# CS447-Network and Data Communication
# Project Specification/Project 1, Fall 2017

# Due October 3, 2017

## 1. Introduction

In this project, we will design and implement "multi-threaded application-layer security firewall" using the Windows standard socket interface ("Winsock"). Security firewall is an intermediate server (A.K.A, "proxy"), which works as a middleman between clients and a server as shown in Fig. 1 below. The example in the figure shows an application-layer security firewall for a web server, but such a security firewall can be used virtually for any client-server based network applications. In the figure, the proxy behaves as if it were the real web server to clients. The client first transmits a HTTP GET request (as a text message) to the proxy, requesting a target web page. Then, the proxy scans the contents in the GET command, looking for any possibility hazardous contents to be removed before such hazardous contents reach the real web server. The security firewall also monitors any hazardous contents to prevent any secure information from leaking to the outside of the web server.

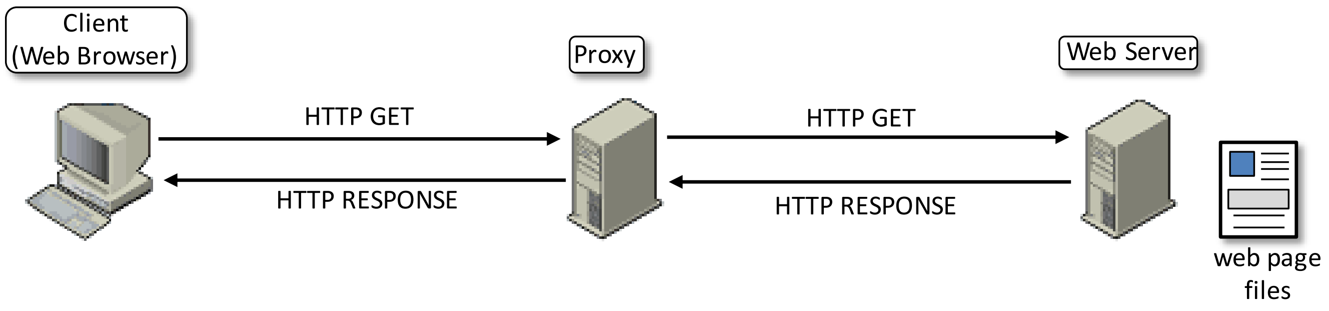

**Fig. 1 – Concept of an application-layer security firewall as a proxy**

## 2. Requirements

The security firewall we design and implement for this project should satisfy the following requirements:

(a) The security firewall accepts HTTP requests from the existing commercial web browsers (MS IE, FireFox, Netscape Navigator, and etc.) at port 9080 as a proxy.

(b) The IP address (and the port number) of a real web server can be hard coded in the proxy (should be defined as a constant using #define pre-compiler directive).

(c) The security firewall should establish a TCP connection with a client for each HTTP GET request session as a proxy (web browsers try to establish a TCP connection with a web server for each HTTP GET request session).

(d) After the proxy accepts a HTTP GET request from a client, and if there is no hazardous contents in the HTTP GET, it establishes a TCP/IP connection to a real web server at a remote server, which should be listening to port 80. If any hazardous contents are detected, the proxy should respond to the requesting client by "HTTP 401: Unauthorized Access" and the proxy should drop the TCP connection.

(e) When the remote web server responds, first, the proxy should receive the response, and scan the contents in the server's response, looking for any hazardous contents in the response. If any hazardous contents are detected, the proxy should immediately drop the TCP connection with the client and the web server.

(f) Most of the commercial web browsers generate multiple HTTP requests in parallel. Thus, your proxy should be able to handle such concurrent multiple HTTP requests to a remote web server (and the responses from the remote web server should be correctly processed) using multiple concurrent threads.

(g) No data should be dropped at the proxy for authorized requests and responses (those that do not contain any hazardous contents).

(h) The proxy should work with any existing commercial web browsers and web servers.

