Code & Go

Web server administration

Day 2

Teaching managers

pedagowac@epitech.eu

Table of contents

Administrative details

- The project is to be done alone.
- **Nothing to submit** as this day will help you prepare everything for the big journey ahead ☺

Introduction

Today, you will discover how to secure a server and install additional components.

Since there is nothing to hand in, you are strongly advise to spend this day along with your teacher so that you can ask questions at the right time.

All product names, *logos*, and brands are *property* of their respective *owners*.

Securing

We are now going to see how to set minimum security to your previously installed web server.

Apache

An Apache server with basic configuration makes some information available that can be used by sketchy peoples ☹

We are now going to deal with this.

Server information

When your server encounters an error and cannot process the request, Apache will, by default, give information about the server's type and version.

In your browser, access a page that does not exist on your server:

```
http://localhost/doesnt_exists
```

Go to folder *etc/apache2*

Edit the file *conf-available/security.conf*

Locate the line *ServerTokens* and change its value to: *Prod*

Locate the line *ServerSignature* and change its value to: *Off*

Restart apache

```
$> service apache2 restart
```

Refresh the page that does not exist.

Disable the content listing of the folder

By default, when you access a folder of your server without any *index.html* or *index.php* file, Apache will list the content of the folder and enable its file tree.

To disable this behavior, open the file *apache2.conf* in */etc/apache2*

Modify the directive *Options* of Directory */var/www*

```
<Directory /var/www/>
        Options FollowSymLinks Multiviews
        AllowOverride None
        Require all granted
</Directory>
```

Restart apache.

Security module

Install the Apache security module

```
$> apt-get install libapache2-modsecurity
```

Enable the previously installed module

```
$> a2enmod security2
```

Obviously ☺ you need to restart apache.

We are now going to define rules for the security module. For this task, we use the rules given by the developer of the module. To go further, you can later visit the developer's site.

Create a folder */etc/apache2/crs-tecmint*

```
$> mkdir /etc/apache2/crs-tecmint
$> cd /etc/apache2/crs-tecmint
```

Download the configuration files.

```
$> wget http://tinyurl.com/security-coding-academy -O tarball.zip
```

Unzip the archived file, rename the folder and delete the tarball.

```
$> unzip tarball.zip
$> mv owasp-modsecurity-crs-master/ owasp-mod-security-crs
$> rm tarball.zip
```

Prepare the configuration file.

```
$> cd owasp-mod-security-crs
$> cp modsecurity_crs_10_setup.conf.example modsecurity_crs_10_setup.conf
```

Open the apache configuration file

```
$> cd ../..
$> emacs apache2.conf
```

At the end of the file add the following lines to include security rules.

```
<IfModule security2_module>
    Include crs-tecmint/owasp-mod-security-crs/modsecurity_crs_10_setup.conf
    Include crs-tecmint/owasp-mod-security-crs/base_rules/*.conf
</IfModule>
```

Restart apache.

Evasive module

Installation of the module

```
$> apt-get install libapache2-mod-evasive
```

Restart apache.

IP filtering

If you have a site which can only be accessed by a certain network or IP address, you can filter the authorized Ips by editing the *apache2.conf* for the directory you want. Theses are examples, the three together doesn't really mean a thing.

```
#The client must correspond to the ip
    Require ip 176.16.1.1
#Accept all clients
    Require all granted
#Refuse all requests
    Require all denied
```

Or by using some “Require” scopes:

```
<RequireAny>
    #require one of the next rule to be true
    <RequireAll>
        #require all the next rules to be true
        <RequireNone>
            #require all the next rules to be false
            Require host bad.host.com
        </RequireNone>
        <RequireAll>
            Require user www-data
            Require ip 127.0.0.1
        </RequireAll>
    </RequireAll>

    <RequireAll>
        #require all the next rules to be true
        Require user root
        Require ip 123.123.123.123
    </RequireAll>
</RequireAny>
```

Play a little bit with thoses rules.

To understand how to use it. Look at http://httpd.apache.org/docs/2.4/en/howto/access.html detailed information.

SSH

To grant SSH minimum security, we can make some simple adjustments.

Open the file */etc/ssh/sshd_config*

Change the listening port on port 42.

