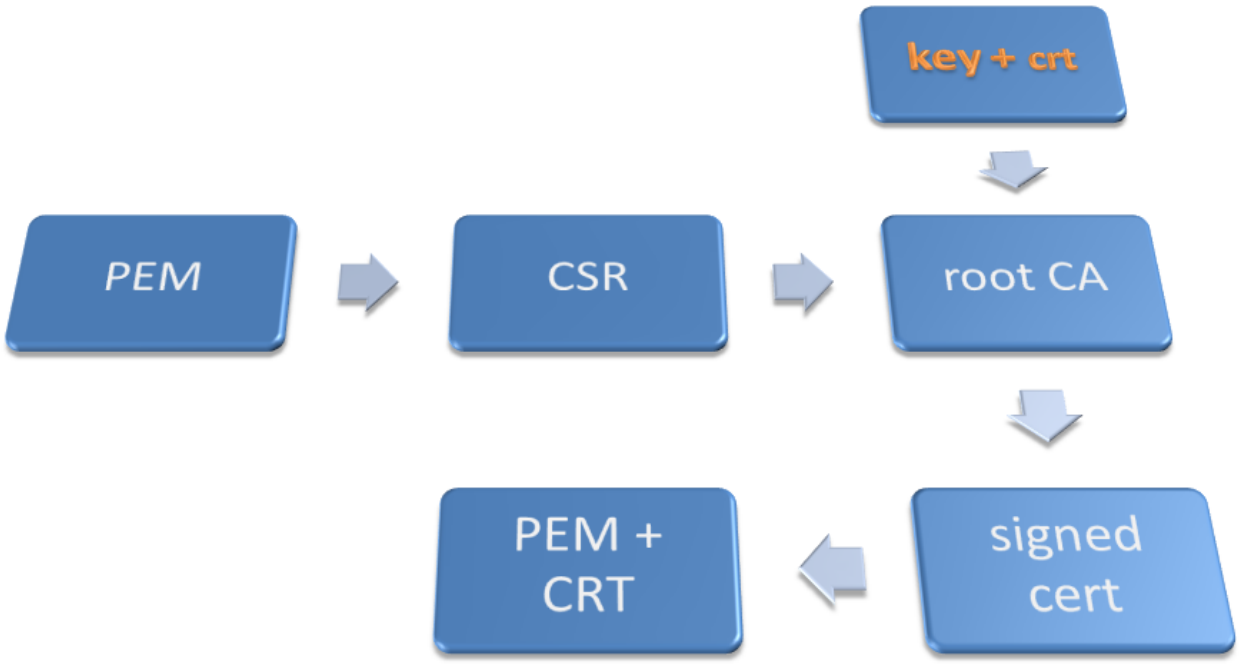
PKI / SSL

written by archi | 17 października 2019

Generowanie Certificate Authority



(Uwaga: w przypadku braku polecenia openssl należy zainstalować pakiet o takiej samej nazwie)

W pierwszej kolejności generujemy 4096-bit długi klucz RSA dla naszego centrum (root CA) i zapisujemy go w pliku ca.key:

```
openssl genrsa -out ca.key 4096
```

LUB dodając opcję –des3 możemy klucz zabezpieczyć hasłem:

```
openssl genrsa –des3 -out ca.key 4096
```

Następnie tworzymy własny samodzielnie podpisany przez root CA certyfikat o nazwie ca.crt (konieczne będzie wprowadzenie identyfikacji naszej organizacji). –x509 używane jest dla certyfikatów z własnym podpisem (self-signed)

Kiedy system zapyta o dane dotyczace właściciela certyfikatu proszę postępować zgodnie z podpowiedziami. Gdzie trzeba to proszę podawać wyłącznie duże litery (np.: country PL), w innych przypadkach dowolnie. Pole CN Common Name powinno być ustawione na adres IP maszyny wirtualnej!

```
openssl req -new -x509 -days 3650 -key ca.key -out ca.crt
```

Utworzyliśmy centrum certyfikacji. To centrum może podpisywać dowolne certyfikaty przyszłych klientów.

