Investigation on New Fuzzing Techniques to Address Navigation System Testing

Nina Haag, *Collins Aerospace & École nationale de l'aviation civile*
Christophe Ouzeau, Lotfi Fejri, Patrick Bartolone, *Collins Aerospace*
Antoine Blais, Daniel Prun, *École Nationale de l'Aviation Civile*

BIOGRAPHY

Nina Haag Nina Haag holds a Master's degree in Electrical Engineering with a specialization in Space Technology from the Technical University of Denmark. In January 2024, she began her Ph.D. at Collins Aerospace and the National School of Civil Aviation (ENAC). Her research focuses on developing and applying model-based fuzz testing techniques to GNSS receivers, aiming to address challenges in navigation system robustness, particularly in the context of aviation safety. By leveraging state machine models and innovative fuzz testing strategies, her work seeks to uncover vulnerabilities, improve system resilience, and contribute to the advancement of GNSS testing standards.

Dr. Christophe Ouzeau graduated in 2005 with a master in astronomy at the Observatory of Paris. He holds a Ph.D. thesis on degraded modes resulting from the multi-constellation use of GNSS, supported by DGAC DTI and supervised by ENAC. His Ph.D. was funded to support the EUROCAE WG-62 activities and successfully published in 2010. He created and managed internal research activities for several companies in the field of GNSS for various users' applications and joined Collins Aerospace in 2022. He is now Principal System Engineer and Author/Editor of ED-259/DO-401 standards for the EUROCAE WG-62 and RTCA WG-2 respectively.

Lotfi Fejri holds a Master's degree in Embedded Systems Electronics and Telecommunications, with a background in Aeronautical and Space Telecommunications from ENAC. He began his career at ESSP as a GNSS Engineer, contributing to the European Geostationary Navigation Overlay Service (EGNOS). He then joined Collins Aerospace as a Navigation Systems Engineer, where he worked on the MUGG (Multi-Mode GPS Galileo) project—a DFMC-SBAS prototype developed under ED-259A and co-funded by EUSPA. Currently, Lotfi is part of the Melbourne product line team, focusing on the development of an advanced GLU-2100 "GNSS denied" prototypes, intended for the Boeing 787.

Antoine Blais Dr. Antoine Blais has earned an engineering degree from ENAC (École Nationale de l'Aviation Civile, the French Civil Aviation University), Toulouse, France, in electrical engineering in 1993, a Master of Science in Signal, Image and Acoustics from the Institut National Polytechnique de Toulouse (INPT), France, in 2003 and a Ph.D. degree from INPT in 2014. Antoine Blais is currently an assistant professor in the Telecommunication Research Team of ENAC. His research activities relate to Global Navigation Satellite System (GNSS) receiver signal processing and Software Defined Radio (SDR) in particular.

Daniel Prun is associate professor in System Engineering at ENAC. After obtaining a PhD in computer science at French University "Pierre et Marie Curie" (Paris 6), he became consultant specialized in technical processes (specification, design, system integration and validation) in various sectors (defense, air traffic control, aeronautic, railway, medical. . .). In 2009, Daniel joined ENAC and the associated Interactive Informatics. Its research interests focus on modeling interaction systems, with the aim of improving their specification and verification processes.

Patrick Bartolone is a Senior GNSS System Engineer and Navigation Expert at Collins Aerospace. He has over ten years of experience in developing and implementing innovative solutions for navigation with integrity and safety, combining GNSS with inertial systems and ensuring resilience against GNSS spoofing and radio frequency interference. Prior to joining Collins Aerospace, Patrick worked at THALES Avionics as a Product Design Authority & Technical Leader, he has led the development of a Galileo/GPS and EGNOS receiver that can operate on Dual Frequency L1/E1-L5/E5a Multi-Constellation, as well as a military Galileo/GPS multi-constellation prototype receiver with inertial hybridization & anti-jamming / anti-spoofing capabilities. He has also evaluated and tested the ARAIM algorithm for embedding in the DFMC receiver as part of European research projects. His work enables safe and efficient transportation, navigation, and communication across various industries and sectors. He is passionate about advancing the state-of-the-art in GNSS and navigation technology and contributing to the global GNSS community.

ABSTRACT

Fuzz testing is a method used in software testing that involves inputting random or unexpected data into a system to identify vulnerabilities. Unlike deterministic methods, which test performance under controlled and predictable conditions, fuzz testing introduces variability to uncover hidden issues. This variability simulates real-world scenarios, uncovering weaknesses that might otherwise remain unnoticed. For instance, fuzz testing can effectively reveal how GNSS receivers respond to rapid signal fluctuations and other anomalous behaviors, situations often overlooked by standard tests. Unlike traditional methods that rely on predefined inputs, Collins Aerospace works on a new fuzz testing framework for GNSS, which employs advanced techniques such as automated input generation and real-time response monitoring. This approach not only facilitates a comprehensive assessment of receiver resilience but also allows for the dynamic adaptation of test scenarios in real-time, ensuring that a wide range of operational conditions is explored. The navigation equipment minimum testing procedures must be defined and need scenarios definitions as well as test steps and pass/fail criteria to provide minimum guidance to manufacturers for future equipment certification. The limitations of current testing methods further highlight the necessity of adopting fuzz testing. These methods predominantly rely on deterministic approaches, which do not effectively simulate the unpredictable nature of real-world signal degradation or complex interference scenarios posed by advanced spoofing techniques. As technology advances, the techniques utilized by malevolent actors likewise evolve, emphasizing the necessity for adaptive testing methodologies capable of responding to these changes. By introducing randomness and variability, fuzz testing plays a critical role in bolstering the reliability and operational integrity of GNSS systems by rigorously assessing their ability to withstand both known and unknown threats. The anticipated results from this fuzz testing framework are expected to identify vulnerabilities and enhance the resilience of GNSS receivers, suggesting that fuzz testing can play a transformative role in GNSS validation.

I. INTRODUCTION

Global Navigation Satellite Systems (GNSS) have become indispensable for a wide range of applications, from navigation and timing to precision agriculture and autonomous vehicles. However, the increasing sophistication of GNSS receivers, coupled with the evolving threat landscape, necessitates robust testing methodologies to ensure their reliability and security. Traditional testing approaches, while valuable, often rely on predefined test cases, limiting their ability to uncover unexpected vulnerabilities.

Fuzz testing, a powerful technique for uncovering software bugs and security vulnerabilities, has emerged as a promising approach to enhance GNSS receiver testing. By systematically generating and injecting diverse, often malformed, inputs, fuzz testing can reveal weaknesses in the receiver's handling of unexpected conditions, including signal anomalies, interference, and spoofing attacks.

