

NATIONAL UNIVERSITY OF COMPUTER & EMERGING SCIENCE

Computer Network Lab (CL-3001)

Lab Session 05

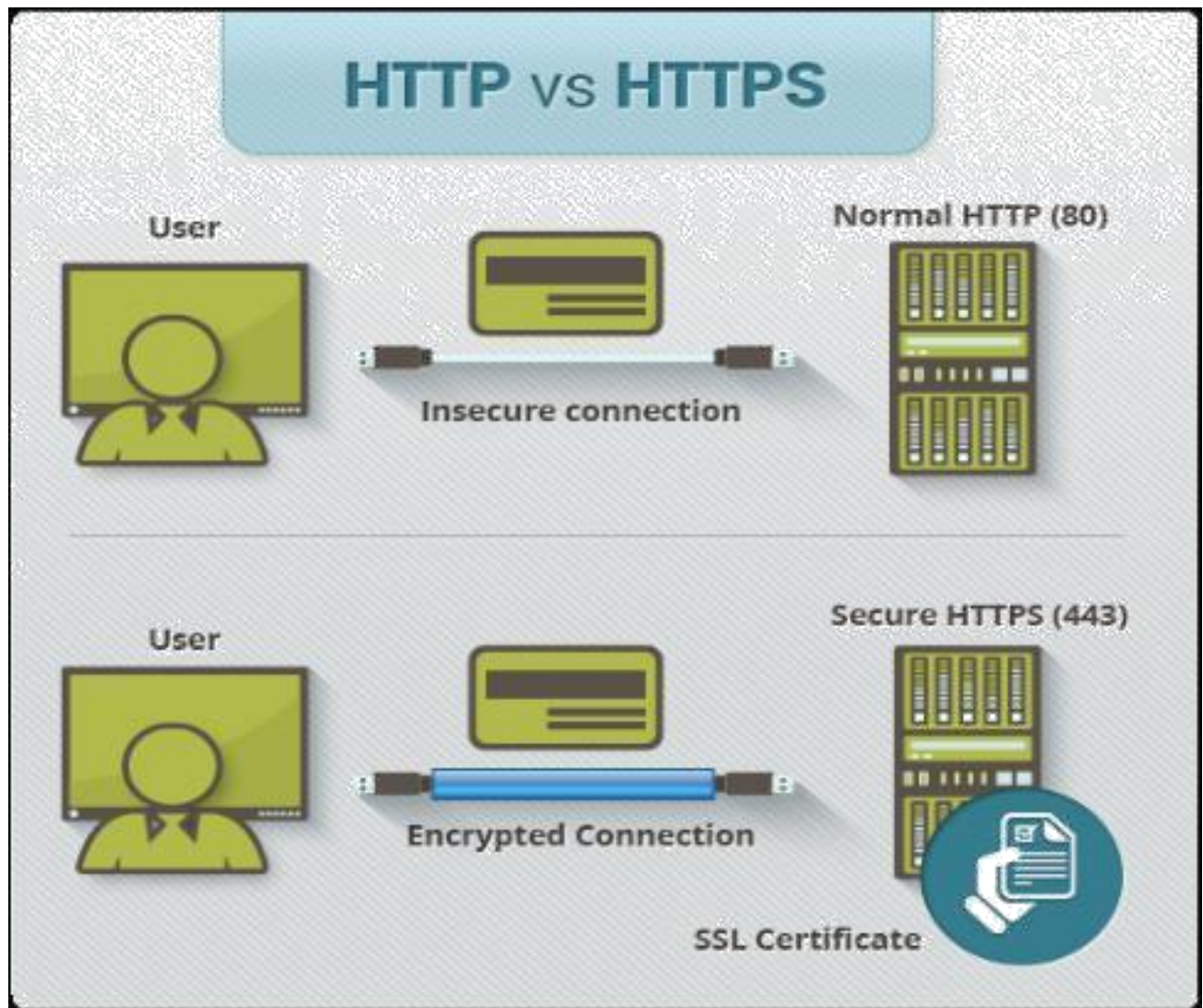
In this lab we will learn about following objectives using cisco packet tracer

