**Министерство науки и высшего образования Российской Федерации**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Факультет безопасности информационных технологий**

**Дисциплина:**

«Операционные системы»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №8**

«Контроль доступа»

Содержание

[Введение 3](#_Toc184404821)

[1 Apparmor 4](#_Toc184404822)

[1.1 Задание 4](#_Toc184404823)

[1.2 Ход работы 4](#_Toc184404824)

[1.2.1 Теория 4](#_Toc184404825)

[1.2.2 Настройка 4](#_Toc184404826)

[1.2.3 Демонстрация работы 4](#_Toc184404827)

[2 selinux 5](#_Toc184404828)

[2.1 Задание 5](#_Toc184404829)

[2.2 Ход работы 5](#_Toc184404830)

[2.2.1 Теория 5](#_Toc184404831)

[2.2.2 Настройка 5](#_Toc184404832)

[2.2.3 Демонстрация работы 5](#_Toc184404833)

[3 сложн вариант 6](#_Toc184404834)

[3.1 Задание 6](#_Toc184404835)

[3.2 Ход работы 6](#_Toc184404836)

[3.2.1 Теория 6](#_Toc184404837)

[Заключение 7](#_Toc184404838)

[Список использованных источников 8](#_Toc184404839)

Введение

Цель работы – познакомиться с работой Apparmor и selinux, придумать и написать свой PAM-модуль.

Для достижения поставленной цели необходимо решить следующие задачи:

* настроить Apparmor для мониторинга сложного приложения;
* продемонстрировать работу Apparmor при ограниченных правах (оконное приложение или веб-сервер);
* настроить selinux в режиме мандатного доступа (CentOS и др.);
* продемонстрировать работу selinux в двухуровневой модели;
* придумать и написать свой PAM-модуль.

# Apparmor

## Задание

Настроить Apparmor для мониторинга сложного приложения и продемонстрировать его работу при ограниченных правах (оконное приложение или веб-сервер).

## Ход работы

### Теория

AppArmor — программный инструмент упреждающей защиты, основанный на политиках безопасности (известных также как профили), которые определяют, к каким системным ресурсам и с какими привилегиями может получить доступ то или иное приложение. В AppArmor включён набор стандартных профилей, а также инструменты статического анализа и инструменты, основанные на обучении, позволяющие ускорить и упростить построение новых профилей.

### Настройка

Установим необходимые модули:

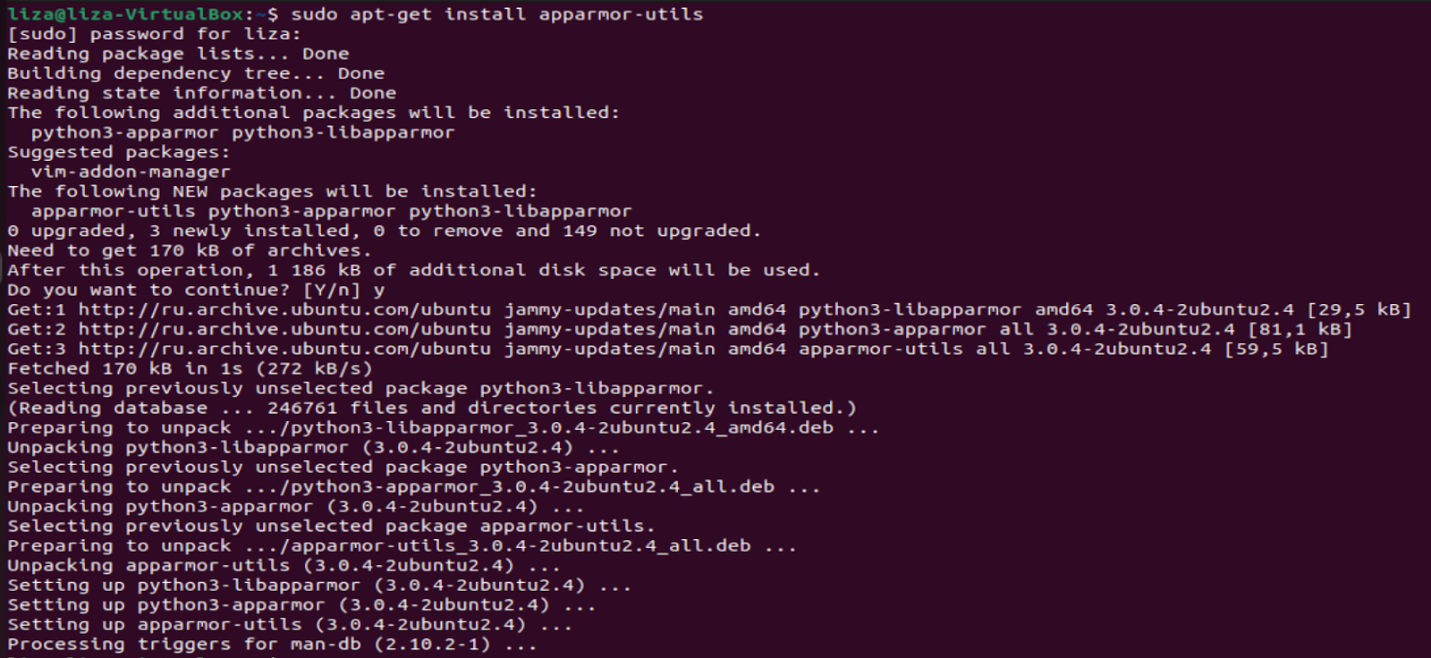


Рисунок 1 – установка apparmor-utils

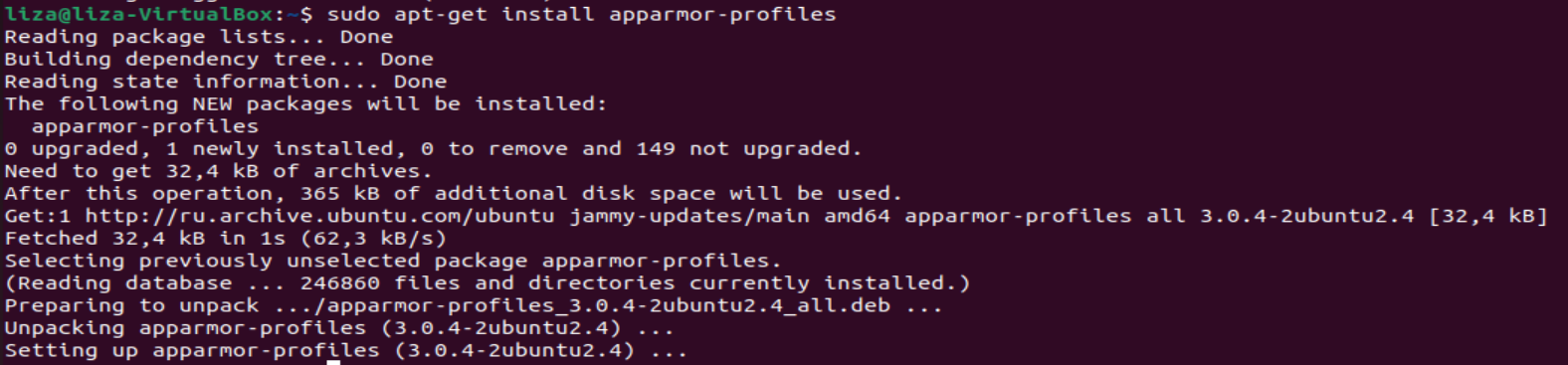


Рисунок 2 – установка apparmor-profiles

Дальше будем настраивать Apparmore для оконного приложения, а именно предустановленный текстовый редактор Ubuntu gedit.

С помощью aa-autodep создаем новый профиль.



Рисунок 3 – создание профиля

Затем обновляем профиль, используя aa-genprof.

****

Рисунок 4 – обновление профиля

Таким образом, получаем следующий профиль.

