Learning-Based Vulnerability Analysis of Cyber-Physical Systems

Amir Khazraei
Duke University
Durham, NC
amir.khazraei@duke.edu

Spencer Hallyburton
Duke University
Durham, NC
spencer.hallyburton@duke.edu

Qitong Gao
Duke University
Durham, NC
qitong.gao@duke.edu

Yu Wang
University of Florida
Gainesville, FL
yuwang1@ufl.edu

Miroslav Pajic
Duke University
Durham, NC
miroslav.pajic@duke.edu

ABSTRACT

This work focuses on the use of deep learning for vulnerability analysis of cyber-physical systems (CPS). Specifically, we consider a control architecture widely used in CPS, where the low-level control is based on a feedback controller and an observer (e.g., the extended Kalman filter (EKF)), while also employing an anomaly detector. To facilitate analyzing the impact potential sensing attacks could have on systems with *general nonlinear* dynamics, we develop *learning-enabled attack generators* capable of designing stealthy attacks that maximally degrade system operation. We show how such problem can be cast within a learning-based *grey-box* framework where only parts of the runtime information are known to the attacker. We then introduce two methods for generating *effective stealthy attacks*, based on feed-forward neural networks (FNN) and recurrent neural networks (RNN). Both types of attack-generator models are trained offline, using a cost function that combines the attack impact on the estimation error (and thus control) and the residual signal used for anomaly detection; this enables the trained models to recursively generate effective yet stealthy sensor attacks in real-time while requiring different levels of system information at *runtime*. The effectiveness of the proposed methods is demonstrated on several case studies with varying levels of complexity and nonlinearity: inverted pendulum, autonomous driving vehicles (ADV), and unmanned areal vehicles (UAVs).

KEYWORDS

Security of Cyber-Physical Systems, Deep Learning, Stealthy Attacks, Vulnerability Analysis, Secure Autonomy

1 INTRODUCTION

Although many cyber-physical systems (CPS) operate in safety-critical scenarios and the heterogeneous component connectivity provides numerous possible points of attack, most of existing systems are only weakly protected by legacy components, such as anomaly detectors. The challenge of securing CPS is even more formidable as the long system lifetime and resource constraints prevent the full use of new and existing security mechanisms. On the other hand, security-aware resource allocation can significantly reduce the security-related overhead and thus system cost [26–28]; the idea is to focus on protecting the critical system components and communication links, which if compromised could significantly degrade system performance. Yet, to achieve this, we need methods to analyze system vulnerability, in terms of performance degradation under attack, for different types of attacks.

In this work, we investigate the use of deep learning for the vulnerability analysis of control mechanisms in CPS, focusing on attacks on system sensing. CPS controllers are commonly equipped with a state estimator used for low-level control and anomaly detection. Therefore, attacks on sensing may have tremendous impact on the system performance (i.e., quality of control – QoC), by introducing errors in state estimation. In such setting, to maximize the damage by exploiting the compromised components, the goal of the attacker is to modify sensor measurements delivered to the controller such that the system is forced into an unsafe region, while the attack remains undetected (i.e., stealthy).

Consequently, a critical part of the vulnerability analysis are methods/models for design of effective and stealthy attack vectors. Such *attack generators* should capture how both attack stealthiness and effectiveness are affected by system dynamics, which in general is nonlinear; this prevents the use of existing model-based methods derived for linear time-invariant (LTI) systems (e.g., [24, 43, 44, 47, 50]). To address this challenge, we employ deep learning to develop generators of such effective yet stealthy attack signals (i.e., time series). Specifically, we provide grey-box yet model-free methods that only use the estimator model (and not the plant model) to train stealthy attack generators. We show that to remain stealthy, the attack generator should exhibit a suitable unstable dynamics, resulting *in large (potentially unbounded) attack vectors over time*; this also prevents the use of standard robustness-based analysis techniques that consider performance degradation in the presence of bounded input disturbances.

We propose two attack-generator models for design of such stealthy attack vectors. Each model requires different levels of runtime information from the state estimator – i.e., the current sensor measurements and the previous state estimation, or only the current sensor measurements. The two models, based on feed-forward neural networks (FNN) and recurrent neural networks (RNN), are trained offline using a cost function that captures the impact the attack would have on the estimation error (and thus QoC) as well

This work is sponsored in part by the ONR under agreements N00014-17-1-2504 and N00014-20-1-2745, AFOSR under award number FA9550-19-1-0169, NSF under CNS-1652544 award as well as the National AI Institute for Edge Computing Leveraging Next Generation Wireless Networks, Grant CNS-2112562, and a grant from Intel.

978-1-6654-0967-4/22/$31.00 ©2022 IEEE
DOI 10.1109/ICCPS54341.2022.00030

as stealthiness requirements. To capture the expectation operation in the cost function, we employ Monte Carlo (MC) simulation.

Finally, we illustrate the use and evaluate effectiveness of our approach on three case studies, from inverted pendulum to autonomous driving vehicles (ADVs) and unmanned aerial vehicles (UAVs). We show that when a suitably large time-duration is used for offline training, on average the learned FNN-based attacks slightly outperform the RNN-based attacks. However, this comes at a price; they require additional system information at the runtime – i.e., local state estimation. Furthermore, we demonstrate attack generalizability on a complex case study based on CARLA urban autonomous driving simulator [6]; by training the attack generator models on a simple path and then showing their effectiveness on more complex scenarios. We also show the robustness of the proposed attack design to changes in sensing frequencies.

Related Work. This work is related in spirit to adversarial machine learning focused on methods to generate adversarial examples that degrade performance of deep neural network (DNN) models. The initial work [46] showed that even small perturbations of a DNN's input could drastically change the output, starting a line of research on vulnerability (in terms of robustness) of DNNs. For instance, [12, 40, 46, 51] study vulnerability of classifiers by adding a small perturbation z to the input x, and design an adversarial example $x^* = x + z$ that results in miss-classification error $C(x^*) \neq C(x)$, for some classifier C. In [2, 13, 45], the same idea is applied to self-driving vehicles, where the attacker's goal is to fool a DNN perception model into 'detecting' fake objects in front of the vehicle or removing an existing object, in order to maliciously alter its driving decisions. Some recent works also consider adversarial machine learning beyond the image domain [9, 29, 52]. For example, [29] studies vulnerability of machine learning models applied in CPS by proposing methods for generating adversarial examples that satisfy some physical constraints.

However, the common assumption among such approaches (e.g., [3, 4, 9, 12, 22, 29, 36, 45, 52]) is that the predicted target only depends on its input and not internal dynamics – i.e., previous states; thus, considering bounded perturbation and a single time-instance (i.e., without longitudinal effects) was sufficient in those cases. On the other hand, to address requirements of attacking a system with internal dynamics, in this work, we show that we *have to consider attack models whose output should also depend on the previous outputs (i.e., previous state).* In addition, unlike in the aforementioned works, due to the CPS control perspective, both input and outputs belong to a continuous space.

From the control perspective, it was shown that deep reinforcement learning models are susceptible to adversarial examples [14, 30, 49]. Since the mapping between the observation to actions is achieved by a DNN, the idea has been to add small (i.e., bounded) perturbations on observations to alter the actions in a way that minimizes the expected cumulative reward function, even driving the system to unsafe states [49]. On the other hand, in this work, we show that due to the stealthiness constraint, the time series for an effective additive attacks (on sensor measurements) cannot be bounded; rather, the injected attack signal over time should comply with a certain underlying unstable dynamics that depends on the dynamics of the controlled physical process.

Finally, significant efforts focused on model-based (i.e., using more traditional control techniques) design of effective stealthy attacks on CPS controllers [17, 18, 24, 31, 43, 44, 47, 50], including replay [34], covert [43], zero dynamic [47] and false data injection attacks [50]. However, these methods can be used only for LTI dynamical systems, and thus have limited applicability in practice. For example, [42] designs stealthy sensing attacks on autonomous vehicles. However, the work effectively employs a standard LTI-based attack design where the attack vector is obtained using evolution of the system's linearized model around the equilibrium point. In this work, we show that, as expected, such LTI-based attack designs are only effective in a small neighborhood around the equilibrium point where the linear approximation is valid. As the states move further away from the equilibrium point, the error of the linear approximation significantly increases, resulting in attack detection.

Some recent works have also studied learning-based attack design for control systems [20, 41]. However, they assume the system has an LTI dynamical model as opposed to this work where we design stealthy impactful attacks for general nonlinear systems. To the best of our knowledge, this is the first study focused on design of stealthy attack signals – potentially unbounded (in size) vector time-series – that degrade QoC performance of control systems with general nonlinear dynamics, and for which only limited knowledge of the physical model is available.

Notation. We use $\mathbb{R}$ to denote the set of real numbers, and $\mathbb{P}$ and $\mathbb{E}$ denote the probability and expectation for a random variable. For a matrix A, A^T denotes its transpose and for a square matrix, $trace(A)$ denotes its trace. In addition, I is the identity matrix in general, while I_p is the identity matrix with dimension $p \times p$. Matrix $A \in \mathbb{R}^{n \times n}$ is positive semidefinite (denoted by $A \geq 0$) if $x^T A x \geq 0$ holds for all $x \in \mathbb{R}^n$. For a vector $x \in \mathbb{R}^n$, $||x||_p$ is the p-norm of x; when p is not specified, the 2-norm is implied. Also, $\text{supp}(x)$ denotes the indices of the nonzero elements of $x \in \mathbb{R}^n$ – i.e., $\text{supp}(x) = \{i \mid i \in \{1, ..., n\}, x_i \neq 0\}$. Finally, a function $h : \mathbb{R}^n \to \mathbb{R}^p$ is L-Lipschitz if for any $x, y \in \mathbb{R}^n$ it holds that $||h(x) - h(y)|| \leq L||x - y||$.

