

Сканирование кластера с помощью инструмента kube-hunter в двух режимах

--

Для начала работ вам понадобится:

- *docker*
- *k3d*

Подробная установка docker рассматривалась в методических материалах к предыдущим урокам

--

Часть 1. Внешнее сканирование кластера из контейнера

Создадим подопытный кластер k3d:

```
$ k3d cluster create mycluster
```

```
filipp@filipp-notebook:~$ k3d cluster create mycluster
INFO[0000] Prep: Network
INFO[0000] Created network 'k3d-mycluster'
INFO[0000] Created volume 'k3d-mycluster-images'
INFO[0000] Starting new tools node...
INFO[0000] Starting Node 'k3d-mycluster-tools'
INFO[0001] Creating node 'k3d-mycluster-server-0'
INFO[0001] Creating LoadBalancer 'k3d-mycluster-serverlb'
INFO[0001] Using the k3d-tools node to gather environment information
INFO[0001] HostIP: using network gateway 172.18.0.1 address
INFO[0001] Starting cluster 'mycluster'
INFO[0001] Starting servers...
INFO[0001] Starting Node 'k3d-mycluster-server-0'
INFO[0006] All agents already running.
INFO[0006] Starting helpers...
INFO[0006] Starting Node 'k3d-mycluster-serverlb'
INFO[0012] Injecting '172.18.0.1 host.k3d.internal' into /etc/hosts of all nodes...
INFO[0012] Injecting records for host.k3d.internal and for 2 network members into CoreD
figmap...
INFO[0013] Cluster 'mycluster' created successfully!
INFO[0014] You can now use it like this:
kubectl cluster-info
```

Запустим контейнер сканера kube-hunter, но пока не будем вводить значения для настройки сканирования:

```
$ docker run -it --rm aquasec/kube-hunter
```

```
filipp@filipp-notebook:~$ docker run -it --rm aquasec/kube-hunter
Choose one of the options below:
1. Remote scanning      (scans one or more specific IPs or DNS names)
2. Interface scanning   (scans subnets on all local network interfaces)
3. IP range scanning    (scans a given IP range)
Your choice: 
```

В другом окне терминала вызовем команды просмотра запущенных контейнеров и доступных сетей:

```
$ docker ps
```

```
filipp@filipp-notebook:~$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED
daca632eeff7	aquasec/kube-hunter	"kube-hunter"	About a minute ago
829dc24621c8	rancher/k3d-proxy:5.2.2	"/bin/sh -c nginx-pr..."	3 minutes ago
96a65252d3eb	rancher/k3s:v1.21.7-k3s1	"/bin/k3d-entrypoint..."	3 minutes ago

```
$ docker network ls
```

```
filipp@filipp-notebook:~$ docker network ls
```

NETWORK ID	NAME	DRIVER	SCOPE
ccd6e5deee21	bridge	bridge	local
34a7c0b58f04	host	host	local
1279d94bfba3	k3d-mycluster	bridge	local
4ed5072f636c	none	null	local

Выполним добавление контейнера kube-hunter в bridge-сеть кластера k3d (не забудьте подставить в команду ваш networkid и containerid:

```
$ docker network connect k3d-mycluster daca632eeff7
```

```
filipp@filipp-notebook:~$ docker network connect k3d-mycluster daca632eeff7
```

Выполним команду инспектирования сети, чтобы убедиться в наличии контейнера в ней:

```
$ docker network inspect k3d-mycluster
```

```
"Containers": {  
  "829dc24621c84200408a64ff74ab5979dfb9806067ba7780dd0632a8820ee479": {  
    "Name": "k3d-mycluster-serverlb",  
    "EndpointID": "470feab4dde7c59bc102ddc2fa46b736201f5dc7b73c816ed2b9f62ec95e0509",  
    "MacAddress": "02:42:ac:12:00:02",  
    "IPv4Address": "172.18.0.2/16",  
    "IPv6Address": ""  
  },  
  "96a65252d3eb8dfee7a97df6c19662aef389406de0435cdc4686f7135c6fec7": {  
    "Name": "k3d-mycluster-server-0",  
    "EndpointID": "132cdac55bc1ad7019decd25fff3edc03cacc6939296a5ba25b54f58f892ca35",  
    "MacAddress": "02:42:ac:12:00:03",  
    "IPv4Address": "172.18.0.3/16",  
    "IPv6Address": ""  
  },  
  "daca632eef740140efeb966426d44bb589c6299213eb93b87fa9a4424ad20b2": {  
    "Name": "compassionate_gauss",  
    "EndpointID": "dccf713289cf78fa346beaff4fe848e6db093c5c047ec6fc0ca70baa19abf316",  
    "MacAddress": "02:42:ac:12:00:04",  
    "IPv4Address": "172.18.0.4/16",  
    "IPv6Address": ""  
  }  
}
```

Контейнер присутствует в сети кластера. В этом же выводе запомним IPv4Address контейнера с k3d (k3d-mycluster-server-0).

Теперь вернемся к окну терминала с настройках сети

Выбираем опции:

- тип сканирования: 1 (Remote scanning)
- remotes: <k3d-mycluster-server-0 IPv4Adress>

