**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»**

**Факультет безопасности информационных технологий**

**Дисциплина:**

«Операционные системы»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №8**

**Выполнили:**

Чу Ван Доан - N3247

Нгуен Тхе Вьет - N3247

Доан Тхи Хоай Тхыонг - N3245

**Проверил:**

Савков Сергей Витальевич

Санкт-Петербург

2023г.

**Запросы**

1. Простой вариант (обе):

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

2. Сложный вариант (или):

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

I) Простой вариант:

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

* AppArmor — это модуль безопасности Linux, реализующий обязательный контроль доступа на основе имени.
* AppArmor устанавливается и загружается по умолчанию. Он использует профили приложения, чтобы определить, какие файлы и разрешения требуются приложению. Некоторые пакеты будут устанавливать свои собственные профили, а дополнительные профили можно найти в пакете apparmor-profiles.
* Чтобы установить пакет apparmor-profiles из командной строки терминала:

**sudo apt install apparmor-profiles**

* Я создал каталог «data», а затем создал скрипт с именем example2.sh (Цель состоит в том, чтобы продемонстрировать, что мы можем использовать AppArmor, чтобы предотвратить доступ сценария к каким-либо другим путям):

#!/bin/bash

echo "This is an apparmor example."

touch data/sample.txt

echo "File created"

rm data/sample.txt

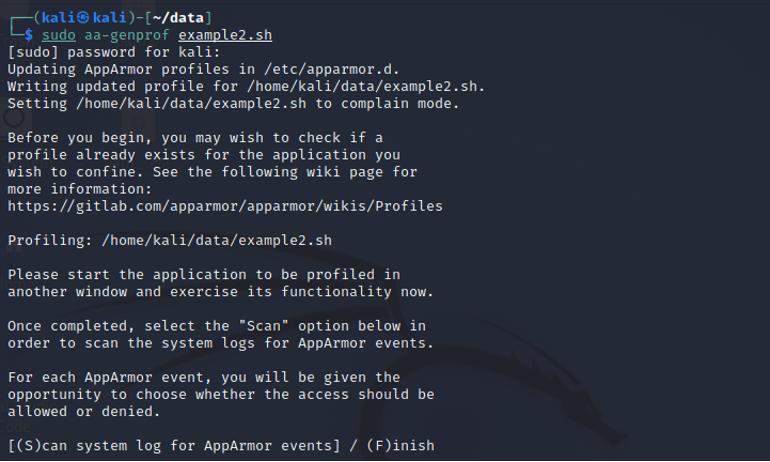
echo "File deleted"

* Обязательно дать ему разрешение на выполнение, а затем запустите его:

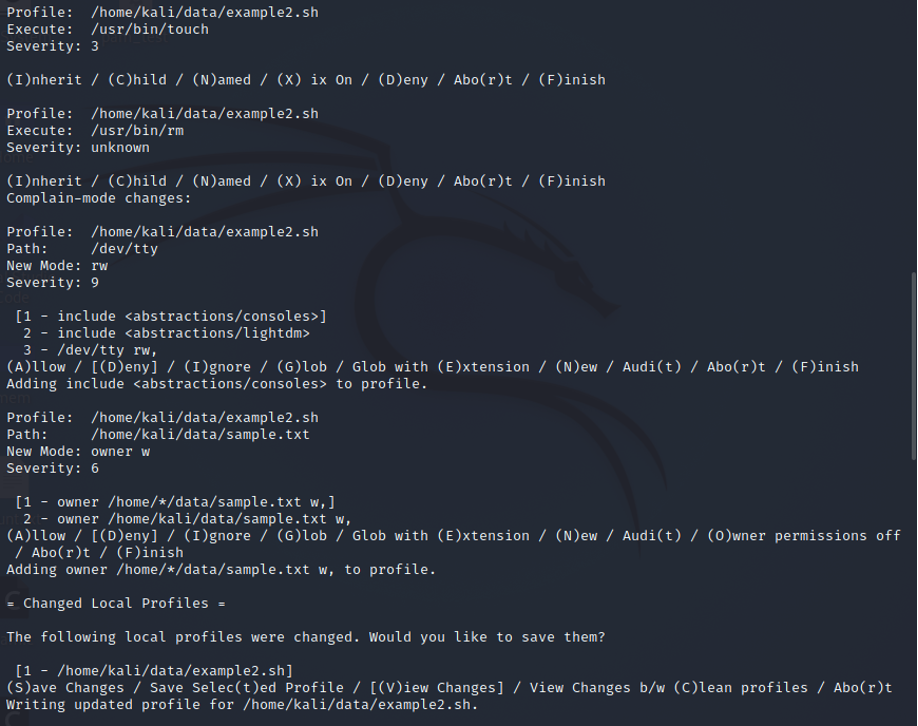


* Есть два основных инструмента, которые мы используем для создания и добавления в профили: aa-genprof и aa-logprof.

**aa - genprof**

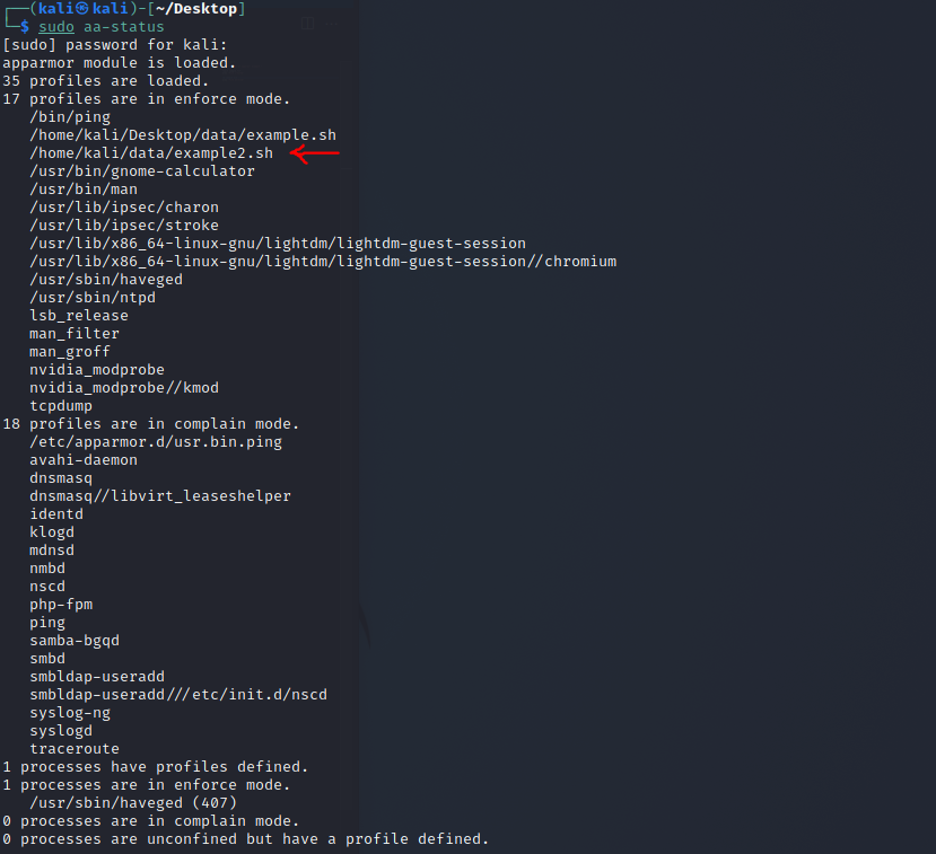
****

* Вот как выглядит запрос от инструментов AppArmor, он говорит вам, что профиль (обычно обозначаемый абсолютным путем к программе) пытается получить доступ к определенному файлу или выполнить его ИЛИ что процессу требуется доступ к определенной возможности из ядро.



**aa-logprof**

* Этот инструмент используется для сканирования журнала аудита на предмет любых событий, которые AppArmor не может сопоставить с существующими правилами в профиле, а затем предложит вам внести изменения.
* убедитесь, что файл находится в режиме enforce.



