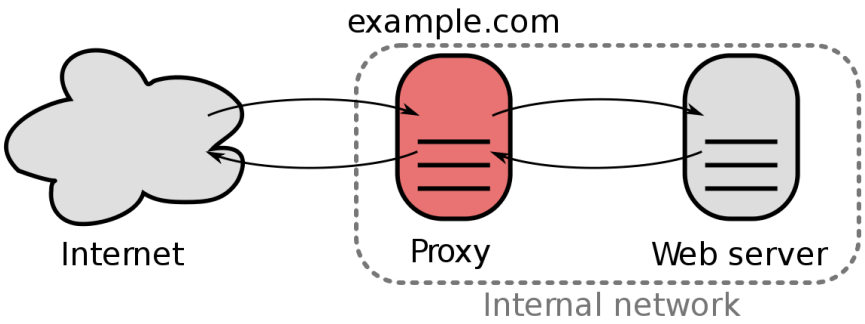
HOWTO - Enable Nginx Reverse Proxy

Overview

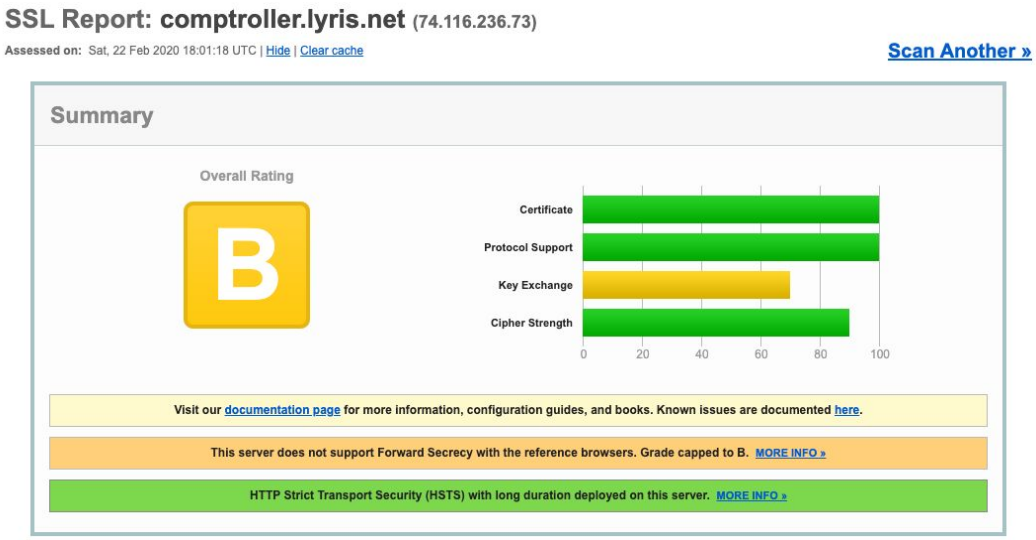
What is the reverse proxy?

A reverse proxy server is a type of proxy server that typically sits behind the firewall in a private network and directs client requests to the appropriate backend server. A reverse proxy provides an additional level of abstraction and control to ensure the smooth flow of network traffic between clients and servers, including security, availability, performance and traffic shaping.



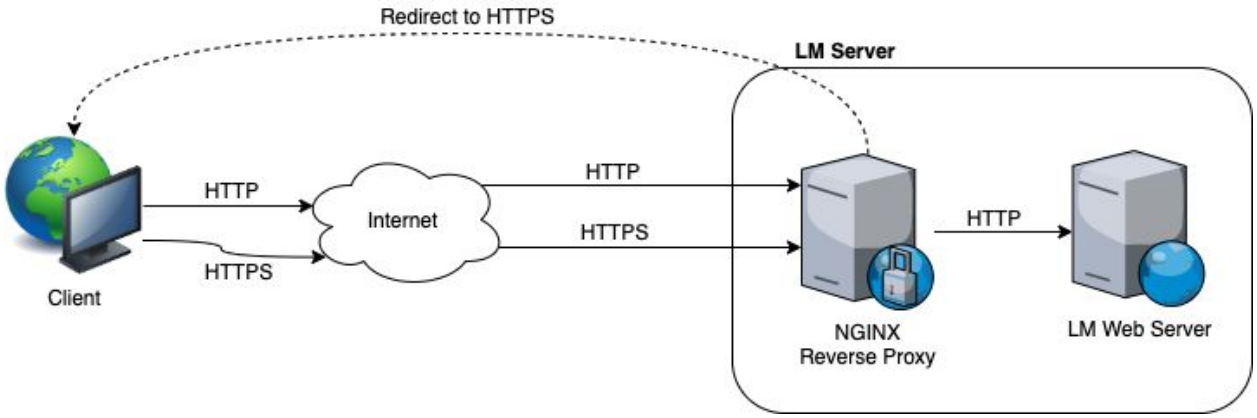
Why do we need it?

The LM web server is not able to get an "A" rate on SSL Test Rating and It is caused by LM doesn't support strong/new ciphers for TLS encrypted connections (HTTPS/SSL). This inability to handle secure TLS/SSL connections is caused by an incomplete HTTP/HTTPS server implementation provided by ActiveTCL 1.4/1.5 and OpenSSL 1.0.2u.

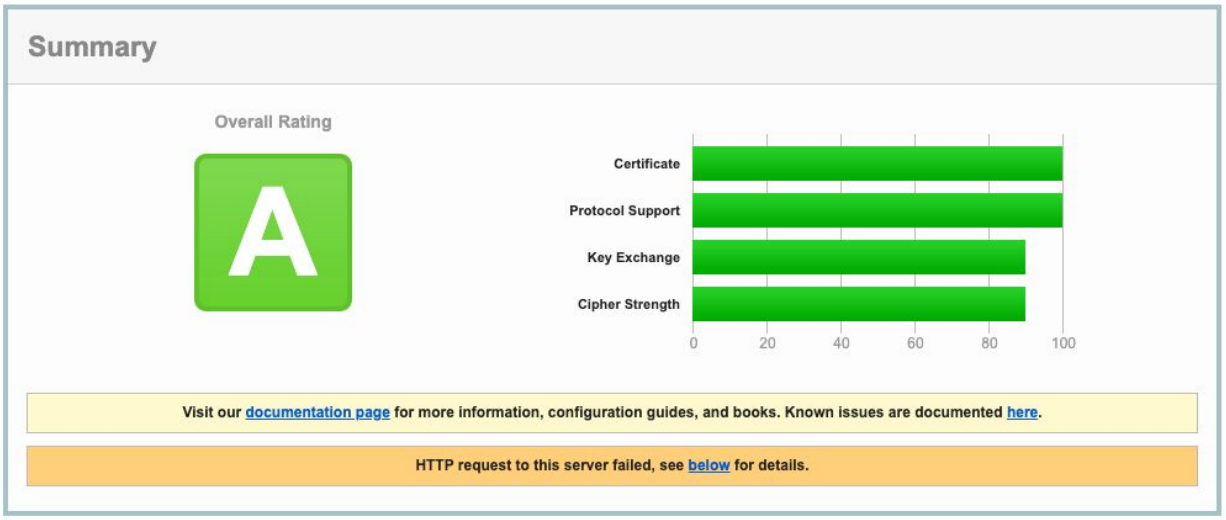


How does it work?

