Autenticação Web com certificados digitais

Leonel Filipe Simões Santos e Nuno Filipe Pedro Jacinto
Departamento de Engenharia Informática
Escola Superior de Tecnologia e Gestão de Leiria
Morro do Lena, Alto do Vieiro – Leiria
Telf: +351 244 820 300; fax: +351 244 820 310;
e-mail: ei06707@student.estg.ipleiria.pt, ei07013@student.estg.ipleiria.pt

***Resumo* — Este artigo pretende indicar uma opção de configuração do servidor Apache para suporte de autenticação de utilizadores com certificados digitais.**

1. Introdução

O objectivo deste artigo é mostrar como se pode configurar o Apache de forma a permitir que um utilizador se autentique utilizando certificados digitais.
Para tal, será apresentada uma introdução teórica do processo de autenticação, seguindo-se os passos de instalação e configuração das tecnologias utilizadas.
A utilização de uma ligação segura e a autenticação do cliente, obriga à criação de certificados tanto para o cliente como para o servidor. Estes terão de ser autenticados por uma CA (Certificate Authority), a fim de se poder verificar a sua autenticidade. Para tal criamos uma CA privada, sendo desta forma possível autenticar os nossos certificados de teste, tanto do servidor como dos clientes.
A verificação do certificado e a aceitação é feita de forma automática pelo Apache, sendo apenas necessário a configuração correcta do mesmo.
Como exemplo desta capacidade do Apache apoiada pelo PHP, criou-se um *script* em PHP que permite, acedendo às informações existentes no certificado, autenticar o cliente, verificando a sua existência na base de dados.
Por fim será apresentado sucintamente como se poderá importar para o Firefox um certificado a fim de testar o uso de certificados no processo de autenticação entre o cliente e o servidor.

2. Processo de autenticação com certificados

O processo de autenticação e comunicação vai utilizar criptografia assimétrica. Este tipo de criptografia surgiu para resolver dois grandes problemas da criptografia simétrica: a distribuição de chaves e as assinaturas digitais. Neste tipo de cifragem são utilizadas duas chaves relacionadas matematicamente, para cada entidade. São elas a chave pública, que é divulgada, e a chave privada, que é mantida em segredo. A sua principal utilização prende-se com o facto de, não só garantir a autenticação e o não repudio do emissor, mas também a garantia e confidencialidade e integridade das mensagens.

A divulgação ou distribuição da chave pública pode ser efectuada através de técnicas de distribuição de chaves públicas, tais como: o anúncio público, a disponibilização de uma directoria pública, a utilização de uma autoridade de chave pública e certificados de chave pública. A técnica que vamos utilizar é a utilização de certificados de chave pública. Esta técnica resolve o problema do ponto de engarrafamento que se pode gerar através da técnica da autoridade de chave pública devido a não ser necessário a comunicação com a autoridade para requerer a chave pública de um interveniente, mas apenas confiar na autoridade certificadora (CA) que assinou o certificado que é usado por um interveniente numa comunicação segura e usar a chave pública que consta nesse mesmo certificado (Fig.1).

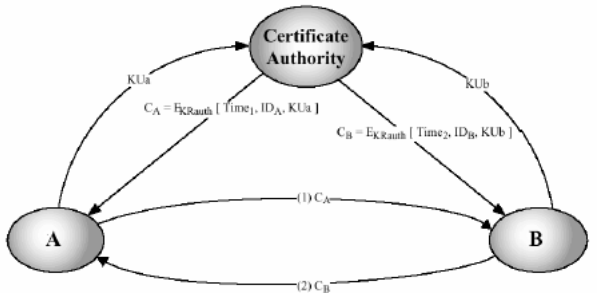


Fig .1.Troca de certificados de chave pública

Cada interveniente pode gerar o seu próprio certificado e enviar um pedido de assinatura a uma CA, comercial ou não, para depois o usar nas comunicações com outros participantes. Esse certificado contém diversas informações que podem ser vistas pelos participantes como o nome e a chave pública do dono do certificado, bem como a validade e o emissor do certificado. Pode também ser visto se o certificado já foi alterado após ter sido originado na CA, que é quem pode criar e actualizar certificados.
Este tipo de autenticação tem o problema de quando uma chave privada fica comprometida, isto porque mesmo que seja detectada e anulado certificado podem-se correr riscos enquanto nem todos os participantes tiverem conhecimento desse cancelamento.
Neste tipo de distribuição de chaves é utilizada a norma X.509 da *ITU-T* que é o standard que vamos utilizar para a geração de certificados com a estrutura do mesmo (Fig.2).

