**3GPP TSG-SA3 Meeting #124 draft\_S3-253150-r1**

**Wuhan, China, 13 – 17 October 2025**

**Source: BSI (DE)**

**Title: Assets and Threats for SCAS for Container-based Products**

**Document for: Approval**

**Agenda item: 5.2.8**

**Spec: 3GPP TR 33.730**

**Version: 0.1.0**

**Work Item: FS\_SCAS\_CP**

**Comments**

This contribution proposes to add content to the Assets and Threats section for TR 33.730 „Study on Security Assurance Specification (SCAS) for Container-based Products“.

\* \* \* First Change \* \* \* \*

# 5 Assets and threats for Container-based Products

## 5.1 Introduction

The present clause contains assets and threats that are believed to apply to more than one container-based network product (GCNP).

The format follows TR 33.926 [2] and TR 33.927 [3] to allow alignment with existing SCAS threat catalogues, with adaptations for containerized deployments.

Container-based network products may consist of multiple container images orchestrated by a container orchestration platform (e.g. Kubernetes), either included in the product boundary or assumed to have undergone equivalent evaluation. The threats below cover both intra-GCNP interactions and interfaces between the GCNP and external entities (e.g. OAM systems, service-based interfaces).

## 5.2 Critical Assets

### 5.2.1 Mapping of existing Critical Assets from GNP

Mapping of critical assets of GNP (see TR 33.926 [2], clause 5.2) to GCNP.

|  |  |
| --- | --- |
| **Critical Asset for GNP** | **Applicablity for GCNP** |
| User account data and credentials (e.g. passwords) | applicable for GCNP |
| Log data | applicable for GCNP |
| Configuration data, e.g. GNP's IP address, ports, VPN ID, Management Objects (e.g. user group, command group) etc. | applicable for GCNP with adaptations |
| Operating System (OS), i.e. the files that make up the OS and its processes (code and data) | applicable for GCNP with adaptations |
| GNP Application | applicable for GCNP with adaptations |
| Sufficient processing capacity: that processing powers are not consumed close to limits | not applicable |
| Hardware, e.g. mainframe, board, power supply unit etc. | not applicable |
| Console interface, for local access | applicable for GCNP |
| OAM interface, for remote access | applicable for GCNP |
| GNP Software: binary code or executable code | applicable for GCNP |

### 5.2.2 Mapping of existing Critical Assets from GVNP

Mapping of critical assets of GVNP (see TR 33.927 [3], clause 5.2.1) to GCNP.

|  |  |
| --- | --- |
| **Critical Asset for GVNP** | **Applicablity for GCNP** |
| User account data and credentials (e.g. passwords, private key) | applicable for GCNP |
| Log data | applicable for GCNP |
| Configuration data, e.g. GVNP's IP address, ports, VPN ID, Management Objects (e.g. user group, command group) etc. | applicable for GCNP with adaptations |
| Guest Operating System, i.e. the files that make up the guest OS and its processes (code and data) | applicable for GCNP with adaptations |
| GVNP Application | applicable for GCNP with adaptations |
| Sufficient processing capacity: that processing powers are not consumed close to limits | not applicable |
| OAM interface, for remote access: interface between GVNP and OAM system | applicable for GCNP with adaptations |
| Interface between virtualised network function (VNF) and VNFM | applicable for GCNP with adaptations |
| Interface between VNF and virtualisation layer, for providing the execution environment to run VNF | applicable for GCNP with adaptations |
| GVNP Software package (binary code or executable code) which includes: - VNFD;- VNF image and image description file;- Configuration data (e.g. manifest file as defined in [5]) | applicable for GCNP with adaptations |

### 5.2.3 Critical Assets for GCNP

List of new, copied and derived critical assets for GCNP.

