ETSI GS NFV-SEC 023 V5.4.1 (2026-04)

GROUP SPECIFICATION

Network Functions Virtualisation (NFV) Release 5; Security; Container Security Specification

Disclaimer

The present document has been produced and approved by the Network Functions Virtualisation (NFV) ETSI Industry Specification Group (ISG) and represents the views of those members who participated in this ISG.
It does not necessarily represent the views of the entire ETSI membership.

Reference
DGS/NFV-SEC023

Keywords
container, cyber security, NFV, security

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI IPR online database.

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™**, **LTE™** and **5G™** logo are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Group Specification (GS) has been produced by ETSI Industry Specification Group (ISG) Network Functions Virtualisation (NFV).

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document specifies the security and hardening requirements for running NFV software (e.g. VNFs) in containerized environments. The present document provides a threat analysis for a container-based NFV deployment and a state of the art on container security. The present document proposes a set of requirements and solutions for attack surface reduction and privilege limitation and analyses existing solutions for resource limitations (cgroups); Hardware protections (HMEE); Container hardening (including patching); Containers in VMs and containers on bare metal.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found in the ETSI docbox.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents are necessary for the application of the present document.

[1] ETSI GS NFV-IFA 040: "Network Functions Virtualisation (NFV) Release 5; Management and Orchestration; Requirements for service interfaces and object model for OS container management and orchestration specification".

[2] ETSI GS NFV-SEC 021: "Network Functions Virtualisation (NFV) Release 4; Security; VNF Package Security Specification".

[3] ETSI GS NFV-SOL 004: "Network Functions Virtualisation (NFV) Release 5; Protocols and Data Models; VNF Package and PNFD Archive specification".

[4] ETSI GS NFV-IFA 011: "Network Functions Virtualisation (NFV) Release 5; Management and Orchestration; VNF Descriptor and Packaging Specification".

[5] ETSI GS NFV-SEC 012: "Network Functions Virtualisation (NFV) Release 5; Security; System architecture specification for execution of sensitive NFV components".

[6] Common Vulnerability Scoring System v4.0: Specification Document.

[7] ETSI GS NFV-SEC 026: "Network Functions Virtualisation (NFV) Release 5; Security; Isolation and trust domain specification".

[8] ETSI GS NFV-SOL 013: "Network Functions Virtualisation (NFV) Release 5; Protocols and Data Models; Specification of common aspects for RESTful NFV MANO APIs".

[9] ETSI GS NFV-SEC 022: "Network Functions Virtualisation (NFV) Release 4; Security; Access Token Specification for API Access".

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long-term validity.

The following referenced documents may be useful in implementing an ETSI deliverable or add to the reader's understanding, but are not required for conformance to the present document.

[i.1] ETSI GR NFV 003: "Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV".

[i.2] ETSI GR NFV-IFA 029: "Network Functions Virtualisation (NFV) Release 3; Architecture; Report on the Enhancements of the NFV architecture towards "Cloud-native" and "PaaS"".

[i.3] OCI™ Image Format Specification v1.0.1.

[i.4] ETSI GS NFV-SOL 018: "Network Functions Virtualisation (NFV) Release 5; Protocols and Data Models; Profiling specification of protocol and data model solutions for OS Container management and orchestration".

[i.5] ETSI GS NFV-IFA 036: "Network Functions Virtualisation (NFV) Release 5; Management and Orchestration; Requirements for service interfaces and object model for container cluster management and orchestration specification".

[i.6] MITRE ATT&CK®: "Containers Matrix".

[i.7] Microsoft®: "Threat Matrix for Kubernetes".

[i.8] ETSI GS NFV-SEC 014: "Network Functions Virtualisation (NFV) Release 3; NFV Security; Security Specification for MANO Components and Reference points".

[i.9] ETSI GS NFV-SEC 006: "Network Functions Virtualisation (NFV); Security Guide; Report on Security Aspects and Regulatory Concerns".

[i.10] NIST Special Publication 800-190: "Application Container Security Guide".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in ETSI GR NFV 003 [i.1] apply.

3.2 Symbols

For the purposes of the present document, the symbols given in ETSI GR NFV 003 [i.1] apply.

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in ETSI GR NFV 003 [i.1] and the following apply:

ASLR	Address Space Layout Randomization
AVS	Attestation Verifier Service
BPF	Berkeley Packet Filter
CI/CD	Continuous Integration / Continuous Delivery or Deployment
DAC	Discretionary Access Control
DMZ	Demilitarized Zone
DoS	Denial of Service
DRTM	Dynamic Root of Trust Measurement
eBPF	Extended Berkley Packet Filter
eDRTM	External Dynamic Root of Trust Measurement
HMEE	Hardware Mediated Execution Enclave
iDRTM	Internetal Dynamic Root of Trust Measurement
MAC	Mandatory Access Control
MCS	Multi-Category Security

MFA	Multi-factor Authentication
OTP	One Time PIN or Passcode
RAVS	Remote Attestation Verifier Service
RTR	Root of Trust for Reporting
RTS	Root of Trust for Storage
SDLC	Software Development Life Cycle
SLO	Service Level Objective

4 NFV container deployments

4.1 Introduction

A VNF or VNFC designed to be deployed and managed on Container Infrastructure Service instances (CIS) is represented by a containerized workload running on OS containers. The management and orchestration of OS containers require the services of the following two functions:

- Container Infrastructure Service Management (CISM) ; and
- Container Image Registry (CIR).

Requirements on the list of services to be offered by architectural elements providing the CISM and CIR functions and requirements on the interfaces to expose these services to various consumers are specified in ETSI GS NFV-IFA 040 [1].

The study ETSI GR NFV-IFA 029 [i.2] identified several possible container deployment scenarios. These include:

Nested Container Infrastructure Services in VMs, Container Infrastructure Services on bare-metal, or NFVI providing Container Infrastructure Services on bare-metal and in VMs. Additionally, different mapping options of the CISM function to NFV-MANO are presented. The characteristics, particularly those important for security aspects, of these deployment configurations are described in the following sub-clauses.

4.2 Assumptions

4.2.1 Introduction

The assumptions in the present clause are general and applicable to any deployment option or scenario in the subsequent clauses. They relate to:

a) modelling the containerized workloads; and

b) general assumptions on the CIS infrastructure.

4.2.2 Assumptions on modelling the containerized workloads

Existing de-facto standard container infrastructure management systems consider managing containerized workloads at a granularity which is higher than on a single container level. A set of OS containers that are tightly coupled to run in the same execution environment (i.e. on the same CIS instance) and designed to be scheduled together, is defined as the smallest manageable workload object. A VNFC instance is assumed to be 1:1 mapped to such the smallest manageable unit of a containerized workload.

The CISM manages and exposes Managed Container Infrastructure Objects (MCIOs). An MCIO whose declarative descriptor specifies compute/storage infrastructure resource requests represents such smallest manageable unit and is assumed to be 1:1 mapped to a VNFC.

The MCIOs are assumed to be specified agnostic to the possible container infrastructure deployment scenario.

As described in ETSI GS NFV-IFA 040 [1], clause 5.2.5, the OS container image is an abstract NFV object for OS container management and orchestration, representing a software image for an OS container. OS container images are referenced in the declarative descriptors of MCIOs. The MCIOs, which are built from OS container images, are created by deploying their OS containers from their respective images. The OS container images are typically pulled by the CIS instances from the CIR based on the image names and CIR URLs. Dependent on the delivery method of the VNF software, the VNF Package may include the OS container images as artifacts (e.g. as archive files). OS container images are referenced in the declarative descriptors of the MCIOs.

An OS container image is assumed to consist of several components, one of them supporting the image content description and addressability of other image components. Existing de-facto standard image formats refer to such component as image manifest, which typically enables image content-addressability. An image manifest may point to other components necessary for the OS container runtime filesystem to be built. Existing de-facto standard image formats refer to such components as image layers. One or several layers may be required to create a complete OS container filesystem. The necessary ordered set of layers is also expected to be described in the image manifest. The image layers themselves can add, modify or remove files when applied on top of another image layer.

NOTE: The OCI™ Image Format Specification v1.0.1 [i.3] is identified to map to the OS container image of the NFV object model, as described in ETSI GS NFV-SOL 018 [i.4], clause 5.5. The OCI™ image consists of several components: manifest, image index, filesystem layers, configuration.

4.2.3 General assumptions on the CIS infrastructure

A CISM instance manages multiple containerized workloads in a cluster of CIS instances. The cluster operator is assumed to follow the general principle of least privilege for namespace definition and namespace quota assignment, and for access policy definition.

