VSRC2019城市沙龙

容器运行时安全的建设实践

唯品会信息安全部 吴辉

VSRC 2019-07-27

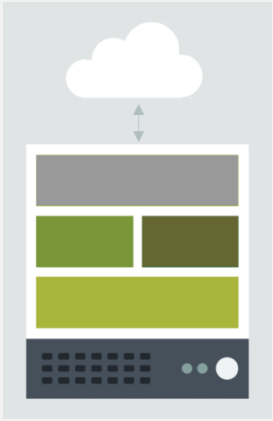
唯品会安全应急响应中心

目录

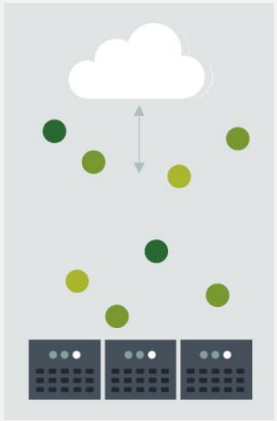
CONTENTS

01 容器下的安全挑战

软件服务容器化



传统应用服务



容器化微服务

- 弹性伸缩
- 高可用性
- 自动化运营

Kubernetes & Docker正在改变传统软件服务

唯品会容器云平台

NOAH云平台：基于容器的虚拟化技术，提供高性能的、弹性伸缩的、高可管理性的，并结合业界最佳实践与唯品会运维流程的基础云平台，提供业务镜像烘培、镜像管理、应用发布及运行的全流程支持。

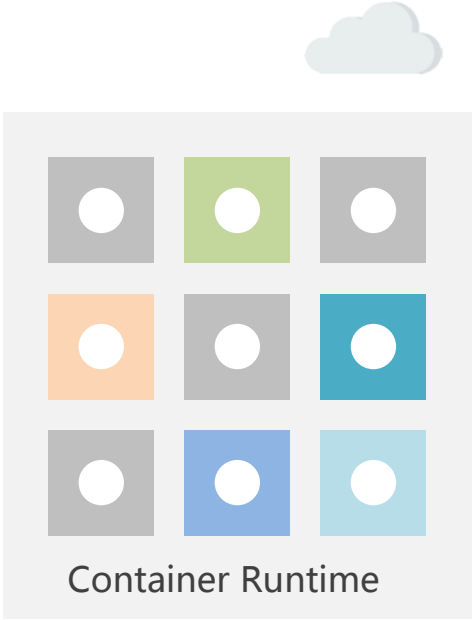
运行情况

- 已经接入900+个应用 ，其中341个核心域
- 2000+服务器，运行了10000+应用实例
- 14+k8s集群，最多一个集群588个节点
- 承担80%核心域流量

