

Security requirement

PostgreSQL databases

Deutsche Telekom Group

Version	10.0
Date	Dec 1, 2025
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
	3.60	Security requirement
Version	State	Status
10.0	Dec 1, 2025	Released
Contact	Validity	Released by
Telekom Security psa.telekom.de	Dec 1, 2025 - Nov 30, 2030	Stefan Pütz, Leiter SEC-T-TST

Summary
This paper describes all requirements on security issues related to PostgreSQL.

Copyright © 2025 by Deutsche Telekom AG.
All rights reserved.

Table of Contents

1.	Introduction	4
2.	General	5
3.	Directory and File Permissions	6
4.	Secure transmission of passwords	7
5.	Principle of least privilege and separation of privileges	10
6.	SQL extensions with operating system or network access	14
7.	Data communication	16
8.	Logging and Monitoring	18
9.	Technical Baseline Security for IT/NT Systems	24
9.1.	Authentication and authorization	24

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. General

Req 1 In the production environment, community-supported or commercially supported version of the PostgreSQL software must be used.

Motivation: Developers doesn't provide security fixes for PostgreSQL versions which are in EOL (End Of Life) stage.

Implementation example: The exact version number can be determined using the following command:

`SELECT version();`

The client version can be checked using the following command:

`psql -V`

The version numbers can then be checked against the vendor information published on the following website:

<https://www.postgresql.org/support/versioning/>

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

ID: 3.60-1/10.0

Req 2 Multiple PostgreSQL instances may only be operated together on one operating system instance (with physical or virtualized hardware) if they have the same low protection requirements, are under a single customer administrative authority and are operated, in terms of administration by the same group of people.

Motivation: Operating multiple database instances together increases the risk that users will gain unauthorized access to instances for which they are neither technically nor administratively responsible.

Implementation example: To separate the individual database instances, administrators can use physical server hardware or virtualization solutions.

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

ID: 3.60-2/10.0

3. Directory and File Permissions

Req 3 A database service must not run with root or administrative rights.

Motivation: If a database server runs with highly privileged operating-system rights (root, administrator), an attacker can completely take over a database system by exploiting a vulnerability (access to the file system or execution of files).

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

ID: 3.16-18/9.0

Req 4 The PostgreSQL "data_directory" and config files must be assigned exclusively to the database systems operating-system account (such as "postgres"). Other systems users must not have access to the "data_directory" and config files.

The permissions of the folders must be set as follows:

Folder path	Folder permission
'data_directory'	'700:postgres:postgres'
'data_directory'/initdb.log	'600:postgres:postgres'
'data_directory'/'other_subdirectories'	'700:postgres:postgres'

In addition, the umask of the postgres user must be set to "0077" so that new files are automatically created with appropriately restrictive permissions.

Motivation: If file permissions on data and config files are not properly defined, other users may read, modify or delete those files.

For example: it may be possible to tamper log files or copy whole database with all data inside.

Implementation example: For PostgreSQL 15, the folder permissions can be checked with the following command:
`ls -la ~postgres/15`

If necessary, the permissions can be adjusted with the Linux commands "chmod" and "chown".

The current user's umask can be checked using the "umask" command.

Depending upon the postgres user's environment, the umask of the postgres user can be set in their ".bash_profile", ".profile" or ".bashrc" files using the "umask 0077" entry.

For this requirement the following threats are relevant:

- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources

ID: 3.60-4/10.0

4. Secure transmission of passwords

Req 5	If Windows is used as operating system, default passwords must be changed.
-------	--

Motivation: Exploitation of default (easy guessable) passwords is easier way to hack into system.

Implementation example: Especially with the "postgres" account, it must be ensured that the password is changed.

The following command can be used to change passwords:

```
#\password <username>
```

This statement will not be logged in the logfile itself.

For this requirement the following threats are relevant:

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

ID: 3.60-5/10.0

Req 6	Passwords in PostgreSQL must be stored and transmitted hashed using a suitable and approved "password hashing" method.
-------	--

This requirement can be met by using the "scram-sha-256" authentication method.

The authentication method "md5" does not meet this requirement.

Motivation: The use of hashes (both in storage and in transmission) increases security significantly.

