Лабораторная работа №6

Информационная безопасность

Николаев Дмитрий Иванович

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Развить навыки администрирования ОС Linux. Получить первое практическое знакомство с технологией SELinux.

Проверить работу SELinux на практике совместно с веб-сервером Apache.

2 Теоретическое введение

Домен — список действий, которые может выполнять процесс. Обычно в качестве домена определяется минимально возможный набор действий, при помощи которых процесс способен функционировать. Таким образом, если процесс дискредитирован, злоумышленнику не удастся нанести большого вреда.

Роль — список доменов, которые могут быть применены. Если какого-то домена нет в списке доменов какой-то роли, то действия из этого домена не могут быть применены.

Тип — набор действий, которые допустимы по отношения к объекту. Тип отличается от домена тем, что он может применяться к пайпам, каталогам и файлам, в то время как домен применяется к процессам.

Контекст безопасности — все атрибуты SELinux — роли, типы и домены [1].

3 Выполнение лабораторной работы

Установим веб-сервер Apache с помощью команды yum install httpd, после чего следуем согласно [2].

1. Войдем в систему и убедимся, что SELinux работает в режиме enforcing политики targeted с помощью команд getenforce и sestatus. ([3.1]).

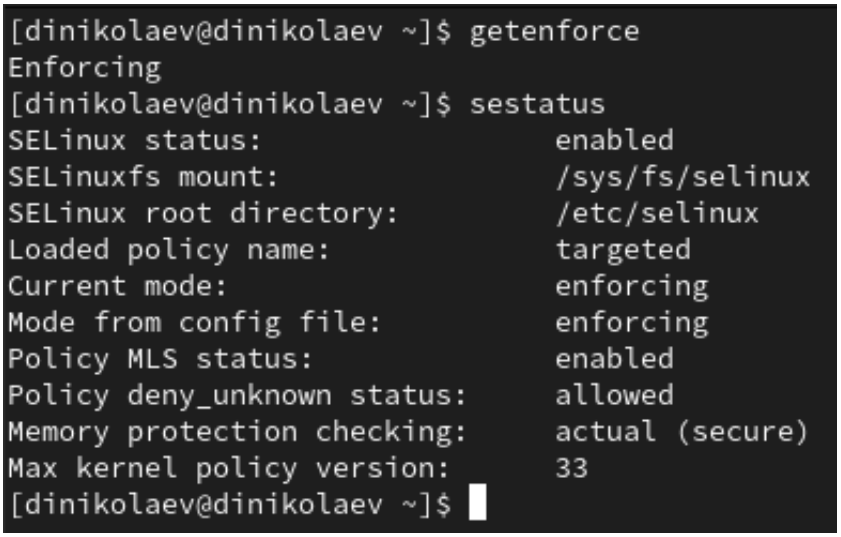

Рис. 3.1: Режим работы SELinux

2. Обратимся с помощью браузера к веб-серверу, запущенному на нашем компьютере, и убедимся, что последний работает: service httpd status ([3.2]).

