《信息系统安全》

实验指导手册

华中科技大学计算机学院

二零二四年 六月

实验二

系统安全

目 录

第一章 实验目标和内容 1

1.1 进程约束 1

第二章 实验指南 8

2.1 APPARMOR 8

2.2 服务程序调试 14

2.3 SECCOMP 使用 17

第一章 实验目标和内容

1.1进程约束

1.1.1 实验目的

- ✧ 特权隔离（Privilege Separation）、最小特权（Least Privilege）、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

1.1.2 实验环境

- ✧ VMware Workstation 虚拟机。
- ✧ Ubuntu 操作系统或其它 Linux，无具体版本要求。

1.1.3 实验要求

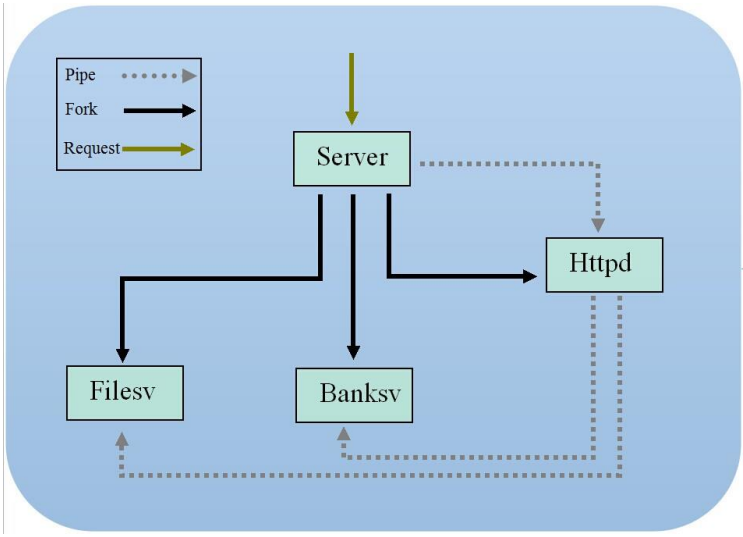
- ✧ 熟悉进程约束的原理，包括 chroot、setuid、seccomp、apparmor 等，针对实验所提供的 web 服务程序，完成本实验指导书所要求的实验内容。

1.1.4 实验内容

实验所提供的代码中，包括 web server 程序源代码、示例的 exploit 代码（作为 web server 的客户端）等。

其中示例 exploit 代码，只需针对自己的实验环境，进行修改就可以进行 exploit。

注意：任务 2、任务 3、任务 4、任务 5，都需在原始代码的基础上，分别进行实验，不是递进关系。



上图描述了名为 touchstone Web 服务器的整体设计和架构。服务器由三个主要部分组成：服务器（server）、HTTP 请求调度器（httpd）和一个或多个服务（Filesv、Bankv 等）。

服务器启动 HTTP 调度器 httpd，以及两个服务：filesv 和 bankv（用黑色实心箭头表示）。服务器接受 80 端口的连接（上面的灰绿色箭头），并将请求路由到 HTTP 调度器（灰色虚线箭头）。

HTTP 调度器从传来的套接字中读取请求行，并决定该请求属于哪一类（GET、POST 等）。一般来说，网络服务器会接受静态和动态请求。为了说明这一点，touchstone Web 服务器由两个样本服务组

成，为静态网页（file service）和动态银行服务（bank service）提供服务，这是一个典型的基于 Web 的电子银行模型。尽管如此，一个生产型网络服务器可能会包含更多种类的服务。

接下来，调度器 httpd 将把请求路由到相应的服务。该服务继续读取 HTTP headers 和 body，并作出适当的回应。

当前的程序架构设计中，实际上已近实现了某种程度上的特权隔离（Privilege Separation），通过将不同的请求路由到不同的服务中，每个服务的实现都可以保持尽可能的简单，并且，每个服务的特权可以进一步进行限制（实现最小特权，Least Privilege）。

