Creating a Trusted Embedded Platform for MLS Applications

Contents

Executive Summary 1

Introduction 2

MILS Architecture 2

Embedded Separation Kernels 3

NEAT Security Properties 3

Secure Communication 4

Covert Channels 4

Covert Channels Mitigation on Multicore Processors 5

Extended Security Functionality 6

Certification to High Robustness 6

OS Protection Profiles at Basic and Medium Robustness 7

Secure Virtualization 7

Relationship Between Safety and Security 8

Security Assurance for Airborne Systems 9

INTEGRITY-178 tuMP High Assurance RTOS 9

Example INTEGRITY-178 Deployment: MLS and MILS in Action 10

References 11

Executive Summary

Multi-level security (MLS) applications, such as cross-domain solutions (CDS), require a secure environment in which to execute. A proven path to an MLS-capable environment is a Multiple Independent Levels of Security (MILS) operating system based on a Separation Kernel certified to high robustness. MILS decomposes the security certification effort into smaller pieces that can be evaluated to the highest assurance levels in a timely and cost-effective manner. High robustness is needed for the MLS application to fully trust the separation kernel to keep applications and data isolated and to permit only preauthorized communications between applications. The only way to fully trust the separation kernel is to have it evaluated and certified to high robustness.

Security policy enforcement by the separation kernel is non-bypassable, evaluatable, always invoked, and tamperproof (NEAT) because it is small and the only software that runs in privileged mode on the processor. The guarantee of NEAT properties in a MILS operating environment enables the design of an MLS system as a set of independent system high partitions with cross-domain solutions enabling secure communications both among those partitions and also with external systems.

One of the most challenging requirements to meet for high robustness is covert channel mitigation. The use of multicore processors increases the potential for covert channels even with applications on each core partitioned in time and memory space. Systematic covert channel analysis is required to identify covert channels, estimate the channel capacity, and reduce the bandwidth to an acceptable level.

The INTEGRITY®-178 safety and security-critical real-time operating system (RTOS) has been evaluated and certified to the NSA-defined separation kernel protection profile (SKPP) for high robustness. That certification included verification and testing of the covert channel analysis by the NSA as well as NSA penetration testing. This security pedigree has been extended to the INTEGRITY-178 tuMP™ multicore RTOS, including

covert channel analysis across processor cores and shared resources. Today, INTEGRITY-178 remains the only commercial OS or hypervisor ever certified to high robustness, and it is capable of hosting MLS applications such as guards and other cross-domain solutions.

Introduction

Multi-level security applications, such as cross-domain solutions, require a secure environment in which to execute. The foundational layer of such an environment is a trusted operating system. Trust in the operating system (OS) depends on two factors: how robust is the operating system from a security perspective, and assurance that the OS was loaded and configured correctly and not tampered with.

The robustness of computer software is determined by evaluation to the "Common Criteria for Information Technology Security Evaluation" (ISO/IEC 15408).[1] Typically, software is evaluated against a government-defined protection profile that includes both functional and assurance requirements. Evaluations can be done to different levels of depth and rigor, called Evaluation Assurance Levels (EAL), with EAL 1 being the least rigorous and EAL 7 being the most rigorous (figure 1). In the U.S., each government-defined protection profile is declared to one of three levels of robustness: basic, medium, or high.[2] High robustness requires EAL 6 or higher and is appropriate for protecting MLS systems against extremely sophisticated and well-funded threats. Software certified to high robustness, therefore, can be used to protect high-value data in an MLS system.

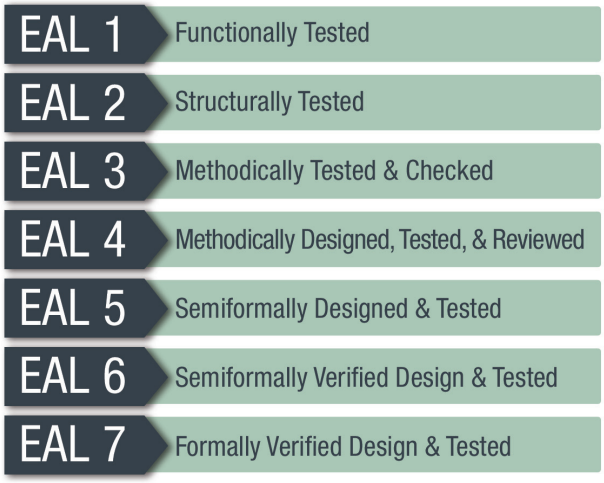


Figure 1: Evaluation Assurance Levels

One path to a trusted OS is an MLS OS, which can access data at multiple levels of classification and permits multiple users cleared to different security levels to access only the data they are permitted at their level and need-to-know (figure 2). Having multiple users at different security clearances and permissions promotes collaboration, and having data at multiples levels on the same system removes the need to duplicate information across different security domains.

