Lab 1: Investigating HTTP with Ncat, Wireshark, & Python

Objective:

Manually 'speak' the HTTP protocol to a web server using ncat, capture the traffic to analyze the TCP/HTTP structure in Wireshark, and then replicate the client functionality using Python sockets.

Part 0: Pre-Lab Setup (Install Ncat)

Before starting, you must have ncat installed. It is a powerful command-line tool that reads and writes data across networks.

- **Windows Users:**
  - Install **Nmap**, which includes ncat.
  - Download the "Nmap executable installer" from nmap.org/download.html.
  - During installation, ensure the box for **Ncat** is checked.
  - *Verification:* Open PowerShell or CMD and type ncat --version.
- **macOS Users:**
  - You likely have nc installed by default. Open a terminal and try nc -h.
  - If you need the specific Nmap version (preferred), use Homebrew: brew install nmap.
- **Linux Users:**
  - Install via your package manager:
    - Ubuntu/Debian: sudo apt install ncat (or nmap)
    - Fedora: sudo dnf install nmap-ncat

Part 1: Wireshark Preparation

We need to set up Wireshark to filter out background noise so we only see our specific traffic.

1. In **Wireshark**, select your active network interface (Wi-Fi or Ethernet).
2. In the capture filter bar, enter the following to isolate web traffic:
   tcp.port == 80
3. Click the blue shark fin icon to **Start Capture**.

Part 2: The Manual HTTP Request

You will now act as the web browser. Instead of clicking a link, you will type the raw protocol commands.

1. Open your terminal (CMD, PowerShell, or Terminal).

2. Connect to the server gaia.cs.umass.edu on port 80.
   - **Windows (Nmap ncat):** ncat -C gaia.cs.umass.edu 80
   - **macOS/Linux:** nc -C gaia.cs.umass.edu 80
     (Note: The -C flag ensures we send the correct CRLF line endings required by HTTP).
3. Once the cursor sits waiting (you won't see a "connected" message), type the following request exactly. **Press Enter TWICE** after the second line.

```
HTTP
GET /kurose_ross/interactive/index.php HTTP/1.1
Host: gaia.cs.umass.edu
```

4. **Observe:** The server should blast back text starting with a response, maybe HTTP/1.1 200 OK, followed by headers and HTML code.

Question #1: What was the HTTP response status code? What is it's meaning? What should you do now?

5. Now try it again with this new options (SSL activated)

   ncat --ssl -C gaia.cs.umass.edu 443

Question #2: What was the response status code now? What is it's meaning?

Part 3: The Dissection

Let's see what that conversation looked like on the wire.

1. **Stop** the Wireshark capture.
2. Locate the packets generated by your ncat command.
3. **Analyze the TCP Handshake:**
   - Find the first three packets (SYN, SYN-ACK, ACK).
   - ***Deliverable***: Screenshot this sequence. Capture the local computer time and date on the screenshots.
4. **Analyze the HTTP Request:**
   - Find the packet labeled **GET**.
   - In the middle pane, expand **Hypertext Transfer Protocol**.

Question #3: What "User-Agent" did you send? (Since you used ncat, it might be missing or different from a browser like Chrome).

5. **Analyze the HTTP Response:**
   - Find the packet labeled **HTTP/1.1 200 OK**.

Question #4: What server software is running on the remote machine (look at the Server: header)?

Part 4: Python Socket Client

Now that you've done it manually, write a script to do it automatically.

Task: Create a file named http_client.py using the starter code below. Adjust the code to the appropriate settings and fill in the blanks to replicate your ncat session.

Python Code

```
from socket import *

server_name = 'gaia.cs.umass.edu'
server_port = 80

# 1. Create a TCP socket (IPv4)
# Hint: Use AF_INET for IPv4 and SOCK_STREAM for TCP
client_socket = socket(AF_INET, SOCK_STREAM)

# 2. Connect to the server
client_socket.connect((server_name, server_port))

# 3. Prepare the HTTP request
# Critical: HTTP requires carriage return & new line (\r\n) at the end of headers
# The double \r\n\r\n indicates the end of the header section.
request = "GET /kurose_ross/interactive/index.php HTTP/1.1\r\nHost: gaia.cs.umass.edu\r\n\r\n"

# 4. Send the request
# We must encode the string into bytes before sending
client_socket.send(request.encode())

# 5. Receive the response
# We read up to 4096 bytes
response = client_socket.recv(4096)

# 6. Decode and print the result
print(response.decode())
```

```
# 7. Close the connection
client_socket.close()
```

Question #5: Do you get a response status message 200? If not, what changes are needed to make it respond with a 200?

Submission Requirements

One document (Word or PDF) with:

1. **Screenshots with local time/date:**
   - Your terminal showing the manual ncat output.
   - Wireshark showing the "GET" packet details.
2. **Code:**
   - Your working http_client.py file on a public repo in your GitHub account. I only need the **link to the repo/file**. Your code should have your student's name and ID at the top, and relevant comments on the lines that deal with the network protocol.
3. **Answers:**
   - Answer the five questions from Parts 2, 3, and 4.