容器安全产品形态及典型应用整理

三种运行方式

- 运行在容器内 缺点： 每个容器内部署一个Agent比较耗资源，对容器环境的侵入性较高。
- 运行在主机上 缺点：无法在CoreOS,RancherOS,Atomic操作系统上运行。
- 容器化部署运行，包括Sidecar和Daemonset模式。

容器操作系统

- CoreOS

基于Chrome OS再定制的轻量级Linux发行版本，剔除了其他对于服务器系统非核心的软件，比如GUI和包管理器。在 CoreOS 中，所有应用程序都被装在一个个 Docker容器中。

- RancherOS

Rancher OS是生产规模中运行Docker最小，最简单的方式。RancherOS 的所有东西都作为Docker管理的容器。这些系统服务包括udev 和rsyslog。RancherOS仅仅包括最少运行Docker所需要的软件。

结论：titanagent支持容器操作系统上安装的难度和支持容器化部署的难度一致，都需要打包成镜像。

容器能力及特权

Docker在镜像启动时，可以为容器添加能力(Capability)及特权

```
--cap-add: Add Linux capabilities
--cap-drop: Drop Linux capabilities
--privileged=false: Give extended privileges to this container
--device=[]: Allows you to run devices inside the container without the --privileged flag.
```

linux内核的Capabilities大部分是不区分namespace的，如果容器内进程拥有某个Capability，就和主机上进程有相同的能力。 docker为了管控容器进程的能力，默认删除了容器的部分Capability。

容器默认拥有能力

Capability Key	Capability Description
SETPCAP	Modify process capabilities.
MKNOD	Create special files using mknod(2).
AUDIT_WRITE	Write records to kernel auditing log.
CHOWN	Make arbitrary changes to file UIDs and GIDs (see chown(2)).
NET_RAW	Use RAW and PACKET sockets.
DAC_OVERRIDE	Bypass file read, write, and execute permission checks.
FOWNER	Bypass permission checks on operations that normally require the file system UID of the process to match the UID of the file.
FSETID	Don't clear set-user-ID and set-group-ID permission bits when a file is modified.
KILL	Bypass permission checks for sending signals.
SETGID	Make arbitrary manipulations of process GIDs and supplementary GID list.
SETUID	Make arbitrary manipulations of process UIDs.
NET_BIND_SERVICE	Bind a socket to internet domain privileged ports (port numbers less than 1024).
SYS_CHROOT	Use chroot(2), change root directory.
SETFCAP	Set file capabilities.

容器默认删除能力

Capability Key	Capability Description
SYS_MODULE	Load and unload kernel modules.
SYS_RAWIO	Perform I/O port operations (iopl(2) and ioperm(2)).
SYS_PACCT	Use acct(2), switch process accounting on or off.
SYS_ADMIN	Perform a range of system administration operations.
SYS_NICE	Raise process nice value (nice(2), setpriority(2)) and change the nice value for arbitrary processes.
SYS_RESOURCE	Override resource Limits.
SYS_TIME	Set system clock (settimeofday(2), stime(2), adjtimex(2)); set real-time (hardware) clock.
SYS_TTY_CONFIG	Use vhangup(2); employ various privileged ioctl(2) operations on virtual terminals.
AUDIT_CONTROL	Enable and disable kernel auditing; change auditing filter rules; retrieve auditing status and filtering rules.
MAC_ADMIN	Allow MAC configuration or state changes. Implemented for the Smack LSM.
MAC_OVERRIDE	Override Mandatory Access Control (MAC). Implemented for the Smack Linux Security Module (LSM).
NET_ADMIN	Perform various network-related operations.
SYSLOG	Perform privileged syslog(2) operations.
DAC_READ_SEARCH	Bypass file read permission checks and directory read and execute permission checks.
LINUX_IMMUTABLE	Set the FS_APPEND_FL and FS_IMMUTABLE_FL i-node flags.
NET_BROADCAST	Make socket broadcasts, and listen to multicasts.
IPC_LOCK	Lock memory (mlock(2), mlockall(2), mmap(2), shmctl(2)).
IPC_OWNER	Bypass permission checks for operations on System V IPC objects.
SYS_PTRACE	Trace arbitrary processes using ptrace(2).
SYS_BOOT	Use reboot(2) and kexec_load(2), reboot and load a new kernel for later execution.
LEASE	Establish leases on arbitrary files (see fcntl(2)).
WAKE_ALARM	Trigger something that will wake up the system.
BLOCK_SUSPEND	Employ features that can block system suspend.

非特权容器实现安全能力

1. 容器启动时添加相关能力

- SYS_ADMIN: 包括很多管理能力，包括mount、setdomainname等，在容器内执行的相关操作在主机上生效。
- NET_ADMIN: 能在容器内控制主机防火墙、iptable等。
- AUDIT_CONTROL: 实现Audit监控，能在容器内监控主机上所有Audit事件。链接: https://lwn.net/Articles/699819/
- SYS_PTRACE: 能在容器内ptrace主机进程(包括其他容器内的进程)。

2. 容器启动时映射主机路径

- /： 读取主机文件系统
- /proc: 获取所有进程信息
- /var/lib/docker: 读取镜像和容器文件系统(实现镜像扫描、容器扫描)
- /sys/fs/cgroup: 监控容器启动，获取容器资源占用(实现资源监控)
- /var/run/docker.sock: 访问docker api(实现docker资产清点)
- /usr/bin/docker: ptrace docker cli操作(实现docker操作审计)

