Security requirement

Private Clouds

Deutsche Telekom Group

Version	1.0
Date	Dec 1, 2021
Status	Released

Publication Details

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

Reuterstrasse 65, 53315 Bonn
Germany

File name	Document number	Document type
	3.86	Security requirement

Version	State	Status
1.0	Dec 1, 2021	Released

Contact	Validity	Released by
Telekom Security psa.telekom.de	Dec 1, 2021 - Nov 30, 2026	Stefan Pütz, Leiter SEC-T-TST

Summary
The goal of this document is to provide a security baseline for architecture of secure private cloud infrastructure, especially for modern cloud-native workloads. In addition, a part of requirements in this document is intended for cloud consumers which use such cloud platform. Requirements covering both cloud infrastructure and workloads in the cloud are intended for modern cloud deployment, cloud-native workloads and are compatible with zero-trust approach. The requirements cover different sizes of cloud environment and different cloud offering models, although the focus is on most typical use cases which are small to medium cloud deployments with IaaS offerings. The requirements are generally technology agnostic, but wherever possible the references are given towards open source solutions.

Table of Contents

1. 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.

Private cloud installation is a set of hardware resources like servers, switches, routers, firewalls, and possibly external storage components which are driven by automation software enabling programmable use of available resources.

Scope

This document covers the following topics:

- security requirements for cloud infrastructure, including compute, storage and network resources (also covering PaaS and SaaS)
- security requirements for cloud management and orchestration
- security requirements for tenants (cloud users)

for the following type of cloud environments:

- private cloud installations, owned and operated (or controlled) by DTAG for internal workloads, with workloads also operated (or controlled) by DTAG (which means, no direct service offering to non-DTAG personnel), irrespective of size (central "big" cloud environments or edge "small" cloud environments)

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

- Public cloud

Software in scope:

- OpenStack
- VMWare Cloud
- Other solutions which are like abovementioned

Software out of scope of this document:

- Docker
- Kubernetes
- Other container technologies

Upfront clarifications

Term node is often used interchangeably with server.

Term server or node are typically also referring to multiple physical servers, but a singular was used in the text for clarity.

Examples are given in most cases for OpenStack since it is most widely used open source cloud solution with entire documentation publicly available.

As very usual for the cloud world (*aaS world), the word service refers most often to the cloud service offered by the cloud to cloud consumers. However, the term service is sometimes used to denote internal service as a general piece of software running on the server (e.g. daemon). The word function and feature denote necessary functionality of the cloud or cloud service.

2. Cloud Subtypes

In next part, there will be frequent use of the following terms:

- (normal) cloud - cloud where each server runs exactly 1 type of service (i.e. server runs compute, storage, networking, or cloud management function)
- converged clouds are clouds where more than one function of cloud runs on the same physical server (example: combined compute and storage server)
- hyperconverged clouds are converged clouds where cloud management functions run alongside cloud functions offered to tenants (example for OpenStack: OpenStack control services like Keystone and Glance run on the same server(s) with storage and/or tenant virtual machines)

2.1. Cloud Function Separation

Req 1	One physical server must not run more than one type of cloud function.

Cloud functions used for cloud management, compute, storage and network functions must run on different physical servers. Mixing multiple cloud functions on the same physical server is allowed for converged and hyperconverged cloud environments.

Motivation: Physical separation of different cloud functions reduces the risk of attack propagation and increases attack detection if used with other security measures. Cloud installations are typically big, and each cloud function (management, compute, storage and networking) sometimes itself requires more than 1 physical server due to its scale. Term management cloud functions also is sometimes called control plane, while compute, storage and networking are sometimes called data plane. This type of separation also improves availability because a single server outage will impact less cloud functions.

Implementation example: Installation of these functions on separate physical servers (or physical server sets):

- management cloud functions (MySQL, Keystone, Glance API, Nova API, ...)
- storage cloud functions (OpenStack Swift)
- compute (Nova compute, KVM, OpenVSwitch)
- network (Neutron)

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

For this requirement the following warranty objectives are relevant:
- Availability

ID: 3.86-1/1.0

Req 2	In converged clouds each cloud function must run in a separate virtual machine.

Cloud functions used for cloud management, storage and network functions must run in separate virtual machines. Mixing multiple cloud functions in the same virtual machine is allowed only for hyperconverged cloud environments with limitations.

Customers must not under any circumstances be able to directly work with resources not running inside VMs, e.g. object storage gateway is not allowed to be hosted on bare metal which runs other cloud functions if there is no virtualization.

There is no exact definition for converged clouds, but all which have up to a dozen servers can fall into this category (or alternatively - sometimes mentioned as one rack deployments).

Motivation: Hypervisor separation of different cloud functions is not necessary as effective as physical separation, but still offers decent level of risk reduction of attack propagation and increases chances of attack detection.

Implementation example: Typical converged cloud setup would include mixing storage and compute functions on the same physical machines. In this case, storage functions require higher level of protection, so data encryption at rest and in transit for all storage functions is mandatory. Containers are generally not considered as sufficient, although in some specific cases, other security measures might offset for lack of hardware-based virtualization.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-2/1.0

Req 3	In hyperconverged clouds, each cloud function must be logically separated and use different encryption keys for each function. Keys must not be stored in the set of machines used for cloud environment.

Additional security measures, including intrusion detection/prevention systems (IDS, IPS) may be needed for hyperconverged clouds.

Although there is no strict definition of the term hyperconverged cloud, this is kind of setup is intended for very small cloud installations (e.g. up to 5 servers) without plans of further expansion. It is also common for edge cloud installations which host 1 tenant only.

Motivation: Attacker might be able to gain access to data but would not be able to do anything with data from other virtual machines, including cloud control functions. A direct attack against other machines would be more easily detected if appropriate security measures are applied on hypervisors and inside virtual machines.

Implementation example: Separation of cloud functions in hyperconverged clouds is commonly available through running different kinds of cloud functions in separate virtual machines. Each cloud function (and virtual machine) must use its own encryption key for data storage. In addition, key management system needs to be in place and the process must be made in way which guarantees that a virtual machine is able to retrieve only and only its own key necessary for runtime.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-3/1.0

2.2. Cloud Infrastructure Management and Orchestration

In some cases, multiple cloud environments (data centers) may be coupled together in one big cloud. In such case, there is a need to have so called umbrella management, which has to be hosted in secure location(s) (e.g. one or more TMDC sites, although one is sufficient, more are recommended for redundancy) and connected over a secure network (e.g. DCN may be used while VPN over Internet is allowed but not encouraged since there is no guaranteed availability).

If only federation between cloud environments is used, the connectivity between management systems of all cloud environments must be accordingly secured.

Req 4	Central cloud management systems must be hosted in secure locations and connected via secure network.

Hosting central management systems in secure locations and connecting cloud environments to central cloud management system via secure network allows for better protection and availability.

Term secure network here is best described as a network with encryption and appropriate security filtering, so a VPN could be one implementation, the other one could be a set of tunnels (or access lists and firewalls) with encryption and verification of both clients and servers through certificates.

Motivation: If a central cloud management is used, an attack can be much more devastating since an intrusion into central management system could be sufficient to compromise all cloud environments.

Implementation example: Example of some centralized systems used by many cloud environments include KMS (Key Management System) and central identity provider.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-4/1.0

Req 5	In case of central cloud management, critical and locally needed functions must be hosted in each cloud environment.

All functions crucial for functionality of a specific cloud environment must be hosted locally in that same cloud environment to mitigate a risk of losing manageability of all cloud environments due to the failure of central cloud management system.

Motivation: Without crucial functions hosted locally, a remote site could stop working if a connection to the central cloud management is temporarily lost.

Implementation example: Some examples of such crucial functions include Keystone, Glance, and network boot servers.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-5/1.0

2.3. Large and Small Deployments

While larger cloud installations (in central locations) are not expected to be hyperconverged, converged scenario are to be evaluated on case-by-case basis. Extremely small cloud installations, typically known by name edge clouds, (example: < 5 servers) may use hyperconverged concept, but security details need to be adjusted on case by case basis, depending on many factors including physical security of a location where cloud is located.

2.4. Undercloud And Overcloud

Some cloud vendors deliver the cloud platform in form of 2 cloud environments, with a small cloud used for deployment of larger cloud which is in turn used by real cloud consumers. The approach is taken to minimize the amount of technologies and software necessary for the maintenance of the whole cloud platform. One notable example is RedHat's OpenStack.

Undercloud is minimalistic OpenStack installation on a single machine or on just a few machines which treats test rest of the hardware available as its own cloud.

Overcloud consists of the rest of hardware with a separate OpenStack installation (which is fully managed by undercloud). All servers of overcloud (all of OpenStack control, compute, or storage nodes of overcloud) are treated as compute (bare metal) nodes in undercloud.

This draws a need for undercloud to have full administrative M2M access to all machines in overcloud so they can be managed without human interaction. It is typical to see that undercloud management nodes can access all overcloud machines via passwordless and non-interactive SSH connections (either to root user or to normal user which has passwordless sudo). In such cases, some security requirements from this document which are about controlling access to overcloud hardware by mandatory use of central IAM do not necessary apply since all users need to first go through undercloud.

