Secure Service Function Chaining in the Context of Zero Trust Security

Leonard Bradatsch[†]*, Marco Haeberle[‡]*, Benjamin Steinert[‡§]*, Frank Kargl[†], Michael Menth[‡]
[†] Ulm University, Institute of Distributed Systems, Ulm, Germany
[‡] University of Tuebingen, Chair of Communication Networks, Tuebingen, Germany
[§] University of Tuebingen, Zentrum für Datenverarbeitung, Tuebingen, Germany
Email: {leonard.bradatsch,frank.kargl}@uni-ulm.de
{marco.haeberle,benjamin.steinert,menth}@uni-tuebingen.de
* These authors contributed equally

***Abstract*—Service Function Chaining (SFC) enables dynamic steering of traffic through a set of service functions based on classification of packets, allowing network operators fine-grained and flexible control of packet flows. New paradigms like Zero Trust (ZT) pose additional requirements to the security of network architectures. This includes client authentication, confidentiality, and integrity throughout the whole network, while also being able to perform operations on the unencrypted payload of packets. However, these requirements are only partially addressed in existing SFC literature. Therefore, we first present a comprehensive analysis of the security requirements for SFC architectures. Based on this analysis, we propose a concept towards the fulfillment of the requirements while maintaining the flexibility of SFC. In addition, we provide and evaluate a proof of concept implementation, and discuss the implications of the design choices.**

I. Introduction

Service Function Chaining (SFC) is a recent technology that aims to make packet processing more flexible. With traditional deployment models, Service Functions (SFs) such as Firewalls or Deep Packet Inspections (DPIs) are tied to the actual network topology. This greatly limits the possibility of applying SFs dynamically to specific packets. RFC 7665 [17] addresses this limitation by introducing a standardized SFC architecture. This SFC architecture allows packets to be dynamically forwarded through an ordered set of SFs regardless of the underlying forwarding topology.

However, although many SFs come from the security domain, the question of how to make SFC networks themselves secure remains largely unaddressed, both in the RFCs and in the literature. To be considered secure, an SFC network must meet current network security requirements. General security requirements can be derived from the concept of Zero Trust (ZT) security [31]. At least since the presidential executive order of May 12, 2021 [22], ZT has enjoyed pervasive attention and is the new de-facto standard for network security. ZT is based on the principle of *never trust, always verify*. Always verifying is an ongoing process and leads to special security requirements for networks and its components, i.e., authentication, authorization, encryption, and inspection of all network traffic [25], [31]. Consequently, these requirements must also be met by an SFC architecture in order to be considered secure by current standards. In addition, there are SFC-specific security requirements which we explain in detail in Section III.

This work was supported by the bwNET2020+ project which is funded by the Ministry of Science, Research and the Arts Baden-Württemberg (MWK). The authors alone are responsible for the content of this paper.

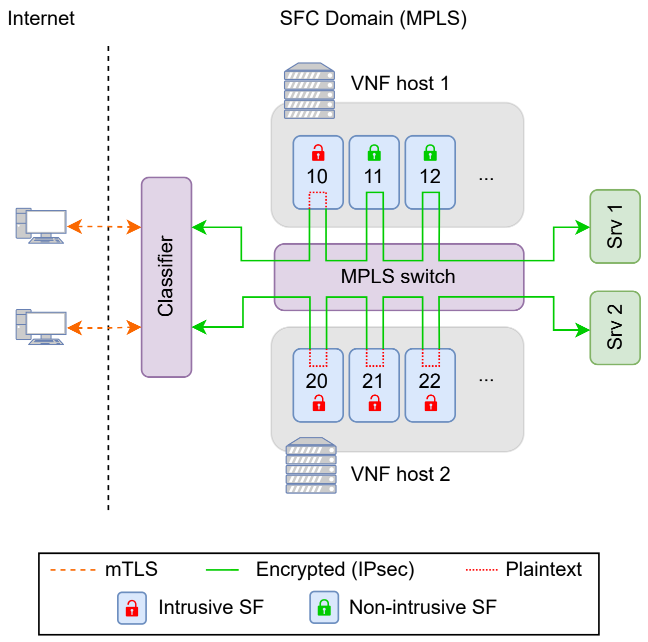

Fig. 1. Overview of the proposed SFC concept. The classifier ensures confidentiality and integrity for external and internal communication. In addition, all clients are authenticated, and authorized.

However, there are only few publications that specifically address how these requirements can be implemented in the context of SFC [5], [14]–[16].

This lack motivates the main contributions of this paper:

- First, a comprehensive requirements and security analysis regarding the security of an SFC architecture is conducted.
- Second, we introduce a novel security-driven SFC concept, illustrated in Fig. 1. Here, the classifier serves as endpoint for mTLS connections from remote clients.

Inside the SFC domain, an IPsec tunnel from the classifier to the service ensures confidentiality and integrity. The credentials for this tunnel are distributed to intrusive SFs that require access to the unencrypted payload. SR-MPLS is used as SFC encapsulation. This concept is able to provide all the flexibility benefits of SFC such as multi-protocol support or support of arbitrary SFs. At the same time, it meets most of the security requirements such as traffic confidentiality and integrity. To our knowledge, this is the first concept to achieve this.

- Third, we offer a publicly accessible Proof of Concept (PoC). This PoC is evaluated in a first evaluation regarding its performance.

In Section II, we first provide the necessary technical background. In Section III, we define the exact requirements to secure SFC architectures. Based on this, existing work is discussed in Section IV. We then present our concept in detail in Section V. Section VI then describes the corresponding PoC implementation. The PoC is evaluated in Section VII, followed by a detailed security analysis in Section VIII. We finally give a conclusion and discuss future work in Section IX.

