containerday
DIGITAL EDITION

Falco & Falcosidekick: Osservabilità, Sicurezza, Monitoraggio e Automazione nel mondo K8S

ANDREA VIVALDI
Team Leader & Solution Architect @ Vista Technology Srl

8 Ottobre 2021 | Online | containerday.it

GRUSP

MI PRESENTO, PIACERE DI CONOSCERVI!

https://github.com/andreavivaldi

https://www.linkedin.com/in/andreavivaldi/

andrea.vivaldi@vistatech.it

https://www.vistatech.it/

Mi chiamo ANDREA VIVALDI

- Sono Team Leader e Solution Architect, in ambito DevOps e tecnologie abilitanti al Cloud Native, per **Vista Technology**
- Sono da sempre appassionato ai temi di **Automazione**, **Osservabilità**, **Telemetria** e **Monitoraggio**
- Ultimamente mi districo tra **Secure DevOps** e **Network Automation**

DI COSA PARLEREMO OGGI

- **Osservabilità** e **Sicurezza** in ambito K8S
  - Criticità
  - Importanza di utilizzare gli strumenti giusti
  - **Closed-loop** con **Automazione/Remediation**
- Tools eBPF
  - Falco
  - Falcosidekick
- DEMO
  - Esempio pratico (molto «*opinionato*») su un possibile **stack** da utilizzare

DISCLAIMER

- Non è un talk di approfondimento tecnico sulla tecnologia eBPF
- Non è un deep dive su Falco
- Non è un deep dive su Kubeless e le funzioni Serverless

… Allora cos'è?

Semplicemente un caso di studio su come iniziare ad approcciare ed utilizzare tecnologie innovative per tematiche «calde» in ambito K8S e Secure DevOps… dal punto di vista di un umile utilizzatore

PARTIAMO DA UN PRESUPPOSTO

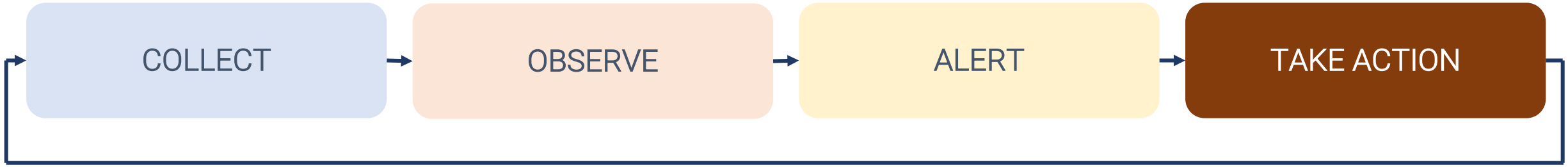
- **Kubernetes** è il «*nuovo*» **S.O.** nel mondo Cloud Native e Microservizi
- Nuovi **drivers**:
  - Velocità e di innovazione
  - Efficienza esasperata nel rapporto costi/benefici
  - Mitigazione continua del rischio

- Le nuove sfide dei team di **DevOps: Secure DevOps**

ERA «CLOUD-NATIVE»: NUOVE SFIDE

- Approccio «*all*»OPS
  - Qualsiasi cosa, dalle infrastrutture di rete ai server allo sviluppo applicativo, si basa sui principi e sulle pratiche del DevOps
  - Contesto comune tra team eterogenei
- Hyper-fragmentation
  - Architetture orientate ai Microservizi
- Velocità e resilienza
  - Necessità di velocizzare i processi di rilascio e la raccolta dei dati di feedback
- Componenti effimeri
  - *«non posso misurare ciò che non riesco a vedere»*

MODELLO CLOSED-LOOP TELEMETRY E AUTOMATION



CONTINUOUS FEEDBACK	Integrare continuous monitoring & observability nei propri processi di CI/CD, per ottenere feedback in real-time
PERFORMANCE	Ottenere la maggior parte di dati sensibili, out-of-the-box, nel minor tempo possibile, estrarli e visualizzarli in maniera agile
COVERAGE	Garantire il pieno accesso e l'allarmistica agli operatori
AUTOMATION	Avere dati che siano utili per le toolchain, in maniera tale che possano reagire ad eventi e triggerare automazioni

E LA SICUREZZA CHE C'AZZECCA?...