3. Cloud Infrastructure Network and Cloud Management

Cloud infrastructure network (also known as underlay network) of the cloud is cloud internal network necessary for IP connectivity between all hardware components and cloud software, including:

- compute nodes
- storage nodes
- network nodes
- server management (BMC - Basedboard Management Controller, e.g. IPMI or Redfish)
- network devices (switches, routers, firewalls, ...)
- cloud management functions

This does not include tenant networks.

Req 6	Connectivity between cloud infrastructure network and external networks must be established through device(s) with stateful L3/L4 filtering.

Device (router, L3 switch, firewall, server) in this context is functionally firewall with stateful L3/L4 filtering. This device represents the demarcation line between distribution network and cloud infrastructure. Most of the traffic for cloud infrastructure will traverse inside the cloud environment. This device must implement rough filtering (access control lists - ACLs) to block malicious traffic coming from outside and to control outgoing traffic, however, it is not expected that this device also controls tenant traffic.

Cloud infrastructure management functions must be protected from outside traffic. Only a minimal set of cloud infrastructure components need to be accessible from outside.

Motivation: Control of incoming and outgoing traffic can block attempts of data extraction if some components of cloud get compromised.

Implementation example: Examples of cloud infrastructure components which typically do not need to be accessible from outside:

- connectivity to BMC (IPMI/Redfish) interfaces of servers from outside is not allowed
- direct connectivity from outside to internal storage network of the cloud is not allowed
- access to necessary cloud APIs is allowed while still requiring additional controls, like WAF (Web Application Firewall)

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-6/1.0

Req 7	Cloud infrastructure networks of cloud infrastructure must not be directly reachable from external and tenant (overlay) networks.

In broadest term, direct IP connectivity from external networks or tenant (overlay) networks to cloud infrastructure is not allowed.

This is an extension of previous requirements with same goal.

Motivation: Control of incoming and outgoing traffic can block attempts of data extraction if some components of cloud get compromised.

Implementation example: Access to internal systems of a cloud infrastructure must be done in a controlled and protected way over other security devices like higher layer firewalls, WAFs (web application firewalls) or proxies (not transparent naturally).

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-7/1.0

Req 8	Different traffic types in cloud infrastructure must be separated from each other in dedicated logical or physical networks.

At least the following traffic types in the cloud infrastructure must be separated by means of logical (e.g. VLAN) or physical separation:

- production traffic (network carrying traffic from tenants)
- infrastructure management and orchestration traffic (management and orchestration of the cloud itself)
- regular storage traffic (between storage and compute nodes)
- internal storage traffic replication (between storage nodes, in case of Ceph this is known as replication network)
- BMC network (server management through dedicated IPMI/Redfish port)
- generic device management traffic (mainly switch and router management)

If bare metal as a service (BMaaS) is offered, further separation of traffic is necessary.

Listed traffic types are very distinctive in terms of their functionality, performance/availability requirements and protection levels.

Motivation: Separation of different traffic types would hinder an attacker in progressing further, while at the same time could improve detection capabilities if other security measures are appropriately used.

Implementation example: In addition to the listed traffic types which need to be separated, prioritization of certain traffic types (example: higher priority for infrastructure management and orchestration traffic over production/tenant traffic) also enables easier handling of congestions which are result of ([D]DoS) attacks or misbehaving tenant workloads.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-8/1.0

Req 9	If there is no out-of-band management network, the infrastructure management traffic, orchestration traffic, BMC and generic device management traffic must have higher priority to guarantee reachability and manageability of cloud infrastructure in case of network overload.

Prioritization of traffic in these networks must be performed in all necessary places. This applies to all devices (network related or servers) which operate in oversubscription mode - which is very common for cloud environments.

Motivation: The manageability of the cloud infrastructure has the highest priority in cloud environment. In overload cases (caused by bugs, errors, regular network congestion or [D]DoS attacks), manageable infrastructure enables troubleshooting, incident analysis, mitigation, and removal of root causes, which in turn enables recovery of workloads running in the cloud.

Implementation example: Prioritization of traffic with either through QoS or with rate limits (or both).

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-9/1.0

Req 10	The cloud infrastructure management services and systems must be protected by MFA for human access.

Implementation of MFA (minimum of 2FA) for authentication is mandatory for managing highly sensitive infrastructure.

Also, it is worth mentioning that direct access to management systems of the cloud environment or any other part of cloud infrastructure is generally not allowed. It is common that cloud is managed by M2M accounts from a central point (e.g. Ansible server).

Where MFA is not possible due to the technology choice, a jump host must be deployed in front which deals with AAA aspects of human users, including MFA.

(This requirement does not contradict with existence of emergency access accounts, also known as break glass accounts.)

Motivation: Authentication via only one factor is not considered sufficient for managing highly sensitive infrastructure.

Implementation example: Use of MFA like TOTP (Time-based One-Time Password) or hardware based devices.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-10/1.0

Req 11	The cloud infrastructure management services must be protected by mutual strong authentication for machine-to-machine (M2M) accounts.

As an equivalent of MFA for human users, a strong authentication for machine-to-machine users must be implemented. This typically includes authentication of both parties in the communication.

Motivation: Authentication via simple M2M authentication (e.g. simple tokens) is not considered sufficient for managing highly sensitive infrastructure.

Implementation example: One possible example might be:

- server presenting its authenticity via certificate
- client using a certificate or token which can be proven and has maximum lifetime set to a sufficiently short period

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-11/1.0

3.1. Cloud APIs

Req 12	Access to cloud API endpoints must be controlled and protected by higher layer firewalls or proxies, both for access from external networks and tenants. Communication must be encrypted. Authentication is mandatory.

Per requirement which mandates absence of direct IP connectivity from external networks and tenant networks to cloud API endpoints, access would be necessary for management functions of applications running in the cloud and for external people and/or machines. The access to cloud API endpoints must traverse through higher layer firewalls (above L4 - TCP), WAFs or proxies.

Communication from end-clients (coming from both external and tenant networks) to firewall/WAF/proxy and from firewall/WAF/proxy to the exposed cloud API endpoints must be encrypted, including both-way certificate verification to avoid man-in-the-middle attacks. Internal cloud APIs must be naturally not reachable by tenants.

Alongside all regular APIs, this applies to Nova metadata server as well. In typical modern OpenStack deployments, there is Nova metadata proxy which fulfills this requirement for Nova metadata server. As of today, there is no encryption available for Nova metadata, so this endpoint does not have to be encrypted.

Specific web application checks need to be used for web-based UI (e.g. dashboards).

Strong authentication must be used for getting access to all API endpoints, according to existing security requirement documents about web applications and web services (e.g. document 3.02 - "Web Services").

Motivation: Cloud API endpoints have big attack surface with relatively low built-in protection against unexpected input and potentially high impact if breached.

Implementation example: Securing cloud APIs them with proxies, limiting API requests (including restricting possible API calls to only necessary) and filtering input is crucial to hinder possible attacks through cloud APIs. Firewalls, WAFs and proxies can be integrated with logging and monitoring so that attacks are detected early and properly mitigated.

In some very specific use cases where APIs cannot be properly secured while there is very small number of only internal tenants, there is another additional measure which can be used - hiding API endpoints behind jump host, assuming if tenant use cases allow for this. Then the access to cloud API endpoints for the tenant would be possible only from jump hosts, where each tenant could get a jump host created during the tenant creation.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-12/1.0

Req 13	Access to administrative interfaces, APIs, dashboards and systems must be controlled and protected by higher layer firewalls or proxies (preferably hidden behind jump host), while communication must be encrypted.

The previous requirement about cloud API security also applies to all supporting systems used by administrators of the cloud infrastructure.

Motivation: Cloud API endpoints have big attack surface with relatively low built-in protection against unexpected input and potentially high impact if breached.

Implementation example: It is preferred that all administrative communication is channeled through jump host. If an approach with jump host is not possible, strict VPN or ACLs can be used to limit the access to administrative cloud interfaces, APIs, dashboards etc. for only cloud administrators.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-13/1.0

Req 14	APIs and web UIs offered to tenants and applications must be restricted to necessary minimum.

One can achieve filtering of administrative access and separate it from regular user API requests on WAF / reverse proxy where only certain APIs are allowed.

Motivation: APIs and web UIs (like dashboards) offer both administrative and regular user functionality at the same time. If only regular user functionality is exposed in APIs to tenants, there is no way for them to execute any of administrative functions in case of a breach or privilege escalation.

Implementation example: The WAF rules should be constructed based on the documentation (e.g. https://service/admin/* links can be forbidden for regular tenant access), but in case where documentation and APIs are not simple enough, rules can be generated by using learning mode.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-14/1.0

3.2. Around Cloud Management

Cloud environment requires variety of other systems to be in place for secure operation of the cloud environment. Most of the systems (e.g. logging, monitoring, authentication, ...) are well known and covered in other security documents in details, however, here is a shorter reminder in form of a list of the most important systems.