2 SYSTEM AND ATTACK MODELS

In this section, we formalize the problem considered in this work. We start from the security-aware system model (i.e., including the attack impact) illustrated in Fig. 1, with each component described in detail as follows.

2.1 System Model

We consider general nonlinear dynamics of a physical system (i.e., plant) compromised by attacks on system sensing, modeled as

$$\begin{aligned} x_{t+1} &= f(x_t, u_t) + w_t, \\ y_t^c &= y_t + a_t = h(x_t) + v_t + a_t. \end{aligned} \tag{1}$$

Here, $x_t \in \mathbb{R}^n$ and $u_t \in \mathbb{R}^m$ denote the plant's state and input vectors at time t, whereas the output vector received by the controller $y_t^c \in \mathbb{R}^p$ contains the measurements from p sensors from the set $\mathcal{S} = \{s_1, ..., s_p\}$, including compromised measurements provided by sensors from the set $\mathcal{K}_a \subseteq \mathcal{S}$; $a_t \in \mathbb{R}^p$ denotes the attack signal injected at time t, and thus the vector is sparse with support in $\mathcal{K}_a$

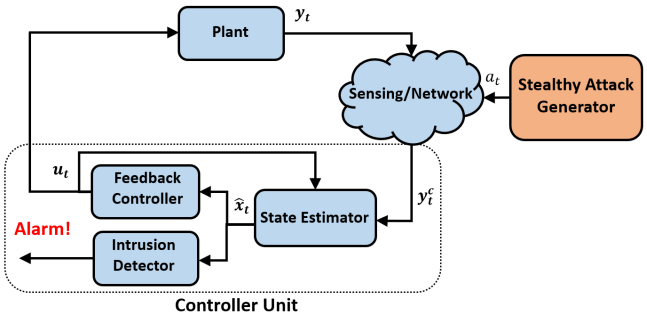

Figure 1: CPS architecture under attacks on system sensing; the considered general attack model captures the impact of both network-based attacks (e.g., man-in-the-middle attacks) and direct sensor attacks (e.g., sensor spoofing).

– i.e., $\text{supp}(a_t) = \mathcal{K}_a$ and $a_{t,i} = 0$ for $i \in \mathcal{K}_a^{\complement}$.[1] The observation function $h : \mathbb{R}^n \to \mathbb{R}^p$ is assumed to be L-Lipschitz. Finally, w_t and v_t are the state and measurement noise, respectfully.

In a special case, if the plant (1) is linear time-invariant (LTI), we use $f(x_t, u_t) = Ax_t + Bu_t$ and $h(x_t) = Cx_t$, where A, B and C are matrices of suitable dimensions.

Control Architecture. We consider a common control architecture, with three main components (as illustrated in Fig. 1): a state estimator, a feedback controller, and an anomaly detector.

The ***State Estimator*** (observer) employs the system model to predict its (state) evolution, and thus provide an estimate $\hat{x}_t$ of its state at time t; in general, this can be captured as

$$\hat{x}_t = O_t(\hat{x}_{t-1}, u_{t-1}, y_t), \quad \hat{y}_t = h(\hat{x}_t). \tag{2}$$

The mapping O_t is commonly designed so that (2) minimizes a norm of the estimation error defined as

$$\Delta x_t = x_t - \hat{x}_t. \tag{3}$$

Depending on the system model and statistical characteristics of the noise, different estimation methods may be employed. Kalman filters are widely used for LTI systems, whereas Extended Kalman filters (EKFs) are mainly utilized for nonlinear systems with Gaussian noise, e.g., the autonomous driving and UAV applications considered in this work. Thus, we particularly focus on EKFs. The EKF functionality for a system (1) is described by

$$\hat{x}_{t|t-1} = f(\hat{x}_{t-1}, u_{t-1}),\ \hat{x}_t = \hat{x}_{t|t-1} + L_t(y_t - h(\hat{x}_{t|t-1})), \hat{y}_t = h(\hat{x}_t);$$

here, $\hat{x}_{t|t-1}$, $\hat{x}_t$ and $\hat{y}_t$ denote the predicted state estimate, (updated) state estimate, and predicted output, respectively. The EKF gain L_t is also updated as

$$\begin{aligned} L_t &= A_t P_t C_t^T \left(C_t P_t C_t^T + R\right)^{-1}, \\ P_{t+1} &= A_t P_t A_t^T + Q - L_t \left(C_t P_t C_t^T + R\right) L_t^T, \end{aligned} \tag{4}$$

where $A_t = \frac{\partial f(x_t, u_t)}{\partial x_t}|_{\hat{x}_{t-1}, u_t}$ and $C_t = \frac{\partial h(x_t)}{\partial x_t}|_{\hat{x}_{t|t-1}}$ are the Taylor expansion of f and h around $(\hat{x}_{t-1}, u_t)$ and $\hat{x}_{t|t-1}$, respectively. Also, Q and R are the covariance matrices of the Gaussian noises w_t and v_t, respectively. Finally, the residue signal (or innovation noise) is defined as

$$z_t = y_t - h(\hat{x}_{t|t-1}). \tag{5}$$

For systems with Gaussian noise, its covariance matrix is $S_t = \mathbb{E}\{z_t z_t^T\} = C_t P_t C_t^T + R_t$ [16].

The ***Feedback Controller*** employs the control law $u_t = \pi(\hat{x}_t)$; without loss of generality, we assume the control goal is to regulate the states to $0 \in \mathbb{R}^n$. Hence, the estimator (2) can be modeled as

$$\hat{x}_t = O_t(\hat{x}_{t-1}, \pi(\hat{x}_{t-1}), y_t) \triangleq O_t(\hat{x}_{t-1}, y_t). \tag{6}$$

The ***Anomaly Detector*** (AD) is used to detect the presence of system anomalies, including intrusions (i.e., attacks). The standard approach is to use the system model to predict the future system behavior and compare it with the actual observation (e.g., see [11] and the references within); capturing the discrepancy between the system and its predicted behavior with a detection function g_t.

In feedback-control based CPS, the residue (5) is widely used for anomaly detection – χ^2 detector in [35, 50], cumulative sum in [48], sequential probability ratio test (SPRT) detector in [25], and a general window-type detector in [15]. For instance, for χ^2-based detectors, the detection function g_t is a weighted norm of z_t (with the χ^2 distribution) – i.e.,

$$g_t = z_t^T S_t^{-1} z_t; \tag{7}$$

the other detectors (e.g., from [15, 23–25, 32, 33, 35, 48, 50]) use some forms of a windowed extension of (7). Therefore, to simplify our presentation, we focus on the detection function g_t from (7), and our results can be directly extended to other cases.

Finally, the system triggers alarm if the detection function satisfies that $g_t > \eta$, for some predefined threshold value η. Usually the value η is assigned such that under normal conditions (i.e., when the system is not compromised) it holds that $\mathbb{P}(g_t > \eta) \leq \epsilon$ – i.e., the system has a low false alarm rate.

2.2 Attack Model

We assume that the attacker has access to the system (or an instance or the model of the system) offline, allowing offline design of a suitable attack generator, which is then employed at runtime to degrade system operation by compromising the sensing measurements.

Attacker capabilities during offline training. Let T be the duration of the training phase; we define $\hat{\mathbf{X}}_{t|t-1} = \{\hat{x}_{0|-1}, \hat{x}_{1|0}, ..., \hat{x}_{t|t-1}\}$, $\mathbf{Y}_t = \{y_0, ..., y_t\}$ and $\mathbf{L}_t = \{L_0, ..., L_t\}$ as the sequences of the predicted states, plant outputs, and EKF gains for $t \geq 0$, with $t = 0$ denoting the time starting the training phase. We assume that the attacker has access to the EKF values over time (either directly, or knowing the EKF design and running a copy of the EKF in parallel) – i.e., has access to $\hat{\mathbf{X}}_{T|T-1}$, $\mathbf{L}_T$, $\mathbf{Y}_T$ and the function h; specifically, the attacker does not need to know the actual function h, but rather its potential approximation used in (2) to implement the state estimator. Meanwhile, for $0 \leq t < T$, the attacker can compromise the sensor measurements according to the model from (1).[2]

[1] We refer to sensors from $\mathcal{K}_a$ as compromised, even if a sensor itself is not directly compromised but its measurements may be altered due to e.g., network-based attacks.

[2] In general, the training (i.e., offline) time is different than the run-time t, as offline training and runtime-deployment are performed on different instances of the system (1). However, to simplify our notation we do not differentiate between these time axes, unless the use of specific time (offline vs. runtime) is not clear from the context.

Attacker capabilities at runtime – i.e., during attack. Let t_0 denote the start time of the attack, modeled as in (1), and T' its duration. Again, we assume that the attacker has access to the sensor measurements y_t. In addition, we will consider two attack scenarios: when the attacker (*i*) **does (i.e., grey-box)**, or (*ii*) **does not (i.e., black-box)** have access to the state estimation $\hat{x}_{t-1}$ in the previous time step; the latter threat model is especially impactful, as it assumes that *the attacker does not have access to the internal controller variables at runtime, but only measurements from the (compromised) system sensors.*

Attacker's goal. is to *maximize degradation of control performance – i.e., QoC.* Specifically, as only sensor data may be compromised, the attack objective is to *maximize the state estimation error* Δx_t. In addition, the attacker wants *to remain stealthy – i.e., undetected by the anomaly detector.* These notions are formalized as follows.

