Security requirement

Container

Deutsche Telekom Group

Version	4.0
Date	Dec 1, 2023
Status	Released

Publication Details

Published by
Deutsche Telekom AG
Vorstandsbereich Technology & Innovation
Chief Security Officer

Reuterstrasse 65, 53315 Bonn
Germany

File name	Document number	Document type
	3.64	Security requirement

Version	State	Status
4.0	Dec 1, 2023	Released

Contact	Validity	Released by
Telekom Security psa.telekom.de	Dec 1, 2023 - Nov 30, 2028	Stefan Pütz, Leiter SEC-T-TST

Summary
This document describes the functional security requirements used to secure Containers additional to the CIS Benchmark and hardening guides which also cover implementation-oriented requirements.

Table of Contents

1. Introduction

This security document has been prepared based on the general security policies of the Group.

The security requirement is used as a basis for an approval in the PSA process, among other things. It also serves as an implementation standard for units which do not participate in the PSA process. These requirements shall be taken into account from the very beginning, including during the planning and decision-making processes. When implementing these security requirements, the precedence of national, international and supranational law shall be observed.

If compliance with the described requirements can not be achieved or is only partially feasible in individual cases, risk assessments must be carried out together with a Security and/or Data Privacy Expert (in accordance with the relevant requirements) and possible alternative protective measures must be agreed.

2. Requirements

2.1. General

Req 1	Containers must be treated as immutable.

Containers are supposed to be immutable, hence they must not be modified during runtime. Instead of patching containers while they are running, patch the image and redeploy it. You must only alter your container images by using a existing CI/CD pipeline matching the CI/CD requirements.

Motivation: You will have a fresh container after each update and in the case of a vulnerability or injection they will be cleaned during the update.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized modification of data

For this requirement the following warranty objectives are relevant:

ID: 3.64-1/4.0

Req 2	Fixed tags must be used for immutability.

Use the most specific tag available. If an image has multiple tags e.g. :8 and :8.0.1 or :8.0.1-alpine, you must prefer the last, as it is the most specific reference. Avoid using generic tags like :latest. Remember that specific tags might be deleted.

Motivation: To avoid a specific image tag to become unavailable you must be running a trusted registry or account that is under your own control. Building images by yourself can be an advantage , because you maintain control of all components shipped with it.

For this requirement the following threats are relevant:
- Unauthorized modification of data
- Disruption of availability

For this requirement the following warranty objectives are relevant:

ID: 3.64-2/4.0

Req 3	Unnecessary packages must be avoided.

Containers must only have the essentials needed to run the intended application. An image must only contain a single piece of functionality for an application.

Motivation: This avoids running not needed software within containers to lower the attack surface.

For this requirement the following threats are relevant:
- Unauthorized use of services or resources

For this requirement the following warranty objectives are relevant:

ID: 3.64-3/4.0

Req 4 Least privileged user must be used.

When a Containerfile doesn’t specify a user, it defaults to executing the container using the root user. When that namespace is then mapped to the root user in the running container, it means that the container potentially has root access on the Docker host. The container must not run using the default root user, privileges must be dropped and a specific user must be used.

Motivation: Prevent containers running as the default root user.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized modification of data
- Unauthorized use of services or resources

For this requirement the following warranty objectives are relevant:

ID: 3.64-4/4.0

Req 5 Containers must be scanned for vulnerabilities.

Images must be scanned within your registry and during runtime for known vulnerabilities.

Motivation: Find known vulnerabilities within containers.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources

For this requirement the following warranty objectives are relevant:

ID: 3.64-5/4.0

Req 6 Container-specific host OS must be used.

Container-specific host operating systems must be used instead of general-purpose OSs. When using a container-specific host OS, attack surfaces are typically much smaller than they would be with a general-purpose host OS, so there are fewer opportunities to attack and compromise a container-specific host OS like e.g.: RedHat Atomic or Core OS.

Motivation: Container-specific OSs reduce the attack surfaces because of the minimalistic approach to run exclusively containers.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources

For this requirement the following warranty objectives are relevant:

ID: 3.64-6/4.0

Req 7	If there is CIS Benchmark available for the Container Runtime it must be applied.

For Docker the latest CIS Benchmark must be implemented: https://cisecurity.telekom.de/de/

- CIS Level 1 Must be implemented.
- CIS Level 2 should be implemented as far as possible.

