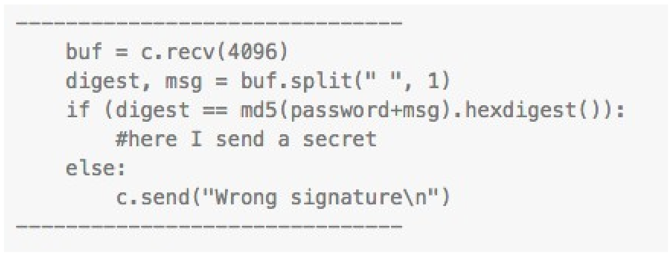
HLE (Hash Length Extension)

Problem: We have found one authorized message “c974b779d095f5772a36e2139276ffdc testing connection”. Can you find another authorized message and send it to nc 127.0.0.1 41300?

Given:



```
-------------------------------
    buf = c.recv(4096)
    digest, msg = buf.split(" ", 1)
    if (digest == md5(password+msg).hexdigest()):
        #here I send a secret
    else:
        c.send("Wrong signature\n")
-------------------------------
```

Hint: Can you extend your luck?

Intro:
This is a hash length extension problem. Luckily there are tools to help solve this so it does not have to be brute forced. To learn more about hash length extension at https://en.wikipedia.org/wiki/Length_extension_attack.

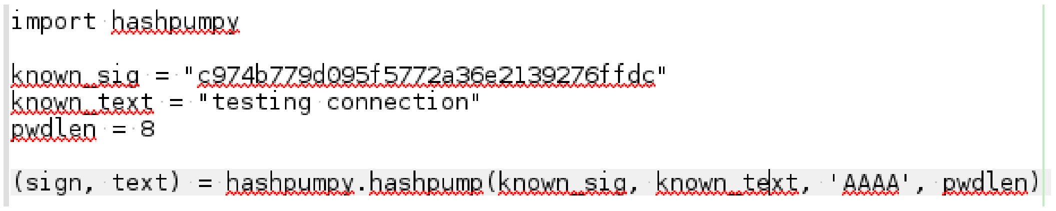
There are a few tools out there to help with this type of attack. The one that will be used for this write up is called hashpump and can be found at https://github.com/bwall/HashPump.

Steps:
1) We are given a valid pair of message and digest. The message is “testing connection” and the digest is “c974b779d095f5772a36e2139276ffdc”. The given code shows the digest is a MD5 hash of the concatenation of the message and a secret password.

Since we don’t know the password, we need to generate a new pair of digest and message based on the given pair of digest and message. The new digest must match the MD5 hash of the new message and the secret password.

2) Now we utilize the hashbump python library to generate and test new pairs of digest and message. An example python script is below. The hashpump function takes four arguments. The first and the second arguments are the known digest and known message. The third argument is a string to added to the known message. The fourth argument is a guessed length of the password, since we don't know the password. The function returns a pair of digest and message.

In this example, we guess that the password length is 8. In the next step, we will see that our guess is wrong. It is important to know that in the hash length extension attack, we do not need to know the exact password, but we need to guess the length of the password.

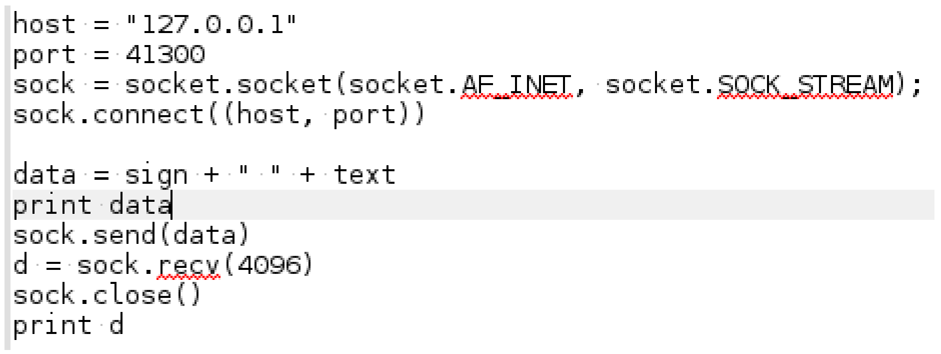


```
import hashpumpy

known_sig = "c974b779d095f5772a36e2139276ffdc"
known_text = "testing connection"
pwdlen = 8

(sign, text) = hashpumpy.hashpump(known_sig, known_text, 'AAAA', pwdlen)
```

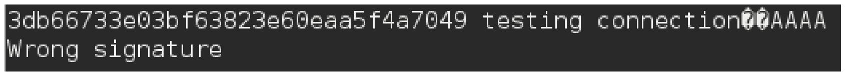
3) Then, we connect the service and send the newly generated pair of digest and message to the service to see if they are valid. The script is below.



```
host = "127.0.0.1"
port = 41300
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM);
sock.connect((host, port))

data = sign + " " + text
print data
sock.send(data)
d = sock.recv(4096)
sock.close()
print d
```

The result below shows that the pair of digest and message is not valid.



```
3db66733e03bf63823e60eaa5f4a7049 testing connection��AAAA
Wrong signature
```

4) The reason they are not valid is that the guessed length of the password is wrong. Hence, we need to try other lengths of the password. Sure we make a script to do this…