Bundesamt
für Sicherheit in der
Informationstechnik

ICS Security Compendium

Test recommendations and requirements for product suppliers of components

Version: 19.11.2014

Federal Office for Information Security
P.O.B. 20 03 63
D-53133 Bonn (Germany)
Phone: +49 22899 9582-0
E-mail: ics-sec@bsi.bund.de
Internet: https://www.bsi.bund.de

Table of contents

Tabellenverzeichnis

1. Introduction

This part of the ICS Security Compendium is written for product suppliers of ICS (Industrial Control System) components and covers both hardware and software components. The subject of security must be taken into account in the design and development of ICS components and plants/machines.

This document introduces assistance for testing the components and measures to avoid and identify vulnerabilities. A comprehensive description of the bases and threats of ICSs can be found in [BSI-ICS].

Further information on the subjects of threats, recommended measures and the realisation of security tests supplement the individual process steps.

1. Identifying assets
   Assets may be an individual component to be protected, a machine, a plant part or an overall plant. Assets also include data which is exchanged between them or stored on them, for example the required software or recipes. An entire distributed control system with the related business secrets and intellectual property can also be considered an asset.
   The selection and delimitation is carried out by the product supplier or integrator.
2. Analysing threats
   An overview of the most critical threats can be found in [CS-E-TOP10] and [BSI-ICS]. Due to the different framework conditions, a separate threat analysis is not replaced by this. Only the introduction to this subject should be simplified.
3. Determining relevant security objectives
   Security objectives within the meaning of this document can include
   - Confidentiality of data on devices (e.g. recipes) and during the transmission
   - Integrity of data on devices and during the transmission
   - Availability of components and reliability
   - Authenticity of data and instructions
4. Analysing and assessing threats
   An analysis and assessment of threats within the component(s) is performed by the product supplier. The integrator takes the application conditions specified by the product supplier into account. The asset owner is responsible for testing the security during subsequent operation according to [BSI-ICS] and according to the corresponding specifications by the integrator.
5. Assessing measures and their effectiveness
   Chapter 2 includes a collection of measures. When selecting the measures, they must be examined for their suitability and effectiveness for the respective component. An example of this is the use of anti-virus software. If a component does not provide adequate resources, this measure cannot be implemented in a reasonable manner. The same applies if virus signatures cannot be distributed at regular intervals. This would lead to a very bad detection rate. The measure would then not provide additional protection and could thus be omitted.
6. Selecting and implementing measures
   The selection and implementation of the measures within the component(s) is performed by the product supplier on the basis of the specifications of the asset owner and/or integrator. The asset owner's security objectives must be taken into account. It may therefore be the case that several measures may only be selected and implemented in collaboration with the integrator and asset owner.
   The product supplier, however, should take into account a preselection of effective and reasonable measures during the development process to be able to support the integrators and asset owners better at a later point in time.

7. Performing the security test of the components
   To test the effectiveness of the measures taken, questions concerning security tests are described in chapter 3. They are intended to enable the product supplier to integrate them into the processes starting with the development up until the system test.
8. Regular re-evaluation
   In the case of changes to the component or defined framework conditions, a test as to whether the measures taken are still adequate or need to be improved should be carried out at regular intervals.

When integrating the components, it must be ensured that the security mechanisms offered are activated and used. When designing new and further developing existing components, the subject of security should also be taken into account to be able to respond to new threats. Furthermore, the implementation of the defence-in-depth concept which specifies the layer-by-layer utilisation of further measures is recommended.

The document serves as an introduction to the subject of security. Many of the measures can also be found in other standards (e.g. ISO/IEC 62443, IT-Grundschutz etc.) or recommendations. This document focuses on a selection of important measures which should be started with. To firmly anchor the subject of "security" in the enterprise, using the standards is recommended. The presented measures are therefore also assigned to these standards to make switching to a procedure conforming to standards possible.

1.1. Security & safety

At the beginning, a distinction between the terms "safety" and "security" should be made. The term "safety" refers to the functional safety of the machine or plant. The term "security" describes the protection of IT-supported systems against deliberate or undesired errors. Safety systems must also be protected against such attacks.

1.2. Glossary

- Attack
  An attack is a deliberate form of threat, namely an undesired or unauthorised action performed with the goal of gaining advantages and/or causing damage to third parties. [BSI-Glossar]
- Asset
  Assets are tangibles and intangibles which might be threatened and are worthy of protection. [VDI2182-1]
- Authentication
  On the one hand, authentication refers to the submission of some type of verification of the communication partner's identity. On the other hand, authentication can also refer to the examination of this verification. [BSI-Glossar]
- Authorisation
  During authorisation, it is checked whether a person, an IT component, or an application is authorised to perform a specific action. [BSI-Glossar]
- Threats
  A threat in general terms is an event or condition which involves the risk of damage . The possible damage in this case relates to actual values, for example to financial assets, knowledge, objects or people’s health. [BSI-Glossar]

  A threat may also have a direct effect on an object as the result of a vulnerability. A threat therefore only becomes an imminent threat for an object when it is combined with a vulnerability. [BSI-Glossar]

- Security
  Security refers to a condition where risks existing as the result of threats and vulnerabilities during the use of information technology are limited to an acceptable level by suitable measures. Security is therefore also the condition in which confidentiality, integrity and availability of information and information technology are protected by appropriate measures. [BSI-Glossar]
- Risk analysis
  A risk analysis provides information on the probability of occurrence of a damaging event, and which negative consequences the damage would have. [BSI-Glossar]
- Robustness
  The ability of a system to respond to errors and unpredicted conditions so that no incorrect actions are performed. [ACATECH-1]
- Protection requirements
  Protection requirements describe which protection is adequate and appropriate to protect the processed information and the employed information technology. [BSI-Glossar]
- Vulnerability
  A vulnerability is a security-relevant error. Causes may include the design, the algorithms used, the implementation, the configuration, or the operation, as well as the organisation itself. A vulnerability may cause a threat to become effective and damage an organisation or a system. As a result of a vulnerability an object (an organisation or a system) is susceptible to threats. [BSI-Glossar]

2. Measures

The following measures are primarily aimed at product suppliers who must take them into account within the scope of development. It is a very generic listing of measures which relate to very general characteristics of the components.

It may be the case that certain measures cannot be implemented in the components. An example of this is the anti-virus software. There is often no anti-virus software available for the operating systems used in programmable logic controllers (PLCs) or human machine interfaces (HMIs). In these cases, the measure cannot be implemented. The product supplier should nevertheless consider whether there is an increased risk and take alternative measures if necessary.

The suggested measures are provided as general recommendations and are intended to illustrate the importance of the procedure described in the introduction. The focus is on the reduction of risks and providing assistance. Specific particularities of the components and threats resulting from them can only be identified by the product supplier and/or the integrator.

The measures refer to the software used in the components and its configuration. The characteristics of the hardware, for example regarding reliability, must be taken into consideration additionally even though an attack may also impair availability.

2.1. Organisation, development & planning of ICS components

In general, an information security management system (e.g. according to ISO/IEC 27001, IT-Grundschutz, ISO/IEC 62443-2-1 "Requirements for an IACS Security Management System") should be implemented in an enterprise. This serves to provide the enterprise with general protection against failures or damage. This includes in particular the protection of the development department against the outflow of information (e.g. outflow of design data) and manipulations of development data (e.g. integration of malicious functions). The focus must be on the secure distribution of the data (within the meaning of security).

In addition, this document takes up several aspects which should be taken into account during the planning and development process. The measures are used as introduction to the subject and should be extended successively by further requirements from the standards mentioned above. The focus is on the software components which have become a significant part of the development work in the meantime. This includes for example aspects regarding the product design, the use of libraries during the software development as well as the realisation of test phases (so-called "reviews") or security tests.

M 1 Introduction of a development cycle for secure software in components

A fundamental improvement of the security of a product can be achieved by establishing a security-oriented development cycle (so-called "secure software development life cycle"). There should be uniform and mandatory (according to the state of the art) specifications for the implementation, for example regarding the use of libraries during the software development as well as with respect to the realisation of test phases (so-called "reviews") and security tests.

Further information in this respect can be found, among other things, in the following documents:

- *BSI-Leitfaden zur Entwicklung sicherer Webanwendungen* [BSI guide for the development of secure web applications] [BSI-WEB] (the applicability of the recommendations is not only restricted to web applications)
- ISO/IEC 62443-4-1 "*Anforderungen an die Komponentenentwicklung*" [Requirements for the development of components] [62443-4-1]

In general, the next development step should only be taken if the pre-defined security requirements have been complied with verifiably.

From the mentioned standards, the following measures are of particular importance. They aim at avoiding error sources at an early stage and identification of frequently occurring errors in the development process.

M 2 Raising the awareness of developers and product managers
The awareness of developers and product managers in terms of security should be raised appropriately. Training programmes are especially suitable for this purpose. The possible attacks, their techniques and methods and the corresponding countermeasures should be addressed. "ICS TOP 10 Threats and Countermeasures" [CS-E-TOP10] and "CWE/SANS TOP 25 Most Dangerous Software Errors" [CWE-TOP25] [SANS-Top25] are ideal as a starting point.

M 3 Static code analysis
As part of the development process, static code analysis tools should be used. They support the developer and help to avoid errors. An overview of different tools can be found in [OWASP-1].

To ensure the implementation, the code should only be integrated into the version management if it complies with the static code analysis guidelines (so-called "gated check-in").