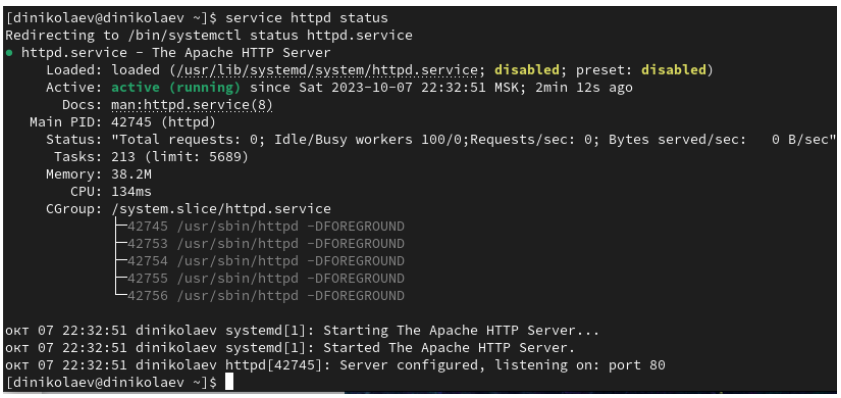

Рис. 3.2: Обращение к веб-серверу

3. Найдем веб-сервер Apache в списке процессов, определим его контекст безопасности командами ps auxZ | grep httpd и ps -eZ | grep httpd. Имеем следующий контекст: system_u:system_r:httpd_t (пользователь:роль:тип) ([3.3]).

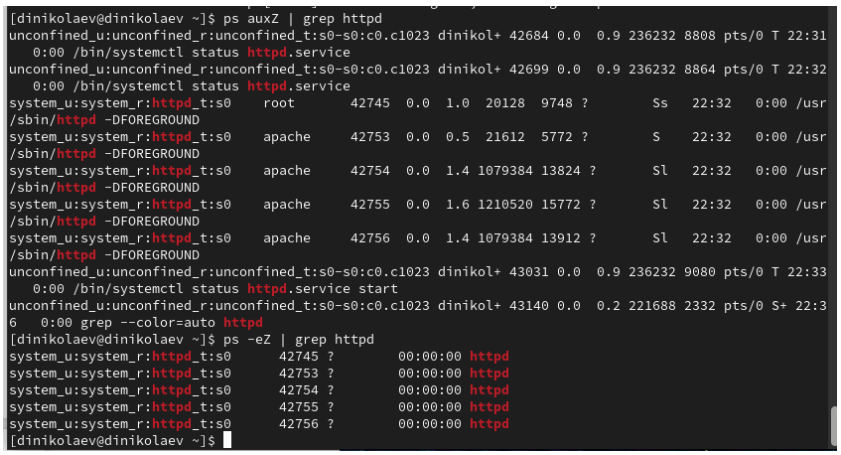

Рис. 3.3: Проверка контекста безопасности процессов

4. Посмотрим текущее состояние переключателей SELinux для Apache с помощью команды sestatus -b | grep httpd ([3.4]):

