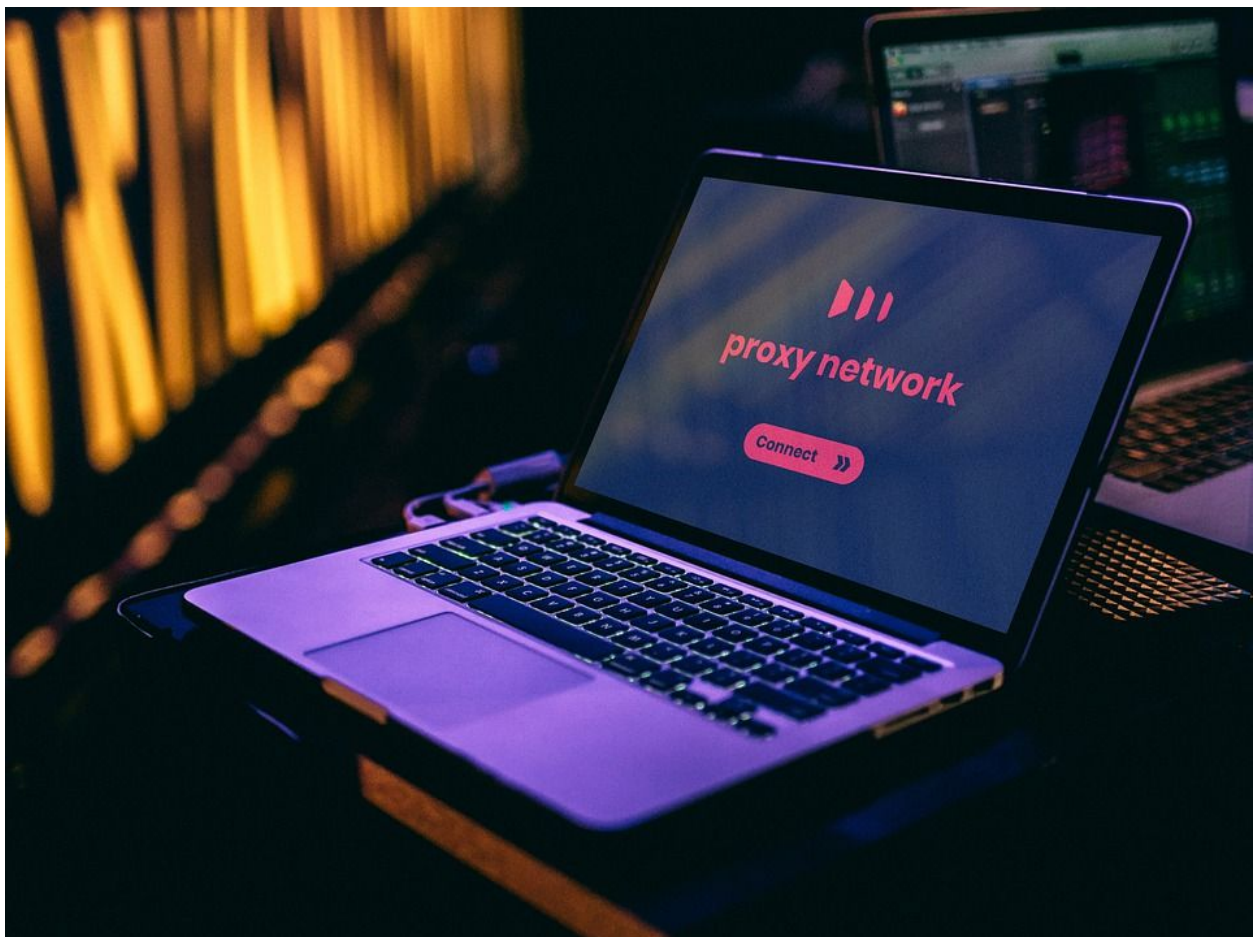


# Proxy HTTPS Cache

Gaëtan RIVAL - Raphaël AUDIN





# Table of contents

<b>Introduction</b>	<b>3</b>
Presentation of project	3
Work access links	4
Tools used (software)	5
<b>Squid Answer</b>	<b>6</b>
The first proof of the concept: HTTP	6
The second proof of the concept: HTTPS	8
Conclusion on the Squid solution	10
<b>CProxy Answer</b>	<b>11</b>
I. The first proof of the concept (self-certification HTTPS)	11
II. The second proof of the concept (combine Cproxy with Kameleon)	12
<b>Conclusion</b>	<b>13</b>
<b>Bibliography</b>	<b>14</b>
<b>Appendix</b>	<b>15</b>



## Introduction

### I. Presentation of project

The objective of this project was to search for solutions for the realization of a cache supporting HTTPS (and HTTP). This project follows a project carried out several years ago when HTTPS was less developed.

