Создание CA на базе OpenSSL для установки mTLS в браузере и в любых запросах.

На основе статьи https://habr.com/ru/articles/671730/

```
root@localhost: Student$openssl version
OpenSSL 1.0.2k-fips  26 Jan 2017
root@localhost: Student$
```

Генерируем ключ для сервера:

`openssl genrsa -out root_ca.key 2048`

```
root@localhost: Student$cd /opt/
root@localhost: opt$mkdir CA
root@localhost: opt$cd CA/
root@localhost: CA$openssl genrsa -out root_ca.key 2048
Generating RSA private key, 2048 bit long modulus
...+++
..............+++
e is 65537 (0x10001)
root@localhost: CA$
```

Получаем самоподписанный сертификат:

`openssl req -x509 -new -key root_ca.key -days 365 -out root_ca.crt`

```
root@localhost: CA$openssl req -x509 -new -key root_ca.key -days 365 -out root_ca.crt
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:
State or Province Name (full name) []:
Locality Name (eg, city) [Default City]:
Organization Name (eg, company) [Default Company Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:192.168.0.9
Email Address []:
root@localhost: CA$ll
итого 8
-rw-r--r--. 1 root root 1277 июн  3 23:09 root_ca.crt
-rw-r--r--. 1 root root 1679 июн  3 23:07 root_ca.key
root@localhost: CA$
```

Генерируем приватный ключ для HTTPS сервера

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

```
root@localhost: CA$openssl req -new -key server.key -subj "/CN=192.168.0.9" -out server.csr
root@localhost: CA$ll
итого 16
-rw-r--r--. 1 root root 1277 июн  3 23:09 root_ca.crt
-rw-r--r--. 1 root root 1679 июн  3 23:07 root_ca.key
-rw-r--r--. 1 root root  895 июн  3 23:10 server.csr
-rw-r--r--. 1 root root 1675 июн  3 23:10 server.key
root@localhost: CA$
```

Генерируем сертификат X.509 (server.crt) на 365 дней для HTTPS сервера и подписываем его приватным ключом CA

```
openssl x509 -req -in server.csr -CA root_ca.crt -CAkey root_ca.key -CAcreateserial -out server.crt -days 365 -extensions SAN
```

```
root@localhost: CA$openssl x509 -req -in server.csr -CA root_ca.crt -CAkey root_ca.key -CAcreateserial -out server.crt -days 365 -extensions SAN
Signature ok
subject=/CN=192.168.0.9
Getting CA Private Key
root@localhost: CA$
```

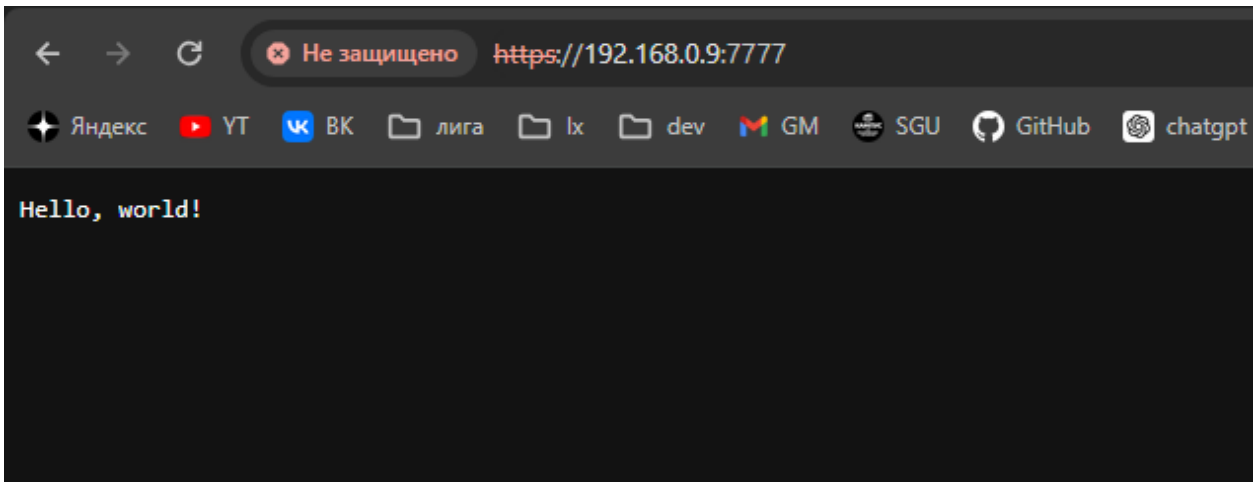
Копируем root_ca.crt, server.key, server.crt на сервер с приложением (это эта же ВМ), и пишем небольшой сервер, работающий на https (и не забываем открыть соответствующий порт):

```
root@localhost: CA$mkdir /opt/server
root@localhost: CA$cp root_ca.crt server.key server.crt /opt/server/
root@localhost: CA$mkdir /opt/server/app/
root@localhost: CA$vi /opt/server/app/server.js
root@localhost: CA$
```

```
const fs = require('fs');
const https = require('https');

https
  .createServer(
    {
      requestCert: false,
      rejectUnauthorized: false,
      ca: fs.readFileSync('root_ca.crt'),
      cert: fs.readFileSync('server.crt'),
      key: fs.readFileSync('server.key')
    },
    (req, res) => {
      console.log("req.client.authorized: ", req.client.authorized)
      if (req.client.authorized) {
        const cert = req.socket.getPeerCertificate(false) // true - получить все сертификаты, false - последний
        console.log("cert: ", cert)
    }
      res.end('Hello, world!');
    }
  )
  .listen(7777);
~
```

