Лабораторная работа №5

Дискреционное разграничение прав в Linux. Исследование влияния дополнительных атрибутов

Дарья Эдуардовна Ибатулина

Содержание

# 1 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.

# 2 Задание

1. Подготовить лабораторный стенд;
2. Поработать с компилятором gcc;
3. Попрактиковаться в установке и снятии атрибутов с файлов.

# 3 Теоретическое введение

1. Дополнительные атрибуты файлов Linux

В Linux существует три основных вида прав — право на чтение (read), запись (write) и выполнение (execute), а также три категории пользователей, к которым они могут применяться — владелец файла (user), группа владельца (group) и все остальные (others). Но, кроме прав чтения, выполнения и записи, есть еще три дополнительных атрибута. [1]

Sticky bit Используется в основном для каталогов, чтобы защитить в них файлы. В такой каталог может писать любой пользователь. Но, из такой директории пользователь может удалить только те файлы, владельцем которых он является. Примером может служить директория /tmp, в которой запись открыта для всех пользователей, но нежелательно удаление чужих файлов.

SUID (Set User ID) Атрибут исполняемого файла, позволяющий запустить его с правами владельца. В Linux приложение запускается с правами пользователя, запустившего указанное приложение. Это обеспечивает дополнительную безопасность т.к. процесс с правами пользователя не сможет получить доступ к важным системным файлам, которые принадлежат пользователю root.

SGID (Set Group ID) Аналогичен suid, но относиться к группе. Если установить sgid для каталога, то все файлы созданные в нем, при запуске будут принимать идентификатор группы каталога, а не группы владельца, который создал файл в этом каталоге.

Обозначение атрибутов sticky, suid, sgid Специальные права используются довольно редко, поэтому при выводе программы ls -l символ, обозначающий указанные атрибуты, закрывает символ стандартных прав доступа.

Пример:

rwsrwsrwt

где первая s — это suid, вторая s — это sgid, а последняя t — это sticky bit

В приведенном примере не понятно, rwt — это rw- или rwx? Определить это просто. Если t маленькое, значит x установлен. Если T большое, значит x не установлен. То же самое правило распространяется и на s.

В числовом эквиваленте данные атрибуты определяются первым символом при четырехзначном обозначении (который часто опускается при назначении прав), например в правах 1777 — символ 1 обозначает sticky bit. Остальные атрибуты имеют следующие числовое соответствие:

1 — установлен sticky bit 2 — установлен sgid 4 — установлен suid 2. Компилятор GCC

GСС - это свободно доступный оптимизирующий компилятор для языков C, C++. Собственно программа gcc это некоторая надстройка над группой компиляторов, которая способна анализировать имена файлов, передаваемые ей в качестве аргументов, и определять, какие действия необходимо выполнить. Файлы с расширением .cc или .C рассматриваются, как файлы на языке C++, файлы с расширением .c как программы на языке C, а файлы c расширением .o считаются объектными. [2]

# 4 Выполнение лабораторной работы

## 4.1 Подготовка лабораторного стенда

Проверяем, установлен ли gcc (рис. [[1](#fig:001)]).

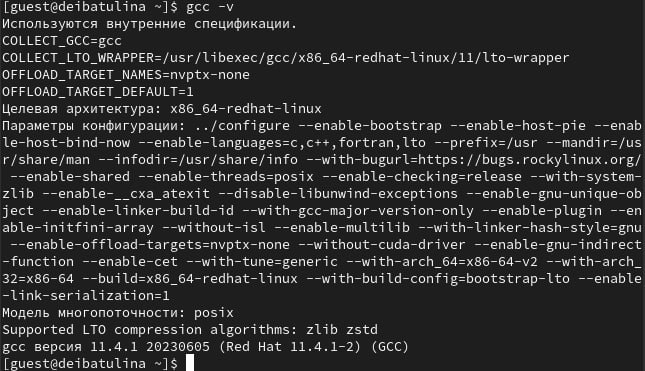


Figure 1: Проверка установки gcc

Устанавливаем setenforce 0 и проверяем что getenforce выдает: “Permissive” (рис. [[2](#fig:002)]).

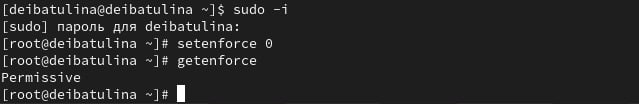


Figure 2: Установка setenforce, проверка getenforce

## 4.2 Создание программы

Переключаемся на учетную запись администратора и создаем файл simpleid.c, заполняем его предложенной программой (рис. [[3](#fig:003)], [[4](#fig:004)]).

