

# Investigating the Effectiveness of Zero–Trust Architecture for Satellite Cybersecurity

Masrur Masqub Utsash<sup>1</sup><sup>a</sup>, Georgios Kavallieratos<sup>1,2</sup><sup>b</sup>, Konstantinos Antonakopoulos<sup>3,4</sup> and Sokratis Katsikas<sup>1</sup><sup>c</sup>

<sup>1</sup>*Department of Information Security and Communication Technology (IIK), Norwegian University of Science and Technology, Gjøvik, Norway*

<sup>2</sup>*Department of Technology Systems, University of Oslo, 2007 Kjeller, Norway*

<sup>3</sup>*Testify SA, Oslo, Norway*

<sup>4</sup>*Bitrezus PC, Athens, Greece*

{*firstname.lastname*}@ntnu.no, {*firstname.lastname*}@its.uio.no, ka@bitrezus.com

Keywords: Satellite, Cybersecurity, Zero–Trust Architecture.

Abstract: The increasing adoption of edge computing platforms poses significant challenges to traditional perimeter-based security architectures. Zero–Trust architecture has gained traction and is now widely utilized as the preferred security architecture in critical infrastructures. However, even though the advantages of using such architecture towards improving the cybersecurity posture of satellites have been analyzed, very little has been done in demonstrating such advantages by experimenting with an implementation of the architecture. In this paper, we experimentally investigate the effectiveness of Zero–Trust architecture in improving satellite cybersecurity by analyzing two critical attacks against satellites. We describe the experimental setup and the experimentation process, and we present and discuss our findings, that demonstrate that zero-trust architecture is successful in mitigating attacks that would otherwise disrupt the operations of the satellite.


## 1 INTRODUCTION


Satellites are leveraged by several critical sectors to facilitate the provision of services such as earth observation, communication, and navigation. As of May 4, 2024, there were 3135 satellites dedicated to communication, 1052 for earth observation, and 154 for navigation, among others (Ieva, 2024). According to the satellite tracking website ‘Orbiting Now’ there are 10807 active satellites in various Earth orbits as of September 24, 2024 (Orbiting now, 2024). The emergence of New Space era, and the confluence of several challenges, such as the single point of failure systems, the highly complex supply chain, and the prolonged system life cycle, have exposed a unique vulnerability: enormous attack-surface and insufficient focus on the cybersecurity of satellites (Kavallieratos and Katsikas, 2023). The intricacies and sensitivity of satellite operations render them prime targets for malicious actors. Several attacks in space infrastruc-


ture have been reported in the past few years (Black, 2023; Singh, 2023; Corfield, 2023). These incidents underscore the urgent need to prioritize satellite cybersecurity.

In the face of such evolving cybersecurity landscape, the conventional perimeter-based defenses, often employed in other domains, proves to be inadequate for satellites (Syed et al., 2022). This motivates transcending these limitations by introducing a more robust and adaptive security framework. Zero–Trust Architecture (ZTA) has gained momentum and is increasingly emerging as the security architecture of choice (Rose et al., 2020). This shift is due to the inherent limitations of perimeter–based security models, which struggle to adapt to the decentralized and diverse network landscape of edge computing and space industry scenarios (Han et al., 2021). As the space industry increasingly relies on edge computing platforms, adopting ZTA reflects a strategic response to the evolving threat landscape, enhancing resilience against cyber threats and protecting critical assets and infrastructures.

Although several research works have explored the application of ZTA to secure satellites, its effec-

<sup>a</sup> <https://orcid.org/0009-0007-7613-4354>

<sup>b</sup> <https://orcid.org/0000-0003-1278-1943>

<sup>c</sup> <https://orcid.org/0000-0003-2966-9683>

tiveness in doing so has only partially been discussed, let alone demonstrated. The focus of this work is to experimentally assess the effectiveness of ZTA in safeguarding the cybersecurity of a satellite. Given the critical nature of such infrastructure, it is impractical to use actual, real-world systems for analyzing cybersecurity attacks, as this could disrupt regular operations. Consequently, specialists often recommend employing simulation environments that closely resemble real-world scenarios. This approach allows us to understand the potential impact of cybersecurity attacks on actual infrastructure while maintaining on-going operations.

