# Eliminating Software Defined Radio Dependency in Satellite Security Research: A Software Based Attack Model

1st Kelli Yuvraj
*Department of Computer Science*
*Silicon University*
Bhubaneswar, India
ce.23bceh56@silicon.ac.in

2nd Abhay Kumar Das
*Department of Computer Science*
*Silicon University*
Bhubaneswar, India
cen.23bcei06@silicon.ac.in

***Abstract*—The proliferation of open-source software in satellite operations has introduced significant security vulnerabilities, exposing critical control systems to sophisticated cyber threats. Replay attacks on satellite command protocols have traditionally relied on Software-Defined Radios (SDRs) for signal interception and retransmission. This research represents a transformative change in satellite cybersecurity by showing that replay attacks, which formerly depended on SDRs, can now be performed entirely using software-based techniques. Utilizing Python and Scapy, we intercept and alter command packets within OpenSatKit(OSK), a popular open-source satellite control platform, proving that unauthorized command execution is possible without any physical hardware. This discovery creates a new attack surface that questions traditional security beliefs in space systems, highlighting the pressing need for strong software-based defenses. Experimental results confirm that OSK lacks robust command authentication, allowing replayed commands to be executed without detection, reinforcing the need for cryptographic validation in open-source satellite frameworks. By removing the hardware requirement for satellite penetration testing, this research democratizes cybersecurity studies within the space sector while revealing significant vulnerabilities that need urgent attention. This study establishes a new benchmark for assessing the resilience of spaceborne systems against software-driven attacks, representing a major advancement in the analysis of satellite security.**

***Index Terms*—OpenSatKit (OSK), core Flight System (cFS), replay attacks, software-defined radio (SDR), software-based penetration testing, satellite command authentication, open-source satellite security, spaceborne cybersecurity vulnerabilities**

## I. Introduction

The cost of accessing space has progressively decreased due to advancements in small satellite (smallsat) design and the increasing availability of ride-share launch opportunities. The cost of designing and manufacturing smallsats has decreased significantly. Furthermore, essential satellite components are now commercially available, reducing both development time and costs. This shift has enabled researchers, educators, commercial entities, and even hobbyists to develop and deploy smallsat constellations at a fraction of the cost of traditional large satellites [1].

The NASA Goddard Space Flight Center (GSFC) released its core Flight System (cFS) as open-source software in 2015. The cFS is a highly mature and very reliable Flight Software (FSW) framework that is currently utilized on a number of operational NASA Class B missions [2].

OpenSatKit (OSK) serves as an open-source, freely available framework designed to streamline the integration and deployment of the cFS. This framework provides eight preconfigured cFS applications, along with a modular architecture for developing and integrating mission-specific functionalities. Having a preconfigured flight-to-ground interface significantly accelerates the development lifecycle for Flight Software (FSW) engineers by reducing initial configuration overhead. Developers can concentrate on adapting the kit's cFS components to their requirements, implementing new mission-specific applications, porting the cFS to the target platform, and verifying system functionality and performance.

Thus, due to the nature of unauthorized command execution, causing malfunctions in the system and therefore mission failure, replay attacks pose a severe risk in satellite operations.

A critical threat to satellite control systems is replay attack, in which an attacker intercepts, records, and sends back real command packets to change how the system works. Thus, due to the nature of unauthorized command execution, causing malfunctions in the system and therefore mission failure, replay attacks pose a severe risk in satellite operations. Without robust authentication mechanisms, systems like OSK remain susceptible to exploitation due to inadequate command validation, allowing adversaries to inject unauthorized instructions.

An SDR consists of a radio frequency front-end integrated with an analog-to-digital (ADC) and digital-to-analog (DAC) converter within a dedicated hardware module. GNU Radio Companion (GRC) leverages modular signal processing blocks to implement and simulate various radio frequency (RF) communication algorithms. These elements offer demodulation and decoding of the signals that are sent by the satellite [3].

