

Tutorial

Criando uma Infraestrutura de PKI com uma CA Própria Usando OpenSSL

Este tutorial descreve como criar uma Infraestrutura de Chave Pública (PKI) com uma Autoridade Certificadora (CA) própria e como gerar um certificado de cliente para uso com mTLS. Vamos usar o OpenSSL, uma poderosa ferramenta de linha de comando para criptografia e gestão de certificados.

Passo 1: Configuração Inicial

Antes de começar, certifique-se de ter o OpenSSL instalado. Em sistemas baseados em Linux, você pode instalar o OpenSSL com:

```
sudo apt-get install openssl
```

Passo 2: Configurar o Diretório da CA

Crie um diretório para a CA e subdiretórios para armazenar certificados, chaves e outros arquivos necessários.

```
mkdir -p ~/myCA/{certs,crl,newcerts,private}  
chmod 700 ~/myCA/private  
touch ~/myCA/index.txt  
echo 1000 > ~/myCA/serial
```

Passo 3: Criar o Arquivo de Configuração OpenSSL

Crie um arquivo de configuração para a CA (`~/myCA/openssl.cnf`). Aqui está um exemplo básico:

```
[ ca ]
default_ca = CA_default

[ CA_default ]
dir = ~/myCA
certs = $dir/certs
crl_dir = $dir/crl
new_certs_dir = $dir/newcerts
database = $dir/index.txt
serial = $dir/serial
RANDFILE = $dir/private/.rand
private_key = $dir/private/ca.key.pem
certificate = $dir/certs/ca.cert.pem
crlnumber = $dir/crlnumber
crl = $dir/crl.pem
crl_extensions = crl_ext
default_crl_days = 30
default_md = sha256
name_opt = ca_default
cert_opt = ca_default
default_days = 375
preserve = no
policy = policy_strict
```

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

```
[ req ]  
default_bits = 2048  
distinguished_name = req_distinguished_name  
string_mask = utf8only
```

```
[ req_distinguished_name ]  
countryName = Country Name (2 letter code)  
stateOrProvinceName = State or Province Name  
localityName = Locality Name  
O.organizationName = Organization Name  
commonName = Common Name  
emailAddress = Email Address  
[ v3_ca ]  
subjectKeyIdentifier = hash  
authorityKeyIdentifier = keyid:always,issuer  
basicConstraints = critical, CA:true  
keyUsage = critical, digitalSignature, cRLSign, keyCertSign
```

```
[ v3_req ]  
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
```

```
[ usr_cert ]
basicConstraints = CA:FALSE
nsCertType = client, email
nsComment = "OpenSSL Generated Client Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, nonRepudiation, digitalSignature,
keyEncipherment
extendedKeyUsage = clientAuth, emailProtection
```

Substitua os valores da seção [req_distinguished_name] por valores que identifique a sua instituição.

Passo 4: Gerar a Chave Privada e o Certificado da CA

1. Gerar a chave privada da CA:

```
openssl genrsa -aes256 -out ~/myCA/private/ca.key.pem 4096
chmod 400 ~/myCA/private/ca.key.pem
openssl genrsa -aes256 -out ~/myCA/private/ca.key.pem 4096
chmod 400 ~/myCA/private/ca.key.pem
```

2. Gerar o certificado da CA:

```
openssl req -config ~/myCA/openssl.cnf -key
~/myCA/private/ca.key.pem -new -x509 -days 7300 -sha256 -extensions
v3_ca -out ~/myCA/certs/ca.cert.pem
chmod 444 ~/myCA/certs/ca.cert.pem
```

Passo 5: Gerar uma Chave e Certificado de Cliente

1. Gerar a chave privada do cliente:

```
openssl genrsa -out ~/myCA/private/client.key.pem 2048  
chmod 400 ~/myCA/private/client.key.pem
```

Atenção: A chave privada e a senha do certificado do cliente não deverão ser enviadas para o FITBANK em nenhuma hipótese.

