openEuler内核编程技术

实训指导书

第十章 第1讲

《内核审计机制》

软件所制

**术语表**

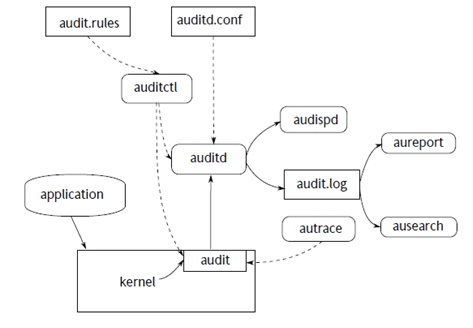
|  |  |  |  |
| --- | --- | --- | --- |
| 英文 | 中文 | 含义 | 别名 |
|  |  |  |  |
|  |  |  |  |

第十章 第1讲 内核审计机制

**任务1：安装与启动audit服务，完成auditd的配置和规则(30min)**

* 1. **相关知识**

Audit是系统中的一个服务进程，它是Linux Audit Subsystem（Linux审计子系统）的缩写。Audit在审计系统中用于调用记录，并把记录写入文件中。Audit的总体架构图如下：



我们可以看到，Audit是内核中的一个模块，内核的运行情况都会在Audit中记录，当然，这个记录的规则是由root来设置的。（注：Audit只有root才有权限操作）

事实上，内核的Audit模块是由应用层的一个应用程序auditd来控制的。Audit产生的数据都会传送到auditd中，然后再由auditd进行其它操作。

auditd.conf是auditd的配置文件，确定auditd是如何启动的，日志文件放在哪里等等，后面会有详细说明。

audit.rules是Audit的规则文件，确定Audit的日志中记录哪些操作。它通过一个对Audit进行控制的应用程序auditctl进行操作。当然，root也可以直接调用auditctl进行操作。

audit安装后在/etc/audit/目录下会有相关的配置文件：

* /etc/audit/auditd.conf 守护程序的默认配置文件
* /etc/audit/audit.rules 守护程序的默认规则文件

auditd配置文件文件详解

|  |  |
| --- | --- |
| log\_file | 审计日志文件的完整路径。如果您配置守护进程向除默认/var/log/audit/外的目录中写日志文件时，一定要修改它上面的文件权限，使得只有根用户有读、写和执行权限。所有其他用户都不能访问这个目录或这个目录中的日志文件 |
| log\_format | 写日志时要使用的格式。当设置为RAW时，数据会以从内核中检索到的格式写到日志文件中。当设置为NOLOG时，数据不会写到日志文件中，但是如果用dispatcher选项指定了一个，则数据仍然会发送到审计事件调度程序中 |
| priority\_boost | 审计应采用多少优先级推进守护进程。必须是非负数。0表示没有变化 |
| Flush | 多长时间向日志文件中写一次数据。值可以是NONE、INCREMENTAL、DATA和SYNC之一 |
| Freq | 如果flush设置为INCREMETNAL，审计守护进程在写到日志文件中前从内核中接收的记录数 |
| num\_logs | max\_log\_file\_action设置为ROTATE时要保存的日志文件数目。必须是0~99之间的数。如果设置为小于2，则不会循环日志。如果递增了日志文件的数目，就可能有必要递增/etc/audit/audit.rules中的内核backlog设置值，以便留出日志循环的时间。如果没有设置num\_logs值，它就默认为0，意味着从来不循环日志文件 |
| dispatcher | 当启动这个守护进程时，由审计守护进程自动启动程序。所有守护进程都传递给这个程序。可以用它来进一步定制报表或者以与您的自定义分析程序兼容的不同格式产生它们。由于调度程序用根用户特权运行，因此使用这个选项时要极其小心 |
| disp\_qos | 控制调度程序与审计守护进程之间的通信类型。有效值为lossy和lossless。如果设置为lossy，若审计守护进程与调度程序之间的缓冲区已满(缓冲区为128千字节)，则发送给调度程序的引入事件会被丢弃。然而，只要log\_format没有设置为nolog，事件就仍然会写到磁盘中。如果设置为lossless，则在向调度程序发送事件之前和将日志写到磁盘之前，调度程序会等待缓冲区有足够的空间 |
| max\_log\_file | 以兆字节表示的最大日志文件容量。当达到这个容量时，会执行max\_log\_file \_action指定的动作 |
| max\_log\_file\_action | 当达到max\_log\_file的日志文件大小时采取的动作。值必须是IGNORE、SYSLOG、SUSPEND、ROTATE和KEEP\_LOGS之一。如果设置为IGNORE，则在日志文件达到max\_log\_file后不采取动作。如果设置为SYSLOG，则当达到文件容量时会向系统日志/var/log/messages中写入一条警告。如果设置为SUSPEND，则当达到文件容量后不会向日志文件写入审计消息。如果设置为ROTATE，则当达到指定文件容量后会循环日志文件，但是只会保存一定数目的老文件，这个数目由num\_logs参数指定。老文件的文件名将为audit.log.N，其中N是一个数字。这个数字越大，则文件越老。如果设置为KEEP\_LOGS，则会循环日志文件，但是会忽略num\_logs参数，因此不会删除日志文件 |
| space\_left | 以兆字节表示的磁盘空间数量。当达到这个水平时，会采取space\_left\_action参数中的动作 |
| space\_left\_action | 当磁盘空间量达到space\_left中的值时，采取这个动作。有效值为IGNORE、SYSLOG、EMAIL、SUSPEND、SINGLE和HALT。如果设置为IGNORE，则不采取动作。如果设置为SYSLOG，则向系统日志/var/log/messages写一条警告消息。如果设置为EMAIL，则从action\_mail\_acct向这个地址发送一封电子邮件，并向/var/log/messages中写一条警告消息。如果设置为SUSPEND，则不再向审计日志文件中写警告消息。如果设置为SINGLE，则系统将在单用户模式下。如果设置为SALT，则系统会关闭 |
| action\_mail\_acct | 负责维护审计守护进程和日志的管理员的电子邮件地址。如果地址没有主机名，则假定主机名为本地地址，比如root。必须安装sendmail并配置为向指定电子邮件地址发送电子邮件 |
| admin\_space\_left | 以兆字节表示的磁盘空间数量。用这个选项设置比space\_left\_action更多的主动性动作，以防万一space\_left\_action没有让管理员释放任何磁盘空间。这个值应小于space\_left\_action。如果达到这个水平，则会采取admin\_space\_left\_ action所指定的动作 |
| admin\_space\_left\_action | 当自由磁盘空间量达到admin\_space\_left指定的值时，则采取动作。有效值为IGNORE、SYSLOG、EMAIL、SUSPEND、SINGLE和HALT |
| disk\_full\_action | 如果含有这个审计文件的分区已满，则采取这个动作。可能值为IGNORE、SYSLOG、SUSPEND、SINGLE和HALT |
| disk\_error\_action | 如果在写审计日志或循环日志文件时检测到错误时采取的动作。值必须是IGNORE、SYSLOG、SUSPEND、SINGLE和HALT之一 |

