Simulating and Observing Satellite Threats: A Monitoring-Aware Cyber Range for Satellite Security Training

Lorenzo Bracciale, Matteo Ciccaglione, Alessandro Amici, Fabio Patrone, Nour Badini, Mario Marchese, Andrea Detti, Daniel Xhakalliu, Giuseppe Bianchi, Michele Luglio, Luca Fiscariello, Cesare Roseti, Arianna Miraval and Pierpaolo Loreti

***Abstract*—Satellite systems are increasingly targeted by cyber threats, yet training platforms that reproduce realistic space-ground conditions remain limited. OpenSatRange is a domain-specific cyber range designed to support satellite security training through integrated simulation, emulation, and monitoring capabilities. It combines accurate network simulation (NS-3) for orbital and link modeling, low-level emulation of communication links (OpenSAND), and full-stack scenario deployment using containerized sandboxes orchestrated via KYPO and SDN. Unlike generic cyber ranges, OpenSatRange supports both LEO and GEO constellations and enables detailed observability of trainee actions through embedded telemetry, facilitating real-time instructor feedback and post-exercise analysis. We describe the system architecture, deployment workflow, and telemetry design, and present illustrative training scenarios such as broadcast hijacking and insecure ground control that demonstrate the platform's flexibility and educational impact. The platform enables training that goes beyond binary outcomes, offering visibility into the reasoning processes that lead to success or failure, crucial for effective cybersecurity education in the space domain.**

***Index Terms*—Satellite security, cyber range, emulation, simulation, SDN, OpenSAND, KYPO, cybersecurity training, telemetry, LEO, GEO, NS-3**

I. Introduction

The increasing reliance on satellite infrastructures for communication, navigation, earth observation, and defense operations has amplified their strategic relevance and consequently, their exposure to cyber threats. Historically, space systems were perceived as inherently secure due to their physical isolation, proprietary protocols, and high cost of access [1], [2]. However, this assumption no longer holds. The proliferation of affordable ground station kits, open-source satellite software, and cloud-based services such as Ground Station as a Service (GSaaS) has dramatically lowered the technical and economic barriers to accessing satellite networks [3].

At the same time, adversaries have demonstrated growing interest and capability in targeting the satellite ecosystem. Real-world incidents such as the KA-SAT modem wipe during the 2022 Ukraine conflict [4], [5], [6], GPS spoofing affecting maritime and aerial operations [5], [7], and (so far) theoretical attacks on broadcast encryption schemes [8] illustrate the diversity and severity of emerging threats. These attacks affect not only the satellite payloads but also the extended infrastructure, including control segments, communication channels, firmware updates, and inter-satellite links.

As the complexity and accessibility of the attack surface grow, so does the need for proper cybersecurity risk assessment procedures [9], but also targeted training and preparation. Operators, engineers, and security analysts must be equipped not only with theoretical knowledge but with practical, domain-specific skills to detect, analyze, and respond to threats across the satellite-ground continuum. Traditional security training platforms, however, rarely capture the constraints and idiosyncrasies of satellite systems. This motivates the development of specialized environments that simulate real-world space system behaviors while enabling hands-on cybersecurity training under realistic conditions.

A. Gaps in existing training infrastructures

While cyber ranges have become increasingly common for security training across various domains—including enterprise IT, industrial control systems, and critical infrastructure they often fall short when applied to the satellite domain. General-purpose platforms such as KYPO [10] or Nautilus [11] provide valuable abstractions for simulating adversarial behavior and conducting Capture-The-Flag (CTF) exercises, but they typically lack support for satellite-specific elements such as orbital dynamics, radio link modeling, and multi-layered ground-space architectures.

Moreover, most existing cyber ranges reduce training outcomes to binary success metrics, such as whether a specific flag has been captured, without capturing the detailed reasoning or decision-making path followed by the trainee [12]. This lack of process-level visibility limits the pedagogical value of the exercise, as instructors are unable to assess the

L.Bracciale, G.Bianchi, M.Luglio, A.Detti and P.Loreti are with Department of Electronic Engineering, University of Rome Tor Vergata, Rome, Italy (email: {lorenzo.bracciale,giuseppe.bianchi,michele.luglio,andrea.detti, pierpaolo.loreti}@uniroma2.it)

L.Bracciale, M.Ciccaglione, A.Amici, D.Xhakalliu, G.Bianchi and P.Loreti are with National Laboratory of Network Assessment, Assurance and Monitoring, CNIT, Rome, Italy (email: {matteo.ciccaglione,alessandro.amici,daniel.xhakalliu}@cnit.it

M.Ciccaglione is with DICII, Tor Vergata University of Rome, Rome, Italy.

A.Miraval is with ASI - Agenzia Spaziale Italiana (email: arianna.miraval@asi.it)

L.Fiscariello and C.Roseti are with RomARS - Rome Applications Research Systems (email:{fiscariello,roseti}@romars.it)

F.Patrone,M.Marchese and N.Badini are with the Department of Electrical, Electronic, Telecommunications Engineering and Naval Architecture (DITEN), University of Genoa (UniGe), 16145 Genoa, Italy (e-mail: {fabio.patrone01,mario.marchese}@unige.it, nour.badini@edu.unige.it)

student's strategic approach, tool choices, or misconceptions that occurred during the scenario. This lack of observability limits the instructor's ability to evaluate progress, provide targeted feedback, or adapt the difficulty in real time. In satellite systems, where latency, determinism, and physical constraints affect system behavior, such limitations are particularly pronounced.

Another common limitation is the rigid user management model. Many platforms require centralized administration for creating or modifying users and scenarios, which is impractical in educational or operational contexts where instructors must rapidly configure exercises for diverse teams.

These gaps underscore the need for a satellite-focused cyber range that not only supports realistic space-specific scenarios but also integrates advanced monitoring, flexible role management, and scenario customization as first-class capabilities.

B. Contributions

This paper presents *OpenSatRange* (OSR), a modular and monitoring-aware cyber range specifically designed for satellite cybersecurity training. OSR addresses the limitations of traditional training platforms by combining realistic satellite simulation and emulation with comprehensive user monitoring and flexible role-based access management.

Our main contributions are as follows:

- **Domain-specific architecture:** We design and implement a cyber range tailored to satellite environments, supporting both LEO and GEO configurations, custom radio link models, and realistic training scenarios derived from actual attack patterns.
- **Integrated monitoring infrastructure:** We introduce a telemetry subsystem that collects, processes, and visualizes real-time user activity, including terminal commands, web interactions, and exploit usage, enabling trainers to supervise and evaluate exercises in detail.
- **Multi-role training workflow:** The platform supports scalable training sessions involving instructors and students with distinct access rights, enforced through federated authentication and a role-aware API layer.
- **Deployment and usage in realistic settings:** We report on the deployment of OSR in a functional training context, including validation through operational testing and preliminary user feedback in educational environments.