This paper presents a comprehensive framework for applying fuzz testing to GNSS receivers. The framework leverages both black-box and white-box techniques to systematically explore the receiver's behavior under various input conditions. By combining these approaches, we aim to identify vulnerabilities that may be missed by traditional testing methods and enhance the overall security and reliability of GNSS receivers.

The paper is structured as follows:

- **Fuzz Testing Techniques:** Discusses various fuzzing techniques, including grammar-based, mutation-based, and evolutionary fuzzing, and their applicability to GNSS receivers.
- **Proposed Framework:** Details the architecture and components of the proposed fuzz testing framework, including input generation, execution, and output analysis.
- **Case Study:** Presents a case study demonstrating the application of the framework to a real-world GNSS receiver.
- **Conclusion and Future Work:** Summarizes the findings of the research and outlines potential future directions for enhancing GNSS receiver security through fuzz testing.

By adopting this framework, researchers and industry practitioners can contribute to the development of more secure and resilient GNSS receivers, safeguarding critical infrastructure and applications that rely on accurate and reliable positioning, navigation, and timing information.

II. FUZZ TESTING OVERVIEW

Fuzz testing is a traditional software testing method that has been applied since the 80s. It involves that "random" generated inputs are sent to the target software, which is then monitored to find bugs or crashes. In software testing this is a very well-established method. In the context of this project this method is being used to perform performance tests (hardware and software combined) of a GNSS-receiver. There are several fuzz testing approaches, but not all of them are suitable for GNSS receiver testing. In the following the most suitable approaches will be introduced and discussed.

Core Categories of Fuzz Testing

Fuzz testing can be grouped into three main areas: black-box, white-box, and grey-box testing (Sutton et al. (2007)).

- **Black-Box Testing:** Operates without knowledge of the system's internal structure, focusing solely on input and output behavior.
- **White-Box Testing:** Utilizes system internals to guide test case generation, such as analyzing code paths.
- **Grey-Box Testing:** Combines aspects of both black- and white-box approaches, using limited internal knowledge to enhance test efficiency.

A common assumption is that white-box testing is inherently more efficient than black-box testing. While white-box testing is crucial for detecting bugs at the source code level, black-box testing remains essential for identifying issues introduced during compilation, such as assembly-level bugs. For this reason, black-box, white-box, and grey-box approaches should be viewed not as mutually exclusive but as complementary techniques thatM together enhance overall testing effectiveness (Sutton et al. (2007)).

Fuzzing Techniques for GNSS Receiver Testing

Over the past decades multiple fuzzing techniques have been developed. In the following the most relevant for this project are discussed:

- **Grammar-Based Fuzzing:** This technique generates test inputs based on a predefined grammar that defines the valid structure of input data. In the context of GNSS, grammar-based fuzzing can be applied to create both valid and malformed GNSS signal. For instance, by defining the syntax of NMEA sentences, researchers can produce signals that adhere to the expected format, as well as those that deviate in crucial ways (e.g., incorrect checksums, malformed fields). This approach is essential for testing the receiver's ability to detect and reject invalid data while correctly processing legitimate signals. Yang et al. (2012)
- **Probabilistic Fuzzing:** This method utilizes statistical models to produce random inputs that reflect real-world conditions. In GNSS applications, probabilistic fuzzing can simulate varying environmental factors such as atmospheric disturbances or multi-path effects. By introducing noise and random variations in signal parameters, this approach can help reveal vulnerabilities in the receiver's algorithms that handle signal integrity. For example, it can test how well a receiver maintains lock under fluctuating signal strength or unpredictable interference patterns (Sutton et al. (2007)).
- **Mutation-Based Fuzzing:** This technique involves modifying existing valid GNSS signal data by introducing random changes, such as bit flips or alterations to specific fields. Mutation-based fuzzing is particularly useful for assessing the receiver's tolerance to minor signal perturbations, which can occur due to various factors such as interference or hardware imperfections. By generating inputs that mimic real-world signal distortions, this method can uncover how small deviations in the signal impact the receiver's performance and decision-making processes (Qian et al. (2023)).
- **Evolutionary Fuzzing (Genetic Algorithms):** Utilizing genetic algorithms, this fuzzing technique iteratively evolves test inputs based on previous testing outcomes. In GNSS testing, evolutionary fuzzing can be particularly effective in generating sophisticated attack vectors that mimic advanced spoofing and jamming techniques. For instance, by analyzing the receiver's responses to initial inputs, the algorithm can adjust parameters to create more challenging scenarios, ultimately revealing weaknesses that might be exploited in real-world attacks (Rawat et al. (2017)).
- **Concolic (Concrete + Symbolic) Fuzzing:** By combining concrete execution with symbolic execution, concolic fuzzing generates test inputs that explore both real and hypothetical scenarios. This method is beneficial for identifying control flow vulnerabilities in GNSS receivers, ensuring comprehensive testing of how the system reacts to unexpected inputs. For example, concolic fuzzing can help uncover pathways in the code that lead to crashes or unintended behavior under specific conditions, such as receiving corrupted signal data (Sen et al. (2005)).

Table 1: Fuzzing Techniques for GNSS Receiver Testing

Technique	Key Features	Application in GNSS Testing	Strengths	Weaknesses
Grammar-Based Fuzzing	Generates inputs based on a predefined grammar for valid/invalid signal structures (e.g., NMEA).	Tests signal validation and error detection, e.g., malformed fields or incorrect checksums.	Highly systematic; ensures compliance with input structure.	Limited to the accuracy of the grammar; does not cover random or environmental variations.
Probabilistic Fuzzing	Uses statistical models to generate inputs mimicking real-world conditions (e.g., interference).	Simulates environmental challenges like atmospheric noise or multipath effects.	Realistic; reveals weaknesses in signal integrity handling.	Less effective for systematic state coverage.
Mutation-Based Fuzzing	Modifies valid inputs by introducing random changes.	Tests receiver tolerance to minor signal distortions, such as bit flips or altered fields.	Simple to implement; mimics real-world signal imperfections.	Random mutations may lack focus, leading to inefficient testing.
Evolutionary Fuzzing	Employs genetic algorithms to evolve inputs based on feedback.	Generates sophisticated attack vectors, e.g., advanced spoofing or jamming scenarios.	Adapts to exploit vulnerabilities; excellent for advanced threat simulation.	Computationally intensive; requires iterative feedback.
Concolic Fuzzing	Combines concrete and symbolic execution to explore control flow vulnerabilities.	Identifies issues like crashes or unintended behaviors under specific input conditions.	Uncovers hidden code paths and edge cases.	Resource-intensive; requires access to the receiver's internal execution pathways.