* Теперь откройте скрипт и отредактируйте его так, чтобы вместо использования каталога данных скрипт пытался читать и записывать в свой собственный каталог:

#!/bin/bash

echo "This is an apparmor example."

touch sample.txt

echo "File created"

rm sample.txt

echo "File deleted"

* Я запустил скрипт, и вы увидели сообщение об ошибке:

This is an apparmor example.

touch: cannot touch ‘sample.txt’: Permission denied

File created

rm: cannot remove ‘sample.txt’: No such file or directory

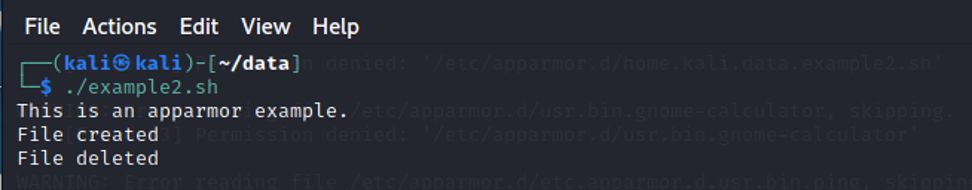
File deleted

* AppArmor заблокировал скрипт от внесения изменений в систему, так как он больше не соответствует правилам, описанным в профиле. Чтобы исправить это, нам нужно обновить профиль AppArmor, поэтому запустите aa-logprof и следуйте инструкциям, чтобы разрешить доступ к новым путям и сохранить изменения:

**aa-logprof**

~$aa-logprof  
Reading log entries from /var/log/audit/audit.log.  
Updating AppArmor profiles in /etc/apparmor.d.  
Enforce-mode changes:  
   
Profile: /home/user/bin/example.sh  
Path: /home/user/bin/sample.txt  
Mode: w  
Severity: unknown  
   
 [1 - /home/user/bin/sample.txt]  
(A)llow / [(D)eny] / (I)gnore / (G)lob / Glob with (E)xtension / (N)ew / Abo(r)t / (F)inish / (M)ore  
Adding /home/user/bin/sample.txt w to profile= Changed Local Profiles =  
   
The following local profiles were changed. Would you like to save them?  
   
 [1 - /home/user/bin/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/user/bin/example.sh.

* Я запустил скрипт, он успешно завершился, как и раньше. И точно так же обновили наш профиль скриптов!



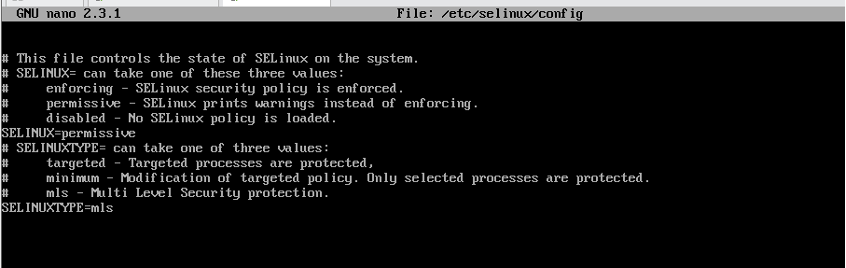
**2) Настроить selinux в режиме мандатного доступа (CentOS и др.) и продемонстрировать работу в двухуровневой модели.**

* Для начало установим на VirtualBox CentOS 7.
* su - root (root mode)
* Установим нужные пакеты командой **$ yum install selinux-policy-mls**

