scquery tool

Pascal Bourguignon

October 11th, 2018

Contents

1 Purpose

This tool can be used to list:

- the authentication certificates avalable on an IAS-ECC SmartCard
- the UPN stored in those certificates.

The certificate ID is reported as a Kerberos `X509_user_identity` string, such as:

```
PKCS11:module_name=/usr/lib/libiaspkcs11.so:slotid=1:token=ECC MI:certid=e828bd080fd2500000104d494f430
```

The UPN is usually formated as an email address.

2 Usage

```
$ scquery --module=/usr/lib/libiaspkcs11.so
PKCS11:module_name=/usr/local/lib/libiaspkcs11.so:slotid=1:token=ECC MI:certid=e828bd080fd2500000104d49
subjectAltName:email:pascal.bourguignon-obiane@interieur.gouv.fr
PKCS11:module_name=/usr/local/lib/libiaspkcs11.so:slotid=1:token=ECC MI:certid=e828bd080fd2500000104d49
subjectAltName:othername:1.3.6.1.4.1.311.20.2.3:pascal.bourguignon.1468520@minint.fr
subjectAltName:othername:1.3.6.1.5.2.2:KRB.MININT.FR:1:pascal.bourguignon.1468520
subjectAltName:email:pascal.bourguignon-obiane@interieur.gouv.fr
```

If no smartcard is inserted, then an error message "No smartcard" is printed on `stderr`, and it exits with status 1.

3 Output format

`scquery` scans the smartcards for X509 certificates that have signing RSA private key with the same ID.

`scquery` outputs lines containing fields separated by colons. If a field value contains a colon or a backslash, it is escaped with a backslash.

For each such certificate, `scquery` issues one `PKCS11` line, followed by zero or more `subjectAltName` lines containing the subjectAltName entries in the certificate.

3.1 subjectAltName of type email

Records of `subjectAltName` of type `email` contain three fields:

- `subjectAltName`
- `email`
- the email address.

```
subjectAltName:email:firstname.surname@domain.example.com
```

3.2 subjectAltName of type othername

Records of `subjectAltName` of type `othername` contain at least four fields:

- `subjectAltName`
- `othername`
- the **OID** of the othername type.
- one or more fields depending on the othername in question.

ln	sn	oid
Microsoft Universal Principal Name	msUPN	`1.3.6.1.4.1.311.20.2.3`
Kerberos Principal Name	KPN	`1.3.6.1.5.2.2`

For a Microsoft Universal Principal Name, theere is one additionnal field containing the `UPN` (looks like an email address).

For a Kerberos Principal Name, the princial structure which is encoded as a hierarchical sequence is flattened with each element as a separate field:

- the realm
- the principal type (normally 1 for `KRB5_NT_PRINCIPAL`),
- a list of components (usually one).

```
subjectAltName:othername:1.3.6.1.4.1.311.20.2.3:pascal.bourguignon.1468520@minint.fr
subjectAltName:othername:1.3.6.1.5.2.2:KRB.MININT.FR:1:pascal.bourguignon.1468520
```

3.3 Example

This `scquery` tool can be used to list the certificates available, select one that has a `msUPN`, and use it with `kinit`\(1) to get a ticket.

The script `sources/sckinit` is an example of use:

```
#!/bin/bash

scquery=scquery-cl
upn_oid=1.3.6.1.4.1.311.20.2.3
kpn=oid=1.3.6.1.5.2.2
oid=$upn_oid

printf 'Please, insert your smart card, and press RETURN:'
read line

${scquery} \
  | grep -e "^PKCS11:\|:${oid}:" | grep -B1 ":${oid}:" | tail -2 \
  | ( read X509_user_identity
      IFS=: read s o oid upn
      kinit -V \
          -C \
          -X 'X509_anchors=FILE:/etc/chaine-kdc.pem' \
          -X 'X509_anchors=FILE:/etc/chaine2.pem' \
          -X "X509_user_identity=${X509_user_identity}" \
          -E "${upn}" )

klist
```

4 Compilation

4.1 scquery-cl

This tool is implemented in Common Lisp. It requires the following dependencies:

- some `quicklisp` libraries (including `cl+ssl` and other subdependencies).
- ASN.1 parser `asinine` as patched in `https://github.com/informatimago/asinine.git`
- PKCS11 API `com.informatimago.clext.pkcs11` from `https://github.com/informatimago/lisp.git`

These two libraries must be git cloned into `~/quicklisp/local-projects/` first.

`scquery-cl` can be compiled with Clozure CL from `https://ccl.clozure.com/download.html` (other CL implementations may (and should) be able to compile it, but it has not been tested).

You can use either the `Makefile` in `sources/` to generate the `scquery-cl` executable, or `loader.lisp` to compile and load it into a lisp image. It can be run in the lisp REPL as:

```
(scquery:main) ; or:
(scquery:main '("--module" "/usr/lib/libiaspkcs11.so"))
```

4.2 scquery-c

This tool is implemented in C11.