- Implementation/understanding of HTTP/HTTPS
- Introduction to Wireshark (Network traffic analysis of HTTP/S protocol, Header, Cookies)

Hypertext Transfer Protocol (HTTP) is a protocol used in networking. When you type any web address in your web browser, your browser acts as a client, and the computer having the requested information acts as a server. When client requests for any information from the server, it uses HTTP protocol to do so. The server responds back to the client after the request completes. The response comes in the form of web page which you see just after typing the web address and press “Enter”.

Hypertext Transfer Protocol Secure (HTTPS) is a combination of two different protocols. It is more secure way to access the web. It is combination of Hypertext Transfer Protocol (HTTPS) and SSL/TLS protocol. It is more secure way to sending request to server from a client, also the communication is purely encrypted which means no one can know what you are looking for. This kind of communication is used for accessing those websites where security is required. Banking websites, payment gateway, emails (Gmail offers HTTPS by default in Chrome browser), and corporate sector websites are some great examples where HTTPS protocols are used.

For HTTPS connection, public key trusted and signed certificate is required for the server. These certificates come either free or it costs few dollars depends on the signing authority. There is one other method for distributing certificates. Site admin creates certificates and loads in the browser of users. Now when user requests information to the web server, his identity can be verified easily.



Here are some major differences between HTTP and HTTPS:

HTTP

- URL begins with "http://"
- It uses port 80 for communication
- Unsecured
- Operates at Application Layer
- No encryption
- No certificates required

HTTPS

- URL begins with "https://"
- It uses port 443 for communication
- Secured
- Operates at Transport Layer
- Encryption is present
- Certificates required

Client Error

The 4xx class of status code is intended for cases in which the client seems to have erred. Except when responding to a HEAD request, the server should include an entity containing an explanation of the error situation, and whether it is a temporary or permanent condition. These status codes are applicable to any request method. User agents should display any included entity to the user.

400 Bad Request

The server cannot or will not process the request due to something that is perceived to be a client error (e.g., malformed request syntax, invalid request message framing, or deceptive request routing).

401 Unauthorized (RFC 7235)

Similar to 403 Forbidden, but specifically for use when authentication is required and has failed or has not yet been provided. The response must include a WWW-Authenticate header field containing a challenge applicable to the requested resource. See Basic access authentication and Digest access authentication.

403 Forbidden

The request was a valid request, but the server is refusing to respond to it. Unlike a 401 unauthorized response, authenticating will make no difference.

404 Not Found

The requested resource could not be found but may be available again in the future. Subsequent requests by the client are permissible.

408 Request Timeout

The server timed out waiting for the request. According to HTTP specifications: "The client did not produce a request within the time that the server was prepared to wait. The client MAY repeat the request without modifications at any later time."

Server Error

The server failed to fulfill an apparently valid request.

Response status codes beginning with the digit "5" indicate cases in which the server is aware that it has encountered an error or is otherwise incapable of performing the request. Except when responding to a HEAD request, the server should include an entity containing an explanation of the error situation, and indicate whether it is a temporary or permanent condition. Likewise, user agents should display any included entity to the user. These response codes are applicable to any request method.

500 Internal Server Error

A generic error message, given when an unexpected condition was encountered and no more specific message is suitable.

501 Not Implemented

The server either does not recognize the request method, or it lacks the ability to fulfil the request. Usually this implies future availability (e.g., a new feature of a web-service API).