If a level 2 Requirement would break functionality or would need disproportionate effort to implement it may be skipped.

Motivation: Establishing a secure configuration posture for Docker containers.

Implementation example: https://cisecurity.telekom.de/de/cis/benchmarks/
Example for LXD Containers : https://cisecurity.telekom.de/download/CIS/operating_systems/LXD/CIS_Ubuntu_18.04_LXD_Container_Benchmark1.0.0.pdf
Example for Docker : https://cisecurity.telekom.de/download/CIS/server_software/Docker/CIS_Docker_Benchmark1.5.0.pdf

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources

For this requirement the following warranty objectives are relevant:

ID: 3.64-7/4.0

Req 8	Namespace isolation must be ensured.

Resources a container can interact with must be limited. This includes file systems, network interfaces, interprocess communications, host names, user information, and processes. Ensure that apps and processes inside a container only see the physical and virtual resources allocated to that container. Namespace isolation provides each container with its own networking stack, including unique interfaces and IP addresses.

Motivation: Make containers more isolated

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources

For this requirement the following warranty objectives are relevant:

ID: 3.64-8/4.0

Req 9	Secret management must be provided.

Secrets must be stored outside of images and provided dynamically at runtime as needed.

Motivation: Most orchestrators, such as Docker Swarm and Kubernetes, include native management of secrets. These orchestrators not only provide secure storage of secrets and 'just in time' injection to containers, but also make it much simpler to integrate secret management into the build and deployment processes. For example, an organization could use these tools to securely provision the database connection string into a web application container. The orchestrator can ensure that only the web application container has access to this secret, that it is not persisted to disk, *and that anytime the web app is deployed, the secret is provisioned into it.*

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources

For this requirement the following warranty objectives are relevant:

ID: 3.64-9/4.0

Req 10 Security zones must be established.

Containers that are assigned to different security zones (e.g. DMZ-Zone, Application/Database- Zone) must be deployed on different hosts/host pools. A pool refers to a group of at least one or more VMs. In case of using Container as a Service (CaaS) platforms, these pools can be shared for several applications. For critical applications dedicated hosts/host pools must be used.If the only exposed service is an ingress controller (e.g. nginx) hardened regarding Security guidelines you can implement this by using only container specific measures e.g. Pod/Network Security Policies, furthermore there must be a runtime protection in place for all of your containers.

Motivation: Segmenting containers by purpose, sensitivity, and threat posture provides additional defense in depth. By grouping containers in this manner, it's more difficult for an attacker who compromises one of the groups to expand that compromise to other groups.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources

For this requirement the following warranty objectives are relevant:

ID: 3.64-10/4.0

Req 11 Production systems must be strictly separated from test and development systems.

This must be done by ensuring a strong tenant separation or using specific hosts/host pools for production and test/development.

Motivation: Maintaining multiple environments provides better security: To protect the integrity of your production data, you should limit access to it.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized use of services or resources

For this requirement the following warranty objectives are relevant:

ID: 3.64-11/4.0

Req 12 Pod Security Admissions / Pod Security Context or a 3rd party admission plugin must be used limit container privileges and provide platform rules.

Containers must specify their Security Context : https://kubernetes.io/docs/tasks/configure-pod-container/security-context/

A restricted default policy must be enabled. A very basic setup consists of an unprivileged policy and exception policies. Resources and volumes a Pod can access must be limited.

Policies can be enforced in multiple ways:

- Pod Security Admissions https://kubernetes.io/docs/concepts/security/pod-security-admission/
- Open Policy Agent / Gatekeeper https://www.openpolicyagent.org/
- Kyverno https://kyverno.io/

Policies must limit:

Control Aspect	**Field Names**
Running of privileged containers	privileged
Usage of host namespaces	hostPID, hostIPC
Usage of host networking and ports	hostNetwork, hostPorts
Usage of volume types	volumes
Usage of the host filesystem	allowedHostPaths
Allow specific FlexVolume drivers	allowedFlexVolumes
Allocating an FSGroup that owns the pod's volumes	fsGroup
Requiring the use of a read only root file system	readOnlyRootFilesystem
The user and group IDs of the container	runAsUser, runAsGroup, supplementalGroups
Restricting escalation to root privileges	allowPrivilegeEscalation, defaultAllowPrivilegeEscalation
Linux capabilities	defaultAddCapabilities, requiredDropCapabilities, allowedCapabilities
The SELinux context of the container	seLinux
The Allowed Proc Mount types for the container	allowedProcMountTypes
The sysctl profile used by containers	forbiddenSysctls,allowedUnsafeSysctls

Motivation: Restrict resources and rights of containers and minimize the attack surface.

