PhyFuzz: Detecting Sensor Vulnerabilities with Physical Signal Fuzzing

Zhicong Zheng*
USSLab, Zhejiang University
Hangzhou, China
zheng_zhicong@zju.edu.cn

Jinghui Wu*
USSLab, Zhejiang University
Hangzhou, China
jinghuiwu@zju.edu.cn

Shilin Xiao*
USSLab, Zhejiang University
Hangzhou, China
xshlin@zju.edu.cn

Yanze Ren
USSLab, Zhejiang University
Hangzhou, China
yzren@zju.edu.cn

Chen Yan†
USSLab, Zhejiang University
Hangzhou, China
yanchen@zju.edu.cn

Xiaoyu Ji
USSLab, Zhejiang University
Hangzhou, China
xji@zju.edu.cn

Wenyuan Xu
USSLab, Zhejiang University
Hangzhou, China
wyxu@zju.edu.cn

***Abstract*—Sensor vulnerabilities can be exploited by physical signal attacks to cause erroneous sensor measurements, endangering systems that rely on sensors to make critical decisions. While hundreds of existing studies have discovered numerous sensor vulnerabilities, they are all driven by manual expert analysis and require a time-consuming process of trial and error. The absence of automated approaches to assist in the detection of sensor vulnerabilities has posed a major roadblock to bridging the gap between sensor security research and industrial applications. In this paper, we propose PhyFuzz, a new *physical signal fuzzing* paradigm that relies on physical testing signals to detect existing and potentially new types of sensor vulnerabilities without humans in the loop. To cope with the unprecedented challenges of fuzzing with physical signals, such as the infinite searching space of signal parameters and the black-box design of diverse sensor hardware, we design a unique fuzzing algorithm that enables efficient testing signal construction and effective feature discretization for sensor vulnerability identification and assessment. We implement PhyFuzz as a prototype that can support fuzz testing with acoustic, laser, and electromagnetic signals. Our experiment shows that it can identify 46 vulnerabilities on 13 sensors of 9 different types, including 6 undisclosed cases.**

I. Introduction

Sensors are devices that can convert the physical world into electrical signals. They are essential in cyber-physical systems (CPS) and have been widely adopted in intelligent applications [1]. Over the last decade, security researchers have discovered various sensor vulnerabilities that enable attacks based on malicious physical signals such as sound, electromagnetic (EM) waves, and laser [2]. These attacks can disrupt or falsify sensor measurements and cause malfunction, manipulation, or even damage to other systems that rely on sensors [3]–[7].

Detecting sensor vulnerabilities, including existing and new instances on a large number of sensors, is fundamental to making trillions of sensors [8] more trustworthy. However, compared with the academic and industrial achievements in detecting software and system vulnerabilities, we are still in an early stage with sensors. Different from vulnerabilities in the digital domain, sensor vulnerabilities originate from hardware defects and are exploited by physical signals. For example, studies have shown that signals outside the designed input signal type or range of microphones, such as a modulated laser [9], EM wave [10], or ultrasound [11], can be converted into illusive human voices due to unexpected photoacoustic effects, EM coupling, and nonlinearity of amplifiers. These vulnerabilities are difficult for security practitioners to identify by analyzing sensor firmware, circuit, or other design documents following existing security procedures. As a result, all existing research has manually conducted exhaustive physical experiments to discover new sensor vulnerabilities, which heavily depend on expert knowledge and are time-consuming. In addition, existing practices are difficult to scale to other types of sensors due to divergent hardware structures and working principles. To the best of our knowledge, there is no automated,expert-knowledge-independent, and generalized method to assist in the detection of sensor vulnerabilities, impeding sensor security research and industrial security testing.

In this paper, we design PhyFuzz, the first fuzz testing approach based on physical signals to automatically detect existing and potentially new sensor vulnerabilities. The basic strategy is to generate various unexpected physical signals as fuzz testing input to a sensor under test (SUT) and then monitor for exceptions in its output to reveal vulnerabilities. However, due to the distinct nature of sensor vulnerabilities compared with those of software systems, fuzzing a sensor requires solving several unprecedented challenges.

Challenge 1: How to efficiently generate physical fuzzing signals? Since the nature of sensor vulnerabilities suggests that sensors may potentially respond to any unexpected physical stimuli, there is no restraint on physical signals, such as

*These authors contributed equally to this work
†Chen Yan is the corresponding author

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.240029
www.ndss-symposium.org

the length, format, modality, etc. Unlike the digital input in traditional fuzzing, the input space of physical signal fuzzing is continuous and infinite.

Challenge 2: How to detect vulnerabilities with limited feedback information? Traditional feedback-guided fuzzing method relies on code coverage or a state machine to drive case generation and monitors for resource leaks or crashes to reveal vulnerabilities [12]. However, such information is unavailable in sensors that are often treated as a black box. It is difficult to access sensor runtime information apart from its input signals and output measurement readings.

To address these challenges, we propose a comprehensive solution comprising five key technical components: 1) a signal construction set that reduces the input searching space; 2) a feature discretization algorithm that transforms continuous sensor output into discrete feature coverage, thereby promoting efficient exploration of diverse vulnerabilities; 3) an integrated hardware and software platform capable of generating physical signals and communicating with the SUT; 4) enhanced fuzzing techniques specifically optimized for `PhyFuzz`; 5) an automated bug indicator to analyze fuzzing result and quantify vulnerability impacts.

In this paper, we implemented `PhyFuzz` on a prototype that can support fuzz testing with acoustic, laser, and electromagnetic signals. We validated the effectiveness of `PhyFuzz` on 13 sensors of 9 different types and found that: 1) It can reliably detect a wide range of known vulnerabilities reported in existing studies. 2) It can detect more numbers and types of sensor vulnerabilities within the same time budget compared with manual analysis, such as sweep testing. 3) It can discover indications of new types of vulnerabilities and assist in further expert analysis.

Our contribution can be summarized as follows:

- We propose the first physical signal fuzzing paradigm that generates physical testing signals to detect sensor vulnerabilities without requiring manual trial and error.
- We design a fuzzing algorithm that minimizes the input space of physical test signals with a signal construction set and extracts the discrete feature coverage from the sensor output as feedback to guide black-box sensor fuzzing.
- We implement `PhyFuzz` with three physical signal modalities on a prototype. Our experiment shows that it can identify 46 vulnerabilities on 13 sensors of 9 different types, including 6 undisclosed cases.

II. Background

A. Sensor Vulnerability

Definition. *Sensor vulnerability* refers to hardware defects arising from the non-ideal characteristics of materials, electrical components, mechanical structures, and other factors during the design and manufacturing processes of sensors. These vulnerabilities can be exploited by attackers to carry out transduction attacks [2], resulting in the generation of false sensor readings. For example, in the case of a microphone sensor, an attacker may exploit the nonlinearity of its amplifier to inject ultrasonic signals [11], the photoacoustic effect of its diaphragm to inject laser signals [9], or the electromagnetic coupling of its internal wiring to inject electromagnetic signals [10]. These attacks can cause the microphone to produce false audio outputs without the presence of audible audio inputs. The discovery of such vulnerabilities poses a significant threat to CPS. Since CPS relies heavily on sensors to perceive and interact with the physical environment, sensors represent a broad attack surface due to their direct exposure to external physical stimuli and their inability to distinguish between legitimate and malicious signals. Therefore, identifying and mitigating sensor vulnerabilities is critical to ensuring the security of CPS, particularly in safety-critical applications such as unmanned aerial vehicles (UAVs) and robotics.

Vulnerability Detection vs. Performance Test. There is a fundamental distinction between sensor performance testing and sensor vulnerability detection. Performance testing aims to evaluate a sensor's performance under its intended operating conditions, typically assessing metrics such as sensitivity, linearity, accuracy, and precision [13]. These tests are conducted in accordance with established standards or specifications [14]. In contrast, vulnerability detection serves as a complementary approach designed to uncover abnormal sensor behavior under unintended conditions, specifically when exposed to malicious signals such as acoustic, optical, or electromagnetic signals. Moreover, while sensors are generally subject to electromagnetic compatibility (EMC) testing, existing research has highlighted limitations in these tests. In particular, the test signals used in EMC assessments suffer from limited frequency range, signal strength, and diversity in signal construction to effectively reveal electromagnetic-related vulnerabilities [15]. Consequently, we argue that there is a critical need to explore a comprehensive method for sensor vulnerability detection.

B. Fuzzing

Fuzzing, also known as fuzz testing, has been proven effective in detecting vulnerabilities in numerous applications, especially in software testing. A fuzzing process typically includes case generation and execution monitoring [12]. During case generation, a generator, which can be a set of pre-defined grammars [16], [17], a data-based neural network [18], or a feedback-guided mutation unit [19], [20], will iteratively generate a number of test cases as input. Execution monitoring tracks the program's runtime information with each input, reports the discovered bugs, and provides feedback to the generator. Traditional software fuzzing typically adopts the coverage of code or logical paths triggered under the test as feedback. Based on accessible knowledge of the target object, fuzzing can be classified into three types: black-box, grey-box, and white-box fuzzing [12]. Black-box fuzzing can only utilize the input and output of the program, while white-box fuzzing can obtain all internal states during execution. Grey-box fuzzing can access partial internal information via a monitor tool or side channel.

While recent work has extended fuzzing to hardware and CPS targets as Tab. I shows, including robotic vehicles (RV)

TABLE I: Existing fuzzing methods targeted at sensors, hardware, and CPSs

Method	Target Application	Know.	Target Object	Feedback Mechanism
DeFUZZ [21]	Sensor	○	Firmware	Path Transition
RVFuzzer [22]	RV System	○	Control Algorithm	Control Instability
PGFUZZ [23]	RV System	○*	Control Algorithm	Safety Policy
DriveFuzz [24]	AD System	●	Control Algorithm	Driving Quality
SensorFuzz [25]	RV System	○	Control Algorithm	Resilience Score
Trippel et al. [26]	RTL Hardware	○	Hardware Design	Edge Coverage
Zhang et al. [27]	CPS	●	Control Algorithm	Deviation

○: White-box. ●: Black-box.
* Previous research [12] regards PGFUZZ as a black-box fuzzing method since the RV's source code is optional in its workflow. However, unlike the AD simulator, the RV simulator, like Gazebo [28], needs to be adapted and fine-tuned according to the source code.

and autonomous driving (AD) systems [23], [25], [27], these approaches primarily focus on the digital components of the target, such as firmware, control algorithms, and hardware designs [23], [25]–[27]. Consequently, existing fuzzing techniques remain confined to the digital domain, leaving the physical domain unexplored.

The key challenges of fuzzing sensors stem from two limitations of current methods. 1) The physical signal input to sensors lacks "grammar" or "semantic" constraints that help reduce the input space, making existing input analysis techniques such as dependency analysis, code fragment assembly, or language models [18], [29], [30] inapplicable. 2) The black-box sensor hardware provides severely limited feedback on the internal states, making existing fuzzing algorithms that rely on state machine inference [31], [32], crash signals [33], or predefined variable deviation [23], [27] inappropriate. These limitations of existing techniques motivate us to explore a new fuzzing paradigm in the language of physical signals.

III. Preliminary Investigation

In this section, we conduct a comprehensive investigation into two key challenges and derive insights, thereby aiding in the design of our physical fuzzing framework for sensor vulnerability detection.