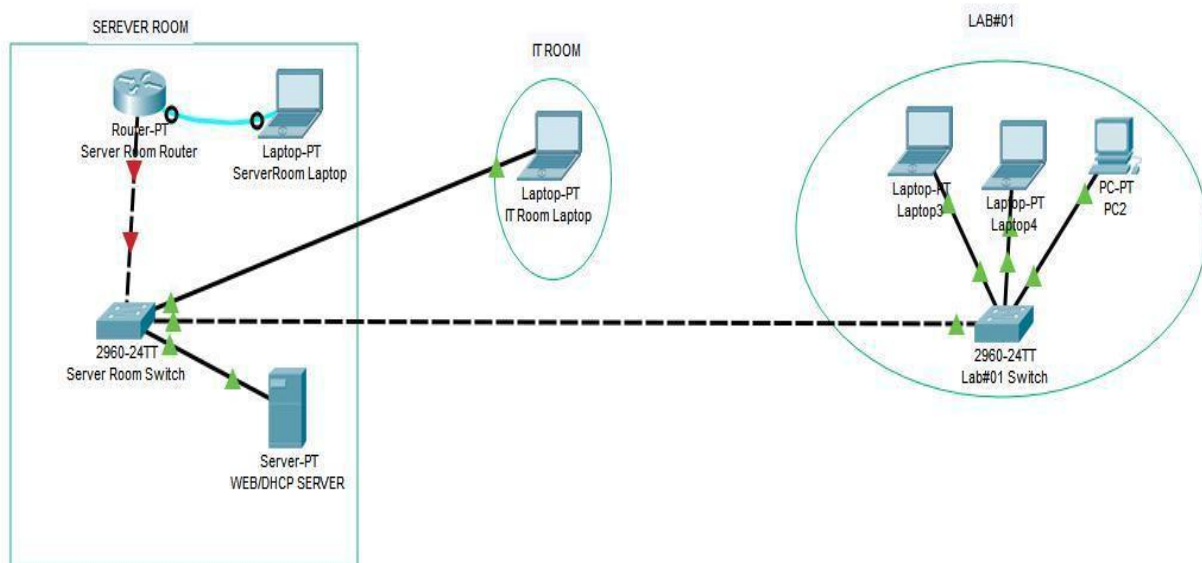
502 Bad Gateway

The server was acting as a gateway or proxy and received an invalid response from the upstream server.

503 Service Unavailable

The server is currently unavailable (because it is overloaded or down for maintenance). Generally, this is a temporary state.

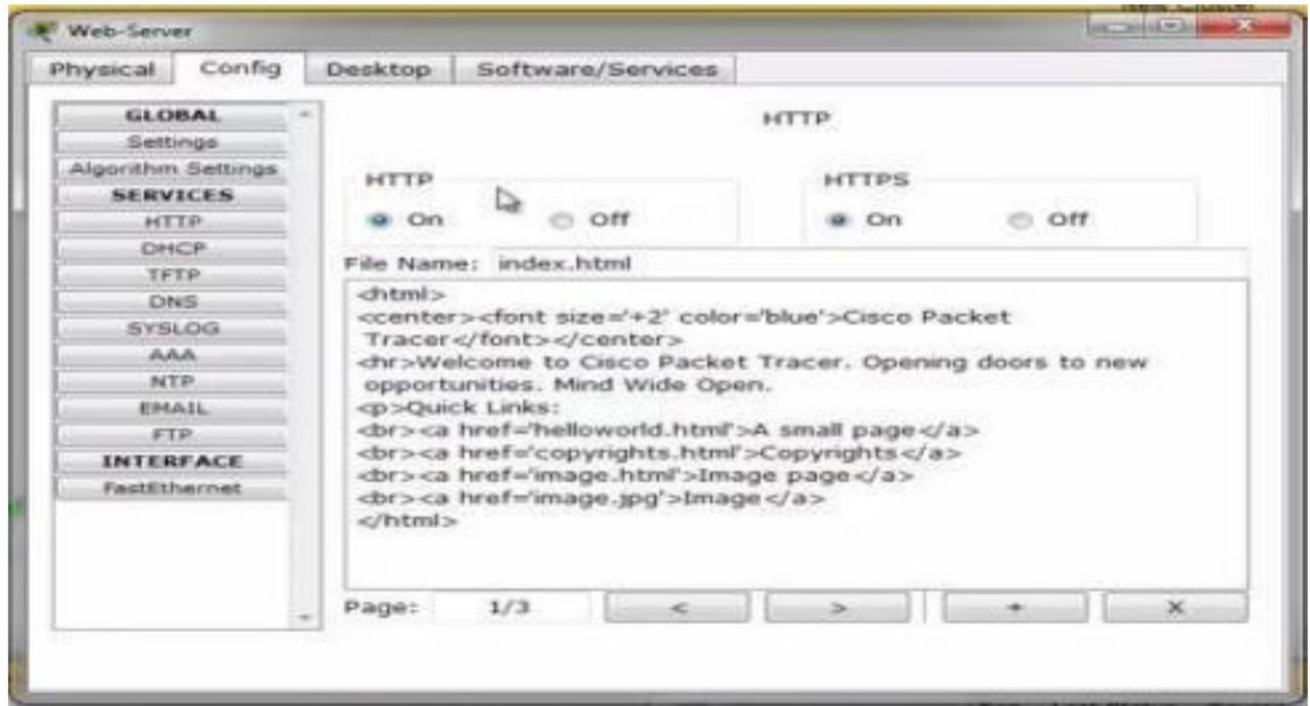
Let's design a Topology which concludes the above theoretical concepts



