Name Ni, Tianshu Date: 2023.9.28

(last name, first name)

NYU ID: N16371283

Course Section: 001

**Lab #2 – Application Layer**

**Total in points** (100 points total):

**Professor’s Comments:**

**Affirmation of my Independent Effort:** Tianshu Ni

**(Sign here)**

1. Problem 1 – Protocols Analysis:

1. Use a packet sniffer (e.g., Wireshark) to capture the sequence of ASCII characters that are sent and received by a web browser as a result of a request of your choice to a web server. Provide a screen capture of these sequences and add carriage return and line feed characters as needed to improve readability.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Identify the complete URL of the document requested, the HTTP protocol version for both the request and response, the operating system that the web browser is running on, and the kind of web server that answered the request.

* The complete URL of the document requested: <http://ocsp.pki.goog/gts1c3/MFAwTjBMMEowSDAHBgUrDgMCGgQUxy55it3%2FYTSzuu1HQri7xsAkB2MEFIp0f6%2BFze6VzT2c0OJGFPNxNR0nAhEAoWc75TesavwSx8I4oV0sOA%3D%3D>
* The HTTP protocol version for both the request and response: HTTP/1.1
* The operating system that the web browser is running on: User-Agent: com.apple.trustd/3.0 -> Apple OS
* The kind of web server that answered the request: ocsp\_responder

1. Is the HTTP connection persistent?

Yes, since the sequence says “Connection: keep-alive”.

1. Which web browser sent the request and why is it important for the server to know this information?

We can know it from “User-Agent: com.apple.trustd/3.0”.

Knowing the **User-Agent** information is crucial for servers as it allows them to optimize content delivery based on the client's software or device. This ensures compatibility and a tailored user experience. Additionally, it aids in analytics, troubleshooting, and security, as servers can identify usage patterns, resolve reported issues more effectively, and detect potential malicious activity or anomalies based on unusual client identifiers.

1. Was the request successful and, if so, what type of document was received by the server?

Yes, and the document type is OCSP response. We know it from “Status Code: 200, [Status Code Description: OK], Response Phrase: OK, Content-Type: application/ocsp-response”.

1. Experiment with a different application layer protocol of your choice and use Wireshark to analyze the corresponding requests.

A screenshot of a computer

Description automatically generated

2. Problem 2 – Network Application Deployment:

Deploy a simple application to K8s that returns the current time when visited using the path “/time”.

Prerequisites:

i. Install Docker from https://www.docker.com/

ii. Create an account in Docker Hub: https://hub.docker.com/

(a) Clone the sample Python Flask application provided at the following link:

https://github.com/metacomp/nyu-cs2262-001-fa20/tree/master/sample\_time\_app

i. Refer to the sample Dockerfile provided at the above link.

ii. To build an image: “docker build -t USERNAME/sample-timeapp:latest” (USERNAME is your dockerhub username)

iii. To run: “docker run --name sample-time-app -p 8080:8080 -it sample- time-app”

iv. Make sure that your app is up and running.

v. Make sure that your app returns the current time when visited using the “/time” path; make changes to the application accordingly.

(b) Once changes are done, push the image to Docker hub

i. First login with: “docker login” (remember the username and password created earlier)

ii. Push the image: docker push USERNAME/sample-time-app:latest

iii. Check in dockerhub profile to see if you can see the image

(c) Now deploy your image to the IBM Cloud K8s instance.

i. Create a deployment: “kubectl create deployment sample-time-app

-- image=docker.io/USERNAME/sample-time-app:latest”

ii. Expose your app’s port: “kubectl expose deployment/sample-timeapp -- type="NodePort" --port 8080”

iii. Check the exposed port: kubectl get services

iv. Check the IP of your worked node from IBM Cloud

v. Access the http://IP:NodePort and confirm your app works

(d) Commit your code to Github (also make sure to submit the NYU Brightspace). Following should be included in the submission.

i. Dockerfile

ii. Any other file that your Dockerfile uses in order to successfully build the image.

My work is uploaded on github, you may git clone https://github.com/cedricni/Workspace.git