* 1. **任务描述**

1、安装audit并开启服务

2、查看并了解auditd的配置文件和规则文件

* 1. **审核要求**

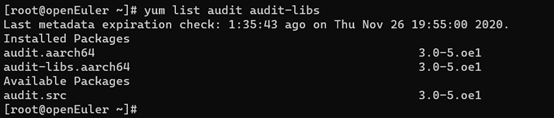
1. 正确编写满足功能的源文件，正确编译。
2. 正常加载、卸载内核模块；且内核模块功能满足任务所述。
3. 提交相关源码与运行截图。
   1. **参考答案**

一、安装audit并开启服务

Audit系统使用如下包:audit和audit-libs。Openeuler上已经默认安装。可以通过如下命令验证系统中是否已经安装好：

dnf list audit audit-libs

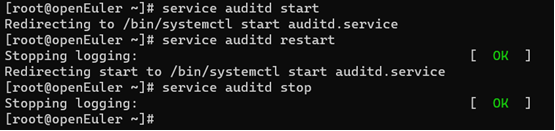
您应该在输出中看到Installed Packages下的几个包：



在Openeuler中，可以手动控制服务的启动和关闭，命令如下：

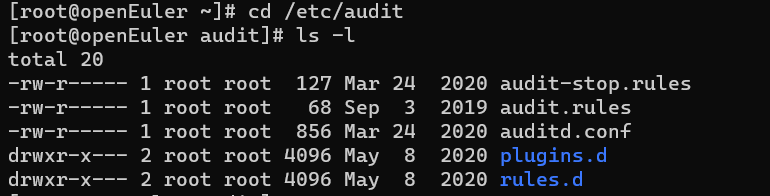
* 启动XXX服务：service XXX start
* 关闭XXX服务：service XXX stop
* 重启XXX服务：service XXX restart
* 重载XXX服务：service XXX reload

启动auditd进程的过程如下：

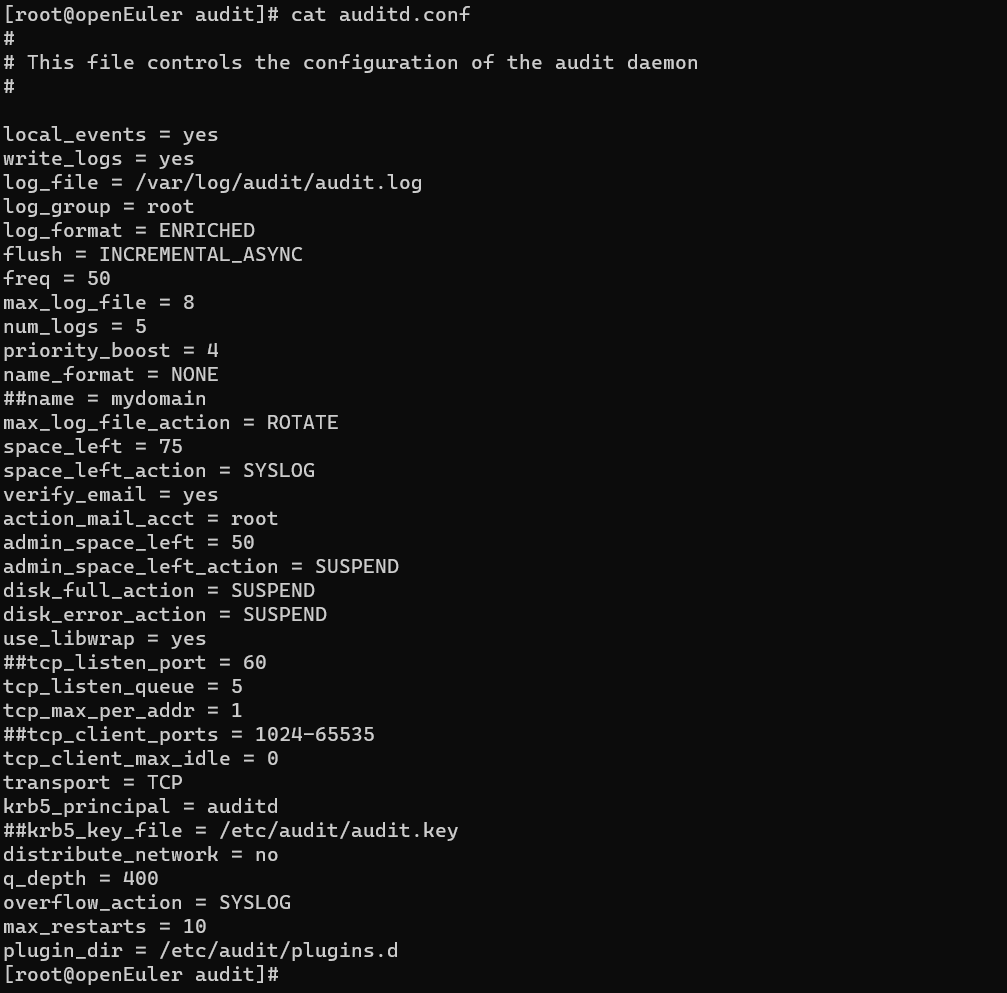


二、查看并了解auditd的配置文件和规则文件

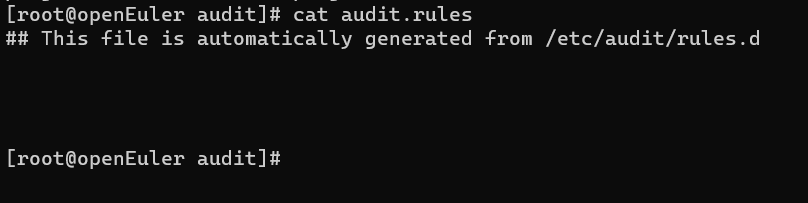
切换进/etc/audit目录，查看目录下的文件信息



查看auditd的配置文件



查看auditd的规则



audit.rules初始时没有具体规则，添加了规则后文件内容会发生变化。

**任务2：使用auditctl命令控制audit(30min)**

**2.1 相关知识**

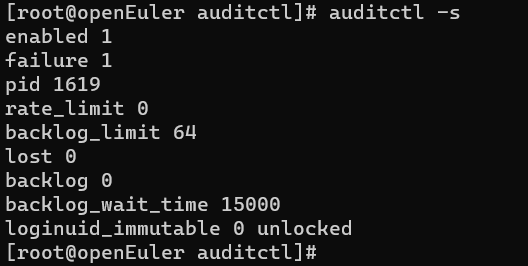
auditctl 命令允许您控制审核系统的基本功能并且限定规则来决定哪些审核项目要记录。所有与审核服务交互的命令以及审核日志文件都需要 root 特权。作为 root 用户确保您执行这些命令。

可以使用auditctl -l命令查看当前的审核规则集。如果不存在则不显示任何规则（这是默认值）：

No rules



可以使用auditctl -s查看审计系统的当前状态：



enabled 1值显示在此服务器上启用了审核。pid值是审计守护程序的进程号。pid为0表示审计守护程序未运行。lost条目将告诉您由于内核审计队列溢出而丢弃了多少事件记录。backlog字段显示当前有多少事件记录排队等待auditd读取它们。