(i) All the responses to a client's HTTP request must come from a web server (i.e., the proxy can't synthesize any web contents).

(j) The proxy should be implemented for MS Windows platform using C++ (Visual C++ and W32 environment).

(k) Two hazardous contents should be declared for each direction (client to server and server to client). Each of hazardous contents should be declared as a char string. For example, the first hazardous contents in the messages from a client to server should be declared as "char hazardous_contents_CS_01[256];". The four hazardous contents should be declared at the beginning of "main" in your C++ source code file.

## 3. Suggested proxy organization

The following figure (Fig. 2) shows a suggested proxy structure. The suggested organization is not the mandatory structure.

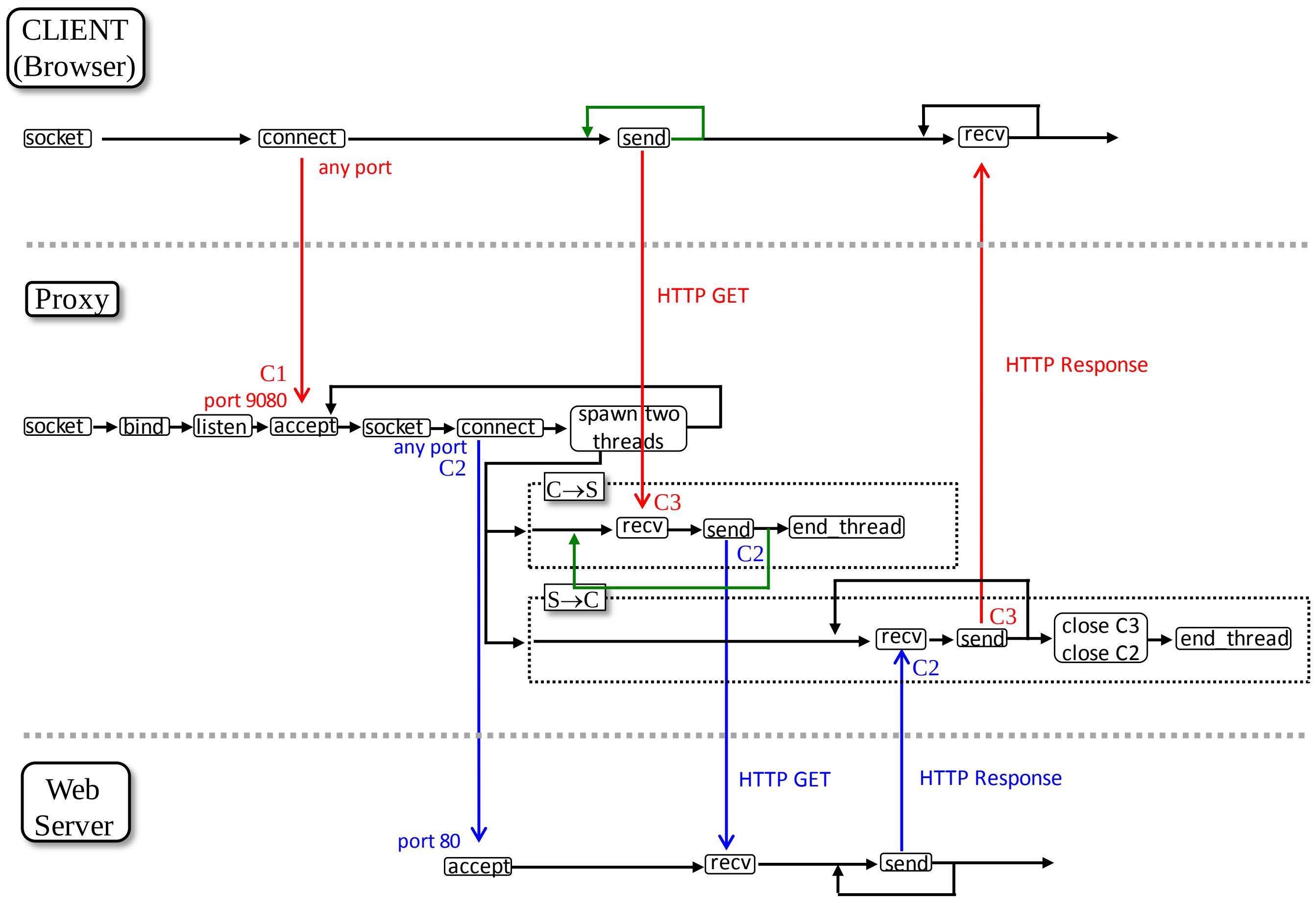

**Fig. 2 – Suggested proxy organization**

## 4. Expectations when a question is asked:

Dr. Fujinoki is willing to assist you for your successful completion of this project and if you have any conceptual and/or technical issues, you are encouraged to ask your questions to Dr. Fujinoki. However, to maximize your learning, each of you is expected to do the following (i.e., Dr. Fujinoki will welcome you when you are asking questions, if you do the followings):

(1) Identify where (in your source code) the problem exists

(2) Describe the symptom(s) of the problem

(3) Describe how the problem happens (always happen, sometime happen, the condition(s) for the problem to happen, etc.)

(4) Describe what you tried (to understand and/or solve the problem)

(5) Stop by Dr. Fujinoki's office (no question through e-mail).

(6) Bring your source code (either hard copy or soft copy)

## 5. Grading Criteria (as a guideline – some deviations may be possible):

If your web proxy is capable of:

(1) Handling a web page that does not contain any other files using "a single-thread web proxy design"): 50/100

(2) Handling a web page that contains more than one file download, using "a single-thread web proxy design"), but not all content files are successfully transferred (some multiple files are successfully transferred): 65/100

