Defending Small Satellites from Malicious Cybersecurity Threats

Banks Lin, Wayne Henry and Richard Dill
Air Force Institute of Technology, Wright-Patterson Air Force Base, USA
banks.lin@afit.edu
wayne.henry@afit.edu
richard.dill@afit.edu

Abstract: The connection between space and cyberspace domains is increasingly intertwined. Advancements in space technology, decreasing costs for satellite development, and the use of commercial off-the-shelf products present many cybersecurity challenges to space infrastructure. Additionally, space-based global critical infrastructure makes the space domain a prime target for malicious cyber threats. Software-defined radios introduce a potential attack vector for adversaries planning malicious satellite activity. This paper demonstrates how an adversary would send malicious commands via a software-defined radio to affect the integrity of the sensors on the satellite running NASA's core Flight System software. The experiment demonstrates one possible threat vector using a commercially available USRP N210 software-defined radio. The results show that well-constructed messages can be created to manipulate sensors on a target small satellite system. Identifying cybersecurity vulnerabilities like these in space systems can improve security and prevent disruptions for the global space enterprise.

Keywords: small satellite, space cybersecurity, software-defined radio, insider threat, integrity

1. Introduction

Space technology continues to march towards new exploration and innovation. The race to launch satellites to orbit is on for many nations in academia and the commercial sector. The cost and timeline to develop small satellites are appealing and are becoming increasingly common. This is especially true with developers using open-source software like NASA's core Flight System (cFS) (Wilmot and Kane, 2021). The motivation to gravitate towards small satellites is compelling; however, it introduces cybersecurity challenges for space assets and infrastructure (Tucker, 2019). Cybersecurity testing is one way to ensure the satellite in development is secure and ready for launch. From a test perspective, it is safe to assume that adversaries possess the intent, knowledge, time, and money to exploit a vulnerability.

There is an absence of cybersecurity standards for space systems. Without an international or industry standard, organizations can launch satellites that may lack protection against cyber-attacks which may cause security concerns for years to come. The "security by obscurity" model alone is obsolete (Jameel, 2020). The model depends on the complexity of the system design and implementation primary method for security (Waterman, 2019). Instead, the obscurity of satellite systems must couple with other layers of security. A cybersecurity standard for design, development, and execution for space systems enables all satellites to meet a minimum viable product capable of cyber resiliency. Maintaining a set of principles addresses the strategic and technical approach for space system stakeholders to combat space cybersecurity (Falco, 2019). The standard should also apply during the development phase, where the lifecycle of the space system integrates cybersecurity testing. Satellite development teams that add cybersecurity defenses after launch may risk unintended side-effects that may cost more in the long run. It is imperative to conduct cybersecurity assessments of satellites during development and before launch to ensure the spacecraft can be trusted and secured for the duration of the satellite's life in orbit.

Cybersecurity assessments on the ground segment are similar to ones conducted on traditional information technology network. However, there is less research on testing the security posture of the spacecraft itself. The consequence of this is failing to learn the unknown to the cyber threats that face spacecraft. Threats to the spacecraft include but are not limited to electromagnetic interference or signal jamming. This paper takes the approach to determine if it is possible to send malicious commands to small satellites with software-defined radios (SDRs). An SDR is a radio where the physical layer functions traditionally implemented in hardware are software-defined. Software configuration to the mixer, filter, amplifier, or modulator components reduces development costs. SDRs have human configurable software code to instruct the device how to modulate or demodulate the data. Anytime humans are in the loop, there is always a method to exploit the process. Demonstrating the existence of compromising the integrity of a satellite through insider threat is only the first step of a broader research objective to discover the capabilities to compromise and spoof satellites.

Section 2 details previous research on space system cybersecurity. Section 3 covers the research methodology of this experiment and presents the system under test. Section 4 discusses the procedures done to alter commands with malicious intent to the spacecraft and reveals the experiment results. Lastly, Section 6 concludes the experiment and discusses future work.

2. Related Work

The technological advancement in space is no longer limited to nation-states and large corporations. The commercial and civilian sectors are diving into the space enterprise because of the decreasing costs of technology when it was only attainable to nation-states in the past. Commercial off-the-shelf (COTS) or open-source software and hardware, inexpensive small satellites, and decreasing cost of space launches allow more organizations to be part of the space enterprise (Nussbaum and Berg, 2020). Companies and academic institutions take on more risks to push boundaries in space technologies and applications. Although the growth in the space industry is good for research and academia, it introduces side effects that require management. Open-source products can lower the cost and development time; however, system engineers must be aware of the cybersecurity challenges. NASA's cFS is the most common open-source flight software used in small satellites. Its software code is publicly available on GitHub for any entity to download. This research will assess NASA's cFS and leverage its software code as a potential attack vector. Space systems are increasingly becoming an attractive target for adversaries because of the significant impacts on their capabilities (Manulis et al., 2020). Adversaries avoid companies that have heavy experience and investment in cybersecurity. Instead, they target sectors prone to cybersecurity vulnerabilities, such as the space sector. Nation-states have launched orbital assets for anti-satellite capabilities; therefore, the likelihood of cyber-attacks targeting space systems is high (Vessels et al., 2019).

