

1. Understanding HTTP

<https://api.github.com/repos/amehlhase316/memoranda/commits>

The image shows a Wireshark packet capture of an HTTP GET request to the GitHub API. The packet list shows a TCP segment from 12934 to 12935. The packet details pane shows the Ethernet II, Internet Protocol Version 6, and Transmission Control Protocol (Source Port: 515) layers. The packet bytes pane shows the raw data. To the right, a browser window displays the JSON response from the GitHub API, which is a list of commit objects. The first commit object is visible, showing details like the commit hash, author, and files.

https://api.github.com/repos/amehlhase316/memoranda/commits/autocomplete?q=canis&per_page=100

1. API calls I used modified the given URL according to the statement asked in questions. The initial call specifying the repository name and its owner, followed by appending "/commits" to retrieve all the commits from the repository's default branch. The next api call I added autocomplete. Next I added "?per_page=100" which makes the call display 100 up to commits per page.
2. Stateless Protocol is a network protocol in which Client sends a request to the server and server response back as per the given state. In Stateful Protocol, if a client sends a request to the server then it expects some kind of response, in case of no response then it resends the request.

2. Setup your second system and run Server on it

2.2

The image shows a terminal window with the following content:

```
ec2-user@ip-172-31-91-62:~/ser32examples/Socket...$  
You can make the following GET requests  
• /file/sample.html -- returns the content of the file sample.html  
• /json -- returns a json of the /random request  
• /random -- returns index.html  
File Structure in www (you can use /file/www/FILENAME):  
• index.html  
• root.html  
[ec2-user@ip-172-31-91-62 Sockets]$ cd WebServer  
[ec2-user@ip-172-31-91-62 WebServer]$ gradle FunWebServer  
Task :FunWebServer  
Received: GET / HTTP/1.1  
Received: Host: 3.89.88.126:9000  
Received: Connection: keep-alive  
Received: Upgrade-Insecure-Requests: 1  
Received: User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36  
Received: Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7  
Received: Accept-Encoding: gzip, deflate  
Received: Accept-Language: en-US,en;q=0.9  
Received: FINISHED PARSING HEADER  
----- 75% EXECUTING [3m 35s]  
ec2-user@ip-172-31-91-62:~/ser32examples/Socket...$
```

2.3

The image shows a Wireshark packet capture on interface wlo1. The filter is set to 'tcp.port==9000'. The packet list shows a GET request from 10.0.0.151 to 3.89.88.126 on port 9000. The packet details pane shows the HTTP request structure, and the packet bytes pane shows the raw data. The status bar at the bottom indicates 163746 packets captured and 20 displayed.

1. tcp.port==9000, because it filters all the traffic going through port 9000.
2. On "<http://3.89.88.126:9000/random>" I can first see bread picture and then after clicking on Random street pictures come.

Random

bread



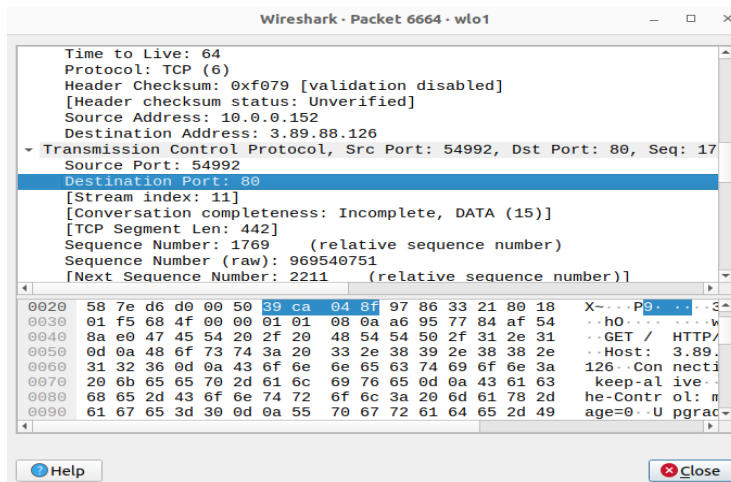
3.
 - a. 200 OK
 - b. 404 Not Found
 - c. 400 Bad Request
4.
 - a. 200 OK - Everything is fine.

- b. 404 Not Found - File does not exist.
- c. 400 Bad Request - Request not recognized.

