Лабораторная работа #1

Классическая модель разграничения доступа Unix и PAM

Цель: рассмотреть стандартные механизмы реализации дискреционной модели доступа в Unix-подобных ОС, различные способы организации исполнения привилегированных программ, а также механизмы аутентификации, в частности особенности хранения паролей в Unix-подобных ОС.

0. Подготовка к выполнению лабораторной

Читайте [1-32]. Да прибудет с вами сила ✊

1. Полезные функции

Изменение и сброс пароля

1. **Изменение владельца файла**

```
#include <unistd.h>

int chown(const char* pathname, uid_t owner, gid_t group);
int fchown(int fd, uid_t owner, gid_t group);
                                        0, если успешно, -1 в случае ошибки
```

owner задаёт uid нового владельца файла, group определяет новую группу владельцев; для изменения только одного ID второй необходимо задать равным -1.

2. **Изменение прав доступа к файлу**

```
#include <unistd.h>

int chmod(const char* pathname, mode_t mode);
int fchmod(int fd, mode_t mode);
                                        0, если успешно, -1 в случае ошибки
```

mode_t – битовое поле (тоже int), описывает права файла; права задаются логическим объединением констант полей (например, S_IRUSR | S_IWUSR | S_IXUSR)

3. **Проверка установленных прав доступа к файлу**

```
#include <unistd.h>

int access(const char* pathname, mode_t mode);
                                    0, если предоставлены все права, -1 иначе
```

mode_t – битовое поле (тоже int), описывает права файла; права задаются логическим объединением констант полей (например, S_IRUSR | S_IWUSR | S_IXUSR)

4. **Получить идентификатор пользователя по имени/uid**

```
#include <pwd.h>

struct passwd* getpwnam(const char* name);
struct passwd* getpwuid(uind_t uid);
                          возвращают указатель в случае успеха, NULL в случае ошибки
```

пример использования:

```
uid_t ruid = getuid();
    struct passwd *pwd = getpwuid(ruid);
    if (pwd == NULL) {
        err_exit("getpwuid failed", 2);
    }
    const char *user = pwd->pw_name;
```

Интерфейс *libpam*

1. **Инициализация и открытие PAM-транзакции**

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

int pam_start(const char *service_name, const char *user,
                    const struct pam_conv *pam_conversation,
                    pam_handle_t **pamh);
                                                PAM_SUCCESS в случае успеха
```

в структуре `pamc` передаётся указатель на т.н. conversation function, на Debian-based можно использовать `misc_conv` из `<security/pam_misc.h>`

2. **Аутентификация**

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

int pam_authenticate(pam_handle_t *pamh, int flags);
                                                PAM_SUCCESS в случае успеха
```

`pamh` – битовое поле (тоже `int`), описывает права файла; права задаются логическим объединением констант полей (например, `S_IRUSR | S_IWUSR | S_IXUSR`)

3. **Установка реквизитов пользователя**

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

int pam_setcred(pam_handle_t *pamh, int flags);
                                                PAM_SUCCESS в случае успеха
```

4. **Открытие сессии**

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

int pam_open_session(pam_handle_t *pamh, int flags);
                                                PAM_SUCCESS в случае успеха
```

5. **Закрытие сессии**

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

int pam_close_session(pam_handle_t *pamh, int flags);
                                                    PAM_SUCCESS в случае успеха
```

6. **Закрытие PAM транзакции**

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

int pam_end(pam_handle_t *pamh, int pam_status);
                                                    PAM_SUCCESS в случае успеха
```

Задания

1. Импортируйте виртуальную машину из предоставленного образа *OSSec_lab1.*.ova*, используйте тип подключения сетевого адаптера «Сетевой мост» (**важно**: не используйте NAT!), организуйте доступ к машине по ***ssh*** (после выполнения задания 3).
2. Измените пароли пользователей *user* и *root*. Определите правила парольной политики, установленные в изучаемой системе (минимальная длина пароля, маска пароля и т.п.); каким образом реализована парольная политика?
   [**Бонус**] В качестве бонусного задания, подберите пароли пользователей *user* и *root*. Пароль *user* можно найти в любой энциклопедии, пароль *root* сложнее, но не длиннее шести символов…
3. Попробуйте войти в систему как *root*. Определите, какие механизмы препятствуют входу в систему. Отключите блокировку логина от имени *root*.
4. Попробуйте войти в систему как *user*. Определите, какие механизмы препятствуют входу в систему. Отключите блокировку логина от имени *user*.
5. Измените парольную политику средствами модуля PAM *pam_pwquality.so* в соответствии со следующими правилами:
   - две попытки на изменение пароля;
   - минимальная длина пароля не менее 12 символов;
   - не менее двух символов верхнего регистра;
   - не менее двух цифр;
   - не менее трёх специальных символов;
   - не более двух подряд идущих **одинаковых** символов;
   - не более двух подряд идущих символов **одного класса** (цифры, верхний регистр, нижний регистр, спец. символы);
   - запрещено использование паролей, содержащих **в качестве подстрок** следующие слова:
     pass password p@ss p@$$ p@ssw0rd p@$$w0rd user root admin sudo lab laba
   - запрещено использование паролей, содержащихся в словаре rockyou.txt [14].

