SFC
Internet-Draft
Intended status: Standards Track
Expires: August 3, 2020

M. Boucadair
Orange
T. Reddy
McAfee
D. Wing
Citrix

January 31, 2020

Integrity Protection for Network Service Header (NSH) and Encryption of Sensitive Context Headers

draft-rebo-sfc-nsh-integrity-03

Abstract

This specification adds integrity protection and optional encryption of sensitive metadata directly to Network Service Headers (NSH) used for Service Function Chaining (SFC).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 3, 2020.

Copyright Notice

Table of Contents

1. Introduction

Many advanced Service Functions (e.g., Performance Enhancement Proxies, NATs, firewalls, etc.) are invoked for the delivery of value-added services, particularly to meet various service objectives such as IP address sharing, avoiding covert channels, detecting Denial-of-Service (DoS) attacks and protecting network infrastructures against them, network slicing, etc. Because of the proliferation of such advanced SFs together with complex service

deployment constraints that demand more agile service delivery procedures, operators need to rationalize their service delivery logics and master their complexity while optimising service activation time cycles. The overall problem space is described in [RFC7498].

[RFC7665] presents an architecture addressing the problematic aspects of existing service deployments, including topological dependence and configuration complexity. It also describes an architecture for the specification, creation, and maintenance of Service Function Chains (SFC) within a network. That is, how to define an ordered set of SFs and ordering constraints that must be applied to packets/flows selected as a result of traffic classification. [RFC8300] specifies the SFC encapsulation: Network Service Header (NSH).

NSH data is unauthenticated and unencrypted [RFC8300], forcing a service topology that requires security and privacy to use a transport encapsulation that supports such features. Note that some transport encapsulation (e.g., IPsec) only provide hop-by-hop security between two SFC data plane elements (e.g., two service function forwarders (SFFs), SFF to SF) and do not provide SF-to-SF security of NSH metadata. For example, if IPsec is used, SFFs or SFs within a Service Function Path (SFP) not authorized to access the privacy-sensitive metadata will have access to the metadata. As a reminder, the metadata referred to is an information that is inserted by intermediate SFs and which is not visible to the communication endpoints (Section 4.9 of [RFC7665]).

The lack of such capability was reported during the development of [RFC8300] and [RFC8459]. The reader may refer to Section 3.2.1 of [I-D.arkko-farrell-arch-model-t] for a discussion on the need for more awareness about attacks from within closed domains.

This specification fills that gap. Concretely, this document adds integrity protection and optional encryption of sensitive metadata directly to NSH (Section 4); integrity protects the packet payload, and provides relay protection (Section 7.4). Thus, the NSH do not have to rely upon an underlying transport encapsulation for security and confidentiality.

This specification introduces new Variable-Length Context Headers to carry fields necessary for integrity protected NSH headers and encrypted Context Headers (Section 5), and is therefore only applicable to NSH MD Type 0x02, as defined in Section 2.5 of [RFC8300].

This specification limits thus access to an information within an SFP to entities that have a need to interpret it. Typically, SFF should not act or process the context headers.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119][RFC8174] when, and only when, they appear in all capitals, as shown here.

This document makes use of the terms defined in [RFC7665] and [RFC8300].

The document defines the following terms:

- SFC data plane element: Refers to SFC-aware Service Function, Service Function Forwarder (SFF), SFC proxy, or classifier as defined in the SFC data plane architecture [RFC7665].

- SFC Control Element: A logical entity that instructs one or more SFC data plane functional elements on how to process NSH packets within an SFC-enabled domain.

- Key Identifier: A key identifier or kerberos-like object used to identify and deliver keys to authorized entities. See for example, 'kid' usage in [RFC7635].

- NSH data: The NSH is composed of a Base Header, a Service Path Header, and optional Context Headers. NSH data refers to all the above headers and the packet or frame on which NSH is imposed to realize a Service Function Path (SFP).

- NSH imposer: Refers to the SFC data plane element that is entitled to impose NSH with the Context Headers defined in this document.

3. Assumptions & Basic Requirements

The NSH format is defined in Section 2 of [RFC8300]; the NSH data can be spread over three headers:

- Base Header: Provides information about the service header and the payload protocol.

- Service Path Header: Provides path identification and location within a Service Function Path (SFP).

- Context Header(s): Carries metadata (i.e., context data) along a service path.

NSH allows to share context information (a.k.a., metadata) with upstream SFC-aware data elements on a per SFC/SFP basis. To that aim:

The control plane is used to instruct the SFC classifier about the set of context information to be supplied in the context of a given chain.

The control plane is also used to instruct an SFC-aware SF about any metadata it needs to attach to packets for a given SFC. This instruction may occur any time during the validity lifetime of an SFC/SFP. The control plane may indicate, for a given service function chain, an order for consuming a set of contexts supplied in a packet.

An SFC-aware SF can also be instructed about the behavior it should adopt after consuming a context information that was supplied in the NSH header. For example, the context can be maintained, updated, or stripped.

An SFC proxy may be instructed about the behavior it should adopt to process the context information that was supplied in the NSH header on behalf of an SFC-unaware SF, e.g., the context can be maintained or stripped.

The SFC proxy may also be instructed to add some new context information into the NSH header on behalf of an SFC-unaware SF.

In reference to Figure 1,

- Classifiers, SFC-aware SFs, and SFC proxies are entitled to update the Context Header(s).

- Only SFC-aware SFs and SFC proxies are entitled to update the Service Path Header.

- SFFs are entitled to modify the Base Path header (TTL value, for example). Nevertheless, SFFs are not supposed to act on the Context Headers or look into the content of the Context Headers.

  Discussion Note: This design decision should be discussed with the working group.

