RC 23624 (W0506-041), 6 June 2005 (Revision 1 19 October 2005)
Computer Science

IBM Research Report

Multi-Level Security Requirements for Hypervisors

Paul A. Karger
IBM Research Division
Thomas J. Watson Research Center
P. O. Box 704
Yorktown Heights, NY 10598, USA

This paper has been accepted by the 21st Annual Computer Security Applications Conference, to be held 5-9 December 2005 in Tucson, AZ. It will appear in the conference proceedings, published by the IEEE, and downloadable from http://www.acsac.org

Multi-Level Security Requirements for Hypervisors

Paul A. Karger
IBM Thomas J. Watson Research Center
P.O. Box 704, Yorktown Heights, NY 10598, USA
karger@watson.ibm.com

Abstract

Using hypervisors or virtual machine monitors for security has become very popular in recent years, and a number of proposals have been made for supporting multi-level security on secure hypervisors, including PR/SM, NetTop, sHype, and others. This paper looks at the requirements that users of MLS systems will have and discusses their implications on the design of multi-level secure hypervisors. It contrasts the new directions for secure hypervisors with the earlier efforts of KVM/370 and Digital's A1-secure VMM kernel.

1 Purpose of this paper

There have been a number of recent efforts to develop multi-level security (MLS) for hypervisors or virtual machine monitors (VMMs), such as NetTop [39], sHype [43], and a proposed combination of Xen [16] and sHype [32]. There has been a lot of confusion about what the requirements are to adequately support multi-level security (MLS) in a hypervisor. The hypervisor is being used to separate multiple instances of untrusted operating systems, running at different security levels. The purpose of this paper is to clarify what end-users of MLS expect to be able to do[1], and what technical issues impact those requirements at Common Criteria levels EAL4 and above.[2] This paper presents no major new ideas or innovations. The goal is to assist developers of hypervisors to decide which of these ideas and features are important to make multi-level security useful to the end-users. A hypervisor with fewer features is less expensive to build and is easier to evaluate under the Common Criteria. However, if the hypervisor is too restrictive, then the customers will be unable to implement the MLS applications that they want to run. This paper identifies a set of features that are needed to make the hypervisor useful, yet are still simple enough to assure its security.

2 End-User Expectations

2.1 What does Multi-Level Secure Mean?

MLS systems can mean many things to many people. What this paper will describe are the requirements and implications of a multi-level secure mode of operation as was defined many years ago in DoD Directive 5200.28 [11] and in the implementing manual [13].[3] A system that runs in *multi-level secure mode* has information at a variety of classification levels[4], but not all users are cleared for all information. By contrast, most classified systems in the DoD today run in *system-high mode* and have information at a variety of classification levels, but all users are cleared for the most sensitive information in the system. The system may be a single machine or an entire network. For example, the DoD's SIPR network stores information marked from Unclassified through Secret, but all users are required to have at least a Secret clearance. There is also a *controlled mode* of operation in which all users are cleared to some level, but not necessarily the highest level of information. The first successfully deployed controlled mode

[1] The end-user requirements are derived from the author's personal experience designing, deploying, and supporting a variety of MLS systems within the DoD and in designing high security hypervisors.

[2] Trusted Information Systems, Inc. developed a proposed interpretation of the Orange Book for Virtual Machine Monitors [10] that attempted to clarify some of these issues, but it did not address the networking issues on which this paper particularly focuses.

[3] A more modern version of these definitions can be found in [6]

[4] This paper will speak of security levels in most cases to make the language simpler. However, using only hierarchic security levels is an over-simplification of the model. The DoD security model is actually a lattice-structure with both levels and categories. A point in the lattice is usually called an access class, and any pair of access classes may be comparable (<, =, or >) or they may be disjoint and totally incomparable. See [17] for details.

system was the Multics system at the Air Force Data Services Center in the Pentagon that processed Top Secret information, but allowed users who were only cleared for Secret. Under the old Orange Book evaluation system [5] and as recommended in the Yellow Books [4, 12], system-high systems were typically evaluated B1 or below. Controlled mode systems were typically evaluated at B2, and true multi-level mode required B3 or higher. Translating to the Common Criteria [7-9], B1 and below are roughly EAL4 and below, B2 is roughly EAL5, and B3 and higher are roughly EAL6 and higher.

