Master 1 Cryptis - Computer Science
Faculty of Science and Technology

Network Protocols and Programming

TP Project - Proxy Server

Authors :
Minh Luan NGUYEN
Thi Ha Trang NGUYEN

Academic year 2023 - 2024

Contents

1 Introduction

1.1 Project's source code

The work is done with the source code

1.2 General

This report contains analysis, understanding, and implementation of the TP Project *Proxy Server*. A proxy server functions as a middleman connecting the user (web browser) and the Internet (servers).

Figure 1: Role of a proxy server in Web browser-server connection

The roles of the proxy server:

- Enhanced privacy and security: By going through the proxy, the client can avoid threads from malicious content.
- Caching: By caching frequently accessed resources locally, proxy servers accelerate subsequent access to those resources.
- Content filtering and access control: Proxy servers can filter content based on predefined rules, replacing or redacting inappropriate content, and blocking access to specific websites or types of content.

2 Implementation

2.1 Setup

The Proxy Server uses two ports, one for handling the proxy (default 1234) and one for the configuration web interface (default 8080).

To run the Proxy Server, execute the *python* file with:

```
python ./ProxyServer.py
```

The web interface is hosted on port 8080, so we can access it through *localhost:8080*

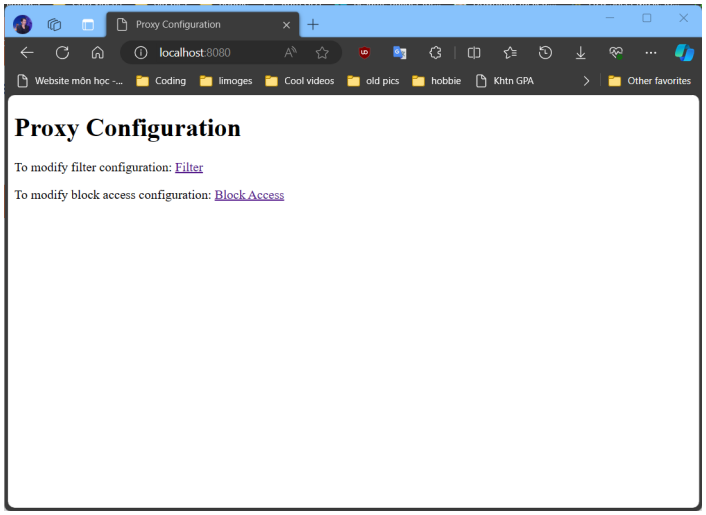

Figure 2: Firefox setting to use proxy

In order to test the Proxy Server, we can set our web browser to use the proxy. In this example, we will you Mozilla Firefox. Under *Seetings*, find *Network Settings* and set the configuration using port 1234 as follow:

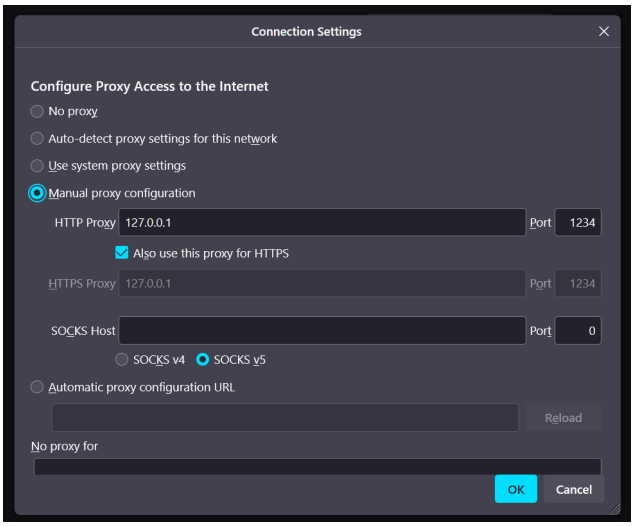

Figure 3: Firefox setting to use proxy

In the process of testing, the browser might save the cache to help load the web faster, so some of the changes we make will not be shown. To resolve this, we can use *Ctrl + F5* or delete the cache frequently.

2.2 Block access

2.2.1 Modify the list of blocked access

We create a file named *blockAccessRules.txt* which contains a list of URLs that will be blocked when running through Proxy server:

```
http://p-fb.net/master1/proto_prog
http://p-fb.net/master1/proto_prog//cours/cours_python.pdf
```

We also allow user to add or delete blocked access according to their requirement.

