Security-03

服务安全与监控

# 系统审计

## 概述

### 审计的介绍：

基于事先的规则生成日志，记录可能发生在系统上的事件

审计不会为体通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、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，可以生成是及报告

### 部署audit

使用审计系统需要安装audit软件包

主配置文件为: /etc/audit/auditd.conf

[root@mysql51 ~]# rpm -q audit(**默认是安装的**)

audit-2.7.6-3.el7.x86\_64

[root@mysql51 ~]# cat /etc/audit/auditd.conf

log\_file = /var/log/audit/audit.log //日志文件

### 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

[root@mysql51 ~]# auditctl -s //查询状态

[root@mysql51 ~]# auditctl -l //查看规则

[root@mysql51 ~]# 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

[root@mysql51 ~]# auditctl -w /etc/passwd -p rwa -k mon\_passwd

//设置规则所有对passwd文件的读写、属性修改操作都会被记录审计日志

[root@mysql51 ~]# auditctl -w /etc/ -p rwx -k mon\_etc

//设置规则，监控/etc/目录

[root@mysql51 ~]# auditctl -w /usr/bin/rm -p x -k mon\_rm

//设置规则，监控rm程序

## 定义永久规则

写入配置文件/etc/audit/rules.d/audit.rules

[root@mysql51 ~]# vim /etc/audit/rules.d/audit.rules

-w /etc/passwd -p rwa -k mon\_passwd

-w /etc/ -p rwx -k mon\_etc

-w /usr/bin/rm -p x -k mon\_rm

## 赋加扩展

### 通过审核也可以监控系统调用

[root@mysql51 ~]# cat /usr/include/asm/unistd\_64.h

### 规则参考模版

[root@mysql51 ~]# ls /usr/share/doc/audit-2.7.6/

## 查看日志

type为类型

- SYSCALL //系统调用

msg为（time\_stamp:ID），时间是date + %s

arch=c0000003e,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进制编码了

ppid父进程ID，如bash,pid进程ID，如cat命令

auid是审核用户的id,su - test,依然可以追踪su 前的账户

uid，gid用户与组

tty:从哪个终端执行的命令

common=”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 //用来记录当前工作目录

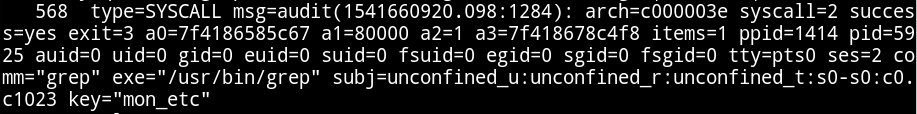
- cwd=”/home/username”