However, we are starting from scratch as HTTPS is the secure version of HTTP and it is important to fully understand the HTTP version before starting to work on HTTPS. The difference with HTTP is that HTTPS has an encryption layer like SSL or TLS.

The big difference with the HTTP version is that we will have to set up a certification authority in order to be able to manipulate the certificates needed to encrypt HTTPS communications. This requires an understanding of the so-called MITM(Men In The Middle) approach.

Once the cache is operational, it will be necessary to integrate this solution into the Kameleon system image generation tool.



## II. Work access links

A common workspace has been set up for this project. Our work including proofs of concepts, written documents such as the SRS, our progress has been regularly updated on the university's GitLab gricad.

Our project can be found at the following address:

<https://gricad-gitlab.univ-grenoble-alpes.fr/Projets-INFO4/19-20/21>

For the sake of structuring, we have separated the drafting of proofs of concept and documents.

We also used the <https://air.imag.fr> page where the objective of the project and the work plan to carry out the project were formulated.

Our project page can be found at the following link:

[https://air.imag.fr/index.php/Proxy\\_Cache\\_HTTPS](https://air.imag.fr/index.php/Proxy_Cache_HTTPS)

On this site, you will also find a link to our progress report and the various evaluation documents (final report, mid-term presentation,...)



### III. Tools used (software)

This project required the use of many utilities. To start we were interested in *Mitmproxy* and *CProxy* software.

The first one is an interactive interception proxy, SSL/TLS compatible. It has different functionalities such as *Mitmdump* who is the command-line version of *Mitmproxy* and *Mitmdump* who is a web-based interface for *Mitmproxy*.

The second (*CProxy*) is HTTP and HTTPS caching proxy server. It will intercept the Browser's HTTP/HTTPS requests and decrypt the traffic on-the-fly to store every successful HTTP/HTTPS answer.

We were also interested in a side solution that was not presented to us: Squid.

This software is quite well known and used that's why we decided to deepen this approach in addition to *CProxy* and *Mitmproxy*. We installed *squid* on *Ubuntu* on a virtual machine. So we used *VirtualBox*. *Squid* is normally managed in command line but we installed on our virtual server the utility *Webmin* to have a graphical interface of management.

As with any network-related project, we need a tool to look at what is going to happen on the network during our manipulations. We will use *Wireshark* to view and analyze packets.

During the development of our project, before the starting, we have worked for 3 weeks to find different answers at this issue, in other words how to set up Proxy Https Cache in Kameleon. With our searches, we have obtained two final answers. The first is Squid that is a tool very strong in the network world. Then, we have found a developing project that bases only on aim HTTPS Proxy.

## Squid Answer

### I. The first proof of the concept: HTTP

In order to isolate the systems (so as not to install Squid locally) in order to avoid technical errors and to have a simplified view of network exchanges via Wireshark, we have installed an OS called Lubuntu (a lighter version of ubuntu).

We installed Squid by command line once the service is launched and operational. As we explained in the previous chapter, Squid is normally administered by command line. Most of the actions are done in the file: /etc/squid/squid.conf

Squid is very sensitive, i.e. a single space can sometimes stop the service and put it in error while preventing its restart.

This is the reason why we installed Webmin (by command line too) allowing us to have a web interface in order to administer Squid more easily.

**Configure Proxy Access to the Internet**

No proxy

Auto-detect proxy settings for this network

Use system proxy settings


Manual proxy configuration

HTTP Proxy: 192.168.0.54 Port: 8080

Also use this proxy for FTP and HTTPS

HTTPS Proxy: 192.168.0.54 Port: 8080

FTP Proxy: 192.168.0.54 Port: 8080



Once the clients were configured (previous capture of the Firefox browser's network configuration page), we just had to test and analyze by Wireshark the requests transiting on the network.

Note: We checked the box "Also use this proxy for FTP and HTTPS" but there is an important configuration to do at the server level, we will come back to this later.

To test we looked for an unsecured site (HTTP) it's not so simple anymore that's why we had to take over this project realized several years ago which was initially intended for HTTP.

For the HTTP Squid tests, we used [www.onisep.fr](http://www.onisep.fr).

We made requests on this site. We do the manipulations when the cache of the Squid server is empty. We can see that our client transfers his request to the server and that in a second time the server addresses directly to the Onisep web site to make the request and finally the Squid server gives the answer to the server.

In a second time (a few minutes later) we repeat the interrogation to access the Onisep web site. This time there is an exchange only between the client and the Squid server because the information on the Onisep page is stored in the cache of our Squid server.

Thanks to this part we check that our server is operational for unsecured requests, i.e. requests using HTTP.

Now we will be able to configure this same server to integrate HTTPS. This is what we will develop next.

