# 12_security05 系统审计+服务安全+Linux 安全之打补丁

## 一 系统审计-概述

### 1.1 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

审计不会为系统提供额外的安全保护,但它会发现并记录违反安全策略的人及其行为

审计能够记录日志内容:

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

### 1.2 审计的案例

1.2.1 监控文件访问

1.2.2 监控系统调用

1.2.3 记录用户运行的命令

1.2.4 审计可以监控网络访问行为

**1.2.5 auserch -k key_name 工具,可以根据条件过滤审计日志**

**1.2.6 aureport 工具,可以生成审计报告**

1.3 部署 audit

```
svr72 ~]# yum -y install audit
```

主配置文件:/etc/audit/auditd.conf  #不需要修改

日志文件:/var/log/audit/audit.log  #该路径由主配置文件定义

服务名称:auditd.service

启动服务:

svr72 ~]# systemctl start auditd.service  #启动服务,默认不能停

svr72 ~]# systemctl status auditd.service

二 系统审计-配置审计

**2.1 auditctl** 命令:

auditctl -s 查询状态

auditctl -l 查看规则

auditctl -D 删除所有规则

**2.2** 定义临时规则,命令行设置,即刻生效

**auditctl -w path -p permission -k key_name**

**path** 为需要审计的文件或目录

**permission** 权限,可以是 **r,w,x,a(**文件或目录的属性[大小\归属等]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

svr72 ~]# auditctl -w /etc/passwd -p wa -k passwd_change

设置所有对 passwd 文件的写\属性修改操作都会被记录到审计日志

svr72 ~]# auditctl -w /etc/selinux/ -p wa -k selinux_change

设置规则,监控/etc/selinux 目录

```
[root@svr72 ~]# which fdisk
svr72 ~]# audit -w /usr/sbin/fdisk -p x -k disk_partion
```

设置规则,监控 fdisk 程序

```
svr72 ~]# auditctl -l
-w /etc/passwd -p wa -k passwd_change
-w /etc/selinux -p wa -k selinux_change
-w /usr/sbin/fdisk -p x -k fdisk_change
```

**2.3 定义永久规则,配置文件:/etc/audit/rules.d/audit.rules**

```
svr72 ~]# vim /etc/audit/rules.d/audit.rules
```

**最下面添加以下 3 行:(命令行中用 auditctl -l 显示出来的结果)**

```
-w /etc/passwd -p wa -k passwd_change
-w /etc/selinux -p wa -k selinux_change
-w /usr/sbin/fdisk -p x -k fdisk_change
```

**2.4 测试**

```
svr72 ~]# useradd bob
svr72 ~]# fdisk -l
```

## 三 系统审计-审计日志

### 3.1 查看日志

```
svr72 ~]# ls /var/log/audit/audit.log
```

### 3.2 搜索日志

```
svr72 ~]# ausearch -k fdisk_change
time->Tue Sep 24 11:17:46 2019          #日志生成时间
uid=0                                   #命令执行用户
comm="fdisk" exe="/usr/sbin/fdisk"      #执行的命令及命令路径
argc=2 a0="fdisk" a1="-l"               #执行的命令的详细
success=yes                             #是否执行成功

@svr72 ~]# ausearch -k passwd_change
time->Tue Sep 24 11:17:33 2019
uid=0
comm="useradd" exe="/usr/sbin/useradd"
success=yes
#无 argc 执行的命令的详细
```

### 3.3 生成审计报告

```
svr72 ~]# aureport
```

## 四 服务安全-nginx 安全

```
room9pc01 ~]$ scp /linux-soft/03/redis/lnmp/nginx-1.12.2.tar.gz
root@192.168.4.72:/root
svr72 ~]# yum -y install gcc pcre-devel zlib-devel  #安装依赖包
svr72 ~]# tar -xf nginx-1.12.2.tar.gz       #解压
svr72 ~]# cd nginx-1.12.2/                  #进入目录
svr72 nginx-1.12.2]# ./configure            #编译检测
svr72 nginx-1.12.2]# make && make install   #编译及安装
svr72 ~]# echo "web72" > /usr/local/nginx/html/test.html
svr72 ~]# /usr/local/nginx/sbin/nginx       #启动 nginx 服务
svr72 ~]# ss -antulp | grep :80             #端口查看
svr72 ~]# firefox http://192.168.4.72/test.html
```