任务 1：删除特权文件

为 touchstone 程序添加 **setuid root 权限**，并启动执行

```
$ sudo chown root touchstone
$ sudo chmod +s touchstone
$ ./touchstone
```

进一步，可以使用 web browser 登录该 server，进行 register 和 login

在/tmp 目录下面创建/tmp/test.txt 文件，并将其 owner 改成 root。

```
$touch /tmp/test.txt
$sudo chown root /tmp/test.txt
```

修改 exploit 代码，尝试利用 touchstone 的漏洞，删除特权文件

/tmp/test.txt 文件。

$ python3 ./exploit.py 127.0.0.1 80

任务 2：chroot

修改 server.c，增加 chroot 支持，并重新 make；

同时，使用代码目录中的 chroot-setup.sh，改变 root directory 从/ 到 /jail，并在 jail 中启动 server。

$ chmod +x chroot-setup.sh chroot-copy.sh

$ sudo chroot-setup.sh

$ cd /jail

$ sudo ./touchstone

测试是否可以继续通过上述 exploit 代码，删除（unlink）/tmp 目录下的 root privilege 的文件 /tmp/test.txt；

测试方法：可以在 /jail/tmp 下面创建 test.txt 文件，观察该文件（/jail/tmp/test.txt）和 /tmp/test.txt 文件，哪个被删除。

sudo touch /jail/tmp/test.txt

注意：jail 中的 library 是单独的，位于/jail/lib 下（不同于原先的路径），所以需要重新寻找 libc 的 base 地址（方法见 2.2）。

任务 3：改变进程 euid

修改源代码，使用 setuid (或 setresuid 等)，及时减少 privilege，使得 banksv 进程没有（不必要的）root 权限，并重新 make。

测试是否可以继续删除 root privilege 的文件 /tmp/test.txt。

任务 4：使用 seccomp 限制系统调用

修改源代码，使用 seccomp 方法，对 vulnerable 进程进行约束，并重新 make。参考资料见课件

采用两种方法进行约束：

1）**默认允许，显式拒绝**（Fail Securely）

显式拒绝 unlink，测试是否可以删除特权数据文件；

测试其他的 exploit 途径，如获得 shell 等；

2）**默认拒绝，显示允许**

测试是否可以删除特权数据文件；

测试其他的 exploit 途径，如获得 shell 等；

任务 5：使用 AppArmor 限制进程权限

使用 AppArmor 对 vulnerable 进程进行强制访问控制（无需修改源代码）。

测试是否可以删除特权数据文件；

测试其他的 exploit 途径，如获得 shell 等；

第二章 实验指南

2.1 AppArmor

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。

根据不同程序的访问控制需求，可使用 AppArmor 进行访问控制配置，实施最小特权原则。

检查 apparmor 服务状态：

```
systemctl status apparmor # Checks status
systemctl start apparmor # Starts the service
systemctl enable apparmor # Makes apparmor start on boot
```

检查加载的 profiles：

sudo aa-status

为程序创建 profile：

aa-genprof　　#为首次运行的程序创建 profile

aa-logprof　　#为已存在 profile 的程序，根据 log 修改权限

示例 1：example.sh

创建 data 目录，并创建如下 example.sh 脚本（添加执行权限）：

```
#!/bin/bash
echo "This is an apparmor example."
touch data/sample.txt
echo "File created"
rm data/sample.txt
echo "File deleted"
```

两个终端，终端 1：执行 aa-genprof

```
test@ubuntu:~/apparmor$ sudo aa-genprof example.sh
Writing updated profile for /home/test/apparmor/example.sh.
Setting /home/test/apparmor/example.sh 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:
http://wiki.apparmor.net/index.php/Profiles

Profiling: /home/test/apparmor/example.sh

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.

[(S)can system log for AppArmor events] / (F)inish
Reading log entries from /var/log/syslog.
Updating AppArmor profiles in /etc/apparmor.d.
```