Insider threat is one of the many prevalent threats that space missions face. Insider threats can cause the most impact because these individuals possess the necessary permissions to conduct their malicious operations. A NASA report concluded that insider threats are often dishonest systems personnel, disgruntled staff members, or trusted business partners (Book, 2015).

Reputable companies and universities have conducted extensive research on the cyber resiliency of flight software on satellites. The Aerospace-Carnegie Mellon team describes cyber resilience as needing to be an integrated capability of every system, not a "bolt-on" capability or an afterthought (Wheeler et al., 2017). Their paper emphasized two approaches to conducting a security assessment for flight software: dynamic application security testing (DAST) and static application security testing (SAST). The DAST and SAST are part of a seven-step lifecycle of analysis, design, coding, testing, integration, deployment, and maintenance, shown in Figure 1.

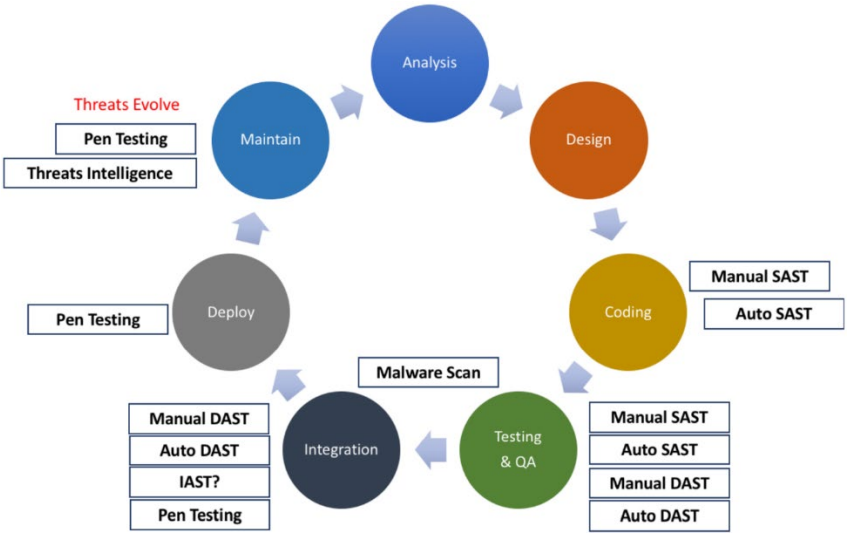

Figure 1: Software Security Assessment Lifecycle (Wheeler et al., 2017)

Hacking with SDRs is not a new concept. With low-cost SDRs and open-source software, many enthusiasts can get their hands on these tools to experiment with the internet of things (IoT). Garage door openers are common targets where attackers can capture the radio frequency (RF) signal and replay the signal from an SDR (Gupta,

2017, pp.223-263). With the right tools and skillset, enthusiasts can access hacking IoT devices that use RF communication. Hung and Vinh studied vulnerabilities in IoT devices from unknown RF protocols by analyzing the frequency and decoding the RF signal (Hung and Vinh, 2019). After decoding the recorded signal data, they rebroadcasted it to the IoT devices, performing a replay attack. This paper applies Hung and Vinh's method to satellite communication.

Hacking satellites with SDRs is also not a new concept. Lukin and Haselberger conducted a replay attack using a HackRF SDR on a Reaktor Hello World satellite (Lukin and Haselberger, 2020). In their experiment, the communication between satellite and ground station lacked proper authentication and failed to enforce encryption of the communication channel. They used HackRF as their SDR to transmit the command and open-source software called GQRX, or GNU Radio, Qt, Rx, to capture the received frame. They translated the raw data in the hexadecimal bitstream for the replay attack and sent it over Gaussian frequency-shift keying (GFSK) modulation. Their goal was to demonstrate sending commands to a satellite, simulating real communication through low-cost COTS products and open-source software.

Although Lukin and Haselberger could conduct a replay attack, they could not send commands with their own values to the satellite. This research builds on their work to send altered malicious commands to the satellite to disrupt the integrity of the systems on the spacecraft.

3. Methodology

This section discusses the research methodology, explains the motivation and approach, describes the system under test, and details the operationally representative test environment.

3.1 Problem Statement

Cyber attacks can alter the confidentiality, integrity, and availability of a system. Cyber threats to space systems are persistent, with increasingly severe effects against military, commercial and academic institutions. Satellite development teams that delay adding cybersecurity defenses until after launch incur unintended risk side-effects that may cost more in the long run. It is imperative to conduct cybersecurity assessments on space systems to mitigate potential cybersecurity vulnerabilities during the design and development. Cybersecurity assessments ensure that the systems released for production uphold a secure cybersecurity posture.

