



Securing A Basic HTCondor Pool

Basic Concepts

- › You have an HTCondor pool
 - Personal HTCondor (1 node)
 - 1000 node cluster

- › Who can use your pool?

Basic Concepts

- › “Who can use it” is really two concepts:
- › The “Who” is authentication
- › The “can” is authorization

Basic Concepts

- › Authentication is finding out WHO some entity is.
- › How is this done?
 - Common methods:
 - Present a secret that only you should know
 - Perform some action that only you can do
 - Present a credential that only you could have

Basic Concepts

- › Authorization is deciding what someone is allowed to do.
- › You must know who they are before you can decide this!

Basic Concepts

- › I'm using "they" pretty loosely here.
- › "They" could be:
 - A user
 - A machine
 - An agent/daemon/service

Basic Concepts

- › In the context of a HTCondor pool:
 - You want only machines that you know to be in the pool
 - You want only people you know to submit jobs

Authentication

- › When users submit jobs, HTCondor authenticates them:
 - FS on Unix
 - NTSSPI on Windows
- › The HTCondor SCHEDD daemon now “owns” the jobs, and acts on their behalf.

Authentication

- › So how can we trust the SCHEDD?
- › Daemon-to-daemon authentication

Authentication

- › A HTCondor daemon must prove to other HTCondor daemons that it is authentic.
- › Quick and Easy: Pool Password

Pool Password

- › All daemons know a “password”
- › This password (hash) is stored:
 - In a permissions-protected file on UNIX
 - In the encrypted part of the registry on Windows

Pool Password

› To set it:

```
% condor_store_cred -c add  
Account: condor_pool@cs.wisc.edu
```

```
Enter password:
```

```
Operation succeeded.
```

Pool Password

- › This is typically done locally on each machine that will use the password
- › On UNIX, you can copy the file containing the hash to each machine
 - COPY IT SECURELY!
 - CHECK THE PERMISSIONS!

Pool Password

- › Configure HTCondor to use it
- › Set your condor_config:

```
SEC_DAEMON_AUTHENTICATION = REQUIRED
```

```
SEC_DAEMON_AUTHENTICATION_METHODS = PASSWORD
```

Pool Password

- › So, are we “All Good”?
- › What about flocking to other pools?
- › HTCondor-C?

Pool Password

- › Password must be the same for everyone – are you prepared to give it to another administrator?
- › What if they also flock with other pools, are you prepared for them to give it to their flocking friends?
- › And so on?

Flexibility

- › It would be nice if each pool could have its own credential
- › Well, you can! Use the SSL authentication method.

Why use SSL?

- › Widely used and deployed
- › Flexible enough for securing communications between HTCondor daemons and also for authenticating users

Basics: OpenSSL

- › OpenSSL is typically already installed on modern Linux systems
- › On more obscure flavors of Unix, and on Windows, you will likely need to install it yourself
- › Can be obtained here:
<http://www.openssl.org/>

Basics: OpenSSL

- › Or, instead of installing OpenSSL everywhere, you can create your credentials on a Linux machine and securely move them to another machine where they will be used
- › Make sure the permissions are such that only the proper people can read the key!

Basics: SSL config

- › You can use the default from the openssl package or start with my simplified version here:
- › <http://www.cs.wisc.edu/~zmiller/cw2014/openssl.cnf>
- › Find the section [req_distinguished_name] and customize it:

```
[ req_distinguished_name ]
stateOrProvinceName_default = Wisconsin
localityName_default       = Madison
0.organizationName_default = University of Wisconsin -- Madison
1.organizationName_default = Computer Sciences Department
organizationalUnitName_default = HTCondor Project
```

Single Credential

- › In this example, we will create a single key/certificate pair and use that to secure communications between HTCondor daemons
- › This is roughly equivalent to the pool password method – it is a shared secret stored in a file

Single Credentials

- › First, create the private key file:

```
openssl genrsa -out cndrsrvc.key 1024
```

```
Generating RSA private key, 1024 bit long modulus
```

```
.....++++++
```

```
...++++++
```

```
e is 65537 (0x10001)
```

```
chmod 600 cndrsrvc.key
```

Single Credential

> Now, create a self-signed certificate

```
openssl req -new -x509 -days 3650 -key cndrsrvc.key \  
-out cndrsrvc.crt -config openssl.cnf
```

You are about to be asked to enter information that will be incorporated into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank

For some fields there will be a default value,

If you enter '.', the field will be left blank.

