IGTF Trust Anchor Distribution

in the wake of OpenSSL1

January 19, 2010

18th EUGridPMA Dublin meeting

Structure of the trust anchor directory

Hash functions and code

OPENSSL1

Structure of the trust anchor store

```
bosui:certificates:1008$ cd /etc/grid-security/certificates/

bosui:certificates:1008$ ls -l|grep 16da
-rw-r--r-- 1 root root   5341 Oct 26 21:02 16da7552.0
-rw-r--r-- 1 root root     40 Oct 26 21:02 16da7552.crl_url
-rw-r--r-- 1 root root    442 Oct 26 21:02 16da7552.info
-rw-r--r-- 1 root root    630 Oct 26 21:02 16da7552.namespaces
-rw-r--r-- 1 root root   4842 Jan 13 08:17 16da7552.r0
-rw-r--r-- 1 root root    342 Oct 26 21:02 16da7552.signing_policy

bosui:certificates:1009$ ls -l policy-*
-rw-r--r-- 1 root root 2633 Oct 26 21:03 policy-igtf-classic.info
-rw-r--r-- 1 root root  152 Oct 26 21:03 policy-igtf-mics.info
-rw-r--r-- 1 root root  256 Oct 26 21:03 policy-igtf-slcs.info

bosui:certificates:1010$ cat policy-igtf-mics.info
# @(#)policy-igtf-mics.info - IGTF mics authorities
# Generated Monday, 26 Oct, 2009
version = 1.32
requires = TACC-MICS = 1.32, \
    NCSA-mics = 1.32
```

Hashes

- Hashes used for lookup of the issuing CA for a given certificate
- Independent from the 'fingerprint'

```
$ openssl x509 -noout -subject -issuer \
    -fingerprint -sha1 -hash \
    -in 16da7552.0
subject= /C=NL/O=NIKHEF/CN=NIKHEF medium-security certification auth
issuer= /C=NL/O=NIKHEF/CN=NIKHEF medium-security certification auth
SHA1 Fingerprint=E5:FA:C3:3B:44:8F:26:1B:3D:D1:DE:BA:5F:EC:ED:35:A9:3F:23:21
16da7552
```

Code changed in OpenSSL 1 betas

```
unsigned long X509_NAME_hash(X509_NAME *x)
        {
        unsigned long ret=0;
        unsigned char md[SHA_DIGEST_LENGTH];

        /* Make sure X509_NAME structure contains valid cached encoding */
        i2d_X509_NAME(x,NULL);
        EVP_Digest(x->canon_enc, x->canon_enclen, md, NULL, EVP_sha1(), NULL);

        ret=(   ((unsigned long)md[0]     )|((unsigned long)md[1]<<8L)|
                ((unsigned long)md[2]<<16L)|((unsigned long)md[3]<<24L)
                )&0xffffffffL;
        return(ret);
        }

unsigned long X509_NAME_hash_old(X509_NAME *x)
        {
        unsigned long ret=0;
        unsigned char md[16];

        /* Make sure X509_NAME structure contains valid cached encoding */
        i2d_X509_NAME(x,NULL);
        EVP_Digest(x->bytes->data, x->bytes->length, md, NULL, EVP_md5(), NULL);

        ret=(   ((unsigned long)md[0]     )|((unsigned long)md[1]<<8L)|
                ((unsigned long)md[2]<<16L)|((unsigned long)md[3]<<24L)
                )&0xffffffffL;
        return(ret);
        }
```

Impact of change

- Today: lookup will fail if a new OpenSSL is used
- Lookups happen
  - via the system OpenSSL, dynamically linked
  - Via application-specific OpenSSL versions
  - Via statically linked openssl libraries
  - Via other implementations of the hash algorithm
    e.g. in the TrustManager Java implementation
    or in Apache, or ...
- Both implementations will co-exist on single system
- Will thus require
  - separate trust stores
  - A single trust store that supports both algorithms

Proposal for the IGTF distribution

- For installation bundles, tar-balls and RPMs: all CAs and files are named after their alias from the info file
- Symlinks are used to generate the structure for both versions of OpenSSL
- installation bundle (the "./configure && make && make install" tarball) will create both symlinks
- pre-installed bundles have both hashes, also using symlinks

What gets into /etc/grid-security/certificates?

```
393f7863.0 -> AEGIS.pem
393f7863.info -> AEGIS.info
393f7863.namespaces -> AEGIS.namespaces
393f7863.signing_policy -> AEGIS.signing_policy
AEGIS.crl_url
AEGIS.info
AEGIS.namespaces
AEGIS.pem
AEGIS.signing_policy
cc5645bd.0 -> AEGIS.pem
cc5645bd.info -> AEGIS.info
cc5645bd.namespaces -> AEGIS.namespaces
cc5645bd.signing_policy -> AEGIS.signing_policy
```

Dual packaging

- Pro
  - Target system does not need to run c_rehash
  - Works for hybrid deployments,
    with some software linked to 0.x, others to 1.0, and others have built-in hashing codes
- Con
  - Symlinks may upset (or be messed up by) locally running c_rehash
  - Unclear which one is used - when the links are severed different programs will react differently
  - Doesn't work on filesystems that do not support symlink

Fetch-crl

```
locationFiles=`${ls} \
  "${locationDirectory}"/*.${crlLocationFileSuffix} \
  2>/dev/null`
```

- Will find all files with a ".crl_url" suffix
- Use the OpenSSL in the path (since v2.7) to compute the hash locally

Caveats

- The .r0 files will be hashed according to openssl version used *by fetch-crl*, not target software
  - make symlinks locally to get a dual-use CRL?
  - Should we provide a script, or an update to fetch-crl?

Alternatives

- Use c_rehash (or other OS specific tools) locally
- Use hard copies instead of symlinks
  - Either as copies of PEM files, or only two copies of same
- Make the links on the target system
  - Using ln -s from the make file
  - Using a %post script in the RPM
    - needs to be matched with a %preun which could fail
    - resulting links are not 'owned' by an RPM

Collateral changes

- Update RPM format to v4.4 series
- Update Java keytool to Java6
  - Now supports CA keys with keysize > 2048
  - Keys will not work with old (pre-Java5) tools with Sun JCE
- Auto-generated sequence numbers support multiple CAs with the same subject name but different aliases
  - Alias MUST be unique for each trust anchor
  - Still not supported in fetch-crl, though
  - And it's dangerous anyway
    (see issues related to the two UTNUserFirst-Hardware CA certs, one self-signed and one by Comodo AAA)

Implementation plan

Options for discussion:

- For 1.33 release still have only the old version since “OpenSSL1 is still in beta only”
- For 1.33 and 1.34 releases, have both versions available with the old style as the advertised one for 1.33, and changing to the new one for 1.34
- For 1.33 release, have both versions available with the **new** style as the advertised one
- For 1.33 release, just have the new one, since it’s fully backward-compatible anyway
- *Or something else …*