Query-Based Black-Box Stealthy Sensor Attacks on Cyber-Physical Systems

Shixiong Jiang*
University of Notre Dame
sjiang5@nd.edu

Weizhe Xu*
University of Notre Dame
wxu3@nd.edu

Mengyu Liu
University of Notre Dame
mliu9@nd.edu

Fanxin Kong
University of Notre Dame
fkong@nd.edu

***Abstract*—We study the vulnerability of Cyber-physical systems (CPS) under stealthy sensor attacks in black-box scenarios. "Black-box" refers to scenarios where the attacker has minimal knowledge of the target system. Designing a stealthy sensor attack sequence under this scenario has two main challenges. The first one lies in ensuring the stealthiness of the sensor attack, meaning does not trigger an alert when applying the generated sensor attack sequence to the CPS. The second one is maintaining stealthiness throughout the attack generation process, indicating the limitation on the alarm frequency when generating the attack sequence. To address the above challenges, we develop a query-based black-box stealthy attack framework to violate the safety of the CPS. To maintain stealthiness during training, an active learning method has been introduced to extract the detector's information to a time series model. The stealthy attack sequence is then generated from that model. Experiments on four numerical simulations and a high-fidelity simulator demonstrate the effectiveness of the proposed framework.**

***Index Terms*—cyber-physical systems, vulnerability, stealthy, sensor attack, query, safety**

I. Introduction

Cyber-physical systems (CPS) integrate computing, networking, and physical systems via sensors and actuators. Widely used in applications like autonomous vehicles, drones, and power grids, CPS face significant security and safety challenges due to their complex environments, high connectivity, and long working periods. Sensor attacks are particularly severe, allowing attackers to manipulate sensor data and mislead controllers into issuing dangerous commands, such as manipulating the LiDAR sensor to steer a vehicle into obstacles [1] or crushing a drone by corrupting the values of an altitude sensor [2]. These emerging threats have spurred innovative research into sensor attack detectors [3]–[5]. The detectors trigger an alert by analyzing the difference between the predicted system states and system states from sensor readings.

Despite the presence of these detectors, some well-designed stealthy sensor attack methods can still achieve their objectives without triggering alarms. A stealthy sensor attack refers to the successful steering of a system into an unsafe zone while simultaneously avoiding the activation of detector alarms. These methods [6]–[8] often require significant knowledge of the system, such as system dynamics. However, having a thorough knowledge of the system is often unrealistic in practical applications. For example, attackers may find it hard to know the system model of a specific chemical plant. To fill this gap, we introduce the query-based method outlined in this paper for generating effective stealthy sensor attack sequences aimed at black-box CPS. In this context, "black-box" refers to scenarios where the attacker has minimal knowledge of the target system.

Designing a successful black-box stealthy sensor attack is promising yet challenging. The first challenge lies in ensuring the stealthiness of the sensor attack. Maintaining stealthiness of attack resides in devising an entire attack sequence that avoids triggering any alerts. Due to security and privacy considerations, CPS applications often obscure domain knowledge, including system dynamics and detector design, from external entities. Consequently, existing stealthy attack approaches that assume access to such knowledge are not applicable in this black-box scenario.

The second challenge is maintaining stealthiness throughout the attack generation process. This challenge implies that the process of obtaining the attack generator should not excessively trigger alarms, as system administrators may notice these additional alarms and modify the system or detector parameters, rendering the attack ineffective or exposing the attacker. Query is a widely used technique in black-box vulnerability analysis and attack synthesis. However, generating the attack through queries will inevitably trigger alarms. Reducing the number of alarms during the query process to ensure stealthiness during training poses a significant challenge.

To address these challenges, we propose a query-based black-box stealthy sensor attack framework to generate an efficient stealthy attack sequence while adhering to these constraints. We employ an active learning approach to generate attack sequences offline and utilize online queries to collect data. This enables the training of a neural network to replicate the system's detector, ultimately producing effective and stealthy attack sequences.

Our contributions are as follows. 1) We develop a framework to train a time series model that mimics a black-box system's detector behavior, capable of creating stealthy sensor attacks that significantly alter the system state. 2) Our method, based on active learning, efficiently gathers training data without triggering frequent alerts, maintaining training stealth. 3) We implement and evaluate our framework on various numerical and high-fidelity simulators, demonstrating its efficiency and effectiveness compared to existing methods.