II. Related Work

A. Satellite-specific cyber threats

The security of satellite systems has historically been underestimated, often relying on high deployment costs and proprietary protocols as implicit deterrents to cyber attacks [1], [13], [14]. However, recent incidents have shown that such assumptions are no longer tenable. A prominent case is the 2022 cyberattack on Viasat's KA-SAT network (known as AcidRain) which occurred during the Russian invasion of Ukraine. The attack involved the deployment of a wiper malware that rendered tens of thousands of satellite modems inoperable across Europe and Ukraine, indirectly disrupting critical services including over 5,800 wind turbines in Germany [4], [5].

Satellite positioning services are among the most widely known and frequently targeted components of satellite infrastructure. Attacks such as spoofing and jamming have become increasingly accessible due to the availability of low-cost tools like software-defined radios (SDRs), significantly lowering the entry barrier for adversaries [8], [7].

Real-world cases of GPS spoofing and meaconing have been reported in both academic literature and operational scenarios. One widely cited example is the alleged capture of the RQ-170 Sentinel drone by Iran, reportedly achieved through spoofed navigation signals [5]. The continued reliance on unauthenticated civilian GNSS signals makes such attacks feasible even with modest technical resources [7], [15].

In parallel, researchers have uncovered critical vulnerabilities in the firmware and onboard communication protocols of satellites. Willbold et al. [16] conducted an analysis showing that operational systems often lack robust authentication, firmware integrity verification, and access controls, leaving both spaceborne and ground-based components susceptible to compromise.

In the context of LEO constellations, Giuliari et al. [8] demonstrated the feasibility of DDoS-style attacks that saturate inter-satellite links, potentially degrading communication services across entire geographic areas. The widespread availability of orbital data and satellite configuration parameters further increases the exposure of these systems to targeted reconnaissance and attack planning. A comprehensive threat landscape for the New Space paradigm is provided by Manulis et al. [17], who map attack vectors across spaceborne, ground, and communication segments.

B. Cyber ranges for vertical domains (ICS, military, space)

Cybersecurity training has progressively moved away from generic, IT-focused models toward domain-specific programs tailored to the distinct characteristics of vertical sectors. This evolution reflects the increasing need for specialized expertise in environments where operational constraints, legacy systems, and unique threat models render traditional IT training inadequate.

In the industrial sector, for instance, initiatives like CISA's ICS training programs [18] emphasize the challenges of securing real-time control systems, low-latency networks, and long-lived infrastructure. Similarly, academic platforms such as KYPO4Industry [19] provide hands-on training in realistic industrial control environments, allowing trainees to explore attacks and defenses under near-operational conditions.

In the healthcare domain, programs developed by institutions like SANS [20] address sector-specific issues such as ransomware targeting hospital infrastructure and the protection of sensitive medical data, aligning technical exercises with regulatory frameworks like HIPAA.

The defense and aerospace sectors have also adopted this verticalized model. The U.S. Department of Defense, through the Cyber Crime Center (DC3), delivers advanced courses on critical infrastructure protection and forensic investigation in operational scenarios [21]. Similarly, centers such as the University College Dublin's CCI [22] train law enforcement professionals in OSINT and digital forensics, adapting techniques to real-world investigative contexts.

These examples illustrate the growing recognition that cybersecurity training must be context-aware. In this landscape, the space domain presents a particularly challenging environment: physical separation between segments, proprietary communication protocols, broadcast links, and orbital dynamics define a threat surface unlike any other. Addressing these challenges requires training platforms specifically engineered for space operations, bridging simulation, emulation, and vertical observability.

C. ESA initiatives and satellite-focused training platforms

Unlike conventional cyber ranges targeting IT or ICS domains, satellite-oriented platforms must account for the distinctive constraints of space systems, including orbital dynamics, proprietary communication protocols, and the coupling between ground and space segments.

In Europe, the European Space Agency (ESA) has established the first dedicated Space Cyber Range in Tallinn, Estonia, to support attack-defense simulations and resilience testing for aerospace infrastructure [23]. In parallel, NASA has developed NASA IV&V's Cyber Range for Space Systems [24] a cyber range supporting the "Capture the Spacecraft" feature, and NOS3 (NASA Operational Simulator for Small Satellites), an open-source framework originally designed for the STF-1 mission, which enables full-cycle software testing for CubeSats in virtual environments, supporting early validation and risk reduction [25].

Academic efforts such as the SAAMD testbed (Satellite, Aerospace, Avionics, Maritime, and Drone) further illustrate the need for integrated environments that span multiple domains, enabling realistic security assessments across interconnected platforms [26].

Building on this landscape, OSR offers a modular and open-source platform promoted by the Italian Space Agency (ASI). It enables realistic training exercises involving attack simulation, GNSS interference, and ground-segment compromise. Distinctive features include support for both LEO and GNSS scenarios, SDR integration, and a telemetry infrastructure designed to enhance instructor insight and post-exercise evaluation [27], [28].

III. System Overview

A. Design goals and assumptions

The design of OpenSatRange (OSR) is guided by the need to provide a realistic, flexible, and observable environment for satellite cybersecurity training. Unlike traditional IT infrastructures, satellite systems pose unique challenges due to their physical constraints, real-time communication properties, and tight coupling between software, hardware, and mission-specific configurations.

We identify the following design goals:

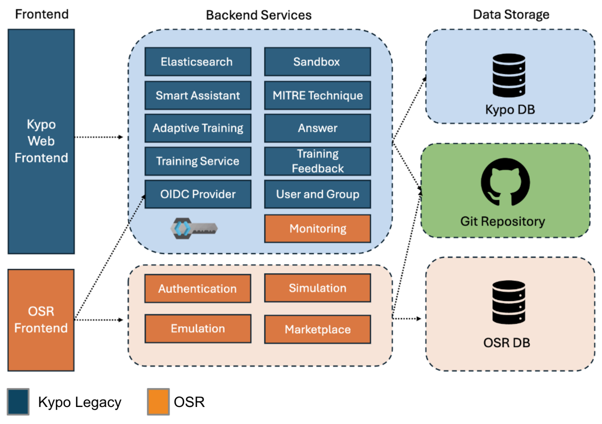


Fig. 1: OSR architecture based on KYPO, with extended modules highlighted

- **Domain realism:** Accurately model satellite-specific communication dynamics, including orbital motion, signal degradation, and broadcast/multicast patterns typical of LEO and GEO networks.
- **Hands-on training capability:** Allow trainees to interact directly with simulated and emulated components using real tools (e.g., Metasploit, custom firmware payloads), in a way that mirrors operational environments.
- **Observability and supervision:** Enable instructors to monitor user activity in real time, trace command execution, and collect telemetry for post-exercise evaluation and behavioral analysis.
- **Scenario modularity and reuse:** Allow trainers to create, adapt, and share scenarios easily across multiple use cases and training sessions.
- **Multi-role access control:** Support instructors and trainees with distinct privileges, relying on scalable authentication and authorization mechanisms.