type=PATH

- ouid(owner’s user id) //对象所有者id

- guid(owner’s groupid) //对象所有者id

[root@mysql51 ~]# grep "mon\_etc" /var/log/audit/audit.log | cat -b | tail -1



### 搜索日志

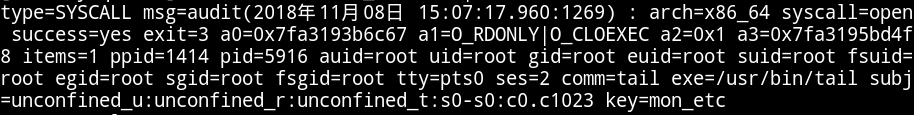
系统提供的ausearch命令可以方便的搜索特定日志

- 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

根据key搜索日志，-i 为交互式操作

[root@mysql51 ~]# ausearch -k mon\_etc -i | tail -1



# 服务安全

## Nginx安全

### Nginx是模块化设计

- 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

--without-http\_autoindex\_module //自动索引功能

自动索引：

访问的网站根目录下没由网站默认页面（index.html）时自动列出所有文件

--without-http\_ssi\_module //

最小化安装永远是对的！！！

[root@client50 ~]# tar -xf nginx-1.12.2.tar.gz

[root@client50 ~]# cd nginx-1.12.2/

[root@client50 nginx-1.12.2]# yum -y install pcre-devel zlib-devel

[root@client50 nginx-1.12.2]# ./configure --prefix=/usr/local/nginx

[root@client50 nginx-1.12.2]# make && make install

[root@client50 nginx-1.12.2]# /usr/local/nginx/sbin/nginx

[root@client50 nginx-1.12.2]# ss -natpul | grep :80

**测试索引功能**

[root@client50 nginx-1.12.2]# vim /usr/local/nginx/conf/nginx.conf

server {

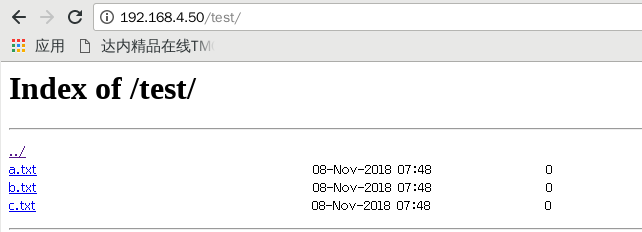
autoindex on; //开启模块自动索引功能

listen 80;

[root@client50 nginx-1.12.2]# /usr/local/nginx/sbin/nginx -s reload

[root@client50 nginx-1.12.2]# mkdir -p /usr/local/nginx/html/test

[root@client50 nginx-1.12.2]# touch /usr/local/nginx/html/test/{a,b,c}.txt



### 修改版本信息

如何修改版本信息(修改源码)

如何隐藏版本号信息(server\_tokens off)

查看版本信息：

[root@client50 nginx-1.12.2]# /usr/local/nginx/sbin/nginx -V

HTTP/1.1 200 OK

Server: nginx/1.12.2

**开启隐藏版本功能**

[root@client50 nginx-1.12.2]# vim /usr/local/nginx/conf/nginx.conf

http {

include mime.types;

default\_type application/octet-stream;

server\_tokens off; //隐藏软件版本

[root@client50 nginx-1.12.2]# /usr/local/nginx/sbin/nginx -s reload

[root@client50 nginx-1.12.2]# curl -I http://192.168.4.50

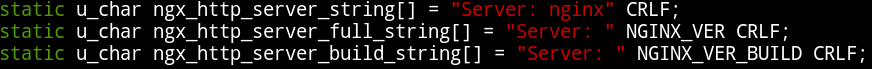
HTTP/1.1 200 OK

Server: nginx

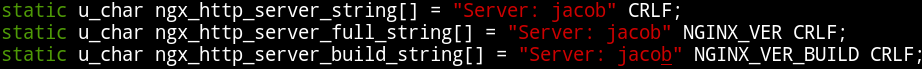
**进入源码目录进行修改：**

nginx-1.12.2]# vim +48 src/http/ngx\_http\_header\_filter\_module.c

修改前：



修改后：



**重新编译安装，进行测试：**

[root@client50 nginx-1.12.2]# ./configure --prefix=/usr/local/nginx

[root@client50 nginx-1.12.2]# make && make install

[root@client50 nginx-1.12.2]# killall nginx

[root@client50 nginx-1.12.2]# /usr/local/nginx/sbin/nginx

[root@client50 nginx-1.12.2]# curl -I http://192.168.4.50

HTTP/1.1 200 OK

Server: jacob

### 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

下面配置的功能为：

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，空间为10M

1M可以存储8千个请求，多于的放入漏斗

每秒中仅接受1个请求，多余的放入漏斗

漏斗超过5个则报错

[root@client50 nginx-1.12.2]# vim /usr/local/nginx/conf/nginx.conf

http {

.....

limit\_req\_zone $binary\_remote\_addr zone=**one**:10m rate=1r/s;

server {

.....

limit\_req zone=**one** burst=5;

