DEPLOYMENT GUIDE

Packet Mirroring in Google Cloud Platform Using FortiGate

Table of Contents

Introduction To Google Cloud Platform (GCP)

Google Cloud Platform (GCP) is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search and YouTube. Together with a set of management tools, it provides a series of modular cloud services, including computing, data storage, data analytics, and machine learning.

Packet Mirroring in GCP

Packet mirroring can be useful in a variety of use cases, such as intrusion detection and network forensics for compliance. It can also be useful for analyzing protocols, understanding traffic patterns for specific applications, and troubleshooting some network issues. From a security perspective, using packet mirroring can capture different attack vectors.

The packet mirroring feature of GCP can be used to capture all ingress/egress traffic along with the payloads and headers. This mirrors the traffic from a network interface or a subnet in the specified virtual private cloud (VPC) and sends it to the internal load balancer specified as the destination in the packet mirroring policy. The source and the mirrored subnets must be in the same region.

When a packet mirroring policy is created, filters are used to narrow the traffic that needs to be mirrored. As of now, the supported filters are transport protocols (Transmission Control Protocol/User Datagram Protocol/Internet Control Message Protocol [TCP/UDP/ICMP]) and Internet Protocol (IP) address ranges.

FortiGate in Packet Mirror Mode Architecture

A typical installation of a FortiGate next-generation firewall (NGFW) has at least two interfaces: one internet facing and another for internal communication within GCP. Packet mirroring requires its own dedicated interface, so organizations need to launch FortiGate with three network interfaces in their own VPCs. If there is not a dedicated interface for packet mirroring, a routing/mirroring loop would need to be created. A typical deployment for packet mirroring is shown in Figure 1.

In this architecture, any VPC whose traffic needs to be inspected by the FortiGate would be peered to the traffic-mirror-internal VPC. The VPC called "peer1west" is peered to the traffic-mirror-internal. To mirror the traffic between the hosts in the peer1west VPC, this VPC also needs to be peered with traffic-mirror-interface VPC.

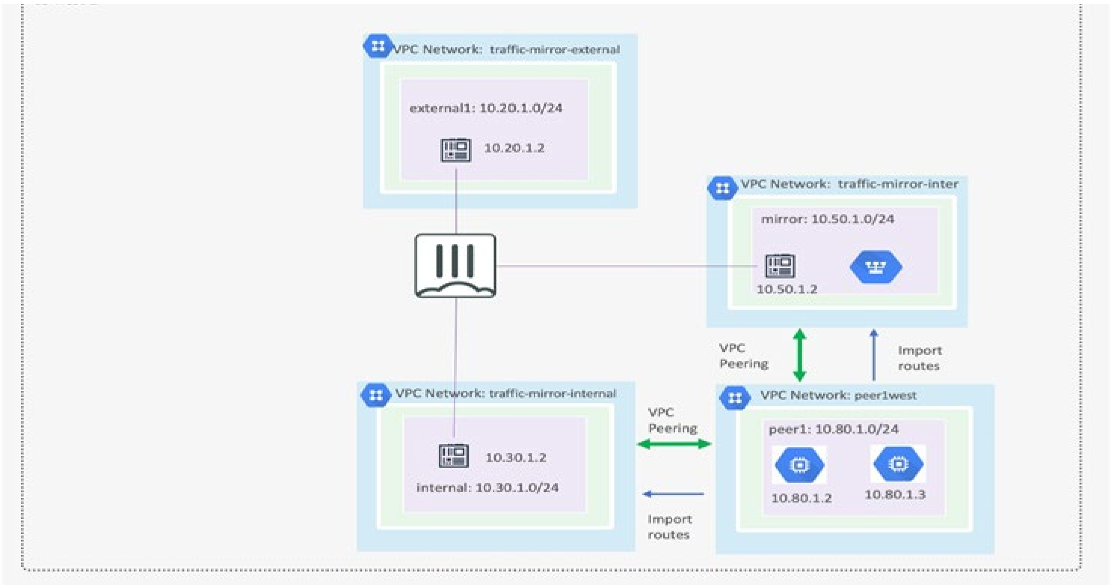


Figure 1: "peer1west" peered to the traffic-mirror-internal.