3. Tecnologias utilizadas

Para a realização de autenticação com certificados digitais no Apache utilizamos o servidor Apache, o OpenSSL, o PHP e o MySql. Estas serão apresentadas de forma sucinta neste capítulo, sendo também apresentada, sem grande profundidade, a forma como estas poderão ser instaladas.

O OpenSSL é uma implementação *open source* da *Secure Socket Layer* (versão 2 e 3) e da *Transport Layer Security* (versão 1). Este permite, em conjunto com o Apache através do modulo *ssl*, o estabelecimento de ligações seguras e a autenticação dos clientes através de certificados.
O processo de instalação, a partir do código fonte, é bastante simples, como se pode verificar no exemplo 1.

```
./config
make
make install
```

Exemplo 1 – Instalação do OpenSSL

Este processo irá instalar o OpenSSL no directório */usr/local/ssl*. Para definir outra directoria é necessário introduzir o parâmetro *–prefix*.

O Apache é um servidor de http, pelo menos esta é uma das principais tarefas que desempenha.
A sua instalação passa por definir os parâmetros de compilação e instalação, compilar e instalar. Os passos que se seguem assumem que nos encontramos na directoria onde temos o código fonte do Apache.
Para definir os parâmetros de compilação e instalação utiliza-se o *script* `configure`. Este irá necessitar de pelo menos um parâmetro adicional, definindo o directório onde o Apache será instalado. Assim, o comando para instalar o apache na directoria `/usr/local/apache` seria:

```
./configure –prefix /usr/local/apache
```

De forma a permitir que o Apache suporte *ssl*, é necessário compilar o Apache com este módulo. A partir da versão 2 do Apache este módulo já vem incluído no código fonte, bastando activá-lo. Para isso basta introduzir essa opção no *script* `configure`. Assim, o comando anterior teria de ter pelo menos as seguintes opções:

```
./configure –prefix /usr/local/apache --
     enable-ssl
```

Para compilar o apache utiliza-se o comando `make` e para que este seja colocado na directoria definida utiliza-se o comando `make install`. O exemplo 2 mostra todos os passos necessários para a instalação do apache.

```
./configure --prefix=/usr/local/apache \
     --enable-so --enable-cgi \
     --enable-info --enable-rewrite \
     --enable-speling \
     --enable-usertrack \
     --enable-deflate --enable-ssl \
     --enable-mime-magic
make
make install
```

Exemplo 2 – Instalação do Apache

O PHP é um linguagem de *scripting* para a construção de páginas html dinâmicas. A sua utilização em conjunto com o Apache permite obter a informação referente aos dados do certificado do utilizador.
A instalação depende das funcionalidades que se pretende que este disponibilize. O exemplo 3 demonstra uma possibilidade, disponibilizando o acesso ao MySql.

```
./configure \
     --with-apxs2=\
     /usr/local/apache/bin/apxs \
     --with-mysql \
     --prefix=/usr/local/apache/php \
     --with-config-file-path=\
     /usr/local/apache/php \
     --enable-force-cgi-redirect \
     --disable-cgi \
     --with-zlib \
     --with-gettext \
     --with-gdbm
make
make install
```

Exemplo 3 – Instalação do PHP

A utilização do exemplo anterior pressupõe que o Apache se encontra instalado na directoria */usr/local/apache*.
Para além da instalação do PHP, também será necessário acrescentar algumas opções no ficheiro de configuração do Apache, no httpd.conf. No exemplo 4 apresenta algumas dessas opções.

```
LoadModule php5_module
     modules/libphp5.so
# Add index.php to your DirectoryIndex
DirectoryIndex index.html index.php
# Use for PHP 5.x
AddHandler php5-script php
AddType text/html       php
# PHP Syntax Coloring
AddType application/x-httpd-php-source
     phps
```

Exemplo 4 – Exemplo de opções introduzidas no httpd.conf para o PHP

Os processos de instalação referidos anteriormente estão muito sucintos. Apenas servem como introdução para uma configuração básica.

