通过 LSM 赋予容...

> 本篇是第六部分“安全篇”的第三篇，在这个部分，我将用四篇内容为你介绍包括镜像，容器和 Linux 内核的 LSM 等内容。前面两篇，我为你介绍了镜像及容器安全相关的内容。本篇，我们将重点放在 Linux 内核为容器提供的安全保障上。

上一篇，我为你介绍了如何通过 Linux capabilities 来为容器提供安全的能力。本篇，我们继续将焦点放在容器安全上。

注意：本文所用的 Linux 内核为 5.4.10-100.fc30.x86_64，不同版本内核略有差异。

LSM 基础

LSM 即 Linux Security Modules，翻译为 Linux 安全模块。可能很多人都没有接触过它，我先来介绍下 LSM 出现的背景及它具体是什么。

LSM 出现的背景

上一篇我已经为你介绍过 Linux 的权限模型，最初它只支持检查是否为特权用户，之后的演进中逐步增加了 capabilities 相关的功能。

但仅仅是这样还不足以称其为一个“安全”操作系统。各种组织和机构在使用 Linux 时，对它提出了多种安全方面相关的需求。

此时也有不少人实现了一些安全方面的功能。大多数实现都是通过给内核打 patch 实现的，并没有合并进 Linux 内核的主线中。

这对于普通用户而言是无法接受的，大多数用户是没有编译和定制内核的能力的，而且这些补丁也不够通用。

所以在 2001 年时，Linux 中逐步引入了一个全新的通用安全访问控制框架，即现在的 LSM。

LSM 是什么

LSM 整体而言是一个框架，提供了多种通用的安全访问控制能力，主要是各种 Hook 对权限做访问控制，但具体功能则依赖于特定的实现。自 Linux 2.6 正式成为内核一部分。

主流的一些兼容实现如下：

- AppArmor
- SELinux
- Smack（Simplified Mandatory Access Control Kernel）
- TOMOYO Linux

大家比较常见的可能是 SELinux 和 AppArmor 这两个。

如何使用

通常情况下，内核所使用的 LSM 可以在构建时就通过 CONFIG_DEFAULT_SECURITY 进行设置，也可以在启动时，通过设置内核命令行参数来覆盖它。比如，查看当前内核配置的默认使用的 LSM 模块：

```
(MoeLove) ➜  ~ grep CONFIG_DEFAULT_SECURITY /boot/config-$(uname -r)
CONFIG_DEFAULT_SECURITY_SELINUX=y
# CONFIG_DEFAULT_SECURITY_DAC is not set
CONFIG_DEFAULT_SECURITY="selinux"
```

复制

注意：不同的系统配置输出结果可能会不相同。

我这里输出的是默认使用 SELinux。

当然你也可以通过查看 /sys/kernel/security/lsm 进行检查：

```
(MoeLove) ➜  ~ cat /sys/kernel/security/lsm
lockdown,capability,yama,selinux
```

复制

这是一组以逗号分割的列表，这个列表反映了权限检查的先后顺序。排在最后的是主要模块，我这里是 selinux。

对于大多数系统，比如 RHEL/CentOS/Fedora 之类的默认都是 SELinux，Debian 默认发行的内核中也带有 SELinux 的支持，只不过默认是关闭的。

那么接下来我们就来看看 SELinux 是如何为 Docker 容器提供安全保障的吧。

为 Docker 开启 SELinux 支持

除了检查 LSM 的配置外，为 Docker 开启 SELinux 之前，需要先确认下 SELinux 是否已经开启。

```
(MoeLove) ➜  ~ getenforce
Enforcing
```

复制

这个命令可以有三种结果：

- enforcing：强制执行 SELinux；
- permissive：只是打印警告而非强制执行；
- disabled：关闭 SELinux 规则。

此外 Docker Daemon 提供了一个配置项 `--selinux-enabled` 用来启动 SELinux 的支持，默认是 false，我们将它设置为 true。

