Tree: b9c339f509 myBusiness / 使用OpenSSL生成含有Subject Alternative Name(SAN)的证书.md

Find file Copy path

0079123 Add files via upload b9c339f on Sep 7, 2018

1 contributor

271 lines (219 sloc) 11 KB

Raw Blame History

背景

SSL 在浏览器兼容方面，主要是新版本对域名的认证放弃了CommonName，因此本文介绍的方法全部都使用 Subject Alternative Name (SAN)， 从而支持目前的主流浏览器。

生成测试环境证书的两种方法

简单来说，生成测试证书大概有两种方式。 一种是生成一个CA根证书和多个服务需要的证书，然后使用CA根证书去签名多个证书。这种方式可以一次管理多个证书，也比较贴近真实情况，当然也可以用来做证书过期、更新等试验，缺点是操作起来略微麻烦一点。 还有一种方式就是，直接用根证书作为网站的HTTPS证书使用，这样只需要建立一个证书即可，比较适合小范围的测试，操作快捷。当然，上一种方式的优点就是这种方式的缺点。

第一种方式：首先建立**CA**，然后给自己签发**HTTPS**证书

Setp 1 建立**CA**证书

注：本文所有操作均在同一目录下进行

首先建立创建CA证书需要的配置文件`ca.cnf`，OpenSSL在生成证书时需要读取其中的内容，有一些无法通过OpenSSL命令行传递的参数也需要通过配置文件指定，其内容如下。

```
HOME = .
RANDFILE = $ENV::HOME/.rnd
####################################################################
[ ca ]
default_ca = CA_default # The default ca section
[ CA_default ]
default_days = 1000 # how long to certify for
default_crl_days = 30 # how long before next CRL
default_md = sha256 # use public key default MD
preserve = no # keep passed DN ordering
x509_extensions = ca_extensions # The extensions to add to the cert
email_in_dn = no # Don't concat the email in the DN
copy_extensions = copy # Required to copy SANs from CSR to cert
#====Following 7 lines are for signing other certs, not for making the CA cert.====
base_dir = .
```

```
certificate = $base_dir/cacert.pem # The CA certifcate
private_key = $base_dir/cakey.pem # The CA private key
new_certs_dir = $base_dir # Location for new certs after signing
database = $base_dir/index.txt # Database index file
serial = $base_dir/serial.txt # The current serial number
unique_subject = no # Set to 'no' to allow creation of several certificates with same subject.
####################################################################
[ req ]
default_bits = 4096
default_keyfile = cakey.pem
distinguished_name = ca_distinguished_name
x509_extensions = ca_extensions
string_mask = utf8only
####################################################################
[ ca_distinguished_name ]
countryName = Country Name (2 letter code)
countryName_default = CN
stateOrProvinceName = State or Province Name (full name)
stateOrProvinceName_default = Heilongjiang
localityName = Locality Name (eg, city)
localityName_default = Harbin
organizationName = Organization Name (eg, company)
organizationName_default = BY2HIT
organizationalUnitName = Organizational Unit (eg, division)
organizationalUnitName_default = R&D
commonName = Common Name (e.g. server FQDN or YOUR name)
commonName_default = Test CA
emailAddress = Email Address
emailAddress_default = test@example.com
####################################################################
[ ca_extensions ]
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always, issuer
basicConstraints = critical, CA:true
keyUsage = keyCertSign, cRLSign
```

```
#====All lines below are for signing other certs, not for making the CA cert.======
#########################################################################
[ signing_policy ]
countryName = optional
stateOrProvinceName = optional
localityName = optional
organizationName = optional
organizationalUnitName = optional
commonName = supplied
emailAddress = optional
#########################################################################
[ signing_req ]
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
basicConstraints = CA:FALSE
keyUsage = digitalSignature, keyEncipherment
```

上述配置文件中包含两部分内容，不同用途的配置已经通过注释标明，分别是

- 生成CA证书需要的配置
- 使用CA证书签署其他证书需要的配置

`ca_distinguished_name`部分设置了一些默认值，可以在调试的过程中避免重复输入多次。其他配置项的介绍可以参阅参考文献或OpenSSL官方文档。

设置好配置文件后，使用下列命令调用openssl的req子模块产生我们的CA证书，由于在配置文件中已经指定了选项的默认值，所以一路回车即可。

```
openssl req -x509 -config ca.cnf -newkey rsa:4096 -sha256 -nodes -out cacert.pem -outform PEM
```

上述命令中

- -x509 表示我们要对证书进行自签名
- -newkey 表示我们要为证书创建一个新的私钥，采用4096bit的秘钥长度（也可以指定已有的私钥，不过证书与私钥应该是一对一的，两份证书不应该使用同一个私钥。不过在更新到期证书时，需要使用到原来的秘钥，超出本文讨论范围了。）
- -sha256 指定签名算法，如果不指定，默认为sha-1，但是sha-1已经被很多浏览器认为不安全，因此需要采用sha-256
- -nodes 表示不要使用des对证书进行加密（不推荐，但是为了演示方便，不加密码，网上其他例子也都是这么干的……）

上述命令执行完之后，当前工作目录下会出现两个文件：

- `cacert.pem` CA证书文件
- `cakey.pem` CA证书对应的私钥，也就是上面通过指定`-newkey rsa:4096`生成的私钥。这个文件名是在配置文件中指定的，实际上也可以通过命令行来指定输出的私钥文件名。

