PROTOCOLES ET PROGRAMMATION RÉSEAUX

Anh-Kiet DUONG

December 21, 2022

Contents

1 Introduction

1.1 General

My name: Anh-Kiet DUONG.
Program: Master (M1) CRYPTIS.
Demo: demo.mp4
Source code: github

1.2 Proxy

A "Proxy" server is a "mandatory" process, which will serve as an intermediary for communication between the Web client (the browser) and the Web server. This communication is organized according to the HTTP protocol, based on TCP.

The role of the proxy is to:

- Filter communications by prohibiting access to sites based on their content (following a list of keywords).
- Serve as a "cache", ie to memorize the content of the most frequently visited pages and thus avoid having to retrieve them again from the server that hosts them.
- Modify the data transmitted (implementation of censorship for parental protection, for example).
- Ensure better security. The client machines do not communicate directly with the servers located outside the local network, but only with the proxy (one can also prohibit any direct transaction, without going through the proxy, from a client to a server outside the using a firewall).

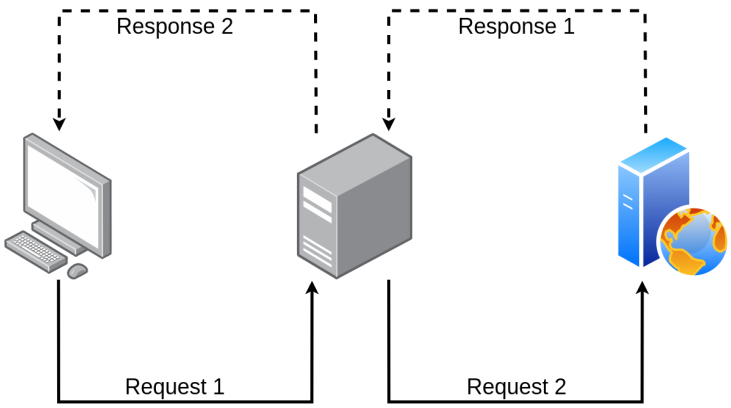


Figure 1: Proxy

2 Proxy for the HTTP

2.1 Performs filtering

We call the data that the browser receives as input and the data that the browser sends output. The data will include two parts, one is the header and the other is the content.

Config website interface (by default config URL: http://myproxy.com:1234/)

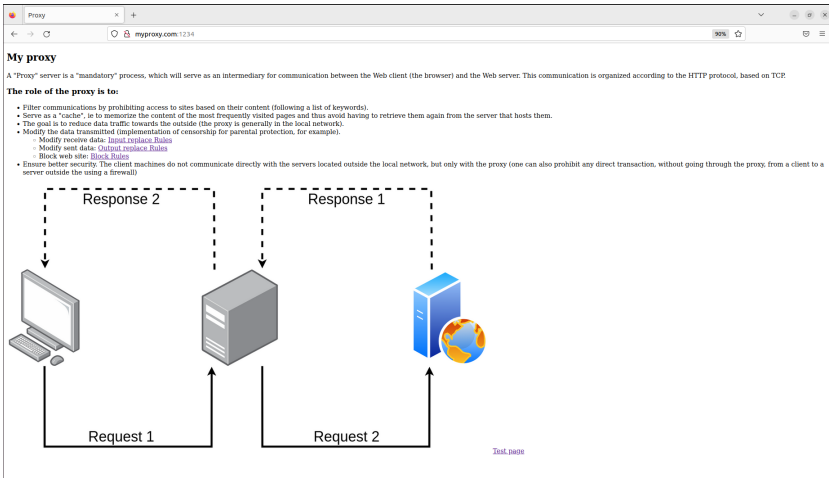


Figure 2: Config website (index page)

2.1.1 Modify header

By default, the rules will be saved in the file **"headerRules.txt"**. An example of "headerRules.txt":

```
Connection: Keep-Alive
Proxy-Connection: Keep-Alive
Accept-Encoding: gzip
```

The purpose is to filter in the header to remove lines starting with the lines listed in the file.

In the above example, we need to delete the lines that start with *Connection: Keep-Alive, Proxy-Connection: Keep-Alive* the purpose is so that when the web server finishes sending data, it will shut down the connection. Normally when using a "socket" the receiver side will receive each buffer and wait for the next buffer. If the web server does not close the connection, the proxy does not know if there is data or not and continues to wait. To overcome this situation, we can remove the above lines or set the socket timeout. And *Accept-Encoding: gzip* is for data compression.