Goal-Oriented Fuzz Testing

These are just some general methods that can be used by fuzzers to generate input for the system under test. To even increase the prospect of the results of the fuzz test, especially for a complex system as an GNSS-receiver, the fuzzer can be even more guided to test specific goals like:

- **Boundary and Edge Case Analysis:** This test targets edge cases in the system under test such as boundary values or extreme input conditions. By ensuring that the system under test behaves correctly at the boundaries, we verify its stability under rare or critical scenarios that could otherwise lead to failures. This is especially important for ensuring the system's resilience in unpredictable real-world conditions.
- **Error Handling and Recovery:** This strategy introduces fault conditions, such as network failures or invalid inputs, to test how the system under test handles and recovers from errors. By simulating real-world failures, we ensure that the system under test can recover gracefully, maintaining its integrity and reliability even after encountering faults.
- **Fuzzing with Randomized Inputs:** Randomized input sequences stress the system under test with unpredictable events, aiming to discover unexpected behaviors or failures that may arise from non-deterministic inputs. This type of fuzz testing ensures that the system under test remains stable and consistent even when faced with random or irregular inputs, helping identify edge cases that might otherwise go unnoticed.
- **Concurrency Testing:** This test examines the system under test's behavior when multiple transitions occur simultaneously or in quick succession. By fuzzing for potential race conditions, deadlocks, or other concurrency-related issues, this strategy ensures that the system under test can handle parallel events without errors, providing a deeper understanding of the system's behavior under multi-threaded or multi-process conditions.
- **Performance Testing:** Performance testing involves stress testing the system under test with large input sequences to identify performance bottlenecks, memory leaks, or degradation in performance under heavy load. This fuzz test ensures that the system under test remains efficient, even when handling a high volume of transitions, helping to pinpoint areas where performance might be optimized.
- **Resource Usage Monitoring:** This fuzz test focuses on monitoring memory and resource consumption during execution. It helps identify issues like excessive resource usage, memory leaks, or inefficient resource handling. By ensuring that the system under test maintains optimal resource utilization, this test ensures that the system operates efficiently even during resource-intensive operations.

Performance Metrics

After choosing the testing approach, target and fuzzer strategy it is also vital to identify which parameters can be collected during the testing from the system, apart the initial testing results.(Pargaonkar (2023)) Indeed, these parameters are from high interest because they are used to guide the fuzzing test process:

- **Error Rate:** The error rate tracks the percentage of actions or events resulting in errors compared to the total number of actions performed. This metric provides insights into the robustness of the system. A high error rate could indicate problems with the system's handling of certain inputs or events, or with its error recovery mechanisms. By identifying areas with high error rates, testers can focus on improving fault tolerance and resilience.
- **CPU Usage:** CPU usage measures the percentage of CPU resources consumed during the test. Monitoring CPU usage helps assess the computational demands of the system under test. High CPU usage could indicate inefficient processing or resource-heavy operations, which may need optimization to avoid performance degradation during runtime. This metric is critical for ensuring the system operates within acceptable computational limits.
- **Memory Usage:** Memory usage tracks the percentage of memory resources utilized during testing. This metric is important for detecting potential memory inefficiencies, such as memory leaks, that could negatively affect system performance. By monitoring memory usage, testers can ensure the system efficiently manages memory, particularly when handling complex operations or processing large amounts of data.
- **Disk Usage:** Disk usage measures the percentage of disk space consumed during testing, particularly relevant for systems that involve disk I/O operations. This metric helps identify whether the system is excessively using disk resources, which could lead to storage limitations or performance slowdowns. Ensuring that disk usage remains within acceptable levels is crucial for the stability of the system, especially in environments with limited storage capacity.

To conclude, we observe that a lot of fuzzing techniques can be relevant for GNSS testing. One approach is not better then another, and all depends on the end wished end results.

Feasibility of Fuzz Testing in Enhancing GNSS Receiver Testing

Traditional deterministic testing methods, widely used in evaluating GNSS receivers, focus on predefined input sets designed to systematically explore expected operational scenarios. These methods often emphasize compliance with established standards and specific performance metrics, providing a controlled environment to validate a receiver's functionality under anticipated conditions. However, while deterministic testing is effective in confirming that systems meet predefined specifications, it inherently limits the exploration of unexpected or anomalous behaviors, leaving significant gaps in the assessment of a receiver's robustness against real-world threats.

In contrast, fuzz testing introduces a paradigm shift by generating a wide array of random, or unexpected inputs, thereby expanding the test coverage and enhancing the detection of vulnerabilities that traditional methods may overlook. The stochastic nature of fuzz testing allows for the exploration of input space that deterministic approaches cannot adequately address. For instance, while deterministic testing may only examine typical GPS signal structures, fuzz testing can introduce invalid signal formats, erroneous message sequences, and extreme variations in signal parameters, exposing weaknesses in the receiver's handling of atypical situations.

The primary advantage of fuzz testing lies in its ability to simulate dynamic and unpredictable environments, making it a vital tool for improving GNSS receiver resilience. As GNSS systems face increasingly sophisticated spoofing and jamming attacks, the capacity to reveal vulnerabilities through the exploration of unanticipated input scenarios becomes essential for ensuring operational integrity. This expanded coverage is critical not only for enhancing security but also for facilitating the development of more robust GNSS architectures capable of adapting to emerging threats.

Integrating fuzz testing into existing GNSS testing frameworks necessitates careful consideration of several practical aspects. First and foremost, the computational requirements associated with fuzz testing can be considerable, particularly when employing advanced techniques such as evolutionary fuzzing or concolic testing. These approaches often necessitate substantial computational resources to generate and evaluate a wide range of inputs effectively. Organizations may need to invest in robust testing infrastructure or utilize cloud-based solutions to scale their fuzz testing efforts, ensuring they can accommodate the resource-intensive nature of these techniques.

Moreover, input modification for fuzz testing requires a thorough understanding of GNSS signal structures and operational protocols and effort to include it in existing testing environment. Developing effective fuzzing inputs entails not only generating random values but also ensuring that these inputs reflect realistic scenarios while challenging the receiver's capabilities. Collaboration among software developers, system engineers, and testing teams is crucial to achieve this balance. By fostering interdisciplinary cooperation, teams can design fuzzing inputs that adequately stress-test the system while remaining grounded in the realities of GNSS operations.

Evaluating the outcomes of fuzz testing presents its own set of challenges. Unlike deterministic tests, which yield predictable results based on known input parameters, fuzz tests generate a diverse array of responses that require sophisticated analysis techniques. To effectively interpret the findings, testing teams must implement comprehensive metrics that encompass error rates, signal integrity, and state coverage. Metrics such as transition coverage can help assess how well the receiver navigates through its state space under various inputs, while error tracking can identify specific failure modes that require attention.

To facilitate the analysis of extensive data generated during fuzz testing, automated analysis tools can play a pivotal role. By employing machine learning algorithms or statistical analysis techniques, testing teams can sift through the data to identify critical weaknesses in GNSS receivers, allowing for data-driven decision-making in the refinement of receiver designs and testing protocols.