II. Background and Related Works

In this section, we introduce some existing works on stealthy sensor attacks. The first group is conventional model-based approaches. The replay attack [9], a basic stealthy sensor attack, replaces real-time sensor data with previously recorded data. It simulates disturbance factors such as environmental noise, achieving a stealthy effect. However, it fails to effectively guide the system into the unsafe region. Covert attack [6], zero dynamic attack [7], and false data injection attacks [8] are all capable of performing stealthy sensor attacks. These methods typically generate a stealthy attack sequence offline, relying on complete knowledge of the system dynamics. However, such approaches necessitate an in-depth understanding of the system model, which significantly limits their applicability in real-world scenarios.

The second group uses data-driven methods to attack CPS. Studies like [10] and [11] target linear time-invariant (LTI) systems by learning the open-loop gain from control inputs, systems states, and

* denotes equal contribution.

some other information to generate attacks. Khazraei et al. [12] go further by designing a neural network-based attack generator to produce effective and covert sensor attacks. This is achieved by incorporating the detector's residuals and system deviation values as loss functions.

Although these learning-based methods can generate effective stealthy sensor attacks, they are ineffective in black-box scenarios, as they require access to internal system information such as control inputs and detector residuals. In contrast, our approach is designed for black-box settings, relying solely on sensor measurements and detector alarm status. These can be obtained through external observation: for sensor measurements, attackers can leverage their own additional sensors; for detector alarm status, the system's abnormal responses, such as recovery or reboot, often indicate an alarm has been triggered.

III. Preliminaries

A. System Model

The CPS model considered in the paper is a physical process governed by a discrete-time feedback controller. The controller operates at every time step. At the beginning of the time step t, the controller reads the state estimation $\hat{x}_{t|t-1}$ from the state estimator and uses a control algorithm to calculate a control input u_t. The control input u_t is then sent to the plant and drives the plant's evolution. Then the state estimator reads the sensor measurement y_{t+1} and gives the new state estimation $\hat{x}_{t+1|t}$. The system can be modeled as below:

$$\begin{aligned} x_{t+1} &= f(x_t, u_t) + w_t \\ u_t &= \Omega(\hat{x}_{t|t-1}),\ y_t = h(x_t) + v_t \end{aligned}$$

where $x_i \in \mathbb{R}^n$, $u_t \in \mathbb{R}^m$ and $y_t \in \mathbb{R}^p$ are the real system state, control input, and sensor measurements. n, m and p denote the corresponding dimensions. $f : \mathbb{R}^m \to \mathbb{R}^n$ is the system dynamics, $\Omega : \mathbb{R}^n \to \mathbb{R}^m$ is the control algorithm, and $h : \mathbb{R}^n \to \mathbb{R}^p$ is the observation function.

B. Threat Model

We investigate the scenario of a black-box sensor attack, where an attacker can only access the sensor measurement and the alarm notifications. This assumption is valid, as attackers can measure accurate sensor values externally. Additionally, many systems visibly respond to detected attacks, such as by initiating recovery [13] or restarting [14].

In real-world scenarios, attacking multiple sensors simultaneously increases complexity and detection risk, especially in the black-box context of this paper. Thus, our approach assumes the attacker targets only one sensor at a time. Shown in the equation: $supp(a_t) = \{i \,|a_{ti} \neq 0\} \subseteq \{0, 1, ..., p\}$, where the support of attack vector $a_t \in \mathbb{R}^p$ at time step t is equal to 1, meaning only one sensor is under attack.

Under these sensor attacks, the state estimator will receive y_t^c as the sensor reading instead, as shown below:

$$y_t^c = y_t + a_t = h(x_t) + v_t + a_t$$

where a_t indicates the sensor attack at time step t. As a result, the estimated system state $\hat{x}_{t|t-1}$ may significantly deviate from the actual state x_t, leading to $\hat{x}_{t|t-1} \neq x_t$. Such discrepancies can prevent the controller from generating accurate control inputs, expressed as $\Omega(\hat{x}_{t|t-1}) \neq \Omega(x_t)$. Consequently, this may lead to the system state deviating to the unsafe area.

C. Ordinary Detector

We assume that the system is equipped with a residual-based detector, such as window-based [3], cumulative sum (CUSUM) [5], and χ^2 [4], which are three most commonly used detectors.

Regardless of the type of state estimator used in the system, the residual r_t, as shown in Equation (1), is calculated by subtracting the real sensor readings from the inferred sensor readings calculated by the state estimator.

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

1) Window-based detector: It is a short-memory detector that focuses on accumulating residuals within an observation window time. If the accumulated residuals r_t^{sum} exceed a predefined threshold τ, meaning $r_t^{sum} > \tau$, an alarm is triggered.