As we can see at "What is the Reverse Proxy" topic, it will work in front of the current LM web server, forcing all HTTP traffic be redirected to HTTPS and using just the LM HTTP implementation to avoid over-heading and improve performance. It will also use all new and well secured TLSv1.2 cyphers to provide an "A" rate on SSL implementation.



It will be implemented by a Nginx instance listening on HTTP (TCP 80) and HTTPS (TCP 443) ports all the internet requests and then mediating all connections to LM HTTP server (TCP 8080 - if they are running on the same server).



Nginx will provide all needed configuration and infrastructure to support an "A" rate.

Installing Nginx

Nginx is an extremely high performance web server which has the ability to handle thousands of requests per second with little hardware requirements. It can be installed on any operating system and it comes as an open source application as well.

PLEASE USE NGINX VERSION >= 1.16 (WE WILL USE 1.18 for this HOWTO).

Linux

Centos 6 / RHEL 6

Install the nginx.repo

Centos 6

Run these commands:

```
# wget
http://nginx.org/packages/centos/6/noarch/RPMS/nginx-release-centos-6-0.el
6.ngx.noarch.rpm
# rpm -ivh nginx-release-centos-6-0.el6.ngx.noarch.rpm
```

RHEL 6

Run these commands:

```
# wget
http://nginx.org/packages/rhel/6/noarch/RPMS/nginx-release-rhel-6-0.el6.ngx.noarch.rpm
# rpm -ivh nginx-release-rhel-6-0.el6.ngx.noarch.rpm
```

Update yum

Run command:

```
$ yum update
```

Install Nginx

Run command:

```
$ yum install nginx
```

Centos 7 / RHEL 7

Create a nginx.repo

Run command:

```
$ vi /etc/yum.repos.d/nginx.repo
```

Centos 7

Paste this at /etc/yum.repos.d/nginx.repo:

```
[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/mainline/centos/7/$basearch/
gpgcheck=0
enabled=1
```

RHEL 7

Paste this at /etc/yum.repos.d/nginx.repo:

```
[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/mainline/rhel/7/$basearch/
gpgcheck=0
enabled=1
```

Update Yum

Run command:

```
$ yum update
```

Install Nginx

Run command:

```
$ yum install nginx
```

Test opening on your browser:

Windows

Download

Nginx comes pre-compiled for Windows which makes it extremely easy to get started. If it did not come pre-compiled, you would need to have a compiler installed on your computer with a full environment. Fortunately, this is not the case.

Download Nginx Windows here: http://nginx.org/en/download.html

Once you've downloaded Nginx for Windows, you can extract it to your folder of choice, we recommend that you install it somewhere easily accessible such as **C:\nginx**.

Verify Nginx Windows Installation

In order to make sure that the service is working with no problems, we recommend that you start a command prompt window and type the following, make sure that you update the path if you've installed it in another folder.

```
C:\nginx\nginx.exe
```

You should be able to go to http://localhost/ and you should see the “Welcome to Nginx” default page. If you see that page, then we can be sure that Nginx has been installed properly. We will now shut it down and install it as a service, to stop it, you can use this command.

```
C:\nginx\nginx.exe -s stop
```

Now, if you were using Nginx as a simple development server, you can use these simple commands to start and stop the server as you need. However, if you will be using it as a production server, you would want to install it as a Windows service.

Configuration

Nginx

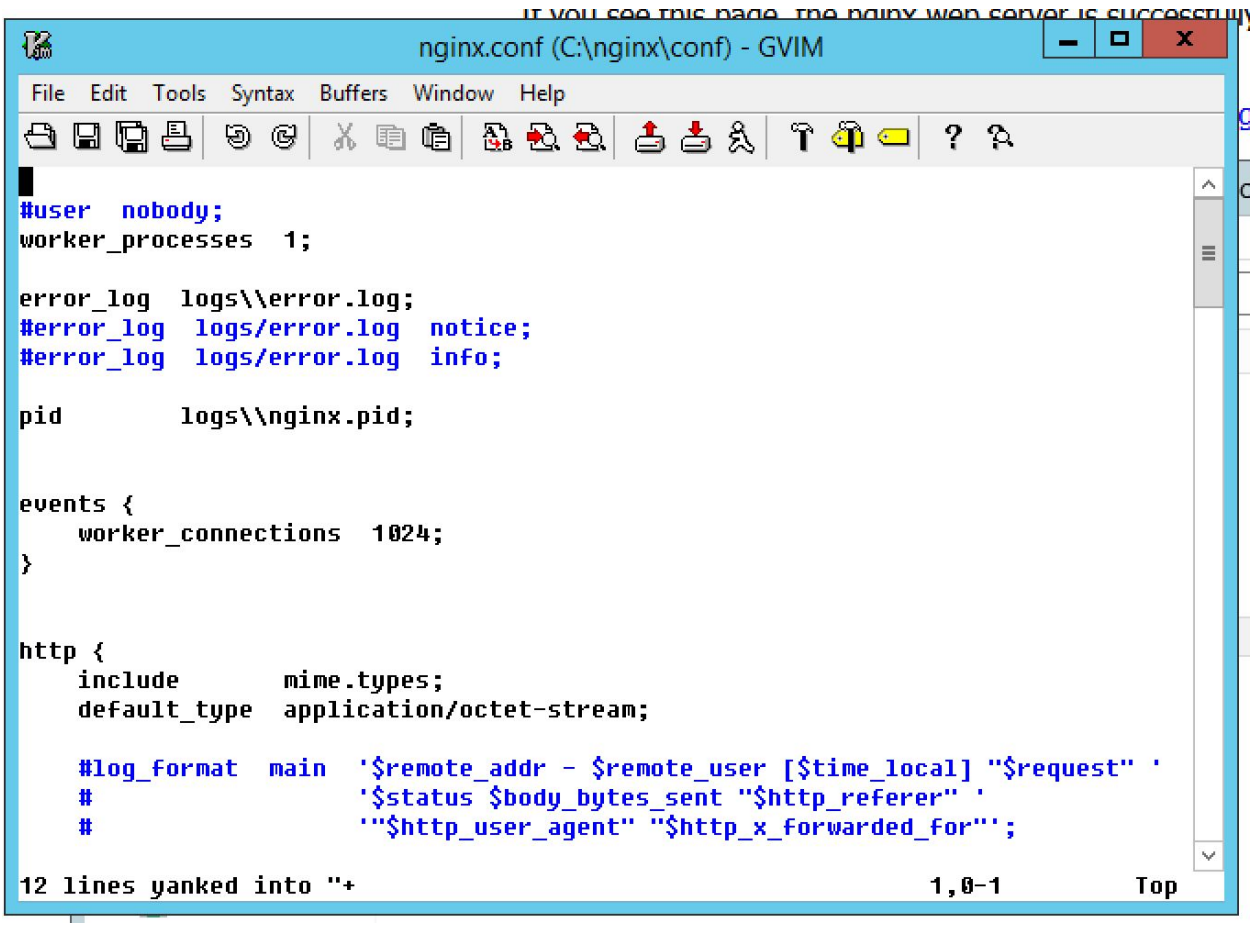
BEFORE START

Linux

No actions required.

Windows

Please be sure to use double-backslashes (\\) to define paths into nginx.conf and also be sure you already have an **error_log** and **pid** directives defined.



SSL Certificates

Certificates and Encodings

At its core an X.509 certificate is a digital document that has been encoded and/or digitally signed according to RFC 5280.

In fact, the term X.509 certificate usually refers to the IETF's PKIX Certificate and CRL Profile of the X.509 v3 certificate standard, as specified in RFC 5280, commonly referred to as PKIX for Public Key Infrastructure (X.509).