NOTE: In a multi-namespace environment, particular CISM internal services may be exposed to all namespaces.

4.3 Deployment Options

4.3.1 Container Platform

4.3.1.1 Introduction

ETSI GR NFV-IFA 029 [i.2] clauses 5.3.2 to 5.3.6 detail a number of deployment use cases for VNFs or VNF components that utilize container-based virtualisation depending on the capabilities of the NFVI. Clause 4.3 lists security considerations for the various deployment options.

NOTE: The option for Container-based NFV Micro-Services within the VNF is not considered as, in this option, the VNFC uses its own internal container technology to execute the VNF components, all running within VMs. This is equivalent to the traditional VM based operation, as neither NFV-MANO nor NFVI provide any container infrastructure services.

4.3.1.2 Container Infrastructure Services on bare metal

In this deployment option, the VNF runs entirely within OS level containers, which are provided by the CIS as depicted in ETSI GR NFV-IFA 029 [i.2], clause 5.3.3. Whether each single VNF component runs in its own container is dependent on the VNF and on how the managed container infrastructure object models are configured. The key aspects are that the NFVI provides the virtualisation layer that holds the container runtime environment as well as the OS for each container running directly on all NFVI hosts. The NFVI also provides the network for communications among VNFCs between NFVI hosts.

The LCM for VNFCs is performed by NFV-MANO using the usual templates such as VNFDs and VDUs. NFV-MANO however, consumes interfaces from the CISM with the latter doing all the necessary virtual resource allocation and management for the container infrastructure.

Due to VNFs running purely in OS containers without the use of VMs, this option is open to the general threats for containerized workloads that are described in clause 5.3. This includes vulnerabilities to the container runtime, such as privilege escalation, escape attacks, information loss through side channel attacks, and remote code execution.

Furthermore, as NFV-MANO entities make use of CISM northbound interfaces, the system is subject to CISM specific software and API vulnerabilities.

4.3.1.3 Container Infrastructure Services Nested within VMs

In this deployment option, the VNF runs within OS level containers, which in turn are executed within VMs as depicted in ETSI GR NFV-IFA 029 [i.2] clauses 5.3.4 and 5.3.5. Whether each VNF component runs in a single container or requires multiple containers is dependent on the VNF and on how the managed container infrastructure object models are configured. The key aspects are that the NFVI provides the virtualisation layer with hypervisors supporting container infrastructure services running within VMs, as well as the OS for each VM on all NFVI hosts. The NFVI also provides the networks for communications among VNFCs within and between VMs across NFVI hosts.

The LCM for VNFCs is performed by NFV-MANO using the usual templates such as VNFDs and VDUs. The VIM manages all the necessary virtual resource allocation for VMs when the container infrastructure is set up. NFV-MANO consumes interfaces from the CISM for managing containers which are also responsible for managing container-related resources.

Due to VNFs running in OS containers nested within VMs, in terms of security, this option gains the isolation that VMs provide, and the ability to follow affinity and anti-affinity rules for VNFC placement in VMs and NFVI nodes in addition to container-level security mechanisms. While this option is also open to the general threats for containerized workloads that are described in clause 5.3, the scope and surface area of these threats are reduced to the set of containers contained in each VM. As NFV-MANO entities make use of CISM northbound interfaces, this option is subject to CISM-specific software and API vulnerabilities.

4.3.1.4 Container Infrastructure Services both on bare-metal and in VMs

In this deployment option, the VNF runs within OS level containers, some of which are executed within VMs and some of which are run directly on hosts, as depicted in ETSI GR NFV-IFA029 [i.2], clauses 5.3.6. Whether each VNF component runs in a container bare-metal, or in a container within a VM is dependent on the VNF and on how the managed container infrastructure object models are configured. The key aspects are that the NFVI provides the virtualisation layer with hypervisors supporting container infrastructure services running within VMs, support for containers running directly on bare-metal, and the OS for each container or VM on all NFVI hosts. The NFVI also provides the virtual network for communications among VNFCs within and between VMs and containers across NFVI hosts.

The LCM for VNFCs is performed by NFV-MANO using the usual templates such as VNFDs and VDUs. The VIM manages all the necessary virtual resource allocation for VMs when the container infrastructure is set up. NFV-MANO consumes interfaces from the CISM for managing containers, which is also responsible for managing container-related resources.

This option, being a combination of containers over bare-metal and containers within VMs, inherits the security aspects of both. Containers running on bare-metal are open to the general threats for containerized workloads that are described in clause 5.3. And containers running within VMs gain the isolation that VMs provide, and the ability to follow affinity and anti-affinity rules for VNFC placement in VMs and NFVI nodes in addition to container level security mechanisms. And finally, as NFV-MANO entities make use of CISM northbound interfaces, this option is subject to CISM specific software and API vulnerabilities.

4.3.2 Orchestration and Management

4.3.2.1 Introduction

ETSI GR NFV-IFA 029 [i.2], clause 7.2.4 presents several options for mapping the CISM functionality to NFV-MANO. Two of the options (clauses 7.2.4.3 option #2 and clause 7.2.4.5 option #4) are excluded from the target architecture. Of the remaining options, two (clauses 7.2.4.6 option #5 and 7.2.4.7 option #6) are essentially the same, but with and without support for shared container services. This clause lists security considerations for the various deployment options.

4.3.2.2 CISM embedded in the VIM or as a stand-alone functional block

As detailed in ETSI GR NFV-IFA 029 [i.2], clauses 7.2.4.2 and 7.2.4.4, the CISM functionality could be embedded within the VIM or as a new NFV-MANO functional block.

In this deployment the containerized workloads (VNF or VNFC) are separated from the management functions not only logically but most commonly physically (NFV-MANO components running on separate compute nodes to workloads). This isolates management functionality in case of a containerized workload being attacked and the attacker being able to escape the container runtime. An attacker would be able to attack other containerized workloads on the CIS cluster node or attacking the CISM via the APIs exposed between the management function (CISM) and CIS cluster.

Within ETSI GS NFV-IFA 036 [i.5], clause 4.2.1, it states that CIS cluster nodes can be used as control nodes and therefore host the CISM management functionality. In this deployment option, the physical separation between the management functionality / CISM and the containerized workloads is removed. If the CIS cluster node itself is hosting both containerized workloads and the CISM, namespace separation may be the only mechanism protecting the CISM from a compromised containerized workload and complete cluster takeover.

4.3.2.3 CISM embedded into VNF

ETSI GR NFV-IFA 029 [i.2], clauses 7.2.4.6 and 7.2.4.7 detail the option of CISM being embedded into the VNF itself. The VNF may provide shared container services to one or more consumer, container-based, VNF instances (ETSI GR NFV-IFA 029 [i.2], clause 7.2.4.6).

In this deployment a compromised containerized workload may not only be able to attack other containerized workloads of the same VNF but also of other VNFs or the CISM depending on how the CIS cluster is physically and logically constructed. Where a CIS cluster node, either VM or bare-metal, hosts container workloads from different VNFs/VNFCs namespace separation may be the only mechanism protecting one containerized VNF/VNFC from another. As with clause 4.3.2.2 above, if the CIS cluster node hosts both a containerized workload and the CISM namespace separation may be the only mechanism protecting the CISM and complete cluster takeover.

5 Security Threats

5.1 Introduction

Clause 5 contains a structured method (based on ETSI GS NFV-SEC 006 [i.9]) to derive security requirements based on the assets that need to be protected, actors that may have an interest in attacking them, the attacks themselves, and the resulting requirements and mitigations that result. Both MITRE ATT&CK® in their Container Threat Matrix [i.6] and Microsoft® in their Kubernetes® Threat Matrix [i.7] have conducted detailed analysis of threats against containerized functions and MANO functional blocks. The analysis below takes the work done by MITRE ATT&CK® and Microsoft® into account.

5.2 Risk Analysis and Requirements

As depicted in Figure 5.2-1 (blue highlights), the security threats and requirements presented in this clause focus on OS Container management functional blocks (CISM, CIR, CCM), the CIS cluster infrastructure, the associated NFV-MANO reference points and any known means of implementing or mapping reference points to corresponding interfaces. Security threats (T) and their associated security requirements (R) are identified. For all threat scenarios, the assumption is that the attackers are attached to the network and have access to the NFV-MANO functional blocks, reference points and MCIO exposed interfaces and services. The present document does not include threat analysis and requirements for other NFV-MANO functional blocks and reference points; more details can be found in ETSI GS NFV-SEC 014 [i.8].

