applied sciences

MDPI

Article

Reinforcement Learning as a Path to Autonomous Intelligent Cyber-Defense Agents in Vehicle Platforms

Stephen Raio [1,*], Kevin Corder [2], Travis W. Parker [3], Gregory G. Shearer [3], Joshua S. Edwards [3], Manik R. Thogaripally [3], Song J. Park [1] and Frederica F. Nelson [1]

[1] U.S. Army Combat Capabilities Development Command (DEVCOM) Army Research Laboratory, Aberdeen Proving Ground, Aberdeen, MD 21005, USA; song.j.park.civ@army.mil (S.J.P.); frederica.f.nelson.civ@army.mil (F.F.N.)
[2] Parsons, Aberdeen Proving Ground, Aberdeen, MD 21005, USA; kevin.m.corder.ctr@army.mil
[3] ICF International, Columbia, MD 21046, USA; travis.w.parker16.ctr@army.mil (T.W.P.); gregory.g.shearer.ctr@army.mil (G.G.S.); manik.r.thogaripally.ctr@army.mil (M.R.T.)
* Correspondence: stephen.raio.civ@army.mil

Abstract: Technological advancement of vehicle platforms exposes opportunities for new attack paths and vulnerabilities. Static cyber defenses can help mitigate certain attacks, but those attacks must generally be known ahead of time, and the cyber defenses must be hand-crafted by experts. This research explores reinforcement learning (RL) as a path to achieve autonomous, intelligent cyber defense of vehicle control networks—namely, the controller area network (CAN) bus. We train an RL agent for the CAN bus using Toyota's Portable Automotive Security Testbed with Adaptability (PASTA). We then apply the U.S. Army Combat Capabilities Development Command (DEVCOM) Army Research Laboratory's methodology for quantitative measurement of cyber resilience to assess the agent's effect on the vehicle testbed in a contested cyberspace environment. Despite all defenses having similar traditional performance measures, our RL agent averaged a 90% cyber resilience measurement during drive cycles executed on hardware versus 41% for a naïve static timing defense and 98% for the bespoke timing-based defense. Our results also show that an RL-based agent can detect and block injection attacks on a vehicle CAN bus in a laboratory environment with greater cyber resilience than prior learning approaches (1% for convolutional networks and 0% for recurrent networks). With further research, we believe there is potential for using RL in the autonomous intelligent cyber defense agent concept.

Keywords: cybersecurity; vehicle; CAN bus; reinforcement learning; autonomous; cyber resilience

Citation: Raio, S.; Corder, K.; Parker, T.W.; Shearer, G.G.; Edwards, J.S.; Thogaripally, M.R.; Park, S.J.; Nelson, F.F. Reinforcement Learning as a Path to Autonomous Intelligent Cyber-Defense Agents in Vehicle Platforms. *Appl. Sci.* **2023**, *13*, 11621. https://doi.org/10.3390/app132111621

Academic Editor: Luis Javier García Villalba

Received: 18 August 2023
Revised: 17 October 2023
Accepted: 19 October 2023
Published: 24 October 2023

1. Introduction

The technological advancement of vehicle platforms—including electronic control units (ECUs) and their firmware, intra-platform communication on control buses between ECUs, and other advancements necessary for future platform autonomization—exposes opportunities for new attack paths and vulnerabilities. This research aims to explore concepts and technologies needed to enable autonomous, active cyber defense of vehicle platforms for survival and recovery from cyberattacks in contested cyberspace environments (CCEs).

Vehicle platforms, being cyber-physical systems, are not currently resilient in a CCE [1]. Cyber resilience is the ability of systems to recover from cyber stress that causes a reduction in performance [2]. Essentially, how well can the platform continue to perform its mission in a CCE? As humans alone cannot defend all vehicle platforms in real-time, it is necessary to develop software able to defend against novel cyberattacks during operation in a CCE.

We propose using reinforcement learning (RL) as an approach for cyber-resilient autonomous intelligent cyber-defense agents (AICAs) [3,4] in vehicle platforms. The software agent is trained to defend a controller area network (CAN) bus from malicious messages. We evaluate the agent using traditional classification measurements such as accuracy, as

well as the U.S. Army Combat Capabilities Development Command (DEVCOM) Army Research Laboratory's proposed quantitative measurement of cyber resilience (QMoCR) [5] (see Section 3.6). QMoCR is calculated using the difference in area under the curve of a measured key performance parameter (KPP); in our case, this performance measure is vehicle speed.

Our contributions are as follows. While RL has been suggested and used in automotive applications, to the best of our knowledge, we are the first to use RL for detecting and preventing CAN bus injection attacks at the network layer. Our feature engineering approach for the RL agent reformulates the observation space from an action-independent multi-armed bandit task into an action-dependent Markov decision process. We argue that this impact on the CAN bus state justifies the use of RL methods over supervised machine learning (ML) methods such as classification, and our results show that RL is superior with and without this feature engineering. Additionally, we use QMoCR to evaluate cyber resilience in Toyota's Portable Automotive Security Testbed with Adaptability (PASTA), which gives a better picture of real-world effects vs. using simple receiver operating characteristic curves or loss metrics.

2. Related Work

In this section, we review related works to our study. Intrusion detection methods have ranged from simple lookup tables [6] to time-series ML models [7]. Non-ML approaches, which focus on static defense methods like lookup tables, are computationally simple enough to run on the limited resources available on CAN bus hardware. Several papers have shown that with time series data alone, it is possible to accurately defend fabrication and suspension attacks [8]. Several others have found vulnerabilities in such simplistic methods using masquerade [9] or cloaking attacks [8]. These attacks mimic or alter the timing between the messages to attempt to bypass the defense.

ML models use computationally intensive training algorithms to maximize performance and typically increase resiliency to attacks. Many previous works have investigated intrusion detection systems for the automotive CAN bus using ML [10]. Most existing ML work frames this problem as a supervised classification task with methods like random forests [11], extreme gradient boosting (XGBoost) [12,13], support vector machines [14], and deep learning architectures such as multilayer perceptrons (MLPs) [10,15], convolutional neural networks (CNNs) [16], long short-term memory (LSTM) networks [17], and generative adversarial networks (GANs) [18]. Recurrent networks are also common among related tasks, such as online purchase error classification [19]. In contrast to classification, an RL approach maximizes long-term success through a reward function, explicitly choosing actions while allowing more specific tuning for desired behavior.

Other related works in intrusion detection for a vehicle network primarily focus on the classification task and do not show how the system recovers [20–22]. In this paper, we develop an agent that detects attacks, responds by taking actions, and evaluates its performance using the aforementioned QMoCR measurement.

Previously, RL has been applied to cybersecurity [23] and intrusion detection systems [24]. For anomalous network traffic, Lopez-Martin et al. [25] used deep RL for intrusion detection using Network Security Laboratory Knowledge Discovery and Data Mining (NSL-KDD) [26] and Aegean WiFi Intrusion Dataset (AWID) [27] datasets. In contrast, our work investigates using RL for maintaining cyber resiliency in a vehicle CAN bus.

Similar to our work, Xiao et al. [28] explore RL for physical layer authentication in a CAN bus. However, this work focuses on analog physical layer characteristics of the message sender's transceiver and does not protect against malicious trusted ECUs.