M 4 Robustness of the implementation
The protocol stacks and applications used should be implemented with corresponding robustness. The term "robustness" also refers to the proper processing of data which is non-compliant with the protocols. A robust application should process or discard non-compliant data in an organised form. The implementation is usually tested by means of a so-called "communication robustness test" using hardware-supported test tools.

Tests with respect to the robustness of implementations can be integrated directly as part of the build process. Only if this has been completed successfully should the code be included in the version management (see S 3).

If an error occurs while the component is operated, this error must be handled in such a way that a consistent and technically sound state of the component is guaranteed and data protection is also maintained.

A component is in an inconsistent state if it is switched to an unexpected state due to a vulnerability and data is therefore processed in an uncontrolled manner (e.g. no error message if data is stored unsuccessfully) (see [BSI-GS] S 4.395).

A detailed list of possible test scenarios for Ethernet-based components can be found in [ISA-EDSA]. For PROFINET, for example, the PROFIBUS Nutzerorganisation e.V. (PNO) has also prepared documents [PNO-1].

M 5 Identification of sensitive data and communication relations
When designing machines/plants, a protection requirements analysis should be carried out for the internal and particularly external communication relations. Critical communication relations and data, which must be protected, are to be identified. Depending on the results, measures to ensure secure communication must be implemented. For individual components (such as a PLC), this cannot be carried out by product supplier, since the function and the data processed depend on the use case and therefore cannot yet be determined.

Information on the procedure to be followed for the protection requirements analysis can be found in [BSI 100-3]. Depending on the assessments, measures should be implemented.

M 6 Ensuring availability
In the planning and implementation process, a reliable and robust possibility for the system recovery should be taken into account. An example of this is the procedure when replacing a defective component. This may also include the exchange of authentication data or other cryptographic keys to access communication services.

Availability should also be ensured if requirements cannot be complied with directly in crisis situations due to the security policy. For example, a local emergency shutdown may be carried out by an unauthorised person, but must be documented accordingly. The expiry of certificates must not lead to total failure either or must be prevented by means of organisational measures (timely renewal).

M 7 Security by design

When designing components, as many security functions as possible should be available. To reduce the associated complexity, they should be restricted to the necessary functions. On the basis of a risk analysis, product suppliers should determine to which threats the device is exposed. The results of measure M 5 should also be used as a basis.
This leads to a selection of functions and, if applicable, specifications which enable the integrator to build a secure machine/plant and, together with the asset owner, ensure secure operation.

M 8 Use of open-source software

Open-source software is available for many areas of application. This is software the source text of which is disclosed. "When using free software, the following technical aspects are particularly important for the BSI:

1. Warning messages regarding errors identified in the course of security tests can be published, because there is no non-disclosure agreement. The user can thus be informed of vulnerabilities quickly and take countermeasures.
2. It should always be possible to test the software for vulnerabilities. This may be a criterion when using software. It is trust versus knowledge." [BSI-FLOSS]

If open-source components are to be used, the authenticity of the program should be verified prior to any use. Many open-source projects provide corresponding information. Only after this test has been completed successfully should the component be installed/used.

When integrating open-source components into the enterprise's own commercial products, the corresponding licence conditions must also be observed. Several licences require for example the disclosure of changes to the source code.

Further information can be found in [GS-OSS].

2.2. Response to vulnerabilities

Each product supplier should be prepared that a vulnerability may be detected in one of their components after they have been released and sold. Vulnerabilities are discovered at regular intervals, since improper implementations of software and hardware are not uncommon. The effects differ depending on the vulnerability type. In general, however, each type of error should be eliminated provided that this is reasonable in cost/benefit terms. In the individual case, it may also be recommended to change to a new product version or successor product. In such a case, a workaround should nevertheless be suggested so that existing customers are not completely unprotected.

M 9 Central contact possibility for vulnerability messages

The product supplier should have a central and confidential contact possibility by means of which the vulnerabilities can be reported. Furthermore, the information on the vulnerability should be communicated to the product supplier in encrypted form, since it is confidential data. For this purpose, a public cryptographic key can be stored in a directory service or on the website.

In addition to the possibility of contacting the product supplier, the product supplier should also respond in a timely manner. This response should include an assessment of the vulnerability and, if applicable, the further procedure.

More detailed information can be found in the cyber security recommendation "Vulnerability handling" [CS-E-Vuln].

M 10 Monitoring of used third-party components

The handling of vulnerabilities is not only limited to in-house developments. It also covers third-party product components which are used in the enterprise's own products. Here, the relevance and the effects for the enterprise's own components must be checked. After the corrections for the third-party components are available, the product supplier should release the automation components and, if applicable, make changes to their own product or define and communicate workarounds as an alternative.

M 11 Analysis of weakness messages
Resources to analyse incoming messages regarding vulnerabilities (from the two measures mentioned above) should be available. This includes tracing the vulnerability and assessing the criticality of this vulnerability as well as checking the developed updates or workarounds.

The enterprise should have an internal list of existing vulnerabilities.

M 12 Information channel for the notification of customers
If a vulnerability was detected or reported, a possibility to eliminate such vulnerability should be developed. Afterwards, the customers should be notified. This includes information describing the affected functions and the criticality of the eliminated problem. The customers should thus be enabled to assess if and to what extent they are affected. However, no detailed information which supports the exploitation of the vulnerability should be provided. Communication should take place via trusted channels.

In the case of a very critical vulnerability or if the elimination takes longer, a workaround which prevents a threat agent from exploiting a vulnerability or at least makes it more difficult for them should be provided in a timely manner if possible.

2.3. Cryptography

M 13 Selection of cryptographic procedures
The need and the requirements (for example, for encryption or signatures) should be determined (see [BSI-GS] S 2.162) and documented. Afterwards, cryptographic procedures (see [BSI-GS] S 2.164) suitable for the application scenario are chosen. The procedures should take the resources which are available in the components (e.g. restricted computing power) into consideration. When selecting the procedures, particular attention should be placed on the interoperability of the protocols. Recommendations for cryptographic algorithms are provided in [BSI-TR-02102-1]. Established libraries should be used. Due to the high complexity of the topic, the implementation of the enterprise's own library should be dispensed with.

When selecting suitable cryptographic procedures, the constraints of the national foreign trade law (keyword: export control) must also be observed. Product suppliers should already take this subject into account in the development process.

M 14 Monitoring / replacement of cryptographic procedures
In addition to the initial selection, it is also necessary to monitor the developments in this area during the entire life cycle of the components. It may well be that vulnerabilities in a procedure become known and that it should/must be replaced as a result of this. This must be taken into consideration in the development process.

M 15 Management of cryptographic secrets
For operation, it is important that

- The storage of the cryptographic secrets is protected against unauthorised access (reading must not be possible),
- It should be possible that the secrets are replaced by authorised persons,
- It should be possible to replace the devices, e.g. as part of automated / independent parameterisation and configuration processes (if secure authentication and authorisation has taken place) and
- The generation takes place in a protected and trusted environment.

To store the cryptographic keys, a Trusted Platform Module (TPM) or a smartcard, for example, may be used. Here, it must be taken into account that the storage space for the cryptographic keys is limited in these devices and the available algorithms are predefined. This means that the key length cannot easily be changed. Replacement, as in the case of a TPM, is only possible by changing the hardware. Long-term planning is required.

The generation of cryptographic secrets is a very complex matter. In this case, [BSI-TR-02102-1] is referred to.

2.4. Documentation

M 16 Internal documentation
During the development process, the decisions regarding the design and selection processes for components and technologies should be documented. This also includes threat and risk analyses as well as implemented measures. The libraries (also third-party components) and versions used as well as the tests performed and their results should also be documented. This information is required for the monitoring of vulnerability messages (see measure M 10).

This documentation must be protected against unauthorised access.

M 17 External documentation
External documents are made available by product suppliers to integrators who use components to build machines or plants and/or by integrators for asset owners of machines and plants. This mainly comprises the manuals for the components, component series, machines or plants. All information which is required for the secure configuration and the secure operation of the components (within the meaning of plant availability and security) should be included. Furthermore, all information which must be taken into account for decisions regarding the system architecture and overall systems to be implemented should also be included. Amongst other things, these include:

- The available interfaces and services (divided into inactive and active),
- The underlying standard protection requirements classification ([VDI2182-1] or [BSI 100-2] may be used for this purpose),
- Listing of the requirements for all external resources / assets requirements for the proper functioning of the overall system,
- Organisational and technical measures required from a a security perspective,
- Documentation of all product functions (especially of security configurations which cannot be changed by the user, since they are hard-coded for instance),
- Instructions for the secure commissioning (e.g. configuration of services, granting of rights, authentication measures),
- Documentation of the hardening measures still to be implemented if applicable (the product supplier should also provide the documentation for the already performed and implemented hardening measures). Especially when commissioning a new device, a performed/implemented and undocumented hardening measure may result in an additional workaround to realise the required functionality. The hardening measures include configurations and design aspects.
- Potential risks for different configurations and
- Information on further security measures.

2.5. Configuration

M 18 Minimality principle regarding authorisations
Users should only be granted the rights which are strictly required for realising their tasks (minimality principle, least-privilege). It should be particularly avoided to grant system administrator rights. Accounts which are not required should be deactivated or removed if possible (see [BSI-GS] S 2.32).

This aspect also includes system accounts. It is for example not necessary that a web server is operated with administrator rights. All applications and services should therefore run with corresponding access rights.