Thus, the following requirements:

- Only Classifiers, SFC-aware SFs, and SFC proxies MUST be able to encrypt and decrypt a given Context Header.

- Both encrypted and unecrypted Context Headers MAY be included in the same NSH. That is, some Context Headers (TLVs) may be protected while others do not.

- The solution MUST provide integrity protection for the Service Path Header.

- The solution MAY provide integrity protection for the Base Header. The implications of disabling such checks are discussed in Section 8.1.

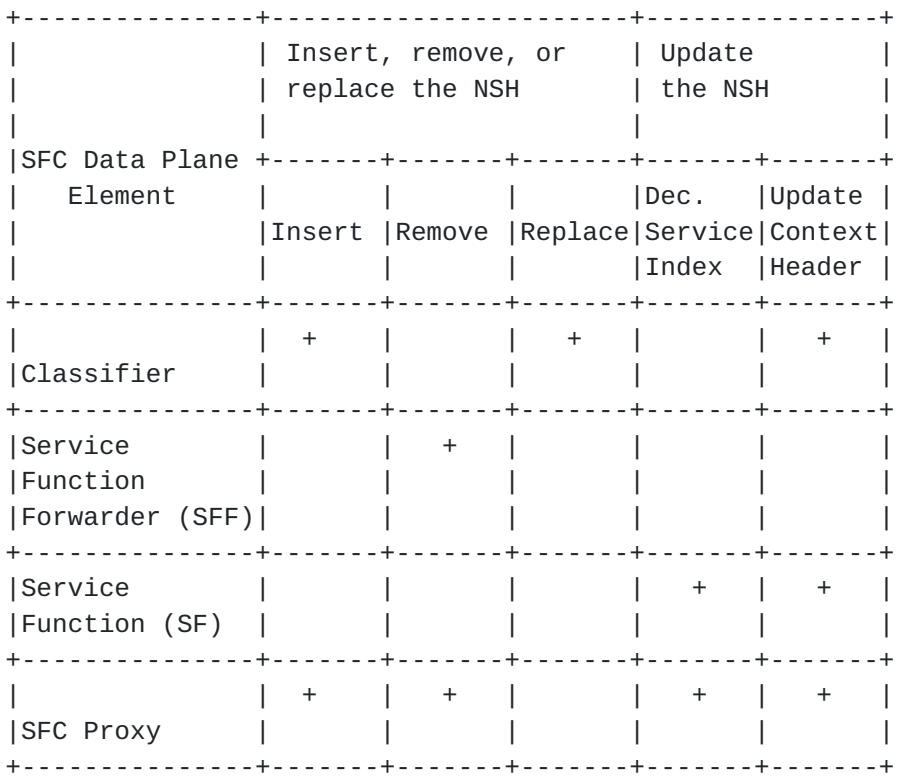

Figure 1: NSH Actions

This document does not make any assumption about the service function chains to be instantiated nor adds any constraint about how NSH can be used within a domain. For example, in reference to Figure 2, the solution accommodates deployment schemes such as: no Context Header is inserted by the Classifier, a first SF inserts two Context Headers that it encrypts, a second SF is entitled to access that encrypted data but it is instructed to strip one particular Context Header, and a third SF is instructed to strip the remaining Context Header. The following behavior will thus be observed:

- No Context Header is inserted by the Classifier: it only proceeds with the NSH integrity protection. The NSH encapsulated packet is then forwarded to the next hop following [RFC7665].

- Once the packet is received by SF1, and assuming integrity checks succeed, SF1 inserts two Context Headers M1 and M2 that it encrypts and recomputes the message integrity for the NSH data.

- Once the packet is received by SF2, and assuming integrity checks succeed, SF2 decrypts M1 and M2, strips M2 (because its instructed to do so via the SFC control plane), and then encrypts M1 and recomputes the message integrity for the NSH data.

- Once the packet is received by SF3, and assuming integrity checks succeed, SF3 decrypts M1, consumes the decrypted M1 data, and then strips it from the NSH. The packet is then forwarded back to SFF3 by SF3 after recomputing the message integrity for the NSH data.

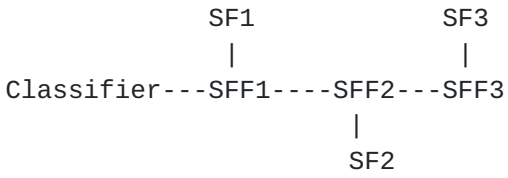

Figure 2: SFC-enabled Domain Example

4. Design Overview

4.1. Supported Security Services

This specification provides the functions described in the following sub-sections:

4.1.1. Encrypt All or a Subset of Context Headers

The solution allows to encrypt all or a subset of NSH Context Headers by Classifiers, SFC-aware SFs, and SFC proxies. As depicted in Table 1, SFFs are not involved in data encryption. This memo enforces this design approach by encrypting a Context Header with keys that are not supplied to SFFs, thus enforcing this limitation by protocol (rather than requirements language).

Data Plane Element	Base and Service Headers Encryption	Metadata Encryption
Classifier	No	Yes
SFF	No	No
SFC-aware SF	No	Yes
SFC Proxy	No	Yes
SFC-unaware SF	No	No

Table 1: Encryption Function Supported by SFC Data Plane Elements

The SFC control plane is assumed to instruct the Classifier(s), SFC-aware SFs, and SFC proxies with the set of Context Headers (privacy-sensitive metadata, typically) that must be encrypted. Encryption keying material is only provided to these SFC data elements.

The control plane may also indicate the set of SFC data plane elements that are entitled to supply a given context header (e.g., in reference to their identifiers as assigned within the SFC-enabled domain). It is out of the scope of this document to elaborate on how such instructions are provided to the appropriate SFC data plane elements, nor to detail the structure used to store the instructions.

