The challenge for legal metrology of operating systems embedded in measuring instruments

F. THIEL, Member of the Department for Metrological Information Technology (florian.thiel@ptb.de)
U. GROTTKER, Head of the Working Group for IT in Legal Metrology (ulrich.grottker@ptb.de)
D. RICHTER, Head of the Department for Metrological Information Technology (dieter.richter@ptb.de)
Physikalisch-Technische Bundesanstalt (PTB), Germany

Key Words: Operating systems, Embedded systems, Software validation, Common Criteria, Protection profile, OIML, D 31, ISO/IEC 15408, Measuring Instruments Directive (MID), WELMEC Guide 7.2, U Type Instrument

Abstract

Software validation is an indispensable part of the type examination process in legal metrology. Since more and more functions are demanded by the users of measuring instruments and consequently offered by the manufacturers, often via the embedded operating system, an increased and evolving level of complexity is introduced which needs to be monitored and catered for.

Nowadays, type examination engineers are confronted with a greater number of risks which must be taken into consideration. It is, therefore, imperative that the validation procedure keep pace with the technological evolution of measuring systems.

However, there is growing conflict between the desired flexibility that an operating system should offer, on the one hand, and the testing effort and challenges that examination engineers are confronted with, on the other hand. Thus, appropriate methods and tools are needed to overcome this challenge.

In this regard an approach has been developed which aims at carrying out automated checks of the parameters related to operating systems embedded in measuring instruments. The origin is a protection profile that was developed for operating systems, and evaluated according to the Common Criteria standard.

The profile has been adapted to the special needs of legal metrology in order to protect software, including the operating system, against accidental, unintentional and/or intentional changes. For practical reasons, this adaptation has been implemented as an applicable test script for Linux-based systems.

With this approach, even without profound knowledge of operating systems, efficient support for the examination of the security-relevant characteristics of measuring instrument software with embedded operating systems is therefore made possible, also for type examination and for verification.

1 Introduction

With the advent of programmable hardware components and their integration into measuring instruments, the influence and hence the importance of software, especially in legal metrology, has remarkably increased. Consequently, the validation of software has become an indispensable part of the type examination process [1], [2], [3].

Software development lead-time, product quality, and mass-customization are important issues in industrial development and manufacturing processes, and so it is comprehensible that manufacturers increasingly rely on programmable components which intrinsically offer a broad ensemble of high level functions which facilitate the development process.

The required functions are provided by, but also limited by, the facilities of the operating system applied - for example graphic performance, processor power, interface drivers, video drivers, and memory management, to name but a few. Therefore, with the availability of embedded personal computers (PC), it followed that this technology should also be exploited for measuring instruments. Instruments based on an embedded PC accompanied by external peripherals, sensors and acquisition interfaces, could accept a large number of commands and data from various nodes (interfaces, sensors, etc.), several different operating modes could be selected, key metrological parameters and software updates could be downloaded to the instrument, and measurement data could be transferred to external modules [4], [5], [6]. But as a consequence, all these features greatly extend the risks with respect to the characteristics to be safeguarded in legal metrology.

It is imperative that the software validation procedure in type examination processes keep pace with technological evolution, in the particular case discussed here, with the increasing use of operating systems in measuring systems.

This paper seeks to offer a principal way of showing how operating systems with growing complexity
embedded into a measuring instrument subject to legal metrological control can be handled in order to ensure conformity with the above-mentioned requirements.

After a brief introduction, the two major operating systems are briefly introduced followed by an illustrative example showing the evolution of the complexity of the Linux kernel with the increasing number of releases.

Subsequently, the contradictions between the general advantages of embedded solutions and the legal metrology regulations are discussed, which results in the analysis of the challenges which are established by the application of operating systems. Starting with an introduction to the Common Criteria approach for security of IT components, protection means for a measuring instruments with embedded operating systems are derived which finally leads to our proposed automatic examination of the system’s configuration.

2 Existing software guidance documents

For the general task of software validation, appropriate support is already offered by OIML D 31:2008 General requirements for software controlled measuring instruments [8], and WELMEC Guide 7.2, available in its current issue 4 [10]. This Guide represents an interpretation of the corresponding parts of the Measuring Instruments Directive (MID) [9].

The objective of these documents is to provide guidance for development engineers of manufacturers as well as for type examination engineers. By doing so, they remove any “uncertainty” as to the interpretation of the more or less generally composed software requirements laid down in regulatory documents such as, for example, in the MID, and establish mutual confidence in the results of software examination.