II. Technical Background

Instead of wiring physical middleboxes together as in traditional networks, SFC enables the creation of dynamic chains of SFs and the forwarding of packets through these SFs in a specific order. This allows to dynamically adapt traffic processing based on network state or security incidents. An SFC reference architecture has been published by the IETF in RFC 7665 [17]. The main components of SFC are a classifier, Service Function Forwarders (SFFs), the SFs, and the SFC encapsulation. The SFC classifier classifies traffic based on packet characteristics such as header fields, payload information, or client-related information. The classification outcome is encoded in an SFC encapsulation, which is added to classified packets. At a minimum, this encapsulation contains steering information that is used by SFFs to forward packets to the next SFF or SF instance. Furthermore, the encapsulation may contain per-packet metadata. Metadata can be used for mechanisms such as Proof of Transit (POT). POT is leveraged to prove that packets actually followed their specified path.

SFC encapsulation may be realized with the Network Service Header (NSH) [30] which encodes steering information using a Service Path Identifier (SPI) and a Service Index (SI). Alternatively, Segment Routing (SR)-based approaches [26] may be used, for example by encoding steering information and metadata within MPLS labels [8]. Hantouti et al. [19] present an analysis of different traffic steering techniques in the context of SFC. Depending on the available hardware and the actual deployment scenario of SFC, only a limited subset of these methods is practical. For example, the NSH is not widely supported by current commercial off-the-shelf (COTS) hardware, whereas an SR-MPLS-based approach can be implemented also with legacy network devices. Certain SFs may have characteristics that directly influence the implementation of the SFC solution.

A. SFC-Aware and SFC-Unaware Service Functions

Traffic in an SFC-enabled domain is steered based on the information in the SFC encapsulation. This means that packets arriving at SFs are generally SFC-encapsulated. SFs that can handle the SFC encapsulation are called SFC-aware SFs. For most legacy functions, this is not the case as they operate on plain IP packets. These SFs are called SFC-unaware. To support SFC-unaware SFs in an SFC architecture, an SFC Proxy is used to strip the SFC encapsulation before handing the packet to the SF, and to re-encapsulate packets correctly after being processed by the SF [17]. This allows to use legacy network functions in a modern SFC system, and enables incremental deployment in traditional environments, for the cost of an additional component that adds further complexity.

B. Intrusive and Non-Intrusive Service Functions

We further differentiate between two different types of SFs that impose different restrictions and implications on our concept, similar to the distinction made by Cunha et al. [5]. On the one hand, there are non-intrusive SFs, such as a simple firewall, that do not need to read data from the payload of a packet, only access to header fields is required. On the other hand, there are intrusive SFs, such as a DPI, that need to inspect the payload of packets. In a ZT environment, the inspection of unencrypted payload is a hard requirement that is enabled by the support of intrusive SFs. In the case of TLS-encrypted traffic, this implies that the end-to-end TLS connection has to be terminated by the SF, which may raise privacy issues in certain environments. Additionally, cryptographic operations may add further delay to the traffic processing.

III. Requirements

Several aspects need to be covered by an SFC architecture to make it useful and flexible for production use, while complying with modern security standards such as ZT. This section provides a detailed analysis of the most important requirements in three different categories: A) general requirements for flexible SFC networks, B) security requirements resulting from the specifics of SFC, and C) security requirements resulting from the principles of ZT. All discussed requirements are listed in the first column of Table I.

A. General SFC Requirements

Regarding the steering method in an SFC-enabled domain, it is desirable to support a wide range of transport, security, and application layer protocols that may be present in modern, heterogeneous networks. This multi-protocol support can be achieved by implementing the SFC mechanisms at the lowest possible ISO/OSI layer.

Most legacy SFs that are deployed in today's production environments are not able to handle specific SFC encapsulation, therefore it is recommended to support SFC-unaware SFs to provide the largest possible number of existing network functionality as SF. This requirement can be fulfilled by the use of an underlying SFC proxy that is responsible for all

SF relevant actions on behalf of the SFC-unaware SF [17]. To fully utilize the potential of the classifier, the classification of packets can be extended such that it is not only based on the packet headers, but also on the payload and user-specific parameters. To support such intrusive classification, encrypted connections need to be terminated at the classifier.

B. SFC-Specific Security Requirements

There are some security requirements specific to an SFC-based architecture [17]. All data such as steering information and metadata that is present in the SFC-encapsulation must be protected from leaking to the outside. This can be achieved by removing the SFC-encapsulation before leaving the SFC-enabled domain. Likewise, to prevent circumvention of certain SFs it must not be possible to inject packets that already contain spoofed steering information into the SFC domain from outside.

Metadata that is transported within the SFC-encapsulation may contain sensitive information that should not be accessible from arbitrary intermediate hops or SFs, therefore it must be ensured that metadata is transmitted confidentially and integrity-protected. Similarly, the steering information in the SFC-encapsulation, which was determined by the classifier, must also be integrity protected to prevent unauthorized alteration of the packet's path to circumvent certain SFs [14].

In a secure SFC architecture, it is crucial to verify the integrity and correctness of packet flows within the SFC-enabled domain. To achieve such verification and therefore prevent misconfigured SFCs, a POT scheme can ensure that SFCs are correctly traversed by respective packets.

C. ZT-Specific Security Requirements

Since ZT is currently considered the most suitable approach to network security, we consider the ZT principles as general requirements to a secure network and thus also for an SFC-based network. The following requirements must be ensured for network traffic entering and leaving the SFC domain as well as for all communication within the domain.

All network communication must be authenticated and authorized following the least privilege principle. Furthermore, confidentiality and integrity for all network communication as well as availability of all network services must be ensured [31]. In addition, Perfect Forward Secrecy (PFS) is an important requirement to protect past communications against possible future leaks of encryption keys.

Another key requirement of ZT is the ability to monitor all network activities. To achieve such pervasive network visibility, all communication must be logged and inspected. Additionally, the network should be logically or physically micro-segmented to prevent lateral movement of malicious insiders [25].

Table I. Requirements towards a secure SFC architecture, and their coverage in this work and selected related work.

