The Gestalt: A Secure, High Performance, Low Cost Satellite Ground Station Architecture and its Implementation

Jason Lowdermilk, Simha Sethumadhavan

Chipscan, Inc. New York, NY
{jlowder, simha}@chipscan.us

ABSTRACT

In this paper we present The Gestalt, a novel security methodology developed with support from the Office of Naval Research for satellite ground stations systems. While security is often a stated priority for these systems, often it is traded off for better performance, lower cost and reduced design complexity.

We identified two main classes of security vulnerabilities that can be exploited by attackers in small-sat systems: 1) intentionally introduced supply chain vulnerabilities in both software and hardware, and 2) inadvertent coding and logic vulnerabilities in code.

Our engineering methodology reduces the risk of attacks through four methods:

1. Debloating: Ground stations are complex and involve the integration of many hardware and software systems. This complexity makes them vulnerable to a range of software, and hardware based attacks. Our method of implementing what was previously software functionality in hardware through system debloating achieves this attack surface reduction.

2. Hardware synthesis from Specifications: The use of legacy-free high-level synthesis (HLS) for the specification of processing functions reduces implementation errors, increases productivity, and permits hardware validation using commercial software fuzz testing techniques.

3. Use of hardware scanning techniques: We use a novel method for performing security scans of hardware blocks generated by High-level Synthesis. This step reduces the risk of backdoors inserted by specification developers, attackers modifying the code without knowledge of developers or high-level synthesis tools going undetected.

4. Static memory allocation: A majority of software attacks today are due to memory safety problems in software: Microsoft revealed that 70% of the exploited software vulnerabilities are related to the absence of memory safety.[1] When we use software in The Gestalt, we take a radical approach to solving the pervasive memory safety problem by completely eliminating the use of dynamic memory. Instead, data processing takes place in hardware using static memory allocation.

The result of these approaches is the Exos FEP, a tightly-integrated ground station system that operates in a bit-serial manner. Compared to conventional designs, the Exos FEP achieves high performance by implementing all data processing functions in hardware. Our solution is able to achieve data rates up to 125 Mbps per FPGA in a commodity, commercially cloud-based environment. Perhaps, the most important benefit is a 1000-fold reduction in lines of code compared to state-of-the-art FEP implementations, and achieves Zero Trust supply chain guarantees.

With the increased adoption of smallsats, the security problems normally only associated with large military control centers are now spreading to smaller organizations which may not have the necessary security infrastructure to fully understand or cope with the threats. The possibility of using a security-forward approach such as The Gestalt methodology and the resulting ground system architecture and implementation are a promising approach for protecting the smallsat ecosystem.

INTRODUCTION

Satellite ground stations generally consist of a network of remote antennas and modems, connected to one or more Satellite Operation Centers (SOCs). Each SOC includes two main components: a Front-End Processor (FEP), and a Command and Control (C2) system. The FEP is responsible for signal processing and encryption/decryption, and the C2 system provides the user interface for operators to interact with the spacecraft. In this paper, we describe our FEP implementation, named Exos (short for ExoSkeleton), which is an instantiation of The Gestalt principles for design of secure systems. These principles bear similarities to the principles developed in academia known as the Hardware-Up method for secure system design.

FEP Security

Military SOCs are physically secure: they are actively guarded, all computer systems are security hardened and air-gapped, and the crews are cleared and trained for cyber operations. For this reason, security is seldom a top concern for FEP developers. In particular, security precautions considered necessary for other types of public-facing industries such as web development are often not followed by FEP developers since they are considered unnecessary, with resources better spent in other areas such as improved performance or reduced cost. While it is true that these systems are largely immune from attacks originating from the internet, other types of threats are still possible. This is particularly true for SOCs since they are considered rich targets by nation-state attackers with ample resources available for carrying out cyber attacks.

Supply-Chain Attacks

Despite their physical security, SOCs can be vulnerable to both software and hardware supply-chain attacks. In this type of attack, malicious intrusions are first introduced into the commercial hardware and software components that make up the systems which are ultimately brought into the SOC. A typical commodity server used in a SOC may contain thousands of electronic components from a supply chain that spans multiple continents, as well as millions of lines of software making up the operating system, libraries, and application software. Software is known to contain exploitable vulnerabilities that scale with the number of lines of code[2], so any large software system can be viewed as having a proportionally large number of yet-undiscovered (zero-day) vulnerabilities.

