Отчет по лабораторной работе №5

Информационная безопасноть

Астафьева Анна Андреевна НПИбд-01-18

Содержание

# Цель работы

Изучить механизмы изменения идентификаторов, применение SetUID- и Sticky-битов. Получить практические навыки работы в консоли с дополнительными атрибутами. Рассмотреть работу механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

# Теоретическое описание

В Linux, как и в любой многопользовательской системе, абсолютно естественным образом возникает задача разграничения доступа субъектов — пользователей к объектам — файлам дерева каталогов.

Setuid, Setgid и Sticky Bit - это специальные типы разрешений позволяют задавать расширенные права доступа на файлы или каталоги. Setuid – это бит разрешения, который позволяет пользователю запускать исполняемый файл с правами владельца этого файла. Другими словами, использование этого бита позволяет нам поднять привилегии пользователя в случае, если это необходимо. Принцип работы Setgid очень похож на setuid с отличием, что файл будет запускаться пользователем от имени группы, которая владеет файлом. Последний специальный бит разрешения – это Sticky Bit . В случае, если этот бит установлен для папки, то файлы в этой папке могут быть удалены только их владельцем.

# Выполнение лабораторной работы

1. Вошла в систему от имени пользователя guest, создала программу simpleid.c. (рис. 1), (рис. 2).

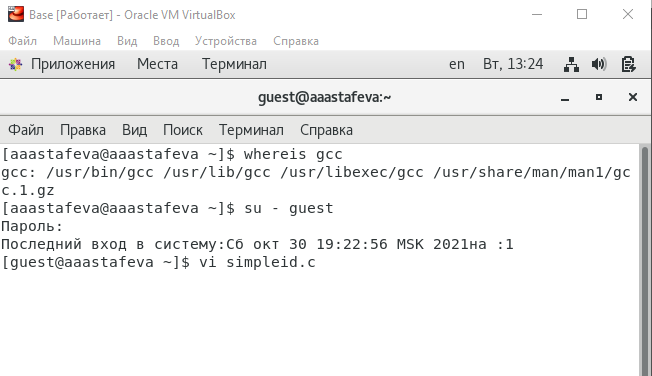


Figure 1: Создание файла simpleid.c

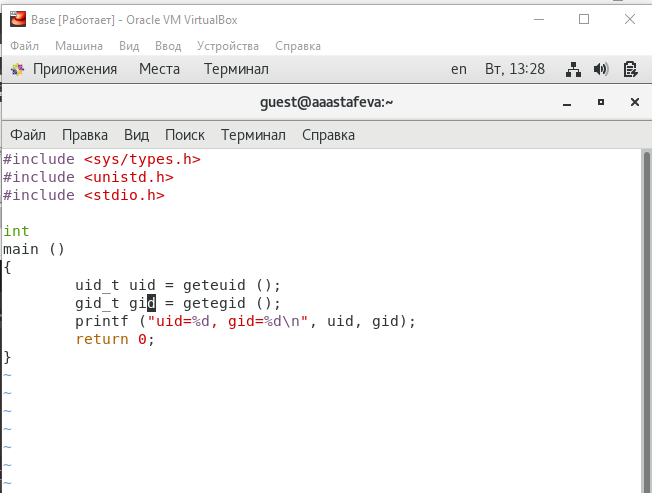


Figure 2: Написание программы simpleid.c

1. Скомплировала программу и убедилась, что файл программы создан: gcc simpleid.c -o simpleid. Выполнила программу simpleid: ./simpleid. Выполнила системную программу id. И сравнила полученный результат с данными предыдущего пункта задания. (Данные одинаковы)(рис. 3).

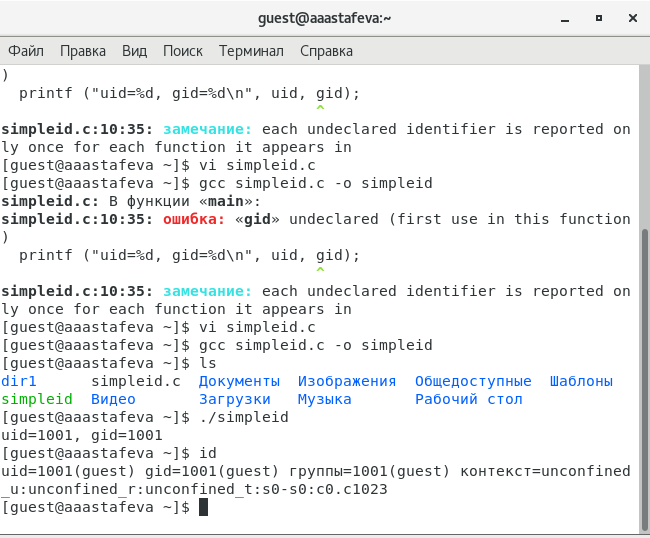


Figure 3: Компиляция, выполнение программы

1. Усложнила программу, добавив вывод действительных идентификаторов. (рис. 4).