The Service Path Header (Section 2 of [RFC8300]) is not encrypted because SFFs use Service Index (SI) in conjunction with Service Path Identifier (SPI) for determining the next SF in the path.

4.1.2. Integrity Protection

The solution provides integrity protection for NSH data. Two flavors are supported.

A first flavor where all NSH data except the Base Header are integrity protected (Figure 3). In this case, the NSH imposer may be a Classifier, an SFC-aware SF, or an SFC proxy. SFFs are not thus provided with authentication material. Further details are discussed in Section 5.1.

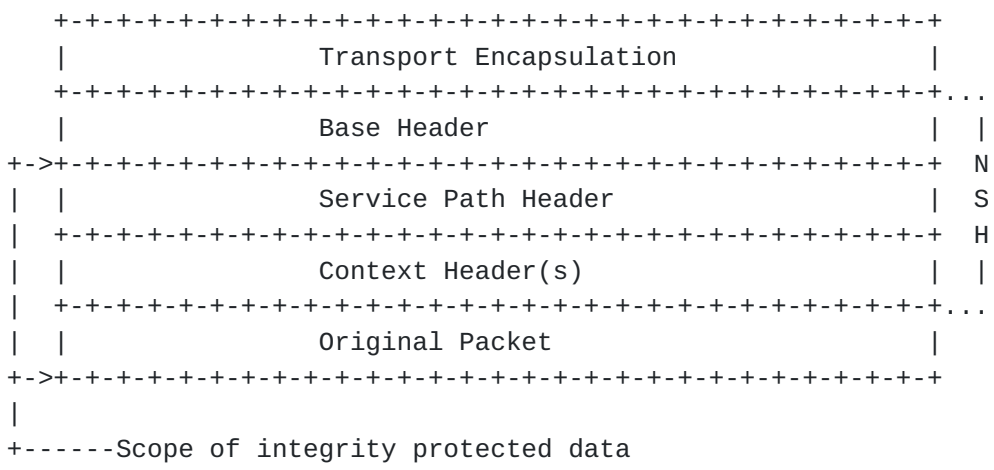

Figure 3: First Integrity Flavor

A second flavor where all NSH data, including the Base Header, are integrity protected (Figure 4). In this case, the NSH imposer may be a Classifier, an SFC-aware SF, an SFF, or an SFC proxy. Further details are provided in Section 5.2.

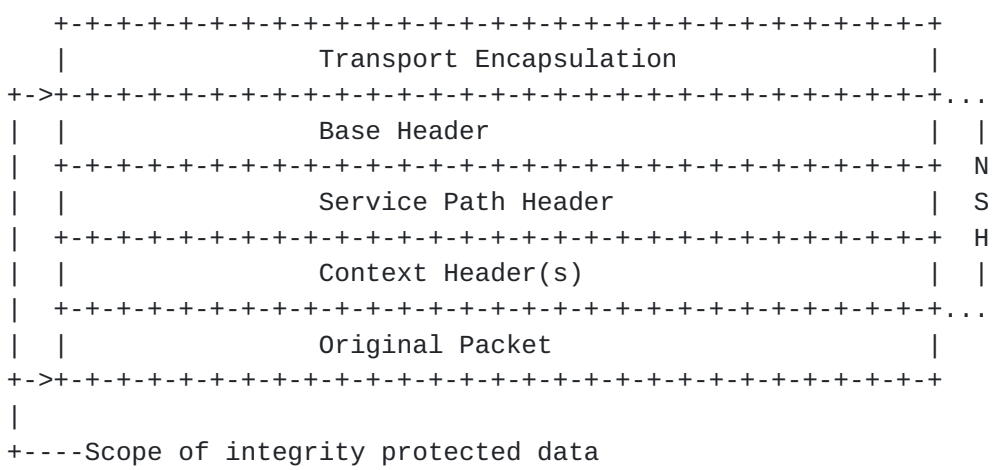

Figure 4: Second Integrity Flavor

The integrity protection scope is explicitly signaled to SFC-aware SFs and SFC proxies in the NSH by means of a dedicated MD Type (Section 5).

In both flavors, the unencrypted Context Headers and the packet on which NSH is imposed are subject to integrity protection.

Table 2 lists the roles of SFC data plane elements in providing integrity protection for the NSH.

Data Plane Element	Integrity Protection
Classifier	Yes
SFF	No (first flavor); Yes (second flavor)
SFC-aware SF	Yes
SFC Proxy	Yes
SFC-unaware SF	No

Table 2: Integrity Protection Supported by SFC Data Plane Elements

4.2. One Secret Key, Two Security Services

The authenticated encryption algorithm defined in [RFC7518] is used to provide NSH data integrity and to encrypt the Context Headers carrying privacy-sensitive metadata.

The authenticated encryption algorithm provides a unified encryption and authentication operation which turns plaintext into authenticated ciphertext and vice versa. The generation of secondary keys MAC_KEY and ENC_KEY from the secret key (K) is discussed in Section 5.2.2.1 of [RFC7518]:

- The ENC_KEY is used for encrypting the Context Headers and the message integrity of the NSH data is calculated using the MAC_KEY.

- If the Context Headers are not encrypted, the Hashed Message Authentication Mode (HMAC) algorithm discussed in [RFC4868] is used to integrity protect the NSH data.

The advantage of using the authenticated encryption algorithm is that SFC-aware SFs and SFC proxies only need to re-compute the message integrity of the NSH data after decrementing the Service Index (SI) and do not have to re-compute the ciphertext. The other advantage is in both the flavors discussed above is that SFFs do not have access to the ENC_KEY and cannot act on the encrypted Context Headers and, only in case of the second flavor, SFFs do have access to the MAC_KEY. Similarly, an SFC-aware SF or SFC proxy not allowed to decrypt the Context Headers will not have access to the ENC_KEY.

