**系统审计**

概述

什么是审计：

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

审计不会在系统提供额外的安全保护，但会发现并记录违反安全策略的人以及对应的行为

审计能够记录的日志内容

日期与事件、事件结果

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

审计的案例

监控文件访问

监控系统调用

记录用户运行的命令

审计可以监控网络访问行为

ausearch工具，可以根据条件过滤审计日志

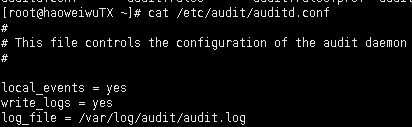
aureport工具，可以生成审计报告

部署audit

使用设计系统需要安装audit软件包

主配置文件为/etc/audit/auditd.conf





配置审计

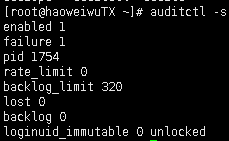
auditctl命令

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

auditctl -s //查询状态

auditctl -l //查看规则

auditctl -D //删除所有规则





定义临时规则

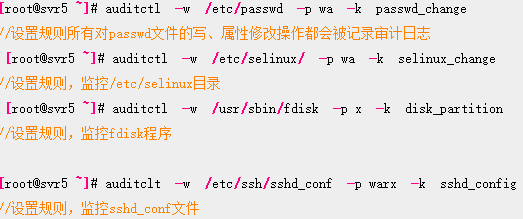
定义文件系统规则，语法如下：

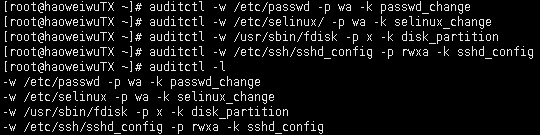
auditctl -w path -p permission -k key\_name

