Agenda

- Introduction
- Observability
- Monitoring
- Demo
Cilium & eBPF

Introduction
- Open Source Projects
- Company behind Cilium
- Provides Cilium Enterprise
eBPF

Makes the Linux kernel programmable in a secure and efficient way.

“What JavaScript is to the browser, eBPF is to the Linux Kernel”
Run **eBPF programs on events**

Attachment points:
- Kernel functions (kprobes)
- Userspace functions (uprobe)
- System calls
- Tracepoints
- Sockets (data level)
- Network devices (packet level)
- Network device (DMA level) [XDP]
- ...

![Diagram](image-url)
What is Cilium?

- **Networking & Load-Balancing**
  - CNI, Kubernetes Services, Multi-cluster, VM Gateway
- **Network Security**
  - Network Policy, Identity-based, Encryption
- **Observability**
  - Metrics, Flow Visibility, Service Dependency

At the foundation of Cilium is the new Linux kernel technology eBPF, which enables the dynamic insertion of powerful security, visibility, and networking control logic within Linux itself. Besides providing traditional network level security, the flexibility of BPF enables security on API and process level to secure communication within a container or pod.

[Read More](#)
<table>
<thead>
<tr>
<th>eBPF-based:</th>
<th>Foundation</th>
<th>Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Networking</td>
<td>CLOUD NATIVE COMPUTING FOUNDATION</td>
<td>eBPF, envoy</td>
</tr>
<tr>
<td>Security</td>
<td></td>
<td>eBPF</td>
</tr>
<tr>
<td>Observability</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Service Mesh &amp; Ingress</td>
<td></td>
<td>envoy</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Company</th>
<th>Use Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>WILDLIFE</td>
<td>Building a Global Multi Cluster Gaming Infrastructure with Cilium</td>
</tr>
<tr>
<td>Adobe</td>
<td>What Makes a Good Multi-tenant Kubernetes Solution</td>
</tr>
<tr>
<td>Capital One</td>
<td>Building a Secure and Maintainable PaaS</td>
</tr>
<tr>
<td>Alibaba Cloud</td>
<td>Building High-Performance Cloud-Native Pod Networks</td>
</tr>
<tr>
<td>MÁSMÓVIL</td>
<td>Scaling a Multi-Tenant k8s Cluster in a Telco</td>
</tr>
<tr>
<td>Trip.com</td>
<td>First step towards cloud native networking</td>
</tr>
<tr>
<td>GENER</td>
<td>Cloud Native Networking with eBPF</td>
</tr>
<tr>
<td>DigitalOcean</td>
<td>Managed Kubernetes: 1.5 Years of Cilium Usage at DigitalOcean</td>
</tr>
<tr>
<td>Google</td>
<td>Google chooses Cilium for Google Kubernetes Engine (GKE) networking</td>
</tr>
<tr>
<td>Bell</td>
<td>Why eBPF is changing the Telco networking space?</td>
</tr>
<tr>
<td>GitLab</td>
<td>Kubernetes Network Policies in Action with Cilium</td>
</tr>
<tr>
<td>AWS</td>
<td>AWS picks Cilium for Networking &amp; Security on EKS Anywhere</td>
</tr>
<tr>
<td>Scaleway</td>
<td>Scaleway uses Cilium as the default CNI for Kubernetes Capsule</td>
</tr>
<tr>
<td>sportradar</td>
<td>sportradar is using Cilium as their main CNI plugin in AWS (using kops)</td>
</tr>
<tr>
<td>Utmost</td>
<td>Utmost is using Cilium in all tiers of its Kubernetes ecosystem to implement zero trust</td>
</tr>
<tr>
<td>yahoo!</td>
<td>Yahoo is using Cilium for L4 North-South Load Balancing for Kubernetes Services</td>
</tr>
</tbody>
</table>
Observability
Connectivity Observability Challenges

#1 - Connectivity is layered (the “finger-pointing problem”)

Cilium
eBPF-powered Observability

Application Layer (HTTP/HTTPS)
- Presentation Layer
- Session Layer
- Transport Layer (TCP/UDP)
- Network Layer (IP)
- Data Link Layer (Ethernet)
- Physical Layer

Traditional App Layer Observability

Traditional Network Monitoring
Connectivity Observability Challenges

#2 - Application identity (the “signal-to-noise problem”)

Cilium eBPF-Powered Networking & Security

- Source Service Identity
- Destination DNS Service Identity
- Process Execution Identity
- API-call Identity
- Destination Label Service Identity

Traditional Linux Networking & Security
Where existing mechanisms fall short

- Traditional network monitoring devices
- Cloud provider network flow logs
- Linux host statistics
- Modifying application code
- Sidecar-based service meshes
Identity-based **Security & Observability**
What is **Hubble**?

