OpenSatRange: An Open Cyber Range for Operators and Users of Satellite Communication Networks

Fabio Patrone[1,*], Pierpaolo Loreti[2], Luca Fiscariello[3], Lorenzo Bracciale[2], Alessandro Amici[2], Andrea Detti[2], Cesare Roseti[3], Francesco Zampognaro[3], Michele Luglio[3], Giuseppe Bianchi[2] and Mario Marchese[1]

[1] *University of Genoa, Genoa, Italy*

[2] *Consorzio Nazionale Interuniversitario per le Telecomunicazioni NAM Lab, University of Rome Tor Vergata, Rome, Italy*

[3] *ROMARS srl, Rome, Italy*

Abstract

In today's rapidly evolving technological landscape and increasing impact of cybersecurity threats and attacks, the development and utilization of cyber ranges have become paramount in enhancing people's knowledge of cybersecurity. Even if multiple of these tools simulating terrestrial communication networks are already in use, a cyber range focused on satellite communication networks is still missing.

This paper describes the ongoing work about the design and development of OpenSatRange (OSR), a cyber range able to simulate/emulate satellite communication networks. Details are provided about the similarities and differences between this tool and the main other cyber ranges, the considered reference scenarios, the defined training exercises, the designed OSR framework and its components, and the planned roadmap to conclude this research project's funded work.

Keywords

Cyber range, Satellite communication networks, OpenStack

1. Introduction

Cybersecurity is an indispensable pillar in today's interconnected digital world, encompassing the protection of systems, networks, and data from malicious cyber threats. Its importance cannot be overstated, given the pervasive nature of cyber attacks that threaten individuals, businesses, governments, and critical infrastructure globally [1]. Effective cybersecurity measures are essential for safeguarding sensitive information, preserving privacy, maintaining trust in digital transactions, and upholding the integrity of systems and services. Moreover, in an era marked by increasing reliance on digital technologies and the proliferation of connected devices, cybersecurity serves as a linchpin for ensuring the resilience and stability of the digital ecosystem. As cyber threats continue to evolve in sophistication and frequency, investing

ITASEC'24, April 08–11, 2024, Salerno, IT

[*] Corresponding author.

fabio.patrone01@unige.it (F. Patrone); pierpaolo.loreti@uniroma2.it (P. Loreti); fiscariello@romars.tech (L. Fiscariello); lorenzo.bracciale@uniroma2.it (L. Bracciale); alessandro.amici@uniroma2.it (A. Amici); andrea.detti@uniroma2.it (A. Detti); roseti@romars.tech (C. Roseti); zampognaro@romars.tech (F. Zampognaro); luglio@romars.tech (M. Luglio); giuseppe.bianchi@uniroma2.it (G. Bianchi); mario.marchese@unige.it (M. Marchese)

in robust cybersecurity measures becomes paramount to mitigate risks, fortify defences, and sustain the functionality of our digital infrastructure and way of life.

One of the first steps to strengthen digital systems making them less vulnerable to cyber attacks is to properly train all people involved in the design, build, configuration, and use of these systems. Cyber ranges are pivotal tools to achieve this goal [2]. They provide controlled, virtual environments where individuals can undergo immersive training to effectively learn how to combat cyber threats. Their significance lies in their ability to simulate diverse scenarios, ranging from basic cybersecurity hygiene practices to advanced threat response strategies [3]. By replicating real-world cyber threats in a safe setting, cyber ranges offer a hands-on learning experience crucial for developing practical skills and critical thinking abilities. Participants can engage in various exercises, including incident response simulations, penetration testing, and malware analysis, enabling them to comprehensively understand and mitigate cyber risks [4].

Cyber ranges offer a versatile toolkit for enhancing cybersecurity knowledge and skills across a broad spectrum of applications and disciplines. Vast is typically the set of offered functionalities and the plethora of considered use cases, as much as the different kinds of systems that can be prone to cyber-attacks in the real world. One example is satellite systems. The risk of cyber attacks on satellite systems is a growing concern in our increasingly interconnected world [5]. These systems play critical roles in various sectors, including telecommunications, navigation, weather monitoring, and national security. However, these systems are not immune to cyber threats. Cyber attackers may target satellite networks to disrupt communication channels, compromise sensitive data, or even sabotage critical infrastructure. The repercussions of such attacks could be severe, potentially leading to widespread disruptions in essential services and compromising national security [6]. Additionally, as satellite technology advances and becomes more integrated with other systems, the attack surface expands, presenting new vulnerabilities that can be exploited by malicious actors.

However, investigating the available cyber ranges, we found that a cyber range dedicated to satellite systems is missing. In this paper, we present OpenSatRange (OSR), an under-development cyber range focused on satellite communication systems. This tool aims to train satellite network operators and users about cyber security practises and principles in order to make them more aware about the cybersecurity aspect and contribute in making these systems more robust against cyber attacks. OSR simulates/emulates satellite communication systems offering ad-hoc training pathways and practice exercises. It allows personalising the scenarios to simulate/emulate allowing users to set multiple network parameters, such as the number of nodes, their position, and the main communication link parameters. Its system architecture has been designed to be scalable and modular in order to allow possible future development and inclusion of additional elements, such as further communication protocols and algorithms.