3. Methods

In this section, we discuss the autonomous intelligent cyber-defense agent concept, the experimentation environment, and the methods used in our research. Specifically, we

describe the hardware and software platforms, agent design, feature engineering, training conditions, defenses for comparison, and the QMoCR measurement used for assessment.

3.1. Overview

Our long-term hypothesis is that by leveraging the AICA concept, we can improve the cyber resilience of systems over existing cyber-defense strategies. To begin testing this hypothesis, we use the commercially available Toyota PASTA [29] as our testbed. We train an autonomous cyber-defense agent using RL and place the agent strategically within the vehicle network so that the agent can control traffic flow through the network. The agent is trained to maximize expected future returns under various environmental conditions. We measure the difference in cyber resilience through the experimentation and quantitative measurement processes developed under the DEVCOM Army Research Laboratory's Quantitative Measurement of Cyber-Resilience effort [5].

3.2. AICA Concept

The North Atlantic Treaty Organization (NATO) task group on "Intelligent, Autonomous and Trusted Agents for Cyber Defense and Resilience" outlined an AICA architecture [3,4]. Members of this group envisioned a future where both defensive and offensive autonomous agents battle for supremacy in cyberspace. Figure 1 illustrates their proposed architecture for an AICA and which parts of that architecture are represented in this research. As the AICA setup is nearly identical to the agent-environment interaction loop of RL, it seems natural to apply this approach to the problem.

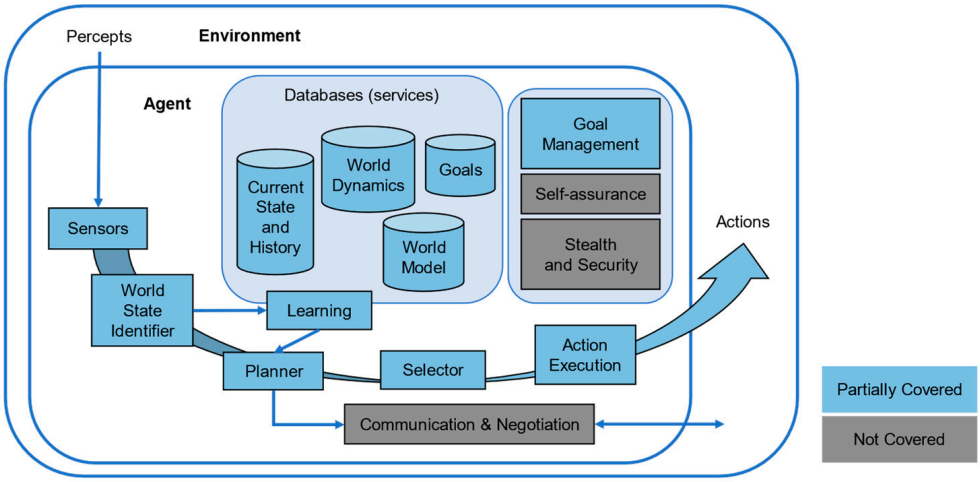

Figure 1. The autonomous intelligent cyber-defense agent reference architecture, as used in our prototype, is color-coded to show which pieces are represented.

The outer bounding box represents an environment. For our use case, the environment is the intra-vehicle control network. The inner bounding box is the agent itself, consisting of software and the required hardware to run it, operating within that environment. Using sensors, the agent receives observations to build a view of the environment state based on relevant parameters that the agent has access to. In our case, this is direct access to the decoded CAN bus messages in the form of a two-dimensional (2D) grayscale image. In the AICA architecture, these data are stored in databases and used for various services, e.g., to keep track of current and historical states or world models to help with planning and learning. Rather than explicit planning via world models, model-free RL algorithms inherently incorporate much of this functionality within the trained agent.

The goal of a vehicle platform AICA is to maintain the greatest degree of cyber resilience for a vehicle platform. In this context, cyber resilience refers to the vehicle's ability to continue operation and achieve its KPPs while under cyberattack. From a vehicle control network standpoint, this equates to assuring the control network remains available

for communication, legitimate nodes communicate their legitimate messages successfully, and illegitimate messages are blocked.

When the state of the world is different than the desired state, the agent must generate plans for attaining the desired state, select a plan, and execute it with feedback. An autonomous intelligent agent can adapt its response over time to attain the desired state more efficiently and effectively. This intelligent and goal-oriented action also allows it to respond to novel attacks in novel ways, unlike manually configured automated defenses. While this is an important long-term goal of this research area, we currently limit our research to initial training and assessment of an AICA without considering the additional desired capabilities obtained through online learning.

AICA components for inter-agent communication and negotiation, stealth and security, and self-assurance are marked in gray in Figure 1. These are important to the overarching AICA concept in which agents collaborate, potentially with competing goals, to thwart offensive agents. These components are beyond the scope of this research.

3.3. Agent Environment

We begin our exploration of the AICA concept within the CAN bus. To experiment in a controlled environment, we use the Toyota PASTA [29] testbed. PASTA consists of three ECUs and a gateway, all connected via CAN bus using an open-source message format. The vehicle physics (e.g., engine revolutions per minute, vehicle speed, coolant temperature) are simulated in a closed-source manner.

Modifications to the testbed layout were necessary for experimentation (see Figure 2). The first modification replaced the physical driver controls with a software-driven serial interface over a universal serial bus (USB) so that a programmed drive cycle can be consistently run for evaluation and to allow for random driver command generation during training. The programmed drive cycle also incorporates proportional-integral-derivative (PID) controllers for certain functions, such as maintaining a set speed and steering angle. The use of PID controllers to simulate human driving allows us to measure a more realistic impact on vehicle KPPs, such as vehicle speed, while the system is under attack. The other modification is to allow for defense agent interception of Chassis ECU generated messages, attack traffic injection, and logging along the Chassis CAN bus segment.

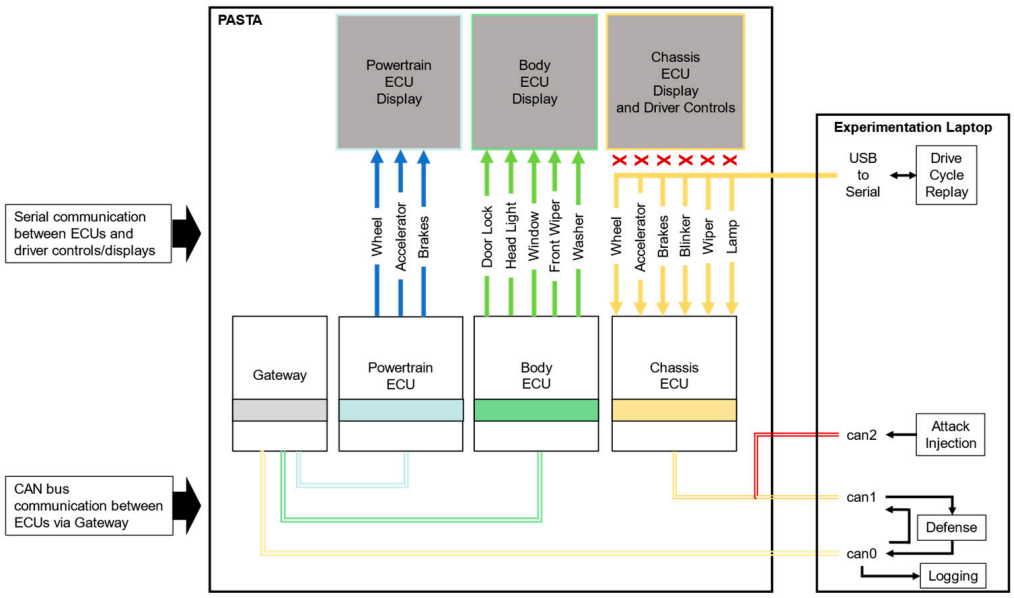

Figure 2. Hardware PASTA experimentation environment.

