Subtask – 6

Secure Packet Transfer application (automated) using Python

Aim:

To create an automated Client-Server program in python demonstrating Secure Packet Transfer, wherein the Client sends a “packet” of data (a string of text) to the Server, that prompts a pop-up box on the Server’s computer asking for approval. If the Server clicks “yes”, the message transfer would be successful and would be saved as a .txt file on the Server computer, if “No”, then the message would be dropped.

Source Code – Server.pyw:

```
import socket
import tkinter as tk
from tkinter import messagebox
import threading
import sys

# Create a socket for communication
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(("0.0.0.0", 12345))
server_socket.listen(1)
Obtained_Messages = []

# Create a tkinter window for server logs
root = tk.Tk()
root.title("Incoming Message Transfers Log")
text_area = tk.Text(root)
text_area.pack()

def update_text_area(text):
    text_area.insert(tk.END, text)
    text_area.see(tk.END)

# Define a function to handle incoming messages
def handle_message(client_socket):
    while True:
        data = client_socket.recv(1024).decode()
        if not data:
            break

        response = messagebox.askyesno("Incoming Message Detected:", 
f"Accept Message \"{data}\"?")
        if response:
            update_text_area(f"Granted Transfer: \"{data}\"\n")
            Obtained_Messages.append(data)
        else:
            update_text_area(f"Denied Transfer: \"{data}\"\n")
        print("Messages Granted: ", end="")
        print(Obtained_Messages)

        # Writing the Messages into a Textfile to ensure Packet
```

```
Received
        with open("Messages.txt", "w") as f:
            for Msg in Obtained_Messages:
                f.write(f"{Msg}\n")

# Accept incoming connections
print("Waiting for incoming connection...")

def accept_connections():
    while True:
        try:
            client_socket, client_address = server_socket.accept()
            print(f"Accepted connection from {client_address}")

            # Create a thread to handle the client
            client_thread = threading.Thread(target=handle_message,
args=(client_socket,))
            client_thread.start()
        except KeyboardInterrupt:
            print("Server interrupted. Closing server socket.")
            server_socket.close()
            sys.exit()

# Create a thread to accept connections
accept_thread = threading.Thread(target=accept_connections)
accept_thread.start()

# Start the tkinter main loop
try:
    root.mainloop()
except KeyboardInterrupt:
    print("Server interrupted. Exiting.")
    sys.exit()
```

Source Code – Client

```
import socket
import threading

# Create a socket for communication
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(("192.168.72.250", 12345))  # Replace with the
server's IP address

# Function to send messages to the server
def send_message():
    while True:
        message = input("Enter the message to be Sent to other
Computer: ")
        client_socket.send(message.encode())

# Create a thread to send messages
send_thread = threading.Thread(target=send_message)
send_thread.start()
```

Steps to Perform Automation:

i) Make sure you save your Server file as a No Console Python Executable File.

ii) Open Task Scheduler in your Windows Machine.

iii) Under the “Actions” drop down, click on “Create Task”.

iv) Give it your preferred name.

v) Under “Actions” in the new Dialog Box, click on “new” and add the path in which your Python executable file is found on your PC. This can be found by running the command “where python” in your command line.

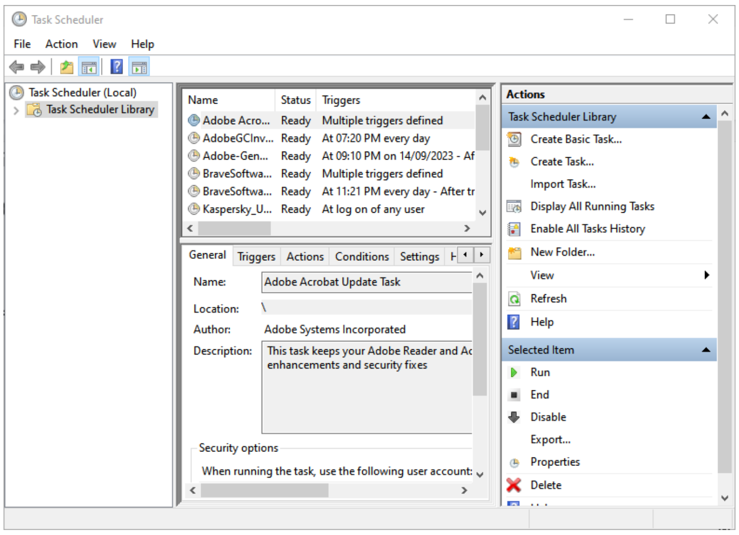
vi) Now in the Arguments (optional) field, give the name of your Server file with the extension.

