European Multilateral Secure Computing Base

Open Trusted Computing for You and Me

Ahmad-Reza Sadeghi, Christian Stüble, Norbert Pohlmann

The proposed open computing platform solves the security problems of conventional platforms through an efficient migration of existing operating systems, a Security Software Layer and hardware functionalities offered by Trusted Computing. In the sense of multilateral security, this platform allows the enforcement of security policies of different parties. Consequently, the platform enables the realisation of various innovative business models, particularly in the area of Digital Rights Management while averting the potential risks of Trusted Computing platforms regarding privacy issues.

State of the art

Existing networked computing platforms are not able to fulfil the multilateral security requirements of all involved parties, i.e., companies, end-users, and content providers. This can be seen by the huge number of exploits and security updates as well as the high number of attacks through viruses, worms and Trojan horses[1]. Furthermore, the security of existing computing platforms could not be vitally improved in the last years due to the conceptual weaknesses, e.g., their monolithic architecture and thus the increased complexity. This pertains Windows-based operating systems as well as Linux-based ones.

Most of the currently used IT-systems lack elemental security properties, such as integrity checks (keyword: *secure booting*) or the generation of secure cryptographic keys using appropriate random number generators. Thus, the existing threats thwart the realisation of a variety of useful applications and business models, particularly in the area of Digital Rights Management (DRM).

Trusted Computing Technology (TC) provides useful functionalities, but is not able to solve the present security problems *without* a secure and trustworthy operating system: The operating system is the instance that controls all information flows above the hardware layer, and has therefore access to all security relevant data (see, e.g., [37]).

Up to now, there exists no *open* platform which offers the necessary basis for the realisation of multilateral security based on TC-extensions such as TPM or the LaGrande technology.

System Description of Proposed Platform

The European Multilateral Secure Computing Base (EMSCB) we propose combines existing operating systems with the Security Software Layer PERSEUS[2] and the hardware functionality offered by Trusted Computing, as shown in Figure 1.

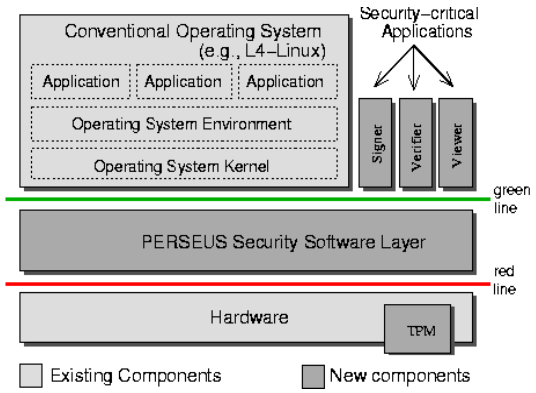

Figure 1: The PERSEUS security software layer works as a control instance between conventional operating systems, security relevant applications, and hardware.

The main components of the proposed platform are:

1. Trusted Computing: Trusted Computing offers hardware upon which a trustworthy system can be built. Thus, it offers functionalities using which it is possible to (i) remotely verify the integrity of the underlying platform (attestation and secure booting), (ii) bind secret keys to a specific platform configuration (sealing), (iii) generate secure random numbers, and to (iv) securely store cryptographic keys.

[1] One example here is the problem of presentation (What You See Is Not What You Get)[33].

[2] www.perseus-os.org

The proposed platform is universal, i.e., independent of the concrete realisation of TC-hardware. It can benefit from the TPM-Chip [43], which was specified by the Trusted Computing Group (TCG) as well as from the LaGrande Technology [21], which is developed by the Intel Corporation.

2. PERSEUS: This is a µ-kernel based Security Software Layer [31][32][37]. It is a very small open-source security kernel, which controls all critical hardware resources (incl. TC-hardware) to protect security relevant applications and sensitive data. In contrast to virtual machine monitors (VMM), PERSEUS does not virtualise the interface of the underlying hardware. Instead, it provides more abstract interfaces that allow a secure and efficient virtualisation of operating system services. The underlying idea of PERSEUS was developed at the University of Saarland in 1999 in cooperation with IBM Research Zurich and was steadily refined in the last years. The PERSEUS project is currently pursued by *eurobits* (European Competence Center for IT Security) at the University of Bochum, Germany.