通过审核系统调用，您可以远远超出应用程序级别跟踪服务器上的活动。系统调用规则的语法是：

auditctl -a action,filter -S system\_call -F field=value -k key\_name

在上述命令中将-a替换为-A将在顶部而不是在底部插入该规则。

action和filter指定何时记录某个事件。action可以是always或never。filter指定将哪个内核规则匹配过滤器应用于事件。规则的匹配滤波器可以是下列之一：task，exit，user，和exclude。action,filter在大多数情况下将是always,exit，它会告诉auditctl你想退出时审核本系统调用。

system\_call按名称指定系统调用。可以将多个系统调用分组为一个规则，每个规则在-S选项后指定。也可以使用all这个词。您可以在root用户下使用 ausyscall --dump命令查看所有系统调用的列表及其编号。

field=value 指定其他选项，这些选项可根据指定的体系结构，用户ID，进程ID，路径等修改规则以匹配事件。

key\_name 是一个可选字符串，可帮助您稍后识别生成特定日志条目的规则或一组规则。

现在让我们看一些示例系统调用规则。

要定义记录特定用户（使用UID 1001）访问的文件的规则，并使用userfileaccess标记日志条目：

auditctl -a always,exit -F arch=b64 -F auid=1001 -S open -k userfileaccess

为了定义创造日志项的规则，每次由 ID 是 500 或更大的系统用户删除或者重命名文件时，使用（-F auid!=4294967295 选项排除没有设定登录 UID的用户），执行以下命令：

auditctl -a always,exit -S unlink -S unlinkat -S rename -S renameat -F auid>=500 -F auid!=4294967295 -k delete

**2.2 任务描述**

任务描述：

1、编写一个程序，使用特定的系统调用，并用auditctl命令编写规则进行监视。（指定uid或者pid）并解析日志文件

2、运行一段时间后，查看程序进行系统调用后的输入日志：cat /var/log/audit/audit.log，并任选几条不同类型的日志进行解析。

**2.3 审核要求**

1. 正确编写满足功能的源文件，正确编译。
2. 正常加载、卸载内核模块；且内核模块功能满足任务所述。
3. 提交相关源码与运行截图。

**2.4 参考答案**

1）get\_id\_info.c

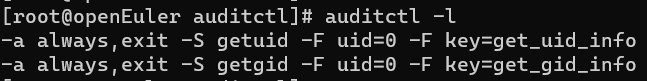
|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  int main(int argc, char \*argv)  {  uid\_t uid = getuid();  gid\_t gid = getgid();  pid\_t pid = getpid();  printf("uid = %u, gid = %u, pid = %u\n", uid, gid, pid);  return 0;  } |

2) 使用auditctl添加审核规则

auditctl -a always,exit -S getuid -F uid=0 -F key=get\_uid\_info

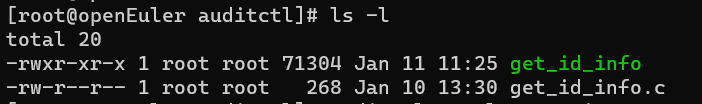
auditctl -a always,exit -S getgid -F uid=0 -F key=get\_gid\_info

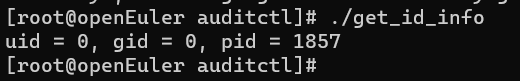


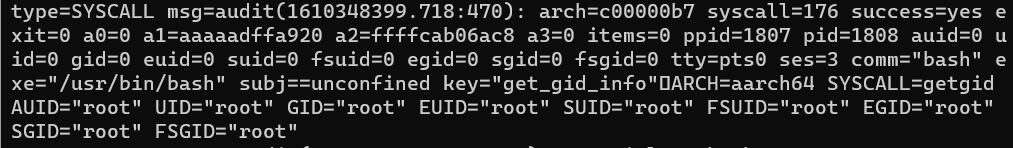
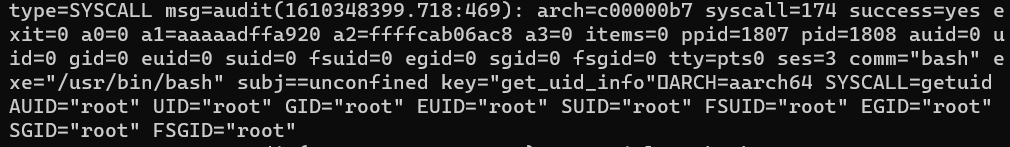
3) 使用gcc编译get\_id\_info.c

gcc -o get\_id\_info get\_id\_info.c



4) 运行get\_id\_info，查看/var/log/audit/audit.log





以上事件由2个记录组成（每个以 type= 作为开始），共享相同的时间戳和编号。每个记录包含好几对 *name*=*value* ，由空格或者逗号分开。以下是对第一个记录的详细分析：

**type=SYSCALL**

每条记录都是以type=”keyword“开头，SYSCALL表示这条记录是向内核的系统调用触发产生的。

**msg=audit(1610348399.718:469):**

在audit(time\_stamp:ID)格式中，记录时间戳，从1970年1月1日00:00:00到现在的时间，ID为记录中唯一的ID标识，同一个事件产生的ID是相同的，如上访问audit\_test目录会触发产生三条日志，但是事件ID是相同的。

**arch=c00000b7**

表示系统的CPU架构，这个十六进制表示“aarch64”，使用命令ausearch -i --arch c00000b7可以打印出有这部分内容的audit.log中日志的解释。

**syscall=174**

系统调用的类型，类型值为174，在/usr/include/asm/unistd\_64.h中有定义，这里174表示openat，可以使用命令ausyscall来查询不同的数字对应的系统调用名称。或者使用ausyscall --dump命令来显示所有的系统调用。

