Security requirement

Tomcat Application Servers

Deutsche Telekom Group

Version	6.1
Date	Jul 1, 2024
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
	3.39	Security requirement
Version	**State**	**Status**
6.1	Jul 1, 2024	Released
Contact	**Validity**	**Released by**
Telekom Security psa.telekom.de	Jul 1, 2024 - Jun 30, 2029	Stefan Pütz, Leiter SEC-T-TST

Summary
This security document has been prepared based on the general security policies of the Group and defines the requirements for securely implementing Tomcat application servers. The requirements described in this document shall be met to ensure that a tomcat application server cannot be easily misused by attackers.

Copyright © 2024 by Deutsche Telekom AG.
All rights reserved.

Table of Contents

1. Introduction 4

2. Tomcat Application Server 5

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.

This document assumes, that a web server is running in front of the application server. If this is not the case, then the security requirements on web servers must be fulfilled by the application server or another component, which receives http requests before the application server. A load balancer could be such component.

The examples are valid for Tomcat version 9.0.

2. Tomcat Application Server

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

ID: 3.01-1/8.0

Req 2	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.

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

ID: 3.01-2/8.0

Req 3	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

ID: 3.01-10/8.0

Req 4	The application server must run under a dedicated (operating-system) account that only has the permissions required for operation.

The account must not have administrator permissions. If the application server needs to be launched with administrator permissions, then it has to be switched to another account once launched. This is comparable to normal web server situations.

Motivation: Every service that runs with administrator permissions constitutes a higher risk in respect of possible vulnerabilities or misuse.

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

ID: 3.10-9/6.1

Req 5	Unused services and protocols must be deactivated.

Some application servers provide multiple interfaces for requests to the applications. Usually only one of each interface is used. All interfaces and their connected services, which are not required, must be deactivated.

Motivation: Services and protocols that are not required for system operation increase the potential attack surface and thus the risk of the system being compromised. This risk is further increased by the fact that a security inspection and an appropriate optimisation of the configuration for unused services and protocols will not be done.

Implementation example: Connectors, that are not used, are deleted in the file $CATALINA_HOME/conf/server.xml.

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

ID: 3.39-5/6.1

Req 6	Stored data in need of protection must be protected against unauthorized access, modification and deletion.

In particular, data used for the configuration and administration of the application server needs protection. The storage of data must use encryption or file access restrictions.

Motivation: If data with a need of protection is not secured, an attacker could, as an example, access passwords, which are stored unencrypted or without proper access restrictions.

Implementation example: On a system, the configuration files of the application server can only be written by the legit-

imate 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 application server.

ID: 3.10-6/6.1

Req 7	If the shutdown port is not needed, it must be deactivated.

Motivation: Via the shutdown port (default 8005) it is possible to terminate all web applications that are running under Tomcat at that moment, and thus provoke a DoS scenario. This option is generally not required and should therefore not be available.

Implementation example: Deactivation of the shutdown port:
In the file CATALINA_HOME/conf/server.xml, the shutdown port is set to -1 and thus deactivated:

```
<Server port="-1" shutdown="SHUTDOWN">
```

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

ID: 3.39-7/6.1

Req 8	If the shutdown port is needed, it must be secured by a strong password.

Motivation: Via the shutdown port (default 8005) it is possible to terminate all web applications that are running under Tomcat at that moment, and thus provoke a DoS scenario. This option must be secured against unauthorized usage.

Implementation example: Protection of the shutdown port by configuring a complex shutdown command:
In the file CATALINA_HOME/conf/server.xml, the shutdown command is filled with a complex value, thus making unauthorized usage more difficult:

```
<Server port="8005" shutdown="Rea!!yC0mplexW0rd">
```

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

ID: 3.39-8/6.1

Req 9	Outputs and messages must not disclose information on internal structures of the system.

Information about the internal structures of a system, including the components used there, and corresponding implementation details are generally considered to be in need of protection.