3. Existing operating systems: A conventional operating system (e.g., Linux) is executed as a task that is controlled and protected by the security software layer, so that the user is able to use the working environment she is used to. Therefore, there will be no incompatibility problems, existing Linux or Windows applications are still usable.[3]

4. New Applications: In parallel to the existing operating system it is possible to execute security-ciritical applications (e.g., digital signatures, DRM applications, etc.), which will be able to use the new features of the TC-technology. EMSCB protects these applications from each other and from the existing operating system (e.g., its viruses, Trojan horses or local user accounts) using approved memory protection mechanisms provided by the CPU and a secure Inter-Process Communication (IPC) offered by the underlying security software layer.

[3] Until now, the existing implementations provide Linux as a tamed operating system and allow the use of Windows applications based on VMware or WINE. If necessary, a naive Windows client could be developed by adapting the Hardware Abstraction Layer (HAL) of Windows.

Advantages

Our proposed platform combines in an innovative way the advantages of a trustworthy open (source security) software layer (PERSEUS) with those provided by trusted computing technology:

1. Multilateral Security: The platform should allow the enforcement of local (e.g., end-user) and external (e.g., content-provider) security resp. access policies. Since we always prefer solutions that consider fair-use and privacy aspects, we avoid the potential dangers of commercial trusted computing platforms the public and the open-source community are concerned about. EMSCB will provide, for instance, a cryptographic solution we call *property-based attestation* [40] that makes software updates and backups easier and prevents discrimination of open-source software.

The proposed platform aims to achieve the following goals: On the one hand it provides users better protection against malicious code (e.g., Trojan horses or viruses), but also against violations of personal security policies. On the other hand, it protects content providers against circumvention of their license agreements, of course, if the consumer has already accepted them.

In contrast to the huge number of existing solutions, our trusted computing platform cannot be circumvented by software attacks. Since TC-hardware is tamper-resistant[4], circumvention of security mechanisms is only possible by performing costly hardware analysis or complex hardware manipulations which is not feasible to be done by ordinary users.

2. Open architecture: Due to the open architecture and the reasonable complexity of security relevant components, this platform has a very high reliability and credibility. The reduced complexity decreases the probability of failures during the development and maintenance process, which in turn increases the trustworthiness of the implementation. In addition to this, an evaluation according to security standards, e.g., the Common Criteria, seems to be possible. Additionally, the open architecture allows necessary updates, improvements, and adaptations of the platform to individual requirements without being dependent on a specific manufacturer.

[4] The current TCG specification version 1.2 demands a Common Criteria evaluation assurance level EAL4.

3. Trustworthy usage of the TCG-technology: Critics of trusted computing are concerned that limited control of the platform by the end-users can principally be misused to deploy censorship, violate the privacy, or restrict the rights of end-users [1][3][5][18][38]. This inherent conflict between the interests respectively security requirements of end-users (protection of privacy and self-determination) and those of content and application providers can be solved by a multilateral trustworthy computing platform that guarantees a balance of the interests of all involved parties [13][20][37].

EMSCB compares in the sense of multilateral security the postulated security requirements of the user with the licence agreements of software to be installed and prevents the installation in case of a conflict.

Moreover, the openness of the proposed platform allows users to evaluate the design and the source code themselves obtaining assurance about the functionalities, e.g., that a system-wide censorship is not provided.

4. Low-cost portability: Since security-critical components of the platform only depend on the interface provided by the underlying µ-kernel, this platform allows a very efficient migration to additional devices, such as PDAs, smart phones and embedded systems. Application examples can be found in new applications of multimedia and information systems, e.g., of the automotive industry.

5. Future assurance: The architecture we propose is compatible to existing operating systems. Future impact and importance of TC-based operating systems is underpinned by the efforts of the existing operating system monopolist Microsoft in the context of its Next-Generation Secure Computing Base (NGSCB). Through an alternative and open platform security-critical applications may, to a reasonable degree, remain independent of operating system manufacturer ensuring the future usage capability of corresponding applications with regards to new demands.

Applications

The proposed platform allows the realization of a variety of business models relying on distributed trusted third parties, or a considerably more efficient configuration of some of the existing applications. In the following, we are going to consider some interesting applications of increasing importance.