In the following example, the traffic between the 10.80.1.2 and 10.80.1.3 will be mirrored. The traffic would be mirrored onto the internal load-balancer in traffic-mirror-interface VPC.

1. **Mirroring traffic within the subnet (E-W)**

   In the setup shown in Figure 1, the traffic between the hosts in the peer1west VPC will be mirrored on the FortiGate without causing any mirroring or routing loops as the traffic between these internal hosts will not traverse through the FortiGate. If there are multiple VPCs or subnets whose traffic needs to be mirrored, multiple policies can be created.

2. **Mirroring internet-bound traffic (S-N)**

   In a typical GCP deployment of FortiGate, the internet-bound traffic from the GCP resources would be traversing the FortiGate. If that traffic needed to be mirrored, even with a dedicated interface on the FortiGate, it would cause packet loss.

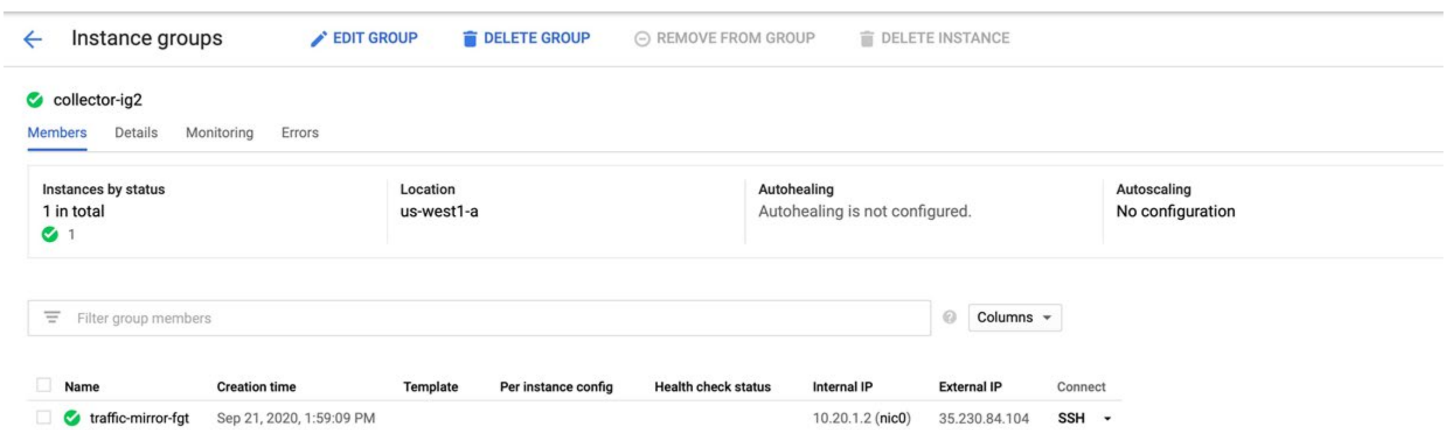
   For example, if the host 10.80.1.2 wants to access a web application on the internet, the traffic would be routed through the FortiGate interface in the traffic-mirror-internal VPC. In this example, it would be port2 of the FortiGate. If the packet mirroring policy captures this traffic, the same traffic would arrive on port3, which is the dedicated interface for packet mirroring on the FortiGate. There are no policies allowing traffic from port3 to the internet, so it would be dropped as expected. This would cause packet loss at the host at 10.80.1.2.

   To avoid this, separate routing domains for the traffic-mirror-interface and the other traffic ports on the FortiGate need to be created. This can be achieved by putting the traffic-mirror-interface in its own VRF.

Packet Mirroring Policy Configuration

1. **Instance group configuration**

   As mentioned earlier, the destination for packet mirroring is an internal load balancer. An unmanaged instance group is created, and the FortiGate instance that will be used for packet mirroring is added to the instance group. To create an unmanaged instance group: from the Google Cloud Console, the navigation is Compute engine → Instance groups.



2. Backend service configuration

A backend service is created with the instance group as the backend. When the instance group is added to this backend service, the interface needs to be specified. The backend service is created and the instance group is added through Google Cloud CLI commands. These configurations are shown in Figure 2.