Software Supply-Chain Attacks

In a software supply-chain attack, the vulnerability lies dormant during normal operations and therefore escapes detection during test. But once placed into operation within the secured facility, the vulnerability can be activated, and exploited through a variety of means. Once activated, the intrusion is able to infect any vulnerable systems within range. In this case, FEPs are particularly vulnerable since security precautions were likely not followed during their development and therefore may contain inadvertent coding and logic vulnerabilities.

Hardware Supply-Chain Attacks

A hardware supply-chain attack consists of a hardware component (typically developed in Verilog or VHDL) that has been compromised through the insertion of a stealthy backdoor. Similar to a software supply-chain attack, the backdoor lies dormant through normal usage and remains inactivated until placed into operation. The backdoor is activated by some means such as the arrival of a specific data pattern, which results in the hardware performing unintended functionality. This may include exfiltration of sensitive data, incorrect functionality resulting in the crippling of a defense or weapon system, or the activation of other malicious intrusions within the secure environment.

These types of attacks may take careful planning and execution over months or years to carry out, or contrary to popular belief, can be orchestrated quickly via software attacks on systems that are used to manufacture or hold hardware. Generally speaking, there is less awareness of these types of attacks because very few have been publicly announced.

Nevertheless, the risk posed by supply chain attacks is substantial, so much so that assurance and trust of hardware components used in National Security Systems is a codified requirement by the United States Government [Dod 5200.44]. More recently, the NSA and DISA have issued a reference architecture for implementing Zero Trust that explicitly calls out for hardware and software supply chain assurance.

THE GESTALT ARCHITECTURE AND THE EXOS FEP IMPLEMENTATION

Exos is a secure cloud-based hardware FEP. It is deployed on an Amazon EC2 F1 instance within AWS GovCloud. The FPGA device used is a Xilinx Virtex Ultrascale+ 9. This is in contrast to a typical FEP used in SOCs today entirely run on a commodity processor and complicated software stacks. Further, custom

"hardware" of yesteryears, FPGAs available in the cloud are commodity devices and easy to procure. As SOCs move towards "cloud-native" operations, our cloud-based architecture offers a significant advantage in terms of benefiting from replenishments and upgrades while retaining the benefits of current cloud native FEPs, and substantially boosting their speed and security.

Debloating

One of our main considerations in the development of a secure FEP is to debloat the system as much as possible. Debloating is a common approach used to protect from supply-chain attacks since the number of vulnerabilities in a system scales with the number of lines of code; the total number of lines of code in a system represents the overall attack surface.

We observed that a primary reason for bloat in current systems is to achieve run-time flexibility, for example the ability to dynamically reconfigure features or dynamically change the routing of data paths during satellite contact operations. These are the types of features that FEP developers incorporate into their products in order to differentiate themselves from their competitors, however, these features are not generally required for satellite operations. All configuration for typical satellite operations can be carried out during prepass, and remain static during the satellite pass.

In order to maximize debloating, our architecture replaces many features commonly performed today in software with hardware equivalents which are deployed on an FPGA. This includes frame synchronization and channel coding, derandomization, forward-error correction, checksum validation, encryption/decryption, and packet demultiplexing. These functions are performed entirely using hardware primitives, without the use of a general purpose processor. No features are run-time configurable. Instead, all configuration decisions are made ahead of time, resulting in potentially many different FPGA bitstreams which are selected and loaded during prepass operations according to the requirements of each satellite pass.

A notable benefit of using the FPGA instead of a general purpose processor is that vulnerabilities introduced due to speculative execution in modern processors (such as Spectre, Meltdown), and for which there are no known satisfactory solutions, are completely avoided. Thus we do not even have to trust the CPU during run time taking us closer to Zero Trust!

High-Level Synthesis

One reason that FEPs are often developed in software today is that software development is generally considered to have a lower barrier to entry than hardware development. In particular, software developers are generally more available in the workforce than hardware developers.

Our approach in The Gestalt Architecture is to develop hardware components using HIgh-level Synthesis (HLS) as a design abstraction rather than Register-Transfer Level (RTL). High-level Synthesis allows hardware component specifications to be developed more like software, using methods that may already be familiar to software engineers. We developed most hardware components for the FEP in the Clash language[3], which is a Haskell-like language for hardware development. In other cases, we converted existing open-source C software into Verilog using Vitis HLS.[4] By utilizing HLS in this manner, we significantly improved developer productivity and reduced development time as compared to developing directly in RTL. Typically, the development of new FEPs takes two or more person years while our fully-featured FEP was developed in about 18 person months.

