Plug and Play CA

Certification Authority which just works
User and Administrator Documentation

Aleksander Nowiński

Spis treści

Introduction

Welcome to Plug-and-Play Certification Authority documentation. We hope you will find this application useful for your grid or network

What is PnP CA

PnP CA is a stand alone X509 Certification Authority application with web interface, that just works. It needs only java installed on dedicated system, with no need of configuring external web server, database or other stuff. Visit http://pnp-ca.sourceforge.net/ to get latest information and get latest version of software.

X509 certificates

X509 certificates are wide accepted standard for PKI (public-key infrastructure) cryptography. X509 Certificates are used for ensuring security in web communication (https), WIFI (WEP-2) or VPN systems. They also form modern grid security systems.

Basically X509 certificate is a standardized way to encode public key of a party (user, web server) together with basic information about the party (name, country) and information about key usage. Such information is encoded, verified and digitally signed by third-party trusted authority, called Certification Authority. Then owner of the private key matching public key in certificate may present a certificate to prove its identity.

Certification Authority is a key point of whole certificate system. CA has its own certificate, usually self-signed, which provides information about its public key. Then any user, which knows CA certificate may validate certificates presented by other parties by verification of digital signature. Such CA certificate is usually stored in so called 'trust store', i.e. security database, which defines trusted authorities.

There exist certain amount of commercially run Certification Authorities, which issue X509 Certificates, which are accepted for WWW sites and so on. But of course this require some money and is not suitable for certain situations. Running your own CA is simple, and with PnP-CA is even simpler now.

Especially security in grid environments (Unicore, Globus) usually base on concept of having own, dedicated CA for a grid. Running such web based CA for project members is very convenient.

Supported platforms

PnP-CA is a Web-interface based. You can run a server on a single machine, and access it with browser from any possible client machine. PnP-CA runs on all platforms correctly supporting Java. It has been developed and is used with Linux, but there is no reason not to run it on Windows, Mac OSX or Solaris. There is no startup support for windows (yet), but it is no problem – you can run it by the hand. It has been tested on Linux, and briefly on Windows Vista with sun java 6.

Supported standards

Currently PnP-CA supports plain X509 certificates with no v3 extensions. We work on adding proper support for v3 extensions soon.

Installation

Prequisities

To install the PnP CA you need only java 6 installed. You may get it from (http://java.com/) ***with unlimited strength cryptography policy files installed***. Sun java requires those files installed to allow using cryptography keys with reasonable strength, which is necessary to run CA. You may download necessary files from http://java.sun.com/ website. In case of running CA without those files remember to refer to “Clean data” section of configuration chapter of this document.

To build project from source you need *apache maven* build system installed (http://maven.apache.org/). You may checkout latest sources at sourceforge, visit http://pnp-ca.sourceforge.net/ to find download instructions.

Building installation package

If you just plan to use PnP-CA skip this step.

To build package unpack source archive, or checkout latest sources from sourceforge page, find the instructions on http://sourceforge.net/projects/pnp-ca/ .

Then enter to the directory with sources, and type:

```
mvn clean install
```

This will build all classes and install it in you local maven cache. Then to build installation package enter '`pnpca-server`' directory and run:

```
mvn assembly:assembly
```

This will build an installation package in '`target`' subdirectory named:

`pnpca-server-<`*`ver`*`>-dist.zip or pnpca-server-<`*`ver`*`>-dist.tar.gz`

This archive is an installation archive.

Installing from Installation archive

To install from installation archive just unpack it in desired location (for example `/opt`):

```
cd /opt

tar zxvf pnpca-server-<ver>-dist.tar.gz
```

A new directory will be created, `pnpca-server-<version>`, which contains complete, ready to run installation of the PnP-CA.

On Windows it is enough to unpack the installation package to a desired directory.

PnP-CA requires no additional privileges (unless you will try to run it on port below 1024, which is not recommended), and it is recommended to run it with non-administrative privileges to avoid security hazard. We made big effort to make it secure, but we are only human beings.

Running for first time

To run service invoke `bin/pnpca` script (on unix like systems). On windows double-click only jar file in main directory. This shall start the server. After run visit with your browser address: http://localhost:8080/ . If you will see PnP-CA server then congratulations – you have successfully installed PnP-CA. If not, look into `logs/` directory of your installation and see what is wrong (usually no java or port is in use).

You may stop the server with `bin/stop_pnpca` script, or by killing server process (on windows).

Now it is good moment to read 'configuration' chapter of this manual, to configure service before run. Next paragraphs of the installation manual describe installing configured PnP-CA as Unix server service, to make it run safe way at startup.

Advanced configuration topics

Running up at startup time

To run a PnP CA at startup time there is a System V init script prepared, `bin/pnpca.init`. If you are using redhat based distribution you may copy it into your `/etc/init.d` directory, edit to match your settings (update installation directory), and add to the init sequence. On fedora system this is done (if script is named `/etc/init.d/pnpca`) with command:

```
chkconfig --add pnpca
chkconfig pnpca on
```

This will make your system to run pnpca on startup.

