

**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**  
«Настройка AppArmor и SELinux»

**Выполнили:**

Бардышев Артём Антонович, студент группы N3246

---

(подпись)

Суханкулиев Мухаммет, студент группы N3246

---

(подпись)

Шегай Станислав Дмитриевич, студент группы N3246

---

(подпись)

**Проверил:**

Савков Сергей Витальевич,  
инженер

---

(отметка о выполнении)

---

(подпись)

Санкт-Петербург  
2024 г.

## СОДЕРЖАНИЕ

Введение .....	3
1     Настройка AppArmor для мониторинга сложного приложения .....	4
1.1    Установка, настройка профиля AppArmor для mousepad .....	4
1.2    Проверка работоспособности профиля для mousepad .....	4
1.3    Настройка профиля AppArmor для Apache .....	5
1.4    Проверка работоспособности профиля для Apache .....	6
2     Настройка SELinux в режиме мандатного доступа .....	7
2.1    Настройка SELinux .....	7
2.2    Проверка SELinux .....	7
3     PAM-модуль .....	9
3.1    Теория .....	9
3.2    Реализация PAM-модуля с тестом .....	9
3.3    Сборка и настройка .....	11
3.4    Тестирование .....	11
Заключение .....	12
Список использованных источников .....	13

## **ВВЕДЕНИЕ**

### **Цель работы:**

1. Настроить Apparmor для мониторинга сложного приложения и продемонстрировать его работу при ограниченных правах (оконное приложение или веб-сервер)
2. Настроить selinux в режиме мандатного доступа (CentOS и др.) и продемонстрировать работу в двухуровневой модели.

### **Усиленный вариант:**

- Придумать и написать свой LSM-модуль (сложная авторизация действий)

# 1 НАСТРОЙКА APPARMOR ДЛЯ МОНИТОРИНГА СЛОЖНОГО ПРИЛОЖЕНИЯ

**AppArmor** (Application Armor) — это система управления доступом в ядре Linux, предназначенная для ограничения действий приложений, чтобы предотвратить их ненамеренные или вредоносные действия. AppArmor использует концепцию **профилей**, которые описывают, какие ресурсы (файлы, устройства, порты и т.д.) могут быть использованы приложением и какие действия с ними разрешены.

## 1.1 Установка, настройка профиля AppArmor для mousepad

```
sudo apt-get install apparmor apparmor-utils apparmor-profiles
sudo systemctl enable apparmor
sudo systemctl start apparmor
```

Создадим профиль для mousepad

```
sudo aa-genprof mousepad
```

Команда **aa-genprof** создаёт профиль приложения на основе его поведения, записывая используемые файлы и ресурсы.

Для активации режима принуждения используем:

```
sudo aa-enforce /etc/apparmor.d/usr.bin.mousepad
```

В этом режиме приложение ограничено в доступе к ресурсам, и нарушения профиля блокируются. Режим предупреждения (**aa-complain**) фиксирует нарушения без блокировки, полезен для тестирования.

Изменим настройки профиля вручную

```
sudo nano /etc/apparmor.d/usr.bin.mousepad
```

Добавим строки

```
/** rw,
deny /home/kali/Desktop/lab8/** r,
```

Перезапускаем профиль

```
sudo apparmor_parser -r /etc/apparmor.d/usr.bin.mousepad
```

Чтобы применить изменения после ручного редактирования профиля, используем:

Команда **aa-status** отображает текущее состояние профилей и их режимы.

## 1.2 Проверка работоспособности профиля для mousepad

Пробуем запустить файл из /home/kali/Desktop/lab8/ с помощью mousepad.