Generowanie certyfikatu usługi / klienta (np. dla Apache2 danego hosta)

Tworzymy klucz dla usługi (w tym przypadku q.pem):

```
openssl genrsa -out q.pem 2048
```

Kolejnym etapem jest generowanie żądania podpisu. Tu też wskazujemy właściciela certyfikatu. Nie może być to ta sama instytucja bo jest to niedozwolone podpisać samemu sobie. Należy podać odpowiednie organizacje i podorganizacje oraz tak jak wcześniej jako CN (Common Name)

podać adres IP maszyny wirtualnej.

```
openssl req -new -key q.pem -out q.csr
```

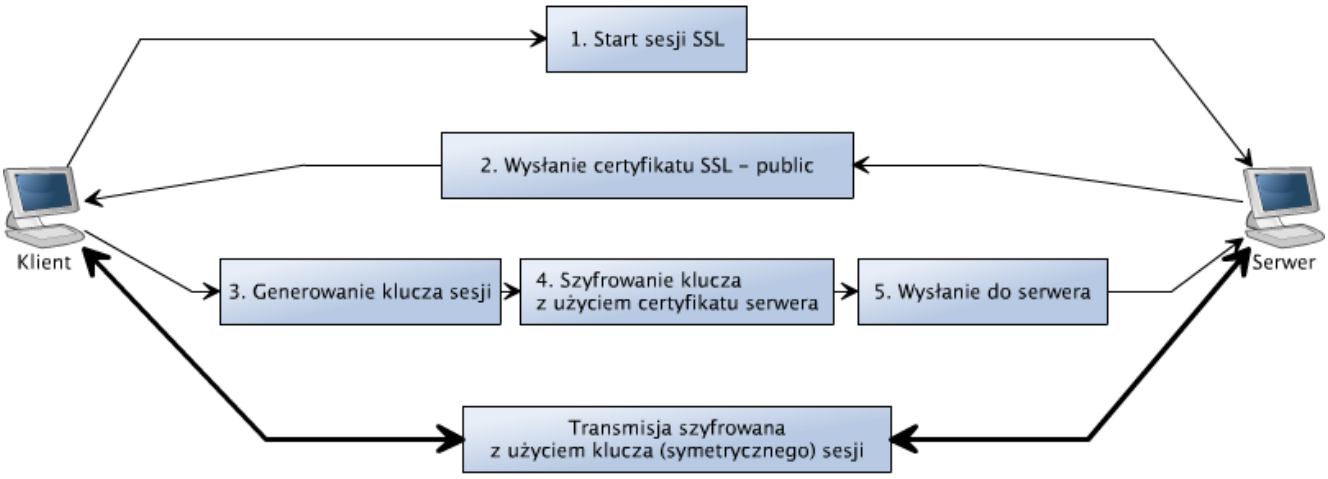
Jako CA podpisujemy przygotowane żądanie i tworzymy q-cert.pem (podpis) z podpisem na 10 lat oraz generowanie numeru seryjnego dla wykonanego podpisu.

```
openssl x509 -req -in q.csr -out q-cert.pem -sha256 -CA ca.crt -CAkey ca.key -CAcreateserial -days 3650
```

Otrzymany podpis sklejamy z generowanym kluczem w jeden plik zawierajacy certyfikat usługi i podpis CA tego certyfikatu.

```
cat q-cert.pem >> q.pem
```

Algorytm działania połączeń HTTPS



Dodawanie PEM do usług

„Apache2”