容器下的安全挑战

- 开源软件安全漏洞
- 内部网络攻击事件
- 容器内的横向移动
- 日益复杂的针对性攻击
- 安全可视化的盲点

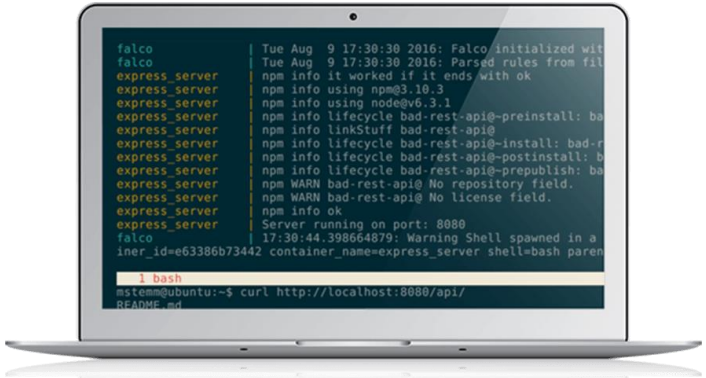
传统主机安全的盲点



针对容器内的安全事件，传统主机安全措施已不再适用，如何保障容器运行安全？

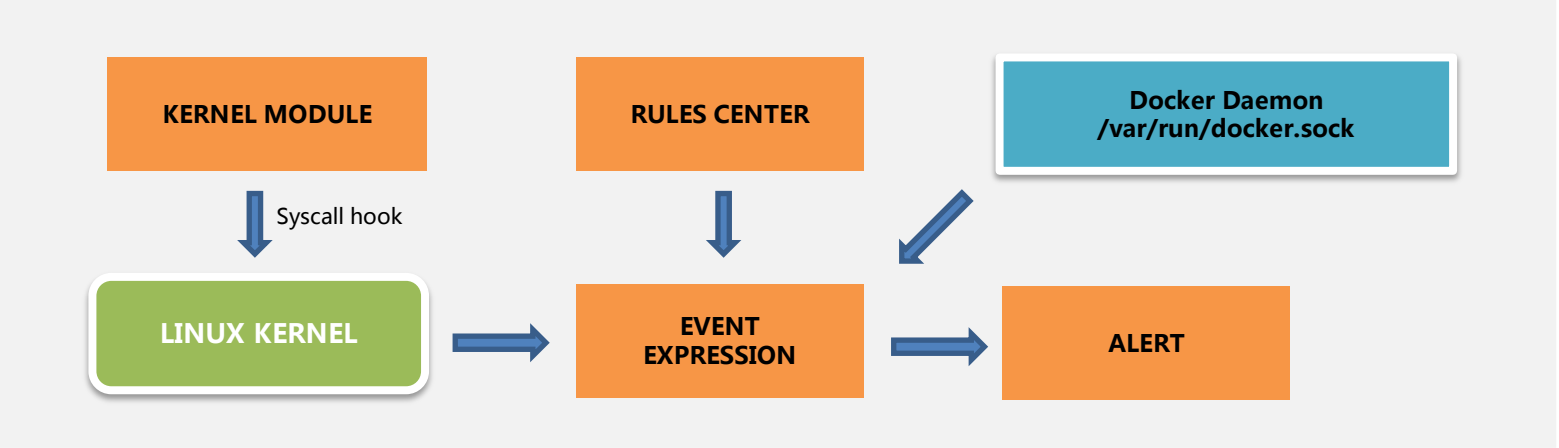
02 容器运行时安全

容器运行时安全



Container Native Runtime Security

常见容器安全监控架构