It is worth repeating baseline principles when it comes to preparation, configuration, installation and operation of cloud infrastructure:

- attack surface of systems and services must be limited
- systems must follow principle of minimal installation
- any sensitive data must be encrypted during transmission and storage (and if possible, in use)
- permissions of users (human and M2M) must follow least privilege principle
- communication partners must be properly authenticated and authorized
- systems must be segregated accordingly
- incoming, outgoing and internal communication must be controlled and filtered
- infrastructure, systems and services must be monitored and regularly tested
- systems must be updated and patched regularly
- incident management process must be in place

Req 15	There must be an internal trusted software repository and it must be used.

The repository must not be accessible from Internet and vendor packages initially downloaded from Internet must be scanned for known malware and vulnerabilities prior to making them available to rest of the cloud infrastructure. Repository must be protected in terms of network reachability (that only necessary systems can access the repository) and authenticity verified with HTTPS certificate, while packages must be signed.

Cloud infrastructure is not allowed to directly use vendor repositories hosted on Internet, and with an internal repository there is no need for any of cloud managements systems to even have reachability to Internet. This specifically applies to certain cloud software which sometimes may require PyPI (Python Package Index) or similar repositories for installation and updates.

Repository needs to be regularly updated and contain recent patches to support the patching procedure.

Motivation: Software packages for the cloud environment come from various sources (e.g. Linux distribution, OpenStack, SDN, GitHub, ...) and can get compromised. To lower the threat, an internal repository which is self-operated must be used for providing software for cloud infrastructure and management systems.

Implementation example: Internally hosted Satellite server for RedHat-based distributions or apt-mirror for Debian/Ubuntu-based distributions.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-15/1.0

Req 16	Integrity of software packages must be verified after transmission and prior to its installation or use.

An adequate algorithm must be used to verify the integrity of software packages after transmission (e.g. after download) and just prior to installation or use (in case no installation is needed, but only execution). In case of packages which came from 3rd party developers, developer-signed packages must be used whenever possible.

Motivation: This reduces risk of a manipulation during transmission or installation.

Implementation example: For Linux packages coming with the Linux distribution, this is generally turned of by default in package managers (DNF and RPM in RedHat-based distributions or apt in Debian/Ubuntu-based distributions).

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-16/1.0

Req 17	There must be a patch management process for the whole cloud environment.

Patch management process in a cloud environment must be implemented in a way that allows patching of all cloud components while keeping given SLA to tenants. If SLA requires so, the patching process must be designed to handle live migration of virtual machines for patching of compute nodes, or to provide cloud management APIs available during patching, so that tenants are able to reschedule workloads which were impacted by the update procedure. With different type of SLA, one might also opt to update whole cloud environment at once and restart all workloads.

All components of the cloud environment must be included in the patch management process, e.g. management, orchestration, compute, storage and networking functions. All supporting systems (monitoring, logging, jump hosts etc.) need to be covered in the procedure as well.

Even in such cases where patching is needed on short notice, the regular testing procedure must be followed.

Motivation: Since new security vulnerabilities emerge daily, it is necessary that patching process is designed so it can be carried out quickly.

Implementation example: Patching procedure should be create so that the patching patching of all compute nodes can be finished within 24-48 hours after critical security issue has been disclosed and patch became available.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-17/1.0

Req 18	Firmware of all devices must be regularly patched. Special attention (more frequent patching) is necessary for devices used by tenants directly (either bare metal servers, PCIe passthrough devices or SR-IOV devices).

Firmware must be treated in the same way as any other software - it must be regularly patched. Often, hypervisor acts as a buffer zone between a tenant space and hardware and it is harder to exploit vulnerabilities in firmware with a hypervisor in between. Using hypervisor cannot be considered as a replacement for firmware patching because relatively recent attacks like SPECTRE and Meltdown required microcode patches and operating system mitigations together to reduce risk of successful attacks. If the bare metal hardware is used (even through SR-IOV), hypervisor offers no additional protection and patching firmware promptly is the only line of defense.

Appropriate procedures for timely firmware patching must be in place, which must account for all workload (live or non-live) migrations.

Motivation: Nowadays, firmware of the devices (BIOS, UEFI, PCIe cards, disk drives, IPMI, ...) is getting more complex than ever, and comes with bugs which can be used for attacks (e.g. to take over the system or steal data from other tenants).

Implementation example: The patching procedure should be created so that it is possible for CVEs with high score to finish patching in roughly 24-48 hours.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-18/1.0

Req 19	Patching of the cloud infrastructure must not be delayed or blocked by tenants (neither technically nor administratively).

It is mandatory that SLA reflects the fact that no tenant can block or prevent patching of the underlaying infrastructure.

The project which wants to get onboarded on cloud even with an idea that they can delay or block patching and other necessary works of the cloud infrastructure must never be onboarded.

Motivation: Patching protects all other workloads and the cloud infrastructure, and it would be extremely unreasonable and dangerous to jeopardize security of whole cloud and all other systems just because of one tenant.

Implementation example: For certain technologies and workloads this can be achieved by live migration, and for those where there is no technical capability, it can be covered by a clause in SLA terms.

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

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-19/1.0

Req 20	Key management system must be available and used for handling secrets.

Secret management plays a crucial role in security of cloud systems. Secrets in this sense include:

- keys
- passwords
- certificates

Here is the list of use case examples for KMS:

- storage encryption
- key management for SSH access
- API communication encryption
- tenant data encryption
- certificates for tenants
- secret management for tenants

The KMS may be delivered as the part of the cloud platform or separately, but it is practically impossible to run secure cloud platform or secure tenant workloads in the cloud without KMS. Therefore, even if the KMS is not delivered as the part of the cloud platform, it is necessary that the KMS is reachable and usable in automated way (like other cloud APIs).

Note: the fulfillment of this requirement is a prerequisite to fulfill many other requirements from this document.

Motivation: KMS allows generation, storage and distribution of secrets in a proper way in cloud environment. If secrets would be stored on systems in the cloud in plaintext, they could be possibly accessed by unauthorized users.

Implementation example: A solution like HashiCorp Vault can be installed as part of the cloud infrastructure.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-20/1.0

Req 21	Appropriate entropy source must be available and used for generation of secrets and cryptographical keys.

Good entropy is the first line of defense for creation of good secrets and cryptographical keys. It is necessary to have a high-quality generator inside the cloud and make it available for tenants so they can use it inside their virtual machines. Use of entropy services provided by 3rd parties is not allowed (e.g. Pollinate from Canonical).

Motivation: Keys and secrets generated with insufficient entropy can be reproduced/guessed by attackers, which allows them to compromise the security of systems in the cloud or even of the whole cloud platform.

Implementation example: Good entropy source can be a custom self-hosted server/service or appropriate RNG offered by CPUs and hardware can be passed through to virtual machines through virtio-rng or similar mechanism (e.g. passing `/dev/urandom` as RNG device).

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-21/1.0

Req 22	Complete cloud infrastructure, including management, orchestration and all supporting systems must have defined metrics on resource allocation and the metrics must be measured and monitored.

It is expected that at least the following metrics are monitored closely:

- system health status
- CPU utilization
- RAM utilization
- network traffic utilization
- disk storage (capacity) usage
- disk traffic utilization
- filesystem utilization

Motivation: Unlike monitoring the load of dedicated systems which is primarily operational concern, utilization monitoring in cloud environment can and must be used for detection and mitigation of attacks.

Implementation example: Any monitoring system (e.g. Nagios or Prometheus+Grafana) can be used if it can monitor the listed metrics.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-22/1.0

Req 23	Complete cloud infrastructure, including management, orchestration and all supporting systems must provide logs of security relevant events.

Logging is mandatory for all systems - compute, storage and management nodes which comprise cloud infrastructure, but also for support systems (repository, KMS, jump hosts, ...). All these systems must generate, save and forward security events like:

- tasks of accounts with high privileges
- user administration tasks
- system health status like CPU, RAM, network and disk usage, and filesystem utilization
- tenant administration (creation, modification, deletion)

- network administration (creation, modification, deletion, including security groups)
- ... (the list is not exhaustive)

Motivation: Appropriate logging of security events is a prerequisite for attack discovery and appropriate mitigation.

Implementation example: Logging configured appropriately (e.g. at least WARN level and in some cases INFO).

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-23/1.0

Req 24	Logs from all cloud infrastructure components must be forwarded to central logging solution. In addition, the logs need to be provided to an appropriate SIEM solution for collection, storage and analysis of security relevant events. SIEM must correlate security events and generate alarms for attacks.

Storing logs is not sufficient, they must be analyzed and properly correlated. A single system may detect an event and log it, but the information might not be sufficient to detect an ongoing attack. This is especially true in cloud environment where a single tenant is distributed among many servers. Anomaly detection should be used in addition to the definition of the rules for log filtering, monitoring and alarming.

Therefore, SIEM (Security Information and Event-Management) solution must be used on top of the central logging solution to analyze and correlate security events, detect anomalies and raise alarms in case of issues. This enables successful implementation of an incident management process.

Due to attack surface of cloud environment, it is necessary that log collection, analysis and correlation, as well as alarming work in real-time or near real-time (with minimal reasonable delay possible) to detect and mitigate ongoing attacks.

Assuming that the SIEM solution is available from responsible SOC (Security Operations Center), the cloud platform is responsible to provide and forward the logs and sufficient information for analysis and alerting, and also to provide sufficient information and help in setting up alerts.

