Exercises on Exploring HTTP Protocol

Experiential Learning Workshop

BITES-BMSCE

Sep 27, 2018

# General Guidelines

1. Make a team of three unless stated otherwise.
2. For each exercise, use wireshark/tcpdump capture to verify contents
3. Ensure to use proper capture filter and don’t capture unnecessary traffic
4. Where appropriate or applicable, use wget or nc to access the web server.
5. The default client for accessing web server is assumed to be browser, preferably firefox. You can use Chrome or any other browser as well.
6. The webserver in the example below is taken as ‘myweb.com’. Please use your hostname or corresponding IP address instead in your exercise.
7. To kill any program in the linux terminal, please press Ctrl-C and not Ctrl-Z. The latter will suspend the program and not stop it.

Note: Appendix provides instructions on installing any package if not already installed.

# Hands-on 1: Tools

## Using wget

1. Open terminal
2. Mimic (option –mk) your college website <http://www.bmsce.in>, and access locally (turn off your internet).
3. Download a large file using the --limit-rate=1m e.g. <http://rprustagi.com/workshops/bites/bmsce/movie.mp4>, break the download by pressing Ctrl-C after about 5MB is downloaded and then download with resume option (-c). Ensure full download occurs and see if you can watch the movie after complete download.
4. Explore other options such as –d for debug headers, -O to save into a file,

## Using nc

1. Open terminal on two machines.
2. Identify each other’s IP address. You can use the command ip addr in the linux terminal, to know the IP address of Ethernet interface. Do not use the IP 127.0.0.1 of lo interface.
3. Run as TCP server on some port e.g. 2345 (nc –l 2345) in one terminal and UDP server (nc –u –l 3456) in another terminal.
4. Connect using clients (from another machine) to both TCP and UDP server and do chat.
5. Analyze wireshark capture of your chat conversation.
6. Transfer some files across machines e.g. cat “file”| nc “serverIP” “serverPort” on the client side and on server side (nc –l “port” >”file”)
7. Login in to remote machine without authentication

## Using Wireshark

1. Open wireshark
2. Select the applicable interface e.g. enp0s1 or eth0.
3. Specify the capture filter to capture traffic with other e.g. ‘host www.bmsce.in’
4. Click start
5. Access few web pages on the website <http://www.bmsce.in>. Analyze layer 2, layer 3, Layer 4 and Layer 7 packets.

## Creating a host name **myweb.com**

1. Assume that the server on which Apache is installed, has the IP Address 10.10.1.1. Create an entry in hosts file as follows (/etc/hosts for Linux, and \windows\system32\drivers\etc\hosts for windows). Use the appropriate IP Address of your server in place of 10.10.1.1.

10.10.1.1 myweb.com

## Using ping

1. Ping google.com and yahoo.com by sending some fixed count packets e.g. 20. Analyze the response times and variation in response times.
2. Ping myweb.com with count of 10 pacekts.
3. Ping these sites again in quite mode.
4. Use ping with changing interval duration to 0.2s from the default of 1s as well as changing packet size from 56bytes to 1000 bytes.
5. Use wireshark to analyze

# Hands-on 2: Basics of HTTP

The example programs mentioned below are listed in appendix. Refer to the same when there is a need to create such programs. You can any text editor e.g. nano, gedit or vi to create these text files.

## Status Code 200

1. Access your college website with wget using debug options e.g.
	1. wget –d <http://www.bmsce.in>
2. Analyze HTTP request and response message i.e.
	1. Request line
	2. HTTP Request headers
	3. Status line of HTTP Response
	4. HTTP Response headers.
3. Create a simple webpage e.g. welcome.html (7.1). Save this page in DocumentRoot directory of Apache web server. Typically, by default this value is /var/www/html.
4. Access the web page in the browser on a different machine i.e. http://<serverIP>/welcome.html
5. Verify the status code 200 and other required headers.
6. Access the same webpage using ‘wget –d’ and verify the status code.
7. Access the same webpage using ‘nc’ and verify the status code

nc myweb.com 80

GET /welcome.html HTTP/1.1

Host: myweb.com

Note: At times nc may result in web server responding with status code 400. This happens when web server does not get HTTP request headers with CR/LF hence responds back. So, in such case, create a text file e.g. req-normal.txt

## Content-Type

1. Copy welcome.html file as welcome.txt
2. Access the url http://myweb.com/welcome.txt.
3. Look at the content displayed on browser.
4. Analyze the wireshark capture to study the header Content-Type:
5. Repeat the exercise with wget.
6. Study the headers.

