

13 | Linux系统安全：多人共用服务器，如何防止别人干“坏事”？

2020-01-08 何为舟

安全攻防技能30讲

[进入课程 >](#)



讲述：何为舟

时长 13:13 大小 10.61M



你好，我是何为舟。

从这一讲开始，我们讨论 Linux 系统和应用安全。我们知道，在开发一个应用的过程中，需要涉及代码、操作系统、网络和数据库等多个方面。所以，只是了解代码安全肯定是不够的，我们还需要了解常见的基础环境和工具中的安全机制，学会通过正确地配置这些安全机制，来提升安全保障。

谈到 Linux，我相信你每天都在使用 Linux 进行各种开发和运维操作。但是，大多数情况下，公司不会给每一个员工分配专有的 Linux 服务器，而是多个开发和运维共用一台 Linux 服务器。那么，其他员工在使用 Linux 服务器的时候，会不会对我们自己的数据和进程产

生影响呢？另外，我在 Web 安全中讲过，黑客可以通过很多漏洞控制 Linux 服务器，那我们又该如何避免和控制黑客的破坏呢？

如何理解 Linux 中的安全模型？

要解决这些安全问题，我们首先要了解一个安全模型，Linux 的安全模型。

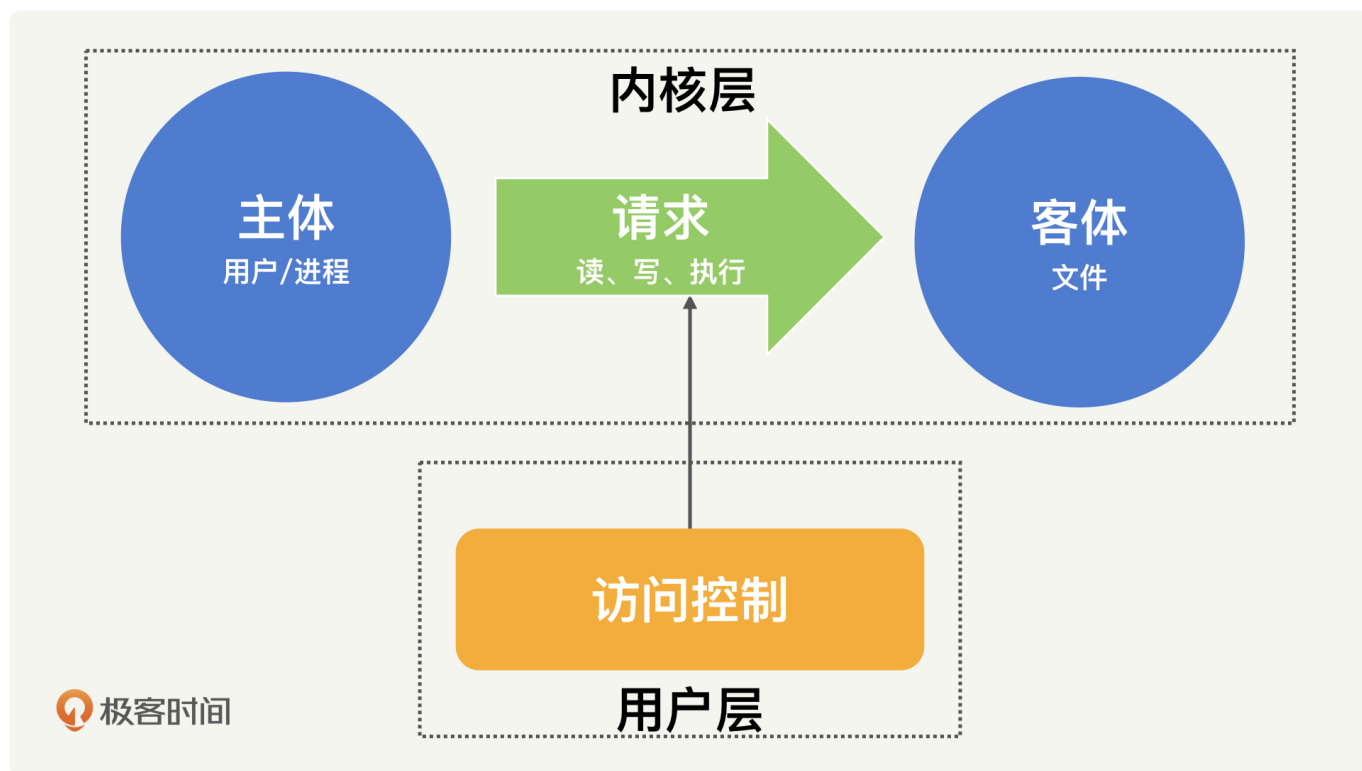
我们先来看一下 Linux 的构成，Linux 可以分为内核层和用户层。用户层通过内核层提供的操作接口，来执行各类任务。

