Отчёт по лабораторной работе 6

Мандатное разграничение прав в Linux

Румянцева Александра Сергеевна

Содержание

[Цель работы 1](#_Toc88869440)

[Задание 1](#_Toc88869441)

[Теория 1](#_Toc88869442)

[SELinux 1](#_Toc88869443)

[Режимы работы SELinux 2](#_Toc88869444)

[Контроль доступа в SELinux 2](#_Toc88869445)

[Выполнение лабораторной работы 3](#_Toc88869446)

[Подготовка к выполнению 3](#_Toc88869447)

[Выполнение основной частии лабораторной работы 5](#_Toc88869448)

[Библиография 15](#_Toc88869449)

[Выводы 15](#_Toc88869450)

# Цель работы

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

# Задание

Лабораторная работа подразумевает изучение технологий SELinux и веб-сервера Apache опытным путём.

# Теория

## SELinux

SELinux (SELinux) — это система принудительного контроля доступа, реализованная на уровне ядра. Впервые эта система появилась в четвертой версии CentOS, а в 5 и 6 версии реализация была существенно дополнена и улучшена. Эти улучшения позволили SELinux стать универсальной системой, способной эффективно решать массу актуальных задач. Стоит помнить, что классическая система прав Unix применяется первой, и управление перейдет к SELinux только в том случае, если эта первичная проверка будет успешно пройдена.

## Режимы работы SELinux

SELinux имеет три основных режим работы, при этом по умолчанию установлен режим Enforcing. Это довольно жесткий режим, и в случае необходимости он может быть изменен на более удобный для конечного пользователя.

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.

## Контроль доступа в SELinux

SELinux предоставляет следующие модели управления доступом:

Type Enforcement (TE): основной механизм контроля доступа, используемый в целевых политиках. Позволяет детально, на самом низком уровне управлять разрешениями. Самый гибкий, но и самый трудоемкий для системного администратора механизм.

Role-Based Access Control (RBAC): в этой модели права доступа реализуются в качестве ролей. Ролью называется разрешения на выполнение определенных действий одним или несколькими элементами системы над другими частями системы. По-сути, RBAC является дальнейшим развитием TE.

Multi-Level Security (MLS): многоуровневая модель безопасности, в которой всем объектам системы присваивается определенный уровень доступа. Разрешение или запрет доступа определяется только соотношением этих уровней.

Все процессы и файлы в рамках SELinux имеют контекст безопасности. Давайте посмотрим на контекст на практике, подробно рассмотрев стартовую страницу веб-сервера Apache, находящуюся по адресу /var/www/html/index.html

$ ls -Z /var/www/html/index.html

-rw-r–r– username username system\_u:object\_r:httpd\_sys\_content\_t /var/www/html/index.html

В дополнение к стандартным правам доступа к файлу, мы можем видеть контекст безопасности SELinux: system\_u: object\_r: httpd\_sys\_content\_t.

Контекст базируется на user:role:type:mls, но поля user:role:type отображаются, в то время как поле mls скрыто. Также мы можем видеть целевую политику, в данном случае httpd\_sys\_content\_t.

Теперь рассмотрим контекст безопасности SELinux для процесса ‘httpd’ (веб-сервер Apache):

$ ps axZ | grep httpd

system\_u:system\_r:httpd\_t 3234 ? Ss 0:00 /usr/sbin/httpd

Как мы видим, этот процесс запущен на домене httpd\_t.

Ну а теперь давайте посмотрим на контекст безопасности файла в нашем домашнем каталоге:

$ ls -Z /home/username/myfile.txt

-rw-r–r– username username user\_u:object\_r:user\_home\_t /home/username/myfile.txt

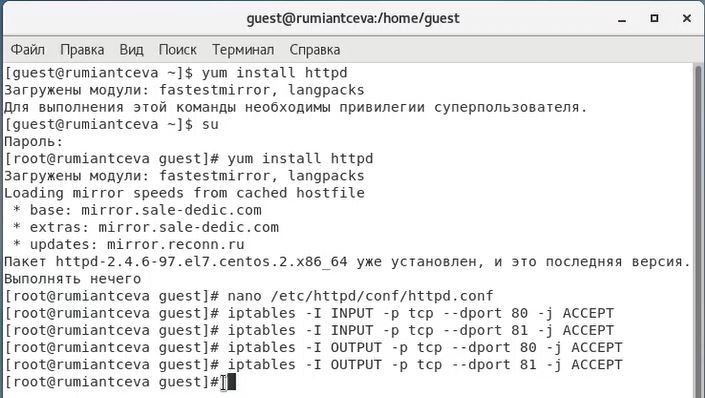
Мы видим, что файл имеет тип user\_home\_t, этот тип присваивается по умолчанию всем файлам в домашнем каталоге.

