Security requirement

Public Cloud Usage

Deutsche Telekom Group

Version	35 (internal)
Date	Jul 19, 2024
Status	In work

Publication Details

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

Reuterstrasse 65, 53315 Bonn
Germany

File name	Document number	Document type
	3.66	Security requirement
Version	**State**	**Status**
35 (internal)	Jul 19, 2024	In work
Contact	**Validity**	**Released by**
Telekom Security psa.telekom.de		

Summary
This document covers secure usage of public cloud services. The requirements in this document are mostly given on high level so they can cover most of cloud services providers and use cases, but the document does not provide technical implementation details for a specific cloud provider.

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

Table of Contents

1.	Introduction	4
2.	Introduction to Cloud Computing	5
3.	Public Cloud Usage Prerequisites	7
4.	Governance and Management	12
4.1.	Session Protection	21
5.	Authentication and Authorization	24
6.	Authentication attribute "password"	34
7.	Protecting Data and Information	40
8.	Protecting Availability and Integrity	45
9.	Architecture and Networking	48
9.1.	Cloud Connectivity	50
9.2.	Cloud Connectivity - External Connectivity	51
9.3.	Cloud Connectivity - Internal Connectivity	53
10.	Logging and Monitoring	55
11.	References	63
12.	Glossary	64

1. Introduction

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

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

2. Introduction to Cloud Computing

Introduction

According to NIST (National Institute of Standards and Technology), cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models.

Scope

This document covers the following topics:

- security guidelines for usage of public cloud APIs, IaaS and other services provided by public clouds
- security guidelines for cloud management
- security guidelines for cloud users and their applications

for the following type of cloud environments:

- public cloud

The following cloud types are not in scope of this document:

- private clouds
- community clouds

Hybrid clouds are not covered in particular. If an application uses both public and private cloud, public cloud requirements apply to all parts of applications which are located in public cloud.

In scope:

- Public cloud API, usage, configuration, limits
- Public cloud services

Out of scope of this document:

- Operating system and application requirements
- Container and orchestration technologies (like Docker and Kubernetes)

Shared Responsibility Model

Security of the cloud and services provided on top of cloud is job for both cloud service provider and cloud consumer. The security of the cloud platform and offered services is a responsibility of the cloud service provider, while their use, either direct or as building blocks for a final service is responsibility of the cloud consumer (called security in the cloud). The boundary depends on the service model used, e.g. when using IaaS, cloud provider needs to secure hypervisor, and separate network and storage between tenants, while cloud consumer needs to properly secure virtual machines and software running inside them.

This approach is called shared responsibility model and one visual example can be seen here https://docs.microsoft.com/en-us/azure/security/fundamentals/shared-responsibility.

It is worth noting that even with shared responsibility model, the cloud customer remains accountable for the data security.

Background Story

Definition of information security according to ISO/IEC 27000:2014 is: "Preservation of confidentiality, integrity and availability of information. Note: In addition, other properties, such as authenticity, accountability, non-repudiation and reliability can also be involved."
Threats, either deliberate or coming from environment, accidents, negligence are typically classified in the following types:

- physical damage
- natural events

- loss of essential services
- compromise of information
- technical failures
- compromise of functions

Protection mechanisms against threats may include protection mechanisms of the cloud service provider (which must be still configured accordingly by customers) and those which customer implements on top of other services provided by the cloud provider, or naturally - combination of both.

In a cloud environment, like in any other data processing environment, it is also necessary to classify the data and to consider not only data availability but also the data lifecycle (e.g. when the data can be archived, deleted etc.) and respectively protect the data during the whole lifecycle.

The term landing zone refers to a configuration of cloud environment, typically fulfilling security and compliance aspects, identity and access management, networking, billing etc. In context of this document, the landing zone can be also considered as a concept which allows central management of identities and projects inside CSP for the whole DTAG group, allows placing certain restrictions for all accounts (e.g. minimal password length or MFA enforcement), may be used to enforce central security logging etc.

3. Public Cloud Usage Prerequisites

Cloud Service Provider (CSP) provides computer system resources (e.g. compute, storage, network and application) to customers. Resources can be provisioned through APIs by customer. CSP is responsible for security of their infrastructure, services and APIs, while customer bears responsibility to secure everything built on top of resources and APIs provided by CSP.

Req 1	Before starting to work in public cloud, all users of cloud services must examine the relevant documentation.

All cloud service users, including maintainers of the landing zone and especially application/system maintainers and users, must make themselves familiar with the cloud services which they are making available or using. Landing zone operators may also prepare relevant information for users of the public cloud to facilitate easier onboarding (or this can be made a formality - as part of acceptance of terms and conditions for using the public cloud).

Motivation: The security of cloud services of the public cloud service providers is generally good because these services are public and under constant attack, and most security incidents are not due to the vulnerabilities in the services, but due to the wrong configuration, misuse or human errors. Some typical examples include:

- *(inadvertently) exposing private object storage bucket to public*
- *creation of allow-all security groups (inbound and outbound), e.g. exposing internal systems (e.g. databases) to Internet*
- *(inadvertently) exposing credentials of cloud services (enabled debug output, keys in plain text in CI/CD chain)*

Implementation example: Relevant documentation includes, but is not limited to:

- approval of a specific CSP and the cloud service inside DTAG, including the details about the data which may (not) be processed in such environment
- the documentation about the cloud service, including the security recommendations from the CSP (and internal security requirements if available)
- other security requirements documents from PSA portal, because for applications hosted in the cloud, they still apply

For this requirement the following threats are relevant:

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

ID: 3.66-1/i35

Req 2	The CSPs and the specific service models or services must not be used unless approved by DTAG Board of Management. The restrictions given in the approval related to approved data classes and information protection classes must be followed. In addition to BoM, if other official procedures and processes exist for the cloud service approval, they may be used.

The CSP is responsible for security of their infrastructure, services and APIs, which is basis for overall security. CSP is also responsible for fulfillment of their legal obligations and SLA. Since it would be impossible for all CSP customers to perform audits of a CSP, this is typically done by independent auditors which carry out audits and create audit reports.

The most relevant report from audit is SOC-2 type 2 report which needs to be evaluated by Telekom Security before

signing the contract with CSP. If BSI C5 audit report is available, it must be evaluated too.

In addition to technical security evaluation of CSP based on audit reports, legal and contractual aspects are combined and recommendation is presented to DTAG Board of Management which in the end decides whether the usage of the particular CSP is allowed, along with the respective list of specific service models and specific services.

One may look at "Trusted Cloud Provider Approach" in YAM because the information there can be refreshed more often than in this document: https://yam-united.telekom.com/pages/cloud-infrastructure-security/apps/blog/blog/view/d8f8b89f-9a94-4c40-a5aa-16fcbf6f1504. Further information may be found also here at DT Cloud Taskforce: https://yam-united.telekom.com/pages/dt-cloud-taskforce/apps/wiki/wiki/list.

Motivation: Every service model and particular service of every CSP may have approval for different data classes, e.g. the approval for processing and storing personal data in one CSP's IaaS does NOT automatically mean that processing and storing of personal data is allowed for another CSP's IaaS even if in both cases IaaS is approved as a service model.

Due to the rising number of public cloud providers and their services, certain PaaS and SaaS providers or services which were not included in initial DTAG BoM approval can be allowed by other management entities, but such management entities must include appropriate DTAG Security responsible persons.

Implementation example: Example of an approval: Based on evaluation, DTAG BoM has decided that cloud provider X is allowed to be used by DTAG for all kinds of IaaS workloads with any kind of data, along with key-value database from the PaaS portfolio for all data except personal data, while all SaaS services are prohibited.

Can DTAG use VMs, virtual routers, firewalls and other services tightly related to VM compute services?

- Yes

Can DTAG store data inside key-value database?

- In general, yes, but details of an approval (fine print) states which kind of data.

Can DTAG store data inside relational database provided as PaaS from this CSP?

- No. Not in the list of approved services.

Can DTAG use IoT SaaS solution from CSP X for management of IoT devices?

- No. SaaS is not approved from CSP X.

Can DTAG use CSP X to work with or store PCI-DSS data?

- Not in general. Although many cloud providers are PCI-DSS compliant, it is necessary to check whether the certification applies to all services of CSP which DTAG would like to use. For use of PCI-DSS there is an additional requirement in this chapter.

Can DTAG process and store personal data in IaaS of CSP X?

- Yes.

Can DTAG store personal data in key-value database from CSP X?

- No.

Can DTAG use software from vendor Z which hosts their software on top of Giant Swarm on Azure?

- Vendor Z is considered as "3rd level" vendor which utilizes Giant Swarm, which is SaaS provided by Microsoft Azure. If Azure is approved as CSP and there is an approval for SaaS services including Giant Swarm on top of Azure, the answer would be generally yes. However, it is still necessary to perform security evaluation of the

software from vendor Z, as much as possible for SaaS solutions, paying special attention to data classification restrictions. If there would be no approval to use Giant Swarm on Azure, a separate DTAG BoM decision (or equivalent process if existing) would be needed for usage of software from vendor Z.

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

ID: 3.66-2/i35

Req 3	All applicable laws, regulations and corporate policies must be considered and followed prior to using public cloud for a system or data.

No matter whether migrating existing system (or part of) or data to the public cloud, creating a new system which will work with new or existing data, it is necessary to verify that all laws, regulations and corporate policies are fulfilled. Some examples include:

- lawful interception
- data retention
- PCI-DSS data restrictions
- restrictions on geographical borders, both for data at rest (e.g. which regions, data centers or availability zones may be used), in use and in transit (e.g. is it allowed to transfer data over Internet or over geographical borders)
- telecommunications/communications act
- data provisioning

It is necessary to verify that the cloud provider offers sufficient controls to fulfill the requirements and that these requirements are implemented as policies for the system or data which is hosted in the public cloud.

Motivation: Not following laws and regulations may incur financial or reputation damage to DTAG and have legal consequences for employees.

Implementation example: In terms of implementation, cloud providers offer a variety of configurable policies which allow the cloud consumer to configure restrictions required by law, regulatory requirements and corporate policies throughout the whole landing zone, i.e. for all consumers of the cloud for a specific organization. For all restrictions which cannot be enforced through CSP's policies, appropriate administrative and/or organizational controls must be in place. People who use the cloud services need to be informed about the restrictions, learn them and follow them.

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

ID: 3.66-3/i35

Req 4	The CSP and it services must not be used for data with need of protection, especially PCI-DSS data, unless the CSP provided appropriate guarantees like PCI-DSS certification and this type of usage was approved by DTAG Board of Management.

Although many cloud providers are PCI-DSS compliant, it is necessary to check whether the certification applies to all services of CSP which DTAG would like to use. In addition to the certification, it is necessary that the DTAG Board of Management approved the CSP and specific services to be used with PCI-DSS data.

Motivation: Not following PCI-DSS requirements for handling credit card data may result in fines for DTAG.

Implementation example: This procedure (verification if the specific service of the cloud service provider) applies generally to data with need of protection, not only to the PCI-DSS data, although PCI-DSS is a very prominent example of

such use cases.

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: 3.66-4/i35

Req 5	Clear exit strategy must exist before onboarding a project or a system on a public cloud.

In the traditional world, this problem was solved by dual (or multi) vendor strategy for avoiding vendor lock-ins, but the transition from vendor to vendor was possible because equipment was bought and owned by DTAG. This is not the case when using public cloud providers. Having 2 or more CSPs at disposal is not sufficient for dual (or multi) vendor strategy, it is also necessary to move impacted systems and data as well, either from the CSP to another CSP or from the CSP to own premises.