内核层提供的权限划分、进程隔离和内存保护的安全功能，是用户层的安全基础。一旦内核安全被突破（比如黑客能够修改内核逻辑），黑客就可以任意地变更权限、操作进程和获取内存了。这个时候，任何用户层的安全措施都是没有意义的。

既然 Linux 的内核安全这么重要，那我们是不是要在防护上付出大量的精力呢？事实上，正如我们不需要在开发应用时（尤其是使用 Java 这类相对高层的语言时），过多地关心操作系统相关的内容一样，我们在考虑 Linux 安全时，也不需要过多地考虑内核的安全，更多的是要考虑用户层的安全。所以，对于 Linux 内核层的安全，我们只需要按照插件漏洞的防护方法，确保使用官方的镜像并保持更新就足够了。

既然，使用最多的是用户层，那我们就来看一下，用户层的操作都有什么。

在 Linux 中，用户层的所有操作，都可以抽象为“主体 -> 请求 -> 客体”这么一个流程。比如，“打开 /etc/passwd”这一操作的主体是实际的用户，请求是读，客体是 /etc/passwd 这个文件。



在这个过程中，Linux 内核安全提供了基于权限的访问控制，确保数据不被其他操作获取。Linux 用户层则需要确保权限的正确配置，这就是我开篇提到的，如何保证多人安全地共用服务器的关键，也是我们这节课需要关注的重点内容。

黄金法则是如何在 Linux 系统中应用的？

现在我们知道了，**Linux 系统安全防护的核心是正确配置用户层权限**。那接下来，我们就从 [黄金法则](#) 的认证、授权和审计这三个方面来看一下，Linux 系统是如何进行权限配置的，这其中，又有哪些值得我们重点关注的安全选项。


1. Linux 中的认证机制

Linux 系统是一个支持多用户的操作系统，它通过普通的文本文件来保存和管理用户信息。这其中，有 2 个比较关键的文件：**`/etc/passwd`**和**`/etc/shadow`**。

我们知道，在 Linux 中，`/etc/passwd`是全局可读的，不具备保密性。因此，`/etc/passwd`不会直接存储密码，而是用 x 来进行占位。那实际的用户密码信息，就会存储到仅 ROOT 可读的**`/etc/shadow`**中。

在**`/etc/shadow`**中，除了加密后的密码，也保存了诸如密码有效天数、失效多少天告警之类的密码管理策略。我们可以通过 Chage 命令来对密码管理策略进行修改，比如，通过下

面的 Chage 命令，就可以强制 Test 用户在 60 天内必须对密码进行修改。通过这样的方式，就可以降低密码泄漏的可能性了。

 复制代码

```
1 chage -M 60 test
```

因为认证这个功能是由 Linux 内核来提供的，所以在用户层，我们需要关心的安全问题，就是弱密码导致的身份信息泄漏。为了解决这个问题，在 `/etc/shadow` 中，我们可以制定适当的密码策略。除此之外，我们也可以通过 [John the Ripper](#)，使用已知的弱密码库，来对 Linux 中的弱密码进行检测。下面的命令，就是使用 John the Ripper 检测弱密码。

 复制代码

```
1 unshadow /etc/passwd /etc/shadow > mypasswd
2 john mypasswd
3 john --show mypasswd
```

2.Linux 中的授权机制

在“黄金法则”中，认证只是第一步，它提供了一个可信的身份标识。有了这个身份标识之后，就需要通过授权，来限制用户能够发起的请求了。

在 Linux 中，客体只有文件和目录两种，针对这两种类型的客体，Linux 都定义了读、写和执行这三种权限。你可以通过我总结的这张对比表格看到，文件和目录在这三种权限上的区别。

权限	文件	目录
读	可以读取文件内容	可以读取目录列表
写	可以修改文件内容	可以修改目录列表 (新建、删除或者重命名文件)
执行	可以执行文件	可以进入目录 (可以cd到这个目录中)



除此之外，Linux 还提供了一些额外的权限标签，来进行更细粒度地权限控制。

比如，Linux 提供了文件属性的概念，来对文件设置更多的保护。通过 `chattr +i /etc/passwd` 可以防止文件被任何用户修改。想要了解更多的文件属性，你可以参考 [Wikipedia](#)。

