Towards Multi–Layer Trusted Virtual Domains

Yasuharu Katsuno *Michiharu Kudo*
Yuji Watanabe *Sachiko Yoshihama*

{katsuno, kudo, muew, sachikoy}@jp.ibm.com
IBM Tokyo Research Laboratory
1623-14, Shimotsuruma, Yamato-shi, 242-8502 Japan

Ronald Perez *Reiner Sailer*
Leendert van Doorn

{ronpz, sailer, leendert}@us.ibm.com
IBM T. J. Watson Research Center
Hawthorne, NY 10532 USA

Abstract

We address the fine-grained control of information flow between distributed applications to protect transactions. Confining transactions in distributed untrusted environments is crucial for Web services, distributed computing, and most e-commerce applications relying on those technologies. Applications must be protected from rogue peer applications but at the same time applications need to communicate to selected peer applications to implement transactions and distributed services.

We propose a solution based on Trusted Virtual Domains (TVD), which represent a new model for achieving IT and business security. TVDs are designed to satisfy business-level goals by simplifying management and providing explicit infrastructure-level containment and trust guarantees. TVDs can be applied to different layers within a system. We consider here TVDs for the virtual machine monitor (VMM) as well as at the application level. Combining protection in those layers is especially useful for systems that intersect multiple virtual machines and need to separate information flow at the applications level.

In this contribution, we briefly describe the design and the prototype implementations of TVDs at the application and VMM levels. We focus on implications of the layering of these two TVDs to achieve consistent trust and confinement properties across distributed applications.

1 Introduction

Computers handle increasing amounts of sensitive and non-sensitive information. As a result, information flows into and out of computers are increasing rapidly. The leaking of information from computer systems or the corruption of computer systems due to malicious incoming data are thus much harder to detect, and this has become a serious problem. This affects not only information leakage due to malicious software running on a computer, but also leakage through careless users ignoring security policies or through configuration errors. These problems persist because no appropriate information flow control mechanism exists. Achieving strict information flow control while maintaining the usability and performance that users require is challenging. The existing situation in which various kinds of policies co-exist to control applications in distributed heterogeneous environments intensifies this problem.

Earlier frameworks for enforcing information flow focused on building confined execution environments and establishing distributed coalitions. Examples are NetTop [1], Terra [2], PERSEUS [3], EMSCB [4], and Shamon [5], which are based on recent advances in distributed computing and security technologies. These solutions establish trust within a group of computing components in distributed environments and manage enforcement of different security policies for the components of each group.

We are proposing Trusted Virtual Domains (TVD [6, 7]) as a new trust-based model for distributed coalitions. A TVD establishes multiple secure coalitions called domains for components on nodes in distributed heterogeneous environments. It supports distributed mandatory access control and also supports security policies that can vary from domain to domain.

In this paper, we propose a multi-layered TVD that combines a workload level TVD and an application level TVD. The multi-layered TVD offers both strong confinement and fine-grained application information flow control. It comprises a virtualization-level TVD component to control the sharing between distributed workloads and an application-level TVD component to protect fine-grained information flow between applications. Considering multiple layers and their specific subject and object abstractions enables us to apply effective protection. We control those objects that are naturally managed in these layers and we control access to those objects non-intrusively by intercepting a few natural object access methods that exist in the layer. We argue that this greatly improves the simplicity and usability of the policies governing those access control decisions.

This paper is structured as follows: Section 2 gives some background information about TVDs. Section 3 introduces a motivating scenario. Section 4 describes two examples of a workload level TVD and an application level TVD. We discuss and evaluate the multi-layered TVD as regards its confinement, complexity and scalability, and assurance in Section 5. This paper concludes with our ongoing work in Section 6.

2 Trusted Virtual Domain Background

Trusted Virtual Domains (TVDs [6, 7]) represent a new model for achieving IT and business security. A TVD is a secure coalition of entities that can trust each other. Within a coalition, members agree upon security policies that are uniformly enforced independently of the boundaries of physical computing resources.

