PROCEEDINGS

# 55. IWK

Internationales Wissenschaftliches Kolloquium
International Scientific Colloquium

13 - 17 September 2010

# Crossing Borders within the ABC

## Automation,

## Biomedical Engineering and

## Computer Science

Faculty of
Computer Science and Automation

www.tu-ilmenau.de

TECHNISCHE UNIVERSITÄT
ILMENAU

**Impressum**
**Published by**

| | |
|---|---|
| Publisher: | Rector of the Ilmenau University of Technology<br>Univ.-Prof. Dr. rer. nat. habil. Dr. h. c. Prof. h. c. Peter Scharff |
| Editor: | Marketing Department (Phone: +49 3677 69-2520)<br>Andrea Schneider (conferences@tu-ilmenau.de)<br><br>Faculty of Computer Science and Automation<br>(Phone: +49 3677 69-2860)<br>Univ.-Prof. Dr.-Ing. habil. Jens Haueisen |
| Editorial Deadline: | 20. August 2010 |
| Implementation: | Ilmenau University of Technology<br>Felix Böckelmann<br>Philipp Schmidt |

**USB-Flash-Version.**

| | |
|---|---|
| Publishing House: | Verlag ISLE, Betriebsstätte des ISLE e.V.<br>Werner-von-Siemens-Str. 16<br>98693 Ilmenau |
| Production: | CDA Datenträger Albrechts GmbH, 98529 Suhl/Albrechts |
| Order trough: | Marketing Department (+49 3677 69-2520)<br>Andrea Schneider (conferences@tu-ilmenau.de) |

ISBN: 978-3-938843-53-6 (USB-Flash Version)

**Online-Version:**

| | |
|---|---|
| Publisher: | Universitätsbibliothek Ilmenau<br>ilmedia<br>Postfach 10 05 65<br>98684 Ilmenau |



**Home / Index:**
http://www.db-thueringen.de/servlets/DocumentServlet?id=16739

# CAUSAL TRUSTED COMPUTING BASES

*Anja Fischer*

a.fischer@tu-ilmenau.de
Ilmenau University of Technology
Ilmenau, Germany

## ABSTRACT

Trusted Computing Bases (TCBs) of today's commodity systems have arbitrarily evolved for decades. As a result, these TCBs are heterogeneous, large, distributed, and complex, and the resulting security architectures inherit these properties. Non-surprisingly, implementations are also large and complex, such that correctness, robustness, and tamperproofness of a TCB's implementation are quite hard to verify.

In this proposal we present an approach that exploits causal dependencies between security policies and TCB functions, in order to precisely define a TCB's functional range. The objective is to set the course for a TCB's implementation whose size allows for verifying their correctness and tamperproofness.

***Index Terms*—** Trusted Computing Bases, Security Architectures, Security Policies, Security Models

## 1. INTRODUCTION

Security properties are not properties of privileged systems running in very security- and safety-critical areas any more; instead almost any IT system features security properties nowadays, and the correctness of a system's security properties is none the less crucial for the functionality of that system. For example, a university information system manages, among others, the examination marks of students for testimonializing. The authenticity and integrity of students' marks are critical for the system's correct functionality; without authenticity and integrity of students' marks the required functionality of testimonializing cannot be provided by the system.

Already this small example shows, that security properties are tailored to a system according to the system's security requirements. In order to develop security properties, IT systems increasingly apply security policies — strategies designed to meet the security requirements of IT systems [1] — and *Trusted Computing Bases* (TCBs) are responsible for pursuing these security strategies in IT systems. In doing so, TCBs are exclusively in charge of establishing and preserving any security property of IT systems. For this reason, a TCB's correctness and tamperproofness are crucial criteria for establishing, preserving and thus guaranteeing a system's security properties.

We consider a TCB to contain all functionalities required to establish and preserve a system's security properties. More precisely, we define a TCB as those system functionalities, and only those, which are necessary and sufficient to establish and preserve a system's security properties.