Definition 1. *The sequence of attack vectors $a_{t_0}, a_{t_0+1}, \ldots$ is referred to as (ϵ, α)-successful if there exists $t_0 \leq t' \leq T' + t_0$ such that $\|\Delta x_{t'}\| \geq \alpha$ and $\mathbb{P}(g_t > \eta) \leq \epsilon$ for all $t_0 \leq t \leq T' + t_0$.*[3]

Therefore, the attacker's goal is to insert a sequence of false data measurements $a_{t_0}, ..., a_{t_0+T'}$ resulting in an (ϵ, α)-successful attack. Note that while Definition 1 focuses on attacks that result in a desired norm of the estimation error (i.e., greater than α), for some systems, attacks may cause arbitrarily large estimation errors [15, 24, 50]. For LTI systems with (standard) Kalman filters, the notion of (ϵ, α)-successful attacks was first introduced in [50]. Also, for LTI systems necessary and sufficient conditions such that (ϵ, α)-successful attacks exist for any $\alpha > 0$ are introduced in [15, 24, 50], along with methods to derive such attacks. However, to the best of our knowledge, there is no method for vulnerability analysis of nonlinear systems from (1) under sensor-based attacks; i.e., the impact that such attacks would have on the estimation error, and thus QoC.

3 ADVERSARIAL LEARNING FOR NONLINEAR DYNAMICAL SYSTEMS

In this section, we present methods to design attack-generators for stealthy and effective sensing attacks. Before considering general nonlinear dynamics, we motivate the considered approaches by considering attacks on LTI systems; we start with design of (ϵ, α)-successful attacks against LTI systems with standard Kalman filters.

Lemma 1 ([15, 24, 50]). *For an LTI system with a Kalman filter-based estimator, there exist (ϵ, α)-successful attacks for any desired $\alpha > 0$ if and only if the matrix A is unstable and at least one eigenvector v corresponding to an unstable eigenvalue satisfies that $supp(Cv) \subseteq \mathcal{K}_a$.*

Note that A being unstable is a necessary condition for existence of (ϵ, α)-successful attack for arbitrarily large α, in LTI systems with Kalman filters. However, if all sensors are under attack (i.e., $\mathcal{K}_a = \mathcal{S}$), this is also a sufficient condition. A similar necessary and sufficient condition **only** for LTI systems with bounded noise that employ novel attack-resilient estimators (e.g., from [8, 38, 39]) is derived in [19]. Now, we show how to use only the current state estimation $\hat{x}_t$, the plant output y_t and input u_{t-1} to compute such attack sequence on LTI systems for arbitrarily large α.

Theorem 1. *Consider an LTI system with unstable matrix A, and let ϕ_t denote a Gaussian noise vector satisfying $\mathbb{E}\{\phi_t\} = 0$ and $\mathbb{E}\{\phi_t \phi_t^T\} \leq S$. The attack sequence generated by $a_t = -y_t + CBu_{t-1} + CA\hat{x}_{t-1} + \phi_t$, for $t \geq 0$, is an (ϵ, α)-successful attack for any $\alpha > 0$.*

Proof of the theorem is available in Appendix 6.1.

Unlike in design of adversarial examples for images (e.g., [12, 40, 46, 51]), Theorem 1 shows that for LTI dynamical systems, an effective and stealthy attack sequence has to evolve over time (i.e., following suitable dynamics). This is well aligned with stealthy attack design methods from [24, 50] that do not directly analyze the attack impact on the LTI system but rather consider a dynamical system capturing the difference between the evolutions of the non-attacked and compromised systems.

The fact that effective stealthy attacks should follow certain dynamics (i.e., evolve over time) inspires us to use a similar structure for generating effective stealthy attacks against system with nonlinear dynamics. Specifically, we consider attacks with dynamics $a_t = F(\hat{x}_{t-1}, y_t)$ (note that y_t and $\hat{x}_{t-1}$ are both functions of a_{t-1}), with the idea to use a DNN to learn F. However, the attacker might not always have access to $\hat{x}_{t-1}$ during the attack; for example, if (s)he used an instance of the system to train the generator, and then uses the derived attack signals to insert attacks over the network without access to the internal execution context of the new system (instance) under attack.

Thus, we will consider both cases where information $\hat{x}_{t-1}$ is (*i*) available to the attacker at runtime, and (*ii*) when it is not. For the latter, the idea is to *replace $\hat{x}_t$ with another signal r_t that is directly constructed by the attacker.* We now summarize our models to design (ϵ, α)-successful attack generators for nonlinear systems.

FNN-based attack design. When $\hat{x}_{t-1}$ is available to the attacker, we design an FNN that uses the current output measurements and the last state estimation to generate the next attack vector (for current time) – i.e.,

$$a_t = H_\theta(y_t, \hat{x}_{t-1}), \tag{8}$$

where H_θ is a deep FNN with parameters θ, input dimension of $n + p$ and output dimension p.

RNN-based attack design. When $\hat{x}_{t-1}$ is not available to the attacker, we consider an RNN architecture, that uses only the current sensor measurements for attack design as follows

$$\begin{aligned} r_t &= G_\theta(y_t, r_{t-1}), \\ a_t &= W r_t; \end{aligned} \tag{9}$$

here, G_θ is the state update function implemented as an RNN parameterized by θ, while $W \in \mathbb{R}^{l \times p}$ is a linear mapping from the state r_t to the attack vector a_t. Intuitively, $r_t \in \mathbb{R}^l$ and G_θ should allow for capturing of the evolution of $\hat{x}_t$, to create a dynamical pattern for the sequence of attack vectors in the RNN design (9).

Model training. To capture the attack impact on the estimation error Δx_t from (3), the first challenge is that the actual true system state x_t is not available to the attacker; instead, only sensor measurements y_t are known. We show that under some mild assumptions, the sensor measurements can be directly used.

[3] Note that any p-norm can be used. In this work, when p is not specified, the use of the standard 2-norm is implied.

Theorem 2. *Consider the system* (1). *If the function* $h : \mathbb{R}^n \to \mathbb{R}^p$ *is Lipschitz with constant* L, *then* $\|y_t - h(\hat{x}_t)\| \geq \alpha$ *implies* $\|x_t - \hat{x}_t\| \geq \frac{\alpha - \sqrt{\sigma k}}{L}$ *with probability* $1 - \frac{p}{k^2}$, *for any* k *such that* $k < \frac{\alpha}{\sigma}$ *and* $R \leq \sigma I$; *here,* R *is the covariance matrices of the Gaussian measurement noise* v_t, I *is the identity matrix, and* σ *is a positive scalar.*

Proof of Theorem 2 is provided in Appendix 6.2.

We use $a_t = F_\Theta(y_t, s_{t-1})$ to capture the attack vectors generated by either (8) or (9); here, s_t indicates either r_t or $\hat{x}_t$. Our goal is to train the parameters in (8) and (9) so that the networks act as generators of (ϵ, α)-successful attacks. To achieve this, we use the following approach.

Starting offline training at time $t = 0$, we seek for a_0 that maximizes $\mathbb{E}\|x_0 - \hat{x}_0\|^2$ with $\hat{x}_0 = O(\hat{x}_{-1}, y_0^c)$ and $y_0^c = y_0 + a_0$; here, the expected value operation $\mathbb{E}\{\cdot\}$ is over random variables w and v. As the true state x_0 is not available, from Theorem 2, we can maximize $\mathbb{E}\|y_0 - h(\hat{x}_0)\|^2$ instead. Also, the generated attack should satisfy the stealthiness condition $\mathbb{P}(g_0 > \eta) \leq \epsilon$, with g_0 defined in (7) for $z_0 = y_0^c - \hat{y}_0$; for χ^2-based ADs, it holds that $\hat{y}_0 = h(\hat{x}_{0|-1})$. Thus, the following optimization problem should be solved at time $t = 0$

$$\begin{aligned} &\max_{\Theta} \mathbb{E}\|y_0 - h(\hat{x}_0)\|^2 \\ &s.t.\ \mathbb{P}(g_0 > \eta) \leq \epsilon \\ &a_0 = F_\Theta(y_0, s_{-1}). \end{aligned}$$

As this constrained optimization problem is challenging to solve, we penalize the norm of the residue signal and incorporate a new term in our objective function, resulting in the optimization problem:

$$\begin{aligned} &\min_{\Theta} \mathbb{E}\{g_0 - \delta\|y_0 - h(\hat{x}_0)\|^2\} \\ &a_0 = F_\Theta(y_0, s_{-1}); \end{aligned} \tag{10}$$

here, $\delta > 0$ is a standard regularization term balancing the stealthiness condition and performance degradation caused by the estimation error.

Let us denote the parameters obtained from (10) as $\Theta^{(0)}$. Now, the attack vector $a_0 = F_{\Theta^{(0)}}(y_0, s_0)$ applied to the sensor measurements at time $t = 0$, would results in state estimate $\hat{x}_0 = O(\hat{x}_{-1}, y_0^c)$. In the next (offline) time step ($t = 1$), we search for parameters such that $\mathbb{E}\{g_1 - \delta\|y_1 - h(\hat{x}_1)\|^2\}$ is minimized. However, in this case the parameters will only be trained to minimize the cost function at time $t = 1$ and thus will disregard minimization of the cost function in the previous time step. Hence, the cost function from the previous time step should be also included – i.e., the objective function to be minimized at time $t = 1$ should be be

$$\mathbb{E}\{g_0 + g_{t_0+1} - \delta\left(\|y_0 - h(\hat{x}_0)\|^2 + \|y_1 - h(\hat{x}_1)\|^2\right)\}.$$

This approach should continue for the following (offline) time steps. Generally, if we consider that the training starts at $t = 0$, for any $t \geq 0$ there should be an instantaneous cost function defined by $J_t = g_t - \delta\|y_t - h(\hat{x}_t)\|$. Therefore, the offline optimization problem that is solved at time step t is

$$\begin{aligned} &\min_{\Theta} \mathbb{E}\{J_t + \lambda_t \sum_{j=0}^{t-1} J_j\} \\ &a_t = F_\Theta(y_t, s_{t-1}). \end{aligned} \tag{11}$$

Again, $\lambda_t \geq 0$ are regularization terms to control the incorporation of previous cost functions. If $\lambda_t = 1$, we effectively penalize all previous and current instantaneous costs equally. For smaller values of λ_t, the cost function at time t will be approximately J_t; i.e., we can do more exploration by only minimizing the cost at time t. However, increasing λ_t helps exploit more by giving more importance to the previous cost functions.

