2021 IEEE 27th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA) | 978-1-6654-4188-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/RTCSA52859.2021.00015

Recovery-by-Learning: Restoring Autonomous Cyber-physical Systems from Sensor Attacks

Francis Akowuah†, Romesh Prasad‡, Carlos Omar Espinoza‡ and Fanxin Kong†
Dept. of Electrical Engineering and Computer Science†, Dept. of Mechanical & Aerospace Engineering‡
Syracuse University, Syracuse NY USA
feakowua@syr.edu, rsprasad@syr.edu, coespino@syr.edu, fkong03@syr.edu

***Abstract*—Autonomous cyber-physical systems (CPS) are susceptible to non-invasive physical attacks such as sensor spoofing attacks that are beyond the classical cybersecurity domain. These attacks have motivated numerous research efforts on attack detection, but little attention on what to do after detecting an attack. The importance of attack recovery is emphasized by the need to mitigate the attack's impact on a system and restore it to continue functioning. There are only a few works addressing attack recovery, but they all rely on prior knowledge of system dynamics. To overcome this limitation, we propose Recovery-by-Learning, a data-driven attack recovery framework that restores CPS from sensor attacks. The framework leverages natural redundancy among heterogeneous sensors and historical data for attack recovery. Specially, the framework consists of two major components: state predictor and data checkpointer. First, the predictor is triggered to estimate systems states after the detection of an attack. We propose a deep learning-based prediction model that exploits the temporal correlation among heterogeneous sensors. Second, the checkpointer executes when no attack is detected. We propose a double sliding window based checkpointing protocol to remove compromised data and keep trustful data as input to the state predictor. Third, we implement and evaluate the effectiveness of our framework using a realistic data set and a ground vehicle simulator. The results show that our method restores a system to continue functioning in presence of sensor attacks.**

***Index Terms*—autonomous cyber-physical systems, sensor attacks, attack recovery, attack resiliency**

I. Introduction

Cyber-Physical Systems (CPS) tightly couple computing and communication components with physical processes. Self-driving cars, smart grid, intelligent manufacturing systems, and many other CPS have been revolutionizing our modern life. For example, autonomous CPS such as unmanned aerial vehicles (UAV) have been used in many applications including aerial photography, construction site management, policing and surveillance, and even package delivery.

CPS has changed from once-isolated systems to open architectures, which makes them susceptible to various cyber and physical attacks. On the one hand, cyber attacks such as network eavesdropping, data modification, packet spoofing, packet replaying, buffer overflow, etc, are attacks that compromise the computing and communication components of the CPS. Physical attacks, on the other hand, perturbs the physical environment of the CPS such that it allows the injection of malicious signals into sensors and actuators. Such attacks include dazzling cameras with light, injecting false radar signals, GPS Spoofing etc. Defense measures for cyber attacks are relatively advanced. Conventional cyber security techniques such as memory isolation [1], control-flow integrity [2], firmware hardening [3] are applicable in defending CPS against cyber attacks. These security techniques are, however, inadequate against physical attacks.

This observation is highlighted by non-invasive sensor attacks in the literature and real world. For example, Rutkin [4] demonstrated a GPS spoofing attack that misguided a yacht off course. Shoukry et al. [5] used an external magnetic field to perturb the physical environment of a vehicle's wheel speed sensors to disrupt the function of Anti-lock Braking System (ABS). Camera and LiDAR devices on modern automobiles can be also attacked from a remote location [6]. Non-invasive physical attacks do not require close proximity to a physical component in order to succeed. In addition, it requires little knowledge and inexpensive tools for an attacker to launch physical attacks. The effectiveness of such attacks will continue to rise as the autonomy in CPS increases.