This paper has focused on the use of MLS for the defense applications, but they are by no means limited to defense applications. MLS can be extremely useful in commercial applications. An example use of MLS in a frequent-flyer smart card application is shown in [28, 29]. IBM has developed a new extended mandatory access control model, designed to provide multi-organizational MLS in a meaningful way to the entire Internet. This is described in [24] and in section 3 of [46]. The development of multi-organizational MLS for commercial use also has payoffs for the military. Traditional military MLS models have been single-organization models. Everyone in the Department of Defense follows the same security rules. However, this traditional single-organization model has problems when multi-national coalition forces must work together. Each country's military has its own security policies, and those policies do not easily map into a single policy. By contrast, IBM's multi-organizational MLS, designed to handle many different businesses on a single world-wide Internet, is much better suited to modeling the many different security policies of multi-national coalition forces.

2.2 What do Users Want to do with MLS Systems?

The most basic requirement is that the MLS system keeps highly classified information from leaking to people who are not properly cleared. This requirement is met by a system that implements the Bell and LaPadula security model [17]. However, this requirement can also be met by simply keeping data of different classifications on different computer systems and restricting access to those systems by clearance levels. Most systems in the DoD do exactly that and run in a system-high mode.

The biggest problem with system-high mode is that sharing information across security levels is very hard. Users at high levels of security want to be able to read low-level information, even though they do not want to contaminate that low-level information with high-level secrets. Keeping multiple copies of the low level information on different machines running at different system-high levels is not acceptable. First, you need to have significantly larger amounts of storage in such a case, and keeping the data synchronized can be very difficult. If you update the low-level data on a low-level machine, that update must be replicated onto all the other copies. Such replication is particularly difficult, because machines running at different system-high levels must NOT be networked together. The DoD frequently has to resort to *sneakernet* to apply these types of updates.

Users also want to downgrade information from higher security levels to lower security levels. The simplest form of this is the statutory downgrading required after the passage of specific numbers of years. Since statutory downgrading only happens after multiple decades have passed, there is little need to make it happen in real time, although there is a need for efficiently downgrading large numbers of files from archival storage.

However, there is another form of downgrading that does need to be done quickly and in real time. A user at a high security level may determine that a particular piece of information needs to be made available to someone at a lower security clearance. For example, an intelligence analyst may determine from a spy's report that the enemy is going to attack at dawn. The defenders who need to know about the upcoming attack, but those defenders should not know who is the spy. The analyst must sanitize the information, removing any indicator of who the spy is, but leaving the information that the enemy will attack at dawn.[5] The analyst needs to be able to isolate the information to be downgraded, ensure that the particular information cannot be modified until the downgrade operation has completed, and then release that information to the recipient on a timely basis.

3 Implications of the Bell and LaPadula Security Model

The Bell and LaPadula security model [17] imposes a number of constraints on possible implementations of MLS systems. In particular, Bell and LaPadula require

[5] Sanitization without leaving indicators is often very tricky, but for this paper, we assume that the analyst can easily determine which information is safe to downgrade.

that each process in a single system (or each system-high machine in a network) be identified at a particular security level. That process is allowed to read lower-classified information, but it is not allowed to write files that are marked at a lower classification level. This is to prevent Trojan horses from releasing arbitrary information. Note that this is a basic requirement of the model at evaluation levels EAL4 and above. It is not to be confused with covert channel issues [33, 36] that only come into play at B2 or EAL5 and above.

The result of this no-write-down requirement is that network connections between system-high systems are only generally useful if the systems are at precisely the same system-high level. Most network protocols require two-way communications (if only for packet acknowledgements), and acknowledgements cannot be permitted from high to low. This requirement is made clear in the Trusted Network Interpretation (TNI) [14] of the Orange Book [5].[6] It is possible to build truly one-way networks. Such networks were first proposed in chapter 7 of [25] and in [26]. Rushby and Randell [42] proposed a complete implementation of such a system, based on the Newcastle Connection, developed at University of Newcastle. There have been several commercial products evaluated in Australia[7] to implement one-way networks of one kind or another. These products from BAE Systems, Compucat, and Tenix Defence Systems all provide very limited communications capabilities.[8]

Why are these one-way networks so limited? Most network protocols use two-way communications to implement both flow control and error control. If you cannot have two-way communications, then there needs to be a trusted intermediary that accept and error check all messages sent by the sender, even if the receiver is refusing all input. This means that the intermediary may need huge amounts of buffer memory to hold hours or days worth of traffic. In addition, many protocols that run on top of TCP need two-way communications. For example, the FTP protocol [40] cannot run over a one-way network. The above-mentioned products use their own proprietary protocol to transfer files from low to high.

