SR-Snort: IPv6 Segment Routing Aware IDS/IPS

Ahmed Abdelsalam*, Stefano Salsano†, Francois Clad‡, Pablo Camarillo‡, Clarence Filsfils‡,
*Gran Sasso Science Institute, †University of Rome Tor Vergata, ‡Cisco Systems

***Abstract*—Service Function Chaining (SFC) allows the delivery of advanced end-to-end services composed of one or more network functions. IPv6 Segment Routing (SRv6) is a network architecture based on source routing, where a list of segments is attached to packets to enforce different path from the shortest one. SRv6 supports SFC by assigning each network function a segment and combining these segments into a segment list. In order to fully leverage the SRv6 network programming capabilities, network functions are required to be SR-aware. In this paper, we present our implementation of SR-Snort, a SR-aware intrusion detection system (IDS) and intrusion prevention system (IPS). We extended the open-source implementation of Snort, so it can apply the configured rules to the inner packet of SR traffic. SR-Snort can handle both inner IPv4 and inner IPv6 traffic. It can work in either IDS or IPS mode.**

***Index Terms*—Service Function Chaining (SFC), NFV, SR, SRv6, Snort**

I. Introduction

Network Functions Virtualization (NFV) offers an agile way to design and deploy network services [1]. In an NFV infrastructure, traditional specialized physical appliances are being replaced with software modules, known as Virtual Network Functions (VNFs). Service Function Chaining (SFC) is the process of forwarding packets through a set of VNFs required to deliver an end-to-end service. SFC requires a steering mechanism which can be offered through IPv6 Segment Routing (SRv6). SRv6 is the instantiation of the Segment Routing (SR) architecture over the IPv6 data plane. It defines a new IPv6 Routing Extension header type, known as Segment Routing Header (SRH), that includes a list of segments to be traversed by the packet [2]. Each segment represents a function to be called at a specific location in the network. A SRv6 encapsulated packet is as shown in Figure 1. SFC can be achieved with SRv6 by associating each VNF with a segment and combining such segments in segment list.

VNFs can be categorized into SR-aware and SR-unaware, depending on their ability to process SR encapsulated packets [3]. SR-unaware VNFs are not able to process SR traffic and require an *SR-proxy* to be included in a SR based SFC [4]. SR-aware VNFs are able to process SR traffic, which imply being able to process the original packet despite the fact that it has been encapsulated within a SR packet. SR-aware VNFs can fully leverage the SRv6 network programming capabilities to implement advanced SFC features, such as conditional branching and jumping to arbitrary segment in the segment list and exchanging information between network functions [5].

This work has been partially supported by the Cisco University Research Program (URP) fund

In this paper, we show a demo of SR-Snort, a SR-aware Intrusion Detection System (IDS) and Intrusion Prevention system (IPS). SR-Snort extends the open-source implementation of Snort, so it can apply Snort rules to inner packets of SRv6 encapsulated traffic. Section II presents the architecture of Snort. The architecture of SR-Snort is explained in section III. In section IV, we show a demo for SR-Snort included in a SRv6 based SFC.

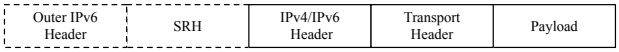

Fig. 1: SRv6 encapsulated packet

II. Snort

Snort is an open-source network IDS and IPS. It is the world's most widely deployed IPS technology [6]. Snort can be used as a packet sniffer (like tcpdump) to read packets from a specific network interface and prints their headers fields. It can also be used to perform analysis of network traffic for detecting different security attacks [7]. The design of Snort was initially relying on direct calls to the libpcap library functions to acquire network packets. Then, it was changed in Snort 2.9 by introducing the Data Acquisition library (DAQ), which provides an abstraction layer between Snort and the different types of network interfaces [8]. DAQ offers a variety of modes for packets acquisition that can be chosen at run time.

Figure 2 shows the packet processing architecture of Snort. In a Linux environment, the DAQ module runs on top of the Linux network stack. First, Snort acquires packets from the DAQ module by using the *Acquire* module. Next, the *Decoder* module decodes packets and builds up the Snort's main data structure (`SFSnortPacket`), which contains the information required to process the packet. Then, the Snort *preprocessors* plugins (if any) are invoked. The *Preprocessors* provide a modular way to easily extend the functionalities of Snort by developing custom plugins. The *Log* module logs the packet's information. After that, the *Detection* module compares the `SFSnortPacket` against the configured Snort rules. Finally, the *Verdict* module returns the action to be performed on the packet (e.g., accept or drop).

III. SR-Snort

SR-Snort is an extended SR-aware version of Snort. It can apply Snort rules to inner packets of SRv6 encapsulated traffic. The packet processing architecture of SR-Snort is shown in Figure 3. We extended the packet processing architecture