These new CPS threats have motivated many research efforts to defend against sensor attacks. However, most of them have focused on attack detection rather than recovery measures. Raising attack alerts, usually done in attack detection works, do not ensure the continuous functioning of the CPS [7]–[9]. Hence, to respond to attack alerts, recovery measures are required to mitigate the effect of an attack on a system and continue the system's operation with minimum interruption. Only a few existing works have addressed attack-recovery in any form. As the pioneer work in this research thread, Kong et al. [10], [11] assumes full observability of the system and replaces measurements of compromised sensors by model-predicted values. Fei et al. [12] follow the idea of [10] and proposes a redundant controller for attack recovery that is trained also based on the system model. Although these works validate the performance under their own settings, they require prior knowledge of system dynamics that builds the system model.

To address these limitations, we propose Recovery-by-Learning, a data-driven attack recovery framework that restores automotive cyber-physical systems from sensor attacks. The framework requires little knowledge of the system's dynamics, but leverages natural redundancy among heterogeneous sensors and historical data for attack recovery. Specially, the framework consists of two major components: state predictor and data checkpointer. At the core of this paper are novel techniques to realize these components.

First, the state predictor is activated to estimate system states when an attack is detected. The predicted states are forwarded to the controller to calculate and issue appropriate control commands to bring the system back to normalcy. The predictor is built on a deep learning model that captures the nominal system behavior. The model exploits the natural redundancy as well as the short and long term temporal correlation among heterogeneous sensors on an autonomous CPS, through combining convolutional neural network (CNN) and recurrent neural network (RNN).

Second, the data checkpointer executes in the normal mode when no attack is detected. It employs a checkpointing protocol to remove corrupted data and keep valid historical data as input to the state predictor to make state estimation. The protocol uses double sliding windows: detection window and logging window. The former accommodates the substantial detection delay (i.e., the time interval between the start of an attack and the detection of it), during which the correctness of the sensor data is still in question and thus using them may result in unsuccessful recovery. The logging window governs sufficient trustful data for the state prediction.

We implement and evaluate the effectiveness of our framework using a real-world data set, AEGIS Big Data Project [13], and a ground vehicle simulator, Ardupilot SITL Rover [14]. The results show that the proposed framework is capable of ensuring continuous functionality in presence of sensor attacks.

Scope and Contributions. This paper focuses on sensor attack-recovery. We assume an attack detector is already in place, and our goal is to take the alerts generated by the detector to recover the system from the attacks. The contributions of this work are as follows. (i) We propose Recovery-by-Learning, a model-free attack recovery framework. (ii) We propose a deep learning based state prediction method and a double sliding window based checkpointing protocol. (iii) We perform extensive data-driven simulations to validate the proposed methods.

The rest of the paper is organized as follows. Section II describes the system overview and the threat model we consider. The design details of the proposed system are provided in section III. We evaluate our approach in Section IV. Section V concludes the paper.

II. Preliminaries

A. Sensor Correlation

Autonomous CPSs are equipped with a number of sensors that enable them to perform their function. The sensors monitor various physical properties such as engine revolutions, vehicle and wheel speed, oil temperature, boost pressure, accelerator pedals, location, etc. It is observed that a subset of sensors on the CPS responds to a physical phenomenon in a correlated or related manner. Such a group of sensors are referred to as heterogeneous sensors or are said to exhibit inherent sensor redundancy. For instance, applying the brakes

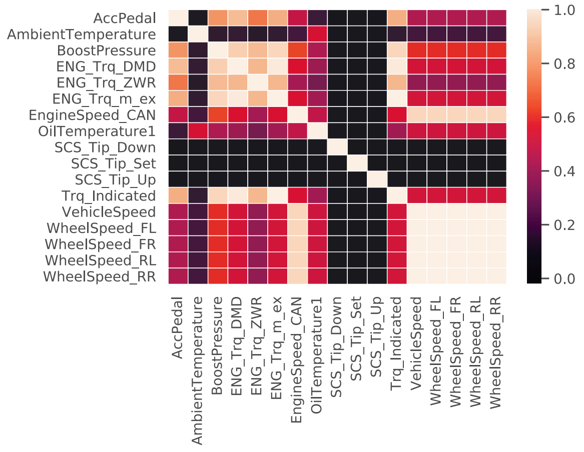

Fig. 1. Pearson Correlation Coefficients heatmap of some some sensors in the dataset.

