SecureCPS: Defending a Nanosatellite Cyber-Physical System

Lance Forbes[1], Huy Vu[1], Bogdan Udrea[2], Hamilton Hagar[2],
Xenofon D. Koutsoukos[3], Mark Yampolskiy[3,4]
[1]Global InfoTek Inc. (GITI), [2]Embry-Riddle Aeronautical University (ERAU),
[3]Vanderbilt University, Institute for Software Integrated Systems (ISIS),
[4]University of South Alabama (USA)

ABSTRACT

Recent inexpensive nanosatellite designs employ maneuvering thrusters, much as large satellites have done for decades. However, because a maneuvering nanosatellite can threaten high-value assets (HVA) on-orbit, it must provide a level of security typically reserved for HVAs. Securing nanosatellites with maneuvering capability is challenging due to extreme cost, size, and power constraints. Our low-cost *SecureCPS* architecture dramatically improves security, to include adapting and preempting most attacks in a nanosatellite. *SecureCPS* also applies to a broad class of cyber-physical systems (CPS), such as aircraft, cars, and trains. This paper focuses on Embry-Riddle's ARAPAIMA nanosatellite architecture, where we assume any off-the-shelf component could be compromised by a supply chain attack.[1] Based on these assumptions, we have used Vanderbilt's Cyber Physical - Attack Description Language (CP-ADL) to represent realistic attacks, analyze how these attacks propagate in the ARAPAIMA architecture, and how to defeat them using the combination of a low-cost Root of Trust Module, Global InfoTek's Advanced Malware Analysis System (GAMAS), and Anomaly Detection by Machine Learning (ADML).[2] Our most recent efforts focus on documenting the design of *SecureCPS*.

Keywords: cyber-physical system, security, nanosatellite, machine learning, root of trust, malware, advanced persistent threat, supply chain injection

1. INTRODUCTION

Maneuvering thrusters have traditionally allowed large satellites to dramatically increase the range of missions they can accomplish. Now, inexpensive nanosatellite designs include maneuvering thrusters for similar purposes. Unfortunately, if their command and control system is compromised, even the smallest nanosatellite can become a threat to space-based High-Value Assets (HVA). Securing nanosatellites which require a maneuvering capability is challenging due to the extremely low cost, size, and power constraints inherent in a nanosatellite design. However, because a nanosatellite can threaten an HVA, it must provide a level of security typically reserved for HVAs, at least for the functionality of a nanosatellite that might threaten an HVA.

Similar problems are prevalent in many other systems that combine propulsion systems and computer control, such as automobiles, trains, and aircraft. As a result, providing a low-cost architecture to secure a nanosatellite propulsion system against cyber-attack, would also benefit a broad class of cyber-physical systems (CPS).

For our research, we focus on the ARAPAIMA nanosatellite under development at Embry-Riddle Aeronautical University (ERAU). We also assume any commercial off-the-shelf (COTS) component used in ARAPAIMA could be compromised (e.g., supply chain attack). Based on these assumptions, we analyze how such attacks affect the security mechanisms needed to secure the ARAPAIMA architecture. The systematic approach used for this analysis is presented along with the Cyber-Physical Attack Description Language (CP-ADL) used to formally describe the attack scenarios.

[1] For more information about ARAPIAMA, go to http://eraucubesat.org/..
[2] For more information about GAMAS, go to www.globalinfotek.com/gamas.htm.

The SecureCPS architecture we propose to secure ARAPAIMA is divided into three main components. First, a low-cost radiation hardened trusted module provides the "root of trust" from which all operations must derive their approval to operate. In this paper we describe the properties of this module. Second, we propagate and extend the trust relationship from the "root of trust" module for all ARAPAIMA architectural components using Global InfoTek Inc.'s Automated Malware Analysis System (GAMAS). Once trust has been extended to GAMAS, GAMAS then validates and ensures other software in the system is trusted prior to execution by employing a low-level driver and library of known-good signatures. GAMAS also provides low-level instrumentation to support our Anomaly Detection by Machine Learning (ADML) subsystem. To ensure the runtime GAMAS has not been corrupted (e.g. radiation event), each instance is validated by the trusted module on a recurring basis. Third, past history shows known good software can perform malicious actions, so we employ machine learning in the ADML subsystem to learn models of normal behavior on the ground, so we can identify anomalous behavior in orbit. This paper describes the SecureCPS architecture, the individual components of SecureCPS, and how CP-ADL will support scenario-based testing of the entire SecureCPS system.

2. ARAPAIMA: A REPRESENTATIVE NANOSAT ARCHITECTURE[3]

The ARAPAIMA nanosatellite is a three-axis stabilized satellite, of 11x26x34cm overall dimensions, that classifies as a 6U CubeSat. The launch mass is 11 kg which includes 1 kg of propellant and generous component and subsystem margins. Similar to larger satellites that perform rendezvous and proximity operations, ARAPAIMA has a full set of subsystems that enable it to perform its mission. A functional diagram of ARAPAIMA is described in detail in Appendix A and shown in Figure B-1. The most important difference between ARAPAIMA and a traditional satellite is the use of COTS components. Another difference is the lack of redundancy in some areas. The use of COTS components and lack of redundancy reduce the overall cost of the mission, however, they create cybersecurity concerns that are addressed by the work presented in this paper. A full description of the ARAPAIMA architecture is presented in Appendix A. The following paragraphs present the propulsion system, which is the focus of our SecureCPS work.

ARAPAIMA's propulsion subsystem consists of eight reaction control system (RCS) and one orbital maneuvering thruster (OMT). The RCS controls torques in positive and negative directions about each body axis. During orbital maneuvers the attitude control is performed in closed loop with the RCS thrusters, which are commanded with pulse width modulation (PWM) signals.

The command and data handling (C&DH) subsystem consists of an S-band radio, two patch antennas, and an on-board computer (OBC). The L-3 Cadet radio was chosen because it fits the ARAPAIMA's 6U form factor and uses relatively low power. It already has flight heritage on the CINEMA mission that was launched on Sept 13, 2012.[4] The radio operates in the S-band for downlink, and the UHF-band for uplink, it utilizes a half-duplex architecture, and it is capable of providing up to 10 Mbps downlink and 250 kbps uplink. The radio has a large data buffer, 4GB, which makes it a good choice for a store and forward communications architecture. The data buffer of the radio can store, for example, over 1000 images of 640x 512 pixels taken at 12 bit depth with the IR camera. It can also store a significant amount of health and status data, which becomes important for our approach to anomaly detection, which is described in Section 3 below. The radio supports the commercial grade Advanced Encryption Standard (AES) with a 256 bit key. For the time being the patch antennas are placed on the deployable solar panels.