A. Reducing Input Space (Challenge 1)

Unlike conventional software fuzzing, which operates over discrete input spaces, our physical fuzzing faces a continuous and diverse parameter space for physical signals. It results in a practically unbounded range of input signals. Nevertheless, a large body of existing research on transduction attacks suggests that attack signals should be carefully crafted to exploit sensor vulnerabilities and influence sensor outputs effectively. This indicates that we can filter out the vast majority of irrelevant input signals to reduce the time-consuming physical interaction process. Therefore, we review and summarize previous sensor vulnerabilities in Table II. Specifically, we categorize vulnerabilities based on the modality of the attack signal (acoustic and ultrasonic, laser, and electromagnetic) and identify each according to its underlying physical principle. We classify sensor vulnerabilities into two main types: *signal injection vulnerabilities*, which allow attackers to inject malicious signals, and *measurement shaping vulnerabilities*, which impact the sensor outputs. For each vulnerability, we analyze the corresponding signal requirements. Our analysis reveals that the analog signals capable of triggering sensor vulnerabilities are typically neither complex nor composed of elusive, composite patterns. On the contrary, most vulnerabilities are triggered by common, repeatable factors, as sensor component weaknesses often stem from overlooked or unavoidable physical principles. This observation aligns with established research in electromagnetics, which shows that emitting basic waveforms with varying parameters can effectively identify and analyze electromagnetic interference in complex systems [38]. Based on the summary in Table II, we now discuss several key signal requirements:

Frequency. Frequency is one of the most commonly reported signal parameters in existing research. Mechanical and electrical components in sensors often exhibit inherent resonant frequencies. For example, a mechanical wave at the resonant frequency can cause vibration in the MEMS transducers of microphones [39] or in the inertial units of motion sensors [4]. Similarly, an electromagnetic wave at the EM resonant frequency can induce a significant response in internal wiring [2]. Attackers can exploit these vulnerable frequencies to inject malicious signals and manipulate sensor measurements. Moreover, signals within specific frequency ranges can make transduction attacks more stealthy, for instance, using invisible lasers to interfere with lidars [40].

Amplitude. Due to the signal attenuation, shorter attack distances and higher signal amplitude can significantly increase the attack success rate (ASR). Besides, a high enough amplitude is also reported to induce the nonlinear effects of the sensor amplifier [5]. Besides, in most transduction attack scenarios, amplitude is a sensitive parameter of concern because of the constraint of the attacker's capabilities. Emitting a high-amplitude signal requires a non-portable device and carries risks of being detected, compelling attackers to resort to low-power attacks [41]. Therefore, the amplitude bound can act as a metric in evaluating the hardness of a transduction attack, which contributes to security analysis.

Amplitude Modulation. In addition to the fundamental sine wave, signal modulation is widely used to inject malicious signals into sensors. In previous research, amplitude modulation is the most frequently used modulation mode, especially in electromagnetic wave injection [37]. Besides, amplitude-modulated ultrasonic signals can also exploit the nonlinear effects of microphones, injecting malicious audio or backdoor triggers [11], [34].

Moreover, sensor vulnerabilities have an antecedent relationship. Yan et al. [2] categorized them as the *signal injection vulnerability* and *measurement shaping vulnerability*. Theoretically, the signal injection vulnerability pertains to how a physical signal can be injected into the sensor circuit, while the measurement shaping vulnerability refers to how the injected electrical signal influences the sensor output. For example, the resonance of the accelerometer introduces a high-frequency vibration (signal injection), which further causes signal aliasing in the digital AD converter (measurement

TABLE II: The attack signal requirements of previous transduction attacks.

Signal Modality	Vulnerability	Sensor	Attack Point	Signal Requirements	Response Features
Sound and Ultrasonic	Resonance* (**R.**) [4], [5]	Accelerometer & Gyroscope	Transducer	Resonant Frequency (**F**)	Fluctuating Measurements
	Asymmetric Saturation+ (**A.S.**) [5]		Amplifier	Resonant Frequency (**F**) High Amplitude (**A**)	Constant Offsets Harmonics
	Aliasing+ (**A.**) [5]		Filter	Resonant Frequency (**F**) Specific Frequency Point (**F**)	New Frequency Components Constant Offsets
	Frequency Leakage+ (**F.L.**)	Microphone	Filter	Boundary Frequency (**F**)	Unfiltered Frequency
	Nonlinear Effects+ (**N.E.**) [11], [34]		Amplifier	Ultrasonic (**F**) Amplitude Modulation (**M**)	Controllable Harmonic
Laser	Photoelectronic Effects* (**P.E**) [9], [35]	Microphone	Transducer	Wavelength (**F**) Amplitude Modulation (**M**)	Signal Injection
Electromagnetic	EM coupling* (**E.C.**) [36], [37]	Touchscreen	Excitation Circuit	Electromagnetic (**F**)	Capacitance Variation
		Image Sensor	Transmission Wire	Electromagnetic(**F**) Amplitude Modulation (**M**)	Changing Pixel Color Adding Image Noise
	Nonlinear Effects+ (**N.E.**) [10]	Microphone	Amplifier	Electromagnetic (**F**) Amplitude Modulation (**M**)	Controllable Harmonic

F: Frequency. **A**: Amplitude. **M**: Modulation mode. *: Signal Injection Vulnerability. +: Measurement Shaping Vulnerability.

shaping).

Summarizing these findings, we can propose an insight to reducing the input space by establishing a signal construction set consisting of prominent signal parameters for case generation, by which `PhyFuzz` can use as few cases as possible to excavate as many flaws as possible.

Insight 1: The physical signals that have the potential to induce sensor vulnerability tend to exhibit distinct characteristics. These characteristics can be used to construct a set of signals in order to reduce the input space.

B. Leveraging Output Information (Challenge 2)

A sensor consists of a series of interconnected, fine-grained components whose structure is integral to its working principle; any attempt to disassemble it would affect its output. This limits us to observing only the final output, making it a black-box model. Consequently, the black-box hardware of sensors motivates us to design a fuzzing feedback mechanism only based on sensor outputs. Two key questions are: 1) How to increase the coverage of vulnerability detection based on limited sensor output information? 2) How to increase the generality of testing among a diversity of sensor output formats?

First, based on related sensor security studies, we observe that different physical attacks on the same sensor produce distinct outputs due to their reliance on distinct physical principles. For instance, the measurements of accelerometers fluctuate under their resonant frequency but exhibit constant offsets under the output control attack [5]. Similarly, Fig. 1 also illustrates that the output of microphones varies under two distinct transduction attacks. (B) indicates the microphone leaks the ultrasonic signal and induces aliasing in the digital AD converter, and (C) indicates that an amplitude-modulated ultrasonic signal induces the nonlinear vulnerability of the microphone amplifier. Yan et al. [2] further discussed how vulnerabilities of different sensor components contribute to various measurements. This result suggests that the sensor output diversity can reflect the sensor vulnerability diversity, enabling comprehensive vulnerability exploration. By analogy to traditional software fuzzing, we define this concept as the **coverage** in the context of sensor vulnerability detection.

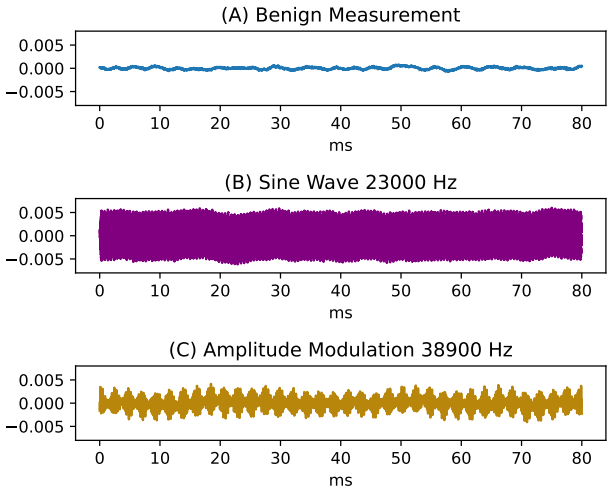


Fig. 1: The time-domain waveform of microphones output under frequency leakage and aliasing (B) and nonlinear effects (C) vulnerabilities.

However, the continuous and multi-format nature of raw sensor data still leaves a dilemma for designing available fuzzing feedback. For instance, mono microphones can output a fragment of time-series data, while color cameras can output a high-dimensional stream. To achieve generality, we adopt feature engineering to transform raw sensor output into feature-level information. Then, we utilize differential analysis [42] to normalize and discretize output variations across sensors.

Insight 2: Sensor output diversity implies sensor vulnerability diversity. With proper feature extraction and discretization, it can serve as feedback to guide fuzzing.

IV. System Design

Based on our insights in Sec.III, we propose `PhyFuzz`, which aims to fuzz various sensor vulnerabilities and verify their security in a black-box scenario. The entire pipeline is illustrated in Fig. 2 and can be divided into five modules. Briefly, the initial seed is generated with the *Signal Construction Set*, which contains the essential parameters for each test

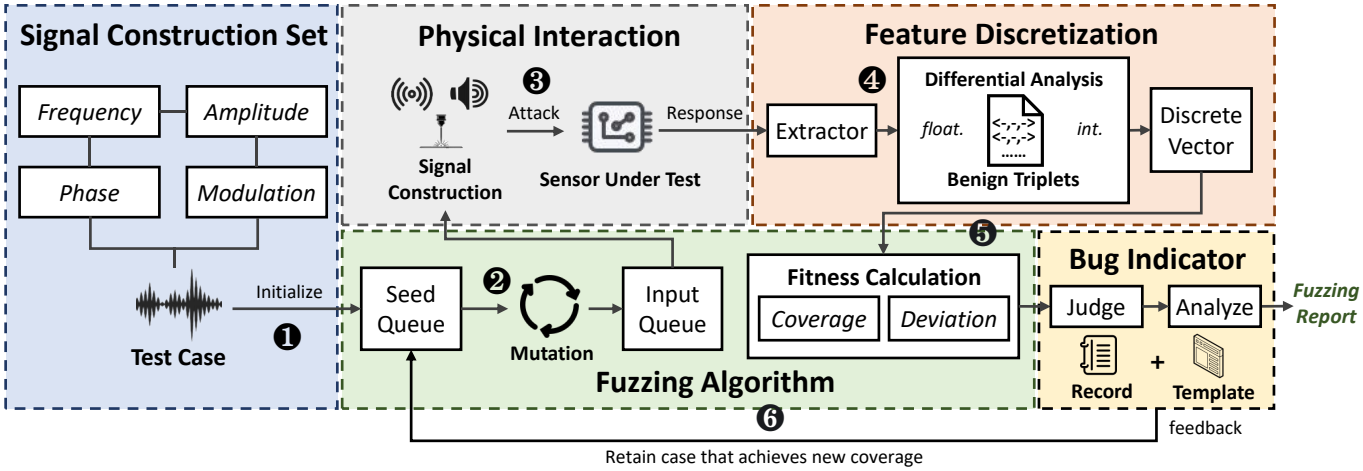


Fig. 2: The overview of PhyFuzz. ❶: Initialize the seed randomly with the signal construction set and append it to the queue. ❷: Pick out the best seed in the queue and mutate it several times. ❸: Emit the case signal to attack the SUT. ❹: Extract the response of SUT as features and convert them into a discrete vector with differential analysis. ❺: calculate the fitness with the coverage and deviation information. ❻: retain or eliminate a case according to the coverage changes and fruitless mutation times.

case. In each iteration, a seed is selected, mutated to create several cases by *Fuzzing Algorithm*, and transmitted to the sensor under test (SUT) through *Physical Interaction*. The response of SUT will be processed by *Feature Discretization* and then fed back to *Fuzzing Algorithm* for fitness calculation and case retention. After fuzzing, *Bug Indicator* will judge the potential sensor vulnerabilities from the result and provide a semi-automatic analysis.

A. Signal Construction Set

Sec. III-A presents an insight that identifying prominent signal parameters aids in reducing the input space. In this section, we define a signal construction set to reduce the input space of the physical domain, represented as a four-tuple $\{A, F, P, M\}$. Each element in the tuple represents amplitude, frequency, initial phase, and modulation mode, respectively. Previous investigations have highlighted the importance of frequency, amplitude, and amplitude modulation.