**success=yes**

　　表示系统调用成功与否

**exit=0**

exit 输入栏包含详细说明由系统调用所返回的退出代码的值。在不同的系统调用中，值各不相同。

**a0=a a1=0 a2=474f444843544157 a3=313d**

a0 到 a3 输入栏记录了前四个参数，在这个事件中使用 16 进制编码系统调用。这些参数取决于使用的系统调用。

**items=0**

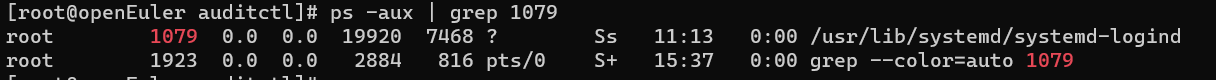
表示跟在系统调用后，补充记录的个数。

**ppid=1**

父进程ID，如bash的ID。

**pid=** **1079**

进程Id。使用ps-aux | grep 1079命令可以得知该进程号属于systemd-logind



**auid=** **4294967295**

auid 输入栏记录了审核用户 ID

**uid=0**

uid 输入栏记录了开始分析进程的用户 ID。

**gid=0**

gid 输入栏记录了开始分析进程用户的 ID组。

**euid=0**

euid 输入栏记录了开始分析进程用户的有效用户 ID。

**suid=0**

suid 输入栏记录了开始分析进程的用户的设置用户 ID。

**fsuid=500**

fsuid 输入栏记录了开始分析进程用户的文件系统用户 ID。

**egid=0**

egid 输入栏记录了开始分析进程用户的有效群组 ID。

**sgid=0**

sgid 输入栏记录了开始分析进程用户的设置群组 ID。

**fsgid=**

fsgid 输入栏记录了开始分析进程的用户的文件系统群组 ID。

**tty=(none)**

tty 输入栏记录了调用分析进程的终端。

**ses=** **4294967295**

ses 输入栏记录了调用分析进程会话的会话 ID。

**comm="** **systemd-logind "**

comm 输入栏记录了命令行的名字，它被用于调用分析进程。

**exe="** **/usr/lib/systemd/systemd-logind "**

exe 输入栏记录了被用来调用分析进程的可执行的路径。

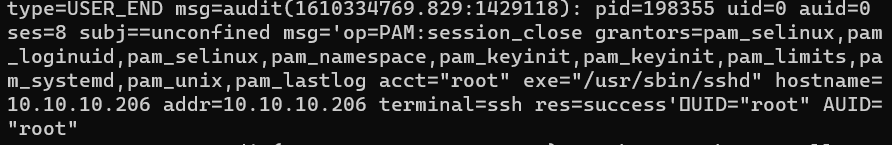
**subj==unconfined**

subj 输入栏记录了 SELinux 语境，运用此语境可以在执行时间中标注分析进程。

**key="get\_uid\_info"**

key 输入栏记录了所有管理者定义的字符串。

5)查看/var/log/audit/audit.log选择部分日志文件解析



type=USER\_END

USER\_END表示当用户空间会话被终止时被触发。



**type=CWD**

在第二记录中，type 输入栏数值是 CWD — 当前工作目录。这种类型用于记录在被执行的第一记录中详细说明的触发系统调用的进程。

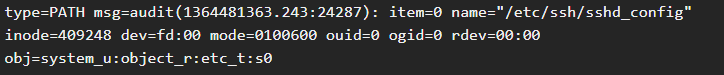
这个记录的目的是为了记录当前进程的位置以防在相关的 PATH 记录中捕捉到相对路径。运用这个方法可以重塑绝对路径。

**msg=audit(1364481363.243:24287)**

msg 输入栏持有与第一记录中的数值相同的时间戳和 ID 值。

**cwd="/home/shadowman"**

cwd 输入栏含有进入目录的路径，在目录中触发系统调用。



**type=PATH**

在第三记录中， type 输入栏值是 PATH。每个审核事件包含一个 PATH 对于每条路径种类的记录作为一个参数，传输给系统调用。在审核事件中，只有一条路径（/etc/ssh/sshd\_config）被用来作为参数。

**msg=audit(1364481363.243:24287):**

msg 输入栏持有与第一和第二记录中的值相同的时间戳和 ID 值。

**item=0**

item 输入栏表明在所有项目中，哪个项目在 SYSCALL 类型记录中，参考了当前记录。这个是数字是以零为基准；值 0 意味着它是第一项。

**name="/etc/ssh/sshd\_config"**

name 输入栏记录了文件或者目录的所有路径，作为参数被传输给系统调用。在这种情况下，它是 /etc/ssh/sshd\_config 文件。

**inode=409248**

inode 输入栏包含索引结点数字，与记录在事件中的文件和目录有关。以下命令体现了与 409248 索引结点数字相关的文件和目录：

~]# find / -inum 409248 -print

/etc/ssh/sshd\_config

**dev=fd:00**

dev 输入栏明确说明了设备的次要和主要 ID，它包含记录在事件中的文件和目录。在这种情况下，值代表 /dev/fd/0 设备。

**mode=0100600**

mode 输入栏记录了文件和目录权限，用 16 进制表示法编码。在这种情况下，0100600 可以被解释为 -rw-------，意味着对于 /etc/ssh/sshd\_config 文件，只有 root 用户拥有读取并且输入权限。

**ouid=0**

ouid 输入栏记录了对象所有者的用户 ID。

**ogid=0**

ogid 输入栏记录对象拥有者的群组 ID。

**rdev=00:00**

rdev 输入栏包含记录的设备识别器只用于特殊文件。在这种情况下，正常文件是不用来作为记录文件的。

**obj=system\_u:object\_r:etc\_t:s0**

obj 输入栏记录了 SELinux 语境，运用此语境可以在执行时间中标注分析进程。

**任务3：使用aureport命令与ausearch命令(30min)**

**3.1 相关知识**

从2.2已经知道RAW类型的audit log会存放在/var/log/audit目录下，这些log体量大而且比较难懂，用aureport可以轻易的统计量化日志报告。默认情况下，aureport查询在 /var/log/audit/ 目录中的所有 audit.log 文件来创建报告。您可以指定不同的文件来运行报告而不使用 aureport options -if file\_name 命令：

aureport的用法：

-au, --auth：生成关于认证请求（尝试）的report

-a, --avc：生成avc message的report

–comm：生成关于执行命令的report

-c, --config：生成关于配置修改的report

-cr, --crypto：生成crypto events的report

-e, --event：生成events的report，

-f, --file：生成关于files和af\_unixsocket的report

–failed：打印所有failed的events

-h, --host：打印host的report

–help：打印帮助

-i, --interpret：将数字编码解析问字符串（很多字段都进行了数字编码，通过此参数还原）

-if, --input file|directory：使用给定的文件或者目录来生成report，用于分析其他机器的日志

–input-logs：使用auditd.conf中配置的日志文件路径来做检索，该参数用于写cron job的时候配合使用

–integrity：生成完整事件的report

-k, --key：生成audit rule key的report（标明日志来源的key）

-l, --login：生成login的report

-m, --mods：生成account modification（账号修改）的report

-ma, --mac：生成MAC（MandatoryAccess Control，Linux系统中有两种自主访问控制策略，一种是9位权限码（User-Group-Other），另一种是访问控制列表ACL（Access Control List）。）events的report

-n, --anomaly：生成不正常的event report，例如网卡变为混杂模式，程序异常等等

–node node-name：生成特定node的report

-nc, --no-config：排除CONFIG\_CHANGE event，生成report

-p, --pid：生成process的report

-r, --response：将所有不正常event的response做成report

-s, --syscall：生成syscall的report

–success：生成所有成功的events的report

–summary：将report整理成一个概述（统计数量分类输出）

-t, --log：生成所有日志文件的起止时间

–tty：生成tty（终端设备）的输入（按键）report

-te, --end[end-date][end-time]：匹配time stamp等于或早于给定参数的event来生成report，时间格式与本地时间格式匹配，如果省略掉日期，则默认指定today作为默认值。如果省略掉时间，则now作为默认值，使用24小时制。例如09/03/2009 18:00:00。常用的一些预定义的关键字：now/recent/today/yesterday/this-week/week-ago/this-month/this-year，其中，recent表示10分钟之前，其他的都可以按照字面意思理解

-ts, --start[start-date][start-time]：与-te参数类似，用于过滤日期和时间晚于给定时间的event生成report。区别是如果时间参数省略，会以midnight作为默认值。