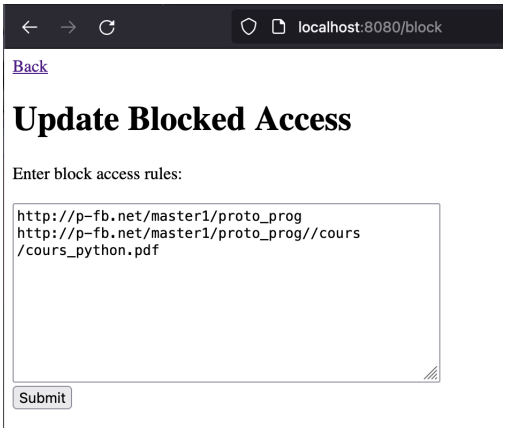

Figure 4: Interface to up date block access rule

All the modifications will be saved after clicking the *Submit* button.

2.2.2 Block an access

When a client sends a request to a URL that is listed in *blockAccessRules.txt* file, it will go directly to an HTML page that notifies the client that the URL is blocked.

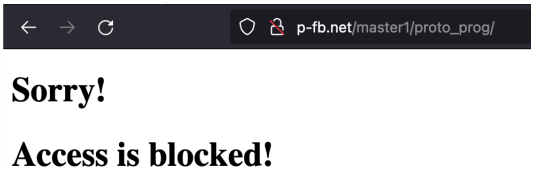

Figure 5: Access is blocked

In *Figure 3*, we tried to access *http://p-fb.net/master1/proto_prog*.

2.3 Modify header

Default fields to replace:

```
'HTTP/1.1' with 'HTTP/1.0'
'Connection: keep-alive'
'Proxy-Connection: keep-alive'
'Accept-Encoding: gzip'
```

This function essentially sanitizes the header, removing specific fields and altering the HTTP version, ensuring the transmitted header adheres to defined criteria or preferences before reaching the server.

Since the client is communicating through the proxy, we remove this field so that the connection is shut down once the server finishes sending data. In this way, the proxy will know if the server is done sending data or not.

Accept-Encoding: gzip is for data compression.

2.4 Filtering

2.4.1 (De)activate filter

We can toggle the filter on or off by changing the configuration in the filter configuration interface.

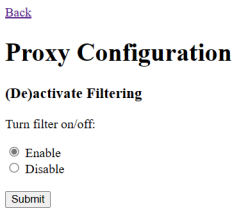

Figure 6: Configure filter rule

2.4.2 Default filters

When the filter is on, the proxy server will first modify the *title* of the website, changing it to: *Surfing with Proxy Server* to help indicate that the user is currently using the proxy to serve the web.

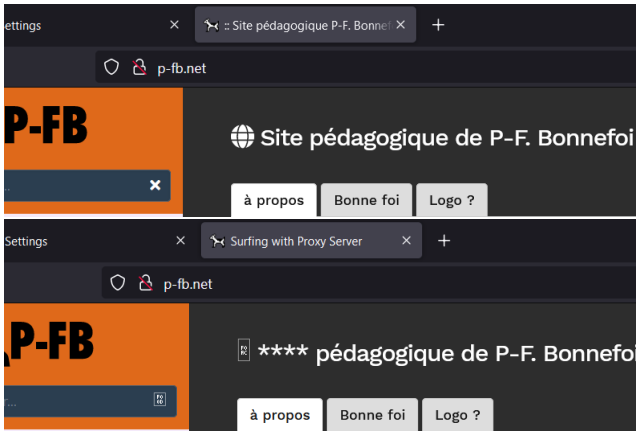

Figure 7: Website title before and after filter

Another rule of this proxy server is to prohibit content that includes *.mp4* format.

```python
#Filter data before send to client
def filterData(self, data):
    ...
    # Modify title
    body = re.sub(b'<title>.*?</title>', b'<title>Surfing with Proxy Server</title>', body)

    # Delete all mp4 resources
    body = re.sub(rb'<source.*?type="video/mp4".*?>', b'', body)
    ...
```

2.4.3 Replace text

The proxy server has the feature of replacing text that is served by the web servers.