of a vehicle causes decrements in the engine's RPM, wheel speed, vehicle speed and GPS speed sensors measurements. Similarly, pressing the accelerator pedal leads to increases in the readings of these heterogeneous sensors. Fig. 1 shows the pairwise correlation among sensors in an automobile using the dataset of [13]. Details of the data set will be given in Section IV-B. It is observed in the figure that the wheel speed sensors have a strong correlation with the engine RPM and boost pressure sensors. Hence, when the wheel speed sensor reading increases as a result of applying the accelerator pedals, the readings of the engine RPM and boost pressure will also be observed to increase under normal conditions [15]–[18].

We believe that exploiting and capturing this inherent sensor redundancy allows us to approximate the nominal system behavior which in turn, enables the accurate prediction of system behavior such as sensor readings. We leverage this notion in our proposed attack recovery system.

B. System Overview

Fig. 2 shows an overview of the proposed system. It consists of an offline phase and an online phase. The offline phase involves the data collection, pre-processing of the data, and training the model on the data. The online phase has two major components: the State Predictor and Checkpointer. The state predictor estimates system states when the observed sensor data are no longer trustworthy. The state predictor is built on a deep learning model that captures the nominal system behavior. The Checkpointer ensures valid historical state estimates are stored so that the state predictor can output accurate state estimations.

The proposed system works as follows: When a time-window-based attack detector raises an alert, the state predictor is activated. The checkpointer provides valid and trustful sensor values as input to the state predictor to predict state estimates. The predicted values are forwarded to the controller to perform recovery control commands. In the event the attack detector does not raise an alert, the system continues to function and only the *checkpointer* performs tasks to warrant that only valid historical sensor data is stored as checkpoints.

C. Threat Model

We consider the attack scenario where an attacker launches physical attacks against the CPS sensors. Examples of such attacks include optical sensor spoofing [19], gyroscope sensor spoofing [20], accelerometer spoofing attacks [21] among others. These attacks transmit compromised sensor data which does not reflect the actual system state. When the controller receives and processes such data, erroneous control inputs are calculated and issued resulting in safety problems, abnormal system operation and possibly stalling the CPS.

As noted above, there are many proposed attack detection solutions, therefore, we assume the existence of a sensor attack detector that is able to raise an alert whenever any of these attacks occur. Our goal is to automatically respond to the attack alert and steer the system towards a reference state thereby ensuring safety and CPS operation continuity. We also assume the controller, actuator, the proposed deep learning model and the stored historical sensor data are not compromised.

III. Recovery System Design

We describe the details of the proposed attack recovery system in this section.

Rationale. Estimating the accurate behavior of a cyber-physical system, specifically sensor measurement is non-trivial, yet it is a crucial step in attack recovery. The actual behavior of the CPS is guarded by physical laws hence under normal conditions, the physical system properties, called *physical invariant* should always hold. While physical invariant can be captured using a physical system model, it requires in-depth knowledge of the system dynamics which may not be easy to attain.

We propose an attack recovery system that does not require substantial knowledge of system dynamics. We treat the physical system as a black box yet we are able to approximate the nominal system behavior. We achieve this through a deep learning technique that explores the natural redundancy among its heterogeneous sensors. Our approach is based on the insight that under normal conditions, where physical laws are obeyed, the sensor readings also indirectly obey physical laws. Therefore, by learning the relationships among sensor data, the physical invariants are approximated. Having modeled the system from sensor data, we are able to estimate sensor readings with a near-zero error.

A. Problem formulation

We envision a solution for the attack recovery problem to involve two main steps. The first step seeks to replace the corrupted sensor data that no longer reflect the true state of the system with reconstructed system state estimates. The second step attempts to control the CPS with the reconstructed values which we call *recovery control*. Given that we consider multiple heterogeneous sensor time-series data or measurements in making a prediction, we formulate the first step as a deep learning multivariate time series forecasting problem [22].