M 19 Minimality principle regarding system settings
Components in a critical environment should be configured in such a way that they provide as few points of attack as possible and only support the required functions. Therefore, all drivers which are not required, for example, should be removed and development tools (such as compilers / runtime environments) should not be installed. This not only reduces the vulnerabilities which might be exploited by a threat agent, but also makes it easier to maintain the system, since updates only have to be installed for the installed software (see [BSI-GS] S 4.95).

If it cannot be determined from the product supplier's perspective which software components a user may need, this must be determined as part of a specific system integration.

For optional services, it should be possible for asset owners or integrators to activate them when they are in use or to deactivate them when they are not in use.

M 20 Anti-virus software
The possible use of anti-virus software on the components should be tested by the product supplier/integrator. During this test, it should be ensured that

- Corresponding solutions are available,
- They were tested for absence of feedback (i.e. no negative impact on the ICS system performance or the system behaviour),
- The virus signatures can be updated at regular intervals (i.e. that there is a corresponding possibility to install regular updates) and
- A strategy was developed as to whether and how the anti-virus software can be updated during the operation of the plant (if the product supplier of the anti-virus software no longer provides updates of the signatures).

The result of this test may well be that it is not possible or reasonable to use anti-virus software. For example, this is the case if no suitable products are available or current virus signatures cannot be supplied due to the compartmentalisation of the systems or a change is not allowed. In these cases, other measures such as M 21, M 22 and M 23 should be implemented.

The information required for the configuration should be made available by the product supplier/Integrator.

In the case of embedded devices, there should be at least change detection in order to identify manipulations and perform tests.

M 21 Application whitelisting
In addition to the protection provided by anti-virus software, monitoring measures should be provided when starting applications.

Thus, it should be ensured that only released versions of executable files and no modified versions or malware which have been installed in an authorised manner can be started and executed.

Changes to the binary data of the applications as well as to the whitelist (list of executable files) may only be permitted to administrators (see [BSI-GS] S 4.23).

The configuration can already be made as part of the delivery by the product supplier. For the installation and configuration (e.g. in the case of the installation of plant-specific software by the asset owner) of the application whitelisting software, corresponding documentation must be provided by the product supplier.

M 22 Memory protection
Mechanisms should be used to prevent the memory from being accessed incorrectly (e.g. buffer overflow) and malicious exploitation. This basically applies, for example, to HMIs, distributed control systems and other high-performance systems. In embedded devices, these functions are usually not available.

M 23 Protection against process manipulation
Mechanisms should be used to prevent currently running processes from being manipulated, for example by the unauthorised dynamic exchange of runtime libraries or thread injection. This basically applies, for example, to HMIs, distributed control systems and other high-performance systems. In embedded devices, these functions are usually not available.

M 24 Activation of protection mechanisms for interfaces
For all interfaces, the security mechanisms offered should be activated. This includes, for example, the use of secure encryption procedures such as authentication. For example, this applies to Bluetooth, WLAN (see [BSI-GS] S 4.362) and Ethernet-based fieldbus systems if they provide these corresponding functions.

M 25 Hardware
Hardware interfaces should be deactivated or protected against misuse unless they are required for operation. They include, for example, USB, JTAG or SPI. Depending on the protection requirements, deactivation can be achieved for example by operating system functions, JTAG fuses or blocking the interfaces physically with locks. Another means is the use of SecureJTAG.

If the protection requirements are very high, measures against the physical manipulation of the devices (so-called tamper resistance) can also be considered. In general, only authorised personnel should be allowed and able to physically access the hardware (e.g. using lockable cabinets/racks and access controls).

2.6. Security of applications

2.6.1. General information

M 26 Validation of input and output data
All data transmitted to the component should be considered as not trusted prior to validation. Validation includes for example checking length limitations of fields and the coding of values for their correctness. If this is not checked, this may result in buffer overflows or the incorrect interpretation of data during the processing. Buffer overflows are one of the greatest threats, since it is possible to inject malicious code and have it executed.

This applies in particular to input fields of graphical user interfaces. If such a filtering of input and output data is technically impossible (e.g. due to very fast cyclic process data), it can be dispensed with. In these cases, increased focus must be placed on the robustness (see measure M 4) and it must be checked by performing tests.

If corresponding information is available, validation may also include checking the integrity and authenticity (see [BSI GS S 4.393]).

In general, these checks should take place in the component. To perform these checks, the following steps should be taken:

- Checking the length and the format of transmitted data, characters and character strings,
- Comparing the data with lists in which only allowed (whitelist) or prohibited (blacklist) characters/character strings are included. The whitelisting procedure should be used preferably.
- All characters classified as insecure for a possible interpreter should be removed or converted. This may be a browser which displays the data again at a later stage or a database into which the data is entered. These include, for example,
  - Unexpected JavaScript and HTML to be delivered to the client (web browser),
  - SQL statements inserted into the database in an unauthorised manner (e.g. from inputs in form fields)[1],

1 In this case, [BSI GS S 2.363] on the protection against SQL injection is also referred to.

– Commands for the operating system (e.g. in manipulated HTTP variables).

M 27 Error handling
In this context, an error refers to the failure of a part of a component (e.g. a network connection) or the interruption of the normal processing caused by incorrect data (e.g. input value outside the defined value range due to a transmission error or manipulation).

If such an error occurs while the component is operated, this error must be handled in such a way that a consistent and stable state of the component is guaranteed and data protection and the secure operation of the plant are therefore maintained if this is possible.

A component is in an inconsistent state if it is switched to an unexpected state due to an error and data is therefore processed in an uncontrolled manner (e.g. no error message if data is stored unsuccessfully) (see [BSI-GS] S 4.395).

The following items should be taken into consideration during error handling:

- No output of information which enables a threat agent to draw conclusions to the layout of the internal structure or to exploit vulnerabilities more easily,
- Internal logging of the error state with the possibility of reading the error states and their background (what has caused them) using secure procedures,
- Organised cancellation of the activity upon occurrence of the error
- Release of previously reserved resources upon notification / occurrence of the error,
- Taking into account additional aspects such as the time behaviour in the event of an error to prevent the control process from being affected or conclusions to the program run/internal information from being drawn,
- Ensuring the operation of the plant (e.g. performing no undesired status changes for plant components),
- Technical development measures in applications to prevent the memory from being accessed incorrectly (e.g. buffer overflow) and malicious exploitation.

2.6.2. Authentication & authorisation

Authentication refers to the confirmation of identity towards a communication partner. During authorisation, it is checked whether a person, an IT component, or an application is authorised to perform a specific action.

M 28 Changing standard authentication data
Operating a product with unchanged (i.e. default factory) security settings is a security risk. The number of possible solutions includes:

- Forcing the change during the installation and/or initial configuration,
- Note in the administration interface that a default password/cryptographic key has been set,
- Note in the documentation, preferably in a prominent place, that a default password/cryptographic key has been set and that it must be changed urgently.

In addition to the mentioned implementation options, there are further accompanying measures such as recommending requirements for secure passwords in the documentation or the technical implementation of such password policies.

It must be possible that an asset owner can exchange all passwords/cryptographic keys unless this is excluded by the requirements in the operational scenario and framework conditions.

M 29 Secure replacement of authentication data
Before authentication data is replaced, the user must provide authentication with the previous data.

When a user is required to authenticate themselves using a password, then some users will eventually forget their password. On the one hand, users should be helped quickly in such a situation so that they can get back to work. On the other hand, however, unauthorised persons must be prevented from gaining access to IT systems due to inadequate authorisation checks.

It is important when selecting a password reset procedure to specify flexible policies for resetting passwords. On the one hand, the procedure should meet the protection requirements of the particular password, and on the other hand, the access requirements of the users must be taken into account (see. [BSI-GS] S 2.402).

M 30 Secure storage of authentication data
In the case of passwords, they must not be stored as plaintext. So-called "salted hashes" for which each password is supplemented by a random value should be stored. This value can be stored together with the password. The random value does not have to be kept secret. It is used to make it more difficult to calculate the hashed password. If this value is not included in the hashed password, it is possible to calculate so-called "rainbow tables" in advance. They contain a password and the related hash value. With this information, only the suitable hash value must be searched for and the original password is known.

If alternative methods are used, information must be prevented from being read (see M 15). In the case of biometric procedures, [BSI-BioKeyS] is referred to.

M 31 Authentication in the case of remote access
Remote access should only be possible after successful authentication. For remote access, VPN techniques (e.g. IPSec, TLS) should be used. Additional, optional restrictions, e.g. based on IP address ranges (both on the remote and on the plant side) or disconnection after a certain period of inactivity, make sense.

M 32 Authorisation
After successful authentication, it should be checked at regular intervals in the application if the user had the rights to access the requested resources/data. This is particularly necessary if there are different roles.

2.6.3. Operation and monitoring

M 33 Logging
Security-relevant events (e.g. access to resources, authentication attempts) should be logged by specifying the time, the affected service/system/application, the user account used and the cause so that the logged data can be used to find the cause in the event of failures or errors or after attempted attacks (see [BSI-GS] S 4.397). Here, the following events in particular are of interest (see [BSI-GS] S 5.9):

- Input of an incorrect password for a user ID,
- Attempts of unauthorised (local and remote) access,
- Data on resource utilisation and overload,
- Connecting, removing and replacing removable media,
- Configuration changes.

The logged data should be transmitted to a central system. To be able to detect unauthorised changes to logged data, this data must be protected (e.g. using checksums). During the transmission to the central system, the additional occurring load on the network must be taken into account. Overloads should be avoided in any case.

