15th IMEKO TC10 Workshop on Technical Diagnostics
Technical Diagnostics in Cyber-Physical Era
Budapest, Hungary, June 6-7, 2017

Security Concepts for Software in Measuring Instruments

Patrick Scholz[1], Daniel Peters[1], Florian Thiel[1]

[1] *Physikalisch-Technische Bundesanstalt (PTB), Abbestraße 2-12, 10587 Berlin, Germany*
E-mail: {patrick.scholz, daniel.peters, florian.thiel}@ptb.de

***Abstract* – In the age of the Internet of Things and Industry 4.0, more and more embedded systems are connected through open networks, which also concerns measuring instruments under legal control (e.g. smart meters). Therefore, cyber-security for measuring instruments is becoming increasingly important. In this paper, possibilities to design secure measuring software running on general-purpose operating systems are analyzed according to legal requirements set up by European Directives, e.g., the Measuring Instruments Directive (2014/32/EU), which define the mandatory security level. Concrete security concepts for Windows (Mandatory Integrity Control) and Linux (SELinux and AppArmor) are described.**

I. INTRODUCTION

Embedded systems play an increasingly important role in industry and economy and, in Europe, currently have an annual growth rate of 12% [1]. Software security is becoming a major focus of research, because today's embedded systems are often connected to insecure networks such as the Internet. Nevertheless, software architectures are becoming more and more complex, making it increasingly difficult to design secure systems.

Focusing on measuring instruments in Europe, in Germany alone, more than 100 million legally relevant measuring instruments are in use, mainly electricity, gas, water and heat meters which in total lead to an annual gain between €104 billion and €157 billion [2]. A malfunction or manipulation of these measuring devices could cause considerable damage. Therefore, it is particularly important to focus on the security of such systems and their software.

Manufacturers of measuring instruments prefer to use general-purpose operating systems (GPOSs) such as Windows or Linux due to the comfortable software infrastructure and the broad availability of device drivers. These operating systems include several million source lines of code (SLOC). Studies have revealed that on average a software bug can be found in every 2300 lines of code [3]. The presence of only one bug can be sufficient for an attacker to gain unauthorized access, which leads to the assumption, that measuring devices running on GPOSs are under great risk of manipulation.

The main focus of the basic requirements for measuring instruments under legal control is the protection of consumers and the correctness of the measurements. In Europe that trust is ensured by institutions, so-called *notified bodies*, which devote themselves to the validation of measuring instruments before commissioning and the assessment of conformity. The tests in laboratories primarily focus on the hardware, such as the physical sensors. Testing the software proves to be more difficult, since operating systems and their additional software have too many lines of code and are often not open source. To support this challenge, GPOSs enhance their systems with security concepts. Therefore, the analysis this paper provides, focuses on how *"Mandatory Integrity Control"* (Windows), *SELinux* and *AppArmor* (Linux) can be used to fulfill the requirements in legal metrology.

II. LEGAL METROLOGY

In order to help consumers to rely on the correctness of measurement results without having to check them themselves, legal regulations at national and international level are necessary. Legal metrology thus creates the prerequisite for producing high-quality products and contributes significantly to the functioning of an economy. It is estimated that in industrialized countries, 4% to 6% of the gross national income is settled by measuring instruments and the related measurements [2].

At the international level, the "Organisation Internationale de Métrologie Légale" (OIML) is an intergovernmental organization that has set up a world-wide technical set of proposals to help with the harmonization of legal metrology. For example, software requirements for measuring instruments are formulated in the *OIML D 31* [4] guide.

In Europe, the *Measuring Instruments Directive* (MID) [5] of the European Union formulates concrete legal requirements. In addition, *WELMEC* is the committee at the European level in legal metrology, which includes the member states of the EU and EFTA. The guidelines for the conformity assessment of measuring instruments, the so-called WELMEC Guides, help the manufacturers and the notified bodies to build or check measuring instruments, respectively. Here, *WELMEC 7.2* [6] is the

Software Guide for measuring instruments. It states that the software-related requirements of the MID are fulfilled if the requirements of the guide are upheld [6].

III. BACKGROUND

In legal metrology, the risks of manipulating measuring instruments and the theft of sensitive data are increasing, because of the integration of general-purpose operating systems such as Windows or Linux, and because many devices are connected to the Internet. So nowadays, security is becoming an important focus.

In general, **security** is the ability of a component to protect resources for which it announces protection responsibility, and the component's security policies are its security-enforcing properties and requirements. Generally, security policies try to enforce three points:

- **Confidentiality** ensures no unauthorized disclosure of information;
- **Integrity** prevents modifications or corruptions of a resource without authorization;
- **Availability** ensures that authorized users have access to resources whenever needed.