The payload and on-board computer modules are connected to a radiation hardened FPGA, which also provides the SecureCPS "root of trust", which is described in more detail in Section 3. In addition, the FPGA runs a simple but robust supervisory routine and safe mode controller and periodically stores the state vector of the nanosat subsystems. In the event of a fault in the C&DH computer, the supervisor on the FPGA triggers the safe mode controller and attempts to restart the C&DH computer. If the C&DH computer cannot be started, the supervisor transfers the control of the nanosat to the payload computer which loads the flight control software.

[3] For more information about ARAPIAMA, go to http://eraucubesat.org/.

[4] https://directory.eoportal.org/web/eoportal/satellite-missions/c-missions/cinema

2.1 ARAPAIMA Operational Modes Overview

The complex ARAPAIMA mission and sporadic access to ground control dictate some level of autonomy. Autonomy is provided by a finite state machine. At any time during the mission, the nanosatellite is in a specific operational mode which is a superstate of a finite state machine. Within each mode the status of each active component of each subsystem is tracked together with the state vectors of the orbital and attitude dynamics of the nanosatellite.The modes are designed to satisfy both operational requirements and operational constraints and allow the nanosatellite to perform some tasks without the intervention of ground control. All nominal and off-nominal entry and exit conditions for each mode are predefined. The transition between modes is either time based or event based. For example, the simplest mode describes the nanosatellite in its launch canister. The entry to the canister mode takes place when the technicians insert the nanosatellite in the launch canister and latches the canister door. During the store mode. the main power switch is in the off position and all the nanosatellite components are off. The nominal exit from the store mode takes place when the nanosatellite is ejected from the launch canister and a change in voltage is detected in the separation mechanism.

In contrast, the delta-vee mode is a complex mode in which the nanosatellite fires the OMT to perform an orbital maneuver. The entry condition for this mode is the receipt of a command from the ground to prepare for the delta vee mode. Once the nanosatellite enters the delta-vee mode, it starts a message exchange with the ground station. The message exchange consists of orbital maneuver profiles uploaded from the ground station and messages from the nanosatellite that it has validated the maneuver profiles on-board and is ready to proceed. A maneuver profile can be a simple station keeping maneuver or a complex "teardrop" maneuver with respect to its target Remote Space Object (RSO). Exit from the delta-vee mode takes place when the orbital and attitude state vectors are verified at the end of the maneuver. The verification of the maneuver is performed on board using the sensors of the attitude and orbit control system and the results are also verified by the ground controllers. On-board state propagation is used to perform a deterministic validation of the maneuver, which is then combined with a statistical approach based on covariance propagation that establishes a metric that is used both the operational control software and by the SecureCPS ADML to detect anomalous behavior. ADML is discussed in more detail below.

Off-nominal exits from a mode trigger the safe mode, which limits the nanosatellite to "stay alive" operations, such as generating power, avoiding large thermal gradients, and communicating with the ground. The safe mode can also be triggered by flags (events) generated by GAMAS. A graceful degradation of performance is designed into the safe mode that turns off components one by one and periodically checks , either fully or periodically, the conditions which triggered the transition to the safe mode.

3. SECURECPS ARCHITECTURE

The three major subsystems of SecureCPS are the Root of Trust (RoT), GAMAS, and ADML (Anomaly Detection by Machine Learning) subsystems. These subsystems are highly interdependent and, while each has specific interfaces with the ARAPAIMA architecture, they are designed to operate within a much broader class of resource constrained cyber-physical systems (CPS). This section will describe each of the three SecureCPS subsystems in more detail.[5]

3.1 Root of Trust

Complete "chain-of-custody", where all aspects of hardware and software are carefully controlled, was once, and still is for niche products, an effective means of safeguarding component provenance. However the feasibility of such an approach, given the existing economic climate, is no longer an option due the pressure of time to market or cost effectiveness requirements. Only a few niche companies and government organizations can afford to control the chain-of-custody for a system. Even then, only critical subsystems receive this level of control. For example, most modern devices use an operating system with published APIs (e.g., Linux). Economies of scale make multi-purpose integrated circuits (e.g., Intel x86), development boards, and software more capable than a custom solution at lower cost. However, using these COTS components risks having malware embedded deep within a system's software or hardware

[5] For more information about GAMAS go to www.globalinfotek.com/gamas.htm.

and the associated, potentially catastrophic, consequences. In the case of ARAPAIMA, such a compromise could convert a benign nanosatellite into a hazard to other space systems.

When developing our design, we examined various commercial trusted modules such as HP, Dell and Amtel's Trusted Platform Module (TPM) and other types of BIOS level protection, but we determined that technology was too specifically tailored to either designed for digital rights management or enterprise asset tracking. Furthermore, we provenance of these TPM was questionable. For SecureCPS we need a cost effective component with a well-established "chain-of-custody" and specific features designed to support GAMAS as it extends trust throughout the entire nanosatellite system.

The SecureCPS Root of Trust (RoT) is a cost effective complete "chain-of-custody" device that is small, simple, verified hardware component providing the basis for trust that is extended throughout the ARAPAIMA nanosatellite architecture by GAMAS. The RoT assumes any nanosatellite subsystem could be compromised. The RoT is the only trusted sub-system from which GAMAS verifies its own execution and then extends trust throughout the SecureCPS architecture. The RoT then becomes the basis for:

1. Check the Checker: The RoT ensures that GAMAS is a known good binary and periodically checks the operation of GAMAS to ensure it is not subsequently compromised.
2. Binary Signatures: The RoT provides known good signed binary signatures to GAMAS. These signatures are used by GAMAS to allow good (i.e., white listed) binaries to run and prevent the execution of all other binaries. For enterprise applications, GAMAS has the ability to use white lists, black lists, and

We are developing the SecureCPS RoT in two phases. The first phase uses a general purpose computer to provide low-cost emulation of the RoT. This provides a flexible development platform as SecureCPS, and future applications of SecureCPS evolve. The second phase implements a cost effective hardened version of the RoT (e.g., FPGA or ASIC) based on the emulated design. Similar to DARPA's SHIELD program SecureCPS studies techniques to extend trust throughout ARAPAIMA. However, SHIELD focus on the miniaturization of the integrated circuit for use in hardware devices, while we focus on a complete trust solution that includes software running on untrusted parts on a scale suitable for a nanosatellite and similar cyber-physical systems.