The authenticated encryption algorithm or HMAC algorithm to be used by SFC data plane elements is typically controlled using the SFC control plane. Mandatory to implement authenticated encryption and HMAC algorithms are listed in Section 4.3.

The authenticated encryption process takes as input four octet strings: a secret key (K), a plaintext (P), Additional Authenticated

Data (A) (which contains the data to be authenticated, but not encrypted), and an Initialization Vector (IV). The ciphertext value (E) and the Authentication Tag value (T) are provided as outputs.

In order to decrypt and verify, the cipher takes as input K, IV, A, T, and E. The output is either the plaintext or an error indicating that the decryption failed as described in Section 5.2.2.2 of [RFC7518].

4.3. Mandatory-to-Implement Authenticated Encryption and HMAC Algorithms

Classifiers, SFC-aware SFs, and SFC proxies MUST implement the AES_128_CBC_HMAC_SHA_256 algorithm and SHOULD implement the AES_192_CBC_HMAC_SHA_384 and AES_256_CBC_HMAC_SHA_512 algorithms.

Classifiers, SFC-aware SFs, and SFC proxies MUST implement the HMAC-SHA-256-128 algorithm and SHOULD implement the HMAC-SHA-384-192 and HMAC-SHA-512-256 algorithms.

SFFs MAY implement the aforementioned cipher suites and HMAC algorithms.

- Note: The use of AES-GCM + HMAC may have CPU and packet size implications (need for a second 128-bit authentication tag).

4.4. Key Management

The procedure for the allocation/provisioning of secret keys (K) and authenticated encryption algorithm or MAC_KEY and HMAC algorithm is outside the scope of this specification. As such, this specification does not mandate the support of any specific mechanism.

The documents does not assume nor preclude the following:

- The same keying material is used for all the service functions used within an SFC-enabled domain.

- Distinct keying material is used per SFP by all involved SFC data path elements.

- Per-tenant keys are used.

In order to accommodate deployments relying upon keying material per SFC/SFP and also the need to update keys after encrypting NSH data for certain amount of time, this document uses key identifier (kid) to unambiguously identify the appropriate keying material. Doing so allows to address the problem of synchronization of keying material.

Additional information on manual vs. automated key management and when one should be used over the other can be found in [RFC4107].

4.5. New NSH Variable-Length Context Headers

New NSH Variable-Length Context Headers are defined in Section 5 for NSH data integrity protection and, optionally, encryption of Context Headers carrying privacy-sensitive metadata. Concretely, an NSH imposer includes (1) the key identifier to identify the keying material, (2) the timestamp to protect against replay attacks (Section 7.4), and (3) the Message Authentication Code (MAC) for the target NSH data (depending on the integrity protection scope) calculated using the MAC_KEY and optionally Context Headers encrypted using ENC_KEY.

An NSH data plane element that needs to check the integrity of the NSH data uses the MAC_KEY and the HMAC algorithm for the key identifier being carried in the NSH.

An SFC-aware SF or SFC proxy that needs to decrypt some Context Headers uses ENC_Key and the decryption algorithm for the key identifier being carried in the NSH.

Section 7 specifies the detailed procedure.

4.6. Encapsulation of NSH within NSH

As discussed in [RFC8459], an SFC-enabled domain (called, upper-level domain) may be decomposed into many sub-domains (called, lower-level domains). In order to avoid maintaining state to restore back upper-lower NSH information at the boundaries of lower-level domains, two NSH levels are used: an Upper-NSH which imposed at the boundaries of the upper-level domain, and a Lower-NSH that is pushed by the Classifier of a lower-level domain in front of the original NSH (Figure 5). As such, the Upper-NSH information is carried along the lower-level chain without modification. The packet is forwarded in the top-level domain according to the Upper-NSH, while it is forwarded according to the Lower-NSH in a lower-level SFC domain.

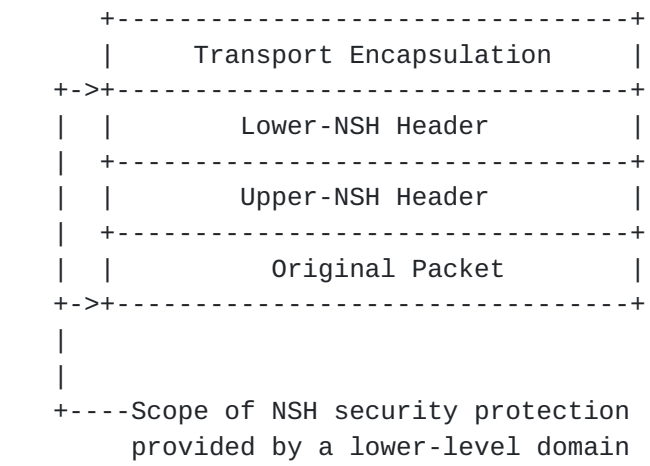

Figure 5: Encapsulation of NSH within NSH

SFC data plane elements of a lower-level domain includes the Upper-NSH when computing the MAC.

Keying material used at the upper-level domain should not the same as the one used by a lower-level domain.

5. New NSH Variable-Length Context Headers

This section specifies the format of new Variable-Length Context headers that are used for NSH integrity protection and, optionally, Context Headers encryption.

In particular, this section defines two MAC and Encrypted Metadata Context Headers; each having specific deployment constraints. Unlike the flavor discussed in Section 5.1, the flavor sketched in Section 5.2 requires sharing MAC_KEY with SFFs. Both TLVs have the same format as shown in Section 5.

