Computers & Security 171 (2026) 105128

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Full length article

CCSDS-based firewall for modern satellite system

Woohyeop Im [a], Gyeongje Oh [a], Shinichi Kan [b], Kosuke Ito [b], Hikohiro Lin [b], Sayeon Kim [c], Jinwoo Jeong [c], Sangbom Yun [c], Daehee Jang [a,*]

[a] Kyung Hee University, Republic of Korea
[b] GMO Cybersecurity by Ierae, Inc, Japan
[c] LIG Defense&Aerospace, Republic of Korea

ARTICLE INFO

Dataset link: https://github.com/4g6dcbd21B/CCSDS-based-Firewall, CCSDS-Based Firewall Implementation and Evaluation Dataset (Original data)

Keywords:
CCSDS
Satellite security
Space systems
Firewall
Intrusion detection
Embedded systems
Protocol analysis
CubeSat security
Cybersecurity
Anomaly detection

ABSTRACT

As satellite missions increasingly adopt standardized protocols and software-defined infrastructure, protocol compliance alone cannot fully prevent operational misconfigurations and security flaws, motivating additional mechanisms to monitor and validate Consultative Committee for Space Data Systems (CCSDS) traffic at the onboard ingress boundary.

This paper addresses the critical need for an onboard security audit system capable of inspecting CCSDS traffic for semantic inconsistencies and policy violations in real time, even within correctly encrypted links that may be compromised by administrative errors. To this end, we propose a satellite-onboard CCSDS-compliant firewall that observes TC/TM traffic at the OBC and automatically detects insecure configurations, nonstandard customizations, missing or improperly implemented protocol security, and anomalous traffic conditions, based on predefined risk scenarios derived from the CCSDS standards and operational recommendations.

We implemented and evaluated our system on both a simulation-based environment and two flight-proven CubeSat hardware platforms. To ensure mission reliability, we applied architecture-specific optimizations to minimize performance overhead. Experimental results demonstrate the feasibility of integrating this onboard security-audit capability within the resource constraints of CubeSat platforms.

1. Introduction

Modern space systems increasingly rely on software-defined ground infrastructure, network-connected mission operations, and growing onboard autonomy. These trends can expand the cyberattack surface of telemetry, tracking, and command (TT&C) pathways and increase the likelihood that unexpected inputs reach spacecraft flight software (Ahmad et al., 2022; Kavallieratos and Katsikas, 2023; Ansong et al., 2025; Wu et al., 2023).

Real-world incidents show that satellite cyberattacks have caused loss or disruption. Reported cases include unauthorized access to the ROSAT X-ray observatory that issued damaging commands, intrusions through a commercial ground station affecting Landsat-7 and Terra, and the 2022 KA-SAT/Viasat cyberattack that disabled large numbers of terminals (Soesanto, 2021; U.S.China Economic and Security Review Commission, 2011; Kang et al., 2024). These incidents indicate that compromise of the TT&C chain can translate into mission loss or service denial. Moreover, as space-sector companies grow, space data and operational access are being re-shaped based on standardized protocols and COTS components, reducing the effectiveness of security through obscurity and making it easier for adversaries to emulate uplink interfaces. As a result, security by obscurity alone is no longer sufficient (European Union Agency for Cybersecurity (ENISA), 2024; Stensberg, 2024). Even worse, cloud-based ground-station services further expand the attack surface; a breached or misconfigured provider can become a gateway for malicious uplink traffic (Stensberg, 2024; European Union Agency for Cybersecurity (ENISA), 2025).

In response to these developments, since the new-space era (Manulis et al., 2021), the majority of missions exchange telecommand (TC) and telemetry (TM) using protocols from the Consultative Committee for Space Data Systems (CCSDS) (CCS, 2021a,c, 2020), which has become the de facto standard for space system communications and security.

In space communication, due to its vast distance and RF signal spreading issues, link layer protocol encryption (e.g., Space Data Link Security (SDLS)) is crucial to prevent eavesdropping attacks. When SDLS is securely enabled with correct key management and anti-replay, many command-injection and replay scenarios are mitigated. However, these measures do not remove all operational risk at the onboard

* Corresponding author.
E-mail address: daehee87@khu.ac.kr (D. Jang).

https://doi.org/10.1016/j.cose.2026.105128
Received 26 March 2026; Received in revised form 5 August 2026; Accepted 20 August 2026
Available online 28 August 2026
0167-4048/© 2026 Elsevier Ltd. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

ingress boundary. In fact, CCSDS guidance explicitly discusses operational cases where a link (or selected logical channels) may or may not operate in a non-protected *Clear Mode* (e.g., temporarily disabling encryption in recovery mode), and how to properly manage encryption keys.

However, the breadth of the CCSDS standards makes correct configuration and implementation difficult in practice. Nonstandard customizations, omitted or incorrectly implemented protections, cryptographic misconfiguration, and operational errors can therefore produce TC/TM traffic that is syntactically plausible but inconsistent with the applicable standard or mission security policy. This motivates an onboard security-audit system (e.g., a firewall) that checks observable CCSDS traffic against explicitly configured protocol and protection requirements.

To address this problem, this paper proposes a satellite-onboard CCSDS security-audit system (for simplicity, the CCSDS firewall) that automatically detects and reports covered instances of four classes of security-relevant conditions: insecure configurations, nonstandard customizations, missing or improperly implemented protocol protection, and anomalous traffic. Since the firewall operates inside the satellite system in space, the main challenge is minimizing performance degradation and reducing resource consumption, including power usage and memory footprint, which is crucial for space mission reliability. Based on this motivation, the main contributions of this paper are threefold.

First, based on CCSDS standards and operational recommendations, we define a set of satellite security-risk scenarios—covering insecure configurations, nonstandard customizations, missing protocol protections, and anomalous traffic—to demonstrate the needs and impact of TM/TC security inspection. Second, with these scenarios in mind, we design and implement a lightweight satellite-oriented firewall that complements existing CCSDS security mechanisms across three CubeSat On-Board Computer (OBC) environments: (1) a NASA simulation-based OBC, (2) the EnduroSat OBC, and (3) the NanoMind A3200 OBC. The first environment is based on simulation; however, the remaining two environments are based on actual space-grade platforms obtained from GomSpace (GomSpace, 2026) and EnduroSat EnduroSat (2026). Finally, to evaluate feasibility and optimize performance, we apply architecture-specific optimizations to each OBC environment (except for the simulation) and evaluate detailed performance such as changes in CPU loads, execution time, board temperature, and so forth. Through these experiments, we analyze design trade-offs tailored for space and resource-constrained satellite environments.

Taken together, this study demonstrates the practical integration of CCSDS-aware security inspection within resource-constrained satellite platforms. By deploying the firewall across multiple OBC environments, we show that onboard protocol-aware inspection can be achieved with manageable overhead while remaining compatible with existing cryptographic protections such as SDLS.

The remainder of this paper is organized as follows. Section 2 presents the necessary background and the threat model. Section 3 reviews related work and positions the proposed firewall with respect to existing protections. Section 4 describes the firewall design, including the inspection boundary and rule structure. Section 5 describes the implementation. Section 6 presents the performance evaluation of the firewall across three deployment environments, including its security efficacy. Section 7 discusses limitations and future work. Section VIII provides discussion and implications for deployment. Section IX concludes the paper.

2. Background & threat model

2.1. CCSDS

The Consultative Committee for Space Data Systems (CCSDS) defines a comprehensive set of standards that govern space mission systems, spanning telemetry (TM), telecommand (TC), tracking, and their associated communication and data handling architectures. These standards specify packetization and framing mechanisms, while CCSDS security standards such as SDLS provide authentication, encryption, and replay protection services for TT&C communications (CCS, 2021b, 2025, 2021a,c, 2020).

CCSDS security mechanisms, such as SDLS, are designed to protect space links against unauthorized access and manipulation. However, CCSDS standards focus primarily on ensuring syntactic correctness and cryptographic protection of communication data units. They do not fully address behavioral anomaly detection or mission-specific security policies at the onboard command ingress point.

In the CCSDS protocol stack, TC and TM use different coding units. For TC, CLTU (Communications Link Transmission Unit) is the transmit unit, and it carries a TC Transfer Frame inside. For TM, CADU (Channel Access Data Unit) is the receive unit, and is made of an Attached Sync Marker (ASM) and a TM Transfer Frame. Above the Transfer Frame, SPP (Space Packet Protocol) defines the packet unit that the OBC software processes. Therefore, CLTU/CADU represent the earliest CCSDS communication units arriving at the spacecraft communication interface and provide the protocol information used by the firewall.

As shown in Fig. 1, CLTU carries TC Transfer Frame between a Start Sequence and a Tail Sequence. Before parsing Transfer Frame, the receiver uses these markers to detect CLTU boundaries. In the Transfer Frame header, fields such as Version Number, Spacecraft ID, Frame Length, and Frame Sequence Counter are key checks for structural validity and sequence anomalies. When SDLS is applied, the protected region is usually the Transfer Frame Data Field, while the Transfer Frame header stays in plaintext for routing and protocol handling.

This paper builds upon CCSDS-defined communication and security standards and proposes an onboard CCSDS-compliant firewall that enforces protocol-level validation of TC/TM traffic.

2.2. Satellite OBC and payload

A typical satellite onboard system consists of an On-Board Computer (OBC) responsible for platform control and one or more payload subsystems responsible for mission-specific functions. The OBC executes flight software that handles TC processing, TM generation, attitude control, power management, and fault handling, while payload processors may operate either as tightly coupled modules or as semi-autonomous units connected via onboard buses.

In many CubeSat and small-satellite platforms, the OBC directly interfaces with the communication subsystem and serves as the primary entry point for uplinked TC packets. The CCSDS packets received are typically parsed, dispatched, and executed by flight software components with limited runtime isolation. Consequently, malformed or semantically unexpected TC packets that reach the OBC may directly affect critical control logic, even if they are syntactically compliant with CCSDS specifications. Therefore, the OBC represents a critical security boundary for satellite communication, and thus we position the proposed firewall within the OBC.

2.3. Ground station

The ground segment comprises mission operations software, ground stations, and supporting network infrastructure that generate TC and receive TM. Traditionally, ground stations were mission-specific and physically isolated systems. However, recent ground stations (especially for LEO satellites) increasingly adopt network-connected and cloud-based services to reduce operational cost and increase coverage (Manulis et al., 2021; Salim et al., 2024).

While such architectures improve scalability, they also expand the attack surface of the TM/TC chain. A compromised ground-station component, misconfigured mission script, or unauthorized operator access may result in the generation of TC packets that are formally valid but operationally unsafe. In these cases, the satellite cannot rely solely on the ground segment to enforce the correct security policies.

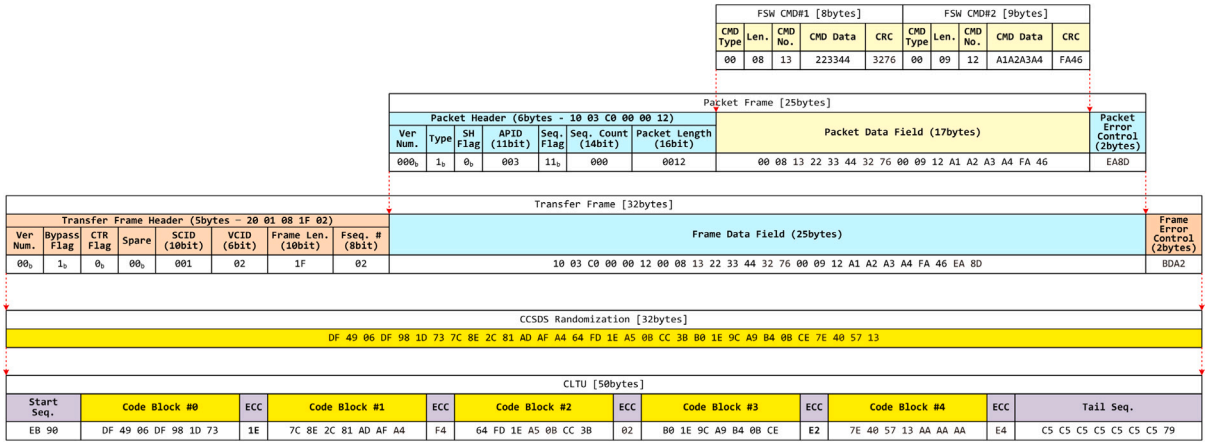


Fig. 1. Layered CCSDS telecommand encapsulation, with flight software (FSW) commands packed into a Space Packet, framed into a Transfer Frame, and segmented into CLTU code blocks with ECC and a tail sequence.

2.4. Space communication

Satellite communication fundamentally differs from terrestrial communication in that it is conducted over extremely long distances and is based on predictable orbital dynamics. In most Earth-orbiting missions, satellite positions can be estimated with high accuracy using publicly available Two-Line Element (TLE) data, enabling any sufficiently equipped third party to predict satellite passes and access the corresponding RF channels.

As a result, the physical broadcast nature of space links implies that telemetry downlinks are inherently observable, and telecommand uplinks are, in principle, reachable by unauthorized transmitters. Although cryptographic protections such as SDLS are intended to prevent unauthorized command execution, the openness of the RF environment means that TM/TC channels are continuously exposed to potential probing, replay attempts, and protocol-level experimentation (Pavur and Martinovic, 2022; Salim et al., 2024). Unlike terrestrial networks, satellites cannot easily rely on physical access control or geographic isolation to limit exposure.

From a security perspective, the onboard communication stack is responsible for all external inputs and outputs. In our system model, the proposed CCSDS firewall is positioned at this boundary, logically after RF demodulation and before the flight software TC dispatcher. The firewall inspects post-demodulation TC ingress and pre-modulation TM egress traffic before mission-critical logic is invoked.

2.5. Security scope and threat model

The security model considered in this paper is centered on observable conditions at the TT&C boundary rather than on attacker identity or intent. The conditions in scope are insecure configurations, nonstandard customizations, missing or improperly implemented protocol protection, and anomalous traffic. They may arise from developer or operator error, a mismatch between communicating components, channel effects, or deliberate manipulation.

For the deliberate-manipulation case, we assume that an adversary does not physically compromise the satellite hardware or break the cryptographic primitives used by CCSDS security mechanisms. The adversary may instead gain access to a ground-segment component, exploit a software or configuration weakness, or abuse a legitimate operational mode such as Clear Mode or a recovery procedure. These assumptions describe possible sources of the observed conditions; they are not labels that the firewall attempts to infer.

