Point d'avancement PRIM Exploitation de Cilium et Hubble pour détecter et se protéger d'attaques DNS exfiltration

30 Mai 2024

Gilles HOPIN

Professeurs : Jean-Louis ROUGIER, Patrice NIVAGGIOLI

Sommaire

- Rappels sur Cilium
- Rappels sur Hubble
- Les metrics d'Hubble
- Les network flow logs d'Hubble
- Simulation d'une DNS exfiltration
- Détection et limite avec les metrics
- Protection avec une policy rule => restriction DNS servers
- Inspecter les domaines des queries => Exporter les network logs
- Remarque : Hubble for enterprises et Hubble data source plug-in

A propos de Cilium

Cilium est un logiciel open-source exploitant la techno eBPF pour:

- Du networking avancé (CNI, Load balancer, etc.)
- De la sécurité (Network Policies enforcement)

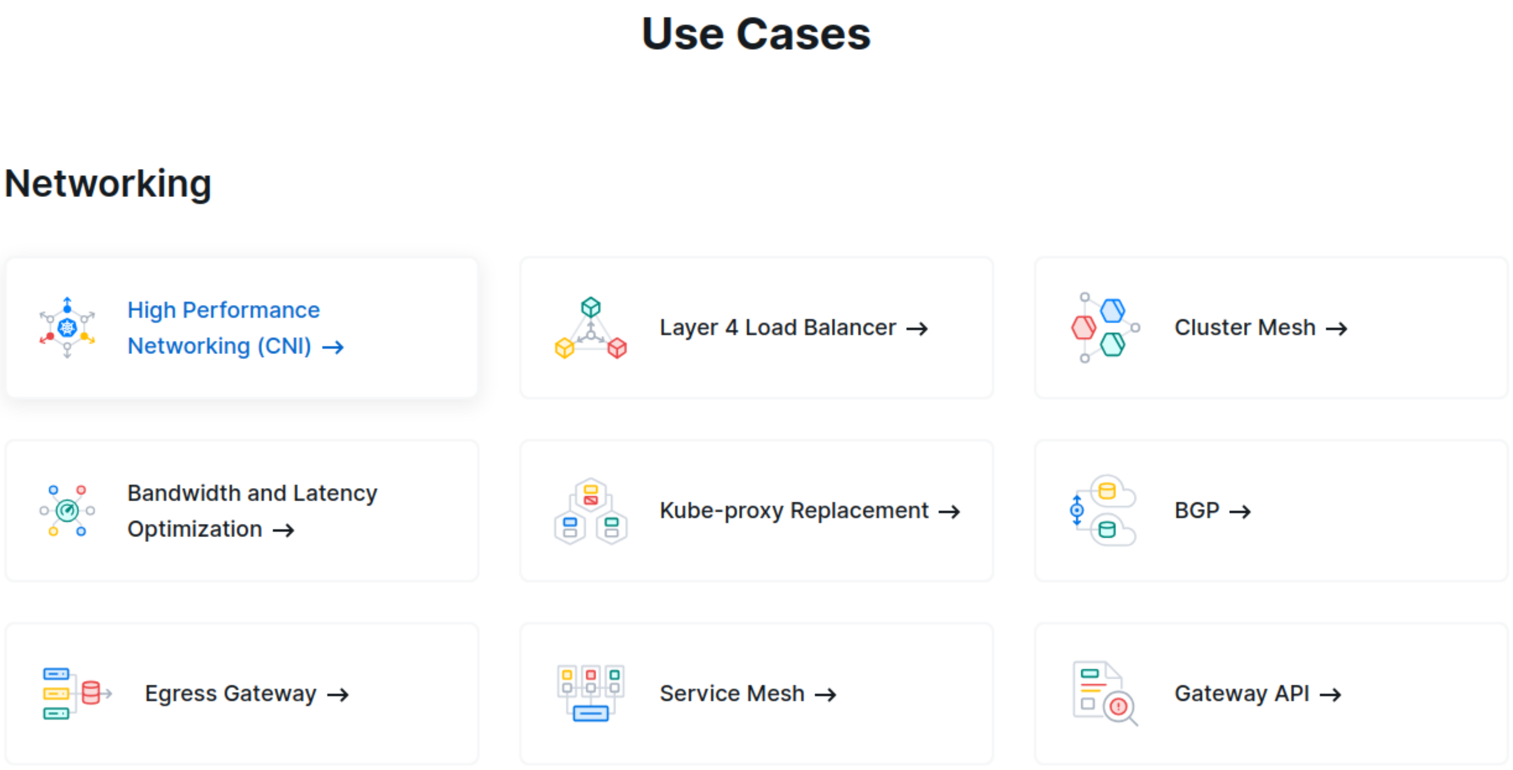
Cilium permet de déployer d'autres outils au-dessus de lui :

- Hubble : pour l'observabilité du réseau
- Tetragon : pour l'observabilité du kernel

EBPF permet :

- Peu de latence
- De la visibilité sur toute la stack : jusqu'à L7 => contextualise grandement, permet d'appliquer des règles de routing selon un label…

A propos de Cilium



Use Cases

Networking

High Performance Networking (CNI) →

Layer 4 Load Balancer →

Cluster Mesh →

Bandwidth and Latency Optimization →

Kube-proxy Replacement →

BGP →

Egress Gateway →

Service Mesh →

Gateway API →

A propos d'Hubble

Hubble is a fully distributed networking and security observability platform.