|  |  |
| --- | --- |
| **Critical Asset for GCNP** | **Origin** |
| User account data and credentials (e.g. passwords, private key, API tokens, Kubernetes service account tokens) | copied from GNP and GVNP |
| Log data (container logs, orchestrator audit logs, security event logs) | copied from GNP and GVNP |
| Configuration data - including CNF’s network configuration (IP addresses, ports, VPN IDs), orchestration manifests, Helm charts, Kubernetes RBAC policies, and CNI network policies | copied from GNP and GVNP |
| Container images - including base images, application layers, manifests, and associated image signatures | derived from GNP and GVNPThe GCNP shares the hosts kernel and a base image is provided containing the minimal userland from another OS. |
| Guest operating system layers inside containers - including files and processes of the container image OS layer | derived from GNP and GVNP |
| Container orchestration configuration - e.g. Deployment/StatefulSet specs, PodSecurity settings, NetworkPolicies | new for GNP |
| GCNP Application - the software components implementing 3GPP-defined NF functionality | derived from GNP and GVNP |
| Sufficient processing capacity: that processing powers are not consumed close to limits | derived from GNP and GVNP |
| Sufficient storage capacity: limited or exhausted storage capacity should not hinder the functionality | new for GNP |
| Service interfaces defined in relevant 3GPP specifications | copied from GNP and NF-specific sections |
| Service interfaces not defined by 3GPP but exposed by the CNF, container orchestration API | new for GCNP |
| OAM interface, for remote access: interface between GCNP and OAM system | derived from GNP and GVNP |
| Interface between GCNP workloads and the orchestration control plane (e.g. Kubernetes API) - In the container SCAS context, the VNFM role is effectively handled by the Container Infrastructure Service Management (CISM), | new for GCNP |
| Interface between GCNP workloads and containerization layer, for providing the execution environment to run CNF; CNI - Execution environment interface between container runtime and orchestration platform | new for GCNP |

## 5.3 Threats

### 5.3.1 Generic threats format

Threats are described using the following format:

- Threat Name:

- Threat Category:

- Threat Description:

- Threatened Asset:

### 5.3.2 Generic threats for GCNP

#### 5.3.2.1 Introduction

#### The common STRIDE threat categories used in TR 33.926 [2], clause 5.3.1 also apply to GCNP. Many generic threats from TR 33.926 clause 5.3 are applicable with adaptation for container contexts. In addition, GCNP have unique threats due to container runtime, orchestration APIs, and image distribution.5.3.2.2 Threats related to 3GPP-defined interfaces

GCNP inherit all the threats related to 3GPP-defined interfaces in TR 33.926 [2], clause 5.3.2, without any changes. It means that there is no need repeat the threats relating to 3GPP-defined interfaces which are covered in 3GPP security specifications. If threats relating to 3GPP-defined interfaces are found to be not sufficiently covered in existing 3GPP security specifications, they need to be addressed in the SCAS for containerized network products.

As in TR 33.927. clause 5.3.2.2, threats for 3GPP-defined interfaces are as per TR 33.926, clause 5.3.2 unless GCNP-specific considerations arise (e.g. exposure of SBA endpoints through orchestration misconfiguration). If existing protections are absent or misconfigured, these interfaces remain in scope for container SCAS.

#### 5.3.2.3 Threats related to interfaces introduced in container environments

Two interfaces unique to GCNP are identified as critical assets:

- Interface between GCNP workloads and the orchestration control plane (e.g. Kubernetes API).

- Interface between GCNP workloads and the container runtime API (e.g. Docker socket, containerd API).

If unprotected, these interfaces can be exploited for privilege escalation, container escape, or manipulation of other workloads.

#### 5.3.2.4 Spoofing identity

##### 5.3.2.4.1 Default Accounts

The threat in clause 5.3.3.1 of TR 33.926 [2] applies to GCNP.

The difference is that VNF is accessed through VNC (Virtual Network Console) rather than through the physical console interface, an attacker can use a default account to access a CNF via VNC.

Default accounts can be present in container images.

##### 5.3.2.4.2 Weak Password Policies

The threat in clause 5.3.3.2 of TR 33.926 [2] applies to GCNP.

However, the attacker using the weak password accesses GCNP through VNC (Virtual Network Console) rather than through the physical console interface.

