22Kubernetes 系列（十七）增强 Kubernetes 安全的 7个 开源工具

Kubernetes 系列（十七）增强 Kubernetes 安全的 7个 开源工具

本文探讨了如何在开源工具的帮助下保护 Kubernetes生产集群。作为先决条件，你需要具备Docker和 Kubernetes 的基本初级知识。

简而言之，Kubernetes是一个容器编排工具，Docker 是一个容器平台。由云服务提供商管理的一些最著名的 Kubernetes 集群包括 AWS EKS、Azure AKS 和 谷歌CKE。

在接下来的章节中，我们将讨论各种对保护Kubernetes集群有用的开源工具，包括有助于Docker 镜像的静态扫描、安全审计、加固 Kubernetes 集群和实现运行时安全的代码片段。

静态扫描 Docker 镜像

Dockerfile 是一个纯文本文件，它包含一组指令，提供了创建 Docker 镜像的配置。此外，容器是Docker 镜像的运行实例，Kubernetes 支持 Docker 运行时。

强烈建议在将 Docker 镜像推送到 Kubernetes 运行时环境之前扫描它们的安全漏洞。这将帮助您避免供应链攻击和通过左移提高安全性。

```
docker scan
```

Docker Engine 现在已经集成了 Synk 的扫描功能，以识别镜像中的漏洞。扫描的输出是 CVE 列表和建议。

以下是可用于执行安全扫描的开源工具，可以集成到CI/CD管道任务中，在构建应用程序时扫描图像:

1. Trivy , 操作系统层面漏洞检测的扫描工具
2. Clair ,容器漏洞静态扫描工具
3. Kube-bench , 这个工具可以判断Kubernetes是否得到了最优部署
4. kubeaudit , 一个针对常见安全控制，对Kubernetes Deployment 进行安全审计的工具
5. Kubescape , 一个检查Kubernetes是否按照主要的合规框架部署的工具
6. kube-hunter , 发现并利用漏洞的渗透测试工具
7. Sysdig Falco , Kubernetes集群风险和威胁检测的运行时安全解决方案

1.利用 Trivy 进行漏洞检测

Trivy是一个开源的扫描工具，对于希望实现静态应用程序安全性测试和快速扫描的团队来说，它是一个很好的选择。它为操作系统、作为代码文件的基础设施和语言包提供全面的漏洞检测，还用于检测配置问题。Trivy还会自动更新漏洞数据库。