The contributions of this work are as follows:

- Propose a ZTA solution for satellite cybersecurity;
- A laboratory environment designed to allow the implementation of and experimentation with the proposed ZTA solution towards assessing its effectiveness against selected cyberattacks;
- Demonstration of the effect that replay attacks and malicious software injection attacks have on the operational capacity of a satellite.

The structure of the remaining of this paper is as follows; Section 2 discusses related work. Section 3 presents our proposal of a ZTA for satellites and Section 4 describes the experimental setup, process and the effectiveness of the proposed ZTA in satellites. Section 5 summarizes our conclusions.

## 2 RELATED WORK

There are several approaches in the literature focusing on the cybersecurity of satellites. Zatti (Zatti, 2020) analyzed unauthorized operation of satellites, launchers, or ground satellite facilities. Amin et al. (Amin et al., 2016) identified two main threats to satellite communications, namely jamming interference and spoofing attacks. Fowler (Fowler, 2016) discussed several threats to satellite systems such as unauthorized operations, eavesdropping, jamming, hijacking, and control. Hasan et al. (Hasan and Hasan, 2022) presented a systematic threat model and security analysis of a satellite system using the STRIDE technique. Falco (Falco, 2020) delves into cyber attacks originating from one satellite and targeting another. Breda et al. (Breda et al., 2022) focused on the use of artificial intelligence (AI) in space operations, focusing on vulnerabilities such as lack of transparency, vulnerability in input data, inherent weaknesses in mathematical procedures, and the potential for exploitation by adversaries. Unal (Unal, 2019) discussed replay attacks

in space infrastructure considering NATO's space-based strategic assets. Falco (Falco, 2018) provided a comprehensive framework for enhancing security measures in space infrastructure. NASA (Office of Inspector General, 2021) leveraged the NIST cybersecurity framework to define security controls for their operations. Baker et al. (Baker et al., 2019) discussed the security benefits of software-defined networking (SDN) in space communication networks, focusing on its elastic networking, resource distribution, and network configurability.

The adoption of Zero-Trust security systems in the space industry is also gaining momentum (Quiquet, 2023) (White and White, 2024). Thangavel et al. (Thangavel et al., 2022) highlighted the advantages of ZTA in space cybersecurity. Protik (Protik, 2023) proposed an updated standard for secure satellite communications and space data systems based on Zero-Trust. Khamvilai et al. (Khamvilai and Pakmehr, 2023) explored the application of Zero-Trust across various components of complex cyber-physical systems. Schalk et al. (Schalk and Brown, 2023) highlighted the advantages of implementing a ZTA in space network software bus architectures. Fu et al. (Fu et al., 2022) introduced an architecture and scheme for continuous authentication in satellite networks. Lowdermilk et al. (Lowdermilk and Sethumadhavan, 2021) discussed the importance of minimizing trust in satellite processors. Driouch et al. (Driouch et al., 2023) discussed the challenges and risks of securing new space missions. Gvozdev et al. (Gvozdev and Vorobev, 2020) highlighted the advantages of a Zero-Trust Network in a multi-satellite system for Earth remote sensing. Based on the above findings, a research gap on the implementation of a ZTA solution in a satellite system, towards exploring its effectiveness in improving the security posture of the satellite is identified. This paper aspires to contribute towards partially bridging this gap.

## 3 A PROPOSED ZTA FOR SATELLITES

The core features of Zero Trust Architecture (ZTA), as defined by NIST (Rose et al., 2020), include strict identity verification, least privilege access, micro-segmentation<sup>1</sup>, continuous monitoring, endpoint security, and automated threat detection. This work utilizes the OpenZiti (Ziti) framework, an open-source project dedicated to integrating Zero-Trust networking principles into any application seamlessly (Open

<sup>1</sup>resources are segmented into smaller, protected zones

Source Zero Trust Networking, 2023). It offers a comprehensive toolkit for implementing a Zero-Trust overlay network<sup>2</sup>, including controllers, edge routers<sup>3</sup>, and fabric routers<sup>4</sup>. Additionally, Ziti provides various software development kits (SDKs) to facilitate the integration of Zero-Trust directly into applications. It also offers tunneling applications for granting Zero-Trust access to applications where direct integration is not feasible. As per NIST guidelines (Rose et al., 2020), the components within a ZTA are not always distinct systems. Given the distinct characteristics of the space system and the objectives of this work, we have chosen to adopt a hybrid approach that combines elements from both the Device Agent/Gateway Model and the Resource Portal Model. In this setup, the PEP functions as a singular gateway through which all subject requests pass, while our assets utilize a device agent to establish connections with the gateways. This hybrid approach to ZTA offers a secure method for implementing Zero-Trust in our experimental setup, described in the next section.

