Using an Aladdin eToken PRO to store grid certificates

From GridWiki
(Redirected from EToken)

A very secure way to store grid certificates is on an Aladdin eToken (http://www.aladdin.com/eToken/default.asp). These tokens are so-called smartcards with a USB form factor. They can be used to securely generate and store X509 certificates and/or SSH keys. The public part of an X509 certificate can be accessed by an application, but the corresponding private key can never be copied off an eToken. This, in theory, makes such a device ideal for storing sensitive data such as grid certificates.

Contents

Platform support

With some tinkering it is possible to use an eToken on

- Windows
- Linux:
  - Redhat Enterprise Linux 4 and compatible (Scientific Linux 4, CentOS 4)
  - Fedora Core 4 or higher
  - Suse 9.3 or higher

- MacOS X

This document tries to explain the *tinkering* ...

Notes

- not all functions are available on all platforms. Currently, it is not possible to reformat an eToken on Linux. This can only be done on Windows (and perhaps MacOS, but this is untested).
- there is no native 64bit platform support. It is possible to use an eToken on an x86_64 architecture but it requires 32bit versions of all relevant tools (`pcsc-lite`, `openssl`, etc)

Downloading the Aladdin eToken RTE software

Due to licensing restrictions we cannot supply the eToken drivers and libraries on this site, these need to be downloaded from Aladdin. You can find the required software on the web:

- Windows: http://www.aladdin.ru/bitrix/redirect.php?event1=download&goto=/upload/iblock/2c0/RTE_3.65.zip
- Linux: http://www.aladdin.ru/upload/iblock/609/eToken_PKI_Client_4_55_Linux.rar
- Mac OS X: http://www.aladdin.ru/bitrix/redirect.php?event1=download&goto=/upload/iblock/973/PKI_3_65_Mac.zip

If you're running Windows XP or Vista you can also use the newer PKI Client 4.5/4.55 software:

- PKI Client 4.55: http://www.aladdin.ru/bitrix/redirect.php?event1=download&goto=/upload/iblock/547/eToken_PKI_Client_4.55_Win.zip

However, you need to make sure that your eToken is initialized in **3.65 compatible mode** under the **Advanced Settings** screen otherwise your token is inaccessible on any other platform than Windows.

(the files on Aladdin's Russian site do not require a password to unpack them, the ones on the US site do...)

To unpack the Linux archive, the rar command is required.

Important

Do **NOT** install the PKI Client 4.0 software (Windows only)! eTokens initialized with this version of the Aladdin software are completely unusable by older releases. If you want to use your eToken on any other platform than Windows then stick with the RTE_3.65 software release instead.

Installing the Aladdin eToken RTE software

Windows

Unzip the RTE_3.65.zip archive and install `RTE_3.65.msi` file. After rebooting the operating system should recognize the eToken automatically when it is inserted (a red light will start to glow inside the eToken).

The RTE software is now installed in "default" mode. To get a few more administration options, including a nifty `initialization` button in the eToken Properties screen, set/change the registry key

```
HKLM\SOFTWARE\Aladdin\eToken\eTProperties\Advanced:DWORD = 0x1F
```

(default value is 0x1).

You can now continue on to Testing the eToken RTE software.

Linux

There are two ways to install the necessary tools:

- manual installation using the Aladdin `petoken` installation script. You have chosen the difficult path. Instructions can be found in Aladdin eToken PRO Manual Installation.
- install a package for your distribution which does all the hard work for you. There are two flavours: one for RPM based systems, and one for Debian based systems.
  The RPM has been tested on
  - CentOS 4 / Scientific Linux 4 (rhel4), i686 and x86_64 architectures
  - Fedora Core 5 (fc5), i686 architecture
  - OpenSuSE 10.1, 10.3 (suse10), i686 architecture,
    while the Debian package is known to work on
  - Debian 4.0 stable (codename etch)
  - Ubuntu 6.06 LTS
  - Ubuntu 7.04

Contents of the pre-built packages

The RPM and Debian packages contain the following.

