Example of a client and server who both validate each other’s certificates:

For this example, we’ll create Self-signed server and client certificates. Normally you’d use a server certificate from a Certificate Authority such as [Let’s Encrypt](https://letsencrypt.org/), and would setup your own Certificate Authority so you can sign and revoke client certificates.

Create server certificate:

openssl req -new -newkey rsa:2048 -days 365 -nodes -x509 -keyout server.key -out server.crt

Make sure to enter ‘example.com’ for the Common Name.

Next, generate a client certificate:

openssl req -new -newkey rsa:2048 -days 365 -nodes -x509 -keyout client.key -out client.crt

The Common Name for the client certificate doesn’t really matter.

Client code:

#!/usr/bin/python3

import socket

import ssl

host\_addr = '127.0.0.1'

host\_port = 8082

server\_sni\_hostname = 'example.com'

server\_cert = 'server.crt'

client\_cert = 'client.crt'

client\_key = 'client.key'

context = ssl.create\_default\_context(ssl.Purpose.SERVER\_AUTH, cafile=server\_cert)

context.load\_cert\_chain(certfile=client\_cert, keyfile=client\_key)

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

conn = context.wrap\_socket(s, server\_side=False, server\_hostname=server\_sni\_hostname)

conn.connect((host\_addr, host\_port))

print("SSL established. Peer: {}".format(conn.getpeercert()))

print("Sending: 'Hello, world!")

conn.send(b"Hello, world!")

print("Closing connection")

conn.close()

Server code:

#!/usr/bin/python3

import socket

from socket import AF\_INET, SOCK\_STREAM, SO\_REUSEADDR, SOL\_SOCKET, SHUT\_RDWR

import ssl

listen\_addr = '127.0.0.1'

listen\_port = 8082

server\_cert = 'server.crt'

server\_key = 'server.key'

client\_certs = 'client.crt'

context = ssl.create\_default\_context(ssl.Purpose.CLIENT\_AUTH)

context.verify\_mode = ssl.CERT\_REQUIRED

context.load\_cert\_chain(certfile=server\_cert, keyfile=server\_key)

context.load\_verify\_locations(cafile=client\_certs)

bindsocket = socket.socket()

bindsocket.bind((listen\_addr, listen\_port))

bindsocket.listen(5)

while True:

print("Waiting for client")

newsocket, fromaddr = bindsocket.accept()

print("Client connected: {}:{}".format(fromaddr[0], fromaddr[1]))

conn = context.wrap\_socket(newsocket, server\_side=True)

print("SSL established. Peer: {}".format(conn.getpeercert()))

buf = b'' # Buffer to hold received client data

try:

while True:

data = conn.recv(4096)

if data:

# Client sent us data. Append to buffer

buf += data

else:

# No more data from client. Show buffer and close connection.

print("Received:", buf)

break

finally:

print("Closing connection")

conn.shutdown(socket.SHUT\_RDWR)

conn.close()

Output from the server looks like this:

$ python3 ./server.py

Waiting for client

Client connected: 127.0.0.1:51372

SSL established. Peer: {'subject': ((('countryName', 'AU'),),

(('stateOrProvinceName', 'Some-State'),), (('organizationName', 'Internet

Widgits Pty Ltd'),), (('commonName', 'someclient'),)), 'issuer':

((('countryName', 'AU'),), (('stateOrProvinceName', 'Some-State'),),

(('organizationName', 'Internet Widgits Pty Ltd'),), (('commonName',

'someclient'),)), 'notBefore': 'Jun 1 08:05:39 2018 GMT', 'version': 3,

'serialNumber': 'A564F9767931F3BC', 'notAfter': 'Jun 1 08:05:39 2019 GMT'}

Received: b'Hello, world!'

Closing connection

Waiting for client

Output from the client:

$ python3 ./client.py

SSL established. Peer: {'notBefore': 'May 30 20:47:38 2018 GMT', 'notAfter':

'May 30 20:47:38 2019 GMT', 'subject': ((('countryName', 'NL'),),

(('stateOrProvinceName', 'GLD'),), (('localityName', 'Ede'),),

(('organizationName', 'Electricmonk'),), (('commonName', 'example.com'),)),

'issuer': ((('countryName', 'NL'),), (('stateOrProvinceName', 'GLD'),),

(('localityName', 'Ede'),), (('organizationName', 'Electricmonk'),),

(('commonName', 'example.com'),)), 'version': 3, 'serialNumber':

'CAEC89334941FD9F'}

Sending: 'Hello, world!

Closing connection