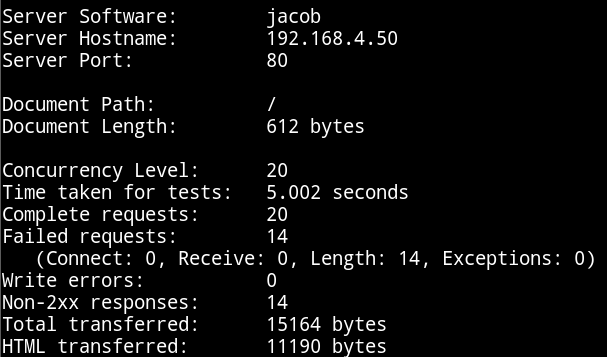
}

}

[root@client50 nginx-1.12.2]# /usr/local/nginx/sbin/nginx -s reload

[root@client50 nginx-1.12.2]# ab -c 20 -n 20 http://192.168.4.50/

//发起20次请求，14次请求失败



### 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，并返回实体主题 |
| HEAD | 类似与get请求，只不过返回的响应种没有具体的内容，用于获取报头 |
| POST | 向指定资源提供数据进行处理请求（例如提交表或者上传文件） |
| DELETE | 请求服务器删除指定的页面 |
| PUT | 向服务器特定位置上船资料 |
| ... | 其他 |

[root@client50 nginx-1.12.2]# vim /usr/local/nginx/conf/nginx.conf

http{

server {

listen 80;

server\_name localhost;

if ($request\_method !~ ^(GET|POST)$ ) {

内置变量 return 500;

}

}

}

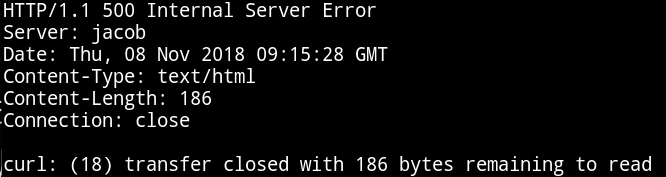
[root@client50 nginx-1.12.2]# /usr/local/nginx/sbin/nginx -s reload

[root@room9pc01 ~]# curl -i -X GET <http://192.168.4.50/>

屏幕输出中默认网页：

<title>Welcome to nginx!</title>

[root@room9pc01 ~]# curl -i -X HEAD http://192.168.4.50/



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

缓冲区溢出会导致数据泄露

[root@client50 ~]# vim /usr/local/nginx/conf/nginx.conf

http {

......

client\_body\_buffer\_size 1k;

client\_header\_buffer\_size 1k;

client\_max\_body\_size 16k;

large\_client\_header\_buffers 4 4k;

......

}

## 数据库安全

初始化安全脚本

输入旧密码，配置新root密码

Remove anonymous users（删除匿名账户）

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）

[root@node1 ~]# mysql\_secure\_installation //执行初始化安全脚本

密码安全

修改MySQL密码的若干方法

[root@node1 ~]# mysqladmin -uroot -p123456 password 'mysql'

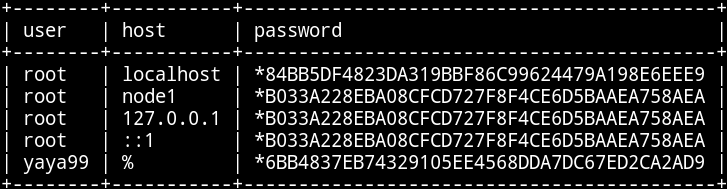
//修改密码，旧密码为123456，新密码为mysql

[root@node1 ~]# mysql -uroot -pmysql

MariaDB [(none)]> set password for root@'localhost'=password('redhat')

//修改账户登录数据库，修改密码

MariaDB [(none)]> select user,host,password from mysql.user;



问题是i历史记录会出卖你！

binlog日志里有明文密码(5.6版本后修复了)

[root@node1 ~]# cat .bash\_history

.......

mysqladmin -uroot -p123asd...A password 'mysql'

[root@node1 ~]# cat .mysql\_history

show databases;

grant all on gamedb.\* to yaya99"%" identified by "123456";

set password for root@'localhost'=password('redhat');

.......

