94: Развертывание PKI

Информационное обеспечение:

Общие сведения о PKI: (предыдущее занятие)

Что такое PKI? Главное об инфраструктуре открытых ключей. – https://habr.com/ru/post/655135/

Инфраструктура открытых ключей. – http://security.demos.ru/crypto/pki/

Инфраструктура открытых ключей. – https://dic.academic.ru/dic.nsf/ruwiki/151605

Public Key Cryptography: осваиваем открытые ключи на практике. – https://xakep.ru/2016/03/11/pki/

Развертывание PKI:

Инфраструктуры открытых ключей: Учебный курс / ИНТУИТ. – https://intuit.ru/studies/courses/110/110/info

Руководство по OpenSSL: SSL-сертификаты, ключи и CSR. – https://wiki.merionet.ru/servernye-resheniya/45/rukovodstvo-po-openssl-ssl-sertifikaty-klyuchi-i-csr/

Как развернуть и настроить инфраструктуру открытого ключа (PKI) с корпоративным центом сертификации (CA). – https://blog.sedicomm.com/2020/05/19/kak-razvernut-i-nastroit-infrastrukturu-otkrytogo-klyucha-pki-i-korporativnym-tsentom-sertifikatsii-ca/?ysclid=lg6lubv637775191906

Развёртывание PKI в информационной системе на базе решений компании АО «Аладдин Р.Д.». – https://www.aladdin-rd.ru/upload/catalog/aladdin-eca/Deployment_PKI_Samba.pdf

Актуализация опорных знаний

Общие сведения об инфраструктуре открытых ключей (PKI)

Инфраструктура открытых ключей (ИОК, англ. ***PKI*** — *Public Key Infrastructure*) — это термин, подразумевающий **набор мер и политик**, позволяющих развертывать и управлять одной из наиболее распространенных форм онлайн-шифрования — шифрованием с открытым ключом.

Другое определение:

Public Key Infrastructure (PKI) - совокупность сервисов для управления ключами и цифровыми сертификатами пользователей, программ и систем.

Упрощенно,

PKI представляет собой систему, основным компонентом которой является удостоверяющий центр и пользователи, взаимодействующие между собой используя сертификаты, выданные этим удостоверяющим центром.

PKI использует технологию открытых ключей для:

- идентификации участников электронного обмена (пользователей, программ, систем),
- обеспечения конфиденциальности информации,
- контроля за целостностью информации,
- установления происхождения информации.

Основные принципы работы PKI

В основе PKI лежит использование криптографической системы с открытым ключом и несколько основных принципов:

- ❑ закрытый ключ известен только его владельцу;
- ❑ удостоверяющий центр создает сертификат открытого ключа, таким образом удостоверяя этот ключ;
- ❑ никто не доверяет друг другу, но все доверяют удостоверяющему центру;
- ❑ удостоверяющий центр подтверждает или опровергает принадлежность открытого ключа заданному лицу, которое владеет соответствующим закрытым ключом.

Задачей PKI является определение политики выпуска цифровых сертификатов, выдача их и аннулирование, хранение информации, необходимой для последующей проверки правильности сертификатов.

> Инфраструктура открытых ключей основывается на использовании принципов криптографической системы с открытым ключом.

Инфраструктура управления открытыми ключами состоит из центра сертификации (удостоверяющего центра — УЦ), конечных пользователей, и опциональных компонентов: центра регистрации и сетевого справочника.

> **PKI оперирует в работе сертификатами**. Сертификат — это электронный документ, который содержит электронный ключ пользователя, — открытый или же ключевую пару (keypair), — информацию о пользователе, которому принадлежит сертификат, удостоверяющую подпись центра выдачи сертификатов (УЦ) и информацию о сроке действия

Структура и механизмы работы

> PKI реализуется в модели клиент-сервер, то есть проверка какой-либо информации, предоставляемой инфраструктурой может происходить только по инициативе клиента.

Основные компоненты PKI:

- **Удостоверяющий центр** (УЦ):
- **Сертификат открытого ключа** (чаще всего просто *сертификат*) — это данные пользователя и его открытый ключ, скрепленные подписью удостоверяющего центра.
- **Регистрационный центр** (РЦ) — необязательный компонент системы, предназначенный для регистрации пользователей.
- **Репозиторий** — хранилище, содержащее сертификаты и списки отозванных сертификатов (COC) и служащее для распространения этих объектов среди пользователей. В Федеральном Законе РФ № 63 «Об электронной подписи» он называется *реестр сертификатов ключей подписей*.
- **Архив сертификатов**;
- **Центр запросов** — необязательный компонент системы, где конечные пользователи могут запросить или отозвать сертификат.
- **Конечные пользователи** — пользователи, приложения или системы, являющиеся владельцами сертификата и использующие инфраструктуру управления открытыми ключами.

> PKI выстраивается вокруг двух основных концепций – **ключи** и **сертификаты**.

Инфраструктура открытых ключей называется именно так, потому что каждая сторона в защищенном соединении имеет два ключа: открытый и закрытый.

Открытый ключ известен всем и используется во всей сети для кодирования данных, но доступ к данным невозможен без закрытого ключа, который используется для декодирования.

Проверка личности не менее важна при работе с PKI, и именно здесь в игру вступают сертификаты.

> Сертификаты PKI чаще всего рассматриваются как цифровые паспорта, содержащие множество присвоенных данных.

УЦ и его работа

> Основная работа удостоверяющего центра заключается в идентификации пользователей и их запросов на сертификаты, в выдаче пользователям сертификатов, в проверке подлинностей сертификатов

и в проверке по сертификату, не выдаёт ли пользователь сертификата себя за другого, в аннулировании или отзыве сертификатов, в ведении списка отозванных сертификатов.

Процесс работы с сертификатами

> Для того чтобы получить сертификат, нужно найти какой-либо УЦ в интернете (альтернативным решением является использование ПО PGP или ему подобных), после чего выписать сертификат и установить его себе в систему.

После установки сертификата его можно будет увидеть у себя в хранилище личных сертификатов. Для того чтобы просмотреть его свойства, достаточно просто открыть его. (Для операционных систем семейства Windows: Пуск -> Выполнить -> **`certmgr.msc`** -> OK). В свойствах можно увидеть время действия сертификата, кем он был выдан, кому был выдан, его уникальный номер и прочие свойства.