By using access control strategies and a secure boot mechanism, it is possible to ensure adequate security (upholding the three points from above) for connected devices. In addition, separation of functionalities in IT systems can provide even more security. For example, a further concept for high security IT systems is the "**Multiple Independent Levels of Security**" (**MILS**). *MILS* is a high-assurance security architecture based on the concepts of separation and controlled information flow. The foundation of the system is a small kernel implementing a limited set of critical functional security policies [7].

A. Legal Requirements

The *WELMEC 7.2* Software Guide tries to break down the requirements for legal metrology software of the *MID* to special technical examples and recommendations. One important point is verifying measuring instruments in commission. Validating the software identification ensures that software was not switched or manipulated. Out of the *MID* and the assessment of hazards, the *WELMEC 7.2* Software Guide defines six risk classes from A to F, evaluating the need for **software protection**, **software examination** and **software conformity**. The risk classes are ascending (from A to F) in their demand for security. For our purposes, the best way to start is to construct a system for risk Class F, because it conforms to all requirements of the other classes. Hence, the solution provided here can be downscaled to lower risk classes.

Furthermore the *WELMEC 7.2* clarifies what **legally relevant parts are**. According to *WELMEC 7.2*, all modules are legally relevant that make a contribution to or influence measurement results. These modules facilitate auxiliary functions, like ***displaying data, protecting data, saving data, identifying the software, executing downloads, transferring data and checking received or stored data.*** Based on these seven points, we construct our security concepts in the next section.

Additionally, the W7.2 differentiates between measuring instruments that are built solely for the measuring purpose and the ones that run universal software. The two classes are called P and U. Normally one can say, if a measuring instrument has an operating system installed, it is a U type, else it is situated in the P class. For both classes, four subclasses are defined which deal with following IT functions:

- L: long-term storage of measurement data,
- T: transmission of measurement data,
- D: software download,
- S: software separation.

To be more specific, the WELMEC 7.2 Software Guide tries to break down the requirements for legal metrology software of the MID to technical examples and recommendations, looking more closely at the four points from above. The important MID software requirements, we spotted, are:

1. The measuring instrument must guarantee reproducibility, i.e., measurement results of the same thing must yield the same result, even if handled by different users. This implies that a measurement result should not depend on the user/consumer employing the instrument. From the software point of view, different processes with varying access rights performing the same measurement should yield the same result.
2. A measuring instrument shall be designed to reduce as far as possible the effect of a defect (bug) that would lead to an inaccurate measurement result, unless the presence of such a defect is obvious.
3. A measuring instrument shall have no feature to facilitate fraudulent use, and possibilities for unintentional misuse shall be minimal.
4. Software identication shall be easily provided by the measuring instrument. Additionally, evidence of an intervention shall be logged to verify measuring instruments in commission.
5. If a measuring instrument has associated software which provides other functions besides the measuring function, the software that is critical for the measurement purpose shall be identifiable.

6. Measurement data and software that is critical for measurement characteristics and has metrologically important parameters stored or transmitted, shall be adequately protected against accidental or intentional corruption.
7. For utility measuring instruments the display of the total quantity supplied or the displays from which the total supplied quantity can be derived, whole or partial reference to which is the basis for payment, shall not be able to be reset during use.
8. The indication of any result shall be clear and unambiguous, i.e., easy reading of the presented result shall be permitted under normal conditions of use. Still, additional indications may be shown provided they cannot be confused with the metrologically controlled indications.
9. A durable proof of the measurement result and the information to identify the transaction shall be available.

IV. SECURITY CONCEPTS

A concept to enhance the security in IT systems is by the use of **access control strategies**. These determine with which access type (e.g., *read*, *write*, or *execute*) a *subject* (e.g., user, process, or device) may have access to an *object* (e.g., file, table, or subject). The access rights are controlled and enforced by a **reference monitor** (**RM**).

One of these access control strategies is called **Discretionary Access Control** (**DAC**). Here, access rights are defined only by the identity of a subject. Another one is the **Mandatory Access Control** (**MAC**), where an access decision is made by the *properties* of subjects and objects, and additionally by well-defined *rules*. In legal metrology, it is best to enhance the DAC with the MAC, because confidence in the correctness of a measurement and the integrity of measurement data is of vital importance. Additionally, the **Role Based Access Control** (**RBAC**) strategy exists, which does not give users direct access to objects. An access decision is made based on *roles* assigned to a *user* and *groups* that are accessible by these roles. So permissions are directly linked to tasks by assigning roles.