Authorize only the user jerry to connect to SSH.

Block a list of users.

Refuse the login root in SSH.

Add a SSH connection banner.

```
#/etc/ssh/sshd_config

Port 42

AllowUsers jerry

DenyUsers foo

PermitRootLogin no

Banner /etc/issue
```

Restart ssh.

```
$> service ssh restart
```

FireWall

IPtables (associated to Netfilter) is one of the best firewalls for Linux, and certainly the most widespread. You can find many configuration scripts on this subject. Here is one which you need to adapt to your configuration. At any moment, use the command *`iptables -L -v`* to list the rules in place.

These apply on 3 chains: *`INPUT`* (in input), *`FORWARD`* (in case of network routing) and *`OUTPUT`* (in output). Actions to take are *`ACCEPT`* (accept the package), *`DROP`* (throw it away), *`QUEUE`* and *`RETURN`*.

Arguments used:

i : input interface

i : output interface

t : table (by default *filter* containing the chains INPUT, FORWARD, OUTPUT)

j : rule to apply (Jump)

A : add the rule at the end of the chain (Append)

I : inserts the rule at the start of the chain (Insert)

R : replace the rule in the chain (Replace)

D : deletes a rule (Delete)

F : deletes all the rules (Flush)

X : deletes the chain

P : rule by default (Policy)

lo : localhost (or 127.0.0.1, local machine)

We are going to create a script that will be launched at startup to set the basic rules.

Create the file *`/etc/init.d/firewall`*

```
#!/bin/sh

#/etc/init.d/firewall

# Empty the tables

iptables -t filter -F

# Empty personal rules

iptables -t filter -X

# Forbid any incoming or outgoing connection

iptables -t filter -P INPUT DROP

iptables -t filter -P FORWARD DROP

iptables -t filter -P OUTPUT DROP

# ---

# Do not break established connections

iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

iptables -A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

# Authorize loopback

iptables -t filter -A INPUT -i lo -j ACCEPT

iptables -t filter -A OUTPUT -o lo -j ACCEPT

# ICMP (Ping)

iptables -t filter -A INPUT -p icmp -j ACCEPT

iptables -t filter -A OUTPUT -p icmp -j ACCEPT

# ---

# SSH In

iptables -t filter -A INPUT -p tcp --dport 42 -j ACCEPT

# SSH Out

iptables -t filter -A OUTPUT -p tcp --dport 42 -j ACCEPT

# DNS In/Out

iptables -t filter -A OUTPUT -p tcp --dport 53 -j ACCEPT

iptables -t filter -A OUTPUT -p udp --dport 53 -j ACCEPT

iptables -t filter -A INPUT -p tcp --dport 53 -j ACCEPT

iptables -t filter -A INPUT -p udp --dport 53 -j ACCEPT
```

```
# NTP Out

iptables -t filter -A OUTPUT -p udp --dport 123 -j ACCEPT

# If you host a web server (Apache):

# HTTP + HTTPS Out

iptables -t filter -A OUTPUT -p tcp --dport 80 -j ACCEPT

iptables -t filter -A OUTPUT -p tcp --dport 443 -j ACCEPT

# HTTP + HTTPS In

iptables -t filter -A INPUT -p tcp --dport 80 -j ACCEPT

iptables -t filter -A INPUT -p tcp --dport 443 -j ACCEPT

iptables -t filter -A INPUT -p tcp --dport 8443 -j ACCEPT

#If you host an FTP server:

# FTP Out

iptables -t filter -A OUTPUT -p tcp --dport 20:21 -j ACCEPT

# FTP In

modprobe ip_conntrack_ftp # ligne facultative avec les serveurs OVH

iptables -t filter -A INPUT -p tcp --dport 20:21 -j ACCEPT

iptables -t filter -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
```

When you have defined all the rules, make this file executable:

```
chmod +x /etc/init.d/firewall
```

You can test it by directly executing it in command line. **Make sure you always have the control over your machine** (reconnect in SSH, verify the availability of services such as web, ftp, mail...). In case of error, restart the server, the rules will be forgotten so that you may proceed. But, if the tests are conclusive, add the script at startup so that it protects the server as soon as it boots.