Traditionally, Software-Defined Radio (SDR) has been an

incredible tool for testing and performing replay attacks on satellite communication systems. SDRs allow security researchers and adversaries to capture signals in the radio frequency domain and retransmit them, thereby simulating real-world cyber threats against satellite command and telemetry links. Prior research extensively documents SDR-based replay attacks targeting Global Navigation Satellite Systems (GNSS) and satellite telemetry frameworks [4]. The use of Software-Defined Radios (SDRs) to compromise control links of space-based systems further demonstrates their effectiveness in testing security vulnerabilities of such systems [5]. However, reliance on SDRs for executing such attacks poses several challenges, including high costs, regulatory constraints, and the requirement for advanced technical expertise.

In this study, we introduce a novel approach to executing replay attacks without requiring SDRs, demonstrating that satellite control vulnerabilities can be exploited entirely through software-based methods. Our methodology involves the use of Scapy, Python, and network packet analysis tools, eliminating the dependency on specialized radio hardware. This SDR-free attack model allows for efficient, cost-effective, and legally compliant security testing, broadening accessibility for cybersecurity researchers and space agencies.

Our research makes the following contributions:

- The feasibility of replay attacks on OSK has been demonstrated without SDR capabilities, confirming that command injection vulnerabilities can be exploited solely in a software-based environment. Therefore, our work has essentially proven that SDR-free replay attacks are viable.
- Through extensive testing we have exposed some serious vulnerabilities in the OSK command authentication mechanisms, highlighting the urgent need for enhanced security for open-source satellite systems.
- By eliminating SDR dependency, the proposed methodology enables more accessible and legally compliant cyber-resilience assessments of satellite control frameworks

As open-source software is increasingly adopted in space systems of critical importance, it becomes imperative to proactively assess and mitigate all security risks before real-world exploitation can occur. The results highlight the pressing need for enhanced authentication mechanisms in OSK or similar platforms so that unauthorized command execution and system manipulation would be prevented. This research lays the foundation for further work to be done towards developing more secure software-driven testing environments for satellite cybersecurity to ensure mission integrity in a rapidly evolving arena of cyber threats.

## II. Related Work

### A. Overview of Replay Attacks in Network Security and IoT

Replay attacks have posed serious concerns for decades in various security domains, including network security, industrial control systems (ICS), and the Internet of Things (IoT) in general. These attacks exploit the lack of robust authentication mechanisms by intercepting and retransmitting legitimate data packets to manipulate system behavior. In IoT systems, replay attacks are particularly dangerous due to the widespread adoption of lightweight communication protocols, which often lack built-in cryptographic protection [6]. Studies have shown that replay attacks can be used to manipulate smart grid communications [7], vehicle-to-everything (V2X) networks, and industrial automation.

In the context of cybersecurity research, replay attacks are frequently studied alongside man-in-the-middle (MITM) attacks, where an adversary intercepts a communication channel and relays messages with potential modifications. The emergence of software-defined networking (SDN) and corresponding cloud-based IoT environments have widened the attack surface, thus calling for even more potent countermeasures, such as time-based authentication and validation using cryptographic nonces. Despite these advancements, most existing studies focus on terrestrial networks, leaving a critical gap in understanding replay attacks within satellite communication systems.

### B. Existing Research on Replay Attacks in Satellite Security

Replay attacks in satellite security present unique challenges due to the high latency and long-distance nature of space communications. In contrast to terrestrial networks, satellites are often resource-constrained, making the implementation of strong cryptographic protection computationally expensive. Many studies have provided evidence of replay attacks on GNSS, where attackers employ meaconing techniques to deceive GNSS receivers into accepting false location data [8]. Another major area of concern is replay attacks targeting telemetry and telecommand links, which can lead to unauthorized execution of satellite commands [9].