X509 File Extensions

The first thing we have to understand is what each type of file extension is. There is a lot of confusion about what DER, PEM, CRT, and CER are and many have incorrectly said that they are all interchangeable. While in certain cases some can be interchanged the best practice is to identify how your certificate is encoded and then label it correctly. Correctly labeled certificates will be much easier to manipulat

Encodings (also used as extensions)

- **DER** = The DER extension is used for binary DER encoded certificates. These files may also bear the CER or the CRT extension. Proper English usage would be “I have a DER encoded certificate” not “I have a DER certificate”.
- **PEM** = The PEM extension is used for different types of X.509v3 files which contain ASCII (Base64) armored data prefixed with a “—— BEGIN …” line.

Common Extensions

- **CRT** = The CRT extension is used for certificates. The certificates may be encoded as binary DER or as ASCII PEM. The CER and CRT extensions are nearly synonymous. Most common among *nix systems
- **CER** = alternate form of .crt (Microsoft Convention) You can use MS to convert .crt to .cer (.both DER encoded .cer, or base64[PEM] encoded .cer) The .cer file extension is also recognized by IE as a command to run a MS cryptoAPI command (specifically rundll32.exe cryptext.dll,CryptExtOpenCER) which displays a dialogue for importing and/or viewing certificate contents.
- **KEY** = The KEY extension is used both for public and private PKCS#8 keys. The keys may be encoded as binary DER or as ASCII PEM.

The only time CRT and CER can safely be interchanged is when the encoding type can be identical. (ie PEM encoded CRT = PEM encoded CER)

Configuring Nginx Certs

At `/etc/nginx/nginx.conf`, and inside the `server { .. }` block, use the `ssl_certificate` and `ssl_certificate_key` like this example:

```
  ssl_certificate /usr/local/lm/tclweb/bin/lyris.net/lyris.net.pem;
  ssl_certificate_key
/usr/local/lm/tclweb/bin/lyris.net/lyris.net.key;
```

SSL Ciphers

At `/etc/nginx/nginx.conf`, and inside the `server { .. }` block, use the `ssl_prefer_server_ciphers` and `ssl_ciphers` like this example:

```
  # enables server-side protection from BEAST attacks
  # http://blog.ivanristic.com/2013/09/is-beast-still-a-threat.html
  ssl_prefer_server_ciphers on;
  #ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3;
  ssl_protocols TLSv1.2 TLSv1.3;
  # ciphers chosen for forward secrecy and compatibility
```

```
  #
http://blog.ivanristic.com/2013/08/configuring-apache-nginx-and-opens
sl-for-forward-secrecy.html
  ssl_ciphers
'ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDS
A-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GC
M-SHA384:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RS
A-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256
:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA:
ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA:ECDHE-RSA-AES256-SHA
:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-RSA-AES256-SHA256:DHE-R
SA-AES256-SHA:ECDHE-ECDSA-DES-CBC3-SHA:ECDHE-RSA-DES-CBC3-SHA:EDH-RSA
-DES-CBC3-SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128-SHA256:AES25
6-SHA256:AES128-SHA:AES256-SHA:DES-CBC3-SHA:!DSS';
```

For more information about ciphers please check this article.

SSL TLVv1.2 only

At `/etc/nginx/nginx.conf` use the `ssl_protocols` to enable just *TLSv1.2*.

Example:

```
ssl_protocols TLSv1.2;
```

Running as a Service

Linux

Type the following chkconfig command:

```
# chkconfig nginx on
```

Windows

We downloaded from https://github.com/kohsuke/winsw/releases the last stable version (**2.10.1 - *WindSW.NETCore31.x64.exe***) and copied it to the Nginx folder (`c:\nginx`) as **`nginxscv.exe`**.

After download, copy, and rename the **`nginxsvc.exe`**, you will need to create a service file inside the Nginx (**`c:\nginx`**) folder, please be sure to create a file with the name **`nginxsvc.xml`** with the following contents:

```
<service>
  <id>nginx</id>
```

```
  <name>nginx</name>
  <description>nginx</description>
  <executable>c:\nginx\nginx.exe</executable>
  <logpath>c:\nginx\logs</logpath>
  <logmode>roll</logmode>
  <depend></depend>
  <startarguments></startarguments>
  <stoparguments>-s stop</stoparguments>
  <workingdirectory>c:\nginx</workingdirectory>
</service>
```

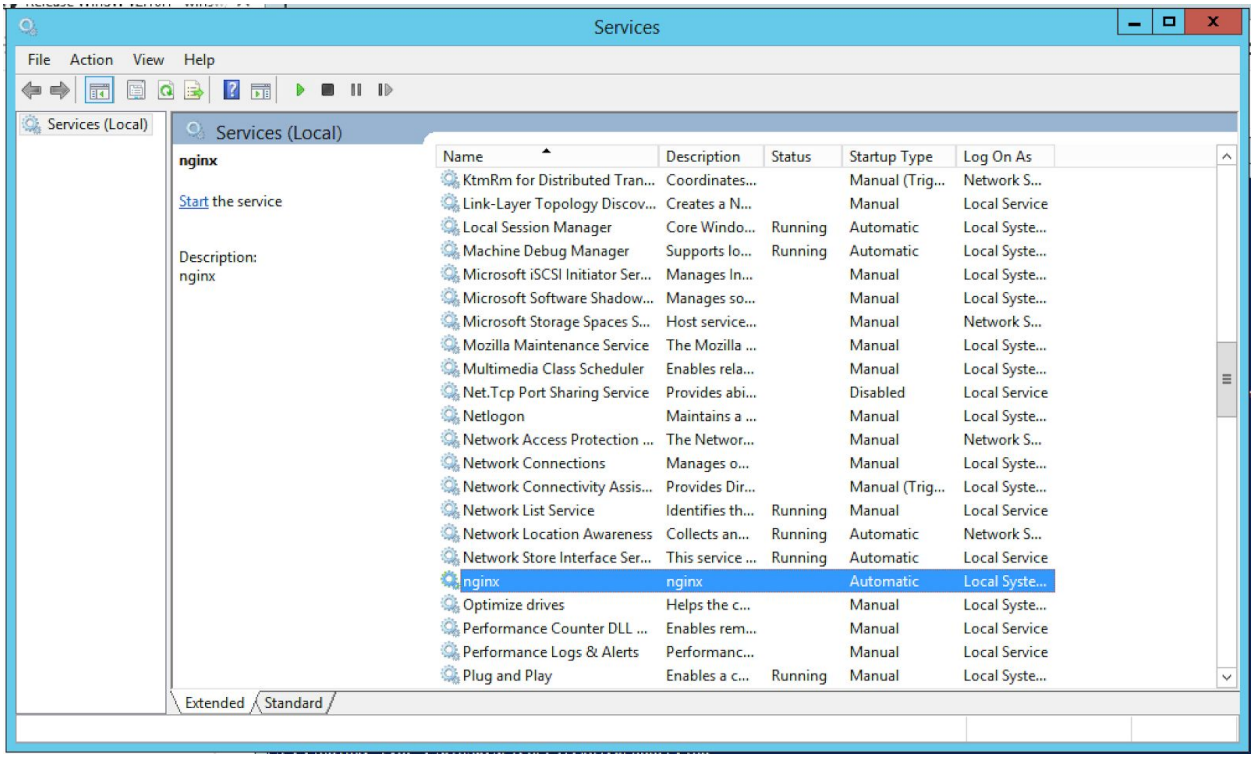
You are now ready to install the Windows service, you can proceed to run the following command:

```
C:\nginx\nginxsvc.exe install
```

