EXPLOITER CILIUM ET HUBBLE POUR DETECTER DES EXFILTRATIONS DNS

Gilles HOPIN

Professeurs : Jean-Louis ROUGIER, Patrice NIVAGGIOLI

25 juin 2024

SOMMAIRE

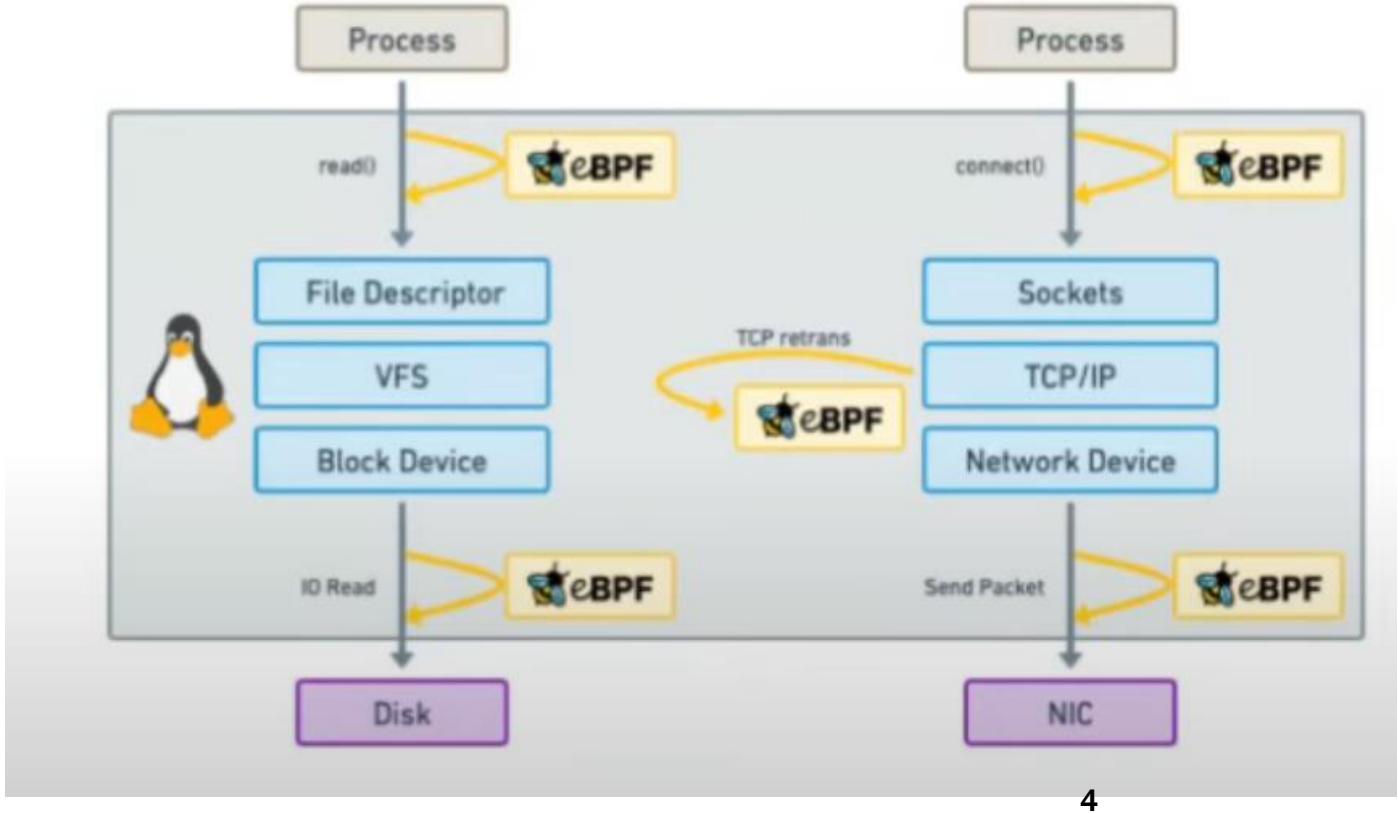
PRÉSENTATION DE CILIUM : EBPF

- Programming language and runtime to extend an os
- Embedded in the kernel => no context switches => low latency and reduced resource footprint
- "What JavaScript is to the browser, eBPF is to the Linux Kernel"
- Aimed at kernel developers => hard to learn
- All major cloud providers have picked eBPF-based Networking & Security for their Kubernetes platforms (AWS, Azure and Google Cloud)

PRÉSENTATION DE CILIUM : EBPF

Many attachment points:

- Kernel functions
- System calls
- Userspace functions
- Network devices
- ...



PRÉSENTATION DE CILIUM

Fueled by eBPF (no need to know it though)

Open source (but an Enterprise version exists)

Integrates seamlessly with Kubernetes as a CNI plugin

By itself, provides mainly :

- Advanced networking
- Security

PRÉSENTATION DE CILIUM

Advanced networking :

- IPAM, NAT
- L2 connectivty, L3 routing
- VXLAN
- Load Balancing (based on L4 or L7)
- Gateway
- ...

PRÉSENTATION DE CILIUM

Security :

- Transparent encryption of network traffic
- Network Policy enforcement :
  - At L3 and L4
  - But also to L7 : HTTP, Kafka, gRPC => API security
  - Identity-based (kubernetes' labels)
  - DNS-aware (but not DNS as a L7 protocol)

```yaml
apiVersion: cilium.io/v2
kind: CiliumNetworkPolicy
metadata:
  name: egress-policy
  namespace: endor
spec:
  endpointSelector:
    matchLabels:
      class: tiefighter
      org: empire
  egress:
  - toFQDNs:
    - matchName: disney.com
    toPorts:
    - ports:
      - port: "443"
  - toFQDNs:
    - matchName: swapi.dev
    toPorts:
    - ports:
      - port: "443"
    toEndpoints:
    - matchLabels:
        class: deathstar
        org: empire
    toPorts:
    - ports:
      - port: "80"
```

PRÉSENTATION DE CILIUM

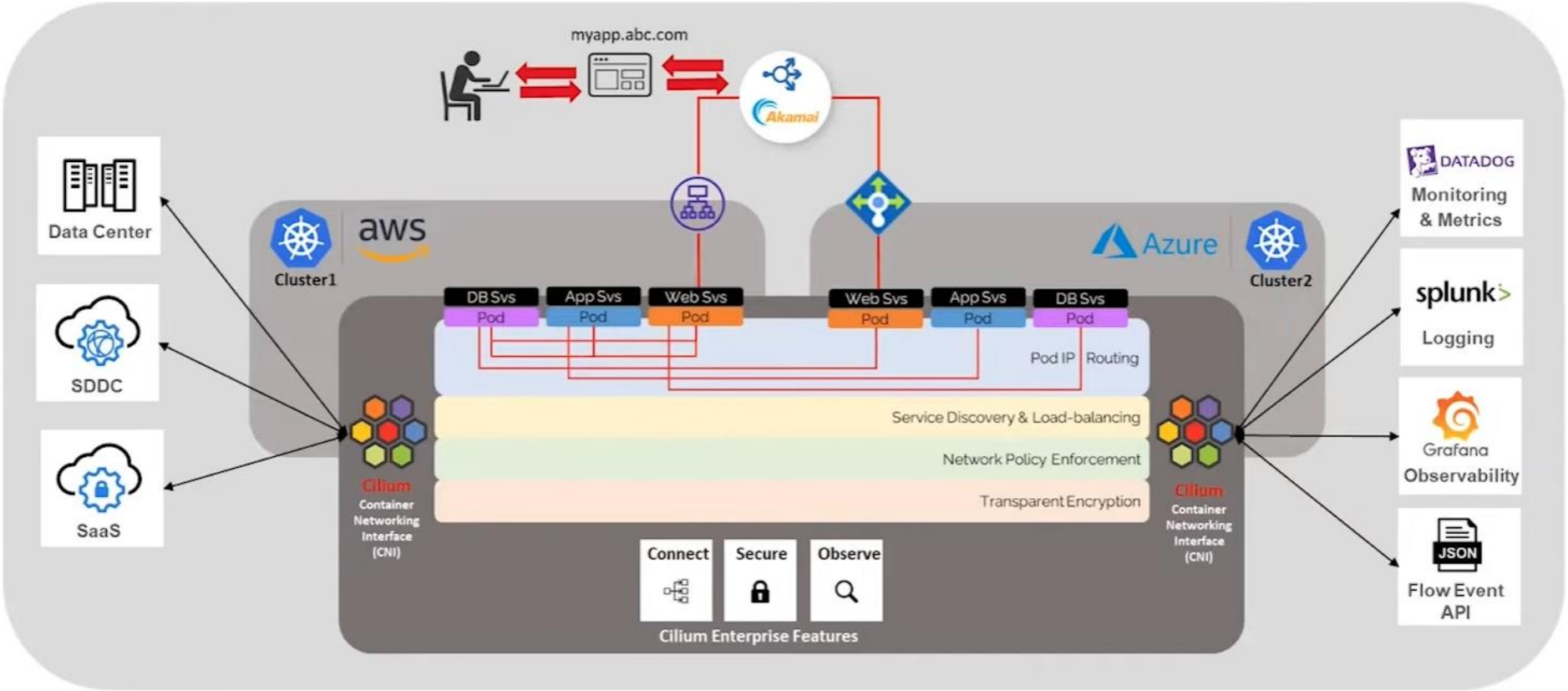
What about observability ? On top of Cilium, we can deploy :