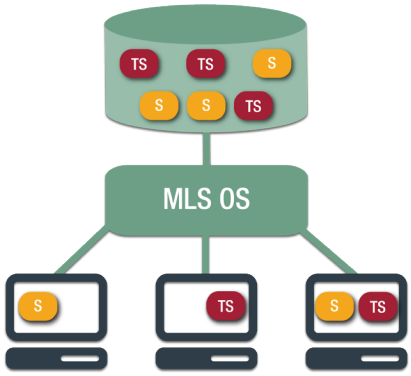


Figure 2: Representative monolithic MLS system

In general, an MLS OS requires mandatory access control (MAC) that supports security labels, for instance, the Bell–La Padula model.[3] Security extensions to common commercial operating systems, such as security-enhanced Linux (SE Linux), provide MLS capability by creating complex, monolithic security kernels. Historically, such monolithic kernels have been challenging and expensive to develop and deploy, and most failed to achieve the highest levels of certification.[4] Specifically, the highest evaluation assurance level achieved by such operating systems has been EAL 4+ using protection profiles defined for medium robustness (see the "OS Protection Profiles" sidebar on page 7). Those profiles are only designed for "protection against threats of inadvertent or casual attempts to breach the system security." Therefore, any MLS OS evaluated for medium robustness is not sufficient for MLS applications with high-value information requiring protection from a determined attacker.

MILS Architecture

The main alternative to a monolithic security kernel is a Multiple Independent Levels of Security operating environment. MILS divides the software architecture into three layers: the separation kernel, middleware, and applications.

Each layer enforces a separate portion of the security policy set, with the separation kernel the only layer executing in privileged mode. Applications can enforce their own security policies, enabling application-specific policies instead of relying on broad security policies in a monolithic kernel. Each layer and application can be evaluated separately without impact to the evaluation of the other layers/applications, making the MLS system easier to implement, certify, maintain, and reconfigure.[5]

First proposed by John Rushby,[6] a separation kernel divides memory into partitions using a hardware-based memory management unit (MMU) and allows only carefully controlled communications between non-kernel partitions. Furthermore, operating system services, such as networking stacks, file systems, and most device drivers, execute in a partition instead of in the kernel in privileged mode (figure 3). This enables the separation kernel to focus on providing four foundational security policies: [4, 5]

- **Data Isolation:** Information in a partition is accessible only by that partition, and private data remains private.
- **Control of Information Flow:** Information flow from one partition to another originates only from authorized sources, is delivered only to intended recipients, and the source of information is authenticated to the recipient.
- **Resource Sanitization:** The microprocessor itself will not leak information from one partition to another as it switches from partition to partition.
- **Fault Isolation:** A failure in one partition will not cascade to any other partition. Failures are detected, contained, and recovered from locally.

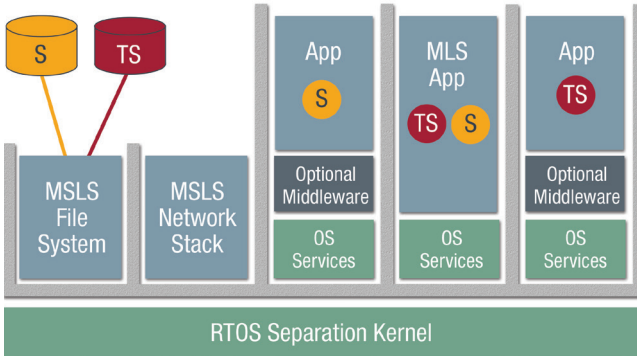


Figure 3: Representative MILS architecture. Applications run in partitions and access external data through a multiple single level security (MSLS) file server.

Focusing on only those four security policies and moving operating system services out of the kernel into the middleware layer enables a separation kernel to be very small in size. Dramatically reducing the size of the kernel code enables dramatically increasing the rigor of scrutiny. This makes possible certifying a separation kernel to EAL 6+ and "high robustness." The INTEGRITY-178 RTOS is an example of a MILS separation kernel, and it has been certified to EAL 6+ and high robustness.

Embedded Separation Kernels

Within the context of embedded systems, a separation kernel generally operates autonomously. The lack of human intervention has two broad implications for high robustness. First, there is no need to support the identification and authentication of users and administrative roles. Second, without administrators to monitor and maintain the kernel during runtime, additional assurance measures are inherently required to ensure highly robust autonomous execution of the security management functions.[7]

Embedded systems typically are designed to perform a specific set of tasks. This predetermined functionality generally allows for the static configuration of both the resources to partitions and the permitted communication between partitions. If the configuration does not change during runtime, the kernel design and implementation can be relatively small, thereby easing the validation and certification to high robustness.[7] Embedded systems may, however, need to adapt to changes in the environment, such as the failure of a peripheral device. This can be accommodated by defining multiple static configurations and allowing an authorized application or process to switch to a different preloaded configuration. When a runtime change is invoked, the security functions have the additional requirement to continuously maintain a secure state before, during, and after the configuration change.[7]

NEAT Security Properties

Because the separation kernel enforces the data isolation and controls communication between partitions, untrusted applications and data objects at various levels of classification can reside on a single processor. The separation kernel also enables trusted applications to execute on the same processor as less-trusted applications while ensuring

that the trusted applications will not be compromised or interfered with in any way by the less-trusted applications. Security policy enforcement by the separation kernel is non-bypassable, always invoked, and tamperproof because it is the only software that runs in privileged mode on the processor.[8] Together with the small size of the separation kernel making it "evaluatable," these four properties go by the acronym NEAT.

NEAT Security Policy Attributes

The four main security attributes of a high-assurance separation kernel (i.e. security monitor):

Non-bypassable: An application cannot bypass the security monitor.

Evaluatable: The security monitor is modular, small in size, and sufficiently low in complexity to support rigorous evaluation.

Always-invoked: The security monitor checks every access and communication.

Tamperproof: The system prevents unauthorized changes to the security monitor code, configuration, and data.

