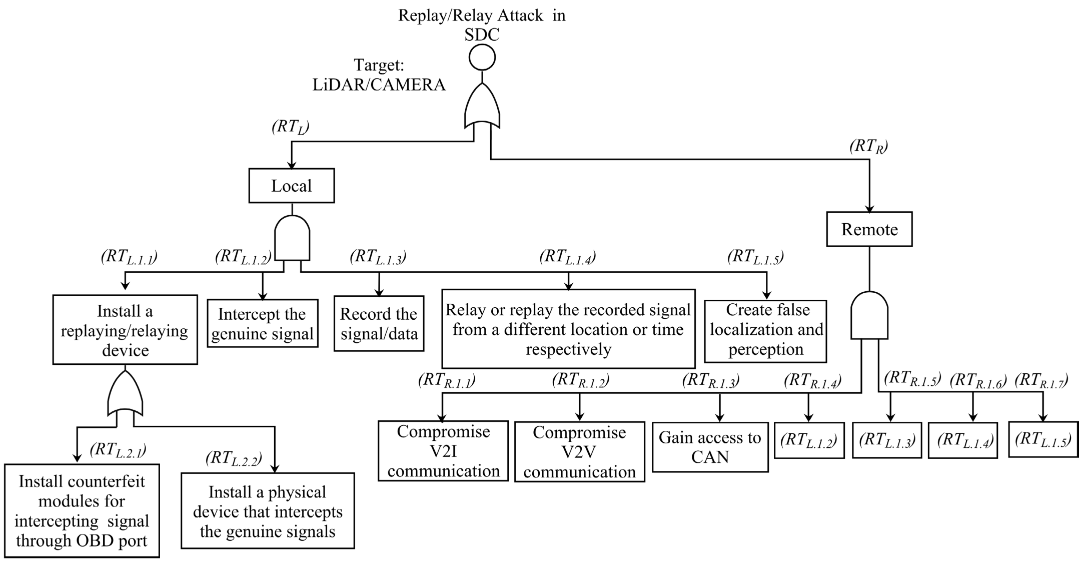
To do the SAAMR severity analysis of the attacks identified against SDC, we leveraged the CWE dataset. This step of the SAAMR severity analysis process can be further summarized in the following sub-steps:

1) Description of the CWE dataset.
2) Data preprocessing, curation and encoding.
3) SAAMR Severity Score.

1) DATA DESCRIPTION

This study leveraged the CWE vulnerability data and mapped the vulnerabilities to five core attacks against SDC recognized from the literature reviews. The literature review considered those attacks that have been designed as a proof-of-concept. Most of these attacks considered were against SDCs or connected vehicles or were related to Intelligent Transportation Systems (ITS). A summary of the attacks was given in the previous sub-section of the model proposed.

The CWE data considered in this work contains 88,777 instances of vulnerabilities identified from the year 2000 to 2019. Although there was a recent data set available through the year 2021, the data was unverified and



Replay/Relay Attack in SDC
Target: LiDAR/CAMERA
(RT_L)
(RT_R)
Local
Remote
(RT_L.1.1)
(RT_L.1.2)
(RT_L.1.3)
(RT_L.1.4)
(RT_L.1.5)
Install a replaying/relaying device
Intercept the genuine signal
Record the signal/data
Relay or replay the recorded signal from a different location or time respectively
Create false localization and perception
(RT_R.1.1)
(RT_R.1.2)
(RT_R.1.3)
(RT_R.1.4)
(RT_R.1.5)
(RT_R.1.6)
(RT_R.1.7)
Compromise V2I communication
Compromise V2V communication
Gain access to CAN
(RT_L.1.2)
(RT_L.1.3)
(RT_L.1.4)
(RT_L.1.5)
(RT_L.2.1)
(RT_L.2.2)
Install counterfeit modules for intercepting signal through OBD port
Install a physical device that intercepts the genuine signals

FIGURE 10. Replay/relay attack tree.

TABLE 4. Keywords.

Keywords	CAN, Controller area network, CAN bus, Bus, ECU, Electronic Controller Unit, Sensor, Sensor fusion, GPS, OBD-II, OBD-II Port, USB-PORT, controller, GPU, Vehicle, Remote, RSU, Road side unit, LiDAR, IMU, Inertial Measurement Unit, DDoS, Distributed denial of service, Denial of service, Spoofing, Replay attack, Relay Attack, Code injection, Injection, Code modification, Modification, Jamming, OBD-II Attack, Bluetooth, Wifi, Air bag, Tire pressure, TPM, Tire Pressure Measurement, Tesla, Toyota, Uber, Baidu, Mercedes, Waymo, BMW, Continental, Ford, Mitsubishi, Subaru, Mercedes, GMC, Parking Lot, Keyless Entry, MobilEye, Firmware code, Usb-interface, Head-unit, PCU, Pyrotechnical controller unit, Command injection, Automobile, infineon, Flexray, Telematic Control Unit, TCU, Cellular Network, Man in the middle attack, Sensor network, Traffic control.

incomplete for the years 2020 and 2021. Therefore, the data was restricted to entries through 2019. The vulnerabilities have been reported in different software products designed by different companies. Some of the instances were clearly identified against the SDCs, while some had to be checked for a thorough description. Each vulnerability is described by 170 CWE-Identifiers, which gives a description and detail of the vulnerability. Each vulnerability is defined by six attributes viz. "access authentication", "access authentication", "access complexity", "access vector", "impact confidentiality", "impact availability", and "impact integrity".

2) DATA PREPROCESSING

This subsection describes the preprocessing methods for curation, labeling, and encoding of the dataset. The keywords mentioned in Table 4 are used to identify instances of specific attacks and if any were against the SDC. Out of 88,777, the instances were reduced to 34,347 instances of vulnerabilities. Out of these instances, the vulnerabilities were manually mapped to the core attacks identified. This was done by searching the CWE-code ID of the vulnerabilities and checking the description of each vulnerability on their proprietary website [93]. Initially, these vulnerabilities were broadly classified under 170 unique CWE-code ID values. Each CWE-code gave a broad description of the vulnerability. Each CWE-code was labelled manually as one of the five attack categories identified earlier through the literature review. The labeling was based on the thorough description and consequences of such vulnerabilities explained on their proprietary website as mentioned earlier. Thus, the study mapped 170 of the CWE-Code instances to the five attack categories identified. The vulnerabilities for which there was no description or which had already been discarded were eliminated from the data. This further reduced the data to 33,667 instances of unique vulnerabilities labeled under the five attack categories.

As there were no instances of jamming attacks against SDC being recorded, thereby after filtering the data, the jamming attack was discarded as there were no instances identified for SDC, eliminating it from the analysis of the severity and prediction of the attack. For simplicity, the spoofing, replay, and relay attacks were collated, as the resources and vulnerabilities described in the CWE database would lead to similar outcomes for these attacks.

Further, each feature of the data was encoded to the existing categories to fit the data for the classification and prediction model on the described attacks. Each feature was label encoded for its categories. The encoding is listed in Table 5.

3) SAAMR SEVERITY SCORE

To calculate the accumulative severity score for attack categories using CVSS, the score was based on each of the

TABLE 5. Label encoding.

Access Authentication		Access Complexity	
Category	Code	Category	Code
Multiple	2	High	2
None	0	Low	0
Single	1	Medium	1
Access Vector		**Impact Confidentiality**	
Category	Code	Category	Code
V2V	0	Complete	2
InV	1	None	0
V2I	2	Partial	1
Impact Availability		**Impact Integrity**	
Category	Code	Category	Code
Complete	2	Complete	2
None	0	None	0
Partial	1	Partial	1

attack categories. The calculation was made according to the Risk Priority Number (RPN) [94]. The RPN is a numeric assessment of risk assigned to each mode with a numeric value and quantifies the likelihood of occurrence, likelihood of detection, and severity of impact. In this context, it is called the SAAMR severity score and is denoted as R_s. The higher the value of R_s, the more the likelihood of occurrence, detection, and impact of the attack. The R_s is calculated for each category by:

$$R_s = S_v * O_c * D \tag{1}$$

where S_v is the mean CVSS score for a Attack Group. O_c is an instance of CVSS score for each attack in the Attack Group (A_g). And D is the number of times the attack was detected, which is assumed to be at least 10 in the present context.

