A Systematic Study of Physical Sensor Attack Hardness

Hyungsub Kim
Purdue University
kim2956@purdue.edu

Rwitam Bandyopadhyay
Purdue University
bandyopr@purdue.edu

Muslum Ozgur Ozmen
Purdue University
mozmen@purdue.edu

Z. Berkay Celik
Purdue University
zcelik@purdue.edu

Antonio Bianchi
Purdue University
antoniob@purdue.edu

Yongdae Kim
KAIST
yongdaek@kaist.ac.kr

Dongyan Xu
Purdue University
dxu@purdue.edu

2024 IEEE Symposium on Security and Privacy (SP) | 979-8-3503-3130-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/SP54263.2024.00143

***Abstract*—Physical sensor attacks against robotic vehicles (RV) have become a serious concern due to their prevalence and potential physical threat. However, RV software developers often do not deploy appropriate countermeasures. This hesitance stems from their belief that attackers face substantial challenges when conducting sensor attacks, e.g., nullifying sensor redundancy in hardware and circumventing sensor filters in software. Yet, we discover that attackers can overcome the challenges by fulfilling specific prerequisites and finely tuning attack parameters. The misconceptions that the developers have arisen from a lack of study regarding the level of difficulty attackers face in successfully achieving their attack goals, which we call "attack hardness".**

In this paper, we examine the hardness of 12 well-known sensor attacks. We first identify the prerequisites required to conduct the attacks successfully. We then quantify the hardness of each attack as how frequent the prerequisites enabling a specific attack are in the real world. To automate this analysis, we introduce RVPROBER, an attack prerequisite analysis framework. RVPROBER discovered that the 12 sensor attacks require, on average, 4.4 prerequisites, highlighting that previous literature has often missed important details required to perform these attacks. By satisfying the identified prerequisites and tuning attack parameters, we increased the number of successful attacks from 6 to 11. Moreover, our analysis showed that an average of 57.08% of actual RV users are vulnerable to sensor attacks. Finally, starting from the identified prerequisites, we analyzed the reasons behind the success of each attack and found previously-unknown root causes, such as design flaws in the RV software's fail-safe logic.

1. Introduction

Attacks affecting robotic vehicles (RVs) have become a serious concern in recent years due to their prevalence and their potentially-severe consequences (ranging from destroying physical infrastructure to causing human injuries). Many of these attacks fall into the category of "physical sensor attacks", which focus on tampering with the mechanisms an RV uses to perceive its operating environment [22], [42], [43], [82], [85], [86], [88], [89], [93], [97]. For instance, GPS spoofing [83], [86], [97] aims to deceive an RV's GPS receiver by sending fake GPS signals, potentially causing it to lose position control and crash, resulting in severe damage.

Unfortunately, RV control software remains vulnerable to physical sensor attacks. In fact, a preliminary analysis we performed on the most popular open-source RV software (ArduPilot [4], PX4 [66], and Paparazzi [64]), discovered that RV software developers do not deploy any countermeasures intended to prevent this kind of attacks.

Additionally, throughout our communication with RV software developers (detailed in Section 2), we observed a lack of motivation to defeat sensor attacks. In fact, they believe that these attacks are not practical since they require attackers to overcome hard challenges, such as nullifying sensor redundancy in hardware and circumventing sensor filters in software. Thus, they hold the view that it is hard for the attackers to replicate the attacks in real world scenarios.

Indeed, attempting to replicate these attacks is challenging, and in a preliminary study we also failed to replicate six out of 12 well-known sensor attacks. However, as we will show, a more in-depth analysis of these attacks reveals that the attackers can use these attacks in real-world scenarios by fulfilling specific attack prerequisites and finely tuning attack parameters. Particularly, our investigation reveals that successful sensor attacks require prerequisites in hardware, software, and environmental configuration categories. Additionally, we find that sensor attacks do not function on a "plug-and-play" basis; instead, they demand meticulous tuning of attack parameters, such as the intensity and duration of spoofed signals, to achieve attack goals.

Fundamentally, the misunderstanding that the developers have arises from a lack of "attack hardness" study. Attack hardness refers to a quantification of the difficulty involved in executing an attack. Specifically, for a given sensor attack, it quantifies the attack hardness as the likelihood that these prerequisites are met among a representative sample of real-world RV usage scenarios.

Identifying prerequisites and attack hardness enables developers to successfully replicate the attacks, identify the RV configurations vulnerable to such attacks, and understand the attack impact and prevalence. In turn, these capabilities allow

DOI 10.1109/SP54263.2024.00143

developers to develop countermeasures (i.e., code patches and/or hardware modifications) and test them, prioritizing fixes for the most impactful attack. However, current research literature [22], [42], [43], [82], [83] does not provide a complete list of required prerequisites, and there currently exists no automated tool that identifies such prerequisites and leverages them to quantify the attack hardness.

In this paper, motivated by the lack of effective approaches to understanding the hardness of attacks, we study the hardness of 12 well-known physical sensor attacks. In particular, we first identify the prerequisites required to conduct the attacks successfully. We then quantify the hardness of each attack as how frequent the prerequisites enabling a specific attack are in the real world.

To automate this process, we develop RVPROBER, an attack prerequisite analysis framework. To simulate realistic sensor attacks while considering various attack parameters, RVPROBER takes, as input, sensor values compromised under the attack. Then, it mutates the sensor values based on four different attack parameters (`intensity`, `duration`, `start time`, and `attack algorithm`). To address the high-dimensional "search spaces" involving the attack prerequisites, RVPROBER reduces the spaces via dedicated static and dynamic analyses.

Using RVPROBER, we discover that the 12 physical sensor attacks require, on average, 4.4 different types of prerequisites, many of them not reported in the research papers presenting them. We share the identified prerequisites with the authors of research papers proposing the 12 physical sensor attacks [22], [42], [43], [82], [83]. Thus, we confirm that our findings are generally in line with the authors' observations. Additionally, by ensuring the identified prerequisites and attack parameters are met, we increase the number of successful attacks from 6 to 11. To quantify the real-world hardness of conducting the attacks, we verify whether the required prerequisites are satisfied by the conditions and the configurations of RVs operated in real-world scenarios. To do so, we collect a total of 30,059 real RV logs from actual users of two popular RV control software (ArduPilot [4] and PX4 [66]). As a result, we discover that physical sensor attacks are possible, on average, in 57.08% of real-world RV configurations.

Finally, we use our ability to successfully replicate attacks to study in-depth their root causes. This study reveals previously-unknown root causes of successful attacks. In particular, we found four of these attacks are possible due to design flaws in the RV control software's fail-safe logic.

Overall, our findings shed light on the feasibility (or unfeasibility) of physical sensor attacks against RVs, and they will help current and future RV control software developers to better understand these attacks, allowing them to introduce, when needed, effective countermeasures against them.

In summary, we make the following contributions:

- **Discovery of Attack Difficulty.** Our research highlights the difficulty of conducting and studying sensor attacks due to the prerequisites required for successfully conducting the attacks.
- **Attack Hardness Analysis Tool.** RVPROBER conducts realistic sensor attacks using simulations. Each simulation tests distinct prerequisites. Based on the prerequisites found by RVPROBER, we increase the number of successful attacks from 6 to 11. The identified prerequisites not only help developers understand the impact and prevalence of these attacks, but also provide the insights needed to devise and test countermeasures.
- **Evaluation with Real-world Data.** We analyze how frequently RV configurations used by actual users satisfy the required prerequisites by analyzing a total of 30,059 RV logs. Our analysis shows that an average of 57.08% of actual RV users are vulnerable to the attacks.
- **Root Cause Analysis.** By analyzing their prerequisites, we identify the root causes enabling the successful attacks. This analysis reveals that many attacks are possible due to previously-unknown design flaws in RV software's fail-safe algorithms.

We make RVPROBER and experimental data publicly available (https://github.com/purseclab/RVProber).

Ethical Considerations and Responsible Disclosure. In the analysis of public RV logs of actual users, we analyze the publicly available RV logs and do not collect any personally identifiable information (PII). Additionally, we responsibly disclosed the physical sensor attacks and the discovered design flaws to the corresponding RV software developers.

2. Background

Sensor Fusion. Each RV control software can leverage a different sensor fusion algorithm. Here, we explain sensor fusion commonly used in popular open-source RV control software (ArduPilot [4], PX4 [66], and Paparazzi [64]). Sensor fusion aims to improve the accuracy and reliability of estimating an RV's physical states (e.g., position). It comprises three primary components: sensors, sensor filters, and sensor fusion algorithms (e.g., EKFs), as shown in Figure 1. Most RVs are equipped with redundant homogeneous and heterogeneous sensors to address the randomness of sensor readings (e.g., disturbances and failures). For example, an RV is equipped with three redundant gyroscopes, three accelerometers, and two magnetometers to measure the RV's attitude (roll, pitch, and yaw angles), angular velocity, and position (❶ in Figure 1). The two GNSS receivers can also measure the RV's attitude (yaw angles) [35] and are not the only sensors that measure the RV's positions. In fact, RV control software leverages all sensors (except for the barometer and rangefinder) to determine the RV's latitude and longitude coordinates. RV control software next inputs the measured sensor values into the sensor filters, e.g., low-pass and harmonic notch filters [11], [76] (❷). Lastly, RV control software simultaneously feeds the filtered sensor values to multiple extended Kalman filters (EKFs) (❸).

The number of EKFs is equal to the number of redundant IMU sensors (three gyroscopes and accelerometers in Figure 1). Each EKF takes, as input, a different instance of the

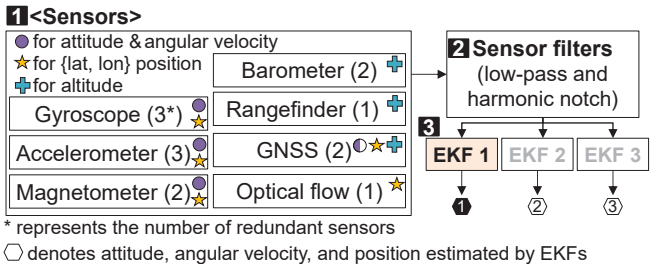


Figure 1: Illustration of different types of sensors (❶), sensor filters (❷), and sensor fusion processes on RVs (❸).

TABLE 1: Illustration of the EKF switching [23]

	EKF-1	EKF-2	EKF-3
Gyroscope (Gyro)	Gyro 1	Gyro 2	Gyro 3
Accelerometer (Acc)	Acc 1	Acc 2	Acc 3
Magnetometer (Mag)	Mag 1	Mag 2	Mag 1
Barometer (Bar)	Bar 1	Bar 2	Bar 1
Rangefinder (Ran)	Ran 1	Ran 1	Ran 1
GNSS	GNSS 1	GNSS 2	GNSS 1
Optical flow (Opt)	Opt 1	Opt 1	Opt 1

sensor, as shown in Table 1. For example, EKF-2 uses the second instance of a gyroscope, accelerometer, magnetometer, barometer, and GNSS, and the first instance of a rangefinder and optical flow sensors. Each EKF then estimates the RV's current physical states separately. RV software uses only one of the EKFs' outputs rather than combining them.

EKF Switching. This sensor fusion design allows excluding a sensor that returns erroneous measurements. Particularly, the RV control software calculates each EKF's error score by considering differences between the EKF's estimation and the actual sensor measurement. It then switches the primary EKF to the secondary EKF (EKF-1 to EKF-2) if the primary EKF shows the poorest error score among EKFs, i.e., the RV leverages the second instance of the gyroscope sensor rather than the first and third ones [23].

Attacks. Physical sensor attacks remotely inject/jam signals to disrupt sensors attached to RVs, as shown in Table 2. Throughout the paper, we consistently use the term "attacks" to refer to these physical sensor attacks.

GNSS receivers[1] are vulnerable to spoofing and jamming attacks because (*i*) the strength of legitimate GNSS signals is extremely weak (-125 dBm to -130 dBm in open space) and (*ii*) GNSS receivers do not leverage authentication methods. Attackers deceive GNSS receivers by either spoofing fake signals that are stronger than legitimate ones or by jamming legitimate signals [83], [86], [97]. Such attacks may lead to unstable position control and a physical crash into an obstacle. To address natural disturbances in outdoor spaces, GNSS receivers parse and track multiple types of satellite signals instead of relying solely on one type (e.g., GPS). Further, the GNSS receivers detect drastic signal level changes (e.g., noise level [38], [41]) and return warning messages, e.g., u-blox series raise the warnings in UBX-NAV-STATUS [90], [91]. Researchers have proposed a defense method that

1. GNSS includes any constellation systems such as GPS, Galileo, and GLONASS. In contrast, GPS denotes only the USA's positioning system.

TABLE 2: Physical sensor attacks and goals of an adversary

Attack (S): Spoofing (J): Jamming	Target sensor	Attack's goals		
		Mission failure	Unstable attitude/position	Physical crash
(S) GNSS signals [83]	GNSS	✓	✓	✓
(J) GNSS signals [83]		✓	✗	✗
(S) Acoustic noises [43], [85], [88], [89], [93]	Gyroscope Accelerometer	✓	✓	✓
(S) Magnetic fields [82]	Magnetometer	✓	✓	✓
(S) Magnetic fields [42]	SPI/I2C buses	✓	✓	✓
(S) Images on floors [22]	Optical-flow	✓	✓	✓

leverages angle-of-arrival measured by multi-GNSS receivers or antenna-array [49]. Yet, this method is rarely adopted due to the required additional hardware. While Galileo supports authenticated navigation messages (Galileo's OSNMA [28]), Motallebighomi et al. show that attackers can still spoof the GNSS receivers without modifying these authenticated messages [56].