Implementation example: In the file "postgresql.conf" the parameter "password_encryption" has to be set to "scram-sha-256":

```
password_encryption = scram-sha-256
```

Furthermore, the relevant entries must be configured in the "pg_hba.conf" file according to the following examples:

```
# TYPE DATABASE USER ADDRESS METHOD
# "local" is for Unix domain socket connections only
local database_1 user_1 scram-sha-256
```

```
# IPv4 local connections:
host database_2 user_2 127.0.0.1/32 scram-sha-256
```

```
# IPv6 local connections:
host database_2 user_3 ::1/128 scram-sha-256
```

```
# IPv4 remote connections:
hostssl database_1 user_4 10.23.24.123/32 scram-sha-256
```

To ensure encrypted password storage, the following command can be used:

```
CREATE USER user_1 PASSWORD '<password>';
```

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

ID: 3.60-6/10.0

Req 7	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 than 25 characters. This will provide more freedom to End Users when composing individual memorable passwords and helps to prevent undesired behavior in password handling.

However, PostgreSQL does not natively provide a secure function to check compliance with password rules. Therefore, compliance must be ensured either through organizational measures or by using external authentication methods.

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.

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.

ID: 3.60-7/10.0

Req 8	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: 3.01-27/9.0

Req 9	If a password is used as an authentication attribute, it must be changed after 12 months at the latest.
-------	---

The maximum permitted usage period for passwords is 12 months.

If a password reaches the maximum permitted usage period, it must be changed.

For this purpose, the system must automatically inform the user about the expired usage period the next time he logs on to the system and immediately guide him through a dialog to change the password. Access to the system must no longer be permitted without a successfully completed password change.

For technical user accounts (M2M or Machine-2-Machine), which are used for the authentication and authorization of systems among themselves or by applications on a system, automated solutions must also be implemented to comply with the permitted usage period for passwords.

Alternatively, if such an automatic mapping of the process for changing the password cannot be implemented, an effective organizational measure must be applied instead, which ensures a binding manual password change at the end of the permissible period of use.

Motivation: Unlike more modern authentication attributes, passwords are easier to attack. Without specific measures for reliable, technically automated detection of compromises, the risk of a password being discovered or broken by an attacker can increase considerably over time.

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
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

ID: 3.01-30/9.0

5. Principle of least privilege and separation of privileges

Req 10 Only the Database Administrator must have SUPERUSER, CREATEROLE or CREATEDB privileges.

Motivation: Granting extensive privileges to ordinary users can cause various security problems, such as: intentional/unintentional access, modification or destroy of data.

Implementation example: The following command can be used to determine roles with corresponding privileges:
`SELECT rolname,rolsuper,rolcreatorole,rolcreatedb FROM pg_roles WHERE rolsuper IS TRUE OR rolcreatorole IS TRUE or rolcreatedb IS TRUE;`

The following commands can be used to remove the corresponding privileges:

```
ALTER ROLE role_name WITH NOSUPERUSER;  
ALTER ROLE role_name WITH NOCREATOROLE;  
ALTER ROLE role_name WITH NOCREATEDB;
```

For this requirement the following threats are relevant:

- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources

ID: 3.60-10/10.0

Req 11 Only the Database Administrator must have privileges on pg_catalog.pg_authid table.

Motivation: In pg_catalog.pg_authid table there are stored credentials such as username and password. If hacker has access to the table, then he can extract these credentials.

Implementation example: The following command can be used to determine users with privileges on pg_catalog.pg_authid table:
`\dp pg_catalog.pg_authid`

The following command can be used to remove all privileges on pg_catalog.pg_authid:
`REVOKE ALL PRIVILEGES ON pg_catalog.pg_authid FROM user_name;`

For this requirement the following threats are relevant:

- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources

ID: 3.60-11/10.0

Req 12 The "trust" authentication method must NOT be used.

Motivation: This method allows a login from a host without knowledge of a respective password.

Implementation example: From "pg_hba.conf" file comment out every entry with "trust" ending.

For this requirement the following threats are relevant:

- Unauthorized modification of data
- Unauthorized use of services or resources
- Unnoticeable feasible attacks

ID: 3.60-12/10.0

Req 13	The authentication method "peer authentication" may only be used for administrative or technical accounts.
--------	--

If unavoidable (e.g. for technical accounts and e.g. maintenance purposes), this feature should be used very carefully.

Motivation: Administrative actions must be protected by strong authentication mechanisms.

Implementation example: Commenting out or changing the corresponding entries in the "pg_hba.conf" file from "peer" to another authentication method (e.g. scram-sha-256) satisfies the requirement.

For this requirement the following threats are relevant:

- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources

ID: 3.60-13/10.0

Req 14	The default "CREATE" permission on the "public" schema must be revoked from all users in all databases.
--------	---

The default settings of PostgreSQL versions prior to version 15 allow users to change the behavior of queries for other users by creating like-named objects in the "public" schema (see CVE-2018-1058).

Note: Upgrading from a previous version does not automatically fix the CVE-2018-1058 vulnerability.