**hubble UI**
- Service Dependency Maps
- Flow Display and Filtering
- Network Policy Viewer

**hubble CLI**
- Detailed Flow Visibility
- Extensive Filtering
- JSON output

**HUBBLE METRICS**
- Built-in Metrics for Operations & Application Monitoring

---

**cilium**

---

**Pod**

---

**eBPF**

---

**Pod**
Flow Visibility

```
$ kubectl get pods
NAME          READY STATUS    RESTARTS AGE
tiefighter    1/1   Running   0   2m34s
xwing         1/1   Running   0   2m34s
deadstar-5b7489bc84-crlxh 1/1   Running   0   2m34s
deadstar-5b7489bc84-j7qwq 1/1   Running   0   2m34s

$ hubble observe --follow -l class=xwing
# DNS lookup to coredns
default/xwing:41391 (ID:16092) -> kube-system/coredns-66bff467f8-28dgp:53 (ID:453) to-proxy FORWARDED (UDP)
kube-system/coredns-66bff467f8-28dgp:53 (ID:453) -> default/xwing:41391 (ID:16092) to-endpoint FORWARDED (UDP)
# ...
# Successful HTTPS request to www.disney.com
default/xwing:37836 (ID:16092) -> www.disney.com:443 (world) to-stack FORWARDED (TCP Flags: SYN)
www.disney.com:443 (world) -> default/xwing:37836 (ID:16092) to-endpoint FORWARDED (TCP Flags: SYN, ACK)
www.disney.com:443 (world) -> default/xwing:37836 (ID:16092) to-endpoint FORWARDED (TCP Flags: ACK, FIN)
default/xwing:37836 (ID:16092) -> www.disney.com:443 (world) to-stack FORWARDED (TCP Flags: RST)
# ...
# Blocked HTTP request to deathstar backend
default/xwing:49610 (ID:16092) -> default/deathstar:80 (ID:16081) Policy denied DROPPED (TCP Flags: SYN)
```
Service Map
Service identity-aware network and API-layer observability with eBPF & Cilium
HTTP Golden Signals

eBPF powered metrics without Application changes or Sidecars required:

- HTTP Request Rate
- HTTP Request Latency
- HTTP Request Response Codes / Errors
Detecting Transient Network Layer Issues

eBPF powered observability in Cilium for TCP Golden Signals:

- TCP layer bytes sent/received
- TCP layer retransmissions to measure network layer loss/congestion
- TCP round-trip-time (RTT) to indicate network layer latency
Identifying problematic API request with transparent tracing
Identifying problematic API request with transparent tracing
Monitoring
Ready to use Cilium Dashboards
https://grafana.com/orgs/isovalent/dashboards
Cilium Dashboards on Grafana

Agent Metrics

[Diagram showing various metrics such as Errors & Warnings, CPU Usage per node, Virtual Memory Bytes, Resident memory status, Open file descriptors, System-wide BPF memory usage, and BPF map pressure.]
Cilium Dashboards on Grafana

Hubble Metrics
Cilium Dashboards on Grafana

Operator Metrics
Cilium Dashboards on Grafana

Cilium Network Policy Verdict Metrics
Demo
Learn more!

For the Enterprise
Hardened, enterprise-grade eBPF-powered networking, observability, and security.

isovalent.com/product
isovalent.com/labs

OSS Community
eBPF-based Networking, Observability, Security

cilium.io
cilium.slack.com
Regular news

Base technology
The revolution in the Linux kernel, safely and efficiently extending the capabilities of the kernel.
ebpf.io
What is eBPF? - ebook
Thank you!