ID: 3.64-12/4.0

Req 13 Linux kernel capabilities must be restricted within containers.

By default containers start with a restricted set of Linux kernel capabilities. This means that any process can be granted the required capabilities instead of giving it root
access. Using Linux kernel capabilities, processes in general do not need to run as the root user.
Rationale:
You must remove all capabilities not required for the correct function of the container. Specifically, in the default capability set provided by Docker, the NET_RAW capability should be removed if not explicitly required, as it can give an attacker with access to a container the ability to create spoofed network traffic
Overview of Docker capabilities : https://docs.docker.com/engine/security/#linux-kernel-capabilities

Motivation: The Linux kernel is able to narrow down the privileges of the Root user into distinct units referred to as capabilities.
Minimize attack surface of a container by restriciting it to only the capabilities it needs to function.

Implementation example: Example for Docker :

```
$ docker run --cap-drop ALL
Now add specific privileges to the container with the –cap-add
argument.
```

ID: 3.64-13/4.0

Req 14	The software used must be obtained from trusted sources and checked for integrity.

The software used on the system must be obtained from trusted sources and checked for integrity before installation.

This requirement applies to all types of software:

- Firmware and microcode for hardware components
- Operating systems
- Software Libraries
- Application Software
- Pre-integrated application solutions, such as software appliances or containers

as well as other software that may be used.

Trusted Sources
Trusted sources are generally considered to be:

- the official distribution and supply channels of the supplier
- third party distributors, provided they are authorized by the supplier and are a legitimate part of the supplier´s delivery channels
- internet downloads, if they are made from official provisioning servers of the supplier or authorized distributors
  (1) If the provisioning server offers various forms of downloads, those protected by encryption or cryptographic signatures must be preferred to those without such protection.
  (2) If the provisioning server secures the transport layer using cryptographic protocols (e.g. https, sftp), the associated server certificates or server keys/fingerprints must be validated with each download to confirm the identity of the provisioning server; if validation fails, the download must be cancelled and the provisioning server has to be considered an untrusted source.

Integrity Check
The integrity check is intended to ensure that the received software is free of manipulation and malware infection. If available, the mechanisms implemented by the supplier must be used for checking.
Valid mechanisms are:

- physical seals or permanently applied certificates of authenticity (if the software is provided on physical media)
- comparison of cryptographic hash values (e.g. SHA256, SHA512) of the received software against target values, which the supplier provides separately
- verification of cryptographic signatures (e.g. GPG, certificates) with which the supplier provides its software

In addition, a check of the software using an anti-virus or anti-malware scanner is recommended (if the vendor has not implemented any of the aforementioned integrity protection mechanisms for its software, this verification is mandatory).

Extended integrity checking when pulling software from public registries
Public registries allow developers to make any of their own software projects available for use. The range includes projects from well-known companies with controlled development processes, as well as from smaller providers or amateur developers.
Examples of such registries are:

- Code registries (e.g. GitHub, Bitbucket, SourceForge, Python Package Index)
- Container registries (e.g. Docker Hub)

Software from public registries must undergo an extended integrity check before deployment.
In addition to the integrity check components described in the previous section, the extended check is intended to explicitly ensure that the software actually performs its function as described, does not contain inherent security risks such as intentionally implemented malware features, and is not affected by known security vulnerabilities. If the software has direct dependencies on third-party software projects (dependencies are very typical in open source software), which must also be obtained and installed for the use of the software, these must be included in the extended integrity check.

Suitable methods for an extended integrity check can be, for example:

- Strict validation of project/package names (avoidance of confusion with deliberately imitated malicious software projects)
- dynamic code analysis / structured functional checks in a test environment
- static code analysis using a linter (e.g. Splint, JSLint, pylint)
- Examination using a security vulnerability scanner (e.g. Qualys, Nessus)
- Examination using a container security scanner (e.g. JFrog Xray, Harbor, Clair, Docker Scan)
- Examination using an SCA (Software Composition Analysis) tool or dependency scanner (e.g. OWASP Dependency Check, Snyk)

The test methods must be selected and appropriately combined according to the exact form of software delivery (source code, binaries/artifacts, containers).