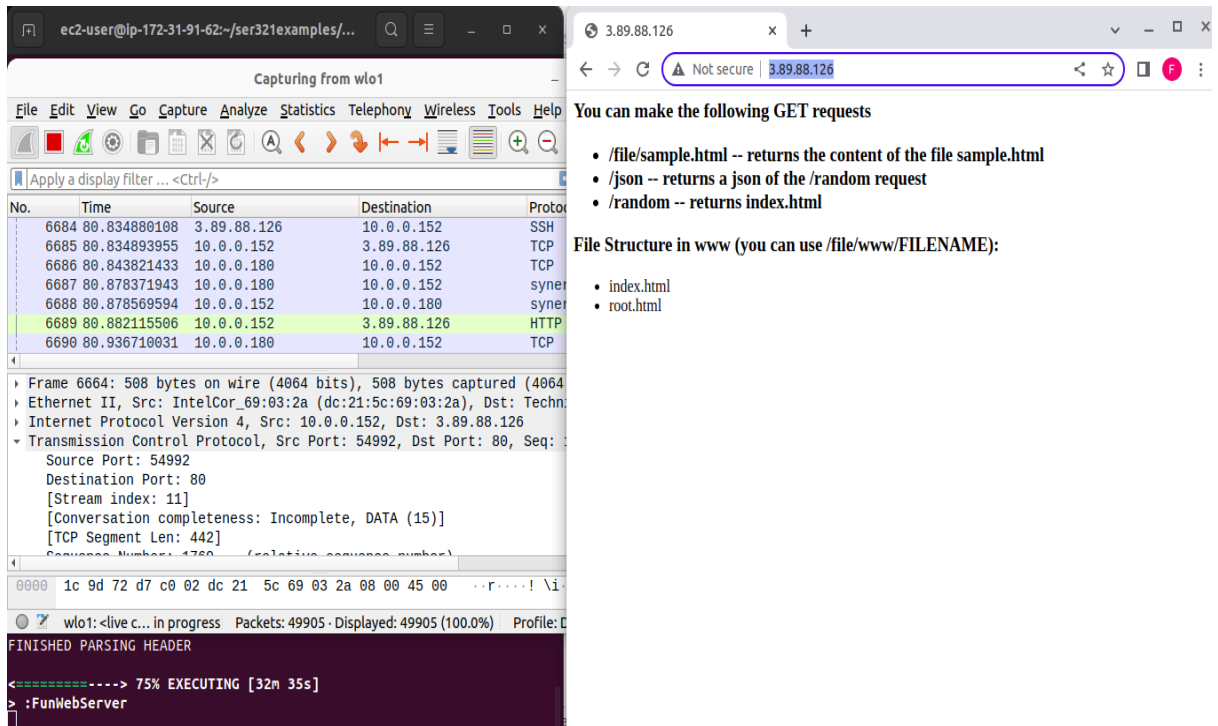
- 5. Yes, HTTP is not encrypted.
- 6. HTTPS is HTTP with encryption. HTTPS uses TLS (SSL) to encrypt normal HTTP requests and responses.
- 7. tcp.port == 80; Src port:9000;
- 8. Src port:9000, SMTP is for email service.

2.4 Setting up a “real” Web server

- 1. <http://3.89.88.126/>
- 2.



- 3. It is still HTTP, because there is no SSL certification.
- 4.



2.6.1

The multiple case did not have error handling. I check if the request contains “num1=” and “num2=” then check if the parameters are integers. Code 400 as Bad Request:

```
else if (request.contains("multiply?")) {
    // This multiplies two numbers, there is NO error handling, so when
    // wrong data is given this just crashes

    Map<String, String> query_pairs = new LinkedHashMap<String, String>();
    // extract path parameters
    query_pairs = splitQuery(request.replace("multiply?", ""));

    // TODO: Include error handling here with a correct error code and
    // a response that makes sense

    if (!query_pairs.containsKey("num1") || !query_pairs.containsKey("num2")) {
        builder.append("HTTP/1.1 400 Bad Request\n");
        builder.append("Content-Type: text/html; charset=utf-8\n");
        builder.append("\n");
        builder.append("Missing parameters: num1 and/or num2");
    } else {
        try {
            // extract required fields from parameters
            Integer num1 = Integer.parseInt(query_pairs.get("num1"));
            Integer num2 = Integer.parseInt(query_pairs.get("num2"));

            // do math
            Integer result = num1 * num2;

            // Generate response
            builder.append("HTTP/1.1 200 OK\n");
            builder.append("Content-Type: text/html; charset=utf-8\n");
            builder.append("\n");
            builder.append("Result is: " + result);
        } catch (NumberFormatException e) {
            // Handle invalid number format
            builder.append("HTTP/1.1 400 Bad Request\n");
            builder.append("Content-Type: text/html; charset=utf-8\n");
            builder.append("\n");
            builder.append("Invalid number format: num1 and num2 must be integers");
        }
    }
}
```