Можем получить доступ к серверу, но видим предупреждения о проблемах с сертификатом:




```
PS C:\Users\IgorG> curl https://192.168.0.9:7777/
curl : Базовое соединение закрыто: Не удалось установить доверительные отношения для защищенного канала SSL/TLS.
строка:1 знак:1
+ curl https://192.168.0.9:7777/
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation: (System.Net.HttpWebRequest:HttpWebRequest) [Invoke-WebRequest], WebExc
   eption
    + FullyQualifiedErrorId : WebCmdletWebResponseException,Microsoft.PowerShell.Commands.InvokeWebRequestCommand

PS C:\Users\IgorG>
```

Это значит, что самоподписанный корневой сертификат нашего CA (root_ca.crt) не находится в хранилище доверительных сертификатов. Процесс загрузки сертификатов в это хранилище специфичен для операционной системы. Пример для CentOS7

```
root@localhost: CA$openssl x509 -in root_ca.crt -out root_ca.pem -outform PEM
root@localhost: CA$cp root_ca.pem /etc/pki/ca-trust/source/anchors
root@localhost: CA$
root@localhost: CA$update-ca-trust
root@localhost: CA$cp root_ca.pecurl https://192.168.0.9:7777/
Hello, world!root@localhost: CA$curl https://192.168.0.9:7777/
Hello, world!root@localhost: CA$curl https://192.168.0.9:7777/
Hello, world!root@localhost: CA$
root@localhost: CA$curl https://192.168.0.9:7777/
Hello, world!root@localhost: CA$
```