Note that the firewall does not attempt to identify the adversary or infer intent; it automatically detects and reports the insecure conditions themselves—insecure configurations, nonstandard customizations, missing or improperly implemented protocol security, and anomalous traffic—whether they are caused by an adversary exploiting them or by an operator's mistake. Based on this threat model, the firewall inspects only protocol-level information, including packet structure, indicators of cryptographic misuse, and traffic behavior. It does not interpret the operational meaning of telecommands. A structurally valid command that carries an in-range APID may therefore pass even when it is operationally illegitimate; auditing such operational-level violations is intentionally outside the protocol-level scope of the proposed firewall and is discussed further in Section 7.

2.6. Positioning against existing protections

Several established mechanisms already address aspects of satellite communication security. SDLS (CCS, 2022) provides cryptographic confidentiality, integrity, and replay protection at the data link layer. Command authentication schemes (Bader, 2024) verify the origin of telecommand transmissions. Onboard access control and anomaly detection approaches (Falco et al., 2021; Driouch et al., 2024) monitor system-level behavior for deviations from expected operational patterns.

The proposed CCSDS firewall complements these existing mechanisms. First, unlike SDLS, the firewall detects protocol-level anomalies when SDLS is misconfigured, inconsistently deployed, or temporarily disabled during recovery or maintenance procedures. Second, unlike mission-specific anomaly detection approaches, the firewall relies solely on observable CCSDS protocol fields, traffic behavior, and a compact deployment profile, making its inspection logic reusable across diverse satellite platforms. At deployment, the profile supplies the applicable CCSDS protocol parameters without requiring payload semantics or cryptographic keys.

Therefore, the proposed firewall is not intended to replace SDLS, authentication, or anti-replay enforcement. It is a lightweight audit layer that reports the misconfiguration, customization, protection, and traffic-anomaly conditions at the onboard ingress boundary.

3. Related work

This section reviews prior work most relevant to the proposed CCSDS firewall, including CCSDS security standards, protocol fuzzing for space systems, onboard and ground-segment intrusion detection, telemetry anomaly detection, RF-layer anomaly detection, and real-world satellite cybersecurity incidents. Table 1 compares the proposed firewall with closely related satellite and radio-frequency security approaches across five dimensions: deployment location, CCSDS awareness, hardware validation, payload-decryption requirements, and onboard feasibility. We include RF-layer detection approaches alongside

Table 1
Comparison of the proposed CCSDS firewall with related satellite security approaches across deployment location, CCSDS awareness, hardware validation, payload-decryption requirements, and onboard feasibility.

Approach	Deploy	CCSDS	HW Valid.	Payload decryption required	OBC feasible
SDLS (CCS, 2022)	Onboard + Ground	Yes	Yes	No	Yes
CANSat-IDS (Driouch et al., 2024)	Onboard	No	Simulation	Yes	Partial
TinyML CubeSat IDS (Fayyaz et al., 2026)	Onboard	No	Survey	Yes	Partial
NOS3/cFS IDS (Thebarge et al., 2022)	Onboard	Partial	Simulation	N/A	Not demonstrated
ML Telemetry (Hundman et al., 2018)	Ground	No	No	Yes	No
Space Protocol Fuzzing (Havermans et al., 2025)	Testing	Partial	No	No	No
Link-Segment Security Monitoring (Salim et al., 2024)	Space/Ground/Link	No	Survey	No	No
Proposed CCSDS Firewall	Onboard	Yes	Yes	No	Yes

protocol-layer approaches because both represent alternative points in the communication stack at which satellite security mechanisms can be placed, and contrasting them clarifies why protocol-level, CCSDS-aware inspection is necessary in addition to physical-layer monitoring.

3.1. CCSDS security standards and link-layer protection

CCSDS provides standardized approaches for protecting space links, including confidentiality, integrity, and authentication services at the data link layer. In particular, SDLS specifies how security services can be applied to CCSDS space links and cross-support settings (CCS, 2022, 2024). These standards mainly focus on how to apply cryptographic services correctly, but do not directly provide protocol-level rules for identifying insecure configurations, nonstandard customizations, or abnormal traffic patterns such as malformed traffic, replay-like behavior, or misconfigurations (Bader, 2024). Our work complements such standards by introducing protocol analysis rules that can operate even when payload decryption is not available or not desirable, and by providing rule-based audit logic that can be implemented as a firewall component.

3.2. Experimental security analysis and fuzzing of space protocols

Recent work has shown that satellite software and space communication stacks can contain practical security weaknesses, motivating systematic testing of satellite systems (Pavur and Martinovic, 2022; Jero et al., 2024; Willbold et al., 2023). Previous studies have analyzed real satellite software stacks and communication behaviors, highlighting attack surfaces on ground-to-space and onboard components (Salkield et al., 2023; Willbold et al., 2024a; Costin et al., 2023). In addition, protocol fuzzing efforts have emerged that explicitly target space communication protocols (including the CCSDS space packet protocol and related layers) to find parsing and implementation bugs (Havermans et al., 2025; Scharnowski et al., 2023; Willbold et al., 2024b). These works motivate the need for protocol-aware defenses and robust validation logic. Compared to fuzzing-oriented research that aims to discover vulnerabilities, our work focuses on deployable runtime detection rules in a CCSDS firewall, designed to flag observable protocol anomalies and operationally risky configurations.

3.3. Onboard and ground-segment intrusion detection for space systems

Several studies and reports discuss the principles of defense-in-depth for spacecraft cybersecurity, including onboard monitoring, intrusion detection concepts, and secure operational practices (Falco, 2019; Falco et al., 2021; Salim et al., 2024). More recently, embedded intrusion detection approaches for satellites have been proposed in the simulation-based environments (e.g., NOS3/cFS-based setups), demonstrating the feasibility of IDS-like monitoring in space mission contexts (Thebarge et al., 2022; Driouch et al., 2024). However, many approaches are based on system telemetry, mission-specific assumptions, or high-level anomaly detection models (Saha et al., 2019). Our work differs by focusing on onboard CCSDS protocol-level rules that are directly derived from packet/frame structures and traffic behavior.

3.4. Telemetry-based anomaly detection using machine learning

A substantial body of work studies anomaly detection for spacecraft telemetry using machine learning, including RNN/LSTM-based approaches, explainable ML models, and survey papers summarizing telemetry anomaly detection methods and datasets (Hundman et al., 2018; Cuéllar et al., 2024; Katsube and Sahara, 2025; Chen et al., 2021). These approaches are useful for detecting complex failures and system-level anomalies, but may require extensive training data, feature engineering, and careful control of false positives (Nalepa et al., 2022; Li et al., 2019). In contrast, our CCSDS firewall rules are lightweight and protocol-driven: they target observable CCSDS violations, replay-like patterns, and encryption configuration risks without requiring labeled telemetry datasets. As a result, the proposed rules are suitable for deployments where the computational budget is limited or where operational explainability is required.

A related line of work focuses on monitoring and protecting the satellite communication link rather than inspecting protocol-level CCSDS traffic. Comprehensive surveys of satellite communication cybersecurity (Salim et al., 2024) describe threats and defenses across the space, ground, and communication-link segments, including signal interference, spoofing, and link-layer attacks. Such approaches are complementary to the proposed firewall: link-segment monitoring seeks to identify abnormalities in the communication channel, whereas the proposed CCSDS firewall begins at the post-demodulation channel-symbol interface on TC ingress and performs CCSDS-aware protocol inspection. Similarly, recent surveys on TinyML-based intrusion detection for CubeSats (Fayyaz et al., 2026) highlight the potential of lightweight onboard ML techniques for resource-constrained spacecraft. In contrast, our work performs rule-based CCSDS inspection without requiring trained models or telemetry datasets.

3.5. Motivating incidents and threat reports for satellite cybersecurity

Public reports have described interference or suspected unauthorized access attempts against satellite systems, including incidents involving Earth observation satellites and broader discussions of cyber threats to space systems (Boschetti et al., 2022; Manulis et al., 2021; Salim et al., 2024). These reports highlight that both ground links and operational workflows can be targeted and that early detection at the protocol and link levels can be valuable (Hamill-Stewart and Rashid, 2024). Our work is motivated by these operational and threat contexts and provides concrete CCSDS-focused audit rules for nonstandard framing and randomization, insecure or missing protection, and anomalous TC sequence behavior.

4. Design

4.1. Overall system design

This section presents the overall design of the proposed onboard CCSDS firewall, which operates as a lightweight proxy and inspection module embedded in the satellite OBC. The primary design objective is to validate telemetry (TM) and telecommand (TC) traffic at the CCSDS protocol level before such data are processed by mission-critical flight software (e.g., propulsion system, ADCS system) and mission payloads (e.g., Earth observation camera, communication module).

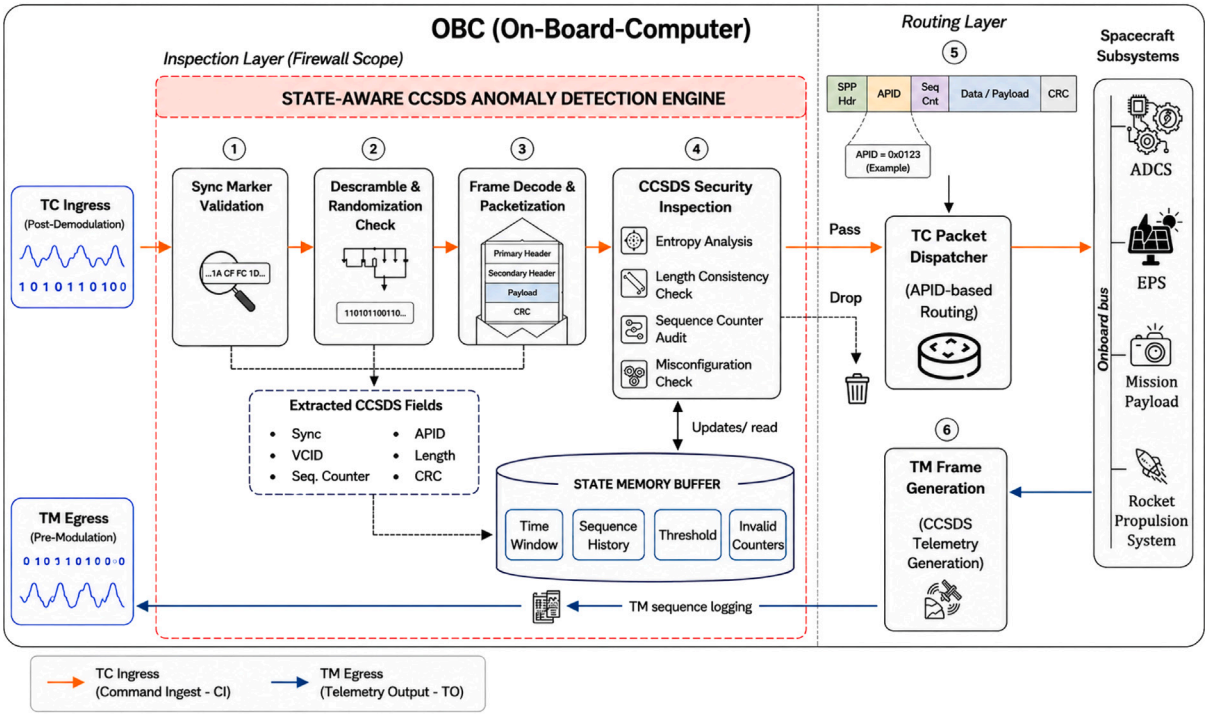


Fig. 2. CCSDS firewall architecture with separate telecommand (TC) uplink and telemetry (TM) downlink processing paths. The firewall performs protocol-aware inspection using observable CCSDS metadata and traffic behavior before forwarding validated TC packets to onboard subsystems and monitoring TM traffic for downlink transmission.

4.1.1. Design goals and constraints

The firewall is designed under the following constraints: (i) CCSDS compliance: The system must remain compatible with standard CCSDS TC/TM processing pipelines and must not require modifications to the high-level protocol. (ii) Onboard deployability: The firewall must operate within strict resource constraints in terms of CPU cycles, memory usage, and power consumption considering the space environment. (iii) Security Efficacy: The firewall should flag the rule-specific insecure conditions and anomalies associated with the predefined security-risk scenarios during normal satellite operation.

4.1.2. Architectural placement

Fig. 2 illustrates the logical placement of the firewall within the satellite communication stack. The firewall is deployed as an onboard software module and logically positioned between the RF demodulation layer and the TC/TM flight software dispatcher. The post-demodulation bitstream is made available to the firewall, which performs protocol parsing, conformance checks, and profile-guided security auditing before forwarding packets to payload or other satellite components.

In the simulation setup, the firewall is implemented as a NASA Core Flight System (cFS) application, allowing effective integration with various spaceflight software while preserving modularity. Because the cFS system provided by NASA does not implement CCSDS protocols, we modified the cFS system to support CCSDS and validated the entire system, encompassing both the ground station and the satellite, using the NASA Operational Simulator for Small Satellites (NOS3) (NASA, 2023). Based on the simulation, we also implemented and evaluated the firewall using two space-grade OBC platforms: the GomSpace NanoMind A3200 and the EnduroSat OBC. Note that we use the cFS architecture for the NOS3 simulation platform but not for the GomSpace and EnduroSat OBC platforms, as they provide their own proprietary SDKs.

4.2. Security risk scenarios

To demonstrate the efficacy of the CCSDS firewall, we developed a set of security-risk scenarios covering insecure configurations, non-standard customizations, missing protocol protections, and anomalous traffic. Five of these, labeled AS-1 to AS-5, are detectable by the onboard firewall and are the focus of this paper; they are summarized in Table 2, and each corresponds to a class of observable CCSDS protocol violations, cryptographic misuse, or anomalous traffic behavior. Each condition may be produced by a deliberate adversary or by a configuration or implementation mistake; the firewall detects and reports the condition itself in either case.

AS-1: Nonstandard Synchronization Markers. A CLTU or CADU may contain a synchronization marker that differs from the value specified by CCSDS. Such use may result from an incorrect communication configuration or an attempt to use a customized marker as security by obscurity. Because synchronization markers are exposed for frame acquisition and have small search spaces, customized marker values do not provide meaningful access control even when their values are not immediately observed (CCS, 2021b, 2025).

AS-2: Nonstandard Randomization Sequence. A TC unit is non-conforming when its transfer-frame bytes are randomized by XOR with a sequence other than the standard CCSDS randomization sequence. When the firewall applies the standard randomization sequence, such a mismatch generally produces header fields that are inconsistent with the configured profile. CCSDS randomization is a standardized public process for robust radio communication rather than encryption (CCS, 2021b).

