Out of Control: Stealthy Attacks Against Robotic Vehicles Protected by Control-based Techniques

Pritam Dash
University of British Columbia
Vancouver, Canada
pdash@ece.ubc.ca

Mehdi Karimibiuki
University of British Columbia
Vancouver, Canada
mkarimib@ece.ubc.ca

Karthik Pattabiraman
University of British Columbia
Vancouver, Canada
karthikp@ece.ubc.ca

ABSTRACT

Robotic vehicles (RVs) are cyber-physical systems that operate in the physical world under the control of software functions. They are increasing in adoption in many industrial sectors. RVs rely on sensors and actuators for system operations and navigation. Control algorithm based estimation techniques have been used in RVs to minimize the effects of noisy sensors, prevent faulty actuator output, and recently, in detecting attacks against RVs. In this paper, we propose three kinds of attacks to evade the control-based detection techniques and cause RVs to malfunction. We also propose automated algorithms for performing the attacks without requiring the attacker to expend significant effort or know specific details of the RV, making the attacks applicable to a wide range of RVs. We demonstrate these attacks on ArduPilot simulators and two real RVs (a drone and a rover) in the presence of an Intrusion Detection System (IDS) using control estimation models to monitor the runtime behavior of the system. We find that the control models are incapable of detecting our stealthy attacks, and that the attacks can have significant adverse impact on the RV's mission (e.g., cause the RV to crash or deviate from its target significantly).

CCS CONCEPTS

• **Security and Privacy → Intrusion detection systems**; • **Computer systems organization → Sensors and actuators**.

KEYWORDS

Cyber Physical Systems (CPS), Invariant Analysis, Robotic Vehicle Security

ACM Reference Format:
Pritam Dash, Mehdi Karimibiuki, and Karthik Pattabiraman. 2018. Out of Control: Stealthy Attacks Against Robotic Vehicles Protected by Control-based Techniques. In *Annual Computer Security Applications Conference (ACSAC'19), December 09-13, 2019, San Juan, PR, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/1122445.1122456

ACSAC '19, Dec 09-13, 2019, San Jose, PR

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Robotic Vehicles (RVs) are cyber-physical systems that operate autonomously leveraging closed-loop feedback control mechanisms (*e.g.*, PID controller [19]). Two prominent examples of such systems are Unmanned Aerial Vehicles (UAVs also known as drones) and autonomous robotic cars (also known as rovers). Such vehicles are utilized in a variety of industrial sectors (*e.g.*, agriculture, surveillance, package delivery [6, 8, 49], warehouse management [52]) and even critical missions such as space exploration [38]. Unfortunately, such vehicles are not well protected, and are vulnerable to both physical and cyber attacks. A few examples of such attacks demonstrated in previous research are GPS spoofing [25, 56], gyroscope sensor tampering [51], attacks on vehicles' braking system [50].

Because RVs rely on control algorithms for path-following and trajectory planning [45], using control properties as invariants to monitor systems' behaviour has been proposed to detect attacks. Control Invariants (CI) [12] and Extended Kalman Filter (EKF) [9] are two techniques that derive invariants using apriori knowledge of the system. Using the CI and EKF methods, the next state and control output signal of the RV is estimated. The estimated values are used to monitor the RV's runtime behaviour and flag anomalous behaviour, thus detecting potential attacks.

In this paper, we propose a technique to perform targeted attacks against RVs protected with the CI and EKF techniques. Our technique can work for any RV with little to no human intervention. Our main insight is that by design, CI and EKF techniques have to accommodate some degree of deviation from the planned trajectory due to environmental factors such as friction or wind, and hence have a certain threshold for flagging deviations as attack. Further, sensor noise and actuator defects exacerbate the problem. We propose an automated process by which an attacker can learn the thresholds and the tolerances of each system for any arbitrary RV that uses Proportional Integral Derivative (PID) control, the most commonly used control technique [31], and consequently perform targeted attacks against the RV. By controlling the deviation introduced and the timing of the attacks, we show that the attacker can remain stealthy and not be detected by techniques such as CI and EKF. Furthermore, though the deviations may be small, the consequences of the attacks are severe as they can be performed over a prolonged period of time, and at a time and place of the attacker's choosing. This makes them particularly insidious when RVs are used in safety-critical scenarios.

We propose three types of attacks that are undetectable by current control-based techniques on RVs. i) *False data injection*: We

devise an automated approach through which the attacker can derive the control state estimation model of RVs and reverse engineer it to obtain the detection threshold and monitoring window used in the IDS. Exploiting the aforementioned threshold related imperfections, the attacker can launch sensor and actuator spoofing attacks such that the deviations in the control output is always maintained under the detection threshold, i.e., a false data injection attack [32]. By performing such a controlled false data injection over a period of time, the attacker will be able to significantly deviate the RV from its original mission path. ii) *Artificial delay*: We launch artificial delays into the system's control execution process which will affect the timing behaviour of crucial system functions. We show that the attacker can inject intermittent delays in the reception of the RVs gyroscopic sensor measurements, which will, in turn influence the estimation of RV's angular orientation while eluding detection. By launching stealthy, intermittent delays, the attacker can adversely influence the RV's performance and efficiency. iii) *Switch-mode attack*: Finally, we identified that the invariants derived by CI and EKF do not provide tight bounds when the RV switches modes, *e.g.*, when a drone switches from steady flight to landing. We exploit this weakness to launch another form of false data injection attack on actuator signals, which is triggered upon the RV switching modes.

Prior work has focused on exploiting the vulnerabilities in communication channels, and attacks on the RV's sensors through noise injection [12, 50, 51] in the absence of any protection. In contrast, we consider a scenario where the RV is protected by control invariants, and the EKF technique, which makes the attacker's job much more difficult. Further, unlike prior work, we make minimal assumptions on the RV itself, and instead completely automate the attack generation and learning process, without requiring any apriori knowledge of the system on the part of the attacker (other than that the RV is using a PID control system). *To the best of our knowledge, we are the first technique to automatically find attacks against the control state estimation model of RVs without being detected by existing techniques, or targeting a specific type of RV.*

We make the following contributions in this paper:

(1) Demonstrate three types of stealthy attacks namely: false data injection, artificial delay, and switch mode attacks against RVs in the presence of attack detection techniques such as CI and EKF. Through the attacks, one can significantly deviate the RVs from their missions without being detected.
(2) Propose automated algorithms for launching the above three attacks against any *arbitrary* RV without apriori knowledge of its internals. We derive the thresholds and states of the RVs, and the protection techniques by repeated observations, and learn the control models used for state estimation.
(3) Implement the attacks on two real RV platforms, namely a robotic rover, and a drone, both based on the Ardupilot stack. We also use simulation to demonstrate the attacks on a wider range of RVs and trajectories.
(4) We find that attackers can learn the thresholds and states of the RVs using a modest amount of effort (typically 5 to 7 missions). We further show that the stealthy attacks can have severe repercussions such as deviating a drone by more than 100 meters from its trajectory (for a mission distance of 5 Kilometers), and deteriorating the efficiency and performance of rovers by increasing their mission duration by more than 50%. If launched strategically at vulnerable states, the stealthy attacks can also cause a drone to crash while landing, or cause other undesirable effects.

2 BACKGROUND

In this section, we first discuss the architecture and control of RVs, followed by a description of its modes of operation. Then, we present the attack detection mechanisms namely Control Invariants [12] and Extended Kalman Filter [9] (EKF).

2.1 Robotic Vehicle Control

RVs use Proportional-Integral-Differential (PID) feedback control for their navigation. Typically, an RV system has a number of sensors (*e.g.*, barometer, gyroscope, accelerometer, and magnetometer) that capture the physical state of the vehicle in the environment. The information about the vehicle's current state (*e.g.*, angular and linear position) is used as feedback to estimate the actuator signals (*e.g.*, rotors speed, steering) for positioning the vehicle in the next state. In the case of drones or rovers, a PID controller is used for position (*e.g.*, altitude, latitude, longitude) control, and attitude (*e.g.*, yaw, roll, pitch) control. Figure 1 (based on ArduPilot [7]) shows an example illustrating the PID controller used to perform attitude control along each axis.

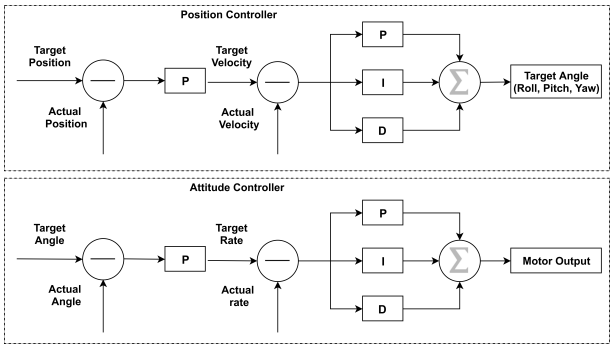


Figure 1: PID Control operations in RVs: Position Control and Attitude Control.