Distributed policy enforcement: Existing technical measures of copyright handling on digital content resp. services (see, e.g., [9] [28][35]) on end-user devices only registered moderate success, since most of the technical solutions can be totally controlled by the end-users due to the lack of appropriate protection in hardware and software.[5] Experiences in the past have shown that hardware solutions (e.g. dongles) cannot be established because of their high complexity, incompatibility, insufficient security, and limited user acceptance [2]. Moreover, a variety of these techniques were treated as trade secrets; a strategy which contradicts the cryptographic principals, because security should not rely on the secrecy of an algorithm but on the secrecy of a secret parameter (e.g., cryptographic keys). In spite of nondisclosure and legal threats by content providers, most of the methods have been broken in the past (see, e.g., [2][8]).

In contrast to existing insecure solutions, the proposed platform combines the Security Software Layer (PERSEUS) and the features offered by the TC-hardware and provides the appropriate basis for the realisation of more secure applications.

For instance, it can enforce licence agreements, if these were accepted by the consumer of digital content: On the one hand, the platform ensures that users of online-information (e.g., travelling or navigation information, electronic magazines, etc.) can get access to the desired information only against payments, and that they cannot arbitrarily distribute this information to others. On the other hand it prevents that providers get more private information about the user than actually needed for providing the offered service.

Possible applications with short term potential are copyright protection, eLearning, eBooks, geographical information systems, as well as the area of Telematics in car navigation systems.

Another field of application is the long-term high sale-expecting area of providing multimedia content, e.g., video and audio data. Here the platform will considerably complicate the unauthorised distribution of digital content.[6]

Certainly, this platform will build the basis for a *pragmatic fair* copyright protection. In this context, we are particularly interested in adapting the development of our platform to the concepts of *fair use* and *first sale*, which allow the private (e.g. one-time transfer) or non-profit (e.g., for educational purpose) usage of the content [13][35].

Compartmented mode security: Business processes between companies often require the exchange of sensitive data and documents (e.g., financial accounting, patent motions, technical cooperation), whose usage is regulated by contracts (e.g., through secrecy acknowledgements). Company-internal protection measures are essential as well, so that access on documents outside the desired workflow is prohibited. This, for example, shall prevent that employees read sensitive documents, distribute documents (accidentally or purposely) outside the company or perform unauthorized changes.

The existing computing platforms cannot securely handle with classified documents (e.g., unclassified, secret, top secret), so that the users can circumvent control mechanisms by using available functions for their own purpose or by exploiting known security holes of existing software components.

Many security problems occur, because companies or public departments are not able to successfully prevent their users to (accidentally or purposely) break the security policies. They are able to install software components on their own or manipulate the IT-system otherwise, which leads to potential security lacks, e.g., through viruses, Trojan horses, worms and configuration errors.

EMSCB will provide functionalities that allow to securely enforce external and company-wide security policies. This is the basis for the realisation of a system with Multi-Level Security (MLS), which is customized by practical conditions. Existing MLS-solutions are not satisfactory up to now because of their high complexity resp. inefficient configuration (strictly separated hardware).

Another important example application, which will be realisable in association with a secure computing platform, are Multi-Server Systems (MSS), which run, like virtual machine monitors (VMM), different isolated services (e.g., a database, a web-server, and a security gateway) parallel on one server.

Secure end-user systems: Today, a standard personal computer or mobile device, with an off-the-shelf operating system and all the software that one mainly buys for this system, is not secure at all, particularly in the context of digital signatures, eCommerce and eGovernment. Different applications of the same user are not protected from each other and the end-users are confronted with frequent security updates. Moreover, almost all data may nowadays carry executable code and the execution often starts without knowledge of the computer owner. Hence, it is impossible to administer a standard end-user system such that a critical application is protected from all others.

The proposed platform offers with its secure booting and authentication mechanisms a necessary and sufficient basis for security relevant applications like secure signature generation, home banking or eGovernment and eCommerce applications.

Embedded security: Another important application area for this security platform arises due to increasing integration of computer platforms in different products and devices (embedded systems), e.g., as done by the automotive industry. The high complexity of the used software leads to higher error probability, which can be compensated by the use of a security kernel. Furthermore, the integration of information- and multimedia systems in cars (infotainment) will play an important role in the future, which will offer new business opportunities for suppliers and manufacturers. Our platform offers the required features to develop many innovative products, especially in the automotive industry [30][36][39].

[5] Reaching manipulation protection in software (secure container) is highly difficult, if not even impossible (see e.g., the problem of software solutions using so-called "code *obfuscation*" [4] or [16] for attacks on different software-based DRM-solutions).