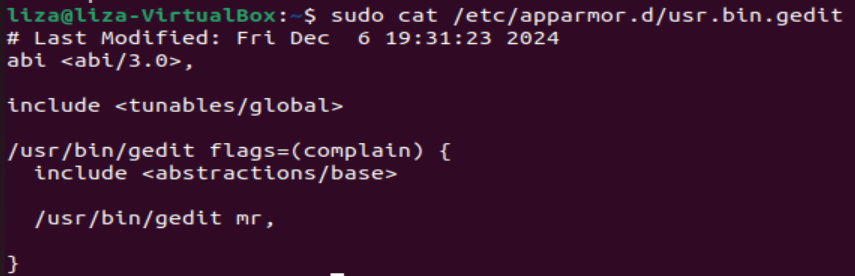


Рисунок 5 – полученный профиль

### Демонстрация работы

Переходим в enforce-режим (режим защиты).



Рисунок 6 – включение режима защиты

Попробуем открыть текстовый редактор. И увидим, что совершить данное действие невозможно.



Рисунок 7 – попытка запуска gedit

Причем, не получается открыть даже с sudo правами.



Рисунок 8 – попытка запуска gedit c sudo

Вернемся в режим мониторинга и попробуем открыть gedit. Теперь все опять работает.

****

Рисунок 9 – переход в режим мониторинга

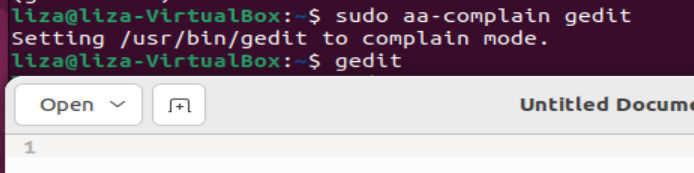


Рисунок 10 – открытие gedit

# selinux

## Задание

Настроить selinux в режиме мандатного доступа (CentOS и др.) и продемонстрировать работу в двухуровневой модели.

## Ход работы

### Теория

SELinux (Security-Enhanced Linux) — реализация системы мандатного управления доступа для систем Linux. Может работать параллельно с классической избирательной системой контроля доступа. Основная цель SELinux — дать системным администраторам больше контроля над предоставлением доступа к файлам и процессам.

### Настройка

Установим selinux и все нужные модули.

