CS 315: Computer Networks Lab
Spring 2024-25, IIT Dharwad
Assignment-3
Wireshark Lab: HTTP
January 20, 2025

Lab Instructions

- Please leave your bags on the Iron shelf near the SP16 entrance.
- Login to the Ubuntu OS on your machine. The login credentials are as follows:
  - Username: user
  - Password: 123456
- Mark your attendance in the attendance sheet before leaving the lab.
- Handle the lab resources with utmost care.
- Please go through the following exercises in today's lab.
- It is recommended that you complete all the following exercises during the lab slot itself.
- If you face any difficulties, please feel free to seek help online or from your peers or TAs.
- After finishing all exercises, please carry your solutions with you (via email/pen drive) for future reference, and delete the files from the desktop.

Introduction

Having gotten our feet wet with the Wireshark packet sniffer in the introductory lab, we're now ready to use Wireshark to investigate protocols in operation. In this lab, we'll explore several aspects of the HTTP protocol: the basic GET/response interaction, HTTP message formats, retrieving large HTML files, HTML files with embedded objects, and HTTP authentication and security.

Part-1: The Basic HTTP GET/response interaction

Do the following:

- Start your web browser, and make sure your browser's cache is cleared (or try *Incognito mode* in Google Chrome / *Private Browsing* in Mozilla Firefox).
- Start the Wireshark packet sniffer, as described in the Introductory lab (but don't yet begin packet capture). Enter "http" (just the letters, not the quotation marks) in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window. (We're only interested in the HTTP protocol here, and don't want to see the clutter of all captured packets)
- Enter these links on your browser one after another:

  `http://httpforever.com/`

  `http://web.simmons.edu/`

  `http://www.vulnweb.com/`

  `http://www.testingmcafeesites.com/`

  Your browser should display a very simple HTML file.
- Stop Wireshark packet capture.

Answer the following questions based on the information you observed when tracing the above HTTP requests and responses.

1. What type of HTTP version do you observe in the above trace?
2. Mention the HTTP request method used to request these four websites.
3. What are the status codes returned by the server to the browser?
4. List the number of HTTP GET requests and frame number for "`http://httpforever.com/`" request.
5. What languages does your browser indicate that it can accept to the server?
6. List the source and the destination IP address details for all the above HTTP requests.
7. By inspecting the raw data in the packet content window, do you see any headers within the data that are not displayed in the packet-listing window? If so, name one.
8. For each url requested, provide the number of bytes returned by the server corresponding to the first HTTP GET request.
9. Obtain the number of lines of data and the content being received by your browser for the above first HTTP GET requests.

Part 2: The HTTP CONDITIONAL GET/response interaction

Do the following:

- Start up your web browser, and make sure your browser's cache is cleared.
- Start up the Wireshark packet sniffer
- Enter the following URL into your browser http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html Your browser should display a very simple five-line HTML file.
- Quickly enter the same URL into your browser again (or simply select the refresh button on your browser)
- Stop Wireshark packet capture, and enter "`http`" in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window.

Answer the following questions:

1. Inspect the contents of the first HTTP GET request from your browser to the server. Do you see an "IF-MODIFIED-SINCE" line in the HTTP GET?
2. Inspect the contents of the server response. Did the server explicitly return the contents of the file? How can you tell?
3. Now inspect the contents of the second HTTP GET request from your browser to the server. Do you see an "IF-MODIFIED-SINCE:" line in the HTTP GET? If so, what information follows the "IF-MODIFIED-SINCE:" header?
4. What is the HTTP status code and phrase returned from the server in response to this second HTTP GET? Did the server explicitly return the contents of the file? Explain.

Part 3: Retrieving Long Documents

In our examples thus far, the documents retrieved have been simple and short HTML files. Let's next see what happens when we download a long HTML file.

Do the following:

- Start up your web browser, and make sure your browser's cache is cleared.
- Start up the Wireshark packet sniffer
- Enter the following URL into your browser http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file3.html Your browser should display the rather lengthy US Bill of Rights.
- Stop Wireshark packet capture, and enter "`http`" in the display-filter-specification window, so that only captured HTTP messages will be displayed.

