Домашнее задание №3

Contents

[Первая часть 2](#_Toc210320463)

[Установить Apache и убедиться, что он работает через команду 2](#_Toc210320464)

[Просмотреть страницу по URL-адресу для проверки рабочего состояния 2](#_Toc210320465)

[Для обеспечения безопасности данных настроить HTTPS-подключение с использованием самоподписанных SSL-сертификатов в Apache 3](#_Toc210320466)

[Ставим mod\_ssl и openssl 3](#_Toc210320467)

[Генерим сертификат 4](#_Toc210320468)

[Настраиваем права 4](#_Toc210320469)

[Создаем отдельный файл конфигурации 4](#_Toc210320470)

[Настраиваем файерволл 5](#_Toc210320471)

[Финальные проверки 5](#_Toc210320472)

[Часть 2 8](#_Toc210320473)

[Установить Nginx и убедиться, что он работает: 8](#_Toc210320474)

[Просмотреть страницу по URL-адресу для проверки рабочего состояния 9](#_Toc210320475)

[Настройте прямой и обратный прокси в Nginx для перенаправления запросов 9](#_Toc210320476)

[Обратный (reverse) прокси 9](#_Toc210320477)

[Прямой (forward) прокси 10](#_Toc210320478)

[Для обеспечения безопасности данных настроить HTTPS-подключение с использованием самоподписанных SSL-сертификатов в Nginx 11](#_Toc210320479)

[Установить и настроить ModSecurity, в том числе настроить фильтрацию запросов для обеспечения безопасности от SQL-запросов 17](#_Toc210320480)

[Установка ModSecurity 17](#_Toc210320481)

[Настройка ModSecurity 17](#_Toc210320482)

[Создание файла main.conf 18](#_Toc210320483)

[Продемонстрировать успешную блокировку SQL-инъекций 19](#_Toc210320484)

[Демонстрация работы Modsecurity (в примере показана XSS, вам необходимо показать на SQLi) 19](#_Toc210320485)

[Закомментировать правило в файле конфигурации Modsecurity и показать, что Nginx возвращает код 200 20](#_Toc210320486)

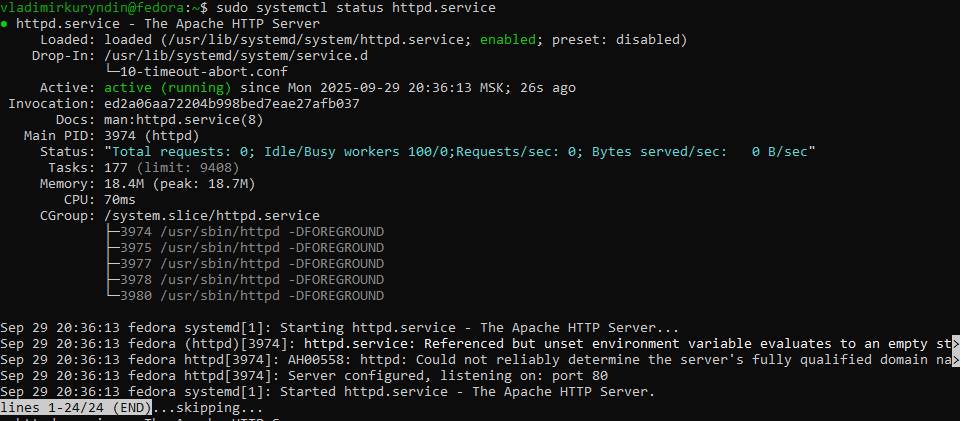
[Дополнительная проверка различных паттернов 20](#_Toc210320487)

# Первая часть

## Установить Apache и убедиться, что он работает через команду

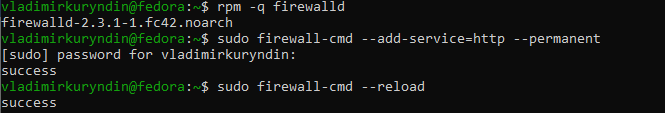
sudo systemctl status apache2.service

У меня Fedora, там вместо apache есть сервис httpd.