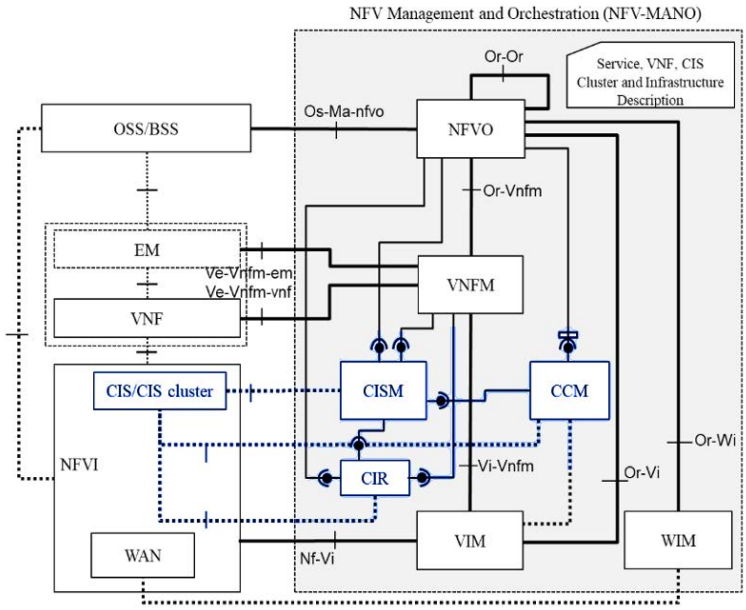

Figure 5.2-1: NFV-MANO Scope (blue highlights)

Although the code itself of software applications is an asset that needs to be protected, the reason this code is attached is to get to data manipulated by the code, which has inherent value that can be of benefit to the attacker outside the system. The threat analysis in Table 5.2-1 is based on the format found in Annex A of ETSI GS NFV-SEC 006 [i.9].

Table 5.2-1: Risk analysis and assessment

A Security Environment		
a.1 Assumptions		
a.1.1	Internal attackers have access to the network	
a.1.2	NFV-MANO functional blocks and reference points support NFV management entities	
a.1.3	Internal attackers have access to the CCM	
a.1.4	Internal attackers have access to the CISM	
a.1.5	CISM supports NFV OS container package operations	
a.1.6	CCM supports CIS management operations	
a.1.7	NFVO supports Network service instances and VNF instances operations	
a.1.8	NFV-MANO manages many OS Container MCIOs	
a.1.9	An NFVI is managed by only one NFV-MANO	
a.1.10	The CISM and CIS cluster may be virtualised entities created by CCM on demand	
a.2 Assets		
a.2.1	OS Container management functional blocks and reference points	
a.2.2	The credentials of authorized systems with legitimate access to the OS Container management functional blocks and reference points	
a.2.3	The credentials of authorized administrators with legitimate access to the CCM	
a.2.4	The credentials of authorized administrators with legitimate access to the CISM	
a.2.5	The credentials of authorized administrators with legitimate access to the CIR	
a.2.6	The credentials of authorized administrators with legitimate access to the CIS cluster	
a.2.7	The OS Container MCIO and exposed interfaces/services	
a.2.8	OS Container image	

a.3 Threat agent		
a.3.1	Malicious users or administrators with access the NFV-MANO OS Container functional blocks and reference points	
a.3.2	Malicious users with access to the containerized workload exposed interfaces/services	
a.3.3	Compromised containerized workload	
a.3.4	Malicious server with access to the NFV-MANO OS Container functional blocks and reference points	
a.4 Threats		
a.4.1	Exploit Externally-Facing containerized workload	See clause 5.3.3
a.4.2	Exploit Externally-Facing Interfaces/Services	See clause 5.3.5
a.4.3	Exploit Valid Accounts	See clause 5.3.5
a.4.4	Compromised OS Container Image	See clause 5.3.1
a.4.5	Instantiation of unauthorized containerized workload	See clause 5.3.5
a.4.6	Exploit OS Container management Interfaces	See clause 5.3.5
a.4.7	Compromise of administrators	See clause 5.3.5
a.4.8	Credential Manipulation	See clause 5.3.5
a.4.9	Containerized Workload Escape	See clause 5.3.4
a.4.10	Masquerade of NFVO to OS Container management function blocks (CCM, CISM, CIR)	See clause 5.3.5
a.4.11	Masquerade of VNFM to OS Container management function blocks (CISM, CIR)	See clause 5.3.5
a.4.12	Masquerade of CCM to CISM	See clause 5.3.5
a.4.13	Masquerade of CCM to CIS cluster	See clause 5.3.5
a.4.14	Masquerade of CISM to CIR	See clause 5.3.5
a.4.15	Masquerade of CISM to CIS cluster	See clause 5.3.5
a.4.16	Masquerade of CIS cluster to CIR	See clause 5.3.2
B Security Objectives		
b.1 Security objectives for the asset		
b.1.1	Ensure that there is a secure software development lifecycle for the containerized VNF	c.1.1.1
b.1.2	Ensure that the CIS infrastructure, CISM and CIS platforms have hardened secure configuration in accordance with industry best practice	c.1.1.2 c.1.1.3 c.1.1.4 c.1.1.5 c.1.1.6 c.1.1.7
b.1.3	Ensure that hardware security capabilities are utilized to protect sensitive VNFs	c.1.1.8
b.1.4	Ensure that the storage used by containerized VNFs is secure	c.1.1.9
b.1.5	Ensure that the transmission of sensitive containerized VNFs is secure	c.1.1.10
b.1.6	Ensure that OS container management and orchestration is operated in a secure manner	c.1.1.11 c.1.1.12 c.1.1.13 c.1.1.14 c.1.1.15
b.1.7	Ensure that workloads are segregated appropriately	c.1.1.16 c.1.1.17
b.1.8	Ensure the threats and risks to the CIS management and orchestration function are taken into account during the deployment design	c.1.1.18

C Security Requirements		
c.1 Asset security requirements		
c.1.1 Asset security functional requirements		
c.1.1.1	Secure the software development lifecycle	See clause 6.2
c.1.1.2	Security harden the CIS cluster host operating system	See clause 6.3.1
c.1.1.3	Configure namespace separation	See clause 6.3.2
c.1.1.4	Isolate and control access to resources using Linux® (see note) control groups	See clause 6.3.3
c.1.1.5	Restrict or drop unneeded Linux® capabilities from OS container workloads	See clause 6.3.4
c.1.1.6	Define and restrict which system call which an OS container workloads can access during runtime	See clause 6.3.5
c.1.1.7	Enable Mandatory Access Controls	See clause 6.3.6
c.1.1.8	For sensitive containerized VNFs instantiate the OS container within an HMEE	See clause 6.6.5.3
c.1.1.9	Secure the storage for sensitive OS container images	See clause 6.4
c.1.1.10	Secure the transmission of sensitive OS container images	See clause 6.5
c.1.1.11	Account management	See clause 6.6.1
c.1.1.12	Account authentication	See clause 6.6.2
c.1.1.13	Secure the Container Image Repository	See clause 6.6.3
c.1.1.14	Validate the containerized VNF packages during onboarding and instantiation	See clause 6.6.4.1
c.1.1.15	Segregate OS container workload network connectivity	See clause 6.6.5.1
c.1.1.16	Segregate OS container workloads of differing sensitivity levels	See clause 6.6.5.2
c.1.1.17	Protect the CISM from potential OS container escape	See clause 6.3.5 See clause 6.3.6 See clause 6.4
c.1.1.8	Deployment design	See clause 6.7
NOTE: Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.		

5.3 Threat Descriptions

5.3.1 Compromised Container OS Image

The primary source of vulnerabilities in containerized environments is in application and container images themselves. A common risk with containers is the usage of an image that contains vulnerabilities for generating a new container. A malicious container OS image can compromise the CIS cluster. Gaining access to the CIR can allow an attacker to install their own compromised images in the registry.

Container image configuration flaws: a common vulnerability in images is to create standard users with unnecessary high privileges. This kind of common configuration weakness increases the attack surface of the entire system.

Keys encoded in images: every administrator or user that has access to the image may gain an unauthorized access to all the information that is encoded in the image. Encoding sensitive information like passwords, private keys like SSH private keys or X.509 private keys exposes the whole system to internal and external attacks.