Motivation: Revoking the "CREATE" permission on the "public" schema prevents a user from exploiting the vulnerability described in CVE-2018-1058.

Implementation example: The following command can be used to remove the default "CREATE" permissions for all users in a database:

```
REVOKE CREATE ON SCHEMA public FROM PUBLIC;
```

The following command can be used to check in a database whether users other than "postgres" have "CREATE" permissions on the "public" schema:

```
SELECT u.username, pg_catalog.has_schema_privilege(u.username, 'public', 'CREATE') AS "create"
FROM pg_user u
WHERE u.username != 'postgres' AND pg_catalog.has_schema_privilege(u.username, 'public', 'CREATE') = 't';
```

The following commands can be used to check which functions have been created in the "public" schema:
`\df public.*`

For this requirement the following threats are relevant:

- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources

ID: 3.60-14/10.0

Req 15	Privileges on database objects must not be granted to "PUBLIC".
--------	---

It should be noted that PostgreSQL grants the following default privileges to "PUBLIC" when objects are created:

- "CONNECT" and "TEMPORARY" privileges for databases,
- "EXECUTE" privilege for functions and procedures; and
- "USAGE" privilege for languages and data types (including domains).

These default privileges must be revoked after creating the object.

The default privilege settings can be changed using the "ALTER DEFAULT PRIVILEGES" command.

Motivation: By granting privileges on database objects to "PUBLIC", all roles/users (including those created in the future) receive these permissions. This can result in users being granted unauthorized privileges that could be misused (see e.g. CVE-2018-1058 and CVE-2020-14349).

Implementation example: The following commands can be used to check whether permissions on non-default objects have been granted to "PUBLIC":

For databases:

\l

For entries beginning with "=", the named permissions were given to "PUBLIC".

For tables:

```
SELECT * FROM information_schema.table_privileges WHERE grantee = 'PUBLIC' AND table_schema != 'pg_catalog' AND table_schema != 'information_schema';
```

For functions and procedures:

```
SELECT * FROM information_schema.routine_privileges WHERE grantee = 'PUBLIC' AND specific_schema != 'pg_catalog' AND specific_schema != 'information_schema';
```

For languages and data types:

```
SELECT * FROM information_schema.usage_privileges WHERE grantee = 'PUBLIC' AND object_schema != 'pg_catalog' AND object_schema != 'information_schema';
```

Permissions granted to "public" can be revoked with the following command:

```
REVOKE ALL ON <object> "<object_name>" FROM PUBLIC;
```

„<object>“ and „<object_name>“ must be replaced with the appropriate object type and object name.

For this requirement the following threats are relevant:

- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources

ID: 3.60-15/10.0

Req 16	Privileges on objects must not be granted with "WITH GRANT OPTION".
--------	---

Motivation: If privileges have been granted with "WITH GRANT OPTION", the recipient can also grant those privileges with "GRANT" to other users. This can result in violation of segregation of privileges principle.

Implementation example: In the "psql" client, the following command can be used to identify such privileges (privileges with "WITH GRANT OPTION" are displayed there with "**"):

\dp

The following command can be used to remove these privileges:

```
REVOKE GRANT OPTION FOR ALL ON table_name FROM user_name;
```

For this requirement the following threats are relevant:

- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unnoticeable feasible attacks

ID: 3.60-16/10.0

Req 17 SQL functions with SECURITY DEFINER must be used in a restrictive manner.

Motivation: The SECURITY DEFINER property of functions is similar to the "setuid" feature in Linux Operating Systems. This property allows users to execute functions with the privileges of the owner of the functions rather than with the privileges of the user invoking the function.

Implementation example: The following query returns functions with SECURITY DEFINER:

```
SELECT pg_proc.proname, pg_namespace.nsname, pg_user.username FROM pg_proc JOIN pg_namespace ON  
pg_proc.pronamespace=pg_namespace.oid JOIN pg_user ON pg_proc.proowner=pg_user.usesysid WHERE pro-  
secdef='t';
```

The following command deletes individual functions:

```
DROP FUNCTION function_name(param_1 TEXT, param_2 TEXT);
```

For this requirement the following threats are relevant:

- Unauthorized access or tapping of data
- Unauthorized modification of data

ID: 3.60-17/10.0

6. SQL extensions with operating system or network access

Req 18 The "TRUSTED" flag may only be set for procedural languages if they do not allow access to the internals of the database server or the file system.

By setting the "TRUSTED" flag, even unprivileged database users can use the procedural language to create functions and procedures.

Since procedural languages functions are executed inside the database server and thus with its privileges, the "TRUSTED" flag should only be given for languages that do not allow access to database server internals or the file system.