Przykładowy plik z zawartością SSL dla usługi „`/etc/apache2/sites-available/default-ssl`” (***na niebiesko oznaczono konieczne zmiany w pliku***):

```
ServerAdmin webmaster@localhost
DocumentRoot /var/www

Options FollowSymLinks
AllowOverride None


Options Indexes FollowSymLinks MultiViews
AllowOverride None
Order allow,deny
allow from all

ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/

AllowOverride None
Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
Order allow,deny
Allow from all

ErrorLog /var/log/apache2/error.log


# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.

LogLevel warn

CustomLog /var/log/apache2/ssl_access.log combined
```

```
Alias /doc/ "/usr/share/doc/"

Options Indexes MultiViews FollowSymLinks
AllowOverride None
Order deny,allow
Deny from all
Allow from 127.0.0.0/255.0.0.0 ::1/128


# SSL Engine Switch:
# Enable/Disable SSL for this virtual host.
SSLEngine on

# A self-signed (snakeoil) certificate can be created by
installing
# the ssl-cert package. See
# /usr/share/doc/apache2.2-common/README.Debian.gz for more info.
# If both key and certificate are stored in the same file, only
the
# SSLCertificateFile directive is needed.
SSLCertificateFile /etc/apache2/q.pem
# SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key

# Server Certificate Chain:
# Point SSLCertificateChainFile at a file containing the
# concatenation of PEM encoded CA certificates which form the
# certificate chain for the server certificate. Alternatively
# the referenced file can be the same as SSLCertificateFile
# when the CA certificates are directly appended to the server
# certificate for convinience.
#SSLCertificateChainFile /etc/apache2/ssl.crt/server-ca.crt

# 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)
# Note: Inside SSLCACertificatePath you need hash symlinks
# to point to the certificate files. Use the provided

# Makefile to update the hash symlinks after changes.
```

```
#SSLCACertificatePath /etc/ssl/certs/
#SSLCACertificateFile /etc/apache2/ssl.crt/ca-bundle.crt

# Certificate Revocation Lists (CRL):
# Set the CA revocation path where to find CA CRLs for client
# authentication or alternatively one huge file containing all
# of them (file must be PEM encoded)
# Note: Inside SSLCARevocationPath you need hash symlinks
# to point to the certificate files. Use the provided
# Makefile to update the hash symlinks after changes.
#SSLCARevocationPath /etc/apache2/ssl.crl/
#SSLCARevocationFile /etc/apache2/ssl.crl/ca-bundle.crl

# 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.
#SSLVerifyClient require
#SSLVerifyDepth 10

# Access Control:
# With SSLRequire you can do per-directory access control based
# on arbitrary complex boolean expressions containing server
# variable checks and other lookup directives. The syntax is a
# mixture between C and Perl. See the mod_ssl documentation
# for more details.
#
#SSLRequire ( %{SSL_CIPHER} !~ m/^(EXP|NULL)/ \
# and %{SSL_CLIENT_S_DN_O} eq "Snake Oil, Ltd." \
# and %{SSL_CLIENT_S_DN_OU} in {"Staff", "CA", "Dev"} \
# and %{TIME_WDAY} >= 1 and %{TIME_WDAY} <= 5 \
# and %{TIME_HOUR} >= 8 and %{TIME_HOUR} <= 20 ) \
# or %{REMOTE_ADDR} =~ m/^192\.76\.162\.[0-9]+$/

# SSL Engine Options:
# Set various options for the SSL engine.
# o FakeBasicAuth:
# Translate the client X.509 into a Basic Authorisation. This
means that
```

```
# the standard Auth/DBMAuth methods can be used for access
control. The
# user name is the 'one line' version of the client's X.509
certificate.
# Note that no password is obtained from the user. Every entry in
the user
# file needs this password: 'xxj31ZMTZzkVA'.
# o ExportCertData:
# This exports two additional environment variables:
SSL_CLIENT_CERT and
# SSL_SERVER_CERT. These contain the PEM-encoded certificates of
the
# server (always existing) and the client (only existing when
client
# authentication is used). This can be used to import the
certificates
# into CGI scripts.
# o StdEnvVars:
# This exports the standard SSL/TLS related 'SSL_*' environment
variables.
# Per default this exportation is switched off for performance
reasons,
# because the extraction step is an expensive operation and is
usually
# useless for serving static content. So one usually enables the
# exportation for CGI and SSI requests only.
# o StrictRequire:
# This denies access when "SSLRequireSSL" or "SSLRequire" applied
even
# under a "Satisfy any" situation, i.e. when it applies access is
denied
# and no other module can change it.
# o OptRenegotiate:
# This enables optimized SSL connection renegotiation handling
when SSL
# directives are used in per-directory context.
#SSLOptions +FakeBasicAuth +ExportCertData +StrictRequire

SSLOptions +StdEnvVars
```

```
SSLOptions +StdEnvVars

# SSL Protocol Adjustments:
# The safe and default but still SSL/TLS standard compliant
shutdown
# approach is that mod_ssl sends the close notify alert but
doesn't wait for
# the close notify alert from client. When you need a different
shutdown
# approach you can use one of the following variables:
# o ssl-unclean-shutdown:
# This forces an unclean shutdown when the connection is closed,
i.e. no
# SSL close notify alert is send or allowed to received. This
violates
# the SSL/TLS standard but is needed for some brain-dead browsers.
Use
# this when you receive I/O errors because of the standard
approach where
# mod_ssl sends the close notify alert.
# o ssl-accurate-shutdown:
# This forces an accurate shutdown when the connection is closed,
i.e. a
# SSL close notify alert is send and mod_ssl waits for the close
notify
# alert of the client. This is 100% SSL/TLS standard compliant,
but in
# practice often causes hanging connections with brain-dead
browsers. Use
# this only for browsers where you know that their SSL
implementation
# works correctly.
# Notice: Most problems of broken clients are also related to the
HTTP
# keep-alive facility, so you usually additionally want to disable
# keep-alive for those clients, too. Use variable "nokeepalive"
for this.
# Similarly, one has to force some clients to use HTTP/1.0 to
workaround
# their broken HTTP/1.1 implementation. Use variables
```