Built on top of Cilium and eBPF, so it enables :

- Deep visibility into the communication and behavior of services and the networking infrastructure
- Programmable visibility
- Contextualisation with high-level data (e.g. kubernetes' labels)

Hubble offers

- metrics collection,
- logging of network flows
- distributed tracing (integrates with OpenTelemetry)

A propos d'Hubble

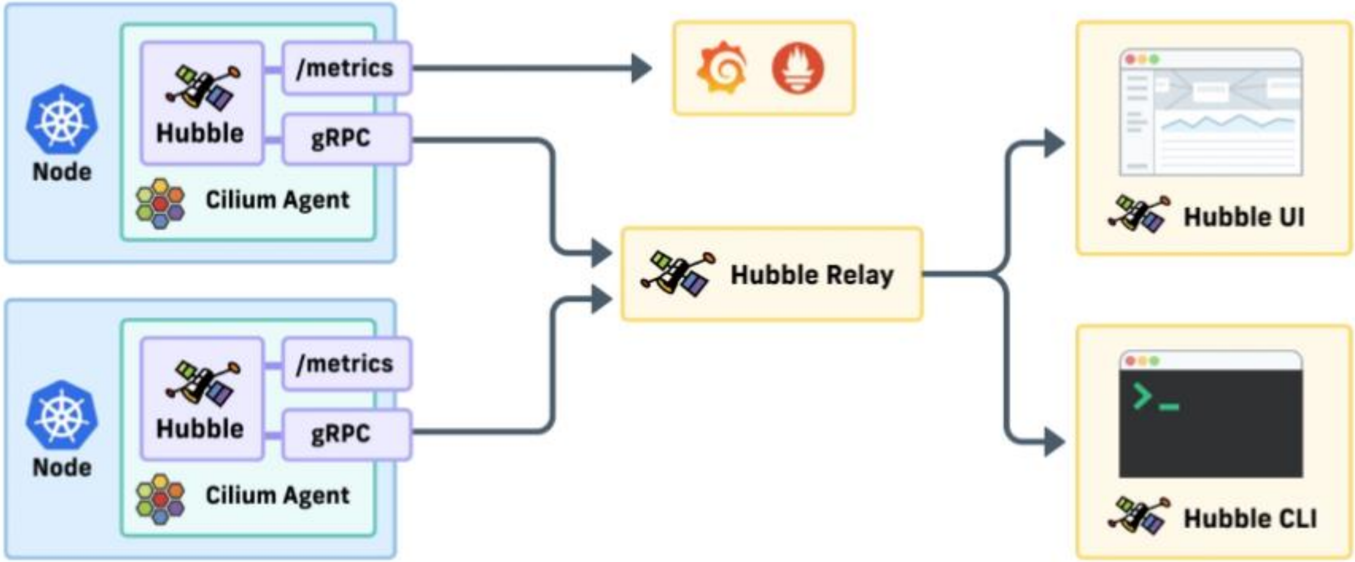
A Hubble server runs within each Cilium Agent (so one in each node)

A Hubble server exposes metrics (:9965) that can be scraped by a collector (e.g. Prometheus)

The network flow logs are gathered by Hubble Relay (deployment).

The user has 2 ways to interact with Hubble Relay :

- Hubble UI, which display the logs and a service map
- Hubble CLI, to filter the logs



A propos d'Hubble

Générer des traces ? Hubble en soi ne génère pas de traces

En revanche, il peut extraire le TraceContext d'un header HTTP et lier son traceID à son network flow et à ses metrics associées

"If your application exports tracing headers, Hubble can be configured to extract these trace IDs from http headers and export them with the Hubble HTTP metrics as Exemplars which allow us to link from metrics to traces in Grafana."

(source = https://github.com/isovalent/cilium-grafana-observability-demo/tree/main)

A propos d'Hubble

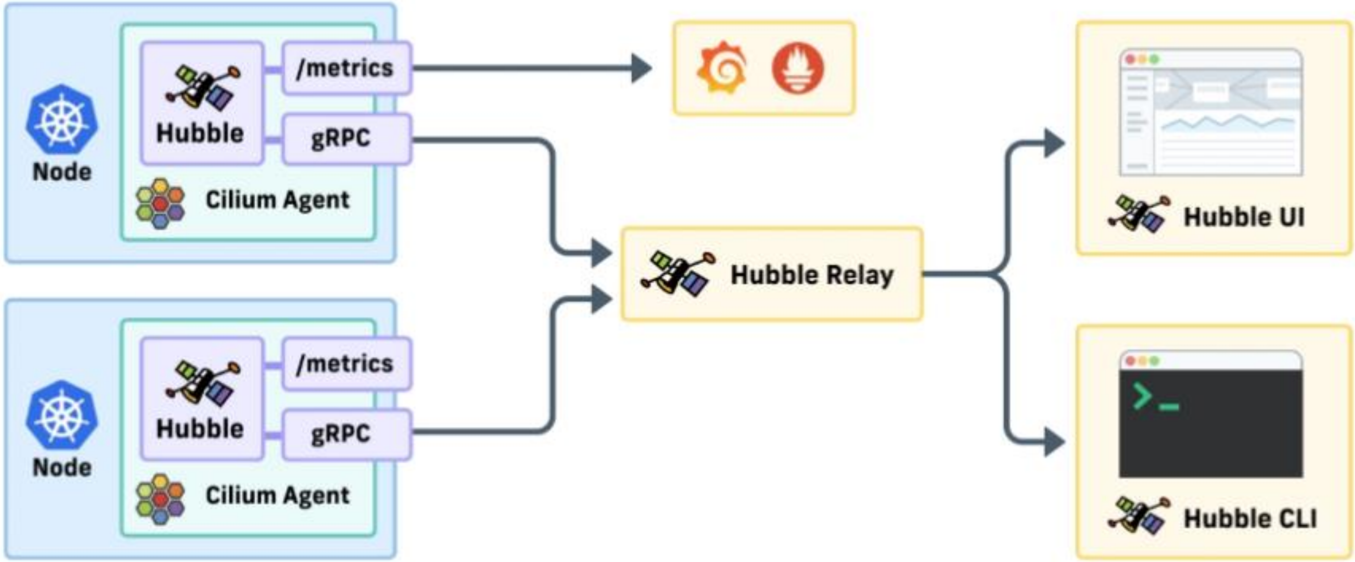
"In Cilium 1.13, Hubble’s Layer 7 HTTP visibility feature was enhanced further to automatically extract the existing, app-specific OpenTelemetry TraceContext headers from Layer 7 HTTP requests into a new field in Hubble flows. In fact, TraceContext is a specification that is now widely adopted and can be configured in Datadog and others. This allows for correlating distributed traces with detailed network-level events. If traces are stored in Grafana Tempo, the Hubble datasource plugin will automatically link Layer 7 flows to traces.