There are five classes of groups (g) that range from 0 to 4. For each attack group (A_g), the count of instances, N_g is given as:

$$N_g = \sum_{I=1}^{n_g} C_{ou}(A_{Ig}) \tag{2}$$

where $C_{ou}(A_{Ig})$ is the instance of attack (I) of Attack Group (g). For each g, the occurrence, O_{c_g} is given by:

$$O_{c_g} = \frac{\sum_{I=1}^{n_g} C_{ou}(A_{Ig})}{\sum_{g=0}^{4}\{\sum_{i=1}^{n_g} C_{ou}(A_{ig})\}} \tag{3}$$

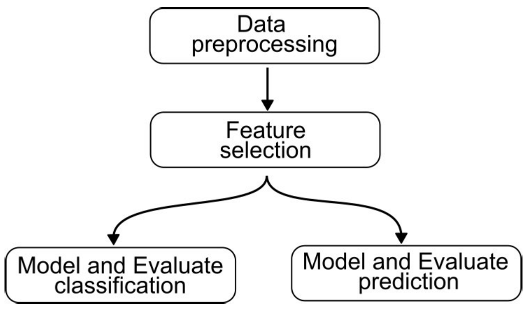

Data preprocessing
Feature selection
Model and Evaluate classification
Model and Evaluate prediction

FIGURE 11. Data modelling steps.

where $\sum_{I=1}^{n_g} C_{ou}(A_{Ig})$ is the sum total of count (N_g) of instances of an Attack Group (A_g). And $\sum_{g=0}^{4}\{\sum_{i=1}^{n_g} C_{ou}(A_{ig})\}$ is the sum total of count (N) of instances of all the Attack Groups. S_{v_g} is calculated as:

$$S_{v_g} = \left\{\frac{\sum_{i=1}^{N_g} cvss_{ig}}{N_g}\right\} \tag{4}$$

where $cvss_{ig}$ is the score of occurrence (instance) attack (i) for the group (g) and $\sum_{i=1}^{N_g} cvss_{ig}$ is the sum of CVSS scores for an Attack Group A_{ig}. And N_g is the count of instances in the group. Thus, the severity score calculated for each attack group is enlisted in the Table 6.

G. CLASSIFICATION AND PREDICTION

1) FEATURE ENGINEERING AND SELECTION

The steps followed for the classification and prediction are depicted in Fig. 11. Further, in this study, we modeled the data as a dataframe using Pandas [95], [96] library package in python. The input feature vectors in the dataframe were analysed for correlation using Spearmans and Pearson coefficient; the results are very similar and are depicted in Fig. 12 and Fig. 13 respectively. Impact Availability, Impact Integrity, Impact Confidentiality, and CVSS seem to show a strong correlation among each other. Since this study was dealing with categorical values, both the correlations did not show any significance of the target variable to that of input variables. Hence, a Chi-square test was done to confirm the null hypothesis that each of the featured variables was independent of the target variable. The values of significance were improbably large, which affirms the null hypothesis. Since the P values are larger than the conventional threshold of 0.05,

TABLE 6. SAAMR severity score.

Attack	Target	S_v	O_c	D	R_s
Adversarial	Sensor data/fusion/Deep learning	7.465	0.029	10	2.19
Code injection/modification	ECU, Sensor fusion	6.233	0.640	10	39.89
DoS	CAN, sensors, ECU	6.446	0.098	10	6.32
OBD-Port	OBD-Port, CAN, sensors, ECU	4.935	0.002	10	0.10
Replay/Relay/Spoofing	LiDAR, Camera, GPS	5.771	0.230	10	13.30

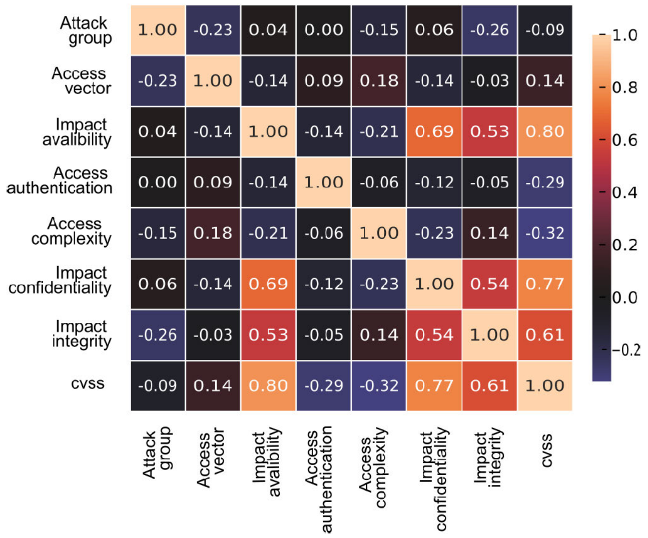


FIGURE 12. Spearman correlation matrix.

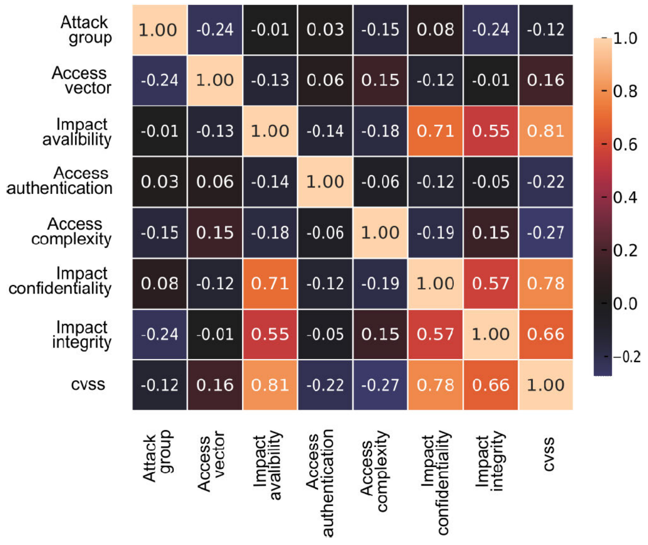


FIGURE 13. Pearson correlation matrix.

all the features were considered to have independence. This might be attributed to the fact that since all the attributes are categorical or ordinal, it is difficult to establish a significant value in this case. Also, dropping and keeping only one of the highly correlated features, namely Impact Integrity, Impact Confidentiality, and CVSS, resulted in a lower accuracy rate. Hence, all the features were input for classifying and predicting the target variable, which resulted in better accuracy rates. The distribution of each feature vector is depicted in Fig. 14.

2) MODELLING THE DATA FOR CLASSIFICATION

After pruning the data, the features were input using seven features, viz. "access authentication", "access complexity", "access vector", "impact availability", "impact confidentiality", "impact integrity", and "CVSS" for classification of the "Attack Group". Entropy was used to measure the impurity in the given data set instead of the Gini index, since the attributes used here are categorical. The lesser the entropy, the more information gained. The formula for the Entropy is given as:

$$Entropy = \sum_{i=1}^{n} -p_i * \log_2(p_i) \tag{5}$$

where n is the number of classes and p_i can be defined as the probability associated with the *ith* class.

3) EVALUATING CLASSIFICATION MODEL

The study achieved 73.8171% level of accuracy. Although several methods were tried to increase the accuracy by manipulating the model itself, it resulted in over-fitting of the model. One of the techniques attempted was the addition of another feature called references, which was the number of references of proofs-of-concept found through literature review for each category of attack. However, as mentioned, this resulted in over-fitting of the model. Hence, the best score for the said model was 73.8171% using Decision Tree. This is explained by the skewed data that has fewer instances for the attacks that are rare, specifically adversarial attacks and physical attacks like OBD-Port attacks. Physical attacks are difficult to achieve and have a higher chance of being detected, and they require direct contact with the target. The instances for both attacks are very few, and hence the model fails to learn to classify them entirely. Adversarial attacks, however, are very specific to SDCs and compromise their perception module. Once the prevalence of SDCs increases on the road, more adversarial attacks may be observed. But for now, very little data is available for such attacks. The availability of more data should help in creating classification models with a higher accuracy rate. The confusion matrix with precision and recall values is mapped in Table 7. The representation of the decision tree employed is in Fig. 15.