Researchers have demonstrated that acoustic noises can produce resonance inside gyroscope and accelerometer sensors (i.e., IMU) [43], [85], [88], [89], [93]. This can lead to unstable attitude control and, eventually, a crash on the ground. In response to this attack, researchers have proposed sensor filter algorithms to be implemented either within or outside the sensors [43], [94]. Additionally, fiber-optic gyroscopes have emerged as a promising alternative due to their inherent immunity to resonance-based attacks [19], [48]. Yet, all of these defense methods are not currently adopted by the developers due to the required hardware modifications, processing overhead, and/or increased costs.

Electromagnetic interference (EMI) can disturb either magnetometers [82] or data on SPI and I2C buses [42], i.e., corrupting all sensors on an RV. Such attacks lead to physical crashes due to the corrupted sensor readings. Unfortunately, RV developers have implemented EMI filters (e.g., L-C, Pi, T filters) [44] in the hardware level for only power lines (i.e., battery) [30]. Thus, sensors and SPI/I2C buses are still vulnerable to EMI attacks. To tackle natural magnetic anomalies, some RV developers attach EMI shielding (e.g., Mu-metal) to RVs [25]. Yet, as Jang et al. discussed [42], complete shielding of flight controller boards is not feasible due to the potential for heat buildup, which can degrade the performance of circuit components.

Lastly, attackers can inject images on floors via laser beams [22], causing an RV's optical flow sensor to calculate fake motions. This attack results in unstable position control and potentially causes a crash into an obstacle. To address this attack, researchers have developed algorithms to filter out injected images [22], [29]. Despite the availability of defense algorithms against this attack, RV developers have yet to incorporate these methods into their software. This reluctance stems from the inherent limitations of existing defense algorithms, which assume that attackers can only manipulate a portion of the field of view. This limitation makes these defenses vulnerable to attacks that inject spoofed images into the entire field of view.

Reasons for Not Using Countermeasures. We could not find discussions regarding the attacks except for GNSS jamming in GitHub repositories and developer community websites [6],

TABLE 3: 12 attacks (six different types of attacks with two different levels of adversary capabilities)

Attack (S): Spoofing (J): Jamming	Adversary Capabilities	
	Attacks demonstrated in the papers	Worst-case attacks
(S) GNSS signals [83]	GPS signals	GNSS signals
(J) GNSS signals [83]	Intermittently jam	Fully jam
(S) Acoustic noises [43], [85], [88], [89], [93]	One IMU	All IMUs
(S) Magnetic fields [82]	One mag sensor	All mag sensors
(S) Magnetic fields [42]	Partial corruption	Full corruption
(S) Images on floors [22]	Partial field of view	Entire field of view

[68]. Thus, we made responsible disclosure, and in this regard, we reported the 10 well-known attacks (except for two GNSS jamming attacks, as the developers already discussed) to the developers of ArduPilot [4], PX4 [66], and Paparazzi [64].

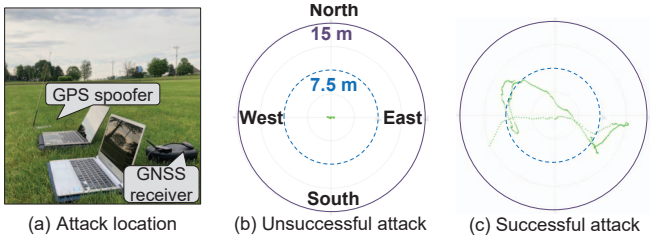
We received two distinct types of replies from the developers. Firstly, they believe that the attacks can be countered by different hardware and software configurations (e.g., the latest flight control boards and finely tuned sensor filters). Thus, they hold the belief that the attacks do not pose genuine threats. Secondly, the developers believe that addressing the attacks falls within the purview of hardware designers rather than RV software developers, as these attacks start at the sensor hardware. Yet, we will detail that the beliefs held by the developers are misconceptions in Section 9.

3. Threat Model and Scope

Our analysis focuses on six different types of remote attacks on RVs that disrupt an RV's sensors and result in a mission failure, unstable attitude/position control, or physical crash, as shown in Table 2. Within our scope, we include attacks directed toward RVs that have been documented and reported in the research literature. We assume attackers cannot inject messages between the RV and the ground control station (GCS). This is because RVs can use network packet-level authentication and encryption according to their communication protocol and hardware specifications [54], [55], [81]. Thus, we exclude remotely exploiting bugs that corrupt sensor values stored in the RV's memory.

Adversary's Capabilities. The majority of research literature [22], [42], [43], [82], [83] leverages two distinct levels of an adversary's capabilities: (*i*) attacks demonstrated in research papers and (*ii*) worst-case attacks. Thus, we also consider these two levels of an adversary's capabilities. Table 3 shows 12 attacks classified by these two capabilities.

The term "demonstrated attacks" denotes attacks that have been substantiated by conducting real-world experiments in the research literature. For instance, acoustic noise attacks demonstrated in the papers [43], [85], [88], [89], [93] inject sound at a specific frequency and compromise one of the IMUs on an RV. To compromise all IMUs, attackers need to inject sound at multiple frequencies because each IMU may have a different resonance frequency. Yet, the acoustic signals at multiple frequencies suffer from sound wave interference (i.e., phase shifts and inversion of the signal polarity [18], [87]); thus, the demonstrated attacks do not simultaneously disturb multiple IMUs.



(a) Attack location (b) Unsuccessful attack (c) Successful attack

Figure 2: (a) The location where we performed GPS spoofing. The green line represents how much the location deviates from the correct position (in the center of the circle).

On the contrary, the term "worst-case attacks" denotes the most powerful adversary's capabilities, which are not verified by conducting real-world experiments due to their hardware and software limitations. We assume that the worst-case attacks overcome the challenges of conducting attacks. Thus, such attacks can simultaneously and completely compromise all of the homogeneous sensors. For instance, the worst-case attacks inject sound at multiple frequencies at the same time and succeed in disturbing all IMUs of an RV.

Adversary's Knowledge. We consider two distinct adversary knowledge models: black-box and white-box attacks. Firstly, the black-box attacks lack knowledge about a target RV's software, hardware, and environmental configurations. Secondly, the white-box attacks have such knowledge about the target RV before they conduct attacks. Thus, the white-box attacks can dynamically select an optimal attack from the 12 available attacks and their associated parameters, thereby increasing the likelihood of achieving the attack goals.

4. Motivating Example

We demonstrate the importance of understanding attack hardness by showcasing GPS spoofing that aims to control RVs. To conduct this attack, we follow the attack steps in [97] for RVs in an outdoor environment (See Appendix B for details). This attack only requires affordable commodity hardware, while open-source software is readily available to execute the required attack steps. To address safety and ethical concerns, we ensure that there were no man-made structures, vehicles, and people within 30 meters around the experiment location, as shown in Figure 2-(a), and we used 10 dBm low-power signals (so that the signal can reach, at most, targets within a 15-meter radius).

We fail to fully observe the reported effects of the GPS spoofing attack. As shown in Figure 2-(b), the location perceived by the RV does not deviate from the real RV's location while performing the attack.

Reasons for Failure. With further investigation, we realized there were two main reasons why the attack failed. In other words, this attack requires two prerequisites. First, the attack is only possible under an RV's specific "sensor configuration". Specifically, this attack is only possible against RVs using a GPS receiver instead of a GNSS receiver. Unlike GPS receivers, GNSS receivers use data from multiple satellite constellations (e.g., GPS, Galileo, GLONASS, BeiDou, and SBAS). For this reason, the attempted attack, which only

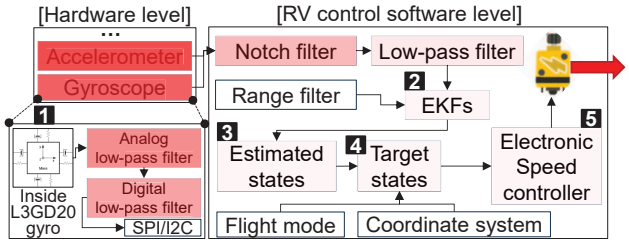


Figure 3: A simplified attack-step model illustrates the propagation of acoustic noise injection attacks within RVs.

affects the signal coming from the GPS satellite constellation, is not possible[2]. Second, the RV control software leverages sensor fusion algorithms, as explained in Section 2. Even though the GNSS receiver incorrectly measures its positions (i.e., measuring false velocity values), the RV's IMUs still correctly estimate the RV's current velocity.

Attack Based on the Identified Attack Prerequisites. Taking into account the discovered prerequisites, we change our experiment setup as follows: (1) we replace the u-blox M8N receiver with the u-blox M6 receiver, because the u-blox M6 receiver parses only GPS signals; and (2) we change PX4's configuration parameters (`EKF2_AID_MASK`, `CAL_GYRO_PRIO`, and `CAL_ACC_PRIO`) to decrease the weight given to data coming from IMU sensors and increase the weight of GPS data. After these modifications, we successfully deceive the positions estimated by the RV, as shown in Figure 2-(c).

Quantified Real-world Hardness. We can use the knowledge about the required prerequisites to estimate how frequent they are to be true in real-world scenarios. To do so, we collect 30,059 real RV logs from actual users of two popular RV software [4], [66]. We discover that in 18,271 out of the 30,059 RV logs (60.8%), these two prerequisites are true.

As we will show in Section 9, RVPROBER helps us identify the required prerequisites for attacks. In turn, the identified prerequisites to understand how likely an attack is (consulting real-world RV logs, as explained) and conducting the attack help developers test their software and patch it.

5. Attack Prerequisites

We first introduce our methodology to identify attack prerequisites. Then, we provide an in-depth explanation of the prerequisites that are required for successful attacks.

Identification of Attack Prerequisites. To discern components of RV hardware and software that amplify or mitigate the impacts of the 12 considered attacks, we leverage a single "attack-step model" using hierarchical design modeling (HdM) [31], as shown in Figure 3. Each component is recursively decomposed into subsystems (Figure 3-❶). With this model, we abstract the paths through which an attack propagates across both cyber and physical components and its interactions with the environment.

The attack-step model exhibits generality across the most popular RV hardware and software platforms. Specifically, identifying hardware components within the attack-step model involves manually analyzing the most widely used open-standard flight control boards (Pixhawk series [65]). Likewise, determining software components in the attack-step model entails manually examining the documentation and source code of the most popular open-source RV software (ArduPilot, PX4, and Paparazzi). We then extract only the common hardware and software components from the flight control boards and RV software packages.

To illustrate, Figure 3 shows a simplified attack-step model to understand the propagation of acoustic noise injection attacks. A series of sensor filters and sensor fusions mitigate deviated sensor measurements (❶-❷), but they still allow RV software to incorrectly estimate physical states of RVs, e.g., roll, pitch, and yaw angles (❸). Thus, RV software targets incorrect physical states (❹). In turn, the RVs lose attitude control due to the incorrectly set throttles of actuators (❺).

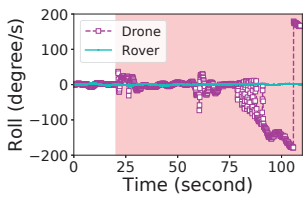
We identify seven different types of components within the attack-step model as prerequisites: RV types, sensor configurations, RV control software, software versions, flight modes, configuration parameters, and environmental conditions.

We note that these prerequisites partially overlap due to their hierarchical interdependence (e.g., RV control software implies a possible set of software versions). Yet, we separate prerequisites based on whether they can independently make attacks succeed or fail. For instance, each RV control software package configures hardware (e.g., sensors) in a peculiar way, regardless of its software versions. In parallel, the software version influences the attack success. Thus, RV control software and its software version independently changes the attack's success.

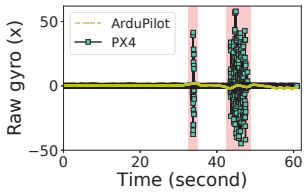
❶ **RV Types.** Attacks may require a specific RV type for two main reasons. First, signals at a high frequency (e.g., GPS L1 band is 1575.42 MHz) can only be effectively propagated through a specific type of medium, like air. Yet, such signals are unlikely to penetrate other types of mediums, like water [84]. Thus, some attacks, such as injecting fake GPS signals, cannot affect certain RV types (e.g., unmanned underwater robots [16]). Second, each RV type leverages a different coordinate system when the RV measures its attitude. This difference makes attacks fail to cause negative impacts. In particular, drones in the air and robots underwater measure their attitudes in a 3-D coordinate system. On the contrary, rovers on the ground are operated in a 2-D coordinate system, i.e., they do not use roll and pitch angles. Thus, rovers heavily depend on position sensors (e.g., GPS/GNSS receiver and optical flow sensor) but do not mainly rely on IMUs. Such a difference in the coordinate system makes the rover immune to disturbances in the IMUs. Indeed, as shown in Figure 4a, the drone crashes on the ground under an acoustic noise injection attack (the purple dashed line in Figure 4a). Yet, the rover does not face negative impacts (the cyan straight line in Figure 4a) under the same attack.

❷ **Sensor Configurations.** Attacks may require an RV that is equipped with a target sensor but is not equipped

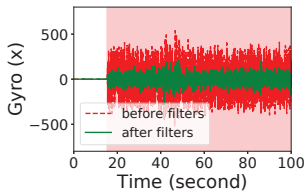
2. Attackers can first jam legitimate GNSS signals and then spoof fake GPS signals. Yet, we note that we conduct GPS spoofing without the help of jamming according to steps reported in [97]



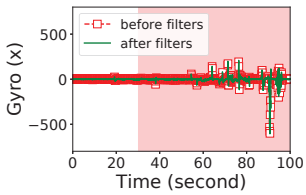
(a) EKF's estimations. The spoofed gyroscope returns up to 515.67 degree error per sec.



(b) Raw gyroscope values. PX4 is vulnerable to the attack due to disabled low-pass filters.

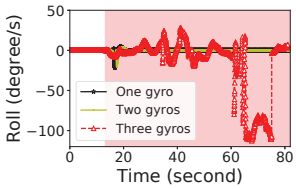


(c) Sensor filtering in ArduPilot. *We compromised all IMUs*. The spoofed gyroscope returned up to 538.15 degree error per sec.

