
Systems and Network Analysis Center
Information Assurance Directorate


# Separation Kernels on Commodity Workstations

11 March 2010

**SNAC** DoD, 9800 Savage Rd. Ft. Meade, MD 20755-6704 410-854-6632 DSN: 244-6632 FAX: 410-854-6604
www.nsa.gov/snac SNAC@radium.ncsc.mil

# Separation Kernels on Commodity Workstations

# 1 Executive Summary

The “U. S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness” (a.k.a., Separation Kernel Protection Profile or SKPP) defines functional and assurance requirements for a specialized operating system known as a separation kernel (SK). The goal of an SKPP evaluation is to verify that there is a robust operating system layer that only provides information control. Unlike many other operating systems that perform a large number of functions, the separation kernel only performs mediation of information flow between partitions. The SKPP has previously been applied to real-time, embedded operating systems, and while a certification of the Green Hills Software INTEGRITY-178B separation kernel has been completed, NSA evaluation efforts of these kinds of systems are ongoing. Unfortunately, the use of an SKPP certified kernel as one part of a system does not immediately make a system in totality highly robust.

The One-Box One-Wire (OB1) project is attempting to use a version of the INTEGRITY-178B separation kernel to provide multi-level separation between virtual machines on common workstations. It was hoped that by using an assurance maintenance process (vs. re-certification from scratch), we could leverage the extensive certification investment already made with INTEGRITY-178B for OB1. However, INTEGRITY-178B was certified with a relatively simple hardware platform in its Target of Evaluation (TOE) and unfortunately, the underlying commodity workstation (as part of a separation platform) does not appear to be appropriate for SKPP certification due to its complexity. This document outlines specific issues and estimates some of the resources that would be required, by using OB1 as an example certification.

Although, the validation of assurance arguments on these systems is not feasible to the degree required by the SKPP, NSA should re-visit the issue when simpler commodity parts exist. Also, this does not mean that the mechanism(s) (security or otherwise) in the commodity workstations today are somehow broken or should be abandoned. Rather, the risk of complex interactions between subsystems within the commodity platforms and the impact on assurance (including the lack of SKPP/high robustness certification) for the system should be understood and may prove acceptable in many situations. Because this finding limits the application of COTS platforms to SKPP and high robustness certification, this document also recommends a frank assessment of solutions that require high robustness to determine if a lower degree of robustness would be sufficient. Also, this document recommends continuing or even increasing support to commodity platform security technologies and the application of separation kernels on commodity platforms, since these emerging technologies do have merit. These efforts should help create a path toward higher levels of robustness in the future.

# 2 Introduction

The SKPP provides requirements for a small kernel in environments requiring high robustness, where a capable and motivated adversary is likely to devote significant resources toward compromising the system. It is important to note, that an SKPP evaluation only benefits the DoD or U. S. Government (USG) if it is successfully leveraged in secure operational systems. Also, it is important to consider whether a given evaluation is likely to succeed at providing a convincing security argument and at being used operationally. Therefore, any SKPP evaluation that results in a false sense of security or a secure but useless system is counter-productive.

This document will refer to high, medium, and low robustness. The SKPP specifically discusses the term high robustness in section 2.8, using the model of threat agents and asset value. It says that high robustness is appropriate when sophisticated threat agents and high value assets make the likelihood of an attempted compromise high.

For comparison, Department of Defense Instruction 8500.2 (DODI 8500.2) defines these terms as follows:

> High robustness – Security services and mechanisms that provide the most stringent protection and rigorous security countermeasures.
> Medium robustness – Security services and mechanisms that provide for layering of additional safeguards above good commercial practices.
> Basic robustness – Security services and mechanisms that equate to best commercial practices.

These terms, especially the term High Robustness, will be used in this document based on the above definitions from the SKPP and DODI 8500.2 "Subject: Information Assurance (IA) Implementation". While the terms do not necessarily imply what specific actions were performed, they do imply a level of confidence, even in the face of an attacker willing to devote resources toward the compromise of the TOE. High robustness would thus imply the most confidence that is realistic for a real system, even when exposed to a motivated attacker that can devote considerable resources toward a successful compromise.

## *2.1 SKPP Evaluation*

In order to obtain certification against the SKPP criteria, a vendor must submit a product to the National Information Assurance Partnership (NIAP). A Common Criteria Testing Laboratory (CCTL) will be contracted to do some validation activities, and NIAP will oversee the process. Because formal methods evidence is required, consultation with IAD will be required for the development of the design and implementation of the formal model. Eventually, the product must be sent to IAD for penetration testing. The SKPP contains explicit requirements that specifically call for an "NSA evaluator" to perform vulnerability analysis, penetration testing, and a review of covert channels. The evaluation effort encompasses these specific activities as well as an unconstrained search for vulnerabilities.

A thorough explanation of the evaluation activities for the SKPP is beyond the scope of this document. However, as discussed earlier, this evaluation includes significant developer effort (including formal verification), CCTL evaluation, and NSA evaluation and penetration testing. In order to provide convincing arguments to all the parties involved and incorporate the requisite verification evidence, a product will almost certainly implement only the minimum required features. This facilitates convincing security arguments about even a very capable adversary, because there is very little to attack and exploit.

