Отчет по лабораторной работе №7

Основы информационной безопасности

Чванова Ангелина Дмитриевна

Содержание

Список иллюстраций

Список таблиц

Цель работы

Развить навыки администрирования ОС Linux. Получить первое практическое знакомство с технологией SELinux1. Проверить работу SELinx на практике совместно с веб-сервером Apache.

Теоретическое введение

SELinux (Security-Enhanced Linux)

— обеспечивает усиление защиты путем внесения изменений как на уровне ядра, так и на уровне пространства пользователя, что превращает ее в действительно «непробиваемую» операционную систему. Впервые эта система появилась в четвертой версии CentOS, а в 5 и 6 версии реализация была существенно дополнена и улучшена.

SELinux имеет 3 основных режим работы:

- 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.
- Permissive: в случае использования этого режима, информация о всех действиях, которые нарушают текущую политику безопасности, будут зафиксированы в журнале, но сами действия не будут заблокированы.
- Disabled: полное отключение системы принудительного контроля доступа.

Политика SELinux определяет доступ пользователей к ролям, доступ ролей к доменам и доступ доменов к типам. Контекст безопасности — все атрибуты SELinux — роли, типы и домены.

Apache

— это свободное программное обеспечение, с помощью которого можно создать веб-сервер. Данный продукт возник как доработанная версия другого HTTP-клиента от национального центра суперкомпьютерных приложений (NCSA).

Для чего нужен Apache сервер:

- чтобы открывать динамические PHP-страницы,
- для распределения поступающей на сервер нагрузки,
- для обеспечения отказоустойчивости сервера,
- чтобы потренироваться в настройке сервера и запуске PHP-скриптов.

Apache является кроссплатформенным ПО и поддерживает такие операционные системы, как Linux, BSD, MacOS, Microsoft, BeOS и другие.

Выполнение лабораторной работы

Для начала был выполен вход в систему под своей учетной записью. После чего необходимо было проверить, что SELinux работает в режиме enforcing политики targeted с помощью команд getenforce и sestatus (рис. [-@fig:001]).

Рис. 1: проверка режима работы SELinux

Запускаем сервер apache, далее обращаемсь с помощью браузера к веб-серверу, запущенному на компьютере, он работает, что видно из вывода команды `service httpd status` (рис. [-@fig:002]).

Рис. 2: Проверка работы Apache

С помощью команды `ps auxZ | grep httpd` находим веб-сервер Apache в списке процессов. Его контекст безопасности - httpd_t (рис. [-@fig:003]).

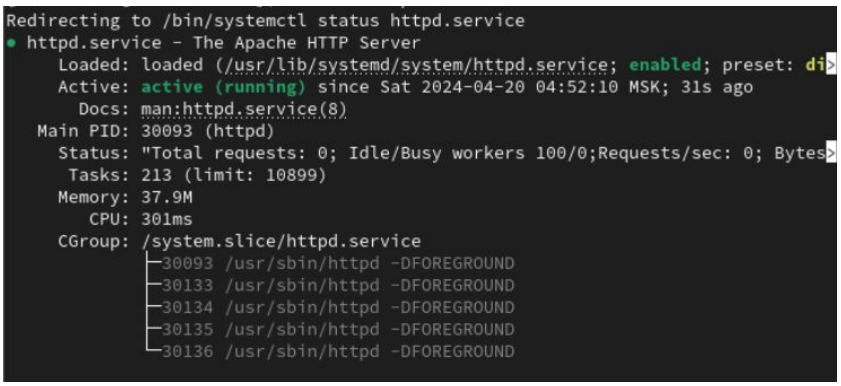

Рис. 3: Контекст безопасности Apache

Просмотрим текущее состояние переключателей SELinux для Apache с помощью команды `sestatus -bigrep httpd` (рис. [-@fig:004]).

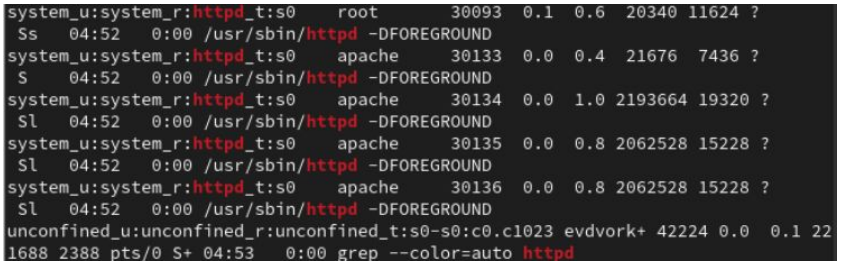

Рис. 4: Состояние переключателей SELinux

Просмотрим статистику по политике с помощью команды `seinfo`. Множество пользователей - 8, ролей - 39, типов - 5135. (рис. [-@fig:005]).