A. SELinux

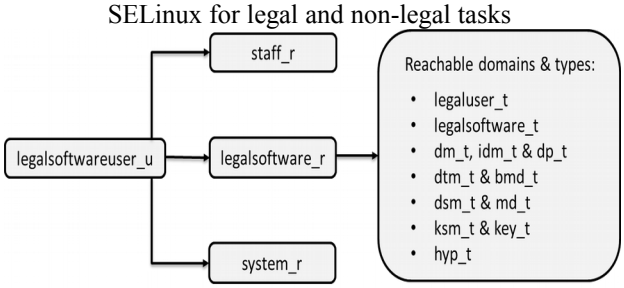
In Linux, the *Discretionary Access Control* (DAC) is used by default. Still, *Mandatory Access Control* (MAC) and *Role Based Access Control* (RBAC) strategies can additionally be used. One example is the **Security-Enhanced Linux** (**SELinux**), a Linux kernel module.

SELinux is based on an advanced security kernel called **Flask**, which was developed by a research group at the university of Utah named Flux [8]. In the Flask framework, a **Security Server**, which contains the *security policies*, communicates via interfaces to a so-called **Object Manager**. At each request of a subject to an object the Object Manger checks the policies, with the help of the Security Server, to grant or deny access.

In SELinux each *user* at any moment is assigned exactly one *role*. If allowed, the user can herself switch into another role. Each role has access to a set of *domains* and *types*. With the construct of *user*, *roles* and *domains* (or *types)* well-defined security policies can be constructed.

The **Strict Policy**, which is available in SELinux, is well suited to construct a secure framework, because it upholds the principle of least privilege, which states that every component should only have the access permissions it really must have to function, no more. Hence, each subject and each object are controlled by their own domain and have just the access rights they need. In Figure 1, one can see our framework of user, roles and domains based on the legal requirements listed in Section III.

Figure 1: Framework of the user, roles and types in SELinux for legal and non-legal tasks



As can be seen, we created a user called *legalsoftwareuser_u* in SELinux that can reach three roles: *staff_r*, *system_r* and *legalsoftware_r*. With the roles *staff_r* and *system_r* the user can have access to non-legal objects, which are not necessary for the measuring purpose. In our construct, only the *legalsoftwareuser_u* user has access to the role *legalsoftware_r* and only in this role the legally relevant domains can be used, which represent software processes that are needed for measuring. The legally relevant software parts can interact with each other in the way shown in Figure 2, and explained in the text above the figure. Users without the *legalsoftware_r* role cannot influence the legally relevant tasks, because they have no access to them.

B. AppArmor

A second way to use *Mandatory Access Control* (MAC) in Linux is the **ApplicationArmor** (**AppArmor**) framework. AppArmor is like SELinux a Linux kernel extension, which does not use *Role Based Access Control* (RBAC) strategies.

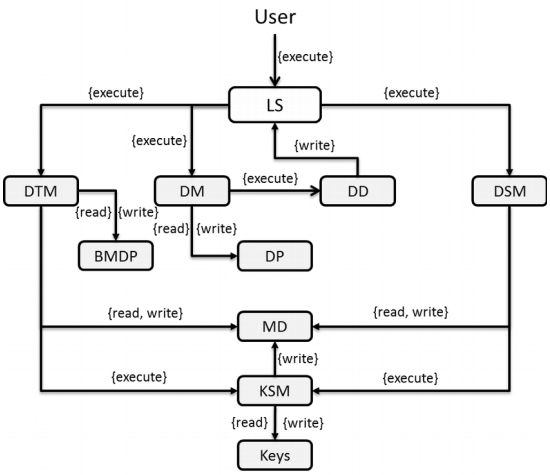
In AppArmor *profiles* [9] and individual *security rules* are used to implement the MAC and to determine

whether the access of a subject to an object is allowed or not. Each profile is created for exactly one process. In AppArmor only processes are supervised that have such profiles. Because only newly started processes are monitored, AppArmor must be started at the beginning. In general, AppArmor divides processes into trustworthy and untrustworthy, and just guards the untrustworthy, because it is assumed that the others are not a danger to the IT system.

AppArmor can operate with much less rules then SELinux, making it easier to use. A disadvantage is that rules in AppArmor are based on absolute file-names. By renaming a file, the protection of AppArmor is subverted. It should be noted that AppArmor and SELinux cannot be used at the same time. If both are modules are compiled into the Linux kernel, one has to be deactivated. The framework we use to fulfill the legal requirements of the *WELMEC 7.2 Software Guide* is shown in Figure 2.

