Лабораторная работа №6
«Дискреционное управление доступом»

1. Теоретическая часть

1.1. Введение в управление доступом

Управление доступом — это механизм, который определяет, какие пользователи или группы пользователей могут взаимодействовать с ресурсами системы (файлами, процессами и т.д.). В современных операционных системах, таких как Astra Linux, управление доступом реализуется через две основные модели: дискреционное управление доступом (DAC) и мандатное управление доступом (MAC)

1.2. Дискреционное управление доступом

Дискреционное управление доступом (Discretionary Access Control, DAC) — это модель контроля доступа, которая предоставляет владельцам объектов (файлов, каталогов и т.д.) возможность самостоятельно управлять правами доступа к своим ресурсам. В контексте операционной системы Linux, DAC является одной из основных систем безопасности, которая определяет, кто может читать, записывать или выполнять файлы и другие ресурсы.

1.2.1. Основные концепции DAC в Linux

- Права доступа. В Linux каждый файл и каталог имеют три основных вида прав доступа
  - Чтение (r): Позволяет просматривать содержимое файла или каталога
  - Запись (w): Позволяет изменять содержимое файла или структуры каталога
  - Исполнение (x): Позволяет выполнять файл (если это исполняемый файл или скрипт) или заходить в каталог
- Категории пользователей. Права доступа разделяются на три категории

- Владелец (user): Пользователь, которому принадлежит файл или каталог
- Группа (group): Группа пользователей, которая может разделять права доступа к файлу или каталогу
- Остальные пользователи (others): Все остальные пользователи, которые не являются владельцем или не входят в группу

- Модель управления правами. Права доступа в Linux представлены в виде трех битов для каждой категории пользователей: владельца, группы и остальных. Эти биты определяют права чтения, записи и выполнения
  - Чтение — r (4)
  - Запись — w (2)
  - Исполнение — x (1)

Например, права *rwxr-xr--* означают, что владелец может читать, записывать и выполнять файл. Группа может читать и выполнять файл, но не может его изменять. Остальные пользователи могут только читать файл.

Обратите внимание на особенности значения прав доступа до каталогов. Атрибут «r» – дает доступ на чтение содержимого каталога – т.е. возможно узнать имена файлов и номера их inode. Атрибут «x» дает право чтения данных из inode файлов, хранящихся в каталоге. На практике для каталогов используется три режима доступа: 7 (rwx), 5 (r-x) и 0 (—).

1.3. Управление правами доступа в Linux

1. Команда chmod (change mode) в Linux используется для изменения прав доступа к файлам и директориям. С её помощью администраторы и пользователи могут настраивать, кто может читать, записывать или выполнять файлы.

1.1. Формат команды

chmod [опции] <права> <файл/директория>

1.2. Пример использования:

$ chmod o-x filename

1.3. Права доступа

Права доступа делятся на 3 категории

- Чтение (r) — право на чтение файла или списка файлов в директории
- Запись (w) — право на изменение содержимого файла или добавление/удаление файлов в директории
- Выполнение (x) — право на выполнение файла как программы или скрипта

1.4. Пользовательские группы

- u — владелец файла (user)
- g — группа, к которой принадлежит файл (group)
- o — остальные пользователи (others)
- a — все пользователи (all)

1.5. Формы задания прав

Существует две формы задания прав: символьная и цифровая

В символьной форме права задаются с помощью букв:

- Для добавления прав используется `+`
- Для удаления прав используется `-`
- Для установки конкретных прав используется `=`

Примеры:

- Добавить право на выполнение для владельца

$ chmod u+x filename

- Удалить право на запись для группы

$ chmod g-w filename

- Установить права на чтение и выполнение для всех

$ chmod a=rx filename

В цифровой форме права представляются числами:

- 4 — право на чтение (r)
- 2 — право на запись (w)
- 1 — право на выполнение (x)

Суммируя эти значения, можно задать права:

- 7 (4+2+1) — чтение, запись и выполнение (rwx)
- 6 (4+2) — чтение и запись (rw-)
- 5 (4+1) — чтение и выполнение (r-x)
- 4 — только чтение (r--)
- 3 (2+1) — запись и выполнение (wx)
- 2 — только запись (w-)
- 1 — только выполнение (x)
- 0 — отсутствие прав (---)

Пример:

Требуется установить права *rwxr-xr--* (владелец: чтение, запись, выполнение; группа: чтение, выполнение; остальные: только чтение)

```
adminmsc@ru01wks001:~$ chmod 755 file
adminmsc@ru01wks001:~$ ls -l file
-rwxr-xr-x 1 adminmsc adminmsc 0 окт 28 16:29 file
```

1.6. Опции команды

- Для того чтобы применить изменения рекурсивно ко всем файлам и подкаталогам в указанной директории, нужно использовать ключ *-R*

```
adminmsc@ru01wks001:~$ chmod -R 755 directory/
adminmsc@ru01wks001:~$ ls -l directory/
итого 0
-rwxr-xr-x 1 adminmsc adminmsc 0 окт 28 16:31 file1
-rwxr-xr-x 1 adminmsc adminmsc 0 окт 28 16:31 file2
```

- Для вывода информации о каждом изменении воспользоваться ключом *-v*

```
adminmsc@ru01wks001:~$ chmod -v u+x file
права доступа 'file' изменены с 0644 (rw-r--r--) на 0744 (rwxr--r--)
```

2. Команда *chown*. Используется для изменения владельца файла или каталог, а также его группы. Основной синтаксис команды *chown*:

chown [опции] владелец:группа файл_или_каталог

- *владалец* — имя пользователя, которому будет принадлежать файл или каталог
- *группа* — группа, которая будет владеть файлом или каталогом (указывать необязательно)
- *файл_или_каталог* — файл или каталог, для которого нужно изменить владельца и/или группу

Если вы хотите изменить владельца файла на другого пользователя, то команда будет следующей:

$ chown tmp_user file.txt

Изменение владельца и группы:

$ chown tmp_user:group1 file.txt

Изменение только группы:

$ chown :group1 file.txt

3. Команда *chgrp*. Меняет только группу для файла или каталога. Основной синтаксис команды *chgrp:*

chgrp [опции] группа файл_или_каталог

- *группа* — название группы, которая должна стать владельцем файла или каталога
- *файл_или_каталог* — имя файла или каталога, для которого нужно изменить группу

1.4. Специальные атрибуты файлов

В Unix-подобных ОС помимо стандартных прав на чтение, запись и выполнение (*rwx*), существуют специальные атрибуты.

Указанные атрибуты приведены в таблице.

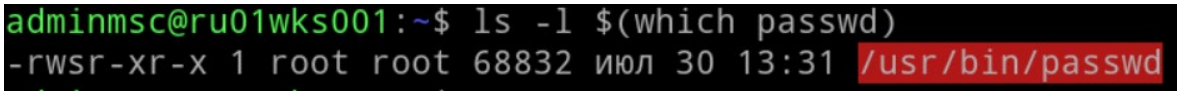
Права	Численное значение	Относительный режим	Применение к файлам	Применение к каталогам
SUID	4	u+s	Пользователь выполняет файл с разрешениями владельца файла	-
SGID	2	g+s	Пользователь выполняет файл с разрешениями владельца группы	Файлы, созданные в каталоге, получают одного владельца группы
Sticky bit	1	+t	-	Запрещает пользователям удалять файлы других пользователей

Рассмотрим их подробнее.

1.4.1. SUID (Set User ID)

Описание: этот бит задается для исполняемых файлов и позволяет временно получить права владельца файла при выполнении программы. Это важно, если программе нужно выполнять действия с правами, которых у запускающего пользователя нет

Если установлен бит SUID, то в строке прав доступа вместо обычного бита выполнения для владельца (*x*) будет стоять `*s*`



На рисунке выше приведен пример использования бита SUID. Таким образом, можно сделать вывод, что команда *passwd*, работающая с файлами, доступ к которым имеет только суперпользователь, запускается с правами root даже если её запускает обычный пользователь

Чтобы установить бит SUID на исполняемом файле, используется команда *chmod*:

$ chmod u+s <filename>

1.4.2. SGID (Set Group ID)

Описание: этот бит можно установить как для файлов, так и для директорий. Если бит SGID установлен на файл, то программа будет выполняться с правами группы файла, а не группы пользователя, запустившего её. Для директорий SGID меняет поведение создания новых файлов: новые файлы в такой директории наследуют группу этой директории, а не группу создателя файла

Если установлен SGID, то в строке прав доступа вместо обычного бита выполнения для группы (*x*) будет стоять `*s*`

Чтобы установить бит SGID на исполняемом файле, используется команда *chmod*:

$ chmod g+s <filename>

1.4.3. Sticky Bit

Описание: Sticky Bit используется преимущественно для директорий. Если он установлен, то файлы в такой директории могут удалять или изменять только владельца этих файлов, даже если у других пользователей есть права на запись в эту директорию

Если установлен Sticky Bit, в строке прав доступа будет буква `*t*` вместо обычного `*x*` в правах для остальных

```
adminmsc@ru01wks001:~$ ls -l / | grep tmp
drwxrwxrwt  12 root root       4096 окт 28 16:42 tmp
```

Директория */tmp* — пример использования Sticky Bit. Это общедоступная директория, куда любой пользователь может записывать файлы, но удалять их могут только владельцы этих файлов

Чтобы установить бит SGID на исполняемом файле, используется команда *chmod*:

$ chmod +t /path/to/directory

1.4.4. Числовой метод установки атрибутов

Можно также использовать числовой метод для установки прав доступа. Для этого добавляются дополнительные цифры перед основными правами:

- SUID = 4

  $ chmod 4755 <filename>

- SGID = 2

  $ chmod 2755 <directory>

- Sticky Bit = 1

  $ chmod 1755 <directory>

  Для одновременной установки битов:

  $ chmod 7755 <directory>

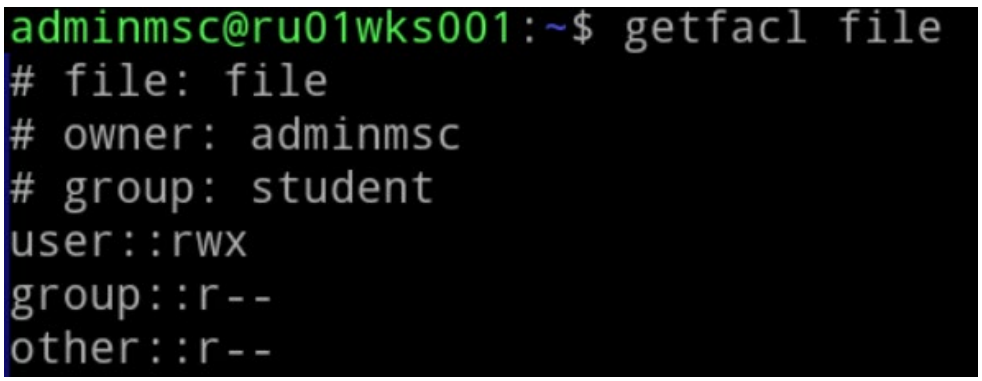
1.4.5. Access Control Lists (ACL)

ACL представляет более гибкий способ управления доступом к файлам и директориям, позволяя задавать права для конкретных пользователей и групп. Стандартная система прав доступа (*rwx*) ограничена тремя уровнями: владелец, группа, остальные. С помощью ACL можно настроить права для отдельных пользователей или дополнительных групп, что делает управление доступом более точным

1.4.5.1. Основные команды для работы с ACL

- Просмотр ACL. Чтобы просмотреть текущие ACL для файла или директории, используется следующая команда:

$ getfacl <filename>



- Чтобы добавить или изменить права для конкретного пользователя, используется команда *setfacl*:

$ setfacl u:username:permissions <filename>

  - *u:username:permissions* — задает права для пользователя (*u*), где *username* — это имя пользователя, а *permissions* — права (*r*, *w*, *x*)

- Удаление прав доступа. Чтобы удалить ACL-запись для конкретного пользователя или группы, используется команда:

$ setfacl -x u:username <filename>

- Наследование прав в директориях. Для каталогов можно настроить так, чтобы файлы и подкаталоги автоматически наследовали ACL от родительской директории. Это делается с флагом *-d*. Например, чтобы настроить наследование для всех новых файлов в каталоге, необходимо ввести следующее:

$ setfacl -m d:u:username:rwx <directory>

1.5. Особенности и ограничения DAC

- Гибкость, но отсутствие строгого контроля. DAC предоставляет пользователям большую гибкость. Владелец может в любой момент изменить права на свой файл. Однако, это также представляет собой определённую угрозу безопасности: если владелец неправильно настроит права доступа, данные могут стать доступны для несанкционированных пользователей
- Наследование прав. В обычной реализации DAC в Linux права доступа к файлам и каталогам не наследуются автоматически от родительского каталога. Например, если у вас есть каталог с ограниченными правами, новый файл, созданный в этом каталоге, может иметь другие права доступа, зависящие от настроек пользователя