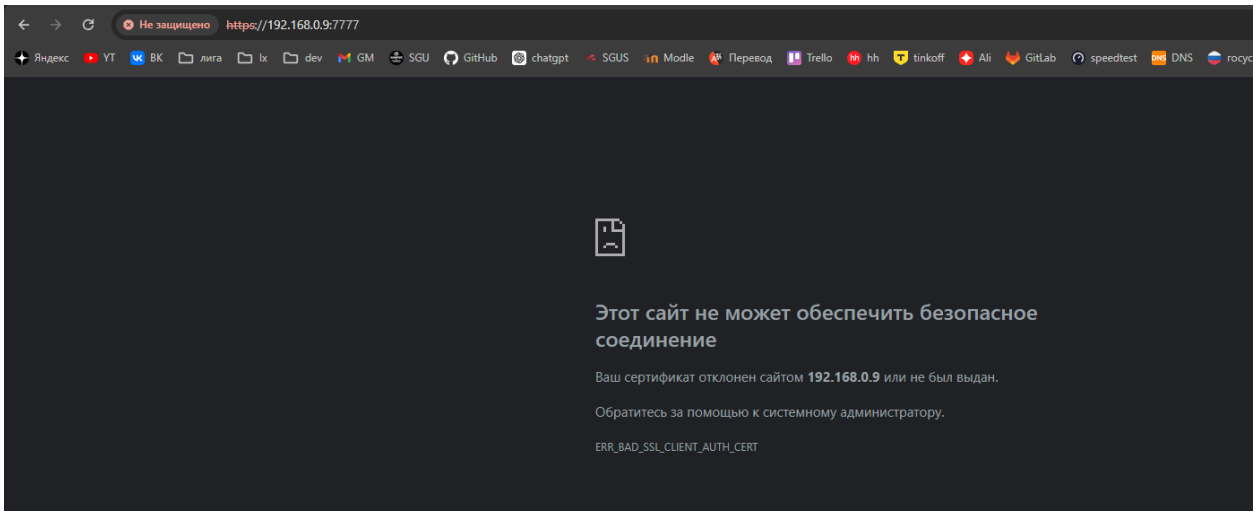
Реализация mTLS

В коде сервера включаем mTLS, а именно выставляем в true параметр requestCert, чтобы HTTPS сервер запрашивал клиентский сертификат и параметр rejectUnauthorized, чтобы сервер отклонял запросы клиентов, которые не прошли авторизацию:

```
{
  requestCert: true,
```

```
  rejectUnauthorized: true,
}
```

Теперь клиент без ключа не может попасть на сайт:



Алгоритм генерации клиентских сертификатов аналогичен серверным.

Генерируем приватный ключ для клиента:

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

Генерируем запрос на сертификат CSR со списком CN:

```
openssl req -new -key client.key -subj "/CN= 192.168.0.9 " -out client.csr
```

```
Hello, world!root@localhost: CA$openssl genrsa -out client.key 2048
Generating RSA private key, 2048 bit long modulus
...................+++
..............................+++
e is 65537 (0x10001)
root@localhost: CA$openssl req -new -key client.key -subj "/CN= 192.168.0.9 " -out client.csr
root@localhost: CA$ll
итого 36
-rw-r--r--. 1 root root  895 июн  4 02:10 client.csr
-rw-r--r--. 1 root root 1675 июн  4 02:10 client.key
-rw-r--r--. 1 root root 1277 июн  3 23:09 root_ca.crt
-rw-r--r--. 1 root root 1679 июн  3 23:07 root_ca.key
-rw-r--r--. 1 root root 1277 июн  3 23:23 root_ca.pem
-rw-r--r--. 1 root root   17 июн  3 23:11 root_ca.srl
-rw-r--r--. 1 root root 1070 июн  3 23:11 server.crt
-rw-r--r--. 1 root root  895 июн  3 23:10 server.csr
-rw-r--r--. 1 root root 1675 июн  3 23:10 server.key
root@localhost: CA$
```

Генерируем сертификат для клиента и подписываем его приватным ключом CA:

```
openssl x509 -req -in client.csr -CA root_ca.crt -CAkey root_ca.key -CAcreateserial -out client.crt -days 365 -extensions SAN -extfile openssl.cnf
```

Копируем следующие файлы на компьютер, на котором будет работать клиент: root_ca.crt, client.key, client.crt

```
root@localhost: CA$cp root_ca.crt client.key client.crt /client/
root@localhost: CA$cp root_ca.crcd /client/
root@localhost: client$
root@localhost: client$ll
итого 12
-rwxrwx---. 1 root vboxsf 1074 июн  4 02:12 client.crt
-rwxrwx---. 1 root vboxsf 1675 июн  4 02:12 client.key
-rwxrwx---. 1 root vboxsf 1277 июн  4 02:12 root_ca.crt
root@localhost: client$curl https://192.168.0.9:7777/
curl: (35) NSS: client certificate not found (nickname not specified)
root@localhost: client$curl --cert ./client.crt --key client.key --cacert root_ca.crt https://192.168.0.9:7777/
Hello, world!root@localhost: client$
```

Это значит, что мы установили mTLS соединение с проверкой сертификата как сервера, так и клиента.