## 4 EXPERIMENTAL SETUP

In the experimental setup (Figure 1), the components in green represent the elements and entities within the Zero-Trust environment, while the components in red represent the outsider entities. This facilitates the clarification of the boundaries of the ZTA and highlights the different roles and interactions of each component within the experimental setup.

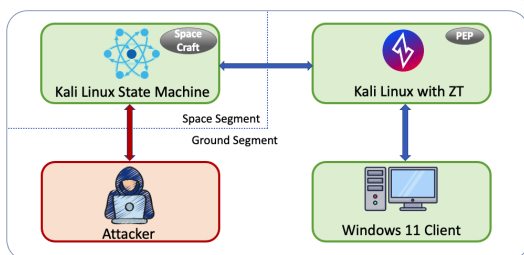


Figure 1: Experimental Setup.

We established a Zero Trust design setup with multiple computers on the same network. A Kali Linux computer represents the satellite and runs the state machine. This machine is the primary target for

<sup>2</sup>ensures the characteristics of zero-trust for every component no matter where they are on the network

<sup>3</sup>secures traffic between internal and external systems by verifying identities and enforcing access controls

<sup>4</sup>securely manages traffic between devices by ensuring all connections are authenticated and authorized

both the attacker and the security mechanisms. Another Kali Linux computer is used to deploy our Zero-Trust solution enabling the Policy Enforcement Point (PEP) and acting as the gateway through which all connections to the satellite must pass, as part of the ZTA. This machine also houses the policy engine and the policy administrator, allowing it to control all network traffic as a single gateway point. At the bottom right of the figure, a Windows 11 client is used by the authorized user as a mission operation terminal. The client is equipped with all necessary agents and certificates to securely access the satellite within the Zero-Trust network. A Kali Linux machine is utilized as the attacker due to its extensive pre-installed penetration testing tools. This simplified, yet comprehensive, setup allows us to simulate a space mission operation center where both legitimate and malicious activities are executed. This provides a robust environment to test the effectiveness of our Zero-Trust design.

### 4.1 The Zero-Trust Network

Establishing a connection to the satellite is possible either directly or via a ground station. Accordingly, our experimental setup involves two distinct scenarios. In the first scenario, we anticipate direct communication between the clients and the satellite. The second scenario assumes communication between the clients and the satellite through the ground station. In this section, we outline the network models for both scenarios. The configuration of the network is achieved by leveraging several computer devices running different operating systems to create an environment closely resembling real-world conditions.

Figure 2 illustrates the network architecture for communication with the satellite via the Ziti overlay. In this setup, a Windows 11 client establishes a connection to the Ziti Controller through the Ziti Desktop Edge using a tunnel. Simultaneously, the satellite environment, simulated on a Kali Linux machine, is connected to the Ziti Controller through another tunnel. This configuration enables direct communication between the client and the satellite environment facilitated by the Ziti overlay network. The internal communication between these components utilizes the standard TCP protocol.

Figure 3 depicts the network architecture where clients must communicate with the satellite through the ground station. In this setup, the Ziti Controller is deployed at the ground station, facilitating the connection between the clients and the satellite. The satellite is linked to the ground station through a dedicated edge router connection (OpenZiti virtual

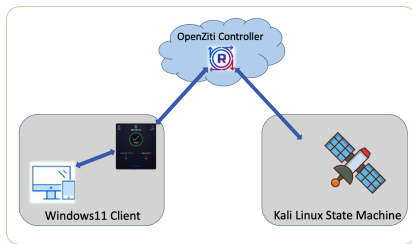


Figure 2: Client-satellite communication network.

router) assuming that the spacecraft has the capability to switch between authorized ground stations. Additionally, a public edge router is established to allow authorized entities to access the ground station and communicate with the satellite. This configuration ensures secure and controlled communication between clients and the satellite via the Ziti overlay network through the ground station. The edge routers are equipped with Ziti capabilities and are registered as distinct router entities via the Ziti console. They enable connections with the Ziti controller at the ground station via a tunneler, thereby creating a secure communication channel for all established connections. In a real-world operational environment with real satellites running in orbit and having ground operations, the CCSDS (Consultative Committee for Space Data Systems) Space Packet Protocol <sup>5</sup> would be used. Since for this work we use an IP-network of emulated satellites, communications utilize the standard TCP protocol.

