Service Function Chaining
Internet-Draft
Intended status: Informational
Expires: March 27, 2016

E. Wang
K. Leung
J. Felix
J. Iyer
Cisco Systems Inc.
September 24, 2015

Service Function Chaining Use Cases for Network Security
draft-wang-sfc-ns-use-cases-00

Abstract

Enterprise networks deploy a variety of security devices to protect the network, hosts and endpoints. Network security devices, both hardware and virtual, operate at all OSI layers with scanning and analysis capabilities for application content. Multiple specific devices are often deployed together for breadth and depth of defense. This document describes use cases of Service Function Chaining (SFC) when deploying network security devices in the manner described above and also puts forth requirements for their effective operation.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Network security service nodes participate in Service Function Chaining (SFC) to provide comprehensive solutions for securing campus and data center enterprise networks. Often, network operators deploy various types and instances of security service nodes. These nodes are complementary to one another for the purpose of coverage, depth of defense, scalability and availability.

In addition to packet forwarding, network security devices can buffer, inject or block certain packets, as well as proxy entire connections. Most of the network security devices maintain state at the connection, session or transaction levels. When used in a SFC environment these security Service Function actions and properties require careful design and extension including the Service Classifier and Service Function itself. This document attempts to describe the detailed use cases that lead to the requirements to support network security functions in SFC.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Definition Of Terms

This document uses the terms as defined in RFC 7498 [RFC7498], [I-D.ietf-sfc-architecture] and [I-D.ietf-sfc-nsh].

In addition the following terms are defined.

Security Service Function (Security SF): A Security Service Function is a Service Function that carries out specific security tasks. We limit the scope of security functions to network security in this document (as opposed to functions such as endpoint security). In addition to the general forwarding action, a Security Service Function can buffer, proxy, inject or block certain packets based on its policy. A Security Service Function can maintain state at the connection, session or transaction levels. Sample Security Service Functions are: Firewall, Intrusion Prevention/Detection System (IPS/IDS), Deep Packet Inspection (DPI), Application Visibility and Control (AVC), network virus and malware scanning, sandbox, Data Loss Prevention (DLP), Distributed Denial of Service (DDoS) mitigation and TLS proxy.

Flow: A flow is a uni-directional traffic stream identified by network layer attributes, specifically IP addresses and TCP/UDP ports for TCP/UDP traffic.

Connection: A connection is a bi-directional traffic stream composed of two flows sharing the same network layer attributes.

3. Characteristics of Security Service Functions

Most Security Service Functions are stateful. They maintain state at the connection, session or transaction levels, depending on the OSI layers that they act on. Many Security Functions require seeing both directions of the client-server traffic in order to maintain state properly. Asymmetric traffic must be normalized before packets reach the Security Functions.

Security Service Functions operate on network layer data with specific behaviors. For example:

1. A Firewall tracks TCP state between the TCP client and server. TCP packets that do not correspond to the Firewall’s maintained state are likely to be dropped.

2. A Firewall can modify the L3/L4 headers for NAT translation. The flow attributes in the packet header may be changed after the packet egresses the Firewall.

3. A Firewall can proxy a TCP connection by sending a TCP ACK on behalf of the endpoint. From the SFC perspective, this results in Service Function generated packets being injected into the service path in the reverse direction.

4. A Firewall or DDoS mitigator can inject TCP layer challenges to the originating client before the intended server receives a packet from the client.

Security Functions also handle packets and examine data at higher OSI layers. For example:

1. A Firewall can inspect the HTTP header and body data. Based on the inspection results, the firewall can decide to drop the packet and/or block the connection completely.

2. A Web proxy can inject an HTTP challenge page into an HTTP transaction for the purposes of authentication and identity collection.

3. At the enterprise edge, a TLS proxy, when authorized, operates as a trusted Man-in-the-Middle to proxy the TLS handshake and decrypt the packet data. The TCP payload may be completely different between ingress and egress of TLS Proxy.