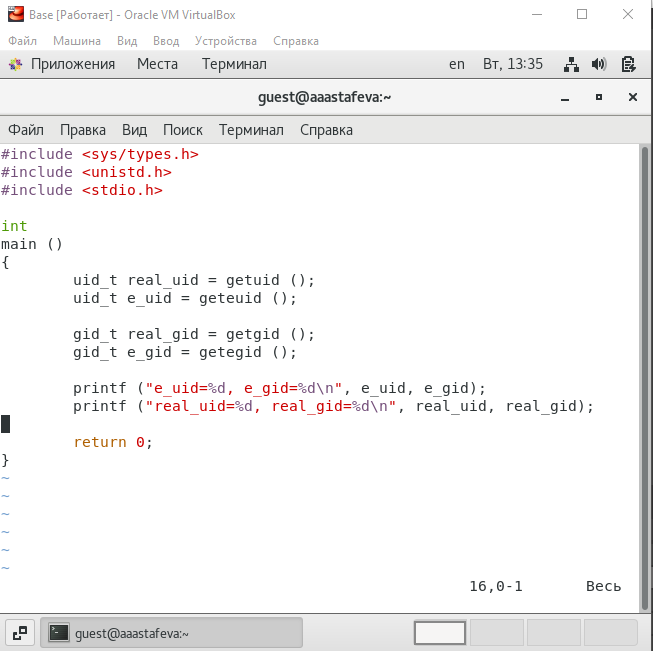


Figure 4: Создание программы simpleid2.c

1. Скомпилировала и запустила simpleid2.c: gcc simpleid2.c -o simpleid2; ./simpleid2 (рис. 5).

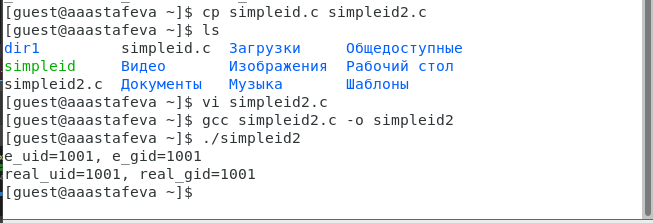


Figure 5: Компиляция, выполнение программы

1. От имени суперпользователя выполнила команды: chown root:guest /home/guest/simpleid2; chmod u+s /home/guest/simpleid2. (рис. 6).

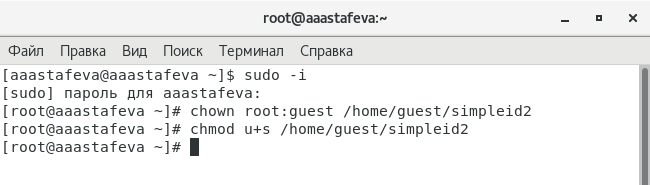


Figure 6: Выполнение

С помощью первой команды для файла simpleid2 мы поменяли пользователя и группу на root и guest соответственно. С помощью второй установили разрешение для владельца на выполнение с разрешением суперпользователя.

1. Выполнила проверку правильности установки новых атрибутов и смены владельца файла simpleid2: ls -l simpleid2. Запустила simpleid2 и id. (рис. 7).

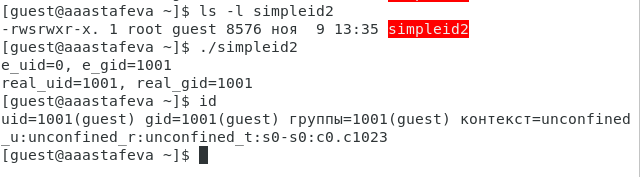


Figure 7: Выполнение

1. Создала и откомпилировала программу readfile.c: (рис. 8).

Figure 8: Создание и компиляция программы readfile.c

Figure 8: Создание и компиляция программы readfile.c

1. Сменила владельца у файла readfile.c и изменила права так, чтобы только суперпользователь (root) мог прочитать его, a guest не мог. (рис. 9).