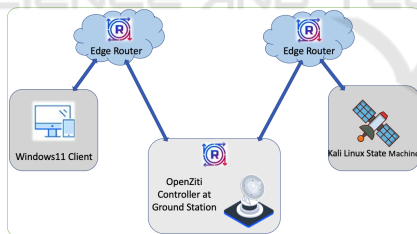


Figure 3: Client-satellite communication network through ground station.

## 4.2 Deploying Openziti

The flexibility of the OpenZiti solution has significantly facilitated our experiment by providing all the necessary options to create and maintain the components of our chosen ZTA. For this work, a local environment is using Kali Linux with 8 vCPU cores and 16GB of RAM.

<sup>5</sup><https://public.ccsds.org/Pubs/133x0b2e1.pdf>

## 4.3 Defining the State Machine

A state machine can effectively portray the activities of a satellite. In our state machine, we have outlined several states to signify both the operational health status of the satellite and its mission-related activities. The state machine initiates at an initial mode and concludes with the de-orbit mode. Each state represents a specific condition or stage in the satellite’s operation, and transitions between states occur based on predefined events or conditions. The state machine changes its state upon receiving valid event commands, otherwise the satellite disregards the command.

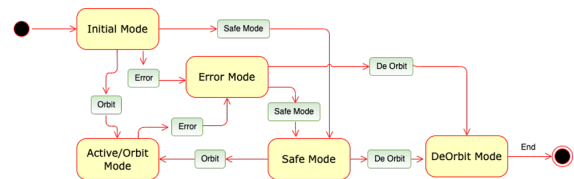


Figure 4: State machine for satellite.

Concerning Fig. 4, we start at the Initial Mode state, the starting point for our system’s journey. As the system switches to the Active/Orbit Mode, representing regular operational state, like when a satellite is smoothly orbiting Earth, performing its tasks. However, if something goes wrong, like a glitch or malfunction, the system enters Error Mode, similar to when any device malfunctions or displays an error message. In such cases, the system shifts to Safe Mode, a protective fallback, shutting down non-essential functions to maintain stability. For our satellite, this might involve conserving power or reducing activity. Eventually, when the mission concludes or a critical problem arises, the system moves into DeOrbit Mode, facilitating a graceful exit strategy, like, for example, concluding the orbit and returning the satellite safely to Earth. In this setup, we assume that the attacker will primarily target the active/orbit mode, as this represents the regular state of a satellite.

## 4.4 Attacks

The Space Attack Research and Tactic Analysis (SPARTA) <sup>6</sup> framework provides information on Tactics, Techniques, and Procedures (TTP) that may be used to compromise spacecraft. In this work, we conducted further analysis to prioritize risks by employing a simplified 4x4 risk matrix (Figure 5), compared to the more detailed approach used by Bailey in the SPARTA framework<sup>7</sup>. Further threat and risk descrip-

<sup>6</sup><https://sparta.aerospace.org/>

<sup>7</sup><https://aerospace.org/paper/cybersecurity-protections-spacecraft-threat-based-approach>



tions for the risk matrix (Figure 5) can be found in (Bailey, 2022).

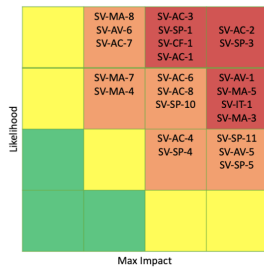


Figure 5: 4x4 risk matrix for ranking cyber threats.

In our analysis, we assigned to each threat a likelihood of occurrence score ranging from 1 to 10. Additionally, we used an impact score, also ranging from 1 to 10. The likelihood score assessed the probability of the threat occurring based on factors like historical trends, system vulnerabilities, and the resources needed for the attack. The impact score measured the potential damage the attack could cause, including operational disruption, financial losses, data breaches, and overall disruption of space mission. The accordant risk value is estimated based on the formula:  $Risk\ Score = Likelihood\ score \times Impact\ score$ . This work focuses on the most critical attacks as these are identified in the aforementioned analysis. These are; (1) Replay of recorded authentic communications traffic at a later time with the hope that the authorized communications will provide data or some other system reaction, and (2) Introduction of malicious software such as a virus, worm, Distributed Denial-Of-Service (DDOS) agent, rootkit, or Trojan Horse.