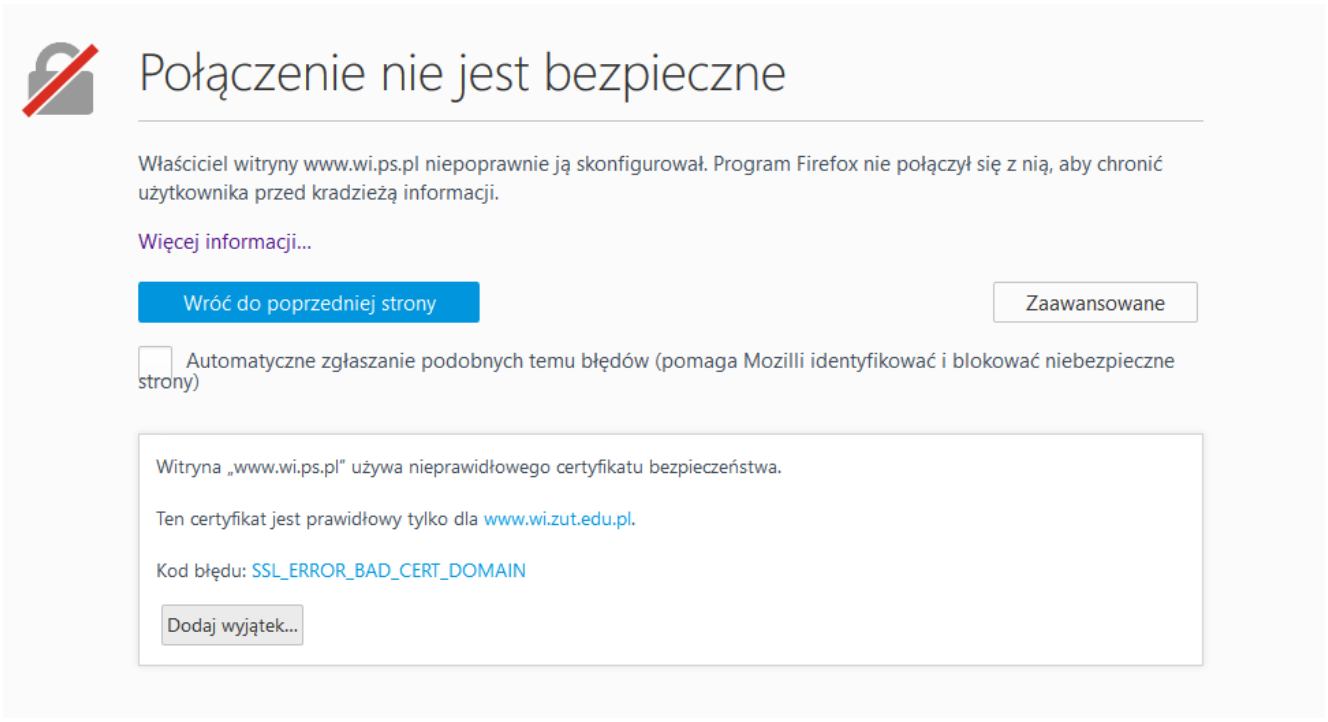
```
"downgrade-1.0" and
# "force-response-1.0" for this.
BrowserMatch "MSIE [2-6]" \
nokeepalive ssl-unclean-shutdown \
downgrade-1.0 force-response-1.0
# MSIE 7 and newer should be able to use keepalive
BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown
```

