# IMUFUZZER: Resilience-based Discovery of Signal Injection Attacks on Robotic Aerial Vehicles

Sudharssan Mohan*, Kyeongseok Yang†, Zelun Kong*, Yonghwi Kwon‡, Junghwan Rhee§, Tyler Summers*, Hongjun Choi¶, Heejo Lee†, Chung Hwan Kim*

*University of Texas at Dallas, †Korea University, ‡University of Maryland, §University of Central Oklahoma, ¶DGIST
{sudharssan.mohan, zelun.kong, tyler.summers, chungkim}@utdallas.edu, {ks8171235, heejo}@korea.ac.kr,
yongkwon@umd.edu, jrhee2@uco.edu, hongjun@dgist.ac.kr

***Abstract*—Robotic aerial vehicles (RAVs), particularly drones, are crucial in civil and military sectors. However, researchers have found that adversaries can inject noise into sensor measurements and cause physical impacts on the RAVs like crashes. Although identifying such signal injection attacks is essential to evaluate and improve the robustness of an RAV, it is challenging to discover them since their impact depends on the RAV's physical states and the search space of noise signals and physical states is vast due to its dynamic nature.**

**This paper proposes IMUFUZZER, a feedback-driven fuzzing framework, to automatically test an RAVs system and discover signal injection attacks. IMUFUZZER generates realistic noise signals for various inertial measurement unit (IMU) sensors, and monitors their impact on RAV control to detect mission failures, leveraging a high-fidelity RAV simulator. To find the physical states that attacks depend on, IMUFUZZER generates various mission paths that the RAV will fly through. We develop a novel feedback mechanism to quantify the resilience of the RAV against attacks and efficiently guide the fuzzing process to find signal injection attacks. Using IMUFUZZER, we have discovered 23 successful signal injection attacks on popular RAV control software (ArduPilot). We evaluate the correctness and effectiveness of our feedback-based sensor fuzzing and demonstrate the feasibility of the discovered attacks through physical experiments.**

## I. INTRODUCTION

Robotic vehicles (RAVs), including unmanned aerial, terrestrial, and marine vehicles, have gained significant traction in both the commercial and military sectors. These vehicles are crucial in various tasks, such as delivery, search, and rescue missions. RAVs use sensor data to continuously adjust their physical states. In particular, inertial measurement unit (IMU) sensors, such as accelerometers and gyroscopes, are essential for real-time velocity and angular rate measurements.

Accurate IMU measurements are critical for RAV navigation. However, IMU sensors are susceptible to physical noise attacks. Prior studies have demonstrated *signal injection attacks* on accelerometers [1], [2], [3] and gyroscopes [4], [1], [3], where adversaries use acoustic noise to resonate microelectromechanical system (MEMS) IMU sensors and remotely manipulate their outputs. By exploiting these noise signals, attackers can influence RAV sensor readings [2], [1] and even cause crashes [3], [4], [5], despite the presence of sensor noise filters [6] and recovery mechanisms [7]. While other sensors, such as GPS and magnetometer, are also susceptible to noise-based attacks, this work focuses on signal injection attacks against IMU sensors as they constitute the most fundamental sensing component for RAV operation.

Although prior studies have investigated the feasibility of signal injection attacks [1], [2], [3], [4], their effectiveness in causing RAV malfunctions remains inconsistent, as their impact highly depends on the *physical states* of the RAV. For instance, certain signal injection attacks may lead to crashes only during specific missions (when the RAV enters particular physical states), but not in others. Therefore, accurately identifying signal injection attacks necessitates the joint analysis of both the injected signals and corresponding physical states, a challenge that has not yet been fully addressed.

In this paper, we investigate *the exploitability of signal injection attacks* by uncovering the relationships between signal injection attacks, the characteristics of IMU sensors, and the RAV system's physical states (*e.g.*, position, velocity, acceleration, and roll/pitch/yaw attitude). This problem is particularly challenging due to the existence of unexplored *ranges of effective noise signals*, which dynamically change based on the RAV's physical states. The vast space of possible signals and physical states makes it difficult to systematically identify effective signal injection attacks. Existing fuzzing approaches for RAVs (*e.g.*, RVFuzzer [8]) primarily target invalid control parameters or static system states [9], and *do not* explore mutations of sensor inputs. Similarly, RVProber [5] focuses on mutating raw sensor values of *known attacks*, with the goal of assessing their robustness, rather than discovering new vulnerabilities.

To the best of our knowledge, no existing fuzzing framework systematically explores new signal injection attacks, each of which may be effective under different physical states. To address this gap, we present IMUFUZZER, a novel fuzzing framework designed to systematically explore a wide range of effective sensor noise patterns (*i.e.*, signal injection attacks) targeting the IMU sensors of RAV systems across various physical states. When successfully triggered, these attacks can cause severe malfunctions, such as mission-critical deviations or crashes, preventing the RAV from completing its mission.

IMUFUZZER is a feedback-guided fuzzer that automatically discovers signal injection attacks without requiring prior

knowledge of the RAV's software stack. It adopts a black-box approach, operating without any domain-specific information of the RAV's control software and does not require instrumentation or modifications to that software. To ensure safety and enable repeatability, IMUFUZZER leverages a high-fidelity RAV simulator [10] to explore the interaction between signal injection attacks, sensor noise signals, and physical states, without the risks associated with testing on real vehicles. Our framework can incorporate physical profiles of real-world MEMS IMU sensors (when available) and is designed to identify malicious signals that remain within the physically plausible bounds of those sensors. It generates candidate noise signals by mutating the amplitude and frequency of signal waveforms, guided by formal models of physical sensor behavior [1], [2], [4]. To uncover the physical states that influence the effectiveness of these signals, IMUFUZZER systematically mutates the geometric properties of mission paths, such as distances and angles between waypoints, in three-dimensional space. These mutations directly affect the RAV's maneuvers, enabling the exploration of a diverse range of physical states in which signal injection attacks may succeed.

To efficiently guide the search process, IMUFUZZER introduces a new metric called *resilience score*, which quantifies the impact of a signal injection attack on RAV control. This score serves as feedback to steer the generation of subsequent test cases, prompting the creation of mission paths and noise signals that induce greater control disruption.

Our contributions can be summarized as follows:

- We propose IMUFUZZER, a black-box fuzzing framework that systematically discovers IMU signal injection attacks by mutating flight missions to explore diverse physical states of robotic vehicles.
- We introduce a novel *resilience* metric to efficiently guide the fuzzing process and incorporate the physical characterisics of real-world MEMS IMU sensors to generate realistic and physically feasible sensor signals.
- We implement and evaluate IMUFUZZER on a widely used RAV test suite [11], using 6 real IMU sensor profiles. Our framework discovers 23 unique attacks through software-in-the-loop (SITL) simulation and validates their feasibility through physical drone experiments. We publicly release our code, experimental data, and video records to facilitate further research [12].

## II. BACKGROUND

**RAV System and IMU Sensors.** Most RAVs operate on a closed-loop control program that continuously monitors and adjusts its operation to maintain the physical states to achieve an assigned mission plan (or mission in short). The control program uses readings from various sensors to estimate the *current states* of the vehicle, including its acceleration, speed, location, and orientation in the three-dimensional space. Its goal is to align the current states with the *reference states* by controlling the actuators – for example, increasing the throttle to achieve a higher altitude. The reference states serve as intermediate goals that facilitate the transition toward achieving

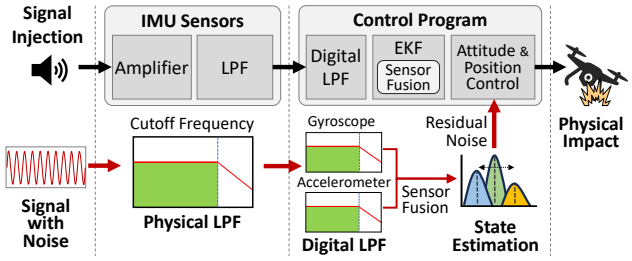



Figure 1: An RAV system under signal injection attack.

the *mission states*, which are given by the ground control station (GCS) to represent the mission. The control program runs continuously to minimize *control errors* (*i.e.*, deviations) between the current and reference states, and consequently to complete the mission.