We differentiate between a TCB, its *security architecture*, and the security architecture's *implementation*. That part of a system architecture whose functional properties are defined by the system's TCB, is called the system's security architecture. In contrast to system architectures, there are specific non-functional requirements for security architectures, e.g. the reference monitor principles [2, 3], which define particular architectural patterns for security architectures. Data types, functions, and algorithms within the implementation of a security architecture representing the TCB's functionalities are then referred to as the security architecture's implementation which is composed of cooperating *security mechanisms*. Figure 1 shows the interrelations between a TCB, its security architecture and security mechanisms.

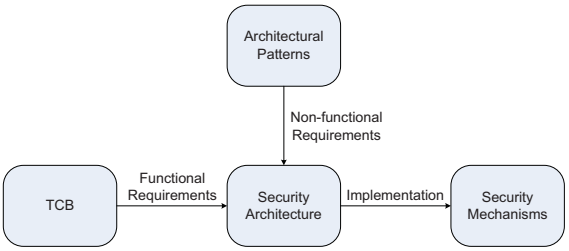


**Fig. 1**. Interrelations of TCBs, Security Architectures, and Security Mechanisms

## 2. MOTIVATION

The lower the complexity of a TCB and the smaller its functional range, the easier it is to verify the correctness and tamperproofness of a TCB's implementation

This work is supported by Carl-Zeiss-Stiftung.

[3, 4, 5]. Designing a small TCB with low complexity then sets the course for allowing its security architecture to be strictly isolated from all security-insensitive system components. This, in turn, provides a promising breeding ground for a small implementation with low complexity and verifiable correctness and tamperproofness.

Considering today's commodity systems, however, we make the observation that their TCBs and security architectures do not meet these design criteria. The prime driver is that system designs are guided by seeking generality, in order to be both useful and usable [6]. This results in large and complex, general-purpose systems providing a wide variety of security properties for many purposes. TCBs of such systems — in the following referred to as *evolved TCBs* — have not been engineered; instead they have arbitrarily evolved for decades due to ever growing and changing security requirements. This leads to a large, complex, and chaotic design, and the resulting security architectures inherit these properties. Consequently, implementations of these security architectures are also large and complex with hundreds of thousands of lines of code; and a positive correlation between software complexity and coding errors is well known [7].

Moreover, it can be observed that more and more applications confide in their own security properties, e.g. web browsers rely on a wide variety of security plug-ins to prevent executing malicious code. The reasons are twofold. On the one hand, the high probability of errors and faults results in a lack of trust in the system's security properties. On the other hand, even though today's systems provide numerous security properties, they may be inadequate for applications since they do not fulfill the applications' security requirements. Therefore, applications increasingly try to establish security properties on their own, which are then immanent parts of the applications since they are implemented by application-specific software components. It follows that security properties are not only established and preserved by a system's core component — the operating system — but also by middleware and application components. This leads to heterogeneous functional ranges of TCBs, and implementations of such TCBs are composed by many independent and not always cooperating security mechanisms resulting in that the same security properties are implemented several times by several mechanisms. This again results in even larger and more complex implementations.

These problems have been known for a long time and are mainly caused by the arbitrary evolvement of TCBs. However, research projects such as [8, 9, 10, 11, 12] show that it doesn't have to be that way. These projects aim at IT systems supporting a wide variety of problem-specific security policies like operating systems that support a wide range of applications. The results are *universal TCBs* which allow for flexibly integrating problem-specific security policies that can now be carefully engineered according to a system's security requirements. Whenever a system's security requirements change due to organizational or technical reasons, e.g. by installing new application software, the policy specification can be reengineered and integrated again in the TCB without having to adapt the TCB itself. The basis is a shift in the TCB design which is now guided by seeking centralization of TCB components. This leads to a new TCB component on the operating system level — the policy manager — and overall there are fewer and smaller TCB components on the application level as well as smaller and less complex TCB components on the operating system level. Nevertheless, universal TCBs are still large, complex and distributed over different system layers such that their security architectures cannot be properly isolated from security-insensitive system components which, in turn, still inevitably leads to complex and large security architectures whose correctness, robustness and tamperproofness are hard to verify.