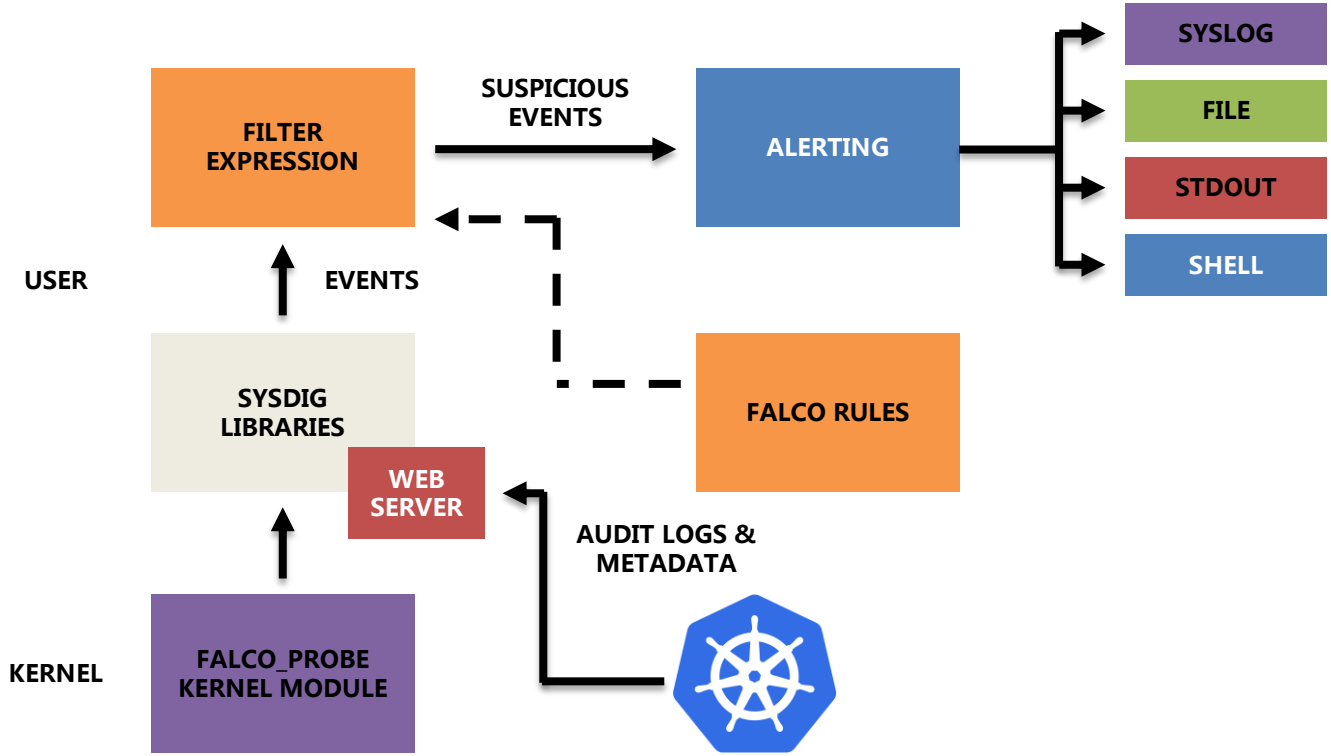
修改linux系统调用表实现宿主机内的事件活动监控，规则匹配分析异常行为，通过docker Daemon关联对应的容器

Sysdig Falco简介

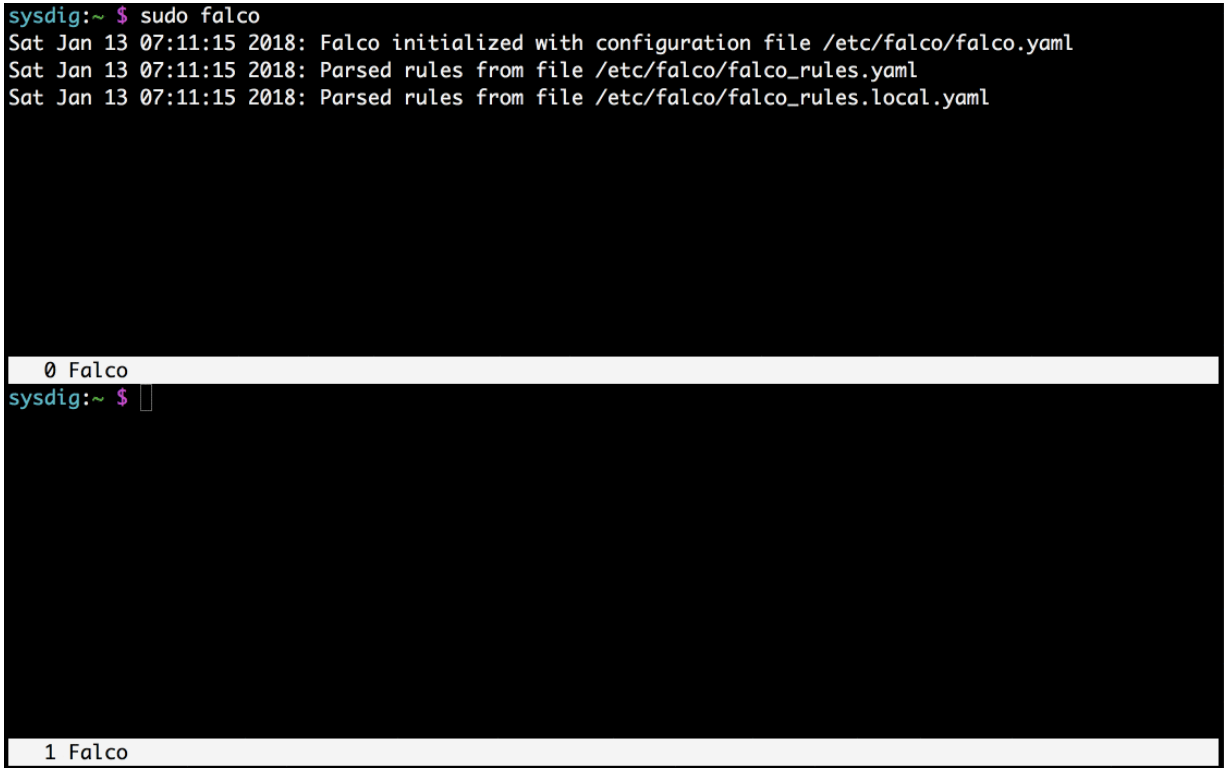
Falco是开源的用于原生容器平台的行为活动检测系统，旨在检测容器内部的异常行为。 在Sysdig系统调用捕获的基础架构的支持下，可通过制定一些列规则来检测容器平台内的应用程序、进程、网络等的异常行为。