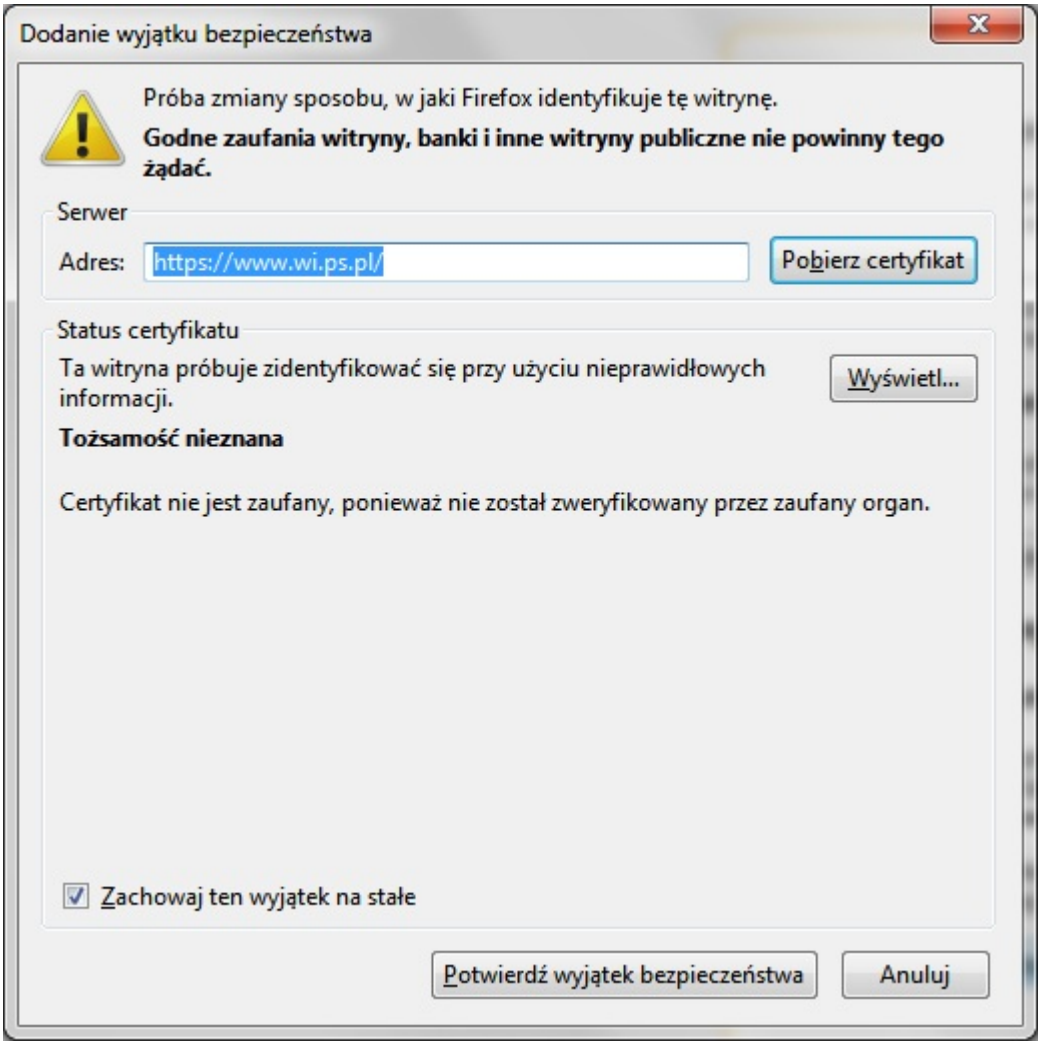
Po naniesieniu zmian należy:

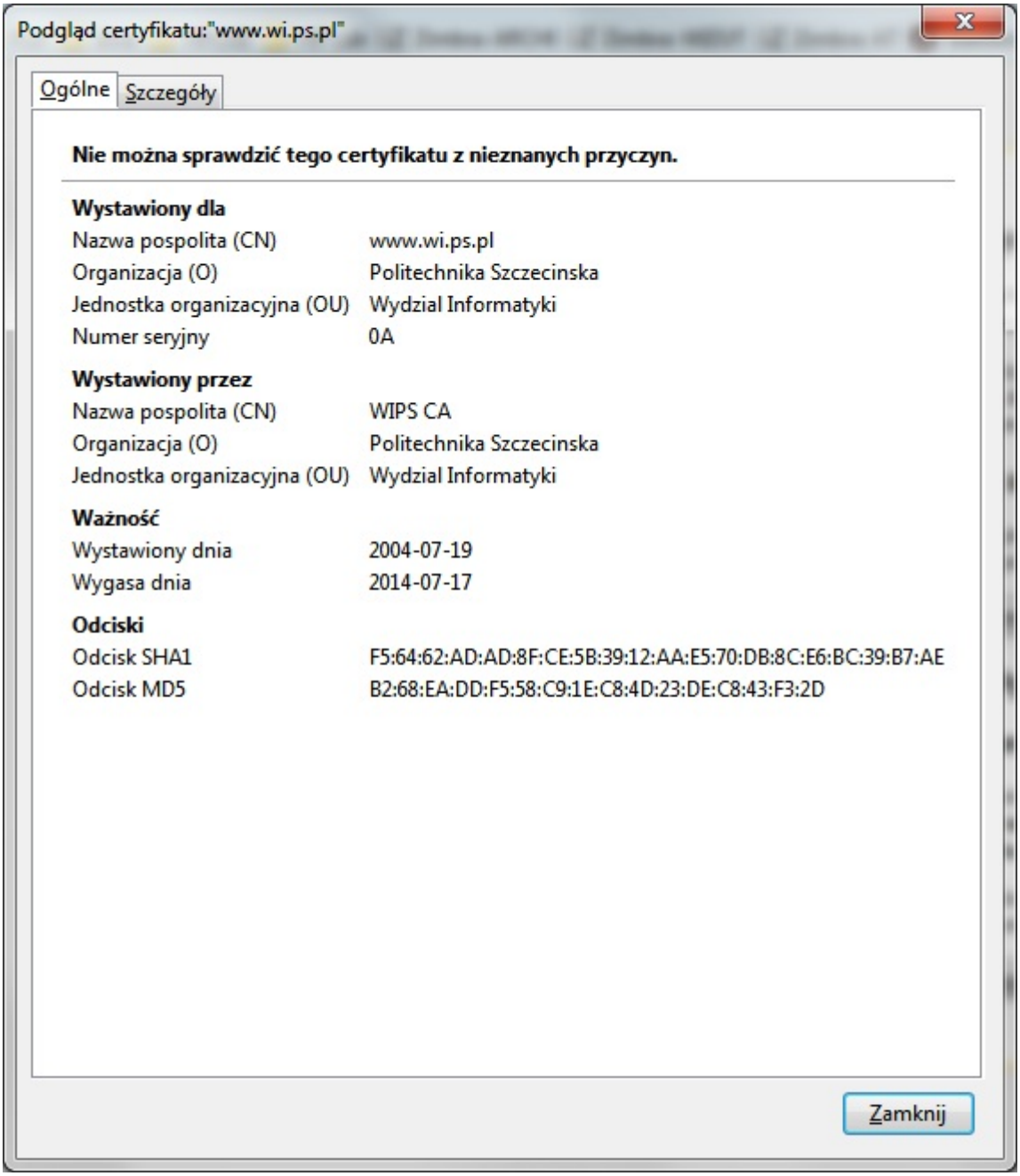
1. Przekopiować plik wynikowy po operacji tworzenia certyfikatu „`q.pem`” do katalogu apache wskazanego w pliku konfiguracyjnym. W tym przypadku „`/etc/apache2`„
2. Włączyć moduł SSL dla usługi apache poleceniem:
   `a2enmod ssl`
3. Włączyć strukturę dodatkową dla apache w drugim pliku konfiguracyjnym opisanym powyżej „default-ssl” poleceniem:
   `a2ensite default-ssl`
4. Zrestartować usługę apache poleceniem:
   `/etc/init.d/apache2 restart`
   lub poleceniem:
   `service apache2 restart`

