计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(8):286−291 [doi: 10.15888/j.cnki.csa.006500]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于软件定义安全的服务功能链设计①

梁琼瑶[1], 秦　华[1], 刘文懋[2]

[1](北京工业大学 信息学部, 北京 100124)
[2](北京神州绿盟信息安全科技股份有限公司, 北京 100089)

摘　要: 随着云计算的高速发展和广泛应用, 云环境下安全防护成为亟待解决的问题. 针对如何灵活、动态的调度虚拟化的安全设备问题, 论文提出基于软件定义安全的服务功能链, 根据用户需求构建安全服务功能链, 调度安全资源池中的虚拟安全设备, 实验证明本文方法的有效性.

关键词: 软件定义安全; 云计算; 服务功能链

引用格式: 梁琼瑶,秦华,刘文懋.基于软件定义安全的服务功能链设计.计算机系统应用,2018,27(8):286–291. http://www.c-s-a.org.cn/1003-3254/6500.html

Service Function Chain Design Based on Software Defined Security

LIANG Qiong-Yao[1], QIN Hua[1], LIU Wen-Mao[2]

[1](Fabulty of Information Science, Beijing University of Technology, Beijing 100124, China)
[2](NSFOCUS Information Technology Co. Ltd., Beijing 100089, China)

Abstract: With the rapid development of cloud computing and its widespread application, security protection in the cloud environment has become a pressing issue. Aiming at the problem of how to flexibly and dynamically schedule virtualized security devices, this study proposes a service function chain based on software-defined security, builds the security service function chain according to user requirements, and schedules virtual security devices in secure resource pools. The experiment proves the validity of the method in this paper.

Key words: software defined security; cloud computing; service function chain

1　前言

传统网络中部署安全设备往往受限于网络物理拓扑的紧耦合性, 且部署后的安全设备功能相对固定, 不易扩容. 当前特别是在云计算环境下, 面对快速、多变、持续性的安全威胁, 亟需具有快速、按需而变能力的安全防护方案. 2012 年 Gartner 机构提出软件定义安全[1](Software Defined Security, SDS), 通过分离安全控制平面和数据平面, 将网络安全设备与其接入模式、部署方式进行解耦, 底层通过网络功能虚拟化(Network Function Virtualization, NFV) 将安全设备虚拟化形成安全资源池, 顶层通过软件定义的方式进行自动化管理, 实现新业务安全需求的快速开发和部署.

基于 SDS 构建的安全服务功能链可动态、灵活的提供各种安全服务. IETF SFC 工作组的草案[2,3]中对服务功能链的定义、架构、使用场景、路由转发、协议及报文格式等进行了详细阐述. 文献[4,5]主要描述了在 NFV 架构下对服务功能链资源调度工作, 充分考虑了对网络资源的优化利用, 但未提及如何构建服务功能链. 文献[6]提出了基于 SDN/NFV 的安全服务链架构, 描述了 SDN, NFV 与安全服务链的关系, 并通过组合虚拟安全应用模块来构建安全服务链, 但未具体描述安全服务链的实现以及安全资源调度的问题.

综上, 本文基于 SDS 的服务功能链框架, 研究在虚拟网络环境中动态编排部署安全服务功能链, 主要

① 收稿时间: 2018-01-01; 修改时间: 2018-01-23; 采用时间: 2018-02-06; csa 在线出版时间: 2018-07-28

讨论了安全服务功能链的构建、虚拟安全设备的调度以及流量调度方面的问题, 实现按需而变的安全防护机制.

2 软件定义安全服务功能链设计

2.1 软件定义安全架构

SDS 架构可分为四个部分: 安全应用接口, 实现安全功能的安全资源池, 软件定义的安全控制器和软件定义的 SDN 控制器, 如图 1 所示.

