openssl签署和自签署证书的多种实现方式

分类: OpenSSL, Linux 基础篇

undefined

openssl系列文章:http://www.cnblogs.com/f-ck-need-u/p/7048359.html

1.采用自定义配置文件的实现方法

1.1 自建CA

自建CA的机制:1.生成私钥;2.创建证书请求;3.使用私钥对证书请求签名。

由于测试环境, 所以自建的CA只能是根CA。所使用的配置文件如下。

```
[default]
name = root-ca    /* 变量*/
default_ca = CA_default
name_opt = ca_default
cert_opt = ca_default

[CA_default]
home = .     /* 变量*/
database = $home/db/index
serial = $home/db/serial
crlnumber = $home/db/crlnumber
certificate = $home/$name.crt
private_key = $home/private/$name.key
RANDFILE = $home/private/random
new_certs_dir = $home/certs
unique_subject = no
copy_extensions = none
default_days = 3650
default_crl_days = 365
default_md = sha256
policy = policy_to_match

[policy_to_match]
countryName = match
stateOrProvinceName = optional
organizationName = match
organizationalUnitName = optional
commonName = supplied
emailAddress = optional

[CA_DN]
countryName = "C"
contryName_default = "CN"
organizationName = "O"
organizationName_default = "jmu"
commonName = "CN"
commonName_default = "longshuai.com"

[req]
default_bits = 4096
encrypt_key = no
default_md = sha256
utf8 = yes
string_mask = utf8only
# prompt = no  /* 测试时该选项导致出错, 所以将其注释掉*/
```

```
distinguished_name = CA_DN
req_extensions = ca_ext

[ca_ext]
basicConstraints = critical,CA:true
keyUsage = critical,keyCertSign,cRLSign
subjectKeyIdentifier = hash
```

(1).创建openssl的目录结构

(a).创建配置文件

```
[root@xuexi ~]# mkdir /ssl;touch /ssl/ssl.conf

[root@xuexi ~]# cd /ssl

[root@xuexi ssl]# vim ssl.conf
```

(b).创建openssl的目录结构中的目录，在上述配置文件中的目录分别为/ssl/db、/ssl/private和/ssl/certs，可以考虑将private目录的权限设置为600或者400。

```
[root@xuexi ssl]# mkdir /ssl/{db,private,certs}

[root@xuexi ssl]# chmod -R 400 private/
```

(2).CA自签名

普通的证书请求需要使用CA的私钥进行签名变成证书，既然是自签名证书那当然是使用自己的私钥来签名。可以使用伪命令req、ca、x509来自签名。

使用req伪命令创建CA

这里有两种方法：1.一步完成，即私钥、证书请求、自签名都在一个命令中完成2.分步完成，先生成私钥、再创建证书请求、再指定私钥来签名。方法2中其实生成私钥和证书申请可以合并在一步中完成，证书申请和签名也可以合并在一步中完成。

方法一：一步完成

在下面的一步命令中，使用-new由于没有指定私钥输出位置，所以自动保存在ssl.conf中default_keyfile指定的private.pem中；由于ssl.conf中的req段设置了encrypt_key=no，所以交互时不需要输入私钥的加密密码；由于使用req -x509自签名的证书有效期默认为30天，而配置文件中req段又不能配置该期限，所以只能使用-days来指定有效期限，注意这个-days选项只作用于x509签名，证书请求中如果指定了时间是无效的。

```
[root@xuexi ssl]# openssl req -x509 -new -out req.crt -config ssl.conf -days 365
[root@xuexi ssl]# ll
total 24
drwxr-xr-x 2 root root 4096 Nov 22 09:05 certs
drwxr-xr-x 2 root root 4096 Nov 22 09:05 db
drwx------ 2 root root 4096 Nov 22 09:05 private
-rw-r--r-- 1 root root 3272 Nov 22 10:52 private.pem   /* 注意权限为644 */
-rw-r--r-- 1 root root 1753 Nov 22 10:52 req.crt
-rw-r--r-- 1 root root 1580 Nov 22 10:51 ssl.conf
[root@xuexi ssl]# openssl x509 -noout -dates -in req.crt
notBefore=Nov 22 02:52:24 2016 GMT
notAfter=Nov 22 02:52:24 2017 GMT
```