[root@node1 ~]# > .mysql\_history //清空有关数据库的历史命令

[root@node1 ~]# > .bash\_history

解决：

管理好自己的历史，不使用明文登录，选择合适的版本

日志，行为审计

防火墙从TCP层设置ACL（禁止外网接触数据库）

数据备份与还原

备份

还原

数据安全

创建可以远程登录的账户

[root@node1 ~]# mysql -uroot -predhat

MariaDB [(none)]> grant all on \*.\* to tom@'%' identified by '123';

使用tcpdump抓包

[root@node1 ~]# tcpdump -w log -i eth1 src or dst port 3306

客户端远程登录数据库，查看抓包数据

[root@room9pc01 ~]# mysql -utom -h192.168.2.5 -p123

[root@node1 ~]# tcpdump -r log

解决：使用SSL或SSH加密数据传输

## Tomcat安全

### 安装Tomcat

[root@client50 ~]# tar -xf apache-tomcat-8.0.30.tar.gz -C /usr/local/

[root@client50 ~]# cd /usr/local/

[root@client50 local]# mv apache-tomcat-8.0.30/ tomcat

[root@client50~]# echo 123 > /usr/local/tomcat/webapps/ROOT/test.html

[root@client50 local]# /usr/local/tomcat/bin/startup.sh

[root@client50 local]# ss -natpul | grep :8080

### 隐藏版本信息

修改tomcat主配置文件，隐藏版本信息

[root@client50 lib]# yum -y install java-1.8.0-openjdk-devel

[root@client50 ~]# cd /usr/local/tomcat/

[root@client50 tomcat]# cd lib/

[root@client50 lib]# jar -xf catalina.jar //jar解java包工具

[root@client50 lib]# ls

META-INF org //解压之后会在当前目录下多出两个目录

**修改服务器的信息文件：**

[root@client50 lib]# vim org/apache/catalina/util/ServerInfo.properties

//根据自己的需要，修改版本信息的内容

server.info=nginx nginx/1.2.9

server.number=8.1.0.0

server.built=Dec 1 2015 22:30:46 UTC

**修改Tomcat主配置文件**

[root@client50 ~]# vim /usr/local/tomcat/conf/server.xml

<Connector port="8080" protocol="HTTP/1.1"

connectionTimeout="20000"

redirectPort="8443" server="nginx" />

[root@client50 lib]# /usr/local/tomcat/bin/shutdown.sh //关闭服务

[root@client50 lib]# /usr/local/tomcat/bin/startup.sh //启动服务

[root@client50 lib]# ss -natpul | grep 8080

测试

[root@room9pc01 ~]# curl -I <http://192.168.4.50:8080>

//访问存在的页面文件，查看头部信息

[root@room9pc01 ~]# curl -I <http://192.168.4.50:8080/xxx>

//访问不存在的页面文件，查看头部信息

[root@room9pc01 ~]# curl <http://192.168.4.50:8080/xxx>

//访问不存在的页面文件，查看报错页面

### 降权启动

使用非root启动tomcat服务

[root@client50 lib]# /usr/local/tomcat/bin/shutdown.sh

[root@client50 lib]# useradd tomcat

[root@client50 lib]# chown -R tomcat:tomcat /usr/local/tomcat/

//修改tomcat目录的权限，让tomcat账户对改目录有操作权限

[root@client50 lib]# su -c /usr/local/tomcat/bin/startup.sh tomcat

//使用su 命令切换为tomcat账户，以tomcat账户的身份启动tomcat服务

### 开机启动

[root@client50 lib]# chmod +x /etc/rc.local //该文件为开机启动文件

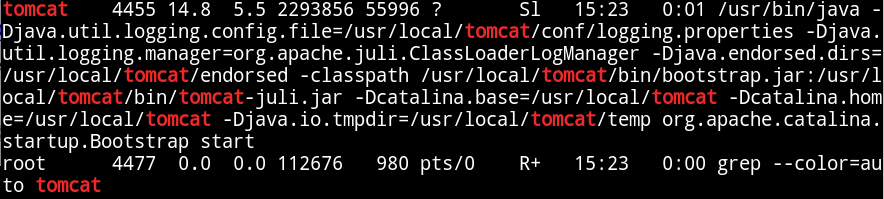
[root@client50 lib]# vim /etc/rc.local //添加如下内容

su -c /usr/local/tomcat/bin/startup.sh tomcat

[root@client50 lib]# chmod +x /etc/rc.d/rc.local

**注释：在安装nginx服务时,直接指定执行服务的用户和组，那么，无论是谁启动服务，最终服务都属于此用户**

[root@client50 lib]# ps aux | grep tomcat



### 删除软件默认的测试页面,书写自己攻击的首页文件

[root@client50 ~]# rm -rf /usr/local/tomcat/webapps/\*

# Linux安全之打补丁

## 补丁的原理

### diff逐行比较（diff生成补丁包）

### duff的原则是：**（显示差异）**

- 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 输出统一内容的头部信息（打补丁使用），计算机知道哪个文件需要修改