4 Hypervisor Implications

There are two classes of hypervisors that must be considered when examining the technical implications of MLS for hypervisors. The two classes are pure isolation hypervisors and sharing hypervisors.

4.1 Pure Isolation Hypervisors

A pure isolation hypervisor simply divides a machine into partitions, and permits no sharing of resources between the partitions (other than CPU time and primary memory). Implementing a pure isolation hypervisor is very easy, because the only security policy to be enforced is isolation. IBM's EAL5-evaluated PR/SM system [2] for the z/Series mainframes is a good example of a pure isolation hypervisor. There is essentially no sharing between partitions in PR/SM. PR/SM does have features for certain very limited forms of sharing (such as channel to channel connections, etc.), but under the EAL5 evaluation certificate, such sharing is absolutely forbidden. If a customer site turned on such sharing, they would no longer be running an evaluated configuration.

The partitions of a pure isolation hypervisor are essentially just like a collection of system-high separate computers. Each partition has its own disks and network connections, and if one partition is unclassified and the other is secret, then there cannot even be a network connection between them.

A valid question is, "Who would want a pure isolation hypervisor? You can get the same results by running several separate machines." In the case of a z/Series mainframe, there is a good reason. Mainframes are so expensive that the ability to partition one system into several isolated systems will save the customer lots of money, even if no sharing is permitted.

[6] The TNI [14] explicitly calls for strictly one-way networking at level B2 in section 3.2.1.3.4. However, in the B1 sections of the TNI, section 3.1.1.3.1 requires accurate labels on information transferred between network trusted computing base (NTCB) partitions, and section 3.1.1.4 requires that subjects and objects used for communication with other components are under control of the NTCS partition. The phrase "under control" is critical here, because the distinction between overt communications channels that must be secure at B1 and covert communications channels that need not be secure until B2 is whether or not they are "under control" of the TCB. Since the subjects and objects for communication are under control of the NTCB, the issues of one-way communications and packet acknowledgements are NOT covert channel issues. This is an inconsistency in the TNI and not an unexpected one. The TNI has been criticized in a number of ways for inconsistencies like this in [44].

[7] http://www.dsd.gov.au/infosec/evaluation_services/epl/dap.html

[8] The BAE Systems product evaluation report [3] indicates that it may have covert channel issues that are discussed in classified supplementary reports. The covert channel situation seems better on the other two products.

However, for smaller systems, such as small departmental servers or desktop or workstation clients, the benefits of a pure isolation hypervisor are much harder to justify. Additional separate systems are sufficiently cheap with modern technology that the inherent performance costs and complexity of hypervisors become significant issues.

One could argue for a pure isolation desktop client as an alternative to four or five clients that some DoD end-users have find room for in their offices. One such pure isolation desktop client is NetTop [39][9], developed by the NSA specifically to reduce the number of distinct client machines that an analyst must have on their desktop. NetTop allows separate partitions to connect to separate external networks, but allows no sharing of any kind between partitions. However, the end user of NetTop or other pure isolation hypervisors will almost immediately want to transfer information from one client to another, and they will get very frustrated when they can't.

Thus, pure isolation hypervisors are really only useful for customers of the largest and most expensive servers. Using multiple separate machines makes more sense for smaller configurations.

4.2 Sharing Hypervisors

Sharing hypervisors permit significant resource sharing between partitions. z/VM[10] is the best example of a sharing hypervisor. Virtual machines under VM can share either virtual or physical disk, network connections, etc. In the early days, two virtual machines under VM communicated by connecting the output of the virtual card punch of one into the input of the virtual card reader of another. Secured versions of sharing hypervisors can support a variety of secure applications, including fast, easy low to high sharing, sophisticated downgrading, etc. The idea of a secure sharing hypervisor originated with Madnick and Donovan [38]. The best examples of such secure sharing hypervisors are KVM/370 [45] and Digital's A1-secure VMM [31].

