MyProxyClient Documentation

Release 1.4.0

P J Kershaw

February 05, 2015

This a pure* Python implementation of a client to the MyProxy Credential Management Server (http://grid.ncsa.uiuc.edu/myproxy/)

- i.e. MyProxy C client libraries are not required for this package.

It uses pyOpenSSL to make an SSL connection to the server following the messaging interface as outlined in: http://grid.ncsa.uiuc.edu/myproxy/protocol/

The code is based on an original program myproxy_logon by Tom Uram of ANL.

CHAPTER

ONE

RELEASES

1.4.0

- Fix for SSL to use TLS instead of SSLv3 to address POODLE vulnerability
- Fix for SSL verification for PyOpenSSL version 0.14 - v1.3.1 was broken because it passed the call back method to OpenSSL using verification classes' `__call__` method.

Tested on CentOS 6.4.

1.3.1

- Fix to `MyProxyClient.writeProxyFile` and `MyProxyClient.readProxyFile` to correctly pick-up overridden file setting. Thanks to Nicolas Carenton, IPSL.

CHAPTER TWO

TESTS

Unit test module with test files is in `test/`. See the README in that directory.

Documentation

Sphinx generated documentation is available in `documentation/`. Run the `Makefile` to regenerate if required.

Thanks

- to OMII-UK for funding development of NDG Security (2007-2008)
- Tom Uram who wrote the `myproxy_logon` program on which this package is based.

Contents:

2.1 MyProxy Client Module

class `myproxy.client.`**`MyProxyClient`**(*cfgFilePath=None*, ***prop*)

Bases: `object`

MyProxy client interface

Based on protocol definitions in:

http://grid.ncsa.uiuc.edu/myproxy/protocol/

Variables **`SSL_METHOD`** – encryption method used for connecting to MyProxy server

•set to TLS version 1

Variables

- **`MYPROXY_SERVER_ENVVARNAME`** – server environment variable name
- **`MYPROXY_SERVER_PORT_ENVVARNAME`** – port environment variable name
- **`MYPROXY_SERVER_DN_ENVVARNAME`** – server certificate Distinguished Name

environment variable name

Parameters **`GLOBUS_LOCATION_ENVVARNAME`** (*string*) – 'GLOBUS_LOCATION' environment variable

name

Variables

- **`GET_CMD`** – get command string

- **INFO_CMD** – info command string
- **DESTROY_CMD** – destroy command string
- **CHANGE_PASSPHRASE_CMD** – command string to change cred pass-phrase
- **STORE_CMD** – store command string
- **GET_TRUST_ROOTS_CMD** – get trust roots command string

Parameters **TRUSTED_CERTS_FIELDNAME** (*string*) – field name in get trust roots response for

trusted certificate file names

Parameters **TRUSTED_CERTS_FILEDATA_FIELDNAME_PREFIX** (*string*) – field name prefix in get

trust roots response for trusted certificate file contents

Variables

- **HOSTCERT_SUBDIRPATH** – sub-directory path host certificate (as tuple)
- **HOSTKEY_SUBDIRPATH** – sub-directory path to host key (as tuple)
- **PRIKEY_NBITS** – default number of bits for private key generated
- **MESSAGE_DIGEST_TYPE** – message digest type is MD5
- **SERVER_RESP_BLK_SIZE** – block size for retrievals from server
- **MAX_RECV_TRIES** – maximum number of retrievals of size

SERVER_RESP_BLK_SIZE before this client gives up

Variables

- **DEF_PROXY_FILEPATH** – default location for proxy file to be written to
- **PROXY_FILE_PERMISSIONS** – file permissions returned proxy file is

created with

Variables **PROPERTY_DEFAULTS** – sets permissable element names for MyProxy config

file

Variables **ROOT_USERNAME** – root username - used to determine output directory

for trust roots

Variables **ROOT_TRUSTROOT_DIR** – default trust root directory if running as root

user

Variables **USER_TRUSTROOT_DIR** – default trust root directory for users other

than root

Variables **X509_CERT_DIR_ENVVARNAME** – environment variable name 'X509_CERT_DIR',

which if set points to the location of the trust roots

Variables **X509_USER_PROXY_ENVVARNAME** – environment variable name

'X509_USER_PROXY' if set points to the output location of the output EEC / Proxy certificate. Not currently used by this class, included for reference only

caCertDir

trust roots directory containing PEM encoded CA certificates to validate MyProxy server certificate

changePassphrase (*username*, *passphrase*, *newPassphrase*, *sslCertFile=None*, *sslKeyFile=None*, *sslKeyFilePassphrase=None*)

change pass-phrase protecting the credentials for a given username

Raises

- **MyProxyClientGetError** –
- **MyProxyClientRetrieveError** –

Parameters

- **username** – username of credential
- **passphrase** – existing pass-phrase for credential
- **newPassphrase** – new pass-phrase to replace the existing one.
- **sslCertFile** – certificate used for client authentication with

the MyProxy server SSL connection. This ID will be set as the owner of the stored credentials. Only the owner can later remove credentials with myproxy-destroy or the destroy method. If not set, this argument defaults to $GLOBUS_LOCATION/etc/hostcert.pem :param sslKeyFile: corresponding private key file. See explanation for sslCertFile :param sslKeyFilePassphrase: passphrase for sslKeyFile. Omit if the private key is not password protected. @return none

destroy (*username*, *sslCertFile=None*, *sslKeyFile=None*, *sslKeyFilePassphrase=None*)

destroy credentials from the server for a given username

Raises

- **MyProxyClientGetError** –
- **MyProxyClientRetrieveError** –

Parameters

- **username** – username selected for credential
- **sslCertFile** – certificate used for client authentication with

the MyProxy server SSL connection. This ID will be set as the owner of the stored credentials. Only the owner can later remove credentials with myproxy-destroy or the destroy method. If not set, this argument defaults to $GLOBUS_LOCATION/etc/hostcert.pem :param sslKeyFile: corresponding private key file. See explanation for sslCertFile :param sslKeyFilePassphrase: passphrase for sslKeyFile. Omit if the private key is not password protected. @return none

getDelegation (**arg*, ***kw*)

Retrieve proxy cert for user - same as logon

getTrustRoots (*username=''*, *passphrase=''*, *writeToCACertDir=False*, *bootstrap=False*)

Get trust roots for the given MyProxy server

Parameters

- **username** (*basestring*) – username (optional)
- **passphrase** (*basestring*) – pass-phrase (optional)

server

Parameters **writeToCACertDir** (*bool*) – if set to True, write the retrieved trust roots

out to the directory specified by the "caCertDir" attribute

Parameters **bootstrap** (*bool*) – If set to True, bootstrap trust roots i.e. connect to

MyProxy server without verification of the server's SSL certificate against any CA certificates. Set to False, for default behaviour: verify server SSL certificate against CA certificates held in location set by the "caCertDir" attribute.

@return: trust root files as a dictionary keyed by file name with each item value set to the file contents @rtype: dict

hostname
hostname of MyProxy server

info (*username*, *sslCertFile=None*, *sslKeyFile=None*, *sslKeyFilePassphrase=None*)
return True/False whether credentials exist on the server for a given username

Raises

- **MyProxyClientGetError** –
- **MyProxyClientRetrieveError** –

Parameters

- **username** (*string*) – username selected for credential
- **sslCertFile** (*string*) – certificate used for client authentication with

the MyProxy server SSL connection. This ID will be set as the owner of the stored credentials. Only the owner can later remove credentials with myproxy-destroy or the destroy method. If not set, this argument defaults to $GLOBUS_LOCATION/etc/hostcert.pem :type sslKeyFile: string :param sslKeyFile: corresponding private key file. See explanation for sslCertFile :type sslKeyFilePassphrase: string :param sslKeyFilePassphrase: passphrase for sslKeyFile. Omit if the private key is not password protected.

classmethod locateClientCredentials (*enableTmpFileLoc=False*)
Find the location of a client certificate and private key to use to authenticate with the server based on the various default locations that MyProxy/Globus support

Parameters **enableTmpFileLoc** – enable setting based on /tmp/x509up_<uid>,

defaults to False :type enableTmpFileLoc: bool @return: private key and certificate file location to use based on the current environment @rtype: tuple

logon (*username*, *passphrase*, *credname=None*, *lifetime=None*, *keyPair=None*, *certReq=None*, *nBitsForKey=4096*, *bootstrap=False*, *updateTrustRoots=False*, *authnGetTrustRootsCall=False*, *sslCertFile=None*, *sslKeyFile=None*, *sslKeyFilePassphrase=None*)
Retrieve a proxy credential from a MyProxy server

Exceptions: MyProxyClientGetError, MyProxyClientRetrieveError

Parameters

- **username** (*basestring*) – username of credential
- **passphrase** (*basestring*) – pass-phrase for private key of credential held on

server

Parameters **credname** (*string / None type*) – optional credential name - provides additional means

to specify credential to be retrieved

Parameters

- **lifetime** (*int*) – lifetime for generated certificate
- **keyPair** (*OpenSSL.crypto.PKey*) – Public/Private key pair. This is ignored if a

certificate request is passed via the certReq keyword

Parameters **`certReq`** (*string*) – ASN1 format certificate request, if none set, one is

created along with a key pair

Parameters **`nBitsForKey`** (*int*) – number of bits to use when generating key pair,

defaults to the PRIKEY_NBITS class variable setting. This keyword is ignored if a key pair is passed in from an external source via the keyPair keyword

@rtype: tuple @return credentials as strings in PEM format: the user certificate, it's private key and the issuing certificate. The issuing certificate is only set if the user certificate is a proxy

Parameters **`bootstrap`** (*bool*) – If set to True, bootstrap trust roots i.e. connect to

MyProxy server without verification of the server's SSL certificate against any CA certificates. Set to False, for default behaviour: verify server SSL certificate against CA certificates held in location set by the "caCertDir" attribute. If bootstrap is set, updateTrustRoots will be forced to True also

Parameters

- **`updateTrustRoots`** (*bool*) – set to True to update the trust roots
- **`authnGetTrustRootsCall`** (*bool*) – pass username and password to

getTrustRoots call. getTrustRoots is invoked if the "updateTrustRoots" or "bootstrap" keywords are set. This is not recommended for bootstrap since in this case the server is NOT authenticated by this client.

Parameters **`sslCertFile`** – applies to SSL client based authentication -

alternative to username/pass-phrase based. This certificate is used for authentication with MyProxy server over the SSL connection. If not set, this argument defaults to $GLOBUS_LOCATION/etc/hostcert.pem :param sslKeyFile: corresponding private key file. See explanation for sslCertFile :param sslKeyFilePassphrase: passphrase for sslKeyFile. Omit if the private key is not password protected.

`openSSLConfFilePath`
file path for OpenSSL config file

`openSSLConfig`
OpenSSLConfig object

`parseConfig` (*cfg*, *section='DEFAULT'*)
Extract parameters from _cfg config object

`port`
Port number for MyProxy server

`proxyCertLifetime`
Default proxy cert. lifetime (seconds) used in logon request

`proxyCertMaxLifetime`
Default max. lifetime allowed for Proxy Certificate retrieved - used by store method

`put` (*username*, *passphrase*, *userCertFile*, *userKeyFile*, *lifetime=None*, *sslCertFile=None*, *sslKeyFile=None*, *sslKeyFilePassphrase=None*)
Store a proxy credential on the server

Unfortunately this method is not implemented as it requires the creation of a proxy certificate by the client but PyOpenSSL doesn't currently support the required proxyCertInfo X.509 certificate extension

Raises NotImplementedError see above

Parameters

- **`username`** (*string*) – username selected for new credential
- **`passphrase`** (*string*) – pass-phrase for new credential. This will be used

by the server to authenticate later requests. IT must be at least 6 characters. The server may impose other restrictions too depending on its configuration. :type certFile: string :param certFile: user's X.509 proxy certificate in PEM format :type keyFile: string :param keyFile: equivalent private key file in PEM format :type sslCertFile: string :param sslCertFile: certificate used for client authentication with the MyProxy server SSL connection. If not set, this argument defaults to $GLOBUS_LOCATION/etc/hostcert.pem :type sslKeyFile: string :param sslKeyFile: corresponding private key file. See explanation for sslCertFile :type sslKeyFilePassphrase: string :param sslKeyFilePassphrase: passphrase for sslKeyFile. Omit if the private key is not password protected. :type lifetime: int / None :param lifetime: the maximum lifetime allowed for retrieved proxy credentials in seconds. defaults to proxyCertMaxLifetime attribute value

classmethod `readProxyFile` (*filePath=None*)

Read proxy cert file following the format used by myproxy-logon - proxy, cert, private key, user cert.

@rtype: tuple @return: tuple containing proxy cert, private key, user cert

`serverDN`

Distinguished Name for MyProxy Server Certificate

`serverSSLCertVerify`

Class with a __call__ method which is passed to the SSL context to verify the peer (MyProxy server) certificate in the SSL handshake between this client and the MyProxy server

`setDefaultCACertDir`()

Make default trust root setting - the directory containing the CA certificates for verifying the MyProxy server's SSL certificate.

The setting is made by using standard Globus defined locations and environment variable settings

`store`(*username*, *passphrase*, *certFile*, *keyFile*, *sslCertFile=None*, *sslKeyFile=None*, *sslKeyFilePassphrase=None*, *lifetime=None*, *force=True*)

Upload credentials to the server

Raises

- **`MyProxyClientGetError`** –
- **`MyProxyClientRetrieveError`** –

Parameters

- **`username`** (*string*) – username selected for new credential
- **`passphrase`** (*string*) – pass-phrase for new credential. This is the pass

phrase which protects keyfile. :type certFile: string :param certFile: user's X.509 certificate in PEM format :type keyFile: string :param keyFile: equivalent private key file in PEM format :type sslCertFile: string :param sslCertFile: certificate used for client authentication with the MyProxy server SSL connection. This ID will be set as the owner of the stored credentials. Only the owner can later remove credentials with myproxy-destroy or the destroy method. If not set, this argument defaults to $GLOBUS_LOCATION/etc/hostcert.pem or if this is not set, certFile :type sslKeyFile: string :param sslKeyFile: corresponding private key file. See explanation for sslCertFile :type sslKeyFilePassphrase: string :param sslKeyFilePassphrase: passphrase for sslKeyFile. Omit if the private key is not password protected. Nb. keyFile is expected to be passphrase protected as this will be the passphrase used for logon / getDelegation. :type Force: bool :param force: set to True to overwrite any existing creds with the same username. If, force=False a check is made with a call to info. If creds already, exist exit without proceeding

classmethod `writeProxyFile` (*proxyCert*, *proxyPriKey*, *userX509Cert*, *filePath=None*)

Write out proxy cert to file in the same way as myproxy-logon - proxy cert, private key, user cert. Nb. output from logon can be passed direct into this method

Parameters

- **`proxyCert`** (*string*) – proxy certificate

- **`proxyPriKey`** (*string*) – private key for proxy
- **`userX509Cert`** (*string*) – user certificate which issued the proxy
- **`filePath`** (*string*) – set to override the default filePath

CHAPTER

THREE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*