3.2 Motivation

There is a gap in testing the space vehicle of a satellite system. Often, it is due to budget constraints, lack of suitable simulation, or prioritization to allocate resources towards cyber testing. Often, system stakeholders conduct cybersecurity vulnerability assessments on satellite systems' ground stations because they resemble traditional information technology network structures in non-space systems. The immature standard on conducting cybersecurity assessments for the space vehicle of satellite systems warrants attention, especially in small satellites where the technology is more readily accessible.

Discussing cyber vulnerabilities on satellite systems is a sensitive topic. The studies and research done on military or commercial satellite systems are often classified or kept close hold. The minimal amount of public and unclassified research in cybersecurity testing satellites affirms the need for research on this topic. Research is particularly scarce when focused on hacking and causing adverse effects to satellite systems through the satellite communication links.

3.3 Approach

This research produces an assessment of cybersecurity vulnerabilities and attack vectors that threaten a space vehicle. The approach sets up an operationally representative test environment for the system under test. The data flow in this experiment must parallel an environment operating in the real world. The best approach to create this test environment is to incorporate a FlatSat. A FlatSat is a large motherboard that includes the satellite components installed and connected as if it was the actual satellite. The environment connects to a computer that hosts virtual machines (VM) to simulate the ground and flight software in the test setup. The FlatSat used in this experiment consists of two components from the satellite: an electrical power supply and a telemetry, tracking, and commanding component. The VM emulates the rest of the components in software. Connections between the computer, SDR, and the FlatSat create a test environment that accurately simulates the uplink and downlink between the satellite and the ground station.

3.4 System Under Test

Small satellites, characterized by their low cost, and small size, enable organizations to conduct their mission without the complexity of a large satellite. Their size and weight make it more affordable to launch into orbit. Due to low costs, it creates the ability to reach space widely accessible to organizations and countries when they previously could not. CubeSats are small satellites made for research. A 10 cm x 10 cm x 10 cm cube makes up one unit or 1U. CubeSats have shorter development times enabling launch within a couple of years, whereas conventional satellites could take 5 to 15 years. Low cost and short development times allow CubeSats to launch into space every 2 to 4 years to start new research or even update technology (Space, 2019).

This research paper will refer to the 6U CubeSat used in this experiment as “SmallSat”. The SmallSat uses open-source flight and ground software, cFS and COSMOS, respectively. This particular SmallSat is a research satellite with a primary objective to fly the software bus and communicate to the ground station.

3.4.1 COSMOS

COSMOS is the ground software or front-end application used by operators in the satellite operations center. It is an open-source software developed by Ball Aerospace that provides a user interface and functionality to command and control space systems (Ball Aerospace, 2020). For the SmallSat, operators interface COSMOS to send commands to the spacecraft.

3.4.2 Titan/Blockbuilder

Titan is the software interface to the Universal Software Radio Peripheral (USRP) N210, designed to communicate to the Cadet Plus. Blockbuilder is the software tool that provides signal processing blocks to implement to USRP N210. For this paper, Titan and Blockbuilder are interchangeable terms. The result of Titan is a waveform and message format that the Cadet Plus accepts. This experiment uses Titan unaltered for the SmallSat. Figure 2 is a snippet of Titan showing the global variables declared for the uplink. This XML file is the configuration sent to the USRP to process the waveform to send to the Cadet Plus.

```
▼<BLUEPRINT>
   <GLOBAL name="g_CommandPort" val="30555"/>
   <GLOBAL name="g_UsrpIp" val="192.168.10.2"/>
   <GLOBAL name="g_CadetAddress" val="0x6D"/>
   <GLOBAL name="g_TxCenterFrequency" val="449.775e6"/>
   <GLOBAL name="g_LogLevel" val="info"/>
   <GLOBAL name="g_TxBitRate" val="9600"/>
   <GLOBAL name="g_TxSampleRate" val="2000000"/>
   <GLOBAL name="g_TxGain" val="0"/>
   <GLOBAL name="g_BurstCount" val="16"/>
   <GLOBAL name="g_BurstDelayMs" val="150"/>
  ▼<CHAIN name="Uplink">
```

Figure 2: Titan XML Configuration

3.4.3 Cadet Plus Radio

The Cadet Plus radio is a split band, full-duplex, store, and forward radio (Space Dynamics Laboratory, 2017). It is the physical component that operates the telemetry, tracking, and commanding of the satellite. The Cadet Plus is the uplink and downlink interface for the spacecraft. This experiment focuses on the ultra-high frequency uplink configured with a frequency range of 449.75 Mhz to 450.25Mhz, modulated at 9600 bps, as defined in the Interface Control Document (ICD).