The guarantee of NEAT properties in a MILS operating environment enables the design of an MLS system as a set of independent system high partitions with cross-domain solutions enabling secure communications both among those partitions and also with external systems. By leveraging the NEAT security policy enforcement provided in a separation kernel evaluated to high robustness, those cross-domain servers, down-graders, and guards can be small and tightly focused. This, in turn, makes high assurance evaluations of those cross-domain solutions practical, achievable, and affordable.[4]

Secure Communication

In a MILS architecture, the networking components such as protocol stacks and their associated interface device drivers are placed in user-space partitions of their own. Network data is processed in least-privileged user mode, thereby eliminating a vulnerability that is a common avenue of attack,[4] such as the URGENT/11 zero-day vulnerabilities.[9]

Complex networking protocol code can be evaluated and certified independently of the applications that use that code, enabling reuse of the evaluation artifacts.[4]

One source of requirements for secure communication channels between security partitions can be found in the Trusted Network Interpretation,[10] which described four types of vulnerabilities that must be protected against:

1. **Communication security**: unauthorized disclosure or modification of sensitive information in transit
2. **Communication reliability**: unreliable delivery of information (e.g., non-delivery, misdelivery, late delivery, and delivery of erroneous data), the delivery of which is required for the correct operation of the trusted computer base (TCB) (such as audit records or inter-partition security coordination)
3. **Communication fidelity**: changes to security-critical data, such as transmitted security labels, due to noise
4. **Covert signaling**: manipulation of the channel mechanisms to signal information covertly

Some standard protection mechanisms can protect against multiple vulnerabilities. Cryptographic sealing, for example, addresses the issues of both prevention of unauthorized modification and error-detection. Covert signaling, known more commonly as covert channels, arguably is the most challenging type of vulnerability to protect against. While covert signaling can hide within established communications channels, more often it exploits a mechanism not intended to be used for communication.

Covert Channels

A covert channel is an unintended or unauthorized communications path that can be used to transfer information in a manner that violates a security policy.[11] Covert channels can be categorized as storage-based, timing-based, or a hybrid of the two. Covert storage channels transfer information through the setting of bits by one application in a location that is readable by another application. Covert timing channels convey information by modulating some aspect of system behavior over time in a way that can be observed by another application.

An example of a potential covert storage channel is changing a filename. A small amount of covert information

could be sent in the value of each new filename, or a single bit could be sent by either changing or not changing the filename at a regular interval.[12] Another potential covert storage channel occurs if a low assurance application can see how much memory is left after a high assurance application allocates memory. The size of free memory or the change in the size of free memory can be a numeric value transmitted by the high assurance application. This potential channel can be eliminated by allocating a maximum amount of memory for each application.

An example of a potential covert timing channel is communication between two applications scheduled adjacent to each other. The first application can modulate its execution time, which can be observed and measured by the subsequent application. This potential channel can be eliminated by the use of a partition scheduler in the operating system, which assigns fixed-length execution periods to each application. Note that some administrative actions are also required for maximum protection, such as not allowing background tasks to run when an application finishes before its fixed time allocation expires.

Many covert channels are much more difficult to identify and mitigate. A high robust separation kernel needs to demonstrate that a systematic approach was taken to identify and mitigate covert channels across the range of possible communication mechanisms. Mitigation techniques include: shutting down or preventing the covert channel, limiting the bandwidth of potential covert channels to where the assurance outweighs the risks, and ensuring that only highly trusted applications have access to the covert channels. In the U.S., high robustness software undergoes covert channel analysis and testing by the NSA as part of obtaining a high robustness certification.

Covert Channels Mitigation on Multicore Processors

The use of multicore processors dramatically increases the number of potential covert channels. Any processor resource that is shared among concurrently executing cores is a potential means for covert communications. Therefore, multicore processors must mitigate possible covert channels between both concurrently executing partitions on different cores as well as partitions executing subsequently on the same core.[13]

One possible mitigation strategy is to schedule only partitions at the same security assurance level to run on any given core. This strategy eases mitigation of the potential covert channels executing subsequently on the same core but does not address the larger number of potential covert channels between cores. Other means of mitigation will be needed to handle shared resources such as shared cache. Those mitigations potentially can be very expensive in terms of complexity and execution overhead. The large size of modern L2 caches makes flushing or invalidating on every partition switch impractical.[13] On the plus side, this strategy is relatively simple to implement. It only requires the separation kernel to support the simplest form of multi-processing, known as asymmetric multi-processing (AMP), where the OS running on each core is unaware of and uncoordinated with those running on other cores.

An alternative strategy[13] is to have only partitions of the same security assurance level executing concurrently on different cores. This strategy greatly mitigates a large number of potential covert channels between cores and enables additional mitigation techniques to focus on the less complex potential covert channels between subsequent partitions. The overhead of cache flushing or invalidating cache can be minimized by scheduling the order of partitions to execute, starting with the lowest security assurance level and sequentially moving to higher assurance levels. Since a lower security level partition communicating to a higher security level partition is an accepted information security practice, the use of the processor core's state on these types of partition switches is not a security issue. The covert signals that need to be prevented are the transitions from a higher security level to a lower level. Although it is possible to use this strategy using AMP, it is much more effective to employ symmetric multiprocessing (SMP) and bound multi-processing (BMP), where a multithreaded application/partition can run across multiple cores for increased performance and security. The INTEGRITY-178 tuMP RTOS is an example implementation of a separation kernel that supports mixing AMP, BMP, and SMP on the same multicore processor.