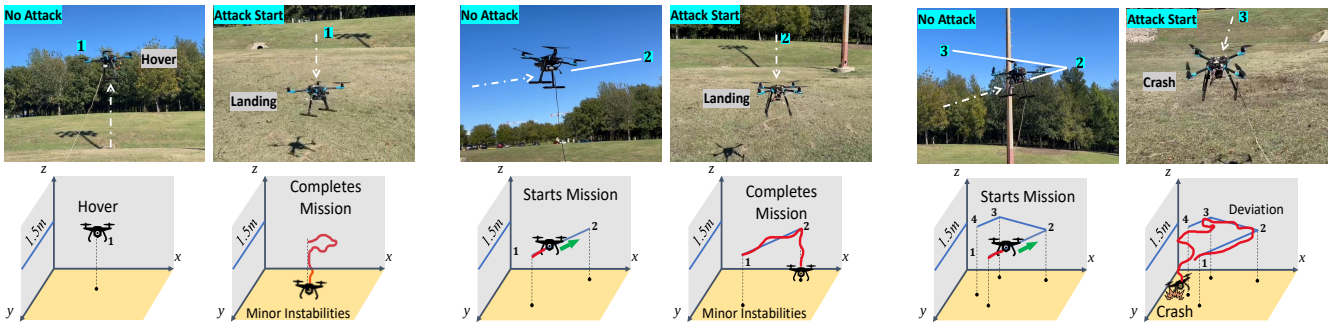
**Signal Injection Attacks.** IMU sensors on RAVs are primarily micro-electromechanical system (MEMS) sensors, which consist of electrical and moving mechanical parts. These MEMS sensors typically contain a sensing mass suspended between capacitive plates, which vibrates along different axes due to the vehicle's momentum. Vibrations at the sensing mass's resonant frequency cause it to oscillate significantly more, leading to erroneous sensor readings. Thus, an adversary can exploit these noise signals to displace the sensing mass and manipulate the sensor values that the control program reads [1], [2], [4], [3]. As illustrated in Fig. 1, such noise signals can eventually cause a malfunction of the RAV system and even crash the vehicle.

Identifying and mitigating signal injection attacks is challenging since they are data and physical state-dependent, unlike traditional software bugs that stem from faulty code. The signal injection attacks manifest as a result of the resonant characteristics in the sensor hardware coupled with the inability of the noise filters to filter out the resulting disturbances entirely. The impact is exacerbated when the controllers attempt to compensate for state errors based on corrupted sensor data.

## III. MOTIVATING CHALLENGES

In this section, we outline the key challenges that we face and our approaches to addressing them to discover the signal injection attacks while exploring diverse physical states of an RAV systematically.

- **Limitation of Existing Works.** Existing fuzzing approaches for CPS primarily mutate only sensor data [4], [1], [2], [5] or configuration/high-level commands [13], [8] without systematically exploring physical state dependencies, incorporating real-world signal profiles, or mission-level trajectory changes. There is no existing CPS fuzzer to explore and realize the relationship between injected sensor signals and the RAV's corresponding physical states. This limitation hinders the discovery of state-dependent vulnerabilities that emerge only under specific physical conditions or during dynamic transitions. Tab. I summarizes




(a) Hover mission. (b) Linear mission. (c) State-rich mission.

Figure 2: Physical experiments to show the distinct impacts of a signal injection attack on an RAV flying three different missions. The RAV's flight trajectories (red) show varying impacts of the attack.

the differences in approaches and attack discovery methods of existing works.

- **Attack Impacts and Physical States.** We highlight the challenges of discovering state-dependent attack outcomes by experimentally showing one of the attacks found by IMUFUZZER. We perform physical experiments (Fig. 2) with an RAV to illustrate how the same signal injection attack can have varying effects on the RAV depending on the physical states during the attack. We inject the same malicious acoustic noise into the RAV's gyroscope using a ultrasonic speaker and waveform generator while flying three different missions: a hover mission (Fig. 2(a)), a linear mission (Fig. 2(b)), and a state-rich mission (Fig. 2(c)). The RAV completes the first two missions with minor errors but crashes during the state-rich mission. This demonstrates our key idea, to *generate missions* that systematically explore diverse physical states. Prior works typically use fixed missions and manual testing, focusing only on the feasibility of known attacks.
- **Large and Dynamic Search Space.** A major challenge in discovering signal injection attacks lies in the vast space of physical-state-to-noise-signal combinations. Their amplitude and frequency parameters determine the nature of the acoustic signals, and their effects on IMU sensors depend heavily on the RAV's physical states. This makes the effective signal space both large and dynamically state-dependent, rendering manual or random exploration infeasible. To address this, we employ feedback-driven fuzzing, a technique not used in prior sensor attack discovery efforts (Tab. I). Instead of brute-force search, we introduce a novel feedback metric, resilience score, which quantifies the RAV's control performance during missions by monitoring physical states, enabling efficient guided exploration.

## IV. Attack Model and Assumptions

We consider an acoustic signal injection attack model, where an external adversary (armed with knowledge of the target MEMS IMU sensor's characteristics) *injects resonant-frequency acoustic signals* to compromise sensor readings. This attack does not require physical access to the RAV itself. Instead, the adversary can operate externally using off-the-shelf equipment such as an acoustic emitter, directional horn, amplifier, and function generator to emit continuous, targeted sound waves. This reflects well-established and successfully demonstrated attacks in prior works [4], [1], [3], [2].

Additionally, we assume the adversary cannot modify the vehicle's hardware, software, or internal control logic and has no access to the internal system states. The RAV is assumed to follow a known or similar flight path to those identified by IMUFUZZER, though the attack's success is not contingent on an exact trajectory. Instead, a successful signal injection attack depends on the RAV entering specific physical states (*e.g.*, combinations of position, velocity, and orientation) where the injected signal becomes effective. Importantly, these *attack windows* (*i.e.*, ranges of physical states in which the vehicle is susceptible to the injected signal) are shown to be significantly wide (§VII-D). As a result, the same malicious signal can be effective across multiple distinct missions that induce similar state transitions, increasing the practicality and generalizability of the attack.

## V. Design

IMUFUZZER is a black-box feedback-guided fuzzer designed to explore signal injection attacks successfully causing mission failure, with respect to various physical states of an RAV. We model the discovery of signal injection attacks as a feedback-driven search problem over a joint input space of missions and sensor signals. The objective of IMUFUZZER is to identify $(M^*, s^*)$ such that:

$$\min_{M,s} \ \rho(M, s)$$

Let $M \in \mathcal{M}$ denote a mission, defined as a sequence of waypoints whose geometric parameters determine the physical state transitions of the RAV. Let $s(t; A, F, \phi)$ represent a time-dependent signal injected into an IMU sensor, parameterized by amplitude $A$, frequency $F$, and optional phase $\phi$. For each execution of a mission under signal injection, we define a resilience score $\rho(M, s)$ that quantifies the control error as feedback to guide the fuzzer efficiently.