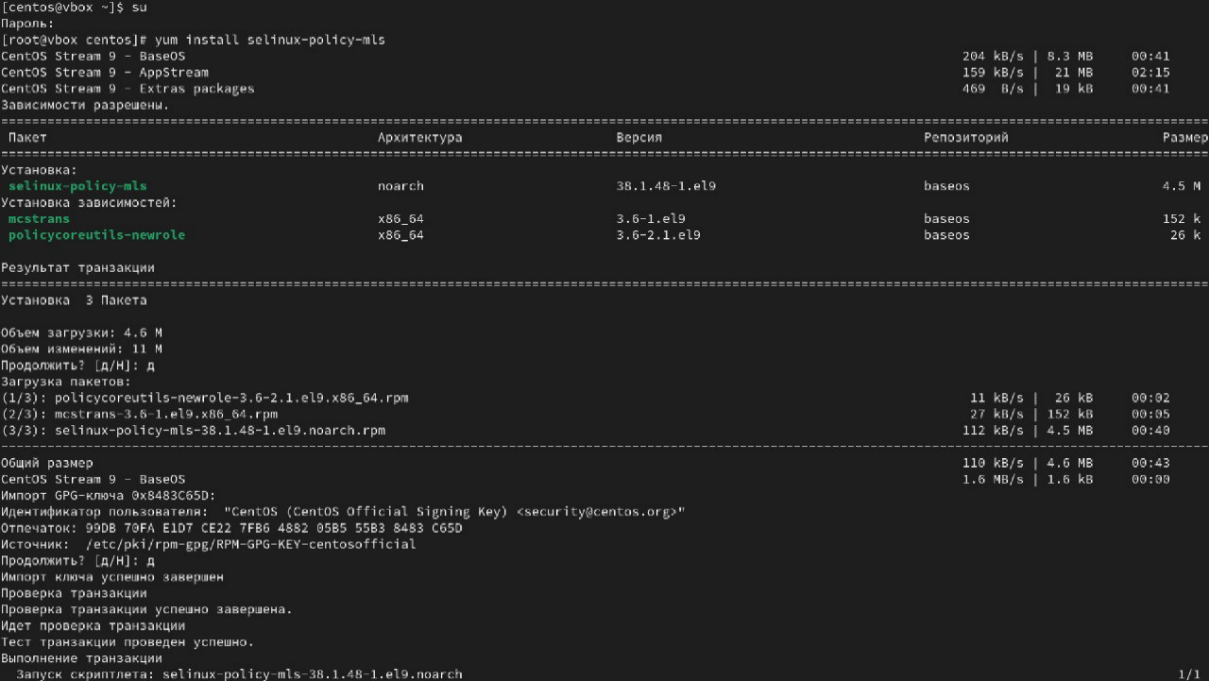


Рисунок 11 – установка selinux

После перезагрузки системы используем команду sestatus, чтобы посмотреть ее вывод.

