High-Assurance Cyber Space Systems for Small Satellite Mission Integrity

Daria C. Lane, Enrique S. Leon, Francisco C. Tacliad, Dexter H. Solio, Dmitriy I. Obukhov, Daniel E. Cunningham
Space and Naval Warfare Systems Center Pacific
53560 Hull Street, San Diego, CA 92152
dlane@spawar.navy.mil

ABSTRACT

As the complexity of embedded computing platforms continues to grow, small satellites are increasingly deployed with operating systems having known cybersecurity vulnerabilities. Using common exploit techniques, potential intruders may compromise the capabilities and the integrity of the space mission. Moreover, the prominent use of commercial off-the-shelf (COTS) products in small satellites also increases the probability of attack via widely known vulnerabilities associated with commercially manufactured parts. System developers frequently assume that software and hardware components communicate through specified interfaces and primary data paths, but these assumptions cannot be fully guaranteed. By architecting secure space vehicle and ground control systems with functionally correct software components, mission critical vulnerabilities may be reduced.

INTRODUCTION

Cybersecurity threats present a systemic challenge to modern space missions. Networked embedded systems are highly susceptible to cyber attacks. Researchers and hackers have shown that embedded systems, such as those deployed in small satellite architectures, are vulnerable to remote attacks that can cause physical damage to the system and ultimately compromise mission integrity [1].

The High-Assurance Cyber Space Systems (HACSS) approach utilizes a secure framework to ensure mission critical functionality of space vehicle (SV) and ground station operations.

High Assurance Cyber Military Systems

In order to provide a higher level of trust between components and achieve greater assurance of mission integrity for small satellite systems, we apply the methodologies and tools developed under the DARPA High-Assurance Cyber Military Systems (HACMS) program to HACSS. HACMS technologies are directly applicable to autonomous vehicles [1, 2] including small satellites and larger space platforms.

HACMS technologies enable the synthesis of functionally correct software components in order to ensure the integrity of unmanned cyber physical systems. We utilize the HACMS tools to abstractly define the small satellite system architecture and generate a functionally correct set of software components. The architecture is built upon a proven-correct microkernel to ensure the integrity of the system. This approach ensures a secure space platform, with a verified system architecture and proper isolation between respective software components. By implementing a formal methods based approach, where high assurance is defined as functionally correct [1], mission critical and security properties are satisfied.

Our analysis of the HACMS toolset includes investigation or applicability to space vehicles and satellite ground stations. We have identified exploit mechanisms for which satellites and ground stations may be compromised, and provide solutions using integrated HACMS methodologies and enhanced network cyber security protocols. By securing operations on both ground and flight end-points, we expect to radically improve the end-to-end security and integrity for small satellite space operations.

CYBER VULNERABILITIES

Our discussion of potential cyber vulnerabilities examines a modular, open architecture satellite system and considers two vulnerability categories: hardware and software.

Hardware Vulnerabilities

For small satellite systems, many of the components are externally sourced, including the electrical power system, attitude control, command and data handling, antennas, and even mission payload. During fabrication these COTS components can be maliciously modified

[3]. A hardware component infected with malicious software could spread to other components or modify the intended behavior of the system.

Physical sensors and ports are other potential hardware vulnerabilities. False-data-injection (FDI) attacks, either by hacking the physical sensor device or tampering with sensor data, can trigger incorrect control actions [4]. Furthermore, an adversary can hijack command and control by injecting a Trojan or malware via an open physical port or interface.

Software Vulnerabilities

Unsecure code, lack of authentication, unencrypted traffic, and poor protocol implementation are a few examples of potential software vulnerabilities. Human coding errors from poor coding practices, insufficient unit test coverage, legacy code, or code that originates from an untrusted source, may contain bugs or undefined behavior that could be exploited resulting in code injection, remote code execution, or denial of service (DoS).

Ground stations are often networked to accommodate a variety of missions and users, making them more prone to network-based attacks such as replay, packet injection, session hijacking or eavesdropping. An adversary may also be able to exploit software vulnerabilities in poor network stack implementations and weak cryptography. Small satellite subsystems are often built to operate over UDP/IP. Such implementations are designed to work with specific inputs but should be robust enough to gracefully handle malformed data. Faults that can be triggered by input sent by a user will correspond to a bug in the implementation of the application. Ultimately this bug may result in an exploitable feature within the software code.