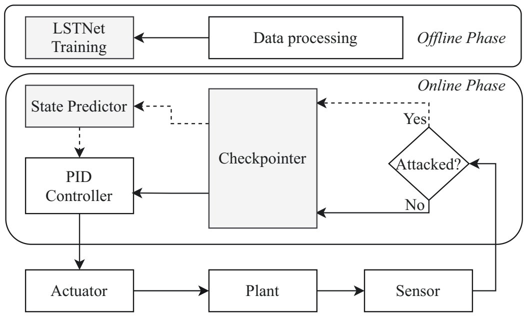

Fig. 2. Design overview of our Recovery-by-Learning framework.

Hence, given valid historical time series output of n heterogeneous sensors $Y = \{y_1, y_2, y_3, ...y_K\}$ where $y_k \in \mathbb{R}^n$, we aim to predict y_{k+h} where h is a time ahead of the current time k. We ensure $Y = \{y_1, y_2, y_3, ...y_K\}$ is always available by proposing a checkpointing protocol to store such valid historical sensor data. For ease of presentation, we assume that the system has full observability. Thus, we formulate the input matrix as $X_K = \{y_1, y_2, y_3, ..., y_K\} \in \mathbb{R}^{n \times K}$. Once a valid prediction is made, the second step undertakes a recovery control that uses the predicted values to drive the system.

B. System Components

Data Processor: Data pre-processing is an important step in a machine or deep learning task. Specifically, we ensure we extract only features that have a strong correlation with the sensor of interest. While this can be achieved from domain knowledge, we leverage Pearson Correlation Coefficients (PCC) statistic to observe this correlation in the dataset. This step also has the potential to reveal correlations that may not be obvious to humans. PCC outputs values between -1.0 and $+1.0$. Feature pairs that have a strong positive correlation have PCC values close to $+1.0$. Conversely, a strong negative correlation has PCC values close to -1.0. A zero PCC value indicates there is no correlation between the features. For example, in Fig. 1, the vehicle speed sensor has PCC of approximately 1.0 with the engine RPM and the wheel speed sensors. The vehicle speed sensor, however, has a PCC value close to zero with the ambient temperature sensor.

LSTNet Training: In order to automatically exploit the correlation that exists in the sensor data, we train a deep learning model based on LSTNet [22]. LSTNet was originally developed to model long and short term temporal forecasting for multivariate time series. The deep learning architecture captures nonlinear aspects of the system by using a convolutional neural network (CNN) and recurrent neural network (RNN) for exploiting short and long term correlations respectively. To improve scalability and robustness, the model also includes autoregressive units that enable the DL model to capture linear aspects of the system as well. Fig. 3 shows the deep learning architecture that trains our model. Training the

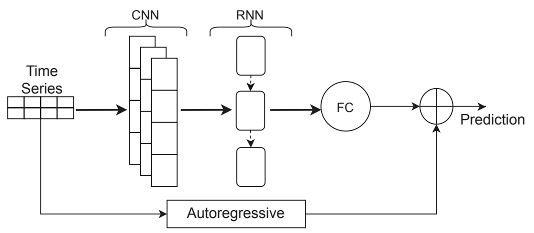

Fig. 3. Overview of deep learning model architecture (LSTNet [22]). FC refers to Fully Connected.

model requires a number of hyperparameters to be specified, notable among them is the *window* p. The window p specifies how many historical data points should be used in making a prediction.

State Predictor: This component is built on the trained deep learning model training discussed above. At this point, the data model has captured the nominal behavior of the system by exploring the correlation among heterogeneous sensors. It serves as a nominal approximation of the system behavior and hence it is able to make predictions of the system behavior given the right input. In order to make the deep learning model useful in a non-Python environment, we utilize Torchscript to build an intermediate representation (IR) so that it can be used in high-performance environments such as C++ to make predictions. The firmware of the ground vehicle simulator used in our experiment is written in C++.

