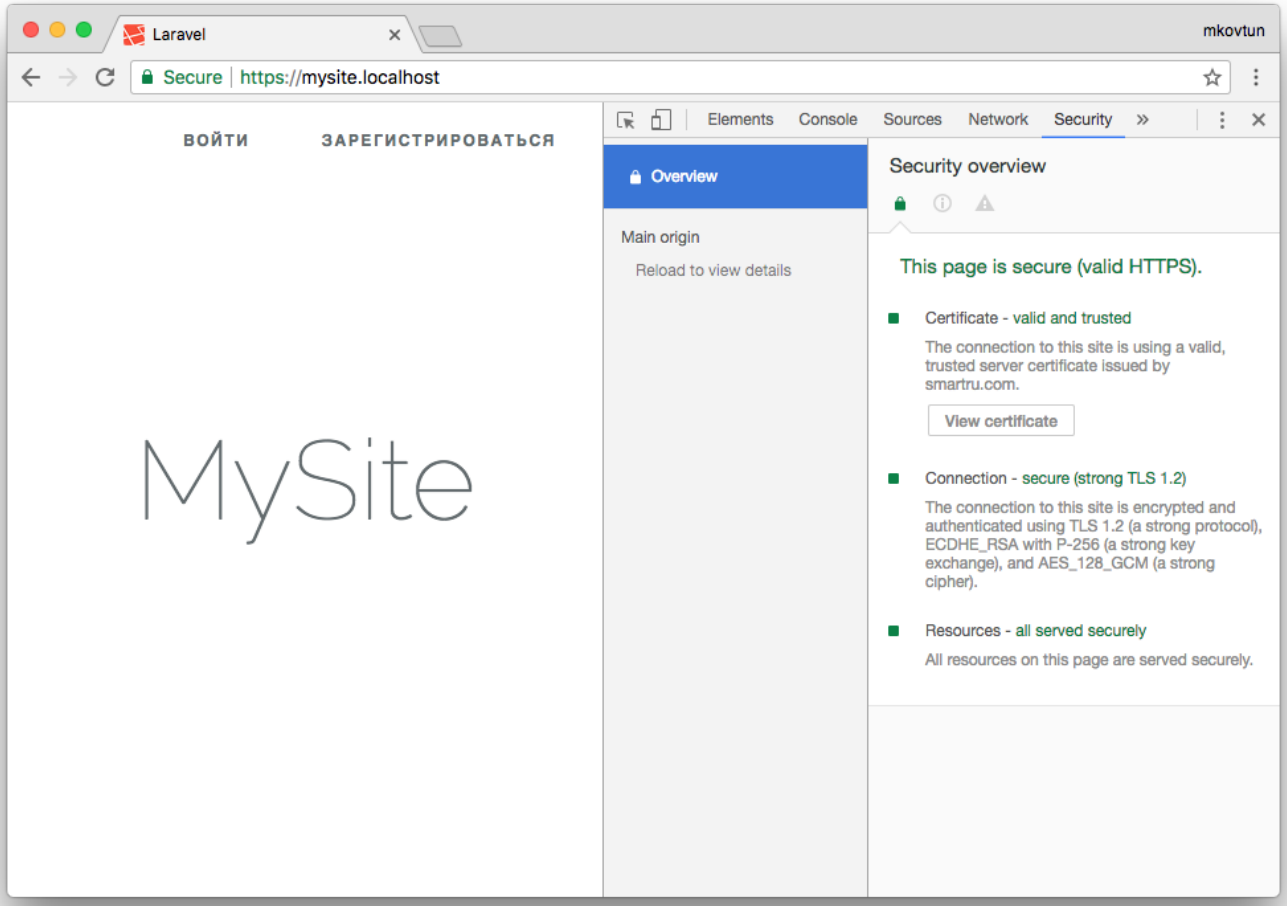
evAPPs
3 апр 2018 в 17:31

Как выпустить самоподписанный SSL сертификат и заставить ваш браузер доверять ему

3 мин 244K

Веб-разработка*

Туториал



Все крупные сайты давно перешли на протокол https. Тенденция продолжается, и многие наши клиенты хотят, чтобы их сайт работал по защищенному протоколу. А если разрабатывается backend для мобильного приложения, то https обязателен. Например, Apple требует, чтобы обмен данными сервера с приложением велся по безопасному протоколу. Это

На production нет проблем с сертификатами. Обычно хостинг провайдер предоставляет удобный интерфейс для подключения сертификата. Выпуск сертификата тоже дело не сложное. Но во время работы над проектом каждый разработчик должен позаботиться о сертификате сам.
В этой статье я расскажу, как выпустить самоподписанный SSL сертификат и заставить браузер доверять ему.