[6] Nevertheless, we should emphasize that the capability of computing platforms to *prevent* unauthorized copying of multimedia content is limited, since users can always make analog copies.

Related Work

We have to emphasise that technologies like smartcards or firewalls do not increase security of existing operating systems; trustworthy hardware and trustworthy software basis are always needed. In the past, several approaches for a secure operating system have been published. Most of them were results of research projects, like EROS [42] or Multics [29], but until now, they have a shadowy existence, because they are not compatible to common operating systems.

Another common approach is the hardening of an existing operating system through eliminating of conceptual weaknesses, e.g., SE-Linux [26]. Certainly, this is very risky, because the complexity of existing operating systems has become very high. A never ending installation of patches to eliminate security holes is the consequence and not the goal.

EMSCB does not have these disadvantages, because all security relevant components and applications can operate independently from the common operating system. This also allows a careful and economical evaluation of security-critical components according to the Common Criteria.

In the following section, we briefly compare the proposed EMSCB architecture with two alternative trusted computing architectures, namely Microsoft's Next-Generation Secure Computing Base (NGSCB) and Terra.

NGSCB

With Next-Generation Secure Computing Base (NGSCB) Microsoft presents a security platform based on the TCG-specifications [27], which will be integrated in future Windows versions. Unfortunately, little technical information about NGSCB has been published, and existing documents seem to be outdated. Based on the available documents and presentations, one can highlight two hardware extensions as the most important differences between TCG and NGSCB:

The first one is a modification of the CPU that allows to execute a security kernel (called Nexus) in parallel to (resp. below) a conventional operating system. Different realisations have been suggested by Microsoft so far (see, e.g., [7]) that mainly differ in the level of virtualizability of the CPU[7]. The main advantage of this improvement is that unmodified conventional operating systems can efficiently be used.[8]

The second hardware extension solves the problem of Direct Memory Access (DMA) enabled adapters (e.g., graphic, sound, and network card) that can access every physical memory region and thus bypass all security mechanisms provided by the software (see, e.g., [22] and [31] for a more detailed discussion). The NGSCB solution is to encrypt the data channel between the Nexus and the hardware adapters. The additional advantage of this approach is that device drivers not necessarily have to be trusted any more, since they cannot access the transmitted content (e.g., a secret key written to the hard disk or a movie sent to the video card).

In contrast to NGSCB, our proposed platform is using the functions of the TC-Hardware as a Black-Box. Therefore, it can, dependent on the demanded security-level, be used with or without a TPM, as well as with NGSCB-hardware or other, future TC-technologies. The user of our platform is not committed to a specific hardware vendor, since the critical security component TPM can be supplied by different companies.

The second important difference between EMSCB and NGSCB is the enforcement of multilateral security under consideration of end-user policies and the German and European legislation. In general it cannot be assumed, according to legislation, that the NGSCB architecture is offering a comparable component. An *alternative Nexus* ensures the privacy of the user particularly in the context of DRM-capable platforms.

Terra

Terra is a trusted virtual machine monitor (VMM) proposed by the Stanford University [11]. The Terra VMM partitions a hardware platform into multiple, isolated virtual machines (VM) which can be either a so-called "closed box", or an "open box". Applications running in a closed box can cryptographically identify itself using the attestation function offered by the TCG hardware, while open boxes are used to execute uncritical code. Terra and EMSCB have the following differences:

As it is common to all VMMs [12], Terra virtualises the interfaces offered by the underlying hardware. This allows the use of unmodified conventional software, but leads to a lot of performance overhead and increased complexity. Using our approach, it is possible to provide a VMM as a µ-kernel application [45]. This moves the complexity of the VMM out of the security-critical kernel and allows the development of optimised applications based on the interface offered by the PERSEUS layer.

Terra (VMMs in general) strictly separates the different VMs from each other, while our approach allows a controlled communication between processes. The communication is required for our purposes, since we aim to extend the conventional operating system with security-critical applications rather than providing an isolated secure environment. For instance, we want to be able to securely sign a document that was edited under the conventional operating system. The isolating behaviour of a VMM is only one possible security policy to be enforced, as suggested in the last paragraph of the section about compartmented mode security.