-tm, --terminal：生成terminal的report

-u, --user：生成user的report

-v, --version：打印版本

–virt：生成虚拟化事件的report

-x, --executable：生成executable的report

要以更可读的格式产生结果，比如用它们映射到的用户名替换UID，则也要使用-i选项：

aureport - -i

　　要显示每个日志的启动和停止时间，可以添加-t选项：

aureport - -i -t

　　要显示等于或早于特定时间的事件，可以添加-te选项，并在后面跟着结束日期和结束时间。用数字格式表示您所在地点的日期和时间，并以24小时制格式表示时间。例如，对于en\_us.UTF-8这个地方，可使用日期格式MM/DD/YY：

aureport - -i -te

　　要显示等于或者晚于特定时间的事件，添加-ts选项，后面跟着开始日期和时间。采用与-te选项相同的日期和时间格式化规则。

aureport - -i -ts

　　要仅显示失败事件，则使用- -failure，注意这个选项前面有两条虚线而不是一条：

aureport - -i --failed

　　要仅显示成功事件，则使用- -success，注意这个选项前面有两条虚线而不是一条：

aureport - -i --success

　　有些报表也可以用- -summary选项以汇总格式生成；注意这个选项前面有两条虚线作前缀：

aureport - -i --summary

　　要产生汇总报表而不是关于一个地区的报表，可使用-r选项：

aureport -r -i

　　要产生来自一个日志文件的报表而不是默认报表，则可用-if选项指定它：

aureport - -i -if /var/log/audit/audit.log.1

aureport帮助我们生成总体的日志总结， 如果我们对特定的event感兴趣，我们可以通过ausearch去过滤想要的日志。

ausearch - option -if myfile

它可以指定特定的日志文件进行分析， 通过加上"-i"可以将数据格式的，转化成可读的文本格式，比如user ID 和ASCII 码形式的cmd。

ausearch的用法：

ausearch -m #按消息类型查找

ausearch -ul #按登陆ID查找

ausearch -ua #按uid和euid查找

ausearch -ui #按uid查找

ausearch -ue #按euid查找

ausearch -ga #按gid和egid查找

ausearch -gi #按gid查找

ausearch -ge #按egid查找

ausearch -c #按cmd查找

ausearch -x #按exe查找

ausearch -sc #按syscall查找

ausearch -p #按pid查找

ausearch -sv #按syscall的返回值查找（yes/no）

ausearch -f #按文件名查找

ausearch -tm #按连接终端查找(term/ssh/tty)