Motivation: Storing logs on servers where they originated is not sufficient since the attacker can then easily manipulate logs and erase the evidence of a successful attack. Therefore, all logs must be forwarded to the central system.

Implementation example: SyslogNG, CheckMK, Nagios, Splunk etc. can be used as a central 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
- Unnoticeable feasible attacks
- Attacks motivated and facilitated by information disclosure or visible security weaknesses

For this requirement the following warranty objectives are relevant:

ID: 3.86-24/1.0

Req 25	The cloud orchestration framework must log (create an audit trail) all security relevant events.

Cloud orchestration framework (in simple terms - internal parts of cloud responsible for managing resources) is used by tenants through APIs, dashboards etc. and is used to manipulate (create, view, modify, delete) tenant resources. Cloud orchestration framework events include all actions which manipulate resources in the cloud.

In addition to regular logging of cloud infrastructure events, cloud infrastructure must provide an audit trail (in practice this is typically done by creation of logs) for all actions performed on all cloud resources.

The security related events coming from cloud orchestration framework need to be forwarded and corelated with the rest of logs and security events.

Motivation: Logging of orchestration events for tenants is a prerequisite for the security of tenants' workloads (e.g. who, when created a specific cloud resource, modified it or deleted it).

Implementation example: Some examples include logging of:

- creation, deletion and changes of tenants (users, projects, roles)
- creation, deletion and changes to tenant networks
- creation, deletion and other virtual machine actions
- creation, deletion, modification and reading of object storage buckets and objects

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-25/1.0

Req 26	Central solution for authentication and authorization of users (both human and M2M) must be used.

All systems of the cloud infrastructure (and supporting systems) must use central system for user administration and, if possible, they must be mapped to one of existing DTAG's IAM solutions.

In an unlikely case of not being able to integrate DTAG's own IAM solution for managing identities, it is necessary to automate account management. Such automation for management of identities must fulfill the "IAM (Identity Access Management) - Framework" (3.69.) requirements 01-57 and 65.

Use of an online/live central system is encouraged, but depending on the availability of the central identity management system (partially due to the reliability of connection), cloud operator might choose another type of central user management so that cloud remains manageable during outage of the central identity management system.

The same approach applies also to tenant accounts for accessing cloud APIs.

Motivation: Central administration of accounts and rights allows easier maintenance compared to approach where accounts and rights are managed on each system individually. The security advantage is that a user account and its rights are known only on a central system, can be transmitted to necessary systems and devices (provisioning), centrally administered (reconciliation) and centrally deleted (deprovisioning). This reduces a risk of forgotten accounts

(e.g. employees which left the company or changed the role).

Implementation example: Examples of such central identity system in DTAG are cIAM, WiW and ZAM and they offer account administration and rights management. 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.

TACACS+ or Radius can be used for certain network device as well.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-26/1.0

Req 27	Cloud infrastructure systems must be hardened in an automated way.

All operating systems components, applications and devices must be hardened to minimize the attack surface. The hardening tasks which cannot be automated needs to be carried manually for each system before the system becomes reachable by other components or is included in production.

Motivation: In a cloud environment where changes happen often, it is hard to maintain good level of security hardening manually, therefore, it is necessary to automate the hardening process. The automated hardening allows consistency, it is less prone to errors and allows even faster changes of the cloud infrastructure.

Implementation example: Some examples of hardening tools which can be used include:

- Ansible
- Chef
- Puppet

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-27/1.0

Req 28	Security compliance check must be implemented in an automated way.

Due to variety of systems which will be hosted in the cloud and complexity of the cloud platform, it is necessary to implement compliance checks for relevant security requirements on all servers and devices.

The implementation must be done as part of CI/CD chain in pre-deployment (if CI/CD and automation are used) and during lifetime on a regular basis. Again, as for hardening, automation must be used where possible to avoid risk of

missing compliance checks and/or systems.

Motivation: Regular compliance checks validate current security status of systems and can identify weaknesses in security configurations or detect unpatched systems with known vulnerabilities.

Implementation example: Some examples of compliance verification tools include:

- CIS Benchmarks
- Ansible
- InSpec
- Orpheus (Telekom internal)
- TASTE-OS (Telekom internal, list of all TOS instances can be found athttps://yam.telekom.de/docs/DOC-685659)

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-28/1.0

Req 29	The cloud infrastructure should contain detection and protection against abuse like ransomware and cryptocurrency mining.

Operation resource monitoring metrics relevant for detection of ransomware and cryptocurrency mining and appropriate thresholds must be defined. The thresholds must be monitored and exceeded thresholds must be evaluated and create a notification in an automated manner.

The detection can be implemented in combination with cooperation of tenants.

Motivation: In addition to regular protection against malware, it is necessary to highlight the need for protection against ransomware which would in a cloud environment have much broader impact because it may impact all the systems running in the cloud at the same time.

Cryptocurrency mining would create additional costs for DTAG by unauthorized use of the equipment.

Implementation example: As a precaution, at least a detection and alarming in form of detection of suspicious activity with high IO should be in place. In addition to that, the cryptocurrency mining detection should be in place, at least by monitoring CPU usage.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-29/1.0

4. Metal

4.1. Bare Metal as a Service

Using bare metal (either as complete servers or specific hardware) has certain benefits for tenants who require special features, greater control over the hardware or greater/stable performance. Even with the expectation that the cloud is built in the best way to enable most features, control and performance towards tenants, there are some use cases which will not fit into virtualized approach. Special care needs to be taken to ensure that security does not get compromised for bare metal cloud offerings because security features offered by virtualization are sometimes taken for granted and can get forgotten.

Req 30	Bare metal nodes must be wiped clean and have hardware-based settings reverted to tenant specific or defaults before first use, after reassignment or before decommissioning.

Before a tenant can use a bare metal node, the machine needs to be entirely wiped (including disk drives, UEFI/BIOS firmware settings, BMC interface, ...). In addition to regular wiping, it is necessary to verify firmware of each piece of hardware (by making checksum or any other method available). In case of frequent tenant changes, this might reduce lifespan of flash memory (UEFI/BIOS/firmware or SSDs). It is worth noting that some modern storage devices (flash-based especially) allow for quick deletion of the content by erasing the encryption key in secure way.

Wiping the bare metal node and resetting settings would be best done after every use, i.e. when machine is about to be released to a generic pool. Final step should be power cycle might be needed to entirely wipe complete RAM content. If a machine cannot be fully wiped, it cannot be reused, and in case of hardware failure (especially relevant for disks), the hardware has to be properly decommissioned (in case of disks - demagnetization and/or destruction) according to standard procedures.

The following list is not complete, but covers components with biggest risk:

- BMC interface (IPMI/Redfish) - firmware/software and settings (like username and passwords/keys)
- UEFI/BIOS - firmware and settings
- Extension cards (NIC, GPU, FPGA, RAID/HBA/generic storage cards) - firmware and in some cases settings
- disk drives (firmware and content)

Depending on the exact use case, additional precautions need to be taken, for example: user must not be allowed to set supervisor UEFI/BIOS password which cannot be reset afterwards because that would make machine unusable for others.

This process ensured removal of all sensitive data. The process also mitigates malware implementation on the bare metal machine if placed elsewhere on the machine. Resetting settings like BMC ensures that machine is not accessible by an attacker afterwards. Resetting UEFI/BIOS settings is beneficial because it ensures that every machine comes with the same (secure) baseline.

Motivation: Data left on the previously used bare metal node might include sensitive data (secrets or personal data) which is not supposed be accessed anymore, especially not by somebody else.

Implementation example: The process of hardware decommissioning can be implemented as a step to be performed on every hardware release operation.

For this requirement the following threats are relevant:

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-30/1.0

4.2. PCI Passthrough

PCI passthrough is a technology which enables a virtual machine to directly access to PCI[e] device or subdevice (through use of SR-IOV, mentioned later). This is done by direct mapping of memory address space of PCI device or subdevice to the virtual machine.

Req 31	Use of PCI passthrough is not allowed unless no other possibilities exist to utilize the hardware.

PCI passthrough also brings native features and performance of given hardware (GPUs, NICs, FPGAs, ...) with extremely low or insignificant overhead to the virtual machine, which makes it very compelling for use cases like machine learning, AI, cloud gaming, network packet processing etc. Applications which utilize PCI passthrough typically have a huge footprint and usage of bare metal machines is favored compared to PCI passthrough.

Motivation: PCI passthrough bypasses hypervisor and security measures of the hypervisor, allowing direct access to the device which may be abused for attacking hypervisor or other tenants.

Implementation example: Use of Open vSwitch instead of dedicated ports on the network card with PCI passthrough.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-31/1.0

Req 32	Hardware used with PCI passthrough must be securely prepared for further use after being released.

As it the case with bare metal hardware decommissioning, the hardware which was used for PCI passthrough need to be wiped and reset.

Motivation: Hardware, especially FPGAs and GPUs still contain leftovers of previous use in working memory (RAM) or in flash memory. These leftovers may contain sensitive data, which must not be available to another tenant. Also, leftovers may contain some insecure settings left by previous tenant of malicious software.

