Лабораторная работа №5

Дискреционное разграничение прав в Linux. Исследование влияния дополнительных атрибутов

Ильинский Арсений Александрович

Содержание

[Цель работы 1](#_Toc116131190)

[Задание 1](#_Toc116131191)

[Теоретическое введение 1](#_Toc116131192)

[Выполнение лабораторной работы 2](#_Toc116131193)

[Подготовка к работе 2](#_Toc116131194)

[Создание программы 2](#_Toc116131195)

[Работа с е SetUID-битом и SetGID-битом 3](#_Toc116131196)

[Исследование Sticky-бита 6](#_Toc116131197)

[Выводы 9](#_Toc116131198)

[Список литературы 9](#_Toc116131199)

# Цель работы

Изучение механизмов изменения идентификаторов, применения SetUID- и Sticky-битов. Получение практических навыков работы в консоли с дополнительными атрибутами. Рассмотрение работы механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

# Задание

Выполнить задания из лабораторной работы и проанализировать полученные результаты.

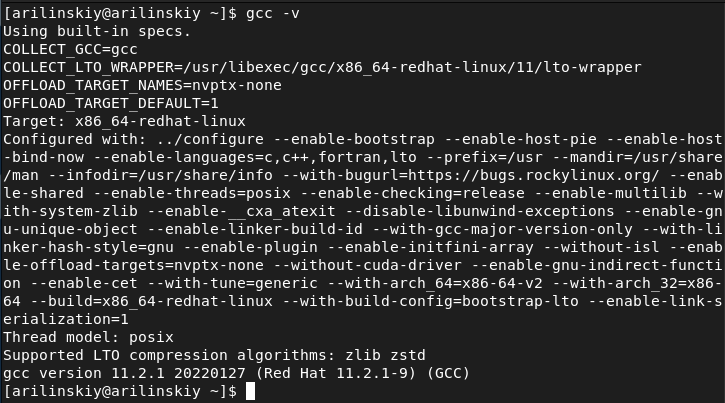
# Теоретическое введение

Для выполнения данной лабораторной нет специальной теории.

# Выполнение лабораторной работы

## Подготовка к работе

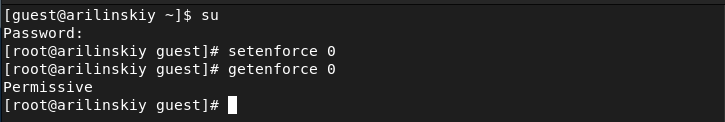
Проверю, установлен ли у меня компилятор gcc командой gcc –v:



Компилятор gcc

У меня он уже установлен.

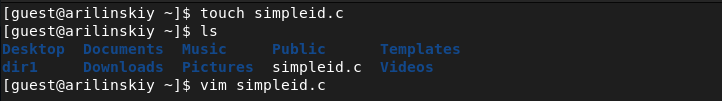
Установил setenforce в 0 и проверил, что данная команда выполнилась:



Отключение SELinux

## Создание программы

Вошел в систему от имени пользователя guest и создал программу simpleid.c



Отключение SELinux

#include <sys/types.h>  
#include <unistd.h>  
#include <stdio.h>  
int  
main ()  
{  
 uid\_t uid = geteuid ();  
 gid\_t gid = getegid ();  
 printf ("uid=%d, gid=%d\n", uid, gid);  
 return 0;  
}

Скомпилирую программу командой *gcc* *simpleid.****c -****o* *simpleid* и запустил ее

Компиляция и выполнение simpleid.c

Компиляция и выполнение simpleid.c

Выполню системную программу id командой *id*. Как видно результат совпадает:

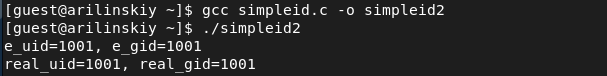
Результат команды id

Результат команды id

Усложню программу, добавив вывод действительных идентификаторов.

#include <sys/types.h>  
#include <unistd.h>  
#include <stdio.h>  
int  
main ()  
{  
 uid\_t real\_uid = getuid ();  
 uid\_t e\_uid = geteuid ();  
 gid\_t real\_gid = getgid ();  
 gid\_t e\_gid = getegid () ;  
 printf ("e\_uid=%d, e\_gid=%d\n", e\_uid, e\_gid);  
 printf ("real\_uid=%d, real\_gid=%d\n", real\_uid,real\_gid);  
 return 0;  
}

Скомпилирую в новый файл simpleid2

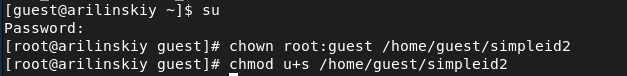


Компиляция и выполнение simpleid2

## Работа с е SetUID-битом и SetGID-битом

От имени суперпользователя выполню команды:

1. *chown root:guest /home/guest/simpleid2*
2. *chmod u+s /home/guest/simpleid2*



Изменение владельца и прав на файл simpleid2

*Пояснение*:

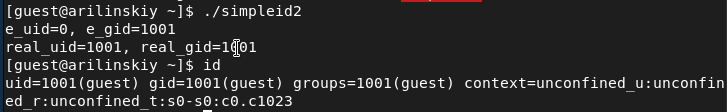
* Команда *chown root:guest /home/guest/simpleid2* меняет владельца файла.
* Команда *chmod* *u+****s /****home/****guest/****simpleid2* меняет права доступа к файлу.