The TVD model is designed to be independently and concurrently implementable using a variety of underlying technologies and mechanisms but includes several key functionalities: 1) isolated execution environment, 2) enforcement of mandatory access control policies, 3) establishment of trusted coalitions based on platform integrity measurements, and 4) secure communication channels that bridge entities in the coalition.

3 Motivating Scenario

As an example we assume that a fictitious consulting company yourconsultant.com has multiple customers. The information exchange between yourconsultant.com and its customers is based on documents, such as text files, spreadsheets, and presentation slides. We assume in this example, that one of the customers (referred to as customer A) has a simple two-level document classification policy; any document is either confidential (conf.) or unclassified (u.c.).

The goals of yourconsultant.com are twofold. First, yourconsultant.com must ensure that the documents of each of its customers remain isolated from those of other customers. Second, yourconsultant.com must guarantee that it consistently enforces the policy of its customers with regard to their documents and its own policy for its own documents and workloads.

The information flow policy of yourconsultant.com with regard to customer A is that 1) no confidential information should flow into unclassified documents, and 2) no confidential document should be disclosed to external parties.

The policy of yourconsultant.com with regard to its own operations is more elaborate and depends on both the application itself and the application data. We assume that yourconsultant.com has a Web Services based system for internal workflows, such as Human Resources, Payroll, and Travel Reimbursement for employees. For example, when an employee submits a travel reimbursement request, the request includes the employee ID, the date and destination of the trip, itemized expenses, and the employee's bank account number for receiving the reimbursement. This information is visible only to those people who need to know it. E.g., when the manager of an employee approves a reimbursement request, she cannot see the bank account number since it is regarded non-essential to deciding the approval. In addition, the manager is not allowed to make local copies of the reimbursement request. This way, sensitive information are prevented from being stored persistently on computers that are subject to offline attacks (e.g., theft or loss of laptops).

4 Trusted Virtual Domain Case Studies

In the following subsections, we review two examples of TVDs that focus on the different layers – the VM level and the application level.

4.1 Trusted Virtual Data Center

The Trusted Virtual Data Center (TVDc [8]) represents a realization of Trusted Virtual Domains offering strong enterprise-level security guarantees in hosted data center environments. The Trusted Virtual Data Center is designed to satisfy business-level security goals by simplifying management and providing explicit infrastructure-level containment and trust guarantees for data center environments based on virtualization.

The Trusted Virtual Data Center isolates multiple customer sets, e.g., Human Resources and Financing. Related processes and their resources (workloads) can be distributed across multiple VMs and VMMs. Consequently, this is not limited to isolation capabilities in the platform (secure hypervisors), but also includes isolation of the network (VLANs, labeled IPSec tunnels), routers, management consoles, etc. The fundamental isolation mechanisms are largely in place; to add the ability to coherently manage and attest to these capabilities is the major contribution of TVDc. TVDc also offers interfaces through which additional controls can be implemented seamlessly on top of TVDc in order to enforce fine-grained information-flow control between applications.

Furthermore, these isolation guarantees must be translated into the strong containment guarantees that businesses expect. For example, a virus outbreak in one workload should not spill over to another workload even if both workloads use the same physical resources (servers, networks, routers, etc). The management of these Trusted Virtual Data Centers must be integrated into VMM management applications to ensure that customers and administrators are using simple interfaces to make informed decisions and that formal security policies are enforced in management and operation, which can be translated into customer guarantees regarding trust and confinement.

Implementation of the virtualization layer TVD (TVD_{vm}). The TVDc is the most promising embodiment of a virtualization layer TVD. Providing workload confinement, it relies on the virtualization layer to enforce core isolation between virtual machines and resources. We use sHype [9] (Secure HYPErvisor) to provide coarse-grained but very robust confinement guarantees for distributed workloads running inside of virtual machines. To securely connect virtual machines running on different platforms, we use labeled IPSec tunnels. We require that the VMs connected by such tunnels have the same workload type [5].

The TVDc trusted computing base consists of the policies and their enforcement inside the virtual machine management and the hypervisor. To establish trust into the consistent security policy enforcement across multiple hypervisor platforms, we use remote software attestation based on the Trusted Platform Module [10] or a virtual Trusted Platform Module [11]. We equip trusted virtual machines,such as labeled IPSec gateways that connect peer VMs over untrusted networks, with the Integrity Measurement Architecture (IMA [12]). IMA logs provable evidence about all programs and policies loaded into the VM run-time. This evidence is offered to user Virtual Machines to provide integrity guarantees and to establish trust into their VM policy and enforcement.