2) Cumulative sum (CUSUM): This approach enhances long-term memory for a window-based detector. It is predicated on the observation that the residual value, typically a small quantity associated with noise, remains low in the absence of an attack on the system. Unlike window-based detector, CUSUM lacks a fixed observation window w and updates s_t using the equation: $s_{t+1} = (s_t + |r_t| - b)^+$, where r_t is the current residual, drift b counters noise to prevent s_t from growing infinitely in the absence of attacks. An alert is triggered when $s_t > \tau$.

3) χ^2: This detector is also a long memory detector. χ^2 is based on the distribution of cardinality. It is depicted in equation: $g_t = r_t^T S_t^{-1} r_t$, where S_t is the covariance matrix of r, thus g_t is the weighted norm of r_t, with χ^2 distribution. The detector will raise an alert if g_t exceeds the threshold τ, indicating that the residuals' distribution has altered, possibly as a result of attacks.

D. Problem Statement

The attacker aims to perform effective and stealthy sensor attacks in black-box scenarios. Specifically, the goal is to generate an attack sequence targeting one sensor over a duration T, causing significant system deviation from reference values while avoiding detection and alarm triggers.

1) Effectiveness: The attacker seeks to generate an attack sequence that maximizes the impact on the target sensor, measured by state estimation errors, defined as:

$$\Delta x_t = x_t - \hat{x}_{t|t-1} \tag{2}$$

Before the sensor attack starts, the value of Δx_t is constrained by noise, and the real system state x_t is close to the reference value. Sensor attacks corrupt the data received by the estimator, causing significant deviations from the system's actual state and resulting in estimation errors. Furthermore, this leads the system to deviate from the reference.

2) Stealthiness: Regarding stealthiness, we require not only that the final attack sequence generated avoids triggering system alarms, but also that the process of creating the attack sequence through the algorithm minimizes excessive or frequent alarm triggers.

Specifically, the problem can be formulated as the following: Consider a CPS where the attacker has access only to sensor measurements and the detector's alarm status. The attacker generates an attack sequence targeting a single sensor. During the attack generation process, the attacker ensures that the detector alarm is not triggered frequently. Once the attack is initiated, the goal is to maximize state estimation errors while avoiding triggering the detector's alarm.

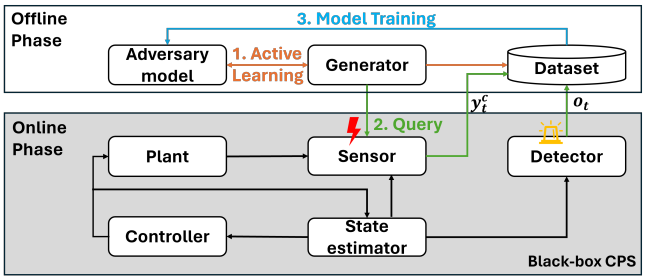


Fig. 1: The overview of the framework.

IV. Overview of the Framework

This section provides an overview of our query-based black-box stealth sensor attack framework, as shown in Fig. 1.

The grey box at the bottom represents the black-box CPS, whose internal information is inaccessible to us. We can only obtain sensor data y_t^c and the detector's result o_t. The white box at the top illustrates our query-based black-box stealth sensor attack framework. The adversary model is a time series model. It takes the sensor measurement y_t^c as the input and output alert or not o_t, allowing it to simulate the behavior of the detector. The generator is responsible for creating attack sequences based on our specifically designed active learning algorithm. The dataset is tasked with collecting data that is used to train the adversary model.

The orange, green, and blue arrows represent the three phases: active learning, querying, and model training. These phases are executed sequentially in a loop until a stealthy attack sequence meeting the requirements is generated. Active learning and model training occur offline, while the querying phase involves interaction with the CPS and operates online.

The process begins with the active learning phase, where the generator creates an initial attack sequence $a = \{a_0, a_1, ..., a_T\}$ of length T by interacting with the adversary model. In the query phase, this sequence is applied to the target system, producing sensor measurements y^c and detector alarm information o, which are collected into the dataset. A "one-step query" stops when the detector triggers an alarm, discontinuing any further attacks in the sequence. In the model training phase, the updated dataset, containing each attack sequence and corresponding alarm responses, is used to train the time-series model. The process then cycles back to active learning to generate a new attack sequence. This loop continues until a sequence of length T is produced that avoids triggering any alarms, resulting in the final stealthy attack sequence.