终端 2，执行 example.sh 程序。

然后，在终端 1 中，进行选择（多步选择）：

```
Profile:  /home/test/apparmor/example.sh
Execute:  /bin/touch
Severity: unknown

(I)nherit / (C)hild / (N)amed / (X) ix On / (D)eny / Abo(r)t / (F)inish
```

创建的 profile 如下：

```
test@ubuntu:~/apparmor$ sudo cat /etc/apparmor.d/home.test.apparmor.example.sh
# Last Modified: Tue Jun 15 20:45:51 2021
#include <tunables/global>

/home/test/apparmor/example.sh {
  #include <abstractions/base>
  #include <abstractions/bash>
  #include <abstractions/consoles>

  /bin/bash ix,
  /bin/rm mrix,
  /bin/touch mrix,
  /home/test/apparmor/example.sh r,
  /lib/x86_64-linux-gnu/ld-*.so mr,
  owner /home/*/apparmor/data/sample.txt w,

}
```

查看 profile 加载状态：

sudo aa-status

验证访问控制策略，修改 example.sh 如下，

```
#!/bin/bash

echo "This is an apparmor example."

touch sample.txt
echo "File created"

rm sample.txt
echo "File deleted"
```

```
test@ubuntu:~/apparmor$ ./example.sh
This is an apparmor example.
touch: cannot touch 'sample.txt': Permission denied
File created
rm: cannot remove 'sample.txt': No such file or directory
File deleted
```

执行 aa-logprof，根据 log，调整权限

```
test@ubuntu:~/apparmor$ sudo aa-logprof
Reading log entries from /var/log/syslog.
Updating AppArmor profiles in /etc/apparmor.d.
Complain-mode changes:
Enforce-mode changes:

Profile:  /home/test/apparmor/example.sh
Path:     /home/test/apparmor/sample.txt
New Mode: owner w
Severity: 6

 [1 - owner /home/*/apparmor/sample.txt w,]
  2 - owner /home/test/apparmor/sample.txt w,
(A)llow / [(D)eny] / (I)gnore / (G)lob / Glob with (E)xtension / (N)ew / Audi(t) / (O)wner permissions off / Abo(r)t / (F)inish
Adding owner /home/*/apparmor/sample.txt w, to profile.

= Changed Local Profiles =

The following local profiles were changed. Would you like to save them?

 [1 - /home/test/apparmor/example.sh]
(S)ave Changes / Save Selec(t)ed Profile / [(V)iew Changes] / View Changes b/w (C)lean profiles / Abo(r)t
Writing updated profile for /home/test/apparmor/example.sh.
```

```
test@ubuntu:~/apparmor$ sudo cat /etc/apparmor.d/home.test.apparmor.example.sh
# Last Modified: Tue Jun 15 21:04:30 2021
#include <tunables/global>

/home/test/apparmor/example.sh {
  #include <abstractions/base>
  #include <abstractions/bash>
  #include <abstractions/consoles>

  /bin/bash ix,
  /bin/rm mrix,
  /bin/touch mrix,
  /home/test/apparmor/example.sh r,
  /lib/x86_64-linux-gnu/ld-*.so mr,
  owner /home/*/apparmor/data/sample.txt w,
  owner /home/*/apparmor/sample.txt w,

}
```

除了根据 log 修改 profile，也可以手动修改 profile，但是如果手动修

改，需要重新加载 profile。

sudo apparmor_parser -r /etc/apparmor.d/home.test.apparmor.example.sh

卸载 profile：

sudo apparmor_parser -R /path/to/profile

示例 2：passwd

将 passwd 程序拷贝到用户路径，添加 chown 到 root，chmod 添加 s 权限。

同上例，在两个终端上分别进行操作。

可以通过 dmesg 查看错误日志。

需要多次运行 passwd 程序，并多次运行：sudo aa-logprof

初始 profile：

```
test@ubuntu:~/apparmor$ sudo cat /etc/apparmor.d/home.test.apparmor.passwd
# Last Modified: Wed Jun 16 00:05:11 2021
#include <tunables/global>

/home/test/apparmor/passwd {
  #include <abstractions/authentication>
  #include <abstractions/base>
  #include <abstractions/nameservice>

  /home/test/apparmor/passwd mr,
  /lib/x86_64-linux-gnu/ld-*.so mr,
  owner /proc/*/loginuid r,
  owner /{usr/,}lib{,32,64}/** mr,

}
```