The position control is done using a *P* controller to convert the target position error (difference between the target position and actual position) into target velocity, followed by a PID controller to derive the target angle (roll, pitch, yaw). Similarly, the target angles are given as input to the attitude controller, and using the PID control functions, a high-level motor command is calculated. *P* is the proportional term, which aims to adjust the control signal (e.g., the rotor currents) proportional to the error; *I* is the integral term, which is for tracing the history of the error. It compensates for *P*'s inability to reduce the error in the previous iterations. *D* is the derivative term to avoid stark change in the error.

2.2 Modes of Operation in RV Mission

For a given flight path, an RV transitions through a series of high level states typically referred as modes of operation. In the case of

a drone for instance, when a mission starts, the drone is armed at its home location. When the *Takeoff* mode is triggered, the drone takes off vertically to attain a certain height. Subsequently, a series of modes can be performed such as *Loiter* mode, *Waypoint* mode, which will prompt the drone to fly autonomously to a pre-defined location, and Return to launch (*RTL*) mode, which will prompt the drone to return to home. When the drone arrives at the destination, its mode becomes *Land* mode, and finally the drone is disarmed. Figure 2 (based on ArduPilot SITL [7]) shows a state diagram of the various mode of operation commonly deployed in a drone. The change in mode of operation causes a change in the angular orientation, control input and actuator signals.

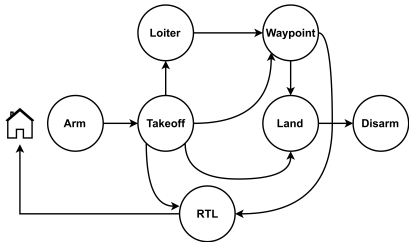


Figure 2: Modes of operation in RVs.

2.3 Control Invariants

The control invariant (CI) approach [12] models the physical dynamics of an RV and leverages its control laws to derive invariants (*e.g.*, control outputs). The control invariants are determined by two aspects namely, vehicle dynamics, and the underlying control algorithm. For a given RV, the CI model captures the system's sensor inputs, based on its current state to estimate the systems' control outputs. The approach then derives invariants using the following state estimation equations.

$$x'(t) = Ax + Bu \tag{1}$$

$$y(t) = Cx + Du \tag{2}$$

Where $x(t)$ is the current state, and $u(t)$ is the control input. A, B, C, D are state space matrices determined through system identification [34]. The above equations determine the next state $x'(t)$ and output $y(t)$ of the system based on the current state and control input signal. The CI model uses a stateful error analysis, where it accumulates the error (deviation) between the estimated output and the actual output in a pre-defined monitoring window. When the accumulated error exceeds a pre-defined threshold, the CI technique raises an alert *e.g.*, if the error for roll angle ($error = |y(t)_{est} - y(t)_{act}|$) is larger than 91 degrees (threshold) for a window of 2.6 seconds.

2.4 Extended Kalman Filter

Extended Kalman Filter [9] is commonly used in RVs to fuse multiple sensor measurements together to provide an optimal estimate of the position and/or orientation. For example, EKF fuses accelerometer, gyroscope, GPS and magnetometer measurements with a velocity estimate to estimate the UAV's yaw, pitch and roll. Assuming that the system is operating in a steady state, the estimate of the system's state is given by the following equation:

$$x'(t) = Ax + Bu + K(y(t) - C(Ax + Bu)) \tag{3}$$

Where K is the steady-state Kalman gain, and A, B, C are the state space matrices. An IDS based on EKF uses the residual analysis technique to detect sensors and actuator attacks. The difference between the real-time sensor measurement and the estimate of the sensor measurement is the residual vector, which is defined as:

$$r(t) = y(t) - C(Ax + Bu) \tag{4}$$

Where $r(t)$ is residual at each time-instant t. An IDS based on EKF compares if the residual $r(t)$ is larger than a pre-defined threshold for a certain monitoring window, and raises an alarm when such anomalous behaviour is observed [33].

3 LIMITATIONS IN EXISTING METHODS AND ATTACK SCENARIOS

This section describes the limitations in CI and EKF models, and how we exploit those limitations to design stealthy attacks. Then, we discuss a few attack scenarios to analyze the repercussions of such attacks when targeted at RVs deployed in industrial use-cases. Finally, we describe the main challenge we address.

3.1 Limitations in Existing Methods

As mentioned, the CI and EKF models derive invariants leveraging the control laws, and estimate the vehicles' position and angular orientation. An IDS based on CI and EKF models will analyze the error (*i.e.*, deviation) between the real-time values and the estimated values. If the error is substantial for a pre-defined monitoring window (t_w), it is treated as an anomaly and an alarm is raised. However, RVs may incur natural errors caused due to environmental factors. Therefore, to avoid false positives due to natural errors, and to accommodate overshooting of the RV, the IDS accumulates errors in a monitoring window, and compares the aggregated error with a pre-defined threshold. Therefore, instead of performing direct comparison between the real-time control outputs and the predicted control outputs, the detection techniques perform a threshold-based (τ) comparison as shown below.

$$IDS(t_w) = \begin{cases} 1, & if \sum_{t_i}^{t_j} |V_{predicted} - V_{realtime}|_n > \tau \\ 0, & otherwise \end{cases} \tag{5}$$

Attackers can exploit the aforementioned attack detection principle and successfully perform stealthy false data injection attacks on sensor readings and actuator signals in three ways as follows.

First, the error values under the threshold limit for a certain monitoring window are acceptable, and will not be reported as anomalies. Assuming the attacker figures out the threshold, he/she can trigger stealthy attacks by injecting false data to the sensor readings in a controlled manner, causing the RV to gradually deviate from the defined path. By performing such an attack for a long period of time, the attacker will be able to cause a substantial deviation. Because the deviation is within the accepted threshold, the invariant-based techniques (CI and EKF) will not be able to detect it.

Attack Goal	Attack Scenarios	Attack Type	Consequences
Deviate the RV to a desired location	Deviating a delivery drone	False data injection	Drone may deliver a package at wrong location
Influence RVs performance	Disrupting productivity of warehouse rovers	Artificial delay	Rovers may not follow the right organization pattern and products will be stored randomly.
Damage, crash or cause major disruptions	Crash a drone while landing	Switch mode	Delivery items could be damaged

Table 1: Attacker's goal, types of attack and its consequences.

Second, because the detection techniques employ a fixed monitoring window for threshold comparison, an attacker can inject artificial delays, which will obstruct the system from receiving the current set of sensor measurements. Such delays can stop the system for a few seconds, and prevent the system from performing critical operations such as mode changes. The attacker can inject the delay attacks intermittently to avoid accumulating large errors, which might trigger the IDS.

Finally, we found that the invariants derived using CI and EKF are insufficient in providing a close estimate of target angles when the RV switches modes (*e.g.*, when the drone is commanded to land after flying at a fixed height). In other words, the difference between the runtime values and the estimated values becomes larger when the RV switches to *Land* mode from *Waypoint* mode. Therefore, the detection techniques will have to employ a larger threshold to avoid false positives. This enables the attacker to inject large false data, and thereby cause drastic changes in the RV's path.

3.2 Attack Scenarios

Each of the stealthy attacks presented in this paper exploits a weakness in the CI/EKF techniques identified in the previous section. In this section, we discuss the impact of the attacks when performed against RVs in industrial scenarios. Table 1 shows the attackers' goal, the type of attack to achieve the goals, and how the attack would affect the RVs operations in an industrial use-case.

False Data Injection This attack enables the attacker to mutate the sensor measurements to a desirable value (for them). For instance, an attacker may inject false readings to the gyroscopic sensor measurements, which would make the drone unstable. Prior work [12, 51] simulated similar attacks using acoustic noise signals to tamper with the sensors of an RV, causing a major deviation in the intended path of the RV. However, we are interested in performing more subtle mutations to sensor readings. The goal is to simulate subtle and minor deviations in a controlled manner for an extended time period, and to maintain the deviation factor just under the threshold pre-defined by an IDS using CI and EKF. Instead of deviating the drone by a large factor (*e.g.*, 60 degrees) at once (which might trigger the IDS and result in the attack being detected), the attacker can intermittently inject small false data values to the sensor readings. This will influence the control operations causing a difference in the drone's position and angular orientation. By performing such minor deviations for a period of time, the attacker will be able to divert the drone to his/her desired location.

Artificial Delay With this attack, the attacker influences the timing behaviour of the system events or the controller events by injecting artificial delays. Such artificial delays can allow attackers to change the timing of important system actions (*e.g.*, change in mode of operation), delay essential API calls, or cause other controller functionality to be suppressed. For instance, autonomous rovers are increasingly deployed in warehouses to facilitate inventory management and packaging. These rovers receive real-time commands to pick up or drop a package at a given location in the warehouse area. With artificial delay attacks, the attacker can cause an RV to receive a particular command at a delayed time. However, if the RV receives the sensor data of a previous state in the mission, the difference between the estimated behavior and observed behaviour for a pre-defined motioning window will increase. This may potentially trigger an alert by the IDS. Therefore, to maintain stealthiness, the attacker will need to inject such delays intermittently and not perpetually.