AS-3: Improper Encryption Usage. Satellite communication employs encryption in an incorrect or insecure manner. This includes the use of inappropriate encryption modes such as ECB for data with visible patterns, insufficient encryption entropy, improper selection of encrypted payloads, or misconfigured cryptographic parameters. Such weak encryption configurations may arise from developer mistakes, misconfiguration, or supply-chain compromise, resulting in a satellite system that operates with vulnerable encryption settings.

AS-4: Lack of Encryption. Telecommand packets or sensitive satellite data are transmitted without encryption. This situation may occur due to non-compliance with security policies, misconfigured satellite operations, or intentional interference by an attacker. As a result, CCSDS packet elements that are expected to carry confidential

Table 2
Summary of Onboard CCSDS Security Risk Scenarios and Their Impacts.

Scenario	Observed abnormality	Impact
AS-1	Use of nonstandard CLTU start sequences or synchronization markers	Ineffective access control or communication misconfiguration
AS-2	Repeated post-derandomization header-validation failures	Communication disruption or configuration mismatch
AS-3	Encrypted traffic with visible patterns or low entropy	Leaks structural patterns in encrypted data
AS-4	Matching declared and actual payload lengths across all PDU layers	Command manipulation and data exposure
AS-5	Unexpected TC sequence-number repetitions, rollbacks, or jumps	Denial of service or unintended command execution

or critical information are exposed in plaintext, creating a high-risk operational condition.

AS-5: TC Sequence Anomalies. Received TC frames may show unexpected repetitions, rollbacks, or forward jumps in their sequence numbers. Possible causes include protocol-consistent retransmission, loss and recovery, reinitialization, a state mismatch, or deliberate replay. Because these causes can produce the same observable sequence behavior, the firewall reports a persistent sequence anomaly without attributing an individual frame to replay. Replay of downlink TM after transmission remains outside this onboard rule and must be assessed at the receiving ground segment.

4.3. CCSDS-based anomaly detection rules

This section describes the protocol-level audit rules implemented in the CCSDS firewall. Each rule is designed to flag the rule-specific insecure conditions and anomalies associated with the corresponding scenario (AS-1 to AS-5) based on CCSDS traffic characteristics. For general applicability, the firewall rules we propose rely on general CCSDS protocol specifications and a compact deployment profile, rather than payload-specific data or customized space applications. The profile specifies the applicable protocol path together with the managed coding and framing parameters used to determine which standardized CCSDS values apply and to locate the payload regions inspected by each rule.

4.3.1. AS-1: Detection of nonstandard synchronization markers

For AS-1, the firewall checks whether the synchronization marker associated with each received CLTU/CADU conforms to the standard value for the framing option specified by the deployment profile. A mismatch raises a configuration alert indicating the use of a nonstandard framing value. Such a mismatch may indicate either a communication misconfiguration or reliance on a customized marker as security by obscurity.

4.3.2. AS-2: Detection of nonstandard randomization sequence

For AS-2, the firewall checks whether received frames conform to the standard CCSDS randomization (scrambling) process. Repeated reception of frames that fail derandomization is interpreted as the use of a nonstandard randomization sequence.

To reduce performance overhead, our firewall does not fully derandomize every received frame. Instead, partial derandomization is applied to initial portions of the frame to quickly verify the randomization validity. If the number of invalid randomized frames exceeds a predefined threshold within a given time window, the traffic is classified as nonconforming to the standard CCSDS protocol.

4.3.3. AS-3: Detection of improper encryption usage

For AS-3, the firewall applies heuristic statistical checks to encrypted CCSDS packet portions. Specifically, based on the CCSDS packet analysis and the deployment profile, our firewall identifies the application-data region selected for inspection. CCSDS does not specify how internal application data should be encrypted. Therefore, developers may implement encryption incorrectly or choose insecure encryption methods. (e.g., just using XOR keys for encryption). To evaluate the safety of encrypted payloads, the firewall measures bit-level entropy.

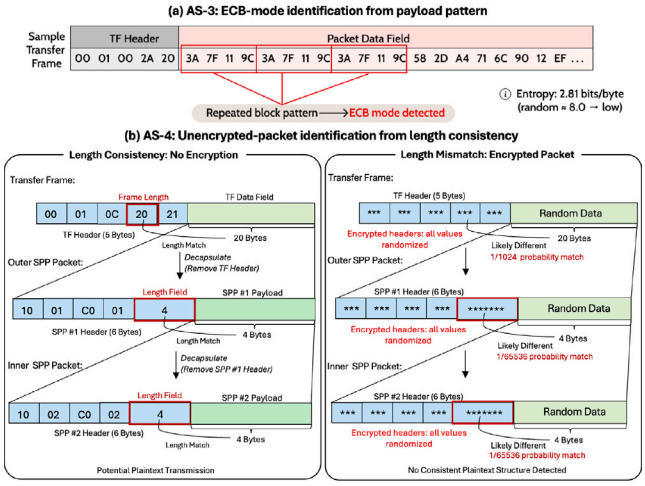


Fig. 3. Detection logic for AS-3 and AS-4. (a) AS-3 flags repeated-block patterns consistent with ECB-like behavior; (b) AS-4 flags a complete expected PDU hierarchy visible in the configured inspection region.

Although the firewall does not know the encryption algorithm and does not have the decryption key, such an assessment is still possible through entropy analysis. Low entropy values or visible data patterns may indicate insecure encryption status such as improper encryption modes, custom encryption algorithms, or weak configurations, etc. In our implementation, the Shannon entropy algorithm (Shannon, 1948) is used as a baseline to assess the security of encrypted data, and a lightweight approximation is applied to fit satellite hardware constraints. Encrypted packets that consistently show insufficient entropy are flagged as security risks. Fig. 3 illustrates the detection logic for AS-3 and AS-4. Part (a) shows how repeated ciphertext blocks can indicate improper ECB-mode encryption, while part (b) shows how a complete expected PDU hierarchy can indicate plaintext-like structure.

4.3.4. AS-4: Detection of plaintext-like structure

In AS-3, we mentioned that the firewall inspects the security of encrypted data. However, the CCSDS protocol does not provide any standard header information such as a flag bit to distinguish whether the internal (upper-layer) payload is encrypted. The applicable protocol hierarchy and inspection region are therefore supplied through the deployment profile.

One of the key design insights to address this challenge is to verify whether the payload lengths remain consistent throughout the configured protocol hierarchy. Although the CCSDS headers do not provide any encryption flag information, they provide the lengths of their embedded payloads. The firewall therefore parses the hierarchy recursively from the configured root PDU to the end of the inspection region. At each layer, it validates the header-declared length against the enclosing payload boundary and applies the same validation to the configured child PDU. For example, at the SPP layer, the Packet Data Length is used to determine the packet boundary before parsing continues with any configured child structure.

An encrypted header may coincidentally yield a plausible length at one layer. However, AS-4 requires all configured headers, lengths, and

boundaries to remain structurally consistent until the inspection region is consumed; a single coincidental match is therefore insufficient to trigger an alert.

Based on this principle, the detection rule for AS-4 focuses on identifying plaintext-like structure in the CCSDS layers. A frame whose configured PDU hierarchy is parsed consistently from the root PDU to the end of the inspection region is flagged as a potential plaintext transmission. Repeated occurrences across multiple frames further strengthen the evidence of sustained plaintext communication. A failed structural parse does not by itself indicate that encryption was applied. The same result can arise from a malformed or truncated PDU, an unsupported protocol format, or a mismatched deployment profile.

This heuristic rests on the observation that, in the CCSDS protocol hierarchy, the header of each layer declares the byte length of its payload field. When that layer is encrypted, the ciphertext replaces the original payload with a fixed-size block or stream that almost never matches the original plaintext length declared in the header. The check is therefore valid whenever the deployment profile correctly describes the expected layer hierarchy and the encryption scheme does not preserve payload length by design (e.g., stream ciphers in CTR mode without padding do preserve length and would evade this check). Missions in which the encrypted payload length coincidentally or by design equals the plaintext length should treat AS-4 as a configuration-dependent heuristic rather than a universal plaintext detector, and may combine it with AS-3 entropy checks for additional confidence.

4.3.5. AS-5: Detection of TC sequence anomalies

For AS-5, the firewall monitors TC Transfer Frame sequence numbers without inspecting the payload, since a captured frame can be retransmitted verbatim.

The rule checks for excessive repetitions, unexpected rollbacks, and jumps beyond the expected range. COP-1 defines the receiver-side sequence state used for reliable TC delivery and reports this state through Communications Link Control Words (CLCWs) (Consultative Committee for Space Data Systems (CCSDS), 2010). When observable in the Operational Control Field of outgoing TM frames, the CLCW Report Value and status flags can be used to track and resynchronize the expected sequence state, reducing false alerts caused by protocol-consistent retransmission, sequence-number wraparound, or reinitialization (CCS, 2021a,c). Without CLCW state, locally observed sequence history provides a heuristic baseline.

Because a legitimate retransmission and a verbatim replay may remain observationally identical, AS-5 reports persistent, unexplained sequence behavior as a replay-like anomaly rather than treating a single mismatch as conclusive evidence of replay.

Only a dedicated anti-replay service, such as the SDLS sequence-number mechanism, can reliably prevent replay (CCS, 2022). Where such a service is correctly enforced, replayed frames are already rejected at the link layer, and this rule is redundant with it. The firewall remains useful when anti-replay protection is absent or misconfigured, or when encryption is deployed without an accompanying anti-replay mechanism. In these cases, the firewall flags replay-like or otherwise abnormal sequence behavior as a complementary, defense-in-depth check.

Threshold and window selection. The detection parameters differ across rules and should be configured per mission as deployment inputs rather than fixed values. For the windowed rules, AS-2 and AS-5, `TIME_WINDOW` should be aligned with the expected ground-contact duration (typically 5–15 min for LEO passes) and `THRESHOLD` set to tolerate the worst-case benign anomaly rate on that link. For AS-2, sporadic demodulation errors at the grounded bit-error rate ($\lesssim 10^{-4}$) produce at most a handful of header-validation failures per window at real CubeSat uplink rates (1–38 kbps (GomSpace A/S, 2019)), based on calculation, a threshold of two to five errors in a single LEO pass (typically 10 min) provides a reasonable accuracy. For AS-5, the false-alarm analysis in Section 6.4 (Fig. 9) shows that a threshold of two already yields a false-alarm probability of 8×10^{-9} per window at realistic channel conditions, and the rate drops by roughly three orders of magnitude per unit increase in threshold, so the exact choice is not critical within this range. AS-1 requires no window or counter: it raises an alert on every frame whose synchronization marker does not match the standard value for the configured framing option, and no threshold parameter applies. AS-3 requires no observation window; its sensitivity is governed by the entropy threshold and block size supplied in the deployment profile, as discussed in Section 4.3.3. AS-4 requires neither a window nor a threshold: detection is determined entirely by whether the configured PDU hierarchy parses consistently to the end of the inspection region, as discussed in Section 4.3.4.

5. Implementation

This section describes how the proposed CCSDS firewall is implemented to inspect TC and TM packets based on the security-risk scenarios discussed in Section 4.2. The firewall is designed to analyze the structure and traffic behavior of the CCSDS protocol to detect abnormal conditions without relying on payload decryption or mission-specific payload semantics.

The implementation was developed and validated in three environments: a NASA cFS-based simulator, the EnduroSat onboard computer (OBC), and the GomSpace NanoMind A3200 OBC. Although the detection logic remains identical across all three environments, the deployment progresses from a simulator to flight-grade hardware with increasingly stringent resource constraints. The cFS simulator was used to validate basic detection behavior, followed by deployment on EnduroSat OBC as a representative satellite environment, and finally on GomSpace NanoMind A3200 as a highly constrained, mission-grade satellite platform. The implemented firewall serves as the basis for the evaluation presented in Section 6.

5.1. Satellite environment setup

The two flight-grade platforms selected for this study, the EnduroSat OBC and the GomSpace NanoMind A3200, were chosen because they are representative of resource-constrained CubeSat and small-satellite computing environments and have established precedent in the satellite security literature. ARM Cortex-M class onboard computers, the architecture family underlying the EnduroSat OBC, are the dominant computing platform for CubeSat missions and are explicitly identified as the representative resource-constrained target for onboard intrusion detection research (Fayyaz et al., 2026). The GomSpace NanoMind A3200, based on the AVR32 architecture, has also served as a representative platform in prior satellite security research: (Willbold et al., 2023) built an AVR32 emulator around the NanoMind A3200 architecture to conduct an experimental software security analysis of the ESA OPS-SAT spacecraft, demonstrating that this platform is both representative of operational small-satellite computing and a meaningful target for security evaluation. Together, these two platforms cover two distinct hardware architectures (ARM and AVR32) and operational design points commonly encountered in modern small-satellite systems, consistent with how prior work has scoped onboard security evaluation on CubeSat-class hardware. Unlike simulation-only environments, both platforms were obtained directly from their respective vendors and evaluated using their official software development kits (SDKs), allowing the proposed CCSDS firewall to be assessed under realistic deployment conditions.

5.1.1. NASA cFS simulator

NASA's Core Flight System (cFS) is a reusable flight software framework widely adopted in space missions. It provides a modular architecture for developing flight applications and supports both flight and simulation builds.

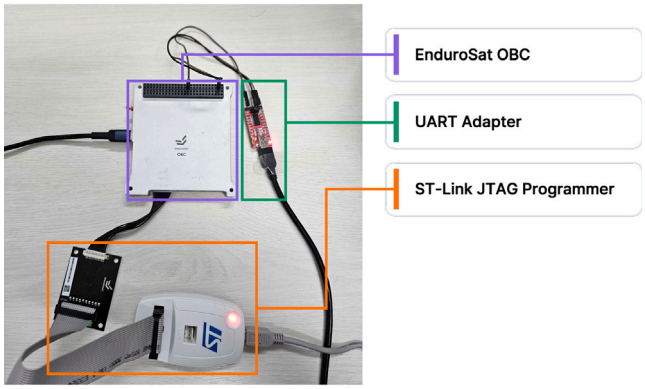


Fig. 4. EnduroSat onboard computer used for firewall implementation.