The design assumes a trusted deployment environment, where the infrastructure can be provisioned on a private cloud or secure physical server, and where trainers have administrative control over scenario instantiation and monitoring. The platform is designed to operate without internet connectivity when necessary, in line with air-gapped training requirements.

B. Platform architecture

OpenSatRange builds upon the open-source KYPO Cyber Range [10], extending it to support training scenarios specific to satellite networks.

As shown in Figure 1, OSR follows a three-layer architecture: a frontend interface, backend microservices, and a storage layer with databases and repositories.

The frontend layer includes both the legacy *KYPO Web Frontend*, which continues to provide access to existing training functionalities, and the new *OSR Frontend*, developed specifically to handle satellite-specific features and workflows. These frontends interact with the backend microservices via standard APIs and share a common authentication system based on protocols like OAuth, ensuring seamless integration and secure user management.

The backend uses a microservices architecture with components deployed as containers or VMs. OSR reuses core KYPO services such as sandbox management, orchestration, user handling, adaptive training, to maintain compatibility and ensure a unified experience.

In addition to the KYPO components, OSR introduces several new backend modules to support satellite-based training. Specifically:

- A *Simulation module* that models satellite constellations, delays, intermittent channels, and orbital behavior, enabling realistic scenario setup.
- An *Emulation module* that supports satellite-specific protocols and allows the execution of scenarios in environments that mimic real-world satellite network conditions.
- A *Marketplace module* designed to facilitate the reuse and sharing of training scenarios. This component enables instructors to browse, retrieve, and upload preconfigured satellite and ground scenarios through an organized interface backed by Git repositories.

The data layer includes both the original KYPO database and a new OSR database, along with integration with Git repositories, hosted on GitLab in OSR's case, for storing configuration files, sandbox descriptors, and training definitions. Importantly, the satellite scenarios maintain full compatibility with KYPO's data formats and training descriptors. This design choice ensures that satellite-based scenarios can be deployed using the same emulation infrastructure as terrestrial ones.

Finally, OSR enhances the monitoring subsystem by extending its capabilities to visualize and track dynamic satellite components during training sessions. This provides instructors with greater insight into the behavior of the simulated environment and allows for a more accurate assessment of student performance in satellite-specific exercises.

C. Satellite simulation layer

The satellite simulation layer of OpenSatRange offers a flexible and extensible framework for evaluating satellite communication scenarios prior to full-scale emulation. Built on top of the NS-3 discrete-event network simulator, this layer extends NS-3's native functionality with satellite-specific models for mobility, antennas, and channels, enabling the simulation of both LEO and GEO communication systems.

Figure 2 illustrates the simulation workflow. Users can define the scenarios through the OSR web interface by specifying parameters such as satellite type (LEO/GEO), orbit characteristics, ground stations, user terminals, frequency bands, antenna gains, and environmental settings. The system accepts user-provided TLE files or generates synthetic constellations as needed. Once the configuration is complete, the simulation is triggered via REST API and executed in the VM-SIM backend. Simulation results including metrics like satellite visibility, propagation delay, throughput, and routing paths, are exported as json files and saved in the market place. These outputs support scenario validation and serve as ground truth for downstream emulation phases, where they can guide SDN route activation or the configuration of OpenSAND emulation links. All artifacts are stored per user to ensure reproducibility and version control.

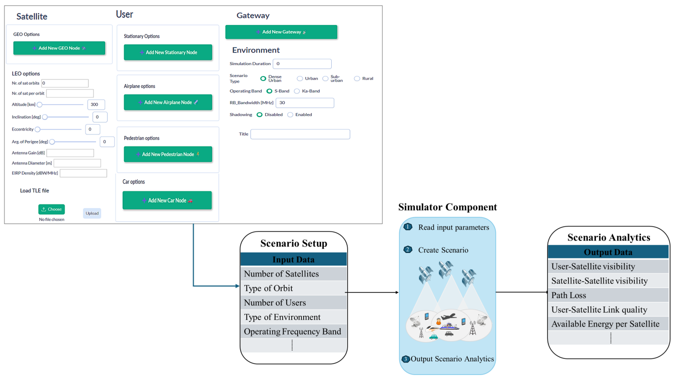


Fig. 2: Workflow of the simulation engine.

To accurately simulate the physical and spatial dynamics of satellite networks, the simulator replaces NS-3's default Cartesian coordinate system with an Earth-Centered Earth-Fixed (ECEF) model, as recommended by 3GPP. In this model, node positions are referenced to the Earth's center, with a mean Earth radius of $R_e = 6{,}371$ km. This enables realistic modeling of long-range communications where curvature and elevation angles must be accounted for. A custom satellite mobility model was implemented in C++, integrating NORAD's SGP4 algorithm for precise orbit tracking using TLE data. This allows the simulation of real satellite constellations such as Starlink or the generation of synthetic orbital patterns by adjusting key orbital parameters.

The simulator also incorporates detailed antenna models to capture directional gain and polarization effects. Satellite antennas are modeled as circular apertures with circular polarization, following 3GPP TR 38.811. The gain pattern is given by:

$$G_a(\theta) = \begin{cases} 1 & \text{if } \theta = 0^\circ, \\ 4\left|\frac{J_1(k\cdot\ell\cdot\sin\theta)}{k\cdot\ell\cdot\sin\theta}\right|^2 & \text{if } 0^\circ < |\theta| \le 90^\circ \end{cases} \quad (1)$$

where θ is the elevation angle, $J_1(\cdot)$ is the Bessel function of the first kind, ℓ is the aperture radius, and $k = 2\pi f_c / c$ with f_c the carrier frequency and c the speed of light.

To model signal propagation, a custom satellite channel model was developed based on 3GPP TR 38.811. The total path loss is computed as:

$$PL = PL_{fs}(d, f_c) + PL_{sh} + PL_c(\theta, f_c) + PL_{sc} + PL_a \quad (2)$$

where PL_{fs} represents the free-space path loss given by

$$PL_{fs}(d, f_c) = 32.45 + 20\log_{10}(d) + 20\log_{10}(f_c), \quad (3)$$

and PL_{sh} accounts for log-normally distributed shadowing. PL_c captures clutter effects due to scattering and reflection, though typically negligible under Line-of-Sight (LOS) conditions. Scintillation loss PL_{sc} varies by frequency and elevation angle—capturing ionospheric effects below 6 GHz and tropospheric effects above 6 GHz—while atmospheric absorption PL_a is modeled as:

$$PL_a(\theta, f_c) = \frac{A_{zenith}(f_c)}{\sin(\theta)}, \tag{4}$$

where $A_{zenith}(f_c)$ is the zenith attenuation.

These enhancements allow the simulator to capture temporal link variations, mobility-induced visibility constraints, and multi-user dynamics in realistic satellite-ground configurations. Static gateways, vehicular terminals, and UAVs can all be modeled, enabling thorough evaluation of coverage, throughput, and routing behavior under diverse operational scenarios.