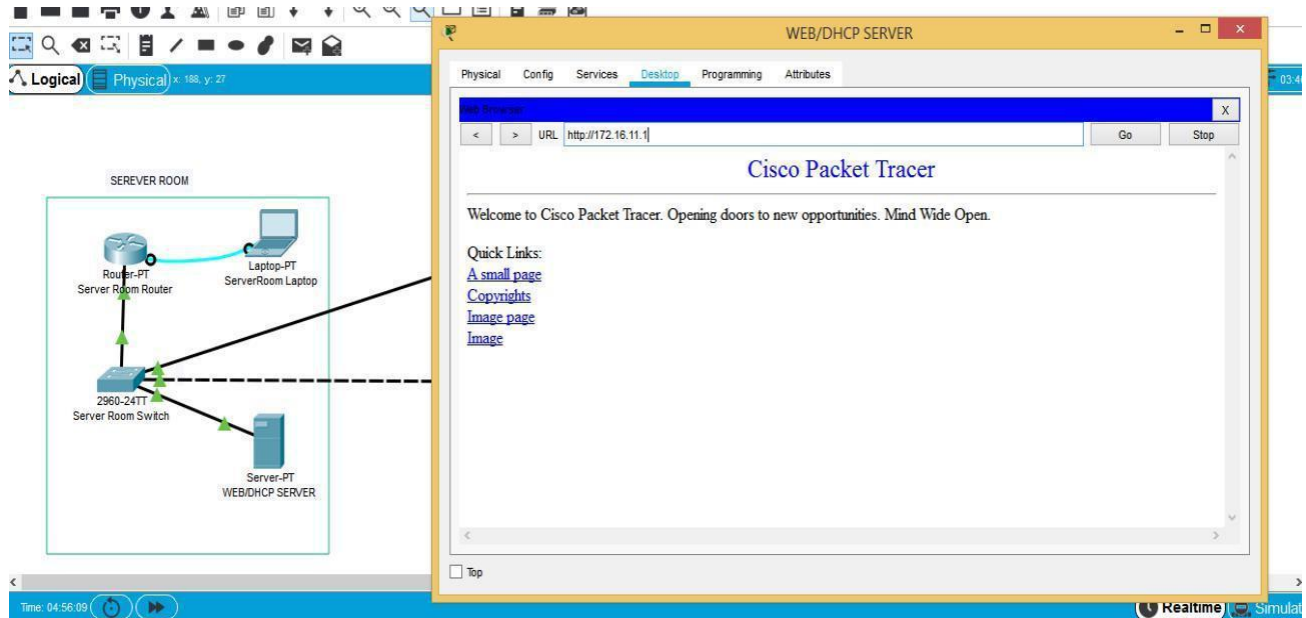
The above topology configured as “one server room”, “one IT room: and “Lab#01 environment having three system”. On our server we have enabled web services as well as dhcp services.