You can now proceed to manage the service from your service manager. The easiest and fastest way to access it is to type the following in your command prompt:

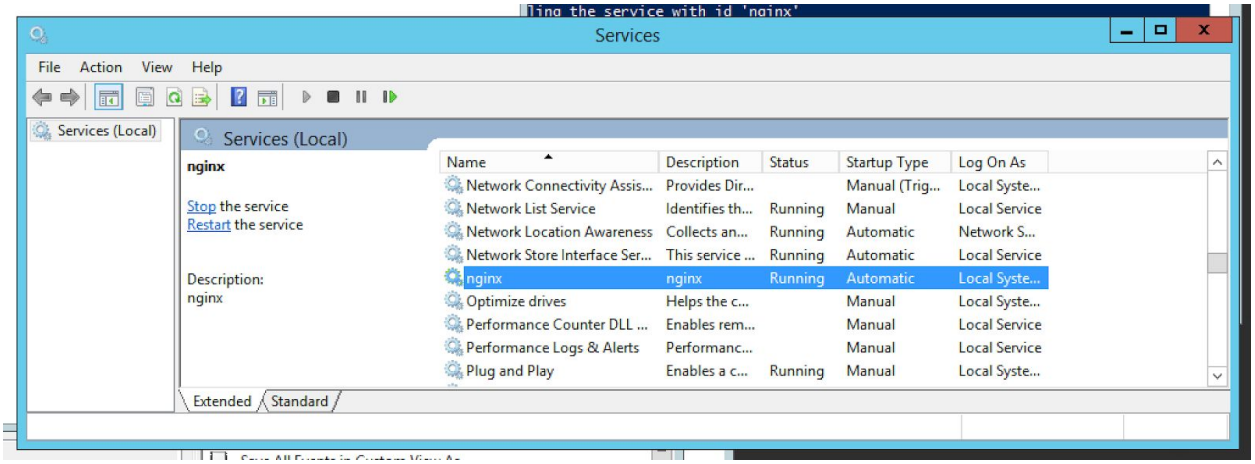
```
services.msc
```

You should be all done at this point.

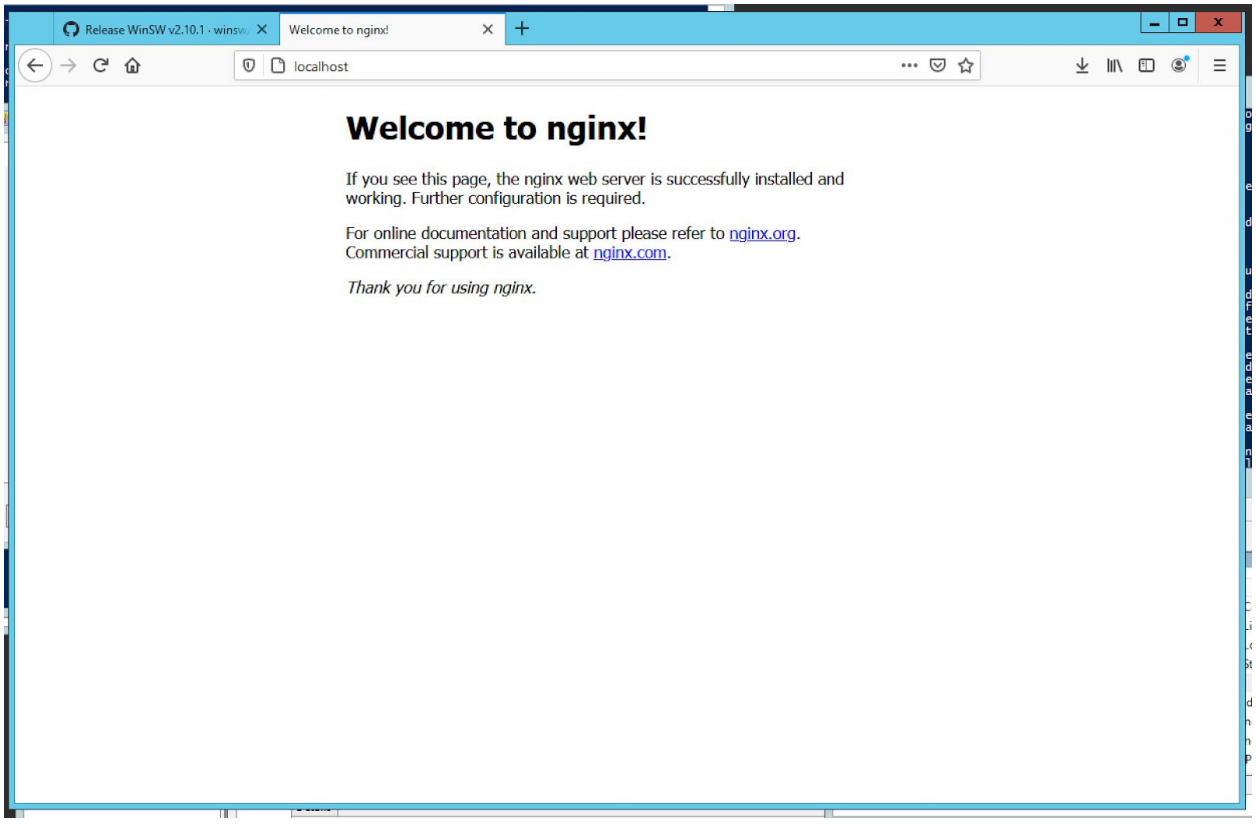


You have Nginx as a service and you can set it up to start automatically when it is booted with your operating system!

After you click on the "Start" button you will see the status "Running"



and then you can open the browser at http://localhost to check if it is opening the default Nginx page.



If you need to disable the Windows default port 80 (HTTP.sys), please run as "Administrator" at the command line:

```
net stop http /y
sc config http start= disabled
```

For more information please check this article:
http://www.devside.net/wamp-server/opening-up-port-80-for-apache-to-use-on-windows

HTTP to HTTPs redirection

At `/etc/nginx/nginx.conf`, and inside the `location /path { .. }` block, use the `return <http_answer_code> <url>` like this example:

```
# redirect all http traffic to https
server {
  listen 80 default_server;
  listen [::]:80 default_server;
  server_name my.servername.com;
  return 301 https://$host$request_uri;
}
```

Reverse Proxy

At `/etc/nginx/nginx.conf`, and inside the `location /path { .. }` block, use the `proxy_pass, proxy_set_header, proxy_cache_valid, proxy_cache_use_stale_error, and proxy_redirect` like this example:

```
  # ... the rest of your configuration
  location / {
    proxy_set_header Host $host;
    proxy_cache_valid 200 7d;
    proxy_cache_use_stale error timeout invalid_header updating
http_500 http_502 http_503 http_504;
    proxy_set_header X-Forwarded-Proto https;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $http_host;
    proxy_redirect off;

    proxy_pass http://url.to.your.service[:<port>]/;
  }
```

Listmanager

Web UI

Disable SSL

1. Edit your `${LM_HOME}/tclweb/bin/tclhttpd.rc` file
2. Find this section

```
#########################
# SSL Configuration

# SSL_REQUEST - should the server ask for certificates from clients?

Config SSL_REQUEST      1

# SSL_REQUIRE - should the server require certificates?

Config SSL_REQUIRE      1
```