D. Satellite emulation layer

The emulation layer in OpenSatRange is designed to support both *GEO and LEO satellite constellations*, enabling realistic reproduction of end-to-end communication scenarios across diverse orbital regimes. Upon instantiation, OSR leverages OpenStack [29], an open-source cloud computing platform designed to manage large pools of compute, storage, and networking resources, to dynamically allocate the virtual infrastructure defined in a sandbox descriptor. The resulting topology consists of multiple Linux-based virtual machines, each representing a functional component within a satellite communication ecosystem. These virtual entities include not only satellite elements, such as payload controllers, but also terrestrial components such as ground stations and user terminals. Each virtual machine hosts real services, enabling students to interact with them using standard tools in a realistic training environment.

For **GEO constellations**, the emulation relies on OpenSAND, a user-space satellite network emulator developed by CNES [30]. OpenSAND is deployed as a transparent bridge between virtual machines and enables the execution of the actual satellite communication protocol stack (e.g., DVB-S2), while also replicating the physical and data-link layer characteristics such as latency, bandwidth, framing, and bit error rate. Although physical transmission does not occur, this set-up provides a high-fidelity reproduction of real satellite communication links, particularly suited for space-ground scenarios with high latency and asymmetric bandwidth.

For **LEO constellations**, which present a much more dynamic and scalable scenario, a different strategy has been adopted. A dedicated module was developed to interpret the output of the simulation component, such as orbital positions and inter-satellite visibility, and translate it into a structured virtual network. The emulation of a LEO constellation is performed entirely within a single Linux-based virtual machine, taking advantage of lightweight, kernel-level virtualization tools. In this architecture, each satellite is modeled as an isolated Linux network namespace [31]. These namespaces are arranged in a grid topology that mirrors the logical structure of the constellation itself: columns correspond to orbital planes, while rows represent satellites within each plane. Each namespace is configured with up to four virtual interfaces, enabling it to communicate only with immediate neighbors: the satellites above and below in the same orbital plane, and the adjacent satellites in neighboring planes. The intra satellite links are implemented using virtual Ethernet interfaces (veth) while Linux bridges are employed to interconnect satellites across different planes, maintaining fidelity to the intended constellation topology. Based on this structure, the network is dynamically configured using software defined networking (SDN) technologies, which allow enforcing precise connectivity constraints according to satellite visibility and controlling routing policies as the topology evolves. Emulation of physical layer impairments, such as propagation delay and packet loss, is achieved using the Linux traffic control (*tc*) and network emulation (*netem*) tools, which inject these effects at the network interface level. Thanks to this lightweight and modular design, OpenSatRange can emulate even large-scale LEO constellations without incurring excessive computational overhead. Instead of instantiating one virtual machine per satellite, the use of namespaces significantly reduces resource usage while preserving the flexibility needed to model realistic inter-satellite links, frequent handovers, and fast-changing network topologies.

This hybrid architecture enables the emulation of realistic communication scenarios, preserving physical realism while maintaining full control and traceability. As a result, trainees can interact with actual software stacks and experience authentic satellite conditions—such as delayed acknowledgments, asymmetric bandwidth, and link interruptions—in a controlled and observable environment.

IV. TRAINING WORKFLOW AND MONITORING INFRASTRUCTURE

A. Role-based access control and user management

User authentication and authorization in OpenSatRange are managed by a centralized Keycloak identity provider [32], which supports standard protocols such as OpenID Connect and OAuth2. Each user is assigned a custom *Role* attribute, either *student*, *professor*, or *admin*, which is embedded within the JSON Web Token (JWT) issued at login. This role governs both frontend behavior and backend access control.

The OSR frontend is accessible exclusively to users with the *professor* role, who can manage student accounts, configure simulations, and create sandbox descriptors. Students cannot access the OSR frontend but interact directly with the KYPO platform. Both user types authenticate via Keycloak and receive role-specific permissions for KYPO features. FastAPI backend microservices enforce access control by validating JWTs and authorizing operations based on the embedded role.

Professor accounts are manually provisioned by administrators through the Keycloak console, with the *Role* attribute set accordingly. On the other hand, student accounts are created and managed by professors via RESTful FastAPI endpoints exposed by the OSR frontend. These endpoints require JWT-based authentication, and the backend authorizes CRUD operations on student entities exclusively to users with the *professor* role. This design decentralizes user management while maintaining centralized, role-based security enforcement.

B. Marketplace, sandbox instantiation and execution

The training workflow is organized around the concept of a marketplace, where each training scenario is encapsulated as

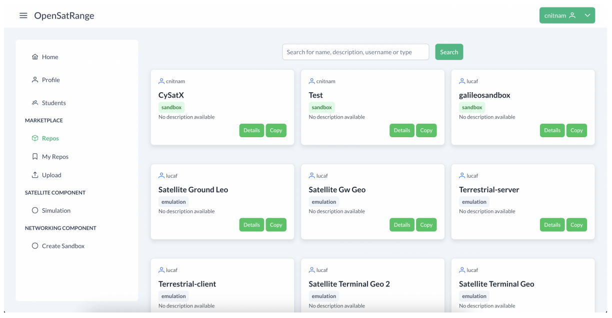


Fig. 3: OSR Marketplace. Each scenario contains infrastructure-as-code descriptors and pedagogical content

a GitLab repository comprising infrastructure-as-code (IaC) descriptors and detailed exercise documentation. Instructors interact with the OSR frontend to browse, clone, and tailor these repositories to their specific teaching needs, as shown in figure 3.

Upon initiation of a training session, the instructor selects one or multiple scenarios from the marketplace and triggers sandbox instantiation. The KYPO orchestrator interfaces with the corresponding Git repository, parses the declarative YAML configuration files, and dynamically provisions the defined containerized workloads and virtual network topologies. Each sandbox instance is deployed on the underlying physical or virtualized infrastructure with strict Linux namespace and filesystem isolation, and is integrated into the designated segment of the software-defined networking (SDN) overlay.

Sandbox lifecycle management—including instantiation, suspension, reset, and teardown—is fully controlled via the instructor's dashboard. The execution environment provisions real-world software stacks and intentionally vulnerable services, with embedded monitoring agents configured at deployment time for runtime telemetry and behavior analysis. Students access their sandboxes through secure terminal or web-based interfaces, authenticated using scenario-specific credentials and governed by role-based access policies defined within the scenario descriptor.

This architecture supports scalable concurrent training sessions across multiple users or teams, ensuring consistent sandbox state management and comprehensive audit trails for all modifications and interactions.

C. Logging infrastructure and telemetry design

The monitoring component developed for the OSR cyber range extends and integrates with the existing KYPO infrastructure, providing a scalable and modular solution for real-time activity tracking during cyber training sessions together with post-exercise analysis. It consists of a *multilayered telemetry infrastructure* that captures user activity across all components of the training environment. Lightweight agents deployed in virtual machines sandboxes (VMs), record satellite commands and informations, shell commands, web interactions and Metasploit commands; the *Management Area Network (MAN)* which is a special VM, deployed in every sandbox, transparent to the user, handles the routing of monitoring messages from the agents of the VMs sandboxes to the central system; central system components perform log ingestion, processing, storage, and visualization.