4) MODELLING THE DATA FOR PREDICTION

Finally, the study mapped the features to predict attacks using long short-term memory networks (LSTM). We mapped all the features for prediction as our conclusion was derived

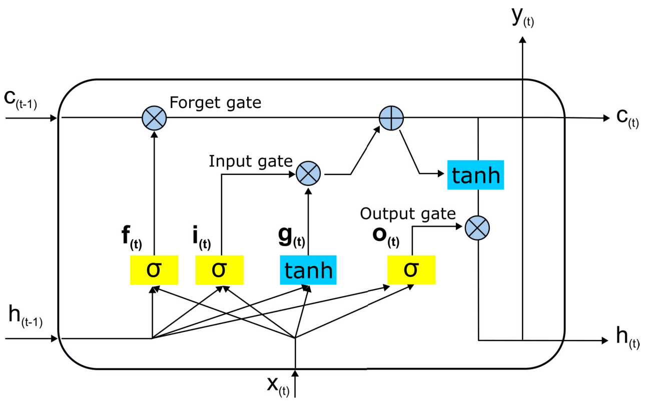


FIGURE 16. LSTM Cell.

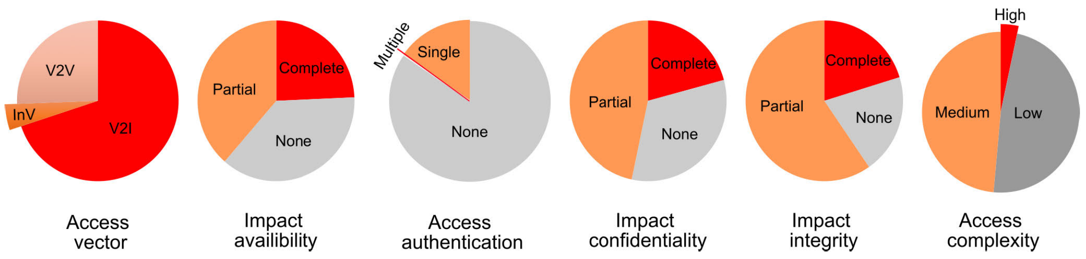


FIGURE 14. Distribution of modeled features.

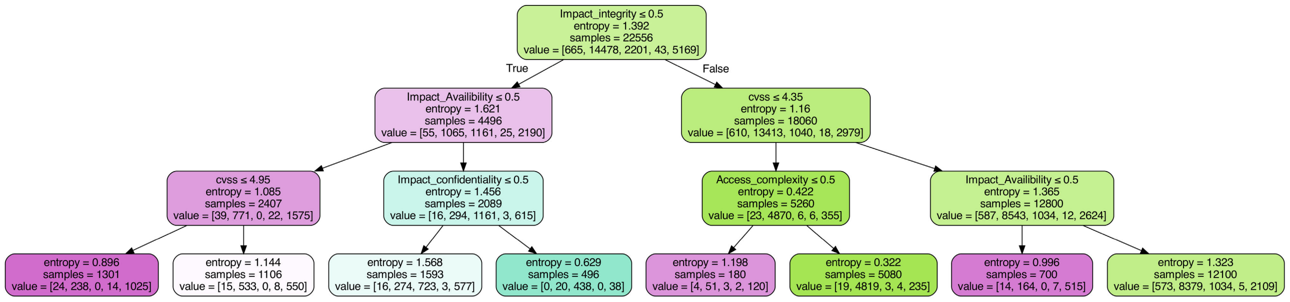

Impact_integrity ≤ 0.5
entropy = 1.392
samples = 22556
value = [665, 14478, 2201, 43, 5169]
True
False
Impact_Availibility ≤ 0.5
entropy = 1.621
samples = 4496
value = [55, 1065, 1161, 25, 2190]
cvss ≤ 4.35
entropy = 1.16
samples = 18060
value = [610, 13413, 1040, 18, 2979]
cvss ≤ 4.95
entropy = 1.085
samples = 2407
value = [39, 771, 0, 22, 1575]
Impact_confidentiality ≤ 0.5
entropy = 1.456
samples = 2089
value = [16, 294, 1161, 3, 615]
Access_complexity ≤ 0.5
entropy = 0.422
samples = 5260
value = [23, 4870, 6, 6, 355]
Impact_Availibility ≤ 0.5
entropy = 1.365
samples = 12800
value = [587, 8543, 1034, 12, 2624]
entropy = 0.896
samples = 1301
value = [24, 238, 0, 14, 1025]
entropy = 1.144
samples = 1106
value = [15, 533, 0, 8, 550]
entropy = 1.568
samples = 1593
value = [16, 274, 723, 3, 577]
entropy = 0.629
samples = 496
value = [0, 20, 438, 0, 38]
entropy = 1.198
samples = 180
value = [4, 51, 3, 2, 120]
entropy = 0.322
samples = 5080
value = [19, 4819, 3, 4, 235]
entropy = 0.996
samples = 700
value = [14, 164, 0, 7, 515]
entropy = 1.323
samples = 12100
value = [573, 8379, 1034, 5, 2109]

FIGURE 15. Decision tree for classification.

TABLE 7. Confusion matrix for decision tree attack classification.

		Actual						
		Adversarial	**Cod/Mod**	**DoS**	**OBD**	**Spoof/Replay/Relay**	**Precision**	**Support**
Predicted	**Adversarial**	0	277	10	0	38	0.0	325
	Cod/Mod	0	6443	135	0	491	0.77	7069
	DoS	0	508	595	0	0	0.57	1103
	OBD	0	4	2	0	19	0.00	25
	Spoof/Replay/Relay	0	1138	310	0	1141	0.68	2589
	Recall	0.00	0.91	0.54	0.00	0.44		

from correlation analysis done during the feature selection stage of modeling the data. The prediction of the type of attack was set to an hourly basis, which was projected as the time steps. The premise of using LSTM is a successful implementation for predicting time series data against traditional neural networks. LSTM, like traditional RNNs, has a memory cell that provides the information of the current moment to the subsequent moment, but with the advantage of optimizing memory cells through the introduction of gates; otherwise, this leads to gradient vanishing (more often) or gradient explosion in traditional RNNs.

In the traditional RNN, at each time step, the recurrent neuron receives the input x_t as well its own output from the previous step, y_{t-1}. Because of this phenomenon, it can be said, that output of a neuron at a time step t is a function of the previous time steps and hence forms a sort of memory. However, this leads to RNNs having only one state h_t, which is the output of the neural network, and it can read short-term information but is unable to decipher any long-term information. To address the issue, LSTM explicitly employs memory cell and gates viz. *forget gate* (f_t), *input gate* (i_t) and *output gate* (o_t) that decide which information to exclude or store in the memory cells and thereby control the information flow in neural network. The state of the LSTM cell is split into two vectors, that is, h_t and c_t. h_t can be thought of the short-term state, while c_t can be thought of as the long-term state.

This can be understood from the basic architecture of the LSTM cell represented in the Fig. 16. In LSTM, there are four neural network layers instead of just one, and they interact in a very unique way. The layer that outputs $g_{(t)}$ is the primary layer. The layer is akin to a basic cell and has the usual role of analyzing the current inputs $x_{(t)}$ and the previous short term state. It is expressed as:

$$g_{(t)} = tanh({W_{xg}}^{\top} x_{(t)} + {W_{hg}}^{\top} h_{(t-1)} + b_g) \tag{6}$$

where W_{xg} is the weight matrix for the main layer assigned for its connection to the input vector $x_{(t)}$. W_{hg} is the weight matrix assigned for its connection to the previous short-term state $h_{(t-1)}$. And, b_g is the bias term for the layer.