## *2.2 Assurance Maintenance*

SKPP certification is applied to a specific TOE, including the software, configuration, and specific hardware platform. Because the certification is so specific, attempts to maintain assurance on other platforms and configurations are expected. However, changes to the source code that implements abstract (i.e., not platform-specific) functions of the kernel are expected to be very rare. Thus, some evidence, such as formal verification, may verify the abstract functions of the kernel irrespective of hardware specifics. This leads to assumptions on the hardware and platform-specific implementation. These assumptions are critical to security, because operations like achieving secure initial state and granting access to memory through the memory management unit depend heavily on this platform-specific code. Assurance maintenance on a new platform is therefore expected to be largely an effort to validate these assumptions.

The originally certified TOE is likely to further simplify evaluation by disabling significant portions of the hardware functionality. This allows as much evaluation as possible to be dedicated to the portion of the kernel that will not change for assurance maintenance activities. As such, this optimizes NSA evaluation resources, because the largest and most feature-rich portion of the originally-evaluated TOE will enjoy the additional resources of the original evaluation and this effort will be reused on maintenance activities. The expectation, then, is that maintenance activities will require minimal resources.

For INTEGRITY-178B, the only separation kernel certified against the SKPP to date, an Impact Analysis Report is required for assurance maintenance. This allows the NSA evaluator to quickly understand the intended effects of changes for this assurance maintenance activity. It is expected that this will permit the evaluator to quickly validate or even test key aspects of the hardware and configuration. In addition, the report will present an assurance argument for the hardware assumptions and external interfaces.

In the final analysis, however, the NSA evaluator will still be responsible for an unconstrained search for vulnerabilities, even on assurance maintenance. If a new platform is used, the evaluator will need to understand the implications of the new platform, using the Impact Analysis Report and other information as needed. It stands to reason that simple hardware platforms will be easier to understand, while complex platforms may require great study.

Regardless of the effort required, assurance maintenance requires that the security arguments be strong enough to convince an evaluator that even a highly capable adversary would not compromise this system in the intended environment. In some cases, it may be the case that platform vulnerabilities prevent certification or similar validation until the vulnerabilities are resolved.

# 3 Commodity Workstation Security

This section outlines the trust relationships between layers of a common workstation platform. It is important to understand these dependencies in order to properly assess their impact on SKPP evaluation.

## *3.1 Workstation Security Argument*

A common workstation consists of many layers. For simplicity, these will be considered to be applications, operating system, and hardware platform. Each layer builds on the properties of the lower layers. In a secure system, the properties of all the layers, taken together, must guarantee that the security policy will hold. In the case of the SKPP, this policy defines information flows between partitions (which are presumed to contain applications).

### 3.1.1 The Applications

The applications are the highest layer of a commodity workstation, forming the portion of the system that the user is expected to interact with directly in order to accomplish mission goals. In the context of separating different partitions, the applications may provide virtualization of guest operating systems, multi-level window management, data processing, communication, or other useful services. In a secure environment, the available applications are expected to be controlled and carefully configured.

Responsible for: Data processing, display, user input, useful features, etc.

Assurance Arguments: largely never change during use, uses specific interfaces enforced by kernel, verification activities

Required Properties of Lower Layer: Operating System enforcing separation

### 3.1.2 The Operating System

The operating system (OS) considered here would be a separation kernel running on a desktop platform. The separation kernel simply provides a software interface for applications and mediates information flow between partitions. It will need to initialize itself in order to create a secure initial state, and from that point all operations must preserve the security properties.

Responsible for: Separating partitions and controlling information flow

Assurance Arguments: Original SKPP evaluation, assumptions validation

Required Properties of Lower Platform Layer: platform correctness, platform separation, secure initial state (e.g. proper isolation of the components and their access that must inherently share the same memory with other components)

### 3.1.3 The Hardware Platform

The desktop platform hardware is the lowest layer of the workstation. It is used by the operating system in order to interact with devices, initialize the system without specific knowledge of each and every hardware manufacturer's configuration, and sometimes to perform trusted security functions. It also includes input and output devices, such as the keyboard, mouse, and network interface. The platform is implemented with both firmware and hardware, and it is usually developed and maintained by separate vendors from the operating system and applications. The platform usually includes a Basic Input Output System (BIOS), which standardizes interaction with hardware devices on the platform, facilitating operating system interaction with these devices. The BIOS initializes platform hardware before passing control over to an operating system, and it therefore controls the initial configuration for the OS. Recently, platform manufacturers have added platform security technology in order to mitigate some threats that apply to the platform firmware and hardware. This section specifically addresses Intel® technologies that have been published and are currently available in new systems, some of these features have been considered for use in OB1. Competing manufacturers such as AMD™ have similar capabilities.

Responsible for: Secure initial state (device initialization and configuration), operating as expected (for example, enforcing the permissions configured by an operating system), separating different devices

Required Properties: Manufacturers' completeness/verification (BIOS, Original Equipment Manufacturer (OEM), devices, integration), platform's security itself

Security Technologies: BIOS, Intel® VT-x, VT-d, TXT, ME (AMT), TPM, etc.