2. Gerar a CSR (Certificate Signing Request) do cliente:

```
openssl req -config ~/myCA/openssl.cnf -key  
~/myCA/private/client.key.pem -new -  
sha256 -out ~/myCA/client.csr.pem
```

3. Assinar o CSR com a CA para criar o certificado do cliente:

```
openssl ca -config ~/myCA/openssl.cnf -extensions usr_cert -days 375 -  
notext -md sha256 -in ~/myCA/client.csr.pem -out  
~/myCA/certs/client.cert.pem  
chmod 444 ~/myCA/certs/client.cert.pem
```

Passo 6: Verificar os Certificados

Verifique o certificado da CA:

```
openssl x509 -noout -text -in ~/myCA/certs/ca.cert.pem
```

Verifique o certificado do cliente:

```
openssl x509 -noout -text -in ~/myCA/certs/client.cert.pem
```

Passo 7: Criar um Arquivo de Cliente no Formato PKCS#12 (.pfx)

Para facilitar a importação do certificado e chave privada em diversas aplicações e dispositivos, você pode criar um arquivo PKCS#12 (.pfx).

1. Criar o arquivo PKCS#12 (.pfx):

```
openssl pkcs12 -export -out ~/myCA/certs/client.pfx -inkey  
~/myCA/private/client.key.pem -in ~/myCA/certs/client.cert.pem -certfile  
~/myCA/certs/ca.cert.pem
```

Certificados simples, armazenados em arquivos .pem ou .crt, como o que foi gerado no passo 5, os arquivos em formato .pfx geralmente são acompanhados de arquivos de chave privada separados (.key). Embora esses arquivos possam ser protegidos por permissões de sistema de arquivos, eles não oferecem criptografia embutida e proteção por senha como os arquivos .pfx.

Um arquivo .pfx combina o certificado digital e a chave privada em um único arquivo. Isso facilita o gerenciamento e a distribuição segura desses componentes críticos, reduzindo a probabilidade de perda ou separação acidental da chave privada e do certificado. Além disso, os arquivos .pfx são protegidos por criptografia forte. Quando você cria um arquivo .pfx, é possível definir uma senha que protege o conteúdo do arquivo. Isso adiciona uma camada extra de segurança, pois mesmo que o arquivo seja interceptado, ele não pode ser acessado sem a senha correta.

Passo 8: Enviar o certificado PÚBLICO da CA para a trust store do FITBANK

Para uso com mTLS, é necessário exportar a cadeia de certificados (CA e intermediários, se houver) para que o cliente possa apresentar essa cadeia ao servidor durante o processo de handshake mTLS. Como se trata de uma CA privada a instituição deverá enviar o certificado público da CA para nós através de e-mail informado no processo de integração.

Atenção: A chave privada e a senha da CA **não deverão** ser enviadas para o FITBANK em nenhuma hipótese.

Como proceder para efetuar a chamada utilizando mTLS?

A forma de efetuar a chamada vai variar de acordo com o sistema ou linguagem de programação escolhidos para a realização dos requests. Nesse sentido, é recomendável executar primeiro a chamada utilizando o Postman, para garantir que a solução está funcionando como se espera e depois, "traduzir" para plataforma escolhida.

Atenção: Os endpoints do FITBANK só aceitam conexões com TLS 1.2, outras conexões serão rejeitadas. **A aplicação do cliente deverá realizar somente conexões TLS 1.2.**

Conclusão

Você criou com sucesso uma infraestrutura de PKI com uma CA própria e gerou um certificado de cliente para uso com mTLS. Esse processo garante que ambos os lados da comunicação (cliente e servidor) possam autenticar-se mutuamente, aumentando a segurança das transações e comunicações.

Agora, você pode usar esses certificados em suas configurações de mTLS para autenticação mútua, protegendo suas APIs contra acessos não autorizados e garantindo a integridade das comunicações.



Av. Cidade Jardim, 400 – 20º andar
Jardim Europa – São Paulo/SP

www.fitbank.com.br