方法二：分步完成，这里把各种可能的步骤合并都演示一遍

>>创建私钥和证书请求合并而签名独自进行的方法<<

```
[root@xuexi ssl]# openssl req -newkey rsa:1024 -keyout key.pem -out req1.csr -config ssl.conf -days 365
[root@xuexi ssl]# openssl req -x509 -in req1.csr -key key.pem -out req1.crt
[root@xuexi ssl]# openssl x509 -noout -dates -in req1.crt /* 注意签名不要配置文件 */
notBefore=Nov 22 02:58:25 2016 GMT
notAfter=Dec 22 02:58:25 2016 GMT   /* 可以看到证书请求中指定-days是无效的 */
[root@xuexi ssl]# ll
total 36
drwxr-xr-x 2 root root 4096 Nov 22 09:05 certs
drwxr-xr-x 2 root root 4096 Nov 22 09:05 db
-rw-r--r-- 1 root root  912 Nov 22 10:57 key.pem
drwx------ 2 root root 4096 Nov 22 09:05 private
-rw-r--r-- 1 root root 3272 Nov 22 10:52 private.pem
-rw-r--r-- 1 root root  826 Nov 22 10:58 req1.crt
-rw-r--r-- 1 root root  688 Nov 22 10:57 req1.csr
-rw-r--r-- 1 root root 1753 Nov 22 10:52 req.crt
```

```
-rw-r--r-- 1 root root 1580 Nov 22 10:51 ssl.conf
```

>>独自生成私钥，而请求和签名合并的方法<<

```
[root@xuexi ssl]# (umask 077;openssl genrsa -out key1.pem 1024)
[root@xuexi ssl]# openssl req -x509 -new -key key1.pem -out req2.crt -config ssl.conf -days 365
[root@xuexi ssl]# openssl x509 -noout -dates -in req2.crt
notBefore=Nov 22 03:28:31 2016 GMT
notAfter=Nov 22 03:28:31 2017 GMT
[root@xuexi ssl]# ll
total 44
drwxr-xr-x 2 root root 4096 Nov 22 09:05 certs
drwxr-xr-x 2 root root 4096 Nov 22 09:05 db
-rw-r--r-- 1 root root  912 Nov 22 10:57 key1.pem
-rw------- 1 root root  887 Nov 22 11:26 key2.pem
drwx------ 2 root root 4096 Nov 22 09:05 private
-rw-r--r-- 1 root root 3272 Nov 22 10:52 private.pem
-rw-r--r-- 1 root root  826 Nov 22 10:58 req1.crt
-rw-r--r-- 1 root root  688 Nov 22 10:57 req1.csr
-rw-r--r-- 1 root root  709 Nov 22 11:28 req2.crt
-rw-r--r-- 1 root root 1753 Nov 22 10:52 req.crt
-rw-r--r-- 1 root root 1580 Nov 22 10:51 ssl.conf
```

>>完全分步进行<<

```
[root@xuexi ssl]# rm -rf key* req* private.pem
[root@xuexi ssl]# (umask 077;openssl genrsa -out key.pem 1024)
[root@xuexi ssl]# openssl req -new -key key.pem -out req.csr -config ssl.conf
[root@xuexi ssl]# openssl req -x509 -key key.pem -in req.csr -out req.crt -days 365
[root@xuexi ssl]# openssl x509 -noout -dates -in req.crt
notBefore=Nov 22 04:29:21 2016 GMT
notAfter=Nov 22 04:29:21 2017 GMT
[root@xuexi ssl]# ll
total 28
drwxr-xr-x 2 root root 4096 Nov 22 09:05 certs
drwxr-xr-x 2 root root 4096 Nov 22 09:05 db
-rw------- 1 root root  887 Nov 22 12:28 key.pem
drwx------ 2 root root 4096 Nov 22 09:05 private
-rw-r--r-- 1 root root  826 Nov 22 12:29 req.crt
-rw-r--r-- 1 root root  688 Nov 22 12:28 req.csr
-rw-r--r-- 1 root root 1580 Nov 22 10:51 ssl.conf
```