An illustration of that strategy running on a BMP-capable separation kernel is shown in figure 4.[13] NATO rated (NR) partitions precede Secret (S) level partitions, which precede Top Secret (TS) partitions. The abstract machine test (AMT)

partition always resides between such transitions and ensures the core's state is flushed/sanitized as part of its testing operations. The configuration illustrated in figure 4 reduces the cache flush operations to a small set of partition switches, thereby minimizing the mitigation overhead.

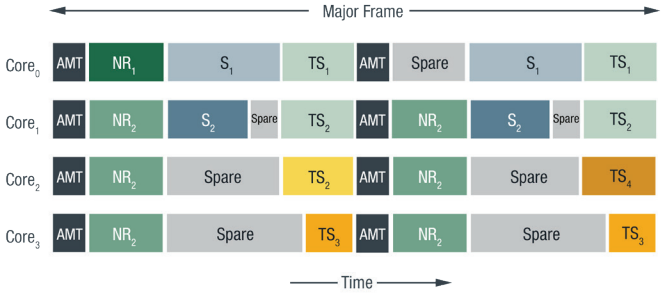


Figure 4: Only partitions of the same security level run concurrently, with some partitions running across multiples cores in BMP mode.

Extended Security Functionality

Other functionality that may be required, depending on the level of security robustness required, includes audit logging, integrity tests, and abstract machine tests. Audit logging records specific events during execution of the separation kernel to detect potentially malicious code behavior. Integrity tests ensure the integrity of the executable images of the separation kernel stored in both volatile and non-volatile RAM. Those include continuous tests of the separation kernel's active executable image in RAM as well as a set of power-up tests. Abstract machine tests are continuous tests that ensure the hardware protection mechanisms are being enforced. This includes, for example, tests that attempt memory violations and privileged instruction execution in order to ensure the hardware that enforces separation between the virtual address spaces is still operational. Audit logging, integrity tests, and AMT are all required to meet High Robustnes[11] and are implemented for INTEGRITY-178 tuMP.

Certification to High Robustness

As described earlier, the use of a MILS architecture and minimization of the TCB are not the end goal but a means to achieve certification to high robustness. The only government-defined operating system protection profile ever designed for high robustness or EAL 6 and above is the "U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness" (SKPP),[11] which was issued by the Information Assurance Directorate of the US National Security Agency (NSA).

The robustness of an information assurance solution is rated as high when it is determined to be resistant to an extremely sophisticated adversary with abundant resources such as a nation-state threat. Levin et al[14] state that a product that satisfies EAL 7 does not necessarily satisfy high robustness. To achieve high robustness, Levin reports that the development team for the SKPP:

1. Assembled an initial set of EAL 6 assurance requirements based on Common Criteria 2.3.
2. Augmented the initial set with EAL 7 assurance requirements necessary for high robustness.
3. Extended the requirements (partly by creating new assurance requirements) for topics important to high robustness that are not covered by Common Criteria. These topics included trusted initialization and inclusion of the hardware.

The result was a protection profile that met the military and security information assurance needs as defined by security analysts associated with the NSA and security specialists from other organizations. Products that conform to the SKPP support information flow control, resource isolation, trusted initialization, trusted delivery, trusted recovery, and audit capabilities. As stated in the SKPP, "when integrated within a high-assurance security system architecture, the SKPP separation mechanisms are appropriate to support critical security policies for the Department of Defense (DoD), Intelligence Community, the Department of Homeland Security (DHS), [and] Federal Aviation Administration (FAA)."[11]

In 2007, the NSA published the SKPP version 1.03. Approximately one year later, the National Information Assurance Partnership (NIAP) certified INTEGRITY-178 against the SKPP to high robustness and EAL 6+.[15] That spurred a host of vendors of other operating systems and hypervisors to announce that they, too, would work to achieve SKPP certification.[16] After three more years, none of those OSes and hypervisors managed to pass certification. After failing to achieve certification, some vendors still claim that their product "met the requirements of," "complies with," "is certifiable to," or even claim to be "SKPP-conformant"[17] with

no evidence to back those claims. In 2011, the NSA ceased general certification to the SKPP in favor of evaluating OS and separation kernels as part of specific government systems in the context of specific programs. NSA continues to recommend the use of separation kernels and the requirements in the SKPP.[18] Programs of record continue to use the requirements in the SKPP as a path for system certification to high robustness.

Secure Virtualization

When virtualization is needed in a secure setting, it is tempting to jump right to a Type 1 hypervisor that claims to be secure using separation kernel technology. A Type 1 hypervisor, which runs on bare metal, is often the best choice for enterprise IT virtualization, but it is neither the most secure nor the highest performance option for most embedded systems. The first can be seen by examining the TCB; the second by examining assumptions and configurations of embedded systems.

When assessing a security solution, a key concept is the size of the TCB, which is comprised of the hardware, software, and controls that enforce the security policy.[23] The general goal is to minimize the size of the TCB and number of interfaces so that it can be verified more easily. The larger the TCB and number of interfaces, the larger the attack surface is. Minimizing the TCB requires moving many non-critical services out of the TCB, which in turn requires both the ability to isolate those services and to provide secure communication between trusted and non-trusted components. Note that minimizing the TCB is not the end goal but only a means to ease verification. For systems requiring a high level of security, the end goal is certification to applicable security assurance requirements.