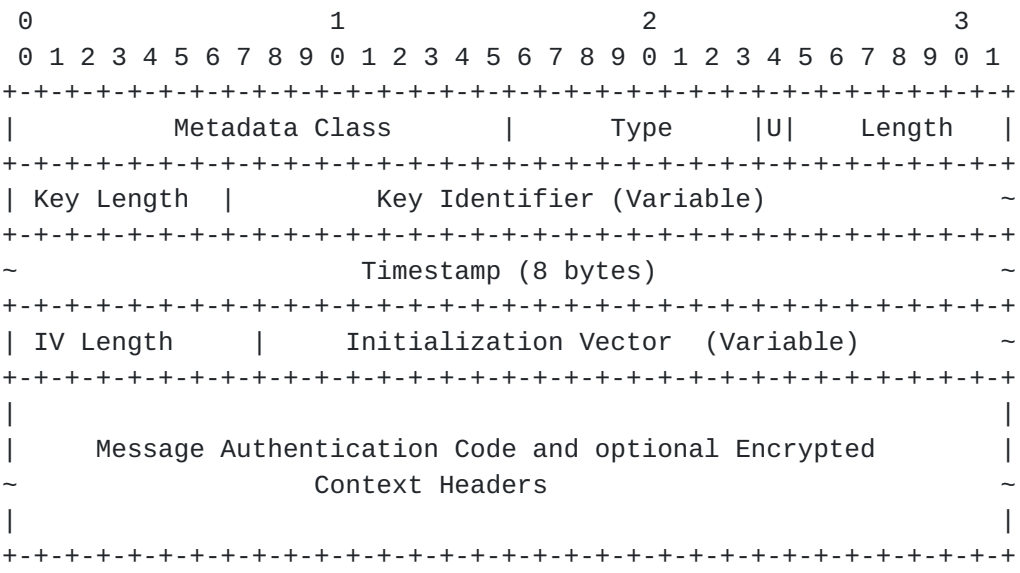

Figure 6: MAC and Encrypted Metadata Context Header

5.1. MAC#1 Context Header

MAC#1 Context Header is a variable-length TLV that carries the Message Authentication Code (MAC) for the Service Path Header, Context Headers, and the inner packet on which NSH is imposed, calculated using MAC_KEY and optionally Context Headers encrypted using ENC_KEY. The scope of this TLV is depicted in Figure 7.

This MAC flavor does not require sharing MAC_KEY with SFFs. It does not require to re-compute the MAC by each SFF because of TTL processing. Section 8.1 discusses the possible threat associated with this flavor.

Figure 7: Scope of MAC#1

In reference to Figure 6, the description of the fields is as follows:

- Metadata Class: MUST be set to 0x0 [RFC8300].
- Type: TBD1 (See Section 9)
- U: Unassigned bit [RFC8300].
- Length: Variable.
- Key Length: Variable. Carries the length of the key identifier.
- Key Identifier: Carries a variable length Key Identifier object used to identify and deliver keys to SFC data plane elements.

This identifier is helpful to accommodate deployments relying upon keying material per SFC/SFP. The key identifier helps in resolving the problem of synchronization of keying material.

- Timestamp: Carries an unsigned 64-bit integer value that is expressed in seconds relative to 1970-01-01T00:00Z in UTC time. See Section 6 for more details.

- IV Length: Carries the length of the IV (Section 5.2 of [RFC7518]). If HMAC algorithm is used, IV length is set to zero.

- Initialization Vector: Carries the IV for authenticated encryption algorithm as discussed in Section 5.2 of [RFC7518].

- The Additional Authenticated Data (defined in [RFC7518]) MUST be the Service Path header, the unencrypted Context headers, and the inner packet on which NSH is imposed .

- Message Authentication Code covering the entire NSH data excluding the Base header.

5.2. MAC#2 Context Header

MAC#2 Context Header is a variable-length TLV that carries the MAC for the entire NSH data calculated using MAC_KEY and optionally Context Headers encrypted using ENC_KEY. The scope of this TLV is depicted in Figure 8.

Figure 8: Scope of MAC#2

In reference to Figure 6, the description of the fields is as follows:

- Metadata Class: MUST be set to 0x0 [RFC8300].
- Type: TBD2 (See Section 9)
- U: Unassigned bit [RFC8300].
- Length: Variable.
- Key Length: See Section 5.1.
- Key Identifier: See Section 5.1.

- Timestamp: See Section 5.1.

- IV Length: See Section 5.1.

- Initialization Vector: See Section 5.1.

- The Additional Authenticated Data (defined in [RFC7518]) MUST be the entire NSH data (i.e., including the Base Header) excluding the Context Headers to be encrypted.

- Message Authentication Code covering the entire NSH data and optional encrypted Context Headers.

6. Timestamp Format

This section follows the template provided in [I-D.ietf-ntp-packet-timestamps].

The format of the Timestamp field introduced in Section 5 is depicted in Figure 9.

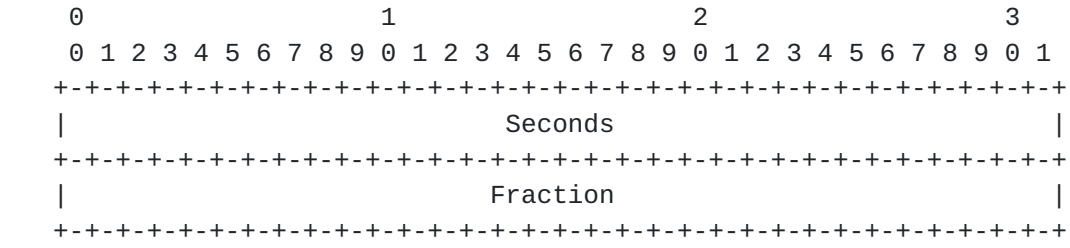

Figure 9: Timestamp Field Format

Timestamp field format:

Seconds: specifies the integer portion of the number of seconds since the epoch.