在本节的开头说明了创建证书请求时需要提供私钥，这个私钥的作用是为了提供公钥。下面是验证。

```
/* 提取私钥key.pem中的公钥到key.pub文件中 */
[root@xuexi ssl]# openssl rsa -in key.pem -pubout -out key.pub
/* 输出证书请求req.csr中的公钥部分 */
[root@xuexi ssl]# openssl req -noout -pubkey -in req.csr
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC+YBneLYbh+OZWpiyPqIQHOsU5
D8il6UF7hi3NgEX/6vtciSmp7GXpXUV1tDglCCTPOfCHcEzeO0Gvky21LUenDsl/
aC2lraSijpl41+rT4mKNrCyDPZw4iG44+vLHfgHb3wJhBbBk0aw51dmxUat8FHCL
hU7nx+Du637UDlwdEQIDAQAB
-----END PUBLIC KEY-----
/* 查看key.pub，可以发现和req.csr中的公钥是一样的 */
[root@xuexi ssl]# cat key.pub
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC+YBneLYbh+OZWpiyPqIQHOsU5
D8il6UF7hi3NgEX/6vtciSmp7GXpXUV1tDglCCTPOfCHcEzeO0Gvky21LUenDsl/
aC2lraSijpl41+rT4mKNrCyDPZw4iG44+vLHfgHb3wJhBbBk0aw51dmxUat8FHCL
hU7nx+Du637UDlwdEQIDAQAB
-----END PUBLIC KEY-----
```

虽然创建证书请求时使用的是公钥，但是却不能使用-key选项指定公钥，而是只能指定私钥，因为req -new或-newkey选项会调用openssl rsa命令来提取公钥，指定公钥该调用将执行失败。

使用x509伪命令创建CA

使用x509伪命令需要提供请求文件，因此需要先创建证书请求文件。由于x509伪命令签名时不读取配置文件，所以不需要设置配置文件，若需要某选项，只需使用x509中对应的选项来达成即可。

以下x509 -req用于自签名，需要-signkey提供签名所需私钥key.pem。

```
[root@xuexi ssl]# openssl req -new -keyout key.pem -out req.csr -config ssl.conf
[root@xuexi ssl]# openssl x509 -req -in req.csr -signkey key.pem -out x509.crt
```


最新评论

使用ca伪命令创建CA

使用ca伪命令自签名会读取配置文件中的ca部分，所以配置文件中所需的目录和文件结构都需要创建好，包括目录db、private、certs，文件db/index、db/serial，并向serial中写入一个序列号。由于是自签名，可以自行指定私钥文件，因此对于签名所需CA私钥文件无需放置在private目录中。

```
[root@xuexi ssl]# touch db/{serial,index}
[root@xuexi ssl]# echo "01" > db/serial
[root@xuexi ssl]# openssl req -new -keyout key.pem -out req.csr -config ssl.conf
[root@xuexi ssl]# openssl ca -selfsign -keyfile key.pem -in req.csr -config ssl.conf
```

在此签名过程中有两次询问，如下：

```
Certificate is to be certified until Nov 20 06:34:41 2026 GMT (3650 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

若要无交互，则使用-batch进入批处理模式。