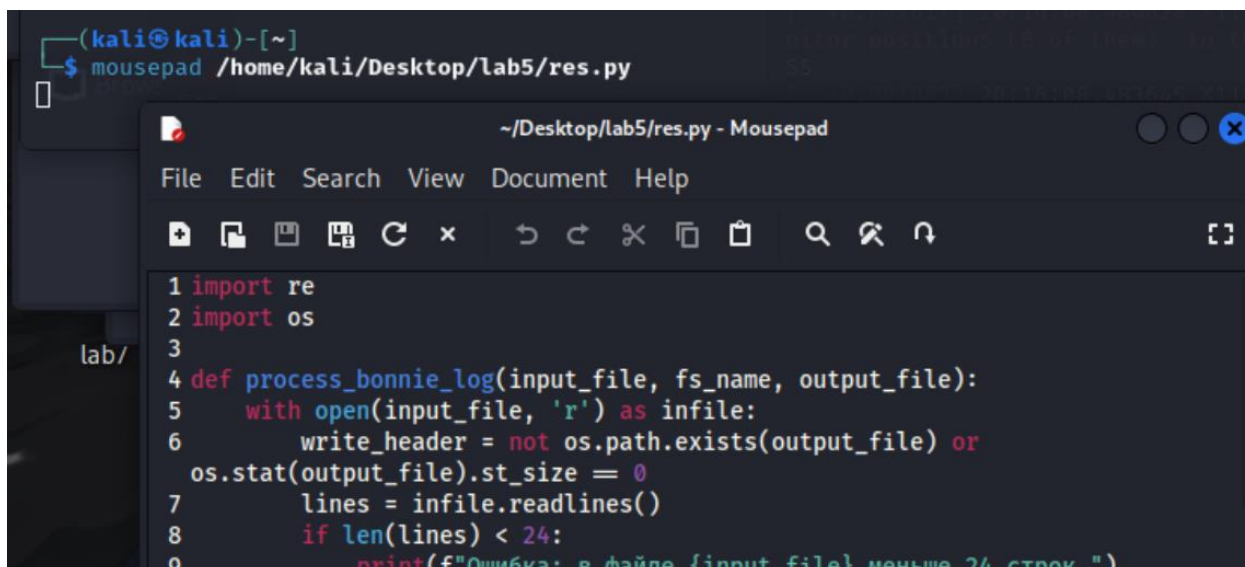


Рисунок 1 – Попытка открыть файл в папке /lab5/

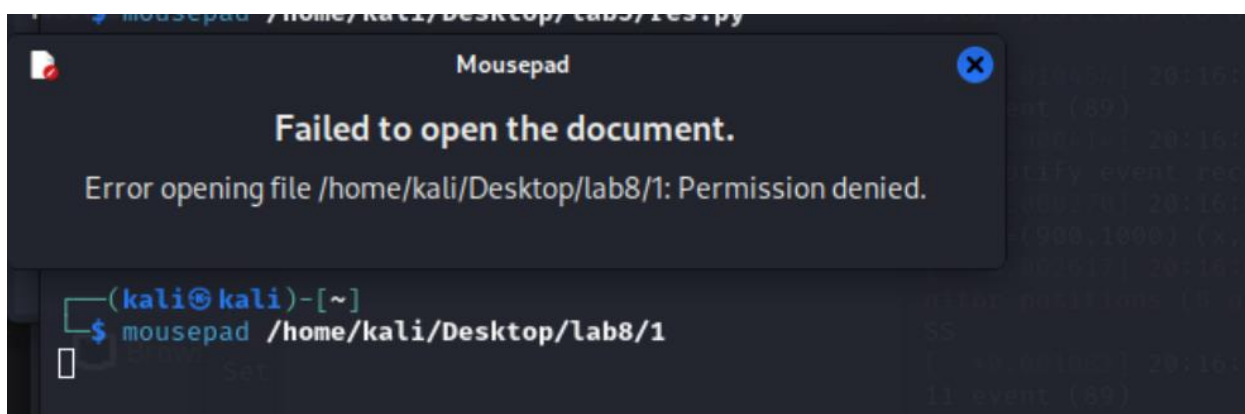


Рисунок 2 – Попытка открыть файл в папке /lab8/

### 1.3 Настройка профиля AppArmor для Apache

**Apache2** — популярный веб-сервер с открытым исходным кодом, используемый для обслуживания статических и динамических веб-страниц и запуска веб-приложений. Он поддерживает виртуальные хосты, позволяя запускать несколько сайтов на одном сервере с отдельными конфигурациями для каждого домена. Для повышения безопасности Apache2 часто используется вместе с SELinux или AppArmor, которые ограничивают доступ к системным ресурсам и предотвращают нежелательные действия.

```

sudo aa-genprof apache2
sudo aa-complain /etc/apparmor.d/usr.sbin.apache2
sudo nano /etc/apparmor.d/usr.sbin.apache2
    deny /home/* r,
    /var/www/* r,
sudo apparmor_parser -r /etc/apparmor.d/usr.sbin.apache2
sudo systemctl restart apache2

