【应用安全】Docker 安全配置(一)

0x00 Docker应用介绍

Docker 是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的容器中，然后发布到任何流行的Linux机器上，也可以实现虚拟化，容器是完全使用沙箱机制，相互之间不会有任何接口。

一个完整的Docker有以下几个部分组成：Docker Client客户端、Docker Daemon守护进程、Docker Image镜像、Docker Container容器 。

Docker 使用客户端-服务器 (C/S) 架构模式，使用远程API来管理和创建Docker容器。Docker 容器通过 Docker 镜像来创建。容器与镜像的关系类似于面向对象编程中的对象与类。

0x01 为什么要做安全配置

Docker自2013年以来非常火热，无论是从 github 上的代码活跃度，还是Redhat在RHEL6.5中集成对Docker的支持, 就连 Google 的 Compute Engine 也支持 Docker 在其之上运行。

Gartner公布2017年顶级安全技术中就包含了容器安全。容器使用的是一种共享操作系统（OS）的模型。对宿主OS的某个漏洞利用攻|击可能导致其上的所有容器失陷。即便容器本身并非不安全，但如果缺少安全团队的介入，以及安全架构师的指导，容器的部署过程可能产生不安全因素。传统的基于网络或者主机的安全解决方案对容器安全是没有作用的。容器安全解决方案必须保护容器从创建到投产的整个生命周期的安全。目前大部分容器安全解决方案都提供投产前扫描和运行时监测保护的能力。

从报告分析中可以看出，容器安全真正的问题在于Docker的部署过程中缺乏安全团队的接入，并没有使用安全的部署方式进行部署，同时报告中也指出了Docker安全需要对Docker从创建到投产的整个生命周期进行安全安全防护。在Docker的部署中对Docker进行安全配置能够有效的减少安全威胁。对Docker进行安全配置可以简单的分为一下几个方面：Docker容器安全、Docker镜像安全、Docker主机安全。接下来我们会对这三个方面进行详细的分析。

0x02 如何进行Docker安全配置

2.1主机安全配置

2.1.1为容器创建一个单独的分区

所有Docker容器及数据和元数据都存储在/var/lib/docker目录下。默认情况下，/var/lib/docker将根据可用性挂载在/或/var 分区下。Docker依赖于/var/lib/docker作为默认目录，其存储所有Docker相关文件，包括镜像文件。该目录可能会被恶意的写满，导致 Docker、甚至主机可能无法使用。因此，建议为存储 Docker 文件创建一个单独的分区（逻辑卷）。

检查方法：执行命令，应该返回/var/lib/docker挂载点的分区详细信息。

```
grep /var/lib/docker /etc/fstab
```

加固方法：新安装 Docker 时，为/var/lib/docker挂载点创建一个单独的分区。对于先前安

装的系统，请使用逻辑卷管理器（LVM）创建分区。

2.1.2加固容器宿主机

容器在 Linux 主机上运行。容器主机可以运行一个或多个容器。加强主机以缓解主机安全配置错误是非常重要的。应该遵循基础架构安全最佳实践并加强宿主机操作系统。保证主机系统的安全可以确保主机漏洞得控制。若不加固主机系统可能导致安全问题。

2.1.3更新Docker到最新版本

Docker软件频繁发布更新，旧版本可能存在安全漏洞。通过及时了解Docker 更新，Docker软件中的漏洞可以得到修复。攻击者可能会尝试获得访问权限或提升权限时利用已知的漏洞。不安装常规的Docker更新可能会让现有的Docker软件受到攻击。可能会导致提升权限，未经授权的访问或其他安全漏洞。所以需要跟踪新版本并根据需要进行更新。

检查方法：执行以下命令并验证Docker版本是否为必要的最新版本。

```
Docker Version
```

加固方法：跟踪 Docker 发布并根据需要进行更新。

2.1.4审计Docker守护进程、文件和目录

审计所有活动的Docker守护进程，除了审核常规的Linux文件系统和系统调用外，还要审核所有Docker相关的文件和目录。 Docker守护进程以'root'特权运行。它的运行取决于一些关键文件和目录。