4.2 Application Level Information Flow Control

Distributed applications usually require finer-grained confinement than that offered by workload-level management. Financing applications, for example, might need to manage documents with regard to each subject, or each project. Likewise, Human Resources applications might allow managers to access their employee's performance rating; such ratings must not be accessible to regular employees. The situation becomes more challenging if we consider the situation that employees can check only their own attendance record, or that managers can only check ratings of their staff.

Implementing finer-grained information flow control requires not only isolation on virtual machines but also on the application level.

Operating system security such as SELinux [13] or sandbox mechanisms such as the JVM can provide isolation among applications running in the same execution environment. A runtime process monitor [14] observing running application processes can provide an alternative way to achieve the same level of security. However, the security of the monitoring mechanism depends on the protection mechanism, such as privilege control and confinement, provided in the lower layer.

The controlled sharing between confined environments established in the application layer is the key issue when considering practical scenarios. For example, it is desirable that information can be disclosed if the output does not contain any confidential data [15].

Information Flow Control for Java (IFC-J [16]) provides information-flow control mechanism for Java programs by using the technique of an Inline Reference Monitor (IRM). The Secure Message Router (SMR [17]) provides trusted mediation for interconnected TVDs supporting heterogeneous and decentralized environments. WS-Attestation [18] is a WS-Security based attestation protocol which allows Web services to bind mutual platform trust information into the communication contexts of SOAP messages.

Application-level information flow control provides a useful solution to achieve fine-grained access control. However, to be effective it relies on secure run-time environments. Such secure run-time environments can be established through virtualization security mechanisms, such as hypervisor reference monitors. Thus, this implies the need to develop a layering technique which connects multiple layered security components without spoiling the generality and usability.

Implementation of the application-layer TVD (TVD_{app}). An OS-level reference monitor like SELinux fills the gap between TVD_{vm} and upper layers by providing isolation and policy enforcement capability. IMA enables measurement of precise OS configurations and applications, and thus produces richer platform integrity information that the upper layer can utilize.

The runtime environment middleware, such as J2EE or OSGi, provides a weakly isolated environment for applications. However, the security model of Java is designed for restricting access to OS resources rather than controlling application level information flow within the Java code. IFC-J enables stronger information flow control on top of a JVM,

Message level protocols, such as SOAP, allow for fine-granular communication channels that are capable of exchanging security related meta-information (e.g., security labels) for each data element. In addition, WS-Security may be employed to protect the integrity and confidentiality of each data element.

These constituents can be cascaded on one another, until the system obtains the desired level of granularity.

5 Multi-Layer TVD

As we have seen, the concept of TVD can be implemented on different layers and each approach has pros and cons.

Advantages of the workload level TVDs (i.e., TVD_{vm} is that they can provide strong workload level isolation and access control between isolated workloads. However, a drawback is that the access control provided by the workload level TVDs is at the level of virtual hardware and resources, and thus it is not an efficient mechanism to enforce information flow-control policies on files, data, or messages.

On the other hand, the application level TVDs (i.e., TVD_{app}) enable finer grained information flow control on data, but assurance heavily depends on the underlying system. That is, for an application level TVD to be effective, the underlying layer must (i) provide basic protection such as isolation from malicious executions and (ii) attest to the integrity of the policy enforcement function.

The multi-layer TVD we propose in this paper mitigates these drawbacks by combining the application-level TVD (TVD_{app}) on top of the workload level TVD (TVD_{vm}) as illustrated in Figure 1. The multi-layer TVD will enable fine granularity and confinement at the same time:

Fine-grained information flow control on the application level. The application level TVD enables controlled information flow not only within a single platform but across distributed platforms. Therefore, it can enforce application-level security policies; e.g., protects each element in transaction messages.

