Отчёт по лабораторной работе №6

Основы информационной безопасности

Бережной Иван Александрович

Содержание

Список иллюстраций

# 1 Цель работы

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

# 2 Задание

1. Создать программы
2. Исследовать Sticky-бит

# 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. [**u?**]

**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

1. 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 [**gcc?**].

# 4 Выполнение лабораторной работы

## 4.1 Создание программ

Войдём в систему от имени пользователя guest (рис. 1) и создадим программу simpleid.c (рис. 2 и рис. 3).

Рис. 1: Вход в систему под guest

Рис. 1: Вход в систему под guest

Рис. 2: Создание simpleid.c 1

Рис. 2: Создание simpleid.c 1



Рис. 3: Создание simpleid.c 2

Листинг 1:

#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;  
}

Скомпилируем программу и убедимся, что файл создан (рис. 4). Выполним программу, а также команду id, и сравним результаты (рис. 5). Как видно, встроенная команда даёт больше информации.

Рис. 4: Компиляция simpleid.c

Рис. 4: Компиляция simpleid.c

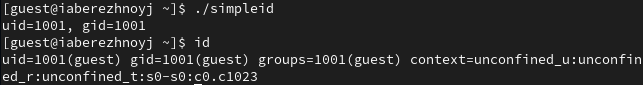


Рис. 5: Сравнение simpleid и команды id

Усовершенствуем нашу программу, добавив вывод действительных идентификаторов (рис. 6).



Рис. 6: Модификация simpleid.c

Листинг 2:

#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;

Скомпилируем и запустим новую программу (рис. 7).

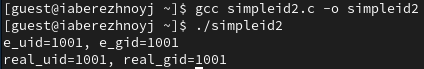


Рис. 7: Запуск simpleid2

От имени суперпользователя меняем владельца файла на суперпользователя и меняем права с помощью chmod (рис. 8). Проверим правильность установки атрибутов (рис. 9).

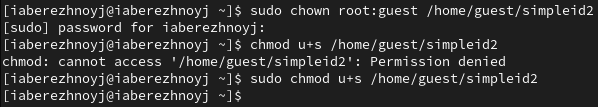


Рис. 8: Смена владельца и прав

Рис. 9: Проверка атрибутов файла

Рис. 9: Проверка атрибутов файла

Запустим simpleid2 и id. Собственная команда выводит всё ещё ограниченное количество информации (рис. 10).

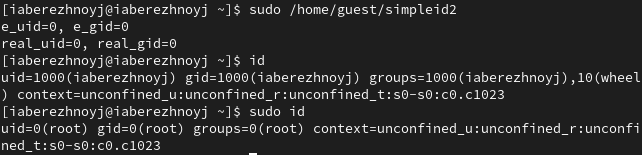


Рис. 10: Запуск simpleid2 и id после смены прав

Создадим программу readfile.c (рис. 11 и рис. 12).

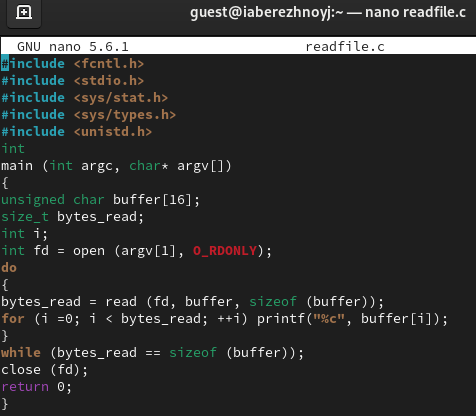


Рис. 11: Создание readfile.c 1

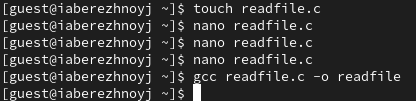


Рис. 12: Создание readfile.c 2

Листинг 3:

#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);  
ret

Сменим владельца у файла readfile.c и изменим права так, чтобы только суперпользователь смог его прочитать (рис. 13).

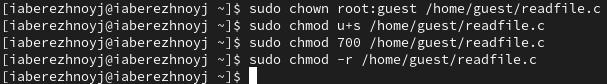


Рис. 13: Ограничение доступа к readfile.c

Попробуем прочитать файл от имени guest. Получили ошибку (рис. 14). Попытки прочитать файл другими путями также не увенчались успехом (рис. 15).

Рис. 14: Ошибка чтения readfile.c от guest

Рис. 14: Ошибка чтения readfile.c от guest



Рис. 15: Неудачные попытки чтения readfile.c

## 4.2 Исследование Sticky-бита

Посмотрим, установлен ли атрибует Sticky на директории /tmp (рис. 16). В выводе присутствует буква t, значит, установлен.

Рис. 16: Проверка Sticky-бита в /tmp

Рис. 16: Проверка Sticky-бита в /tmp

От имени пользователя guest создадим фалй file01.txt в директории /tmp, разрешим чтение и запись для остальных польователей (рис. 17). Теперь от имени пользователя guest2 попробуем прочитать файл (успешно) и дописать что-либо в него (ошибка доступа). Также не получается удалить файл (рис. 18).

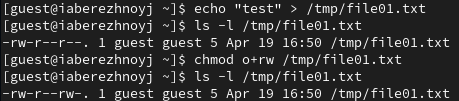


Рис. 17: Создание file01.txt в /tmp

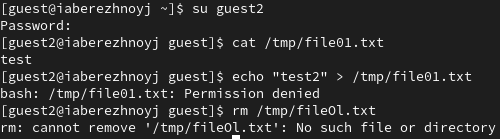


Рис. 18: Попытка записи и удаления file01.txt от guest2

Повысим права до суперпользователя и снимем атрибут t с файла (рис. 19). Снова от имени пользователя guest2 повторим попытку выполнить команды. Кроме того, что получилось удалить файл, ничего не поменялось.

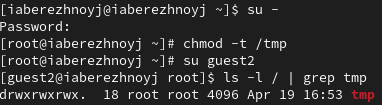


Рис. 19: Снятие Sticky-бита с /tmp

Вернём атрибут (рис. 20).

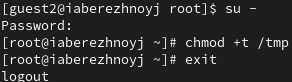


Рис. 20: Возврат атрибуета t

# 5 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.