Recent studies have shown replay-based attacks to be particularly potent against Low Earth orbit (LEO) satellite constellations due to their rapid and frequent handover between ground stations. [10] During handover events, an attacker can inject malicious commands or replay previously recorded control signals. Although some studies propose hardware-based cryptographic authentication as a countermeasure [11], such approaches remain largely impractical on small satellite platforms due to their power and storage constraints.

### C. Role of SDRs in Satellite Security Testing

SDRs have become a standard tool for the evaluation of satellite security enabling researchers to simulate real-world adversarial attacks in a controlled environment. SDRs provide the capability to transmit, receive, and analyze RF signals across various frequency bands, making them ideal for testing satellite ground station vulnerabilities. Several studies have demonstrated the use of SDRs in executing replay attacks on satellite downlink communications and demonstrating that illegal command execution is possible without accessing the ground control infrastructure directly [3].

A well-documented instance is Hacking Satellites with Software Defined Radios [3], wherein the legitimate status

updates were replayed to inject false readings into the satellite's onboard sensors. SDRs have also been used to reverse-engineer proprietary protocols, exposing encryption weaknesses in both commercial and military satellites. [5] These findings underscore the critical role of SDRs in cybersecurity research, particularly in analyzing how satellite command authentication mechanisms can be bypassed.

### D. Gaps in Existing Research

While SDRs have proven to be invaluable tools for security analysis, existing research has significant limitations when applied to modern software-driven satellite control systems:

- Limited exploration of replay attack scenarios at the firmware level: Most prior studies deal with attacks at the radio-frequency level. However, the security implications at the firmware and network layers remain largely unexplored.
- Scarcity of research on SDR-free replay attack methodologies: Existing studies predominantly focus on hardware-based attack models, with limited exploration of software-only replay attack methodologies.
- Legal and regulatory barriers to SDR-based attacks: As SDRs broadcast RF signals, their activity is limited in ethical cybersecurity research by spectrum licensing restrictions.

The research aims to demonstrate the feasibility of executing replay attacks on satellite command systems using software-based approaches instead of SDR hardware. Furthermore, Scapy and Wireshark are utilized to demonstrate the lack of robust authentication mechanisms on open-source satellite control systems such as OpenSatKit (OSK), rendering them susceptible to replay attacks even without RF signal manipulation.

## III. Methodology

### A. Experimental Setup

*1) OpenSatKit (OSK) Installation on Ubuntu:* OpenSatKit (OSK) was chosen as the primary testbed for software-based replay attack evaluation due to its open-source nature and its use of NASA's cFS for spacecraft command and control. The OSK environment was set up on an Ubuntu 18.04 LTS virtual machine to ensure compatibility with the required dependencies. The necessary software packages, including Python, Scapy, Wireshark, and OSK, were installed to facilitate packet capture, command execution, and replay attack simulations.

Upon launch, OSK provided a fully functional command and telemetry interface for sending and monitoring satellite control packets.

*2) Eliminating the Need for SDRs:* Traditionally, replay attacks in satellite security research have relied on SDRs to intercept and replay radio frequency (RF) signals, taking advantage of weaknesses in satellite telemetry and command transmissions. This research, however, eliminates the dependency on SDRs by employing a purely software-based method. Key advantages of this approach include:

- Command packets are intercepted at the software level instead of RF signal interception.
- Leveraging network-layer packet manipulation software like Wireshark and Scapy to examine and manipulate satellite command traffic.
- Replaying satellite commands in real-time directly in OSK's software-based framework, without the need for physical RF interfaces.

This approach makes satellite security testing more accessible while avoiding legal and regulatory challenges associated with SDR-based transmission.