3.2 GAMAS

GAMAS is low-level software that extends trust throughout the nanosatellite architecture. GAMAS ensures all programs running on all nanosatellite computers are valid by comparing signed hashes of the binary to the hash provided by the RoT. If a program's hash is not identical to the hash stored in the RoT, or a program attempts to run in a mode where it is not expected, GAMAS prevents program execution before it begins. GAMAS also provides low level instrumentation of the ARAPAIMA's use of APIs, thus providing the data needed for the SecureCPS Anomaly Detection by Machine Learning (ADML) subsystem described in the next section.

The following observations provide the foundation for the GAMAS design:

1. Whether injected from software or hardware, an advanced persistent threat (APT) must interact with the operating system and system libraries via actual code to pose a threat. Thus any APT must execute some code.
2. An APT leverages existing operating services (e.g., *sys_open*) because rebuilding such services is either impossible or, when possible, would create a prohibitively large binary.
3. Based on observation 1 and 2 above, the behavior of an APT can be observed by instrumenting key points (e.g. system traces, library traces) without the need for full binary inspection.
4. The information obtained at these key points, in addition to data provided by the CPS itself, is sufficient to detect most, if not all, anomalous behavior based on known good software (i.e., good software gone bad) -- this data is provided to the SecureCPS ADML.
5. GAMAS integrity will be confirmed by relying on observations 1, 2, and 3 to check itself, preparing GAMAS to extend trust and check other ARAPAIMA executables.

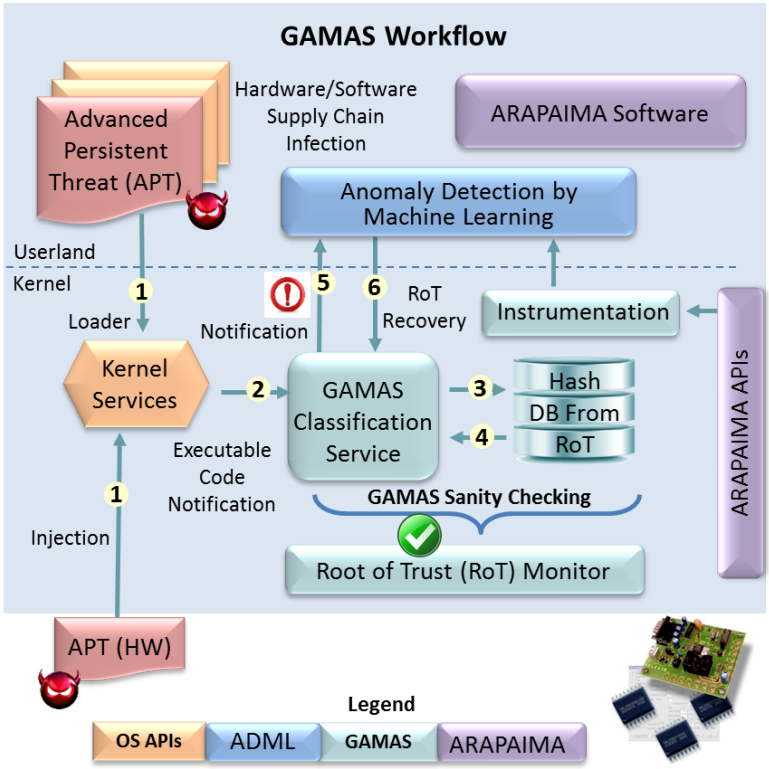

FIgure 3-1: GAMAS within the SecureCPS Architecture

As shown in Figure 3-1, GAMAS has three distinct subsystems:

Root of Trust Monitor: The Root of Trust Monitor implements a cryptographic challenge response mechanism for the purpose of handshaking between the GAMAS components. GAMAS RoT monitor achieves this goal by performing a calculation of the challenge, the GAMAS executable, and the data it stores. If the answer to the challenge is correct, the hardware RoT confirms that GAMAS is trusted, and can extend the trust relationship to the whole system. In the event that GAMAS has been altered, the computed challenge response would fail and, depending on the nature of the failure, ARAPAIMA would enter safe mode and send an alert to the ground. This process runs continuously to ensure continued trust.

GAMAS Classification Service: Once GAMAS is verified, the GAMAS Classification Service ensures ONLY known good executables are allowed to run. Signatures for "known good" executables are provided by the hardware root of trust (updatable with the correct authentication sequences).

GAMAS Instrumentation: The third, and independent, feature is the GAMAS instrumentation. It captures detailed data about the use of ARAPAIMA resources analogous to a "black box" flight recorder. The most recent GAMAS Instrumentation data are made available for use by the ADML and transmission to the ground. A persistent flight log, as opposed to a data stream, provides a loose coupling between GAMAS and the ADML, so the ADML can perform anomaly detection when resources are available. In some ARAPAIMA modes, very little compute resources will be available, so ADML processing must wait. This loose coupling also allows the L-3 Cadet radio to downlink flight log data when ground contact is available.

3.3 Anomaly Detection by Machine Learning (ADML)

3.3.1 ADML Overview

GAMAS ensures only known good binaries run during each mode of the nanosatellite operation. However, good software can be subverted and do bad things – consider the email program subverted to send a virus to everyone on your contact list. Your email program is expected to run and send email messages, but sending such a large number of emails in a short period of time would be suspicious, if you were aware of it. GAMAS provides the instrumentation to ensure such an event would be observed and ADML identifies anomalous behavior.

To detect "good software gone bad", *SecureCPS* employs machine learning. Unlike a general purpose computer (e.g., desktop, laptop), a cyber-physical system, like a nanosatellite, car, train, or aircraft, runs a more limited set of software and it runs it in a much more predictable manner. Much like completely depressing the accelerator of our car (i.e., flooring it) is unusual, and "flooring it" for thirty minutes would likely cause engine damage, not to mention safety issues, there are predictable limits for ARAPAIMA's propulsion commands. Using statistical models, *SecureCPS* can build (i.e., learn) these models during ground-based simulation. In addition, SecureCPS can learn models for normal communications on the internal and external communications systems, and commands to the sensors.

