1. Министерство науки и высшего образования Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. **Институт кибербезопасности и защиты информации**

**ЛАБОРАТОРНАЯ РАБОТА № 5**

1. «Контроль доступа в ОС Linux»
2. по дисциплине «Безопасность операционных систем»
3. Выполнил
4. студент гр. 4851003/70801 Гасанов Э.А.

<*подпись*>

1. Преподаватель
2. ассистент Ахтямов Д.Р.

<*подпись*>

1. Санкт-Петербург
2. 2020

# Цель работы

Изучение подсистемы контроля доступа ОС Linux.

# Задачи

1. Изучить механизмы разграничения доступа в ОС Linux: стандартный дискреционный контроль доступа (DAC) Linux, списки контроля доступа (ACL) и систему мандатного управления доступом SELinux.
2. Описать три ситуации, когда стандартного механизма разграничения доступа недостаточно, и продемонстрировать эти ограничения, используя классическую проверку прав доступа и списки контроля доступа.
3. Сконфигурировать SELinux таким образом, чтобы предотвратить возможные нарушения доступа из пункта 2.

# Ход работы

Базовая модель контроля доступа в Linux позволяет задать следующие права для файлов и каталогов (таблица 1).

|  |  |  |
| --- | --- | --- |
|  | Действие прав | |
| Права | Файлы | Каталоги |
| r | Чтение содержимого файлов | Просмотр содержимого директории |
| w | Запись в файлы | Создание файлов и каталогов |
| x | Исполнение файлов | Переход в каталог |

Таблица 1. Базовые права в Linux

Права можно устанавливать для каждой из 3 групп пользователей, связанных с объектом:

* Владелец — пользователь, назначенный владельцем объекта
* Группа-владелец — назначенная группа
* Остальные

Так же дополнительно у объектов есть ещё 3 особых права

(таблица 2).

|  |  |  |
| --- | --- | --- |
|  | Действие прав | |
| Права | Файлы | Каталоги |
| SetUID | Исполнение файл от имени владельца. | Обычно не определён и игнорируется. В некоторых системах имеет аналогичное SetGID поведение. |
| SetGID | Исполнение файл от имени группы-владельца. | Созданные в этом каталоге новые файлы и каталоги по умолчанию наследуют группу-владельца. Каталоги также наследуют SetGID. |
| Sticky bit | Не поддерживается. | Удалять или переименовывать файлы в каталоге могут только их владельцы, члены группы-владельца, или root. |

Таблица 2. Особые права в Linux

Основные проблемы такого подхода разграничения прав заключаются в следующем:

1. Невозможность удобным разграничить права для конкретных пользователей (например, запретить доступ к файлу только одному конкретному пользователю), а также невозможность назначить разные права 4 и более пользователям.
2. Ограниченный набор прав, который не разграничивает, например, запись и дописывание в файл. Или, разграничение на удаление и переименование файлов возможно только для каталога целиком, а также эти 2 права неотделимы друг от друга.

Для решение первой проблемы можно использовать ACL (Access Control List). С их помощью можно устанавливать права для любых пользователей или групп. Есть 2 вида ACL:

1. ACL для доступа — ACL для данного файла или каталога
2. ACL по умолчанию — может быть назначен для каталога и будет применён к файлам внутри него, если у тех нету своих ACL.

Для решения обеих проблем, и, в частности, второй, можно использовать систему SELinux. Это система контроля доступа, которая является надстройкой над базовой моделью в Linux. Если доступ разрешён согласно базовой модели, то обращение перехватывается системой SELinux и окончательное решение принимается согласно установленной политике. В политике все действия, не разрешённые явно, считаются запрещёнными. Основные понятия, связанные с SELinux:

1. Домен — набор действий, разрешённых процессу. Рекомендуется делать домены минимально достаточными для работы процесса.
2. Роль —список доменов, которые могут быть применены.
3. Тип — набор действий, допустимых по отношению к объекту. Тип, во отличие от домена, может применяться к различным объектам, таким как файлы, каталоги, сокеты и другие.
4. Контекст безопасности — все перечисленные выше атрибуты SELinux, заданные в системе.

Существует 3 режима работы SELinux:

1. Enforcing. Режим по умолчанию. При выборе этого режима все действия, которые каким-то образом нарушают текущую политику безопасности, будут блокироваться, а попытка нарушения будет зафиксирована в журнале;
2. Permissive. В случае использования этого режима информация обо всех действиях, которые нарушают текущую политику безопасности, будут зафиксированы в журнале, но сами действия не будут заблокированы;
3. Disabled. Полное отключение системы принудительного контроля доступа.