3. Switch all to `0` (zero)

```
#########################
# SSL Configuration

# SSL_REQUEST - should the server ask for certificates from clients?

Config SSL_REQUEST      0

# SSL_REQUIRE - should the server require certificates?

Config SSL_REQUIRE      0
```

4. Find this section

```
# USE_SSL2 - Allow the use of SSL version 2
# (You cannot get this with a "no patents" build of OpenSSL)

Config USE_SSL2         1

# USE_SSL3 - Allow the use of SSL version 3
```

```
Config USE_SSL3         1

# USE_TLS1 - Allow the use of TLS version 1

Config USE_TLS1         1
```

5. Switch them to 0 (zero)

```
# USE_SSL2 - Allow the use of SSL version 2
# (You cannot get this with a "no patents" build of OpenSSL)

Config USE_SSL2         0

# USE_SSL3 - Allow the use of SSL version 3

Config USE_SSL3         0

# USE_TLS1 - Allow the use of TLS version 1

Config USE_TLS1         0
```

6. Save the file
7. Restart LM

HTTP and HTTPs ports

8. Edit your `${LM_HOME}/tclweb/bin/tclhttpd.rc` file
9. Find this section

```
# port - the listening port for the server for HTTP requests.
# The standard web port is 80.

Config port             80

# https_port - the listening port for the server for HTTPS requests.
# The standard SSL port is 443.

Config https_port       443
```

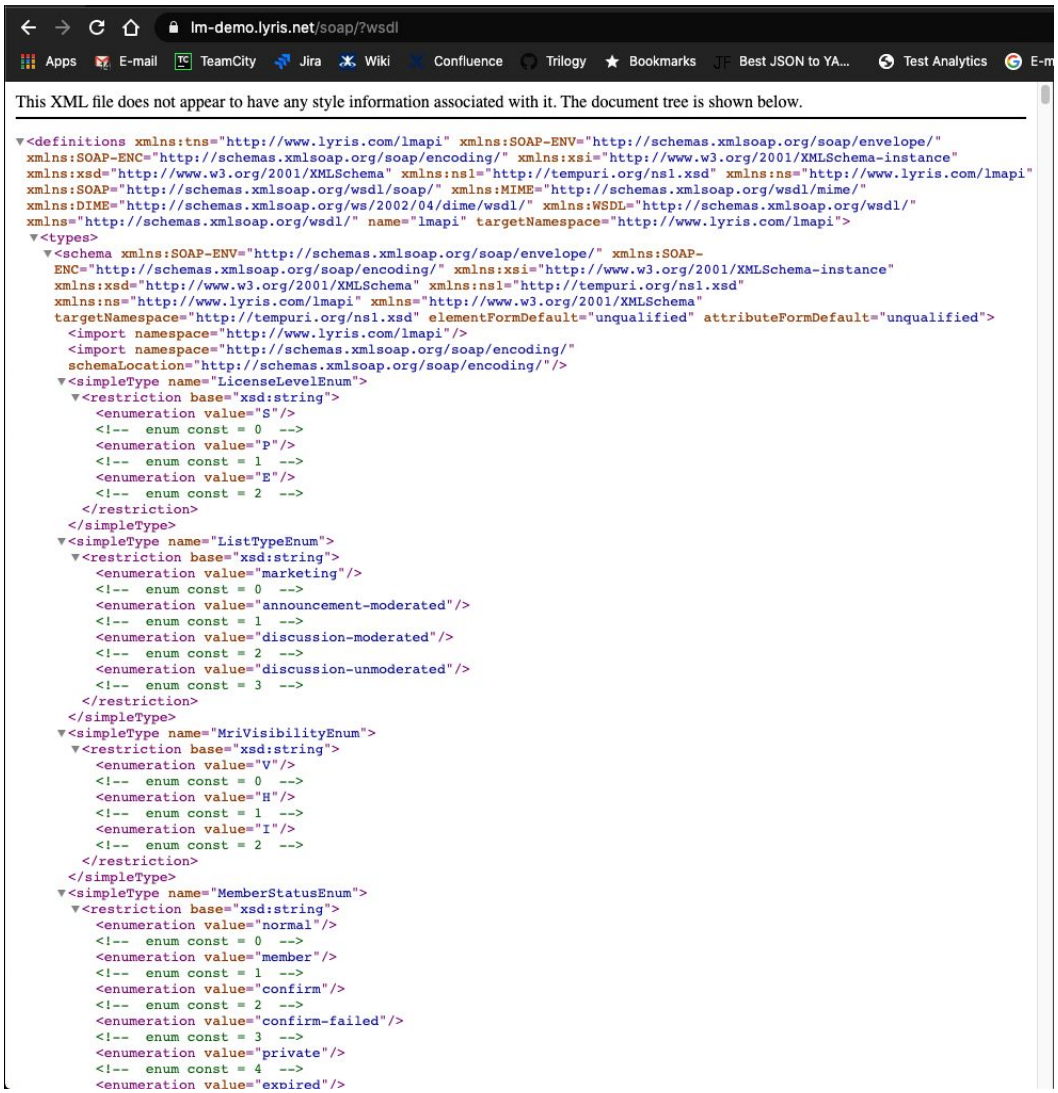
10. Switch all to `8080` and `8443`

```
# port - the listening port for the server for HTTP requests.
```

```
# The standard web port is 80.

Config port             8080

# https_port - the listening port for the server for HTTPS requests.
# The standard SSL port is 443.

Config https_port       8443
```

SOAP API

Unfortunately we can't assign a different port for SOAP API service so it always will run on port 82 (HTTP or HTTPS). It force us to use an path assignment for the SOAP service (**`/_soap/`**) like



To enable it you just have to add a new entry above the "**`location / {`**" block to map it to the LM SOAP API interface.

```
    location /_soap {
        proxy_set_header Host $host;
        proxy_cache_valid 200 7d;
        proxy_cache_use_stale error timeout invalid_header updating
http_500 http_502 http_503 http_504;
        proxy_set_header X-Forwarded-Proto https;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_redirect off;

        proxy_pass http://url.to.your.service:82/;
    }
```

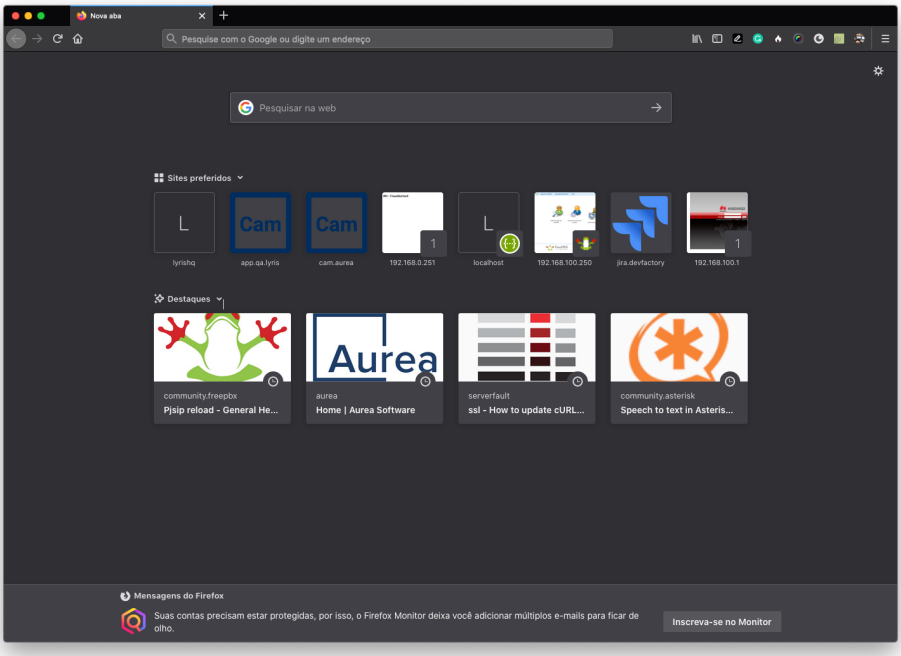
Then it will be available at https://url.to.your.service/_soap/?wsdl.