By default, PostgreSQL offers the procedural languages "SQL" and "PL/pgSQL" with the flag "TRUSTED". In addition, the languages "PL/TCL" and "PL/Perl" are also considered trustworthy and can be used if necessary. For other languages, the "TRUSTED" flag must not be set.

Motivation: Procedural languages that have the "TRUSTED" flag set can be used by all database users to create functions and procedures. If a non-restricted language has been marked as trusted, an unprivileged user can use that language to access data that they would otherwise not have access to.

Implementation example: The following command can be used to determine all languages that have the "TRUSTED" flag set:

```
SELECT lanname AS language, lanpltrusted AS trusted FROM pg_language WHERE lanpltrusted = 't';
```

If the result contains non-restricted languages, either the value of „lanpltrusted" must be changed to "f" or the language must be removed.

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

ID: 3.60-18/10.0

Req 19 Only "C" and "Internal" may be used as untrusted procedural languages.

Procedural languages that have been marked as "untrusted" can only be used by database superusers to create functions. In general, procedural languages should always be installed in "untrusted" mode if it cannot be guaranteed that the languages do not grant access to data to which the user would otherwise not have access.

The procedural languages "C" and "Internal" are offered as "untrusted" by default in PostgreSQL. If other "untrusted" procedural languages are needed, their use must be described and justified.

Motivation: Procedural languages installed in "untrusted" mode can be used to gain access to the database server process or to gain operating system level access. To reduce the attack surface, the number of procedural languages should be kept to a minimum.

Implementation example: The following command can be used to determine all languages marked as "untrusted":

```
SELECT lanname AS language, lanpltrusted AS trusted FROM pg_language WHERE lanpltrusted = 'f';
```

The following command removes a not required language:

```
DROP LANGUAGE proclang_name;
```

For this requirement the following threats are relevant:

- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources

- Unnoticeable feasible attacks

ID: 3.60-19/10.0

Req 20	Procedural languages which are not needed must be removed.
--------	--

The following procedural languages can remain as they are standards:

- internal
- c
- sql
- plpgsql

Motivation: Every additional procedural language that is not needed unnecessarily increases the attack surface.

Implementation example: The following command lists all available procedural languages:

```
SELECT lanname AS language FROM pg_language;
```

Unneeded procedural languages can be removed with the following command:

```
DROP LANGUAGE plang_name;
```

For this requirement the following threats are relevant:

- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unnoticeable feasible attacks

ID: 3.60-20/10.0

7. Data communication

Req 21	Each client connection to the database server must be implemented using a separate entry in <code>pg_hba.conf</code> . Only one database, one database user and an IP address range that is as restrictive as possible may be defined for each connection entry.
--------	--

When defining the connection entries, specifying individual IP addresses is preferable to specifying an IP address range. If this is not possible, the IP address range must be defined in such a restrictive manner that it only contains clients that require the respective access.

Exception: In the connection entry for the database administrator, the value "all" may be used in the "database" field.

Alternatively, this requirement can be implemented with other suitable filter elements (e. g. using an appropriately configured "connection pooler"). In this case, the used alternative must be named.

Motivation: This allows for privilege separation and therefore reduction of risk of misuse

Implementation example: # Connection entry 1
hostssl database_1 user_1 10.23.24.123/32 scram-sha-256
Connection entry 2
hostssl database_1 user_2 10.23.24.11/32 scram-sha-256
Connection entry 3
hostssl database_2 user_1 10.23.24.123/32 scram-sha-256

For this requirement the following threats are relevant:

- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unnoticeable feasible attacks

ID: 3.60-21/10.0

Req 22	Transport Layer Security (TLS/SSL) must be enabled in the PostgreSQL configuration and configured according to requirements document 3.50 "Cryptographic Algorithms and Security Protocols".
--------	--

Motivation: If TLS/SSL is not enabled and properly configured on the server, the risk of the transmitted data being compromised increases.

Implementation example: In the file "postgresql.conf" the following parameters must be set:

```
# Changes require restart!
# Enables SSL connections.
ssl = on
# Specifies the file containing the SSL server certificate.
ssl_cert_file = '/path/to/server.crt'
# Specifies the file containing the SSL server private key.
ssl_key_file = '/path/to/server.key'
# Specifies allowed SSL cipher suites
ssl_ciphers = 'EECDH+AESGCM:EDH+AESGCM'
# Sets the minimum allowed SSL/TLS protocol version.
ssl_min_protocol_version = 'TLSv1.2'
```

```
password_encryption = scram-sha-256
```

In addition, each entry for remote connections in the "pg_hba.conf" file must start with the "hostssl" prefix, as shown in the following example:

```
hostssl database_1 user_1 10.23.24.123/32 scram-sha-256
hostssl database_1 user_2 10.23.24.11/32 scram-sha-256
```

For this requirement the following threats are relevant:

- Unauthorized access or tapping of data

ID: 3.60-22/10.0

8. Logging and Monitoring

Req 23 Logging functions must be turned on and properly configured according / compliant to local law.