Requirements	This Work	[15]	[5]	[3]
Multi-Protocol Support	✓	×	×	×
SFC-unaware SFs	✓	×	✓	×
Intrusive Classification	✓	×	(✓)	✓
Leak Protection	✓	✓	×	✓
Spoof Resistant	✓	×	×	✓
Metadata Protection	×	×	×	✓
SFC Integrity	×	×	×	✓
Proof of Transit	×	×	×	×
Authentication	✓	✓	×	✓
Least-Privilege Authorization	✓	✓	×	✓
Confidentiality	✓	✓	(✓)	✓
Integrity	✓	✓	(✓)	✓
Resilience/Availability	×	(✓)	×	×
Perfect Forward Secrecy	×	×	×	✓
Pervasive Network Visibility	✓	✓	✓	✓
Micro-Segmentation	✓	✓	✓	✓

IV. Related Work

Many works have been published in the context of SFC, but few of them consider encryption mechanisms to secure data within the SFC-encapsulation or information within the payload of packets [18]. Several general encryption options for SFC are presented in [13]. For NSH-based SFC, the IETF published an RFC that defines new NSH context headers which allow for authenticated and encrypted SFC [2].

For securing traffic in multi-domain deployments, Huff et al. [23] present a respective scheme which is based on IPsec tunnels for the encryption of SFC traffic between domains.

To outsource SFs to the cloud, Wang et al. [37] presented a scheme where SFC headers are encrypted before transmission in order not to leak sensitive information to the cloud provider.

For ensuring integrity of the SFC steering data in the SFC-encapsulation and for verifying the correct order of traversal, Pattaranantakul et al. [28] present a scheme based on the enforcement of a sequentially-created multisignature where each SF has to sign all received packets.

For ensuring encryption between SFs, suitable key exchange methods must be in place. A scheme for enabling session key sharing between endpoints and SFs is presented by Lui et al. [27]. Gunleifsen et al. [16] present an approach including a Software Defined Security Association (SD-SA) that enables dynamic setup of hop-by-hop, per-flow, IPsec-based tunnels. Additionally, a PoC demonstration of isolated and encrypted SFCs is presented [15]. It is based on a tiered multi-layer scheme for SFC isolation [13]. The scheme relies on a custom extension of NSH and requires modification of SFC components. It is therefore not suitable for incremental deployment in a legacy network. In Table I, this work is listed and analyzed regarding the requirements described earlier.

For inspecting the payload of encrypted packets within SFs, it is necessary to decrypt traffic at the SF. This means breaking the end-to-end encryption principle that is present in most Internet communication today. However, an analysis of the unencrypted payload increases the overall visibility into the network flows. This is desirable in certain networks and use-cases, especially in the context of ZT, but should also be critically assessed regarding privacy. Cunha et al. [32] propose an SFC-enabled Man in the Middle (MITM) that allows intrusive SFs to inspect the payload of packets, re-enabling functionalities that are not possible on encrypted traffic such as content optimization, caching, or content-filters [5]. Furthermore, a distinction between intrusive and non-intrusive analysis of traffic is made and is reflected in the proposed architecture. The authors evaluate their architecture using Open Source MANO (OSM), showing the feasibility of the concept. A major shortcoming of this approach is that traffic is not encrypted between SFs. In a ZT-compliant architecture, traffic should not be transmitted unencrypted at any point in the network to prevent eavesdropping and tampering attacks. For comparison, this work is also listed and analyzed in Table Table I.

Bradatsch et al. [3] introduced a concept called ZTSFC. This enables fine-grained chaining of traffic based on the calculated trust score for each resource access. The prototype used for this purpose implements SFC with HTTPS, achieving an even better degree of security. However, no protocols apart from HTTPS are supported, and SFs need to be adapted to be able to handle custom header extensions. This work is also listed and analyzed in Table I.

V. Concept

In this section, we present a novel security-driven SFC concept with associated architecture and prototype implementation that is described in Section VI. Table I shows which requirements from Section III are covered by the concept. A detailed discussion of the covered requirements and how they are achieved is given in Section VIII. In the following, the assumptions underlying the concept are discussed. Then, an overview of the concept is given, the elementary components are explained, and the workflow is described.

A. Assumptions

Our SFC concept is based on certain assumptions. As ZT security becomes the de-facto standard for network security, we consider the ZT attacker model as the threat scenario. Here, not only compromised communication channels in the network but also compromised communication partners are assumed. The former could lead to threats such as passive traffic sniffing or active man-in-the-middle attacks. As a protective measure against these threats, all network traffic must be encrypted, integrity-protected, and authenticated. The latter could lead for example to unauthorized service accesses. As protection against it, any communication partner must be sufficiently authenticated and authorized [12]. In this paper, we present a basic concept that provides these protections. We do not cover specific attack vectors. We see further ZT processes such as determining the trustworthiness of each partner as out of scope of this paper as it focuses on an SFC network architecture. This extra functionality must be provided by the ZT components in the network and is discussed in related work [6], [31], [35]. We also consider the management of SFs to be out of scope. Existing work on this can be found in [1], [24], [36]. We also assume that encryption keys used for encrypting the communication between the SFs are securely