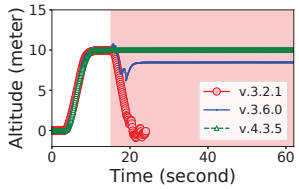


(d) Sensor filtering in PX4. *We compromised one IMU*. The spoofed gyroscope returned up to −601.1 degree error per sec.

Figure 4: Impacts of acoustic noises in ArduPilot *v.4.3.5* and PX4 *v.1.13.0*. The red area denotes the time under attack. Only the PX4 drone physically crashes on the ground.



(a) EKF's estimations according to the number of compromised gyroscope sensors.



(b) Injected electromagnetic field's impacts according to different ArduPilot versions.

Figure 5: Impacts of (a) acoustic noise injection attacks and (b) EMI on magnetometers.

with a secondary sensor. This is because the RV can still correctly measure sensor values from the secondary sensor under attack. Indeed, compromising one or two of the three gyroscopes in an RV cannot cause significant physical impacts (black and yellow straight lines in Figure 5a). The RV shows 20 degrees of error at 18 seconds. Yet, the RV immediately recovers the stable roll angles because the RV's EKF switching logic (as explained in Section 2) stops using the compromised gyroscopes and starts using the intact one.

To cause significant impacts, attackers need to spoof the primary and secondary sensors simultaneously. Yet, these multiple attacks require the adversaries to address signal interference issues [18], [87]. For instance, attackers may need to inject acoustic signals at multiple frequencies because (*i*) an RV is normally equipped with two/three different types of IMUs [65], and (*ii*) each IMU sensor only can be compromised at a unique frequency [85], [88], [89], [93]. Yet, the acoustic signals at multiple frequencies can experience phase shifts and inversion of the signal polarity [18], [87].

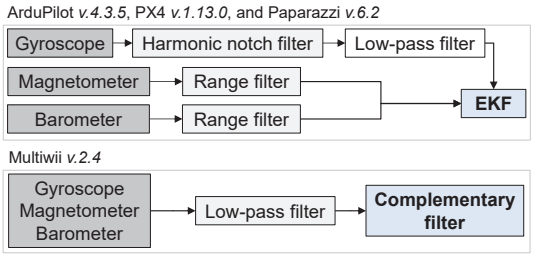
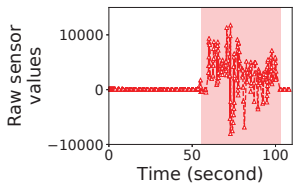
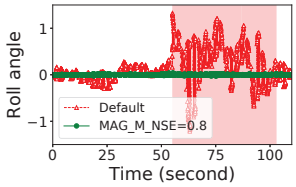


Figure 6: Illustration of sensor filters in the most recent versions of each RV control software.



(a) Raw magnetometer values.



(b) Roll estimated by an EKF.

Figure 7: Impacts of EMI on magnetometers in ArduPilot.

❸ **RV Control Software.** The effects of attacks completely vary depending on the RV control software for two main reasons. First, each RV software configures sensor parameters differently (e.g., sensor filters). Thus, the sensor measurements are also different, even under the same attack. PX4 [66] disables low-pass filters inside IMUs (Listing 1 in Appendix). On the contrary, ArduPilot and Paparazzi leverage low-pass filters inside IMUs (Listing 2 and Listing 3 in Appendix). Thus, as shown in Figure 4b, impacts of acoustic injection attacks are different according to RV software, although we conduct the same attack. Second, each RV software implements its sensor filtering algorithms differently, as shown in Figure 6. Thus, the impacts of attacks are also different. Compared to the complementary filters in MultiWii *v.2.4* [58], EKFs show better filtering performance at high levels of disturbance [27], [61]. Thus, MultiWii is more vulnerable to attacks than others. Interestingly, ArduPilot, PX4, and Paparazzi leverage the same types of filtering algorithms (i.e., harmonic notch, low-pass, and range filters). Yet, the developers of each RV software implement the filtering algorithms differently.

For instance, dynamic harmonic notch filters [11], [76] automatically determine the primary noise frequency and adjust the notch's center frequency to filter out the noise. The harmonic notch filters in PX4 and Paparazzi can find and attenuate the maximum nine peak frequencies [76]. On the contrary, ArduPilot's harmonic notch filters track and attenuate the maximum 72 peak frequencies [11]. Thus, ArduPilot more effectively filters out spoofed sensor values compared to other RV control software. Indeed, as shown in Figure 4d, PX4's filters fail to filter out compromised gyroscope values, although we compromise only one IMU. Thus, the RV crashes on the ground at around 90 seconds (green line in Figure 4d). On the contrary, ArduPilot's filters stably filter out compromised gyroscope values, although we compromise all IMUs (green line in Figure 4c).

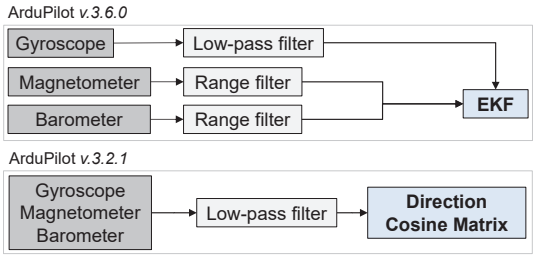


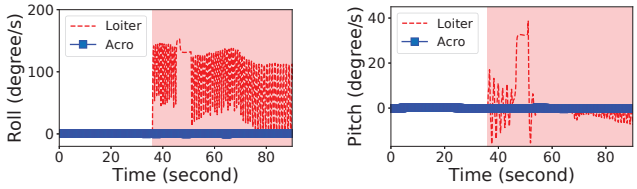
Figure 8: Sensor filters in different versions of ArduPilot.

❹ **Software Versions.** The attacks require specific software versions for two main reasons. First, RV developers have improved sensor filters while updating software versions. Second, they have implemented fail-safe logic [7], [62], [71] in recent software versions.

As of this writing, the latest versions of RV control software (e.g., ArduPilot *v.4.3.5*) are equipped with multiple levels of sensor filtering and improved sensor fusion algorithms, as shown in Figure 6. Such facts make it hard for an RV to retain compromised sensor values. For instance, attackers can inject electromagnetic fields to disturb an RV's magnetometers [82], as shown in Figure 7a. Yet, ArduPilot's range filter and EKFs filter out the spoofed sensor values, and the attack causes only around two degrees per second errors in roll angles (red line in Figure 7b). On the contrary, as shown in Figure 8, old versions of RV software (ArduPilot *v.3.6.0* and *v.3.2.1*) leverage a single level of sensor filtering or a less accurate sensor fusion algorithm (e.g., direction cosine matrix). Such facts allow attackers to easily spoof/jam sensors. Indeed, as shown in Figure 5b, ArduPilot *v.4.3.5* does not show any negative symptoms under the attack [82]. On the contrary, ArduPilot *v.3.6.0* incorrectly decreases its altitude, although the RV must maintain the current altitude. The RV, which uses ArduPilot *v.3.2.1*, crashes on the ground due to its poor filtering algorithms [14] (Figure 8).

Moreover, RV control software is equipped with fail-safe logic [7], [62], [71] designed to recover RVs from unexpected sensor disturbances. For instance, GPS glitch fail-safe logic [34] in ArduPilot discontinues using the GNSS data and switches to rely on positions predicted by IMUs when the disparity between positions estimated by GNSS and IMUs surpasses the threshold. Thus, the fail-safe logic can mitigate attacks. Yet, old versions of RV software, unfortunately, do not have such fail-safe logic (e.g., ArduPilot *v.3.2.1*). In particular, the fail-safe logic can (*i*) stop using the primary sensor and start using the secondary one or (*ii*) change the RV's flight mode into a manual mode, requiring the user to manually operate the RV. Thus, the RV continues its mission through the secondary sensor or the user's manual inputs.

❺ **Flight Modes.** Attacks require specific flight modes because an RV leverages different sensors or even stops using sensors according to its current flight mode. For instance, an RV automatically operates during `Loiter` mode [50], which uses all types of sensors attached to the RV. Conversely, `Acro` mode [2] operates based on a user's remote controller inputs rather than sensor readings. Thus, as shown by the blue lines in Figure 9, attacks (e.g., EMI injection to corrupt all different types of sensor measurements [42]) are hard to cause negative impacts when the RV is in `Acro` mode.



(a) Roll estimated by an EKF. (b) Pitch estimated by an EKF.

Figure 9: Impacts of EMI attack on SPI/I2C buses [42] according to flight modes in ArduPilot *v.4.3.5*.

❻ **Configuration Parameters.** The attacks require specific configurations related to the sensor filtering and sensor fusion parameters for two main reasons. First, the configuration parameters determine the cutoff frequencies in the sensor filters in Figure 6 and Figure 8. The spoofed sensor values can be filtered out depending on the cutoff frequencies [11], [76]. Second, the configuration parameters determine each sensor's weight for the sensor fusion algorithms (i.e., EKFs). It is hard for an attack to cause negative impacts if attackers spoof/jam a sensor that an RV uses with a low weight. For example, as shown in Figure 7b, the injected electromagnetic field [82] causes around two degrees per second errors in roll angles (red line in Figure 7b). Yet, when we change ArduPilot's `MAG_M_NSE` configuration parameter [51] to make the RV reduce the weight of magnetometers and increase the weight of the gyroscopes, the attack fails to cause any negative impact (green line in Figure 7b).

❼ **Environmental Conditions.** Attacks need specific environmental prerequisites. This is because each sensor requires a unique environmental condition to correctly measure its values; thus, the impacts of the attacks also change according to the environment. For example, both optical flow sensors and GNSS receivers are used to measure an RV's current position. Yet, optical flow sensors fail to calculate changes in positions on floors without feature points (e.g., deserts) [22]. Similarly, GNSS receivers fail to calculate precise positions in an urban canyon due to multipath issues [39].

Attackers can exploit such a fact to achieve their goals. To illustrate, as demonstrated in Section 4, GPS spoofing fails to deceive the GNSS receiver because the GNSS receiver uses data from multiple satellite constellations. Yet, in specific scenarios where a target RV is in proximity to man-made structures, GPS spoofing can successfully achieve its attack goals. This is because the RV's GNSS receiver is hard-pressed to receive data from benign satellites, but it receives data from fake satellite signals created by the attackers.

6. Design Challenges

Our goal is to infer the prerequisites that make attacks possible; however, this process raises three unique challenges.

C1: Simulating Attacks. The first challenge is simulating attacks accurately. The simulators create sensor values with the following steps: (1) randomly select a sensor value, (2)

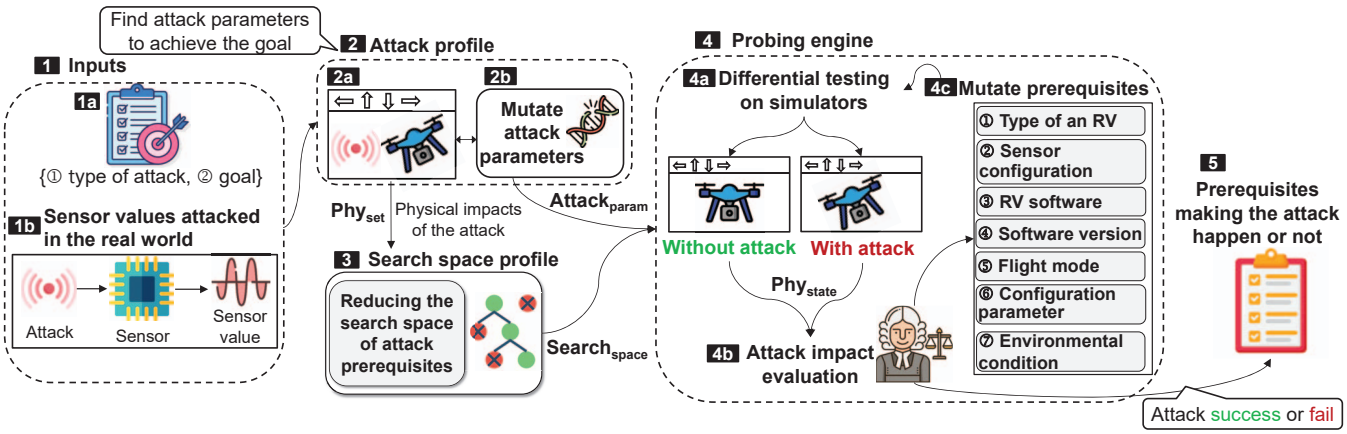


Figure 10: Overview of RVPROBER's workflow.

add noises created by a standard normal distribution to the sensor value, and (3) feed the sensor value into filters. RV developers simulate sensor disturbances by adjusting the noise levels in the second step. Yet, attacks in the real world can have varying effects and may not follow a standard normal distribution due to different hardware and attack strategies used by attackers. To address this, RVPROBER first takes, as input, sensor values compromised under the attack; it then mutates the compromised sensor values based on four attack parameters: `intensity`, `duration`, `start time`, and `attack algorithm`.

C2: Natural Deviations. The second challenge concerns how to distinguish attack failure and success from an RV's physical states (e.g., position errors). This task is challenging because the RV's operating conditions may naturally cause negative effects regardless of the attacks. For instance, the GNSS receiver may show more than 15 meters of position errors without any attacks when there is a multipath issue [39] in an urban canyon. Further, the RV can deviate from a planned navigation path or even crash on the ground due to naturally occurring sensor glitches and wind gusts. To address this issue, RVPROBER first runs the same test case with and without an attack. It then extracts the RV's altered physical states caused by the attacks.