The other three layers act as gate controllers and are simple sigmoid threshold units. Their outputs range from 0 to 1. As the long-term information c_{t-1} flows from left to right in the network, the information is first intercepted by the forget gate f_t, allowing to drop some memories or to add some through pointwise operations. The forget gate $f_{(t)}$ regulates and determines which elements of the long-term state should be erased. It is computed as:

$$f_{(t)} = \sigma({W_{xf}}^{\top} x_{(t)} + {W_{hf}}^{\top} h_{(t-1)} + b_f) \tag{7}$$

where σ is the sigmoid function, W_{xf} is the weight matrix for forget layer assigned for its connection to the input vector $x_{(t)}$. W_{hf} is the weight matrix assigned for the layers connection to the previous short-term state $h_{(t-1)}$. And, b_f is the bias term for the layer.

The added memories are selected by the input gate $i_{(t)}$. Without any additional alteration, the result c_t is simply sent out. Thus, at each time step, some memories are deleted while others are added. The input is computed as:

$$i_{(t)} = \sigma({W_{xi}}^{\top} x_{(t)} + {W_{hi}}^{\top} h_{(t-1)} + b_i) \tag{8}$$

where σ is the sigmoid function, W_{xi} is the weight matrix for input layer assigned for its connection to the input vector $x_{(t)}$. W_{hi} is the weight matrix assigned for the layers connection to the previous short-term state $h_{(t-1)}$. And, b_i is the bias term for the layer.

Additionally, the long-term state is duplicated after the addition operation, and is transmitted through the *tanh* function, and then the output gate o_t filters the outcome. This produces the short-term state h_t which is equivalent to the cells output of current time-step y_t. Thus, the function o_t controls which portions of the long-term state should be read and output to h_t and y_t. The function is computed as:

$$o_{(t)} = \sigma({W_{xo}}^{\top} x_{(t)} + {W_{ho}}^{\top} h_{(t-1)} + b_o) \tag{9}$$

where σ is the sigmoid function, W_{xo} is the weight matrix for output layer assigned for its connection to the input vector $x_{(t)}$. W_{ho} is the weight matrix assigned for the layers connection to the previous short-term state $h_{(t-1)}$. And, b_o is the bias term for the layer.

Based on the equations (6)-(9), c_t and y_t and are computed as:

$$c_{(t)} = f_{(t)} \otimes c_{(t-1)} + i_t \otimes g_{(t)} \tag{10}$$

$$y_{(t)} = h_{(t)} = o_{(t)} \otimes tanh(c_{(t)}) \tag{11}$$

As explained above, the information transmission in LSTM is through hidden layer cells and is controlled by the input gate, forgotten gate, and output gate. In our work, LSTM was defined with 100 neurons in the first hidden layer and 1 neuron in the output layer for predicting the Attack Group in time steps. The normalized data set was constructed as such that each observation of the input variable was set as the previous time-step $(t-1)$ and the target variable as the current time-step (t). Mean Absolute Error (MAE) was used as a loss function and Adam as the optimization algorithm.

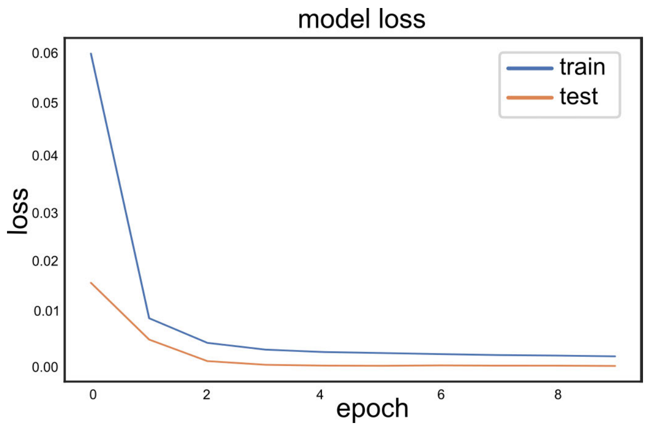


FIGURE 17. Loss values.

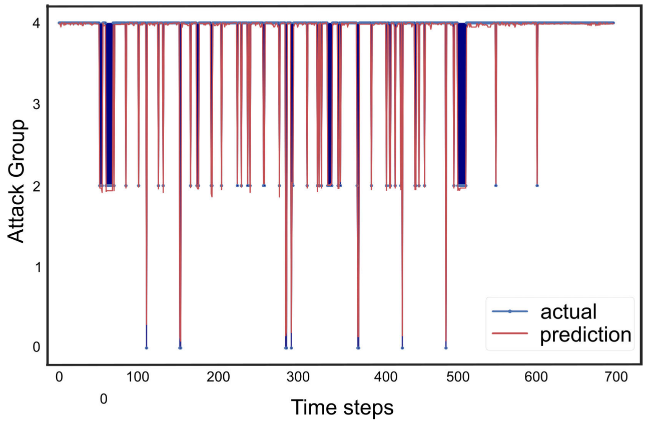


FIGURE 18. Prediction.

The training and test loss is plotted in Fig. 17. The model was fit for 10 training epochs with a batch size of 70 and a dropout rate of 20%. The root mean square error (RMSE) of the test is 0.218. The actual and predicted values are depicted in Fig. 18. From Fig. 17 and Fig. 18, it seems that LSTM is performing well in predicting the attack types, as the loss values drop considerably after 4 epochs and stabilize after that. Even performing better on the test data than on the training data. However, this phenomenon can be attributed to having smaller test data, and even so, having fewer instances of some of the attacks. Another common cause of performance is the regularization technique (dropout) used. The dropout is only used during the training phase and not the testing phase. This could also be the reason that LSTM performs better on the validation set than on the training set. As of now, the study has not emphasized much on the performance of LSTM, as we do understand that the results will be better and more interpretable as more attacks against SDC get registered and the data becomes less skewed. Although one of the approaches would be to do cross-validation to remove any doubts about over-fitting. However, we do have the dropout rate set at 20% to better generalize the results and avoid overfitting. Since many of the limitations are from the dataset itself, we have kept the recommendations for future work.

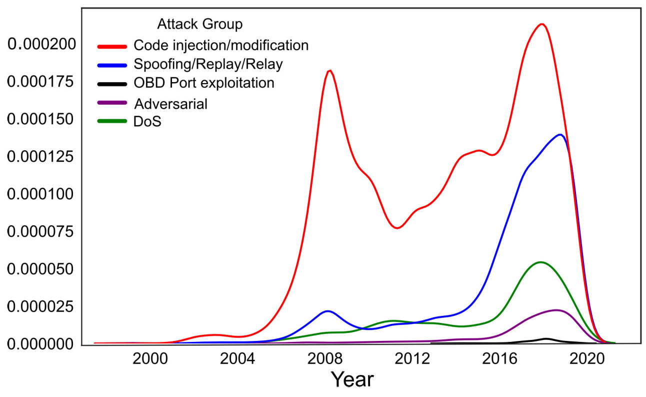


FIGURE 19. SDC attack group density.

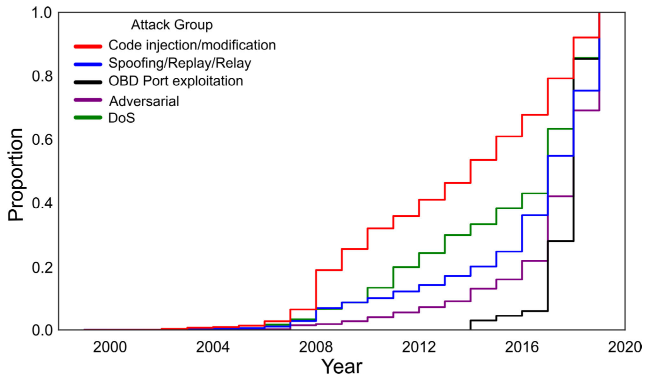


FIGURE 20. SDC attack group proportion.

IV. DISCUSSION

It is important to note that this study argues that in the foreseeable future, there will be an increase in the number of attacks, and traditional measures of security will not suffice, hence the need for appropriate attack modeling techniques to develop better resilient approaches. This section gives an insight into the argument made and reveals how susceptible SDCs will be in the near future, besides discussing the limitation and future directions of the work.