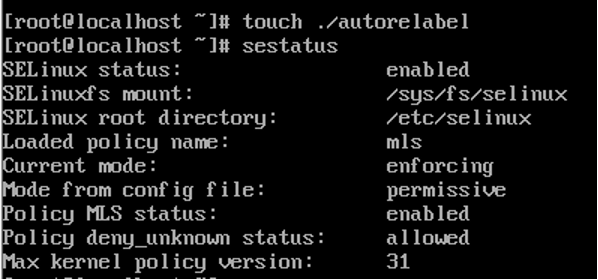
****

****

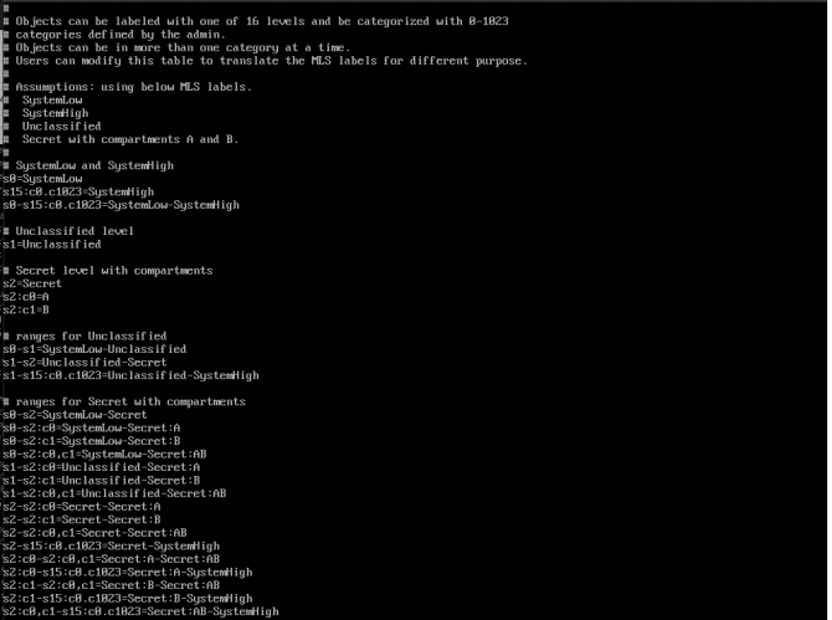
* Заходим в конфигурационный файл командой: **$nano /etc/selinux/config**
* Отключим режим enforcing на permissive и в последней строки на mls



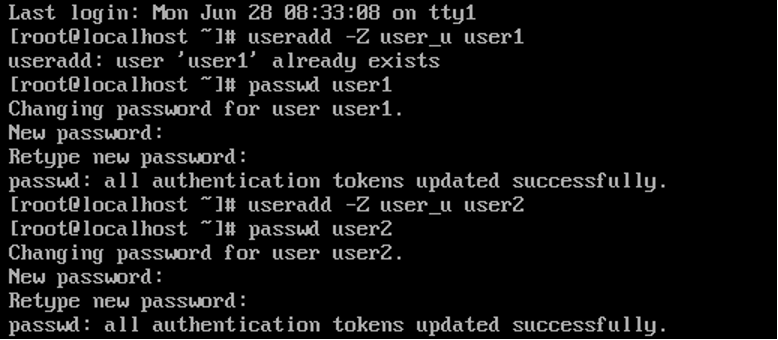
* Для обновления меток в файловой системе создаем пустой файл в autorelabel

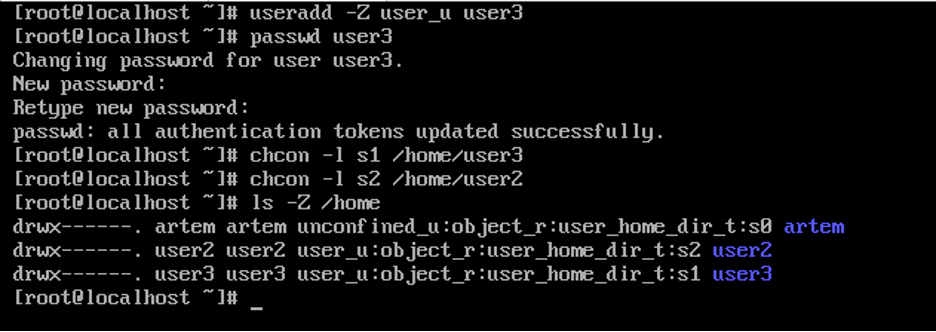


* Посмотрим файл /etc/selinux/targeted/setrans.conf Он отвечает за уровни безопасности. Двухуровневая система уже реализована (Unclassified и Secret) поэтому ничего менять не надо.