Switch Mode The switch mode attack is a form of false data injection launched at highly vulnerable states in the RV's mission. Knowing the current mode of operation the attacker can inject malicious code, which is triggered when the RV switches its mode of operation. For instance, when a drone switches to *Land* mode, a malicious code snippet will overwrite the actuator signals. This will prompt the drone to gain elevation instead of landing, or increase the rotor speed causing the drone to land harder than is safe, potentially resulting in a crash. When such an attack is launched against delivery drones, it may damage packages or may hurt the recipients of the package. Because the attack will not cause the monitoring parameters to exceed the pre-defined threshold, the IDS will not be able to detect it.

3.3 Challenges

The main challenge for the attacker is to launch automated attacks against RVs while remaining undetected by CI and EKF IDS systems, if they are deployed. Therefore, the attacker needs to (1) learn the parameters of the system and the IDS, and (2) manipulate the sensor readings or inject delays by just the right amounts, and at the right times to remain stealthy, even while achieving their aims. This is the challenge we address in this paper.

4 APPROACH

In this section, we describe the approach for performing each of our stealthy attacks. First, we describe our attack model, followed by the steps necessary for preparing and performing the stealthy attacks by an attacker. We then present automated algorithms for executing the stealthy attacks on RVs.

4.1 Attack Model

The goal of the attacker is to i) perform stealthy attacks and prompt deviations in the RV's mission by manipulating sensor measurements and actuator signals in the presence of an IDS using the CI and EKF techniques, and ii) modify the timing behaviour of the system events or control events of the RV and adversely influence

its performance and efficiency. Stealthy means the attack does not cause any unexpected system behaviour that is detected by the IDS. For instance, in a stealthy attack, the sensor readings and actuator signals are within the expected thresholds for a particular state in the RV's mission path.

We assume that the attacker has the following capabilities:

- Can compromise only the RVs (drones or rovers) and no other components (*e.g.*, GCS, telemetry etc.) of the system.
- Can snoop on the control inputs and outputs, and derive the RV's state estimation model (*i.e.*, the state space matrices).
- Can replace the libraries used in the RV's software stack through code injection [4]. This will enable the attacker to modify the program code and inject malicious behaviour.
- Using these capabilities, the attacker can perform targeted spoofing attacks such as compromising the gyroscopic sensor and accelerometer of the RV. The attacker can arbitrarily manipulate sensor readings to his/her desired values.

However, we assume that the attacker cannot tamper with the firmware, cannot have root access to the Operating System (OS), or delete system logs. Furthermore, the attacker does not know the physical properties of the RV, such as the detailed specifications of its shape. In addition, the low-level control parameters (*e.g.*, how the vehicle reacts to control signals) and the commands from the auto-navigation system (*e.g.*, mission semantics of the vehicle) are not known to the attacker. However, the attacker does need to know that the IDS uses CI and/or EKF models to derive invariants - this is so that he/she can modulate the attack accordingly (if not, the attacker can assume both techniques are deployed and be conservative).

4.2 Attack Preparation

Figure 3 presents an overview of the common steps required for carrying out each attack. This section describes the steps in detail.

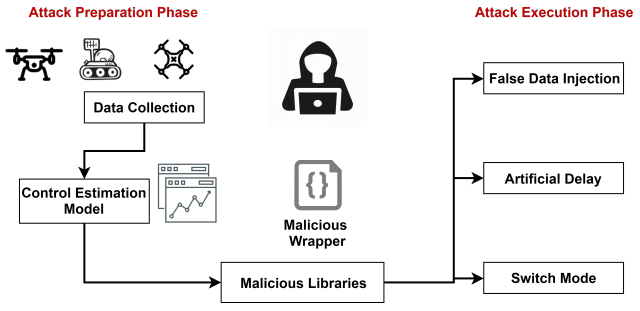


Figure 3: Attack Overview.

Data Collection: The first step in attack preparation is to collect mission profile data of the RV. The attacker can either collect mission profile data from a real RV, or he/she can simulate the missions for the RV to achieve a realistic mission profile. The time series data of the target state $x'(t)$, current state $x(t)$, control input $u(t)$, control output $y(t)$ parameters will be used to derive the state estimation model (*i.e.*, the state space matrices). Ideally, the attacker will collect traces from two control operations namely, position control and attitude control (Figure 4). The *Position Controller* takes the target position as input, and applies the PID control algorithm to calculate the target angles along X, Y, and Z axis). The actual position is looped back as feedback to the controller. The *Attitude Controller* takes the target angle as input, and calculates the motor outputs (rotation speed). The actual angles are looped back to the controller. The attacker will record the parameters pertaining to the above mentioned control operations (*e.g.*, target velocity and actual velocity along x, y, z axis, target acceleration, target and actual angles, angular velocity and angular rate). Ideally, the attacker will collect mission profile data from different mission trajectories, covering multiple modes of operation to generate an accurate state estimation model. However, the data does not have to be comprehensive to derive the state estimation model for RVs [12].

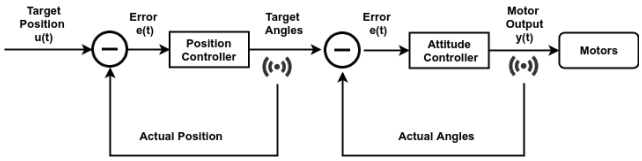


Figure 4: Position and Attitude Controllers in RV.

Control State Estimation Model: Both CI and EKF derive invariants based on the vehicle dynamics and the underlying control algorithm (typically PID control in the case of RVs). The invariant generation process heavily relies on the state estimation model as shown in Equations 1,2, 3 and 4. The attacker's goal is to derive the unknown coefficients for solving the aforementioned equations. The mission profile data collected in the above steps can be used to derive the state estimation model (*i.e.*, state space matrices). To derive the A, B, C, D state space matrices, the attacker can use system identification (SI) [34], which is a process to develop a dynamic model of the system using the system's control input and control output data. From the state space matrices, the attacker can derive the Kalman gain K. The procedure is explained in Appendix A.

Malicious Libraries Typically, the RV's software uses two broad set of libraries for i) control operations such as PID control, attitude estimation (AHRS), and motor mixing etc. ii) sensor operations such as performing inertial measurements, GPS interface, optical interface etc. The APIs are specific to each class of RVs, but do not vary within a class (typically distributed as shared libraries). For example, the Autopilot software stack, which is deployed on many RVs, has a common set of shared libraries. One of the ways the attacker can perform the stealthy attacks is by replacing the original shared libraries with malicious ones. The malicious libraries will contain the attack code snippets. Once the unknown coefficients ($A, B, C, D,$ *and* K) for solving the control equations are derived, the attacker will package them with the malicious library to perform threshold comparisons in runtime.

Malicious Wrapper: The attacker will design a malicious wrapper which will overwrite the original control and sensor libraries with malicious libraries by exploiting the dynamic linking feature [4]. When the RV software makes an API call to the control or sensor libraries, the malicious libraries will be called.

The attacker can also inject acoustic noise at the resonant frequency [51] to achieve the same results. However, because of the difficulties associated with such noise injection (*e.g.*, the noise source has to be in close proximity of the RV, and the impact of the attack is unknown) it will be harder to perform the attacks in realistic settings. Our approach is similar to that of Choi *et al.*[12], who also simulated noise injection through a piece of attack code.

4.3 Attack Execution

False Data Injection (FDI) To perform an FDI attack, the attacker will need to derive the threshold and the monitoring window for the CI and EKF approaches as follows. i) CI approach - To derive the monitoring window, the attacker can use the time series data collected in the steps above to figure out the maximum temporal deviation between the observed control output sequences and the corresponding estimated control output sequences via a sequence alignment algorithm (*e.g.*, [48]). Once the window is obtained, the attacker can calculate the accumulated error in this window and select the accumulated value as the threshold. This is similar to the dynamic time wrapping technique used in Choi *et al.*[12] ii) Leveraging EKF's state correction - EKF accumulates the error between the predicted angular orientation and the measurements of accelerometer and gyroscope in a large matrix called *State Covariance Matrix*. When the error is larger than the threshold, it applies a filter which is referred to as *State Correction*, and the state covariance matrix is updated. The attacker can perform experiments on the simulator by injecting noisy sensor measurements to observe the time interval at which the state covariance matrix is updated. This time interval is the most viable monitoring window the state estimation model based IDS can employ, and the accumulated error in this monitoring window will be the threshold. To remain stealthy, the attacker will manipulate the control input parameters such that the deviations in the control output signal are within the detection threshold of both CI and EKF.

Algorithm 1 shows the algorithm to launch FDI attack on the RV's position controller by manipulating the angular orientation measurements. The function `falseDataInjection` will get triggered when the RV's software components make an API call to the malicious libraries. The pre-computed state space matrices and the threshold values will be packaged with the malicious library (Lines 2 to 5). Based on the error threshold, the attacker will derive a value f for a target sensor. The duration of false data injection t_{attack} is based on the monitoring window. Lines 10 to 13 manipulate the control input $u(t)$ by injecting false data f in the sensor measurements. Lines 18 to 24 manipulate the value of f when the deviation approaches the detection threshold in order to remain stealthy. Since the detection procedure resets the accumulated error (Line 25) for each monitoring window, the attack will not be detected by the CI and EKF techniques.