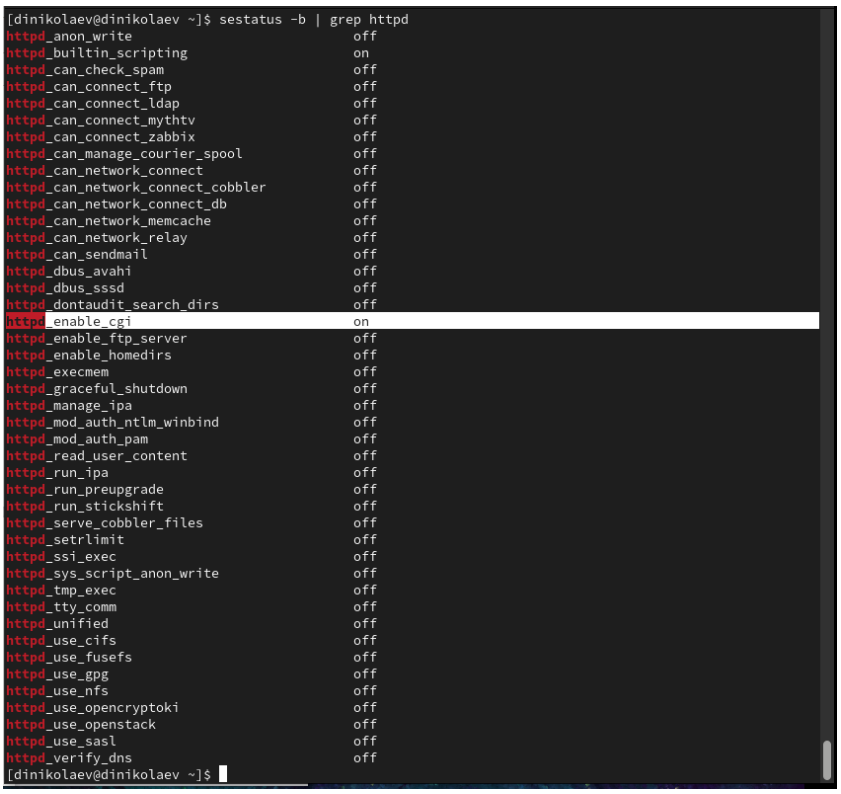

Рис. 3.4: Состояние переключателей SELinux

5. Посмотрим статистику по политике с помощью команды seinfo. Как видим, имеется 8 пользователей, 15 ролей и 5135 типов ([3.5]).

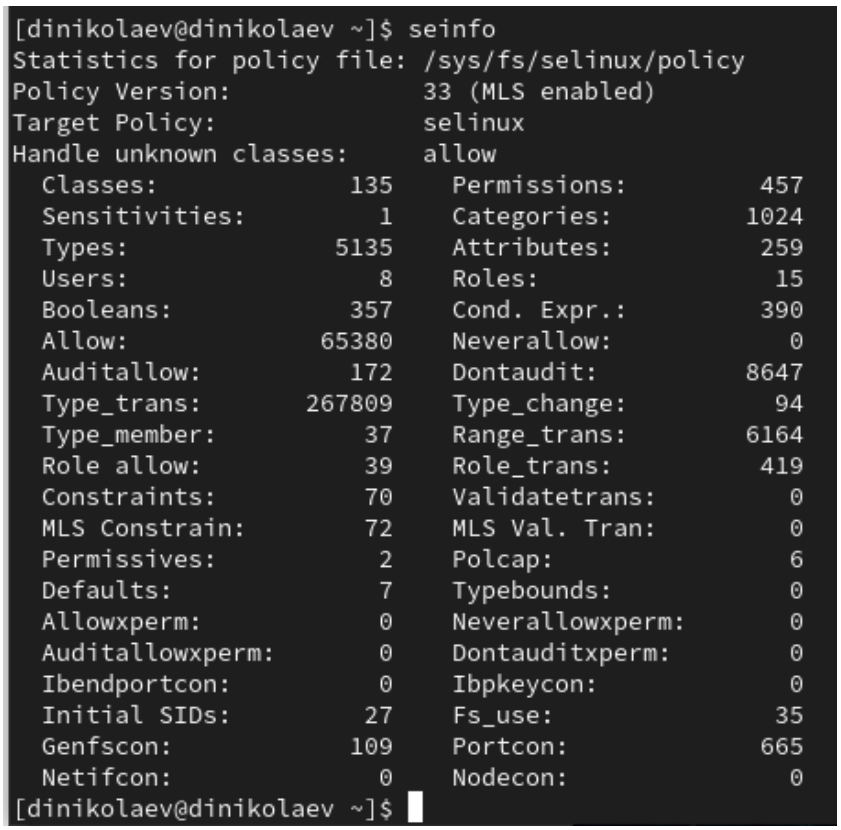

Рис. 3.5: Статистика по политике

6. Определим тип файлов и поддиректорий, находящихся в директории /var/www с помощью команды ls -lZ /var/www — в ней находится две поддиректории. Определим тип файлов, находящихся в директории /var/www/html командой ls -lZ /var/www/html — директория пуста. Определим круг пользователей, которым разрешено создание файлов в директории /var/www/html — разрешено только владельцу директории ([3.6]).