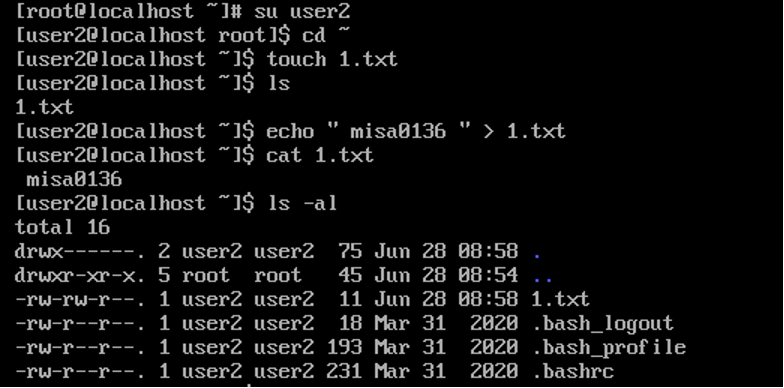


* В selinux используется модель Белла-Лападулы
* Пользователь с уровнем Secret может читать и писать в свои файлы и читать файлы уровня Unclassified.
* Пользователь с уровнем Unclassified может читать и писать в свои файлы, но не может читать файлы уровня Secret, но он может в них писать.
* Для тестирования создадим 2 пользователей и создадим файл, а потом прочитаем их.

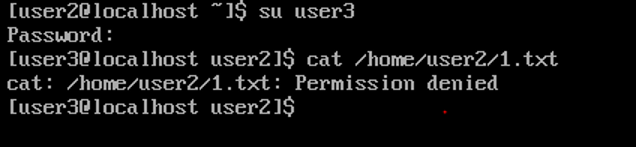




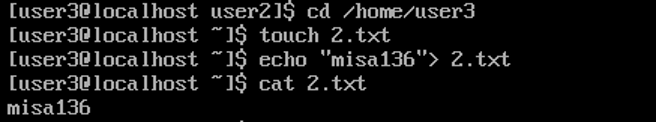
* Теперь можем протестировать. Создадим файл за user2

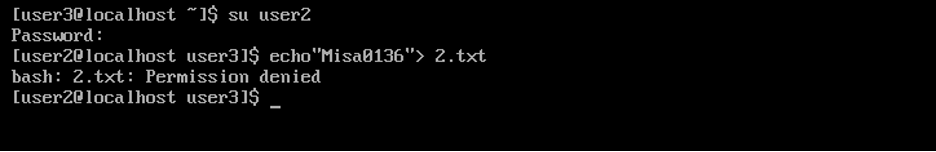


* Прочитаем за user3



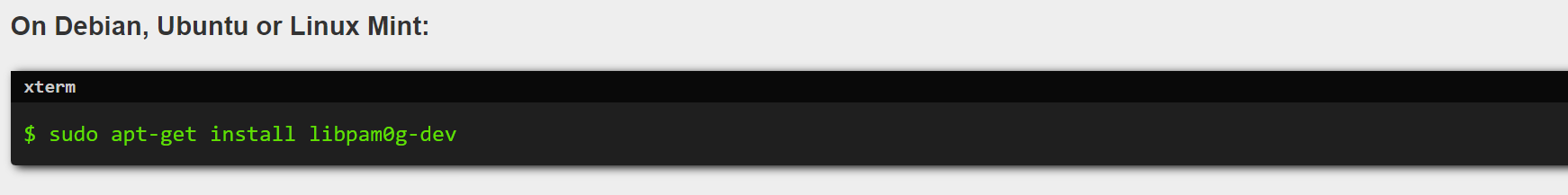
* Как видно не получилось. Попробуем наоборот создадим файл за user1 и записать информацию за user2.





