Certificate Authorities: Information and Usage

Marcus Christie

December 16, 2003

1 Introduction

It is assumed that the reader knows, basically, what a certificate authority (CA) is. This document describes the usage of CA's in the Extreme Lab.

A X509 certificate (and thus a CA) is required for using the Globus Toolkit. This certificate forms the basis by which security is implemented in the Globus Toolkit. Both users and resources on the grid need to have their own certificates. Initially, the Globus organization ran a CA and issued certificates. They now plan to discontinue this service and will do so by the end of 2003. Since all of our certificates were based on this CA, we needed to find a new CA.

As it has turned out, we now have a hybrid approach. We currently use the NCSA CA for our user certificates. We also have a locally installed CA that we use for issuing host/server certificates. There are several advantages and disadvantages to running one's own CA, which will be explored in section 2.4. But, for now at least, it seems to be a workable solution. We may decide in the near future, however, to use the NCSA CA for our host certificates as well.

2 Tiny CA

TinyCA (http://tinyca.sm-zone.net/) is, according to the web page:

> ... a simple graphical userinterface written in Perl/Gtk to manage a small CA (Certification Authority). TinyCA works as a frontend for openssl

We are using TinyCA to implement our own CA. Currently, we only use this CA for issuing host certificates, but we could use it to issue user certificates if necessary. In this section, I'll cover installation issues and usage of TinyCA.

2.1 Installation

Currently, version 0.5.2 of TinyCA is installed in `/l/esysadm/packages/noarch/TinyCA/0.5.2`. While TinyCA is written in Perl and is thus architecture neutral, the Perl modules that TinyCA needs are built in an architecture specific way. Currently, I've only built the modules on Solaris.

Here are the installation steps:

1. Download the tar-ball from the web site

2. Unpack tar-ball into directory where you want to install TinyCA (no compilation is needed).

3. Download the required modules and build them into the architecture specific Perl module directory. For the Solaris build, this meant building and then installing the Perl modules in `/l/esysadm/arch/solaris/lib/perl`. Here are the modules that had to be built (depending on your system, you may or may not need all of these; you may want to try running TinyCA first and then see which modules are missing).

   (a) Gtk-Perl (http://www.gtkperl.org/). Download latest from cpan.org.

   (b) XML::Writer (http://www.megginson.com/Software/index.html). Download from cpan.org.

   (c) gettext-1.01. Download latest from cpan.org.

   Installation of Perl modules is pretty standard. Just make sure to install into the appropriate directory as explained above.

2.2 Post-Installation

First, we need to create a shell script that will make the Perl modules we just installed available to Perl and that will then launch the TinyCA executable. I create the shell script `tinyca-solaris.sh` in the directory `/l/esysadm/packages/noarch/TinyCA/0.5.2` containing the following:

```
#!/bin/sh

PERL5LIB=/l/esysadm/arch/solaris/lib/perl:\
/l/esysadm/arch/solaris/lib/perl/lib/site_perl/5.6.0/sun4-solaris
export PERL5LIB

cd /l/esysadm/packages/noarch/TinyCA/0.5.2
./tinyca
```

This sets up the Perl module libraries that we just installed and then calls TinyCA.

Next, we need to modify the TinyCA script so that it points to the proper location of openssl. I changed line 52 of the `tinyca` script from

```
$init->{'opensslbin'} = "/usr/bin/openssl";
```

to

```
$init->{'opensslbin'} = "/l/ssl/bin/openssl";
```

Finally, we need to create a link in the architecture specific `bin/` directory to the shell script we created above. In `/l/esysadm/arch/solaris/bin`, I issued the following command:

```
ln -s /l/esysadm/packages/noarch/TinyCA/0.5.2/tinyca-solaris.sh \
tinyca
```

2.3 Usage

To use TinyCA, ssh as user globus to one of the Extreme Lab's Solaris machines. Issue the following commands:

```
globus@babel:/u/globus> /l/local/bin/tinyca &
```

The configuration files for our CA are stored in `~globus/.TinyCA`.

2.3.1 Creating New CA

This has already been done, but just a few comments in case another CA is needed or a new CA from scratch is needed. In TinyCA, click the "New CA" button. You'll be given a dialogue box for configuring the CA. First of all, use the defaults as much as possible. Secondly, you'll want to configure the Distinguished Name of your CA. Currently, the DN of our CA is "/C=US/O=Indiana University/OU=Computer Science/CN=Certificate Authority" which forms the basis of the DN's of all the certificates this CA will issue.

2.3.2 Creating a New Server Certificate

Here are the steps for creating a new server (host) certificate.

1. Create a new certificate request. Click on the "Requests" tab and then click on the "New" button.
2. For the common name, enter `host/hostname.extreme.indiana.edu`, substituting "hostname" as appropriate. Note, Globus Toolkit expects the common name of a host certificate to begin with the "host/" string, so this is essential[1].
3. Enter a password of `temp`. It's okay that it isn't a good password since we'll be storing them without the password encrypted but readable only by root on the machines we issue them on. This is the standard Globus way of doing things.
4. Leave the defaults and click OK.
5. Sign the request. Click on the certificate request just generated and then click on the "Server" button.
6. You'll be prompted for the password. For security reasons, the password is not given in this documentation. See the CA administrator (currently Marcus Christie) for further information.
7. This generates the certificate and the key which can be viewed in their respective tabs.