Overall, the density and occurrences of each attack have been increasing over the years, with code injection/modification attacks having a much higher density, that is, such attacks have taken place more frequently and have been increasing over the years. This was followed by incidents of spoofing, replay and relay attacks, and so on. This finding is depicted in Fig. 19. Similarly, the proportion of each attack has also grown over the years. That is, the vulnerabilities that lead to each attack category have also increased and been exploited. This finding is depicted in Fig. 20. Likewise, the mean impact of categories for all the attacks has been depicted in Fig. 21. From the figure, it can be concluded that the general increase in the number of attacks throughout the years has resulted in each of the features being exploited considerably to cause severe effects.

In summary, the paper identified accidental factors and gave an attack analysis of the vulnerabilities faced by SDCs. Although there are limitations to the approach, the authors believe that this is the first work that details the attack

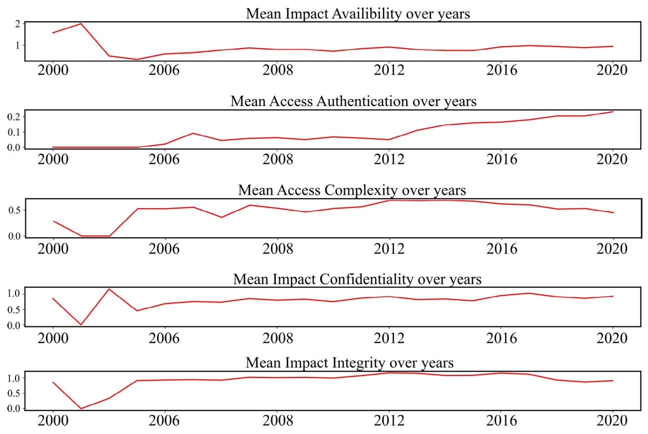


FIGURE 21. Impact of attacks over the years.

analysis model of SDCs. Further, there is no data-oriented approach that gives the attack model and that highlights the vulnerabilities in the SDC infrastructure. With SDC concepts gaining investment from major companies around the world, their safety and reliability remain a concern. Further, SDCs will likely have to be maintained remotely by their vendors for security and maintainability, which will open doors for exploits against them. With a greater number of SDCs on the road, the possibility and likelihood of being compromised will also increase. Unfortunately, unlike other cyber-physical systems, in SDCs, this can lead to fatal consequences.

Currently, there are a few limitations to the approach used in this study. The authors did not consider an accident analysis separately, since the focus of the study was more on the attacks, and the assumption was that the outcome of the accidents would be the same as the attacks. However, in the case of attacks, there is adversarial motivation. Future work aims to propose a thorough accident analysis based on fault trees and data, if available, on failures of SDCs. Further, owing to the lack of data available on attacks against SDC, the data is a bit skewed. As more data is made available on different attacks on SDC, it will improve the classification and prediction results. For future work, the authors would also like to engineer more features that would help to increase the accuracy results for classification and prediction results. For the current work, one of the pragmatic approaches would have been to drop the classes with fewer instances; however, the current paper is not focused on the accuracy of the classification but rather on proposing an attack model and identifying the vulnerabilities and attacks that SDCs would be facing. Further, the authors plan to employ NLP techniques for the identification of attacks through multi-word expressions in future work. This would allow automating the manual process of selecting SDC vulnerabilities from the summary of each vulnerability description.

Further, the SAAMR severity score could be improved and fine-tuned to SDC architecture. Currently, contrary to other fields of research, the domain lacks enough data to give details pertaining to the SDC vendors, the technologies they use, and the vulnerabilities identified with each technology.

Hence, further improvements could be made in streamlining the data.

V. CONCLUSION

There is currently a lack of study on identifying vulnerabilities in the SDC architecture. This work aims to fill the gap, although there is scope for improvement over the model proposed. However, this work lays a sound foundation in the direction and is a preliminary step in devising an SDC framework that is resilient to attacks, which is a work in progress. For now, in the context of current work, the authors presented an LSTM-based deep-learning approach that predicts the type of attack on SDC architecture in addition to providing a model for identifying vulnerabilities in SDC architecture.

ACKNOWLEDGMENT

The authors gratefully acknowledge the technical and financial support from the Deanship of Scientific Research (DSR) at the King Abdulaziz University (KAU), Jeddah, Saudi Arabia, under the grant number RG-8-611-40.

REFERENCES

[1] *J3016: Taxonomy and Definitions for Terms Related to Driving Automation Systems for on-Road Motor Vehicles*, SAE, Warrendale, PA, USA, 2021. [Online]. Available: https://www.sae.org/standards/content/j3016_202104/

[2] T. Davenport, "Toyota looks pretty smart right now on autonomous vehicles," Forbes, 2020. [Online]. Available: https://www.forbes.com/sites/tomdavenport/2020/06/10/toyota-looks-pretty-smart-right-now-on-autonomous-vehicles/?sh=5313a1d87123

[3] N. Shevchenko, T. A. Chick, P. O. Riordan, T. P. Scanlon, and C. Woody. (Jul. 2018). *Threat Modeling: A Summary of Available Methods*. [Online]. Available: https://apps.dtic.mil/sti/citations/AD1084024%0 and https://resources.sei.cmu.edu/asset_files/WhitePaper/2018_019_001_524597.pdf

[4] D. J. Bodeau, C. D. Mccollum, and D. B. Fox, "Cyber threat modeling: Survey, assessment, and representative framework," MITRE, Bedford, MA, USA, Tech. Rep. 18, 2018. [Online]. Available: http://www.mitre.org/HSSEDI%0 and https://www.mitre.org/sites/default/files/publications/pr_18-1174-ngci-cyber-threat-modeling.pdf and https://www.mitre.org/sites/default/files/2021-11/prs-18-1174-ngci-cyber-threat-modeling.pdf and https://www.mitre.org/locations

[5] C. Paulsen and R. Byers, "Glossary of key information security terms," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. Revision 2, Jul. 2019. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/ir/2013/NIST.IR.7298r2.pdf and https://nvlpubs.nist.gov/nistpubs/ir/2019/NIST.IR.7298r3.pdf

[6] S. Paudel, P. Smith, and T. Zseby, "Attack models for advanced persistent threats in smart grid wide area monitoring," in *Proc. 2nd Workshop Cyber-Physical Secur. Resilience Smart Grids*. New York, NY, USA: ACM, Apr. 2017, pp. 61–66, doi: 10.1145/3055386.3055390.

[7] A. V. Uzunov and E. B. Fernandez, "An extensible pattern-based library and taxonomy of security threats for distributed systems," *Comput. Standards Interface*, vol. 36, no. 4, pp. 734–747, Jun. 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0920548913001827

[8] R. N. Dahbul, C. Lim, and J. Purnama, "Enhancing honeypot deception capability through network service fingerprinting," *J. Phys., Conf.*, vol. 801, Jan. 2017, Art. no. 012057, doi: 10.1088/1742-6596/801/1/012057.

[9] M. Frydman, G. Ruiz, E. Heymann, E. César, and B. P. Miller, "Automating risk analysis of software design models," *Sci. World J.*, vol. 2014, pp. 1–12, Jun. 2014, [Online]. Available: http://www.hindawi.com/journals/tswj/2014/805856/, doi: 10.1155/2014/805856.

[10] D. Dhillon, "Developer-driven threat modeling: Lessons learned in the trenches," *IEEE Secur. Privacy*, vol. 9, no. 4, pp. 41–47, Jul. 2011.

[11] J. Meier, A. Mackman, S. Vasireddy, M. Dunner, R. Escamilla, and A. Murukan. (2003). *Improving Web Application Security: Threats and Countermeasures*. [Online]. Available: https://www.microsoft.com/en-us/download/confirmation.aspx?id=1330

[12] N. Leveson, "A new accident model for engineering safer systems," *Saf. Sci.*, vol. 42, no. 4, pp. 237–270, 2004. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S092575350300047X

[13] T. Ucedavélez and M. M. Morana, "Intro to pasta," in *Risk Centric Threat Modeling*. Hoboken, NJ, USA: Wiley, May 2015, pp. 317–342. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/9781118988374.ch6

[14] K. Wuyts, D. Van Landuyt, A. Hovsepyan, and W. Joosen, "Effective and efficient privacy threat modeling through domain refinements," in *Proc. 33rd Annu. ACM Symp. Appl. Comput.* New York, NY, USA: ACM, Apr. 2018, pp. 1175–1178, doi: 10.1145/3167132.3167414.