Another benefit of the HLS method is the ability to create software models of each hardware component. These models may be used to simplify the validation of dataflow functionality outside of the hardware environment. And, since these models are software, they may be used in combination with advanced commercial software tools such as code coverage and fuzz testing frameworks. In effect, HLS makes it possible to include these types of tools for hardware development flows, which is not generally possible with low-level hardware development used in FEPs.

Hardware Scanning

The development process used in The Gestalt requires the high-level HLS specifications to be compiled into low-level RTL (Verilog), which is then synthesized into logic gates which ultimately become programmed into the FPGA. From a security perspective, this implies a degree of trust in the tools being used for this conversion. Specifically, it would be possible for language converters and synthesizers to become compromised through software attacks on development systems, and insert backdoors without the developer's knowledge.

In order to mitigate against this type of attack, we utilize ESPY, a commercially available scanning tool.[5] This tool is able to detect backdoors and other stealthy intrusions in RTL or netlists. ESPY has been used to scan every component included in The Gestalt in order to ensure that no backdoors have been inserted by the synthesis tools or HLS compilers.

It is a common practice in hardware development to include 3rd-party Intellectual Property (IP) components in the design. These are often provided for free as part of the synthesis tool framework. In many cases, these are provided as encrypted netlists, and are therefore not possible to scan for security. We avoided the use of any 3rd-party IP during the development of The Gestalt for this reason. For any situation where a 3rd-party solution was needed, we utilized open-source software instead and converted to RTL using HLS, and then performed a security scan on the resulting RTL.

Static Memory Allocation

Memory safety is a long-recognized issue related to software security. Dynamic memory allocation is responsible for many unintentional security vulnerabilities resulting from access errors. This relates to buffer overflows, use-after-free errors, type confusion errors, and many others. This has led to the recent popularity of memory-safe languages such as Rust and Go.[6] However, in our implementation, all memory is statically allocated as it is easy to estimate the total amount of memory necessary for the operation of the system.

In our Exos FEP implementation data buffering is limited to the network interface. Specifically, data arrives from the modem in UDP datagrams which are a fixed length dictated by the Maximum Transmission Unit (MTU) size. The contents of each datagram is stored in a static memory buffer as it is written to the FPGA through a FIFO. After being processed by the FPGA, the resulting CCSDS Space Packet data is returned to the software layer and is again stored in a static memory buffer before being written to the network. CCSDS Space Packets have a maximum size of 64 kilobytes per virtual channel, over a maximum of 64 channels.[7] This represents a total of 4 megabytes of static memory in a worst-case scenario to store the maximum amount of data from every virtual channel.

In terms of hardware design, the HLS and synthesis tools allow different types of memory to be used. Options include DDR4 memory through a memory controller, on-chip high-speed memory or block RAM, and flip-flop memory. For The Gestalt, we chose flip-flop memory in every case in order to avoid the use of memory controllers, as they represent a 3rd-party IP that we are unable to security scan.

Conclusions

In this paper we presented the Gestalt principles for achieving Zero-Trust in large scale systems. The key to achieving zero trust is to think of the large system as a whole, instead of seeing them as composed of small individual parts that can be hardened. The problem with the piecewise hardening approach is that it becomes a combinatorial problem because of how the individual parts may interact with each other and the environment which cannot be predicted. In our approach, by judiciously implementing some parts in hardware we show how the total number of parts can be reduced thus making the problem much more tractable. Using the example of a hardware FEP we showed, contrary to expectations that providing Zero Trust, may be onerous and negatively impact system performance, we show that not only is the security improved, but also the performance and maintainability of the system improves substantially. As such this paper offers an engineering existence proof that systems can be secured in productive and practical manner even against the most advanced, and most insidious threats, and even for legacy systems.

References

1. Miller, Matt. (2019, February 14). BlueHat IL 2019 [Video file]. YouTube. https://youtu.be/PjbGojjnBZQ
2. Jones, Jeffrey. (2007). Estimating Software Vulnerabilities. Security & Privacy, IEEE. 5. 28 - 32. 10.1109/MSP.2007.81.
3. Clash Language. 2021. Clash:Home. https://clash-lang.org/ (2021).
4. Vitis HLS. 2021. Vitis High-Level Synthesis. https://www.xilinx.com/HLS (2021).
5. ESPY Prequal Appliance. 2021. Chipscan Products. https://chipscan.us/products (2021).
6. Pinho, Andre & Couto, Luis & Oliveira, Jose. (2019). Towards Rust for Critical Systems. 19-24. 10.1109/ISSREW.2019.00036.
7. Space Packet Protocol. 2020. CCSDS Blue Books: Recommended Standards. https://public.ccsds.org/Pubs/133x0b2e1.pdf (2020).