C3: High-dimensional Search Spaces. Each prerequisite creates an expansive search space. Thus, the last challenge is the high-dimensional search spaces of seven prerequisites and four attack parameters. In particular, the following three prerequisites create high-dimensional search spaces: software versions, configuration parameters, and environmental conditions (e.g., ArduPilot has 15,840 software versions, 1,140 parameters, and 168 environmental conditions). The combinations of these prerequisites are almost boundless. Thus, a method is required to reduce the search space while discovering the combinations of the prerequisites and attack parameters that make the attack possible. To overcome this challenge, RVPROBER leverages a binary search algorithm to probe software versions. Additionally, RVPROBER extracts relevant configuration parameters and environmental conditions through static and dynamic analysis; it then tests only the extracted ones.

7. RVPROBER

We introduce RVPROBER, a framework for probing the seven types of prerequisites required for attacks. Figure 10 shows the four main parts of RVPROBER. First, it takes, as input, a type of attack, its corresponding goal, and sensor values compromised by the attack. Second, it finds an appropriate set of attack parameters to achieve the goal. Third, it narrows down the search space by excluding particular configuration parameters and environmental conditions (from attack prerequisites) that are not relevant to the attack. Fourth, it mutates seven different types of prerequisites to discover the prerequisites required for conducting attacks successfully.

7.1. Inputs

RVPROBER takes, as input, a type of attack, its goal, and the sensor values compromised by the attack (1a and 1b in Figure 10). The users can select one of the 12 attacks in Table 3 and one of the three attack goals (mission failure, unstable attitude/position, or physical crash) in Table 2. Here, the users can optionally input a threshold for unstable attitude/position. For example, {`position`, `10 meters`} denotes that RVPROBER will consider more than 10 meters of deviated positions as successful "unstable position" attacks.

We use the compromised sensor values as a guide to find the sensor values required to achieve attack goals in our setting. Users can obtain compromised sensor values taken during an attack with two different methods. First, the users can directly obtain compromised sensor values captured during the attack if they have the hardware and software required to conduct the attack. Second, they can obtain a log file taken during the attack. Then, they upload the file into log analysis software [8], [73] and obtain compromised sensor values stored in the log file.

7.2. Attack Profiling

This step's goal is twofold: (*i*) identifying physical states (e.g., position) affected by the attack (selected in Section 7.1) and (*ii*) determining the appropriate attack parameters necessary to achieve the attack goal on simulators.

Monitoring Physical States. The 12 attacks aim to cause physical crashes, instability, and mission failure. Thus, to

identify the physical states affected by the attack, we need to monitor an RV's position, attitude, and status. To do so, RVPROBER parses status messages sent by an RV to a ground control station (GCS). From the status messages, it monitors six physical states, including `latitude`, `longitude`, `altitude`, `roll`, `pitch`, and `yaw`, as well as three warning messages, namely "triggering fail-safe logic", "crashing on the ground", and "disarming motors".

Attack Parameters. RVPROBER requires attack parameters before probing the prerequisites. The reason is that the given compromised sensor values may not be sufficient to achieve the attack's goal. For example, filtering algorithms in an RV's software may filter most of the compromised sensor values, resulting in no noticeable impact on the RV's physical behavior. Unfortunately, the research literature [22], [42], [43], [82], [83] does not mention any specific attack parameters to achieve the attack's goals. To address this issue, RVPROBER automatically finds the attack parameters.

The attack parameters include (1) intensity of the spoofed sensor values, e.g., how much positioning error a GPS spoofing attack can cause; (2) attack duration, e.g., how long the attack disturbs a GPS receiver; (3) attack start time, e.g., at which flight stage is the RV under the GPS attack; and (4) attack algorithms, e.g., suddenly changing GPS lock or gradually overtaking GPS lock [60].

We note that RVPROBER identifies the required attack parameters from the software version mentioned in the research literature. In the absence of such software version information, RVPROBER determines the attack parameters from the latest RV control software version.

Determining Attack Success Thresholds. We need to determine how to establish attack success conditions for assessing whether an attack achieves its goal. This is because users might not input the specific thresholds in the previous step (Section 7.1), and RVPROBER requires reasonable thresholds to evaluate attacks. We can easily detect mission failures and physical crashes on simulators. Yet, unfortunately, none of the research literature mentions exact attitude/position errors as their attack goals. To address this issue, we obtain such thresholds (e.g., 5 meters of position error [12]) from test cases created by RV control software developers.

Detailed Steps in Attack Profile. RVPROBER performs the following steps: (1) it runs a default mission[3] created by ArduPilot developers [10], (2) it injects the compromised sensor values into the RV's sensor(s), (3) it mutates four different attack parameters if the injected attack fails to achieve the attack's goal (❷b in Figure 10), and (4) RVPROBER stores the RV's changed physical states and attack parameters if the injected attack achieves the goal (❷a in Figure 10).

Mutating Attack Parameters. To discover appropriate attack parameters, RVPROBER randomly selects one of them. Then, it (*i*) switches the brute-force attack to a hill climbing algorithm [53], [60], [96] (❶ in Figure 11 and purple lines in Figure 12c), (*ii*) multiplies an initial `intensity` by a factor of two, injecting more severe sensor disturbances (❷ in Figure 11 and blue lines in Figure 12a), (*iii*) multiplies an initial `duration` by a factor of two, injecting the compromised sensor values multiple times (❷ in Figure 11 and cyan lines in Figure 12b), or (*iv*) divides `start time` by a factor of two, injecting the compromised sensor values at an earlier time (❸ in Figure 11).

3. Other RV software packages can also execute this mission if they follow the MAVLink protocol [54].

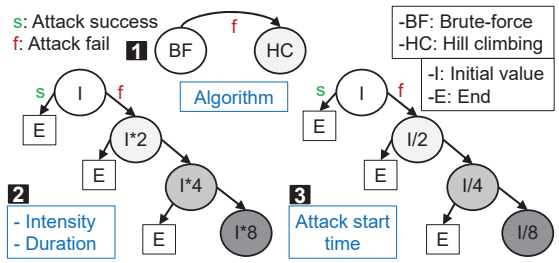
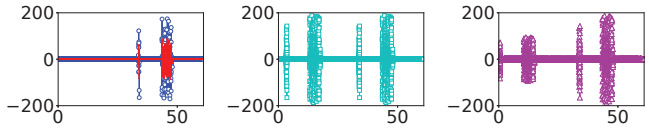


Figure 11: Mutation strategies for attack parameters.



(a) Intensity. (b) Duration and start time. (c) Algorithm.

Figure 12: Mutating attack parameters. X-axis and Y-axis denote time (seconds) and degrees per second measured in a gyroscope sensor, respectively. The red line in Figure 12a represents sensor values under an acoustic noise injection attack in the real world.

7.3. Reducing the Search Space of Prerequisites

This step (❸ in Figure 10) reduces the high-dimensional search spaces of configuration parameters and environmental conditions as attack prerequisites (*C3* in Section 6). We obtain an RV's physical states that are affected by the attack in the previous step (Section 7.2). Here, we map the identified physical states to particular configuration parameters and environmental conditions. To do so, we adapt PGFuzz's input profiling engine [45] that allows RVPROBER to map the configuration parameters and environmental conditions to the physical states through static and dynamic analysis. Yet, unlike PGFuzz, RVPROBER probes the configuration parameters only related to sensor filters, EKF sensor fusions, and fail-safe logic rather than probing the entire set of configuration parameters. This is because other configuration parameters directly alter hardware configurations, thus disrupting an RV's operation even without any attack, or do not directly change the impacts of the attack.

7.4. Probing Engine

We note that RVPROBER's probing engine aims to achieve soundness rather than completeness from the dynamic analysis. The soundness denotes that: (a) when RVPROBER identifies prerequisites that enable an attack in a simulated environment, those prerequisites are also required for conducting the attack successfully in the real world; and (b) when RVPROBER identifies prerequisites that disable an

Algorithm 1 RVPROBER's probing engine for configuration parameters and environmental conditions.

Input: A time limit τ, a simulator SIM, RV's physical states $\texttt{Phy}_{\texttt{set}}$, configuration parameters and environmental conditions related to the attack $\texttt{search}_{\texttt{space}}$, attack parameters $\texttt{Attack}_{\texttt{param}}$
Output: Mutated configuration parameters and environmental conditions $\texttt{I}_\texttt{s}$ making the attack successful, $\texttt{I}_\texttt{f}$ making the attack fail

1: **function** RVPROBER($\texttt{Attack}_{\texttt{param}}$, τ) ▷ Main
2: $\texttt{I} \leftarrow$ RANDOM($\texttt{search}_{\texttt{space}}$) ▷ Get randomly selected inputs
3: **while** time $< \tau$ **do**
4: $\texttt{SIM}_{<1,2>} \leftarrow$ SIM.initialize() ▷ Initialize simulators
5: $\texttt{State}_\texttt{n} \leftarrow \texttt{SIM}_1$.execute(I, $\texttt{Attack}_{\texttt{off}}$) ▷ Collect RV's state error
6: $\texttt{State}_\texttt{a} \leftarrow \texttt{SIM}_2$.execute(I, $\texttt{Attack}_{\texttt{on}}$) ▷ Collect RV's state error
7: $\texttt{Result} \leftarrow$ ORACLE($\texttt{State}_\texttt{a} - \texttt{State}_\texttt{n}$) ▷ Check attack success
8: **if** Result = success **then**
9: $\texttt{I}_\texttt{s} \leftarrow \texttt{I}_\texttt{s} \cup \texttt{I}$ ▷ Store the input making the attack successful
10: **else**
11: $\texttt{I}_\texttt{f} \leftarrow \texttt{I}_\texttt{f} \cup \texttt{I}$ ▷ Store the input making the attack fail
12: **end if**
13: $\texttt{I} \leftarrow$ MUTATE($\texttt{search}_{\texttt{space}}$, $\texttt{State}_\texttt{n}$, I) ▷ Mutate the input
14: **end while**
15: **return** $\texttt{I}_\texttt{s}$, $\texttt{I}_\texttt{f}$
16: **end function**
17: **function** MUTATE($\texttt{search}_{\texttt{space}}$, $\texttt{State}_\texttt{n}$, I) ▷ Pick another input
18: **if** ORACLE($\texttt{State}_\texttt{n}$) = *success* **then** ▷ Input is self-sabotaging
19: $\texttt{search}_{\texttt{space}} \leftarrow \texttt{search}_{\texttt{space}} - \texttt{I}$ ▷ Exclude self-sabotaging input
20: **end if**
21: $\texttt{I} \leftarrow$ RANDOM($\texttt{search}_{\texttt{space}}$) ▷ Pick another input
22: **return** I
23: **end function**

attack in a simulated environment, those prerequisites also make the attack fail in the real world.

Differential Testing. This step extracts the attack's impacts under different prerequisites (4a in Figure 10). To achieve this, RVPROBER conducts the same attack on two different simulators, one with the attack and another one without it (*C2* in Section 6). The reason is that the prerequisites (e.g., wind speed) can also naturally disturb the RV's physical states, causing similar symptoms to those of successful attacks. RVPROBER monitors the RV's physical states for a user-defined period. It then extracts the differences in the physical states between the two simulation results.

Evaluating Attack Impacts. This step decides whether the attack achieves its goal or not (4b in Figure 10). RVPROBER leverages thresholds given by users if provided. Otherwise, RVPROBER employs the thresholds established by RV control software developers in their test cases [12].

Mutating Prerequisites. RVPROBER leverages three strategies to test the seven types of prerequisites. First, RVPROBER tests all possible combinations of RV types, sensor configurations, RV control software, and flight modes (4c in Figure 10). Second, RVPROBER uses a binary search to discover from which software version the attack is possible. Third, RVPROBER mutates configuration parameters and environmental factors using Algorithm 1.

The algorithm repeatedly conducts the following. (1) It takes, as input, the attack parameters discovered in 2 in Figure 10 (Line 1). (2) It randomly selects an input from $\texttt{search}_{\texttt{space}}$ identified in 3 in Figure 10, and assigns a random value to the selected input (Line 2). Here, the random value for the input is still within the valid range defined by RV software developers. We note that RVPROBER tests a single input at a time, e.g., changing wind speed. (3) It conducts differential testing to verify the attack's success is due to the injected sensor readings (Lines 5 and 6). (4) It checks whether the executed input still makes the attack successful (Line 7). Here, we consider the attack successful only if the simulation with the attack achieves the attack's goal (Line 6), but another simulation without the attack fails to achieve the goal (Line 5). (5) It selects another input if RVPROBER detects a successful attack (Lines 8 and 21), i.e., the attack is independent of the tested input (configuration parameter or environmental condition). (6) In the event that the assigned value of the input leads the simulation, without the attack, to achieve the attack's goal (Lines 18 and 19). It means that the input-value pair is self-sabotaging, resulting in attack symptoms. (e.g., turning off sensors). Thus, the algorithm, during its random selection of a value in step (2), excludes such self-sabotaging pairs. RVPROBER terminates when the testing time exceeds a user-defined time limit τ.

8. Implementation

Attack Implementation. We performed all the demonstrated attacks presented in Table 3 in a real-world setting to obtain the compromised sensor values, with the exception of "EMI injection on SPI/I2C" attacks [42]. This is because we do not have the EMI generation hardware used in [42]. Instead, we first manually analyzed the maximum sensor reading errors from the demo video available in [42]. We then injected an equivalent level of sensor noises into SPI/I2C. We detail the experimental equipment we used in Table 5 in Appendix B.

To simulate the attacks in a software-in-the-loop (SITL) simulator, we (*i*) added scheduling jitters in SITL, and (*ii*) injected sensor values compromised by the attack in the real world. The motivation for adding scheduling jitters is that Jeong et al. [43] show such scheduling jitters contribute to the attack's negative impacts. Yet, SITL simulators on x86 machines do not have such scheduling jitters due to x86 machines' fast processing times compared to microcontrollers. The scheduling jitters were set at a standard deviation of approximately 103 μs based on the experimental results presented in [43].