V. Query-based black-box stealthy sensor attack

This section outlines the process for generating a stealthy attack sequence targeting a black-box CPS. The first step is selecting a victim sensor, as the attacker in our threat model targets only one sensor at a time. The sensor with the highest potential to deviate the system from its reference should be chosen. A simple method is selecting the sensor most closely tied to the attack objective, such as targeting the distance sensor in a vehicular platooning system to cause collisions. Alternatively, a more refined approach involves testing the query-based attack method on each sensor and selecting the most effective one. For simplicity, we assume the appropriate sensor is pre-selected as the target.

A. Query

In this phase, a sensor attack sequence is applied to a stable system. The detector then identifies whether the sensors are maliciously perpetuated based on the ordinary detectors. The attacker collects the post-attack sensor measurements $y_t^c = y_i + a_i$ and alarm information o and compiles them into a set of training data.

When an alarm is triggered, the attacker halts the current attack and ends the one-step query. The query process concludes either when the detector raises an alarm or when the attack sequence reaches its end at T. For instance, if the query stops at step k, the training data includes $y^c = \{y_0^c, y_1^c, ..., y_k^c\}$, $y \in \mathbb{R}^{k*p}$ and the labels are $o = \{0, 0, ..., 1\}$, $o \in \mathbb{R}^k$. The attack sequence is then refined using an active learning algorithm, which is detailed later, and the refined sequence becomes the input for the next one-step query.

To maintain system stability, the next query is only initiated after the system returns to a stable state, allowing the attacker to control the interval between alarms and reduce detection risks. This flexibility improves stealth by spacing alarms and is especially useful during early training when initial attack sequences are more likely to trigger alerts.

B. Model Training

In this phase, we train the time series model to mimic the detector using the updated dataset.

Existing studies [10], [12] demonstrate that attackers can design stealthy attack sequences when they have access to the system model, detector parameters, and control signals. To reduce alarm triggers and minimize queries, having an approximate replica of the detector is crucial, as it allows for evaluating the likelihood of an alarm before launching a query.

The detector evaluates sensor readings across multiple time steps, forming correlated time series data, to decide whether to trigger an alarm. To replicate this behavior, we use a time series model, specifically a Long Short-Term Memory (LSTM) network in this paper. The LSTM model takes the attacked sensor readings y_t^c at time t as input and outputs the likelihood of the detector raising the alarm, shown in the equation below:

$$h_t = W_\theta(h_{t-1}, y_t^c),\ \hat{o}_t = \sigma(h_t)$$

where h_t denotes the hidden state calculated in the LSTM model at time t. The function σ is a mapping from h_t to $\hat{o}_t$, which is the alarm signal predicted by the current model.

Using the updated training dataset, the LSTM model learns the detector's behavior under various outputs and attack sequences. Since this is a binary classification task, the binary cross-entropy loss (BCEloss) [15] is used to calculate the loss.

At each training step, the LSTM model is trained on the previously collected dataset until it converges, ensuring $\hat{o}_t > 0.5$ if $o_t = 1$ and vice versa. This alignment ensures the model accurately reflects the detector's behavior in the real system. Consequently, the attack sequences generated using the LSTM model remain stealthy while achieving the maximum possible attack amplitude. Moreover, this approach enhances data utilization, which is especially valuable given the limited data available in this scenario.

C. Active Learning

In this phase, the generator generates the new attack sequence based on the LSTM model.

Attacking a system repeatedly to gather large amounts of data for training can significantly improve the LSTM model's performance. However, this is impractical under constraints on the number of alarms during training. To address this, we propose a generator based on an active learning algorithm to create new attack sequences using the trained LSTM model. By analyzing confidence scores, we identify

regions where the LSTM model is uncertain, guiding us to explore these areas to better simulate the detector's behavior. The details of

Algorithm 1 Active Learning

Input: y, a, β, k, a_{min}, a_{max}
Output: a_k
1: **while** $\beta > 0$ **do**
2: **for** $t = 0 : k$ **do**
3: $h_t = W_\theta(h_{t-1}, y_t^c)$
4: $\hat{o}_k = \sigma(h_k)$
5: **if** $ground(\hat{o}_k) == 0$ **then**
6: $a_{min} \leftarrow a_k$
7: **else**
8: $a_{max} \leftarrow a_k$
9: $a_k \leftarrow (a_{min} + a_{max})/2$
10: $\beta \leftarrow \beta - 1$