HIGH ASSURANCE SPACE VEHICLE

The space vehicle contains most of the critical functionality of the system and is a prime location for many of these cyber threats to occur. The SV is composed of various subsystems and interfaces, and the interactions with these subsystems must properly conform to the given requirements. This becomes more difficult as the size and complexity of the interactions grows. Thus, the need arises for a method to prove the correctness of the vehicles subsystem interactions in order to guarantee adherence to system requirements [5].

By modeling our system with architecture verification tools such as the Architecture Analysis and Design Language (AADL), we abstractly define the system architecture, interfaces, and their interactions. By asserting certain conditions for our subsystem input, we may attempt to verify if the defined architecture can guarantee adherence to the requirements [6].

Once the architecture can be verified to adhere to the requirements, the subsystem implementations must then be able to correctly execute the subsystems functionality and correctly communicate to other subsystems. With proper engineering practices, standard software implementations can avoid most of the common errors which can lead to critical system failure or exploitation. Implementing the software subsystems with type and memory safe languages may aid in completely removing these types of errors. A Meta programming language that employs these features could also be used to generate safe code if the proper backend exists. Code synthesis tools for the generation of serializing and parsing components may also be utilized instead of manually produced code. In the context of a small satellite, these languages should best be employed in the development of satellite subsystem components and messaging layers [7].

Even when the architecture is verified and the implementation was written with a safe language, we cannot assume that the system is invulnerable. The system should have some form of guaranteed isolation between software components. If the isolation is strong enough, the software system should be able to contain faults or intrusions into a single running process. There are many isolation kernels available with varying degrees of formal verification which can provide such a type of process isolation. In some configurations for these environments, it is possible to restart an anomalous process and leave the rest of the system unharmed.

One possible implementation includes a type of common interface gateway that is able to route traffic to and from the software control system and the various types of physical interfaces. These interfaces should be separate and only terminate at the gateway. The gateway should then be able to only allow traffic according to a statically defined set of rules. The data flow could then be directed to a form of stateful firewall, or contact based command arbitrator, that validates the contents of the message according to the source and destination. A set of guardian rules on the gateway can also be used to prevent access to the interface gateway from unauthorized processes. Control arbitration methodologies should be implemented associated with the current state of the space vehicle. Message forwarding from the control software is checked against the satellite's state. The state of the satellite can be continuously verified by a

form of safety monitor that can place the satellite into a minimal safe mode until recovery is possible. The legacy control software can be initially implemented in an unsecure, isolated environment until it can be re-implemented in a verified manner and integrated into the isolation kernel, as shown in Figure 1 [8].

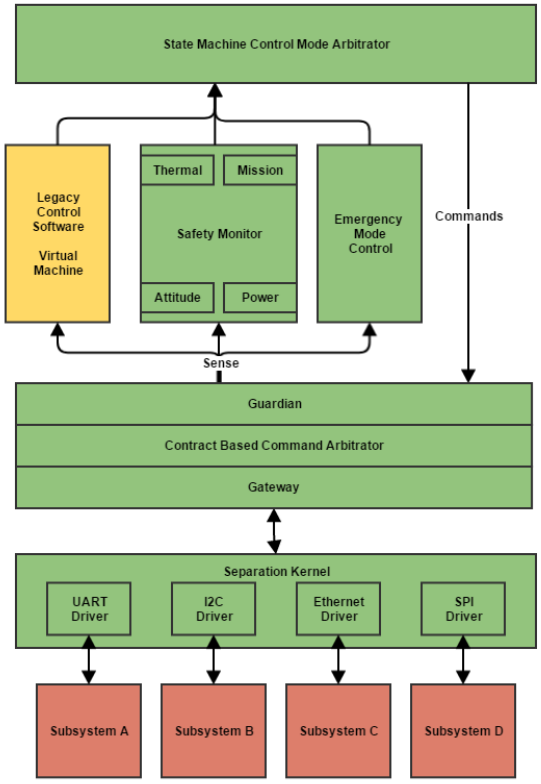

Figure 1: High Assurance Architecture for Space Vehicle

By combining the tools listed above, one should have the ability to design software for small satellites that is verified to adhere to the requirements, diminish instances of implementation errors, and have each process executed in an isolated environment to reduce the vulnerability attack surface. The satellite should then be resistant to attacks that exploit faults in the architecture or implementation. By modularizing the system, one can begin to secure the system by implementing each component with safe-generated code, running the module in an isolation kernel, and verifying its adherence to system requirements. Once complete, the satellite software system should now be composed of verified functionally correct components that adhere to a strict contract of execution and communication.