Linux 还提供了“粘滞位”的功能，主要用来防止用户随意操作其他用户的文件。比如 `chmod +t /tmp` 可以阻止删除 `/tmp` 目录下其他用户的文件。

这些都是 Linux 在授权中的自我保护机制，那我们能在这个过程中进行怎样的防护呢？

前面，我们一直在强调，**Linux 系统面临的安全威胁其实就是权限问题**。也就是说，要么就是敏感文件的权限配置不当，导致这些文件可以被额外的用户访问或执行；要么就是应用存在漏洞或密码泄漏，导致低权限用户可以获得更高的权限。

要解决权限问题，我们就要实践最小权限原则。

我们先来看一个 Linux 系统安全中最普遍的问题：滥用 ROOT。很多人在登录 Linux 系统后，第一个命令就是通过 `su` 来获取 ROOT 的 Shell 环境，这样我们就不需要在每次操作的时候，通过 `sudo` 来临时提升至 ROOT 权限。

但是，这里你需要注意一点，在 ROOT 的 Shell 环境中，启动的所有进程也都具备 ROOT 权限。如果启动的是一个立即返回的进程，如 CAT，不会有太多问题，但如果是一个长期运行的进程，就很容易产生权限的滥用。

比如，当你以 ROOT 的身份启动 Redis 或者 MySQL 等存储工具时，如果这时有其他用户连入 Redis 或者 MySQL，那他们也能间接地获取 ROOT 的权限。在大部分服务器入侵的场景中，黑客都是通过这些具备 ROOT 权限的进程漏洞，来实现权限提升的。

因此，在运行任何长驻进程时，我们都需要谨记“最小权限”原则。也就是说，我们可以根据要执行的操作等级，配置“最小权限”来启动常驻进程。比如，如果只是在 Redis 和 MySQL 这样的数据库中进行文件读写操作，根本不需要 ROOT 这种最高等级的权限。

因此，“最小权限”原则在 Linux 系统中的应用是非常重要的。那你可能会问了，Linux 系统中的操作那么多，每个操作都需要自己进行权限配置吗？当然不是，我们常常会使用一些已知的工具，来实现“最小权限”启动长驻进程的功能，而你需要做的，就是正确地启动或者配置这些工具。

比如说，我们可以通过 mysqld 启动 MySQL 服务，mysqld 会将 MySQL 的进程分配到“mysql”这个用户，并在 ROOT 下建立守护进程。具体的效果如下：

复制代码

```
1 root      297353  0.0  0.0 115432  1360 ?        S    Aug12   0:00 /bin/sh /usr,
2 mysql     297553 31.3  4.3 11282756 5729572 ?      Sl   Aug12 22593:40 /usr/local,
```

类似的，当启动 Nginx 时，Nginx 会将 Worker 节点以 nobody 的用户身份来执行。具体的效果如下：

复制代码

```
1 root      7083  0.0  0.0  61032  5324 ?        Ss   Aug12   0:01 nginx: maste
2 nobody    331122 0.0  0.0  90768  31776 ?        S    11:44   0:00 nginx: worke
3 nobody    331123 0.0  0.0  90768  32720 ?        S    11:44   0:00 nginx: worke
4 nobody    331124 0.0  0.0  90768  31776 ?        S    11:44   0:00 nginx: worke
```


当然，也有一些工具不提供这类最小权限切换的功能，比如，在直接执行`redis-server`启动 Redis 的时候，就需要我们自己对用户身份进行切换。那用户身份切换怎么做呢？

我们首先来看 Nginx 的例子，在启动 Nginx 的时候，Linux 提供了 `nobody` 这么一个用户的身份。实际上，任何人进入 Linux 系统首先获得的用户身份就是 `nobody`，然后再从 `nobody` 进行登录，切换到其他正常用户身份上。

因此，**`nobody` 通常拥有整个操作系统中最小的权限**。所以，对于不提供最小权限切换功能的工具，我们就可以使用 `nobody` 的用户身份，来进行主动切换了。

在执行`redis-server`启动 Redis 的时候，我们就可以通过以下命令，以 `nobody` 的身份执行`redis-server`了（前提是，我们需要对日志和 PID 等目录进行适当配置，确保能够以 `nobody` 身份写入）：

```
1 su -s /bin/redis-server nobody
```