The frequency domain is the most critical search space due to its impact being agnostic for different sensors. In comparison, the impact of amplitude is of direct physical significance. Low amplitude tends to have a linear effect on sensor measurements, whereas high amplitude is more likely to induce nonlinear responses. Therefore, we use a $sigmoid$ function to distribute the amplitudes more towards the ends of the range, as shown in Equ. 1:

$$A' = g(x) = A_{\min} + (A_{\max} - A_{\min}) \cdot \frac{1}{1 + e^{-k(x-c)}} \quad (1)$$

where A_{min} and A_{max} is the lower and upper bounds of the amplitude. k and c are the hyperparameters of the $sigmoid$ function, representing the steepness and midpoint of the curve, respectively.

In addition, to construct a wider variety of physical signals and cover more potential sensor vulnerabilities, PhyFuzz also introduces additional parameters, including more modulation modes and variable phases.

Modulation Mode. In addition to amplitude modulation, we incorporate phase and frequency modulation in our algorithm to enhance our construction set. While the essence of all modulation modes is to transfer information with carrier signals, different modulation modes result in variations in signal shape and the derivative frequency components that are valuable to study.

Phase. As one of the three primary parameters of signals, phase describes the signal offset at the beginning point. Although previous work has not found that the initial phase can directly induce sensor vulnerability, dynamically modifying the phase of existing attack signals can help to manipulate sensor measurements. Therefore, in PhyFuzz, the phase will be activated by our mutator when a case is proven to be able to impact the sensor measurements.

Each test case $s(t)$ can be constructed according to Equ. 2. The base signal $m(t)$ is typically a standard sine wave with a fixed low frequency. Modulation parameters, k_p and k_f, serve as hyperparameters.

$$s(t) = \begin{cases} sine: & A'sin(2\pi Ft + P) \\ AM: & A'(m(t)+1)sin(2\pi Ft + P) \\ PM: & A'sin(2\pi Ft + k_p m(t) + P) \\ FM: & A'sin(2\pi Ft + k_f \int_0^t m(\tau)d\tau + P) \end{cases} \quad (2)$$

B. Physical Interaction

Compared with traditional software-level fuzzing, PhyFuzz features a unique and crucial module, physical interaction. The test case generated in the digital domain is transmitted into the physical world by the signal transmitter and then delivered to the SUT. PhyFuzz collects the SUT's

digital response (raw measurements) and feeds it to the next computing unit, completing the cross-domain process.

Signal Transmitter. A complete signal transmitter typically consists of a signal generator, an amplifier, and a transducer. For different signal modalities, the transducer varies; for example, ultrasonic signals use ultrasonic probes, and electromagnetic signals use antennas. In consideration of lightweight testing demands, `PhyFuzz` simplifies the signal transmitter with existing integrated equipment, which will be further introduced in our implementation.

Sensor Under Test. The goal of `PhyFuzz` is to find vulnerabilities of SUT in a physical environment without requiring expert knowledge. Although the SUTs are completely black-box for us, `PhyFuzz` can still be compatible with various SUTs due to their commonality in converting the physical sensing into the digital output. Therefore, `PhyFuzz` only requires collecting SUT measurements with an adapter board and the corresponding driver program rather than developing SUT-specific tracing instrumentation [26].

C. Feature Discretization

The insight proposed in Sec. III-B demonstrates that we can design a novel coverage metric to guide `PhyFuzz` for sensor vulnerability detection. In this section, we propose a feature discretization algorithm to effectively leverage the output information of SUTs and formulate the fuzzing feedback mechanism.

Specifically, our algorithm consists of two steps: feature extraction and differential analysis. We first transform the raw sensor data into feature-level representations by feature engineering. Then, differential analysis discretizes the continuous feature space by comparing test data and benign data, generating a discrete vector for each test case. By aggregating these discrete vectors into a set, we can construct a coverage metric. Notably, our notion of "coverage" differs from conventional metrics. In black-box sensor fuzzing, where internal states are unobservable, conventional state coverage can not be used. Instead, our "coverage" is derived from output variations, reflecting the diversity of vulnerabilities. Both approaches aim to guide exploration and avoid low-value trails.

The detailed algorithm is illustrated in Alg. 1. Initially, we extract a group of statistical features from a long period of benign measurements. For each feature, we take the maximum and minimum values and calculate the interval. This process builds a triplet $T_{ij} = \{Min, Max, Interval\}$ for each feature as the baseline. The feature categories we used are predefined by a feature template (detailed in Sec. IV-E).

For different categories of sensors, the feature triplet achieves sensor-agnosticity. Even for sensors with unique and complicated output formats, e.g., cameras, we can leverage the edge computing capability of data collector devices to reduce the output dimension by pre-extracting several features for acceleration.

In each test, we extract corresponding features from the measurement, compare them with the baseline triplet, and obtain the discrete vector. This vector not only indicates the potential sensor malfunction but also provides coverage information, guiding `PhyFuzz` towards exploring more diverse sensor vulnerabilities.

Algorithm 1 Feature discretization Algorithm

Require: A long period of benign measurements M_b, measurements in each test m_t, data dimension D, extracted feature category F

Ensure: The discrete vector of the test result V

for $i = 0$ to D **do**
  Split M_b according to m_t and get $Array_b$
  // Extract the statistical features of benign data as triplets
  for fea_j in F **do**
    $f_{ij}^B = Feature\_Extract(fea_j, Array_b[i])$;
    $T_{ij} = \{min(f_{ij}^B), max(f_{ij}^B), max(f_{ij}^B) - min(f_{ij}^B)\}$;
  end for
  // Extract the statistical features of a piece of test data
  for fea_j in F **do**
    $tf_{ij} = Feature\_Extract(fea_j, m_t[i])$;
    // Compare two features and assign discrete value
    if $f_{ij}^t > T_{ij}[0] \& f_{ij}^t < T_{ij}[1]$ **then**
      $v_{ij} = 0$;
    else if $f_{ij}^t > T_{ij}[1]$ **then**
      $v_{ij} = floor((T_{ij}[1] - f_{ij}^t)/T_{ij}[2])$;
    else
      $v_{ij} = -floor((T_{ij}[0] - f_{ij}^t)/T_{ij}[2])$;
    end if
  end for
end for
Concat all v_{ij} together and get V.

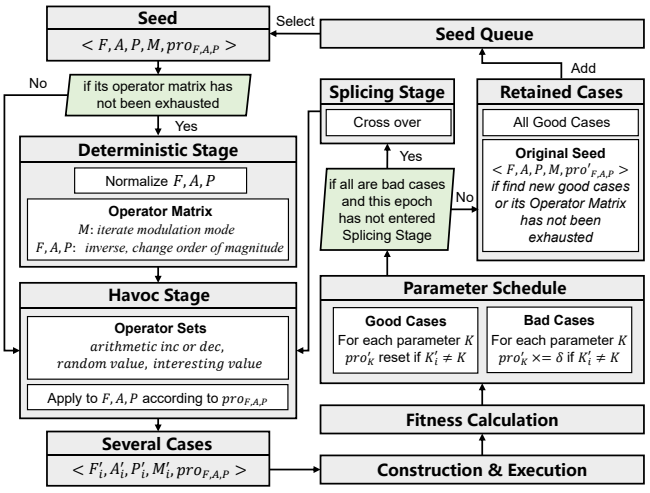


Fig. 3: The thorough pipeline of fuzzing algorithm.

D. Fuzzing Algorithm

In this section, we present a comprehensive overview of our entire fuzzing algorithm. Unlike traditional software fuzzing, physical fuzzing has unique features, encouraging us to im-

prove some techniques of case generation, fitness calculation, and case retention.

Case Generation. The first test case of PhyFuzz is randomly generated with the signal construction set (❶) and put into the seed queue. In each iteration, as shown in Fig. 3, a seed will be selected from the seed queue and mutated according to our mutation schedule, which adopts the three-stage mutation strategy (deterministic, havoc, and splicing stage) proposed in AFL [20]. Compared to traditional software fuzzing, PhyFuzz has improved in mainly two aspects: mutation operator and parameter schedule.

1) Mutation Operator determines the basic unit of seed mutation. The traditional fuzzing framework uses byte-level mutation operators, such as $bitflip$, $delete$, or $insert\ byte$. However, our seed consists of a group of parameters with real physical meanings rather than binary files, motivating PhyFuzz to improve the design mutation operator for different parameters. Specifically, among all four parameters, the modulation mode is discrete and finite, while the other hard-to-exhaust parameters have bound constraints. Hence, in the deterministic stage, we will maintain an operator matrix for each seed to exhaust all modulation modes while adjusting other parameters with the fixed arithmetic operation. In the havoc stage, we randomly choose $arithmetic$, $random$, and $interest\ value$ as our mutation operators to generate several test cases. If no coverage improvement occurs after physical interaction testing in a havoc turn, the $crossover$ operator will be employed to recombine parameters of those mutated cases in the splicing stage, and then re-enter the havoc stage again.

2) Parameter Schedule is designed to determine the importance of each signal element. This technique is derived from the byte schedule in a traditional fuzzing framework, with which the fuzzer can assign a higher mutation frequency to the more important bytes. Inspired by Ant Colony Optimization (ACO) [43], PhyFuzz uses mutation probability pro in the havoc stage to measure the importance of each parameter of the test case. Whenever a new case is generated, pro will be assigned with initial values and dynamically adjusted during its lifetime. Specifically, the initial values of F, A, P are assigned according to prior knowledge mentioned in Sec. IV-A. Whenever a seed is fed into the fuzzing loop if its generated case achieves a new coverage (noted as a good case), the probabilities of changed parameters $pro_K, K \in \{F, A, P\}$ will be reset to the initial value. Conversely, if the coverage does not increase (i.e., a bad case), the pro_K will be decreased by multiplying $\delta \in (0, 1)$. Detailed hyper-parameter settings are illustrated in Tab. V.

Fitness Calculation. During fuzzing, each test case will be constructed and emitted to the SUT (❸) and obtain a discrete vector, as we mentioned in Sec. IV-C. From the vector, PhyFuzz will compute a fitness value to measure the case's significance. Before each fuzzing epoch, PhyFuzz will select the seed with the highest fitness from the seed queue for mutation. As shown in Alg. 2 (❺), fitness is calculated by the discrete vector V and the global coverage array $\mathbb{C}$. Briefly, the fitness function consists of two items, as shown in *line 15* in Alg. 2. The first term quantifies how markedly the sensor response differs from the normal response, indicating potential vulnerabilities. The second term represents the minimum distance to all previously observed vectors in $\mathbb{C}$, which contains all discrete vectors of previous tests.

Algorithm 2 Fitness Calculation Algorithm

Require: The discrete vector of test signal V, the global coverage array $\mathbb{C}$, hyper parameters α and β
Ensure: The Fitness of the test signal $Fitness$; Whether achieving new coverage $new\_coverage$

```
1:  deviation = |V|;
2:  min_distance = ∞
3:  new_coverage = False
4:  for cov in ℂ do
5:      if cov is equal to V then
6:          min_distance = 0;
7:          Break the loop;
8:      end if
9:      distance = |V − cov|;
10:     min_distance = min(min_distance, distance);
11: end for
12: if min_distance! = 0 then
13:     new_coverage = True
14: end if
15: Fitness = α · deviation + β · min_distance;
16: Add V into ℂ.
```

Case Retention. In terms of the case retention strategy, all newly generated cases that achieve new coverage, i.e., good cases, will be retained in the seed queue(❻). In addition, if the original seed generates any good cases or its operator matrix has not been exhausted, it will also be retained.

E. Bug Indicator

In parallel to the fuzzing loop, PhyFuzz can record and process the bug cases by the bug indicator. It works in two steps: 1) determining whether a case induces the malfunction of the SUT, and 2) automatically conducting preliminary analysis on the bug case in order for humans to understand it better.