+ Size: 32 bits.

+ Units: seconds.

Fraction: specifies the fractional portion of the number of seconds since the epoch.

+ Size: 32 bits.

+ Units: the unit is 2^(-32) seconds, which is roughly equal to 233 picoseconds.

Epoch:

The epoch is 1970-01-01T00:00Z in UTC time.

Leap seconds:

This timestamp format is affected by leap seconds. The timestamp represents the number of seconds elapsed since the epoch minus the number of leap seconds.

Resolution:

The resolution is 2^(-32) seconds.

Wraparound:

This time format wraps around every 2^32 seconds, which is roughly 136 years. The next wraparound will occur in the year 2106.

Synchronization aspects:

It is assumed that SFC data plane elements are synchronized to UTC using a synchronization mechanism that is outside the scope of this document. In typical deployments SFC data plane elements use NTP [RFC5905] for synchronization. Thus, the timestamp may be derived from the NTP-synchronized clock, allowing the timestamp to be measured with respect to the clock of an NTP server. Since the NTP time format is affected by leap seconds, the current timestamp format is similarly affected. Therefore, the value of a timestamp during or slightly after a leap second may be temporarily inaccurate.

7. Processing Rules

The following sub-sections describe the processing rules for integrity protected NSH and optionally encrypted Context Headers.

7.1. Generic Behavior

This document adheres to the recommendations in [RFC8300] for handling the Context Headers at both ingress and egress SFC boundary nodes. That is, to strip such context headers.

Failures to inject or validate the Context Headers defined in this document SHOULD be logged locally while a notification alarm MAY be sent to an SFC Control Element. Similarly, failure to validate the integrity of the NSH data MUST cause that packet to be discarded while a notification alarm MAY be sent to an SFC Control Element.

The details of sending notification alarms (i.e., the parameters affecting the transmission of the notification alarms depend on the information in the context header such as frequency, thresholds, and content in the alarm) SHOULD be configurable by the SFC control plane.

SFC-aware SFs and SFC proxies MAY be instructed to strip some encrypted Context Headers from the packet or to pass the data to the next SF in the service function chain after processing the content of the Context Headers. If no instruction is provided, the default behavior for intermediary SFC-aware nodes is to maintain such Context Headers so that the information can be passed to next SFC-aware hops. SFC-aware SFs and SFC proxies must re-apply the integrity protection if any modification is made to the Context Headers (strip a Context Header, update the content of an existing Context Header, insert a new Context Header).

An SFC-aware SF or SFC proxy that is not allowed to decrypt any Context Headers MUST NOT be given access to the ENC_KEY.

Otherwise, an SFC-aware SF or SFC proxy that receives encrypted Context Headers, for which it is not allowed to consume a specific Context Header it decrypts (but consumes others), MUST keep that Context Header unaltered when forwarding the packet upstream.

Only one instance of MAC and Encrypted Metadata Context Header is allowed. If multiple instances of MAC and Encrypted Metadata Context Header are included in an NSH packet, the SFC data element must process the first instance and ignore subsequent instances, and may log or increase a counter for this event as per Section 2.5.1 of [RFC8300].

MTU and fragmentation considerations are discussed in Section 5 of [RFC8300]. Those considerations are not reiterated here.

7.2. MAC NSH Data Generation

If the Context Headers are not encrypted, the HMAC algorithm discussed in [RFC4868] is used to integrity protect the target NSH data. An NSH imposer inserts a "MAC and Encrypted Metadata" Context Header for integrity protection (Section 5).

The NSH imposer computes the message integrity for the target NSH data (depending on the integrity protection scope discussed in Section 5) using MAC_KEY and HMAC algorithm. It inserts the MAC in the "MAC and Encrypted Metadata" Context Header. The length of the MAC is decided by the HMAC algorithm adopted for the particular key identifier.

The Message Authentication Code (T) computation process can be illustrated as follows:

T = HMAC-SHA-256-128(MAC_KEY, A)

An entity in the SFP that intends to update NSH MUST follow the above behavior to maintain message integrity of the NSH for subsequent validations.

7.3. Encrypted NSH Metadata Generation

An NSH imposer can encrypt Context Headers carrying privacy-sensitive metadata, i.e., encrypted and unencrypted metadata may be carried simultaneously in the same NSH packet (Figure 10).