## 3. CAUSAL TCBS

IT systems with advanced security requirements increasingly apply security policies for describing, analyzing and implementing security properties [13, 14, 15, 12]. In order to precisely describe security policies, formal security models such as [16, 17, 18, 19] are applied, allowing for formal analyses of security properties and serving as specifications from which policy implementations are generated [20]. Consequently, security policies specify the functional requirements of TCBs.

Our objective is to set the course for verifying the correctness and tamperproofness of a TCB's implementation by establishing a small and functionally minimal implementation of a TCB. We are convinced, that the essential prerequisite for such an implementation is a functionally minimal security architecture which again is based on a functionally minimal TCB. Our approach deals with the starting point of these dependencies — the functional range of TCBs.

The proposed approach is based on causal dependencies between a system's security policies and TCB functions. The objective is to design causally determined TCBs containing those functions, and only those, which are necessary and sufficient to enforce and protect the TCBs' security policies. Hence, the distinguishing characteristic of causally determined TCBs is that each TCB functionality is precisely motivated by the system's security policies (Figure 2). As a result, causal TCBs are functionally minimal regarding the system's security policies, and the TCB's complexity represents the security policies' complexity.

Our position is that pursuing this approach does

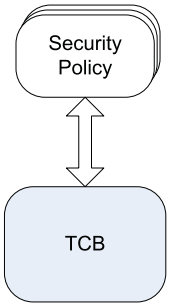


**Fig. 2**. Causal Dependencies

not lead to evolved or universal TCBs; instead resulting TCBs are *causal TCBs* which are tailored to and causally-driven by their security policies. Deriving a TCB's functional range from security policies based on causal dependencies results in both policy runtime as well as policy protection support (policy independent), as well as policy-dependent functionality. Therefore, *causal TCBs* are split into three functionality parts: a *TCB-kernel*, *TCB-modules*, and *security-policy-modules* (SP-modules) as shown in Figure 3. The TCB-kernel provides the runtime system for security policies and contains those, and only those functions, which are required by all security policies in equal measure (*kernel functionality*). TCB-modules consist of those, and only those functions, which are required by all security policies to protect and encapsulate them (*module functionality*). SP-modules are policy-specific and contain those, and only those functions, which are required by problem-specific security policy components (policy-specific *SP functionality*).

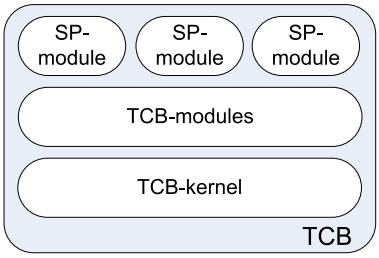


**Fig. 3**. TCB Design

Security policies are not static; instead whenever security requirements of IT systems change due to technical or organizational reasons, the system's security policies may have to be adapted. In today's commodity systems organizational reasons are alterations of a company's policy, e.g. because of new business areas, projects, jurisdictions, or mergers. Technical reasons are modifications of systems, e.g. by installing, removing, or updating application software. These examples show, that security requirements of today's commodity systems can change both rarely, e.g. in case of merger, as well as frequently, e.g. by doing software updates at regular intervals. This situation will be intensified since systems are moving from today's standalone to service oriented systems allowing for spontaneous communication and interaction. Future systems are very dynamic applications and security requirements will change spontaneously much more often. Because of this combination of flexibility and dynamics along with the growing demand for security, systems also need to provide concepts that are able to manage the increasing dynamics of security policies.

Security policy dynamics as well as functional minimality of causal TCBs regarding a system's security policies result in the need of causal TCBs to be able to flexibly adapt to security policy dynamics and thus become dynamic. Our objective is to equip causal TCBs with functional scalability properties such that dynamics of security properties is reflected in TCBs' dynamics. Prerequisites are rules which precisely define security policy components and their interrelations. Based on these rules, scalability properties then specify whether new functions (derived from causal dependencies) have to added to or removed from a TCB, and which interrelations between TCB functions emerge/drop when adding/removing functions.