## Status code 404

1. Access a non-existence webpage e.g nonexist.html
2. Check the status code in wireshark.
3. Verify this status code using wget as well.

## Status code 403

1. Copy welcome.html file as restricted.html
2. Remove the read permission for others i.e. issue the command on the web server (myweb.com)

chmod o-r retricted.html

1. Access this file using browser i.e. http:/myweb.com/restricted.html
2. Browser should display the contents of the file without formatting.
3. Analyze the response in the browser. Analyze HTTP status codes.

## Status code 400

1. To experience this access code, we need to use nc. By default, both browser and wget send the proper HTTP header.
2. This exercise requires that client should send invalid header e.g. ‘Host myweb.com’ instead of ‘Host: myweb.com’. Please note that HTTP header field name should be separated by its value by Colon (:) character.
3. Create a text file e.g. req-badhdr.txt (7.3) with bad headers.

GET /welcome.html HTTP/1.1

Host 10.211.55.9

User-Agent: BMSCE and BITES

Accept: \*/\*

Accept-Language: en-us,en;q=0

1. Send the bad headers using nc e.g.

cat req-badhdr.txt | nc myweb.com 80

1. Analyze the response given by web server and verify that it corresponds to ‘400 Bad Request’.
2. Make another access with different header with syntax error.
3. Verify the Bad Request error

# Hands-on 3: Understanding HTTP

## Using Accept-Language.

1. Change the preferred language setting in firefox browser to language of your choice.
2. Access google.com.
3. The browser should display the content of web page in your preferred language.

## Status code 301 (or 302)

1. Access google.com using wget –d. Identify the first response.
2. On your web server (myweb.com) make following changes in Apache config file /etc/apache2/sites-enabled/000-default.conf.

Redirect /oldwelcome.html /welcome.html

1. Restart Apache webserver (sudo service apache2 restart)
2. Access the url [http://myweb.com/oldwelcome.html](http://mywww.com/oldwelcome.html)
3. Verify that content is served from the file welcome.html
4. Verify that status code 301 being returned and second access to new url in the wireshark
5. Repeat the exercise using ‘wget’ and verify HTTP redirect.
6. Analyze all HTTP headers and status code in wireshark

## Using chunk based delivery

1. In firefox, access <http://rprustagi.com/workshops/web/chunk-xfer.php> . See how the image is incrementally displayed.
2. Create a php file like chunk-xfer.php (7.4) to generated HTTP Response in terms of chunks.
3. Copy your favourite image file (e.g. your own photo) and save it in DocumentRoot.
4. Ensure that your image filename is correctly described in the program file chunk-xfer.php
5. From the browser, access the URL <http://myweb.com/chunk-xfer.php>
6. Analyze the HTTP response in wireshark and identify chunks.
7. Edit the program file and change chunk sizes or mimic the internet delay i.e. sleep time.
8. Re-access the URL and analyze the response.

## Status 206: Partial Content Delivery

1. Create a simple text file req-partial.txt (7.5) to send the HTTP headers requesting specific range of content.
2. Make a request to web server for partial content delivery i.e.
3. Analyze the HTTP response headers as well wireshark capture.