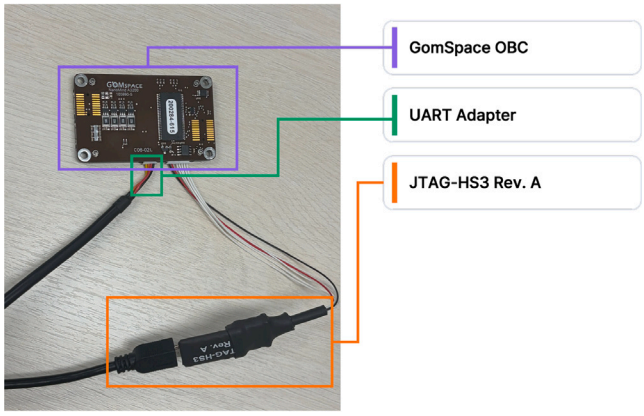


Fig. 5. GomSpace NanoMind A3200 onboard computer.

In this work, cFS was built in a simulation configuration targeting the x86_64 architecture. The CCSDS firewall logic was implemented and tested on an Ubuntu 24.04 system equipped with an AMD Ryzen 9 CPU (32 cores) and 64 GB of DDR5 memory. Since the simulator environment does not reflect realistic resource constraints, performance measurements were not a primary concern at this stage. Instead, the simulator was used to verify that the firewall logic correctly enforces the CCSDS protocol-analysis rules and responds appropriately to each scenario.

5.1.2. EnduroSat OBC

The EnduroSat OBC is a commercial off-the-shelf onboard computer widely used in low Earth orbit CubeSat missions. It is based on an ARM processor and provides sufficient computational capability for both mission logic and security monitoring tasks.

To develop satellite firmware for the EnduroSat OBC, the vendor-provided SDK is required. For this study, both the hardware and the SDK were purchased and used for implementation, with a combined cost of approximately USD 20,000. The firewall logic initially validated in the cFS simulator was extended and optimized for deployment in a real satellite environment. This involved adapting the logic to the embedded execution model and re-implementing components using only SDK-supported system services. The board provides sufficient computational resources to host the firewall logic alongside standard mission software.

At this stage, the CCSDS firewall was evaluated under realistic resource and timing constraints while maintaining functional equivalence with the simulator-based implementation. The EnduroSat OBC used in this study is shown in Fig. 4.

5.1.3. Gomspace NanoMind A3200 OBC

The GomSpace NanoMind A3200 OBC is a mission-grade onboard computer commonly used in operational satellite missions. Compared to the EnduroSat OBC, it is more tightly optimized for space deployment and is based on the AVR32 CPU architecture (GomSpace, 2026).

Similar to EnduroSat, firmware development for the NanoMind A3200 requires a dedicated SDK provided by the vendor. The combined cost of the hardware and SDK is approximately USD 30,000. Based on the EnduroSat implementation, the firewall logic was further optimized for low-power operation and strict embedded constraints. This included minimizing memory usage, reducing computational overhead, and selectively applying protocol checks based on the operational context. The GomSpace NanoMind A3200 OBC used in this study is shown in Fig. 5. Compared to the EnduroSat OBC, this platform imposes stricter memory and power constraints, making it the most challenging deployment environment evaluated in this study.

The GomSpace platform represents the most realistic deployment environment considered in this study and serves as the primary reference for performance evaluation.

The architecture-specific optimizations applied to the EnduroSat and GomSpace implementations, including fixed-point entropy arithmetic, static buffer allocation, and partial derandomization, affect only computational efficiency and not the underlying detection logic. Accordingly, the protocol-analysis rules, decision criteria, and alert conditions defined in Algorithms 1– 5 remain unchanged, thereby preserving the detection semantics of the reference cFS implementation.

5.2. Adding CCSDS firewall to OBC

This section describes how the CCSDS firewall implements detection logic for each scenario discussed in Section 4.2. Each rule is designed as a lightweight protocol-analysis algorithm suitable for embedded satellite environments. The implementation avoids payload decryption and relies only on observable CCSDS protocol fields and traffic behavior.

5.2.1. AS-1: Detection of nonstandard synchronization markers

For AS-1, the firewall compares the synchronization marker included in each inspected CLTU/CADU with the standard value for the framing option specified by the deployment profile. Any mismatch immediately raises an alert for a nonstandard communication configuration. Algorithm 1 presents the pseudocode for this detection logic.

Algorithm 1: Nonstandard Synchronization Marker Detection

```
for each received CLTU/CADU unit do
    if sync marker ≠ standard sync marker for configured framing
     option then
        raise alert
    end
end
```

It reports both communication misconfiguration and the use of a customized marker as security by obscurity. The implementation of this detection logic is provided in Listing 1 in Appendix.

5.2.2. AS-2: Detection of nonstandard randomization sequence

The AS-2 detection rule was implemented as a lightweight threshold-based routine that performs partial derandomization on the initial portion of each received CLTU and validates the header fields. The implementation maintains a counter of header-validation failures within a configurable time window and raises an anomaly alert once the predefined threshold is exceeded. The `TIME_WINDOW` and

THRESHOLD parameters are mission configurable and should be selected according to the expected traffic characteristics and channel conditions. Algorithm 2 presents the pseudocode for this detection logic.

Algorithm 2: Nonstandard Randomization Sequence Detection

```
Parameters: TIME_WINDOW, THRESHOLD
invalid_count ← 0
for each received CLTU unit do
    if TIME_WINDOW expired then
        invalid_count ← 0
    end
    if partially derandomized header fails validation then
        invalid_count++
    end
    if invalid_count > THRESHOLD then
        raise alert
    end
end
```

Partial derandomization is limited to the header bytes required for these checks, minimizing computational overhead while retaining the same header-validation logic.

5.2.3. AS-3: Improper encryption usage

For AS-3, the firewall analyzes the CCSDS packet structures and the deployment profile to identify the application-data region selected for inspection. The firewall applies a heuristic statistical check to this region by measuring bit-level entropy. Algorithm 3 presents the pseudocode for this detection logic.

Algorithm 3: Improper Encryption Usage

```
Parameters: BLOCK_SIZE, THRESHOLD
for each protected payload region do
    repeat_found ← false
    split protected payload region into blocks
    for each block i do
        for each block j after i do
            if block i equals block j then
                repeat_found ← true
            end
        end
    end
    entropy ← Shannon entropy of payload
    if repeat_found ∨ entropy < THRESHOLD then
        raise alert
    end
end
```

To accommodate the hardware constraints of the satellite, the firewall estimates the Shannon entropy (Shannon, 1948) of the protected payload using integer arithmetic: it builds a byte-value frequency histogram and obtains the required logarithms from a precomputed fixed-point lookup table rather than evaluating them in floating-point arithmetic. The entropy is reported in bits per byte, ranging from 0 for a constant byte stream to 8 for a uniformly random one; a payload whose estimated entropy falls below a threshold, or that contains repeated ciphertext blocks, triggers a statistical anomaly alert.

5.2.4. AS-4: Plaintext-like structure detection

The detection rule for AS-4 focuses on identifying payload regions that expose plaintext-like CCSDS structure. The firewall uses the deployment profile to select the expected protocol hierarchy and inspection region. Rather than relying on a single length equality, it recursively validates the configured PDU hierarchy by checking each header and declared length against its enclosing boundary. When packet blocking is permitted, the validation continues until the region is fully consumed.

A complete valid hierarchy causes a potential plaintext-transmission alert. Algorithm 4 presents the pseudocode for this detection logic.

Algorithm 4: Plaintext-Like Structure Detection

```
Function ValidateHierarchy(pdu):
    expected_len ← payload length declared in pdu header
    observed_len ← actual payload extent within the enclosing
      boundary
    if expected_len ≠ observed_len then
        return false
    end
    if pdu is the final configured layer then
        return true
    end
    if parsing the configured child-PDU layer fails then
        return false
    end
    for each parsed child_pdu do
        if not ValidateHierarchy(child_pdu) then
            return false
        end
    end
    return true
for each configured inspection region do
    root_pdu ← parse the configured root PDU
    if ValidateHierarchy(root_pdu) then
        raise alert
    end
end
```

A successful recursive validation flags the frame as a potential plaintext transmission. Repeated occurrences across multiple frames further strengthen the evidence of sustained plaintext communication.

5.2.5. AS-5: TC sequence anomaly detection

For AS-5, the firewall monitors sequence numbers in received TC Transfer Frame Headers without inspecting the payload. It counts unexpected repetitions, rollbacks, or jumps within an observation window configured according to the mission-specific principles. Algorithm 5 presents the pseudocode for this detection logic.

Algorithm 5: TC Sequence Anomaly Detection

```
Parameters: TIME_WINDOW, THRESHOLD
anomaly_count ← 0
for each received TC transfer frame do
    if TIME_WINDOW expired then
        anomaly_count ← 0
    end
    if sequence number repeats, rolls back, or jumps unexpectedly
      then
        anomaly_count++
    end
    if anomaly_count > THRESHOLD then
        raise alert
    end
end
```

As discussed in Section 4.3, observable CLCWs can resynchronize the detector and reduce false alerts caused by protocol-consistent retransmission or reinitialization; otherwise, the detector relies on locally observed sequence history. A threshold violation therefore indicates

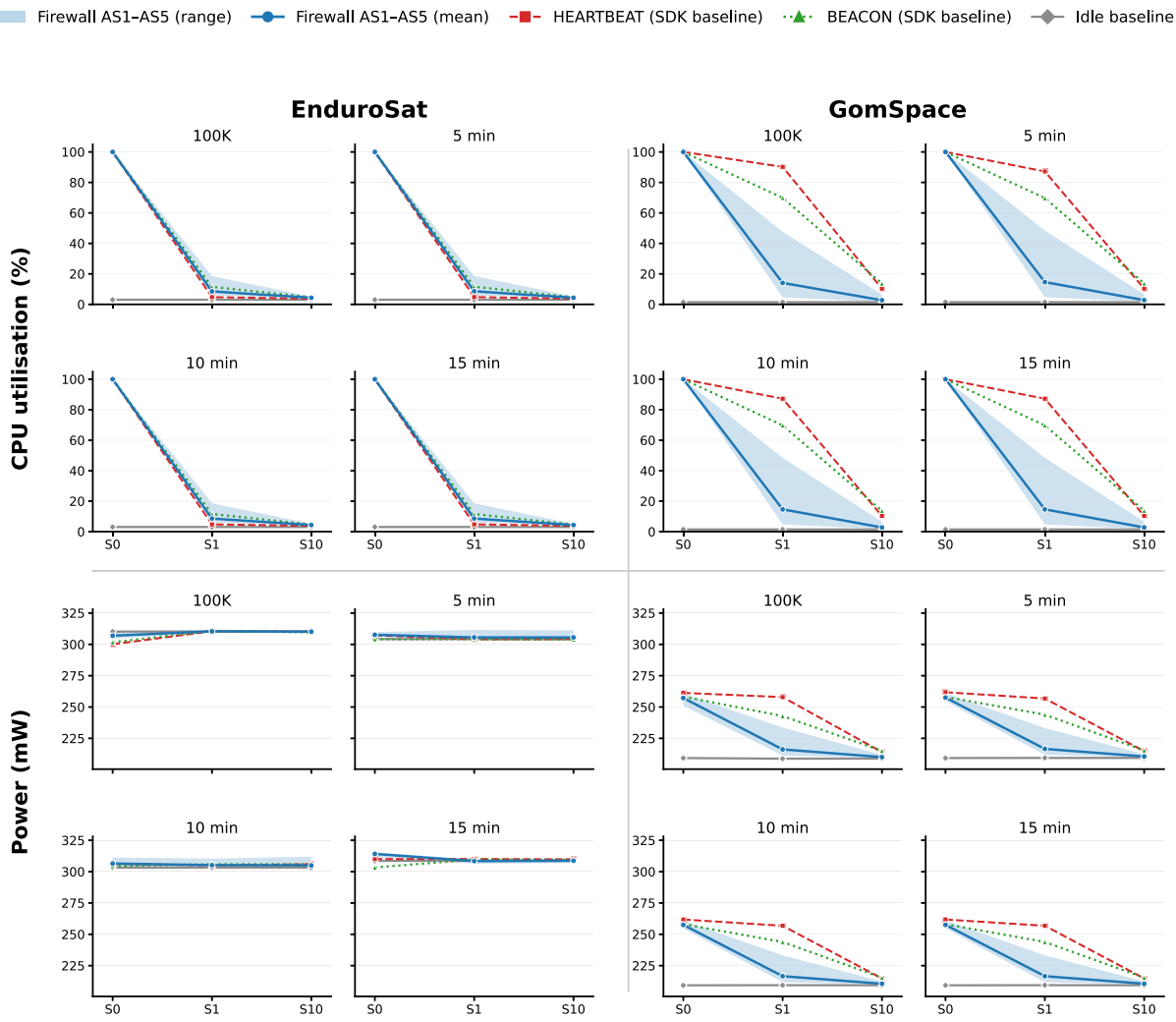


Fig. 6. CPU usage and power consumption for the firewall detectors (AS1–AS5) and the platforms' HEARTBEAT/BEACON reference workloads under the three controlled load profiles S0, S1, and S10. CPU usage is highest at S0 and decreases at S1 and S10 on both OBCs. The idle measurement (gray), shown only in the power row, reports each platform's board-level power while idle; it is omitted from the CPU row, where idle corresponds to near-zero usage.

persistent replay-like behavior rather than definitive attribution of an individual frame.

6. Evaluation

We evaluate the proposed CCSDS firewall in two parts. The first is a *performance evaluation* that measures what the firewall costs when it runs on real satellite hardware, and the second is a *security evaluation* that characterizes, rule by rule, the conditions under which the firewall can produce false positives or false negatives. The performance evaluation is carried out on three platforms: the NASA cFS simulator, the EnduroSat OBC, and the GomSpace NanoMind A3200 OBC. The security evaluation is presented in Section 6.4.

For the performance evaluation, we run the same TC/TM traffic that triggers each detection rule on all three platforms, so that the measurements are directly comparable. Each function is measured under three controlled load profiles—non-sleep (S0), a 1 ms inter-invocation delay (S1), and a 10 ms inter-invocation delay (S10)—and in two kinds of runs. The fixed-iteration run repeats a fixed amount of work (100 K operations) to compare per-operation cost, while the fixed-time run observes CPU and temperature over a sustained window; the latter is needed because thermal drift only becomes visible after the platform stays under load long enough. We set the fixed-time windows to 5, 10, and 15 min because a LEO satellite is only visible to a given ground station for a few minutes per pass before it drops below the horizon, so these durations bound how long a single uplink session—and therefore any sustained high-rate traffic from one station—can realistically last. S0 is a deliberately extreme worst case in which packets arrive back-to-back, while S1 and S10 provide reproducible load conditions based on fixed inter-invocation delays rather than representing any particular mission workload. Prior CubeSat software and OBC evaluations similarly report explicit task periods and distinguish deliberately high stress rates from expected operational rates (Bassani et al., 2020; Fuchs et al., 2019). At S10, the TC-side checks inspect approximately 47–71 kbps at their respective input layers, exceeding the published maximum data rates of representative CubeSat UHF transceivers (EnduroSat, 2022; GomSpace A/S, 2019) and providing a UHF-class reference for the applied load.