- Hubble => observability of the network
- Tetragon => observability of the kernel

Hubble brings visibility through the stack, and adds identity-based informations

So we get a fully contextualised observability ! (more on that later)

EXAMPLE OF USE CASE OF CILIUM (AND HUBBLE)



Link : https://www.youtube.com/watch?v=6CZ_SSTqb4g

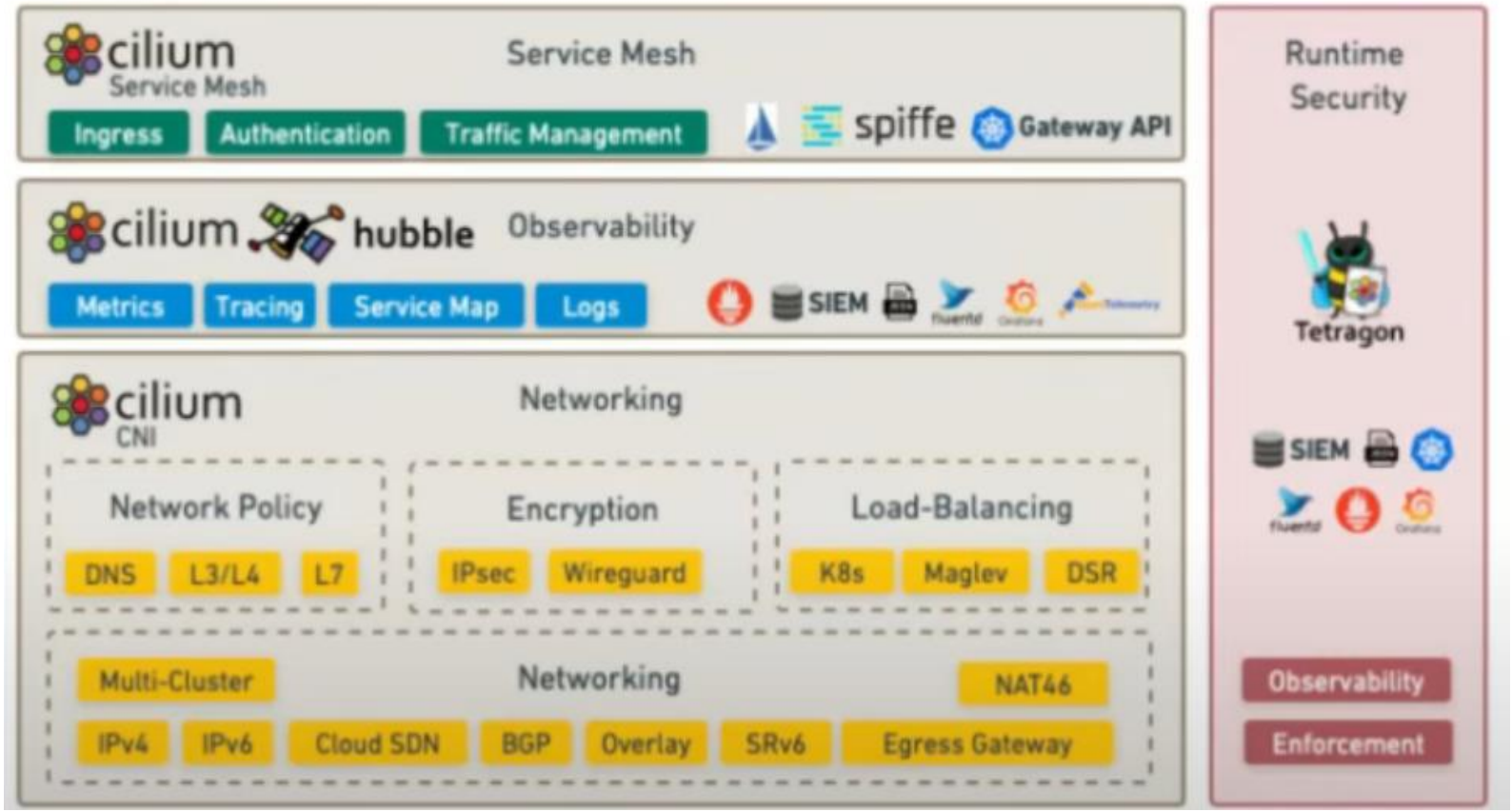
EXAMPLE OF USE CASE OF CILIUM (AND HUBBLE)

Problem Statement	Solved using Cilium CNI
Exhaustion of IP Address	Leveraged In-built VX LAN technology to overcome IP address Exhaustion
Non-standard CNI for multi-cloud use case	Simplified and Standardized the delivery of multi-cloud Kubernetes network services
Lack of Network Observability upto Layer7 stack	Leveraged well integrated Hubble UI capability to deliver visibility up to Layer7 stack
Increased application latency due to side car	Non-usage of sidecar at pod level improved the latency and enables better performance for apps within the POD

EXAMPLE OF USE CASE OF CILIUM (AND HUBBLE)

Problem Statement	Solved using Cilium CNI
Lack of granular network security policies	Ability to enforce identity-based (labels), DNS-aware (FQDN), API-aware (URL path) and Data protocol (Layer4, TCP/UDP) aware controls using eBPF program
Limited high availability capability	Leveraged cluster mesh to design apps for a global services or shared services use cases
Limited encryption capability	Ability to enforce encryption between the worker nodes or clusters using inbuilt transparent encryption (IPsec)
Lack of multi-cluster capability	Leveraged cluster mesh to address multi-cloud / multi-cluster use cases

PRÉSENTATION DE CILIUM



PRÉSENTATION D'HUBBLE

Hubble offers ...

- metrics collection,
- logging of network flows,
- distributed tracing (integrates with OpenTelemetry),
- And a service map (Hubble UI)