Judging Bugs. When the $deviation$ in Alg. 2 exceeds the pre-defined deviation threshold, the bug indicator automatically logs this case. The $deviation$ is computed as a relative measure of the sensor readings, representing a multiple (e.g., three times) of the sensor's baseline noise range to ensure applicability across various sensor types. In our evaluation, the threshold is empirically set at 2, which provides a good balance between effective anomaly detection and strong resistance to noise. For a few sensors with low precision or without measurable noise, the threshold is set as a small percentage of the full sensor output range. In practice, the threshold could be tuned according to the detection strategy—higher thresholds reduce false positives, whereas lower ones reduce false negatives. Upon completion of the entire fuzzing process, PhyFuzz will conduct replicate experiments on recorded

bugs to confirm these bugs for each SUT and exclude ambient noise interference.

Unlike traditional software fuzzing, physical sensor vulnerabilities cannot be located via error code fragments. Prior CPS fuzzing approaches only identified bugs according to sensor readings above or below the upper or lower boundary [27]. While recording which analog signal can induce sensor vulnerability suffices for typical users, we still hope PhyFuzz can help sensor developers and the security community in in-depth bug analysis.

Analyzing Bugs. The bug analysis hinges on systematic classification and characterization to yield concise and interpretable results. In Sec. III, we proposed to categorize the different types of sensor vulnerabilities via the variation of output features.

An intuitive idea is to employ machine learning to cluster or classify the bug list. However, due to the discrete and concealed nature of vulnerability triggers, there is a lack of sufficient and balanced data samples, hindering conventional ML algorithms' effectiveness. Consequently, we turned to empirical knowledge of sensor vulnerabilities. While the expert knowledge only encompasses previously discovered vulnerabilities and lacks foresight, it effectively filters well-documented cases, allowing analysts to concentrate on discovering and investigating novel vulnerability patterns.

As mentioned before, the signal injection vulnerabilities can be identified by the modality of the signal stimulation. As for the measurement shaping vulnerabilities, we find they possess similar feature variations under different signal stimulations, as they stem from the shaping of injected electrical signals. Therefore, we can create predefined feature templates of measurement shaping vulnerabilities to identify and categorize well-studied vulnerabilities, thus reducing the burden of manual review. For instance, most bug records of an accelerometer are caused by resonance and aliasing. Except for a few resonance frequency points, the response exhibits a significant change in standard deviation and almost no change in the average. Based on these observations, we propose a feature-matching algorithm to automatically analyze the human-readable but time-consuming record. Note that the use of heuristic templates does not contradict our claim of being expert-knowledge-independent, as relevant knowledge is embedded in the framework design rather than required from end-users during testing.

We present example templates for several well-studied measurement shaping sensor vulnerabilities in Tab. III, in which the feature categories will be used in Sec. IV-C. In this template, we select five time-domain features and five frequency-domain features to distinguish several well-studied sensor vulnerabilities in Tab. II. Specifically, aliasing may result in fluctuation or output bias (noted as $\mathbf{A.}_1$ and $\mathbf{A.}_2$), inducing the distinguish variation of **Std.** and **Avg.** [5]. Asymmetric saturation is caused by the intense signal injection and the asymmetric clipping of the amplifier, resulting in a constant shift and probably a slight fluctuation (corresponding to **Avg.** and **Std.**). Frequency-domain features can be used to differentiate similar vulnerabilities. Nonlinear effects will introduce the baseband signal, inducing the energy increase at baseband frequency (**E.B.**). Constant shift can also be captured by an increase in energy at zero frequency (**E.Z.**). Other features also enrich the representation capability of the template. To leverage these feature variations, we formulate a vector consisting of 0, 1, and -1 to depict the anticipated affected features of each vulnerability. Features marked as 1 indicate the primary characteristics of the vulnerabilities, while such vulnerabilities should not impact those marked as -1. Some less prominent affected features are denoted as 0.

TABLE III: The templates of several well-studied measurement shaping sensor vulnerabilities.

Vul.	Feature Template									
	Max.	**Min.**	**Avg.**	**Std.**	**RMS**	**E.D.**	**F.E.**	**F.C.**	**E.Z**	**E.B**
$\mathbf{A.}_1$	0	0	0	1	0	1	1	0	-1	0
$\mathbf{A.}_2$	1	1	1	-1	1	1	1	0	1	-1
A.S.	0	0	1	0	0	1	0	0	1	0
N.E.	0	0	0	1	0	1	0	1	-1	1
F.L.	1	1	0	1	1	1	1	1	0	-1

E.D.: Euclidean Distance. F.E: Frequency Entropy. F.C.: Frequency Center. E.Z.: Energy at Zero. E.B.: Energy at Baseband.

Note that these templates can only assist users in dealing with a large number of bug records. In some complex cases (e.g., when plural vulnerabilities are triggered simultaneously), manual analysis combined with injected signal parameters is still necessary, although our evaluation has shown that the small amount of overlap does not affect the ability of our method to find these vulnerabilities. In addition, it is important to clarify that measurement shaping is not always an absolute prerequisite. Certain injection methods, like laser signals, can be directly injected into microphones without specific measurement shaping.

The detailed pseudocode of these two steps is presented in Alg. 3. Finally, by following these two steps, we can confirm the bug record and promptly eliminate well-studied records. The remaining records will constitute a valuable repository of vulnerabilities that are more likely to represent new security threats, warranting further analysis by sensor security professionals.

V. Evaluation

A. Implementation

To systematically conduct PhyFuzz, we have developed and implemented a prototype comprising several signal transmitters, a sensor data collector, and an upper computer.

Signal Transmitter. Based on our comprehensive review of prior transduction attacks, we have selected three classical signal modalities for attacks: acoustic (sound and ultrasonic), laser, and electromagnetic. As shown in Fig. 4, we equip a JBL speaker for audible sound generation, a ViFa speaker for ultrasonic emission, a laser probe for laser stimulation, and a portable USRP B210 to generate electromagnetic waves. All these devices are commonly used in relevant research. In our implementation, the distance between the transmitters and the bare SUT is 10 cm, eliminating the need for additional amplifiers. This setting reflects practical attack constraints,

TABLE IV: The overview fuzzing result of PhyFuzz under 13 sensors of 9 different types.

Application	Sensor Type	Model	Manuf.	Output	S.R. (Hz)	Signal Injection Vulnerability			Measurements Shaping Vulnerability				
						R.	P.E.	E.C.	A.	A.S.	F.L.	N.E.	*N.C.*
RV System	Accelerometer	ADXL345	ADI	I2C/SPI	200	🔊		📶	🔊 📶	🔊 📶			
		LIS2DW12	STM	I2C/SPI	200	🔊		📶	🔊 📶	🔊 📶			
	Gyroscope	MPU6050	INVN	I2C	100	🔊			🔊	🔊			
		L3G4200D	STM	I2C/SPI	100	🔊			🔊	🔊			
Voice System	Microphone*	SPH0645	Knowles	I2S	48k		⚙		🔊		🔊	🔊	
		INMP441	INVN	I2S	48k		⚙	📶	🔊 📶	📶	🔊	🔊	
Vision System	Color sensor+	TCS3472	TAOS	I2C	200			📶	📶				
	Light sensor+	BH1750	ROHM	I2C	8			📶	📶	📶			
	CMOS Camera+	IMX219	Sony	CSI	30								
	CCD Camera+	1200TVL	Sony	USB	25			📶					📶
Environmental Monitoring	Pressure sensor	BMP180	Bosch	I2C	100		⚙	📶	📶	📶			
		BMP280	Bosch	I2C/SPI	10		⚙	📶	📶	📶			
	H&T sensor	SHT30	Sensirion	I2C	20			📶	📶	📶			

🔊: Sound and Ultrasonic Signal. ⚙: Laser Signal. 📶: Electromagnetic Signal.
*: No audible sound test for microphones. +: No laser test for optical sensors. *N.C.*: No Conclusion.

Algorithm 3 Bug Indicator Algorithm

Require: The raw bug record of PhyFuzz R_r, the bug threshold bt, the heuristic feature template $\mathbb{F}$, the parameter of the repeat test m and n ($m > n$), the matching threshold mt.

Ensure: The confirmed bug record R_c, The confirmed bug record excluded the well-studied vulnerabilities R_e.

// Repeat tests for m times to confirm the bugs
for $case$ in R_r) **do**
  $hit\_count = 0$
  for i in range (m) **do**
    $deviation = Repeat\_Test(case)$
    if $deviation > bt$ **then**
      $hit\_count+ = 1$
    end if
  end for
  if $hit\_count > n$ **then**
    Add $case$ into R_c
  end if
end for
// Exclude cases of well-studied vulnerabilities
for $case$ in R_c **do**
  for $template$ in $\mathbb{F}$ **do**
    $V_{ori} = case.V$
    $V_{mask} = V_{ori}.copy(); V_{mask}[template == -1] = 0$
    $key\_arr = V_{mask}[template == 1]$
    if $\frac{|V_{mask} - V_{ori}|}{|V_{ori}|} < mt$ and $np.all(key\_arr \neq 0)$ **then**
      continue
    end if
  end for
  Add $case$ into R_e
end for

as vulnerabilities requiring stronger signals at this distance would be unrealistic for real-world exploitation. If the user deems it necessary to add a power amplifier, it will not affect the working framework of PhyFuzz. Additionally, since the wavelength of the laser probe is fixed, the fuzzing variables of the laser test are simplified. Only DC signals and AM signals are validated, replacing the original modulation mode.

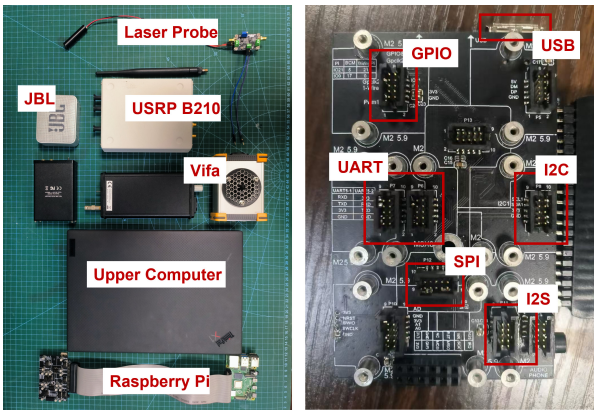


Fig. 4: The picture of our evaluation implementation. Left: the hardware devices (excluding some wires). Right: our adaptor board.

Other Components. For the sensor data collector, we employ a Raspberry Pi with a customizable adapter board, enabling compatibility with diverse SUT categories. A host PC will execute the fuzzing algorithm and coordinate the workings of other components.

Configurable Parameter. Our fuzzing framework includes several major hyperparameters, with default values documented in Tab. V as follows. The test duration is conservatively set to 3 seconds to ensure compatibility with low-sample-rate sensors while maintaining testing efficiency. Additional parameters, including the probabilities of mutation operators and modulation parameters, can be referred to in our code. The default settings for PhyFuzz's key parameters establish the boundaries of our search space and the practical constraints of our implementation.

B. Fuzzing Result

1) Overall Performance: To verify the performance of PhyFuzz, we conduct a series of evaluations with 13 different sensors of 9 categories representative of prevalent intelligent applications. With our bug indicator algorithm, we pre-process

TABLE V: The default value of major configurable parameters.

