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 28, 2014

1 Introduction

In order to learn more about the World Wide Web and HTTP and to learn about TCP/IP and socket programming, we designed and implemented a simple web proxy. This software allows one 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 x specified by the user 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 Basic tests

The following HTTP messages display the working of the proxy with simple web pages.

Listing 1: A simple text file.
The GET message as modified by the proxy and the server's response.

```
GET http://www.ida.liu.se/~TDTS04/labs/2011/ass2/goodtest1.txt HTTP/1.1
Host: www.ida.liu.se
Proxy-Connection: close
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/537.36 (
    KHTML, like Gecko) Chrome/37.0.2062.124 Safari/537.36
Accept-Encoding: gzip,deflate,sdch
Accept-Language: fr-FR,fr;q=0.8,en-US;q=0.6,en;q=0.4

HTTP/1.1 200 OK
Date: Sat, 27 Sep 2014 16:01:28 GMT
Server: Apache/2.2.24 (Unix) DAV/2 SVN/1.6.17 PHP/5.3.23 mod_fastcgi/2.4.6
    mod_auth_kerb/5.4+ida mod_jk/1.2.31 mod_ssl/2.2.24 OpenSSL/0.9.7d
Last-Modified: Mon, 21 Jan 2008 07:14:16 GMT
ETag: "4c3fae-77-444363d682200"
Accept-Ranges: bytes
Content-Length: 119
Content-Type: text/plain
```

Listing 2: A simple HTML file.
The GET message as modified by the proxy and the server's response.

```
GET http://www.ida.liu.se/~TDTS04/labs/2011/ass2/goodtest2.html HTTP/1.1
Host: www.ida.liu.se
Proxy-Connection: close
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/537.36 (
    KHTML, like Gecko) Chrome/37.0.2062.124 Safari/537.36
Accept-Encoding: gzip,deflate,sdch
Accept-Language: fr-FR,fr;q=0.8,en-US;q=0.6,en;q=0.4

HTTP/1.1 200 OK
Date: Sat, 27 Sep 2014 16:01:55 GMT
Server: Apache/2.2.24 (Unix) DAV/2 SVN/1.6.17 PHP/5.3.23 mod_fastcgi/2.4.6
    mod_auth_kerb/5.4+ida mod_jk/1.2.31 mod_ssl/2.2.24 OpenSSL/0.9.7d
Last-Modified: Mon, 21 Jan 2008 07:18:17 GMT
ETag: "4c3faf-ea-444364bc58040"
Accept-Ranges: bytes
Content-Length: 234
Content-Type: text/html
```

Listing 3: An HTML file with a bad name.
The GET message from the browser, the response from the server, the proxy's response after checking the content and the redirection.

```
GET http://www.ida.liu.se/~TDTS04/labs/2011/ass2/SpongeBob.html HTTP/1.1
Host: www.ida.liu.se
Proxy-Connection: keep-alive
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/537.36 (
    KHTML, like Gecko) Chrome/37.0.2062.124 Safari/537.36
Accept-Encoding: gzip,deflate,sdch
Accept-Language: fr-FR,fr;q=0.8,en-US;q=0.6,en;q=0.4

HTTP/1.1 302 Found
Location: http://www.ida.liu.se/~TDTS04/labs/2011/ass2/error2.html
Connection: close

GET http://www.ida.liu.se/~TDTS04/labs/2011/ass2/error1.html HTTP/1.1
Host: www.ida.liu.se
Proxy-Connection: close
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/537.36 (
    KHTML, like Gecko) Chrome/37.0.2062.124 Safari/537.36
Accept-Encoding: gzip,deflate,sdch
Accept-Language: fr-FR,fr;q=0.8,en-US;q=0.6,en;q=0.4

HTTP/1.1 200 OK
Date: Sat, 27 Sep 2014 16:02:14 GMT
Server: Apache/2.2.24 (Unix) DAV/2 SVN/1.6.17 PHP/5.3.23 mod_fastcgi/2.4.6
    mod_auth_kerb/5.4+ida mod_jk/1.2.31 mod_ssl/2.2.24 OpenSSL/0.9.7d
Last-Modified: Wed, 06 May 2009 18:12:35 GMT
ETag: "4c4174-126-4694256fcaac0"
Accept-Ranges: bytes
Content-Length: 294
Content-Type: text/html
```