Table I: Comparison of different RAV sensor attack discovery techniques.

| Project | Attack Method | Attack Target | Attack Discovery | Physical State Exploration | Fuzzing Feedback |
|---|---|---|---|---|---|
| RVProber [5] | Signal Injection, GPS Spoofing | Accel, Gyro, GPS, ... | Stress Testing with Known Attacks | Fixed Missions | N/A |
| Son *et al.* [4] | Signal Injection | Gyro | Manual Physical Exp. | Hover Mission Only | N/A |
| Trippel *et al.* [2] | Signal Injection | Accel | Manual Physical Exp. | No (Sensor Board) | N/A |
| Tu *et al.* [1] | Signal Injection | Accel, Gyro | Manual Physical Exp. | No (Sensor Board) | N/A |
| Jeong *et al.* [3] | Signal Injection | Accel, Gyro | Semi-automated Testing | Fixed Missions | N/A |
| **IMUFUZZER** | **Signal Injection** | **Accel, Gyro** | **Feedback-guided Fuzzing** | **Mission Generation** | **Resilience Score** |

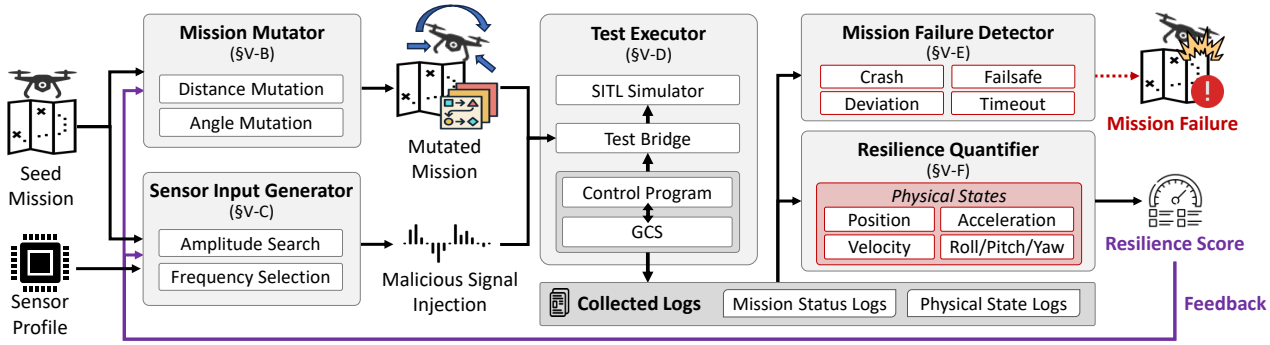



Figure 3: IMUFUZZER Architecture.

## A. Overview

Fig. 3 presents the architecture of IMUFUZZER. The fuzzer takes two key inputs: **a seed mission** and **a sensor profile**. A seed mission defines an initial flight path as a sequence of waypoints, each with its corresponding distance and angle. The sensor profile specifies physical displacement bounds of a real-world IMU sensor under acoustic injection attack. These guide the fuzzer to explore diverse physical states and generate physically plausible signal inputs.

Using these inputs, the **mission mutator** (§V-B) generates multiple variants of the mission by altering the geometric properties of the mission paths to induce varied physical states. **sensor input generator** (§V-C) generates a realistic range of noise signals modeled after studying the physical characteristics of real-world MEMS IMU sensor(s) when under acoustic injection attack and injects them into the RAV's raw sensor inputs. With the mutated missions and sensor inputs, the **test executor** (§V-D) flies the RAV in a high-fidelity SITL simulator to find signal injection attacks that cause critical mission failure. While the RAV is carrying out the mission under attack, the **mission failure detector** (§V-E) examines logs from the test executor to identify any mission failure. The **resilience quantifier** (§V-F) computes the overall resilience of the RAV against the injected signal by aggregating control errors across physical state controllers. This quantified score serves as feedback in the fuzzing loop, effectively acting as an optimization oracle that guides IMUFUZZER to generate new missions and noise signals with progressively higher potential to destabilize RAV control.

**Sensor Profile.** IMUFUZZER leverages a real-world sensor profile to constrain the amplitude range of injected signals, ensuring that discovered attacks remain physically realistic. Each profile is empirically derived for a specific MEMS IMU sensor (*e.g.*, $\pm 80 m/s^2$ for MPU-6050) and guides the fuzzer toward identifying feasible and impactful attacks within realistic displacement boundaries, thereby avoiding the generation of physically implausible signals. A sensor profile can be created through a semi-automated physical experiment, which varies acoustic inputs to determine the sensor's resonant frequency and amplitude ranges (*e.g.*, Tab. II).

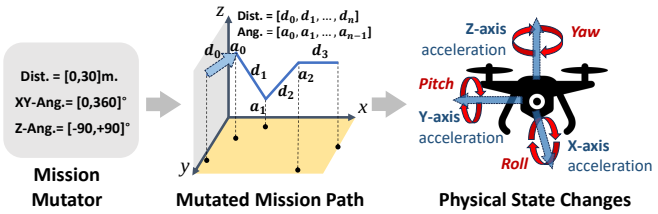



Figure 4: Mission mutation to explore diverse physical states.

## B. Mission Mutator

To explore various physical states and identify those related to signal injection attacks, IMUFUZZER creates diverse mutations of the seed mission by leveraging factors associated with RAV movements: (1) position changes in the 3D space, (2) speed and acceleration, (3) orientation (roll, pitch, and yaw), and (4) timing of physical state changes (*i.e.*, when and how often the vehicle has to turn). Fig. 4 illustrates our mission mutation. We identify the following *mission parameters* that can be mutated, allowing our approach to conduct black-box testing without altering the control program.

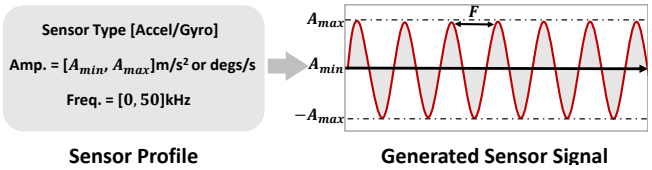



Figure 5: Sensor input generation using a sensor profile.

A mission $M$ is represented as a tuple of $n$ waypoints:

$$\mathcal{M} = \left(\{w_i\}_{i=0}^{n}, \{d_i\}_{i=0}^{n-1}, \{a_i\}_{i=0}^{n-2}\right)$$

where $w_i$ are waypoint coordinates, $d_i$ the distances between consecutive waypoints, and $a_i$ the turn angles.

Using resilience-based feedback, IMUFUZZER mutates the seed mission in three sequential phases. After each mutation phase, the variant with the lowest resilience score is selected as the new seed.

**(1) Distance Mutation:** Given the seed mission, the mission mutator generates a (pre-configured) number of mutated missions by altering the distances ($d_i$) between consecutive waypoints. The new distances are randomly selected within a user-specified range. During the distance mutation phase, the mutator keeps the angle and altitude of the waypoints unchanged.

**(2) Angle Mutation:** Given the selected mission in the distance mutation, the angles between the pairs of the mission's waypoints are mutated. The mutator then adjusts turn (XY-plane) and climb/descent (Z-axis) angles to induce roll, yaw, and pitch changes.

**(3) Mission Path Generation:** Using the set of distances and angles from the mutated mission, IMUFUZZER generates a concrete mission path in 3D waypoint coordinates (*i.e.*, GPS coordinates and altitude) by leveraging the Haversine formula [14], which is widely used in navigation. The coordinates are subsequently sent to the test executor for mission execution. The mission path generation takes the same amount of time to generate various distances and angles, regardless of the length of the mission, ensuring scalability, because we mutate all the distances and angles of the mission path as a whole instead of mutating a single segment at a time. Furthermore, our design leverages the simulator configurations to speed up the execution of each test case and increase the fuzzing efficiency.

### C. Sensor Input Generator

The sensor input generator injects noise signals into the raw sensor values generated by the high-fidelity simulator. Fig. 5 illustrates this process using a sensor profile. We make a small modification to the simulator to inject noise signals into the original sensor values, adjusting several *signal injection parameters* that we define. A selected mission from the mission mutator is used (§V-B) to search the acoustic signal space. Once the signal injection parameters are selected, IMUFUZZER performs the attack *persistently* throughout the mission without changing the parameters after the RAV takes off. Resonant noise signals impacting IMU sensors can be represented as *sine waves* propagating through the air as vibrations.

Eq. 1 models the signal injection representing the relationship between the benign and injected sensor signals.

$$\hat{s}(t) = s(t) + A * sin(2\pi F t + \phi) \quad (1)$$

A sine wave has an amplitude ($A$), which determines the magnitude of its vibration, and a frequency ($F$), which dictates the rate at which it propagates. The phase ($\phi$) of the wave determines the wave's relative position in time ($t$). The (attacked) output sensor value ($\hat{s}(t)$) is a combination of the original sensor value ($s(t)$) and the sine wave.

Since the phase only shifts the sine wave temporally without altering its shape, we primarily explore amplitude and frequency, as they have the most impact. We first focus on searching the *frequency space* and then exploring the amplitude space because we observe that the amplitude space is highly dependent on the specific frequency of the sine wave.

IMUFUZZER performs the frequency and amplitude mutation using resilience feedback to infer which combination of physical states and malicious acoustic signals ($\sum(s(t))$) is more likely to yield mission failure as follows.