 复制代码

这样一来，我们就能通过“最小权限”原则，提升 Linux 系统授权的安全性了。

3.Linux 中的审计机制

我们在前面的课程中说过，“黄金法则”中的审计主要就是日志记录和分析。那么，Linux 系统中的日志都有哪些呢？在 Linux 系统中，系统的日志信息通常存储在 `/var/log` 目录下，部分应用程序也会把相关日志记录到这个目录中。系统日志主要分为 3 类，用户登录日志、特殊事件日志和进程日志。

用户登录日志主要是 `/var/log/wtmp` 和 `/var/run/utmp`，用来保存用户登录相关的信息。用户登录日志本身为二进制文件，我们无法直接通过文本方式查看，但是可以配合 `who/users/ac/last/lastlog` 这样的命令来获取。

特殊事件日志主要包括 `/var/log/secure` 和 `/var/log/message`。其中，`/var/log/secure` 主要记录认证和授权相关的记录，如果有人试图爆破 SSH，我们就可以从这个日志中观察出来。`/var/log/message` 由 `syslogd` 来维护，`syslogd` 这个守

守护进程提供了一个记录特殊事件和消息的标准机制，其他应用可以通过这个守护进程来报告特殊的事件。

进程日志：当通过 `accton` 来进行系统进程管理时，会生成记录用户执行命令的 `pacct` 文件。

默认情况下，Linux 会通过 `logrotate` 对日志执行相应的保留策略（比如日志切割和旧日志删除等）。通过配置 `/etc/logrotate.conf` 可以对不同日志的保留策略进行修改。

那如何对日志进行监控呢？这里，我向你推荐 2 种常见的日志分析工具 ELK 和 Zabbix，你可以利用这些工具来监控 Linux 的安全日志。也就是说，我们可以通过在这些分析平台配置恰当的规则（如 SSH 登录尝试失败 3 次以上），来及时发现黑客的部分入侵尝试，迅速产生报警。然后，我们就可以针对具体的问题，进行人工复查了。