The defense agent is placed between the Chassis ECU and the Gateway, providing defense for outgoing messages from the Chassis ECU to the rest of the vehicle. Incoming

Chassis ECU messages from elsewhere bypass the defense. Details on agent design and training are provided in Section 3.4.

There are three fundamental attack methods on a CAN bus at the open systems interconnection (OSI) model equivalent to the network layer. These are injecting a message, blocking a message, or masquerading, which is essentially coordinated blocking and injecting. For simplicity, we only consider the injection attack method. We do not consider physical and data link layer attacks in this work.

3.4. Agent Design and Training Using Reinforcement Learning

RL inherently includes many of the AICA components, as discussed in Section 3.2. In an ideal world, with vast amounts of training data and model refinement, RL should be able to achieve strong policy optimization that combines the effectiveness of various static algorithms with an understanding of the typical behavior of the contextual environment and provides at least some ability to succeed in novel situations. Combined with eventual online learning capabilities, it should be possible to realize the full AICA vision.

An RL agent is defined as a mapping $\pi : S \mapsto A$, called a *policy*, from state s to action a. The policy is defined on an environment represented by a Markov decision process (MDP) $\langle S, A, P, R \rangle$ with state space S, action space A, transition function $P(s, a, s') = \Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$, and reward function $R(s, s')$. The transition function P describes the environment's probability of transitioning from state s to s' when the action a is chosen. The reward function R provides an immediate scalar value to the agent when transitioning from state s to s'. The optimization objective for the RL agent is then to maximize the expected *return*, the cumulative discounted reward across the (potentially infinite) time horizon. The expected return is expressed as $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, s_{t+1}; a_t)\right]$, with action $a_t = \pi(s_t)$ and discount factor $\gamma \in (0, 1]$. See [30] for more foundational information on RL.

In our problem setup, the RL agent observes a fixed-length history of messages as a 2D image (see Figure 3). The image transformation is particularly useful, so we may use a CNN architecture in the agent's value and policy networks, as in [31]. Specifically, each row encodes a single Chassis ECU emanated message with a single pixel representing the one-hot encoded message type, with 15 types in total. The value of the pixel is linearly scaled from 128 to 255 according to the time delta since the last message, computed by $128 \cdot \left(1 + \frac{0.001 - \delta}{0.001}\right)$ up to a maximum time δ of 0.001 s = 1 ms represented by a value of 128. If no message is received within 1 ms, a blank row is inserted. The decoded message values are represented as linearly scaled values between 0 and 255. Since CAN messages can have multiple data fields, we allowed for representing up to 8 decoded values per message. Following the decoded messages, we encode the data deltas between the current and previous messages of this type with a ceiling of 255. Message history is encoded in the row dimension, and we use a history of 100 messages. The evaluation image—used solely for human understanding of the learning process—includes an agent action column indicating whether the model decided to block or allow, with color coding indicating decision correctness.

The action space A is a set of two actions: allow or block the incoming message. The transition function P receives the action from the agent, transitions the environment to a new state and gives the agent a new observation. Specifically, if the message is blocked, then it is discarded from the message history going forward. This paradigm is called *observation dropping* in our results and tends to outperform including these messages in history. Importantly, observation dropping allows the agent to affect its observation space, which converts this classification task into a true sequential decision-making problem for which RL algorithms are suited.

If evaluating with hardware in the loop, an allowed message will be emitted to the vehicle bus. The next message will then be received by the environment, and the observation will be updated with the message type, content, and timing information. During training, there is a probability p of the next message being malicious. These

malicious messages are injected into the stream of messages with the intent to cause improper vehicle operation, but they do not modify or replace any legitimate messages. The environment determines if a message is malicious by a signature detected in the message data padding. Note that no indication of malice is provided to the agent in the observation. It is only used in the reward function.

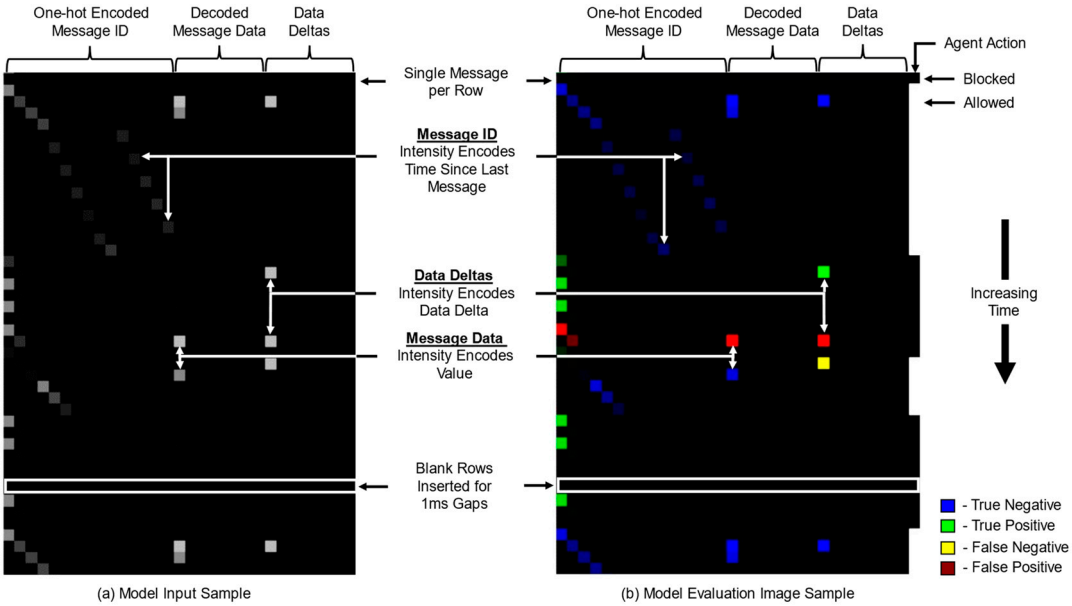

Figure 3. (**a**) Sample grayscale image used as input to defense agent. Each row contains up to 10 highlighted gray pixels that encode the message ID, time delta since the last message of the same ID, message data, and message data delta since the last message of the same ID. (**b**) Corresponding color representation for human evaluation of the agent learning process during training.

The reward function R evaluates if the action taken by the agent was correct for each message and provides a scalar feedback signal to train on. As the training and evaluation environments have ground truth knowledge of message tampering, the reward function simply compares the action taken and tampering status to determine the reward. Table 1 shows a summary of the reward function. These rewards were chosen empirically as equal rewards or larger rewards for correct decisions that did not result in learning an effective policy at lower attack injection rates.

Table 1. Reward function for allowing and blocking real and malicious messages.