[1] It turns out this isn't as essential as I originally thought. For example, if you get a server certificate from the NCSA CA, the common name is simply the hostname. Still, I recommend using the "host/" prefix because it offers a simple way of telling a host certificate from a service certificate.

2.3.3 Deploying a Server Certificate

1. Click on the "Certificate" tab. Select the server certificate for deployment.
2. Click the "Export" button. You'll want to export the certificate as a `.pem` file to a temporary location.
3. Move the file to `/etc/grid-security/hostcert.pem`. If there is a `hostcert.pem` file already there, rename it according to what type of certificate it is. For example, I renamed the globus host certificates as `hostcert.pem.globus` and then I moved in the new certificate. This file should be owned by root with permissions 644.
4. Click on the "Keys" tab. Select the server key for deployment.
5. Click the "Export" button. You'll want to export the key as a `.pem` file and **without** a passphrase. Select a temporary location. When you click OK, TinyCA will ask you for the password protecting the key. If you created the key as specified above, then enter `temp` for the password.
6. Move the file to `/etc/grid-security/hostkey.pem`. If there is a `hostkey.pem` file already there, rename it according to what type of key it is. For example, I renamed the globus host key as `hostkey.pem.globus` and then I moved in the new keys. This file should be owned by root with permissions 400. This is **very important** since the host key is saved without password protection.

2.3.4 Other Tasks

Obviously there is much more that TinyCA can do. This is all that I've done with TinyCA at this point, so this is all that is covered. I haven't covered renewing a certificate (although I'll add that to this documentation once that becomes relevant), for example. If you wish to do something with TinyCA that isn't covered here, either experiment or consult online documentation.

2.4 Pros and Cons of Running CA

Pros:

No need to wait on a CA to sign your certificates.

Central management; since the CA administrator will be in charge of the host certificates anyways, it make sense that s/he has control over the issuing of those certificates. This simplifies the Globus certificate administrator's job.

Cons:

Others must install your CA's certificate to be able to access your resources on the grid. (Although, this is a very simple procedure.) See Section 4 for details on how to install the IUCS CA.

3 Enabling CA for Globus

To enable one's CA for use with the Globus Toolkit, follow the these instructions on the Globus website. In summary, the steps are as follows:

1. Export the CA's certificate as a *.pem file. To do this, launch tinyca as specified above and click on the "CA" tab (it opens to this by default). Click on the "Export CA" button. Export the CA's certificate in the PEM format to some location.

2. Run the following command on the file just exported with OpenSSL

   ```
   babel cacert 1 548 $ /l/ssl/bin/openssl x509 -in \
   > /tmp/local-cacert.pem -hash -noout
   aaaddcdf
   ```

3. The output of the previous step, `aaaddcdf`, is a hash of the CA's name. Rename the CA's certificate PEM file to this hash with ".0" appended to it like so:

   ```
   cp /tmp/local-cacert.pem /tmp/aaaddcdf.0
   ```

4. Copy this file into the `/etc/grid-security/certificates` directory. On the Extreme machines, this folder is linked to the following location, which is where this file should go:

   ```
   /l/local/packages/../noarch/globus/certificates/
   ```

5. Create a `signing_policy` file. It should begin with the hash discovered above and end with ".signing_policy". See the link above for more information regarding these documents. Here is the file I created for our CA:

   ```
   # IUCS CA Policy

   access_id_CA   X509    '/C=US/O=Indiana University/OU=Computer\
    Science/CN=Certificate Authority'
   pos_rights     globus  CA:sign
   cond_subjects  globus  '/C=US/O=Indiana University/OU=Computer Science/*'
   ```

   Be sure to place this file in the certificates directory with the CA certificate. In this case it would be named `aaaddcdf.signing_policy`.

6. Finally, make sure the permissions on these files is 644 (world readable, but definitely **not** world writable).

4 Installing the IUCS CA as a Trusted CA

To install the IUCS CA as a Trusted CA in your Grid, do the following:

1. Download iucs-ca.tar.gz.

2. Copy this file to your trusted CA directory. On a Unix system this is `/etc/grid-security/certificates` for a global installation. For a per-user installation, this would be `~/.globus/certificates`. If you are unsure, check with your Grid administrator.

3. Untar and unzip the tarball with something like:

   ```
   tar zxvf iucs-ca.tar.gz
   ```

   This will put the needed files in your trusted CA directory.

4. (Optional) You may need to update your `~/.globus/cog.properties` file. See Section 5.1 for more information.

5 Issues

This section lists known issues and their workarounds.

5.1 Updating cog.properties files

Although one would like to be able to install the CA certificate in `/etc/grid-security/certificates` and be done with it, unfortunately, there seem to be some files that hard code file paths to trusted CA's. One such file is the cog.properties file found in the user's `~/.globus/` directory. As necessary, the `cacert=` line needs to be updated to point to the new trusted CA certificate. Fortunately, this need only be done once in a great while (which is probably why it causes so much consternation; no one thinks to update it).