Artificial Delay The attacker can trigger the artificial delay (AD) attack by including a code snippet in the malicious library called `ArtificialDelay`, which when triggered will perform certain resource intensive operations. Such delays will obstruct other system calls and control operation, thereby disrupting the timing behaviour of the systems. However, if the delay is triggered for a long time period, the error accumulation in the invariant analysis will increase and the IDS might raise an alarm. To remain stealthy under such an IDS, the attacker can use the monitoring window found in the above steps as a threshold (T_{AD}) and not allow delays longer than this threshold. By triggering the snippet `ArtificialDelay` intermittently and under the threshold T_{AD}, the attacker will be able to bypass the detection mechanism. For the artificial delay attack, the algorithm is similar to the one shown in Algorithm 1, where the t_{attack} will be derived based on the monitoring window of CI and EKF techniques. The code snippet `delay_attack` will be triggered when an API in the malicious libraries is triggered. The algorithm for artificial delay attack is presented in Appendix B.1

Algorithm 1: False data injection in sensor readings

```
1  Function falseDataInjection():
2      A, B, C, D: pre-calculated state-space matrices.
3      K: pre-calculated Kalman gain.
4      T_CI : pre-defined threshold for CI.
5      T_EKF : pre-defined threshold for EKF.
6      t_attack: duration of attack.
7      f: false data
8      while (t_attack) do
9          T_Angle ← Target angle;
10         A_Angle ← Actual angle; (read data from sensor)
11         A_Angle ← A_Angle + f;
12         attitude target_X = A_angle − T_angle;
13         u = attitude target_X;
14         X_n = A ∗ x + B ∗ u;
15         Y_Roll = C ∗ x + D ∗ u;
16         R = Y_Angle − C ∗ X_n;
17         d = |Y_Angle − T_Angle|;
18         error_CI = error_CI + d;
19         error_EKF = error_EKF + R;
20         d_sum = d_sum + d
21         if d_sum > T_CI and d_sum > T_EKF then
22             f = 0;
23         end
24     end
25     error_CI, error_EKF, d_sum = 0;
26 return T_Angle;
```

Switch Mode The switch mode (SM) attack is a form of FDI attack launched at a few, highly vulnerable states of an RV mission. To execute this attack, in addition to the detection threshold and the monitoring window as per the CI and EKF techniques, the attacker will have to monitor the mode of operations of the RV. Algorithm 2 shows an example of switch mode (SM) attack launched when the RV changes its operations to *LAND* mode (Line 11). The attacker can also launch such attacks at other mode transitions (*e.g.*, from *Takeoff* to *Waypoint*). Similar to the FDI attack, here the attacker will derive a value f, which when injected to the motor thrust value will disrupt the RV's behaviour (Line 23). Further, to remain stealthy this attack will be carried out for a specific attack duration t_{attack}, which is derived based on the monitoring window. In this case, as

Algorithm 2: Switch mode attack - influencing actuator signals

```
Function SwitchModeAttack():
    A, B, C, D: pre-calculated state-space matrices.
    K: pre-calculated Kalman gain.
    T_CI : pre-defined threshold for CI.
    T_EKF : pre-defined threshold for EKF.
    t_attack: duration of attack;
    f: false data;
    while i<num-motors do
        T_motor = getPWMOutput(i);
        Mode = getCurrentMode();
        if Mode = LAND then
            while (t_attack) do
                X_n = A * x + B * u;
                Y_motor = C * x + D * u;
                R = Y_motor − C * X_n;
                d = |Y_motor − T_motor|;
                error_CI = error_CI + d;
                error_EKF = error_EKF + R;
                d_sum = d_sum + d
                if d_sum > T_CI or d_sum > T_EKF then
                    f = 0;
                end
                motor[i] = thrustToPWM() + f;
            end
        else
            motor[i] = thrustToPWM();
        end
    end
end
```

the threshold is found to be larger than normal, the attacker can inject larger false values which may result in severe consequences in a short time duration.

5 EXPERIMENTAL EVALUATION

In this section, we discuss the experimental setup, followed by the research questions (RQs) we ask. Then, we present the results of the experiments to answer the RQs.

5.1 Experimental Setup

To demonstrate the stealthy attacks, we use both real RVs and simulation platforms. We use two types of RVs in each, namely (1) Pixhawk based drone [37] (henceforth called *Pixhawk drone*) , and an Aion R1 [47] ground rover (henceforth called *R1 rover*) for real RVs, and (2) quadcopter (henceforth called *ArduCopter SITL*), and ground rover (henceforth called *ArduRover SITL*) for simulation. Figure 5 shows the real RVs used to test the stealthy attacks. For vehicle simulation, we use APM SITL [7] and JSMSim [27]. We run the simulators on an Ubuntu 16.0 64-bit machine with Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz processor and 8 GB RAM.

(a) Pixhawk based DIY drone

(b) Aion R1 Rover

Figure 5: RVs used for Experiments.

Software All the subject vehicles are based on ArduPilot's software suite [7]. Though there are other popular software suites such as PX4 [55] and Paparazzi [54], we chose to use ArduPilot because it is the largest open-source autopilot software suite deployed in over one million UAVs [23], and supports a wide variety of RV hardware platforms (*e.g.*, Pixhawk, Bebop, Navio etc.) [53]). That said, our attacks are not tied to a specific hardware or software platform.

Hardware Both the drone and the rover (Figure 5) used in our experiments are based on the Pixhawk platform [37]. Pixhawk is an ARM Cortex based all-in-one hardware platform, which combines flight management unit (FMU) controller, I/O, sensors and memory in a single board. It runs NuttX, which is a Unix-based real-time operating system, on a 32-bit Cortex processor and 256 KB RAM [58].

Attack Setup We performed 20 missions on both the simulations and the real vehicles, and collected the time series data of control input $u(t)$, system state $x(t)$, and the control output $y(t)$. The time series data was collected from position control and attitude control operations of the RV (Figure 4). These data sets were used to derive the state estimation model [1].

To perform the attacks, we designed a set of malicious libraries for the following control libraries of the ArduPilot software suite: `AHRS`, `AttitideControl`, and `PositionControl`. We overwrote the environment variable (*LD_LIBRARY_PATH*) in the *.bashrc* file to point to the malicious libraries instead of the originals - this technique has been used in prior work as well [4][2]. Henceforth, when the RV software components call functions defined in the above libraries, the corresponding function in the malicious library will be called (as the malicious libraries have a function with same name as defined in the original library). The malicious libraries can be timed to stay dormant until the RV is deployed on a critical mission, at which point, they can get triggered.

5.2 Research Questions

RQ1. How much effort does the attacker need to expend to derive the state estimation model?

RQ2. What are the impacts of the stealthy attacks on the subject RVs?

RQ3. How effective are the attacks in achieving the attacker's objectives?

[1] All the code and data-sets used in this paper can be found at *https://github.com/DependableSystemsLab/stealthy-attacks*

[2] This can also be done by executing a Trojan program or shell code, for example.

5.3 Results

In this section, we present the results of the stealthy attacks experiments performed on the subject RVs to address the RQs.

RQ1: Attacker's effort The first set of experiments aim to quantify the effort required on the attacker's part in deriving an accurate state estimation model for a subject RV. We divided the mission data into two sets: i) Model extraction set - used to derive the state estimation model (15 missions for each subject RV, both simulation and real vehicles), and ii) Model testing set - used to test the accuracy of the obtained state estimation model (5 missions for each subject RV, both simulation and real vehicles).

We followed an iterative approach in deriving the state estimation model and evaluating its accuracy. In the first iteration, we randomly picked 5 mission profiles from the model extraction set and using system identification [34, 35], we derived the A, B, C, D matrices and the Kalman gain K. Following Equations 1, 2, 3, and 4, we estimated the system output (*e.g.*, roll, pitch, yaw) for missions in the model testing set. Then we analyzed the accuracy of the state estimation model by comparing the estimated and the realtime system outputs.

For each subsequent iteration, we added 1 more mission profile data from the model extraction set, derived an updated model, and performed the above analysis to identify if the accuracy of the state estimation model has converged. From this experiment, we found that all model estimated outputs converged with the realtime outputs by the third iteration. For some subject RVs, the convergence occurred after the first iteration. Overall, the model converged with just 5 to 7 mission data, and hence the attacker can derive an accurate state estimation model with modest effort.

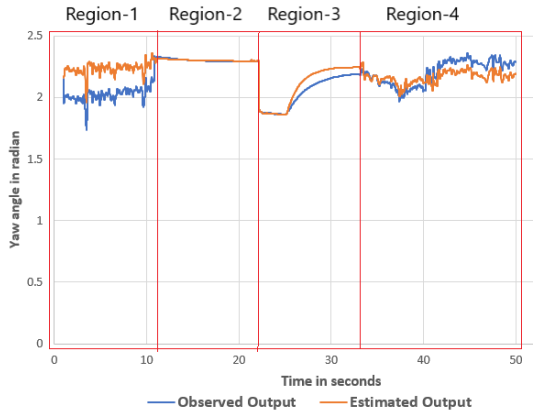
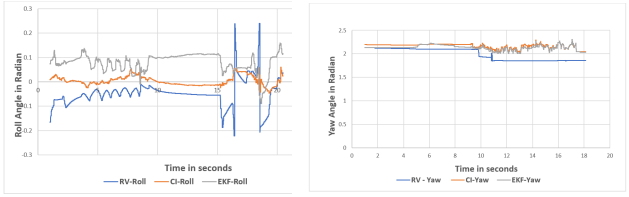


Figure 6: State space model - Realtime vs Estimated values.

