TECHNOLOGY WHITEPAPER

Considering the Role of Hardware in Securing OS and Hypervisor Operation

Trusted Computing: The COTS Perspective Series

CURTISS-WRIGHT

DEFENSE SOLUTIONS

Read About

Introduction

In our previous Trusted Computing white paper, ("Hardware Features for Maintaining Security During Operation"), we discussed the powerful hardware features resident in the most popular defense and aerospace processor architectures and how they can be implemented to ensure the continued security of a trusted system during operation. In this column, our focus turns to the role that hardware plays in ensuring the security of operating systems (OS) and hypervisors.

Many applications run on the concept of "least privilege," meaning that the software is only provided access to the minimum set of resources (for example, the hardware and other applications) that they need to complete their tasks. A security context separation between an application and other resources is an important method to ensure that less secure applications and software can't access critical data from more secure and critical applications. Highly sensitive data needs to be protected to ensure that only the code that needs to operate on that data has access to it.

The responsibility of maintaining this type of secure application separation belongs to the OS and the Hypervisor (if one exists on the system). Think of an application as sitting on top of a software stack. Each lower layer of that stack must do its part to ensure that the application layer's security is maintained. At the bottom of the stack resides the hardware, which must be able to enforce the access controls. Running directly on the hardware will be either an OS or a Type-1 hypervisor (Type-2 hypervisors, instead of running on top of the hardware, run on top of an OS) that must manage the access controls. For optimal protection, the system should be configured to ensure that the OS (or Type-1 hypervisor) exploits all available hardware security features so that it can appropriately manage scheduling, resources, processes, and security from the next layer. It is very difficult to build a secure application if the foundation is missing those essential security building blocks.

First let's consider the OS, and how it plays an integral role in ensuring system security on any moderately powerful processor. For many years now, OSs have been given the responsibility for maintaining separation between kernel processes and userspace applications. In fact, the entire function of an OS is to ensure the consistent operation of multiple applications running on a single piece of hardware. As part of their responsibility, OSs have evolved to ensure that a malicious actor in one application is prevented or limited in its ability to influence other concurrently running applications. As processors have become faster and more efficient, they have also become more complicated, which in turn has made the responsibilities of OSs even more complex. For example, one complex process that OSs are responsible for is handling cache management between tasks. As processes move in and out of memory, the OS must ensure that any cached memory is appropriately flushed and/or invalidated. This process can be both difficult and error prone. Add to that challenge factors such as DMA access, side effects, and security issues (such as the row hammer, meltdown, and spectre attacks), and the security responsibilities of the OS become immense.

OS Security Concepts

Today, most processing hardware includes security capabilities that the OS (or hypervisor) must take advantage of in order to be effective. That's because many of these hardware capabilities require operations to be performed in supervisor mode. In the case of processor-based trusted boot resources, like Intel® SGX or Arm® TrustZone, the OS is required to create, and then manage, the access that processes and resources are provided to the appropriate security domains.

The OSs and hypervisor must be designed to take advantage of the appropriate security features built into the hardware. Typically, the OS and hypervisor will run in privileged mode, which means that they are the only entities in a position to make use of all of the features of the processing architecture.

Using an older OS on latest generation hardware can result in some beneficial security capabilities inherent in the newer hardware going unused, limiting the system's potential overall security. Compounding this problem is the fact that updating OSs within military and aerospace systems can be a costly effort. The potential tradeoffs between risk and program cost must be considered in order to best determine when to insert new OS versions into the technology refresh lifecycle.

In addition to ensuring that security boundaries between processes are appropriately maintained, and that the latest security capabilities of the hardware are being used to their full potential, the OS is also responsible for ensuring that resource access is securely managed. It's not enough to maintain process separation. If a process can read another process's data from a peripheral device as the data flows in and out, the result is a compromised trusted computing environment.

The OS software drivers used for accessing peripherals must be designed with appropriate security in mind. This requires an understanding of the tradeoff that that must be made between increasing the access and availability of I/O resources and ensuring that appropriate access separation is controlled and maintained. Some processors provide enhanced capabilities for managing I/O, such as an input–output memory management unit (IOMMU). In that case, the OS and software drivers can take advantage of the enhanced capabilities to provide increased security while maximizing availability of I/O resources.

Hypervisors

In some systems, the OS may also share responsibility for security with a hypervisor. A hypervisor is designed to manage virtualizing resources so that multiple OSs can operate on the same hardware at the same time. When running in a virtualized environment, the OSs and the hypervisor must work together to ensure that a secure and trusted environment is maintained.

As stated earlier, there are two types of hypervisors in use, Type-1 and Type-2. Type-1 hypervisors operate directly on the processor. In that case, located on the next layer in the stack, above the hardware and the Type-1 hypervisor, is the set of guest OSs, each of which will individually perform a function similar to the single OS when a hypervisor is not present. The Type-1 hypervisor virtualizes all hardware resources, and manages access to all the OSs running above it in the stack. VMware® ESX and Xen are examples of Type 1 hypervisors.

In contrast, Type-2 hypervisors run on top of another OS, using that parent OS to access the hardware resources. The Type-2 hypervisor virtualizes those resources to the guest OSs that reside above. VMware Workstation and Oracle® VirtualBox are examples of Type 2 hypervisors.