In the packet-listing window, you should see your HTTP GET message, followed by a multiple-packet TCP response to your HTTP GET request. This multiple-packet response deserves a bit of explanation; the HTTP response message consists of a status line, followed by header lines, followed by a blank line, followed by the entity-body. In the case of our HTTP GET, the entity-body in the response is the entire requested HTML file.

Answer the following questions:

1. How many HTTP GET request messages did your browser send? Which packet number in the trace contains the GET message for the Bill of Rights?
2. Which packet number in the trace contains the status code and phrase associated with the response to the HTTP GET request?
3. What is the status code and phrase in the response?
4. How many data-containing TCP segments were needed to carry the single HTTP response and the text of the Bill of Rights?

Part-4: HTML Documents with Embedded Objects

Now that we've seen how Wireshark displays the captured packet traffic for large HTML files, we can look at what happens when your browser loads a webpage with embedded objects, i.e., a file that includes other objects (in the example below, image and video files) that are stored locally.

Do the following:

- Use the following code to create an HTML webpage named `embedded_obj.html` as follows:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
          <meta   name="viewport"   content="width=device-width,
initial-scale=1.0">
    <title>Embedded Objects</title>
</head>
<body>
    <h1>Sample HTML with Embedded Objects</h1>
```

```
    <!-- Embedded Image -->
    <h2>Embedded Image</h2>
        <img  src="image.jpg"  alt="Example  Image"  width="500"
height="300">

    <!-- Embedded Video -->
    <h2>Embedded Video</h2>
    <video width="500" height="300" controls>
        <source src="video.mp4" type="video/mp4">
        Your browser does not support the video tag.
    </video>
</body>
</html>
```

- Download any sample image and video from the Internet and store it in the same location of this created HTML file.
- Open the terminal and type the following commands
    - `resolvectl flush-caches`
    - `python3 -m http.server 8080`
    - On executing the above command you will see the following message on the terminal `Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) …`
- Open Wireshark and choose the interface as **loopback: lo** (works only on Ubuntu)
- Start the trace on Wireshark
- Open Firefox browser and enter the following URL in the address bar: http://localhost:8080/embedded_obj.html
- Stop Wireshark packet capture, and enter "`http`" in the display-filter-specification window, so that only captured HTTP messages will be displayed.

Answer the following questions:

1. How many HTTP GET request messages did your browser send? To which Internet addresses were these GET requests sent?
2. Is the size of the image file matching with the size of the file in the Wireshark? Provide the image file size in bytes.
3. Were there any HTTP response codes indicating errors (e.g., 4xx or 5xx)? If so, what do they indicate?
4. Mention the source and destination IP addresses for the HTTP requests made to the image and video files. Explain.

Part-5: HTTP Authentication

Do the following:

- Open the browser and paste the URL: `http://testphp.vulnweb.com/login.php`
- Add the username and password as "test" and click on the "login "button
- Change any one of the field's values of any of the information presented on the screen and click on the "update" button

Answer the following questions:

1. How many GET and POST packets do you observe in the trace?
2. Write the domain and the corresponding IP address that you visited.
3. What is the web server's destination port you have requested, and is it a standard port? If yes, then which protocol?
4. Inspect the contents of the first HTTP POST request sent from your browser to the server. Do you notice an If-Modified-Since line in the HTTP POST request? If so, what is its significance?
5. Now inspect the contents of the second HTTP POST request from your browser to the server. What information follows the "IF-MODIFIED-SINCE:" header?
6. Trace the HTTP communication when accessing the authentication server for the website http://testphp.vulnweb.com/login.php. Analyze the following:
    a. Request Details: What information (e.g., username and password) is included in the HTTP request when you attempt to log in?
    b. Response Behavior: After submitting the login details, does the server respond with a new webpage? If so, what information is displayed on this page?
7. Update information
    a. Request Packet: Which packet in the HTTP trace contains the updated information you entered, and can you observe this information in the request?
    b. Response Packet: Do you see the updated information reflected in the response packet for the corresponding HTTP request?

Submission Details

- Write your answers in a single doc/tex file, and submit its PDF named after your IIT Dharwad roll number**,** which contains all answers (with screenshots, if necessary).