检查方法：

验证是否存在Docker守护程序和文件目录的审核规则。

```
1）auditctl-l | grep /usr/bin/docker
2）auditctl-l | grep /var/lib/docker
3）auditctl-l | grep /etc/docker
4）"systemctlshow -p FragmentPath docker.service
  auditctl -l | grep docker.service"
5）"systemctlshow -p FragmentPath docker.socket
```

```
  auditctl -l | grep docker.socket"
6）auditctl-l | grep /etc/default/docker
7）auditctl-l | grep /etc/docker/daemon.json
8）auditctl-l | grep /usr/bin/docker-containerd
9）auditctl-l | grep /usr/bin/docker-runc
```

加固方法：

在/etc/audit/audit.rules 文件中添加以下配置。

```
-w /usr/bin/docker -k docker
-w /var/lib/docker -k docker
-w /etc/ docker -k docker
-w /usr/lib/systemd/system/docker.service -k docker
-w /usr/lib/systemd/system/docker.socket -k docker
-w /etc/default/docker -k docker
-w /etc/docker/daemon.json -k docker
-w /usr/bin/docker-containerd -k docker
-w /usr/bin/docker-runc -k docker
```

2.2Docker 守护进程配置

2.2.1 限制默认网桥上容器之间的网络流量

默认情况下，默认网桥上同一主机上的所有容器之间启用不受限制的网络通信。因此，每个容器都有可能读取同一主机上容器网络上的所有数据包。这可能会导致意外和不必要的信息泄露给其他容器。因此，限制默认网桥上的容器间通信。

检查方法：运行以下命令并确认默认网桥已被配置为限制集装箱间通信。它应该返回默认网桥的 com.docker.network.bridge.enable_icc:false。

```
docker network ls --quiet | xargs docker network inspect --format '{{.Name}}:{
```

加固方法：在守护进程模式下运行 Docker 并传递--icc = false作为参数。

例如，dockerd --icc = false或者可以遵循 Docker 文档并创建自定义网络，并只加入需要与该自定义网络通信的容器。 --icc参数仅适用于默认泊坞桥，如果使用自定义网络，则应采用分段网络的方法。

2.2.2设置日志级别为info

将Docker守护程序日志级别设置为"info"。设置适当的日志级别，配置 Docker 守护程序以记录需要查看的事件。"info"及以上的基准日志级别将捕获除调试日志之外的所有日志。若无必须，不应该在'debug'日志级别运行Docker 守护进程。

检查方法：执行以下命令确保--log-level 参数不存在或存在，然后将其设置为 info。

```
ps -ef | grep dockerd
```

加固方法：运行Docker守护进程如下：

```
dockerd --log-level ="info"
```

2.2.3 允许Docker 更改Iptables

Docker会根据用户为容器选择网络选项的方式自动对iptables进行必要的更改（如果允许的话）。建议让Docker 服务器自动更改iptables，以避免可能妨碍容器与外界通信的网络配置错误。另外，当每次选择运行容器或修改网络选项时，可以自动更新iptables的配置。

检查方法：执行以下命令确保'--iptables'参数不存在或不设置为'false'。

```
ps -ef | grep dockerd
```

加固方法：

不要使用'--iptables = false'参数运行Docker守护程序。例如，不要像下面那样启动Docker守护进程：

```
dockerd --iptables = false
```

2.2.4 Docker 守护进程配置TLS 身份认证

默认情况下，Docker守护程序绑定到非联网的Unix套接字，并以"root"权限运行。如果将默认的Docker守护程序更改为绑定到TCP端口或任何其他 Unix 套接字，那么任何有权访问该端口或套接字的人都可以完全访问Docker守护程序，进而可以访问主机系统。因此，不应该将Docker 守护程序绑定到另一个 IP/端口或Unix套接字。如果必须通过网络套接字暴露Docker 守护程序，请为守护程序和Docker Swarm API配置TLS身份验证。