The paper is structured as follows: Section 2 describes the main cybersecurity frameworks and available cyber ranges in order to offer a description of the current situation on this topic; Section 3 describes the reference scenarios that the OSR platform will focus on; Section 4 details the considered exercises to test the skills of the OSR platform's users tailored on the reference scenarios; Section 5 illustrates the OSR framework structure describing its components into the detail. Conclusions are drawn in Section 6.

2. Open cyber range platforms

In this section, we will describe the characteristics of Nautilus [7] and Kypo [8] cyber range platforms we have studied as a starting point for designing OSR. We will highlight their respective strengths and weaknesses, analyze the functionalities they offer, and identify any missing functionalities needed to support our vision and scenarios.

2.1. Nautilus

Nautilus is a Cyber Range developed within the European H2020 SPARTA project that enables the creation, execution, and automatic sharing of scenarios for cybersecurity exercises. It utilizes advanced cloud technologies to simplify the configuration of vulnerable virtualized environments. It offers a graphical interface and a descriptive language for scenarios and integrates features for data and knowledge sharing. Nautilus aims to foster collaboration among training teams, allowing less experienced trainers to access and reuse advanced exercises prepared by others.

Nautilus consists mainly of three elements: the web application, the deployment framework, and the database. Both the web application and the framework interact with the database to retrieve and update information: the framework has read-only access and does not directly access the database but through an authenticated HTTP endpoint.

The Nautilus database contains all the components of training scenarios, as well as user and information sharing. The deployment framework is a key component as it handles all phases of distributing and provisioning a training scenario. The web application is the main interface of the Nautilus cyber range, allowing the creation, modification, and distribution of scenarios, both locally and remotely; the web application also implements knowledge-sharing logic.

The entire Nautilus architecture can be self-hosted. A scenario can be deployed remotely, from the web application, or locally, interacting with the framework's command-line interface. The Deployment Framework, if configured to perform this operation, provides an authenticated HTTP endpoint used to specify the scenario to be launched. In the remotely distributed scenario, the framework collects all scenario information from the database and distributes it. In the local scenario, however, scenario information can be loaded directly into the Deployment Framework and used to start the virtual environment without any connection to the Nautilus database, thus keeping the information on the test bench private.

2.2. Kypo

Kypo is an open-source cyber range accessible via a web interface that emulates, within a virtual environment, an IT infrastructure for educational purposes. It is designed to conduct exercises, simulations, and training in the field of cybersecurity. It consists of a dedicated infrastructure that allows cybersecurity experts, students, and industry professionals to practice in realistic scenarios without risking damage to real systems or networks. Like any other cyber range, Kypo offers the possibility to design the infrastructure on which to conduct an attack and to establish guidelines to follow during its execution.

The platform is also centred around some essential concepts:

Sandbox An isolated test environment containing virtual networks that allow trainees to connect to a VM and communicate with the remaining components of the network. A sandbox is described by a set of configuration files called sandbox-definition or sandbox-descriptor.

Sandbox-definition A set of configuration files that define the internal structure of the sandbox. It describes the network topology and VM configurations. The sandbox-definition is created by instructors.

Training An exercise that requires interaction between trainees and the virtualized infrastructure defined in a sandbox definition. Trainees are required to perform activities sequentially: if an activity is performed correctly, access to the next one is allowed.

Training-definition A configuration file that defines a training. It describes the actions a trainer must perform. The training-definition is entirely defined by instructors.

Pool A group of instantiated sandboxes described by the same sandbox definition.

Conceptually, the architecture of Kypo involves a series of components that make it possible to allocate a sandbox and subsequently interact between trainees and the virtual infrastructure:

Git repositories Once a sandbox definition is created, it can be uploaded to a remote Git repository. This repository, at a later time, can be imported into Kypo through a particular service of the Kypo portal to be instantiated and give life to a sandbox instance.

Kypo portal A set of services that allow users to use the cyber range. It consists of an Angular front-end and a back-end based on the microservices architectural paradigm.

OpenStack A platform that controls, monitors, and manages virtual resources that can be allocated to build a sandbox instance.

Kypo portal is the heart of the cyber range, hosting the logic to store sandbox descriptors and modify and instantiate training. Specifically, it consists of two entities: Kypo head and Kypo proxy. Kypo head hosts the server running the Kubernetes controller which in turn manages the execution of all other services each in a separate docker container. Kypo proxy is instead the server that allows access to the sandboxes only via SSH. Both servers are connected to a management network called kypo-base-network that interfaces them with the sandbox VMs.

Kypo head hosts both the backend and the frontend that allow user interaction with the cyber range. The frontend is implemented in Angular. The backend consists of a series of microservices implemented both in Java and Python executed in isolated containers and orchestrated by Kubernetes.

Some of the most important services are:

User and group service Provides functionalities to manage users and roles and register other microservices with specific responsibilities.

Training service Provides functionalities to create, manage, and launch training.

Sandbox service Provides functionalities to manage the lifecycle of a sandbox. It includes creating a sandbox definition and creating and removing an instance.