Contracts and court proceedings cannot be considered as a viable replacement for an exit strategy.

Examples of problems:
- CSP X raised prices which makes usage too costly (production cost of a service is higher than revenue).
    - CSP Y did not, so migration of services and data makes sense, but there must be a clear concept from the very beginning to avoid vendor lock-in.
- CSP went bankrupt, equipment is seized, and operations are terminated.
- CSP is providing bad service (not fulfilling SLA) and cannot solve technical issues for hours/days/months.
- CSP will discontinue service which DTAG is using within weeks/months (periods are typically much lower compared to EOL announcements for traditional software and hardware).
- CSP cannot be longer used due to the new laws and regulations.
- Developing application only around specific APIs from one CSP (e.g. AWS BeanStalk) would mean that the whole application needs to be rewritten for a migration to another CSP.

Motivation: Despite good intentions and contracts, it is possible that the CSP will not be able to fulfill the needs of DTAG and respective systems in the future. In order to account for potential legal, financial or technical problems which may arise in the future, it is necessary to have a clear plan how to get services and the data out of a specific cloud provider and at any given time.

Implementation example: If possible, cloud consumer should consider using more generic cloud services, so that transition to the public cloud from another CSP (or into own premises) is done with minimal modifications (or no modifications in best case scenario).

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

ID: 3.66-5/i35

Req 6	The cloud services of the CSP must not be used unless done through the approved landing zone.

CSP and DTAG typically sign a single contract and DTAG gets a single big account inside CSP. Inside that account, the required hierarchy can be created, e.g. mimicking organization structure. Basic rules (e.g. minimal password length policy) would apply to all DTAG users of this CSP, while some system/application specific (whether a VM can or cannot access the Internet) would be driven by owners of specific organization or a project.

Each exception to this rule requires a separate BoM (Board of Management) decision, according to the current situation at the time of writing.

Motivation: Bypassing landing zone (e.g. by creation of another root account) would allow bypassing all the policies and is forbidden.

Implementation example: Use of only approved landing zones, i.e. do not separately create another landing zone without BoM approval.

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

ID: 3.66-6/i35

Req 7	Unnecessary services must be disabled.

After the installation of systems and software products, supplier-preset, local or network-accessible services are often active that are not required for the operation and functionality of the specific system in the intended operating environment.

However, in principle only the services actually required may be active on a system.

Accordingly, all services that are not required on a system must be completely disabled immediately after installation. It must be ensured that these services remain disabled even after the system is restarted.

In the context of the cloud this means: All cloud services which are not needed or out of scope for a landing zone or a project needs to be disabled as far as the cloud service provider supports this, e.g. by using service control policies.

Motivation: Active services that are not required unnecessarily increase the attack surface of a system and, as a direct consequence, the risk of a successful compromise. This risk can be further increased if - as is often observed with services that are not required - a targeted examination and optimization of the configuration with regard to security does not take place sufficiently.

Implementation example: For Azure or AWS you can use Azure policies or service control policies to deny access to the services which are not needed.

ID: 3.66-7/i35

4. Governance and Management

Req 8	Process for handling abuse and incident notifications from and to the cloud service provider must be established.

Cloud service providers typically notify customers on abuse and incidents which they detect, and sometimes even on identified irregularities (e.g. huge jump in resource consumption). The landing zone operators need to, in cooperation with responsible SOC, establish a process which enables handling of such notifications 24/7.

In the same way, it needs to be possible for a cloud user or landing zone operator to report incidents and abuse to the cloud service provider, which can be organized through the landing zone operators in agreement with responsible SOC.

Motivation: CSPs may detect certain abuse and incidents, even if the customer does not notice that, and will in such cases notify their customers.

Implementation example: Mailbox which is monitored 24/7 or phone number of 24/7 on-duty personnel.

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

ID: 3.66-8/i35

Req 9	Asset inventory must be maintained for all assets in the cloud and the process for management of assets must be established.

Each system in the cloud is responsible to keep track of their assets. Assets should be tagged if possible, or marked in any other way so that it is possible to easily identify system owner and people who need to be notified in case of security incidents, along with the criticality and business relevance of the specific asset. Details will depend on the landing zone and the way how the security incident handling has been organized between landing zone operators, system owners and responsible SOC.

Motivation: All cloud assets (virtual machines, IP addresses, storage buckets, URLs of serverless functions, ...) typically come with a price tag (monthly), therefore it is extremely important to keep track of all the assets. Additional cost is not the only concern in publich cloud, leftover assets can be also used as easier point for attacks.

Implementation example: If available, central asset management system should also include the asset information from the public cloud, preferably through the landing zone.

The best way a responsible SOC to access asset inventory in the cloud is through API because the information is then up-to-date, however, in some cases regular periodic exports and snapshots may serve the purpose if API access is not possible or convinient.

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

ID: 3.66-9/i35

Req 10	Security tasks must be automated to reduce human errors during implementation.

Typical tasks which must be automated whenever feasible are listed in Example section.

Motivation: In modern highly/fully automated environments like cloud (and CI/CD chains used by DevOps/DevSecOps organizations), manual processes only hinder fast pace which is expected and lead to even more common human/operator errors.

Implementation example: It is necessary to automate crucial processes:

- IAM related tasks - account, user, group and policy management
- security hardening of systems (security as code)
- security scanning and testing in CI/CD chains
- compliance and security checks of cloud services and systems running in the cloud (including verification of software/packages)
- vulnerability detection (both in CI/CD chains and in live environments)
- log data correlation and attack detection
- key, certificate and secret management
- detection and alerting of misconfigurations which cannot be enforced by policies
- security documentation generation for PSA process

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

ID: 3.66-10/i35

Req 11	Not approved images must not be used.

In order to avoid potential security incidents, it is recommended to use the constraints of the landing zone and allow use of only pre-approved images (including, but not limited to virtual machine images and container images). Depending on the selected cloud provider, these images may be the default ones made available by the cloud provider if they fulfill the DTAG security requirements for operating system and software in question. If such images are not available from the cloud provider, the next best option is to create appropriate hardened images and upload them and make them available for use, so that application/system owners cannot upload their own new images.

If the restriction for using only predefined images is not practical or possible in some cases (in terms of setting up landing zone restrictions), monitoring of images uploaded by users must be established to ensure that only appropriate and intended images are used.

Images delivered directly from vendors should be generally treated in the same way as usual: signatures should be verified, and contents of the image should be verified that it does not contain at least known vulnerabilities and issues.

Motivation: This prevents the possibility of the attacker who gained access to the public cloud accounts (personal or technical) to use their own images which restricts the attackers capability for misuse of the cloud resources and further attacks towards other systems hosted in the public cloud or DTAG internal systems in general.

Implementation example: A system like Magenta Trusted Registry (MTR) can be used for scanning container images: https://yam-united.telekom.com/pages/digital-hub-ci-cd/apps/content/magenta-trusted-registry-mtr.

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

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

ID: 3.66-11/i35

Req 12	Only required software may be used on the system.

In the installation routines for software provided by the supplier, individual components of the software are often preselected as standard installations, which are not necessary for the operation and function of a specific system. This also includes parts of software that are installed as application examples (e.g. default web pages, sample databases, test data), but are typically not used afterwards.

Such components must be specifically deselected (not installed) during the installation of the system or - if deselection during installation is not possible - removed immediately afterwards.

In principle, no software may be used that is not required for the operation, maintenance or function of the system.

Motivation: Vulnerabilities in a system's software are gateways for attackers. By uninstalling unnecessary components, the potential attack surfaces can be significantly reduced.

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

ID: 3.01-3/8.0

Req 13	Features that are not required in the software and hardware used must be deactivated.

During the initial installation of software, features may have been activated by default that are not necessary for the operation and functionality of the specific system. Features are usually an integral part of the software that cannot be deleted or uninstalled individually.

Such features must be disabled immediately after the initial installation through the software's configuration settings, so that they remain permanently disabled even after the system is rebooted.

Even before delivery or during initial commissioning, features may have been activated by default in the hardware that are not required for the purpose of the specific system. Such functions, for example unnecessary interfaces, must also be permanently deactivated immediately after initial commissioning.

Motivation: A system's hardware or software often contains enabled features that are not being used. Such features can be an unnecessary target for manipulation. Furthermore, there is a potential that unauthorized access to areas or data of the system can be created.

Implementation example: [Example 1]
Deactivation of debugging functions in the software that are used in the event of fault analysis, but do not have to be active during normal operation.

[Example 2]
Disabling unused network interfaces of a server.

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

ID: 3.01-4/8.0

Req 14	Virtual machines and containers which are immutable (not changed during their lifetime) must not be accessible by human users except for debugging purposes.

The concept of using immutable virtual machines and containers is known as immutable infrastructure. Each virtual machine or a container does not change during their lifetime. Instead of patching or reconfiguring a live system, the approach relies on creating new virtual machines or containers and deleting the older ones.

All production systems using immutable virtual machines and containers must not be accessible by human users for management purposes unless required for debugging, and therefore allowed only for a specific time period (conditional access).

This concept is typically not fully applicable to development, and in some cases to test systems.

Patching of immutable systems must be done through redeployment.

Motivation: This practice is very common in combination with CI/CD chains where automated testing and deployment are used to test new functionality, patch images and bringing them to life. The approach also yields the following benefits:

- *no need for remote management access*
- *no need for complex and time-consuming patching procedures*
- *both functional and security testing is integrated from the beginning*

Implementation example: Build a new image, including necessary changes, replacing the old virtual machine or container with the new one and removal of the old virtual machine or the container.

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

ID: 3.66-14/i35

Req 15	Software and hardware of the system must be covered by security vulnerability support from the supplier.

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

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

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

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

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

Motivation: Hardware and software products for which there is no comprehensive security vulnerability support from the supplier pose a risk. This means that a product is not adequately checked to determine whether it is affected by

further developed forms of attack or newly discovered vulnerabilities in technical implementations. Likewise, if there are existing security vulnerabilities in a product, no improvements (e.g. updates, patches) are provided. This results in a system whose weak points cannot be remedied, so that they remain exploitable by an attacker in order to compromise the system or to adversely affect it.

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

ID: 3.01-1/8.0

Req 16	The software used must be obtained from trusted sources and checked for integrity.

The software used on the system must be obtained from trusted sources and checked for integrity before installation. This requirement applies to all types of software:

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

as well as other software that may be used.

Trusted Sources
Trusted sources are generally considered to be:

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

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

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

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

ory).

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

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

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

Methods for an extended integrity check can be, for example:

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

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

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

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

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

ID: 3.01-2/8.0

Req 17	Known vulnerabilities in the software or hardware of the system must be fixed or protected against misuse.

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

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

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

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

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

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

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

ID: 3.01-10/8.0

Req 18	Systems must be included in company compliance/security tool.

The system (as the whole system - not only virtual machines and containers, but images, storage buckets, serverless functions, etc.) must be included in company compliance/security tool(s), no matter if they are deployed as immutable systems or in a more traditional way.

For a system which cannot be directly integrated in company compliance/security tool(s) there must be an appropriate process in place to allow for daily checks of patch level and comparison against known vulnerabilities.

Regular monitoring of advisories of Telekom Security CERT is necessary, unless a company compliance/security tool provides only information applicable to the specific system.

In case of a system where 3.63. ("Operations in DT IT") applies, Requirement 10 and Requirement 11 from 3.63. apply as more specific requirements.

Motivation: Having a system included in compliance tool allows faster detection of vulnerabilities and facilitates timely remediation of vulnerabilities.

