Demo: Basics of Cilium

Cilium is like a traffic inspector for the Kubernetes cluster. It checks every piece of data going in and out, decides whether it should be allowed or not based on set of the rules, and provides a deep understanding of network traffic flow. It does all of this efficiently and without disrupting normal operations.

Working of Cilium

- When a packet is sent between two pods, Cilium's data plane intercepts the packet and it inspects the packet's source and destination, examining who is trying to send data and who is meant to receive it.
- Based on the packet's source and destination, Cilium decides if the packet should be allowed to pass through by policy enforcement. This decision is based on security policies set up by the Kubernetes administrator.

---

eBPF Usage: Cilium's ability to enforce these policies is powered by eBPF (Extended Berkeley Packet Filter). eBPF is like a mini virtual machine that allows complicated policies to be run directly in the Linux kernel, the core part of the operating system.

- In addition to enforcing security policies, Cilium provides other network features that can improve the performance and reliability of a Kubernetes cluster. For example, it can help balance network load between pods and route traffic efficiently.
- Cilium also offers advanced tools for monitoring network traffic. These tools allow administrators to have a clear view of how data is flowing through the system, which can help with troubleshooting and improving performance.

Service Mesh Architecture: Cilium uses a service mesh architecture, which means it inserts eBPF programs into the Linux kernel to enable transparent policy enforcement and deep visibility into network traffic. This gives Cilium the ability to enforce identity-based policies, ensuring secure communication between services.

Hands on Cilium Lab Explained

- Star Wars Scenario: In this demo, we are creating a Star Wars-themed Kubernetes cluster. We have three microservices that represent starships from the Star Wars universe - the Death Star, the TIE fighter, and the X-wing.
- Cilium's Status: We need to ensure that Cilium, our network and security manager, is properly deployed and active.
- Deployment of Microservices: We then deploy these microservices using a specific Kubernetes YAML file. We also have a Death Star service which load-balances the traffic to all pods labelled as the Death Star.
- Deployment Verification: We check if all our pods and services have been deployed correctly, using a simple Kubernetes command.
- Endpoints in Cilium: In the world of Cilium, every pod is considered an Endpoint. We fetch all these endpoints using a specific Kubernetes command.
- Access Restrictions: Now, the rules of our story dictate that only starships labelled as the Empire can land and connect to the Death Star service. We emulate this scenario by making API calls from our TIE fighter (belonging to the Empire) to the Death Star.
- Network Access Anomalies: We discover that the X-wing (representing the Rebel Alliance) can also land on the Death Star, indicating a loophole in our security policies.
- Cilium's Identity Handling: Instead of using ever-changing IP addresses, Cilium uses the static labels assigned to the pods to define and manage security policies.
- Network Policy Creation: To resolve the issue of the X-wing landing, we create a network policy that allows only Empire starships to land on the Death Star.
- Enforcing the Network Policy: We then enforce this newly created network policy. Post this, our TIE fighter can land on the Death Star, but the X-wing's requests time out.
- Refining the Policies: But we realize that we can't trust all the TIE fighters, so we decide to further strengthen our security. We decide that we need to restrict access to certain API calls to prevent any misuse.
- Extending the Network Policy: We extend the existing policy with an HTTP rule. This rule restricts API access to only the /v1/request-landing path, thereby preventing users from accessing the /v1/exhaust-port path which can lead to the destruction of the Death Star.

- Updated Policy Application: We enforce this updated policy, and voila! Our TIE fighter can no longer access the /v1/exhaust-port path.

Real-world cilium use cases

Cilium has been adopted by a number of organizations across the industries.

- Datadog: Cilium is used by Datadog to secure and monitor the network traffic.
- Cloudflare: Cilium is used by Cloudflare to secure the network traffic of their Kubernetes-based Edge Compute platform.
- GitLab: Cilium is used by GitLab to secure the network traffic of their containerized applications running on Kubernetes.