Use of third-party images: portability of containers encourages developers to build new container images from existing ones making the usage of third party resources. As a result, this increases the risk to reuse vulnerable images or corrupted ones. Configuration flaws may be embedded intentionally or inadvertently in these images. Even worse, some images can embed malwares or hard coded routines that allow downloading malwares after image deployment.

Vulnerabilities within container CIS: exploit of the container runtime may allow exploits for privilege escalation and escape attacks.

Compromise of MCIO descriptor: manipulation of the descriptor to weaken the security of the deployed containerized workload could allow an attacker to gain initial access.

5.3.2 Image Management

Hijacked Container Image Registry (CIR): image registries may be subject to attacks as they contain NFV images and confidential data like credentials and secret keys. If such an attack succeeds and OS container images are compromised, potentially all system will be compromised including applications and container OS. Such an attack may succeed through the exploitation of poor access policy configurations and software flaws. With this attack, unauthorized user may gain access to confidential data, modify the image repository, or in the worst scenario succeed to delete the entire image registry.

Connection to CIR: connection to registries through unsecure channels may allow attackers to intercept traffic and access to confidential data like credentials and secret keys.

Exploitation of outdated images: non patched, unmaintained images that are still present in repositories expose vulnerabilities that can be exploited by attackers to hijack the repository. Moreover, this practice increases the risk of deploying old image versions that contain unpatched vulnerabilities.

5.3.3 Containerized Workload Exploitation

Information leakage through side channel attacks: information leakage occurs when malicious process gains illegitimate access to a co-resident process' data. In the container use case, one container may be able to access information within another container's namespace. Spectre and Meltdown are examples of high-risk vulnerabilities that can be exploited by attackers to execute side channel attacks. These vulnerabilities affect containers and have been demonstrated in popular container platforms.

Theft of resource: the objective of theft of resource attacks is to unduly obtain resources at the expense of the provider or at the one of co-localized containers, possibly preventing them from properly benefiting from their own resources. This type of isolation breakout may result in billing issues, cross-container performance degradation, resource starvation and denial of service. Some Linux® scheduler versions are vulnerable to this attack.

MAC and ARP spoofing: when multiple containers share the same network card, one container may exploit driver vulnerabilities through spoofing of co-resident container MAC address. This attack allows a malicious user to receive authorized traffic, to delete the traffic or to redirect the traffic to an external entity.

Denial of Service attacks: as I/O resources are shared among multiple containers, a bystander container may suffer from an external attack targeting a co-resident container. Moreover, malicious users can exploit this resource sharing to attack co-resident container by running intensive I/O tasks at their own containers (e.g. auto network flooding). Performance sensitive functions may experience SLO violation and DoS.

Port scanning: vulnerable network function could have an exposed port that leaks its sensitive information upon port scanning. Exposed ports can lead to API abuse.

Externally-Facing Containerized Workload: could lead to unauthorized access to the CIS cluster. Exploiting an externally-facing containerized workload typically involves attackers targeting a containerized service or application that is accessible from outside the organization's network, such as a web frontend, APIs, or any container with a public IP or open network port. If the exposed containerized workload contains a vulnerability (either a software bug or a misconfiguration), threat actors may exploit it to gain unauthorized access and potentially escalate privileges to compromise the entire Kubernetes cluster.

5.3.4 Container Escape Attacks

Share of the same kernel: containers use OS-level virtualisation to offer lightweight virtualisation where containers run as user space processes that share the host kernel. Sharing the same host kernel among multiple containers makes them by-design less resistant to isolation breakout attacks. As an example, recent vulnerability CVE-2019-5736 allowed a maliciously crafted container to arbitrarily rewrite the containerization's core software (runC, used in most containerization engines), enabling it to get full root access over the host, incidentally taking control of the other containers collocated on the node and providing an entry point in the Demilitarized Zone (DMZ) for further attacks. The damage of these attacks may be exacerbated when workloads with different privilege levels share the same host OS.

Enlarged attack surface: in full VM virtualisation, the hypervisor plays the role of the software component responsible for running and managing resources allocated to different VNFCI. In the NFVI design, the hypervisor is considered as a thin software layer designed to be easily audited. Containers rely on general-purpose operating systems, not designed for multi-tenant environments, but to be compatible with most existing hardware architectures.

Tampering host file systems: a container with unnecessary high permissions can mount a local file system on the host to tamper with data belonging to other containers.

5.3.5 Orchestration

Externally-facing interfaces/services: Initial access may be gained through leveraging externally-facing interfaces (e.g. CISM APIs) and services. Those exposed interfaces and services may also be exploited to gain persistence within a network. Externally-facing interfaces and services may be used for interconnecting services with a hybrid cloud architecture or for interworking between different CSPs. They could also be accidentally exposed due to misconfiguration.

CISM API abuse: compromised containers may exploit vulnerabilities related to the management API to attack the other containers or the container OS. These vulnerabilities can allow malicious (including compromised) containers access to directories outside their volume(s) or to delete arbitrary host files and directories.

CISM specific vulnerabilities: software vulnerabilities (including API) within the management layer can lead to attack the orchestration layers and result in container escape.

Credential manipulation: an attacker may obtain and abuse credentials of existing accounts with access to the OS Container NFV-MANO functional blocks. Compromised credentials may be abused to gain initial access, persist existing unauthorized access, escalate privileges, or to evade security defences. An attacker may also create new credentials to maintain access to a compromised OS Container NFV-MANO functional block.

Unbounded administrative access: giving unnecessary high access privileges to users enlarges the attack surface of the whole platform. When access rights are not tailored to the specific needs of different users, the risk to attack or alter container execution either maliciously or inadvertently by users increases.

Valid accounts: an attacker could abuse credentials of existing accounts (e.g. default accounts, local accounts or service level accounts) as a means of gaining Initial Access, Persistence, Privilege Escalation, or Defence Evasion. Account details may be exposed through configuration with embedded credentials being uploaded to support forums or saved to public code repositories. Exploitation of such accounts could lead to unauthorized access to CIS cluster APIs and services leading to cluster takeover.

Compromise of administrator: administrators with legitimate access to the OS Container NFV-MANO functional blocks (CISM, CIR, CIS cluster) may be exploited to execute malicious code, making changes to configuration or to instantiate a compromised/malicious OS Container VNFI/VNFCI images.

Instantiation of unauthorized containerized workload: an attacker may instantiate an unauthorized MCIO container into an CIS cluster containing malicious code or attack tools to facilitate further exploitation or to evade defences. An attacker may also instantiate a valid OS Container image with weakened configuration to bypass existing defences within the environment.

6 Security Requirements

6.1 Introduction

In contrast to virtual machines that run their own kernel instances, containerized workloads running on the same physical host share the same OS kernel. Sharing this host OS kernel enlarges the attack surface of the network functions that are executed within containers. While major current container implementations offer software solutions to enforce isolation between containers, there are still multiple security challenges that require complementary hardware solutions. This clause details security requirements and software and hardware solutions to mitigate the threats identified in clause 5 above.

6.2 Software Development Life Cycle

Securing the Software Development Life Cycle (SDLC) is essential to protect sensitive data, prevent unauthorized access, and mitigate security risks throughout the development process. By embedding security requirements, practices, and controls from the earliest phases of software design, vulnerabilities can be proactively identified and addressed before they reach production. This approach helps ensure robust software integrity, regulatory compliance, and the overall reduction of operational risk.

- Sensitive data and credentials shall never be stored insecurely within application code, configuration files, repositories, or cloud storage (e.g. avoid plaintext passwords or publishing secrets to version control).
- All credentials and secrets shall be managed securely with access controls, encryption, and secret management tools.
- Third-party libraries shall be evaluated for security risks prior to integration. Developers should monitor for vulnerabilities and restrict dependencies to specific, approved versions rather than automatically updating during builds.
- Implement security gates and automated checks within the CI/CD pipeline to prevent insecure or non-compliant code from being built, merged, or pushed to Container Image Registry (CIR).
- Conduct regular security reviews, code scanning, static and dynamic analysis, and dependency checks as part of development and before deployment.
- Developers should receive ongoing training on secure coding standards, best practices, and emerging threats.
- Enforce audit logging and change tracking throughout the development environment to ensure accountability and enable rapid response to security incidents.
- Security policies and requirements shall be defined at the outset and revisited throughout the project to align with regulatory and compliance obligations.
- All software shall be scanned for vulnerabilities using appropriate tools (e.g. Static and Dynamic Application Security Tools, vulnerability scanners, protocol fuzzers).