Separate user account

It is recommended to run PnP-CA with separate user account, to reduce possibly security hazard. To do this, create a new user account (named for example pnp-ca) and update startup script to run the server with this account:

```
        daemon $PNPCA_DIR/bin/pnpca
#       daemon --user pnp-ca $PNPCA_DIR/bin/pnpca
```

This will make PnP-CA run on account pnp-ca.

Configuration

Configuration of the system is performed by editing some configuration files manually.

Directory Layout

Whole installation of the PnP CA is located in one directory, as in installation archive. In this directory you will find following subdirectories:

- `/bin` – directory with executable scripts and script utilities. Most important:
  - `pnpca` – starts server
  - `stop_pnpca` – stops the service
  - `clear_ca` – removes all CA keys/certificates, which makes system to create new crypto matherial at next system startup. Handle with care!
  - `pnpca.init` – script to be used as System V init script to make server run at system startup
- `/conf` – configuration file directory Most important files here:
  - `config.xml` – main system configuration file, see description below
  - `log4j.properties` – logging system configuration. See log4j configuration file syntax reference
  - `*.msg` – files used as message templates for email notification
- `/local` – local server files, which contains sensitive information: keystore, userMap etc.
- `/data` – server data storage directory. In subdirectories of this directory all certificates and certificate request are stored together with textual metadata.
- `/logs` – logs are stored here.
- `/webapp` – web server root directory
- `/docs` – documentation of the program
- `/lib` – libraries (java .jar files) are stored here

Configuring base system

Configuration of the system is defined by certain amount of files. Core configuration file is conf/config.xml. Administrator users and passwords are defined in local/userMap file.

Data cleanup

When some configuration parameters are changed, especially referring to cryptographic material and CA certificate, there is a need of purging old data. If CA certificate and keystore are not present, they are recreated at system startup. In such case, also all issued certificates are become invalid – they are signed with key and certificate which no longer exist.

Also if you accidentally start server on java with no unlimited strength cryptography policy files installed, it may happen that generated keystore will be corrupted. In such case you also need to clean CA before starting it again.

Therefore if you want to make changes in configuration parameters referring to CA key or CA certificate (`localDN`, whole `auth` section and others) you have to remove all obsolete data. To help you with this task you may use bin/clear_ca script, which takes care of removal all copies of CA certificate, issued certificates, keystore etc.

Be careful! Invoking this on CA which is used will effectively disable it, and make all issued certificates problematic. Never do this with production CA!

Main configuration file – config.xml

It is best idea to use predefined `config.xml` file as base for your configuration, by adopting it to your needs. Note, that although it is an XML file, it has no defined dtd, and options which are not understood are silently discarded.

Most important configuration options are described below:

- `<auth>` - defines cryptography parameters of the CA key used to sign certificates. Defaults are reasonable, it is recommended to change password.
- `<web>` - defines operation of web server. Please note, that some options from default are ignored (this is going to be properly updated in next revisions). Most important subelements are:
    - `<title>` - title that will appear as CA name in all web pages
    - `<admin>` - defines contact to CA administrator, as it will appear in footage of web pages
    - `<http>`, `<https>` - sections defining whether an plain http and secure http (https) are enabled, and which port shall be used. At least one of those two shall be enabled
    - `<ca_url>` - some modules (RSS feed) require to know a url at which application is being referred by the client. Place proper url of the CA here.
    - `<passwordFile>` - file where user map is stored. This is file in jetty HashUserRealm format. Refer to its documentation.
    - `<debug>` - set this to false in production environment. If this is true, then admin has one more menu option, which allows to halt server remotely
- `<mail>` - defines mail notification configuration. You need an SMTP server which does not require authentication for sending email (typically linux sendmail on localhost will do). Configure this according to your preferences. Remember to set `<enabled>` to true.
- `<ca>` - section containing definition of the core crypt service
    - `<localDN>` - DN of the CA certificate (used only at first start). If you want to change this you must clear all CA data (see Data Cleanup section)
    - `<signatureAlgorithm>` - algorithm used to sign certificates. It is recommended to use SHA1withRSA as this value. It is not recommended to use old MD5withRSA, as it has been prooved to be vulnerable.
    - `<notBefore>`, `<notAfter>` - validity of CA certificate set during certificate creation
- `<policy>` - defines policy on certificate request and issuing
    - `<dnRule>` - rules applied to certificate DN, among them:
        - `<required>` - defines required DN fields for certificates. CSRs which miss one or more of those fields will not be accepted

        - `<regexp>` - defines which regexp is applied to DN fields to ensure proper form (default is to accept only limited set of characters) subelements are:
            - `<element>` - field name
            - `<regexp>` - regexp which field has to match (in java regex format)
- `<certtypes>` - defines a list of certificate types which may be issued by CA. In future releases those certificates will differ with extensions, but this is not implemented yet.
- `<extensions>` contains definitions of the extensions for CA, which allow f.e. To run a program on certificate issuing or automatically sign an incoming requests

Note on the autosign extension

There is a very special extension which allows to sign CSRs automatically, without administrator intervention. If this extension is enabled, then any CSR submitted into CA will be signed instantly. This is suitable for demonstration CA and for other demonstration purposes.