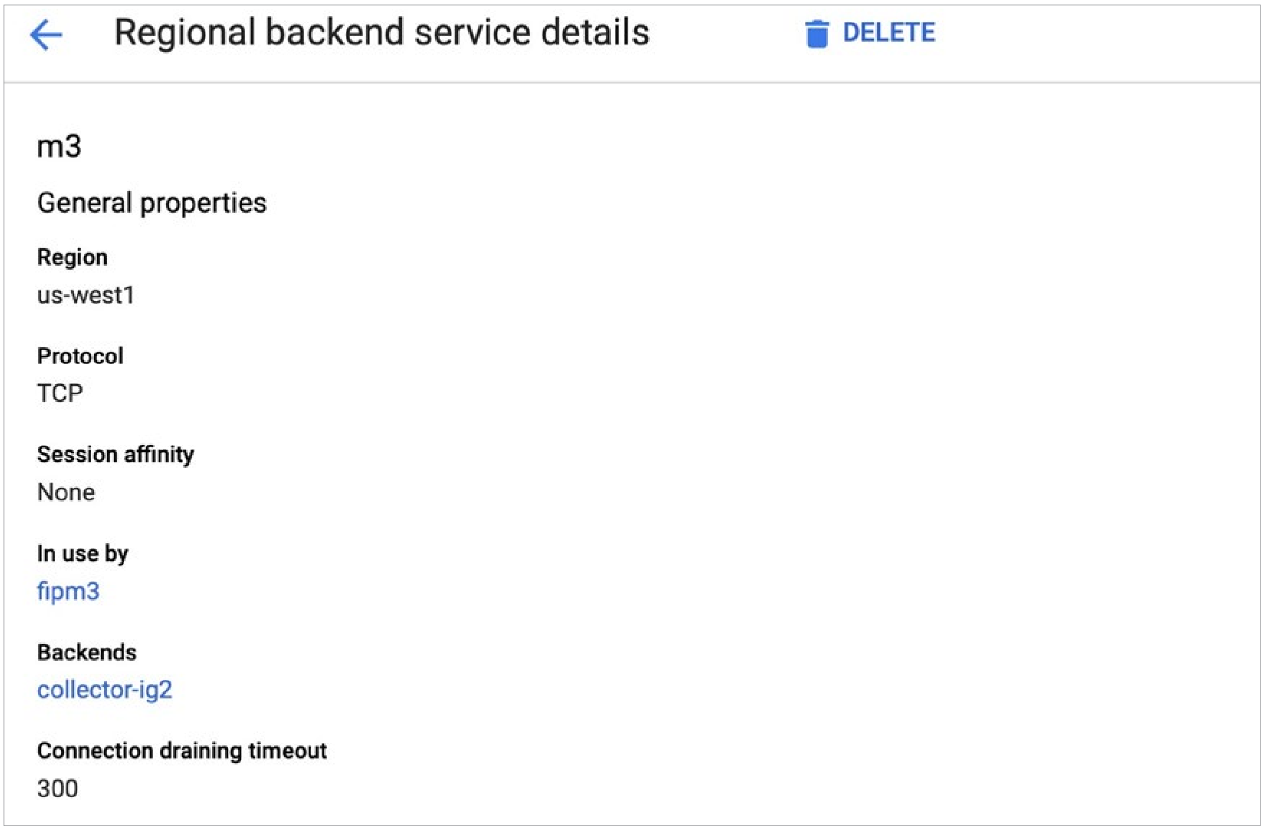


Figure 2: Backend service and instance group configurations.

3. Internal load balancer creation

The instance group and the backend service are created. Now the internal load balancer that will be used for the packet mirroring policy can also be created. The internal load balancer will be a Layer 4 (TCP) load balancer. A corresponding health check is also created to be used by the backend. In the example, a health check on TCP 22 is being used. To create an internal load balancer: from the Google Cloud Console, navigation is Network services → Load balancing.

While creating a packet mirroring policy, a collector destination is defined (an internal load balancer), which contains an unmanaged instance group. The unmanaged instance group contains the interface(s) that get the mirrored traffic. The mirrored traffic then gets forwarded to the collector instances. The frontend that is created for the internal load balancer should be enabled for packet mirroring. The settings for the frontend are shown in Figure 3.