One of the perceived benefits of virtualization is the added security that comes from isolating applications into virtual machines enforced by hardware mechanisms such as the MMU and IOMMU. However, that isolation and use of hardware mechanisms are already inherent in the separation kernel,[6] and including virtualization in the kernel does not increase security. To the contrary, including virtualization in the kernel code actually represents an increased security risk. That is because the amount of code required for virtualization can be huge, dramatically increasing the TCB. That increase in the TCB virtually nullifies the main benefit

OS Protection Profiles at Basic and Medium Robustness

The framework for Common Criteria is very generic, allowing suppliers to define their own security requirements for the evaluation. The primary value of Common Criteria comes when the evaluation is done against a government-defined protection profile. In the U.S., the National Information Assurance Partnership (NIAP) defines protection profiles and manages the Common Criteria Evaluation and Validation Scheme (CCEVS) validation body. When the evaluation is at EAL 5 or higher, the NSA participates in the evaluation.

The level of security does not come directly from the evaluation assurance level but from the security functional requirements in the protection profile. It is only when a high EAL is achieved with a very demanding protection profile that the best security is achieved.

An example basic protection profile is the Protection Profile for General Purpose Operating Systems (GPOS-PP)[19] to which some enterprise operating systems have been certified, such as Windows and Red Hat Enterprise Linux.[20] The GPOS-PP was not designed for a specific EAL but is based mostly on EAL 2 requirements.

Additional protection profiles were defined to enhance certain aspects of system security to a medium level of robustness. Examples include the Controlled Access Protection Profile (CAPP)[21] and the Labeled Security Protection Profile (LSPP).[22] CAPP defines access controls that are capable of enforcing access limitations on individual users and data objects while also providing audit capability. LSPP specifies a set of access controls that 1) allow individual users to specify how resources (e.g., files, directories) under their control are to be shared and 2) enforce limitations on sharing among users by the use of security markings (i.e., "labels"). Both CAAP and LSPP explicitly state that they are only intended to "provide a level of protection which is appropriate for an assumed non-hostile and well-managed user community requiring protection against threats of inadvertent or casual attempts to breach the system security." The assurance requirements are specified at EAL 3. Some OS suppliers choose to certify at EAL 4, but that does not increase the security level because it does not change the security requirements specified in the protection profile.

of a separation kernel, namely to increase significantly the level of rigor when inspecting that security-critical code by dramatically reducing the amount of security critical-code to be inspected.

Two factors drive the need for virtualization code to be very large: portions of virtualization without hardware acceleration and performance optimizations. Although hardware technologies accelerate many functions of virtualization, only recently are some processors beginning to accelerate some portions of I/O. Some examples of virtualization without acceleration can include device emulation, bus emulation, interrupt emulation, and routing. The code for all that emulation is quite large and also creates a performance penalty. Every call to the kernel from the guest OS needs to be trapped, examined, and determined if the guest OS is permitted that access. In order for a hypervisor to be efficient, it needs to virtualize sequences of instructions instead of single instructions. Such look-ahead capability is just one example of increasing the already large code base of a hypervisor in pursuit of minimizing the virtualization performance penalty.

The alternative to including virtualization in the kernel-level code is to provide the functionality in user space. A separation kernel already moves other higher-level OS services to user space, including the network stack and file system, and this is the best place for virtualization as well in order to minimize the TCB and increase the ability to be certified to high robustness. This approach has some similarities to a Type 2 hypervisor, which runs on top of a host OS, except that the isolation foundation is implemented in the microkernel.

One reason to consider a Type 1 hypervisor, which runs natively on the host hardware, is the potential higher performance over a Type 2 hypervisor that comes from eliminating one layer of software, namely the host OS. That assumes, however, that all or most applications require virtualization. While that is often true for enterprise servers, in many embedded systems the vast majority of applications run on a real-time OS (the host OS) and don't require virtualization. Those applications actually have lower performance and determinism if they have to run on top of a Type 1 hypervisor as well as RTOS (figure 5). For most real-time systems, a Type 2 hypervisor yields higher overall system performance than a Type 1 hypervisor by eliminating another level of software, namely the hypervisor itself. Only the applications that require virtualization pay the performance penalty of running on top of a hypervisor.

The INTEGRITY-178 tuMP RTOS is a separation microkernel with OS services, including virtualization, implemented in user space instead of in the kernel. A separation microkernel that includes a virtualization layer in user space has some similarities to a Type 2 hypervisor in that the virtualization layer runs on top of a host OS and that it can be applied selectively to only the applications that require virtualization. However, it differs in that the isolation function is provided by the separation microkernel (acting as the host OS), and isolation is enforced even between different instances of the virtualization layer (figure 6). The virtualization layer in user space makes use of the fundamental separation mechanisms implemented in the separation microkernel, including the use of MMU and IOMMU. The resulting reduction in TCB enables INTEGRITY-178 tuMP to meet all the requirements of the SKPP at High Robustness while providing guest OS virtualization. Real-time, high-assurance applications run directly on INTEGRITY-178 tuMP, thereby achieving the tightest determinism and the highest performance.