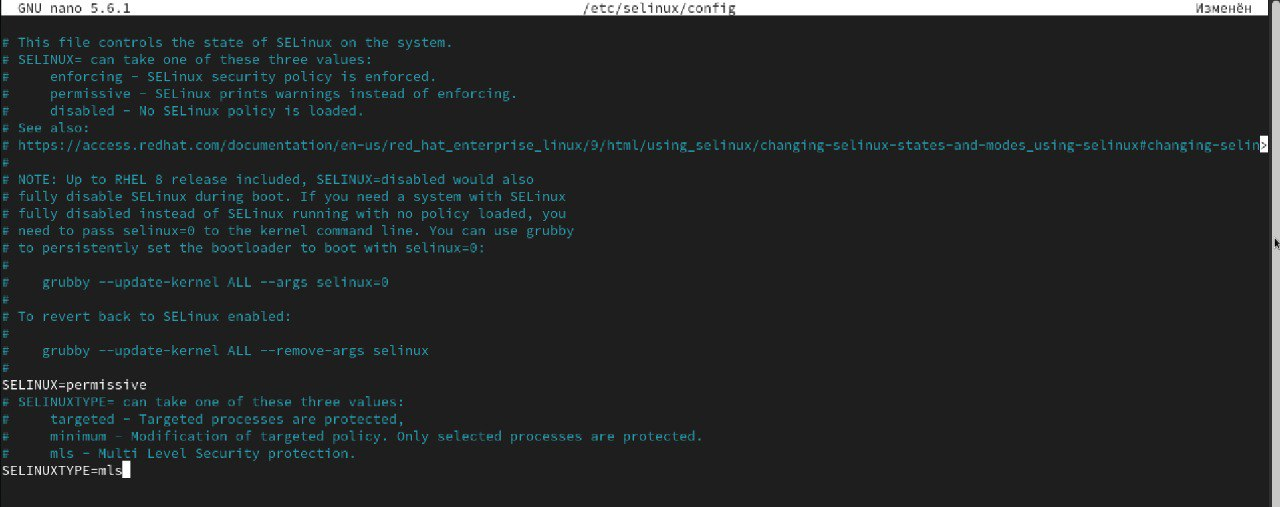


Рисунок 12 - изначальный конфигурация selinux

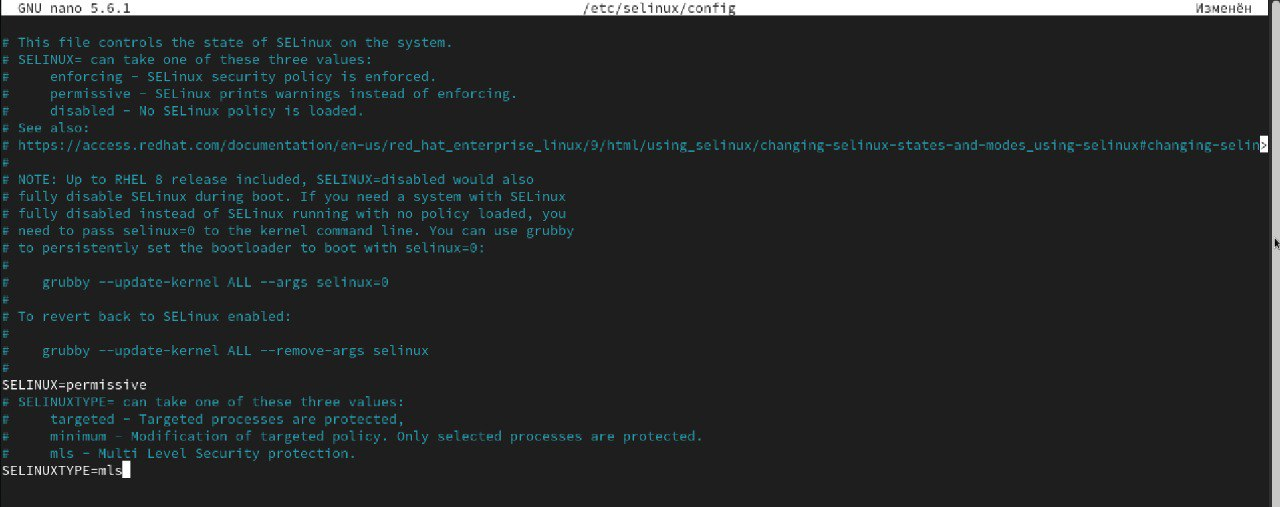


Рисунок 13 - установка нужных значений в конфигурации (permissive, mls)

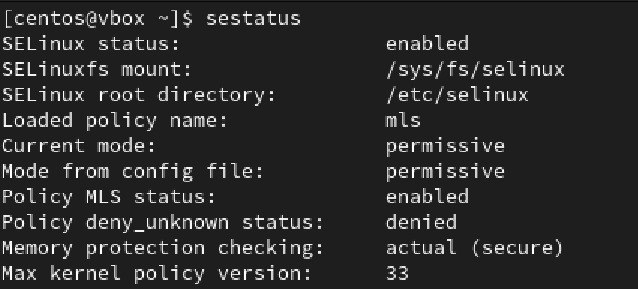


Рисунок 14 – вывод команды sestatus

Посмотрим файл /etc/selinux/targeted/setrans.conf, он отвечает за уровни безопасности.

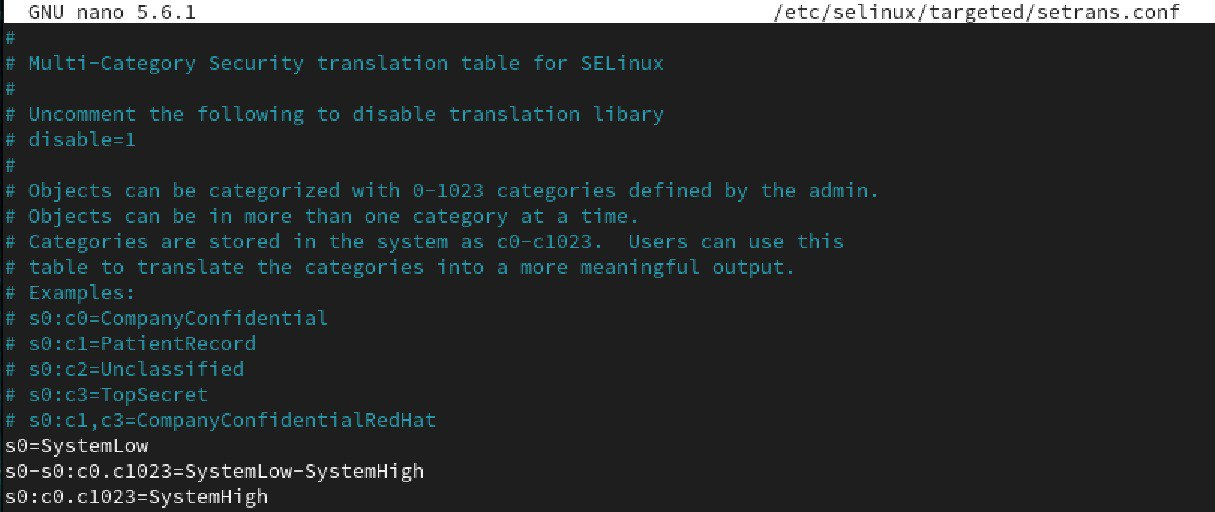


Рисунок 15 - файл /etc/selinux/targeted/setrans.conf

### Демонстрация работы

Создадим пользователей user1 и user2 с ролями staff\_u.