A key innovation is the implementation of a dedicated *satellite logging pipeline*, designed to capture user interactions with aerospace-related components. Commands and telemetry related to satellite subsystems are intercepted via a custom logging script and tagged with the identifier `sat`. These structured logs (in JSON/RFC5424 format) include unique correlation IDs and metadata and are routed in real time via MAN to the *syslog-ng* server.

This monitoring system is seamlessly integrated with the emulation part to log all user actions and all topology changes within the constellation, ensuring accurate traceability of the network's evolution over time. Each event is recorded with a corresponding type, a descriptive comment, and the update status. The system captures different types of updates: inter- and ground-to-satellite visibility changes; link metrics (bandwidth, loss, delay) for both ground and inter-satellite connections; routing paths dynamically computed between ground nodes; and user interactions such as command-line executions.

All logged information is periodically transmitted to the monitoring platform as soon as the corresponding updates are applied to the virtual infrastructure, enabling real-time visibility and post-analysis of the emulated environment.

Syslog-ng service routes logs based on their tags (e.g., *bash*, *metasploit*, *training*, *sat*) to dedicated *Logstash* pipelines. The satellite pipeline specifically parses and enriches satellite-related data (e.g. visibility, configuration parameters) and forwards the processed information to *Elasticsearch*, enabling advanced queries, indexed search, and historical data analysis.

A major strength of the architecture lies in the ease of integration of the monitoring component into sandbox environments. The deployment of monitoring agents is fully automated and triggered during the *User Ansible* stage of the OSR provisioning toolchain (during the creation of the sandbox-pool). Trainers can enable monitoring by simply referencing the GitLab repository `osr-sandbox-logging.git` in the `requirements.yml` file of their training project. This GitOps-based approach allows for selective inclusion of the logging agents on a per-VM basis with minimal configuration overhead.

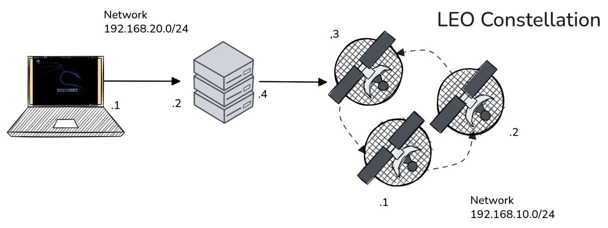
The web-based frontend enables trainers to visualize user activity in real time, as shown in the Appendix.

V. Training Scenarios

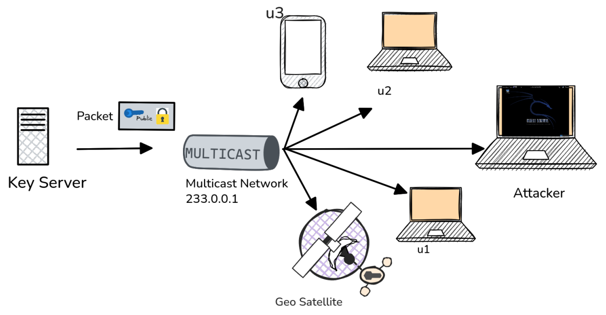
A. Ground segment compromise

Story: A large company operates a Ground Station as a Service (GaaS) platform accessible through a web application hosted internally. Due to a complex security flaw, an external attacker gains partial access to the server running this service. Leveraging that access, the attacker extracts and modifies the satellite firmware, which is later uploaded to a satellite through an automated update process. This allows the attacker to compromise orbiting satellites by pivoting through the ground station.

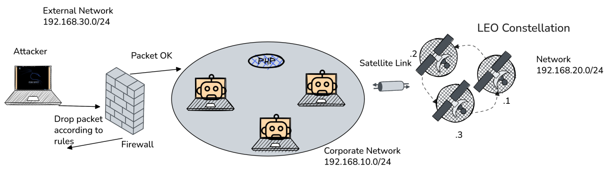
Network: The network setup, shown in Figure 4a, includes a constellation of LEO satellites. The configuration guarantees



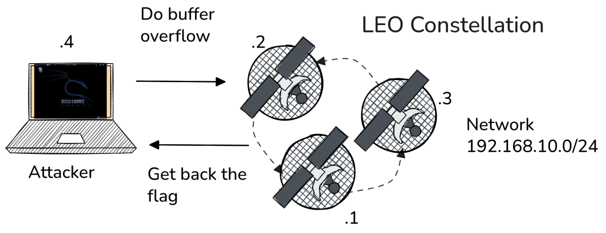
(a) Ground segment compromise



(b) Broadcast channel decryption



(c) LEO constellation DDoS



(d) Firmware exploitation and satellite hijacking

Fig. 4: Network Topology for all scenarios

that at least one satellite is within communication range of the ground station at all times, enabling continuous interaction for updates and telemetry.

Walkthrough: The exercise begins with reconnaissance on the ground station's exposed web assets. Students perform port scanning and directory enumeration, leading to the discovery of an exposed Git repository [33]. Analysis of the repository reveals configuration files referencing a hidden virtual host. This host runs a test interface that allows basic network checks and is vulnerable to a command injection flaw. Exploiting this bug gives shell access to the ground station. Once inside, students must locate the firmware binary, extract it, and reverse-engineer it. Their goal is to modify the logic to bypass conditional checks and force the firmware to transmit a predefined secret flag. The altered firmware is then repackaged into CCSDS-compliant packets. A scheduled routine simulates the update process, uploading the patched firmware to a satellite, which then emits the flag—marking successful exploitation.

Relevance: Cyberattacks targeting ground infrastructures in satellite systems are increasingly common. In 2022, Viasat's KA-SAT network was hit by the *AcidRain* malware, which wiped satellite modem flash memory across Europe and Ukraine [4]. Similarly, Russian satellite operator Dozor-Teleport experienced a breach affecting its customer management systems, disrupting terminal operations [34]. These incidents highlight the critical role of ground segment security in protecting space-based assets.

B. Broadcast channel decryption

This scenario focuses on the role of cryptography in securing satellite communications.

Story: This scenario demonstrates how a misconfiguration in a custom key exchange protocol can compromise the confidentiality of satellite communications. A central satellite periodically broadcasts a message containing an encrypted symmetric key. The message includes:

- An RSA public key (PEM format);
- The symmetric key encrypted with that public key.

Each terminal checks whether the public key matches its own private key and, if so, decrypts the symmetric key for secure communications. An attacker, passively monitoring the broadcast channel, leverages a flawed distribution of RSA keys to perform a common factor attack, recovering the symmetric key and decrypting sensitive data.

Network: As shown in Figure 4b, the setup involves a GEO satellite broadcasting to multiple ground terminals over a shared communication channel.

Walkthrough: The scenario begins with network traffic analysis to identify the broadcast channel, transport protocol (UDP), and port number. Using Wireshark, students isolate relevant packets and extract messages. A Python script is then developed to parse these messages, extract RSA public keys in PEM format, and perform an RSA Common Factor Attack [35]. Successful exploitation reveals two private keys, allowing decryption of two encrypted payloads containing the full flag.