4. A stream scanning service examines a certain set of application data. File scanning engines examine file streams of specific types.

4. Use Cases

4.1. Service Classification Use Cases

4.1.1. Service classification for bi-directional traffic

Many Security Service Functions require receiving bi-directional traffic of a connection. For example, a DDoS mitigator requires to see the return traffic to maintain proper state.

Return traffic (i.e. server to client response) should be classified based on the forward traffic (i.e. the client to server request). This allows server's return traffic to be associated with the clients forward traffic. The forward and return traffic forms a single bi-directional connection and shares Service Function Paths with similar set of Service Functions.

In the figure below, the Service Classifier handling traffic from Host B must be able to identify return traffic (flow 2) and select the Service Function Path with "DDoS". Flow 1 and 2 form a connection and traverse DDoS in both directions.

```
                     ,--------------.
                    /                \
                   /      .--.        \
                  /      /    \        \
                 /      ( DDoS )        \
    ------------'      / \    / \        '--(1) Forward Flow (A=>B)-->
                      /   '--'   \
                     /            \
   +---+        .--./              '-------.                      +---+
   |   |       /    \             /         \                     |   |
   | A +------(  SC  )-----------(  Firewall )--------------------+ B |
   |   |       \    /             \         /                     |   |
   +---+        '--'               '-------'                      +---+

                         (a) Flows from Host A

                      ,------------.
                     /              \
                    /     .--.       \
                   /     /    \       \
                  /     ( DDoS )       \
    <------------'     / \    / \       '---(2) Return Flow (A<=B)----
                      /   '--'   \
                     /            \
   +---+            /              '-------.         .--.         +---+
   |   |           /              /         \       /    \        |   |
   | A +----------+--------------(  Firewall )------(  SC  )------+ B |
   |   |                          \         /       \    /        |   |
   +---+                           '-------'         '--'         +---+

    <-------------(3) Forward Flow (A<=B)----------------------------

                         (b) Flows from Host B
```

Figure 1: Forward and return flows between two hosts

4.1.2. Service Classifier to distinguish initiator and responder

Even if a Security Service Function requires receiving bi-directional traffic of a connection, it should not necessarily receive traffic initiated from all network segments for performance, availability, and scalability reasons. For instance, a DDoS mitigator is configured to receive bi-directional traffic initiated from the Internet, but skip traffic initiated from the internal network.

Traffic initiated from a network segment should be classified independently. In Figure 1(b), the Service Classifier for Host B must identify traffic initiated by Host B (flow 3) and classify it independently. Such traffic bypasses the DDoS Service Function in this example.

The Service Classifier must distinguish between flow 2 and flow 3, both of which are from Host B to Host A. In other words, it must be able to identify the initiator and responder of a connection.

A Service Classifier that keeps certain state would be able to handle the above requirements with ease. The state should be accessible by each Service Classifier if there are multiple instances handling traffic sources from various network segments.

4.1.3. Service Classification based on network and application criteria

The Service Classifier evaluates SFC Policies (i.e. Service Policies) in order to determine the traffic and associated Service Function Paths. In the case of Security Service Functions, the Service Policies can contain match criteria derived from all OSI layers of the packet.

SFC classification is often based on network data, including but not limited to: Network interface port, VLAN, source and destination IP addresses, source and destination TCP and UDP ports, IP protocol, etc. These properties can be derived from the packet headers and are consistent across every packet of a flow.

There are match criteria that are desired by Security Service Functions that are either not present in the first packet, or are not present in every packet.

Those criteria may comprise "application data" from above the network layer, referred to as "application criteria". For example, a policy rule may state:

> for all TLS traffic, run the traffic through Service Function "TLS Proxy"

Another example of an application layer policy rule is:

> for all HTTP traffic with content containing file types of interest, run the traffic through Service Function "File Stream Scanner"

The Service Classifier for Security Service Functions needs to handle complex Service Policy. In some cases, this can be achieved by embedding the Service Classifier function into a Security Service Function, such that it can evaluate the application data as it becomes available.