Implementation example: Currently, one such tool which is available is TASTE-OS (OS = offline scanner), provided by Telekom Security. In example of TASTE-OS, appropriate scripts must be included into CI/CD build chains and in production systems, no matter if they are deployed as immutable systems or in traditional way. List of all TOS instances

can be found at https://yam-united.telekom.com/pages/taste-os-anwender/apps/wiki/dokumentation/list/view/8a2d0018-3bfb-4933-9289-61a022ceb5d3.

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

ID: 3.66-18/i35

Req 19	Intrusion detection and intrusion protection systems must be installed in virtual machines in the cloud, if the CSP allows it, for all workloads and data hosted in the public cloud which are sufficiently sensitive and require such protection.

Intrusion detection systems (IDS) and intrusion prevention systems (IPS) can improve security of virtual machines running in the public cloud, although their use is not mandatory in general. However, depending on the system classification, sensitivity of the data and potential threats, it may be required to use additional IDS or IPS for certain workloads. Most of public cloud service providers allow running such systems, but it is necessary to verify for each particular use case.

Mandatory access controls (like AppArmor or SELinux, which can be considered as a basic IDS/IPS) must be used in usual way defined by other valid security requirements for operating systems and applications.

Motivation: IDS/IPS systems may identify unusual system behavior and could be very useful also in cases of zero-day attacks.

Implementation example: At minimum use of AppArmor or SELinux.

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

ID: 3.66-19/i35

Req 20	A configuration repository must be used as a single source of truth for all configuration files and infrastructure-as-code (IaC) scripts that define the cloud resources.

The establishment of a configuration repository as the single source of truth for all configuration files and infrastructure-as-code (IaC) scripts is essential to ensure efficient, organized, and reliable management of cloud resources. By having a dedicated repository with version control, teams can easily track changes, maintain consistency, enhance collaboration, and support disaster recovery, contributing to a streamlined and transparent cloud infrastructure management process.
The repository must be protected with appropriate access controls to enforce the "need-to-know" and least privilege principle, allowing only authorized personnel access to specific configurations and the deployment to the cloud.

Motivation: Having a dedicated config repository, helps reduce configuration errors, streamline configuration management processes, enhance security by enforcing standardized configurations, and simplify troubleshooting and auditing activities.

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

ID: 3.66-20/i35

Req 21	All cloud resources (virtual machines, storage buckets, databases, network components, ...) that support tags, must use tags with appropriate and meaningful metadata.

Tagging is an essential feature that must be implemented in cloud environments to organize resources within an organization and reduce complexity and administrative overhead. Enforcing a tagging policy can ensure consistency, accuracy, and standardization of all resources in a cloud environment, allowing for better resource tracking and management.

Motivation: Tags enable the categorization of resources with own metadata. This can be useful to implement e.g. security configurations in an automated manner and to classify data.

Implementation example: The specifications of which tags to use are defined by the landing zone. Additional tags can be defined by applications themselves.

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

ID: 3.66-21/i35

Req 22	Permissions and controls implemented as the landing zone must be verified before use and after significant changes.

Verification of the correctness of controls implemented as the landing zone is mandatory before use and whenever there are substantial changes in the landing zone (e.g. complete redefinition of roles). This applies to at least the main landing zone and to the parts of the landing zone which have diverging (overriding) policies compared to the top-level landing zone.

The usage of automated tools for penetration tests and CIS benchmarks can become very expensive in no time, so it is very important to appropriately set the right scope of tests and scans according to the needs.

Motivation: Any change and of configuration and controls (especially policies in terms of cloud landing zone) may result in new security vulnerabilities.

Implementation example: The following tools and methods can be used for performing the verification:
- automated penetration test tools (most common example is privilege escalations)
- custom (manual) penetration tests - typically require a creative approach
- CIS benchmarks (https://www.cisecurity.org/) and similar automated verification/reporting tools

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

ID: 3.66-22/i35

Req 23	Outputs and messages must not disclose information on internal structures of the system.

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

In general, this concerns information on

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

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

- Login windows and dialogs
- Error messages
- Status messages
- Banners of active network services
- System logs and log files
- Debug logs, stack traces

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

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

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

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

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

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

For this requirement the following threats are relevant:

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

ID: 3.01-9/8.0

4.1. Session Protection

Session protection is essential to prevent unauthorized access, session hijacking, and maintain data confidentiality for user interactions with webservices. In the cloud environment especially the webportal of the cloud is the place to

check for session protection.

Req 24	Sessions must be protected against unauthorized takeover ("session hijacking").

Interfaces that provide session functionality to the system must implement technical measures to prevent a legitimate user's session from being taken over and continued by an unauthorized third party.

Such protection can be achieved, for example, by implementing a combination of the following options that makes sense for the specific system:

- At the transport layer: Use of the TCP protocol (with its sequence numbers) and corresponding filter lists
- At the session layer: Use of the TLS Protocol
- At the application layer: Negotiation of a random secret session key between sender and receiver to authorize all session traffic (e.g. session ID, session cookie, session token)
- Use of cryptographic methods to protect session keys from eavesdropping or modification attacks

Motivation: Unprotected sessions can potentially be hijacked and continued by an attacker in order to exercise unauthorized access to the system in the context of the affected user.

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: 3.01-16/8.0

Req 25	The system must allow users to log out of their current session.

The system must have a feature that enables the logged-in user to log out at any time. It must not be possible to resume a logged-out session without re-authenticating the user.

Motivation: A user must retain complete control over the sessions he has established in order to be able to terminate his access to a system at any time according to the situation and thus protect data and functions exposed via this access. In addition, the user must be able to assume that sessions specifically terminated by him cannot subsequently be resumed and continued by unauthorized third parties.

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: 3.01-17/8.0

Req 26	Sessions must be automatically terminated after a period of inactivity adapted to the intended use.

It is necessary that sessions on a system are automatically terminated after a specified period of inactivity.

For this reason, a time-out for sessions must be set. The time period to be selected here depends on the use of the system and, if applicable, the physical environment. For example, the time-out for an application in an unsecured environment must be shorter (a few minutes) than the time-out for an application used by operations personnel for system monitoring tasks in an access-protected area (60 minutes or more).

Motivation: For an open but unused session, there is a risk that an illegitimate user may take over and continue it un-

noticed in order to exercise unauthorized access to the system and the data contained therein on behalf of the affected user.

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: 3.01-18/8.0

5. Authentication and Authorization

In order to make use of approved CSP through a proper contract, it is necessary to create "Landing Zone" for every CSP. Landing zone is a concept which allows central management of identities and projects inside CSP for the whole DTAG group.

Landing zone starts with "root account" for whole DTAG, managed by team responsible for managing CSP usage (cloud management team in rest of the chapter). The cloud management team manages set of basic (security) policies, users and projects. Each project/system/department/organization would get access to CSP over the landing zone and they would be able to consume cloud services with predefined quota limits and within basic security policies. Each organization has a freedom to customize their portion under given initial constraints. Each project under every organization would be able to customize further.

Landing zone is prerequisite for complete and comprehensive security monitoring and accounting.

Req 27	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 the user name and at least one authentication attribute. 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

Examples of authentication features that can be used:

- Passwords
- cryptographic keys or certificates (e.g., in the form of smart cards)

This requirement also applies without restriction to any machine access to the system (here the implementation is usually carried out by using so-called M2M - "Machine-to-Machine" - user 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: 3.01-19/8.0

Req 28	User accounts must be protected with at least one authentication attribute.

All user accounts in a system must be protected against unauthorized use.

For this purpose, the user account must be secured with an authentication attribute that enables the accessing user to be unambiguously authenticated. Common authentication attributes are e.g.:

- passwords, passphrases, PINs (factor KNOWLEDGE: "something that only the legitimate user knows")
- cryptographic keys, tokens, smart cards, OTP (factor OWNERSHIP: "something that only the legitimate user has")
- biometric features such as fingerprints or hand geometry (factor INHERENCE: "something that only the legitimate user is")

The authentication of users by means of an authentication attribute that can be faked or spoofed by an attacker (e.g. telephone numbers, IP addresses, VPN affiliation) is generally not permitted.

In companies of Deutsche Telekom group where the MyCard or a comparable smartcard is available this should be a preferred authentication attribute.

If the system and the application scenario support it, multiple independent authentication attributes should be combined if possible in order to achieve an additional increase in security (so-called MFA or Multi-Factor-Authentication).

Motivation: User accounts that are not protected by appropriate authentication attributes can be abused by an attacker to gain unauthorized access to a system and the data and applications stored 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: 3.01-20/8.0

Req 29	Personal user accounts (for persons) must be personalized and must use multi-factor authentication (MFA).

All users must be unambiguously identifiable. This requires having a unique user account for every person. This is mandatory to assign necessary user rights for performing tasks on the system. It also allows attributing of logged activities and actions on a system to a specific individual.

CSPs offer web console access and access to APIs over the Internet. Protection of an account with password only is not considered sufficient. Account hijacking and API misuse can be mitigation with MFA, where at least 2 factors are needed. Second and all subsequent factors need to use hardware solutions (smart card like MyCard or USB token) or software solutions (an app on mobile device). SMS should not be used.

Persons are not supposed to use non-interactive access directly (like API access), but rather to use technical user (sub)accounts which belong to them. Technical user accounts are covered later in the document.

In an extremely unlikely case where a personal user account cannot create technical user (sub)accounts for non-interactive access because CSP does not have the feature (please let authors of this document know about such bad CSP), or in the case where the required tasks are practically impossible to perform using very limited technical user accounts, personal user account may use short lived tokens, like for technical user accounts, but with reasonably short lifespan - just sufficient time to perform currently necessary duties (e.g. such token should not last longer than a day).

Motivation: The assignment of identities helps to assign user rights granularly and to assign actions unambiguously to a user.

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

ID: 3.66-29/i35

Req 30	Privileged user accounts must be protected with at least two authentication attributes from different factors.

A privileged user account is a user account with extended authorizations within a system. Extended authorizations enable access to configuration settings, functions or data that are not available to regular users of the system. In direct dependence on the special tasks that are carried out via a privileged user account within a system, the assigned extended authorizations can be specifically restricted or include completely 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; assigning permissions or roles; resetting passwords)
- Accounts that are authorized to legitimize, initiate or prevent business-critical processes
- Accounts that have access to data classified as SCD (Sensitive Customer Data) in the interests of 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 complete database)

A single authentication attribute for privileged user accounts with their extended authorizations is usually no longer sufficient.

In order to achieve an adequate level of protection, at least two mutually independent authentication attributes must be used. The authentication attributes must come from various factors (knowledge, ownership, inherence). A combination of authentication attributes of the same factor (e.g. two different passwords) is not permitted

This approach is commonly referred to as MFA (Multi-Factor Authentication).
A specific form of MFA is 2FA (2-factor authentication), which combines exactly two authentication attributes.

Motivation: Privileged user accounts represent an increased risk to the security of a system. If an attacker successfully compromises such a user account, he receives extensive authorizations with which he can bring the system or system parts under his control, disrupt system functions, view/manipulate processed data or influence business-critical processes. The combination of multiple authentication attributes of different types significantly minimizes the risk of a user account being compromised.

Implementation example: Very popular is 2FA in a variant consisting of an attribute that the user knows (factor KNOWLEDGE) and an attribute that the user possesses (factor OWNERSHIP).
Examples of such a 2FA are:

- smartcard (e.g. MyCard) plus PIN
- private key plus passphrase
- classic password plus hardware token for the generation of OTPs

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: 3.01-21/8.0

Req 31	Technical user accounts must use appropriate authentication mechanisms (short lived tokens or access keys instead of MFA).