Relationship Between Safety and Security

Similarities exist between the concept of partitioning for safety and the concept of separation for security. In practice, there is also significant overlap. Safety partitioning and security separation share the fundamental requirements of data integrity and availability. At a high level, certification to DO-178C DAL A[24] demonstrates that a software solution does what it is supposed to do and nothing else. Security separation has the additional requirement of confidentiality, which includes verification that there are no unintended side effects. Although ensuring confidentiality, such as all potential read accesses to memory or devices are authorized, is not generally required for safety-critical systems, it can be perceived as a benefit due to the strengthening of arguments of independence.[13]

One of the more challenging aspects for achieving confidentiality to high robustness is mitigating covert channels. Yet even here, there is a relationship to safety assurance

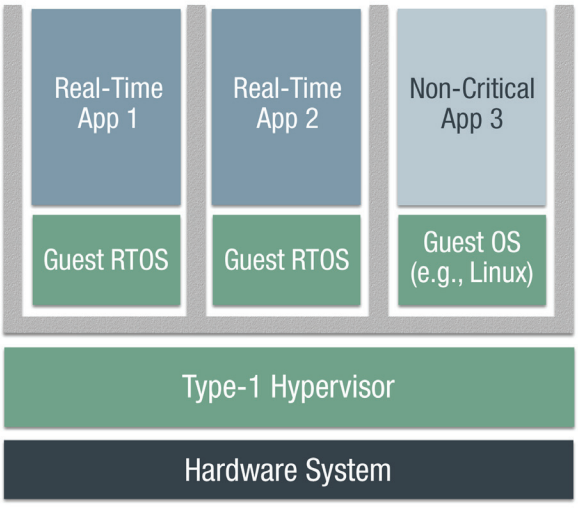


Figure 5: With a Type 1 hypervisor, all applications run on two layers of software, the hypervisor and a guest OS.

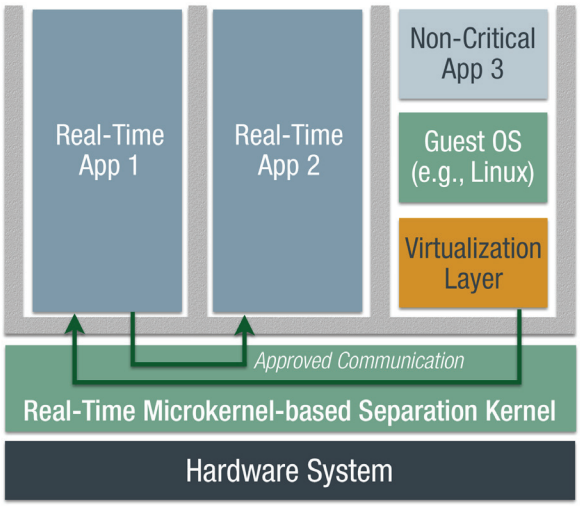


Figure 6: With a microkernel, the real-time applications do not incur the overhead of the virtualization layer.

as the sources of covert timing channels required for many types of security attacks are often the same sources of multicore interference with respect to availability concerns impacting determinism and ultimately safety. Therefore, an RTOS that meets the security assurance requirements, as defined in the SKPP's separation kernel requirements, also enhances the safety attributes. Minimizing the TCB helps in achieving high assurance levels for both safety and security.

Given the overlapping and complementary relationship between safety and security, it makes sense for a single product to meet the high-assurance requirements for both safety and security. The INTEGRITY-178 tuMP separation kernel meets the requirements for both DO-178C at DAL A and the SKPP at high robustness.

Security Assurance for Airborne Systems

The aviation industry now has security guidance as well. As a complement to DO-178C Software Considerations in Airborne Systems and Equipment Certification,[23] there is a set of high-level specifications for airborne security starting with DO-326A Airworthiness Security Process Specification.[25] That specification contains security objectives at the system and aircraft level. Any new commercial aircraft system fielded should consider the DO-326A requirements.

DO-326A does not specify how to implement the required security objectives; it only provides guidance on the process to identify threat vectors and make sure adequate mitigation measures are in place. Many of the processes in DO-326A parallel those in DO-178C, such as requiring a plan for security aspects of certification (PSecAC) similar to the plan for software aspects of certification (PSAC).

Software products that are certified to Common Criteria at EAL 5 or higher have a head start on meeting DO-326A because there is a significant overlap in the processes. The SKPP, in particular, has much more stringent security requirements and testing than is required for DO-326A.

INTEGRITY-178 tuMP High Assurance RTOS

The INTEGRITY-178 tuMP real-time operating system (RTOS) from Green Hills Software provides a MILS operating environment based on a separation microkernel that is capable of hosting MLS applications, including cross-domain solutions. Designed from the beginning for both safety and security, INTEGRITY-178 tuMP provides the high level of data isolation, control of information flow, resource sanitization, and fault isolation required for a separation kernel of high robustness. Those foundation security policies are non-bypassable, evaluateable, always invoked, and tamperproof (NEAT), providing the high assurance level needed to enable the design of an MLS system as a set of independent, secure partitions with cross-domain solutions enabling secure communications among those partitions.

In 2008, the INTEGRITY-178 RTOS became the first and only operating system to be certified against the SKPP, and that same codebase simultaneously complied with the requirements defined in RTCA/DO-178B Level A. The certification against the SKPP was to both "High Robustness" and EAL 6+. Green Hills Software's latest RTOS version for multicore processors, INTEGRITY-178 tuMP, continues to meet the SKPP's rigorous set of functional and assurance requirements for those customers needing it.