**(1) Frequency Selection:** The sensor input generator employs a two-phase search over the resonant frequency range (e.g., 0–52 kHz), guided by resilience scores. In phase one, it tests frequencies at coarse intervals (e.g., 2.5 kHz) with random amplitudes, selecting the frequency yielding the lowest resilience. Phase two refines the search in a narrow window (e.g., ±5 kHz) around the selected frequency using finer steps (e.g., 100 Hz). This two-phase search strategy efficiently converges on frequencies that most significantly destabilize the RAV.

**(2) Amplitude Search:** Given the frequency selected in the previous phase, IMUFUZZER searches amplitude values within a sensor-specific physical range $[A_{min}, A_{max}]$, representing feasible displacement magnitudes. The search begins at $A_{min}$ and incrementally increases in steps (user-configurable) until $A_{max}$. This helps uncover subtle attacks that only manifest under certain physical states, where injected signals may be benign in some cases and destabilizing in others.

Our search strategy prioritizes frequency because it is the dominant factor in triggering resonance-based IMU attacks. Amplitude is explored only after frequency since its effective range is narrower and its influence is minimal without resonance. During the frequency search, we allow amplitude to vary randomly within sensor bounds to avoid biasing toward ineffective combinations, while keeping the search primarily frequency-driven. This sequential approach balances attack discovery efficiency with practical fuzzing time constraints.

### D. Test Executor

The test executor manages the control program, GCS, and the SITL simulator. It features a test bridge that facilitates communication and coordination among them.

**Test Bridge.** The test bridge seamlessly connects the control program and GCS with the simulator. Given a mutated mission path and signal injection parameters as input, it creates a mission plan and transmits the signal injection parameters to the simulator.

**Mission Execution and Logging.** The GCS issues commands to the control program to load the mission plan and initiate the mission. During this mission, the test executor records the communication between the control program and GCS to monitor mission status and physical states for resilience quantification.

**SITL Simulator.** IMUFUZZER leverages a high-fidelity SITL simulator widely used in the drone industry to conduct missions on an RAV for testing. It also provides detailed feedback on how the system responds to various inputs and conditions, which allows IMUFUZZER to analyze the potential physical implications of signal injection attacks.

### *E. Mission Failure Detector*

IMUFUZZER continuously monitors the mission status and physical states of the RAV in real-time. via communication channels and logs. Upon detecting the end of a mission, the success or failure of the mission can be determined by analyzing the collected logs. Specifically, IMUFUZZER categorizes the outcome of a mission based on the following criteria.

- **Completion:** The RAV completes the mission successfully despite signal injection, without triggering any significant errors.
- **Fail-safe:** The RAV triggers a built-in fail-safe (*e.g.*, max altitude limit or EKF error threshold) and aborts the mission by landing or returning home.
- **Crash:** If a signal injection attack successfully destabilizes RAV movements and goes undetected by the fail-safe mechanism, it may crash the vehicle.
- **Deviation:** A mission deviation (D(M)) occurs if at any time the RAV's actual position deviates from its planned trajectory beyond a user-configurable threshold.
- **Timeout:** A timeout occurs when the RAV fails to reach a new waypoint in its current mission ($M$) within a configured time window due to the signal injection attack. Formally, if $t_{\text{last}}$ is the timestamp of the last reached waypoint and $t_{\text{now}}$ is the current time, we define:

$$\mathrm{T}(M) \iff (t_{\text{now}} - t_{\text{last}}) > T_{\max}$$

The overall mission failure condition is defined as:

$$\text{Failure} = \text{Crash} \vee D(M) \vee \mathrm{T}(M)$$

### *F. Resilience Quantifier*

Injected noise signals cause cumulative errors in an RAV's position, velocity, acceleration, and orientation controllers, potentially leading to mission failures. Quantifying such control instability is crucial for guiding IMUFUZZER efficiently toward impactful signal injection attacks since not all errors would result in a crash or mission failure. Thus, we propose a resilience score ($\rho$) metric:

$$\rho = 1/mean(\{e(t_1), \dots, e(t_n)\}) \tag{2}$$

$$e(t) = \int_{t}^{t+w} mean(\{|x(s) - r(s)|, |m(s) - r(s)|\})^2 ds \tag{3}$$

Here $x(s)$, $r(s)$, and $m(s)$ is the current, reference, and mission states of any given controller, respectively. We design the function $e(t)$ to evaluate the *Integral Squared Error (ISE [15])*, a well-known metric in the field of control systems. The mission state corresponds to the physical state required to complete the mission; the reference state serves as an intermediate set point used by the control program to guide the RAV's current state toward mission completion. Errors are collected within a sliding time window $w$, capturing transient and persistent deviations. The resilience quantifier uses the collected logs of physical states to compute a resilience score after each test execution, which in turn reflects the performance of the RAV. The resilience score acts as a fitness function guiding IMUFUZZER efficiently towards inputs that significantly destabilize the RAV (§VII-C).

**Resilience-guided Fuzzing Scheme.** Alg. 1 presents the pseudo-code for the feedback-driven fuzzing process of

**Algorithm 1:** Resilience-guided fuzzing scheme.

**Input :** $M$ – Seed mission, $(A_{min}, A_{max})$ – Amplitude search range, $(F_{min}, F_{max})$ – Frequency search range, $C$ – Max iterations, $A_{step}$ – Amplitude step size.
**Output:** $m_f$ – Final mission, $F_f$ – Final frequency, $\rho$ – Resilience score.

```
foreach M ∈ get_seed(), F, A do
    fuzz(M, F, A);
procedure fuzz(M, F, A):
    /* Resilience-guided mission mutation      */
    foreach param ∈ M do
        ρ_lowest ← None;
        for I to C do
            m' ← mutate(param);
            states ← test_execute(m', F, A);
            ρ ← resilience_calc(states);
            if failure_detect(states) and ρ < ρ_lowest then
                m_f ← m'; ρ_lowest ← ρ;
            save_failure_logs(states, m', F, A);
    /* Resilience-guided frequency selection  */
    ρ_lowest ← None;
    for I to C do
        F' ← search(F_min, F_max);
        states ← test_execute(m_f, F', A);
        ρ ← resilience_calc(states);
        if failure_detect(states) and ρ < ρ_lowest then
            F_f ← F'; ρ_lowest ← ρ;
        save_failure_logs(states, m_f, F', A);
    /* Amplitude search w/ selected frequency */
    for A' ∈ range(A_min, A_max, A_step) do
        states ← test_execute(m_f, F_f, A');
        if failure_detect(states) then
            save_failure_logs(states, m_f, F_f, A');
```

IMUFUZZER. At line 7, given the mission path of a seed mission ($M$), it first mutates the mission path for each parameter ($M_p$) by randomly choosing the distances and angles. For each mutation, the test executor executes the mission and performs a signal injection attack on the RAV (line 8). The resilience score ($\rho$) is calculated (line 9) after each test execution and used as the optimization feedback. On detecting a mission failure (line 10), IMUFUZZER prioritizes inputs that yield the lowest resilience (line 11), treating them as high-impact seeds to further mutate in the next search stages. The logs of the failed missions are recorded (line 12) irrespective of the resilience score. Similarly, the resilience score is used during the two-phase frequency selection phase.

## VI. IMPLEMENTATION

IMUFUZZER is implemented in approximately 2.6K lines of Python and 470 lines of C++ code. We modified the SITL simulator for ArduPilot [10] for noise signal injection, and utilized MAVLink [16] and its open-source APIs [17] to control and monitor the communication between control program and GCS [18], [19], [20]. The in-flight logs of ArduPilot are used to monitor physical state changes and compute resilience score. Such logs are commonly available in open-source RAV control systems [21], [22].

**Extensibility.** IMUFUZZER can also be implemented using other SITL simulators [23], [24] with a little effort because it only requires a small modification (less than 470 lines) to update the IMU sensor values when they are generated by the simulator. Moreover, it does not require any modification to the control program to test since it only relies on the external communication between the program and GCS, and logs that RAV control programs commonly produce.

## VII. EVALUATION

In this section, we present a series of experiments designed to address the following research questions (RQs).