Algorithm 1 Learning Stealthy and Effective FNN-based Attack-Generator Models, using MC Simulation

Set the learning rate β, training period T, sample number N
for $t = 0 : T$ **do**
  $x_t^{1:N} = f(x_{t-1}^{1:N}, u_{t-1}^{1:N}) + w_{t-1}^{1:N}$, $y_t^{1:N} = h(x_t^{1:N}) + v_t^{1:N}$
  $\hat{x}_{t|t-1}^{1:N} = f(\hat{x}_{t-1}^{1:N}, u_{t-1}^{1:N})$
  repeat
    $J_t' = \frac{1}{N}\sum_{i=1}^{N}\left(\lambda_t \sum_{j=0}^{t-1} J_j^i + J_t^i\right)$ with $y_j^{c\,1:N} = y_j^{1:N} + H_\theta(\hat{x}_{j-1}^{1:N}, y_j^{1:N})$
    $\theta^{(t)} \leftarrow \theta^{(t)} - \beta\nabla_\theta J_t'$
  until Convergence
  $a_t^{1:N} = H_{\theta^{(t)}}(\hat{x}_{j-1}^{1:N}, y_j^{1:N})$, $y_t^{c\,1:N} = y_t^{1:N} + a_t^{1:N}$
  $\hat{x}_t^{1:N} = \hat{x}_{t|t-1}^{1:N} + L_t^{1:N}(y_j^{c\,1:N} - h(\hat{x}_{t|t-1}^{1:N}))$, $u_t^{1:N} = \pi(\hat{x}_t^{1:N})$

Training Algorithm. In our proposed algorithms, we use Monte Carlo (MC) averaging to approximate the expectation in the cost function (11) by the sample mean over N number of simulated trajectories starting at time $t = 0$. Specifically, the i-th, $i = 1, ..., N$, trajectory at time t is obtained by

$$x_t^i = f(x_{t-1}^i, u_{t-1}^i) + w_{t-1}^i, \quad y_t^i = h(x_t^i) + v_t^i, \quad \hat{x}_{t|t-1}^i = f(\hat{x}_{t-1}^i, u_{t-1}^i).$$

The cost that the trajectory i at time t imposes is $J_t^i = g_t^i - \delta\|y_t^i - h(\hat{x}_t^i)\|$. However, at time step t the DNN is trained such that $J_t' = \frac{1}{N}\sum_{i=1}^{N}\left(\lambda_t \sum_{j=0}^{t-1} J_j^i + J_t^i\right)$ is minimized, where J_t' is the approximated expectation of cost function in (11). Finally, once the model parameters are obtained at time t, the attack vector $a_t^i = F_{\Theta^{(t)}}(y_t^i, s_{t-1}^i)$ is applied to the system output y_t^i, and the process is repeated until the training completes.

When the EKF is used, Algorithm 1 captures pseudocode for learning the FNN-based stealthy attack generator, and Algorithm 2 summarizes pseudocode to learn the RNN-based attack generators.

4 CASE STUDIES

We illustrate and thoroughly evaluate our attack-design framework on three case studies, inverted pendulum, autonomous driving vehicles (ADVs) and unmanned aerial vehicles (UAVs), with varying level of complexity due to system dynamics. For all case studies, the workstation used for training is powered by Nvidia RTX 6000 GPUs with 24 GB of memory each, two Intel Xeon Silver 4208 CPUs with 16 cores each, and a total of 192 GB RAM. The code was developed using Python and Pytorch deep learning libraries.

4.1 Inverted Pendulum

To illustrate the effectiveness of our algorithm compared to the LTI-based methods from [15, 50], we first considered a fixed-base inverted pendulum. We used the nonlinear dynamical model from [10]. Two sensors were used to measure the states of the system, θ and $\dot{\theta}$, where θ was the angle of pendulum rod from the vertical axis

Algorithm 2 Learning Stealthy and Effective RNN-based Attack Generator Models, using MC Simulation

Set the learning rate β, training period T, sample number N
for $t = 0 : T$ **do**
  $x_t^{1:N} = f(x_{t-1}^{1:N}, u_{t-1}^{1:N}) + w_{t-1}^{1:N}$
  $y_t^{1:N} = h(x_t^{1:N}) + v_t^{1:N}$
  $\hat{x}_{t|t-1}^{1:N} = f(\hat{x}_{t-1}^{1:N}, u_{t-1}^{1:N})$
  repeat
    $J'_t = \frac{1}{N}\sum_{i=1}^{N}\left(\lambda_t \sum_{j=0}^{t-1} J_j^i + J_t^i\right)$ with $y_t^{c\,1:N} = y_j^{1:N} + WG_\theta(r_{j-1}^{1:N}, y_j{}^{1:N})$
    $\theta^{(t)} \leftarrow \theta^{(t)} - \beta\nabla_\theta J'_t$
    $W^{(t)} \leftarrow W^{(t)} - \beta\nabla_W J'_t$
  until Convergence
  $r_t^{1:N} = G_{\theta^{(t)}}(r_{t-1}^{1:N}, y_t^{1:N})$
  $a_t^{1:N} = W^{(t)} r_t^{1:N}$
  $y_t^{c\,1:N} = y_t^{1:N} + a_t^{1:N}$
  $\hat{x}_t^{1:N} = \hat{x}_{t|t-1}^{1:N} + L_t^{1:N}(y_t^{c\,1:N} - h(\hat{x}_{t|t-1}^{1:N}))$
  $u_t^{1:N} = \pi(\hat{x}_t^{1:N})$

Table 1: Attack success rate (SR) for different training duration T with $N = 200$, $\lambda = .5$ and varying α and θ over 100 inverted pendulum experiments; FNN/RNN denote FNN vs. RNN success rates.

SR % FNN/RNN	$\alpha = .5$	$\alpha = 1$	$\alpha = 2$	$\alpha = 4$	$\alpha = 8$	$\lvert\theta\rvert = \frac{\pi}{8}$	$\lvert\theta\rvert = \frac{\pi}{4}$	$\lvert\theta\rvert = \frac{\pi}{3}$	$\lvert\theta\rvert = \frac{\pi}{2}$	$\lvert\theta\rvert = \pi$
$T = 50$	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0
$T = 80$	100/99	26/67	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0
$T = 100$	100/100	100/100	100/78	0/3	0/0	100/81	0/9	0/0	0/0	0/0
$T = 120$	100/100	100/100	100/100	100/100	0/0	100/100	100/100	100/99	0/52	0/0
$T = 150$	100/100	100/100	100/100	100/100	89/97	100/100	100/100	100/100	100/100	50/58
$T = 170$	100/100	100/100	100/100	100/100	100/100	100/100	100/100	100/100	100/100	100/100
SR % LTI Model	100	100	100	10	0	100	14	0	0	0

Table 2: Attack success rate (SR) for different values of N with $T = 150$, $\lambda = .5$ and different values of α and θ over 100 experiments for the inverted pendulum. The FNN/RNN numbers denote FNN vs. RNN success rate values.

SR % FNN/RNN	$\alpha = .5$	$\alpha = 1$	$\alpha = 2$	$\alpha = 4$	$\alpha = 8$	$\lvert\theta\rvert = \frac{\pi}{8}$	$\lvert\theta\rvert = \frac{\pi}{4}$	$\lvert\theta\rvert = \frac{\pi}{3}$	$\lvert\theta\rvert = \frac{\pi}{2}$	$\lvert\theta\rvert = \pi$
$N = 1$	35/4	35/2	35/1	35/1	10/0	35/1	35/1	35/1	35/0	0/0
$N = 10$	100/19	100/10	100/4	100/2	45/1	100/4	100/2	100/1	100/1	0/0
$N = 50$	100/74	100/66	100/56	100/38	100/4	100/56	100/39	100/32	100/9	0/0
$N = 100$	100/96	100/96	100/96	100/95	90/64	100/96	100/96	100/95	100/90	15/1
$N = 250$	100/100	100/100	100/100	100/100	95/97	100/100	100/100	100/100	100/100	94/95
$N = 500$	100/100	100/100	100/100	100/100	100/100	100/100	100/100	100/100	100/100	100/99

measured counterclockwise. The threshold η was set to have $\epsilon = .01$ (i.e., on average, every one hundred time steps a false alarm occurs). A feedback controller was used to keep the pendulum inverted around $\theta = 0$ equilibrium point.

Table 3: Attack success rate (SR) for different values of λ with $T = 100$, $N = 200$ and different values of α and θ over 100 experiments for the inverted pendulum. The FNN/RNN numbers denote FNN vs RNN success rate values.