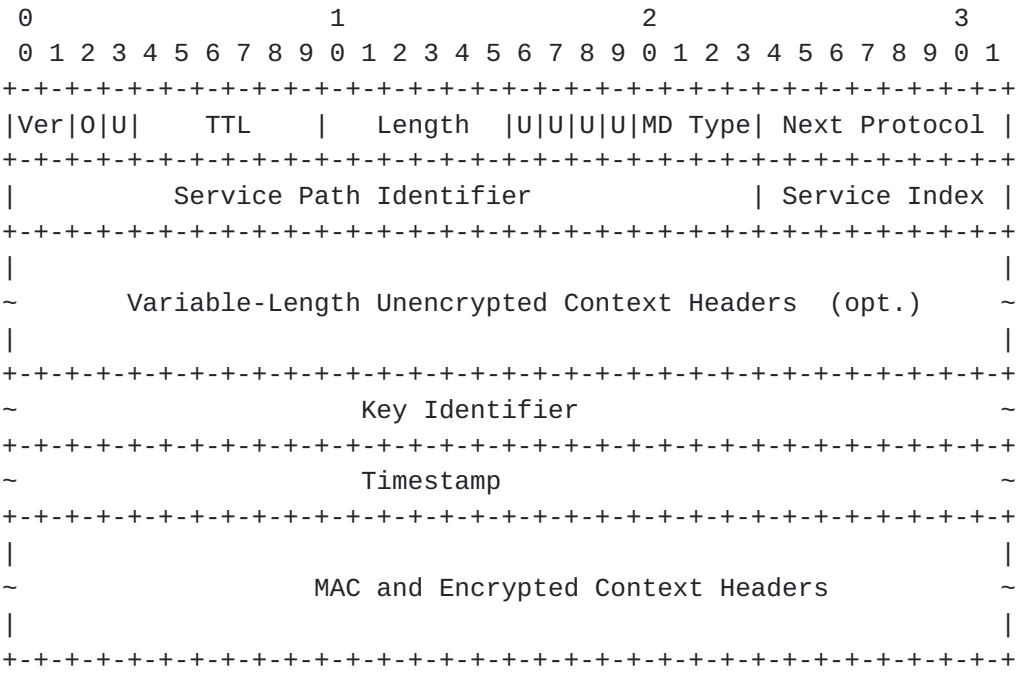

Figure 10: NSH with Encrypted and Unencrypted Metadata

In an SFC-enabled domain where pervasive monitoring [RFC7258] is possible, all Context Headers carrying privacy-sensitive metadata MUST be encrypted; doing so privacy-sensitive metadata is not revealed to attackers. Privacy specific threats are discussed in Section 5.2 of [RFC6973].

Using K and authenticated encryption algorithm, the NSH imposer encrypts the Context Headers (as set by the control plane Section 3), computes the message integrity for the target NSH data and inserts the resulting payload in the "MAC and Encrypted Metadata" Context Header (Section 5). The entire TLV carrying a privacy-sensitive

metadata is encrypted (that is, including the MD Class, Type, Length, and associated metadata of each Context Header).

The message Authentication Tag (T) and ciphertext (E) computation process can be illustrated as follows:

```
MAC_KEY = initial MAC_KEY_LEN octets of K,
ENC_KEY = final ENC_KEY_LEN octets of K,
E = CBC-PKCS7-ENC(ENC_KEY, P),
M = MAC(MAC_KEY, A || IV || E || AL),
T = initial T_LEN octets of M.
MAC and Encrypted Metadata = E || T
```

As specified in [RFC7518], the octet string (AL) is equal to the number of bits in the Additional Authenticated Data (A) expressed as a 64-bit unsigned big-endian integer.

An authorized entity in the SFP that intends to update the content of an encrypted Context Header or needs to add a new encrypted Context Header MUST also follow the aforementioned behavior.

An SFF or SFC-aware SF or SFC proxy that only has access to the MAC_KEY, but not the ENC_KEY, computes the message Authentication Tag (T) after decrementing the TTL (by the SFF) or SI (by an SF or SFC proxy) and replaces the Authentication Tag in the NSH with the computed Authentication Tag. Similarly, an SFC-aware SF (or SFC proxy) that does not modify the encrypted Context headers also follows the aforementioned behavior.

The message Authentication Tag (T) computation process can be illustrated as follows:

```
M = MAC(MAC_KEY, A || IV || E || AL),
T = initial T_LEN octets of M.
```

7.4. Timestamp for Replay Attack

The received NSH is accepted if the Timestamp (TS) in the NSH is recent enough to the reception time of the NSH (TSrt). The following formula is used for this check:

```
-Delta < (TSrt ? TS) < +Delta
```

The RECOMMENDED value for the allowed Delta is 2 seconds. If the timestamp is not within the boundaries, then the SFC data plane element receiving such packet MUST discard the NSH message.

All SFC data plane elements must be synchronized among themselves. These elements may be synchronized to a global reference time.

- TODO: timestamp validation on the receiver needs further text refinement.

7.5. NSH Data Validation

When an SFC data plane element receives an NSH packet, it MUST first ensure that a MAC Context Header is included. It MUST silently discard the message if the timestamp is invalid (described in Section 7.4). It MUST log an error at least once per the SPI for which the MAC Context Header is missing.

If the timestamp check is successfuly passed, the SFC data plane element should then proceed with NSH data integrity validation. The SFC data plane element computes the message integrity for the target NSH data (depending on the integrity protection scope discussed in Section 5) using the MAC_KEY and HMAC algorithm for the key identifier. If the value of the newly generated digest is identical to the one enclosed in NSH, the SFC data plane element is certain that the NSH data has not been tampered and validation is therefore successful. Otherwise, the NSH packet MUST be discarded.

7.6. Decryption of NSH Metadata

If entitled to consume a supplied encrypted Context Header, an SFC-aware SF or SFC proxy decrypts metadata using (K) and decryption algorithm for the key identifier in NSH.

Authenticated encryption algorithm has only a single output, either a plaintext or a special symbol (FAIL) that indicates that the inputs are not authentic (Section 5.2.2.2 of [RFC7518]).

8. Security Considerations

NSH security considerations are discussed in Section 8 of [RFC8300]. The guidelines for cryptographic key management are discussed in [RFC4107].

The interaction between the SFC-aware data plane elements and a key management system MUST NOT be transmitted in clear since this would completely destroy the security benefits of the integrity protection solution defined in this document. The secret key (K) must have an expiration time assigned as the latest point in time before which the key may be used for integrity protection of NSH data and encryption of Context Headers. Prior to the expiration of the secret key, all participating service function nodes SHOULD have the control plane

distribute an new key identifier and associated keying material, so that when the secret key is expired those nodes are prepared with the new secret key. This allows the NSH Imposer to switch to the new key identifier as soon as necessary. It is RECOMMENDED that the next key identifier be distributed by the control plane well prior to the secret key expiration time.