2.3. Strenght and Weekness

We identified three main strengths of Kypo: i) Kypo boasts a robust system architecture that supports multi-step training paths and facilitates the use of Capture The Flag (CTF) challenges. This architecture enables seamless integration of various training modules and ensures scalability for handling complex scenarios. ii) Kypo provides a structured environment for conducting training exercises utilizing formalism to define scenarios within a sandboxed environment. Kypo offers advanced training management features, allowing instructors to easily organize and monitor training sessions. iii) Kypo leverages Git as a distributed storage solution, providing a reliable and flexible platform for storing training materials, code repositories, and collaboration among users. This enables version control, collaboration, and easy access to training resources across distributed teams. The main strength of Nautilus is the Marketplace which allows users to access a wide range of training modules, challenges, and resources. The marketplace fosters community collaboration, allows for the sharing of best practices, and encourages continuous improvement in cybersecurity skills.

However, both have some weaknesses:

Lack of support for variable network topologies Both Nautilus and Kypo fall short in providing adequate support for scenarios involving variable network topologies. This limitation restricts the realism and applicability of training exercises in simulating diverse network environments.

Absence of support for georeferenced scenarios Neither Nautilus nor Kypo offer support for georeferenced scenarios, which limits their effectiveness in simulating real-world cybersecurity challenges that involve geographical constraints or location-specific threats. This gap hinders the development of skills relevant to cyber defence in specific geographic contexts.

Absence of student action monitoring Both platforms lack robust mechanisms for monitoring and analyzing student actions during training sessions. This deficiency may hinder trainers' ability to provide targeted guidance and feedback, as well as evaluate learners' progress accurately.

3. Reference scenarios

We decided to consider the two reference scenarios shown in Figures 1 and 2.

In Scenario 1, the satellite segment is composed of one Geostationary (GEO) satellite. Given the vast extension of the Earth's surface covered by one GEO satellite, the ground segment is composed of a high number of user devices (at least 1000) and a high number of ground stations (at least 100). Each user device accesses the network directly via satellite. The ground stations are assumed to be all interconnected with each other via a high-speed terrestrial network. Each user terminal generates/receives data destined to / coming from a Point-of-Presence (PoP) of the network and will be routed through the bidirectional path <user device-GEO satellite-ground station(s)-PoP>.

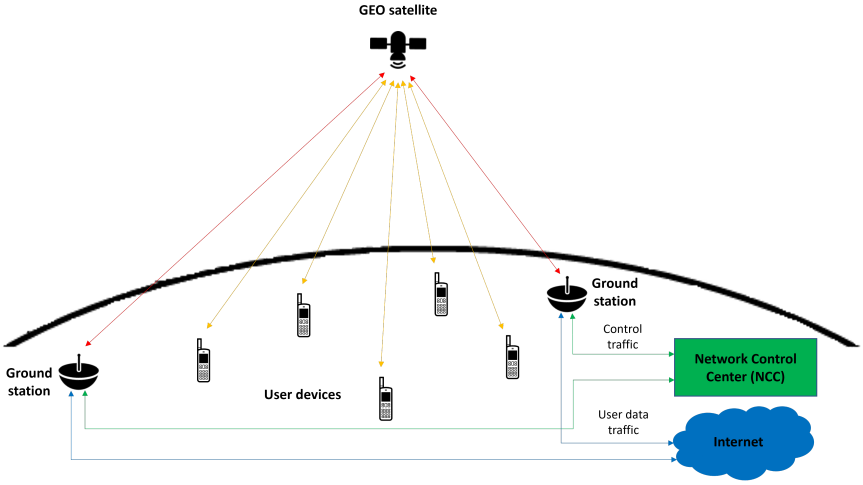

Figure 1: Scenario 1: Single GEO satellite network

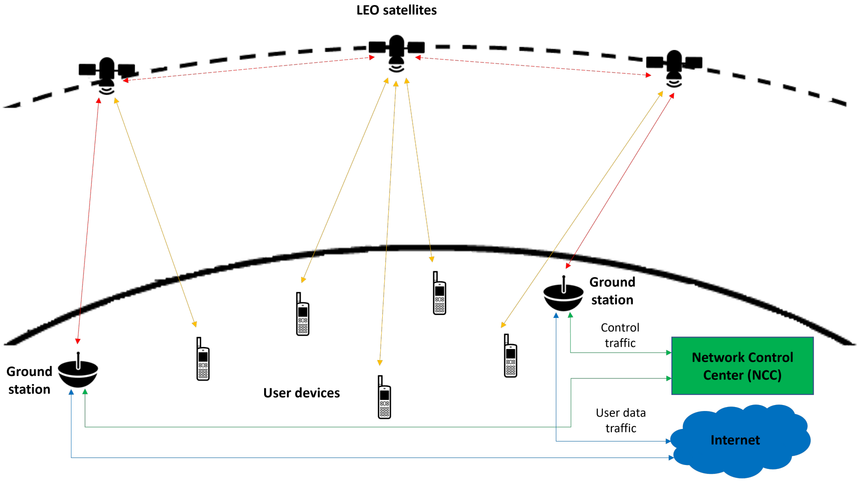

Figure 2: Scenario 2: LEO satellite constellation network

In Scenario 2, the satellite segment is composed of a Low Earth Orbit (LEO) satellite constellation. This segment is composed of a sufficiently high number of satellites (at least 100) to simulate/emulate a satellite network able to cover a significant portion (at least 80%) of the Earth's surface. The satellites are interconnected through Inter-Satellite Links (ISLs) in the classic four-neighbours "mesh" scheme.