В системе SELinux определены следующие 17 возможных полномочий по отношению к файлу (таблица 3).

|  |  |
| --- | --- |
| append | Запись в конец файла |
| create | Создание файла |
| execute | Исполнение файла с переключением домена |
| getattr | Получение атрибутов файла |
| ioctl | Системные запросы ввода/вывода |
| link | Создание жесткой ссылки |
| lock | Установка и снятие блокировки |
| mounton | Использование в качестве точки монтирования |
| quotaon | Разрешает квоты |
| read | Чтение содержимого |
| relabelfrom | Смена текущего типа на другой |
| relabelto | Смена предыдущего типа на текущий |
| rename | Смена имени файла |
| setattr | Смена атрибутов файла |
| swapon | Разрешить использовать файл для свопинга. (не используется) |
| unlink | Удаление файла или жесткой ссылки |
| write | Запись в файл (в том числе в конец (append)) |

Таблица 3. Список прав по отношению к файлу в SELinux

# Применение политик SElinux

Мы будем использовать CentOS8 с уже запущенным SELinux:

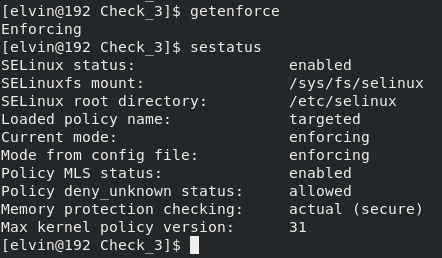


Рисунок 1 – Запущенный SELinux.

Примеры работы политик будут показаны ниже, а здесь будет описано применение политик в силу.

|  |
| --- |
| policy\_module(lab,1.0)  require {  type unconfined\_t;  type setroubleshootd\_t;  }  type rule1\_t;  fs\_associate(rule1\_t);  allow unconfined\_t rule1\_t:{ file} {relabelto getattr append open read};  allow setroubleshootd\_t rule1\_t:{ file} {relabelto getattr append open read};  type rule2\_t;  fs\_associate(rule2\_t);  allow unconfined\_t rule2\_t:file {relabelto getattr read write open unlink};  allow setroubleshootd\_t rule2\_t:file {relabelto getattr read write open unlink};  type rule3\_t;  fs\_associate(rule3\_t);  allow unconfined\_t rule3\_t:file {relabelto getattr read write open rename};  allow setroubleshootd\_t rule3\_t:file {relabelto getattr read write open rename}; |

Листинг 1 – Устанавливаемая политика для 3 случаев.

Политика разграничивает доступ к файлам соответствующих типов со стороны доменов *unconfined\_t* и *setroubleshootd\_t*. Для применения политики её необходимо сохранить в файл с расширением .te. Файл был назван lab.te.

Для того чтобы скомпилировать политику через Makefile, нужно установить его:

|  |
| --- |
| [root@192 te\_files]# yum install policycoreutils-devel |

Затем необходимо собрать политику командой:

|  |
| --- |
| make -f /usr/share/selinux/devel/Makefile lab.pp |

Собранную политику необходимо применить командой *semodule -i lab.pp*

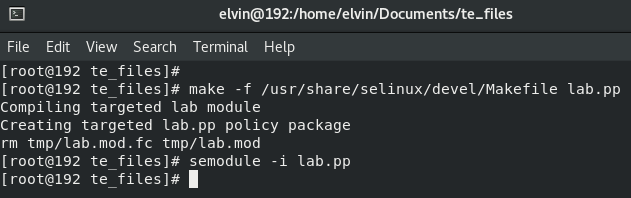


Рисунок 2 – Применение политик.

После этого необходимо назначить желаемым файлам новые типы(контекст), и с этого момента на них будут распространяться соответствующие ограничения. Делается это при помощи команды chcon.

Применим эти политики на файлы test1.txt, test2.txt, test3.txt:

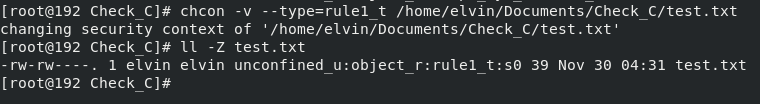


Рисунок 3 – применение rule1\_t на test1.txt

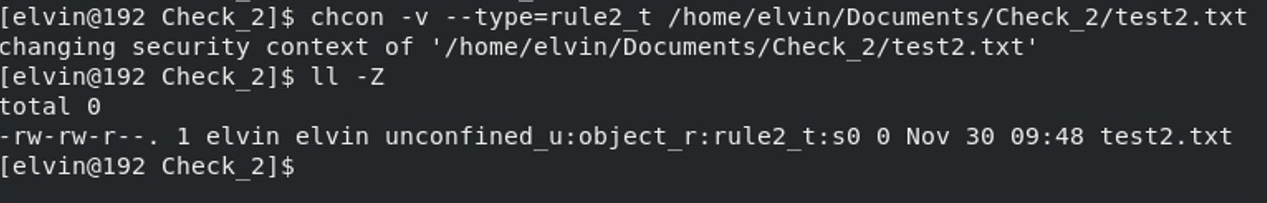


Рисунок 4 – Применение rule2\_t на test2.txt.