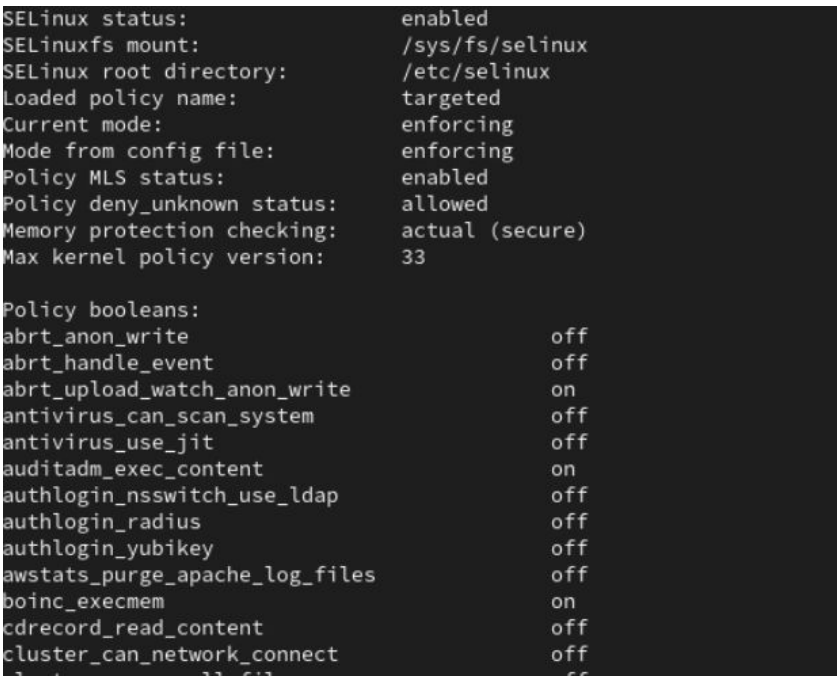

Рис. 5: Статистика по политике

Типы поддиректорий, находящихся в директории `/var/www`, с помощью команды `ls -lZ /var/www` следующие: владелец - root, права на изменения только у владельца. Файлов в директории нет (рис. [-@fig:006]).

```
Statistics for policy file: /sys/fs/selinux/policy
Policy Version:             33 (MLS enabled)
Target Policy:              selinux
Handle unknown classes:     allow
  Classes:             135    Permissions:         457
  Sensitivities:         1    Categories:         1024
  Types:              5135    Attributes:          259
  Users:                 8    Roles:                15
  Booleans:            357    Cond. Expr.:         390
  Allow:             65409    Neverallow:            0
  Auditallow:          172    Dontaudit:          8647
  Type_trans:       267813    Type_change:          94
  Type_member:          37    Range_trans:        6164
  Role allow:           39    Role_trans:          419
  Constraints:          70    Validatetrans:         0
  MLS Constrain:        72    MLS Val. Tran:         0
  Permissives:           2    Polcap:                6
  Defaults:              7    Typebounds:            0
  Allowxperm:            0    Neverallowxperm:       0
  Auditallowxperm:       0    Dontauditxperm:        0
  Ibendportcon:          0    Ibpkeycon:             0
  Initial SIDs:         27    Fs_use:               35
  Genfscon:            109    Portcon:             665
  Netifcon:              0    Nodecon:               0
```

Рис. 6: Типы поддиректорий

В директории `/var/www/html` нет файлов. (рис. [-@fig:007]).

```
итого 0
drwxr-xr-x. 2 root root system_u:object_r:httpd_sys_script_exec_t:s0 6 окт 28 12:35 cgi-bin
drwxr-xr-x. 2 root root system_u:object_r:httpd_sys_content_t:s0     6 окт 28 12:35 html
```

Рис. 7: Типы файлов

Создать файл может только суперпользователь, поэтому от его имени создаем файл touch.html со следующим содержанием:

```
<html>
<body>test</body>
</html>
```

(рис. [-@fig:008]).

```
sudo touch /var/www/html/test.html
```

Рис. 8: Создание файла

Проверяем контекст созданного файла. По умолчанию это httpd_sys_content_t (рис. [-@fig:009]),(рис. [-@fig:010]).

```
sudo nano /var/www/html/test.html
sudo cat /var/www/html/test.html
```

```
итого 4
-rw-r--r--. 1 root root unconfined_u:object_r:httpd_sys_cor
```

Обращаемся к файлу через веб-сервер, введя в браузере адрес http://127.0.0.1/test.html. Файл успешно отображается (рис. [-@fig:011]).

