

“Chat” timing covert channel tutorial-ish

by varying the timing of messages going from a server to a client, we can covertly transmit a message
the overt message can be anything and is what's being received by the client
the timing in between pieces of that message coming in “hides” a covert message; e.g.,:
0.01s delay in between receiving parts of the message → 0
0.1s delay in between receiving parts of the message → 1

we could setup a server that delivers the text of a book, one line at a time
in between each line, we could add a slight delay to allow a 0 or 1 to be transmitted
how many lines would it take to covertly transmit a single character of a covert message?
since we're using delays in between lines, and a character is 1 byte: 9 lines
note the low bandwidth!
we could, instead, deliver the book one character at a time
in between each character, we introduce a slight delay (0 or 1)
we now need 9 characters of the book to covertly transmit one character of a covert message
that's better

we could carefully craft the overt and covert messages
so that transmission of the overt message is done just as the overt message is
but that's not reasonable
so repeat the covert message for as long as needed to completely transmit the overt message
but how do we know when the covert message is done?
it's repeated, so that should be evident
but we could append something to the end of it (e.g., “EOF”)

we could do the same with the overt message
so that the client knows when the transmission is done and can disconnect

server

we can use Python's socket library:

```
import socket
```

to create a socket:

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

to bind the socket to all interfaces on some **port**:

```
port = 1337
```

```
s.bind("", port)
```

to listen for connections:

```
s.listen(0)
```

probably best to put all of the above in a try/except

to accept a connecting client:

```
c, addr = s.accept()
```

note that **c** becomes a socket representing the connection of the client to the server

we could multi-thread the server to accept other clients (using **s**) simultaneously

addr is the address of the connecting client

to send a message, one character at a time:

```
msg = "Some message..."
```

```

    for i in msg:
        c.send(i)
but we want to introduce a delay in between characters:
import time

```

```

msg = "Some message..."
for i in msg:
    c.send(i)
    if (<zero>):
        time.sleep(0.01)
    else:
        time.sleep(0.1)

```

better (using constants):

```

import time

```

```

ZERO = 0.01
ONE = 0.1

msg = "Some message..."
for i in msg:
    c.send(i)
    if (<zero>):
        time.sleep(ZERO)
    else:
        time.sleep(ONE)
c.send("EOF")
c.close()

```

but how do we know to transmit a 0 or 1 of the covert message?

first, the covert (hidden) message:

```

covert = "secret" + "EOF"

```

let's convert it to binary

```

from binascii import hexlify

```

```

covert = "secret" + "EOF"
covert_bin = ""
for i in covert:
    # convert each character to a full byte
    # hexlify converts ASCII to hex
    # int converts the hex to a decimal integer
    # bin provides its binary representation (with a 0b
    # prefix that must be removed)
    # that's the [2:] (return the string from the third
    # character on)
    # zfill left-pads the bit string with 0s to ensure a
    # full byte
    covert_bin += bin(int(hexlify(i), 16))[2:].zfill(8)

```

e.g., if covert == "ABC"
then covert_bin == "010000010100001001000011"

so now:

```
import time

ZERO = 0.01
ONE = 0.1

msg = "Some message..."
n = 0
for i in msg:
    c.send(i)
    if (covert_bin[n] == "0"):
        time.sleep(ZERO)
    else:
        time.sleep(ONE)
    n = (n + 1) % len(covert_bin)
c.send("EOF")
c.close()
```

client

again, we can use Python's socket library:

```
import socket
```

to create a socket:

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

to connect to the server on some **ip** and **port**:

```
s.connect((ip, port))
```

to receive data (until the string "EOF"):

```
import sys
```

```
data = s.recv(4096)
while (data.rstrip("\n") != "EOF"):
    sys.stdout.write(data)
    sys.stdout.flush()
s.close()
```

note **sys.stdout.write** instead of **print** or **print**,
why?

and why **sys.stdout.flush**?

what about timing in between each received character?

you know, to determine the covert message?

grab the current time (say into **t0**)

receive data (a character)

grab the current time (say into **t1**)

calculate the time elapsed from **t0** to **t1**

0.01s → 0

0.1s → 1

```

from time import time

covert_bin = ""
t0 = time()
data = s.recv(4096)
t1 = time()
delta = round(t1 - t0, 2)
if (delta >= ONE):
    covert_bin += "1"
else:
    covert_bin += "0"

```

so now we have a bit string in **covert_bin**
we simply need to convert that to ASCII

```

covert = ""
i = 0
while (i < len(covert_bin)):
    # process one byte at a time
    b = covert_bin[i:i + 8]
    # convert it to ASCII
    n = int("0b{}".format(b), 2)
    try:
        covert += unhexlify("{0:x}".format(n))
    except TypeError:
        covert += "?"
    # stop at the string "EOF"
    i += 8

```

of course, putting it all together will require some “massaging”...