path为需要审计的文件或目录

权限可以是r,w,x,a（文件或目录的属性发生变化）

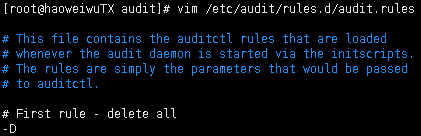
key\_name为可选项，方便识别哪些规则生成特定的日志项

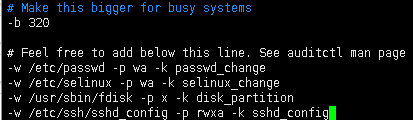




定义永久规则

写入配置文件/etc/audit/rules.d/audit.rules





附加扩展

扩展知识

通过审核也可以监控系统调用

规则参考模版

审计日志

查看日志

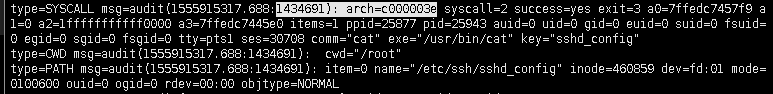
定义规则

auditctl -w /etc/ssh/sshd\_config -p warx -k sshd\_config

查看日志

tailf /var/log/audit/audit.log





type为类型

msg为（time\_stamp:ID），时间是date +%s

arch=c000003e，代表x86\_64（16进制）

success=yes/no，事件是否成功

a0-a3是程序调用时前4个参数，16进制编码

ppid父进程ID

auid是审核用户的id,su - test,依然可以追踪su前的用户

uid,gid用户和组

tty：从哪个终端执行的命令

comm=”cat”：用户在命令行执行的指令

exe=”/bin/cat” 实际程序的路径

key=”sshd\_config”管理员定义的策略关键字key

type=CWD

cwd=”/home/username”用来记录当前工作目录

type=PATH

ouid(owner’s user id)对象所有者id

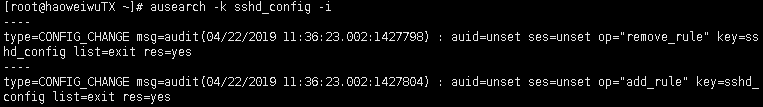
guid(owner’s group id)对象所有组id

搜索日志

系统提供的ausearch命令可以方便的搜索特定日志

默认程序会搜索/var/log/audit/audit.log

ausearch options -if file\_name可以指定文件名



**服务安全**

nginx安全

删除不需要的模块

需要的模块使用--with加载模块

不需要的模块使用--without禁用模块

./configure --without-http\_autoindex\_module

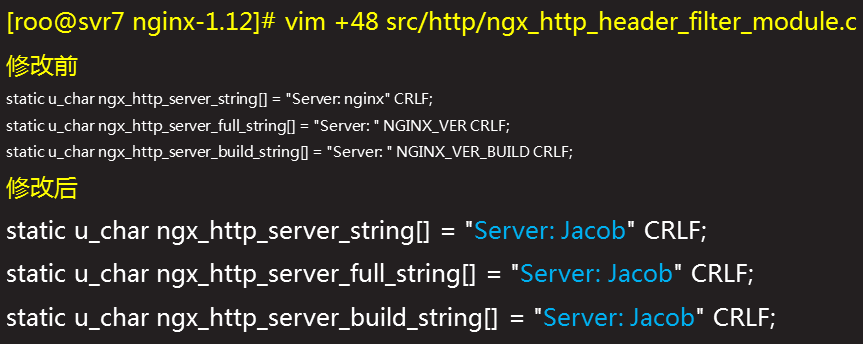
make && make install



修改版本信息

修改版本信息（修改源码）

隐藏版本号信息（server\_tokens off）



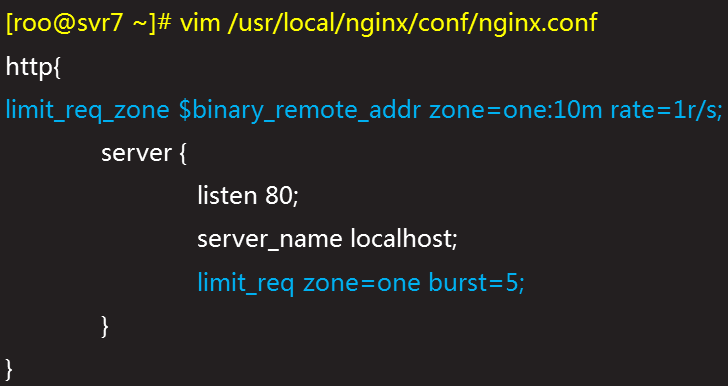


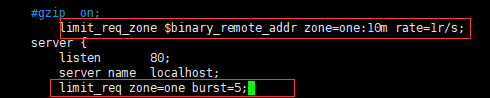


限制并发

ngx\_http\_limit\_req\_module为默认模块

该模块可以降低DDOS攻击风险





下面配置的功能是：

语法：limit\_req\_zone key zone=name:size rate=rate

将客户端IP信息存储名为one的共享内存，空间为10M

1M可以存储8千个IP的信息，10M存8万个主机状态

每秒钟仅接受1个请求，多余的放入漏斗，处理完一个，则会从漏斗里面取出一个再处理

漏斗超过5个则会报错和丢弃

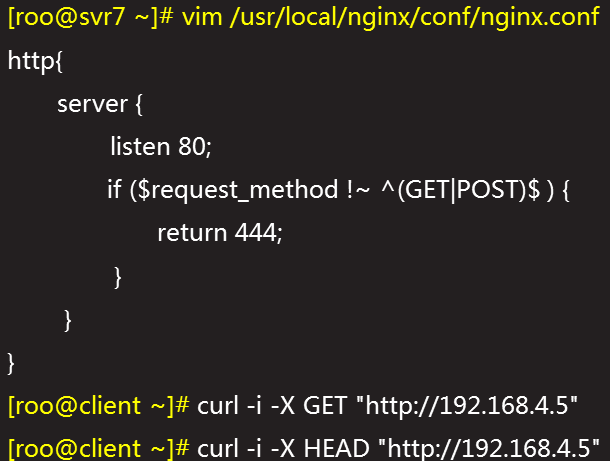
拒绝非法请求

常见HTTP请求方法

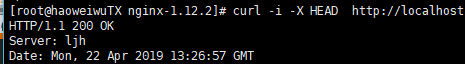
HTTP定义了很多方法，实际应用中一般仅需要get和post

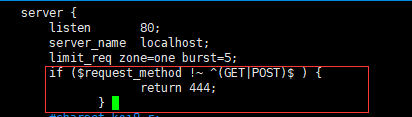
|  |  |
| --- | --- |
| 请求方法 | 功能描述 |
| GET | 请求指定的页面信息，并返回实体主体 |
| HEAD | 类似于GET请求，只不过返回的响应中没有具体的内容，用于获取报头 |
| POST | 向指定资源提交数据进行处理请求（例如提交表单或者上传文件） |
| DELETE | 请求服务器删除指定的页面 |
| PUT | 向服务器特定位置上传资料 |
| ... | 其他 |