The most critical feature of a secure sharing hypervisor is a secure shared file store[11]. The secure shared file store allows a high level partition to have read-only access to low-level data, while a low-level partition gets read-write access to the same data. This avoids the clumsy one-way networking approaches described in section 3. Only a single copy of the data is required and updates are visible immediately to all partitions.[12]

The secure shared file store is so important, because it makes a variety of MLS applications possible. The most obvious is the secure read-down capability described in the previous paragraph. However, downgrading applications also need a secure shared file store. This is because the downgrading application needs to isolate the file to be downgraded, allow trusted programs and/or human beings to review what is to be downgraded, and only after all electronic or human approvals have been completed are the markings on the file changed. During that whole process that could take minutes or even hours, the candidate file must not be modified in any way by other than totally trusted software. Once approved, the remarking must be an atomic operation – either it totally completes or it doesn't happen at all. A secure shared file store makes this much easier to implement and to assure correctness.

5 Evolving from a Pure Isolation Hypervisor to a Sharing Hypervisor

An obvious question is what has to be done to evolve from a pure isolation hypervisor to a sharing hypervisor. The major constraints on this evolution

[9] The paper on NetTop [39] describes it as a high-assurance system. However, this description by the NSA is quite inaccurate. High assurance conventionally means evaluation at levels EAL6 or above. EAL4 and EAL5 can be considered medium assurance, although some would classify EAL4 as low assurance. Anything lower than EAL4 is certainly low assurance. NetTop is based on SE/Linux [37, 47] which has never been evaluated. Regular Linux kernels have been evaluated to EAL3 and are under evaluation at EAL4. Linux itself is so complex that it is likely to never reach high assurance without major reimplementation. SE/Linux, while it adds security features to Linux, also significantly increases the complexity of Linux [30] which makes it even less likely to ever achieve a high assurance evaluation. A quote from [44] well summarizes the need for genuine high assurance: "That which *must* be Trusted had best be Trustworthy."

[10] z/VM is the latest version of IBM's primary virtual machine monitor product. IBM invented the concept of hypervisors or virtual machine monitors in the mid 1960s at the Cambridge Scientific Center. The first prototype system was CP/40 [15, 35] on a specially modified System 360/40. The first version available outside of IBM was CP-67/CMS [21] for the System 360/67. The first fully supported product was VM/370 [21]. A full history of VM has been written by Varian [48].

[11] A secure shared file store is simply a secure file system available at the hypervisor level, rather than at the guest operating system level. A guest operating system might store its entire file system within a single file of secure shared file store. The mini-disks of z/VM are a good example of a secure shared file store.

[12] Properly synchronizing those updates, so that all partitions see consistent date requires a mechanism, such as version numbers [22] or event counts [41] to solve the secure readers-writers problem.

are that the add-ons must be secure, but they also must perform very well. It is very easy for a hypervisor implementation to add huge amounts of overhead to a system, and adding such overhead needlessly could easily result in customers abandoning either security or hypervisors or both.

5.1 One-Way Network Options

As discussed in section 3, it is possible to implement semi-usable one-way networks, if you have a highly trusted intermediary. The intermediary must have a very large buffer store that is totally protected. The easiest way to construct such a buffer is in a secure shared file store. The reason that a huge buffer is needed is because there cannot be any flow control from a high partition to a low partition. Even if the high partition cannot accept more packets, the intermediary must continue to accept traffic and store it until the high partition can again accept input. This is discussed in some detail in chapter 7 of [25]. You could give the intermediary access to an entire real disk drive as its buffer store, but based on the principle of least privilege, you would like a different intermediary for each pair of security levels. Managing those buffers in a secure file store will be much easier than with many separate real physical drives.

Another way to implement a one-way network is with one of the Australian-evaluated one-way network products that were mentioned in section 3. The product from Tenix Defence Systems deserves careful examination, because it even supports a one-way cut and paste capability (although only for Windows operating systems). However, these products all use hardware add-ons that are external to the hypervisor and would therefore require a separate physical network card for each partition.

5.2 Secure Shared File Store Options

A secure shared file store need not be a fully general-purpose file system. Most files in the file store will be virtual disks (or mini-disks as called in z/VM). A virtual disk contains an entire guest OS file system within it, so the virtual disk is likely to be large and is not likely to change its size very often. The secure shared file store can easily insist that such files always be contiguous and always be of fixed length. The other use for secure file store is to support downgrading operations. Files to be downgraded are likely to be much smaller than entire virtual disks, but once created, they never change at all, so again, fixed size contiguous files are acceptable. Since there may be many of these files and their size is likely to be small, using entire disk drives is not a viable option. They also have relatively short lifetimes while the data is being reviewed and then the data is likely to be moved into the virtual disks of the intended destination partition.