## *3.2 Evaluating the Workstation Security Argument*

From the above description of the various layers, it should be apparent that a thorough evaluation of any layer is a significant undertaking. In fact, the goal of an SKPP evaluation is to verify that there is a robust operating system layer that *only* provides information flow control. Other security critical features will be required, and these will appear in the application layer, requiring additional evaluation effort. However, the expectation is that a single (albeit expensive) evaluation of a separation kernel will result in a very robust component that can be reused with many groups of applications.

Unfortunately, the operating system layer relies critically upon the hardware, and the complex hardware of commodity workstations is very imposing. Because hardware mechanisms have the ability to bypass even a perfect operating system, these complexities cannot be ignored at the highest levels of robustness. Therefore, the already difficult and resource intensive task of high robustness operating system evaluation on a commodity workstation is further complicated by a necessary and complex hardware evaluation.

## 3.3 Too Many Cooks in the Kitchen

Ultimately, the problem with commodity desktop platforms comes down to the fact that too many developers and vendors are interdependent. Each organization involved in the creation of a desktop workstation has an economic interest in adding features to distinguish their version from the competition and often needs a particular, unique, and potentially powerful access. This prevents a unified strategy for security from emerging. As an example, a single workstation is likely to involve the following components and manufacturers:

- CPU/microcode – Intel®, AMD™, VIA™, etc.
- Chipset hardware/firmware – Intel®, AMD™, VIA™, etc.
- Core BIOS – Phoenix®, American Megatrends®, Insyde®, Tiano Core, etc.
- Motherboard and BIOS Customization – Dell®, HP®, etc.
- BIOS modules/expansion ROMs – any PCI card vendors
- SuperIO Chip – Winbond®, ITE™, SMC®, etc.
- Hard Drive/Firmware – Seagate®, Western Digital®, etc.
- SPI Flash – Winbond®, Atmel®, etc.

As multiple parties (chip manufacturer, motherboard manufacturer, BIOS developer, device manufacturer, etc.) attempt to add value, the product will be subject to *orthogonal feature creep*. Put together, these extra features create complexity. In order to facilitate interoperability, additional layers of interdependency are added, requiring trust in multiple parties. It is understandably difficult, for example, for one organization to understand, verify, and correct bugs that have been created by another organization, especially since the standardized abstraction does not necessarily represent the real hardware. This is especially difficult when one organization's "bug" is considered to be a "feature" by others.

Even if good security arguments could be made, they would only apply to a single hardware version. This version will then have to deal with limitations such as reliance on particular manufacturers, resistance to change, and obsolescence. It will probably no longer be considered "commodity" equipment, and even if it is, that will only last until the next version is released. Current platform security technology (such as Intel® TXT) is attempting to address this, but an assurance argument about the platform security technology itself is needed. It appears that such verification activity must occur on (and apply to) a particular platform, thereby losing any "maintenance" advantage for commodity platforms.

## 3.4 Known Platform Vulnerabilities

A summary (not exhaustive) of recent research into platform attacks is given in Appendix A. The particular attacks described have been patched or are well-known and normally mitigated through good security practices. These attacks exemplify a fundamental issue: "too many cooks in the kitchen" and the complexity of *orthogonal feature creep* leads to unverified security dependencies. For example, BIOS is used to

abstract the hardware specific interfaces of commodity platforms. Without it, the operating system would be hardware dependent. However, reliance on BIOS creates a trusted interface for an operating system, and this trusted interface has been exploited in publicly known attacks (see Appendix A) using the BIOS, SMM, and even TXT. A similar argument can be made for other attacks described.

As one manufacturer adds features to a component, the security implications for other components are difficult to understand. New features, which change the trust model, may not be designed with the requirements of other components in mind. For example, the designers of a device that writes directly into host memory might not have envisioned the notion that an entity with access to the device's control registers does not also have access to all physical memory. This would likely make the secure configuration of Intel® VT-d very difficult. Similarly, the addition of firmware update features to one device may significantly change the trust model of the system, depending on how the update is performed and what the device can do.

In order to evaluate the secure use of such a platform, the trust models of all these various components must come together, forming a consistent security argument. Even though some of the vendors involved may participate in the evaluation and certification activities, it is unlikely that all parties who have contributed intellectual property (e.g. firmware, source code, etc.) to the platform will release that information to NSA for evaluation. This forces the evaluation to require hostile reverse engineering, assurance arguments that are independent of this information, or an understanding and acceptance of risk. Reverse engineering is almost certain to be prohibitively expensive for evaluation, and assurance arguments that do not require this information will likely result in hardware platform changes. The only tractable solution is to understand and accept the risk of security flaws in the platform. Depending on the level of understanding, this strategy is probably appropriate for basic or medium robustness only.

Many recent hardware platforms, especially embedded devices, have begun to support special interfaces that are available only to the manufacturer. These interfaces are often undocumented and protected. This may be used to enforce a degree of content copy protection or prevent alteration of the device functionality. For example, a digital rights management system might use an undocumented and protected interface to load firmware to a playback device. Unfortunately, undocumented and protected features directly conflict with the goals of evaluation. If the hardware platform contained such a device, how would evaluators assess the security impact of these undocumented interfaces? This is especially difficult if the manufacturer is not participating in the evaluation and does not wish to disclose the existence of these interfaces.