Don't forget to disable your firewall access from the Internet to the LM TCP 82 port (SOAP) and change your scripts to use the new URL based on SSL reverse proxy.

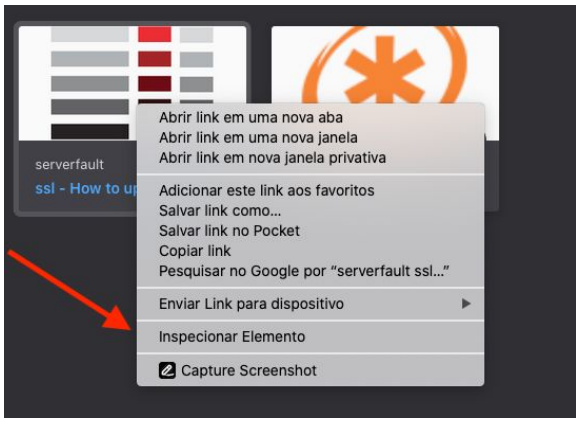
Tests

Application

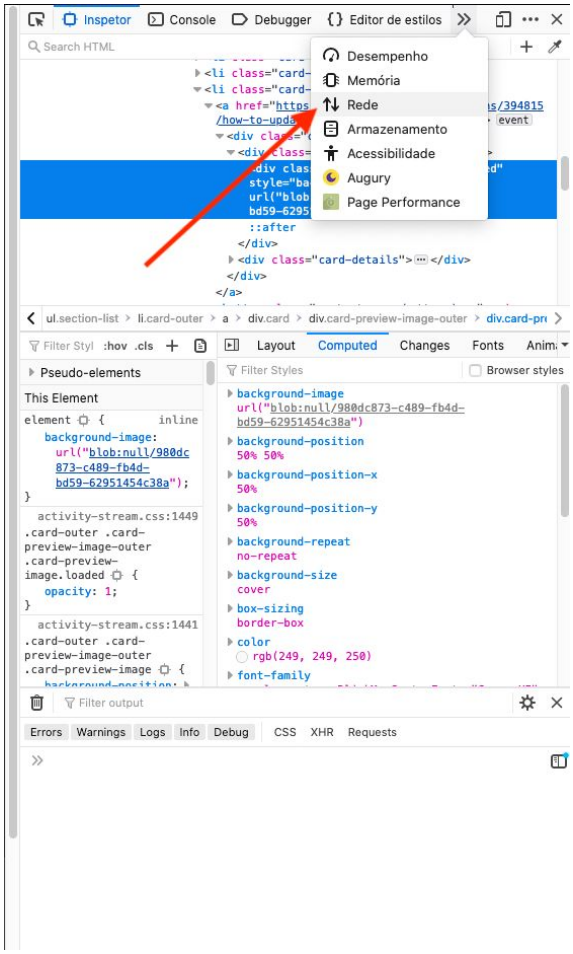
1. Open the Chrome/Firefox



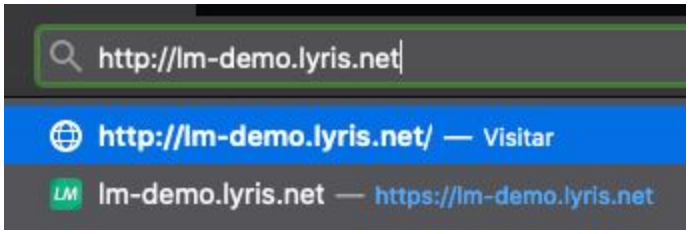
2. Open the Inspector



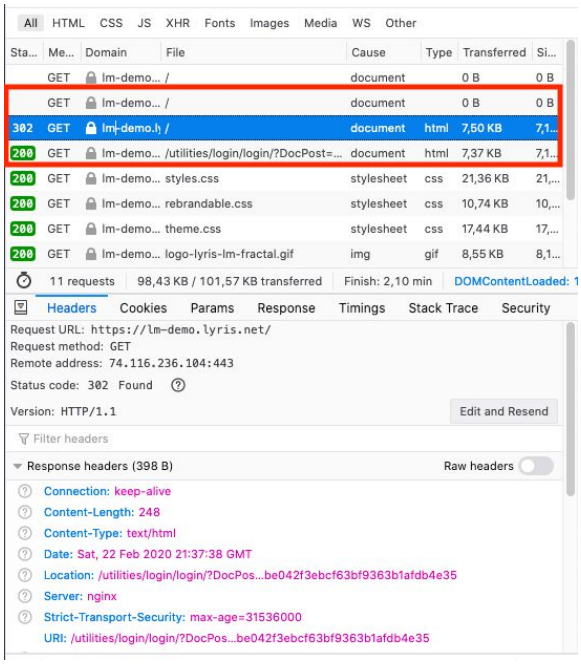
3. Open the Network Tab



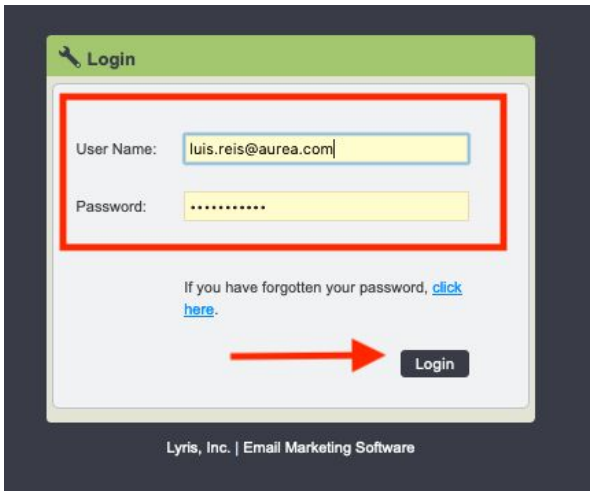
4. Into the URL bar, type the LM URL to be tested using the http:// at the beginning