The secure shared file store could be implemented in two ways – as a subsystem of the hypervisor or as a separate highly trusted partition. Either approach can be made to be secure, but the overriding consideration must be performance. I/O operations are often the Achilles heel for performance in a hypervisor. Many workloads will be extremely I/O intensive and some will have downgrades occurring frequently. However, this is implemented, minimizing the cost of context switching into and out of the secure shared file store will be crucial.

If the secure file store is a subsystem of the hypervisor, the calls can be made as cross-ring calls that can often be implemented with little performance overhead and without having to flush caches and translation buffers. If the secure file store is in a separate partition, then the interface is essentially a message-passing interface. Lauer and Needham [34] have shown the duality of subroutine calls and message-passing calls, but in section 6.8 of [23], Karger argues that the calling interface will always (or at least almost always) will have better performance. This can be seen in the hypervisor context as follows. If the secure shared file system is a subsystem of the hypervisor itself, then a call to perform a read or a write consists of a cross-ring call into the kernel, followed by a cross-ring return. The calls and returns should require no flushing of translation buffers or caches. Between the call and the return, the file system must perform I/O and those operations may have context switches, depending on the driver architecture, of course.

By contrast, if the secure shared file system is implemented in a separate partition, then a call to perform a read or a write consists of a cross-ring call into the kernel to initiate a message pass, followed by a full context switch to the file system partition and a cross-ring return out to the file system code itself. The file system code now performs I/O, just as the file system subsystem would with essentially the same performance overheads for communicating to and from drivers. However, when the I/O has completed, the file system code must now do a cross-ring call into the kernel to initiate a message pass back to the caller, the kernel must do a context switch to the calling partition, followed by a cross-ring return to the actual calling code in the guest OS. The exact amount of overhead for the extra context switches to and from a different partition will depend on the precise implementation, of course, but even if the underlying CPU supports multiple address spaces in the TB simultaneously, the overhead will be significant, just from register saving, clearing, and restoring.[13] In a CPU without multiple address space support (such as the VAX or current generation x86 processors), the context switching overhead could easily be doubled or more. In an I/O intensive workload, this type of overhead could be prohibitive. Real decisions must be made on real performance benchmarks, of course. It is always dangerous to assume how a hypervisor will perform just from theoretical analyses. Section 7.3 of [20] shows how initial performance assists on the VAX VMM proved to be not very useful and how only measurement of those initial designs led to the proper optimizations.

Even if the secure shared file store is implemented as part of the hypervisor, this does not mean that the code needs to be dispersed all over the hypervisor, leading to increased complexity. As required at the higher levels of assurance of the Common Criteria, the hypervisor should be implemented as a layered architecture, with the file store in separate layers from the other parts of the hypervisor. An example of such a layered design for a secure file store can be seen in [31].

However the secure shared file store is implemented, the most important point of this paper is that the complexity and cost of a one-way network solution is likely to be comparable to that of a secure shared file store, yet the file store approach makes implementing sophisticated multi-level applications much simpler.

6 I/O Memory Management Units

A variety of processors are starting to deploy I/O memory management units (MMUs) that can significantly help performance by allowing a hypervisor partition to directly control unshared I/O devices. In most computers, I/O devices reference main memory using absolute addresses. This means that I/O drivers must be completely trusted, because they can address any location of main memory. An I/O MMU would provide the same concepts of virtual address translation and protection to I/O drivers that MMUs provide to the CPU. With an I/O MMU, a device driver would be constrained to only using those memory locations allocated to it by the operating system, just like an application program is similarly constrained. With such hardware support, most I/O drivers could become ordinary unprivileged programs. (The exception would be I/O drivers for shared multi-user devices, such as disks or networks. Such I/O drivers must provide secure multiplexing of those shared devices.)

I/O MMUs are not a new idea. The use of an I/O MMU was first proposed in 1975 for the Multics Secure Front-End Processor (SFEP) [18], and the first practical implementation was for the Honeywell SCOMP processor [19] which received the first-ever A1 security evaluation in 1985. An example of a modern implementation of I/O MMUs can be found in section 2.19 of [1].

7 Conclusion

This paper has shown what some of the critical issues are in adding multi-level security (MLS) to a hypervisor. If the users are content with a pure isolation hypervisor that makes sharing between partitions extremely difficult, then an approach such as NetTop is viable. A pure isolation hypervisor, such as PR/SM, can be extremely cost effective at sharing very expensive server hardware, such as IBM's z/Series mainframes. However, the paper has shown that many important applications cannot be easily implemented on a pure isolation hypervisor, and that implementing such applications creates the need for a secure file store, even at Common Criteria level EAL4. The paper contains theoretical analyses suggesting that implementing such a file store as a hypervisor subsystem will likely give much better performance, but tempers those recommendations with evidence that purely theoretical performance analysis of hypervisors can be misleading. Regardless of the implementation approach, the performance of such a secure file store will be extremely critical to the success of any hypervisor design.