检查方法：执行以下命令确保存在以下参数：'--tlsverify' · '--tlscacert' · '--tlscert' · '--tlskey'

```
ps -ef | grep dockerd
```

加固方法：在启动参数中配置'--tlsverify' · '--tlscacert' · '--tlscert' · '--tlskey'参数。

2.2.5 配置合适的ulimit

ulimit 提供对shell 可用资源的控制。设置系统资源控制可以防止资源耗尽带来的问题，如fork炸弹。有时候合法的用户和进程也可能过度使用系统资源，导致系统资源耗尽。

为Docker守护程序设置默认ulimit将强制执行所有容器的ulimit。不需要单独为每个容器设置ulimit。但默认的ulimit可能在容器运行时被覆盖。因此，要控制系统资源，需要自定义默认的ulimit。

检查方法：执行以下命令确保根据需要设置'--default-ulimit'参数

```
ps -ef | grep dockerd
```

加固方法：在守护进程模式下运行docker，并根据相应的ulimits传递'--default-ulimit'作为参数。例如：

```
dockerd --default-ulimit nproc = 1024:2408 --default-ulimit nofile =100:200
```

2.2.6 设置容器的默认空间大小

守护进程重启时可以增加容器空间的大小。用户可以通过设置默认容器空间值来进行扩大，但不允许缩小。设立该值的时候需要谨慎，防止设置不当带来空间耗尽的情况。

检查方法：执行以下命令执行上述命令，它不应显示任何--storage-opt dm.basesize 参数。

```
ps -ef | grep dockerd
```

加固方法：如无需要，不要设置--storage-opt dm.basesize

2.2.7 启用docker 客户端命令的授权

使用本机 Docker 授权插件或第三方授权机制与Docker守护程序来管理对 Docker客户端命

令的访问。

检查方法：执行以下命令检查如果使用Docker本地授权，可使用--authorization-plugin参数加载授权插件。

```
ps -ef | grep dockerd
```

加固方法：

第 1 步：安装/创建授权插件。

第2 步：根据需要配置授权策略。

第 3 步：重启docker守护进程，如下所示：

```
dockerd --authorization-plugin = <PLUGIN_ID>
```

2.2.8 启用实时恢复

可用性作为安全一个重要的属性。在Docker守护进程中设置'--live-restore'标志可确保当Docker守护进程不可用时容器执行不会中断。这也意味着当更新和修复Docker守护进程而不会导致容器停止工作。

检查方法：执行以下命令并确保使用--live-restore。

```
ps -ef | grep dockerd
```

加固方法：在守护进程模式下运行docker并传递'--live-restore'作为参数。

例如，dockerd --live-restore

2.2.9禁用userland 代理

Docker引擎提供了两种机制将主机端口转发到容器,DNAT和userland-proxy。在大多数情况下，DNAT模式是首选，因为它提高了性能，并使用本地 Linux iptables功能而需要附加组件。

如果DNAT可用，则应在启动时禁用 userland-proxy 以减少安全风险。

检查方法：执行以下命令确保--userland-proxy 参数设置为 false。

```
ps -ef | grep dockerd
```

加固方法：运行docker守护进程如下：dockerd --userland-proxy = false

2.2.10 应用守护进程范围的自定义 seccomp 配置文件

大量系统调用暴露于每个用户级进程，其中许多系统调用在整个生命周期中都未被使用。大多数应用程序不需要所有的系统调用，因此可以通过减少可用的系统调用来增加安全性。

可自定义seccomp配置文件，而不是使用Docker的默认seccomp配置文件。如果Docker的默认配置文件够用的话，则可以选择忽略此建议。

检查方法：运行以下命令并查看"SecurityOptions"部分中列出的seccomp配置文件。如果它是默认，那就意味着，应用了Docker的默认seccomp配置文件。

```
docker info --format='{{.SecurityOptions}}'
```