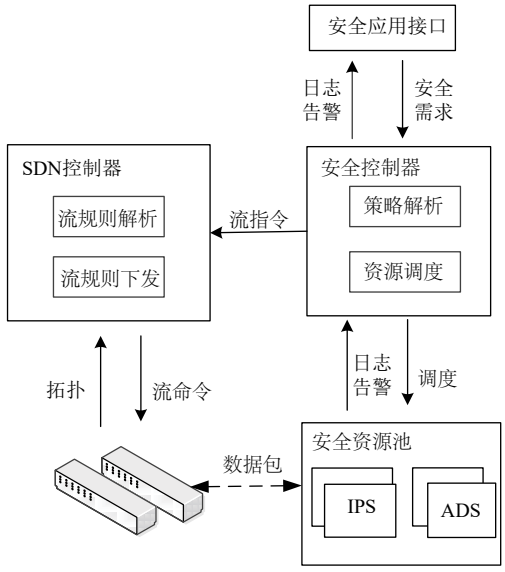

图 1 SDS 架构

安全应用接口: 位于架构的顶层, 向安全控制器推送用户定义的安全需求;

安全资源池: 由传统的物理安全防护组件、虚拟化安全防护组件组成, 通过安全能力抽象和资源池化, 将安全设备抽象为具有安全能力的资源池, 提供基础的安全防护能力;

安全控制器: 安全控制器北向与安全应用进行数据和安全需求的交互; 南向提供对基础安全防护组件的注册、调度的管理; 西向与 SDN 控制器对接, 生成需要的逻辑拓扑、数据流的调度指令;

SDN 控制器: 维护全网视图, 监控全网拓扑, 根据安全控制器传递的流指令实现网络流量重定向的功能.

2.2 安全设备虚拟化

将安全设备部署到云计算环境中形成集中管理的安全资源池, 需要使用 NFV 虚拟化技术实现安全设备虚拟化, 以便根据需求动态灵活的部署.

2.2.1 基于 NFV 的安全架构

基于 NFV[7]的安全架构, 如图 2 所示, 底层的虚拟安全设备 SFVI 抽象为安全资源池里的资源, SFV 通过软件编程的方式进行智能化、自动化的业务编排和管理, 为 VSF 提供各种安全能力, 以完成相应的安全功能, 从而实现一种灵活的安全防护. 当用户申请某种安全能力时, 通过统一的资源调度算法选择合适的节点, 在该节点上完成相应的业务.

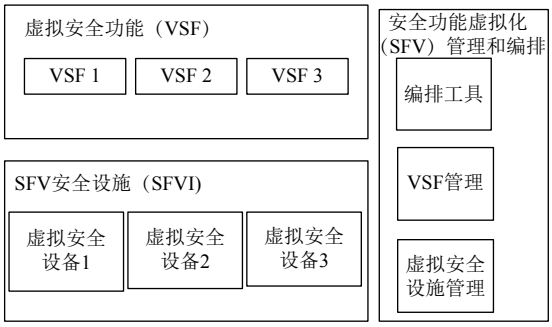

图 2 基于 NFV 的安全架构

2.2.2 基于 NFV 的安全资源调度

为了充分利用安全资源池中宿主机的资源, 改善负载平衡, 在启动虚拟安全设备时综合考虑三种资源指标 (内存、CPU 和磁盘空间) 调度算法.

调度工作分为两个阶段:

(1) 过滤阶段: 比较宿主机节点每个资源指标 (空闲内存、空闲 CPU、空闲磁盘空间) 是否大于所启动虚拟安全设备占用的资源空间, 若其中一个指标不符合, 则将该宿主机节点过滤掉.

(2) 计算阶段: 计算剩余宿主机资源指标归一化效用值.

1) $MEM_{\max}$ 表示剩余宿主机中最大空闲内存值, $MEM_{\min}$ 表示剩余宿主机中最小空闲内存值, 空闲内存为 X 的宿主机其内存的归一化效用值 U_m 的计算如公式 (1) 所示.

$$U_m = (X - MEM_{\min}) / (MEM_{\max} - MEM_{\min}) \quad (1)$$

2) $CPU_{\max}$ 表示剩余宿主机中最大空闲 CPU 数, $CPU_{\min}$ 表示剩余宿主机中最小空闲 CPU 数, 空闲 CPU 数为 Y 的宿主机其 CPU 的归一化效用值 U_c 的计算如公式 (2) 所示.

$$U_c = (Y - CPU_{\min}) / (CPU_{\max} - CPU_{\min}) \quad (2)$$

3) $DISK_{\max}$ 表示剩余宿主机中最大空闲磁盘空间

值, $DISK_{\min}$ 表示剩余宿主机中最小空闲磁盘空间值, 空闲磁盘空间为 Z 的宿主机其磁盘空间的归一化效用值 U_d 的计算如公式 (3) 所示.

$$U_d = (Z - DIS\,K_{\min}) / (DIS\,K_{\max} - DIS\,K_{\min}) \quad (3)$$

4) 综合三项资源指标的效用值, 该宿主机节点的综合效用值 U 的计算如公式 (4) 所示.

$$U = U_m * W_1 + U_c * W_2 + U_d * W_3 \quad (4)$$

其中, W_1、W_2 和 W_3 分别代表三种资源指标 (内存、CPU、磁盘空间) 对应的权重值且 W_1+W_2+W_3=1, U 值最大的节点即为要选择的最优调度节点.