[15] B. Potteiger, G. Martins, and X. Koutsoukos, "Software and attack centric integrated threat modeling for quantitative risk assessment," in *Proc. Symp. Bootcamp Sci. Secur.* New York, NY, USA: ACM, Apr. 2016, pp. 99–108, doi: 10.1145/2898375.2898390.

[16] P. Saitta, B. Larcom, and M. M. Eddington, *Trike V.1 Methodology Document [Draft]*, 2005. [Online]. Available: https://www.octotrike.org/papers/Trike_v1_Methodology_Document-draft.pdf

[17] C. Alberts, A. Dorofee, J. Stevens, and C. Woody, "Introduction to the OCTAVE approach," Softw. Eng. Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, Aug. 2003. [Online]. Available: https://resources.sei.cmu.edu/asset_files/UsersGuide/2003_012_001_51556.pdf and https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=51546

[18] H. S. Lallie, K. Debattista, and J. Bal, "An empirical evaluation of the effectiveness of attack graphs and fault trees in Cyber-attack perception," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 5, pp. 1110–1122, May 2018. [Online]. Available: http://ieeexplore.ieee.org/document/8101532/

[19] A. AlSabeh, J. Khoury, E. Kfoury, J. Crichigno, and E. Bou-Harb, "A survey on security applications of P4 programmable switches and a STRIDE-based vulnerability assessment," *Comput. Netw.*, vol. 207, Apr. 2022, Art. no. 108800. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128622000287

[20] D. Sattar, A. H. Vasoukolaei, P. Crysdale, and A. Matrawy, "A stride threat model for 5G core slicing," in *Proc. IEEE 4th 5G World Forum (5GWF)*, Oct. 2021, pp. 247–252.

[21] FIRST. (2021). *FIRST Forum of Incident Response and Security Teams*. [Online]. Available: https://www.first.org/about/

[22] O. Flåten and M. S. Lund, "How good are attack trees for modelling advanced cyber threats?" in *Proc. Norwegian Inf. Secur. Conf.*, Fredrikstad, Norway, Nov. 2014. [Online]. Available: http://ojs.bibsys.no/index.php/NISK/article/view/105

[23] L. H. Fla, R. Borgaonkar, I. A. Tondel, and M. Gilje Jaatun, "Tool-assisted threat modeling for smart grid cyber security," in *Proc. Int. Conf. Cyber Situational Awareness, Data Analytics Assessment (CyberSA)*, Jun. 2021, pp. 1–8.

[24] J. Geismann, "Traceable Threat Modeling for Safety-Critical Systems," in *Proc. IEEE Int. Conf. Softw. Archit. Companion (ICSA-C)*, Apr. 2018, pp. 41–42. [Online]. Available: https://ieeexplore.ieee.org/document/8432172/

[25] W. Xiong and R. Lagerström, "Threat modeling—A systematic literature review," *Comput. Secur.*, vol. 84, pp. 53–69, Jul. 2019.

[26] M. Hamad and V. Prevelakis, "Savta: A hybrid vehicular threat model: Overview and case study," *Information*, vol. 11, no. 5, p. 273, 2020. [Online]. Available: https://www.mdpi.com/2078-2489/11/5/273

[27] N. Rashid, J. Wan, G. Quiros, A. Canedo, and M. A. Al Faruque, "Modeling and simulation of cyberattacks for resilient cyber-physical systems," in *Proc. 13th IEEE Conf. Autom. Sci. Eng. (CASE)*, Aug. 2017, pp. 988–993.

[28] T. Ghasempouri, J. Raik, C. Reinbrecht, S. Hamdioui, and M. Taouil, "Survey on architectural attacks: A unified classification and attack model," 2022, *arXiv:2208.14194*.

[29] S. Park and H. Park, "PIER: Cyber-resilient risk assessment model for connected and autonomous vehicles," *Wireless Netw.*, 2022. [Online]. Available: https://link.springer.com/article/10.1007/s11276-022-03084-9#citeas, doi: 10.1007/s11276-022-03084-9.

[30] H.-K. Kong, T.-S. Kim, and M.-K. Hong, "A security risk assessment framework for smart car," in *Proc. 10th Int. Conf. Innov. Mobile Internet Services Ubiquitous Comput. (IMIS)*, Jul. 2016, pp. 102–108.

[31] T. T. O. Takleh, N. A. Bakar, S. A. Rahman, R. Hamzah, and Z. A. Aziz, "A brief survey on SLAM methods in autonomous vehicle," *Int. J. Eng. Technol.*, vol. 7, no. 4.27, p. 38, Nov. 2018. [Online]. Available: https://www.sciencepubco.com/index.php/ijet/article/view/22477

[32] Á. Llamazares, E. J. Molinos, and M. Ocaña, "Detection and tracking of moving obstacles (DATMO): A review," *Robotica*, vol. 38, no. 5, pp. 761–774, May 2020.

[33] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, L. Jesus, R. Berriel, T. M. Paixão, F. Mutz, L. D. P. Veronese, T. Oliveira-Santos, A. Forechi, and A. F. D. Souza, "Self-driving cars: A survey," *Exp. Syst. Appl.*, vol. 165, Mar. 2021, Art. no. 113816. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741742030628X

[34] F. Munir, S. Azam, M. I. Hussain, A. M. Sheri, and M. Jeon, "Autonomous vehicle: The architecture aspect of self driving car," in *Proc. Int. Conf. Sensors, Signal Image Process. (SSIP)*. New York, NY, USA: Association for Computing Machinery, 2018, pp. 1–5, doi: 10.1145/3290589.3290599.

[35] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, M. Sokolsky, G. Stanek, D. Stavens, A. Teichman, M. Werling, and S. Thrun, "Towards fully autonomous driving: Systems and algorithms," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2011, pp. 163–168.

[36] T. Yamada, M. Sato, R. Kuranobu, R. Watanabe, H. Itoh, M. Shiokari, and T. Yuzui, "Evaluation of effectiveness of the STAMP/STPA in risk analysis of autonomous ship systems," *J. Phys., Conf.*, vol. 2311, no. 1, Jul. 2022, Art. no. 012021.

[37] T. Watanabe and M. Itoh, "Validity and considerations of the safety analysis method STAMP/STPA on emergency stop—Case: Unprecedented systems," in *Proc. 61st Annu. Conf. Soc. Instrum. Control Engineers (SICE)*, Sep. 2022, pp. 1194–1200.

[38] S. Karatzas and A. Chassiakos, "System-theoretic process analysis (STPA) for hazard analysis in complex systems: The case of 'demand-side management in a smart grid,'" *Systems*, vol. 8, no. 3, p. 33, 2020.

[39] Z. A. Biron, S. Dey, and P. Pisu, "Resilient control strategy under denial of service in connected vehicles," in *Proc. Amer. Control Conf. (ACC)*, May 2017, pp. 4971–4976.

[40] Y. Nakahira and Y. Mo, "Attack-resilient H_2, $H\infty$, and ℓ_1 state estimator," *IEEE Trans. Autom. Control*, vol. 63, no. 12, pp. 4353–4360, Dec. 2018.

[41] *Highway Accident Report NTSB/HAR-17/02: Collision Between a Car Operating With Automated Vehicle Control Systems and a Tractor-Semitrailer Truck Near Williston, Florida*, NTSB, Washingon, DC, USA, 2017. [Online]. Available: https://www.ntsb.gov/investigations/AccidentReports/Reports/HAR1702.pdf

[42] *Highway Accident Report NTSB/HAR-19/03: Collision Between Vehicle Controlled by Developmental Automated Driving System and Pedestrian*, Nat. Transp. Saf. Board, Washington, DC, USA, 2018. [Online]. Available: https://www.ntsb.gov/investigations/AccidentReports/Reports/HAR1903.pdf

[43] J. Cui, L. S. Liew, G. Sabaliauskaite, and F. Zhou, "A review on safety failures, security attacks, and available countermeasures for autonomous vehicles," *Ad Hoc Netw.*, vol. 90, Jul. 2019, Art. no. 101823.