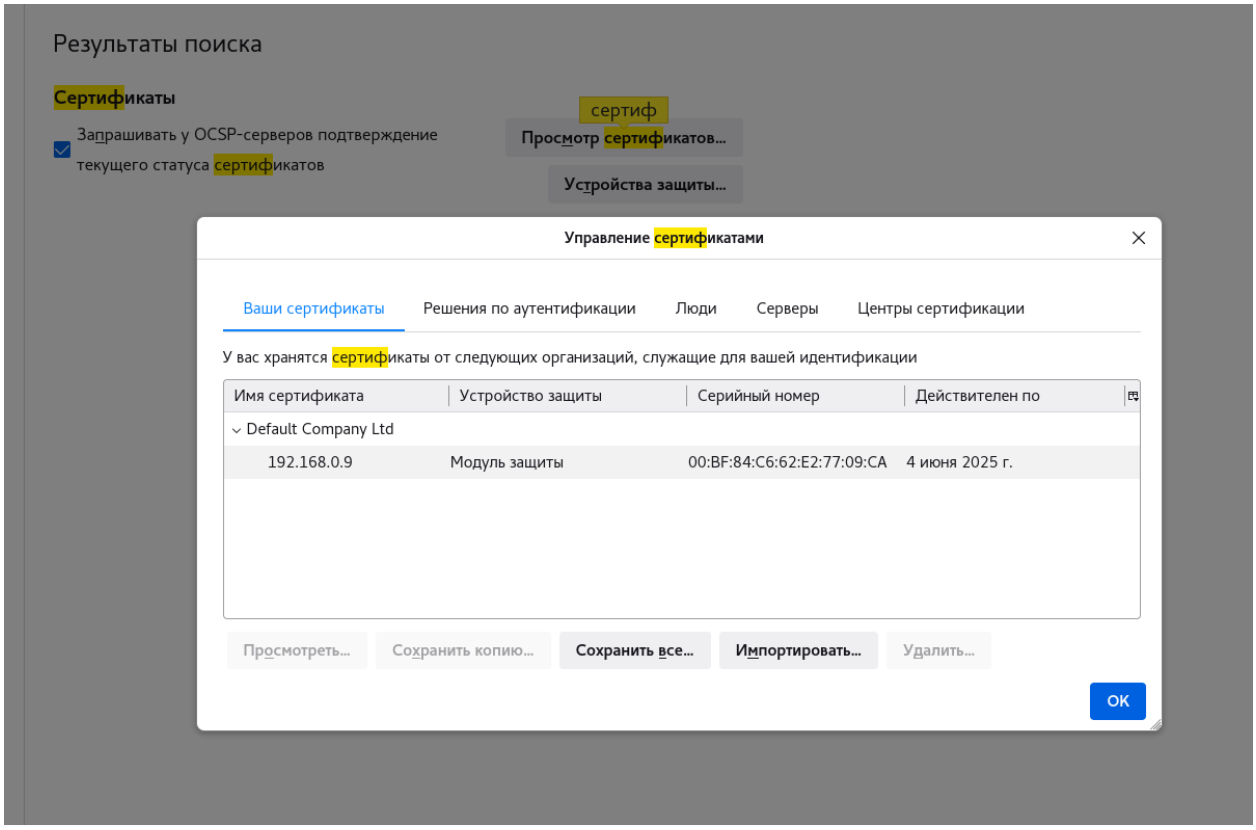
Реализация mTLS в браузере. Чтобы открыть сайт в браузере, нужно:

Клиентский сертификат и приватный ключ надо сконвертировать в формат p12.

```
openssl pkcs12 -inkey client.key -in client.crt -export -out client.p12
```

```
Hello, world!root@localhost: client$openssl pkcs12 -inkey client.key -in client.crt -export -out client.p12
Enter Export Password:
Verifying - Enter Export Password:
root@localhost: client$ll
итого 16
-rwxrwx---. 1 root vboxsf 1074 июн  4 02:12 client.crt
-rwxrwx---. 1 root vboxsf 1675 июн  4 02:12 client.key
-rwxrwx---. 1 root vboxsf 2357 июн  4 02:15 client.p12
-rwxrwx---. 1 root vboxsf 1277 июн  4 02:12 root_ca.crt
root@localhost: client$
```

Импортируем полученный сертификат в браузер:



Сайт доступен по httpsбез предупреждений, с действительным сертификатом: