7 июня 2023 Москва, МЦК ЗИЛ

БЕКОН'23

Первая в России конференция
по БЕзопасности КОНтейнеров и контейнерных сред

NetworkPolicy для системных и инфраструктуры компонент

Александр Кожемякин

SRE, VK

Обо мне

Александр Кожемякин

SRE, VK

Intro

Или зачем нужна эта презентация

- Вокруг много облаков в целом и kubernetes в частности
- Zero-trust networks
- Новые сервисы хочется выкатывать как можно скорее
- Много новых угроз

Немного тезисов

А вот о чем поговорим

- Как обеспечить безопасность базовых компонент как k8s, так и инфрастуктурных на уровне сетевых политик
- Какими инструментами воспользоваться
- Делаем политики один раз и они подходят для всех продуктовых сервисов, как текущих, так и будущих

cni

- Лучше всего нам подойдут cilium или calico
- Понимание того как работают наши инфраструктуры компоненты
- Много времени на создание и улучшение сетевой безопасности

API GROUP	NAMESPACED	KIND
cilium		
cilium.io	false	CiliumClusterwideNetworkPolicy
cilium.io	true	Cilium NetworkPolicy
networking.k8s.io	true	NetworkPolicy
calico		
crd.projectcalico.org	false	GlobalNetworkPolicy
crd.projectcalico.org	true	NetworkPolicy
networking.k8s.io	true	NetworkPolicy
weave		
networking.k8s.io	true	NetworkPolicy
kube-router		
networking.k8s.io	true	NetworkPolicy
flannel		
networking.k8s.io	true	NetworkPolicy (Не уМЕЕТ!)

Без чего нам точно не справиться?

Без DNS

DNS и политики:

- вы ним точно пользуетесь для своих инфраструктуры компонент
- ваши пользователи его точно используют

Как сделать:

- политики должны быть применимы ко всем компонентам
- надо написать один раз

Не надо так

```yaml
apiVersion: cilium.io/v2
kind: CiliumClusterwideNetworkPolicy
metadata:
  name: dns-policy
spec:
  endpointSelector: {}
  egress:
   - toEndpoints:
      - matchLabels:
          io.kubernetes.pod.namespace: kube-dns
     toPorts:
      - ports:
        - port: "53"
          protocol: UDP
        - port: "53"
          protocol: TCP
```

```yaml
apiVersion: cilium.io/v2
kind: CiliumClusterwideNetworkPolicy
metadata:
  name: dns-policy
spec:
  endpointSelector: {}
  egress:
  - toEntities:
    - cluster
    toPorts:
      - ports:
        - port: "53"
          protocol: UDP
        - port: "53"
          protocol: TCP
```

Базовые политики DNS

- Простая политика, легко пишется
- Применяется сразу ко всему кластеру
- Нет необходимости в настройке для каждого нового компонента

```yaml
apiVersion: cilium.io/v2
kind: CiliumClusterwideNetworkPolicy
metadata:
  name: simpleDnsPolicy
spec:
  endpointSelector: {}
  egress:
  - toEndpoints:
    - matchLabels:
        k8s:io.kubernetes.pod.namespace: dns
        k8s:app.kubernetes.io/instance: core-dns
        k8s:app.kubernetes.io/name: core-dns
    toPorts:
      - ports:
        - port: "53"
          protocol: UDP
        - port: "53"
          protocol: TCP
```

А теперь усложним

- Не так уж и сложно
- Помогает фильтровать запросы по зонам
- Можем создавать гибкие правила и паттерны с привязкой к протоколам, например

```yaml
apiVersion: cilium.io/v2
kind: CiliumClusterwideNetworkPolicy
metadata:
  name: zoneDnsPolicy
spec:
  endpointSelector: {}
  egress:
  - toEndpoints:
    - matchLabels:
        k8s:io.kubernetes.pod.namespace: dns
        k8s:app.kubernetes.io/instance: core-dns
        k8s:app.kubernetes.io/name: core-dns
    toPorts:
      - ports:
        - port: "53"
          protocol: ANY
        rules:
          dns:
            - matchPattern: "*.internal.zone"
```

Так безопаснее

```yaml
apiVersion: cilium.io/v2
kind: CiliumNetworkPolicy
metadata:
  name: verySecure
  namespace: test-app-ns
spec:
  endpointSelector: {}
  egress:
  - toEndpoints:
    - matchLabels:
        k8s:io.kubernetes.pod.namespace: dns
        k8s:app.kubernetes.io/instance: core-dns
        k8s:app.kubernetes.io/name: core-dns
    toPorts:
      - ports:
        - port: "53"
          protocol: UDP
        rules:
          dns:
            - matchPattern: "*.internal.zone"
  - toFQDNs:
    - matchName: "safety.internal.zone"
```