III. PROPOSED FRAMEWORK FOR GNSS FUZZ TESTING

Overview of the GNSS-Fuzz Testing Framework

The GNSS-Fuzz Testing Framework aims to bolster the robustness of GNSS receivers by incorporating both black-box and white/grey-box testing approaches. The black-box methodology approach is invaluable for identifying vulnerabilities that could be exploited by external attackers. In contrast, the white-box methodology provides a deeper examination of the GNSS receivers internal operations, helping to pinpoint implementation-specific weaknesses, logic errors, and potential flaws that may not be observable through black-box testing.

By synthesizing these two methodologies, the GNSS-Fuzz Testing Framework facilitates a comprehensive evaluation of the receiver's resilience against a wide spectrum of potential threats, including spoofing and jamming. This integrated approach ensures that both the functional performance and security robustness of the GNSS receiver are rigorously assessed, thereby providing meaningful insights that can guide subsequent improvements and refinements.

Model-Based Fuzz Testing for GNSS Systems

This research focuses on the development and application of model-based fuzz testing as an innovative approach to evaluating the performance and robustness of GNSS receivers. The main characteristic of this approach lies in developing a model of the system to be tested, and then using this model to guide the fuzzy testing process. GNSS receivers, especially in critical applications such as aviation, are vulnerable to a variety of threats, including signal spoofing, jamming, and environmental interference. While traditional deterministic testing methods are effective for compliance with established standards, they often fail to expose vulnerabilities arising from unexpected inputs or environmental variables. Model-based fuzz testing, however, extends beyond conventional software testing by offering a system-level testing approach that aligns with the unique challenges of GNSS systems.

A core component of this framework is the use of diverse models to represent the dynamic behavior of GNSS receivers. These models encompass both software and hardware components, offering a holistic view of system behavior. By leveraging these models, fuzz tests can be generated systematically, allowing for a comprehensive evaluation of GNSS receivers' performance, fault tolerance, and resilience under adverse conditions.

The primary objectives of this research are as follows:

- **Development of Model-Based Testing Tools:** Design and implement tools that utilize diverse system models, such as functional, behavioral, and data flow models, to automate fuzz test generation. These tools will focus on both valid and malformed input sequences to comprehensively evaluate system performance.
- **Assessment of GNSS Receiver Robustness:** Apply fuzz tests to real-world GNSS receiver models to simulate threats like signal degradation, environmental interference, and spoofing, assessing the receivers' resilience in these scenarios.
- **Performance Benchmarking:** Measure the effectiveness of fuzz testing relative to traditional deterministic approaches, using key performance metrics such as transition coverage, error rates, and resource utilization.

In model-based fuzz testing, the GNSS receiver is modeled across multiple layers, each representing a different aspect of system behavior. Testing at each of these layers helps to identify specific vulnerabilities and provides insights into different facets of system performance and security. This multi-layered approach aligns closely with the principles of Model-Based Systems Engineering (MBSE), a well-established methodology in systems engineering that leverages models to develop and manage complex systems. By structuring the GNSS receiver model into these five layers—system architecture, functional, behavioral, data flow, and simulation—the methodology adopts MBSE elements to guide the testing process systematicallyEstefan (2008).

Consequently, the framework can be characterized as Model-Based Fuzz Testing (MBFT), wherein MBSE principles and fuzz testing techniques are integrated to evaluate system robustness and reliability. MBFT employs MBSE models not just for system design but as foundational elements driving fuzz test generation and execution. This integration of MBSE and fuzz testing

provides a structured yet innovative way to uncover vulnerabilities and optimize GNSS receiver performance under a variety of conditions. These layers include:

- **System Architecture Layer** At this layer, the focus is on the high-level design and structure of the GNSS receiver, encompassing hardware components (e.g., antennas, processors) and software modules (e.g., navigation algorithms). Testing at this level verifies that the system meets key performance criteria such as scalability, fault tolerance, and robustness.These models that display the system level have to describe the interaction and communication between various components, offering a comprehensive understanding of the overall system behavior.
- **Functional Layer** The functional layer models specific tasks and operations performed by the GNSS receiver, such as signal acquisition, tracking, navigation computation, and fault detection. Testing at this layer ensures that the receiver's functions meet expected benchmarks under a variety of operational conditions. Therefor Control Flow Models, State Machine Models and Data Flow Models can be used. These models help verify the correct operation of functions and the receiver's response to different input sequences, ensuring functional integrity.
- **Behavioral Layer** The behavioral layer models the system's dynamic responses, including state transitions and reactions to external inputs. This layer is vital for evaluating how the receiver adapts to changing conditions in real time, such as transitions between signal acquisition and tracking, or the system's response to error recovery. Testing at this layer is particularly useful for assessing robustness in edge-case scenarios. The models used in this layer include State Machine Models, Temporal Logic Models or Finite State Machines (FSM) These models represent system states and their transitions, ensuring the receiver's behavior aligns with operational expectations even in stressful or anomalous conditions.
- **Data Flow Layer** The data flow layer focuses on the movement of data, such as satellite signals and navigation solutions, within the system. Testing at this level evaluates how efficiently the system handles data, performs error correction, and manages data transformations. The model used at this layer is could be for example a Data Flow Model. These models describe how data flows and is transformed throughout the system, providing insights into the accuracy and efficiency of data handling processes.
- **Simulation Layer** The simulation layer replicates the behavior of the GNSS receiver under controlled or extreme conditions, such as spoofing, jamming, or signal blockage. Testing at this layer is crucial for assessing system resilience in scenarios that may be unsafe or impractical to simulate in real-world environments. These models simulate real-world threats to evaluate the robustness of the GNSS receiver against attacks such as jamming and spoofing.

In this context, the system architecture layer and data model are particularly suitable for black-box fuzzing, as they describe the system's high-level interaction with its environment. Black-box fuzzing operates without knowledge of the system's internal mechanisms, making these models ideal for generating random but controlled inputs to test system responses in uncertain or adversarial environments.

On the other hand, the functional, behavioral, and simulation layers rely on models of internal system components, such as state machine models and temporal logic representations. These models enable grey-box or white-box fuzzing, where some or full knowledge of the system's internals is used to guide the test generation. This approach allows for targeted testing, focusing on edge cases, fault recovery, and resilience under specific adverse conditions.

By combining these methodologies, model-based fuzz testing offers a comprehensive evaluation framework that leverages the strengths of black-box and grey/white-box testing techniques. It ensures both high-level system robustness and detailed validation of internal processes under a variety of test conditions.

By integrating these layers within the GNSS-Fuzz Testing Framework, a comprehensive evaluation of the GNSS receiver's performance, resilience, and robustness can be achieved. Each layer offers a unique perspective on the system's behavior, and the corresponding models enable simulation of a wide range of conditions, including both typical and extreme operational scenarios. This multi-layered approach ensures that the system is not only functionally correct but also capable of performing reliably under adversarial conditions, such as spoofing and jamming.

