Лабораторная работа №5

Основы Информационной Безопасности

Чистов Даниил Максимович

Содержание

# 1 Цель работы

Целью данной лабораторной 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

# 2 Задание

1. Исследование SetUID и SetGID битов
2. Исследование Sticky-битов

# 3 Выполнение лабораторной работы

## 3.1 Подготовка к лабораторной работы

Перед выполнение требуется выяснить, установлен ли компилятор gcc. Всё установлено. Также нужно отключить систему запретов командой setenforce 0, после чего вывод команды getenforce становится “Permissive” (рис. 1).

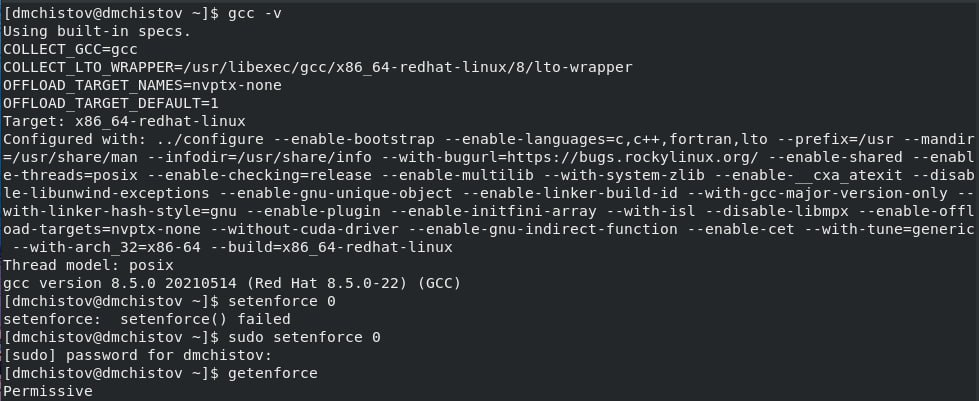


Рис. 1: Подготовка к выполнению

## 3.2 Исследование SetUID и SetGID битов

Вхожу в систему как пользователь guest и создаю файл simpleid.c (рис. 2).



Рис. 2: Создание файла simpleid.c

Вставляю в файл код из задания, компилирую, запускаю, после чего прописываю команду id и сравниваю результаты. Всё сходится (рис. 3).



Рис. 3: Код и вывод программы simpleid

Усложняю программу, вставив новый код из задания, сохраняю файл как simpleid2.c, компилирую и запускаю (рис. 4).



Рис. 4: Работа программы simpleid2

Прописываю в консоль следующие команды, как на скриншоте ниже (рис. 5).

Первая комнада меняет владельца файла на root. Теперь суперпользователь является владельцем файла. Вторая команда устанавливает SetUID-бит, т.е. программа будет запускаться от имени владельца файла, а не от того, кто её запустил.

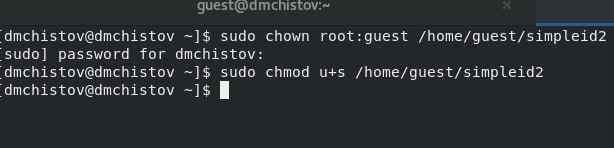


Рис. 5: Команда смены владельца и SetUID бит

В предисловии к данной лабораторной работе сказано: “любая bash-программа интерпретируется в процессе своего выполнения, т.е. существует сторонняя программа-интерпретатор, которая выполняет считывание файла сценария и выполняет его последовательно. Сам интерпретатор выполняется с правами пользователя, его запустившего, а значит, и выполняемая программа использует эти права.”

Проверяю правильно ли установились новые атрибуты и сменился ли владелец у программы simpleid2 - всё верно (рис. 6).

Новый владелец и атрибуты программы simpleid2

Рис. 6: Новый владелец и атрибуты программы simpleid2

Теперь запускаю simpleid2 и id, сравниваю результат. e\_uid стал равен 0 при выполнении программмы simpleid2 (рис. 7).

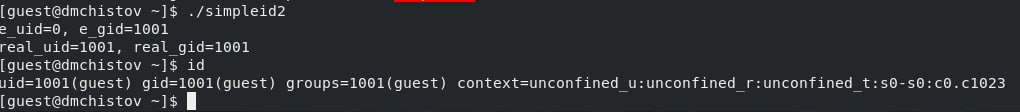


Рис. 7: simpleid2 и id

Тоже самое проделываю относительно SetGID-бит, т.е. в этот раз пишу не chmod u+s, а chmod g+s (рис. 8).