在执行Trivy扫描之后，您将得到一个漏洞列表、它们的严重性和一个CVE编号(如果存在的话)。

安装

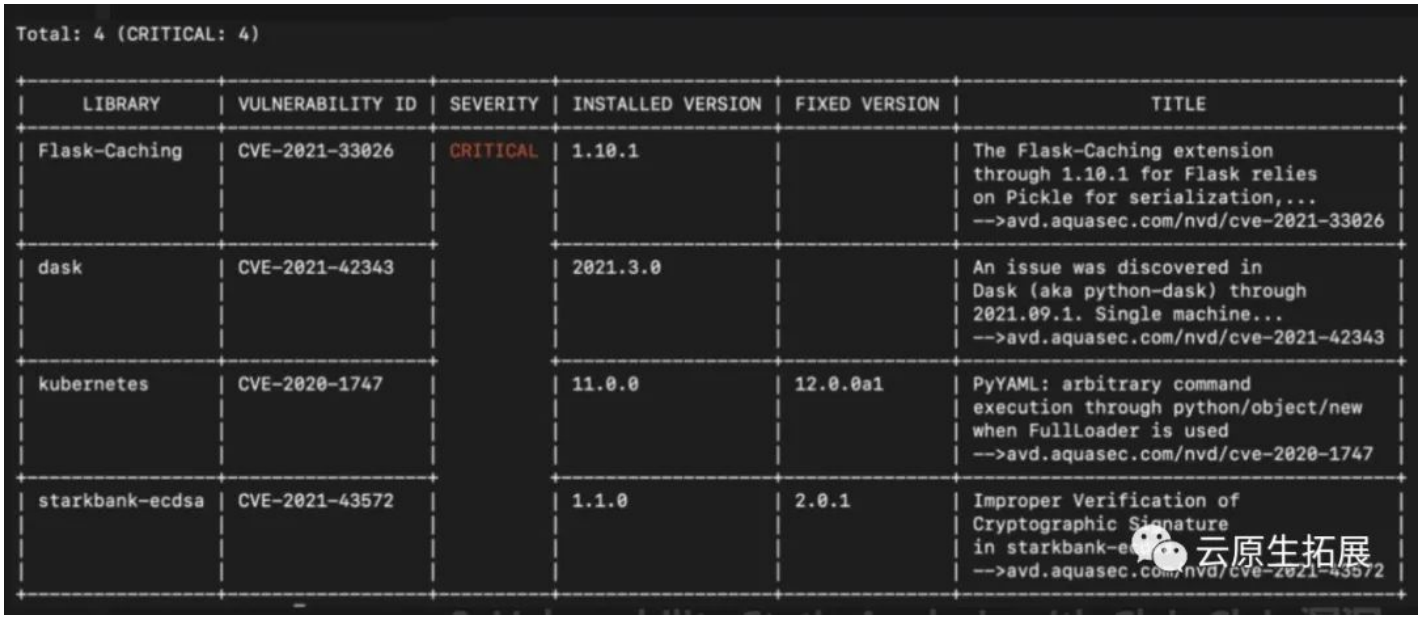
这个脚本将根据您的操作系统下载和安装Trivy。

```
curl -sfL https://raw.githubusercontent.com/aquasecurity/trivy/main/contrib/install.sh | sh -s -- -b /usr/local/bin v0.21.1
```

Scanning and output

```
trivy image --severity CRITICAL,HIGH apache/airflow:3.4-alpine
```

上面的命令会扫描 Apache Airflow 镜像，查找严重和高危的漏洞。一旦完成，扫描报告看起来像这样:



```
Total: 4 (CRITICAL: 4)

+---------------+-----------------+----------+-------------------+---------------+---------------------------------------+
|    LIBRARY    | VULNERABILITY ID | SEVERITY | INSTALLED VERSION | FIXED VERSION |                 TITLE                 |
+---------------+-----------------+----------+-------------------+---------------+---------------------------------------+
| Flask-Caching | CVE-2021-33026  | CRITICAL | 1.10.1            |               | The Flask-Caching extension           |
|               |                 |          |                   |               | through 1.10.1 for Flask relies       |
|               |                 |          |                   |               | on Pickle for serialization,...       |
|               |                 |          |                   |               | -->avd.aquasec.com/nvd/cve-2021-33026 |
+---------------+-----------------+          +-------------------+---------------+---------------------------------------+
| dask          | CVE-2021-42343  |          | 2021.3.0          |               | An issue was discovered in            |
|               |                 |          |                   |               | Dask (aka python-dask) through        |
|               |                 |          |                   |               | 2021.09.1. Single machine...          |
|               |                 |          |                   |               | -->avd.aquasec.com/nvd/cve-2021-42343 |
+---------------+-----------------+          +-------------------+---------------+---------------------------------------+
| kubernetes    | CVE-2020-1747   |          | 11.0.0            | 12.0.0a1      | PyYAML: arbitrary command             |
|               |                 |          |                   |               | execution through python/object/new   |
|               |                 |          |                   |               | when FullLoader is used               |
|               |                 |          |                   |               | -->avd.aquasec.com/nvd/cve-2020-1747  |
+---------------+-----------------+          +-------------------+---------------+---------------------------------------+
| starkbank-ecdsa | CVE-2021-43572 |          | 1.1.0             | 2.0.1         | Improper Verification of              |
|               |                 |          |                   |               | Cryptographic Signature               |
|               |                 |          |                   |               | in starkbank-ec                       |
|               |                 |          |                   |               | -->avd.aquasec.com/nvd/cve-2021-43572 |
+---------------+-----------------+----------+-------------------+---------------+---------------------------------------+
```