```

## 1.4 Проверка работоспособности профиля для Apache

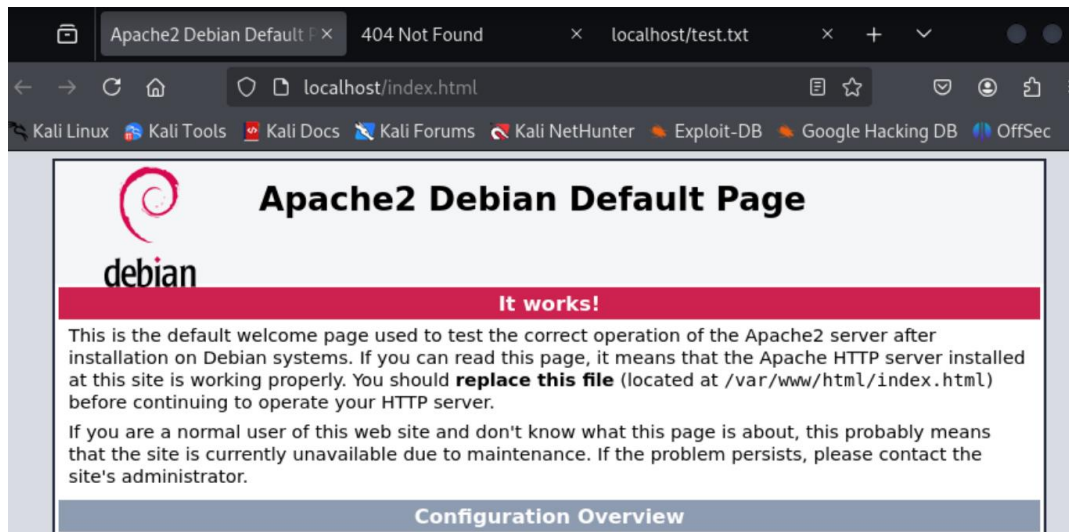


Рисунок 3 – <http://localhost/index.html>

```
sudo echo "Test" > /home/kali/test.txt
```

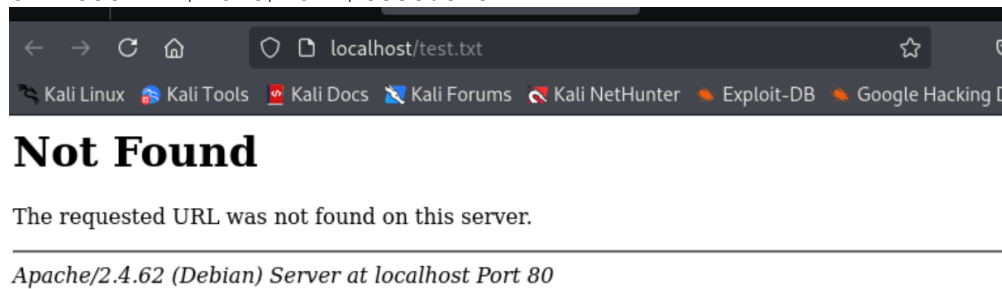


Рисунок 4 – <http://localhost/test.txt>

```
sudo echo "This is a test file" > /var/www/html/test.txt
```

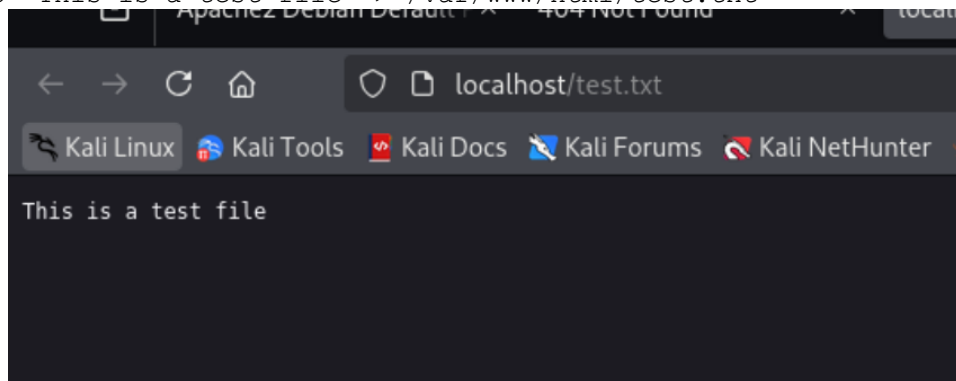


Рисунок 5 – <http://localhost/test.txt>

## 2 НАСТРОЙКА SELINUX В РЕЖИМЕ МАНДАТНОГО ДОСТУПА

**SELinux (Security-Enhanced Linux)** — это механизм управления доступом в ядре Linux, который предоставляет реализацию политики безопасности. SELinux использует концепцию **мандатного контроля доступа (MAC)**, который накладывает ограничения на то, какие действия могут выполняться различными объектами системы (например, пользователями, процессами, файлами).