Implementation example: Use of driver supported feature to wipe the content and reset the settings after the device has been freed.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-32/1.0

Req 33	All PCIe passthrough devices (with or without SR-IOV) must have the hardened configuration.

Most common use cases of PCIe passthrough devices and known risks are covered in the requirements in this chapter. Even for those most common use cases, but also more importantly for all other general use cases, it is necessary to apply security hardening of the specific PCIe device in most strict way described by the manufacturer or the community.

Motivation: Insecure configuration of devices may lead to exploitation by tenants in a way that was not intended.

Implementation example: The hardening should be applied according to the manufacturer's description or community guidelines.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-33/1.0

4.3. SR-IOV

SR-IOV stands for Single Root I/O Virtualization and in practical terms it splits a PCI[e] device into subdevices, in particular into physical function (PF, typically accessed from hypervisor) and one or more virtual functions (VFs, typically used by VMs). Using of VF inside a VM is still done by means of PCI passthrough.

Example:
One typical example of SR-IOV usage is for network cards. Once SR-IOV is enabled, a NIC is split into PF which is controlled from hypervisor, while multiple VFs are available for usage in VMs. Hypervisor controls PF, therefore, hypervisor controls link state (physical network interface - or laser if fiber is used), manages mapping of VFs to VLANs (or any other kind of separation is available) and assignment of VFs to VMs. PF and each VF get their own PCI address space. By using PCI passthrough or respective VF, a VM gets access to one or more VFs and each VF is presented as a network card with somewhat limited features. VM cannot change link state through VF and cannot change MAC address or VLAN (or any other encapsulation used) of that specific VF. In combination with configuring appropriate VLANs on the other side (switch or router), it is possible to ensure that VMs stay confined to expected networks. SR-IOV for NICs is typically used when VMs need bare metal like performance, but if multiple VMs utilize the same physical link, laws of physics still apply (one cannot have two VMs each utilizing 10Gbit/s on a single 10 Gbit/s link).

Generic list of security drawbacks for SR-IOV:

- live migration of virtual machines (in case of compute node patching) is more complicated or impossible
- virtual machines hosted on the same hardware can influence each other

The list of security drawbacks for NIC SR-IOV:

- there is no standard host-based traffic filtering (like security groups)
- standard hypervisor-based virtualization of networking cannot be used for traffic separation
- virtual machines hosted on the same hardware can influence each other by PAUSE frame injection
- firmware of the NIC and PF driver have larger attack surface in case of security issues

The security requirements listed here for SR-IOV are applicable also to containers in case where PCIe devices are used directly by containers.

Req 34	When using bare metal networking, or networking through PCI passthrough (with or without SR-IOV) where NIC cannot perform L3/L4 filtering, L3/4 traffic filtering must be implemented on a switch or a router.

Modern NICs which can perform L3/L4 filtering in PCI passthrough (with or without SR-IOV) can be used to perform L3/L4 traffic filtering. However, if there is no traffic filtering possible on the hypervisor or NIC (not counting the tenant defining rules inside the VM or a container), it is necessary to configure tagging (e.g. VLANs) and/or ACLs on a switch or router adjacent to the given bare metal or hypervisor node so that tenant is securely isolated from other tenants.

Motivation: While using SR-IOV allows certain level of filtering or tenant separation on the NIC, it is not always a sufficient replacement for traditional L3/L4 filtering provided by overlay networking (which depends on the NIC capabilities). Traffic filtering per virtual function (per VF) enables tenant separation and protection of tenant according to security groups defined by the tenant.

Implementation example: L3/L4 filter configuration on routers or switches can be achieved by e.g. Ansible plugin in OpenStack Neutron for switch configuration (or any other capable SDN).

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-34/1.0

Req 35	When using PCI passthrough with or without SR-IOV for networking, the traffic separation between tenants must be enforced by NIC.

In addition to previous requirement for traffic filtering on the switch/router side, it is also necessary to apply strict tenant separation on NICs. Modern NICs which support L3/L4 filtering for overlay traffic typically do this properly, but in case of a less capable hardware this can be achieved by use of separate physical ports, VLANs, VXLAN or any similar means which both NIC and the adjacent switch/router support.

Motivation: Without full traffic separation enforced by NIC, a tenant would be able to at least observe (or inject) network traffic from/for other tenants.

Implementation example: Each port for each VM on the hypervisor needs to be configured as a separate VF, with appropriate unique VLAN identifier which is recognized by the switch/router where the overlay network is terminated/routed.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-35/1.0

Req 36	When using SR-IOV NIC, mitigation for PAUSE frame injection must be implemented.

Applications which use SR-IOV typically expect better network performance and reliability compared to usual virtualized networking, and an attack with PAUSE frame injection could become an effective measure for denial of service. Therefore, flow control must be disabled to mitigate the injection of PAUSE frames. Alternatively, one could use a virtualization aware network flow controller.

Motivation: A misbehaving (or malicious) VM can disrupt network availability of other virtual machines on the same host by injection of PAUSE frames, leading to denial of service.

Implementation example: Disabling flow control.

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

For this requirement the following warranty objectives are relevant:
- Availability

ID: 3.86-36/1.0

Req 37	Hosts utilizing PCIe passthrough with SR-IOV for networking must not mix traffic types on the same physical device.

This generally applies also to converged and hyperconverged clouds, so it is expected that generally at least different physical ports are used to separate cloud management from overlay traffic.

Motivation: To prevent exploitation of vulnerabilities in SR-IOV where a tenant may be able to compromise infrastructure in easier way, it is necessary to separate traffic types between physical instances of devices (e.g. most preferred approach would be full network card).

Implementation example: One NIC can carry internal cloud management traffic and the other NIC can carry data plane (overlay traffic), reducing a possibility of attack from tenant space to the infrastructure.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-37/1.0

5. Hypervisor and Servers

5.1. Generic Server Requirements

Req 38	A server (with special care of hypervisors) must be completely wiped before first use, after reassignment or before decommissioning.

To prepare server for the first use, the server must be wiped, firmware needs to be updated and/or verified and operating system installed from scratch. The same approach applies when server changes function (e.g. when it will be used for generic tenant workloads instead of network protection mechanisms). Decommissioning of the system requires full wiping of the data on servers followed by a shutdown to remove all data from volatile memory (RAM).

Motivation: General purpose servers are great in terms of versatility, but versatility may be also abused by an attacker who could use this to extend his presence to other cloud subsystems.

Implementation example: The process of server decommissioning can be implemented as a step to be performed on every hardware release operation, reassign or before first use.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-38/1.0

Req 39	All servers (hypervisor or bare metal used for the cloud platform) must be protected by firewall.

All servers (except bare-metal server given to tenants) are under control of the cloud operator with direct connection to cloud infrastructure network and they need to be protected by local firewalls (like iptables) against possible attacks from other servers. This local firewall needs to be configured so that it allows only necessary communication, including ingress, forwarding and egress traffic rules. All violations need to be logged (and sent to the centralized system for threat detection).

Bare-metal servers given to tenants would be protected and separated from other tenants on the adjacent switch because cloud operator is unable to control local firewall of the tenant.

Motivation: Detection of incoming traffic which is not necessary (and thus not allowed by firewall) can point to already breached device on cloud infrastructure network, while detection of unwanted forwarded or outgoing traffic can point to a breach attempt or a breach of the hypervisor. Early detection increases capability of successful mitigation and can limit the spread of the attack.

Implementation example: Enabling firewall (e.g. iptables) with logging and log forwarding to the central system which is monitoring logs for suspicious activity.

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
- Disruption of availability
- Unnoticeable feasible attacks

For this requirement the following warranty objectives are relevant:

ID: 3.86-39/1.0

Req 40	MAC solution for protection must be implemented on all hosts.

Turning on MAC (Mandatory Access Control) solution, like SELinux or AppArmor, in enforcement mode is not sufficient since they rely on properly configured profiles. Very strict MAC profile (allowing only necessary system actions) must be used for hypervisor software, supporting systems and other critical system services.

All MAC violations need to be logged, raise an alarm, and treated like security events. Even if the attacker manages to breach to hypervisor level with MAC solution and proper profiles, the access to system resources will be somewhat limited and MAC solution will hinder further spread of the attack and increase chances of early detection.

Motivation: MAC solutions offer more strict access model than typically offered by default Linux kernel functions.

Implementation example: Enabling enforcement mode for SELinux and using appropriate profiles for software like KVM and libvirtd.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-40/1.0

Req 41	Integrity protection solution must be implemented on all hosts.

Manipulation of system files (both binaries and configuration files) and data may happen on hosts in case of an attack. The biggest attention must be paid to compute hosts (hypervisors). With integrity protection solutions (e.g. AIDE, Tripwire, OSSEC) in place, integrity of at least binaries and configuration files can be checked, and violations reported. Violations of integrity must be treated as security events.

Some solutions can also check for existence of rootkits and their use is encouraged (not the use rootkits, but solutions which detect rootkits). Checking configuration files is necessary because an attacker could use only configuration changes to spread without being detected.

Motivation: An attacker who gained access to the system may install additional software or change the configuration so that it does not lose access on restart.