Process	Parameter	Value
Case Generation	Frequency range (sound)	20-20k Hz
	Frequency range (ultrasonic)	20k-48k Hz
	Frequency range (EM)	70M-6000M Hz
	Frequency range (laser)	20-20k Hz
	Attack duration	3 s
	Population of each iteration	8
	Mutation time	1-4
	Initial probability(F, A, P)	[0.8, 0.5, 0]
	Amplitude gain range	0.3-1.0
	Precision (sound, ultrasonic)	0.1 Hz
	Precision (EM)	0.01 Mhz
	Precision (laser)	0.1 Hz
Fitness Calculation	Deviation factor (α)	0.2
	Coverage factor (β)	0.8
Bug Indicator	Threshold of Deviation	2

the fuzzing records, reaffirm, and identify 46 vulnerabilities, as shown in Tab. IV.

Under the default settings and implementation outlined in Table V and Section V-A, PhyFuzz requires approximately 6-7 hours for 400 iterations in sound and ultrasonic tests, 2 hours for 200 iterations in laser tests, and 12-14 hours for 640 iterations in electromagnetic tests. Sensors with complex outputs, such as cameras, require more time for each iteration. Subsequent replicate experiments will also take additional time, depending on the number of bug records. Additionally, for low-sample-rate sensors, we dynamically extend attack durations to ensure data validity during vulnerability confirmation. We repeatedly conducted three fuzzing trials, where the initial seed inputs were randomly generated for each trial.

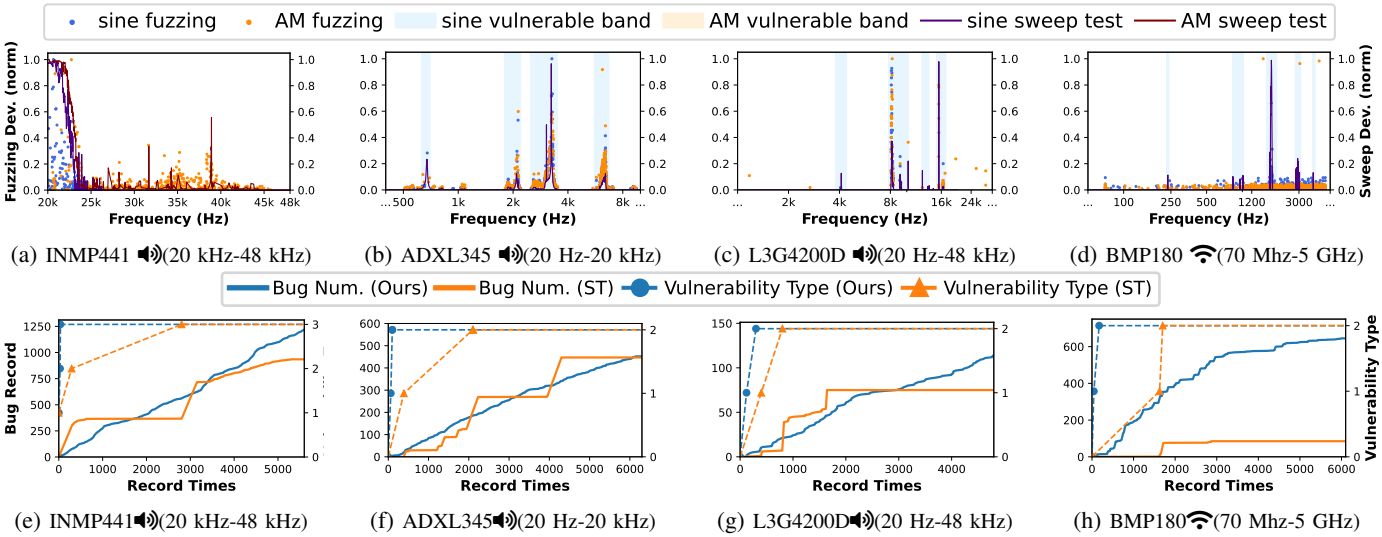
Tab. IV summarizes the vulnerabilities of 13 sensors that PhyFuzz found. Our results reveal that mechanical motion sensors show particular susceptibility to acoustic signals, while the electromagnetic signal demonstrates broad effectiveness across most sensor types. Specifically, PhyFuzz successfully reproduced known vulnerabilities in well-studied sensors, such as the ADXL345 and MPU6050, validating against prior research findings [5], [44]. Additionally, for other sensors that have not yet received attention, e.g., color and light sensors, PhyFuzz effectively demonstrated its adaptability by identifying **five** novel vulnerability classes from them. In particular, we identified a specific vulnerability in a 1200TVL camera but could not draw precise conclusions. Although similar vulnerabilities have been reported in CCD cameras [45], we cannot assert that they share the same underlying mechanism with CMOS cameras. Further discussion and analysis of these **six** vulnerabilities are provided in the Case Study and Appendix B.

We have reported all these vulnerabilities to the corresponding manufacturers via email. All have responded, and three engaged in further discussions for additional details. We present their responses in Appendix A. These results collectively demonstrate PhyFuzz's effectiveness in both reproducing known vulnerabilities and discovering novel ones. Complete fuzzing results are available in the supplementary materials. We will show some specific attack effects in subsequent case studies.

2) Compared with Sweep Test: Given that PhyFuzz is the first fuzzing framework specifically designed for sensor vulnerability identification, direct comparison with existing fuzzing approaches is not feasible. Therefore, we benchmark our framework against traditional sweep testing, which is widely adopted in sensor security research. Unlike the intelligent fuzzing strategy of PhyFuzz, sweep testing uses a series of signals with uniformly varying frequencies to identify the vulnerable points of the SUT. To align with PhyFuzz, we follow the common setup: 1) The default waveform is a sine wave with maximum amplitude and an initial phase of 0. 2) Crucial parameters, including frequency range and attack duration, maintain consistency with PhyFuzz. 3) The frequency step, which directly controls the searching precision, is adjusted to match the testing time of PhyFuzz, ensuring a fair comparison of the efficiency. Through detailed experiments, we compare the performance of PhyFuzz and sweep testing, verifying the superior performance of PhyFuzz in both comprehensiveness and efficiency.

First, the pictures at the top of Fig. 5 show the vulnerability discovery result of PhyFuzz and the sweep testing across four representative sensors involving sound, ultrasound, and EM signals. For sweep testing, we use lines to represent the normalized deviation across frequency spectra and highlight the distinct vulnerable frequency bands identified by sweep testing with color blocks. As for our fuzzing approach, we provided no prior knowledge to qualify the signal parameters and demonstrated the bug records with scatters. Notably, conventional sweep testing uses sine signals with variable frequency and fixes other parameters. In the experiment of INMP441, extended AM sweep testing is conducted due to the well-known nonlinear effects of microphones [11] (split by the grey dashed line). These figures reveal that PhyFuzz is capable of covering all the vulnerabilities identified by sweep testing under all settings. Moreover, in some cases, e.g., L3G4200D and BMP280, PhyFuzz can discover other vulnerable frequency ranges induced by modulation signals. These results validate PhyFuzz's dual capability in both reproducing known vulnerabilities and uncovering novel attack surfaces through its intelligent fuzzing strategy.

Second, we thoroughly compare the efficiency of PhyFuzz and the sweep test and represent it at the bottom of Fig. 5 and Tab. VI. Due to the exhaustive frequency-space exploration of the sweep testing, its time cost is inevitably high. As mentioned earlier, we equalize the comparison by reducing the search precision to match PhyFuzz's time budget. To fairly quantify performance, we propose three metrics: 1) **Efficiency**, the number of bugs found per unit test time; 2) **TFB**, the time to discover the first bug; and 3) **TDMVT**, the time to discover the most vulnerability types. Specifically, we introduce **TFB** because early feedback on discovered bugs can improve the user experience and help address vulnerabilities without stagnation periods. Besides, **TDMVT** meets the requirement of finding diverse vulnerability types, whereas **Efficiency** cannot represent it.



(a) INMP441 (20 kHz-48 kHz) (b) ADXL345 (20 Hz-20 kHz) (c) L3G4200D (20 Hz-48 kHz) (d) BMP180 (70 Mhz-5 GHz)

(e) INMP441 (20 kHz-48 kHz) (f) ADXL345 (20 Hz-20 kHz) (g) L3G4200D (20 Hz-48 kHz) (h) BMP180 (70 Mhz-5 GHz)

Fig. 5: Comparison between PhyFuzz and sweep testing. (a)-(d) represent the frequency distribution and the deviation degree of bug cases. (e)-(f) represent the efficiency of finding bugs and vulnerability types.

TABLE VI: Detailed metric comparison between PhyFuzz and sweep testing.

Sensor	Precision	Algorithm	Test Time	Bug Number	Efficiency	TFB	TDMVT
INMP441	10 Hz	Sweep Test	5600	935	0.1670	1	2802
(20 kHz-48 kHz)	0.1 Hz	PhyFuzz	5784	1249	**0.2159**	1	**37**
ADXL345	3 Hz	Sweep Test	6660	447	0.0671	388	2098
(20 Hz-20 kHz)	0.1 Hz	PhyFuzz	6156	452	**0.0734**	**63**	**100**
L3G4200D	10 Hz	Sweep Test	4798	75	0.0156	400	800
(20 Hz-48 kHz)	0.1 Hz	PhyFuzz	5747	144	**0.0251**	**121**	**300**
BMP180	1 MHz	Sweep Test	4930	85	0.0172	1625	1700
(70 MHz-5 GHz)	10 kHz	PhyFuzz	5999	645	**0.1075**	**45**	**172**

Our experiment results illustrate that PhyFuzz consistently outperforms sweep testing across all evaluation metrics. We can conclude that PhyFuzz not only detects a higher quantity of bug cases but also identifies the most bug types faster.

Specifically, in terms of **Efficiency**, PhyFuzz presents comparable performance over sweep testing. Notably, while sweep testing occasionally benefits from starting frequencies within vulnerable ranges (e.g., INMP441), PhyFuzz still shows a superior long-term efficiency. When dealing with large search spaces (e.g., electromagnetic experiment on BMP180), PhyFuzz achieves a 6.25× speedup in **Efficiency**. Furthermore, PhyFuzz excels in TFB, delivering valid responses up to 12× faster than sweep testing. The TDMVT metric also confirms the most striking advantage of PhyFuzz: it maintains an average 27× higher speed in discovering most vulnerability types, while sweep testing often stagnates by repeatedly identifying similar vulnerabilities.

Another distinguishing superiority of PhyFuzz lies in its ability to efficiently discover more vulnerability types. In contrast, traditional sweep testing often stagnates by repeatedly identifying similar vulnerabilities through minor parameter adjustments.

Our experiments yield two critical findings that demonstrate PhyFuzz's superiority:

1. Time Efficiency Advantage. In our comparison experiments, PhyFuzz operates with significantly finer step sizes (indicating a more comprehensive search space). Still, it outperforms sweep testing within equivalent time budgets in terms of the number of bug records and the variety of vulnerability types.

2. Expert Knowledge Independence. The fundamental limitation of sweep testing lies in its frequency exploration. Thus, it requires expert knowledge to manually preset other signal parameters (e.g., the experiment with INMP441). However, this may be manageable for an experienced researcher or a sensor developer, it presents challenges for the B2B consumers, who are also responsible for validating the security of third-party sensors and ensuring the overall security of the CPS product. In contrast, PhyFuzz can cover more signal parameters by automatically generating test cases without requiring prior knowledge while maintaining testing effectiveness.

3) Effectiveness of Bug Indicator: As shown in Alg. 3, PhyFuzz has the capability to match bug records with known measurement shaping vulnerabilities automatically. Our algo-

rithm will first filter partial cases based on the deviation and the pre-defined threshold. Then, we utilize feature templates to classify the remaining cases automatically.

To ensure the vulnerabilities are reproducible and exploitable, we additionally manually re-tested and verified that all vulnerabilities could be exploited to spoof the sensor's output, some of which are demonstrated in the Case Study and Appendix B. Moreover, we analyzed the measurements and quantified the attack impact on each sensor vulnerability to illustrate the susceptibility of SUTs.