the active learning are summarized in Algorithm 1, which takes as inputs the time step k where the detector triggers an alert, sensor measurements, the partial attack sequence from step 0 to k, and the attacker-defined parameter a_{min}, a_{max}. The algorithm operates as follows: 1) Lines 2-4 are the LSTM inference process to get the output $\hat{o}_k$, the likelihood of an alert at step k. 2) Lines 5-9 update the a_k based on the output similar to binary search. If $\hat{o}_k < 0.5$, we change the a_{min} to a_k; If $\hat{o}_k > 0.5$, we set a_{max} to a_k. Then the new a_k is calculated from $(a_{min} + a_{max})/2$. 3) After β iterations, the updated a_k converges to a value near the boundary of the LSTM model's classification. This boundary value is used for subsequent queries as it represents data points where the model is most uncertain, making it highly valuable for refining the model.

The refined attack sequence a_t is generated based on the current LSTM model, minimizing the risk of triggering the detector during real-system queries. The active learning method further enables exploration near the LSTM model's decision boundary. Since the LSTM model is trained to approximate the system detector, the attack sequence performs similarly on both, ensuring stealthiness by avoiding alerts on the LSTM model. Additionally, the attack sequence is crafted using values near the detector's threshold at each time step, potentially maximizing Δx_t under the trained LSTM model, as per Equation (2). This aligns with our objective of pushing the system to deviate significantly without triggering alarms.

Although this method does not guarantee optimal attack performance, it achieves strong results in black-box scenarios, as the well-trained LSTM model closely mimics the real system detector. The active learning approach reduces the dataset size needed for LSTM model training, thereby minimizing alarm occurrences during data collection, and enhancing both the efficiency and stealth of the generator.

To summarise, the whole method is illustrated as algorithm 2. We first initialize a sequence of relatively large a and the LSTM model parameter θ. The T is the total length of the attack sequence which is defined by the attacker. The method works as follows: 1) Line 3-4: The attacker queries the system with the current attack sequence a; 2) Lines 5-8: Collect the data from time step 0 to k if the detector alerts at time step k; 3) Lines 9-10: Train the LSTM model until convergence and update the a_k using Algorithm 1; 4) Lines 11-13: Reset k to 0 if alarms, continue if not; 5) repeat the process after the attacker obtains a length of T attack sequence. By updating the attack duration steps, our LSTM model can effectively simulate the behavior of the detector in the actual system. The attack sequence

Algorithm 2 Query-based black-box stealthy attack algorithm

1: $Initialize\ \theta, T, a, \beta, a_{min}, a_{max}$
2: **for** $t = 0 : T$ **do**
3: **while** $k \leq t$ **do**
4: $a \rightarrow System$
5: **if** $alarm$ **then**
6: $y = \{y_0, y_1, ..., y_k\} \in \mathbb{R}^{p*k}$
7: $y^c = \{y_0 + a_0, y_1 + a_1, ..., y_k + a_k\} \in \mathbb{R}^{p*k}$
8: $o = \{o_0, o_1, ..., o_k\} \in \mathbb{R}^k$
9: $\theta \leftarrow LSTM\ update(\theta, y^c, o)$
10: $a_k \leftarrow Active\ Learn(y, a, \beta, k, a_{min}, a_{max})$
11: $k \leftarrow 0$
12: **else**
13: $k \leftarrow k + 1$

updated through this process can be utilized as a stealthy attack, causing the system to deviate from the reference without triggering an alert.

VI. Evaluation

We now compare the performance of our proposed stealthy attack strategy with two baselines. We test them on four numerical simulators and a high-fidelity simulator.

A. Numerical Simulation

To evaluate our algorithm and compare it with the baseline method, we consider several numerical CPS simulations: vehicle platoon, quadruple tank, DC motor speed, and aircraft pitch. We set $\beta = 25$ in all four benchmarks.

1) Simulator setting: **Vehicle platoon.** This system consists of four vehicles maintaining a stable 1-meter gap, forming a platoon, modeled after [16]. Each vehicle uses a lidar to measure the gap from the vehicle ahead. It also monitors the velocities of all vehicles. An attacker aims to reduce the gap between the front two vehicles and can only manipulate the distance sensor readings between them.

Quadrotor. The system model for controlling a four-rotor quadrotor is described in [17]. The input of the system is the total vertical thrust of the quadrotor, along with the three angular motions, The output of the system consists of six variables: The first three represent the relative position, while the last three correspond to the angles of pitch, yaw, and roll, respectively. In our scenario, we assume that the quadrotor is maintaining its altitude at 5 meters. An attacker manipulates the sensor measuring the altitude, causing it to drop.