Стандарты и протоколы

> Информация, необходимая для работы PKI, содержится в стандарте **X.509**

- **IPSec (IP Security)**. Набор протоколов разрабатываемых Internet Engineering Task Force (IETF) для встраивания улучшенных средств безопасности в IP уровень.
- **LDAP (Lightweight Directory Access Protocol)**. Упрощенная реализация стандартов X.500, которая совместима с TCP/IP сетями.
- **PKIX (PKI for X.509 certificates)**. - это передовое средство совместимости PKI стандартов.

- **S/MIME (Secure Multipurpose Internet Mail Extensions)**. Определяет тип шифрования и/или цифровой подписи электронного сообщения, используя шифрование с открытым ключом.
- **SSL (Secure Sockets Layer)**. SSL и TLS (Transport Layer Security), который основан на SSL, самые важные протоколы для обеспечения безопасного доступа к web-серверам. SSL и TLS так же используются для обеспечения общей безопасности при обращении пользователя к серверу во множестве не-web-приложений. Оба используют PKI при получении сертификатов для пользователей и серверов.
- **VPN (Virtual Private Network)**.

Развертывание PKI

Процесс развертывания PKI состоит из нескольких этапов, каждый из которых должен сопровождаться соответствующим документированием и проверками:

- ❑ Предварительный этап
- ❑ Проектирование.
- ❑ Создание прототипа.
- ❑ Пилотный проект.
- ❑ Внедрение.

Каждый из этапов создания PKI дает результат в виде явно оформленного "продукта", позволяющего убедиться в законченности и общем продвижении процесса.

Предварительный этап

Предварительный этап включает:

- ❑ подготовительную работу для принятия решения о необходимости развертывания инфраструктуры,
- ❑ оценку материальных ресурсов и финансовых возможностей организации,
- ❑ определение цели развертывания и сферы применения PKI,
- ❑ выбор приоритетных сервисов безопасности,
- ❑ анализ данных и приложений PKI.

В процессе подготовки принятия решения специалисты организации, планирующей развернуть PKI, должны изучить возможности и риски инфраструктур открытых ключей, ознакомиться с предложениями и PKI-решениями различных поставщиков программных продуктов и услуг в этой области.

Многие компании берутся за развертывание крупномасштабных систем типа PKI, не имея необходимых ресурсов. Поскольку развертывание PKI требует значительных капиталовложений, для принятия решения необ-

ходима оценка материальных ресурсов и финансовых возможностей организации на настоящий момент и в ближайшие годы, экономического эффекта от использования новой технологии, начальных затрат и стоимости функционирования PKI.

Проведя тщательную оценку своих потребностей, некоторые организации вообще могут прийти к выводу, что инфраструктура открытых ключей им не нужна. Развертывание PKI целесообразно для крупных территориально распределенных организаций, где необходимо наладить контролируемую защиту документов и серверов при использовании разнообразных приложений.

Для решения менее масштабных задач пригоден другой инструментарий безопасности. Так, например, для поддержки виртуальных частных сетей имеются специальные программные средства, как правило, уже оснащенные надежными функциями аутентификации. Сервисы безопасности виртуальных частных сетей используются главным образом организациями, которым необходимо защитить внутреннюю сеть от внешнего доступа через Интернет. Программы персонального шифрования обеспечивают защиту документов и данных в локальных системах и удобны для небольших групп пользователей. Сервер сертификатов может решить проблемы несанкционированного доступа к web-контенту, особенно в интрасетях и на порталах.

> Организация выбирает **инсорсинг**, когда принимает решение о развертывании внутренней PKI на базе собственных ресурсов

(включая персонал, аппаратное обеспечение и т.д.) и (или) с привлечением внешних ресурсов для выполнения некоторых или всех операций PKI. Ключевым моментом является то, что **инсорсинговая PKI находится под контролем организации.**

> Выбирая аутсорсинг, организация разрешает внешней стороне поддерживать и выполнять некоторые (а возможно, и все) операции своей корпоративной PKI.

В этом случае эти **операции не контролируются самой организацией**.

Проектирование PKI

Ключевым аспектом развертывания PKI является выбор архитектуры и проектирование. PKI допускает гибкость проектирования независимо от выбранной технологии. Этап проектирования занимает длительное время, так как на этом этапе должна быть сформирована политика PKI и регламент, задана архитектура PKI, определены аппаратные и программные средства поддержки инфраструктуры, выбраны ее компоненты, сервисы, режимы работы, протоколы и базовые стандарты.

> Проектирование PKI невозможно без рассмотрения правовых аспектов

ее функционирования: ППС и регламента, ответственности, страхования и др.

Если организация решает самостоятельно сформировать правовую политику, то должна разработать следующие документы:

- политику применения сертификатов и регламент УЦ;
- политику аутентификации;
- политику конфиденциальности (в отношении сведений, предоставляемых пользователями в целях аутентификации).

OpenSSL и сертификаты X.509

Установка

OpenSSL - это широко используемый инструмент для работы с CSR-файлами и SSL-сертификатами, который можно загрузить с официального сайта OpenSSL.

Это инструмент реализации с открытым исходным кодом для SSL/TLS, который используется примерно на 65% всех активных интернет-серверов, что делает его неофициальным отраслевым стандартом.

Установка OpenSSL

Сначала проверим, установлена ли утилита OpenSSL при помощи команды:

Debian, Ubuntu: **`dpkg -l |grep openssl`**

Red Hat и CentOS: **`rpm -qa | grep -i openssl`**

Установка:

Debian, Ubuntu: **`apt-get install openssl`**

Red Hat и CentOS: **`yum install openssl openssl-devel`**

Создание закрытого ключа

Мы будем использовать папку **`~/certs`** для создания сертификатов. Вы можете создать эту папку и сразу перейти в неё с помощью команды:

```
mkdir ~/certs && cd ~/certs
```

В пакете OpenSSL есть две команды, которые выполняют очень похожее действие — генерируют пару приватный-публичный ключ RSA:

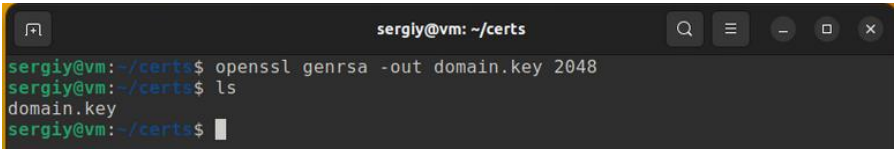
```
openssl genpkey -algorithm RSA
openssl genrsa
```

Форматы создаваемых этими программами RSA ключей немного различаются. Рекомендуется использовать **`genpkey`** с которой нужно указать алгоритм ключа опцией **`-algorithm`**.