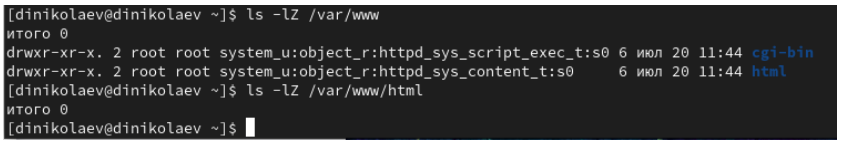

Рис. 3.6: Информация о файлах и поддиректориях директории /var/www

7. Создадим от имени суперпользователя (так как в дистрибутиве после установки только ему разрешена запись в директорию) html-файл /var/www/html/test.html следующего содержания ([3.7]):

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

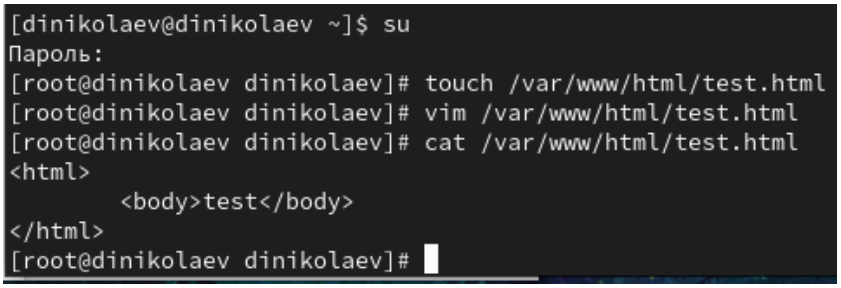

Рис. 3.7: Создание html-файла

8. Проверим контекст созданного нами файла. По умолчанию устанавливается следующий контекст: пользователь — unconfined_u (несвязанный), роль — object_r, тип — httpd_sys_content_t ([3.8]).

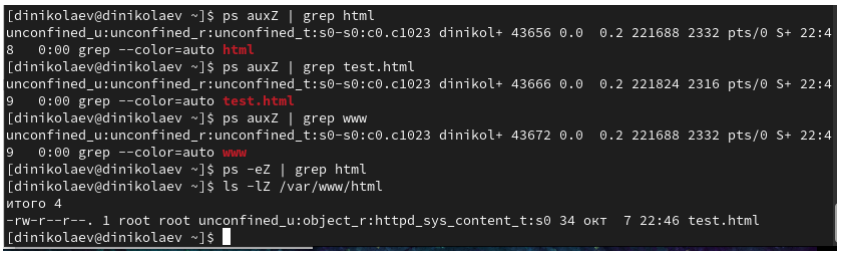

Рис. 3.8: Проверка контекста созданного файла

9. Обратимся к файлу через веб-сервер, введя в браузере адрес http://127.0.0.1/test.html. Убедимся, что файл был успешно отображён ([3.9]).

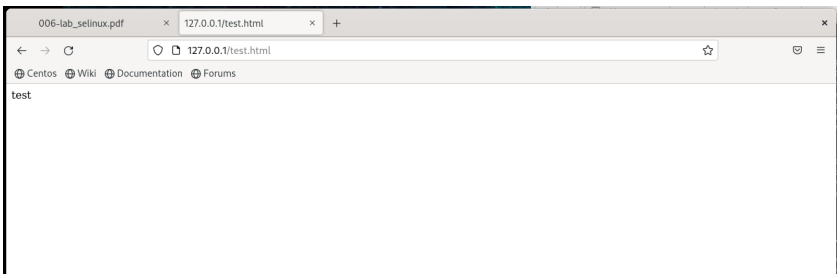

Рис. 3.9: Обращение к файлу через веб-сервер

10. Изучим справку man httpd_selinux. Справки по httpd_selinux нет, а справки по httpd и selinux не содержат информации о возможных контекстах. Проверим контекст нашего файла командой ls -Z /var/www/html/test.html ([3.10]).

```
[dinikolaev@dinikolaev ~]$ man selinux
[dinikolaev@dinikolaev ~]$ man httpd
[dinikolaev@dinikolaev ~]$ man httpd_selinux
Нет справочной страницы для httpd_selinux
[dinikolaev@dinikolaev ~]$ ls -Z /var/www/html/test.html
unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/test.html
```