Since the TCB-kernel provides a runtime system for any security policy and particularly does not contain any other functionality that is not necessary and sufficient for a security policy runtime system, alterations of a system's security policies do not have any impact on the kernel functions but only on the module and SP functionality. We consider TCB-modules and SP-modules to be sets of TCB functions. Thus, set characteristics hold and a set is scalable with respect to the number of its elements by definition — an element is added to or removed from a set using set operators and prohibiting element redundancies. Consequently, module and SP functionality can only be added if they are not already contained by either TCB-modules or a specific SP-module. However, since the functionality is implied by the TCB's security policies, removing module and SP functionality due to removal of security policies is critical and may result in functionality gaps — if a function which is required by more than one security policy is removed from a TCB-module, there will be a lack of functions for the remaining security policies of this TCB-module. Functionally scalable TCBs deal with that.

If at all times the TCB-modules and the SP-modules contain exactly those functions implied by the integrated security policies, i.e. each of these and no other functions, then we call a causal TCB *functionally scalable*.

On the one hand, a TCB's functional scalability is similar to a set's scalability by avoiding functional redundancy. On the other hand, functional scalability is beyond a set's scalability, since the TCB's functions are based on causal dependencies of security policies. The

additional value becomes apparent when removing security policies from a TCB. In that case, characteristics of set operators are no longer met, because the relevant TCB functions may only be removed if they are not implied by other security policies that are not removed. This means, that TCB functions might not be removed even if security policies implying them are extracted. Consequently, a causal and functionally scalable TCB contains at all times exactly all functions implied by the integrated security policies.

Concluding, the approach of causal and functionally scalable TCBs promises a new quality of TCBs whose functions are precisely defined by their security policies and which are able to flexibly adapt to security policy dynamics. Consequently, at all times these TCBs are functionally minimal regarding the system's security policies, and the TCB's complexity represents the security policies' complexity. This sets the course for precise arguments about a TCB's size and complexity. In turn, this leads to security architectures whose complexity and size comply with the properties of their security policies. The consequence is that for many practical systems the complexity and size of their security architectures can be significantly reduced. The proposed approach thus sets the course for reducing the number of coding errors and verifying the correctness and tamperproofness of a TCB's implementation.

## 4. CHALLENGES AND IDEAS

The challenges of this approach can be classified along two questions.

1. How to identify causal dependencies between security policies and TCB functions?
2. How to design functional scalability?

However, answers to these questions are not autonomous; in fact, some of the challenges pose similar problems which have to be considered from different points of view and whose solutions have to be properly combined. The challenges raised by both questions as well as ideas to approach them are discussed below.

### 4.1. Identifying Causal Dependencies

**Challenges** Causal dependencies define a mapping from security policies to those TCB functions required to enforce and protect these security policies. In order to identify this security-policies-to-TCB-functions mapping, precise necessary and sufficient conditions are needed. Thus, the challenge is to find these conditions that control the mapping. Security policies are too informal, in order to derive precise causal dependencies, or to analyze security properties. The latter was solved by applying formal security models which not only allowed for analyzing security properties but also for generating policy implementations. The challenge is now, to apply formal security models to our approach in order to identify causal dependencies between security policies and TCB functions. The objective is to derive TCB functionality which can be assigned either to the TCB-kernel, the TCB-modules, or the SP-modules (Figure 4).