## Using compression

1. To ensure that HTTP Request contains header corresponding to compression, create a simple text file e.g. req-gzip.txt (7.6) containing header Accept-Encoding: gzip.
2. Make a request using this header e.g.

cat req-gzip.txt | nc myweb.com 80 >abc.html.gz

1. Analyze the response. Uncompress (gunzip abc.html.gz) it to get the original contents.
2. Make a request using wget for HTTP response with compression. Use the wget option --header to make such a request.

wget --header=”Accept-Encoding: gzip” <http://myweb.com/welcome.html> -O welcome.html.gz

1. Analyze the response as well as wireshark capture.

# HTTP Persistent Connections and Caching

## HTTP Non-Persistent Connections.

1. Configure Apache web server with KeepAlive Off and restart.
2. Create a web page (e.g. pictures.html) with multiple embedded images (say 10) images as in <http://rprustagi.com/workshops/web/pictures.html>
3. Access the web page from your local web server with KeepAlive Off in firefox browser and do a wireshark capture. How many TCP connections you notice. There should be as many connection as number of embedded objects plus 1.

## HTTP Persistent Connections.

1. Configure Apache web server with following configurations.
	1. KeepAlive On.
	2. MaxKeepAliveRequests 10
	3. KeepAliveTimeout 5
2. Restart Apache.
3. Access the web page again with firefox browser. Analyze the number of TCP Connections. By default, firefox makes 6 concurrent TCP connections. You should see similar number and on some connections you should see two or more HTTP requests (e.g. images).
4. Configure Firefox to setup only 3 persistent connection.
	1. Type about:config in firefox browser.
	2. Search the field *max-persistent-connections-per-server*
	3. Set the value to 3.
5. Access the page again and analyze the number of TCP Connections.
6. Refresh the page after 5 seconds. Analyze setup of new TCP connections.
7. Refresh the web page within 4 seconds multiple times e.g. 10 times. Analyze the wireshark capture on when does a new TCP connection is made.
8. Tweak (or reconfigure) the value MaxKeepAliveRequests to your other values.
9. Continue to refresh the page multiple times less than configured timeout value (e.g. 5s). Analyze and understand when does a browser make a new TCP Connection.

## Using Caching.

1. Using wget – <http://rprustagi.com/workshops/web/welcome.html> , note down value of response header “Last-Modified:”, as well as that of “Etags:”
2. Using wget header option, pass on the value corresponding to response header “Last-Modified” in previous step, e.g.

wget --header=”If-Modified-Since: Tue, 03 Jul 2018 17:27:18 GMT”

1. Analyze the response and verify that status is 304 and not 200.
2. Create such a file on your webserver (myweb.com) and repeat the exercise.
3. On the web server modify the date/time of the file using touch command e.g.

touch welcome.html

1. Make the request again using wget as in step 2. Note down the header values of “Last-Modified:” and “Etags:”
2. You should get the full content with status code 200.
3. Modify the date/time of welcome.html file
4. Make another request using the header i.e. “If-None-Match: “ with the value corresponding to “Etags:” in the response as in step. e.g.

wget -d --header='If-None-Match: "xxxx…"' http://myweb.com/accs/welcome.html

1. Analyze the response and HTTP Status code.

# HTTP Authentication and Dynamic Web Access

## HTTP Authentication.

1. Modify apache config file to specify a directory that requires authentication e.g

<Directory /var/www/html/private>

 AuthType Basic

 AuthName “For Bites Workshop”

 AuthBasicProvider file

 AuthUserFile /etc/apache2/passwdfile

 Require valid-user

</Directory>

1. Using the tool htpasswd, create a username and password for web authentication e.g.

sudo htpasswd /etc/apache2/passwdfile ksit

1. Create the directory /var/www/html/private as specified in directive “<Directory ” in the step 1.
2. Create a file e.g. restricted.html in this authentication directory

 sudo touch /var/www/html/private/restricted.html

1. Access the URL <http://myweb.com/private/restricted.html>.
2. Enter the username (e.g. ksit) but incorrect password. Understand the response.
3. Next time enter the correct username and password.
4. Analyze the wireshark capture. Compute your own value of base64 for username and password and compare it with the value in wireshark capture.