Click on the web server, go to config --->services--
HTTP

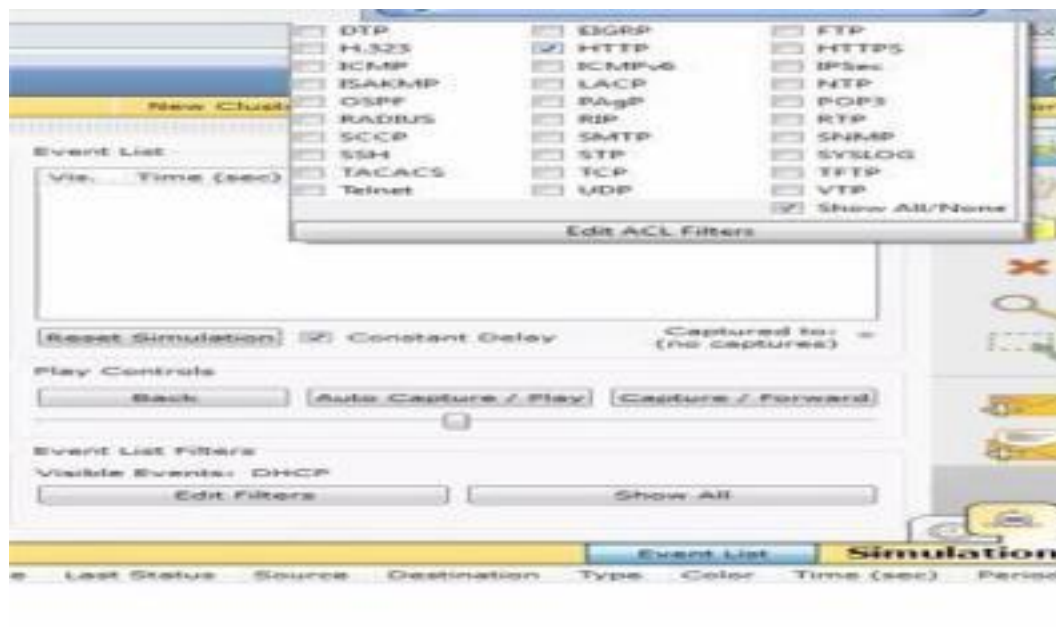
- Here you can see that http and https server are on.