	Allow	**Block**
Real	1.0	−1.0
Malicious	−1.5	1.5

We trained an Advantage Actor–Critic (A2C) agent [32] using ~17 h of data collected from the hardware PASTA environment Chassis CAN bus. Driver actions were randomly generated via software during the capture. The drive cycle generator communicates with the PASTA Chassis ECU via serial connection and emulates control inputs. At random intervals between 10 and 60 s, a new speed between 0 and 100 kph, a forward or reverse direction, and a steering wheel position were randomly selected. Other vehicle controls, such as turn signals, headlights, and wipers, are operated randomly or in connection with drive cycle actions. For example, the appropriate turn signal is activated if the steering wheel is turned more than 33% from straight.

For training, the capture was replayed along with random injection attacks of the 15 relevant Chassis ECU messages. Every 5 ms, there is a probability p of Chassis ECU

message type m being injected. A new p between 25% and 100% is randomly chosen for each message type m selected for injection at random intervals between 1 and 10 s. Message injections occur according to the probabilities in Table 2, and the injected message types are selected according to the probabilities in Table 3. Attack messages were padded with specific values so that the accuracy of the agent actions could be measured in terms of true positive rate and true negative rate.

Table 2. Message injection selection probabilities during training.

Selection Option	Option Probability
No injections	5%
Single message type injection	22%
Add message type to prior message injection list	73%

Table 3. Message type injection probabilities during training.

Message Type	Type Probability
Brake, acceleration, steering	~15%
Remaining 12 message types	~5%

3.5. Defenses for Comparison

For comparison to existing anomaly detection techniques, we implemented a detector solely using message timing (defense-static-timing-only), a variant additionally using a simple frequency analysis step, an LSTM-based detector CANet [33] and a custom CNN binary classifier with an identical network architecture to our proposed RL approach. The frequency analysis in the message timing variant allows a limited number of repetitions (up to 2) of the same message type–value pair occurring in the correct time window since the last allowed message (defense-static-timing-frequency). To choose appropriate timing windows, we analyzed the message time deltas for each message frequency (e.g., 20 ms, 50 ms) occurring within the collected training data (see Figure 4). The chosen timing windows for the static timing-based defenses are as follows: ±5% for 0x01A, 0x02F, and all 100 ms messages; ±10% for 0x058, 0x06D, 0x083, 0x098, and all 50 ms messages. Note that the static timing defense requires exact knowledge of the expected timing for each message type; this must be hard-coded for each individual system and cannot detect masquerading attacks whatsoever.

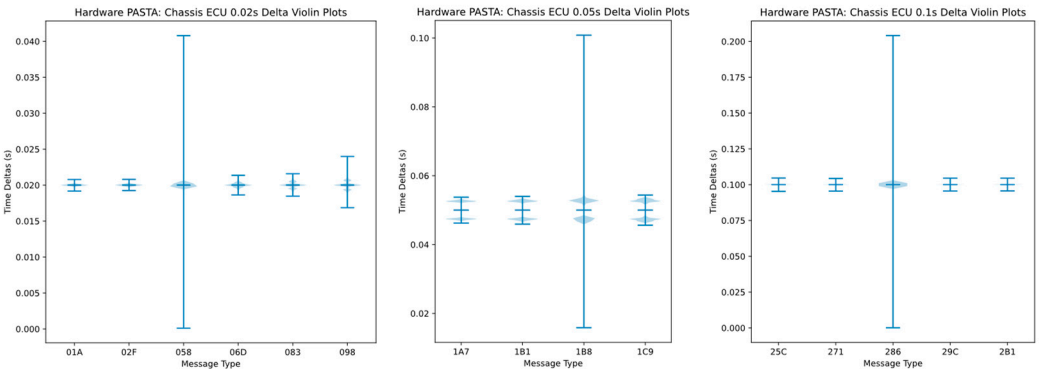

Figure 4. Violin plots of the time deltas for all Chassis ECU emanated messages. Message types are in hexadecimal.

CANet [33] is an LSTM- and autoencoder-based anomaly detection algorithm for CAN bus data. The model is trained on a dataset with no injections, and messages with reconstruction errors exceeding a threshold are deemed anomalous. Each message is fed to a separate LSTM encoder corresponding to its message type. The latent output from each LSTM is then concatenated for a temporal representation vector of the message history. This vector is then fed to a shared three-layer MLP network with exponential linear unit activation functions, which outputs the full message space. The mean squared error reconstruction loss is weighted by the message type frequency in the dataset to counteract bias. During evaluation with injections, a threshold loss value of 0.0001 was chosen for anomaly detection.

To ablate our use of RL in place of the usual supervised learning techniques for this problem, we implemented a binary classification model with identical CNN architecture and relevant hyperparameters. Specifically, this network consists of three 2D convolution layers with rectified linear unit activation functions, followed by a single fully connected layer with one output node and sigmoid activation to map into (0, 1) output logits. The CNN classifier is trained with binary cross-entropy loss between the predicted and true message labels. By changing as little as possible from our approach to this classifier, we hope to highlight the benefits of RL.

3.6. Quantitative Measurement of Cyber Resilience

Measurement is paramount in any scientific endeavor. The DEVCOM Army Research Laboratory has researched the meaning of cyber resilience and offered a definition suitable for military systems [2], a quantitative measurement of cyber resilience in this context [5], and a methodology for obtaining that measurement [34]. To summarize, the QMoCR measurement is the area-under-curve (AUC) ratio between baseline system performance and the system's performance under attack. For this research, we define our KPP as attaining and maintaining certain vehicle speed thresholds as seen in the baseline case (i.e., defense off, attack off). Therefore, we calculate AUC using the trapezoidal rule on the graph of time versus vehicle speed. We compare the baseline performance to the system's performance while under cyberattack and protected by the various defenses.

4. Results and Discussion

Evaluation of the trained models on the hardware PASTA testbed (see Figure 2) used a single drive cycle consisting of three 30 s periods of acceleration to, and holding of certain speeds (i.e., 50 kph, 100 kph, and 199 kph) separated by 10 s stopped intervals. Since we use vehicle speed as our performance parameter, attack messages consist of injection attacks targeting the accelerator and brake pedal inputs from the driver. The attack consists of simultaneously injecting 0x01A messages (brake_operation_indicator) with a value of 100% and 0x02F messages (accelerator_pedal_operation_indicator) with a value of 0% at 5 ms intervals. We perform ~15 runs with each type of defense and average the results to obtain the final QMoCR measurement. For each learning approach, five independent models were initialized and trained, and their evaluation results were averaged.

The performance of each defense is visualized in Figures 5 and 6, and average performance measurements are provided in Table 4. While we do not have sufficient samples to quantify the correlation between traditional evaluation measurements and real-world performance, as measured by QMoCR and depicted in Figure 6, the relationship does not appear to scale linearly. Although all classifiers achieve accuracy above ~70%, the timing of allowed attack messages (i.e., false negatives) in relation to legitimate control messages (i.e., true negatives) is also important to overall system performance.

Without defense, the vehicle is undrivable when under attack (Figure 6a). All defenses performed in line with their accuracy with regard to vehicle speed when not subjected to an attack (not illustrated), resulting in no distinguishable degradation in measured performance over the baseline run for defenses with accuracy above 90%. This is notable because there is potential for impact on the vehicle caused simply by the addition of the

defense without any attack present. In fact, there would be noticeable degradation had we focused on vehicle functionality enabled by brief changes in CAN bus message data, such as on/off activation messages. In those cases, large impacts to vehicle functionality can occur even with few but critically timed false positives that block the intended control signal.

