LINKÖPINGS UNIVERSITET

TDTS04 Computer Networks and Distributed Systems

Assignment 2: Net Ninny

Author

Christer Vesterlund, chrve180@student.liu.se
Denis I. Blazevic denbl369@student.liu.se

INSTITUTIONEN FÖR DATAVETENSKAP

2017-02-05

Innehåll

1 User manual **2**
1.1 Set-up and installation . . . 2
1.1.1 Which Ruby version? . . . 2
1.1.2 Starting the proxy . . . 2
1.1.3 Configuring Firefox . . . 2
1.2 Features . . . 3

2 Proxy testing **3**

3 Summary **4**

1 User manual

1.1 Set-up and installation

1.1.1 Which Ruby version?

Our proxy is implemented in Ruby 2.3 but runs in any ruby version above 1.9. To ensure that you have the right version installed, start up your terminal and type:

```
$ ruby -v
ruby 2.3.1p112 (2016-04-26) [x86_64-linux-gnu]
```

If your version is at least over 1.9, then you can use the proxy.

1.1.2 Starting the proxy

To start the proxy you will need to open up the folder where the source files are located in a terminal emulator. To start the proxy on its default port (2000), run the following command:

```
$ ruby proxy.rb
```

This will start the proxy on the default port (2000).
To specify a port for the proxy to run on, simply run the following command where you replace [port] for the port you want:

```
$ ruby proxy.rb [port]
```

1.1.3 Configuring Firefox

To configure Firefox to use the proxy, make sure to do the following steps.

1. Open Firefox **“Preferences”**.
2. On the left side of your browser, click on **“Advanced”**.
3. Click on **“Network”**.
4. Press on **“Settings...”** right next to the **“Connection”**.
5. Make sure **“Manual proxy configuration”** is **ticked**.
6. In the text field **“HTTP Proxy”**, type **“127.0.0.1”**
7. In the **“Port”** field, input the port the proxy was started with (default port is 2000 for the proxy).
8. Make sure **“Use this proxy for all protocols”** is **unticked**.

You can now enjoy the security of the proxy.

1.2 Features

Our goal for this assignment was to fulfill the requirements. There were a total of nine requirements, and one which was optional.

1. The proxy should support both HTTP/1.0 and HTTP/1.1.
2. Handles simple HTTP GET interactions between client and server.
3. Blocks requests for undesirable URLs, using HTTP redirection to display error page instead.
4. Detects inappropriate content bytes within a Web page before it is returned to the user, and redirecting to error page.
5. Imposes no limit on the size of the transferred HTTP data.
6. Is compatible with all major browsers, without the requirement to tweak any advanced feature
7. Allows the user to select the proxy port at startup (i.e. the port number should not be hard coded).
8. Is smart in selection of what HTTP content should be searched for the forbidden keywords.
9. (Optional) Supporting file upload using the POST method.

Number 6 in the requirements has not fully been tested. Chromium and Firefox has been tested, but not Internet Explorer/Edge.
Number 9 has not been implemented and therefore not tested. Our proxy only works with **"GET"** requests.

2 Proxy testing

The Net Ninny Ruby proxy was being tested continuously while developed by us. At first we made a simple socket server and socket client in C++ to learn how sockets works. The C++ proxy was more learn by doingbecause we couldn't find any "good" enough information on how to program sockets in C++. While the C++ proxy was developed we used the "console logdebug method, outputting any information to the console while testing. This gave us information on how the HTTP headers are constructed (which the RFC confirmed was correct) and how the servers communicate at connection, transfer and disconnection.
After the C++ simple server and client was finished, we moved on to Ruby for the actual proxy we wanted to get more knowledge of Ruby. This is where we started to test the RegEX capture strings for various of HTTP headers and URLs. When we found working capture strings for every text data we needed to search through, the actual work on Client-Proxy-Server started. Because we had the C++ version we easily rewrote the code into Ruby.

We managed to get video streaming somewhat working, Dailymotion works for standard quality but not for HD quality. By "somewhat working", we mean some video streaming web pages doesn't work and we can not find the issue e.g Aftonbladet TV is not working but Dailymotion is.

3 Summary

The Net Ninny Ruby proxy is an HTTP proxy that works with HTTP 1.0 aswell as HTTP 1.1, but the proxy only works with regular HTTP "GET" requests. We did not implement HTTPS or "POST" requests. The main reasons for this is that the majority of the http request sent by the browser is a "GET" request and very few are in fact "POST" requests.

The proxy works by first accepting connections from the browser used (in our case Firefox or Chromium) into a new socket connection that is threaded off into a separate thread. This makes it possible to have multiple connections working at the same time. The Net Ninny is also capable of filtering url's and page contents for inappropriate words.

One thing that we have been working on and tried to resolve is streaming video at "www.aftonbladet.se". We have tried many types of solutions, but no one have solved our problem. The strange thing is that streamning works at "www.dalymotion.com" (it's very slow, but it works). The biggest question mark we have is why streamning works on one site but not the other?! This remains to be solved.

Overall this proxy is very limited and lacks streaming support at "www.aftonbladet.se", but this have been a very fun assignment and a big learning experience especially the socket part of the assignment.