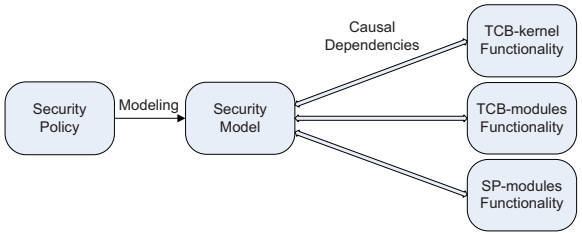


**Fig. 4**. Deriving TCB Functionality from Security Policies and Security Models

**Ideas** For precisely identifying these causal dependencies, we refer to formal security models of security policies since any well-engineered security policy comes with a precise and formal security model, e.g. different kind of access control polices can be modeled with [16, 21, 19, 22], multi-level security policies are based on the Denning model [23], and noninterference security policies are modeled with [17]. Remarkably, all of the aforementioned security models reveal a common, formal model foundation for which rewriting rules can be defined that map the standard model calculi onto the uniform model foundation. This model foundation serves as a uniform basis for all security models. It consists of model components which are required by all security policies (called *core model*). Additionally, the model foundation is extensible by policy-specific model components in order to model problem-specific parts of the security policies (called *model extensions*). Consequently, various kinds of security policies can be modeled with the model foundation and this sets the preconditions for identifying causal dependencies between security policies and TCBs.

### 4.2. Designing Functional Scalability

**Challenges** In order to map security policy dynamics onto TCB functions, we need to identify interrelations between security policies and their components. Thus, the challenge is to define rules that precisely describe these interrelations. For this purpose, a main requirement is to express security policies and their components in such terms that any interrelation becomes explicit. Doing so, raises the following questions: Which interrelations between security policies do exist and how to describe them? Do security policy

interrelations impose additional functions on a TCB's functional range?

**Ideas** Since the formal and uniform model foundation introduced in Section 4.1 allows for modeling various kinds of security policies, we also refer to this model foundation in order to identify security policy interrelations. Already modeling the security policies mentioned above shows, that security policy blood lines exist, allowing us to define rules that precisely describe security policy interrelationships in such a way that whenever a security policy interrelationship exists, there is a security policy lineage. In doing so, we have to clarify whether security policies have to meet any prerequisites in order to belong to the evolution hierarchy, and what the root element of the security policy blood line is.

## 5. CONCLUSION

This approach leads to establishing causal dependencies between security policies and TCB functions. The result is a precise reasoning drawn from a system's security policies about the functions of a system's TCB. These *causal TCBs* are functionally minimal regarding the system's security policies, and the TCB's complexity represents the security policies' complexity.

Functional minimality of causal TCBs regarding a system's security policies, as well as security policy dynamics results in the need of causal TCBs to become able to flexibly adapt to security policy dynamics. This leads to equipping causal TCBs with functional scalability properties such that dynamics of security properties is reflected in TCBs' dynamics.

Modeling causal TCBs within a security architecture then results in security architecture whose complexity and size comply with the properties of their security policies. The bottom line is, that we establish a promising breeding ground for analyzing the correctness, and tamperproofness of TCBs.

## 6. REFERENCES


[1] DSD/GCSB (Australia), CSE (Canada), DCSSI (France), BSI (Germany), ITPA (Japan), NNCSA (The Netherlands), MAPCCN (Spain), CESG (UK), NSA/NIST (USA), *Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 3*, July 2009.

[2] James P. Anderson, "Computer Security Technology Planning Study," Tech. Rep. ESD-TR-73-51, Air Force Electronic Systems Division, Hanscom AFB, Bedford, MA, USA, 1972, Also available as Vol. I, DITCAD-758206. Vol. II DITCAD-772806.

[3] Morrie Gasser, *Building a Secure Computer System*, van Nostrand Reinhold, 1988, ISBN 0-442-23022-2.

[4] Bruce Schneier, "Crypto-Gram Newsletter: Software Complexity and Security," http://www.schneier.com/crypto-gram-0003.html, 2000, Accessed on 30. November 2009.

[5] Lenin Singaravelu, Calton Pu, Hermann Härtig, and Christian Helmuth, "Reducing TCB Complexity for Security-Sensitive Applications: Three Case Studies," in *Proc. of the 2006 EuroSys Conference*. ACM SIGOPS, Apr. 2006, pp. 161–174.

[6] Ben Laurie and Abe Singer, "Choose the Red Pill and the Blue Pill: A Position Paper," in *Proceedings of the 2008 Workshop on New Security Paradigms*, Lake Tahoe, California, USA, 2008, pp. 127–133, ACM.

