VK WorkSpace

Самоподписанные SSL-сертификаты

Пример последовательности действий для выпуска с помощью OpenSSL

Портал с документацией: https://biz.mail.ru/docs/on-premises/

Оглавление

Назначение документа

Документ описывает процедуру создания самоподписанных сертификатов с помощью OpenSSL.

1. Создание CA (корневой сертификат)

CA – это корневой сертификат, по сути третья сторона, которой при шифровании данных доверяют клиент и сервер. На его основе будут выпускаться все остальные сертификаты. Для каждого нового домена верхнего уровня необходимо будет выпускать свой сертификат.

Чтобы сгенерировать любой x509 сертификат, необходима пара ключей, открытый и закрытый. В рассматриваемом примере будем использовать RSA ключ.

1. Создаем ключ:

   ```
   openssl genrsa -out ca-key.pem 2048
   ```

   В каталоге появится файл **ca-key.pem**. Это – приватный ключ, он же закрытый ключ.

2. Генерируем сам CA-сертификат:

   ```
   openssl req -x509 -new -key ca-key.pem -days 10000 -out ca.pem
   ```

   В процессе генерации будут заданы вопросы, ответить на которые можно, например, так:

   ```
   Country Name (2 letter code) []:RU
   State or Province Name (full name) []:Moscow
   Locality Name (eg, city) []:Moscow
   Organization Name (eg, company) []:MyOwnCompany
   Organizational Unit Name (eg, section) []:DevOps
   Common Name (eg, fully qualified host name) []:
   Email Address []:admin@myowncompany.com
   ```

   На выходе будет получен файл **ca.pem**. Это – наш корневой CA, которым мы будем подписывать конечные сертификаты.

2. Генерация конечных сертификатов

1. Создаем ключ для конечного сертификата:

   ```
   openssl genrsa -out cert-key.pem 2048
   ```

2. Создаем файл **tls.cnf** следующего вида:

```
[req]
distinguished_name = req_distinguished_name
req_extensions = req_ext
prompt = no
[req_distinguished_name]
C = RU
ST = Moscow
L = Moscow
O = MailRU LLC
OU = DevOps
CN = www.domain.tld

[req_ext]
keyUsage = keyEncipherment, dataEncipherment, digitalSignature
extendedKeyUsage = serverAuth
subjectAltName = @alt_names

[alt_names]
DNS.1 = www.domain.tld
DNS.2 = cloud.domain.tld
DNS.3 = '*.cloud.domain.tld'
IP.1 = 127.0.0.1
IP.2 = 192.168.1.4
```

В этом файле указываем интересующие нас домены, ip-адреса и информацию о сертификате.

3. Генерируем запрос на сертификат:

```
openssl req -new -sha256 -out cert.csr -key cert-key.pem -config tls.cnf
```

В результате появится файл **cert.csr**, который можно проверить:

```
openssl req -text -noout -in cert.csr | grep 'DNS'
```

4. Генерируем сам сертификат:

```
openssl x509 -req -in cert.csr -CA ca.pem -CAkey \
  ca-key.pem -CAcreateserial -out cert.pem \
  -days 1000 -extensions req_ext -extfile tls.cnf
```

5. Проверяем что в сертификате есть все необходимые данные:

```
openssl x509 -in cert.pem -text | grep -E '(Issuer:|Subject:|DNS:)'
```

Должена быть выведена следующая информация:

- Issuer - данные с CA.
- Subject - данные из конфига tls.cnf.
- DNS собственно - alternative names.

3. Итоговый результат

После выполнения описанных выше действий должны получить следующий набор файлов:

- **ca.pem** - ставим на клиентские машины, в систему, в браузер.
- **cakey.pem** - храним в секрете.
- **cert.pem** и **cert-key.pem** - должны находиться на ресурсе, который хотим перевести на HTTPS.

Автор: Груздев Никита

11 декабря 2024 г.