- Aladdin eToken RTE 3.65 software (in binary form only).
- the `mkproxy` script to generate grid proxies (see Using an Aladdin eToken PRO to generate grid proxies for details).
- `pkcs11-tool` command from the `opensc` package (http://www.opensc-project.org/)
- a patched version of the `engine_pkcs11` module, also from the `opensc` package (http://www.opensc-project.org/). This patch allows for PINs longer than 11 characters.
- a patched version of `openssl` v0.9.8d to allow the user to generate short-lived proxies (the patched file is x509.c; the patch has been submitted to the `openssl-dev` mailing list).
- system `/etc/init.d` startup scripts to correctly start the `etokend` and `etsrvd` daemons at system startup.
- hotplugging scripts to allow the correct hotplugging of your USB eToken device. These hotplugging scripts work on all Linux 2.6+ kernels, including 2.6.16 and above.
- PC/SC-lite `pcscd` Smart Card daemon v1.3.1, plus system startup script.

All binaries are installed in `/opt/etoken-pro`. The system startup and configuration scripts are installed in their appropriate location.

Debian packages

Instructions for obtaining and installing the software for Debian based systems can be found here.

RPM packages

Instructions on how to build and install the `etoken-mkproxy` rpm are here.

For Nikhef, SARA and IGTF members the following will also work:

```
# FC5:
rpm -ivh http://www.nikhef.nl/grid/ndpf/files/Aladdin-eToken/etoken-mkproxy/etoken-mkproxy-LATEST.fc5.i386.rpm
```

```
# RHEL4:
rpm -ivh http://www.nikhef.nl/grid/ndpf/files/Aladdin-eToken/etoken-mkproxy/etoken-mkproxy-LATEST.rhel4.i386.rpm
# Suse10:
rpm -ivh http://www.nikhef.nl/grid/ndpf/files/Aladdin-eToken/etoken-mkproxy/etoken-mkproxy-LATEST.suse10.i386.rpm
```

Manual installation

Instructions on how to manually install the Aladdin eToken software using the `petoken` install script can be found in Aladdin eToken PRO Manual Installation.

Differences between manual and packaged installations

There are some differences between manual installations and installation of the pre-built packages above:

Manual installation:

- Most of the files end up in `/usr/local/bin`, `/usr/local/lib` and `/usr/local/sbin`.
- the `mkproxy` script is not included in the manual installation. You can download it separately, including all required binaries by following the instructions in Using an Aladdin eToken PRO to generate grid proxies.

Package installation:

- Most of the files end up in `/opt/etoken-pro`, with a single symlink in `/usr/local/lib`.
- the package includes a patched version of the `openssl x509` command which allows you to specify short-lived certificates/proxies, much like the `grid-proxy-init` tool:

```
/opt/etoken-pro/bin/mkproxy --valid 4:00
```

This patched version of the `openssl` command is now also included in the `mkproxy` tarballs.

Mac OS X

You can use the eToken PRO on Mac OS X 10.4 and above in the same way as on Linux. Just download and install the Aladdin drivers (http://www.aladdin.ru/bitrix/redirect.php?event1=download&goto=/upload/iblock/973/PKI_3_65_Mac.zip) and the etoken_mkproxy package (http://www.nikhef.nl/grid/ndpf/files/Aladdin-eToken/etoken_mkproxy_1.41.dmg) (universal binaries).

The latest 4.55 package for MacOSX is also at at Aladdin.ru (http://www.aladdin.ru/upload/iblock/672/eToken_PKI_Client_4_55_Mac.zip)

The software installs into

```
/usr/local/etoken-pro
```

by default.

Testing the eToken RTE software

Windows

You can access your eToken using the software installed by the RTE_3.65.msi installation package (usually in Start->Programs->eToken).