Even in cases where the model converged, for some states of the RV's mission path, the state estimation model failed to provide precise estimates. Figure 6 shows an example of the Pixhawk drone, where the model did not converge. We have divided the graph into 4 regions: 1, 2, 3, and 4, based on the different modes of the RV's mission. As can be seen in the figure, the model estimated output converges with the realtime outputs in Regions 2 and 3, but not in Regions 1 and 4. The RVs realtime control outputs relies on the realtime sensor measurements (control input $u(t)$) and the P, I, D gains (Section 2). Based on the RV's trajectory and its current mode of operation, the sensor measurements may incur additional noise. The PID control functions may consequently adjust the gain param to mitigate the effects of the noise, which will influence the realtime control outputs. However, the model estimated control outputs are not affected by runtime PID parameter adjustments. Therefore, it is difficult to achieve 100% convergence between the model estimated and the realtime values.

RQ2: Impact of the stealthy attacks In the second set of experiments, we performed the stealthy attacks in the presence of an IDS using the CI and EKF models respectively. Before performing the attacks, the attacker will have to derive the detection threshold and the monitoring window. For the CI model, we followed the dynamic time wrapping method (explained in Section 4), to derive the monitoring window. To identify the monitoring window of the EKF model, we performed experiments in ArduCopter SITL by injecting smaller noise into the sensor measurements and observed the intervals at which the state covariance matrix is updated (explained in Section 4) . The detection thresholds and monitoring windows for all the subject RVs are presented in Table 2.

Table 2 also shows the impact of the attacks on the subject RVs. The results shown in the table are based on data from 5 missions, and consist of the average values of the attack's outcomes (deviations, delays) in the presence of CI-based IDS, EKF-based IDS, and both together. Our results show that the thresholds set by CI and EKF model allow a considerable margin for stealthy attacks to be launched. Though the attacks do not always cause drastic repercussions, they cause substantial deviation in the RVs trajectory (deviation of 8 to 15 m for a mission distance of 35-50 m), and adversely influence their efficiency by increasing the mission duration by 30% to 50%. We discuss a few examples of the attacks.



(a) Aion R1 Rover **(b) Pixhawk Drone**

Figure 7: FDI attacks on subject RVs.

False Data Injection. For all the subject vehicles, we injected false data in the sensor measurements as per Algorithm 1 which influences the *yaw, roll and pitch* angles. The injected false data modified the angles in the range $0 - 45$ degrees. Figure 7 shows how the FDI attack manipulates the RV's Euler angles. The intermittent and controlled false data injection prevents accumulation of large error within the monitoring windows, thereby bypassing the CI and EKF invariant analysis. Table 2 shows that for the Pixhawk drone, the average deviation caused by FDI attack is 11 m for a mission distance of 50 m. The deviation is much larger than standard GPS offsets [46] (video can be found at [43]). As it is difficult to perform experiments on real vehicles with large mission duration and distance, we instead perform detailed analysis of this attack

on our simulator by increasing the mission distance (from 100m to 5000m). We found that the deviation increases almost linearly with mission distance. Therefore, for a mission distance of 5000 m, the FDI attack can deviate the drone by more than 100 m.

Artificial Delay The artificial delay attacks were also launched intermittently, and the duration of the delay was under the monitoring window duration found above. For a given monitoring window, we injected delay attacks in the vehicles' position controller estimations. We found that these attacks were more disruptive for the rovers than the drones, both in the real world and in simulations (video can be found at [42]). As shown in Table 2, the attack increases the mission time of the rover and drones by more than 50% and 30% respectively.

Switch Mode We only applied the switch mode attack on drones (both real and simulated), as the rovers used in our experiments only had a few operational modes, and hence did not experience many mode transitions. Figure 8 shows the divergence between the roll angle and the realtime roll angles when the Pixhawk drone switches to *Land* mode. As we discussed above (Figure 6), the model estimated values and the realtime values do not converge for all modes, and hence we posit that the detection threshold should accommodate large offsets to prevent false positives. We found that for the Pixhawk drone, the offset was as large as 11 degrees. This enabled us to inject large false values into the roll angle during mode switching (Figure 8), which resulted in major disruptions.

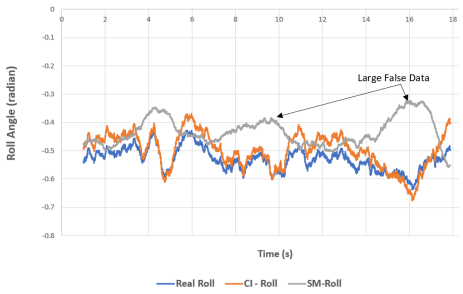


Figure 8: Switch Mode Attack against Pixhawk Drone.

RQ3: Effectiveness of the attacks From the above experiments, we found that the false data injection attack can cause a deviation of 30 m (for a mission distance of 100 m) in a drone's trajectory. The deviation increases to more than 100 m for a mission distance of 5 km. Typically, drones deployed in industrial use-cases such as package delivery, surveillance, etc., will operate autonomously for a mission duration of about 30 minutes [8, 10], and cover a distance up to 20 km [24]. In such missions, the impact of the FDI attacks can be much more significant.

The switch mode attacks can also cause major repercussions, even for short missions. From our experiments (not presented in the table) we found that for a mission distance of 50 m, the switch mode attack prevented the drone from flying to the destination. Instead the drone loitered at a certain height. In another instance, the attack caused the drone to ignore the "land" command, and the drone kept gaining elevation (video can be found at [44]). Further, in simulations, we were able to crash the drone by strategically launching the switch mode attack when the drone switched to the *Land* mode. When such attacks are launched against drones in industrial use-cases, it can damage the drone and other nearby objects as well.

The artificial delay attack increased the mission duration by more than 50% for rovers and by 30% for drones. Although this attack does not directly deviate or damage the RVs, it can have major performance and efficiency related consequences when launched against RVs in industrial use-cases. For example, drones are used for delivery of time critical items such as blood samples and drugs [8, 10], and rovers are used to increase the productivity in warehouses [52]. Delay attacks can have adverse consequences in these cases.

6 DISCUSSION

The main limitation of our attack approach is that the state estimation model, as well as the threshold and monitoring window values, vary for RVs using different hardware platforms (*e.g.*, the model derived from Pixhawk drone does not not apply to Bebop2 drone). Therefore, the attacker will have to expend the effort of repeating the steps in the *Attack Preparation Phase* for each class of RVs. In this section, we present the design of a self-learning malware program that will attack an RV adaptively without any human effort. Then, we discuss the other limitations of our attacks, followed by a discussion on how IDSes can be better designed for dynamic CPS such as RVs. Finally, we discuss some of the threats to the validity of our results.

6.1 Self-Learning Malware

We propose a technique though which an attacker with the same capabilities as in our attack model (Section 4.1) can automate the attack preparation phase through a self-learning malware. The attacker can design a program called `modelExtractor` to collect the sensor measurement and the mission profile data from position controller and attitude controller. The `modelExtractor` will work in tandem with the malicious library on the RV.

For each mission, the `modelExtractor` will collect the data, and create an archive of mission profile data for various mission trajectories and mode of operations of the RV. As we said earlier (Section 4.2), the mission profile data from 5-7 missions is sufficient to derive an accurate state estimation model. After the RV has completed n missions, the `modelExtractor` will trigger a system identification library [18, 41] to derive the state space matrices A, B, C, D and the Kalman gain K.

From our experiments, we have found that the values of the monitoring window and the detection threshold based on EKF are typically smaller than those of CI. Therefore, the attacker can focus on deriving the EKF's threshold and monitoring window, by recording the time intervals at which EKF's state covariance matrix is updated. This way, the monitoring window can be extracted, and the error accumulated in this monitoring window will be the threshold. The `modelExtractor` program will pass these values to the attack algorithms (Algorithm 1, Algorithm 2), which will trigger the attacks based on the RVs mode of operation and mission state. While we have not implemented such a malware program and hence cannot measure its overhead, our preliminary experiments

Vehicle Type	Attack Types	Mission Distance (in m)	Flight Time (in sec)	Threshold CI - IDS (in degrees)	Monitoring window CI-IDS (in seconds)	Deviation CI-IDS (in m)	Threshold EKF - IDS (in degrees)	Monitoring window EKF-IDS (in seconds)	Deviation EKF-IDS (in m)	Deviation CI & EKF (in m)	Attack Impact
ArduCopter SITL	FDI	50	45	60 (yaw angle estimate)	2.0	11	45 (yaw angle estimate)	2.3	8	8	drone deviated from its trajectory
	SM	50	49			7			7	7	drone landed (crash landing) at wrong location
	AD	50	71			-			-	-	mission took more than 20s longer than usual
ArduRover SITL	FDI	50	42	72 (roll angle estimate)	2.6	14	60 (roll angle estimate)	2.3	11	11	rover followed wrong path
	AD	50	72			-			-	-	mission took more than 30s longer than usual
Pixhawk Quadcopter	FDI	50	32	60 (yaw angle estimate)	2.0	11	45 (yaw angle estimate)	1.9	8	8	drone deviated from its defined trajectory
	SM	50	34			6			6	6	drone landed at wrong location
	AD	50	41			-			-	-	mission took 10s longer than usual
Aion R1 Rover	FDI	36	35	82 (roll angle estimate)	2.6	11.2	60 (roll angle estimate)	2.5	9	9	rover followed wrong path
	AD	36	59			-			-	-	mission took more than 20s longer than usual

Table 2: Results of the attacks on different types of RVs and the impacts of the attacks. Note that SM attacks are only applicable to drones and not to rovers in our experiments.

on performing these measurements and calculating the thresholds, indicate that the overhead of the malware will likely be small.

