Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 27, 2020

CT. NGUYEN
M. Park
Soongsil University
November 24, 2019

A Security Architecture Against Service Function Chaining Threats
draft-nguyen-sfc-security-architecture-00

Abstract

Service Function Chaining (SFC) provides a special capability that defines an ordered list of network services as a virtual chain and makes a network more flexible and manageable. However, SFC is vulnerable to various attacks caused by compromised switches, especially the middlebox-bypass attack. In this document, we propose a security architecture that can detect not only middlebox-bypass attacks but also other incorrect forwarding actions by compromised switches. The existing solutions to protect SFC against compromised switches and middlebox-bypass attacks can only solve individual problems. The proposed architecture uses both probe-based and statistics-based methods to check the probe packets with random pre-assigned keys and collect statistics from middleboxes for detecting any abnormal actions in SFC.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

In recent years, Service Function Chaining (SFC) has emerged with the robust development of Software Defined Networking (SDN) and Network Function Virtualization (NFV). SFC defines ordered virtual chains of service functions (e.g., firewalls, load balancing, network address translation, etc.) and steers the network traffic through them, which brings many benefits from virtualized software-defined infrastructure. Service functions are provided by specialized network entities called middleboxes. One middlebox is commonly connected to a switch, and SFC connects switches to make a chain with the required services. Middleboxes are responsible for processing packet and forwarding packet to the attached switches in the service chain.

However, there are some security vulnerabilities for packets traverse in SFC, especially with compromised switches. A special attack called "middlebox-bypass attack" was proposed, which happens when compromised switches forward packets to the next-hop middlebox in the SFC without sending them to its attached middlebox. This means that packets are not processed by all service functions inside middleboxes, which does not meet the original goal of SFC. Attackers, therefore, can bypass some important service functions, e.g., firewall or IDS, and perform more attack cases. Furthermore,

compromised switches can drop, duplicate, forward incorrectly or modify the packet without notifying the controller. Packets and all network information can be sent to attackers, and all of these problems breach the policy of SFC.

Various countermeasures have been proposed to protect SFC from these attacks. They prevents the middlebox-bypass attack by adding special tags to packets in the same flow and verify these tags on every middlebox and egress switches. For the compromised switches attacks, there are two main categories of the solution: probe-based and statistics-based method. Probe-based mechanisms inject probe packet in networks and check the integrity of these packets, while statistics-based mechanisms collect and compare all of the statistics from network components to find out any abnormality. However, these solutions still have some limitations, which are described in detail in the next section.

In this document, we propose a security architecture that can simultaneously detect middlebox-bypass attacks and compromised switches in SFC. The proposed architecture uses the hybrid of probe-based and statistics-based methods, which surmounts the disadvantages of each solution above. The probe-based method uses probe packets to investigate the operation of network components in SFC. Middleboxes are programmed to handle the random pre-assigned key in the probe packet and forward back to the attached switch. If the next-hop middlebox defines incorrect handled key verification, which means the middlebox-bypass attack happened, an alarm is triggered. The statistics-based method helps the controller to find out the irregularities by monitoring every information of the packets which pass the middlebox (e.g., packet type, packet size, processing time, number of packets, etc.).

2. Terminology

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 [RFC2119].

3. Compromised Switches

The compromised switch is a serious issue for SDN in general and SFC in particular. There are many types of compromised switches attack: packet dropping, packet duplicating, packet manipulating, incorrect forwarding, eavesdropping, weight adjusting, man-in-the-middle, state-spoofing, control-channel hijacking, etc. These attacks happen when compromised switches perform some attack actions besides forwarding the packets as the commands from controllers. By

controlling the compromised switches to do one or all of these attacks, attackers can bring serious problems to the whole network.