We focus on the realisation of fair use aspects (e.g., property-based attestation [40]) in the context of multilateral security, while Terra (to our knowledge) only implements the trusted computing functionality offered by TC hardware.

The Idea of OpenNexus

One motivation behind the proposed EMSCB architecture is to offer an open alternative to the solutions proposed by the industry (e.g., an OpenNexus that can be used as an open alternative to the nexus offered by NGSCB). In the following, we shortly discuss why an open alternative is, in our opinion, necessary:

Competition. The experience with the open-source operating system Linux has shown that vendors of commercial products will rather fulfil user requirements if there exists an alternative offering similar func-

[7] Because of some design shortcomings, the Intel CPU currently does not allow to build an efficient virtual machine monitor (VMM) on top of it.

[8] Note that until now it was always necessary to either adapt the conventional operating system, or to live with loss of efficiency resulting from inefficient VMMs.

tionality. For instance, there is a general consensus that Windows 2000 and its successors would not be as stable and secure, if Linux as an alternative has not been existed.

Security. From a security perspective the existence of several trusted computing architectures that are used in parallel is to be favoured, since heterogeneous operating system environments are more resistant against attacks of, e.g., viruses and internet worms.

Moreover, open architectures can suffer from the use of the hardware extensions that come with, e.g., NGSCB resp. LaGrande for the following reasons:

- Taming of DMA-enabled devices is currently an open issue. While several solutions are currently under discussion (see, e.g., [15][22]), the encryption of the data channel between security kernel and devices allows to extract device drivers out of the trusted computing base and thus to reduce its size and complexity.
- Improving CPU virtualizability[9] makes it possible to efficiently use existing operating systems without the need to modify them. Thus, cooperation of the operating system manufacturer would not be necessary any more.

Open Interfaces. In the past, some manufacturers were reproached that their programming interfaces have not been fully documented or published too late, so that their own products always had a competition advantage. If security architectures like NGSCB will prevail, this competition advantage will not only touch the area of office-applications, but will relocate the market in the area of security products as well.

EMSCB is an open architecture, and all programming interfaces and the source-code of security relevant components will be published for evaluation to increase the trustworthiness of the implementation. Therefore, an open trusted computing platform is enabling the OpenSource- resp. Linux-Community to prospectively remain competitive.

Compatibility. The compatibility of open solutions and commercial products is of high importance, especially in the context of operating system architectures. It is therefore important to consider to what extend vendors can prevent or aggravate a compatible open-source alternative:

- We first have to emphasise that an open alternative to, e.g., NGSCB, will only depend on the products provided by hardware vendors like Intel, AMD, Infineon, etc. who, until now, published their specifications also to the open-source community. Moreover, Microsoft announced not to aggravate open source alternatives to their NGSCB architecture.
- According to the currently available documentation about NGSCB, several implementations of the Nexus kernel can alternatively be used. Therefore it is possible, for instance, to use the Microsoft Nexus to run the media player, and to use an open alternative (e.g., EMSCB) to securely sign a contract.
- For the long term, the open source community can aim to provide a binary-compatible interface (but, of course, a multilateral-secure implementation) to commercial Nexus kernels. The open-source project WINE[10] and the commercial CrossOver Office[11] demonstrate, that this is hard, but possible. Providing a compatible interface to the Nexus should be easier, because of its reduced complexity and because Microsoft announced to publish the source-code of their Nexus. If they do not publish it, the need for an open alternative will then be stronger.

Technical Realisation

Conventional operating systems like Microsoft's Windows or Linux have monolithic kernels which means that all operating system services that require special privileges are summarised in one component that has full control over the hardware and the applications. Examples of such services are process- and memory management, but also file systems, a TCP/IP stack and device drivers. The concept of monolithic operating system kernels does not follow the least privilege paradigm, since all kernel components share all privileges. Today, monolithic kernels can become very complex, which increases the probability of security critical bugs. Additionally, source code of device drivers frequently changes, since new hardware of peripheral devices appear in open architectures in short intervals. As a consequence, monolithic operating systems cannot become stable over time. The result of this facts are frequent security patches that vainly try to solve existing security holes[12].