- 基于apache协议的开源项目
- 云原生计算基金会(CNCF)沙箱托管项目
- https://falco.org/#about

Sysdig Falco系统架构

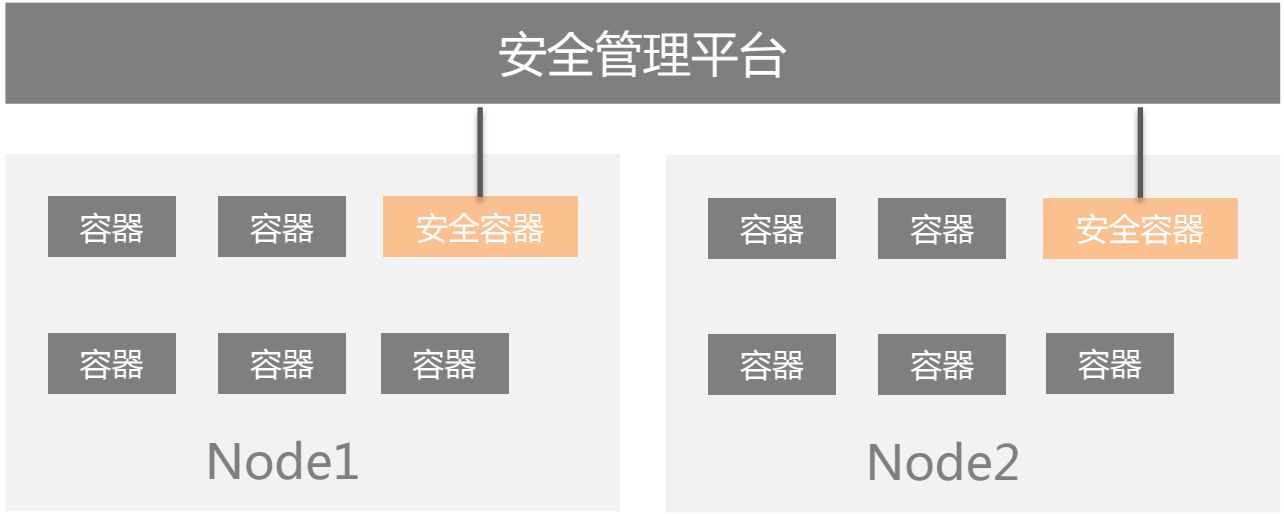


Falco测试案例


sysdig:~ $ sudo falco
Sat Jan 13 07:11:15 2018: Falco initialized with configuration file /etc/falco/falco.yaml
Sat Jan 13 07:11:15 2018: Parsed rules from file /etc/falco/falco_rules.yaml
Sat Jan 13 07:11:15 2018: Parsed rules from file /etc/falco/falco_rules.local.yaml
0 Falco
sysdig:~ $
1 Falco