Country Name (2 letter code) [US]:

State or Province Name (full name) [Wisconsin]:

Locality Name (eg, city) [Madison]:

Organization Name (eg, company) [University of Wisconsin -- Madison]:

Second Organization Name (eg, company) [Computer Sciences Department]:

Organizational Unit Name (eg, section) [HTCondor Project]:

Common Name (eg, YOUR name) []:**Service**

Email Address []:

Single Credential

> Inspect the certificate we made:

```
openssl x509 -noout -text -in cndrsrvc.crt
```

Certificate:

Data:

Version: 3 (0x2)

Serial Number:

8c:94:7b:b1:f9:6a:bd:72

Signature Algorithm: sha1WithRSAEncryption

Issuer: C=US, ST=Wisconsin, L=Madison, O=University of Wisconsin -- \\
Madison, O=Computer Sciences Department, OU=HTCondor Project, CN=Service

Validity

Not Before: May 1 14:31:09 2014 GMT

Not After : Apr 28 14:31:09 2024 GMT

Subject: C=US, ST=Wisconsin, L=Madison, O=University of Wisconsin -- \\
Madison, O=Computer Sciences Department, OU=HTCondor Project, CN=Service

...

Single Credential

- › Great! Now what?
- › Create a map file
 - HTCondor needs to know how to map the distinguished name to an actual username.

For example:

```
/C=US/ST=Wisconsin/L=Madison/O=University of Wisconsin -- Madison/O=Computer  
Sciences Department/OU=HTCondor Project/CN=Service
```

Should map to:

```
condor
```

- › Configure the HTCondor daemons

HTCondor Mapfile

- › Simple format
- › Three fields (on one line)
 - Authentication method (SSL in this case)
 - Source DN
 - Mapped user

SSL

```
"/C=US/ST=Wisconsin/L=Madison/O=University of Wisconsin --  
Madison/O=Computer Sciences Department/OU=HTCondor Project/CN=Service"
```

```
condor
```

condor_config

› Add the following entries:

```
AUTH_SSL_CLIENT_CAFILE = /path/to/cndrsrvc.crt  
AUTH_SSL_CLIENT_CERTFILE = /path/to/cndrsrvc.crt  
AUTH_SSL_CLIENT_KEYFILE = /path/to/cndrsrvc.key
```

```
AUTH_SSL_SERVER_CAFILE = /path/to/cndrsrvc.crt  
AUTH_SSL_SERVER_CERTFILE = /path/to/cndrsrvc.crt  
AUTH_SSL_SERVER_KEYFILE = /path/to/cndrsrvc.key
```

› And the map file:

```
CERTIFICATE_MAPFILE = /path/to/condor_mapfile
```

condor_config

> Tell HTCondor to use SSL:

```
SEC_DAEMON_AUTHENTICATION = REQUIRED
```

```
SEC_DAEMON_AUTHENTICATION_METHODS = SSL
```

That's (mostly) It!

- › You have now enabled SSL authentication between all your HTCCondor daemons
- › But at this point, it isn't much different than using a Pool Password

Creating a CA

- › The solution is to issue separate credentials for each entity that will be involved in authenticating
- › Can't do this with Pool Password, but you can with SSL

Creating a CA

- › This involves creating a Certificate Authority which is trusted by HTCondor
- › All certificates issued by the CA are then trusted
- › Certs can be easily issued for hosts and users

Creating a CA

- › Create the root key and cert which will be used to sign all other certificates
- › This key should be protected with a password (don't forget it!!)

Creating a CA

› Generate a key:

```
openssl genrsa -des3 -out root-ca.key 1024
Generating RSA private key, 1024 bit long modulus
.....+++++
.....+++++
e is 65537 (0x10001)
Enter pass phrase for root-ca.key:
Verifying - Enter pass phrase for root-ca.key:
```

Creating a CA

› Now create a self signed certificate

```
openssl req -new -x509 -days 3650 -key root-ca.key -out root-ca.crt -config openssl.cnf
```

```
Enter pass phrase for root-ca.key: CA PASSWORD HERE
```

```
You are about to be asked to enter information that will be incorporated  
into your certificate request.
```

```
What you are about to enter is what is called a Distinguished Name or a DN.
```

```
There are quite a few fields but you can leave some blank
```

```
For some fields there will be a default value,
```

```
If you enter '.', the field will be left blank.
```

```
-----
```

```
Country Name (2 letter code) [US]:
```

```
State or Province Name (full name) [Wisconsin]:
```

```
Locality Name (eg, city) [Madison]:
```

```
Organization Name (eg, company) [University of Wisconsin -- Madison]:
```

```
Second Organization Name (eg, company) [Computer Sciences Department]:
```

```
Organizational Unit Name (eg, section) [HTCondor Project]:
```

```
Common Name (eg, YOUR name) []:ROOT CA
```

```
Email Address []:
```

Creating a CA

> Again, you can inspect the certificate

```
openssl x509 -noout -text -in root-ca.crt
```

Certificate:

Data:

Version: 3 (0x2)

Serial Number:

c7:99:e5:f7:c6:54:00:7a

Signature Algorithm: sha1WithRSAEncryption

Issuer: C=US, ST=Wisconsin, L=Madison, O=University of Wisconsin -

Madison, O=Computer Sciences Department, OU=HTCondor Project, CN=ROOT CA

...