Some hardware platforms also support additional features that can be enabled or disabled by the manufacturer. Often this is used as a cost-saving measure, allowing the manufacturer to create one hardware design and offer a range of products with specific features enabled or disabled in order to increase or decrease the price. This creates additional difficulty for security evaluation. While a feature may be disabled in order to

simplify a platform, this is fundamentally different from the feature not being present at all in the platform. If features can be enabled or disabled by a manufacturer, this can also be done by an attacker, and it may invalidate the security arguments of the system.

## *3.5 Problems for SKPP Requirements*

Evaluation of a separation kernel involves significant effort, even on ideal hardware. The complexities of real hardware, especially commodity workstation hardware and firmware, make successful evaluation even more difficult. While many aspects of the SKPP may be affected by the issues presented in this document, the following specific requirements appear to be the most likely to fail evaluation as a result of incomplete evidence or the existence of vulnerabilities.

### 3.5.1 Trusted Initialization (ADV_INI)

Because it is the responsibility of BIOS to handle hardware-specific details, evaluators will have difficulty precisely understanding its role in trusted initialization. With a significant investment of time and effort, it might be possible to identify the effects of specific actions performed by a particular implementation on a particular platform. However, this will not easily scale to many platforms, and it will likely require more evaluation resources than are available. Moreover, the only component developed by the vendor attempting SKPP certification is the separation kernel. If a vendor did not write the BIOS, it is not necessarily going to be available for evaluation. Similarly, the necessary hardware datasheets and board specifications may not be available for validation of BIOS actions. (Appendix G of the SKPP states that, for APT_PDF_EXP – Platform Definition, “This PP mandates the highest level which requires that detailed specifications for all components be available.”) In a commodity workstation, neither the hardware nor the BIOS were developed to be simple with highly robust assurance arguments and yet the SKPP vendor must rely on these components to make their case.

Intel® TXT is intended to ensure that even if BIOS execution did not result in a secure state, the system will be secure if the TXT launch succeeds. However, SKPP certification would then be contingent on a certification of Intel® TXT, and the resources required for this make it impractical in the context of an already significant SKPP evaluation. There are many complex dependencies between the CPU, Authenticated Code module, and BIOS/chipset. One such dependency was insecure and exploitable in the ITL VT-d attack described in Appendix A. Furthermore, an evaluation of Intel® TXT must occur on a particular motherboard, because the details of the verification performed by TXT must be evaluated for correctness. This would limit the certification to a particular commodity motherboard, but might make it possible.

Another solution might be to customize a BIOS implementation for evaluation on a particular platform. This would be roughly equivalent to writing the Board Support Package (BSP) on other assurance maintenance activities. Unfortunately, this requires specific knowledge of the exact motherboard, the devices on it, and their configuration. The purpose of BIOS is to abstract this, because there are so many vendors in the

commodity workstation market. If this were done, only one specific and probably customized motherboard would be compatible. This would make the solution no longer a commodity workstation, but might make evaluation closer to the intended assurance maintenance.

Both cases would still need to handle the issue of BIOS updates. Bug fixes and additional features will require a mechanism to deploy new versions. This mechanism must not allow adversaries to corrupt the integrity of the BIOS, and the integrity check must be robust against a highly capable adversary. This update mechanism must be compatible with the SKPP requirements for flaw remediation (ALC_FLR).

### 3.5.2 Platform Assurance (APT)

The SKPP contains specific requirements for Platform Definition (APT_PDF), Specification (APT_PSP), Conformance Testing (APT_PCT), Security Testing (APT_PST), and Vulnerability Assessment (APT_PVA). Fulfilling these requirements requires a thorough description of all of the platform components that the TOE will rely upon. This information will be used to construct documents and tests for both internal and external resources that are part of the platform. Normally, this information would come from hardware data sheets and other vendor-specific information.

Section 2.6 of the SKPP discusses platform considerations, including components and interfaces. While it provides examples of more and less restrictive platform definitions, it also cautions that less restriction complicates evaluation. Identification of all the interfaces and devices on a commodity workstation will be one such complexity. The platform definition must enumerate the devices on the platform and document their configuration. This requires significant effort, either requiring the developer to fully understand a commodity motherboard or create a simplified motherboard that is not commodity. Without this sort of understanding, an enumeration of components and interfaces does not appear to be possible.

Orthogonal feature creep (e.g. innovation) will be perpetual and rapid. Without an ongoing effort, closely tied in with a particular model of chip and motherboard, it will likely be not feasible to gain sufficient assurance evidence on a system before it becomes obsolete.

Recent platform security features such as Intel® TXT claim to protect much of the critical platform interfaces. However, if security arguments of the separation kernel rely upon this, then Appendix G of the SKPP states that these hardware mechanisms “cannot be considered part of the platform and must be evaluated in accordance with the non-platform assurance requirements (e.g., ADV).” As stated above, this amounts to an evaluation of the platform security features (such as Intel® TXT) along with the SKPP evaluation of the kernel.