In both scenarios, the geographical distribution (initial position and movement model) of the user terminals and the channel model of the user-satellite links can be set to simulate the following environments:

Dense urban (metropolis) A high number of user devices are located very close to each other (within an area of size $\leq 10km^2$), with fixed or variable position over time following certain movement patterns (e.g., pedestrians and vehicles in city traffic) and with a channel model including attenuation factors representing the presence of numerous tall buildings that could potentially even block the signals.

Urban (city) Conditions similar to the dense urban environment but more relaxed both in terms of user density and attenuation factors due to buildings in the channel model.

Suburban (town) Compared to the urban environment, lower user density and smaller size of the area within which they move in order to simulate more sparse user terminals and with different user mobility models (e.g., devices with vehicular movement model can move at greater speeds than in the urban environment).

Rural (small town) Compared to the suburban environment, further lower user density and smaller size of the movement area in order to simulate user terminals that are even more sparse and with different user mobility models (e.g., devices onboard ships or aircraft).

The user devices in the scenario can differ from the data application running onboard and generating data with different traffic volumes and statistical distributions. The aim is to emulate/simulate different applications with different performance requirements that generate/receive data streams with different statistics. Different user devices can also have different hardware/software configurations in order to simulate/emulate different kinds of terminals (e.g., smartphones, single sensors, or hubs that aggregate traffic flows generated by / destined to nodes not equipped with a satellite interface).

Beyond the validation tests and reference scenarios, we are designing the OSR platform to also allow users to personalise the scenarios they want to consider, allowing them to set a wide range of parameters related to the scenario's general configuration. This list includes parameters related to:

Satellites The number of satellites in the scenario and, for each satellite: (i) *Orbital parameters*: a set of orbital parameters used to specify the satellite's orbit and its initial position within the orbit; (ii) *Hardware configuration*: the amount of computational and data storage resources (CPU, RAM memory, data disk memory) available to manage data communication aspects (data reception, transmission, and processing); (iii) *Software configuration*: details about the operating system and application services running onboard.

Ground Stations The number of ground stations in the scenario and, for each station: (i)*Position*: assumed fixed; (ii) *Hardware configuration*; *Software configuration.*

User devices The number of user devices in the scenario and, for each device: (i) *Initial position*: assumed fixed or variable depending on the device's movement model; (ii) *Movement model*: specify how the device's position changes (or not) over time following a specific movement model (e.g., pedestrian, road vehicle, train, ship, and aircraft); (iii) *Hardware configuration*; (iv) *Software configuration.*

Links For each link: (i) *Central frequency*; (ii) *Bandwidth*; (iii) *Channel model*: the model of the communication channel that depends on the kind of link (user device - satellite, satellite - satellite, and satellite - ground station) and environment (dense urban, urban, suburban, and rural).

4. Exercises for testing student skills

In light of the diverse and dynamic landscape of satellite technologies, understanding potential cybersecurity challenges is of paramount importance.

The following exercises have been designed to immerse students in simulated scenarios reflective of real-world satellite communication environments.

Ground Station Attacks This exercise explores potential vulnerabilities in ground stations used for satellite communication. Participants will learn how attackers could target ground stations to compromise satellite communications or gain unauthorized access to satellite systems.

Satellite Protocol Attacks Participants will examine vulnerabilities in satellite protocols and their implications. The focus will be on direct attacks targeting protocols used for satellite-to-ground communications.

Intermittent Channel Attacks (LEO Sat) This exercise involves analyzing attacks on LEO satellites that communicate via intermittent channels. Participants will learn to exploit communication windows for hijacking or interception attacks.

Satellite network routing hacking Participants will explore vulnerabilities in routing protocols used in satellite networks. They will learn to conduct attacks to modify the path of satellite data traffic.

Sniffing satellite broadcast protocols + DoS This exercise focuses on intercepting broadcast protocols used in satellite communications, such as DVB-SAT. Participants will conduct amplified Denial-of-Service (DoS) attacks to overwhelm satellite networks.

Satellite Key Distribution Participants will analyze vulnerabilities in the distribution and management of encryption keys in satellite systems. They will learn to conduct key distribution hacking attacks.

Hacking Satellite Apps This exercise involves exploring applications running on satellites. Participants will learn to identify and exploit vulnerabilities in these apps, conducting attacks to compromise them.

Satellite network flooding on return links Participants will analyze techniques for flooding satellite networks, focusing on the return link. They will learn to conduct flooding attacks to overload satellite communication capacity.

Satellite PEP hacking This exercise explores vulnerabilities in the Performance Enhancement Protocol (PEP) used in satellite communications. Participants will learn to conduct various attacks against the PEP, such as HTTP prefetching, TCP spoofing, the use of proxies, and DoS attacks.