#### 4.4.1 Replay Attack

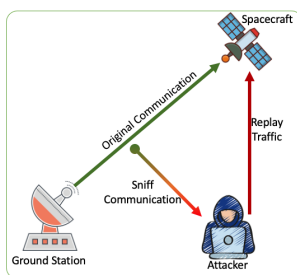


Figure 6: Attack Model: Replay Attack.

We assume a scenario where an attacker intercepts a genuine command intended to alter a state within the satellite, as depicted in Figure 6. In this experiment, we are using tools like Wireshark<sup>8</sup> and Ettercap<sup>9</sup> for

<sup>8</sup>Wireshark: <https://www.wireshark.org/>

<sup>9</sup>Ettercap: <https://www.ettercap-project.org/>

sniffing and capturing original communication packets. For sending the packet back to the satellite we are using *tcpreplay*<sup>10</sup>. To simulate a replay attack on a satellite system, we followed a structured approach as outlined in Figure 7.

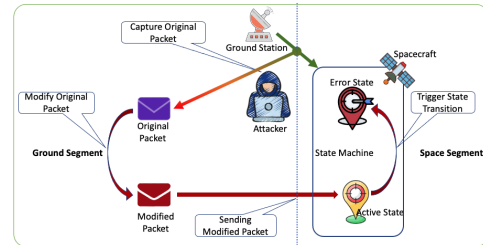


Figure 7: Replay attack simulation steps.

Initially, we assume that the Zero-Trust approach is not yet implemented and the attack is simulated by trying to capture communication between the satellite (Kali Linux state machine) and the windows client computer. In the controlled environment, the infrastructure for implementing the attack involved setting up various software tools on our testing machines. In a real-world scenario, this preparation phase would also include procuring the necessary hardware components, such as antennas and other communication devices, to target the satellite.

Next, a specific satellite is targeted for the attack, represented by a state machine running on a Kali Linux computer. The attacker gathers information to target the satellite, scanning the network to identify connected components. Using Ettercap, we launch a man-in-the-middle (MITM) attack, placing the attacker’s computer between the satellite and the client to capture traffic in real-time. Wireshark is used to intercept and store data packets during communication. We then modify the captured packets with a Python script, changing any ‘event’ variable to ‘error’ and saving the altered packet. The modified packet is sent back to the satellite using the *rcpreplay* command. In this controlled environment, we capture packets for 10 seconds, ensuring at least one event command is intercepted and modified before being resent to the satellite.

After executing the replay attack, we observed the following results. Initially, the state machine was operating in the active/orbit mode. However, upon launching the replay attack, the state machine transitioned to the error mode. This experiment demonstrates that such an attack can effectively compromise the satellite’s operational state, leading to mission disruptions. The response to a replay attack can vary depending on the sent command and the current state of

<sup>10</sup>Tcpreplay: <https://github.com/appneta/tcpreplay>

our state machine representing the satellite. Following the successful execution of the attack in a non-ZTA environment, we proceeded to test the same attack within a Zero-Trust environment. The simulation steps are depicted in Figure 8.

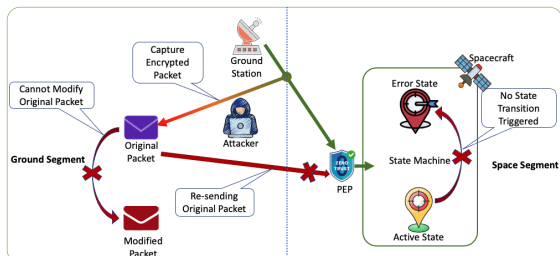


Figure 8: Replay attack simulation steps with Zero-Trust implemented.