Доступ разрешен только между элементами с одинаковым типом, именно поэтому веб-сервер Apache может без проблем читать файл /var/www/html/index.html, который имеет тип httpd\_sys\_content\_t. В то же самое время, так как Apache запущен на домене httpd\_t и не имеет заполненных полей userid:username, он не может получить доступ к файлу home/username/myfile.txt, хотя этот файл доступен для чтения процессам, для которых не определена целевая политика. Таким образом, если веб-сервер Apache будет взломан, то злоумышленник не сможет получить доступ к файлам или запускать процессы, которые не находятся в домене httpd\_t.

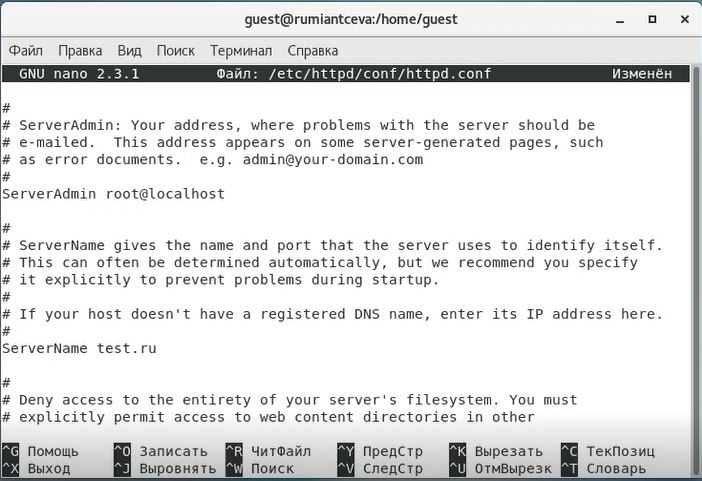
# Выполнение лабораторной работы

## Подготовка к выполнению

1. Установила от имени суперпользователя веб-сервер Apache с помощью команды yum install httpd. В моём случаи оказалось, что он уже установлен (рис. 1)

* 
* Figure 1: рис.1. Установка веб-сервер Apache. Добавле разрешающих правил.

1. В конфигурационном файле /etc/httpd/httpd.conf задала параметр ServerName: ServerName test.ru чтобы при запуске веб-сервера не выдавались лишние сообщения об ошибках, не относящихся к лабораторной работе (рис. 2).

* 
* Figure 2: рис.2. Файл /etc/httpd/httpd.conf

1. Отключила пакетный фильтр, точнее сделала так, что он в своей рабочей конфигурации позволял подключаться к 80-му и 81-му портам протокола tcp, добавив разрешающие правила с помощью команд (рис. 1):

iptables -I INPUT -p tcp –dport 80 -j ACCEPT

iptables -I INPUT -p tcp –dport 81 -j ACCEPT

iptables -I OUTPUT -p tcp –sport 80 -j ACCEPT

iptables -I OUTPUT -p tcp –sport 81 -j ACCEPT

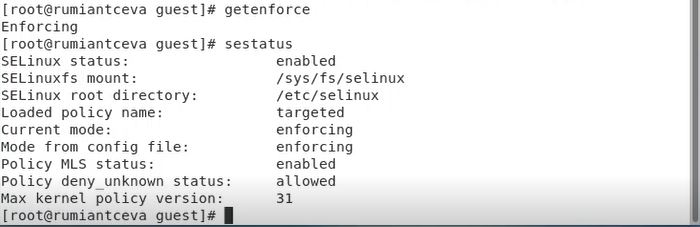
Можно было бы также отключить фильтр командами:

iptables -F

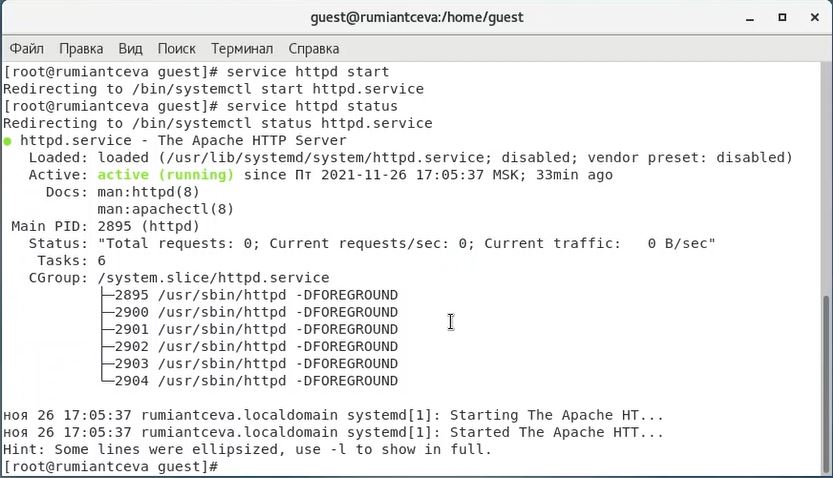
iptables -P INPUT ACCEPT iptables -P OUTPUT ACCEPT

## Выполнение основной частии лабораторной работы

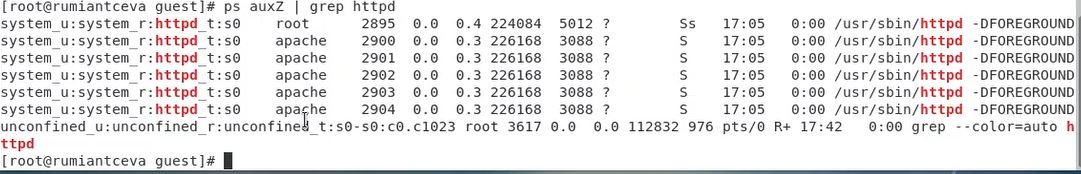
1. Вошла в систему с полученными учётными данными и убедилась, что SELinux работает в режиме enforcing политики targeted с помощью команд getenforce и sestatus (рис.3).

* 
* Figure 3: рис.3. Команды getenforce и sestatus.

1. Обратилась к веб-серверу, запущенному на нашем компьютере, и убедимся, что последний работает: service httpd status (рис. 4).

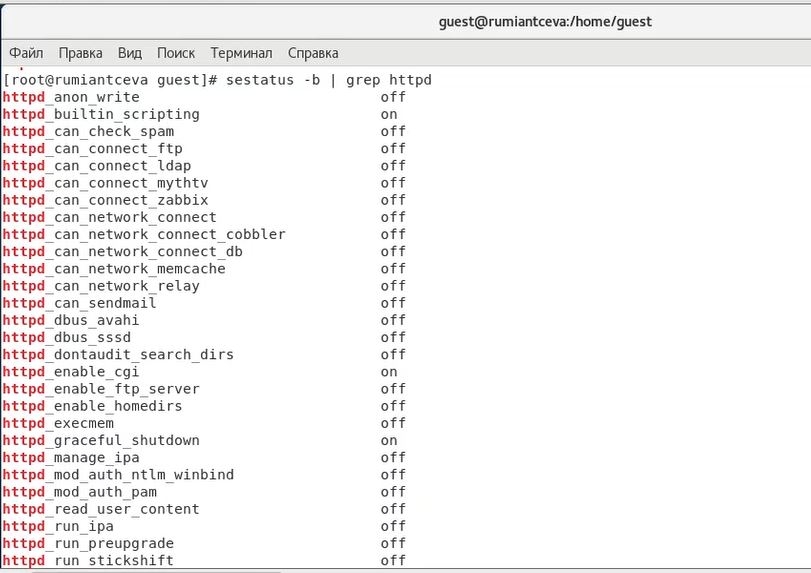
* 
* Figure 4: рис.4. service httpd status

1. Нашла веб-сервер Apache в списке процессов, определила его контекст безопасности, используя команду ps auxZ | grep httpd (рис. 5).

* 
* Figure 5: рис.5. Команда ps auxZ | grep httpd.

В моём случае процесс запущен на домене httpd\_t.

1. Посмотрела текущее состояние переключателей SELinux для Apache с помощью команды sestatus –b | grep httpd (рис. 6).