DC Motor speed. The DC motor is an actuator to provide rotary motion and is usually coupled with wheels, drums and cables. The input of the system is the voltage source applied to the motor's armature, while the output is the rotational speed of the shaft. An attacker can manipulate the rotational speed sensor and the goal is to decrease the speed of the shaft.

Aircraft pitch. The aircraft pitch system [18] is an autopilot controlling the pitch angle of an aircraft. The system takes the elevator deflection angle as input and outputs the pitch angle. Assuming the yaw angle remains at 0 radians, the attacker aims to disrupt this stability by manipulating the pitch angle to deviate from its original value of zero.

2) Detector setting: For each benchmark, we employ both the CUSUM and χ^2 detectors. Process noise is modeled as zero-mean white noise. We tune the detectors to maintain a false alarm rate below 0.01 in the absence of attacks.

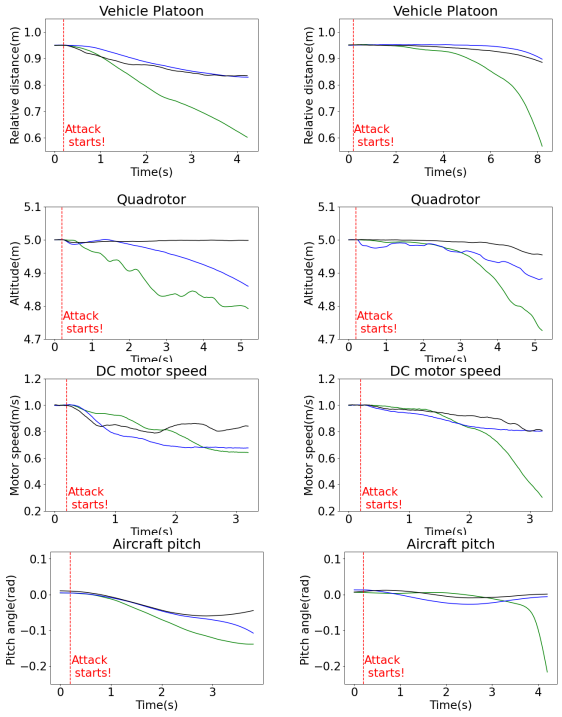


Fig. 2: Attack on each benchmark with CUSUM (left column) and χ^2 (right column) detector. The x-axis represents the time steps, while the y-axis represents the target system states. Green, blue, and black denote our query-based methods, the learning method, and random search methods, respectively.

3) Baseline setting: **Learning method.** We use the learning-based method [12] as a baseline. This method generates attack sequences using sensor measurements y_t and the detection function g_t during training. It also relies on N offline copies of the system, which is impractical in real-world scenarios. To adapt the baseline to our case, we set $N = 1$. **Random search method.** We introduce a second baseline method that uses a modified random search strategy to select the next attack sequence, as a purely random search is too inefficient. This method begins with the same initial value as our query-based approach and updates the attack sequence if it triggers an alarm, using the equation $a_t = random(a_t * 0.8, a_t)$, where the random function selects a value between $a_t * 0.8$ and a_t.

4) Attack effect analysis: We evaluated our query-based attack method against the two baseline methods in terms of attack effectiveness without imposing time constraints on the attacker. As shown in Fig. 2, our method drives all four benchmark systems away from their stable states more quickly and significantly than the baselines, demonstrating superior effectiveness. The results also reveal that with χ^2 detectors, the learning-based method often fails to consistently destabilize the system. Meanwhile, the random search method can generate attack sequences but performs the worst, as it lacks the capability to maximize the attack's impact. In contrast, our query-based black-box stealthy attack method significantly enhances the attack's impact compared to baselines.

5) Stealthiness analysis: In the experiments, all methods generate stealthy final attack sequences, making a comparison of final stealthiness irrelevant. Instead, we analyze stealthiness during the training process, focusing on two key aspects: 1) the number of queries and 2) the number of alarms triggered during training. Notably, the learning-based method leverages knowledge of the detector's residual, enabling it to avoid many alarm triggers during training. As a result, this method is excluded from the comparison in this subsection.

For the first aspect, we analyze how the number of queries varies with the length of the attack sequence. As shown in Table I, our method substantially reduces the number of queries compared to the random search method. This reduction highlights the effectiveness of the trained LSTM model in capturing the dynamic changes in the system's sensor values, enabling fewer queries and faster generation of the attack sequence within a shorter training period.