4.1.4. Switching Service Function Paths based on inspection and scanning results

Network data is likely to be available on the first packet of the flow. When only network data is used as Service Policy match criteria, a stateful Service Classifier will be able to determine the forward and reverse Service Function Paths from the first packet (initial classification). The forward and reverse Service Function Paths remain unchanged for the entire life of the flow for these types of policies.

When the Service Policy contains application criteria, the policy rule may not be fully evaluated until several packets have passed through the chain. For example, TLS traffic can be identified only after the TLS Client Hello handshake message is observed.

Multiple classifiers may be required to provide sufficient classification granularity and complete a full evaluation of the Service Policy. In many cases, classification will be co-located with a Security Service Function that has the ability to inspect and scan the application data.

A new Service Function Path may be selected by a non-initial classification, different from the one determined by the initial classification.

The selection of a new Service Function Path can be reflected in the NSH Service Path Header as a new Service Path ID for the Service Function Forwarder to direct the packet accordingly.

The decision of a new Service Function Path often needs to be stored in Service Function and/or Service Classifier to ensure that subsequent packets of the flow follow the new path. This is because the data that triggers a new Service Function Path may be available from one particular packet only. For example, the packet with the TLS Client Hello message is used to identify a TLS session. Subsequent packets may not contain information for identifying the TLS sessions. All subsequent packets, without being classified again, must travel through the path with the "TLS Proxy" Service Function.

```
                       ,-----.
                      /       \
                     /  .---.  \
                    /  /     \  \
                   /  (  TLS  )  \
       -----------'   ( Proxy )   `-SFP-2. AVC:TLS Proxy:Firewall->
                      /\     / \
                     /  `---'   \                   .--.
      +---+      .-./            .`------.        (    )-.      +---+
      |   |     /   \           /         \     .'        )     |   |
      | A +----( AVC )---------(  Firewall )---(  Internet )----+ B |
      |   |     \   /           \         /     (        -'     |   |
      +---+      `-'             `-------'       '-(     )      +---+
                                                    '---'
       -------------SFP-1. AVC:Firewall ------------------------->
```

Figure 2: Mid-stream service function path update

Figure 2 illustrates a simple set of Security Functions deployed at the Internet edge. The default Service Function Path is SFP-1, with Service Functions "AVC" and "Firewall". When a TLS session is detected (e.g. by detecting the TLS Client Hello in the AVC Service Function), packets of the flow from that point on are switched to SFP-2, which contains "TLS Proxy" between "AVC" and "Firewall" to decrypt the TLS traffic for inspection.

Packets	Service Function Path
TCP Handshake	SFP-1. AVC:Firewall
TLS Client Hello	SFP-1; Switched to SFP-2 after AVC
Rest of TLS HS	SFP-2. AVC:TLS Proxy:Firewall
HTTPS Data	SFP-2. AVC:TLS Proxy:Firewall

Table 1: SFP taken by each packet in an HTTPS connection

Table 1 lists the Service Function Path for each packet in an HTTPS connection, from the TCP 3-way handshake to the HTTPS data packets. A new Service Function Path is selected in the middle of the connection after the TLS Client Hello is observed.

4.2. Service Function Use Cases

4.2.1. Service Classifier-capable Service Function

Service Functions that are capable of selecting a new Service Function Path must have the Service Classifier function integrated. Such Service Functions are often responsible for classification using their inspection and scanning results and updating Service Function Paths based on the Service Policy.

4.2.2. Service Functions operating on L5 or L7 data

Certain Security Service Functions operate on L5 to L7 data. For example, a "TLS Proxy" consumes a TCP stream without retransmitted or overlapping TCP segments. A "Web Proxy" operates on TCP stream of HTTP traffic. The data consumed by such Service Functions may not be in the original packet frame format, and the data may not contain the original L2-L4 header information. Such Service Functions can obtain the session or flow information from the SFC metadata carried in NSH.