   Проверьте корректность работы настроенной парольной политики (все правила!), изменяя пароль какого-нибудь пользователя системы.
6. Создайте временного пользователя. Средствами PAM организуйте блокировку пользователей на **один час** после **семи** неудачных попыток ввода пароля в течение **трёх минут**. Сымитируйте нарушение озвученных правил с помощью созданного временного пользователя, проверьте, заблокирован ли пользователь: i) пробуя выполнить вход от имени заблокированного пользователя, ii) с помощью утилиты ***faillock***.
7. Создайте временного пользователя. Средствами PAM заблокируйте возможность доступа по ***ssh*** созданному временному пользователю со всех IP-адресов, кроме IP вашей хостовой машины (на которой запущена виртуальная машина). Проверьте доступность входа в систему от имени временного пользователя по ***ssh*** с хоста и блокировку входа с любого другого IP (подумайте, как можно выполнить такую проверку).
8. Попробуйте **от имени пользователя *user*** прочитать располагающийся в домашней директории */home/user* файл *secret.txt*. Определите, почему не удаётся открыть файл для чтения. Попробуйте изменить права доступа к файлу.
   [**N.B.**] Разумеется, вы можете открыть этот файл от *root*, однако требуется прочитать его именно от имени *user*!

[**Бонус**] Определите механизмы, не позволяющие изменять права доступа к файлу стандартными утилитами ***ch****.

9. Реализуйте (напишите программу на языке *C* или *rust*) собственные упрощённые аналоги утилит ***chmod*** и ***chown***:
    - ***my_chmod***
      программа принимает два аргумента – права (выраженные числом в восьмеричной системе счисления) и название файла, пример вызова:
      ```
      user@vbox:~/lab1$ my_chmod 0777 no_more_secrets.txt
      ```
    - ***my_chown***
      программа принимает два аргумента – имя нового владельца и название файла, пример вызова:
      ```
      user@vbox:~/lab1$ my_chown root i_only_answer_to_root.txt
      ```
    - программы должны работать без повышенных привилегий, т.е. для исполнения ***my_chown*** от *root* нужно использовать внешние механизмы.

    С помощью собственного аналога утилиты ***chmod*** измените права доступа к файлу */home/user/secret.txt* и прочитайте его содержимое.
    [**Бонус**] Определите настоящий тип файла *secret.txt* и извлеките его содержимое.
10. Попробуйте **от имени пользователя *user*** изменить владельца файла *secret.txt*. Какие права необходимы для изменения владельца файла в данной системе?
    [**N.B.**] И вновь root даёт вам возможность изменить владельца, но требуется получить повышенные привилегии в рамках сессии пользователя *user* (т.е. без логина в систему как *root*).
    [**Бонус**] Определите механизмы, не позволяющие пользователю *user* исполнять команды в оболочке (терминале) с повышенными привилегиями.
11. Реализуйте (напишите программу на языке *C* или *rust*) собственный упрощённый аналог утилиты ***sudo***:
    - программа ***my_sudo*** должна иметь аналогичный интерфейс, т.е. принимать в качестве аргументов команду оболочки, которую требуется выполнить, пример запуска:
      ```
      user@vbox:~/lab1$ my_sudo my_chown root secret.txt
      ```
    - перед исполнением указанной команды программа ***my_sudo*** должна запрашивать пароль запускающего её пользователя;
    - программа ***my_sudo*** должна исполнять указанные команды **с правами суперпользователя** (реализуйте этот функционал стандартными средствами Unix – доступ к *root* позволит вам это сделать).
12. С помощью интерфейса сокетов реализуйте на языке *C* (или *rust*) эхо-сервер ***my_echo***, возвращающий клиенту присланные данные. Требования:
    - сервер не должен создавать дочерние процессы (используйте пул рабочих потоков или реализуйте блокирующий однопоточный режим работы);
    - сервер должен иметь возможность слушать входящие соединения **на любом порту**, в том числе из IANA-диапазона общеизвестных (системных) TCP-портов (реализуйте этот функционал стандартными средствами Unix);
    - сервер должен быть реализован, следуя концепции ***защищённого программирования*** (*secure programming*) [23]: i) процесс должен работать с повышенными правами только когда это требуется для выполнения конкретного привилегированного системного вызова, ii) по окончании выполнения всех

привилегированных операций необходимо окончательно (*навсегда*) сбросить повышенные привилегии, без возможности восстановить эти права позднее.