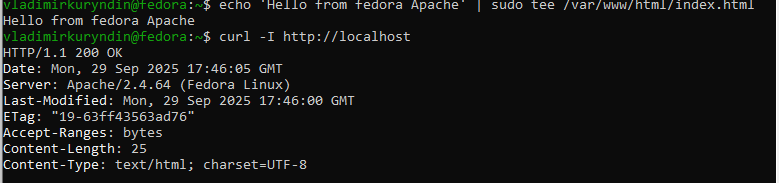


## Просмотреть страницу по URL-адресу для проверки рабочего состояния

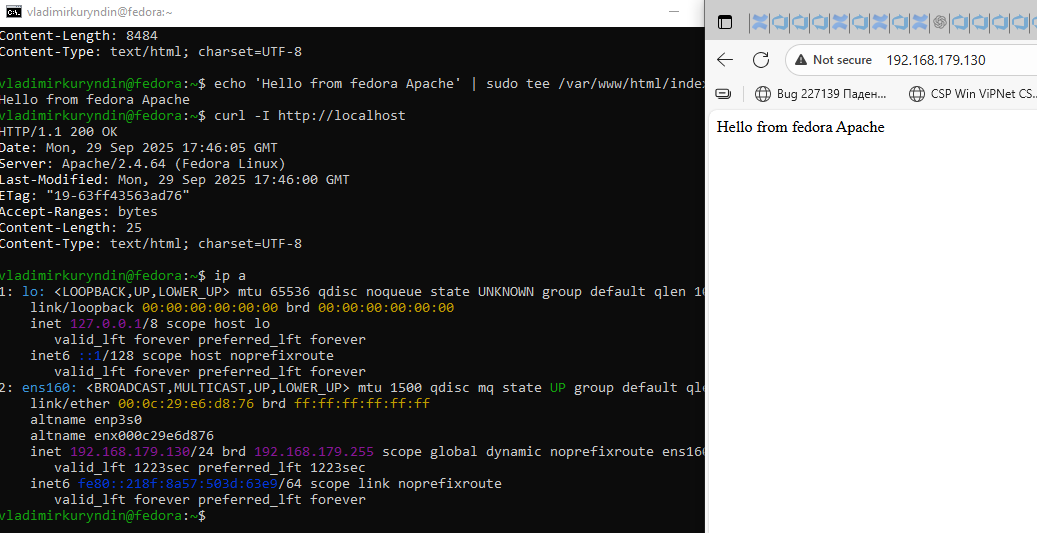
Для начала разрешим в файерволле



Создадим пустую страницу, чтобы не возвращался 403

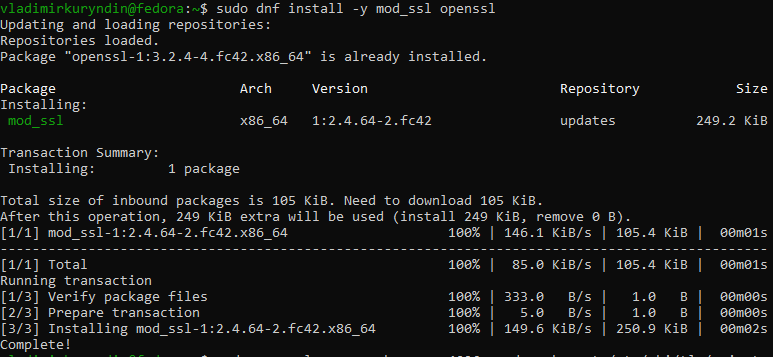


Теперь проверим магическую страницу

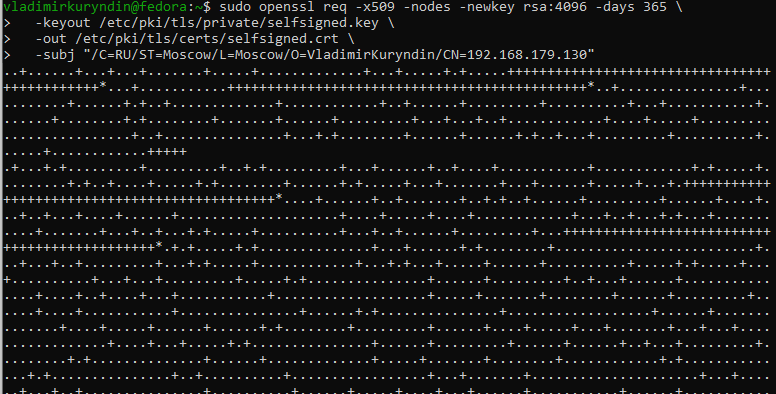


## Для обеспечения безопасности данных настроить HTTPS-подключение с использованием самоподписанных SSL-сертификатов в Apache

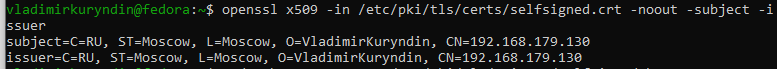
### Ставим mod\_ssl и openssl



### Генерим сертификат



Проверяем



### Настраиваем права



### Создаем отдельный файл конфигурации

После множества попыток все отлаить он получился таким

GNU nano 8.3 /etc/httpd/conf.d/ssl-localhost.conf Listen 443 https

<VirtualHost \*:443>

ServerName 192.168.179.130

DocumentRoot "/var/www/html"

SSLEngine on

SSLCertificateFile /etc/pki/tls/certs/selfsigned.crt

SSLCertificateKeyFile /etc/pki/tls/private/selfsigned.key

# using Fedora system cryptopolicies

SSLCipherSuite PROFILE=SYSTEM

SSLProxyCipherSuite PROFILE=SYSTEM

SSLProtocol all -SSLv3 -TLSv1 -TLSv1.1

<Directory "/var/www/html">

AllowOverride None

Require all granted

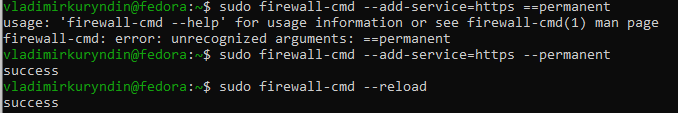
</Directory>

ErrorLog logs/ssl\_error\_log

CustomLog logs/ssl\_access\_log combined

</VirtualHost>

### Настраиваем файерволл



### Финальные проверки

Проверяем, работает ли порт 443

vladimirkuryndin@fedora:~$ sudo ss -tlnp | grep httpd

LISTEN 0 511 \*:443 \*:\* users:(("httpd",pid=5469,fd=6),("httpd",pid=5468,fd=6),("httpd",pid=5467,fd=6),("httpd",pid=5466,fd=6),("httpd",pid=5464,fd=6))

LISTEN 0 511 \*:80 \*:\* users:(("httpd",pid=5469,fd=4),("httpd",pid=5468,fd=4),("httpd",pid=5467,fd=4),("httpd",pid=5466,fd=4),("httpd",pid=5464,fd=4))

Собственно проверяем, что работает https

vladimirkuryndin@fedora:~$ curl -vkI https://192.168.179.130

\* Trying 192.168.179.130:443...