Setp 2 建立**HTTPS**证书

建立以下配置文件`https.cnf`

```
HOME = .
RANDFILE = $ENV::HOME/.rnd
####################################################################
[ req ]
default_bits = 2048
default_keyfile = httpskey.pem
distinguished_name = server_distinguished_name
req_extensions = server_req_extensions
string_mask = utf8only
####################################################################
[ server_distinguished_name ]
```

```
countryName = Country Name (2 letter code)
countryName_default = CN
stateOrProvinceName = State or Province Name (full name)
stateOrProvinceName_default = Heilongjiang
localityName = Locality Name (eg, city)
localityName_default = Harbin
organizationName = Organization Name (eg, company)
organizationName_default = BY2HIT
commonName = Common Name (e.g. server FQDN or YOUR name)
commonName_default = by2hit.net
emailAddress = Email Address
emailAddress_default = test@example.com
######################################################################
[ server_req_extensions ]
subjectKeyIdentifier = hash
basicConstraints = CA:FALSE
keyUsage = digitalSignature, keyEncipherment
subjectAltName = @alternate_names
nsComment = "OpenSSL Generated Certificate"
######################################################################
[ alternate_names ]
DNS.1 = by2hit.net
DNS.2 = localhost
DNS.3 = 127.0.0.1
# IPv4 localhost
IP.1 = 127.0.0.1
# IPv6 localhost
IP.2 = ::1
```

在上述配置文件中，`alternate_names`是最重要的一个部分，请根据自己的实际情况进行修改。需要注意的是，Chrome浏览器要求IP地址必须写成IP.1=xxx的格式，而不能写成DNS.1=xxx的形式，否则会报`ERR_CERT_COMMON_NAME_INVALID`。据说IE浏览器要求必须写DNS.1=127.0.0.1的形式，没有验证是不是真的，为了兼容，两种形式在上面都写了。

使用下列命令，生成HTTPS证书的签署请求文件(CSR, Certificate Signing Request)。这里没有指定-x509, 说明这不是一个自签署的证书，而是要交给CA签名认证。

```
openssl req -config https.cnf -newkey rsa:2048 -sha256 -nodes -out httpscert.csr -outform PEM
```

上述步骤过后，工作目录多了两个文件

- httpscert.csr 请求文件，如果不是使用自己搭建的CA，则这个文件要提交给第三方CA进行审核。
- httpskey.pem 未来申请到的HTTPS证书对应的私钥

通过下列命令检查一下生成的csr文件是否信息正确无误。

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

Setp 3 使用**CA**证书签名**HTTPS**证书

由于一个根证书可以签署很多其他证书，因此需要一个地方来记录这个根证书都签署过哪些证书。 使用下列命令，建立两个文件用于存储这些信息

```
touch index.txt
echo '01' > serial.txt
```

最后，使用下列命令来根据csr文件生成一个证书

```
openssl ca -config ca.cnf -policy signing_policy -extensions signing_req -out httpscert.pem -infile
```

最后生成的`httpscert.pem`文件即为我们的证书。另外，还生成了很多其他的文件用于记录生成证书的系列信息，不再深入研究。

检查生成的证书

```
openssl x509 -in httpscert.pem -text -noout
```

至此，我们的证书已经生成完毕，总结一下，包含4个重要文件

- `cakey.pem` CA根证书私钥
- `cacert.pem` CA根证书
- `httpskey.pem` 被签署的证书私钥
- `httpscert.pem` 被签署的证书

其余文件多为配置文件或信息交换文件，也应当妥善保管。

第二种方式：直接使用CA证书作为HTTPS证书

准备配置文件，保存为 `mysite.cnf`, 根据情况修改 重点注意alt_names

```
[ req ]
default_bits = 2048
distinguished_name = server_distinguished_name
req_extensions = req_ext
x509_extensions = x509_ext
[ server_distinguished_name ]
countryName = Country Name (2 letter code)
countryName_default = CN
stateOrProvinceName = State or Province Name (full name)
stateOrProvinceName_default = Heilongjiang
```

```
localityName = Locality Name (eg, city)
localityName_default = Harbin
organizationName = Organization Name (eg, company)
organizationName_default = BY2HIT
commonName = Common Name (e.g. server FQDN or YOUR name)
commonName_default = by2hit.net
emailAddress = Email Address
emailAddress_default = test@example.com
[ x509_ext ]
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
basicConstraints = CA:FALSE
keyUsage = digitalSignature, keyEncipherment
subjectAltName = @alt_names
nsComment = "OpenSSL Generated Certificate"
[ req_ext ]
subjectAltName = @alt_names
[alt_names]
DNS.1 = by2hit.net
DNS.2 = localhost
DNS.3 = 127.0.0.1
# IPv4 localhost
IP.1 = 127.0.0.1
# IPv6 localhost
IP.2 = ::1
```

生成证书，使用以下命令

```
openssl req -config mysite.cnf -new -newkey rsa:2048 \
-nodes -sha512 -keyout mysitekey.pem -x509 -days 365 \
-out mysitecert.pem
```

至此，我们就得到了需要的证书和其对应的私钥，分别为：

- `mysitekey.pem` 证书私钥
- `mysitecert.pem` 证书文件

参考文献