Tab. VII presents the result of our bug indicator. During impact quantification, we identify the case with the most significant deviation for each vulnerability and define the ratio of its absolute offset and sensor range as the attack impact. Unlike the sensor-agnostic feature-level deviation used during fuzzing, the absolute offset of the time-domain signal straightforwardly represents the physical significance of vulnerabilities, especially for sensors of the same type.

From the results, we can observe that our template can achieve high matching rates across all test cases, effectively identifying well-studied vulnerabilities and filtering relevant records. Notably, the impact of mismatched cases is typically less prominent, with manual analysis showing that most of them can be categorized as other vulnerabilities. These mismatches primarily occur when the input signal strength is insufficient to trigger feature deviations above the threshold. In particular, some bugs of 1200TVL can not be identified by any template, which we will further discuss in the case study.

In addition, due to the power difference of the signal transmitter, the impact of acoustic signals is greater than that of EM signals. In subsequent case studies, we can verify the correctness of these bug records with an extra amplifier for EM signals. More importantly, it indicates that `PhyFuzz` can detect the subtle variations in sensor readings that are difficult for the human eye, thereby avoiding power consumption and *bodily harm* to testers from high-power attacks.

In summary, our bug indicator can effectively classify existing vulnerabilities and filter out most cases. It allows us to provide comprehensive qualitative vulnerability reports for consumers and alleviate manual efforts for experts through the functionality of bug Indicators.

C. Case Study

The above experiments demonstrate that `PhyFuzz` specializes in identifying the malfunctions of SUTs. However, when we find some new bugs, `PhyFuzz` cannot reveal their incentive and principle. In this section, we manually analyze some typical and crucial cases to verify whether the bugs detected by `PhyFuzz` are correct and valuable.

1) Inertial Sensor: Inertial sensors are used to measure the orientation, position, and motion of an object in space. In our evaluation, we select four types of inertial sensors: two accelerometers (ADXL345, LIS2DW12) and two gyroscopes (MPU6050, L3G4200D). In particular, MPU6050 can also act as an accelerometer, providing output with six-axis data.

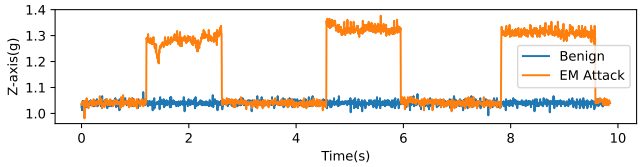


Fig. 6: The measurement under 1510 MHz electromagnetic attack from the Z-axis of ADXL345.

These sensors typically share similar vulnerabilities due to their common principle of inertia.

Sound and Ultrasonic. Acoustic signals pose a significant threat to inertial sensors as they can induce resonance and vulnerabilities in the sensing unit. In our evaluation, we subjected the inertial sensor to experiencing 0 g along the X and Y axes and 1 g along the Z axis. Then, `PhyFuzz` successfully identified several vulnerabilities. Taking the Z axis as an example, we demonstrate some details in Tab. VIII.

ADXL345 and MPU6050 have been extensively studied in the cyber-physical system community [5]. Although `PhyFuzz` finds the same vulnerabilities as previous research, we observed that their vulnerable frequency ranges are not entirely identical, suggesting that the manufacturing and product batch can influence the sensor vulnerabilities. We also notice that LIS2DW12 does not exhibit asymmetric saturation vulnerabilities. The bug record indicates that the measurement deviation of LIS2DW12 is not significant enough to exceed its amplifier's dynamic range. Notably, L3G4200D is the only sensor with an analog filter before the AD converter, which helps mitigate resonance and thus avoids signal aliasing. However, `PhyFuzz` still finds that L3G4200D has a very narrow vulnerable frequency range, which can induce aliasing and asymmetric saturation.

Electromagnetic. For inertial sensors, `PhyFuzz` successfully identified electromagnetic vulnerabilities in three sensors, as shown in Tab. IX. The result indicates that `PhyFuzz` discovered a similar vulnerable frequency range of ADXL345 and LIS2DW12, even without an amplifier. To demonstrate the attack performance clearly, we reproduce the EM attack with a square wave on ADXL345 with a 20 W power amplifier. We show the result in Fig. 6, demonstrating the Z-axis output of ADXL345 with a square electromagnetic wave. It is evident that the measurement exhibits discernible changes in response to the electromagnetic signals.

In comparison, the output of MPU6050 and L3G42000D demonstrates greater resistance to electromagnetic interference. `PhyFuzz` does not find significant measurement deviations under our evaluation settings, suggesting it is relatively resistant to electromagnetic interference.

2) Camera: Camera sensors are more intricate than traditional sensors due to their multi-dimensional output. To seamlessly adapt the camera sensor, we extract statistical features through edge computing of data collectors, similar to the driven programs of other sensors. In our evaluation, we selected two sensors, IMX219 and 1200TVL, representing

TABLE VII: Template matching and impact identification result for each sensor and sensor vulnerability.

Sensor	Range	Signal	Bug Number	Record Num / Impact (%)				
				A.	A.S.	N.E.	F.L.	Miss
ADXL345	±2g	🔊	452	452/27.2	93/27.2	–	–	0
		📶	150	4/3.2	150/4.8	–	–	0
LIS2DW12	±2g	🔊	152	60/4.9	86/4.9	–	–	6/0.2
		📶	150	4/0.5	150/0.7	–	–	0
MPU6050	±2g	🔊	832	557/19.7	802/19.7	–	–	3/0.1
L3G4200D	±250	🔊	60	54/2.7	1/1.8	–	–	6/1.0
SPH0645	±1	🔊	1345	440/19.9	–	463/46.7	1182/29.0	5/0.0
INMP441	±1	🔊	1079	931/30.2	–	893/33.9	296/41.1	11/0.1
		📶	518	493/42.2	62/16.5	–	–	5/0.0
TCS34725	0∼255	📶	7	7/0.7	–	–	–	0
BH1750	0∼65536	📶	125	117/0.4	125/0.5	–	–	0
1200TVL	0∼255	📶	6	–	–	–	–	6/0.3
BMP180	-40∼85	📶	295	4/0.1	295/1.3	–	–	0
BMP280	300∼1100	📶	645	7/0.2	644/0.4	–	–	0
SHT30	0∼100	📶	151	150/1.2	151/1.2	–	–	0

TABLE VIII: Details fuzzing result of Inertial Sensors under the sound and ultrasonic test

Sensor	Vulnerability	Signal Charateristic	
		Freq. (kHz)	Amp.* (gain)
ADXL345	R. + A.	0.65-0.9, 2.5-3.5, 5.4-6.8	0.3
	R. + A.S.	2.74-2.77	0.9
LIS2DW12	R. + A.	0.5-1.1, 1.9-2.1, 3.7-3.8	0.6
MPU6050+	R. + A.	5.0-5.2	0.5
	R. + A.S.	5.15-5.18	1
L3G42000D	R. + A.	8.00-8.06	0.8
	R. + A.S.	8.04	1

* Amplitude indicates the lowest amplitude to induce the vulnerability.
+ Presenting the vulnerability of X axis due to no bug record at Z axis.

TABLE IX: Details fuzzing result of Inertial Sensors under the electromagnetic test

Sensor	Vulnerability	Signal Charateristic	
		Freq. (MHz)	Amp.* (gain)
ADXL345	E.C.+A.S.	1470-1610	0.4
LIS2DW12	E.C.+A.S.	1480-1600	0.45
MPU6050	N.S.D.+	–	–
L3G42000D	N.S.D.+	–	–

* Amplitude indicates the lowest amplitude to induce the vulnerability.
+ N.S.D.: No Significant Deviation.

CMOS and CCD cameras, respectively. As they are not equipped with an anti-shake module, no significant deviation was observed during sound and ultrasonic tests. However, as Tab. X shows, `PhyFuzz` still successfully identifies the electromagnetic vulnerabilities in 1200TVL.

TABLE X: Details fuzzing result of Camera Sensors under the electromagnetic test

Sensor	Vulnerability	Signal Charateristic	
		Freq. (MHz)	Amp.* (gain)
IMX219	N.S.D.+	–	–
1200TVL	E.C.+A.S.	1452-1600, 1700, 2800,...	0.8
	E.C.	1058, 1224	1

* Amplitude indicates the lowest amplitude to induce the vulnerability.
+ N.S.D.: No Significant Deviation.

Due to the limited amplitude of our electromagnetic transmitter, the deviations in camera output are only discernible at the feature level. To better illustrate the effects of the attack at the pixel level, we employ a 20 W amplifier to enhance the electromagnetic interference. The image results are presented in Fig. 7 in Appendix B, revealing two notable phenomena. First, the image output of the 1200TVL gets bright (overexposed) under several frequencies of EM signals, which we assume is due to sensor saturation. Second, we also observe that several black-and-white strips appear in the image under some frequency points.

We further conducted a complementary case study on microphone interference. Due to space constraints, this part of the analysis will be detailed in Appendix C.

VI. Related Work

Hardware and CPS Fuzzing. The exponential growth of critical infrastructure and IoT deployments has driven the extension of software fuzzing to hardware and CPS, as illustrated in Table I. DeFUZZ adapts conventional fuzzing algorithms to sensor firmware [21]. Trippel et al. demonstrate a hardware fuzzing pipeline by converting RTL hardware designs into equivalent software models [26]. PGfuzz [23] constructs a policy-guided fuzzing framework for RV systems, uncovering 156 unknown bugs in multiple autopilot frameworks. CPS-Fuzz [27] deploys fuzzing over a real-world water treatment system and discovers 15 potential unsafe system states. Yang et al. develop a sensor-input fuzzer to expose spoofing attacks on RV systems [25]. While these approaches advanced the field, they primarily focus on the non-physical layers of devices or systems, including the firmware [21], hardware designs [26], and control algorithms of CPS [23], [25], [27].

Black-Box Fuzzing. Tab. XI demonstrates that existing black-box fuzzing algorithms primarily follow two paradigms. The input analysis paradigm, which is particularly effective for applications with strict grammar and semantics, such as JavaScript engines, utilizes structured input models for test case generation. Montage [18] designs a language model-

TABLE XI: The strategy of existing blackbox fuzzing algorithm

Method	Target Application	Para.	Strategy
Favocado [29]	JavaScript Engine	**I.**	Dependency Analysis
Codealchemist [30]	JavaScript Engine	**I.**	Code Chunk Assembly
Montage [18]	JavaScript Engine	**I.**	Language Model
Joeri et al. [31]	Protocol	**O.**	State Machine
zhang et al. [23]	CPS	**O.**	Maximize Deviation
DriveFuzz [24]	ADS	**O.**	Minimize Quality
Schiller et al. [33]	Drone Firmware	**O.**	Split Error Commands
DifFuzz [42]	Program Side-channel	**O.**	Differential Analysis

I.: Input Analysis. **O.**: Ouput Analysis.

guided fuzzer to uncover vulnerabilities in the JavaScript engine. Favocado [29] generates test cases based on a rule set of grammar and semantic information. CodeAlchemist [30] produces effective test cases with code fragment assembly. The alternative output analysis paradigm analyzes output information to guide fuzzing. Protocol fuzzing work infers a state machine for the TLS protocol or web applications [31], [32] based on the system output. Practical CPS or RV fuzzing frameworks adopt the drone's crash signal [33] or the deviation [23], [27] as a feedback mechanism. Despite their success in respective domains, these approaches either exhibit narrow applicability or rely on oversimplified feedback mechanisms. For sensor vulnerability detection, such approaches prove inadequate due to the diversity of the transduction attack.