OIML D 31 may specify details concerning the validation procedure subdivided into general and specific requirements. The general requirements specify the basic requirements which have to be fulfilled by every system. This includes the identification of the legally relevant part of the software, the correctness of the algorithms and functions, the protection against misuse and fraud, i.e. the legally relevant software shall be secured against unauthorized modification, loading, or changes by swapping the memory device.

Furthermore it deals with the support of hardware features by software including fault detection and durability protection. The specific requirements of D 31 cover all deviations from the general requirements, i.e. the validation of the specification and separation of metrologically critical parts and the specification of the interfaces between the parts, the indications of measurement values and other legally relevant information, the storage of data, the transmission via the communication system, the compatibility of the operating system and the hardware, and how to handle maintenance and reconfigurations.

The validation of software with respect to the developed requirements basically remains a demanding task due to the absence of straightforwardly defined procedures, i.e. of dedicated technical standards for metrological software. Despite the fact that when following the guiding documents, a considerable number of evaluations are necessary for the particular implementations they are, nevertheless, useful for a systematic, step by step analysis of the system under test.

In particular, the validity of these requirements implies their regular update. For the increased use of PC-based systems utilizing operating systems (OS), the above-mentioned requirements are still valid but are not qualified to answer all the particular problems that developers and testers are confronted with, because of an increased and evolving level of complexity of the operating systems [11], [12].

3 The major operating systems and their characteristics

The two OS most widely used in measuring systems are Microsoft Windows and Linux. Linux and MS Windows differ in their philosophy, cost, versatility and stability, each seeking to improve in their perceived weaker areas. Comparisons of the two OS tend to reflect their origins, historic user bases and distribution models [14], [15], [16], [17], [18], [19]. Typically perceived and regularly cited weaknesses mostly include poor consumer familiarity with Linux, and MS Windows’ susceptibility to viruses and malware. MS Windows is a series of commercial software operating systems and graphical user interfaces produced by Microsoft. Thus, in contrast to Linux, its code is kept secret (proprietary).

Throughout the entire period of the Windows 9x systems through to the introduction of Windows 7, Windows has retained an extremely large share of retail sales among operating systems for personal desktop use, while Linux has sustained its status as the most prominent free software operating system. After their initial clash, both operating systems moved beyond the user base of the personal computer market and share a rivalry for a variety of other devices, with products for the server and embedded systems markets, and mobile internet access.

Each of the three families of Windows operating systems: Windows 9x (legacy), Windows NT, and Windows Embedded has its own code base and design.
The configuration options are the most essential part of the Linux kernel configuration. The granularity, meaningfulness and complexity of the configuration options and their dependencies within an operating system are fundamental for the end user to successfully configure the kernel in a Linux system [11], [12].

Studies of software complexity have shown that complexity can be measured by so-called software metrics [20]. Consequently, this approach was also applied by R. Lotufo in 2009 [12] to measure the complexity of 29 stable versions of the Linux kernel configuration options in order to study their evolution in terms of complexity [12].

We will not go into the details of complexity analysis, but want to mention certain illustrative parameters (metrics) such as the size and the depth of the kernel. Size is a simple but nonetheless an appropriate metric for software complexity, as shown in [13]. It was found that the larger the size of the scripts, the more configuration options and dependencies between them exist. Another metric is depth as a complexity measure. This can also be understood as the average length of every different path from a configuration option node to its leaves, i.e. configuration options without dependencies. With the increasing depth of a given configuration option, the chain of dependencies lengthens and the more complex it is to comprehend all of its dependencies, and therefore the more complex it is to make changes to them.

The first noticeable thing Lotufo figured out was the steady increase in size, such as in the number of Source Lines Of Code (SLOC), the number of configuration options, and the number of dependencies between them. For example, the number of configuration options increased from approximately 100 in version 2.4.18 to more than 600 in version 2.6.24. This increase is mostly due to the expansion of the kernel and the addition of new features. The average depth of the configuration options also increased, indicating a more complex configuration system. The depth of the configuration options increased from approximately 2 in version 2.4.18 to more than 5 in version 2.6.24. This increase is mostly due to the expansion of the kernel and the addition of new features. The average depth of the configuration options also increased, indicating a more complex configuration system.