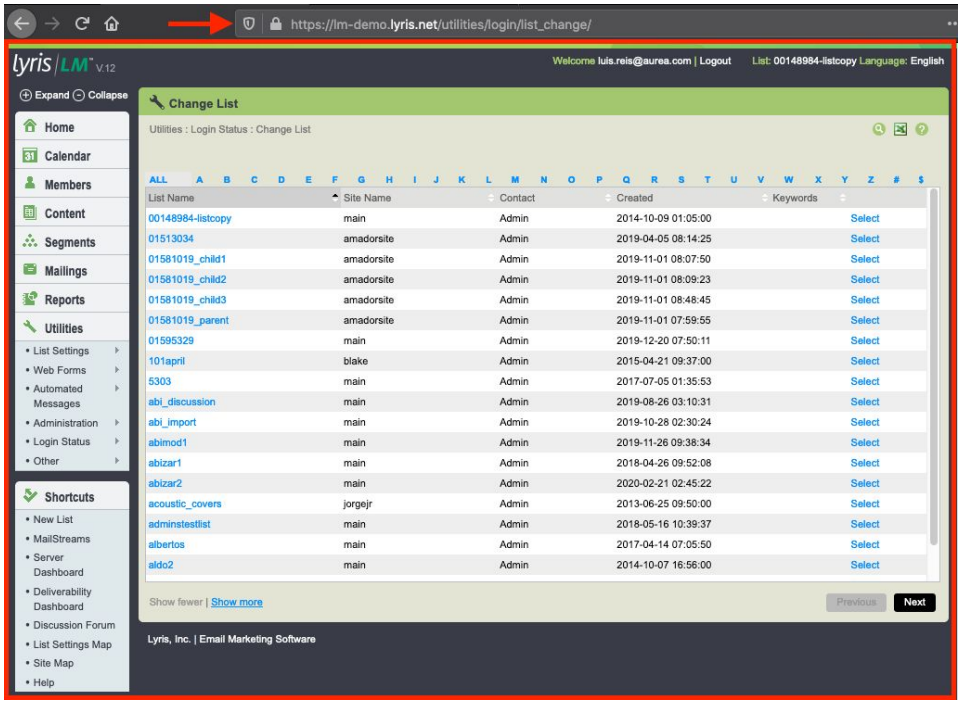
5. Check into the Network Tab if you got a 301 Redirect to the https:// url



6. Inform your login/password and click on the Login button



7. Check if the LM loaded without any alert at the URL bar



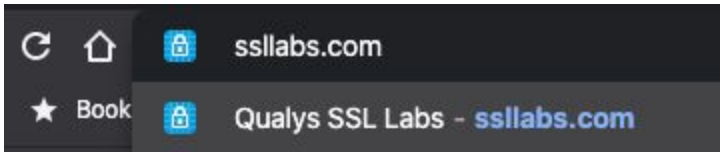
Rating

The Secure Sockets Layer (SSL) protocol is a standard for encrypted network communication. We feel that there is surprisingly little attention paid to how SSL is configured, given its widespread usage. SSL is relatively easy to use, but it does have its traps. This guide aims to establish a straightforward assessment methodology, allowing administrators to assess SSL server configuration confidently without the need to become SSL experts.

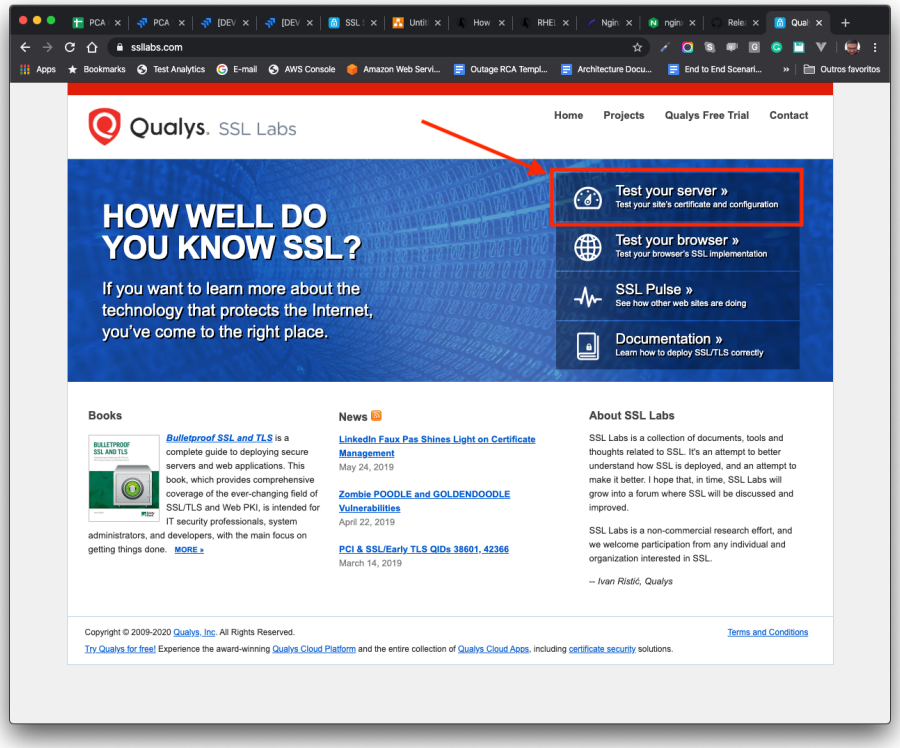
Complete Guide: [SSL Server Rating Guide]

Test

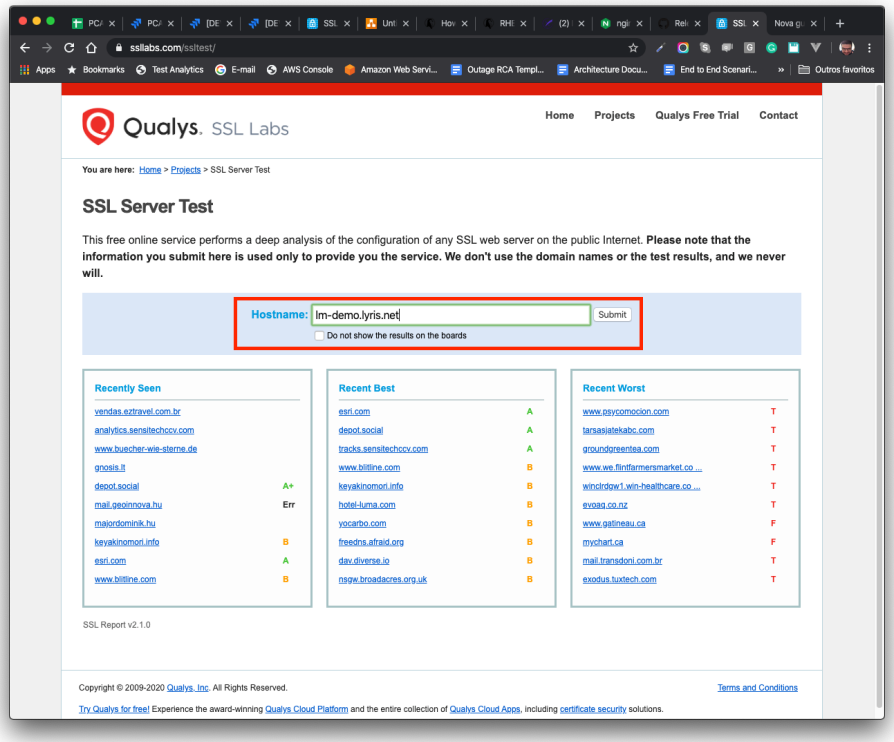
1. Open the SSL Labs website: [https://www.ssllabs.com]