2.3 安全服务功能链

2.3.1 术语定义

服务功能术语定义如下:

安全服务功能 (Security Service Function, SSF): 安全服务功能负责对收到的报文进行具体处理, 如防火墙、IDS、IPS 等安全设备.

分类器 (Classifier): 根据不同的用户安全需求, 对不同的流量进行分类. 分类后的流量会经过不同的 SSF.

服务功能转发器 (Service Function Forwarder, SFF): 根据报文携带的安全服务功能链封装信息将流量转发至 SSF 上, 同时处理从 SSF 回传的流量. 可以把 SFF 理解为一个 OpenvSwitch 虚拟交换机.

安全服务功能链 (Security Service Function Chain, SSFC): 一条 SSFC 定义了一个有序的虚拟安全功能 (VSF) 集合, 网络流量需按既定顺序通过这些 VSF.

安全服务功能路径 (Security Service Function Path, SSFP): 介于 SSFC 和 SFF 之间, 给出了每个 SSF 对应的 SFF.

2.3.2 SSFC 架构

SSFC 的实现包含如下几个组件[8], 如图 3 所示.

逻辑层面上根据用户安全需求, 生成安全服务功能链, SDN 控制器为该安全服务功能链选择经过的 SSF 实例, 实现逻辑 SSFC 到物理转发路径的映射;

物理层面上包含 Classifier、SFF 和 SSF 组件. Classifier 对用户流量进行分类, 确定对应逻辑的服务功能链; SFF 将封装有对应逻辑服务功能链的数据报文进行逐跳转发; SSF 即虚拟安全设备, 用于处理收到的数据报文.

服务功能链实现流程如图 3 所示, 包括以下步骤:

(1/2) 根据用户安全需求, 生成安全服务功能链, 并将信息发送给 SDN 控制器;

(3) 控制器根据安全服务功能链生成安全服务功能路径 SSFP 并将相关的流表信息下发给 Classifier、SFF;

(4) 用户数据报文进入 Classifier, Classifier 根据 3 下发的规则对流进行分类, 匹配相应的 SSFP;

(5/6) 与 SSFP 相匹配的数据报文发送到 SFF, SFF 根据 SSFP 将流量发送到 SSF1, SSF1 将处理后的报文返回 SFF;

(7/8) 同步骤 5、6;

(9) 根据 SSFP, 报文已完成 SSFC 的转发路径, 将报文发送到传统网络中.

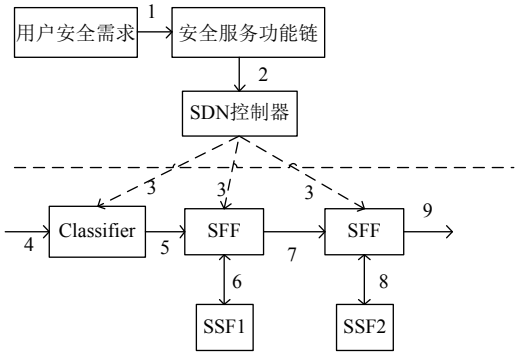

图 3 SFC 架构

2.3.3 SSFC 逻辑表示

SSFC 是一组有序的 VSF 集合, 用户流量按照制定的策略依次通过多个 SSF. 如图 4 所给出的安全服务功能链 SSFC 1、SSFC 2、SSFC 3 为例, 所提供的 SSF 依次为:

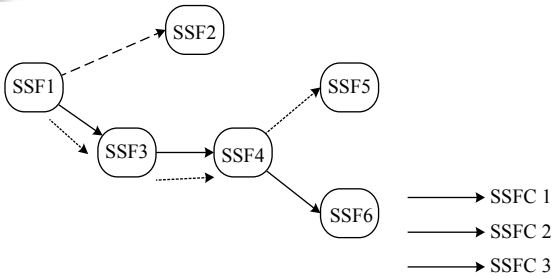

图 4 SSFC 逻辑表示图

SSFC 1: SSF1-->SSF2;

SSFC 2: SSF1-->SSF3-->SSF4-->SSF6;

SSFC 3: SSF1-->SSF3-->SSF4-->SSF5.