Motivation: Software supply chains contain various attack vectors. An attacker can start at various points to manipulate software or introduce his own routines and damage or control the target environment in which the software is subsequently used. The attack can occur on the transport or transmission path or on the provisioning source itself. Accordingly, an attack is facilitated if software is not obtained from official and controlled sources or if an integrity check is omitted.
There is a particular risk for software obtained from public registries, as these are open to anyone for the provision of software projects. Perfidious attack methods are known, in which the attacker first provides completely inconspicuous, functional software for a while and as soon as it has established itself and found a certain spread, deliberately hidden malicious code is integrated in future versions. Other methods rely on similar-sounding project names for widely used existing projects or overruling version numbers to inject manipulated software into any solutions based on them.

Implementation example: Obtain the software via the official delivery channels of the supplier. Upon receipt of the software, immediately check for integrity using cryptographic checksums, as provided by the supplier, as well as scan for any infections by known malware using anti-malware / anti-virus scanners. Storage of the tested software on an internal, protected file storage and further use (e.g. rollout to the target systems) only from there.

For this requirement the following threats are relevant:
- Unauthorized modification of data
- Unnoticeable feasible attacks
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.01-2/7.0

Req 15	Known vulnerabilities in the software or hardware of the system must be fixed or protected against misuse.

Known vulnerabilities in software and hardware components must be fixed by installing available system updates from the supplier (e.g. patches, updates/upgrades). Alternatively, the use of workarounds (acute solutions that do not fix the vulnerability, but effectively prevent exploitation) is permissible. Workarounds should only be used temporarily and should be replaced by a regular system update as soon as possible in order to completely close the vulnerabilities.

Components that contain known, unrecoverable vulnerabilities must not be used in a system.

The treatment of newly discovered vulnerabilities must also be continuously ensured for the entire deployment phase of the system and implemented in the continuous operating processes of security patch management.

Motivation: The use of components without fixing contained vulnerabilities significantly increases the risk of a successful compromise. The attacker is additionally favored by the fact that, as a rule, not only detailed information on vulnerabilities that have already become known is openly available, but often also already adapted attack tools that facilitate active exploitation.

Implementation example: Following the initial installation of an operating system from an official installation medium, all currently available patches and security updates are installed.

Additional information:
The primary sources of known vulnerabilities in software/hardware are lists in the release notes as well as the security advisories from the official reporting channels of the supplier or independent CERTs. In particular, the reporting channels are sensibly integrated into continuous processes of security patch management for a system, so that newly discovered vulnerabilities can be registered promptly and led into operational remedial measures.
As a complementary measure to the detection of potentially still contained types of vulnerabilities that have in principle already become known, targeted vulnerability investigations of the system can be carried out. Particularly specialized tools such as automated vulnerability scanners are suitable for this purpose. Examples include: Tenable Nessus, Qualys Scanner Appliance.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources
- Disruption of availability
- Denial of executed activities
- Unnoticeable feasible attacks
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.01-10/7.0

2.2. Registry

Req 16 Registries must have user management in place.

For any registry, there must be a multi tenancy support, either done by the use of dedicated environments or by user management on the same instance.

Motivation: Separate and manage access to registries.

ID: 3.64-16/4.0

Req 17 Images from external registires must be downloaded to an internal registry first.

Images from external registires must be downloaded to an internal registry first so they can be scanned for vulnerabilities there. Within DT you can use the Magenta Trusted Registry https://mtr.devops.telekom.de./

Motivation: Ongoing, continuously updated, centralized reporting and monitoring of image compliance/vulnerabilities.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized modification of data

For this requirement the following warranty objectives are relevant:

ID: 3.64-17/4.0

Req 18 Connections to registries must be encrypted.

Registries must be configured to only be reachable from development tools, orchestrators and container runtimes over encrypted channels.

Motivation: Key goal is to ensure that all data pushed to and pulled from a registry occurs between trusted endpoints and is encrypted in transit.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data

For this requirement the following warranty objectives are relevant:

ID: 3.64-18/4.0

Req 19 Sufficient authentication and authorization must be used.

Any write access to a registry must require authentication. For example, only allow developers to push images to the specific repositories they are responsible for, rather than being able to update any repository. Registries also provide an opportunity to apply context-aware authorization controls to actions. For example, you can configure your continuous integration processes to allow images to be signed by the authorized personnel and pushed to a registry only after they have passed a vulnerability scan and compliance assessment.

