Security requirement

Nginx

Deutsche Telekom Group

Version	5.0
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
	3.80	Security requirement
Version	**State**	**Status**
5.0	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 was created on the basis of the general security policies of the Group and defines the requirements for securely implementing nginx webservers. The requirements described in this document shall be met to ensure that a nginx webserver cannot be easily misused by attackers.

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.

2. Nginx webserver platform requirements

Req 1	Software and hardware of the system must be covered by security vulnerability support from the supplier.

Only software and hardware products for which there is security vulnerability support by the supplier may be used in a system.

Such support must include that the supplier

- continuously monitors and analyzes the product for whether it has been affected by security vulnerabilities,
- informs immediately about the type, severity and exploitability of vulnerabilities discovered in the product
- timely provides product updates or effective workarounds to remedy the vulnerabilities.

The security vulnerability support must be in place for the entire period in which the affected product remains in use.

Support phases with limited scope of services
Many suppliers optionally offer time-extended support for their products, which goes beyond the support phase intended for the general market, but is often associated with limitations. Some suppliers define their support fundamentally in increments, which may include limitations even during the final phase before the absolute end date of regular support.
If a product is used within support phases that are subject to limitations, it must be explicitly ensured that these restrictions do not affect the availability of security vulnerability support.

Open Source Software and Hardware
Open Source products are often developed by free organizations or communities; accordingly, contractually agreed security vulnerability support may not be available. In principle, it must also be ensured here that the organization/community (or a third party officially commissioned by them) operates a comprehensive security vulnerability management for the affected product, which meets the above-mentioned criteria and is considered to be reliably established.

Motivation: Hardware and software products for which there is no comprehensive security vulnerability support from the supplier pose a risk. This means that a product is not adequately checked to determine whether it is affected by further developed forms of attack or newly discovered vulnerabilities in technical implementations. Likewise, if there are existing security vulnerabilities in a product, no improvements (e.g. updates, patches) are provided. This results in a system whose weak points cannot be remedied, so that they remain exploitable by an attacker in order to compromise the system or to adversely affect 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
- 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: 3.01-1/7.0

3. Nginx webserver installation requirements

Req 2	Access rights for web server configuration files must only be granted to the owner of the web server processs or a user with system privileges.

Motivation: Configuration files may only be written by the owner of the web server process or a user with system privileges. Otherwise it would be possible for unauthorized users to change the configuration of the web server or to obtain configuration information which could be used for an attack.

Implementation example: Delete “read” and “write” access rights for “others.” Only grant “write” access to the user who configures the web server.

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

For this requirement the following warranty objectives are relevant:

ID: 3.03-12/6.0

Req 3	Information on the Nginx web server in error pages, which are supplied by the web server, must be removed.

Default error pages must be replaced with user-defined error pages.
User-defined error pages must not include version information about the web server and the modules/enhancements used. Error messages must not include internal information such as internal server names, version numbers, error codes, etc.

Motivation: Any information about the web server could allow conclusions to be drawn about security vulnerabilities.

Implementation example: The status information supplied by the Nginx web server must be disabled in the configuration using the command "server_tokens off".
The separate error pages are also defined in the configuration:

```
location / {
error_page 404 /mein_error_404.html;
error_page 500 502 503 504 /mein_error_50x.html;
}
```

Furthermore, error messages can also be summarized here or refer to a URL.
In order to conceal the web server itself as well, the name of the web server can be changed in the file "src/http/ngx_http_header_filter_module.c".
Nginx must then be recompiled.

Line:

```
static char ngx_http_server_string[] = "Server: nginx" CRLF;
static char ngx_http_server_full_string[] = "Server:" NGINX_VER CRLF;
```

change in:

```
static char ngx_http_server_string[] = "Server: my Web Server" CRLF;
static char ngx_http_server_full_string[] = "Server: my Web Server" CRLF;
```

For this requirement the following threats are relevant:
- Disruption of availability
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.80-3/5.0

4. Nginx webserver configuration requirements

Req 4	All web server processes must not run with system privileges.

If the process is started by a user with system privileges, a different user ID without system privileges must be used after the start.