4. Criação de uma CA e de certificados

Para a realização do cenário demonstrativo da autenticação com certificados é necessária a existência de certificados tanto para os clientes que se pretendem autenticar no servidor como para o servidor Apache para se autenticar nos clientes. Para isso optou-se por criar uma CA privada. O OpenSSL tem um *script* que permite a criação de uma CA, o CA.sh que se encontra na da directoria `misc` do OpenSSL. Utilizando este *script* basta adicionar o parâmetro `-newca` para que seja criada a CA. O comando completo seria:

```
./CA.sh -newca
```

Este *script* irá criar um certificado que permitirá assinar os certificados dos clientes e do servidor. Será este certificado que iremos mais tarde importar para o *browser* na parte das CA em que confiamos e colocaremos na configuração do apache para verificar os certificados dos clientes.
Para além do certificado para a CA o *script* anterior irá também criar uma estrutura de directórios onde serão colocados os ficheiros necessários para a gestão das assinaturas dos certificados que assinar. O exemplo 5 contém os comandos necessários para a criação da CA manualmente.

```
# create a Certificate Authority
# create the directory hierarchy
mkdir /etc/CA
#  directory containing the certificate
mkdir /etc/CA/certs
# directory containing the certificate
# revocation list files (invalid
# certificates)
mkdir /etc/CA/crl
# directory containing new certificates
mkdir /etc/CA/newcerts
# directory containing the private key
mkdir /etc/CA/private
# create file serial initiating to 01
# the serial file is a text file
# containing the next serial number to
# use in hex format
echo "01" > /etc/CA/serial
# create index.txt file
# in this file an index of all
# certificates authenticated by this CA
# will be created
echo "" > /etc/CA/index.txt
# create CA certificate valid for 10
# years
openssl  req  -new -x509 \
    -keyout /etc/CA/private/cakey.pem \
    -out /etc/CA/cacert.pem -days 3650
```

Exemplo 5 – Criação de uma CA

Para o estabelecimento de uma ligação ssl pelo Apache é necessário que o servidor tenha um certificado autenticado. Para isso é necessário criar um pedido de certificado que depois será autenticado pela CA. Neste exemplo o certificado será autenticado pela CA criada anteriormente. Para a criação de um pedido de certificado utiliza-se o comando:

```
openssl req -new -keyout new_req.pem -out
     new_req.pem 365
```

Neste exemplo criou-se um certificado válido por 1 ano (365 dias).
Este pedido de certificado contém a chave privada e pública. Para retirar a chave privada utiliza-se o comando :

```
openssl rsa < newreq.pem > new_key.pem
```

o ficheiro *new_key.pem* irá conter a chave privada do servidor.
Para assinar o certificado utilizando a CA privada utiliza-se o comando:

```
openssl ca -policy policy_anything \
    -out new_cert.pem -infiles new_req.pem
```

O ficheiro *new.crt* irá conter o novo certificado. O certificado do servidor e a sua chave pública serão utilizados para autenticar o servidor no cliente, aquando o estabelecimento da ligação. A configuração do apache para utilizar estes certificados será apresentada posteriormente.
A criação de um certificado para um cliente é igual à criação de um certificado para um servidor. No entanto, porque a maioria dos browsers só aceita certificados no formato pkcs12, sendo necessário converter o certificado. Para converter o certificado para o formato pkcs12 utiliza-se o comando:

```
openssl pkcs12 -export -in new_cert.pem -
     inkey new_key.pem -out new_pkcs12.p12
```

O ficheiro *new_pkcs12.p12* será o ficheiro importado pelo *browser*.
O Exemplo seguinte mostra todos os passos referidos anteriormente sobre a criação de um certificado para servidor ou para cliente, não sendo necessário executar o último comando para a criação de um certificado para um servidor.

```
# create a new certificate request valid
# for 1 year
openssl req -new -keyout new_req.pem \
    -out new_req.pem 3650

# sign a certificate request
openssl ca -policy policy_anything \
    -out new_cert.pem \
    -infiles new_req.pem

# extract the private key from the
# certificate
openssl rsa < newreq.pem > new_key.pem

# convert the certificate to pkcs12 file
# (suported by browsers)
openssl pkcs12 -export \
    -in new_cert.pem \
    -inkey new_key.pem \
    -out new_pkcs12.p12
```