Profile & Probing Engine. We added 246 lines of code (LoC) into PGFuzz's profiling engine [45] to extract configuration parameters related to sensor filters, EKF sensor fusions, and fail-safe logic. We wrote 2,695 LoC using Pymavlink *v2.4.16* [80] that enables RVPROBER to communicate with simulated RVs via MAVLink protocol [54]. ArduPilot and PX4 implement MAVLink protocol differently. Thus, we additionally modified 284 LoC for PX4's attack profile and probing engine. We used Gazebo *v.11.12.0* [32] to simulate RVs. We tested released software versions in ArduPilot and PX4 rather than testing every minor software update. We used Gazebo plugins [33] to simulate GNSS multipath regarding GNSS spoofing and jamming. Further, we adopted the list of environmental conditions specified in PatchVerif [46].

TABLE 4: Summary of required prerequisites to achieve, as the attack's goal, "Physical crash" in ArduPilot and how many real-world RV configurations satisfy the required prerequisites.

	Attack prerequisites that make the attack successful / all prerequisites related to the attack						How many RV configurations satisfy prerequisites?	
	(C_1) Type of RV[α]	(C_2) Sensor configuration	(C_3) Software version	(C_4) Flight mode A: Autonomous M: Manual	(C_5) Configuration parameter	(C_6) Environmental condition	C_1-C_2	C_1-C_6
GNSS spoofing (d)[β]	D, R	1/6	*v.3.2.0 - v.3.5.0*	A	6/9	Downtown	5.94%	0%
GNSS spoofing (w)	D, R	2/6	*v.3.2.0 - v.4.3.5*	A	6/9	Downtown/rural	90.09%	58.41%
GNSS jamming (d)	D, R	2/6	*v.3.2.0 - v.3.5.0*	A	6/9	Downtown/rural	90.09%	0%
GNSS jamming (w)	D, R	2/6	*v.3.2.0 - v.3.5.0*	A	6/9	Downtown/rural	90.09%	0%
Acoustic noise (d)	D, S	1/3	*v.3.2.0 - v.3.3.2*	A	3/9	N/A	18.81%	0%
Acoustic noise (w)	D, S	3/3	*v.3.2.0 - v.4.3.5*	A	3/9	N/A	94.05%	64.35%
EMI Injection on mag (d)	N/A	N/A	N/A	N/A	N/A	N/A	0%	0%
EMI Injection on mag (w)	D	2/2	*v.3.2.0 - v.3.4.0*	A	2/3	N/A	94.05%	0%
EMI Injection on SPI (w)	D, R, S	1/2	*v.3.2.0 - v.4.3.5*	A	3/3	N/A	100%	64.35%
EMI Injection on SPI (w)	D, R, S	1/2	*v.3.2.0 - v.4.3.5*	A	3/3	N/A	100%	64.35%
Optical flow spoofing (d)	D	1/2	*v.3.2.0 - v.4.3.5*	A	3/12	N/A	0.99%	0%
Optical flow spoofing (w)	D	1/2	*v.3.2.0 - v.4.3.5*	A	3/12	N/A	0.99%	0%

[α] D: Drone, R: Rover, S: Submarine [β] d: Demonstrated attack, w: Worst-case attack

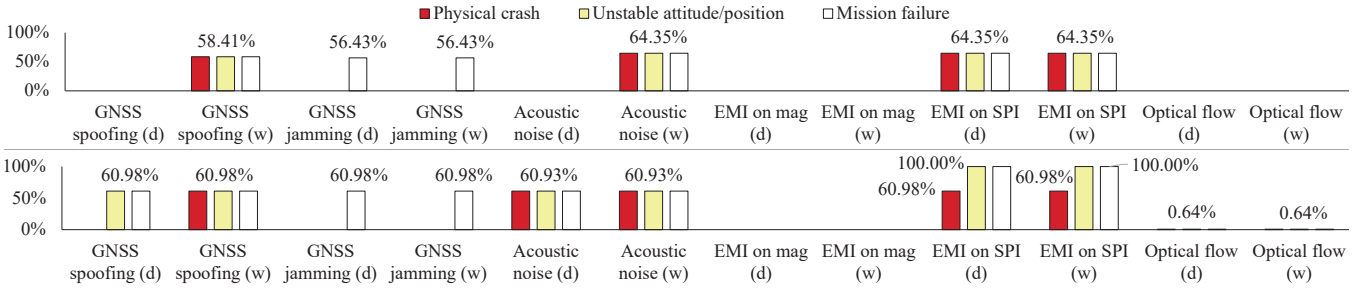


Figure 13: The quantified real-world hardness in ArduPilot (upper) and PX4 (lower). Percentages represent how many real-world RV configurations satisfy the required attack prerequisites. `d` and `w` on the x-axis denote demonstrated attacks and worst-case attacks, respectively.

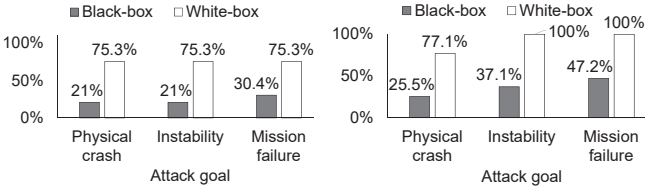
Collecting RV Logs. PX4 users upload their RV logs into PX4's publicly accessible log database [77] for the purpose of log analysis. We collected 29,958 logs from the log database using a script provided by developers. We then wrote 554 LoC using the PX4ULog Python library to parse the logs. We manually collected 101 ArduPilot logs from its community website [6] because ArduPilot does not have any public log databases. The collected logs from PX4 and ArduPilot were created from December 2016 to February 2023.

9. Evaluation

We evaluated RVPROBER on 12 well-known attacks (in Table 3) with ArduPilot [4] and PX4 [66]. We excluded Paparazzi from our evaluations because its control algorithm is ported from PX4's control algorithm [70]; thus, the majority of its control algorithms overlaps with PX4's ones. Nevertheless, we acknowledge that Paparazzi's code differences could potentially change its behavior when some of the considered attacks are performed. We leave conducting a full evaluation of Paparazzi as future work.

9.1. Prerequisites Required to Conduct Attacks

The required prerequisites vary according to each attack, adversary capability, and the attack's goal. Table 4 shows (*i*) the discovered prerequisites to achieve, as the attack's goal, "physical crash" in ArduPilot, and (*ii*) how many real-world RV configurations (log files) are vulnerable to attacks (details of PX4's ones in Appendix C). The 12 attacks require, on average, 4.2 different types of prerequisites. `GNSS spoofing` (d) requires the most prerequisites, while `EMI injection on SPI` requires the fewest. `EMI injection on mag` (d) cannot achieve the attack goals because RV software assigns small weights to magnetometers in the sensor fusion algorithms (EKFs); thus, spoofing one of the magnetometers cannot cause noticeable negative impacts.

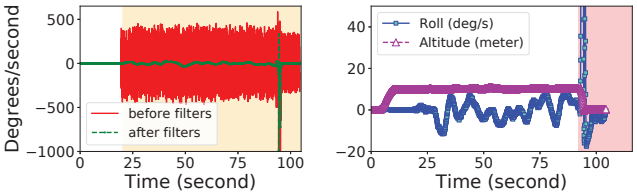


(a) ArduPilot. (b) PX4.

Figure 14: Comparison of black-box and white-box attack models in terms of quantified real-world attack hardness.

9.2. Analysis of Quantified Attack Hardness

We analyze how many real-world RV configurations among 30,059 RV logs are vulnerable to each attack in terms of the attack's goals and adversary capabilities, as shown in Figure 13. Only "GNSS spoofing", "acoustic noise", and "EMI on SPI" attacks can make RVs physically crash. Optical flow spoofing attacks do not require many prerequisites, but a few real-world RV configurations leverage the optical



(a) Gyroscope measurements. (b) Roll angle and altitude.

Figure 15: Impacts of injected acoustic noises on ArduPilot. The yellow area on the figure denotes the activated vibration fail-safe. The RV crashes on the ground at around 90 seconds.

flow (10.08%). Thus, only 0.64% of the actual users are vulnerable to the optical flow spoofing attacks.

PX4 control software requires fewer prerequisites in comparison to ArduPilot, with an average of 2.8 prerequisites versus 4.2 for ArduPilot. Thus, PX4 is more susceptible to the attacks than ArduPilot. Notably, when the real-world RV configurations satisfy the "type of RVs" and "flight mode" prerequisites, a majority of the attacks can successfully achieve their goals. Such a fact results in the quantified hardness of the attacks being consistent at around 61%.

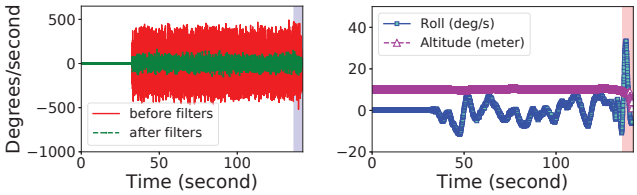
White-box Attack Model. Figure 13 assumes a black-box attack model, wherein attackers lack knowledge about an RV's software, hardware, and environmental configurations. This attack model accounts for scenarios in which a user changes an RV's configurations, leading attackers to blindly carry out their attacks. On the contrary, the white-box attack model considers scenarios in which the user does not alter the RV's configurations, and the attackers possess knowledge about the RV's configurations. The white-box attack model achieves an average 2.9-fold increase in attaining attack goals compared to the black-box model when the attackers conduct attacks against all RVs, as shown in Figure 14. 24.7% of ArduPilot users are still not vulnerable to white-box attacks because they leverage manually-controlled flight modes. On the contrary, none of PX4 users are immune to white-box attacks due to design flaws (Detailed in Section 9.3.1).

9.3. Case Studies

We explain required prerequisites and root cause analysis pertaining to acoustic noise [43], [85], [88], [89], [93] and electromagnetic interference (EMI) injection [42] attacks. We detail the experimental setups for these two attacks in Appendix E and Appendix F. During the root cause analysis, we reported the identified design flaws of the RV software to the developers and are awaiting their responses.

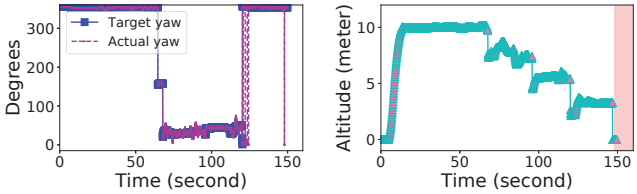
9.3.1. Case Study 1 - Acoustic Noise Injection Attacks. We successfully replicate the attacks based on the prerequisites identified by RVPROBER.

Attack Prerequisites (Demonstrated Attack). RVPROBER discovers that this attack requires five different types of prerequisites. First, the types of RVs must be drones or submarines because rovers on the ground are not vulnerable to this attack (as shown in Figure 4a). Second, an RV must be equipped with only one IMU (gyroscope and accelerometer).



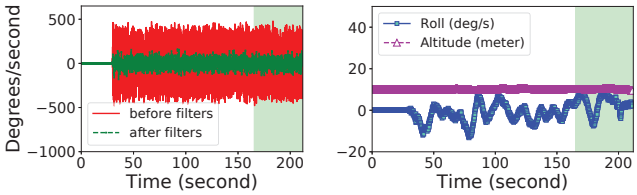
(a) Gyroscope measurements. (b) Roll angle and altitude.

Figure 16: After turning off the vibration fail-safe. The purple area on the figure denotes the activated GPS glitch fail-safe logic. The RV crashes on the ground at around 140 seconds.



(a) Yaw angles. (b) Altitude.

Figure 17: Turning off vibration and GPS glitch fail-safe. The RV crashes on the ground at around 150 secs.



(a) Gyroscope measurements. (b) Roll angle and altitude.

Figure 18: Turning off (*i*) the vibration fail-safe, (*ii*) GPS glitch fail-safe, and (*iii*) "yaw reset". The green area on the figure represents turned on EKF fail-safe.

This attack fails to achieve its goal against an RV that has two or three IMUs (the sensor configuration column) due to the EKF switching, as shown in Figure 5a. Third, this attack succeeds in achieving the attack goal when ArduPilot software versions are from *v.3.2.0* to *v.3.3.2*. Other software versions (*v.3.3.3* to *v.4.3.5*) show unstable attitude/position control and mission failure rather than physically crashing on the ground due to the improved sensor filter and sensor fusion algorithms (Figure 6 and Figure 8). Fourth, only autonomous flight modes leverage the IMU sensors. Thus, manually-controlled flight modes are not vulnerable to the attack. Fifth, RVPROBER discovers that configuration parameters (e.g., `EK3_GYRO_P_NSE` and `EK3_ACC_P_NSE`) directly determine the attack's success regardless of the attacker's capabilities (in Table 3) and attack parameters (in Figure 12). The reason is that these two parameters set how much to weigh IMU readings in the sensor fusion step.

Attack Prerequisites (Worst-case Attack). This attack does not require, as prerequisites, particular sensor configurations and software versions compared to `acoustic noise` (d) because `acoustic noise` (w) disturbs an RV's all IMUs.

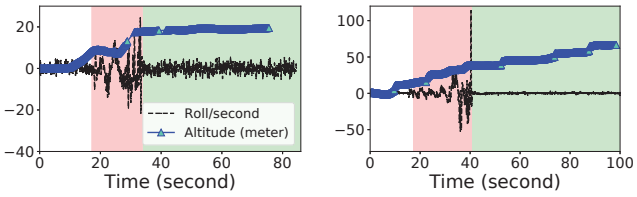
Root Cause Analysis (ArduPilot). RV software is designed to detect sensor disturbances and then activate EKF fail-safe logic [24] to react to the disturbances, e.g., changing the RV's current flight mode to a manually-controlled mode. Yet, ArduPilot fails to trigger the EKF fail-safe logic while testing the attacks, as shown in Figure 15.

We discover previously-unknown root causes of the successful attacks by manually analyzing multiple software versions. This is because we notice that each software version uses slightly varied fail-safe logic, leading to entirely distinct symptoms under the attacks. Interestingly, the fail-safe algorithms of RV control software, intended to address sensor disturbances, are the primary cause of the physical crashes due to three design flaws.