Motivation: If the web server process runs with administrative access rights, an attacker who obtains control over this process would be able to control the entire system.

Implementation example: ...

```
# nginx.conf
#
user www-data nogroup;
# User and group without system authorization to execute the web server.
...
```

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
- Unnoticeable feasible attacks

For this requirement the following warranty objectives are relevant:

ID: 3.80-4/5.0

Req 5	The web server service must be bound only to interfaces, which are necessary to connect the service.

In most cases the web server service needs to be bound only to one interface.

Motivation: The more interfaces provide access to the web server, the higher is the attack risk.

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

For this requirement the following warranty objectives are relevant:

ID: 3.03-5/6.0

Req 6	Any add-ons and components that are not required must be deactivated.

All optional add-ons and components of the web server must be deactivated if they are not required. In particular,

- CGI
- Server Side Includes (SSI)
- WebDAV

must be deactivated if they are not required.

Motivation: Each add-on, component or function can have security vulnerabilities.

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: 3.03-7/6.0

Req 7 Prevent long runtimes of web server queries.

Long runtimes of web server queries by clients should be prevented with the aid of defined timeouts.

Motivation: The web server's performance can be substantially impaired through long timeouts. This may occur deliberately through an attack. In order to prevent such attacks, fixed timeout periods must be defined in the configuration. Nginx offers a timeout directive in this respect virtually for every function.

Implementation example:

...

```
# nginx.conf
#
client_body_timeout 8;
client_header_timeout 8;
keepalive_timeout 5 5;
send_timeout 8;
```

Since the used directives are deployed individually for each web server, this can only serve as an example and apply as a reference.
Other directives: http://nginx.org/en/docs/dirindex.html

...

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

For this requirement the following warranty objectives are relevant:

ID: 3.80-7/5.0

Req 8 Directory listings (indexing) must be deactivated.

Motivation: Directory listings provide information on files and directory structure which could be misused.

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

For this requirement the following warranty objectives are relevant:

ID: 3.03-14/6.0

Req 9 Unnecessary HTTP methods must be disabled.

Standard requests to web servers only use GET, POST and HEAD. If other methods are required, they must be processed securely.

Motivation: HTTP TRACE could be misused by an attacker. This method allows for debugging and the trace analysis for connections between the client and the web server. Other HTTP methods could also be used to obtain information about the server, or they could be directly misused by an attacker.

Implementation example:

```
...
# nginx.conf
#
if ($request_method !~ ^ (GET|POST|HEAD)$ ) {
return 444;
}
...
```

For this requirement the following threats are relevant:

- Unauthorized access or tapping of data
- Unauthorized modification of data
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.80-9/5.0

Req 10 CGI shall not be used.

Motivation: Inappropriate CGI configuration may allow multiple attack vectors. Modern web servers provide safer and more performant alternatives to CGI. Therefore CGI is neither necessary nor recommended.

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: 3.03-8/6.0

4.1. HTTPS

Req 11 The TLS protocol must be used for encryption with HTTPS.

SSL must be regarded as outdated and must therefore not be enabled or must be disabled.
Instead, the TLS protocol must be used.

Motivation: A host of vulnerabilities are known especially with regard to SSLv2, which makes usage impossible from a security perspective. TLS has now been established as a further development of SSL for many years so that there is no reason to continue using SSL.

Implementation example: The current version of Nginx uses
“ssl_protocols TLSv1.2” as default.

```
...
# nginx.conf
#
worker_processes 2;
```

```
http {
ssl_session_cache shared:SSL:5m;
ssl_session_timeout 5m;
server {
listen 443 ssl;
server_name www.server1.com;
ssl_certificate www.server1.com.crt;
ssl_certificate_key www.server1.com.key;
ssl_protocols  TLSv1.2;
...
ssl_prefer_server_ciphers on;
```

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

For this requirement the following warranty objectives are relevant:

ID: 3.80-11/5.0

Req 12	The web server must be configured in such a way that the use of the latest version of the TLS protocol is enabled.

Motivation: The latest version of the protocol offers the best possible protection and contains fixes to known vulnerabilities in previous versions of the protocol.