* 
* Figure 6: рис.6. Команда sestatus –b | grep httpd.

Наглядно видно, что многие из переключателей находятся в положении «off».

1. Посмотрела статистику по политике с помощью команды seinfo, также определила множество пользователей, ролей и типов. (рис. 7)

Замечу, что для выполнения команды пришлось выполнить установку setools-console

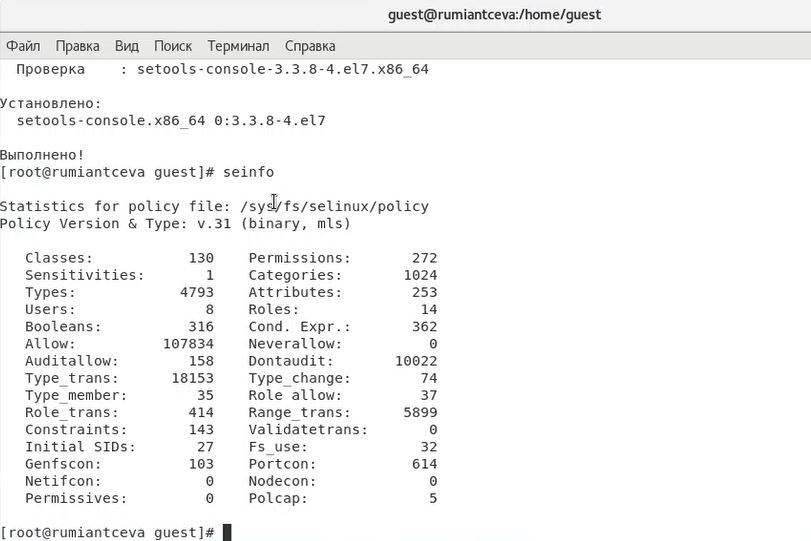
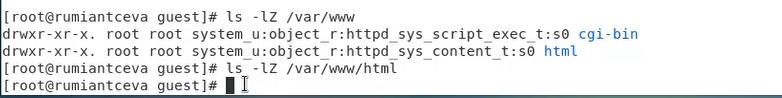


Figure 7: рис.7. Команда seinfo.

Из рисунка наглядно видно, что пользователей: 8, ролей: 14, типов: 4793.

1. Определила тип файлов и поддиректорий, находящихся в директории /var/www с помощью команды ls -lZ /var/www (рис. 8).
2. Определила тип файлов, находящихся в директории /var/www/html с помощью команды ls –lZ /var/www/html (рис. 8). Директория оказалась пустой

* 
* Figure 8: рис.8. Программа readfile.c

1. Определила круг пользователей, которым разрешено создание файлов в директории /var/www/html (рис. 9).

Я выполнила команду touch, команду создания файла, для каждого пользователя. Таким образом я опытным путёи определила, что только суперпользователь может создать файл в данной директории.

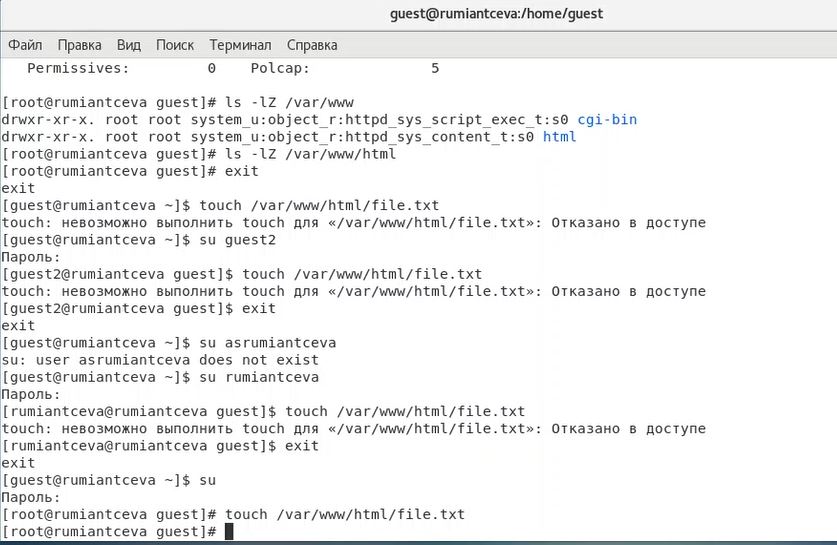


Figure 9: рис.9. Определение пользователей, которым разрешено создание файлов в директории /var/www/html.