[13] See [27] for details on the costs of register saving, clearing, and restoring in these cases, together with possible optimization techniques.

8 Acknowledgements

I want to especially thank Dave Safford for his suggestions and for pushing me to publish this paper. I would also like to acknowledge the useful comments on the paper provided by John Griffin, Ron Perez, J. R. Rao, Reiner Sailer, Leendert Van Doorn, and the anonymous referees.

9 References

1. *AMD64 Virtualization Codenamed "Pacifica" Technology: Secure Virtual Machine Architecture Reference Manual*, Publication No. 33047, Revision 3.01, May 2005, Advanced Micro Devices: Sunnyvale, CA. URL: http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/33047.pdf

2. *Certification Report for Processor Resource/ System Manager (PR/SM) for the IBM eServer zSeries 900*, BSI-DSZ-CC-0179-2003, 27 February 2003, Bundesamt für Sicherheit in der Informationstechnik: Bonn, Germany. URL: http://www.commoncriteriaportal.org/public/files/epfiles/0179a.pdf

3. *Certification Report: BAE SYSTEMS - Trusted Filter Version 1.0*, Certificate Number: 2001/19, July 2001, Defense Signals Directorate - Australasian Certification Authority: Kingston, ACT, Australia. URL: http://www.dsd.gov.au/infosec/evaluation_services/epl/network_security/BAESystems_TrustedFilter.html

4. *Computer Security Requirements -- Guidance for Applying the Department of Defense Trusted Computer System Evaluation Criteria in Specific Environments*, CSC-STD-003-85, 25 June 1985, DoD Computer Security Center: Ft. George G. Meade, MD. URL: http://www.radium.ncsc.mil/tpep/library/rainbow/index.html

5. *Department of Defense Trusted Computer System Evaluation Criteria*, DOD 5200.28-STD, December 1985: Washington, DC. URL: http://csrc.nist.gov/publications/history/dod85.pdf

6. *DoD Information Technology Security Certification and Accreditation Process (DITSCAP)*, DoD Instruction 5200.40, 30 December 1997, Department of Defense: Washington, DC. URL: www.dtic.mil/whs/directives/corres/pdf/i520040_123097/i520040p.pdf

7. *Information technology - Security techniques -- Evaluation criteria for IT security -- Part 1: Introduction and general model*, ISO/IEC 15408-1, 1999, International Organization for Standardization.

8. *Information technology - Security techniques -- Evaluation criteria for IT security -- Part 2: Security functional requirements*, ISO/IEC 15408-2, 1999, International Organization for Standardization.

9. *Information technology - Security techniques -- Evaluation criteria for IT security -- Part 3: Security assurance requirements*, ISO/IEC 15408-3, 1999, International Organization for Standardization.

10. *A Proposed Interpretation of the TCSEC for Virtual Machine Monitor Architectures*, 31 March 1989, Trusted Information Systems, Inc.: Glenwood, MD.

11. *Security Requirements for Automatic Data Processing (ADP) Systems*, DoD Directive 5200.28, 18 December 1972, Department of Defense: Washington, DC.

12. *Technical Rationale Behind CSC-STD-003-85: Computer Security Requirements -- Guidance for Applying the Department of Defense Trusted Computer System Evaluation Criteria in Specific Environments*, CSC-STD-004-85, 25 June 1985, DoD Computer Security Center: Ft. George G. Meade, MD. URL: http://www.radium.ncsc.mil/tpep/library/rainbow/index.html

13. *Techniques and Procedures for Implementing, Deactivating, Testing, and Evaluating Secure Resource-Sharing ADP Systems*, DoD 5200.28-M, January 1973, Department of Defense: Washington, DC.

14. *Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria*, NCSC-TG-005, Version-1, 31 July 1987, National Computer Security Center: Ft. George G. Meade, MD. URL: http://www.radium.ncsc.mil/tpep/library/tcsec/index.html

15. Adair, R.J., R.U. Bayles, L.W. Comeau, and R.J. Creasy, *A Virtual Machine System for the 360/40*, Report 320-2007, May 1966, IBM Cambridge Scientific Center: Cambridge, MA.