4.2.3. Service Function mid-stream pick-up

When a new Service Function Path is selected as a result of Service Policy re-evaluation with application layer policy metadata, a new Service Function may need to start handling packet frames in the middle of a flow. This is referred to as "mid-stream pick-up". Although this is mid-stream from a flow perspective, it is still a complete data stream from the Service Function perspective (e.g., although "TLS Proxy" Service Function may not see the prior TCP handshake packets, it still sees the entire TLS stream). Similarly, transaction based Service Functions only handle packets belonging to a particular transaction. Such Service Function may use the flow ID metadata carried in NSH to link the session back to the flow.

Packet	AVC	TLS Proxy	Firewall
TCP SYN	X		X
TCP SYN/ACK	X		X
TCP ACK	X		X
TLS Client Hello	X	X	X
Rest of TLS HS	X	X	X
HTTPS Data	X	X	X

Table 2: Service Functions visited by each packet in an HTTPS connection

Table 2 lists the Service Functions visited by each packet from an HTTPS connection. The first packet that the Service Function "TLS Proxy" receives is the TLS Client Hello, as opposed to the TCP handshake packets prior to it.

4.2.4. Bypassing for a particular Service Function

Certain Security Service Functions can be compute-intensive while only serving a particular task. It may be required to bypass such a Service Function in the middle of a flow. For example:

- o "Firewall" may request offloading of certain flows to fast forwarding engine with minimal inspection

- o "HTTP Inspector" may decide to not inspect video streams from a site with a high reputation

- o "TLS Proxy" may have to avoid decryption of banking traffic for compliance reasons

The decision to bypass a Service Function is made by the Service Function with its static policy, the inspection results and/or mid-stream evaluation of Service Policy.

Even if a flow is offloaded or bypassed, the Security Service Function may want to continue receiving critical packets for state tracking purposes. For example, "Firewall" may want to receive TCP control packets, and "HTTP Inspector" may want to track each transaction in the same flow.

```
                          .-------.
                         /         \
                        (   HTTP    )
                        ( Inspector )
                    ,---------------------------.
                   /      `---+---'              \
                  /           |                   \
                  |           |                   |
                  |       .---+---.               |
                  |      /         \              |
                  |     (  Firewall )             |
                  |   ,---------------.           |
                  |  /    `---+---'    \          |
                  | /         |         \         |
                  | |         |         |         |
                  | |     .---+---.     |TCP      |HTTP
                  | |    /         \    |control  |Headers
                  | |   ( Offloading)   |packets  |
                  | |   (    Node   )   |         |
                  | |  ,-------------.  |         |
                  | | /   `-------'   \ |         |
                  | | |               | |         |
                  | | |Offloaded      | |         |
                  | | |packets        | |         |
                  | | |               | |         |
                  v v v               v v         v
```

Figure 3: Service function bypass examples

The offloading node can be either the Service Function Forwarder or a capable Service Function with a built-in stateful offloading path (Figure 6). The offloading path tracks the flow state and identifies critical packets to be sent to the bypassed Service Function.

```
                                         Offloading Node
                                        .--------------.
                                       /                \
                                      /        SFF       \
                                     (                    )
                                     (   ++++++++++++++   )
                                     (   + Offloading +   )
                                      \.'+    Path    +  /
                 ++++++++++++++      .'\ ++++++++++++++ /
                 + /-------\  +    .'   '--------------'
                 + | Flow  |  +  .'
                 + | State |  +.'
                 + \-------/  +
                 +            +.
                 + Offloading + '.       Offloading Node
                 +    Path    +   '.    .--------------.
                 ++++++++++++++     '. /                \
                                      /.       SF        \
                                     (   '.               )
                                     (   ++++++++++++++   )
                                     (   + Offloading +   )
                                      \  +    Path    +  /
                                       \ ++++++++++++++ /
                                        '--------------'
```

Figure 4: Service function offloading node

To steer traffic to the path that avoids the bypassed Service Function, a Service Function may update the SFC metadata in the packet if the Service Function has knowledge of the relevant Service Function Paths. Alternatively, a Service Function may signal the Service Classifier to update the Service Function Path to exclude the Service Function. Service Function Path updates may be accomplished by selecting a new path (i.e. a new Service Path ID) with the Service Function excluded.

