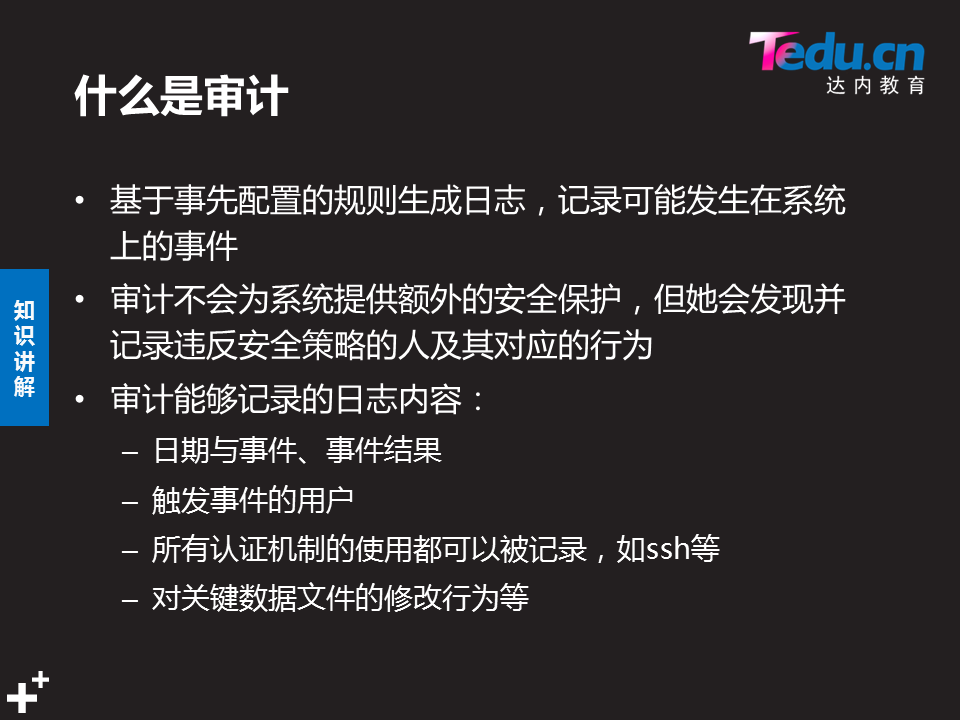
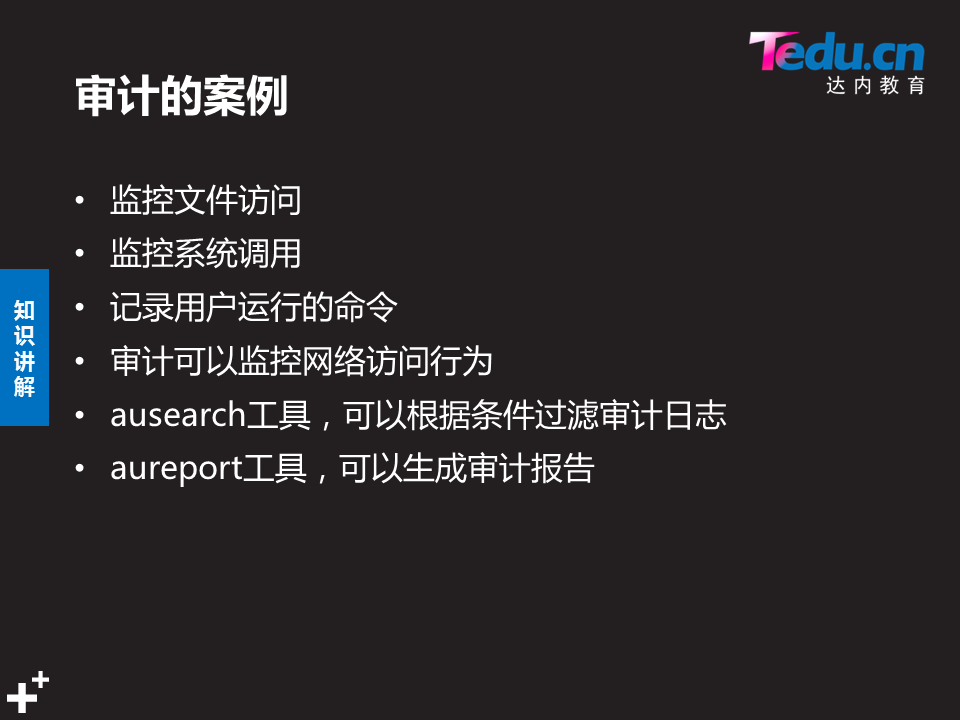
**[Top](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/SECURITY/DAY05/CASE/01/index.html" \l "page_top_case)**