By engaging in these exercises, students will not only deepen their understanding of cybersecurity threats in satellite systems but also develop practical skills to mitigate risks and safeguard critical.

5. OpenSatRange framework

OSR will extend the Kypo platform by integrating new features and components to enable exercises in satellite training scenarios. The system architecture is depicted in Figure 3.

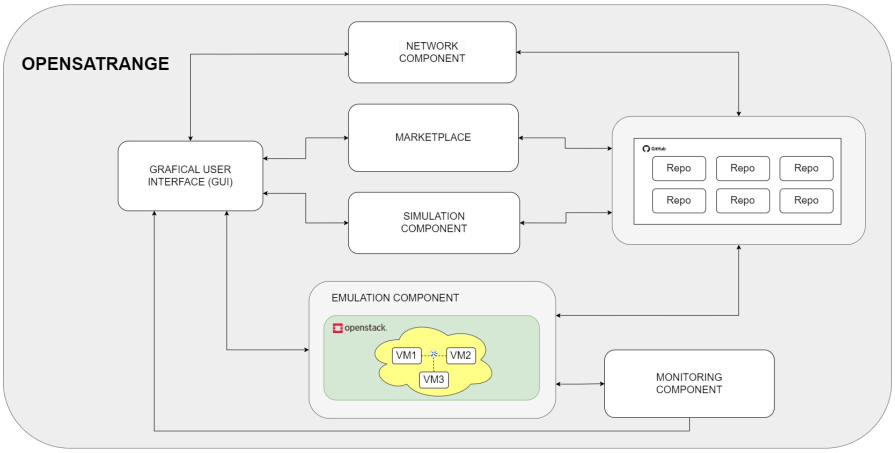

Figure 3: OSR platform architecture

The OSR platform comprises the following components:

Graphical User Interface (GUI) allows users to select already defined scenarios or define new ones.

Network Component Dedicated to create training scenarios containing the desired satellite network.

Marketplace Provides resources and materials for exercises and training, operating as a market for scenarios and courses.

Simulation Component Dedicated to simulate the desired scenario to collect and process data necessary for instantiating the scenario within the emulative environment.

Emulation Component Dedicated to create and manage the satellite scenario within an emulative environment based on OpenStack.

Monitoring Component Focuses on monitoring exercises, ensuring data collection and feedback on the performed activities.

These modules constitute the core of OSR, extending the functionalities of the Kypo cyber range within the OpenStack cloud environment. They exploit the previously introduced concepts of sandbox, sandbox-definition, training, and training-definition, while also maintaining the division of user roles into trainees, instructors, and administrators. Additionally, OSR includes a marketplace that facilitates instructors in creating new attack scenarios to simulate and offers additional functionalities for creating satellite scenarios.

5.1. Marketplace

The marketplace can be described as a platform where scripts, configuration files, sandbox definitions, and training descriptors present in various git repositories are maintained in a hierarchical and organized structure, already fully configured and ready to be instantiated. OSR not only offers the possibility to download and use such configurations but also to upload new ones. The presence of a marketplace and the ability to upload/download new configurations pave the way for three possible usage scenarios primarily involving the figure of instructors in the creation and modification of sandbox and training descriptors.

At the implementation level, the marketplace can be viewed as a single database containing other data categories:

- *Network Nodes*: configurations for terrestrial network nodes;
- *Satellite Nodes*: configurations for satellite constellations;
- *OpenSAND*: configurations of OpenSAND nodes;
- *Sandbox Descriptor*;
- *Training Descriptor.*

Each category consists of many sub-repositories, one for each node/constellation/entity to be configured.

Each repository of the network nodes category must have a well-defined structure allowing integration with a sandbox descriptor. Each repository should contain only two folders, named `files` and `tasks`, which hold all the files and scripts needed to configure a VM. In the `files` directory, any type of file can be placed regardless of its format. In the `tasks` directory, a YAML file describing the set of ansible scripts necessary to configure the VM must be placed.

The satellite nodes category will have a similar structure, formed by many repositories, one for each constellation. Each repository will consist of only the `files` folder. In particular, the `files` folder will contain text files that will be used later to emulate the constellation. These text files must describe the number of satellites in the constellation, the number and names of the terrestrial nodes of the satellite constellation, and the evolution over time of the coordinates of the satellites and terrestrial nodes of the constellation.

The OpenSAND category should be structured in the same way. It will be formed by many repositories, each of which will maintain the two `files` and `tasks` folders. Each repository represents a node of the terrestrial satellite component (Gateway or user terminal) capable of communicating with a satellite constellation. In the `files` folder, XML files necessary to activate and configure the OpenSAND satellite emulator will be kept. In the `tasks` folder, a YAML file describing the ansible scripts necessary to configure the network and thus allow communication between the emulated entities will be kept.

The Sandbox descriptor category is also an aggregate of repositories where each repository is a sandbox descriptor. A sandbox descriptor must be structured as previously described and therefore maintain the provisioning, roles, files, and tasks folders.