NSH data are exposed to several threats:

- A man-in-the-middle attacker modifying NSH data.
- Attacker spoofing NSH data.
- Attacker capturing and replaying NSH data.
- Metadata in Context Headers revealing privacy-sensitive information to attackers.
- Attacker replacing the packet on which NSH is imposed with a bogus or malicious packet.

In an SFC-enabled domain where the above attacks are possible, NSH data MUST be integrity-protected and replay-protected, and privacy-sensitive NSH metadata MUST be encrypted for confidentiality preservation purposes. The Base and Service Path headers are not encrypted.

Two MAC flavors are defined in Section 5. Considerations specific to each flavor are discussed in the following sub-sections.

The attacks discussed in [I-D.nguyen-sfc-security-architecture] are handled owing to the solution specified in this document, except for attacks dropping packets. Such attacks can be detected relying upon statistical analysis; such analysis is out of scope of this document. Also, if SFFs are not involved in the integrity checks, a misbehaving SFF which decrements SI while this should be done by an SF (SF bypass attack) will be detected by an upstream SF because the integrity check will fail.

8.1. MAC#1

An active attacker can potentially modify the Base header (e.g., decrement the TTL so the next SFF in the SFP discards the NSH packet). In the meantime, an active attacker can also drop NSH packets. As such, this attack is not considered an attack against the security mechanism specified in the document.

No device other than the SFC-aware SFs in the SFC-enabled domain should be able to update the integrity protected NSH data. Similarly, no device other than the SFC-aware SFs and SFC proxies in the SFC-enabled domain be able to decrypt and update the Context Headers carrying privacy-sensitive metadata. In other words, if the SFC-aware SFs and SFC proxies in the SFC-enabled domain are considered fully trusted to act on the NSH data, only they can have access to privacy-sensitive NSH metadata and the keying material used to integrity protect NSH data and encrypt Context Headers.

8.2. MAC#2

SFFs can detect whether an illegitimate node has altered the content of the Base header. Such messages MUST be discarded with appropriate logs and alarms generated.

9. IANA Considerations

This document requests IANA to assign the following types from the "NSH IETF-Assigned Optional Variable-Length Metadata Types" (0x0000 IETF Base NSH MD Class) registry available at: https://www.iana.org/assignments/nsh/nsh.xhtml#optional-variable-length-metadata-types.

Value	Description	Reference
TBD1	MAC and Encrypted Metadata#1	[ThisDocument]
TBD2	MAC and Encrypted Metadata#2	[ThisDocument]

10. Acknowledgements

This document was edited as a follow up to the discussion in IETF#104: https://datatracker.ietf.org/meeting/104/materials/slides-104-sfc-sfc-chair-slides-01 (slide 7).

Thanks to Joel Halpern, Christian Jacquenet, Dirk von Hugo, Tal Mizrahi, and Daniel Migault for the comments.

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4107] Bellovin, S. and R. Housley, "Guidelines for Cryptographic Key Management", BCP 107, RFC 4107, DOI 10.17487/RFC4107, June 2005, <https://www.rfc-editor.org/info/rfc4107>.

[RFC4868] Kelly, S. and S. Frankel, "Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec", RFC 4868, DOI 10.17487/RFC4868, May 2007, <https://www.rfc-editor.org/info/rfc4868>.

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <https://www.rfc-editor.org/info/rfc7518>.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <https://www.rfc-editor.org/info/rfc7665>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <https://www.rfc-editor.org/info/rfc8300>.

11.2. Informative References

[I-D.arkko-farrell-arch-model-t]
Arkko, J. and S. Farrell, "Challenges and Changes in the Internet Threat Model", draft-arkko-farrell-arch-model-t-01 (work in progress), December 2019.

[I-D.ietf-ntp-packet-timestamps]
Mizrahi, T., Fabini, J., and A. Morton, "Guidelines for Defining Packet Timestamps", draft-ietf-ntp-packet-timestamps-07 (work in progress), August 2019.

[I-D.nguyen-sfc-security-architecture]
Nguyen, T. and M. Park, "A Security Architecture Against Service Function Chaining Threats", draft-nguyen-sfc-security-architecture-00 (work in progress), November 2019.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <https://www.rfc-editor.org/info/rfc5905>.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <https://www.rfc-editor.org/info/rfc6973>.

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <https://www.rfc-editor.org/info/rfc7258>.

[RFC7498] Quinn, P., Ed. and T. Nadeau, Ed., "Problem Statement for Service Function Chaining", RFC 7498, DOI 10.17487/RFC7498, April 2015, <https://www.rfc-editor.org/info/rfc7498>.

[RFC7635] Reddy, T., Patil, P., Ravindranath, R., and J. Uberti, "Session Traversal Utilities for NAT (STUN) Extension for Third-Party Authorization", RFC 7635, DOI 10.17487/RFC7635, August 2015, <https://www.rfc-editor.org/info/rfc7635>.

[RFC8459] Dolson, D., Homma, S., Lopez, D., and M. Boucadair, "Hierarchical Service Function Chaining (hSFC)", RFC 8459, DOI 10.17487/RFC8459, September 2018, <https://www.rfc-editor.org/info/rfc8459>.

Authors' Addresses

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Tirumaleswar Reddy
McAfee, Inc.
Embassy Golf Link Business Park
Bangalore, Karnataka 560071
India

Email: TirumaleswarReddy_Konda@McAfee.com

Dan Wing
Citrix Systems, Inc.
USA

Email: dwing-ietf@fuggles.com