\* ALPN: curl offers h2,http/1.1

\* TLSv1.3 (OUT), TLS handshake, Client hello (1):

\* TLSv1.3 (IN), TLS handshake, Server hello (2):

\* TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):

\* TLSv1.3 (IN), TLS handshake, Certificate (11):

\* TLSv1.3 (IN), TLS handshake, CERT verify (15):

\* TLSv1.3 (IN), TLS handshake, Finished (20):

\* TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):

\* TLSv1.3 (OUT), TLS handshake, Finished (20):

\* SSL connection using TLSv1.3 / TLS\_AES\_256\_GCM\_SHA384 / x25519 / RSASSA-PSS

\* ALPN: server accepted http/1.1

\* Server certificate:

\* subject: C=RU; ST=Moscow; L=Moscow; O=VladimirKuryndin; CN=192.168.179.130

\* start date: Sep 29 18:03:16 2025 GMT

\* expire date: Sep 29 18:03:16 2026 GMT

\* issuer: C=RU; ST=Moscow; L=Moscow; O=VladimirKuryndin; CN=192.168.179.130

\* SSL certificate verify result: self-signed certificate (18), continuing anyway.

\* Certificate level 0: Public key type RSA (4096/152 Bits/secBits), signed using sha256WithRSAEncryption

\* Connected to 192.168.179.130 (192.168.179.130) port 443

\* using HTTP/1.x

> HEAD / HTTP/1.1

> Host: 192.168.179.130

> User-Agent: curl/8.11.1

> Accept: \*/\*

>

\* Request completely sent off

\* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):

\* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):

< HTTP/1.1 200 OK

HTTP/1.1 200 OK

< Date: Mon, 29 Sep 2025 18:54:19 GMT

Date: Mon, 29 Sep 2025 18:54:19 GMT

< Server: Apache/2.4.64 (Fedora Linux) OpenSSL/3.2.4

Server: Apache/2.4.64 (Fedora Linux) OpenSSL/3.2.4

< Last-Modified: Mon, 29 Sep 2025 17:46:00 GMT

Last-Modified: Mon, 29 Sep 2025 17:46:00 GMT

< ETag: "19-63ff43563ad76"

ETag: "19-63ff43563ad76"

< Accept-Ranges: bytes

Accept-Ranges: bytes

< Content-Length: 25

Content-Length: 25

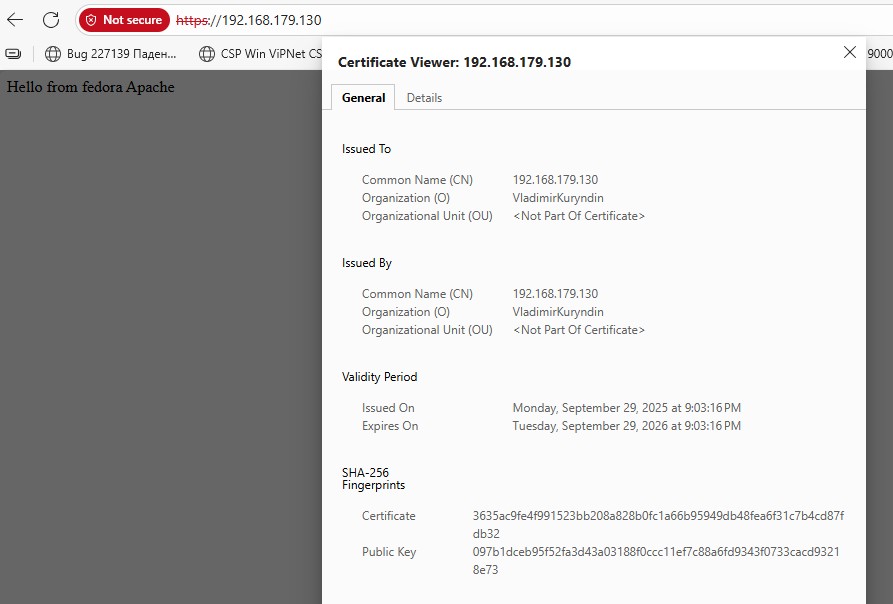
< Content-Type: text/html; charset=UTF-8