Implementation example: A combination of package management utilities and configuration management tools may provide, depending on the overall implementation, sufficient level of integrity checks.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-41/1.0

5.2. Hypervisor separation

Hardware separation was preferred choice for separation in virtualized environments, but it is practically impossible to implement the same approach in a typical cloud deployment. Cloud platforms already offer various protection mechanisms running as a platform services, reducing the need to have implementation of DoS, DDoS, and other protection mechanisms in each application.

Note: Simple firewall for virtual machines (known as security groups in OpenStack) is mandatory, but not in most cases not considered as sufficient to be called network protection mechanism.

Req 42	Software offered as a platform service performing entirely different functions must be placed on different set of hosts or hypervisors.

In addition to general workloads from tenants, cloud platform may offer various services. Two most common examples include network protection mechanisms and management (e.g. secret handling) functions, but this applies to any kind of future offering not fitting under general workloads.

The cloud platform needs to enable tenants to have hosts dedicated for certain use cases, while tenants are necessary (in discussion with their respective security person) to select dedicated hosts for scenarios which deviate from general workloads.

Motivation: Platform services offering network protection mechanisms are at higher risk due to their exposure to raw or roughly filtered outside traffic (Internet traffic or traffic coming from potentially malicious clients). Running protection mechanisms on the same hypervisors as regular tenant workloads would increase the risk of impacting tenant workloads, either through hypervisor breach, side-channel attacks or just by simple resource starvation. Although separation might not eliminate all potential threat vectors, it would drastically reduce the attack surface and make detection and mitigation easier.

Implementation example: Use of dedicated hosts for network protection mechanisms and secret handling functions, respectively.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-42/1.0

Req 43	Platform services offered by the cloud platform must include protection from network attacks in form DoS and DDoS protection, web application firewalls (WAFs), proxies etc.

Private cloud needs to offer tenant-configurable network protection mechanisms. The list is not exhaustive, but it must include at least:

- DoS protection
- DDoS protection
- WAF (Web Application Firewall)
- Proxy
- Firewall

Motivation: In addition to directly used functions, the cloud platform must offer the way for tenants to secure their workloads.

Implementation example: They can be implemented as software running on hypervisors, software on dedicated bare metal servers, as part of shared network hardware (routers, switches, firewalls) or as dedicated device(s).

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-43/1.0

Req 44	Cloud platform services should include at least secret management (or any further management functions with broad reachability).

These services do not have to be integral part of the cloud platform but are in most cases mandatory to enable secure usage of the cloud by tenants. Therefore, the services may be provided by the cloud platform or separately but must be usable from the tenant workloads.

Motivation: In addition to directly used functions, the cloud platform must offer the way for tenants to secure their workloads.

Implementation example: The most typical example in modern applications which requires additional layer of security is secret management, and it is expected that the cloud platform offers at least the following capabilities:

- secret (keys, passwords) creation, usage, management, revocation, deletion
- certificate management
- could include CI/CD runners with capability of managing production environment, depending on the setup of CI/CD chain

One very popular example of secret management tool would be Hashicorp Vault. Example of a certificate management tool may include Certbot utilizing secret store and ACME protocol for interaction between certificate authority and servers using certificates (e.g. web servers).

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-44/1.0

Req 45	Each tenant is obliged to properly use network protection mechanisms and secret management functions according to available platform technologies.

These are two most common deployment scenarios for the network protection services:

- network protection services are provided by the cloud platform and tenant configured network protection in appropriate way for its application, and this is the preferred way of protection
- network protection services are not provided by the cloud platform or they are provided by the cloud platform but tenant cannot use them, therefore tenant should allocate specific set of dedicated hypervisors for running network protection mechanisms, following the assessment of the responsible PSM

In similar way, the tenants in the cloud typically require secret (e.g. keys, passwords, certificates) management and other management and support functions (e.g. configuration management) available on demand to support their workloads. Therefore, there are two most common deployment scenarios:

- secret management are provided by the cloud platform and tenant uses them in appropriate way for its application, which is the preferred way of use (this also applies where management services are available separately - not directly part of the cloud platform but can be used like they are)
- secret management and additional management functions are not provided by the cloud platform or are provided by the cloud platform, but tenant cannot use them, therefore tenant needs to allocate specific set of dedicated hypervisors for running secret management and other necessary services, following the assessment of the responsible PSM

Note: Hypervisor cannot be used for multiple functions at the same time, e.g. network protection functions and secret management cannot run on the same server in such case.

Motivation: The security of applications in the cloud is not given if security functions are not utilized in proper way.

Implementation example: Use of security services offered by the platform to achieve the security of the application.

ID: 3.86-45/1.0

5.3. Hypervisor Protection

Req 46	Management tools allowing direct communication between a VM and a hypervisor must not be used.

There are tools allowing for direct communication between a virtual machine and a hypervisor, like VMWare Tools.

Motivation: Although the tools for direct communication between a virtual machine and a hypervisor sometimes offer certain benefits, they also can be abused for attacks on hypervisor.

Implementation example: Tools inside a virtual machine must not be installed (and used), while all features allowing such communication on the hypervisor must be disabled.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-46/1.0

Req 47	Usage of system resources must be restricted for each virtual machine.

The resources which need to be restricted are:

- vCPU count and/or CPU percentage
- memory (RAM)
- network bandwidth
- network packet rate
- storage space
- storage bandwidth
- storage IOPS

In lack of capabilities for fine grained restrictions of some sort, monitoring and automated mitigation need to be in place.

Motivation: Rogue virtual machines may consume more resources than they are supposed to unless they are restricted. This behavior can lead to resource starvation for other virtual machines running on the same hypervisor, affecting availability of other systems. Worst case, it can completely stall hypervisor and cause unpredicted behavior of hypervisor or compute node completely.

Implementation example: OpenStack in its default configuration, assuming security hardening was applied.

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

For this requirement the following warranty objectives are relevant:
- Availability

ID: 3.86-47/1.0

Req 48	Overbooking of resources must be implemented so that it does not compromise cloud stability and security.

Overbooking of resources needs to be reflected in SLA of the cloud platform, clearly presented from the cloud operator to application owners and it has to be used together with resource restriction (from previous requirement) so that hypervisors do not stall under increased load.

Systems which use high ratio of overbooking need to be closely monitored and should preferably have automated mitigation mechanisms (e.g. live migration).

Motivation: The cloud is typically designed and operated with overbooking of system resources, which may impact stability, availability and security of the hypervisors and workloads running in the cloud. Traditional network functions which are running virtualized in the cloud tend to be less resilient to resource starvation (e.g. because they keep complete session list in RAM and cannot tolerate delays for loading entries), while cloud native applications mitigate such events by scaling out.

Implementation example: If necessary, one may divide cloud into several types based on overbooking, e.g. hypervisors which use overbooking and those which do not use overbooking.

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

For this requirement the following warranty objectives are relevant:

- Availability

ID: 3.86-48/1.0

Req 49	Compute nodes (hypervisors) must expose sufficient entropy to virtual machines.

In addition to existing security requirement in the document describing entropy source for secret generation, it is necessary to highlight that hypervisors need to make proper entropy available to virtual machines, e.g. via virtio-rng device passed to virtual machines.

If a mechanism with transparent access to random number generator from virtual machines cannot be used, tenants need to be aware of the issue and cloud operator must provide access to appropriate entropy source (random number generator).

Motivation: Keys and secrets generated with insufficient entropy can be reproduced/guessed by attackers, which allows them to compromise the security of systems in the cloud or even of the whole cloud platform.

Implementation example: By default in KVM, a virtio-rng device can be passed to virtual machines.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-49/1.0

Req 50	Virtual machine memory encryption must be enabled and used if feasible.

With appropriate hardware which supports AMD Secure Encrypted Virtualization (SEV) or equivalent, and the appropriate software (kernel and libvirt which support SEV) it is possible to encrypt virtual machine (guest machine) memory and protect users against attacks or rogue administrators. This is particularly useful for environments which are not located in highly secure locations. Such memory encryption has impact on live migration, so it is not usable for all workloads.

https://www.zdnet.com/article/the-openstack-train-keeps-chugging-on/

https://arxiv.org/pdf/1901.01759.pdf

Motivation: Memory encryption also significantly decreases possibility of an attack on hypervisor or other tenants from attackers which managed to already compromise one tenant.

Implementation example: Turning on AMD SEV.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-50/1.0

6. Storage

Storage in the private cloud can be realized as software stack running on servers, as dedicated SAN or NAS device(s), or as combination. The focus in this document will be on the most popular solution - installation of storage software on servers. One such solution widely deployed in OpenStack clouds is Ceph.

6.1. Storage Separation

Req 51	Separated storage (logical and/or physical) must be used for different cloud functions.

Logical and/or physical separation of storage is a first step for securing the storage and stored data.

In case of physical separation, it must be ensured that no access from unwanted part of the cloud is possible. For example, if a storage solution is deployed as a set of two clusters, like one for sensitive data and the other for everything else, the hypervisors which needs to run regular workloads must not be able to access storage cluster with sensitive data. This also applies to logical separation, which is considered sufficient if protocols allow for sufficiently strong authentication.