2.3.4 SSFC 物理实现

SSFC 架构中各种组件完成 SSFC 报文转发的实

现依据是 IETF 定义的 SSFC 数据传输协议 Network Service Header(NSH), NSH 专门用于创建动态服务功能链, NSH 报文格式如下:

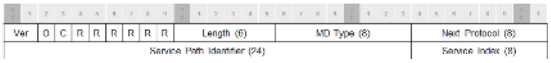

图 5 NSH 报文格式

Service Path Identifier(SPI) 和 Service Index(SI) 为两个重要字段.

SPI: 标识服务路径, 参与节点必须使用此标识符进行服务功能路径选择.

SI: 提供 SSFP 的位置, 给定 SSFP 的初始分类器应该将 SI 设置为 255, 但是控制平面可以适当地配置 SI 的初始值 (即考虑服务功能路径的长度). 在执行所需服务后, NSH 数据包中的 SI 值自减 1.

SI 与 SPI 一起用于确定一条服务功能链, 并用于确定路径中的下一个 SFF/ SSF. 图 4 中 SSFC1、SSFC2 的表示如表 1 所示.

表 1 SSFC1、SSFC2 表示

SPI	SI	Next hop SSF	SPI	SI	Next hop SSF
1	255	SSF1	2	254	SSF3
1	254	SSF2	2	253	SSF4
2	255	SSF1	2	252	SSF6

逻辑层面 SSFC 到物理层面转发路径的映射表示:

(1)SSFC={SSF1, SSF2, …, SSFn}. 表示一条 SSFC 由多个 SSF 组成有顺序的安全服务功能链;

(2) Classifier={DPID, SPI, SI, PORT}. DPID 表示 OVS 的标识符, SPI 表示链的标识符, SI 为初始值, PORT 表示数据包封装上 NSH 协议后从这个 OVS 端口出去;

(3) SFF={DPID, SPI, SI, PORT, SSF}. DPID 表示 OVS 的标识符, SPI、SI 为数据报文的匹配项, PORT 表示连接 SSF 的 OVS 端口;

(4) SSF={TYPE, SFF}. TYPE 表示虚拟安全设备的类型, 如 WAF、ADS 等; SFF 表示虚拟安全设备所映射的服务功能转发器, 通常指 OVS 桥.

2.4 流表下发

流表下发是由 SDN 控制器控制, 安全控制器将解析的 SSFC 流指令传递到 SDN 控制器, SDN 控制器获取拓扑并结合安全控制器的流指令, 通过下发流表的方式, 实现流量重定向操作, 使分类后的流量依次经过相应的 SSFs 再到目标网络. 本文的流表规则是通过 OpenFlow[9]多级流表的方式实现的.

2.4.1 流表规则

根据 SSFC 架构, 流表下发在 Classifier 和 SFF 上, 设计如下:

Classifier: 数据包到达 Classifier, 首先与 table=0 的流表匹配, 经匹配项 (如源 IP、目的 IP、协议类型、MAC 地址等) 匹配后进行 NSH 封装 actions={set_nsp, set_nsi, output}, set_nsp 表示加载链的标识符, set_nsi 表示加载虚拟安全设备的次序, output 表示将封装 NSH 的数据包从 OVS 的端口出去.

SFF: 数据包达到 SFF 后, 首先匹配 nsp, 再匹配 nsi, 这里使用到了多级流表, 流程如下: {(table=0, actions=goto_table:1); (table=1, match=nsp, actions=goto_table:4);(table=4, match=(nsp, nsi), actions=goto_table:10);(table=10, match=(nsp, nsi), actions=output)}, 不同级别的流表有不同的作用:

Table0: 将数据包分类并进行 NSH 封装;

Table1: 识别数据包经过哪条服务功能链;

Table4: 数据包的下一跳, 即下一跳要经过的虚拟安全设备;

Table10: 数据包的出口.