The basic idea behind the µ-kernel approach is to minimise the security-critical part and to implement outside the kernel whatever possible [23]. Nearly all operating system services can be extracted into separated processes – providing isolation between them and to the µ-kernel through conventional memory protection mechanisms. The µ-kernel itself contains only elementary functions that require the highest level of privileges, namely process management, memory management, and interprocess communication (IPC). Therefore, the concept of µ-kernels has (not only) from a security perspective several advantages compared to monolithic kernels:

- Operating system services act independent from each other, therefore it is possible to follow the least privilege paradigm and assign to them only those privileges that are necessary to perform their task. Malfunction or malicious misbehaviour is locally isolated.
- The complexity of µ-kernels is drastically reduced compared to monolithic kernels. For instance, the L4 µ-kernel [24], which we use, contains only about 7000 lines of code, while the current Linux kernel contains more than 2.8 Million source code lines.
- Since a µ-kernel provides only elementary functions based on concepts that change only slowly, the µ-kernel code has the chance to become stable over time.
- The reduced complexity of µ-kernels makes it possible to prove the correct-

[9] Robin and Irvine identified seventeen instructions of the Intel x86 architecture that violate requirements of efficient VMM realisations [34]. While earlier documents about NGSCB suggested to improve only a minimal number of instructions [7], one can derive from current design studies that Microsoft plans to improve a larger instruction set to be able to use the Nexus as a VMM.

[10] www.winehq.com

[11] www.codeweavers.com

[12] Microsoft observed that very often exploits appear *after* a security patch has been published.

ness of the implementation using formal methods.
- Moreover, the reduced complexity and the stability of the µ-kernel implementation makes an evaluation, e.g., according to the Common Criteria, efficient, since re-evaluations are time consuming and expensive.

Figure 2 shows an overview of the PERSEUS security software layer.

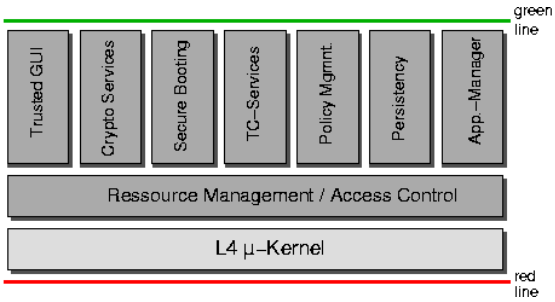

Figure 2: A more detailed view into the PERSEUS layer; isolated processes provide security-critical services and share hardware resources.

The most important services include:

Secure Booting. We extended the open-source bootloader GRUB[13] by TCG support necessary to realise secure booting, attestation, and sealing. Our *TrustedGRUB* can be freely downloaded from our project homepage.[14]

µ-Kernel. We decided to use the µ-kernel implementations of the L4 family[15] that are developed at the University of Karlsruhe [25] and at the Technical University of Dresden [17]. Besides the advantage that L4 kernels are very efficient, they already provide a L4/Linux implementation [14] that only requires slightly modifications to adopt it to our security-relevant requirements.

Resource Management. On top of the µ-kernel, elementary resource management services (e.g., memory management, process management, interrupt management, etc.) are executed. The services are also responsible to provide device drivers (including a TPM-driver), and to enforce system-wide security policies.

Secure User Interface. Based on the resource management layer, a secure user interface service provides a *trusted path* between applications and the user including a TrustedGUI. It ensures that user input (e.g., passwords) cannot be eavesdropped by unauthorized applications and protects the integrity of application output necessary, e.g., to realise a *trusted viewer*. Authenticity of application output is required to prevent Trojan horse attacks (see, e.g., [44]). We currently implement an improved version of the labelling policy suggested by Epstein [10]. Additionally, the secure user interface allows information flows between applications (e.g., copy & paste) only according to a local security policy.

Application Manager. This service ensures a controlled installation of new applications. Firstly, it ensures that only applications that are compatible to the end-user policy can be installed and derives the appropriate privileges. Secondly, the application manager provides for the secure user interface the necessary information used to distinguish different applications (e.g., by unique names and icons).

Trusted Computing Service. This service virtualises the functions offered by the trusted computing hardware and thus allows applications to use attestation and sealing functionality.

Conclusion

Trusted Computing allows security engineers to build more secure computing platforms. It is therefore realistic to assume that trusted computing hardware will be increasingly deployed in the future.

Nevertheless, the use of trusted computing hardware makes only sense in combination with a trustworthy software stack between hardware and applications. Moreover, the limited control of the platform by end-users can be misused.