VII. Limitation and Future Work

In this paper, we propose PhyFuzz to automatically identify vulnerabilities on diverse sensors. We improved several fuzzing techniques and developed an approach compatible with different types of sensors and signal modalities. However, certain types of sensors and signal modalities are still not covered by PhyFuzz. For example, active sensors are excluded since their main vulnerability is the lack of echo authentication [46], [47], which can be easily detected by replaying the active signal. Nonetheless, our fuzzing framework could apply to active sensors like LiDAR in detecting other common vulnerabilities, such as EM coupling. Additionally, our prototype is not compatible with large-scale integrated sensor systems [40], [48]. Their large size makes them susceptible to transduction attacks from specific directions. A practical solution is to test their individual submodules.

Regarding the signal modalities, we adopt acoustic, laser, and electromagnetic signals as attack signals in our evaluations. However, other modalities, such as voltage [15] and magnetic [49], [50], can also induce transduction attacks. As the first work on physical signal fuzzing, this paper focused on proposing a scalable framework and validating its effectiveness with the most common signal modalities in prior studies. We believe extension to other modalities is viable and would be a promising future direction.

VIII. Conclusion

In this paper, we present PhyFuzz, a novel physical fuzzing framework for automatically uncovering sensor vulnerabilities in an expert-knowledge-independent manner. Our approach bridges the gap between traditional software fuzzing and transduction attacks, providing a practical tool for sensor developers and users to identify potential risks. PhyFuzz addresses key challenges in the physical domain—such as the large input space and the black-box nature of sensors—through a signal construction set and a feature discretization algorithm. To validate the efficiency of PhyFuzz, we implemented our approach by building an integrated toolkit in the physical scenario. Extensive experiments are conducted with 13 sensors across 9 types, revealing 46 vulnerabilities. All discovered vulnerabilities were responsibly disclosed to the respective manufacturers. The complete results and PhyFuzz source code will be released to support future research.

Acknowledgement

We sincerely appreciate our anonymous reviewers and shepherd for their valuable comments and suggestions. This work was supported by China NSFC Grant 62201503 and 62222114.

References

[1] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: The next computing revolution," in *Proceedings of Design Automation Conference*, 2010, pp. 731–736.

[2] 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 2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 233–248.

[3] D. Davidson, H. Wu, R. Jellinek, T. Ristenpart, and V. Singh, "Controlling uavs with sensor input spoofing attacks," in *Proceedings of the 10th USENIX Conference on Offensive Technologies*, ser. WOOT'16. USA: USENIX Association, 2016, p. 221–231.

[4] Y. Son, H. Shin, D. Kim, Y. Park, J. Noh, K. Choi, J. Choi, and Y. Kim, "Rocking drones with intentional sound noise on gyroscopic sensors," in *Proceedings of 24th USENIX security symposium (USENIX Security 15)*, 2015, pp. 881–896.

[5] 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 2017 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 2017, pp. 3–18.

[6] X. Ji, Y. Cheng, Y. Zhang, K. Wang, C. Yan, W. Xu, and K. Fu, "Poltergeist: Acoustic adversarial machine learning against cameras and computer vision," in *Proceedings of 2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 160–175.

[7] H. Kim, R. Bandyopadhyay, M. O. Ozmen, Z. B. Celik, A. Bianchi, Y. Kim, and D. Xu, "A systematic study of physical sensor attack hardness," in *Proceedings of 2024 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2024, pp. 143–143.

[8] D. Hage, "The trillion sensor economy is coming," https://rfidgs.com/the-trillion-sensor-economy-is-coming/, 2021.

[9] T. Sugawara, B. Cyr, S. Rampazzi, D. Genkin, and K. Fu, "Light commands: Laser-based audio injection attacks on voice-controllable systems," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 2631–2648.

[10] D. F. Kune, J. Backes, S. S. Clark, D. Kramer, M. Reynolds, K. Fu, Y. Kim, and W. Xu, "Ghost talk: Mitigating emi signal injection attacks against analog sensors," in *Proceedings of 2013 IEEE Symposium on Security and Privacy*. IEEE, 2013, pp. 145–159.

[11] G. Zhang, C. Yan, X. Ji, T. Zhang, T. Zhang, and W. Xu, "Dolphinattack: Inaudible voice commands," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 103–117.

[12] X. Zhu, S. Wen, S. Camtepe, and Y. Xiang, "Fuzzing: a survey for roadmap," *ACM Computing Surveys (CSUR)*, vol. 54, no. 11s, pp. 1–36, 2022.

[13] J. Fraden and J. King, *Handbook of modern sensors: physics, designs, and applications*. Springer, 2004, vol. 3.

[14] "Ieee standard for sensor performance parameter definitions," *IEEE Std 2700-2017 (Revision of IEEE Std 2700-2014)*, pp. 1–64, 2018.

[15] K. Wang, S. Xiao, X. Ji, C. Yan, C. Li, and W. Xu, "Volttack: Control iot devices by manipulating power supply voltage," in *Proceedings of 2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 1771–1788.

[16] K. Dewey, J. Roesch, and B. Hardekopf, "Language fuzzing using constraint logic programming," in *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*, ser. ASE '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 725–730. [Online]. Available: https://doi.org/10.1145/2642937.2642963

[17] K. Dewey, J. Roesch, and B. Hardekopf, "Fuzzing the rust typechecker using clp (t)," in *Proceedings of 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2015, pp. 482–493.

[18] S. Lee, H. Han, S. K. Cha, and S. Son, "Montage: A neural network language model-guided javascript engine fuzzer," in *Proceedings of 29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 2613–2630.

[19] M. Böhme, V.-T. Pham, M.-D. Nguyen, and A. Roychoudhury, "Directed greybox fuzzing," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 2329–2344. [Online]. Available: https://doi.org/10.1145/3133956.3134020

[20] M. Zalewski, "Afl (american fuzzy lop)," https://github.com/google/AFL, 2021.

[21] X. Zhu, S. Liu, and A. Jolfaei, "A fuzzing method for security testing of sensors," *IEEE Sensors Journal*, 2023.

[22] T. Kim, C. H. Kim, J. Rhee, F. Fei, Z. Tu, G. Walkup, X. Zhang, X. Deng, and D. Xu, "RVFuzzer: Finding input validation bugs in robotic vehicles through Control-Guided testing," in *Proceedings of 28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 425–442. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/kim

[23] H. Kim, M. O. Ozmen, A. Bianchi, Z. B. Celik, and D. Xu, "Pgfuzz: Policy-guided fuzzing for robotic vehicles." in *Proceedings of NDSS*, 2021.

[24] S. Kim, M. Liu, J. J. Rhee, Y. Jeon, Y. Kwon, and C. H. Kim, "Drivefuzz: Discovering autonomous driving bugs through driving quality-guided fuzzing," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1753–1767. [Online]. Available: https://doi.org/10.1145/3548606.3560558

[25] 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 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 3503–3505.

[26] T. Trippel, K. G. Shin, A. Chernyakhovsky, G. Kelly, D. Rizzo, and M. Hicks, "Fuzzing hardware like software," in *Proceedings of 31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 3237–3254.

[27] F. Zhang, Q. Wu, B. Xuan, Y. Chen, W. Lin, C. M. Poskitt, J. Sun, and B. Chen, "Constructing cyber-physical system testing suites using active sensor fuzzing," *IEEE Transactions on Software Engineering*, 2023.

[28] "Gazobe," https://gazebosim.org/, 2020.

[29] S. T. Dinh, H. Cho, K. Martin, A. Oest, K. Zeng, A. Kapravelos, G.-J. Ahn, T. Bao, R. Wang, A. Doupé *et al.*, "Favocado: Fuzzing the binding code of javascript engines using semantically correct test cases." in *Proceedings of NDSS*, 2021.

[30] H. Han, D. Oh, and S. K. Cha, "Codealchemist: Semantics-aware code generation to find vulnerabilities in javascript engines." in *Proceedings of NDSS*, 2019.

[31] J. De Ruiter and E. Poll, "Protocol state fuzzing of tls implementations," in *Proceedings of 24th USENIX Security Symposium (USENIX Security 15)*, 2015, pp. 193–206.

[32] A. Doupé, L. Cavedon, C. Kruegel, and G. Vigna, "Enemy of the state: A State-Aware Black-Box web vulnerability scanner," in *Proceedings of 21st USENIX Security Symposium (USENIX Security 12)*. Bellevue, WA: USENIX Association, Aug. 2012, pp. 523–538. [Online]. Available: https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/doupe

[33] N. Schiller, M. Chlosta, M. Schloegel, N. Bars, T. Eisenhofer, T. Scharnowski, F. Domke, L. Schönherr, and T. Holz, "Drone security and the mysterious case of dji's droneid," in *Proceedings of 30th Annual Network and Distributed System Security Symposium, NDSS 2023, San Diego, California, USA, February 27 - March 3, 2023*. The Internet Society, 2023. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/drone-security-and-the-mysterious-case-of-djis-droneid/

[34] N. Roy, H. Hassanieh, and R. Roy Choudhury, "Backdoor: Making microphones hear inaudible sounds," in *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, 2017, pp. 2–14.

[35] B. Cyr, T. Sugawara, and K. Fu, "Why lasers inject perceived sound into mems microphones: Indications and contraindications of photoacoustic and photoelectric effects," in *Proceedings of 2021 IEEE Sensors*. IEEE, 2021, pp. 1–4.

[36] K. Wang, R. Mitev, C. Yan, X. Ji, A.-R. Sadeghi, and W. Xu, "Analyzing and defending ghosttouch attack against capacitive touchscreens," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–16, 2024.

[37] S. Köhler, R. Baker, and I. Martinovic, "Signal injection attacks against ccd image sensors," in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, 2022, pp. 294–308.

[38] D. Su, S. Xie, A. Chen, X. Shang, K. Zhu, and H. Xu, "Basic emission waveform theory: A novel interpretation and source identification method for electromagnetic emission of complex systems," *IEEE Transactions on Electromagnetic Compatibility*, vol. 60, no. 5, pp. 1330–1339, 2018.

[39] A. Novak and P. Honzík, "Measurement of nonlinear distortion of mems microphones," *Applied Acoustics*, vol. 175, p. 107802, 2021.

[40] H. Shin, D. Kim, Y. Kwon, and Y. Kim, "Illusion and dazzle: Adversarial optical channel exploits against lidars for automotive applications," in *Proceedings of Cryptographic Hardware and Embedded Systems – CHES 2017*, W. Fischer and N. Homma, Eds. Cham: Springer International Publishing, 2017, pp. 445–467.

[41] Z. Zheng, X. Li, C. Yan, X. Ji, and W. Xu, "The silent manipulator: A practical and inaudible backdoor attack against speech recognition systems," in *Proceedings of the 31st ACM International Conference on Multimedia*, ser. MM '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 7849–7858. [Online]. Available: https://doi.org/10.1145/3581783.3613843

[42] S. Nilizadeh, Y. Noller, and C. S. Pasareanu, "Diffuzz: differential fuzzing for side-channel analysis," in *Proceedings of 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 176–187.

[43] X. Zhu and M. Böhme, "Regression greybox fuzzing," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 2169–2182. [Online]. Available: https://doi.org/10.1145/3460120.3484596

[44] A. Pahl, K.-U. Rathjen, and S. Dickmann, "Intended electromagnetic interference with motion detectors," in *Proceedings of 2021 IEEE International Joint EMC/SI/PI and EMC Europe Symposium*, 2021, pp. 324–328.

[45] S. Köhler, R. Baker, and I. Martinovic, "Signal injection attacks against ccd image sensors," in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 294–308. [Online]. Available: https://doi.org/10.1145/3488932.3497771

[46] W. Xu, C. Yan, W. Jia, X. Ji, and J. Liu, "Analyzing and enhancing the security of ultrasonic sensors for autonomous vehicles," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 5015–5029, 2018.

[47] C. Yan, W. Xu, and J. Liu, "Can you trust autonomous vehicles: Contactless attacks against sensors of self-driving vehicle," *Def Con*, vol. 24, no. 8, p. 109, 2016.