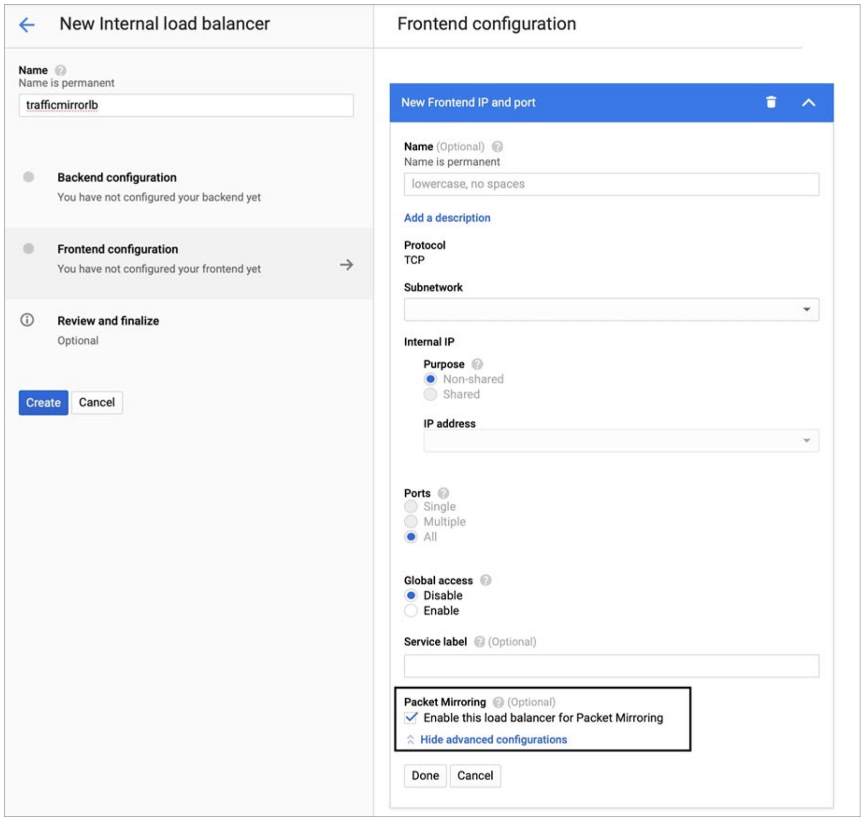
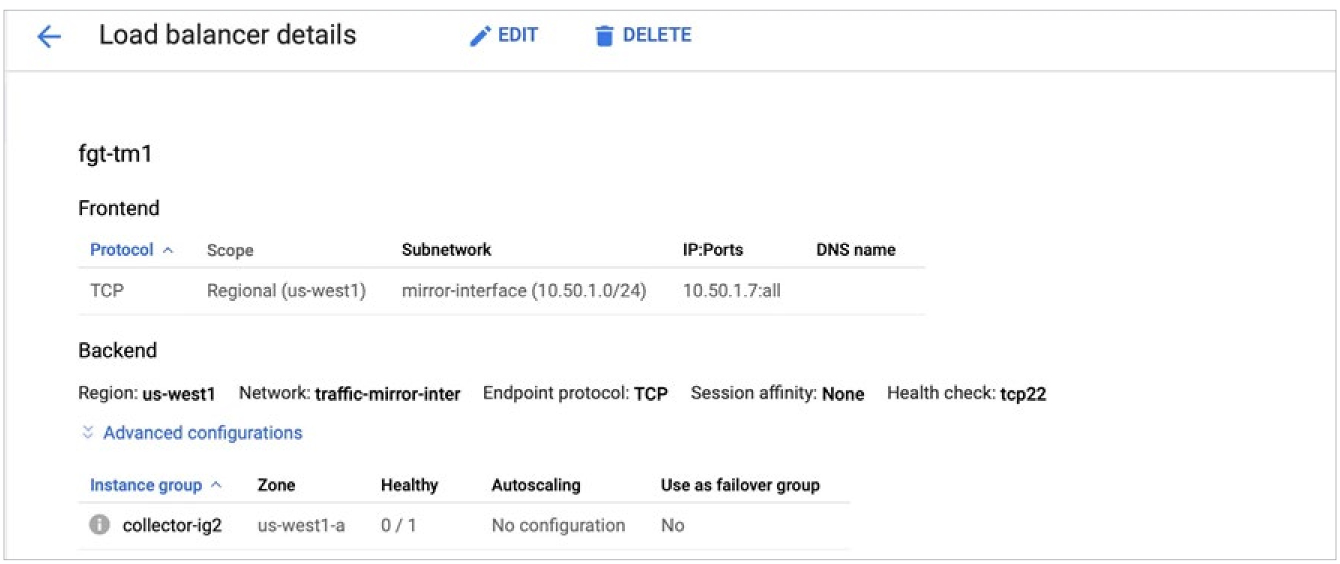


Figure 3: Frontend load balancer configuration.



4. Policy creation

The final step is to create a packet mirroring policy: from the Google Cloud Console, the navigation is VPC network → Packet mirroring. When a new policy needs to be created, a policy name, region, and priority need to be provided.

Then the mirrored source and the collector need to be selected. “Mirrored source and collector destination are in separate, peered VPC networks” needs to be chosen. The source is the peered VPC peer1west. And finally, the traffic-mirror-interface as the collector is the destination. The appropriate subnet then must be selected. There is also the choice to select individual instances. After that, the internal load balancer that was created is selected as the collector destination. In the final step, all traffic or any filters can be specified for the traffic that needs to be mirrored.

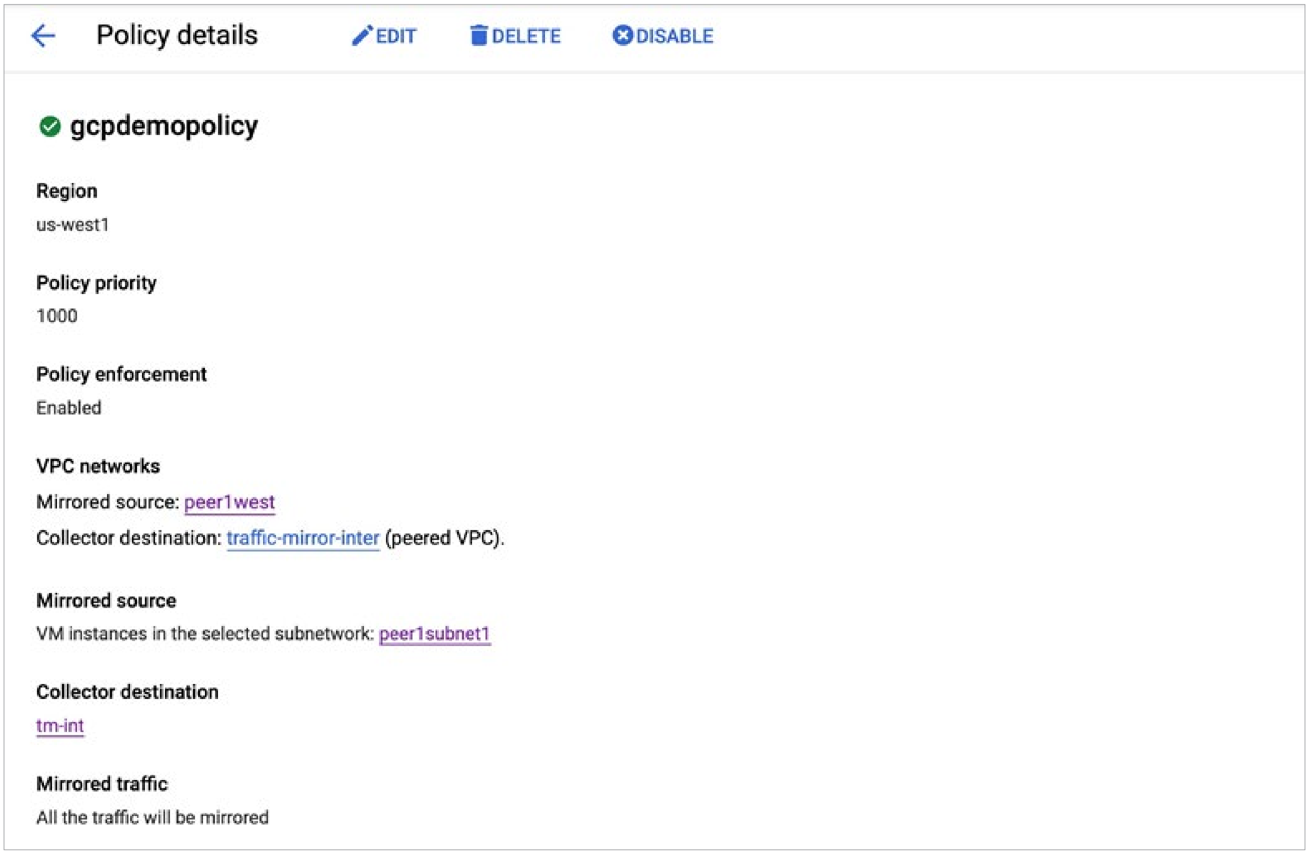


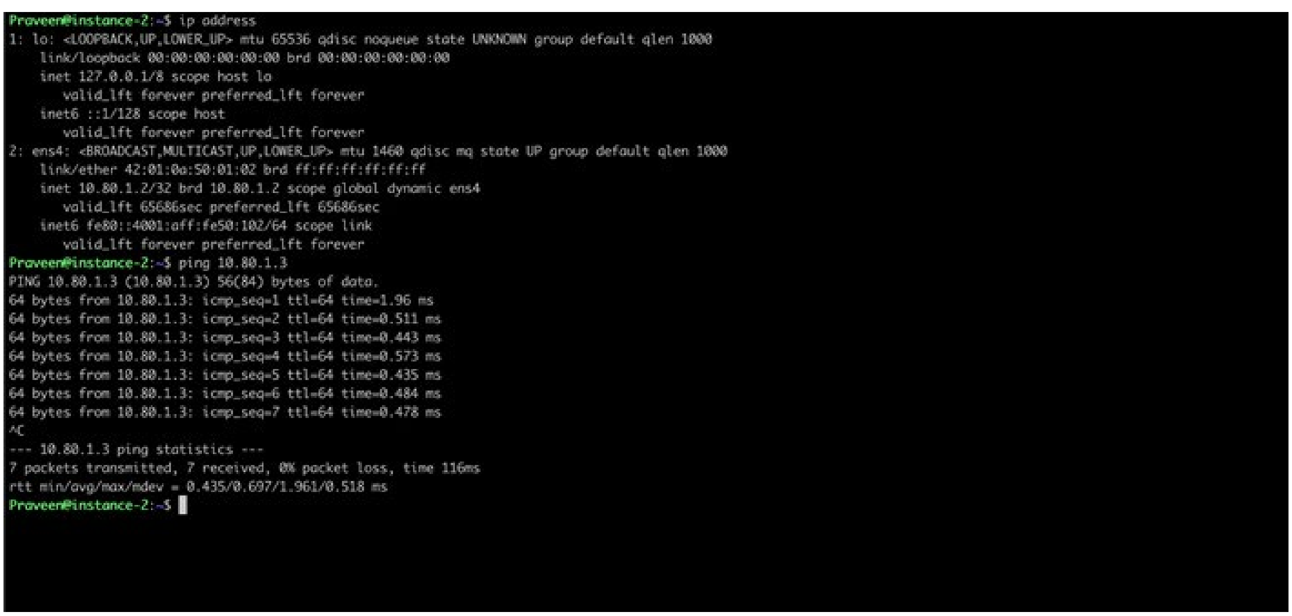
Figure 5: Example policy.

Validation

Once the packet mirroring policy is created and enabled, the FortiGate will start seeing all the traffic between the hosts in the peer1west VPC. To validate this, from the FortiGate CLI, a packet capture can be started. The following command can be executed to start a packet capture, after connecting via SSH to the FortiGate.