2.1.2 Replacing output text

By default, the rules will be saved in the file **"outReplaceRules.txt"**. An example of "outReplaceRules.txt":

```
"hihi": "hehe",
"hehe": "haha"
```

The purpose is to filter (replace) the data sent by the browser. As in the example above, "hihi" will be changed to "hehe", and then "hehe" will be changed to "haha" (including "hihi": "hehe"). Replace is used by regex, so we can add advanced rules like text containing keywords, ending with,...

When users want to add, remove or change these rules, they can access the link http://myproxy.com:1234/outReplaceRules.html.

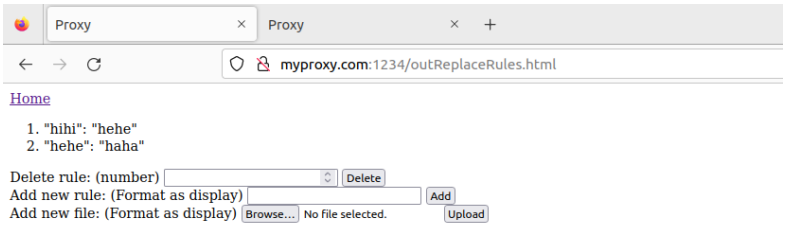
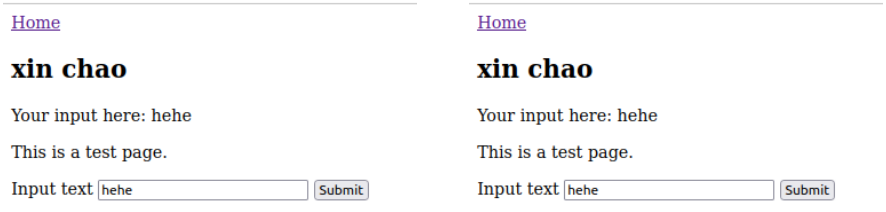


Figure 3: Output Replacing Rules Modify

Illustrate an example before and after adding the rule to the filter. I have developed a website to test the features http://myproxy.com:1234/test.html. A simple web page that shows what users submit. And here are the before and after pictures with the filter, with the same "hehe". Watch the demo video to see more clearly.



(a) Test without filter

(b) Test with filter on

Figure 4: Illustrate an example before and after adding the rule filtering output

2.1.3 Replacing input text

By default, the rules will be saved in the file **"inReplaceRules.txt"**. An example of "inReplaceRules.txt":

```
"hello":"bonjour",
"bonjour":"xin chao"
```

The purpose is to filter (replace) the data sent by the web server. As in the example above, "hello" will be changed to "bonjour", and then "bonjour" will be changed to "xin chao" (including "hello": "bonjour"). Replace is used by regex, so we can add advanced rules like text containing keywords, ending with,...

When users want to add, remove or change these rules, they can access the link http://myproxy.com:1234/inReplaceRules.html.

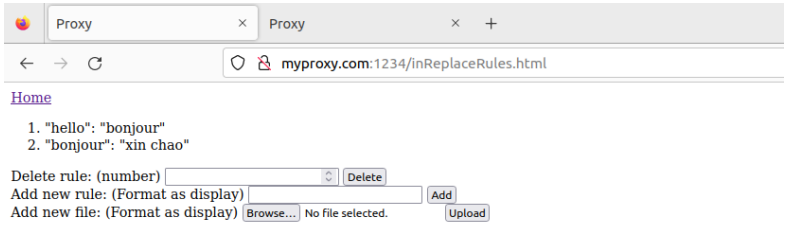
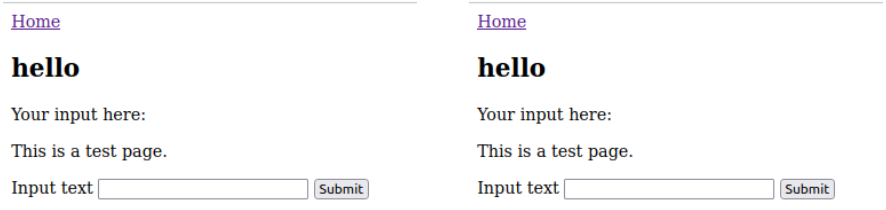


Figure 5: Input Replacing Rules Modify

Illustrate an example before and after adding the rule to the filter. I have developed a website to test the features http://myproxy.com:1234/test.html. A simple web page that shows "hello". And here are the before and after pictures with the filter, with the same page. Watch the demo video to see more clearly.