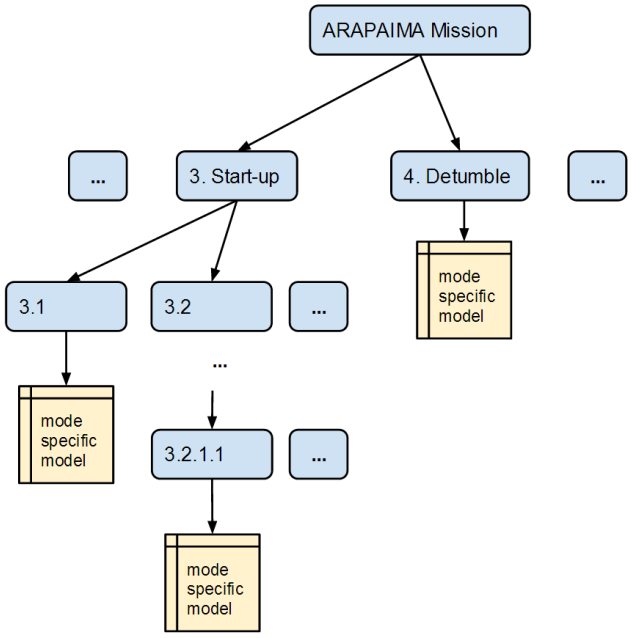

Figure 3-2. Mode-Specific Machine Learning Models

To increase the accuracy (i.e., F-measure) of our learned models, which means less nominal activity is flagged as anomalous (i.e. false positives), while less anomalous activity is flagged as normal (i.e., false negatives), *SecureCPS* will build mode-specific models of nominal activity. Mode-specific models allows each model to more simply and accurately model the activity in each mode. Specifically, mode-specific models reduce the time required to learn a model, reduces the time required to test the model against current activity during on-line use, and makes each model more accurate.

Reducing Learning Time: Anomaly detection requires an an accurate model of nominal activity. Whatever is not nominal is anomalous. For complex systems, such as a nanosatellite, this model must include a large number of contextual factors because what is nominal in one context is anomalous in another context. Rather than forcing the machine learning system to learn all of this contextual information, SecureCPS uses the modes typically found in

cyber physical system to simplify the machine learning problem. As shown in Figure 3-2, we learn a model of nominal activity at the most specific mode possible. In essence, this tree of modes explicitly captures much of the contextual information, so the machine learning system does not have to infer it from the data. This dramatically reduces the time required to learn a model of nominal ARAPAIMA activity.

Reducing Run Time: A simpler statistical model equates to simpler computation, and less load on the OBC. As compared to a single model for the entire ARAPAIMA system, each mode-specific model is simpler. Each of these simpler model requires less storage and less processing to classify the activity reported to it.

Increasing Accuracy: The accuracy of a statistical model depends upon factors like the number of samples (i.e., the law of large numbers), the number of dimensions (i.e., the curse of dimensionality), and the complexity of boundary between nominal and anomalous (e.g., partitioning a high-dimensional space). All such factors affect the accuracy of a learned model -- less samples reduce accuracy, greater dimensions reduce accuracy, and complex relationships reduce accuracy. While this list is not exhaustive, it demonstrates why the simplicity of mode-specific models reduces the impact of such factors to improve overall SecureCPS ADML accuracy.

Efficiently running a machine learning algorithm is critical, given the limited processing power available on a nanosatellite like ARAPAIMA. However, the availability of computer resources will vary greatly, depending on the current ARAPAIMA mode. For example, during imaging and station-keeping, the OBC is expected to have little, if any additional bandwidth for ADML. During the "start-up" or "trailing" modes, extensive resources should be available to look for anomalous activities. Because the ADML is loosely coupled through a persistent record of recent flight events, the ADML can look back through history for past anomalous activity in order to "catch-up" with current status, when compute resources are available.

If extremely resource intensive machine learning is desirable, models which ARAPAIMA simply cannot support, the ADML will gathering data (to include filtering, statistical sampling, and summarization) and send it to the ground. On the ground, computer resources are relatively plentiful, so complex multivariate models can be employed to detect far more subtle anomalies. One of the goals for the next phase of our research and development is measuring the compute loads for each mode of ARAPAIMA operation and determining what modes can support on-board statistical anomaly detection, and which cannot, and what amount of computer resources are available in each operating mode. These measurements will drive our selection of machine learning algorithms and where they run (i.e., on-orbit, ground).

If possible, we prefer on-orbit anomaly detection because it can immediately flag an anomaly and ARAPAIMA can take action. Ground-based anomaly detection incurs the delay associated with a transmission to the ground and transmitting a response to ARAPAIMA on-orbit.

3.3.2 ADML Example

An example will help illustrate the function of the ADML and how interacts with ARAPAIMA and GAMAS to detect anomalous behavior. While our example will focus on detecting anomalous orbital maneuvering due to malicious intent. this is by no means the only type of anomaly ADML will learn to detect. This particular type of anomaly merely serves to illustrate an approach that will be applied to many areas within our nanosatellite. Other areas of application include areas as diverse as abnormal volumes of bus activity, abnormal levels of computer processing activity, RAM utilization, and sensor activity.

The example focuses on the orbital maneuvering of the nanosatellite. The orbital maneuvers are performed by turning on the OMT and by pointing the nanosatellite such the thrust of the OMT is applied in the direction required for the maneuver. The OMT is commanded on for a certain amount in time, i.e., the application of the thrust force is performed in open loop. On the other hand, the pointing of the nanosatellite during the orbital maneuver is performed in closed loop. The RCS thrusters deliver torques to point the nanosatellite in the required direction and to reject any disturbances that might affect the pointing. A maneuver profile is planned on ground by solving the non-linear 6 DoF equations of motion or, if appropriate, by solving the equations of motion (EoMs) linearized about a certain equilibrium point or trajectory. The maneuver is tested either by Monte Carlo analysis, in the case of the non-linear EoMs or by covariance analysis and propagation in the case of linear EoMs. The ADML will employ the results of the tests to

determine the nominal manifold of the state variable (x, y, z, v_x, v_y, v_z). In practice, we may add states such as acceleration, derivatives of acceleration (i.e., jerk) for translational motion, and structural bending modes and propellant slosh for rotational motion. We also represent our propulsion commands in a simple manner (i.e., p_x, p_y, p_z) the components of the OMT thrust vector on the reference frame in which the maneuver is specified..