3 实验验证

3.1 实验拓扑

以图 6 中 SSFC1 为例, 构建如下图所示拓扑, 图中最顶端的计算机中启动 SDN 控制器 (ODL) 和安全控制器, Classifier、SFF1、SFF2 中为 OpenvSwitch 交换机, VSD1、VSD2 为虚拟安全设备. H1 与 H2 通信, 要求通信过程中依次经过虚拟安全设备 VSD1、VSD2.

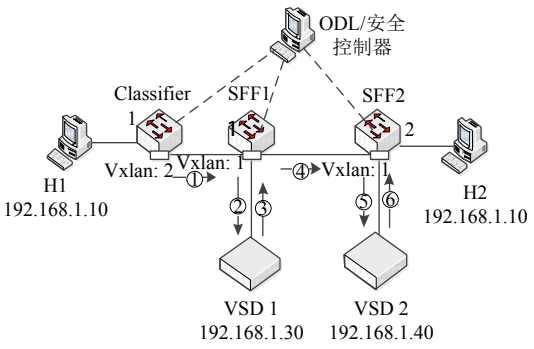

图 6 实验拓扑图

用户配置所需的安全服务功能链即依次经过 VSD1(firewall)、VSD2(ips), 如图 7 所示. 安全控制器根据用户配置在安全资源池中启动相应的虚拟安全设备, VSD1 的启动如图 8 所示.

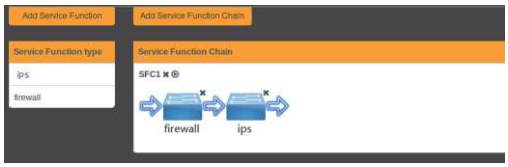

图 7 SSFC 前端显示

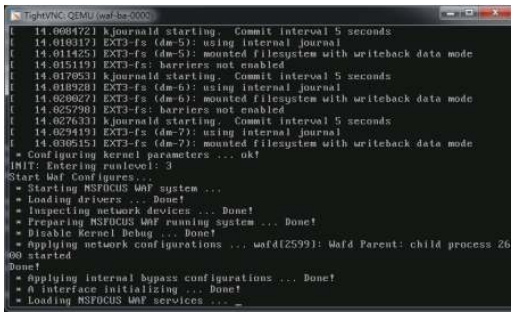

图 8 VSD1(firewall) 启动图

3.2 实验结果分析

3.2.1 流表数据

安全控制器将 SSFC 的流指令发送给 ODL 控制器, ODL 生成相应的流表下发给 Classifier、SFF. H1 到 H2 的数据流流向按图 6 中的①~⑥各步骤所示, 实验数据如下:

① H1 的数据包到达 Classifier, 数据包封装上 NSH, 从 2 端口出. 流表如下:

cookie=0x0, duration=2010.659s, table=0, n_packets=31, n_bytes=2631, idle_age=753, priority=1000, in_port=1, nw_src=192.168.1.10, nw_dst=192.168.1.60, actions=set_nsp:0x2c, set_nsi:255, output:2

② 数据流达到 SFF1 并匹配以下流表, nsp=44, nsi=255, 目的地址=192.168.1.30, 从 1 端口出.

cookie=0x14, duration=2286.533s, table=0, n_packets=92, n_bytes=19782, priority=5, in_port=1 actions=goto_table:1

cookie=0x14, duration=2285.412s, table=1, n_packets=62, n_bytes=5262, priority=350, nsp=44 actions=goto_table:4

cookie=0x14, duration=2285.333s, table=4, n_packets=31, n_bytes=2631, priority=650, nsp=44, nsi=255 actions=load:0xc0a8011e->NXM_NX_TUN_IPV4_DST[], goto_table:10

cookie=0xba5eba11ba5eba11, duration=2285.143s, table=10, n_packets=31, n_bytes=2631, priority=751, nsp=44, nsi=255, in_port=1 actions=IN_PORT

③ VSD1 对数据包进行处理, nsp=44, nsi=254, 再将数据报文转发到 SFF1

④ 数据报文匹配以下流表, nsp=44, nsi=254, 从 1 端口出.