- Le funzionalità di **osservabilità** e **telemetria** sono «*sorgente*» per i processi di **sicurezza** e **auditing**
  - Integrazione K8S Audit Log
  - Runtime detection & protection
  - Performance & Capabilities
  - …

MA HO GIÀ TANTI TOOL A DISPOSIZIONE...

STRUMENTI LEGACY

- Non «container-native»
- Non tengono in considerazione il contesto K8S
- Nessun concetto di DevOps

SOLUZIONI IPER-SPECIFICHE

- Instrumentazione spesso invasiva
- Limitato il contesto K8S
- Mancanze in termini di scalabilità e granularità del dato

STRUMENTI PURPOSE-BUILT

Tecnologie nate e/o pensate per un mondo Cloud Native

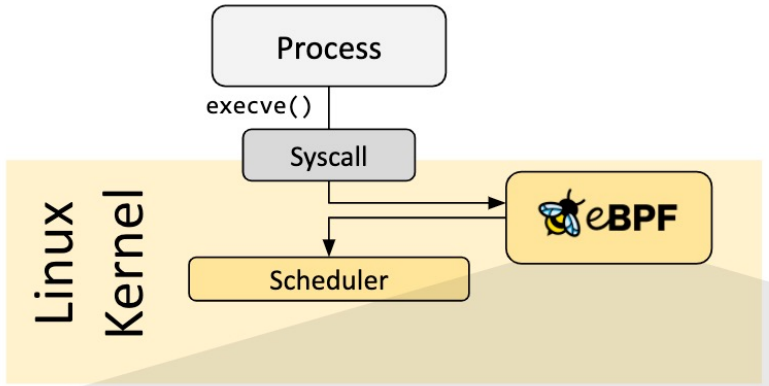
UNA POSSIBILE STRADA TECNOLOGICA: FALCO

- Open Source
- Creato da Sysdig nel 2016
- Primissimo progetto di runtime security ad essere «*incubato*» dentro la CNCF

- Cloud Native **runtime Security** & **Visibility**
- Motore di **Threat Detection**, standard *de-facto* in ambito K8S
- Utilizza la tecnologia **eBPF** per intercettare
  - System calls
  - Eventi kernel
  - Kubernetes audit events
- **Monitoraggio** del comportamento di un cluster sotto tutti i punti di vista
  - Accesso a dati sensibili
  - Attività dei nodi del cluster
  - Attività dei POD

eBPF (…GIUSTO UN CENNO…)

- *Extended Berkeley Paket Filter*
- Tecnologia **kernel** che consente ai programmi di girare come se fossero dentro una **sandbox**, in cui poter beneficiare di **capabilities** specifiche del kernel senza aggiungere moduli o toccare il codice sorgente
- I programmi eBPF sono **event-driven** e vanno in esecuzione nel momento in cui il kernel o un'applicazione passa attraverso un certo **hook**