1. Создала от имени суперпользователя html-файл /var/www/html/test.html следующего содержания (рис. 10):

test



Figure 10: рис.10. html-файл /var/www/html/test.html.

1. Проверила контекст созданного файла (рис. 11).

Контекст, присваиваемый по умолчанию вновь созданным файлам в директории /var/www/html: unconfined\_u:object\_r:httpd\_sys\_content\_t

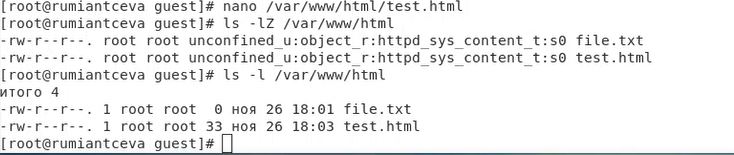


Figure 11: рис.11. Контекст созданного файла.

1. Обратилась к файлу через веб-сервер, введя в браузере firefox адрес: http://127.0.0.1/test.html. Убедилась, что файл был успешно отображен (рис. 12).

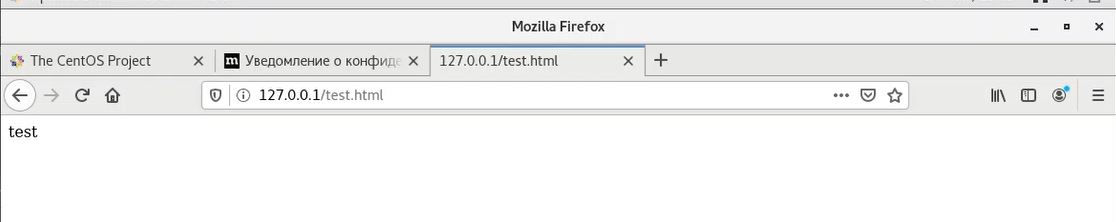


Figure 12: рис.12. http://127.0.0.1/test.html.

1. Изучила справку man httpd\_selinux и выяснила, какие контексты файлов определены для httpd и сопоставим их с типом файла test.html. Проверила контекст файла командой ls –Z /var/www/html/test.html (рис. 13).

Т.к. по умолчанию пользователи CentOS являются свободными (unconfined) от типа, 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.

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.

1. Изменила контекст файла /var/www/html/test.html с httpd\_sys\_content\_t на другой, к которому процесс httpd не должен иметь доступа, в нашем случае, на samba\_share\_t и проверила изменения (рис. 13).



Figure 13: рис.13. Работа с контекстом файла /var/www/html/test.html.

1. Попробовала еще раз получить доступ к файлу через веб-сервер, введя в браузере firefox адрес http://127.0.0.1/test.html. Но получила сообщение об ошибке (рис. 14).

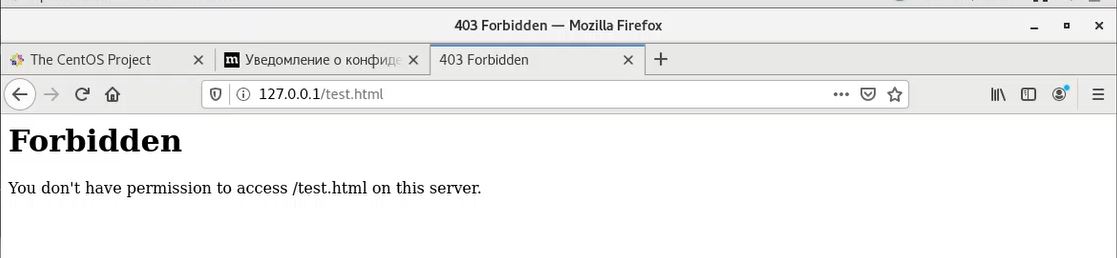


Figure 14: рис.14. http://127.0.0.1/test.html при контексте samba\_share\_t.

1. Проанализировала ситуацию, просмотрев log-файлы веб-сервера Apache, системный log-файл и audit.log при условии уже запущенных процессов setroubleshootd и audtd (рис. 15).

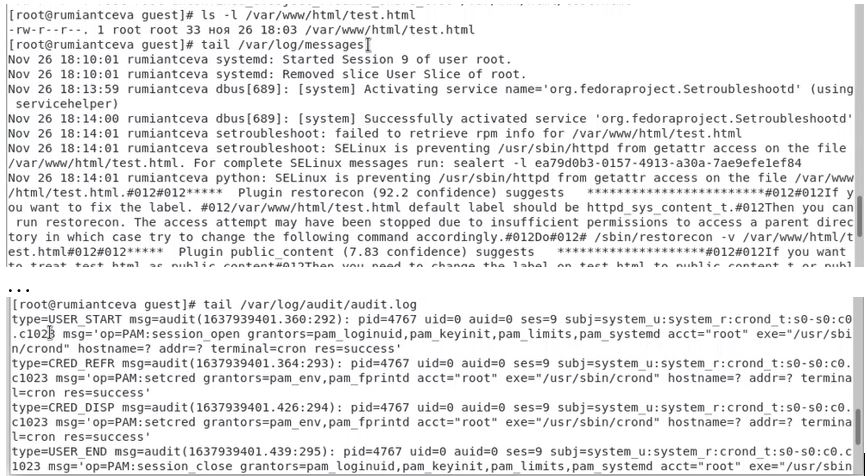


Figure 15: рис.15. Просмотр log-файлы веб-сервера Apache.

Исходя из log-файлов, мы можем заметить, что проблема в измененном контексте на шаге 13, т.к. процесс httpd не имеет доступа на samba\_share\_t. В системе оказались запущены процессы setroubleshootd и audtd, поэтому ошибки, связанные с измененным контекстом, также есть в файле /var/log/audit/audit.log.

1. Попробовала запустить веб-сервер Apache на прослушивание TCP-порта 81 (а не 80, как рекомендует IANA и прописано в /etc/services), заменив в файле /etc/httpd/conf/httpd.conf строчку Listen 80 на Listen 81 (рис. 16).

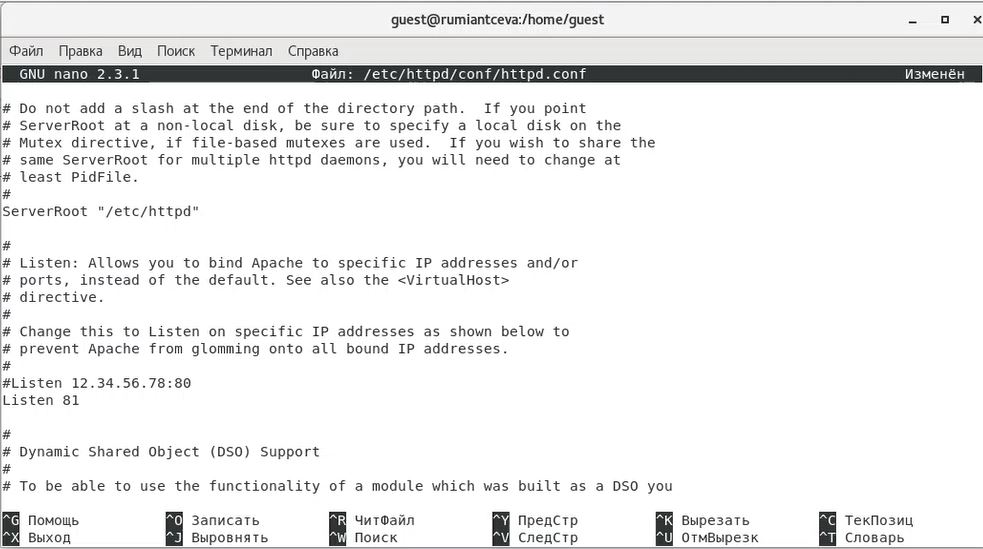


Figure 16: рис.16. Просмотр log-файлы веб-сервера Apache.

1. Перезапустила веб-сервер Apache и попробовала обратиться к файлу через веб-сервер, введя в браузере firefox адрес http://127.0.0.1/test.html (рис. 17, 18).