The NT family is available on x86, x64, and Itanium, although Itanium compatible versions of Windows are only sold as servers and x86 is being phased out. Because of the diversity of supported CPU types, Linux finds applications today in routers, set-top boxes, PDAs and mobile phones as well as in servers and desktops. Windows Embedded has a long history, starting with DOS on Point Of Sale (POS) terminals. Microsoft has based many embedded platforms on the core Windows CE operating system, including AutoPC, Windows Mobile, Mediaroom, Portable Media Center, and many industrial devices and embedded systems.

In the following we will assume a Linux-based system, since this operating system and its derivatives are commonly used to facilitate the development of PC-based measurement systems subject to legal metrology, primarily due to its open source philosophy and other special advantages.
evolutions

The outstanding advantages of measuring systems exploiting an OS are first and foremost the facilitated access to rather complex processes such as, for example, different data transmission techniques. A second point is the immense processor and graphic power which, together with the large memory available, allows the design of very comfortable and convenient graphical user-interfaces. The multi-user access to one particular system and the interconnection of distributed sensors or different measurement units is also a very important advantage.

And not to be forgotten, marketing has to be considered, since highly flexible systems will be sold much more easily.

The aforementioned advantages underline that PCs are designed for flexibility, so that - with the support of the manufacturer or programmer - the user can determine and modify software-controlled functions without too much effort.

options or the number of dependencies (depth) with an increasing number of distribution. The correlation of the evolution of SLOC and the configuration options was ~ 0.99, which is very high, as can be seen in Figure 2.

The results for all the metrics extracted from the Linux kernel configuration scripts from release 2.6.0 to release 2.6.28 indicated that the complexity of the code for the configuration options has increased consistently. This implies that the Linux kernel development team should have difficulties in maintaining these configuration options, especially when dealing with configuration options with several dependencies, or with a long chain of dependencies.

With these findings, it was considered that there may be severe inconsistencies with the configuration options that just have not been found yet, because not many developers or users have fine-tuned all of the more than 9000 configuration options in the Linux kernel. The Linux kernel configuration has reached a size and depth and therefore a level of complexity that is no longer feasible to handle and maintain without effective tools.

5 General advantages of embedded solutions vs. legal metrology regulations

The outstanding advantages of measuring systems exploiting an OS are first and foremost the facilitated access to rather complex processes such as, for example, different data transmission techniques. A second point is the immense processor and graphic power which, together with the large memory available, allows the design of very comfortable and convenient graphical user-interfaces. The multi-user access to one particular system and the interconnection of distributed sensors or different measurement units is also a very important advantage.

And not to be forgotten, marketing has to be considered, since highly flexible systems will be sold much more easily.

The aforementioned advantages underline that PCs are designed for flexibility, so that - with the support of the manufacturer or programmer - the user can determine and modify software-controlled functions without too much effort.
In contrast to this flexibility, however, definitely stands the concept of type approval where certain characteristics and/or components are fixed so that they cannot be modified without the official consent of the approving body, because all processes or techniques which impact on the acquisition, processing, transmission and visualization of the measurement value have to be - in principle - under legal control.

Strict attempts were initially made to apply these principles to PCs as well, with the result that the newly gained flexibility of PC-based measuring systems was almost completely lost for applications in legal metrology. It is obvious that an area of eternal conflicts will develop if the antagonism described above cannot be resolved. A sealed PC would be a regression, even an anachronism. It is obvious that new solutions and correspondingly extended guidance are necessary.

To answer such pressing questions requires a sound knowledge of the software, software engineering techniques, and the OS, which is seldom the case even for the OS developers as we have seen above. Exemplary challenges type approval authorities will face are [23]:

(i) How to prevent the activation of non-authorized parts or functions of the software.

(ii) What possibilities exist to access “protected” data or commands through the operating system?

(iii) Is it possible to capture data from peripherals and sensors without using the software of the instrument?

(iv) Is it possible to install different, alternative software on the same hard disk, i.e. software that could use the same user interface and which could easily be confused with the approved original software?

6 Challenges of operating systems

The use of operating systems is often challenging for metrologists and even for system developers. Both operating systems’ philosophies mentioned above comprise highly elaborate code which requires profound knowledge to be comprehensible and applicable. Hence an in-depth code validation might fail and might not be applicable, even if the code is open source as is the case for Linux-based systems. Even well-educated and experienced developers or type approval engineers might be confused by the interconnectivity these systems provide. Therefore an observation of the operating system must be carried out from an elevated level. The level of higher functions is proposed here to check for conformity with valid requirements.