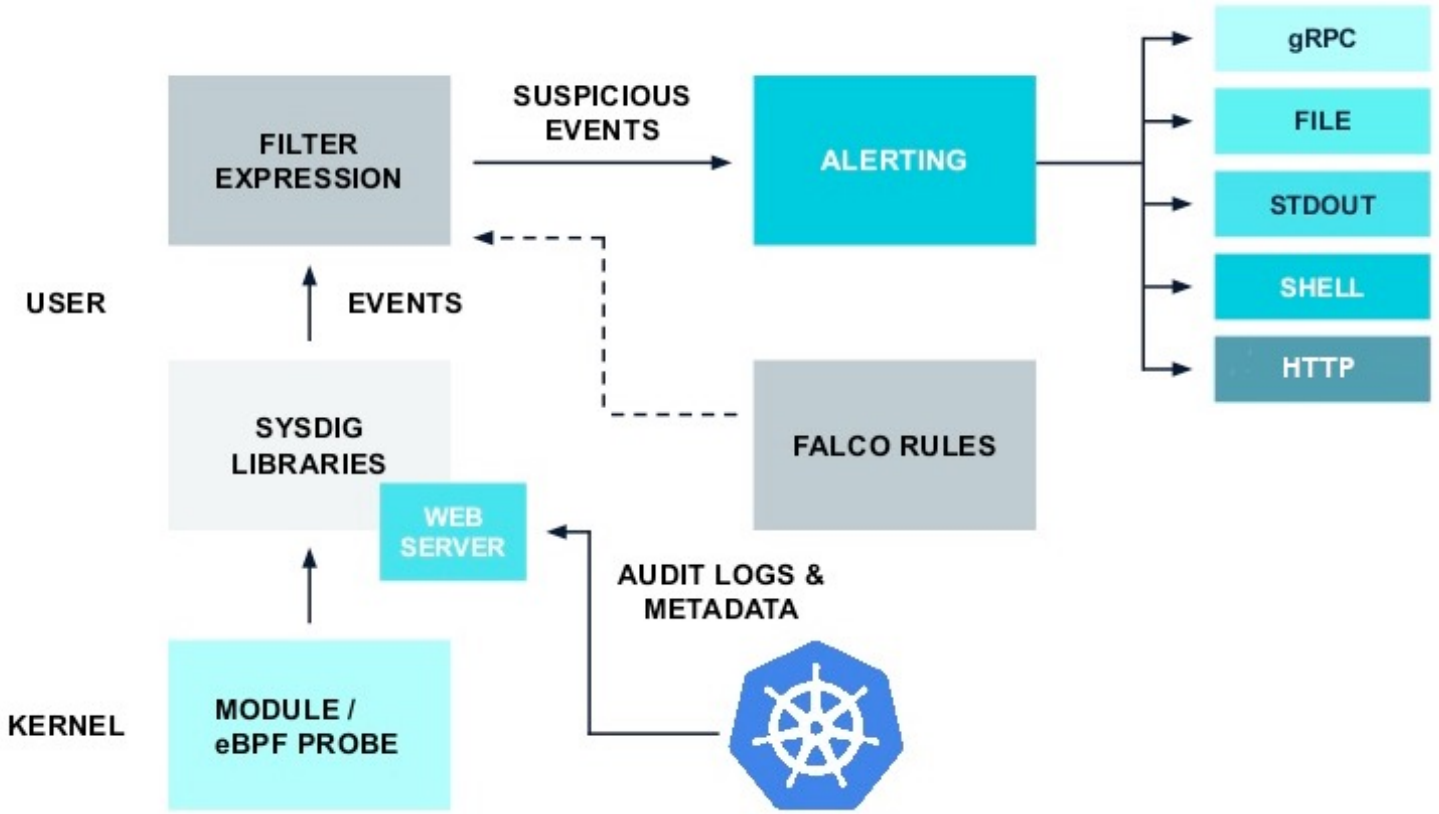



```
int syscall__ret_execve(struct pt_regs *ctx)
{
        struct comm_event event = {
                .pid = bpf_get_current_pid_tgid() >> 32,
                .type = TYPE_RETURN,
        };

        bpf_get_current_comm(&event.comm, sizeof(event.comm));
        comm_events.perf_submit(ctx, &event, sizeof(event));

        return 0;
}
```

INTERESSANTE… E COME FUNZIONA FALCO?

Falco utilizza i dati raccolti dai driver kernel per:

- Parsare le system call a livello Kernel
- Verificare i dati in base al motore di Rules
- Allertare in output in caso di violazione



COSA CERCA DI PRECISO FALCO?