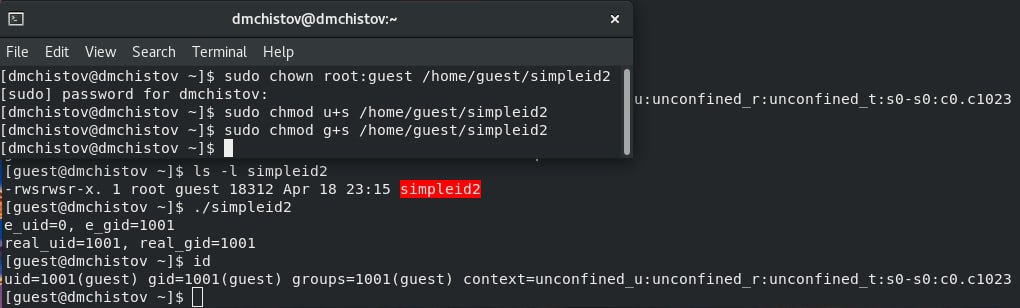


Рис. 8: simpleid2 и id при SetGID-бите

Создаю новую программу readfile.c, беру код из задания, компилирую программу (рис. 9).

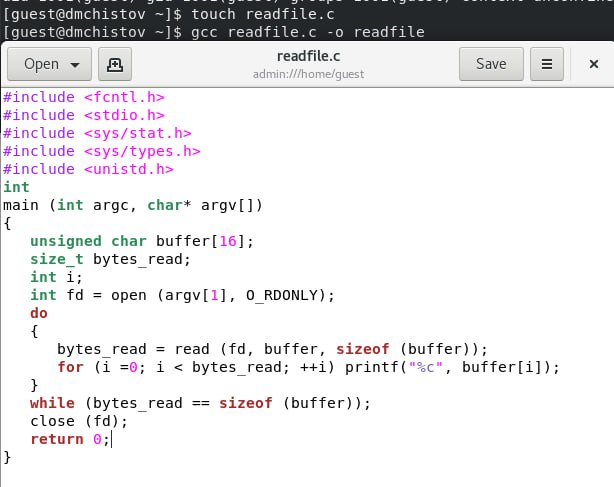


Рис. 9: readfile.c

Делаю так, чтобы только суперпользователь мог читать файл readfile.c, а guest не мог. Также изменяю владельца этого файла. На скриншоте ниже видно, как в нижней консоли, пользователь guest не может прочитать этот файл (рис. 10).

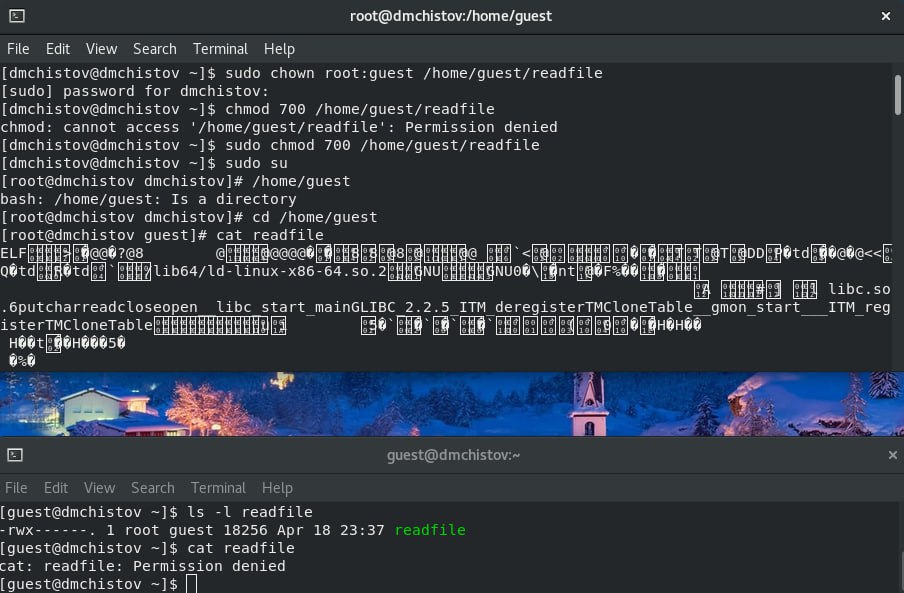


Рис. 10: Смена прав и владельца readfile

Теперь меняю владельца у скомпилированной программы readfile и устанавливаю SetUID-бит. Проверяю, может ли наша программа прочитать файл readfile.c. Если запускать её в консоли суперпользователя, то она, естественно всё прочитает (рис. 11).

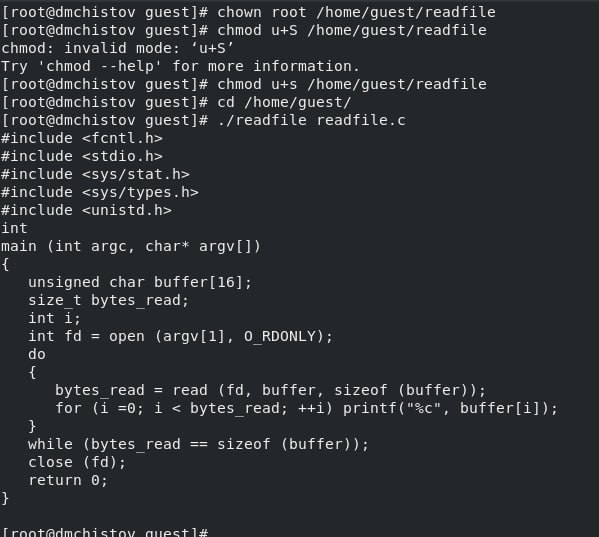


Рис. 11: Попытка прочесть readfile.c программой от лица суперпользователя

Однако, если запустить её в консоли пользователя guest, то нам выдают сообщение об ошибке - недостаточно прав (рис. 12).