HIGH ASSURANCE GROUND STATION

Implementing a high-assurance ground station requires a life-cycle design approach that focuses on securing communication gateways. In small satellite systems, the primary function of the ground station is Telemetry, Tracking, and Control (TT&C). Commands are sent and received through the protocol gateway. Ground station software can be designed to formally guarantee that only legitimate traffic flows to and from the system [9]. Additionally, integrating formal methods and secure coding requirements into the ground station software design can assure that the code satisfies its safety requirements and is free of undefined behavior.

HACMS NANOSATELLITE APPLICATION

A compromised ground station is used by an adversary to send commands to the space vehicle, preventing the SV from executing its mission(s). In this sample scenario, our goal is two-fold. First, prevent ground station compromise. Second, prevent a compromised ground station from significantly impacting the SV's mission(s).

To prevent ground station compromise, we plan to instill good physical security to minimize unauthorized physical access to the ground station, and to utilize the HACMS tools to develop functionally correct code to prevent software exploits. Developing functionally correct code is not enough to prevent intrusions. We also plan to develop software that would detect intrusions, and respond accordingly to regain control of the compromised system.

To ensure SV mission integrity, we aim to limit the impact of a compromised ground station. We assume that an adversary has somehow physically obtained a ground station through various means. The SV receives commands from a ground station and needs to determine whether that specific ground station is compromised. Our HACSS solution is to develop an attestation protocol that validates ground station messages as they are received. Untrusted messages are simply ignored and not executed by the flight software.

Once the high-assurance system architecture for the ground and space platform has been fully integrated, we anticipate on-orbit survivability and resilience to attack. Penetration testing and analysis shall be conducted at the bus and ground station level to verify the effectiveness of the HACSS implementation.

CONCLUSIONS

The research presented in this paper is a critical step towards establishing a baseline for high assurance space system architectures to ensure space mission integrity.

By integrating formally verifiable, functionally correct software tools within the space vehicle and securing the ground station segment, mission critical functionality may be preserved during cyber attack.

Acknowledgments

This work was supported by the Defense Advanced Research Projects Agency (DARPA) Information Innovation Office (I2O) High Assurance Cyber Military Systems project.

References

1. Fisher, K. "Using formal methods to enable more secure vehicles: DARPA's HACMS program." *Proceedings of the 19th ACM SIGPLAN International Conference on Functional Programming*, Gothenburg, Sweden, September 2014.
2. M. W. Whalen, D. Cofer, and A. Gacek, "Requirements and architectures for secure vehicles," *IEEE Software*, vol. 33, no. 4, 22–25, 2016.
3. Tehranipoor, M. and Koushanfar, F. "A survey of hardware trojan taxonomy and detection." *IEEE Design & Test of Computers*, 27.1, 2010.
4. Franchetti, F., Low, T. M., Mitsch, S., Mendoza, J. P., Gui, L., Phaosawasdi, A., Padua, D., Kar, S., Moura, J., Franusich, M., Johnson, J., Platzer, A., Veloso, M. M., "High-Assurance SPIRAL: End-to-End Guarantees for Robot and Car Control." *IEEE Control Systems*, 37(2), 82-103, 2017.
5. Murugesan, A., Whalen, M.W., Rayadurgam, S., and Heimdahl, M., "Compositional Verification of a Medical Device System," *Proceedings of the 2013 ACM SIGAda Annual Conference on High Integrity Language Technology*, Pittsburgh, PA, November 2013.
6. Backes J., Cofer, D., Miller, S., Whalen, M.W., "Requirements Analysis of a Quad-Redundant Flight Control System" *Proceedings of the 2015 NASA Formal Methods International Symposium*, Pasadena, CA, April 2015.
7. Trevor, E., Pike, L., Winwood, S., Hickey, P., Bielman, J., Sharp, J., Launchbury, J., "Guilt Free Ivory," *Proceedings of the 2015 ACM SIGPLAN Symposium on Haskell*, September 2015.
8. Nogin, A., "High Assurance Cyber Military Systems (HACMS) Program Update: Ground Systems," HRL Laboratories, October 2016.
9. Ellison, R., Householder, A., Hudak, J., Kazman, R., and Woody, C., "Extending AADL for Security Design Assurance of Cyber-Physical Systems," CMU/SEI Report 2015-TR-014, Software Engineering Institute, December 2015.