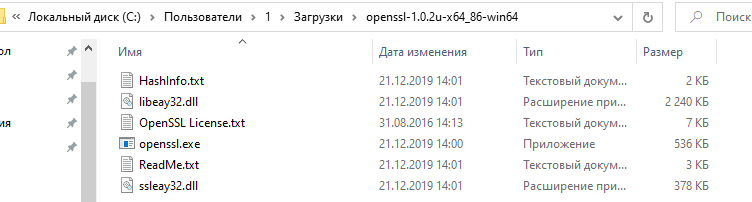
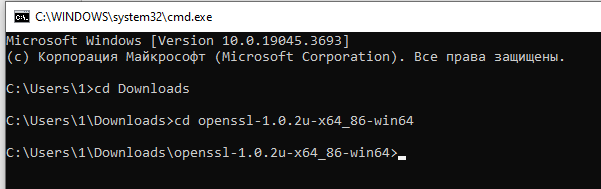
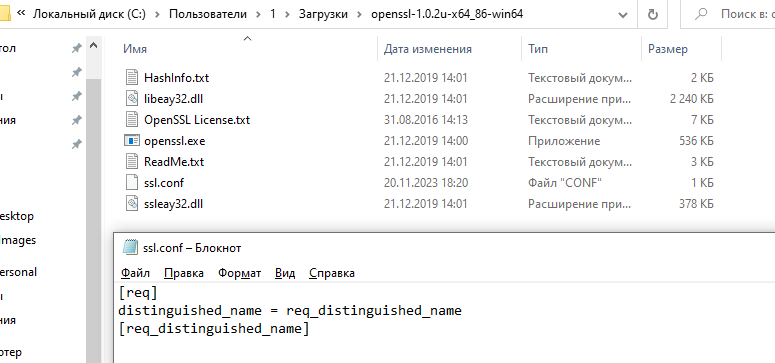
**Лабораторная работа №7**

**Получение самоподписанного сертификата и защита соединения с веб-сервером**

1. Распакуйте архив openssl-1.0.2u-x64\_86-win64.zip, который приложен к лабораторной

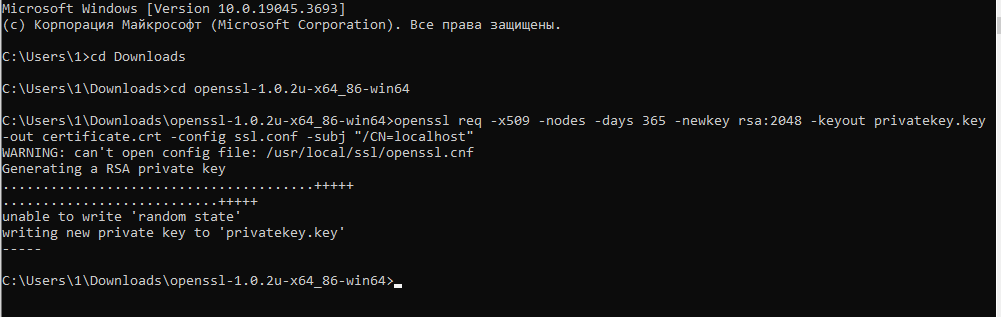


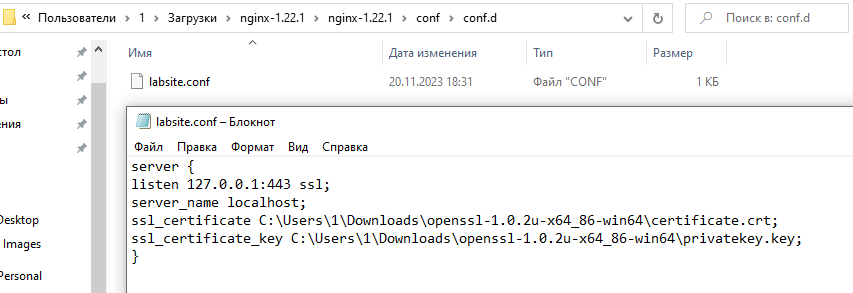
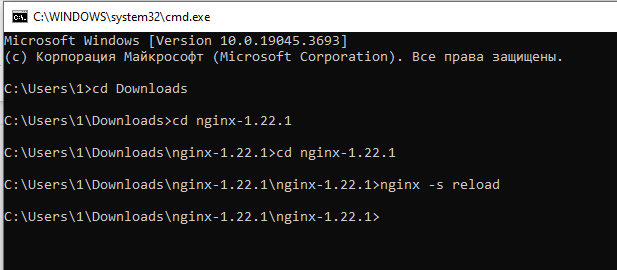
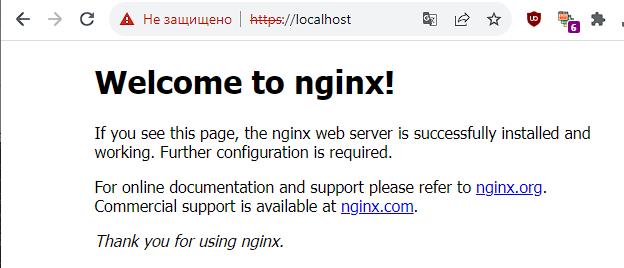
1. Запустите терминал (cmd.exe), и перейдите в каталог, в котором лежит распакованный файл openssl.exe
2. Создайте в этом же каталоге файл с именем ssl.conf

