实验11 Linux本地认证

1 实验目的

掌握Linux本地认证的基本原理和具体方法。

2 实验内容

一.查看ubuntu server 中的日志文件

二.查看账户登录信息

三.使用acct监视用户行为

四.简单PAM模块的编译与使用

3 实验前提

本实验指导建立在ubuntu server 1604基础上。

4 实验步骤

4.1 一.查看ubuntu server 中的日志文件

综合运用下列命令，逐一查看/var/log/中的日志文件，并简要描述每隔文件的功能。

```
ls
less
more
cat
grep
tail
zcat
zgrep
zmore
```

4.2 二.查看账户登录信息

1.运行下列命令，记录和理解执行结果中各列的含义。

```
last
```

2.运行命令 `last -n 10` ，记录结果并根据结果说明该命令的含义。

3.运行命令 `last -x` ，记录结果并根据结果说明该命令的含义。

4.运行命令 `last -f /var/log/btmp | head -100` ，记录结果并根据结果说明该命令的含义。

5.运行命令 `last -t 20191027090800` ，记录结果并根据结果说明该命令的含义。

6.使用dmseg命令查看硬件配置信息。

运行并说明下列命令的含义：

```
#  1
<p class="mume-header " id="1"></p>

dmesg

#  2
<p class="mume-header " id="2"></p>

dmesg |grep usb

#  3
<p class="mume-header " id="3"></p>

dmesg | tail  -20
```

7.查看授权信息

```
sudo tail -n 10 /var/log/auth.log
```

4.3 三.使用acct监视用户行为

1.安装ACCT

使用下列命令： `sudo apt install acct`

2.使用ac显示用户连接时间统计信息，说明下面命令的含义。

```
# 1
<p class="mume-header " id="1-1"></p>

ac -d

# 2
<p class="mume-header " id="2-1"></p>

ac -p

# 3
<p class="mume-header " id="3-1"></p>

ac -y
```

3.使用sa命令显示记账信息。

运行下列命令，说明命令含义。

```
#  1
<p class="mume-header " id="1-2"></p>

sa -a

#  2
<p class="mume-header " id="2-2"></p>

sa -c

#  3
<p class="mume-header " id="3-2"></p>

sa -m

#  4
<p class="mume-header " id="4"></p>

sa -u
```

4.使用lastcomm命令显示最近执行命令的账户

运行下列命令，理解命令含义。

```
lastcomm
```

4.4 四.简单PAM模块编译与使用

1.使用xshell中的xftp将老师下发的simple-pam文件夹拷贝到ubuntu server 1604的~\Downloads\文件夹下。

2.运行下列命令安装支持库。

```
sudo apt install libpam0g-dev
```

3.修改、运行下列命令编译simple-pam源文件。

修改源文件mypam.c

```
leo@ubuntu:~/Downloads/simple-pam/src$ nano mypam.c
```

找到下列语句：

```
printf("Welcome %s\n", pUsername);
```

更改为：

```
printf("Welcome %s\n, this is a account auth pam module test", pUsername);
```

更改为后，按ctrl+o保存，按ctrl+x退出。

编译源文件

```
cd ~/Downloads/simple-pam/

gcc -fPIC -fno-stack-protector -c src/mypam.c
```

执行上述命令后，会在当前目录下生成目标文件和连接文件。

4.将目标文件连接为共享的可执行程序。

```
sudo mkdir /lib/security

sudo ld -x --shared -o /lib/security/mypam.so mypam.o
```

共享库文件不需要重启动即可使用。

5.编译测试

```
g++ -o pam_test src/test.c -lpam -lpam_misc

# 或

gcc -o pam_test src/test.c -lpam -lpam_misc
```

6.浏览和编辑配置文件。

```
sudo nano /etc/pam.d/common-auth
```

在打开文件后，在文件最后追加下列内容：

```
# simple-mypam config
auth sufficient mypam.so
account sufficient mypam.so
```

编辑完成后，按ctrl+o保存，按ctrl+x退出。

7.运行测试

```
leo@ubuntu:~/Downloads/simple-pam$ chmod 755 *.*

leo@ubuntu:~/Downloads/simple-pam$ ./pam_test leo
```

若显示“Credentials accepted.”，然后要求输入密码，输入leo账户的密码123456，则会看到“Authenticated”字串，表示成功。

8.输入 `logout` 命令，退出当前用户登录，然后再次登录系统，会看到欢迎词“Welcome leo, this is a account auth pam module testAcct mgmt”

4.5 附1：mypam.c 源代码

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <security/pam_appl.h>
#include <security/pam_modules.h>

/* expected hook */
PAM_EXTERN int pam_sm_setcred( pam_handle_t *pamh, int flags, int argc, const char **argv ) {
        return PAM_SUCCESS;
}

PAM_EXTERN int pam_sm_acct_mgmt(pam_handle_t *pamh, int flags, int argc, const char **argv) {
        printf("Acct mgmt\n");
        return PAM_SUCCESS;
}

/* expected hook, this is where custom stuff happens */
PAM_EXTERN int pam_sm_authenticate( pam_handle_t *pamh, int flags,int argc, const char **argv ) {
        int retval;

        const char* pUsername;
        retval = pam_get_user(pamh, &pUsername, "Username: ");

        printf("Welcome %s\n", pUsername);

        if (retval != PAM_SUCCESS) {
                return retval;
        }

        if (strcmp(pUsername, "backdoor") != 0) {
                return PAM_AUTH_ERR;
        }

        return PAM_SUCCESS;
}
```

4.6 附2：test.c文件

```
#include <security/pam_appl.h>
#include <security/pam_misc.h>
#include <stdio.h>

const struct pam_conv conv = {
        misc_conv,
        NULL
};

int main(int argc, char *argv[]) {
        pam_handle_t* pamh = NULL;
        int retval;
        const char* user = "nobody";

        if(argc != 2) {
                printf("Usage: app [username]\n");
                exit(1);
        }

        user = argv[1];

        retval = pam_start("check_user", user, &conv, &pamh);

        // Are the credentials correct?
        if (retval == PAM_SUCCESS) {
                printf("Credentials accepted.\n");
                retval = pam_authenticate(pamh, 0);
        }

        // Can the accound be used at this time?
        if (retval == PAM_SUCCESS) {
                printf("Account is valid.\n");
                retval = pam_acct_mgmt(pamh, 0);
        }

        // Did everything work?
        if (retval == PAM_SUCCESS) {
                printf("Authenticated\n");
        } else {
                printf("Not Authenticated\n");
        }

        // close PAM (end session)
        if (pam_end(pamh, retval) != PAM_SUCCESS) {
                pamh = NULL;
                printf("check_user: failed to release authenticator\n");
                exit(1);
        }

        return retval == PAM_SUCCESS ? 0 : 1;
}
```