### 4.1 删除不需要的模板(在安装目录下执行./configure --help 查看模块)

nginx 是模块化设计的,

需要的模块使用--with 加载模块,

不需要模块使用--without 禁用模块,

开启 antoindex 自动索引功能

```
svr72 ~]# vim /usr/local/nginx/conf/nginx.conf
   server {
```

```
autoindex on;  #在 server 行下添加此行,开启 autoindex
```

创建测试文件

```
svr72 ~]# mkdir /usr/local/nginx/html/game

svr72 ~]# echo "aaa" > /usr/local/nginx/html/game/a.html

svr72 ~]# echo "bbb" > /usr/local/nginx/html/game/b.html

svr72 ~]# echo "ccc" > /usr/local/nginx/html/game/c.html
```

重启服务

```
svr72 ~]# /usr/local/nginx/sbin/nginx -s stop

svr72 ~]# /usr/local/nginx/sbin/nginx
```

查看 game 页面

```
svr72 ~]# firefox http://192.168.4.72/game/
```


此类情况下,页面的源代码容易泄露,不建议开启自动索引功能

停止 nginx 服务

```
svr72 ~]# /usr/local/nginx/sbin/nginx -s stop
```

安装目录内重新编译检查及编译

```
svr72 nginx-1.12.2]# ./configure --without-http_autoindex_module
svr72 nginx-1.12.2]# make
```

修改配置文件,注释自动索引语句,重启 nignx 服务

```
svr72 ~]# vim /usr/local/nginx/conf/nginx.conf
 #autoindex on;    #注释该行
svr72 ~]# /usr/local/nginx/sbin/nginx   #重启服务
```

检查端口及访问网页检查自动索引共能是否关闭

```
svr72 ~]# ss -antulp | grep :80
```

此时再 svr72 ~]# firefox http://192.168.4.72/game/显示 forbidden

但 firefox http://192.168.4.72/game/a(b\c).html 能正常访问

**4.2 修改版本信息**

获取版本信息

```
svr72 ~]# curl -I http://192.168.4.72/test.html #第 2 行 Server...
```

停止服务

```
svr72 ~]# /usr/local/nginx/sbin/nginx -s stop
```

进入安装目录,编辑 src/http/ngx_http_header_filter_module.c

```
svr72 ~]# vim
/root/nginx-1.12.2/src/http/ngx_http_header_filter_module.c
 49 static u_char ngx_http_server_string[]
```

= **"Server: tom" CRLF;**

50 static u_char ngx_http_server_full_string[]

= **"Server: tom" CRLF;**

51 static u_char ngx_http_server_build_string[]

= **"Server: tom" CRLF;**

停止服务,安装目录内重新编译检查 编译 安装,并重启服务,检查端口

svr72 logs]# /usr/local/nginx/sbin/nginx -s stop

svr72 logs]# /usr/local/nginx/sbin/nginx -v

nginx version: nginx/1.12.2

svr72 ~]# /usr/local/nginx/sbin/nginx -V

nginx version: nginx/1.12.2

svr72 nginx-1.12.2]#./configure --without-http_autoindex_module

&& make && make install

vr72 ~]# /usr/local/nginx/sbin/nginx

vr72 ~]# ss -antulp | grep :80

**4.3** 限制并发

**4.3.1** 真机检查 **ab** 是否存在

room9pc01 ~]# rpm -qf /bin/ab

httpd-tools-2.4.6-80.el7.centos.x86_64

**4.3.2 真机向 192.168.4.72 进行 ab 测试**

```
room9pc01 ~]# ab -c 100 -n 100 http//192.168.4.72/  #测试全部成功
Complete requests:     100
Failed requests:       0
```

ngx_http_limit_req_module 模块可以降低 DDos 攻击风险

定义一块内存区域给 nginx 使用,存客户端访问 nginx 时的 IP 地址

语法: **limit_req_zone key zone=name:size rate=rate;**

将客户端 IP 信息存储名称为 one 的共享内存,空间 10M

1M 可以存储 8 千个 IP 的信息,10M 存 8 万个主机状态

相同 IP 的请求,1 秒内只处理一个,多余的放入漏斗

漏斗超过 5 个则报错

**4.3.3 停止服务,修改配置文件,并启动服务**

```
svr72 ~]# /usr/local/nginx/sbin/nginx -s stop    #停止服务
svr72 ~]# vim /usr/local/nginx/conf/nginx.conf
20 limit_req_zone $binary_remote_addr zone=one:10m rate=1r/s;
```