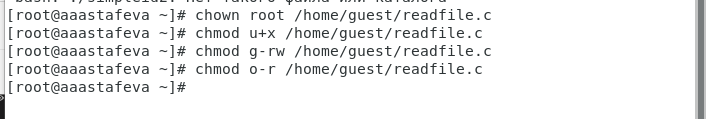


Figure 9: Изменение владельца и прав

1. Проверила, что пользователь guest не может прочитать файл readfile.c. (рис. 10).

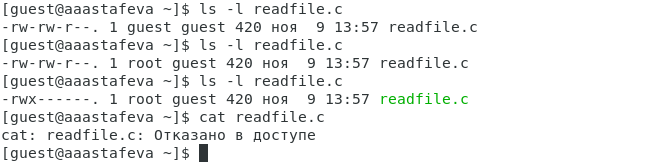


Figure 10: Проверка

1. Сменила у программы readfile владельца и установила SetU’D-бит. (рис. 11).

Figure 11: Изменение для программы readfile

Figure 11: Изменение для программы readfile

1. Проверила, может ли программа readfile прочитать файл readfile.c (может), проверила, может ли программа readfile прочитать файл /etc/shadow (может). (рис. 12). (рис. 13).

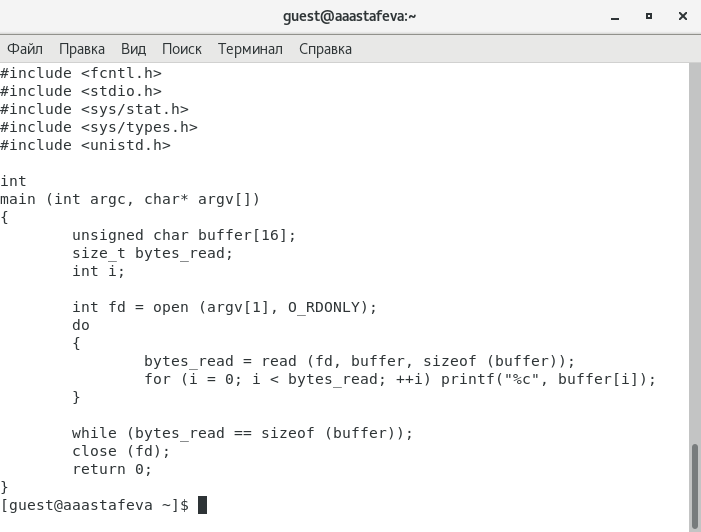


Figure 12: Проверка

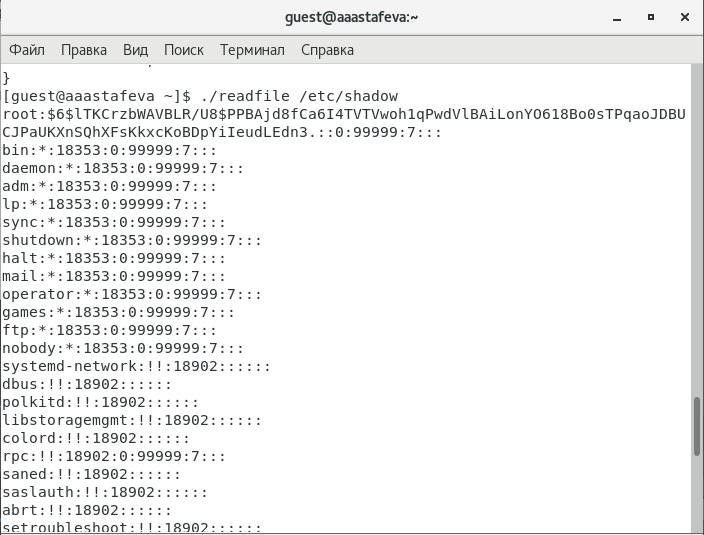


Figure 13: Проверка

1. Выяснила, установлен ли атрибут Sticky на директории /tmp, для чего выполнила команду: ls -l / | grep tmp. От имени пользователя guest создала файл file01.txt в директории /tmp со словом test: echo “test” > /tmp/file01.txt. Просмотрела атрибуты у только что созданного файла и разрешила чтение и запись для категории пользователей «все остальные»: ls -l /tmp/file01.txt, chmod o+rw /tmp/file01.txt, ls -l /tmp/file01.txt. (рис. 14).