Service Function bypass may also follow the procedure described in "Service Function Simple Offloads" [I-D.kumar-sfc-offloads], where the Service Function signals the Service Function Forwarder to offload a flow. The Service Function Forwarder caches the offload request and bypasses the Service Function in the service path for the remainder of the flow.

4.2.5. Tap mode Service Functions

Certain Service Functions such as an IDS may operate in "tap" mode, i.e. they consume a packet instead of passing the packet through.

The Service Function Forwarder should send copies of packets to tap mode Service Functions.

```
                               .---.
                              /     \
                             ( [IDS] )
                              \     /
                               '-+-'
                         .---.  /|\      .-------.
                        /     \  |      /         \
                       (  DDoS ) |     (  Firewall )
                        \     /  |      \         /
                         '-+-'   |copy   '---+---'
                          /|\    |          /|\
                           |     |           |
              +---+       \|/    |          \|/         +---+
              |   |     ,--+-----+-----------+----.     |   |
              | C +----(             SFF           )----+ S |
              |   |     '-------------------------'     |   |
              +---+                                     +---+

              [ ] denotes a packet sink
```

Figure 5: Tap mode service functions in SFC

Figure 3 illustrates an example of tap mode Service Function and their insertion into a Service Function Chain. The IDS Service Function receives copies of packets from the Service Function Forwarder.

4.3. Service Data Handling Use Cases

4.3.1. Dropping packets and closing flows

A Security Service Function may decide to drop the current packet or close a particular flow based on its inspection and scanning results, and the associated security policy.

A Service Function may drop packets without forwarding them out, or it may forward and mark such packets to be dropped by the Service Function Forwarder, referencing the flow by its flow ID in the SFC metadata.

A flow-close action usually needs to be taken by multiple stateful Service Functions, as well as the Service Function Forwarder and the Service Classifier, in order to clear their state for such a flow. Any subsequent packets of the closed flow are denied.

```
                           .----.       .----.       .----.
                          /      \     /      \     /      \
                         (  SF-1  )   (  SF-2  )   (  SF-3  )
                          \      /     \      /     \      /
                           `+--+'       `+--+'       `----'
                           /|\ |        /|\ |
                            |  |         |  |Block
                            | (P)        |  |Flow(P)
                            |  |         |  |
                            |  |         |  |
                           (P) |        (P) |
                            |  |         |  |
                            | \|/        | \|/
                 .--.    ,--+--+---------+- : ------------.    .--.
                /    \  /                                  \  /    \
      --(P)--->(      )(               State Update         )(      )
               (  SC  )(                                    )(  SC  )
      --(Q)-X->(      )(     SFF                            )(      )
                \    /  \                                  /  \    /
                 `--'    `--------------------------------'    `--'
```

Figure 6: Flow close action example

Figure 4 shows an example of closing a flow after SF-2 processes packet P. The flow close indication can be included in the packet or message returned from SF-2 to the Service Function Forwarder. The flow state update may be distributed to the Service Function Forwarder, Service Classifier and other Service Functions. The distribution mechanism is outside the scope of this document.

4.3.2. Service Function injected new packet

Security Service Functions may inject new packets into an existing flow in either direction. For example,

- "Web Proxy" inserts an HTTP page challenging the client to login, in order to obtain the client’s identity. This is in response to a packet (likely HTTP Request) but in the opposite direction of the flow.

- "Firewall" checks an idle TCP connection by sending TCP keepalives to the client and/or server (known as "TCP dead connection detection"). This is on existing flows but not responding to a prior packet.