### 2.1 Настройка SELinux

```
sudo yum install selinux-policy selinux-policy-mls
```

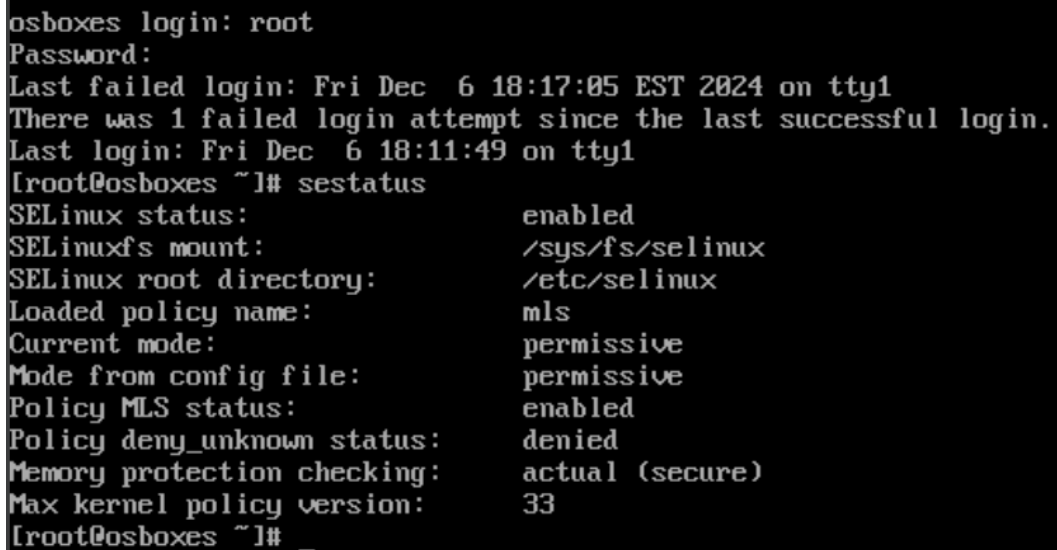
После установки пакетов нужно настроить SELinux для работы в режиме мандатного доступа (MLS – Multi-Level-Security):

```
sudo vi /etc/selinux/config
```

Внесем следующие изменения:

```
SELINUX=permissive  
SELINUXTYPE=mls
```

```
sudo reboot
```



```
osboxes login: root  
Password:  
Last failed login: Fri Dec 6 18:17:05 EST 2024 on tty1  
There was 1 failed login attempt since the last successful login.  
Last login: Fri Dec 6 18:11:49 on tty1  
[root@osboxes ~]# sestatus  
SELinux status:                enabled  
SELinuxfs mount:                /sys/fs/selinux  
SELinux root directory:         /etc/selinux  
Loaded policy name:              mls  
Current mode:                   permissive  
Mode from config file:          permissive  
Policy MLS status:              enabled  
Policy deny_unknown status:     denied  
Memory protection checking:     actual (secure)  
Max kernel policy version:      33  
[root@osboxes ~]# _
```

Рисунок 6 – sudo sestatus

В файле /etc/selinux/mls/setrans.conf можем заметить, что уже определены два уровня безопасности: Unclassified (s1) и Secret (s2).