SR % FNN/RNN	$\alpha = .5$	$\alpha = 1$	$\alpha = 2$	$\alpha = 4$	$\alpha = 8$	$\lvert\theta\rvert = \frac{\pi}{8}$	$\lvert\theta\rvert = \frac{\pi}{4}$	$\lvert\theta\rvert = \frac{\pi}{3}$	$\lvert\theta\rvert = \frac{\pi}{2}$
$\lambda = .01$	100/12	100/2	100/1	50/1	0/0	100/1	50/1	0/0	0/0
$\lambda = .05$	100/98	100/97	100/90	58/0	0/0	100/92	53/0	0/0	0/0
$\lambda = .1$	100/98	100/97	100/91	73/38	0/0	100/93	55/45	0/8	0/0
$\lambda = 1$	100/100	100/100	100/70	97/59	0/0	100/75	100/60	54/3	0/0
$\lambda = 5$	100/100	100/100	100/100	70/67	0/0	100/100	90/78	10/27	0/0
$\lambda = 100$	100/100	100/64	100/42	0/12	0/0	100/43	1/16	0/0	0/0

Table 4: Attack success rate (SR) for different values of training duration T with $N = 200$, $\lambda = .5$ and different values of α and y over 100 experiments for the ADV.

SR % FNN/RNN	$\lvert y\rvert = .2$	$\lvert y\rvert = 1$	$\lvert y\rvert = 2$	$\lvert y\rvert = 4$	$\lvert y\rvert = 6$	$\lvert y\rvert = 8$	$\lvert y\rvert = 10$	$\lvert y\rvert = 14$
$T = 100$	100/100	0/0	0/0	0/0	0/0	0/0	0/0	0/0
$T = 200$	100/100	98/100	16/0	0/0	0/0	0/0	0/0	0/0
$T = 300$	100/100	100/100	100/100	32/100	0/0	0/0	0/0	0/0
$T = 500$	100/100	100/100	100/100	100/99	100/99	50/96	23/96	0/0
$T = 700$	100/100	100/100	100/100	100/100	100/100	100/100	100/100	99/82

First, we trained both FNN and RNN models with $N = 200$ MC samples, $\lambda = .5$ and different values of training period T when both sensors are compromises. Table 1 shows the success rate (in percentage) of both attack generator models, which were obtained for each training period T, and for different values of α and θ. For example, consider the entry associated with row $T = 150$ and $\alpha = 8$. It shows that 89 percent of times the FNN and 97 percent of times the RNN model was successful in driving the system to $\alpha = 8$, with α being the norm of the state estimation error. This also applies to the columns shown by $|\theta|$; for example, the entry in row $T = 150$ and column $|\theta| = \frac{\pi}{2}$ means that 100 percent of times both FNN and RNN models were successful in driving the pendulum rod angle to more than $\frac{\pi}{2}$ degrees.

As summarized in Table 1, when trained with $T = 50$, neither of the learned models were successful for any values of α and θ. However, by increasing the training duration T, the performance of both designed attack generator models improved. Specifically, for $T = 170$ both learned models were able to drive the pendulum rod to fall without being detected. Also, only for shorter training durations T, RNN slightly outperformed FNN, since the RNN was better than FNN in capturing the optimal non-linear attack dynamics in parts of the state-space not explored during the training.

We also generated stealthy attacks on both sensors using the model-based LTI method from [15, 24, 50] that employs the linearized model of the system dynamics around the zero equilibrium point. Attacks generated using the LTI model were only successful for smaller errors – i.e., $\alpha < 4$ and $\theta < \frac{\pi}{3}$. This shows that the ***LTI-based attacks are only effective around the equilibrium point, where the linearization error is small***.

Table 5: Attack success rate (SR) for different values of N with $T = 300$, $\lambda = .5$ and different values of y over 100 experiments for the ADV; FNN/RNN denote FNN vs. RNN SRs.

SR % FNN/RNN	$\|y\| = .2$	$\|y\| = .6$	$\|y\| = .8$	$\|y\| = 1.2$	$\|y\| = 1.5$	$\|y\| = 2$	$\|y\| = 3$	$\|y\| = 4$
$N = 1$	100/53	75/3	75/1	75/0	73/0	0/0	0/0	0/0
$N = 10$	100/50	100/41	100/32	100/18	100/12	0/6	0/1	0/0
$N = 50$	100/99	100/99	100/99	100/99	100/99	100/99	0/57	0/0
$N = 100$	100/100	100/100	100/100	100/100	100/100	100/100	21/100	0/0
$N = 500$	100/100	100/100	100/100	100/100	100/100	100/100	50/100	0/100

Table 2 shows the effect of the MC sample number on the performance of the learned attack-generators for a fixed $T = 150$. For smaller values of N, FNN performed better; however, when both models had enough data (e.g., $N \geq 250$), both worked equally well. Similarly, Table 3 shows how different values of λ in training phase affect the attack model performance. For smaller values of λ, the obtained RNN worked poorly because the data in previous time steps were important to train the model, and accurately capture the desired attack dynamics. On the other hand, the learned FNN generator performed worse for very large values of λ.

4.2 Autonomous Driving Vehicles

4.2.1 Generic Vehicle Model. We first considered a simple nonlinear dynamical model of ADV from [21], with four states $[x\ y\ \psi\ v]^T$; here, x and y represent the position of the center of mass in x and y axis, respectively, ψ is the inertial heading, and v is the velocity of the vehicle. The states x, y, ψ were measured using noisy sensors, with zero-mean noise with covariance matrix $R = .01I$. The system noise was zero-mean, with covariance $Q = .001I$ and we set the threshold η to achieve $\epsilon = .01$.

We considered scenario where the car had a constant speed of $25m/s$, with a feedback controller keeping the car between the lanes. We trained offline the FNN and RNN models for generating effective stealthy attacks. The network H_θ was fully connected with 20 neurons and the ReLU activation function, whereas G_θ was an RNN with one layer and 20 neurons, and the ReLU activation function.

First, we trained both models with $N = 200$ MC samples, $\lambda = .5$ and different duration of training period T; we attacked only sensors that measure y and ψ (i.e., not all of the sensors were compromised). Table 4 shows the success rate (in %) for both learned attack models for each training period T and different values of α and $|y|$, the car's distance from the center of the lane. As summarized, *increasing the attack training duration helps learn attack generators that can drive the system towards the unsafe region* (i.e., increasing $|y|$).

We also analyzed the impact of N, the MC sampling number during training, on attack performance (Table 5); we showed that using $N \geq 100$ in training is sufficient. Specifically, for smaller values of N, the learned FNN-based generator outperformed the RNN model. However, as N increased both models performed equally good, and even RNN performed slightly better.

Figure 2(a) shows the trajectory of the car. Before starting the attack at the location $X = 75m$, the car (blue line) was kept between the lanes and the estimated trajectory (green line) had a very small estimation error. However, using either attacks derived by the FNN or RNN-based attack generators, the car was being pushed off the road while the estimated position showed that the car was still in the road between the lanes. Furthermore, the attacks were stealthy – the AD could not detect the presence of either of the attacks. Figure 2(b),(c) show the estimation error along the X and Y axis for these scenarios.

4.2.2 Autonomous Driving Simulator: Evaluating on CARLA. To evaluate our methodology on complex, realistic systems, for which we do not know the model of the non-linear vehicle dynamics, we used ADV scenarios in vehicle simulator CARLA [6]. CARLA is an urban driving simulator built on Unreal Engine 4, and providing realistic physics and sensor models in complex urban environments with static and dynamic actors. We defined a planning-navigation-control loop that drove the autonomous vehicle, leveraging the EKF structure for χ^2 anomaly detector; CARLA setup details are presented in Appendix 6.3 and the videos of our CARLA experiments are available at [5].

We evaluated our FNN and RNN attack-generators for *performance and generalizability*, and compared to the nominal case without attacks. We demonstrated how both FNN and RNN-based attack generators were able to drive the vehicle into unsafe situations (e.g., crashes into other cars or static objects) over short times, while remaining undetected. We highlight here the results when *not all of the sensors were compromised* (i.e., $\mathcal{K}_a \neq \mathcal{S}$) – i.e., when the FNN and RNN-based attack generators *were only able to attack GNSS position measurements and to only have knowledge of positions states* (i.e., no knowledge of velocity or heading). Despite these restrictions, both attack generators produced stealthy attacks that significantly moved the vehicle off-course (e.g., resulting in collisions – see videos at [5]).

Additionally, we demonstrated the attack generalizability twofold. First, we trained the FNN and RNN-based models offline on a simple path (i.e., not the testing path). We then tested those same models on the full CARLA environment and paths. Second, we demonstrated the proof-of-concept that the learned attack generator models were robust to changes in rates of sensor data by training at 100 Hz measurements and testing at 120 Hz measurements, and retaining attack stealthiness and effectiveness.

Figure 3 presents some of the results. Specifically, Figure 3(d) shows the path and residue values when the sensors were **not** under attack. Figure 3(e) and Figure 3(f) show the path and the residue signal values of the compromised car when the FNN and RNN-based generators were used to create inserted attack signals. The endpoint of the path is when the car hits an object and stops moving – the residue signal has a spike only when the car hits the object (due to the collision); by this point it was too late for any recovery/avoidance action.