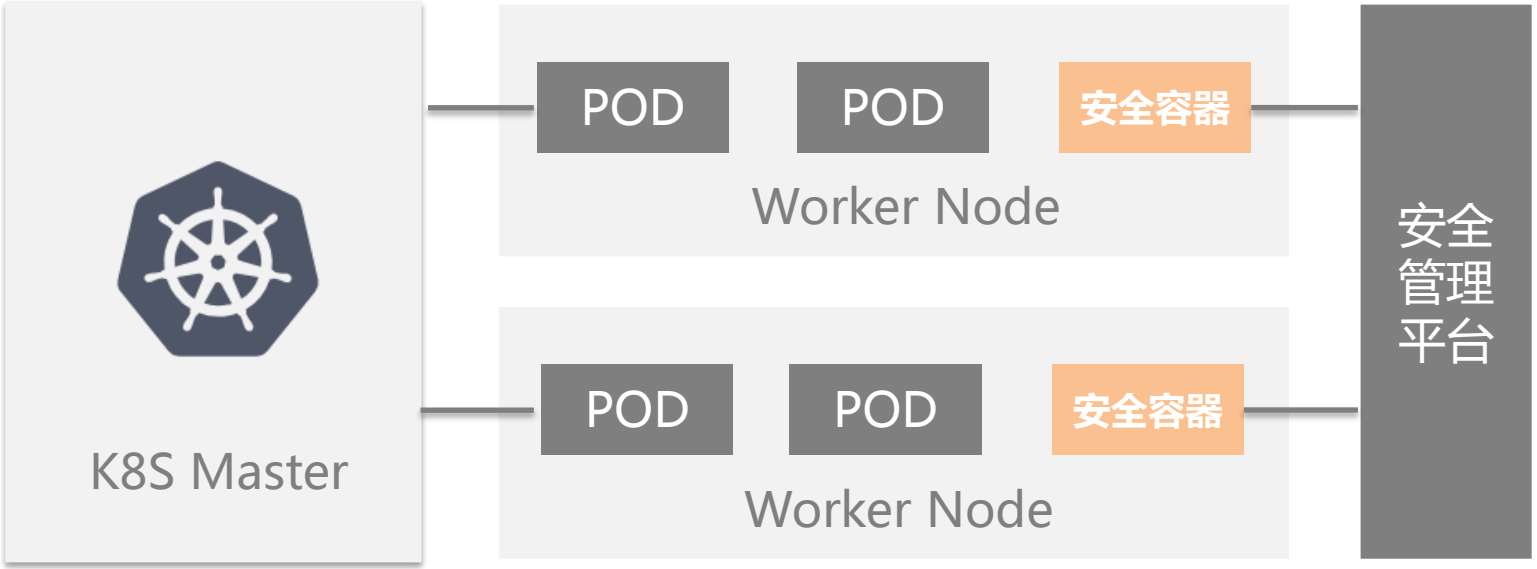
03 安全建设与实践

容器化的部署方案(一)



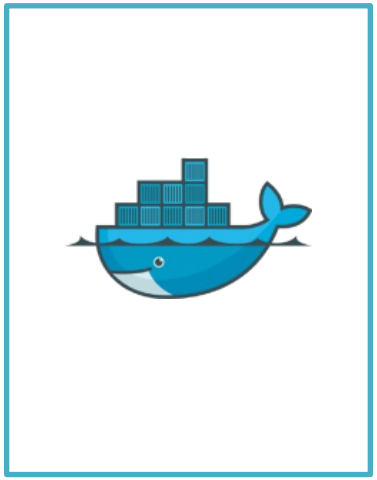
一，基于单机docker容器的安全监控部署方案

容器化的部署方案(二)



