Security requirement

PostgreSQL databases

Deutsche Telekom Group

Version	7.0
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
	3.60	Security requirement
Version	**State**	**Status**
7.0	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 paper describes all requirements on security issues related to PostgreSQL.

Copyright © 2024 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 11
6. SQL extensions with operating system or network access 15
7. Data communication 16
8. Logging and Monitoring 18

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: 3.60-1/7.0

Req 2	On one operating system instance (both physical HW or virtualized) must only run one instance of a Postgres SQL system.

Motivation: If multiple database instances for different tasks (e.g., Internet and intranet) are running on one operating-system instance, the instances are not separated from one another. There is a risk that attackers could also corrupt the second database system. To separate the individual database instances deploy virtualization solutions.

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

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

ID: 3.60-2/7.0

3. Directory and File Permissions

Req 3	A database service muss 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: 3.16-18/7.0

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_subdirectorys'	'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 property 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: 3.60-4/7.0

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: 3.60-5/7.0

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: 3.60-6/7.0

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 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 preferably 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 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: 3.01-27/8.0

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, wich 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: 3.01-30/8.0

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,rolcreaterole,rolcreatedb FROM pg_roles WHERE rolsuper IS TRUE OR rolcreaterole 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 NOCREATEROLE;
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: 3.60-10/7.0

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: 3.60-11/7.0

Req 12	The “rolcanlogin” privilege may only be set for users/roles that also have an entry in pg_hba.conf.

Motivation: The privilege allows a login via the network and thus increases the attack surface.

Implementation example: To identify roles with login privileges execute following query:
SELECT rolname,rolcanlogin FROM pg_roles;

Use following command to removing rights:
ALTER ROLE user_name WITH NOLOGIN;

Additionally, in the "pg_hba.conf" file, any unnecessary entry beginning with host, hostssl, hostnossl should be commented out.

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

- Unnoticeable feasible attacks

ID: 3.60-12/7.0

Req 13	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: 3.60-13/7.0

Req 14	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: 3.60-14/7.0

Req 15	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.usename, pg_catalog.has_schema_privilege(u.usename, 'public', 'CREATE') AS "create"
FROM pg_user u
WHERE u.usename != 'postgres' AND pg_catalog.has_schema_privilege(u.usename, '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: 3.60-15/7.0

Req 16	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: 3.60-16/7.0

Req 17	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 privilleges 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: 3.60-17/7.0

Req 18	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.nspname, pg_user.usename FROM pg_proc JOIN pg_namespace ON pg_proc.pronamespace=pg_namespace.oid JOIN pg_user ON pg_proc.proowner=pg_user.usesysid WHERE prosecdef='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: 3.60-18/7.0

6. SQL extensions with operating system or network access

Req 19	Unless otherwise necessary, only "c" and "internal" may be used as non-trusted procedural languages.

Motivation: Programming languages installed in "non-trusted" mode can be used to gain access to the operating system level.

Implementation example: The following command can be used to determine all languages marked as "non-trusted":
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;

Additional background info: https://www.postgresql.org/docs/current/plperl-trusted.html

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: 3.60-19/7.0

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

Motivation: Additional procedural languages installed on the database can create new attack vector for the attacker.

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;

The following languages can remain as they are standards:

- internal
- c
- sql
- plpgsql

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

ID: 3.60-20/7.0

7. Data communication

Req 21	Each client connection to the database server must be implemented using a separate entry in pg_hba.conf. 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: 3.60-21/7.0

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: 3.60-22/7.0

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: 3.60-23/7.0

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: 3.01-34/8.0

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

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

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

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

ID: 3.01-35/8.0

Req 26	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 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.

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: 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 4: 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: 3.01-37/8.0

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: 3.60-28/7.0