## II. The second proof of the concept: HTTPS

Now that the HTTP part works we can look at the HTTPS/SSL part, a useful part of this project. The secure part of Squid is quite complex and requires a good understanding of how proxies and certificates work. The complex part is that you have to set up a certification authority and it is this certification authority that will generate certificates for the sites going through the proxy.

So there is a very important part of configuration on the server but it is also necessary to import the certificate created by our authority in the browsers of our client machines. We used the following website to make the configuration: <https://techexpert.tips/fr/squid-fr/installer-squid-avec-le-support-https-sur-ubuntu-linux/>

We started by adapting to our configuration (address space) the file *openssl.conf* (*/etc/ssl/openssl.conf*), then we created the different directories needed to store the certificates making sure we have the right permissions on these directories.

We'll be able to create our own authority. As shown in the capture below we will create a self-signed root certificate:

```
srv@srv-VirtualBox:~$ sudo openssl req -new -newkey rsa:4096 -sha256 -days 3650
-nodes -x509 -extensions v3_ca -keyout certificat.pem -out certificat.pem
Generating a 4096 bit RSA private key
.....
.....++++
.....++++
writing new private key to 'certificat.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]:Grenoble
Locality Name (eg, city) []:Grenoble
Organization Name (eg, company) [Internet Widgits Pty Ltd]:PolytechGR
Organizational Unit Name (eg, section) []:PGR
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:contact@pgr.net
srv@srv-VirtualBox:~$
```



We have just created our certificate valid for 10 years (this is the -days 3650 option). After that, we create a DER (Distinguished Encoding Rules) encoded certificate. This is the certificate that will be imported in the client browsers.

```
srv@srv-VirtualBox:~$ sudo openssl x509 -in certificat.pem -outform DER -out certificat.der
```

```
srv@srv-VirtualBox:~$ ls | grep certificat
certificat.der
certificat.pem
srv@srv-VirtualBox:~$
```

It's this moment, that we can import the certificate with the .der extension on client browsers.

In the browser settings, in the "certificate" section we can download our certificate.



Now it's possible to test in the same way as for HTTP. We notice that in spite of the HTTP or HTTPS site this last one is stored in the cache of our proxy.



### III. Conclusion on the Squid solution

Squid is a very interesting solution because it is complete. It's very widespread in the professional world.

This solution was not proposed to us as a possible solution for this project but after an exchange with Mr Richard, we decided to deepen this solution in parallel with the solutions that were proposed to us, i.e. CProxy, Mitmproxy.

After various researches and experiments, we managed to set up a functional solution in HTTP and HTTPS with this tool.

Nevertheless, we decided not to continue in this way because Squid although complete is very complex and heavy to manage because Squid is to be used with the access list (ACL) which is not necessarily adequacy with the finality of what we want to make of it.

Indeed once the cache is functional, our goal is to integrate our solution with the Kameleon tool to create a system image. After a research session, we have not found how to link Squid and Kameleon, that's the main reason why we did not choose this solution.



## CProxy Answer

*Cproxy is a project currently in developing that we can found on the GitHub platform. This is the link to this project if you wanna more information about this:*  
<https://github.com/coiby/cproxy>

### I. The first proof of the concept (self-certification HTTPS)

In the first time, we have worked about the self-certification HTTPS that we find in Proxy HTTPS tool. This allows the security and viability of many identities. Before to work on Cproxy, we have seen that lot of tools use Mitmproxy project's tool. It gets certificates that can decrypt encrypted traffic on connection. This certificate authority is very famous and the same used by Cproxy tool. So we have gathered our works to Mitmproxy and its working.

Moreover, we have learned to have expert knowledge of Mitmproxy and CProxy. So we have worked the certificate authority at the same time on Mitmproxy and CProxy. We have done the installation of these two tools and test them.

Our proof of the concept on GitHub :

1. To install one on these two tools (Mitmproxy or Cproxy)
2. To set up proxy of browser to listen on the port 80 in localhost
3. To access on the web site '<http://cp.ca>' to accept and install certificate authority from Mitmproxy
4. The tool is done and you can use it to test him and see traffic between network identities and our proxy



## II. The second proof of the concept (combine Cproxy with Kameleon)

Once time that we have the knowledge about the working of CProxy with the certificate authority from Mitmproxy, the new issue was to combine CProxy and Kameleon. Our teacher tells us that Kameleon already gets cache proxy but only to HTTPS traffic. So we have to switch the Kameleon's cache Proxy (cf. Polipo) with that of CProxy.

Before getting started this task, we have done the same understanding of work. As we have said in the introduction, Kameleon is a tool to generate customized appliances by recipe making. To this, we have manipulated several recipes to understand how Kameleon is working and how to configure HTTP cache of this tool.