更多详细使用，可以参考 trivy 开源站点 Trivy(https://aquasecurity.github.io/trivy/v0.30.4/)）。

2. Clair 漏洞静态分析

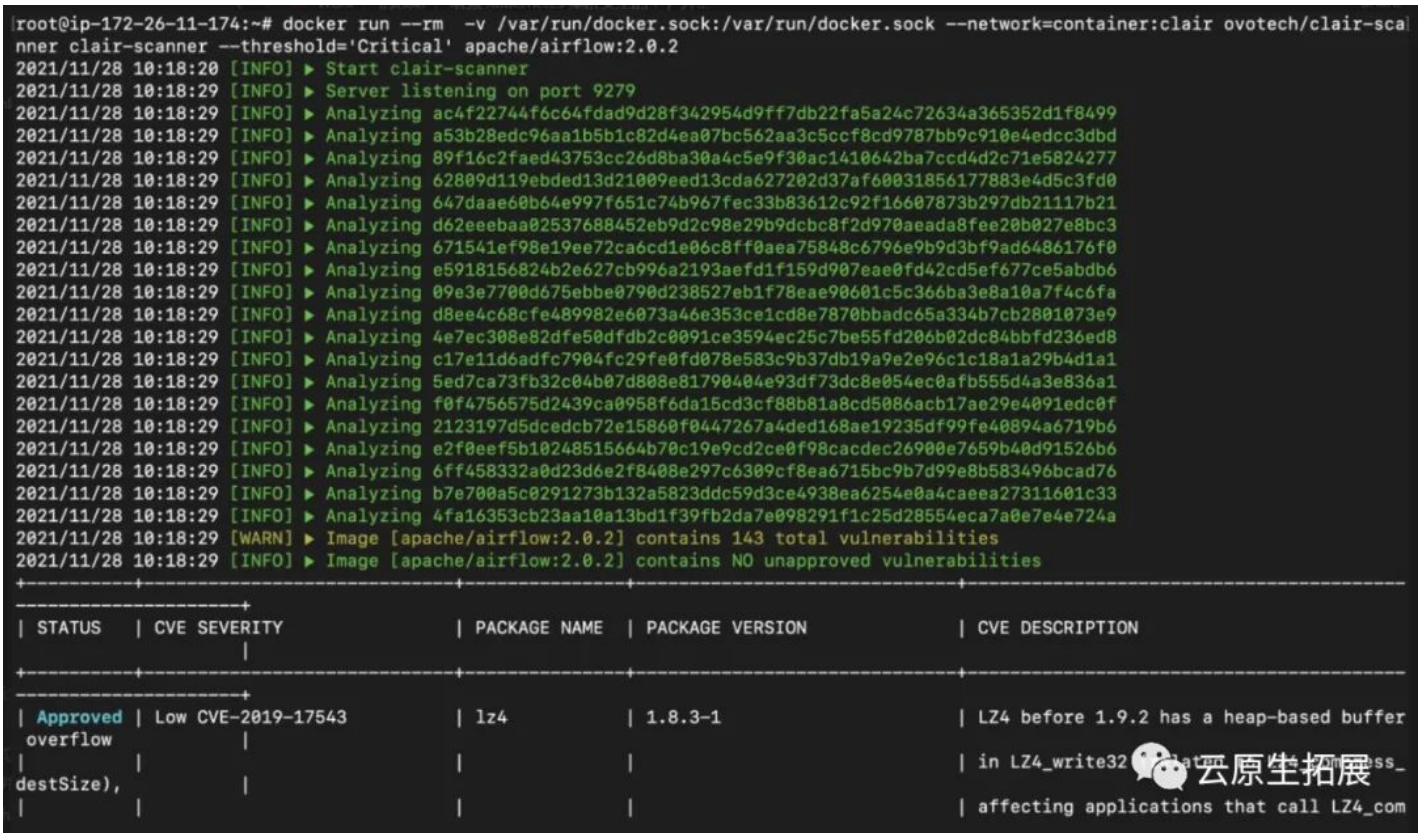
Clair 是一个开放源码的容器漏洞静态扫描工具。该工具有多种部署模式，最适合高可伸缩性和可用性。Clair 支持REST api，并提供HTML扫描报告。Amazon Elastic Container Registry (Amazon ECR)使用来自 Clair 项目的 CVES 数据库，并提供了一个发现列表。