##### 5.3.2.4.3 Password peek

The threat in clause 5.3.3.3 of TR 33.926 applies to GCNP.

However, the attacker using the peeked password accesses GCNP through VNC (Virtual Network Console) rather than through the physical console interface.

##### 5.3.2.4.4 Direct Root Access

The threat in clause 5.3.3.4 of TR 33.926 [2] applies to GCNP.

##### 5.3.2.4.5 IP Spoofing

The threat in clause 5.3.3.5 of TR 33.926 [2] applies to GCNP.

However, the objective of unauthorized access is a VNF, not a computer.

##### 5.3.2.4.6 Malware

The threat in clause 5.3.3.6 of TR 33.926 [2] applies to GCNP.

##### 5.3.2.4.7 Eavesdropping

The threat in clause 5.3.3.7 of TR 33.926 [2] applies to GCNP.

##### 5.3.2.4.8 Service Account Token Abuse

*- Threat Name*: Service Account Token Abuse

*- Threat Category*: Spoofing identity

- *Threat Description*: An attacker could steal a Kubernetes service account token from a pod and use it to impersonate the GCNP, resulting in the attacker being able to interact with the container API, enumerate resources, privilege escalation, lateral movement, data exfiltration and abuse of resources resulting in denial of service.

*- Threatened Asset*: Kubernetes API credentials

##### 5.3.2.4.9 API Endpoint Impersonation

An attacker could spoof an orchestration API or SBA endpoint to mislead GCNP components.

#### 5.3.2.5 Tampering

##### 5.3.2.5.1 Software Tampering

The threat in clause 5.3.4.1 of TR 33.926 [2] applies to GCNP.

Different from traditional physical network products, the entire GCNP is instantiated from the container image(s) and other information (e.g. configuration data, software environmental parameters, license terms information, script, manifest file, checksum, etc.).

*- Threat Name*: Software Tampering

*- Threat Category*: Tampering

- *Threat Description*: Compared with GNP software, GCNP software has additional attack surfaces, e.g. in the process of CNF package onboarding, during which the software package of a GCNP can be tampered/altered if not protected. An attacker, for example, can inject malicious code or tamper the information inside the unprotected package during on boarding. Then after the instantiation of the GCNP, the tampered code can be executed to conduct several attacks (e.g. DoS, Information Stealing, Frauds and so on).

*- Threatened Asset*: all critical assets of GCNP as listed in clause 5.2.1.

##### 5.3.2.5.2 Ownership File Misuse

The threat in clause 5.3.4.2 of TR 33.926 [2] applies to GCNP.

##### 5.3.2.5.3 Boot tampering

This threat is not applicable for GCNP since GCNP do not have a boot process in the traditional sense.

##### 5.3.2.5.4 Log Tampering

The threat in clause 5.3.4.4 of TR 33.926 [2] applies to GCNP.

##### 5.3.2.5.5 OAM traffic Tampering

The threat in clause 5.3.4.5 of TR 33.926 [2] applies to GCNP.

##### 5.3.2.5.6 File Write Permissions Abuse

The threat in clause 5.3.4.6 of TR 33.926 [2] applies to GCNP.

##### 5.3.2.5.7 User Session Tampering

The threat in clause 5.3.4.7 of TR 33.926 [2] applies to GCNP.

##### 5.3.2.5.8 Exposed Containerization API

*- Threat Name*: Exposed Containerization API

*- Threat Category*: Tampering

- *Threat Description*: An attacker who gains access to this API can exploit it to escalate their privileges within the system, potentially gaining unauthorized access to sensitive container configurations, network settings, and runtime data. This elevated access allows them to manipulate container security contexts, modify resource allocations, and potentially compromise both the containerized applications and the underlying host system's security boundaries.

*- Threatened Asset*: orchestrator and runtime APIs

##### 5.3.2.5.9 Image Registry Tampering

*- Threat Name*: Image Registry Tampering

*- Threat Category*: Tampering

- *Threat Description*: An attacker who gains unauthorized access to a container image registry can insert malicious layers or replace trusted images with backdoored versions. This allows the attacker to embed malware, backdoors, or exploit code within images that are later pulled and run by production environments. When a compromised image is deployed, the attacker can gain initial access to target systems, escalate privileges, or persist undetected within the cluster. The threat is particularly severe if production systems automatically pull images from registries without rigorous validation or scanning, potentially enabling widespread compromise across multiple services or environments.