In this paper, we propose the development of an open security kernel that solves this conflicts in the sense of multilateral security by considering both security and privacy requirements of content providers and end-users. We proposed a µ-kernel based security architecture and presented our implementation results, e.g., a secure boot loader and a secure user interface service necessary to realise a trusted viewer.

The use of open architectures for security-critical applications has many advantages. To accomplish the required high acceptance of an open architecture, it is in our opinion important to focus existing competences within international projects. Finally the governments and the public have to be willing to support an open and trustworthy security architecture.

Literature

[1] Ross Anderson: „TCG/Palladium Frequently Asked Questions, Version 1.0”,http://www.cl.cam.ac.uk/~rja14/ TCG-faq.html.

[2] Ross Anderson: Security Engineering: A guide to building dependable distributed systems, John Wiley & Sons, 2001.

[3] Bill Arbaugh: „Improving the TCG Specification”, IEEE Computer, S. 77-79, August, 2002.

[4] Boaz Barak, Oded Goldreich, Russel Impagliazzo, Steven Rudich, Amit Sahei, Salil Vadhan and Ke Yang: “On the Impossibility of Obfuscating Programs”, CRYPTO, 2001.

[5] Stefan Bechthold: „Trusted Computing Initiatives – Protecting Virtual Troy or Creating a Trojan Horse?”, in [41].

[6] CEN, European Committee for Standardization; Digital Rights Management, Final Report, 2003, available at http://www.cenorm.be/cenorm/.

[7] Yuqun Chen, Paul England, Marcus Peinado, Bryan Willman: “High Assurance Computing on Open Hardware Architectures”, Microsoft re-Rearch Technical Report MSR-TR-2003-20, March 2003.

[8] Scott A. Craver, Min Wu, Bede Liu, Drew Dien, Edwar W. Felton: “Reading Between the Lines: Lessons Learnd form SDMI Challenge”, USNIX 2001.

[9] Digital Rights Management, Technological, Economics, Legal and Political Aspects, Volume 2770 of Lecture Notes in Computer Science, Springer-Verlag Berlin.

[10] Jeremy Epstein: “A Prototype for Trusted X Labeling Policies”, Annual

[13] www.gnu.org/software/grub/
[14] www.prosec.rub.de
[15] www.l4hq.org

Computer Security Application Conference (ACSAC), 1990.

[11] Tal Garfinkel, Ben Pfaff, Jim Chow, Mendel Rosenblum, Dan Boneh: "Terra: A Virtual Machine-Based Platform for Trusted Computing", In proceedings of the 19th ACM symposium on Operating System Principles, 2003.

[12] R. Goldberg: "Survey of virtual machine research", IEEE Computer Magazine 7:34-45, June 1945.

[13] Dirk Günnewig, Ahmad-Reza Sadeghi, Christian Stüble: Trusted Computing ohne Nebenwirkungen Datenschutz und Datensicherheit (DuD) 9/2003, Vieweg Verlag, pp. 556-560.

[14] Hermann Härtig, Michael Hohmuth, Jean Wolter: "Taming Linux", In Proceedings of PART, 1998.

[15] H. Härtig, J. Loeser, F. Mehnert, L. Reuther, M. Pohlack, A. Warg: "An I/O Architecture for Mikrokernel-Based Operating Systems", Technical Report TUD-FI03-08-Juli-2003, TU Dresden, July 2003.

[16] Tobias Hauser, Christian Wenz: DRM Under Attacks: Weaknesses in Existing Systems, Digital Rights Management, Technological, Economics, Legal and Political Aspects, Volume 2770 of Lecture Notes in Computer Science, pp. 206-223, 2003. Springer-Verlag Berlin.

[17] Michael Hohmuth: „The Fiasco Kernel: Requirements Specification", TU Dresden Technical Report TUD-FI98-12, 1998.

[18] Christian Koenig, Andreas Neumann: „Wettbewerbsrechtliche Aspekte vertrauenswürdiger Systemumgebungen", in [41].

[19] Dirk Kuhlmann, Robert A. Gehring: Trusted Platforms, DRM, and Beyond, Digital Rights Management, Technological, Economics, Legal and Political Aspects, Volume 2770 of Lecture Notes in Computer Science, pp. 178-205, 2003. Springer-Verlag Berlin.

[20] Klaus Kursawe and Christian Stüble: Improving End-user Security and Trustworthiness of TCG Platforms. 33. GI-Fachtagung, Frankfurt, 2003.