可以成功修改 passwd 的 profile：

```
test@ubuntu:~/apparmor$ sudo cat /etc/apparmor.d/home.test.apparmor.passwd
# Last Modified: Wed Jun 16 00:21:26 2021
#include <tunables/global>

/home/test/apparmor/passwd {
  #include <abstractions/authentication>
  #include <abstractions/base>
  #include <abstractions/dovecot-common>
  #include <abstractions/nameservice>
  #include <abstractions/postfix-common>

  capability audit_write,
  capability chown,
  capability dac_override,
  capability dac_read_search,
  capability fsetid,

  /home/test/apparmor/passwd mr,
  /lib/x86_64-linux-gnu/ld-*.so mr,
  owner /etc/.pwd.lock wk,
  owner /etc/nshadow rw,
  owner /etc/shadow rw,
  owner /proc/*/loginuid r,
  owner /{usr/,}lib{,32,64}/** mr,

}
```

2.2 服务程序调试

1）调试

启动 server 程序，两种方法：

a) 方法一

sudo ./touchstone

b) 方法二，改成 setuid 程序

sudo chown root ./touchstone

sudo chmod +s ./touchstone

查看 server 的进程

ps -a

```
$ ps -a
  PID TTY          TIME CMD
 3053 pts/0    00:00:00 sudo
 3054 pts/0    00:00:00 touchstone
 3055 pts/0    00:00:00 filesv
 3056 pts/0    00:00:00 banksv
 3058 pts/0    00:00:00 httpd
 3255 pts/1    00:00:00 ps
```

使用 gdb 来 attach 程序，并在需要 check 的函数上设置断点（程序编译时，使用了 -g 选项 (见 Makefile)，所以可以以函数名来设置断点）。

sudo gdb

(gdb) attach 3056

(gdb) b Handle_post

(gdb) set follow-fork-mode child

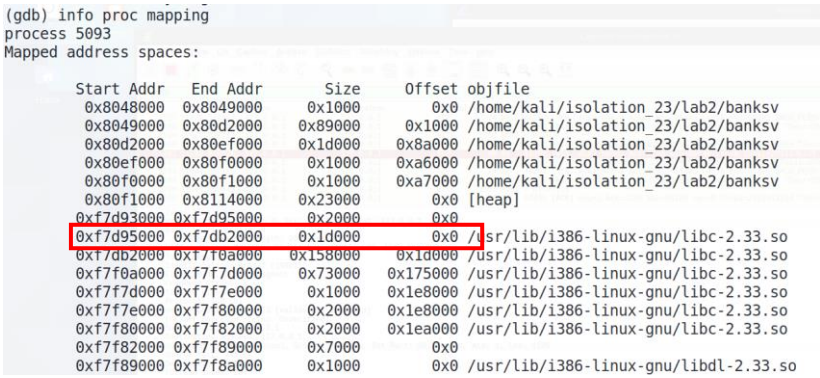
(gdb) c

打开 web browser，输入 127.0.0.1，点击 “Register” or “Login”，发起 POST 请求（POST 请求由 banksv 进程处理，GET 请求由 filesv 进程处理），将触发以上 breakpoint。

同样，运行 exploit 脚本（脚本中构造 POST 请求），也可触发以上 breakpoint。

在 gdb 中，可以查看当前 attach 的进程的信息，如查看进程内存布局，可以看到当前进程使用的 library 的路径，和 library 在内存中的地址：

info proc mapping



需要注意的是，通过 ldd ./banksv 查看的 library，和以上 gdb 里面（通过 info proc mem）查看的 libc，路径可能不一致，需要尝试确定一下路径。