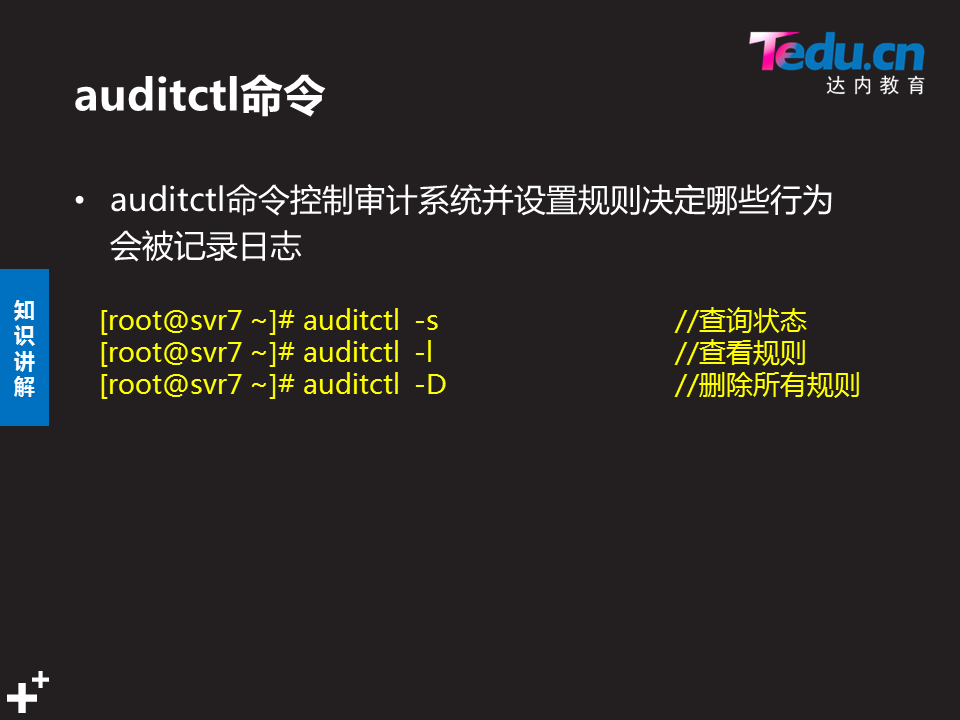
# NSD SECURITY DAY05

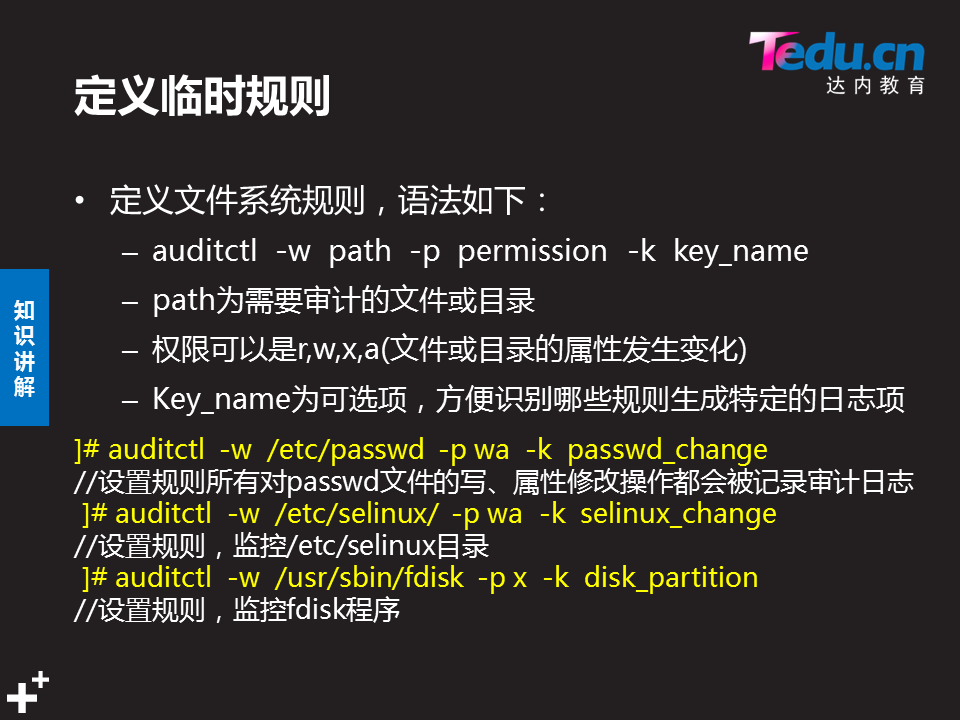
1. **[案例1：部署audit监控文件](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/SECURITY/DAY05/CASE/01/index.html" \l "case1)**