Content-Type: text/html; charset=UTF-8

<

\* Connection #0 to host 192.168.179.130 left intact

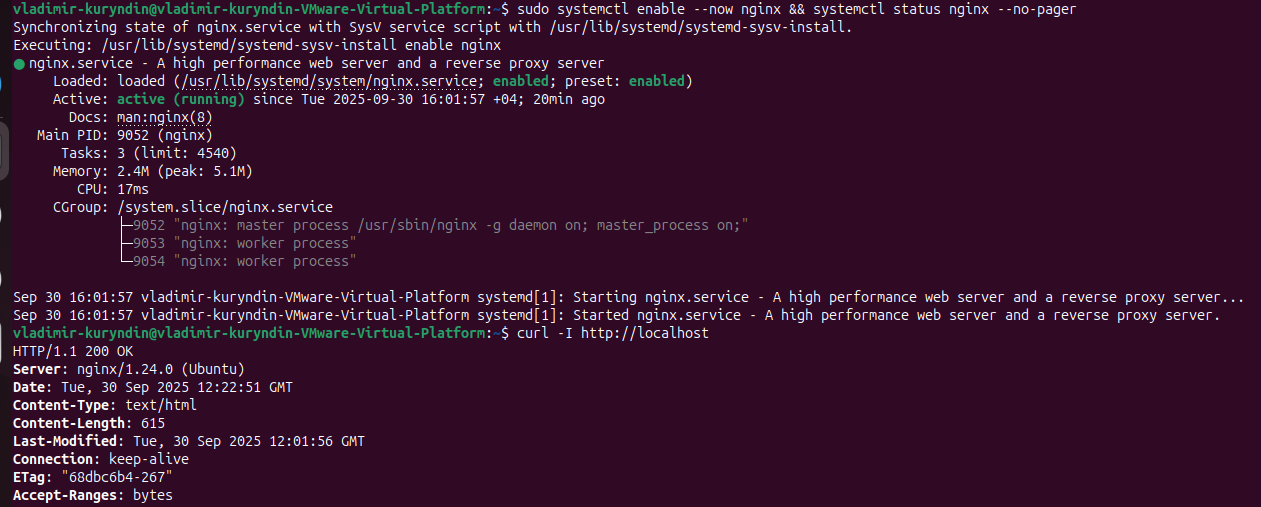
Проверяем в Windows



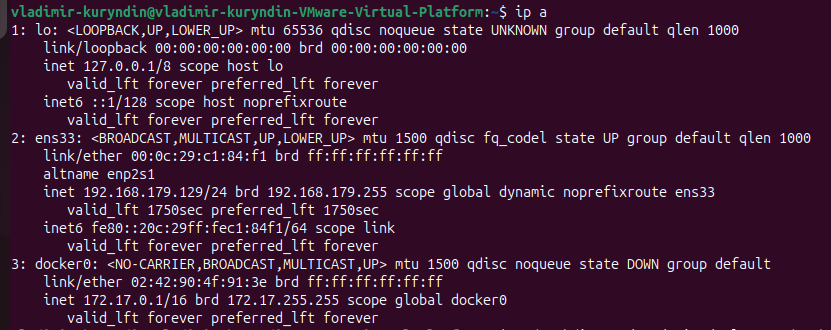
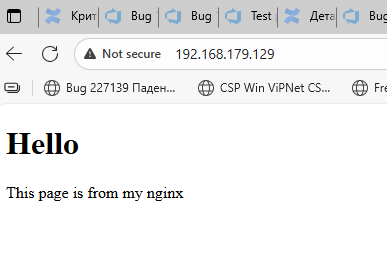
# Часть 2

## Установить Nginx и убедиться, что он работает:

sudo systemctl status nginx



## Просмотреть страницу по URL-адресу для проверки рабочего состояния



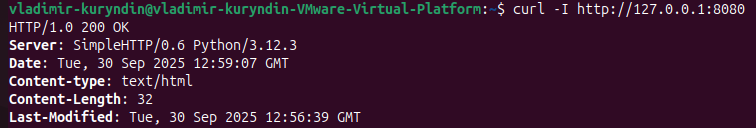
## Настройте прямой и обратный прокси в Nginx для перенаправления запросов

### Обратный (reverse) прокси

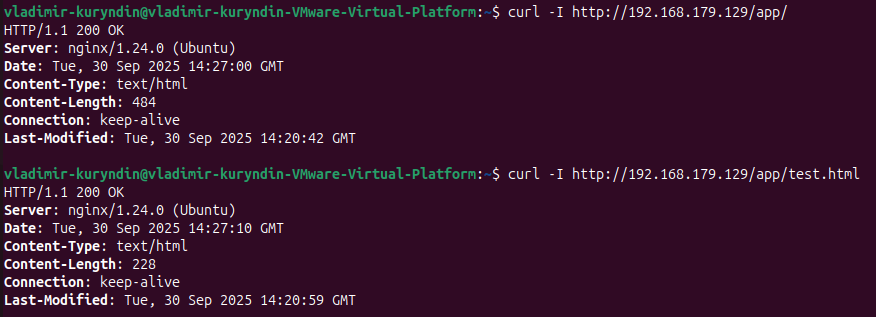
Поднимем простой сервер на Python



Проверим, что он работает



Проверим сам прокси



Клиент → http://192.168.179.129/…

├─ если путь начинается НЕ с /app → Nginx читает файл с диска /var/www/html (без прокси)

└─ если путь начинается с /app → Nginx проксирует на 127.0.0.1:8080

└─ из-за конечного "/" в proxy\_pass префикс /app/ отрезается:

/app/foo → backend получает /foo

/app/docs/ → backend получает /docs/

В конфиг nginx добавили:

#reverse proxy

location /app/ {

proxy\_pass http://127.0.0.1:8080/;

allow 192.168.179.0/24

deny all;

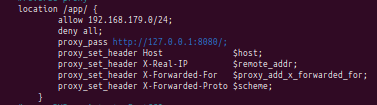
proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

}



### Прямой (forward) прокси

В конфиг nginx добавляем

server {

listen 8888;

# DNS-resolvers for proxy\_pass with variables

resolver 127.0.0.53 valid=30s ipv6=off;

resolver\_timeout 5s;

# optional: disable access to the proxy for all hosts outside my network

allow 192.168.179.0/24;

deny all;

# all requests will be proxied to the URL, which came from the client

location / {

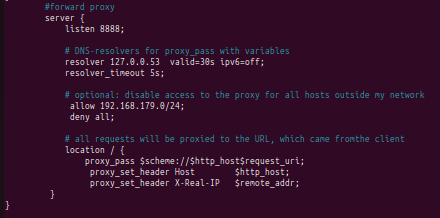
proxy\_pass $scheme://$http\_host$request\_uri;

proxy\_set\_header Host $http\_host;