We have been meeting many issues understanding about coding :

- Create and execute a recipe
- Locate code source of Polipo with its dependencies
- Understand the factors to change to CProxy

At this moment, we have become aware of the Kameleon's complexity but also its power in this field. Primarily, it's a work of reading and understanding to merge Kameleon and HTTPS cache of CProxy.

Then, we are success to remove Polipo, that was the ex HTTP cache of Kameleon and set up Cproxy inside the source code of Kameleon. Unfortunately, the current issue that we have, it's a compatible issue, more precisely a TCP connection issue. We have able to create new recipes with this new HTTPS cache but we don't know yet why it doesn't work. We think that we miss knowledge about network used by Kameleon in this application. It isn't as simple as we think, just to switch between Polipo and CProxy. There are other factors that we ignore and that must be main elements.



## Conclusion

This project has been an interesting and rich experience in the network world. Gaëtan RIVAL and Raphaël AUDIN, we are happy to work about this because it's a field which captivates us each day and we know the importance of the cybersecurity.

We have learned to work on a new technology with any knowledge. We can tell that we have done three different steps. The first is to understand the expectation of customers, the second is to inform it-self technologies asked and in last time, once time all is clear, to develop and test our ideas.

In the end, we haven't reached our main goal, to set up an HTTPS cache Proxy inside Kameleon application. But, we have found an eventually answer at this issue with CProxy which reply at our expectation. It's possible with more time or maybe with more knowledge about the Kameleon's working, we will be able to finish our project.

To conclude, the CProxy tool can be merged with Kameleon because CProxy gets already Proxy HTTPS cache used the certificate authority from Mitmproxy which can be also used by Kameleon. Our proofs of concept show conclusive results so a person who works on the Kameleon development could easily include this new feature.



## Bibliography

Project Git documentation:

<https://gricad-gitlab.univ-grenoble-alpes.fr/Projets-INFO4/19-20/21/docs>

Project Git Prof Concept:

<https://gricad-gitlab.univ-grenoble-alpes.fr/Projets-INFO4/19-20/21/preuveconcept>

Dashboard Project:

[https://gricad-gitlab.univ-grenoble-alpes.fr/Projets-INFO4/19-20/21/docs/-/blob/master/REA\\_DME.md](https://gricad-gitlab.univ-grenoble-alpes.fr/Projets-INFO4/19-20/21/docs/-/blob/master/REA_DME.md)

CProxy Git: <https://github.com/coiby/cproxy>

Mitmproxy web site: <https://docs.mitmproxy.org/stable/>

Kameleon web site: <http://kameleon.imag.fr/>

Kameleon Git: <https://github.com/oar-team/kameleon>

Illustration: <https://pixabay.com/fr/photos/proxy-serveur-proxy-proxy-gratuit-4620558/>

(Authorized reuse without commercial purpose)

## Appendix

### SQUID - Certificate view created by our certification authority

Could not verify this certificate because the issuer is unknown.

#### Issued To

Common Name (CN)  
Organization (O) PolytechGR  
Organizational Unit (OU) PGR  
Serial Number 00:D7:8F:14:BE:74:24:9E:58

#### Issued By

Common Name (CN)  
Organization (O) PolytechGR  
Organizational Unit (OU) PGR

#### Period of Validity

Begins On 30 avril 2020  
Expires On 28 avril 2030

#### Fingerprints

SHA-256 Fingerprint 46:78:4C:F4:FA:E5:53:C0:F4:E5:DD:93:C8:71:D3:6B:  
52:06:BD:6D:20:74:1B:52:AD:18:25:F2:8F:C1:23:C9  
SHA1 Fingerprint 93:49:5F:8F:24:47:7B:E2:BD:46:7F:AE:AE:E8:F1:6D:AD:B0:35:54

## SQUID - web interface and server configuration

The screenshot shows the Squid Proxy Server web interface. The main menu includes: Ports and Networking, Other Caches, Memory Usage, Logging, Cache Options, Helper Programs, Access Control, Administrative Options, Authentication Programs, Delay Pools, Header Access Control, and Refresh Rules. The 'Cache Options' page is currently selected, displaying 'Caching and Request Options'.

**Caching and Request Options**

Default (/var/spool/squid)    Listed..

Cache directories	Directory	Type	Size (MB)	1st level dirs	2nd level dirs	Options
	/var/spool/squid	UFS ▼	100	16	256	
		UFS ▼				

**Average object size**    Default     kB  

**Objects per bucket**    Default  

**Don't cache URLs for ACLs**

all    localnet    SSL\_ports    Safe\_ports

CONNECT

**Maximum cache time**    Default     seconds



## CProxy - Diagram of the first proof (self-certification HTTPS)