产品：cAdvisor, Neuvector， Twistlock, StackRox

特权容器

以docker run --privileged运行的容器，特权包括：

- 开启后具有所有设备的读写权限（默认不允许访问设备）
- 配置 AppArmor & SELinux
- 加载内核驱动

产品: Sysdig, Falco

特权容器 **or** 非特权容器 ？

结论：非特权容器，启动时添加必要的能力(Capabilities)，并且映射必要的主机目录，即可实现大部分的安全功能。

容器模式：**Sidecar**和**Daemonset**

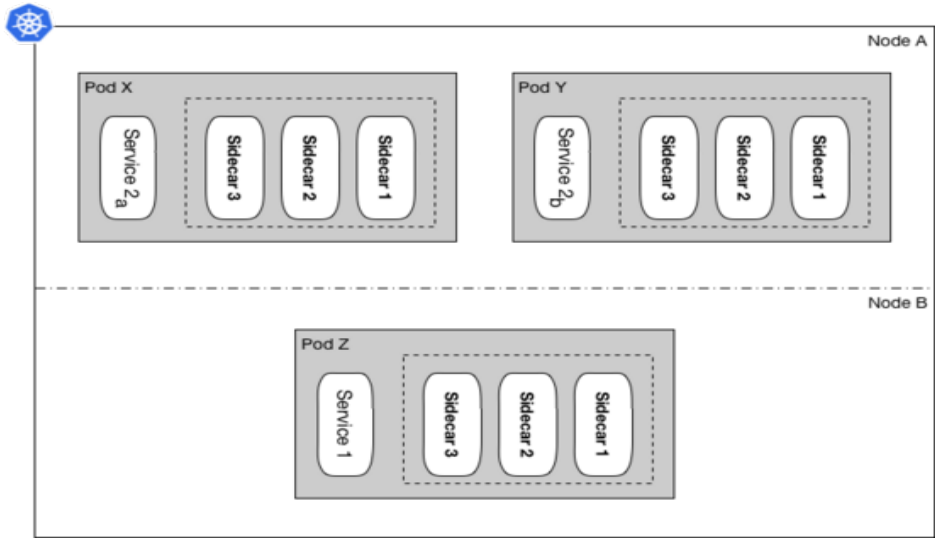
Sidecar和Daemonset是k8s的两种容器模式。

- Sidecar模式下每个Pod内部署一个Agent容器，一个Pod内可以有多个不同的Sidecar容器。
- Daemonset模式下每个Node(主机)部署一个Agent容器，一个Node上可以部署多个不同类型的Daemonset容器。

Sidecar优点：减少容器间通信延迟

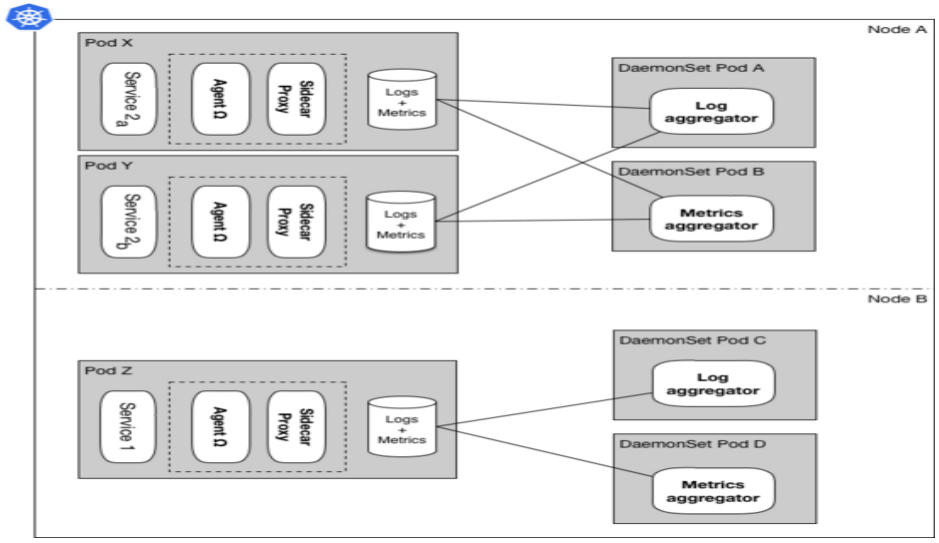
Daemonset优点：降低资源占用

Sidecar



Sidecar产品：Aqua Enforcer

Daemonset



Daemonset产品：StackRox, NeuVector, Twistlock

Sidecar or Daemonset ?

结论：这两种形态都有安全产品使用，不过Daemonset方式不会侵入Pod，且资源占用有优势。

安全切入点

1. Jenkins编译阶段：检查Dockerfile，隐私扫描，漏洞检测。以jenkins插件形式。 例: Neuvector, Tenable, Anchore Engine, Aqua

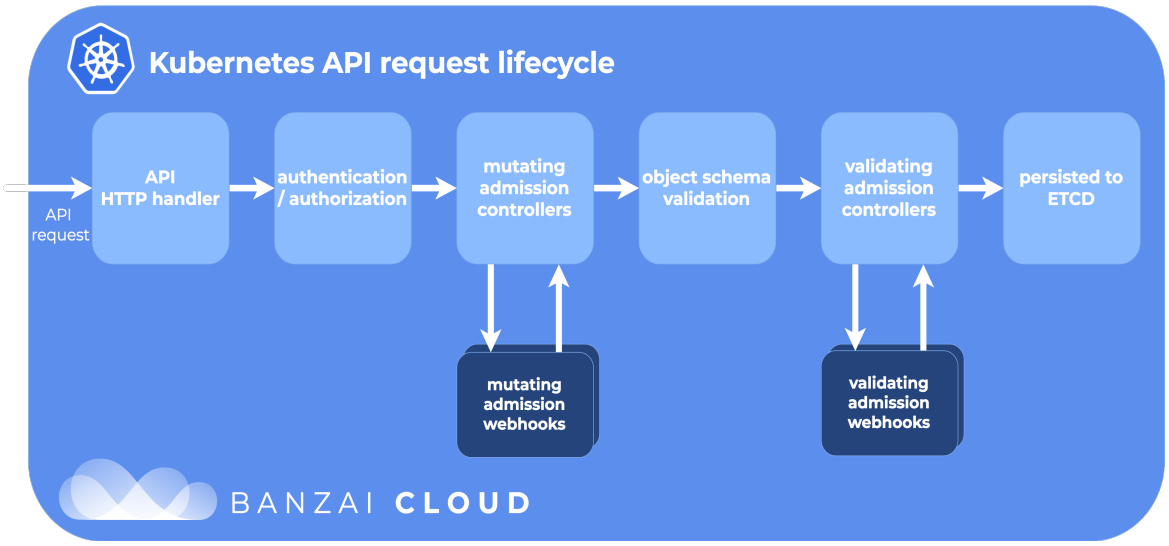
2. 仓库集成：隐私扫描，漏洞检测。需要仓库支持。 例: Clair

3. Kubernetes api server集成

Kubernetes以webhook方式提供动态准入控制(Dynamic Admission Controll)，可以对所有api请求进行修改或者校验，包括mutating admission webhook和validating admission Webhook。