If you have installed Cygwin (http://www.cygwin.com/) and the Mkproxy-cygwin.tar.gz tarball you can also access your eToken using the `pkcs11-tool` command:

- start a Cygwin shell
- go to the directory where you have unpacked the Mkproxy-cygwin.tar.gz tarball
- type

```
./etoken-pro/bin/pkcs11-tool --module=$WINDIR\\system32\\etpkcs11.dll -L
```

to list all inserted tokens.

Note This works only if you are logged in locally on the Windows machine. This will **not** work when logging in remotely using either a Cygwin `sshd` service or Remote Desktop.

Linux

If you have installed the `etoken-mkproxy` RPM you can access your eToken using the `pkcs11-tool` command:

```
/opt/etoken-pro/bin/pkcs11-tool --module=/usr/local/lib/libetpkcs11.so -L
```

which should return

```
Available slots:
Slot 0           AKS ifdh 00 00
  token state:   uninitialized
Slot 1           (empty)
Slot 2           (empty)
Slot 3           (empty)
```

If you have performed a manual installation then you can find the `pkcs11-tool` in the tarball for your platform (or in the `opensc` toolkit):

- FC5: Mkproxy-fc5.tar.gz
- RHEL4: Mkproxy-rhel4.tar.gz
- Suse10: Mkproxy-suse10.tar.gz

Then

- go to the directory where you have unpacked the `Mkproxy-`*`<platform>`* tarball
- type

```
export PATH=$PATH:$PWD/etoken-pro/bin
export LD_LIBRARY_PATH=$PWD/etoken-pro/lib:$LD_LIBRARY_PATH
```

- then type

```
pkcs11-tool --module=/usr/local/lib/libetpkcs11.so -L
```

which should return

```
Available slots:
Slot 0           AKS ifdh 00 00
  token state:   uninitialized
Slot 1           (empty)
Slot 2           (empty)
Slot 3           (empty)
```

Congratulations, your eToken is ready for use!

Mac OS X

If you have installed the eToken PRO mkproxy package **and** the Aladdin drivers, you can open a terminal window with a command prompt and type (with the token inserted, of course):

```
/usr/local/etoken-pro/bin/pkcs11-tool --module /usr/local/lib/libetpkcs11.so -L
```

which should return:

```
Available slots:
Slot 0           AKS ifdh 0 0
  token state:   uninitialized
Slot 1           (empty)
```

If this succeeds, your eToken is ready for use.

Why not use the OpenSC tools?

The OpenSC project (http://www.opensc-project.org/) also provides driver software for Aladdin eTokens. Versions up to and including 0.11.2 did not support the CardOS version used on our eTokens.

Update: As of `opensc-0.11.3` CardOS version 4.2B is supported!

The OpenSC `cardos-info` command still gives

```
Info : CardOS V4.2B (C) Siemens AG 1994-2005
Chip type: 123
Serial number: 26 57 ad 07 0f 21
Full prom dump:
33 66 00 45 CB CB CB CB 7B FF 26 57 AD 07 0F 21 3f.E....{.&W...!
00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 ................
OS Version: 200.9 (unknown Version)
Current life cycle: 32 (administration)
Security Status of current DF:
Free memory : 0
ATR Status: 0x0 ROM-ATR
Packages installed:
Ram size: 4, Eeprom size: 32, cpu type: 66, chip config: 63
Free eeprom memory: 23167
System keys: PackageLoadKey (version 0xfe, retries 10)
System keys: StartKey (version 0xff, retries 10)
Path to current DF:
66 66 50 00 02 01 ffP...
```

but you can access the token using the `opensc-explorer` and `pkcs15-*` commands.

However....

it seems you can either use an eToken using the OpenSC software or (*XOR*) use Aladdin's proprietary software, that is, information stored on an eToken using the OpenSC is not visible to the Aladdin software and vice versa. Also, as the OpenSC software has no notion of 'FIPS compliant' it will (probably) not be possible to put the eToken in FIPS-compliant mode.

Thus, we stick with Aladdin's eToken software and our mixed set of utilities for now...