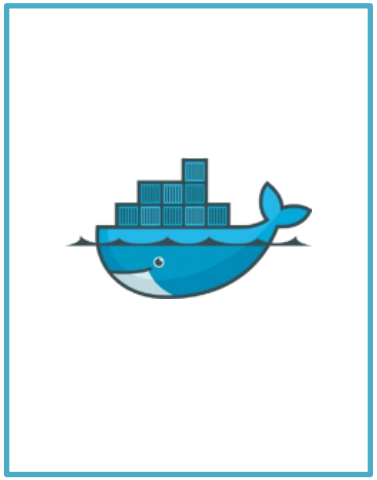
二，基于K8S集群的容器监控部署方案

安全监控镜像定制



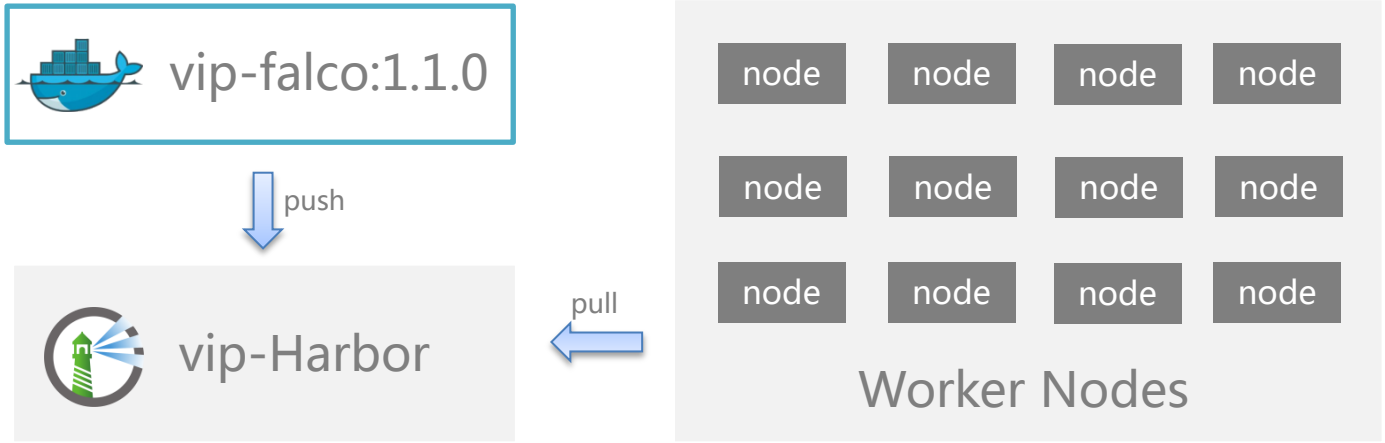
- 基于centos7
- Falco容器安全监控
- Sysdig容器全量事件收集
- health check监控
- Smart Agent监控插件
- 配置文件、规则文件动态更新插件

安全监控镜像加固



- 禁用无用账号和服务
- 设置系统目录读写执行权限
- 设置网络访问策略
- 容器运行时使用资源的限制
- /var/run/docker.sock访问权限控制