Coarse-grained and robust workload confinement in the virtualization layer. The workload level TVDs provide strong workload isolation at the granularity of the virtual hardware resources. Therefore, even if the application level TVD is attacked, the damage is confined within the workload, and will not contaminate other workloads which may run on the same physical platform.

The lower virtualization layer is strictly protected through hardware protection mechanisms from the higher application layer. It can therefore contain an exposure and provide a safety net in case security in the more complex application layer fails [19] and implement basic confinement properties when an appropriate operating system security is not available.

5.1 Use Cases

In this section, we briefly review several use cases to exemplify the value of the multi-layer TVD.

Enterprise Application Infrastructure In this scenario, we consider the special case of deploying a single TVD_{vm}, which controls the sharing of enterprise application service infrastructure. Trusted enterprise applications run in their own VMs and those VMs are participating in a single TVD_{vm}; only VMs with trusted applications can share information among each other. Less trusted applications (such as web surfing and video streaming) run in other VMs that remain outside the TVD_{vm}; those cannot share information with the trusted applications running in TVD_{vm}. To refine information flow control among trusted applications, we deploy TVD_{app} across the VMs of TVD_{vm}. For example, TVD_{app} can restrict the disclosure of HR information to trusted applications depending on their need to know. To enable these fine-grained controls, it keeps track of document contents by monitoring copy-and-paste, read, and write operations using OS level reference monitors inside the VMs.

Customer Data Management Imagine a consulting company which serves multiple customers who have conflicting interests. The strong isolation provided by TVD_{vm} will isolate workloads of the different customers. In addition, TVD_{app} provides information flow control according to the customers' policies. Note that the customers' policies may vary (e.g., Biba [20], Bell-LaPadula [15], or flexible policy based control) and TVD_{app} can implement flexible policies to accommodate fine-grained customer requirements.

Service Oriented Computing As Service-Oriented Computing becomes popular, there are increasing needs to protect data not only in message exchanges between services but also after the data is consumed by the backend servers. The multi-layer TVD can protect the Service Oriented Computing environment from two aspects; 1) TVD_{vm} protects the enterprise service bus by allowing only equally trustworthy service workloads to connect to the same bus, and 2) TVD_{app} enforces application level security policies among the services. Extended Web Services messaging will communicate security labels of the exchanged data, and thus the policy can be enforced consistently across all of the relevant services.

5.2 Trade-offs of TVD characteristics

The multi-layer TVD can be structured in various ways to support different scenarios. This section reviews some trade-offs of Trusted Virtual Domains.

Fine-granularity vs. Confinement Strength Confinement implemented in lower layers requires less code to be trusted and results in robust but coarse-grained properties. Confinement at higher layers requires more code to be trusted and results in less strong but more fine-grained properties. Hence, if we exclusively use TVD_{vm} to divide a domain into workloads, then the confinement of these workloads becomes very strong. However, the flexibility with which we can enforce information flow between those workloads is coarse-grained because it is based on VMs and virtual resources. Conversely, if there are fewer segmentations by TVD_{vm} and more information flow is controlled by TVD_{app}, then policy enforcement becomes more flexible but the confinement becomes less strong. As a result, achieving both strong and fine-grained granularity requires cooperating TVD layers.

Policy Complexity and Scalability We look at the security policy from a perspective of (i) security officers being able to easily define and understand the policy, (ii) the policy being expressive enough to enforce the required confinement effectively, and (iii) the enforcement code remaining non-intrusive to the common usage and maintenance of the system environment.

TVD_{vm} is best suited to isolate workloads in enterprise hosting environments or workspaces on clients. Related constraints can be very effectively described and enforced by controlling the objects of the virtualization layer (VMs and resources). Similarly, fine-grained application-level objects are most effectively controlled by TVD_{app} access control since it operates on the objects that are defined and dealt with in the application layer. In general, security policies are simple if they concern a limited number of subjects and objects. Layering and confinement enables the use of simple policies by dealing with few coarse-grained subjects and objects on the virtualization layer. It deals with fine-grained subjects and objects (transactions and data items) within very confined environments that restrict the number of access relationships to the minimum (e.g., a single application or JVM).