To add it to the scripts called on startup:

```
$> update-rc.d firewall defaults
```

To remove it, you can use the following command:

```
$> update-rc.d -f firewall remove
```

Restart or execute

```
$> /etc/init.d/firewall
```

to enable filtering.

Fail2Ban

Fail2ban is a script which monitors network access using the server logs. When it detects recurring authentication errors, it takes counter-measures by banning the IP address thanks to **iptables**. This can prevent many *bruteforce* attacks and/or by dictionary.

Installation

```
$> apt-get install fail2ban
```

Configuration

```
$> emacs /etc/fail2ban/fail2ban.conf
```

loglevel

Level of detail of the logs (3 by default)

logtarget = /var/log/fail2ban.log

Path to the log file (description of actions undertaken by fail2ban)

Services to monitor are stored in **jail.conf**. We recommend to make a copy named **jail.local** that will automatically be used instead of the example file.

```
$> cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local

$> emacs /etc/fail2ban/jail.local
```

Several global parameters:

```
ignoreip = 127.0.0.1
```

List of IP addresses to ignore by fail2ban

```
bantime = 600
```

Ban duration in seconds

```
maxretry = 3
```

Number of tries authorized by a connection before being banned

```
destemail monitoring@test.com
```

email address recipient of notifications

```
Action
```

Action to undertake in case of positive detection (see in/etc/fail2ban/action.d/)

Each section possesses its own parameters that supersede the global ones if they are mentioned :

```
Enabled
```

Monitoring enabled (true) or not (false)

```
maxretry, bantime, ignoreip, destmail
```

See above

```
Port
```

IP port in question

```
Logpath
```

log file to analyse to detect anomalies

```
Filter
```

Filter used to analyse the log

By default, filters are stored in **/etc/fail2ban/filter.d**. They contain generally an instruction **failregex** followed by a regular expression which matches the detection of an authentication error. For example, for the mail service:

```
failregex = LOGIN FAILED, ip=[<HOST>]$
```

Note: This can be set directly in **jail.local** in the appropriate section to override the directive **filter**.

If needed, modify the ports in the *ssh* section if you followed the above recommendation...

```
enabled = true

port    = 42
```

After modifying the configuration, don't forget to restart fail2ban:

```
$> service fail2ban restart
```

Load Balancing

Definition

Load balancing is a set of techniques which can distribute a workload among various computers of a group. These techniques can both manage service overloads by distributing them on several servers, and reduce eventual unavailability of the service in case of failure of a software or device of a single server 1,2.

These techniques are commonly used, for example, in HTTP services where a high-traffic website must manage hundreds of thousands of requests every second.

Load balancing stems from research on parallel computers. The most common architecture is composed of several load balancers (type of routers dedicated to this task), one being the main, and one or several others as backup which can take over, and a collection of similar computers doing the calculations. We can call this set of servers a server farm or, more generally, a server cluster. The expression server pool is also used.

NGINX

Installation

```
$> apt-get install nginx
```

Launch

If you try launching nginx, you soon realize that it's not possible.

```
$> service nginx start
```

This is normal, because it tries to start on port 80 which is also used by apache. We therefore change the listening port of apache. Also, we are going to change the ports configuration :

```
#/etc/apache2/ports.conf
Listen 88
```

```
#/etc/apache2/site-available/000-default.conf
<VirtualHost *:88>
```

We restart apache.

We take this opportunity to delete the warning "Could not reliably determine the server's fully qualified domain name". At the end of the apahe2.conf file, add :

```
#/etc/apache2/apache2.conf
ServerName 127.0.0.1
```

Then, restart apache.

We restart Nginx

```
$> service nginx restart
```

Go to your browser to the **localhost** address.

Configuration

We are now going to configure Nginx to communicate with apache.