Рис. 3.10: Вызов справки и проверка контекста созданного файла

11. Изменим 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 ([3.11]):

    - chcon -t samba_share_t /var/www/html/test.html
    - ls -Z /var/www/html/test.html

```
[dinikolaev@dinikolaev ~]$ su
Пароль:
[root@dinikolaev dinikolaev]# chcon -t samba_share_t /var/www/html/test.html
[root@dinikolaev dinikolaev]# ls -Z /var/www/html/test.html
unconfined_u:object_r:samba_share_t:s0 /var/www/html/test.html
[root@dinikolaev dinikolaev]#
```

Рис. 3.11: Смена контекста файла

12. Попробуем ещё раз получить доступ к файлу через веб-сервер, введя в браузере адрес http://127.0.0.1/test.html. Получили сообщение об ошибке ([3.12]):

    Forbidden You don’t have permission to access /test.html on this server

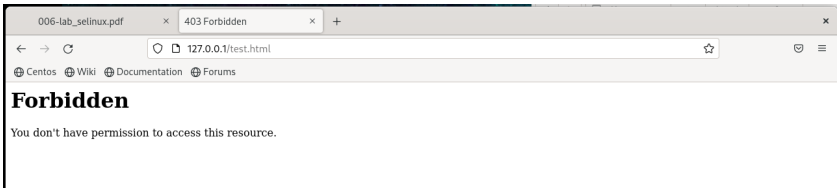

Рис. 3.12: Попытка получения доступа к файлу через веб-сервер

13. Проанализируем ситуацию командой ls -l /var/www/html/test.html. Просмотрим log-файлы веб-сервера Apache. Также просмотрим системный лог-файл: tail /var/log/messages ([3.13]).

```
[dinikolaev@dinikolaev ~]$ ls -Z /var/www/html/test.html
unconfined_u:object_r:samba_share_t:s0 /var/www/html/test.html
[dinikolaev@dinikolaev ~]$ ls -l /var/www/html/test.html
-rw-r--r--. 1 root root 34 окт  7 22:46 /var/www/html/test.html
[dinikolaev@dinikolaev ~]$ tail /var/log/messages
tail: невозможно открыть '/var/log/messages' для чтения: Отказано в доступе
[dinikolaev@dinikolaev ~]$ sudo tail /var/log/messages
[sudo] пароль для dinikolaev:
Oct  7 23:11:58 dinikolaev packagekitd[2066]: Failed to get cache filename for webkit2gtk3-jsc
Oct  7 23:11:58 dinikolaev packagekitd[2066]: Failed to get cache filename for kernel
Oct  7 23:11:58 dinikolaev packagekitd[2066]: Failed to get cache filename for kernel-core
Oct  7 23:11:58 dinikolaev packagekitd[2066]: Failed to get cache filename for kernel-modules
Oct  7 23:11:58 dinikolaev packagekitd[2066]: Failed to get cache filename for kernel-modules-core
Oct  7 23:11:58 dinikolaev packagekitd[2066]: Failed to get cache filename for kernel-devel
Oct  7 23:12:07 dinikolaev systemd[1]: dbus-:1.1-org.fedoraproject.SetroubleshootPrivileged@0.service:
Deactivated successfully.
Oct  7 23:12:07 dinikolaev systemd[1]: setroubleshootd.service: Deactivated successfully.
Oct  7 23:14:32 dinikolaev systemd[1]: Starting Fingerprint Authentication Daemon...
Oct  7 23:14:32 dinikolaev systemd[1]: Started Fingerprint Authentication Daemon.
```

Рис. 3.13: Последние сообщения log-файла

Не смогли получить доступ к кэшу и сообщение о деактивации от setroubleshootd.service.

14. Попробуем запустить веб-сервер Apache на прослушивание TCP-порта 81 (а не 80, как рекомендует IANA и прописано в /etc/services). Для этого в файле /etc/httpd/conf/httpd.conf найдем строчку Listen 80 и заменим её на Listen 81 ([3.14]).