Besides the five firewall checks (AS-1 to AS-5), we also measure two SDK-provided functions, HEARTBEAT and BEACON, and use them as platform-native reference workloads. Periodic health monitoring, housekeeping, and beaconing are recurring flight-software services. Because they represent routine onboard activity, these measurements show how the firewall's absolute resource demand compares on the same OBC.

We use a Nordic Power Profiler Kit II (PPK2) (Nordic Semiconductor, 2026), a USB instrument that can supply power to a board while measuring the board's input current over a range from sub-microampere levels up to 1 A at a sampling rate of 100,000 samples per

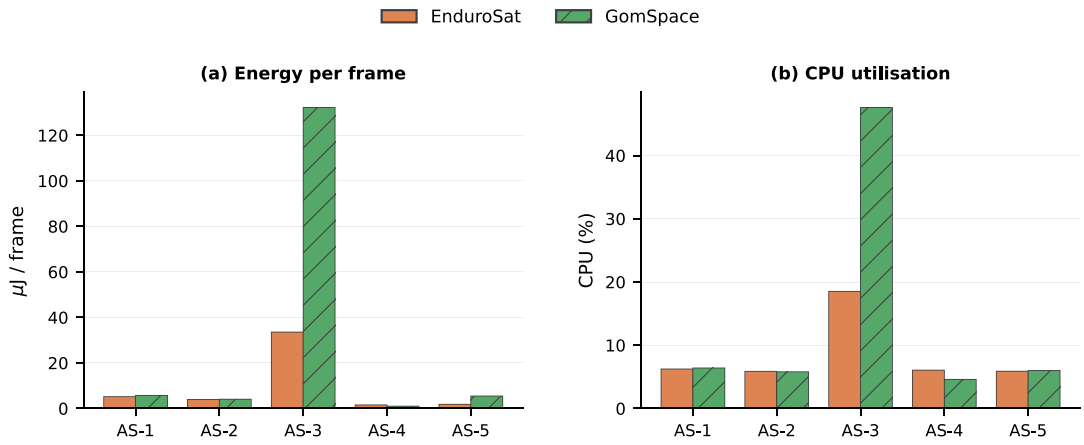
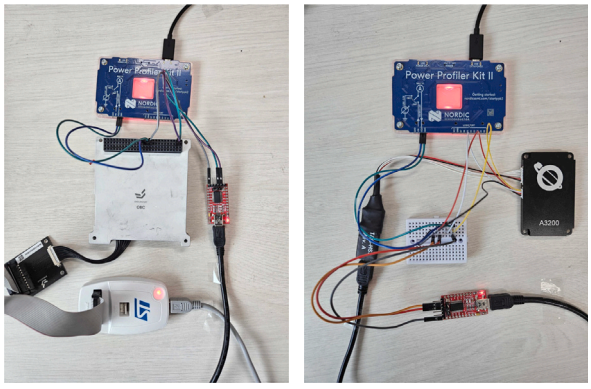


Fig. 7. Per-detector cost of the five firewall checks on the two flight OBCs, shown as (a) energy per inspected frame (μJ/frame) and (b) CPU usage. By both measures, AS-3 is markedly heavier than the other four checks because it inspects the full payload, while AS-1, AS-2, AS-4, and AS-5 stay low.



(a) EnduroSat OBC (b) GomSpace A3200 OBC

Fig. 8. Power-measurement setup: (a) EnduroSat and (b) GomSpace A3200 OBCs, each powered from a PPK2 over its 3.3 V rail in source mode while its current is measured.

second. The PPK2 offers two modes: a *source* mode, in which it powers the target and measures the current it delivers, and an *ampere* mode, in which an external supply powers the target and the PPK2 measures the current in series. We use the source mode, supplying the 3.3 V rail of each flight OBC directly from the PPK2 and recording each OBC's input current. Using these current measurements, we calculate the mean board-level input power as $\bar{P} = V\bar{I}$, with $V = 3.3\,\mathrm{V}$, and report it in mW. For the fixed-iteration runs, we calculate the board-level energy per inspected frame as $E_{\mathrm{frame}} = \bar{P}T/N$, where T is the run duration and $N = 100,000$, and report it in μJ/frame. The measurements are automated through the open-source `ppk2-api` Python library (IRNAS, 2024), and Fig. 8 shows the two flight boards wired to the PPK2 in this configuration. The same instrument has been used to measure power on other resource-constrained IoT and satellite-IoT platforms (Mäkitalo et al., 2025; Bocus et al., 2025; De Sombre et al., 2026).

Fig. 6 summarizes the performance results on the two flight OBCs across the three traffic patterns. At S0, where packets arrive back-to-back, every function—the firewall checks and the SDK reference workloads alike—keeps the CPU near 100%, which is the worst case. At S1, with short sleep intervals between packets, the firewall's CPU usage drops to the low single digits, close to or below the platform's own HEARTBEAT and BEACON functions. At S10, with longer sleep intervals, it falls further to roughly a couple of percent. Power likewise stays within the platform's normal range across all three patterns. The figures report CPU usage and power; more detailed per-function measurements, such as the temperature drift, are left to the appendix tables, where the drift stays small (about 1 °C or less). Taken together, the measured overhead suggests that the firewall can run alongside standard mission software. Fig. 7 breaks the cost down per check and shows that AS-3 is the only noticeably heavier rule, because it inspects the full payload rather than a few header fields. The complete per-platform numbers, including throughput and temperature for all sleep modes, are listed in Appendix A.3 (Tables 5–9).

Table 6 reports the measured S1 detector-level cost per invocation on the two flight OBCs. We use S1 as a reproducible intermediate reference that avoids the continuously saturated S0 loop while keeping each 100 K run substantially shorter than under S10. Processing latency is measured around each detector routine, excluding the 1-ms inter-invocation delay and UART output. Across the two flight OBCs, the S1 benchmark achieves 831–999 invocations/s, corresponding to an aggregate detector-input rate of approximately 0.39–0.95 Mbps depending on the detector-specific input size. Both boards show the same ordering: AS-3 is the heaviest, followed by AS-2, whose measured routine includes CLTU-to-TF extraction, while AS-1, AS-4, and AS-5 remain light. On EnduroSat, the three light checks are at or below 5.4 µs, compared with 8.2 µs for AS-2 and 139 µs for AS-3. On GomSpace, the corresponding mean latencies are about 13, 84, and 487 µs. The firewall performs no dynamic heap allocation, and the measured peak stack usage remains within 2 KiB on both boards. According to the linker maps, the compiled module containing the firewall occupies 12.5 KiB of flash and 604 B of static RAM on EnduroSat, and 12.3 KiB and 616 B, respectively, on GomSpace.

For the security evaluation, we characterize each detection rule according to its observable and alert logic, as detailed in Section 6.4.

6.1. NASA simulator based evaluation

The cFS evaluation runs on the x86_64 simulator described in Section 5, on a general-purpose server (a 32-core AMD Ryzen 9). Because there is no flight board at this stage, the purpose of this evaluation is only to confirm that the five detection routines work as intended, not to measure realistic onboard cost. The runs serve as a proof of concept: each rule processes its TC/TM traffic correctly, and at S0 all checks complete quickly, with the entropy-based AS-3 taking longer than the header-only checks but still finishing in a fraction of a second per 100 K operations (Table 5). Under S1 and S10, the inserted sleep intervals, not the checks themselves, dominate the measured CPU (about 11%–18%), as expected on a fast desktop-class CPU. These numbers establish a baseline for the relative cost of each rule, but because they come from server-class hardware they do not reflect the constraints of a real OBC; the flight-grade results in the next two subsections do.

6.2. EnduroSat based evaluation

The EnduroSat evaluation runs on a commercial ARM-based OBC representative of CubeSat-class missions, with the firewall deployed through the EnduroSat SDK. The left column of Fig. 6 shows the result. At S0, where packets arrive back-to-back, every function saturates the CPU, which is expected for a tight processing loop. Once packets are interleaved with idle intervals at S1 and S10, CPU use drops to the low single digits, and the firewall tracks the platform's own HEARTBEAT and BEACON functions closely: the difference between running the firewall and running the board's routine functions is negligible. Among the checks, only AS-3 stands out as slightly heavier (about 18% CPU at S1 versus roughly 5% for the others; Fig. 7 b) because it inspects the full payload. The fixed-iteration and fixed-time runs give consistent numbers, and the temperature drift, reported in detail in the appendix tables, stays small (typically around 1 °C or less).

The active-power curves in Fig. 6 are almost flat: power stays around 305–310 mW for every function and every sleep mode, and, more notably, it does not drop even when the board is largely idle, so the firewall adds no measurable power over the idle measurement. This is a property of the EnduroSat firmware rather than a sign that the firewall is free. By inspecting the EnduroSat SDK source code, we confirmed that its default idle scheduling is implemented as a busy spin, so the CPU keeps running even when it has no work to do, and the board therefore draws roughly the same power whether it idles or runs the firewall. On this platform, power alone cannot separate the firewall from idle; what it does show is that adding the firewall does not raise the board's power envelope. Overall, these results suggest that the firewall can operate within the resource budget of a CubeSat-class ARM platform.

6.3. GomSpace based evaluation

The GomSpace NanoMind A3200 is the most constrained platform in our study, a mission-grade OBC with tight power and memory limits; the firewall was optimized for this board and deployed through the GomSpace SDK (Section 8), and the runtime sampler used to collect these metrics is shown in Listing 2. The right column of Fig. 6 differs from EnduroSat in one important way. Here, the idle measurement sits low and clearly apart from the active curves: idle power is about 209 mW, while running the firewall at S0 draws about 258 mW, and adding sleep intervals brings it back down (about 212 mW at S1 and 209 mW at S10). In other words, on GomSpace every computation—the firewall and the SDK functions alike—uses visibly more power and CPU than idle. The reason is the opposite of EnduroSat's: the GomSpace SDK puts the CPU into a genuine low-power idle state when there is no work, so the hardware actually stops between packets, and the cost of computation shows up clearly above the idle floor.

For CPU, most firewall checks fall to about 5%–7% at S1 and to a few percent at S10, staying well below the platform's own HEARTBEAT and BEACON, which remain high at S1 (HEARTBEAT around 90% and BEACON around 70%). The SDK functions stay high because their paths keep doing real background work (CSP and housekeeping) that the inserted idle intervals do not suppress, whereas the firewall genuinely yields the CPU. AS-3 is the heaviest firewall check—about 132 µJ per frame at S0 and 47% CPU at S1 (Fig. 7; every check saturates the CPU at S0, so CPU is compared at S1 and per-frame energy at S0)—because it runs full-payload entropy and repeated-block inspection; even so it stays below the SDK reference workloads and drops to about 6% at S10. We reduce this cost with the integer and fixed-point optimization described in Section 8.

Across the GomSpace runs the firewall checks stay in the single digits at S1 and around 2% at S10 (AS-3 excepted, as noted above), while the temperature drift, reported in detail in the appendix tables, remains small (about 1 °C or less). Because this is the most constrained platform we evaluate, these results provide our primary evidence that the firewall can operate within the resource budget of a mission-grade OBC.

6.4. Security efficacy

The five rule conditions defined in Section 4.2 each violate a security practice recommended by the CCSDS standards, and depending on the mission such a violation can lead to a serious security problem. This subsection therefore examines, rule by rule, the situations in which the firewall's detection itself can fail—that is, when a false positive (an alert without an actual violation) or a false negative (a missed violation) can occur—and quantifies the conditions under which neither error arises. The rules are evaluated exactly as deployed, over a wide range of channel and traffic conditions, including deliberately extreme ones. Throughout, the firewall reports the observed condition without inferring its cause or requiring payload decryption. Table 3 summarizes the outcome of this analysis.

AS-1 concerns the use of a nonstandard synchronization marker. The rule is implemented as a deterministic comparison on the demodulated bitstream: the marker of each received CLTU/CADU is compared with the standard value selected by the deployment profile. Because the outcome is fully determined by the inspected bytes, the rule produces neither false positives nor false negatives as long as demodulation itself completes correctly, and we verified this behavior on all three platforms.

AS-2 checks whether the standard CCSDS randomization has been applied. The key idea of the implementation is that the firewall does not derandomize the whole packet: it derandomizes only the header bytes needed for validation, which keeps the per-frame cost low. Its verdict can be wrong in only two situations: when demodulation errors corrupt the inspected header bytes, and when the packet-processing queue overflows so that frames are dropped before they are inspected. When demodulation completes without errors and no frame is dropped, the false-positive and false-negative rates are zero, as for AS-1.

To determine when frames can be dropped, we measured the packet-processing path of our EnduroSat deployment, where 64 KB of memory is available at run time for buffering packets awaiting inspection, and the sample CCSDS packets used in the experiments average 120 B, so the queue holds about 546 packets. The measured sustained processing rate of this path is 0.9 Mbps, or about 937 packets per second. As long as the incoming TM/TC rate stays at or below this rate, the queue never fills, no frame is dropped, and no false positive or false negative occurs. Input bursts above 0.9 Mbps are absorbed by the queue for a bounded time before frames are lost; the measured tolerance is listed in Table 3, ranging from about 5.2 s for a 1.0 Mbps burst to about 65 ms for a 9.0 Mbps burst.

AS-3 is evaluated as a heuristic statistical test. Its repeated-block check can indicate a weak encryption mode, while its entropy test is only an estimate. The entropy test can err in both directions: a very short encrypted frame may look low-entropy and produce a false positive, while high-entropy plaintext such as a compressed payload may pass unflagged as a false negative. These are well-known limits of entropy-based checks, so we state them as conditions and quantify them with a simulation below.

AS-4 is evaluated as a profile-guided structural heuristic rather than an exact classifier of encrypted and plaintext traffic. Encrypted bytes that coincidentally satisfy the configured header, length, and boundary checks can cause false positives, whose likelihood depends on the configured protocol hierarchy. Its false negatives are plaintext frames that do not expose the complete configured hierarchy: traffic that is malformed, truncated, or inconsistent with the deployment profile may fail to parse, so the absence of an alert does not establish that encryption was applied.

