**3GPP TSG-SA3 Meeting #102Bis-e *draft\_S3-211047-r1***

**e-meeting, 1 - 5 March 2021** Revision of S3-20xxxx

**Source:** **Ericsson**

**Title:**  **New Solution to KI#5****: End-to-end integrity protection of HTTP body and method**

**Document for: Approval**

**Agenda Item: 2.20**

# 1 Decision/action requested

***It is proposed to add the following solution to the TR 33.875 [1]***

# 2 References

[1] 3GPP TR 33.875 "Study on enhanced security aspects of the 5G Service Based Architecture (SBA)"

# 3 Rationale

 The solution proposes a new way of End-to-end integrity protection of HTTP messages.

# 4 Detailed proposal

\*\*\*\*\*\*BEGIN CHANGES\*\*\*\*\*

# 2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non‑specific.

- For a specific reference, subsequent revisions do not apply.

- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[x] 3GPP TS 33.501: "Security architecture and procedures for 5G System".

[y] 3GPP TS 33.220: "Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (GBA)".

## \*\*\*\*\*\*NEXT CHANGE\*\*\*\*\*

## 6.X Solution #X: End-to-end integrity protection of HTTP body and method

### 6.X.1 Introduction

This solution addresses the key issue #5 (End-to-end integrity protection of HTTP messages).

The core steps of this solution are:

- Use Client credentials assertions (CCAs) based authentication as specified in TS 33.501 [x] Clause 13.3.8 for NF-NRF or/and NF-NF communication.

- Enhance the Client credentials assertions (CCAs) to include a hash of the HTTP body and HTTP method to protect the message itself.

- The receiving node (NRF or NF producer) computes the hash of the HTTP body and HTTP method and validates that it is identical to the hash received in the Client credentials assertions (CCAs).

Editor's Note: Backwards compatibility with Rel-16 NF producers supporting only existing CCA is ffs.

### 6.X.2 Solution details

 

 Figure 6.X.2-1 CCA based Authentication with HTTP hash enhancement

1. NF service consumer sends a service request including a signed Client credentials assertion (CCA) token to authenticate against NF service producer or NRF as described in TS 33.501 [x] Clause 13.3.8. But for this solution it is also proposed to add an optional field in CCA to protect the part of the message itself. The added field is a hash of HTTP body and HTTP method.

2. NF service producer or NRF validates the CCA as described in 3GPP 33.501 Clause 13.3.8.3. But since one optional field is supposed to be added to the CCA, the receiving end point (NF service producer or NRF) also needs to compute the hash of the HTTP body and HTTP method and validates that it is identical to the hash received in the Client credentials assertion.

The details of the hash are proposed to be specified as following:

For computation of the hash of the HTTP body and HTTP method for inclusion into the Client credential assertion, the input S to the KDF specified in Annex B of 3GPP TS 33.220 [y] is computed as follows:

 - P0 = HTTP body;

- L0 = length of the HTTP body;

- P1 = HTTP method;

- L1 = length of HTTP method.

The input key KEY is equal to null. Note that the FC value will be allocated in the normative phase.

### 6.X.3 Evaluation

This solution provides a simple proposal to add an optional field to the existing CCA token. In this way, it meets the potential security requirement that 5GS should support end-to-end integrity protection of critical elements of an HTTP message while allowing the SCP to continue to perform necessary HTTP message mediation. The solution covers most of the cases with an exception that SCP modifies the HTTP body or HTTP method. But for most of the cases SCP only modifies the HTTP header.

\*\*\*\*\*\*END OF CHANGES\*\*\*\*\*