Technical user accounts are necessary for automation and they cannot use typical MFA which is mandatory for per-

sons. Fortunately, there are other options which can be used for automated M2M communication scenario. One such options is use of tokens, access keys and certificates (later, only the term token is used) which have very limited scope, privileges and lifetime. Depending on the selected CSP, options might vary but the following restrictions need to be applied whenever possible:

- Technical user accounts or their access keys/tokens are not allowed to be used by persons
- Tokens for technical user accounts must have limited lifetime and periodically exchanged.
    - If there is no way to specify initial lifetime of a token, there must be an established process for rotation of older tokens.
    - Process for rotation of tokens needs to consider smooth transition from old to new token.
    - Token lifetime needs to be appropriately monitored to avoid downtime caused by expired tokens.
    - Stolen or leaked tokens must be immediately revoked/deleted.

Motivation: Short lived credentials help to minimize the impact of potential security breaches, limit the exposure of information with need of protection, and maintain a proactive security posture.

Implementation example: Use OpenID Connect with Gitlab to obtain temporary technical user account credentials for the CSP.

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: 3.66-31/i35

Req 32	Secret (key, token, certificate) rotation must be frequent. State-of-the-art secret protection mechanisms must be used.

Secrets used in a public cloud requirement are at higher risk of being stolen or leaked. To minimize the impact of stolen secrets, they need to be generated with shorter lifetime compared to traditional deployments. It is practically impossible to predefine specific lifetime of secrets for every possible system and data class, therefore the final decision is left to agreement between application owner and responsible security officer. The currently valid security requirements for secrets are contained in PSA document 3.50 "Cryptographic Algorithms and Security Protocols". As of this writing, a rotation of secrets is necessary every 24 months or on indication of compromise. Rotation of secrets needs to be implemented in a way which allows smooth transition, i.e. without service interruption.

Secrets will be used in the public cloud environment itself, so it is necessary to employ state-of-the-art secret protection mechanisms. One such example is protection of private keys at rest in RAM against speculation and memory channel attacks like Spectre, Meltdown, Rowhammer and RAMBleed. More information about this specific protection mechanism can be found at: https://marc.info/?l=openbsd-cvs&m=156109087822676&w=2.

Examples:

- Example for HTTPS certificate duration installed on front-end web servers or proxies: validity of a certificate for more than 1 year might be too high. 3 months would be probably a better choice.
- Private-public key pair in asymmetric encryption (RSA key pairs for SSH) duration for more than 2 years might not be the best idea (and the rotation is mandatory), even with protection of private keys in RAM offered by SSH software.
- Some further typical examples can be found athttps://www.keylength.com/.

This requirement is in line with Requirement 16 from 3.63. ("Operations in DT IT") for those systems where 3.63. applies.

Motivation: Regularly rotating secrets help minimize the impact of potential security breaches, limit the exposure of information with need of protection, and maintain a proactive security posture. Security breaches are, in most cases, exposed secrets through direct use in the source code that is visible to unauthorized persons through a cross-code repository (Gitlab, Github, etc.), or the propagation of the keys through other unintended actions.

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: 3.66-32/i35

Req 33	Predefined authentication attributes must be changed.

After the takeover or initial installation of a system, there are usually predefined authentication attributes (e.g. passwords, SSH keys, SSL/TLS Certificates) in the system, as assigned by manufacturers, developers, suppliers or automated installation routines.

Such predefined authentication attributes must be changed to new, individual values immediately after the takeover or installation of the system.

In the context of the cloud this means:
Also in the cloud, the creation or initiation of a new service can include a standard/predefined authentication attribute, for example a standard password for a service user. As mentioned above these attributes must be changed to a new individual value after provisioning the service in the cloud.

Motivation: Values predefined by third parties in authentication attributes cannot be trusted because they do not represent a controlled secret. Affected authentication attributes can be misused by unauthorized persons to access and compromise systems. This risk is significantly increased if commonly known default values are used for authentication attributes (e.g. a default password for the administrator user account in a particular software product).

ID: 3.66-33/i35

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: 3.01-22/8.0

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: 3.01-23/8.0

Req 36	Technical user accounts must have limited privileges. API access for technical user accounts must be limited to minimum which is necessary.

Permissions for technical user accounts must be restricted to such an extent that technical user accounts only access data and use features that are needed in the course of their work.
That is very important, because technical user accounts cannot use typical MFA as an additional security mechanism, which would give them an additional protection.
Depending on the selected CSP, options might vary but the following restrictions need to be applied whenever possible:

- Technical user accounts must have no more than bare minimum of privileges needed for tasks they perform
    - Although this is a valid requirement for all user accounts, technical user accounts are typically used by automation tools which have predefined API calls and therefore do not need more privileges than required by possible API calls.
- Technical user accounts must have limited scope for resource access (only a subset of resources)
- Technical user accounts or their access keys/tokens are not allowed to be used by persons

- Tokens for technical user accounts must have limited lifetime and periodically exchanged.
- Access to CSP APIs must be limited in terms of network reachability.
    - APIs of CSP are available over Internet, however, it is possible in some cases to restrict network access for technical user accounts to APIs so that connections from only specific IP address ranges or locations is possible.

Motivation: If the rights granted to a user/service on a system are too broad, it could be possible to access data and applications for which viewing or the use is not permitted. This would give the opportunity to disclose or modify confidential data and to manipulate system files.

Implementation example:

- If technical user account needs only to read object storage, it must not have a privilege to modify, write or delete privileges for object storage.
- If technical user account is used for storing logs to object storage, it needs to have appropriate create/write privileges, but must not have privileges for reading, modifying or deletion of objects.
- If technical user account is used for provisioning of virtual machines, it must not have privileges for modification of security groups.
- If technical user account is used for management of only a subset of virtual machines (e.g. VMs with a specific tag), it must not have possibility to manage other virtual machines (VMs with different tags).
- If technical user account accesses CSP's APIs from internal networks (e.g. from VMs inside cloud) and not over Internet, and if CSP allows restriction of API access for given technical user account, Internet access for such technical user account must be removed.

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: 3.66-36/i35

Req 37	All public cloud user accounts must be under root account. Other root accounts must not be created.

As specified previously, usage of CSP is allowed only through the proper landing zone, and since the landing zone is created by initial default root account, all subsequently created users must belong to that main account.

Creating another personal or business root account on that CSP would create a new landing zone for that user, which would allow bypassing security policies set for the previously described landing zone and therefore is forbidden.

Examples:

My project is special, and I need to use X and we must have a separate root account.

- A new BoM decision is necessary.

Motivation: Creating another personal or business root account on that CSP would create a new landing zone for that user, which would allow bypassing security policies set for the previously described landing zone and therefore is forbidden.

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

- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources
- Disruption of availability
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

ID: 3.66-37/i35

Req 38	Default (root) account(s) must be protected and used only with 2FA/MFA, and where possible, following four-eyes principle.

Root account for the whole DTAG inside CSP is created with all available privileges. Such account is a security risk and must not be used for daily operations. Other accounts with extremely restricted set of privileges need to be used for daily operations. However, in some rare cases, it is necessary to perform tasks with the root account, and therefore this type of an account must be protected with the following:

- 2FA/MFA with hardware token (like USB token) must be activated
- if possible, use FIDO2 strong authentication instead of password
- if password must be used, password must be extremely secure (more complex than DTAG password policy), e.g. at least 15 characters long of all 4 classes of characters (lowercase, uppercase, numbers, special characters) or a password with otherwise sufficiently high entropy
- upload of password to services like pwned is discouraged because it is expected that such passwords are computer generated, and revealing a password to anyone means it is no longer a secret
    - in contrast to that, it is a very good practice to monitor if e-mail address (account identifier in general) is included in breaches
- if available, security challenges for account recovery must be configured, and answers must be non-trivial; they must be at least 20 characters long and comprise of all 4 classes of characters or have equivalent entropy

All above mentioned credentials and devices must be securely stored in a secure location, like physical safe. It is recommended to store them on 2 locations (e.g. password + token at one location, recovery secrets on another location) for redundancy because earthquakes, fires and floods still happen.

Every action of default/root account(s) ("superadmins") must be fully logged for auditing purposes. It is advisable to also configure informing of all individuals involved in the top-level cloud administration (e.g. via e-mail) whenever a login to such account occurs, or in case of some other changes to such account(s).

This requirement is in line with Requirement 16 from 3.63. ("Operations in DT IT") for those systems where 3.63. applies, however, it is necessary to note that password length/complexity from this requirement overrides Requirement 16 from 3.63.

Motivation: Reduction of the attack surface.

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

ID: 3.66-38/i35

Req 39	Predefined user accounts that are not required must be deleted or at least disabled.

On many systems, there are predefined but unused user accounts (e.g. "guest") after the initial installation.

These predefined user accounts must be deleted or at least disabled immedately after the initial installation; if these measures are not feasible, the corresponding user accounts must be blocked for remote access. In any case, disabled or blocked user accounts must also be provided with an authentication attribute (e.g. a password or an SSH key) so that unauthorized use of such a user account is prevented in the event of a misconfiguration.

Except from the requirement to delete or disable predefined user accounts are user accounts that are used exclusively for internal use on the corresponding system and that are required for the functionality of one or more applications of the system. Even for such a user account, it must be ensured that remote access or local login is not possible and that a user of the system cannot misuse such a user account.

In the context of the cloud this means: For the usage of a service from a CSP, a service specific account is needed or the usage of a specific set of permissions, depending on the CSP and the service. These service accounts or set of permissions are most likely not least privilege and needed to be replaced by individual service accounts or set of permissions. It follows, that these service accounts and set of permissions needs to be deleted or if this can not be done, it needs to be made sure, that they are disabled (nobody can use the service account or permissions).

Motivation: User accounts that are predefined by default in a product are typically common knowledge and can be targeted by an attacker for brute force and dictionary attacks. If these user accounts are not needed in a specific system, their existence represents an unnecessary attack surface. A particular risk is posed by predefined user accounts that are preconfigured without a password or with a well-known standard password. Such user accounts can be misused directly by an attacker if their security hardening was missed due to the unplanned use in the specific system.

ID: 3.66-39/i35

Req 40	IAM solution of DTAG must be used for user authentication if supported by CSP. If not, automation for managing identities in the cloud must be in place.

Double administration of users brings issues like untimely creation and deletion of accounts (which can lead to data leaks), so it is necessary that accounts in the cloud are mapped to accounts in own DTAG's IAM. In terms of technologies - Active Directory identity federation, AFDS Shared (AAD expected in near future), SAML 2.0, OpenID+OAuth (2.0) and LDAPS (inside VPN connection only) are viable options.

In an unlikely case of not being able to integrate DTAG's own IAM solution for managing identities in the cloud, it is necessary to automate account management. Such automation for management of identities has to fulfill the "IAM (Identity Access Management) - Framework" (3.69.) requirements.

Preferred solution would be session based since no employee personal data would be exchanged with the cloud provider.

It is worth mentioning that this requirement also applies to applications running in the cloud, however, that is already covered with other security requirements (for operating systems and applications).

Example:

- Employee has left DTAG and his account in the cloud must be deleted to prevent him accessing the cloud resources.

In case of a system where 3.63. ("Operations in DT IT") applies, Requirement 7 from 3.63. applies as more specific requirement.

Motivation: The use of two or more IAM systems for one solution leads to a high level of complexity, increases the risk of inconsistencies and security gaps, compared to one IAM system. Always use already approved standard systems, to remove complexity and increase security.

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: 3.66-40/i35

Req 41	Privileges for cloud usage must not be associated directly with users or services.

