CS 145 Lab Exercise 13
Wireshark Lab: A RESTful Application
A.Y. 2018-2019, 2nd Semester

1 Introduction

As discussed in class, the HyperText Transfer Protocol (HTTP) defines how web clients request web pages from web servers and how servers transfer web pages to clients. Also discussed in class are the HTTP methods, which are specified to control the interaction between the client and the server. HTTP methods are also known as "HTTP verbs," since they denote an action that a client wants to do a certain URL. In this laboratory exercise, you will see that the application of HTTP is not confined in web pages. Specifically, HTTP can be used to manage resources that are hosted on remote servers through the use of HTTP verbs.

In this exercise, you are going to analyze the packets used by a provided RESTful application. You are also going to extend the RESTful application to add more resources and analyze packets on your extended application. Students are expected to be familiar with Python3 as they will use a Python3 virtual environment to extend the application and analyze packets. Familiarity with the Python3 packages `Flask`[1], `Flask-RESTful`[2], and `tinydb`[3] is required, but it can be learned as you work on the lab exercise.

Note that this laboratory exercise will be done by *pairs*. Only one student per pair needs to submit the laboratory report, so please make sure that both member names are written in the laboratory report.

2 Materials

For this laboratory exercise you would need the following:

1. Two computers. Each computer must have the following programs installed:
    - Python3 3.6 or higher with `venv` module.
    - pip 9.0.1 or higher.

[1] `http://flask.pocoo.org/`

[2] `https://flask-restful.readthedocs.io/en/latest/`

[3] `https://tinydb.readthedocs.io/en/latest/`

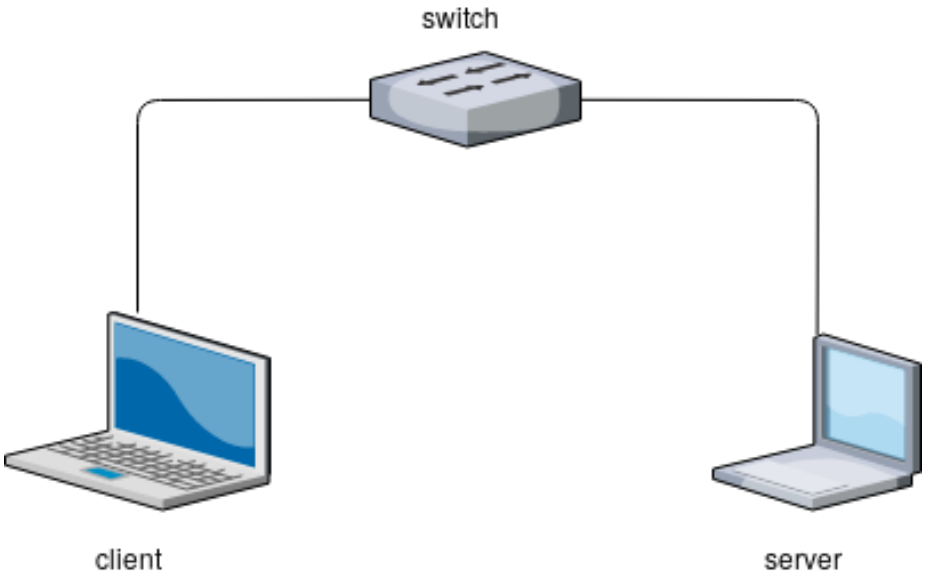

Figure 1: Target Setup for Lab Exercise 13

- Wireshark
- `cURL` 7.58 or higher.

The computers must also have an ethernet port.

2. A network switch that is not connected to a DHCP server. If you are working in the teaching laboratories, make sure that the switch is *not* connected to the UP network.
3. A Cat 5 (LAN) cable, for connecting the computers to the switch. See Figure 1 for the target setup for the lab exercise.

3 Restrictions

For this laboratory exercise the following restrictions apply:

- This exercise has two components: a client component and a server component. You need to use two machines for the client and the server.
- Since the lab computers are not properly configured to run the components of this lab, we are allowing you to use any two machines for this exercise, as long as the following programs and interfaces mentioned in Section 2 are installed. See Figure 1 for the target setup.
- Trace file generation must also be done in the machines that you will choose, using the Ethernet (not WiFi) connection.
- Trace analysis may be done in any machine with the Wireshark software installed.