- "Firewall" sends ICMP error message after dropping a packet. This is in response to the prior packet but on a new flow.

The Service Function or Service Classifier needs to conduct a lookup of the reverse Service Function Path and populate the NSH Service Path Header. The approaches described in [I-D.penno-sfc-packet] may be adopted to support this use case.

4.3.3. Service Function initiated connections

A Service Function may need to create its own connections that are not associated with any client connection. Use cases include probing of servers behind a web proxy. In such cases, there will be no existing metadata for the Service Function to use to establish this connection. Such connections should be classified just like any other connections traversing the Service Function Path, as there may be Service Functions that are required to perform operations such an NAT on such connections in order for it to reach its destination.

```
             <------ SFC-1. Firewall:LB:IPS ------>

       +---+     .-------.       .--.       .-.     +---+
       |   |    /         \     /    \     /   \    |   |
       | C +---(  Firewall )---(  LB  )---( IPS )---+ S |
       |   |    \         /     \    /     \   /    |   |
       +---+     `-------'       `--'       `-'     +---+

                                   <-- SFC-2. LB:IPS -->
```

Figure 7: SFC for service function initiated connection

A Service Classifier-capable Service Function may conduct service classification to determine the Service Function Path for the Service Function initiated connection. It can add an NSH with the proper Service Path Headers to the packets, and the Service Function would be the first SF on the chain. Response traffic follows a reverse Service Function Path and terminates at the Service Function. The number of Service Path Identifiers increases with more Service Functions bearing such capability.

A Service Function may send native packets without NSH when it is not capable of service classification. Such traffic is handled by the Service Classifier, which will populate the traffic with the appropriate NSH.

4.3.4. Security classification results

Security Service Functions may generate security classification results (e.g. policy actions and inspection results) while processing the packet data. Certain actions such as packet drop and flow closure can be taken immediately.

However, Service Functions can choose not to take any action immediately. Instead, it may pass the classification results to the subsequent Service Functions or to a control point.

Security classification results may be carried in NSH metadata as a score value. The score can be relayed and refined by other Security Service Functions along the path. Figure 8 below depicts an example of accumulating the client’s score based on the Service Function’s classfication result. The client’s reputation score is 6 as reported by the Service Function "Reputation", and the score is then passed to the next Service Function "Web Proxy" as the initial score for the connection. "Web Proxy" reduces the score to 3 after detecting access to a low reputation website. The Service Function "File Scanner" is involved due to the low score so far. After the "File Scanner" conducts scanning on the downloaded file and identifies it to be a malware, it updates the score to be -5 which is below the threshold for the connection to be blocked.

```
                                                 .------.
       +---+     .--------.      .--------.     /        \    +---+
       |   |    /          \    /          \   (   File   )   |   |
       | C +---( Reputation )--(  Web Proxy )--(  Scanner )---+ S |
       |   |    \          /    \          /    \        /    |   |
       +---+     '--------'      '--------'      '------'     +---+
                      |              |              |
                      V              V              V
      +----------------------+--------------+-------------+
      |        | Client       | Accessing     | Malware      |
      | Reason | Reputation   | website rep   | found in     |
      |        | score 6      | score 2       | download     |
      +--------+------+------+-------+-------+-------+-----+
                      |              |              |
                  +---+---+          |              |
                  |   6   |          V              |
                  +---+---+          |              |
                      '------>-------+              V
                                     |              |
                                 +---+---+          |
                                 |   3   |          |
                                 +---+---+          |
                                     '------>-------+
                                                    |
                                                +---+---+
                                                |  -5   |
                                                +---+---+
                                                    |       /-------\
                                                    |       |       |
                                                    '------>| Block |
                                                            |       |
                                                            \-------/
```