- Отсутствие защиты от взлома привилегий. Так как владелец ресурса может изменять права доступа, в случае компрометации учетной записи владельца злоумышленник может изменить права на ресурсы и получить доступ к ним

Дискреционное управление доступом в Linux является гибкой и удобной моделью, которая позволяет владельцам файлов и каталогов самостоятельно управлять доступом к своим ресурсам. Это одна из основных моделей управления доступом, широко применяемая в системах с низкими и средними требованиями к безопасности. Однако для более строгого контроля доступа, особенно в системах с высокими требованиями к безопасности, могут потребоваться дополнительные механизмы, такие как мандатное управление доступом

2. Практическая часть

Пользователи соответствуют созданным в предыдущей лабораторной работе. Для простоты будем обозначать их user1, user2, user3. Основного пользователя системы будем называть adminstd.

2.1. Основы прав доступа

2.1.1. От имени пользователя **adminstd** создайте каталог *lab6* в директории /tmp. Выдайте на каталог права на чтение (r), запись (w) и вход (x) для всех пользователей.

2.1.2. Внутри каталога *lab6* создайте от имени пользователя **user1** файл file1.txt с содержимым “TEXT OF FILE1.TXT” и установите для него следующие права:

- Владелец имеет право на чтение и запись
- Группа имеет только право на чтение
- Остальные пользователи не имеют доступа к файлу

2.1.3. Проверьте права доступа к файлу

2.1.4. Попробуйте прочитать файл от имени пользователей **user2** и **user3.** Включите пользователя **user2** в группу **user1** и повторите попытку чтения.

2.2. Управление владельцем и группой файла

2.2.1. От имени пользователя **adminstd** внутри каталога lab6 создайте файл *file2.txt* с содержимым “TEXT OF FILE2.TXT”.

2.2.2. Создайте группу *fileowners* и назначьте её основной группой для файла *file2.txt.* Проверьте, что права на файл соответствуют изменениям

2.2.3. Добавьте user1 в группу *fileowners* и проверьте доступ к файлу от имени **user2** и **user1.**

2.3. Sticky Bit

2.3.1. Создайте директорию *dir1* и установите для нее права, позволяющие чтение, запись и выполнение для всех пользователей.

2.3.2. Добавьте в директорию файл *file3.txt* от имени одного пользователя и попробуйте удалить его с другого пользователя

2.3.3. Установите Sticky Bit на директорию и повторите попытку удаления файла. Что изменилось?

2.4. SGID

2.4.1. Создайте директорию *dir2*, установите на нее SGID-бит. Внутри этой директории создайте файл от имени одного пользователя и проверьте, какой группой владеет этот файл

2.5. Управление правами с помощью ACL

2.5.1. Создайте файл *file4.txt* и установите для него стандартные права (например, чтение и запись для владельца, чтение для группы и других пользователей).

2.5.2. Используя команды ACL, добавьте для пользователя *user1* права на запись в этот файл без изменения стандартных прав.

2.5.3. Проверьте, какие права на файл имеет пользователь *user1*, и попытайтесь изменить содержимое файла от имени этого пользователя. Аналогично попытайтесь изменить файл от пользователя *user2*.

2.5.4. Удалите права записи для *user1* с помощью ACL и убедитесь, что доступ ограничен.

2.6. Комбинирование команд для сложных сценариев

2.6.1. Создайте директорию *shared_dir*, в которой:

- Ограничено удаление файлов
- Новые файлы наследуют группу директории
- Пользователю *user2* предоставлены права на чтение и запись

2.6.2. Добавьте несколько файлов в директорию от имени разных пользователей и проверьте, как наследуются права и группы, а также как работают ограничения на удаление файлов

Контрольные вопросы

1. Как работает механизм наследования прав доступа с SGID на директории? Объясните, что произойдет с правами доступа к файлам, созданным в такой директории пользователями из разных групп. Укажите, как SGID влияет на групповые права в этом случае.

2. Может ли пользователь, не являющийся владельцем файла и не имеющий на него стандартных прав записи, получить доступ к изменению этого файла с помощью механизма ACL? Опишите возможные ситуации, в которых права,

установленные через ACL, изменяют поведение системы контроля доступа. Приведите пример команд.

3. Какие риски безопасности могут возникнуть при неправильной настройке битов SUID, SGID или Sticky Bit для исполняемых файлов и директорий? Опишите примеры уязвимостей, которые могут быть использованы злоумышленниками, если эти биты настроены некорректно. Как предотвратить такие риски?