*- Threatened Asset*: container image integrity

##### 5.3.2.5.10 Pod Spec/Manifest Modification

*- Threat Name*: Pod Spec/Manifest Modification

*- Threat Category*: Tampering

*- Threat Description*: An attacker who alters deployment manifests or pod specifications can modify pod configurations to add elevated capabilities, host mounts, or enable privileged mode. This manipulation enables the attacker to bypass container isolation, gain root-level access on the host, and access sensitive files or resources outside the container. By exploiting these changes, the attacker can escalate privileges, compromise cluster security, persist undetected, and move laterally within the Kubernetes environment. Such unauthorized modifications increase the risk of data theft, operational disruption, and full cluster compromise, especially if security controls such as least privilege or Pod Security Standards are not enforced.

*- Threatened Asset*: deployment/manifest configurations

##### 5.3.2.5.11 File Tampering inside Containers

*- Threat Name*: File Tampering inside Containers

*- Threat Category*: Tampering

- *Threat Description*: An attacker who gains access to a container with writable filesystem layers can modify container files if read-only enforcement is not applied. Such tampering allows insertion or alteration of binaries, scripts, or configuration files within the container environment. This can lead to persistence of malicious code, privilege escalation, disruption of application behaviour, data theft, or lateral movement within the cluster. The risk increases significantly when containers are configured without strict immutability policies or security contexts that enforce read-only root filesystems. File tampering may also undermine the integrity and trustworthiness of container images and deployed workloads, potentially causing widespread impact across the environment.

*- Threatened Asset*: in-container filesystem integrity

#### 5.3.2.6 Repudiation

##### 5.3.2.6.1 Lack of User Activity Trace

The threat in clause 5.3.5.1 of TR 33.926 [2] applies to GCNP.

##### 5.3.2.6.2 Lack of Container-Level Audit Logging

*- Threat Name*: Lack of Container-Level Audit Logging

*- Threat Category*: Repudiation

- *Threat Description*: Absence of comprehensive audit logs for container-related events - such as container start/stop, image pulls, and capability assignments - creates a blind spot in monitoring and security. Without these logs, it becomes difficult or impossible to track user actions, detect unauthorized changes, or investigate suspicious activity within the container environment. This lack of traceability undermines accountability, making it easier for attackers or malicious insiders to repudiate their actions and evade detection or forensic analysis. The absence of container-level audit logging also hinders compliance with regulatory requirements and weakens the overall security posture by masking operational anomalies and potential attacks.

*- Threatened Asset*: container event traceability

##### 5.3.2.6.3 Orchestrator Audit Logs Disabled

*- Threat Name*: Orchestrator Audit Logs Disabled

*- Threat Category*: Repudiation

- *Threat Description*: When Kubernetes orchestrator audit logs are disabled or not properly configured, it becomes impossible to prove or track actions taken via kubectl commands or API requests. This lack of audit trail severely undermines accountability and traceability within the cluster, enabling attackers or malicious insiders to perform unauthorized activities without leaving evidence. Without these logs, organizations lose critical visibility into who accessed or modified cluster resources, hindering detection of malicious behaviour, incident investigation, forensic analysis, and compliance with security policies or regulatory requirements. This gap increases the risk of undetected privilege escalation, unauthorized configuration changes, data tampering, or service disruptions, ultimately weakening the security posture and (Global Container Network Platform) can lead to unauthorized visibility and access to network communications between containers, pods, or services. Without proper isolation mechanisms - such as Kubernetes Network Policies, namespace segmentation, or service mesh controls - traffic can flow freely across workloads that should be isolated. This exposes sensitive data in transit, increases the risk of eavesdropping, data leakage, and lateral movement by malicious actors who compromise one component of the cluster. Attackers may intercept unencrypted or unauthorized traffic, gain insights into internal service architectures, and exploit this information to escalate attacks or exfiltrate confidential information. Effective traffic isolation is critical to maintaining confidentiality and limiting the blast radius of breaches especially in multi-tenant or complex microservices environments.