3.4.4 Core Flight System

NASA's cFS is an open-source platform consisting of reusable software frameworks and software applications (Wilmot and Kane, 2021). In the SmallSat, the Command & Data Handler (C&DH) is the physical board that hosts cFS. For the experiment, cFS is emulated on the Titan VM.

3.5 Test Environment

The virtual test environment simulates the data being transmitted and received by the ground station and spacecraft. Hardware equipment required for this setup includes a desktop to host a VM, USRP N210, Cadet Plus radio, power supply, and cables for connection. Software required for this setup includes an Ubuntu Titan VM,

COSMOS, cFS, Blockbuilder, GHex, and Wireshark. USRP N210 was the SDR used in this experiment because it is the exact SDR model used for the system under test. This avoids disparity and variance. Figure 3 depicts the test setup architecture for the scope of this experiment.

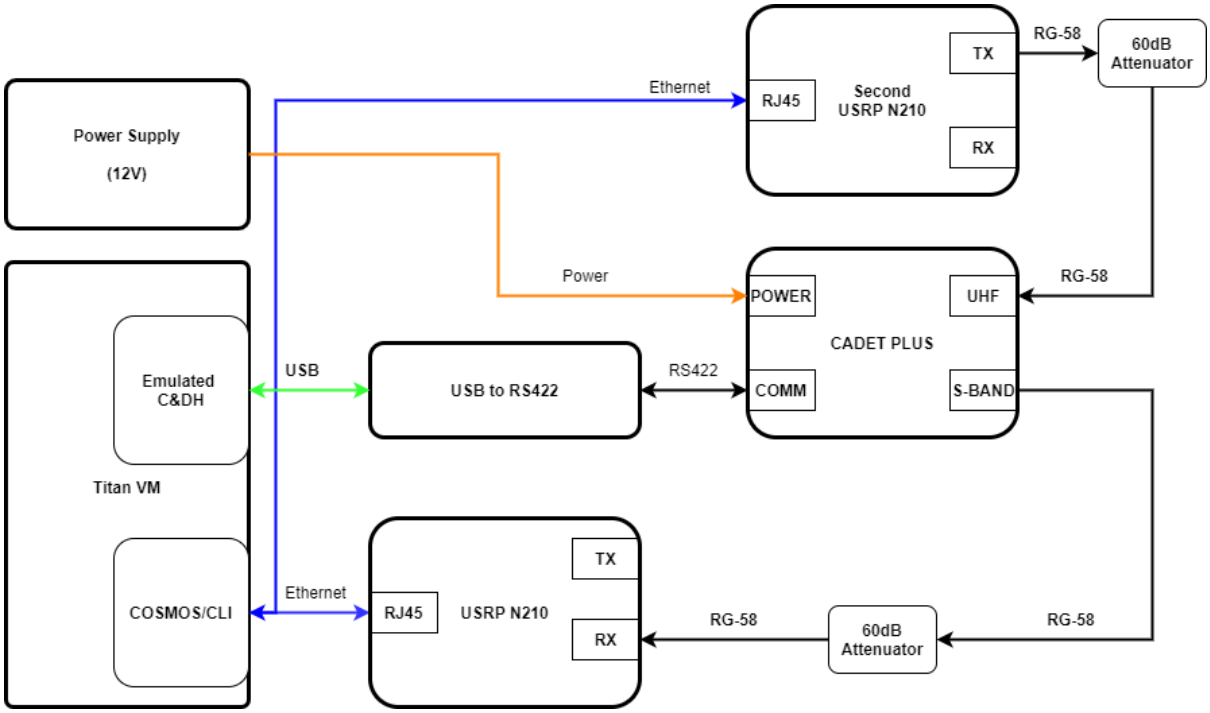

Figure 3: Test Setup Architecture

SDRs connect to the Titan VM via Ethernet cables. SDRs forwards the modulated data to the Cadet Plus radio, thereby simulating the data passing to and from the ground station and the spacecraft. Flight software runs on a VM in an emulated C&DH rather than a physical C&DH board. Cadet Plus radio then forwards validated data to the emulated C&DH, where the rest of the data processing occurs. Figure 4 depicts the actual test setup in the lab.