#diag sniffer packet any ‘icmp and host 10.80.1.2 and host 10.80.1.3’ 4 0 a

After the packet capture is started, from 10.80.1.2 start a ping should be started to 10.80.1.3.



```
Packet-Mirror # diag sniffer packet any 'icmp and host 10.80.1.2 and host 10.80.1.3' 4 0 a
interfaces=[any]
filters=[icmp and host 10.80.1.2 and host 10.80.1.3]
2020-10-27 21:37:18.172999 port3 in 10.80.1.2 -> 10.80.1.3: icmp: echo request
2020-10-27 21:37:18.173550 port3 in 10.80.1.2 -> 10.80.1.3: icmp: echo request
2020-10-27 21:37:18.173997 port3 in 10.80.1.3 -> 10.80.1.2: icmp: echo reply
2020-10-27 21:37:18.175254 port3 in 10.80.1.3 -> 10.80.1.2: icmp: echo reply
2020-10-27 21:37:19.173321 port3 in 10.80.1.2 -> 10.80.1.3: icmp: echo request
2020-10-27 21:37:19.173420 port3 in 10.80.1.2 -> 10.80.1.3: icmp: echo request
2020-10-27 21:37:19.173618 port3 in 10.80.1.3 -> 10.80.1.2: icmp: echo reply
2020-10-27 21:37:19.173634 port3 in 10.80.1.3 -> 10.80.1.2: icmp: echo reply
2020-10-27 21:37:20.193054 port3 in 10.80.1.2 -> 10.80.1.3: icmp: echo request
2020-10-27 21:37:20.193098 port3 in 10.80.1.2 -> 10.80.1.3: icmp: echo request
2020-10-27 21:37:20.193315 port3 in 10.80.1.3 -> 10.80.1.2: icmp: echo reply
2020-10-27 21:37:20.193366 port3 in 10.80.1.3 -> 10.80.1.2: icmp: echo reply
2020-10-27 21:37:21.216955 port3 in 10.80.1.2 -> 10.80.1.3: icmp: echo request
2020-10-27 21:37:21.216984 port3 in 10.80.1.2 -> 10.80.1.3: icmp: echo request
2020-10-27 21:37:21.217251 port3 in 10.80.1.3 -> 10.80.1.2: icmp: echo reply
2020-10-27 21:37:21.217304 port3 in 10.80.1.3 -> 10.80.1.2: icmp: echo reply
2020-10-27 21:37:22.240973 port3 in 10.80.1.2 -> 10.80.1.3: icmp: echo request
2020-10-27 21:37:22.241022 port3 in 10.80.1.2 -> 10.80.1.3: icmp: echo request
2020-10-27 21:37:22.241205 port3 in 10.80.1.3 -> 10.80.1.2: icmp: echo reply
2020-10-27 21:37:22.241254 port3 in 10.80.1.3 -> 10.80.1.2: icmp: echo reply
2020-10-27 21:37:23.264809 port3 in 10.80.1.2 -> 10.80.1.3: icmp: echo request
2020-10-27 21:37:23.264840 port3 in 10.80.1.2 -> 10.80.1.3: icmp: echo request
2020-10-27 21:37:23.265088 port3 in 10.80.1.3 -> 10.80.1.2: icmp: echo reply
2020-10-27 21:37:23.265095 port3 in 10.80.1.3 -> 10.80.1.2: icmp: echo reply
2020-10-27 21:37:24.288811 port3 in 10.80.1.2 -> 10.80.1.3: icmp: echo request
2020-10-27 21:37:24.288906 port3 in 10.80.1.2 -> 10.80.1.3: icmp: echo request
2020-10-27 21:37:24.289300 port3 in 10.80.1.3 -> 10.80.1.2: icmp: echo reply
2020-10-27 21:37:24.289363 port3 in 10.80.1.3 -> 10.80.1.2: icmp: echo reply
2020-10-27 21:37:25.312699 port3 in 10.80.1.2 -> 10.80.1.3: icmp: echo request
2020-10-27 21:37:25.312818 port3 in 10.80.1.2 -> 10.80.1.3: icmp: echo request
2020-10-27 21:37:25.313031 port3 in 10.80.1.3 -> 10.80.1.2: icmp: echo reply
2020-10-27 21:37:25.313069 port3 in 10.80.1.3 -> 10.80.1.2: icmp: echo reply
```

As seen in the screenshots, the ICMP packets are captured on the FortiGate on interface port3, which is the dedicated packet-mirror-interface. The packet filtering policy is working as expected.

February 15, 2021 10:38 PM

D:\Fortinet\2020 Old templates\Deployment Guides\Blue Deployment Guides\Packet Mirroring in Google Cloud\dg-FA-packet-mirroring-in-google-cloud-platform-2152021\dg-FA-packet-mirroring-in-google-cloud-platform-using-fortigate-ngfw-2152021

123456-0-0-EN