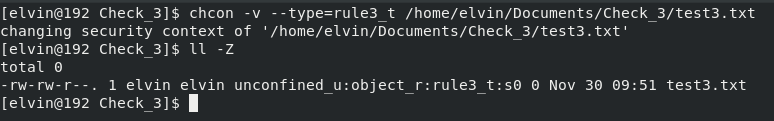


Рисунок 5 – Применение rule3\_t на test3.txt.

# Демонстрация ситуаций

Были составлены следующие 3 ситуации для демонстрации недостатков стандартных мер контроля доступа и возможностей SELinux:

1. Необходимо разрешить запись в конец файл, но запретить перезапись;
2. Запретить переименовывать файл.
3. Запретить удалять файл.
4. Для начальной проверки будем использовать тестовую программу, которая имеет две функции, выполняющаяся одна за другой:

* Whole\_file\_to\_rewrite – перезапись файла сначала. Используется fopen c параметром “w”. Тогда каждый новый вызов этой функции перезаписывает файл.
* Append – до запись в конец файла. Используется fopen c параметром “a”. Тогда каждый новый вызов этой функции дописывает данные в конец файла.

Мы воспользуемся важной особенностью fprintf – эта функция возвращает количество байтов, которая она записала в файл. По этому значению мы будем ориентироваться успешно ли прошла запись в файл или нет.

На рисунке ниже видно, что сначала мы изменили с помощью команды chmod права на чтение и запись пользователю elvin в группе elvin.

Затем запустили тестовое приложение, которое делает проверку, какой тип записи в файл возможен. Пока что возможны оба варианта, так как стандартные права не так гибки, как хотелось бы.



Рисунок 6 – Изменение прав на чтение и запись.

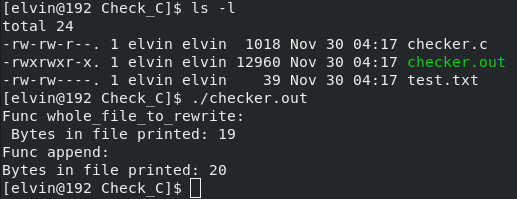


Рисунок 7 –test.txt с правами на чтение запись (и дозапись в конец файла).

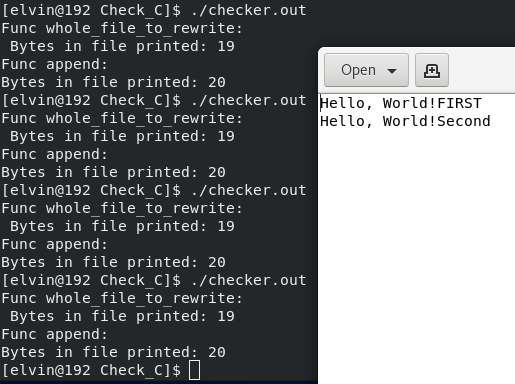


Рисунок 8 – Происходит постоянная перезапись файла с начала из-за первой функции.

Наша же задача состоит в том, чтобы разрешить запись в конец файла, но запретить его полную перезапись.

Тогда первая функция (которая пытается перезаписать файл) закончится неудачей. А дозапись сработает:

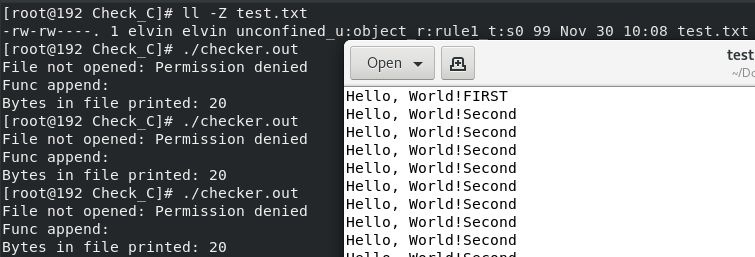


Рисунок 9 – Дозапись в файл позволительна, а перезапись – нет, так как применено первое правило.

1. Во второй ситуации мы хотим показать, что с помощью стандартных прав мы не сможем запретить переименовывать файл.

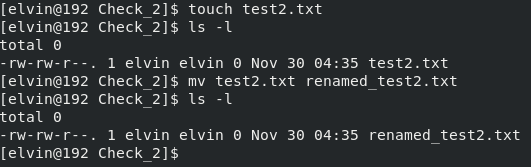


Рисунок 10 – Пока что успешное переименовывание файла.