Ground testing provides a large collection of statistically significant data suitable to modeling the range of nominal ARAPAIMA motion. However, trying to accurately model all possible ARAPAIMA motion in a single model would be challenging. Additional information provided by GAMAS simplifies the modeling problem. By virtue of GAMAS' instrumentation of the interfaces to the C&DH computer, GAMAS can tell ADML the type of maneuver associated with the delta-vee command. This gives ADML the opportunity to load a simpler model that is specific to the maneuvering mode identified by GAMAS Instrumentation. While we have not yet reviewed ARAPAIMA ground test data, a high-dimensional manifold describing the boundaries of nominal motion and delta-vee commands is one way to model nominal activity. Such a model could be learned by an artificial neural network (ANN), where all of the dimensions (state and command) would be real-valued inputs and the indicator of nominal or anomalous would be a single value. While we have not settled on a neural network, an ANN is well-suited to learning complex boundaries in a high-dimensional real-valued space. The ANN weights would be learned on the ground during simulation (i.e., back propagation) and used in orbit to transitions outside the nominal state-space manifold (i.e., forward propagation and classification). One of the advantages to having a model for each ARAPAIMA operating mode is the ability to pick entirely different types of models for each mode, if deemed appropriate.

A reader unfamiliar with the differences between control and anomaly detection might think we are merely running two control systems and comparing their results, but this is not the case. While it is possible that the control system and anomaly detection system might run the same class of algorithm, the filters would have fundamentally different goals. The control system must use the current state and the desired state to issue propulsion commands. In practice, the control system cares very little about the probability that a particular state occurs, it simply must respond and attempt to eliminate the difference between current and desired states. In contrast, ADML does not generate propulsion commands from the current state and desired state. Instead, it only needs to know if a particular combination of current state, desired state, and propulsion commands are nominal or anomalous. In terms of the functionality, as shown below, the control system computes the following function.

$$p_x, p_y, p_z = F_{Controll}(x, y, z, v_x, v_y, v_z)$$

While the ADML computes this function

$$P(\text{anomaly} \mid x, y, z, v_x, v_y, v_z, p_x, p_y, p_z) = F_{ADML}(x, y, z, v_x, v_y, v_z, p_x, p_y, p_z)$$

This makes the design of the control system and the ADML fundamentally different. If these two systems were focused on the cruise control of a car, the control would focus on maintaining the set speed, while ADML determines the probability that the combination of set speed, acceleration commands, and gear are nominal.

4. USING ATTACK SCENARIOS TO EVALUATE SECURECPS

The design of SecureCPS is driven by realistic attacks on the ARAPAIMA architecture. One of our scenarios is outlined in Section 4.1. In order to capture an unambiguous representation of each attack scenario, as well as establish a basis for automated testing, we describe our scenarios in Cyber-Physical Attack Description Language (CP-ADL). Section 4.2 provides a CP-ADL example which describes one attack scenarios with no defensive measures installed, and the same attack scenario with SecureCPS installed.

4.1 Possible Attack Scenario

For the purposes of this attack scenario, we assume malicious code is present in the Linux Real-Time OS (Linux RTOS). Given the size of the Linux OS, hidden code is a serious threat.[6] The hypothesized APT in the OS has been added by a supply chain injection (e.g., an open source contributor). Both of ARAPAIMA's PC-104 computers, the Payload and Command and Data Handling (C&DH) computers, run the same Linux RT OS.

As part of its nominal mission, ARAPAIMA collects images of remote space objects (RSO). Some preliminary processing (e.g., compression) is performed on-board ARAPAIMA before this data are sent to the ground. In our scenario, malicious code in the OS waits for one of two trigger events.

> The first trigger event (i.e., a QRC in an image acquired by ARAPAIMA) is designed to work on-board the Payload Computer. If a QRC were displayed on the side of an RSO, then it would appear in an image and be available to the malicious code in the OS. If the malicious code detects the QRC, a Trojan is unpacked, executed, and a predefined packet is transmitted on all available interfaces.
>
> The second trigger is a predefined network packet. In response to the predefined packet sent by the malicious software on the Payload Computer, a second copy of the same implant in the Linux RTOS on the C&DH is started. This network packet triggers the same malware that was triggered by the QRC.

In response to the second trigger event, the C&DH computer uses its access to the L-3 Cadet Radio to install a new cryptographic key. Installation of this new key denies access to the ARAPAIMA ground control team and provides secure access to a malicious ground station. This change of access allows the malicious ground station to completely reprogram the ARAPAIMA system, if desired, and redirect ARAPAIMA as desired.

The description of the attack scenario provided above is easy for a human to understand. However, in order to describe this scenario in sufficient detail to develop instrumentation, detection, and prevention methods, we need to describe it in a more explicit, structured, and machine-readable manner. All relevant aspects of the attack scenario must be reflected in this description. The schema for this representation is provided by Vanderbilt's Cyber-Physical Attack Description Language (CP-ADL).

4.2 Description Language

One of the most distinctive properties of cyber-physical system attacks is the cause-effect causality chains that cross cyber-physical boundaries. Therefore, any description language for CPS must necessarily reflect such cross-domain effects – alongside single-domain cause-effects – in the description of relevant attack stages. For this purpose, we structure attack description according to the dimensions of the taxonomy capable of covering such cross-domain aspects [1] (see Figure 4-1). In this taxonomy, the tuple (*Influenced Element*; *Influence*) represents the cause and the tuple (*Affected Element*; *Impact*) represent the effect of the attack stage. In the case when the taxonomy is used to describe the defensive measures, these tuples describe changes introduced by the measures and their effects respectively. Dimensions *Method* and *Preconditions* describe means used to exhibit influence on the influenced element as well as conditions under which these means can be executed, and under which this leads to described effects.