In general, this concerns information on

- Product names and product identifiers of implemented system components
- Operating systems, middleware, backend software, software libraries and internal applications as well as their software versions
- installed service packs, patches, hotfixes
- Serial numbers of components as well as stored product licenses
- Database Structures

Typical examples of outputs and messages in which disclosure of such system information can potentially occur:

- Login windows and dialogs
- Error messages
- Status messages
- Banners of active network services

- System logs and log files
- Debug logs, stack traces

As far as it is technically feasible without impairing the function and operation of the system, the output of affected system information must always be deactivated.

Access to affected system information must only be possible for authorized users of the system. As a rule, this circle of authorized users is to be limited to administrators and operators of the system. Access for authorized monitoring and inventory systems within the operating environment is also permitted.

A permissible exception to these restrictions exists for specific individual system information, the disclosure of which is technically mandatory for the intended function of the system in conjunction with third-party systems; For example, the presentation of supported protocols and their versions during the initial parameter negotiation in session setups between a client and a server.

Motivation: Information about the internal structures of a system can be used by an attacker to prepare attacks on the system extremely effective. For example, an attacker can derive any known vulnerabilities of a product from the software version in order to exploit them specifically during the attack on the system.

Implementation example: [Example 1]
Deactivation of the display of the product name and the installed version of a Web server in its delivered error web pages.

[Example 2]
Removal of the product name and the corresponding version string from the login banner of a deployed SSH server.

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

ID: 3.01-9/8.0

Req 10	Sample applications and unnecessary standard tools must be deleted.

The Tomcat application server includes samples and administration tools in the standard configuration.

Motivation: Sample applications and administration tools may provide points of attack.

Implementation example: Delete all unnecessary directories below
`CATALINA_HOME/webapps`

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

ID: 3.39-10/6.1

Req 11	If the “manager” application is used, this must be protected against unauthorized use.

To use the “manager” application securely, it is mandatory to set up an administration account and to configure the access via https.
Especially in automated roll-outs ensure that a password is only used in one configuration.

Motivation: Without sufficient access protection, the “manager” application can be misused.

Implementation example: The protection is done in a few steps:

1. Create a hash of the password for the admin account, e.g.

```
./digest.sh -a SHA-256 -i 1 -s 18 -h
org.apache.catalina.realm.MessageDigestCredentialHandler
<password>
```

2. Configure the hash algorithm in the file CATALINA_HOME/conf/tomcat-users.xml, e.g.

```
 <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
              resourceName="UserDatabase" >
              <CredentialHandler class-
Name="org.apache.catalina.realm.MessageDigestCredentialHandler"
                        algorithm="SHA-256" iterations="1" sal-
tLength="18" />
```

3. Define administrator accounts in the file CATALINA_HOME/conf/tomcat-users.xml, e.g.

```
<user username="admin1" password="<created hash>"
roles="manager-status,manager-gui"/>
```

The name for the administrator and the password must be suitably replaced.

4. Configuration of an https connector: The setting up of an https connector is described in the Tomcat documentation ("SSL Configuration HOW-TO").

5. Redirect the manager app to https:
https is redirected by adjusting the file CATALINA_HOME/webapps/manager/WEB-INF/web.xml in the <security-constraint> tags of all <web-resource-name> entries. Here three lines are added under the </auth-constraint> in each case:

```
<security-constraint>
  …
  </auth-constraint>
  <user-data-constraint>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
</security-constraint>
```

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

ID: 3.39-11/6.1

Req 12	If a password is used as an authentication attribute, it must have at least 12 characters and contain three of the following categories: lower-case letters, upper-case letters, digits and special characters.

A system may only accept passwords that comply with the following complexity rules:

- Minimum length of 12 characters.
- Comprising at least three of the following four character categories:
    - lower-case letters
    - upper-case letters
    - digits
    - special characters

The usable maximum length of passwords shall not be limited to less then 25 characters. This will provide more freedom to End Users when composing individual memorizable passwords and helps to prevent undesired behavior in password handling.

When a password is assigned, the system must ensure that the password meets these policies. This must be prefer-

ably enforced by technical measures; if such cannot be implemented, organizational measures must be established. If a central system is used for user authentication [see also Root Security Requirements Document[i] "3.69 IAM (Identity Access Management) - Framework"], it is valid to forward or delegate this task to that central system.