Gitlab-нашевсе-runner

- Простейший вариант политики
- Доступ по ip адресам
  не лучший вариант

```yaml
apiVersion: cilium.io/v2
kind: CiliumNetworkPolicy
metadata:
  name: gitlab-runner-shared.egress.to.gitlab
  namespace: executors
spec:
  description: access from gitlab-runner-shared instance to gitlab
  egress:
  - toCIDRSet:
    - cidr: 10.0.0.1/32
    - cidr: 20.0.0.2/32
    - cidr: 30.0.0.3/32
    toPorts:
    - ports:
      - port: "443"
        protocol: TCP
  endpointSelector:
    matchLabels:
      k8s:app.kubernetes.io/instance: gitlab-runner-shared
      k8s:app.kubernetes.io/name: gitlab-runner-shared
      k8s:io.kubernetes.pod.namespace: executors
```

А если у нас calico

- Не стоит грустить, все так же просто
- Выдаем доступ по нашим адресам
- Синтаксис у cilium и calico очень похож
- Переход с одного cni на другой довольно интуитивен в ключе политик

```yaml
apiVersion: crd.projectcalico.org/v1
kind: NetworkPolicy
metadata:
  name: allow-gitlab-runner
  namespace: executors
spec:
  selector: gitlab-runner-instance == gitlab-shared
  order: 10
  egress:
  - action: Allow
    protocol: TCP
    destination:
      nets:
        - 10.0.0.1/32
        - 20.0.0.2/32
        - 30.0.0.3/32
      ports:
      - 443
  types:
  - Egress
```

А как же fqdn?

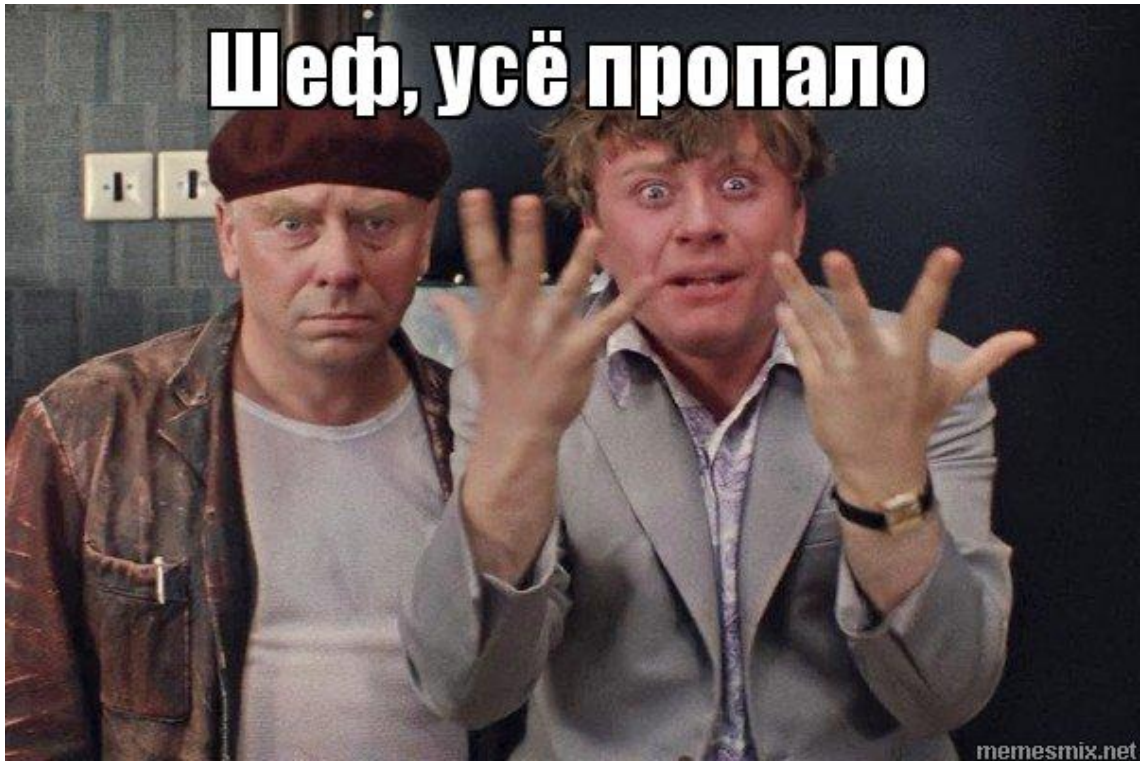
```yaml
apiVersion: projectcalico.org/v1
kind: NetworkPolicy
metadata:
  name: alertwebhook.egress.to.external.vault
  namespace: vmalert-ns
spec:
  tier: security
  selector: app == "alertwebhook"
  order: 90
  types:
    - Egress
  egress:
  - action: Allow
    protocol: TCP
    source:
      selector: app == 'alertwebhook'
    destination:
      domains:
      - 'vault.server.com'
      ports:
      - 8200
```