PRÉSENTATION D'HUBBLE

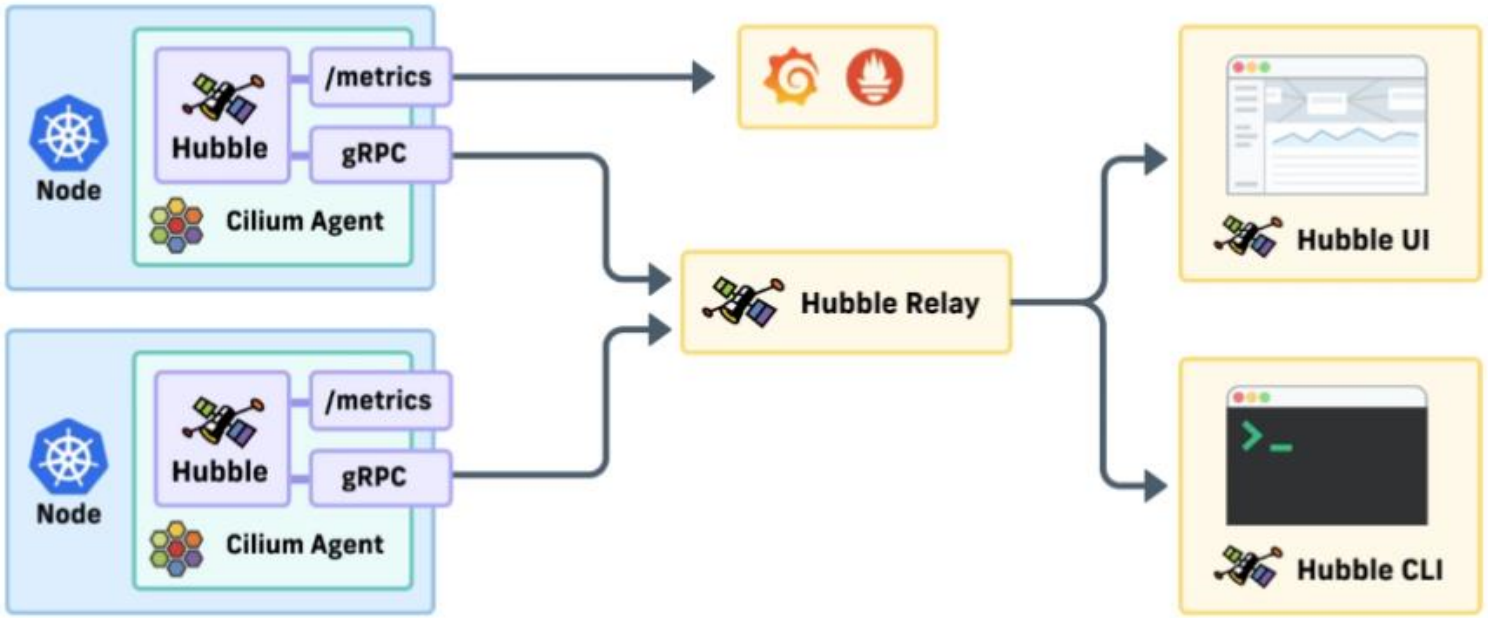
A **Hubble server** runs within each Cilium Agent (so one in each node)

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



PRÉSENTATION D'HUBBLE : LES TRACES

Hubble by itself doesn't generate traces

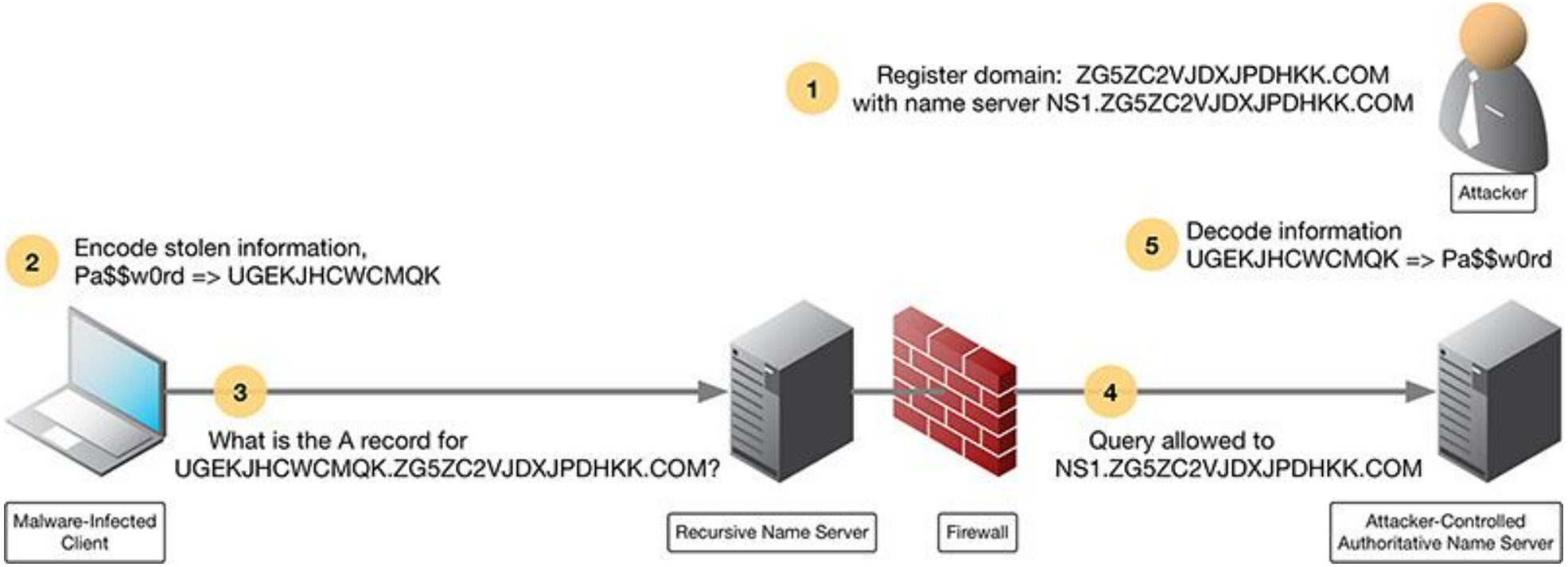
But if your app exports tracing headers :

- Hubble can extract the trace IDs from http headers
- And export them with the Hubble HTTP metrics as Exemplars

Exemplar = a way to link the metrics (aggregated view) and the traces (single request view) => useful for investigations (e.g http latency)

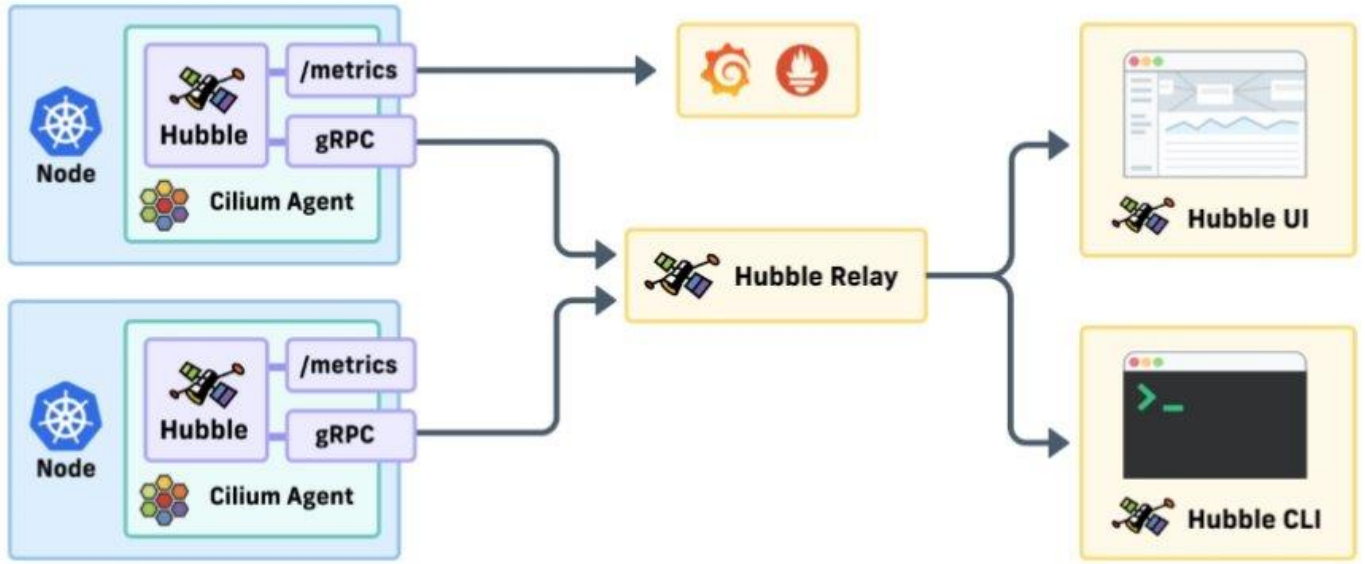
Demo : https://github.com/isovalent/cilium-grafana-observability-demo/tree/main

DETECTION D'EXFILTRATION DNS



DNS EXFILTRATION DETECTION WITH HUBBLE'S METRICS

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.