Listing 4: An HTML file with a good name but bad content.
The GET message as modified by the proxy, the server's response and the redirection.

```
GET http://www.ida.liu.se/~TDTS04/labs/2011/ass2/badtest1.html HTTP/1.1
Host: www.ida.liu.se
Proxy-Connection: keep-alive
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
```

```
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/537.36 (
    KHTML, like Gecko) Chrome/37.0.2062.124 Safari/537.36
Accept-Encoding: gzip,deflate,sdch
Accept-Language: fr-FR,fr;q=0.8,en-US;q=0.6,en;q=0.4

HTTP/1.1 200 OK
Date: Sun, 28 Sep 2014 20:24:04 GMT
Server: Apache/2.2.24 (Unix) DAV/2 SVN/1.6.17 PHP/5.3.23 mod_fastcgi/2.4.6
    mod_auth_kerb/5.4+ida mod_jk/1.2.31 mod_ssl/2.2.24 OpenSSL/0.9.7d
Last-Modified: Mon, 21 Jan 2008 07:27:50 GMT
ETag: "4c4173-1e4-444366decc980"
Accept-Ranges: bytes
Content-Length: 484
Content-Type: text/html

HTTP/1.1 302 Found
Location: http://www.ida.liu.se/~TDTS04/labs/2011/ass2/error2.html
Connection: close

GET http://www.ida.liu.se/~TDTS04/labs/2011/ass2/error2.html HTTP/1.1
Host: www.ida.liu.se
Proxy-Connection: keep-alive
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_5) AppleWebKit/537.36 (
    KHTML, like Gecko) Chrome/37.0.2062.124 Safari/537.36
Accept-Encoding: gzip,deflate,sdch
Accept-Language: fr-FR,fr;q=0.8,en-US;q=0.6,en;q=0.4

HTTP/1.1 200 OK
Date: Sun, 28 Sep 2014 20:24:04 GMT
Server: Apache/2.2.24 (Unix) DAV/2 SVN/1.6.17 PHP/5.3.23 mod_fastcgi/2.4.6
    mod_auth_kerb/5.4+ida mod_jk/1.2.31 mod_ssl/2.2.24 OpenSSL/0.9.7d
Last-Modified: Mon, 21 Jan 2008 07:21:47 GMT
ETag: "4c4175-13c-444365849d8c0"
Accept-Ranges: bytes
Content-Length: 316
Content-Type: text/html
```

3.2 More advanced tests

We have successfully tested the proxy with several web pages, among which the following: `http://www.aftonbladet.se/`, `http://www.google.se/?gws_rd=ssl`, `http://www.yahoo.com/`. Google requests were also handled.
`http://www.wikipedia.org/` worked partially: the content was displayed, but the layout was not preserved. `http://www.youtube.com/` did not work.
In general the proxy was not able to handle HTTPS. When using Chrome, it did sometimes switch to HTTPS automatically.

4 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 42.

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 104 in `client.cpp` and 57 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 104.

Capabilities and limitations The proxy server is fully functional for a basic use. As seen in the tests above, it manages to filter url and text content. Because the file containing the bad words is not hard coded into the program, it is also very modular.
We have, however, noticed a series of limitations. First of all, special characters in the bad words, such as the “ö” in “Norrköping”, are not always recognized, and the word is therefor not always filtered. This has to do with the different types of text encoding. Second, there are websites, such as YouTube, which could simply not be displayed at all. Some others (Wikipedia) did not keep their original layout.