vs

```yaml
apiVersion: cilium.io/v2
kind: CiliumNetworkPolicy
metadata:
  name: alertwebhook.egress.to.external.vault
  namespace: vmalert-ns
spec:
  endpointSelector:
    matchLabels:
      k8s:app.kubernetes.io/name: alertwebhook
  egress:
    - toFQDNs:
        - matchName: vault.server.com
      toPorts:
        - ports:
            - port: "8200"
              protocol: TCP
```

Legacy и L7

- Политики внедряются
- Движемся в светлое будущее
- Легаси сервисы, дорабатывать долго/дорого/лень



Или не все?

- Политики для L7 нас спасут
- И даже помогут с авторизацией запросов
- Можем выбирать комбинации путей и типов запросов
- Можем проверять метаданные для каждой комбинации

```yaml
apiVersion: "cilium.io/v2"
kind: CiliumNetworkPolicy
metadata:
  name: "l7-rule"
  namespace: verySecure
spec:
  endpointSelector:
    matchLabels:
      app: myService
  ingress:
  - toPorts:
    - ports:
      - port: '80'
        protocol: TCP
      rules:
        http:
        - method: GET
          path: "/path1"
        - method: PUT
          path: "/path2"
          headers:
          - 'X-My-Header: true'
```

Кафка. Греффневая кафка

```yaml
apiVersion: "cilium.io/v2"
kind: CiliumNetworkPolicy
metadata:
  name: "rule1"
spec:
  description: "star wars"
  endpointSelector:
    matchLabels:
      app: kafka
  ingress:
  - fromEndpoints:
    - matchLabels:
        app: empire-hq
    toPorts:
    - ports:
      - port: "9092"
        protocol: TCP
      rules:
        kafka:
        - role: "produce"
          topic: "deathstar-plans"
        - role: "consume"
          topic: "empire-announce"
```

```yaml
apiVersion: "cilium.io/v2"
kind: CiliumNetworkPolicy
metadata:
  name: "rule1"
spec:
  description: "darth tum tududum"
  endpointSelector:
    matchLabels:
      app: kafka
  ingress:
  - fromEndpoints:
    - matchLabels:
        app: empire-hq
    toPorts:
    - ports:
      - port: "9092"
        protocol: TCP
      rules:
        kafka:
        - apiKey: "apiversions"
        - apiKey: "metadata"
        - apiKey: "produce"
          topic: "deathstar-plans"
        - apiKey: "produce"
          topic: "empire-announce"
```

И еще немного про кафку

Плюсы:

- углубляемся на уровень сервиса
- разграничиваем доступы наших приложений к коммунальной Кафке
- широкие политики по ролям
  и супер гибкие по ключам api

Минусы :

- фича пока в бете
- время на написание такого рода политик

Куда это все сложить

- В git конечно
- Активно используем шаблонизаторы
- Максимально унифицируем политики

```
apiVersion: crd.projectcalico.org/v1
kind: NetworkPolicy
metadata:
  name: default.ingress-{{ .Release.Name }}-webhook-api
spec:
  types:
  - Ingress
  ingress:
  - action: Allow
    protocol: TCP
    source:
      nets:
      {{- toYaml .Values.global.networkAddresses.masters | nindent 6 }}
    destination:
      ports:
      - 8443
  order: 90
  selector: app == '{{ .Release.Name }}' && gatekeeper.sh/operation == 'webhook'
```

Осталось не запутаться

- Храним максимально гранулярно
- Используем один источник правды
- Политики не добавляются руками
- Управление всегда глобально, применяется ко всем

Что же получилось

- Внедрив сетевые политик для инфраструктурных компонент мы смогли повысить безопасность кластера
- Частично обезопасили легаси приложения
- Получили хорошее обсервабилити сетевых взаимодействий
- Научились лучше понимать, как работает наша инфраструктура
- Сумели построить zero-trust

Итог

1. Политики точно стоит использовать, хотя бы для части компонент кластера.
2. А знания полученные в процессе — документировать
3. Хорошая экономия времени в будущем — один раз пишем и отлаживаем политику — применяем ко всем кластерам

Хочу больше знаний

Больше примеров политик

Как сделать Deny all политики в живом кластере

NetworkPolicy — родной межсетевой экран Kubernetes

Contacts:

Email: avernusalex@gmail.com Tg: @kozhemyash