The initial steps - preparing the infrastructure, targeting the satellite, setting up sniffing tools, and capturing packets - were conducted as before. However, due to the data encryption inherent in the ZTA, the captured packets could not be analyzed or modified. Despite this, we attempted to resend the original captured packet to the state machine, to determine whether it would cause some disruption within the satellite system. Once again, the ZTA proved its effectiveness. As the external attacker lacked a specific agent necessary for communication within the Zero-Trust environment, the packet could not be delivered to the state machine. During the attempt to send the modified packet, the PEP tried to verify the identity of the attacking computer and could not find any valid certificate; as a result the PEP disregarded the connection request. As an attacker, we also tried to bypass this issue by capturing the communication packet with the certificate. While trying to replay the communication of the authentication command, the PEP tried to validate the certificate and could not match the entity with the already enlisted entities within the Zero-Trust network, eventually disregarding the connection request. Consequently, no changes occurred in the state transition, and the state machine remained in its original state. This result demonstrates the robustness of ZTA in preventing unauthorized access and maintaining the integrity of the satellite’s operational state.

#### 4.4.2 Malicious Software Injection Attack

We assume a scenario where an attacker sends a malicious script to the satellite (Fig. 9). Our primary objective was to explore the feasibility and relevance of employing a malicious software injection attack against a satellite. We aimed to identify an approach

that would not only be impactful but also pertinent to the unique challenges of securing space systems.

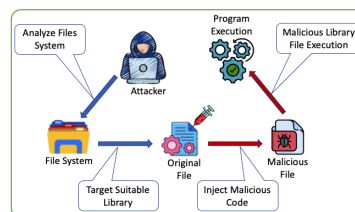


Figure 9: Attack Model: Malicious Software Injection Attack.

By focusing on this specific type of attack, we sought to gain insights into both its potential impact and its practicality within the satellite landscape. We used an online resource named *fakelib.sh*<sup>11</sup>. Fake-lib.sh (Blázquez, 2021) is a streamlined tool designed for generating Linux shared libraries, particularly for use in library hijacking scenarios. It enables the creation of empty libraries with embedded payloads, which can be executed using LD\_PRELOAD. Though fakelib.sh was originally developed for research and practice purposes for strengthening the security posture, it could be used in real-life by the attacker to pursue malicious objectives. This tool operates by analyzing a shared library binary, extracting its functions and symbols, and subsequently generating another library that closely resembles the original one. However, this fabricated library contains a payload specified by the user. For employing the fakelib.sh tool, we initially analyze the library files using *readelf*<sup>12</sup>. Readelf is a versatile command-line tool used for displaying detailed information about ELF (Executable and Linkable Format) files, which are commonly used for executables, object code, shared libraries, and core dumps in Unix-based systems. To simulate this attack, we followed the procedure outlined in Figure 10.

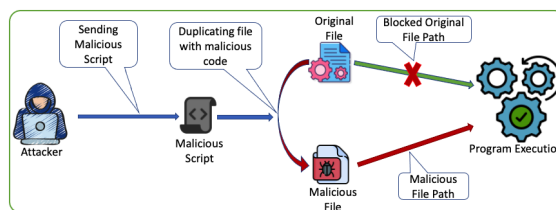


Figure 10: Malicious Software Injection attack simulation steps.

Initially, from the attacker’s machine, we sent the fakelib.sh script to the Kali Linux computer representing the satellite. We then analyzed the library files to identify the most suitable target for the attack.

<sup>11</sup>fakelib: <https://github.com/eblazquez/fakelib.sh>

<sup>12</sup>readELF: [uxhint.com/readelf-linux-command/](http://uxhint.com/readelf-linux-command/)

After selecting the library file, we analyzed the executable in order to identify its functionalities and extract function names, utilizing the *readelf* command. *Fakelib.sh* offers attackers the capability to designate a specific function as the injection point, where the payload code will be inserted. The attacker retains the flexibility to select an appropriate function for injecting their malicious code. The *fakelib.sh* script offers multiple execution options, including a basic *echo* command, running code in a shell environment, or executing custom shellcode. For our experiment, we used the *echo* command to demonstrate the potential for injecting and executing malicious code. We selected a target library file and executed the *fakelib.sh* script to duplicate the original library, embedding the malicious code. The script allows for either overwriting an existing library or creating a duplicate in a different directory; we chose the latter for greater control and evasion of detection. We blocked the original file path and introduced the path to the malicious file, ensuring its use during program execution. The attack was confirmed to be successful by the execution of the injected *echo* command, validating its effectiveness in compromising the satellite system.