复制

```
(MoeLove) ➜  ~ dockerd --help |grep selinux
      --selinux-enabled                        Enable selinux support
```

可以将这个参数放在 Docker Daemon 的启动参数中，或者是添加到 /etc/docker/daemon.json 文件中：

复制

```
{
  "selinux-enabled": true
}
```

增加配置后重启 Docker Daemon 即可生效，通过以下方式进行验证：

复制

```
(MoeLove) ➜  ~ docker info |grep selinux
  selinux
```

只有开启 SELinux 支持后，`docker info` 的输出中才会有 SELinux 字段。

验证 SELinux 对 Docker 容器的保护

我以挂载一个目录为例，以下是我本地的一个普通目录：

复制

```
(MoeLove) ➜  ~ ls -al helm
总用量 29444
drwxr-xr-x.   2 tao tao     4096  5月 15  2018 .
drwx------. 126 tao tao    90112  2月  5 22:47 ..
-rwxr-xr-x.   1 tao tao 30033696  5月 15  2018 helm
-rw-r--r--.   1 tao tao    11373  5月 15  2018 LICENSE
-rw-r--r--.   1 tao tao     3310  5月 15  2018 README.md
```

启动一个 Debian 的容器，并将此目录挂载到容器的 /helm 目录下：

复制

```
(MoeLove) ➜  ~ docker run --rm -it -v $PWD/helm:/helm debian bash
root@6a935d60f701:/# ls -al /helm
ls: cannot open directory '/helm': Permission denied
```

你会发现，这个目录竟然没有访问权限！

这里我就直接说出原因吧，**这个和 SElinux 有关**。所以我们可以通过 `ls -Z` 查看文件的 security context：

```
# 省略了部分输出
root@6a935d60f701:/# ls -alZ /
drwxr-xr-x.   1 root root system_u:object_r:container_file_t:s0:c127,c532 4096 F
drwxr-xr-x.   1 root root system_u:object_r:container_file_t:s0:c127,c532 4096 F
drwxr-xr-x.   2 1000 1000 system_u:object_r:container_file_t:s0:c7,c505   4096 M
drwxr-xr-x.   2 root root system_u:object_r:container_file_t:s0:c127,c532 4096 M
```

可以看到被挂载进来的这个 /helm 目录和其他目录的 security context 是不一样的。

但是如果我们修改下容器的启动命令，再试试看：

```
(MoeLove) ➜  ~ sudo docker run --rm -it -v $PWD/helm:/helm:Z debian bash
root@d8fcabf52d11:/# ls /helm/
LICENSE  README.md  helm
```

可以看到可以正常访问了，此时再检查下其 security context：

```
# 省略了部分输出
root@d8fcabf52d11:/# ls -alZ /
drwxr-xr-x.   1 root root system_u:object_r:container_file_t:s0:c30,c802 4096 Fe
drwxr-xr-x.   1 root root system_u:object_r:container_file_t:s0:c30,c802 4096 Fe
drwxr-xr-x.   2 1000 1000 system_u:object_r:container_file_t:s0:c30,c802 4096 Ma
drwxr-xr-x.   2 root root system_u:object_r:container_file_t:s0:c30,c802 4096 Ma
```

可以看到，通过给 `-v` 选项的参数中增加了一个 Z 标志，现在它的 label 与其他目录均已经相同了。

事实上，此处可用的选项包括 z 和 Z，使用它们均可令挂载目录在容器内访问，但挂载进去的目录 label 却不完全相同，此处希望你可以实际挂载看看它们的区别，同时也可对比下根目录下其他目录的 label。

总结

本篇，我为你介绍了 LSM（Linux Security Modules）产生的背景及其作用和配置。其中比较典型的便是 SELinux。

之后重点介绍了 Docker 对 SELinux 的支持，包括如何配置和启用它，以及挂载目录时候可能遇到的问题及解决办法。

Docker 开启 SELinux 支持后，可避免恶意程序攻破容器后，访问到主机上的其他目录，或者访问其他容器的目录，以此来提供更高级别的安全保障。