Попытка прочесть readfile.c программой от лица guest

Рис. 12: Попытка прочесть readfile.c программой от лица guest

Аналогичная ситуация происходит при попытке прочесть /etc/shadow (рис. 13).



Рис. 13: Попытка прочесть /etc/shadow программой

## 3.3 Исследование Sticky-битов

Выясняю, установлен ли sticky-бит на директорию /tmp. Да, установлен, т.к. в конце сть буква t (рис. 14).

Наличие атрибута sticky у директории tmp

Рис. 14: Наличие атрибута sticky у директории tmp

От имени пользователя guest создаю файл file01.txt со словом “test”, затем разрешаю чтение и запись для категории пользователей “все остальные” для данного файла (рис. 15).

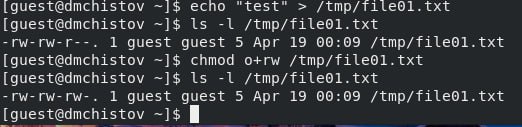


Рис. 15: Успешно добавленные атрибуты у file01.txt

От пользователя guest2 пытаюсь прочитать файл, дозаписать в него слово и перписать его полностью. Всё работает, следовательно мы успешно установили атрибуты в прошлом шаге (рис. 16).

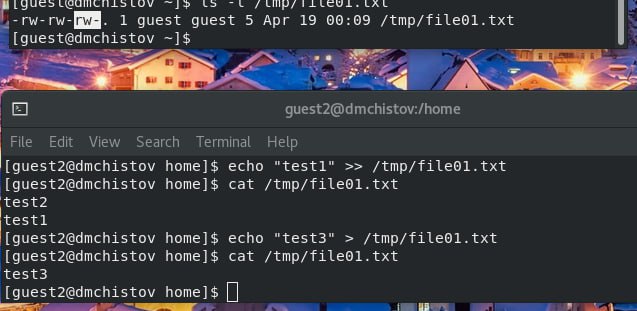


Рис. 16: Проверка работы атрибутов файла file01.txt

Однако, удалить этот файл от лица guest2 не получается (рис. 17).



Рис. 17: guest2 не может удалить file01.txt

От лица суперпользователя снимаю атрибут t (Sticky-бит) с директории /tmp, после чего командой “ls -l / | grep tmp” проверяю, что атрибута t действительно больше нет (рис. 18).

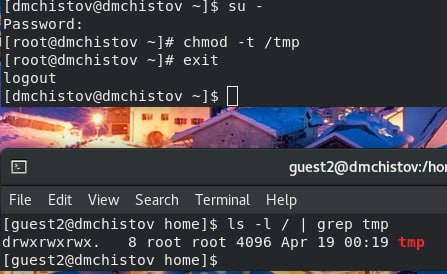


Рис. 18: Снятие атрибута t с директории tmp

Пытаюсь повторить все предыдущие шаги. Guest2 до сих пор может прочитать файл, дозаписать в него что-либо, переписать его, но теперь он также может его удалить, благодаря тому, что мы сняли атрибут t с папки, в которой лежит этот файл (рис. 19).

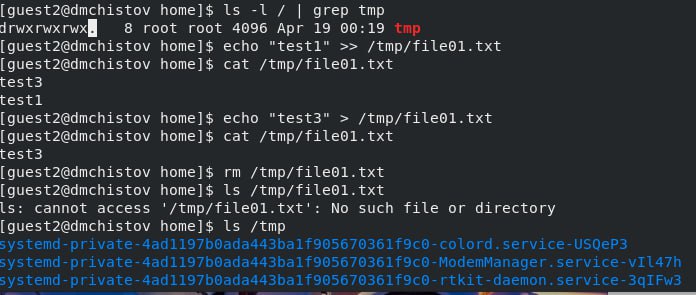


Рис. 19: Новые возможности пользователя guest2

Папка tmp довольно важна, поэтому лучше атрибут t вернуть на место, после наших экспериментов (рис. 20).

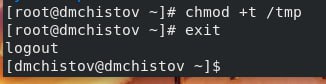


Рис. 20: Возвращение атрибута t

# 4 Выводы

В результате выполнения данной лабораторной работы я изучил механизмы изменения идентификаторов, применение SetUID- и Sticky-битов. Получил практические навыки работы в консоли с дополнительными атрибутами. Рассмотрел работы механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов

# 5 Список литературы

[Лабораторная работа №5](https://esystem.rudn.ru/pluginfile.php/2580598/mod_resource/content/2/005-lab_discret_sticky.pdf)