Проверьте корректность работы реализованной программы, запустив ***my_echo*** на порту 777.

13. Организуйте получение необходимых привилегий программой ***my_echo*** с помощью механизма привилегий Linux (capabilities) [24-26].
14. Обеспечьте возможность изменения владельца **любого** файла с помощью программы ***my_chown*** посредством механизма привилегий Linux (не стандартными средствами Unix). Проверьте корректность работы программы, изменяя владельца файлов без использования ***my_sudo***.
15. [**Бонус x3**] Организуйте дополнительный уровень защиты программ ***my_sudo*** и ***my_echo*** с помощью фильтрации доступных системных вызовов посредством механизма seccomp [28], для чего требуется:
    - определить минимальное подмножество требуемых для корректной работы названных программ системных вызовов;
    - в случае ***my_sudo*** ограничение списка сисколов приведёт к невозможности выполнить с её помощью от *root* некоторые команды – определите и мотивируйте список разрешённых системных вызовов;
    - реализовать с помощью т.н. classic BPF [28, 32] фильтры, запрещающие использование ненужных программам ***my_sudo*** и ***my_echo*** системных вызовов (можно использовать *libseccomp*);
    - модифицировать код программ для установки соответствующих BPF-фильтров;
    - проверить корректность работ программ;
    - проверить работу фильтра программы ***my_sudo***, запуская с её помощью приложение, использующее запрещённый системный вызов (можно написать такую программу самостоятельно).

Литература

Изменение и сброс пароля

1. https://adamtheautomator.com/linux-password-reset/
2. https://www.cyberciti.biz/faq/linux-set-change-password-how-to/

Классическая модель разграничения доступа и специальные биты

3. https://help.ubuntu.ru/wiki/%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BD%D1%8B%D0%B5_%D0%BF%D1%80%D0%B0%D0%B2%D0%B0_unix
4. Майкл Керриск. Linux API. Исчерпывающее руководство, гл. **2, 8, 9, 15**. Питер, 2022

Pluggable Authentication Modules

5. https://redos.red-soft.ru/base/manual/admin-manual/safe-redos/pam/
6. https://fossies.org/linux/Linux-PAM-docs/doc/sag/Linux-PAM_SAG.pdf
7. https://man7.org/linux/man-pages/man8/pam.8.html
8. https://man7.org/linux/man-pages/man5/pam.d.5.html
9. https://man7.org/linux/man-pages/man8/pam_unix.8.html
10. https://man7.org/linux/man-pages/man8/pam_time.8.html
11. https://man7.org/linux/man-pages/man5/time.conf.5.html

12. https://linux.die.net/man/8/pam_listfile
13. https://linux.die.net/man/8/pam_pwquality
14. https://github.com/brannondorsey/naive-hashcat/releases/download/data/rockyou.txt
15. https://kifarunix.com/lock-linux-user-account-after-multiple-failed-login-attempts/
16. https://linux.die.net/man/8/pam_faillock
17. https://linux.die.net/man/8/pam_access
18. https://linux.die.net/man/5/access.conf
19. https://docs.freebsd.org/ru/articles/pam/#pam-sample-appl
20. https://habr.com/ru/company/otus/blog/675934/
21. https://habr.com/ru/company/aktiv-company/blog/144700/
22. https://man7.org/linux/man-pages/man3/pam.3.html

Secure Programming

23. Майкл Керриск. Linux API. Исчерпывающее руководство, **гл. 38**. Питер, 2022

Linux Capabilities

24. https://habr.com/ru/company/otus/blog/471802/
25. Майкл Керриск. Linux API. Исчерпывающее руководство, **гл. 39**. Питер, 2022
26. https://man7.org/linux/man-pages/man7/capabilities.7.html

Secure Computing Mode, seccomp

27. https://ru.wikipedia.org/wiki/Seccomp
28. https://lwn.net/Articles/656307/
29. https://habr.com/ru/company/selectel/blog/322046/
30. https://habr.com/ru/post/270165/
31. https://man7.org/linux/man-pages/man2/seccomp.2.html
32. https://habr.com/ru/post/493880/