Прежде всего необходимо создать закрытый ключ для сертификата. Будем использовать команду **genrsa**:

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

Ей необходимо передать имя файла ключа с помощью опции **-out**. Вы можете выбрать любое имя, оно не имеет значения. Также можно указать размер ключа, например **2048**. Если этого не сделать, то будет создан ключ размером 512 бит. Все ключи размером меньше 512 бит считаются небезопасными. После выполнения команды в текущей папке появится файл ключа:



Для того чтобы получить сертификат, который можно использовать нужно этот ключ подписать. А для этого надо создать запрос на подпись.

Создание запроса на подпись

Запрос на подпись сертификата (**CSR - Certificate Signing Request**) содержит наиболее важную информацию о вашей организации и домене. Это зашифрованный блок текста, который включает информацию о вашей организации, такую как страна, адрес электронной почты, полное доменное имя и так далее. Он отправляется в центр сертификации при подаче заявки на сертификат SSL.

SSL использует две длинные строки случайно сгенерированных чисел, которые известны как закрытые и открытые ключи. Открытый ключ доступен для публичного домена, так как он является частью вашего SSL-сертификата и сообщается вашему серверу.

CSR обычно содержит следующую информацию:

Параметр	Описание	Пример
Common Name (CN) или FQDN	FQDN (fully qualified domain name) - это полное доменное имя вашего сайта. Он должен совпадать с тем, что пользователи вводят в веб-браузере	wiki.merionet.ru
Organization Name (e.g., company)	Полное юридическое название вашей организации, включая суффиксы, такие как LLC, Corp и так далее	Merion Networks LTD
Organizational Unit Name	Отдел в вашей организации, который занимается этим сертификатом	Technology Division
Locality Name	Город, в котором находится ваша организация	Moscow
State/Region/Province	Штат или регион, в котором	Moscow

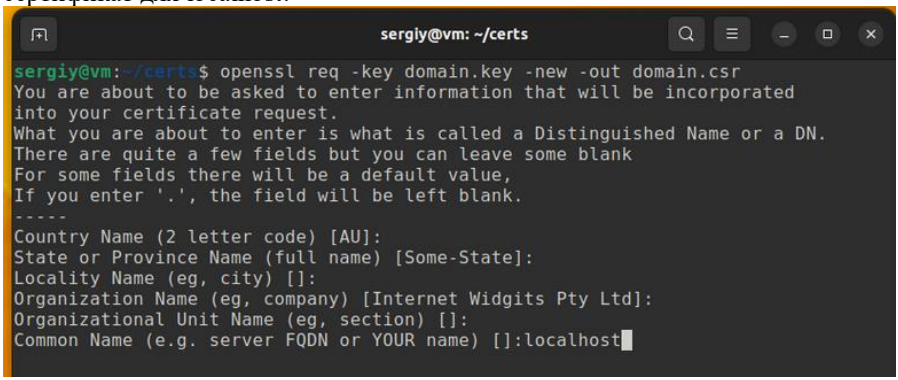
(full name)	находится ваша организация	
Country Code (2 letter code)	Страна, в которой находится ваша организация. Всегда вводится в виде двухбуквенного кода ISO	RU
Email Address	Адрес электронной почты, используемый для связи с веб-мастером сайта	info@merionet.ru
Public Key	втоматически созданный ключ, который создается с помощью CSR и входит в сертификат.	Закодированный текстовый блок похож на закрытый ключ. Смотрите пример закрытого ключа ниже.

Обратите внимание, что существуют определенные соглашения об именах, которые необходимо учитывать. Название организации и название организационной единицы не должны содержать следующие символы: < > ~! @ # $% ^ * / \ ()?., &

При создании запроса на подпись нужно указать необходимую информацию. Обязательное поле здесь только одно. Это CN. Здесь должно быть указанно ваше доменное имя, для которого вы собираетесь использовать сертификат, также можно указать дополнительную информацию о вашей компании, адресе и организации, но это уже необязательно. Для создания запроса на подпись для ранее созданного ключа используйте такую команду:

```
openssl req -key domain.key -new -out domain.csr
```

Во время создания запроса большинство параметров можно оставить по умолчанию. Обязательно заполнять только **Common Name (e.g. server FQDN or YOUR name).** Если вы хотите создать сертификат для сайта, то в этом поле нужно указать домен этого сайта. В этом примере будет создан сертификат для localhost:



Вы также можете создать закрытый ключ и запрос на подпись открытого ключа одной командой:

```
openssl req -newkey rsa:2048 -nodes -keyout domain.key \
-out domain.csr
```

Опция **-newkey** указывает, что нужно создать новую пару ключей, а в параметрах мы сообщаем тип rsa и сложность 2048 байт. Опция **-nodes** указывает, что шифровать ключ не нужно, опция **-new** указывает что нужно создать запрос csr.

Кроме того, можно создать csr запрос из уже существующего сертификата и закрытого ключа, тогда вам не придется вводить информацию, она будет получена из сертификата:

```
openssl x509 -in domain.crt -signkey domain.key -x509toreq \
-out domain.csr
```

Параметр **-x509toreq** указывает, что нужно использовать сертификат для X509 для получения CSR. X509, это сертификаты, подписанные сами собой. Обычно сертификат подписывается другим сертификатом, а этот был подписан сам собой. Если вы получили сертификат от CA, то этот параметр не нужен.

Подпись сертификатов OpenSSL

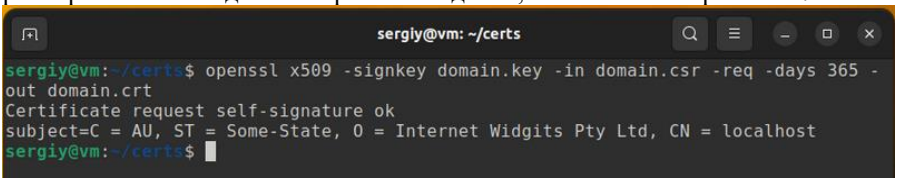
Допустим, у вас есть приватный ключ и запрос на подпись, фактически, открытый ключ. Теперь вам нужно его подписать чтобы получить сертификат, который можно использовать. Тут есть несколько вариантов. Можно отправить csr файл на подпись какому-либо центру сертификации, например, LetsEncrypt. Можно подписать сертификат тем же ключом, с помощью которого он был создан, и третий вариант - создать свой центр сертификации.