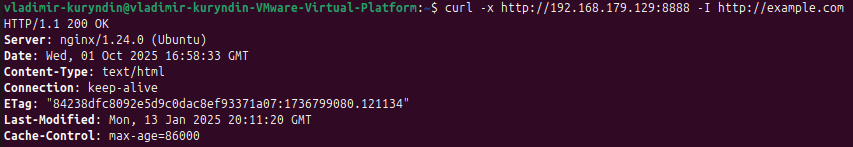
proxy\_set\_header X-Real-IP $remote\_addr;

}

}

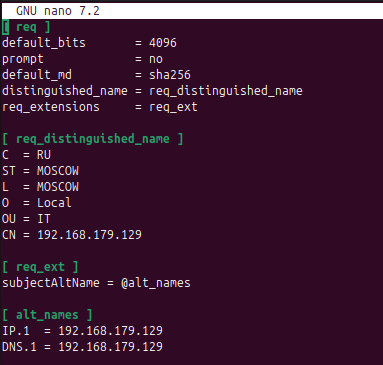


Проверяем



## Для обеспечения безопасности данных настроить HTTPS-подключение с использованием самоподписанных SSL-сертификатов в Nginx

Создаем конфиг для OpenSSL для выпуска сертификата со всеми нужными нам полями

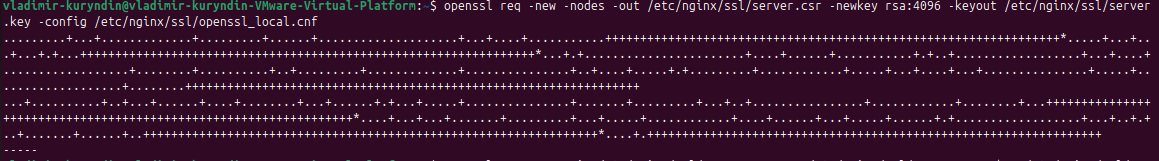


Я решил сделать сертфикат с SAN и использовать корневой центр сертификации CA (чтобы на windows мой Edge корректно обработал сертификат, учесть неточности в выпуске сертификатов при настройке Apache (см выше))

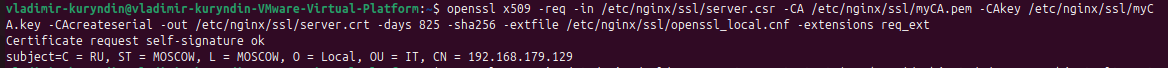
Далее создаем ключ СA и сертификат CA (корневой)



Делаем CSR и ключ сервера по нашему файлу конфигурации OpenSSL



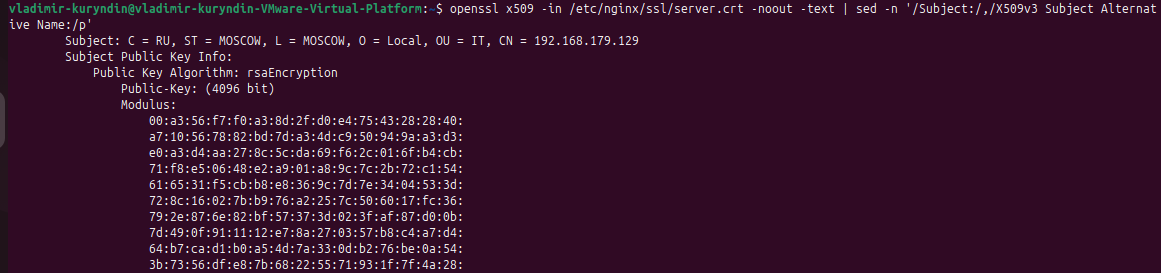
Подписываем CSR нашим CA и получаем доверенный серверный сертификат



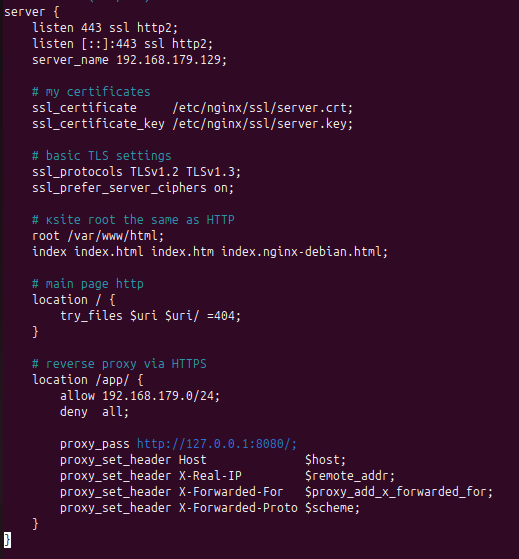
Проверяем, что SAN присутствует



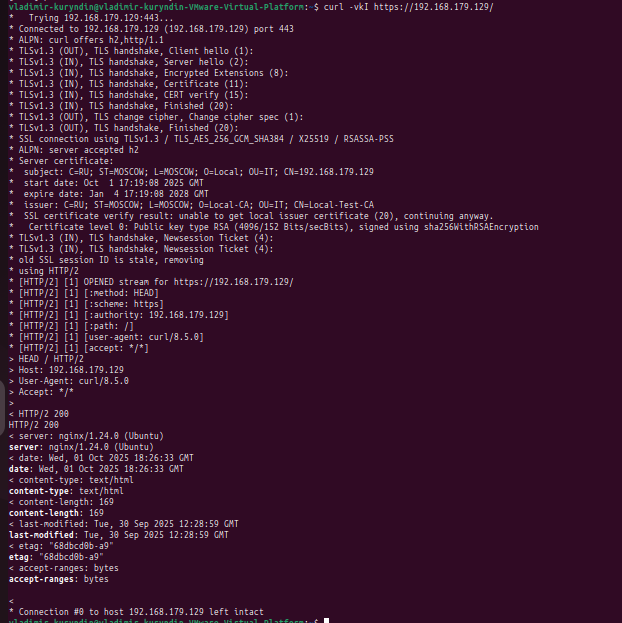
Также проверим ,что с сертифкатом все ок