[6] Ars Technica reported the core of the Linux 3.0 OS contained 15 million LOC in 37,000 files from 1,300 developers. On-line publication,
http://arstechnica.com/business/2012/04/linux-kernel-in-2011-15-million-total-lines-of-code-and-microsoft-is-a-top-contributor/

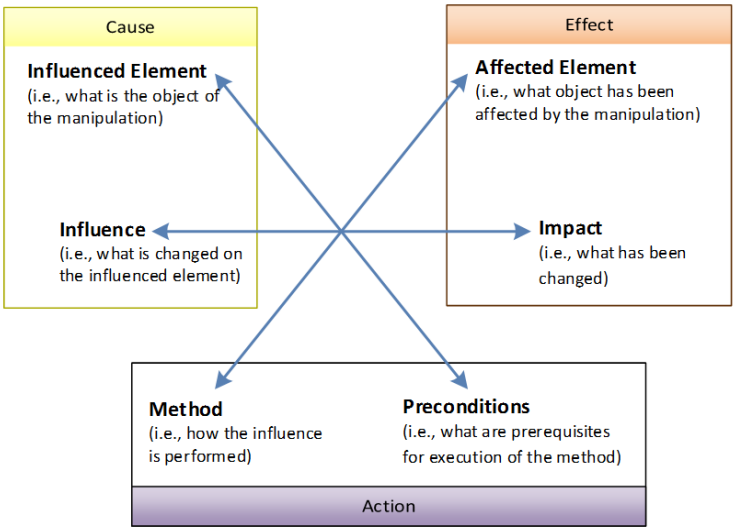

Figure 4-1: Taxonomy for description of cross-domain attacks on CPS (Based on [1])

The last two stages of the attack scenario described in Section 4.1 are presented in Table 4-1 and Table 4-2. This is still a human readable description. Nevertheless, this description presents a clear separation in attack stages, and explicitly describes information required by the taxonomy dimensions.

Table 4-1: Stage 8, Overwrite keys of UHF radio

Influenced Element	L-3 Cadet UHF Radio
Influence	Overwrite keys, which are used for secure communication
Affected Element	ERAU team
Impact	Locked out of communication with the UHF radio
Affected Element	Malicious ground station
Impact	Malicious ground station can communicate with and control ARAPAIMA.
Method	Malware commands Cadet Radio to overwrite the benign keys with malicious keys
Preconditions	- Malicious implant on the OBC is activated - OBC has access to Cadet UHF

As shown in Table 4-1, the attack produced multiple effects on different Affected Elements. It also created the preconditions necessary for the follow-up attack stage described in Table 4-2. These, as well as additional complex relationships between taxonomic dimensions, cannot be easily reflected in the table-like form. For this reason, we use the Cyber-Physical Attack Description Language (CP-ADL) [2] taxonomy. Among others, CP-ADL enables us to define relationships between attack aspects as well as reuse different attack description parts. For instance, different attacks can lead to the same consequences which can be "reused" in the description of two or more different chains of events.

Table 4-2: Stage 9, Adversarial control of satellite

Influenced Element	ARAPAIMA

Influence	Receive valid navigation commands from malicious ground station
Affected Element	ARAPAIMA
Impact	On a collision course with a HVA
Method	The malicious ground station commands ARAPAIMA to change orbits to threaten a HVA
Preconditions	Malicious ground station can communicate with and control ARAPAIMA

```
▼<Tuple_Targets_Attacks>
  ▼<Targets>
    ▼<Tuple_InfluencedElement_Influence>
      ▼<InfluencedElement>
          <Category>Physical</Category>
          <Name>Satellite/PropulsionSystem</Name>
        </InfluencedElement>
      ▼<Influence>
          <Type>StateChange</Type>
          <Description>Satellite/PropulsionSystem/Thrusters/Activation</Description>
        </Influence>
      </Tuple_InfluencedElement_Influence>
    </Targets>
  ▼<Attacks>
    ▼<AttackOption>
      ▼<Tuple_Means_Preconditions>
        ▼<Means>
            <Category>Physical</Category>
            <Description>Satellite/PropulsionSystem/Thrusters/Activation</Description>
          </Means>
        ▼<Preconditions>
```

Figure 4-2: Example of a machine-readable attack description in CP-ADL, excerpt

The machine readable data format (XML) for CP-ADL bridges the gap between human and machine readable versions of the attack scenarios. One goal of this program is using CP-ADL to help describe and design attack scenarios and implement an automated test capability. In later phases of the SecureCPS project, we anticipate a red team describing an attack as shown above, representing that description in CP-ADL XML, then using the XML to run the scenario on our SecureCPS test system. We also anticipate the use of CP-ADL to determine if anomalous activity fits one of our attack definitions. Matching anomalous activity with a specific attack definition provides a means to differentiate between attacks and other types of anomalous activity (e.g., hardware failure).

4.3 Attack Prevention with SecureCPS

We reuse the same taxonomic structure used in CP-ADL for the description of the countermeasures executed by SecureCPS. The reason is that the countermeasures follow the same patterns: countermeasures (described in the "Method" field) are triggered by a combination of preconditions. These countermeasure cause-effect chains (in the simplest case, only one stage) are described in the remaining four dimensions.

The evolution of the scenario outlined in Section 4.2 only considers functional architecture of the satellite in the case where no defensive measures are implemented. Possible impacts of this and other attacks are dramatic. As shown in Table 4-3, the situation changes when SecureCPS is installed on the satellite (see Table 4-3).

Table 4-3: Attack Prevention with ARAPAIMA/RoT

Influenced Element	C&DH

Influence	Patch uploaded by the ground station is applied.
Influenced Element	RoT
Influence	Hash tables updated by the ground station
Affected Element	C&DH
Impact	Malicious payload removed
Affected Element	RoT
Impact	Contains new hash tables
Method	- Bootstrap routine applies patch uploaded by the ground station to the C&DH. - Bootstrap routine updates the RoT hash tables uploaded by the ground station.
Preconditions	- Bootstraps routine is active. - Update patch is received. - ARAPAIMA receives update command from the ground.

With SecureCPS in place, the malicious behavior of the kernel is detected by GAMAS. GAMAS raises the flag and the nanosatellite enters Safe Mode, during which it downlinks the most recent Flight Recorder data (e.g., GAMAS Instrumentation, ADML analysis) associated with this interrupt. ARAPAIMA ground control analyzes the data, attempts to identify the source of the anomaly and prepares a patch fixing the identified nanosatellite problem. Before the patch is transmitted to the nanosatellite, it is extensively tested on the ground on the hardware-in-the-loop engineering qualification model, in addition to computing a new signed hash table for the RoT.

Once the tests are successfully completed, the ground station uploads the patch along with the signed hash table, which are updated in the RoT module. GAMAS applies the patch during a bootstrap mode. After the update is verified and upon confirmation from the ground station, the nanosatellite resumes nominal operation.

This scenarios assumes that both malware and GAMAS are asynchronously executing on different hardware and software modules. This means that although the hypothetical APT could be executing at different points in time, the injection mechanisms are prevented by SecureCPS on any computer when they attempt to execute.