4.3 Unmanned Aerial Vehicles

Finally, we considered a quadrotor with complex highly nonlinear model from [1] that has 12 states $\left[x,\ y,\ z,\ \psi,\ \theta,\ \phi,\ \dot{x},\ \dot{y},\ \dot{z},\ \dot{\psi},\ \dot{\theta},\ \dot{\phi}\right]^T$; x, y and z represent the quadrotor position along the X, Y and Z axis, respectively, while $\dot{x}$, $\dot{y}$ and $\dot{z}$ are their velocity. ψ, θ and ϕ are yaw, pitch and roll angles respectively, and $\dot{\psi}$, $\dot{\theta}$ and $\dot{\phi}$ represent their corresponding angular velocity. The system was discretized using

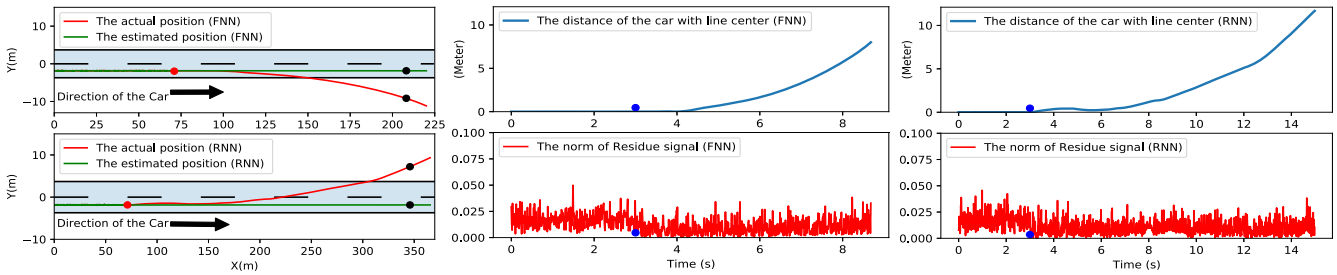

Figure 2: (a) The trajectory of the compromised car (green - the estimated, and the red - the actual vehicle position; the red dot shows the place where the attack started, the black dots show the actual and estimated position of the car at the same time). (b,c) The above sub-figures show the distance of the car with the center of the lane; the bellow sub-figures illustrate the norm of residue signal before and after the start time of attack $t = 3s$ (the blue dots) for each of the FNN and RNN-based generators.

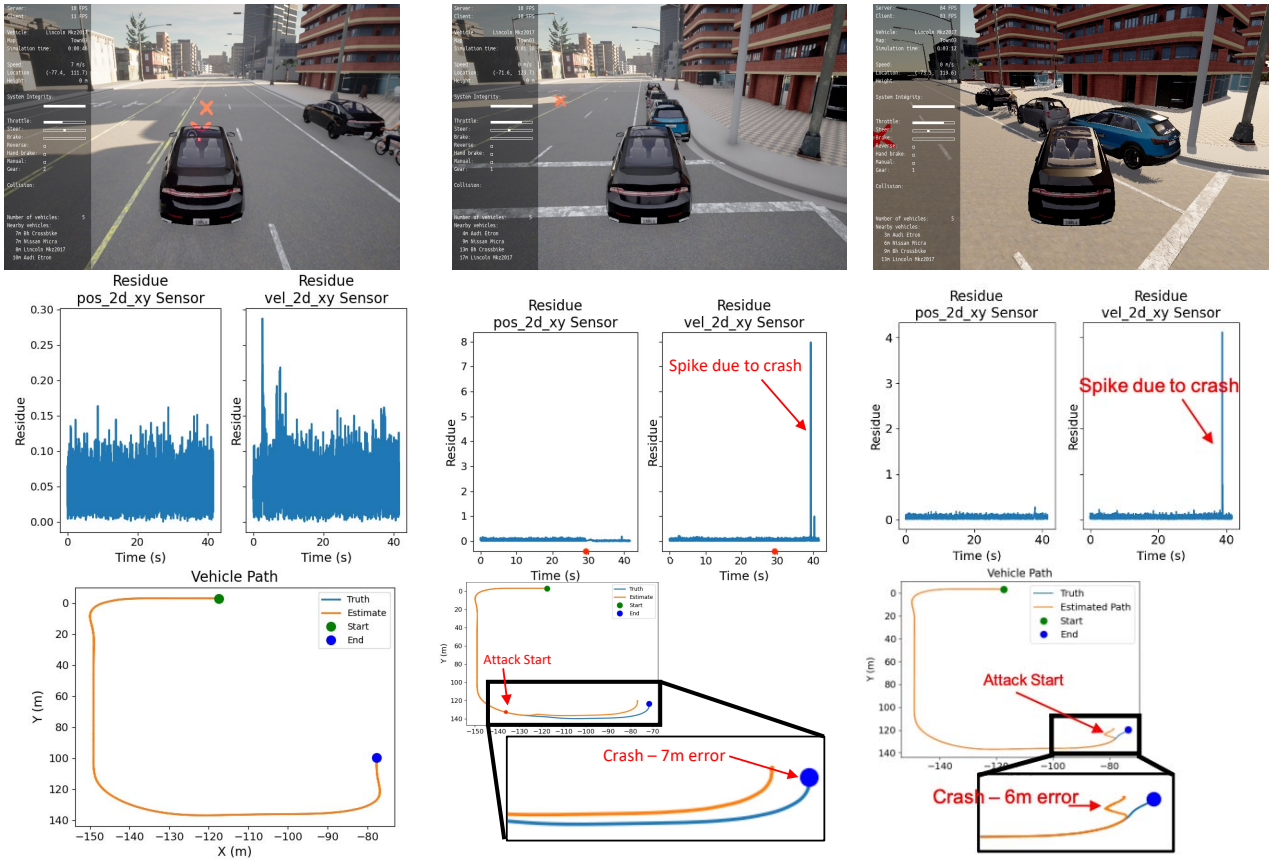

Figure 3: Example results from evaluations on CARLA scenarios: (a) CARLA simulation when the car is free of attack; (b,c) The vehicle collisions with off-road objects due to the injecting sensor attacks using the FNN and RNN-based attack generators, respectively; (d) The vehicle trajectory without attack and the residue signals for both velocity and position sensors; (e,f) The trajectory when the position sensors are compromised using the FNN and RNN-based methods, respectively, and the corresponding residue signals (note different $y-$axes scaling on subfigures (d)-(f).

Euler method with $T_s = .01s$. The states $\left[x,\ y,\ z,\ \psi,\ \theta,\ \phi, \dot{\psi},\ \dot{\theta},\ \dot{\phi}\right]^T$ were measured and were affected by zero-mean Gaussian noise with the covariance matrix $R = .01I$. We assumed standard disturbance on the input modeled by system noise with zero mean Gaussian with the covariance matrix $Q = .001I$. We also set η such that we obtained $\epsilon = .01$. We considered the position control task [1], where the drone should reach a predefined height ($10m$) and stay there – i.e., stay at coordinates $X = 0$, $Y = 0$ and $Z = 10$ if the initial point was denoted as $(0, 0, 0)$. UAV control for this task was based on a standard feedback-based controller.

Table 6: Attack success rate (SR) for different values of T with $N = 300$, $\lambda = .5$ and different values of α over 100 experiments for the UAV.

SR % FNN/RNN	$\alpha = .2$	$\alpha = .5$	$\alpha = .7$	$\alpha = 1$	$\alpha = 3$	$\alpha = 5$	$\alpha = 7$	$\alpha = 9$
$T = 100$	100/100	100/87	89/49	0/6	0/0	0/0	0/0	0/0
$T = 200$	100/100	95/58	65/53	32/7	0/0	0/0	0/0	0/0
$T = 400$	100/100	100/100	100/100	67/100	0/21	0/0	0/0	0/0
$T = 600$	100/100	100/100	100/100	100/100	80/63	4/9	0/0	0/0
$T = 800$	100/100	100/93	100/91	100/91	99/83	86/68	69/57	53/43
SR % LTI Model	100	62	0	0	0	0	0	0

Table 7: Attack success rate (SR) for different values of T with $N = 300$, $\lambda = .5$ and different values of x along X axis over 100 experiments for the UAV.

SR % FNN/RNN	$\|x\| = .2$	$\|x\| = .4$	$\|x\| = .6$	$\|x\| = .8$	$\|x\| = 1$	$\|x\| = 1.5$	$\|x\| = 3$	$\|x\| = 4$
$T = 100$	93/50	1/2	0/0	0/0	0/0	0/0	0/0	0/0
$T = 200$	47/5	17/0	5/0	1/0	0/0	0/0	0/0	0/0
$T = 400$	100/57	44/14	0/5	0/0	0/0	0/0	0/0	0/0
$T = 600$	83/78	66/24	50/4	37/2	30/1	13/0	0/0	0/0
$T = 800$	97/77	90/36	85/22	75/13	68/10	51/1	0/0	0/0

G_θ used for synthesizing the stealthy attack was a 2-layer RNN with ReLU activation function and 55 neurons per layer. H_θ was also a 2-layer FNN with ReLU activation function and 55 neuron for each layer. First, we trained both models with $N = 300$ MC samples, $\delta = .2$, $\lambda = .5$ and different values of training period T. We also considered the case where all sensor are under attack.

Tables 6-9 (due to space constraints, Tables 8 and 9 are in Appendix 6.4) show the success rate (in %) for both learned attack generators obtained for each training period T and different values of α, $|x|$, $|y|$ and $|z|$ (i.e., the drone's distance from the desired position along each axis). As summarized, increasing the attack training duration helps learn effective stealthy attack generators capable of driving the system towards the unsafe region (i.e., increasing $|x|$, $|y|$ and $|z|$). Furthermore, for suitably large training periods T, on average the learned FNN-based attack generators outperforms the RNN-based attack generators. Moreover, we evaluated the effectiveness of the LTI-based attacks (i.e., which linearize the UAV model) [15, 24, 50]; our results, summarized in the last line in Table 6, show that the LTI-based attacks are only 62% successful in reaching $\alpha = .2$, and unsuccessful for higher values of α, for the same reasons as in the pendulum study – linearization error becomes too large for large state deviations, limiting their applicability.

Figure 4(a) illustrates the deviation of the drone from the desired hovering point over time for a successful attack sequence obtained from a generator trained with $T = 800$ and $N = 300$. The attack started at $t = 0$; over time, the drone's deviation from the desired position will increase. Figure 4(b) shows the norm of the attack vector of both FNN and RNN models for three different attack-generator models trained with $T = 200, 400$ and 600. Note that unlike adversarial machine learning in other domains (e.g., image classification) where the norm of the attack is limited to be bounded, the stealthiness condition in CPS requires the norm of the attack vector to gradually increase over time.