Exemplo 5 – Criação de um certificado

5. Configuração do Apache

O ficheiro de configuração principal do Apache é o *httpd.conf*, no entanto a versão 2 contem as instruções para a introdução das directivas ssl no ficheiro *ssl.conf*. Estas poderão ser introduzidas em qualquer um dos ficheiros.
O exemplo 6 mostra as directivas necessárias para configurar o Apache a aceitar ligações ssl e autenticar clientes através de certificados.

```
# SSL Engine Switch: Enable/Disable
# SSL for this virtual host.
SSLEngine on
# don't use SSlv2, as it has
# fundamental design problems
SSLProtocol all -SSLv2
# all ciphers using strong encryption
# (Triple-DES)
SSLCipherSuite HIGH
# Client Authentication (Type):
# Client certificate verification type
# and depth. Types are
# none, optional, require and
# optional_no_ca. Depth is a
# number which specifies how deeply to
# verify the certificate issuer chain
# before deciding the certificate is
# not valid. (depth of 1 means the
# client certificate has to be
# signed by a CA which is directly
# known to the server
SSLVerifyClient optional
SSLVerifyDepth 10
# Certificate Authority (CA):
# Set the CA certificate verification
# path where to find CA
# certificates for client
# authentication or alternatively one
# huge file containing all of them
# (file must be PEM encoded)
SSLCACertificateFile \
     /etc/pki/CA/cacert.pem
# Server Certificate:
# Point SSLCertificateFile at a PEM
# encoded certificate. If the
# certificate is encrypted, then you
# will be prompted for a pass phrase.
SSLCertificateFile \
    /etc/pki/tls/certs/server_cert.pem
# Server Private Key: If the key is
# not combined with the certificate,
# use this directive to point at the
# key file.
SSLCertificateKeyFile \
   /etc/pki/tls/private/server_key.pem
# Location of the Revocation List
#SSLCARevocationFile \
     /etc/pki/CA/crl/cacrl.crl

# Export environment variables for php
SSLOptions +StdEnvVars
```

Exemplo 6 – Directivas do Apache para configuração do ssl

Neste exemplo são apresentadas várias directivas. As especificas relativas à aceitação e validação dos certificados dos clientes são apresentadas no exemplo 7.

```
# Client Authentication (Type):
# Client certificate verification type
# and depth.
SSLVerifyClient optional
SSLVerifyDepth 10
# Certificate Authority (CA
SSLCACertificateFile \
     /etc/pki/CA/cacert.pem
# Location of the Revocation List
SSLCARevocationFile \
     /etc/pki/CA/crl/cacrl.crl
# Export environment variables for php
SSLOptions +StdEnvVars
```

Exemplo 7 – Directivas do Apache para autenticação de clientes através de certificados

Como se pode verificar neste exemplo as directivas `SSLVerifyClient` e `SSLVerifyDepth` definem como será efectuada a verificação e se esta será efectuada. A primeira directiva define se serão aceites os certificados dos clientes e a segunda até que profundidade, em termos de CA, essa verificação será efectuada.
Segue-se a directiva que informa o Apache onde poderá consultar os certificados das CAs. Nesta apenas é introduzido o nome de um ficheiro contendo um ou vários certificados de CAs concatenados. Existe outra opção que poderá ser utilizada em substituição ou em conjunto com esta, a `SSLCACertificatePath`. Esta permite introduzir o directório onde estão os certificados de todas as CAs aceites pelo servidor.
A directiva `SSLCARevocationFile` indica o ficheiro contendo os certificados revogados. A directiva `SSLCARevocationPath` permite introduzir o directório onde se encontram os certificados revogados.
Por fim temos a directiva `SSLOptions`, esta irá permitir exportar as variáveis de ambiente para o PHP, permitindo que através do PHP se possa aceder à informação referente ao certificado do cliente. Desta forma será possível validar através do PHP se o utilizador é válido ou não, que tipo de utilizador é ou que permissões tem.

6. Base de Dados

Para exemplo, apenas foi construída uma tabela, a TAB_User. A função desta é armazenar os dados referentes ao utilizador. Como estamos a permitir dois tipos de autenticação, pelo par login/password e por certificados, a tabela irá conter a informação referente ao par login / password e dois campos referentes ao certificado. Os dados do certificado verificados são o CN (common name) e o endereço de correio electrónico.
O exemplo 8 apresenta os comandos sql utilizados para a criação da tabela em MySql.

```
CREATE DATABASE apacheauth;
USE apacheauth;
CREATE TABLE TAB_User (
      login varchar(25) not null,
      password varchar(50),
      cn varchar(255),
      email varchar(255),
      Primary Key(login)
);
```