A potential solution might be for the USG to create demand for a greatly simplified “secure version” of a commodity platform with evaluation closely tied in with development of this particular model of chip and motherboard. Each time this model is

upgraded to include additional features, evaluation would be updated as well. In order to prevent this product from lagging too far behind the regular market, perhaps it could always be based on the newest version on market but with limited features. Clearly, this would require very close and ongoing relationships between NSA and OEMs. It would also require a lot of dedicated resources (not a one-time cost), but this is likely to be less costly than the evaluation of commodity parts in general.

### 3.5.3 Covert Channel Analysis (AVA_CCA)

In order to perform Covert Channel Analysis, the developer must consider the interactions between devices on the platform. This is especially true for devices which might become exported (allocated to potentially untrusted partitions) or external (available to potentially untrusted subjects outside the kernel, such as other hardware devices or systems) resources. Like the previous issues with Platform Assurance, a detailed specification of the platform hardware is required. It will be hard to obtain this information for a commodity platform, since it comes from a different vendor than the one seeking certification of a separation kernel.

Once this documentation is obtained, performing a systematic analysis on it will be extremely complex. Commodity platforms have numerous microcontrollers for a large number of devices, busses, and peripheral interfaces. Simply determining which devices are interconnected and the amount of information that can flow through these connections will require iterating over a very long list of devices. Performing this analysis on a commodity motherboard would certainly be enlightening, but it is likely to require a lot of time and effort that might be better spent elsewhere. If a significant number of high-bandwidth covert channels are found, the TOE may not be able to meet its functional requirements for separation, and in that case, it will be difficult to fix the problem. Even if it is not the case, this analysis would likely be invalid for any new version of the platform, and the evaluated version would likely become obsolete quickly.

# 4 One-Box One-Wire (OB1)

The One-Box One-Wire project (OB1) is a good example of a desktop workstation that uses a separation kernel. One of the project's goals is to use an SKPP certified version of INTEGRITY-178B on a desktop workstation that will separate multiple classification levels in virtual machines. This kernel will also use Intel® platform security technology (including VT-x, VT-d, and TXT).

OB1 can be broken down into the following components:

1. Intel® Hardware Platform
2. Separation Software (kernel and user-level)
3. BlackChannel Network Card & Switch
4. System Management

OB1 benefits from an architecture that carefully selects the trusted properties of its components to be those properties that are expected to have the most rigorous verification. For example, the workstation assurance arguments boil down to the separation kernel's information flow control properties whenever possible, and the network and configuration assurance arguments largely rest upon the protected portion of the BlackChannel hardware. This is intended to maximize the utility of costly but rigorous verification activities such as SKPP certification.

USCENTCOM, the intended customer for OB1, has requested NSA support for OB1 evaluation, especially with regard to SKPP evaluation and assurance maintenance. In response, a number of activities have been performed, including the creation of this report.

## *4.1 Evaluation of OB1*

The architecture and verification activities planned for OB1 provide significant assurance arguments, and only some specific features that distinguish OB1 are mentioned here. First, OB1 plans to use a separation kernel to enforce separation between guest virtual machines. The use of a simple separation kernel supports verification activities (such as high-coverage testing or formal methods) of the desired separation property. In addition, OB1 plans to provide virtualization support using independent user-space processes for each virtual machine as much as possible. This reduces the shared resources between virtual machines and makes privilege escalation more difficult for an attacker. Another critical component of OB1 is the use of the BlackChannel network card and switch, which provide hardware-based, encrypted tunnels for each virtual network. These components are designed to prevent attackers from retrieving key material, affecting the hardware operation, or confusing virtual networks. Because OB1 developers have thought through security arguments at many levels, NSA analysts believe the project has great potential.

A rough order of magnitude estimate for the evaluation work needed was created by NSA analysts familiar with SKPP evaluation. The goal was to provide an estimate for a reasonably thorough evaluation of OB1 (both for SKPP certification and in general)

while accepting that full verification was not feasible. Given the known attacks and current understanding of the OB1 components, a set of evaluation activities was composed for each one. This includes activities like evaluation of SKPP requirements, source code analysis for specific applications, design review, evaluation of the use of Intel® TXT, vulnerability analysis of platform devices, and other items. A rough estimate, in multiples of two-week iteration cycles, was given for each activity.

All together, the work is expected to require roughly 8.4 man-years, spread across multiple organizations with relevant expertise. Even if some activities are dropped, the work estimate will still remain high until many components are not thoroughly examined at all, and even with all of the planned and estimated activities, it is not clear that a security argument could be formed that justifies use in cases where a dedicated adversary may attempt to compromise the system (i.e., High Robustness). Significant coordination among different organizations will be required in order to meet the very tight schedule for the OB1 project. There is some risk of delays due to setting up environments for multiple organizations and communication of relevant information.

# 5 Conclusions

Analyses of the desktop workstation platform, SKPP, and OB1 have resulted in a number of general conclusions. These conclusions are described for SKPP evaluation in general and OB1 in particular, and recommendations are provided to enhance future work.