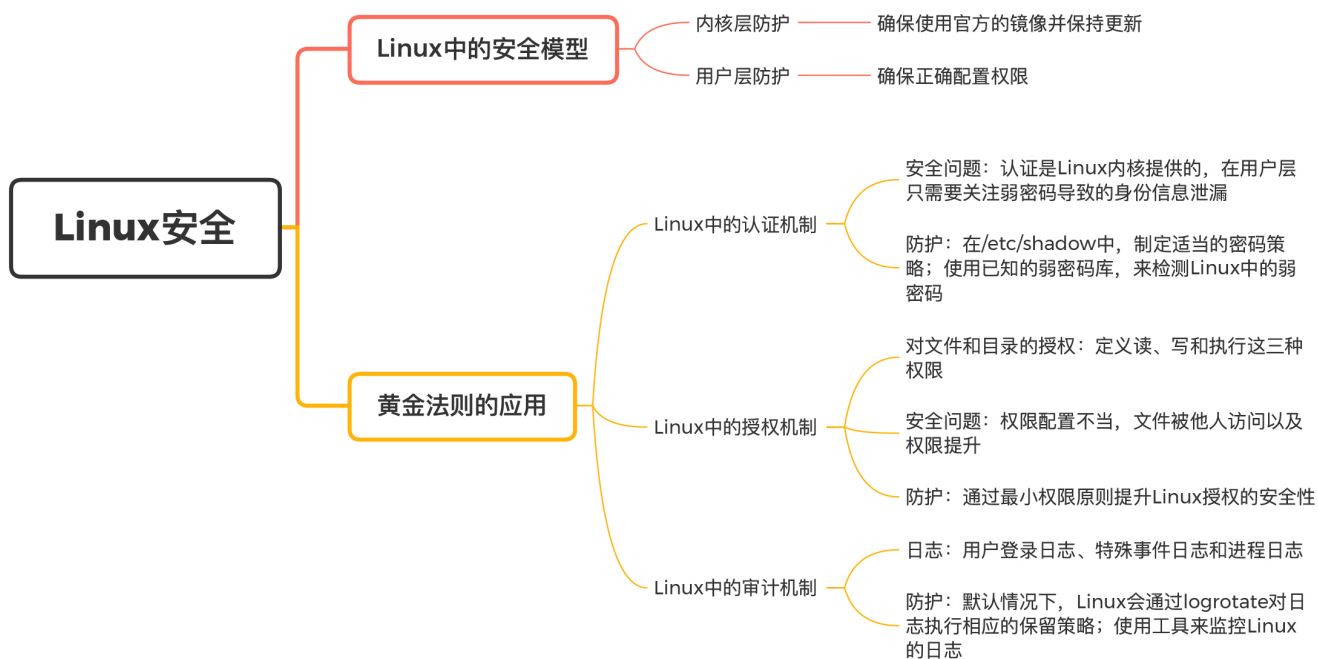
总结

好了，今天的内容讲完了。我们来一起总结回顾一下，你需要掌握的重点内容。

Linux 系统安全可以说是“最小权限”原则的最佳实践平台，尤其是当存在多用户共同维护和使用一台服务器的时候，正确的配置权限将是一件很有挑战的工作。为此，我们必须严格限制 ROOT 权限的使用。同时，为了避免进程漏洞，适当的通过 `iptables` 进行访问限制，也能够起到不错的保护效果。

在 Linux 系统的自我保护基础之上，也有一些安全工具能够为系统提供额外的保护功能（如杀毒软件、HIDS 等），在后续的内容中，我们会深入讲解这些工具。

最后，我把这一讲的重点内容梳理了一个脑图。你可以用它来查漏补缺，也可以自己来梳理看看，加深印象。



思考题

最后给你留一个思考题。

检查一下你的 Linux 服务器，看一下哪些用户具备 ROOT 权限？那些进程具备 ROOT 权限？这些用户和进程，真的需要 ROOT 权限吗？我们是否可以利用今天学到的知识，对这些 ROOT 权限进行限制呢？

欢迎留言和我分享你的思考和疑惑，也欢迎你把文章分享给你的朋友。我们下一讲再见！

点击查看 

来参加打卡，攻克 工作中 80% 的安全问题



PC端用户扫码参与



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 模块串讲（一） | Web安全：如何评估用户数据和资产数据面临的威胁？

下一篇 14 | 网络安全：和别人共用Wi-Fi时，你的信息会被窃取吗？

精选留言 (6)

 写留言



hello

2020-01-09

请教老师，您说的“为了避免进程漏洞，适当的通过 iptables 进行访问限制，也能够起到不错的保护效果”，这句话不太明白，能否例举一两个事实应用的例子，多谢！

作者回复: 可以详细了解下iptables的使用。简单来说，可以通过iptables阻止或者允许向Linux系统发起的请求。比如，我不希望ssh的22端口被随意的访问，那就通过iptables，给22端口设定几个白名单的ip。那么，对于非白名单的ip来说，本机的22端口就相当于下线了。这样一来，即使ssh存在弱密码，风险性也小很多。



1



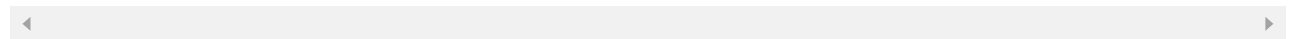
小晏子

2020-01-08

我的留言居然丢了...再写一下，
曾经使用过的linux服务器极少会用root账号登陆，系统管理员不会分配这个权限的，但是却有sudo用户权限，感觉sudo用户权限也是很高的权限了，个人认为给很多用户分配了sudo权限也是不安全的，应该限制，按照“最小权限”原则，应该区分用户权限，如果用户有安装软件需求，那就提交申请，不过这样会比较麻烦。

展开 ∨

作者回复: 如果只是两三个人公用的话还好，要是人多了，就容易出现问題，比如把别人的进程kill了之类的。另外，sudo启动的进程也是root的，root权限的进程过多，更容易对安全造成威胁。



💬 2

👍 1



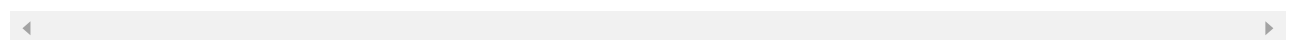
弹弹君

2020-01-09

针对公司很多服务器的情况，使用Ansible这类工具批量管理密码策略，iptables，日志等，是常见的做法吗？

展开 ∨

作者回复: Ansible属于运维类的工具，对于多服务器的管理是十分高效的，是比较常见的工具。



💬

👍



leslie

2020-01-09

切换到nobody用户去执行确实是个非常实用的招数：学到了一招了；其它方面之前做的还算可以。

💬

👍



小老鼠

2020-01-08

老师，咬字要清楚

展开 ∨

💬

👍



Cy23

2020-01-08

加深印象，看来我还差很多啊，基本上都给root权限了

展开 ∨

作者回复: 谨慎谨慎

