Web Proxy: “Net Ninny”

By: Hannah Börjesson (hanbo174), Per Jonsson (perjo927)

Manual

Assignment 2

TDTS04 Computer Networks and Distributed Systems

Linköping university Feb 2013

Compile

We have written some parts of the code using C++11 standard, so to compile you will need a version of the GCC compiler which is at least 4.7.x.

To compile, first unpack the -gzipped tar file to a folder of interest, and then please type the following command in a terminal window:
g++ -std=c++11 proxy.cc -o proxy

The application was developed in an Ubuntu 12.x environment and works perfectly there.

Configuration and usage

Let's say for instance that you are using the web browser Mozilla Firefox; to enable the “Net Ninny”, please visit the menu bar and find the tab **Edit** where you then will select **Preferences**, and then choose **Advanced → Network → Settings**, and finally choose **Manual Proxy Configuration**. In the field **HTTP proxy** please type: *localhost*, and in the field **port**: *8085* (for example) and hit **OK**.

You might also want to change which words to enable for filtering content. If so, you should then go to line 146 in the source code, and change, remove or add a string to the vector named *bad_content:*

```
vector<string> bad_content = {"britney spears", "norrköping", "paris hilton", "spongebob"};
```

Run

To run the program, just fire up a terminal window, visit the folder where you unpacked the “Net Ninny”, type: *./proxy* and as our proxy allows the user to select the proxy port, just follow up with the one you have chosen (in this case: *./proxy 8085*).

Features

The Web Proxy handles simple HTTP GET interactions between client and server. In the main() function we handle file descriptors, **bind**, **listen** and **accept** methods between client and proxy, and in **conversation** function we handle send and receive between proxy and server. We use **fork** to create child processes.

From the main function we call the **forbidden_words** function, which returns true or false depending on the URL – if it contains forbidden words or not. If the URL contains a forbidden word the function's return value is set to true (1) and the main function redirects the client to 302 (Bad URL) and closes client and servers sockets. If no forbidden words are found in the URL the return value is set to false (0) and main calls **create_http_hdr** which changes the

HTTP header to request a closed connection and returns it back to main. Main then calls the **conversation** function which prepares to launch to server, send and receive content from a page. In the function **hdr_to_map** we map the header information to make it easy to find, for instance content-type. If the content-type is non-text, we send it directly to the client, and if the content-type is text (see line 274:

```
vector<string> content_types = {"html", "xml", "text"};
```

which is a vector with strings with room for more types of text if we wish to include that), we call function **forbidden_words** to check if the content is OK or not. If forbidden_words return value is false the conversation function sends the content to the client, and if its true the client will be redirected to a 302 (Bad content).

•