- mutating admission webhook允许对api请求进行修改。
- validating admission webhook不能修改请求，只能返回validate结果为true或false。

通过准入控制，可以检查RBAC,secret，配置，检查特权容器，必须容器化部署。 例: Anchore Image Validator



4. Docker http代理

Docker daemon默认监听/var/run/docker.sock,或者部署到远程ip:port，所有docker api请求都要经过docker.sock或者ip:port，通过设置中间代理，实现拦截api调用。

5. Docker认证插件

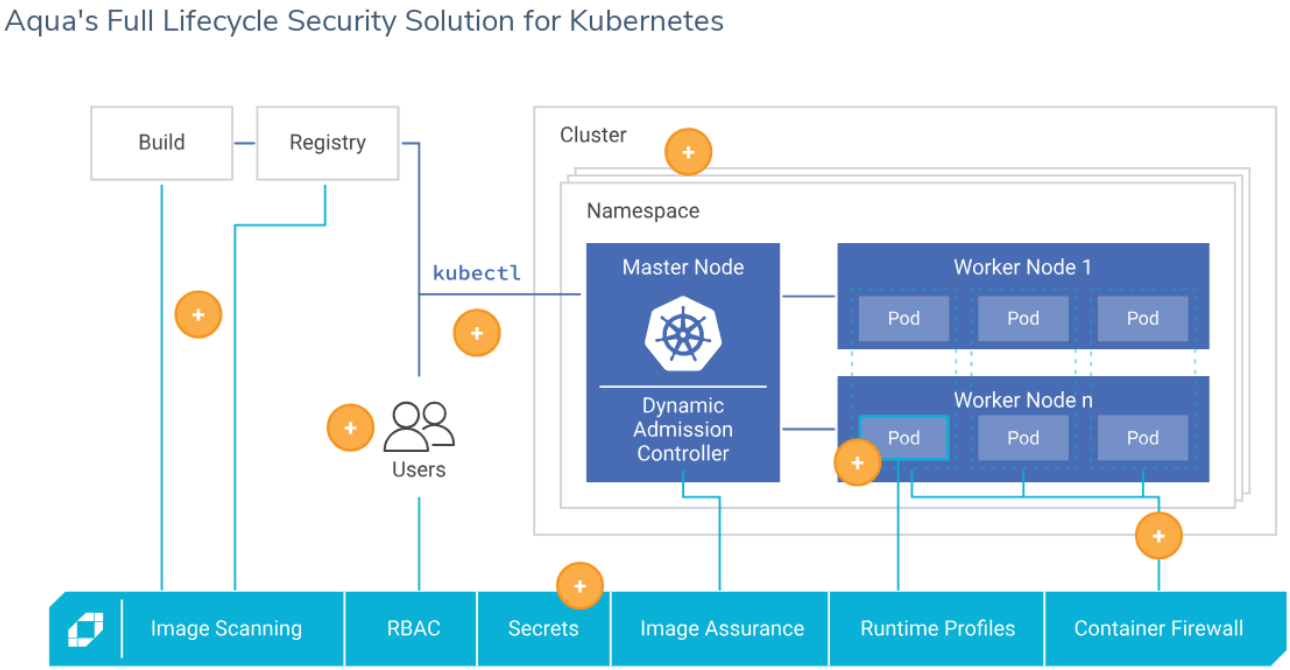
Docker插件是增强Docker引擎功能的进程外扩展，其中授权插件允许接管Docker Daemon及其远程调用接口(REST API)的认证和授权。Docker Cli的HTTP请求会被Docker Daemon转发给AuthZPlugin(即授权插件)，AuthZPlugin允许该请求执行，则把授权结果返回给Daemon，然后Daemon接下来处理后续的具体操作；若AuthZPlugin拒绝了该请求，则Daemon会直接返回Docker Client错误。 例: Twistlock

6. k8s pod隔离

k8s提供Network Policy机制，可以控制pod的网络联通性(Outbound&Inbound)。例: Aqua

Network Policy介绍: https://kubernetes.io/docs/concepts/services-networking/network-policies/#isolated-and-non-isolated-pods

安全切入点比较全面的产品：**Aqua**



来源: https://www.aquasec.com/solutions/kubernetes-container-security/

Gartner报告里的容器安全厂商

- Monitoring: Datadog, Dynatrace, Instana, Sysdig
- Security: Aqua Security, NeuVector, StackRox, Twistlock

基于主机的安全产品

Symantec Cloud Workload Protection Suite

- CI/CD集成(镜像杀毒)
- CIS, NIST, SOC2, ISO/IEC, PCI, HIPAA基线
- 容器杀毒，进程阻断，文件隔离

Symantec Data Center Security

容器化的安全产品

cAdvisor

google出的容器资源和性能监控工具 - 监控cgroup根目录(启动时要映射主机根目录)，获得容器启动(ContainerAdd)和退出(ContainerDelete)事件 - 通过docker api(docker stats)获取容器的资源(cpu,内存，磁盘，网络，io，进程)信息 - 通过映射主机端口，在cAdvisor容器内部提供dashboard

```
docker run \
  --volume=/:/rootfs:ro \
  --volume=/var/run:/var/run:ro \
  --volume=/sys:/sys:ro \
  --volume=/var/lib/docker/:/var/lib/docker:ro \
  --volume=/dev/disk/:/dev/disk:ro \
  --publish=8080:8080 \
  --detach=true \
  --name=cadvisor \
  google/cadvisor:latest
```