DNS EXFILTRATION DETECTION WITH HUBBLE'S METRICS

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`

DNS EXFILTRATION DETECTION WITH HUBBLE'S METRICS

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

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`

DNS EXFILTRATION DETECTION WITH HUBBLE'S METRICS

There are 3 DNS-related metrics, each with their own options availabe

`dns`

Name	Labels	Default	Description
`dns_queries_total`	`rcode`, `qtypes`, `ips_returned`	Disabled	Number of DNS queries ol
`dns_responses_total`	`rcode`, `qtypes`, `ips_returned`	Disabled	Number of DNS response
`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.

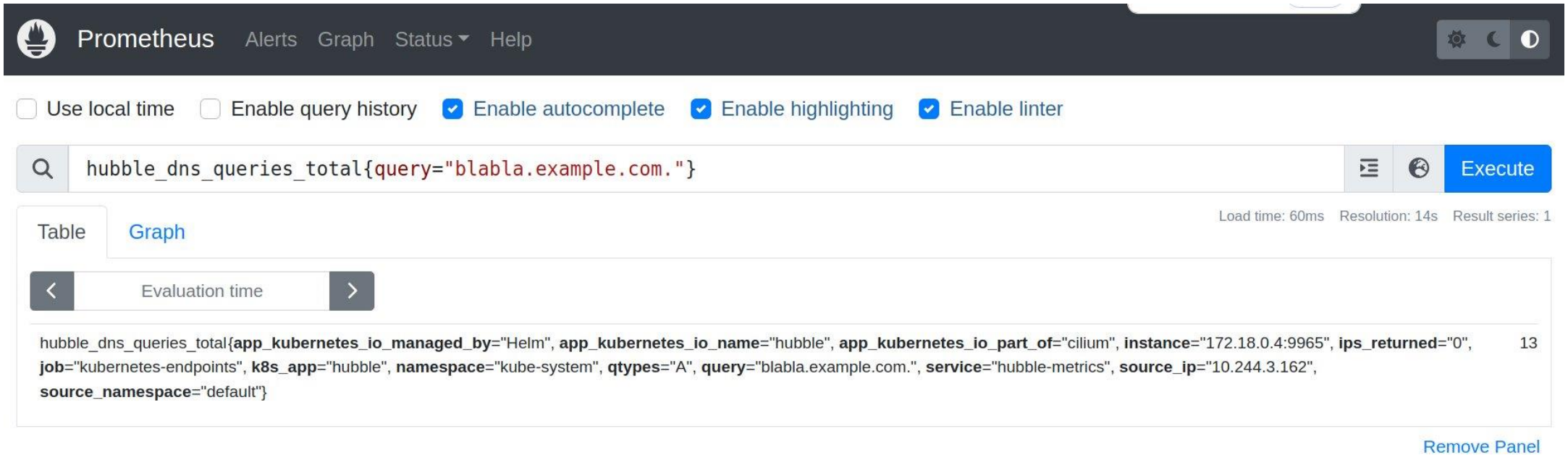
DNS EXFILTRATION DETECTION WITH HUBBLE'S METRICS

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
```

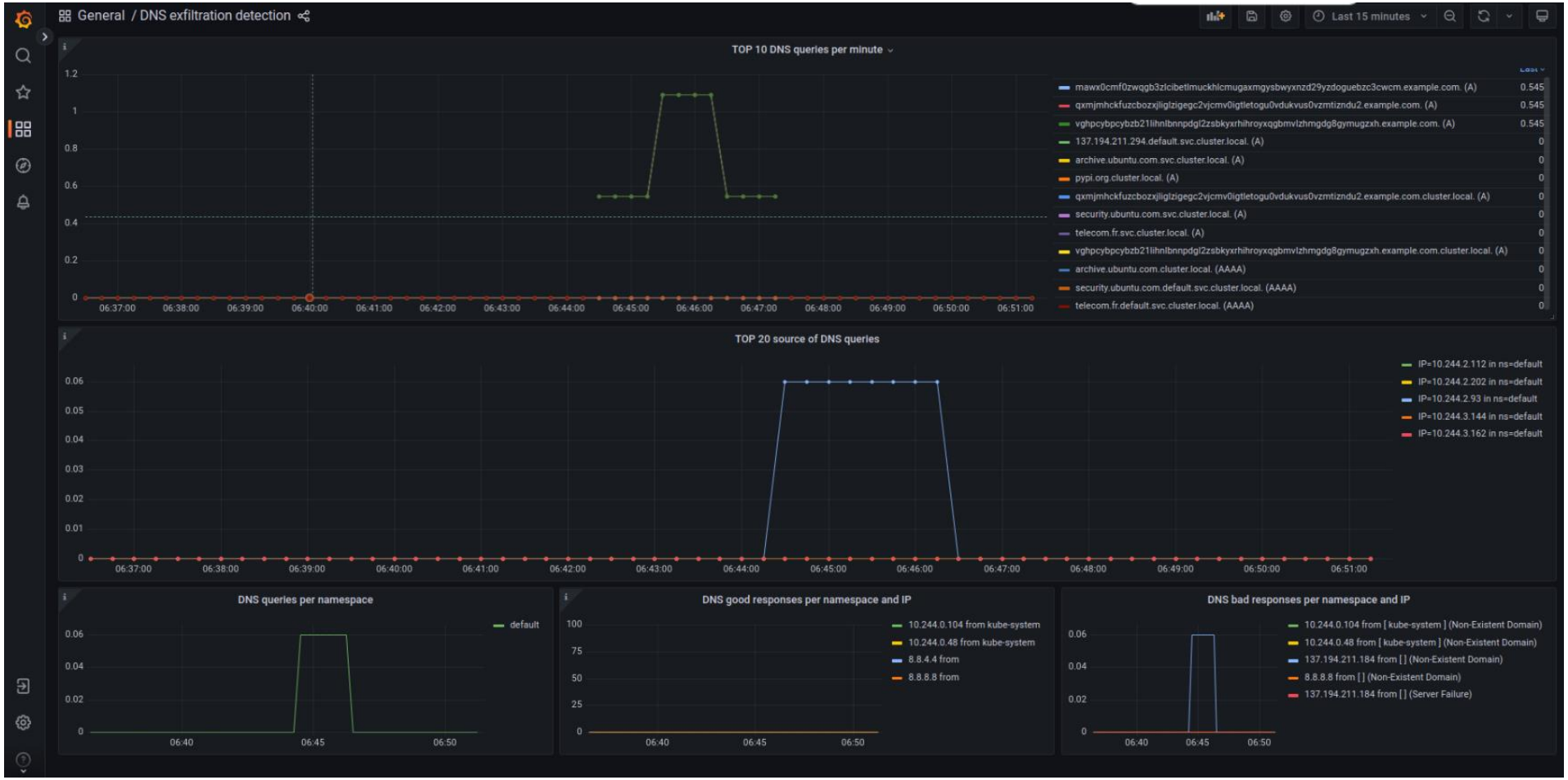
DNS EXFILTRATION DETECTION WITH HUBBLE'S METRICS



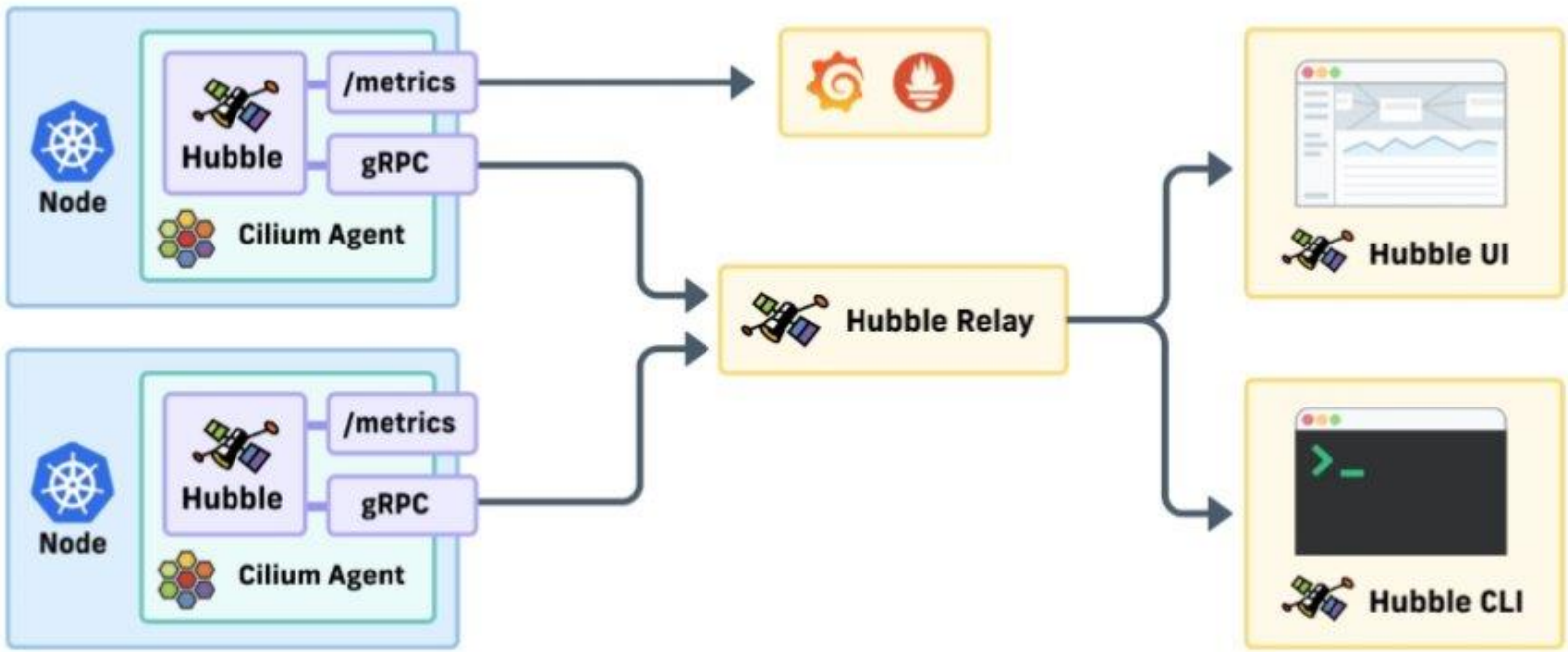
Not all labels added are here...

DNS EXFILTRATION DETECTION WITH HUBBLE'S METRICS

...but it's enough against high-troughput DNS exfiltrations thanks to Grafana dashboards !



DNS EXFILTRATION DETECTION WITH HUBBLE'S NETWORK FLOW LOGS



DNS EXFILTRATION DETECTION WITH HUBBLE'S NETWORK FLOW LOGS

Using Hubble CLI
: $hubble observe

(requires to enable L7 visibility)