To make these conditions concrete, we simulated both rules exactly as deployed; Table 4 shows the results. For AS-3, we applied the entropy and repeated-block checks to 100 K payloads per length, using random bytes as encrypted payloads and a compressed telemetry-like corpus as plaintext, over the payload sizes used in our experiments (Table 6). The numbers confirm the two limits stated above

Table 3
When each rule can raise a false alarm (false positive) or miss a violation (false negative), and the condition under which neither happens. The bottom rows show how long the EnduroSat packet queue can absorb an input burst faster than its 0.9 Mbps processing rate before frames are dropped.

Rule	Alert logic	False positive	False negative	Condition for no errors
AS-1	Per-unit check	None	None	Correct demodulation
AS-2	Windowed count	None	None	Error-free demodulation; no queue overflow (input rate ≤ 0.9 Mbps)
AS-3	Per-frame heuristic	Very short encrypted frame may look low-entropy	Compressed (high-entropy) plaintext may pass	None; inherent limit of entropy estimation
AS-4	Per-frame structure check	Encrypted bytes may parse as a valid hierarchy by chance	Plaintext that does not parse as the full hierarchy	Depends on the configured PDU hierarchy
AS-5	Windowed count	Heavy frame loss makes retransmissions look like replay	Replay kept below the alarm threshold	Normal channel error rates
EnduroSat queue burst tolerance (time before frames are dropped when the input rate exceeds 0.9 Mbps) 1.0 Mbps: ≈5.2 s 1.8 Mbps: ≈0.58 s 9.0 Mbps: ≈65 ms				

Table 4
Simulated false-positive (FP) and false-negative (FN) rates for AS-3 and AS-4, with 100 K frames per condition. AS-3 uses a 5.0 bits/byte entropy threshold with 16 B blocks; its FP is measured on encrypted (random) payloads and its FN on compressed plaintext. AS-4 parses random payloads against the configured Transfer Frame and Space Packet hierarchy.

AS-3 (entropy + repeated blocks)		
Payload length	FP (encrypted)	FN (compressed)
16 B	100%	0%
32 B	86.9%	12.4%
64 B	0%	99.9%
128 B	0%	100%
AS-4 (recursive length consistency)		
Configured hierarchy	FP (encrypted payloads)	
1 layer (Transfer Frame)	9.8×10^{-4} ($\approx 2^{-10}$)	
2 layers (+ Space Packet)	none in 10^6 ($\approx 2^{-26}$)	

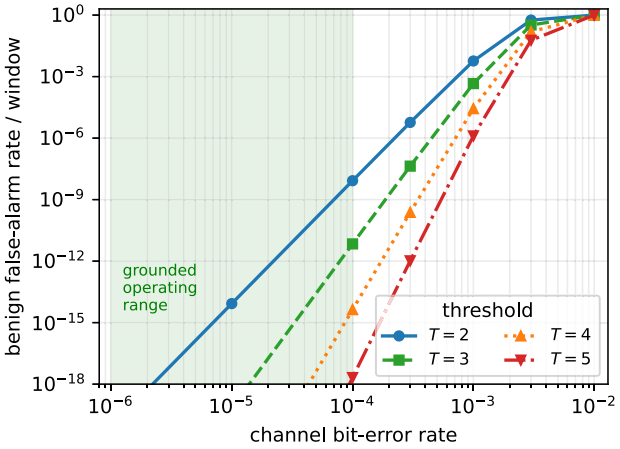


Fig. 9. AS-5 false-alarm rate versus channel bit-error rate at a grounded 10 kbps uplink (64-byte commands, ≈19 frames per one-second window) for several alarm thresholds T, staying far below the benign noise floor (8×10^{-9} at $T = 2$) until the error rate passes 10^{-3}.

and show where each one applies: below about 32 B the entropy estimate collapses, so even properly encrypted payloads are flagged, while from 64 B upward compressed plaintext becomes statistically indistinguishable from ciphertext and almost always passes. Structured, uncompressed plaintext was detected in every case. For AS-4, a single-layer check on random payloads produces a coincidental match at the expected rate of about 2^{-10}, but adding the second configured layer left no false positive in 10^6 frames (about 2^{-26} analytically): each additional layer of the recursive check sharpens it by orders of magnitude. Well-formed plaintext hierarchies were detected in every case, and truncating them prevented the parse, which matches the false-negative condition stated above.

For AS-5, the main source of ambiguity is that frame loss, recovery, and protocol-consistent retransmissions can produce the same sequence transitions as deliberate replay. A single unexpected transition therefore cannot be assigned a cause, and AS-5 reports only when the anomaly count exceeds a threshold over an observation window. To determine the channel conditions under which this false positive does not occur, we compute the benign false-alarm rate as a function of the channel bit-error rate, approximating benign anomalies from frame loss and retransmission using the frame-error rate, and sweep the alarm threshold T—the number of unexpected sequence transitions in a window that triggers an alarm; Fig. 9 summarizes the result. For a 10 kbps uplink at a realistic bit-error rate (10^{-4} or below), the false-alarm probability is about 8×10^{-9} per window at a threshold of two, and it drops by roughly three orders of magnitude for each additional unit of T, staying negligible until the error rate passes 10^{-3}; the exact choice of T is therefore not critical. The false-negative boundary of AS-5 is shared by any windowed rule: sequence anomalies held below the configured threshold do not raise an alert. It remains a low-cost onboard audit that reports when unexpected TC sequence transitions exceed that threshold, complementing COP-1 sequence handling and SDLS anti-replay enforcement (Consultative Committee for Space Data Systems (CCSDS), 2010; CCS, 2022).

Table 3 consolidates these results: for each rule, when a false positive or a false negative can occur, and the operating criteria under which neither does, together with the measured burst tolerance of the EnduroSat packet queue. Detection latency is reported in Table 6: for the per-unit rules (AS-1, AS-3, AS-4) the per-invocation processing latency is the detection latency, since the alert is raised on the offending unit itself, whereas for the windowed rules (AS-2, AS-5) the alert latency is additionally bounded by the configured observation window. The false-alarm curves in Fig. 9 are computed analytically from the frame-error rate, so confidence intervals do not apply to them; the measured quantities in Table 6 carry their run-to-run standard deviations. Overall, the results characterize the operating boundary of each rule rather than supporting a general attack-detection claim. The firewall adds an audit layer on top of a mission's cryptographic and sequence-control protections rather than replacing them.

7. Limitation and future work

This study has several limitations that suggest directions for future work. First, the proposed firewall focuses on protocol-level analysis by detecting observable CCSDS structural violations and traffic patterns. This is because command-level validation requires the mission-specific command dictionary and operational policies owned by the spacecraft and ground-station designers, whereas a reusable firewall core operates

without mission-specific command semantics. Its protocol parameters and inspection boundaries are supplied through the deployment profile. Extending the framework to incorporate semantic payload inspection is a natural direction for future work, provided that the firewall is authorized to access the decryption keys for protected payloads.

Second, transmitter attribution for otherwise valid TC frames requires ground-segment evidence and therefore remains outside the current onboard implementation and evaluation. Nevertheless, the proposed architecture provides a practical foundation for future integration with ground-segment monitoring while preserving the core protocol-analysis rules, as discussed in Section 8.

Third, the evaluation primarily considers controlled and repeatable traffic patterns so that comparisons between platforms remain consistent and interpretable. This baseline can be extended to capture a wider diversity of real-world RF interference, noise conditions, and adversarial strategies that may arise during long-term satellite operations.

Future work will also consider adaptive thresholding, where detection parameters are dynamically adjusted based on mission phase, traffic patterns, or environmental conditions.

8. Discussion

The results of this study demonstrate that effective security monitoring for satellite systems can be achieved using CCSDS protocol analysis without relying on payload decryption or mission-specific command semantics. By focusing on observable protocol violations, cryptographic misconfigurations, and traffic behavior, the proposed firewall provides a lightweight and deployable security mechanism suitable for resource-constrained satellite systems.

A key observation is that many security-relevant conditions manifest themselves as deviations from the expected CCSDS behavior. Examples include nonstandard synchronization markers, abnormal randomization patterns, replay-like sequence number behavior, and insufficient entropy in encrypted payload regions. These characteristics can be detected using simple rules and thresholds, making them well-suited for embedded implementation.

The staged implementation in a simulator, EnduroSat OBC, and GomSpace NanoMind A3200 OBC highlights the practicality of the approach. Although performance overhead increases under sustained high-rate traffic, the firewall remains operational in all tested environments. This suggests that protocol-level security auditing can be realistically deployed on modern CubeSat and mission-grade OBCs.

More broadly, the proposed CCSDS firewall does not replace cryptographic protections such as SDLS but complements them as an independent audit layer, adding value precisely where link-layer security is misconfigured or temporarily disabled during operations. For example, a controller who disables encryption on a module to run a quick test and forgets to re-enable it leaves that module silently transmitting in the clear; SDLS raises no alarm, since Clear Mode is a permitted configuration, whereas AS-4 raises a plaintext-structure alert when the inner CCSDS PDU hierarchy remains structurally visible. Likewise, a controller who keeps encryption enabled but, lacking cryptographic background, selects a non-recommended mode such as ECB produces nominally protected traffic that SDLS does not reject—it specifies approved AES-based cipher suites but does not enforce their correct use—whereas the firewall catches the resulting repeated ciphertext blocks and low-entropy payloads (AS-3). In both, a routine operator convenience opens a silent protection gap that the firewall detects on top of SDLS rather than in place of it.

A related scope boundary concerns unauthorized uplinks that produce otherwise valid TC frames. If an attacker obtains all the information required to generate a legitimate telecommand (TC) uplink, the satellite cannot easily distinguish the malicious packet from an authorized one using protocol information alone. However, because satellite RF signals cover a wide area, such an attack may be detectable

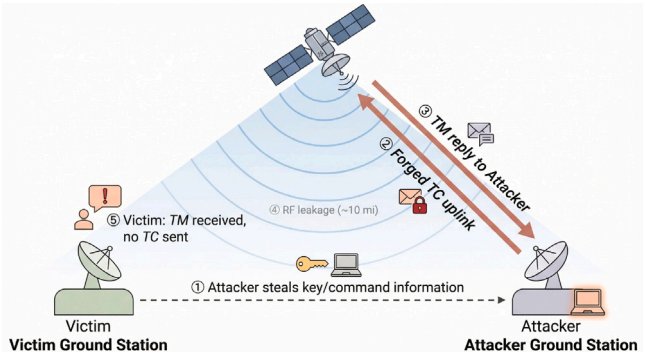


Fig. 10. Ground-side detection idea for the unauthorized C2 uplink by correlating legitimate telecommand transmissions with telemetry responses.

when the attacker and the legitimate ground station are located in proximity, for example, within the same RF coverage footprint of a LEO pass. As illustrated in Fig. 10, the detection approach is based on correlating every TC transmission with its subsequent TM responses. If telemetry responses are observed without any corresponding TC transmission recorded by the authorized ground station, this pattern may indicate an unauthorized entity sending legitimate TC packets to the satellite within the same RF coverage area.

Transmitter attribution of this kind requires evidence from outside the spacecraft, particularly transmission records maintained by authorized ground stations. Such a pattern would not by itself prove an unauthorized uplink, because logging loss, delayed telemetry, delayed execution of previously uploaded time-tagged commands, autonomous spacecraft behavior, or multi-station operation may produce the same observation. Because this correlation cannot be reliably observed onboard, detection must be performed at the ground station. This cross-segment mechanism was not implemented or evaluated in this study.

The performance results also indicate that the runtime cost of the firewall is not uniform across rules. In particular, AS-3 introduces the largest overhead because it applies full-payload statistical inspection, combining repeated-block detection with entropy estimation, whereas most other rules mainly examine fixed-size headers or short protocol fields. During evaluation on the GomSpace platform, this routine accounted for the highest CPU usage among the other checks, which appears to be mainly caused by unnecessary stack allocation and array initialization each time the entropy routine runs. To reduce this cost while preserving the same detection semantics, we replaced the per-symbol floating-point logarithm with a fixed-point integer lookup table and reused a statically allocated frequency buffer, clearing only the entries touched by each payload. This removes the dominant per-packet floating-point and allocation overhead noted above.

9. Conclusion

As satellite missions move toward standardized protocols and software-defined operations, protocol compliance alone no longer guarantees security at the onboard ingress boundary. In this paper, we defined a set of realistic, CCSDS-based security-risk scenarios—insecure configurations, nonstandard customizations, missing protocol protections, and anomalous traffic—and showed that even protocol-compliant traffic can carry unsafe or misconfigured commands when that boundary is not inspected. We then designed and implemented a CCSDS-compliant onboard firewall that automatically detects and reports these conditions, adding a layer of inspection on top of existing link-layer protections such as SDLS rather than replacing them.

We evaluated the firewall on a NASA cFS simulator and on two flight-grade OBCs, the EnduroSat board and the GomSpace NanoMind A3200. When packet processing was evaluated under the S10 controlled load profile, the firewall's CPU usage dropped into the single

digits—roughly 1.5 to 4% for the lighter checks and about 5 to 6% for the heaviest check (AS-3). The temperature drift stayed small, on the order of 1 °C across all runs. These results suggest that the firewall can audit onboard traffic within the resource budgets of CubeSat-class and mission-grade OBCs. Future work includes automating rule generation for different mission profiles and exploring hardware acceleration to further lower the cost of onboard inspection.

CRediT authorship contribution statement

Woohyeop Im: Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Conceptualization. **Gyeongje Oh:** Software, Investigation, Formal analysis. **Shinichi Kan:** Resources, Formal analysis. **Kosuke Ito:** Validation. **Hikohiro Lin:** Formal analysis. **Sayeon Kim:** Validation. **Jinwoo Jeong:** Data curation. **Sangbom Yun:** Resources. **Daehee Jang:** Writing – review & editing, Supervision, Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the National Research Foundation (NRF) grant (No. RS-2026-25489421) and the Convergence Security Core Talent Training Business Support Program (IITP-2026-RS-2023-00266615) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation), and by the Korea Research Institute for Defense Technology Planning and Advancement (KRIT) grant funded by the Korean government (Defense Acquisition Program Administration) (KRIT-CT-24-001, Defense Space Security Research Lab, 2025).