At first, Nginx has no built-in command to enable/disable sites, unlike apache. Thus, we are going to create them.

```
$> cd /usr/bin
$> wget http://tinyurl.com/coding-nginxcmd -O nginx_modsite
$> chmod +x nginx_modsite
```

If you go in the folder `/etc/nginx`, we find similarities with apache

```
$> ls -la /etc/nginx
total 72
drwxr-xr-x   6 root root  4096 Oct  9 15:21 ./
drwxr-xr-x 144 root root 12288 Oct  9 15:21 ../
drwxr-xr-x   2 root root  4096 Dec  1  2014 conf.d/
-rw-r--r--   1 root root  1034 Dec  1  2014 fastcgi.conf
-rw-r--r--   1 root root   964 Dec  1  2014 fastcgi_params
-rw-r--r--   1 root root  2837 Dec  1  2014 koi-utf
-rw-r--r--   1 root root  2223 Dec  1  2014 koi-win
-rw-r--r--   1 root root  3957 Dec  1  2014 mime.types
-rw-r--r--   1 root root  1459 Dec  1  2014 nginx.conf
-rw-r--r--   1 root root   180 Dec  1  2014 proxy_params
-rw-r--r--   1 root root   596 Dec  1  2014 scgi_params
drwxr-xr-x   2 root root  4096 Oct  9 15:21 sites-available/
drwxr-xr-x   2 root root  4096 Oct  9 15:21 sites-enabled/
drwxr-xr-x   2 root root  4096 Oct  9 15:21 snippets/
-rw-r--r--   1 root root   623 Dec  1  2014 uwsgi_params
-rw-r--r--   1 root root  3071 Dec  1  2014 win-utf
```

Our site configuration will be found in `sites-available`

We are going to create a "site" that will contain our upstream and our basic configuration.

```
$> cd /etc/nginx
$> emacs sites-available/upstreams
```

```
# Our first upstream which point on apache on port 88
upstream prod1 {
         server 127.0.0.1:88 ;
}

# As Apache, we don't want to send server information in request's
header
server_tokens off;

# Forbid Iframe
add_header X-Frame-Option SAMEORIGIN;

# Prevent content sniffing
add_header X-Content-Type-Option nosniff;

# XSS protection
add_header X-XSS-Protection "1; mode=block";
```

We create a site by default

```
$> mv sites-available/default sites-available/000-default
$> emacs sites-available/default
```

```
server {
       listen 80;
       server_name coding-academy.prof;

       location/ {
               proxy_pass http://prod1/;
                include    /etc/nginx/proxy.conf;
       }
}
```

Create the file `/etc/nginx/proxy.conf` and put it inside :

```
proxy_redirect          off;
proxy_set_header        Host $host;
proxy_set_header        X-Real-IP $remote_addr;
proxy_set_header        X-Forward-For $proxy_add_x_forwarded_for;
proxy_connect_timeout   90;
proxy_send_timeout      90;
proxy_read_timeout      12000;
proxy_buffer_size       32k;
proxy_buffers           4 32k;
proxy_busy_buffers_size 64k;
proxy_temp_file_write_size      64k;
client_max_body_size            32M;
client_body_buffer_size         512k;
```

We enable the sites

```
$> nginx_modsite -e upstreams
$> nginx_modsite -e default
```

In the file `/etc/hosts`

Add a line with your IP address and *`coding-academy.prof`*

Go to url *`coding-academy.prof`*

SSL

Transport Layer Security (TLS), and its predecessor Secure Sockets Layer (SSL), are security protocols for exchanges on the Internet. The SSL protocol was originally developed by Netscape. IETF continued development by renaming it Transport Layer Security (TLS). We sometimes speak of SSL/TLS to designate both SSL or TLS.

TLS (or SSL) works in client-server mode. It helps to achieve the following security goals:

- authentication of the server;
- confidentiality of exchanged data (or encrypted session);
- integrity of exchanged data;
- in option, authentication of the client (but in reality, this is usually done by the server).

The protocol is very widely used. Its implementation is made easy because the protocols of the application layer, such as HTTP, do not need any major modification to use a secured connection. They just have to be implemented over SSL/TLS, which, for HTTP, has given rise to the HTTPS protocol.