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

For this requirement the following warranty objectives are relevant:

ID: 3.03-22/6.0

Req 13	The TLS configuration must use secure cipher suites.

Acceptable cipher suites may only use the following algorithms:

Server/Client Authentication & Key Agreement	**Encryption**	**Message Authentication & Integrity (MAC)**

ECDHE_ECDSA	AES_128_CBC	SHA256
ECDHE_RSA	AES_128_GCM	SHA384
DHE_DSS[1]	AES_128_CCM	SHA512
DHE_RSA[1]	AES_192_CBC	SHA-3-256
	AES_192_GCM	SHA-3-384
	AES_192_CCM	SHA-3-512
	AES_256_CBC	
	AES_256_GCM	
	AES_256_CCM	
	CHACHA20_POLY1305	

[1] min. 4096-bit Parameter

TLS 1.3 explicitly specifies the usage of only DHE and ECDHE for server/client authentication and key agreement. Thus TLS 1.3 cipher suite notation does not contain an indication in this regard.

By fulfilling this requirement the Perfect Forward Secrecy (PFS) property in the TLS/SSL implementation will be achieved.

Motivation: Cipher suites known to be unsecure do not offer sufficient protection.

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

For this requirement the following warranty objectives are relevant:

ID: 3.03-23/6.0

Req 14	The TLS configuration must provide that the cipher suite considered most secure is being chosen with highest priority.

A cipher suite contains the definition of four algoritthms. These are used for key exchange, authentication, encryption and as a hash function. General guidelines for the prioritization are

- For the key exchange the Diffie-Hellman method must be used because it offers perfect forward secrecy. Cipher suites using the Diffie-Hellman method may be identified by the strings DHE or ECDHE. ECDHE has higher priority than DHE.
- For encryption the Advanced Encryption Standard (AES) with a key length as big as possible has to be used
- As a hash function SHA-2 or SHA-3 has to be used. These functions usually may be identified by the string SHA or SHA-3**followed by a number**(256, 384 or 512). Warning: if the string SHA is not followed by a number this identifies the SHA-1 function which is significantly less secure.

Motivation: When a TLS connection is being established a cipher suite is selected based on the cipher suites available both on client and on server side. In order to ensure a high compatibility to all kinds of client systems the web server must not only allow for the cipher suites considered most secure. To make sure that nevertheless for each client the best possible cipher suite is selected and thus the connection is best protected the configuration must contain an according prioritization.

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

For this requirement the following warranty objectives are relevant:

ID: 3.03-24/6.0

Req 15	Certificates must be issued by a certification authority whose certificates are recognized by the commonly used web browsers.

For critical applications that can be used via the Internet, use of an extended validation certificate (EV certificate) is recommended.

Motivation: Only if the certificate authority (CA) is contained in the CA list of the browser being used the browser can verify the authenticity of the server.or web application
Stricter issuing criteria apply to EV certificates. If an EV certificate is used, this is visualized in the browser. Even if EV certificates do not improve security, their use increases the trustworthiness of the server for the user.

For this requirement the following threats are relevant:
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.03-25/6.0

Req 16	Certificates must lose their validity after a maximum of 1 year.

In the case of certificates of an internal CA, in particular for machine interfaces, the period may be extended to a maximum of 3 years.

Motivation: The methods used for analysing and breaking cryptographic processes are improved continuously. Therefore the security of the certificates can be ensured for a limited period only. But, according to a general estimation, the security of the certificates is ensured for the required validity period of one year, if an appropriate key length is used.

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

For this requirement the following warranty objectives are relevant:

ID: 3.03-26/6.0

Req 17	Certificates must have a key length of at least 3072 bits when using RSA or 256 bits when using ECC.

Remarks on DSA and RSA certificates:
For DSA and RSA, key lengths smaller than 3000 bits may only be used in legacy systems [BSI TR-02102-1] until the end of 2025 and
should be substituted at the next opportunity. Because of the better performance, elliptic curve (EC-DSA) certificates shall be preferred (if supported and technically doable).
RSA-PKCS#1 v1.5 may only be used in legacy systems and should be (if feasible) substituted at the earliest opportunity [BSI TR-02102-1].