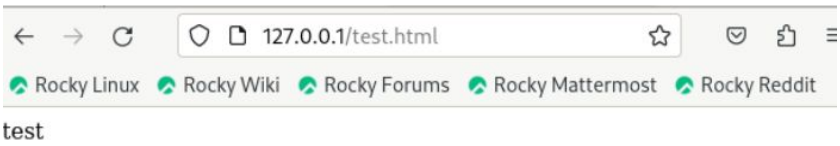

Рис. 9: Отображение файла

Рассмотрим полученный контекст детально. Так как по умолчанию пользователи CentOS являются свободными от типа (unconfined в переводе с англ. означает свободный), созданному нами файлу test.html был сопоставлен SELinux, пользователь unconfined_u. Это первая часть контекста. Далее политика ролевого разделения доступа RBAC используется процессами, но не файлами, поэтому роли не имеют никакого значения для файлов. Роль object_r используется по умолчанию для файлов на «постоянных» носителях и на сетевых файловых системах. (В директории /proc файлы, относящиеся к процессам, могут иметь роль system_r. Если активна политика MLS, то могут использоваться и другие роли, например, secadm_r. Данный случай мы рассматривать не будем, как и предназначение :s0). Тип httpd_sys_content_t позволяет процессу httpd получить доступ к файлу. Благодаря наличию последнего типа мы получили доступ к файлу при обращении к нему через браузер. (рис. [-@fig:012]).

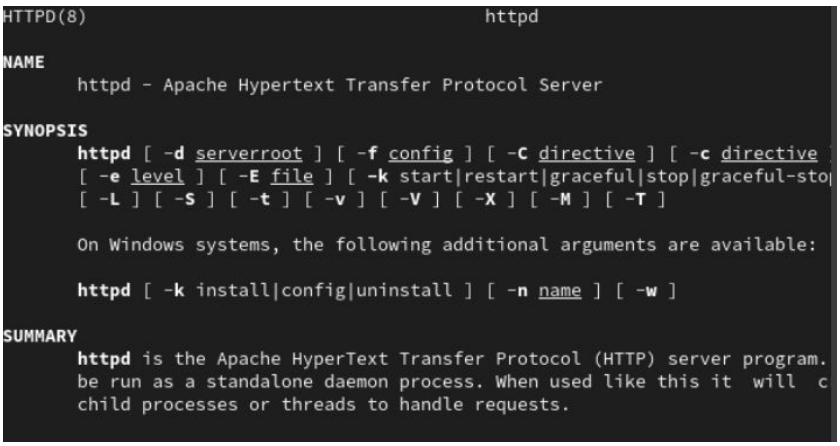

Рис. 10: Изучение справки по команде

Изменяем контекст файла `/var/www/html/test.html` с `httpd_sys_content_t` на любой другой, к которому процесс httpd не должен иметь доступа, например, на `samba_share_t`: `chcon -t samba_share_t /var/www/html/test.html ls -Z /var/www/html/test.html` Контекст действительно поменялся (рис. [-@fig:013]).

Рис. 11: Изменение контекста

При попытке отображения файла в браузере получаем сообщение об ошибке (рис. [-@fig:014]).

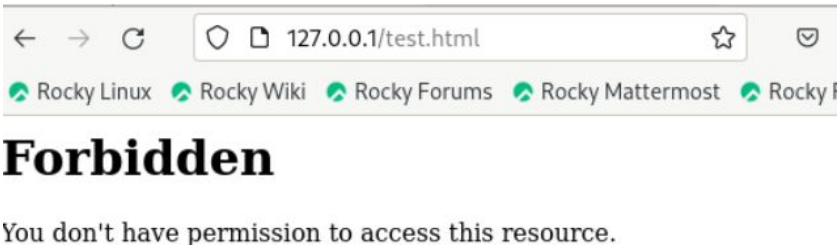

Рис. 12: Отображение файла

файл не был отображён, хотя права доступа позволяют читать этот файл любому пользователю, потому что установлен контекст, к которому процесс httpd не должен

иметь доступа.

Просматриваем log-файлы веб-сервера Apache и системный лог-файл: `tail /var/log/messages`. Если в системе окажутся запущенными процессы setroubleshootd и audtd, то также можно увидеть ошибки, аналогичные указанным выше, в файле /var/log/audit/audit.log. (рис. [-@fig:015]).