加固方法：默认情况下，Docker使用默认seccomp配置文件。如果这对当前环境有益，则不需要采取任何行动。当然，也可以选择应用自己的seccomp配置文件，需在守护进程启动时使用--seccomp-profile标志，或将其放入守护程序运行时参数文件中。

```
dockerd --seccomp-profile </path/to/seccomp/profile>
```

2.2.11生产环境中避免实验性功能

Docker实验功能现在是一个运行时Docker守护进程标志, 其作为运行时标志传递给Docker守护进程，激活实验性功能。实验性功能现在虽然比较稳定，但是一些功能可能没有大规模经使用，并不能保证 API 的稳定性，所以不建议在生产环境中使用。

检查方法：运行下面的命令，并确保在Server部分将Experimental属性设置为 false。

```
docker version --format='{{.Server.Experimental}}'
```

加固方法：运行下面的命令，并确保在Server部分将Experimental属性设置为 false。不要将--experimental 作为运行时参数传递给 docker 守护进程。

2.3docker 守护程序文件配置

docker.service文件包含可能会改变Docker守护进程行为的敏感参数；docker.socket 文件包含可能会改变Docker远程 API 行为的敏感参数；/etc/docker目录除了各种敏感文件外还

包含证书和密钥因此；/etc/docker/certs.d/<registry-name>目录包含Docker 镜像仓库证书；TLSCA 证书文件、Docker服务器证书文件、Docker服务器证书密钥文件应受到保护，不受任何篡改，它用于指定的 CA 证书验证。这些文件及目录应该由"root"拥有和归属，以保持文件的完整性，除"root"之外的任何其他用户不应该保留文件的完整性。

检查方法：

1）docker.service 文件的所有权为 root:root

2）docker.service 文件权限为 644 或更多限制性

3）docker.socket 文件所有权为 root:root

4）docker.socket 文件权限为 644 或更多限制性

5）/etc/docker 目录所有权为 root:root

6）/etc/docker 目录权限为 755 或更多限制性

7）仓库证书文件所有权为 root：root

8）仓库证书文件权限为444 或更多限制性

9）TLS CA 证书文件所有权为 root：root

10）TLS CA 证书文件权限为444 或更多限制性

11）docker 服务器证书文件所有权为 root：root

12）docker 服务器证书文件权限为 444 或更多限制

13）docker 服务器证书密钥文件所有权为 root：root

14）docker 服务器证书密钥文件权限为 400

15）docker.sock 文件所有权为root：docker

16）docker.sock 文件权限为660 或更多限制性

17）设置/etc/default/docker 文件所有权为 root：root

18）设置/etc/default/docker 文件权限为 644 或更多限制性

加固方法：修改以下文件的权限

```
chown root:root /usr/lib/systemd/system/docker.service
chmod 644 /usr/lib/systemd/system/docker.service
chown root: root /usr/lib/systemd/system/docker.socket
chmod 644 /usr/lib/systemd/system/docker.socket
chown root:root /etc/docker
```

```
chmod 755 /etc/docker
chown root:root /etc/docker/certs.d/<registry-name>/*
chmod 444 /etc/docker/certs.d/<registry-name>/*
chown root:root <路径到 TLS CA 证书文件>
chmod 444 <路径到 TLS CA 证书文件>
chown root:root <路径到 Docker 服务器证书文件>
chmod 444 <路径到 Docker 服务器证书文件>
chown root:root <路径到 Docker 服务器证书密钥文件>
chmod 400 <路径到 Docker 服务器证书密钥文件>
chown root:docker /var/run/docker.sock
chmod 660 /var/run/docker.sock
```

我们在本文中对Docker的主机安全配置进行了详细的分析，在Docker 安全配置(二)中我们将继续对Docker的容器和镜像安全进行分析。

参考链接

- Gartner Identifies the Top Technologies for Security in 2017
- CIS_Docker_1.6_Benchmark_v1.0.0
- Docker 容器最佳安全实践白皮书（V1.0）