## Dynamic web - success

1. Enable apache to use cgi-bin and restart e.g.

sudo a2enmod cgi

sudo service apache2 restart

1. Create executable program goodcgi.sh (7.7) in cgi-bin directory for apache web server e.g. /usr/lib/cgi-bin.
2. Access this URL e.g. <http://myweb.com/cgi-bin/goodcgi.sh>
3. Analyze the response and wireshark capture.

## Dynamic web – Internal server.

1. Copy the file goodcgi.sh to badcgi.sh

Sudo cp goodcgi.sh badcgi.sh

1. Edit badcgi.sh (7.8) to comment the line which outputs empty line after the header. As a result, when this script is executed, there will not be separator empty line between HTTP header and web page content.
2. Access the URL <http://myweb.com/cgi-bin/badchi.sh> corresponding to bad cgi program.
3. You should get the response as 500 Internal Server Error.
4. Modify the badcgi.sh to uncomment the empty line output but insert a bad header format e.g. “Server myweb.com”. Please note that do not put colon(:) between header name and value. This makes it as a bad header.
5. Re-access the URL and analyze the response.

# Appendix A

# A1: Sample Programs

## A1: welcome.html

<html>

 <head>

 <title>Welcome Page</title>

 </head>

 <body>

 <h1>Welcome to HTTP Learning</h1>

 Welcome to experiential learning of HTTP protocol.

 </body>

</html>

## A2: req-normal.txt

Note:: There should be an empty line at the end

GET /welcome.html HTTP/1.1

Host: ksit

User-Agent: KSIT with BITES

Accept: \*/\*

Accept-Language: en-us,en;q=0.5

↵

## A3: req-badhdr.txt

GET /accs/welcome.html HTTP/1.1

Host 10.211.55.9

User-Agent: KSIT and BITES

Accept: \*/\*

Accept-Language: en-us,en;q=0.5

↵

## A4: chunk-xfer.php

$file = '../img/img-07.jpg';

if (is\_file($file)) {

 header('Content-Type: image/jpeg');

 header('Transfer-Encoding: chunked');

 $chunkSize = 1000;

 $handle = fopen($file, 'rb');

 while (!feof($handle)) {

 $buffer = fread($handle, $chunkSize);

 # send chunk size in hex, chunk content, empty new line

 echo sprintf("%x\r\n", $chunkSize);

 echo $buffer; echo "\r\n";

 ob\_flush(); flush();

 usleep(500000); # emulate network latency

 }

 fclose($handle);

 exit;

} else {

 header('Content-Type: text/html');

 echo "\r\nNo picture available for $file\r\n";

}

?>

## A3: req-partial.txt

GET welcome.html HTTP/1.1

Host: myweb.com

User-Agent: KSIT and BITES

Accept: \*/\*

Accept-Language: en-us,en;q=0.5

Range: bytes=50-100

↵

## A3: req-gzip.txt

GET welcome.html HTTP/1.1

Host: myweb.com

User-Agent: KSIT and BITES

Accept: \*/\*

Accept-Language: en-us,en;q=0.5

Accept-Encoding: gzip

↵

## A3: goodcgi.sh

#!/bin/bash

# program logic

echo "Content-type: text/plain";

echo "";

echo "<h1>hello! welcome</h1>";

## A3: badcgi.sh

#!/bin/bash

# program logic

echo "Content-type: text/plain";

#echo "";

echo "<h1>hello! welcome</h1>";

# B: Installing Apache and other programs

## B1: Installing Apache

$ sudo apt-get install apache2

## B2: Installing Wireshark

$ sudo apt-get install wireshark

Running wireshark

$ sudo wireshark

## B3: Overcoming DNS resolution.

To ensure that your webserver name myweb.com is resolved to a proper IP address in the client machine, make an entry in /etc/hosts file of client machine like below.

10.1.1.101 myweb.com

## B4: Enabling php in Apache

$sudo apt-get install php5 libapache2-mod-php5

$ sudo service apache2 restart

🡨 end of exercises handout 🡪