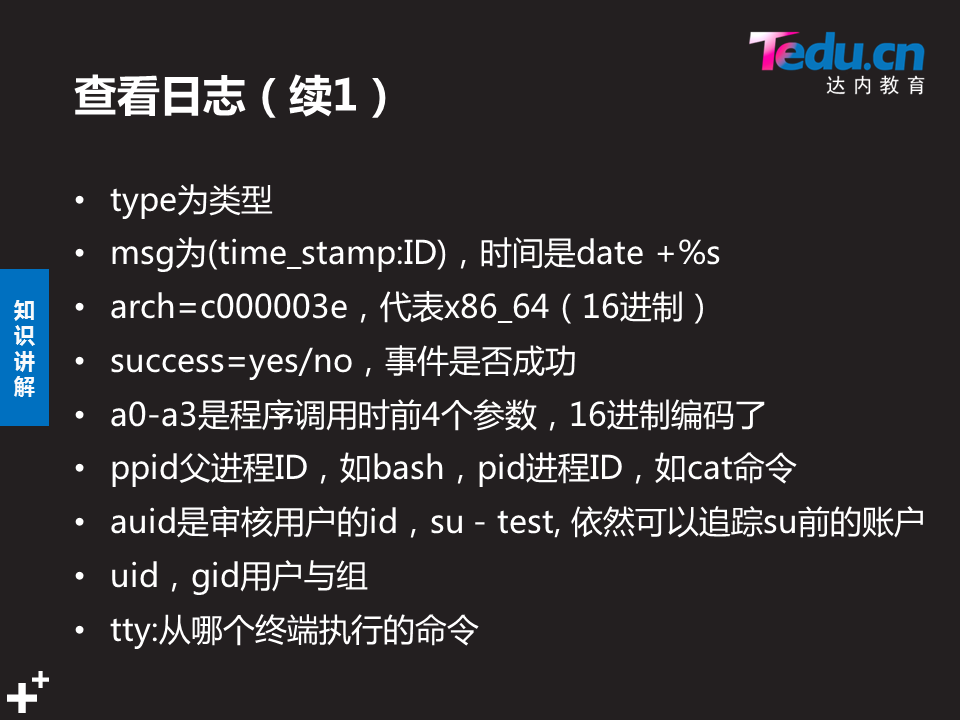


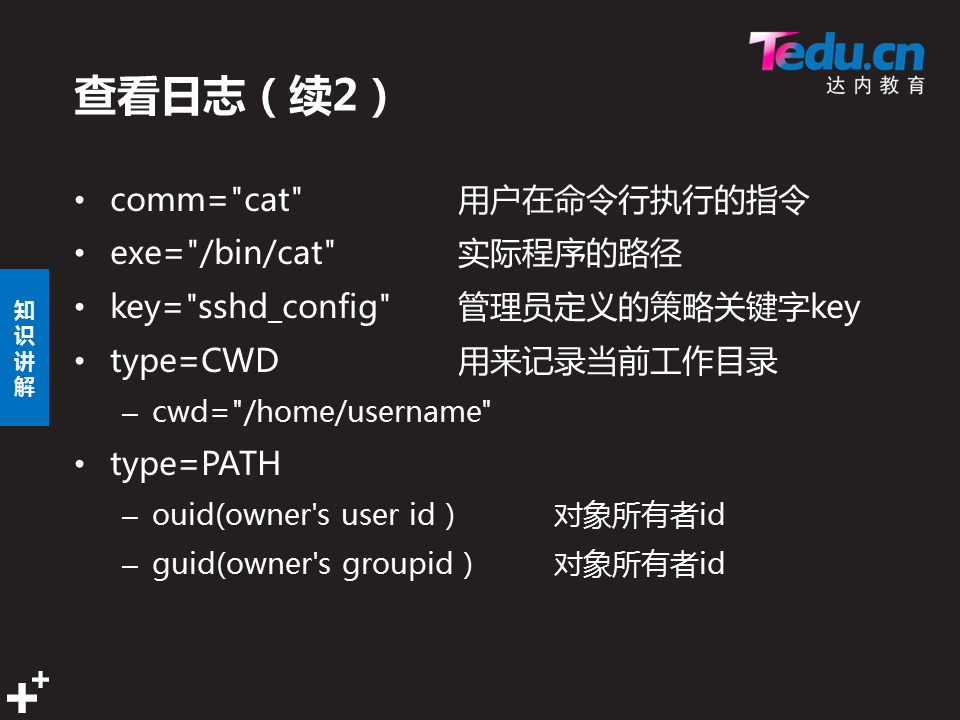






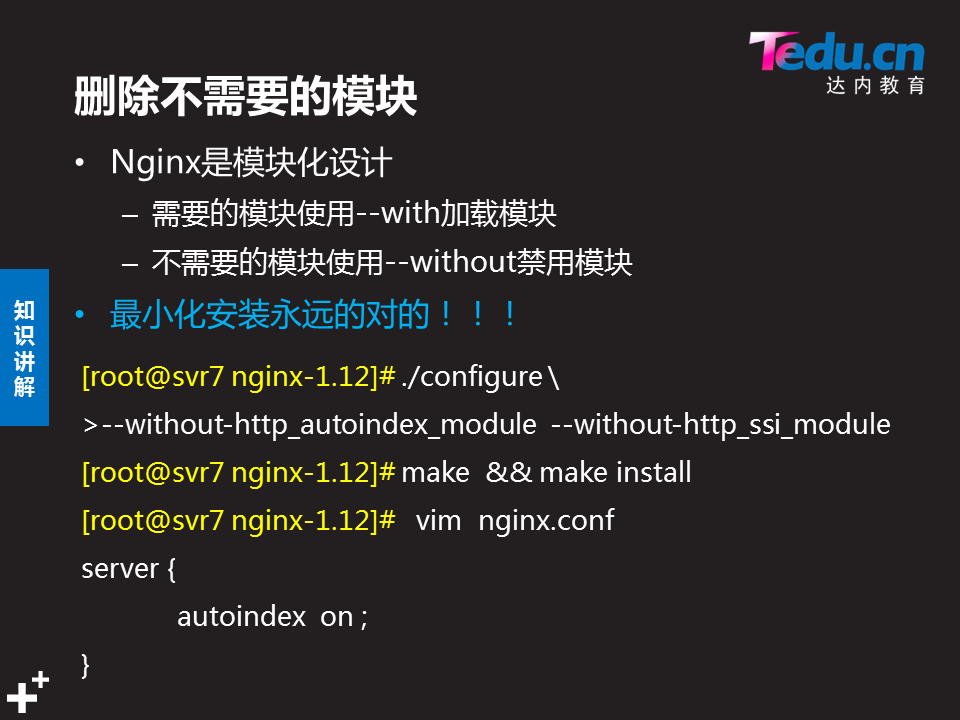




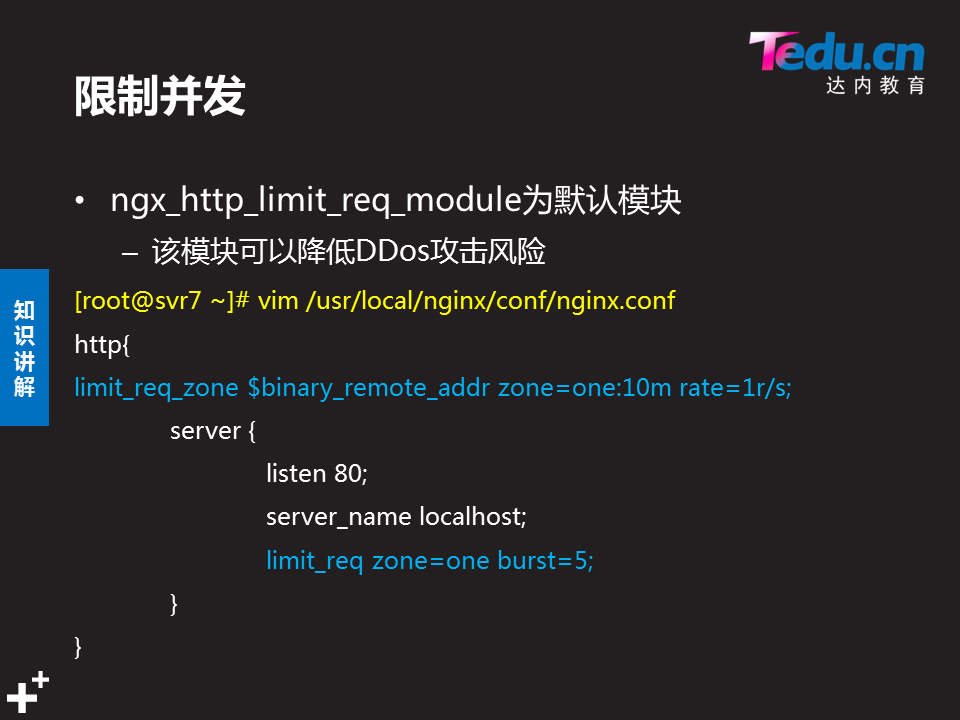


1. **[案例2：加固常见服务的安全](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/SECURITY/DAY05/CASE/01/index.html" \l "case2)**