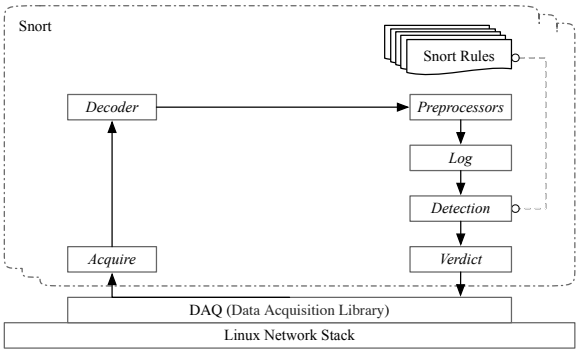

Fig. 2: Snort architecture

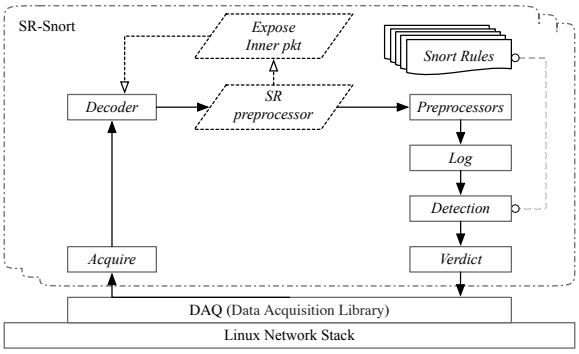

Fig. 3: SR-Snort architecture

of Snort by adding two new modules: *SR preprocessor* and *Expose Inner pkt*. The *SR preprocessor* module is invoked immediately after the *Decoder* to detect SRv6 encapsulated packets, and hence classify packets into SRv6 and non SRv6. Non SRv6 packets are processed as described in section II. SRv6 packets are encapsulated as shown in Figure 1. In order to be able to apply Snort rules to the inner packet, the SRv6 encapsulation (i.e, Outer IPv6 Header and SRH) has to be removed before invoking the *Preprocessors*, the *Log*, and the *Detection* modules. SRv6 packets are directed to the *Expose Inner pkt* module, which removes the SRv6 encapsulation from packets and feeds them to the Snort *Decoder* to build up a new `SFSnortPacket`. The new `SFSnortPacket` has the information of the inner packet and is processed by the packet processing architecture as described in section II. Accordingly, Snort rules are applied to the inner packet. SR-Snort can be included in an SRv6 policy, where it uses the same set of legacy Snort rules and applies them to the exposed inner packet. The implementation of SR-snort is open-source and publicly available on GitHub [9]. It supports inner IPv4 and IPv6 packets and can work in either IDS or IPS mode.

IV. Demo Description

The demo showcases the integration of SR-Snort in a SRv6 NFV Infrastructure deployed as a virtualized Linux environment. We run the demo on a simple network topology composed of three Linux nodes representing the Ingress, NFV, and Egress nodes of SRv6 domain. The Ingress node is configured with an SRv6 NFV policy to steer traffic from *Site A* towards *Site B* through VNF1 and VNF2, which results in having SRv6 encapsulated packets with an SRH containing segments for the two VNFs. Node 3 is configured with SRv6 decapsulation SID to remove the SRv6 encapsulation from packets as they leave the SRv6 domain. VNF1 and VNF2 run SR-Snort in IDS mode and IPS mode respectively.

Using iperf, we generate UDP traffic, with different destination port numbers (5000 and 6000 in this demo), from *Site A* destined to *Site B*, which is SR encapsulated at the Ingress node. VNF1 is configured with a snort rule to raise alert for UDP traffic with destination port number of 5000. VNF2 is configured with a snort rule to drop UDP traffic with destination port number of 6000. SR-Snort at VNF1 and VNF2

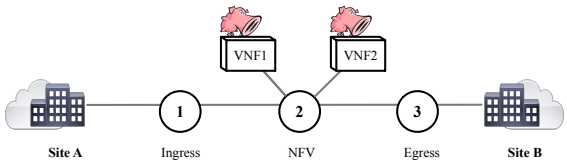

Fig. 4: SR-Snort demo

is able to apply Snort rules to the inner packets without the need to run on top of a SR-Proxy. The demo is open-source and can be replicated on any commodity hardware. Scripts and instructions to deploy the demo are available on GitHub [10].

V. Conclusions

In this paper, we have presented our implementation of SR-Snort, a SR-aware IDS/IPS able to apply a set of legacy Snort rules to the inner packets of SRv6 encapsulated traffic. We have shown the integration of SR-Snort in an SRv6 NFV policy both as IDS and IPS. We provided an open-source implementation of SR-Snort and scripts to deploy the demo.

References

[1] B. Han et al. , "Network function virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, 2015.
[2] C. Filsfils et al., "IPv6 Segment Routing Header (SRH)," June 2018. [Online]. Available: https://tools.ietf.org/html/draft-ietf-6man-segment-routing-header-14
[3] F. Clad *et al.*, "Service Programming with Segment Routing," July 2018. [Online]. Available: https://tools.ietf.org/html/draft-xuclad-spring-sr-service-programming-00
[4] A. Abdelsalam et al., "Implementation of Virtual Network Function Chaining through Segment Routing in a Linux-based NFV Infrastructure," in *3rd IEEE Conference on Network Softwarization (NetSoft 2017)*.
[5] A. Abdelsalam et al., "SERA: SEgment Routing Aware Firewall for Service Function Chaining scenarios," in *IFIP Networking 2018*.
[6] "Snort: The world's most widely deployed IPS technology." [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/security/brief_c17-733286.html
[7] M. Roesch *et al.*, "Snort: Lightweight intrusion detection for networks." in *Lisa*, vol. 99, no. 1, 1999, pp. 229–238.
[8] "Snort Users Manual." [Online]. Available: http://manual-snort-org.s3-website-us-east-1.amazonaws.com/
[9] "SR-Snort: IPv6 Segment Routing Aware Snort." [Online]. Available: https://github.com/SRouting/SR-Snort
[10] "Demo: IPv6 Segment Routing Aware Snort." [Online]. Available: https://github.com/SRouting/SR-Snort-demo