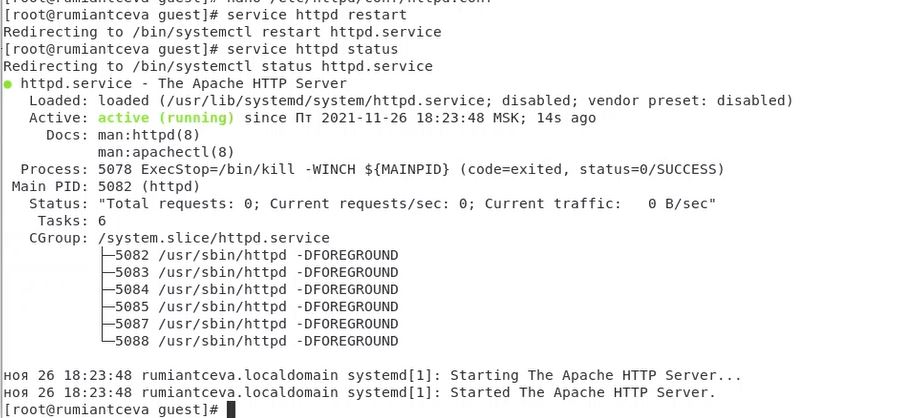


Figure 17: рис.17. Перезапуск веб-сервера Apache.

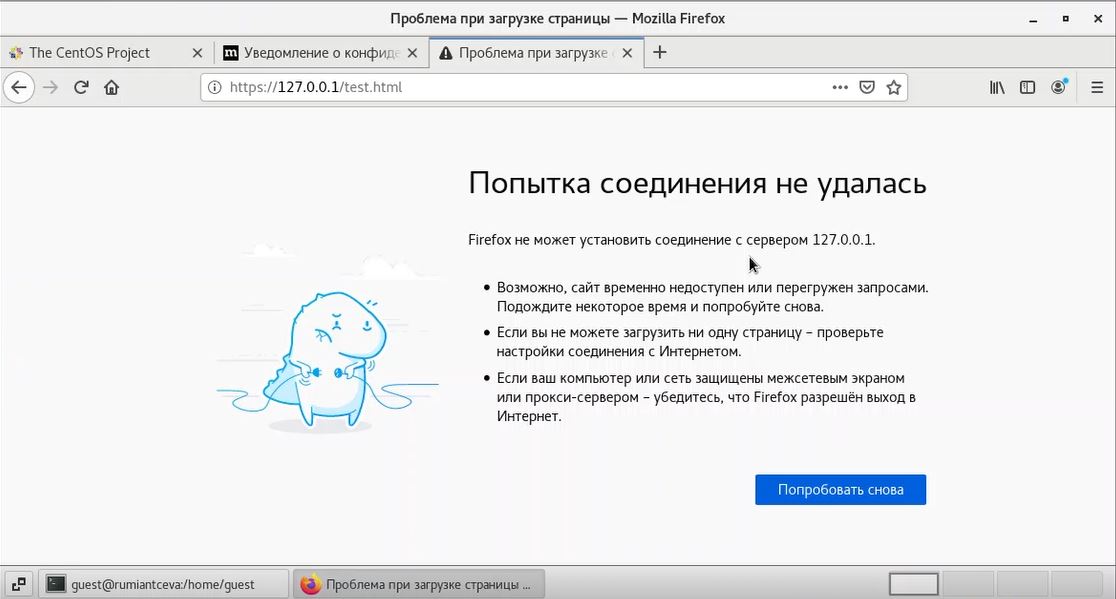


Figure 18: рис.18. http://127.0.0.1/test.html при Listen 81.

Из того, что при запуске файла через браузер появилась ошибка, можно сделать предположение, что в списках портов, работающих с веб-сервером Apache, отсутствует порт 81.

1. Проанализировала log-файлы: tail –n1 /var/log/messages и просмотрела файлы /var/log/http/error\_log, /var/log/http/access\_log и /var/log/audit/audit.log. Во всех log-файлах появились записи, кроме /var/log/messages.
2. Выполнила команду semanage port –a –t http\_port\_t –p tcp 81 и после этого проверила список портов командой semanage port –l | grep http\_port\_t (рис. 19).

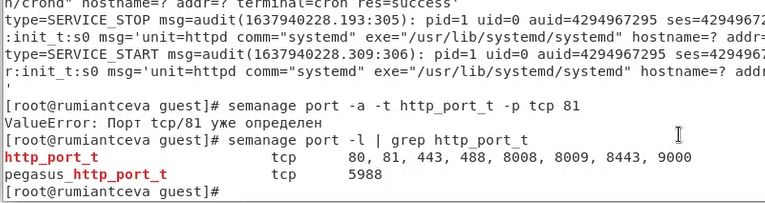


Figure 19: рис.19. Порт 81.