Additionally, the trace ID is included in Hubble metrics as OpenMetrics exemplars. This effectively links Hubble metrics to distributed traces, which enables engineers to quickly investigate problems. They can, for example, use Hubble metrics to define an alert on high HTTP latency, and when it fires, use exemplars to jump to a distributed trace, which highlights details of problematic requests."

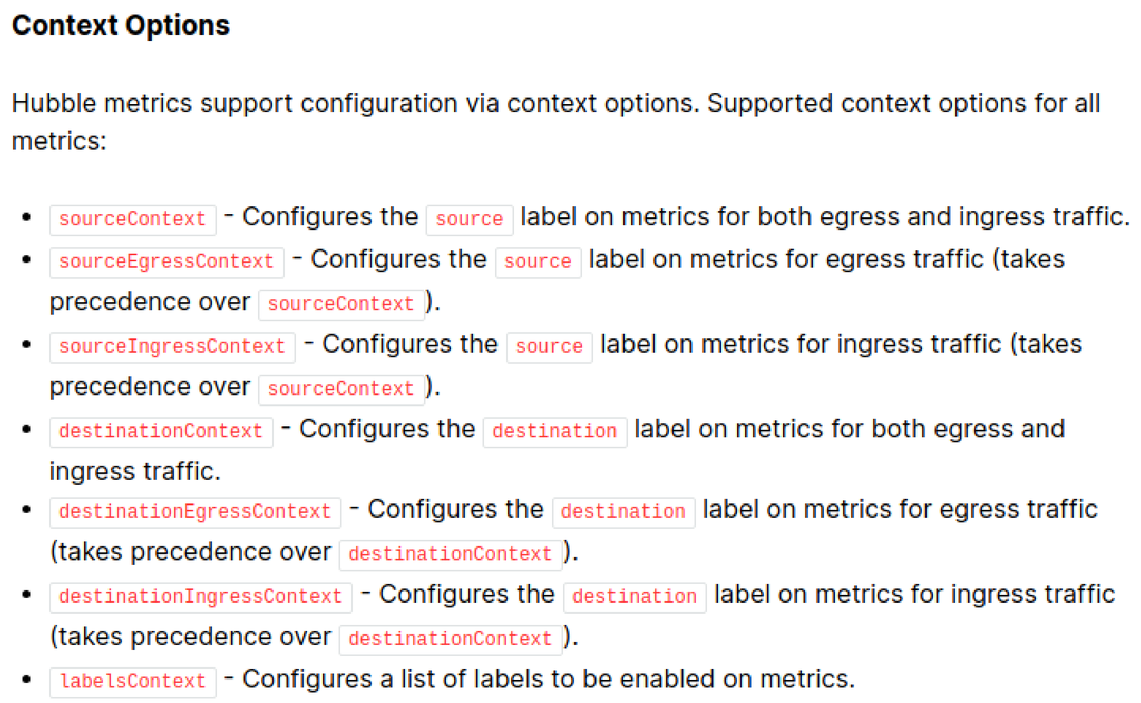
(source = https://isovalent.com/blog/post/cilium-hubble-with-grafana/)

Les metrics d'Hubble

While Cilium metrics allow you to monitor the state Cilium itself, Hubble metrics on the other hand allow you to monitor the network behavior of your Cilium-managed Kubernetes pods with respect to connectivity and security.