Relevance: Cryptographic weaknesses in satellite systems are a tangible risk. For instance, real-time attacks on the GMR-2 encryption used in satellite phones have demonstrated how improperly implemented ciphers can be reversed in practice [36].

C. LEO constellation DDoS

Story: This scenario emulates the ICARUS attack [8], targeting a misconfigured corporate network protected by a faulty firewall and gateway. By compromising one internal machine, the attacker builds a botnet to launch a Distributed Denial of Service (DDoS) attack against a satellite link, saturating the limited bandwidth and disrupting inter-satellite communications.

Network: The topology (Figure 4c) includes a corporate network connected to a satellite system via a limited-bandwidth link. A test machine periodically pings two satellites and confirms the attack's success when one becomes unreachable.

Walkthrough: The attack begins with reconnaissance on the firewall, which proxies access to an internal web server. This proxy performs MIME-type validation for uploaded image files. The student bypasses this check (e.g., using Burp Suite to modify HTTP headers) to upload a PHP reverse shell. Triggering the payload grants shell access. Next, the student sets up a Command and Control (C2) system. Due to network isolation, the web server machine is promoted as a local C2 server. It relays commands from the student's external C2 host to internal client agents, pre-installed on all network machines. After establishing the botnet, the student identifies a vulnerable satellite link (via `traceroute`) and launches a DDoS attack using `iperf`. Once the link is saturated, the test machine detects the disruption and returns the flag.

Relevance: While DDoS attacks are well-known in traditional networks, recent research, such as the ICARUS study [8], highlights their applicability to satellite systems, where limited bandwidth and critical links create high-impact vulnerabilities.

D. Firmware exploitation and satellite hijacking

This attack targets a vulnerable implementation of a library used to parse telecommands.

Story: This scenario highlights how a low-level vulnerability, such as a buffer overflow in satellite firmware, can be exploited to hijack satellite behavior. The firmware uses unsafe functions like `strcpy`, enabling an attacker to overwrite memory and alter control logic. The target is a weather satellite responding to telemetry requests, with communications wrapped in CCSDS format [37].

Network: Although only one satellite is needed for the attack, the topology (Figure 4d) includes multiple LEO satellites to minimize visibility gaps and make the scenario more realistic.

Walkthrough: Students begin by reverse-engineering the satellite firmware to extract key information: the service port, the CCSDS protocol format, and a vulnerable data structure. The firmware includes a fixed-size buffer for message content, followed by critical control data in memory. Due to the use of unsafe functions like `strcpy`, inputs exceeding the buffer length can overwrite adjacent control bytes. The student crafts a payload to overwrite these values, hijacking execution flow. Since satellite visibility varies, they must scan multiple IPs to find one within range. A successful attack corrupts satellite behavior and triggers the flag.

Relevance: Satellite systems often rely on obscurity for security, using proprietary firmware and undocumented protocols. However, studies like Space Odyssey [16] show that reverse engineering these components is feasible, and vulnerabilities such as buffer overflows remain a critical threat.

VI. Deployment and Preliminary Evaluation

A. End-to-end system deployment on cloud

The OpenSatRange system is fully deployed within a private OpenStack cloud infrastructure, leveraging the flexibility and scalability offered by cloud-native technologies. At the core of the deployment, the system services (Fig. 1) are orchestrated through a Kubernetes cluster instantiated on top of a set of dedicated OpenStack virtual machines. This architectural choice brings several advantages. First, Kubernetes provides a robust and fault-tolerant environment for managing containerized services, enabling automated deployment, self-healing, and load balancing of the core components of the cyber range. Second, the native integration between Kubernetes and OpenStack enables the system services to dynamically interact with the underlying infrastructure, provisioning new virtual machines and virtual networks that form the training sandboxes on demand.

This capability is particularly beneficial in the context of training activities, where each session may involve multiple user sandboxes with isolated and complex network topologies. Thanks to the elasticity of the OpenStack cloud and the orchestration power of Kubernetes, OpenSatRange can scale horizontally by deploying and managing a large number of training environments in parallel. This ensures that multiple sessions can be hosted simultaneously, efficiently utilizing cloud resources while maintaining strong isolation and reproducibility guarantees.

B. Functional testing

Functional testing of the platform was conducted through a combination of developer-led validation and controlled user sessions. Each component was tested individually and in integration, following black-box and white-box methodologies.

The KYPO orchestrator was validated by deploying multiple concurrent sandbox instances for different users, ensuring proper resource isolation, network segmentation, and consistent scenario execution. All predefined scenarios from the OSR marketplace were tested, including those requiring OpenSAND integration and satellite-specific topologies. The configuration parameters, such as satellite type, link conditions, and SDN routing policies, were verified to be correctly interpreted and enforced.

The monitoring infrastructure was tested by executing known activity sequences within the sandboxes (e.g., bash commands, web-based exploitation, reverse shells), and verifying their accurate collection and display in the instructor dashboard. Event latency and logging throughput were evaluated under load, and the system was able to sustain real-time updates for over 10 concurrent users with no observable lag.

Simulation modules were tested independently by running predefined satellite configurations via the NS-3 backend and comparing link availability predictions against expected orbital patterns. Logs and graphs generated by the simulator were successfully exported and reused during emulation setup, confirming the intended integration flow.

Overall, functional validation confirms the readiness of the platform for structured training exercises and highlights the effectiveness of its modular design. Preliminary feedback from instructors and testers emphasized the transparency of the orchestration process and the richness of telemetry as key strengths.

VII. Conclusions

This paper presented OpenSatRange, a cyber range architecture designed to support security training in the satellite domain. The platform integrates simulation of orbital and link-layer behavior, network emulation, and automated orchestration of containerized scenarios, enabling controlled and reproducible exercises that reflect the specificities of satellite-ground infrastructures.

Unlike generic cyber ranges, OpenSatRange models satellite-specific characteristics such as orbital dynamics, broadcast communication, and ground-space interactions. Its monitoring infrastructure captures telemetry across multiple layers, providing detailed insights into trainee actions and enabling more structured feedback and assessment.

The current deployment demonstrates the feasibility of this approach and highlights the importance of observability and domain fidelity in vertical training contexts. Future work includes extending the telemetry analysis pipeline and conducting longitudinal studies on learning outcomes across different user profiles.

Appendix
Real-Time Dashboard in Action

This appendix illustrates how the real-time monitoring dashboard was used during a training session based on Scenario 1. The dashboard aggregates telemetry from all active sandboxes into a unified web interface, continuously updated via WebSocket channels.

It provides two main views: *Progress*, showing user advancement through the training flow (e.g., level completions, hints used, answers, assessments), and *Command Timeline*, logging executed commands including satellite terminal actions, Bash shell activity, and Metasploit usage. Examples are shown in Figures 5b and 5a.