Проверю правильность установки новых атрибутов и смены владельца файла simpleid2 командой: *ls -l simpleid2*

Атрибуты и владелец файла simpleid2

Атрибуты и владелец файла simpleid2

Запущу simpleid2 и id, командами соответственно: *./simpleid2* и *id*



Выполнение simpleid2 и id

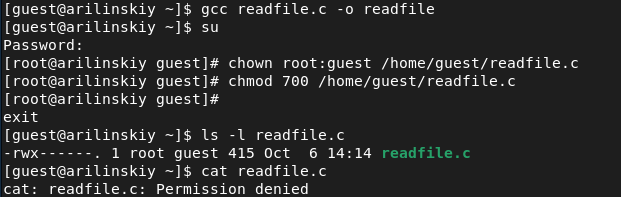
Создам программу readfile.c:

#include <fcntl.h>  
#include <stdio.h>  
#include <sys/stat.h>  
#include <sys/types.h>  
#include <unistd.h>  
int  
main (int argc, char\* argv[])  
{  
 unsigned char buffer[16];  
 size\_t bytes\_read;  
 int i;  
 int fd = open (argv[1], O\_RDONLY);  
 do  
 {  
 bytes\_read = read (fd, buffer, sizeof (buffer));  
 for (i =0; i < bytes\_read; ++i) printf("%c", buffer[i]);  
 }  
 while (bytes\_read == sizeof (buffer));  
 close (fd);  
 return 0;  
}

Программа readfile.c

Программа readfile.c

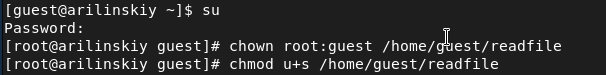
Скомпилирую её командой: *gcc readfile.c -o readfile*, а затем сменю владельца у файла readfile.c и изменю права так, чтобы только суперпользователь (root) мог прочитать его, a guest не мог:



Изменение владельца и прав на файл readfile.c

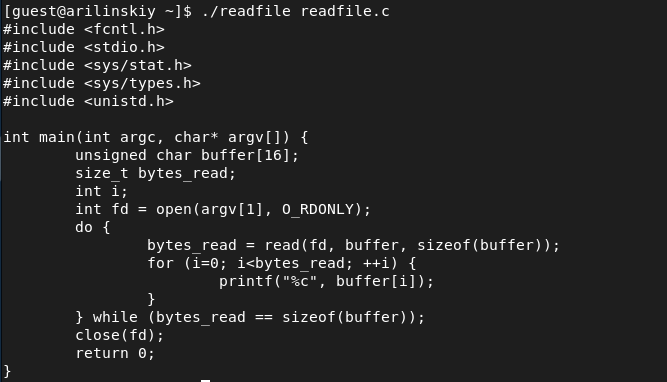
Как видим, пользователь guest не может прочитать файл readfile.c.

Сменю у программы readfile владельца и установлю SetUID-бит:

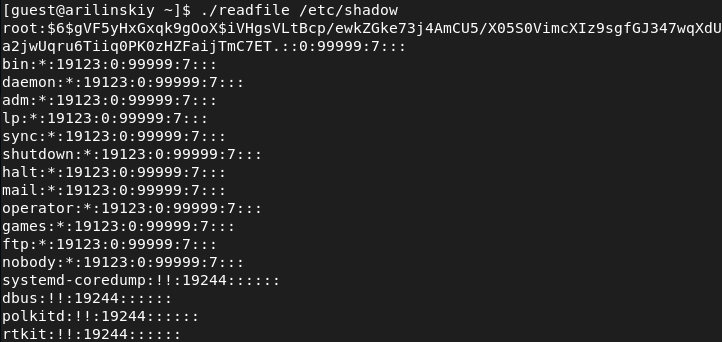


Установка UID бита для readfile.c

Проверю, может ли программа readfile прочитать файл readfile.c:



Выполнение программы для файла readfile.c



Выполнение программы для файла /etc/shadow

Поскольку у программы установлен SetUID-бит, то ей временно предоставляются права владельца файла (суперпользователя). Поэтому программа может прочитать файл с правами доступа только для владельца суперпользователя

## Исследование Sticky-бита

**Проделаю ряд действий**:

1. Выясню, установлен ли атрибут Sticky на директории /tmp, для чего выполню команду:

* *ls -l / | grep tmp*

1. От имени пользователя guest создам файл file01.txt в директории /tmp со словом test:

* *echo “test” > /tmp/file01.txt*

1. Просмотрю атрибуты у только что созданного файла и разрешу чтение и запись для категории пользователей «все остальные»:

* *ls -l /tmp/file01.txt*
* *chmod o+rw /tmp/file01.txt*
* *ls -l /tmp/file01.txt*

1. От пользователя guest2 (не являющегося владельцем) попробую прочитать файл /tmp/file01.txt:

* \*cat /**tmp/**file01.\*\*txt\*

1. От пользователя guest2 попробую дозаписать в файл /tmp/file01.txt слово test2 командой:

* *echo “test2” >> /tmp/file01.txt*.
* Мне удалось выполнить операцию.