By following these requirements, threats a.4.3, a.4.4 and a.4.8 can be mitigated and help safeguard sensitive data and maintain secure, resilient software across the entire SDLC.

6.3 Platform Security Hardening

6.3.1 Operating System Hardening

Operating Systems (OS) hardening to industry or OS vendor best practices reduces the threats to the OS removing many of the simpler attack vectors. Hardening benchmarks and guides, such as those available from the Center for Internet Security® cover several common operating system areas to be configured:

- **Filesystem:** including removing support for unused and therefore unnecessary kernel modules in Linux and filesystem partitioning and permissions to reduce the threat from capacity exhaustion and unauthorized access.
- **Software and operating systems features:** removal or disablement of unused and unnecessary software packages and features.
- **Mandatory Access Controls:** enforce strict policy-based access to system resources.
- **Bootloader configuration:** restricting access permissions to the bootloader configuration.
- **Additional process hardening:** Enabling Address Space Layout Randomization (ASLR).
- **Service configuration:** disabling any unused and unnecessary services and daemons.

By hardening the host OS threats a.4.2, a.4.6, a.4.8 and a.4.9 can be mitigated.

When considering the base OS for OS Container images, it is also important to minimize the software packages, utilities and libraries. Not only will this reduce the size of the resulting OS Container image, but it will also remove potential attack vectors.

6.3.2 Namespaces

Namespaces are system abstractions designed to create resource virtualisation at the process level. Namespaces ensure that processes running within a container cannot access or see resources of processes running within another container. To do so, containers are executed as runtime processes within given namespaces. By limiting the visibility that a group of processes has over system resources, namespaces control what resources a container can see. The isolated resources include process pids, filesystem mounts, network stack, user IDs, etc. Table 6.3.2-1 summarizes available Linux namespaces with the resources that are isolated by each of them. Namespaces are created through clone and unshare system calls. A namespace keeps existing on the system until the last of the processes that belong to it is destroyed.

Table 6.3.2-1: Namespaces with isolated resources

Namespace	Command	Isolated resource
Cgroup	CLONE_NEWCGROU	Cgroup root directory
Namespace IPC	CLONE_NEWIPC	System V IPC, POSIX, message queues
Network	CLONE_NEWNET	Network devices, stacks, port, etc.
Mount	CLONE_NEWNS	Mount points
PID	CLONE_NEWPID	Process ID
User	CLONE_NEWUSER	User and group IS
UTS	CLONE_NEWUTS	Hostname, NIS domain names

6.3.3 Control Groups

Cgroups is designed to organize processes hierarchically and control how many resources can be used by them. By doing so, Cgroups ensure performance and resource isolation among multiple containers. CGroups rely on resource tracking and limiting. Thanks to Cgroups, one container cannot consume more resources (CPU, memory, storage, network) than its fair share. This protects container infrastructures from resource starvation attacks that can target either co-localized containers or the host OS. To summarize, cgroups provide four main functions:

- **Resource control:** controls container status through commands (stop, restart, frozen).
- **Resource limit:** defines the amount of computing resources that can be used by a given container.
- **Accounting:** monitoring of consumed resources by a given group of containers.
- **Prioritization:** during resource contention, it allows to give more resources for containers that belong to high priority group.

There are twelve cgroups that control different resources:

- cpu: providing cgroup tasks access to the CPU.
- cpusets: assigning a set of CPUs and memory nodes to cgroups. Tasks in a cpuset cgroup may only be scheduled on CPUs assigned to that cpuset.
- blkio: limits per-cgroup block io.
- cpuacct: provides per-cgroup cpu usage accounting.
- devices: controls the ability of tasks to create or use devices nodes using either a blacklist or whitelist.
- freezer: provides a way to 'freeze' and 'thaw' whole cgroups. Tasks in the cgroup will not be scheduled while they are frozen.
- hugetlb: facilitates limiting hugetlb usage per cgroup.
- memory: allows memory, kernel memory, and swap usage to be tracked and limited.

- net_cls: provides an interface for tagging packets based on the sender cgroup. These tags can then be used by tc (traffic controller) to assign priorities.
- net_prio: allows setting network traffic priority on a per-cgroup basis.
- cpu: enables setting of scheduling preferences on per-cgroup basis.
- perf_event: enables per-cpu mode to monitor only threads in certain cgroups.

6.3.4 Linux Capabilities

It is possible within Linux® OS Container workloads to restrict or drop certain Linux capabilities. By using such capabilities, it is possible to lock down the root account within the container. There are thirty-eight capabilities which were added over time as different versions of the Linux kernel were released. When an application running within a container is executed with only the necessary capability, this protects the container from any malicious exploits that target services running without root privileges.

6.3.5 Secure computing mode

Seccomp is a Linux security tool that allows administrators to define system call security that shall be blocked during container runtime. This Linux feature relies on the Seccomp system call that is called from the container during its execution. The seccomp system call executes a Berkeley Packet Filter (BPF) program. The seccomp feature defines a permit list of system call profiles.

Seccomp policies are defined using JSON files. It is particularly useful to manage situations where a container requires a critical capability, e.g. CAP_SYS_ADMIN, the seccomp control will restrict the syscall that would allow an attacker to escape from the container and gain access to the underlying file system. If the syscall mount is not allowed with capability CAP_SYS_ADMIN, most of the capacity-based escape techniques would fail.

6.3.6 Mandatory Access Control

Mandatory Access Control (MAC) is a Linux security framework that enforces strict policy-based access to system resources, preventing processes, including those running with elevated privileges, from performing unauthorized actions. Unlike Discretionary Access Control (DAC), where resource owners determine access rights, MAC policies are centrally defined and enforced by the kernel, ensuring consistent control across all users and processes. This model is critical in containerized environments, where an attacker compromising one container could otherwise exploit shared kernel resources or file systems to escalate privileges or move laterally within the host system.

Security-Enhanced Linux (SELinux) and AppArmor are the two primary MAC implementations in Linux that greatly reduce the risks posed by a compromised container. SELinux uses a label-based model, assigning security contexts to processes, files, and other system objects; it enforces access decisions using finely grained policies defined in Mandatory Access Control rules. This approach isolates containers both from the host and from one another using mechanisms such as Multi-Category Security (MCS), which ensures that even if one application or container is breached, the attacker cannot interact with or compromise others. AppArmor, on the other hand, adopts a path-based model, applying predefined profiles that restrict applications to only specific files, capabilities, or network resources. This makes AppArmor easier to manage in dynamic environments while still providing effective isolation by limiting what compromised processes can access.

SELinux and AppArmor should be used to strengthen container security by confining processes within strict security boundaries and preventing privilege escalation or host escape attacks. When integrated with container runtimes, these tools enforce the principle of least privilege, ensuring containers can perform only the exact actions required and nothing more.

6.4 Storage

Persistent data storage should be encrypted at rest to protect data used by the containers. Storage of container image hardening is covered by clauses 6.6.3 and 6.6.4. Protection of data at rest using transparent encryption is covered further in ETSI GS NFV-SEC 026 [7], clause 5.3.3. It is also possible to use read-only volumes where an OS Container workload only needs access to data (e.g. configuration files). By making the storage volume read-only it will prevent an attacker from persisting malicious files or the changing configuration. Container workloads should also have limited access to the host file system which will help mitigate potential avenues for container escape.

6.5 Secure communication

Communication should be encrypted to protect data in transit. All MANO APIs shall be protected against loss of confidentiality, integrity, and availability, in line with ETSI GS NFV-SOL 013 [8]. For intra-container communication, hardware access controls may suffice, negating the need for encryption. For multi-tenancy deployments, see ETSI GS NFV-SEC 026 [7], clause 5.3.2.

6.6 Operational Security

6.6.1 Account Management

It is important to manage accounts not only within the containerized Network Service but also within the NFV-MANO functions and host OS. By enforcing the principle of least privilege, the effect of a compromise of user or machine accounts can be minimized. Reducing the number of highly privileged accounts will also reduce the attack surface. Accounts and their permissions should also be regularly audited for activity and any inactive accounts disabled or removed.

It is also important to locate and remove (where possible) all default accounts. Default accounts become commonly known and can be used to gain initial access or to escalate privileges within a compromised system. As a minimum, these accounts shall either have their password changed or removed and the account locked/disabled.

Accounts used by NFV-MANO functions (e.g. the NFVO, VNFM, CCM, CISM, etc.) shall also be audited and their permissions set to only permit operations that the function requires to operate.