References:
[BSI TR-02102-1] Bundesamt für Sicherheit in der Informationstechnik: Cryptographic Mechanisms: Recommendations and Key Lengths, TR-02102-1, Version 2022-01, 28.01.2022

Motivation: In order to guarantee the security of certificates over the validity period, the cryptographic keys must have an appropriate length. According to a general estimation, a key length of 3072 bits provides sufficient protection for the next years. For ECC algorithms, shorter key lengths already provide the same level of security.

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

For this requirement the following warranty objectives are relevant:

ID: 3.03-27/6.0

5. Nginx webserver logging

Req 18 Access to the web server must be logged.

The logging format must include the following information:

- Access timestamp
- Source (IP address)
- User (if known)
- URL
- Status code of the response from the web server

The applicable statutory, rate-plan and commercial provisions must be taken into account during logging. These provisions state, among other things, that event logging must only take place for a specific purpose. The logging of events to monitor employee work is not permitted.

Motivation: To analyze security incidents, it is very important to have basic information on how an attack was conducted. Since a web server constitutes an external interface, certain information exists solely on the web server even with an attack on a downstream system and must therefore be logged on the web server.

Implementation example:

```
...
#
# nginx.conf
#
http {
log_format compression '$remote_addr - $remote_user
[$time_local] ' '"$request" $status $body_bytes_sent '
'"$http_referer" "$http_user_agent" "$gzip_ratio"';
server {
      gzip on;
      access_log /var/log/nginx-access.log compression;
      # only log critical errors
      error_log /var/log/nginx/error.log crit;
...

      }
}
```

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: 3.80-18/5.0

Req 19 The system clock must be synchronized to an accurate reference time (Time Standard).

A time reference source must be used which provides a time signal based on the Coordinated Universal Time ("UTC" = "**U**niversal **T**ime **C**oordinated").

Please Note: The UTC-synchronized system time may be transformed to local time using a corresponding timezone configuration setup for any output of time information, as long as this timezone adjustment is fully accountable.

Systems belonging to the same security domain must synchronize to one and the same time reference source.

Motivation: Reference time synchronization may be a technical prerequisite for many time-dependent mechanisms, for example: Validation of Certificates; Authentication. It is also much-needed to generate exact timestamps for logged events, since without the often required time-related correlation in case of a Security Incident or during a Problem Analysis cannot be achieved.

Implementation example: some valid time reference sources:

- trustworthy NTP ("**N**etwork**T**ime**P**rotocol") Server on the IP network
- DCF77 radio signal received via a physically connected receiver
- GPS radio signal received via a physically connected receiver

For this requirement the following threats are relevant:

- Disruption of availability
- Denial of executed activities
- Unnoticeable feasible attacks

For this requirement the following warranty objectives are relevant:

ID: 3.01-32/7.0

Req 20	Applicable retention and deletion periods must be observed for security-relevant logging data that is recorded locally.

From an IT security perspective, local storage of security-relevant logging data on a system is not mandatory. Since the local storage can be damaged in the event of system malfunctions or manipulated by a successful attacker, it can only be used to a limited extent for security-related or forensic analyses. Accordingly, it is relevant for IT security that logging data is forwarded to a separate log server.

Local storage can nevertheless take place; for example, if local storage is initially indispensable when generating the logging data due to technical processes or if there are justified operational interests in also keeping logging data available locally.

The following basic rules must be taken into account when storing logging data locally:

- Security-related logging data must be retained for a period of 90 days.
  (*This requirement only applies if no additional forwarding to a separate log server is implemented on the system and the logging data is therefore only recorded locally.*)
- After 90 days, stored logging data must be deleted immediately.

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 (DPA) or are specified by them.

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 locally stored security-relevant logging data are implemented on an exemplary telecommunications system:

- 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: 3.01-34/7.0

Req 21	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

6. Technical Baseline Security for IT/NT Systems

6.1. Technical and organizational requirements for the use of components in the system

Req 22	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

6.2. Protecting data and information

Req 23	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 meas-

ures 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

6.3. Authentication attribute "password"

Req 24	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

6.4. Logging

Req 25	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