5. CONCLUSIONS

The growing role of industry in near-earth space as well as the growing use of off-the-shelf hardware and software is inevitable. The uncontrollable supply chain of such off-the-shelf systems makes injection of malicious payloads possible, even likely, which threatens the security of commercial space systems, small or large. However, the low-cost objectives of small space systems make them particularly vulnerable.

SecureCPS provides a cost-efficient model for securing a satellite against attacks which could make a satellite a threat to other orbital systems, despite malicious payloads in one or more COTS components. SecureCPS derives its security from the Root of Trust (RoT), and Global InfoTek Inc.'s Automated Malware Analysis System (GAMAS) modules propagates this security throughout the ARAPAIMA architecture. For the case where security detects a problem, which cannot be overcome by more precise measures, GAMAS secures the load of updated software. In the worst case, where repeated attempts to restore the system fail, GAMAS ensures ARAPAIMA is disabled. This multi-layer protection mechanism ensures that the satellite will not create a danger to HVAs in near-by orbits.

Our SecureCPS solution is cost-efficient and is applicable to other classes of cyber-physical systems. Similar problems are prevalent in many other systems that combine propulsion systems and computer control, such as automobiles, trains, and aircraft. As a result, providing a low-cost architecture to secure a nanosatellite propulsion system against cyber-attack also benefits a broad class of cyber-physical systems (CPS) across a broad set of defense, intelligence, and consumer industries.

We cannot precisely determine the cost of SecureCPS, because our design is not complete. However, in terms of the per-unit cost, where a nanosatellite like ARAPAIMA would be a unit, we expect the RoT to be the most expensive per-unit subsystem. Our goal is a very simple hardened ASIC or small FPGA, which should minimize the cost of this subsystem. Even for the most expensive ASIC, a dramatic cost savings is realized. If the ARAPAIMA software had to be developed in a trusted and secure environment, the cost would be many times greater than an ASIC, and the trusted software would still require a root of trust as protection against radiation events -- probably ending maneuvering nanosatellite programs because the cost would be too high to justify the limitations of the platform. By permitting the use of software developed in an untrusted environment, SecureCPS dramatically reduces the cost of a cyber-physical system like ARAPAIMA, yet provides a level of security, in terms of the threat posed to other orbital systems, normally associated with large national systems.

6. ACKNOWLEDGMENTS

This work is sponsored by the Air Force Research Laboratory (AFRL) as part of the Department of Defense (DoD) Small Business Technology Transfer (STTR) program.

REFERENCES

[1] Yampolskiy, M., Horvath, P., Koutsoukos, X. D., Xue, Y., Sztipanovits, J., "Taxonomy for description of cross-domain attacks on CPS", Proceedings of the 2nd ACM International Conference on High Confidence Networked Systems, pp. 135-142, (2013).

[2] Yampolskiy, M., Horvath, P., Koutsoukos X. D., Yuan Xue, and Janos Sztipanovits, "CP-ADL: Cyber-Physical Attack Description Language", submitted to International Journal of Critical Infrastructure Protection (IJCIP).

[3] Yampolskiy, M., Horvath, P., Koutsoukos, X. D., Xue, Y., Sztipanovits, J., "Systematic analysis of cyber-attacks on CPS-evaluating applicability of DFD-based approach", Proceedings of the 5th International Symposium on Resilient Control Systems (ISRCS), pp. 55-62, (2012).

Appendix A - Additional Information about the ARAPAIMA Nanosatellite

During nominal operations, ARAPAIMA uses its star tracker (STR) for attitude determination and a set of three reaction wheels (RWs) for attitude control. The RWs produce attitude control torques for both slewing the nanosat and to reject perturbations due to aerodynamic forces, gravity gradient, and solar radiation pressure (SRP). The reaction control system (RCS) thrusters produce torques to unload the reaction wheels. The Blue Canyon STR and the RW-0.03-4 reaction wheels (RWs), from Sinclair Interplanetary seem, for the time being, to offer the best price-performance combination on the market. The manufacturer quoted accuracy of the star tracker is 7 arcsec normal to the STR optical axis (cone angle) and 70 arcsec around the optical axis (clock angle). In the current configuration the STR baffle provides an exclusion half-angle of 30 deg. The STR is installed so that its optical axis is anti-parallel with the optical axes of the payload instruments.

The instruments of the payload are an MLR2K rangefinder from FLIR Systems Inc., a short wave (SW) IR camera from Goodrich Aerospace Inc., and a monochrome visible spectrum camera from SenTech, Ltd. Due to the high computational load of the payload it has been decided to fly a dedicated payload computer based on an Intel 32-bit processor. Two frame grabbers one for each camera, will be used to connect the cameras to the payload computer module.

In the current design the CPU modules are connected to a radiation hardened FPGA on which components of the root of trust are implemented. In addition, the FPGA runs a simple but robust supervisory routine and safe mode controller and periodically stores the state vector of the nanosat subsystems. In the event of a fault in the C&DH computer the supervisor on the FPGA triggers the safe mode controller and attempts to restart the C&DH computer. If the C&DH computer cannot be started, the supervisor transfers the control of the nanosat to the payload computer which loads the flight control software.

The electrical power subsystem (EPS) of the MkX configuration consists of five solar panels, a battery, and an electrical power conditioning/battery charging unit. Two large deployable solar panels of 20x30 cm carry 20 solar cells each and two smaller deployable solar panels carry four solar cells each. The two smaller deployable solar panels also serve as aperture covers for the payload instruments and the star tracker when the nanosat is in storage and during launch and detumble. A power regulation/battery charging module and two 30 Wh batteries complete the EPS. The solar panels, power module, and the batteries will most likely be procured from Clyde Space. The thermal control subsystem has not been specified during the first design iteration. The specification of its requirements are pending consolidation of the design of the nanosat. However, it is expected that it will consist of passive thermal control. such as insulators and paints and possibly heaters for the control of the temperature for sensitive components such as batteries.

During the communication phases of the mission the nanosat will be commanded such that the face with the antennas points in the nadir direction. To close the communication link a ground station with an unsophisticated S-band antenna can provide a bandwidth of up to 10kbps to a small patch antenna in LEO. A more advanced ground station with a 1 m parabolic dish can provide bandwidths of 1 Mbps and better. The OBC is the same model as the payload computer. The choice reduces the risk for the on-board software development (OBSW) and increases the reliability of the computer architecture by providing the means to implement dual computer redundancy. Due to the critical aspect of proximity operations it has been decided to fly a simple and inexpensive dual redundant system made of two VersaLogic Tiger PC/104-Plus CPU modules, one for the payload computer and the other for the C&DH computer.