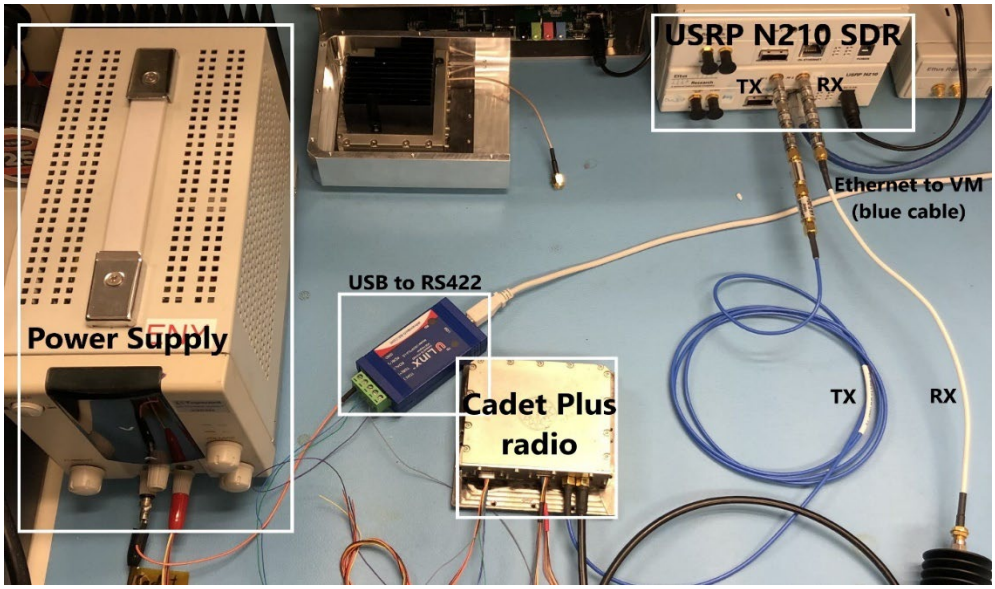

Figure 4: Image of Test Setup in Lab

4. Test Scenario and Results

This section discusses the test scenario of interest and the experimental methodology.

4.1 Test Scenario

The test scenario in this research examines the insider threat as the first step to a greater research area toward identifying vulnerabilities in small satellites. An insider threat emerges from the ground station where the operator resides. Origin of attack can either be on the computer running the ground software or be on a computer on the same network. The insider sends an altered command with malicious intent via command line interface (CLI) terminal without access to COSMOS. The purpose of sending this command from a different source other than COSMOS is to demonstrate a proof-of-concept that the spacecraft can accept commands originating from a different source. The specific command executed in this scenario is called "SET_PARAMETER" that targets SmallSat's MAI401 system. MAI401 is a physical component of the spacecraft part of the Attitude Determination and Control System (ADACS) (Adcole Marland Aerospace, Inc, 2020). MAI401 has many sensors that keep the spacecraft afloat while in orbit. One of the sensors is a sun calibration that calculates how much sun it receives. The solar cell outputs an analog signal with a voltage range of 0 to 3.3 volts of direct current (VDC) where 0 is dark and 3.3 in full sun. SmallSat uses the equation shown below to determine the level of sun exposure.

$$sunVec_b = CssGain * (css + CssBias) / CssMax$$

From the $sunVec_b$ equation, the "SET_PARAMETER" command can change the *CssGain* value. For this experiment, it is the only variable modified. The $sunVec_b$ variable increases as *CssGain* increases due to the multiplication sign in the equation. This experiment intends to increase the $sunVec_b$ value by increasing the *CssGain* variable to manipulate the sensor reading to extremely high, preventing the MAI401 to track the sun and calculate an attitude.

The biggest challenge of sending a successfully altered command is ensuring the message format is accepted by Cadet Plus. If the message format is off by one byte, it flags the cyclic redundancy check (CRC) and throws the message out. Therefore, the goal is to send an accepted command for the Cadet Plus to push it on the software bus. The C&DH then routes the command to the appropriate application.

4.2 Attack Implementation

The altered command must have a message format that the Cadet Plus accepts. Sniffing traffic with Wireshark between the VM and the USRP N210 enables capturing the correct message format. To capture the command, an insider must execute the command from COSMOS. The data section shows the raw hexadecimal values, containing data from the Titan header to CRC, as shown in Figure 5.