Exemplo 8 – Criação da tabela TAB_User

7. Autenticação utilizando o PHP

Como exemplo, e utilizando a tabela referida anteriormente, construiu-se um *script* em PHP que permite a autenticação do utilizador através dum *login* e *password* ou através de um certificado. Como o Apache controla a validação do certificado, ao nível do PHP a única operação necessária é a verificação do utilizador na base de dados. Este *script* é apresentado no exemplo 9.

```
<?php
if ($_SERVER['SSL_CLIENT_VERIFY'] !=
     "SUCCESS") {
   // Not using a client certificate
   if (isset($_POST['login']) &&
        isset($_POST['password'])) {
      // verify login
      $db = mysql_connect("localhost",
         "root", 'pass');
      mysql_select_db("apacheauth",$db);
      $result = mysql_query("SELECT *" .
         " FROM TAB_User where login =".
         " '".$_POST['login']."' and ".
         " password ='".
         $_POST['password']."'",$db);
      if($myrow =
           mysql_fetch_row($result)) {
         include("loginOk.html");
         exit();
      }
   }
} else { //client using certificate
   $name =
      $_SERVER['SSL_CLIENT_S_DN_CN'];
   $email =
      $_SERVER['SSL_CLIENT_S_DN_Email'];
   if ($name != "" && $email != "") {
      $db = mysql_connect("localhost",
         "root", 'pass');
      mysql_select_db("apacheauth",$db);
      $result = mysql_query("SELECT *".
         " FROM TAB_User where cn ='".
         $name."' and email = '".
         $email."'",$db);
      if($myrow =
           mysql_fetch_row($result)) {
         include("loginOk.html");
         exit();
      }
   }
}
// if not a valid login or certificate
include("login.html");
?>
```

Exemplo 9 – Script em PHP para autenticar utilizadores

Como se pode verificar, a variável `$_SERVER['SSL_CLIENT_VERIFY']` permite controlar se o cliente submeteu um certificado ou não para autenticar-se. Caso tenha submetido e este seja válido, esta variável terá o valor `SUCCESS`. Sendo este o caso temos de verificar se o utilizador existe na base de dados. A verificação é efectuada através de uma consulta à base de dados com o CN e o email definidos no certificado.
Estes valores são obtidos através das variáveis `$_SERVER['SSL_CLIENT_S_DN_CN']` e `$_SERVER['SSL_CLIENT_S_DN_Email']`, respectivamente.
Caso não exista certificado será mostrada a página de login permitindo ao utilizador aceder ao site através do *login* e da *passowrd*.
No exemplo 7 apenas utilizamos o *email address* e o *common name* para verificar se o utilizador é válido. No entanto outros campos poderão ser utilizados. A tabela 1 apresenta a lista de variáveis existentes no PHP.

[SSL_CIPHER]
[SSL_CIPHER_ALGKEYSIZE]
[SSL_CIPHER_EXPORT]
[SSL_CIPHER_USEKEYSIZE]
[SSL_CLIENT_A_KEY]
[SSL_CLIENT_A_SIG]
[SSL_CLIENT_I_DN]
[SSL_CLIENT_I_DN_C]
[SSL_CLIENT_I_DN_CN]
[SSL_CLIENT_I_DN_Email]
[SSL_CLIENT_I_DN_L]
[SSL_CLIENT_I_DN_O]
[SSL_CLIENT_I_DN_OU]
[SSL_CLIENT_I_DN_ST]
[SSL_CLIENT_M_SERIAL]
[SSL_CLIENT_M_VERSION]
[SSL_CLIENT_S_DN]
[SSL_CLIENT_S_DN_C]
[SSL_CLIENT_S_DN_CN]
[SSL_CLIENT_S_DN_Email]
[SSL_CLIENT_S_DN_L]
[SSL_CLIENT_S_DN_O]
[SSL_CLIENT_S_DN_OU]
[SSL_CLIENT_S_DN_ST]
[SSL_CLIENT_V_END]
[SSL_CLIENT_V_START]
[SSL_CLIENT_VERIFY]
[SSL_PROTOCOL]

Tabela 1 – Variáveis SSL importadas para o PHP

8. Manual de instalação de certificados num Web Browser

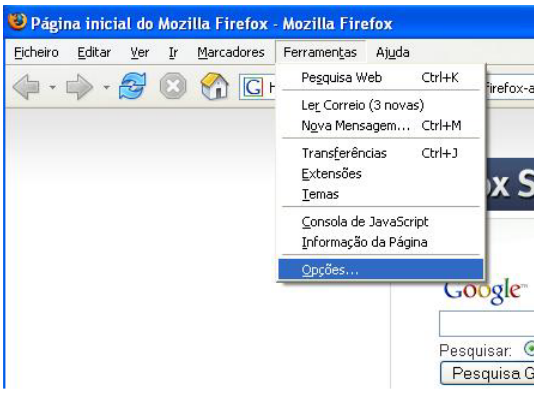
Serve o presente manual para elucidar o utilizador sobre o processo de instalação de um certificado digital num Web Browser. Na demonstração aqui efectuada utilizamos o Web Browser da Mozzila Firefox.