安装及扫描

在这个例子中，我们使用的是预加载了CVE数据的 arminc/clair-db Docker 镜像。Clair -scanner将通过本地网络与 Clair 端点连接。

```
docker run -d --name db arminc/clair-db:latest
docker run -d --link db:postgres --name clair arminc/clair-local-scan:v2.0.6
docker run --rm -v /var/run/docker.sock:/var/run/docker.sock --network=container:clair ovotech/clair-scanner clair-scanner --thresh
```

一旦 Clair 扫描完成，你会看到下面的输出:



如果愿意，还可以获得JSON格式的输出报告。

在大多数情况下，解决Clair报告的漏洞的最佳方案是使用内置的包管理器更新底层操作系统，或者更新到Docker 镜像的最新版本。如果脆弱包的数量太多，无法管理，那么可以考虑使用无害的基础映像：https://github.com/GoogleContainerTools/distroless。

保护和审计您的 Kubernetes Deployment

为了改善 Kubernetes 的安全状况，您需要定期审计您的部署，并根据安全基准对设置执行安全扫描。本节主要介绍如何使用各种开放源码工具对Kubernetes集群进行审计和保护。

3. 使用 Kube-bench 实现 CIS Benchmarking

CIS Benchmarking 是保护 IT 系统和数据的全球公认标准，它提供了一系列的指导手册来保护容易受到网络攻击的操作系统、软件和网络。

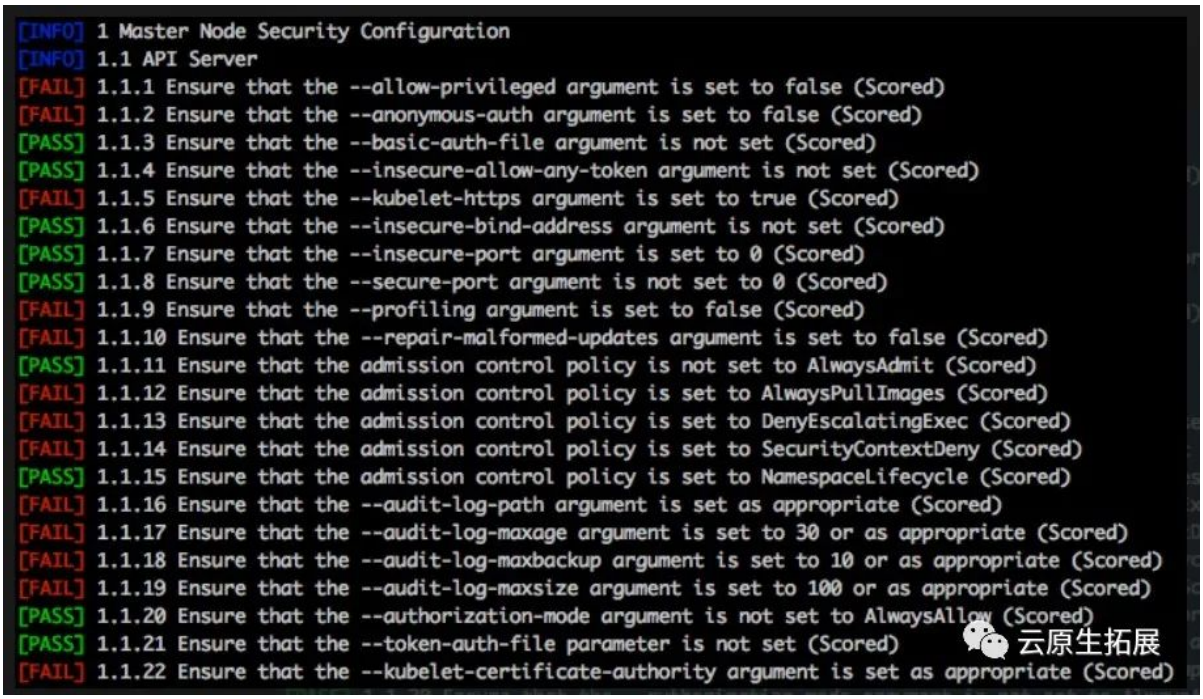
Kube-bench是一个开源工具，它检查Kubernetes是否按照CIS Kubernetes基准(包含一组Kubernetes安全最佳实践)优化部署。因此，当仅为CIS基准测试目的需要扫描时，kube-bench是最好的。