(3) Handling a web page that contains more than one file download, using "a multi-thread web proxy design"), but not all content files are successfully transferred (some multiple files are successfully transferred): 80/100

(4) Handling a web page that contains more than one file download, using "a multi-thread web proxy design"), and all content files are successfully transferred: 90/100

(5) Successfully performing "fire wall features" on top of (4) above: 100/100

- Serious violations of the course academic dishonesty policies will result in a failing grade.

## 6. Submissions

Please e-mail your cpp file (or any other necessary files, such as *.h file) to Dr. Fujinoki for the earliest possible deadline below:

(a) Extra-credit internal deadline*[3]: September 26th (Tuesday, 11:11:59 PM) (+4 points to your midterm exam)

(b) Free feedback internal deadline*[3]: September 28th (Thursday, 11:11:59 PM)

(c) Hard deadline: October 3rd (Tuesday, 11:00:00 AM)

*3: No "late submission" for early deadlines. The extra credit will be given only if all the three programs are completed by the extra-credit early submission deadline. No "partial extra credit" will be given for this extra credit opportunity.

## 7. Rules to avoid academic dishonesty

(a) Sharing source code file(s) is considered academic dishonesty

(b) Exchanging ideas through oral communications is <u>not</u> allowed (this project is completely an individual project).

(c) Regarding samples you find anywhere except in the references (the ones Dr. Fujinoki introduced in the first programming assignment), only "syntax-level reference" (which should not be more than three C++

statements) is acceptable. For any case that exceeds three statements, please consult to Dr. Fujinoki instead of making your own assumptions.

**Note:** If you have anything you are not sure, please do not make your own assumptions. If you are making your own assumptions for anything you are not sure, please be responsible for your own assumptions (Dr. Fujinoki will not be responsible for your assumptions). Remember: asking a question is free (no matter what is your question) and once you got an answer from Dr. Fujinoki, Dr. Fujinoki will be responsible for whatever answer he gave to you.

## 8. Other issues

- For those who have technical problems to finish this project, please consult to Dr. Fujinoki as soon as possible.