After the NIAP and NSA sunsetted software-only certifications to the SKPP in 2011 in favor of evaluation of specific system configurations against specific program requirements, programs of record continue to use the requirements in the SKPP as a path to system certification. Green Hills Software continues to maintain INTEGRITY-178 and INTEGRITY-178 tuMP to meet the SKPP requirements, address new threats, and expand functionality in support of programs requiring high robustness. INTEGRITY-178 and INTEGRITY-178 tuMP have been used successfully in more than a dozen high-security assurance systems requiring certification to high robustness. See the next section for an example certification and deployment.

By meeting the requirements for high robustness, INTEGRITY-178 tuMP serves as the trusted MILS implementation that enables the design of an MLS system as a set of independent system high partitions with cross-domain solutions enabling secure communications among them.

As part of certification to the SKPP, INTEGRITY-178 underwent independent vulnerability analysis and penetration testing by NSA to demonstrate both that it is resistant to an attacker possessing a high attack potential and that it does not allow attackers with high attack potential to violate the security policies. Additionally, it underwent covert channel analysis by NSA to demonstrate that it satisfies all covert channel mitigation metrics. The BMP and SMP capabilities in INTEGRITY-178 tuMP allow a security software architect to mitigate a large portion of potential covert channels between cores by scheduling partitions of the same security level to execute concurrently, including the execution of multithreaded applications across multiple cores for increased security and performance.

Beyond the approval as an MILS separation kernel, INTEGRITY-178 provides a complete set of APIs that were also evaluated by the NSA for use by MLS applications within a secure partition, e.g., an MLS guard, which is a fundamental requirement in a cross-domain system. Because both safety and security are designed into the same product, those secure APIs include support for multithreading, concurrent execution on multiple cores, and flexible core assignments at the configuration file level, all within the secure MILS environment.

INTEGRITY-178 tuMP contains extended security assurance functionality as required by the SKPP for audit logging, integrity tests, and abstract machine tests to ensure that the hardware protection mechanisms are operational and being enforced. Beyond that, the unique bandwidth allocation and monitoring (BAM) capability in INTEGRITY-178 tuMP designed to increase safety assurance by mitigating multicore interference can be used to thwart denial-of-service attacks from compromised partitions/applications resulting from the unintended or malicious use of the multicore processor's shared resources.

For applications that require virtualization to run legacy or untrusted applications, INTEGRITY-178 tuMP provides a virtualization layer as part of the operating services in user space. This greatly reduces the TCB, enabling easier verification and certification to high assurance levels. It also provides more determinism, performance, and security for applications that do not require virtualization, which likely constitute the vast majority in a real-time system. Being a true RTOS, INTEGRITY-178 tuMP also provides many more OS services than a hypervisor-based separation kernel. Most MLS applications, such as cross domain guards, require a rich set of OS functions, including tasking services, semaphores, and message passing. With a hypervisor-based solution, you would need to run a secure guest OS on top of a secure hypervisor to provide that level of service while taking a performance penalty from the extra layer of software.

Together, the safety and security assurance certifications of INTEGRITY-178 are proven in real-world customer applications with over 80 DO-178B/C Level A/EAL 6+ unique customer certification packages delivered across more than 30 different microprocessors.

Example INTEGRITY-178 Deployment: MLS and MILS in Action

The U.S. Navy's Tactical Combat Training System Increment II (TCTS Inc. II) program uses multi-level security (MLS) system solutions from Collins Aerospace to allow for simultaneous processing of data at different classifications and levels. TCTS Inc. II enables the rapid adaptation of new missions and threats into training as well as joint and coalition interoperability with fourth- and fifth-generation aircraft platforms.

Collins implements their MLS system solution with a multiple independent levels of security (MILS) architecture based on the INTEGRITY-178 tuMP separation kernel. Because INTEGRITY-178 tuMP is built to EAL 6+ and high-robustness requirements, the MLS applications can depend upon the INTEGRITY-178 tuMP separation kernel to enforce the data isolation between applications and to restrict interactions between applications to only explicitly authorized communication flow. This combined MILS architecture drove reduced size, weight, and power (SWaP), reduced certification costs and schedule, and lower lifecycle costs.[26, 27]

The key factors for Collins Aerospace selecting INTEGRITY-178 tuMP were its proven MILS architecture, the ability to host MLS applications such as cross domain solutions, the ability to host a guest OS and legacy applications in a secure virtualized partition, and its conformance to the Future Airborne Capability Environment (FACE™) 3.0 Technical Standard.[27] The high-assurance security documentation and evidence available for INTEGRITY-178 tuMP also eases security certification effort. According to Collins, the TCTS Inc. II solution, including INTEGRITY-178 tuMP, is the only air-combat training system with security certification that supports security requirements of today's 5th-gen and 4th-gen fighters and can be run at system high.[28]