[21] LaGrande: Technology Archtitectural Overview, Intel White Paper, Intel, September 2003.

[22] Ben Leslie and Gernot Heiser: "Towards Untrusted Device Drivers", Technical Report UNSW-CSE-TR-0303, March 2003.

[23] Jochen Liedke: "On µ-Kernel Construction", In Proceedings of Symposium on Operating System Principles (SOSP), 1995.

[24] Jochen Liedke: "L4 Reference Manual", GMD/IBM Research Technical Report, 1996.

[25] J. Liedtke, U. Dannowski, K. Elphinstone, G. Liefländer, E. Skoglund, V. Uhlig, C. Ceelen, A. Haeberlen, M. Völp: "The L4KA Vision", University of Karlsruhe, 2001. White Paper, April 2001

[26] Peter Loscocco and Stephen Smalley: Integrating Flexible Support for Security Policies into the Linux Operating System, U.S. National Security Agency (NSA), 2001.

[27] Craig Mundie, Pierre de Vries, Peter Haynes, Matt Corwine, Trustworthy Computing, Microsoft Corporation, Oktober, 2003.

[28] National Research Council: The Digital Dilemma, Intellectual Property in the Information Age, National Academy Press, 2000.

[29] Elliott I. Organick: The Multics System: An Examination of its Structure, MIT, 1972.

[30] Christof Paar: „Eingebettete Sicherheit im Automobil", Embedded IT-Security in Cars (ESCAR), Köln, 2003. www.escarconference.org

[31] Birgit Pfitzmann, James Riordan, Christian Stüble, Michael Waidner, Arnd Weber: „The PERSEUS System Architecture", IBM Technical Report #93381, IBM Research Division, Zürich, 2001.

[32] Birgit Pfitzmann, James Riordan, Christian Stüble, Michael Waidner, Arnd Weber: „Die PERSEUS Sicherheitsarchitektur", Verlässliche Informationssysteme (VIS) 2001, S. 1-18, 2001.

[33] Ulrich Pordesch: "Die elektronische Form und das Präsentationsproblem", Nomos Verlaggesellschaft, 2002.

[34] John S. Robin and Cynthia E. Irvine: „Analysis of the Intel Pentium's Ability to Support a Secure Virtual Machine Monitor", 9th USENIX Security Symposium, 2000.

[35] Bill Rosenblatt, Bill Trippe, Stephan Mooney: Digital Rights Management: Business and Technology, John Wiley & Sons, 2001.

[36] Michael Rudorfer: „IT Trends im Fahrzeug", Embedded IT-Security in Cars (ESCAR), Köln, 2003. http://www.escarconference.org.

[37] Ahmad-Reza Sadeghi, Christian Stüble: „Taming "Trusted Computing" by Operating System Design", Proceedings of the 4th International Workshop on Information Security Applications (WISA), Korea, 2003.

[38] Ahmad-Reza Sadeghi and Christian Stüble: Sinn und Unsinn von TCPA und Palladium; c't 13/2003, Heise-Verlag.

[39] Ahmad-Reza Sadeghi, Christian Stüble: „Die Rolling Jukebox – Digital Rights Management in Car", Embedded IT-Security in Cars (ESCAR), Köln, 2003. www.escarconference.org

[40] Ahmad-Reza Sadeghi, Christian Stüble: "Property-based Attestation for Computing Platforms: Caring about policies, not mechanisms", New Security Paradigm Workshop NSPW, Nova Scotia, 2004.

[41] Schriftreihe Kommunikation & Recht "Trusted Computing", Verlag Recht und Wirtschaft Heidelberg, 2004.

[42] Jonathan S. Shapiro, Jonathan S. Smith and David J. Farber: "EROS: A Fast Capability System", Symposium on Operating System Principles (SOSP), 1999.

[43] Trusted Computing Group: „Trusted Platform Module (TPM) Main Specification v1.2", December 2003.

[44] J. D. Tygar and A. Whitten: "WWW Electronic Commerce and Java Trojan Horses", Proceedings of the 2nd USENIX Workshop on Electronic Commerce, 1996.

[45] V. Uhlig, J. LeVasseur, E. Skoglund, and U. Dannowski: "Flexible and Scalable Virtual Machines", in poster session of 19th ACM Symposium on Operating Systems Principles, 2003.