Figure 3: Создание файла simpleid.c

Figure 3: Создание файла simpleid.c



Figure 4: Файл simpleid.c

Скомпилируем программу, выполним её и сравним вывод команды id с результатом исполнения программы. Они совпадают (рис. [[5](#fig:005)]).

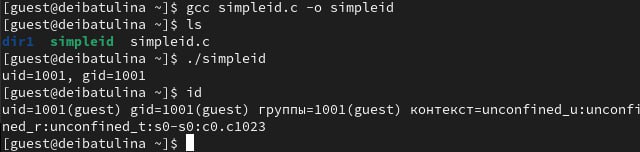


Figure 5: Компиляция, исполнение файла simpleid.c, сравнение результата с выводом команды id

Листинг программы файла simpleid.c:

#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;  
}

Создадим другой файл simpleid2.c, введём в него код более сложной программы, выполним его (рис. [[6](#fig:006)], [[7](#fig:007)]).



Figure 6: Файл simpleid2.c

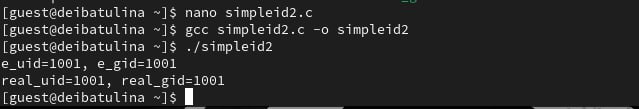


Figure 7: Запуск программы simpleid2.c

Листинг программы файла simpleid2.c:

#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;  
}

Переключаемся на администратора и выполняем следующие команды (рис. [[8](#fig:008)]).



Figure 8: Смена владельца файла и передача некоторых прав пользователю guest

Этими командами была произведена смена пользователя файла на root и установлен SetUID-бит.

Выполним проверку правильности установки атрибутов, а также скомпилируем и запустим файл simleid2, сравним результат с выводом команды id (рис. [[9](#fig:009)]).

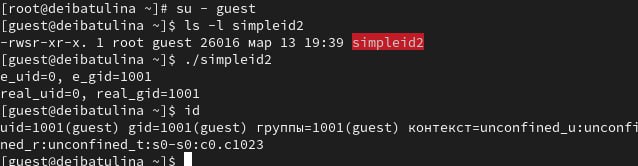


Figure 9: Проверка правильности установки атрибутов

Создаём файл readfile.c (рис. [[10](#fig:010)]) и изменяем его владельца так, чтобы только суперпользователь мог прочитать его, а guest не мог (рис. [[11](#fig:011)]).



Figure 10: Файл readfile.c

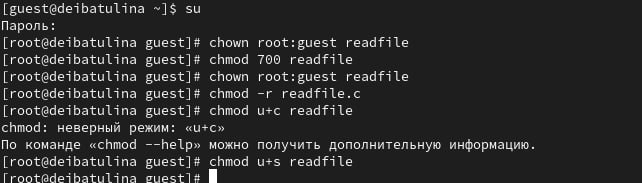


Figure 11: Установка прав на файл readfile.c

Листинг программы файла 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;  
}

Проверим, что пользователь guest не может прочитать данный файл. Также программа readfile не сможет прочитать файл readfile.c и readfile не может прочитать файл /etc/shadow (рис. [[12](#fig:012)]).

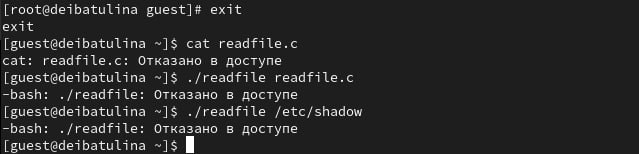


Figure 12: Проверка правильности выполненных действий

Но от имени суперпользователя удаётся выполнить все вышеприведённые команды (рис. [[13](#fig:013)], [[14](#fig:014)]).