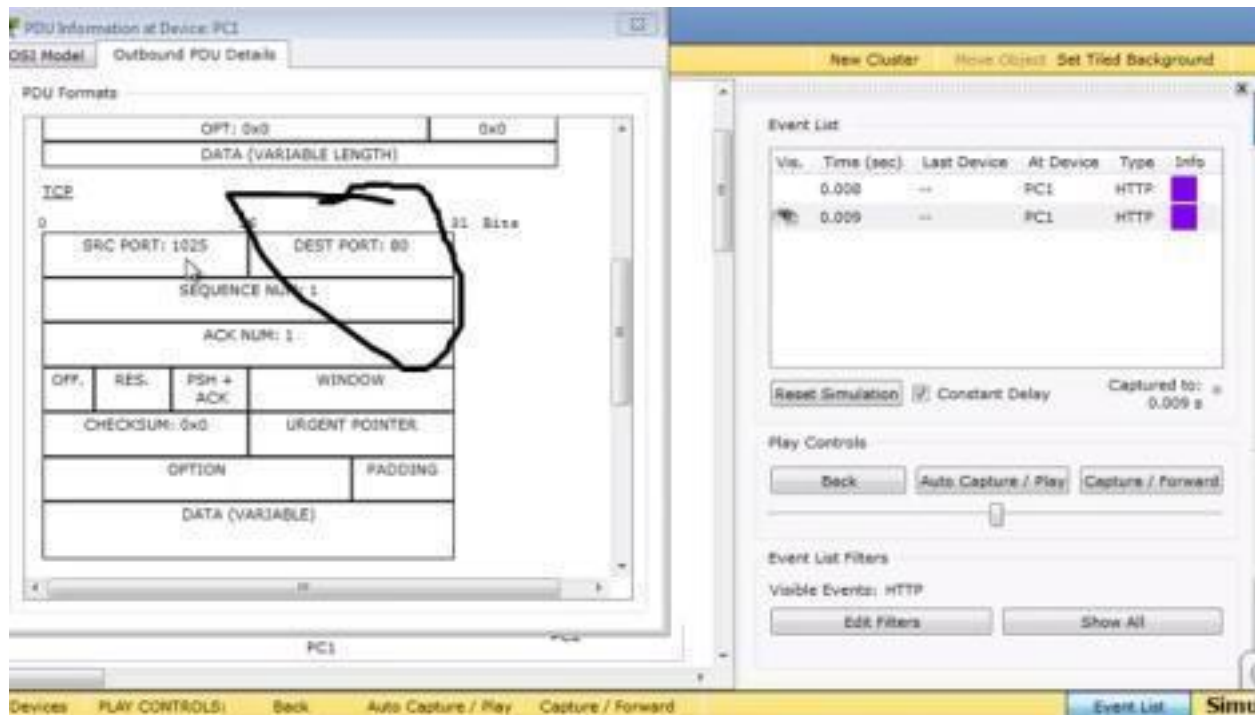
Now click on PC0 and go to Desktop -> Web Browser. Now type web-server IP 172.16.11.1



Now to note the http header format information go to simulation mode edit filters and click on http check box then click on capture/forward button.



Now click on the http packet, you can note that the destination port is 80.



Now scroll the Outbound PDU Details, you can see the http protocol information.

Wireshark: HTTP

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, retrieving HTML files with embedded objects, and HTTP authentication and security.

The Basic HTTP GET/response interaction

Let's begin our exploration of HTTP by downloading a very simple HTML file - one that is very short and contains no embedded objects.

Do the following:

1. Start up your web browser.
2. Start up 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).

3. Wait a bit more than one minute (we'll see why shortly), and then begin Wireshark packet capture.
4. Enter the following to your browser <http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html>. Your browser should display the very simple, one-line HTML file.
5. Stop Wireshark packet capture.

Your Wireshark window should look like the window shown in Figure 1. If you are unable to run Wireshark on a live network connection, you can download a packet trace that was created when the steps above were followed.