- **RQ1:** How effective is IMUFUZZER in discovering signal injection attacks? (§VII-A)
- **RQ2:** What are the characteristics of the discovered signal injection attacks? (§VII-B)
- **RQ3:** How effective is the proposed *resilience score* metric in guiding the fuzzing process? (§VII-C)
- **RQ4:** How feasible are the discovered attacks when executed in real-world mission scenarios? (§VII-D)

**Experimental Setup.** We evaluate IMUFUZZER on a desktop machine running Ubuntu 22.04.1 LTS, powered by a 16-core Intel Core i7-10700 CPU, 48 GB DRAM, and a GeForce RTX 2060 GPU. Using IMUFUZZER, we test ArduCopter v4.5.0, the latest version of ArduPilot for quadcopter RAVs at the time of this work. We use the sensor profiles of 6 MEMS IMU sensors (3 accelerometers and 3 gyroscopes) to discover signal injection attacks, as shown in Tab. II. For physical experiments, we construct an attack device using an SDG1032X waveform generator [25], a K3118 digital amplifier [26], and an ultrasonic speaker to emit sine wave-based acoustic noise. The target RAV is a custom-built quadcopter equipped with a Pixhawk4 controller board [27] running the same ArduCopter version. The board integrates two IMUs: an ICM-20689 as the primary and an ICM-20602 as the secondary sensor. We release the source code of IMUFUZZER, the ArduPilot version used in our experiments, and video records of the physical experiments in our code repository [12].

Table II: Sensor profiles of the IMU sensors used in our experiments. The sensor value displacement values are in $m/s^2$ for the accelerometers and $deg/s$ for the gyroscopes.

| | Sensor | Amp. (V) | Freq. (KHz) | Displacement | |
|---|---|---|---|---|---|
| | | | | Min | Max |
| Accel. | AXDL345 | 10.3 | 3.8 | -134.0 | 53.2 |
| | MPU-6050 | 9.5 | 5.3 | -80.0 | 80.0 |
| | MPU-9250 | 4.2 | 20.6 | -22.2 | -4.3 |
| Gyro. | L3G4200D | 8 | 8.1 | -105.1 | 105.3 |
| | MPU-6050 | 3.2 | 26.86 | -270.4 | 260.3 |
| | MPU-9250 | 3.2 | 27.3 | -232.6 | 247.1 |

### A. Signal Injection Attack Discovery

IMUFUZZER has discovered a total of 23 successful signal injection attacks on ArduPilot, targeting accelerometer and gyroscope sensors. Tab. III summarizes the details of the discovered attacks. We do not include the cases that triggered the fail-safe landing of the RAV in this number since the fuzzer has discovered more than 500 such cases. Each attack we discovered uses a unique spectrum of acoustic signals (frequency and amplitude range) and physical states generated by the mission. We manually identified the impacts of the attacks on the physical states in the controller to confirm that they do not overlap. Attacks with similar signal spectrums and physical states are counted as one.

In our experiment, the fuzzer took approximately one hour to discover the first signal injection attack and eventually discovered 113 mission failure cases in 24 hours caused by the 23 attacks. Tab. III shows the following key results.

**Signal Injection.** We present the range of amplitude (**Amp.**) and frequency (**Freq.**) of the noise signals discovered by IMUFUZZER for a given physical sensor profile.

**Mission.** For each signal injection attack discovered, we perform the attacks on a hovering mission (**Hover**) and a linear mission (**Linear**) to evaluate the significance of exploring different physical states (§VII-B).

**Attack Impact.** Our results show that two signal injection attacks using the identical amplitude and frequency (A3 and A7) can create distinct attack impacts when their mission paths differ, clearly showing *the dependency between the signal injection attack and physical states* induced by the missions.

The discovered attacks demonstrate IMUFUZZER's effectiveness in systematically uncovering state-dependent vulnerabilities that would be difficult to detect through manual testing with fixed missions.

### B. Signal Injection Attack Analysis

Unlike traditional software vulnerabilities, signal injection attacks manifest due to multiple factors, not explicitly shown

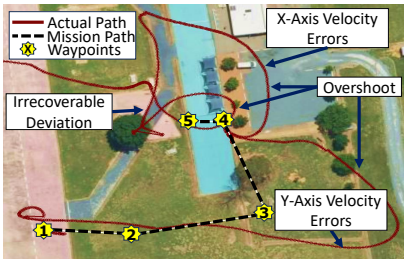
Table III: Signal injection attacks discovered by IMUFUZZER on accelerometers (top half: A* cases) and gyroscopes (bottom half: G* cases). Sensor Profile indicates the physical sensor configuration used. Max Control Error measures the maximum control error between the current and reference state values of the Initially Corrupted Physical State. C and D in the Final Impact column indicate a crash and severe deviation from the mission path, respectively. We did not have a timeout case detected although IMUFUZZER can detect it.

| # | Signal Injection | | | Mission | | Attack Impact | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Amp. | Freq. (KHz) | Sensor Profile | Hover | Linear | Most Affected IMU Axis | Initially Corrupted Physical State | Max Control Error | Other Impacted Physical States | Final Impact | Time till Final Impact (min) |
| A1 | 19.75-29.75 | 20 | AXDL345 | ✗ | ✓ | Accel Y | Vel Y | 20.9 m/s | Pos Y, Yaw | C | 1.3 |
| A2 | 19.25-29.50 | 16 | AXDL345 | ✗ | ✗ | Accel Y | Vel Y | 32.59 m/s | Pos Y, Roll | C | 10.0 |
| A3 | 9.25-11.00 | 40 | MPU-6050 | ✗ | ✓ | Accel Y | Vel Y | 31.38 m/s | Pos Y, Yaw | C | 2.1 |
| A4 | 13.25-19.25 | 28 | MPU-6050 | ✗ | ✗ | Accel X | Vel X | 28.07 m/s | Pos X, Yaw | C | 3.4 |
| A5 | 18.5-23.50 | 11.2 | MPU-6050 | ✗ | ✗ | Accel Z | Vel Z | 2.27 m/s | Pos Z, Roll | C | 0.1 |
| A6 | 20.25-28.25 | 16 | AXDL345 | ✗ | ✗ | Accel X | Vel X | 22.49 m/s | Pos X, Pitch | C | 4.5 |
| A7 | 10.00-26.75 | 40 | MPU-6050 | ✓ | ✓ | Accel X | Vel X | 43.4 m/s | Pos X, Roll | C | 0.1 |
| A8 | 17.50-19.50 | 24 | MPU-6050 | ✓ | ✓ | Accel X | Vel X | 25.54 m/s | Pos X, Yaw | C | 3.5 |
| A9 | 4.25-8.50 | 52 | AXDL345 | ✓ | ✓ | Accel Y | Vel Y | 27.33 m/s | Pos Y, Yaw | C | 2.2 |
| A10 | 15.00-22.50 | 28 | AXDL345 | ✓ | ✓ | Accel Y | Vel Y | 31.83 m/s | Pos Y, Yaw | C | 2.9 |
| A11 | 2.75-8.00 | 44 | AXDL345 | ✗ | ✗ | Accel Y | Vel Y | 35.85 m/s | Pos Y, Yaw | D | 4.0 |
| A12 | 14.00-29.50 | 47.2 | AXDL345 | ✗ | ✓ | Accel X | Vel X | 19.34 m/s | Pos X, Yaw | D | 1.5 |
| G1 | 13.50-30.00 | 35.4 | MPU-6050 | ✓ | ✓ | Gyro X | Roll | 63.26 degs | Pos Y, Vel Y | C | 0.5 |
| G2 | 7.25-30.00 | 48 | L3G4200D | ✗ | ✗ | Gyro X | Roll | 14.28 degs | Pos Z, Vel Z | C | 0.9 |
| G3 | 9.00-29.00 | 48 | MPU-6050 | ✗ | ✗ | Gyro X | Roll | 13.01 degs | Pos Y, Vel Z | C | 0.8 |
| G4 | 7.25-30.00 | 40 | L3G4200D | ✗ | ✗ | Gyro X | Roll | 13.54 degs | Pos Y, Vel Y | C | 1.3 |
| G5 | 20.25-29.75 | 23.8 | L3G4200D | ✗ | ✗ | Gyro Y | Pitch | 42.04 degs | Pos Y, Vel Y | C | 1.7 |
| G6 | 11.00-17.00 | 6.4 | MPU-9250 | ✓ | ✓ | Gyro Y | Pitch | 25.59 degs | Pos X, Vel X | C | 0.2 |
| G7 | 27.25-29.75 | 15.8 | MPU-9250 | ✗ | ✗ | Gyro Y | Pitch | 73.94 degs | Pos Y, Vel Y | C | 2.3 |
| G8 | 22.50-28.75 | 47.8 | MPU-6050 | ✗ | ✗ | Gyro X | Roll | 56.43 degs | Pos Y, Vel Y | C | 3.8 |
| G9 | 17.50-29.75 | 23.9 | MPU-9250 | ✗ | ✗ | Gyro X | Roll | 50.72 degs | Pos X, Vel X | C | 1.2 |
| G10 | 22.25-30.00 | 8.3 | MPU-6050 | ✓ | ✓ | Gyro X | Roll | 35.77 degs | Pos Y, Vel Y | D | 5.3 |
| G11 | 17.00-18.50 | 39.8 | MPU-9250 | ✗ | ✗ | Gyro Y | Pitch | 50.71 degs | Pos Y, Vel Y | D | 3.8 |