(a) Test without filter (b) Test with filter on

Figure 6: Illustrate an example before and after adding the rule filtering input

2.1.4 Block a website

By default, the rules will be saved in the file **"blockRules.txt"**. An example of "blockRules.txt":

```
http://p-fb.net/master1/proto_prog/
.png$
```

The purpose is to block the website to match the listed format. As in the example above, http://p-fb.net/master1/proto_prog/ will be blocked. And any URL ending with ".png" will also be blocked. Replace is used by regex, so we can add advanced rules.

When users want to add, remove or change these rules, they can access the link http://myproxy.com:1234/blockRules.html.

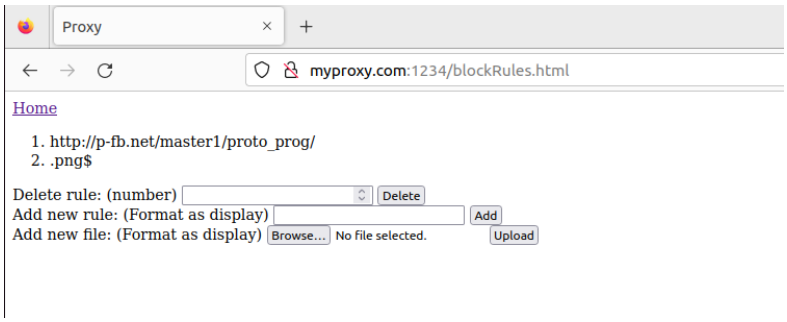


Figure 7: Block a website Rules Modify

And result:

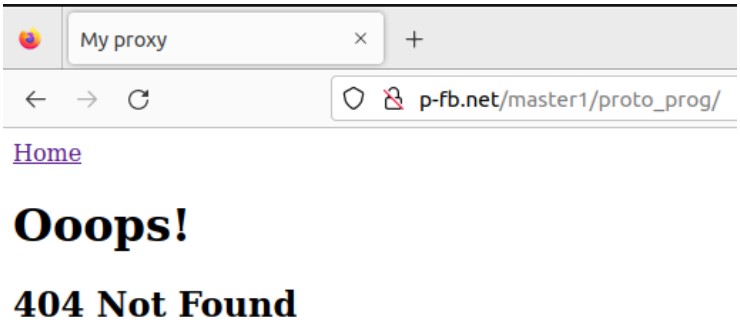
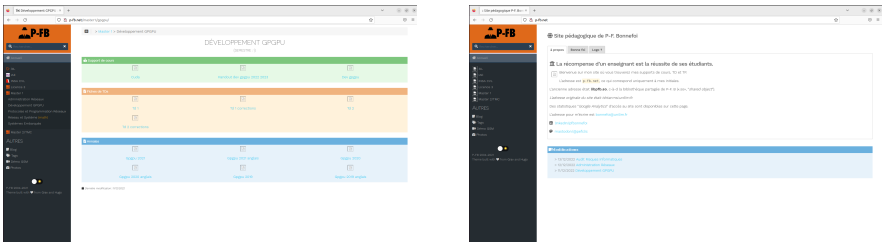


Figure 8: Try accessing the path http://p-fb.net/master1/proto_prog/ and the result



(a) GPU cached some images (b) CRTL + F5 on http://p-fb.net/

Figure 9: Block any URL ending with ".png"

2.2 Manages the request

We use Burp Suite [1] software to capture packets sent by the browser. For an overview of GET and POST requests along with other types like HEAD,...