Initializing your eToken (Windows only)

Note Currently you can only initialize your eToken on the Windows platform. It will **not** work on Linux, especially changing the SOPIN is not working. This is most likely due to missing functionality in the Linux Aladdin RTE software.

To initialize your eToken for the first time you can use either the Windows client software (Start->Programs->eToken->eToken Properties) or you can use `pkcs11-tool` :

```
pkcs11-tool --module $WINDIR\\system32\\etpkcs11.dll --init-token --label "Your Label" --so-pin Your_New_SO_PIN
```

where `Your_New_SO_PIN` is the new **S**ecurity **O**fficer Password. You will be prompted for the existing SOPIN (through a pop window). The default value for the SOPIN is '1234567890' After typing in the correct (old) SOPIN you will see a message like this:

```
Token successfully initialized
```

You can now use

```
pkcs11-tool ---module $WINDIR\\system32\\etpkcs11.dll -L
```

to view the status of your eToken:

```
Available slots:
Slot 0           AKS ifdh 00 00
  token state:   uninitialized
Slot 1           (empty)
```

As you can see it remains in the status "uninitialized" but it **is** ready for use.

Initializing your user PIN

After initializing or reformatting your eToken you must initialize the user password ('PIN') before you can store any data on it. Initializing the user PIN can be done on both Windows and Linux.

Note

On Windows

```
PKCS11_MOD=$WINDIR\\system32\\etpkcs11.dll
```

On Linux

```
PKCS11_MOD=/usr/local/lib/libetpkcs11.so
```

The user PIN is initialized using

```
# pkcs11-tool --module $PKCS11_MOD --init-pin
Please enter SO PIN:
Please enter the new PIN:
Please enter the new PIN again:
User PIN successfully initialized
```

Please choose your user PIN carefully. It must be between 6 to 12 characters long. If your PIN is more than 12 characters then you will not be able to access your eToken using `openssl` commands , nor will you be able to generate grid proxies using your eToken!

Using your eToken in Firefox

A very easy method for importing (or removing) keys in your eToken is to add the eToken as a Security Device in Firefox.

This is explained in Using an Aladdin eToken with firefox.

Generating or storing a grid certificate on the eToken

Now that you have initialized your eToken you can either generate a new certificate/private key pair on the eToken itself (very secure!) or you can load your existing grid certificate onto the eToken.

This is explained in Storing your grid certificate on an Aladdin eToken.

Generating grid proxies using an eToken

It is also possible to generate a grid proxy using the eToken.

This is explained in Using an Aladdin eToken PRO to generate grid proxies.

Using an eToken in Java

Sun's Java SDK has pretty good support for external PKCS11 libraries. See http://java.sun.com/j2se/1.5.0/docs/guide/security/p11guide.html for details.

To use your eToken create a configuration file, e.g. /tmp/java-pkcs11.cfg:

```
name = eToken
library = /opt/etoken-pro/lib/libetpkcs11.so
```

Now you can use the Java keytool to access your eToken:

```
# keytool -keystore NONE -storetype PKCS11 -list \
  -providerClass sun.security.pkcs11.SunPKCS11 \
  -providerArg   /tmp/java-pkcs11.cfg
Enter keystore password:

Keystore type: PKCS11
Keystore provider: SunPKCS11-eToken

Your keystore contains 1 entry

(eTCAPI) Jan Just Keijser's NIKHEF ID, PrivateKeyEntry,
Certificate fingerprint (MD5): 39:04:61:C9:D8:8F:0D:2E:F0:59:B5:59:28:06:7D:47
```

To avoid having to type

```
-providerClass sun.security.pkcs11.SunPKCS11 \
-providerArg   /tmp/java-pkcs11.cfg
```

you can statically configure the java.security file: add a line

```
security.provider.9=sun.security.pkcs11.SunPKCS11 <PATH>/java-pkcs11.cfg
```

to the `$JAVA_HOME/jre/lib/security/java.security` file and now you can use

```
# keytool -keystore NONE -storetype PKCS11 -list
```

to get the same output as before.