cookie=0x14, duration=2286.533s, table=0, n_packets=92, n_bytes=19782, priority=5 actions=goto_table:1

cookie=0x14, duration=2285.412s, table=1, n_packets=62, n_bytes=5262, priority=350, nsp=44 actions=goto_table:4

cookie=0x14, duration=2285.066s, table=4, n_packets=31, n_bytes=2631, priority=650, nsp=44, nsi=254 actions=goto_table:10

cookie=0xba5eba11ba5eba11, duration=2285.037s, table=10, n_packets=31, n_bytes=2631, priority=751, nsp=44, nsi=254, in_port=1 actions=IN_PORT

⑤ 数据报文匹配以下流表, nsp=44, nsi=254, 目的地址=192.168.1.40, 从 1 端口出.

cookie=0x14, duration=2385.368s, table=0, n_packets=91, n_bytes=31671, priority=5 actions=goto_table:1

cookie=0x14, duration=2385.297s, table=1, n_packets=31, n_bytes=2631, priority=350, nsp=44 actions=goto_table:4

cookie=0x14, duration=2385.283s, table=4, n_packets=31, n_bytes=2631, priority=650, nsp=44, nsi=254 actions=load:0xc0a80128->NXM_NX_TUN_IPV4_DST[], goto_table:10

cookie=0xba5eba11ba5eba11, duration=2285.259s, table=10, n_packets=31, n_bytes=2631, priority=751, nsp=44, nsi=254, in_port=1 actions=IN_PORT

⑥ VSD2 对数据包进行处理, nsp=44, nsi=253, 再将数据报文转发到 SFF2.

3.2.2 报文数据

SFF1 到 VSD1: nsp=44, nsi=255, SFF1 到 SFF2: nsp=44, nsi=254. 分别如图 9、图 10 所示.

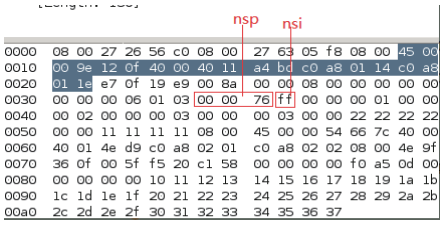

图 9 报文分析

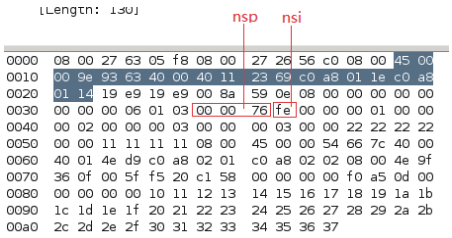

图 10 报文分析

通过实验证明, 基于软件定义安全的服务功能链的设计是可行的. 在虚拟网络环境中, 根据用户的安全需求使数据流有序的依次经过各个虚拟安全设备, 实现网络流量的动态控制.

4 结束语

本文基于软件定义安全的服务功能链框架, 充分利用 NFV 虚拟化的特性, 实现了根据用户安全需求动态编排部署安全功能服务链, 并通过实验证明了基于软件定义安全的服务功能链可以实现按需而变的安全防护. 本文研究的服务功能链是在单一的 SDN 控制器基础上, 然而实际网络应用中, 单一的 SDN 控制器对全网进行监管理负载过重, 下一步在分布式控制器的基础上研究服务功能链.

参考文献

1 Gartner. The impact of software-defined data centers on information security. https://www.gartner.com/doc/2200415. [2012-10-06].

2 Service function chaining use cases in data centers. https://datatracker.ietf.org/doc/draft-ietf-sfc-dc-use-cases/. [2016-11-29].

3 Network service header. https://datatracker.ietf.org/doc/draft-ietf-sfc-nsh/. [2016-11-29].

4 唐宏, 罗雨佳. NFV 业务链资源分配技术. 电信科学, 2015, 31(11): 2015300.

5 Shen WY, Yoshida M, Kawabata T, *et al*. vConductor: An NFV management solution for realizing end-to-end virtual network services. Proceedings of the 16th Asia-Pacific Network Operations and Management Symposium (APNOMS). Hsinchu, China. 2014. 1–6.

6 Lee W, Choi YH, Kim N. Study on virtual service chain for secure software defined networking. Advanced Science and Technology Letters, 2013, 29(13): 177–180.

7 NFV white paper. https://portal.etsi.org/nfv/nfv_white_paper.pdf.

8 Halpern J, Pignataro C. Service function chaining (SFC) architecture. RFC 7665, 2015.

9 McKeown N, Anderson T, Balakrishnan H, *et al*. OpenFlow: Enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69–74. [doi: 10.1145/1355734]