Permissible deviation in the password minimum length
Under security-related criteria, conditions can potentially be identified for a system that enable the minimum password length to be reduced:

- It is generally permissible to reduce the minimum password length for systems that use additional independent authentication attributes within the authentication process in addition to the password (implementation of 2-Factor or Multi-Factor Authentication).
- Any reduction in the minimum password length must be assessed individually by a technical security advisor (e. g. a PSM from Telekom Security) and confirmed as permissible. In the assessment, the surrounding technical, organizational and legal framework parameters must be taken into account, as well as the system-specific protection requirements and the potential amount of damage in the event of security incidents.
- The absolute minimum value of 8 characters length for passwords must not be undercut.

Motivation: Passwords with the above complexity offer contemporary robustness against attacks coupled with acceptable user friendliness. Passwords with this level of complexity have proven their efficiency in practice. Trivial and short passwords are susceptible to brute force and dictionary attacks and are therefore easy for attackers to determine. Once a password has been ascertained it can be used by an attacker for unauthorized access to the system and the data on 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
- Denial of executed activities

ID: 3.01-26/8.0

Req 13	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

ID: 3.01-27/8.0

Req 14	Access to the application server must only be possible from approved IP addresses (IP whitelisting).

Motivation: In most cases the application server is only accessed from certain, i.e., known web servers. The restriction to selected source addresses prevents possible misuse from other computers and substantially reduces the risk of an attack.

Implementation example: The allowed IP addresses can be defined for each application separately.
This is done in the file $CATALINA_HOME/webapps/<application>/META-INF/context.xml.
In the example the access is restricted to localhost for both, the IPv4 and the IPv6 address.

```
<Context ...>
  ...
  <Valve className="org.apache.catalina.valves.RemoteAddrValve"
allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" />
</Context>
```

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

ID: 3.39-14/6.1

Req 15	Access to the application server must be logged.

The application server log must contain the following information:
- Access timestamp
- Source (IP address)
- Account (if known)
- URL
- http status code of application server response

Logging must be done considering the currently valid legal, wage and company regulations. This regulations state among others that logging of events can be done only earmarked. Logging of events for doing a work control of employees is not allowed.

Motivation: For the analysis of security incidents it is very important to have basic information on how the attack has been carried out.

For this requirement the following threats are relevant:
- Unnoticeable feasible attacks

ID: 3.10-14/6.1

Req 16	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

ID: 3.01-34/8.0

Req 17	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

ID: 3.01-36/8.0

Req 18	If the application server provides a login to an admin interface, it must provide logging data about unsuccessful login attempts to a security incident detection and response solution in near-real-time

An acceptable detection and response solution for the purposes of this requirement must meet the following criteria:

- There is a continuous analysis and correlation of the logging data provided by connected IT/NT systems.
- The identification of threatening activities and ongoing attacks takes place on the basis of a continuously updated detection pattern database
- Integrated alerting processes trigger the initiation of countermeasures in the event of a find.

Such a solution is not the same as a log server in terms of its function. Accordingly, the provision to a detection and response solution must always be in addition to an existing forwarding to a log server. An exception exists if the log server used provides the logging data to a detection and response solution immediately after receipt. In this case, it is sufficient for the system to forward the logging data only to the log server.
The logging data must be provided to the detection and response solution within 5 minutes ("near-real-time") from the time it is generated in the system.

Motivation: IT/NT systems are constantly exposed to security threats and attacks. By means of a central analysis and the correlation of recorded system events of entire IT landscapes in almost real time, current threats and ongoing attacks can be detected even beyond system boundaries and suitable countermeasures can be initiated promptly. The prompt delivery of logging data to the detection and response solution is essential for its effectiveness.

Implementation example: The application server is configured to log each failed login event. By sending this logging data to a detection and response solution in parallel, which can detect in real time that an attack is evidently taking place, alert it and thus enable immediate countermeasures.

For this requirement the following threats are relevant:

- Unauthorized access to the system
- Unnoticeable feasible attacks

ID: 3.10-17/6.1