## *5.1 SKPP Evaluation on Commodity Workstations*

The commodity workstation platform uses components from multiple vendors, and many of these components are relied upon for trusted initialization. Attacks that take advantage of vulnerabilities in these trust relationships have been publicly presented. In response to many of the attacks against these platforms, manufacturers have begun including platform security technology. While this technology is promising, the platform complexity makes it very difficult to verify its correctness, and any attempt to do so would verify only a particular platform, limiting the scope and scalability of the solution.

Because of the complexity and evaluation difficulty, SKPP evaluation on a commodity workstation would be costly in time and effort, limited in applicability (only a single motherboard could be verified), and unlikely to result in confidence that justifies certification against the SKPP. Because the SKPP is specifically targeted to environments requiring high robustness, the adversary is expected to have invested significant resources into platform vulnerabilities and attacks. Without fundamental changes to normal commodity platforms, it is unlikely that any reasonable evaluation effort could provide enough assurance to justify certification of such a platform against the SKPP.

A corollary to this conclusion is that the SKPP and high robustness have a limited scope. In order to achieve sufficient simplicity to allow for proper assurance, only tightly controlled systems are appropriate. Standard commodity equipment is likely to be acceptable only in the form of simple components combined in simple ways. There will be practical limits to what can and cannot be simplified, while still resulting in a useful end product. However, because a large portion of commodity equipment that is familiar to end users will not be sufficiently simple for SKPP evaluation, end-user solutions that meet these requirements will probably be highly customized and difficult to scale.

Nonetheless, commodity workstations may present a completely acceptable risk profile given available options. The fact that sufficient assurance to justify SKPP certification or high robustness is not feasible on these platforms does not change operational needs and available resources. It will be necessary to balance risk aversion with a realistic assessment of current capabilities. In doing so, iterative improvement in the robustness of DoD systems should be the goal. Therefore, the findings in this document do not condemn OB1 or the use of separation kernels in commodity workstations. Instead, they should drive improvements that make higher levels of robustness possible in the future.

## 5.2 OB1 and Desktop Virtualization

The OB1 project is performing potentially useful work by applying separation kernel technology to desktop virtualization. The Information Assurance Directorate's (IAD's) effort to understand the architecture and assurance arguments for the system has revealed that much thought has been placed into developing a system with strong arguments for its security. While there is a great deal of work needed to improve OB1, it appears to be a promising application of separation kernels.

Unfortunately, the desktop platform targeted by OB1 undermines SKPP certification, due to the requirement for high robustness. A useful system, incorporating the platform technologies that OB1 already plans to use, could probably be built if the target was in environments requiring medium robustness. Reducing this requirement would also have the benefit of making the solution easier to scale, and it does not necessarily imply anything with regard to an accreditation decision to permit operation with specific networks having different classification levels.

Therefore, attempting assurance maintenance against the SKPP on INTEGRITY-178B using a desktop workstation platform is unlikely to be successful for OB1. In order to continue with the current platform and architecture, OB1 should consider dropping the SKPP certification and corresponding high robustness requirements. Otherwise, if SKPP certification and high robustness are required, then a new architecture, based on simpler components, should be developed.

This does not necessarily prevent OB1 from obtaining approval to operate in its intended environments. Other systems have been approved for connection to networks of different classification levels without SKPP certification. However, these issues should be considered as part of a realistic assessment of the risk that OB1 is expected to assume. Given available options, this may even be an improvement in the overall assurance of the system, because nearly all commodity equipment will be subject to these risks. Even if the project does not address the platform assurance issues to the level required for the SKPP and high robustness, it is still possible for OB1 to make significant improvements as compared with the average commodity workstation. Depending on the customer's environment and needs, this may be sufficient for operational use.

The security argument of the proposed OB1 system is based on both commodity platform security technology and a separation kernel. This potentially offers greater assurance than current systems that provide separation using virtualization, even though it does not meet the requirements for the SKPP. The use of a separation kernel should reduce the complexity of the kernel component and facilitate robust verification activities. A simple and robust kernel does not create simplicity and robustness in other layers of the platform (i.e., applications and hardware), but it does represent an improvement. Therefore, it is recommended that NSA attempt to support as many of the planned and estimated evaluation activities as possible. The resulting evaluation should help improve desktop workstation security in general and the OB1 product specifically, advancing the missions of NSA and IAD.

## 5.3 Recommendations

For high robustness environments, where a highly capable and motivated adversary is expected to devote significant resources toward compromising the system, it is recommended that complex platforms such as desktop workstations NOT be relied upon. A highly capable adversary is expected to research platform vulnerabilities in commodity workstations, and the platform security of this technology is subject to significant complexity, which stands in the way of robust assurance arguments. Improvements to platform security technology should continue, and as it matures, this technology may be appropriate for many practical situations. However, without significant simplification of the platform, it does not appear that the security arguments of this technology can justify high robustness. Therefore, these systems are not appropriate for certification against the SKPP.