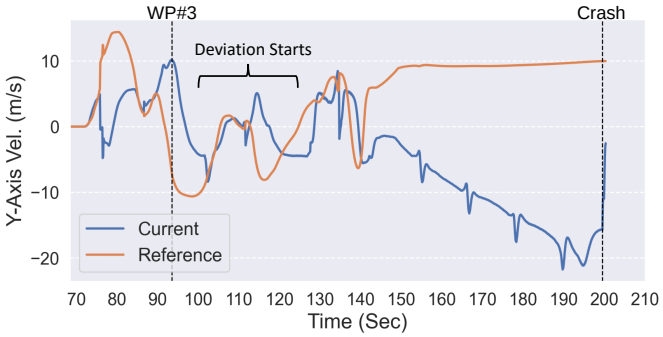
on the control program. First, the inherent resonant characteristics of the MEMS IMU sensor hardware enable acoustic injection attacks. Second, various noise filters in the control program (*e.g.*, LPFs) and state estimators like EKF fail to effectively remove/suppress the resonant noises. These noises propagate through the controllers, leading to physical state errors that keep increasing over time. We have manually analyzed the signal injection attacks and identified the key factors that caused the mission failure, examining the impact of the attacks on RAV control.

**Accelerometer Attacks (A1-12).** These attacks introduce faults in linear acceleration measurements, corrupting the RAV's internal velocity estimation. The severity and nature of the disruption depend on the mission's trajectory. For missions with sustained translation along a particular axis (e.g., *A4, A6–A8, A12 for X-axis; A1, A3, A11 for Y-axis*), injected perturbations on the corresponding IMU axis cause compounding velocity errors. This results in overshooting or undershooting at the turns, ultimately leading to deviation from the trajectory or collision. For example, in case A3 the RAV significantly overshoots the turns, deviating from its intended mission path, and crash (Fig. 6). Manual analysis identified the Y-axis velocity controller as the root cause, where residual noise distorted accelerometer readings, causing errors in velocity estimation (Fig. 6(b)). Eventually, the accumulated control errors force the RAV to deviate irrecoverably, crashing approximately 752 meters from the intended landing point. Altitude-sensitive missions (e.g., A5) are especially fragile since Z-axis perturbations can trigger premature thrust corrections leading to crashes.

**Gyroscope Attacks (G1-11).** Displacements in the attacked gyroscope values affect the estimation of the RAV's angular positions across all three axes, similar to accelerometer attacks. However, corrupted yaw (Z-axis) is not a primary cause of mission failure in our experiments, as the control program periodically corrects yaw using GPS, rendering these errors less impactful. We instead observe differences in how roll and pitch corruption affect the RAV. In missions requiring significant roll changes (*G1–G4, G8–G10*), injected displacements caused the RAV to roll in the wrong direction or overshoot the intended roll angle. For example, case G9 illustrates the gradual accumulation of control errors, which eventually




(a) Mission and flight paths.




(b) Control errors in Y-axis velocity.

Figure 6: Attack analysis of case A3.

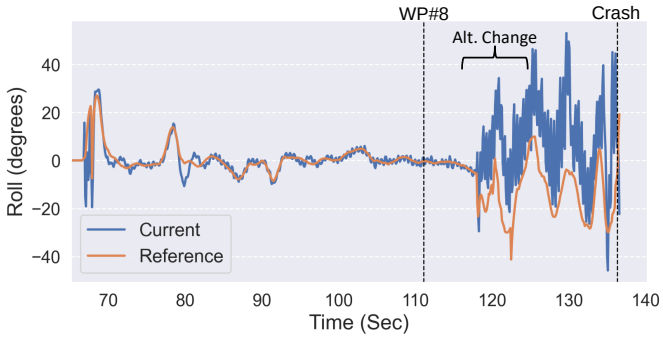



Figure 7: Control errors in roll angle of case G9.

results in a crash. Upon analysis, we discovered the root cause to be the roll controller, which accumulated significant errors upon reaching a waypoint that required a significant change in altitude. At the start of the mission, the initial digression of the roll error, as shown in Fig. 7, is only up to ≈20 degrees. However, these control errors *spiked after the altitude change at waypoint #8 (WP #8)*, leading to an error of up to 50.72 degrees on the roll controller.

In some cases (*G4, G8*), the RAV attempts to recover but lacks sufficient thrust to stabilize its orientation, leading to crashes. Pitch (Y-axis angle) directly affects velocity control: pitching forward increases forward thrust, while pitching backward reduces it. In missions with pitch state errors (*G5-G7, G11*), this results in impaired velocity regulation and deviation in flight direction.

In summary, our analysis revealed that signal injection attacks manifest differently depending on mission trajectories and physical states. By examining their effects on controller states (*e.g.*, velocity, roll, and pitch), we showed that IMUFuzzer uncovers diverse failure modes that propagate from corrupted IMU measurements to mission-critical crashes or deviations.

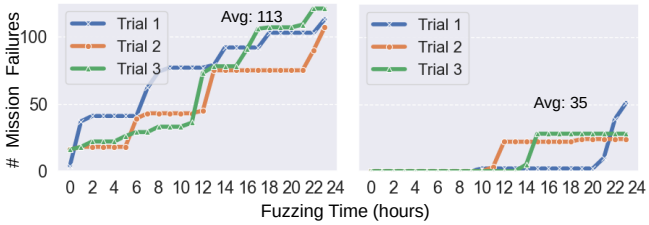



Figure 8: Number of mission failures detected while fuzzing ArduPilot for 24 hours using the resilience score as feedback (left) and without any feedback (right). Each configuration is repeated three times to provide the average numbers.

### C. Effectiveness of Resilience Feedback

We evaluated IMUFuzzer's resilience-guided fuzzing by running: (1) feedback-guided mutation of IMUFuzzer using the resilience score and (2) random mutation without any feedback; in three 24-hour runs each. Missions took 1.5–3 minutes, depending on their length.

As demonstrated in Fig. 8, while only 35 mission failures were recorded with the random mutation, IMUFuzzer (with the resilience-guided mutation) identified an average of 113 mission failures, showing an over 2.2x increase in the number of discovered mission failures. Note that each of the mission failures is unique, as different variations of the injected signals and mission paths are used. Thus, no mission failure was recorded more than once. The results validate the effectiveness of our resilience score as a feedback metric for guiding the fuzzer toward impactful signal–mission combinations. By favoring signal injections and mission parameters with lower resilience scores, the fuzzer increases the likelihood of uncovering signal injection attacks.

### D. Feasibility of Discovered Attacks

We conduct a feasibility analysis and physical experiments to demonstrate signal injection attacks that IMUFuzzer discovers are plausible in the real world.