5 CONCLUSION

In this work, we have utilized deep learning to generate stealthy attacks on control components in cyber-physical systems, focusing on a widely used architecture where the low-level control employs the extended Kalman filter and an anomaly detector. We have considered a grey box setup, with unknown nonlinear plant dynamics and known observation functions and Kalman filter gains. We have shown that feedforward and recurrent neural networks (FNN and RNN, respectively) can be used to generate stealthy adversarial attacks on sensing information delivered to the system, resulting in large errors to the estimates of the state of the system without being detected. Both FNN and RNN are trained offline from a cost function combining the attack effects on the estimation error and the residual signal of the EKF; thus, the trained model is capable of recursively generating such effective sensor attacks in real-time using only current sensor measurements. The effectiveness of the proposed methods has been illustrated and evaluated on several case studies with varying complexity.

REFERENCES

[1] Samir Bouabdallah and Roland Siegwart. 2007. Full control of a quadrotor. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 153–158.

[2] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Z Morley Mao. 2019. Adversarial sensor attack on lidar-based perception in autonomous driving. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*. 2267–2281.

[3] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*. 39–57.

[4] Francesco Croce and Matthias Hein. 2020. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning*. PMLR, 2196–2205.

[5] CSPL@Duke. 2021. Attacks on Autonomous Driving in CARLA. https://cpsl.pratt.duke.edu/research/security-aware-cps.

[6] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. 2017. CARLA: An open urban driving simulator. In *Conference on robot learning*. PMLR, 1–16.

[7] Mümin Tolga Emirler, İsmail Meriç Can Uygan, Bilin Aksun Güvenç, and Levent Güvenç. 2014. Robust PID steering control in Parameter Space for highly automated driving. *International Journal of Vehicular Technology* 2014 (2014).

[8] Hamza Fawzi, Paulo Tabuada, and Suhas Diggavi. 2014. Secure estimation and control for cyber-physical systems under adversarial attacks. *IEEE Transactions on Automatic control* 59, 6 (2014), 1454–1467.

[9] Cheng Feng, Tingting Li, Zhanxing Zhu, and Deeph Chana. 2017. A Deep Learning-Based Framework for Conducting Stealthy Attacks in Industrial Control Systems. *arXiv:1709.06397 [cs]* (2017).

[10] AM Formal'skii. 2006. An inverted pendulum on a fixed and a moving base. *Journal of applied mathematics and mechanics* 70, 1 (2006), 56–64.

[11] Jairo Giraldo, David Urbina, Alvaro Cardenas, Junia Valente, Mustafa Faisal, Justin Ruths, Nils Ole Tippenhauer, Henrik Sandberg, and Richard Candell. 2018. A survey of physics-based attack detection in cyber-physical systems. *ACM Computing Surveys (CSUR)* 51, 4 (2018), 1–36.

[12] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[13] R. Spencer Hallyburton, Yupei Liu, Yulong Cao, Z. Morley Mao, and Miroslav Pajic. 2022. Security Analysis of Camera-LiDAR Fusion Against Black-Box Attacks on Autonomous Vehicles. In *31th {USENIX} Security Symposium ({USENIX} Security 22)*.

[14] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel. 2017. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284* (2017).

[15] I. Jovanov and M. Pajic. 2019. Relaxing Integrity Requirements for Attack-Resilient Cyber-Physical Systems. *IEEE Trans. Automat. Control* 64, 12 (Dec 2019), 4843–4858.

[16] Simon J Julier and Jeffrey K Uhlmann. 2004. Unscented filtering and nonlinear estimation. *Proc. IEEE* 92, 3 (2004), 401–422.

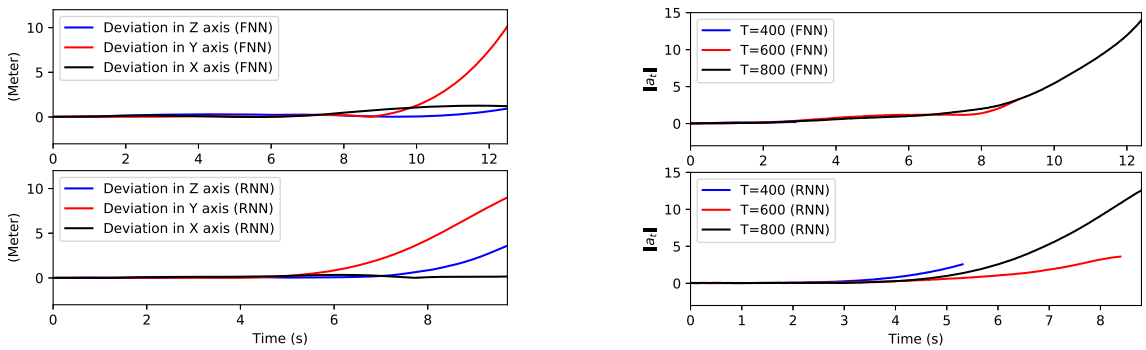

Figure 4: UAV altitude control: (a) The deviation of the drone along each axis from the desired coordinates $X = 0$, $Y = 0$ and $Z = 10$, for RNN and FNN-based attack (the attack starts at time zero); (b) The norm of the attack vector over time for both FNN and RNN attack generators with $T = 400, 600, 800$.

[17] Amir Khazraei, Hamed Kebriaei, and Farzad Rajaei Salmasi. 2017. A new watermarking approach for replay attack detection in lqg systems. In *56th IEEE Annual Conf. on Decision and Control (CDC)*. 5143–5148.

[18] A. Khazraei and M. Pajic. 2020. Perfect Attackability of Linear Dynamical Systems with Bounded Noise. In *2020 American Control Conference (ACC)*. 749–754.

[19] A. Khazraei and M. Pajic. 2021. Attack-Resilient State Estimation with Intermittent Data Authentication. *Automatica* (2021).

[20] Mohammad Javad Khojasteh, Anatoly Khina, Massimo Franceschetti, and Tara Javidi. 2020. Learning-based attacks in cyber-physical systems. *IEEE Transactions on Control of Network Systems* 8, 1 (2020), 437–449.

[21] Jason Kong, Mark Pfeiffer, Georg Schildbach, and Francesco Borrelli. 2015. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 1094–1099.

[22] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533* (2016).

[23] Cheolhyeon Kwon and Inseok Hwang. 2017. Reachability analysis for safety assurance of cyber-physical systems against cyber attacks. *IEEE Trans. Automat. Control* 63, 7 (2017), 2272–2279.

[24] Cheolhyeon Kwon, Weiyi Liu, and Inseok Hwang. 2014. Analysis and design of stealthy cyber attacks on unmanned aerial systems. *Journal of Aerospace Information Systems* 11, 8 (2014), 525–539.

[25] Cheolhyeon Kwon, Scott Yantek, and Inseok Hwang. 2016. Real-time safety assessment of unmanned aircraft systems against stealthy cyber attacks. *Journal of Aerospace Information Systems* 13, 1 (2016), 27–45.

[26] V. Lesi, I. Jovanov, and M. Pajic. 2017. Network Scheduling for Secure Cyber-Physical Systems. In *2017 IEEE Real-Time Systems Symposium (RTSS)*. 45–55.

[27] Vuk Lesi, Ilija Jovanov, and Miroslav Pajic. 2017. Security-Aware Scheduling of Embedded Control Tasks. *ACM Trans. Embed. Comput. Syst.* 16, 5s, Article 188 (Sept. 2017), 21 pages.

[28] Vuk Lesi, Ilija Jovanov, and Miroslav Pajic. 2020. Integrating Security in Resource-Constrained Cyber-Physical Systems. *ACM Trans. Cyber-Phys. Syst.* 4, 3, Article 28 (May 2020), 27 pages.

[29] Jiangnan Li, Jin Young Lee, Yingyuan Yang, Jinyuan Stella Sun, and Kevin Tomsovic. 2020. ConAML: Constrained Adversarial Machine Learning for Cyber-Physical Systems. *arXiv:2003.05631 [cs]* (2020).

[30] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. 2017. Tactics of adversarial attack on deep reinforcement learning agents. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 3756–3762.

[31] Yao Liu, Peng Ning, and Michael K Reiter. 2011. False data injection attacks against state estimation in electric power grids. *ACM Transactions on Information and System Security (TISSEC)* 14, 1 (2011), 1–33.

[32] Fei Miao, M. Pajic, and G.J. Pappas. 2013. Stochastic game approach for replay attack detection. In *IEEE 52nd Annual Conference on Decision and Control (CDC)*. 1854–1859. https://doi.org/10.1109/CDC.2013.6760152

[33] F. Miao, Q. Zhu, M. Pajic, and G. J. Pappas. 2017. Coding Schemes for Securing Cyber-Physical Systems Against Stealthy Data Injection Attacks. *IEEE Transactions on Control of Network Systems* 4, 1 (March 2017), 106–117.

[34] Yilin Mo and Bruno Sinopoli. 2009. Secure control against replay attacks. In *47th Annual Allerton Conference on Communication, Control, and Computing*. 911–918.

[35] Yilin Mo and Bruno Sinopoli. 2015. On the performance degradation of cyber-physical systems under stealthy integrity attacks. *IEEE Trans. Automat. Control* 61, 9 (2015), 2618–2624.

[36] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2574–2582.

[37] Jorge Navarro. 2016. A very simple proof of the multivariate Chebyshev's inequality. *Communications in Statistics-Theory and Methods* 45, 12 (2016), 3458–3463.

[38] M. Pajic, I. Lee, and G. J. Pappas. 2017. Attack-Resilient State Estimation for Noisy Dynamical Systems. *IEEE Transactions on Control of Network Systems* 4, 1 (March 2017), 82–92.