Terminology may vary between CSPs, so most common one was used here.

Privileges for cloud service consumption must be assigned to roles and never to individual users.

Users need to be organized in groups, and these group names must clearly identify their meaning (e.g. administrators, operators, reporters). Multiple users may share the same group.

Groups (or policies) allow mapping of between users and roles (set of privileges). In some instances, this allows users who have multiple roles to select an appropriate role (getting all necessary privileges) for the work.

This concept simplifies assignment and removal of privileges to specific users and allows better control over privileges, and even allows users switching from one area of responsibility to another one (e.g. developer -> operations).

Motivation: Assigning permissions to roles or groups offers benefits such as simplified management, consistency, granular control, enhanced security, scalability, and improved auditability, making it a best practice for managing access controls in cloud environments.

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

ID: 3.66-41/i35

Req 42	Privileged Access Workstations (PAWs) must be used for administrative Tasks.

A Privileged Access Workstation (PAW) is a isolated operating system (hardware/virtualized) used for the purpose of securely accessing privileged accounts and resources.

A PAW must be used when performing administrative Task on a Public Cloud.

PAWs usualy prohibit the use of risky applications like e.g. Web Browsers and Email Clients. They idea behind a PAW is to disconnect daily activities like e.g answering mails from administrative tasks.

Motivation: PAWs provide enhanced security for administrators working with high risk resources vulnerable to compromise, due to the use of trusted hardware and hardened dedicated systems.

Implementation example: Use a PAW solution for administrative tasks.

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

ID: 3.66-42/i35

6. Authentication attribute "password"

Req 43	If passwords are used as an authentication attribute, those must be stored using a suitable and approved "Password Hashing" method to protect against offline-attacks like brute force or dictionary attacks.

This requirement relates to the storage of passwords in all types of user databases, as used in this system, in order to authenticate incoming access (local or remote) by users or other systems.

If an attacker obtains the copy of a user database of the system, he is able to bring it into a fully independent environment and utilize automatized dictionary or brute force attacks to determine contained passwords. Specialized tools in combination with high computing power allow for high cracking rates in a relatively short period of time, if protective measures are insufficient. Due to the independency from the source system, such an offline attack happens unnoticed.

The following countermeasure must be implemented, since this ensures best possible protection against offline attacks:

- passwords must be stored using a cryptographic one-way function (known as "Password Hashing") that is recognized as effective for this purpose and is classified as secure at the current state of the art.

Please Note: valid password hashing algorithms are described in Security Requirement Catalog "3.50 Cryptographic Algorithms and Security Protocols".

Explicitly NOT PERMISSIBLE is:

- to store passwords in cleartext
- to store passwords in any format which can be directly backcalculated
- to store passwords using reversible encryption

Please Note:
In this context, "directly backcalculatable formats" refers to those that simply encode the password, without involving a secret key in the transformation process. Since the password will no longer show up as original cleartext after it has been processed, those formats may easily be mistaken to provide confidentiality. Effectively, they do not offer any protection. The enconding is fixed and therefore an attacker can easily make use of it to compute the original cleartext password from the encoded string.
Examples for directly backcalculatable formats are: "base64", "rot13"
"Reversible" are all encryption methods which, using the appropriate key, enable encrypted content to be transformed back into the original content. Accordingly, with reversible encryption there is always the challenge of keeping the key secure and protecting it from unauthorized access. Reversibility is a required fundamental property in many areas of encryption applications, e.g. for transferring confidential messages, but it is counterproductive for storing passwords: a stored password must remain comparable by means of technical methods, but it must no longer be possible to convert it back into plain text in order to protect it as well as possible from unauthorized viewing.
Examples for reversible encryption are: "AES", "CHACHA20", "3DES", "RSA"

Motivation: Without protective measures, an attacker in possession of a user database copy is able to determine masses of contained passwords in short time by merely trying out character string combinations or making use of dictionaries. Passwords stored in cleartext or any backcalculatable format are fully defenseless to an offline attack. 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
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

ID: 3.01-24/8.0

Req 44	If a password is used as an authentication attribute, a protection against online attacks like brute force and dictionary attacks that hinder password guessing must be implemented.

Online brute force and dictionary attacks aim for a regular access interface of the system while making use of automated guessing to ascertain passwords for user accounts.

To prevent this, a countermeasure or a combination of countermeasures from the following list must be implemented:

- technical enforcement of a waiting period after a login failed, right before another login attempt will be granted. The waiting period shall increase significantly with any further successive failed login attempt (for example, by doubling the waiting time after each failed attempt)
- automatic disabling of the user account after a defined quantity of successive failed login attempts (usually 5). However, it has to be taken into account that this solution needs a process for unlocking user accounts and an attacker can abuse this to deactivate accounts and make them temporarily unusable
- Using CAPTCHA ("**C**ompletely**A**utomated**P**ublic**T**uring test to tell**C**omputers and**H**umans**A**part") to prevent automated login attempts by machines ("robots" or "bots") as much as possible. A CAPTCHA is a small task that is usually based on graphical or acoustic elements and is difficult to solve by a machine. It must be taken into account that CAPTCHA are usually not barrier-free.

In order to achieve higher security, it is often meaningful to combine two or more of the measures named here. This must be evaluated in individual cases and implemented accordingly.

Motivation: Without any protection mechanism an attacker can possibly determine a password by executing dictionary lists or automated creation of character combinations. With the guessed password than the misuse of the according user account is possible.

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: 3.01-25/8.0

Req 45	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 46	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 47	If a password is used as an authentication attribute, the reuse of previous passwords must be prevented.

A history of the previously used passwords must be recorded for each user account. When a password change is initiated for a user account, the new password must be compared with this password history. If the reuse of a password is detected, the password change must be rejected. This validation process must be implemented in the system on the basis of technical measures. If a central IAM system is used for user authentication, the implementation can be forwarded to the central IAM system or outsourced there [see also Root Security Requirements Document[i] "3.69 IAM (Identity Access Management) - Framework"].

In general, the password history should ensure that a password that has already been used can never be used again.

However, due to technical limitations, a password history cannot be recorded indefinitely in many IT/NT products. In this case, the following basic rules must be observed:

- a password that has already been used must not be reusable for a period of at least 60 days (measured from the point in time at which the affected password was replaced by another)
- in systems in which the period of at least 60 days cannot be implemented, the longest possible period must be configured. In addition, it must be confirmed by a Project Security Manager (PSM) that the configured period is still sufficient in the overall context of the system with regard to the security requirement.

Annotation:
Some IT/NT products do not offer any technical configuration parameters with which the password history can be linked directly to a time period, but only allow the definition of the number of passwords to be recorded. In such cases, the time period can alternatively be ensured by linking the following, usually generally available configuration parameters. Within the resulting policy, a user can only change his password once a day and, due to the number of passwords recorded, can reuse an old password effectively after 60 days at the earliest.

- Minimum Password Age: 1 day
- Password History: Record of the last 60 passwords used

With this implementation variant, it should be noted that the minimum age for the password should not be more than one day in order not to inappropriately restrict the user with regard to the fundamental need to be able to change the password independently at any time.

Motivation: Users prefer passwords that are easy to remember and often use them repeatedly over long periods of time when the system allows. From the user's point of view, the behavior is understandable, but effectively leads to a considerable reduction in the protective effect of this authentication parameter. With adequate knowledge of the user or information obtained from previous system compromises, an attacker can gain access to supposedly protected user accounts. Particularly in situations in which new initial passwords are assigned centrally as part of an acute risk treatment, but users change them immediately to a previous password for the sake of simplicity, there is a high risk that an attacker will resume illegal access. It is therefore important to prevent users from reusing old passwords.

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

ID: 3.01-28/8.0

Req 48	If a password is used as an authentication attribute, users must be able to independently change the password anytime.

The system must offer a function that enables a user to change his password at any time.

When an external centralized system for user authentication is used, it is valid to redirect or implement this function on this system.

Motivation: The fact that a user can change his authentication attribute himself at any time enables him to change it promptly if he suspects that it could have been accessed by a third party.

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

ID: 3.01-29/8.0

Req 49	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

Req 50	If passwords are used as an authentication attribute, they must not be displayed in plain text during input.

Passwords must not be displayed in legible plain text on screens or other output devices while they are entered. A display while entering must not allow any conclusions to be drawn about the characters actually used in the password.

This requirement applies to all types of password input masks and fields.
Examples of this are dialogs for password assignment, password-based login to systems or changing existing passwords.

Exceptions:

- Within an input field, an optional plain text representation of a password is permitted, provided that this plain-text representation serves a valid purpose, exists only temporarily, has to be explicitly activated by the legitimate user on a case-by-case basis and can also be deactivated again immediately by the latter.
  A valid purpose would be, for example, to allow the legitimate user an uncomplicated visual check, if necessary, that he has entered the password correctly in a login dialog before finally completing the login.
  Such an optional plain text representation of a password must remain fully in the control of the legitimate user so that he can decide on its activation/deactivation according to the situation. In the default setting of the system, the plain text representation must be deactivated.
- The typical behavior on many mobile devices (smartphones) of displaying each individual character very briefly in plain text when entering a password - in order to make it easier for the user to control input - is fundamentally permissible there. However, the full password must never be displayed in plain text on the screen.

Motivation: In the case of a plain text display, there is a risk that third parties can randomly or deliberately spy on a password via the screen output while typing.

Implementation example: When displayed on the screen, each individual character is uniformly replaced by a "*" while entering a 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
- Denial of executed activities
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

ID: 3.01-31/8.0

7. Protecting Data and Information

Req 51	Stored data in need of protection must be protected against unauthorized access, modification and deletion.

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

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

Basically, access to data in need of protection in a system must be fully regulated on the basis of technically implemented authorization assignments and controls.
If such technical access control alone is no longer sufficient to ensure the necessary protection requirements of stored data, or if its effectiveness cannot be consistently ensured, additional cryptographic methods (e.g. encryption, signing, hashing) must be implemented. Cryptographic methods used for the storage of data must be recognised and effective for this purpose and must not have any known vulnerabilities.

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

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

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

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

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

ID: 3.01-14/8.0

Req 52	Data in need of protection must be protected against unauthorized access and modification during transmission.

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

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

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

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

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

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

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

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

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

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

ID: 3.01-15/8.0

Req 53	Key Management System for secure storage of encryption keys must be used. For confidential data, the key management system must be under DTAG's full control.

Attacker who manages to obtain the data from the cloud will most likely deal with encrypted data, but the data would be useless without encryption keys (or sufficiently advanced quantum computers). Therefore, it is important to store all the encryption keys in appropriately secure key management system (KMS). The term encryption key here refers to the

key used for symmetric encryption (where encryption key is the same as decryption key and which is the most common use case for data storage in the cloud), or the private key from the public-private keypair in case of asymmetric encryption; or in other words - it is about protecting the key which would enable attacker to convert encrypted data into clear text.

KMS might come in various flavors:

- under full control of CSP (mostly transparent to customers)
- provided by CSP, but keys managed by customers
- DTAG's own KMS collocated at CSP
- DTAG's own KMS

For open and internal data, use of KMS provided by CSP is generally allowed. The last solution (DTAG's own KMS) is the only acceptable solution and allowed for confidential data since it protects the data also against a rogue cloud provider or intelligence agencies.

Not all systems and data hosted in the cloud will have same secrecy requirements, so it is necessary to select right option for the given system or data. The BoM decision typically includes the level of data which can be hosted in the cloud (and under which conditions), and it would be possible to have a cloud provider approved (or some services of the cloud provider) which, despite technical capabilities, must not be used for storing confidential (or personal) data.