Because platform security technology is developing and appears promising, it is recommended that NSA and other USG organizations continue to follow this technology, encourage its general use, and support its improvement. This technology has the potential for generating high impact by raising the bar for computer security in general. The significance of this suggests that investment in studying and improving this technology may be applicable to current and proposed DoD systems more often than SKPP and high robustness solutions, even though it would not be able to provide sufficient platform assurance for SKPP certification. Organizations involved in the OB1 project should consider working with platform manufacturers to make simpler, supportable platforms. If simpler commodity platforms can be easily used to create the desktop environment targeted by OB1, it may be possible to support SKPP certification and high robustness in the future. Such a platform would need to include both evaluation and product support on a continuing basis in order to maintain assurance with future, updated versions.

Lastly, when considering engagement of a product, the scalability of the successfully evaluated solution must be considered. NSA may need to review the environments that necessitate high robustness and assess the need for scalability. If the highest level of confidence (and not some lower level) is truly required in many real DoD environments, a strategy for scaling high robustness solutions to the level required by the DoD should be developed. In any case, security solutions should ensure that future developments proceed along a path toward supporting higher levels of robustness.

# 6 Appendix A: Public Platform Attacks

## *6.1 BIOS Attacks*

**Persistent BIOS Infection** (Core Security Technologies, CanSecWest 2009) – Security researchers from Core Security Technologies have published a generic attack that provides rootkit-like behavior by modifying the BIOS image. They identify the flashrom tool, which supports rewriting many motherboard and flash chip combinations as part of the CoreBOOT project. By patching a BIOS and fixing the checksums, they demonstrated attacks targeting OpenBSD and Windows with persistence against OS reinstallation.

**Attacking the Intel BIOS** (ITL, Black Hat USA, July 2009) – The security researchers at Invisible Things Lab (ITL) have published a technique for rewriting the flash memory used to boot the BIOS for an Intel® platform. This memory would normally be protected from writes, but by exploiting an overflow in BIOS image parsing, they were able to execute arbitrary code before the BIOS locked the ability to write to the flash memory. This results in the ability to arbitrarily and persistently modify the BIOS code, executing malicious code before booting an operating system boots.

**SMM Rootkits**: A New Breed of OS Independent Malware (University of Central Florida, SecureComm 2008) – Security researchers presented a proof of concept System Management Mode (SMM) rootkit. SMM is a special x86 processor mode, which can transparently execute privileged code. After the rootkit was installed, they demonstrated its ability to hide from the operating system, log keystrokes, and send data out over the network. The work did not outline how to get the SMM rootkit installed through exploitation of secure components (they installed the rootkit by manually executing a malicious kernel driver), but other research has since demonstrated this ability.

**Attacking Intel Trusted Execution Technology** (ITL, Black Hat DC, February 2009, details presented at Black Hat USA, July 2009) – ITL identified a portion of SMM code that used a pointer to memory that was accessible by the host. By exploiting this dependency, they were able to trigger execution of arbitrary malicious code from within SMM. They used this to successfully bypass Intel® Trusted Execution Technology (TXT), which is intended to ensure the integrity of system initialization.

**Attacking SMM Memory via Intel CPU Cache Poisoning** (ITL, March 2009) – ITL demonstrated that an attacker with kernel privilege can attack SMM code by manipulating the processor's machine-specific registers to make the protected SMRAM cacheable with write-back. Then, by performing writes to these memory locations, the attacker would fill the cache with modified values for SMRAM. Finally, after triggering execution of a System Management Interrupt (SMI), the CPU will execute code from the cache, because new values have not yet been written back. This will permit arbitrary code execution from within SMM.

## 6.2 Device Firmware Attacks

**Implementing and Detecting a PCI Rootkit** (NGSSoftware Insight Security Research, Black Hat DC, November 2006) – The author described an attack using PCI expansion ROM, causing malicious code to execute during system startup. This malicious code implemented a rootkit, and was capable of remote updates through PXE boot. The author suggested detection of such rootkits by observing the effects of the rootkit on the operating system and/or comparing the expansion ROMs to known good versions. The author also suggested the use of the Trusted Platform Module (TPM) and BIOS to protect against such attacks.

**Project Maux Mk.II "I 0wn the NIC, now I want a shell!"** (Arrigo Triulzi, PacSec 2008) – The researcher first described previous work that installed a rootkit into the firmware of a network interface card (NIC). The rootkit allowed covert sniffing of network traffic. The new research presented was applying the same firmware modification techniques to a new hardware device—the graphics card. Using the compromised NIC from the previous work, commands are issued to the graphics card, resulting in a secure remote terminal available to the adversary. This covert, malicious service was independent of the OS or application software. Some defenses, including firmware verification and secure boot, were suggested.

**Introducing Ring -3 Rootkits** (ITL, Black Hat USA, July 2009) – ITL described an attack that caused malicious code to be executed from the Intel® Manageability Engine (ME), which is responsible for Intel® Active Management Technology (AMT). In order to obtain execution on the ME, they reverted to an older, vulnerable version of the Intel® BIOS and used a remapping attack that they had previously presented, making a portion of the ME memory accessible to the operating system. They described how the ME could make a direct memory access (DMA) between itself and the host memory, allowing attacks against the host operating system.