Чтобы выпустить сертификат для вашего локального домена, понадобится корневой сертификат. На его основе будут выпускаться все остальные сертификаты. Да, для каждого нового top level домена нужно выпускать свой сертификат. Получить корневой сертификат достаточно просто.
Сначала сформируем закрытый ключ:

```
openssl genrsa -out rootCA.key 2048
```

Затем сам сертификат:

```
openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 1024 -out rootCA.pem
```

Нужно будет ввести *страну*, *город*, *компанию* и т.д. В результате получаем два файла: rootCA.key и rootCA.pem

Переходим к главному, выпуск самоподписанного сертификата. Так же как и в случае с корневым, это две команды. Но параметров у команд будет значительно больше. И нам понадобится вспомогательный конфигурационный файл. Поэтому оформим все это в виде bash скрипта create_certificate_for_domain.sh

Первый параметр обязателен, выведем небольшую инструкцию для пользователя.

```
if [ -z "$1" ]
then
  echo "Please supply a subdomain to create a certificate for";
  echo "e.g. mysite.localhost"
```

```
  exit;
fi
```

Создадим новый приватный ключ, если он не существует или будем использовать существующий:

```
if [ -f device.key ]; then
  KEY_OPT="-key"
else
  KEY_OPT="-keyout"
fi
```

Запросим у пользователя название домена. Добавим возможность задания “общего имени” (оно используется при формировании сертификата):

```
DOMAIN=$1
COMMON_NAME=${2:-$1}
```

Чтобы не отвечать на вопросы в интерактивном режиме, сформируем строку с ответами. И зададим время действия сертификата:

```
SUBJECT="/C=CA/ST=None/L=NB/O=None/CN=$COMMON_NAME"
NUM_OF_DAYS=999
```

В переменной SUBJECT перечислены все те же вопросы, который задавались при создании корневого сертификата (*страна*, *город*, *компания* и т.д). Все значение, кроме CN можно поменять на свое усмотрение.

Сформируем *csr* файл (Certificate Signing Request) на основе ключа. Подробнее о файле запроса сертификата можно почитать в этой статье.

```
openssl req -new -newkey rsa:2048 -sha256 -nodes $KEY_OPT device.key -subj "$SUBJECT" -out
```

Формируем файл сертификата. Для этого нам понадобится вспомогательный файл с настройками. В этот файл мы запишем домены, для которых будет валиден сертификат и некоторые другие настройки. Назовем его v3.ext. Обращаю ваше внимание, что это отдельный файл, а не часть bash скрипта.

```
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment
subjectAltName = @alt_names

[alt_names]
DNS.1 = %%DOMAIN%%
DNS.2 = *.%%DOMAIN%%
```

Да, верно, наш сертификат будет валидным для основного домена, а также для всех поддоменов. Сохраняем указанные выше строки в файл v3.ext

Возвращаемся в наш bash скрипт. На основе вспомогательного файла v3.ext создаем временный файл с указанием нашего домена:

```
cat v3.ext | sed s/%%DOMAIN%%/$COMMON_NAME/g > /tmp/__v3.ext
```

Выпускаем сертификат:

```
openssl x509 -req -in device.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial -out dev
```

Переименовываем сертификат и удаляем временный файл:

```
mv device.csr $DOMAIN.csr
cp device.crt $DOMAIN.crt
```