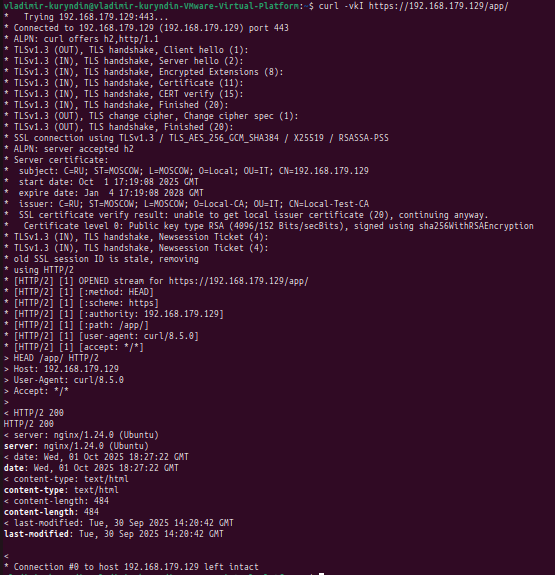
Добавим новую секцию в файл конфигурации nginx



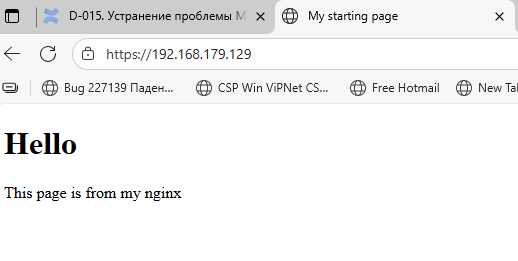
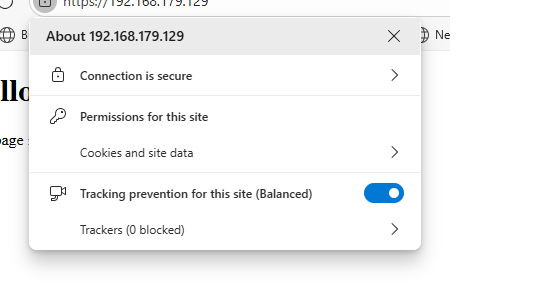
Проверка

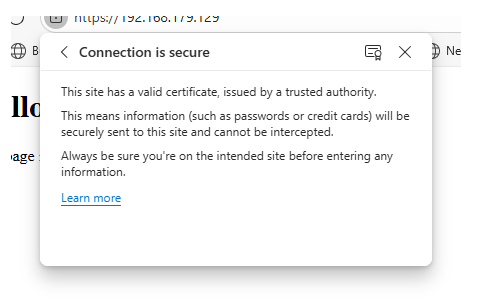


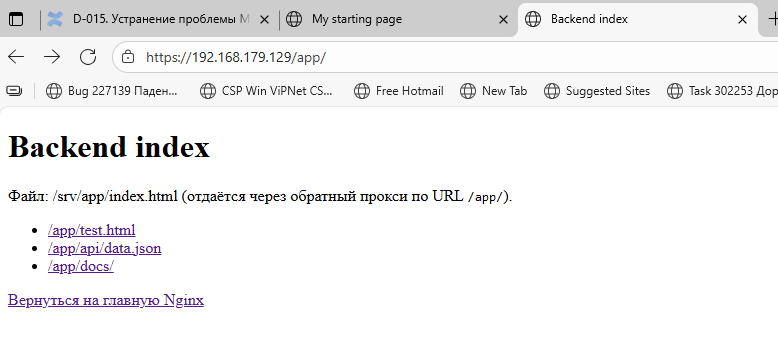
Проверка, что работает reverse proxy на /app/

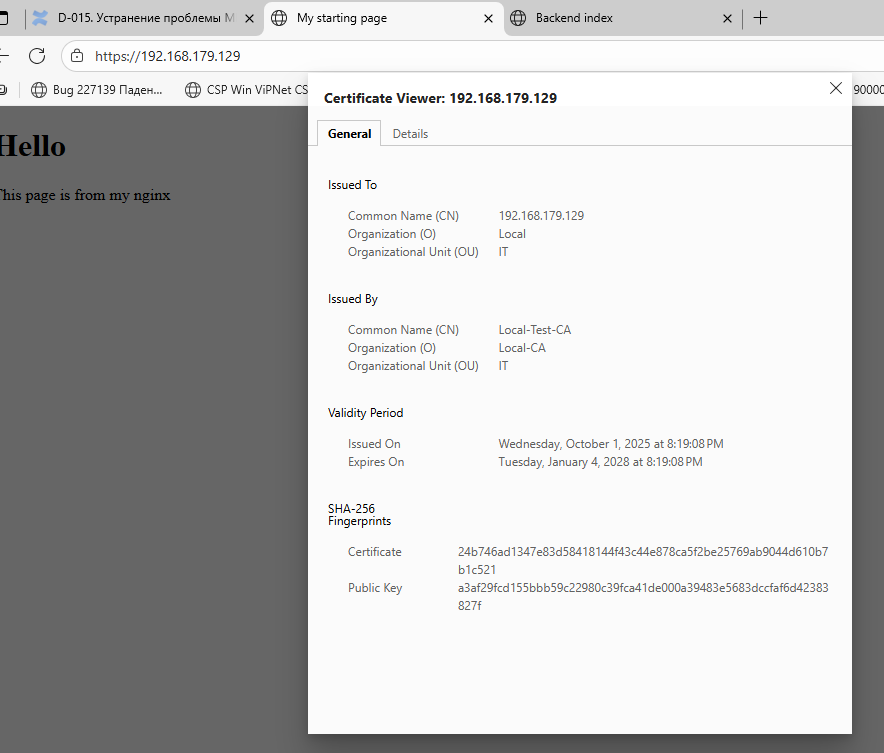


Ну и самая главная проверка на windows







## Установить и настроить ModSecurity, в том числе настроить фильтрацию запросов для обеспечения безопасности от SQL-запросов

### Установка ModSecurity

Ставим Modsecurity (попытка поставить из исходников не увенчалась успехом)



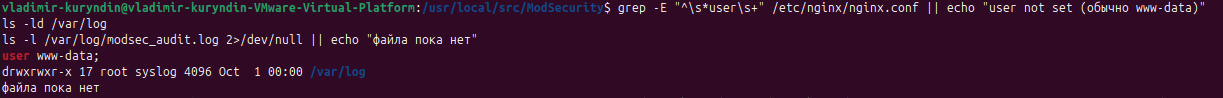
### Настройка ModSecurity

Далее нам надо настроить modsecurity.conf. Я нашел файл modsecurity.conf-recommened и скопировал его в modsecurity.conf.

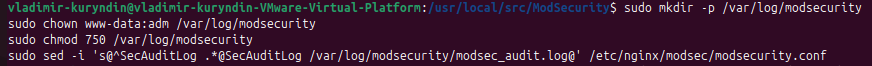




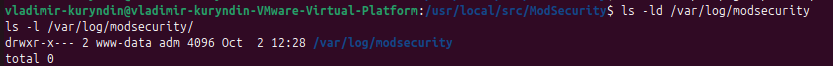
Далее я настроил файл журнала событий для modsecurity. Для этого пришлось узнать под каким пользователем работает nginx и разрешить этому пользователю работать с логом modsecurity



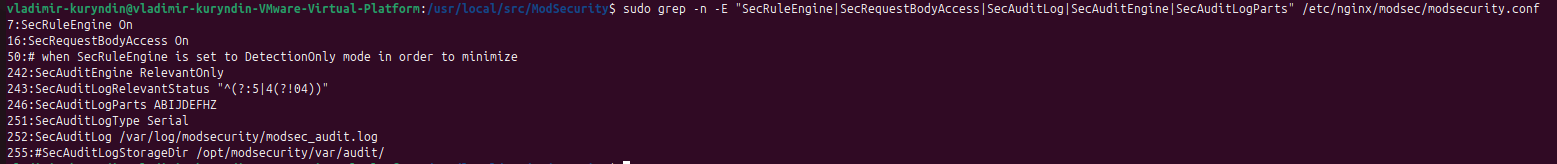
Далее создал папку и файл лога



Проверил, что правильно поставили права на папку с логами

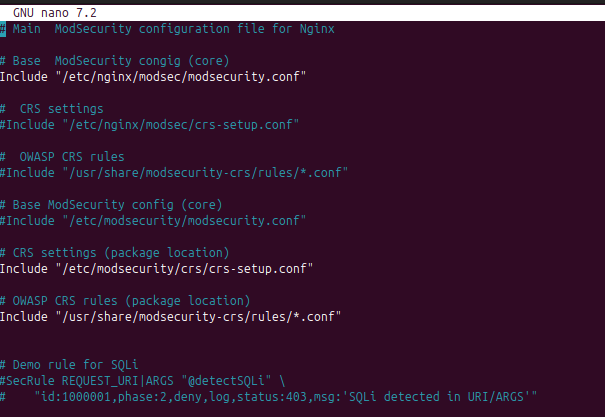


Далее внес изменения в файл modsecurity.conf и проверил, что все ОК. В том числе убедился, что папка с логами указана верно.

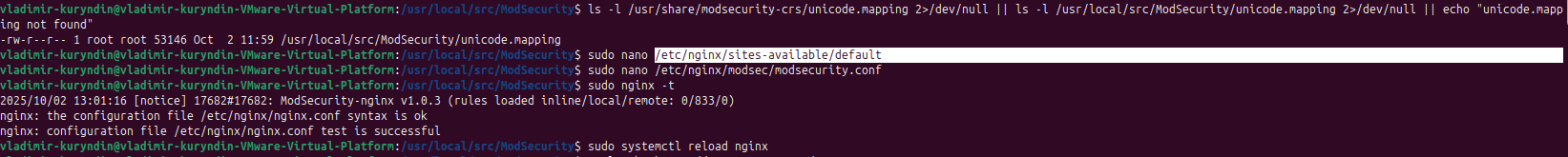


### Создание файла main.conf

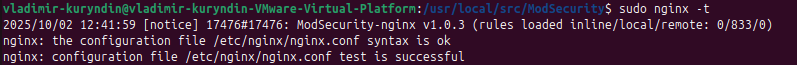
Создал основной файл конфигурации ModSecurity для Nginx - main.conf. Экспериментировал с путями, сначала немножко напутал с путями, но потом разобрался, неправильные пути оставил для себя закомментированными.



Далее возникла проблема с Unicode mapping, я ее решил поиском файла и явном указании на него в файле /etc/nginx/sites-available/default



Проверил, что все работает



## Продемонстрировать успешную блокировку SQL-инъекций

### Демонстрация работы Modsecurity (в примере показана XSS, вам необходимо показать на SQLi)

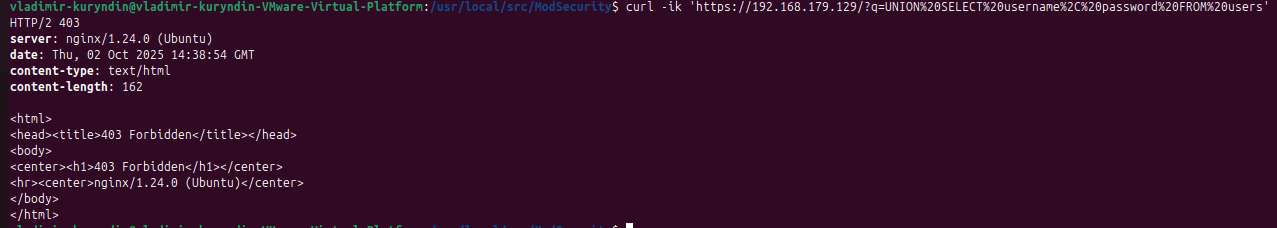
#### Показать, что при отправке запросов командой curl выводится ошибка 403

##### XSS



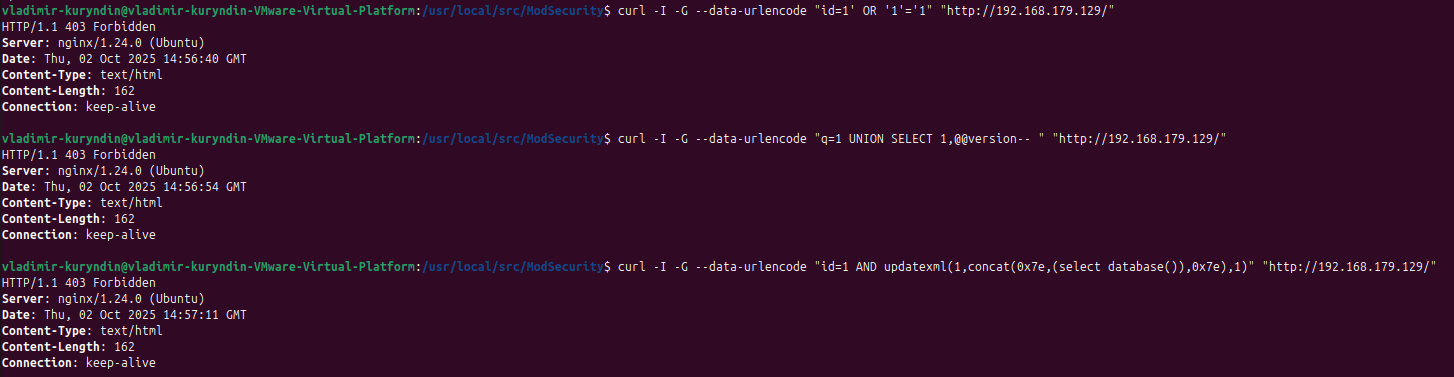
##### SQLi

Показываем на https

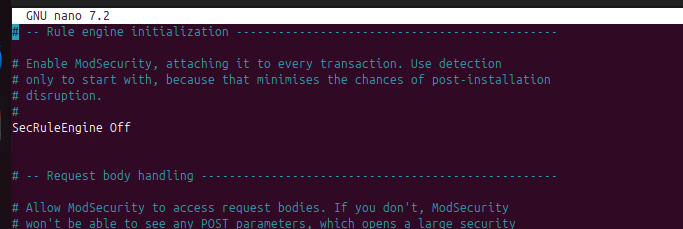


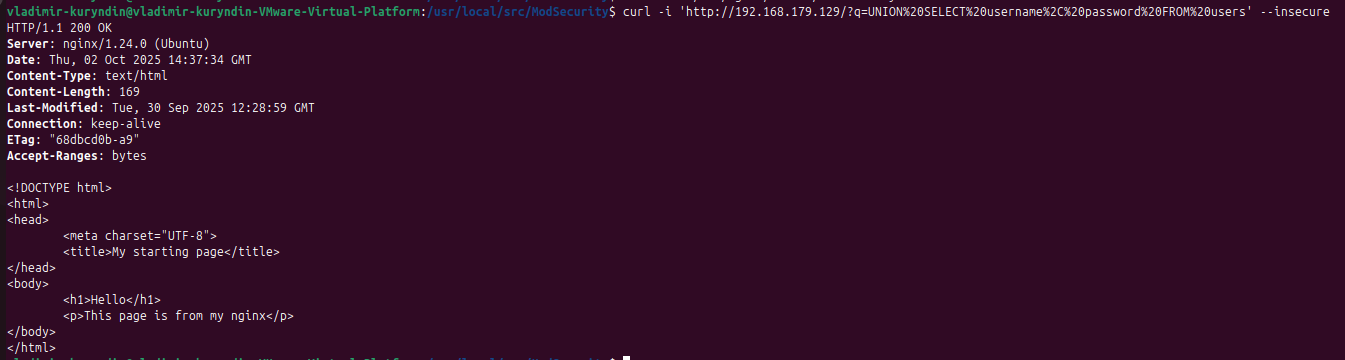
И на обычном http





### Закомментировать правило в файле конфигурации Modsecurity и показать, что Nginx возвращает код 200





## Дополнительная проверка различных паттернов