6.6.2 Account Authentication

Legacy authentication (e.g. basic authentication using usernames and passwords), shall be disabled and the use of modern authentication protocols shall be used instead. For user access to containerized Network Services or NFV-MANO function Multi-factor Authentication (MFA) and conditional access (utilizing policies to block logins from non-compliant devices or from invalid geolocations) should be used. MFA provides a critical layer of security by requiring multiple forms of verification. The use of MFA can mitigate the threat of compromised accounts. It is important to train users in the proper use and handling of credentials including One Time PINs and challenges used by Multi-Factor Authentication. There are a number of common phishing and spear phishing techniques which try and trick users into accepting invalid OTP or push notifications. Users should be trained in how to spot these and only accept valid requests and to report anything suspicious. For MANO API authentication and authorization ETSI GS NFV-SOL 013 [8] and ETSI GS NFV-SEC 022 [9] shall be followed.

Where basic authentication is used, security policy shall enforce the use of high entropy passwords (complex, unique passwords. Although the policy of password rotation is now largely discouraged it is vital to monitor accounts for signs of compromise or attacks against a credential. Where this is detected the account password should be changed and the event investigated.

Where passwords and API keys are used, they should be stored in a secure vault / Key Management System (KMS).

6.6.3 Container Image Registry Protection

The Container Image Registry is a critical part of the overall Container Infrastructure Service as it performs all OS container image management service. As detailed in ETSI GS NFV-IFA 040 [1], the CIR function offers multiple types of OS container image management services. These are exposed via OS container image management service interfaces. A compromise of the CIR or its management service interfaces could allow an attacker the ability to delete container images (resulting in a loss of availability and Denial of Service), upload a malicious or weakened OS container image (potentially allowing for further compromise and persistence within the CIS cluster).

The use of strong access control policies (e.g. Attribute and Role Based Access controls), authentication and encryption of the management and control planes (to the CIR host OS and to the CIR OS container image management service interfaces) can mitigate many threats. In addition, the registry should be monitored and images reviewed so that stale (OS container images which are no longer needed or are of an outdated version and may have vulnerabilities) or duplicate images can be removed.

The CIR should also perform OS container image validation and integrity checks. This is covered more in the following clauses.

6.6.4 Image Management

6.6.4.1 Validation of the containerized VNF Package during onboarding and instantiation

6.6.4.1.1 Introduction

The present clause describes the validation of the containerized VNF Package during onboarding by a service provider and the validation of the MCIOP file artifacts and OS container images artifacts during instantiation of containerized workloads. The requirements specified in ETSI GS NFV-SEC 021 [2], shall apply. In addition, it is recommended that the cryptographic signature metadata be stored in an immutable tamper resistant ledger.

NOTE: The containerized VNF Package contains one or multiple MCIOPs.

6.6.4.1.2 Validation of the containerized VNF Package during onboarding

The requirements specified in ETSI GS NFV-SEC 021 [2], clause 5.1 shall apply.

During the onboarding of the containerized VNF Package, a validation of the VNF Package is performed. As described in ETSI GS NFV-SOL 004 [3], the validation relies on the existence in the NFVO of a root certificate of a trusted CA that shall have been delivered via a trusted channel that preserves its integrity (separate from the VNF package) to the NFVO and be pre-installed in the NFVO before the on-boarding of the VNF package. Where the trusted CA is a public entity (e.g. a commercial CA), the signing certificate shall be securely pre-installed into the NFVO. During verification of the containerized VNF package the certificate within or with the package shall be checked with those which are already trusted before checking the rest of the chain of trust.

Assumption: Service provider has obtained from the VNF provider in advance all necessary VNF Package artifacts (keys, certificates, other information) in order to proceed with the process below.

The procedures for containerized VNF Package signing, in CMS format, and verification shall comply with ETSI GS NFV-IFA 011 [4] and ETSI GS NFV-SOL 004 [3], with the following additions:

1) The service provider shall ensure scanning for vulnerabilities and malware (either automated or manually driven) within the OS container image(s) is performed at on-boarding prior to any other onboarding or validation. Any high or critically rated vulnerabilities (as defined in clause 6 of the Common Vulnerability Scoring System v4.0: Specification Document [6]) or any malware shall prevent automatic certification and onboarding.

NOTE 1: Any manual onboarding of OS container images which do not fully pass the test criteria should have a full risk assessment, appropriate sign off and be recorded within an audit trail.

2) NFVO shall obtain a VNF provider-signed containerized VNF Package.

3) NFVO shall verify the VNF provider's signature(s) of the containerized VNF Package.

4) Service provider shall perform necessary steps to validate and test the containerized VNF Package as described in the use case "VNF Package validation and certification" in ETSI GS NFV-IFA 011 [4], clause 5.5.

This shall include scanning for vulnerabilities and malware (either automated or manually driven) within the OS container image(s). Any high or critically-rated vulnerabilities (as defined in clause 6 of the Common Vulnerability Scoring System v4.0: Specification Document [6]) or any malware shall prevent automatic certification and onboarding.

NOTE 2: Any manual onboarding of OS container images which do not fully pass the test criteria should have a full risk assessment, appropriate sign off and be recorded within an audit trail.

5) Optional step: if the service provider policy mandates signing containerized VNF Package artifacts, NFVO shall sign the containerized VNF Package artifacts using the appropriate signing key(s).

6) NFVO shall store the signed containerized VNF Package artifacts in the corresponding catalogue(s). Provided the VNF Package includes the OS container images, NFVO shall store the signed OS container images in the corresponding CIR(s). The signed MCIOPs file artifacts shall be stored in the corresponding repository.

6.6.4.1.3 Validation of the MCIOP file artifacts and OS container images during instantiation

The requirements specified in ETSI GS NFV-SEC 021 [2], clause 5.2 shall apply.

The objective of this step is to verify the containerized VNF Package artifacts authenticity and integrity during instantiation. Prior to instantiation of the containerized workloads, if the service provider policy for onboarding includes signing of containerized VNF Package artifacts, those signature(s) shall be verified. The pre-loaded root certificate, along with any other pre-loaded certificates in the chain of trust, are used in conjunction with the certificate chain, that is part of the VNF package, to validate the integrity and origin of the OS container images and MCIOP file artifacts.

Assumption: the cluster of CIS instances and the CISM function are available at the point when the containerized VNF is instantiated.

Verification of the integrity and authenticity of the OS container images at OS container image instantiation shall be performed at CIS instance runtime execution level.

The validation of the MCIOP file artifacts and OS container images during VNF instantiation shall be performed as follows:

- The signature validation of the MCIOP file artifacts is performed by the consumer of the CISM OS container workload management service interface. The consumer of this interface is responsible for validation when the MCIOP file artifacts are pulled from the corresponding repository in order to instantiate containerized workload(s) based on the MCIOP. The pre-loaded root certificate is used in conjunction with the certificate chain, that is part of the VNF package, to validate the integrity and origin of the MCIOP file artifacts.

- The signature validation of the OS container images when pulling from CIR is performed by the CIS instance at CIS instance runtime level. The CIS instance is responsible for validation at container runtime level when image pull is done from the CIR and the images are instantiated. The pre-loaded root certificate is used in conjunction with the certificate chain that is part of the VNF package, to validate the integrity and origin of the OS container images.

6.6.4.1.4 OS container image software integrity validation at instantiation with attestation

The present clause describes an alternative solution for validating the OS container software integrity during instantiation. This alternative solution can be employed in attestation environments and ensures a binding between the OS container image post-build stage by the VNF provider, and the container instantiation stage (e.g. by the service provider or tenant).

The current solution is applicable to deployments where the VNF provider indicates the entire OS container filesystem including the application software (i.e. the service), except for an indicated list of absolute paths, that can be measured for integrity during VNF instantiation. The measurement is realized as a digest (i.e. with a given hash function) performed over the indicated OS container filesystem excluding the indicated list of absolute paths.

When an OS container image is launched, the resulting OS container filesystem is instantiated from the merged image layers according to the image manifest rules. Knowing that the content of some file(s) may be different and unpredictable per container instance when the OS container image is launched, such files are expected to be part of the indicated list of absolute paths, which is excluded from the filesystem integrity measurement. Checks shall be performed on such a list based on an attribute that the vendor specifies for the named files in the list of absolute paths. Such checks ensure that files with attributes different from those specified are not allowed.

The indicated OS container filesystem and the list of absolute paths excluded from the integrity measurement are part of a configured policy, hereafter referred to as the "software digest policy", which is set during the OS container image build phase by the VNF provider. The software digest policy shall include the measurement reference values.