Motivation: Ensure that only images from trusted entities can be added to regisries.

For this requirement the following threats are relevant:
- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data

For this requirement the following warranty objectives are relevant:

ID: 3.64-19/4.0

3. Technical Baseline Security for IT/NT Systems

3.1. Protecting data and information

Req 20	Stored data in need of protection must be protected against unauthorized access, modification and deletion.

The need for protection of stored data depends on its classification (e.g. according to applicable legal data privacy requirements, regulatory requirements, contractual obligations), the potential damage in the event of its misuse, and other relevant factors (e.g. the location of storage). The nature and extent of protective measures must be appropriately chosen.
Stored authentication attributes such as passwords, private keys, tokens or certificates etc. are generally considered to be in need of protection. Data that determines the functionality and security-relevant behavior of a system (e.g. system configuration files, operating systems and kernels, drivers) are also considered to be fundamentally in need of protection.

Compliance with the protection objectives of confidentiality, integrity and availability must be consistently guaranteed for stored data in need of protection. This also applies during only short-term storage (e.g. when storing in a web cache or in a temporary folder within a data processing chain).

Basically, access to data in need of protection in a system must be fully regulated on the basis of technically implemented authorization assignments and controls.

If such technical access control alone is no longer sufficient to ensure the necessary protection requirements of stored data, or if its effectiveness cannot be consistently ensured, additional cryptographic methods (e.g. encryption, signing, hashing) must be implemented. Cryptographic methods used in the storage of data must be suitable for this purpose and must have no known vulnerabilities.

Motivation: The storage of data on a system without adequate protection enables an attacker to view, use, disseminate, modify or destroy it without authorization. This potentially opens up additional attack vectors on the immediate and connected other systems and can lead to significant failures, loss of control and damage as well as resulting penalties and loss of reputation towards customers and business partners.

Implementation example: [Example 1]
A system exports data for transport to mobile media. Since the system's technical access control at the file permission level no longer applies as soon as the mobile media is removed from the system, additional measures must be taken to protect the data. Before the system writes the data to the mobile media, it is encrypted accordingly using a suitable algorithm. The associated encryption key is exchanged on a separate channel so that the data can be decrypted and processed again in the legitimate target system. An attacker who takes possession of the mobile media, on the other hand, has no access to the data.

[Example 2]
Only cryptographic hashes of passwords generated with a secure password hashing method are stored in the local user database of a system. For the system, these hashes are sufficient to authenticate users when they log on to the system. However, if an attacker can copy the user database, he does not immediately come into possession of plaintext passwords with which he could log on to the system on behalf of the users.

[Example 3]
On a system, the configuration files of the Web server can only be written by the legitimate admin in which corresponding permissions have been set in the file system. The access control of the operating system kernel thus denies all other users of the system to make changes to the configuration files of the web server; including the web server service account itself, which also reduces the attack surface from the outside in case of vulnerabilities in the web server.

For this requirement the following threats are relevant:
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Disruption of availability
- Unnoticeable feasible attacks
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.01-14/7.0

Req 21	Data in need of protection must be protected against unauthorized access and modification during transmission.

The need for protection of data to be transmitted depends on its classification (e.g. according to applicable legal data privacy requirements, regulatory requirements, contractual obligations), the potential damage in the event of its misuse, and other relevant factors (e.g. transmission via public networks). The nature and extent of the protective measures must be appropriately chosen.
Authentication attributes such as passwords or tokens etc. are generally considered to be in need of protection. Data that determines the functionality and security-relevant behavior of a system (e.g. updates & patches, configuration parameters, remote maintenance, control via APIs) are also considered to be fundamentally in need of protection.

Compliance with the protection objectives of confidentiality and integrity must be consistently guaranteed during the transmission of data in need of protection.

As a rule, this requires the implementation of cryptographic methods (e.g. encryption, signatures, Hashes).
Cryptographic methods may

- be applied directly to the data before transmission, which can make subsequent transmission acceptable even via insecure channels
- be used on the transmission channel to create a secure channel and protect any kind of data passing through it
- or be implemented as a combination of both.

Cryptographic methods used in the transmission of data must be suitable for this purpose and must have no known vulnerabilities.