As mentioned in [12], not many developers or users have fine-tuned every one of the more than 9000 configuration options in the Linux kernel, or even know all the interdependencies.

For instance, the same functionality which should be blocked for metrological/legal reasons, could be made accessible by a completely different function. Though blocked in one way, it can be circumvented by another application or configuration option, e.g. a blocked USB interface is activated by a sound driver.

To this end, the ensemble of functions, i.e. packages, offered by the individual Linux distribution must also be open source.

A further question is whether the relevant security functions of the kernel are exploited to an adequate extent or not.

If the measuring instrument provides access to the internet, the kernel cannot be regarded as a fixed structure, since the “kernel is alive”. Updates of the kernel’s functionality are available daily via the internet which on the one hand ensures the kernel’s tolerance against current threats, but on the other hand, changes its configuration, which might have an impact on the acquisition, processing and visualization of measured data. To this end, the update functionality must be inhibited by choosing an appropriate increased risk class. A risk class is defined by the combination of the appropriate levels required for software protection, software examination and software conformity [10].

As with any operating system, there are risks that a user might misuse it by injecting a code into the kernel which intercepts process calls and alters them, thereby changing the values produced by the measurement system. This scenario can almost not be prevented programatically, as the implemented security measures themselves are a target for malpractice.

The prevention of the misuse of the OS can be basically obtained by the physical sealing of i) the computer hardware and ii) the console access, preventing user access to any component. If a software component under legal protection, despite the sealing, is altered, it will be revealed by the software identification match between the version number and a qualified checksum, which is common practice recommended by the guides mentioned earlier. However, these hardware solutions either suffer from a limited flexibility or can be circumvented.

All of these exemplary scenarios have to be addressed to keep the system sound in the metrologist’s view, i.e. to prevent accidental or unintentional changes and intentional changes. To provide help to secure an IT-system in legal metrology, requires the same basic approaches as to harden conventional IT-systems against manipulations. For the latter guidance is offered by several national institutions, e.g. the Federal Office for Information Security [24] (BSI), Germany or the National Security Agency [25] (NSA), USA.
7 The Common Criteria approach for security of IT components

In the last decade, an international standard for IT security, the so-called Common Criteria for Information Technology Security Evaluation (CC) [7] has been developed. It provides guidance for security evaluation by laying down generic requirements for major security functionalities of IT products and assurance measures to be applied to these functionalities. The means to adapt the generic requirements to particular application areas are the so-called Protection Profiles, which are pre-designed in the CC standard.

While the protection profile is a means to express the security requirements in a comparable way, another instrument of the CC standard, the so-called Security Target, is a means to express the security function implemented in a particular product. By matching security targets with protection profiles, the evaluation of compliance with requirements is supported.

The big advantage of the CC approach is the comparability of evaluation results independently achieved by security evaluation bodies, besides the reference to the latest state-of-the-art of IT security. An aspect of special importance is that CC enforces the definition and clear description of assumptions and environmental conditions, under which a protection profile and a security target have been set up. Comparability is obviously only given under the same assumptions and conditions.

The approach is applicable to products that may be implemented in hardware, firmware, application software or any combination of them. The evaluation process establishes a level of confidence that the security functionalities of IT products under evaluation meet the requirements. The CC is therefore useful as a guide for the development and evaluation of IT products, even for their procurement. The CC is intentionally flexible, enabling a range of evaluation methods to be applied to a broad variety of IT products.

Though the approach of CC is very valuable and the only systematic and comprehensive one available, it is not a scheme that can simply be applied by the developer of a system quasi as an add-on to the normal specification. The levels of definition of the security functions and the protection profiles differ considerably from those a programmer needs. Especially for already existing complex systems like an operating system, it is necessary to interpret the security means implemented with regard to the security functions required by the protection profile.

Several evaluations of operating systems have been performed (Windows 2000 to Windows Vista, SuSE Linux, Red Hat Linux [26]) by the National Information Assurance Partnership (NIAP) [27]. Because of the complexity of the evaluation object, this was only possible by means of a cooperation of the evaluator with the manufacturer of the operating system.