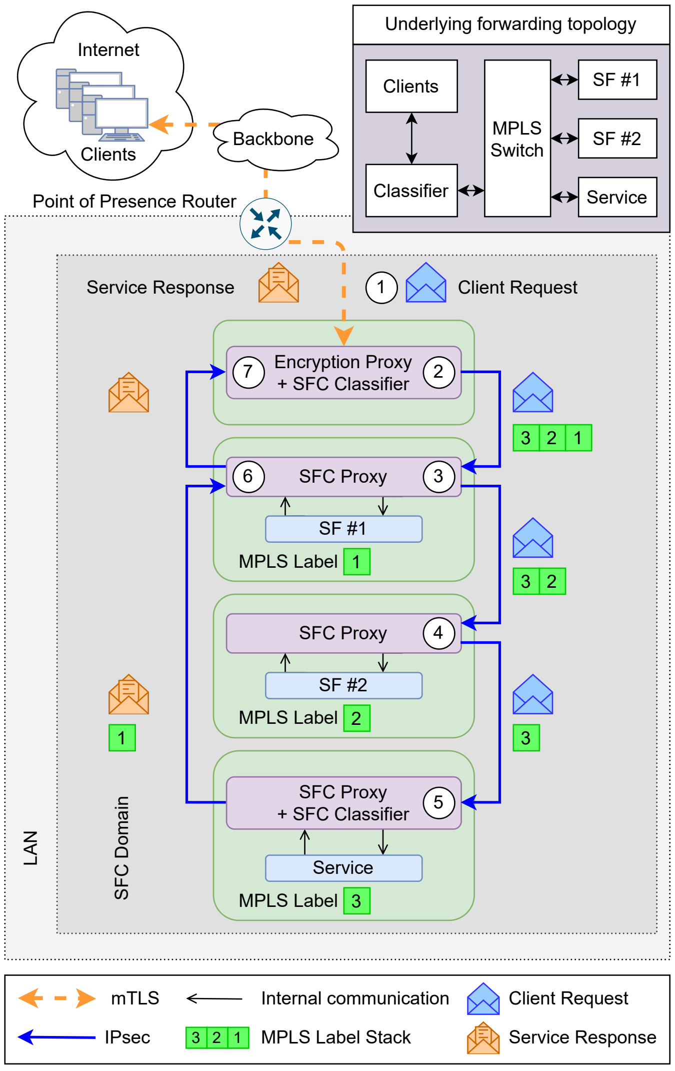

Fig. 2. Security-driven SFC network architecture.

distributed in advance to all communication partners in the form of pre-shared keys (PSKs). A method for efficient and regular distribution of new keys is discussed by Gunleifsen et al. [16]. Protocols like BRSKI [29] may also be used for key and certificate distribution.

B. Overview

The concept is based on the reference SFC architecture described in RFC 7665 [17]. Traffic steering is implemented with SR-MPLS as described in RFC 8595 [9]. A suitable underlying forwarding topology is presented in Figure 2 in the upper right corner. Instead of MPLS, NSH may also be used to achieve similar functionality. The security-driven SFC architecture is illustrated in the remainder of Figure 2. The green boxes represent the execution environments containing the logical components such as the SFC proxy and the SF. The execution environment may be realized e.g., by Linux servers, virtual machines, or containers. Note that virtualization of the execution environments allows to place multiple SFs with different characteristics on the same physical host. Depending on the bandwidth / QoS requirements and to increase availability, the logical components may be scaled across different hosts and deployed such that latency or other parameters are optimized, depending on the use case. Many works, e.g., [34] or [33], have been published that examine different types of SF deployment optimizations, therefore this aspect is omitted in our discussion. For the logical components described below, we restrict ourselves to an SFC domain localized in a LAN and do not consider cross-LAN scenarios.

C. Components

The following description of the components is provided in an abstract manner independent of implementation.

SFC Classifier: At the ingress of the SFC domain, the SFC classifier classifies incoming traffic based on predefined rules containing, e.g., IP addresses or port numbers. Resulting from this classification process, an SFC is selected, such that traffic has to traverse a specific set of SFs in a specific order. This decision is encoded in an MPLS label stack, where each label identifies either an SF, or the target service application. The classifier can dynamically chain the SFs in any order and number by placing the corresponding SF labels in this order on the label stack. As shown in Fig. 2, an SFC is unidirectional. Thus, the path to the service may differ from the path back from the service to the classifier.

Encryption Proxy: The encryption proxy works similarly to a reverse proxy and is part of the SFC classifier. It is responsible for terminating secure communication channels with clients as well as for the secure communication with SFs within the SFC domain. The encryption proxy strictly separates the encryption of the communication outside and inside the SFC domain. This makes it possible to choose two different encryption protocols depending on the use case. For example, (m)TLS could be supported for communication with clients while IPsec could be used inside the SFC domain.

SFC Proxy: SFC-unaware SFs are explicitly supported by our proposed concept. Therefore, an SFC-Proxy that handles the MPLS encapsulation and forwarding is deployed for each SF. In addition, it can perform de- and encryption of the packets. This way, intrusive SFs that need access to a packet's payload such as a DPI can be applied as well.

Service Function: Several SFs are instantiated within the SFC-enabled domain. A unique MPLS label is assigned to each SF. Each of these SFs can be dynamically inserted into the path of a packet, e.g., for adding additional service security. Due to the SFC Proxy, SFC-aware as well as SFC-unaware SFs can be deployed.

Service: The service is included in the packets' path via a dedicated MPLS label. As with the SFs, both SFC-unaware and SFC-aware services are supported.

D. Workflow

In the following, we describe an exemplary interaction of the components in case of an incoming client request. The description follows the numbering as shown in Fig. 2. (1) When a client request arrives, the encryption proxy establishes an mTLS connection with the client. According to the Server Name Indication (SNI) in the HTTPS request, the proxy knows the intended destination (here: Service #1) of the packet. Accordingly, it chooses the IPsec tunnel to use for the connection within the SFC domain. This IPsec tunnel is a logical tunnel to the service. All intrusive SFs also have access to the PSK. The encryption proxy then encrypts the client packet with IPsec. (2) Before forwarding the packet, the SFC classifier chooses the appropriate SFC depending on the defined rules. Accordingly, the MPLS labels (here: 3,2,1) are added to the packet and the packet is sent to SF #1. (3) Upon arrival at corresponding SF execution environment, the SFC proxy removes the topmost label from the MPLS label stack and caches the remaining MPLS labels. Next, it decrypts the IPsec packet. The unencrypted packet is then passed on to the service function (here: SF #1). After the packet has been processed by the SF, the SFC proxy encrypts the packet again. After that, the proxy retrieves the corresponding cached MPLS labels, adds them to the packet and sends the packet on to SF #2. (4) SF #2 processes the packet in a similar way as at SF #1. In the case that SF #2 is a non-intrusive SF, the SFC proxy skips de- and encryption. In any case, the topmost MPLS label is removed before the packet is forwarded to the service. (5) The behavior of the SFC proxy localized at the service also includes SFC classifier functionalities. A new label stack identifying the return path is added to the response packet. The selection process of the SA and the corresponding PSK for the IPsec encryption remain the same. (6) On the way back the packet is treated the same at SF #1 as in step 2. (7) Once the response packet arrives back at the SFC classifier, the encryption proxy decrypts the IPsec packet. (8) The service response is now mTLS encrypted again and sent back to the client.