禁用其他方法，仅允许（POST|GET）



-i显示头部信息，-X指定访问方法



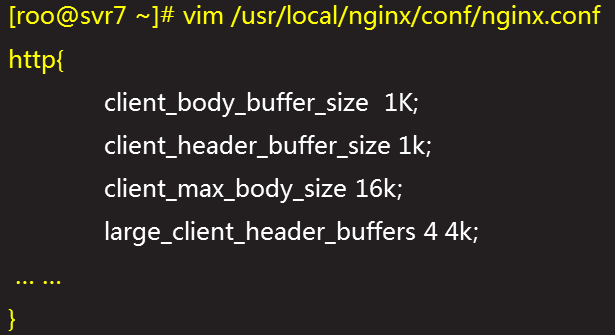




防止buffer溢出

防止客户端请求数据溢出

有效降低机器Dos攻击风险



数据库安全

初始化安全脚本mysql\_secure\_installation

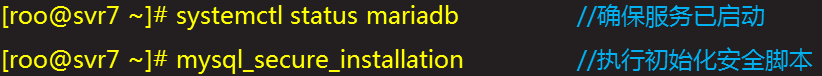
输入新密码，配置新root密码

remove anonymous users（删除匿名用户）

disallow root login remotely（禁止root远程登录）

remove test database（删除测试数据库）

reload privilege（刷新权限）



密码安全

修改mysql密码的若干方法

mysqladmin -uroot -p123456 password ‘mysql’



mysql -uroot -p123456

set password for root@’localhost’=password(‘redhat’)

清理历史（不显示明文密码）：.mysql\_history、.bash\_history、binlog（mysql目录里，5.6版本后已修复）

数据的备份与还原

备份

mysqldump -uroot -predhat mydb table > table.sql

mysqldump -uroot -predhat mydb > mydb.sql

mysqldump -uroot -predhat --all-databases > all.sql





还原

mysql -uroot -predhat mydb < table.sql //还原表

mysql -uroot -predhat mydb < mydb.sql //还原数据库

mysql -uroot -predhat <all.sql //还原所有



数据安全

创建可以远程登录的账户

mysql -uroot -predhat

grant all on \*.\* to tom@’%’ identified by ‘123’;

使用tcpdump抓包

tcpdump -w log -i eth0 src or dst port 3306

客户端远程登录数据库，查看抓包数据

client：mysql -utom -p123 -h 192.168.4.5

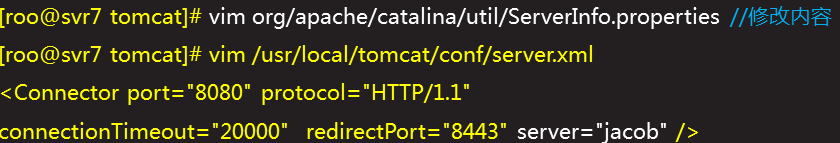
svr：tcpdump -A -r log

解决：使用SSL或SSH加密数据传输

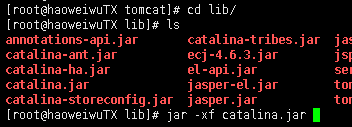
Tomcat安全

隐藏版本信息

修改tomcat主配置文件，隐藏版本信息



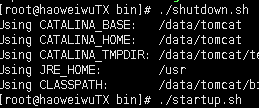


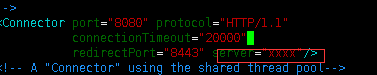












测试

curl -I <http://192.168.2.100:8080/xx> //头部信息

curl -I <http://192.168.2.100:8080> //头部信息

curl <http://192.168.2.100:8080/xx> //报错页面

降权启动

使用非root启动tomcat服务

useradd tomcat

chown -R tomcat:tomcat /data/tomcat

su -c /data/tomcat/bin/startup.sh tomcat

开机启动

chmod +x /etc/rc.local

vim /etc/rc.local

su -c /data/tomcat/bin/startup.sh tomcat

删除默认测试页面

rm -rf /data/tomcat/webapps/\*

**linux安全之打补丁（diff生成补丁包，patch打补丁）**

补丁的原理

源代码的不同版本

V1版本

cat test1.sh

echo “hello wrld”

V2版本（修复错误，添加功能）

cat test2.sh

#!/bin/bash

echo “hello world”

echo “test file”

diff逐行比较

diff的原则是：

告诉我们怎么修改第一个文件后能得到第二个文件

选项：

-u 对比差异，怎么从第一个文件变为第二个文件（打补丁使用）

-r 递归对比目录中的所有资源（可以对比目录）

-a 所有文件均视为文本（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）

//A目录下没有txt文件，B目录下有txt文件

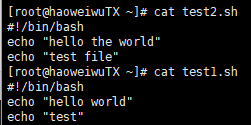
//diff比较两个目录时，默认会提示txt仅在B目录有（无法根据补丁修复A缺失的文件）