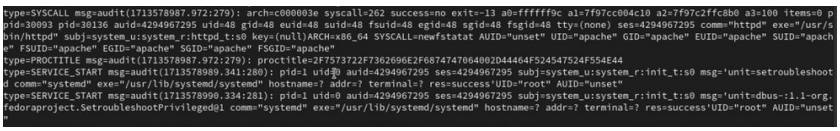

Рис. 13: Попытка прочесть лог-файл

Чтобы запустить веб-сервер Apache на прослушивание TCP-порта 81 (а не 80) открываем файл /etc/httpd/httpd.conf для изменения. (рис. [-@fig:016]).

Рис. 14: Изменение файла

Находим строчку Listen 80 и заменяем её на Listen 81. (рис. [-@fig:017]).

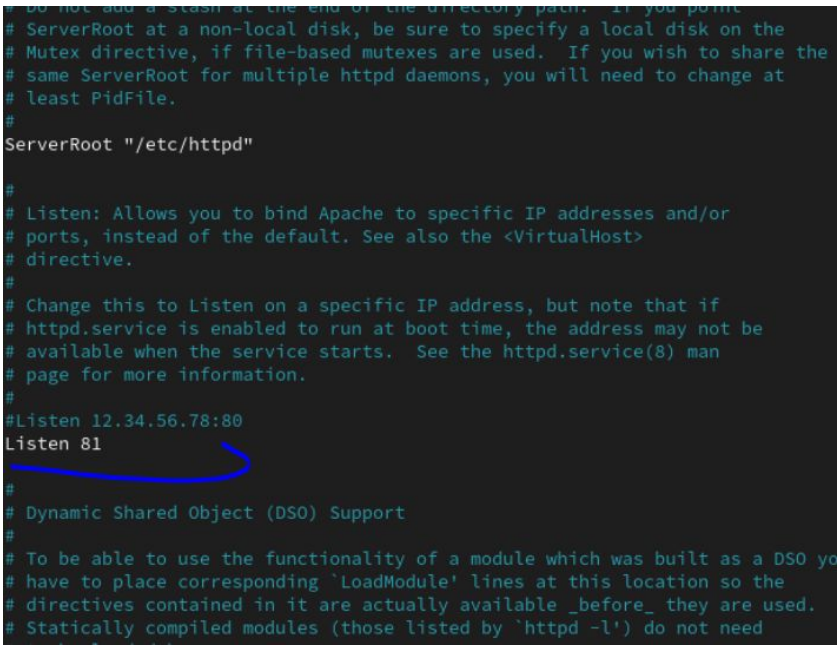

Рис. 15: Изменение порта

После чего выполняем перезапуск веб-сервера Apache. Произошёл сбой, потому что порт 80 для локальной сети, а 81 нет (рис. [-@fig:018]).

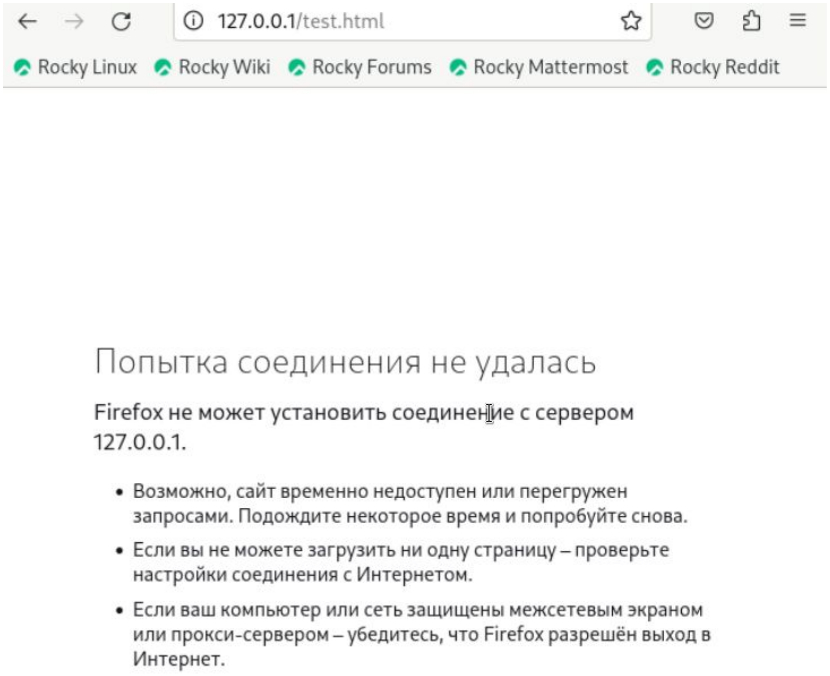

Рис. 16: Попытка прослушивания другого порта

Проанализируем лог-файлы:

`tail -nl /var/log/messages` (рис. [-@fig:019]).