* Тоже не получилось как и должно быть.

II) Сложный вариант:



**pam.c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <security/pam\_appl.h>

#include <security/pam\_modules.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) {

printf("Acct mgmt\n");

return PAM\_SUCCESS;

}

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;

}

**test.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;

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);

if (retval == PAM\_SUCCESS) {

printf("Credentials accepted.\n");

retval = pam\_authenticate(pamh, 0);

}

if (retval == PAM\_SUCCESS) {

printf("Account is valid.\n");

retval = pam\_acct\_mgmt(pamh, 0);

}

if (retval == PAM\_SUCCESS) {

printf("Authenticated\n");

} else {

printf("Not Authenticated\n");

}

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

pamh = NULL;

printf("check\_user: failed to release authenticator\n");

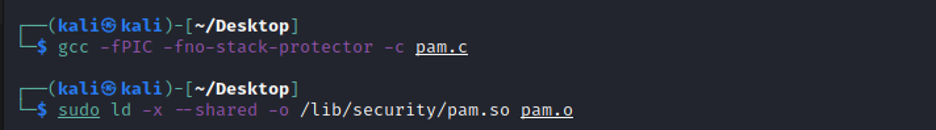
exit(1);

}

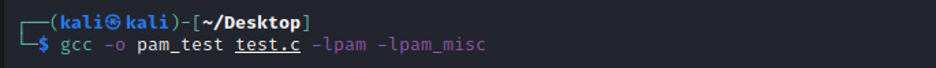
return retval == PAM\_SUCCESS ? 0 : 1;

}

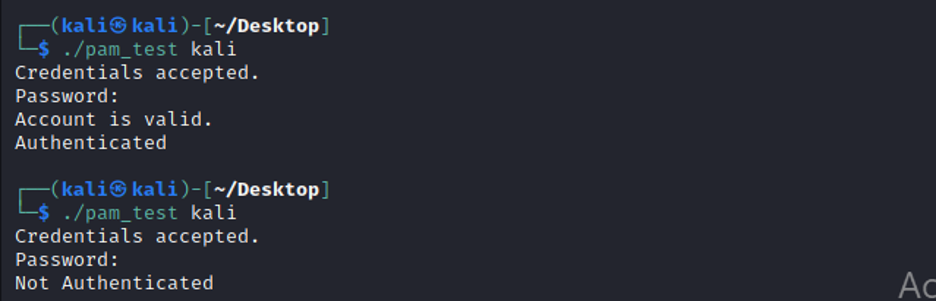
* Создание модуля PAM:



* Первая команда создает объектный файл в текущем каталоге, а вторая связывает его с PAM. Поскольку это разделяемая библиотека, PAM может использовать ее «на лету» без перезапуска.
* Тест сборки



* Запускать и получить результаты:



II) Вывод:

* В ходе выполнения лабораторной работы изучена система Apparmor, в том числе создание профиля для словаря gnome-dictionary. Была проверена его работа при ограниченных правах.
* Есть два основных инструмента, которые мы используем для создания и добавления в профили: aa-genprof и aa-logprof. Первый используется для мониторинга и создания профиля для приложения при его первом запуске (т. е. когда вы впервые создаете профиль), чтобы AppArmor мог узнать, каковы тенденции приложений, и подсказать вам, какое поведение необходимо предпринять. при определенных обстоятельствах. Второй полезен, если у вас есть существующий профиль и вам нужно разрешить/запретить доступ к определенным задачам, которые уже были зарегистрированы в режимах принудительного применения или жалобы.
* Помимо этого, проверена на практике мандатная модель доступа, а именно модель Белла-ЛаПадула, которая функционирует, как и положено.
* Я создал программу для проверки подлинности моих учетных данных. Linux Pluggable Authentication Modules (PAM) — это набор библиотек, который позволяет системному администратору Linux настраивать методы аутентификации пользователей. Он обеспечивает гибкий и централизованный способ переключения методов проверки подлинности для защищенных приложений с помощью файлов конфигурации вместо изменения кода приложения.