Подпись сертификата самим собой

Мы подпишем наш сертификат сами, ключом, на основе которого он был создан:

```
openssl x509 -signkey domain.key -in domain.csr -req \
-days 365 -out domain.crt
```

С помощью параметра **-days** мы указываем что сертификат будет действительным в течение 365 дней, то есть в течение года. Обратите внимание, что во время подписи проверяется CN, поэтому если вы не зададите этот параметр на этапе создания запроса на подпись, то ничего не заработает.



Вы можете объединить все в одну команду и сразу создать закрытый ключ, csr и подписанный сертификат:

```
openssl req -newkey rsa:2048 -nodes -keyout domain.key -x509
-days 365 -out domain.crt
```

Или создать самоподписанный сертификат openssl из существующего закрытого ключа без csr:

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

Опция **-new** говорит, что нужно запросить информацию о csr у пользователя. Чтобы браузер доверял ключу нужно этот же сертификат импортировать в список доверенных.

Подпись сертификатов с помощью OpenSSL

А теперь рассмотрим третий способ выполнить создание сертификата OpenSSL - подписать его с помощью собственного CA, центра сертификации.

Вот вы сейчас думаете что это что-то такое сложное, да? А нет, это обычная папка, в которой лежит защищенный паролем закрытый ключ, с помощью которого мы будем подписывать все другие ключи. А открытая часть этого ключа должна быть добавлена во все браузеры, которые будут ему доверять.

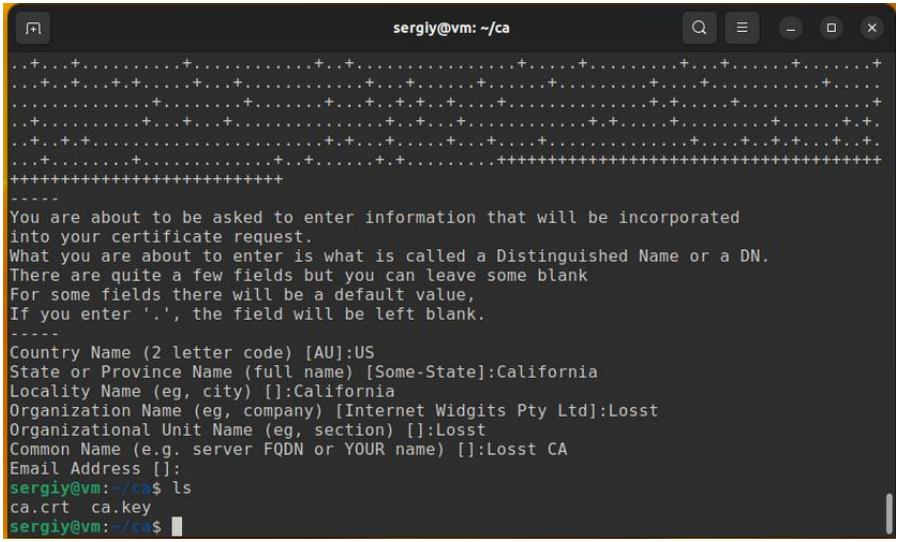
Вообще, центр сертификации в крупных корпорациях находится на отдельных компьютерах, которые даже к сети не подключены. Но для примера мы разместим папку в нашей домашней папке:

```
mkdir ~/ca && cd ~/ca
```

Дальше нужно создать самоподписанный сертификат openssl для нашего CA.

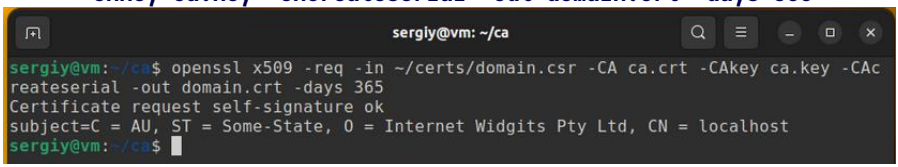
```
openssl req -newkey rsa:2048 -nodes -keyout ca.key -x509 \
-days 3654 -out ca.crt
```

С помощью параметра **-days** мы устанавливаем долгий строк действия - десять лет. Программа запросит стандартные данные, которые используются при создании сертификатов. Эти данные будут выводится в браузере при просмотре информации о центре сертификации если вы будете использовать подписанные сертификаты для HTTPS. В поле **Common Name** можно указать имя вашей организации:



Осталось подписать наш сертификат, созданный ранее:

```
openssl x509 -req -in ~/certs/domain.csr -CA ca.crt \
-CAkey ca.key -CAcreateserial -out domain.crt -days 365
```



Готово, теперь наш сертификат подписан. Но теперь, чтобы браузеры ему доверяли нужно добавить сертификат CA в список доверенных сертификатов браузера.

Подпись сертификатов с помощью EasyRSA

Создание центра сертификации можно немного упростить. Существует набор скриптов под названием EasyRSA, который позволяет инициализировать центр сертификации, а также подписывать и отзывать сертификаты. Сначала установите пакет easy-rsa из официальных репозиториев:

```
sudo apt install easy-rsa
```

Далее нужно создать папку в которой будут находится файлы центра сертификации. Например, в домашней папке:

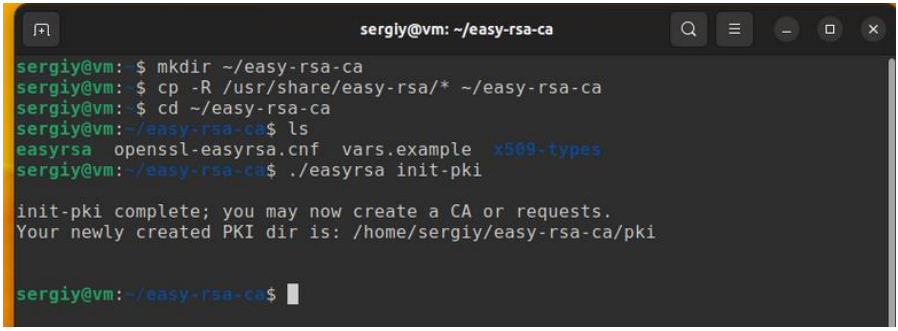
```
mkdir ~/easy-rsa-ca
```

После этого скопируйте файлы из /usr/share/easy-rsa/ в эту папку:

```
cp -R /usr/share/easy-rsa/* ~/easy-rsa-ca
```