Figure 8: Security classification result with accumulated client score

Alternatively, each participating Service Function may send its own classification result to a central Service Function or control point for aggregation. Actions are then taken by a specific Service Function or control point based on the accumulated results. Figure 9 illustrates this option.

```
                                              .------.
    +---+     .--------.      .--------.     /        \    +---+
    |   |    /          \    /          \   (   File   )   |   |
    | C +---( Reputation )--(  Web Proxy )--(  Scanner )---+ S |
    |   |    \          /    \          /    \        /    |   |
    +---+     '--------'      '--------'      '------'     +---+
                  |                |              |
                  V                V              V
    +--------+-------------+--------------+------------+
    |   SF   | Client      | Accessing    | Malware    |
    |Classify| reputation  | website rep  | found in   |
    | Results| score 6     | score 2      | download   |
    +--------+------+------+-------+------+-----+------+
                    |              |            |
                    |              |            |
                    |              |            |
                    |              |            |  +------+  /-------\
                    |              |            '->|      |  |       |
                    |              '-------------->| Eval +->+ Block |
                    '----------------------------->|      |  |       |
                                                   +------+  \-------/
```

Figure 9: Aggregation of security classification results

5. General Requirements

The above use cases lead to the following requirements for applying SFC to security traffic.

1. SFC MUST support the use of stateful Service Classifiers and Service Functions if present.

2. Service Classifiers MUST have the ability to classify forward and the corresponding reverse Service Function Paths.

3. SFC MUST support the use of Service Policies with network and application layer match criteria if supported by Service Classifier.

4. SFC MUST support Service Function Path update or selection of a new path by a Service Classifier in the middle of a flow.

5. SFC SHOULD allow packet frames carrying only L5 and upper layer traffic data without L2-L4 headers.

6. SFC MUST allow tap mode Service Functions.

7. SFC policies MUST support tap mode Service Functions.

8. SFC MUST support packet injection to the opposite direction of a Service Function Path.

9. SFC SHOULD support bypass of a Service Function in the middle of a connection while allowing necessary control packets to reach the Service Function.

6. Security Considerations

This document describes use cases for Security Service Functions to participate in SFC. There are cases such as picking up traffic from the middle of a packet stream or handling packets without L2-L4 headers. Security Service Functions must process those types of traffic properly and associate them with the appropriate internal state.

While each Security Service Function applies its own implementation to secure the internal data, communications between Service Functions need to be secured as well. Measures must be taken to ensure metadata such as security classifications carried in NSH is not tampered.

7. Acknowledgments

The authors would like to thank Paul Quinn, Reinaldo Penno and Jim Guichard for their detailed review, comments and contributions.

8. IANA Considerations

This document includes no request to IANA.

9. References

9.1. Normative References

[I-D.ietf-sfc-architecture]
Halpern, J. and C. Pignataro, "Service Function Chaining (SFC) Architecture", draft-ietf-sfc-architecture-08 (work in progress), May 2015.

[I-D.ietf-sfc-nsh]
Quinn, P. and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-00 (work in progress), March 2015.

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

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

9.2. Informative References

[I-D.kumar-sfc-offloads]
Surendra, S., Guichard, J., Quinn, P., and J. Halpern, "Service Function Simple Offloads", draft-kumar-sfc-offloads-01 (work in progress), September 2015.

[I-D.penno-sfc-packet]
Penno, R., Pignataro, C., Yen, C., Wang, E., and K. Leung, "Packet Generation in Service Function Chains", draft-penno-sfc-packet-00 (work in progress), September 2015.

Authors' Addresses

Eric Wang
Cisco Systems Inc.
170 W Tasman Dr
San Jose, CA 95134
U.S.A.

Email: ejwang@cisco.com

Kent Leung
Cisco Systems Inc.
170 W Tasman Dr
San Jose, CA 95134
U.S.A.

Email: kleung@cisco.com

Jeremy Felix
Cisco Systems Inc.
170 W Tasman Dr
San Jose, CA 95134
U.S.A.

Email: jefelix@cisco.com

Jay Iyer
Cisco Systems Inc.
170 W Tasman Dr
San Jose, CA 95134
U.S.A.

Email: jiyer@cisco.com