4 Instructions: Trace Generation, Application Specification

For this laboratory exercise you would neet to generate *three pairs* of traces.

4.1 Part 1: Provided RESTful application

1. Designate one of the two computers as the *client* and the other the *server*.
2. From the UVLE website, download `lab13server.zip` onto the *server* computer. Unzip the contents into a folder named `lab13server` and go to that folder.
3. Use `ifconfig` to determine the IP address of the *server* computer.
4. Start up the Wireshark software on both computers.
5. You will run your server application on a Python3 virtual environment[4]. To set up the environment, go to the `lab13server` folder and enter the following command:

   ```
   python3 -m venv labenv
   ```

6. Your virtual environment is named `labenv`. Enter the following to use the virtual environment:

   ```
   # For Linux/OSX
   source labenv/bin/activate


   # For Windows
   labenv\Scripts\activate.bat
   ```

7. You will use three external Python3 packages for the server application, namely `Flask`, `Flask-RESTful`, and `tinydb`. To install the following packages to your virtual environment, run the following command.

   ```
   pip3 install -r requirements.txt
   ```

8. On the *server* computer, start the RESTful server by entering the following command:

   ```
   python3 app.py
   ```

   This should run a server application that uses port 80. To avoid running into errors, please make sure that the *server* computer is not running any process that uses port 80 before you enter the command. Also, you might need elevated access (e.g., `sudo`) to allow `python3` to use port 80.

9. Once the *server* is ready, select the appropriate interface and begin packet capture on *both* computers.

[4] `https://docs.Python3.org/3/tutorial/venv.html`

10. From the *client* computer, you need to run the following commands in sequential order. For each `curl` command (there are 10), change "10.40.71.118" to the IP address of the server computer and execute the command.

```
curl http://10.40.71.118/zodiac/chinese -X GET -v
curl http://10.40.71.118/zodiac/chinese/horse -X GET -v
curl http://10.40.71.118/zodiac/chinese/ram -X GET -v
curl http://10.40.71.118/zodiac/chinese/horse -X DELETE -v
curl http://10.40.71.118/zodiac/chinese/horse -X GET -v
curl http://10.40.71.118/zodiac/chinese -X POST -v \
-H "Accept: application/json" \
-H "Content-Type: application/json" \
-d '{ "animal": "horse",
    "characteristics": "Energetic, Passionate, Upright, Aspirant.",
    "position": 6 }'
curl http://10.40.71.118/zodiac/chinese/horse -X GET -v
curl http://10.40.71.118/zodiac/chinese/sheep -X PUT -v \
-H "Accept: application/json" \
-H "Content-Type: application/json" \
-d '{ "animal": "ram" }'
curl http://10.40.71.118/zodiac/chinese/ram -X GET -v
curl http://10.40.71.118/zodiac/chinese/sheep -X GET -v
```

11. After executing all the commands in the client, stop packet capture on both computers.

12. On the client side, save the packet trace file as `labexercise13_A_client.pcapng`. On the server side, save the packet trace file as `labexercise13_A_server.pcapng`.

13. Exit the Wireshark software on both computers.

14. On your *server* computer, terminate the Python3 application by entering `Ctrl+C` (or ⌘+C on OSX).

15. You may choose to exit from the virtual environment by typing the following command:

    ```
    deactivate
    ```

    However, parts 2 and 3 of this exercise will require you to use the same environment. It is suggested that you only exit from the virtual environment once you are done the entire lab exercise.

4.2 Part 2: Extended RESTful application I (Western Zodiac)

1. Using the source code from the previous part as the starting point, modify the code so that the following resources in Tables 1 and 2 will also be available to the client.