To date, a successful application of CC-based methods in the legal metrology area is not known. This is, on the one hand, not fully explainable since the IT related requirements, as outlined above, are well structured in guides such as [8] and [10] and, therefore, well prepared for a revision in terms of a protection profile. On the other hand, this is comprehensible since the way of thinking and the terminology used in metrology and IT security are quite different. It will be a challenge in the near future to bring these two worlds together for the significant benefit of the IT security of measuring instruments.

Basically, the development process of a protection profile starts with an analysis of the threats the target of evaluation is exposed to. This comprises the identification of what is threatened as well as who or what is doing the threatening. Concerning a measuring instrument subject to legal control, matters that are threatened can be summarized as follows:

(A) The falsification of measurement values or the assignment of a measurement value to a wrong measurement.

(B) Wrong measurement functions or parameters of the measuring instrument.

(C) Inappropriate or missing protection means of the measuring instrument, implying that further threats such as A) or B) are facilitated.

In a standard analysis of who or what is the origin of a threat, roles are defined that are performed by persons. It is presumed that some intend to perform threats (A) to (C). A protection profile therefore does not require the same protection means for each role.

In countries where metrology is legally controlled, the roles and threats to be assumed are defined by law, to some extent. In general only the surveillance officer and no other person has privileges. When a measuring instrument is in use, it has to be protected against threats by anyone else.

8 Protection means for a measuring instrument with an embedded operating system

As discussed above, due to miniaturization, future measuring instruments will more and more often be equipped with embedded processors and additional components forming a circuit equivalent to that of a PC.
On such circuits it is possible to implement an operating system designed for personal computers.

In the following an approach is discussed that indirectly takes advantages of a CC-based security evaluation for a Linux distribution Red Hat RHEL 5. This distribution has received a Common Criteria Certificate EAL 4+ by the US authority NIAP [27], [28]. This certificate is only valid if the operating system is configured correctly according to strict rules. These configuration rules have been edited by another agency, NSA [25], [29].

The configuration rules are written for a desktop computer or a server. In the following chapter, modifications of the configuration rules [29] are discussed, that take into consideration the software requirements for legal metrology defined in OIML D 31 [8] and adapt the configuration rules to these requirements.

OIML D 31 [8] gives examples of configurations with PCs by focusing on the variant with a computer fully embedded into the measuring instrument. For this case a description will be given of how the protection measures offered by the operating system are used to repel the threats and possible attacks introduced above.

Up-to-date multi-tasking and multi-user operating systems support the protecting of programs and data domains as well as the functions of the operating system against inadmissible impacts by users or attackers. However, the integrated mechanisms only work if the operating system is configured correctly. Because of its complexity - as discussed above - this is not or not easily possible in each case.

The secure configuration of a computer as part of a measuring instrument differs from that of a part of a desktop in some aspects. Due to the requirements of legal metrology, measurement data, parameters, and measuring functions of a measuring instrument in use shall not be changeable, at least not without leaving a trace.

The idea of the security of operating systems is based upon the role of a system administrator who must be regarded as trustworthy. He is able to access all security parameters making him responsible for the security of the whole system. For a measuring instrument in use such a role is not planned in the regulations: the legally relevant provisions for communication via network, and restrictions for the plug & play mechanism.

8.1 Protection of the kernel

The heart of the operating system is a “kernel” (see Figure 1). As for Linux it consists of fixed parts, additional modules (kernel extensions) and device drivers that accomplish adaptation to the hardware platform and are loaded automatically if necessary. The kernel has its own domain in the virtual main memory, the so-called ’kernel space’. The kernel programs, active modules as well as the kernel data reside in this domain during runtime. The kernel space has to be protected against inadmissible access (threat (C)) and should not be changeable.

As the kernel performs the memory management during runtime and the processor runs with the highest hardware protection level, the kernel can supervise that there is no violation of the kernel space on its own.

The program code of the kernel and of the kernel modules is stored on mass storage devices. These are protected against falsification by means described in section 8.4, but only as long as the operating system is running. Conventional means, e.g. sealing the housing or hard disks, have to be used to ensure that the mass storage device cannot be exchanged or that protection is not circumvented by booting from another device instead of the regular one.

In the following the procedures at type examination, when putting on the market and in case of supervision, will be described.