Instructors were able to inspect each student's behavior in real time: observing command sequences, open connections, use of offensive tools, and interactions with emulated services. Telemetry events were visualized as synchronized timelines and could be exported as ZIP archives for offline analysis.

This setup allowed trainers to interpret the reasoning behind student actions, regardless of whether a task was successfully completed. Feedback was delivered during the session via chat, video, or scenario annotations. Actions were also tagged with evaluation metrics (e.g., tool selection, detection accuracy, time to compromise) to support both formative and summative assessment.

In multi-session use, the dashboard enabled comparison of user behavior across training cohorts and supported adjustments to scenario difficulty and pacing. The logging system was extended to capture domain-specific events, as detailed in Section IV-C and illustrated in Figure 6, integrating satellite-related telemetry while remaining compatible with KYPO's experimental monitoring pipeline.

References

[1] G. Falco, "The Vacuum of Space Cyber Security," in *AIAA SPACE and Astronautics Forum and Exposition*. American Institute of Aeronautics and Astronautics, 2018.

[2] S. Ansong, W. Rankothge, S. Sadeghi, H. Mohammadian, and F. B. Rashid, "Role of cybersecurity for a secure global communication ecosystem: A comprehensive cyber risk assessment for satellite communications," *Computers & Security*, p. 104156, 2024.

[3] J. Willbold, F. Sciberras, M. Strohmeier, and V. Lenders, "Satellite cybersecurity reconnaissance: Strategies and their real-world evaluation," in *2024 IEEE Aerospace Conference*, 2024, pp. 1–13.

[4] CyberPeace Institute, "Viasat cyberattack case study," https://cyberconflicts.cyberpeaceinstitute.org/law-and-policy/cases/viasat, 2022, accessed: 2025-04-11.

[5] M. Kang, S. Park, and Y. Lee, "A survey on satellite communication system security," *Sensors*, vol. 24, no. 9, p. 2897, 2024. [Online]. Available: https://www.mdpi.com/1424-8220/24/9/2897

[6] S. Salim, N. Moustafa, and M. Reisslein, "Cybersecurity of satellite communications systems: A comprehensive survey of the space, ground, and links segments," *IEEE Communications Surveys & Tutorials*, 2024.

[7] Z. Wu, Y. Zhang, Y. Yang, C. Liang, and R. Liu, "Spoofing and anti-spoofing technologies of global navigation satellite system: A survey," *IEEE Access*, vol. 8, pp. 165 444–165 496, 2020.

[8] G. Giuliari, T. Ciussani, A. Perrig, and A. Singla, "ICARUS: Attacking Low Earth Orbit Satellite Networks," in *2021 USENIX Annual Technical Conference (USENIX ATC 21)*. USENIX Association, 2021, pp. 317–331.

[9] L. Vessels, K. Heffner, and D. Johnson, "Cybersecurity risk assessment for space systems," in *2019 IEEE Space Computing Conference (SCC)*. IEEE, 2019, pp. 11–19.

[10] J. Vykopal, R. Ošlejšek, P. Čeleda, M. Vizváry, and D. Tovarňák, "KYPO Cyber Range: Design and Use Cases," *IEEE Transactions on Education (preprint)*, 2017, available from Masaryk University technical report or conference proceedings; see also https://crp.kypo.muni.cz.

[11] G. Bernardinetti, S. Iafrate, and G. Bianchi, "Nautilus: A Tool for Automated Deployment and Sharing of Cyber Range Scenarios," in *Proceedings of the 16th International Conference on Availability, Reliability and Security (ARES)*. ACM, 2021, pp. 1–7.

[12] M. Glas, G. Messmann, and G. Pernul, "Complex yet attainable? an interdisciplinary approach to designing better cyber range exercises," *Computers & Security*, vol. 144, p. 103965, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404824002700

[13] U.-E. Botezatu, "Space cybersecurity: A survey of vulnerabilities and threats," *Romanian Cyber Security Journal*, vol. 6, no. 2, pp. 53–60, Dec. 2024.

[14] A. Carlo and K. Obergfaell, "Cyber attacks on critical infrastructures and satellite communications," *International Journal of Critical Infrastructure Protection*, vol. 46, p. 100701, 2024.

[15] S. Soderi and J. Iinatti, "CLPDI-based fast secure code estimation and replay attacks on GNSS signals," in *2024 Security for Space Systems (3S)*. IEEE, May 2024.

[16] J. Willbold, M. Schloegel, M. Vögele, M. Gerhardt, T. Holz, and A. Abbasi, "Space Odyssey: An Experimental Software Security Analysis of Satellites," in *2023 IEEE Symposium on Security and Privacy (SP)*. San Francisco, CA, USA: IEEE, 2023, pp. 1–19.

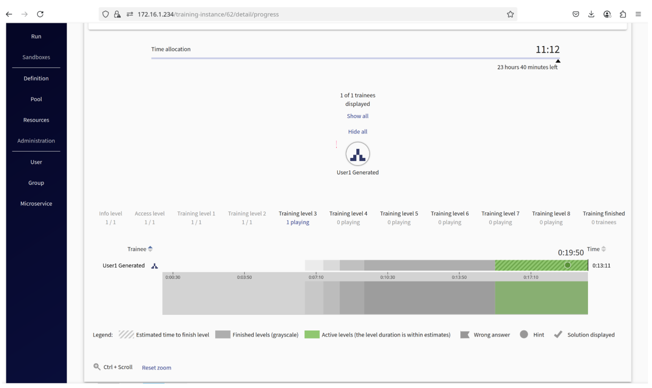
[17] M. Manulis, C. P. Bridges, R. Harrison, V. Sekar, and A. Davis, "Cyber security in new space: Analysis of threats, key enabling technologies and challenges," *International Journal of Information Security*, vol. 20, pp. 287–311, 2021. [Online]. Available: https://doi.org/10.1007/s10207-020-00503-w

[18] Cybersecurity and Infrastructure Security Agency (CISA), "Ics training available through cisa," 2025, accessed April 11, 2025. [Online]. Available: https://www.cisa.gov/resources-tools/programs/ics-training-available-through-cisa

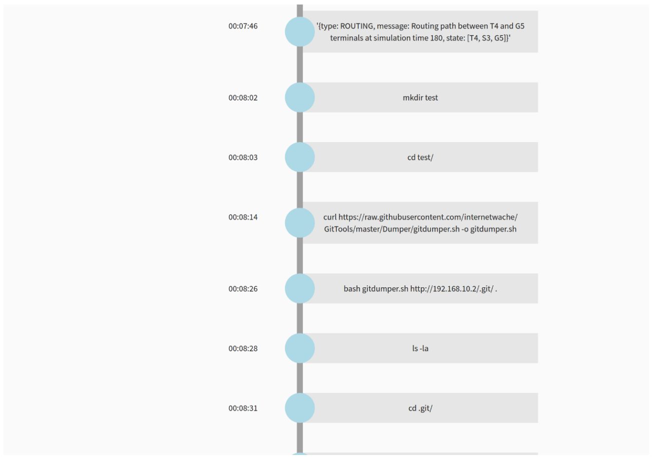
[19] P. Čeleda, J. Vykopal, V. Švábenský, and K. Slavíček, "Kypo4industry: A testbed for teaching cybersecurity of industrial control systems," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE)*, 2020, pp. 1234–1240. [Online]. Available: https://doi.org/10.1145/3328778.3366908