However, using TVD_{vm} to protect finer-grained objects, such as selected application data (e.g., credit card numbers) or document content tags would result in (i) a policy that is hard to understand for workload managing administrators. Further, the fine-grained application-level objects (ii) cannot easily be expressed with the coarse-grained security policy and would likely require (iii) a restructuring of the information infrastructure that is very intrusive to the normal system operation. Similarly, using TVD_{app} to achieve coarse grained workload confinement yields very complex policies because many processes and data items must be controlled to indirectly enforce very coarse-grained confinement requirements. The resulting policy will be (i) over-expressive and therefore very complex and (ii) hard to understand, and due to policy decisions at many locations, the enforcement code would become (iii) very intrusive.

Level of System Assurance The enforcement granularity increases from TVD_{vm} towards TVD_{app}, while the assurance of the security strength of the system decreases, due to the increased size and complexity of the Trusted Computing Base (TCB). In addition, increasing exposure of administrative interfaces resulting from fine-grained TVD_{app} enforcement can increase vulnerabilities of such interfaces. Finally, the TVD life-cycle management, which takes application level policies into account, can become complicated and thus become a point-of-failure if not managed carefully.

5.3 An Example of a Multi-Layer TVD

To implement the example scenario in Section 3, we deploy layered TVDs as depicted in Figure 1.

The first goal, i.e., isolating the workload for each customer and yourconsultant.com, is achieved by deploying Trusted Vir-

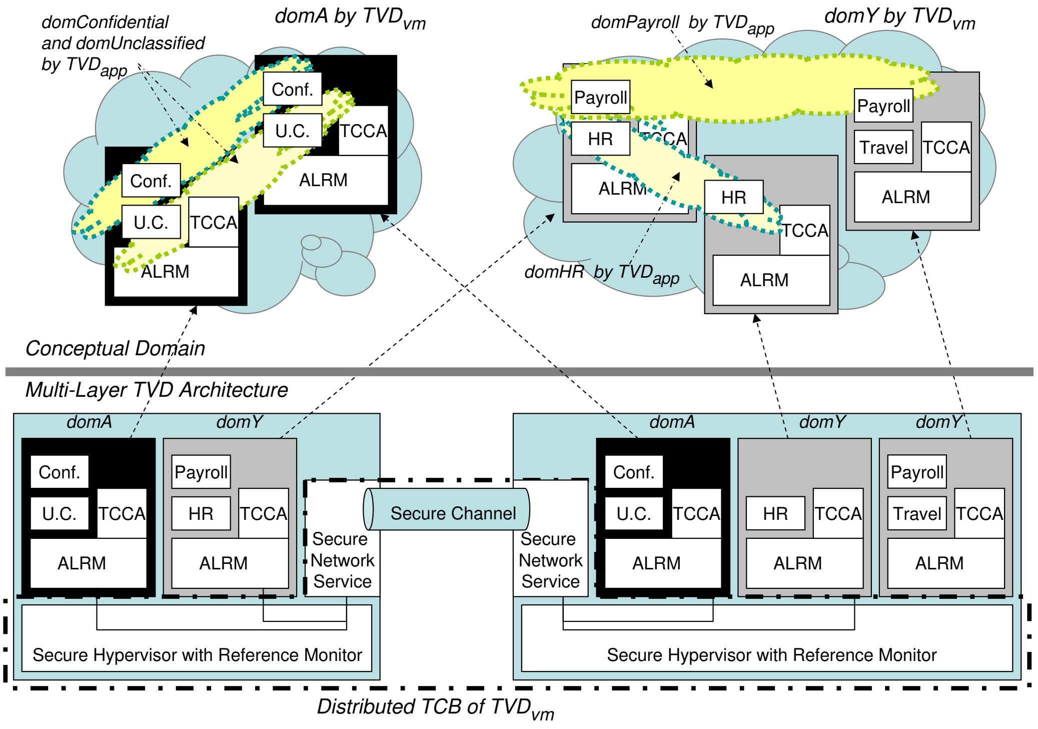
domConfidential and domUnclassified by TVD_{app}
domA by TVD_{vm}
Conf.
U.C.
TCCA
ALRM
domPayroll by TVD_{app}
domY by TVD_{vm}
Payroll
HR
Travel
domHR by TVD_{app}
Conceptual Domain
Multi-Layer TVD Architecture
domA
domY
Secure Network Service
Secure Channel
Secure Hypervisor with Reference Monitor
Distributed TCB of TVD_{vm}