1. Проверю содержимое файла командой:

* *cat /tmp/file01.txt*

1. От пользователя guest2 попробую записать в файл /tmp/file01.txt слово test3, стерев при этом всю имеющуюся в файле информацию командой:

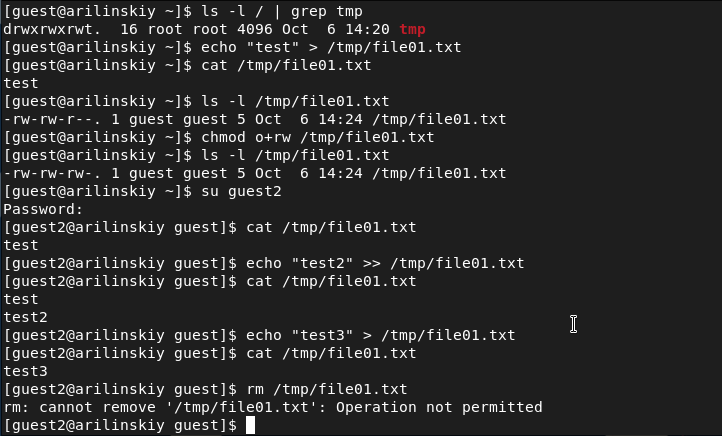
* echo *“test3” > /tmp/file01.txt*
* Мне удалось выполнить операцию.

1. Проверю содержимое файла командой:

* *cat /tmp/file01.txt*

1. От пользователя guest2 попробую удалить файл /tmp/file01.txt командой:

* *rm /tmp/file01.txt*
* Мне не удалось удалить файл



Работа со Sticky битом

**Повышу свои права до суперпользователя и сниму Sticky-бит, после чего повторю проделанные раннее шаги**:

1. Повышу свои права до суперпользователя следующей командой *su* и выполню после этого команду, снимающую атрибут t (Sticky-бит) с директории /tmp:

* *chmod -t /tmp*

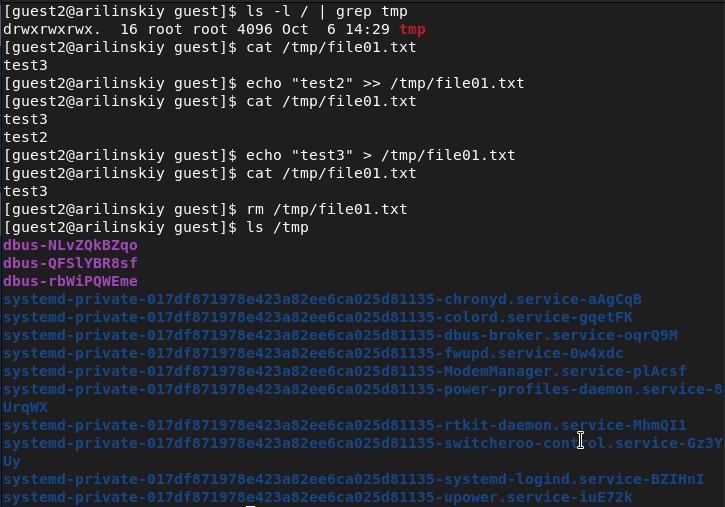
1. От пользователя guest2 проверил, что атрибута t у директории /tmp нет:

* *ls -l / | grep tmp*

1. Повторю предыдущие шаги.

Работа с файлом без Sticly бита (1/2)

Работа с файлом без Sticly бита (1/2)



Работа с файлом без Sticly бита (2/2)

Мне удалось удалить файл от имени пользователя, не являющегося его владельцем. Это связано с тем, что Sticky-bit позволяет защищать файлы от случайного удаления, когда несколько пользователей имеют права на запись в один и тот же каталог. Если у файла атрибут t стоит, значит пользователь может удалить файл, только если он является пользователем-владельцем файла или каталога, в котором содержится файл. Если же этот атрибут не установлен, то удалить файл могут все пользователи, которым позволено удалять файлы из каталога.

**Повышу свои права до суперпользователя и верну Sticky-бит**:

1. Верну атрибут t на директорию /tmp:

*su*

*chmod +t /tmp*

*exit*



Установление атрибута t

# Выводы

Благодаря данной лабораторной работе я изучил механизмы изменения идентификаторов, применения SetUID-, SetGID- и Sticky-битов. Рассмотрел работу механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

# Список литературы

* [Кулябов Д.С., Королькова А.В., Геворкян М.Н *Лабораторная работа №5*](https://esystem.rudn.ru/pluginfile.php/1651753/mod_resource/content/2/005-lab_discret_sticky.pdf)