ausearch -hn #按主机名查找

ausearch -k #按特定的key值查找

ausearch -w #按在audit rule设定的字符串查找

**3.2 任务描述**

任务描述：

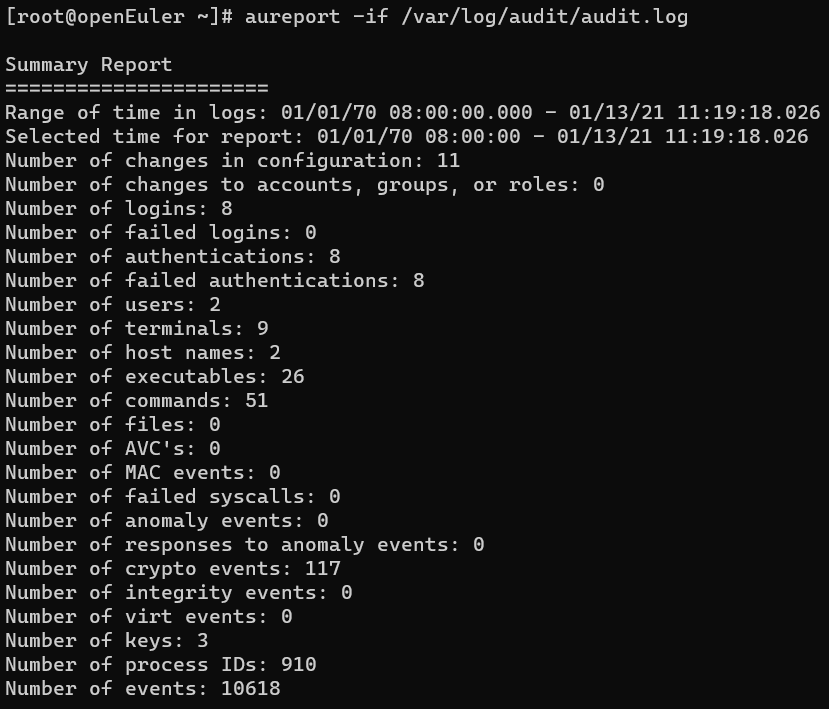
1、使用aureport命令查看日志的报表；

2、改写先前的程序和规则，尽量添加不同的规则，并使用ausearch搜索与之相关的审计记录。

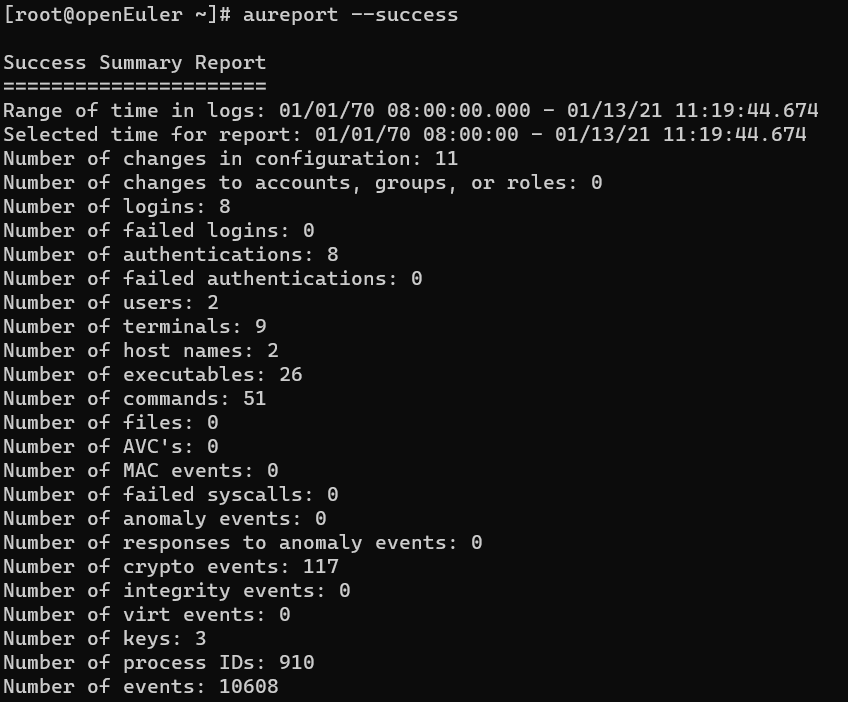
**3.3 审核要求**

1. 正确编写满足功能的源文件，正确编译。
2. 正常加载、卸载内核模块；且内核模块功能满足任务所述。
3. 提交相关源码与运行截图。

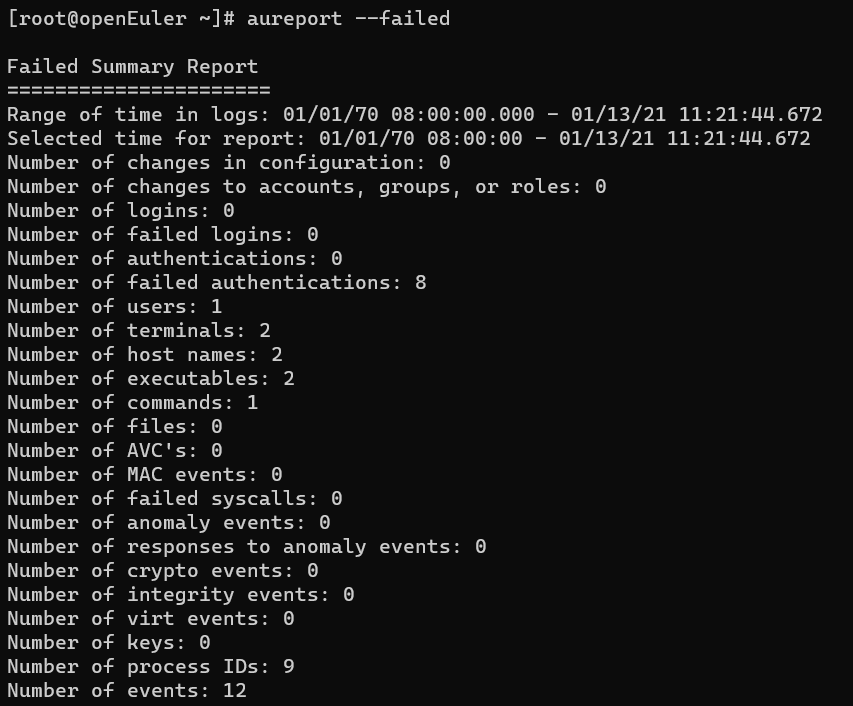
**3.4 参考答案**

1）使用aureport统计/var/log/audit/audit.log的日志报告