References

1. Common Criteria for Information Technology Security Evaluation, Version 3.1 Revision 5, CCIMB-2017-04-[001, 002, 003], Apr-2017.
2. Consistency Instruction Manual For Development of US Government Protection Profiles For Use in Medium Robustness Environments, Release 3.0, National Security Agency, 1-Feb-2005.
3. D. E. Bell and L. E. La Padula, "Secure Computer Systems: Mathematical Foundations," MTR–2547, Vol. I, The MITRE Corporation, 1-Mar-1973.
4. W. M. Vanfleet, et al, "MILS: Architecture for High Assurance Embedded Computing," CrossTalk, Aug-2005.
5. J. Alves-Foss et al, "Enabling the GIG", PowerPoint presentation, Integrated Defense Architectures Conf., 11-May-2005.
6. J. Rushby, "Design and Verification of Secure Systems," ACM Operating Systems Review, vol.15, no.5, Dec-1981.
7. T. Nguyen et al, "High Robustness Requirements in a Common Criteria Protection Profile," Proceedings of the Fourth IEEE International Information Assurance Workshop, Apr-2006.
8. IAEC 3285, NSA Infosec Design Course, High Robustness Reference Monitors version 3, Michael Dransfield, W. Mark Vanfleet.
9. "Multiple Vulnerabilities in Wind River VxWorks Could Allow for Remote Code Execution," Center for Internet Security, 31-Jul-2019. https://www.cisecurity.org/advisory/multiple-vulnerabilities-in-wind-river-vxworks-could-allow-for-remote-code-execution_2019-075/
10. National Computer Security Center, Trusted Network Interpretation, NCSC-TG-005, section B.6.2, 31-Jul-1987.
11. 1U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness, Version 1.03, National Security Agency, 29 July 2007.
12. H. Okhravi, S. Bak, S. King, "Design Implementation and Evaluation of Covert Channel Attacks", Proc. IEEE Int'l Conf. Technologies for Homeland Security, pp. 481-487, 2010. http://web.mit.edu/ha22286/www/papers/HST10.pdf
13. P. Huyck, "Safe and Secure Data Fusion – Use of MILS Multicore Architecture to Reduce Cyber Threats," 2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC), Sep-2019.
14. T. Levin et al, "Separation Kernel Protection Profile Revisited: Choices and Rationale", Layered Assurance Workshop, Austin TX, 2010.
15. "Green Hills Software Announces World's First EAL6+ Operating System Security Certification," Green Hills press release, 17-Nov-2008. https://www.ghs.com/news/20081117_integrity_EAL6plus_security.html
16. "Wind River's VxWorks MILS 2 in Evaluation for Certification to Common Criteria EAL6+/NSA High Robustness," Wind River press release, 25-Jun-2008.
17. "Wind River Introduces VxWorks MILS Platform Conformant to Separation Kernel Protection Profile," Wind River press release, 23-Apr-2013. https://www.windriver.com/news/press/pr.html?ID=11742
18. C. Houck, "SKPP Sunset Q&A," National Information Assurance Partnership, Jul-2011. https://www.niap-ccevs.org/announcements/SKPP%20Sunset%20Q&A.pdf

19. “Protection Profile for General Purpose Operating Systems, Version 4.2.1,” NAIP, 22-Apr-2019. https://www.niap-ccevs.org/MMO/PP/PP_OS_V4.2.1.pdf
20. “Red Hat Adds Common Criteria Security Certification for Red Hat Enterprise Linux,” Red Hat, 13-Dec-2017. https://www.redhat.com/en/about/press-releases/red-hat-adds-common-criteria-security-certification-red-hat-enterprise-linux
21. “Controlled Access Protection Profile, version 1.d,” NSA, 8-Oct-1999. https://www.niap-ccevs.org/MMO/PP/pp_os_ca_v1.d.pdf
22. “Labeled Security Protection Profile, version 1.b,” NSA, 8-Oct-1999. https://www.niap-ccevs.org/MMO/PP/pp_os_ls_v1.b.pdf
23. Department of Defense Trusted Computer System Evaluation Criteria, DoD 5200.28-STD, 1985.
24. DO-178C Software Considerations in Airborne Systems and Equipment Certification, RTCA, 13-Dec-2011.
25. DO-326A Airworthiness Security Process Specification, RTCA, 6-Aug-2014.
26. “MLS & MILS Lower Risk, Allow for Greater Interoperability and Data Sharing in LVC Training,” Modern Military Training, 15-Jun-2017. http://modernmilitarytraining.com/featured/mls-mils-lower-risk-allow-greater-interoperability-data-sharing-lvc-training/
27. “Green Hills Software Operating System Selected by Collins Aerospace for U.S. Navy Tactical Combat Training System Increment II Solution,” Green Hill press release, 21-Jan-2020. https://www.ghs.com/news/20200121_INTEGRITY_tuMP_collins_aerospace_USNavy.html
28. Collins Aerospace, Tactical Combat Training System – Increment II brochure, Apr-2020. https://www.collinsaerospace.com/-/media/project/collinsaerospace/collinsaerospace-website/product-assets/files/public/products-and-systems/simulation/jsas/tctsii-brochure.pdf



Corporate Headquarters

30 West Sola Street • Santa Barbara, CA 93101
ph: 805.965.6044 • fax: 805.965.6343 • email: info@ghs.com • www.ghs.com

European Headquarters

Fleming Business Centre • Leigh Road • Eastleigh • Hampshire S050 9PD • United Kingdom
ph: +44 (0)2380 649660 • fax: +44 (0)2380 649661 • email: info-emea@ghs.com

Safety & Security Critical Products

34125 US Hwy 19 North • Suite 100 • Palm Harbor, FL 34684
ph: 727.781.4909 • fax: 727.781.3915 • email: info-sscp@ghs.com

Green Hills, the Green Hills logo, INTEGRITY, and tuMP are trademarks or registered trademarks of Green Hills Software in the US and/or internationally.
All other trademarks (registered or otherwise) are the property of their respective owners.
© 2020 Green Hills Software. v0520