Отчёт по лабораторной работе №5

Дисциплина: Основы информационной безопасности

Зинченко Анастасия Романовна

Содержание

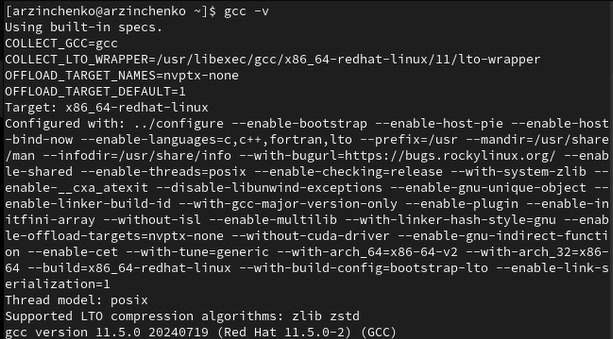
# Цель работы

Целью данной работы является 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.

# Выполнение лабораторной работы

## Подготовка лабораторного стенда

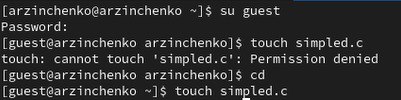
Для лабораторной работы необходимо проверить, установлен ли компилятор gcc. Проверим это с помощью команды *gcc -v*. Также осуществим отключение системы запретов с помощью *sudo setenforce 0* и проверим командой *getenforce*, что выводится Permissive (рис. [-@fig:001])



Подготовка к лабораторной работе

## Создание программы

Войдём в систему от имени пользователя guest. И создадим программу simpleid.c (рис. [-@fig:002]), (рис. [-@fig:003])



Создание и открытие файла simpleid.c

Листинг программы 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;  
 }



Редактирование файла simpleid.c

Скомплилируем программу и убедимся, что файл программы создан: *gcc simpleid.c -o simpleid* (рис. [-@fig:004])



Компиляция программы simpleid.c

Выполним программу simpleid: *./simpleid*. В выводе файла выписыны номера пользоватея и групп, от вывода при вводе id, они отличаются только тем, что информации меньше (рис. [-@fig:005])

Выполнение программы simpleid

Выполнение программы simpleid

Выполните системную программу id: *id* (рис. [-@fig:006])

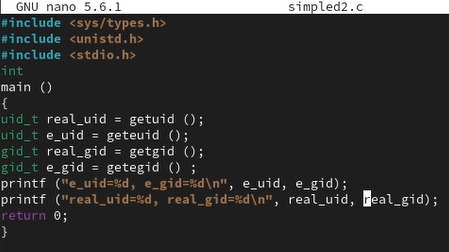
Выполнение системной программы id

Выполнение системной программы id

Далее усложним программу, добавив вывод действительных идентификаторов. Создадим программу под названием simpleid2.c (рис. [-@fig:007])

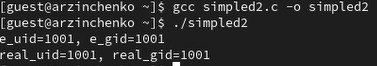
Листинг программы simpleid2.c:

#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;  
 }



Редактирование файла simpleid2.c

Скомпилируем и запустим simpleid2.c: *gcc simpleid2.c -o simpleid2* и *./simpleid2* (рис. [-@fig:008]), (рис. [-@fig:009])

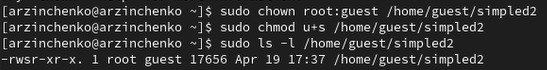


Компиляция программы simpleid2.c

Выполнение программы simpleid2

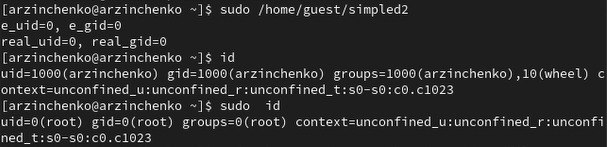
Выполнение программы simpleid2

От имени суперпользователя выполним команды: *chown root:guest /home/guest/simpleid2* и *chmod u+s /home/guest/simpleid2*. Также выполним проверку правильности установки новых атрибутов и смены владельца файла simpleid2: *ls -l simpleid2*. С помощью chown мы меняем владельца файла на суперпользователя, а с помощью chmod меняем права доступа (рис. [-@fig:010])



Смена владельца файла и прав доступа к файлу simpleid2

Запустим simpleid2 и id: *./simpleid2* и *id*. Наша команда снова вывела только ограниченное количество информации (рис. [-@fig:011])



Запуск simpleid2 и id

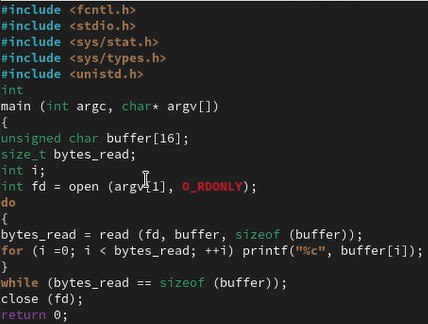
Создадим программу readfile.c (рис. [-@fig:012]), (рис. [-@fig:013])