This research aims to address critical gaps in current GNSS testing methodologies by:

- Providing systematic tools for uncovering vulnerabilities that could compromise GNSS receiver functionality in real-world scenarios.
- Enhancing the resilience and reliability of GNSS receivers, particularly in the face of evolving threats like advanced spoofing techniques.
- Informing stakeholders—such as certification authorities, manufacturers, and researchers—about the potential and limitations of fuzz testing in the GNSS domain.

The integration of fuzz testing within the GNSS receiver development process is expected to significantly improve the robustness of these systems, ensuring they remain secure and reliable in the face of emerging threats.

IV. GNSS USE CASE EXAMPLE

This section describes an example of applying fuzz testing to a GNSS receiver using a black-box approach. In this method, the system is tested by interacting with the hardware directly, without utilizing formal models to represent the system's internal behavior or structure. Grammar-based and mutation-based techniques were employed to generate input scenarios for testing the receiver.

Equipment under test (system example)

Currently, the Collins GNSS Product Line is developing a dual frequency multi-constellation prototype including GPS and Galileo dual frequency processing plus Satellite Based Augmentation Systems inherited from legacy GLU-2100 equipment which is compliant with RTCA DO-229F MOPS.

Figure 1: GLU-2100

This new prototype is a new SBAS DFMC GNSS receiver compliant with a new EUROCAE/RTCA standard, the ED-259. The GLU-2100 is an ARINC 755-4 compliant digital MMR that supports the following TSO functions:

- VOR receiver compliant to DO-196 and ED-22B
- MB receiver functionality compliant to DO-143 and 1/WG7
- ILS Localizer receiver functionality compliant to DO-195 and ED-46B
- ILS Glideslope receiver functionality compliant to DO-192 and ED-47B (up to CAT IIIb installation supported)
- GNSS:
  - L1 GPS – used for navigation
  - SBAS Navigation and Landing compliant to DO-229E – Satellite Based Augmentation System, wide area or regional (EGNOS, GAGAN, MSAS, etc.) augmentation to the GPS navigation system enables higher precision and integrity data to be used
  - GBAS Navigation and Landing (CAT I) compliant to DO-253C – Ground Based Augmentation Systems that supports local area augmentation from a ground station that enables very high precision and integrity data to be used

Test Bench

GLU-2100 tests are based on simulations using a GNSS signal simulator. The test bench is composed of:

- Spirent GSS-9000 simulator, configured with PosApp software v8.01, which allows precise control of signal generation and scenario design. The simulator has its own RF controller and signal generator.
- Collins GLU-2100 receiver, integrating a DFMC-SBAS software prototype, capable of processing dual-frequency GNSS signals and advanced SBAS corrections.

The Spirent simulator allows the user to record, in real-time, three different types of logs (also known as truth files) directly from the simulation engine:

- SBAS L1/L5 message dumps, containing detailed SBAS correction and integrity information.
- GPS LNAV/CNAV navigation message dumps, providing raw broadcast data for GPS satellites.

- Galileo I-NAV/F-NAV navigation message dumps, capturing the primary and secondary navigation message structures for Galileo satellites.

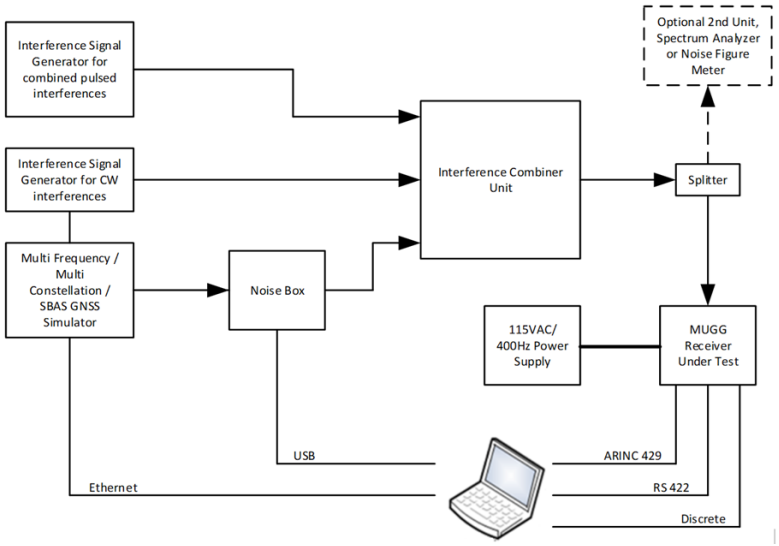


Figure 2: Schema of the Testbench

The information contained in these files is critical for fuzzing, as it provides a precise baseline of the data broadcasted by the simulator and received by the GLU-2100. This allows for the identification and injection of targeted modifications during testing.Fuzzing is performed using the Spirent simulator's "Data Modification" feature, which enables fine-grained control over broadcast messages. This feature works by allowing users to send a series of modification requests to the Spirent simulation engine, instructing it to transmit specific messages with altered content, timing, or format. Modifications can target various message fields, such as:

- Pseudo-range corrections in SBAS messages.
- Ephemeris or almanac parameters in GPS/Galileo navigation messages.
- Timing and clock offset parameters critical for receiver positioning algorithms

The Spirent GSS-9000 has advanced capabilities enabling the creation of highly customized test scenarios, simulating edge cases such as multipath effects, weak signals, or spoofing attempts, which are essential for evaluating receiver robustness under diverse conditions. Additionally, the ability to generate and store truth files ensures complete traceability of all test cases and their impacts, allowing for thorough post-analysis. Finally, the system's scalability supports testing multiple satellites and signal types simultaneously, making it particularly suited for multi-constellation, multi-frequency testing. A key strength lies in its "Data Modification" feature, which enables the user to target and alter specific parts of navigation messages. This granular control is particularly valuable for fuzzing, as it allows testers to inject precise modifications—such as altering ephemeris data, pseudo-range corrections, or integrity bits—without disrupting the entire message structure. Such targeted fuzzing can uncover vulnerabilities in how the GLU-2100 process specific message fields.

To implement our fuzzing framework, we leveraged a custom-built fuzzing tool tailored to the specific requirements of GNSS receiver testing. This tool incorporates a combination of techniques, including mutation-based and grammar-based fuzzing, to generate diverse and realistic input sequences. By employing a combination of these techniques, we were able to effectively explore the input space and identify potential vulnerabilities in the GNSS receiver under test.

Then, the generated navigation data arewill be injected in a SW/HW signal generator which is here represented by the Spirent box. The Spirent is a GNSS constellation simulator which generate RF signals.