*- Threatened Asset*: inter-pod/network traffic confidentiality

##### 5.3.2.7.16 Secrets in Environment Variables

 *- Threat name*: Secrets in Environment Variables.

*- Threat Category*: Information Disclosure.

*- Threat Description*: Storing secrets such as credentials or tokens in environment variables exposes them to significant security risks. These secrets are easily accessible by anyone with access to the container or node since environment variables can be inspected inside the container, appear in pod specs, and may be exposed in logs or debugging output. This exposure increases the chance of credential leakage, unauthorized access, and lateral movement within the cluster. Additionally, environment variables typically lack encryption at rest and in transit, have poor auditability, and are difficult to rotate once compromised, further exacerbating the risk. Attackers who access these environment variables can use the exposed secrets to gain unauthorized access to sensitive systems or data.

*- Threatened Asset*: container runtime secrets

##### 5.3.2.7.17 Secrets in Image Layers

 *- Threat name*: Secrets in Image Layers

*- Threat Category*: Information Disclosure.

*- Threat Description*: Embedding secrets, such as private keys or credentials, within container image layers exposes them to anyone who can pull or inspect the image. Even if later removed in newer layers, these secrets remain retrievable from image history. Attackers gaining access to these secrets can authenticate to sensitive systems, bypass security controls, and potentially compromise the wider environment. This risk is heightened when images are stored in public or unsecured registries without proper scanning or scrubbing.

*- Threatened Asset*: embedded image secrets

#### 5.3.2.8 Denial of Service

The threats in all clauses of clause 5.3.7 for TR 33.926 [2] apply to GCNP.

In addition, the following threats apply to GCNP.

##### 5.3.2.8.1 Resource Starvation via Orchestration

 *- Threat name*: Resource Starvation via Orchestration

*- Threat Category*: Denial of Service.

*- Threat Description*: An attacker who orchestrates pods with excessive CPU and memory requests can deliberately exhaust cluster resources, causing denial of service across workloads. By scheduling malicious pods that consume disproportionate compute or memory resources without proper limits, the attacker starves legitimate applications of critical resources, leading to degraded performance, application crashes, or total service unavailability. This threat is amplified in environments lacking resource quotas, limits, or proper orchestration policies, and can also drive up cloud costs through unnecessary autoscaling. Such attacks impact cluster stability, availability, and reliability, making resource management and enforcement crucial to mitigating risk.

*- Threatened Asset*: cluster resource availability

##### 5.3.2.8.2 Container Spawn Storm

 *- Threat name*: Container Spawn Storm

*- Threat Category*: Denial of Service.

*- Threat Description*: An attacker who abuses the ability to create large numbers of pods or containers can overwhelm cluster resources, causing performance degradation, service disruption, and denial of service. By rapidly spawning excessive pods without proper controls or limits, the attacker exhausts CPU, memory, network, and orchestration resources, destabilizing the Kubernetes environment. This attack may also increase cloud infrastructure costs due to uncontrolled scaling. The threat is particularly severe in clusters lacking effective resource quotas, rate limiting, or admission controls, enabling the attacker to degrade availability or cause outages across multiple applications and services.

*- Threatened Asset*: cluster orchestration capacity

##### 5.3.2.8.3 DoS via Log Volume

 *- Threat name*: DoS via Log Volume

*- Threat Category*: Denial of Service.

*- Threat Description*: An attacker generates excessive container logs to fill storage resources, causing denial of service by exhausting disk space or overwhelming log processing systems. This attack can disrupt cluster operations, block legitimate logging and monitoring, and hinder incident detection and response. Without controls like log rate limiting, retention policies, or alerting on unusual log volumes, excessive logging can degrade cluster performance, cause service outages, and increase operational costs. This threat is especially impactful in busy Kubernetes environments where logs are critical for security and operational visibility.