```
[detached (from session 0)]
ubuntu@kind-2:~$ hubble observe --protocol dns --since=2m
Jun 24 18:30:04.344: default/ubuntu-network-tools-pod:58754 (ID:16401) -> 8.8.4.4:53 (world) dns-request proxy FORWARDED (DNS Query ecampus.paris-saclay.
fr. A)
Jun 24 18:30:04.352: default/ubuntu-network-tools-pod:58754 (ID:16401) <- 8.8.4.4:53 (world) dns-response proxy FORWARDED (DNS Answer "193.104.37.102,185
.35.173.36" CNAMEs: "prod.saclay.cblue.be.,rproxy.saclay.cblue.be." TTL: 66 (Proxy ecampus.paris-saclay.fr. A))
Jun 24 18:30:07.471: default/ubuntu-network-tools-pod:34377 (ID:16401) -> 8.8.4.4:53 (world) dns-request proxy FORWARDED (DNS Query synapses.polytechniqu
e.fr. A)
Jun 24 18:30:07.477: default/ubuntu-network-tools-pod:34377 (ID:16401) <- 8.8.4.4:53 (world) dns-response proxy FORWARDED (DNS Answer "137.194.22.227" CN
AMEs: "rp.enst.fr." TTL: 10892 (Proxy synapses.polytechnique.fr. A))
Jun 24 18:30:10.110: default/ubuntu-network-tools-pod:41164 (ID:16401) -> 8.8.4.4:53 (world) dns-request proxy FORWARDED (DNS Query colab.research.google
.com. A)
Jun 24 18:30:10.117: default/ubuntu-network-tools-pod:41164 (ID:16401) <- 8.8.4.4:53 (world) dns-response proxy FORWARDED (DNS Answer "142.250.179.110" T
TL: 300 (Proxy colab.research.google.com. A))
Jun 24 18:30:33.521: default/python-script-runner:53558 (ID:16401) -> 8.8.8.8:53 (world) dns-request proxy FORWARDED (DNS Query vghpcybpcybzb21lihnlbnnpd
gl2zsbkyxrhihroyxqgbmvlzhmgdg8gymugzxh.example.com. A)
Jun 24 18:30:33.611: default/python-script-runner:53558 (ID:16401) <- 8.8.8.8:53 (world) dns-response proxy FORWARDED (DNS Answer RCode: Non-Existent Dom
ain TTL: 4294967295 (Proxy vghpcybpcybzb21lihnlbnnpdgl2zsbkyxrhihroyxqgbmvlzhmgdg8gymugzxh.example.com. A))
Jun 24 18:30:33.614: default/python-script-runner:54825 (ID:16401) -> 8.8.8.8:53 (world) dns-request proxy FORWARDED (DNS Query mawx0cmf0zwqgb3zlcibetlmu
ckhlcmugaxmgysbwyxnzd29yzdoguebzc3cwcm.example.com. A)
Jun 24 18:30:33.702: default/python-script-runner:54825 (ID:16401) <- 8.8.8.8:53 (world) dns-response proxy FORWARDED (DNS Answer RCode: Non-Existent Dom
ain TTL: 4294967295 (Proxy mawx0cmf0zwqgb3zlcibetlmuckhlcmugaxmgysbwyxnzd29yzdoguebzc3cwcm.example.com. A))
Jun 24 18:30:33.703: default/python-script-runner:48615 (ID:16401) -> 8.8.8.8:53 (world) dns-request proxy FORWARDED (DNS Query qxmjmhckfuzcbozxjliglzige
gc2vjcmv0igtletogu0vdukvus0vzmtizndu2.example.com. A)
Jun 24 18:30:33.877: default/python-script-runner:48615 (ID:16401) <- 8.8.8.8:53 (world) dns-response proxy FORWARDED (DNS Answer RCode: Non-Existent Dom
ain TTL: 4294967295 (Proxy qxmjmhckfuzcbozxjliglzigegc2vjcmv0igtletogu0vdukvus0vzmtizndu2.example.com. A))
ubuntu@kind-2:~$
```

DNS EXFILTRATION DETECTION WITH HUBBLE'S NETWORK FLOW LOGS

Exporting Hubble's network flow logs is easy !

(Requires version 1.16.0-dev)

Other options :

- file rotation
- size limits
- filters
- field masks

values.yaml U

Cilium > values.yaml

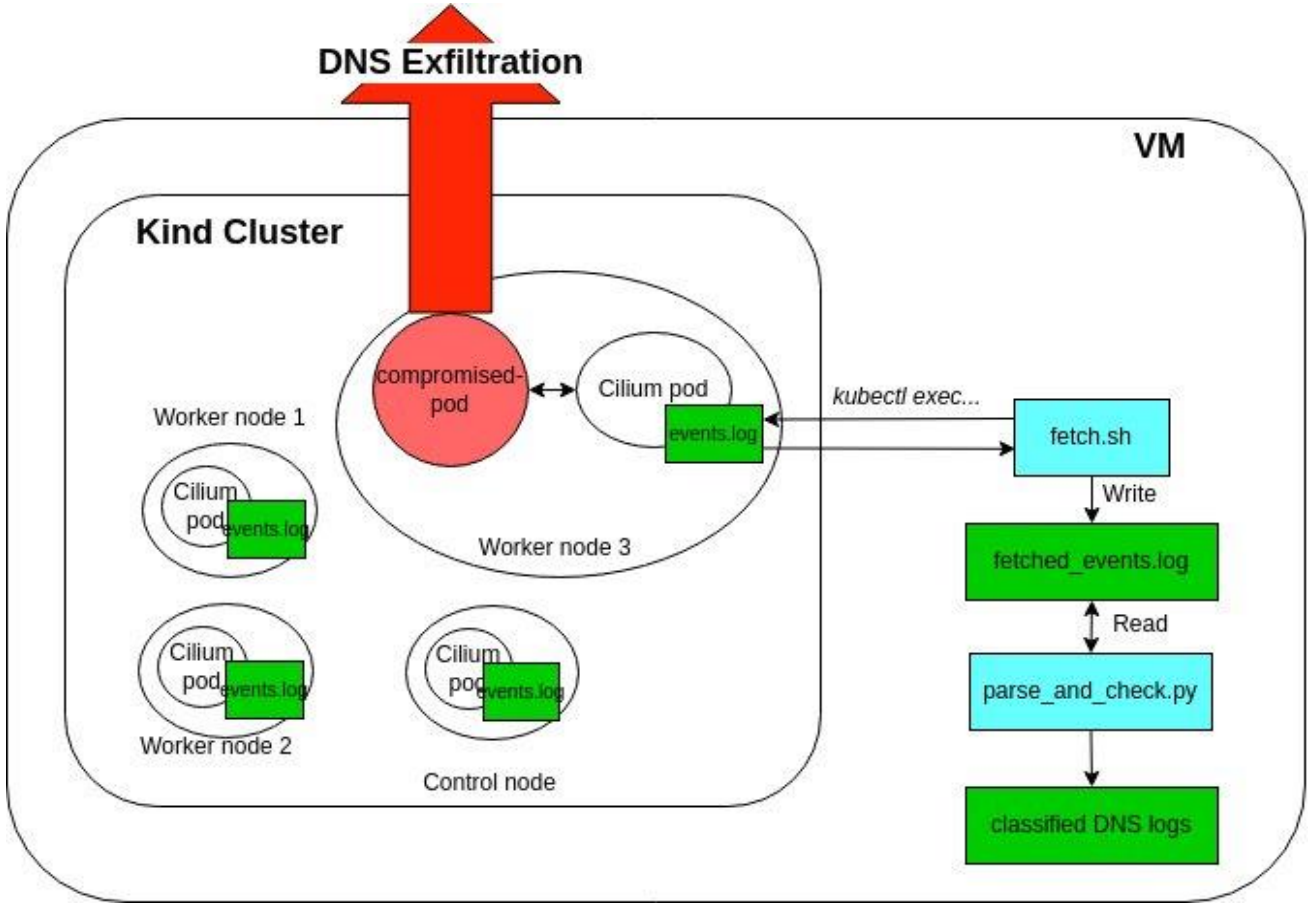
```yaml
USER-SUPPLIED VALUES:
hubble:
  enabled: true
  export:
    static:
      enabled: true
      filePath: /var/run/cilium/hubble/events.log
  relay:
    enabled: true
  ui:
    enabled: true

```

DNS EXFILTRATION DETECTION WITH HUBBLE'S NETWORK FLOW LOGS

One log file in each cilium agent => make sure to ask the right pod !