Existing security requirements for KMS still apply, or in other words - use of KMS for the cloud does not change requirements for key strength etc. Also, communication with KMS needs to be encrypted.

This requirement is in line with Requirement 16 from 3.63. ("Operations in DT IT") for those systems where 3.63. applies.

Motivation: A KMS enables fine-grained access control and auditability of key usage, providing a robust security foundation for protecting data with need of protection at rest and in transit. KMSs usually have HSMs on the backend that provide physical protection, secure key storage, and encryption acceleration, helping to ensure the confidentiality, integrity, and availability of critical cryptographic operations.

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: 3.66-53/i35

Req 54	External key generator must be used if hypervisor from CSP is not able to provide sufficient entropy.

Unlikely, but still possible that a certain CSP uses hypervisors which are not offering sufficient entropy to processes running inside virtual machines (i.e. there is no proper random number generator - RNG device inside VM which is linked to host's or external RNG device). In such cases, an external key generator needs to be used for secret generation since the secrets generated inside virtual machines could be recreated because of lack of entropy.

Transferring externally generated secrets in a secure way to the virtual machine would be an additional challenge since it would be impossible to set up a truly secure connection. Therefore, it would be best to avoid usage of such CSP in general.

At the time of writing, hyperscale CSPs (AWS, Azure and GCP) have good entropy source for their VMs. There were some small cloud providers in the past who did not make entropy available to guest VMs by appropriate means (e.g. passing `/dev/urandom` as RNG device).

As an example of an external key generator, a KMS can be used.

Motivation: By using an external key generator, the risk of compromised or weakly generated keys can be reduced, because the process runs outside the system or infrastructure that uses the keys.

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

ID: 3.66-54/i35

Req 55	Used dedicated hardware resources must be wiped when being released.

In case of using bare metal servers or components (GPUs/FPGAs/disks), content of the memory (both volatile or non-volatile) is typically erased by the cloud provider before machine or a resource is given to another tenant.

In addition to relying on the cloud provider's hygiene, a user can, to avoid any doubt, wipe the volatile memory with random data and/or zeroes followed by or reboot, or just shutdown. Data in memory of devices like GPUs or FPGAs cannot be always encrypted and then it is recommended to perform the deletion before the release, especially because these are fast and easy operations.

In case of non-volatile memory, a user can wipe at least headers of the storage devices, or the whole non-encrypted boot disk. As the storage needs to be encrypted in any case, deletion of the key is enough to make data unreadable. In case of SSDs where sectors are typically only marked for deletion and not all of them really overwritten, a user can only rely on controls and procedures performed by CSP. If non-encrypted boot disk for a dedicated physical server must be used due to necessary automation, it must contain NO data with need of protection - no passwords, password hashes, keys, certificates, and especially NO decryption keys for other data.

In unlikely case where the cloud provider does not wipe resources (which can be verified in SOC2 Type 2 report), the user of the specific cloud resources must take care of the complete data deletion. Sudden hardware failures may result in user's inability to wipe the data after use, so it would be best not to use bare metal resources when the cloud provider does not take care of resource deletion.

Motivation: Secure deletion (wiping) helps protect confidential information, protect intellectual property, and maintain customer trust by ensuring that data is properly deleted and cannot be recovered by unauthorized persons.

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

ID: 3.66-55/i35

Req 56	Data lifecycle must be configured through available policies.

Public cloud providers typically offer configuration of data lifecycle through policies, which is most typically used for object storage. Definition of data lifecycle allows automated movement of data through different storage tiers and appropriate restriction levels until data is deleted. Automation minimizes risk of data leaks and improves regulatory compliance. Main benefits include:

- moving data from hot to cold storage tiers (from regularly used data to archived data), which is typically done for economic reasons
- minimal lifetime of the data necessary by regulatory requirements
- maximal lifetime of the data allowed by regulatory requirements
- automated deletion of the data at the end of lifecycle
- support for setting up more complex lifecycle which includes data operations, e.g. anonymization or tokenization of data at certain point in time

Keys used to encrypt and decrypt data need to follow the same lifecycle.

Motivation: Data lifecycle streamlines data management, reduce storage costs, comply with regulatory requirements,

protect information with need of protection, and maintain data integrity throughout the defined process of the lifecycle.

For this requirement the following threats are relevant:

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

ID: 3.66-56/i35

8. Protecting Availability and Integrity

Req 57	The system must be implemented robustly against unexpected inputs.

Data transferred to the system must first be validated before further processing to ensure that the data corresponds to the expected data type and format. This is intended to eliminate the risk of manipulation of system processes and states by appropriately constructed data content. Validation must be carried out for any data that is transferred to the system. Examples include user input, values in data fields, and log contents.

The following typical implementation mistakes must be avoided:

- lack of validation of the length of passed data
- Incorrect assumptions about the format of data
- lack of validation of received data for conformity with the specification
- Inadequate handling of protocol deviations in received data
- Insufficient limitation of recursion when parsing complex data formats
- Insufficient implementation of whitelisting or escaping to protect against inputs outside the valid value range

Motivation: An attacker can use specifically engineered data content to try to put a system that does not sufficiently validate received data before internal processing into an unstable state or to trigger unauthorized actions within the system. The damage potential of such attacks depends on the individual system, but has a theoretical range from uncontrolled system crashes to a controlled execution of specially injected code and the resulting complete compromise of a system.

For this requirement the following threats are relevant:

- Unauthorized access to the system
- Unauthorized access or tapping of data
- Unauthorized modification of data
- Unauthorized use of services or resources
- Disruption of availability
- Unnoticeable feasible attacks

ID: 3.01-11/8.0

Req 58	The system must be protected against overload situations.

A system must have protective mechanisms that prevent overload situations as far as possible.
In particular, a partial or complete impairment of the availability of the system must be avoided.

Examples of possible protective measures are:

- Limiting the amount of memory (RAM) available per application
- Limiting the maximum sessions of a web application
- Limiting the maximum size of a dataset
- Limiting CPU resources per process
- Prioritizing processes
- Limiting the number or size of transactions by a user or from an IP address over time

Note:
A system can usually not protect itself against network-based attacks with extremely high data or packet rates, the so-called "Distributed Denial of Service" (DDoS) attacks. To defend against DDoS attacks, an upstream solution in the network layer is required.

Motivation: Attackers can try to use up the resources of a system with targeted resource-intensive or large-volume requests, so that the system can no longer fulfill its regular tasks or intended task volumes and the availability of the ser-

vices offered is effectively disrupted. Limiting the maximum resources that can be used per request made to the system is a fundamental measure to reduce the impact of such denial-of-service (DoS) attacks.

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

ID: 3.01-12/8.0

Req 59	In overload situations, the system must behave in a predictable manner.

Even comprehensive native protections may not be able to prevent a system from becoming overloaded in extreme situations.

It must therefore be ensured that, in overload situations, the system does not switch to a state that overrides security-relevant functions or properties of the system. Performance losses (e.g. the reduction of the throughput of legitimate network packets or the number of answered server requests per period) are usually unavoidable in overload situations, but the regular functional behavior of the system must be fundamentally preserved.

In extreme cases, this can mean that a controlled shutdown of the system is more acceptable than continued operation in the event of uncontrolled failure of the security functions and thus the loss of system protection.

Motivation: By means of a denial-of-service attack, an attacker can try to overload a system in a targeted manner. If such a system then reacts unpredictably or fails its regular behavior, especially with regard to its security functions, this can open up an extended attack surface for the attacker on functions and data of the system and potentially endanger other linked systems.

Implementation example: A firewall that discards its filter rules in overload situations and forwards all packets without checking would not meet the requirement. In this case, blocking all packets by shutting down the firewall would be more acceptable than failing their regular task of protecting downstream systems.

For this requirement the following threats are relevant:
- Unauthorized use of services or resources
- Disruption of availability
- Unnoticeable feasible attacks

ID: 3.01-13/8.0

Req 60	Systems which need high availability and reliability must be designed and deployed in appropriately redundant way.

Cloud service providers have typically very resilient infrastructure, but that does not apply to every single component in the cloud. Virtual machines and containers die, even more frequently compared to traditionally built counterparts. Systems which need high availability and reliability in the cloud need to adapt their HA mechanisms, so they remain highly available and resilient in cloud environments, i.e. systems and applications need to be designed to handle failures. Using different regions, availability zones or data centers of CSP (or even multiple cloud providers) for redundant application components is preferred choice for ensuring HA of the overall system deployed in the cloud.

One could consider using chaos engineering to ensure that a system deployed in the cloud remains available and resilient during failures (e.g. Chaos Monkey or Simian Army).

Motivation: Redundancy helps mitigate the impact of hardware failures, software glitches, or natural disasters, ensuring uninterrupted service delivery, reducing downtime, and enhancing overall system reliability and performance.

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

ID: 3.66-60/i35

Req 61	Backup solution must be used. And restore function must work.

The requirement can be skipped if there is no need to preserve the data.

Cloud providers typically offer excellent reliability, availability and durability of the data stored in the cloud, except for cases when their data centers burn down (which has happened). They, on the other hand, do not perform backup for customers automatically and it is responsibility of the system owner to implement a backup and restore strategy according to business needs for the given system, with considering the following scenarios:

- accidental or intentional (insider) data overwrite/deletion
- data corruption due to bugs
- ransomware attack
- fire, flood, earthquake or any other disaster which can impact a data center, availability zone or even a region of the cloud provider

It is worth mentioning that mounting a backup disk on a system which is being backed up is probably worst practice in case of ransomware attack because ransomware would lock both the system and the backups. If possible, a backup solution should use a backup storage type which allows writing data once, but no further changes of already written data and with policies which prevent too early deletion of backups. Also, backup system (e.g. backup disk or storage bucket) should generally belong to another tenant with different credentials so that an accident, attacker or malware cannot destroy the backups at the same time as the system (one may think of an analogy to running `'rm -rf /'` with backup disk mounted).

Backup data needs to include all data necessary for business continuity, e.g. "regular data", system configurations, encryption keys (which must be handled separately), logs, ...

Backup without working restore function is not enough, so the overall concept needs to include restore mechanism and restore tests, which are preferably automated. Testing procedure for the restore must be designed in a way that there is no possibility to accidentally cause downtime or data loss to the production system or cause any other kind of data leak.

Depending on the data classification, backups can reside:

- in the same cloud (from same CSP)
- in another cloud (from another CSP)
- in DTAG own premises (would be unusual approach, but still possible)

Motivation: Backups ensure data resilience, mitigate the risk of data loss, and facilitate data recovery in the event of system failures, disasters, or security incidents. Backups serve as a safeguard against accidental deletions, hardware failures, ransomware attacks, or other unexpected events that can result in data corruption or loss.

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

ID: 3.66-61/i35

9. Architecture and Networking

Req 62	Each application/system including environment type (e.g. development, test, production) in the cloud must be separated from others in form of a tenant/project. Each tenant/project must be handled in the same way regardless of the environment type.

Applications must be (logically) separate from others, in a way of separate tenants (projects). Separation is also needed for different environment type so that there is no accidental transfer of data between environment or no unwanted actions (e.g. preventing leaking test data to public, preventing leaking production data to development environment, or preventing accidental deployments of software or configuration to production environment instead to development environment).

The separation also applies to the management part of the cloud application/system. Example could be element managers or CI runners which must be separate per environment type, despite development, test and production part of the application using same (multitenant, of course) CI/CD chain.

This approach enables mandatory segregation of duties and prevents unwanted access to data and systems, while also allows to control communication between different services.