### 2.2 Проверка SELinux

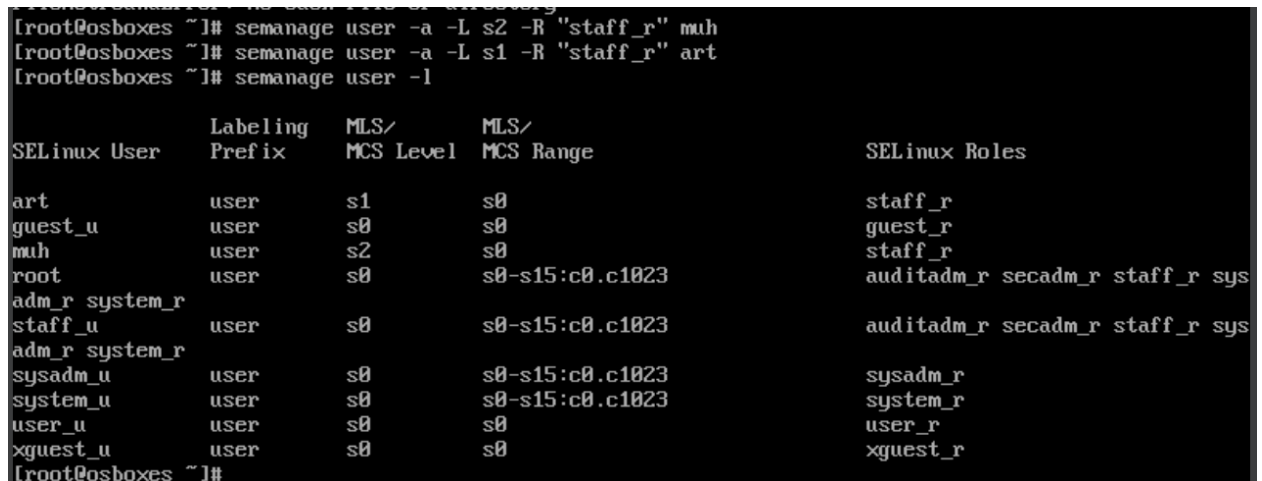
Добавление пользователей:

```
useradd muh  
useradd art  
passwd muh
```

```

qwe
passwd art
123
sudo semanage user -a -L s2 -R "staff_r" muh
sudo semanage user -a -L s1 -R "staff_r" art
-R "staff_r" — назначает роль staff_r, которая позволяет пользователям выполнять
стандартные операции.

```



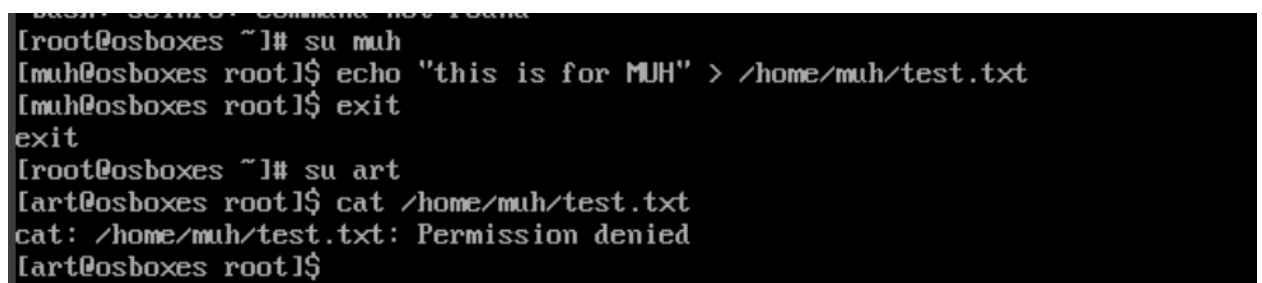
```

[root@osboxes ~]# semanage user -l
[root@osboxes ~]# semanage user -a -L s1 -R "staff_r" art
[root@osboxes ~]# semanage user -l

```

SELinux User	Labeling Prefix	MLS/MCS Level	MLS/MCS Range	SELinux Roles
art	user	s1	s0	staff_r
guest_u	user	s0	s0	guest_r
muh	user	s2	s0	staff_r
root	user	s0	s0-s15:c0.c1023	auditadm_r secadm_r staff_r sys
adm_r system_r				
staff_u	user	s0	s0-s15:c0.c1023	auditadm_r secadm_r staff_r sys
adm_r system_r				
sysadm_u	user	s0	s0-s15:c0.c1023	sysadm_r
system_u	user	s0	s0-s15:c0.c1023	system_r
user_u	user	s0	s0	user_r
xguest_u	user	s0	s0	xguest_r