[39] M. Pajic, J. Weimer, N. Bezzo, O. Sokolsky, G. J. Pappas, and I. Lee. 2017. Design and Implementation of Attack-Resilient Cyberphysical Systems: With a Focus on Attack-Resilient State Estimators. *IEEE Control Systems Magazine* 37, 2 (April 2017), 66–81.

[40] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*. 506–519.

[41] Anshuka Rangi, Mohammad Javad Khojasteh, and Massimo Franceschetti. 2021. Learning based attacks in Cyber Physical Systems: Exploration, Detection, and Control Cost trade-offs. In *Learning for Dynamics and Control*. PMLR, 879–892.

[42] Junjie Shen, Jun Yeon Won, Zeyuan Chen, and Qi Alfred Chen. 2020. Drift with devil: Security of multi-sensor fusion based localization in high-level autonomous driving under {GPS} spoofing. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*. 931–948.

[43] Roy S Smith. 2015. Covert misappropriation of networked control systems: Presenting a feedback structure. *IEEE Control Systems Magazine* 35, 1 (2015), 82–92.

[44] Tianju Sui, Yilin Mo, Damian Marelli, Xi-Ming Sun, and Minyue Fu. 2020. The Vulnerability of Cyber-Physical System under Stealthy Attacks. *IEEE Trans. Automat. Control* (2020).

[45] Jiachen Sun, Yulong Cao, Qi Alfred Chen, and Z Morley Mao. 2020. Towards robust lidar-based perception in autonomous driving: General black-box adversarial sensor attack and countermeasures. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*. 877–894.

[46] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199* (2013).

[47] André Teixeira, Iman Shames, Henrik Sandberg, and Karl H Johansson. 2012. Revealing stealthy attacks in control systems. In *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 1806–1813.

[48] Rohit Tunga, Carlos Murguia, and Justin Ruths. 2018. Tuning windowed chi-squared detectors for sensor attacks. In *2018 Annual American Control Conference (ACC)*. IEEE, 1752–1757.

[49] Tsui-Wei Weng, Krishnamurthy (Dj) Dvijotham*, Jonathan Uesato*, Kai Xiao*, Sven Gowal*, Robert Stanforth*, and Pushmeet Kohli. 2020. Toward Evaluating Robustness of Deep Reinforcement Learning with Continuous Control. In *International Conference on Learning Representations*.

[50] Y. Mo and B. Sinopoli. 2010. False data injection attacks in control systems. In *First workshop on Secure Control Systems*. 1–6.

[51] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. 2019. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems* 30, 9 (2019), 2805–2824.

[52] G. Zizzo, C. Hankin, S. Maffeis, and K. Jones. 2019. Adversarial Machine Learning Beyond the Image Domain. In *56th Annual Design Automation Conference*. 1–4.

6 APPENDIX

6.1 Proof of Theorem 1

First, we will show that applying such attack sequence results in an unbounded estimation error. For LTI systems, the dynamic of the state estimation error follows

$$\begin{aligned}\Delta x_t &= x_t - \hat{x}_t \\ &= A\Delta x_{t-1} + w_t - L(y_t^c - CA\hat{x}_{t-1} - CBu_{t-1}) \\ &= A\Delta x_{t-1} + w_t - L\phi_t\end{aligned}$$

As the matrix A is unstable, it follows that $\|\Delta x_t\|$ will be unbounded as $t \to \infty$.

We now show that the attack is stealthy from the perspective of the IDS. In this case, the residue signal z_t satisfies

$$\begin{aligned}z_t &= y_t^c - C(A\hat{x}_{t-1} + Bu_{t-1}) = \\ &= y_t + a_t - C(A\hat{x}_{t-1} + Bu_{t-1}) = \phi_t.\end{aligned} \tag{12}$$

Therefore, it follows that

$$\begin{aligned}\mathbb{E}\{g_t^a\} &= \mathbb{E}\{z_t^T S^{-1} z_t\} = \mathbb{E}\{\phi_t^T S^{-1}\phi_t\} \\ &= trace(\mathbb{E}\{\phi_t^T S^{-1}\phi_t\}) = \mathbb{E}\{trace(\phi_t\phi_t^T S^{-1})\} \\ &= trace(\mathbb{E}\{\phi_t\phi_t^T\}S^{-1}) \leq trace(SS^{-1}) = p,\end{aligned}$$

where we used the linearity of expectation and trace operation. Note that for LTI systems, the expectation of g_t (also known as the degrees of freedom of the distribution) satisfies that $\mathbb{E}\{g_t\} = p$. Based on the properties of the χ^2 distribution, since $\mathbb{E}\{g_t^a\} \leq \mathbb{E}\{g_t\} = p$, it follows that $\mathbb{P}(g_t^a > \eta) \leq \mathbb{P}(g_t > \eta) = \epsilon$, and thus the attack sequence is stealthy.

6.2 Proof of Theorem 2

From the multivariate Chebyshev's inequality [37], it holds that $\mathbb{P}(v_t^T R^{-1} v_t \leq k^2) \geq 1 - \frac{p}{k^2}$. On the other hand, using our assumption $R \leq \sigma I$, it holds that $\sigma^{-1} v_t^T v_t \leq v_t^T R^{-1} v_t$ for any $v_t \in \mathbb{R}^p$. Therefore, $\mathbb{P}(v_t^T v_t \leq \sigma k^2) \geq 1 - \frac{p}{k^2}$ or equivalently $\mathbb{P}(\|v_t\| \leq \sqrt{\sigma}k) \geq 1 - \frac{p}{k^2}$. Now, with the probability of at least $1 - \frac{p}{k^2}$, we have that

$$\begin{aligned}\alpha &\leq \|y_t - h(\hat{x}_t)\| = \|h(x_t) + v_t - h(\hat{x}_t)\| \leq \\ &\leq L\|x_t - \hat{x}_t\| + \|v_t\| \leq L\|x_t - \hat{x}_t\| + \sqrt{\sigma}k,\end{aligned}$$

which results in $\|x_t - \hat{x}_t\| \geq \frac{\alpha - \sqrt{\sigma}k}{L}$.

6.3 Details of Employed CARLA Setup

For planning, CARLA provides with a state-machine waypoint following algorithm. A vehicle's (estimated) pose and velocity were used along with map-based waypoints to coordinate (i) road-following, (ii) left-turn, (iii) right-turn, (iv) intersection, and (v) hazard-stop conditions [6]. We estimated the pose and velocity using an EKF with high-rate sensor data.

We also leverage the EKF structure to design an industry-standard χ^2 anomaly detector (AD). We set threshold η to result in $\epsilon = .05$ in normal condition. Then, the integrity value shown in the left bar of Figure 3(a),(b),(c) represents the number of measurements that pass the χ^2 AD requirement out of the last 20 measurements. We assume that the attack is detected if more than two sensor measurements cannot pass the requirement in this window of time. As sensor inputs, a Global Navigation Satellite Sensor (GNSS) sensor provides loosely coupled position solutions in global coordinates, a commercial GNSS standard. We similarly define a generalized velocimeter model, derived from Doppler (or, more frequently in safety-critical applications, GNSS delta-range).

State estimates and planning objectives were feed into a standard feedback controller [7] that targeted a cruising speed of 25 km/hr ($\sim 7m/s$). The control algorithm drove the following actuators with associated input ranges: (i) Steering wheel angle on $[-1.0, 1.0]$, (ii) Throttle on $[0.0, 1.0]$, and (iii) Brake on $[0.0, 1.0]$. Finally, we visualize the vehicle trajectory and system integrity with a heads-up-display presented in Figure 3; the videos for our CARLA experiments are available at [5].

6.4 Analysis of Attack Success Rate on Unmanned Aerial Vehicles

Tables 8 and 9 summarize the attack success rate (ASR) for both learned attack generators for UAVs; the generators were obtained for each training period T and different values of $|y|$ and $|z|$ (i.e., the drone's distance from the desired position along these two axis).

Table 8: Attack success rate (SR) for different values of T with $N = 300$, $\lambda = .5$ and different values of y along Y axis over 100 experiments for the UAV.

SR % FNN/RNN	$\|y\| = .2$	$\|y\| = .5$	$\|y\| = .8$	$\|y\| = 1$	$\|y\| = 2$	$\|y\| = 4$	$\|y\| = 6$	$\|y\| = 8$
$T = 100$	0/44	0/0	0/0	0/0	0/0	0/0	0/0	0/0
$T = 200$	92/91	24/20	0/0	0/0	0/0	0/0	0/0	0/0
$T = 400$	2/100	0/100	0/96	0/93	0/25	0/0	0/0	0/0
$T = 600$	99/100	94/97	92/95	91/94	73/66	0/0	0/0	0/0
$T = 800$	100/91	98/91	96/91	95/90	88/81	73/65	62/51	45/41

Table 9: Attack success rate (SR) for different values of T with $N = 300$, $\lambda = .5$ and different values of z along Z axis over 100 experiments for the UAV.

SR % FNN/RNN	$\|z\| = .2$	$\|z\| = .4$	$\|z\| = .6$	$\|z\| = .8$	$\|y\| = 1$	$\|z\| = 2$	$\|z\| = 3$	$\|z\| = 4$
$T = 100$	0/9	0/0	0/0	0/0	0/0	0/0	0/0	0/0
$T = 200$	67/0	24/0	4/0	0/0	0/0	0/0	0/0	0/0
$T = 400$	44/42	0/17	0/10	0/7	0/4	0/2	0/0	0/0
$T = 600$	86/81	59/74	45/60	26/48	11/36	3/28	0/3	0/0
$T = 800$	95/88	88/85	78/83	69/80	64/75	40/66	0/41	0/6