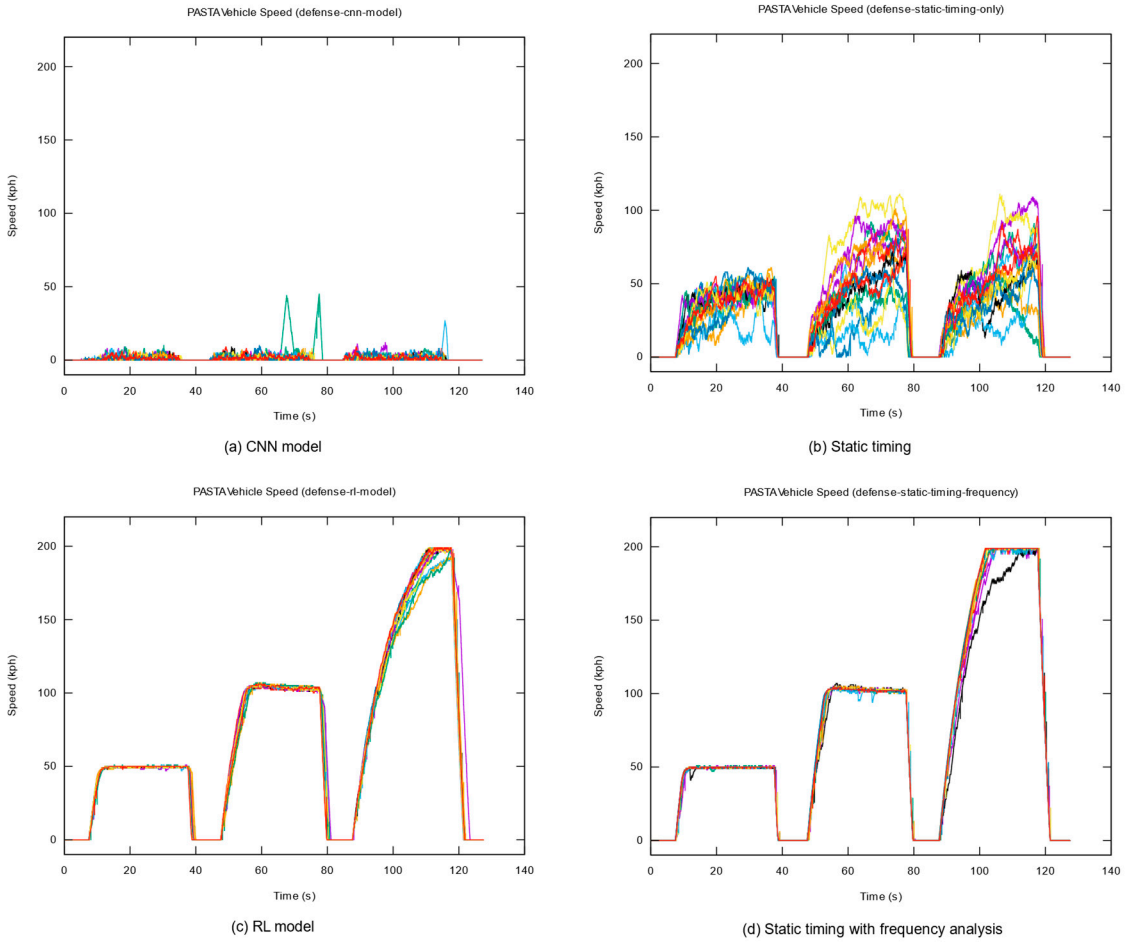

Figure 5. Plots of the multiple evaluations run on PASTA hardware per defense while under attack. LSTM is not shown as it was unable to complete a drive cycle due to blocking transmission shifting messages. Colored lines represent unique trials and are plotted to show the variability of each method. The RL model (**c**) appears as reliable as the hand-crafted oracle method (**d**).

Table 4. Performance measurements of defenses using a drive cycle on the PASTA testbed.

Defense	QMoCR Measurement		Traditional Performance Measures					
	AUC	Defense/Baseline AUC Ratio	Accuracy	Precision	Recall	F-Score	Sensitivity	Specificity
Baseline	9615	...	...	...	...	...	...	...
No Defense	4	0%	...	...	...	...	...	...
Static Timing	3976	41%	0.9325	0.9251	0.9300	0.9275	0.9300	0.9346
RL Model	8606	90%	0.9341	0.943	0.913	0.9277	0.913	0.9523
Static Timing w/Frequency	9430	98%	0.9433	0.9345	0.9443	0.9393	0.9443	0.9425
CNN Model	109	1%	0.7009	0.7592	0.518	0.6158	0.518	0.8585
LSTM Model	0	0%	0.6995	0.7697	0.4998	0.6061	0.4998	0.8714

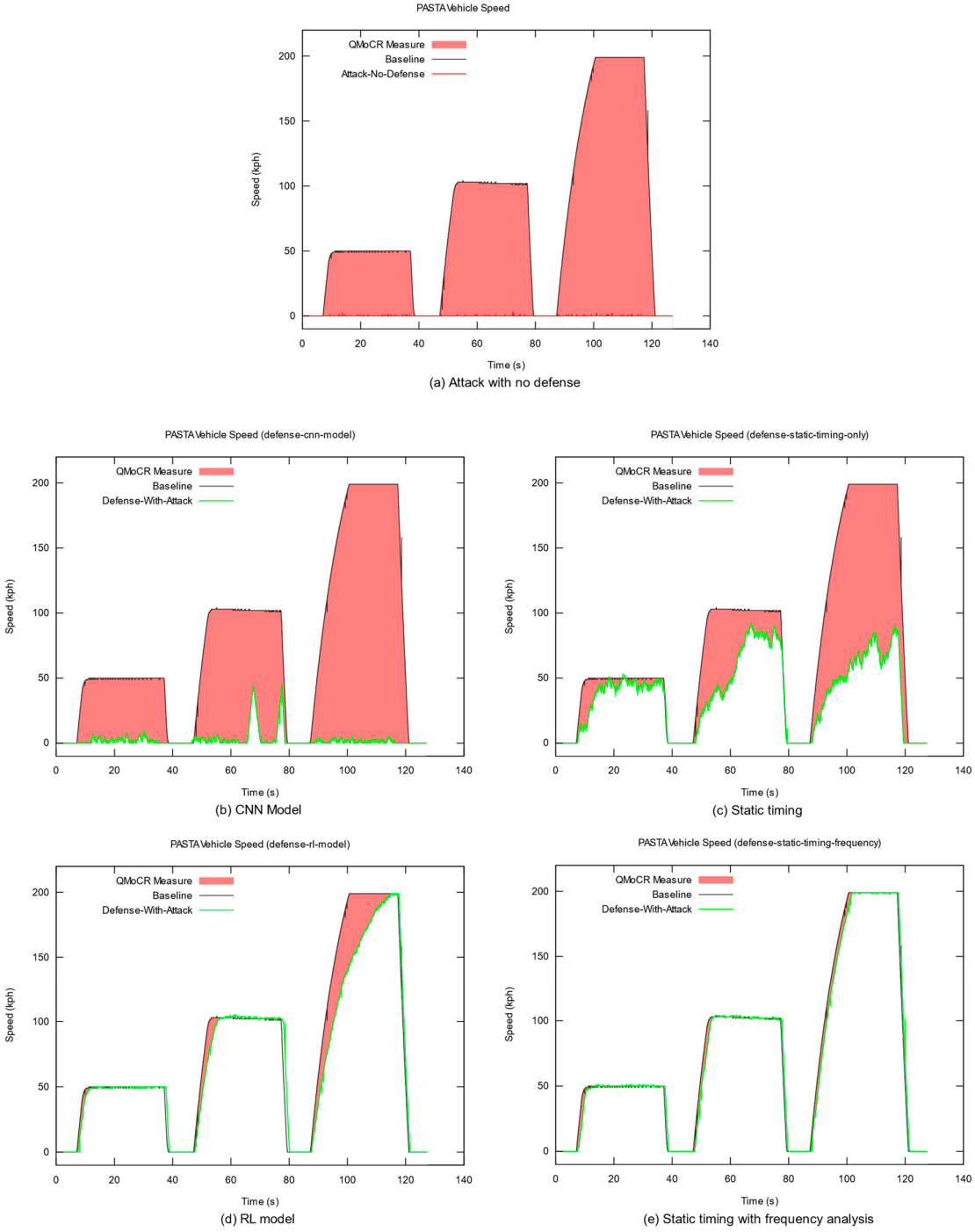

Figure 6. Sample graphs of baseline vehicle speed compared to the attack and defenses. Shading shows an approximation of the AUC difference used in the QMoCR measurement. More shading indicates less cyber resilience under the tested scenario.

The RL model (Figures 5c and 6d) performed better than the static timing defense alone (Figures 5b and 6c). This result is expected because the RL model adapts during training to recognize variations in message timing, order, and data content, whereas the static timing defense allows all traffic in the allowed time window, whether it is malicious or not. If a malicious accelerator or brake pedal message occurs after the legitimate message and is still within the window, which is not uncommon, it will impact the vehicle's performance. Also, as expected, the RL model performed worse than the customized static timing defense with frequency analysis (Figures 5d and 6e). When a defense is custom-crafted for a

particular scenario, even as simplistic as this one is, it can perform very well under those circumstances. The issue with this custom-crafted approach is twofold. First, it takes time, expertise, and awareness to implement custom defenses for every scenario. Second, this custom defense performs poorly when the attack changes (e.g., injecting random message values or time syncing the attack messages to arrive right before the legitimate messages). The RL model is not as sensitive to such changes as it would have experienced similar situations in training.

The CNN (Figures 5a and 6b) and LSTM (not illustrated) models performed worse than the RL model during evaluation. This is exemplified in the QMoCR measurement, where RL averaged 90% while CNN averaged 1%. The LSTM model failed to complete a drive cycle due to blocking legitimate transmission shifting messages, which kept the testbed from shifting into the drive.

This analysis also points out how the QMoCR measurement can benefit from the additional context provided by the AUC graphs. While the overall number may be sufficient for comparison during true tests of a system's ability to meet maximum KPPs, it does not solely provide all relevant information about changes in cyber resilience, which may be important when viewed in the context of a vehicle performing a mission. For example, one could imagine two defenses with identical QMoCR measurements; however, one might have many small performance degradations with gradual recovery, while the other may have a few larger performance degradations with faster recovery. These and other intricacies, such as variation between runs, may be important to a system owner and may not be adequately represented when looking at the final AUC measurement alone.

Additionally, the graphs above focus on vehicle speed and not holistic vehicle performance. The attack performed in this research, even after defenses are in place, could have significant impacts on the vehicles that are not accounted for in this analysis. For example, when any accelerator pedal attack message gets past the defense, that message causes the vehicle's throttle body actuator to change positions from what the driver is requesting. This leads to rapid fluctuations in the throttle body actuation signal, which would cause accelerated wear in the best case and could damage the component in the worst case. However, our intention is not to solve issues already facing the implementation of a network layer CAN bus defense.

In Table 5, we show the baseline performance of each ML algorithm trained and tested with or without observation dropping (described in Section 3.4). Labels "drop/drop" indicate that dropping was used during training and evaluation, and LSTM rows only indicate the test scenario as it must be trained solely on true messages. For the RL model and CNN binary classifier, dropping blocked messages yielded higher mean accuracy than preserving them in future observations, both with and without injected messages. The LSTM model accuracy benefitted from observation dropping only with no injections. The technique also reduced the models' variance, except for the RL model with injections and the LSTM model across both. Our RL approach with observation dropping produced the highest mean accuracy with or without injections and the lowest variance without injections.

Table 5. Accuracy of models trained and evaluated with and without injections on data collected from PASTA hardware environment, ablating for the inclusion of observation dropping.

Defense	Accuracy			
	No Injections		With Injections	
	Mean	Std Dev	Mean	Std Dev
CNN Classifier drop/drop	0.986131	0.009274	0.745306	0.010356
CNN Classifier no_drop/no_drop	0.985406	0.009316	0.709558	0.012134
LSTM drop	0.938945	0.079128	0.686017	0.027225
LSTM no_drop	0.954807	0.057592	0.685402	0.025105
RL drop/drop	**0.997037**	**0.004194**	**0.951322**	0.014763
RL no_drop/no_drop	0.965619	0.008333	0.889717	0.013051

5. Future Work

Several advancements of this research are required to realize the AICA vision within the CAN bus, let alone an entire vehicle platform. Future work in RL experimentation would include the addition of an inject message action so the agent can handle attacks that block legitimate messages, refinement of the learning process through tuning of RL parameters and better reward functions influenced by QMoCR measurements, addition of online learning for the agent to adapt in real or near-real time, and increased complexity of the training environment to the level of an actual vehicle platform. Additionally, to make such a solution viable for implementation within a vehicle platform, we must change from an on-path (i.e., man-in-the-middle) to an off-path (i.e., man-on-the-side) defense architecture through techniques such as physical layer fingerprinting [35] for message authentication and non-protocol-compliant bus interaction (e.g., CAN-stomping [36]) for blocking, injection, or masquerading defense actions.

A realistic training environment is essential for a successful defense agent. To provide a suitable training environment without requiring real-time execution, we created a simulation of the PASTA system (see Figure 7). The simulator replicates the CAN bus layout using Linux Virtual CAN (VCAN) interfaces, message sending, and receipt, including message timing, and estimates the physics of the vehicle platform. It also allows the automation of human drivers and associated vehicle drive cycles, as well as software interfaces for network attack creation and injection. We trained a defense model within our simulation environment, but it performed poorly when evaluated in the hardware PASTA environment. We believe this is due to subtle differences between the simulation and hardware environments, such as a drift in message ordering and timing on hardware arising from the non-determinism of multiple processes. Training models, despite this discrepancy, a task known as *sim2real transfer* [37], is important because any ultimate use of this technology in a real vehicle will require training in simulation to obtain the state space coverage needed for policy optimization. Therefore, additional research is needed into sim2real transfer techniques for PASTA, such as domain randomization, domain adaptation, or algorithmic techniques, such as perturbing model weights and meta-learning.