Appendix. Implementation code listings

This appendix provides representative code listings for the CCSDS firewall implementation. Complete source code is available at the project repository.[1]

A.1. CLTU start sequence validation (AS-1)

```
int32_t Check_Packet_CLTU(const uint8_t
    *data_cltu, uint16_t cltu_len)
{
    /* Basic input validation */
    if (data_cltu == NULL) {
        return CCSDS_CHECK_ERR_NULL_POINTER;
    }

    if (cltu_len < 2) {
        return CCSDS_CHECK_ERR_INVALID_LENGTH;
    }

    /* Verify CLTU start sequence */
    if (data_cltu[0] != CLTU_START_SEQ_0 ||
        data_cltu[1] != CLTU_START_SEQ_1) {
        return
            CCSDS_CHECK_ERR_INVALID_CLTU_START;
    }

    return CCSDS_CHECK_OK;
}
```

Listing 1 CLTU start sequence validation logic.

A.2. GomSpace performance sampler

```
static void gs_perf_sampler_take_sample(void)
{
    /* Read cumulative runtime counters for the
        monitored task */
    gs_perf_read_runtime(
        sampler.target_task,
        &total_rt,
        &task_rt);

    /* Runtime counters are cumulative, so take
        the delta for this sample */
    total_d = total_rt - sampler.prev_total_rt;
    task_d = task_rt - sampler.prev_task_rt;

    /* Save the current counters as the next
        sampling baseline */
    sampler.prev_total_rt = total_rt;
    sampler.prev_task_rt = task_rt;

    /* Store CPU in permille so reports keep
        one decimal place */
    if (total_d > 0U) {
        cpu = (uint16_t)(((uint64_t)task_d *
            1000ULL) /
                         (uint64_t)total_d);
        if (cpu > 1000U) {
            /* Clamp samples so they never
                exceed 100.0            cpu =
                1000U;
        }
    }

    /* Accumulate per-sample values for
        avg/min/max reporting */
    sampler.cpu_sum += cpu;
    sampler.cpu_samples++;
    sampler.samples++;
    if ((sampler.cpu_samples == 1U) || (cpu <
        sampler.cpu_min)) {
        sampler.cpu_min = cpu;
    }
    if ((sampler.cpu_samples == 1U) || (cpu >
        sampler.cpu_max)) {
        sampler.cpu_max = cpu;
    }

    /* Record when the most recent sample was
        collected */
    sampler.end_ts_ms = gs_perf_get_time_ms();
}
```

Listing 2 GomSpace performance sampler: taking one runtime sample.

A.3. Detailed per-platform measurements

This appendix reports the full per-platform measurements summarized by Fig. 6. Table 5 gives the fixed-iteration (100 K) cross-platform comparison; Tables 7–9 give the per-platform fixed-time runs (5/10/15 min; iteration counts in thousands, k).

[1] https://github.com/4g6dcbd21B/CCSDS-based-Firewall

Table 5
Cross-platform fixed-iteration (100 K): wall-clock duration (s), CPU, ΔT, and power consumption (EnduroSat and GomSpace measured at 3.3 V).

Platform	Function	Non-Sleep (S0)				Sleep 1 ms (S1)				Sleep 10 ms (S10)			
		Dur	CPU (%)	TΔ	P (mW)	Dur	CPU (%)	TΔ	P (mW)	Dur	CPU (%)	TΔ	P (mW)
NASA cFS	AS-1	4 ms	100.0	–	–	113.9 s	14.6	–	–	1,013 s	11.1	–	–
	AS-2	2 ms	100.0	–	–	112.4 s	16.9	–	–	1,013 s	11.1	–	–
	AS-3	125 ms	100.0	–	–	112.8 s	17.2	–	–	1,014 s	11.2	–	–
	AS-4	3 ms	100.0	–	–	112.8 s	14.4	–	–	1,013 s	10.6	–	–
	AS-5	2 ms	100.0	–	–	113.6 s	17.1	–	–	1,014 s	11.2	–	–
	HEARTBEAT	3 ms	100.0	–	–	112.5 s	17.0	–	–	1,014 s	11.2	–	–
	BEACON	727 ms	100.0	–	–	116.0 s	31.4	–	–	1,019 s	13.3	–	–
EnduroSat	AS-1	1.7 s	100.0	0.31	304.9	100.7 s	6.2	0.43	310.2	1,000 s	3.9	0.02	310.0
	AS-2	1.3 s	100.0	0.55	304.6	100.6 s	5.9	0.12	310.2	1,000 s	3.9	0.28	309.6
	AS-3	10.8 s	100.0	0.39	310.7	100.6 s	18.5	0.23	311.4	1,000 s	5.1	0.47	309.7
	AS-4	0.5 s	100.0	0.64	309.8	101.6 s	6.1	0.84	310.6	1,003 s	4.8	0.00	310.4
	AS-5	0.6 s	100.0	0.00	304.5	100.7 s	5.9	0.44	309.5	1,000 s	3.9	1.07	311.0
	HEARTBEAT	0.2 s	100.0	0.12	300.1	100.7 s	4.7	0.22	310.5	1,000 s	3.8	0.10	310.2
	BEACON	6.9 s	100.0	0.18	301.7	101.0 s	11.5	0.18	310.8	1,000 s	4.7	1.32	309.5
GomSpace	AS-1	2.2 s	100.0	0.00	260.8	100.2 s	6.4	0.10	212.4	1,000 s	1.8	0.00	209.3
	AS-2	1.6 s	100.0	0.00	254.1	100.2 s	5.8	0.00	211.6	1,000 s	1.8	0.10	209.6
	AS-3	50.9 s	100.0	0.20	259.9	111.8 s	47.6	0.10	233.8	1,012 s	6.4	0.20	212.4
	AS-4	1.7 s	100.0	0.00	251.6	100.3 s	7.1	0.00	213.4	1,000 s	1.9	0.40	210.8
	AS-5	2.1 s	100.0	0.00	259.8	100.3 s	6.0	0.10	212.1	1,000 s	1.9	0.10	209.7
	HEARTBEAT	92.7 s	100.0	0.40	261.4	103.5 s	90.2	0.10	257.9	1,000 s	10.3	0.20	214.6
	BEACON	105.3 s	100.0	0.20	258.1	200.6 s	69.9	0.20	242.8	1,100 s	13.7	0.20	215.2

Table 6
Per-invocation processing cost of the five firewall checks on the two flight OBCs. Each detector processed 100 K invocations per run using a fixed set of 10 test inputs in five runs. Latency was measured using each platform's cycle counter. The mean input sizes [ranges] in bytes are 118 [64–216], 88 [34–186], 59 [10–146], 66 [17–153], and 66 [17–153] for AS-1–AS-5. *Mean ± SD* gives the mean and sample standard deviation across the five run-level means. *Traffic rate* is the measured detector-input rate under S1, whereas *detector throughput* is input bytes divided by detector execution time (MB = 10^6 B). *Peak stack* is the maximum observed stack usage.

Platform	Detector	Inspected layer	Processing latency Mean ± SD (μs)	Traffic rate (Mbps)	Detector throughput (MB/s)	Peak stack (B)
EnduroSat	AS-1	TC CLTU + TM CADU	4.175 ± 0.005	0.944	28.36	1,636
	AS-2	TC CLTU	8.227 ± 0.010	0.705	10.74	1,836
	AS-3	Space Packet	139.395 ± 0.026	0.468	0.42	2,044
	AS-4	TC Transfer Frame	5.355 ± 0.004	0.525	12.29	1,700
	AS-5	TC Transfer Frame	3.672 ± 0.002	0.525	17.92	1,636
GomSpace	AS-1	TC CLTU + TM CADU	12.858 ± < 0.001	0.946	9.21	1,724
	AS-2	TC CLTU	84.168 ± < 0.001	0.706	1.05	1,836
	AS-3	Space Packet	486.826 ± 0.004	0.391	0.12	2,044
	AS-4	TC Transfer Frame	12.936 ± < 0.001	0.526	5.09	1,724
	AS-5	TC Transfer Frame	12.506 ± < 0.001	0.526	5.26	1,724

Table 7
NASA cFS fixed-time (5/10/15 min): measured iterations (k) and CPU.

Window	Function	Non-Sleep (S0)				Sleep 1 ms (S1)				Sleep 10 ms (S10)			
		Iter	CPU (%)	TΔ	P (mW)	Iter	CPU (%)	TΔ	P (mW)	Iter	CPU (%)	TΔ	P (mW)
5 min	AS-1	27,052k	99.7	–	–	265k	17.8	–	–	30k	10.9	–	–
	AS-2	27,559k	99.8	–	–	265k	17.9	–	–	30k	11.1	–	–
	AS-3	25,944k	99.8	–	–	265k	18.0	–	–	30k	11.0	–	–
	AS-4	28,978k	99.7	–	–	266k	17.3	–	–	30k	10.8	–	–
	AS-5	27,333k	99.8	–	–	266k	17.8	–	–	30k	11.1	–	–
	HEARTBEAT	27,810k	99.8	–	–	266k	17.9	–	–	30k	11.1	–	–
	BEACON	10,502k	99.9	–	–	255k	33.1	–	–	29k	13.2	–	–
10 min	AS-1	54,238k	99.8	–	–	531k	18.0	–	–	59k	11.1	–	–
	AS-2	55,071k	99.8	–	–	531k	18.1	–	–	59k	11.1	–	–
	AS-3	52,126k	99.8	–	–	530k	18.1	–	–	59k	11.1	–	–
	AS-4	56,636k	99.8	–	–	532k	17.5	–	–	59k	10.9	–	–
	AS-5	54,573k	99.8	–	–	531k	18.0	–	–	59k	11.1	–	–
	HEARTBEAT	55,595k	99.8	–	–	531k	18.0	–	–	59k	11.2	–	–
	BEACON	21,042k	99.9	–	–	513k	31.1	–	–	59k	13.3	–	–
15 min	AS-1	81,746k	99.8	–	–	796k	18.0	–	–	89k	11.1	–	–
	AS-2	82,633k	99.8	–	–	797k	18.0	–	–	89k	11.1	–	–
	AS-3	77,631k	99.8	–	–	781k	18.2	–	–	89k	11.1	–	–
	AS-4	83,179k	99.6	–	–	797k	17.6	–	–	89k	10.9	–	–
	AS-5	81,500k	99.7	–	–	798k	18.0	–	–	89k	11.1	–	–
	HEARTBEAT	83,189k	99.7	–	–	795k	18.0	–	–	89k	11.1	–	–
	BEACON	31,538k	99.9	–	–	766k	31.4	–	–	88k	13.2	–	–

Table 8
EnduroSat fixed-time (5/10/15 min), measured: iterations (k), CPU, ΔT, and power consumption (PPK2 @ 3.3 V).

Window	Function	Non-Sleep (S0)				Sleep 1 ms (S1)				Sleep 10 ms (S10)			
		Iter	CPU (%)	TΔ	P (mW)	Iter	CPU (%)	TΔ	P (mW)	Iter	CPU (%)	TΔ	P (mW)
5 min	AS-1	18,009k	100.0	0.40	308.7	299k	6.3	0.44	303.8	30k	3.9	0.22	304.2
	AS-2	23,131k	100.0	0.34	308.5	299k	5.9	0.16	304.1	30k	3.9	0.16	303.4
	AS-3	2,046k	100.0	0.92	306.8	299k	18.6	0.16	303.6	30k	5.2	0.20	304.9
	AS-4	67,169k	100.0	1.08	309.8	296k	6.2	0.28	311.5	30k	4.8	0.06	311.2
	AS-5	23,826k	100.0	0.46	307.6	299k	5.9	0.32	304.7	30k	3.9	0.20	303.9
	HEARTBEAT	270,868k	100.0	1.52	307.2	299k	4.8	0.26	304.0	30k	3.8	0.92	304.2
	BEACON	4,250k	100.0	0.74	304.0	297k	11.6	0.02	304.2	30k	4.7	0.52	303.9
10 min	AS-1	36,017k	100.0	0.20	309.6	598k	6.3	0.56	303.9	60k	3.9	0.20	302.5
	AS-2	46,254k	100.0	1.16	308.0	598k	5.9	0.22	303.3	60k	3.9	0.76	303.1
	AS-3	4,085k	100.0	0.70	306.9	597k	18.6	0.56	303.7	60k	5.2	0.40	302.3
	AS-4	134,391k	100.0	0.72	311.2	591k	6.1	0.10	310.4	60k	4.8	0.06	311.8
	AS-5	46,386k	100.0	0.39	304.6	598k	5.9	0.62	304.4	60k	3.9	0.91	304.9
	HEARTBEAT	533,643k	100.0	0.95	306.2	597k	4.8	0.48	305.0	60k	3.8	0.14	305.9
	BEACON	8,497k	100.0	1.38	304.5	594k	11.6	0.16	305.9	60k	4.7	0.60	305.9
15 min	AS-1	54,028k	100.0	0.58	309.6	897k	6.3	0.14	306.3	90k	3.9	0.42	306.9
	AS-2	69,416k	100.0	0.96	308.6	897k	5.9	0.20	307.8	90k	3.9	0.40	308.4
	AS-3	6,127k	100.0	1.28	313.9	896k	18.6	0.12	308.9	90k	5.2	0.46	307.7
	AS-4	201,641k	100.0	0.12	311.5	887k	6.1	0.48	309.6	90k	4.8	0.16	310.6
	AS-5	71,468k	100.0	0.68	315.3	897k	6.0	0.08	309.5	90k	3.9	0.27	309.8
	HEARTBEAT	798,514k	100.0	1.50	310.0	896k	4.8	0.38	310.1	90k	3.8	0.64	309.8
	BEACON	12,748k	100.0	0.72	303.3	891k	11.5	0.12	309.6	90k	4.7	0.06	309.3

Table 9
GomSpace A3200 fixed-time (5/10/15 min), measured: iterations (k), CPU, ΔT, and power consumption (PPK2 @ 3.3 V).