Different cloud functions cannot use the same pool, e.g. cloud management functions are not allowed to use the same pool as regular compute nodes, network protection functions are not allowed to use the same pool as regular workloads etc.

Each separated storage pool must have unique credentials for authentication, but if possible, separate credentials for authentication would be preferred for each volume, bucket and/or object.

Storage offered directly to customers (e.g. object storage), must use the different pool from internal cloud functions (e.g. volumes for compute nodes).

Motivation: Separation of storage can prevent or at least minimize accidental or intentional data leakage.

Implementation example: Implementation of a separate storage pool for each different purpose (object storage, volumes, etc.) and using unique credentials for authentication for each storage pool, and where possible, also separate credentials for authentication would be preferred for each volume, bucket and/or object.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-51/1.0

6.2. Storage Encryption

Req 52	Data on physical hard drives must be encrypted and encryption keys must not be stored on storage nodes.

Storage servers must encrypt the data on all data disks. External storage systems, if used, must also use disk encryption.

If one would store encryption keys on the storage servers (e.g. even on encrypted operating system disks), an attacker would be able to easily retrieve the keys and decrypt the data.

Encryption can be hardware or software based.

(Full operating system disk encryption is not needed if operating system disks contain no sensitive data or secrets. This allows unattended booting of the operating system which should then obtain necessary keys from KMS.)

Motivation: This prevents possibility of data leaks when disks get stolen, misplaced, or returned for warranty, although, proper disk decommissioning procedure (wiping and/or demagnetization) still applies.

Implementation example: Storing the data disk encryption keys in KMS or HSM while using full disk encryption.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-52/1.0

Req 53	All network connections to storage must be encrypted.

All network connections (access) to data, either from hypervisors or object storage proxies for customers, must be encrypted.

Motivation: This prevents attackers on network to obtain data in clear text.

Implementation example: Use of TLS for all connections to storage system.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-53/1.0

Req 54	Storage replication must be encrypted.

Even when using a dedicated network for storage replication, storage replication must be encrypted.

Motivation: Storage replication encryption prevents attackers on the network to obtain data in clear text.

Implementation example: Use of TLS for all internal connections in the storage system (including replication).

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-54/1.0

6.3. Tenant Storage Access

Req 55	Direct use of storage by cloud consumers (tenants) is allowed only through proxy.

Typical direct storage usage is object storage, where cloud consumers directly access data over HTTPS connections to read, write, modify and delete objects.

Proxy can be realized as web application firewall (WAF), it needs to authenticate and authorize clients, and it must

block malicious requests. Connections from client to proxy and from proxy to backend storage system must be encrypted.

Motivation: Storage server included as a part of storage software, e.g. web server in case of object storage, is generally not sufficiently protected and therefore proxy offering additional layer of protection needs to be put between storage software web server and cloud consumer as a client.

Implementation example: Nginx can be used as WAF to authenticate and authorize clients, and block malicious requests.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-55/1.0

Req 56	Storage must not be directly accessible from public networks.

Storage access, even in case of object storage through proxy, is allowed only for cloud consumers (tenants). Therefore, the proxy from the previous requirement must accept only connections coming from this cloud environment (connections from different instances of the same cloud, like data centers and different locations/regions are allowed). Connections may use public IP addresses if necessary, but connectivity needs to be reduced to IP ranges belonging to the cloud or other DTAG's internal systems.

Motivation: Narrowing network access to storage helps to reduce the attack surface.

Implementation example: The proxy can allow access to be only from IP ranges belonging to the cloud or other DTAG's internal systems.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-56/1.0

Req 57	Publicly available object storage must use a separate (external) proxy.

According to all other security requirements, it is necessary to point that external proxy does not always need to use encrypted connections towards clients for non-sensitive data (e.g. CDN delivery of software packages) but must use encrypted connections for sensitive data.

Further details are omitted because external proxy is considered as a separate application and needs to be evaluated depending on intended use case (e.g. such external proxy is supposed to handle application level authentication, authorization etc.)

Motivation: For various use cases (e.g. CDN - Content Delivery Network) where object storage needs to be open to public networks, i.e. accessible from end customers or Internet, additional separate proxy needs to be in place, since

authentication is expected to be different - e.g. on application level.

Implementation example: The preferred way is to run such proxy as an application in the cloud environment.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-57/1.0

7. Overlay Network

Cloud infrastructure network (also known as underlay network) in the cloud environment for connectivity between hypervisors and is used as a transport for overlay network. Cloud infrastructure network also in extended sense includes all necessary communication inside the cloud, like management plane and storage traffic, while it is worth noting that this is not entirely true because it does not transport network from cloud tenants but rather their requests for resource management and their data stored in the cloud. Cloud infrastructure network is typically deployed as a combination of separated networks and VLANs, but this limits number of possible networks usable for tenants.

Overlay network is, unlike cloud infrastructure network, typically deployed as VXLAN or a set of tunnels (GRE most often). This approach allows creation of sufficient number of tenant networks where each tenant can create multiple networks. Control of overlay network is performed by Software Define Network (SDN) controller and most popular SDN software solutions are OpenStack Neutron and Tungsten Fabric (aka Juniper Contrail in commercial version). SND controller in this sense manages virtual networks and tenants get seemingly private networks for their own use, i.e. they do not see the traffic from other tenants unless they expose their virtual machines to external networks (external to the cloud, not the company) like Internet. This construct allows seamless connectivity between virtual machines running on different hypervisors without exposing any details about internal cloud infrastructure network in the cloud.

Req 58	DDoS (and DoS) protection must be used to protect at least Internet and customer facing services hosted in the cloud environment. Cloud APIs need to be protected against DoS and DDoS as well.

Services hosted in the cloud which are open to Internet or customer facing are most vulnerable to various DoS and DDoS attacks. There are 2 solutions available for such protection:

- network based DDoS protection
- inline appliance for DDoS protection

Hardware based inline DDoS protection appliance brings additional benefits, like much faster DDoS attack detection and mitigation, compared to network based DDoS protection. Therefore, use of inline appliance is preferred for critical services facing Internet or customers.

Virtualized DDoS mitigation solution operated on cloud compute nodes are not useful enough and must not be used. Software-based or virtualized solution running on regular servers can be used, however, they need to be inline, i.e. they need to prevent bad DDoS traffic from causing additional load on compute nodes.

DDoS system used for the cloud environment should interoperate with other systems in DTAG's networks to provide optimal level of protection.

Motivation: Proper implementation of DDoS protection helps services running in the cloud to withstand DDoS attacks with minimal impact on their availability, and it also prevents impact on other services running in the cloud due to resource starvation caused by attacked services.

Implementation example: For example, if the cloud infrastructure is connected to AS3320, existing Arbor solution can be used, and appropriate routers must be added to the DDoS monitoring.

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

For this requirement the following warranty objectives are relevant:
- Availability

ID: 3.86-58/1.0

Req 59	Cloud platform must use SDN controller which creates overlay networks for carrying all tenant traffic.

All tenant traffic in the cloud must be carried in overlay network, but the exception are cloud API endpoints which are exposed in a different way (by WAF/proxy).

Motivation: Carrying tenant network traffic in overlay prevent tenants from ever reaching cloud infrastructure network.

Implementation example: The technology used for creation of overlay can be VLAN, VXLAN, MPLS over UDP, GRE tunnels or any other which offers sufficient traffic separation. Also, certain technologies like VLAN can be easy to deploy but they heavily limit number of possible tenant networks.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-59/1.0

7.1. Generic Overlay Network Requirements

Req 60	Tenants must be able to separate their networks from others.

SDN controller needs to allow each tenant to create private networks which are visible and reachable only by the tenant itself. SDN controller also must offer possibility for each tenant to isolate certain part of their networks from other networks of the same tenant.

Motivation: This allows tenants to build n-tier applications.

Implementation example: OpenStack with regular Neutron networking (default example with Open vSwitch) fulfills this requirement.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-60/1.0

Req 61	SDN controller must provide functionality for tenants to define security groups (micro segmentation) for stateful L3/L4 traffic filtering and must enforce the filtering. Tenants must use security groups to protect their virtual machines and services, especially management interfaces and inner tiers.

There is no traditional firewall in front of cloud VMs and the SDN distributes firewall rules described by tenants to compute nodes which perform traffic filtering. Such firewall in OpenStack is named security groups, while on VMWare it is called micro segmentation. Name security groups will be mainly used throughout the rest of the document. Security groups must offer possibility stateful L3/L4 filtering for both ingress and egress traffic, including possibility to specify rules for each virtual machine separately.

Tenants must use security groups for traffic filtering to control ingress and egress traffic in the following way:
- rules need to describe only allowed traffic
- default rule must be to drop traffic, and this must apply to all connections and packets which do not fall into the previously described rules

The default drop action for egress might be omitted for virtual machines which require extremely broad internet access which cannot be easily narrowed to reasonably sized rule set (e.g. outgoing e-mail servers).

Management interfaces of all virtual machines and all interfaces of virtual machines belonging to inner tiers of n-tier application (like management machines and databases) cannot be directly reachable from outside. The exception is possible in certain cases where internal networks of a tenant in the cloud are connected via appropriate VPN to other DTAG's internal networks (e.g. if application partly runs in the cloud and partly on the legacy infrastructure).