-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选项注释：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

[root@client50 ~]# mkdir /damo

[root@client50 ~]# cd /damo

[root@client50 damo]# vim test1.sh

#!/bib/bash

echo "hello world"

echo "test"

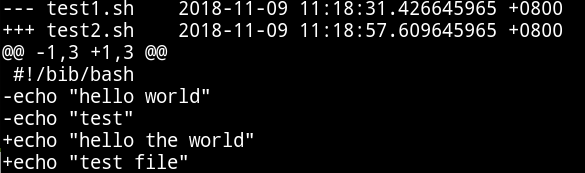
[root@client50 damo]# vim test2.sh

#!/bib/bash

echo "hello the world"

echo "test file"

[root@client50 damo]# diff -u test1.sh test2.s



注释：第一个文件删除前两行后，再把+后面的内容添加上，即可变成第二个文件！

## patch打补丁

### 给文件生成补丁

准备环境，生成补丁文件

[root@client50 ~]# ls /damo/

test1.sh test2.sh

[root@client50 ~]# cd /damo/

[root@client50 damo]# diff -u test1.sh test2.sh > test.patch

[root@client50 damo]# ls

test1.sh test2.sh test.patch //补丁包文件

### 给文件打补丁

对旧版本的代码，使用补丁即可更新，而不需要下载完整的整个代码（因完整的程序会很大）

[root@client50 damo]# yum -y install patch

[root@client50 damo]# patch -p0 < test.patch //对当前版本进行更新

patching file test1.sh

[root@client50 damo]# patch -RE < test.patch //恢复到更新之前的旧版本

patching file test1.sh

### 对比目录中所有文件的差异

准备实验素材：

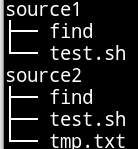
[root@client50 damo]# mkdir source1 source2

[root@client50 damo]# echo "hello world" > source1/test.sh

[root@client50 damo]# cp /bin/find source1/

[root@client50 damo]# echo "hello the world" > source2/test.sh

[root@client50 damo]# echo "test" > source2/tmp.txt

[root@client50 damo]# cp /bin/find source2/

[root@client50 damo]# echo "1" >> source2/find

[root@client50 damo]# yum -y install tree

[root@client50 damo]# tree source1 source2

注释：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

**制作补丁包各个选项的区别**

[root@client50 damo]# diff -u source1 source2

//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

[root@client50 damo]# diff -Nu source1 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

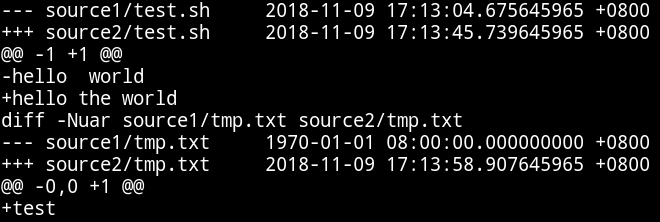
[root@client50 damo]# diff -Nua source1 source2

//对比了test.sh、tmp.txt、find（程序）

**黄金组合：**

[root@client50 damo]# diff -Nuar source1 source2 > source.path

[root@client50 source1]# tail ../source.path



**更新源程序：**

[root@client50 source1]# patch -p1 < ../source.path

patching file find

patching file test.sh

patching file tmp.txt

**对比二进制文件是否一致：**

[root@client50 source1]# md5sum find

29b6030ce0d35abd03d089ce0c6f0b18 find

[root@client50 source1]# md5sum ../source2/find

29b6030ce0d35abd03d089ce0c6f0b18 ../source2/find

**注释：**

//patch -pnum（其中num为数字，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/blurfl/blurfl.c

//-p0则整个路径不变

//-p1则修改路径为u/howard/src/blurfl/blurfl.c

//-p4则修改路径为blurfl/blurfl.c

//-R(reverse)反向修复，-E修复后如果文件为空，则删除该文件