VI. Proof of Concept

In order to demonstrate feasibility of our concept, we provide a PoC implementation which is openly available on GitHub [1]. In the following, we describe the implementation and provide details of the testbed setup used for evaluation.

A. Overview

The PoC consists of two clients, a classifier, two SFs and an HTTP server. They are connected to each other as shown in Figure 3. The connections between the clients and the classifier are secured with mTLS. Traffic from client 1 to the HTTP server is sent through SF 1, traffic from client 2 via SF 1 and SF 2. SF 1 does forwarding only, SF 2 is a DPI that filters all packets containing the string "test". Depending on the experiment, several instances of the SFs may be concatenated.

B. SFC Classifier & Encryption Proxy

Classification of HTTPS traffic is performed using a combination of HAProxy [2] and policy-based routing on Linux.

[1]https://github.com/uni-tue-kn/secure-sfc
[2]http://www.haproxy.org

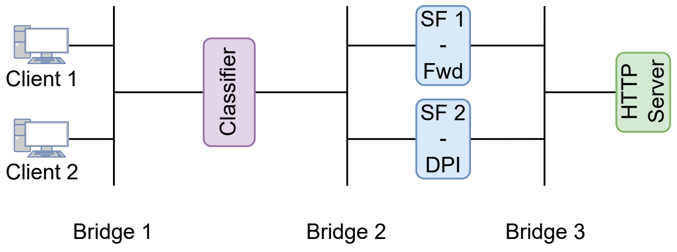

Fig. 3. Setup of the evaluation testbed.

While HAProxy classifies traffic based on the source and destination IP addresses, destination port, and URL of the request, policy based routing is used to push the corresponding MPLS label stack onto the packet. In addition, HAProxy is used to terminate the TLS connection. IPsec encryption is then applied to the unencrypted HTTP traffic using the XFRM module of the Linux kernel.

C. SFC Proxy

The SFC proxy is implemented using Python3 and the dpkt library [3]. Caching of MPLS headers is done by storing the labels in a dictionary with the IP source and destination addresses, IP identification field, TCP source and destination ports, TCP sequence number, and TCP acknowledgement number as key. IPsec in the form of ESP in transport mode with AES-GCM encryption is implemented using the Python cryptography library [4]. Packets are received and sent using raw sockets. Popping the uppermost MPLS label and routing the packet to the next SF is done using the mpls_router module of the Linux kernel.

D. Testbed

The setup is deployed on a VM running Ubuntu 20.04 and Linux kernel 5.4.0. It is equipped with 8 GB RAM and 16 Intel(R) Xeon(R) E5-2683 v4 CPU cores. Core pinning for the virtual cores is enabled to minimize the influence of virtualization. The components of the testbed are executed in separate network namespaces which are akin to containers. They are connected to each other using virtual Ethernet devices and Linux bridges. The HTTP throughput between the clients and the HTTP server are measured by performing downloads with aria2 [5].

VII. Evaluation

In this section, we conducted a first performance evaluation of the PoC and compare it to SFC without the crypto-enabled SFC proxy. First, we measure the page load times of a website with and without the proxy. Then, we compare the throughput in both cases with a varying number of SFs. Lastly, we analyze the code peformance of the SFC proxy.

We perform 10 runs per experiment and report average values together with confidence intervals.

[3]https://dpkt.readthedocs.io/en/latest/
[4]https://cryptography.io/en/latest/
[5]https://aria2.github.io

A. Page Load Times

We measure the page load times by loading a local copy of the starting page of the University of Tuebingen. This website consists of 9 files with a total size of 1.2 MB.

When using a single forwarding SF only without the proxy, the page load time is approx. 0.045 s. It increases to approx. 0.25 s when the forwarding SF is replaced by the DPI SF that is using the crypto-enabled SFC proxy. The increase can be attributed to the Python-based implementation of the SFC proxy (see Section VII-C). For comparison: loading the website via the campus network of the University of Tuebingen from the testbed VM but without the SFC deployment takes approx. 0.63 s, via the Internet approx. 2 s.

B. Throughput

We measure the maximum throughput with a varying number of SFs by downloading a 10 GB large file from one of the clients. The results including the 95% confidence interval are shown in Figure 4.

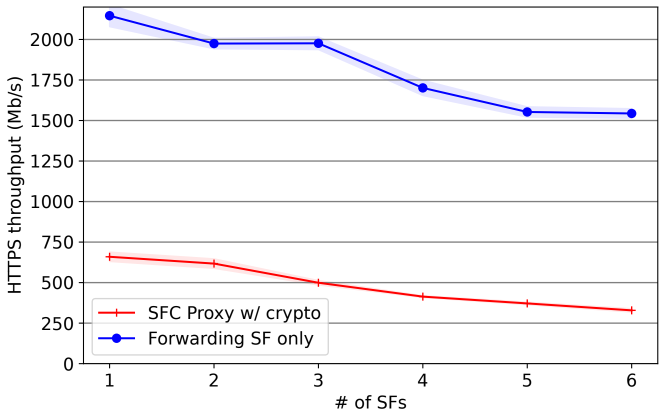

Fig. 4. HTTPS throughput with and without the SFC proxy for 1 to 6 SFs.