Take the SFC chain in Figure 1 as an example. Packets in this chain should follow this path: Source Host-S1-Firewall-S1-S2-IDS-S2-S3-LB-S3-Destination Host. When compromised switch S1 receives a packet, it can drop the packet, forward the packet multiple times to S2, modify the packet, or even send that packet to an attacker, etc. S2 becomes the victim and this can ruin the operation of the network because S2 typically belongs to multiple SFC chains. Furthermore, if S2 is also compromised and confederate with S1, they can spoof information and breach all of the detecting mechanisms.

```
                  Compromised
                    Switch
   +--------+      +======+      +----+      +----+      +------+
   |  Src   |      || S1 ||      | S2 |      | S3 |      | Dst  |
   |  Host  | ---> ||    || ---> |    | ---> |    | ---> | Host |
   +--------+      +======+      +----+      +----+      +------+
                      | ^          | ^         | ^
                      | |          | |         | |
                      v |          v |         v |
                 +----------+    +-----+      +----+
                 | Firewall |    | IDS |      | LB |
                 +----------+    +-----+      +----+
```

Figure 1: Simple service function chain example with 01 compromised switch S1

Current solutions for these attacks were well investigated by other proposals. The probe-based method sends probe packets to each flow or specific switch, then checks the path and the integrity of those packets. This method can be disabled if compromised switches can recognize the probe packets and forward them as commanded. The statistics-based method tries to collect all the information from the data plane (e.g., the number of transmitted/received/dropped packets, packet type, packet size, arrived/departed time,etc.) then compares them to find out the compromised switches. This method does not support real-time detection because it needs time to gather data and only works after packets are forwarded. Moreover, packets can be forwarded without being sent to middleboxes, which bring us to the middlebox-bypass attack in the next subsection.

In this document, the proposed architecture detects compromised switches in SFC by combining probe-based and statistics-based methods. We assume that there is no collaboration between compromised switches. There are few proposed solutions to detect this type of attack but with high delay and low accuracy, or they try

to prevent this collaboration from the beginning. Most of existing solutions also try to avoid this collaboration case because it is hard to detect when compromised switches can help each other to spoof the statistics and share information.

4. Architecture Design

4.1. Methodology

The architecture detects compromised switches and middlebox-bypass attacks by sending probe packets for each SFC chain (probe-based method) and collects information from middleboxes continuously (statistics-based method). Middleboxes alert the controller whenever it receives a probe packet without a correct processed key. By monitoring every information of the packets which pass the middlebox also, the controller can find out the irregularity. The detailed architecture and detecting procedures are described in the next subsections.

4.2. Proposed Architecture

The detailed system architecture is illustrated in Figure 2. For ease of understanding, we assume a system with a single SFC chain (contains hosts, switches, and middleboxes, each middlebox connects to one switch) and a single controller. The system architecture contains 03 components as follows:

Controller: consists of 03 modules. (1) Controller Module: defines the service function chains in the network. This module installs the flow rules on switches as well as connects them to middleboxes and sends the updated network topology to Key Generator Module and Statistics Analyzing Module. (2) Key Generator Module: based on the most up-to-date network topology, this module creates and assigns new key lists to middleboxes. These key lists are used to check the integrity of probe packets in the service chains. (3) Statistics Analyzing Module: based on the most up-to-date network topology, this module analyzes statistics from middleboxes to find out abnormal actions.

Switches: follow the command from the controller to connect middleboxes to make service function chains.

Middleboxes: check every received packet from switches, record the packet information to make statistics, process the probe packet and send statistics to the controller.

```
                                +------------+
                    +---------->| Controller |<------+
                    |           +------------+       |
                    |             ^                  |
                    |             |                  |
                    v             v                  v
   +------+      +----+        +----+            +----+    +------+
   | Src  |      | S1 |        | S2 |            | Sn |    | Dst  |
   | Host | ---> |    | ---- > |    | --> ... --> |    |--->| Host |
   +------+      +----+        +----+            +----+    +------+
                   | ^           | ^                | ^
   Probe packet    | |           | |                | |
      path         v |           v |                v |
              +-----------+ +-----------+      +-----------+
              | Middlebox | | Middlebox |      | Middlebox |
              |      1    | |      2    |      |       n   |
              +-----------+ +-----------+      +-----------+
```

Figure 2: System Architecture

4.3. **Probe Packet Processing**

Probe packets are processed by middleboxes. At first, middleboxes receive pre-assigned key lists from the controller. Each middlebox only knows the compatible processed key list of the previous middlebox in the chain. By creating a key list and randomly assign different keys for each probe packet in the same flow, we reduce the probability that an attacker can guess the exact key and spoof the probe packet. Furthermore, the numerical order and the key value of the probe packet are also monitored by the controller, which restricts other guessing methods. Refreshing key lists periodically or whenever find out an abnormal action is also a solution to this problem.