Sysdig

- 使用Dynamic Kernel Module Support(DKMS)加载Sysdig-probe驱动，或ebpf(kernel version>=4.14)
- 驱动层捕获系统调用

```
docker run -i -t --name sysdig --privileged
  -v /var/run/docker.sock:/host/var/run/docker.sock
  -v /dev:/host/dev
  -v /proc:/host/proc:ro
  -v /boot:/host/boot:ro
  -v /lib/modules:/host/lib/modules:ro
  -v /usr:/host/usr:ro
```

```
      sysdig/sysdig
```

输出示例： 5352241 11:54:08.853532329 0 ssh-agent (13314) < stat res=0 path=/home/cizixs/.ssh

查看所有捕获的字段: https://github.com/draios/sysdig/wiki/Sysdig-User-Guide#user-content-filtering

Falco

容器命令审计工具，容器化部署，特权容器

```
docker run -i -t --name falco --privileged \
  -v /var/run/docker.sock:/host/var/run/docker.sock \
  -v /dev:/host/dev \
  -v /proc:/host/proc:ro \
  -v /boot:/host/boot:ro \
  -v /lib/modules:/host/lib/modules:ro \
  -v /usr:/host/usr:ro \
  sysdig/falco
```

Falco事件来源 - 内核驱动sydig-probe/falco-probe，或ebpf(kernel version>=4.14) - Kubernetes审计事件(审计pod/service创建)

Falco规则示例

```
- rule: Unexpected inbound connection source
  desc: Detect any inbound connection from a source outside of an allowed set of ips, networks, or domain names
  condition: >
    consider_all_inbound_conns and inbound and not
    ((fd.cip in (allowed_inbound_source_ipaddrs)) or
     (fd.cnet in (allowed_inbound_source_networks)) or
     (fd.cip.name in (allowed_inbound_source_domains)))
  output: Disallowed inbound connection source (command=%proc.cmdline connection=%fd.name user=%user.name)
  priority: NOTICE

- rule: Run shell untrusted
  desc: an attempt to spawn a shell below a non-shell application. Specific applications are monitored.
  condition: >
    spawned_process
    and shell_procs
    and proc.pname exists
    and protected_shell_spawner
    and not proc.pname in (shell_binaries, gitlab_binaries, cron_binaries,...)

- rule: Launch Privileged Container
  desc: Detect the initial process started in a privileged container. Exceptions are made for known trusted images.
  condition: >
    container_started and container
    and container.privileged=true
    and not trusted_containers
    and not user_trusted_containers
  output: Privileged container started (user=%user.name command=%proc.cmdline %container.info image=%container.image.repository:%container.image.tag)
  priority: INFO
```