*- Threatened Asset*: storage and logging subsystems

#### 5.3.2.9 Elevation of privilege

All threats in clause 5.3.8 for TR 33.926 [2] apply to GCNP.

In addition, the following threats apply to GCNP:

##### 5.3.2.9.1 Abuse of Linux Capabilities

 *- Threat name*: Abuse of Linux Capabilities

*- Threat Category*: Elevation of privilege

*- Threat Description*: An attacker who exploits excessive or unnecessary Linux capabilities (e.g. CAP\_SYS\_ADMIN) granted to a container can escalate privileges beyond the intended scope. Linux capabilities break down root privileges into fine-grained permissions, and when improperly assigned or not dropped, they enable a compromised container process to perform privileged actions such as modifying system configurations, accessing sensitive kernel interfaces, or escaping container isolation. This abuse can lead to full host compromise, lateral movement within the cluster, or persistent control over the Kubernetes environment. The risk increases when containers run with default or elevated capabilities without careful restriction, lacking security context settings like dropping all unused capabilities or disabling privilege escalation mechanisms. Properly restricting Linux capabilities and using Kubernetes securityContext controls (e.g., allowPrivilegeEscalation: false) is critical to mitigating this threat.

*- Threatened Asset*: host and container privilege boundaries

##### 5.3.2.9.2 Privilege Escalation via Orchestration Misconfiguration

 *- Threat name*: Privilege Escalation via Orchestration Misconfiguration

*- Threat Category*: Elevation of privilege

*- Threat Description*: An attacker who exploits RBAC misconfiguration in a Kubernetes cluster can create pods with elevated privileges by assigning themselves roles or permissions beyond their intended scope. Misconfigured role-based access control (RBAC) settings may allow an attacker to create or modify roles and role bindings that grant them the ability to launch pods with privileged settings, such as adding capabilities, mounting host filesystems, or running in privileged mode. This can lead to container breakout, host compromise, lateral movement within the cluster, and full cluster takeover. The risk is particularly high when the attacker is allowed the escalate permission on roles or clusterroles, enabling them to escalate privileges beyond their assigned limitations.

*- Threatened Asset*: RBAC and orchestration policies

##### 5.3.2.9.3 Running as Root inside Containers

 *- Threat name*: Running as Root inside Containers

*- Threat Category*: Elevation of privilege

*- Threat Description*: When containers run with root user privileges by default, attackers who compromise such containers gain powerful capabilities that facilitate exploitation of container breakout vulnerabilities. Root execution inside containers enables attackers to perform privileged operations, bypass container isolation, manipulate kernel interfaces, and potentially escape to the host system. This gives them the ability to gain full root access on the underlying host, escalate privileges within the cluster, and control critical resources. Running containers as root increases the risk surface for attacks leveraging known and unknown kernel or runtime vulnerabilities, allowing attackers to execute arbitrary code with minimal restrictions and achieve persistent control over the Kubernetes environment.

*- Threatened Asset*: container isolation enforcement

##### 5.3.2.9.4 Use of Privileged Containers

 *- Threat name*: Use of Privileged Containers

*- Threat Category*: Elevation of privilege

*- Threat Description*: Allowing containers to run in privileged mode grants them nearly unrestricted access to the host system, effectively bypassing key security mechanisms and container isolation. This elevated access enables an attacker who compromises such a container to interact directly with the host kernel, modify system files, and access sensitive data on the host and other workloads. Privileged containers can facilitate container escape, lateral movement, and full host takeover, significantly expanding the attacker’s capabilities. Running containers as privileged violates the principle of least privilege and greatly increases the risk of privilege escalation, cluster compromise, and persistence of malicious activity.

*- Threatened Asset*: host and cluster security controls

#### 5.3.2.10 Generic assets and threats for network functions supporting SBA interfaces

The assets and threats for containerized network functions supporting SBA interface are the same as the assets and threats specified in clause 6 for TR 33.926 [2].

\* \* \* End of Changes \* \* \* \*