Checkpointer: Attack detection mechanisms take some time to detect an attack after the attack's launch, called *detection delay*, before raising an alert. As a result, we cannot trust the sensor readings during the detection delay since they may have been compromised. A successful recovery cannot be achieved if we rely on corrupted data, hence to address this issue, checkpointing protocols [10], [23]–[25] have been proposed to provide trustworthy historical data that can be used for recovery. Though viable, especially for model-based recovery methods [11], these protocols have limitations of storing only one data point making it unsuitable for learning-based methods such as ours which require an interval of data points for reconstructing sensor data.

The checkpointer component addresses this limitation by proposing a checkpointing protocol shown in Fig. 4 that is not only applicable to learning-based methods but model-based methods as well. The proposed protocol adopts a double sliding window instead of the single sliding window approach used in existing works. This approach enables the protocol to capture an interval of historical data and the detection delay. The two windows of the protocol slide forward and records the sensor values $x(t)$ as time ticks.

The protocol has three steps namely buffer, store, and delete. (1) *Buffer*: The data in this step is possibly compromised and has a duration equivalent to the detector's detection delay. State estimates or sensor data within the detection window, $x(t_0), ..., x(t_h)$ are first buffered. (2) *Store*: Given that the detection window equals the detection delay, the data points

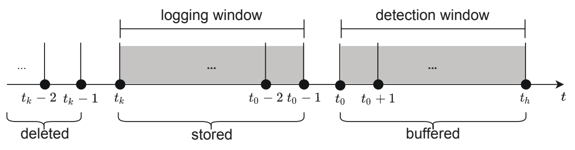

Fig. 4. A double sliding window based checkpointing protocol.

in this step have moved outside the detection window and are therefore considered trustworthy. Note that an interval of time series datapoints (logging window data) are stored instead of a single data point. Hence, for the interval $[t_k, t_0 - 1]$, datapoints $\{x(t_k), ..., x(t_0 - 1)\}$ are stored. Remember that this length is equal to the p window hyperparameter of the LSTNet component discussed above. (3) *Delete*: All historical data that are older than those in the logging window are no longer needed and should therefore be deleted. Data points $x(t_k - 1)$ and $x(t_k - 2)$ are discarded as shown in the figure.

When the detector raises an alarm, time series datapoints of the interval $[t_k, t_0]$ will be used to rebuild estimate $x(t_h)$.

IV. Evaluation

We perform experiments to evaluate the effectiveness of our proposed attack recovery.

A. Implementation and Experimental Setup

We implemented our deep learning model in Python, utilizing PyTorch Deep Learning framework. The experimental model is made up of 120 convolutional layers, 120 GRU layers and an AR model. A batch size of 128 serves as input to the network. We train our proposed DL model on Ubuntu 18.04 64-bit with sixteen Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz CPUs, two Nvidia GeForce GTX 1080 GPUs and 64 GB RAM. We split the original dataset into 60% training, 20% validation and 20% test sets.

B. Dataset Description

We evaluate the efficiency of the proposed recovery system using the publicly-available automotive CAN bus dataset from the AEGIS Big Data Project [13] and data collected from Ardupilot SITL virtual ground vehicle.

The AEGIS data was collected during trips conducted by three drivers driving the same vehicle. It contains more than 40 sensor measurements including but not limited to the four wheel speed sensors, engine speed, vehicle speed, steering angle, ambient temperature, GPS, oil temperature and boost pressure.

Ardupilot SITL (software in the loop) is a simulator package that provides a native executable that allows one to run Plane, Copter or Rover (ground vehicle) without any hardware. The virtual ground vehicle that we use in our experiment runs a firmware (APMrover2 2.5) that is used in real unmanned ground vehicle boards. The vehicle is equipped with a number of sensors including GPS, IMU, RPM, optical flow sensors.

