# Author: Trevor Perrin

# See the LICENSE file for legal information regarding use of this file.

from .compat import \*

import binascii

#This code is shared with tackpy (somewhat), so I'd rather make minimal

#changes, and preserve the use of a2b\_base64 throughout.

def dePem(s, name):

"""Decode a PEM string into a bytearray of its payload.

The input must contain an appropriate PEM prefix and postfix

based on the input name string, e.g. for name="CERTIFICATE"::

-----BEGIN CERTIFICATE-----

MIIBXDCCAUSgAwIBAgIBADANBgkqhkiG9w0BAQUFADAPMQ0wCwYDVQQDEwRUQUNL

...

KoZIhvcNAQEFBQADAwA5kw==

-----END CERTIFICATE-----

The first such PEM block in the input will be found, and its

payload will be base64 decoded and returned.

"""

prefix = "-----BEGIN %s-----" % name

postfix = "-----END %s-----" % name

start = s.find(prefix)

if start == -1:

raise SyntaxError("Missing PEM prefix")

end = s.find(postfix, start+len(prefix))

if end == -1:

raise SyntaxError("Missing PEM postfix")

s = s[start+len("-----BEGIN %s-----" % name) : end]

retBytes = a2b\_base64(s) # May raise SyntaxError

return retBytes

def dePemList(s, name):

"""Decode a sequence of PEM blocks into a list of bytearrays.

The input must contain any number of PEM blocks, each with the appropriate

PEM prefix and postfix based on the input name string, e.g. for

name="TACK BREAK SIG". Arbitrary text can appear between and before and

after the PEM blocks. For example::

Created by TACK.py 0.9.3 Created at 2012-02-01T00:30:10Z

-----BEGIN TACK BREAK SIG-----

ATKhrz5C6JHJW8BF5fLVrnQss6JnWVyEaC0p89LNhKPswvcC9/s6+vWLd9snYTUv

YMEBdw69PUP8JB4AdqA3K6Ap0Fgd9SSTOECeAKOUAym8zcYaXUwpk0+WuPYa7Zmm

SkbOlK4ywqt+amhWbg9txSGUwFO5tWUHT3QrnRlE/e3PeNFXLx5Bckg=

-----END TACK BREAK SIG-----

Created by TACK.py 0.9.3 Created at 2012-02-01T00:30:11Z

-----BEGIN TACK BREAK SIG-----

ATKhrz5C6JHJW8BF5fLVrnQss6JnWVyEaC0p89LNhKPswvcC9/s6+vWLd9snYTUv

YMEBdw69PUP8JB4AdqA3K6BVCWfcjN36lx6JwxmZQncS6sww7DecFO/qjSePCxwM

+kdDqX/9/183nmjx6bf0ewhPXkA0nVXsDYZaydN8rJU1GaMlnjcIYxY=

-----END TACK BREAK SIG-----

All such PEM blocks will be found, decoded, and return in an ordered list

of bytearrays, which may have zero elements if not PEM blocks are found.

"""

bList = []

prefix = "-----BEGIN %s-----" % name

postfix = "-----END %s-----" % name

while 1:

start = s.find(prefix)

if start == -1:

return bList

end = s.find(postfix, start+len(prefix))

if end == -1:

raise SyntaxError("Missing PEM postfix")

s2 = s[start+len(prefix) : end]

retBytes = a2b\_base64(s2) # May raise SyntaxError

bList.append(retBytes)

s = s[end+len(postfix) : ]

def pem(b, name):

"""Encode a payload bytearray into a PEM string.

The input will be base64 encoded, then wrapped in a PEM prefix/postfix

based on the name string, e.g. for name="CERTIFICATE"::

-----BEGIN CERTIFICATE-----

MIIBXDCCAUSgAwIBAgIBADANBgkqhkiG9w0BAQUFADAPMQ0wCwYDVQQDEwRUQUNL

...

KoZIhvcNAQEFBQADAwA5kw==

-----END CERTIFICATE-----

"""

s1 = b2a\_base64(b)[:-1] # remove terminating \n

s2 = ""

while s1:

s2 += s1[:64] + "\n"

s1 = s1[64:]

s = ("-----BEGIN %s-----\n" % name) + s2 + \

("-----END %s-----\n" % name)

return s

def pemSniff(inStr, name):

searchStr = "-----BEGIN %s-----" % name

return searchStr in inStr