vii) In the Start in (optional) field, enter the path of your Server File.

viii) In the “Triggers” category, select the time and frequency during which the code will automatically start to execute.

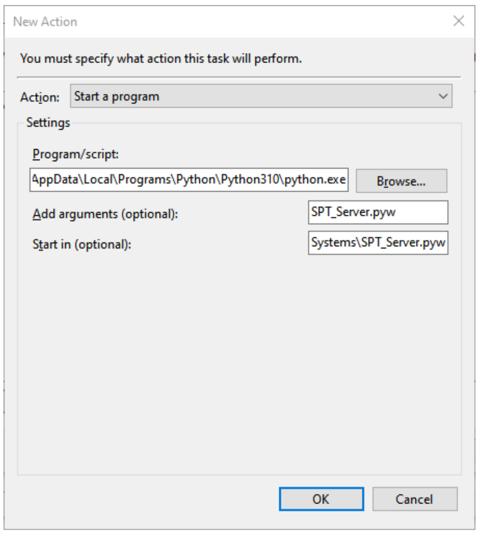
ix) Once all finalised, click ok. The automation has been successfully done and booting the system during the time and day mentioned will automatically execute the Server file in your computer.

Automation Phase – Screenshots:

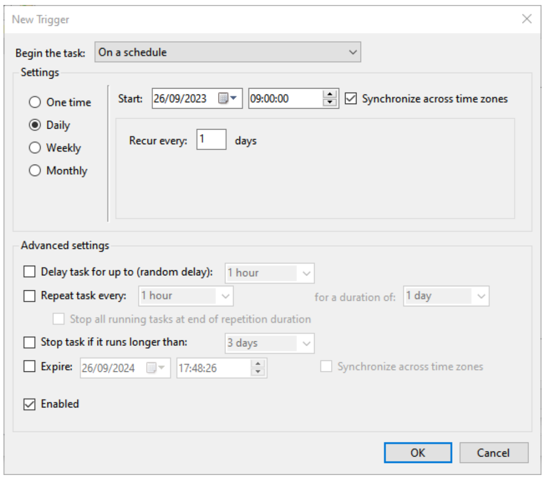


Name	Status	Triggers
Adobe Acro...	Ready	Multiple triggers defined
AdobeGClnv...	Ready	At 07:20 PM every day
Adobe-Gen...	Ready	At 09:10 PM on 14/09/2023 - Af
BraveSoftwa...	Ready	Multiple triggers defined
BraveSoftwa...	Ready	At 11:21 PM every day - After tr
Kaspersky_U...	Ready	At log on of any user

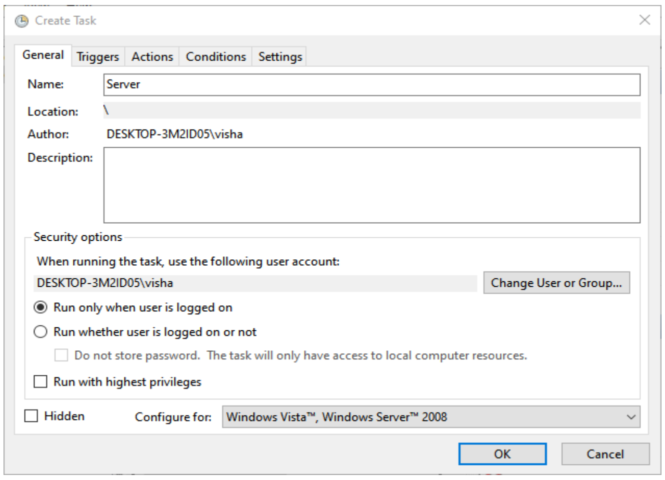
Name: Adobe Acrobat Update Task
Location: \
Author: Adobe Systems Incorporated
Description: This task keeps your Adobe Reader and Ac enhancements and security fixes
Security options
When running the task, use the following user account:



You must specify what action this task will perform.
Action: Start a program
Settings
Program/script: AppData\Local\Programs\Python\Python310\python.exe
Add arguments (optional): SPT_Server.pyw
Start in (optional): Systems\SPT_Server.pyw

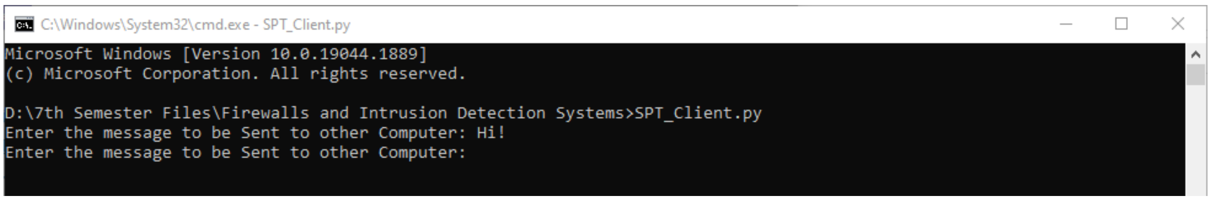


Begin the task: On a schedule
Settings
( ) One time
(●) Daily
( ) Weekly
( ) Monthly
Start: 26/09/2023 09:00:00 [✓] Synchronize across time zones
Recur every: 1 days
Advanced settings
[ ] Delay task for up to (random delay): 1 hour
[ ] Repeat task every: 1 hour for a duration of: 1 day
[ ] Stop all running tasks at end of repetition duration
[ ] Stop task if it runs longer than: 3 days
[ ] Expire: 26/09/2024 17:48:26 [ ] Synchronize across time zones
[✓] Enabled



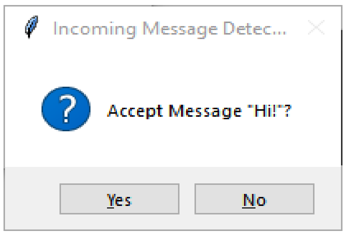
Name: Server
Location: \
Author: DESKTOP-3M2ID05\visha
Description:
Security options
When running the task, use the following user account:
DESKTOP-3M2ID05\visha
(●) Run only when user is logged on
( ) Run whether user is logged on or not
[ ] Do not store password. The task will only have access to local computer resources.
[ ] Run with highest privileges
[ ] Hidden Configure for: Windows Vista™, Windows Server™ 2008

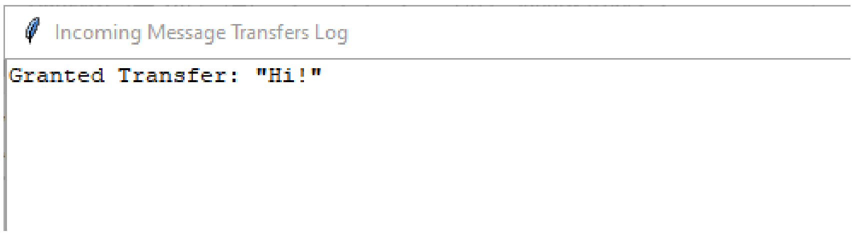
Output Screenshots:



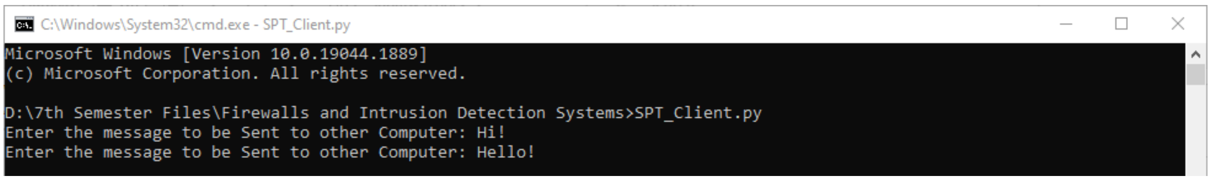
```
Microsoft Windows [Version 10.0.19044.1889]
(c) Microsoft Corporation. All rights reserved.

D:\7th Semester Files\Firewalls and Intrusion Detection Systems>SPT_Client.py
Enter the message to be Sent to other Computer: Hi!
Enter the message to be Sent to other Computer:
```



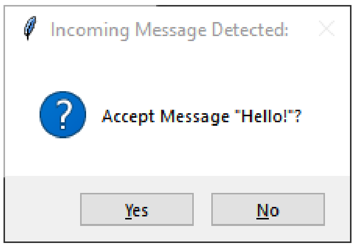


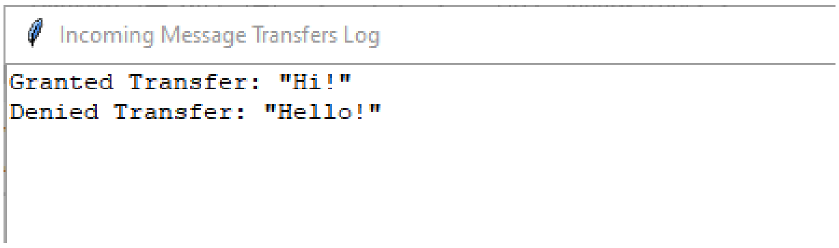
```
Granted Transfer: "Hi!"
```



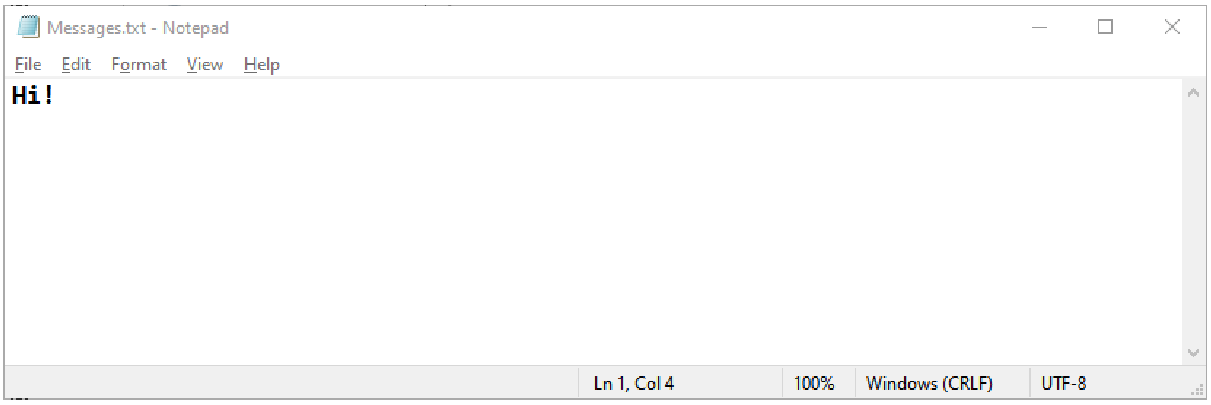
```
Microsoft Windows [Version 10.0.19044.1889]
(c) Microsoft Corporation. All rights reserved.

D:\7th Semester Files\Firewalls and Intrusion Detection Systems>SPT_Client.py
Enter the message to be Sent to other Computer: Hi!
Enter the message to be Sent to other Computer: Hello!
```





```
Granted Transfer: "Hi!"
Denied Transfer: "Hello!"
```



```
Hi!
```

In the above Messages.txt file, only "Hi!" is found, meaning that the packet transfer is approved if "yes" is pressed and is denied when "no" is pressed.

Result:

Thus, we successfully implemented a Packet Transfer Program in Python and accepted/received messages sent from the Client accordingly through a dialog box in the Server Computer and also automated the Server code through the Task Scheduler feature in the Windows Operating System.