**3GPP TSG-SA3 Meeting #101-e *S3-203163***

**e-meeting, 9th – 20th November 2020** Revision of S3-20xxxx

**Source: Nokia, Nokia Shanghai Bell**

**Title: SCAS VNP: Secure Execution Environment**

**Document for: Approval**

**Agenda Item: 5.2**

# 1 Decision/action requested

***SA3 is kindly asked to approve the proposed changes in TR 33.818 v0.8.0.***

# 2 References

[1] 3GPP TR 33.818 v0.8.0 Security Assurance Methodology (SECAM); and Security Assurance Specification (SCAS); for 3GPP virtualized network products

# 3 Rationale

In current TR 33.818 v0.8.0 [1], the analysis of threats on interface between 3GPP VNF and virtualisation layer for GVNP type 1 is not quite complete. This pCR proposes additional text for the threat analysis in clause 5.2.4.2.2.3, so that the threat referenced in clause 5.2.5.5.7.2 can reflect the problems partly addressed by the corresponding requirement and test case.

In addition, the threat analysis for the requiremen and test case in clause 5.2.5.6.7.2 is still to be added.

# 4 Detailed proposal

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Start of the 1st Change \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

5.2.4.2.2.3 Threats relating to ETSI-defined interfaces

Two of the interfaces defined in ETSI NFV specification [11] are identified as the critical assets of GVNP type 1, i.e. interface between VNF and VNFM, interface between 3GPP VNF and virtualisation layer. The threats on these interfaces are as follows.

- Threats on interface between 3GPP VNF and VNFM: an attacker can compromise a VNFM to attack a 3GPP VNF. For example, the attacker illegally terminates a 3GPP VNF or tampers with VNFD of a 3GPP VNF without authorization, resulting in DoS attack or information leak against the 3GPP VNF.

- Threats on interface between 3GPP VNF and virtualisation layer: an attacker can attack a 3GPP VNF through a compromised virtualisation layer. For example, cryptographic keys or other security critical data of a 3GPP VNF could be stolen by an attacker with access to the virtualisation layer, or the virtualized resource provided by the virtualization layer to the 3GPP VNF can be manipulated or the bootloader of Guest OS of a 3GPP VNF can be tampered by an attacker via a compromised virtualisation layer.

Editor’s note: More threats described in 3GPP TR 33.848[9] or/and ETSI specification etc. are to be added if identified as related to the above two interfaces.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Start of the 2nd Change \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

5.2.5.5.7.2 Security functional requirements on execution environment provision

*Requirement Name*: secure execution environment provision

*Requirement Description*:

The VNF shall support to compare the owned resource state with the parsed resource state from VNFD (VNF Description) by the VNFM. The VNF can query the parsed resource state by the VNFM from the OAM. The VNF shall send an alarm to the OAM if the two resource states are inconsistent. This comparing process can be triggered periodically by the VNF, or the administrator can manually trigger the VNF to perform the comparing process.

NOTE: The virtualisation layer provides the execution environment for the VNF. The security of the virtualisation layer is a base of the VNF security. The operators should check whether their VNFs are run on the trusted virtualisation layer.

*Threat Reference*: Threats on interface between 3GPP VNF and virtualisation layer, in clause 5.2.4.2.2.3

*Test case*:

**Test Name:** TC\_SECURE EXECUTION ENVIRONMENT PROVISION

**Purpose:**

1. To test whether the VNF compares the owned resource state with the parsed resource state.

2. To test whether the VNF send an alarm to the OAM if the two resource states are inconsistent.

**Procedure and execution steps:**

**Pre-Condition:**

Test environment with a virtualisation layer (or simulated virtualisation layer), an OAM, a VNFM, a VIM (or simulated OAM, VNFM, VIM).

**Execution Steps**

1. The tester utilizes the virtualisation layer to change the resource state of VNF (e.g. change vCPU size of the VNF).

2. The tester uses the VNF to query the parsed resource state from the OAM.

3. The tester uses the OAM to query the parsed resource state of the VNF from the VNFM and send the received resource state to the VNF.

4. The tester checks whether the VNF sends an alarm to the OAM when the VNF receives the parsed resource state from the OAM and finds that the owned resource state and the parsed resource state are inconsistent.

**Expected Results:**

The VNF send an alarm to the OAM when the VNF receives the parsed resource state from the OAM and find that the owned resource state and the parsed resource state are inconsistent.

**Expected format of evidence:**

Screenshot contains the alarm on the OAM.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Start of the 3rd Change \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

5.2.5.6.7.2 Security functional requirements on execution environment creation

*Requirement Name*: secure execution environment creation

*Requirement Description*:

When an attacker tampers a driver provided by the hardware and used to create the execution environment, the virtualisation layer shall alert the driver error to the administrator for checking the error and finding the attack at latter.

Note: The operators should check whether the hardware is trust or not and ensure the virtualisation layer and the VNF to be run on the trusted hardware.

*Threat Reference:* TBA

*Test case*:

**Test Name:** TC\_SECURE EXECUTION ENVIRONMENT CREATION

**Purpose:**

To test the virtualisation layer alerts the driver error.

**Procedure and execution steps:**

**Pre-Condition:**

- Test environment with a VIM (or simulated virtualisation layer, a VIM) and a host.

**Execution Steps**

1. The tester tampers a driver on the server and implements the execution environment creation.

2. The tester checks whether the virtualisation layer alerts the driver error or not.

**Expected Results:**

The virtualisation layer alerts the driver error.

**Expected format of evidence:**

Screenshot contains the alert.

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