neuvector

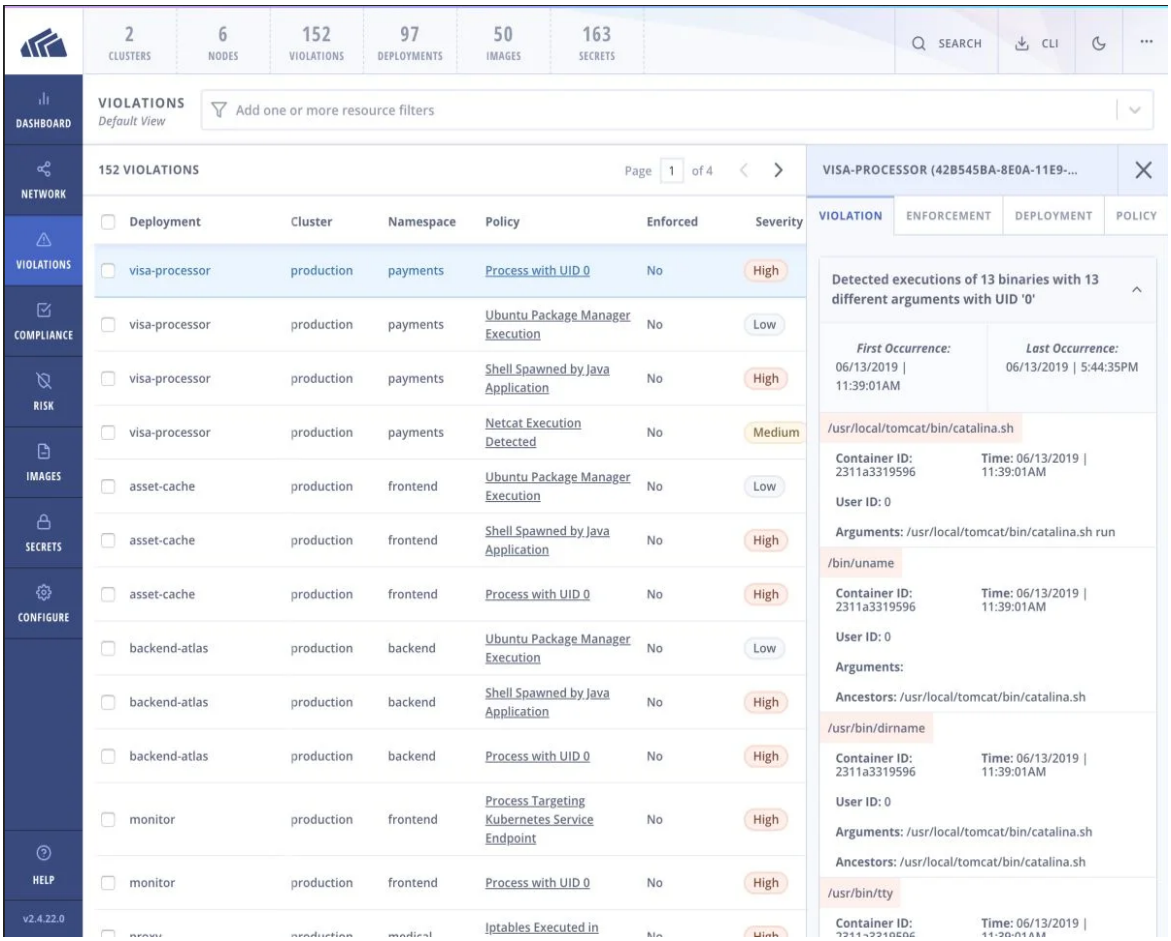
- 容器化部署
- Jenkins插件扫漏洞，扫描仓库镜像
- 7层防火墙

StackRox

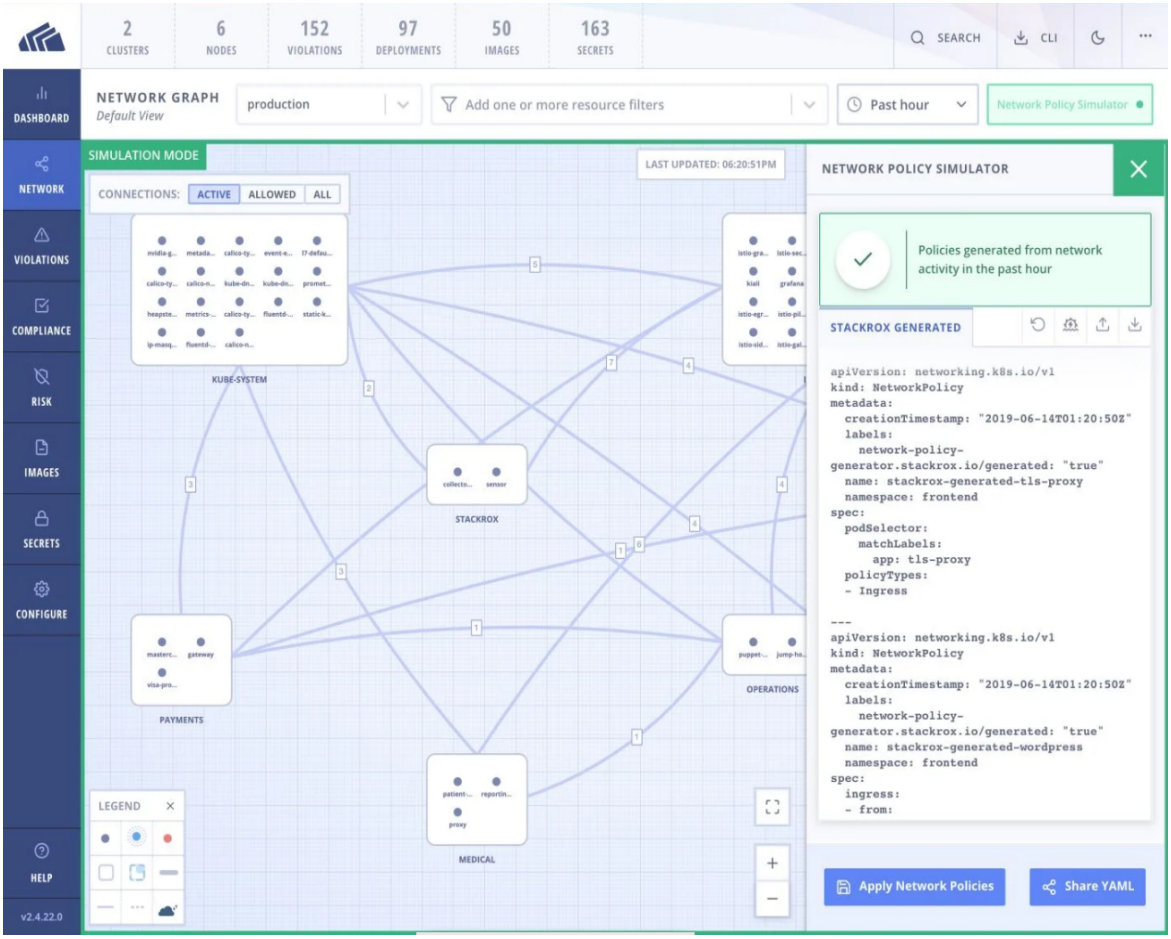
- 漏洞扫描
- 网络拓扑
- 风险发现(恶意进程，RBAC,secret，配置，检测特权容器)
- CIS, NIST, PCI, HIPAA基线
- CI/CD集成
- CI/CD阻断，镜像阻断(Dynamic Admission Controll)
- 容器隔离(Kubernetes network policy)

StackRox介绍: https://security.stackrox.com/rs/219-UEH-533/images/StackRox-Kubernetes-Security-Platform-Solution-Brief.pdf