Tenant can also deploy a jump host to minimize the attack surface for crucial management functions and provide a central point of control for management access.

Motivation: To allow tenants to more efficiently and better protect their workloads (e.g. VMs), a cloud needs to provide traffic filtering capability.

Implementation example: OpenStack with regular Neutron networking (default example with Open vSwitch) fulfills this requirement.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-61/1.0

8. Cloud Usage

Clouds are typically deployed to enable automation and minimize cost. It is very common step also to use the same cloud environment for deployment of development, test, reference and production systems/applications on top. Mixing development, test, reference and production systems brings an additional risk. With proper application of all security measures in this document and best security practices in general, it is possible to lower the risk to an acceptable level for operation of development, test, reference and production systems in the same cloud.

This portion of the document describes in addition mandatory security requirements which allow mixing of development, test, reference and production cloud infrastructures, as well as mixing of development, test, reference and production systems on the same cloud infrastructure.

Req 62	Cloud environment (infrastructure) must pass appropriate robustness testing to ensure that implemented security measures work as expected.

Prior to using the cloud environment for production systems, cloud infrastructure needs to be tested if security measures are implemented correctly, including testing for robustness, tenant separation (network, storage, processing) and quotas. Workloads must not be able to influence each other, e.g. workload from one tenant must not cause resource starvation of another tenant, up to limit which was defined by overbooking and SLA. During high load (from other tenants and/or attacks), all security aspects of multitenancy must remain functional.

Test results must be documented. Every new major upgrade requires repeated testing, e.g. when changing major version of kernel, hypervisor, storage software, networking stack etc.

Motivation: Only a cloud environment which is sufficiently robust may be used to onboard multiple tenants or to run different types of tenants (development, test, production) together.

Implementation example: Methods which can be used for security testing may vary on use case, but in general they encompass everything from simple automated security scans to more creative approach for the penetration testing.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-62/1.0

8.1. Separation of Cloud Infrastructure Depending on the Purpose

Req 63	Development, test and reference cloud infrastructures themselves must be operated on a fully separate hardware from production cloud infrastructure, including cloud management and orchestration system.

Compute nodes, storage nodes, network devices and other supporting systems which belong to development, test or reference cloud infrastructure must be operated on separate hardware from production cloud environment. This extends also to having a separate DCR (data center router).

Even if security requirements for test and reference systems align with security requirements for production systems, one need to consider operational aspects - testing a new change or feature on shared hardware between test and production cloud environments might bring the whole production cloud environment down.

Motivation: Development, test and reference systems do not always have the same security requirements as production systems and mixing them would introduce unnecessary risk to the production cloud environment.

Implementation example: Separate (disjunct) set of hardware (or software or externally used services) for each of the development, test and production cloud infrastructure platforms.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-63/1.0

Req 64	Development, test and reference cloud infrastructures are not allowed to onboard production workloads.

Production systems may run only on top of production cloud environment. Development, test and reference cloud infrastructures can be used only to develop, test and verify cloud infrastructure.

Onboarding of friendly development, test and reference systems (workloads) on top of development, test and reference cloud infrastructures remains allowed for purpose of comprehensive testing, although cloud users need to be aware of different SLA and security measures.

Motivation: Development, test and reference cloud infrastructure does not always have the same security requirements as production cloud infrastructure, so onboarding workloads on development, test or reference cloud infrastructure would introduce unnecessary risk to the workloads.

Implementation example: Use of development, test and reference cloud infrastructure only for development, test and reference purposes and not onboarding any workloads.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-64/1.0

8.2. Separation of Systems/Applications Running in the Cloud

For mixing development, test, reference and production workloads (systems/applications running in the cloud), it is assumed that production cloud environment is used (sufficiently robust cloud environment).

Req 65	Separate tenants must be used for development, test, reference and production systems (workloads). No tenant space resources may be shared between them.

The requirement to fully separate development, test, reference and production systems is still valid for systems which run in cloud requirement, although, logical separation is used instead of physical separation.

Each different type of workload (development, test, reference, production) must have its own tenant. Such tenants must not share neither overlay networks nor storage (applies to both volumes and object storage). Connecting these different types of systems together is also not allowed, as stated in existing valid security requirements.

In exceptional cases, responsible PSM may allow connectivity or data exchange between test and reference systems, or between reference and live systems. This must be done in strictly controlled way and should have limited scope and duration. Controlled way means that there is:

- adequate firewall or proxy in place in terms of network connectivity
- appropriate system for data manipulation (e.g. anonymization, tokenization, ...) in case of data exchange (storage)

Firewall or proxy needs to ensure protection between different types of systems so that an attacker which made its way to e.g. reference system cannot automatically gain access to production system.

Example of proper data exchange: anonymization system sits in between production and reference systems, and it takes data from production object storage bucket, anonymizes the data and puts the anonymized data to the reference object storage bucket.

Motivation: Separation of environments used for development, test, reference and production systems needs to be applied to all cloud resources respectively.

Implementation example: Example of proper data exchange: anonymization system sits in between production and reference systems, and it takes data from production object storage bucket, anonymizes the data and puts the anonymized data to the reference object storage bucket.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-65/1.0

Req 66	Development, test and reference systems (workloads) must not be directly reachable from Internet or customers.

It is not allowed to open development, test and reference systems broadly to any 3rd parties (customers included) or Internet.

In exceptional cases, temporary, controlled, limited connectivity to 3rd parties and/or Internet may be allowed so that systems can be fully tested for future production (however, appropriate security measures still apply).

Motivation: Even development, test and reference systems tend to have equal security level as production systems, due to their nature they typically have lower security level compared to production systems, and opening them to Internet or customers might be dangerous. They also may work with bogus data.

Implementation example: Restrict reachability of development, test and reference systems only to parties which must have access to develop and test those systems.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-66/1.0

Req 67	Privacy relevant customer data and other sensitive data must not be used on development, test and reference systems.

This requirement belongs to regular list of privacy requirements (following applicable laws and regulations, ID: 911.4-3-7/6.0), however, it is worth repeating most crucial points here.

Development, test and reference systems must not process:
- data of real customers
- privacy relevant data
- internal or business critical data
- other sensitive data

Motivation: Even if development, test and reference systems have lower security level and might get easier compromised, the attacker would not get access to real data.

Implementation example: Using only artificially created, anonymized or pseudonymized data (depending on the data type) for development, test and reference systems.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-67/1.0

8.3. Tenant Specific

Req 68	Tenant data must be encrypted while in transit or at rest.

Depending on the data classification and security level of the cloud infrastructure (and network connecting different cloud environments), it is necessary to:
- encrypt all the network traffic between virtual machines (or containers residing in different virtual machines)
- encrypt data which is stored on block or object storage

Application deployed in the cloud might end up running on a single hypervisor, inside same cloud environment (data center) or distributed between different cloud environments (data centers). Although connectivity for the tenant may be transparent, tenants must be aware that communication between their workloads will go through network cables on public land. Therefore, all tenant traffic which leaves a single unit of workload (virtual machine or a container which talks to other containers which may reside elsewhere), it is necessary to encrypt that traffic. Best practice is to encrypt all the traffic to make sure that application doesn't need to be re-architected in case of changes in the cloud environments or further deployment of application to multiple sites. Encryption inside the data center also offers additional protection against hackers who managed to get into the cloud infrastructure or from rogue cloud administrators, so it

is still beneficial to use it even if cloud environments are connected over encrypted links.

Motivation: Not all cloud environments may be deployed in fully secure locations (e.g. "edge clouds") and despite hard disk encryption offered by the platform, attackers who managed to get access to the cloud infrastructure wouldn't be able to easily extract the data if the data stored on volumes and in object storage buckets is encrypted. Encrypted data in storage buckets with proper use of different keys for different buckets also helps to prevent accidental data leaks, either to the public or to other workloads (e.g. accidental transfer from production to the test system).

Implementation example: Use of secure communications protocols (e.g. HTTPS, TLS and SSH) for communication and using encryption for data at rest - encrypted disks or objects.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-68/1.0

Req 69	Tenant data backup must be in place.

Tenants need to have proper backup procedures in place and backup all necessary data (which includes configuration files, images etc.) to be able to restore the system in case of ransomware infection.

Data replication, which is also common method is this context, is not a replacement for backup because replication will happily replicate faulty bits or bits encrypted by ransomware. Snapshot also cannot be considered as proper backup because:

- snapshots do not ensure integrity of the data (some dirty data could be still in memory and not flushed to the disk, like for DBMS)
- a snapshot keeps data on the same storage device together with source data

Motivation: The requirement is pretty much straightforward for operational reasons, however, the requirement here is specifically mentioned because of security aspect. Ransomware is very profitable way for hackers to make money and it is getting more and more sophisticated.

Implementation example: In terms of cloud, an example of proper backup could include write-only storage bucket where a tenant would be able to store the backups but would not be able to read or modify existing objects. Another tenant (or a user with different role) could have permissions to read the backups and delete them, while modification should be forbidden in general for both.

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

For this requirement the following warranty objectives are relevant:

ID: 3.86-69/1.0

9. 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

10. 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.