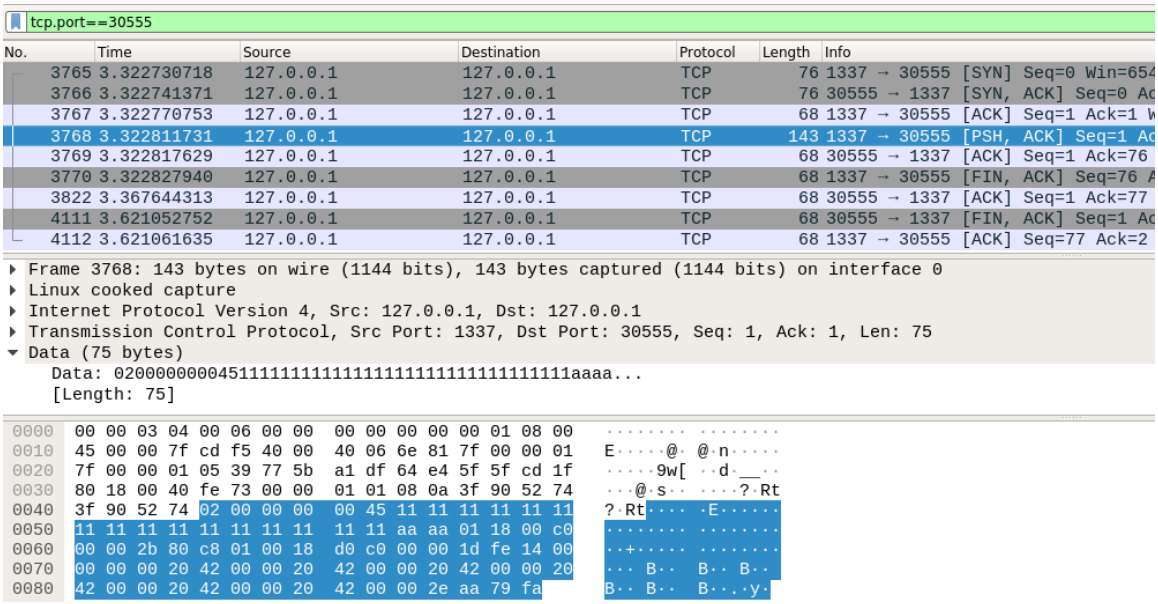

Figure 2: Wireshark Capture of the Command

The attacker captures the altered command traffic from Wireshark and recreates the raw hexadecimal data into a container file. GHex tool converts hexadecimal values to a container file called "PassThroughTest_set_param.dat", shown in Figure 6.

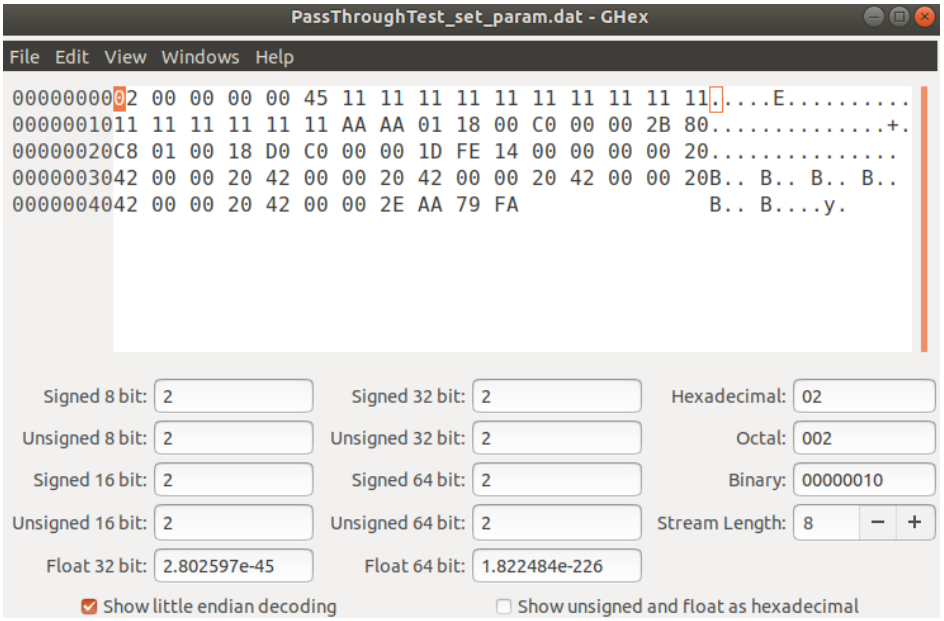

Figure 3: Edit Hexadecimal Data in GHex

The Cadet Plus ICD was used to verify the message format required to send to the USRP. Every byte must match the appropriate field, as portrayed in Table 1.

Table 1: General Cadet Plus Message Format (Space Dynamics Laboratory, 2017)

Bytes	Field
0-5	Space Packet Header
6	Flags (CRC, Quiet, Reserved)
7	Message ID
8-9	Dialog ID
10+N	Payload Data
10+N+1	CRC

The script portrayed in Figure 7 sends data in a .dat file specified with the source and destination port.

```
while [[ $(date -u +%s) -le $endtime ]]
do
  echo $(date +%s) $(cat PassThroughTest_set_param.dat)
  cat PassThroughTest_set_param.dat | nc -p 1337 -q $waitTime $addr 30555
  sleep 5s
done
```

Figure 4: Script to Run Command

Sending the command with a script replaces the need to send the command from COSMOS. Any individual with access to a CLI and a compatible RF software-defined radio can send commands to the spacecraft.