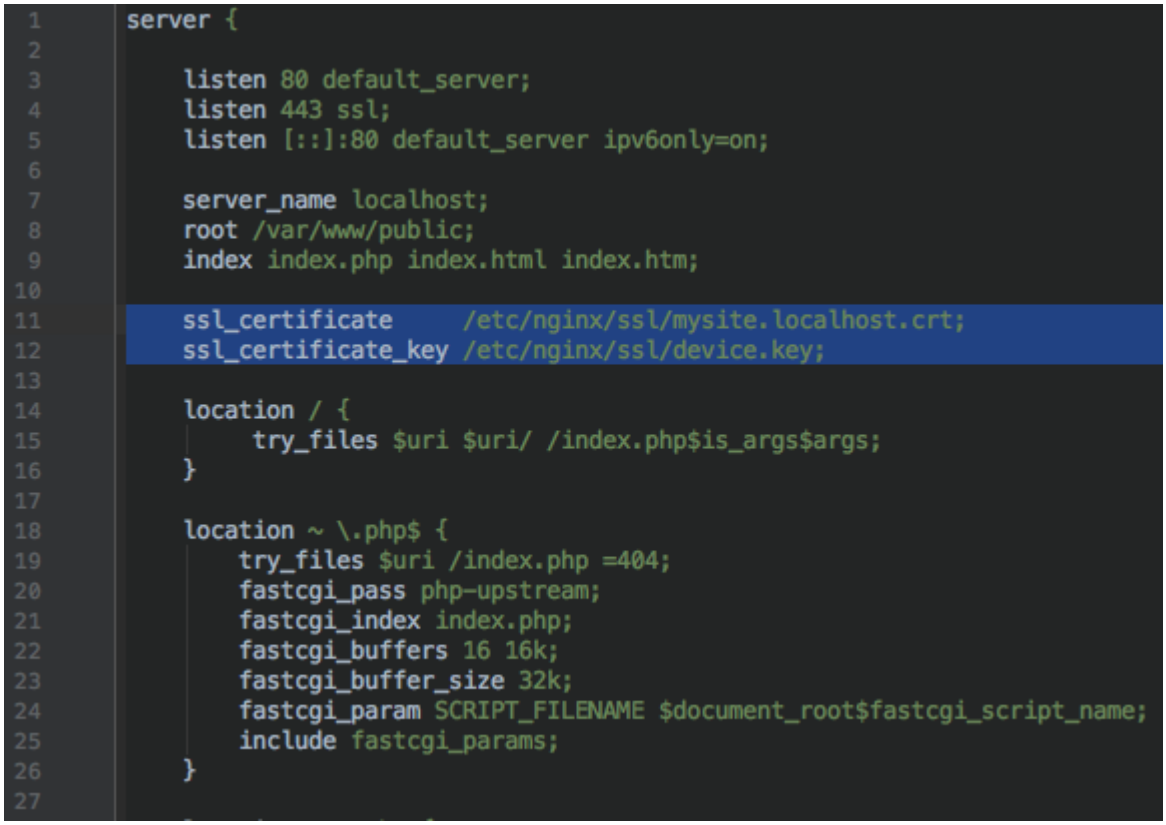
```
# remove temp file
rm -f device.crt;
```

Скрипт готов. Запускаем его:

```
./create_certificate_for_domain.sh mysite.localhost
```