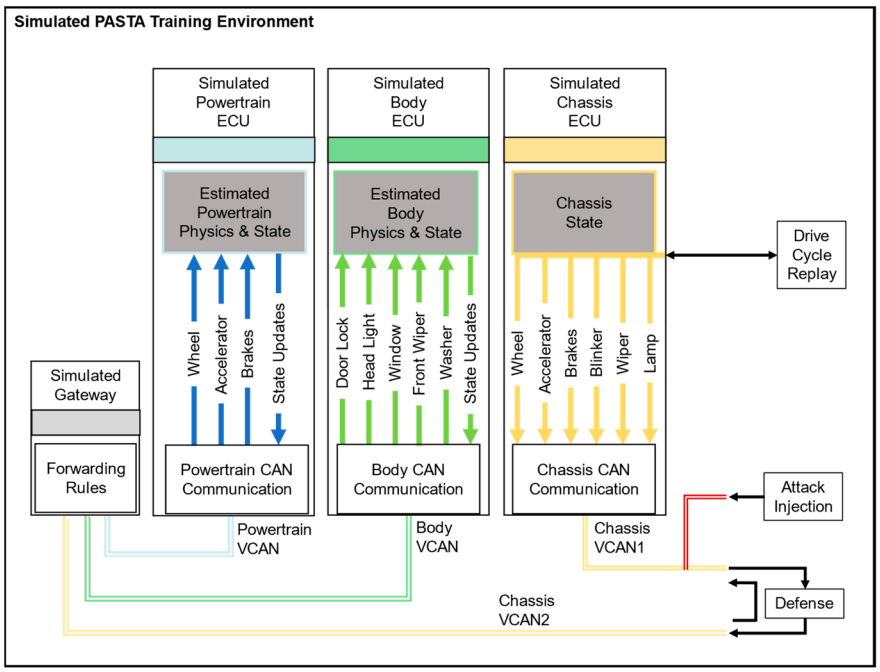

Figure 7. Simulated PASTA training environment.

While we evaluated some out-of-distribution learning and have discussed the need for sim2real transfer techniques, another similar avenue we would like to explore is automatic

curriculum learning via multi-agent interactions. It has been shown that auto-curricula arise when multiple agents are able to learn in a shared, competitive environment [38]. For the CAN bus defense task, one agent may learn to defend (as we do here) while another agent learns to attack. We should expect this system to generate novel attacks if set up appropriately such that the defender agent will generalize better at test time to unknown attacks.

Finally, it would be worth further exploring the utility of additional KPP evaluations and even additional cyber-resilience measurements to complement and expand upon the information conveyed through an area under the curve measurement.

6. Conclusions

Our results show that RL can form a viable base for an AICA, performing on par with other techniques for CAN bus anomaly detection while maintaining greater potential for adapting to novel attacks. This is demonstrated via direct comparison utilizing both detection performance measurements as well as quantitatively measured cyber resilience. During offline evaluation (Table 5), our model has higher accuracy than the CNN and LSTM models in classifying purely benign data (99.7% compared to 98.6% and 93.9 respectively for CNN and LSTM) and under injection attacks (95.1% compared to 74.5% and 68.6%). During the online evaluation of our drive cycle on hardware (Table 4), our agent performed at a 90% QMoCR measurement, which was significantly better than a naïve static timing defense (41%) and expectedly lower than a bespoke timing defense (98%). The non-RL CNN model performed at ~1% QMoCR measurement, and the LSTM model did not perform well enough to complete a drive cycle. Through our experimentation and leveraging the QMoCR measurement, we show that traditional performance measures of defense techniques do not provide an accurate understanding of how well a cyber-physical system will continue to operate through a cyberattack. There is still much work needed to fully realize the intelligence piece of an AICA, and this work provides a step further toward that vision.

Author Contributions: Conceptualization, S.R., G.G.S., K.C., T.W.P., J.S.E. and F.F.N.; methodology, S.R., G.G.S., K.C., T.W.P. and J.S.E.; software, T.W.P., K.C., S.R. and J.S.E.; validation, T.W.P., K.C., S.R. and J.S.E.; formal analysis, T.W.P., K.C. and S.R.; investigation, T.W.P., K.C. and S.R.; resources, S.R. and T.W.P.; data curation, T.W.P., K.C. and S.R.; writing—original draft preparation, S.R., K.C., T.W.P., M.R.T. and S.J.P.; writing—review and editing, S.R., K.C. and G.G.S.; visualization, K.C., S.R. and T.W.P.; supervision, F.F.N. and S.R.; project administration, F.F.N.; funding acquisition, F.F.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data and code presented here are currently unavailable due to security review and will be published on DEVCOM Army Research Laboratory's GitHub page upon approval. For reproducibility until then, existing CAN bus datasets exist, such as [39].