[7] Norman E. Fenton and Niclas Ohlsson, "Quantitative Analysis of Faults and Failures in a Complex Software System," *IEEE Transactions on Software Engineering*, vol. 26, no. 8, pp. 797–814, 2000.

[8] Peter A. Loscocco and Stephen D. Smalley, "Meeting Critical Security Objectives with Security-Enhanced Linux," in *Proceedings of the 2001 Ottawa Linux Symposium*, 2001.

[9] Robert Watson, Brian Feldman, Adam Migus, and Chris Vance, "Design and implementation of the trustedbsd mac framework," *DARPA Information Survivability Conference and Exposition,*, vol. 1, pp. 38, 2003.

[10] Galen Hunt, James Larus, Martin Abadi, Mark Aiken, Paul Barham, Manuel Fähndrich, Chris Hawblitzel, Orion Hodson, Steven Levi, Nick Murphy, Bjarne Steendgaard, David Tarditi, Ted Wobber, and Brian Zill, "An Overview of the Singularity Project," Tech. Rep. MSR-TR-2005-135, Microsoft Research, 2005.

[11] Nickolai Zeldovich, Silas Boyd-Wickizer, and David Mazières, "Securing distributed systems with information flow control," in *NSDI'08: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, Berkeley, CA, USA, 2008, pp. 293–308, USENIX Association.

[12] Petros Efstathopoulos and Eddie Kohler, "Manageable Fine-Grained Information Flow," in *Proc. of the 2008 EuroSys Conference*. ACM SIGOPS, Apr. 2008, pp. 301–313.

[13] Ciarán Bryce, Winfried E. Kühnhauser, Remy Amouroux, and Mauricio Lopéz, "CWASAR: A European Infrastructure for Secure Electronic Commerce," *Journal of Computer Security, IOS Press*, vol. 5, no. 3, pp. 225–235, 1997.

[14] Udo Halfmann and Winfried E. Kühnhauser, "Embedding Security Policies Into a Distributed Computing Environment," *Operating Systems Review*, vol. 33, no. 2, pp. 51–64, Apr. 1999.

[15] Peter A. Loscocco and Stephen D. Smalley, "Integrating Flexible Support for Security Policies into the Linux Operating System," in *Proceedings of the FREENIX Track, 2001 USENIX Annual Technical Conference, June 25-30, 2001, Boston, Massachusetts, USA*, Clem Cole, Ed. 2001, pp. 29–42, USENIX.

[16] Michael A. Harrison, Walter L. Ruzzo, and Jeffrey D. Ullman, "Protection in Operating Systems," *Communications of the ACM*, vol. 19, no. 8, pp. 461–471, Aug. 1976.

[17] J.A. Goguen and J. Meseguer, "Security Policies and Security Models," in *Proceedings of the IEEE Symposium on Security and Privacy*. Apr. 1982, pp. 11–20, IEEE.

[18] David F.C. Brewer and Michael J. Nash, "The Chinese Wall Security Policy," in *Proceedings of the IEEE Symposium on Security and Privacy*. May 1989, pp. 206–214, IEEE Computer Society Press.

[19] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman, "Role-Based Access Control Models," *IEEE Computer*, vol. 29, no. 2, pp. 38–47, 1996.

[20] Sabrina De Capitani di Vimercati, Pierangela Samarati, and Sushil Jajodia, "Policies, Models, and Languages for Access Control," in *4th International Workshop on Databases in Networkes Information Systems (DNIS 2005)*, vol. 3433/2005 of *LNCS*, pp. 225–237. Springer, 2005.

[21] Ravi S. Sandhu, "The Typed Access Matrix Model," in *Proceedings of the IEEE Symposium on Security and Privacy*. May 1992, pp. 122–136, IEEE.

[22] Xinwen Zhang, Yingjiu Li, and Divya Nalla, "An Attribute-based Access Matrix Model," in *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, New York, NY, USA, 2005, pp. 359–363, ACM.

[23] Dorothy E. Denning, "A Lattice Model of Secure Information Flow," *Communications of the ACM*, vol. 19, no. 5, pp. 236–242, May 1976.