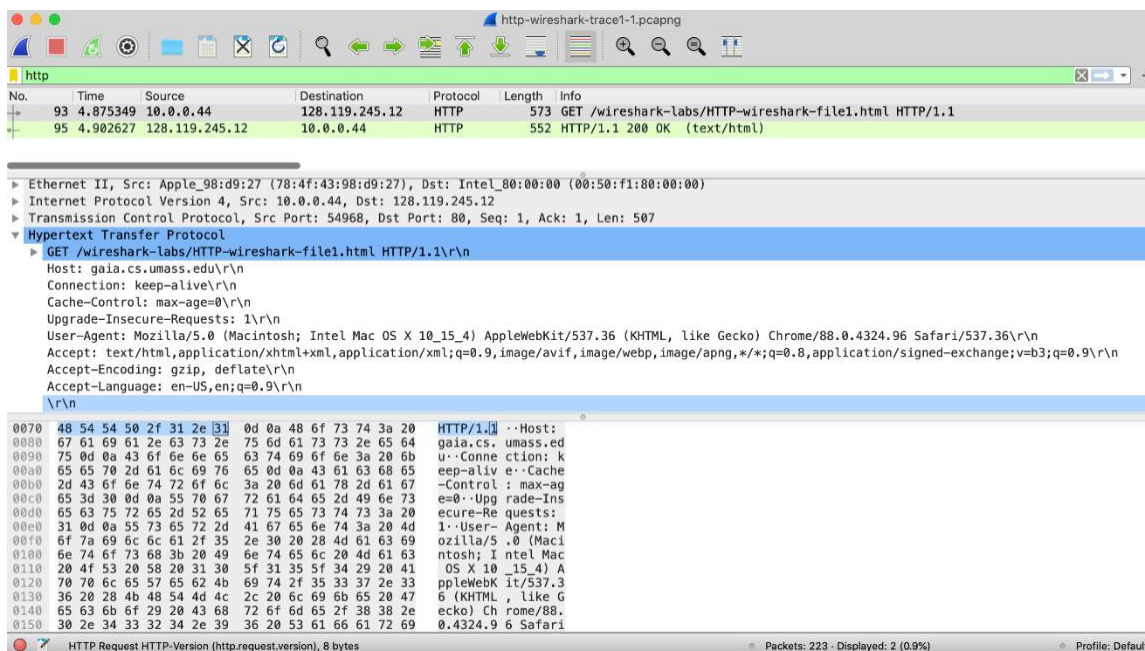


Figure 1: Wireshark Display after <http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html> has been retrieved by your browser

The example in above figure shows in the packet-listing window that two HTTP messages were captured: the GET message (from your browser to the `gaia.cs.umass.edu` web server) and the response message from the server to your browser. The packet-contents window shows details of the selected message (in this case the HTTP OK message, which is highlighted in the packet-listing window). Recall that since the HTTP message was carried inside a TCP segment, which was carried inside an IP datagram, which was carried within an Ethernet frame, Wireshark displays the Frame, Ethernet, IP, and TCP packet information as well. We want to minimize the amount of non-HTTP data displayed (we're interested in HTTP here, and will be investigating these other protocols in later labs), so make sure the boxes at the far left of the Frame, Ethernet, IP and TCP information have a plus sign or a right-pointing triangle (which means there is hidden, non-displayed information), and the HTTP line has a minus sign or a down-pointing triangle (which means that all information about the HTTP message is displayed).

By looking at the information in the HTTP GET and response messages, answer the following questions. When answering the following questions, you should print out the GET and response messages (see the introductory Wireshark lab for an explanation of how to do this) and indicate where in the message you've found the information that answers the following questions. When you hand in your assignment, annotate the output so that it's clear where in the output you're getting the information for your answer (e.g., for our classes, we ask that students markup paper copies with a pen, or annotate electronic copies with text in a colored font).

What are cookies and headers?

HTTP headers are used to pass additional information with HTTP response or HTTP requests. A cookie is an HTTP request header i.e. used in the requests sent by the user to the server

Cookies are strings of data that a web server sends to the browser. When a browser requests an object from the same domain in the future, the browser will send the same string of data back to the origin server.

The data is sent from the web server in the form of an HTTP header called "Set-Cookie". The browser sends the cookie back to the server in an HTTP header called "Cookie". Here is an example of what an HTTP cookie transaction might look like:

HTTP response from the web server:

[...]

Set-Cookie: first.lastname

HTTP GET from the client:

[...]

Cookie: first.lastname

In the sample transaction, the web server told the client to create the cookie "first.lastname". The next time the client requests an object from this domain it sends the cookie with the request. This illustrates how a web server might be able to recall certain information such as user logins.

Types of Cookies

There are two different types of cookies - session cookies and persistent cookies. If a cookie does not contain an expiration date, it is considered a session cookie. Session cookies are stored in memory and never written to disk. When the browser closes, the cookie is permanently lost from this point on. If the cookie contains an expiration date, it is considered a persistent cookie. On the date specified in the expiration, the cookie will be removed from the disk.