Creating a CA

- › In the directory with the Root CA and openssl.cnf file, run these commands:

```
touch ca.db.index
```

```
echo 01 > ca.db.serial
```

Creating a Host Credential

- › Create the key and a signing request

```
openssl req -newkey rsa:1024 -keyout \  
host_omega.key -nodes -config \  
openssl.cnf -out host_omega.req
```

Creating a Host Certificate

Generating a 1024 bit RSA private key

.....++++++

.....++++++

writing new private key to 'host_omega.key'

You are about to be asked to enter information that will be incorporated into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank

For some fields there will be a default value,

If you enter '.', the field will be left blank.

Country Name (2 letter code) [US]:

State or Province Name (full name) [Wisconsin]:

Locality Name (eg, city) [Madison]:

Organization Name (eg, company) [University of Wisconsin -- Madison]:

Second Organization Name (eg, company) [Computer Sciences Department]:

Organizational Unit Name (eg, section) [HTCondor Project]:

Common Name (eg, YOUR name) []:**omega.cs.wisc.edu**

Email Address []:

Creating a Host Credential

```
openssl ca -config openssl.cnf -out \  
  host_omega.crt -infiles host_omega.req
```

Using configuration from openssl.cnf

Enter pass phrase for ./root-ca.key:

Check that the request matches the signature

Signature ok

Certificate Details:

...

Certificate is to be certified until May 01 14:31:09 2015
GMT (365 days)

Sign the certificate? [y/n]:**y**

Configuring HTCondor

- › Each host can now use it's own credential (example for omega.cs.wisc.edu)

```
AUTH_SSL_CLIENT_CAFILE = /path/to/root-ca.crt
AUTH_SSL_CLIENT_CERTFILE = /path/to/host_omega.crt
AUTH_SSL_CLIENT_KEYFILE = /path/to/host_omega.key

AUTH_SSL_SERVER_CAFILE = /path/to/root-ca.crt
AUTH_SSL_SERVER_CERTFILE = /path/to/host_omega.crt
AUTH_SSL_SERVER_KEYFILE = /path/to/host_omega.key
```

Creating a User Credential

```
openssl req -newkey rsa:1024 -keyout zmilller.key -config openssl.cnf -out zmilller.req
```

```
Generating a 1024 bit RSA private key
```

```
.....++++++
```

```
.....++++++
```

```
writing new private key to 'zmilller.key'
```

```
Enter PEM pass phrase:
```

```
Verifying - Enter PEM pass phrase: USER PASSWORD HERE
```

```
-----
```

You are about to be asked to enter information that will be incorporated into your certificate request.

What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank

For some fields there will be a default value,

If you enter '.', the field will be left blank.

```
-----
```

```
Country Name (2 letter code) [US]:
```

```
State or Province Name (full name) [Wisconsin]:
```

```
Locality Name (eg, city) [Madison]:
```

```
Organization Name (eg, company) [University of Wisconsin -- Madison]:
```

```
Second Organization Name (eg, company) [Computer Sciences Department]:
```

```
Organizational Unit Name (eg, section) [HTCondor Project]:
```

```
Common Name (eg, YOUR name) []:Zach Miller
```

```
Email Address []:zmilller@cs.wisc.edu
```

Creating a User Credential

```
openssl ca -config openssl.cnf -out zmilller.crt -infiles zmilller.req
```

```
Using configuration from openssl.cnf
```

```
Enter pass phrase for ./root-ca.key: CA PASSWORD
```

```
Check that the request matches the signature
```

```
Signature ok
```

```
Certificate Details:
```

```
...
```

```
Certificate is to be certified until May 1 14:31:09 2015 GMT (365  
days)
```

```
Sign the certificate? [y/n]:y
```

Mapping Users

› You could have one entry per user:

```
SSL
"C=US/ST=Wisconsin/L=Madison, O=University of Wisconsin -
Madison/O=Computer Sciences Department/OU=HTCondor Project/CN=Zach
Miller/emailAddress=zmiller@cs.wisc.edu"
    zmiller
```

```
SSL
"C=US/ST=Wisconsin/L=Madison, O=University of Wisconsin -
Madison/O=Computer Sciences Department/OU=HTCondor Project/CN=Todd
Tannenbaum/emailAddress=tannenba@cs.wisc.edu"
    tannenba
```

...

Etc.

Mapping Users

- › In the CERTIFICATE_MAPFILE, you can now add a rule to map all users by extracting the username from their email address:

```
SSL emailAddress=(.*)@cs.wisc.edu \1
```

Securing Everything

- › If all hosts and users have credentials, you can then enable SSL authentication for ALL communication, not just daemon-to-daemon. In the condor_config:

```
SEC_DEFAULT_AUTHENTICATION = REQUIRED  
SEC_DEFAULT_AUTHENTICATION_METHODS = SSL
```

More Information

- › Ask me, email me!
- › You can find more detailed information, and examples using multi-level CAs here:

<http://pages.cs.wisc.edu/~zmiller/ca-howto/>