6.2 Limitations

In FDI attacks, the value of the false data to be injected is calculated dynamically based on the threshold and the current state of the system in order to remain stealthy. However, in some situations the threshold values based on the CI and EKF models will not allow much room for performing FDI. For example, for a Bebop2 drone, the CI model employs a threshold of 5.6 degrees (roll angle estimation) for a monitoring window of 1.8s [12]. We did not have access to a Bebop2 drone to perform the attacks ourselves. Therefore, we performed experiments in the simulator using the thresholds and monitoring window for Bebop2 [12]. We found that for a mission distance of 50 meters, the deviation caused by the FDI attack was only 4 meters, which was considerably smaller than what we observed for the Pixhawk drone.

Further, in some of our experiments, we found that if the drone overshoots its trajectory because of the injected false values (for example, in a switch mode attack, where we injected large false values) the drone system activates the (hardware) fail-safe mode and forces the drone to land amidst the mission. This can be considered a limitation of our attack because the attack failed in achieving the desired outcome. However, an attacker can take advantage of such a fail-safe landing mechanism, and force the drone to land at a location conducive to the attacker (different from the original one).

6.3 Countermeasures

One way to mitigate the attacks in this paper is to design adaptive thresholds and variable monitoring windows for the IDS techniques. The conventional methods for invariant extraction (both CI and EKF) use pre-defined fixed thresholds and monitoring windows for a subject RV. We exploit this notion of fixed bounds invariant analysis to trigger our attacks. If the IDS uses an adaptive threshold (*e.g.*, different threshold for steady state flight and $Land/Takeoff$ modes), the leeway for injecting false values into sensor and actuator parameters will decrease, which in turn will reduce the impact of the FDI and switch mode attacks. Similarly, if the IDS employs variable-sized monitoring windows, the impacts of artificial delay attack will decrease. If the attacker injects a fixed size delay (as we do in our artificial delay attack), an IDS using a variable sized monitoring window may detect the attack. We defer detailed design of mitigation techniques for these attacks to future work.

6.4 Threats to Validity

We consider three threats to validity - i) Internal, ii) External and iii) Construct. An internal threat to our work is that we have considered only control-based attack detection techniques. Although we do not evaluate other attack detection techniques against our stealthy attacks, we posit that methods that follow a threshold based detection such as CI and EKF are vulnerable to our stealthy attacks. Another internal threat is that our experiments with real vehicle were not very extensive. This is largely because of the restrictions and regulations around flying unmanned RVs, as well as time limitations. However, we mitigate this threat by performing extensive unbiased experiments on a simulator.

An external threat to work is that we used only one software platform (i.e. Ardupilot) and one type of hardware platform (i.e. Pixhawk). RV software mostly uses pre-designed libraries for control and sensor operations. Therefore, our stealthy attacks can also be extended to other RV software platforms such as PX4[55] and Paparazzi [54], which will require development of platform specific malicious libraries.

Finally, the construct threat to validity is that if the detection threshold and monitoring windows are small, the effects of the stealthy attacks will not be as critical. However, as the detection threshold in CI and EKF methods is calculated using training traces, it is difficult to come up with small and precise threshold boundaries, for reasons such as sensor noise and environmental factors. Moreover, an aggressively calculated small detection threshold may result in high false positives, which is undesirable. Secondly, even with small detection windows, the artificial delay attack will still be able to cause undesirable consequences in the RV mission.

7 RELATED WORK

Sensor spoofing attacks. Previous work has demonstrated sensor spoofing attacks such as GPS spoofing [25][56] to misguide a drone's trajectory, optical spoofing [17] to gain an implicit control channel etc. It has also been shown that inaudible sound noise when injected at resonance frequency can adversely affect the MEMS gyroscopic sensor, which can cause the drone to crash [51]. Likewise, attackers can compromise the accelerometer of drones by injecting acoustic noise in a controlled manner [57]. Jha *et al.*[26] developed a ML-based fault injection technique for autonomous vehicles (AV) that will modify the states of the AV that are most vulnerable. However, these attacks are not necessarily stealthy, as they can be detected by the IDS depending on the degree of deviation they cause. In contrast, our attacks are designed to be stealthy.

Stealthy attacks such as false data injection have been demonstrated on industrial control systems to mislead state estimators and controllers [16, 32, 36]. Stealthy sensor spoofing attacks can induce the supervisor control layer into allowing the system to reach an unsafe state, thereby causing physical damage to the system [21]. Our attacks cause perturbations in the sensor measurements thereby inhibiting the RV from performing its task, and not necessarily causing physical damage (which is easier to detect).

Malware attacks Multiple instances of malware attacks on industrial control systems have been reported. A few prominent examples are the Stuxnet attack [28], and the attack on the power grid in Ukraine [30]. Malware with learning capabilities can derive an optimal attack vector and launch man-in-the-middle attacks [20]. Alamzadeh *et al.* [4] present a malware attack targeting a teleoperated surgical robot, where the malware identities an optimal attack time and injects faults. Similar attacks have been demonstrated on pacemakers [22]. Chung *et al.*[14] demonstrated feasibility of attacking water treatment systems using a self-learning malware. Subsequently, they [15] extended their work to launch MITM attacks on surgical robots by exploiting the vulnerabilities in the underlying runtime environment (ROS). However, none of these attacks have been demonstrated on RVs protected with control-based IDS as in our work (to the best of our knowledge).

Intrusion Detection Systems (IDS) IDS have been proposed that uses physical invariants for each sensor and actuator to tackle attacks against different cyber-physical systems, including UAVs [39]. BRUIDS [40] is a specification based IDS that is adaptive based on the attacker type and environment changes. CORGIDS [2] derives correlations between physical properties using hidden Markov models, and uses these correlations as invariants. Choudhury *et al.*[13] uses scheduling invariants and physical invariants in the form of Lyapunov functions for attack detection. Adepu *et al.*[1] design an IDS for a water treatment plant by manually describing the invariants for a particular sensor in terms of the water level changes between two consecutive readings. ARTINALI [5] dynamically mines data, time and event invariants from an execution trace of the program and use data-time-event invariants to detect anomalies. Chen et al. [11] present an approach for automatically constructing invariants of CPS, by using supervised ML on traces of data obtained from systematically mutated PLC programs. Ahmed *et al.*[3] propose a technique to fingerprint the sensor and process noise to detect sensor spoofing attacks. Kim *et al.*[29] proposed a technique for detecting control semantic bugs (input validation bugs) leveraging a control instability detector to detect RV malfunctions. Most of the above IDS [2, 39, 40] use a threshold based technique to detect deviations from the invariants or models. Therefore, they are vulnerable to stealthy attacks like ours. That said, we did not consider these IDS in our evaluations as our attacks target control-based IDS techniques.

8 CONCLUSION

In this paper, we highlight the vulnerabilities in control-theory based techniques used for attack detection in RVs. We find that these techniques use pre-defined detection threshold and monitoring window based invariant analysis techniques, and are hence susceptible to stealthy attacks.

To demonstrate how an attacker can exploit the vulnerabilities, we designed three stealthy attacks namely: false data injection, artificial delay attack and switch mode attack. We present algorithms that will automate the process of deriving the detection thresholds. Knowing the threshold, an attacker can perform stealthy sensor and actuator spoofing attacks, thereby bypassing the detection mechanisms. Though the attacks are stealthy in nature, and do not cause large-scale disruptions, we found that the consequences can still be quite severe such as: deviating a drone by more than 100 meters from its trajectory, increasing the mission duration of a rover and drone by more than 50% and 30% respectively, and causing a drone to crash while landing (or harming other objects). Furthermore, we discuss the attacker's goals in the context of industrial use-cases, and discuss how the attacker can perform stealthy attacks to achieve his/her goals.

In our future work, we intend to design an intelligent malware that will enable stealthy attacks to be mounted on RVs. We will also explore the design of intrusion detection techniques that can detect such attacks by adjusting its thresholds on the fly.

ACKNOWLEDGEMENT

This research was supported by a research grant from the Natural Sciences and Engineering Research Council of Canada (NSERC), and a research gift from Intel. We thank Prof. Ryozo Nagamune, Department of Mechanical Engineering, University of British Columbia for his valuable feedback. We also thank the anonymous reviewers of ACSAC'19 for their comments which helped improve the paper.