Creation of a self-signed certificate

In order to test how SSL performs, we are going to create a certificate that we will sign ourselves.

```
$> mkdir /etc/nginx/ssl
$> cd /etc/nginx/ssl
$> openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/nginx/ssl/nginx.key -out /etc/nginx/ssl/nginx.crt

Generating a 2048 bit RSA private key
......................+++
........+++
writing new private key to '/etc/nginx/ssl/nginx.key'
-----
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]:IDF
Locality Name (eg, city) []:Paris
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Coding
Academy
Organizational Unit Name (eg, section) []:DSI
Common Name (e.g. server FQDN or YOUR name) []:Your_Name
Email Address []:Your@email.fr
```

Your certificate and your key have been generated

Nginx configuration for SSL

Create now the **ssl.conf** in `/etc/nginx`

```
ssl on;
ssl_certificate/etc/nginx/ssl/nginx.crt;
ssl_certificate_key /etc/nginx/ssl/nginx.key;

# improve ssl performances
ssl_session_cache shared:ssl:50m;
ssl_session_timeout 5m;

# DHE ciphersuits
ssl_dhparam /etc/nginx/ssl/dhparam.pem;

# protection agains BEAST attacks
ssl_prefer_server_ciphers on;

#disable ssl V3 (less secure than TLS)
ssl_protocols TLSv1 TLSv1.1 TLSv1.2;

#ciphers_list
ssl_ciphers 'ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-
SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-
SHA384:kEDH+AESGCM:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-\
AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-
RSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-
SHA:ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE\
-RSA-AES128-SHA:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA:AES128-
GCM-SHA256:AES256-GCM-SHA384:ECDHE-RSA-RC4-SHA:ECDHE-ECDSA-RC4-
SHA:RC4-SHA:HIGH:!aNULL:!eNULL:!EXPORT:!DES:!3DES:\
!MD5:!PSK';
```

We are now going to generate the file **dhparam.pem**

```
$> openssl dhparam -out /etc/nginx/ssl/dhparam.pem 2048
Generating DH parameters, 2048 bit long safe prime, generator 2
This is going to take a long time
..................................................................
.........................+.......................................
...................+..............................................
..................................................................
..................................................................
........................................+......+..................
...................+..............................................
...................
```

Modify the file `/etc/nginx/sites-available/default` to obtain the following

```
server {
        listen 443 ssl;
        server_name coding-academy.prof;

        include  /etc/nginx/ssl.conf;

        location / {
                proxy_pass http://prod1/;
                include /etc/nginx/proxy.conf;
        }
}

server {
        listen 80;
        server_name coding-academy.prof;
        return 301 https://$host$request_uri;
}
```

Restart nginx and refresh your browser. Your certificate is not validated, so your browser may say to you that this domain is unsafe. Proceed anyway, it is your website.

Virtual Host

We now focus on Virtual Host. This will enable us to host different solutions with different domain names on the same infrastructure.

Add the domain name **monsupersite.com** in the list of your **hosts**

Modify the site file **default** of nginx adding a new server

```
server {
       listen 443 ssl;
       server_name monsupersite.com;

       include  /etc/nginx/ssl.conf;

       location / {
                proxy_pass http://prod1/;
          include /etc/nginx/proxy.conf;
       }
}
```

In the available apache sites, add the following directory in /etc/apache2/apache2.conf

```
<Directory /var/www/html/monsupersite.com>
        Require all granted
        Options FollowSymLinks MultiViews Indexes
        AllowOverride None
</Directory>
```

And the corresponding VirtualHost in `/etc/apache2/sites-available/monsupersite.com.conf`

```
<VirtualHost *:88>
        ServerAdmin webmaster@localhost
    ServerName monsupersite.com
    ServerAlias monsupersite.com

        DocumentRoot /var/www/html/monsupersite.com
        ErrorLog ${APACHE_LOG_DIR}/error-monsupersite.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

And enable the site (by the command `a2ensite` seen before)

Create the folder `monsupersite.com` in `/var/www/html`

Inside this folder, create a file named index.html containing "Hello World!"

Don't forgot to add `monsupersite.com` with your ip in `/etc/hosts` file.

Restart both services (apache2 and nginx) and go to `monsupersite.com`

.htaccess

Definition

.htaccess files are configuration files of HTTP Apache servers. What is particular about these files is their location: in the data directories of the web site, instead of the configuration directory of Apache. The extent of their configuration is limited to the content of the directory where they reside. This particularity has two advantages: they can be managed by users which do not have the right to manage the HTTP server itself; modifications can apply without having to restart the HTTP server.

.htaccess files are commonly used to configure access rights, url redirects, customized error messages and associations of file name extensions with a MIME type.

Password protection

To protect a folder with a password, you only need to create a **.htaccess** file and a **.htpasswd** file in the folder in question.

The directive **AllowOverride** must be set to **all** for the directory in question. (You should now know which file you have to modify, don't forget to restart apache after modification).

Create a folder named `secure` in `monsupersite.com` directory. Inside create a `.htaccess` file :