```

Choose one of the options below:
1. Remote scanning      (scans one or more specific IPs or DNS names)
2. Interface scanning   (scans subnets on all local network interfaces)
3. IP range scanning    (scans a given IP range)
Your choice: 1
Remotes (separated by a ','): 172.18.0.3
2023-03-10 11:37:40,134 INFO kube_hunter.modules.report.collector Started hunting
2023-03-10 11:37:40,136 INFO kube_hunter.modules.report.collector Discovering Open
s
2023-03-10 11:37:40,199 INFO kube_hunter.modules.report.collector Found open serv
172.18.0.3:10250
2023-03-10 11:37:40,219 INFO kube_hunter.modules.report.collector Found open serv
s API" at 172.18.0.3:6443

Nodes
+-----+-----+
| TYPE      | LOCATION  |
+-----+-----+
| Node/Master | 172.18.0.3 |
+-----+-----+

Detected Services
+-----+-----+-----+
| SERVICE          | LOCATION          | DESCRIPTION          |
+-----+-----+-----+
| Unrecognized K8s API | 172.18.0.3:6443 | A Kubernetes API service |
+-----+-----+-----+
| Kubelet API       | 172.18.0.3:10250 | The Kubelet is the main component in every Node, all pod operations goes through the kubelet |
+-----+-----+-----+

No vulnerabilities were found

```

Сканирование завершено, уязвимостей в кластере не выявлено

--

Часть 2. Сканирование кластера из pod в развернутом кластере

Произведем сканирование из pod, развернутого в самом кластере

Для данного типа сканирования нам потребуется объект job в кластере (единоразовый запуск рабочего задания в pod).

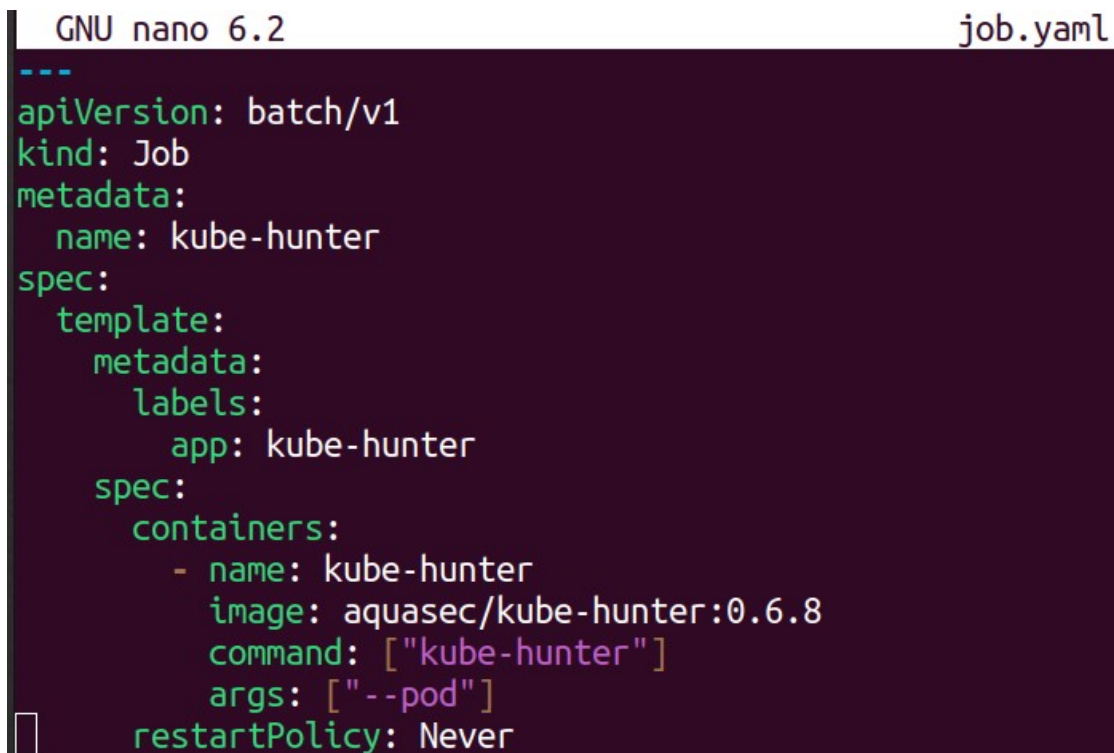
Создадим манифест такого объекта:

```
$ nano job.yaml
```

Добавим в манифест следующее содержание:

```
---
apiVersion: batch/v1
kind: Job
metadata:
  name: kube-hunter
spec:
  template:
    metadata:
      labels:
        app: kube-hunter
    spec:
      containers:
        - name: kube-hunter
          image: aquasec/kube-hunter:0.6.8
          command: ["kube-hunter"]
          args: ["--pod"]
      restartPolicy: Never
```

И сохраним изменения



```
GNU nano 6.2                                     job.yaml
---
apiVersion: batch/v1
kind: Job
metadata:
  name: kube-hunter
spec:
  template:
    metadata:
      labels:
        app: kube-hunter
    spec:
      containers:
        - name: kube-hunter
          image: aquasec/kube-hunter:0.6.8
          command: ["kube-hunter"]
          args: ["--pod"]
      restartPolicy: Never
```

Применим манифест в кластере:

```
$ kubectl apply -f job.yaml
```

```
filipp@filipp-notebook:~/Desktop$ kubectl apply -f job.yaml
job.batch/kube-hunter created
```

Выполним команды просмотра рабочих заданий (job) в кластере:

```
$ kubectl get job
```

```
filipp@filipp-notebook:~/Desktop$ kubectl get job
NAME             COMPLETIONS  DURATION  AGE
kube-hunter      1/1           40s       21m
```

Вызовем подробную информацию о выполненной job kube-hunter:

```
$ kubectl describe job kube-hunter
```

```

filipp@filipp-notebook:~/Desktop$ kubectl describe job kube-hunter
Name: kube-hunter
Namespace: default
Selector: controller-uid=5330c3d2-47cb-48a1-9cd2-785ee3c579f1
Labels: app=kube-hunter
        controller-uid=5330c3d2-47cb-48a1-9cd2-785ee3c579f1
        job-name=kube-hunter
Annotations: <none>
Parallelism: 1
Completions: 1
Start Time: Fri, 10 Mar 2023 14:42:44 +0300
Completed At: Fri, 10 Mar 2023 14:43:24 +0300
Duration: 40s
Pods Statuses: 0 Active / 1 Succeeded / 0 Failed
Pod Template:
  Labels: app=kube-hunter
         controller-uid=5330c3d2-47cb-48a1-9cd2-785ee3c579f1
         job-name=kube-hunter
  Containers:
    kube-hunter:
      Image: aquasec/kube-hunter:0.6.8
      Port: <none>
      Host Port: <none>
      Command:
        kube-hunter
      Args:
        --pod
      Environment: <none>
      Mounts: <none>
      Volumes: <none>
Events:
  Type    Reason             Age   From           Message
  ----    -
  Normal  SuccessfulCreate   22m   job-controller  Created pod: kube-hunter-6wphh
  Normal  Completed          21m   job-controller  Job completed

```

Видим, что по рабочей задаче был создан pod - результаты сканирования будут отображены в его логах. Ознакомимся с ними (подставьте в команду свой идентификатор pod):

```
$ kubectl logs kube-hunter-6wphh
```


Vulnerabilities

For further information about a vulnerability, search its ID in:
<https://avd.aquasec.com/>

ID	LOCATION	MITRE CATEGORY	VULNERABILITY	DESCRIPTION	EVIDENCE
None	Local to Pod (kubernetes-hunter-6wphh)	Lateral Movement // ARP poisoning and IP spoofing	CAP_NET_RAW Enabled	CAP_NET_RAW is enabled by default for pods. If an attacker manages to compromise a pod, they could potentially take advantage of this capability to perform network attacks on other pods running on the same node	
KHV002	10.43.0.1:443	Initial Access // Exposed sensitive interfaces	K8s Version Disclosure	The kubernetes version could be obtained from the /version endpoint	v1.21.7+k3s1
KHV005	10.43.0.1:443	Discovery // Access the K8S API Server	Access to API using service account token	The API Server port is accessible. Depending on your RBAC settings this could expose access to or control of your cluster.	b'{"kind":"APIVersions","versions":["v1"],"serverAddressByClientCIDRs":[{"clientCIDR":"0.0.0.0/0","s...
None	Local to Pod (kubernetes-hunter-6wphh)	Credential Access // Access container service account	Access to pod's secrets	Accessing the pod's secrets within a compromised pod might disclose valuable data to a potential attacker	['/var/run/secrets/kubernetes.io/serviceaccount/ca.crt', '/var/run/secrets/kubernetes.io/serviceacco...
KHV050	Local to Pod (kubernetes-hunter-6wphh)	Credential Access // Access container service account	Read access to pod's service account token	Accessing the pod service account token gives an attacker the option to use the server API	eyJhbGciOiJIUzI1NiIsImtpZCI6IkdVbGVuZGNoWm93a3NFdkVNaZFMThuOWR0VUZseTgzS2o1TW5aVzBCM1MtakkiQ.eyJhdWQiOi...

Как мы видим, сканирование из pod более объемный, точный и таргетированный

--