The software digest policy for each of the OS container images contained in or referenced from the VNF package shall be included in the VNF package and shall be referenced from the VNF package manifest file.

The main functions for this solution are:

- Attest Client: this is a component within every OS container acting as an interface between the associated Dynamic Root of Trust Measurement and the Attestation Verifier Service instance in the corresponding cluster of CIS instances.

- Attestation Verifier Service (AVS): this is a component responsible for managing the reference values and attesting the instantiating OS containers that are subject to this solution. At least one AVS instance is deployed per cluster of CIS instances and represents a cluster-level AVS. It appraises the validity of quotes received from the Attest Clients and DRTMs, based on the reference values in the software digest policy from the VNF package, and produces attestation results to be used by other parties. The AVS acts as a decision gate keeper in case of verification failures and can interact with the cluster CISM instance to handle the required OS container lifecycle actions. Multiple AVS instances are assumed to be deployed in a hierarchy, with a central Remote Attestation Verifier Service (RAVS) at the highest level. An AVS instance consumes NFV-MANO interfaces for VNF LCM occurrence events and can query VNF package information.

NOTE 1: The AVS can be a generic service. The remote AVS can be located at a central location and/or as a tenant function.

- Dynamic Root of Trust Measurement (DRTM): this is a component responsible for performing the required measurement based on a given software digest policy, which is received upon indication from the associated Attest Client, during the OS container image launch. The outcome of this measurement is a verifiable digest that can be matched by the AVS instance with the set of reference values. The matching result provides evidence on the launched and running OS container image. If a list of absolute paths is present in the software digest policy, indicated to be excluded from the filesystem integrity measurement, then additional checks are performed by DRTM based on an attribute that the vendor specifies for the named files in the list of absolute paths to ensure that files with attributes different from those specified are not allowed. There are two types of DRTM:

    1) "internal-to-container" DRTM (iDRTM), which is part of the OS container image; and

    2) "external-to-containers" DRTM (eDRTM), which is not part of the OS container image and is associated with the CIS instance.

  DRTM may be designed to utilize existing security hardware. Verification that such utilization actually happens lies at the AVS. DRTM embeds the RTR and RTS functional-wise, but the specific nature of these RoT engines depends on the DRTM implementation.

NOTE 2: Examples of DRTM realizations include:

    1) a pre-installed host OS program (and/or a kernel eBPF program) deployed per CIS instance as an eDRTM;

    2) part of an OS container image, a program executing in an HMEE as iDRTM.

The AVS instances shall obtain the software digest policy from the VNF package.

The deployment example illustrated in Figure 6.6.4.1.4-1 shows how the eDRTM associates with a CIS instance in a cluster.

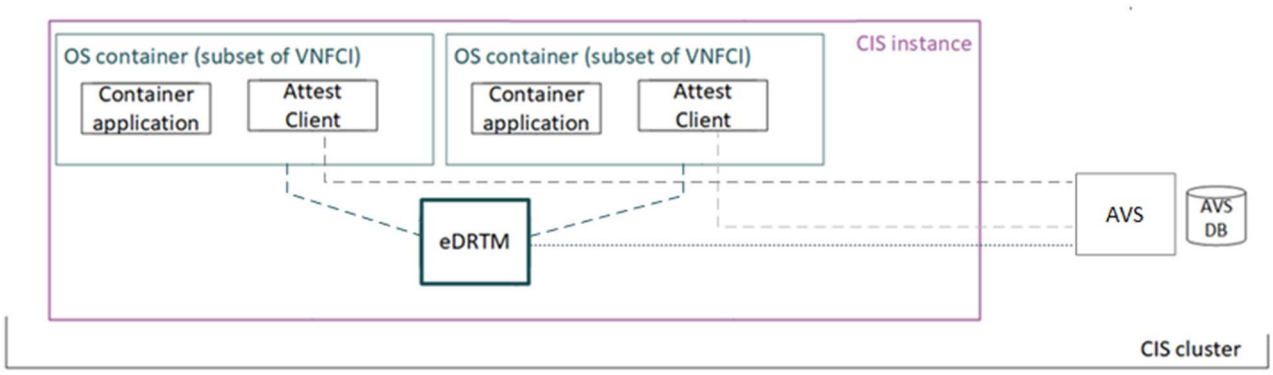

Figure 6.6.4.1.4-1: eDRTM association with a CIS instance running multiple containers

The main steps for the OS container image validation at instantiation corresponding to the solution described in the present clause where DRTM is instantiated per CIS instance are illustrated in Figures 6.6.4.1.4-2, which represents an example of procedure flow for NS/VNF instantiation.

NOTE 3: The exact definition of the different steps and handling in the NS/VNF instantiation procedure (e.g. interactions between the NFV-MANO entities and CISM/CIR/CIS instances, validation of the MCIOP file artifacts as per clause 6.6.4.1.3, secure bootstrap of AVS instances) is not shown.

The procedure in Figure 6.6.4.1.4-2 does not consider the use of the iDRTM and works as follows:

1) Prior to NS/VNF instantiation, a remote AVS (RAVS) instance can be provided with known manifest information (step-1a). Also, the root certificate of a trusted CA used for signature validation is expected to be available at the eDRTM (step-1b).

2) The RAVS can register for consuming NFV-MANO interfaces for VNF LCM occurrence events and for querying VNF package information.

3) Any cluster-level AVS instance is expected to be attested by the RAVS instance at the cluster creation. Reference values can also be managed from the central RAVS instance.

4-5) For every VNF instance added to the NS instance: the software digest policies associated with the OS container images are obtained by the AVS(s) through querying the VNF package information and the selected artifacts.

6) The cluster-level AVS validates the manifest file signature.

7) The Attest Client in a launched OS container image shall be called first.

8) A request from the Attest Client to the cluster-level AVS (i.e. the cluster of the given CIS instance where the OS container image is launched) is sent.

9) The Attest Client receives the signed digest policy indication from the AVS, necessary for, e.g. the corresponding OS container filesystem to be measured.

10) The Attest Client indicates or makes the given software digest policy available to DRTM to perform the measurement. The digest policy signature is first verified by DRTM.

11-12-13) According to the software digest policy, the integrity measurement is taken and the checks on a possible non-empty list of absolute paths performed. The result is signed by DRTM and reported to the cluster-level AVS instance (step-12). Any container-internal measurement may be forwarded as well (see note 4).

NOTE 4: On step-12: for sending the signed digest quote, the DRTM can be configured to reach the cluster-level AVS service when the CIS instance is added to the cluster. On step-13: if the OS container includes its own HMEE instance, then it can provide its own measurements.

14) The AVS instance compares the received digest measurement with expected reference values.

15) The attestation response is received by the container application via the Attest Client and proceeds with the next steps accordingly.

16) In case of a successful OS container image validation in step-14, the container is admitted to the cluster.

17) In case of a failed OS container image validation in step-14, the application logic shall handle the error.