Resource: `/zodiac/western/<symbol-name>`

Method	Description
GET	Get the zodiac symbol that matches with the <symbol-name>
PUT	Modify the zodiac symbol that matches with the <symbol-name>
DELETE	Delete the zodiac symbol that matches the <symbol-name>

Table 1: New RESTful resource: Western Zodiac Symbols

Resource: `/zodiac/western`

Method	Description
GET	Get all western zodiac symbols
POST	Add a new western zodiac symbol

Table 2: New RESTful resource: Western Zodiac List

2. Start up the Wireshark software on both computers.

3. On the *server* computer, run the RESTful application similar to how it is run in the previous part.

4. On *both* computers, select the appropriate interface and begin packet capture.

5. On the *client* computer, run the following commands in sequential order. For each `curl` command (there are 10), change "10.40.71.118" to the IP address of the server computer and execute the command.

```
curl http://10.40.71.118/zodiac/western -X GET -v
curl http://10.40.71.118/zodiac/western/Libra -X GET -v
curl http://10.40.71.118/zodiac/western/Ram -X GET -v
curl http://10.40.71.118/zodiac/western/Libra -X DELETE -v
curl http://10.40.71.118/zodiac/western/Libra -X GET -v
curl http://10.40.71.118/zodiac/western -X POST -v \
-H "Accept: application/json" \
-H "Content-Type: application/json" \
-d '{ "symbol": "Libra",
    "start": {
        "month": "September",
        "day": 23
    },
    "end": {
        "month": "October",
        "day": 22
    },
    "reading": "If you have doubts, you need to have confidence in yourself." }'
curl http://10.40.71.118/zodiac/western/Libra -X GET -v
curl http://10.40.71.118/zodiac/western/Aries -X PUT -v \
-H "Accept: application/json" \
-H "Content-Type: application/json" \
-d '{ "symbol": "Ram" }'
```

```
curl http://10.40.71.118/zodiac/western/Ram -X GET -v
curl http://10.40.71.118/zodiac/western/Aries -X GET -v
```

6. After executing all the commands in the client, stop packet capture on both computers.
7. On the client side, save the packet trace file as `labexercise13_B_client.pcapng`. On the server side, save the packet trace file as `labexercise13_B_server.pcapng`.
8. Exit the Wireshark software on both computers.
9. On your *server* computer, terminate the Python3 application by entering `Ctrl+C` (or ⌘+C on OSX).

4.3 Part 3: Extended RESTful application II (Zodiac Reader)

1. Using the source code from the previous part as the starting point, modify the code so that the following resource in Table 3 will also be available to the client.

Resource: `/zodiac/reading/<year>/<month>/<day>`

Method	Description
GET	Get the zodiac reading given the <year>, <month>, and <day> Both Chinese and Western readings must be returned.

Table 3: New RESTful resource: Zodiac Reading

2. Start up the Wireshark software on both computers.
3. On the *server* computer, run the RESTful application similar to how it is run in the previous part.
4. On *both* computers, select the appropriate interface and begin packet capture.
5. On the *client* computer, run the following commands in sequential order. For each `curl` command (there are 2), change "10.40.71.118" to the IP address of the server computer and execute the command.

```
curl http://10.40.71.118/zodiac/reading/1941/December/7 -X GET -v
curl http://10.40.71.118/zodiac/reading/2007/October/11 -X GET -v
```

6. After executing all the commands in the client, stop packet capture on both computers.
7. On the client side, save the packet trace file as `labexercise13_C_client.pcapng`. On the server side, save the packet trace file as `labexercise13_C_server.pcapng`.
8. Exit the Wireshark software on both computers.
9. On your *server* computer, terminate the Python3 application by entering `Ctrl+C` (or ⌘+C on OSX).
10. At this point, you may now exit from the virtual environment.

After generating the six traces, you are now ready to work on the laboratory report.

5 What to hand in

For this laboratory exercise you would need to hand in three files:

1. A *zip file* named `lastname_firstname_lab13.zip` (where `lastname` is your last name and `firstname` is your first name) containing

   (a) The updated version of `app.py`

   (b) The updated version of `database.py`

   There will be a separate "assignment" page for uploading the zip file. Note that while we will not mark/grade the codes or traces themselves, they are *prerequisites* for us to consider your answers in the laboratory report. For example, without a *functional* `database.py` and `app.py`, we will *not* consider your answers for Parts 2 and 3.

2. The usual laboratory report.

3. A *zip file* named `lastname_firstname_lab13_trace_files.zip` (where `lastname` is your last name and `firstname` is your first name) containing the Wireshark trace files.