*3) Tools Used:* To execute a software-only replay attack, the following tools were employed:

TABLE I
TOOLS USED IN SDR-FREE REPLAY ATTACK

| Tool | Purpose |
|---|---|
| **Wireshark** | Packet capture and analysis of OSK command traffic |
| **tcpdump** | CLI-based network packet capture utility |
| **Scapy** | Python-based network packet manipulation tool |
| **Python 3** | Script execution for automating replay attacks |

### B. Capturing Replay Attack Packets

*1) Packet Capture Using Wireshark and tcpdump:* To identify and capture OSK command packets, Wireshark was configured to monitor UDP traffic on OSK's default command transmission port (1234):

```
udp.port == 1234
```

Captured packets were exported to a file using:

```
tcpdump -i lo -w satellite_traffic.pcap
```

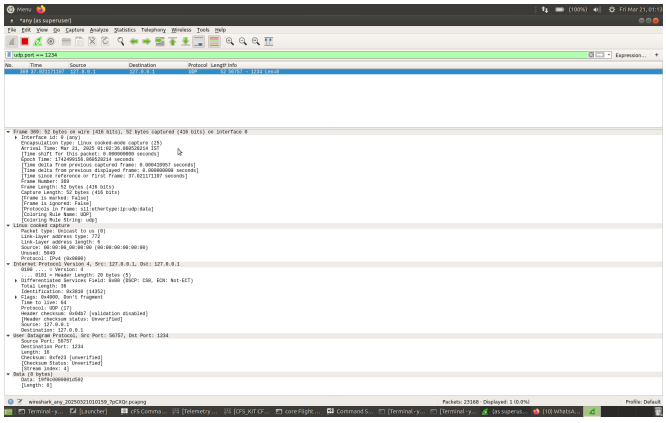

Fig. 1. Wireshark capturing OSK command packets.

*2) Analyzing Command Packets:* The captured packets were analyzed to extract critical details like:

- Source IP: 127.0.0.1
- Destination IP: 127.0.0.1
- Protocol: UDP
- Command Payload: Encoded in hexadecimal format

### C. Executing the Replay Attack

*1) Replaying Captured Commands Using Scapy:* A Python script was developed to resend the captured packets, simulating a real-time replay attack:

```
from scapy.all import rdpcap, send, UDP

# Read the captured packets from the file
packets = rdpcap("replay_attack_1.pcap")

# Loop through each packet and replay if it is a UDP
    packet
for packet in packets:
   if packet.haslayer(UDP):
      send(packet)
      print("Replayed command sent!")
```

*2) Observing OSK's Response:*

- OSK Command & Telemetry Logs were monitored to track execution results as shown in Fig 2.

Expected Outcome:

- If OSK accepted the replayed command, it indicated a successful replay attack.
- If OSK rejected it, it suggested that some form of authentication was present.

Experimental results demonstrated that OSK did not implement built-in protections against replayed commands, allowing successful execution of previously captured command sequences.

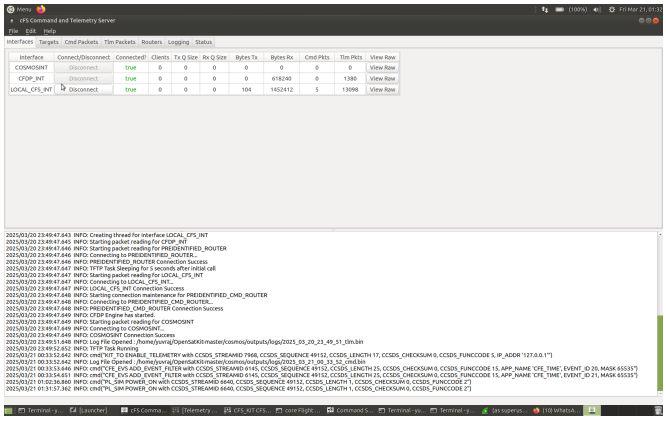

Fig. 2. OSK terminal logs showing a command replayed via Scapy, confirming a successful software-based replay attack.

## IV. Results and Discussion

Experimental results demonstrate that a replay attack can be executed successfully without the use of SDRs. Additionally, evidence of network-layer exploitation and unauthorized command execution suggests that OSK lacks essential security mechanisms, including cryptographic authentication, timestamp validation, and sequence number authentication.