Acknowledgments: We thank the Army Research Laboratory team members who supported this effort, including Sidney C. Smith's Quantitative Measurement of Cyber Resilience team, especially Jason E. Ellis and Michael J. Weisman for their support in leveraging their measurement methodology, and Manuel M. Vindiola, Anjon Basak, and Kelly P. Toppin, Casey F. Rock, and Adam F. Bartlett, for their reinforcement learning guidance.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Russell, W.W. Weapon Systems Cybersecurity: Guidance Would Help DOD Programs Better Communicate Requirements to Contractors. 2021. Available online: https://www.gao.gov/products/gao-21-179 (accessed on 31 July 2023).
2. Smith, S. Towards a scientific definition of cyber resilience. In Proceedings of the International Conference on Cyber Warfare and Security, Towson, MD, USA, 9–10 March 2023; Volume 18.
3. Theron, P.; Kott, A.; Drašar, M.; Rzadca, K.; LeBlanc, B.; Pihelgas, M.; Mancini, L.; Panico, A. Towards an active, autonomous and intelligent cyber defense of military systems: The NATO AICA reference architecture. In Proceedings of the 2018 International Conference on Military Communications and Information Systems (ICMCIS), Warsaw, Poland, 22–23 May 2018; IEEE: New York, NY, USA, 2018.
4. Kott, A.; Théron, P.; Drašar, M.; Dushku, E.; LeBlanc, B.; Losiewicz, P.; Guarino, A.; Mancini, L.; Panico, A.; Pihelgas, M.; et al. Autonomous intelligent cyber-defense agent (AICA) reference architecture. Release 2.0. *arXiv* **2018**, arXiv:1803.10664.
5. Kott, A.; Weisman, M.J.; Vandekerckhove, J. Mathematical modeling of cyber resilience. In Proceedings of the MILCOM 2022-2022 IEEE Military Communications Conference (MILCOM), Rockville, MD, USA, 28 November–2 December 2022; IEEE: New York, NY, USA, 2022.
6. Marchetti, M.; Stabili, D. Anomaly detection of CAN bus messages through analysis of ID sequences. In Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, USA, 11–14 June 2017; IEEE: New York, NY, USA, 2017.
7. Levy, E.; Shabtai, A.; Groza, B.; Murvay, P.S.; Elovici, Y. CAN-LOC: Spoofing detection and physical intrusion localization on an in-vehicle CAN bus based on deep features of voltage signals. *arXiv* **2021**, arXiv:2106.07895. [CrossRef]
8. Cho, K.T.; Shin, K.G. Fingerprinting electronic control units for vehicle intrusion detection. In Proceedings of the USENIX Security Symposium, Austin, TX, USA, 10–12 August 2016; Volume 40.
9. Moore, M.R.; Bridges, R.A.; Combs, F.L.; Starr, M.S.; Prowell, S.J. Modeling inter-signal arrival times for accurate detection of can bus signal injection attacks: A data-driven approach to in-vehicle intrusion detection. In Proceedings of the 12th Annual Conference on Cyber and Information Security Research, Oak Ridge, TN, USA, 4 April 2017.
10. Lokman, S.-F.; Othman, A.T.; Abu-Bakar, M.-H. Intrusion detection system for automotive controller area network (CAN) bus system: A review. *EURASIP J. Wirel. Commun. Netw.* **2019**, *2019*, 184. [CrossRef]
11. Minawi, O.; Whelan, J.; Almehmadi, A.; El-Khatib, K. Machine learning-based intrusion detection system for controller area networks. In Proceedings of the 10th ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications, Alicante, Spain, 16–20 November 2020.
12. Purohit, S.; Govindarasu, M. ML-based Anomaly Detection for Intra-Vehicular CAN-bus Networks. In Proceedings of the 2022 IEEE International Conference on Cyber Security and Resilience (CSR), Rhodes, Greece, 27–29 July 2022; IEEE: New York, NY, USA, 2022.
13. Gundu, R.; Maleki, M. Securing CAN bus in connected and autonomous vehicles using supervised machine learning approaches. In Proceedings of the 2022 IEEE International Conference on Electro Information Technology (eIT), Mankato, MN, USA, 19–21 May 2022; IEEE: New York, NY, USA, 2022.
14. Taylor, A.; Japkowicz, N.; Leblanc, S. Frequency-based anomaly detection for the automotive CAN bus. In Proceedings of the 2015 World Congress on Industrial Control Systems Security (WCICSS), London, UK, 14–16 December 2015.
15. Kang, M.-J.; Kang, J.-W. Intrusion detection system using deep neural network for in-vehicle network security. *PLoS ONE* **2016**, *11*, e0155781. [CrossRef] [PubMed]
16. Song, H.M.; Woo, J.; Kim, H.K. In-vehicle network intrusion detection using deep convolutional neural network. *Veh. Commun.* **2020**, *21*, 100198. [CrossRef]
17. Qin, H.; Yan, M.; Ji, H. Application of controller area network (CAN) bus anomaly detection based on time series prediction. *Veh. Commun.* **2021**, *27*, 100291. [CrossRef]
18. Seo, E.; Song, H.M.; Kim, H.K. GIDS: GAN based intrusion detection system for in-vehicle network. In Proceedings of the 2018 16th Annual Conference on Privacy, Security and Trust (PST), Belfast, UK, 28–30 August 2018; IEEE: New York, NY, USA, 2018.
19. Wang, Z.; Kim, S.; Joe, I. An Improved LSTM-Based Failure Classification Model for Financial Companies Using Natural Language Processing. *Appl. Sci.* **2023**, *13*, 7884. [CrossRef]
20. Yu, Z.; Liu, Y.; Xie, G.; Li, R.; Liu, S.; Yang, L.T. TCE-IDS: Time interval conditional entropy-based intrusion detection system for automotive controller area networks. *IEEE Trans. Ind. Informatics* **2022**, *19*, 1185–1195. [CrossRef]
21. Song, H.M.; Kim, H.R.; Kim, H.K. Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network. In Proceedings of the 2016 International Conference on Information Networking (ICOIN), Kota Kinabalu, Malaysia, 13–15 January 2016; IEEE: New York, NY, USA, 2016.
22. Ji, H.; Wang, Y.; Qin, H.; Wang, Y.; Li, H. Comparative performance evaluation of intrusion detection methods for in-vehicle networks. *EEE Access* **2018**, *6*, 37523–37532. [CrossRef]
23. Nguyen, T.T.; Reddi, V.J. Deep reinforcement learning for cyber security. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *34*, 3779–3795. [CrossRef] [PubMed]
24. Sewak, M.; Sahay, S.K.; Rathore, H. Deep reinforcement learning for cybersecurity threat detection and protection: A review. In Proceedings of the International Conference on Secure Knowledge Management in Artificial Intelligence Era, San Antonio, TX, USA, 8–9 October 2021; Springer International Publishing: Cham, Switzerland, 2021.

25. Lopez-Martin, M.; Carro, B.; Sanchez-Esguevillas, A. Application of deep reinforcement learning to intrusion detection for supervised problems. *Expert Syst. Appl.* **2020**, *141*, 112963. [CrossRef]
26. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009; IEEE: New York, NY, USA, 2009.
27. Kolias, C.; Kambourakis, G.; Stavrou, A.; Gritzalis, S. Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset. *IEEE Commun. Surv. Tutor.* **2015**, *18*, 184–208. [CrossRef]
28. Xiao, L.; Lu, X.; Xu, T.; Zhuang, W.; Dai, H. Reinforcement learning-based physical-layer authentication for controller area networks. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 2535–2547. [CrossRef]
29. Portable Automotive Security Testbed with Adaptability. Ver.1.0. Japan. Toyota. 2018. Available online: https://www.chip1stop.com/sp/products/toyota-pasta_en. (accessed on 31 July 2023).
30. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
31. Oh, J.; Guo, X.; Lee, H.; Lewis, R.L.; Singh, S. Action-conditional video prediction using deep networks in atari games. In Proceedings of the 29th Annual Conference on Neural Information Processing Systems 2015, Montreal, QC, Canada, 7–12 December 2015; Curran Associates, Inc.: Red Hook, NY, USA, 2015.
32. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Koray, D.S.; Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In Proceedings of the International Conference on Machine Learning (PMLR), New York, NY, USA, 19–24 June 2016.
33. Hanselmann, M.; Strauss, T.; Dormann, K.; Ulmer, H. CANet: An unsupervised intrusion detection system for high dimensional CAN bus data. *IEEE Access* **2020**, *8*, 58194–58205. [CrossRef]
34. Kott, A.; Weisman, M.J.; Ellis, J.E.; Parker, T.W.; Murphy, B.J.; Smith, S. *A Methodology for Quantitative Measurement of Cyber Resilience (QMOCR)*; Apr. Report No.: ARL-TR-9672; Army Research Laboratory (US): Adelphi, MD, USA, 2023.
35. Avatefipour, O.; Hafeez, A.; Tayyab, M.; Malik, H. Linking received packet to the transmitter through physical-fingerprinting of controller area network. In Proceedings of the 2017 IEEE Workshop on Information Forensics and Security (WIFS), Rennes France, 4–7 December 2017; IEEE: New York, NY, USA, 2017.
36. Giannopoulos, H.; Wyglinski, A.M.; Chapman, J. Securing vehicular controller area networks: An approach to active bus-level countermeasures. *IEEE Veh. Technol. Mag.* **2017**, *12*, 60–68. [CrossRef]
37. Kaspar, M.; Osorio, J.D.M.; Bock, J. Sim2real transfer for reinforcement learning without dynamics randomization. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 25–29 October 2020; IEEE: New York, NY, USA, 2020.
38. Baker, B.; Kanitscheider, I.; Markov, T.; Wu, Y.; Powell, G.; McGrew, B.; Mordatch, I. Emergent tool use from multi-agent autocurricula. *arXiv* **2019**, arXiv:1909.07528.
39. Lee, H.; Jeong, S.H.; Kim, H.K. OTIDS: A novel intrusion detection system for in-vehicle network by using remote frame. In Proceedings of the 2017 15th Annual Conference on Privacy, Security and Trust (PST), Calgary, AB, Canada, 28–30 August 2017; IEEE: New York, NY, USA, 2017.