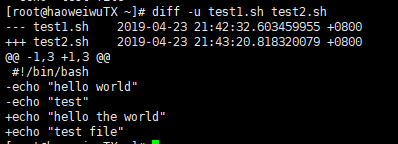
//diff比较时使用N选项，则diff会拿B下的txt与A下的空文件对比

//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

diff文件对比

仅对文件比较





删除第一个的前两行，再加上后两行

diff目录比较

准备环境

mkdir {source1,source2}

echo “hello world” > source1/test.sh

echo “hello the world” > source2/test.sh

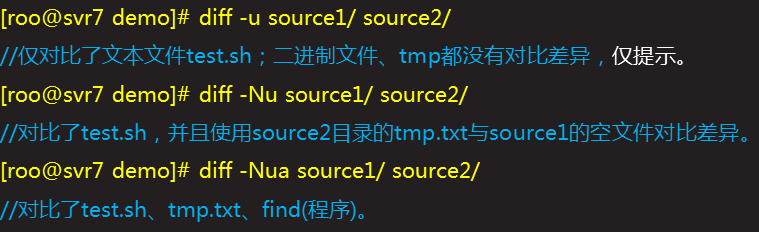
echo “test” > source2/tmp.txt

cp /bin/find source1/

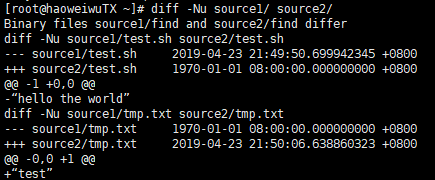
cp /bin/find source2/

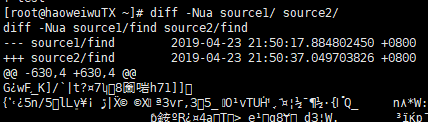
echo “1” >> source2/find

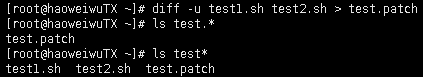
对比差异









生成补丁文件

patch打补丁

给文件生成打补丁

准备环境，生成补丁文件

mkdir demo;cd demo

vim test1.sh

#!/bin/bash

echo “hello world”

echo “test”

vim test2.sh

#!/bin/bash

echo “”hello the world”

echo “test file”

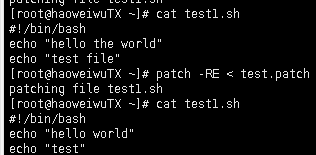
diff -Nua test1.sh test2.sh > test.patch

对旧版本的代码，使用补丁即可更新，而不需要下载完成新代码（旺旺完整的程序很大）

yum -y install patch

patch -p0 < test.patch

打补丁



patch -RE < test.patch //还原旧版本（反向修复）

//patch -pnum指定删除补丁文件中多少层路径前缀

//如原始路径为/u/howard/src/blufl/blurfl.c

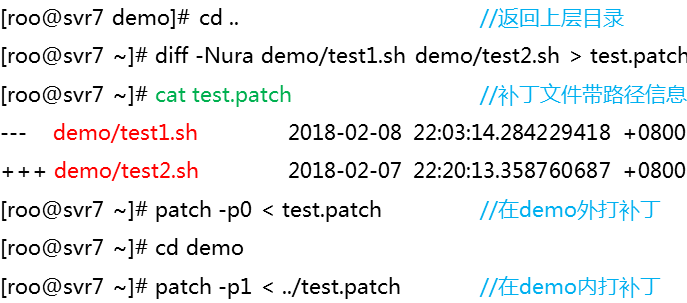
//-p0则整个路径不变

//-p1则修改路径为howard/src/blufl/blurfl.c

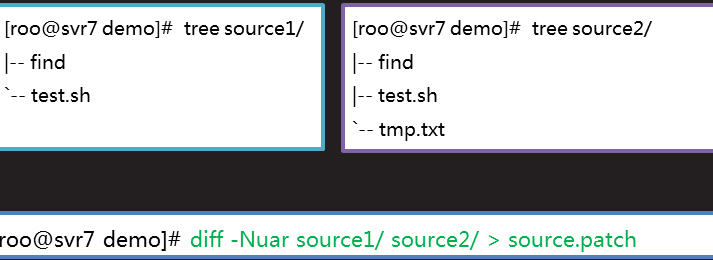
//-p4则修改路径为blurfl.c

//-R(reverse)反向修复，-E修复后如果文件为空，则删除该文件

在不同的目录生成补丁文件



使用前面创建的目录环境



对就版本的代码，使用补丁即可更新，而不需要下载完整的新代码（往往完整的程序很大）