Finally, the Training descriptor category will be an aggregate of repositories where each repository is a training descriptor. In practice, it is a single folder containing a JSON file whose name must suggest the association with a sandbox descriptor to which that training can be associated. It is recalled that the association between an instantiated sandbox and a training descriptor occurs only at the graphical interface level. It is therefore necessary to define a method that allows to logically associate, within the marketplace, which training can be associated with which sandbox descriptor.

5.2. Simulation component

The main purpose of the simulation component is to instantiate the entire chosen scenario within a simulation environment and carry out preliminary calculations on how certain parameters will evolve over time. These calculations will contribute to the creation of data traces which will be stored and available to the emulation component. The simulation component will be based on Network Simulation 3 (NS3), an open-source software for discrete-time and discrete-event simulation of communication networks. NS3 will be used within the simulation component to create the network simulation environment, populate it with the basic network components according to the scenario chosen by the user, and calculate how the network topology and some specific communication parameters change within the considered time window. The modular approach of NS3 and its offered features are the main reasons why we decided to start the development of the simulation component starting from this software. However, given that the official release of NS3 (at least up to NS3.40, the latest version released before the preparation of this paper) does not allow simulating satellite communication networks, we have decided to proceed with the implementation of two additional modules that will implement the following two aspects:

1. *Satellite Motion Model*: NS3 includes several mobility models that aim to simulate moving nodes, such as people walking and vehicles changing their position randomly or with constant direction and speed. However, a suitable mobility model to simulate how LEO satellites change their position along defined orbital planes has not been included yet. Also considering the satellite version of NS3 (SNS3 - https://www.sns3.org/content/home.php), it focuses only on the protocol and physical layer aspects of satellite communications in GEO orbit. We decided to implement the equations of the NORAD Simplified General Perturbations 4 (SGP4) mathematical model, widely considered suitable for modelling the movements of LEO satellites with high precision, and define an ad-hoc NS3 module for this purpose. This module requires as input the initial position and orbital parameters of each satellite formatted as Two-Line Element (TLE) data and the initial position of each ground node (ground station and user devices) formatted in Latitude, Longitude and Altitude (LLA) coordinates. Both the positions of the satellites and the ground nodes are set in the Earth Centered Earth Fixed (ECEF) coordinate system, a three-dimensional Cartesian system with origin in the center of mass of the Earth, positive z axis passing through the North Pole and positive x axis passing through for the crossing point between the Equator and the Greenwich meridian. With this coordinate system, the positions of the moving nodes must be updated periodically while those of the fixed nodes (including the GEO satellites which, although moving, are fixed for the ECEF coordinate system) remain constant.
2. *Satellite link channel model*: NS3 includes several channel models in order to calculate different parameters, such as Signal-to-Noise Ratio (SNR), set other link communication

parameters accordingly, such as achievable data rate and packet loss rate, and use them to simulate different link conditions. However, a channel model for satellite links (both satellite-to-ground and inter-satellite) has not been included yet. We decided to implement an ad-hoc NS3 module for this purpose considering the channel model defined by 3GPP in the technical report TR 38.811 [9]. This channel model is valid for satellite-ground links, both LEO and GEO satellites, frequency range between 0.5 and 100 GHz, and includes the following attenuation factors:

- *Basic path loss*: basic attenuation factor made up of the three main components Free Space Loss, Shadow Fading, and Clutter Loss.
- *Atmospheric absorption*: attenuation factor due the presence of the Earth's atmosphere. The value of this attenuation depends on various parameters, such as the humidity rate, the water vapor density, the air pressure, and the transmission frequency and band.
- *Scintillation*: attenuation factor due the presence of the Earth's atmosphere. It causes rapid fluctuations in the phase and amplitude of the signal due to inhomogeneities in the structure and composition of the various layers of the atmosphere.

For inter-satellite links among LEO satellites, only the Free Space Loss will be considered as the conditions of visibility (Line-of-Sight) and absence of obstacles between a satellite and the four adjacent satellites (two on the same orbital plane, two on the two adjacent orbital planes) are always assumed to be true.

Once the scenario to be considered has been set, it is implemented within the simulation component which simulates its behaviour for the entire set duration of the simulation. Once the simulation is finished, in order to interface the simulation component with the emulation one, various configuration files will be generated. These files will be properly used by the emulation component to set the emulation environment and allocate the needed resources. These configuration files can be described as traces that indicate the following information for each time instant (for example, for each second):

- for each ground node (ground station and user terminal), which satellites are in visibility, i.e. which can be used to send and receive data, depending on the value of the corresponding elevation angle compared with a pre-established threshold.
- the propagation time and the maximum data rate for each pair of satellite-ground and satellite-satellite nodes in visibility, i.e., for each active satellite link. These values depend on the distance between pairs of nodes and the link's SNR, respectively.
- the end-to-end path that a data packet should follow to reach a specific destination starting from a specific source.

The emulation component will use these files to set the initial conditions of the emulated scenario and update them periodically.

5.3. Emulation component

The introduction of satellite emulation into OSR will be done through OpenSAND (Open Source Satellite Network Simulator), an open-source software designed to emulate and analyze satellite communication networks. It allows modelling complex networks composed of GEO satellites, ground stations and user terminals, as shown in Figure 4.