- Una shell aperta all'interno di un container/POD in Kubernetes
- Un container che gira in privileged mode, o che monta un path sensibile direttamente dall'host (es: /proc)
- Un processo server che «spawna» un processo figlio inaspettatamente
- Lettura inaspettate di file particolare, come ad esempio /etc/shadow
- Un file non-device scritto sotto /dev
- Un binario standard di sistema, come ad esempio ls, che esegue una connessione verso l'esterno
- …

(l'elenco non è ovviamente esaustivo…)

BENEFICI NELL'UTILIZZO DI FALCO

- Tutta la **magia** avviene a livello **Kernel**, quindi:
  - **Minimo impatto** su **performance** e sistemi
  - **Granularità** del dato massima
- Naturalmente pensato per lavorare in un mondo *K8S-oriented*
  - Integrazione con **Audit Logs** & **Events**
  - Compatibilità con deploy su qualsiasi piattaforma **Kubernetes**
- Stessa tecnologia condivisa per **Host** e **Container**
- Nessuna necessità di aggiungere e gestire **sidecars** container
- Importante libreria di **Rules** predefinite per il **detect** di attività malevole ed **exploit CVE**
- **Integrazione** con gli standard *de-facto* in termini di **telemetria** e **monitoraggio**

FALCOSIDEKICK: ESTENSIONE A FALCO

- Limite di **Falco**: solo **5 formati** di **output** (stdout, file, gRPC, shell, HTTP)
- Come posso integrare e far interagire Falco con **altri strumenti**, per esempio per poter creare delle **automazioni** in **remediation**?
- Entra in gioco **Falcosidekick**, un **demone** che si mette in ascolto degli output di Falco e li rende **disponibili** con un gran numero di **connettori terze parti**
  - **Chat** (es: Slack, Teams, Discord,...)
  - **Metriche/Osservaibilità** (Prometheus, InfluxDB,...)
  - **Alerting** (AlertManager, PagerDuty, ...)
  - **Logs** (ELK, Loki, Syslog, ...)
  - **Object Storage**
  - **FaaS/Serverless** (Lambda, Kubeless, OpenFaaS, Fission, ...)
  - **Message queue**
  - **Email**
  - **Web UI** (*Falcosidekick UI*)

CAPITO! ORA METTIAMO TUTTI I PEZZI INSIEME…

- Facciamo un esempio pratico e consideriamo il seguente caso d'uso:
  - Requisiti
    - Semplice cluster Kubernetes come punto di partenza
    - Banalissimo POD di prova deployato sul cluster per testare comportamenti «malevoli»
      - ES: detect di una connessione network verso un URL specifico
      - ES: detect di un shell non autorizzata nel POD
  - Obiettivi
    - Step 1:
      - Testare le Falco Rules e intercettare eventuali violazioni
    - Step 2:
      - Integrazione per un monitoraggio a lungo termine
    - Step 3:
      - Implementare una remediation automatizzata

STEP 1

- Installazione di Falco, tramite Helm, all'interno di un namespace dedicato
- Aggiunta di una regola custom
  - Check traffico di rete verso indirizzo IP 213.215.222.58 (https://www.vistatech.it)
- Deploy di app «malevole»
  - ES: POD Nginx
- Test del comportamento
  - Connessione alla shell
  - Esecuzione di un cURL verso https://www.vistatech.it
- Analisi dei risultati di Falco prodotti in STDOUT
  - Verificare che nei log vengano intercettati gli eventi sopra indicati

ANATOMIA DELLA FALCO RULE CUSTOM

```
- macro: outbound
  condition: >
    (((evt.type = connect and evt.dir=<)) or
    (fd.typechar = 4 or fd.typechar = 6) and
    (fd.ip != "0.0.0.0" and fd.net != "127.0.0.0/8") and
    (evt.rawres >= 0 or evt.res = EINPROGRESS))

- list: website_ips
  items: ['"213.215.222.58"', '"213.215.222.59"']

- rule: Connection to vistatech.it
  desc: Detect attempts to connect to Vista Technology website
  condition: outbound and fd.sip in (website_ips)
  output: Outbound connection to Vista Technology website https://www.vistatech.it
    (command=%proc.cmdline connection=%fd.name container_id=%container.id
    container_name=%container.name %container.info image=%container.image)
  priority: WARNING
  tags: [network]
```