Window	Function	Non-Sleep (S0)				Sleep 1 ms (S1)				Sleep 10 ms (S10)			
		Iter	CPU (%)	TΔ	P (mW)	Iter	CPU (%)	TΔ	P (mW)	Iter	CPU (%)	TΔ	P (mW)
5 min	AS-1	11,763k	100.0	0.90	261.4	300k	7.6	0.30	214.3	30k	2.1	0.30	210.7
	AS-2	16,079k	100.0	0.40	257.9	300k	7.4	0.00	213.8	30k	2.1	0.00	209.7
	AS-3	580k	100.0	0.40	258.2	249k	46.0	0.20	232.9	29k	6.6	0.10	213.2
	AS-4	15,885k	100.0	0.20	256.1	300k	6.9	0.10	213.6	30k	1.9	0.30	210.9
	AS-5	12,345k	100.0	0.40	259.3	300k	7.6	0.10	214.0	30k	2.0	0.10	210.9
	HEARTBEAT	302k	100.0	0.10	262.0	194k	67.5	0.30	245.0	29k	11.3	0.00	216.1
	BEACON	291k	100.0	0.10	259.1	150k	72.4	0.40	245.6	27k	14.2	0.20	216.8
10 min	AS-1	23,773k	100.0	1.00	261.6	599k	6.7	0.90	213.1	60k	1.8	0.20	209.7
	AS-2	32,499k	100.0	0.30	257.0	599k	6.1	0.40	211.9	60k	1.9	0.60	209.9
	AS-3	1,173k	100.0	0.10	256.9	538k	48.6	0.10	233.2	59k	6.5	1.00	212.2
	AS-4	31,771k	100.0	1.60	255.9	599k	7.1	0.10	213.2	60k	1.9	0.60	210.7
	AS-5	24,955k	100.0	0.50	258.7	599k	7.1	0.50	213.0	60k	1.9	0.30	210.4
	HEARTBEAT	645k	100.0	0.00	261.8	557k	86.7	0.70	256.8	60k	10.6	0.20	214.9
	BEACON	562k	100.0	0.40	258.5	299k	70.0	0.50	243.6	55k	13.6	0.20	215.5
15 min	AS-1	35,660k	100.0	0.20	261.0	899k	7.0	0.50	212.9	90k	1.8	0.30	210.0
	AS-2	48,748k	100.0	0.70	257.3	899k	6.5	0.40	212.4	90k	1.8	0.90	210.5
	AS-3	1,760k	100.0	0.10	256.8	807k	48.4	0.20	233.1	89k	6.5	0.50	212.5
	AS-4	47,655k	100.0	1.20	256.3	899k	7.1	0.20	213.6	90k	1.9	0.10	210.6
	AS-5	37,429k	100.0	0.70	258.0	899k	6.5	0.40	212.6	90k	1.9	0.30	209.9
	HEARTBEAT	967k	100.0	0.10	261.8	837k	87.2	0.10	256.4	90k	10.3	0.30	214.5
	BEACON	839k	100.0	1.70	258.3	449k	69.7	0.60	243.4	82k	13.7	1.30	215.3

Data availability

Source code available at: https://github.com/4g6dcbd21B/CCSDS-based-Firewall

CCSDS-Based Firewall Implementation and Evaluation Dataset (Original data) (GitHub)

References

CCSDS, 2020. Space Packet Protocol. Recommended Standard (Blue Book), (CCSDS 133.0-B-2), Issue 2, June 2020.

CCSDS, 2021a. TC Space Data Link Protocol. Recommended Standard (Blue Book), (CCSDS 232.0-B-4), Issue 4, October 2021.

CCSDS, 2021b. TC Synchronization and Channel Coding. Recommended Standard (Blue Book), (CCSDS 231.0-B-4), Issue 4, July 2021.

CCSDS, 2021c. TM Space Data Link Protocol. Recommended Standard (Blue Book), Issue 3, October 2021.

CCSDS, 2022. Space Data Link Security Protocol. Recommended Standard (Blue Book), (CCSDS 355.0-B-2), Issue 2, July 2022.

CCSDS, 2024. Space Data Link Security Protocol—Summary of Concept and Rationale. Informational Report (Green Book), (CCSDS 350.5-G-2).

CCSDS, 2025. TM Synchronization and Channel Coding. Recommended Standard (Blue Book), (CCSDS 131.0-B-6), Draft, August 2025.

Ahmad, I., Suomalainen, J., Porambage, P., Gurtov, A., Huusko, J., Höyhtyä, M., 2022. Security of satellite-terrestrial communications: Challenges and potential solutions. IEEE Access 10, 96038–96052.

Ansong, S., Rankothge, W., Sadeghi, S., Mohammadian, H., Rashid, F.B., Ghorbani, A., 2025. Role of cybersecurity for a secure global communication eco-system: A comprehensive cyber risk assessment for satellite communications. Comput. Secur. 149, 104156.

Bader, C., 2024. On requirements & concepts for tt&c link key management. In: 2nd Workshop on the Security of Space and Satellite Systems (SpaceSec).

Bassani, J.E., Villa, P.R.C., Bezerra, E.A., Vargas, F.L., 2020. Energy consumption evaluation aimed at CubeSat software development and test. In: Proceedings of the IV IAA Latin American CubeSat Workshop. Paper IAA-BR-20-08-03.

Bocus, M.J., Hakkinen, J., Fontes, H., Drzewiecki, M., Qiu, S., Eder, K., Piechocki, R., 2025. An energy-aware RIoT system: Analysis, modeling and prediction in the SUPERIOT framework. arXiv preprint arXiv:2501.10093.

Boschetti, N., Gordon, N.G., Falco, G., 2022. Space cybersecurity lessons learned from the viasat cyberattack. In: ASCEND 2022. p. 4380.

Chen, J., Pi, D., Wu, Z., Zhao, X., Pan, Y., Zhang, Q., 2021. Imbalanced satellite telemetry data anomaly detection model based on Bayesian LSTM. Acta Astronaut. 180, 232–242.

Consultative Committee for Space Data Systems (CCSDS), 2010. Communications Operation Procedure-1. Recommended Standard (Blue Book), (CCSDS 232.1-B-2), CCSDS, Issue 2, September 2010.

Costin, A., Khandker, S., Turtiainen, H., Hämäläinen, T., 2023. Cybersecurity of COSPAS-SARSAT and EPIRB: Threat and attacker models, exploits, future research. arXiv preprint arXiv:2302.08361.

Cuéllar, S., Santos, M., Alonso, F., Fabregas, E., Farias, G., 2024. Explainable anomaly detection in spacecraft telemetry. Eng. Appl. Artif. Intell. 133, 108083.

De Sombre, W., Tomová, M., Galinski, M., Klein, A., Ortiz, A., 2026. Deep sleep scheduling for satellite IoT via simulation-based optimization. IEEE Internet Things J..

Driouch, O., Bah, S., Guennoun, Z., 2024. CANSat-IDS: An adaptive distributed intrusion detection system for satellites, based on combined classification of CAN traffic. Comput. Secur. 146, 104033.

EnduroSat, 2022. EnduroSat Product Catalogue: Flight-Proven CubeSat Subsystems. https://www.endurosat.com/wp-content/uploads/2022/06/EnduroSat_Products_Presentation_07_22.pdf. UHF Transceiver II maximum gross data rate: 19.2 kbps. (Accessed 30 July 2026).

EnduroSat, 2026. EnduroSat onboard computer. https://www.endurosat.com/. Product page or datasheet, (Accessed 1 March 2026).

European Union Agency for Cybersecurity (ENISA), 2024. LEO Satcom Cybersecurity Assessment. Tech. Rep., European Union Agency for Cybersecurity (ENISA), (Accessed 18 January 2026). Discusses why operators should not rely on "security through obscurity" and highlights risks amplified by use of off-the-shelf components.

European Union Agency for Cybersecurity (ENISA), 2025. Space Threat Landscape 2025. Tech. Rep., European Union Agency for Cybersecurity, Identifies and assesses cybersecurity threats to commercial satellite systems.

Falco, G., 2019. Cybersecurity principles for space systems. J. Aerosp. Inf. Syst. 16 (2), 61–70.

Falco, G., Viswanathan, A., Santangelo, A., 2021. Cubesat security attack tree analysis. In: 2021 IEEE 8th International Conference on Space Mission Challenges for Information Technology (SMC-IT). IEEE, pp. 68–76.

Fayyaz, Y., Yang, L., El-Khatib, K., 2026. Towards resilient intrusion detection in CubeSats: Challenges, TinyML solutions, and future directions. IEEE Aerosp. Electron. Syst. Mag.

Fuchs, C.M., Murillo, N.M., Chou, P., Liou, J.-J., Cheng, Y.-M., Wen, X., Holst, S., Tavoularis, A., Furano, G., Magistrati, G., Marinis, K., Lu, S.-K., Plaat, A., 2019. Fault tolerant nanosatellite computing on a budget. In: Proceedings of the 33rd Annual AIAA/USU Conference on Small Satellites. Paper SSC19-WKVIII-09.

GomSpace, 2026. NanoMind A3200 onboard computer. https://www.gomspace.com/. Product page or datasheet, (Accessed 1 March 2026).

GomSpace A/S, 2019. NanoCom AX100 Datasheet. https://gomspace.com/UserFiles/Subsystems/datasheet/gs-ds-nanocom-ax100.pdf. Document DS 1013823, revision 3.7; receiver signal bit rate up to 38.4 kbps. (Accessed 30 July 2026).

Hamill-Stewart, J., Rashid, A., 2024. Threats against satellite ground infrastructure: A retrospective analysis of sophisticated attacks. In: Proceedings of the 2024 Workshop on Security of Space and Satellite Systems. vol. 1.

Havermans, S., Baumgärtner, L., Roberts, J., Wallum, M., Caballero, J., 2025. Fuzzing space communication protocols. In: Proc. NDSS Symp.

Hundman, K., Constantinou, V., Laporte, C., Colwell, I., Soderstrom, T., 2018. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 387–395.

IRNAS, 2024. ppk2-api-python: Unofficial python API for the Nordic power profiler kit II. https://github.com/IRNAS/ppk2-api-python. (Accessed 30 June 2026).

Jero, S., Furgala, J., Heller, M.A., Nahill, B., Mergendahl, S., Skowyra, R., 2024. Securing the satellite software stack. In: Proceedings 2024 Workshop on Security of Space and Satellite Systems, San Diego, CA, USA: Internet Society.

Kang, M., Park, S., Lee, Y., 2024. A survey on satellite communication system security. Sensors 24 (9), 2897.

Katsube, S., Sahara, H., 2025. Towards an onboard anomaly detection and identification method for satellites. IEEE Access.

Kavallieratos, G., Katsikas, S., 2023. An exploratory analysis of the last frontier: A systematic literature review of cybersecurity in space. Int. J. Crit. Infrastruct. Prot. 43, 100640.

Li, T., Comer, M., Delp, E., Desai, S.R., Mathieson, J.L., Foster, R.H., Chan, M.W., 2019. A stacked predictor and dynamic thresholding algorithm for anomaly detection in spacecraft. In: MILCOM 2019-2019 IEEE Military Communications Conference. MILCOM, IEEE, pp. 165–170.

Mäkitalo, N., et al., 2025. WebAssembly on Resource-Constrained IoT Devices: Performance, Efficiency, and Portability. arXiv preprint arXiv:2512.00035.

Manulis, M., Bridges, C.P., Harrison, R., Sekar, V., Davis, A., 2021. Cyber security in new space: M. Manulis et al.. Int. J. Inf. Secur. 20 (3), 287–311.

Nalepa, J., Myller, M., Andrzejewski, J., Benecki, P., Piechaczek, S., Kostrzewa, D., 2022. Evaluating algorithms for anomaly detection in satellite telemetry data. Acta Astronaut. 198, 689–701.

NASA, 2023. NOS3: NASA Operational Simulator for Small Satellites. https://github.com/nasa/nos3. (Accessed 1 March 2026).

Nordic Semiconductor, 2026. Power profiler kit II (PPK2). https://www.nordicsemi.com/Products/Development-hardware/Power-Profiler-Kit-2. USB power-measurement instrument; current range sub-µA to 1 A, 100,000 samples/s. (Accessed 30 June 2026).

Pavur, J., Martinovic, I., 2022. Building a launchpad for satellite cyber-security research: lessons from 60 years of spaceflight. J. Cybersecur. 8 (1), tyac008.

Saha, S.S., Rahman, S., Ahmed, M.U., Aditya, S.K., 2019. Ensuring cybersecure telemetry and telecommand in small satellites: Recent trends and empirical propositions. IEEE Aerosp. Electron. Syst. Mag. 34 (8), 34–49.

Salim, S., Moustafa, N., Reisslein, M., 2024. Cybersecurity of satellite communications systems: A comprehensive survey of the space, ground, and links segments. IEEE Commun. Surv. & Tutorials 27 (1), 372–425.

Salkield, E., Köhler, S., Birnbach, S., Baker, R., Strohmeier, M., Martinovic, I., 2023. Firefly: Spoofing earth observation satellite data through radio overshadowing. In: Proceedings of the 1st Workshop on Security of Space and Satellite Systems (SpaceSec).

Scharnowski, T., Buchmann, F., Wörner, S., Holz, T., 2023. A case study on fuzzing satellite firmware. In: Proceedings of the 1st Workshop on Security of Space and Satellite Systems (SpaceSec).

Shannon, C.E., 1948. A mathematical theory of communication. Bell Syst. Tech. J. 27 (3), 379–423.

Soesanto, S., 2021. Terra Calling: Defending and Securing the Space Economy: From Science to Fiction and Back to Reality. Cyberdefence Report, Center for Security Studies (CSS), ETH Zurich, (Accessed 18 January 2026).

Stensberg, L.J., 2024. Small satellites with large exposure: How does new space fare in cyberspace? J. Jt. Air Power Competence Cent. (JAPCC) (38), 20–26, (Accessed 18 January 2026).

Thebarge, J.P., Henry, W., Falco, G., 2022. Developing scenarios supporting space-based ids. In: ASCEND 2022. p. 4219.

U.S.China Economic and Security Review Commission, 2011. 2011 Report to Congress of the U.S.China Economic and Security Review Commission. Tech. Rep., U.S. China Economic and Security Review Commission, (Accessed 18 January 2026). See the text box "Malicious Cyber Activities Directed Against U.S. Satellites" for Landsat-7 and Terra (AM-1) incidents.

Willbold, J., Schloegel, M., Bisping, R., Strohmeier, M., Holz, T., Lenders, V., 2024a. Vsaster: Uncovering inherent security issues in current vsat system practices. In: Proceedings of the 17th ACM Conference on Security and Privacy in Wireless and Mobile Networks. pp. 288–299.

Willbold, J., Schloegel, M., Göhler, F., Scharnowski, T., Bars, N., Wörner, S., Schiller, N., Holz, T., 2024b. Scaling software security analysis to satellites: Automated fuzz testing and its unique challenges. In: 2024 IEEE Aerospace Conference. IEEE, pp. 1–12.

Willbold, J., Schloegel, M., Vögele, M., Gerhardt, M., Holz, T., Abbasi, A., 2023. Space odyssey: An experimental software security analysis of satellites. In: 2023 IEEE Symposium on Security and Privacy. SP, IEEE, pp. 1–19.

Wu, X., Du, Y., Fan, T., Guo, J., Ren, J., Wu, R., Zheng, T., 2023. Threat analysis for space information network based on network security attributes: a review. Complex & Intell. Syst. 9 (3), 3429–3468.