First, ArduPilot incorrectly assumes the IMUs spoofed by the attacks as a high level of physical vibrations on its body frame. This wrong assumption leads to a physical crash as follows: (1) ArduPilot triggers vibration fail-safe logic [92] (yellow area on Figure 15a). It switches from EKF sensor fusion to a complementary filter which is tuned to be more resistant to vibration (but less accurate) than the EKF. Indeed, the complementary filter aggressively removes gyroscope disturbances compared to the EKF (See Figure 15a, Figure 16a, and Figure 18a). (2) The RV starts to lose attitude and position control due to the less accurate complementary filter (around 90 seconds in Figure 15a) and eventually crashes on the ground. In fact, the RV maintains a stable attitude and position for 50 seconds longer than before when we turn off the vibration fail-safe logic, as shown in Figure 16.

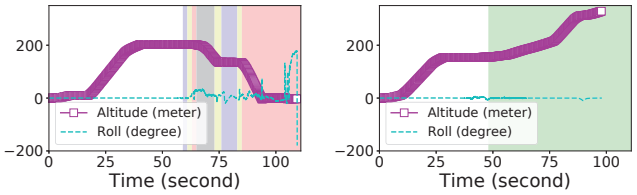
Second, the RV still crashes on the ground despite turning off the vibration fail-safe logic. Another design flaw is that ArduPilot incorrectly assumes compromised IMUs as GNSS signal disturbances. This design flaw leads to a physical crash as follows: (1) ArduPilot triggers GPS glitch fail-safe logic [34] due to large deviations between IMU and GNSS readings. ArduPilot trusts more IMUs than GNSS because GNSS signals are prone to being disturbed by multipath issues [39]. On the contrary, IMUs do not frequently suffer from natural disturbances. (2) ArduPilot stops using GNSS data during the GPS glitch fail-safe mode. (3) The RV loses attitude and position control as the sensor fusion algorithm eliminates the GNSS readings and eventually crashes on the ground (140 seconds in Figure 16b).

Third, the RV still physically crashes on the ground despite turning off both vibration and GPS glitch fail-safe logic. Another design flaw is "yaw reset" [26]. ArduPilot causes a physical crash as follows: (1) It detects large deviations between IMU and magnetometer readings. (2) It incorrectly assumes that magnetometers suffer from natural disturbances. This wrong assumption stems from the RV control software's design to place greater trust in IMUs than in magnetometers. (3) It stops using the magnetometers and leverages IMUs and GNSS to estimate the RV's yaw angle (EKF-Gaussian Sum Filter [9], [75]). (4) Unfortunately, the acoustic noises compromise the IMUs. Thus, the newly calculated target yaw angle is totally different from previous ones (at 53 seconds in Figure 17a). (5) The RV aggressively changes its yaw angle to reach the new target yaw angle.



(a) Compromising an IMU. (b) Compromising all IMUs.

Figure 19: After modifying PX4's fail-safe logic. The red and green areas on the figure represent injected acoustic noises and turned on modified EKF fail-safe, respectively.



(a) Default fail-safe logic. (b) Turned on EKF fail-safe.

Figure 20: EMI attacks succeed or fail according to the fail-safe logic in ArduPilot. The green area on the figure represents the turned-on EKF fail-safe. The purple, yellow, red, and gray areas denote "yaw reset", EKF switching, GPS glitch fail-safe, and vibration fail-safe, respectively.

Yet, such acrobatic movements make the RV lose its position control (at 53 seconds in Figure 17b) and eventually crash on the ground (at 150 seconds in Figure 17b).

We modify ArduPilot's fail-safe logic to trigger the EKF fail-safe. In this case, ArduPilot maintains a quite stable attitude and position under the attack, as shown in Figure 18.

Root Cause Analysis (PX4). PX4 is susceptible to both demonstrated and worst-case acoustic noise injection attacks due to three primary reasons. First, as explained in Section 5, PX4 always disables low-pass filters inside IMUs. The turned-off low-pass filters make PX4's harmonic notch filters less effective than ArduPilot's ones, as shown in Figure 4c and Figure 4d. Second, PX4 leverages too high thresholds (e.g., `EKF2_SEL_IMU_ANG`) to trigger fail-safe logic (Detailed in Appendix D). Thus, it fails to promptly respond to the attacks. Indeed, PX4 triggers EKF switching after the RV crashes on the ground at around 90 seconds in Figure 4d (See details of the RV log [74]). Third, PX4's fail-safe logic sends a warning message to users rather than actively addressing the failure on sensors. Thus, the RV still crashes on the ground even if PX4 triggers the fail-safe. We modify PX4's fail-safe logic to trigger a manually-controlled flight mode (e.g., `Altitude` [3]). In this case, the RV maintains a stable attitude and position under the attacks, as shown in Figure 19.

9.3.2. Case Study 2 - Electromagnetic interference (EMI) on SPI/I2C buses. We succeed in replicating the attacks based on the prerequisite identified by RVPROBER. RVs physically crash on the ground regardless of the adversary's capabilities, as shown in Figure 20a.

Attack Prerequisites (Demonstrated & Worst-case At-

tacks). RVPROBER discovers that these attacks require two different types of prerequisites. First, these attacks require, as prerequisites, particular sensor configurations. This is because an automatic parachute ejection prevents an RV from physically crashing on the ground [17]. ArduPilot detects free fall in the air and automatically releases a parachute [5]. Yet, due to its design flaw, PX4's automatic parachute ejection [78] cannot prevent crashing on the ground. We will detail the design flaw in the root cause analysis. Second, manually-controlled flight modes stop using data on compromised SPI/I2C buses; thus, these attacks fail to achieve its goals, as shown in Figure 9.

Root Cause Analysis (ArduPilot). The root cause of the successful attacks lies in the flawed assumptions underlying ArduPilot's fail-safe logic [7], which assumes the availability of at least one type of sensor that provides accurate readings. Yet, this assumption fails during the attacks since the attacks corrupt data on all sensor types connected via SPI/I2C buses.

EKF fail-safe logic [24] can address the attack by changing the RV's flight mode to a manual flight mode (e.g., `Acro` [2]). Yet, ArduPilot activates a series of fail-safe logic except for EKF fail-safe (Figure 20a). Thus, it wastes around one minute to trigger the inappropriate fail-safe logic, and it eventually crashes on the ground at around 110 seconds in Figure 20a. In fact, the RV maintains a stable attitude and altitude under attacks after modifying ArduPilot to turn off all fail-safe logic except for EKF fail-safe (Figure 20b).

Root Cause Analysis (PX4). PX4 detects disturbances on sensors and triggers "Attitude fail-safe" logic [67] under the attack. Yet, two underlying reasons make the RV crash on the ground. First, the fail-safe is triggered when the RV's roll is deviated by more than 60 degrees (See Figure 23 in Appendix D). Yet, it is too late to recover the RV's lost attitude. Second, the triggered fail-safe logic (*i*) simply sends a warning message to a ground control station (GCS), or (*ii*) deploys a parachute if the RV has one. Yet, the deployed parachute cannot even make the RV safely land on the ground. This is because the parachute is deployed at a time when the RV's attitude is already highly unstable.

10. Limitations and Discussion

Interdependency between Prerequisites. We do not systematically consider interdependencies between configuration parameters and environmental conditions. Thus, RVPROBER may incorrectly judge an attack's negative symptoms. For example, an attack succeeds in disturbing an RV's mission, but the RV does not crash on the ground. In this case, RVPROBER concludes that the attack fails to achieve "physical crash" as its goal. Yet, a physical crash could still occur as a result of various combinations of configuration parameters and environmental conditions, such as strong wind gusts, faulty user configurations, and improperly calibrated IMU sensors [1], [21]. Inferring such interdependencies is challenging due to the high-dimensional input spaces, including human factors. We leave this topic for future work.

Modeling Attacks. We simulate the attacks by mutating the four attack parameters based on compromised sensor readings in the real world (Figure 12). Yet, one may believe that we can generate the attacks by modeling the mechanical and electrical characteristics of sensors. Yet, it is challenging because materials, sensor aging, and imperfections of the sensors during manufacturing processes change the physical phenomena inside the sensors under attack. We also leave this topic for future work.

Limited Capability of Attackers. We use non-powerful hardware resources in our real-world experiments (Appendix B and Figure 21). Yet, the experiments are still reasonable for two main reasons. First, the security community cannot anticipate that RV developers can easily access or purchase powerful hardware (e.g., military-grade) to conduct the attacks. Second, we reveal that each attack requires prerequisites regardless of the attackers' capabilities (demonstrated attacks vs. worst-case attacks in Table 4).

Simulation-to-reality Gap. To validate the identified prerequisites, we conduct real-world attacks using random-sampled sets of identified prerequisites. In particular, we perform a total of 33 real-world attacks: 11 GPS spoofing attacks, 10 acoustic noise injections, 10 EMI injections on magnetometers, and 2 optical flow spoofing attacks. Due to our hardware limitations, we exclude GNSS jamming and EMI injection on SPI attacks from the real-world experiments. We confirm that the real-world results are consistent with the simulated results, validating RVPROBER's soundness. Yet, we acknowledge the potential for discrepancies between simulated and real-world results due to the lack of completeness in RVPROBER's probing engine.

Required Manual Effort. Manual effort was required to construct attack-step models, identify the seven types of prerequisites, and analyze the root causes of sensor attacks. We believe that such a manual workload is not a significant burden for knowledgeable developers. We note that these manual steps are a one-time effort; thus, users do not need to redo these manual steps. To use RVPROBER, the only manual task is to prepare the inputs (Section 7.1). The other components of RVPROBER are automated.

Attack Hardness-Based on Constraints of Attacks. Our quantified attack hardness does not consider all the real-world constraints of attacks (e.g., due to the different availability and cost of the specific hardware components required to perform a specific attack). This choice was motivated by the fact that real-world constraints can vary and change over time. Instead, we represent hardness based on the number of real-world RV configurations (sampled from flight logs) that satisfy the required prerequisites. This is because the historical flight logs can estimate how frequently a specific attack is currently feasible/infeasible in the real world.

New Attack Types and Patterns. To allow users to test new attack types and patterns, RVPROBER provides users with the capability to simulate attacks on vision sensors (e.g., LiDAR and 3D depth camera). RVPROBER also provides an option to simulate sensor attacks without needing compromised sensor values in the real world. This is achieved by allowing users to inject a configurable level of sensor noise.

Reachability Analysis for RVs against Attacks. We may

leverage the attack-step model and the identified prerequisites to conduct a reachability analysis [20], [47] for RVs against attacks. Indeed, we can investigate how successful and unsuccessful attacks cause a victim RV to transition between unsafe and safe states. For instance, rovers on the ground leverage a 2-D coordinate system and do not use roll and pitch angles measured by IMUs. Such a fact prevents attacks targeting IMUs from reaching unsafe states (i.e., making the RV software set incorrect throttle values for the actuators), as depicted by the purple dashed line in Figure 4a. On the contrary, drones in the air use a 3-D coordinate system and heavily depend on IMUs. This dependency makes the attacks on IMUs reach unsafe states, as illustrated by the cyan straight line in Figure 4a.

11. Related Work

Hardness Study. Yan et al. [94] quantify the risk of sensor hardware designs against attacks. Yet, their quantified risk focuses on sensor hardware designs (i.e., no consideration of the entire RV hardware and software components). Nassi et al. [59] classify the hardness of sensor attacks into three levels (high, medium, and low) through the complexity of attacks. Yet, the classification criteria are abstract, impeding the accurate quantification of attack hardness, e.g., if an attack requires environmental conditions to be successful, then they categorize the complexity of the attack as high. On the contrary, we discover previously-unknown prerequisites and show how many actual users are vulnerable to attacks.

Modeling Attacks. SensorFuzz [95] models sensor values compromised by attacks as sine waves. Then, to simulate attacks, SensorFuzz mutates the sine waves by changing the amplitude, frequency, and phase. Yet, this sensor model does not consider mechanical and electrical characteristics of sensors (Section 10). Instead, we mutate compromised sensor values based on sensor measurements under the attack because the compromised sensor values reflect the mechanical and electrical characteristics of sensors.

Destructive Attacks. Martin et al. [52] inject a shock wave in close proximity to an RV. In turn, the shock wave causes deformation in the RV's propellers, leading to an inappropriate thrust and lift, ultimately resulting in crashing on the ground. Yet, in this paper, we focus on evaluating the hardness of sensor spoofing and jamming, rather than delving into physical deformation attacks.

12. Conclusions

Our paper reveals that performing successful physical sensor attacks requires satisfying specific prerequisites and attack parameters. To automatically find these required prerequisites and attack parameters, we introduce RVPROBER. RVPROBER discovers that (*i*) 12 well-known sensor attacks require, on average, 4.4 different types of prerequisites; and (*ii*) on average, 57.08% of actual users are vulnerable to the 12 attacks. By satisfying the prerequisites, we increase the number of successful attacks from 6 to 11. We also discover that 4 out of the 12 attacks are possible due to previously-unknown design flaws in the RV control software's fail-safe algorithms.

Acknowledgments

We thank the anonymous reviewers and shepherd for their valuable suggestions. The authors from Purdue University were supported in part by the Office of Naval Research (ONR) under Grant N00014-20-1-2128. The author from KAIST was supported by the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT; Grant No. 2020-0-01202). Any opinions, findings, and conclusions in this paper are those of the authors only and do not necessarily reflect the views of the aforementioned sponsors.

References

[1] Accelerometer Calibration, https://tinyurl.com/62t4z2hj, 2023.

[2] Acro Mode, https://tinyurl.com/yckzrynb, 2023.

[3] Altitude Mode, https://tinyurl.com/3hzrnp6w, 2023.

[4] ArduPilot, https://ardupilot.org/, 2023.