2.2.1 GET

A GET request would look like this. GET command: the most common way to request a resource from a web server. **My proxy app works with requests like this.**

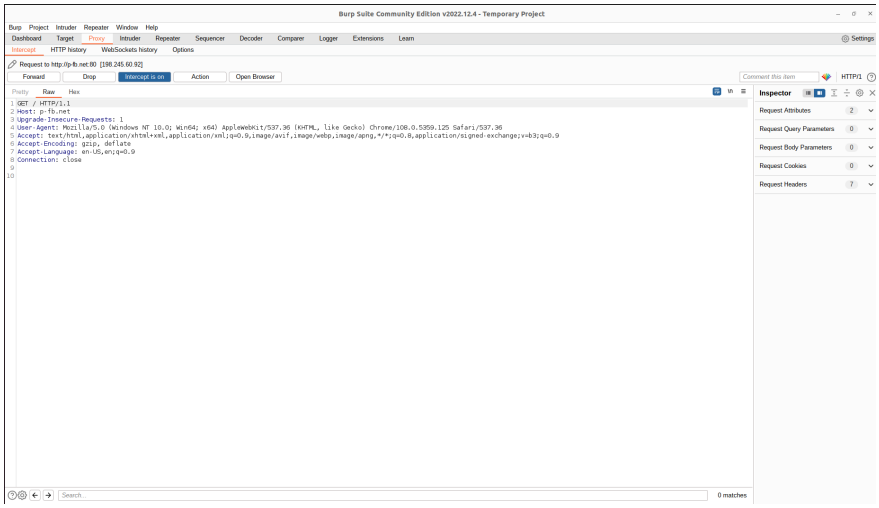


Figure 10: GET request

2.2.2 POST

A POST request would look like this. **My proxy app works with requests like this.**

POST command: it allows to transmit to the Web server, in addition to the request for access to a resource, data in MIME format:

- Transmitted by the browser following its “POST” command after an empty line.
- Accompanied by a size, which allows them to be retrieved, indicated by ”Content-Length: length_in_bytes”.

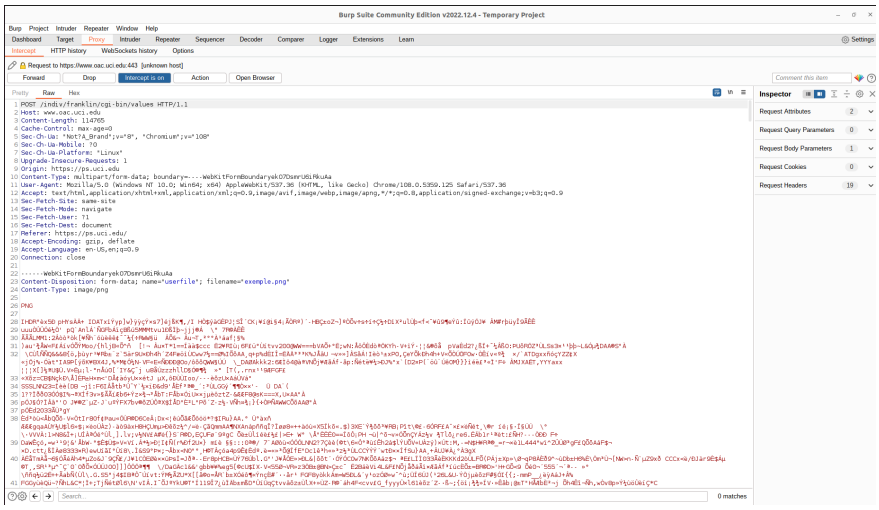


Figure 11: POST request

In the figure 11 is a POST command accompanied by a PNG file.

2.2.3 HTTP/1.1

Based on work Krishnamurthy et al[2]. HTTP/1.1 [3] requires requests to include a **Host** header, that carries the hostname. Because of reasons like the proliferation of IP address allocations,... **My proxy app works with requests like this.**

```
GET http://p-fb.net/ HTTP/1.0
Host: p-fb.net
```

2.2.4 HTTP/1.0

Based on work Krishnamurthy et al[2]. There is no **Host** in HTTP/1.0 [4]. **My proxy app works with requests like this.** By getting the host in the URL.

```
GET http://p-fb.net/ HTTP/1.0
```

References

[1] A. Mahajan, *Burp Suite Essentials*. Packt Publishing Ltd, 2014.

[2] B. Krishnamurthy, J. C. Mogul, and D. M. Kristol, "Key differences between http/1.0 and http/1.1," *Computer Networks*, vol. 31, no. 11-16, pp. 1737–1751, 1999.

[3] H. T. Protocol, "Http/1.1," 1999.

[4] T. Berners-Lee, R. Fielding, and H. Frystyk, "Hypertext transfer protocol–http/1.0," tech. rep., 1996.