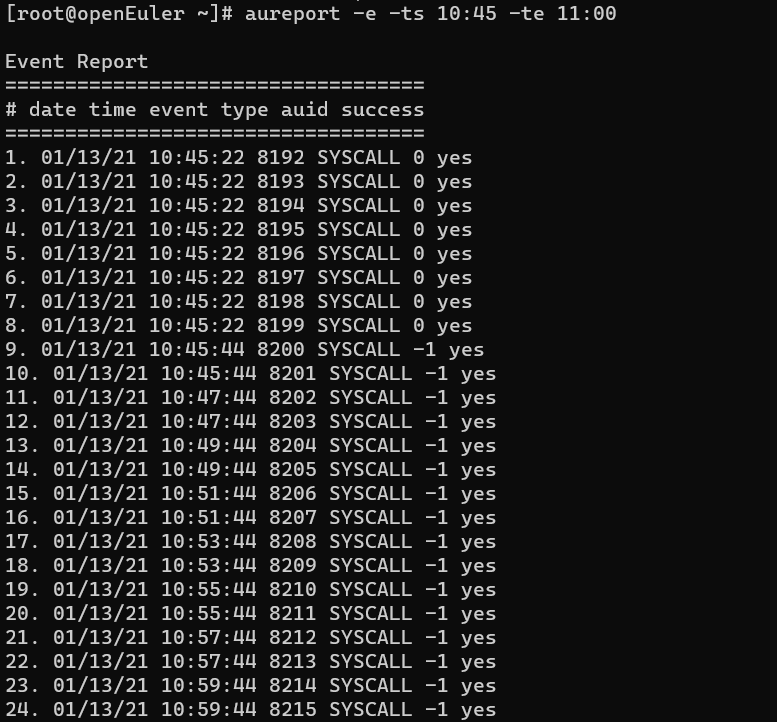
2）使用aureport统计成功的日志报告

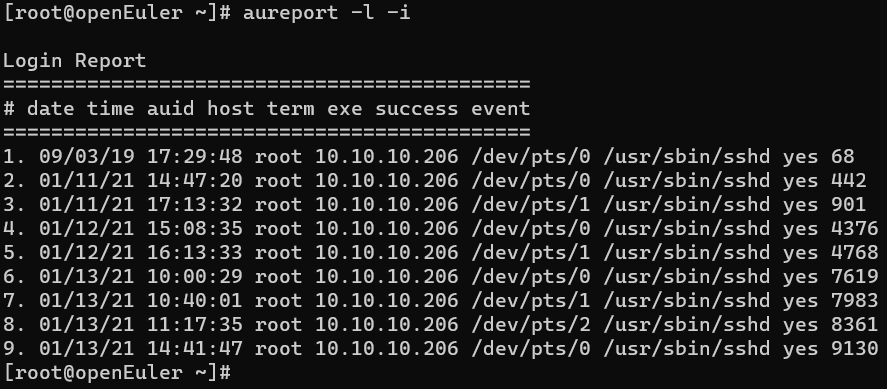


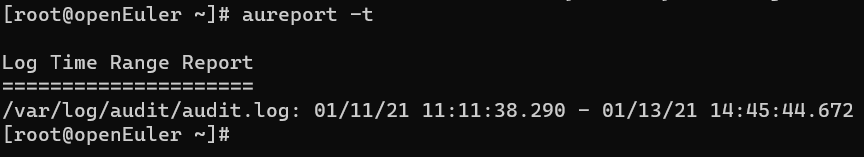
3）使用aureport统计失败的日志报告



4）使用aureport统计10:45 – 11:00的日志报告



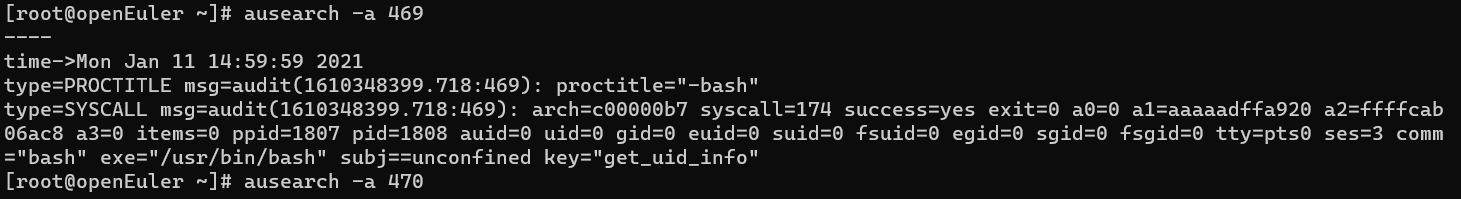
5）使用aureport统计用户登录事件生成的报告

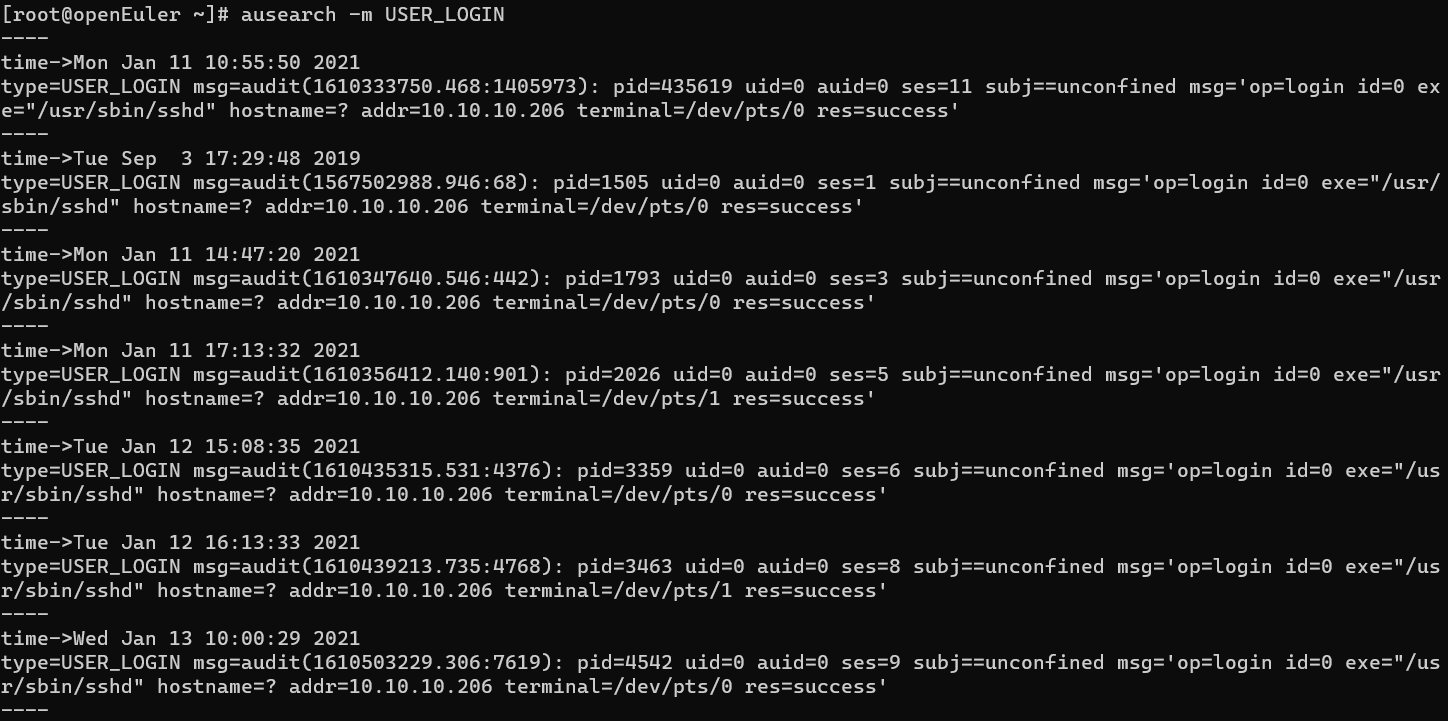
6）查看audit log文件包含日志的起止时间

7）使用auditctl添加审核规则

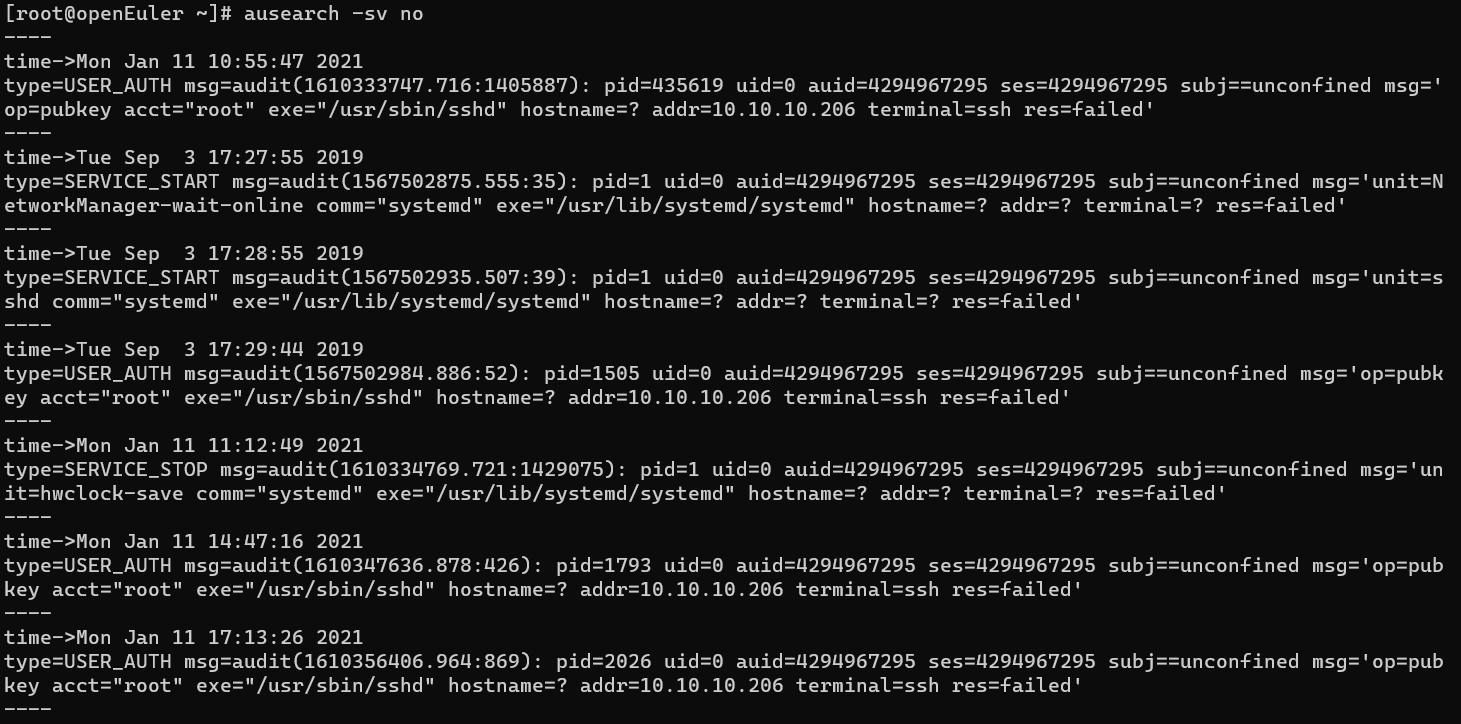
auditctl -a always,exit -S getpid -F uid=0 -F key=get\_pid\_info

重新运行get\_id\_info

8) 使用ausearch查找event ID等于469的log的消息

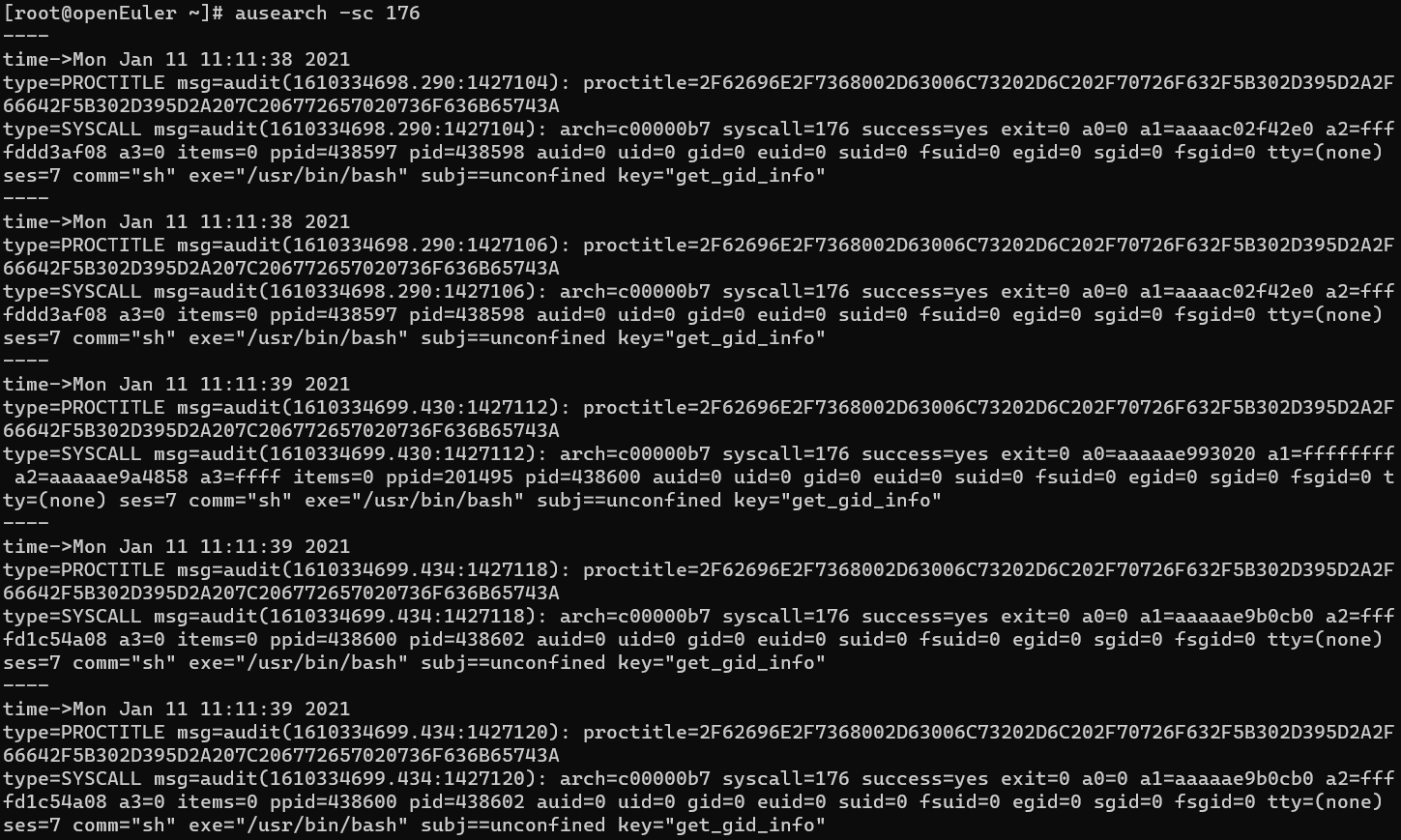
9）使用ausearch查找用户登录类型的log的消息

10）使用ausearch查找系统调用返回值为no类型的log的消息

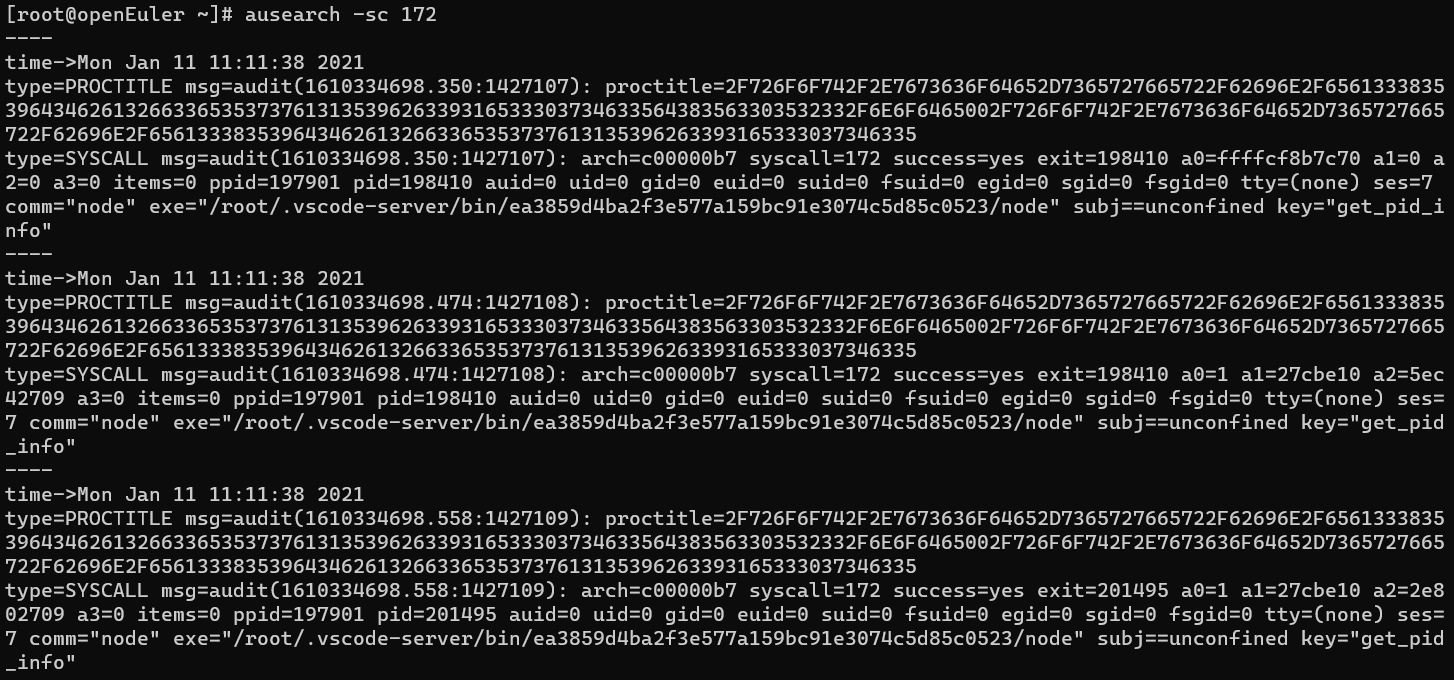


11）使用ausearch查找pid=4636的log的消息

12）使用ausearch查找SYSCALL = get\_gid的log的消息



13) 使用ausearch查找SYSCALL = get\_pid的log的消息



13) 使用ausearch查找SYSCALL = get\_uid的log的消息