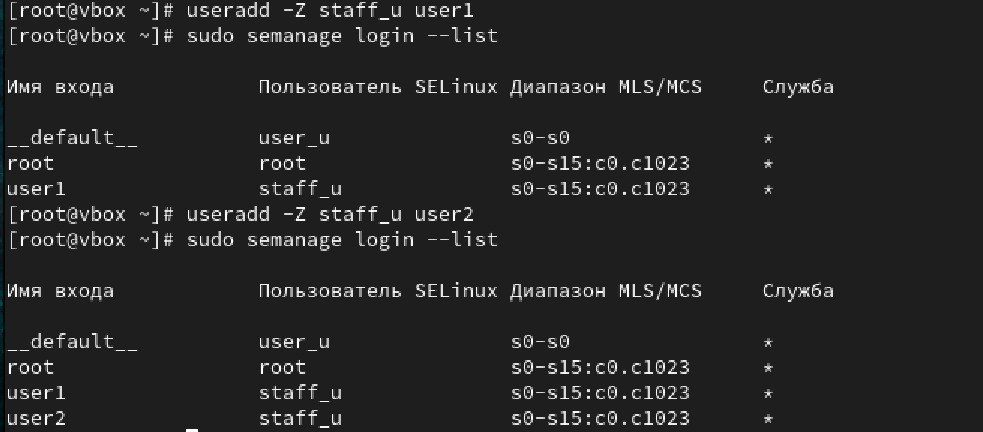


Рисунок 16 – создание пользователей

Назначим первому и второму пользователю уровни доступа s1 и s2 соответственно.

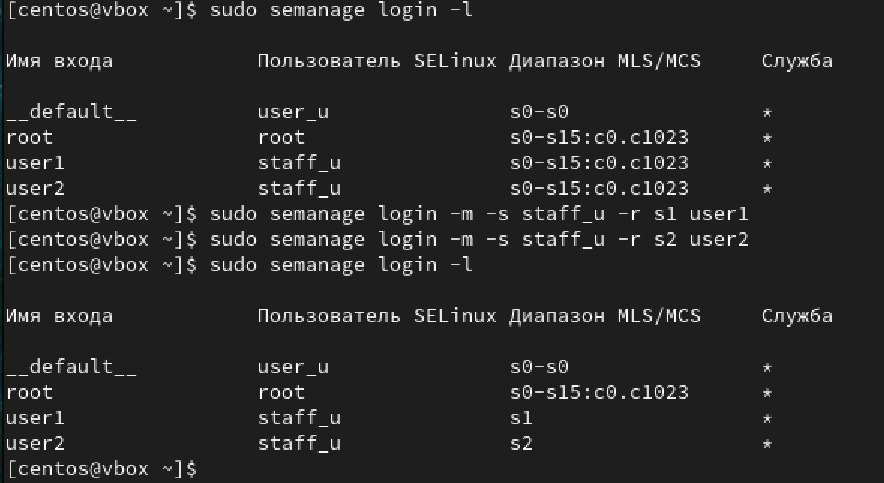


Рисунок 17 – изменение уровня доступа

В папках каждого из пользователей были созданы текстовые файлы, папкам был присвоен соответствующий их владельцам уровень доступа.

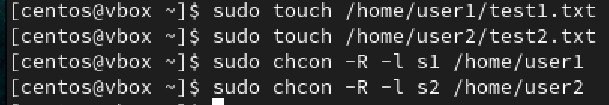


Рисунок 18 – создание файлов

# сложн вариант

## Задание

Придумать и написать свой PAM-модуль

## Ход работы

### Теория

Pluggable Authentication Modules (PAM, подключаемые модули аутентификации) — это набор разделяемых библиотек, которые позволяют интегрировать различные низкоуровневые методы аутентификации в виде единого высокоуровневого API. Это позволяет предоставить единые механизмы для управления, встраивания прикладных программ в процесс аутентификации. Является одной из частей стандартного механизма обеспечения безопасности UNIX-систем.