Motivation: If logging features are not enabled, user activity in the database cannot be tracked, so diagnosis of security incidents may be limited or impossible.

Implementation example: The following entries in the "postgresql.conf" file meet the technical requirements:

```
log_destination = 'syslog'
logging_collector = on
log_directory = '/path/to/log_directory'
log_truncate_on_rotation = on
log_connections = on
log_disconnections = on
log_duration = on
log_error_verbosity = default
log_hostname = on
log_line_prefix = 'db=%d,user=%u,app=%a,client=%h '
log_statement = 'ddl'
```

If "log_destination" contains a value other than "syslog", "log_line_prefix" must contain the following values:

```
log_line_prefix = '%m [%p]: [%l-1] db=%d,user=%u,app=%a,client=%h '
```

The parameter "log_line_prefix" can be extended by additional values upon need.

For this requirement the following threats are relevant:

- Denial of executed activities
- Unnoticeable feasible attacks
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

ID: 3.60-23/10.0

Req 24 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: 3.01-34/9.0

Req 25	Security-relevant logging data must be forwarded to a log server immediately after it has been generated.
--------	---

The system's security-relevant logging data must be forwarded to a log server immediately after it has been generated. For the purposes of the requirement, the log server must meet the following criteria:

- The log server must be dedicated to the storage of logging data. It must not depict any other functions or services that are not related to this task. A permissible exception is the combination of a log server with a SIEM (Security Information and Event Management) solution.
- The log server must be separate from the system. It must not be operated directly on the system as an additional module.
- The log server must be centrally available for the operational environment, to which the system belongs.

To transmit the logging data to the log server, standard protocols specialized for this purpose (e.g. Syslog or SNMPv3) should preferably be used

Motivation: If logging data is stored only locally, it can be manipulated in the event of a successful compromise of the system by an attacker in order to obfuscate their attack and actions taken on the system. For this reason, the forwarding of logging data to a log server must take place very soon after the occurrence of a security-relevant event in the source system. This means that the logging data generated is separated from the source system and is available for analysis at any time, regardless of the source system state. The use of a central log server for an operational environment ensures that the logs of all IT/NT Systems in this operational environment can be accessed quickly and uniformly and that the access itself can be optimally integrated into the incident processes of the environment.

For this requirement the following threats are relevant:

- Unauthorized modification of data
- Disruption of availability
- Denial of executed activities
- Unnoticeable feasible attacks

Req 26	For security-relevant logging data that is forwarded to the 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 log server.
- after 90 days, stored logging data must be deleted immediately on the 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 log server is not within the same operational responsibility as the source system of the logging 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

Req 27	Security-relevant logging data must be provided 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.
- Security experts are actively involved in the operation of the solution as well as the evaluation of alarms

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.

The specific technical interfaces and protocols to be supported for deployment are determined by the detection and response solution available in the system's operating environment as well as the technical basis of the system itself.

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 appropriate countermeasures can be initiated promptly. The prompt delivery of logging data to the detection and response solution is essential for its effectiveness.

Implementation example: **Example 1:** A system based on a classic server collects security-relevant logging data in the operating system's local syslog or Windows Event Log. Logging data is continuously forwarded to a Security Information and Event Management (SIEM) for further analysis, intrusion detection, and alarm handling.

Example 2: An Endpoint Detection and Response (EDR) agent is running on a system. The EDR agent continuously forwards the security-related logging data it collects on the system to the central backend of the EDR solution, where further analysis, intrusion detection, and alarm handling takes place

Example 3: Network components provide events to a SIEM that make it possible to detect suspicious communication behavior such as network scans, unauthorized data flows between network segments, or active connections or connection attempts to suspicious destinations on the Internet. In particular, gateway routers from the CNDTAG provide "Netflow" logs, while forward proxies report access to domains and/or IP addresses and firewalls log blocked connection attempts.

Example 4: Systems designed on the basis of cloud-native functions generate logging data directly from these functions using the cloud's own logging and monitoring infrastructure. In such a system, the generation of security-relevant logging data is specifically activated in the corresponding configuration options of the cloud-native functions used. The resulting logging data is continuously transferred to a SIEM (Security Information and Event Management) for further analysis, intrusion detection and alarm handling. Some cloud providers already provide some attack pattern detection in the cloud's own monitoring infrastructure and generate specific cloud alerts from it. These are also transferred to the SIEM.