### A. Replay Attack Execution and System Response

1) Replay Attack Success Rate

- Command packets were replayed using Scapy multiple times, and in each case, OSK accepted and executed the replayed commands, as shown in Fig. 1, illustrating there is no intrinsic protection against duplicate command execution. To quantify this observation, let us define the probability $p$ of a successful replay attack. In a secure system equipped with proper authentication mechanisms, this probability is expected to be:

$$p \approx 0 \quad \text{(due to authentication failures)} \tag{1}$$

However, in our experimental setup using OSK, we observed:

$$p = 1.0 \quad (100\% \text{ success over } n = 10 \text{ trials}) \tag{2}$$

Such a consistently high success rate empirically and statistically validates that OSK lacks essential anti-replay protections, including cryptographic validation or sequence number checks.

- OSK did not impose any time limits on the execution of previously sent commands, making it susceptible to repeated replay attacks. This vulnerability can be explained using a replay timing window model. Let $P(t)$ represent a valid command packet transmitted at time $t$, and $R(t + \Delta t)$ be a replayed version of the same packet after a time delay $\Delta t$. In a secure system, the command $R(t+\Delta t)$ should only be accepted if it falls within a predefined temporal threshold $\delta$:

$$R(t + \Delta t) \text{ is accepted only if } \Delta t < \delta \tag{3}$$

However, OSK does not enforce any such limit. Therefore, for all delays $\Delta t$, we observe:

$$\forall \Delta t, \quad R(t + \Delta t) \text{ is accepted} \tag{4}$$

This behavior was experimentally validated for delay intervals of 5 seconds and 1 minute, confirming OSK's complete lack of temporal freshness checks.

2) System Log Observations

- OSK was unable to distinguish between original and replayed command traces, considering all received instructions as valid and newly received.
- No error message or security alert was generated, which implies no mechanism to detect replay-based intrusions.
- Wireshark Packet Capture illustrates OSK command packets being intercepted before replay execution.
- The OSK Terminal Logs reflect the successful execution of commands without any rejections as shown in Fig. 2.

### B. Comparison with SDR-Based Attacks

This experiment highlights a significant shift in satellite security testing. Unlike conventional RF-based replay attacks that rely on SDR hardware to receive and retransmit signals, this research demonstrates that network-layer vulnerabilities are equally exploitable. A further comparison is presented in Table II.

TABLE II
COMPARISON OF SDR-BASED VS. SDR-FREE REPLAY ATTACKS

| Feature | SDR-Based Replay Attack | SDR-Free Replay Attack (This Work) |
|---|---|---|
| **Hardware Required** | SDR (HackRF, USRP) | No additional hardware |
| **Attack Vector** | RF Signal Interception | Network Packet Manipulation |
| **Legal Restrictions** | May require RF transmission license | Fully legal |
| **Execution Complexity** | High (RF tuning, signal analysis) | Low (Software-only approach) |
| **Applicability to OSK** | Not applicable (OSK does not use RF) | High (Direct command injection) |

## V. CONCLUSION AND FUTURE WORK

The study successfully demonstrates that satellite control systems can be targeted for replay attacks using a purely software-based approach, without relying on SDRs or RF transmissions. Table III outlines the success of these attacks, revealing OSK's lack of robust authentication. This is further validated by our mathematical analysis: Equations (1) and (2) show a 100% success rate $p = 1.0$ over 10 trials, in contrast to $p \approx 0$ in secure systems, while Equations (3) and (4) confirm OSK accepts replayed commands regardless of delay $\Delta t$. These findings highlight critical flaws in OSK's timing and sequence validation, underscoring the urgent need for cryptographic security enhancements.