There are several different fields a cookie can contain, separated by semicolons. The definitions are:

expires

`expires="Wdy, DD-Mon-YYYY HH:MM:SS GMT"`

Determines when the cookie is to be deleted.

path

`path=`

Determines what path to return the cookie on. In this example, the cookie will be sent when going to the root path in a domain.

domain

`domain=whatever.domain.com`

Specifies what domain the cookie is used for. If this does not match the domain currently being browsed to, it is considered to be a "3rd Party cookie" and will be rejected by the browser. This prevents one domain setting a cookie for a different domain.

Cookies are passed as HTTP headers, both in the request (client -> server), and in the response (server -> client)

Major groups of HTTP headers include General, Request, Response and Entity Headers.

Answer these questions

1. Is your browser running HTTP version 1.0 or 1.1? What version of HTTP is the server running?
2. What languages (if any) does your browser indicate that it can accept to the server?
3. What is the IP address of your computer? Of the `gaia.cs.umass.edu` server?
4. What is the status code returned from the server to your browser?
5. When was the HTML file that you are retrieving last modified at the server?
6. How many bytes of content are being returned to your browser?
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.

In your answer to question 5 above, you might have been surprised to find that the document you just retrieved was last modified within a minute before you downloaded the document. That's because (for this file), the `gaia.cs.umass.edu` server is setting the file's last-modified time to be the current time and is doing so once per minute. Thus, if you wait a minute between accesses, the file will appear to have been recently modified, and hence your browser will download a "new" copy of the document.

HTTP Authentication

Finally, let's try visiting a web site that is password-protected and examine the sequence of HTTP message exchanged for such a site. The URL http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wireshark-file5.html is password protected. The username is "Wireshark-students" (without the quotes), and the password is "network" (again, without the quotes). So, let's access this "secure" password-protected site. Do the following:

Make sure your browser's cache is cleared, as discussed above, and close your browser, then, start up your browser, then start up the Wireshark packet sniffer.

Enter the following URL into your browser http://gaia.cs.umass.edu/wireshark-labs/protected_pages/HTTP-wireshark-file5.html

Type the requested user name and password into the pop up box.

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.

Note: If you are unable to run Wireshark on a live network connection, you can use the http-ethereal-trace-5 packet trace to answer the questions below; see footnote 2. This trace file was gathered while performing the steps above on one of the author's computers.)

Now let's examine the Wireshark output. You might want to first read up on HTTP authentication by reviewing the easy-to-read material on "HTTP Access Authentication Framework" at [http://frontier.userland.com/stories/storyReader\\$2159](http://frontier.userland.com/stories/storyReader$2159)

Answer the following questions

1. What is the server's response (status code and phrase) in response to the initial HTTP GET message from your browser?
2. When your browser's sends the HTTP GET message for the second time, what new field is included in the HTTP GET message?

The username (wireshark-students) and password (network) that you entered are encoded in the string of characters (d2lyZXNoYXJrLXN0dWRIbnRzOm5ldHdvcms=) following the "Authorization: Basic" header in the client's HTTP GET message. While it may appear that your username and password are encrypted, they are simply encoded in a format known as Base64 format. The username and password are not encrypted! To see this, go to <http://www.motobit.com/util/base64-decoder-encoder.asp> and enter the base64-encoded string d2lyZXNoYXJrLXN0dWRIbnRz and decode. Voila! You have translated from Base64 encoding to ASCII encoding, and thus should see your username! To view the password, enter the remainder of the string Om5ldHdvcms= and press decode. Since anyone can download a tool like Wireshark and sniff packets (not just their own) passing by their network adaptor, and anyone can translate from Base64 to ASCII (you just did it!), it should be clear to you that simple passwords on WWW sites are not secure unless additional measures are taken.

EXERCISE

Q1) In caching, what is the difference between the age header and expires?

Q2) What are the four groupings of HTTP headers?

Q3) Implement the below topology using web server.