20 行添加此条语句

```
38 limit_req zone=one burst=5;    #38 行添加此语句
svr72 ~]# /usr/local/nginx/sbin/nginx    #启动服务
```

**4.3.4 真机向 192.168.4.72 进行 ab 测试,此时测试会有大多数失败**

```
room9pc01 ~]# ab -c 100 -n 100 http//192.168.4.72/
Complete requests:     100
Failed requests:       94    #失败了 94 次
```

**4.4 拒绝非法请求**

```
svr72 ~]# curl -i -X GET http://192.168.4.72/test.html
```

返回网页内容　web72

```
svr72 ~]# curl -i -X HEAD http://192.168.4.72/test.html
```

HEAD 方式为非法访问,仅访问头部信息,不返回文件内容,且一直占用 1 个终端

**4.4.1 停止服务,修改配置文件,并重启服务,检查端口**

```
svr72 ~]# /usr/local/nginx/sbin/nginx -s stop
svr72 ~]# vim /usr/local/nginx/conf/nginx.conf
 35    server {
 36        listen      80;  #手动添加以下 3 行
 37        if ($request_method !~ ^(GET|POST)){
 38              return 444;
 39        }
svr72 ~]# /usr/local/nginx/sbin/nginx   #重启服务
svr72 ~]# ss -antulp | grep :80
```

**4.4.2 检测 GET HEAD 方式访问**

```
svr72 ~]# curl -i -X GET http://192.168.4.72/test.html
  #GET 方式仍能访问,且返回页面内容    web72
svr72 ~]# curl -i -X HEAD http://192.168.4.72/test.html
curl: (52) Empty reply from server   #无访问回应
```

**4.5 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

```
svr72 ~]# vim /usr/local/nginx/conf/nginx.conf
```

在 http { 行下添加以下 4 行

```
client_body_buffer_size  1K;
client_header_buffer_size 1k;
client_max_body_size 1k;
large_client_header_buffers 2 1k;
```

重启服务即可

**五 服务安全-数据库安全(mariadb)**

```
svr72 ~]# yum -y install mariadb-server mariadb
```

mariadb-server 提供数据库服务  mariadb 提供命令

```
svr72 ~]# systemctl start mariadb
```

**5.1** 初始化安全脚本

```
svr72 ~]# mysql_secure_installation
Enter current password for root (enter for none):  #无
Set root password? [Y/n] Y
New password:                                  #两次输入新密码
Re-enter new password:
Remove anonymous users? [Y/n] Y          #删除匿名用户? Y
Disallow root login remotely? [Y/n] Y  #禁用 root 远程登录? Y
Remove test database and access to it? [Y/n] Y
                                                #删除 test 数据库? Y
Reload privilege tables now? [Y/n] Y    #刷新权限? Y
```

**5.2** 密码安全**(3** 种修改密码的方法**)**

**5.2.1** 命令行输入旧密码**,**配置 **root** 新密码

```
svr72 ~]# mysqladmin -hlocalhost -uroot -p123456 password "654321"
```

**5.2.2** 数据库内用语句修改

```
MariaDB [(none)]> set password for
root@"localhost"=password("redhat");
```

**5.2.3** 数据库内修改 **mysql.user** 表中 **password** 字段的值

```
MariaDB [(none)]> select user,host,password from mysql.user;
update mysql.user set password=password(“tarena”)
where host=”localhost” and user=”root”;
```

### 5.2.4 刷新授权

```
flush privileges;
```

管理员家目录下面在每次操作数据库后,会生成 binlog 日志,内含明文密码

每次操作数据库后,删除 binlog 日志

```
rm -rf ~/.mysql_history    #内含登录用户及密码及操作
rm -rf ~/.bash_history     #内含命令行历史命令
```

## 5.3 数据备份与还原

```
svr72 ~]# mysqldump -uroot -p密码 库 表 > table.sql  #备份表
svr72 ~]# mysqldump -uroot -p密码 库 > db.sql  #备份库
svr72 ~]# mysqldump -uroot -p密码 -A > all.sql  #备份所有
```