你可以在 POD 中运行kube-bench。GitHub存储库包含特定于云的 job- .yaml 文件，kube-bench将根据机器上运行的Kubernetes版本自动决定运行哪个测试集。

Running kube-bench

```
git clone https://github.com/aquasecurity/kube-bench.git
cd kube-bench

kubectl apply -f job.yaml
```



Source: https://github.com/aquasecurity/kube-bench/

为了减轻kube-bench报告中的任何问题，请参阅Kubernetes CIS官方基准文档，其中包含每个发现的详细信息和补救说明。

4. 使用 kubeaudit 来审计 Kubernetes

由Shopify制作的 kubeaudit 是一个开源工具，它可以针对常见的安全控制对Kubernetes的部署进行审计，比如:

- 以非root用户运行
- 使用只读的根文件系统
- 放弃易受攻击的能力，不要添加新的
- 不要特权运行

Installation and execution

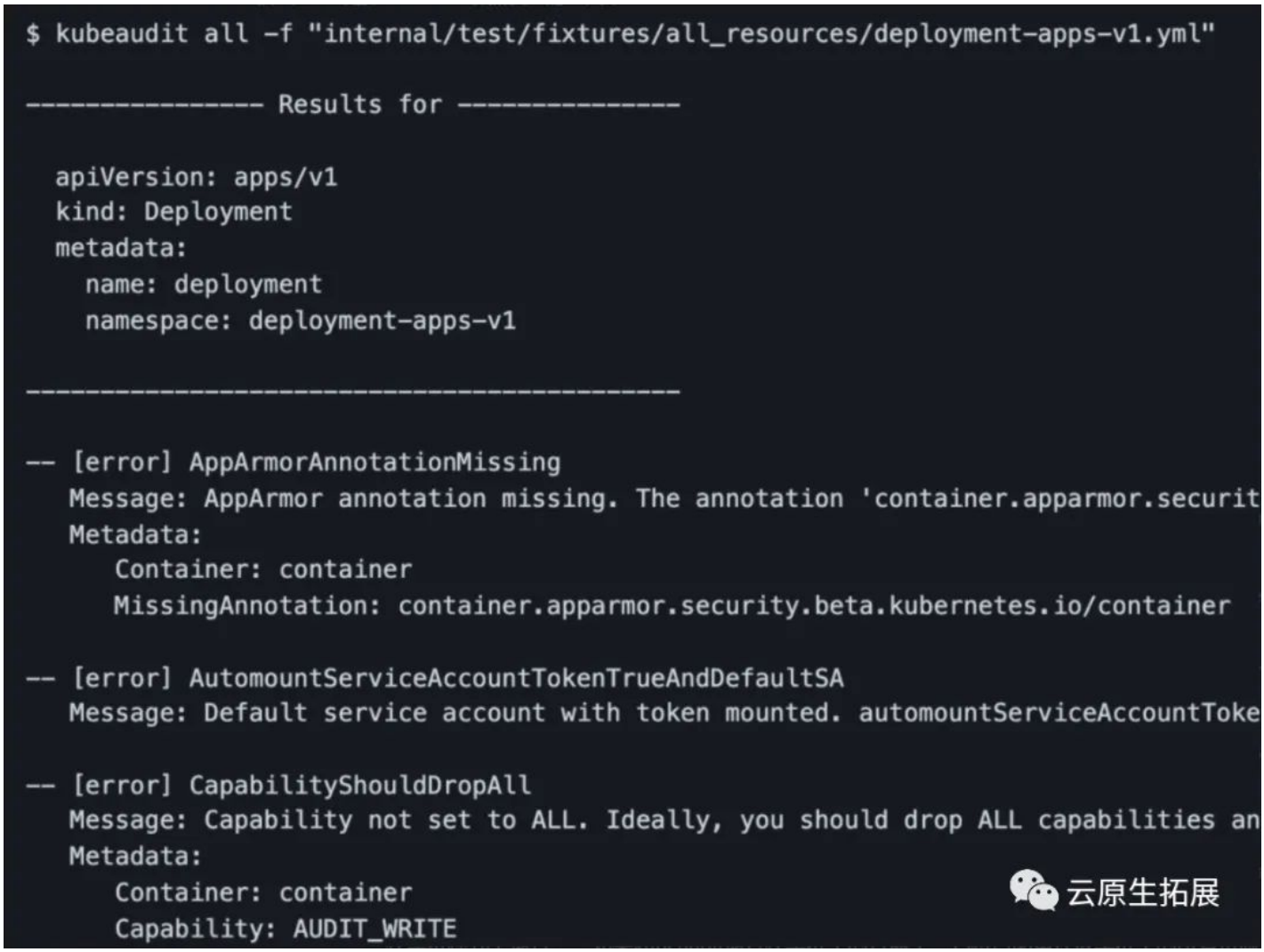
```
go get -v github.com/Shopify/kubeaudit
```