The replacement rules are stored in the *replaceRules.txt* file with the format:

$$< old\_word >:< new\_word >$$

```
2023:2024
War:Special operation
hello:bonjour
```

```python
#Filter data before send to client
def filterData(self, data):
    ...
    # Replace word in giving list
    for rule in self.replaceRules:
        key = rule.split(b':')[0]
        val = rule.split(b':')[1]
        body = re.sub(rb"(?<!<[^>])(%s)(?![^<]*>)" % key, val, body, flags=re.IGNORECASE)
    ...
```

2.4.4 Censored text

The proxy server also provides us with the censor rules the redact inappropriate words that might not be appropriate to use. These rules are stored in the *censorRules.txt* and each line contains one word that needs to be censored.

```
hell
loser
retard
lmao
lmfao
...
```

It is also worth noticing that we need to avoid replacing and censoring the HTML tag and keywords.

```
#Filter data before send to client
def filterData(self, data):
    ...
    # Censor forhibited words, ignore the html tags too ...
    for word in self.censorRules:
        body = re.sub(rb"(?<![^>])(%s)(?![^<]*>)" % word, b'*' * len(word), body,
        ↪  flags=re.IGNORECASE)
    ...
```

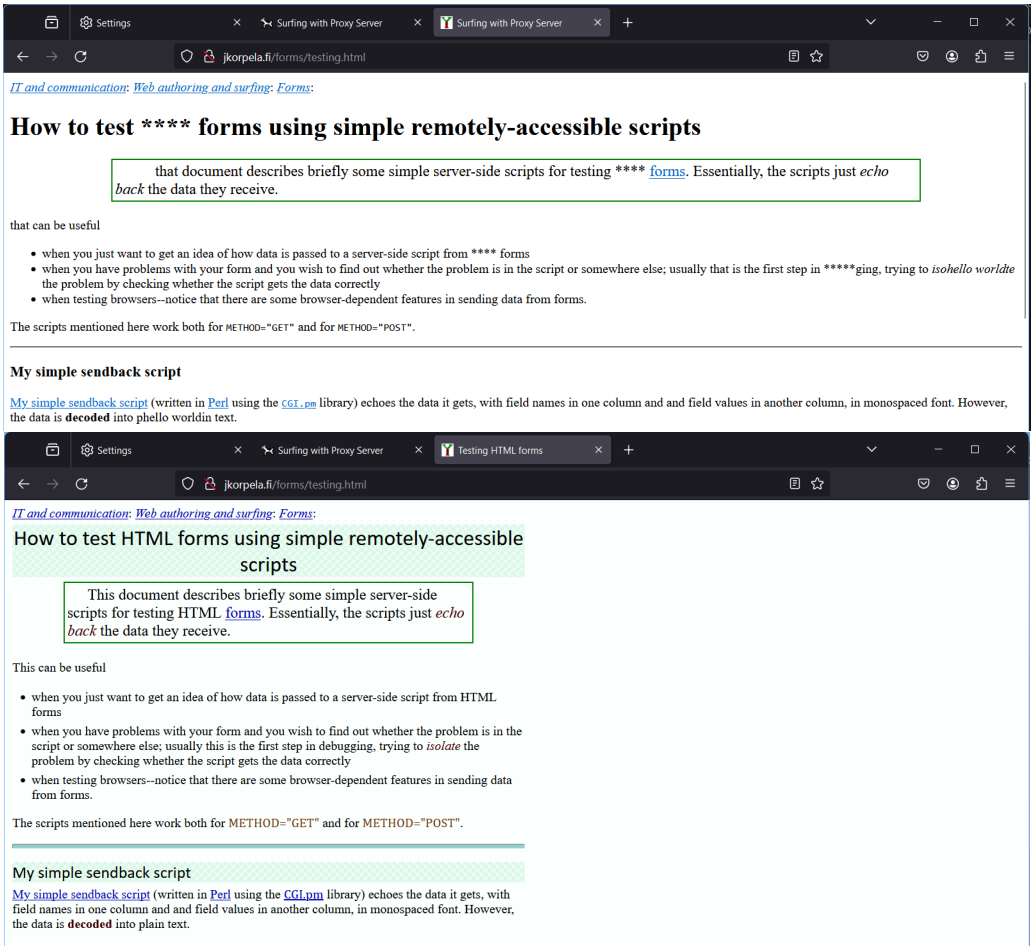

Figure 8: Website with before and after censoring the word "HTML"