Рис. 3.14: Смена порта прослушивания с 80 на 81

15. Выполним перезапуск веб-сервера Apache ([3.15]).

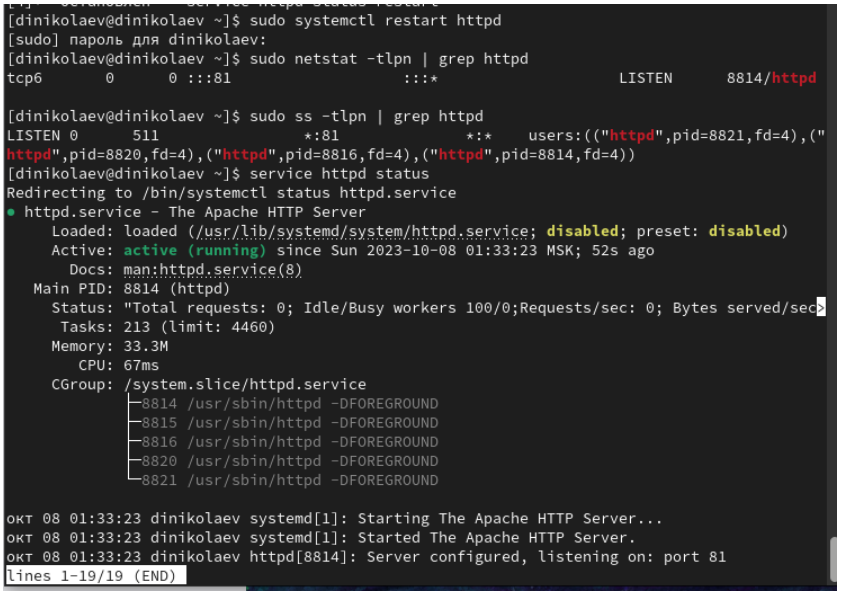

Рис. 3.15: Перезапуск веб-сервера Apache

Снова попытавшись получить доступ через веб-сервер, введя адрес в браузере увидим следующую картину ([3.16]).

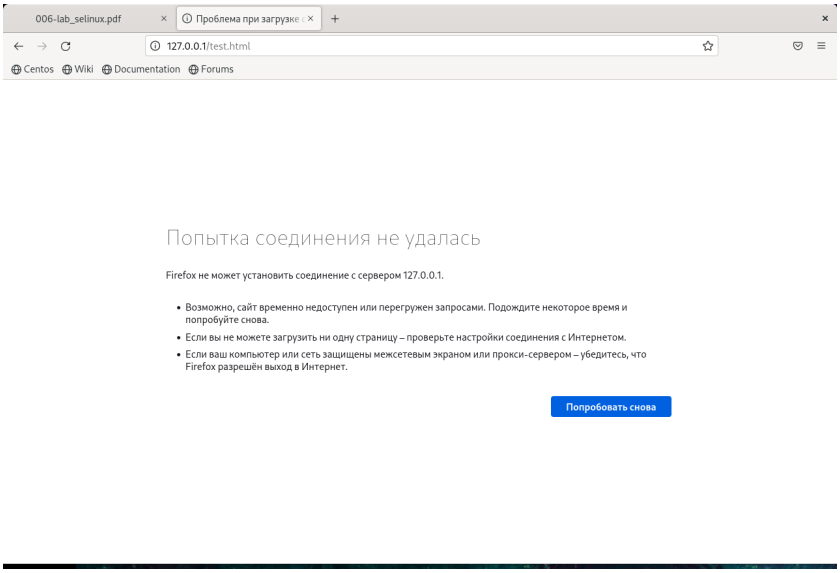

Рис. 3.16: Попытка подключения к веб-серверу через браузер

16. Проанализируем лог-файлы: tail -n10 /var/log/messages. Также просмотрим файлы /var/log/httpd/error_log, /var/log/httpd/access_log и /var/log/audit/audit.log/ Новые записи появились в файлах /var/log/messages и /var/log/httpd/error_log. ([3.17]).