Linux® KVM is an example of a hybrid hypervisor. In the approach the hypervisor executes in Linux kernel mode directly on the hardware, but uses the Linux OS architecture to manage the virtualized resources.

The type of hypervisor being used will dictate where the base security responsibilities for the guest OSs reside. To protect against a compromise on one guest OS leaking information or access across another guest OS, the Type-1 hypervisor must take advantage of all security features available in the hardware. With a Type-2 hypervisor, it's imperative that the host (or parent) OS is written to appropriately take advantage of the hardware security capabilities. That's because the Type-2 hypervisor uses the host OS to manage and control access to the virtualized resources.

Layering Security

The OS (or the set of guest OSs, if running a Type-1 hypervisor) is responsible for ensuring the separation of userspace processes from supervisory processes. The OS runs the set of processes used to operate the system, each of which has defined interfaces enabling them to communicate and interoperate. The OS is also responsible for ensuring that the processes operate within their defined roles and can use appropriate interfaces to communicate. For example, it catches and prevents invalid operations and interface access, which protects against the failure of one process bringing the entire system to a halt via corruption of concurrently running processes.

The Importance of the Lowest Level

When considering the layers of software running on the stack, the lowest layer software, the OS or hypervisor, should be given the most stringent security requirements. The most stringent security needs to reside at the lowest layer because of the risk and potential impact of a security failure occurring at that level. Lowest level software should be analyzed to make sure that no bugs are present to allow for inappropriate access. If the lowest level OS or hypervisor has a bug, failure at that level could result in the compromise of all trusted systems that use that OS or hypervisor. It could also allow for escalation of privilege, which could result in an application being leveraged to compromise all other processes running in the OS or hypervisor.

While applications (or guest OSs, when running a type-1 hypervisor) also need to be reviewed for security, the risk of failure at that level is normally much more constrained. A failure at the application level can result in the compromise of that single application or a guest OS, but the next level below should normally prevent any such compromise from propagating further in the system.

Because of all the OS and hypervisor security issues discussed above, it's of great value, as early as possible in the design cycle of your trusted computing system to have a discussion with your hardware vendor to best understand the system security requirements. A clear understanding of those requirements will help to ensure that when OSs or hypervisors are being evaluated that their security capabilities are given adequate weight, and that they will leverage the hardware to its fullest capabilities.

The best choice will usually be to select the most recent OS or hypervisor that has been designed with security in mind. Even better, by paring down the chosen OS or hypervisor's feature set, you can minimize potential exploits and security vulnerabilities. In embedded systems there is often the opportunity to configure the OS, such as Linux or WindRiver® VxWorks®, to remove unnecessary features, processes, and libraries. Tailoring the operating system, while often done to minimize size, is also a good security practice to minimize any potential exploits.

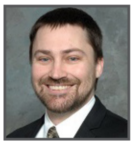
Future white papers will look at processes, procedures, and tools that can be used to analyze the security aspects of the different levels in the hardware/software stack, and the tradeoffs of using these different options for analyzing security of the software used within the system.

For information on Curtiss-Wright's Trusted-COTS™ (TCOTS) program for protecting critical technologies and data in deployed embedded computing systems visit our website at www.curtisswrightds.com/technologies/trusted-computing/.

About Curtiss-Wright Defense Solutions

Curtiss-Wright has a long history of designing embedded computing products for the most demanding environments, and over a decade of experience developing Ethernet switches. Curtiss-Wright networking products are deployed in many of the world's leading military and aerospace platforms where they provide critical connectivity for sensor processing, avionics and mission systems. To support customers with challenging networking requirements, Curtiss-Wright offers a range of engineering services to support architecture, design and troubleshooting activities. For more information on Curtiss-Wright networking products and services, contact your local Curtiss-Wright sales representative.

Author(s)



David Sheets
Security Architect
Curtiss-Wright Defense Solutions

Learn More

Curtiss-Wright Products

- Data at Rest Protection
- Security Enabled Curtiss-Wright SBC and DSP Products

Curtiss-Wright Case Study

- Building a Truly Trusted Computing Solution with COTS Hardware and Intel Security Capabilities

Curtiss-Wright White Papers

- The Many Faces of Trusted Computing: What You Need to Know to Protect Critical Platforms and Data
- Getting Secure, Intel-Based Solutions to Market Faster - Why the Hardware Vendor's Boot Security Implementation Is So Important
- Beyond Trusted Computing: Extending Protection Capabilities to Deliver TrustedCOTS Solutions
- Trusted Computing: The COTS Perspective Series
  1. Introduction to COTS-based Trusted Computing
  2. Trusted Boot
  3. Hardware Features for Maintaining Security During Operation
  4. Considering the Role of Hardware in Securing OS and Hypervisor Operation
  5. Application Development, Testing, and Analysis for Optimal Security
  6. Developing a Secure COTS-Based Trusted Computing System
  7. The Impact of Protecting I/O Interfaces on System Performance
  8. Decomposing System Security Requirements
  9. Establishing a Trusted Supply Chain
  10. Certification Authorities for Trusted Computing in Military and Avionics Products

TRUSTED PROVEN LEADER

CURTISSWRIGHTDS.COM