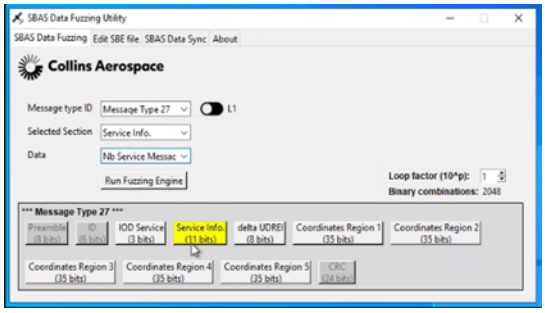
Fuzzing GNSS messages

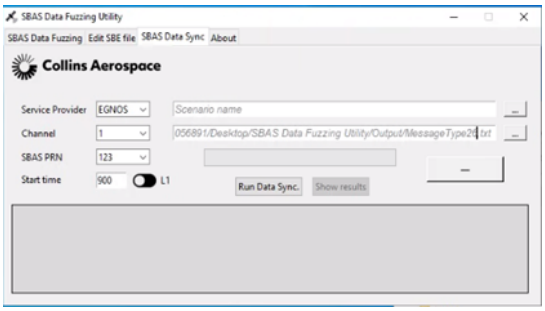
In this fuzzing approach, scenarios are used as inputs, which are sequences of valid signals representing real-world satellite constellations (GPS, Galileo, SBAS, etc.). These scenarios are created without involving formal models but focus on simulating the expected real-world conditions. This allows for testing how the receiver reacts to different sets of input data corresponding to GNSS signals. On the right scheme, we propose to generate as inputs both scenarios and events. In the scenarios, we will

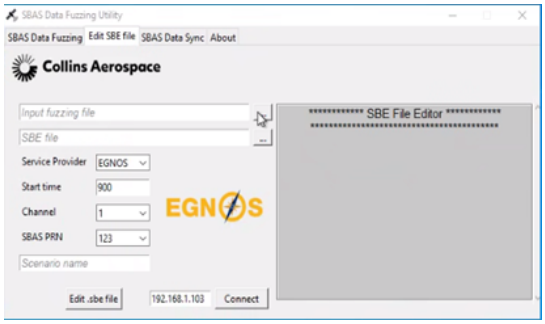
consider all the core constellations satellites signals (GPS and Galileo signals from medium Earth orbits) that are the first targets of spoofers. The GEO satellites which are the SBAS satellites are also considered to provide corrections and integrity in covered areas like in CONUS for the US WAAS or the ECAC for the European EGNOS augmentation systems.

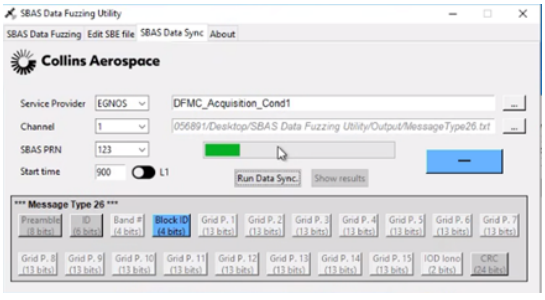
Figure 3: Fuzzing Schema on Testbench

The scenarios include valid signals which refer to signals that are broadcast from the authentic satellites and with navigation messages that embed satellites ephemeris, almanac, Universal Time Coordinate parameters, atmospheric ionospheric parameters needed for satellites position determination and ranging satellites antenna to receiver antenna corrections to provide navigation solutions.
Fuzzing technique is applied to messages generation in scenarios with a dedicated tool developed by Collins and illustrated below:









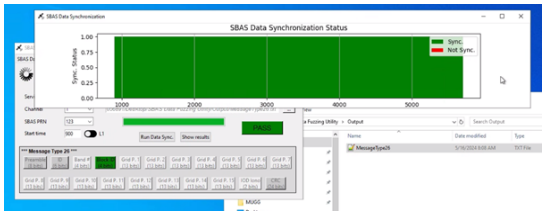


Figure 4: Collins Aerospace Fuzzing Tool

Improved fuzzing model applied to GNSS messages scenarios

The fuzzing model used here is a mutation-based and grammar-based fuzzing approach. These techniques generate realistic, varied inputs (scenarios) based on the actual message structures used in GNSS systems. By making random or targeted changes to the messages (mutation) and following the structure of valid GNSS messages (grammar-based), this approach ensures that the fuzzing inputs are both realistic and diverse. In this paper, we address the following SBAS message fuzzing example.In the following table, the SBAS L1 messages main contents are recalled:

Table 2: SBAS L1 Signal Types and Contents

Type	Contents
0	Do not use this SBAS L1 signal for safety applications
1	PRN Mask assignments, set up to 51 of 210 bits
2 to 5	Fast corrections
6	Integrity information
7	Fast correction degradation factor
8	Reserved for future messages
9	GEO navigation message (X, Y, Z, time, etc.)
10	Degradation Parameters
11	Reserved for future messages
12	SBAS Network parameters Time/UTC offset

Type	Contents
13 to 16	Reserved for future messages
17	GEO satellite almanacs
18	Reserved for future messages
19	Reserved for future messages
20	Reserved for future messages
21	Reserved for future messages
22	Reserved for future messages
23	Reserved for future messages
24	Reserved for future messages
25	Reserved for future messages

The next figure presents the logic between SBAS L1 messages. MT 6 is related to integrity and contain IODF information.

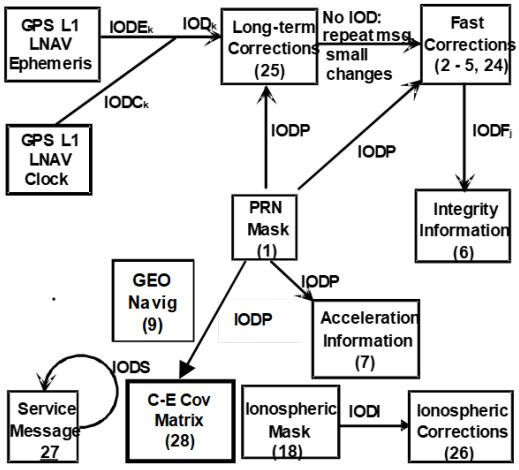


Figure 5: Logic between SBAS L1 messages

MOPS ED-259A and DO-229F provide precise guidance on how to apply SBAS corrections and compute SBAS based horizontal and vertical protection levels. MOPS specifies timeout periods beyond which SBAS messages should not be used. When a MT

6 message with an IODF=3 is received the GNSS software latches the UDREI for all the active slots that belong to the block whose IODF is set to 3. It the next MT6 received after the latch event also has an IODF of 3 then the new MT6 will unlatch the previous UDREI and latch new UDREI. These latching conditions can cause a GNSS receiver to use UDREI beyond the timeout period.(DO2 (2020),EUR (2020))

