Linköping University

Department of Computer and Information Science

TDTS06 Computer Networks

Net Ninny: A Web Proxy Based Service

Report

Group C/D 13

Chvátal	Martin	march011@student.liu.se
Peschke	Lena	lenpe782@student.liu.se

Teaching assistant :

Schmidt Johannes   johannes.schmidt@liu.se

September 27, 2014

1 Introduction

In order to learn more about the World Wide Web and HTTP and to lear about TCP/IP and socket programming, we designed and implemented a simple web proxy. This software allows to filter the interaction between his browser and the Web. We decided to write the code in `C++`.

2 How it works

The proxy consists of a number of files, which can be compiled and linked with the provided `Makefile` and the command `make`. To run the program, simply type `./net_ninny PORT` in the terminal, where `PORT` is the proxy port used with the browser.

The proxy is mainly divided into two parts: a server side and a client side. The server gets the request from the browser through the port specified by the user (port x) and checks the url for bad content. If there is some, it sends back a `302 Found` error with the url of the error page to be displayed to port x. If there is no bad content, the proxy server makes sure to use a non persistent connection (`Connection: close`) and forwards the request to the client, who sends it out to the web through port 80. The client side then gets a response through the same port. If the response contains text, it checks for bad words. If it finds some, it sends an internal message to the server side of the proxy, who then sends a `302 Found` to the browser. If not, or if the content is not text, the server simply forwards the response, which is then transmitted to the browser (via port x).

3 Testing

3.1 Several cases

A simple text file

A simple HTML file

An HTML file with a bad name

An HTML file with a good name but bad content

A valid Google request

Aftonbladet

A YouTube video

A tour through Wikipedia

3.2 Available services

The implemented proxy supports the required features.

1. *The proxy supports both HTTP/1.0 and HTTP/1.1.*
2. *It handles simple HTTP GET interactions between client and server.*
   All messages are transformed into `CTCPBuffer` objects, and the modified and checked as `CHTTPRequest` and `CHTTPResponse` objects.
3. *It blocks requests for undesirable URLs, using HTTP redirection to display an error page instead.*
   This is done by the client when he reads the url in file `server.cpp` on line 45.
4. *It detects inappropriate content bytes within a Web page before it is returned to the user, and redirects to another error page.*
   The server and client work together on this. Lines 107 in `client.cpp` and 45 in file `server.cpp` handle the case.
5. *It imposes no limit on the size of the transferred HTTP data.*
   The proxy always processes the whole transferred data. The proxy receives data by calling `recv` incrementally.
6. *It is compatible with all major browsers (e.g. Internet Explorer, Mozilla Firefox, Google Chrome, etc.) without the requirement to tweak any advanced feature.*
   For our tests, we have used Mozilla Firefox and Google Chrome. In the code, all headers are converted to lower case when we have to parse them, so that the different cases used by the browsers do not cause a problem.
7. *It allows the user to select the proxy port.*
   The proxy port is the only argument the user has to provide to the program. File `main.cpp` line 47.
8. *It is smart in selection of what HTTP content should be searched for the forbidden keywords.*
   The proxy only searches text content. File `client.cpp` line 107.

Capabilities and limitations The proxy server is functional for a basic use. As seen in the tests above,

4 Possible improvements