The logged data must not contain sensitive and confidential data, such as passwords.

It should be possible to connect the component to a central logging and monitoring process.

2.7. Updates & restoration

M 34 Updates
It should be possible to have oneself informed about updates and workarounds, for example if vulnerabilities in used components become known at short notice (see M 12).

When providing updates, it should be ensured that

- They are only accepted if their authenticity and integrity has been checked successfully (e.g. by means of hash values),
- The product supplier documents which effects the update has on the security functions. This includes if and/or which measures may have to be adjusted (e.g. when using application whitelisting, see measure M 21),
- If and/or which non-availability must be expected,
- A distinction is made between critical and optional updates,
- The installation conditions are documented clearly (e.g. which previous versions must be installed),
- Information on new, changed or restricted functionalities is included,
- The asset owner is informed about changed default settings or reset configurations.

M 35 Backups and restoration
It should be possible to back up and restore the configuration and the existing data. Backups should be protected against subsequent changes (e.g. by means of digital signatures) as well as against unauthorised access (e.g. storage in encrypted form).

2.8. Protocol specifics

M 36 Use of secure IT and/or standard protocols
On the application level, secure versions of a protocols as well as accompanying mechanisms for encryption and the integrity and authenticity check should be used. Examples include the use of HTTPS instead of HTTP or FTPS instead of FTP.

For further supporting services, for example DNS, there are also secure versions, such as DNSSEC, which, however, may only be used in individual cases due to the the lack of widespread use. Here, whether they are used must be decided on the basis of the existing risk.

If necessary, precautions must be taken so that recalling the cryptographic keys, for example, does not affect other services (e.g. DNS) and the availability of infrastructures.

M 37 Use of current IT and/or standard protocol versions
Only the respective current protocol versions should be used. For the secure communication via TCP, for example TLS 1.2[2] should be used instead of older versions. In addition to the mere protocol version, cipher suite versions acknowledged as being secure should also be selected.

Much the same applies to the use of SNMP (Simple Network Management Protocol). Here, SNMPv3 should be used. Functions such as encryption and improved authentication were introduced there. It must also be taken into account that certain residual risks also remain when using SNMPv3. Especially brute force and dictionary attacks to authentication are possible. Therefore, it makes sense to integrate a suitable detection mechanism for such attacks (irrespective of the protocol used). Moreover, write access should only be implemented if necessary and/or it should be possible to deactivate it.

If older protocol versions are supported for reasons of compatibility, it should only be possible for the asset owner to use them upon explicit activation or it should be possible to deactivate them. In the

2 Version 10/2014

documentation, the product supplier must explicitly point out older/insecure protocol versions and the functional restrictions to be expected if these protocols are not activated or deactivated (by the asset owner).

M 38 XML
The parser used for XML data should be configured as restrictively as possible. This includes for example that references to external resources are not resolved and retrieved or that system commands are not executed.

Prior to processing, the XML data should be checked for application-specific points of attack (e.g. nesting too strong, too many elements).

2.9. Integration

Integrators should already take security into account as a fundamental aspect in the planning process. Amongst other things, these include:

- Architectural aspects,
- Careful and precise configuration and
- Instructions for the integration of the machine/plant into an existing security management of the asset owner if applicable (e.g. also organisational instructions or regarding the physical structure).

In addition to the measures mentioned in the chapters above, the following measures should also be taken into consideration.

The integrator uses the security functions included in the components to protect the machine/plant on several levels. Each component should be secured depending on the functions available to it (see M 39). To secure the individual components, machine or plant parts among one another, a subdivision into security zones should be made (see M 40 and M 41). Different protection mechanisms are thus lined up. This approach is referred to as defence in depth. With each layer, additional security functions are supplemented.

Supplementing security functions in outer layers does not mean that the subject of security does not have to be taken into account for the inner parts of the machine at all. They are rather additional hurdles which a threat agent has to overcome.

M 39 Activation of the security functions
The integrator should activate and use the security functions of the components offered by the components by default if they have determined during a previous examination that these functions are current and suitable for the use case.

M 40 Vertical segmentation
In a plant, the different areas such as production and office IT must be segmented by firewalls or other filtering or monitoring IT systems. If necessary, segmentation can also be realised by a perimeter network (DMZ). Unauthorised access between the subnetworks is thus made more difficult and/or is prevented.

M 41 Horizontal segmentation
Different plant/machine parts should be segmented of each other by a firewall. This prevents malware, for example, from being able to spread unhindered in the overall plant. Possible adverse effects on components and the operation of the plant thus only affect limited parts of the plant. This also applies to areas with different protection requirements.

M 42 Integration into the information security management system (ISMS)
The integrator should provide the information relevant to the integration of a machine, plant or plant parts into an existing information security management system (e.g. according to ISO 27001, [ISOIEC-62443-2-1] or [BSI-ICS]). If third-party components are currently not being considered, compliance with corresponding security specifications should be recommended nevertheless to the asset owner for planned future connections.

In the case of remote-maintenance installations, this includes details of which activities are possible and/or required and which information is transmitted. A controlled process must be developed for the remote maintenance between the asset owner and the user of the remote maintenance, in which access rights, logging and necessary release and/or approval procedures are clearly and comprehensibly defined.

3. Testing of components

3.1. Introduction to the subject of security tests for components

Security tests are carried out to detect threats as early as possible in order to be able to counteract them with effective measures even prior to delivery and/or commissioning.

As a positive side effect, security tests lead to a lower error rate in products and thus cost savings by reducing the need for subsequent improvement, an enhancement of the test process because of additional negative tests and sustainable learning processes for development.

The test realisation should

- Be carried out during the new or further development of components and plants or factory acceptance tests in the case of product suppliers and integrators,
- As part of site acceptance tests in the case of asset owners

by means of structured and reproducible tests.

The literature offers different approach models for security tests. Some models are referred to at this point:

- The Open Source Security Testing Methodology Manual [OSSTMM] describes different types of security tests,
- The Embedded Device Security Assurance [ISA-EDSA] is aimed at testing embedded devices and
- The OWASP Testing Guide [OWASP-2] is focused on web applications.

The security tests described below were summarised from practical experience gained in regular test management and during penetration tests. The questions shown in chapter 4 should be taken into account during the development process and their effectiveness should be checked at the end of the development process.

3.1.1. Limits

The information provided helps to improve the security in components. However, it cannot ensure that all vulnerabilities are found. This is not possible due to the complexity of the components and limited resources. Moreover, the information provided refers to already known vulnerabilities. Previously unpublished attack routes have therefore not been taken into account and cannot be tested directly.

3.1.2. Test team, test cases, test setup and test process

To reach the test objective, the formation of the test team and the design of the test process should be based on common and proven standards (z. B. ISO/IEC/IEEE 29119 "Software Testing").

It is particularly recommended

- Not to form the test team with employees from the respective developer team in order to ensure a neutral assessment of the system,
- To create test cases in as transparent and reproducible a way as possible.
- To preferably base the test setup on practical scenarios in order to achieve robust test results,
- To create the test process in such a way that transparent results are obtained for the subsequent correction of errors and establishment of effective measures.

- To test the solution again in the test process after the errors have been corrected and effective measures have been established.

3.1.3. Learning process and sustainable increase in quality

The feedback of the test team results in increasing expertise and awareness in the development team with respect to avoiding errors or architectural vulnerabilities.

3.2. Recommendations for the realisation of the tests

This chapter describes recommendations for the implementation of security tests. The recommendations are mainly compatible with test management best practices (e.g. ISO/IEC/IEEE 29119 "Software Testing") and can therefore be integrated into an already existing test management.

3.2.1. Creating a test plan

Like any well-structured project, security tests must be planned. When preparing the project, expenses and time frames should be estimated, and checked and adjusted if necessary during the course of the project.

When selecting the test tools, the function and the area of application should be tested to obtain useful results by means of a suitable selection. Extending the security tests should lead to increased automation. This reduces the expenses. Furthermore, the reproducibility of the tests is improved and more meaningful test results are achieved.

3.2.2. Test preparation

To specify test cases and assess the test results, the test team requires a sophisticated technical and functional understanding of the system to be tested as well as an understanding of possible vulnerabilities and security measures. An adequate familiarisation phase is essential for the test team and provides the basis for meaningful and robust test results.

For an efficient organisation of the upcoming tests, the following procedure has proven successful:

1. Gaining an overview
   The test team gains an overview of the system to be tested and operational scenarios to be expected on a functional level. Product descriptions, meetings with system architects, presentations and professional articles have proven successful. Requirements specifications which include essential requirements in addition to the product features may also be useful.
2. Collecting and examining documentation
   After the test team was able to gain an overview, all available documentation is collected and examined. It is recommended to document open issues, identified contradictions and suggestions for test scenarios unambiguously, but in bullet point form while examining the documentation.
3. Clarifying open issues and contradictions
   Any open issues and contradictions with test relevance must be clarified. Applying the common procedure of trying to clarify open issues with the test object during the test phase is not recommended. This could result in uneconomic expenses, unnecessary rework in test case specifications and especially misinterpretations with useless test results.
4. Initial assessment of possible vulnerabilities
   On the basis of the information collected, an initial assessment of possible vulnerabilities should be provided. They are aspects which should be examined more closely during the first steps of the security tests. However, there should be no restriction to the subjects identified here.

3.2.3. Creation of test cases and test scenarios

To prepare the active test realisation, test cases are developed and created. With each test case, it is to be tested if the system to be tested is resistant to a threat (negative test) or the measures taken counteract a threat effectively (functional test). In the case of logical or chronological relations between individual test cases, they can be summarised to become a test scenario.

Test cases and test scenarios should cover the following areas:

- Proof of defined robustness
- Identification of vulnerabilities (e.g. also in third-party components which are used)
- Targeted system-specific security tests

Test cases and test scenarios are derived from known details and available documents. Suitable written documentation of the tests to be realised and the results to be expected is essential. This reduces both misinterpretations by testing persons during the test realisation and makes the reproduction and assessment of findings easier at a later stage. Experience has shown that the robustness and quality of the test results is directly linked to the form and quality of the created test scenarios.

The following information can be used as test scenarios:

- Suggestions for test questions from chapter 4 of this document
- Which scenarios would jeopardise the security objectives in productive operation?
- How could a threat agent damage the system and the related data and processes technically and/or physically?
- Which security-relevant effects could configuration settings have?
- Which system-relevant effects could security configuration settings have?
- Which security-relevant scenarios have already occurred in the past?
- Which effects could operating errors have?

In the development of security-relevant test scenarios, the following aspects should be taken into account:

- When creating the test scenarios, it should be assumed that a potential threat agent has insider knowledge.
- Test scenarios may contain combined steps, but test scenarios which are too complex should be avoided. Instead, test scenarios with a defined initial state of the system are more purposeful in most cases.
- Individual components should be tested under the aspects of absence of errors and security and should not take any additional protection mechanisms (e.g. external firewall) into account in the first step. Only in the second step, involving these security mechanisms makes sense.
- Vulnerability scanners testing soft- or hardware for already known vulnerabilities in an automated manner should be used. Thus, it can be ensured that known vulnerabilities are avoided in new systems.
- The testing person should be able to write free text in the finding. Based on these remarks, the need for additional test scenarios can be identified in the follow-up. An assessment of the test coverage and depth should also be included.

3.3. General attack vectors

Irrespective of the specific implementation and the application protocol (e.g. Modbus, HTTPS), possible attack vectors via external interfaces to ICSs can generally be subdivided into six areas. It is the response of the system to:

1. Random input data
2. Changes to the transmission parameters (speed of the transmission rate, type and size of data packages etc.)
3. All message types specified for the respective protocol
4. Inputs which comply with the corresponding protocol specification in terms of content, but change the structure of the messages to non-standard values (lengths of variables, order of parameters etc.)
5. Inputs which structurally comply with the protocol specification, but change the content of variables to non-standard values (scope of variables, negative values, incorrect meta data of variables etc.)
6. Replay of recorded messages

Each of these six areas is a potential attack vector, since improper processing may result in the system being compromised or restrictions with respect to availability and/or processing speed.

Threat agents aim at descriptions for tests and/or checks of parameters which are missing in the specification or improper implementations as their targets. Especially the threat to the stability of a component by means of inadequate or the lack of validation of a parameter is an increased risk. It turns out in many cases that developers pay too little attention to this area so that such an unexpected input is responded to in an undefined manner. This may be related to a significantly delayed processing speed due to the full utilisation of internal resources, a restart or the complete "freeze" of the device so that it is necessary to reset the device manually.

3.3.1. Test scenarios for testing robustness

By means of robustness testing, the behaviour of the component to be tested will be simulated and monitored outside structured laboratory operations. This involves interactions or external influences. For example, a component should not be affected adversely by individual or a very large number of ICMP messages.

To suggest further ideas for suitable test scenarios with the objective of testing robustness, several aspects are listed below as examples. Several test scenarios may seem to be trivial, but are often not taken into consideration. It depends on the area of application and the possibilities offered by available interfaces. Not all questions provided below are therefore relevant in each case.

- Does the system remain stable even in the case of frequent function calls / activities changing in rapid sequences?
- Are operating errors at no time a threat to the stability of the system?
- Are value ranges tested correctly and incorrect inputs intercepted? Are the limit areas tested explicitly?
- Does the system remain stable after operations have been interrupted by a power failure?
- Are processes continued consistently after they have been interrupted by a power failure?
- Is the stability ensured in the case of network reception of data of common protocols (such as ICMP by means of Ping, SNMP, Broadcasts etc.)?
- Does the system respond in a stable and integer manner to devices / interfaces (such as USB devices, network cables, routing etc.) which are connected and removed unexpectedly?

3.3.2. Testing for known vulnerabilities

By testing for known and published vulnerabilities, unnecessary susceptibility and the related risks for the test object should be detected at an early stage wherever possible. This mainly applies to the use of third-party components.

The tests for widespread vulnerabilities are intended to prevent intentional and possibly harmful acts by third parties, since they can be exploited with average effort (time and knowledge) in most cases. Widespread vulnerabilities are often determined by threat agents by means of automated vulnerability scans. It is therefore recommended to use comparable tools for the testing.

To suggest further ideas for suitable test scenarios with the objective of testing vulnerabilities, several review questions are listed below as examples.

- Are vulnerabilities found by common and efficient vulnerability scanners and managers (e.g. the free OpenVAS[3])?
- Is the system free from (known) malware when it is delivered?
- Are messages from product suppliers, CVE, CERTs or other sources regarding known vulnerabilities available for all components (hard-/software) used in the test object in the used version and operating mode?

3.3.3. Test approach for devices with interfaces

When testing components with interfaces, two overall test approaches should be taken:

1. Testing the communication protocol
   In this part, the robustness and security of the implementation of the communication protocols are tested. This part only relates to transmission aspects.
2. Testing the data and processing logic
   The second part focuses on content data and its processing. This involves the secure processing of the data (e.g. when value ranges are exceeded).

In the case that the device to be tested supports several interfaces and protocols, all interfaces and protocols must be tested. If it cannot be excluded technically that the interfaces and protocols of a device could also be used in parallel, these tests may also be carried out in parallel to also test the separation of the individual types of access to the device and/or take the maximum resource utilisation of the device, if applicable, into account in such a case (peak and/or worst case load).

The following example is provided to illustrate the test approach:

Example: Web application for administration
Here, the implementation for both the TCP and the HTTPS stack should be tested. This involves the correct use of the certificates and private cryptographic keys for the connection and the response, for example, to expired or untrusted certificates. The actual data and their processing in the web application fall within the scope of the second test aspect.

3.3.4. Targeted security tests

By means of the tests described above, essential vulnerabilities of a component should already have been detected. Moreover, it is possible to search for specific vulnerabilities of a component which result from the characteristics of a combination of soft- and hardware.

3 http://www.openvas.org

This type of test relates to intentional and potentially harmful acts of third parties with above-average knowledge and targeted motivation. This type of test should only be contemplated if the test management has been established.

3.3.5. Planning and provision

As the creation of test scenarios progresses, the framework conditions to be taken into account for the test realisation become clear. Depending on the test scenarios and the system to be tested, a subdivision into subcomponents with final tests of the overall system may be necessary.

The setup of the test environment should be documented to ensure traceability for the following tests.

- If the system to be tested is already equipped with security measures (data diode, firewall, NIDS etc.), it is recommended to perform the planned test scenarios both with and without the security measures activated. This procedure ensures that both possible threats for individual components and/or the overall system and the effectiveness of the intended security measures are included in the test results.

3.4. Realisation

3.4.1. Readiness of the test object

Prior to each test, the test object must be set to the state defined for the test case. This includes, for example, configurations, cabling or other operating states. The correct initial situation is of fundamental importance for the successful realisation of the tests, since otherwise incorrect test results may be obtained or their reproducibility restricted.

3.4.2. Monitoring

The focus of many security tests is the response of the devices to be tested, especially their stability. It is therefore recommended to monitor it during the entire test procedure. Due to the different interfaces, their implementation as well as the possibilities of monitoring the components, only general suggestions can be made at this point. It must be taken into account that interfaces (e.g. a hardware-based communication stack) may be realised independently of the function to be tested. Such interactions must be taken into consideration.

- Use of network connection by permanently measuring the latency times (e.g. ICMP echo ping measurement). Here, the possibly independent realisation of the communication stack must be taken into account.
- In the case of a direct connection (e.g. USB or serial cable), use of a protocol monitor (hardware or software) to detect the outgoing traffic and/or its absence.
- Displays/alarms implemented in the device which visualise the overload and/or restart of the device. Accompanying the development or for closer examination of a vulnerability, debug interfaces or profiling or monitoring tools of binary programs and/or source code (if available) can be used. Here, attention should be paid to possible side effects.
- Visual monitoring by means of a camera in the case of devices with visible displays.

3.4.3. Test realisation

The test is realised according to the test planning in the intended order and with the provided resources.

During the test phase, prompt feedback of the testing person including test results and remarks for the development team have proven successful. This makes it possible to identify misunderstandings and errors in the description of test cases and test scenarios at an early stage and to make the required adjustments. Additional test cases and scenarios can thus be defined on the basis of the notes of the testing person.

3.5. Plausibility check of the test results

Despite careful test preparation and realisation, feedback must be checked for plausibility and confirmed if necessary. It must be excluded that it is an error in the test setup or a false-positive message. This may particularly occur in automated vulnerability scans. To prevent time and energy from being invested badly, the result should be checked for plausibility. An example may be that a vulnerability scanner detects an incorrect version number and, as a result, generates a reference to a vulnerability which may already have been eliminated in the version actually used.

3.6. Assessment of the results

The robust test results must also be prioritised. On the basis of this prioritisation, adequate measures to eliminate the vulnerabilities can be implemented in the next step.

In internal communications, it should be taken into account that results of security tests are recognised as that what they are, i.e. valuable knowledge with regard to quality assurance.

An assessment of the test coverage should also be made when assessing the findings. It is used as a quality measure of the significance of the tests realised.

3.7. Identification of causes and selection of suitable measures

All vulnerabilities found should be prioritised. Depending on the prioritisation, the elimination of the vulnerabilities is worked on. Here, the focus should be on the elimination, for example by eliminating the vulnerability in the source code, updating used libraries or changing the configuration. If this is not possible, the integrators or asset owners should take organisational measures, such as describing workarounds.

Experience has shown that measures are not enough to sustainably counteract threats if they have only a symptom-relieving or even covering-up effect.

After the derived security measures have been implemented, a new test run should be carried out to test the effectiveness of the measures and to be able to exclude adverse effects on other system components.

4. Test scenarios

This chapter is intended to provide support in creating suitable test scenarios by listing different review questions for selected protocols and interfaces. The review questions are based on the objectives described in chapter 3. Due to the high complexity of this subject, however, they make no claims to completeness. They must therefore be understood as a valuable aid.

The large number of questions do not necessarily explicitly address the subject of security, but help to avoid or detect possible errors in the development or configuration. Several questions are also of an organisational nature.

4.1. General questions during the implementation of interfaces

All interfaces with which data is accepted should be tested. Irrespective of the technology and the chosen format, several general aspects which should be taken into consideration apply:

4.1.1. Selection process

- Is the interface, the transmission protocol or the file format suitable to ensure the protection requirements and the requirements of the data to be processed in the respective area of application?
- Is the respective current version of the protocol used and/or are known vulnerabilities taken into account? How are new vulnerabilities responded to?

4.1.2. Implementation of specifications

- Does the implementation conform to the protocol / data specifications?
- Are all mandatory functions of the protocol / data specification implemented?
- Are all mandatory configurations of the protocol specifications implemented?
- Does the implementation remain stable when data for (non-)implemented functions is processed?
- Are all length limitations tested and processed correctly?
- Is it checked prior to the processing of data that the data is available in the correct data type?

4.1.3. Implementation

- Are resources that are assigned when an error occurs released again?
- Are measures for the quality assurance of the source code implemented (e.g. by means of a static code analysis)?
- Are code components and functions which are not used removed prior to delivery?
- Is the authorisation management implemented for users, data and third-party components?
- Can the selected software components be used on the hardware without restrictions? In some cases, functions, such as random number generators, are realised and integrated differently on different hardware platforms.

4.1.4. Vulnerability scanning

- Is active research done in third-party applications/program libraries used after the publication of vulnerability messages?
- Is this knowledge taken into account in the enterprise's own development (e.g. by firmware updates)?
- Are new components or components developed further tested using a vulnerability scanners at regular intervals?

4.1.5. Fuzzing

- Is the implementation of the protocol stable when it is exposed to unexpected data? For example, such unexpected data may include:
  - Character coding which is not supported,
  - Errors in serialised data streams,
  - Specification-compliant control data, but invalid or unauthorised user data,
  - Valid and authorised user data without specification-compliant control data,
  - Data outside the value range,
  - Incorrectly encrypted / signed data,
  - Does the information in the length fields correspond to the data and is it checked,
  - Changed orders of data fields or messages.
- Were all transmission parameters coming into question tested without a negative finding?
- Is data transmitted by the client validated in terms of data length and type?
- Is the implementation stable when it is exposed to interrupted connections in all communication phases?

4.1.6. Load test / availability

- How does the component respond to an overload at an interface (e.g. with too many requests)? Is this behaviour acceptable?
- How does the component respond to latency periods which occur during the transmission?
- How does the component respond to frequently occurring interrupted connections?
- How does the component respond if other services fail or are impaired (e.g. DNS, routing etc.)?

4.1.7. Logging

- Can changes to the security settings in a protocol be traced and proof demonstrated?
- Can the protocol entries be transmitted to a central log server?
- Can the component be included into the monitoring process?

4.1.8. Configuration

- Is the configuration data and its backups protected against malicious changes?

- Are only the required and documented application/services/functions available?
- How is default access data handled? Does this data have to be changed during commissioning?

4.1.9. Interfaces

In this context, an interface refers to an external connection by means of which it is possible to communicate with the component. This may be a network or USB connection.

- Are only the required physical interfaces available?
- Can interfaces which are not required be deactivated in a sustainable manner? Is this procedure recommended in the documentation and the procedure described?
- Can the ICS ensure that only certain components can be used afterwards?
- Were possibly required drivers tested for trustworthiness and stability?
- Were drivers which are not required deactivated?
- Were risks related to hardware manipulations for the ICS considered? Were effective measures taken?
- How does the component respond to an interrupted connection?
  - Is the data becoming available cached and transmitted at a later stage?
  - Are existing connection monitored to detect a failure?
- Have measures been taken to protect the confidentiality, integrity and authenticity of data?
- Is only data from authorised sources processed?

Prior to the commissioning of a plant, the following aspects should be tested:

- Is starting operation rejected with a clear and understandable message due to misconfiguration?
- Are default users and default passwords excluded in the default settings or can they be changed and/or can their change be forced prior to commissioning?
- Were measures ensuring a high password quality taken?
- Are other trust relationships in the default settings excluded?
- Are the default settings oriented towards ensuring the security objectives?

4.1.10. Network services

A network-based interface provides the systems participating in the network with a service via a protocol. Suggestions for test scenarios are described below:

- Which network services wait for incoming connections at the network interfaces?
- Have communication relations in the default configuration already been defined as restrictively as possible? Can the settings be changed further?

4.1.11. Cryptography

With respect to protocols which make use of cryptographic functions, the following aspects must be taken into consideration, amongst other aspects:

- Are suitable cryptographic algorithms and parameters (e.g. cryptographic key lengths as well as those currently considered to be secure) used?

- Was a long product life cycle taken into account during the selection process? This means: Was the further development of the computing power taken into account when selecting, for example, cryptographic key lengths?
- Are cryptographic keys stored securely?
- Is it possible that the asset owner exchanges the certificates and private keys?

An important subject is also the validation of certificates:

- Are the period of validity, holder (common name/subject alternative), issuer and status checked when using certificates?
- Are the certificate parameters, such as the intended purpose for encryption, signature or client and/or server authentication, checked?
- Are the issuer certificates and the certificate chain also checked?
- Is the current state of the certificates checked, for example by using revocation lists?
- Is the end user made aware of the aspects they have to take into account (e.g. comparing the fingerprint of the certificate in the browser with the documentation) when using HTTPS and root certificates which are not stored in the browsers?
- Can certificates be set on a whitelist (e.g. when using self-signed certificates)?

4.2. Industry-specific protocols

4.2.1. Modbus

The Modbus protocol is an industry-specific protocol. It does not offer functions for the authentication, encryption or integrity and authenticity of the data.

In addition to the rather general recommendations from chapter 4.1, several questions which can be used as suggestions for protocol-specific test scenarios are raised below:

- How does the implementation respond to commands which set the slave into a passive mode?
- How does the implementation respond to the instruction to restart the communication?
- How does the implementation respond to the instruction to empty, delete and reset diagnostic counters?
- Does the implementation respond to Modbus TCP packets with an incorrectly specified size or length in a stable manner?
- How does Modbus TCP respond if other data, protocols or mutilated Modbus messages are transmitted?
- How does the implementation respond if further messages or instructions are included in an exception PDU?
- How stable does the implementation remain if messages are sent by one server to all slaves?
- How does the implementation respond if a list of available instructions is retrieved?

4.2.2. PROFINET

The "Process Field Network" (abbreviated: PROFINET) is a modular protocol standard for automation.

With [PNO-1], the PNO already provides comprehensive information on the subject of robustness. In addition to the rather general recommendations from chapter 4.1, several questions which can be used as suggestions for protocol-specific test scenarios are raised below:

- Does the PROFINET implementation have a standard-compliant behaviour according to IEC 61158?
- Would the current implementation complete certification successfully or was a certification process completed successfully?
- Have security measures against DoS attacks and other unauthorised access and inappropriate access to automation components been established in the plant?
- How stable does the IO implementation respond to random or manipulated data in the PROFINET IO, RPC and UDP protocol?
- How stable does the IO implementation respond to random or manipulated data in the PROFINET DCOM & RPC and TCP protocol?
- Which effects does a high RPC / DCOM data payload have on the planned real-time requirements?
- Is the "General Station Description" (GSD) file adequately protected with respect to integrity?

4.2.3. OPC

4.2.3.1. OPC classic

"OLE Process Control" (OPC) is used to transmit measurement and control data. It does not offer functions for the authentication, encryption or integrity and authenticity of the data.

In addition to the rather general recommendations from chapter 4.1, several questions which can be used as suggestions for protocol-specific test scenarios are raised below:

- Was it possible to demonstrate in test scenarios that OLE / DCOM messages with harmful effects are successfully identified and rejected?
- Were the DCOM communication interfaces limited to the greatest possible extent?
- In the case of standardised use of identical Windows authentication data (user name and password) in systems involved: Does the available network interfaces and physical access options permit further possibilities of misuse? Were measures taken to reduce the risk?

4.2.3.2. OPC UA

OPC UA (OPC Unified Architecture, IEC 62541) is used to exchange data and information and to call services. It offers functions for authentication, encryption and integrity up to the data and service level.

- Were the security functions for OPC UA activated for encryption, signature and authentication?
- Are security aspects for providing and changing the certificates taken into account?
- Can access lists be configured for the registration of clients on the OPC UA server? Are they set up?
- Were the OPC client and server tested using the OPC Compliance Test Tools (CTTs) and, if necessary, were they certified by the OPC Foundation?
- Are the signatures checked not only mathematically for correctness, but also with respect to the validity of the certificates (e.g. trusted issuer)?

4.3. Communication interfaces

4.3.1. USB (Universal Serial Bus)

Universal Serial Bus (USB) is a standard for external interfaces, which enables communication between a system and different other peripheral devices.

In addition to the rather general recommendations, several questions which can be used as suggestions for specific test scenarios are listed below:

- Is the use of USB devices restricted?
- Were the components or the device drivers subjected to a conformity evaluation and have they completed this evaluation without any restrictions? For this purpose, the USB compliance test tool[4] made available by the USB-IF (USB Implementers Forum Inc.) may be used.
- Is it possible at all times to asynchronously disconnect and add the USB connection between the USB master and the USB slave without the devices, the USB stack or the software entering into an undefined state?
- If the USB master or the USB slave change their power mode (running → suspend mode and suspend mode → running) during an active connection, is this possible similarly to the disconnection without any loss of stability and data?
- The connection must be prepared for changes to the topology at the USB bus according to the USB standards. To achieve this, the USB master and the USB slave can be connected to each other by means of a USB hub to which other USB devices are asynchronously connected and removed similarly to the disconnection. Here, too, the tested USB stack should process all corresponding hotplug events and, if necessary, forward them to the application.

4.3.2. FireWire

FireWire (IEEE1394) was designed for high data transmission rates. Depending on the driver configuration, it also permits direct access to the memory of the host.

- Has the FireWire device completed the FireWire conformity tests fully and successfully without losing stability and response time?
- How does the FireWire implementation of the test object and, if applicable, the FireWire device respond to fuzzed data?
- Is it possible at all times to asynchronously disconnect and add the FireWire connection between the host and the slave without the devices, the FireWire stack or the software being set into an undefined state?
- If the host or the slave change their power mode (running → suspend mode and suspend mode → running) during an active connection, is this possible, in the same way as the disconnection, without any loss of stability and data?
- Is direct memory access (DMA) via the FireWire interface possible (read / write access)? A vivid example of a targeted attack is the so-called "inception" attack to FireWire interfaces[5]. Can a threat agent thus change the software permanently and thus store backdoors in the device?

4 http://www.usb.org/developers/docs#comp_test_procedures

5 http://www.breaknenter.org/projects/inception/

4.3.3. Wi-Fi

Wi-Fi is the overall term for radio standards which are based on the IEEE standard IEEE-802.11.

Several questions which can be used as suggestions for test scenarios are listed below:

- Were measures for the encryption of the data authentication taken?
- Does documentation provide explicit information about possibly existing risks? Are the required settings explained in the documentation? Are these settings activated in the condition upon delivery?
- When using Wi-Fi in ICSs with operating systems: Does the operating system store previously configured networks and authentication data?
  - Is it ensured that no connection to forged access points is established and that no data is intercepted via such a connection?
  - Is the access data for configured (wireless) networks securely stored in the operating system?

4.3.4. Bluetooth

Bluetooth is used for short-range wireless connections.

- Are only the two obligatory profiles as well as the profiles required for the communication active and/or available?
- Has authentication with a random cryptographic key been implemented?
- Can the authentication data be changed and/or is not the same cryptographic key used for each device?
- How does the Bluetooth stack in the test object respond to high data volumes? Does the Bluetooth stack in the test object also remain stable in the case of massive data traffic by Bluetooth vulnerability scanners?

4.3.5. IEEE 802.15.4 Wireless Personal Area Network

The main area of application of IEEE 802.15.4 should be self-organising ad-hoc networks. It should be possible to establish both meshed and peer-to-peer networks which, in turn, can be connected to larger so-called "cluster trees". Despite these communication-related requirements for the devices, the energy demand should be kept to a minimum. The standard was implemented in different open and proprietary protocols. Examples of this from the industrial environment include ZigBee, WirelessHART or ISA 100.11a.

- Are the encryption functions used?
- Are the access control lists used?
- How do the applications accessing the transmission layer respond to their failure or overload?
- Remain the communication partners stable if the data communication is confronted with unexpected contents by means of fuzzing?
- Can a message with a very high frame counter be transmitted successfully by a communication partner or third parties in the implementation of the test object? Which effects are related to this?

4.4. Application interfaces

4.4.1. HTTP/HTTPS

The Hypertext Transfer Protocol (HTTP) and/or the secure (preferred option) HTTPS (see measure M 36) with transport encryption and authentication of the server is mainly used for the transmission of websites or web services (in the case of SOAP or REST).

- Are only the required methods of the possible HTTP request methods GET, POST, HEAD, PUT, DELETE, OPTIONS, TRACE and CONNECT supported?
- Since HTTP TRACE can be misused for Cross-Site Scripting attacks: Is this HTTP method prevented effectively from being supported?
- Were potential threats adequately considered for the HTTP method CONNECT, such as the side-by-side installation of HTTP proxys?
- Does the server always generate fields in the HTTP response correctly (especially content type, content length, content encoding and transfer encoding)?
- Are HTTP requests to relative path information intercepted correctly (e.g. HTTP GET /subdir/../../)?

Suggestions for web-based applications:

- Were measures taken in all processes to compensate the nature of the HTTP protocol?
- Are authentication and authorisation always tested for access to files, directories, data and web-based functions? Is this also the case even if the requested resource is not supposedly linked otherwise?
- Are there no other temporary data, backups, default documents and scripts of the web server in the directories than the web-based application?
- Is the web application resistant to referer spoofing so that the referer specified by clients in the HTTP request does not affect the logical sequence at any time?
- Are Session IDs generated according to an actually random pattern?
- Are Session IDs never routed via the URL in the planned application process?
- Is a client never automatically assumed to be as authenticated in web-based applications even if certain properties seem to be complied with in the HTTP request (e.g. host name = superuser.kunde.de, client = UnserAdministrationClient v2)?
- Is the web-based application resistant to cookie manipulations?
- Can the web-based application still be used when the cookies are deactivated?
- Does the logic of the web-based application take into account that the specifications regarding the life of a cookie are not necessarily complied with by the other party?

4.4.2. FTP

The "File Transfer Protocol" (abbreviated: FTP) is a protocol used to transmit data between a client and a server.

With "FTP over SSL" (abbreviated: FTPS), an additional protocol layer was added between TCP and FTP to accomplish authentication and encryption between the FTP client and FTP server.

The first objective is achieved by means of a TLS handshake with the validation of the certificate at the client and/or server. The second objective is met afterwards by means of cryptographic encryption with an encryption standard acknowledged as being secure, using the validated certificates.

The "SSH File Transfer Protocol" (abbreviated: SFTP, or also known as Secure FTP) is an FTP implementation which relies on SSH for the management and transfer of data. SFTP can be introduced more easily into security infrastructures. In contrast to FTPS, the protocol requires only one single connection between the client and the server.

4.4.2.1. Suggestions for server-side test scenarios

- Was the choice between active and passive FTP balances against the end user's need-based operation of security infrastructures?
- Has authentication adjusted to the security level been implemented?
- Were measures implemented which make brute force attacks to authentication data more difficult and, at the same time, do not block the affected role / the FTP user completely for an unacceptable period of time?
- Is read and write access to directories and files sustainably bound to users / roles and cannot be bypassed?
- Was the need for encrypted communication between the client and the server checked?
- Can the FTP server be only reached by authorised clients?
- Is the FTP server free from known vulnerabilities?
- Was the FTP server adjusted so that it outputs as little information as possible about its designation and version number?
- Is the server resistant to directory traversal attacks (e.g. cd subdir/../../../)?
- Is the FTP server itself stable under load peaks, unexpected connections and commands?
- Does the FTP server support only the commands required for performing its services?
- Is it avoided that the same account data is used for secure (SFTP, FTPS) and insecure (FTP) applications?

4.4.2.2. Suggestions for client-side test scenarios

- Can the client handle interrupted connections in a stable manner?
- Are effective measures implemented according to the protection requirements in order to protect locally stored data?
- Does the client remain stable in the case of unexpected data encoding, responses and response times?

5. Mapping according to ISO/IEC 62443-4-1 and ISO/IEC 62443-4-2

The mapping of the measures shown was performed on the basis of the ISO/IEC 62443-4-1 draft in the version of June 2014. The mapping according to the ISO/IEC 62443-4-2 will be carried out as soon as the document has reached a stable form.

ICS Compendium	ISO/IEC 62443-4-1
M1	SMP-1.1, MIV-1, SIT-1
M2	SMP-1.4
M3	MIV-3
M4	SIT-2
M5	SRS-2
M6	
M7	
M8	DSG-1.6, SRP-1
M9	MIV-9, SRP-2.1
M10	SMP-1.1
M11	Sre-1,SRE-3
M12	
M13	
M14	
M15	SMP-1.1, SRS-1, SAD-5
M16	SRS-3, SAD-2, DSG-1
M17	DSD-2
M18	DSD-2
M19	
M20	
M21	
M22	
M23	
M24	MIV-7
M25	MIV-7
M26	DSD-3
M27	
M28	
M29	
M30	
M31	
M32	
M33	
M34	
M35	
M36	
M37	
M38	
M39	
M40	MIV-7

M41	(SAD-1)
M42	
M43	

Tabelle 1: Mapping according to ISO/IEC 62443-4-2 and ISO/IEC 62443-4-2

6. Overview of usable tools

As an introduction to the subject, the Linux distribution "Kali" provides a large number of tools. The following overview presents some of these tools. In addition to the open-source applications, there is a large number of commercial products of a comparable scope.

The tools primarily support only protocols which are common in office IT. Up to now, there are no corresponding tools for special industrial applications.

6.1. Vulnerability scanners

Vulnerability scanners offer the possibility to test systems and applications in an automated manner. They have databases which contain known vulnerabilities or have functions to detect them.
The scope of the tests varies. There are scanners which can test different applications from operating systems to individual applications. They are described in the first sub-chapter. They are followed by those scanners which were adjusted to the web interface.

6.1.1. General information

6.1.1.1. OpenVAS

"Open Vulnerability Assessment System" (abbreviated: OpenVAS) is an efficient vulnerability scanner and vulnerability manager, which is able to find vulnerabilities that are already known. For example, this may be the case in used third-party software. In the case of in-house developments, it is significantly less efficient, since there is no information available for this.

Homepage: http://www.openvas.org

6.1.2. Web applications

6.1.2.1. OWASP Zed Attack Proxy

The OWASP Zed Attack Proxy is a tool for HTTP(S)-based applications. The core function is an HTTP(S) proxy for the recording and manipulation of the communication. Moreover, a vulnerability scanner and an HTTP(S) Request Fuzzer are included for the verification of the input validation and processing.

Homepage: https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

6.1.2.2. w3af

w3af focuses on the automated identification of vulnerabilities of web applications. The tool already provides a selection of predefined profiles which simplify the first steps and can be used without further configuration.

Homepage: http://w3af.org

6.1.2.3. sqlmap

The "sqlmap" tool is interesting for applications which use a database in the background. This offers the possibility of testing the susceptibility to SQL injections in an automated manner.

It is suitable for being used in connection with OWASP Zed Attack Proxy for instance. It can collect URLs automatically by means of its web crawler and transfer them to sqlmap.

6.1.2.4. XSSer

Cross-Site Scripting is one of the most common threats to web applications. This is the case if the application accepts data of the users and transfers it to the browser without any validation and coding. Most XSS vulnerabilities can be found using automated tests and the code analysis.

A tool which checks an application for XSS vulnerabilities in an automated manner is the Cross Site Scripting Framework (abbreviated: XSSer) resulting from the OWASP XSSer Project.

Homepage: http://xsser.sourceforge.net/

6.2. Fuzzers

The term "fuzzing" refers to testing the behaviour of an application for random inputs. These may include files or also network data. The range covers completely random data up to specialised fuzzers for individual protocols, in which only individual fields are tested and the generation of checksums is also taken into account.

6.2.1. General information

6.2.2. Peach Fuzzer

Peach is an efficient fuzzer to generate and send both status-free and status-based messages. It must be emphasised in particular that data specifications are defined in the Peach Fuzzer by means of XML and programming skills in C# are useful, but not necessarily required.

The Peach Fuzzer can:

- Generate and send random data based on a predefined protocol specification to be complied with,
- Generate random data based on a predefined protocol specification with valid field values and send it in a random order,
- Generate and send random data on the basis of correct data packages with increasing mutation.

Furthermore, Peach (unlike other fuzzers) offers many monitoring, assessment and reproducibility options.

The tool can fuzz different file formats and COM/DCOM.

Homepage: http://peachfuzzer.com

6.2.3. IP

6.2.3.1. fuzz_ip6

"fuzz_ip6" is a simple and efficient IPv6 protocol fuzzer without a graphical user interface. It can be used to test the protocol stack for its proper and error-tolerant operation.

Homepage: https://www.thc.org/thc-ipv6/

6.2.4. Bluetooth

6.2.4.1. pwntooth

The pwntooth tool set was developed to be able to carry out tests towards Bluetooth devices under Linux in an automated manner. All tools supported by the tool set are delivered together with pwntooth.

6.3. Cryptography

6.3.1. SSL /TLS

6.3.1.1. sslyze

sslyze is suitable for testing interfaces using TLS. The tool checks which options are available for the TLS connections while making it possible to identify and correct any misconfigurations based on the results.

6.4. Static code analysis

An overview of tools for the static code analysis can be found at [OWASP-1]. Due to the different programming languages and development environments, no tool recommendations are given here.

Table of abbreviations

Abbreviation	Meaning
BSI	Federal Office for Information Security
CERT	Computer Emergency Response Team
CSRF	Cross Site Request Forgery
CVE	Common Vulnerability Enumeration
DCOM	Distributed Component Object Model
FTP	File Transport Protocol
HMI	Human Machine Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transport Protocol
ICMP	Internet Control Message Protocol
ICS	Industrial Control System
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IPSec	Internet Protocol Security
ISO	International Organization for Standardization
JTAG	Joint Test Action Group
OLE	Object Linking and Embedding
OPC	OLE for Process Control
OPC UA	OPC Unified Architecture
PNO	Profibus Nutzerorganisation e.V.
RPC	Remote Procedure Call
SNMP	Simple Network Protocol
SOAP	Simple Object Access Protocol
SPI	Serial Peripheral Interface
PLC	Programmable Logic Controller
SQL	Structured Query Language
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TPM	Trusted Platform Module
UDP	User Datagram Protocol
USB	Universal Service Bus
UTF-8	8-Bit UCS Transformation Format
VDE	*Verband der Elektrotechnik, Elektronik und Informationstechnik* [German Association for Electrical, Electronic & Information Technologies]
VDI	*Verein Deutscher Ingenieure* [The Association of German Engineers]
VPN	Virtual Private Network
WLAN	Wireless Local Area Network
XML	Extensible Markup Language

References

References

62443-4-1:	ISO/IEC, 62443-4-1 Product development requirements,
ACATECH-1:	acatech, agendaCPS - Integrierte Forschungsagenda Cyber-Physical Systems, 2012
BSI 100-2:	BSI, BSI-Standard 100-2: IT-Grundschutzvorgehensweise, https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/ITGrundschutzstandards/standard_1002_pdf, 2008
BSI 100-3:	BSI, BSI-Standard 100-3: Risikoanalyse auf der Basis von IT-Grundschutz, https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/ITGrundschutzstandards/BSI-Standard_1003, 2008
BSI-BioKeyS:	BSI, BioKEyS - Kryptografisch-biometrische Authentisierungssysteme mittels biometrischer Template-Protection-Verfahren, 2010, https://www.bsi.bund.de/DE/Themen/DigitaleGesellschaft/Biometrie/BSIProjekte/BioKeyS/biokeys.html
BSI-FLOSS:	BSI, Freie Software (FLOSS: Freier, Libre und Open Source Software), , https://www.bsi.bund.de/DE/Themen/DigitaleGesellschaft/FreieSoftware/index_htm.html
BSI-Glossar:	BSI, Glossar IT-Grundschutz, 2014, https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/glossar/04.html
BSI-GS:	BSI, IT-Grundschutz-Kataloge, 2014, https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/itgrundschutzkataloge_node.html
BSI-ICS:	BSI, ICS-Security-Kompendium, https://www.bsi.bund.de/ICS-Security-Kompendium, 2013
BSI-TR-02102-1:	BSI, Kryptographische Verfahren: Empfehlungen und Schlüssellängen, https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102, 2015
BSI-WEB:	BSI, Leitfäden zur Entwicklung sicherer Webanwendungen, https://www.bsi.bund.de/DE/Publikationen/Studien/Webanwendungen/index_htm.html, 2013
CS-E-TOP10:	BSI, ICS TOP 10 Threats and Countermeasures, https://www.allianz-fuer-cybersicherheit.de/ACS/DE/_/downloads/BSI-CS_005E.html, 2014
CS-E-Vuln:	BSI, Handhabung von Schwachstellen, https://www.bsi.bund.de/ACS/DE/_downloads/techniker/programmierung/BSI-CS_019.pdf, 2013
CWE-TOP25:	Common Weakness Enumeration, Top 25 Most Dangerous Software Errors, 2011, http://cwe.mitre.org/top25/
GS-OSS:	Nikolaus Lefin, IT-Grundschutz-Profil für Open-Source-Software, http://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/Hilfsmittel/Extern/Diplomarbeiten/Erstellung_IT-Profil_Lefin.pdf, 2010
ISA-EDSA:	ISA Secure, Embedded Device Security Assurance, , http://www.isasecure.org/en-US/Certification/IEC-62443-4-2-EDSA-Certification
ISOIEC-62443-2-1:	ISO/IEC, 62443-2-1 Reqirements fpr an IACS security management system Ed. 2.0 Profile of ISO27001/27002,
OSSTMM:	ISECOM, Open Source Testing Methodology Manual, , http://www.isecom/research/osstmm.html
OWASP-1:	OWASP, Static Code Analysis, 2015, https://www.owasp.org/index.php/Static_Code_Analysis

OWASP-2:	OWASP, OWASP Testing Guide, , https://www.owasp.org/index.php/OWASP_Testing_Project
PNO-1:	PROFIBUS Nutzerorganisation e.V., Test Specification PROFINET IO Security Level 1 / Netload,
SANS-Top25:	SANS, Top 25 Most Dangerous Software Errors, 2011, http://www.sans.org/top25-sofwtare-errors/
VDI2182-1:	VDI/VDE, VDI/VDE 2182 Informationssicherheit in der industriellen Automatisierung Blatt 1, 2011