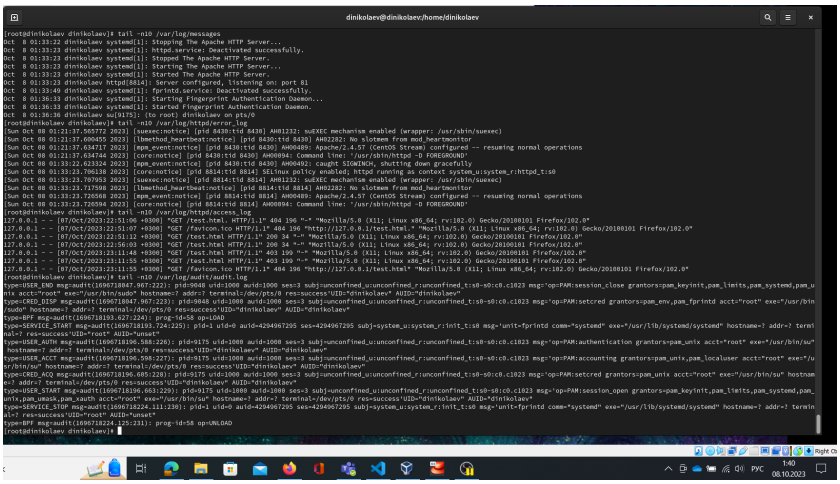

Рис. 3.17: Анализ сообщений лог-файлов

17. Выполним команду semanage port -a -t http_port_t -p tcp 81. После прове-

рим список портов командой semanage port -l | grep http_port_t. Убедимся, что порт 81 появился в списке ([3.18]).

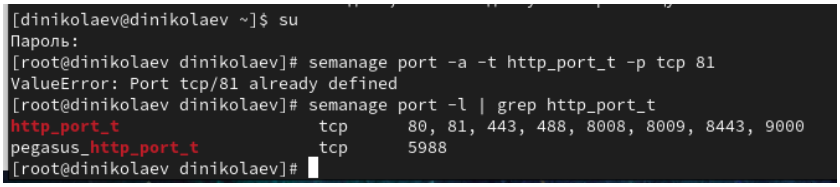

Рис. 3.18: Установка 81 порта и проверка списка всех портов

18. Попробуем запустить веб-сервер Apache ещё раз ([3.19]).

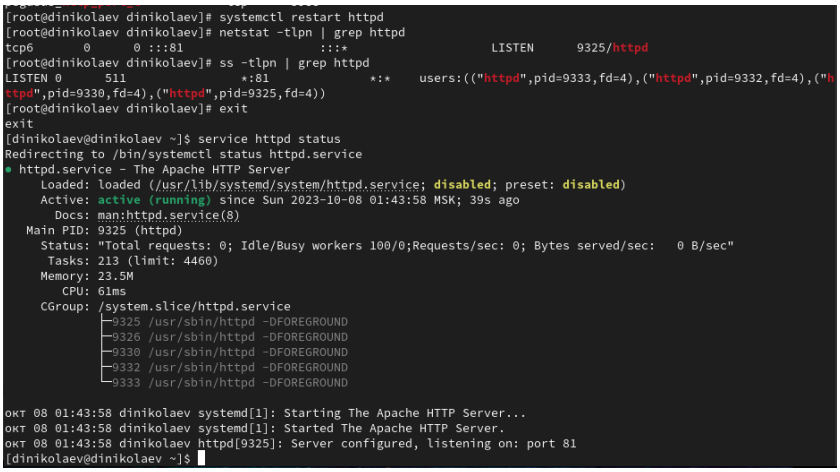

Рис. 3.19: Повторный запуск веб-сервера Apache

19. Вернем контекст httpd_sys_content__t к файлу /var/www/html/test.html: chcon -t httpd_sys_content_t /var/www/html/test.html ([3.20]). После этого попробуем получить доступ к файлу через веб-сервер, введя в браузере адрес http://127.0.0.1:81/test.html. Увидим содержимое файла — слово «test» ([3.21]).