[5] ArduPilot Automatic Parachute Ejection, https://tinyurl.com/34ap96xr, 2023.

[6] ArduPilot Community, https://discuss.ardupilot.org, 2023.

[7] ArduPilot Fail-safe Mechanisms, https://tinyurl.com/5hxxbdn9, 2023.

[8] ArduPilot Flight Log Analysis, https://tinyurl.com/mwuchenp, 2023.

[9] ArduPilot Gaussian Sum Filter, https://tinyurl.com/9amat3zc, 2023.

[10] ArduPilot Generic Mission, https://tinyurl.com/262pzf4y, 2023.

[11] ArduPilot Harmonic Notch Filters, https://tinyurl.com/kftxteu5, 2023.

[12] ArduPilot Position Error Threshold, https://tinyurl.com/yn565cy8, 2023.

[13] ArduPilot Sensor Configurations, https://tinyurl.com/2awccppd, 2023.

[14] ArduPilot Sensor Fusion Evolution, https://tinyurl.com/8prrd83n, 2023.

[15] ArduPilot Source Code of Fail-safe Logic, https://tinyurl.com/3wy895es, 2023.

[16] ArduSub, http://www.ardusub.com, 2023.

[17] Automatic Parachute Ejection, https://tinyurl.com/2nv4796h, 2023.

[18] L. Basano and P. Ottonello, "Complete destructive interference of partially coherent sources of acoustic waves," *Physical review letters*, 2005.

[19] R. Bergh, H. Lefevre, and H. Shaw, "An overview of fiber-optic gyroscopes," *Journal of Lightwave Technology*, 1984.

[20] X. Chen and S. Sankaranarayanan, "Reachability Analysis for Cyber-Physical Systems: Are We There Yet?" in *Proceedings of the NASA Formal Methods Symposium*, 2022.

[21] Compass Calibration, https://tinyurl.com/ymudw2vt, 2023.

[22] D. Davidson, H. Wu, R. Jellinek, V. Singh, and T. Ristenpart, "Controlling UAVs with Sensor Input Spoofing Attacks," in *Proceedings of the USENIX workshop on offensive technologies (WOOT)*, 2016.

[23] EKF Affinity and Lane Switching, https://tinyurl.com/5n6u4jej, 2023.

[24] EKF Fail-safe, https://tinyurl.com/579tv9db, 2023.

[25] Electromagnetic Shielding, https://tinyurl.com/5spwyhvh, 2023.

[26] Emergency Yaw Reset, https://tinyurl.com/5f2k4pa2, 2023.

[27] M. Euston, P. Coote, R. Mahony, J. Kim, and T. Hamel, "A complementary filter for attitude estimation of a fixed-wing UAV," in *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems*, 2008.

[28] I. Fernández-Hernández, V. Rijmen, G. Seco-Granados, J. Simon, I. Rodríguez, and J. D. Calle, "A navigation message authentication proposal for the Galileo open service," *NAVIGATION: Journal of the Institute of Navigation*, 2016.

[29] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, 1981.

[30] FMUv3 Hardware Design, https://tinyurl.com/mvn5wcbk, 2023.

[31] F. Garzotto, P. Paolini, and D. Schwabe, "HDM—a model-based approach to hypertext application design," *ACM Transactions on Information Systems (TOIS)*, 1993.

[32] Gazebo, http://gazebosim.org/, 2023.

[33] GNSS Multipath Plugin, https://tinyurl.com/mtkund6e, 2023.

[34] GPS Fail-safe, https://tinyurl.com/yck8br39, 2023.

[35] GPS for Yaw, https://tinyurl.com/5278z33b, 2023.

[36] GPS-SDR-SIM project, https://github.com/osqzss/gps-sdr-sim, 2023.

[37] HackRF One, https://greatscottgadgets.com/hackrf, 2023.

[38] Z. Haider and S. Khalid, "Survey on effective GPS spoofing countermeasures," in *Proceedings of the IEEE International Conference on Innovative Computing Technology (INTECH)*, 2016.

[39] L.-T. Hsu, "GNSS multipath detection using a machine learning approach," in *Proceedings of the International Conference on Intelligent Transportation Systems (ITSC)*, 2017.

[40] ICM-20689 Datasheet, https://tinyurl.com/4w8h4wn2, 2023.

[41] A. Jafarnia Jahromi, A. Broumandan, J. Nielsen, and G. Lachapelle, "GPS spoofer countermeasure effectiveness based on signal strength, noise power, and C/N0 measurements," *International Journal of Satellite Communications and Networking*, 2012.

[42] J. Jang, M. Cho, J. Kim, D. Kim, and Y. Kim, "Paralyzing Drones via EMI Signal Injection on Sensory Communication Channels," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2023.

[43] 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 Network and Distributed System Security Symposium (NDSS)*, 2023.

[44] M. Kaur, S. Kakar, and D. Mandal, "Electromagnetic interference," in *Proceedings of the IEEE International Conference on Electronics Computer Technology*, 2011.

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

[46] H. Kim, M. O. Ozmen, Z. B. Celik, A. Bianchi, and D. Xu, "PatchVerif: Discovering Faulty Patches in Robotic Vehicles," in *Proceedings of the USENIX Security Symposium*, 2023.

[47] C. Kwon and I. Hwang, "Reachability analysis for safety assurance of cyber-physical systems against cyber attacks," *IEEE Transactions on Automatic Control*, 2017.

[48] H. C. Lefevre, *The fiber-optic gyroscope*, 2022.

[49] S. Liu, X. Cheng, H. Yang, Y. Shu, X. Weng, P. Guo, K. C. Zeng, G. Wang, and Y. Yang, "Stars can tell: a robust method to defend against GPS spoofing attacks using off-the-shelf chipset," in *Proceedings of the USENIX Security Symposium*, 2021.

[50] Loiter Mode, https://tinyurl.com/3dkjsjpx, 2023.

[51] Magnetometer Measurement Weight, https://tinyurl.com/m4355evz, 2023.

[52] J. E. Martin, V. Saul, D. Novick, and D. Allen, "Assessing the Vulnerability of Unmanned Aircraft Systems to Directed Acoustic Energy," Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), Tech. Rep., 2020.

[53] M. Martinez-Diaz, J. Fierrez-Aguilar, F. Alonso-Fernandez, J. Ortega-García, and J. Siguenza, "Hill-climbing and brute-force attacks on biometric systems: A case study in match-on-card fingerprint verification," in *Proceedings of the IEEE Annual International Carnahan Conference on Security Technology*, 2006.

[54] MAVLink, https://mavlink.io/en/, 2023.

[55] Message Signing, https://tinyurl.com/2ahap7tz, 2023.

[56] M. Motallebighomi, H. Sathaye, M. Singh, and A. Ranganathan, "Cryptography is not enough: Relay attacks on authenticated GNSS signals," *arXiv preprint arXiv:2204.11641*, 2022.

[57] MPU-6000 Datasheet, https://tinyurl.com/mr2kv8mr, 2023.

[58] MultiWii, https://github.com/multiwii/multiwii-firmware, 2023.

[59] 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 IEEE Symposium on Security and Privacy (S&P)*, 2021.

[60] J. Noh, Y. Kwon, Y. Son, H. Shin, D. Kim, J. Choi, and Y. Kim, "Tractor beam: Safe-hijacking of consumer drones with adaptive GPS spoofing," *ACM Transactions on Privacy and Security (TOPS)*, 2019.

[61] M. Nowicki, J. Wietrzykowski, and P. Skrzypczyński, "Simplicity or flexibility? Complementary Filter vs. EKF for orientation estimation on mobile devices," in *Proceedings of the IEEE 2nd International Conference on Cybernetics (CYBCONF)*, 2015.

[62] Paparazzi Fail-safe Mechanisms, https://tinyurl.com/ypt6a9pp, 2023.

[63] Paparazzi Sensor Configurations, https://tinyurl.com/39zcjdpa, 2023.

[64] Paparazzi UAS, https://github.com/paparazzi/paparazzi/, 2023.

[65] Pixhawk Series, https://tinyurl.com/48hydrdj, 2023.

[66] PX4, https://px4.io/, 2023.

[67] PX4 Attitude Fail-safe, https://tinyurl.com/mvukaasn, 2023.

[68] PX4 Community, https://discuss.px4.io, 2023.

[69] PX4 EKF Switching, https://tinyurl.com/yma4n26x, 2023.

[70] PX4 Estimation and Control Library, https://github.com/PX4/PX4-ECL, 2023.

[71] PX4 Fail-safe Mechanisms, https://tinyurl.com/yc63vbpy, 2023.

[72] PX4 Filtering Pipeline, https://tinyurl.com/bdfe5p95, 2023.

[73] PX4 Flight Log Analysis, https://tinyurl.com/9ja47bhc, 2023.

[74] PX4 Flight Log with Acoustic Noises, https://tinyurl.com/k2y3965h, 2023.

[75] PX4 Gaussian Sum Filter, https://tinyurl.com/ukf9azyb, 2023.

[76] PX4 Harmonic Notch Filters, https://tinyurl.com/3k2fb9ur, 2023.

[77] PX4 Log Database, https://review.px4.io, 2023.

[78] PX4 Parachute Ejection, https://tinyurl.com/yt3tzsf5, 2023.

[79] PX4 Sensor Configurations, https://tinyurl.com/v7s3a574, 2023.

[80] Pymavlink, https://pypi.org/project/pymavlink/, 2023.

[81] RFD 900x Encrypted Telemetry, https://tinyurl.com/msuwjmw2, 2023.

[82] M. Robinson, "Knocking my neighbor's kid's cruddy drone offline," *DEF CON*, 2015.

[83] H. Sathaye, M. Strohmeier, V. Lenders, and A. Ranganathan, "An Experimental Study of GPS Spoofing and Takeover Attacks on UAVs," in *Proceedings of the USENIX Security Symposium*, 2022.

[84] A. Shaw, A. Al-Shamma'a, S. Wylie, and D. Toal, "Experimental investigations of electromagnetic wave propagation in seawater," in *Proceedings of the IEEE european microwave conference*, 2006.

[85] 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 USENIX Security Symposium*, 2015.

```
1 enum GYRO_CONFIG_BIT : uint8_t {
2         // Disable DLPF
3         FCHOICE_B_8KHZ_BYPASS_DLPF = Bit1 | Bit0};
4 enum ACCEL_CONFIG2_BIT : uint8_t {
5         // Disable DLPF
6         ACCEL_FCHOICE_B = Bit3};
```

Listing 1: PX4 disables digital low-pass filters (DLPF) inside ICM-20689 IMU sensor [79]. (See details in the datasheet [40])

[86] N. O. Tippenhauer, C. Pöpper, K. B. Rasmussen, and S. Capkun, "On the requirements for successful gps spoofing attacks," in *Proceedings of the ACM conference on Computer and communications security (CCS)*, 2011.

[87] K. W. Trantham and L. Janssen, "Multiple source interference with sound," *The Physics Teacher*, 2019.

[88] 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 IEEE European symposium on security and privacy (EuroS&P)*, 2017.

[89] 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 USENIX Security Symposium*, 2018.

[90] u-blox M8 Datasheet, https://tinyurl.com/2st9rphb, 2023.

[91] u-blox M9N Datasheet, https://tinyurl.com/y9vurn2b, 2023.

[92] Vibration Fail-safe, https://tinyurl.com/35nkhchw, 2023.

[93] Z. Wang, K. Wang, B. Yang, S. Li, and A. Pan, "Sonic gun to smart devices: Your devices lose control under ultrasound/sound," *Black Hat USA*, 2017.

[94] 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 IEEE Symposium on Security and Privacy (S&P)*, 2020.

[95] K. Yang, S. Mohan, Y. Kwon, H. Lee, and C. H. Kim, "Poster: Automated Discovery of Sensor Spoofing Attacks on Robotic Vehicles," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2022.

[96] Z. Yu and Y. Guan, "A dynamic en-route scheme for filtering false data injection in wireless sensor networks," in *Proceedings of the international conference on Embedded networked sensor systems*, 2005.

[97] K. C. Zeng, S. Liu, Y. Shu, D. Wang, H. Li, Y. Dou, G. Wang, and Y. Yang, "All your GPS are belong to us: Towards stealthy manipulation of road navigation systems," in *Proceedings of the USENIX Security Symposium*, 2018.

Appendix A.
RV Software's Sensor Parameter Settings

Each RV control software differently configures parameters inside IMUs (gyroscopes and accelerometers), which affects attacks' success or failure. In particular, PX4 disables digital low-pass filters (DLPF) if a sensor provides "turning on/off" options. Otherwise, PX4 leverages the highest level of cutoff frequency for DLPF to invalidate the functionality of DLPF [72] (Lines 3 and 6 in Listing 1). On the contrary, ArduPilot and Paparazzi turn on DLPF inside IMUs, as shown in Listing 2 and Listing 3.

```
1 // Turn on DLPF
2 config |= BITS_DLPF_CFG_188HZ;
3 _register_write(MPUREG_CONFIG, config, true);
```

Listing 2: ArduPilot leverages digital low-pass filters (DLPF) inside ICM-20689 IMU sensor [13]. (See details in the datasheet [40])

```
1 #define IMU_MPU_LOWPASS_FILTER MPU60X0_DLPF_42HZ
2 ...
3 // Turn on DLPF
4 imu_mpu_spi.mpu.config.dlpf_cfg = IMU_MPU_LOWPASS_FILTER;
```

Listing 3: Paparazzi leverages digital low-pass filters (DLPF) inside MPU-6000 IMU sensor [63]. (See details in the datasheet [57])

Appendix B.
Experiment Method and Equipment

Figure 21 and Table 5 shows details of experiment methods and equipment in this paper.