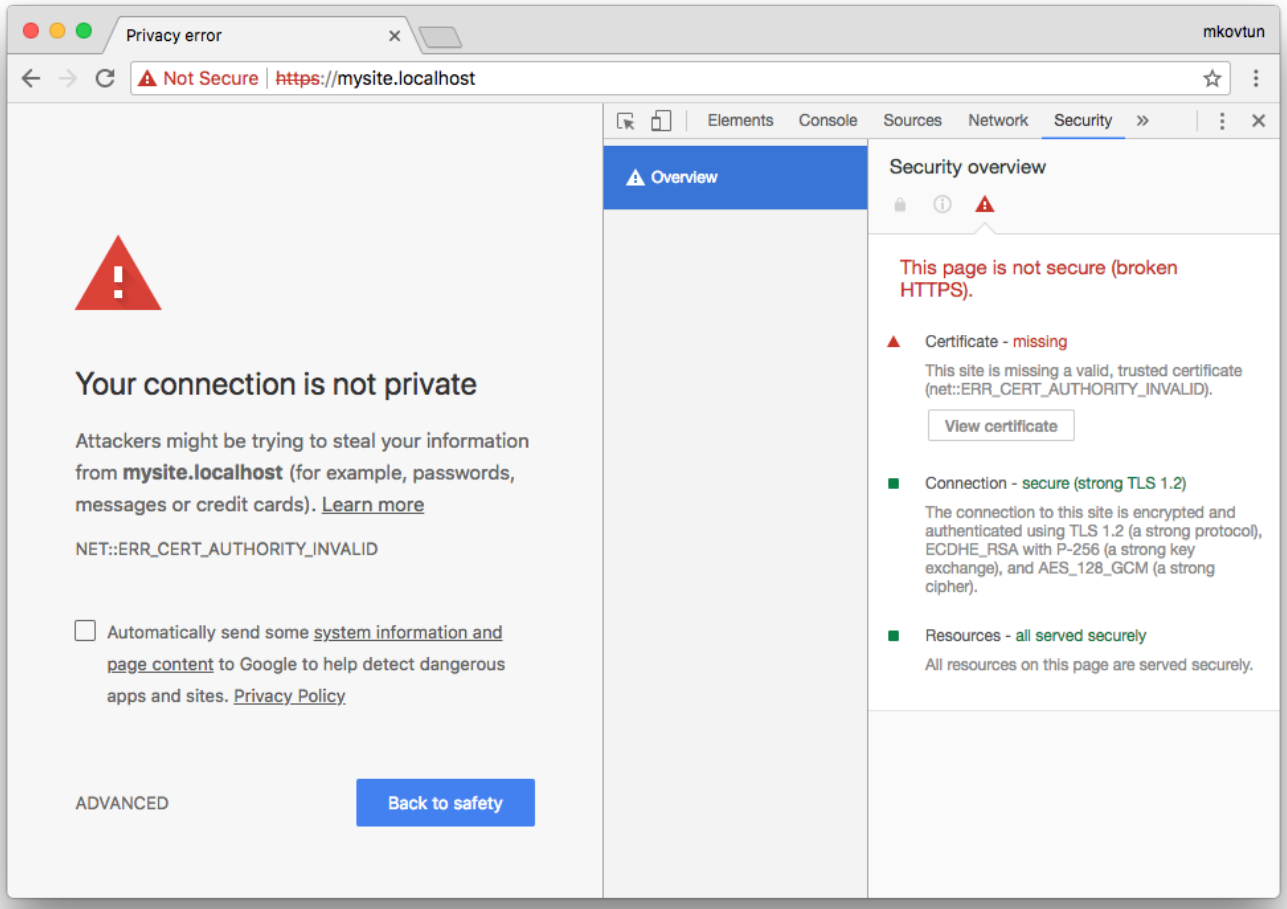
Получаем два файла: mysite.localhost.crt и device.key

Теперь нужно указать web серверу пути к этим файлам. На примере nginx это будет выглядеть так:



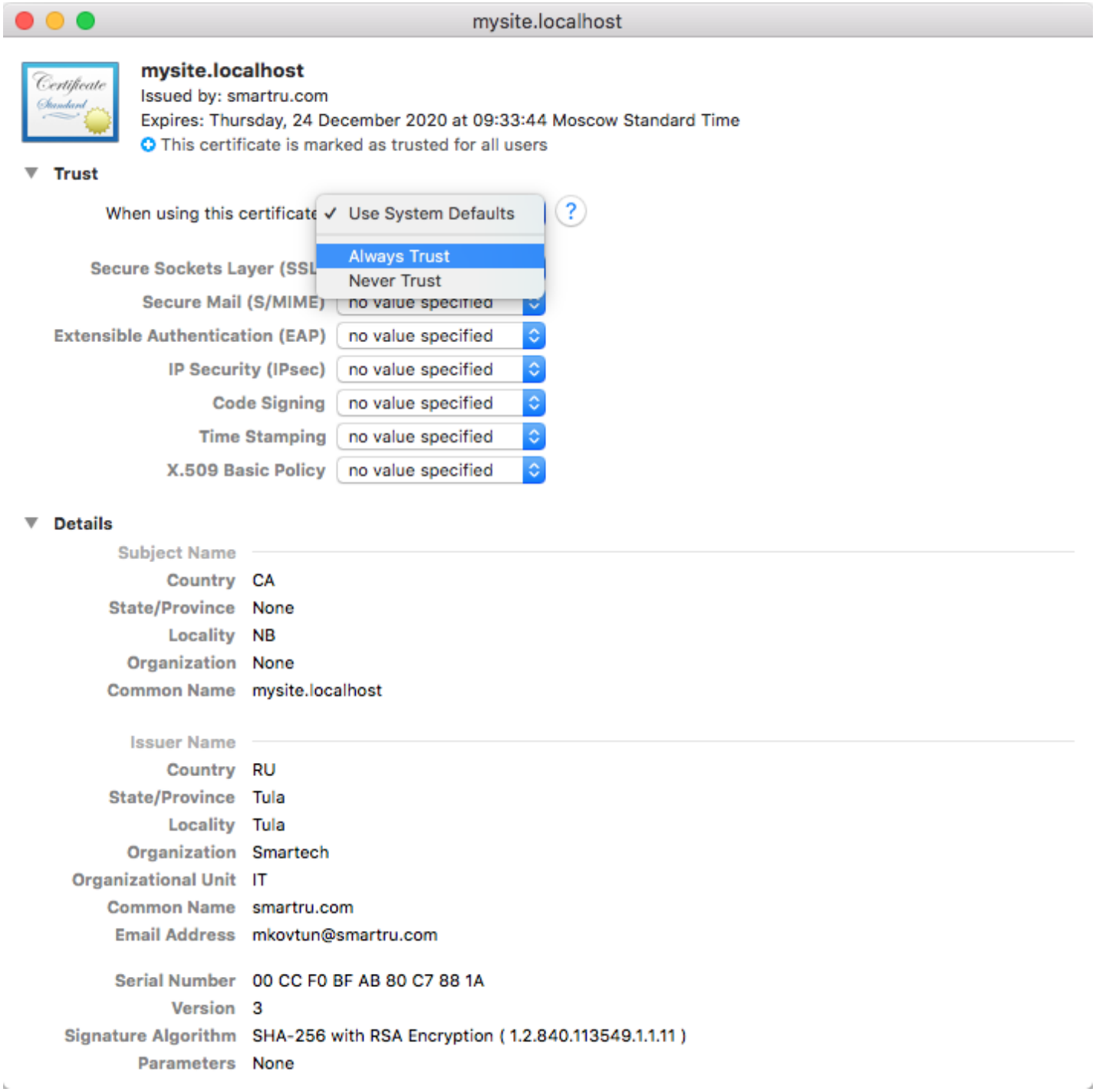
```
server {

    listen 80 default_server;
    listen 443 ssl;
    listen [::]:80 default_server ipv6only=on;

    server_name localhost;
    root /var/www/public;
    index index.php index.html index.htm;

    ssl_certificate     /etc/nginx/ssl/mysite.localhost.crt;
    ssl_certificate_key /etc/nginx/ssl/device.key;

    location / {
        try_files $uri $uri/ /index.php$is_args$args;
    }

    location ~ \.php$ {
        try_files $uri /index.php =404;
        fastcgi_pass php-upstream;
        fastcgi_index index.php;
        fastcgi_buffers 16 16k;
        fastcgi_buffer_size 32k;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
    }

```

Запускаем браузер, открываем https://mysite.localhost и видим:

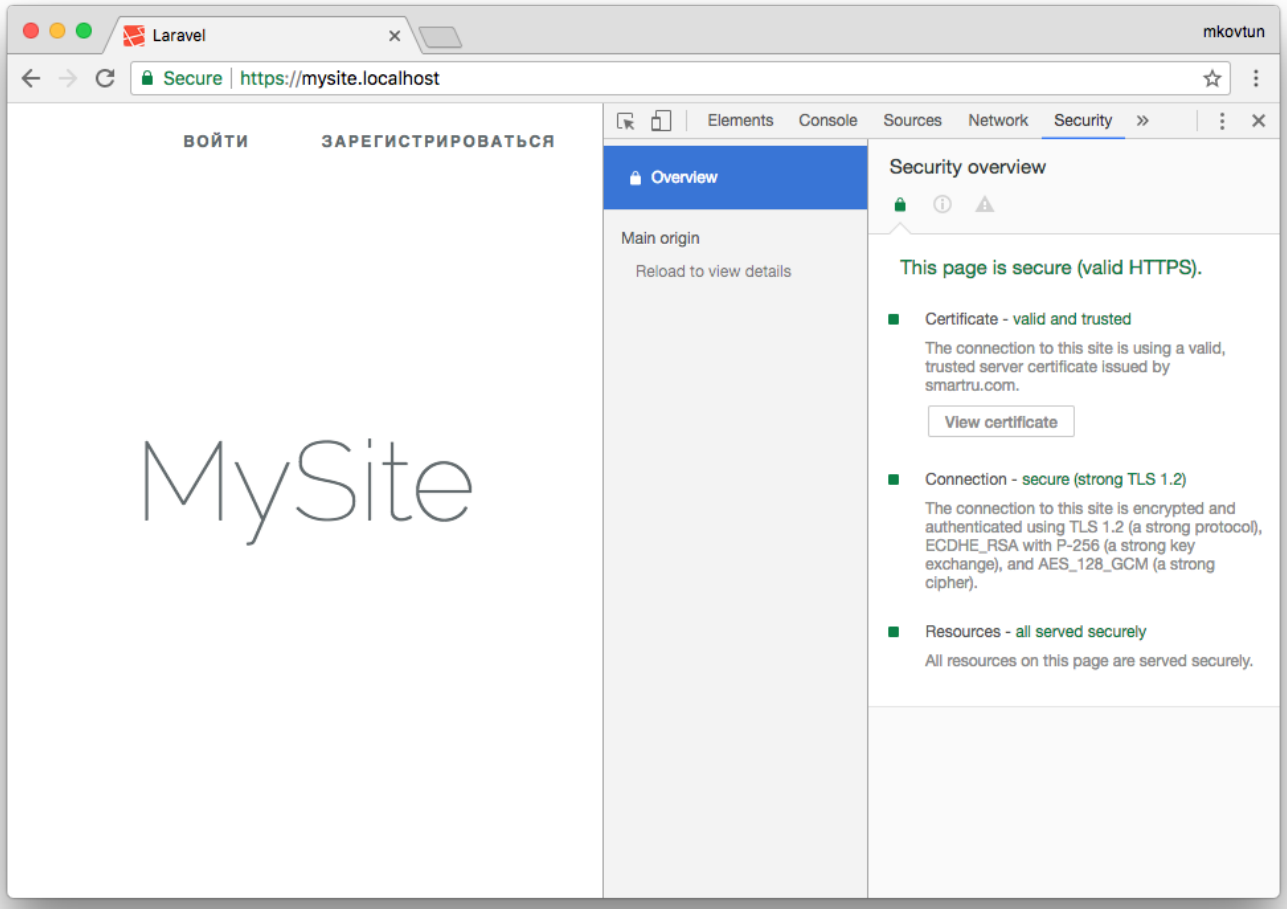


Браузер не доверяет этому сертификату. Как быть?

Нужно отметить выпущенный нами сертификат как Trusted. На Linux (Ubuntu и, наверное, остальных Debian-based дистрибутивах) это можно сделать через сам браузер. В Mac OS X это можно сделать через приложение Keychain Access. Запускаем приложение и перетаскиваем в окно файл mysite.localhost.crt. Затем открываем добавленный файл и выбираем Always Trust:



Обновляем страницу в браузере и:



Успех! Браузер доверяет нашему сертификату.

Сертификатом можно поделиться с другими разработчиками, чтобы они добавили его к себе. А если вы используете Docker, то сертификат можно сохранить там. Именно так это реализовано на всех наших проектах.

Делитесь в комментариях, используете ли вы https для локальной разработки?

Максим Ковтун,
Руководитель отдела разработки

Теги: bash-скрипт, ssl сертификаты, https протокол, google chrome, mac os x, openssl

Хабы: Веб-разработка

14 Карма 5 Рейтинг

@evAPPs

Пользователь

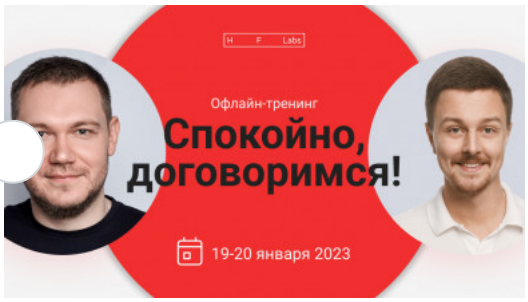
Комментарии 51

Публикации

ЛУЧШИЕ ЗА СУТКИ ПОХОЖИЕ

ИСТОРИИ

БЛИЖАЙШИЕ СОБЫТИЯ



«Спокойно, договоримся!» — тренинг по переговорам и отношениям с клиентами в B2B

19 – 20 января 10:00 – 18:00

Москва

Подробнее в календаре

Be Mi

П