An additional *printf* statement that prints the hexadecimal data from the altered command verifies whether the altered command was successful. If the data passes the checks on the Cadet Plus, it is sent to the software bus. Once the data is on the software bus, cFS will display the *printf* statements, shown in Figure 8. The output from these *printf* statements verifies a successful command execution.

```
// Check for a completed message and send it to SB
if(sequence->ActualBytes == sequence->ExpectedBytes) {

    printf("Message in hex: \n\t");
    for(uint16 i = 0; i < sequence->ActualBytes; i++) {
        printf("%02X ", sequence->Buffer[i] );
        if(i%8 == 7) {
            printf("\n\t");
        }
    }
    printf("\n");

    CI_SendMsg(sequence->RadioID, &(sequence->Buffer));

    // Clear the sequence
    memset(sequence, 0, sizeof(CI_MessageSequence_t));
} else if(sequence->ActualBytes > sequence->ExpectedBytes) {
    CFE_EVS_SendEvents(CI_INVALID_MSG_SEQ_ERR_EID,
                       CFE_EVS_ERROR,
                       "CI Message sequence is invalid
                       (More bytes received than expected)")
```

Figure 5: *Printf* Statements on Microsoft Visual Studio for Verification

The last step to the attack implementation is to execute the command as the insider. To begin test execution, the insider sends the command via CLI to execute the script. Figure 9 depicts the script from a terminal in the virtual test environment during test execution.

```
titan@titan-VirtualBox:~/Documents/Banks$ ./sendPassThrough1_2.sh
./sendPassThrough1_2.sh: line 27: warning: command substitution: ignored null by
te in input
1632249430 [illegible]B B B B B B.
```

Figure 6: Execute Script from Terminal

4.3 Results

After executing the script, the flight software displayed the altered command in hexadecimal and prompted the following messages, shown in Figure 10: "Message sequence successfully completed and sent to the SB" and "MAI401: Set Parameter Command Success". These messages confirms that the message sent from the script was successful and spacecraft executed the command as written.

```
HS: The CPU is alive.
Message in hex:
        18 D0 C0 00 00 1D FE 14
        00 00 00 00 20 42 00 00
        20 42 00 00 20 42 00 00
        20 42 00 00 20 42 00 00
        20 42 00 00
EVS Port1 42/1/CI 23: Message sequence successfully completed and sent to the SB
EVS Port1 42/1/CI 20: CI: Message was sent to SB
[illegible]B B B B B B
[illegible]VS Port1 42/1/MAI401 32: MAI401: Set Parameter Command Success
HS: The CPU is alive.
```

Figure 7: cFS Shows Message in Hex *Printf* Statement Additions

The Cadet Plus radio accepted the command and pushed it onto the software bus. The altered command successfully executed and forced the satellite to change the $sunVec_b$ sensor values to detect its exposure as "full sun" due to the altered *CssGain* value. Figure 11 highlights the attack path in red over a data flow diagram of the system between ground station and spacecraft. It depicts that the command was able to reach to the MAI401 can cause impact.

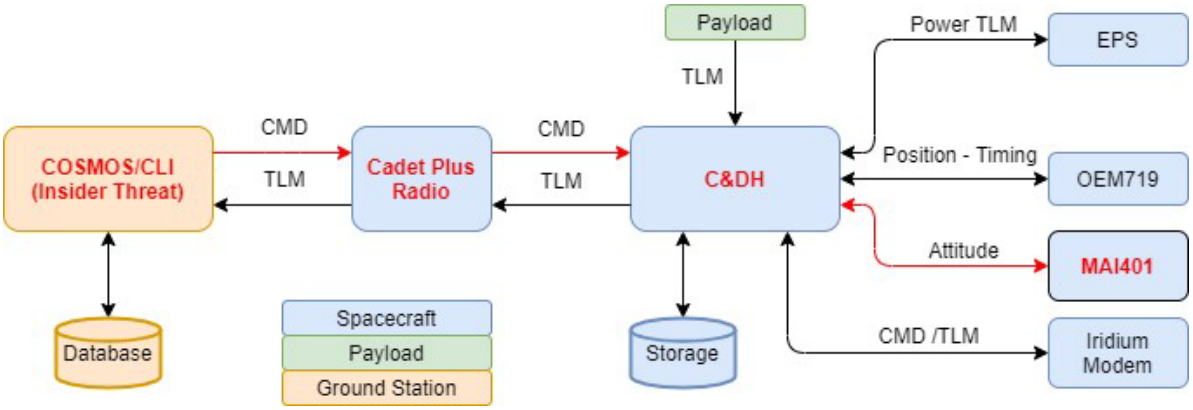

Figure 8: Insider Attack Path

The experimental scenario successfully impacted the satellite's integrity by changing the *CssGain* value, tricking the satellite sensor into believing it is extremely high or at "full sun" exposure. Using the "SET_PARAMETER" command to change the *CssGain* value is just one of many options to alter commands for malicious intent. The ADACS has many more commands, from changing *CssBias* that affects the sun sensors to setting the Cadet Plus to test mode and locking the configurations.