Дальше перейдите в неё и инициализируйте центр сертификации:

```
cd ~/easy-rsa-ca
./easyrsa init-pki
```



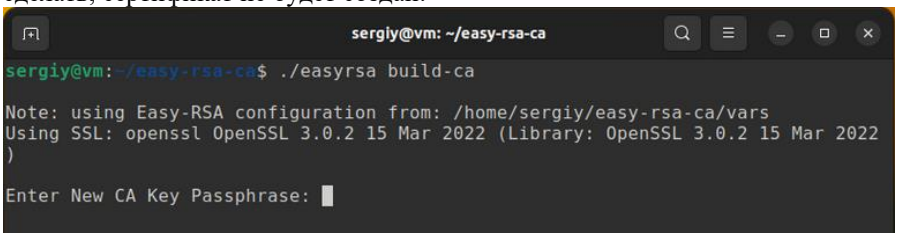
Прежде чем вы сможете создать сертификат, необходимо создать файл **vars** и в нём прописать информацию о центре сертификации. Например:

```
vi ./vars
set_var EASYRSA_REQ_COUNTRY "US"
set_var EASYRSA_REQ_PROVINCE "California"
set_var EASYRSA_REQ_CITY "California"
set_var EASYRSA_REQ_ORG "Losst"
set_var EASYRSA_REQ_EMAIL "support@losst.pro"
set_var EASYRSA_REQ_OU "Losst"
set_var EASYRSA_ALGO "rsa"
set_var EASYRSA_DIGEST "sha512"
```

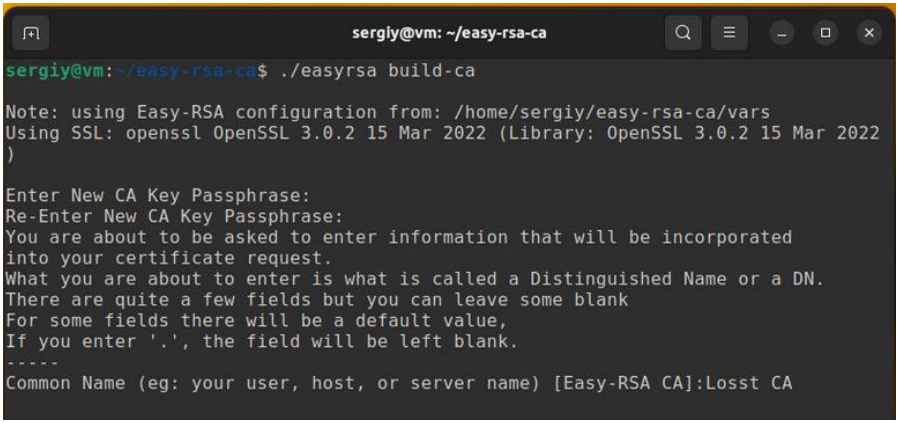
Первые 7 параметров аналогичны тем, что заполняются при создании запроса на подписание сертификата. Параметр **EASYRSA_ALGO** позволяет настроить алгоритм шифрования, можно использовать стандартный RSA или алгоритм на основе эллиптических кривых ES. Последний параметр - это формат подписи сертификатов. После этого можно создать сертификат CA:

```
./easyrsa build-ca
```

Для корневого сертификата необходимо задать пароль. Если этого не сделать, сертификат не будет создан:



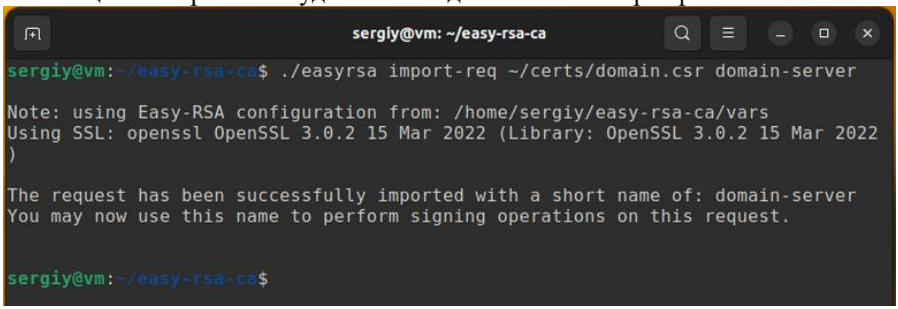
А в Common Name можно указать название центра сертификации, которое будет отображаться в браузере в информации о том кем подписан сертификат:



После этого можно переходить к подписи CSR. Например, давайте подпишем **domain.csr**, который был создан ранее. Сначала этот запрос надо импортировать. Для этого используйте такую команду:

```
./easyrsa import-req ~/certs/domain-server.csr domain-server
```

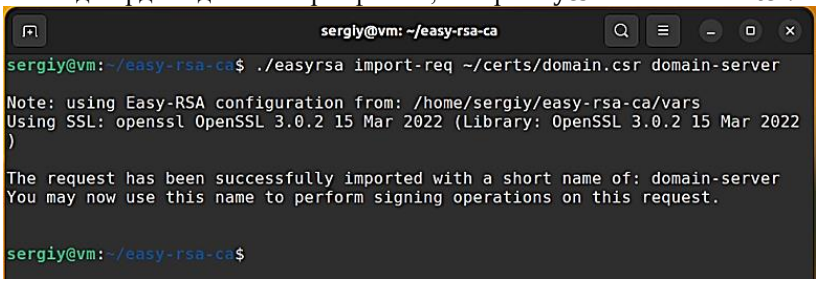
В первом параметре нужно указать путь к фалу CSR, а во втором имя, с помощью которого вы будете взаимодействовать с сертификатом.