**Reversing and exploiting an Apple firmware update** (Georgia Institute of Technology, Black Hat USA, July 2009) – The researcher presents and describes in a paper how the firmware update mechanism of the Apple Aluminum Keyboard may be used to implement a rootkit. Because low-cost components are used in such devices, it is difficult to perform computations needed for strong verification such as a cryptographic signature. Therefore, this sort of attack will be difficult to mitigate. The rootkit developed shows the caps lock LED being flashed when a keyboard is plugged in, but a malicious payload could send keystrokes to the operating system. This attack would survive reinstallation of the operating system.

## 6.3 Device Hardware

**0wned by an iPod** (Laboratory for Dependable Distributed Systems, PacSec 2004) and **Firewire: all your memory are belong to us** (LDDS, CanSecWest 2005) – These presentations detail the vulnerability of the Firewire interface, when not filtered, to malicious devices that attempt to read and write arbitrary memory locations. They demonstrate that these attacks can be effective on real operating systems. As a defense, they suggest implementation of Firewire filtering.

**Remote Code Execution through Intel CPU Bugs** (Endeavor Security, Hack-In-The-Box 2008) – The researcher presented examples of Intel® processor errata that can be used for malicious attacks. He suggested that cache coherence bugs could be used to cause everything from a crash to injecting attack code into an operating system kernel. He also identified specific errata that have been identified in malware and can be used to assist in local privilege escalation and obfuscate reverse engineering.

## *6.4 Platform Security Technology Attacks*

**Attacking Intel Trusted Execution Technology** (ITL, Black Hat DC, February 2009, details presented at Black Hat USA, July 2009) – As discussed above, ITL demonstrated that SMM attacks are successfully able to bypass Intel® TXT. Note that Intel® has plans to create an SMI Transfer Monitor (STM), which should create a controlled, virtual environment similar to a virtual machine for SMM code. This should limit the capability of SMM attacks in the future, but no STM is publicly available at the time of writing.

**Another Way to Circumvent Intel Trusted Execution Technology** (ITL, December 2009) – ITL published a paper detailing how a dependency between Intel®'s Authenticated Code Module (which performs trusted operations in order to implement a secure TXT launch) and the platform configuration tables can lead to insecure configuration after a successful TXT launch. By manipulating the system's ACPI table, it is possible to "trick" the authenticated code module (and subsequent operating system code) into using incorrect configuration registers for one of the VT-d remapping units. This results in VT-d not being configured, leading to vulnerability to traditional DMA attacks from or through devices, even though TXT was explicitly intended to protect against this attack.

# 7 Appendix B: Terminology

**Assumptions:** Fundamental properties that are relied on to be true about a system.

**Assurance argument/Security Argument**: Evidence (e.g. claims) that a system meets a security requirement/assumption. Loosely used and interchanged.

**Assurance maintenance**: A NIAP process that allows evaluated products to maintain their Evaluation Assurance Level (EAL) rating when being updated in a well-defined, controlled and usually limited way.

**Commodity desktop workstations**: Typical desktop platforms (x86 based) made by the usual Original Equipment Manufacturers (OEM) (e.g. Dell, HP, Apple, etc).

**CPU**: Central Processing Unit (e.g. x86, PowerPC, etc).

**Chipset**: the supporting chips included on the motherboard that allow the CPU to be useful. These are not the peripheral devices such as hard drives, NICs, etc.

**Firmware**: This is software that is usually loaded from flash memory and is often associated with devices. It may run natively on the CPU, or natively on the device.

**BIOS**: Basic Input Output System. This is firmware that boots the platform and establishes the initial configuration of the platform as warranted by the OEM. It manages power, temperature, configuration and other features inherent to the platform. It is not the operating system.

**SMM**: System Management Mode. This is a unique mode of x86 CPU that operates in a separate context from the OS. It is firmware that is setup by the BIOS at boot and is entered via system interrupts. It is used to handle things like legacy devices, power management, platform fixes after shipment, and other features inherent to the platform. It is not the operating system.

**Intel® ME**: Manageability Engine. The ME is a separate CPU and subsystem within the platform chipset. This may be marketed also as vPro. The ME is an Intel® chipset-specific technology built-in to many enterprise systems shipping today.

**Intel® AMT**: Active Management Technology. AMT is firmware running within the ME to facilitate remote management of the platform.

**Intel® VT-x**: A set of instructions of the Intel® CPU that enable virtualization. Typically a virtualized solution consists of a controlling entity (e.g. Virtual Machine Monitor a.k.a. hypervisor) and any number of subordinate virtual machines all running on the same CPU.

Intel® VT-d: Virtual DMA protection. This allows the CPU to restrict DMA-capable (direct memory access) devices to specific virtual address ranges. Typically used by the controlling entity (e.g. OS, Hypervisor, etc.) to isolate/protect memory of the specified container (Virtual Machine, process, OS, etc.).

TXT: Trusted eXecution Technology. This is specific set of Intel® x86 CPU instructions that collectively allow a process to start/exit in a known (i.e. secure) state. Among many things, it makes measurements of machine state (BIOS, devices, etc.) and stores them in a Trusted Platform Module (TPM).

TPM: Trusted Platform Module. This is a separate chip on the motherboard that appears as a PCI-device (to the OS it's just another "card") and is used for cryptographic operations. It is enabled/disabled from the BIOS at boot. It is a standard of the Trusted Computing Group consortium.