[48] Z. Jin, X. Ji, Y. Cheng, B. Yang, C. Yan, and W. Xu, "Pla-lidar: Physical laser attacks against lidar-based 3d object detection in autonomous vehicle," in *2023 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: Proceedings of IEEE Computer Society, may 2023, pp. 1822–1839. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP46215.2023.10179458

[49] D. Dai, Z. An, and L. Yang, "Inducing wireless chargers to voice out for inaudible command attacks," in *Proceedings of 2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 1789–1806.

[50] A. Barua and M. A. Al Faruque, "Hall spoofing: A non-invasive dos

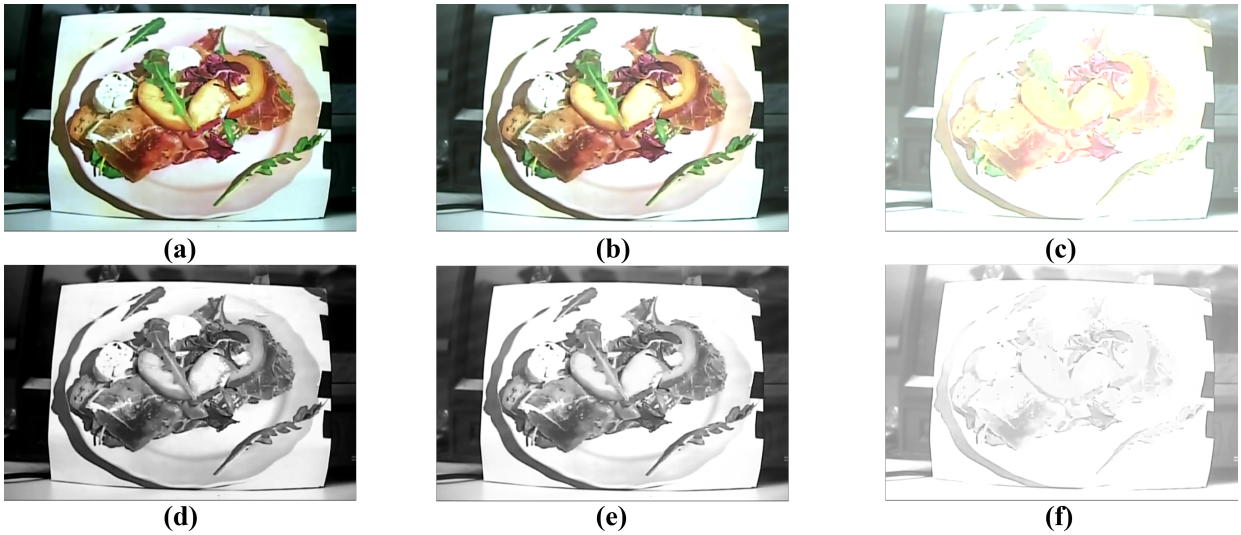
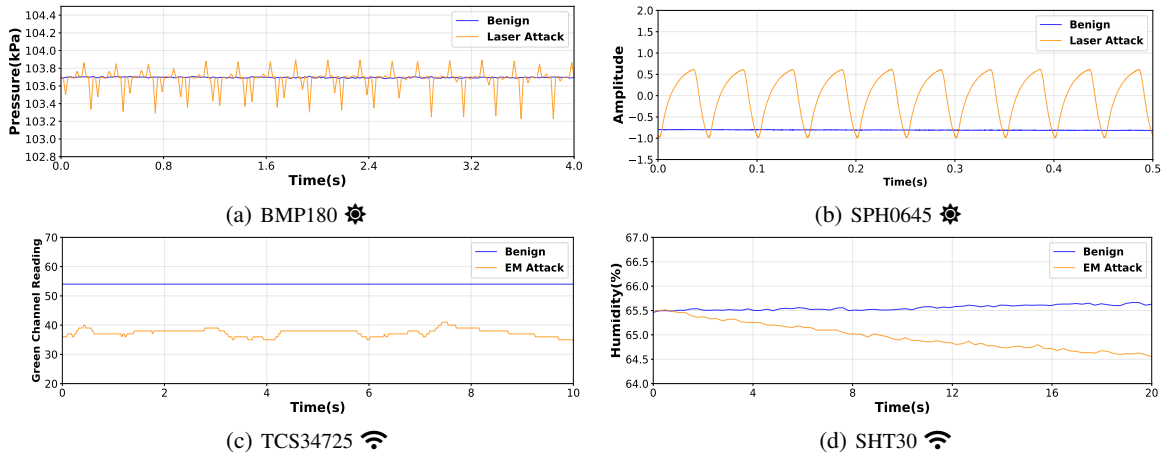


Fig. 7: The image output of 1200TVL under electromagnetic tests. (a): original image; (b): interfered image with black and white strips; (c): interfered image with overexpose. (d)(e)(f): the corresponding grayscale images.



(a) BMP180 (b) SPH0645

(c) TCS34725 (d) SHT30

Fig. 8: Laser and electromagnetic vulnerability validation.

attack on grid-tied solar inverter," in *Proceedings of 29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 1273–1290.

[51] S. Koffas, J. Xu, M. Conti, and S. Picek, "Can you hear it? backdoor attacks via ultrasonic triggers," in *Proceedings of the 2022 ACM Workshop on Wireless Security and Machine Learning*, ser. WiseML '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 57–62. [Online]. Available: https://doi.org/10.1145/3522783.3529523

Appendix A
Disclosure to Manufacturers

All six newly discovered vulnerabilities were reported to their respective manufacturers. We initially contacted the sensor vendors (all authorized distributors) via email, providing the affected sensor model information and a brief description of the vulnerability types. Most vendors acknowledged receipt of our reports and confirmed that they will notify the manufacturers, while two vendors recommended that we contact the upstream manufacturers directly. We subsequently contacted these two manufacturers to disclose the vulnerabilities (involving three vulnerabilities). They requested detailed information on the sensor's physical vulnerability report. Upon receiving our comprehensive vulnerability reports, the manufacturers confirmed that the information had been forwarded to their specialized security teams for evaluation.

Appendix B
Vulnerability Validation

Here, we provide some of the vulnerabilities explored by `PhyFuzz` in our evaluation. Specifically, we focus on sensor types that have received little attention in previous work, especially under laser and electromagnetic signals.

A. Laser Vulnerability Validation

Laser attack can be easily reproduced because it's not sensitive to a specific frequency range. Here, we present two cases of a pressure sensor, BMP180, and a microphone, SPH0645, in Fig. 8 (a-b), which demonstrate both photoacoustic and

photoelectric effects, respectively. The attack signal is a sine wave (amplitude varies sinusoidally). We can find that the outputs of both sensors fluctuate synchronously with the attack signal.

B. Electromagnetic Vulnerability Validation with Amplifier

In our evaluation, we don't equip an amplifier in the implementation of PhyFuzz. However, the malfunction of sensor measurements is sometimes discernible at the feature-level, but difficult to detect with the naked eye, especially in the electromagnetic tests. To better show the attack performance, we reproduce the EM attack on ADXL345, TCS34725, SHT30, BH1750, and 1200TVL with a 20 W power amplifier.

The attack signal is still a sine electromagnetic signal. From the results in Fig. 8 (c-d), we can find that the measurements of the color sensor TCS34725 (Green light output) and the H&T sensor SHT30 (humidity output) are affected under the electromagnetic attack. The sine EM wave can offset the readings of the TCS34725 by a fixed value immediately and gradually decrease those of the SHT30, suggesting a different vulnerability principle. The results of the ADXL345 are presented in the previous Case Study. Regarding the BH1750, we find that a high-power electromagnetic attack can directly induce an I/O error, disabling its operation.

In addition, Fig. 7 shows the image output of 1200TVL, a CCD camera, under the electromagnetic tests. In particular, as shown in Fig. 7 (b), we observe that several black and white strips appear in the darker areas of the image under some frequency points. Besides, we find the image output of the 1200TVL gets bright (overexposed) under several frequencies of EM signals, as illustrated in Fig. 7 (c), which we assume is due to saturation. We also draw the gray scale images of these pictures in Fig. 7 (d)-(f) to demonstrate their difference.

APPENDIX C
CASE STUDY FOR MICROPHONE

Microphone sensors can convert an acoustic signal into an electrical signal, widely used in various intelligent applications such as telecommunication and audio recognition systems. In our evaluation, we chose two MEMS microphone sensor prototypes: SPH0645 and INMP441. Despite the fact that current microphones can achieve sample rates exceeding 96 kHz, many still come equipped with a band-pass filter to eliminate high-frequency components. This is necessary because these components may lead to intermodulation distortion or introduce inaudible backdoor triggers [51]. Therefore, ultrasonic signals should not fall within the expected response range of the microphones.

Ultrasonic. The ultrasonic signal is proven to be a severe threat to microphone sensors, as it could cause the vibration of the microphone diaphragm and generate an electrical signal. While a low-pass filter can mitigate the interference of ultrasonic, previous research has reported several vulnerabilities induced by malicious ultrasonic signals [11], [34]. In our evaluation, PhyFuzz also finds such vulnerabilities under ultrasonic testing, as shown in Tab. XII.

TABLE XII: Details fuzzing result of Microphone Sensors under the ultrasonic test

Sensor	Vulnerability	Signal Charateristic		
		Freq. (kHz)	Amp.* (gain)	Signal Type
SPH0645	F. L.	20-22	0.3	–
	F.L. + A.	23-25	0.5	–
	N.E.	20-26	0.5	Modulated
INMP441	F. L.	20-22	0.3	–
	F.L. + A.	23-25	0.4	–
	N.E.	20-45	0.7	Modulated

*: Amplitude indicates the lowest amplitude to induce the vulnerability.

TABLE XIII: Details fuzzing result of Microphone Sensors under the electromagnetic test

Sensor	Vulnerability	Signal Charateristic		
		Freq. (MHz)	Amp.* (gain)	Signal Type
SPH0645	N.S.D.	–	–	–
INMP441	E.C.+N.E.	1570-1780	0.6	Modulated
	E.C..+N.E.	3000-3005	0.5	FM

*: Amplitude indicates the lowest amplitude to induce the vulnerability.
N.S.D.: No Significant Deviation.

From the result, it can be found that both SPH0645 and INMP441 exhibit similar vulnerabilities, as previously documented in various studies. The occurrence of frequency leakage suggests that their filters are incapable of screening out unexpected ultrasonic frequencies, thereby enabling the injection of ultrasonic signals. It is worth noting that both microphones employ digital filtering after the AD converter. According to the Nyquist-Shannon sampling theorem, the leaked ultrasonic signal could lead to acoustic aliasing, a conclusion supported by our evaluation.

In addition, when the test case involves a modulated signal, particularly the amplitude-modulated signal, multiple high-frequency components will produce unexpected low-frequency components due to the nonlinear effects of the electronic components in the microphone, such as the amplifier. The result also indicates that SPH0645 and INMP441 can not avoid this vulnerability.

Electromagnetic. Previous research has revealed that consumer electronic devices containing microphones are vulnerable to the injection of false audio signals [10]. Tab. XIII shows the result of SPH0645 and INMP441 under the electromagnetic test.

From the result, it can be inferred that SPH0645 exhibits better resilience against electromagnetic interference than INMP441. It's vital to note that the strength of the attack signals directly influences the success of the electromagnetic attack, and our evaluation still follows the implementations in Sec. V-A. Conversely, PhyFuzz finds two vulnerable frequency ranges of INMP441, indicating two potential entry points for the electromagnetic signals. Moreover, the result also demonstrates that normal sine electromagnetic signals can not effectively inject malicious signals into the microphone sensors. Even with a vulnerable frequency, modulated signals can deviate from the measurement of the microphone sensor.