Example 5: A system is based on a SaaS ("Software as a Service") application from a cloud provider. Typically, in the SaaS model, the cloud provider is responsible for all operational aspects of the application, including security management. The cloud customer itself has no influence on the underlying technical infrastructure or the operation of the application. Based on contractual agreements, recognized certificates, and documented self-disclosures from the cloud provider, it is formally verified that the cloud provider has implemented a security incident detection and response solution. Some cloud providers also generate dedicated logging data about security-relevant activities within the separate customer tenants of their SaaS application (e.g. changes to configuration settings, permission assignments or user accounts within the tenants) and make them available to cloud customers. These so-called audit logs are continuously transferred to a SIEM (Security Information and Event Management) for further analysis, attack detection and alarm handling.

For this requirement the following threats are relevant:

- Unnoticeable feasible attacks

ID: 3.01-37/9.0

Req 28 Log files from PostgreSQL must be monitored continually for misuse scenarios.

The monitoring can be e.g. done fully automatically only triggering alarms for the operations unit. At any case, the compliance to local laws / workers council should be taken into account.

Motivation: If regular log review is not performed, then there's higher risk that malicious activity has been not discovered.

Implementation example: Log files must be checked for suspicious activity such as attempts to access tables with a need of protection (e.g. "pg_shadow", "pg_authid").

Below example shows how this requirement can be handled:

```
grep -r 'permission denied' /path/to/log_directory/*.log
```

For this requirement the following threats are relevant:

- Unnoticeable feasible attacks

ID: 3.60-28/10.0

Req 29	Important database services and instances must be monitored continually for misuse scenarios.
--------	---

The monitoring of user actions for misuse shall comply with national legislation currently in force (for details see the "Security Requirement on Misuse Detection").

There must be a case study regarding the type of service/user data/number of entries and possible misuse scenarios to define the necessary level of monitoring of the cases. Insider threats

- Anomaly detection
- Accidental exposure
- Misconfiguration
- Vulnerable application
- Machine Language models
- Insecure data in transit
- ...

Motivation: There are many conceivable ways to misuse database systems. Users generate an unusually high data usage rate or operate at unusual times of day. Attackers utilize unusual and critical commands for database queries, as well as tools and malware to extend their rights. To detect misuse, database systems should be continually monitored for misuse scenarios, e.g. by means of database triggers and log monitoring.

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

ID: 3.16-32/9.0

Req 30	Mass data exfiltration must be detected if more than 100K customer data entry and DPC2 or DPC3 is present.
--------	--

Mass data exfiltration detection is a critical control that monitors for unusually large volumes of data being transferred internally or leaving the corporate network or database systems. When more than 100K customer data records - particularly classified as DPC2 or DPC3 - are accessed, copied, or transferred, automated alerts should be triggered. This threshold-based detection should be integrated with SIEM and User and Entity Behavior Analytics (UEBA) tools to identify deviations from normal access patterns. The system should correlate access context (user role, time, data sensitivity, and transfer channel) to distinguish legitimate bulk operations from potential exfiltration attempts. Upon detection, predefined incident response workflows should immediately isolate the activity, notify the security operations center, and preserve forensic evidence for investigation.

Motivation: Customer data are one of the most valuable assets of the company. Therefore we must take every measures to avoid mass data leakage, malicious act or misuse. For that purpose we need to know how we are handling and using our datasets.

Implementation example: Technical guidance and implementation details are also provided on that page and are continuously maintained:

<https://yam-united.telekom.com/pages/psa/apps/wiki/wiki-sicherheitsanforderungen/list>

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

ID: 3.16-33/9.0

9. Technical Baseline Security for IT/NT Systems

9.1. Authentication and authorization

Req 31	The use of system functions that require protection as well as access to internal or confidential data must not be possible without prior authentication and authorization.
--------	---

The use of functions of the system that require protection as well as access to data classified as internal or confidential must only be possible after the user has been uniquely identified and successfully authenticated by means of a user account. A user account consists of a unique user name and associated authentication features. In addition, it must be verified that the user is authorized to access the affected functions and data within the user role assigned to him or her in the system.

An exception to this are functions and data that may be used publicly without restriction; for example, the area of a website on the Internet where only public information is provided.

Examples of features that require prior authentication include:

- Remote access to network services (such as SSH, SFTP, web services)
- Local access to the management console
- Local use of operating system and applications

This requirement also applies without restriction to any machine access to the system (Here, implementation is usually carried out through the use of so-called "technical user accounts", also known as "machine-2-machine" or "M2M" accounts).