Appendix B. ARAPAIMA Concept of Operations

At this stage of mission planning we assume that the on-orbit systems checkout lasts one week. During this checkout the nanosat will perform several orbital maneuvers that reduce the separation from the upper stage. The upper stage that deployed ARAPAIMA will be used as a surrogate Resident Space Object (RSO) that ARAPAIMA will image. The nanosat is maneuvered, with commands loaded from the ground station, to perform station-keeping (SK) and enter natural motion circumnavigation (NMC) with respect to this surrogate RSO. The imaginary RSO is an orbital position which will be propagated on-board the nanosat and used to safely verify the guidance algorithms. Waypoint navigation maneuvers that modify the NMC are performed to check the performance of both the algorithms and of the nanosat subsystems. While checkout is in progress, a few orbit maintenance maneuvers, also called trim burns, are performed to keep the nanosat from drifting away from the upper stage.

Once the ARAPAIMA mission operations team is confident that the nanosat and its flight software perform nominally, the nanosat is commanded to enter a 10x10x10x25km NMC with respect to the RSO. After the NMC insertion is successfully completed the nanosat is commanded to a drifting spiral NMC of 10x10x10x10km in which it collects visible spectrum and IR images of the RSO with the goal of collecting data to characterize the performance of closed loop performing angles only navigation (AON). The images are downloaded to the ground station for verification and the on-board AON algorithms are validated by comparison with solutions generated by the algorithms run by the operations team.

Imagery of the RSO collected during this subphase of the mission is employed to perform the reconstruction of the 3D shape of the RSO. Further maneuvering is performed to bring the nanosat in a relative orbit of 250m radius with respect to the RSO. A safety sphere with a radius of 200m centered at the RSO is currently under consideration. The nominal 250m radius circular orbit is the closest planned approach to the RSO. Imagery taken from this close orbit allows the verification of a set of feature

detection algorithms that use structure from motion and bounded Hough transforms, as well as Sped-Up Robust Feature (SURF) algorithms to obtain autonomous relative pose estimates. A subphase goal of the mission is to the test the algorithms in the highly dynamic conditions of LEO and collect imagery that can be used: 1) on the ground for the verification and validation of new relative navigation algorithms and 2) to provide data for worst-case lighting simulations for future GEO satellite servicing applications. At the conclusion of the science experiments the nanosat will be commanded to reenter the atmosphere and then deactivate.

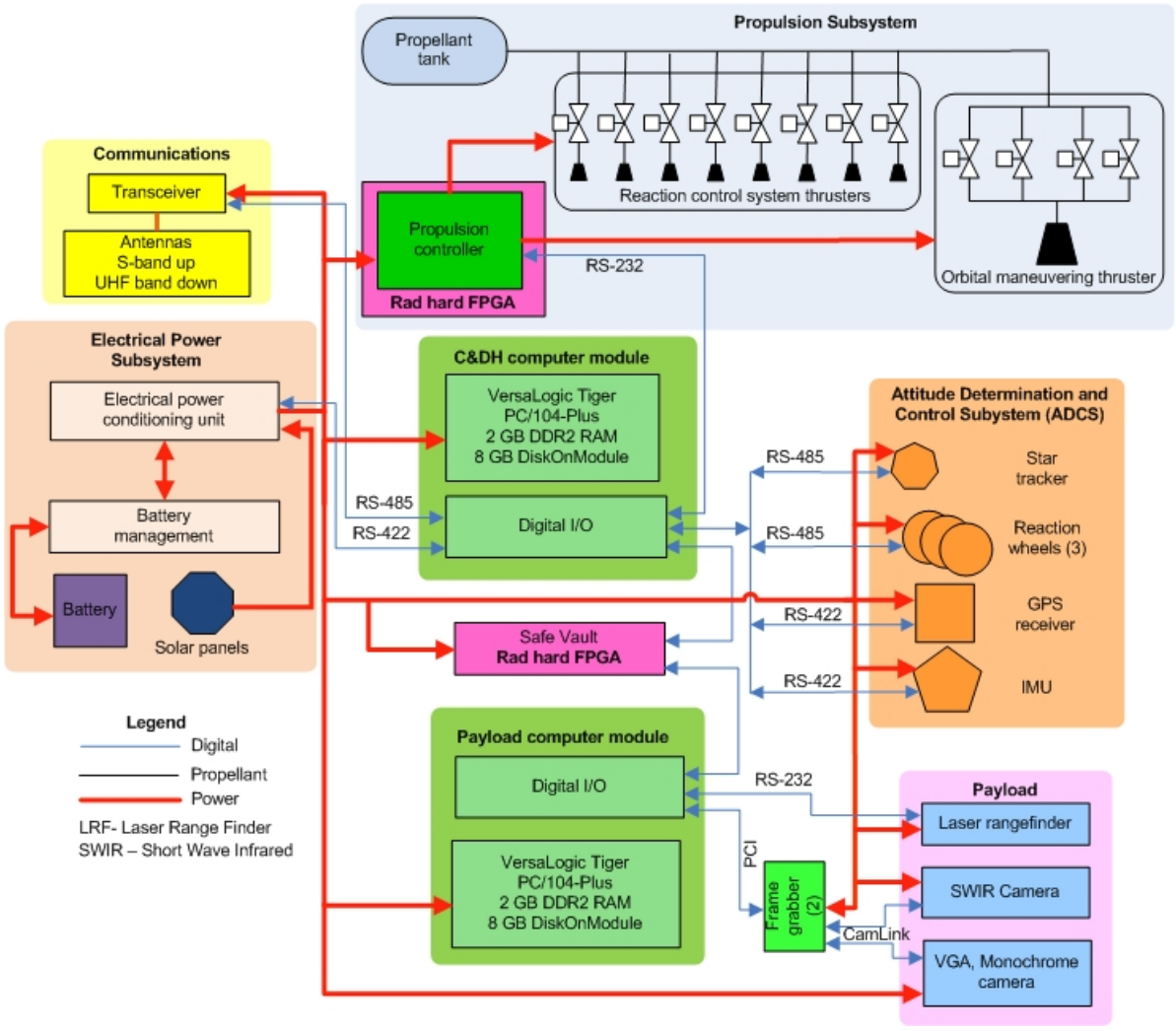

Figure B-1: ARAPAIMA Functional Architecture