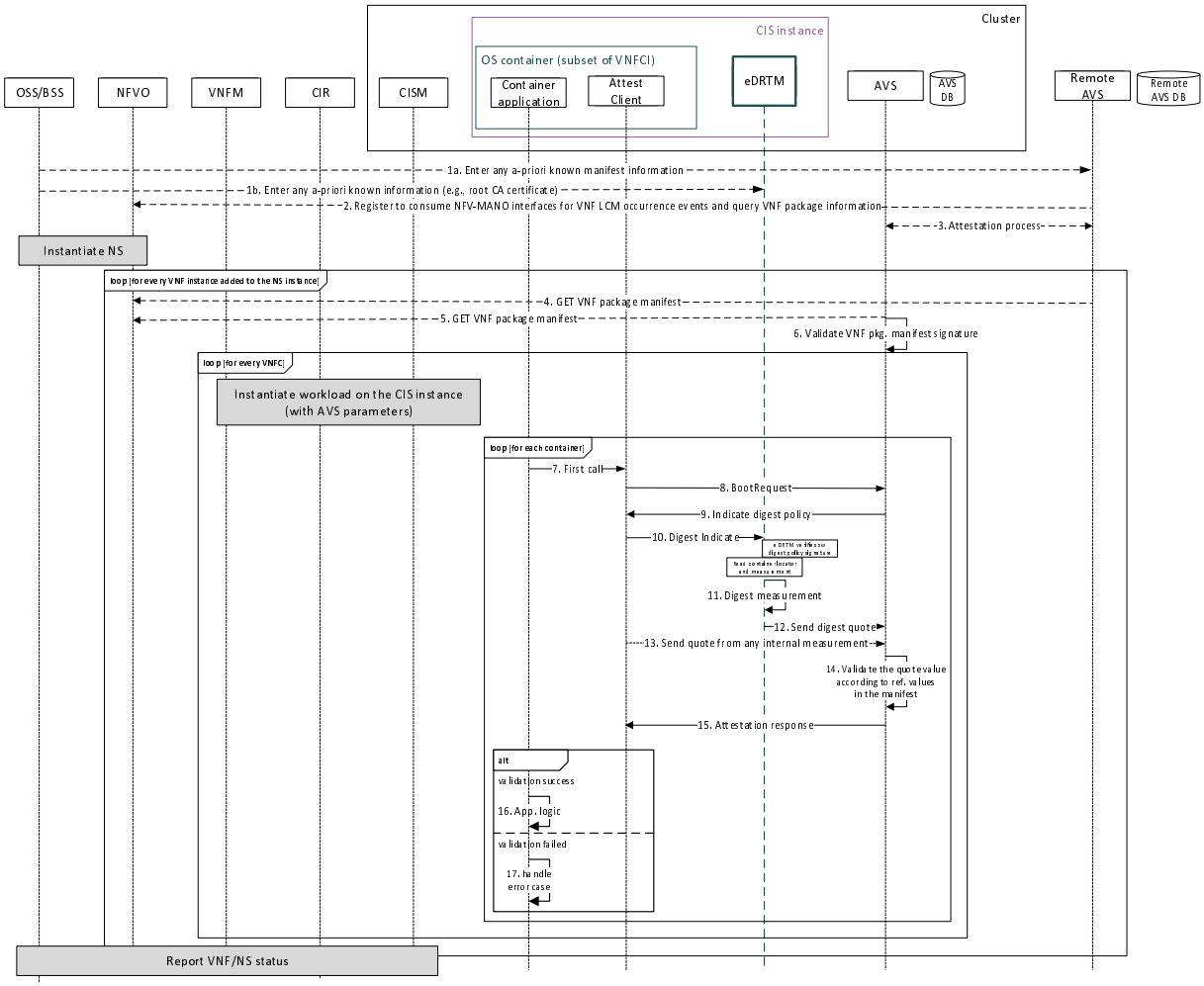

Figure 6.6.4.1.4-2: OS container image validation with AVS

6.6.5 Segregation and Isolation

6.6.5.1 Network Segregation

In containerized environments, separating the control plane, management plane, and data plane traffic is foundational to limiting the blast radius of compromise and maintaining the integrity of workloads. Each plane performs distinct operational functions: the control plane orchestrates the cluster, the management plane handles system monitoring and configuration, and the data plane supports application-level communication. In the NIST SP 800-190 [i.10], it emphasizes that container orchestrators such as the CISM shall be secured through least-privilege configurations, network segmentation, and strong access controls to reduce risks of lateral movement.

Isolating the control, management and data planes ensures that a compromise in one plane cannot cascade into others. Micro-segmentation enforces traffic control through fine-grained security policies that define permissible communication paths between pods, namespaces, and external interfaces. This approach aligns with the NIST SP 800-190 [i.10] recommendation to implement strict network segmentation for containerized environments and is reinforced by NFV security architecture principles. This separation ensures that workloads cannot directly communicate with control plane services without explicit authorization, preserving operational integrity and resilience against unauthorized access.

Restricting inter-container traffic between workloads using micro-segmentation also limits lateral movement within the cluster, a key defence against advanced persistent threats. By adopting a "default deny" network policy posture, organizations ensure that only explicitly approved traffic patterns occur, reducing the attack surface significantly. Implementing micro-segmentation as part of network security policy strengthens the cluster's resilience, ensuring that communication between applications, plane components, and external services is tightly governed.

As well as segregating network traffic, it is important to perform network monitoring for signs of compromise or attempted unauthorized access. Network traffic characteristics and telemetry can be combined with other forms of logging and monitoring data to aid in Threat Monitoring and can be used by the Security Manager.

6.6.5.2 Workload Segregation and Isolation

The use of host-based process segregation, such as namespace separation (see clause 6.3.2), can limit the impact of a compromised container. To further enhance and protect the host and cluster, it is recommended to segregate workloads based on their sensitivity level, threat posture and trust levels. For wider requirements to ensure isolation of sensitive workloads from non-sensitive workloads sharing a platform, see ETSI GS NFV-SEC 012 [5]. This should, as a minimum, be done at a host level, with threat modelling and risk reviews determining whether it should extend to the whole cluster.

This can be achieved through:

1) The use of security policy enforcing affinity and anti-affinity rules.

2) The use of HMEEs and Confidential Containers (see clause 6.6.5.3).

3) Using micro-Virtual Machines (microVM) to implement VM level memory and process isolation.

4) Limiting hosts and clusters to workloads of the same sensitivity and trust level.

By using these techniques, technologies and strategies, the potential impact of an OS container runtime breach can be restricted and lateral movement to more sensitive workloads, if not fully prevented, greatly reduced.

6.6.5.3 Confidential Computing Support

Where containers of different security classifications are instantiated within the same host system, software hardening solutions such as namespace separation may not be sufficient. To mitigate the risk of compromised containers of a lower security classification being able to affect the operation or confidentiality of containers of a higher classification the higher classification container should be instantiated within an HMEE.

Protection of data in use using an HMEE is covered in more detail within ETSI GS NFV-SEC 026 [7], clause 5.3.4.

6.7 Deployment Design

To mitigate the threat from a compromised containerized workload and the potential for lateral movement to the CISM it is important to look at how the CISM is deployed.

Where the CISM is deployed as a stand-alone functional block (see clause 4.3.2.2), to mitigate the threat from a compromised containerized workload, the management functionality / CISM should be deployed either onto dedicated bare-metal compute or as a virtual machine (a hybrid CIS cluster).

Although it is possible for one or more of the CIS cluster nodes to act as control nodes hosting the CISM it is not recommended as this will weaken the separation between management and workload functions.

To mitigate the threat from a compromised containerized workload where a containerized VNF supports shared container services (see clause 4.3.2.3), it is advised that any sensitive containers are instantiated within an HMEE (see clause 6.6.5.3). As with the scenario above, the CISM should be deployed either onto dedicated bare-metal compute or as a separate virtual machine (a hybrid CIS cluster).

Annex (informative): Change history

Date	Version	Information about changes
12-2019	0.0.1	First Draft
05-2020	0.0.2	Agreed output of NFVSEC#157 & 163 (NFVSEC(20)000004, NFVSEC(20)000030r1)
05-2020	0.0.3	Agreed state of TS at NFVSEC#164 meeting start
06-2020	0.0.4	NFVSEC(20)000040r1 (NFVSEC#164-e-meeting
06-2020	0.0.4	NFVSEC(20)000043r1(agreed output of NFVSEC#164-e-meeting)
06-2020	0.0.4	Agreed output of NFVSEC#165
07-2020	0.0.4	NFVSEC(20)000052r2 & NFVSEC(20)000052r3
05-2022	0.0.5	NFVSEC(22)000016 (Agreed output of NFVSEC#204-e-meeting)
09-2022	0.0.6	NFVSEC(22)000041r4 (Agreed output of NFVSEC#214-e-meeting)
11-2023	0.0.7	NFVSEC(23)000194r1 (Agreed output of NFVSEC#239/NFV#43 meeting)
10-2024	0.0.8	Agreed output of NFVSEC#272 (NFVSEC(24)000186, NFVSEC(24)000187r1)
06-2025	0.0.9	NFVSEC(25)000039r1 (Agreed output of NFVSEC#283 @ NFV#49)
10-2025	0.0.10	NFVSEC(25)000122r1 (Agreed output of NFVSEC#297) & NFVSEC(25)000127 (Agreed output of NFVSEC#298)
11-2025	0.0.11	NFVSEC(25)000141 (agreed output of NFVSEC#299) & NFVSEC(25)000150 (agreed output of NFVSEC#300)
11-2025	0.0.12	NFVSEC(25)000154r2 (agreed output of NFVSEC#301)
11-2025	0.0.13	New draft based on latest ETSI GS skeleton
12-2025	0.0.14	Corrected clause 5.1 reference to NFV-SEC 012 [6] which should have been NFV-SEC 006 [5]. Also changed occurrences of "must" to "shall".

History

Version	Date	Status
V5.4.1	April 2026	Publication