### PAM-модуль

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <security/pam\_modules.h>

#include <security/pam\_appl.h>

#include <security/pam\_misc.h>

// Список допустимых пользователей

const char \*valid\_users[] = {"user1", "user2", "user3"};

// Проверка существования пользователя

int user\_exists(const char \*user) {

for (int i = 0; i < sizeof(valid\_users) / sizeof(valid\_users[0]); i++) {

if (strcmp(user, valid\_users[i]) == 0) {

return 1; // Пользователь найден

}

}

return 0; // Пользователь не найден

}

PAM\_EXTERN int pam\_sm\_authenticate(pam\_handle\_t \*pamh, int flags, int argc, const char \*\*argv) {

int retval;

const char \*user;

// Получаем имя пользователя

retval = pam\_get\_user(pamh, &user, "Введите имя пользователя: ");

if (retval != PAM\_SUCCESS || user == NULL) {

fprintf(stderr, "Ошибка получения имени пользователя.\n");

return PAM\_AUTH\_ERR;

}

// Проверяем, существует ли пользователь

if (!user\_exists(user)) {

fprintf(stderr, "Пользователь '%s' не найден в системе.\n", user);

return PAM\_AUTH\_ERR;

}

// Задаём вопрос с фразой

printf("Привет, %s! Введите контрольную фразу для доступа: \n", user);

char answer[256];

// Чтение строки, включая пробелы

if (fgets(answer, sizeof(answer), stdin) == NULL) {

printf("Ошибка при вводе.\n");

return PAM\_AUTH\_ERR;

}

// Убираем символ новой строки в конце строки

answer[strcspn(answer, "\n")] = '\0';

// Проверяем правильность ответа

if (strcmp(answer, "лабораторнаяработа8") == 0) {

printf("Ответ верный. Доступ разрешён.\n");

return PAM\_SUCCESS;

} else {

printf("Ответ неверный. Доступ запрещён.\n");

return PAM\_AUTH\_ERR;

}

}

PAM\_EXTERN int pam\_sm\_setcred(pam\_handle\_t \*pamh, int flags, int argc, const char \*\*argv) {

return PAM\_SUCCESS;

}

Листинг 1 - PAM-модуль

#include <security/pam\_appl.h>

#include <security/pam\_misc.h>

#include <stdio.h>

#include <stdlib.h>

// PAM-конверсия для взаимодействия с пользователем

static struct pam\_conv conv = {

misc\_conv,

NULL

};

int main() {

pam\_handle\_t \*pamh = NULL;

int retval;

const char \*username = NULL;

// Инициализация PAM

retval = pam\_start("pam\_test", username, &conv, &pamh);

if (retval != PAM\_SUCCESS) {

fprintf(stderr, "Ошибка инициализации PAM: %s\n", pam\_strerror(pamh, retval));

return EXIT\_FAILURE;

}

// Аутентификация пользователя

retval = pam\_authenticate(pamh, 0);

if (retval == PAM\_SUCCESS) {

printf("Аутентификация успешна.\n");

} else {

printf("Ошибка аутентификации: %s\n", pam\_strerror(pamh, retval));

}

// Завершение PAM-сессии

if (pam\_end(pamh, retval) != PAM\_SUCCESS) {

fprintf(stderr, "Ошибка завершения PAM: %s\n", pam\_strerror(pamh, retval));

return EXIT\_FAILURE;

}

return (retval == PAM\_SUCCESS) ? EXIT\_SUCCESS : EXIT\_FAILURE;

}

Листинг 2 – тест модуля

Пример работы данного модуля.

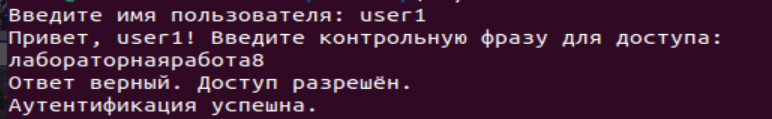


Рисунок – пример работы

Заключение

В ходе данной лабораторной работы мы познакомились программами контроля доступа, а именно Apparmor и selinux. Обе программы были настроены и протестированы. Также был придуман и написан свой PAM-модуль. Это позволило закрепить полученные знания и навыки. В ходе выполнения лабораторной работы были выполнены все задачи и достигнуты поставленные цели.

Список использованных источников

1. Эндрю Таненбаум Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с.