REFERENCES

[1] Sridhar Adepu and Aditya Mathur. 2016. Using process invariants to detect cyber attacks on a water treatment system. In *IFIP International Information Security and Privacy Conference*. 91–104.

[2] Ekta Aggarwal, Mehdi Karimibiuki, Karthik Pattabiraman, and André Ivanov. 2018. CORGIDS: A Correlation-based Generic Intrusion Detection System. In *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy (CPS-SPC '18)*. ACM, New York, NY, USA, 24–35. https://doi.org/10.1145/3264888.3264893

[3] Chuadhry Mujeeb Ahmed, Jianying Zhou, and Aditya P. Mathur. 2018. Noise Matters: Using Sensor and Process Noise Fingerprint to Detect Stealthy Cyber Attacks and Authenticate Sensors in CPS. In *Proceedings of the 34th Annual Computer Security Applications Conference (ACSAC '18)*. ACM, New York, NY, USA, 566–581. https://doi.org/10.1145/3274694.3274748

[4] H. Alemzadeh, D. Chen, X. Li, T. Kesavadas, Z. T. Kalbarczyk, and R. K. Iyer. 2016. Targeted Attacks on Teleoperated Surgical Robots: Dynamic Model-Based Detection and Mitigation. In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 395–406. https://doi.org/10.1109/DSN.2016.43

[5] Maryam Raiyat Aliabadi, Amita Ajith Kamath, Julien Gascon-Samson, and Karthik Pattabiraman. 2017. ARTINALI: Dynamic Invariant Detection for Cyber-physical System Security. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2017)*. ACM, New York, NY, USA, 349–361. https://doi.org/10.1145/3106237.3106282

[6] Amazon Prime [n. d.]. Amazon Prime Delivery. Retrieved January 24, 2019 from https://www.amazon.com/Amazon-Prime-Air/b?node=8037720011

[7] ArduPilot [n. d.]. Ardupilot - Software in the Loop. Retrieved May 24, 2018 from http://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html

[8] Aryn Baker. [n. d.]. Zipline Drone Delivery. Retrieved January 24, 2019 from http://www.flyzipline.com/

[9] P.-J. Bristeau, E. Dorveaux, D. VissiÃÍre, and N. Petit. 2010. Hardware and software architecture for state estimation on an experimental low-cost small-scaled helicopter. *Control Engineering Practice* 18, 7 (2010), 733 – 746. https://doi.org/10.1016/j.conengprac.2010.02.014 Special Issue on Aerial Robotics.

[10] Stephen Burns. [n. d.]. Drone meets delivery truck. Retrieved May 24, 2019 from https://www.ups.com/us/es/services/knowledge-center/article.page?name=drone-meets-delivery-truck&kid=cd18bdc2

[11] Y. Chen, C. M. Poskitt, and J. Sun. 2018. Learning from Mutants: Using Code Mutation to Learn and Monitor Invariants of a Cyber-Physical System. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, 648–660. https://doi.org/10.1109/SP.2018.00016

[12] Hongjun Choi, Wen-Chuan Lee, Yousra Aafer, Fan Fei, Zhan Tu, Xiangyu Zhang, Dongyan Xu, and Xinyan Deng. 2018. Detecting Attacks Against Robotic Vehicles: A Control Invariant Approach. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18)*. ACM, New York, NY, USA, 801–816. https://doi.org/10.1145/3243734.3243752

[13] A. Choudhari, H. Ramaprasad, T. Paul, J. W. Kimball, M. Zawodniok, B. McMillin, and S. Chellappan. 2013. Stability of a Cyber-physical Smart Grid System Using Cooperating Invariants. In *2013 IEEE 37th Annual Computer Software and Applications Conference*. 760–769. https://doi.org/10.1109/COMPSAC.2013.126

[14] Keywhan Chung, Zbigniew T. Kalbarczyk, and Ravishankar K. Iyer. 2019. Availability Attacks on Computing Systems Through Alteration of Environmental Control: Smart Malware Approach. In *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS '19)*. ACM, New York, NY, USA, 1–12. https://doi.org/10.1145/3302509.3311041

[15] Keywhan Chung, Xiao Li, Peicheng Tang, Zeran Zhu, Zbigniew T. Kalbarczyk, Ravishankar K. Iyer, and Thenkurussi Kesavadas. 2019. Smart Malware that Uses Leaked Control Data of Robotic Applications: The Case of Raven-II Surgical Robots. In *22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019)*. USENIX Association, Chaoyang District, Beijing, 337–351. https://www.usenix.org/conference/raid2019/presentation/chung

[16] G. Dan and H. Sandberg. 2010. Stealth Attacks and Protection Schemes for State Estimators in Power Systems. In *2010 First IEEE International Conference on Smart Grid Communications*. 214–219. https://doi.org/10.1109/SMARTGRID.2010.5622046

[17] Drew Davidson, Hao Wu, Rob Jellinek, Vikas Singh, and Thomas Ristenpart. 2016. Controlling UAVs with Sensor Input Spoofing Attacks. In *10th USENIX Workshop on Offensive Technologies (WOOT 16)*. USENIX Association, Austin, TX. https://www.usenix.org/conference/woot16/workshop-program/presentation/davidson

[18] ETH-Agile and Dexterous Robotics Lab. [n. d.]. Control Toolbox. https://ethz-adrl.github.io/ct/ct_doc/doc/html/index.html

[19] Gene F. Franklin, J. David Powell, and Abbas Emami-Naeini. 2018. *Feedback Control of Dynamic Systems (8th Edition) (What's New in Engineering)*. Pearson. https://www.amazon.com/Feedback-Control-Dynamic-Systems-Engineering/dp/0134685717?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0134685717

[20] Luis Garcia, Ferdinand Brasser, Mehmet Hazar Cintuglu, Ahmad-Reza Sadeghi, Osama A. Mohammed, and Saman A. Zonouz. 2017. Hey, My Malware Knows Physics! Attacking PLCs with Physical Model Aware Rootkit. In *NDSS*.

[21] R. M. GÃşes, E. Kang, R. Kwong, and S. Lafortune. 2017. Stealthy deception attacks for cyber-physical systems. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. 4224–4230. https://doi.org/10.1109/CDC.2017.8264281

[22] D. Halperin, T. S. Heydt-Benjamin, B. Ransford, S. S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W. H. Maisel. 2008. Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defenses. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*. 129–142. https://doi.org/10.1109/SP.2008.31

[23] Jason Baker (Red Hat). [n. d.]. Open source drone projects. https://opensource.com/article/18/2/drone-projects

[24] Andrew J. Hawkins. [n. d.]. UPS will use drones to deliver medical supplies in North Carolina. Retrieved May 24, 2019 from https://www.theverge.com/2019/3/26/18282291/ups-drone-delivery-hospital-nc-matternet

[25] Todd E. Humphreys. 2008. Assessing the Spoofing Threat: Development of a Portable GPS Civilian Spoofer. In *In Proceedings of the Institute of Navigation GNSS (ION GNSS*.

[26] Saurabh Jha, Subho S. Banerjee, Timothy Tsai, Siva Kumar Sastry Hari, Michael B. Sullivan, Zbigniew T. Kalbarczyk, Stephen W. Keckler, and Ravishankar K. Iyer. 2019. ML-based Fault Injection for Autonomous Vehicles: A Case for Bayesian Fault Injection. *CoRR* abs/1907.01051 (2019). arXiv:1907.01051 http://arxiv.org/abs/1907.01051

[27] JSMSim [n. d.]. JSBSim Open Source Flight Dynamics Model. Retrieved May 24, 2018 from "http://jsbsim.sourceforge.net/"

[28] S. Karnouskos. 2011. Stuxnet worm impact on industrial cyber-physical system security. In *IECON 2011 - 37th Annual Conference of the IEEE Industrial Electronics Society*. 4490–4494. https://doi.org/10.1109/IECON.2011.6120048

[29] Taegyu Kim, Chung Hwan Kim, Junghwan Rhee, Fan Fei, Zhan Tu, Gregory Walkup, Xiangyu Zhang, Xinyan Deng, and Dongyan Xu. 2019. RVFuzzer: Finding Input Validation Bugs in Robotic Vehicles through Control-Guided Testing. In *28th USENIX Security Symposium (USENIX Security 19)*. USENIX Association, Santa Clara, CA, 425–442. https://www.usenix.org/conference/usenixsecurity19/presentation/kim

[30] Robert M. Lee, Michael J. Assante, and Tim Conway. 2016. Analysis of the Cyber Attack on the Ukrainian Power Grid. Technical Report. *Electricity Information Sharing and Analysis Center (E-ISAC)* (2016).