In a separate test scenario, the injection of malicious code resulted in the corruption of the executable file of our state machine. This outcome underscores the potential real-world risk wherein an attacker could disable a satellite. After successfully executing the attack, we proceeded to evaluate the effectiveness of Zero-Trust in securing the satellite by simulating the same attack scenario within a Zero-Trust environment (Figure 11).

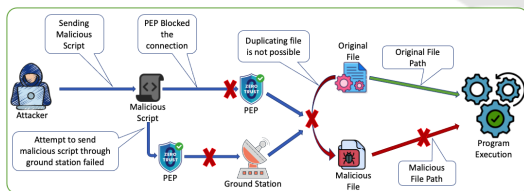


Figure 11: Malicious Software Injection attack simulation with Zero-Trust implemented.

In attempting to execute the same attack within the Zero-Trust environment, we encountered a significant barrier at the very first step i.e., when sending the *fakelib.sh* script to the satellite environment. Since the attacker's computer is an external entity within the Zero-Trust environment and lacks authorization with a specific agent to communicate with the environment, the script could not be delivered to the satellite. During the attempt to send the modified packet, the PEP tried to verify the identity of the attacker computer and could not find any valid certificate; as

a result the PEP disregarded the connection request. To further test the robustness of the Zero-Trust solution, we attempted to bypass this restriction by assuming that the script could be sent to the satellite through an exploited connection via the ground station. In this scenario the ground station also refused the connection of the attacker as the ground station is also secured within the Zero-Trust environment. However, due to the micro-segmentation inherent in the Zero-Trust environment, it was impossible to analyze the existing file system, thus preventing the execution of *fakelib.sh* as the target library could not be defined. Consequently, no modifications were made to the original library file, and no malicious file was created. As a result, when the program was executed, it utilized the original library file, rendering the attack unsuccessful.

## 5 CONCLUSIONS

The increased reliance of everyday, often critical, services on satellites brings both enormous benefits, at the cost of serious cybersecurity risks. In this work we explored how ZTA can be effectively implemented to enhance cybersecurity in satellites. We proposed a ZTA-based solution and developed a controlled environment using multiple computers on the same network to simulate a satellite system in order to experimentally assess its effectiveness. We experimentally demonstrated that the proposed solution constitutes a robust framework for addressing security challenges and effectively mitigates two types of critical attacks. Thus, our approach to ZTA has promising potential to secure satellites. In future work, we will explore the implementation of Zero-Trust principles in satellite constellations to ensure comprehensive security across the entire network and all connected satellites.

## REFERENCES

- Amin, M. G., Closas, P., Broumandan, A., and Volakis, J. L. (2016). Vulnerabilities, threats, and authentication in satellite-based navigation systems [scanning the issue]. *Proceedings of the IEEE*, 104(6):1169–1173.
- Bailey, B. (2022). *Cybersecurity Protections for spacecraft: A Threat Based approach — The Aerospace Corporation*.
- Baker, D. Z., Liu, H., and Roberts, C. (2019). Ensuring flexibility and security in sdn-based spacecraft communication networks through risk assessment. In *2019 IEEE International Symposium on Technologies for Homeland Security (HST)*, pages 1–6. IEEE.
- Black, D. (2023). Japan's JAXA space agency