Figure 1: Multi-Layer TVD Architecture at Yourconsultant.com

tual Domains on VM level (i.e., TVD_{vm}). Those TVDs isolate VM workloads into separate domains for each customer and for yourconsultant.com: $domA$ for customer A (etc.) and $domY$ for internal workloads of yourconsultant.com (Human Resources, Payroll, Travel Reimbursement). The robust confinement offered by TVD_{vm} ensures that, by default, information flow among VMs of the same domain is not restricted but information flow between different domains is prevented.

The second goal, i.e., to guarantee that yourconsultant.com consistently enforces the policy of itself and its customers with regard to their documents and respective policy, is met by deploying Trusted Virtual domains on application level in $domA$ and $domY$.

A customized Application Level Reference Monitor (ALRM) in $domA$ is implemented to monitor copying and moving of each file, so that when a copy of a file is made, the label of the original file is applied to the new file. In addition, the reference monitor checks file operations (i.e., open, save, and close) as well as clipboard operations (i.e,. copy, cut, and paste). Whenever information is copied from a confidential to an unclassified document, the unclassified document will be tainted with the "Confidential" security label.

The server and the client for each application in $domY$, such as Human Resources, Payroll, and the Travel Reimbursement system, are implemented as an application level TVD. They communicate with each other using Web Services protocols and include extended security meta-information in each message. This meta-information is used to associate a security label with a each piece of information (e.g., bank account number) in the workflows. ALRM inspects those messages and enforces the fine-grained policy.

Establishing trust into the application level policy enforcement is achieved in two steps: First, each hypervisor platform involved in a TVD_{vm} establishes mutual trust into its peers using TPM-based remote integrity attestation to determine the integrity of the peer hypervisor and its reference monitor, the peer hypervisor security policy, and the basic security services for workload level isolation. After that, each VM involved in a TVD_{app} establishes mutual trust into peer ALRM, making sure that equivalent ALRMs are running on each peer and are enforcing a consistent policy. To protect data while it travels between the trusted hypervisors, secure network services in TVD_{vm} deploy secure labeled tunnels. Similarly, the Trusted Component Control Agent (TCCA) mediates and protects communication between applications in TVD_{app}.

In summary, the workload level TVDs (TVD_{vm}) such as $domA$ and $domY$ are most effective in providing strong isolation and confinement on the VM level. Application level TVDs (TVD_{app}) are most effective in satisfying fine-grained policies on application level using a distributed application-level reference monitor that spans across multiple virtual machines of the same TVD_{vm}. Both together effectively deliver the robustness and fine-grained application controls that today's enterprise services demand.

6 Outlook and Conclusion

Trusted Virtual Domains ([6], [7]) are a promising technology to control information flow in distributed environments with enterprise-level assurance. Our contribution in this paper is to extend the TVD architecture to address two orthogonal goals: coarse-grained workload confinement and fine-grained information flow control.

Our examination shows that no single TVD layer alone can offer the optimal security with regard to robustness, granularity, usability, and assurance. Only a layered approach to Trusted Virtual Domains can yield a scalable trade-off across many application environments. Such an approach, striking the right balance between the TVD_{app} and TVD_{vm}, supported by an operating system that connects them, will yield not only the best robustness trade-off with regard to confinement but also the best usability.

There are several opportunities for future work to make a multi-layered TVD approach suitable for enterprise environments: (i) the TVD layers need a coherent model to manage their security policies and to synchronize changes in those policies across the layers. (ii) Responsibilities must be defined for each layer in function of the layer's natural abstractions within the system (e.g., workload-based confinement properties by the TVD_{vm} vs. application data tracking by the TVD_{app}). (iii) Layered remote attestation must enable peer layers to establish trust into a common distributed TVD layer, and (iv) different layers within a system need layer interfaces so that higher layers can benefit from and synchronize with lower layers.