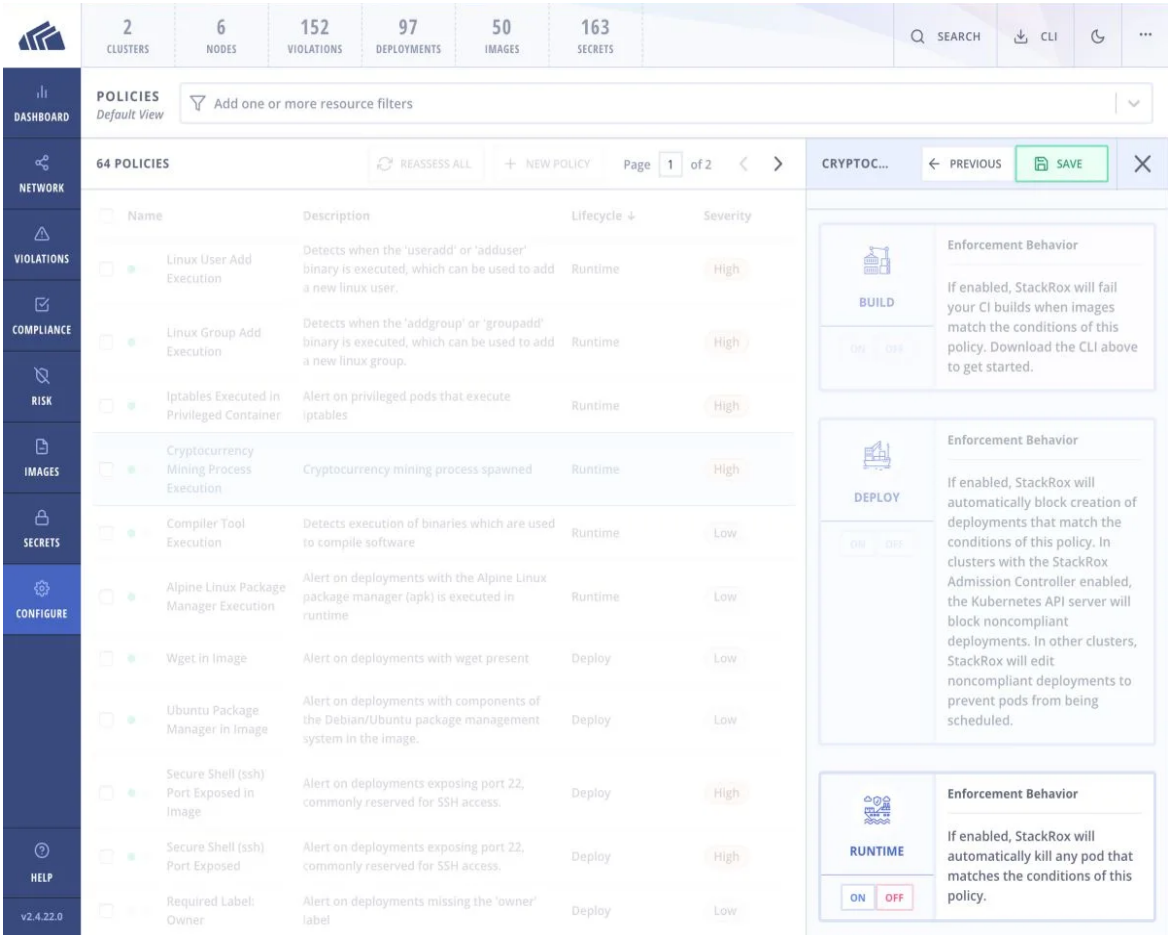
StackRox恶意进程


2 CLUSTERS
6 NODES
152 VIOLATIONS
97 DEPLOYMENTS
50 IMAGES
163 SECRETS
SEARCH
CLI
DASHBOARD
NETWORK
VIOLATIONS
COMPLIANCE
RISK
IMAGES
SECRETS
CONFIGURE
HELP
v2.4.22.0
VIOLATIONS
Default View
Add one or more resource filters
152 VIOLATIONS
Page 1 of 4
Deployment
Cluster
Namespace
Policy
Enforced
Severity
visa-processor production payments Process with UID 0 No High
visa-processor production payments Ubuntu Package Manager Execution No Low
visa-processor production payments Shell Spawned by Java Application No High
visa-processor production payments Netcat Execution Detected No Medium
asset-cache production frontend Ubuntu Package Manager Execution No Low
asset-cache production frontend Shell Spawned by Java Application No High
asset-cache production frontend Process with UID 0 No High
backend-atlas production backend Ubuntu Package Manager Execution No Low
backend-atlas production backend Shell Spawned by Java Application No High
backend-atlas production backend Process with UID 0 No High
monitor production frontend Process Targeting Kubernetes Service Endpoint No High
monitor production frontend Process with UID 0 No High
VISA-PROCESSOR (42B545BA-8E0A-11E9-...
VIOLATION
ENFORCEMENT
DEPLOYMENT
POLICY
Detected executions of 13 binaries with 13 different arguments with UID '0'
First Occurrence: 06/13/2019 | 11:39:01AM
Last Occurrence: 06/13/2019 | 5:44:35PM
/usr/local/tomcat/bin/catalina.sh
Container ID: 2311a3319596
Time: 06/13/2019 | 11:39:01AM
User ID: 0
Arguments: /usr/local/tomcat/bin/catalina.sh run
/bin/uname
Container ID: 2311a3319596
Time: 06/13/2019 | 11:39:01AM
User ID: 0
Arguments:
Ancestors: /usr/local/tomcat/bin/catalina.sh
/usr/bin/dirname
Container ID: 2311a3319596
Time: 06/13/2019 | 11:39:01AM
User ID: 0
Arguments: /usr/local/tomcat/bin/catalina.sh
Ancestors: /usr/local/tomcat/bin/catalina.sh
/usr/bin/tty
Container ID:
Time: 06/13/2019 |

StackRox网络



StackRox阻断能力



clair

- Harbor集成(sh install.sh –with-clair)
- Quay集成(Quay Security Scanner,提供Clair endpoint即可)

安装

```
$ mkdir $PWD/clair_config
$ curl -L https://raw.githubusercontent.com/coreos/clair/master/config.yaml.sample -o $PWD/clair_config/config.yaml
$ docker run -d -e POSTGRES_PASSWORD="" -p 5432:5432 postgres:9.6
$ docker run --net=host -d -p 6060-6061:6060-6061 -v $PWD/clair_config:/config quay.io/coreos/clair-git:latest -config=/config/config.yaml
```

扫描

1. Clair部署在本机

   ```
   $ analyze-local-images <Docker Image>
   ```

2. Clair部署在远程机器

   ```
   $ analyze-local-images -endpoint "http://<CLAIR-IP-ADDRESS>:6060" -my-address "<MY-IP-ADDRESS>" <Docker Image>
   ```