16. Barham, P., B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. *Xen and the Art of Virtualization*. in **Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles (SOSP)**. 19-22 October 2003, Bolton Landing, NY: ACM Press. URL: http://www.cl.cam.ac.uk/Research/SRG/netos/papers/2003-xensosp.pdf

17. Bell, D.E. and L.J. LaPadula, *Computer Security Model: Unified Exposition and Multics Interpretation*, ESD-TR-75-306, March 1976, The MITRE Corporation, Bedford, MA: HQ Electronic Systems Division, Hanscom AFB, MA. URL: http://csrc.nist.gov/publications/history/bell76.pdf

18. Biba, K.J., S.R. Ames, E.L. Burke, P.A. Karger, W.R. Price, R.R. Schell, and W.L. Schiller, *A Preliminary*

Specification of a Multics Security Kernel, WP-20119, April 1975, The MITRE Corporation: Bedford, MA.
19. Broadbridge, R. and J. Mekota, *Secure Communications Processor Specification*, ESD-TR-76-351, Vol. II, June 1976, Honeywell Information Systems, Inc., McLean, VA: HQ Electronic Systems Division, Hanscom AFB, MA.

20. Hall, J.S. and P.T. Robinson. *Virtualizing the VAX Architecture*. in **18th International Symposium on Computer Architecture**. May 1991, Toronto, ON, Canada: published in *Computer Architecture News*, vol. 19. p. 380-389.

21. Hendricks, E.C. and T.C. Hartmann, *Evolution of a Virtual Machine Subsystem.* **IBM Systems Journal**, 1979. **18**(1): p. 111-142. URL: http://domino.research.ibm.com/tchjr/journalindex.nsf/SysVolumes?OpenView

22. Hinke, T.H. and M. Schaefer, *Secure Data Management System*, RADC-TR-75-266 [NTIS AD A019201], November 1975, Rome Air Development Center: Griffiss AFB, NY.

23. Karger, P.A., *Improving Security and Performance for Capability Systems*, Computer Laboratory Technical Report No. 149, October 1988, University of Cambridge: Cambridge, England.

24. Karger, P.A., *Multi-Organizational Mandatory Access Controls for Commercial Applications*, RC 21673 (97655), 22 February 2000, IBM Research Division, Thomas J. Watson Research Center: Yorktown Heights, NY. URL: http://domino.watson.ibm.com/library/CyberDig.nsf/home

25. Karger, P.A., *Non-Discretionary Access Control for Decentralized Computing Systems*, S. M. & E. E. thesis 1977, Laboratory for Computer Science, Massachusetts Institute of Technology: Cambridge, MA. URL: http://ncstrl.mit.edu:80/Dienst/UI/2.0/Describe/ncstrl.mit_lcs%2fMIT%2fLCS%2fTR-179

26. Karger, P.A. *Non-Discretionary Security for Decentralized Computing Systems: Host to Host Protocols*. in **Trends and Applications: 1978 Distributed Processing**. 18 May 1978, National Bureau of Standards, Gaithersburg, MD: IEEE. p. 32-39.

27. Karger, P.A., *Using Registers to Optimize Cross-Domain Call Performance*, in *Proceedings of the Third International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)* 3-6 April 1989: Boston, MA. p. 194-204.

28. Karger, P.A., V.R. Austel, and D.C. Toll. *Using a Mandatory Secrecy and Integrity Policy on Smart Cards and Mobile Devices*. in **EUROSMART Security Conference**. 13-15 June 2000, Marseilles, France: p. 134-148.

29. Karger, P.A., V.R. Austel, and D.C. Toll. *Using Mandatory Secrecy and Integrity for Business to Business Applications on Mobile Devices*. in **Workshop on Innovations in Strong Access Control**. 25-27 September 2000, Naval Postgraduate School, Monterey, CA: published on CD-ROM. URL: http://www.acsac.org/sac-tac/wisac00/wed0830.karger.pdf

30. Karger, P.A. and R.R. Schell. *Thirty Years Later: Lessons from the Multics Security Evaluation*. in **Proceedings of the 18th Annual Computer Security Applications Conference**. 9-13 December 2002, Las Vegas, NV: IEEE Computer Society. p. 119-126. URL: http://www.acsac.org/2002/papers/classic-multics.pdf

31. Karger, P.A., M.E. Zurko, D.W. Bonin, A.H. Mason, and C.E. Kahn, *A Retrospective on the VAX VMM Security Kernel.* **IEEE Transactions on Software Engineering**, November 1991. **17**(11): p. 1147-1165.