[20] SANS Institute, "Cybersecurity courses & certifications," 2025, accessed April 11, 2025. [Online]. Available: https://www.sans.org/cyber-security-courses/

[21] Department of Defense Cyber Crime Center (DC3), "Cyber training academy," 2025, accessed April 11, 2025. [Online]. Available: https://www.dc3.mil/TrainingAcademy


172.16.1.234/training-instance/62/detail/progress
Run
Sandboxes
Definition
Pool
Resources
Administration
User
Group
Microservice
Time allocation
11:12
23 hours 40 minutes left
1 of 1 trainees displayed
Show all
Hide all
User1 Generated
Info level 1 / 1
Access level 1 / 1
Training level 1 1 / 1
Training level 2 1 / 1
Training level 3 1 playing
Training level 4 0 playing
Training level 5 0 playing
Training level 6 0 playing
Training level 7 0 playing
Training level 8 0 playing
Training finished 0 trainees
Trainee
Time
0:19:50
User1 Generated
0:13:11
0:00:30
0:03:50
0:07:10
0:10:30
0:13:50
0:17:10
Legend: Estimated time to finish level
Finished levels (grayscale)
Active levels (the level duration is within estimates)
Wrong answer
Hint
Solution displayed
Ctrl + Scroll
Reset zoom

(a) Live progress monitoring for Scenario 1



```
00:07:46   '{type: ROUTING, message: Routing path between T4 and G5
           terminals at simulation time 180, state: [T4, S3, G5]}'
00:08:02   mkdir test
00:08:03   cd test/
00:08:14   curl https://raw.githubusercontent.com/internetwache/
           GitTools/master/Dumper/gitdumper.sh -o gitdumper.sh
00:08:26   bash gitdumper.sh http://192.168.10.2/.git/ .
00:08:28   ls -la
00:08:31   cd .git/
```

(b) Extract from the command timeline of a Scenario 1 training exercise

Fig. 5: An example of real time monitoring as seen by the trainer

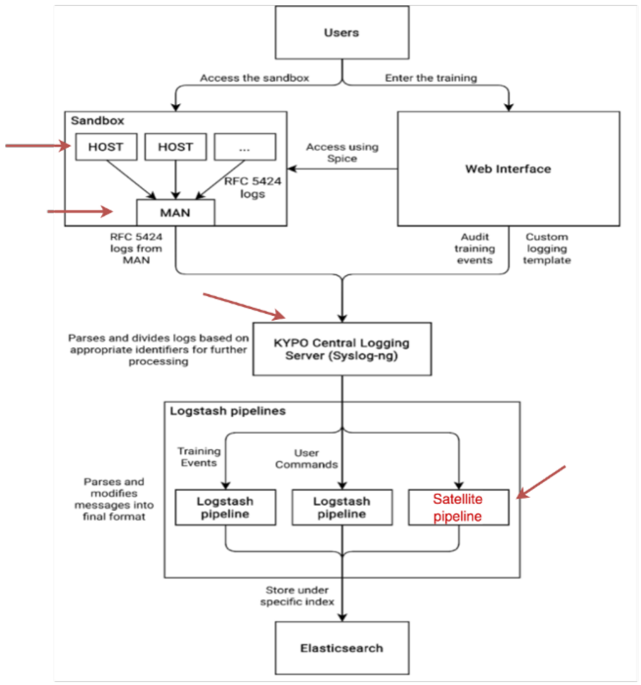


Fig. 6: OSR Monitoring system

[22] University College Dublin, "Centre for cybersecurity & cybercrime investigation (cci)," 2025, accessed April 11, 2025. [Online]. Available: https://www.ucd.ie/cci/

[23] European Space Agency (ESA), "Estonia to host europe's new space cybersecurity testing ground," January 2025, accessed April 11, 2025.

[24] B. Bailey, "Nasa iv&v's cyber range for space systems," in *Annual Ground System Architectures Workshop (GSAW)*, no. GSFC-E-DAA-TN65725, 2019.

[25] S. Zemerick, "Nasa operational simulator for small satellites (nos3)," in *2015 Workshop on Spacecraft Flight Software (FSW-15)*, no. GSFC-E-DAA-TN27300, 2015.

[26] A. Costin, H. Turtiainen, S. Khandker, and T. Hämäläinen, "Towards a unified cybersecurity testing lab for satellite, aerospace, avionics, maritime, drone (saamd) technologies and communications," *arXiv preprint arXiv:2302.08359*, 2023. [Online]. Available: https://arxiv.org/abs/2302.08359

[27] F. Patrone, P. Loreti, L. Fiscariello, L. Bracciale, A. Amici, A. Detti, C. Roseti, F. Zampognaro, M. Luglio, G. Bianchi *et al.*, "Opensatrange: An open cyber range for operators and users of satellite communication networks," in *CEUR WORKSHOP PROCEEDINGS*, vol. 3731. CEUR-WS, 2024.

[28] M. Ciccaglione, L. Bracciale, P. Loreti, and A. M. Zanon, "Cyber range for space systems: Training scenarios for satellite cybersecurity preparedness," 2025.

[29] OpenStack Foundation, "Openstack: Open source software for creating private and public clouds," https://www.openstack.org, 2025, accessed: 2025-06-27.

[30] Centre National d'Études Spatiales (CNES), "Opensand: Open satellite network emulator," https://opensand.org, 2025, accessed: 2025-06-27.

[31] R. Rosen, "Namespaces and cgroups, the basis of linux containers," *Seville, Spain, Feb*, 2016.

[32] S. Thorgersen and P. I. Silva, *Keycloak-identity and access management for modern applications: harness the power of Keycloak, OpenID Connect, and OAuth 2.0 protocols to secure applications*. Packt Publishing Ltd, 2021.

[33] D. Spinellis, "Git," *IEEE Softw.*, vol. 29, no. 3, pp. 100–101, May 2012.

[34] V. Petkauskas, "Russian satellite telecom dozor hit by hackers," 2023, accessed: 2025-04-11. [Online]. Available: https://cybernews.com/cyber-war/dozor-russian-satellite-telecom-hacked/

[35] V. Kumar, A. Roy, S. Sengupta, and S. Sen Gupta, "Parallelized common factor attack on RSA," in *Information Systems Security*, ser. Lecture notes in computer science. Cham: Springer International Publishing, 2017, pp. 303–312.

[36] J. Hu, R. Li, and C. Tang, "A real-time inversion attack on the gmr-2 cipher used in the satellite phones," *Science China Information Sciences*, vol. 61, pp. 1–18, 2018.

[37] The Consultative Committee for Space Data Systems, "CCSDS publications manual," Apr. 2014.