[31] J. Li and Y. Li. 2011. Dynamic analysis and PID control for a quadrotor. In *2011 IEEE International Conference on Mechatronics and Automation*. 573–578. https://doi.org/10.1109/ICMA.2011.5985724

[32] Yao Liu, Peng Ning, and Michael K. Reiter. 2009. False Data Injection Attacks Against State Estimation in Electric Power Grids. In *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS '09)*. ACM, New York, NY, USA, 21–32. https://doi.org/10.1145/1653662.1653666

[33] K. Manandhar, X. Cao, F. Hu, and Y. Liu. 2014. Detection of Faults and Attacks Including False Data Injection Attack in Smart Grid Using Kalman Filter. *IEEE Transactions on Control of Network Systems* 1, 4 (Dec 2014), 370–379. https://doi.org/10.1109/TCNS.2014.2357531

[34] MATLAB. [n. d.]. System Identification Overview. ([n. d.]). https://www.mathworks.com/help/ident/gs/about-system-identification.html

[35] MATLAB. [n. d.]. System Identification Toolbox. ([n. d.]). https://www.mathworks.com/products/sysid.html

[36] S. McLaughlin and S. Zonouz. 2014. Controller-aware false data injection against programmable logic controllers. In *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. 848–853. https://doi.org/10.1109/SmartGridComm.2014.7007754

[37] Lorenz Meier, Petri Tanskanen, Friedrich Fraundorfer, and Marc Pollefeys. 2011. Pixhawk: A system for autonomous flight using onboard computer vision. In *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2992–2997.

[38] MARS 2020 Mission. [n. d.]. MARS Exploration Rover. https://mars.nasa.gov/mer/mission/rover/

[39] Robert Mitchell and Ing-Ray Chen. 2012. Specification based intrusion detection for unmanned aircraft systems. *In Proceedings of the first ACM MobiHoc workshop on Airborne Networks and Communications* (2012), 31–36.

[40] Robert Mitchell and Ing-Ray Chen. 2014. Adaptive intrusion detection of malicious unmanned air vehicles using behavior rule specifications. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 44, 5 (2014), 2014.

[41] GNU Octave. [n. d.]. GNU Octave Scientific Programming Language. https://www.gnu.org/software/octave/

[42] Out of Control. [n. d.]. Artificial Delay Attack Demo Video. https://drive.google.com/open?id=1_CHITopKSraKZXnAIyeUoQZqki8fgUXe

[43] Out of Control. [n. d.]. False Data Injection Attack Demo Video. https://drive.google.com/open?id=1JgrCpwspsBiYdNvKUxeKnl-bZQ9WS_Cg

[44] Out of Control. [n. d.]. Switch Mode Attack Demo Video. https://drive.google.com/open?id=1yUSGa5GoBQYl0GTiTbcPFN5NnjBHXL-Y

[45] Hadi Ravanbakhsh, Sina Aghli, Christoffer Heckman, and Sriram Sankaranarayanan. 2018. Path-Following through Control Funnel Functions. *CoRR* abs/1804.05288 (2018). arXiv:1804.05288 http://arxiv.org/abs/1804.05288

[46] Brent A. Renfro, Miquela Stein, Nicholas Boeker, Emery Reed, and Eduardo Villalba. [n. d.]. An Analysis of Global Positioning System (GPS) Standard Positioning Service (SPS) Performance for 2018.

[47] Aion Robotics. [n. d.]. R1 ArduPilot Edition. https://docs.aionrobotics.com/en/latest/r1-ugv.html

[48] H. Sakoe and S. Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26, 1 (February 1978), 43–49. https://doi.org/10.1109/TASSP.1978.1163055

[49] Nicolas Sheilds. [n. d.]. Walmart Drone Delivery. Retrieved December 09, 2018 from https://www.businessinsider.com/walmart-blockchain-drone-delivery-patent-2018-9

[50] Yasser Shoukry, Paul Martin, Paulo Tabuada, and Mani Srivastava. 2013. Non-invasive Spoofing Attacks for Anti-lock Braking Systems. In *Cryptographic Hardware and Embedded Systems - CHES 2013*, Guido Bertoni and Jean-Sebastien Coron (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 55–72.

[51] Yunmok Son, Hocheol Shin, Dongkwan Kim, Youngseok Park, Juhwan Noh, Kibum Choi, Jungwoo Choi, and Yongdae Kim. 2015. Rocking Drones with Intentional Sound Noise on Gyroscopic Sensors. In *24th USENIX Security Symposium (USENIX Security 15)*. USENIX Association, Washington, D.C., 881–896. https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/son

[52] Christopher Steiner. [n. d.]. Bot-In-Time Delivery. https://www.forbes.com/forbes/2009/0316/040_bot_time_saves_nine.html#68ee53d9b942

[53] ArduPilot Development Team. [n. d.]. Building ArduPilot. https://github.com/ArduPilot/ardupilot/blob/master/BUILD.md

[54] Paparazzi Development Team. [n. d.]. Paparazzi - The Free Autopilot. https://wiki.paparazziuav.org/wiki/Main_Page

[55] Pixhawk Development Team. [n. d.]. Pixhawk AutoPilot. https://docs.px4.io/en/flight_controller/pixhawk_series.html

[56] Nils Ole Tippenhauer, Christina Pöpper, Kasper Bonne Rasmussen, and Srdjan Capkun. 2011. On the Requirements for Successful GPS Spoofing Attacks. In *Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS '11)*. ACM, New York, NY, USA, 75–86. https://doi.org/10.1145/2046707.

2046719
[57] T. Trippel, O. Weisse, W. Xu, P. Honeyman, and K. Fu. 2017. WALNUT: Waging Doubt on the Integrity of MEMS Accelerometers with Acoustic Injection Attacks. In *2017 IEEE European Symposium on Security and Privacy (EuroS P)*. 3–18. https://doi.org/10.1109/EuroSP.2017.42
[58] Zhaolin Yang, Feng Lin, and B. M. Chen. 2016. Survey of autopilot for multi-rotor unmanned aerial vehicles. In *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*. 6122–6127. https://doi.org/10.1109/IECON.2016.7793820

A RESEARCH METHODS

A.1 State Estimation Model

The Algorithm 3 shows how we derive the state estimation model (*i.e.*, A, B, C, D matrices) from the mission profile data. We collected time series data M from n missions, and we use Matlab's SI toolbox to generate the matrices [35].

Algorithm 3: Generating State Space Model

```
1  M ← Mission profile data.
2  n: number of missions.
3  t_s: sampling interval.
4  N_p: poles.
5  N_z: zeroes.
6  while i < k do
7  |  data(i) = iddata (y(i), u(i), ts);
8  end
9  A, B, C, D ← systemIdentification(data, N_p, N_z);
10 while i < n − k do
11 |  if checkModelAccuracy(A, B, C, D) then
12 |  |  break;
13 |  else
14 |  |  data = iddata (y, u, ts);
15 |  |  A, B, C, D ← SystemIdentification(data, N_p, N_z);
16 |  |  checkModelAccuracy(A, B, C, D);
17 |  end
18 end
19 Function SystemIdentification(data, N_p, N_z):
20 |  tf = tfest(data, Np, Nz);
21 |  [num, den] = tfdata(tf);
22 |  [A, B, C, D] = tf2ss(num, den);
23 return A, B, C, D
```

For the first iteration, we randomly select k mission profile data from the data set M (Line 6). The time series data from k missions is combined with `iddata` object, which consists of input and output value matrices and a fixed sampling interval t_s (Line 6 to 8). The control inputs u and control outputs y values collected in mission i are represented as vectors. At line 20, the `tfest` function identifies the optimal coefficients for the model template from the RV's mission profile data. The transformation turns a time-domain function into the frequency domain and hence substantially reduces the complexity of fitting the profile data. At line 21, function `tfdata` accesses the resultant model: *num* and *den* that encode the model in the frequency domain. At line 22, `tf2ss` function converts a discrete-time transfer function into equivalent state-space representation.

We test the accuracy of the state space model by comparing the model estimated values $(y(t), x'(t))$ with the recorded values. To improve the accuracy of the state estimation model, we perform system identification iteratively (Line 10 to 18) by adding 1 more mission profile data to the *iddata* object.

A.2 Kalman Gain

The state space matrices derived using system identification can be used to formulate a system model *sys*. The Matlab function `kalman` creates a state space model K_{ss} of the Kalman estimator given the system model *sys*.

```
[Kss,K,P] = kalman(sys,Qn,Rn,Nn)
```

K is the Kalman gain matrix, P is the error covariance matrix, Q_n, R_n, N_n are the noise covariance data.

B ATTACK ALGORITHMS

B.1 Algorithm for Artificial Delay Attack

The function *ArtificialDelay* shown in Algorithm 4will get triggered when the RV's software components trigger the malicious libraries. The duration of delays to be injected t_{attack} will be derived based on the monitoring window (t_w) used in the invariant analysis model (CI, EKF) as shown in Line 4. The algorithm will execute certain resource intensive operation (*e.g.*, infinite recursion, computationally intensive calculations etc.) to cause delays.

Algorithm 4: Artificial Delay Attack

```
1  Function ArtificialDelay():
2  |  t_Now: current system time.
3  |  t_w: monitoring window.
4  |  t_AD = t_Now + t_w;
5  |  while true do
6  |  |  if t_AD < t_Now then
7  |  |  |  ArtificialDelay();
8  |  |  else
9  |  |  |  break;
10 |  |  end
11 |  end
12 |  t_attack = Null;
13 return
```