It is impossible to analyze the kernel in the framework of a type examination of a measuring instrument. The following procedure is assumed to be an appropriate compromise between security and expenditure: The manufacturer is asked to use the kernel of a freely available Linux distribution but omitting all unnecessary modules and parts. Concerning freely available distributions, it is assumed that they do not contain inadmissible components that affect security. The comparison of the kernel software implemented in the measuring instrument with that of the reference distribution is relatively simple. Often the manufacturer has to enhance the minimum system to meet the needs of the customer. These additional parts have to be documented for the examination.
The seal and use the password. Afterwards the security steps can be carried out anew.

Another option to inhibit inadmissible access to the system is to generate the password automatically when the operating system is started up for the first time. The password is written nowhere and nobody will be able to access the administrator account. If a failure occurs, the operating system has to be re-installed from scratch.

8.4 Protection of files, especially of configuration files

Files on a Linux file system are provided with a differentiated permission system for reading, writing and executing. These permissions can be determined separately for each file, its owner, a group of users, or all users. Files can, e.g., only be used by one account and some cannot even be read by other accounts. This has to be considered especially for legally relevant files. In combination with the secure accounts described above, threats (A), (B), and (C) can be repelled.

Before starting the examination, the settings of at least three selected files (configuration files, system files and legally relevant files) are fixed. Afterwards the corresponding files are read from the measuring system and the permissions are compared with the nominal value.

8.5 Protection against inadmissible services, firewall, protective network interface

A big advantage of PC components compared to solutions with micro-controllers is the perfect ready-to-use support of networking. For many engineers this is the most important argument for integrating such a system into the measuring instrument. The network interface is, however, the entrance for attackers. The operating system has to be configured so that this danger is minimized. Securing steps consist of adjusting certain kernel parameters and of stopping or starting only services that are absolutely necessary. Services are programs that run in the background and react to requests from the network.

When examining the kernel, these parameters are checked and the initiating procedure for services will be analyzed.

8.6 Restrictions for plug & play automatism

For the user of the system, one very comfortable property of the operating system is the plug & play
mechanism. Concerning security, these interfaces are problematic as even booting from external devices and the circumvention of important protection means is possible if the configuration is too weak. In any case, the option of external booting has to be blocked for measuring instruments. Basically, Universal Serial Bus (USB) interfaces should not be accessible. Single USB plug & play functions could be accepted, but their intensive and usually difficult examination is necessary.

When examining a USB interface, it is checked whether the boot option is inhibited, automatic loading of drivers is de-activated and only those drivers are loaded by a script or program that are necessary.

9 Automatic examination of the configuration

The measures for repelling the threats described above comprises many settings. To set them manually is error-prone and time consuming. This procedure has, therefore, been automated. This concept is well established, such hardening programs "lock down" an operating system by proactively configuring the system for increased security and decreasing its susceptibility to compromise. They can also assess a system's current state of hardening, granularly reporting on each of the security settings with which it works. A sophisticated tool in this category is "Bastille" [30]. However, this kind of tool is constructed for application on desktops and servers. They are not appropriate for use in a measuring instrument.

The tool developed by the PTB generates information about the security-relevant settings of the measuring instrument as described in sections 8.1 to 8.4. This information has to be checked and evaluated by the examiner.

Two applications are possible: Before type examination the tool is given to the manufacturer. The tool supports him in applying the right security settings. During type examination, the tool is executed again and the result is evaluated. If the security settings are satisfactory they are archived and a checksum is generated.

For the supervision and checking of an instrument in use, the tool can be applied again. The output has to be compared with the reference or its checksum, respectively. Even without profound knowledge of operating systems, an efficient examination of the security of measuring instruments in the field is possible.

10 Conclusion

To circumvent the pitfalls introduced by the complexity of operating systems embedded in measuring instruments subject to legal verification, we proposed an initial test paradigm. An application was exemplarily designed for measuring instruments with an embedded Linux-based operating system. The paradigm was derived from an existing protection profile which received a formal certification according to the Common Criteria Standard.

Based on this approach and on a basic threat scenario of measuring instruments subject to legal verification, protection means are proposed for the described configuration of measuring instruments. To this end, the compliance of a particular security solution implemented for an operating system with the security requirements established can be achieved. It is expected that the procedure described in this paper can be adapted to other variants of operating systems.

First applications have shown the feasibility of the paradigm. However, these first application tests also have shown the need for further developments and investigations in other fields of applications in order to reach the level of an internationally accepted standard procedure.

Acknowledgements

The authors are grateful to Norbert Greif, Heike Schrepf and Samuel Eickelberg for their helpful discussion and critical remarks in the preparation of this paper.

References