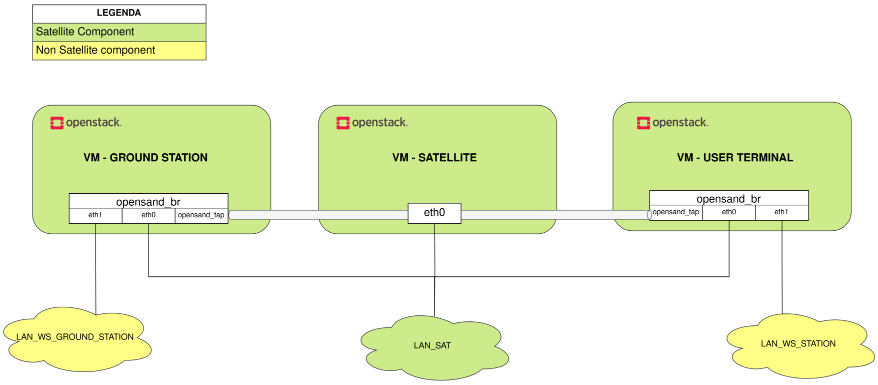

Figure 4: Use of OpenSAND within the emulation component

From an implementation viewpoint, the goal is to create a VM for each entity of the satellite component. Each VM must be configured through configuration files in XML format that describe the characteristics of the physical layer but also the activation of certain services within each entity. To allow multiple VMs to communicate with each other, the underlying network infrastructure must be properly configured.

OpenSAND enables the emulation of satellite entities based on real traffic, recreating the same protocol stack used in real communication, i.e., the DVB-S2 protocol. It also allows setting time-varying delays in communication links. This allows simulating the mobility of both satellite and user terminals. It is also possible to change the attenuation factor and communication bandwidth for each satellite-terminal or satellite-ground station link.

In particular, it is necessary for each ground station or terminal to configure three network interfaces. Considering Figure 4 as a reference, it can be seen that one interface (*eth1*) will be used to receive/send traffic from/to nodes that do not belong to the simulation component, for example these nodes could be servers that expose services, or multimedia content. Another interface (*eth0*) will be used to exchange packets in a control network for satellite emulation management, and yet another will be used to create an IP tunnel connecting the ground stations and terminals via the satellite (*opensand_tap*). These three interfaces, in each virtual machine, are then connected to a bridge. The effect of this configuration is that traffic coming in from the *eth1* interface, will be replicated to the *opensand_tap* interface through the use of the bridge. In this way, traffic traverses an IP tunnel that simulates delays in receiving and sending packets. These delays are configurable according to the scenario to be simulated.

Since OpenSAND was born to emulate only GEO satellite networks, some modifications are needed to emulate LEO satellite scenario. The features offered by OpenSAND will be combined with some precise routing indications dedicated to each VM in order to simulate the movement of the satellites and thus the change of topology of the constellation. These indications will be contained in proper configuration files that will describe, for each entity, which satellites are in visibility and how their positions evolve over time. In addition, it will have to specify what path a packet in the constellation must follow to reach its destination given a specific source. These routing rules will need to be described at regular, discrete time intervals. This configuration file will be particularly useful for simulating ISLs and the communication that occurs between satellites.

In particular, the idea is to create a virtual machine for each terminal or ground station. Instead, the satellite constellation will be modelled through container networks inside a single virtual machine. Each container will maintain within it an Opensand process that emulates a satellite. All containers simulating the satellite constellation are connected to each other via a docker network. In order to simulate the movement of the satellites and thus the change of topology of the constellation, the routing policies of incoming packets will be changes in each container. The routing policies for each node will depend directly on static configuration files created by the simulation component before instantiating the infrastructure.

In a LEO scenario, it is preferable to emulate a satellite by using a container instead of a VM for scalability reasons. Since the goal is to create constellations of at least a hundred satellites, it becomes complicated and expensive to use all the hardware needed to support a virtual machine-based emulation. To have an idea of the computational resources consumed by the implementation of a constellation, Figure 5 shows some measurements about the amount of used RAM memory related to the number of instantiated containers (up to 300).

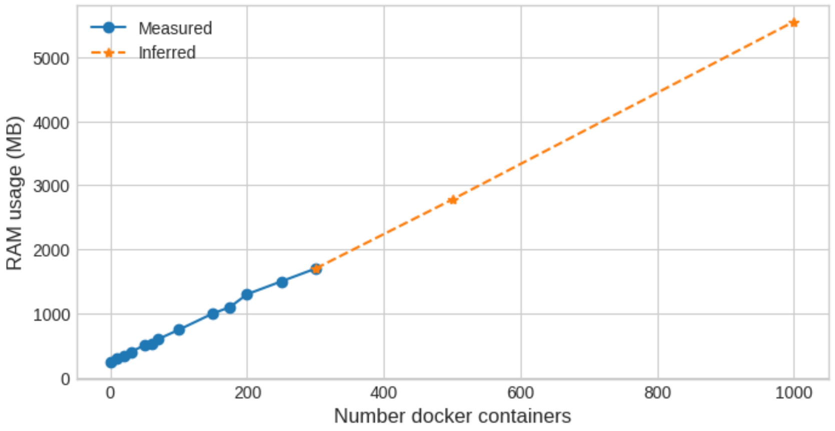

Figure 5: Required RAM memory depending on the number of instantiated containers

After observing a linear trend, subsequent measurements were inferred by regression. For example, to allocate a constellation of 1000 containers, a VM with at least 6 GB of RAM is required.