```
AuthName "Secure Zone"
AuthType Basic
AuthUserFile "/var/www/html/monsupersite.com/secure/.htpasswd"
Require   valid-user
```

Go to http://www.htaccesstools.com/htpasswd-generator/ and generate the directive fot the .htpasswd by entering the user-pass you want. Copy the result in the .htpasswd file, and allow the teacher to connect (the password is teacher too)

```
teacher:$apr1$uihClQqC$cyZSwYrtWVYAynRzrqvpi/
```

Possible actions with a .htaccess

Prevent the listing of a directory content

To prevent surfers from listing all the files contained in a directory when there is no index (.cgi, .html, .php etc ....), create a .htaccess file containing the line below:

```
Options -Indexes
```

Redirecting error messages

If you want to use customized error messages or redirect the errors if a web page, create a .htaccess file containing lines of this type:

ErrorDocument error_number message_or_destination

Replace error_number by the corresponding number. The 3 most common errors are:

404: not found,

403: forbidden,

500: internal server error.

Replace "message_or_destination" by the action to perform. To display a simple message, type in the corresponding message between quotes. To redirect to a page, write the path to the page. Here are 2 examples that can help you:

- you wish to display "Sorry, you cannot access this file" in case of a 403 error. Add the following line to your .htaccess:

```
ErrorDocument 403 "Sorry, you cannot access this file"
```

- you wish to send 404 errors on your customized page 404.html:

```
ErrorDocument 404 /404.html
```

Specify a different index file

By default, the index file of a directory is index.html, index.htm or index.php. If you want another file, you can add a line of this type to your .htaccess:

DirectoryIndex name_of_file

For example, if you want to use the page home.html as index page, use the following line:

DirectoryIndex home.html

Permanent redirect

This send a permanent redirect HTTP 301 which informs users, and more importantly search engines, that they must update their links with the new address.

To redirect the whole site to a new address:

```
Redirect permanent / http://new-site.tld/
```

To modify a directory/file:

```
Permanent redirect /old_directory http://new-site.tld/new_directory

Permanent redirect /old_file.php http://site.tld/new_file.php
```

Redirect gone

If a file does not exist anymore, it's nice to replace the 404 message not found with a more explicit message 410 gone:

```
Redirect gone /delete.html
```

Redirect seeother

If you want to change the extension of a file, seeother lets you modify the type of file by sending a code HTTP 303:

```
Redirect seeother /example.doc http://site.tld/example.pdf
```

Redirect Temp

A temporary redirect, of type HTTP 302, can be used when you temporarily move files on another site:

```
Redirect temp / http://autre_site_web.tld/site/
```

Rewriting url

Simple redirect

```
RewriteEngine On

RewriteRule .* testing.php
```

This formula redirects each request on the script testing.php

```
RewriteEngine On

RewriteRule letstest /test_wslash/testing.php
```

This formula redirects each request /letstest on the script /test_wslash/testing.php

This forces the address of your site to be of type www.example.com, which is useful for referencing:

```
RewriteEngine on

Rewritecond %{HTTP_HOST} ^exemple.com$

Rewriterule ^(.*) http://www.exemple.com/$1 [QSA,L,R=301]
```

Redirect to a particular folder without displaying the folder in question

If your site is not present in the target folder, this forces the address of your site to be of the type www.example.com, whereas in reality the page called is:

```
RewriteEngine on

Rewritecond %{HTTP_HOST} ^exemple.com

Rewritecond %{REQUEST_URI} !^/MonSite

Rewriterule ^(.*)$ /MonSite/
```

Rewriting

```
RewriteEngine On

RewriteCond %{REQUEST_URI} !testing.php

RewriteRule (.*) testing.php?var=$1
```

These rules launch the script testing.php with variable GET containing the URL entered by the user.

Redirects automatically the visitor on the site in ssl when he visits the non-secured site