Но с помощью SELinux это возможно:

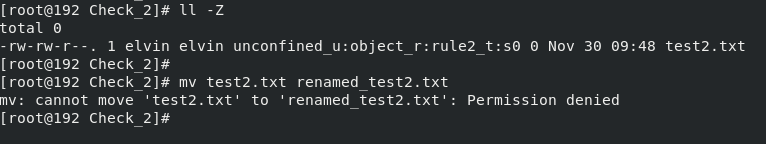


Рисунок 11 – Запрет переименовывания в силе, так как установлено rule2.

1. C помощью DAC не удаётся разрешить чтение, запись, переименовывание и запрет на удаление одновременно.

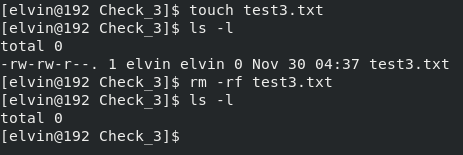


Рисунок 12 – Чтение, запись, переименовывание и удаление возможны одновременно.

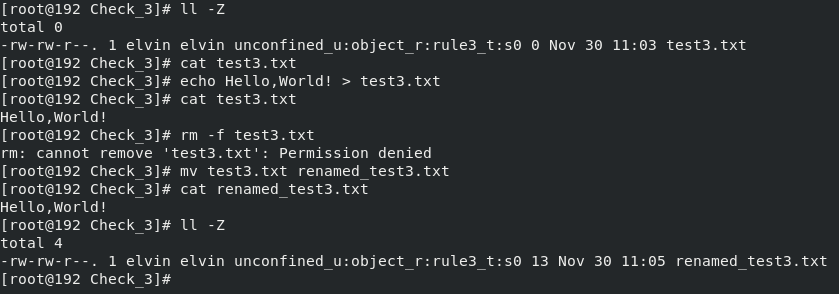


Рисунок 13 – Пример применения политики rule3.

Рассмотрим представленные действия по порядку:

1. Вывод всех файлов в каталоге. Флаг -Z выволит так же информацию касательно SELinux. Можно видеть, что файл test3.txt имеет тип rule3\_t.
2. Вывод содержимого файла. Файл пустой
3. Запись информации в файл. Доступ на запись есть.
4. Демонстрация нового одержимого файла. Запись прошла успешно.
5. Попытка удалить файл. Удаление запрещено политикой.
6. Попытка смена имени файла. Успех.
7. Вывод содержимого файла. Содержимое не изменилось

Тип файла не изменился.

# Вывод

Была изучена стандартная модель управления доступом в Linux. Были получены знания об ACL в Linux. Были получены практические навыки по работе с системой SELinux, в частности, навыки по настройке политики. Далее было составлено три ситуации, демонстрирующие ограниченность стандартного контроля доступа в Linux с ACL. Была написана политика для SELinux, позволяющая реализовать правила для всех трёх ситуаций.

# Приложение

Политика

policy\_module(lab,1.0)

require {

type unconfined\_t;

type setroubleshootd\_t;

}

type rule1\_t;

fs\_associate(rule1\_t);

allow unconfined\_t rule1\_t:{ file} {relabelto getattr append open read};

allow setroubleshootd\_t rule1\_t:{ file} {relabelto getattr append open read};

type rule2\_t;

fs\_associate(rule2\_t);

allow unconfined\_t rule2\_t:file {relabelto getattr read write open unlink};

allow setroubleshootd\_t rule2\_t:file {relabelto getattr read write open unlink};

type rule3\_t;

fs\_associate(rule3\_t);

allow unconfined\_t rule3\_t:file {relabelto getattr read write open rename};

allow setroubleshootd\_t rule3\_t:file {relabelto getattr read write open rename};

Тестовая программа проверки записи/дозаписи

#include<stdio.h>

void whole\_file\_to\_rewrite(void)

{

//С помощью переменной file будем осуществлять доступ к файлу

FILE \*file;

//Открываем текстовый файл с правами на запись

file = fopen("/home/elvin/Documents/Check\_C/test.txt", "w");

if(file==NULL)

{

perror("File not opened");

return;

}

//Пишем в файл

int count=fprintf(file, "Hello, World!FIRST\n");

printf("Func whole\_file\_to\_rewrite:\n Bytes in file printed: %d\n",count);

//Закрываем файл

fclose(file);

}

void append(void)

{

FILE \*file;

//Открываем текстовый файл с правами на дозапись

file = fopen("/home/elvin/Documents/Check\_C/test.txt", "a");

if(file==NULL)

{

perror("File not opened(2)");

return;

}

//Пишем в файл

int count=fprintf(file, "Hello, World!Second\n");

printf("Func append:\nBytes in file printed: %d\n",count);

//Закрываем файл

fclose(file);

}

int main()

{

whole\_file\_to\_rewrite();

append();

return 0;

}