Finally, tools that improve practical usability aspects of a multi-layered TVD environment will be necessary. Tools that decompose a high-level enterprise security policy into policies for the individual TVD layers are beneficial when setting up and maintaining a multi-layer environment. Tools that verify that the composition of individual TVD layer policies implements the intended enterprise security policy will be very valuable during evaluation and testing.

References

[1] R. Meushaw and D. Simard. Nettop: Commercial technology in high assurance applications. In *Tech Trend Notes Volume 9 Edition 4, National Security Agency*, 2000.

[2] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: A virtual machine-based platform for trusted computing. In *Proceedings of the 19th Symposium on Operating System Principles (SOSP 2003)*, 2003.

[3] Ahmad-Reza Sadeghi and Christian Stüble. Taming"trusted computing" by operating system design. In *the 4th International Workshop on Information Security Applications (WISA'03)*, 2003.

[4] Ahmad-Reza Sadeghi, Christian Stüble, and Norbert Pohlmann. European multilateral secure computing base - open trusted computing for you and me. In *Datenschutz und Datensicherheit (DUD) 9/2004*, 2004.

[5] J. M. McCune, S. Berger, R. Cáceres, T. Jaeger, and R. Sailer. Shamon – A system for distributed mandatory access control. In *To appear in: Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, December 2006.

[6] A. Bussani et al. Trusted virtual domains: Secure foundations for business and it services. In *IBM Research Report RC23792*, 2005.

[7] John L. Griffin, Trent Jaeger, Ronald Perez, Reiner Sailer, Leendert van Doorn, and Ramón Cáceres. Trusted virtual domains: Toward secure distributed services. In *Proceedings of the Workshop on Hot Topics in System Dependability*, 2005.

[8] IBM T. J. Watson Research Center. Trusted Virtual Data Center. http://www.research.ibm.com/ ssd_trustedvirtualdatacenter, 2006.

[9] R. Sailer, T. Jaeger, E. Valdez, R. Cáceres, R. Perez, S. Berger, J. Griffin, and L. van Doorn. Building a MAC-based security architecture for the Xen opensource hypervisor. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, December 2005.

[10] Trusted Computing Group. Trusted Platform Module. https://www.trustedcomputinggroup.org/ groups/tpm, 2006.

[11] S. Berger, R. Cáceres, K. Goldman, Ronald Perez, R. Sailer, and L. van Doorn. vTPM: Virtualizing the Trusted Platform Module. In *Proceedings of the USENIX Security Symposium*, July 2006.

[12] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn. Design and implementation of a TCG-based integrity measurement architecture. In *Proceedings of the USENIX Security Symposium*, 2004.

[13] Peter Loscocco and Stephen Smalley. Integrating flexible support for security policies into the linux operating system. In *Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference*, 2001.

[14] S. Furuichi, T. Mishina, and Y. Otani. The design and implementation of real-time program monitoring and controlling mechanism on pc for information security system. In *Proceedings of Symposium on Cryptography and Information Security 2006 (SCIS2006)*, 2006.

[15] D. E. Bell and L. J. LaPadula. Secure computer system: Unified exposition and multics interpretation. Technical report, MITRE MTR-2997, March 1976.

[16] S. Yoshihama et al. Dynamic information flow control for java by inline reference monitor. In *Compute Security Symposium (CSS 2006), Kyoto, Japan*, 2006.

[17] Y. Watanabe et al. Bridging the gap between intercommunication boundary and internal trusted components. In *European Symposium On Research In Computer Security (ESORICS)*, September 2006.

[18] S. Yoshihama et al. WS-attestation: Efficient and fine-grained remote attestation on web services. In *Proceedings of the 2005 IEEE International Conference on Web Services (ICWS 2005)*, July 2005.

[19] S. E. Madnick and J. J. Donovan. Application and analysis of the virtual machine approach to information system security and isolation. *Proceedings of the ACM workshop on virtual computer systems*, 1973.

[20] K. J. Biba. Integrity Considerations for Secure Computer Systems. Technical Report MTR-3153, Mitre Corporation, Mitre Corp, Bedford MA, June 1975.