STEP 2

- Persistere le informazioni all'interno di uno stack di telemetria/osservabilità e agganciare un sistema di messaggistica istantanea per le notifiche
- Installiamo Falcosidekick (anch'esso via Helm)
  - Output verso:
    - Loki
    - Prometheus (via additional scrape config dell'endpoint /metrics)
    - Slack
    - Kubeless function (che servirà per lo Step 3)
- Generiamo nuovamente eventi ed osserviamo i risultati sulle dashboard Grafana
- Possiamo accedere anche alla UI di Falcosidekick per una prima visualizzazione ad alto livello

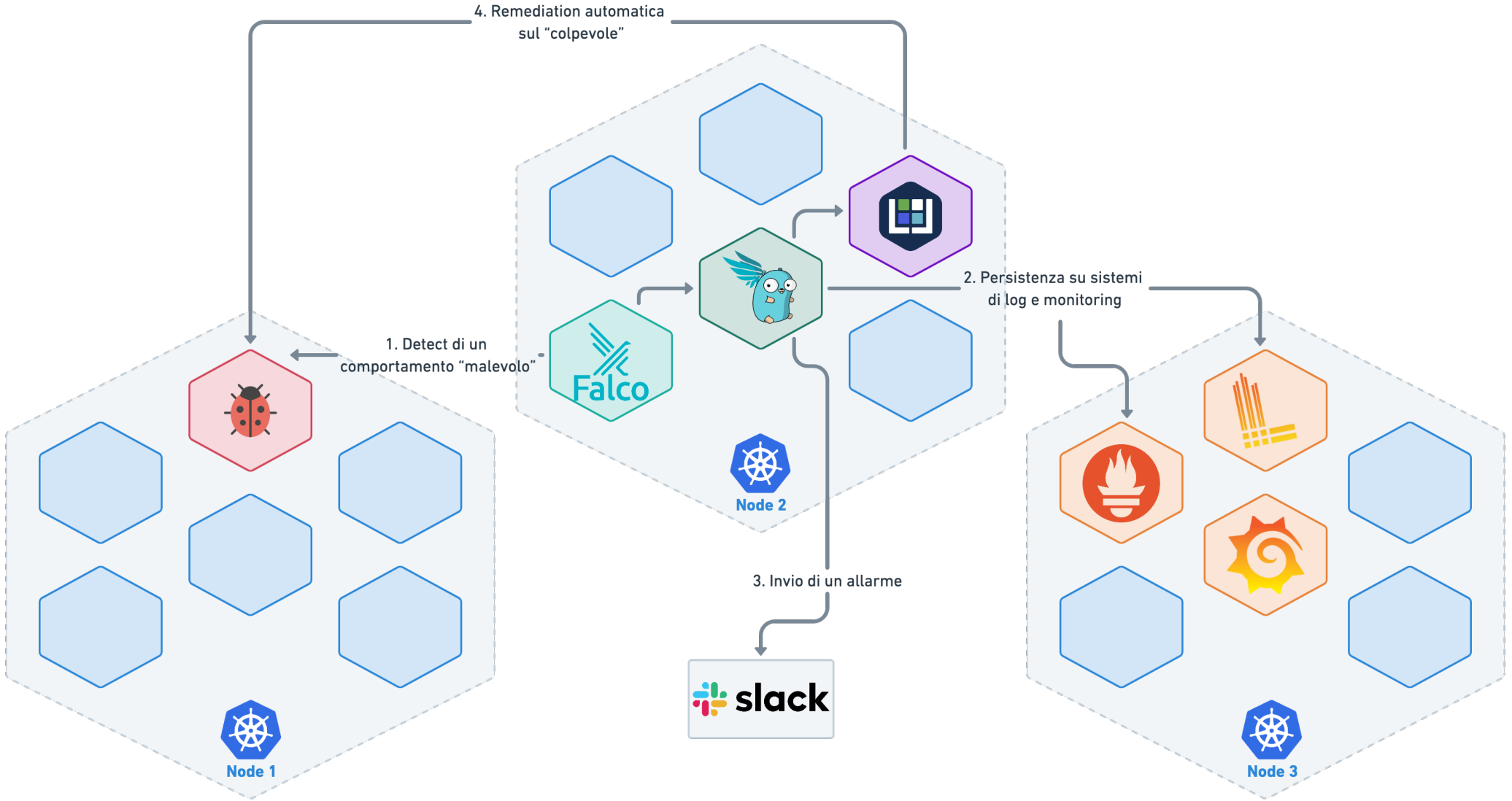
STEP 3

- Facciamo un ultimo passo per andare in direzione di un «closed loop» tra osservabilità ed automazione (in questo caso remediation)
- Utilizziamo Falcosidekick per andare a triggerare una funzione serverless (con Kubeless)
  - Tale funzione intercetterà l'evento della Custom Rule «Connection to vistatech.it» e andrà a simulare un'azione di rimedio semplicemente andando a creare una NetworkPolicy per vietare il traffico in uscita
  - Dopo la prima cURL il POD Ngnix non sarà più in grado di effettuare connessioni di rete verso l'esterno

FUNZIONE SERVERLESS

```
from kubernetes import client,config

config.load_incluster_config()

def isolate_pod(event, context):
    rule = event['data']['rule'] or None
    output_fields = event['data']['output_fields'] or None

    if rule and rule == "Connection to vistatech.it" and output_fields:
        if output_fields['k8s.ns.name'] and output_fields['k8s.pod.name']:
        pod = output_fields['k8s.pod.name']
        namespace = output_fields['k8s.ns.name']
        body = client.V1NetworkPolicy(
            api_version="networking.k8s.io/v1",