**Nginx安全**



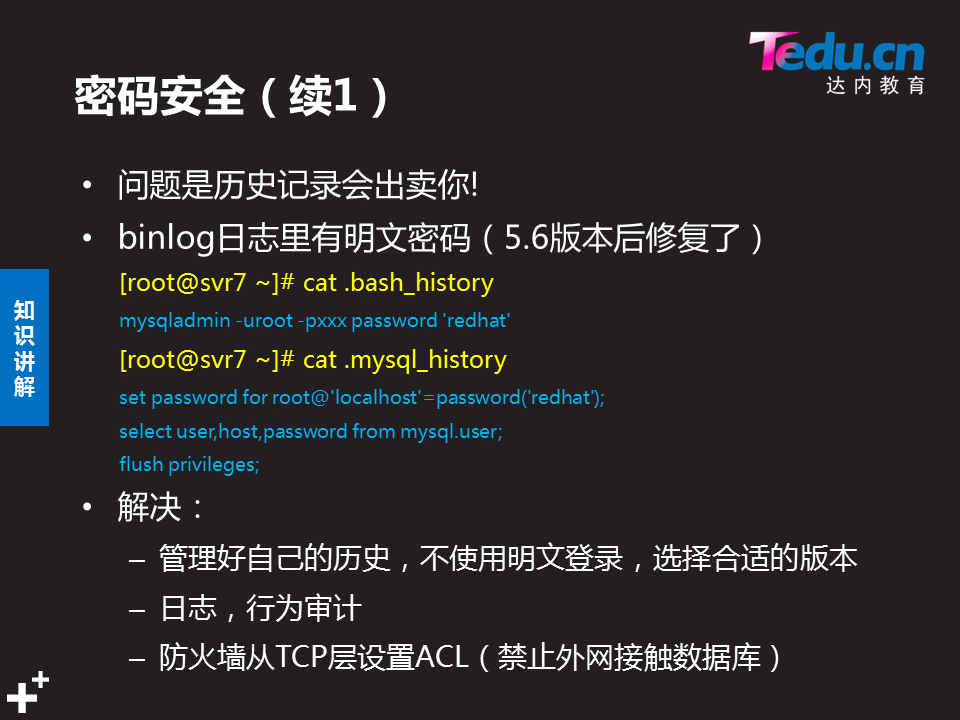






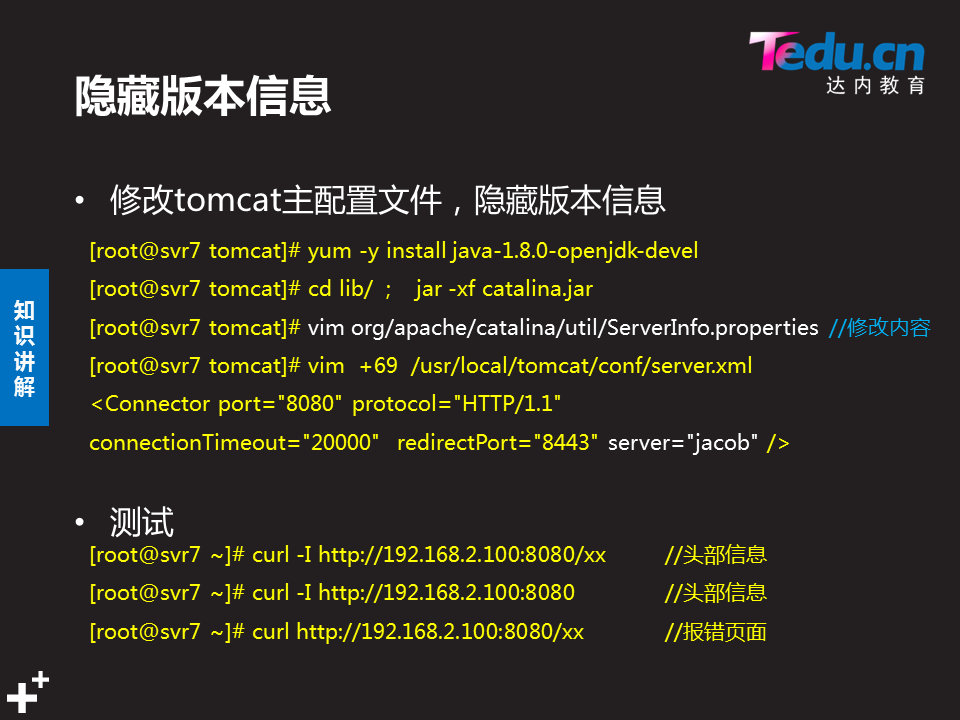
**数据库安全**





**数据安全**



**Tomact**



1. **[案例3：使用diff和patch工具打补丁](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/SECURITY/DAY05/CASE/01/index.html" \l "case3)**

## **1 案例1：部署audit监控文件**

### **1.1 问题**

本案例要求熟悉audit审计工具的基本使用，完成以下任务操作：

1. 使用audit监控/etc/ssh/sshd\_config
2. 当该文件发生任何变化即记录日志
3. 通过手动和ausearch工具查看日志内容

### **1.2 方案**

审计的目的是基于事先配置的规则生成日志，记录可能发生在系统上的事件（正常或非正常行为的事件），审计不会为系统提供额外的安全保护，但她会发现并记录违反安全策略的人及其对应的行为。

审计能够记录的日志内容：

* 日期与事件以及事件的结果
* 触发事件的用户
* 所有认证机制的使用都可以被记录，如ssh等
* 对关键数据文件的修改行为等都可以被记录

### **1.3 步骤**

实现此案例需要按照如下步骤进行。

**步骤一：配置audit审计系统**

1）安装软件包，查看配置文件（确定审计日志的位置）

1. **[**root@proxy **~]**# yum **-**y install audit                //安装软件包
2. **[**root@proxy **~]**# cat **/**etc**/**audit**/**auditd**.**conf            //查看配置文件，确定日志位置
3. log\_file **=** /var/log**/**audit**/**audit**.**log                //日志文件路径
4. **[**root@proxy **~]**# systemctl start auditd                //启动服务
5. **[**root@proxy **~]**# systemctl enable auditd            //设置开机自启

2）配置审计规则

可以使用auditctl命令控制审计系统并设置规则决定哪些行为会被记录日志。

语法格式如下：

1. **[**root@proxy **~]**# auditctl **-**s                        //查询状态
2. **[**root@proxy **~]**# auditctl **-**l                        //查看规则
3. **[**root@proxy **~]**# auditctl **-**D                        //删除所有规则

定义临时文件系统规则：

1. #语法格式：auditctl **-**w path **-**p permission **-**k key\_name
2. # path为需要审计的文件或目录
3. # 权限可以是r**,**w**,**x**,**a**(**文件或目录的属性发生变化**)**
4. # Key\_name为可选项，方便识别哪些规则生成特定的日志项
5. **[**root@proxy **~]**# auditctl **-**w **/**etc**/**passwd **-**p wa **-**k passwd\_change
6. //设置规则所有对passwd文件的写、属性修改操作都会被记录审计日志
7. **[**root@proxy **~]**# auditctl **-**w **/**etc**/**selinux**/** **-**p wa **-**k selinux\_change
8. //设置规则，监控/etc/selinux目录
9. **[**root@proxy **~]**# auditctl **-**w **/**usr**/**sbin**/**fdisk **-**p x **-**k disk\_partition
10. //设置规则，监控fdisk程序
11. **[**root@proxy **~]**# auditclt **-**w **/**etc**/**ssh**/**sshd\_conf **-**p warx **-**k sshd\_config
12. //设置规则，监控sshd\_conf文件

如果需要创建永久审计规则，则需要修改规则配置文件：

1. **[**root@proxy **~]**# vim **/**etc**/**audit**/**rules**.**d**/**audit**.**rules
2. **-**w **/**etc**/**passwd **-**p wa **-**k passwd\_changes
3. **-**w **/**usr**/**sbin**/**fdisk **-**p x **-**k partition\_disks

**步骤二：查看并分析日志**

1）手动查看日志

查看SSH的主配置文件/etc/ssh/sshd\_conf，查看audit日志信息：

1. **[**root@proxy **~]**# tailf **/var**/log/audit**/**audit**.**log
2. type**=**SYSCALL msg**=**audit**(1517557590.644:229228):** arch**=**c000003e
3. syscall**=2** success**=**yes exit**=3**
4. a0**=**7fff71721839 a1**=0** a2**=**1fffffffffff0000 a3**=**7fff717204c0
5. items**=1** ppid**=7654** pid**=7808** auid**=0** uid**=0** gid**=0** euid**=0** suid**=0**
6. fsuid**=0** egid**=0** sgid**=0** fsgid**=0** tty**=**pts2 ses**=3** comm**=**"cat"
7. exe**=**"/usr/bin/cat"
8. subj**=**unconfined\_u**:**unconfined\_r**:**unconfined\_t**:**s0**-**s0**:**c0**.**c1023 key**=**"sshd\_config"
9. **..** **..**
10. #内容分析
11. # type为类型
12. # msg为**(**time\_stamp**:**ID**)**，时间是date **+%**s（**1970-1-1**至今的秒数）
13. # arch**=**c000003e，代表x86\_64（**16**进制）
14. # success**=**yes**/**no，事件是否成功
15. # a0**-**a3是程序调用时前**4**个参数，**16**进制编码了
16. # ppid父进程ID，如bash，pid进程ID，如cat命令
17. # auid是审核用户的id，su **-** test**,** 依然可以追踪su前的账户
18. # uid，gid用户与组
19. # tty**:**从哪个终端执行的命令
20. # comm**=**"cat"         用户在命令行执行的指令
21. # exe**=**"/bin/cat"        实际程序的路径
22. # key**=**"sshd\_config"    管理员定义的策略关键字key
23. # type**=**CWD        用来记录当前工作目录
24. # cwd**=**"/home/username"
25. # type**=**PATH
26. # ouid**(**owner's user id）    对象所有者id
27. # guid(owner's groupid）    对象所有者id

2）通过工具搜索日志

系统提供的ausearch命令可以方便的搜索特定日志，默认该程序会搜索/var/log/audit/audit.log，ausearch options -if file\_name可以指定文件名。

1. **[**root@proxy **~]**# ausearch **-**k sshd\_config **-**i
2. //根据key搜索日志，-i选项表示以交互式方式操作

## **2 案例2：加固常见服务的安全**

### **2.1 问题**

本案例要求优化提升常见网络服务的安全性，主要完成以下任务操作：

1. 优化Nginx服务的安全配置
2. 优化MySQL数据库的安全配置
3. 优化Tomcat的安全配置

### **2.2 方案**

Nginx安全优化包括：删除不要的模块、修改版本信息、限制并发、拒绝非法请求、防止buffer溢出。

MySQL安全优化包括：初始化安全脚本、密码安全、备份与还原、数据安全。

Tomcat安全优化包括：隐藏版本信息、降权启动、删除默认测试页面.

### **2.3 步骤**

实现此案例需要按照如下步骤进行。

**步骤一：优化Nginx服务的安全配置**

1） 删除不需要的模块

Nignx是模块化设计的软件，需要什么功能与模块以及不需要哪些模块，都可以在编译安装软件时自定义，使用--with参数可以开启某些模块，使用--without可以禁用某些模块。最小化安装永远都是对的方案！

下面是禁用某些模块的案例：

1. **[**root@proxy **~]**# tar **-**xf nginx**-1.12.**tar**.**gz
2. **[**root@proxy **~]**# cd nginx**-1.12**
3. **[**root@proxy nginx**-1.12]**# **./**configure **\**
4. **>--**without**-**http\_autoindex\_module **\**            //禁用自动索引文件目录模块
5. **>--**without**-**http\_ssi\_module
6. **[**root@proxy nginx**-1.12]**# make
7. **[**root@proxy nginx**-1.12]**# make install

2） 修改版本信息，并隐藏具体的版本号

默认Nginx会显示版本信息以及具体的版本号，这些信息给攻击者带来了便利性，便于他们找到具体版本的漏洞。

如果需要屏蔽版本号信息，执行如下操作，可以隐藏版本号。

1. **[**root@proxy **~]**# vim **/**usr**/**local**/**nginx**/**conf**/**nginx**.**conf
2. … …
3. http**{**
4. server\_tokens off**;**                            //在http下面手动添加这么一行
5. … …
6. **}**
7. **[**root@proxy **~]**# nginx **-**s reload
8. **[**root@proxy **~]**# curl **-**I http**:**//192.168.4.5 //查看服务器响应的头部信息

但服务器还是显示了使用的软件为nginx，通过如下方法可以修改该信息。

1. **[**root@proxy nginx**-1.12]**# vim **+48** src**/**http**/**ngx\_http\_header\_filter\_module**.**c
2. //注意：vim这条命令必须在nginx-1.12源码包目录下执行！！！！！！
3. //该文件修改前效果如下：
4. **static** u\_char ngx\_http\_server\_string**[]** **=** "Server: nginx" CRLF**;**
5. **static** u\_char ngx\_http\_server\_full\_string**[]** **=** "Server: " NGINX\_VER CRLF**;**
6. **static** u\_char ngx\_http\_server\_build\_string**[]** **=** "Server: " NGINX\_VER\_BUILD CRLF**;**
7. //下面是我们修改后的效果：
8. **static** u\_char ngx\_http\_server\_string**[]** **=** "Server: Jacob" CRLF**;**
9. **static** u\_char ngx\_http\_server\_full\_string**[]** **=** "Server: Jacob" CRLF**;**
10. **static** u\_char ngx\_http\_server\_build\_string**[]** **=** "Server: Jacob" CRLF**;**
11. //修改完成后，再去编译安装Nignx，版本信息将不再显示为Nginx，而是Jacob
12. **[**root@proxy nginx**-1.12]**# **./**configure
13. **[**root@proxy nginx**-1.12]**# make **&&** make install
14. **[**root@proxy nginx**-1.12]**# killall nginx
15. **[**root@proxy nginx**-1.12]**# /usr/local**/**nginx**/**sbin**/**nginx            //启动服务
16. **[**root@proxy nginx**-1.12]**# curl **-**I http**:**//192.168.4.5            //查看版本信息验证

3） 限制并发量

DDOS攻击者会发送大量的并发连接，占用服务器资源（包括连接数、带宽等），这样会导致正常用户处于等待或无法访问服务器的状态。

Nginx提供了一个ngx\_http\_limit\_req\_module模块，可以有效降低DDOS攻击的风险，操作方法如下：

1. **[**root@proxy **~]**# vim **/**usr**/**local**/**nginx**/**conf**/**nginx**.**conf
2. … …
3. http**{**
4. … …
5. limit\_req\_zone $binary\_remote\_addr zone**=**one**:**10m rate**=**1r**/**s**;**
6. server **{**
7. listen **80;**
8. server\_name localhost**;**
9. limit\_req zone**=**one burst**=5;**
10. **}**
11. **}**
12. //备注说明：
13. //limit\_req\_zone语法格式如下：
14. //limit\_req\_zone key zone=name:size rate=rate;
15. //上面案例中是将客户端IP信息存储名称为one的共享内存，内存空间为10M
16. //1M可以存储8千个IP信息，10M可以存储8万个主机连接的状态，容量可以根据需要任意调整
17. //每秒中仅接受1个请求，多余的放入漏斗
18. //漏斗超过5个则报错
19. **[**root@proxy **~]**# /usr/local**/**nginx**/**sbin**/**nginx **-**s reload

客户端使用ab测试软件测试效果：

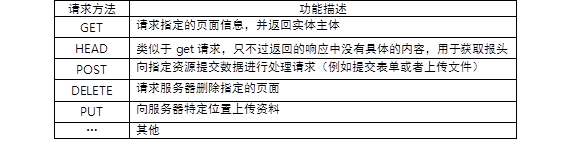
1. **[**root@client **~]**# ab **-**c **100** **-**n **100** http**:**//192.168.4.5/

4） 拒绝非法的请求

网站使用的是HTTP协议，该协议中定义了很多方法，可以让用户连接服务器，获得需要的资源。但实际应用中一般仅需要get和post。

具体HTTP请求方法的含义如表-1所示。

表-1 HTTP请求方法及含义



未修改服务器配置前，客户端使用不同请求方法测试：

1. **[**root@client **~]**# curl **-**i **-**X GET http**:**//192.168.4.5            //正常
2. **[**root@client **~]**# curl **-**i **-**X HEAD http**:**//192.168.4.5            //正常
3. //curl命令选项说明：
4. //-i选项：访问服务器页面时，显示HTTP的头部信息
5. //-X选项：指定请求服务器的方法

通过如下设置可以让Nginx拒绝非法的请求方法：

1. **[**root@proxy **~]**# vim **/**usr**/**local**/**nginx**/**conf**/**nginx**.**conf
2. http**{**
3. server **{**
4. listen **80;**
5. #这里，**!**符号表示对正则取反，**~**符号是正则匹配符号
6. #如果用户使用非GET或POST方法访问网站，则retrun返回**444**的错误信息
7. **if** **(**$request\_method **!~** **^(**GET**|**POST**)**$ **)** **{**
8. **return** **444;**
9. **}**
10. **}**
11. **}**
12. **[**root@proxy **~]**# /usr/local**/**nginx**/**sbin**/**nginx **-**s reload

修改服务器配置后，客户端使用不同请求方法测试：

1. **[**root@client **~]**# curl **-**i **-**X GET http**:**//192.168.4.5            //正常
2. **[**root@client **~]**# curl **-**i **-**X HEAD http**:**//192.168.4.5            //报错

4） 防止buffer溢出

当客户端连接服务器时，服务器会启用各种缓存，用来存放连接的状态信息。

如果攻击者发送大量的连接请求，而服务器不对缓存做限制的话，内存数据就有可能溢出（空间不足）。

修改Nginx配置文件，调整各种buffer参数，可以有效降低溢出风险。

1. **[**root@proxy **~]**# vim **/**usr**/**local**/**nginx**/**conf**/**nginx**.**conf
2. http**{**
3. client\_body\_buffer\_size 1K**;**
4. client\_header\_buffer\_size 1k**;**
5. client\_max\_body\_size 1k**;**
6. large\_client\_header\_buffers **2** 1k**;**
7. … …
8. **}**
9. **[**root@proxy **~]**# /usr/local**/**nginx**/**sbin**/**nginx **-**s reload

**步骤二：数据库安全**

1） 初始化安全脚本

安装完MariaDB或MySQL后，默认root没有密码，并且提供了一个任何人都可以操作的test测试数据库。有一个名称为mysql\_secure\_installation的脚本，该脚本可以帮助我们为root设置密码，并禁止root从远程其他主机登陆数据库，并删除测试性数据库test。

1. **[**root@proxy **~]**# systemctl status mariadb
2. //确保服务已启动
3. **[**root@proxy **~]**# mysql\_secure\_installation
4. //执行初始化安全脚本

2）密码安全

手动修改MariaDB或MySQL数据库密码的方法：

1. **[**root@proxy **~]**# mysqladmin **-**uroot **-**predhat password 'mysql'
2. //修改密码，旧密码为redhat，新密码为mysql
3. **[**root@proxy **~]**# mysql **-**uroot **-**pmysql
4. MariaDB **[(**none**)]>**set password **for** root@'localhost'**=**password**(**'redhat'**);**
5. //使用账户登录数据库，修改密码
6. MariaDB **[(**none**)]>** select user**,**host**,**password from mysql**.**user**;**
7. **+--------+---------+---------------------------------------------+**
8. **|** user     **|** host **|** password **|**
9. **+--------+---------+---------------------------------------------+**
10. **|** root     **|** localhost     **|** **\***84BB5DF4823DA319BBF86C99624479A198E6EEE9 **|**
11. **|** root     **|** **127.0.0.1**     **|** **\***84BB5DF4823DA319BBF86C99624479A198E6EEE9 **|**
12. **|** root     **|** **::1**     **|** **\***84BB5DF4823DA319BBF86C99624479A198E6EEE9 **|**
13. **+--------+-----------+--------------------------------------------+**

修改密码成功，而且密码在数据库中是加密的，有什么问题吗？问题是你的密码被明文记录了，下面来看看明文密码：

1. **[**root@proxy **~]**# cat **.**bash\_history
2. mysqladmin **-**uroot **-**pxxx password 'redhat'
3. //通过命令行修改的密码，bash会自动记录历史，历史记录中记录了明文密码
4. **[**root@proxy **~]**# cat **.**mysql\_history
5. set password **for** root@'localhost'**=**password**(**'redhat'**);**
6. select user**,**host**,**password from mysql**.**user**;**
7. flush privileges**;**
8. //通过mysql命令修改的密码，mysql也会有所有操作指令的记录，这里也记录了明文密码

另外数据库还有一个binlog日志里也有明文密码（5.6版本后修复了）。

怎么解决？

管理好自己的历史，不使用明文登录，选择合适的版本5.6以后的版本，

日志，行为审计（找到行为人），使用防火墙从TCP层设置ACL（禁止外网接触数据库）。

3）数据备份与还原

首先，备份数据库（注意用户名为root，密码为redhat）：

1. **[**root@proxy **~]**# mysqldump **-**uroot **-**predhat mydb table **>** table**.**sql
2. //备份数据库中的某个数据表
3. **[**root@proxy **~]**# mysqldump **-**uroot **-**predhat mydb **>** mydb**.**sql
4. //备份某个数据库
5. **[**root@proxy **~]**# mysqldump **-**uroot **-**predhat **--**all**-**databases **>** all**.**sql
6. //备份所有数据库

接下来，还原数据库（注意用户名为root，密码为redhat）：

1. **[**root@proxy **~]**# mysql **-**uroot **-**predhat mydb **<** table**.**sql            //还原数据表
2. **[**root@proxy **~]**# mysql **-**uroot **-**predhat mydb **<** mydb**.**sql            //还原数据库
3. **[**root@proxy **~]**# mysql **-**uroot **-**predhat **<** all**.**sql                    //还原所有数据库

4）数据安全

在服务器上（192.168.4.5），创建一个数据库账户：

1. **[**root@proxy **~]**# mysql **-**uroot **-**predhat
2. //使用管理员，登陆数据库
3. MariaDB **[(**none**)]>** grant all on **\*.\*** to tom@'%' identified by '123'**;**
4. //创建一个新账户tom

使用tcpdump抓包（192.168.4.5）

1. **[**root@proxy **~]**# tcpdump **-**w log **-**i any src or dst port **3306**
2. //抓取源或目标端口是3306的数据包，保存到log文件中

客户端（192.168.4.100）从远程登陆数据库服务器（192.168.4.5）

1. **[**root@client **~]**# mysql **-**utom **-**p123 **-**h **192.168.4.5**
2. //在192.168.4.100这台主机使用mysql命令登陆远程数据库服务器（192.168.4.5）
3. //用户名为tom，密码为123
4. MariaDB **[(**none**)]>** select **\*** from mysql**.**user**;**
5. //登陆数据库后，任意执行一条查询语句

回到服务器查看抓取的数据包

1. **[**root@proxy **~]**# tcpdump **-**A **-**r log
2. //使用tcpdump查看之前抓取的数据包，很多数据库的数据都明文显示出来

如何解决？

可以使用SSH远程连接服务器后，再从本地登陆数据库（避免在网络中传输数据，因为网络环境中不知道有没有抓包者）。

或者也可以使用SSL对MySQL服务器进行加密，类似与HTTP+SSL一样，MySQL也支持SSL加密（确保网络中传输的数据是被加密的）。

**步骤三：Tomcat安全性**

1） 隐藏版本信息、修改tomcat主配置文件（隐藏版本信息）

未修改版本信息前，使用命令查看服务器的版本信息

注意：proxy有192.168.2.5的IP地址，这里使用proxy作为客户端访问192.168.2.100服务器。

1. **[**root@proxy **~]**# curl **-**I http**:**//192.168.2.100:8080/xx
2. //访问不存在的页面文件，查看头部信息
3. **[**root@proxy **~]**# curl **-**I http**:**//192.168.2.100:8080
4. //访问存在的页面文件，查看头部信息
5. **[**root@proxy **~]**# curl http**:**//192.168.2.100:8080/xx
6. //访问不存在的页面文件，查看错误信息

修改tomcat配置文件，修改版本信息(在192.168.2.100操作)：

1. **[**root@web1 tomcat**]**# yum **-**y install java**-1.8.0-**openjdk**-**devel
2. **[**root@web1 tomcat**]**# cd **/**usr**/**local**/**tomcat**/**lib**/**
3. **[**root@web1 lib**]**# jar **-**xf catalina**.**jar
4. **[**root@web1 lib**]**# vim org**/**apache**/**catalina**/**util**/**ServerInfo**.**properties
5. //根据自己的需要，修改版本信息的内容
6. **[**root@web1 lib**]**# /usr/local**/**tomcat**/**bin**/**shutdown**.**sh        //关闭服务
7. **[**root@web1 lib**]**# /usr/local**/**tomcat**/**bin**/**startup**.**sh        //启动服务

修改后，客户端再次查看版本信息（在192.168.2.5操作）：

1. **[**root@proxy **~]**# curl **-**I http**:**//192.168.2.100:8080/xx
2. //访问不存在的页面文件，查看头部信息
3. **[**root@proxy **~]**# curl **-**I http**:**//192.168.2.100:8080
4. //访问存在的页面文件，查看头部信息
5. **[**root@proxy **~]**# curl http**:**//192.168.2.100:8080/xx
6. //访问不存在的页面文件，查看错误信息

再次修改tomcat服务器配置文件，修改版本信息，手动添加server参数（在192.168.2.100操作）：

1. **[**root@web1 lib**]**# vim **/**usr**/**local**/**tomcat**/**conf**/**server**.**xml
2. **<**Connector port**=**"8080" protocol**=**"HTTP/1.1"
3. connectionTimeout**=**"20000" redirectPort**=**"8443" server**=**"jacob" **/>**
4. **[**root@web1 lib**]**# /usr/local**/**tomcat**/**bin**/**shutdown**.**sh        //关闭服务
5. **[**root@web1 lib**]**# /usr/local**/**tomcat**/**bin**/**startup**.**sh        //启动服务

修改后,客户端再次查看版本信息（在192.168.2.5操作）：

1. **[**root@proxy **~]**# curl **-**I http**:**//192.168.2.100:8080/xx
2. //访问不存在的页面文件，查看头部信息
3. **[**root@proxy **~]**# curl **-**I http**:**//192.168.2.100:8080
4. //访问存在的页面文件，查看头部信息
5. **[**root@proxy **~]**# curl http**:**//192.168.2.100:8080/xx
6. //访问不存在的页面文件，查看错误信息

2）降级启动

默认tomcat使用系统高级管理员账户root启动服务，启动服务尽量使用普通用户。

1. **[**root@web1 **~]**# useradd tomcat
2. **[**root@web1 **~]**# chown **-**R tomcat**:**tomcat **/**usr**/**local**/**tomcat**/**
3. //修改tomcat目录的权限，让tomcat账户对该目录有操作权限
4. **[**root@web1 **~]**# su **-**c **/**usr**/**local**/**tomcat**/**bin**/**startup**.**sh tomcat
5. //使用su命令切换为tomcat账户，以tomcat账户的身份启动tomcat服务
6. **[**root@web1 **~]**# chmod **+**x **/**etc**/**rc**.**local                //该文件为开机启动文件
7. **[**root@web1 **~]**# vim **/**etc**/**rc**.**local                     //修改文件，添加如下内容
8. su **-**c **/**usr**/**local**/**tomcat**/**bin**/**startup**.**sh tomcat

3）删除默认的测试页面

1. **[**root@web1 **~]**# rm **-**rf **/**usr**/**local**/**tomcat**/**webapps/\*

## **3 案例3：使用diff和patch工具打补丁**

### **3.1 问题**

本案例要求优化提升常见网络服务的安全性，主要完成以下任务操作：

1. 使用diff对比文件差异
2. 使用diff生成补丁文件
3. 使用patch命令为旧版本打补丁

### **3.2 方案**

程序是人设计出来的，总是会有这样那样的问题与漏洞，目前的主流解决方法就是为有问题的程序打补丁，升级新版本。

在Linux系统中diff命令可以为我们生成补丁文件，然后使用patch命令为有问题的程序代码打补丁。

### **3.3 步骤**

实现此案例需要按照如下步骤进行。

**步骤一：对比单个文件差异**

1） 编写两个版本的脚本，一个为v1版本，一个为v2版本。

1. **[**root@proxy **~]**# cat test1**.**sh                                //v1版本脚本
2. #**!**/bin/bash
3. echo "hello wrld"
4. **[**root@proxy **~]**# cat test2**.**sh                                //v2版本脚本
5. #**!**/bin/bash
6. echo "hello the world"
7. echo "test file"

2） 使用diff命令语法

使用diff命令查看不同版本文件的差异。

1. **[**root@proxy **~]**# diff test1**.**sh test2**.**sh                     //查看文件差异
2. @@ **-1,3** **+1,3** @@
3. #**!**/bin/bash
4. **-**echo "hello world"
5. **-**echo "test"
6. **+**echo "hello the world"
7. **+**echo "test file"
8. **[**root@proxy **~]**# diff **-**u test1**.**sh test2**.**sh                 //查看差异，包含头部信息
9. **---** test1**.**sh    **2018-02-07** **22:20:02.723971251** **+0800**
10. **+++** test2**.**sh    **2018-02-07** **22:20:13.358760687** **+0800**
11. @@ **-1,3** **+1,3** @@
12. #**!**/bin/bash
13. **-**echo "hello world"
14. **-**echo "test"
15. **+**echo "hello the world"
16. **+**echo "test file"

diff制作补丁文件的原理：告诉我们怎么修改第一个文件后能得到第二个文件。

这样如果第一个版本的脚本有漏洞，我们不需要将整个脚本都替换，仅需要修改有问题的一小部分代码即可，diff刚好可以满足这个需求！

像Linux内核这样的大块头，一旦发现有一个小漏洞，我们不可能把整个内核都重新下载，全部替换一遍，而仅需要更新有问题的那一小部分代码即可！

diff命令常用选项：

-u 输出统一内容的头部信息（打补丁使用），计算机知道是哪个文件需要修改

-r 递归对比目录中的所有资源（可以对比目录）

-a 所有文件视为文本（包括二进制程序）

-N 无文件视为空文件（空文件怎么变成第二个文件）

-N选项备注说明：

A目录下没有txt文件，B目录下有txt文件

diff比较两个目录时，默认会提示txt仅在B目录有（无法对比差异，修复文件）

diff比较时使用N选项，则diff会拿B下的txt与A下的空文件对比，补丁信息会明确说明如何从空文件修改后变成txt文件，打补丁即可成功！

**步骤二：使用patch命令对单文件代码打补丁**

1）准备实验环境

1. **[**root@proxy **~]**# cd demo
2. **[**root@proxy demo**]**# vim test1**.**sh
3. #**!**/bin/bash
4. echo "hello world"
5. echo "test"
6. **[**root@proxy demo**]**# vim test2**.**sh
7. #**!**/bin/bash
8. echo "hello the world"
9. echo "test file"

2) 生成补丁文件

1. **[**root@proxy demo**]**# diff **-**u test1**.**sh test2**.**sh **>** test**.**patch

3）使用patch命令打补丁

在代码相同目录下为代码打补丁

1. **[**root@proxy demo**]**# yum **-**y install patch
2. **[**root@proxy demo**]**# patch **-**p0 **<** test**.**patch                    //打补丁
3. patching file test1**.**sh
4. //patch -pnum（其中num为数字，指定删除补丁文件中多少层路径前缀）
5. //如原始路径为/u/howard/src/blurfl/blurfl.c
6. //-p0则整个路径不变
7. //-p1则修改路径为u/howard/src/blurfl/blurfl.c
8. //-p4则修改路径为blurfl/blurfl.c
9. //-R(reverse)反向修复，-E修复后如果文件为空，则删除该文件
10. **[**root@proxy demo**]**# patch **-**RE **<** test**.**patch                     //还原旧版本，反向修复

**步骤三：对比目录中所有文件的差异**

1） 准备实验环境

1. **[**root@proxy **~]**# mkdir demo
2. **[**root@proxy **~]**# cd demo
3. **[**root@proxy demo**]**# mkdir **{**source1**,**source2**}**
4. **[**root@proxy demo**]**# echo "hello world" **>** source1**/**test**.**sh
5. **[**root@proxy demo**]**# cp **/**bin**/**find source1**/**
6. **[**root@proxy demo**]**# tree source1**/**                        //source1目录下2个文件
7. **|--** find
8. `**--** test**.**sh
9. **[**root@proxy demo**]**# echo "hello the world" **>** source2**/**test**.**sh
10. **[**root@proxy demo**]**# echo "test" **>** source2**/**tmp**.**txt
11. **[**root@proxy demo**]**# cp **/**bin**/**find source2**/**
12. **[**root@proxy demo**]**# echo "1" **>>** source2**/**find
13. **[**root@proxy demo**]**# tree source2**/**                        //source1目录下3个文件
14. **|--** find
15. **|--** test**.**sh
16. `**--** tmp**.**txt
17. //注意：两个目录下find和test.sh文件内容不同，source2有tmp.txt而source1没有该文件

2）制作补丁文件

1. **[**root@proxy demo**]**# diff **-**u source1**/** source2**/**
2. //仅对比了文本文件test.sh；二进制文件、tmp都没有对比差异，仅提示，因为没有-a和-N选项
3. **[**root@proxy demo**]**# diff **-**Nu source1**/** source2**/**
4. //对比了test.sh，并且使用source2目录的tmp.txt与source1的空文件对比差异。
5. **[**root@proxy demo**]**# diff **-**Nua source1**/** source2**/**
6. //对比了test.sh、tmp.txt、find(程序)。

**步骤四：使用patch命令对目录下的所有代码打补丁**

1）使用前面创建的source1和source2目录下的代码为素材，生成补丁文件

1. **[**root@proxy **~]**# cd demo
2. **[**root@proxy demo**]**# diff **-**Nuar source1**/** source2**/** **>** source**.**patch

2）使用patch命令为代码打补丁

1. **[**root@proxy demo**]**# ls
2. source1 source2 source**.**patch
3. **[**root@proxy demo**]**# cat source**.**patch                //对比的文件有路径信息
4. **---** source1**/**test**.**sh **2018-02-07** **22:51:33.034879417** **+0800**
5. **+++** source2**/**test**.**sh **2018-02-07** **22:47:32.531754268** **+0800**
6. @@ **-1** **+1** @@
7. **-**hello world
8. **+**hello the world
9. **[**root@proxy demo**]**# cd source1
10. **[**root@proxy source1**]**# patch **-**p1 **<** **../**source**.**patch