Рис. 17: Проверка лог-файлов

Просмотрим файлы `/var/log/http/error_log`, `/var/log/http/access_log` и `/var/log/audit/audit.log` и выясним, в каких файлах появились записи. Запись появилась в файлу error_log (рис. [-@fig:020]).

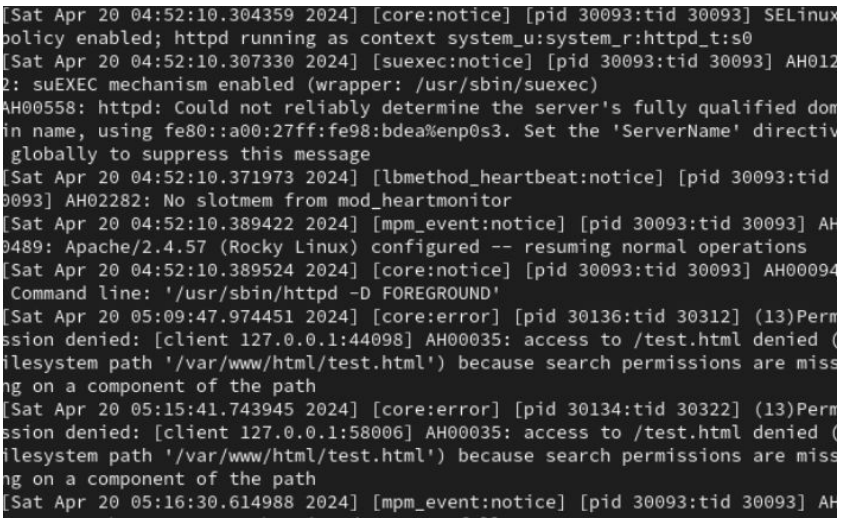

Рис. 18: Проверка лог-файлов

Выполняем команду `semanage port -a -t http_port_t -p tcp 81` После этого проверяем список портов командой `semanage port -l | grep http_port_t` Порт 81 появился в списке (рис. [-@fig:021]).

Рис. 19: Проверка портов

Перезапускаем сервер Apache (рис. [-@fig:022]).

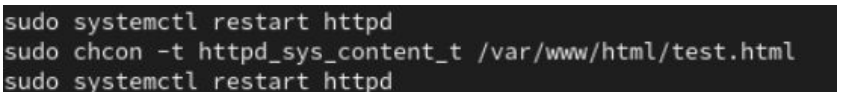

Рис. 20: Перезапуск сервера

Теперь он работает, ведь порт 81 внечен в список портов `htttpd_port_t` (рис. [-@fig:023]).

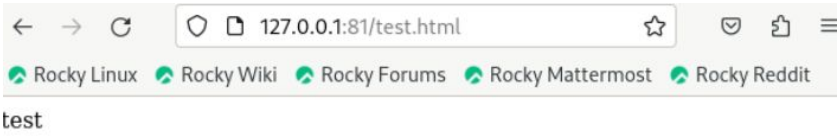

Рис. 21: Проверка сервера

Возвращаем в файле /etc/httpd/httpd.conf порт 80, вместо 81. Проверяем, что порт 81 удален (рис. [-@fig:024]).

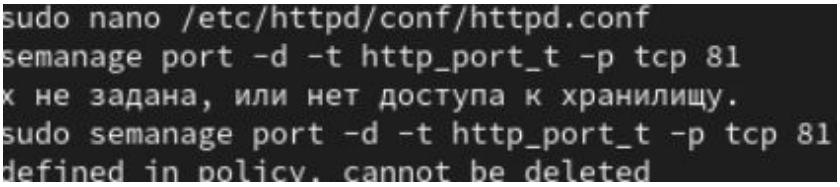

Рис. 22: Проверка порта 81

Далее чего удаляем файл test.html, проверяем, что он удален(рис. [-@fig:025]).

Рис. 23: Удаление файла

Выводы

В ходе выполнения данной лабораторной работы были развиты навыки администрирования ОС Linux, получено первое практическое знакомство с технологией SELinux и проверена работа SELinux на практике совместно с веб-сервером Apache.

Список литературы

[1] Документация по Virtual Box: https://www.virtualbox.org/wiki/Documentation

[2] Документация по Git: https://git-scm.com/book/ru/v2

[3] Документация по Markdown: https://learn.microsoft.com/ru-ru/contribute/markdown-reference