Рисунок 7 – semanage user -l

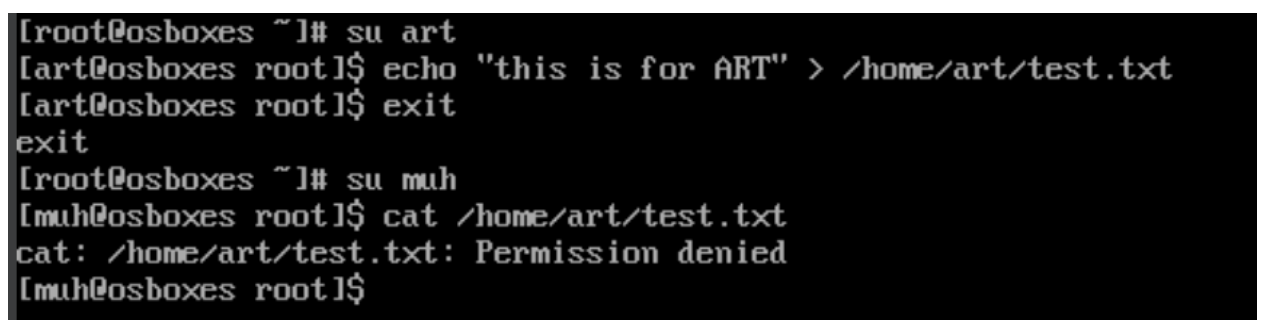


```

[root@osboxes ~]# su muh
[muh@osboxes root]$ echo "this is for MUH" > /home/muh/test.txt
[muh@osboxes root]$ exit
exit
[root@osboxes ~]# su art
[art@osboxes root]$ cat /home/muh/test.txt
cat: /home/muh/test.txt: Permission denied
[art@osboxes root]$

```

Рисунок 8 – Попытка доступа art к файлу muh’a



```

[root@osboxes ~]# su art
[art@osboxes root]$ echo "this is for ART" > /home/art/test.txt
[art@osboxes root]$ exit
exit
[root@osboxes ~]# su muh
[muh@osboxes root]$ cat /home/art/test.txt
cat: /home/art/test.txt: Permission denied
[muh@osboxes root]$

```

Рисунок 9 – Попытка доступа muh к файлу art’a

В обоих случаях доступ к файлам других пользователей будет запрещен, что демонстрирует работу SELinux в двухуровневой модели. Пользователь с более низким уровнем (Unclassified) не может получить доступ к файлам с более высоким уровнем (Secret), и наоборот.



## 3 ПАМ-МОДУЛЬ

```
sudo apt-get install libpam0g-dev build-essential
```

### 3.1 Теория

**ПАМ** (Pluggable Authentication Module) — это архитектура, используемая для добавления гибкости в аутентификацию и управление доступом в операционных системах, таких как Linux. ПАМ позволяет системным администраторам настраивать способы аутентификации пользователей без необходимости изменять приложения, использующие систему аутентификации. Вместо того чтобы приложения напрямую взаимодействовали с базами данных или механизмами аутентификации, они могут использовать ПАМ для проверки учетных данных. ПАМ-модуль — это динамическая библиотека, которая реализует определённый интерфейс для аутентификации пользователей. Он взаимодействует с системой в рамках настроек ПАМ.

**Модули** — это динамические библиотеки, реализующие определенные политики аутентификации. Каждый модуль выполняет одну функцию (например, проверка пароля, учетный контроль, создание учетных данных).

**Конфигурация ПАМ** — конфигурационные файлы в `/etc/pam.d/`, в которых определяются, какие модули должны быть использованы для конкретных действий (например, аутентификация, создание учетных данных, контроль доступа).

### 3.2 Реализация ПАМ-модуля с тестом

`pam.c:`

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <security/pam_appl.h>
#include <security/pam_modules.h>
#include <time.h>
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) {
    return PAM_SUCCESS;
}
PAM_EXTERN int pam_sm_authenticate(pam_handle_t *pamh, int flags, int argc,
const char **argv) {
    int retval;
    const char *username;
    // Получаем имя пользователя
    retval = pam_get_user(pamh, &username, "Username: ");
```