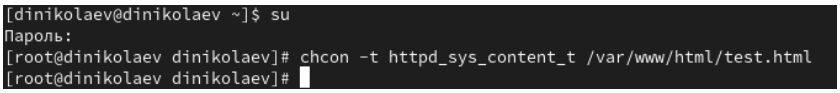

Рис. 3.20: Возвращение контекста httpd_sys_content_t нашему файлу

Рис. 3.21: Получение доступа к файлу через браузер

20. Исправим обратно конфигурационный файл apache, вернув Listen 80 ([3.22]).

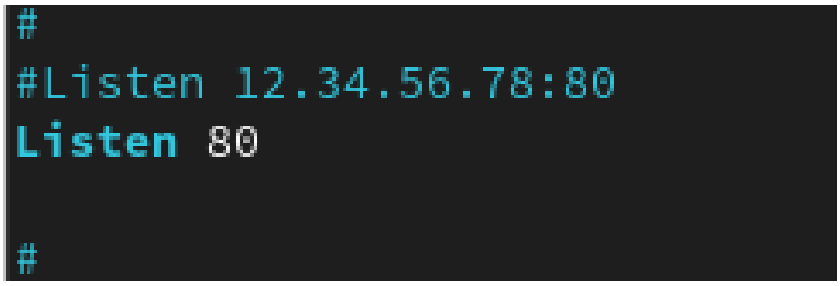

Рис. 3.22: Смена порта прослушивания с 81 на 80

21. Удалим привязку http_port_t к 81 порту: semanage port -d -t http_port_t -p tcp 81. Порт 81 не был удален, так как в начале 80 и 81 порты были настроены по умолчанию ([3.23]).

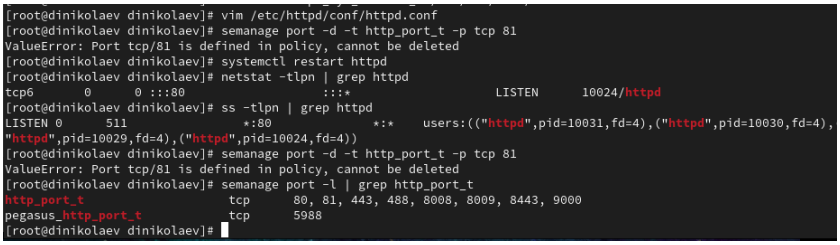

Рис. 3.23: Попытка удаления привязки к 81 порту

22. Удалим файл /var/www/html/test.html: rm /var/www/html/test.html ([3.24]).

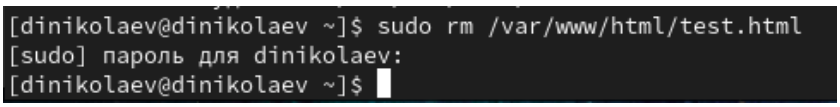

Рис. 3.24: Удаление файла test.html

4 Выводы

В ходе выполнения лабораторной работы я развил навыки администрирования ОС Linux, получил первое практическое знакомство с технологией SELinux и проверил работу SELinux на практике совместно с веб-сервером Apache.

Список литературы

1. SELinux [Электронный ресурс]. URL: https://habr.com/ru/companies/kingservers/articles/209644/.
2. Кулябов Д. С., Королькова А. В., Геворкян М. Н Лабораторная работа №6 [Электронный ресурс]. RUDN, 2023. URL: https://esystem.rudn.ru/pluginfile.php/2090210/mod_resource/content/2/006-lab_selinux.pdf.