# 操作系统实验-RBAC访问控制

## 实现的功能

1. 开关

当开关关闭时则我们的LSM-youho将停止工作，而开关开启时，它将开始工作。

1. 用户管理

我们的用户管理直接采用的是linux自带的机制。

1. 角色管理

包括角色的新增，删除，修改功能，每个角色都有各自的权限。

可以为每个用户分配角色，用户将可以拥有角色所拥有的访问权限。

在这里我们的权限集中于文件打开权限，不同的角色可以有不同的文件打开权限。

## 二、实现思路

1. 内核实现

实现了简单的LSM，取代SELinux，利用文件打开钩子，进行用户权限的判定，如果用户对应的角色对所要访问的文件具有访问权限，则允许访问，否则拒绝。

1. 用户实现

与用户进行交互，然后将角色的相关配置信息写入我们的配置文件中。

1. 内核态与用户态交互实现

我们利用Proc进行系统交互，以达到配置实时更新的目的

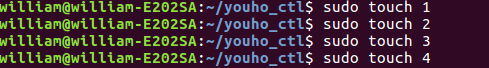
## 实验步骤

### (一) 用户态设置部分

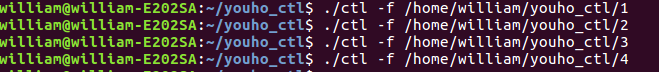
#### 1. 新建用户

利用命令 sudo useradd -m guest 新建一个名为guest的账户。

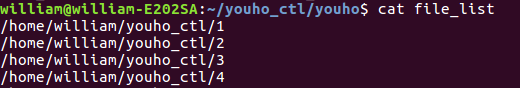
### 2. 将需要LSM管理的文件路径添加到配置文件file\_list中



假设我们有四个文件需要LSM保护，于是我们新建了4个名为1,2,3,4的文件

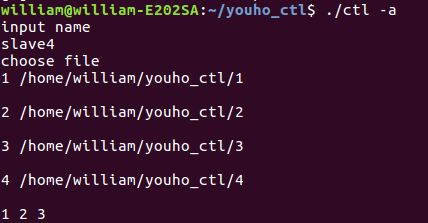


然后我们将它们的地址添加到配置文件file\_list中，方便访问权限的分配。其中ctl是我们的用户态管理工具。

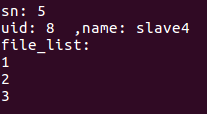


配置文件中的结果如上图。

### 3．添加角色



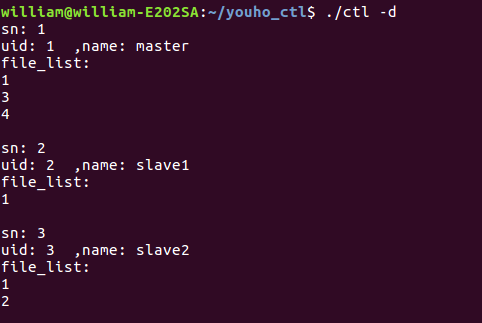
输入角色名称，并且按照序号，授予它访问文件的权限，这里输入1,2,3表示slave4这个角色能访问1,2,3这三个文件。



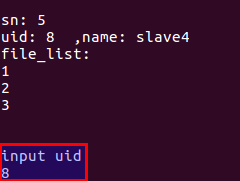
这是添加角色后的详细参数

* SN表示序号
* UID表示角色的id
* Name 表示角色的名称
* File\_list 表示能访问的文件列表

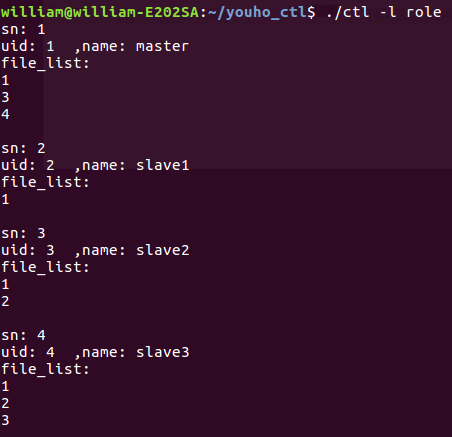
### 删除角色



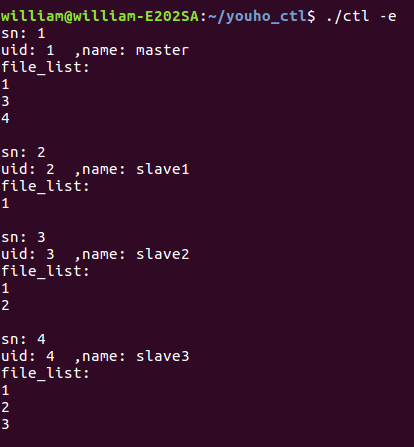
删除时，程序会列出当前的用户列表，选择要删除的用户的uid进行删除，这里我们把刚才新建的slave4删除了



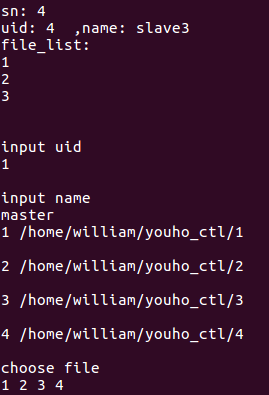
下图是我们通过列表命令展示我们当前的role，可以看到我们的slave4已经被删除了。



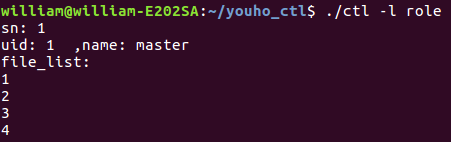
1. 修改角色



编辑命令时将出现当前角色列表，我们选择1，将它的文件权限设置为1,2,3,4

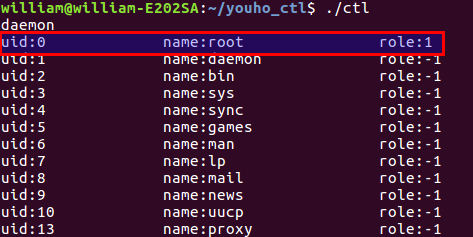


修改成功

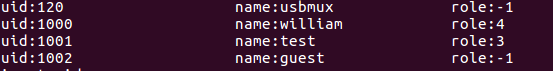


### 分配角色

输入命令后，将调用getpwent()，列出所有的用户



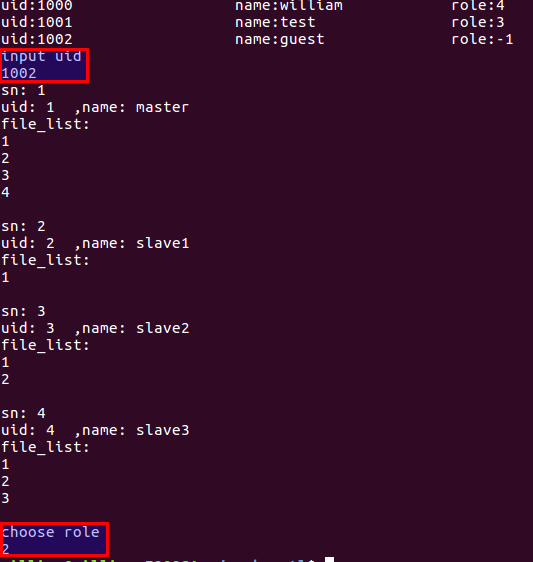
其中role一栏，表示已经分配的角色，比如我们给root分配了1号权限，-1则表示暂未分配角色。



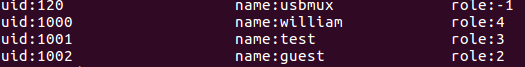
现在这里的william是我当前的账户

Guest是我们刚刚建立的新用户

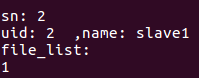
我们现在给它分配2号角色



分配完成后我们再次显示

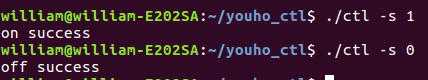


可以发现已经分配成功



Role 2只有访问文件2的权限，因此guest账户也只能访问文件2.

### 开关



1为开，0为关闭

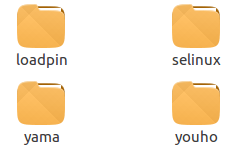
### 实施更新设置

每次设置结束，都会自动调用脚本更新proc文件



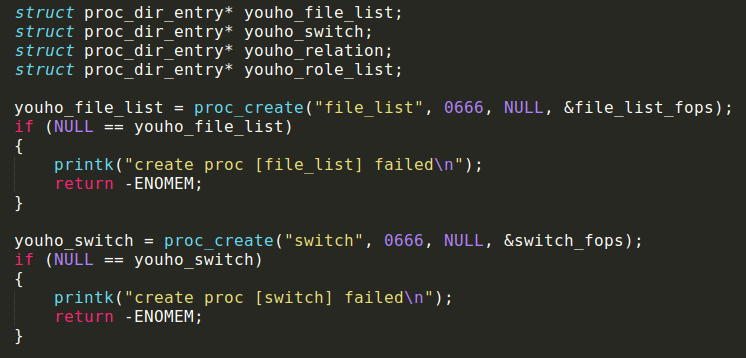
### 内核态的策略

1. 下载内核后在security目录下新建我们的LSM目录youho

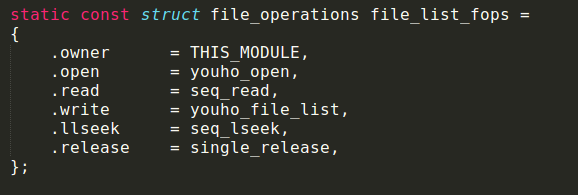


1. 进入这个文件，新建youho\_lsm.c,Kconfig,Makefile这三个文件
2. 修改security目录下以及youho目录下的Kconfig文件和Makefile文件，我们是根据SELinux的配置进行修改的（详细的参考附加的代码）
3. 在youho\_lsm.c中编写代码

首先我们在LSM初始化函数中新建各个配置文件对应的Proc文件



然后绑定我们的proc操作函数。



我们在这里主要是利用write函数，当proc文件被写入的时候，我们的write函数将被调用，然后我们在write函数中进行读取它们的配置文件，存入数组中。

1. 设置钩子

我们了解到要控制文件打开的过程，需要file\_open函数的钩子

我们新建一个static int youho\_file\_open(struct file \*file, const struct cred \*cred)钩子函数。

然后将这个钩子替换 LSM\_HOOK\_INIT(file\_open, youho\_file\_open),放入钩子数组中，其他所需要的钩子都放入这个数组

在我们LSM初始化时，在新建Proc文件之后，执行 security\_add\_hooks(demo\_hooks, ARRAY\_SIZE(demo\_hooks)); 这个添加钩子的函数，将我们的钩子数组注册到系统中去。

1. 进行文件打开策略

算法

传入参数 file和cred /

//file包含着正在访问的文件信息，cred包含着当前用户信息

IF(开关=1)

读取file\_list配置，看访问文件是否在保护列表中

读取relation配置，看当前用户分配到的role

IF(存在这两者的信息)

读取对应role的访问权限

IF(文件在访问列表中)