When one forwarding-only SF without the SFC proxy is used, the throughput is approx. 2150 Mb/s. It drops to approx. 1550 Mb/s with 6 forwarding-only SF.

When using SFs with the crypto-enabled SFC proxy, the throughput is significantly lower. For 1 SF it is approx. 660 Mb/s and drops to approx. 330 Mb/s for 6 SFs with crypto-enabled SFC-proxies. The CPU cores of the evaluation system were never fully utilized during the evaluation. The low performance in contrast to the forwarding-only SF can be attributed to the latency that is induced by the crypto-enabled SFC proxy that was also observed in Section VII-A.

In a further experiment, the MTU inside the SFC domain is reduced from 1500 Bytes to 1000 Bytes. The throughput then decreases by the same scale. This shows that the additional latency that is induced by the crypto-enabled SFC proxy is largely independent of the size of the packets.

C. Code Performance

We evaluate the code performance of the crypto-enabled SFC proxy by using the built-in cProfile Python library that generates statistics about the time that is spent in different parts of the Python program.

While performing a download of a large file, the SFC proxy spends approx. 46.8% of the time in the `recvfrom` function of the socket module of Python. This can be attributed to the time that the proxy is waiting for packets to arrive because the function is blocking. Further 21.5% of the time is spent in functions that are associated with the dpkt library, e.g., parsing and assembling packets. Approx. 3.8% of the time is spent sending packets. Other parts of the proxy, especially performing cryptographic operations for IPsec, and caching the MPLS labels takes less then 1% of the time each. This confirms the observations regarding the impact of the packet size in Section VII-B.

Thus, the limited throughput when using the proxy can be attributed to the latency induced by Python and the dpkt library. A non-prototypical implementation using lower level technologies, e.g., as a Linux kernel module, would provide a significantly higher performance.

D. General Implications

For today's networks with 100G network hardware, a solution close to the hardware should be aimed for in view of the low performance of the shown software-based PoC. Furthermore, the evaluation shows that adding all security features decreases the overall throughput. Modern network interface cards support hardware-offloading features for cryptographic operations which may be leveraged together with Single Root I/O Virtualization (SR-IOV), to support higher bandwidths and generally improve the performance of such a system. Additionally, the classifier, encryption proxy, or SFs may introduce single points of failure. Redundancy schemes with standard load balancers can help to mitigate this risk and can seamlessly be integrated into the presented architecture.

VIII. Security Analysis

In this section we discuss how the security requirements from Section III are achieved by the presented concept. With respect to the PoC, we indicate in parentheses which component implements said feature. In addition, we show possible extensions that can be used to complete the list of fulfilled security requirements.

A. Authentication and Least Privilege Authorization

Each client request is authenticated via mTLS by the Encryption Proxy (PoC: HAProxy). Within the SFC domain, the classifier and service authenticate each other via IPsec. Currently there is no authentication between the SFs. However, this can be achieved by hop-by-hop IPsec, e.g., as proposed in RFC 8994 [7], which provides hop-by-hop authenticated and encrypted communication channels between nodes.

The least-privilege authorization is implemented by the SFC classifier (PoC: HAProxy). Only packets that have been successfully authenticated are forwarded to respective services. However, more elaborate ZT-specific authorization rules can also be implemented.

B. Confidentiality and Perfect Forward Secrecy

All network traffic in our concept is encrypted. This is specifically ensured by the encryption endpoint (PoC: HAProxy & XFRM module) which enforces TLS with the client and IPsec within the SFC domain. Note that IPsec does not protect the MPLS label stack as the labels must be accessible by intermediate hops for correct forwarding of packets. For the IPsec implementation PSKs are used. The PSKs are distributed via a secure channel to the encryption endpoint, the services, and to intrusive SFs that need to inspect the unencrypted payload of a packet. Only authenticated and benign SFs are getting access to the key. Otherwise an attacker can infiltrate an SF and read unencrypted traffic. Remote attestation [10] or authentication of execution environments [21] are approaches to ensure the integrity and authenticity of SFs.

In the current concept, we considered PFS as out of scope. To enable this feature, the encryption keys must be renewed regularly. IKE or a key renewal scheme as described in [15] or in [20] can be used for this.

C. Integrity

The integrity of all packets is protected either by TLS or by IPsec. However, this does not apply to the MPLS label stack. This issue is discussed further below.

D. Availability

We considered availability to be out of scope for our concept. Nevertheless, it is an essential security objective. In case of a soft- or hardware failure, mechanisms must be in place to decrease downtime of an SFC [11]. The SFC classifier and the encryption endpoint impose a single point of failure, and should therefore be protected by (D)DoS mitigation mechanisms, and should ideally be deployed redundantly [38].

E. Pervasive Network Visibility

Accessing payload of all network traffic in the network is an important requirement of ZT. Our concept provides this pervasive network visibility. By using monitoring SFs such as a packet logger, all packets can be captured and inspected. By distributing IPsec PSKs to these SFs, access to the payload of packets is ensured.

F. Micro Segmentation

The SFC classifier and encryption endpoint represent the entry point to the SFC domain. Both (PoC: HAProxy) segment logically. Only authenticated packets are forwarded into the SFC domain. Within the SFC domain, the distribution of PSKs can be used to decide which services and SFs are given access to which packets' payload. Similarly, the SFC classifier (PoC: HAProxy & policy based routing) can use the MPLS labels (and thus the SFCs) to make fine-grained decisions about which packets should be routed through which network paths and segments. Note that missing integrity of the label stack may leave manipulation of the MPLS labels unnoticed.

G. General SFC-specific Security Requirements

Some security objectives in SFCs must be met by any SFC architecture, in order to be safe for an operator to use [17].

Amongst others, it is ensured that no SFC data can be injected into the SFC domain. This is achieved by adequate configuration of the classifier since it is the entry point to the SFC domain. Only packets that are not SFC-encapsulated are accepted from the outside of the domain (PoC: kernel policies), others are dropped, preventing the injection of packets that already carry SFC data.