Создание и открытие файла readfile.c

Создание и открытие файла readfile.c

Листинг программы 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;  
 }



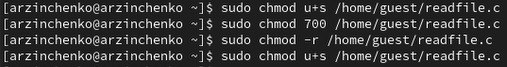
Редактирование файла readfile.c

Откомпилируем её: *gcc readfile.c -o readfile* (рис. [-@fig:014])



Компиляция программы readfile.c

Далее сменим владельца у файла readfile.c и изменим права так, чтобы только суперпользователь (root) мог прочитать его, a guest не мог (рис. [-@fig:015])



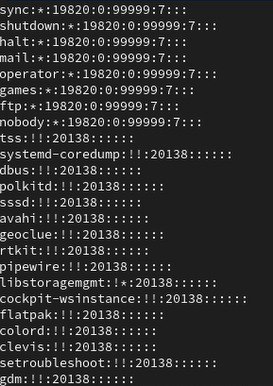
Смена владельца файла и прав доступа к файлу readfile

Проверим, может ли программа readfile прочитать файл readfile.c (рис. [-@fig:016])



Попытка прочесть содержимое файла readfile.c программой readfile

Теперь попробуем прочесть эти же файлы от имени суперпользователя (рис. [-@fig:018])



Попытка прочесть содержимое файлов от имени суперпользователя

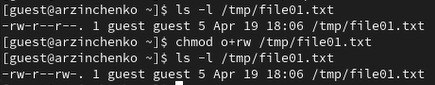
## Исследование Sticky-бита

Выясним, установлен ли атрибут Sticky на директории /tmp, для чего выполним команду *ls -l / | grep tmp*. Так как ввыводе есть буква t, это значит что атрибут установлен (рис. [-@fig:019])

Проверка атрибутов директории tmp

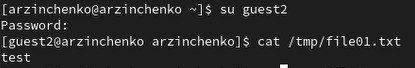
Проверка атрибутов директории tmp

От имени пользователя guest создадим файл file01.txt в директории /tmp со словом test: *echo “test” > /tmp/file01.txt* (рис. [-@fig:020])



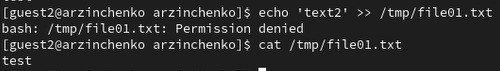
Создание файла file01.txt со словом test

Просмотрим атрибуты у только что созданного файла *ls -l /tmp/file01.txt* и разрешим чтение и запись для категории пользователей «все остальные»: *chmod o+rw /tmp/file01.txt* (рис. [-@fig:021])



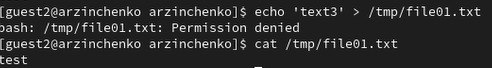
Смена атрибутов файла file01.txt

От пользователя guest2 (не являющегося владельцем) попробуем прочитать файл /tmp/file01.txt: *cat /tmp/file01.txt* (рис. [-@fig:022])



Попытка чтеня файла

От пользователя guest2 попробуем дозаписать в файл /tmp/file01.txt слово test2 командой *echo “test2” > /tmp/file01.txt* (рис. [-@fig:023])



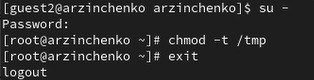
Попытка дозаписи в файл

От пользователя guest2 попробуем записать в файл /tmp/file01.txt слово test3, стерев при этом всю имеющуюся в файле информацию командой *echo “test3” > /tmp/file01.txt* (рис. [-@fig:024])

Попытка записи в файл и удаление всей имеющейся в нём информации

Попытка записи в файл и удаление всей имеющейся в нём информации

От пользователя guest2 попробуем удалить файл /tmp/file01.txt командой *rm /tmp/file01.txt* (рис. [-@fig:025])



Попытка удалить файл

Далее от имени суперпользователя снимем с директории tmp атрибут Sticky командой *chmod -t /tmp* (рис. [-@fig:026]), (рис. [-@fig:027])

Снятие с директории tmp атрибута Sticky

Снятие с директории tmp атрибута Sticky

Проверка того что атрибут снялся

Проверка того что атрибут снялся

# Выводы

В результате выполнения работы мы изучили механизмы изменения идентификаторов, применения SetUID- и Sticky-битов. Получили практические навыки работы в консоли с дополнительными атрибутами. Рассмотрели работы механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов

# Список литературы

1. Лаборатораня работа №5 [Электронный ресурс] URL: https://esystem.rudn.ru/pluginfile.php/2580984/mod\_resource/content/2/005-lab\_discret\_sticky.pdf
2. Инструментарий программиста в Linux: Компилятор GCC [Электронный ресурс] URL: http://parallel.imm.uran.ru/freesoft/make/instrum.html