- 搭建一个http server，监听9279端口
- POST扫描请求到远程机器上的Clair
- Clair通过http下载层文件进行扫描

Tenable

- 支持主机、镜像、容器(基线，漏洞扫描，源码扫描)
- 支持Jenkins, Bamboo, Shippable, Travis CI

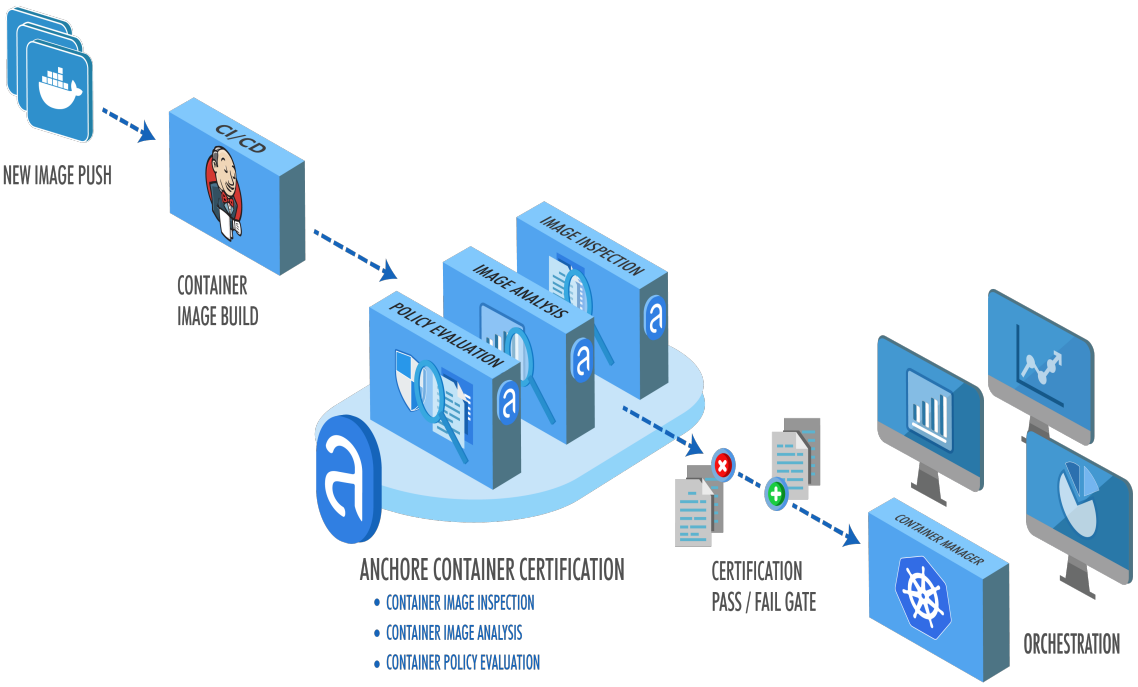
Anchore (Engine)

集成到Jenkins(Jenkins插件+Anchor Engine endpoint)进行

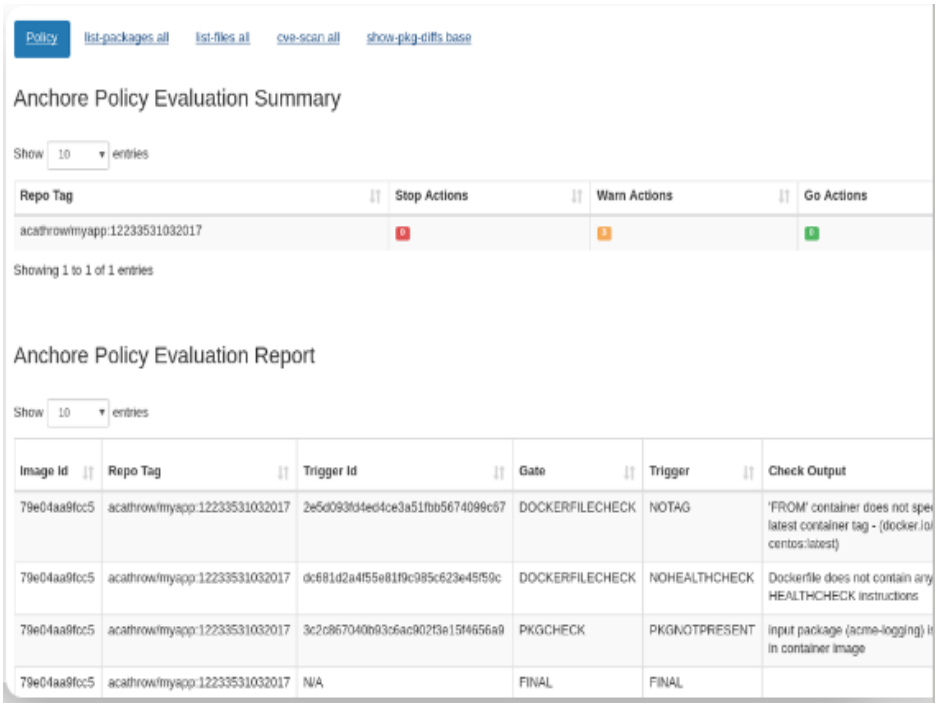
- Dockerfile扫描
- 漏洞扫描

- 隐私保护(防止泄密)