Motivation: The unambiguous authentication and authorization of access to a system are elementary to protect functions and data from misuse.

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: 3.01-19/9.0

Req 32	User accounts must be protected with at least two authentication features from different factors.
--------	---

All user accounts of the system must be protected from unauthorized use.

This is to be achieved by using authentication features that allow the legitimate user to be uniquely identified.

Authentication features are differentiated into superordinate groups, so-called "factors":

- Factor KNOWLEDGE: "something that only the legitimate user knows"
- Factor POSSESSION: "something that only the legitimate user possesses"
- Factor INHERENCE: "something that only the legitimate user is"

At least two authentication features must be combined.

These must come from different factors ("Multi-Factor Authentication"; short: "MFA").

A combination of authentication features of the same factor (example: two different PINs) is not allowed.

Examples of acceptable authentication features

- from the KNOWLEDGE factor: PINs, passphrases (passwords*)
- from the POSSESSION factor: smart cards, FIDO 2 Level 2 security keys, device-bound passkeys, cryptographic keys stored in a TPM/HSM, OTPs (One-Time Passwords) generated in an authenticator app that has been approved for use in Deutsche Telekom
- from the INHERENCE factor: biometric features such as fingerprints, iris or retina scans, face recognition

*) Important note on the use of passwords

Passwords must be strictly avoided as an authentication feature.

They are only acceptable if more secure alternatives cannot be implemented.

Authentication features that can be easily forged or manipulated are generally not permitted.

These include:

- Phone numbers and caller IDs
- IP addresses
- Geographical locations
- VPN affiliations and network information
- Status of clients, such as workstation systems or mobile devices

Note on technical user accounts

Technical user accounts used for machine access (also known as "machine-to-machine" or "M2M" accounts) are exempt from the obligation to use multi-factor authentication, as this is usually not feasible there.

Motivation: User accounts that are not protected by appropriate authentication features can be abused by an attacker to gain unauthorized access to a system and the data and applications stored on it. Due to a variety of common attack methods, the protection of user accounts by a single authentication feature is now considered weak, especially if the classic password is used here. A combination of authentication features from different factors significantly increases security for user accounts.

Implementation example: Examples of MFA

MFA Combination	Description	Application Note
Telekom "myCard" plus PIN	1st Factor: Certificate implemented in hardware ("Possession") 2nd Factor: PIN ("Knowledge")	(none)
FIDO2 Level 2 Security Key plus fingerprint	1st Factor: Crypto key stored in hardware ("Possession") 2nd Factor: Biometric feature ("Inherence")	Biometric feature must be pre-configured, otherwise the biometric sensor will only work as a "button" and does not provide the 2nd factor ("Inherence").
Device-bound Passkey in WH4B* on Nucleus PC plus PIN	1st Factor: Crypto key stored in hardware ("Possession") 2nd Factor: PIN ("Knowledge")	(none)
Device-bound Passkey in Secure Enclave on iOS Device plus FaceID	1st Factor: Crypto key stored in hardware ("Possession") 2nd Factor: Biometric feature ("Inherence")	(none)

Password plus OTP App approved for use (e.g. MS Authenticator)	1st Factor: Password ("Knowledge") 2nd Factor: specific generated One Time Password ("Possession")	Passwords are only eligible if more secure alternatives cannot be implemented. The synchronization of the authentic- ator app via a cloud service as well as sharing (e.g. via a pass- word manager) the OTP is not permitted.
---	--	--

*) WH4B: "Windows Hello for Business"

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: 3.01-20/9.0

Req 33 Privileged user accounts must be protected with at least two authentication features that come from different factors and provide increased resilience against phishing attacks.

A privileged user account is a user account that has advanced permissions within a system. Advanced permissions allow access to configuration settings, features, or data assets that are not available to regular users of the system. In direct dependence on the specific tasks performed by means of a privileged user account within a system, the advanced permissions assigned may be specifically limited or may include fully unrestricted system access.

Examples of privileged user accounts:

- Accounts for administration, maintenance, or troubleshooting tasks
- Accounts for user administration tasks (e.g. creating/deleting users; Assign permissions; password resets)
- Accounts that are authorized to legitimize, trigger, or stop business-critical operations
- Accounts with access to data classified as SCD (Sensitive Customer Data) in the interest of the Group Deutsche Telekom, its customers or the public
- Accounts that have extensive access to data defined as "personal" according to the EU-GDPR (e.g. mass retrieval of larger parts or the entire database)

To protect privileged user accounts, at least two authentication features must be combined. These must come from different factors ("KNOWLEDGE", "POSSESSION" or "INHERENCE"). A combination of authentication features of the same factor is not permitted. At least one of the authentication features must not be vulnerable to phishing attacks.