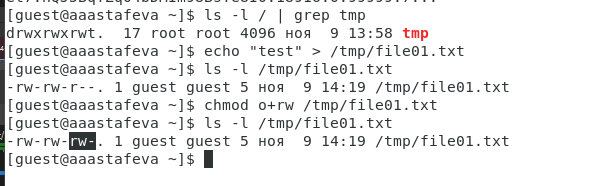


Figure 14: Выполение

1. От пользователя guest2 (не являющегося владельцем) попробовала прочитать файл /tmp/file01.txt: cat /tmp/file01.txt, попробовала дозаписать в файл /tmp/file01.txt слово test2 командой: echo “test2” >> /tmp/file01.txt. Проверила содержимое файла командой: cat /tmp/file01.txt. Также попробовала записать в файл /tmp/file01.txt слово test3, стерев при этом всю имеющуюся в файле информацию командой: echo “test3” > /tmp/file01.txt От пользователя guest2 попробовала удалить файл /tmp/file01.txt командой: rm /tmp/fileOl.txt. (Все действия, кроме удаления файла, выполнить удалось). (рис. 15).

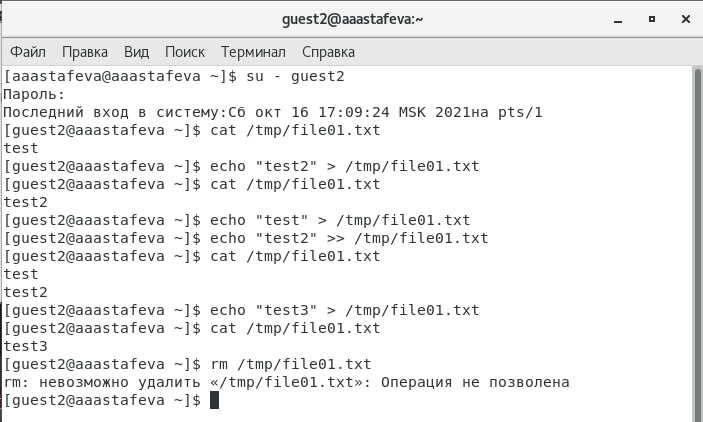


Figure 15: Выполение и проверка от пользователя guest2

1. От суперпользователя выполнила команду, снимающую атрибут t (Sticky-бит) с директории /tmp: chmod -t /tmp. (рис. 16).

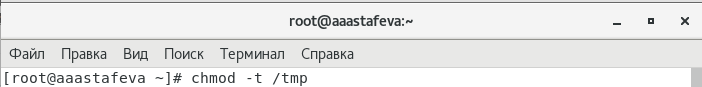


Figure 16: Снятие атрибута “t” с директории /tmp

1. От пользователя guest2 проверила, что атрибута t у директории /tmp нет: ls -l / | grep tmp. Повторила предыдущие шаги. Нам удалось удалить файл от имени пользователя, не являющегося его владельцем, также получилось выполнить дозапись в файл и замену текста в файле. (рис. 17).

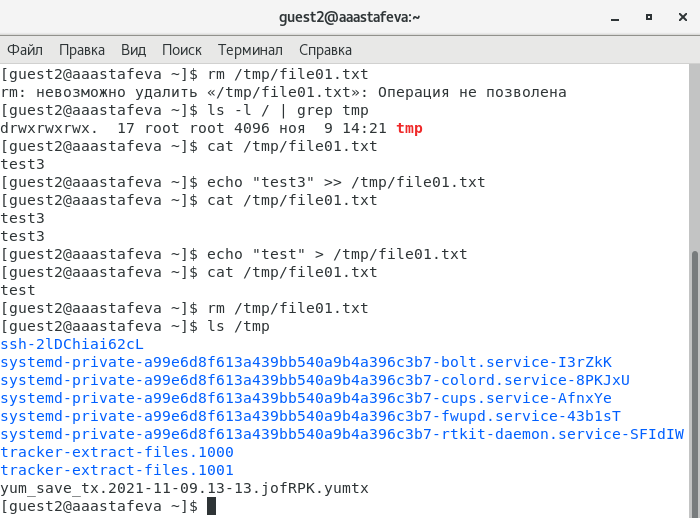


Figure 17: Проверка

1. От суперпользователя вернула атрибут t на директорию /tmp: chmod +t /tmp. (рис. 18).

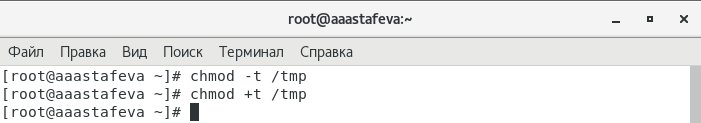


Figure 18: Добавление атрибута “t” на директорию /tmp

# Выводы

На основе проделанной работы я изучила механизмы изменения идентификаторов, применение SetUID- и Sticky-битов. Получла практические навыки работы в консоли с дополнительными атрибутами. Рассмотрела работу механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.