```
[root@xuexi ssl]# openssl ca -selfsign -keyfile key.pem -in req.csr -config ssl.conf -batch
```

1.2 为其他证书请求签名

CA为其他请求或证书签名时，需要使用到的文件有：自己的CA证书和自己的私钥文件。因此签名过程中需要提供这两个文件。

(1).使用ca伪命令为其他证书请求签名

使用ca伪命令自建根CA后，目录结构如下：

```
[root@xuexi ssl]# tree -R -C
.
├── certs
│   └── 01.pem
├── db
│   ├── index
│   ├── index.attr
│   ├── index.old
│   ├── serial
│   └── serial.old
├── key.pem
├── private
├── req.csr
└── ssl.conf
```

其中01.pem是根CA证书，key.pem是根CA私钥。

现在要为其他证书请求签名，首先创建其他请求吧，假设该请求文件/tmp/req.csr。

```
[root@xuexi ssl]# openssl req -new -keyout /tmp/key.pem -out /tmp/req.csr -config ssl.conf
```

使用根证书01.pem为/tmp/req.csr签名。

```
[root@xuexi ssl]# openssl ca -in /tmp/req.csr -keyfile key.pem -cert certs/01.pem -config ssl.conf -batch
```

这样挺麻烦，因为每次为别人签名时都要指定-cert和-keyfile，可以将CA的证书和CA的私钥移动到配置文件中指定的路径下：

```
certificate = $home/$name.crt

private_key = $home/private/$name.key
```

```
[root@xuexi ssl]# mv certs/01.pem root-ca.crt
[root@xuexi ssl]# mv key.pem private/root-ca.key
```

再使用ca签名时将可以使用默认值。

```
[root@xuexi ssl]# openssl ca -in /tmp/req.csr -config ssl.conf -batch
```

(2).使用x509伪命令为其他证书请求签名

现在根CA证书为root-ca.crt，CA的私钥为private/root-ca.key。

下面使用x509伪命令实现签名。由于x509不会读取配置文件，所以需要提供签名的序列号，使用-CAcreateserial可以在没有序列号文件时自动创建；由于x509默认-in指定的输入文件是证书文件，所以要对请求文件签名，需要使用-req来表示输入文件为请求文件。

```
[root@xuexi ssl]# openssl x509 -req -in /tmp/req.csr -CA root-ca.crt -CAkey private/root-ca.key -out x509.crt -CAcreateserial
```

2.采用默认配置文件/etc/pki/tls/openssl.cnf的实现方法

这是推荐采用的方法，因为方便管理，但使用默认配置文件，需要进行一些初始化动作。

由于完全采用/etc/pki/tls/openssl.cnf的配置，所以要建立相关文件。

自建CA的过程：

```
[root@xuexi tmp]# touch /etc/pki/CA/index.txt
[root@xuexi tmp]# echo "01" > /etc/pki/CA/serial
[root@xuexi tmp]# openssl genrsa -out /etc/pki/CA/private/cakey.pem     # 创建CA的私钥
[root@xuexi tmp]# openssl req -new -key /etc/pki/CA/private/cakey.pem -out rootCA.csr    # 创建CA待自签署的证书请求文件
[root@xuexi tmp]# openssl ca -selfsign -in rootCA.csr                       # 自签署
[root@xuexi tmp]# cp /etc/pki/CA/newcerts/01.pem /etc/pki/CA/cacert.pem    # 将自签署的证书按照配置文件的配置复制到指定位置
```

为他人颁发证书的过程：

```
[root@xuexi tmp]# openssl ca -in youwant1.csr
```

签署成功后，证书位于/etc/pki/CA/newcert目录下，将新生成的证书文件发送给申请者即可。

分类: OpenSSL, Linux 基础篇

骏马金龙
关注 - 26
粉丝 - 2017
+加关注

« 上一篇: OpenSSL主配置文件openssl.cnf
» 下一篇: openssl大纲

posted @ 2016-11-22 20:57 骏马金龙 阅读(6649) 评论(0) 编辑 收藏 举报