System or application is a very broad term and it may refer to anything between a single microservice to a whole system providing IPTV platform. In this context, it is left to the system owner to define together with security experts the most reasonable and balanced separation, but the extremely rough guidelines are:

- a single microservice does not require its own tenant/project
- huge systems like IPTV might need to be broken down into several tenants based on functional building blocks (e.g. provisioning and management engine, live stream, video library)

In addition to separation of accounts, it is necessary to protect each project/tenant in the cloud in the same way. It is understandable that tenants of production systems will have the highest possible protection, but development and test tenants also deserve the same level of security. The credentials of all tenants need to be protected with due care, they must not end in Git repositories in clear text and they can be only used from approved workstations (corporate managed devices or development workstations). It is also recommended that development and test environments have lower quotas and earlier alarming to detect the misuse in timely manner. This prevents one of most common types of public cloud abuse - cryptomining, most often caused by leaked credentials due to human mistake or use of insecure development workstations.

This requirement is in line with requirements from 3.14. ("Architecture of systems") and applies in addition to them in case both requirement documents are used together.

Motivation: In the case of test and development systems, it cannot be assumed that the system state is sufficiently safe, especially since they are inherently exposed to permanent changes. If the system types are linked to each other or the same tenants/projects are used, there is a risk that unauthorized persons can access production systems and impact data from the test/development environment or that the stability of the production systems and thus the availability of the associated services will be jeopardized. A test and development infrastructure that is completely independent of production makes it possible to carry out changes for test and development systems even during a "frozen zone". This allows security updates to be tested and deployed to production more quickly.

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

ID: 3.66-62/i35

Req 63	Central services for cloud management must be separated in different tenants/projects.

Central services include logging infrastructure, IAM, various security services, reporting etc. They need to be treated

as separate tenants/projects, although their nature may require more privileges than a typical application hosted in the cloud would normally get (e.g. IAM tenant needs to be able to manage users for everybody else in the cloud under the same root account and organization).

Motivation: Segregating central services in specific tenants/projects, can help to minimize the potential impact of security breaches or operational issues in one account on the entire system. Additionally, separate tenants/projects provide granular access controls, better visibility into resource usage and costs, and enable independent management and governance of each service, promoting a more robust and secure overall architecture.

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

ID: 3.66-63/i35

Req 64	Applications must be, as always, built in multitier models.

This requirement is well known best security practice already, but one should be reminded that this approach is still valid for usage of public clouds (or clouds in general).

Traditional applications, most likely during transition period, still need to have usual network separation between tiers, which is safeguarded by firewalls. This requires building a network structure in public cloud which mimics traditional network separation and placing appropriate security controls between tiers. This is typically done by using virtual networks and corresponding security groups.

Cloud native applications need to follow microsegmentation approach for achieving separation between tiers, which typically results in a flatter network with each component safeguarded with security groups and other policies which are available in cloud. Security requirements from other documents still apply.

Example taken from current version of Container Security Requirements:

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

Motivation: Dividing the application into different layers, such as presentation, application logic, and data storage, helps implement security measures and protection at each level. It is possible to update or change certain levels without affecting the entire system.

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: 3.66-64/i35

Req 65	Resource separation must be performed by using virtual private clouds (VPC). Network communication of an application/service in the cloud must be restricted by network filter elements. Advanced network protection mechanisms offered by CSP must be used.

Virtual private cloud is a concept for separation (primarily network separation) of resources used by a system (or a group of systems) in the public cloud from other resources in the public cloud. It is implemented through set of policies which define connectivity and reachability of resources, even throughout different data centers, availability zones or regions of the public cloud. To achieve easy and effective implementation of basic network perimeter, one must use VPC(s). Using VPC can reduce the need for having separate VPN connections to each data center of CSP and can be used to set appropriate routing tables for controlling network traffic. VPC concept also allows cloud consumers to bring their own IP address ranges for resources used in that cloud.

The most common implementation of network filter elements are security groups. Security groups in cloud are, in addition to VPC, cornerstone of limiting possible communication paths in the cloud. As for any other firewall which be found in traditional deployments, security groups must be made as restrictive as possible to allow only necessary communication paths.

In addition to security groups, cloud providers typically offer DoS, DDoS protection, rate limiting and other network protection mechanisms. Such mechanisms must be used, especially for Internet facing services running in the cloud. Depending on the cloud provider and offered advanced protection mechanisms, additional detection and/or protection mechanisms might be necessary.

Certain cloud providers do not use the term VPC but offer setting it up through means of virtual networks.

Motivation: VPCs add an extra layer of security because they provide separation at both the network level (network and routing) and the access layer. Only allowed users can deploy systems within a specific VPC and configure the connection path for them.
This separation helps prevent unauthorized access, limit the impact of security incidents, and streamline network management, allowing for better resource allocation and an improved overall security posture.

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: 3.66-65/i35

9.1. Cloud Connectivity

Req 66	The accessibility of activated services must be restricted.

In principle, a service provided must be completely deactivated on all interfaces of the system through which accessibility of the service is not required for the proper operation of the system. The deactivation is primarily to be implemented by a corresponding configuration of the service or operating system. In cases where the available configuration options do not allow deactivation on individual interfaces, a local filter ("Host Firewall") may instead be used on the system to block access to the service via unnecessary interfaces.

The accessibility of a service via the required interfaces must also be restricted to legitimate communication partners. The restriction must be implemented by a corresponding configuration of the service or operating system or by means of a local filter ("Host Firewall"). Alternatively, this task may be outsourced to a network-side filter element, provided that the system is located in a separate network segment and communication with this segment is only possible via the network-side filter element.

Motivation: By deactivating services on interfaces through which accessibility is not necessary, as well as by restricting possible communication partners, the attack surface offered by a system can be greatly reduced.

Implementation example: An SNMP service used to monitor a system is enabled exclusively on the dedicated management network interface of the system. A firewall also regulates that only the legitimate monitoring system of the infra-

structure environment can reach this service.

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

ID: 3.01-6/8.0

Req 67	Management services of applications/services in the cloud must not be widely reachable.

SSH and RDP are most common remote management services and they play an important role in securing cloud applications. An attacker who manages to get access to management services can compromise the whole system or even more systems with very low effort. Limiting the attack surface is an important step to mitigate possibility of such attacks. Management services of an application in the cloud must not be widely reachable. They should not be reachable over Internet, however, if connectivity over Internet is necessary, the access must be restricted to trusted parties and/or centrally managed and monitored jump servers need to be used as an entry point.

Use of immutable images typically requires no management access, therefore, it reduces attack surface.

Motivation: Making common remote management services widely accessible exposes them to potential threats from malicious actors attempting to gain unauthorized access to systems or launch password-guessing attacks. Limiting the attack surface is an important step to mitigate the possibility of such attacks.

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: 3.66-67/i35

9.2. Cloud Connectivity - External Connectivity

Traffic going out of the cloud or coming in to our premise within the cloud.

Req 68	Dedicated physical or dedicated virtual connection must be used for connectivity between application in the public cloud and on-premise resources of Deutsche Telekom, or between applications hosted in different public clouds. Internet access is allowed only for workloads which are both cloud and Internet native, but they must be explicitly approved by the security department.

Dedicated physical connections are typically fiber (or copper) cables connected between CSP and DTAG premises. Dedicated virtual connections are VPN connections between a VPN endpoint inside DTAG premises and VPN endpoint in the public cloud. In case of connectivity between two cloud providers, VPN is also needed.

An application running in cloud which requires often or permanent connectivity (one time data transfer only may utilize other means if suitable) to/from DTAG resources, whether running in DTAG own premises or in another public cloud, needs to utilize VPN set up between public cloud where the application runs and DTAG premises (or another public cloud). It is not meant that each application should utilize a separate VPN connection, but that applications should use same connection which is set up once between cloud provider and DTAG for all resources belonging to DTAG in that CSP.

Preferred methods are in the following order:

1. dedicated physical connection (typically more reliable if used exclusively)
2. dedicated virtual connection
3. VPN over Internet (less reliable)
4. (restricted) plain Internet connectivity; approval from technical security advisor necessary

VPN connection which uses shared backbone network with other customers or Internet is typically less reliable in case of congestions (e.g. due to DDoS attack) and that can affect reachability of services.

Plain Internet access is allowed only for fully cloud and Internet native workloads and it is necessary to restrict address ranges allowed for communication to minimum which is possible. This method is mainly reserved for trials, tests or clouds where DTAG has extremely small footprint.

These dedicated connections may be encrypted, but even in that case, this does not change a requirement that all non-open (non-public) data must be encrypted during transport because data flow must be encrypted in all places. VPN must be always encrypted. Redundancy need to be in line with expected availability
to prevent denial of service.

Motivation: A dedicated connection to the cloud is used to achieve a reliable, secure, and high-performance connection while improving privacy and reducing latency compared to public internet connections.

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

ID: 3.66-68/i35

Req 69	Connections between public clouds and resources and networks inside DTAG premises must be protected by an independent network filter element, located inside DTAG premises and under DTAG control. A service running in the cloud must not have direct access to Internet and internal DTAG networks at the same time (the functionality must be split over multiple VMs, containers or in general - multiple tiers).

Connections between public cloud and internal networks of DTAG (e.g. CN-DTAG, Intranet) must be protected with a security device which needs to, depending on specific use case, provide:

- stateful firewalling (as bare minimum)
- web application firewall
- application level gateway
- API gateway
- DoS/DDoS protection
- ...

The security device in this context may contain out of several physical and virtual devices acting as a single unit. Protection provided by the device must be in general bidirectional - mainly protecting DTAG's networks from threats from the public cloud, but also protecting resources inside public cloud from malware and malicious insiders inside DTAG's networks. Since it is expected that applications hosted in the public cloud will utilize security groups and other security features offered by the public cloud itself, controls necessary on the security device do not to be very detailed for the connectivity from DTAG internal network towards the network in the public cloud. For the other direction, protection of DTAG's internal networks from threats from the public cloud, implementation must be done fully on the security device.

For HTTP(S) based services, the provided device should protect against OWASP Top 10 web vulnerabilities at minimum.

The configuration of the device should be automated in a way so that it can cope with rapid provisioning cycles typically found in highly automated cloud environments (e.g. it should be handled so that it can cope with dynamic IP address assignments in cloud).

Internal networks of DTAG are not built in modern way with microsegmentation in mind, and applications deployed in the cloud need to follow guidelines for connectivity to existing DTAG internal networks if they need such connectivity. Applications in the cloud are not allowed to have direct Internet connections and connection to DTAG internal net-

works at the same time because there would be no (at least) 2 layers of security protection in that case. This would allow attackers much easier infiltration into DTAG network and resources through a backdoor placed, intentionally or unintentionally, in an application running in the cloud. For special use cases, e.g. management and monitoring of an Internet facing application running in the cloud, an appropriate solution has to be determined on case-by-case basis.

Motivation: DTAG's internal networks are not built in a modern way with microsegmentation in mind, and applications deployed in the cloud must follow guidelines for connectivity to existing DTAG internal networks if they need such connectivity. Applications in the cloud must not have direct internet connections and connections to DTAG's internal networks at the same time, as in this case there would be no (at least) 2 layers of security. This would allow attackers to penetrate the DTAG network and resources much more easily through a backdoor that is intentionally or unintentionally placed in an application running in the cloud.

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

ID: 3.66-69/i35

9.3. Cloud Connectivity - Internal Connectivity

Communication within the cloud and its security.

Req 70	Traffic between resources within a public cloud provider must use the most secure communication method available.