```

    if (retval != PAM_SUCCESS) {
        return retval;
    }
    printf("Welcome %s\n", username);
    // Проверяем имя пользователя
    if (strcmp(username, "kali") != 0) {
        return PAM_AUTH_ERR;
    }
    // Если имя пользователя корректное, генерируем случайные числа для расчета
    пароля
    long x1, x2, x3, x4, y;
    time_t mytime;
    struct tm *mytm;
    mytime = time(0);
    mytm = localtime(&mytime);
    srand(mytime + (long int)username);
    x1 = random() % 30;
    x2 = random() % 30;
    x3 = random() % 30;
    x4 = random() % 30;
    // Вычисляем значение пароля y по формуле
    y = x1 * x2 * x3 * x4;
    // Печатаем случайные числа, чтобы пользователь мог ввести правильный пароль
    printf("x1 = %ld, x2 = %ld, x3 = %ld, x4 = %ld\n", x1, x2, x3, x4);
    printf("Введите пароль: ");
    long userpwd;
    scanf("%ld", &userpwd);
    // Проверяем введенный пароль
    if (userpwd != y) {
        return PAM_AUTH_ERR; // Ошибка аутентификации
    }
    return PAM_SUCCESS; // Успешная аутентификация
}

```

#### tester.c:

```

#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;
    if (argc != 2) {
        printf("Usage: app [username]\n");
        exit(1);
    }
    const char *user = argv[1];
    // Начинаем аутентификацию с нашим PAM-модулем
    retval = pam_start("check_user", user, &conv, &pamh);
    if (retval == PAM_SUCCESS) {
        printf("Credentials accepted.\n");
        retval = pam_authenticate(pamh, 0); // Проверка пароля
    }
    if (retval == PAM_SUCCESS) {
        printf("Account is valid.\n");
        retval = pam_acct_mgmt(pamh, 0); // Проверка доступа к аккаунту
    }
    if (retval == PAM_SUCCESS) {
        printf("Authenticated\n");
    }
}

```

```

    } else {
        printf("Not Authenticated\n");
    }
    if (pam_end(pamh, retval) != PAM_SUCCESS) {
        printf("check_user: failed to release authenticator\n");
        exit(1);
    }
    return retval == PAM_SUCCESS ? 0 : 1;
}

```

### 3.3 Сборка и настройка

```
gcc -fPIC -shared -o pam_example.so pam.c -lpam
```

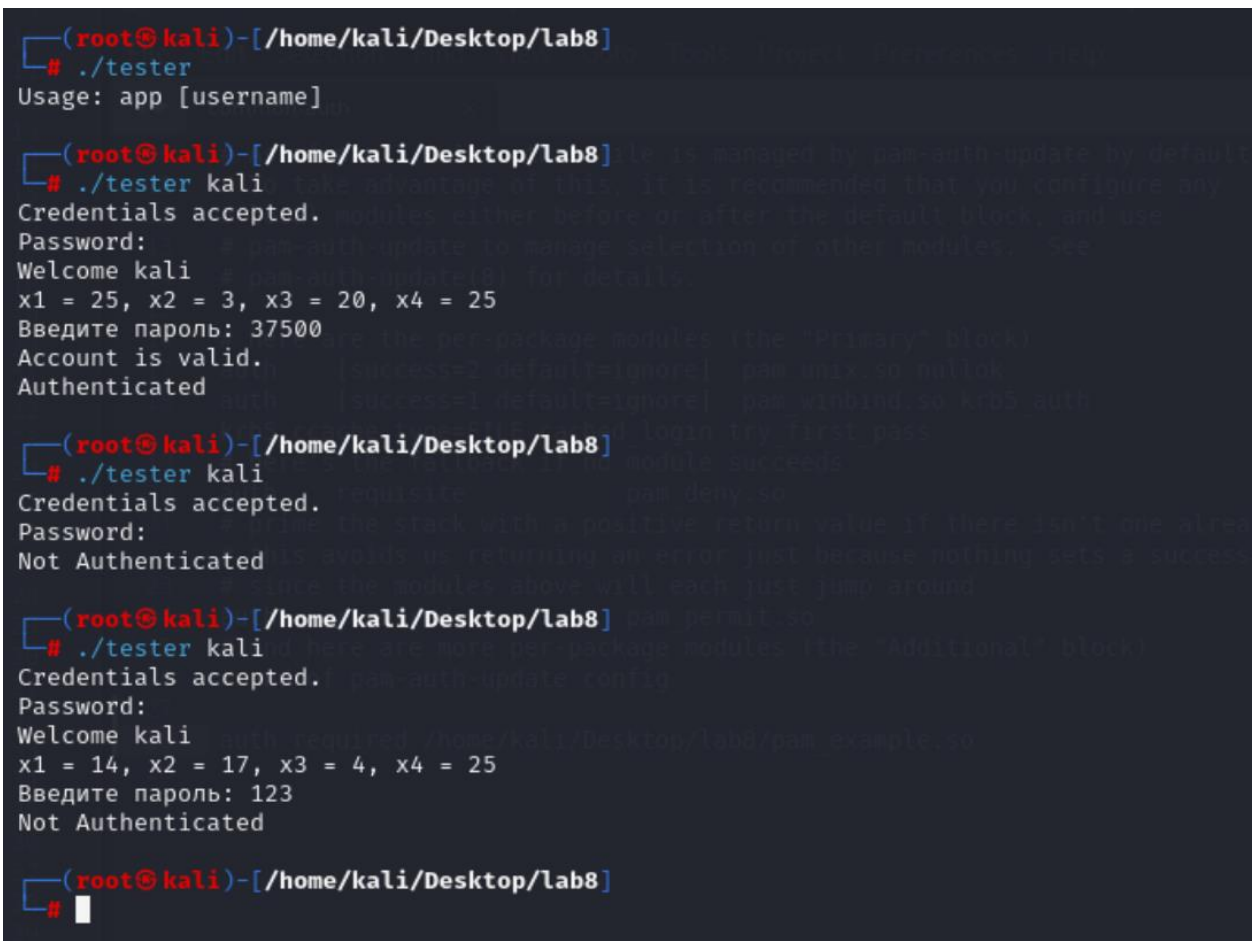
После сборки имеем `pam_example.so`

В файл `/etc/pam.d/common-auth` добавим строку:

```
auth required /home/kali/Desktop/lab8/pam_example.so
```

```
gcc -o tester tester.c -lpam -lpam_misc
```

### 3.4 Тестирование



```

(root@kali)-[/home/kali/Desktop/lab8]
# ./tester
Usage: app [username]

(root@kali)-[/home/kali/Desktop/lab8]
# ./tester kali
Welcome kali
x1 = 25, x2 = 3, x3 = 20, x4 = 25
Введите пароль: 37500
Account is valid.
Authenticated

(root@kali)-[/home/kali/Desktop/lab8]
# ./tester kali
Credentials accepted.
Password:
Not Authenticated

(root@kali)-[/home/kali/Desktop/lab8]
# ./tester kali
Credentials accepted.
Password:
Welcome kali
x1 = 14, x2 = 17, x3 = 4, x4 = 25
Введите пароль: 123
Not Authenticated

(root@kali)-[/home/kali/Desktop/lab8]
#

```

Рисунок 10 – Тестирование PAM-модуля

## ЗАКЛЮЧЕНИЕ

В ходе выполнения работы были успешно настроены и протестированы различные системы безопасности в Linux, включая AppArmor, SELinux и PAM-модули.

1. **AppArmor** был настроен для мониторинга и ограничения действий приложений, таких как `mousepad` и `Apache2`. В результате использования профилей в режиме принуждения удалось ограничить доступ приложений к файлам и ресурсам системы, что продемонстрировало его эффективность в повышении безопасности.
2. **SELinux** был настроен для работы в режиме мандатного доступа (MLS), что обеспечило двухуровневую модель безопасности. Мы проверили ограничения доступа между пользователями с разными уровнями безопасности, подтверждая его способность предотвращать несанкционированный доступ.
3. **PAM-модуль** был разработан для сложной аутентификации пользователей с использованием генерации случайных чисел для пароля. Этот модуль был интегрирован в систему и успешно протестирован, демонстрируя возможность гибкой настройки аутентификации с применением сложных механизмов.

В общем и целом, настройка и тестирование этих механизмов продемонстрировали их важность для обеспечения безопасности и защиты системы от нежелательных действий.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. [OpenNET: статья - Как работает PAM \(Pluggable Authentication Modules\) \(pam auth linux security howto\)](#)
2. [SELinux – описание и особенности работы с системой. Часть 1 / Хабр](#)
3. [DefconRU | Сообщество специалистов в области информационной безопасности.](#)
4. [AppArmor | Русскоязычная документация по Ubuntu](#)
5. [Настройка Apparmor в Ubuntu - Losst](#)
6. [Ubuntu Manpage: apparmor.d - syntax of security profiles for AppArmor.](#)
7. [Пишем модуль безопасности Linux / Хабр](#)