Anchor engine介绍: https://anchore.com/opensource/



Anchore (Engine)界面



Anchore (Engine)策略

来源: https://mydeveloperplanet.com/2019/02/27/anchore-container-image-scanner-jenkins-plugin/

```
$ anchore-cli policy describe
+----------------+-----------------------------------------------------------+
| Gate           | Description                                               |
+----------------+-----------------------------------------------------------+
```

```
| always          | Triggers that fire unconditionally if present in policy,   |
|                 | useful for things like testing and blacklisting.           |
+-----------------+------------------------------------------------------------+
| dockerfile      | Checks against the content of a dockerfile if provided, or |
|                 | a guessed dockerfile based on docker layer history if the  |
|                 | dockerfile is not provided.                                |
+-----------------+------------------------------------------------------------+
| files           | Checks against files in the analyzed image including file  |
|                 | content, file names, and filesystem attributes.            |
+-----------------+------------------------------------------------------------+
| licenses        | License checks against found software licenses in the      |
|                 | container image                                            |
+-----------------+------------------------------------------------------------+
| metadata        | Checks against image metadata, such as size, OS, distro,   |
|                 | architecture, etc.                                         |
+-----------------+------------------------------------------------------------+
| npms            | NPM Checks                                                 |
+-----------------+------------------------------------------------------------+
| packages        | Distro package checks                                      |
+-----------------+------------------------------------------------------------+
| passwd_file     | Content checks for /etc/passwd for things like usernames,  |
|                 | group ids, shells, or full entries.                        |
+-----------------+------------------------------------------------------------+
| ruby_gems       | Ruby Gem Checks                                            |
+-----------------+------------------------------------------------------------+
| secret_scans    | Checks for secrets found in the image using configured     |
|                 | regexes found in the "secret_search" section of            |
|                 | analyzer_config.yaml.                                      |
+-----------------+------------------------------------------------------------+
| vulnerabilities | CVE/Vulnerability checks.                                  |
+-----------------+------------------------------------------------------------+

$ anchore-cli policy describe --gate=vulnerabilities
+--------------------------------+----------------------------------------+---------------------+
| Trigger                        | Description                            | Parameters          |
+--------------------------------+----------------------------------------+---------------------+
| package                        | Triggers if a found vulnerability in an | package_type,       |
|                                | image meets the comparison criteria.   | severity_comparison, |
|                                |                                        | severity,           |
|                                |                                        | fix_available,      |
|                                |                                        | vendor_only         |
+--------------------------------+----------------------------------------+---------------------+
| stale_feed_data                | Triggers if the CVE data is older than | max_days_since_sync |
|                                | the window specified by the parameter  |                     |
|                                | MAXAGE (unit is number of days).       |                     |
+--------------------------------+----------------------------------------+---------------------+
| vulnerability_data_unavailable | Triggers if vulnerability data is      |                     |
|                                | unavailable for the image's distro.    |                     |
+--------------------------------+----------------------------------------+---------------------+
```

Twistlock

- PCI, HIPAA, GDPR, NIST基线
- 7层防火墙
- 镜像阻断

Daemonset容器权限

```
"CapAdd": [
    "NET_ADMIN",
    "SYS_ADMIN",
    "SYS_PTRACE",
    "AUDIT_CONTROL"
],
"Privileged": false,
```

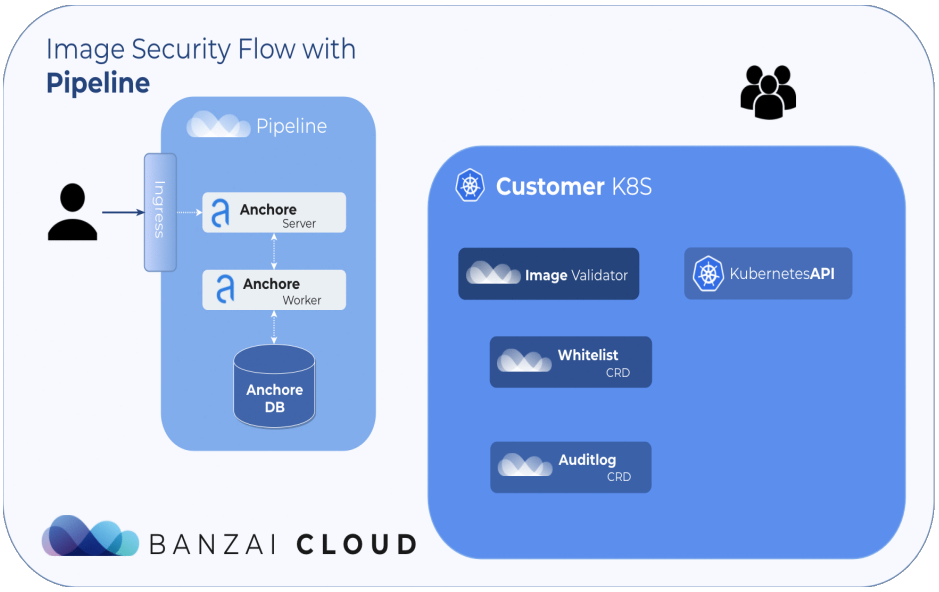
路径映射

```
• /var/lib/docker — Required for accessing Docker runtime.
• /var/run/docker.sock — Required for accessing Docker runtime.
• /usr/bin/docker — Required for capturing Docker CLI access.
• /var/lib/twistlock — Required for storing Twistlock data.
• /dev/log — Required for writing to syslog.
• [optional] /usr/lib/systemd/system/docker-registry.service
• [optional] /usr/lib/systemd/system/docker.service
• [optional] /etc/default/docker
• [optional] /etc/sysconfig/docker-network
• [optional] /etc/sysconfig/docker
• [optional] /etc/sysconfig/docker-registry
• [optional] /etc/sysconfig/docker-storage• /etc/audit/audit.rules
```

参考[容器能力及特权]，可以看到Twistlock虽然不是特权容器，但是通过添加必要的能力(Capabilities)，并且映射必要的主机目录，即可实现相关的安全能力。

Anchore (Image Validator)

利用validating admission Webhook审核Kubernetes api的镜像操作。



开源代码: https://github.com/banzaicloud/anchore-image-validator

Anchor image validator介绍: https://banzaicloud.com/blog/anchore-image-validation/

Aqua

- [MicroScanner] CI/CD集成镜像漏洞扫描(Jenkins, Bamboo, Azure DevOps, TeamCity)
- 镜像仓库镜像漏洞扫描
- 镜像阻断(Kubernetes api server集成和运行阻断)
- 主机、k8s、docker基线，漏洞扫描和病毒扫描
- 微服务级的网络拓扑和防火墙
- Sidecar容器

Aqua容器安全介绍: https://www.aquasec.com/products/aqua-cloud-native-security-platform/#diagram

kubernetes集成: https://www.aquasec.com/solutions/kubernetes-container-security/