Capturing the entire data with Wireshark was a practical approach to generating a file containing the raw hexadecimal of the altered command. Although this attack may seem unlikely outside of an insider threat scenario, a determined adversary can learn the correct message format and CRC calculation given enough time and resource. The fact that unauthenticated commands can be accepted by the space system without COSMOS proves that it is possible to spoof uplink commands from other untrusted sources. The primary factor is that the raw data format must meet the configurations accepted by the Cadet Plus radio.

5. Conclusion and Future Work

In conclusion, the initial experiment demonstrates the capability to send an altered command with malicious intent to the satellite without using COSMOS to send the command. The attack successfully changed the value of *CssGain*, unjustly exposing the sensor to "full sun." The results of this experiment can be replicated for commands only requiring one uplink packet without requiring an acknowledgment.

Software configurations for the USRP N210, remained unaltered for this experiment. It ensures that the correct waveform would reach the Cadet Plus. Since the Titan and USRP N210 are software-defined, an adversary can use open-source software alternatives to Titan, such as GNU Radio Companion, to accomplish the same objective. Adversaries can configure their SDR settings to meet the requirements for Cadet Plus. Future work will investigate sending altered commands external to the ground segment to signify an outsider threat. It requires a well-configured SDR that interfaces with a separate USRP N210 to send the command to Cadet Plus. Additional future work includes characterizing the risk of satellite commands and associating its risk to specific subsystems on the spacecraft.

The space enterprise cannot neglect the legitimate concerns of an insider threat. However, if this cyber-attack demonstrated in this paper evolves into an outsider threat, the impact on space systems can be detrimental. The cyber-attack vector like the ones described in this paper may apply to small satellites used in universities and organizations that run NASA's cFS. The ability to identify, defend, and mitigate these cyber-attacks will position the global space enterprise in a better cybersecurity posture from potential attacks and disruption.

Disclaimer

The views expressed are those of the author and do not reflect the official policy or position of the US Air Force, US Space Force, Department of Defense, or the US Government.

References

Adcole Marland Aerospace, Inc (2020). MAI-401 Mini ADACS. pp.36, 38, 49, 70.
Ball Aerospace (2020). COSMOS. [online] Ball. Available at: https://www.ball.com/aerospace/programs/cosmos.
Book, G., 2015. Security Threats against Space Missions. Informational Report.

Falco, G., 2019. Cybersecurity principles for space systems. Journal of Aerospace Information Systems, 16(2), pp.61-70.
Gupta, A. (2017). The IoT Hacker's Handbook. Apress, pp.223–263.
Hung, P.D. and Vinh, B.T. (2019). Vulnerabilities in IoT devices with software-defined radio. In: 2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS). IEEE, pp.664–668.
Jameel, S. (2020). Security Through Obscurity: Valid Security Layer or Bad Idea? [online] cgscomputer.com. Available at: https://cgscomputer.com/cyber-security-through-obscurity/.
Lukin, K. and Haselberger, M. (2020). Hacking Satellites With Software Defined Radio. In: 2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC). IEEE, pp.1–6.
Manulis, M., Bridges, C.P., Harrison, R., Sekar, V. and Davis, A., 2021. Cyber security in new space. International Journal of Information Security, 20(3), pp.287-311.
Nussbaum, B. and Berg, G. (2020). Cybersecurity implications of commercial off the shelf (COTS) equipment in space infrastructure. Space infrastructures: From risk to resilience governance, pp.91–99.
Space Dynamics Laboratory (2017). Cadet Plus Radio Interface Control Document. Space Dynamics Laboratory (SDL), pp.14–15.
Space, A. (2019). 10 Advantages of CubeSats vs. Conventional Satellites. [online] Alén Space. Available at: https://info.alen.space/advantages-of-cubesats-vs-conventional-satellites.
Tucker, P. (2019). The NSA Is Studying Satellite Hacking. [online] Defense One. Available at: https://www.defenseone.com/technology/2019/09/nsa-studying-satellite-hacking/160009/.
Vessels, L., Heffner, K. and Johnson, D., 2019, July. Cybersecurity Risk Assessment for Space Systems. In 2019 IEEE Space Computing Conference (SCC) (pp. 11-19). IEEE.
Waterman, S. (2019). Satellite Providers Stymied by Lack of Cyber Standards - Via Satellite -. [online] Via Satellite. Available at: https://www.satellitetoday.com/cybersecurity/2019/11/14/satellite-providers-stymied-by-lack-of-cyber-standards/.
Wheeler, W.A., Cohen, N., Betser, J., Meyers, C., Snavely, W., Chaki, S., Riley, M. and Runyon, B. (2017). Cyber Resilient Flight Software for Spacecraft. In: AIAA SPACE and Astronautics Forum and Exposition. p.5305.
Wilmot, J. and Kane, L. (2021). Core Flight System. [online] cfs.gsfc.nasa.gov. Available at: https://cfs.gsfc.nasa.gov/Introduction.html.