Kubeaudit 可以在三种不同的模式下运行:manifest、cluster和local。它还可以自动修复清单，这使它有别于其他竞争对手。

manifest模式运行—在该模式下，需要提供相关Kubernetes资源的manifest文件。

```
kubeaudit all -f "/path/to/manifest.yml"
kubeaudit autofix -f "/path/to/manifest.yml"
```

一旦完成，你会看到如下内容:



在集群模式下运行——如果kubeaudit运行在集群中的容器中，它将自动审计该集群中的所有资源。

```
kubeaudit all
```

运行在本地模式-当kubeaudit在本地机器上运行时，它将尝试连接到Kubernetes集群，通过从$HOME/获取详细信息。默认是Kube /config文件。另外，你也可以为Kube配置提供一个路径:

```
kubeaudit all -f "/path/to/config"
```

此外，kubeaudit有多个审计配置文件，包括外观，能力，限制，特权，rootfs, seccomp, netpols和asat。

5. 使用Kubescape进行安全测试

Kubescape 是一个开源工具，用于检查Kubernetes的部署是否符合几乎所有主要的合规框架，包括NSA-CISA和MITRE ATT&CK®，以及DevSecOps的最佳实践。Kubescape还可以与CI工具集成。

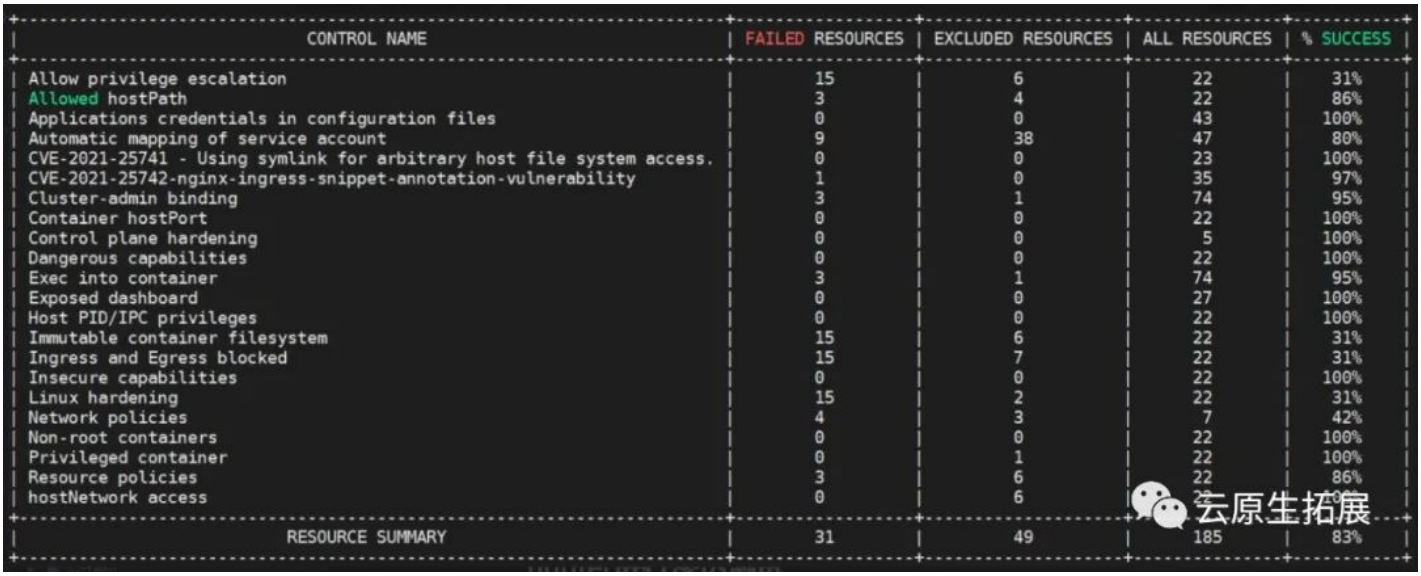
Installation

```
curl -s https://raw.githubusercontent.com/armosec/kubescape/master/install.sh | /bin/bash
```

Running

```
kubescape scan framework mitre --submit --format json --output results.json
kubescape scan framework nsa --submit --include-namespaces development,staging,production
```

运行Kubescape之后，你会看到如下内容:



CONTROL NAME	FAILED RESOURCES	EXCLUDED RESOURCES	ALL RESOURCES	% SUCCESS
Allow privilege escalation	15	6	22	31%
Allowed hostPath	3	4	22	86%
Applications credentials in configuration files	0	0	43	100%
Automatic mapping of service account	9	38	47	80%
