Atahan İyiekici
21801985
CS421-001
PA1

Part 1)

Ncat results:

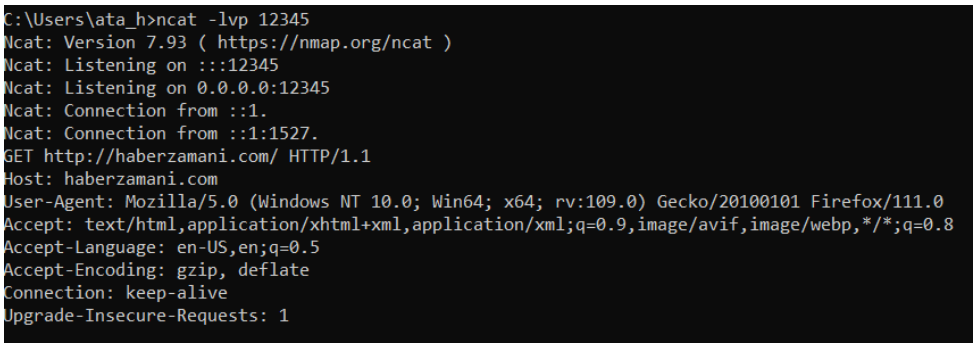
1. 

Figure 1: Ncat result of http://haberzamani.com/ with port 12345

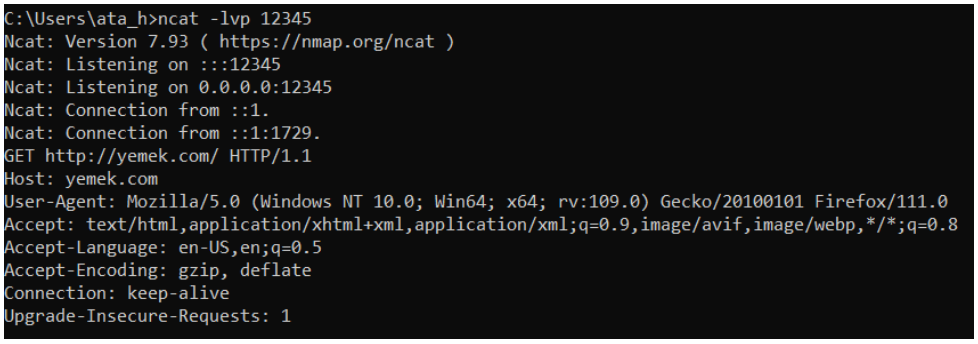
2. 

Figure 2: Ncat result of http://yemek.com/ with port 12345

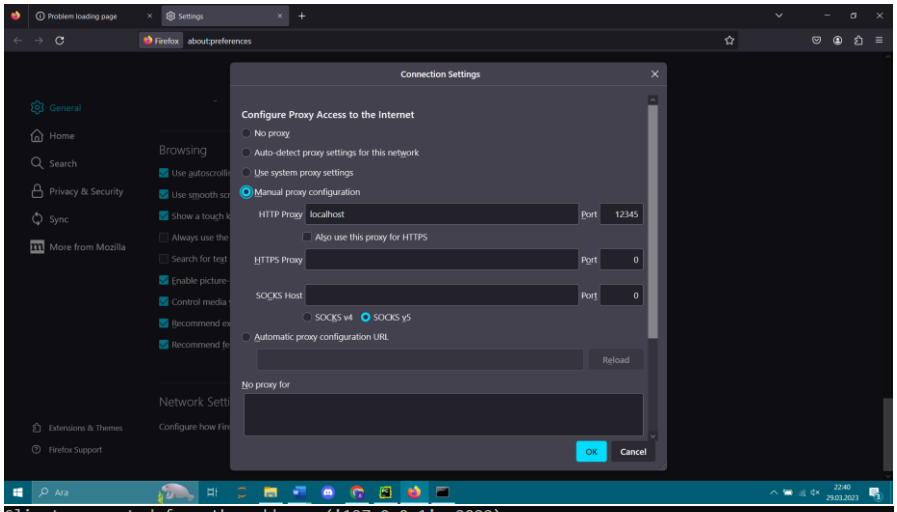
3. 

Figure 3: Ncat result of http://sinefy.com with port 12345

Atahan İyiekici
21801985
CS421-001
PA1

Part 2)

ProxyServer.py Test Results:

1. 

2.
```
Client connected from the address ('127.0.0.1', 2022)
Retrieved request from Firefox:

GET http://www.cs.bilkent.edu.tr/~cs421/spring22/project1/bilkent.txt HTTP/1.1
Host: www.cs.bilkent.edu.tr
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1


Round Trip Time: 2.99 ms
Status code: 200
Downloading file `bilkent.txt`...
Saving file bilkent.txt...
(Continue with next website)
```

3.
```
Client connected from the address ('127.0.0.1', 2063)
Retrieved request from Firefox:

GET http://www.cs.bilkent.edu.tr/~cs421/spring22/project1/decrypted_file_1.txt HTTP/1.1
Host: www.cs.bilkent.edu.tr
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1


Round Trip Time: 2.99 ms
Status code: 200
Downloading file `decrypted_file_1.txt`...
Saving file decrypted_file_1.txt...
(Continue with next website)
```

4.
```
Client connected from the address ('127.0.0.1', 2085)
Retrieved request from Firefox:

GET http://www.cs.bilkent.edu.tr/~cs421/spring22/project1/nums.txt HTTP/1.1
Host: www.cs.bilkent.edu.tr
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1


Round Trip Time: 2.0 ms
Status code: 404
################################################
Error: File not found or another error occurred.
################################################
```