GNSS Spoofing. We generate fake GPS signals through (*i*) GPS-SDR-SIM [36], an open-sourced GPS signal generation software, and (*ii*) a HackRF One software-defined radio (SDR) platform [37] to transmit the generated fake GPS signals into the target RVs. We select, as a target RV, Pixhawk 4 mini microcontroller equipped with two IMUs, u-blox M8N GNSS receiver, and PX4 flight control software (*v.1.13.0*). We do not modify any code lines and configuration parameters of the PX4 software. To make an optimal environment from the attacker's perspective, we make the distance between the GNSS receiver and GPS spoofer less than one meter. Further, we do not move physical positions of the target RV and GPS spoofer while conducting experiments.

We choose a rural area to verify the feasibility of the GPS spoofing attacks because such an open space is normal flight environments for drones in the air. The rural area is an optimal location for the GNSS receiver because there are no multipath issues in the environment [39].

Acoustic Noise Injection. The distance between the speaker and Pixhawk boards are around 15 cm. Further, we make the Pixhawk boards' IMUs directly face the acoustic noises (i.e., we remove the Pixhawk boards' shielding), as shown in Figure 21-(b).

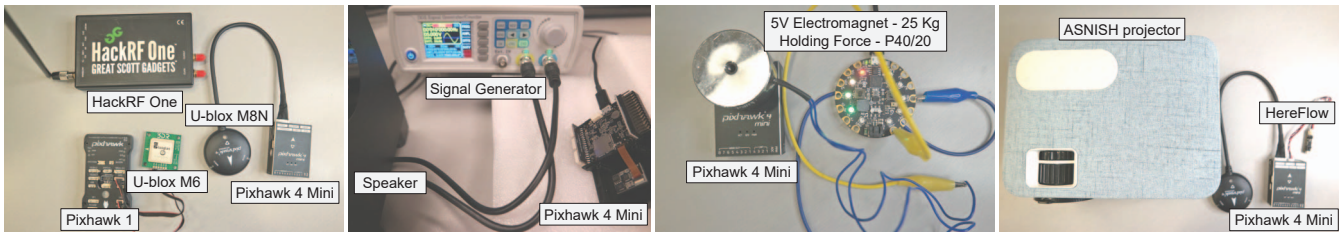
EMI Injection on Magnetometers. We place the electromagnetic generator on top of Pixhawk boards, as shown in Figure 21-(c). The generated electromagnetic field's effective range is less than 30 cm.

Appendix C.
Required Prerequisites (PX4)

Table 6 shows the required prerequisites to achieve, as the attack's goal, "physical crash".

Appendix D.
Conditions to Trigger Fail-safe Logic

As shown in Figure 22, ArduPilot does not specify a specific order of fail-safe logic. Instead, ArduPilot si-



(a) GPS spoofing (b) Acoustic noise injection (c) EMI injection on mag (d) Optical flow spoofing

Figure 21: Equipment for conducting attacks.

TABLE 5: Details of experiment methods and equipment

Attack	Experiment method		Details of experiment equipment
	Real world	Simulation	
GNSS spoofing (d)	✓	✓	- RV controller: Pixhawk 1 and Pixhawk 4 Mini - GPS spoofer (hardware): HackRF One - GPS message generator (software): GPS-SDR-SIM - GPS receiver: u-blox M6 - GNSS receiver: u-blox M8N
GNSS spoofing (w)	✗	✓	Gazebo simulator v.11.12.0
GNSS jamming (d)	✗	✓	Gazebo simulator v.11.12.0
GNSS jamming (w)	✗	✓	Gazebo simulator v.11.12.0
Acoustic noise (d)	✓	✓	- RV controller: Pixhawk 1 and Pixhawk 4 Mini - Signal generator: Koolertron 30MHz DDS Signal Generator - Amplifier: TDA8932 - DC Power: LRS-100-24 - Speaker: 2425T 25 Khz sound transducers
Acoustic noise (w)	✗	✓	Gazebo simulator v.11.12.0
EMI injection on mag (d)	✓	✓	- RV controller: Pixhawk 4 Mini - Electromagnetic generator: 5V Electromagnet - 25 Kg Holding Force - P40/20
EMI injection on mag (w)	✓	✓	- RV controller: Pixhawk 4 Mini with u-blox M8N - Electromagnetic generator: 5V Electromagnet - 25 Kg Holding Force - P40/20
EMI injection on SPI (d)	✗	✓	Gazebo simulator v.11.12.0
EMI injection on SPI (w)	✗	✓	Gazebo simulator v.11.12.0
Optical flow spoofing (d)	✓	✓	- RV controller: Pixhawk 1 and Pixhawk 4 Mini - Hex HereFlow Optical Flow Sensor - ASNISH movie projector
Optical flow spoofing (w)	✓	✓	- RV controller: Pixhawk 1 and Pixhawk 4 Mini - Hex HereFlow Optical Flow Sensor - ASNISH movie projector

TABLE 6: Summary of required prerequisites to achieve, as the attack's goal, "Physical crash" in PX4 and how many real-world RV configurations satisfy the required prerequisites.

	Attack prerequisites that make the attack successful / all prerequisites related to the attack						How many RV configurations satisfy prerequisites?	
	(C_1) Type of RV$^{\alpha}$	(C_2) Sensor configuration	(C_3) Software version	(C_4) Flight mode A: Autonomous M: Manual	(C_5) Configuration parameter	(C_6) Environmental condition	C_1-C_2	C_1-C_6
GNSS spoofing (d)$^{\beta}$	N/A	N/A	N/A	N/A	N/A	N/A	0%	0%
GNSS spoofing (w)	D, R	3/6	*v.1.8.0 - v.1.14.0*	A	4/36	Downtown/rural	91.06%	60.98%
GNSS jamming (d)	N/A	N/A	N/A	N/A	N/A	N/A	0%	0%
GNSS jamming (w)	N/A	N/A	N/A	N/A	N/A	N/A	0%	0%
Acoustic noise (d)	D, S	3/3	*v.1.8.0 - v.1.14.0*	A	N/A	N/A	98.70%	60.93%
Acoustic noise (w)	D, S	3/3	*v.1.8.0 - v.1.14.0*	A	N/A	N/A	98.70%	60.93%
EMI Injection on mag (d)	N/A	N/A	N/A	N/A	N/A	N/A	0%	0%
EMI Injection on mag (w)	N/A	N/A	N/A	N/A	N/A	N/A	0%	0%
EMI Injection on SPI (d)	D, R, S	1/2	*v.1.8.0 - v.1.14.0*	A	1/2	N/A	99.98%	60.98%
EMI Injection on SPI (w)	D, R, S	1/2	*v.1.8.0 - v.1.14.0*	A	1/2	N/A	99.98%	60.98%
Optical flow spoofing (d)	D	1/2	*v.1.8.0 - v.1.14.0*	A	1/9	N/A	10.11%	0.64%
Optical flow spoofing (w)	D	1/2	*v.1.8.0 - v.1.14.0*	A	1/9	N/A	10.11%	0.64%

$^{\alpha}$ D: Drone, R: Rover, S: Submarine $^{\beta}$ d: Demonstrated attack, w: Worst-case attack

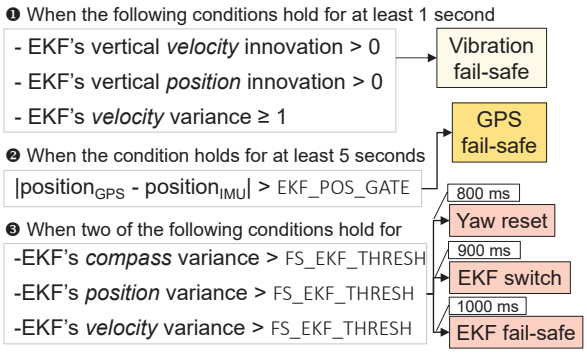


Figure 22: Illustration of conditions to trigger fail-safe logic in ArduPilot [15].

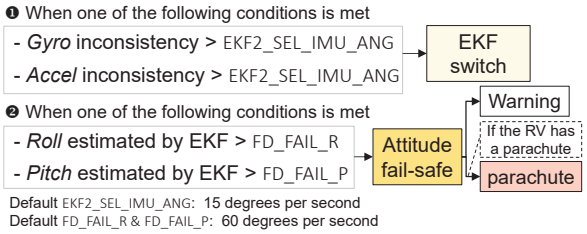


Figure 23: Illustration of conditions to trigger fail-safe logic in PX4 [67], [69].

multaneously checks conditions to trigger fail-safe logic every 100 millisecond (ms) [15]. Attacks (e.g., acoustic noise injection) almost immediately trigger the vibration fail-safe logic because the conditions for the fail-safe logic are not strict. Unfortunately, the vibration fail-safe logic does not appropriately address the attacks, as explained in Section 9.3.1. On the contrary, EKF fail-safe logic can prevent RVs from physically crashing on the ground by changing the flight mode to a manually-controlled flight mode. Yet, EKF fail-safe is the last stage of the fail-safe logic. Thus, it requires the strictest conditions.

The attacks easily trigger PX4's EKF switching algorithm because the conditions are not strict, as shown in Figure 23. Yet, unfortunately, worst-case attacks disturb all homogeneous sensors (e.g., injected acoustic noises disturb all IMUs). Thus, PX4's EKF switching cannot address the attacks. On the contrary, PX4's conditions for attitude fail-safe logic are too strict (60 degree errors per second in roll or pitch angles). Thus, RVs physically crash on the ground before PX4 triggers attitude fail-safe logic, as explained in Section 9.3.1.

Appendix E. Experimental Setup in Case Study 1

We follow the hardware setup for generating acoustic noises explained in [93] because it requires the most affordable commodity hardware among [43], [85], [88], [89], [93].

We obtain the following experimental conditions explained in research papers [43], [85]: (1) The RV type is a drone. (2) Sensor configurations consist mainly of two types. A DIY drone is equipped with a single IMU [85]. Another type of a drone leverages a Pixhawk 2 board that is equipped with two different IMUs [43]. (3) The tested RV software packages are MultiWii [58], ArduPilot [4], and PX4 [66].

We exclude the DIY drone, which is equipped with a single IMU, from our evaluations. The reason is that we find only 2.37% RV logs (712/30,059) leveraging a single IMU. It means that leveraging a single IMU rarely occurs in the real world. To conduct the attacks, we use Pixhawk 1 and Pixhawk 4 Mini boards [65] that are equipped with two different IMUs. We also exclude MultiWii software from our evaluations because (*i*) MultiWii has been discontinued since 2016; (*ii*) this software does not satisfy a commercial level of RV control software (e.g., ArduPilot and PX4), as shown in Figure 6; and (*iii*) we cannot find any commercial RV using MultiWii.

Unfortunately, we still do not know the following experimental conditions: software version, flight mode, and configuration parameter settings. To address this, we select (*i*) the latest versions of RV control software (ArduPilot *v.4.3.5* and PX4 *v.1.13.0*), (*ii*) `Loiter` flight mode [50], and (*iii*) default configuration parameters.

Appendix F. Experimental Setup in Case Study 2

We attempt to replicate electromagnetic interference (EMI) attacks on ArduPilot and PX4 simulators because we do not have a powerful EMI generator leveraged by the research paper [42]. We note that EMI attacks on the simulators are still reasonable for the two main reasons. First, we cannot expect that RV developers have access to such a powerful EMI generator. Second, we can still simulate the impacts of EMI attacks (i.e., corrupting data on SPI/I2C buses) on software-in-the-loop (SITL) simulators.

Experimental Conditions. We obtain the following experimental conditions explained in the research paper [42]: (1) The RV types are drones consisting of Arduino, Pixhawk 4, or DJI. (2) The tested RV control software packages are MultiWii [58] and PX4 [66]. (3) Flight mode is `Loiter` [50].

We exclude MultiWii software from our evaluations due to the reasons explained in Section 9.3.1. We also exclude Arduino and DJI controller boards because (*i*) we cannot find any commercial RV using Arduino, and (*ii*) DJI's control software is not an open-source project. Thus, we cannot conduct a root cause analysis.

Unfortunately, we still do not know, as experimental conditions, the software version and configuration parameters. To address this, we select (*i*) the latest versions of RV control software (ArduPilot *v.4.3.5* and PX4 *v.1.13.0*) and (*ii*) default configuration parameters.

Appendix G.
Meta-Review

The following meta-review was prepared by the program committee for the 2024 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

G.1. Summary

This paper addresses an important research problem in CPS security, exploitability of existing robotic vehicles using sensor attacks as the attack vector while examining the physical state as the outcome. To do so, it presents a systematic study on the manifestation of 12 types of sensor attacks on various RV platforms. The hardness of the attack is characterized using two distinct metrics: the prerequisites needed to carry out these attacks and the real-world prevalence of the prerequisites that make a particular attack feasible. This paper introduces RVPROBER, an automation framework for analyzing the prerequisites of attacks. RVPROBER identified that, on average, 4.4 prerequisites are necessary for the 12 sensor attacks, underscoring the tendency of prior research to overlook crucial details essential for executing these attacks.

G.2. Scientific Contributions

- Independent Confirmation of Important Results with Limited Prior Research
- Creates a New Tool to Enable Future Science
- Provides a Valuable Step Forward in an Established Field

G.3. Reasons for Acceptance

1) The work is timely. Though most of the sensor attacks have been previously presented, this work confirms them once more, and it is great that the authors put all these attacks together.
2) Inspired by the idea of attacks affecting robotic vehicles (RVs), they created a new and better methodology to implement an automation framework that measures the attack hardness by considering both the number of prerequisites required to achieve the attack objective and the probability that these prerequisites exist in a typical real-world RV usage scenario, which counts as a new tool to enable future science.
3) Sensor/Analog attacks represent a large class of emerging threat vectors, but how these threats will manifest in the system as well as the physical state is not well understood right now. This work provided a valuable step forward by deploying successful sensor attacks that require satisfying specific prerequisites when examining 12 well-known sensor attacks. Furthermore, the threat systematization is a good start in laying the foundation for defense.