Communication between cloud services can be implemented in multiple ways, while using a public endpoint is the most insecure way and using private endpoints communication with security groups can be seen as the most secure. The most secure option must be used.
The following table gives an hierarchic overview of possible options (from most secure to insecure):

1. Communicate inside a single VNET/VPC
2. Use private endpoints with security groups of the service or cloud resource (e.g. VNET integration of service, VPC endpoint, private link)
3. Use dedicated connections like VNET peerings/VPC peerings (inside Telekom premise only),
4. Use public endpoints (most insecure)

Motivation: Using the most secure way to interact with cloud services is vital to protect data with need of protection, comply with regulations, and mitigate security risks.

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: 3.66-70/i35

Req 71	Network filter elements must be used to segregate resources at the network level.

Network filter elements (security groups, cloud firewall, security tags) must be used to separate resources at the network level.
The need for protection of resources results from various factors such as data processed and stored, exposed services and applications used. Similarly, resources on which personal data is processed must be protected from unwanted access or data flows from the same network or other networks by state-of-the-art measures (e.g. security groups and firewalls).

Besides security groups also other mechanism of the cloud like network access control lists (NACLs) can also be used

as an additional measure.

Motivation: It is more likely for a less protected system to be compromised. This must not result in other systems with higher protection requirements being easier attacked by this compromised system in the network.

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: 3.66-71/i35

Req 72	Default rules for network filter elements (any to any communication) must not be used.

Rules for network filter elements like security groups or cloud firewalls are divided into rules for ingress and egress communication. The default egress/ingress rule allows communication for every port and address. This is not reduced to the minimal required communication. Therefore, security groups/cloud firewall and rules depending on the application must be created and configured.

Motivation: Permissive communication contradicts the need-to-know principle.

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

ID: 3.66-72/i35

Req 73	Only necessary communication must be configured in network filter elements and filter rules.

Rules for network filter elements (e.g. Security Groups, Cloud Firewall) are divided into rules for ingress and egress communication. By default, egress rules allow communication for every port and address, this have to be reduced to the minimal required communication. Every necessary ingress communication has to be defined with a specific allow rule.
It is recommended to use references to a Security Group (names) instead of specific IP addresses or IP ranges in the Security Group rules.
Security Groups are stateful.
When using dual stack (IPv4 and IPv6), it is important to note that the rules must be set for both protocol versions.

Motivation: Permissive communication contradicts the need-to-know principle.

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

ID: 3.66-73/i35

10. Logging and Monitoring

Req 74	The system clock must be synchronized to an accurate reference time (Time Standard).

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

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

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

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

Implementation example: some valid time reference sources:

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

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

ID: 3.01-32/8.0

Req 75	Security-related events must be logged with an exact timestamp and a unique system reference.

Security-relevant are any events that may have an impact on the information security (confidentiality, integrity, availability) of a system or indicate its misuse. Systems must log the occurrence of security-related events.
In order to enable the assignment and evaluation of these events, they must be logged together with a unique system name (e.g. host name, IP or MAC address) and the exact time of occurrence ("time stamp"). At a minimum, the timestamp of a logged event must include the following information:

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

The security relevance of specific events depends on the type and function of the system as well as the data processed in it. Therefore, it is not possible to specify a uniform list of events that are relevant to security. Relevant events must be identified according to the system. The following list contains typical security-related events and is accordingly to be understood as a basic guide for the identification of security-relevant events for a particular system.

Security-related Event	Information to log to the event
Failed login attempts	• affected user account • number of failed attempts • source of access (IP address, client-ID/client-name, local console, ...)

System access via user accounts (including system accounts or technical user accounts)	• affected user account • Access start/end time (login/logout) • source of access (IP address, client-ID/client-name, local console, ...)
Modifications of user accounts (including system accounts or technical user accounts)	• user account used to make the modification • user account to which the modification was made • type of modification made (create, activate, deactivate, delete, password reset, ...)
Changes to role assignements/permissions of user accounts (including system accounts or technical user accounts)	• user account used to make the modification • user account to which the modification was made • name of the affected role, type of the affected permission • type of modification made (role/permission added, role/permission removed, ...)
Use/initiation of critical system functions	• user account used to make the use • name/type of system function used
Access to sensitive data processed in the system	• user account used to access the data • Name of the affected data field (the contents of the data field should not be logged to prevent data leakage via log files) • type of access (read, modify, delete, ...)
Abnormal increase in the utilization of system resources, such as memory, CPU, network, or disk access, over an extended period of time	• threshold of the affected system resource • duration of exceeding the threshold • threshold exceedance Depending on the individual system, sensible threshold values must be defined.
Other events that may provide evidence of attempted or successful attacks against the system ***)**	• individually depending on the type of event

***)** *This requires identifying the most important forms of attack that the system can typically expect and ensuring that events relevant to their detection are actually logged by the system. As a rule, this can only be guaranteed by the developer/manufacturer of the system.*

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

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

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

Implementation example: Security relevant log sources, which needs to be activated:

- AWS: CloudTrail and VPC flow logs.
- Azure: Activity Log and NSG flow logs.
- GCP: Cloud Operations Suite, Cloud Audit Logs and VPC flow logs.

- OTC: Cloud Trace, VPC flow logs

Security relevant log sources for applications to consider:

- AWS: CloudWatch
- Azure: Log Analytics, Applications Insights
- GCP: Cloud Logging, Cloud Monitoring, Error Reporting
- OTC: Cloud Eye

ID: 3.66-75/i35

Req 76	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 77	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 78	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 79	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 80	Security relevant events in the cloud must be logged and forwarded to a separate log storage and to the Security Information and Event Management System (SIEM) of the responsible SOC to allow 24/7 security monitoring.

General security requirements on logging still apply and are also relevant for cloud environments. Security relevant events like cloud alerts, cloud management and API call logs, configuration changes, access to accounts, access to certain data with need of protection, network logs, system logs etc., must be logged in proper way by using logging and monitoring infrastructure of CSP, for all service types used - IaaS, PaaS and SaaS. Supportive systems like IAM and KMS used for the cloud must be also included in the logging concept. And lastly - existing other security requirements specify that all systems, including those deployed in the cloud, like operating systems running in virtual machines, databases, server software and containers, must provide logs.

Logging and forwarding of security relevant events must be configured by the landing zone for all tenants/accounts/systems in a way that it happens immediately after generation and cannot be de-activated by admins/users of the tenants/accounts/systems. If this cannot be ensured by the landing zone, the landing zone has to offer solutions that the tenants/accounts/systems must use (e.g. by documentation on configuration of log forwarding, use of pre-configured images) and established controls that forwarding is enabled.

Monitoring of security relevant events is a prerequisite for detecting attacks on cloud resources like data exfiltration or resource consumption. The monitoring must be 24/7 for all environments (including test and development environments), because in addition to traditional attacks with malware and ransomware on production environments, cryptocurrency mining by attackers in development environments over the weekend is also a threat which can cause huge financial damage.

Forwarding of logs to both a separate log storage and to SIEM at the same time applies to those cases where SIEM does not store logs for necessary period of time, like it is the case for CDC (Telekom Security Cyber Defense Center - CDC is commonly used in Germany) who perform security monitoring for DT IT. In that specific case, the separate log server stores the logs for forensic analysis and to have evidence in legal actions against attackers.

Logging policy must follow laws, regulations, and company policies. According to current laws, regulations and company policies, logging of such events is permitted only for defined use cases and forbidden for others, like work control of employees.

Examples of some event types/logs:
- cloud alerts: GuardDuty for AWS or SecurityCenter in Azure
- cloud management and API call logs: AWS CloudTrail or Azure Activity Logs
- network logs: all connections to and from the Internet must to be logged by proxy and if feasible all relevant network traffic - VPCFlowlogs
- system logs: syslog/auditd and winlog

In case of a system where 3.63. ("Operations in DT IT") applies, Requirement 25 from 3.63. applies as more specific requirement.

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.
Only if log data are analyzed, threats can be identified. Analyzing needs to be done 24/7, because attackers do not

stop their attacks outside business hours.

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

ID: 3.66-80/i35

Req 81	Incident response process for reacting appropriately to security events and incidents must be established, including provisioning of forensic evidence.

Clearly defined incident response process is mandatory to minimize the impact of an ongoing or successful attack. The process defines responsibilities, contact persons and tasks for attack mitigation, analysis of potential or real damage, procedures to recover system security and preserve evidence for law enforcement.

Consideration of both sides - a system deployed in the cloud and APIs available from the cloud provider is necessary for an incident response process, e.g. in case of dealing with hacked virtual machines it may be beneficial to create snapshots of hacked VMs through APIs of the cloud provider, so they can be preserved as an evidence.

In alignment with the responsible SOC, the landing zone operators must provide incident response capabilities towards SOC with appropriatelly short reaction time and 24/7 availability in order to be able to directly stop attacks and prevent further damages. The capabilities must cover all environments (including test and development environments) and access to all tenants/accounts/resources must be possible even without system owners (e.g. via break glass accounts). Processes must be defined, established, and regularly tested that allow to get always back control over compromised resources and to properly collect the forensic evidence. Examples include:

- Isolation of compromised virtual machines
- Update other relevant security groups
- Revoke misused users, roles or rights and
- Stop malicious services or serverless functions
- Export further relevant data such as copies of virtual machine images, containers and storage to the forensic laboratory of responsible SOC

In case when the attack was not recognized by SOC but the system owner, responsible SOC must be informed about security events and incidents and it is expected that the responsible SOC will provide enough information on how to proceed with the investigation.

An illustration of a few good and bad practices is listed below.

Examples for proper forensics:

- Creating a snapshot of infected containers and/or virtual machines with all relevant logs.
- Isolate impacted workloads and keep them fully isolated (full network and storage separation from rest of healthy environment) and wait for further instructions from a security officer.

Examples of bad forensics or destroying forensics data, such actions must NEVER happen:

- Deleting containers and/or virtual machines because everything is already logged.
- Keep everything running "normally" (as is) when it gets infected/compromised.

In case this requirement is fulfilled, Requirement 8 in 3.63. ("Operations in DT IT") can be skipped.

Motivation: Security incidents must be dealt with quickly and competently in order to detect and avert potential damage to the company, employees and customers.

For this requirement the following threats are relevant:

- Denial of executed activities
- Unnoticeable feasible attacks

ID: 3.66-81/i35

11. References

Some references from the document:

https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-144.pdf

https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-86.pdf

12. Glossary

Public, private, community and hybrid cloud definitions, as well as definitions for IaaS, PaaS and SaaS, are taken from NIST.

Public cloud
The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.

Private cloud
The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g. business units). It may be owned, managed, and operated by the organization, a third party or some combination of them, and it may exist on or off premises.

Community cloud
The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.

Hybrid cloud
The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g. cloud bursting for load balancing between cloud).

IaaS
The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g. host firewalls).

PaaS
The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

SaaS
The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.

CSP stands for Cloud Service Provider. Examples are Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP).

VPC stands for "Virtual Private Cloud," and it is a virtual network environment within a public cloud service (e.g., AWS, Azure, GCP) that allows users to provision and manage their cloud resources in an isolated and private network.

VNET stands for "Virtual Network," and it is a virtualized network infrastructure within the Azure cloud platform. VNET allows users to create and manage their isolated network environments, define IP address ranges, and control traffic flow between Azure resources and the internet or on-premises networks, ensuring secure and efficient communication within the Azure cloud environment.

Personal user account is a user account used by a person (human being).

Technical user account is a user account used by a machine, also known as M2M (machine to machine) user account.

It is mainly used in M2M communication scenarios where communication is established between two machines, typically without human interaction.