Section	Name	Length	Location start	Location end	Scale factor	Range min	Range max	Unit	Description
Common Header	Preamble	8	0	7	-	-	-	-	Preamble (see section B.3.3.4)
	Message Id.	6	8	13	1	0	63	-	Message Type Identifier (= 6)
IODF	IODF 2	2	14	15	1	0	3	-	IODF for MT 2 for satellites from 1 to 13
	IODF 3	2	16	17	1	0	3	-	IODF for MT 3 for satellites from 14 to 26
	IODF 4	2	18	19	1	0	3	-	IODF for MT 4 for satellites from 27 to 39
	IODF 5	2	20	21	1	0	3	-	IODF for MT 5 for satellites from 40 to 51
UDREI	UDREI 1	4	22	25	1	0	15	-	UDREI Indicator for 1st PRN selected in the mask
	to UDREI 51	4	222	225	1	0	15	-	to UDREI Indicator for 51st PRN selected in the mask
Common Trailer	Parity	24	226	249	-	-	-	-	Cyclic Redundancy Check (see section B.3.3.3)

Figure 6: SBAS IODF

Results and Analysis

As an example, the MT6 IODF (values from 0 to 3) was fuzzed as plotted below:

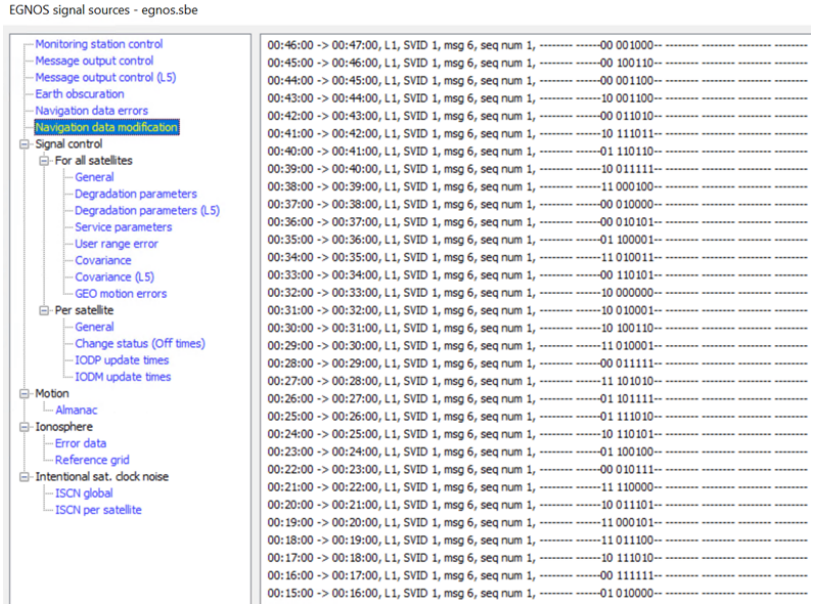


Figure 7: Fuzzed MT6 IODF (values from 0 to 3)

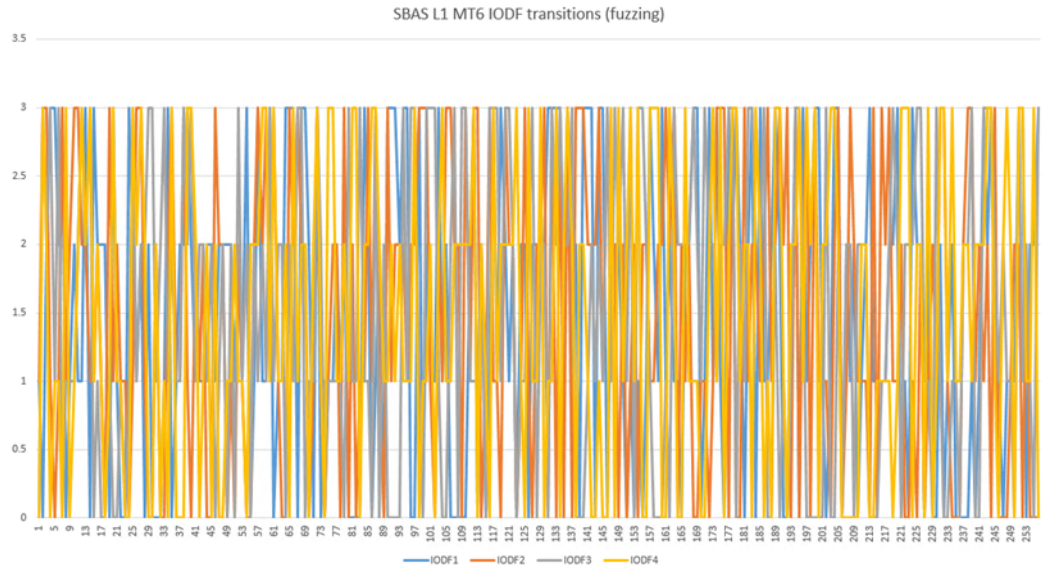


Figure 8: SBAS L1 MT6 IODF transition (fuzzing)

The GLU equipment behavior was analyzed and specific results are shared hereafter. The interest is not only to fuzz the data contents but also the priorities of the messages as well as the intervals between to messages sending. The messages contents are fuzzed according to a uniform law whereas the time intervals follow a Poisson law. Hereafter, the GEO PRN CNO estimation was plotted considering some various priorities examples, indeed, the equipment acquires GEO signals and then demodulates the data only if the PRN numbers are confirmed, so it tracks it only if confirmed and therefore, the C/N0 estimation is provided during tracking only in case the correct acquisition is confirmed. Some C/N0 estimations were provided hereafter when fuzzing the MT6 priorities, the expectation is that the GEO signal is tracked whatever the priority:

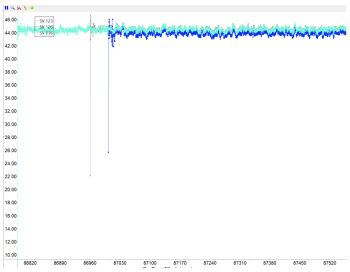


Figure 9: Reference Scenario - MT6 every six second with priority 2

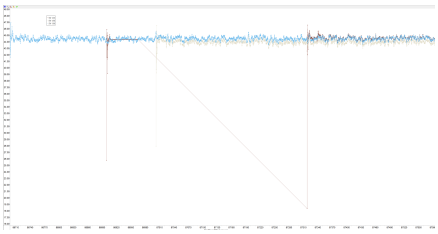


Figure 10: MT6 every second with priority 2

Figure 11: MT6 every second with priority 10

The Carrier-to-Noise ratio (C/No) was monitored across various SBAS message configurations as part of a fuzz testing campaign to evaluate GNSS receiver performance. On the X-axis, the GPS weeks are represented in seconds, and on the Y-axis, the Carrier-to-Noise Ratio (C/No) is displayed in decibels (dB). These tests aimed to explore the receiver's behavior under diverse and unexpected conditions, showcasing fuzz testing as a robust approach for identifying performance trends and potential vulnerabilities.