```
RewriteEngine on

Rewritecond %{HTTP_HOST} ^nom_domaine.tld$

Rewriterule ^(.*) https://ssl5.ovh.net/~login_ftp/$1 [QSA,L,R=301]
```

If you want to go to the secured site only for visiting a precise page:

```
RewriteEngine on

RewriteCond %{HTTP_HOST} ^nom_domaine.tld$

RewriteCond %{REQUEST_URI} ~094/page.php

RewriteRule ^(.*) https://ssl5.ovh.net/~login_ftp/$1 [QSA,L,R=301]
```

Node.JS

Definition

Node.js an open source, event-driven software platform in JavaScript designed for scalable network applications. It uses Chrome’s V8 virtual machine and implements under MIT license CommonJS specifications.

Node.js contains a library of integrated HTTP servers, which makes it possible to make a web server work without having to call upon an external software such as Apache or Lighttpd, and gives better control over how the web server functions.

Node.js is increasingly popular as server platform. It is used by Groupon, Ebay, SAP, LinkedIn, Microsoft, Yahoo!, Walmart, Rakuten and PayPal.

Installation

Let’s install NodeJS.

```
$> curl -sL https://deb.nodesource.com/setup_0.12 | bash
$> apt-get install nodejs
$> node -v
v0.12.7
```

Tests

To try your new Node.JS installation out, create a small web server.

Create a folder named **`nodejs`** in your **`home`**

```
$> mkdir /home/your_login/nodejs
```

Create a file server.js in this folder

```
var http = require('http');

var server = http.createServer(function(request, response) {
    console.log('New request!!');
    response.writeHead(200, {'Content-Type' : 'text/plain'});
    response.end('Hello World !');
});

server.listen(3000);
console.log('Server listening on port 3000');
```

We launch.

```
$> node server.js
```

Go to localhost:3000 and see what append.

RoR

Introduction

The Ruby on Rails framework, developed in Ruby, integrates a web server designed to be a development web server which should never be used past development. Thus, when you are ready to go into production with your Ruby masterpiece, you will need to install it on a web server. By default, Apache 2 does not communicate with Ruby, even less so with Ruby on Rails. Therefore, we are going to configure apache so that is can work with RoR.

Installation

```
$> apt-get install ruby-full gem libmysqlclient-dev
```

Rails

Above, we have installed the Ruby package manager called "gem". To install Rails, we are going to use it. All you need to do is:

```
$> gem install rake --verbose
```

```
$> gem install rails --verbose
```

```
$> gem install mysql2 --verbose
```

This last one should take time.

Now test the version (at the time rails version was 4.1.8 yours should differ):

```
$> rails -v
Rails 4.1.8
```

Passenger

Passenger will make our life easy! It will install and configure for us the apache modules necessary for Ruby and Rails management.

```
$> gem install passenger --verbose
```

```
$> passenger-install-apache2-module
```

We select Ruby. Then if passenger say that you have to install some packages, do it and start over.

We install these packages and start over.

Go to folder */var/www/html/*

Don't run the following command as root.

```
$> rails new rubyTest -d mysql -T
```

If you have permissions issues with this command run:

```
$> sudo chown -R your_login: /var/www/html
```

And then restart the last command.

Go to the folder **rubyTest**

```
$> bundle exec spring binstub --all
```

We now modify the **vhost** of **monsupersite.com** to manage our project, adding the rule RailsEnv and changing the DocumentRoot rule to :

```
DocumentRoot /var/www/html/rubyTest/public
RailsEnv development
```

Then, you must specify in the apache2.conf specific rules for this directory:

```
<Directory /var/www/html/rubyTest/public>
        Require all granted
        Options -MultiViews
        AllowOverride All
</Directory>
```

In the file `/var/www/html/railsTest/Gemfile`, add the line

```
gem 'therubyracer'
```

Modify the line of mysql :

```
gem 'mysql2', '~> 0.3.18'
```

Edit the config/database.yml inserting your database password.

Launch the following commands in the railsTest directory

```
 $> bin/rake db:create db:migrate
 $> bundle install
```

Restart apache and start rails server in the railsTest directory

```
 $> rails server
```

Go to monsupersite.com:3000