auth.credential Documentation

Release 1.0

Massimo Paladin

November 28, 2013

Contents

Contents:

CHAPTER 1

Credential Abstraction

`Credential()` - abstraction of a credential

1.1 Synopsis

Example:

```python
import auth.credential as credential
from auth.credential.modules.plain import Plain

try:
    from urllib.request import Request
except ImportError:
    from urllib2 import Request

# creation
option = {'scheme' : 'plain', 'name' : 'system', 'pass' : 'manager'}
cred = credential.new(**option)
assert option['scheme'] == cred['scheme']
assert option['pass'] == cred['pass']
# idem directly using the sub-class
del(option['scheme'])
cred = Plain(**option)

# access the credential attributes
if (cred.scheme == "plain"):
    print("user name is %s", cred.name)

### HTTP examples

# use the prepare() method to get ready-to-use data
headers = {"Authorization" : cred.prepare('HTTP.Basic')}
req = Request("http://localhost", headers=headers)

### stomppy examples

import stomp
```

```
# plain example
host_and_ports = [ ('localhost', 61613) ]
params = cred.prepare('stomppy.plain')
conn = stomp.Connection(host_and_ports, **params)

# x509 example
host_and_ports = [ ('localhost', 61612) ]
option = {'scheme' : 'x509', 'key' : 'path/to/key', 'cert' : 'path/to/cert'}
cred = credential.new(**option)
params = cred.prepare('stomppy.x509')
conn = stomp.Connection(host_and_ports, **params)
```

1.2 Description

This module offers an abstraction of a credential, i.e. something that can be used to authenticate. It allows the creation and manipulation of credentials. In particular, it defines a standard string representation (so that credentials can be given to external programs as command line options), a standard structured representation (so that credentials can be stored in structured configuration files or using JSON) and "preparators" that can transform credentials into ready-to-use data for well known targets.

Different authentication schemes (aka credential types) are supported. This package currently supports *none*, *plain* and *x509* but others can be added by providing the supporting code in a separate module.

For a given scheme, a credential is represented by an object with a fixed set of string attributes. For instance, the *plain* scheme has two attributes: *name* and *pass*. More information is provided by the scheme specific module, for instance Plain.

1.3 String representation

The string representation of a credential is made of its scheme followed by its attributes as key=value pairs, seperated by space.

For instance, for the *none* scheme with no attributes:

```
none
```

And the the *plain* scheme with a name and password:

```
plain name=system pass=manager
```

If needed, the characters can be URI-quoted, see urllib. All non-alphanumerical characters should be escaped to avoid parsing ambiguities.

The string representation is useful to give a program through its command line options. For instance:

```
myprog --uri http://foo:80 --auth "plain name=system pass=manager"
```

1.4 Structured representation

The structured representation of a credential is made of its scheme and all its attributes as a string table.

Here is for instance how it could end up using JSON:

```
{"scheme":"plain","name":"system","pass":"manager"}
```

The same information could be stored in a configuration file.

`auth.credential.credential.`**`new`**`(`***option*`)`
Return a Credential object according to the option passed and the given scheme.

`auth.credential.credential.`**`parse`**`(`*string*`)`
Parse a string containing authentication information and return a dictionary.

Credential Modules

Credential modules.

Copyright (C) 2013 CERN

2.1 None Credential

`Non()` - abstraction of a *none* credential

2.1.1 Description

This helper module for Credential implements a *none* credential, that is the absence of authentication credential.

It does not support any attributes.

Copyright (C) 2013 CERN

2.2 Plain Credential

`Plain()` - abstraction of a *plain* credential

2.2.1 Description

This helper module for Credential implements a *plain* credential, that is a pair of name and clear text password.

It supports the following attributes:

name the (usually user) name

pass the associated (clear text) password

Copyright (C) 2013 CERN

2.3 X509 Credential

`X509()` - abstraction of an X.509 credential

2.3.1 Description

This helper module for Credential implements an X.509 credential, see http://en.wikipedia.org/wiki/X.509.

It supports the following attributes:

cert the path of the file holding the certificate

key the path of the file holding the private key

pass the pass-phrase protecting the private key (optional)

ca the path of the directory containing trusted certificates (optional)

Copyright (C) 2013 CERN

CHAPTER 3

Errors

Errors used in the module.

exception `auth.credential.error.`**`InvalidCredential`**

Raised when errors occurs during credentials handling.

This module offers an abstraction of a credential, i.e. something that can be used to authenticate. It allows the creation and manipulation of credentials. In particular, it defines a standard string representation (so that credentials can be given to external programs as command line options), a standard structured representation (so that credentials can be stored in structured configuration files or using JSON) and "preparators" that can transform credentials into ready-to-use data for well known targets.

You can download the module at the following link: http://pypi.python.org/pypi/auth.credential/

An Perl implementation of the same credential abstraction is available in CPAN: http://search.cpan.org/dist/Authen-Credential/

CHAPTER 4

Indices and tables

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

Python Module Index