In the presented concept, the classifier is not only the entry point but also the exit point of the SFC domain. Adequate configuration of the classifier ensures that no SFC headers (PoC: MPLS labels) can leak outside of the domain. All labels are removed before forwarding the service response to the client.

Ensuring confidentiality within the SFC domain for steering information and metadata may be achieved by encrypting the whole SFC-encapsulated packet. However, this prevents traffic steering using source routing since the steering information is not available to intermediate hops anymore [14]. It is possible to add sensitive metadata to the packet before encryption, in order to ensure confidentiality. In this case, the SFC proxy must be made aware of this metadata.

Integrity protection of steering information and metadata was not implemented for this concept, but is discussed in RFC9145 [2]. Note that if integrity of steering data is not given, it is possible for malicious intermediate hops (SFC-Proxies or SFFs) to change the steering information in the SFC headers, and therefore change the path of packets, e.g., to circumvent certain security SFs.

For verification of the correctness of an SFC, a POT scheme is necessary to ensure that each SF which is part of the SFC has been traversed in the correct order. This is especially important when integrity of steering data cannot be ensured since it then provides the only way of verification. Proposals for such a scheme exist [4], [28] and may be integrated in the presented concept in future work.

IX. Conclusion & Future Work

In this paper, we conducted a detailed analysis of security requirements that have to be met by an SFC network to be considered secure under current security approaches such as Zero Trust. The resulting set of requirements is listed in Table I. We classified existing work based on these requirements. This revealed a lack of flexible solutions that combine the benefits of SFC, and at the same time meet modern security requirements.

To fill this gap, we proposed a novel SFC concept that combines SFC flexibility features such as the support of many different protocols, the support of SFC-unaware SFs, and the possibility to inspect the payload of packets for classification.

This is achieved by the choice of SR-MPLS as the steering method for forwarding, by leveraging SFC proxies and by terminating encrypted connections at the SFC classifier, respectively.

Furthermore, the proposed concept complies with ZT security principles such as authentication, least privilege authorisation, confidentiality, integrity, pervasive network visibility and micro-segmentation. To the best of our knowledge, this is the first work to combine all of these features in a single concept.

We presented and evaluated a first PoC implementation of the proposed concept that achieves average website load times of 0.25 s and an average throughput of 330 Mb/s with 6 intrusive SFs in the path.

Finally, we conducted a security analysis of the presented concept that shows how the security requirements are met. For requirements not yet met by the concept, concrete approaches were given on how to achieve them.

The security analysis revealed open challenges for future work. The effective protection of SFC components against (D)DoS attacks is an open research question. There is also a need for solutions that address SFC-specific security objectives such as a proof of transit scheme, SFC integrity, as well as metadata protection on the network layer.

In addition to the security aspects, a comprehensive performance evaluation of the PoC is necessary. In addition to the performed HTTP-based evaluation, it is necessary to examine other traffic types and flow sizes. The influence of security-related operations on performance and the resulting limitations, e.g., bandwidth or scalability, must also be investigated further.

References

[1] H. U. Adoga and D. P. Pezaros, "Network Function Virtualization and Service Function Chaining Frameworks: A Comprehensive Review of Requirements, Objectives, Implementations, and Open Research Challenges," *Future Internet*, vol. 14, no. 2, p. 59, 2022.

[2] M. Boucadair, T. Reddy.K, and D. Wing, "Integrity Protection for the Network Service Header (NSH) and Encryption of Sensitive Context Headers," Internet Requests for Comments, RFC 9145, Dec. 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9145

[3] L. Bradatsch, F. Kargl, and O. Miroshkin, "Zero Trust Service Function Chaining," in *Conference on Networked Systems 2021 (NetSys 2021)*, ser. Electronic Communications of the EASST, vol. 80, 2021.

[4] F. Brockners, S. Bhandari, T. Mizrahi, S. Dara, and S. Youell, "Proof of Transit," Internet Engineering Task Force, Internet-Draft draft-ietf-sfc-proof-of-transit-08, Nov. 2020. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-sfc-proof-of-transit-08

[5] V. A. Cunha, M. B. de Carvalho, D. Corujo, J. P. Barraca, D. Gomes, A. E. Schaeffer-Filho, C. R. P. dos Santos, L. Z. Granville, and R. L. Aguiar, "An SFC-enabled approach for processing SSL/TLS encrypted traffic in Future Enterprise Networks," in *2018 IEEE Symposium on Computers and Communications (ISCC)*, 2018, pp. 01 013–01 019.

[6] T. Dimitrakos, T. Dilshener, A. Kravtsov, A. L. Marra, F. Martinelli, A. Rizos, A. Rosetti, and A. Saracino, "Trust Aware Continuous Authorization for Zero Trust in Consumer Internet of Things," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, Feb. 2020, pp. 1801–1812.

[7] T. Eckert, M. H. Behringer, and S. Bjarnason, "An Autonomic Control Plane (ACP)," RFC 8994, May 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc8994

[8] A. Farrel, S. Bryant, and J. Drake, "An MPLS-Based Forwarding Plane for Service Function Chaining," Internet Requests for Comments, IETF, RFC 8595, Jun. 2019. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8595.txt

[9] ——, "An MPLS-Based Forwarding Plane for Service Function Chaining," IETF, RFC 8595, Jun. 2019.

[10] G. Fedorkow, E. Voit, and J. Fitzgerald-McKay, "TPM-based Network Device Remote Integrity Verification," Internet Engineering Task Force, Internet-Draft draft-ietf-rats-tpm-based-network-device-attest-14, Mar. 2022, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-rats-tpm-based-network-device-attest-14

[11] A. Gausseran, A. Tomassilli, F. Giroire, and J. Moulierac, "Don't interrupt me when you reconfigure my Service Function Chains," *Computer Communications*, vol. 171, pp. 39–53, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366421000712

[12] E. Gilman and D. Barth, *Zero Trust Networks: Building Secure Systems in Untrusted Networks*. O'Reilly Media, Inc., Jun. 2017.

