I’m currently investigating the use of the HTTP-Server capabilities of Python. While simple HTTP works like a charm, I was wondering on how to embed SSL into that.

I came across an ActiveState recipe by Sebastien Martini located here: [[http://code.activestate.com/recipes/442473-simple-http-server-supporting-ssl-secure-communica/]]

However, that thing didn’t work. At least not on my mac. What did work was using the embedded ssl library (available from Python 2.6 on) rather than the OpenSSL library.

This is a reworked version of the script, that worked:

```
'''
SimpleSecureHTTPServer.py - simple HTTP server supporting SSL.

- replace fpem with the location of your .pem server file.
- the default port is 443.

usage: python SimpleSecureHTTPServer.py
'''
import socket, os
from SocketServer import BaseServer
from BaseHTTPServer import HTTPServer
from SimpleHTTPServer import SimpleHTTPRequestHandler
import ssl

class SecureHTTPServer(HTTPServer):
def __init__(self, server_address, HandlerClass):
BaseServer.__init__(self, server_address, HandlerClass)
fpem = 'server.pem'
self.socket = ssl.SSLSocket(
socket.socket(self.address_family,self.socket_type),
keyfile = fpem,
certfile = fpem
)

self.server_bind()
self.server_activate()
```

```
class SecureHTTPRequestHandler(SimpleHTTPRequestHandler):
def setup(self):
self.connection = self.request
self.rfile = socket._fileobject(self.request, "rb", self.rbufsize)
self.wfile = socket._fileobject(self.request, "wb", self.wbufsize)

def test(HandlerClass = SecureHTTPRequestHandler,
ServerClass = SecureHTTPServer):
server_address = ('', 8443) # (address, port)
httpd = ServerClass(server_address, HandlerClass)
sa = httpd.socket.getsockname()
print "Serving HTTPS on", sa[0], "port", sa[1], "..."
httpd.serve_forever()

if __name__ == '__main__':
test()
```

Mind you, the script is courtesy of ActiveState. The modifications from OpenSSL to ssl were done by me.