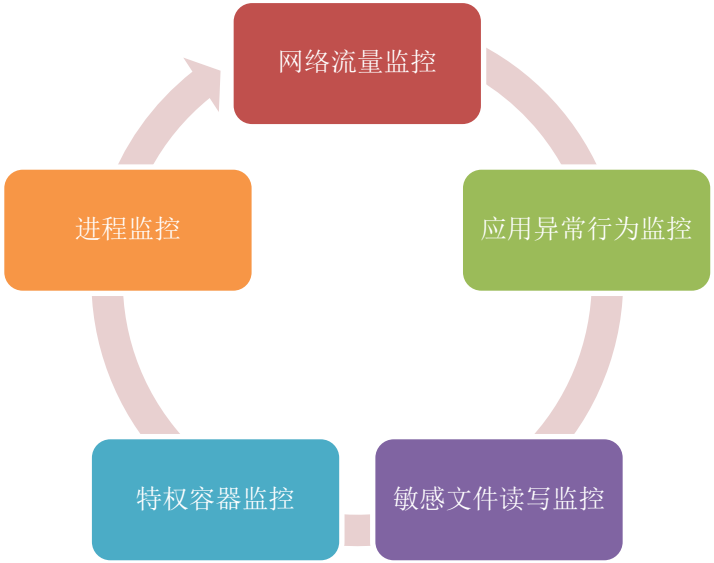
安全监控镜像的下发



vip-falco基础镜像不定期进行更新，push到harbor基础镜像列表，

宿主机在非高峰期预拉取安全镜像。

定制事件告警规则



容器内事件告警规则覆盖：

- 未经授权的网络出口/入口监控。
- 已知的对应用程序的网络攻击。
- 敏感文件指纹文件的读写监控。
- 容器进程权限升级，特权容器的行为监控
- 数据盗窃、反向Shell 通道、隐藏的网络管道。

K8S开启特权开关

1.在master节点中 文件/etc/kubernetes/apiserver 加入：--allow-privileged=true

```
# default admission control policies
# KUBE_ADMISSION_CONTROL="--admission-control=NamespaceLifecycle,NamespaceExists,LimitRanger,SecurityContextDeny,Servic
eAccount,ResourceQuota"
KUBE_ADMISSION_CONTROL="--admission-control=NamespaceLifecycle,NamespaceExists,LimitRanger,SecurityContextDeny,Resource
Quota"

# Add your own!
KUBE_API_ARGS="--allow-privileged=true"
```

2.在node节点中 文件/etc/kubernetes/kubelet 加入：--allow-privileged=true

```
# location of the api-server
KUBELET_API_SERVER="--api-servers=http://127.0.0.1:8080"

# pod infrastructure container
KUBELET_POD_INFRA_CONTAINER="--pod-infra-container-image=pod-infrastructure:latest"

# Add your own!
KUBELET_ARGS="--allow-privileged=true"
```

3.Master节点上重启apiserver服务，node节点上重启kubelet服务。

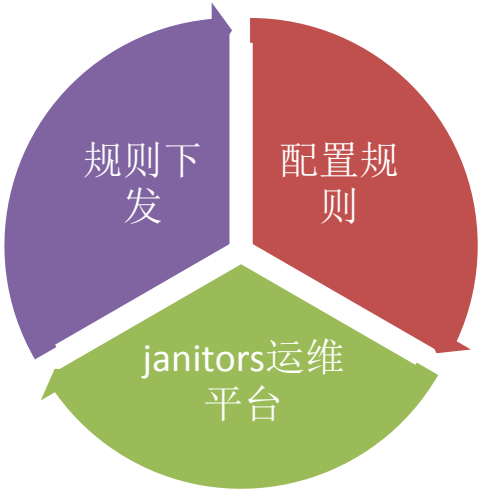
发布容器

```
image: harbor-***.vip.com/vip-falco:1.1.0
securityContext:
 privileged: true
volumeMounts:
 - mountPath: /host/var/run/docker.sock
  name: docker-socket
 - mountPath: /host/run/containerd/containerd.sock
  name: containerd-socket
 - mountPath: /host/dev
  name: dev-fs
  readOnly: true
 - mountPath: /host/proc
  name: proc-fs
  readOnly: true
 - mountPath: /host/boot
  name: boot-fs
  readOnly: true
 - mountPath: /host/lib/modules
  name: lib-modules
  readOnly: true
 - mountPath: /host/usr
  name: usr-fs
  readOnly: true
 - mountPath: /etc/falco
  name: conf-fs
  readOnly: true
```

```
volumes:
    - name: docker-socket
     hostPath:
      path: /var/run/docker.sock
    - name: containerd-socket
     hostPath:
      path: /run/containerd/containerd.sock
    - name: dev-fs
     hostPath:
      path: /dev
    - name: proc-fs
     hostPath:
      path: /proc
    - name: boot-fs
     hostPath:
      path: /boot
    - name: lib-modules
     hostPath:
      path: /lib/modules
    - name: usr-fs
     hostPath:
      path: /usr
    - name: conf-fs
     hostPath:
      path: /apps/conf/vip-falco
```

```
[apps@qa-k8s-1-master vip-falco]$ kubectl create -f falco-daemonset.yaml
daemonset "vip-falco" created
[apps@qa-k8s-1-master vip-falco]$ kubectl get pods
NAME                                          READY     STATUS    RESTARTS   AGE
vip-falco-dxzcr                                  1/1       Running   0          18s
vip-falco-kkht5                                  1/1       Running   0          18s
vip-falco-awadr                                  1/1       Running   0          18s
```