Após o utilizador ser detentor de um certificado digital assinado por uma *Certification Authority* (CA), o mesmo pode instalá-lo no seu Web Browser para proceder

à sua utilização aquando de comunicações seguras com um servidor HTTP através do protocolo HTTPS.

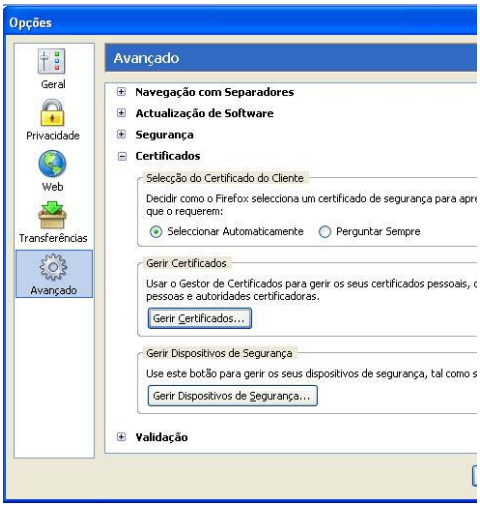
O processo de instalação é efectuado através dos seguintes passos:

1. Abrir o Web Browser Firefox.
2. Prima o botão "Ferramentas" da barra de "Menu" e seleccione "Opções…".

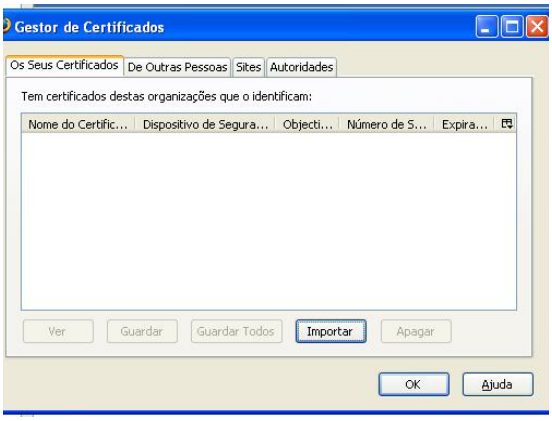


3. Seleccionar a opção "Avançado" no lado esquerdo da janela de opções.

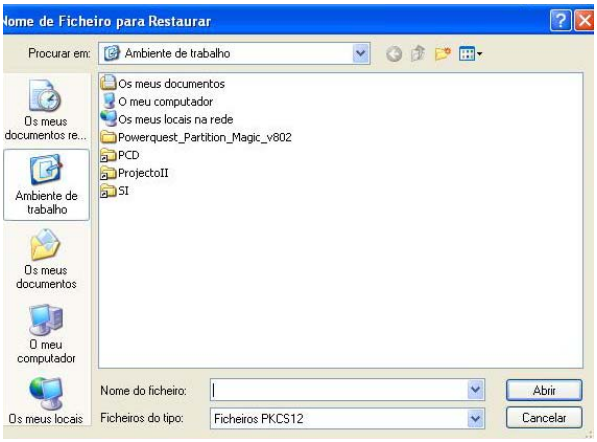
4. Expanda o nó "Certificados".



5. Na secção "Gerir Certificados", prima o botão "Gerir certificados…".



6. No separador "Os seus certificados" prima o botão "Importar" onde aparecerá uma janela de diálogo para seleccionar o certificado a importar.



7. Navegue até à localização do certificado a importar.
8. Seleccione-o e prima o botão "Abrir". Se o certificado estiver protegido por palavra-chave, aparecerá uma janela de diálogo para inserção da palavra-chave.
9. Insira a palavra-chave e prima "Ok". O certificado será importado para o Firefox.
10. A fim de confirmar a correcta importação do certificado, localize o nome do certificado que acabou de importar no separador "Os seus certificados".

9. Mais Informações

a. Apontamentos Teóricos da disciplina de Segurança de Informação, DEI ESTG Leiria.
b. *William Stallings*, "Cryptography and Network Security: Principles and Practice", Prentice Hall, 1999.
c. www.apache.org.
d. www.php.net
e. www.openssl.org
f. www.mysql.com