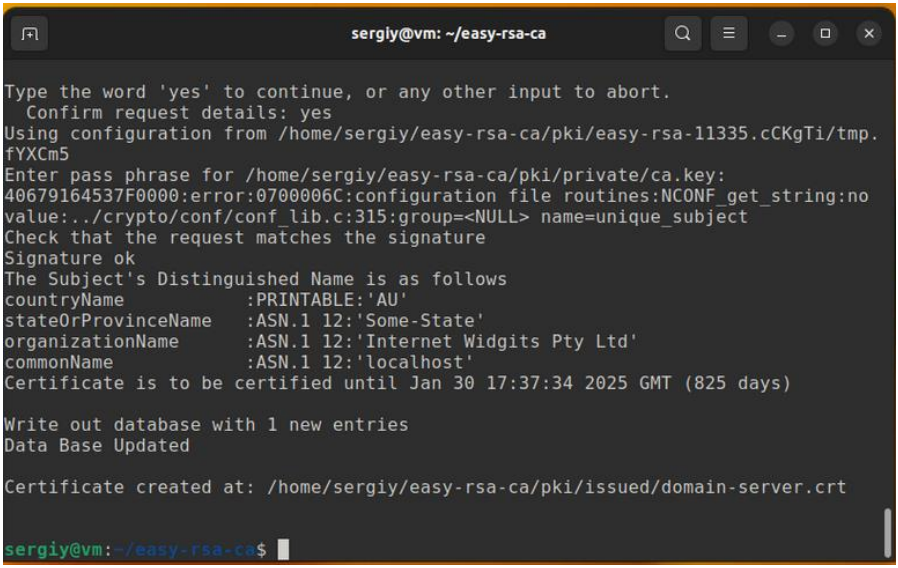
После этого вы можете подписать этот запрос, используя имя, заданное при импорте:

```
./easyrsa sign-req server domain-server
```

Команде необходимо передать два параметра тип сертификата и имя запроса. Тип сертификата может быть **server** или **client**. Программа попросит подтвердить данные сертификата, наберите **yes** и нажмите **Enter**:



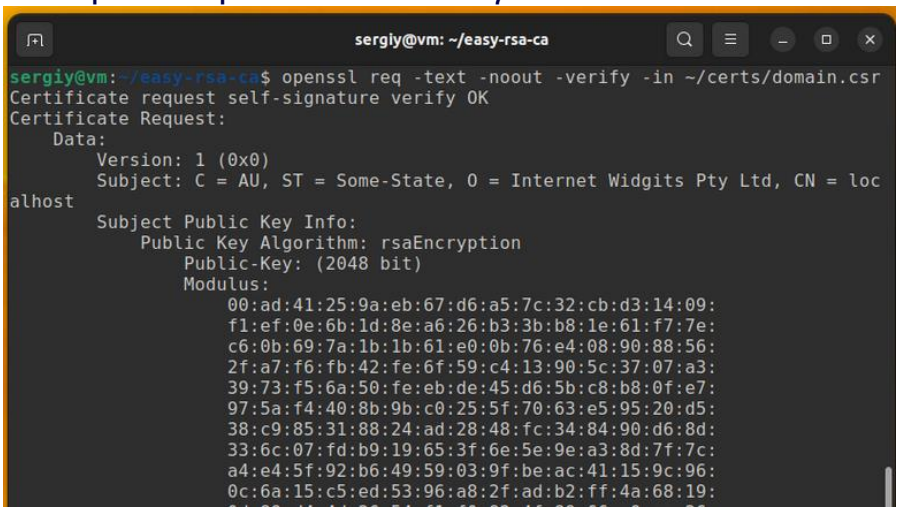
А затем нужно будет ввести пароль от корневого сертификата. После этого будет создан новый сертификат в папке **pki/issued** с расширением **.crt**.



Просмотр сертификатов

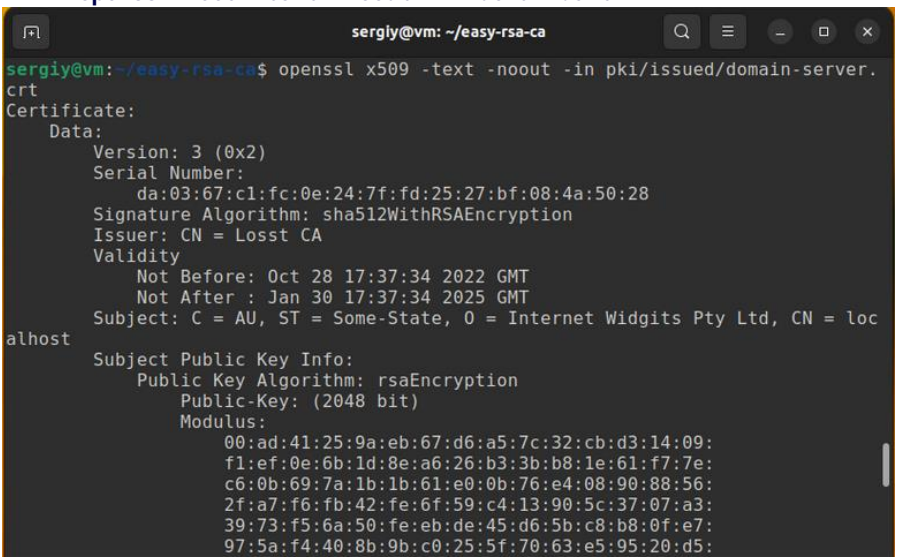
Сертификаты сохраняются в формате PEM, а это значит, что вы не сможете их открыть как текстовый файл и нужно использовать специальные команды для просмотра информации о них. Сначала смотрим содержимое CSR:

```
openssl req -text -noout -verify -in domain.csr
```



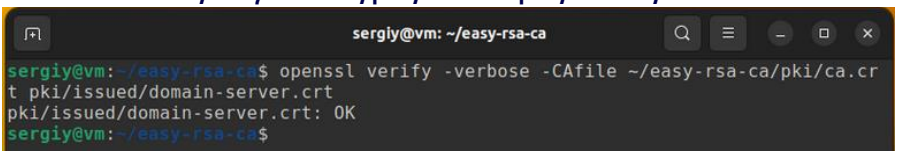
Смотрим содержимое сертификата в режиме обычного текста:

openssl x509 -text -noout -in domain.crt



Проверяем действительно ли сертификат подписан нужным CA:

**openssl verify -verbose **
-CAfile ~/easy-rsa-ca/pki/ca.crt pki/issued/domain-server.crt

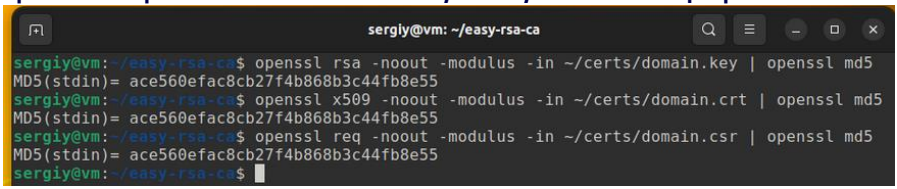


Просмотр закрытого ключа:

openssl rsa -check -in domain.key

Чтобы проверить связаны ли между собой закрытый ключ, сертификат и открытый ключ (запрос на подпись) можно подсчитать сумы md5 для этих ключей, если значения совпадут, то есть вероятность что это ключи из одной пары:

```
openssl rsa -noout -modulus -in ~/certs/domain.key | openssl md5
openssl x509 -noout -modulus -in ~/certs/domain.crt | openssl md5
openssl req -noout -modulus -in ~/certs/domain.csr | openssl md5
```



Как использовать сертификат

Не зависимо от того какой способ создания сертификата OpenSSL вы выбрали у вас будет два файла. Это файл закрытого ключа с расширением **.key** и файл сертификата с расширение **.crt**. В данном примере это:

- `domain.key`
- `domain-server.crt`

Также вам может понадобится сертификат fullchain. Обычно этот файл содержит сам подписанный сертификат и сертификаты промежуточных центров сертификации. Но в случае если вы подписываете сертификат своим центром сертификации, промежуточных сертификатов у вас нет, а значит в качестве fullchain можно использовать сам подписанный сертификат.

Если вы хотите подключить ваши сертификаты к Apache, используйте такие директивы в файле виртуального хоста:

```
vi /etc/apache2/sites-enabled/000-default.conf
SSLEngine on
SSLCertificateFile /путь/к/сертификату/domain-server.crt
SSLCertificateKeyFile /путь/к/файлу/domain.key
SSLCertificateChainFile /путь/к/файлу/domain-server.crt
```

Для Nginx директивы подключения сертификатов выглядят похожим образом:

```
ssl_certificate /путь/к/файлу/domain-server.crt
ssl_certificate_key /путь/к/файлу/domain.key
```

Их нужно добавить внутри блока `server` для сайта на котором вы хотите настроить HTTPS.

****

Дополнительные материалы

Где получить сертификат SSL

Сертификаты SSL проверяются и выдаются **Центром сертификации (CA - Certificate Authorities)**. Вы подаете заявку, генерируя **CSR (Certificate Signing Request - запрос на получение сертификата)** с парой ключей на вашем сервере, которая в идеале будет содержать сертификат SSL. CSR содержит важные организационные детали, которые CA проверяет.

1. Сгенерируйте CSR и пару ключей локально на вашем сервере. Пара ключей состоит из открытого (**public key**) и закрытого (**private key**) ключей.

2. Отправьте CSR и открытый ключ в центр сертификации, который проверит вашу личность, а также владеете ли вы доменом, указанным в заявке. Центр сертификации проверяет вашу организацию и проверяет, зарегистрирована ли организация в расположении, указанном в CSR, и существует ли домен.

3. После проверки организация получает копию своего сертификата SSL, включающего бизнес данные, а также открытый ключ. Теперь организация может установить сертификат на своем сервере.

4. Когда центр сертификации выдает сертификат, он связывается с сертификатом «доверенного корня» (trusted root) центра сертификации. Корневые сертификаты встроены в каждый браузер и связаны с индивидуально выданными сертификатами для установления HTTPS-соединения.

Нередко популярные браузеры не доверяют всем сертификатам, выпущенным одним центром сертификации. Например, Google Chrome не доверяет корневым сертификатам Symantec, поскольку Symantec несколько раз нарушала отраслевые политики. Это означает, что все сертификаты, укоренившиеся в Symantec, стали недействительными независимо от даты их действия.

Как создать CSR

Запросы на подпись сертификата (CSR) создаются с помощью пары ключей - открытого и закрытого ключа. Только открытый ключ отправляется в центр сертификации и включается в сертификат SSL, и он работает вместе с вашим личным ключом для шифрования соединения. Любой может иметь доступ к вашему открытому ключу, и он проверяет подлинность SSL-сертификата.

Закрытый ключ - это блок закодированного текста, который вместе с сертификатом проверяет безопасное соединение между двумя компьютерами. Он не должен быть общедоступным, и его не следует отправлять в ЦС.

Целостность сертификата зависит от того, что только вы знаете закрытый ключ. Если вы когда-либо скомпрометированы или утеряны, как можно скорее введите новый сертификат с новым закрытым ключом. Большинство ЦС не взимают плату за эту услугу.

Большинство пар ключей состоят из 2048 битов. Хотя пары ключей длиной 4096 бит более безопасны, они замедляют SSL-рукопожатия и создают нагрузку на серверные процессоры. Из-за этого большинство сайтов по-прежнему используют 2048-битные пары ключей.

Вариант 1: создать CSR

Первое, что нужно сделать, - это создать 2048-битную пару ключей RSA локально. Эта пара будет содержать как ваш закрытый, так и открытый ключ. Вы можете использовать инструмент Java key или другой инструмент, но мы будем работать с **OpenSSL**.

Чтобы создать открытый и закрытый ключ с запросом на подпись сертификата (CSR), выполните следующую команду OpenSSL:

openssl req –out certificatesigningrequest.csr -new -newkey rsa:2048 -nodes -keyout privatekey.key

Что эта команда означает:

- openssl - активирует программное обеспечение OpenSSL
- req - указывает, что мы хотим CSR
- –out - указывает имя файла, в котором будет сохранен ваш CSR. У нас в примере это certificatesigningrequest.csr
- –new –newkey - создать новый ключ
- rsa:2048 - сгенерировать 2048-битный математический ключ RSA
- –nodes - нет DES, то есть не шифровать закрытый ключ в PKCS#12 файл
- –keyout - указывает домен, для которого вы генерируете ключ

Далее ваша система должна запустить текстовую анкету для заполнения, которую мы описывали в таблице выше:

Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:

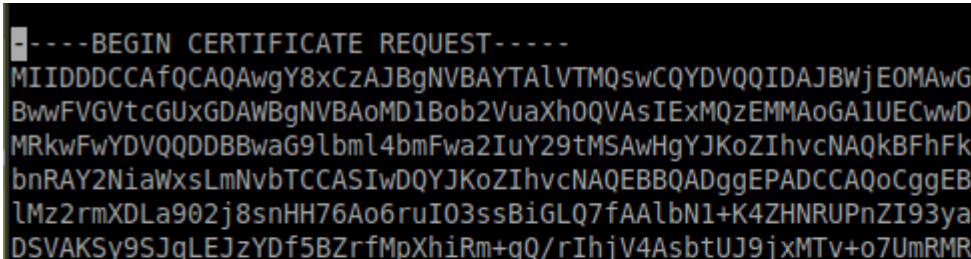
После завершения работы программы вы сможете найти файл CSR в вашем рабочем каталоге. Запрос на подпись сертификата, сгенерированный с помощью OpenSSL, всегда будет иметь формат файла .csr. Чтобы найти в папке все файлы этого формата используйте команду

ls *.csr

Тут будет список всех сертификатов, останется только найти тот, что мы только что сгенерировали.