5.
```
Client connected from the address ('127.0.0.1', 2121)
Retrieved request from Firefox:

GET http://www.cs.bilkent.edu.tr/~cs421/spring22/project1/lorem.txt HTTP/1.1
Host: www.cs.bilkent.edu.tr
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/111.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1


Round Trip Time: 4.0 ms
Status code: 200
Downloading file `lorem.txt`...
Saving file lorem.txt...
(Continue with next website)
```

Source Code Explanation:

```python
# Function to get the request from the client
def get_request(socket_):
    request = socket_.recv(BUFFER_SIZE).decode()
    return request
```

Figure 4: Request getter method

The function in the figure 4 gets the request using socket library function .recv() with parameter BUFFER_SIZE and decodes it with .decode().

```python
def get_parsed_url(request):
    url = request.split(' ')[1]
    # We are dealing with http only
    if url.startswith('http://'):
        url = url[7:]
    server_name = url.split('/')[0]
    file_name = os.path.basename(url)
    parsed_url = '/' + url[len(server_name):]
    return server_name, file_name, parsed_url
```

Figure 5: Function to parse the URL from the request

The function above in the figure 5 takes request as parameter and it splits it and gets first index, then as we are dealing with http we take array after 7. index then takes its parts where server name file name and parsed url is in.

```python
def send_http_request(url, server_name, socket_):
    http_get_request = f"GET {url} HTTP/1.0\r\nHost: {server_name}\r\n\r\n"
    socket_.sendall(http_get_request.encode())
```

Figure 6: Function to send an HTTP request to remote server

The function above takes url, server name and socket as parameters and it creates a http get request then sends it with help of socket library.

```python
def get_http_response(socket_):
    response = socket_.recv(BUFFER_SIZE)
    status_code = response.split()[1].decode('utf-8', 'ignore')
    print(f"Status code: {status_code} OK") if (status_code == 200) else print(f"Status code: {status_code}")
    return status_code
```

Figure 7: Function to get the HTTP response from the remote server and print the status code

The function above takes socket as parameter and receives the response with the first line, then it splits response and takes status code with decode function. This function also have

parameters as utf-8 and ignore which makes function to ignore any invalid bytes which are not uft-8 in this occasion. Also, it displays if the status code is 200

```
def status_code_ops(status_code, file_name, socket_):

    if status_code == "200":
        with open(file_name, "wb") as file:
            print(f"Downloading file `{file_name}`...")
            while True:
                incoming_data = socket_.recv(BUFFER_SIZE)
                if not incoming_data:
                    break
                file.write(incoming_data)
        print(f"Saving file {file_name}...")
        print("(Continue with next website)")
    else:
        print("################################################")
        print("Error: File not found or another error occurred.")
        print("################################################")
```

Figure 8: Function to handle different operations for different status codes

The function in the figure 8, only works if status code is 200, if it is 200, then it writes the data into a file with its file name

```
def main():

    arguments = sys.argv

    if len(arguments) == 2:
        port_ = int(sys.argv[1])
        p_downloader = ProxyDownloader(port_)
        p_downloader.start()
    else:
        print("Usage: python3 ProxyDownloader.py <port>")
        sys.exit(1)


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

Figure 9: Main function that takes arguments

```
class ProxyDownloader:
    def __init__(self, port, host_name='localhost'):
        self.port = port
        self.host_name = host_name
        self.server = None

    # Main function to start the proxy downloader
    def start(self):
```

Figure 10: ProxyDownloader class init and start function (host name “localhost” default)

```
def start(self):

    self.server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    self.server.bind((self.host_name, self.port))
    self.server.listen(8)
    print(f"Proxy server listening on the port {self.port}")
    while True:
        client_socket, client_address = self.server.accept()
        request = get_request(client_socket)
        server_name, file_name, parsed_url = get_parsed_url(request)
        if server_name == bilkent:
            print(f"Client connected from the address {client_address}")
            print("Retrieved request from Firefox: \n")
            print(request)
            try:
                remote_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                start_time = time.time()
                remote_socket.connect((server_name, 80))
                end_time = time.time()
                # Calculating and displaying round trip time
                rtt = round((end_time - start_time) * 1000, 2)
                print(f"Round Trip Time: {rtt} ms")
                send_http_request(parsed_url, server_name, remote_socket)
                status_code = get_http_response(remote_socket)
                status_code_ops(status_code, file_name, remote_socket)
            # Throw an exception if there is
            except Exception as e:
                print(f"Error: {e}")
            finally:
                client_socket.close()
```

Figure 11: Main function to start the proxy downloader

Start function is the main function of this code, it firstly takes socket.socket, and binds the server and listens server with a certain number of “maximum number of queued connections”. Then, using accept() function we can get client_socket and client address, from client socket all data can be extracted. Also get_request method returns the request. Using get_parsed_url() method, server name, file name and parsed url are returned. Code is then filtered with servername==bilkent statement in which bilkent string is the server name of bilkent. After filtering operation, and printing requests and some informations, code tries to connect remote socket, meanwhile the code measures time between trip time and displays round trip time as a little bonus. After the connection, with send_http_request method it sends http response and from get_http_response it receives status code. With this status code, it executes certain operations with status_code_ops method. Also code throws exceptions if there is an error. Finally, it closes the socket connection.