Here, only a user (**User**), which is allowed to execute the legal relevant software, has access to the legally relevant software (**LS**) and through this, access to the legally relevant software modules, like the Download Manager (**DM**), the Data Transfer Module (**DTM**) and the Data Storage Manager (**DSM**). The DTM and the DSM are responsible for the transmission of measurement data and for their long-term storage. Both modules have access to the measurement data (**MD**) and can use a key & signature manager (**KSM**) for de- and encryption of data. In the broken measurement data protocol (**BMDP**), errors, like damaged measurement data can be recorded. The DM is used for the transfer of downloaded data (**DD**), the subsequent verification procedure and the following installation. To verify this data, the DM uses hash values to ensure authentication. Also, a download protocol (**DP**) is being created for each download process.

Figure 2: User, legal software parts and their access permissions in SELinux and AppArmor



C. Mandatory Integrity Control

*S*ince Windows Vista the MAC can be used by a framework called **Mandatory Integrity Control** (**MIC**). It can be used additionally to the DAC, which Windows implements by default, similar to Linux. Hereby, an access allowed from the DAC can still be prevented by the MIC.

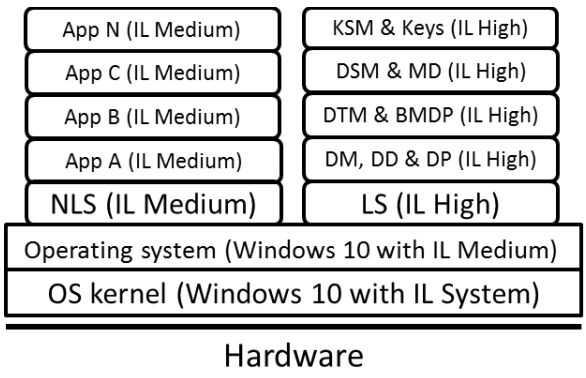
In MIC each object is considered a securable object and is classified into one of five classifications, called the **Integrity Level** (**IL**) that is part of the **Security ID** (**SID**) stored in a **Mandatory Label** of an object [10]. These five classifications are: *Untrusted*, *Low*, *Medium*, *High* and *System*. With these Integrity Levels and *Mandatory Policies* an access decision is made. MIC enforces system-wide no-up policies: *no-write up*, *no-read up* and *no-execute up,* meaning that a user with a lower IL is not allowed to access an object with a higher level.

The IL *System* is preserved for system-related processes such as kernel processes. No user can acquire this IL [10]. *High* is, e.g., for the administration of a system, but can be acquired from normal processes if needed. Medium is the default setting of most processes. The IL *Untrusted* is for unidentified processes and *Low* for low-trusted processes with higher attack-risk.

The Mandatory Label of an object which includes the SIDs, can be found in the **Security Descriptor**. The SIDs of subjects are stored in an **Access Token** alongside a set of privileges and a list of group memberships.

Figure 3 gives an overview of the ILs we use for our software framework, seen in Figure 2. Here, the legally-non relevant software (**NLS**), which is not used for the measurement, is classified as IL *Medium*. The whole legal relevant modules, that are used for measuring processes, are classified with IL *High*. Because of the no-up policy in MIC, a manipulation of the legal relevant modules is so not possible.

Figure 3: Integrity Levels with MIC for measuring instruments in legal metrology



V. PRACTICAL EVALUATION

With the minimal implementation principle in mind, we constructed a minimal Linux with *SELinux* support, to see how much space we need. For demonstration purposes we used *QEMU*, which is a generic and open source machine emulator and virtualizer.

We compiled the Linux kernel (version 4.11.0-rc4) for x86 systems, which most general purpose computers (based on Intel chips) can execute. Additionally we used *Buildroot* to create our root filesystem. *Buildroot* is a tool that helps to generate embedded Linux systems. With this tool it is also simple to cross-compile everything, constructing for example also binaries for *ARM* platforms. It also supports the compilation of *BusyBox* that is often used in embedded systems to replace more than 300 common Linux commands, e.g., *ls, sh, rmdir*, etc. into one executable. In the root filesystem of our Linux system, we added many additional packages to the core *BusyBox*-based system: the bash shell, a GUI (X.org display server, matchbox window manager), mdev as device manager, and of course the necessary infrastructure to activate SELinux (Linux kernel modules, SELinux support options in Busybox). Afterwards, we downloaded example policies (from [11]) and added them to our root filesystem. Finally, we activated SELinux by adding *selinux=1* to the kernel commandline and created a test file */etc/selinux/config* with following parameters: *SELINUX=permissive* and *SELINUXTYPE=targeted*, to set permissive mode, which does not automatically block unauthorised access, but just logs it for our demonstration purposes. By setting *SELINUX=enforce*, access can be blocked when, for example, the device is in real operation mode. The resulting root filesystem including the kernel had a total size of 45.8 MB. In our opinion, the ground framework of around 45 MB is no restriction for measuring instruments, because small boards with gigabytes of RAM and NAND flash storage, are common nowadays and have costs of under 100$.