Testowanie:

1. Połącz się przy pomocy przeglądarki z wykorzystaniem HTTPS do swojego serwera
   https://192.168.200.xxx lub https://192.168.2.xxx
2. Jeżeli nie wystąpi błąd certyfikacji to powinny się wyświetlić podobne okienka z Twoimi informacjami jak poniżej


Połączenie nie jest bezpieczne
Właściciel witryny www.wi.ps.pl niepoprawnie ją skonfigurował. Program Firefox nie połączył się z nią, aby chronić użytkownika przed kradzieżą informacji.
Więcej informacji...
Wróć do poprzedniej strony
Zaawansowane
Automatyczne zgłaszanie podobnych temu błędów (pomaga Mozilli identyfikować i blokować niebezpieczne strony)
Witryna „www.wi.ps.pl" używa nieprawidłowego certyfikatu bezpieczeństwa.
Ten certyfikat jest prawidłowy tylko dla www.wi.zut.edu.pl.
Kod błędu: SSL_ERROR_BAD_CERT_DOMAIN
Dodaj wyjątek...


Dodanie wyjątku bezpieczeństwa
Próba zmiany sposobu, w jaki Firefox identyfikuje tę witrynę.
Godne zaufania witryny, banki i inne witryny publiczne nie powinny tego żądać.
Serwer
Adres: https://www.wi.ps.pl/
Pobierz certyfikat
Status certyfikatu
Ta witryna próbuje zidentyfikować się przy użyciu nieprawidłowych informacji.
Wyświetl...
Tożsamość nieznana
Certyfikat nie jest zaufany, ponieważ nie został zweryfikowany przez zaufany organ.
Zachowaj ten wyjątek na stałe
Potwierdź wyjątek bezpieczeństwa
Anuluj


Podgląd certyfikatu:"www.wi.ps.pl"
Ogólne
Szczegóły
Nie można sprawdzić tego certyfikatu z nieznanych przyczyn.
Wystawiony dla
Nazwa pospolita (CN) www.wi.ps.pl
Organizacja (O) Politechnika Szczecinska
Jednostka organizacyjna (OU) Wydzial Informatyki
Numer seryjny 0A
Wystawiony przez
Nazwa pospolita (CN) WIPS CA
Organizacja (O) Politechnika Szczecinska
Jednostka organizacyjna (OU) Wydzial Informatyki
Ważność
Wystawiony dnia 2004-07-19
Wygasa dnia 2014-07-17
Odciski
Odcisk SHA1 F5:64:62:AD:AD:8F:CE:5B:39:12:AA:E5:70:DB:8C:E6:BC:39:B7:AE
Odcisk MD5 B2:68:EA:DD:F5:58:C9:1E:C8:4D:23:DE:C8:43:F3:2D
Zamknij