-A 等同于 --all-databases

```
svr72 ~]# mysql -uroot -p密码 mydb < table.sql  #还原表
svr72 ~]# mysql -uroot -p密码 mydb < db.sql  #还原库
svr72 ~]# mysql -uroot -p密码 < all.sql  #还原所有
```

**5.4** 数据安全

**5.4.1** 创建可以远程登录的用户,设置该用户的访问权限

```
MariaDB [(none)]> grant all on gamedb.* to tom@"%" identified by
"123qqq";
```

#设置用户 tom 从任何主机上使用密码”123qqq”登录数据库,只能有 gamedb 库的所有权限

**5.4.2 72** 上使用 **tcpdump** 抓包

```
svr72 ~]# tcpdump -w log -i eth0 src or dst port 3306
```

**5.4.3** 客户端远程登录数据库,查看抓包数据

```
srv7 ~]# mysql -h192.168.4.72 -utom -p123qqq
```

**5.4.4 72** 上查看抓包日志

```
svr72 ~]# tcpdump -A -r log
```

5.4.4 解决:网站上使用 SSL 或 SSH 加密数据传输,配置 https 协议

六 服务安全**-tomcat** 安全

**6.1** 部署 **tomcat**

```
svr72 ~]# yum list | grep jdk
svr72 ~]# yum -y install java-1.8.0-openjdk
svr72 ~]# which java
/usr/bin/java
svr72 ~]# java -version
openjdk version "1.8.0_161"
OpenJDK Runtime Environment (build 1.8.0_161-b14)
OpenJDK 64-Bit Server VM (build 25.161-b14, mixed mode)

room9pc01 ~]# scp /linux-soft/02/lnmp_soft.tar.gz
root@192.168.4.72:/root
svr72 ~]# tar -xf lnmp_soft.tar.gz
svr72 ~]# cd lnmp_soft/
svr72 lnmp_soft]# tar -xf apache-tomcat-8.0.30.tar.gz
svr72 lnmp_soft]# mv apache-tomcat-8.0.30/
/usr/local/tomcat

svr72 ~]# echo "abc" >
/usr/local/tomcat/webapps/ROOT/test.html
svr72 ~]# /usr/local/tomcat/bin/startup.sh  #启动服务
```

```
svr72 lnmp_soft]# ss -antulp | grep :8080


svr72 ~]# /usr/local/tomcat/bin/shutdown.sh
svr72 ~]# ss -antulp | grep :8080


svr72 ~]# /usr/local/tomcat/bin/startup.sh
svr72 ~]# curl http://localhost:8080/test.html
abc
```

**6.2** 隐藏软件版本

**6.2.1 3** 种方式访问 **tomcat** 网站,会返回信息及软件版本

```
svr72 ~]# curl -I http://192.168.4.72:8080/test.html  #头部信息
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
svr72 ~]# curl -I http://192.168.4.72:8080/    #头部信息
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
vr72 ~]# curl -I http://192.168.4.72:8080/test2.html  #报错页面
HTTP/1.1 404 Not Found
Server: Apache-Coyote/1.1
```

### 6.2.2 修改 tomcat 配置文件

```
svr72 ~]# /usr/local/tomcat/bin/shutdown.sh
svr72 ~]# yum -y install java-1.8.0-openjdk-devel  #jar 解压软件
svr72 ~]# cd /usr/local/tomcat/lib
svr72 lib]# jar -xf catalina.jar
svr72 ~]# cd /usr/local/tomcat/lib/org/apache/catalina/util
svr72 util]# vim ServerInfo.properties
16 server.info=Nginx/9.0.30^M
17 server.number=9.0.30^M
svr72 ~]# vim /usr/local/tomcat/conf/server.xml
 69    <Connector port="8080" protocol="HTTP/1.1"
 70               connectionTimeout="20000"
 71               redirectPort="8443" server="IIS" />
```

### 6.2.3 测试

```
svr72 ~]# /usr/local/tomcat/bin/startup.sh   #启动服务
```

浏览器访问不存在的网页:http://192.168.4.72:8080/abc

Nginx/9.0.30

命令行访问网站头部:

```
svr72 ~]# curl -I http://192.168.4.72:8080
Server: IIS
```

命令行访问存在的网页:

```
svr72 ~]# curl -I http://192.168.4.72:8080/test.html
Server: IIS
```