配置规则更新



配置更新流程

a. 后台编辑告警规则

b. 通过运维平台推送更新文件到k8s宿主机

c. 容器检测文件变动重新加载规则

d. 监控规则文件生效

事件日志的收集

Selected fields

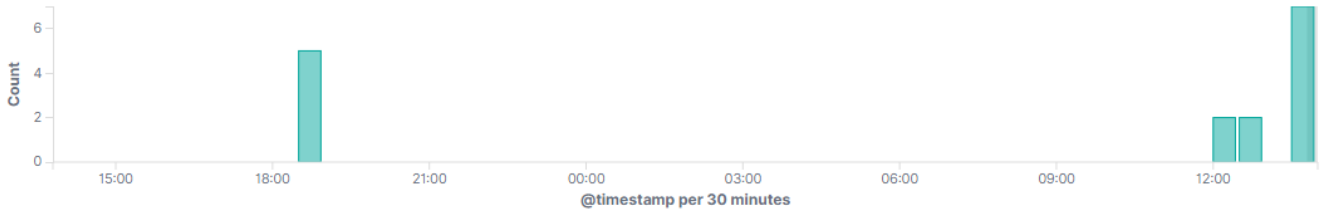
- t rule
- t user.name
- ? docker.container_n...
- ? docker.container_id
- ? docker.container_im...
- ? proc.cmdline
- ? fd.name

Available fields

Popular

- t docker.container_privile...
- t kubernetes.host
- t kubernetes.pod_name
- t kubernetes.pod_id
- t kubernetes.pod_label
- t container.name
- t priority

@timestamp



Time	docker.container_id	docker.container_name	user.name	proc.cmdline	fd.name	rule	docker.container_image
Jul 23, 2019 @ 13:47:19.280	136f0eceec68	elasticsearch01	elasticsearch	curl [redacted]	[redacted] 54:49114-> [redacted]	Notice Known system binary sent/received network traffic	elasticsearch:7.0.0
Jul 23, 2019 @ 13:42:39.269	136f0eceec68	elasticsearch01	elasticsearch	bash -c </dev/tcp/10.1 [redacted]/8080	[redacted] 54:49114-> [redacted] 54:8080	Notice Known system binary sent/received network traffic	elasticsearch:7.0.0
Jul 23, 2019 @ 13:35:09.249	84ef11fc31f5	tf_operator	root	cat /etc/shadow	cat /etc/shadow	Read sensitive file untrusted	tf_operator:0.0.4
Jul 23, 2019 @ 13:34:14.246	84ef11fc31f5	tf_operator	root	bash	bash	A shell was spawned in a container with an attached terminal	tf_operator:0.0.4

阶段性建设成果

安全规则覆盖常规主机安全事件，

更快的定位对应容器。

容器安全无感知 → 事件可监控

后续工作：

容器相资产管理系统，容器安全可视化建设。

04 总结与思考

总结

- 基础镜像功能的集成。
- 安全(特权)容器的性能监控和安全加固
- 告警规则的覆盖量及后续更新
- paas系统，容器审计日志分析接入
- 保障容器安全需要每一个环节

思考

当监控到容器内的安全事件时如何更快响应

安全监控 → 主动防御

安全监控 → 机器学习

感谢聆听

—

THANKS!

SRC
唯品会安全应急响应中心