VI. CONCLUSION

In this paper, an overview of secure software construction for general purpose operating systems, i.e., Linux and Windows is being given. Concretely, measuring instruments under legal control are analysed, which have become powerful devices running under such operating systems. The frameworks presented here, are constructed to fulfil the requirements of legal metrology such as the Measuring Instruments Directive MID 2014/32/EU and the WELMEC 7.2 Software Guide. Hereby, it is most important to find secure methods for the following processes: data transfer, controlling of received and stored data, performing downloads, authentication of software, data storage, data protection and displaying of data.

The security methods, shown in this paper, are SELinux, AppArmor and MIC, which are based on Mandatory Access Control and/or Role Based Access Control strategies (*MAC* and *RBAC*). With these methods well-defined rules for roles, classifications for subjects and objects, and access rights are being defined for legally relevant software that fulfil measuring activities. By using an architecture, build by single components, and by creating every component with least privilege rights in mind, the described rules fulfil the requirements of legal metrology directives. Our framework ensures that non-legal processes, which run on measuring instruments, have no effect on the legally-relevant processes. The goal is to make current measuring instruments more resistant against software vulnerabilities and attacks from open networks like the Internet.

REFERENCES

[1] IDC France (L'Institut Du Comportement), “Special Study - Final Study Report: Design of Future Embedded Systems (SMART 2009/0063)”, IDC - Analyze the Future (International Data Corporation), Final interim Study Report 2, Deliverable D4, April 2012.

[2] N. Leffler, F. Thiel, “Im Geschäftsverkehr das richtige Maß - Das neue Mess- und Eichgesetz”, Monatsbericht, 11-2013.

[3] B. Chelf, “Measuring software quality - A Study of Open Source Software”, Chief Technology Officer, San Francisco, CA, USA, 2011.

[4] OIML D 31, “General requirements for software controlled measuring instruments”, Organisation Internationale de Métrologie Légale, Edition 2008 (E).

[5] Official Journal of the European Union, “Directive 2014/32/EU of the Eurpean Parliament and of the Council”, L 96/149, European Commission: Brussels, Belgium, 2014.

[6] WELMEC (European Cooperation in Legal Metrology), “Softwareleitfaden (Europäische Messgeräterichtlinie 2014/32/EU)”, WELMEC 7.2, 2015.

[7] D. Kleidermacher, M. Kleidermacher, “Embedded Systems Security - Practical methods for safe and secure software and systems development”, Newnes - Imprint of Elsevier, 2012, pp. 27-31.

[8] R. Spencer, S. Smalley, P. Loscocco, M. Hibler, D. Andersen, J. Lepreau, “The Flask Security Architecture: System Support for Diverse Security Policies”, Secure Computing Corporation, National Security Agency, University of Utah, USA, August 1999.

[9] R. Spenneberg, “SELinux & AppArmor - Mandatory Access Control für Linux einsetzen und verwalten”, 2008, Addison-Wesley Verlag.

[10] F. Kuhn, “Mandatory Integrity Control”, ERNW Newaletter 17, Juli 2007.

[11] SELinux policies, accessed: 28.04.2017, link:

https://rpmfind.net/linux/rpm2html/search.php?query=selinux-policy-targeted.

[12] J. Alves-foss, W. S. Harrison, P. Oman, and C. Taylor, "The MILS architecture for high-assurance embedded systems", Journal of Embedded Systems, 2:239-247, 2006.

[13] J. Barr, "The Flask Security Architecture", In Computer Science 574, 2002.

[14] R. W. Beckwith, W. M. Vaneet, and L. MacLaren, "High Assurance Security/Safety for Deeply Embedded, Real-time Systems", Embedded Systems Conference, 2004.

[15] K. Doeornemann, and A. von Gernler, "Cybergateways for Securing Critical Infrastructures", In Proceedings of International ETG-Congress 2013, Symposium 1: Security in Critical Infrastructures Today, Berlin, Germany, 5-6 November 2013.

[16] D. Peters and F. Thiel, "Software in Measuring Instruments: Ways of Constructing Secure Systems", SENSOR 2016, Nuremberg; 05/2016.

[17] F. Thiel, U. Grottker, and D. Richter, "The challenge for legal metrology of operating systems embedded in measuring instruments", OIML Bull. 2011, 52, pp 5-14.