            kind="NetworkPolicy",
            metadata=client.V1ObjectMeta(name="default-deny-egress"),
            spec=client.V1NetworkPolicySpec(
              pod_selector=client.V1LabelSelector(
              match_labels= {"app":"nginx"}),
            policy_types=["Egress"]))
        print (f"Isolating pod \"{pod}\" in namespace \"{namespace}\"")
        client.NetworkingV1Api().create_namespaced_network_policy(namespace=namespace, body=body)
```

RIASSUMENDO… ECCO LO SCHEMA FINALE



DEMO TIME

https://github.com/Vista-Technology/falco-and-falcosidekick-talk-containerday-2021

CREDITS

- https://sysdig.com/blog/ (immagini e articoli)
- https://falco.org/docs (immagini e articoli)
- https://github.com/falcosecurity/falco
- https://github.com/falcosecurity/falcosidekick
- https://ebpf.io/what-is-ebpf/
- https://falco.org/blog/falco-on-rke-with-rancher/
- https://github.com/falcosecurity/charts/tree/master/falco
- https://falco.org/blog/intro-k8s-security-monitoring/
- https://falco.org/blog/falco-kind-prometheus-grafana/
- https://falco.org/blog/extend-falco-outputs-with-falcosidekick/
- https://falco.org/blog/falcosidekick-response-engine-part-1-kubeless/
- https://sysdig.com/blog/unexpected-domain-connection/

GRAZIE A TUTTI PER L'ATTENZIONE!

DOMANDE....?

CONTATTI

MILANO
P.zza Indro Montanelli, 20
20099 Milano

REGGIO EMILIA
Via Louis Pasteur, 16
42122 Reggio Emilia

ROMA
Via Paolo di Dono 73 00142
Roma

GENOVA
Via XX Settembre, 41
16121 Genova

BARI
Via Marco Partipilo, 30
70124 Bari

NAPOLI
Via Giovanni Porzio, 4
80143 Napoli

https://github.com/Vista-Technology

https://www.linkedin.com/company/vista-technology-srl

https://www.vistatech.it/

info@vistatech.it

VISTA TECHNOLOGY