```
ubuntu@kind-2:~$ hubble observe --protocol dns  --since=30s
Jun 12 11:02:52.514: default/python-script-runner:40228 (ID:12263) -> 8.8.8.8:53 (world) dns-request proxy FORWARDED (DNS Query vghpcybpcybzb21lihnlbnnpdgl2zs
bkyxrhihroyxqgbmvlzhmgdg8gymugzxh.example.com. A)
Jun 12 11:02:52.603: default/python-script-runner:40228 (ID:12263) <- 8.8.8.8:53 (world) dns-response proxy FORWARDED (DNS Answer RCode: Non-Existent Domain T
TL: 4294967295 (Proxy vghpcybpcybzb21lihnlbnnpdgl2zsbkyxrhihroyxqgbmvlzhmgdg8gymugzxh.example.com. A))
Jun 12 11:02:52.605: default/python-script-runner:51693 (ID:12263) -> 8.8.8.8:53 (world) dns-request proxy FORWARDED (DNS Query mawx0cmf0zwqgb3zlcibetlmuckhlc
mugaxmgysbwyxnzd29yzdoguebzc3cwcm.example.com. A)
Jun 12 11:02:52.693: default/python-script-runner:51693 (ID:12263) <- 8.8.8.8:53 (world) dns-response proxy FORWARDED (DNS Answer RCode: Non-Existent Domain T
TL: 4294967295 (Proxy mawx0cmf0zwqgb3zlcibetlmuckhlcmugaxmgysbwyxnzd29yzdoguebzc3cwcm.example.com. A))
Jun 12 11:02:52.696: default/python-script-runner:45304 (ID:12263) -> 8.8.8.8:53 (world) dns-request proxy FORWARDED (DNS Query qxmjmhckfuzcbozxjliglzigegc2vj
cmv0igtletogu0vdukvus0vzmtizndu2.example.com. A)
Jun 12 11:02:52.783: default/python-script-runner:45304 (ID:12263) <- 8.8.8.8:53 (world) dns-response proxy FORWARDED (DNS Answer RCode: Non-Existent Domain T
TL: 4294967295 (Proxy qxmjmhckfuzcbozxjliglzigegc2vjcmv0igtletogu0vdukvus0vzmtizndu2.example.com. A))
ubuntu@kind-2:~$ kubectl -n kube-system exec ds/cilium -- tail -f /var/run/cilium/hubble/events.log | grep dns
Defaulted container "cilium-agent" out of: cilium-agent, config (init), mount-cgroup (init), apply-sysctl-overwrites (init), mount-bpf-fs (init), clean-cilium
-state (init), install-cni-binaries (init)
```

DNS EXFILTRATION DETECTION WITH HUBBLE'S NETWORK FLOW LOGS



Using a log aggregator ?

Attempt with Loki and Promtail => failed...

But it's easy to fetch the logs !

DNS EXFILTRATION DETECTION WITH HUBBLE'S NETWORK FLOW LOGS