[44] A. D. Kumar, K. N. R. Chebrolu, R. Vinayakumar, and K. P. Soman, "A brief survey on autonomous vehicle possible attacks, exploits and vulnerabilities," in *Proc. Comput. Vis. Pattern Recognit.*, 2018, pp. 1–5.

[45] S. Malik and W. Sun, "Analysis and simulation of cyber attacks against connected and autonomous vehicles," in *Proc. Int. Conf. Connected Auto. Driving (MetroCAD)*, Feb. 2020, pp. 62–70.

[46] K. Ren, Q. Wang, C. Wang, Z. Qin, and X. Lin, "The security of autonomous driving: Threats, defenses, and future directions," *Proc. IEEE*, vol. 108, no. 2, pp. 357–372, Feb. 2020.

[47] F. Sommer, J. Dürrwang, and R. Kriesten, "Survey and classification of automotive security attacks," *Information*, vol. 10, no. 4, p. 148, Apr. 2019.

[48] M. H. Eiza and Q. Ni, "Driving with sharks: Rethinking connected vehicles with vehicle cybersecurity," *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 45–51, Jun. 2017.

[49] R. G. Dutta, "Security of autonomous systems under physical attacks: With application to self-driving cars," M.S. thesis, Univ. Central Florida, 2018. [Online]. Available: https://stars.library.ucf.edu/etd/5957 and https://stars.library.ucf.edu/etd/5957/?utm_source=stars.library.ucf.edu%2Fetd%2F5957&utm_medium=PDF&utm_campaign=PDFCoverPages

[50] M. Dibaei, X. Zheng, K. Jiang, R. Abbas, S. Liu, Y. Zhang, Y. Xiang, and S. Yu, "Attacks and defences on intelligent connected vehicles: A survey," *Digit. Commun. Netw.*, vol. 6, no. 4, pp. 399–421, Nov. 2020.

[51] R. E. Haas and D. P. F. Moller, "Automotive connectivity, cyber attack scenarios and automotive cyber security," in *Proc. IEEE Int. Conf. Electro Inf. Technol. (EIT)*, May 2017, pp. 635–639.

[52] E. Yağdereli, C. Gemci, and A. Z. Aktaş, "A study on cyber-security of autonomous and unmanned vehicles," *J. Defense Model. Simul.*, vol. 12, no. 4, pp. 369–381, 2015, doi: 10.1177/1548512915575803.

[53] D. Lim, K. Park, D. Choi, and J. Seo, "Analysis on attack scenarios and countermeasures for self-driving car and its infrastructures," in *Advances on Broad-Band Wireless Computing, Communication and Applications*, L. Barolli, F. Xhafa, and K. Yim, Eds. Cham, Switzerland: Springer, 2017, pp. 429–442.

[54] S. Parkinson, P. Ward, K. Wilson, and J. Miller, "Cyber threats facing autonomous and connected vehicles: Future challenges," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 11, pp. 2898–2915, Nov. 2017.

[55] F. Sakiz and S. Sen, "A survey of attacks and detection mechanisms on intelligent transportation systems: VANETs and IoV," *Ad Hoc Netw.*, vol. 61, pp. 33–50, Jun. 2017.

[56] D. Mendes, N. Ivaki, and H. Madeira, "Effects of GPS spoofing on unmanned aerial vehicles," in *Proc. IEEE 23rd Pacific Rim Int. Symp. Dependable Comput. (PRDC)*, Dec. 2018, pp. 155–160.

[57] A. Couts. (2013). *Want to See This $80 Million Super Yacht Sink? With GPS Spoofing, Now You Can!*. Accessed: Nov. 17, 2020. [Online]. Available: https://www.digitaltrends.com/mobile/gps-spoofing/

[58] S. Dasgupta, T. Ghosh, and M. Rahman, "A reinforcement learning approach for GNSS spoofing attack detection of autonomous vehicles," 2021, *arXiv:2108.08628*.

[59] Q. Meng, L.-T. Hsu, B. Xu, X. Luo, and A. El-Mowafy, "A GPS spoofing generator using an open sourced vector tracking-based receiver," *Sensors*, vol. 19, no. 18, p. 3993, Sep. 2019.

[60] C. Yan, W. Xu, and J. Liu, "Can you trust autonomous vehicles: Contactless attacks against sensors of self-driving vehicle," in *Proc. 28th Modern Artif. Intell. Cognitive Sci. Conf. (MAICS)*, 2017, pp. 189–190.

[61] J. Petit, B. Stottelaar, M. Feiri, and F. Kargl. (2015). *Remote Attacks on Automated Vehicles Sensors: Experiments on Camera and LiDAR*. pp. 1–13. [Online]. Available: https://Blackhat.com

[62] H. Shin, D. Kim, Y. Kwon, and Y. Kim, "Illusion and dazzle: Adversarial optical channel exploits against LiDARS for automotive applications," in *Proc. Int. Conf. Cryptograph. Hardw. Embedded Syst.* Springer, 2017, pp. 445–467. [Online]. Available: https://ches.iacr.org/2017/ and https://eprint.iacr.org/2017/613.pdf and https://eprint.iacr.org/2017/613

[63] Y. Cao, C. Xiao, B. Cyr, Y. Zhou, W. Park, S. Rampazzi, Q. A. Chen, K. Fu, and Z. M. Mao, "Adversarial sensor attack on LiDAR-based perception in autonomous driving," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.* New York, NY, USA: ACM, Nov. 2019, pp. 2267–2281.

[64] D. Nassi, R. Ben-Netanel, Y. Elovici, and B. Nassi, "MobilBye: Attacking ADAS with camera spoofing," 2019, *arXiv:1906.09765*.

[65] J. Sun, Y. Cao, Q. A. Chen, and Z. M. Mao, "Towards robust LiDAR-based perception in autonomous driving: General black-box adversarial sensor attack and countermeasures," in *Proc. 29th USENIX Secur. Symp. (USENIX Security)*, 2020, pp. 877–894.

[66] D. Davidson, H. Wu, R. Jellinek, V. Singh, and T. Ristenpart, "Controlling UAVs with sensor input spoofing attacks," in *Proc. 10th USENIX Workshop Offensive Technol. (WOOT)*, 2016, pp. 1–11.

[67] K. Yang, R. Wang, Y. Jiang, H. Song, C. Luo, Y. Guan, X. Li, and Z. Shi, "Sensor attack detection using history based pairwise inconsistency," *Future Gener. Comput. Syst.*, vol. 86, pp. 392–402, Sep. 2018.

[68] H. Ji, Y. Wang, H. Qin, X. Wu, and G. Yu, "Investigating the effects of attack detection for in-vehicle networks based on clock drift of ECUs," *IEEE Access*, vol. 6, pp. 49375–49384, 2018.

[69] Z. Goldberg, "Jamming and spoofing attacks: Physical layer cybersecurity threats to autonomous vehicle systems," Nat. Highway Traffic Saf. Admin., Washington, DC, USA, Tech. Rep., 2016. [Online]. Available: https://tlpc.colorado.edu/wp-content/uploads/2016/11/2016.11.21-Autonomous-Vehicle-Jamming-and-Spoofing-Comment-Final.pdf

[70] B. S. Lim, S. L. Keoh, and V. L. L. Thing, "Autonomous vehicle ultrasonic sensor vulnerability and impact assessment," in *Proc. IEEE 4th World Forum Internet Things (WF-IoT)*, Feb. 2018, pp. 231–236.

[71] G. Kar, H. Mustafa, Y. Wang, Y. Chen, W. Xu, M. Gruteser, and T. Vu, "Detection of on-road vehicles emanating GPS interference," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2014, pp. 621–632.