[13] H. Gunleifsen, V. Gkioulos, and T. Kemmerich, "A tiered control plane model for service function chaining isolation," *Future Internet*, vol. 10, no. 6, p. 46, 2018.

[14] H. Gunleifsen and T. Kemmerich, "Security requirements for service function chaining isolation and encryption," in *2017 IEEE 17th International Conference on Communication Technology (ICCT)*, 2017, pp. 1360–1365.

[15] H. Gunleifsen, T. Kemmerich, and V. Gkioulos, "A Proof-of-Concept Demonstration of Isolated and Encrypted Service Function Chains," *Future Internet*, vol. 11, no. 9, 2019. [Online]. Available: https://www.mdpi.com/1999-5903/11/9/183

[16] ——, "Dynamic Setup of IPsec VPNs in Service Function Chaining," *Computer Networks*, vol. 160, pp. 77–91, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128619300969

[17] J. Halpern and C. Pignataro, "Service Function Chaining (SFC) Architecture," Internet Requests for Comments, IETF, RFC 7665, Oct. 2015. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7665.txt

[18] H. Hantouti, N. Benamar, and T. Taleb, "Service Function Chaining in 5G & Beyond Networks: Challenges and Open Research Issues," *IEEE Network*, vol. 34, no. 4, pp. 320–327, 2020.

[19] H. Hantouti, N. Benamar, T. Taleb, and A. Laghrissi, "Traffic steering for service function chaining," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 487–507, 2018.

[20] F. Hauser, M. Häberle, M. Schmidt, and M. Menth, "P4-IPsec: site-to-site and host-to-site VPN with IPsec in P4-based SDN," *IEEE Access*, vol. 8, pp. 139 567–139 586, 2020.

[21] F. Hauser, M. Schmidt, and M. Menth, "xRAC: Execution and Access Control for Restricted Application Containers on Managed Hosts," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020, pp. 1–9.

[22] T. W. House. (2021, May) Executive Order on Improving the Nation's Cybersecurity. [Online]. Available: https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/

[23] A. Huff, G. Venâncio, V. F. Garcia, and E. P. Duarte, "Building multi-domain service function chains based on multiple NFV orchestrators," in *2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2020, pp. 19–24.

[24] K. Kaur, V. Mangat, and K. Kumar, "A comprehensive survey of service function chain provisioning approaches in SDN and NFV architecture," *Computer Science Review*, vol. 38, p. 100298, 2020.

[25] J. Kindervag, S. Balaouras, and L. Coit, "Build Security Into Your Network's DNA: The Zero Trust Network Architecture," Forrester, Tech. Rep., Nov. 2010.

[26] C. Li, A. E. Sawaf, R. Hu, and Z. Li, "A Framework for Constructing Service Function Chaining Systems Based on Segment Routing," IETF, Internet-Draft draft-li-spring-sr-sfc-control-plane-framework-05, Oct. 2021, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-li-spring-sr-sfc-control-plane-framework-05

[27] C. Liu, Y. Cui, K. Tan, Q. Fan, K. Ren, and J. Wu, "Building generic scalable middlebox services over encrypted protocols," in *IEEE INFOCOM 2018-IEEE conference on computer communications*. IEEE, 2018, pp. 2195–2203.

[28] M. Pattaranantakul, Q. Song, Y. Tian, L. Wang, Z. Zhang, A. Meddahi, and C. Vorakulpipat, "On Achieving Trustworthy Service Function Chaining," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3140–3153, 2021.

[29] M. Pritikin, M. Richardson, T. Eckert, M. H. Behringer, and K. Watsen, "Bootstrapping Remote Secure Key Infrastructure (BRSKI)," RFC 8995, May 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc8995

[30] P. Quinn, U. Elzur, and C. Pignataro, "Network Service Header (NSH)," IETF, RFC 8300, Jan. 2018.

[31] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero Trust Architecture," *NIST Computer Security Resource center*, Aug. 2020.

[32] E. Sousa, V. A. Cunha, M. B. de Carvalho, D. Corujo, J. P. Barraca, D. Gomes, A. E. Schaeffer-Filho, C. R. dos Santos, L. Z. Granville, and R. L. Aguiar, "Orchestrating an SFC-enabled SSL/TLS traffic processing architecture using MANO," in *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2018, pp. 1–7.

[33] G. Sun, Y. Li, H. Yu, A. V. Vasilakos, X. Du, and M. Guizani, "Energy-efficient and traffic-aware service function chaining orchestration in multi-domain networks," *Future Generation Computer Systems*, vol. 91, pp. 347–360, 2019.

[34] M. M. Tajiki, S. Salsano, L. Chiaraviglio, M. Shojafar, and B. Akbari, "Joint energy efficient and QoS-aware path allocation and VNF placement for service function chaining," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 374–388, 2018.

[35] Y. Tao, Z. Lei, and P. Ruxiang, "Fine-Grained Big Data Security Method Based on Zero Trust Model," in *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, Dec. 2018, pp. 1040–1045, ISSN: 1521-9097.

[36] A. N. Toosi, J. Son, Q. Chi, and R. Buyya, "ElasticSFC: Auto-scaling techniques for elastic service function chaining in network functions virtualization-based clouds," *Journal of Systems and Software*, vol. 152, pp. 108–119, 2019.

[37] H. Wang, X. Li, Y. Zhao, Y. Yu, H. Yang, and C. Qian, "SICS: Secure in-cloud Service Function Chaining," *arXiv preprint arXiv:1606.07079*, 2016.

[38] L. Zhang, Y. Wang, X. Qiu, and H. Guo, "Redundancy mechanism of service function chain with node-ranking algorithm," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2019, pp. 586–589.