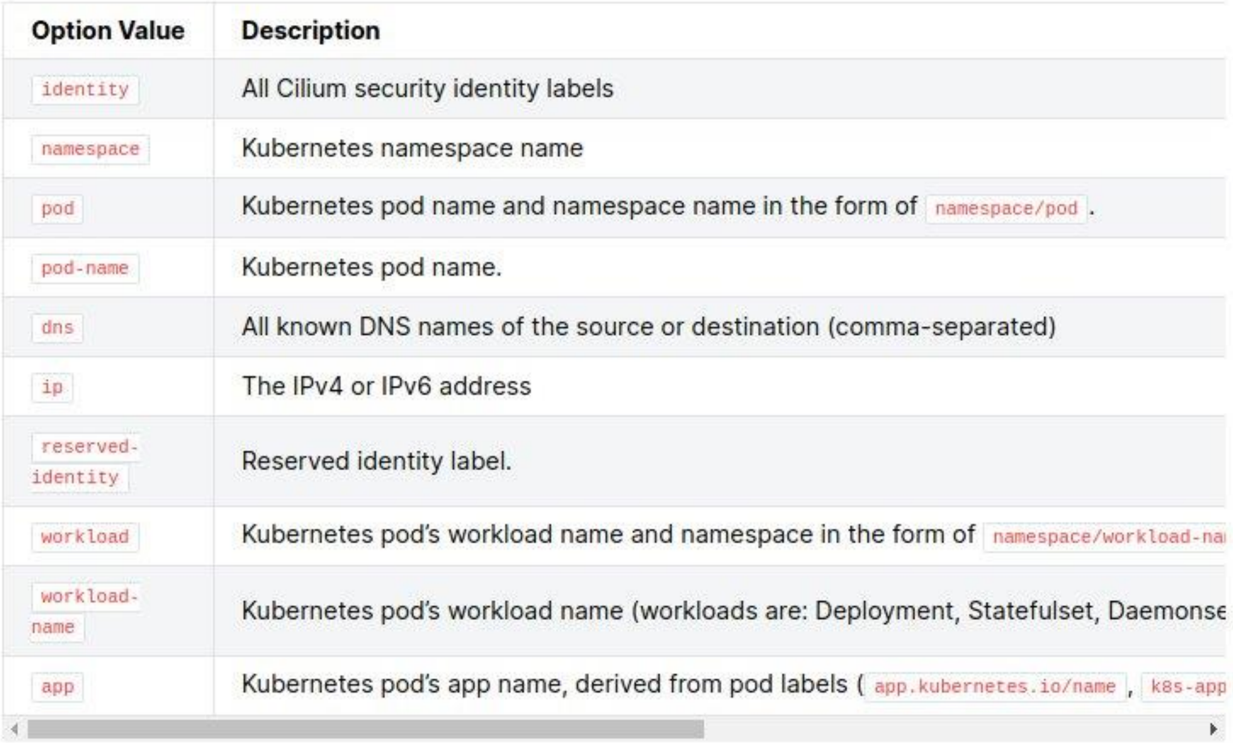
Les metrics d'Hubble



Context Options

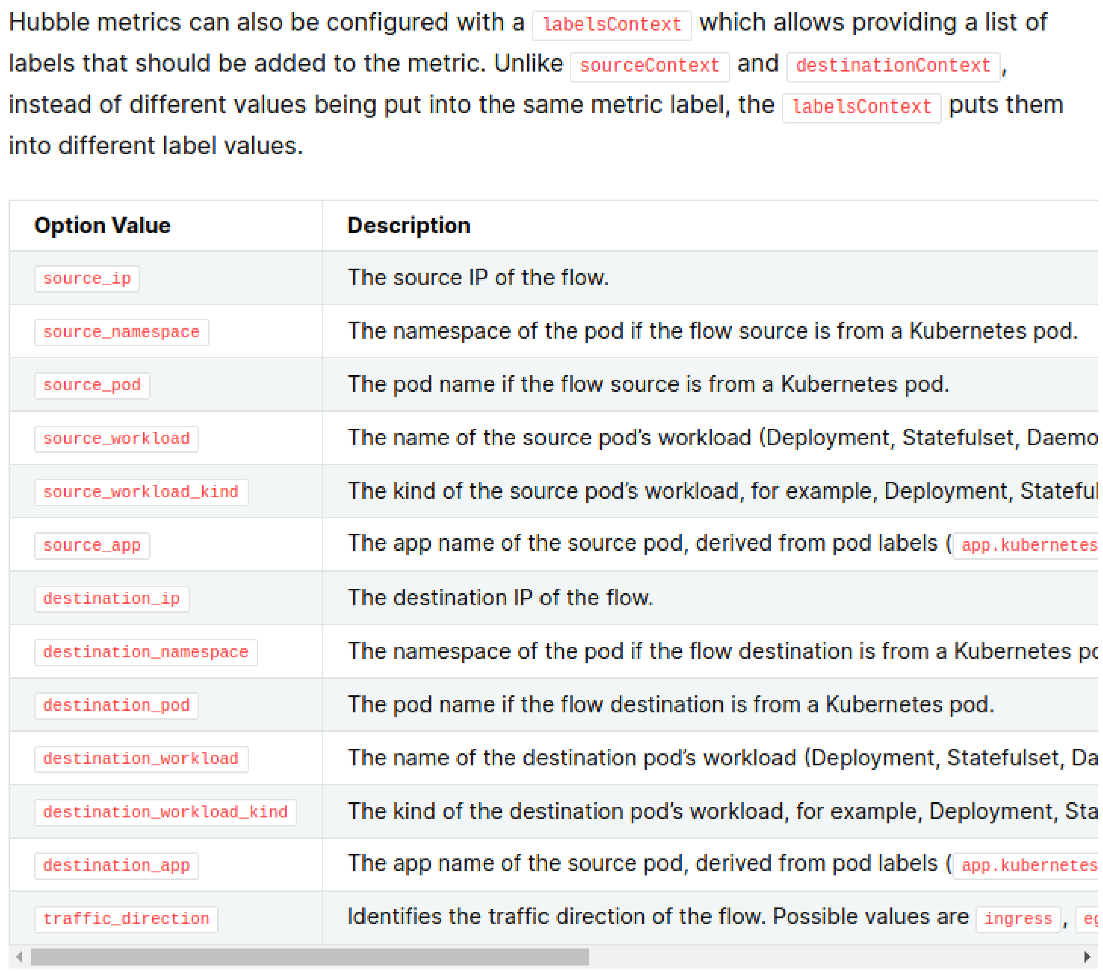
Hubble metrics support configuration via context options. Supported context options for all metrics:

- `sourceContext` - Configures the `source` label on metrics for both egress and ingress traffic.
- `sourceEgressContext` - Configures the `source` label on metrics for egress traffic (takes precedence over `sourceContext`).
- `sourceIngressContext` - Configures the `source` label on metrics for ingress traffic (takes precedence over `sourceContext`).
- `destinationContext` - Configures the `destination` label on metrics for both egress and ingress traffic.
- `destinationEgressContext` - Configures the `destination` label on metrics for egress traffic (takes precedence over `destinationContext`).
- `destinationIngressContext` - Configures the `destination` label on metrics for ingress traffic (takes precedence over `destinationContext`).
- `labelsContext` - Configures a list of labels to be enabled on metrics.



Option Value	Description
`identity`	All Cilium security identity labels
`namespace`	Kubernetes namespace name
`pod`	Kubernetes pod name and namespace name in the form of `namespace/pod`.
`pod-name`	Kubernetes pod name.
`dns`	All known DNS names of the source or destination (comma-separated)
`ip`	The IPv4 or IPv6 address
`reserved-identity`	Reserved identity label.
`workload`	Kubernetes pod's workload name and namespace in the form of `namespace/workload-na`
`workload-name`	Kubernetes pod's workload name (workloads are: Deployment, Statefulset, Daemonse
`app`	Kubernetes pod's app name, derived from pod labels (`app.kubernetes.io/name`, `k8s-app`

Les metrics d'Hubble

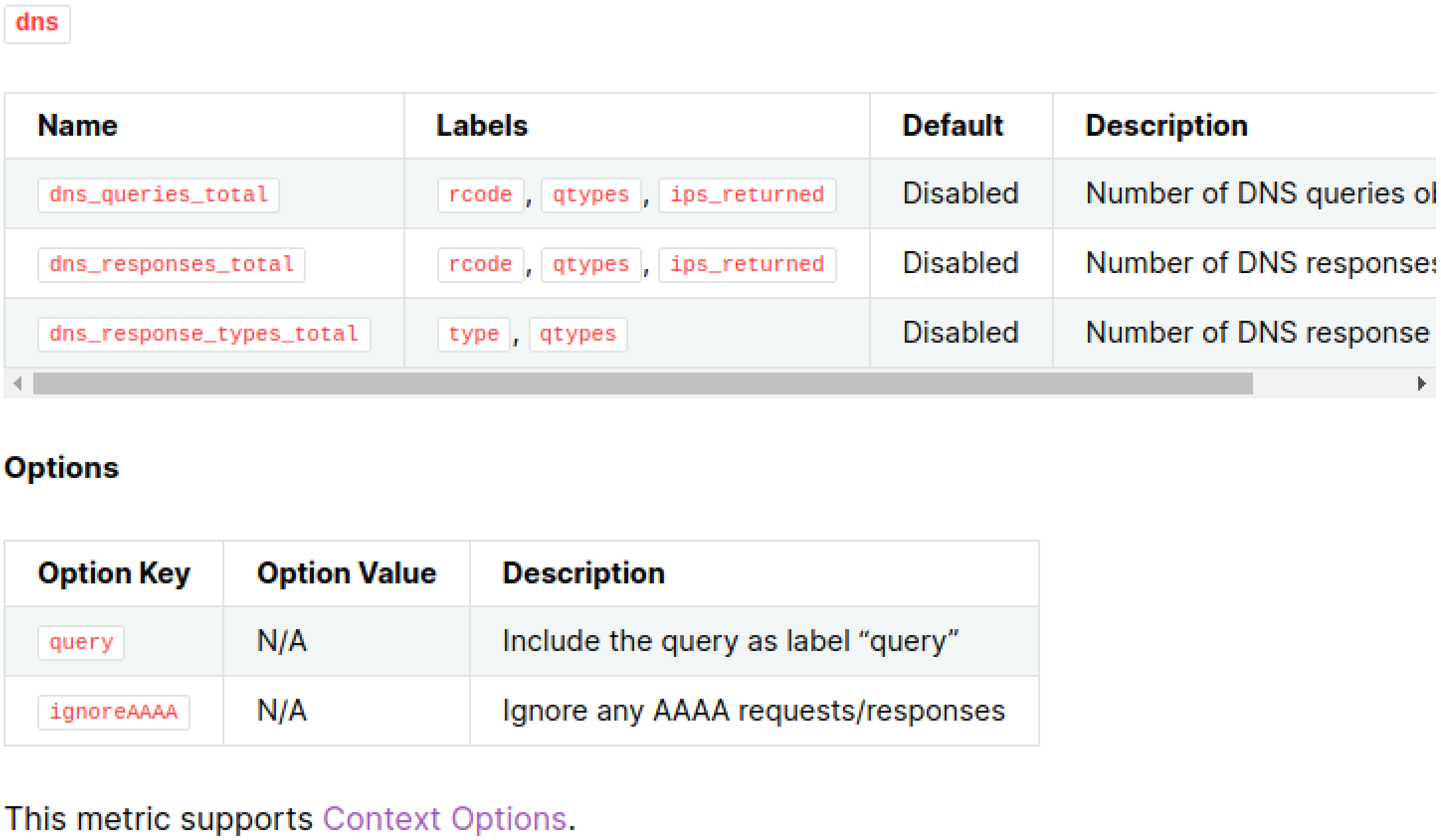


Hubble metrics can also be configured with a `labelsContext` which allows providing a list of labels that should be added to the metric. Unlike `sourceContext` and `destinationContext`, instead of different values being put into the same metric label, the `labelsContext` puts them into different label values.

Option Value	Description
`source_ip`	The source IP of the flow.
`source_namespace`	The namespace of the pod if the flow source is from a Kubernetes pod.
`source_pod`	The pod name if the flow source is from a Kubernetes pod.
`source_workload`	The name of the source pod's workload (Deployment, Statefulset, Daemo
`source_workload_kind`	The kind of the source pod's workload, for example, Deployment, Stateful
`source_app`	The app name of the source pod, derived from pod labels (`app.kubernetes`
`destination_ip`	The destination IP of the flow.
`destination_namespace`	The namespace of the pod if the flow destination is from a Kubernetes pc
`destination_pod`	The pod name if the flow destination is from a Kubernetes pod.
`destination_workload`	The name of the destination pod's workload (Deployment, Statefulset, Da
`destination_workload_kind`	The kind of the destination pod's workload, for example, Deployment, Sta
`destination_app`	The app name of the source pod, derived from pod labels (`app.kubernetes`
`traffic_direction`	Identifies the traffic direction of the flow. Possible values are `ingress`, `eg`

With labelsContext, one can add a list of labels to his metric

Les metrics d'Hubble



`dns`

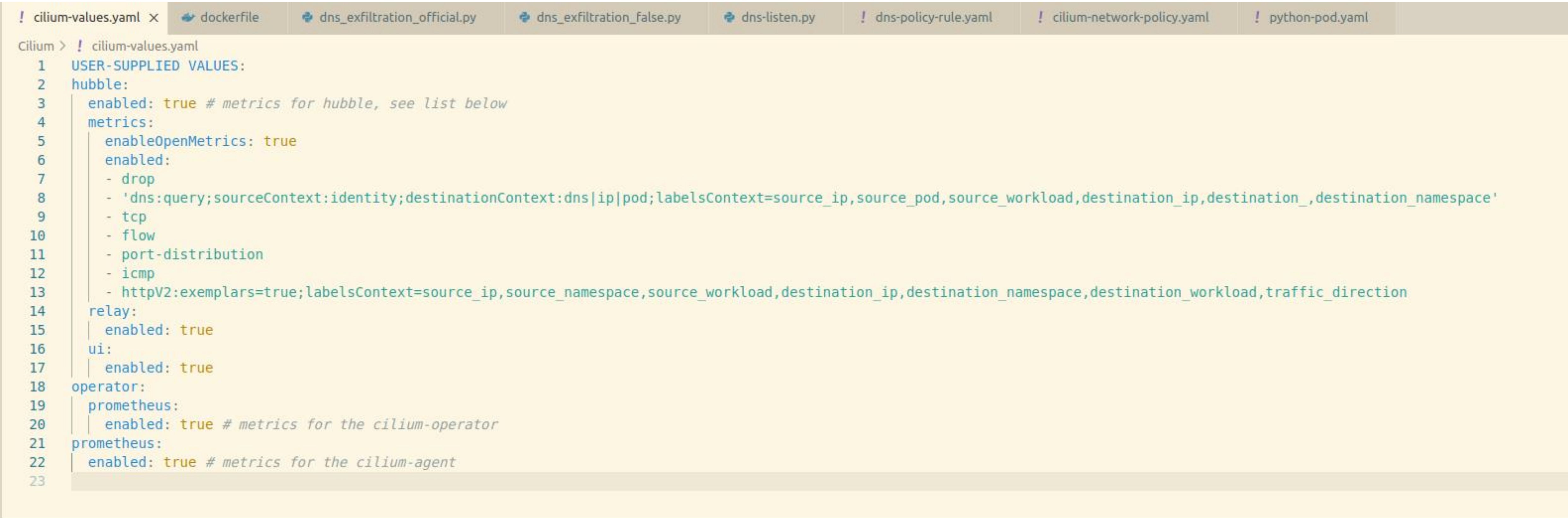
Name	Labels	Default	Description
`dns_queries_total`	`rcode`, `qtypes`, `ips_returned`	Disabled	Number of DNS queries o
`dns_responses_total`	`rcode`, `qtypes`, `ips_returned`	Disabled	Number of DNS responses
`dns_response_types_total`	`type`, `qtypes`	Disabled	Number of DNS response

Options

Option Key	Option Value	Description
`query`	N/A	Include the query as label “query”
`ignoreAAAA`	N/A	Ignore any AAAA requests/responses

This metric supports Context Options.

Les metrics d'Hubble

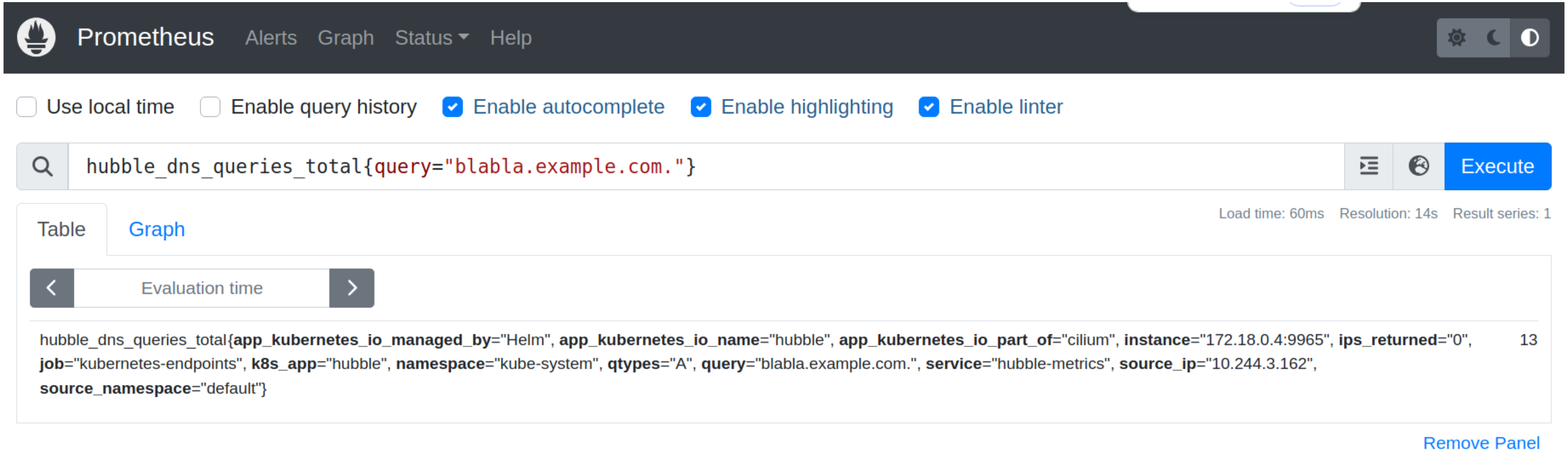

cilium-values.yaml
dockerfile
dns_exfiltration_official.py
dns_exfiltration_false.py
dns-listen.py
dns-policy-rule.yaml
cilium-network-policy.yaml
python-pod.yaml
Cilium > cilium-values.yaml

```yaml
USER-SUPPLIED VALUES:
hubble:
  enabled: true # metrics for hubble, see list below
  metrics:
    enableOpenMetrics: true
    enabled:
    - drop
    - 'dns:query;sourceContext:identity;destinationContext:dns|ip|pod;labelsContext=source_ip,source_pod,source_workload,destination_ip,destination_,destination_namespace'
    - tcp
    - flow
    - port-distribution
    - icmp
    - httpV2:exemplars=true;labelsContext=source_ip,source_namespace,source_workload,destination_ip,destination_namespace,destination_workload,traffic_direction
  relay:
    enabled: true
  ui:
    enabled: true
operator:
  prometheus:
    enabled: true # metrics for the cilium-operator
prometheus:
  enabled: true # metrics for the cilium-agent

```

CAVEAT : Utiliser Cilium 1.15.5 (doc peu explicite...)

Les metrics d'Hubble



□ Use local time □ Enable query history ☑ Enable autocomplete ☑ Enable highlighting ☑ Enable linter

```
hubble_dns_queries_total{query="blabla.example.com."}
```

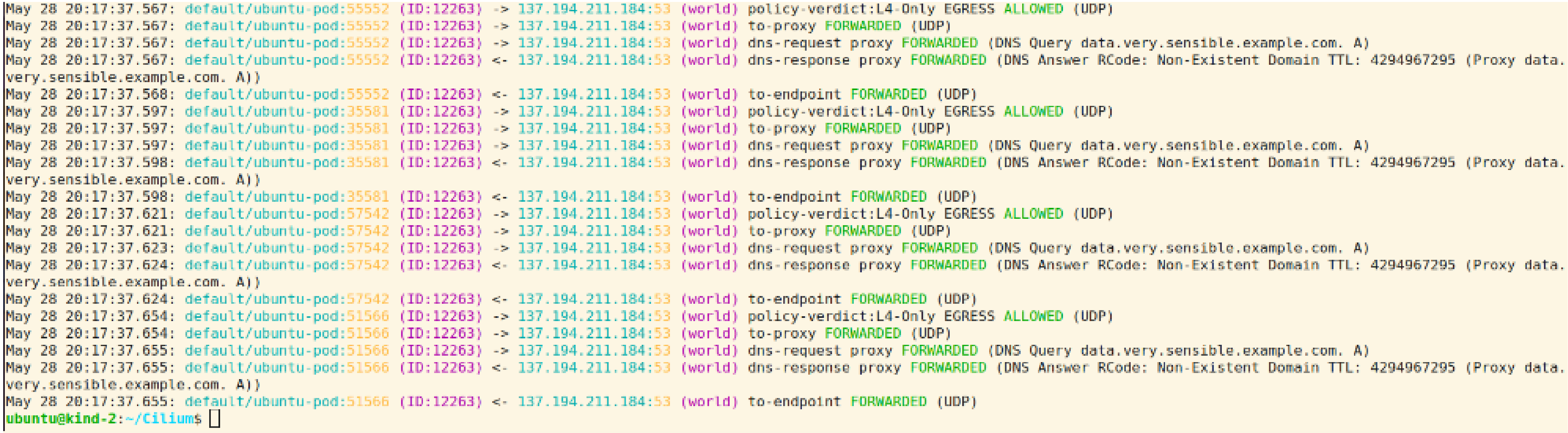
Load time: 60ms Resolution: 14s Result series: 1

```
hubble_dns_queries_total{app_kubernetes_io_managed_by="Helm", app_kubernetes_io_name="hubble", app_kubernetes_io_part_of="cilium", instance="172.18.0.4:9965", ips_returned="0", job="kubernetes-endpoints", k8s_app="hubble", namespace="kube-system", qtypes="A", query="blabla.example.com.", service="hubble-metrics", source_ip="10.244.3.162", source_namespace="default"}    13
```

Problème: pas tous les labels demandés sont remontés dans la metric

Les network flow logs d'Hubble

Commande : $hubble observe
Visibilité jusqu'à L7 ! (CAVEAT : pour permettre cette visibilité, il faut ajouter une Network Policy)



```
May 28 20:17:37.567: default/ubuntu-pod:55552 (ID:12263) -> 137.194.211.184:53 (world) policy-verdict:L4-Only EGRESS ALLOWED (UDP)
May 28 20:17:37.567: default/ubuntu-pod:55552 (ID:12263) -> 137.194.211.184:53 (world) to-proxy FORWARDED (UDP)
May 28 20:17:37.567: default/ubuntu-pod:55552 (ID:12263) -> 137.194.211.184:53 (world) dns-request proxy FORWARDED (DNS Query data.very.sensible.example.com. A)
May 28 20:17:37.567: default/ubuntu-pod:55552 (ID:12263) <- 137.194.211.184:53 (world) dns-response proxy FORWARDED (DNS Answer RCode: Non-Existent Domain TTL: 4294967295 (Proxy data.
very.sensible.example.com. A))
May 28 20:17:37.568: default/ubuntu-pod:55552 (ID:12263) <- 137.194.211.184:53 (world) to-endpoint FORWARDED (UDP)
May 28 20:17:37.597: default/ubuntu-pod:35581 (ID:12263) -> 137.194.211.184:53 (world) policy-verdict:L4-Only EGRESS ALLOWED (UDP)
May 28 20:17:37.597: default/ubuntu-pod:35581 (ID:12263) -> 137.194.211.184:53 (world) to-proxy FORWARDED (UDP)
May 28 20:17:37.597: default/ubuntu-pod:35581 (ID:12263) -> 137.194.211.184:53 (world) dns-request proxy FORWARDED (DNS Query data.very.sensible.example.com. A)
May 28 20:17:37.598: default/ubuntu-pod:35581 (ID:12263) <- 137.194.211.184:53 (world) dns-response proxy FORWARDED (DNS Answer RCode: Non-Existent Domain TTL: 4294967295 (Proxy data.
very.sensible.example.com. A))
May 28 20:17:37.598: default/ubuntu-pod:35581 (ID:12263) <- 137.194.211.184:53 (world) to-endpoint FORWARDED (UDP)
May 28 20:17:37.621: default/ubuntu-pod:57542 (ID:12263) -> 137.194.211.184:53 (world) policy-verdict:L4-Only EGRESS ALLOWED (UDP)
May 28 20:17:37.621: default/ubuntu-pod:57542 (ID:12263) -> 137.194.211.184:53 (world) to-proxy FORWARDED (UDP)
May 28 20:17:37.623: default/ubuntu-pod:57542 (ID:12263) -> 137.194.211.184:53 (world) dns-request proxy FORWARDED (DNS Query data.very.sensible.example.com. A)
May 28 20:17:37.624: default/ubuntu-pod:57542 (ID:12263) <- 137.194.211.184:53 (world) dns-response proxy FORWARDED (DNS Answer RCode: Non-Existent Domain TTL: 4294967295 (Proxy data.
very.sensible.example.com. A))
May 28 20:17:37.624: default/ubuntu-pod:57542 (ID:12263) <- 137.194.211.184:53 (world) to-endpoint FORWARDED (UDP)
May 28 20:17:37.654: default/ubuntu-pod:51566 (ID:12263) -> 137.194.211.184:53 (world) policy-verdict:L4-Only EGRESS ALLOWED (UDP)
May 28 20:17:37.654: default/ubuntu-pod:51566 (ID:12263) -> 137.194.211.184:53 (world) to-proxy FORWARDED (UDP)
May 28 20:17:37.655: default/ubuntu-pod:51566 (ID:12263) -> 137.194.211.184:53 (world) dns-request proxy FORWARDED (DNS Query data.very.sensible.example.com. A)
May 28 20:17:37.655: default/ubuntu-pod:51566 (ID:12263) <- 137.194.211.184:53 (world) dns-response proxy FORWARDED (DNS Answer RCode: Non-Existent Domain TTL: 4294967295 (Proxy data.
very.sensible.example.com. A))
May 28 20:17:37.655: default/ubuntu-pod:51566 (ID:12263) <- 137.194.211.184:53 (world) to-endpoint FORWARDED (UDP)
ubuntu@kind-2:~/Cilium$ 
```

Points restants

- Simulation d'une DNS exfiltration
- Détection et limite avec les metrics
- Protection avec une policy rule => restriction DNS servers
- Inspecter les domaines des queries => Exporter les network logs
- Remarque : Hubble for enterprises et Hubble data source plug-in