Certificate Management

Document revision 2.3 (Fri Mar 05 13:58:17 GMT 2004)
This document applies to V2.8

Table of Contents

General Information

Summary

SSL (Secure Socket Layer) is a security technology to ensure encrypted transactions over a public network. To protect the data, an encryption key should be negotiated. SSL protocol is using Certificates to negotiate a key for data encryption.

Specifications

Packages required: ***system***
License required: ***level1***
Home menu level: ***/certificate***
Standards and Technologies: ***SSLv2***, ***SSLv3***, ***TLS***
Hardware usage: *high CPU usage*

Description

SSL technology was first introduced by Netscape to ensure secure transactions between browsers and web servers. When a browser requests a secure web page (usually on TCP port 443), a web server first sends a Certificate, which contains a public key for the encryption key negotiation to take place. After the encryption key is negotiated, the web server will send the requested page encrypted using this key to the browser (and also the browser will be able to submit its data securely to the server)

SSL Certificate confirms the web server identity. The Certificate contains information about its holder (like DNS name and Country), issuer (the entity has signed the Certificate) and also the public key used to negotiate the encryption key. In order a Certificate to play its role, it should be signed by a third party (Certificate Authority) which both parties trust. Modern browsers that support SSL protocol have a list of the Certificate Authorities they trust (the most known and trusted CA is VeriSign, but that is not the only one)

To use a Certificate (which contain a public key), server needs a private key. One of the keys is used for encryption, and the other - for decryption. It is important to understand, that both keys can encrypt and decrypt, but what is encrypted by one of them can be decrypted **only** by the another. Private key must be kept securely, so that nobody else can get it and use this certificate. Usually private key is encrypted with a passphrase.

Most trusted Certificate Authorities sell the service of signing Certificates (Certificates also have a finite validity term, so you will have to pay regularly). It is also possible to create a self-signed Certificate (you can create one on most UNIX/Linux boxes using openssl toolkit; all Root Certificate Authorities have self-signed Certificates), but if it is not present in a browser's database, the browser will pop up a security warning, saying that the Certificate is not trusted (note also that most browsers support importing custom Certificates to their databases).

Certificates

Home menu level: ***/certificate***

Description

MikroTik RouterOS can import Certificates for the SSL services it provides (only HotSpot for now). This submenu is used to manage Certificates for this services.

Property Description

name (*name*) - reference name

subject (*read-only: text*) - holder (subject) of the certificate

issuer (*read-only: text*) - issuer of the certificate

serial-number (*read-only: text*) - serial number of the certificate

invalid-before (*read-only: date*) - date the certificate is valid from

invalid-after (*read-only: date*) - date the certificate is valid until

ca (yes | no; default: **yes**) - whether the certificate is used for building or verifying certificate chains (as Certificate Authority)

Command Description

import - install new certificates

- **file-name** - import only this file (all files are searched for certificates by default)
- **passphrase** - passphrase for the found encrypted private key
- **certificates-imported** - how many new certificates were successfully imported
- **private-keys-imported** - how many private keys for existing certificates were successfully imported
- **files-imported** - how many files contained at least one item that was successfully imported
- **decryption-failures** - how many files could not be decrypted
- **keys-with-no-certificate** - how many public keys were successfully decrypted, but did not have matching certificate already installed

reset-certificate-cache - delete all cached decrypted public keys and rebuild the certificate cache

decrypt - decrypt and cache public keys

- **passphrase** - passphrase for the found encrypted private key
- **keys-decrypted** - how many keys were successfully decrypted and cached

create-certificate-request - creates an RSA certificate request to be signed by a Certificate Authority. After this, download both private key and certificate request files from the router. When you receive your signed certificate from the CA, upload it and the private key (that is made by this command) to a router and use /certificate import command to install it