   (a) `labexercise13_A_client.pcapng`

   (b) `labexercise13_A_server.pcapng`

   (c) `labexercise13_B_client.pcapng`

   (d) `labexercise13_B_server.pcapng`

   (e) `labexercise13_C_client.pcapng`

   (f) `labexercise13_C_server.pcapng`

For the laboratory report, answer the following questions based on your Wireshark experimentation:

5.1 Part 1: Provided RESTful application

Answer the following based on `labexercise13_A_client.pcapng` or `ifconfig` results.

1. Filter out (remove) all non-HTTP-related packets by applying the "http" filter. What is the IP address of the client computer? What is the IP address of the server computer? Include annotated screenshots or images supporting your answer.

2. How many `curl` commands are executed? How many HTTP requests are sent to the server? Does every `curl` command correspond to an HTTP request? Include annotated screenshots or images supporting your answer.

3. What does the `curl` command do? What is the main parameter of `curl`? Cite references supporting your answer, or show proof.

4. Each `curl` command has the `-v` option. What does the `-v` option do? Cite references supporting your answer, or show proof.

5. Each `curl` command has the `-X` option. What does the `-X` option in `curl` do? Include annotated screenshots or images supporting your answer.

6. What does the `-H` option in `curl` do? Include annotated screenshots or images supporting your answer.

7. What does the `-d` option in `curl` do? Include annotated screenshots or images supporting your answer.

8. Based from your tracefile, the source code, and from the results of your `curl` commands, describe the following URIs:

   (a) `/zodiac/chinese`

   (b) `/zodiac/chinese/<animal>`

   Do the URIs pertain to different resources? What are the resources? Include annotated screenshots or images supporting your answer.

9. The 2nd `curl` command and the 5th `curl` command do the same thing. What are the resulting HTTP response messages? What are the events that may have affected the results of the `curl` commands? Include annotated screenshots or images supporting your answer.

10. The 3rd `curl` command and the 9th `curl` command do the same thing. What are the resulting HTTP response messages? What are the events that may have affected the results of the `curl` commands? Include annotated screenshots or images supporting your answer.

5.2 Part 2: Extended RESTful application I (Western Zodiac)

Answer the following based on `labexercise13_B_client.pcapng` or `ifconfig` results.

1. Filter out (remove) all non-HTTP-related packets by applying the "http" filter. What is the IP address of the client computer? What is the IP address of the server computer? Include annotated screenshots or images supporting your answer.

2. Show that your extended RESTful server application works by identifying the packets exchanged between the client and the server whenever the user executes the `curl` commands. That is, you should be able to show that the results returned by the server correspond to the description of the resources in Tables 1 and 2. Include annotated screenshots or images supporting your answer (I expect several).

3. Are the `curl` commands specified in this part similar to the resources found in the previous part? Include annotated screenshots or images supporting your answer.

4. Did you get results that are similar to the results in the previous part? Include annotated screenshots or images supporting your answer.

5. On the 6th and the 8th `curl` command, the following options are included:

```
-H "Accept: application/json" \
-H "Content-Type: application/json" \
```

How do these options actually affect the command? Cite references supporting your answer.

5.3 Part 3: Extended RESTful application II (Zodiac Reader)

Answer the following based on `labexercise8B_C_client.pcapng`.

1. Filter out (remove) all non-TCP-related packets by applying the "tcp" filter. What is the IP address of the client computer? What is the IP address of the server computer? Include annotated screenshots or images supporting your answer.

2. Show that your extended RESTful server application works by identifying the packets exchanged between the client and the server whenever the user executes the `curl` commands. That is, you should be able to show that the results returned by the server correspond to the description of the resource in Table 3. Include annotated screenshots or images supporting your answer (I expect several).

3. There are two HTTP requests sent to the server. What is the URI of each request? Include annotated screenshots or images supporting your answer.

6 Submission

You can submit the laboratory report via UVLE. Deadline should also be posted in UVLE, along with this document.

7 Troubleshooting

- Please do not modify or remove `chinesezodiac.json` and `westernzodiac.json`. Those files are the seed data of the databese (`database.json`).

- If you want to reset the values in the database (`database.json`), run the following command:

  ```
  python3 seeder.py
  ```