No. of queries: Query / Search	Detector	T = 50	T = 100	T = 150
Vehicle platoon	CUSUM	70 / 291	102 / 514	154 / 916
	Chi-square	111 / 533	190 / 988	293 / 1424
Quadrotor	CUSUM	95 / 373	298 / 671	526 / 998
	Chi-square	172 / 486	323 / 858	410 / 1257
DC motor speed	CUSUM	61 / 251	113 / 488	165 / 716
	Chi-square	103 / 356	262 / 581	489 / 752
Aircraft pitch	CUSUM	122 / 255	235 / 542	368 / 780
	Chi-square	116 / 405	216 / 865	269 / 1286

TABLE I: The number of queries to generate a certain length of attack sequence under different benchmarks and detectors using our query-based method and random search method.

For the second aspect, we ensure the system manager remains unaware of the attack by pausing the attack and loop processes when an alarm triggers during attack generation. The overall alarm rates for our method are shown in Table II. As can be observed, the attack remains highly stealthy during the training process.

B. CARLA

We implement the lane-keeping autonomous driving system [1] and on the high-fidelity simulator CARLA [19]. An attacker can manipulate GPS sensor readings, deviating the vehicle from its path and potentially crashing into other vehicles or obstacles. System variable e_d is used to measure the distance by which the vehicle deviates from the reference.

The results are shown in Fig. 3. The attacker generates an attack sequence for a duration of T=300 control steps. Prior to the start of the attack, the vehicle remains centered within the lane, with $e_d = 0$, and the residual is minimal. Once the attack commences, the vehicle progressively veers away from the lane without triggering any alarms. Ultimately, the vehicle collides with an off-road firetruck, but even then, the residual value fails to activate an alarm. This stealthy behavior indicates the success of the attack. Fig. 4 illustrates the process by which the attack impacts the vehicle. Note that the residual changes once the attack starts, this is inevitable because the χ^2 detector is sensitive to the noise changes. However, our attack does not trigger an alarm during the implementation phase.

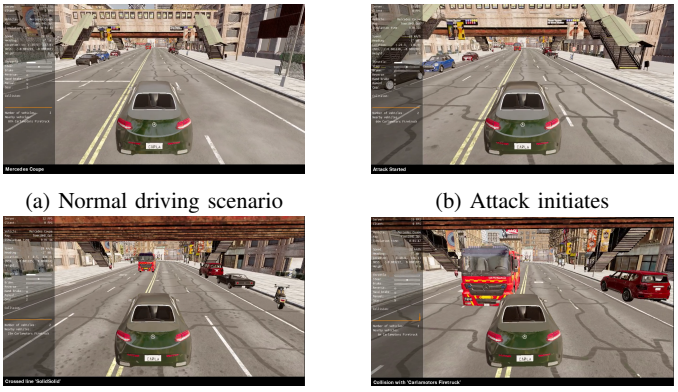
C. Discussion

One possible solution to defend against our proposed attacks is to block malicious users from gaining information about the detector by the query. As we mentioned before, the query process will inevitably generate alarms by the detector to get meaningful information. Defenders can utilize more sophisticated detectors, such as adaptive detector [20] and stealth attack detector [21].

The other direction to defend is protecting the system from being steered to non-secure areas under our proposed sensor attack

	Vehicle platoon	Quadrotor	DC motor speed	Aircraft pitch
Alarm rate	0.0036	0.0034	0.0035	0.0036

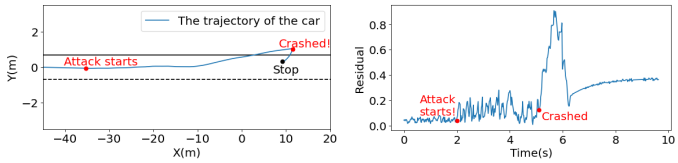
TABLE II: The alarm rate for each benchmark.



(a) Normal driving scenario (b) Attack initiates

(c) Vehicle deviates (d) Collision with firetruck

Fig. 3: The query-based attack on an autonomous vehicle.



(a) Normal state (b) Attack starts

Fig. 4: Illustrations of system states under normal and attack conditions in CARLA.

sequence. The simplest way to block sensor attacks is to leverage sensor redundancy. For example, in aircraft, there is usually more than one pitot tube to obtain airspeed. Some other methods, such as sensor fusion [22], well-designed state estimation [23], [24] can also be helpful.

VII. Conclusion

We propose a query-based black-box sensor attack method for CPS with a residual-based detector. Our approach ensures stealthiness both during the training process and throughout the attack execution. Extensive experiments have demonstrated the effectiveness of the proposed method.