```
"flow": {
  "time": "2024-06-23T21:46:19.420110061Z",
  "uuid": "78f01e2d-fa2b-4b9a-9aab-8f32148b02af",
  "verdict": "FORWARDED",
  "IP": {
    "source": "10.0.0.35",
    "destination": "8.8.4.4",
    "ipVersion": "IPv4"
  },
  "l4": {
    "UDP": {
      "source_port": 46380,
      "destination_port": 53
    }
  },
  "source": {
    "ID": 500,
    "identity": 16401,
    "namespace": "default",
    "labels": [
      "k8s:io.cilium.k8s.namespace.labels.kubernetes.io/metadata.name=default",
      "k8s:io.cilium.k8s.policy.cluster=default",
      "k8s:io.cilium.k8s.policy.serviceaccount=default",
      "k8s:io.kubernetes.pod.namespace=default"
    ],
    "pod_name": "ubuntu-network-tools-pod"
  },
  "destination": {
    "identity": 2,
    "labels": [
      "reserved:world"
    ]
  },
  "Type": "L7",
  "node_name": "kind-worker",
  "node_labels": [
    "beta.kubernetes.io/arch=amd64",
    "beta.kubernetes.io/os=linux",
    "kubernetes.io/arch=amd64",
    "kubernetes.io/hostname=kind-worker",
    "kubernetes.io/os=linux"
  ],
  "l7": {
    "type": "REQUEST",
    "dns": {
      "query": "ecampus.paris-saclay.fr.",
      "observation_source": "proxy",
      "qtypes": [
        "A"
      ]
    }
  },
  "event_type": {
    "type": 129
  },
  "traffic_direction": "EGRESS",
  "is_reply": false,
  "Summary": "DNS Query ecampus.paris-saclay.fr. A"
},
"node_name": "kind-worker",
"time": "2024-06-23T21:46:19.420110061Z"
```

DNS EXFILTRATION DETECTION WITH HUBBLE'S NETWORK FLOW LOGS

How can we tell if a DNS query is actually an exfiltration based on the log ?

Let's use Machine Learning !

DNS EXFILTRATION DETECTION WITH HUBBLE'S NETWORK FLOW LOGS

DNS exfiltration detection in the presence of adversarial attacks and modified exfiltrator behaviour

Kristijan Žiža[1] · Predrag Tadić[1] · Pavle Vuletić[1]

Published online: 8 July 2023
© The Author(s), under exclusive licence to Springer-Verlag GmbH, DE 2023

Abstract

The Domain Name System (DNS) exfiltration is an activity in which an infected device sends data to the attacker's server by encoding it in DNS request messages. Because of the frequent use of DNS exfiltration for malicious purposes, exfiltration detection gained attention from the research community which proposed several predominantly machine learning-based methods. The majority of previous studies used publicly available DNS exfiltration tools with the default configuration parameters, resulting in datasets created from DNS exfiltration requests that are usually significantly longer, have more DNS name labels, and higher character entropy than average regular DNS requests. This further led to overly optimistic detection rates. In this paper, we have explored some of the strategies an attacker could use to avoid exfiltration detection. First, we have explored the impact of DNS exfiltration tools' parameter variation on the exfiltration detection accuracy. Second, we have modified the DNSExfiltrator tool to produce exfiltration requests which have significantly lower character entropy. This approach proved to be capable of deceiving classifiers based on single DNS request features. Only around 1% of modified DNS requests shorter or equal to 9 bytes, and less than one third of DNS exfiltration requests in the overall population were accurately detected. In addition, we present a methodology and an aggregated feature set (including inter-request timing statistics) which can be used for accurate DNS exfiltration in this kind of adversarial settings.

Link : https://link.springer.com/article/10.1007/s10207-023-0723-w

DNS EXFILTRATION DETECTION WITH HUBBLE'S NETWORK FLOW LOGS

Takeaways of the article :

- The ML research applied to DNS exfiltration detection is "optimistic" (using attacks easy to spot)
- Using a simple NN model can detect most encoded DNS exfiltrations !
- For trickier DNS exfiltrations (e.g. low entropy), adding aggregated features works !
- (they provide a dataset !)

DNS EXFILTRATION DETECTION WITH HUBBLE'S NETWORK FLOW LOGS

Therefore :

- Our goal is to detect encoded DNS exfiltrations with single-based features
- The model we use is a simple Neural Network Classifier
- The single based features : request_length, #subdomain, #words (based on a list of words), word_ratio, max_word_length, entropy, digits_ratio

DNS EXFILTRATION DETECTION WITH HUBBLE'S NETWORK FLOW LOGS

About the dataset :

- 50 million entries (from a Serbian ISP's DNS server over 24h) with 22 features (query, timestamp, IP source, single-based and aggregated features)
- Only 0.5% of attacks => very imbalanced !
- 1 corrupted domain (dnsresearch.ml) + some domains considered as exfiltration (mcafee.com, e5.sk, etc.)

DNS EXFILTRATION DETECTION WITH HUBBLE'S NETWORK FLOW LOGS

```python
model3 = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(x_train1.shape[1],)),
    tf.keras.layers.Dropout(0.7),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dropout(0.7),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

model3.compile(optimizer='adam',
               loss=tf.keras.losses.BinaryFocalCrossentropy(apply_class_balancing=True, alpha=0.99, gamma=4.0, from_logits=False),
              metrics=['accuracy', tf.keras.metrics.Precision()])

model3.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_15 (Dense)            (None, 64)                576

 dropout_10 (Dropout)        (None, 64)                0

 dense_16 (Dense)            (None, 64)                4160

 dropout_11 (Dropout)        (None, 64)                0

 dense_17 (Dense)            (None, 1)                 65

=================================================================
Total params: 4801 (18.75 KB)
Trainable params: 4801 (18.75 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Precision = 92%

(92 out of 100 attacks are detected)

```python
[ ]  # Compute the confusion matrix
     cm = confusion_matrix(y_test1, y_pred_classes)
     print(cm)
```

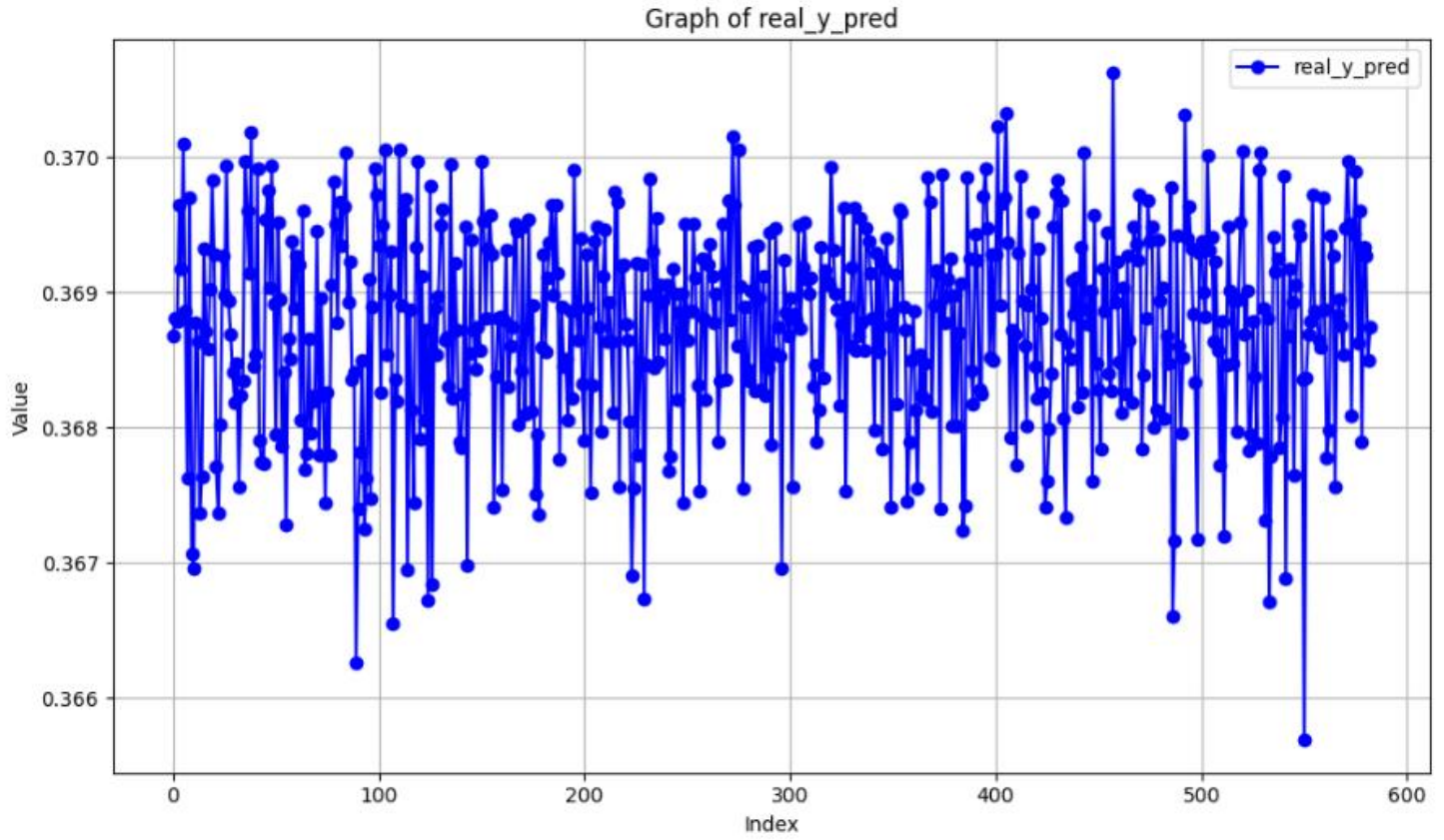
```
[[6976190    3643]
 [   2602   32395]]
```