5.4. Monitoring component

The monitoring component aims to collect all useful information to monitor the exercise during its execution, process it, and provide reports that highlight the path followed to complete the assigned tasks. This component, whose structure is shown in Figure 6, will be pervasively installed in the different network instances and virtual machines installed in the sandbox where the learners perform the exercises.

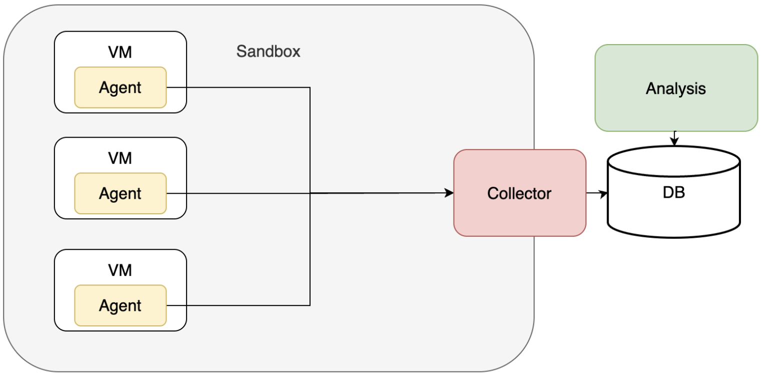

Figure 6: Structure of the monitoring component

The main modules of the monitoring component are the agents, which are distributed in the VMs and networks, and two central modules: the collector module and the analysis module.

The Collector module collects logs and events, interprets and transforms them, saving them in an external data store outside the sandbox. The analysis module acts as a full-text search and analysis engine, indexing and storing the logs from the collector. Monitoring agents are installed on virtual devices such as laptops, desktops, servers, etc., and provide detection of local actions. The collector can also be integrated with agentless devices capable of monitoring devices such as firewalls, switches, and routers.

6. Conclusion

The increasing number and impact of cyber threats and attacks is leading to an increased awareness of the cybersecurity principle and the importance of protecting the interconnected systems against these attacks. The development and use of cyber ranges to effectively train all the people involved both in the management and use of these systems is of primary importance. These tools can let the users experience cyber attacks within a controlled environment offering a practical view of the possible outcomes of these attacks and teaching possible strategies to increase the system's robustness and successfully counteract malicious activities. Satellite systems are among the scenarios to protect against cyber attacks.

In this paper, we describe OpenSatRange, a cyber range focused on satellite communication systems. It aims to increase the consciousness of satellite operators and users regarding cybersecurity, cyber threats, and cyber attacks that could affect satellite systems. Even if the development work is still ongoing, some choices have been already made in terms of the considered reference scenarios, training exercises, and framework structure. The following planned steps involve the full development and consequent integration of the system components in order to subsequently test and validate the cyber range through the planned training exercises.

Acknowledgments

This research is supported by the project “OpenSatRange: Un cyber range aperto per la formazione in cyber security di operatori di sistemi e reti satellitari” supervised and financed by the Italian Space Agency (Agenzia Spaziale Italiana, ASI) in the framework of the Research Day “Giornate della Ricerca Spaziale” initiative through the contract no. ASI-2023-2-U.0,

References

[1] W. Duo, M. Zhou, A. Abusorrah, A survey of cyber attacks on cyber physical systems: Recent advances and challenges, IEEE/CAA Journal of Automatica Sinica 9 (2022) 784–800.
[2] M. M. Yamin, B. Katt, V. Gkioulos, Cyber ranges and security testbeds: Scenarios, functions, tools and architecture, Computers & Security 88 (2020) 101636.
[3] M. Katsantonis, A. Manikas, I. Mavridis, D. Gritzalis, Cyber range design framework for cyber security education and training, International Journal of Information Security (2023) 1–23.
[4] M. M. Yamin, B. Katt, Modeling and executing cyber security exercise scenarios in cyber ranges, Computers & Security 116 (2022) 102635.
[5] D. Housen-Couriel, Cybersecurity threats to satellite communications: Towards a typology of state actor responses, Acta Astronautica 128 (2016) 409–415.
[6] P. Tedeschi, S. Sciancalepore, R. Di Pietro, Satellite-based communications security: A survey of threats, solutions, and research challenges, Computer Networks 216 (2022) 109246.
[7] G. Bernardinetti, S. Iafrate, 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, 2021, pp. 1–7.
[8] J. Vykopal, R. Ošlejšek, P. Čeleda, M. Vizvary, D. Tovarňák, Kypo cyber range: Design and use cases (2017).
[9] 3GPP, “Study on New Radio (NR) to Support Non-Terrestrial Networks”, TR 38.811 v15.0.0, 2018.