# Certificados Digitais com OpenSSL

## Sumário



## 1. Introdução

OpenSSL é uma ferramenta de linha de comando que permite a geração de chaves privadas, certificados digitais, números aleatórios, hashes, etc. É amplamente utilizada no mercado e durante esse roteiro vamos aprender um pouco a usá-la.

Para instalá-la no Linux, você deve usar o comando:

```
$ sudo apt install openssl
```

Verifique se a instalação foi bem sucedida utilizando o comando:

```
$ openssl version
```

Para apendermos alguns comandos da ferramenta vamos criar um cenário imaginário, somos uma empresa e queremos criar nosso certificado TLS para nossa página web e que vamos se autenticar numa Certification Authorithy (CA) para que os visitantes do site possam se conectar de forma segura.

No nosso caso, também vamos criar a CA, para podermos explorar mais comandos.
Utilizaremos os padrões de segurança recomendados pelo NIST como algoritmo de criptografia assimétrica RSA e tamanho de chave privada RSA ≥ 2048.

## Criando chave privada

Como primeiro passo, precisamos criar uma chave privada, aquela que nós não devemos compartilhar com **ninguém**. Faremos isso com o seguinte comando:

```
$ openssl genrsa -out sua-chave-privada.key 2048
```

- genrsa: Gera uma chave privada RSA.
- -out <file-name>: Se refere ao nome que a chave gerada vai receber.
- 2048: É o tamanho da chave privada.

**OBS:** o comando pode receber outros parâmetros que não foram colocados para simplificar. Caso deseje se aprofundar, olhe os demais parâmetros na documentação.

## Gerando uma requisição de assinatura

Bem, agora que temos a nossa chave privada, podemos derivar a chave publica e fazer uma requisição para que alguma CA possa assinar. Isso é feito da seguinte forma:

```
$ openssl req -new -key sua-chave-privada.key -out sua-requisicao.csr
```

- req: Comando para criar/processar requisição de certificado. Também pode gerar um par de chave privada e certificado auto assinado.
- -new: Argumento para pegar os dados do usuário no próprio terminal ao invés de algum arquivo de configuração, que pode ser utilizado. Para mais detalhes, consulte a documentação.
- -key <file-name>: A chave privada que vai ser usado para derivar a chave pública do certificado.
- -out <file-name>: Se refere ao nome que o arquivo (.csr) gerado vai receber.

Perceba que ele pede que alguns parametros sejam preenchidos, como:

- Country Name (2 letter code) [AU]: Código de duas letras que representa o país.
- State or Province Name (full name) [Some-State]: Nome do estado ou provincia.
- Locality Name (eg, city) []: Nome da cidade.
- Organization Name (eg, company) [Internet Widgits Pty Ltd]: Nome da empresa.
- Organizational Unit Name (eg, section) []: Setor de atuação.

- Common Name (e.g. server FQDN or YOUR name) []: DNS (www.google.com).
- Email Address []: Email pra contato.
- A challenge password []: Uma senha que é basicamente um nonce secreto compartilhado entre você e a CA emissora do certificado SSL, incorporado no CSR, que o emissor pode usar para autenticá-lo, caso seja necessário.

## Gerando um certificado auto assinado

Para o nosso caso simulado, vamos precisar criar uma Certification Authority (CA), para isso, devemos criar um certificado auto assinado. Execute o seguinte comando:

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

- req: Comando para criar/processar requisição de certificado. Também pode gerar um par de chave privada e certificado auto assinado.
- -x509: Utilizado para gerar certificados auto assinados.
- -nodes: Não cifra a chave privada com uma senha.
- -days <days>: A quantidade de dias que esse certificado será válido.
- -newkey <key algorithm:size in bits>: Esse parâmetro cria uma nova chave privada e um signing request (que não fica salvo no disco), caso contrário, deve ser passado uma chave privada e um csr previamente criados, como mostrado nos comandos anteriores.
- -keyout <file-name>: O nome que será dado a chave privada gerada.
- -out <file-name>: Se refere ao nome que o certificado gerado (.crt/.cert) vai receber.

## Assinando uma requisição de assinatura

Criamos a nossa CA, então podemos voltar e assinar o certificado do nosso site. Para isso você deve executar:

```
$ openssl x509 -req -days 360 -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out server.crt
```

- x509: Comando para fazer as mais variadas operações utilizando certificados que seguem o padrão x509.
- -req: Ao invés de dar como entrada um certificado vamos colocar uma requisição de assinatura, esse parâmetro serve para indicar isso.
- -days <days>: A quantidade de dias que esse certificado será válido.
- -in <file-name>: Nome do arquivo.csr.

- -CA <file-name>: Certificado da CA.
- -CAkey <file-name>: Chave privada da CA.
- -CAcreateserial: Com esta opção e a opção -CA, o arquivo de número de série da CA é criado, caso não exista. Um número aleatório é gerado para ser usado no certificado e salvo no arquivo de número de série.
- -out <file-name>: Se refere ao nome que o certificado gerado (.crt/.cert) vai receber.

Agora, se todos os passos foram bem executados, temos um certificado digital nosso.

## Vendo certificados

Por fim, para visualizar certificados você poderia dar um `$ cat seu-certificado.crt` porém, você não veria os metadados do certificado, como o nome do detentor do certificado, o prazo de validade do certificado, etc. Para fazer isso, basta usar o seguinte comando:

```
$ openssl x509 -text -noout -in seu-certificado.crt
```

- x509: Comando para fazer as mais variadas operações utilizando certificados que seguem o padrão x509.
- -text: Retorna para a saída padrão o conteúdo completo do certificado.
- -noout: Previne que a resposta seja da versão codificada do certificado.
- -in <file-name>: Nome do arquivo que será lido.

## Atividade

Para o último lab, seu objetivo é utilizar a CA criada e os certificados dos servidores para um serviço web. Você deve criar uma aplicação web simples que retorna um “hello world” (em texto plano) quando acessada no `machine-ip:port/` . Você deve colocar o certificado digital criado no servidor web, de modo que possa ser possível acessá-lo usando **https.**

Após isso, crie uma outra CA independente e grave um vídeo mostrando o seu acesso ao servidor com o comando `curl` , uma vez passando como parâmetro a CA que foi utilizada pra assinar o certificado do servidor e uma vez utilizando a CA que criamos para a atividade. Ao final da atividade você deve explicar o porque das respostas do comando serem aquelas para cada chamada.

Para fazer o curl passando a CA como parâmetro utilize o comando:

```
$ curl https://machine-ip:port --capath <full path to the CA>
```

**Links úteis**

- Exemplo de código Pyhon para criar um web server utilizando Flask.
- Formulário para entrega do vídeo