DNS EXFILTRATION DETECTION WITH HUBBLE'S NETWORK FLOW LOGS

92 out of 100 attacks detected IF they look like the ones in the dataset

Otherwise, on 500 of "our" attacks, here are the predictions

=> threshold at 0.35



DNS EXFILTRATION DETECTION WITH HUBBLE'S NETWORK FLOW LOGS

```
ubuntu@kind-2:~/Cilium/log-processing$ python3 classify-dns.py cd
2024-06-23 21:45:18.579727: I external/local_tsl/tsl/cuda/cudart_stub.cc:32] Could not find cuda drivers on your machine, GPU will not be used.
2024-06-23 21:45:18.583865: I external/local_tsl/tsl/cuda/cudart_stub.cc:32] Could not find cuda drivers on your machine, GPU will not be used.
2024-06-23 21:45:18.643117: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operatio
ns.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2024-06-23 21:45:19.662381: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
DNS log (1), pred=0.018348218873143196, query="trivial.query.example.com.", qtype="A", from="10.0.0.35", to="8.8.4.4"
DNS log (2), pred=0.018348218873143196, query="trivial.query.example.com.", qtype="A", from="8.8.4.4", to="10.0.0.35"
DNS log (3), pred=0.020645102486014366, query="colab.research.google.com.", qtype="A", from="10.0.0.35", to="8.8.4.4"
DNS log (4), pred=0.020645102486014366, query="colab.research.google.com.", qtype="A", from="8.8.4.4", to="10.0.0.35"
DNS log (5), pred=0.0360640250146389, query="synapses.polytechnique.fr.", qtype="A", from="10.0.0.35", to="8.8.4.4"
DNS log (6), pred=0.0360640250146389, query="synapses.polytechnique.fr.", qtype="A", from="8.8.4.4", to="10.0.0.35"
DNS log (7),EXFILTRATION DETECTED! (pred=0.3757328391075134), query="vghpcybpcybzb21lihnlbnnpdgl2zsbkyxrhihroyxqgbmvlzhmgdg8gymugzxh.example.com.", qtype="A", from="10.0.0.63", to="8.
8.8.8"
DNS log (8),EXFILTRATION DETECTED! (pred=0.3757328391075134), query="vghpcybpcybzb21lihnlbnnpdgl2zsbkyxrhihroyxqgbmvlzhmgdg8gymugzxh.example.com.", qtype="A", from="8.8.8.8", to="10.0
.0.63"
DNS log (9),EXFILTRATION DETECTED! (pred=0.37424537539482117), query="mawx0cmf0zwqgb3zlcibetlmuckhlcmugaxmgysbwyxnzd29yzdoguebzc3cwcm.example.com.", qtype="A", from="10.0.0.63", to="8
.8.8.8"
DNS log (10),EXFILTRATION DETECTED! (pred=0.37424537539482117), query="mawx0cmf0zwqgb3zlcibetlmuckhlcmugaxmgysbwyxnzd29yzdoguebzc3cwcm.example.com.", qtype="A", from="8.8.8.8", to="10
.0.0.63"
DNS log (11),EXFILTRATION DETECTED! (pred=0.3734185993671417), query="qxmjmhckfuzcbozxjliglzigegc2vjcmv0igtletogu0vdukvus0vzmtizndu2.example.com.", qtype="A", from="10.0.0.63", to="8.
8.8.8"
DNS log (12),EXFILTRATION DETECTED! (pred=0.3734185993671417), query="qxmjmhckfuzcbozxjliglzigegc2vjcmv0igtletogu0vdukvus0vzmtizndu2.example.com.", qtype="A", from="8.8.8.8", to="10.0
.0.63"
DNS log (13),EXFILTRATION DETECTED! (pred=0.379856139421463), query="ecampus.paris-saclay.fr.", qtype="A", from="10.0.0.35", to="8.8.4.4"
DNS log (14),EXFILTRATION DETECTED! (pred=0.379856139421463), query="ecampus.paris-saclay.fr.", qtype="A", from="8.8.4.4", to="10.0.0.35"
DNS log (15), pred=0.04313861206173897, query="mattfradd.locals.com.", qtype="A", from="10.0.0.35", to="8.8.4.4"
DNS log (16), pred=0.04313861206173897, query="mattfradd.locals.com.", qtype="A", from="8.8.4.4", to="10.0.0.35"
```

DNS EXFILTRATION DETECTION WITH HUBBLE'S NETWORK FLOW LOGS

We can lower the rate of false positives with a better list of words. Here, by adding "ecampus" to it :

```
ubuntu@kind-2:~/Cilium/log-processing$ python3 classify-dns.py cd
2024-06-23 21:51:54.720470: I external/local_tsl/tsl/cuda/cudart_stub.cc:32] Could not find cuda drivers on your machine, GPU will not be used.
2024-06-23 21:51:54.724612: I external/local_tsl/tsl/cuda/cudart_stub.cc:32] Could not find cuda drivers on your machine, GPU will not be used.
2024-06-23 21:51:54.777763: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in
 performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2024-06-23 21:51:55.723840: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
DNS log (1), pred=0.018348218873143196, query="trivial.query.example.com.", qtype="A", from="10.0.0.35", to="8.8.4.4"
DNS log (2), pred=0.018348218873143196, query="trivial.query.example.com.", qtype="A", from="8.8.4.4", to="10.0.0.35"
DNS log (3), pred=0.020645102486014366, query="colab.research.google.com.", qtype="A", from="10.0.0.35", to="8.8.4.4"
DNS log (4), pred=0.020645102486014366, query="colab.research.google.com.", qtype="A", from="8.8.4.4", to="10.0.0.35"
DNS log (5), pred=0.0360640250146389, query="synapses.polytechnique.fr.", qtype="A", from="10.0.0.35", to="8.8.4.4"
DNS log (6), pred=0.0360640250146389, query="synapses.polytechnique.fr.", qtype="A", from="8.8.4.4", to="10.0.0.35"
DNS log (7),EXFILTRATION DETECTED! (pred=0.3757328391075134), query="vghpcybpcybzb21lihnlbnnpdgl2zsbkyxrhihroyxqgbmvlzhmgdg8gymugzxh.example.com.", qtype
="A", from="10.0.0.63", to="8.8.8.8"
DNS log (8),EXFILTRATION DETECTED! (pred=0.3757328391075134), query="vghpcybpcybzb21lihnlbnnpdgl2zsbkyxrhihroyxqgbmvlzhmgdg8gymugzxh.example.com.", qtype
="A", from="8.8.8.8", to="10.0.0.63"
DNS log (9),EXFILTRATION DETECTED! (pred=0.37424537539482117), query="mawx0cmf0zwqgb3zlcibetlmuckhlcmugaxmgysbwyxnzd29yzdoguebzc3cwcm.example.com.", qtyp
e="A", from="10.0.0.63", to="8.8.8.8"
DNS log (10),EXFILTRATION DETECTED! (pred=0.37424537539482117), query="mawx0cmf0zwqgb3zlcibetlmuckhlcmugaxmgysbwyxnzd29yzdoguebzc3cwcm.example.com.", qty
pe="A", from="8.8.8.8", to="10.0.0.63"
DNS log (11),EXFILTRATION DETECTED! (pred=0.3734185993671417), query="qxmjmhckfuzcbozxjliglzigegc2vjcmv0igtletogu0vdukvus0vzmtizndu2.example.com.", qtype
="A", from="10.0.0.63", to="8.8.8.8"
DNS log (12),EXFILTRATION DETECTED! (pred=0.3734185993671417), query="qxmjmhckfuzcbozxjliglzigegc2vjcmv0igtletogu0vdukvus0vzmtizndu2.example.com.", qtype
="A", from="8.8.8.8", to="10.0.0.63"
DNS log (13), pred=0.0444343276321888, query="ecampus.paris-saclay.fr.", qtype="A", from="10.0.0.35", to="8.8.4.4"
DNS log (14), pred=0.0444343276321888, query="ecampus.paris-saclay.fr.", qtype="A", from="8.8.4.4", to="10.0.0.35"
DNS log (15), pred=0.04313861206173897, query="mattfradd.locals.com.", qtype="A", from="10.0.0.35", to="8.8.4.4"
DNS log (16), pred=0.04313861206173897, query="mattfradd.locals.com.", qtype="A", from="8.8.4.4", to="10.0.0.35"
```

CONCLUSION

- Cilium => advanced networking and security
- Hubble => observability (metrics, logs, and integrates traces)
- Detecting DNS with Hubble ?
  - Metrics => flawed but enough to detect high-throughput DNS exfiltrations
  - Logs => easy to export
    Then, a simple NN is engough to classify base64 DNS exfiltrations !