Motivation: The transmission of data without adequate protection enables an attacker to intercept, use, disseminate, modify or remove it from transmission without authorization. This potentially opens up further attack vectors on the immediate target systems as well as connected other systems and can lead to significant failures, loss of control and damage as well as resulting penalty claims and reputational losses towards customers and business partners.

Implementation example: [Example 1]
Confidential documents are encrypted before they are sent by e-mail to the customer.

[Example 2]
An administrator configures a new cloud application over the Internet. Access is via a TLS-encrypted connection ("https").

[Example 3]
A system obtains automatic software updates from an update server. The update server delivers the software updates cryptographically signed. The system can thus validate the received software updates and reliably rule out that they have been manipulated during transmission.

For this requirement the following threats are relevant:

- Unauthorized access or tapping of data
- Unauthorized modification of data
- Disruption of availability
- Unnoticeable feasible attacks
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.01-15/7.0

3.2. Authentication attribute "password"

Req 22	If a password is used as an authentication attribute for technical accounts, it must have at least 30 characters and contain three of the following categories: lower-case letters, upper-case letters, digits and special characters.

Technical user accounts are characterized by the fact that they are not used by people. Instead, they are used to authenticate and authorize systems to each other or applications on a system.

A system must only use passwords for technical user accounts that meet the following complexity:

- Minimum length of 30 characters
- Comprising at least three of the following four character categories:
    - lower-case letters
    - upper-case letters
    - digits
    - special characters

Motivation: Due to their use in machine-to-machine (M2M) communication scenarios, technical user accounts are often equipped with privileges that can be of high interest to an attacker to compromise infrastructures. Without mechanisms of extensive compromise detection, the risk of a password being determined or broken by an attacker can increase significantly over time. A significant increase in password length counteracts these risks and can also be implemented particularly easily in M2M scenarios, since handling a very long password is not a particular challenge for a machine (as opposed to a person).

For this requirement the following threats are relevant:

- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources
- Denial of executed activities

For this requirement the following warranty objectives are relevant:

ID: 3.01-27/7.0

3.3. Logging

Req 23	Security relevant events must be logged with a precise timestamp and a unique system reference.

Systems must log the occurrence of security-relevant incidents. So that these events can be evaluated and classified, they must be logged together with a unique system reference (e.g., host name, IP or MAC address) and the exact time the incident occurred ("Timestamp").

Exceptions of this requirement are systems for which logging cannot be implemented because of building techniques, use case or operation area. Examples for these kind of systems are customer devices such as Smartphones or IADs/ home gateways (e.g. Speedport).

The Timestamp of a logged event must contain at least the following information:

- date of the event (Year, Month, Day)
- time of the event (Hours, Minutes, Seconds)
- Timezone, those information belongs to

When logging, the applicable legal and operational regulations must be observed. The latter also include agreements that have been made with the company's social partners. Following these regulations logging of events is only allowed for a defined use case. Logging of events for doing a work control of employees is not allowed.

In addition - as for any data that is processed by a system - an appropriate protection requirement must also be taken into account and implemented for logging data; this applies to storage, transmission and access. In particular, if the logging data contains real data, the same protection requirements must be taken into account that is also used for the regular processing of this real data within the source system.

Typical event that reasonable should be logged in many cases are:

Event	**Event data to be logged**
Incorrect login attempts	• User account, • Number of failed attempts, • Source (IP address, client ID / client name) of remote access
System access from user accounts with administrator permissions	• User account, • Access timestamp, • Length of session, • Source (IP address) of remote access
Account administration	• Administrator account, • Administered user account, • Activity performed (configure, delete, enable and disable)
Change of group membership for accounts	• Administrator account, • Administered user account, • Activity performed (group added or removed)
Critical rise in system values such as disk space, CPU load over a longer period	• Value exceeded, • Value reached (Here suitable threshold values must be defined depending on the individual system.)

Logging of additional security-relevant events may be meaningful. This must be verified in individual cases and implemented accordingly where required.

Motivation: Logging security-relevant events is a basic requirement for detecting ongoing attacks as well as attacks that have already occurred. This is the only way in which suitable measures can be taken to maintain or restore system security. Logging data could be used as evidence to take legal steps against attackers.

For this requirement the following threats are relevant:
- Denial of executed activities
- Unnoticeable feasible attacks

For this requirement the following warranty objectives are relevant:

ID: 3.01-33/7.0

Req 24	Security-relevant logging data must be forwarded to a separate log server immediately after it has been generated.