32. Kerner, S.M., *IBM Offers Support for Xen.* **internetnews.com**, 19 January 2005. URL: http://www.internetnews.com/dev-news/article.php/3461481

33. Lampson, B.W., *A note on the confinement problem.* **Communications of the ACM**, October 1973. **16**(10): p. 613-615.

34. Lauer, H.C. and R.M. Needham, *On the duality of operating system structures*, in *Operating Systems: Theory and Practice*, D. Lanciaux, Editor. 1979, North-Holland: Amsterdam. p. 371-384.

35. Lindquist, A.B., R.R. Seeber, and L.W. Comeau, *A Time-Sharing System Using an Associative Memory.* **Proceedings of the IEEE**, December 1966. **54**(12): p. 1774-1779.

36. Lipner, S.B., *A comment on the confinement problem.* **Operating Systems Review**, 19-21 November 1975. **9**(5): p. 192-196. Proceedings of the Fifth Symposium on Operating Systems Principles.

37. Loscocco, P. and S. Smalley. *Integrating Flexible Support for Security Policies into the Linux Operating System*. in **Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference (FREENIX '01)**. 2001, Boston, MA. URL: http://www.nsa.gov/selinux/doc/freenix01.pdf

38. Madnick, S.E. and J.J. Donovan. *Application and Analysis of the Virtual Machine Approach to Information System Security*. in **Proceedings of the ACM SIGARCH-SIGOPS Workshop on Virtual Computer Systems**. 26-27 March 1973, Cambridge, MA: Association for Computing Machinery. p. 210-224. URL: http://portal.acm.org/citation.cfm?id=803961

39. Meushaw, R. and D. Simard, *NetTop: Commercial Technology in High Assurance Applications.* **National Security Agency Tech Trend Notes**, Fall 2000. **9**(4): p. 3-10. URL: http://www.vmware.com/pdf/TechTrendNotes.pdf

40. Postel, J. and J. Reynolds, *File Transfer Protocol (FTP)*, RFC 959, October 1985, Network Working Group. URL: http://www.ietf.org/rfc/rfc959.txt

41. Reed, D.P. and R.K. Kanodia, *Synchronization with Eventcounts and Sequencers.* **Comm. ACM**, February 1979. **22**(2): p. 115-123.

42. Rushby, J. and B. Randell, *A Distributed Secure System.* **IEEE Computer**, July 1983. **16**(7): p. 55-67. URL: http://www.csl.sri.com/users/rushby/abstracts/computer83

43. Sailer, R., T. Jaeger, J.L. Griffin, S. Berger, L. van Doorn, R. Perez, and E. Valdez, *Building a General-purpose Secure Virtual Machine Monitor*, RC23537 (W0502-132), 25 February 2005, IBM Research Division, Thomas J. Watson Research Center: Yorktown Heights, NY. URL: http://domino.watson.ibm.com/library/CyberDig.nsf/home

44. Schaefer, M., W.C. Barker, and C.P. Pfleeger. *Tea and I: An Allergy*. in **IEEE Symposium on Security and Privacy**. 1-3 May 1989, Oakland, CA: IEEE Computer Society. p. 178-182.

45. Schaefer, M., B. Gold, R. Linde, and J. Scheid. *Program Confinement in KVM/370*. in **Proceedings of the 1977 ACM Annual Conference**. 16-19 October 1977, Seattle, WA: p. 404-410.

46. Scherzer, H., R. Canetti, P.A. Karger, H. Krawczyk, T. Rabin, and D.C. Toll. *Authenticating Mandatory Access Controls and Preserving Privacy for a High-Assurance Smart Card*. in **8th European Symposium on Research in Computer Security (ESORICS 2003)**. 13-15 October 2003, Gjøvik, Norway:Lecture Notes in Computer Science Vol. 2808. Springer Verlag. p. 181-200.

47. Smalley, S., *Configuring the SELinux Policy*, NAI Labs Report #02-007, June 2002, NAI Labs: Glenwood, MD. URL: http://www.nsa.gov/selinux/policy2-abs.html

48. Varian, M. *VM and the VM Community: Past Present, and Future*. in **SHARE 89, Sessions 9059-9061**. August 1997. URL: http://pucc.princeton.edu/~melinda/25paper.pdf