CVE-2021-25741 - Using symlink for arbitrary host file system access.	0	0	23	100%
CVE-2021-25742-nginx-ingress-snippet-annotation-vulnerability	1	0	35	97%
Cluster-admin binding	3	1	74	95%
Container hostPort	0	0	22	100%
Control plane hardening	0	0	5	100%
Dangerous capabilities	0	0	22	100%
Exec into container	3	1	74	95%
Exposed dashboard	0	0	27	100%
Host PID/IPC privileges	0	0	22	100%
Immutable container filesystem	15	6	22	31%
Ingress and Egress blocked	15	7	22	31%
Insecure capabilities	0	0	22	100%
Linux hardening	15	2	22	31%
Network policies	4	3	7	42%
Non-root containers	0	0	22	100%
Privileged container	0	1	22	100%
Resource policies	3	6	22	86%
hostNetwork access	0	6	22	100%
RESOURCE SUMMARY	31	49	185	83%

Source: https://github.com/armosec/kubescape

6. 用kube-hunter进行渗透测试

kube-hunter是用Python编写的一个开放源码渗透测试工具，它使您能够编写自定义模块，这些模块可以在本地机器、集群内部以及主动和被动模式下远程执行。

在活动模式下，kube-hunter将发现并进一步利用任何漏洞。

```
pip install kube-hunter # Installing Kube-hunter.
kube-hunter --remote some.node.com # Running remotely.
kube-hunter --cidr 192.168.0.0/24. # Network Scanning.
kube-hunter --remote some.domain.com --active # Active Mode.
kube-hunter --list --active # List of test cases.
kube-hunter --remote some.node.com --json # Json output
kube-hunter --k8s-auto-discover-nodes --kubeconfig "/path/config"
```