Take the SFC chain in Figure 2 as an example. The packet path is Source Host-S1-Middlebox1-S1-S2-Middlebox2-S2-...-Destination Host. If we set the chain so that packets are sent from the controller and come back to the controller, compromised switches can realize this and operate like normal switches. From the beginning, Middlebox-2 receives the key list K1 = {Key_1, Key_2...} which belongs to Middlebox-1. The key list K1 contains the exact output keys that Middlebox-1 must give after processing packets. When Middlebox-2 receives a new packet from the attached switch S2, it first checks if this is the probe packet or normal packet. We use an unused bit in the header to help middleboxes recognizes the probe packet. If this is a probe packet, Middlebox-2 needs to check whether it was processed correctly or not by referring to the pre-assigned key list. For example, after receiving a probe packet with the key named Key_X,

Middlebox-2 defines the integrity of this packet by checking whether the Key_X is in the key list K1 or not. If this probe packet is correctly processed, Middlebox-2 will replace the Key_X by Key_Y, which is calculated by hash function. After this process, Middlebox-2 forwards the packet back to the attached switch (S2) to transfer to the next-hop middlebox (Middlebox-3).

If the probe packet is not correct, Middlebox-2 triggers an alarm to the controller by sending the statistics. For other packet types, the information of those packets (e.g., packet type, packet size, processing time, number of packets, etc.) is recorded to make the statistics report. Finally, Middlebox-2 sends the report to the controller and waits for new packets.

In practice, we do not need an additional method to check the integrity of the last switch in the chain. As mentioned above in subsection \ref{CS}, a switch typically belongs to multiple SFC chains, which means that it can be checked through the operation of other chains. In the case of only one chain as the example above, we run a program on the Destination Host to check the probe packets from Sn just like other middleboxes.

4.4. Statistics Checking

To detect other compromised switches attack cases (e.g., packet dropping, packet duplicating, packet manipulating, weight adjusting, etc.), the Statistics Processing Module always listens to statistics sent from middleboxes. The statistics contain the information of the packets which pass the middlebox (e.g., the number of transmitted/received/dropped packet, packet type, packet size, processing time, arrived/departed time, alert signal raised by middleboxes in probe packet processing, etc.). By comparing these statistics between middleboxes and checking the alert signal, this module can detect the compromised switches and middlebox-bypass attacks.

Take the SFC chain in Figure 2 as an example again. If Middlebox-1 reports that it forwarded 100 packets to S1 (75 normal packets and 25 probe packets) in a period (calculated by the controller) so that Middlebox-2 should report that it also received 100 packets with the same number of normal and probe packet in the same period. We set a threshold for the difference of statistics (because of packet processing latency, transmission delay or other reasons). For example, if the threshold is 5\%, it means that Middlebox-2 should receive at least 95 packets in the same period. If Middlebox-2's report shows that it only gets 90 packets, this means that the switch S2 does not forward all of the packets to Middlebox-2 (missing at least 5 packets), and this can be a middlebox-bypass attack or packet

dropping attack. The Statistics Processing Module will raise an alert in this case. In another case, if Middlebox-2 reports that it received 150 packets in that period, this means that an attack is happening (packet duplicating or weight adjusting attack) and an alert is also triggered.

5. Informative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

Appendix A. Acknowledgements

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2018-0-00254, SDN security technology development).

Authors' Addresses

NGUYEN CANH THANG
Department of ICMCT
Soongsil University
369, Sangdo-ro, Dongjak-gu
Seoul, Seoul 06978
Republic of Korea

Phone: +82 10 3408 0483
EMail: nct@soongsil.ac.kr

Minho Park
School of Electronic Engineering
Soongsil University
369, Sangdo-ro, Dongjak-gu
Seoul, Seoul 06978
Republic of Korea

Phone: +82 2 828 7176
EMail: mhp@ssu.ac.kr