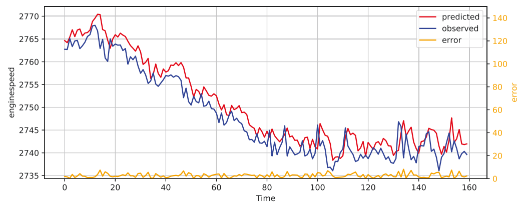

Fig. 5. Observed and predicted engine speed sensor readings in the AEGIS dataset.

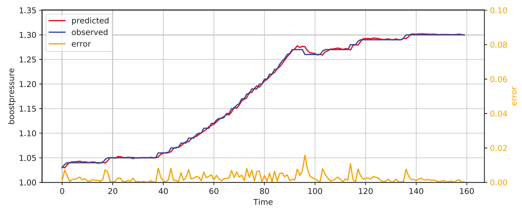

Fig. 6. Observed and predicted boost pressure sensor readings in the AEGIS dataset.

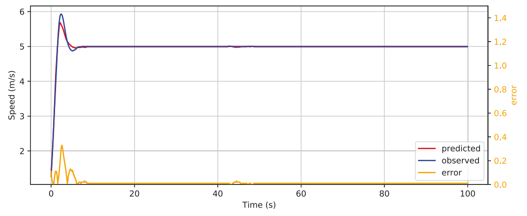

Fig. 7. Observed and predicted boost speed measurements of the unmanned ground vehicle.

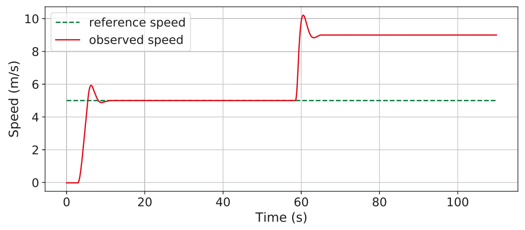

Fig. 8. UGV cruises above its reference speed of 5 m/s.

C. Experiments and Results

Experiment I: This experiment verifies if the State Predictor learned the nominal behavior of the CPS and evaluates its effectiveness in reconstructing sensor data. We build two models: the first one is based on the AEGIS dataset and the second is based on the virtual unmanned ground vehicle's sensor data. The second model is also used in a case study to demonstrate how the proposed framework recovers the unmanned ground vehicle (UGV) from a speed sensor attack.

Note, however, that in this experiment no attack has been launched. Fig. 5 and Fig. 6 show the model predictions for the engine speed and boost pressure sensors in the AEGIS test dataset. In the figures, the predicted speed (red line) closely matches the observed speed (blue line) indicating the model captured the nominal behavior of the vehicle. The figures also show the mean error or residual is near zero indicating the predictor is not biased. Residual analysis shows the error follows a normal Gaussian distribution and it is free from any cyclic, trend and seasonal structures.

We perform a similar experiment on the UGV. Using Mission Planner [26], we generate missions (trajectory) that the vehicle executes. We collected the sensor data from the dataflash log and used it to build a data model. The model's predictions for the UGV's speed sensor test dataset is shown in Fig. 7. The results depict a close match between the model predictions and the observed sensor values. Similar to the results in Fig. 5 and Fig. 6, there is a mean error in the model that is near-zero.

Experiment II - Case Study: We demonstrate attack recovery in this case study. The attacker launches attack on the speed sensor which leads the cruise/speed controller to issue wrong control inputs resulting in the UGV to travel above its cruise speed of 5 m/s. Fig. 8 shows the case study considered in this experiment. At 60 sec, we simulate an attack that transmits $\rho - 4$ m/s as the forward speed of the vehicle, where ρ is the actual speed of the vehicle. Hence, in order to maintain the reference speed, the PID controller issued a higher throttle output that resulted in the vehicle to cruise at about 9 m/s. In real-life, this scenario can be a safety concern.