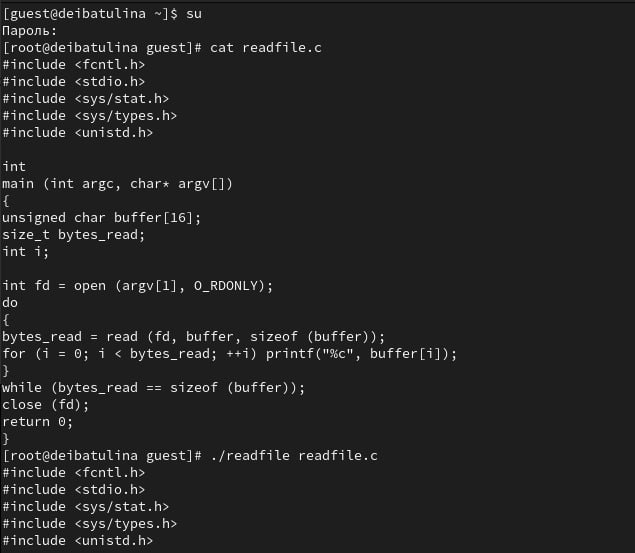


Figure 13: Выполнение команд от имени суперпользователя

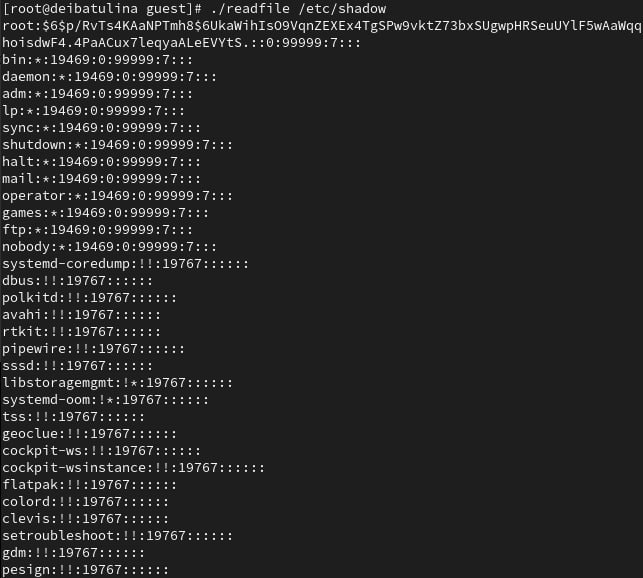


Figure 14: Выполнение команд от имени суперпользователя

## 4.3 Исследование Sticky-бита

Выясним, установлен ли атрибут Sticky на директории /tmp, от имени пользователя guest создадим файл file01.txt в директории /tmp со словом test, и просмотрим атрибуты у только что созданного файла и разрешим чтение и запись для категории пользователей «все остальные» (рис. [[15](#fig:015)]).

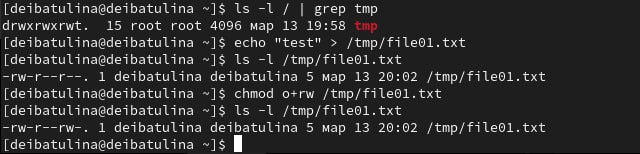


Figure 15: Выполнение заданий

От пользователя guest2 (не являющегося владельцем) попробуем прочитать файл /tmp/file01.txt:, от пользователя guest2 попробуем дозаписать в файл /tmp/file01.txt слово test2, проверим содержимое файла, от пользователя guest2 попробуем записать в файл /tmp/file01.txt слово test3, стерев при этом всю имеющуюся в файле информацию и проверим содержимое файла (рис. [[16](#fig:016)]).



Figure 16: Выполнение заданий

Удалить данный файл от имени guest2 не удаётся.

Повысим свои права до суперпользователя и выполним после этого команду, снимающую атрибут t (Sticky-бит) с директории /tmp, а затем покинем режим суперпользователя. От пользователя guest2 проверим, что атрибута t у директории /tmp нет. Повторив предыдущие шаги от имени других пользователей, я могу заметить, что всё получается. Отвечая на вопрос: Удалось ли вам удалить файл от имени пользователя, не являющегося его владельцем? я могу сказать, что удалось. Повысим свои права до суперпользователя и вернём атрибут t на директорию /tmp (рис. [[17](#fig:017)]).

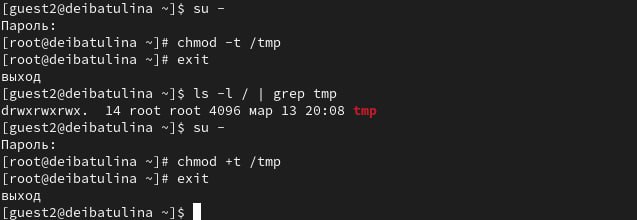


Figure 17: Выполнение заданий

# 5 Выводы

Мною были изучены механизмы изменения идентификаторов и применения SetUID- и Sticky-битов. Получены практические навыки работы в консоли с дополнительными атрибутами. Были рассмотрены работа механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

# Список литературы

[0] Методические материалы курса

[1] Дополнительные атрибуты: https://tokmakov.msk.ru/blog/item/141

[2] Компилятор GSS: http://parallel.imm.uran.ru/freesoft/make/instrum.html