In a phishing attack, the legitimate user is trickily manipulated to reveal his login data to the attacker, for example by means of fake login pages. A phishing attack can also be carried out against a combination of multiple authentication features.

Phishing-resilient authentication features prevent attacks based on their technical implementation. These authentication features use cryptographic methods and create a strong bond between the user, the device and the service. They also require manual user interaction during authentication, which cannot be automated; for example, to submit a fingerprint scan to a hardware token.

Considered not vulnerable to phishing attacks are

- FIDO2 Level 2 Security Keys
- Certificates implemented in hardware

Important note on the use of passwords

Passwords must be strictly avoided as an authentication feature.

They are only acceptable if more secure alternatives cannot be implemented.

Note on technical user accounts

Technical user accounts used for machine access (also known as "machine-to-machine" or "M2M" accounts) are exempt from the obligation to use multi-factor authentication, as this is usually not feasible there.

Motivation: Privileged user accounts pose an increased risk to the security of a system. If an attacker successfully compromises such a user account, they will be granted broad privileges that allow them to take control of the system or system parts, disrupt system functions, view/manipulate processed data, or influence business-critical operations. The combination of multiple authentication features of different types already significantly minimizes the risk of a user account being compromised, but for privileged user accounts, special attention must be paid to resilience against phishing attacks. The phishing attacks that can be observed are becoming increasingly tricky and difficult for users to detect, so that multifactor authentication without increased resilience against phishing can no longer be considered sufficient for privileged user accounts.

Implementation example: Examples of MFA with increased resilience to phishing attacks

MFA Combination	Description	Application Note
Telekom "myCard" plus PIN	1st Factor: Certificate implemented in hardware ("Possession") 2nd Factor: PIN ("Knowledge")	(none)
FIDO2 Level 2 Security Key plus fingerprint	1st Factor: Crypto key stored in hardware ("Possession") 2nd Factor: Biometric feature ("Inherence")	Biometric feature must be pre-configured, otherwise the biometric sensor will only work as a "button" and does not provide the 2nd factor ("Inherence").
FIDO2 Level 2 Security Key plus PIN	1st Factor: Crypto key stored in hardware ("Possession") 2nd Factor: PIN ("Knowledge")	(none)

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: 3.01-21/9.0

Req 34 User accounts must ensure the unique identification of the user.

Users must be identified unambiguously by the system.

This can typically be reached by using a unique user account per user.

So-called group accounts, which are characterized by the fact that they are used jointly by several people, must not be used. This also applies without restriction to privileged user accounts. Most systems initially have only a single user account with administrative privileges after the basic installation. If the system is to be administered by several persons, each of these persons must use a personal, individual user account to which appropriate administrative authorizations or roles are assigned

A special feature are so named technical user accounts. These are used for the authentication and authorization of systems among themselves or of applications on a system and can therefore not be assigned to a specific person. Such user accounts must be assigned on a per system or per application basis. In this connection, it has to be ensured that these user accounts can't be misused.

Ways to prevent misuse of such user accounts by individuals include:

- Configuration of a password that meets the security requirements and is known to as few administrators as possible.
- Configuring the user account that only a local use is possible and a interactive login isn't possible.
- Use of a technique for authentication of the specific user account with public and private key or certificates.
- Limiting the access over the network to legitimate systems.

Additional solution must be checked on their usability per individual case.

Motivation: Unambiguous user identification is mandatory to assign a user permissions that are necessary to perform the required tasks on the system. This is the only way to adequately control access to system data and services and to prevent misuse. Furthermore, it makes it possible to log activities and actions on a system and to assign them to individual users.

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: 3.01-22/9.0

Req 35	The permissions for users and applications must be limited to the extent necessary to fulfill their tasks.
--------	--

The permissions on a system must be restricted to such an extent that a user can only access data and use functions that he needs in the context of his work. Appropriate permissions must also be assigned for access to files that are part of the operating system or applications or that are generated by the same (e.g. configuration and logging files).

In addition to access to data, applications and their components must also be executed with the lowest possible permissions. Applications should not be run with administrator or system privileges.

Motivation: If a user is granted too far-reaching permissions on a system, he can access data and applications to an extent that is not necessary for the fulfillment of the assigned tasks. This creates an unnecessarily increased risk in the event of abuse, in particular if the user or his user account is compromised by an attacker.

Applications with too far-reaching permissions can be misused by an attacker to gain or expand unauthorized access to data and system areas in need of protection.

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: 3.01-23/9.0