TABLE III
ATTACK SCENARIOS IN ORIGINAL AND REPLAYED COMMANDS

| Attack Scenario | Original Command Execution | Replayed Command Execution | Attack Success? |
|---|---|---|---|
| Immediate Replay | ✓Executed | ✓Executed | ✓Successful |
| Delayed Replay (After 5s) | ✓Executed | ✓Executed | ✓Successful |
| Delayed Replay (After 1 min) | ✓Executed | ✓Executed | ✓Successful |
| Random Packet Injection | ✓Executed | ✓Executed | ✓Successful |

This research lays the groundwork for software-based replay attacks, while further refinement of these techniques is essential for building stronger cybersecurity resilience for satellite control architecture. Future studies are directed toward embedding cryptographic sequence validation techniques like Hash-based Message Authentication Code (HMAC) to prevent unauthorized command execution. Additionally, this attack methodology should be evaluated on real-world satellite systems and ground station infrastructure to assess the effectiveness of existing security mechanisms in operational environments. An extension of this study to other open-source satellite control frameworks, like NASA's cFS, Ball Aerospace's COSMOS, and European Space Agency's SCOS-2000, will provide insight into how widespread such vulnerabilities are across several platforms.

Moreover, the establishment of an automated penetration testing framework could facilitate the situational awarenes of security assessments, simulating replay attacks, analyzing system responses, and identifying possible mitigations. The integration of machine-learning anomaly detection and the creation of digital twin environments for real-time cybersecurity tests could provide a strong means to preemptively boost satellite security. Beyond replay attacks, further work should investigate whether other classes of cyber threats-e.g., command-injection attacks, protocol fuzzing, and side-channel exploits-can be executed using similar software-based techniques to broaden the SDR-free satellite security research.

In bridging the gap between theoretical research and practical, software-based penetration testing methodologies, this study lays the foundation for more secure and resilient space communication infrastructures. Future research efforts must focus on integrating authentication safeguards, improving detection to identify these attacks, and testing the implementation against real-world satellite systems to mitigate emerging challenges in space systems as a whole.

## REFERENCES


[1] Heidt, Hank, et al. "CubeSat: A new generation of picosatellite for education and industry low-cost space experimentation." (2000).

[2] McComas, David. "Increasing flight software reuse with OpenSatKit." 2018 IEEE Aerospace Conference. IEEE, 2018.

[3] Lukin, Kimberly, and Maximilian Haselberger. "Hacking Satellites With Software Defined Radio." 2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC). IEEE, 2020.

[4] Papadimitatos, Panos, and Aleksandar Jovanovic. "Global Navigation Satellite Systems (GNSS)-Attacks and Countermeasures." IEEE Military Communications Conference (IEEE MILCOM). 2008.

[5] Willbold, Johannes, et al. "Space odyssey: An experimental software security analysis of satellites." 2023 IEEE Symposium on Security and Privacy (SP). IEEE, 2023.

[6] Carracedo, Jorge Martiez, et al. "Cryptography for security in IoT." 2018 Fifth International Conference on Internet of Things: Systems, Management and Security. IEEE, 2018.

[7] Bouslimani, Mariem, et al. "Replay Attacks on Smart Grids: A Comprehensive Review on Countermeasures." IECON 2024-50th Annual Conference of the IEEE Industrial Electronics Society. IEEE, 2024.

[8] Lenhart, Malte, Marco Spanghero, and Panagiotis Papadimitratos. "Relay/replay attacks on GNSS signals." Proceedings of the 14th ACM conference on security and privacy in wireless and mobile networks. 2021.

[9] Salim, Sara, Nour Moustafa, and Martin Reisslein. "Cybersecurity of satellite communications systems: A comprehensive survey of the space, ground, and links segments." IEEE Communications Surveys & Tutorials (2024).

[10] Yue, Pingyue, et al. "Low earth orbit satellite security and reliability: Issues, solutions, and the road ahead." IEEE Communications Surveys & Tutorials 25.3 (2023): 1604-1652.

[11] Abdulmonem, Mohamed H., Ahmed K. Ismail, and Hassan Mostafa. "Design and implementation of authenticated encryption co-processors for satellite hardware security." 2021 International Conference on Microelectronics (ICM). IEEE, 2021.