The results indicate that, while minor fluctuations in C/No levels were observed, the overall signal strength remained consistently high across all tested scenarios. This highlights the receiver's resilience to increased message loads and varying IODF strategies (random or 3-2 pattern), as uncovered by the fuzz tests. Fuzz testing revealed that increasing the frequency of MT6 messages significantly improves the stability and continuity of the navigation state, demonstrating the receiver's dependence on frequent updates. Conversely, increasing message priority had a less pronounced effect on navigation stability. The specific IODF configuration also showed minimal impact under stable signal conditions. These findings emphasize the receiver's capacity to handle diverse configurations, as effectively stress-tested through fuzzing. On the X-axis, the GPS weeks are represented in seconds, and on the Y-axis, the Navigation State.

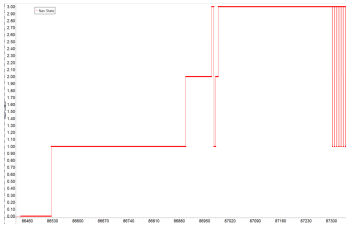


Figure 12: Reference Scenario - MT6 every six second with priority 2

Figure 13: MT6 every second with priority 2

Figure 14: MT6 every second with priority 10

This study demonstrates the value of fuzz testing for systematically evaluating GNSS receiver performance. By simulating high-frequency, priority-based, and randomized configurations, fuzz testing enabled a comprehensive analysis of receiver behavior. To optimize receiver performance, rate limiting of MT6 messages, priority-based processing, and adaptive techniques should be considered. Further fuzz testing under challenging environmental conditions, such as signal interference or multipath effects, will be crucial to fully assess receiver robustness and identify potential bottlenecks.

In this study, we applied a black-box fuzz testing approach to evaluate a GNSS receiver. Initially, we employed a random fuzzing technique, but we recognized its limitations, as it often generated a large number of irrelevant test cases. To address this, we developed an improved fuzzing technique that combines elements of mutation-based and grammar-based fuzzing. This approach allowed us to generate more targeted and effective test cases, leading to a significant improvement in vulnerability detection efficiency.

While our current approach provided valuable insights into the receiver's response to randomized inputs, we believe that incorporating a model-based strategy could further enhance the effectiveness of fuzz testing. By modeling the system beforehand, we could simulate a wider range of scenarios and potential vulnerabilities, allowing for a more comprehensive assessment of the receiver's robustness. This would not only improve the efficiency of fuzz testing but also uncover vulnerabilities that may be missed in a purely black-box testing approach, ultimately providing a more thorough evaluation of the real receiver's resilience to threats like spoofing and jamming.

V. CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

This paper has presented a comprehensive framework for fuzz testing GNSS receiver involving a MBSE (Model based System Engineering) approach and black-box testing for managing fuzzy testing. This approach has been applied for fuzz testing to GNSS receivers, a critical component of modern infrastructure.

By systematically generating and injecting diverse, often malformed, inputs, fuzz testing has proven to be an effective technique for uncovering vulnerabilities that may be missed by traditional testing methods.

The proposed framework, integrating both black-box and white-box techniques, offers a robust approach to assessing the security and reliability of GNSS receivers. By leveraging state-of-the-art fuzzing methodologies and advanced analysis techniques, researchers and industry practitioners can identify potential weaknesses and develop countermeasures to mitigate emerging threats, such as spoofing and jamming attacks.

While this paper has made significant strides in advancing the application of fuzz testing to GNSS receivers, further research is necessary to refine and extend these techniques. Areas for future exploration include:

- **Enhanced Scenario Generation:** Developing techniques to generate more realistic and complex scenarios, including those that simulate real-world threats like spoofing and jamming attacks.
- **Intelligent Fuzzing:** Exploring the use of artificial intelligence and machine learning to optimize the fuzzing process, focusing on high-risk areas and reducing the number of unnecessary test cases.
- **Integration with GNSS Standards:** Aligning fuzz testing methodologies with relevant GNSS standards to ensure compliance and interoperability.
- **Collaboration and Knowledge Sharing:** Fostering collaboration between academia, industry, and regulatory bodies to share knowledge, best practices, and standardized testing methodologies.

By addressing these research challenges, we can further strengthen the security and reliability of GNSS receivers, safeguarding critical infrastructure and ensuring the accuracy of positioning, navigation, and timing services worldwide.

ACRONYMS

Table 3: List of Acronyms

Acronym	Full Name
ARINC	Aeronautical Radio Incorporated
CPU	Central Processing Unit
GEO	Geostationary Earth Orbit
GLU	Global Landing System
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
IODP	Issue Of Data PRN mask
LNAV	Lateral Navigation
MT	Message Type
PRN	Pseudorandom Noise
RTCA	Radio Technical Commission for Aeronautics
SBAS	Satellite-Based Augmentation System

REFERENCES

(2020). Do-229f minimum operational performance standards (mops) for global positioning system/satellite-based augmentation system airborne equipment. Technical report, RTCA.

(2020). Eurocae ed-259 minimum operational performance standard for galileo – global positioning system – dual frequency multi constellation satellite-based augmentation system airborne equipment. Technical report, EUROCAE.

Estefan, J. (2008). A survey of model-based systems engineering (mbse) methodologies, rev. b. Technical Report INCOSE-TD-2007-003-02, International Council on Systems Engineering (INCOSE), San Diego, CA, USA.

Pargaonkar, S. (2023). A comprehensive review of performance testing methodologies and best practices: Software quality engineering. *International Journal of Science and Research (IJSR)*, 12:2008–2014.

Qian, R., Zhang, Q., Fang, C., and Guo, L. (2023). Investigating coverage guided fuzzing with mutation testing. In *Proceedings of the State Key Laboratory for Novel Software Technology*, Nanjing University, China. Explores the intersection of mutation testing and coverage-guided fuzzing, highlighting advanced methodologies to improve fault detection in software systems.

Rawat, S., Mathur, V., Davalos, R. M., and Mounier, L. (2017). Vuzzer: Application-aware evolutionary fuzzing. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, USA. Internet Society. Accessed 2024-12-01.

Sen, K., Marinov, D., and Agha, G. (2005). Cute: A concolic unit testing engine for C. *ACM SIGSOFT Software Engineering Notes*, 30(5):263–272. Accessed 2024-12-01.

Sutton, M., Greene, A., and Amini, P. (2007). *Fuzzing: Brute Force Vulnerability Discovery*. Addison-Wesley Professional.

Yang, D., Zhang, Y., and Liu, Q. (2012). Blendfuzz: A model-based framework for fuzz testing programs with grammatical inputs. In *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 1070–1076, Liverpool, UK.