Acknowledgment

This work was supported in part by NSF CNS-2333980. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the National Science Foundation (NSF).

References

[1] J. M. Snider *et al.*, "Automatic steering methods for autonomous automobile path tracking," *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08*, 2009.

[2] 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.

[3] I. Jovanov and M. Pajic, "Relaxing integrity requirements for attack-resilient cyber-physical systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 12, pp. 4843–4858, 2019.

[4] R. Tunga, C. Murguia, and J. Ruths, "Tuning windowed chi-squared detectors for sensor attacks," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 1752–1757.

[5] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954.

[6] R. S. Smith, "Covert misappropriation of networked control systems: Presenting a feedback structure," *IEEE Control Systems Magazine*, vol. 35, no. 1, pp. 82–92, 2015.

[7] A. Teixeira, I. Shames, H. Sandberg, and K. H. Johansson, "Revealing stealthy attacks in control systems," in *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2012, pp. 1806–1813.

[8] Y. Mo and B. Sinopoli, "False data injection attacks in control systems," in *Preprints of the 1st workshop on Secure Control Systems*, vol. 1, 2010.

[9] ——, "Secure control against replay attacks," in *2009 47th annual Allerton conference on communication, control, and computing (Allerton)*. IEEE, 2009, pp. 911–918.

[10] M. J. Khojasteh, A. Khina, M. Franceschetti, and T. Javidi, "Learning-based attacks in cyber-physical systems," *IEEE Transactions on Control of Network Systems*, vol. 8, no. 1, pp. 437–449, 2021.

[11] A. Rangi, M. J. Khojasteh, and M. Franceschetti, "Learning based attacks in cyber physical systems: Exploration, detection, and control cost trade-offs," in *Proceedings of the 3rd Conference on Learning for Dynamics and Control*, ser. Proceedings of Machine Learning Research, A. Jadbabaie, J. Lygeros, G. J. Pappas, P. A.nbsp;Parrilo, B. Recht, C. J. Tomlin, and M. N. Zeilinger, Eds., vol. 144. PMLR, 07 – 08 June 2021, pp. 879–892. [Online]. Available: https://proceedings.mlr.press/v144/rangi21a.html

[12] A. Khazraei, S. Hallyburton, Q. Gao, Y. Wang, and M. Pajic, "Learning-based vulnerability analysis of cyber-physical systems," in *2022 ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS)*, 2022, pp. 259–269.

[13] L. Zhang, P. Lu, F. Kong, X. Chen, O. Sokolsky, and I. Lee, "Real-time attack-recovery for cyber-physical systems using linear-quadratic regulator," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 20, no. 5s, pp. 1–24, 2021.

[14] F. A. T. Abad, R. Mancuso, S. Bak, O. Dantsker, and M. Caccamo, "Reset-based recovery for real-time cyber-physical systems with temporal safety constraints," in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2016, pp. 1–8.

[15] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on information theory*, vol. 42, no. 2, pp. 429–445, 1996.

[16] S. Dadras, R. M. Gerdes, and R. Sharma, "Vehicular platooning in an adversarial environment," in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, ser. ASIA CCS '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 167–178. [Online]. Available: https://doi.org/10.1145/2714576.2714619

[17] F. Sabatino, "Quadrotor control: modeling, nonlinearcontrol design, and simulation," 2015.

[18] D. Izci, S. Ekinci, A. Demirören, and J. Hedley, "Hho algorithm based pid controller design for aircraft pitch angle control system," in *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, 2020, pp. 1–6.

[19] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[20] F. Akowuah and F. Kong, "Real-time adaptive sensor attack detection in autonomous cyber-physical systems," in *2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2021, pp. 237–250.

[21] M. Liu, L. Zhang, P. Lu, K. Sridhar, F. Kong, O. Sokolsky, and I. Lee, "Fail-safe: Securing cyber-physical systems against hidden sensor attacks," in *2022 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2022, pp. 240–252.

[22] J. Z. Sasiadek, "Sensor fusion," *Annual Reviews in Control*, vol. 26, no. 2, pp. 203–228, 2002.

[23] M. Pajic, J. Weimer, N. Bezzo, P. Tabuada, O. Sokolsky, I. Lee, and G. J. Pappas, "Robustness of attack-resilient state estimators," in *2014 ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2014, pp. 163–174.

[24] J. L. G. Bello and M. Kim, "A novel monocular disparity estimation network with domain transformation and ambiguity learning," in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 474–478.