[72] Y. Son, H. Shin, D. Kim, Y. Park, J. Noh, K. Choi, J. Choi, and Y. Kim, "Rocking drones with intentional sound noise on gyroscopic sensors," in *Proc. 24th USENIX Secur. Symp. (USENIX Security)*, 2015, pp. 881–896.

[73] Z. Zorz. (2018). *Researchers Hack BMW Cars, Discover 14 Vulnerabilities*. [Online]. Available: https://www.helpnetsecurity.com/2018/05/23/hack-bmw-cars/

[74] P. Carsten, T. R. Andel, M. Yampolskiy, and J. T. McDonald, "In-vehicle networks: Attacks, vulnerabilities, and proposed solutions," in *Proc. 10th Annu. Cyber Inf. Secur. Res. Conf.*, vols. 06–08, 2015, pp. 1–8.

[75] Z. Cai, A. Wang, W. Zhang, M. Gruffke, and H. Schweppe, "0-days & mitigations: Roadways to exploit and secure connected BMW cars," *Black Hat USA*, vol. 2019, p. 39, Aug. 2019.

[76] S. Nie, L. Liu, and Y. Du, "Free-fall: Hacking Tesla from wireless to CAN bus," *Briefing, Black Hat USA*, vol. 25, pp. 1–16, 2017.

[77] V. L. Thing and J. Wu, "Autonomous vehicle security: A taxonomy of attacks and defences," in *Proc. IEEE Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCom) IEEE Smart Data (SmartData)*, Dec. 2016, pp. 164–170.

[78] A. Tomlinson, J. Bryans, S. A. Shaikh, and H. K. Kalutarage, "Detection of automotive can cyber-attacks by identifying packet timing anomalies in time Windows," in *Proc. 48th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. Workshops (DSN-W)*, 2018, pp. 231–238.
[79] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," Tech. Rep., 2014.
[80] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. 3rd Int. Conf. Learn. Represent.(ICLR)*, 2015, pp. 1–11.
[81] J. Petit and W. Whyte, "Future threats to connected and automated vehicles," in *Road Vehicle Automation 8. AVS 2020* (Lecture Notes in Mobility), G. Meyer and S. Beiker, Eds. Cham, Switzerland: Springer, 2022. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-80063-5_8#citeas, doi: 10.1007/978-3-030-80063-5_8.
[82] A. Boloor, K. Garimella, X. He, C. Gill, Y. Vorobeychik, and X. Zhang, "Attacking vision-based perception in end-to-end autonomous driving models," *J. Syst. Archit.*, vol. 110, Nov. 2020, Art. no. 101766.
[83] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1625–1634.
[84] C. Sitawarin, A. Nitin Bhagoji, A. Mosenia, M. Chiang, and P. Mittal, "DARTS: Deceiving autonomous cars with toxic signs," 2018, *arXiv:1802.06430*.
[85] T. Wu, X. Ning, W. Li, R. Huang, H. Yang, and Y. Wang, "Physical adversarial attack on vehicle detector in the carla simulator," 2020, *arXiv:2007.16118*.
[86] Y. Xu, X. Han, G. Deng, G. Li, Y. Liu, J. Li, and T. Zhang, "SoK: Rethinking sensor spoofing attacks against robotic vehicles from a systematic view," 2022, *arXiv:2205.04662*.
[87] M. T. Arafin and K. Kornegay, "Attack detection and countermeasures for autonomous navigation," in *Proc. 55th Annu. Conf. Inf. Sci. Syst. (CISS)*, Mar. 2021, pp. 1–6.
[88] Y. Lin, H. Zhao, X. Ma, Y. Tu, and M. Wang, "Adversarial attacks in modulation recognition with convolutional neural networks," *IEEE Trans. Rel.*, vol. 70, no. 1, pp. 389–401, Mar. 2021.
[89] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, pp. 1–91, Aug. 2015.
[90] B. Schneier, "Attack trees," *Dr. Dobb's J.*, vol. 24, no. 12, pp. 21–29, 1999.
[91] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous localization and mapping: A survey of current trends in autonomous driving," *IEEE Trans. Intell. Veh.*, vol. 2, no. 3, pp. 194–220, Sep. 2017.
[92] E. Horton and P. Ranganathan, "Development of a GPS spoofing apparatus to attack a DJI matrice 100 quadcopter," *J. Global Positioning Syst.*, vol. 16, no. 1, pp. 1–11, Dec. 2018.
[93] TM Corporation. (2021). *CWE: Common Weakness Enumeration*. [Online]. Available: https://cwe.mitre.org/index.html
[94] X. Wu and J. Wu, "The risk priority number evaluation of FMEA analysis based on random uncertainty and fuzzy uncertainty," *Complexity*, vol. 2021, pp. 1–15, Feb. 2021.
[95] The Pandas Development Team. (Feb. 2020). *Pandas-Dev/Pandas: Pandas*. [Online]. Available: https://doi.org/10.5281/zenodo.3509134
[96] W. McKinney, "Data Structures for Statistical Computing in Python," in *Proc. 9th Python Sci. Conf.*, S. van der Walt and J. Millman, Eds. 2010, pp. 56–61.

JUNAID M. QURASHI received the B.C.A. degree from Kashmir University, Jammu and Kashmir, India, in 2012, and the master's degree in information technology from International Islamic University Malaysia, Selangor, Malaysia, in 2016. His work experience includes working as a Technical Trainer and a Data Analyst at Sellbytel (now Webhelp) for Cisco, NetApp, and HP clients. He is currently associated with the Faculty of Computing and Information Technology, King Abdulaziz University. His research interests include cyber-physical systems, resilience, cybersecurity, artificial intelligence, machine learning, the Internet of Things, and adoption of information systems.

KAMAL JAMBI received the M.S. degree in computer science from Michigan State University, East Lansing, MI, USA, in 1986, and the Ph.D. degree in computer science from the Illinois Institute of Technology, Chicago, IL, USA, in 1991. He was the Former Vice Dean of Graduate Studies and Scientific Research at FCIT and the Chairperson of the Computer Science Department. He is currently a Professor of computer science at King Abdulaziz University, Jeddah, Saudi Arabia. He was a PI on many research projects at KACST and KAU. His research interests include AI, deep learning, blockchain, resilience, and big data.

FATHY E. EASSA received the B.Sc. degree in electronics and electrical communication engineering from Cairo University, Egypt, in 1978, and the M.Sc. and Ph.D. degrees in computers and systems engineering from Al-Azhar University, Cairo, Egypt, in 1984 and 1989, respectively, with joint supervision from the University of Colorado at Boulder, Boulder, CO, USA. He is currently a Full Professor at the Computer Science Department, Faculty of Computing and Information Technology, King Abdulaziz University, Saudi Arabia. His research interests include agent-based software engineering, cloud computing, software engineering, big data, distributed systems, exa-scale system testing performance analysis, network security, and pattern recognition.

MAHER KHEMAKHEM received the M.Sc. and Ph.D. degrees from Paris-Sud University, France, in 1984 and 1987, respectively, and the Habilitation degree from the University of Sfax, Tunisia, in 2008. He is currently a Full Professor of computer science at the Faculty of Computing and Information Technology, King Abdulaziz University, Saudi Arabia. His research interests include distributed systems, HPC, performance analysis, network security, and pattern recognition.

FAWAZ ALSOLAMI received the M.A.Sc. degree in electrical and computer engineering from the University of Waterloo, Canada, in 2008, and the Ph.D. degree in computer science from the King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia, in 2016. He joined King Abdulaziz University, Saudi Arabia, as an Assistant Professor of computer science, where he has been the Chairperson of the Computer Science Department, since 2018. He has published many articles and one monograph. His research interests include artificial intelligence, machine learning, data mining, and combinatorial optimization.

ABDULLAH BASUHAIL is currently a Professor of computer engineering and sciences at the Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia. His research interests include digital image processing, computer vision, cyber security, and e-learning. He was a member of the IEEE Computer Society and the Saudi Computer Society. He is a member of the Saudi OER Network (Shms).