命令行访问不存在的网页:

```
svr72 ~]# curl -I http://192.168.4.72:8080/abcdefg.html
Server: IIS
```

**6.3 降权启动**

**6.3.1 未降权前,Java 进程的用户为 root**

```
svr72 ~]# ps aux | grep java    #显示用户是 root
```

**6.3.2 停止 tomcat 服务**

```
svr72 ~]# /usr/local/tomcat/bin/shutdown.sh
svr72 ~]# ss -antulp | grep :8080    #确认 tomcat 服务关闭
```

**6.3.3 添加普通用户,并修改 tomcat 目录的归属为新用户**

```
svr72 ~]# useradd tomcat
svr72 ~]# chown -R tomcat:tomcat /usr/local/tomcat/
```

**6.3.5 启动服务,并查看进程用户**

```
svr72 ~]# su - tomcat -c "/usr/local/tomcat/bin/startup.sh"
```

```
svr72 ~]# ps aux | grep java  #此时显示 tomcat 的用户为 tomcat
```

**6.3.6 删除默认的测试页面,重新配置默认页面**

```
svr72 ~]# rm -rf /usr/local/tomcat/webapps/*
```

## 七 Linux 安全之打补丁-补丁的原理

### 7.1 源代码的不同版本


### 7.2 diff 逐行比较

#### 7.2.1 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

#### 7.2.2 选项

**-u** 输出统一内容的头部信息(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,默认会提示 txt 仅在 B 目录有(无法根据补丁修改 A 缺的文件)

#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!

**7.3 diff 文件对比**

```
srv7 ~]# cat test1.sh
#!/bin/bash
echo "hello word"

srv7 ~]# cat test2.sh
#!/bin/bash
echo "hello world"
echo "i love china"

srv7 ~]# diff -u test1.sh test2.sh    #生成.patch 文件
--- test1.sh    2019-09-24 17:44:26.361054398 +0800
+++ test2.sh    2019-09-24 17:45:20.153054398 +0800
@@ -1,2 +1,3 @@
```

```
 #!/bin/bash
-echo "hello word"
+echo "hello world"
+echo "i love china"
```

### 7.4 diff 目录对比

#### 7.4.1 环境准备

```
srv7 ~]# mkdir demo
srv7 ~]# cd demo/
srv7 demo]# mkdir source1 source2
srv7 demo]# echo "hello world" > source1/test.sh
srv7 demo]# echo "hello the world" > source2/test.sh
srv7 demo]# cp /usr/bin/find source1/
srv7 demo]# cp /usr/bin/find source2/
srv7 demo]# echo "1" >> source2/find
srv7 ~]# echo abc /demo/source2/tmp.txt

srv7 ~]# scp -r /demo/source1 root@192.168.4.8:/demo/
```

**7.4.2** 对比差异

srv7 ~]# **diff -u** /demo/source1 /demo/source2

#仅对比了文本文件 test.sh,二进制文件\tmp 文件都没有对比差异,仅提示

srv7 ~]# **diff -Nu** /demo/source1 /demo/source2

#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

srv7 ~]# **diff -Naur** /demo/source1 /demo/source2

#对比了 test.sh tmp.txt find(程序)

八 **Linux** 安全之打补丁**-patch** 打补丁

**命令格式: patch -p** 数字 **<** 补丁文件名 **(**数字表示删除重复目录的个数**)**

**重复目录的个数:** 补丁文件里的路径和被打补丁文件的路径比较

**撤销打补丁: patch -RE <** 补丁文件名 进入撤销补丁文件目录下执行

**8.1** 给文件生成补丁

srv7 ~]# diff -u test1.sh test2.sh > bd.patch

**8.2** 给文件打补丁

**8.3** 给目录生成补丁

srv7 ~]# diff -Nuar /demo/source1 /demo/source2 > /root/cy.patch

srv7 ~]# scp /root/cy.patch root@192.168.4.8:/root

**8.4** 给目录打补丁

```
svr8 ~]# cd /demo/source1/  # / demo source1 分别为重复目录

svr8 source1]# patch -p3 < /root/cy.patch
```