Security warning!!!
This shall not be used if certificate grants access to any really protected resources! Never use this unless you are sure this is what you need! This is usable only to generate certificates with no real world usability!

Originally it has been developed to provide an infrastructure for demo site of the Chemomentum grid infrastructure, where user could type-in his data and get a certificate instantly. Those certificates were valid only on the demonstration site, allowing anybody to try complex grid infrastructure without need to install it.

If you are interested with such solution you shall link somewhere (make an index page?) a jsp page named keygen.jsp. This page is an interface to automatic keystore generation from user supplied form data.

Email message templates

There are .msg files in conf directory, which are templates for email messages sent to user (copies of those messages are sent to administrator to notify him on CA operations). You may edit those templates, and following variables may be used in `${VARIABLE_NAME}` form to include custom request info into email:

- `${DN}` – request/ceritifcate DN field
- `${REQUEST_ID}` – id of the request
- `${CA_NAME}` – as ca title on web page
- `${ADMIN_NOTE}` – note sent by admin on rejection

Administrator passwords

Administrator users and passwords are stored in loca/userMap file. You shall update `loca/userMap` to set your username and password and remove default ones. Syntax for file is:

```
user=password, role
```

And required role is admin. Instead of plain text password you may use MD5 sum of password, with `MD5:<sum value>` instead of password. On linux machines you may get this sum by invoking:

```
echo -n  mySecretPassword | md5sum
```

(-n option is very important!). Examples in both formats are present in provided userMap file.

Customizing web pages

After basic preparation it is recommended to customize your web pages of the server. To do this alter jsp pages located in `/webapp` directory of the installation. You can also alter CSS style sheet (`style.css` file) to match your vision and include your organization logo. You need only basic knowledge of html/CSS to do this.

Using PnP-CA

Using PnP-CA is very easy. When it is started for a first time go to http://localhost:8080/ to see interface of the CA. As a normal user you may submit CSR, view and download certificates and view submitted requests.

On CA services page user may find CRL (certificate revocation list), CA certificate and information about RSS channel connected with server.

If you have administrative privileges, you may log in with user name and password and manage requests or manage certificates. You can approve or reject requests. Rejected requests are removed from the system, but may be resubmitted by the user later (if revocation was accident).

You may revoke already issued certificate. Note, that this cannot be undone, and after this operation any system using CRL from your CA will refuse to accept this certificate.

Administrative tasks

Backups and control

All certificates and requests are stored in data/ subdirectory. Data is stored in standard files, in PEM formats, and metadata is stored in text files. There is no database involved.

This means, that you can backup all server on the fly with no risk on data corruption. Also, if you have doubts about data integrity or correctness, you can examine it (and if you are self-confident alter it).

Vulnerable material is stored in local/ directory, especially keystore and userMap. Remember, that if somebody will get keystore, he may issue as good certificates as your own. Compromising CA keystore effectively means, that you shall trust it no more, and change all certificates in the system.

Importing other CA into PnP CA

This means also, that it is possible to import contents of another CA into PnP CA – you need to store it properely in data directory. There is special java class which does it for you.

First you need to install in /local a CA keystore and certificate, and adjust conf/config.xml to match keystore password.

Next you need to run a `pl.edu.icm.pnpca.util.CertificateImporter` class with a directory as a command line parameter. All certificates from this directory will be imported as if they were created within this PnP CA.

To do this, in installation directory of the server run:

```
java -cp pnpca-server-0.91-SNAPSHOT.jar
        pl.edu.icm.pnpca.util.CertificateImporter <certificate dir>
```

After that you shall be able to use PnP CA as your CA with all former certificates imported.

Troubleshooting

If you happen to have problems, first examine system logs. You will find it in logs/ directory. File startup.log contains only java command output, and if there are errors here, it means that something is wrong with basic system configuration (no java, problem with libraries and so on).

Main system log comes into logs/pnpca.log . If system does not start or you have errors, you shall examine this log to see what is happening. On top of the stack traces, you will usually have some meaningful information.

Typical problems

Remember to install Unlimited Strength cryptography files into your Java! This is most common problem with PnP-CA running.

Second typical problem is address already in use exception at startup. It means, that there is another program (or running PnP-CA) listening on specified ports. You have to select another port or stop the program which uses required port.