**Attack Window Analysis.** To evaluate whether the discovered attacks are exploitable under realistic flight conditions, we analyze how different mission-level parameters affect their success. We show that the physical states resulting from the missions IMUFuzzer generated, have significantly large attack windows for an attacker to perform the signal injection attacks. Tab. IV presents our evaluation of the feasibility of exploiting the cases discovered by IMUFuzzer. We analyze post-mission logs for each of the signal injection attacks in Tab. III and identify mission attributes that can enable a signal injection attack. We begin by identifying the specific segment of each mission where the RAV's physical state changes

Table IV: Enumeration of mission attributes and their attack window for successful signal injection attacks. Attributes with $*$ can have arbitrary values while those with $\diamond$ must be within the attack window.

| # | Mission Attributes | | | Attack Window |
|---|---|---|---|---|
| | Alt. | Ang. | Dist. | |
| A1 | $\diamond$ | $*$ | $*$ | Any decrease in altitude (*e.g.*, landing) |
| A2 | $\diamond$ | $*$ | $*$ | Any angle of decent $>$ 10 degs |
| A3 | $*$ | $\diamond$ | $*$ | Any turn at angle $>$ 20 degs |
| A4 | $*$ | $\diamond$ | $*$ | Any turn at angle $>$ 60 degs |
| A5 | $*$ | $\diamond$ | $*$ | Any turn at angle $>$ 80 degs |
| A6 | $\diamond$ | $\diamond$ | $*$ | Any turn at angle $>$ 100 degs and any decrease in altitude |
| A7 | $*$ | $*$ | $*$ | Any physical state change |
| A8 | $*$ | $*$ | $*$ | Any physical state change |
| A9 | $*$ | $*$ | $*$ | Any physical state change |
| A10 | $*$ | $*$ | $*$ | Any physical state change |
| A11 | $*$ | $\diamond$ | $*$ | Any turn at angle $>$ 10 degs |
| A12 | $*$ | $*$ | $\diamond$ | Fly distance $>$ 70 meters |
| G1 | $*$ | $*$ | $*$ | Any physical state change |
| G2 | $*$ | $\diamond$ | $*$ | Any turn at angle $>$ 15 degs |
| G3 | $\diamond$ | $*$ | $*$ | Any angle of decent $>$ 40 degs |
| G4 | $\diamond$ | $*$ | $*$ | Any angle of decent $>$ 80 degs |
| G5 | $\diamond$ | $*$ | $*$ | Any angle of decent $>$ 10 degs |
| G6 | $*$ | $*$ | $*$ | Any physical state change |
| G7 | $\diamond$ | $*$ | $*$ | Any angle of decent $>$ 25 degs |
| G8 | $\diamond$ | $\diamond$ | $*$ | Turn at angle $>$ 40 degs and any angle of ascent $>$ 60 degs |
| G9 | $\diamond$ | $*$ | $*$ | Any angle of decent $>$ 45 degs |
| G10 | $\diamond$ | $*$ | $*$ | Any angle of decent $>$ 50 degs |
| G11 | $\diamond$ | $\diamond$ | $*$ | Any turn at angle $>$ 45 degs and any angle of ascent $>$ 45 degs |

significantly at the time of failure. We then perturb individual attributes by incrementally modifying the mission attributes to determine whether each attribute independently enables the attack to succeed.

We describe the three mission attributes as follows:

**(1) Angle of Altitude Change (Alt.):** The angle of altitude refers to the vertical angle that a drone has to ascend/descend to change its altitude, including taking off or landing. We found that 8 cases can be attributed solely to altitude changes. Six (G3-5, G7, G9-10) of the cases are targeting the gyroscope, as unregulated orientation can deter the RAV from maintaining a stable position to climb or drop in altitude.

**(2) Angle of Turns (Ang.):** Five of the attacks succeed by simply adjusting a waypoint turn (XY-axis angle), some as little as $\approx$10 degrees (A11, G2) up to 80 degrees (A5). Such changes fall well within normal mission variability, making them hard to detect or mitigate.

**(3) Distance of Flight (Dist.):** Distance refers to the straight-line flight between waypoints. In general, distance has a limited influence compared to altitude or turn angles. However, one case (A12) demonstrates that sustained flight ($\approx$70 meters) under persistent signal injection attack can accumulate sufficient deviation to cause mission failure.

In summary, we observe that six attacks require no specific mission condition. In contrast, 14 require at most one mission attribute to fall within a particular range for successful exploitation, indicating a high degree of feasibility for real-world attacks. Notably, these *attack windows* are easily realizable in practice, as many can be triggered by natural variations in the vehicle's physical state or are effective across a broad range of mission trajectories.

**Physical Experiments.** We conduct physical experiments to demonstrate signal injection attacks that IMUFuzzer discovers are feasible targeting a real drone. Fig. 2 presents snapshots of the experiments. In these experiments, we perform a signal injection attack by directing acoustic signals of 27.4 kHz at the ICM-20689 gyroscope, persistently after the RAV takes off to perform the three different missions. During the flights, we visually validate the attack's impact on the RAV affecting its stability and angular velocity estimates, and confirm that a crash was observed under the state-rich mission, matching our result from SITL-based testing using the sensor profile. These results substantiate that IMUFuzzer's findings extend beyond simulation and reliably manifest under real-world conditions, with broad attack windows across altitude changes, turns, and distances.

## VIII. Related Work

**Sensor Attacks on RVs.** While researchers have studied various RAV sensor attacks comprehensively [28], [29], [30], [31], Son *et al*. [4] first proposed using resonant frequencies of MEMS gyroscopes to launch denial-of-service attacks on an RAV. Trippel *et al*. [2] introduce an attack on MEMS accelerometers leveraging amplitude and phase modulations built on this knowledge. Tu *et al*. [1] leverage modulation of acoustic signals further to achieve fine control of the attacks. RVProber [5] evaluates known attacks by mutating spoofed values but does not discover new vulnerabilities. In contrast, IMUFuzzer aims to discover new sensor attacks through feedback-guided fuzzing.

**Sensor Attack Detection and Recovery.** There have been several efforts to counter sensor attacks, particularly on RAVs. Choi *et al*. [32], [33] employ system identification [34] and control invariants to detect and recover from sensor spoofing attacks. RVFuzzer [8] targets input validation bugs in command parameters via control-guided fuzzing but does not explore sensor signal mutations. It does not aim to discover sensor attacks or mutate sensor inputs. UnRocker [3] analyzes the sampling jitter as one of the causes of acoustic injection attacks on IMU sensors and develops a recovery mechanism. This research establishes a semi-automated testbed to collect benign and malicious sensor values using software- and hardware-in-the-loop simulators.

## IX. Discussion and Future Work

**Fidelity of Simulation.** Although we mitigate discrepancies between simulated and real-world testing through realistic

sensor profiles and validation in physical drone experiments, some gap may still remain. Nonetheless, SITL simulators are the industry's de facto standard for evaluating RAV systems prior to field deployment and have been used extensively in research to enhance RAV safety and security [5], [3], [8], [35], [36]. Furthermore, our physical experiments, which assess the fuzzer using real-world IMU sensor properties and validate the feasibility of the discovered attacks (§VII), confirm the simulator's high fidelity. Consequently, high-fidelity SITL simulation provides the safest and most practical approach for IMUFUZZER, given the inherent safety and financial risks of real-world testing. Future work could investigate hybrid testing setups that integrate SITL with hardware-in-the-loop or partial physical validation, further reducing the fidelity gap while preserving safety and cost efficiency.

**Mission and Sensor Generalization.** Our fuzzing strategy can be applied to any seed mission, including standardized test suites. While this paper focuses on IMU sensors, the framework is extendable to other vulnerable sensors (e.g., barometer, magnetometer) with minor modifications to the input generator.

**Feasibility of Physical Attacks.** Identifying the resonant frequencies of a sensor is often non-trivial in practice. In our experiments, creating accurate sensor profiles required an extensive frequency sweep. Consequently, an attacker may need to perform manual testing with commercially available equipment on the target IMU before they can reliably exploit the discovered attacks.

**Environmental Factors.** IMUFUZZER focuses on mutating missions to explore physical states. By considering environmental factors, such as wind or temperature, future work can explore more dynamic physical states and weather-dependent attack scenarios to find new signal injection attacks. Additionally, future research could develop adaptive fuzzing strategies to prioritize environment–mission combinations most likely to reveal such vulnerabilities.