- **certificate request file name** - name for the certificate request file (if it already exists, it will be overwritten). This is the original certificate that will be signed by the Certificate Authority
- **file name** - name of private key file. If such file does not exist, it will be created during the next step. Private key is used to encrypt the certificate
- **passphrase** - the passphrase that will be used to encrypt generated private key file. You must enter it twice to be sure you have not made any typing errors
- **rsa key bits** - number of bits for RSA (encryption) key. Longer keys take more time to generate. 4096 bit key takes about 30 seconds on Celeron 800 system to generate
- **country name** - (C) ISO two-character country code (e.g., LV for Latvia)
- **state or province name** - (ST) full name of state or province
- **locality name** - (L) locality (e.g. city) name
- **organization name** - (O) name of the organization or company
- **organization unit name** - (OU) organization unit name
- **common name** - (CN) the server's common name. For SSL web servers this must be the fully qualified domain name (FQDN) of the server that will use this certificate (like www.example.com). This is checked by web browsers
- **email address** - (Email) e-mail address of the person responsible for the certificate
- **challenge password** - the challenge password. It's use depends on your CA. It may be used to revoke this certificate
- **unstructured address** - unstructured address (like street address). Enter only if your CA accepts or requires it

Notes

Server certificates may have **ca** property set to **no**, but Certificate Authority certificates must have it set to **yes**

Certificates and encrypted private keys are imported from and exported to the router's FTP server. Public keys are not stored on a router in unencrypted form. Cached decrypted private keys are stored in encrypted form, using key that is derived from the router ID. Passphrases are not stored on router.

Configuration backup does not include cached decrypted private keys. After restoring backup all certificates with private keys must be decrypted again, using **decrypt** command with the correct passphrase.

No other certificate operations are possible while generating a key.

When making a certificate request, you may leave some of the fields empty. CA may reject your certificate request if some of these values are incorrect or missing, so please check what are the requirements of your

CA

Example

To import a certificate and the respective private key already uploaded on the router:

```
[admin@MikroTik] certificate> import
passphrase: xxxx
       certificates-imported: 1
       private-keys-imported: 1
              files-imported: 2
         decryption-failures: 0
    keys-with-no-certificate: 1
[admin@MikroTik] certificate> print
Flags: K - decrypted-private-key, Q - private-key, R - rsa, D - dsa
  0 QR name="cert1" subject=C=LV,ST=.,O=.,CN=cert.test.mt.lv
       issuer=C=LV,ST=.,O=.,CN=third serial-number="01"
       invalid-before=sep/17/2003 11:56:19 invalid-after=sep/16/2004 11:56:19
       ca=yes

[admin@MikroTik] certificate> decrypt
passphrase: xxxx
    keys-decrypted: 1
[admin@MikroTik] certificate> print
Flags: K - decrypted-private-key, Q - private-key, R - rsa, D - dsa
  0 KR name="cert1" subject=C=LV,ST=.,O=.,CN=cert.test.mt.lv
       issuer=C=LV,ST=.,O=.,CN=third serial-number="01"
       invalid-before=sep/17/2003 11:56:19 invalid-after=sep/16/2004 11:56:19
       ca=yes

[admin@MikroTik] certificate>
```

Now the certificate may be used by HotSpot servlet:

```
[admin@MikroTik] ip service> print
Flags: X - disabled, I - invalid
  #   NAME                                PORT  ADDRESS            CERTIFICATE
  0   telnet                              23    0.0.0.0/0
  1   ftp                                 21    0.0.0.0/0
  2   www                                 8081  0.0.0.0/0
  3   hotspot                             80    0.0.0.0/0
  4   ssh                                 22    0.0.0.0/0
  5   hotspot-ssl                         443   0.0.0.0/0          none

[admin@MikroTik] ip service> set hotspot-ssl certificate=
cert1  none
[admin@MikroTik] ip service> set hotspot-ssl certificate=cert1
[admin@MikroTik] ip service> print
Flags: X - disabled, I - invalid
  #   NAME                                PORT  ADDRESS            CERTIFICATE
  0   telnet                              23    0.0.0.0/0
  1   ftp                                 21    0.0.0.0/0
  2   www                                 8081  0.0.0.0/0
  3   hotspot                             80    0.0.0.0/0
  4   ssh                                 22    0.0.0.0/0
  5   hotspot-ssl                         443   0.0.0.0/0          cert1

[admin@MikroTik] ip service>
```