此外，还可以将kube-hunter作为恶意POD 在 staging 环境中运行，它可以充当真实的攻击模拟。

前面的三个工具——kubeaudit、Kubescape和kube-hunter——都提供了关于错误配置、漏洞、遗漏检查、不安全实践等的详细报告。强烈建议您遵循一些Kubernetes安全加固指南，以随着时间的推移改善您的安全状况，并使用这些工具进行验证。更多信息，请查看美国国家安全局和CISA最近发布的Kubernetes强化指南。

7. 运行时安全 Sysdig Falco

sysydig Falco是一个开源的运行时安全解决方案，用于跨Kubernetes集群的持续风险和威胁检测。该工具充当一个安全摄像头，持续实时检测意外行为、配置更改、入侵和数据窃取。目前，sysydig Falco是唯一的cncf批准的Kubernetes运行时安全的开源解决方案。

Falco可以让你编写自定义插件，并拥有其商业竞争对手(例如Aquasec和Twistlock)的所有主要特性，它使用内核模块或eBPF探针等驱动器来消耗原始的系统调用信息流。Falco可以作为守护进程在Kubernetes节点上运行，它可以通过stdout、HTTP钩子和syslog以JSON格式监控和传输实时安全警报。

Installation with HELM

你可以使用Helm Chart 来安装Sysdig Falco守护进程，并使用自定义参数:

```
helm repo add falcosecurity https://falcosecurity.github.io/charts
helm repo update
helm install falco -f values.yaml falcosecurity/falco
```

Falco 安全规则

Falco为运行时安全、Kubernetes控制平面审计和应用程序安全提供了80多个默认安全规则。您可以根据自己的需求定制提供的基于yaml的规则。下面的表格解释了一些更常见的规则:

Rule name	Tags
Detecting CryptoJacking	Mitre
Read Sensitive File Untrusted	File Integrity Monitoring
Launch of Privileged Containers	CIS
Suspicious Network Tool in Container	Network, Mitre
Launch Remote File Copy Tools in Container	Exfiltration

Falco Security Alerts on Mattermost

Falcosidekick是一个守护进程，它收集Falco日志，并将它们传输到各种渠道，例如，Mattermost, Elasticsearch, S3和Kafka。

```
helm install falco falcosecurity/falco -f values.yaml --set falcosidekick.enabled=true --set falcosidekick.webui.enabled=true
```

我们可以在falcosidekick Chart 的 values.yaml 中配置Mattermost。你需要利用Mattermost的传入webhooks特性来利用它。

```
config:
    mattermost:
    webhookurl: "" # (ex: http://XXXX/hooks/YYYY)
    #footer: "" # Mattermost footer
    #icon: "" # Mattermost icon (avatar)
    #username: "" # Mattermost username (default: Falcosidekick)
    outputformat: "all"
    minimumpriority: "debug"
    # messageformat: "**{{ .Rule }}** triggered by user **{{ index .OutputFields \"user.name\" }}**".
    # mutualtls: false
    # checkcert: true
```

一旦启用，最重要频道的安全警报将是这样的:

Source

Falco被要求自动响应它收到的安全警报——比如删除恶意文件，删除受感染的pod，以及应用防火墙规则。 该工具可以与Kubeless、Knative和Argo等公司的主动响应集成。

要了解更多信息，阅读本文(https://falco.org/blog/falcosidekick-response-engine-part-1-kubeless/)。

总结

正如您所看到的，有一个非常丰富的开源工具生态系统，您可以使用它来提高Kubernetes的安全性。在GitHub上有很多工具可以让你执行自动扫描、CI集成、运行时安全、渗透测试和审计检查。使用本文作为实现Kubernetes安全程序并确保应用程序、网络和数据安全的参考指南。

欢迎关注我的公众号“**云原生拓展**”，原创技术文章第一时间推送。