- admits cyberattack. *Cybernews*. url: <https://cybernews.com/news/japan-jaxa-space-agency-cyberattack/>, Accessed: 24-09-2024.
- Blázquez, E. (2021). GitHub - fakelib.sh: Simple tool/script for generating malicious Linux shared libraries. url: <https://github.com/eblazquez/fakelib.sh>, Accessed 24.09.2024.
- Breda, P., Markova, R., Abdin, A., Jha, D., Carlo, A., and Manti, N. P. (2022). Cyber vulnerabilities and risks of ai technologies in space applications. In *73rd International Astronautical Congress (IAC), Paris, France*.
- Corfield, G. (2023). Russian spy agencies targeting Starlink with custom malware, Ukraine warns. *The Telegraph*. url: <https://www.telegraph.co.uk/business/2023/08/12/russian-spy-agencies-targeting-elon-musk-starlink-malware/>, Accessed: 24-09-2024.
- Driouch, O., Bah, S., and Guennoun, Z. (2023). A holistic approach to build a defensible cybersecurity architecture for new space missions. *New Space*, 11(4):203–218.
- Falco, G. (2018). The vacuum of space cyber security. In *2018 AIAA SPACE and Astronautics Forum and Exposition*, page 5275.
- Falco, G. (2020). When satellites attack: Satellite-to-satellite cyber attack, defense and resilience. In *AS-CEND 2020*, page 4014.
- Fowler, B. W. (2016). *Cyber vulnerabilities in space systems*. PhD thesis, Utica College.
- Fu, P., Wu, J., Lin, X., and Shen, A. (2022). Ztei: Zero-trust and edge intelligence empowered continuous authentication for satellite networks. In *GLOBECOM 2022-2022 IEEE Global Communications Conference*, pages 2376–2381. IEEE.
- Gvozdev, O. and Vorobev, V. (2020). The concept of information infrastructure of a multi-satellite system for earth remote sensing. *International Multidisciplinary Scientific GeoConference: SGEM*, 20(2.2):275–283.
- Han, R., Bai, L., Jiang, C., Liu, J., and Choi, J. (2021). A secure architecture of relay-aided space information networks. *IEEE Network*, 35(4):88–94.
- Hasan, R. and Hasan, R. (2022). Towards a threat model and security analysis of spacecraft computing systems. In *2022 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*, pages 87–92. IEEE.
- Ieva (2024). How Many Satellites are in Space? *Kongsberg nanoavionics*. url: <https://nanoavionics.com/blog/how-many-satellites-are-in-space/>, Accessed: 24-09-2024.
- Kavallieratos, G. and Katsikas, S. (2023). An exploratory analysis of the last frontier: A systematic literature review of cybersecurity in space. *International Journal of Critical Infrastructure Protection*, page 100640.
- Khamvilai, T. and Pakmehr, M. (2023). Zero trust avionics systems (zta). In *2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC)*, pages 1–8. IEEE.
- Lowdermilk, J. and Sethumadhavan, S. (2021). Towards zero trust: An experience report. In *2021 IEEE Secure Development Conference (SecDev)*, pages 79–85. IEEE.
- Office of Inspector General (2021). NASA's Cybersecurity Readiness. Report IG-21-019 (A-20-009-00), National Aeronautics and Space Administration. url: <https://oig.nasa.gov/docs/IG-21-019.pdf>, Accessed 24.09.2024.
- Open Source Zero Trust Networking (2023). OpenZiti - Open Source Zero trust Networking. url: <https://openziti.io/>, Accessed 24.09.2024.
- Orbiting now (2024). Active Satellite orbit data. *Orbiting now*. url: <https://orbit.ing-now.com/>, Accessed: 24-09-2024.
- Protik, R. C. (2023). Updated standard for secure satellite communications: Analysis of satellites, attack vectors, existing standards, and enterprise and security architectures. *arXiv preprint arXiv:2310.19105*.
- Quiquet, F. (2023). Reaching for the Stars with Zero Trust: Space Domain Applications.
- Rose, S., Borchert, O., Mitchell, S., and Connelly, S. (2020). Zero trust architecture. NIST Special Publication 800-207, National Institute of Science and Technology.
- Schalk, A. and Brown, D. (2023). Detection and mitigation of vulnerabilities in space network software bus architectures. In *2023 IEEE Aerospace Conference*, pages 1–10. IEEE.
- Singh, K. J. (2023). The first satellite hacking by Russian hackers to Ukraine. *LinkedIn*. url: <https://www.linkedin.com/pulse/first-satellite-hacking-russian-hackers-ukraine-kamal-jeet-singh-jjdic/>, Accessed: 24-09-2024.
- Syed, N. F., Shah, S. W., Shaghghi, A., Anwar, A., Baig, Z., and Doss, R. (2022). Zero trust architecture (zta): A comprehensive survey. *IEEE Access*, 10:57143–57179.
- Thangavel, K., Plotnek, J. J., Gardi, A., and Sabatini, R. (2022). Understanding and investigating adversary threats and countermeasures in the context of space cybersecurity. In *2022 IEEE/AIAA 41st Digital Avionics Systems Conference (DASC)*, pages 1–10. IEEE.
- Unal, B. (2019). *Cybersecurity of NATO's Space-based Strategic Assets*. Chatham House. The Royal Institute of International Affairs.
- White, E. and White, E. (2024). A zero-trust approach to space cybersecurity could be the answer.
- Zatti, S. (2020). Space and cyber threats. *Handbook of Space Security: Policies, Applications and Programs*, pages 245–263.