Также вы можете открыть файл .csr в текстовом редакторе, например nano, чтобы просмотреть сгенерированный буквенно-цифровой код.

sudo nano your_domain.csr



После того, как вы сгенерировали CSR с парой ключей, сложно увидеть, какую информацию она содержит, поскольку она не будет в удобочитаемом формате. Вы можете легко декодировать CSR на своем сервере, используя следующую команду OpenSSL:

openssl req -in server.csr -noout -text

Далее можно декодировать CSR и убедиться, что он содержит правильную информацию о вашей организации, прежде чем он будет отправлен в центр сертификации. В Интернете существует множество CSR-декодеров, которые могут помочь вам сделать то же самое, просто скопировав содержимое файла CSR, например sslshopper.

Вариант 2. Создание CSR для существующего закрытого ключа

Рекомендуется выдавать новый закрытый ключ всякий раз, когда вы генерируете CSR. Если по какой-либо причине вам необходимо сгенерировать запрос на подпись сертификата для существующего закрытого ключа, используйте следующую команду OpenSSL:

openssl req -out CSR.csr -key privateKey.key -new

Вариант 3. Создание CSR для существующего сертификата и закрытого ключа

openssl x509 -x509toreq -in certificate.crt -out CSR.csr -signkey privateKey.key

Один маловероятный сценарий, в котором это может пригодиться, - это если вам нужно обновить существующий сертификат, но ни у вас, ни у

вашего центра сертификации нет первоначального CSR. Это позволит извлечь информацию о вашем домене и организации из сертификата SSL и использовать его для создания нового CSR, что позволит вам сэкономить время. Параметр -x509toreq преобразует сертификат в запрос сертификата.

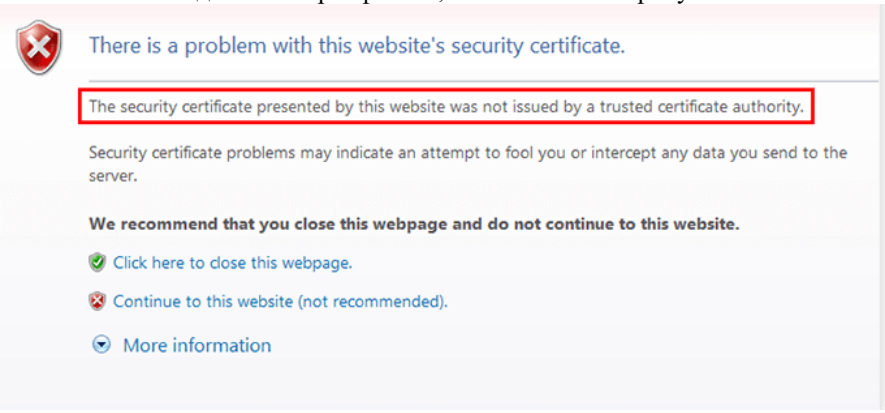
Вариант 4: Генерация самоподписанного(self-signed) сертификата

Самозаверяющий сертификат обычно используется для сред тестирования и разработки, а также в интрасети. Давайте создадим самозаверяющий сертификат, используя следующую команду OpenSSL:

openssl req -newkey rsa:2048 -nodes -keyout domain.key-x509 -days 365 -out domain.crt

Параметр –days установлен на 365, что означает, что сертификат действителен в течение следующих 365 дней. Параметр x509 указывает, что это будет самозаверяющий сертификат. Временный CSR генерируется, и он используется только для сбора необходимой информации. Если вы не хотите защищать свой закрытый ключ паролем, вы можете добавить параметр –nodes.

Центры сертификации не проверяют самоподписанные сертификаты. Таким образом, они не так безопасны, как проверенные сертификаты. Если ЦС не подписал сертификат, каждый основной браузер отобразит сообщение об ошибке «Ненадежный сертификат», как показано на рисунке ниже.



Вариант 5: Генерация самоподписанного сертификата из существующего закрытого ключа и CSR

Если у вас уже есть CSR и закрытый ключ и вам нужно создать самозаверяющий сертификат, используйте следующую команду:

openssl x509 -signkey domain.key -in domain.csr -req -days 365 -out domain.crt

Параметр –days установлен на 365, что означает, что сертификат действителен в течение следующих 365 дней.

Как скопировать содержимое файла CSR

Откройте каталог, в котором находится ваш CSR-файл. Введите следующую команду:

sudo cat domain.csr

Замените domain параметром FQDN вашего CSR. Эта команда отобразит содержимое файла CSR. Скопируйте весь контент, начиная с BEGIN CERTIFICATE REQUEST и заканчивая END CERTIFICATE REQUEST.

Продление сертификата - не используйте старые CSR повторно

То, что некоторые веб-серверы позволяют использовать старые CSR для обновления сертификатов, не означает, что вы должны их использовать. В качестве меры безопасности всегда генерируйте новый CSR и закрытый ключ при обновлении сертификата. Привязка к одному и тому же секретному ключу - это дорога, вымощенная уязвимостями безопасности.

Также рекомендуется обновить SSL-сертификат до истечения срока его действия. В противном случае потребуется покупать новый сертификат.

****

Внедрение инфраструктуры управления открытыми ключами с учетом снижения затрат и сроков внедрения осуществляется в течение семи этапов.

- Этап 1. Анализ требований к системе.
- Этап 2. Определение архитектуры.
- Этап 3. Определение регламента.
- Этап 4. Обзор системы безопасности. Анализ и минимизация рисков.
- Этап 5. Интеграция.
- Этап 6. Развертывание.
- Этап 7. Эксплуатация.

****