Logging data must be forwarded to a separate log server immediately after it has been generated. Standardized protocols such as Syslog, SNMPv3 should be preferred.

Motivation: If logging data is only stored locally, it can be manipulated by an attacker who succeeds in compromising the system in order to conceal his attack and any manipulation he has performed on the system. This is the reason why the forwarding must be done immediately after the event occurred.

For this requirement the following threats are relevant:
- Unauthorized modification of data
- Disruption of availability
- Denial of executed activities
- Unnoticeable feasible attacks

For this requirement the following warranty objectives are relevant:

ID: 3.01-35/7.0

Req 25	For security-relevant logging data that is forwarded to the separate log server, compliance with the applicable retention and deletion periods must be ensured.

The following basic rules must be taken into account:

- security-related logging data must be retained for a period of 90 days on the separate log server.
- after 90 days, stored logging data must be deleted immediately on the separate log server.

Deviances
Different retention periods and deletion periods may exist due to legal or regulatory requirements (especially in connection with personal data) or may be defined by contractual agreements. In these cases, the applicable periods must be agreed individually with a Project Security Manager (PSM) / Data Privacy Advisor (DSB) or are specified by them.

Log server under the responsibility of a third party
If the selected separate log server is not within the same operational responsibility as the source system of the loggin data, it must be ensured that the responsible operator of the log server is aware of the valid parameters for the logging data to be received and that they are adhered to in accordance with the regulations mentioned here.

Motivation: Logging data is an immensely important IT security tool for preventing, detecting and clearing up system faults, security and data privacy incidents. On the other hand, the recording of logging data, like any other data processing, is also subject to legal and regulatory requirements. Accordingly, guidelines must be adhered to that reconcile the two.

Implementation example: Taking into account the current legal situation and applicable data privacy regulations, the following deletion periods for forwarded security-relevant logging data from an exemplary telecommunications system are implemented on the separate log server:

- Standard System Logs: Deletion after 90 days at the latest
- Logging of public IP addresses: Deletion (or anonymization) after 7 days at the latest
- Logging of the assignment of dynamic public IP addresses by the telecommunication solution: Deletion after 7 days at the latest
- Logging of non-billing-relevant call detail records: Deletion after 7 days at the latest
- Logging of the content of e-mail and SMS: Deletion after 24 hours at the latest
- Logging of the domain queries handled by the DNS server of the telecommunications solution: Deletion after 24 hours at the latest

For this requirement the following threats are relevant:
- Unauthorized access or tapping of data

- Denial of executed activities
- Unnoticeable feasible attacks

For this requirement the following warranty objectives are relevant:

ID: 3.01-36/7.0

Req 26	The system must provide logging data that is required to detect the system-specific relevant forms of attack in a SIEM.

The forms of attack that are typically to be expected for the present system must be systematically analyzed and identified.
The MITRE Attack Matrix (https://attack.mitre.org) can be used as a structured guide during such an identification.

It must be ensured that the system generates appropriate logging data on events that are or may be related to these identified forms of attack and that can be used to detect an attack that is taking place.

The logging data must be sent to a SIEM immediately after the system event occurs.
SIEM (Security Information & Event Management) solutions collect event log data from various source systems, correlate it and evaluate it automatically in real time in order to detect anomalous activities such as ongoing attacks on IT/NT systems and to be able to initiate alarms or countermeasures.
The immediate receipt of system events is therefore absolutely crucial for the SIEM to fulfill its protective functions.

Note:
The immediate need to connect a system to a SIEM is specifically regulated by the separate "Operation" security requirements catalogs.
If the present system does not fall under this need, the requirement may be answered as "not applicable".

Motivation: A SIEM as an automated detection system for attacks can only be effective if it continuously receives sufficient and, above all, system-specific relevant event messages from the infrastructures and systems to be monitored. General standard event messages may not be sufficient to achieve an adequate level of detection and only allow rudimentary attack detections.

Implementation example: An example system allows end users to log in using a username and password. One of the typical forms of attack for this system would be to try to discover and take over user accounts with weak or frequently used passwords by means of automated password testing (dictionary or brute force attack). The example system is configured to record every failed login event in system protocols ("logs"). By routing this logging data in parallel to a SIEM, the SIEM can detect in real time that an attack is obviously taking place, alert it and thus enable immediate countermeasures.

ID: 3.01-37/7.0