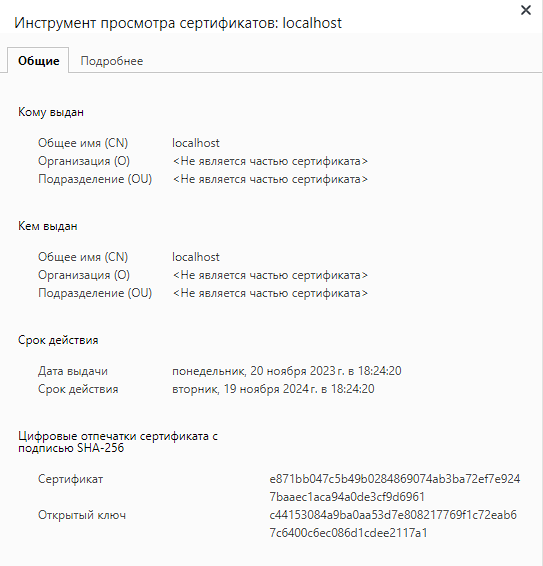
=

1. Сгенерируйте сертификат и приватный ключ, выполнив в терминале команду

*openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout privatekey.key -out certificate.crt -config ssl.conf -subj "/CN=localhost"*

**

1. В конфигурационном файле nginx модифицируйте директиву server следующим образом: 
2. Перейдите в терминале в папку nginx и выполните команду nginx -s reload для обновления конфигурации nginx 
3. Откройте в веб-браузере страницу https://localhost/. Если вы всё сделали правильно, то вы увидите страницу приветствия nginx, но сначала браузер может предупредить вас о том, что сертификат небезопасен. 
4. Нажмите на пиктограмму замочка и откройте диалог информации о сертификате.



**Контрольные вопросы**

1. Как называются файл, содержащий сертификат?

*certificate.crt*

1. А файл, содержащий приватный ключ?

*privatekey.key*

1. Почему nginx нужно знать и сертификат, и приватный ключ?

Основной принцип работы таких алгоритмов заключается в следующем: сервер генерирует у себя пару ключей, один позволяет только зашифровывать сообщения, а второй — только расшифровывать. Первый ключ сервер открыто (откуда и название) сообщает всем желающим (это сертификат), а второй хранит только у себя. Используя открытый ключ, любой клиент может послать зашифрованное сообщение серверу, которое тот расшифрует своим закрытым ключем.

1. Как вы поняли, что ваш веб-сервер работает по протоколу https?

По URL / по наличию сертификата / замочка

**Контрольные вопросы**

1. Какие, по вашему мнению, есть недостатки у HTTPS по сравнению с HTTP?

HTTPS сервер тратит больше ресурсов на обработку запроса из-за шифрования.

Кроме того есть ключевая проблема, которая связана с тем, что для проверки сертификата используется цепочка подписей. Доверенные центры (центры сертификаций) бывают корневыми и промежуточными. Сертификат безопасности считается подлинным, если он подписан корневым центром. Обычно от корневого центра до браузера пролегает длинная цепочка промежуточных центров и их подписей.

Где-то в этой цепочке может встроиться злоумышленник и получить доступ к сертификату, а значит, ко всем данным HTTP-соединения.

1. Существует атака на протокол TLS, которая заключается в том, что вы (злоумышленник) “подсовываете” свой TLS сертификат клиенту в тот момент, когда он ожидает его от сервера. Как называется такая атака? Как с ней бороться?

?

1. Зачем в TLS используется симметричное шифрование? Почему нельзя использовать ассиметричное шифрование для всех сообщений?

Симметричное шифрование — это когда используется один и тот же ключ для шифрования и дешифрования данных. Оно работает эффективно и быстро, но требует предварительного обмена ключом между компьютером и сервером, в ходе которого ключ могут перехватить.

Асимметричное шифрование использует два ключа: публичный для шифрования и приватный для дешифровки. Публичный ключ можно свободно распространять, а приватный должен быть хорошо защищён. Асимметричное шифрование безопаснее, но требует больше вычислительных ресурсов и работает медленнее, чем симметричное.

В протоколе TLS симметричное шифрование используют для шифрования непосредственно сообщений, а асимметричное шифрование — во время рукопожатия, то есть в начале сессии для обмена ключами и аутентификации.