允许访问

ELSE

拒绝访问

END IF

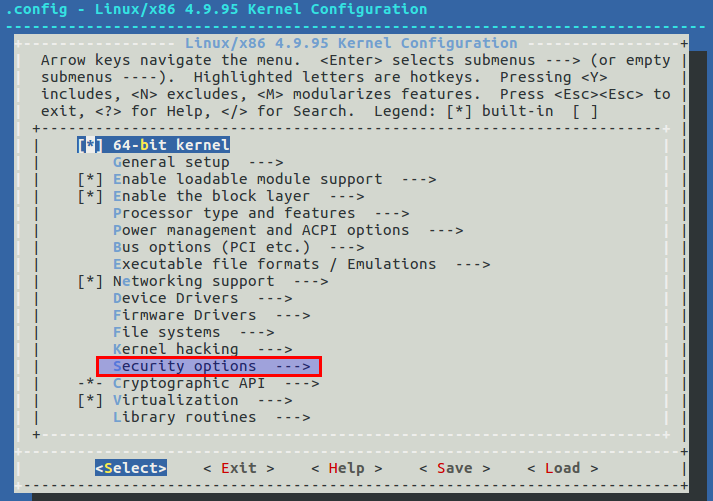
END IF

END IF

7，配置内核

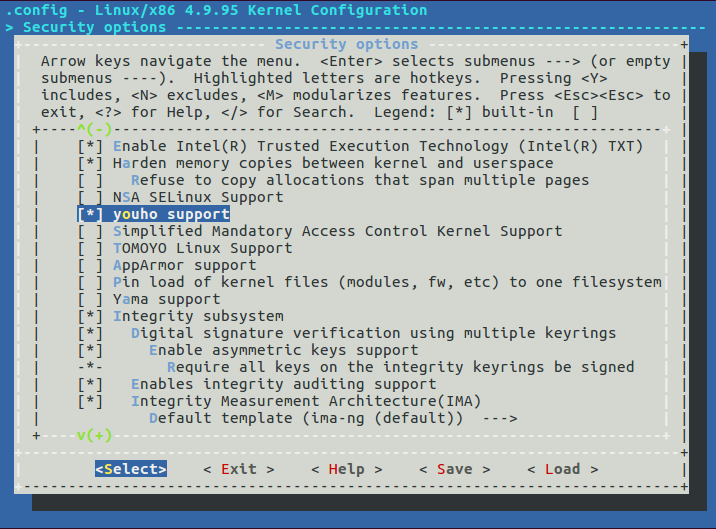
编写完成后，我们需要重新编译和安装内核，才能将我们的LSM应用到系统中

输入sudo make menuconfig，会显示配置界面

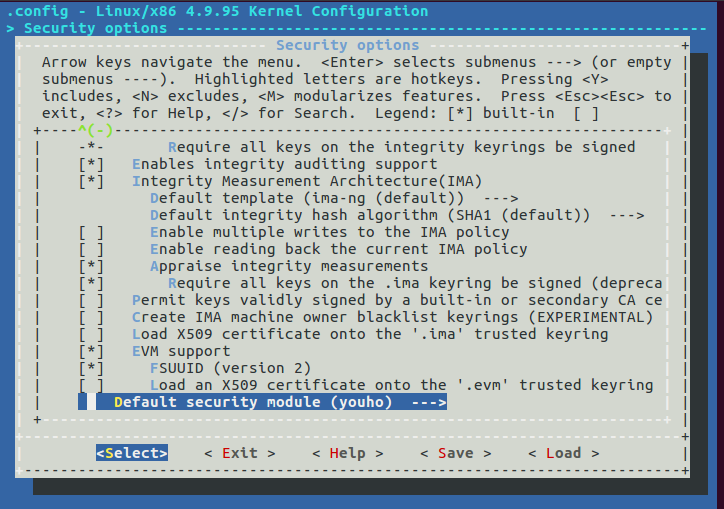


进入“security options”

选中



并且将默认安全模块，选为我们的LSM-youho



保存后分别执行

sudo make -j8

Sudo make modules\_install

Sudo make install

编译安装内核，重新启动后选用我们新的内核进入系统。

### 实现成果展示

uid:0 name:root role:1

uid:1000 name:william role:4

uid:1001 name:test role:3

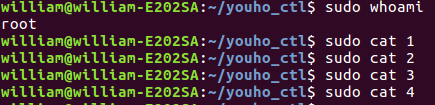
uid:1002 name:guest role:2

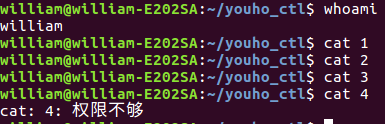
我们给root用户分配了第1个角色master，可以访问1，2，3，4文件

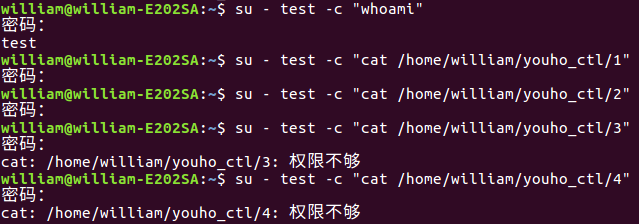
给william分配了第4个角色slave3，可以访问1，2，3文件

给test 分配了第3个角色slave2，可以访问1,2文件

给guest分配了第2个角色slave1，可以访问1文件









实验结果与我们预期的相符合。

## 总结

因为最开始策略设计的不够完善，目前的LSM没有安全性可言，只能达到演示效果，比如

1. 由于文件是根据绝对路径来识别的，将文件换一个地方储存，就可以绕过了。
2. 配置文件的保护未考虑特别完善，目前考虑到将配置文件和proc文件添加到保护列表中，但是系统启动时，登录的用户不一定有权限修改配置文件，因此将配置文件内容再次读入到到proc文件中还没有很好的解决方案