2) 查询函数在 libc 中的相对地址

在以上路径的 libc 中查找相关的函数（和参数）的相对地址，并结合该 libc 的 base 地址（如：上述 libc 的 base 地址为 0xf7d95000），可以进一步构建 exploit。

```
readelf -a /usr/lib/i386-linux-gnu/libc.so.6 | grep " system"
readelf -a /usr/lib/i386-linux-gnu/libc.so.6 | grep " unlink"

ropper --file /usr/lib/i386-linux-gnu/libc.so.6 --string "/bin/sh"
```

3) wireshark 抓包分析

了解 HTTP request 的内容，可以帮助构造有效的 exploit（即，能有效的触发 server 的漏洞，并能执行 shellcode），以及分析 exploit 失败的原因。

4) crash 原因分析

server 开启前：

ulimit -c unlimited

等待 exploit 发起请求，产生 core 文件。

c) 将 core 权限从 root 改到普通用户，以便使用 gdb

sudo chown kali ./core

d) 查看 crash 点的调试信息，分析程序 crash 的原因

gdb ./banksrv core

2.3 seccomp 使用

安装 seccomp 库：

$sudo apt-get install libseccomp-dev:i386

修改 Makefile，添加 -lseccomp 编译选项

```
all:
	gcc -m32 -no-pie -g -o touchstone server.c -lseccomp
	gcc -m32 -no-pie -fno-stack-protector -g -o filesv ./sql_lite3/sqlite3.o -l pthread -l dl ./sql_lite3/sqlhelper.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_lite3/sqlhelper.c banksv.c token.c parse.c http-tree.c handle.c -lseccomp
	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
```

参考课件，以及以下链接（和相关链接）的说明

https://man7.org/linux/man-pages/man3/seccomp_init.3.html

https://man7.org/linux/man-pages/man3/seccomp_rule_add.3.html

默认允许，显式拒绝：

```
ctx = seccomp_init(SCMP_ACT_ALLOW);
rc = seccomp_rule_add(ctx, SCMP_ACT_KILL,
SCMP_SYS(unlink), 0);
```

默认拒绝，显式允许：

```
ctx = seccomp_init(SCMP_ACT_KILL);
rc = seccomp_rule_add(ctx, SCMP_ACT_ALLOW,
SCMP_SYS(unlink), 0);
```

以下是可能使用到的工具：

1）可以使用 strace 来 trace 进程的系统调用情况

```
$ps -a
$sudo strace -p pid  #pid 根据 ps 的结果，来 attach 需要 trace 的进程。
```

https://man7.org/linux/man-pages/man1/strace.1.html

进一步，可以将程序正常执行需要使用的 system calls（从而可排除掉 exploit 调用的非正常 system calls）添加到 seccomp 的规则中（seccomp_rule_add）。

2）可以使用 dmesg 打印 kernel 的 message，来判断是哪个 syscall 被 deny，从而可以进一步根据情况，修改 seccomp 的规则（seccomp_rule_add）。

注意：Web 程序的 Register 功能，涉及到 unlink 系统调用，所以在使用 seccomp 对 unlink 进行约束后，不要再使用 Register 功能。

apparmor 相关资源：

[1]https://medium.com/information-and-technology/so-what-is-apparmor-64d7ae211ed

[2]https://www.inguardians.com/2017/06/08/protecting-the-mr-robot-vuln-hub-machine-part-2-confining-wordpress-with-apparmor/

[3]https://www.secjuice.com/apparmor-say-goodbye-to-remote-command-execution/

[4] https://www.apertis.org/guides/apparmor/

[5] AppArmor_Core_Policy_Reference, https://gitlab.com/apparmor/apparmor/-/wikis/AppArmor_Core_Policy_Reference

[6] AppArmor security profiles for Docker, https://docs.docker.com/engine/security/apparmor/