Recovery control: When this attack is detected, we can no longer trust the sensor data and therefore the system states must be estimated or reconstructed. Our proposed framework responds with the goal of getting the UGV to cruise at its reference speed (5 m/s). It achieves that by activating the state predictor component which uses the data in the checkpointer as input. The state predictor reconstructs sensor values that are forwarded to the cruise controller. The controller calculates throttle outputs based on the reconstructed values that eventually cause the vehicle to travel at the reference speed as seen in Fig. 9.

Experiment III: The effectiveness of the proposed framework largely depends on the p value selected i.e. how many historical values are used for the deep learning model's prediction (see Section III-B). Remember also that the p value is equal to the length of the logging window in the proposed checkpointing protocol. In this experiment, we provide an analysis of the p value so that the optimal value can be selected to predict more accurate sensor values. We compare various p values based on accuracy metrics: mean square error (mse), root relative squared error (rse) and relative absolute error (rae). Table I shows the results of the vehicle speed sensor values in the AEGIS dataset. Values between 56 and 84 produced the highest prediction values. A similar analysis done on the ground vehicle showed a slightly different results which leads us to conclude that the best p value is device/dataset-specific.

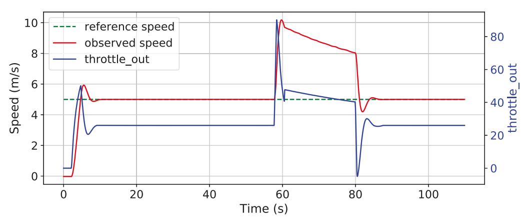

Fig. 9. Speed attack recovery.

TABLE I
COMPARISON OF P VALUES BASED ONE ACCURACY METRICS.

p	mse	rse	rae
28	0.020843	0.0034	0.0020
42	0.019189	0.0034	0.0020
56	0.012859	0.0031	0.0018
84	0.011068	0.0032	0.0018
168	0.021194	0.0034	0.0020
188	0.008365	0.0034	0.0018

V. CONCLUSION

In this paper, we have presented a model-free attack recovery system that does not require in-depth knowledge of system dynamics and also allow autonomous CPS to use existing components (controllers and sensors) without further duplication. We achieve this by applying a novel deep learning framework to capture the nominal behavior of the cyber-physical system. We proposed a new generalized double sliding window checkpointing protocol that is usable both in model-based and learning-based recovery methods. We performed experiments to evaluate the effectiveness of the proposed framework using real-world dataset and realistic unmanned ground vehicle simulator. Our results show that our method restores a system to continue functioning in the presence of sensor attacks.

ACKNOWLEDGMENT

We would like to thank the anonymous reviewers for their constructive comments. This work was supported in part by NSF CCF-2028740.

REFERENCES

[1] C. H. Kim, T. Kim, H. Choi, Z. Gu, B. Lee, X. Zhang, and D. Xu, "Securing real-time microcontroller systems through customized memory view switching." in *NDSS*, 2018.

[2] A. A. Clements, N. S. Almakhdhub, K. S. Saab, P. Srivastava, J. Koo, S. Bagchi, and M. Payer, "Protecting bare-metal embedded systems with privilege overlays," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 289–303.

[3] H. Shacham, M. Page, B. Pfaff, E.-J. Goh, N. Modadugu, and D. Boneh, "On the effectiveness of address-space randomization," in *Proceedings of the 11th ACM conference on Computer and communications security*, 2004, pp. 298–307.

[4] A. H. Rutkin, "Spoofers use fake GPS signals to knock a yacht off course," *MIT Technology Review*, 2013.

[5] Y. Shoukry, P. Martin, P. Tabuada, and M. Srivastava, "Non-invasive spoofing attacks for anti-lock braking systems," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2013, pp. 55–72.

[6] J. Petit, B. Stottelaar, M. Feiri, and F. Kargl, "Remote attacks on automated vehicles sensors: Experiments on camera and lidar," *Black Hat Europe*, vol. 11, p. 2015, 2015.

[7] H. Choi, W.-C. Lee, Y. Aafer, F. Fei, Z. Tu, X. Zhang, D. Xu, and X. Deng, "Detecting attacks against robotic vehicles: A control invariant approach," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 801–816.

[8] R. Quinonez, J. Giraldo, L. Salazar, and E. Bauman, "Savior: Securing autonomous vehicles with robust physical invariants," 2020.

[9] T. He, L. Zhang, F. Kong, and A. Salekin, "Exploring inherent sensor redundancy for automotive anomaly detection," *DAC2020*, 2020.

[10] F. Kong, M. Xu, J. Weimer, O. Sokolsky, and I. Lee, "Cyber-physical system checkpointing and recovery," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2018, pp. 22–31.

[11] L. Zhang, X. Chen, F. Kong, and A. A. Cardenas, "Real-time recovery for cyber-physical systems using linear approximations," in *41st IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2020.

[12] F. Fei, Z. Tu, D. Xu, and X. Deng, "Learn-to-recover: Retrofitting UAVs with reinforcement learning-assisted flight control under cyberphysical attacks," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[13] C. Kaiser, A. Stocker, and A. Festl, "Automotive CAN bus data: An example dataset from the AEGIS Big Data Project," Jul. 2019. [Online]. Available: https://doi.org/10.5281/zenodo.3267184

[14] "SITL Simulator (Software in the Loop)," https://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html, [Online; accessed Nov-2020].

[15] A. Ganesan, J. Rao, and K. Shin, "Exploiting consistency among heterogeneous sensors for vehicle anomaly detection," SAE Technical Paper, Tech. Rep., 2017.

[16] Z. Tyree, R. A. Bridges, F. L. Combs, and M. R. Moore, "Exploiting the shape of can data for in-vehicle intrusion detection," in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*. IEEE, 2018, pp. 1–5.

[17] Z. Wang, F. Guo, Y. Meng, H. Li, H. Zhu, and Z. Cao, "Detecting vehicle anomaly by sensor consistency: An edge computing based mechanism," in *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2018, pp. 1–7.

[18] F. Guo, Z. Wang, S. Du, H. Li, H. Zhu, Q. Pei, Z. Cao, and J. Zhao, "Detecting vehicle anomaly in the edge via sensor consistency and frequency characteristic," *IEEE Transactions on Vehicular Technology*, 2019.

[19] D. Davidson, H. Wu, R. Jellinek, V. Singh, and T. Ristenpart, "Controlling uavs with sensor input spoofing attacks," in *10th {USENIX} Workshop on Offensive Technologies (WOOT 16)*, 2016.

[20] 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 *24th USENIX Security Symposium (USENIX Security 15)*, 2015, pp. 881–896.

[21] T. Trippel, O. Weisse, W. Xu, P. Honeyman, and K. Fu, "Walnut: Waging doubt on the integrity of mems accelerometers with acoustic injection attacks," in *2017 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 2017, pp. 3–18.

[22] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018, pp. 95–104.

[23] F. Kong, O. Sokolsky, J. Weimer, and I. Lee, "State consistencies for cyber-physical system recovery," in *the 2nd Workshop on Cyber-Physical Systems Security and Resilience (CPS-SR)*, 2019, pp. 3–7.

[24] R. Ma, S. Basumallik, S. Eftekharnejad, and F. Kong, "Recovery-based model predictive control for cascade mitigation under cyber-physical attacks," in *2020 IEEE Texas Power and Energy Conference (TPEC)*. IEEE, 2020, pp. 1–6.

[25] M. Pajic, I. Lee, and G. J. Pappas, "Attack-resilient state estimation for noisy dynamical systems," *IEEE Transactions on Control of Network Systems*, vol. 4, no. 1, pp. 82–92, 2016.

[26] Ardupilot, *Mission Planner Home*, 2020 (accessed July 20, 2020), https://ardupilot.org/planner/.