## X. CONCLUSION

This paper presents IMUFUZZER, a black-box feedback-driven fuzzing framework for RAVs, aimed at discovering stateful signal injection attacks that can be exploited by an adversary targeting IMU sensors. IMUFUZZER identifies signal injection attacks that cause mission failures only in specific physical states by (1) automatically mutating seed missions in a high-fidelity simulator, (2) injecting malicious sensor input while executing the mutated missions, and (3) guiding the fuzzer based on the resilience score of the RAV against the attacks as feedback. Using IMUFUZZER, we have discovered 23 signal injection attacks using 6 real-world IMU sensor profiles, that resulted in critical physical impacts. We further analyze these attacks and evaluate the correctness and effectiveness of the key components of our framework. Our experiments show that IMUFUZZER can effectively and efficiently discover new signal injection attacks on RAVs using resilience as fuzzing feedback.

## ACKNOWLEDGMENT

We thank the anonymous reviewers for their constructive feedback. This work was supported in part by the National Science Foundation (under grants 2443487, 2426653, and 2427783), University of Texas at Dallas Office of Research through the NFRS and SPIRe programs, Agency for Defense Development grant funded by the Korean government (U23056TF), and IITP grant funded by the Korea MSIT (No. RS-2022-II220277 Development of SBOM Technologies for Securing Software Supply Chains, No. RS-2024-00440780 Development of Automated SBOM and VEX Verification Technologies). Any opinions, findings, recommendations, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

## REFERENCES

[1] Y. Tu, Z. Lin, I. Lee, and X. Hei, "Injected and Delivered: Fabricating Implicit Control over Actuation Systems by Spoofing Inertial Sensors," in *Proceedings of the 27th USENIX Security Symposium (USENIX Security 2018)*, 2018.

[2] T. Trippel, O. Weisse, W. Xu, P. Honeyman, and K. Fu, "WALNUT: Waging Doubt on the Integrity of MEMS Accelerometers with Acoustic Injection Attacks," in *Proceedings of the 2017 IEEE European Symposium on Security and Privacy (EuroS&P 2017)*, 2017.

[3] J. Jeong, D. Kim, J. Jang, J. Noh, C. Song, and Y. Kim, "Un-Rocking Drones: Foundations of Acoustic Injection Attacks and Recovery Thereof," in *Proceedings of the 30th Network and Distributed System Security Symposium (NDSS 2023)*, 2023.

[4] 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 *Proceedings of the 24th USENIX Security Symposium (USENIX Security 2015)*, Washington, DC, Aug. 2015.

[5] H. Kim, R. Bandyopadhyay, M. Ozmen, Z. Celik, A. Bianchi, Y. Kim, and D. Xu, "A Systematic Study of Physical Sensor Attack Hardness," in *Proceedings of the 45th IEEE Symposium on Security and Privacy (S&P 2024)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2024, pp. 142–142.

[6] "Low Pass Filters," https://en.wikipedia.org/wiki/Low-pass_filter, 2025.

[7] "ArduPilot Failsafe Mechanisms," https://ardupilot.org/copter/docs/failsafe-landing-page.html, 2025.

[8] T. Kim, C. H. Kim, J. Rhee, F. Fei, Z. Tu, G. Walkup, X. Zhang, X. Deng, and D. Xu, "RVFuzzer: Finding Input Validation Bugs in Robotic Vehicles through Control-Guided Testing," in *Proceedings of the 28th USENIX Security Symposium (USENIX Security 2019)*, Santa Clara, CA, Jul. 2019.

[9] J. Lee, E. Viganò, O. Cornejo, F. Pastore, and L. Briand, "Fuzzing for cps mutation testing," in *Proceedings of the 38th IEEE/ACM International Conference on Automated Software Engineering (ASE 2023)*. IEEE, 2023, pp. 1377–1389.

[10] "ArduPilot SITL Simulator," https://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html, 2025.

[11] "ArduPilot," https://ardupilot.org/, 2025.

[12] "Public GitLab Repository for IMUFUZZER," https://gitlab.com/s3lab-code/public/imufuzzer, 2025.

[13] H. Kim, M. O. Ozmen, A. Bianchi, Z. B. Celik, and D. Xu, "PGFUZZ: Policy-Guided Fuzzing for Robotic Vehicles," in *Proceedings of the 28th Network and Distributed System Security Symposium (NDSS 2021)*, Virtual Event, Feb. 2021.

[14] C. Veness, "Haversine Formula," https://www.movable-type.co.uk/scripts/latlong.html, 2025.

[15] D. Graham and R. C. Lathrop, "The synthesis of "optimum" transient response: Criteria and standard forms," *Transactions of the American Institute of Electrical Engineers, Part II: Applications and Industry*, vol. 72, no. 5, pp. 273–288, 1953.

[16] "MAVLink," http://mavlink.io/, 2025.

[17] "Pymavlink," https://github.com/ArduPilot/pymavlink, 2025.

[18] "QGroundControl," http://qgroundcontrol.com/, 2025.

[19] "Mission Planner," https://ardupilot.org/planner/, 2025.

[20] "ArduPilot MAVProxy," https://ardupilot.org/mavproxy/, 2025.

[21] "ArduPilot Flight Logs," https://ardupilot.org/copter/docs/logmessages.html, 2025.
[22] "PX4 Flight Logs," https://clover.coex.tech/en/flight_logs.html, 2025.
[23] "Gazebo," https://gazebosim.org, 2025.
[24] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles," in *Proceedings of the 11th Conference on Field and Service Robotics (FSR 2017)*, 2017, pp. 621–635.
[25] "SDG1032X Function Generator," https://siglentna.com/product/sdg1032x/, 2025.
[26] "Kinter K3118 Amplifier," https://tinyurl.com/yrr4237u, 2025.
[27] "Pixhawk4," https://holybro.com/products/pixhawk-4, 2025.
[28] Y. Xu, X. Han, G. Deng, G. Li, Y. Liu, J. Li, and T. Zhang, "SoK: Rethinking Sensor Spoofing Attacks against Robotic Vehicles from a Systematic View," in *Proceedings of the 8th IEEE European Symposium on Security and Privacy (EuroS&P 2023)*, 2023.
[29] B. Nassi, R. Bitton, R. Masuoka, A. Shabtai, and Y. Elovici, "SoK: Security and Privacy in the Age of Commercial Drones," in *Proceedings of the 2021 IEEE Symposium on Security and Privacy (S&P 2021)*, 2021, pp. 1434–1451.
[30] S. Nashimoto, D. Suzuki, T. Sugawara, and K. Sakiyama, "Sensor CONFusion: Defeating Kalman Filter in Signal Injection Attack," in *Proceedings of the 2018 Asia Conference on Computer and Communications Security (ASIACCS 2018)*, 2018, pp. 511–524.
[31] C. Yan, H. Shin, C. Bolton, W. Xu, Y. Kim, and K. Fu, "SoK: A Minimalist Approach to Formalizing Analog Sensor Security," in *Proceedings of the 2020 IEEE Symposium on Security and Privacy (S&P 2020)*, 2020.
[32] H. Choi, S. Kate, Y. Aafer, X. Zhang, and D. Xu, "Software-based Realtime Recovery from Sensor Attacks on Robotic Vehicles," in *Proceedings of the 23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, 2020, pp. 349–364.
[33] H. Choi, W.-C. Lee, Y. Aafer, F. Fei, Z. Tu, X. Zhang, D. Xu, and X. Deng, "Detecting Attacks against Robotic Vehicles: A Control Invariant Approach," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS 2018)*, 2018, pp. 801–816.
[34] "System Identification," https://www.mathworks.com/products/sysid.html, 2025.
[35] Y. Sun, C. M. Poskitt, J. Sun, Y. Chen, and Z. Yang, "Lawbreaker: An approach for specifying traffic laws and fuzzing autonomous vehicles," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering (ASE 2022)*, 2022, pp. 1–12.
[36] S. Feng, Y. Ye, Q. Shi, Z. Cheng, X. Xu, S. Cheng, H. Choi, and X. Zhang, "Rocas: Root cause analysis of autonomous driving accidents via cyber-physical co-mutation," in *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, 2024, pp. 1620–1632.