Заметим, что порт 81 уже определён, и он лействительно есть в списке портов, так как данный порт определён на уровне политики..

1. Попробовала запустить веб-сервер Apache еще раз. Он успешно запустился.
2. Вернула контекст httpd\_sys\_content\_t к файлу /var/www/html/test.html: chcon –t httpd\_sys\_content\_t /var/www/html/test.html (рис. 20).

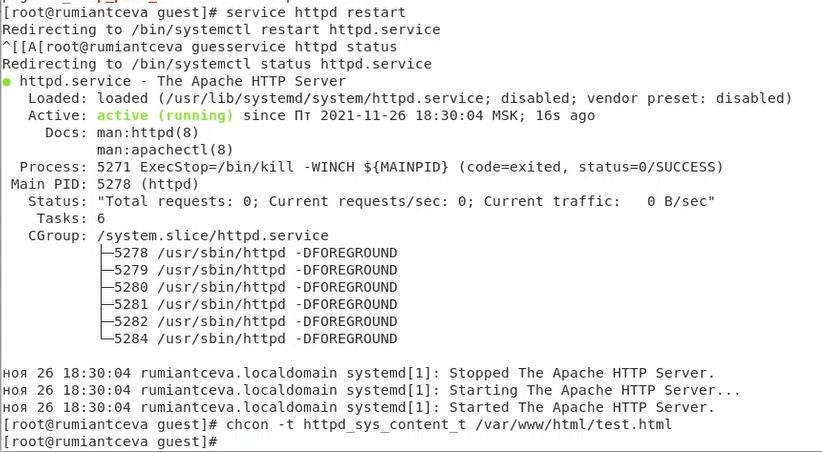


Figure 20: рис.20. Restart и возврат контекста.

После вновь попробовала получить доступ к файлу через веб-сервер, введя в браузере firefox адрес http://127.0.0.1:81/test.html (рис. 21). Увидели слово содержимое файла - слово «test».

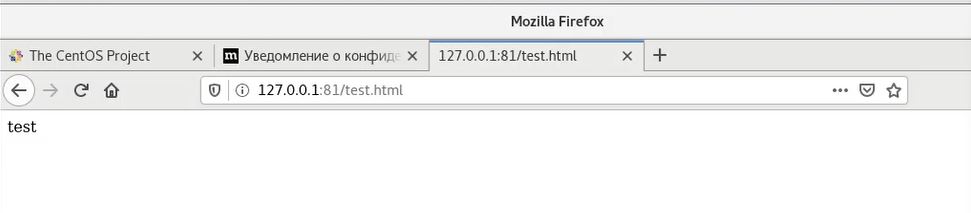


Figure 21: рис.21. http://127.0.0.1:81/test.html.

1. Исправила обратно конфигурационный файл apache, вернув Listen 80.
2. Попробовала удалить привязку http\_port\_t к 81 порту: semanage port –d –t http\_port\_t –p tcp 81. Данную команду выполнить невозможно на моей версии CentOS, так как порт 81 определён на уровне политики (рис. 22).

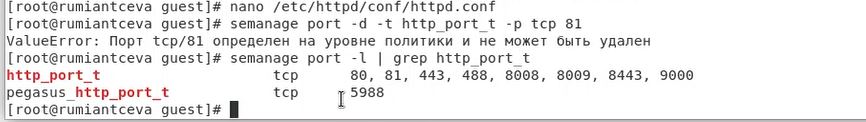


Figure 22: рис.22. Удаление привязки http\_port\_t к 81 порту.

1. Удалила файл /var/www/html/test.html: rm /var/www/html/test.html (рис. 23)

Figure 23: рис.23. Удаление файла /var/www/html/test.html.

Figure 23: рис.23. Удаление файла /var/www/html/test.html.

# Библиография

1. ТУИС РУДН
2. Статья “SELinux – описание и особенности работы с системой.” на сайте harb.com <https://habr.com/ru/company/kingservers/blog/209644/>

# Выводы

Я развила навыки администрирования ОС Linux. Получила первое практическое знакомство с технологией SELinux. Проверила работу SELinux на практике совместно с веб-сервером Apache.