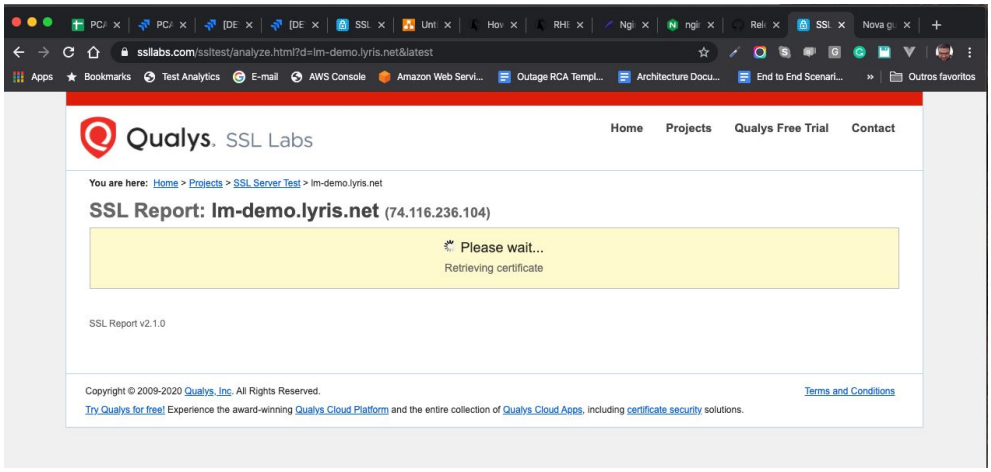
2. Click at "Test your server"



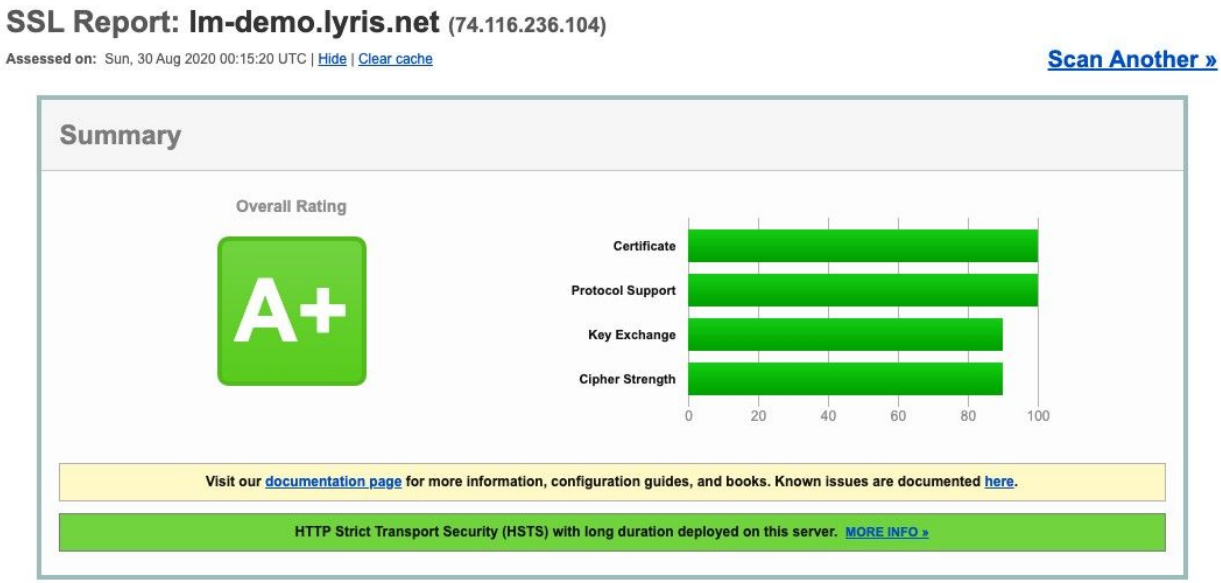
3. At "Hostname", type your URL



4. Wait for the results

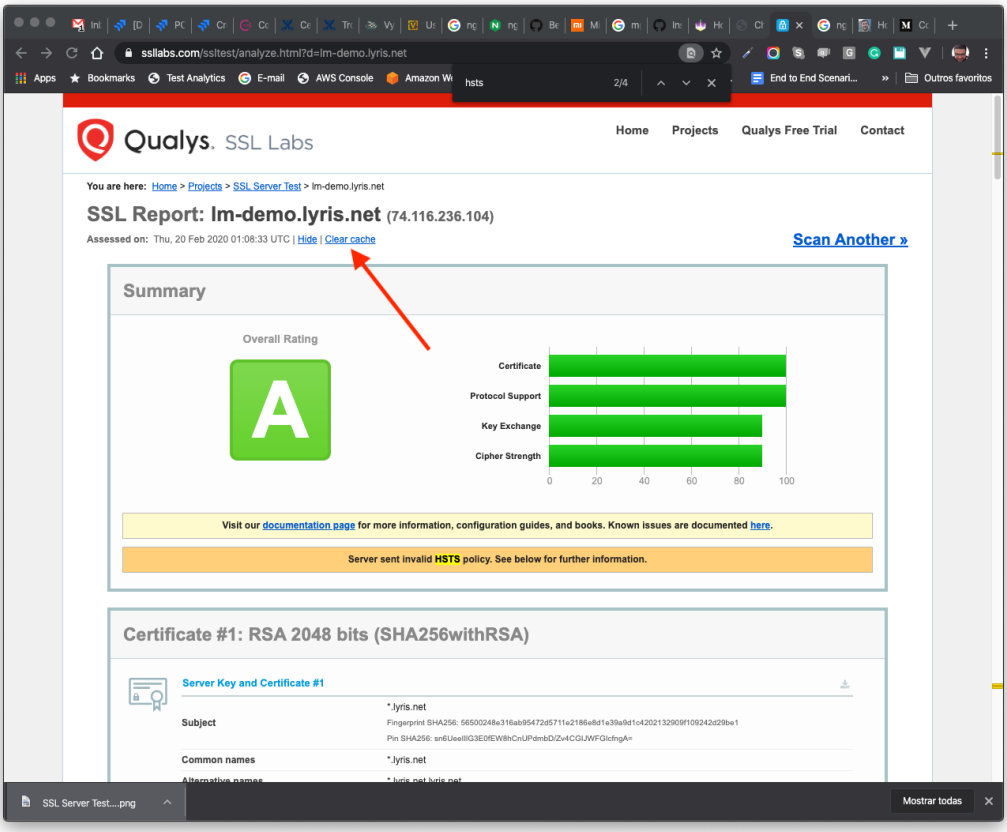


5. Check your rate



Invalidating the Test Cache

Just click at the "Clear Cache" link on the top of the page.



Example Configuration

Save these configurations into `/etc/nginx/conf.d/default.conf`, or at `C:\nginx\conf\nginx.conf` for Windows hosts - inside the **`http`** section, and check if it conflicts with any other existent configuration.

1. Please also replace all variables placeholders, like `%%NGINX_HTTP_PORT%%`, to your own values.
2. Please also fill the `%%NGINX_SSL_CERT%%` and `%%NGINX_SSL_KEYS%%` pointing to your **.CRT/.PEM** and **.KEY** SSL certificates using the **FULL PATH** notation
    a. In Linux use the normal notation like /etc/ssl/certificates.crt or /etc/ssl/certificates.key
    b. In Windows use the normal path location, just replacing slashes (/) by double backslashes (\\) like **`C:\\Windows`** instead of **`C:/Windows`**.
        i. **It must be done for all custom paths**

Single file

```
server {
```

```
    listen %%NGINX_HTTP_PORT%% default_server;
    listen [::]:%%NGINX_HTTP_PORT%% default_server;
    server_name localhost;
    return 301 https://$host$request_uri;

}

server {
    listen %%NGINX_HTTPS_PORT%% ssl http2;
    listen [::]:%%NGINX_HTTPS_PORT%% ssl http2;
    server_name localhost;

    ssl_certificate %%NGINX_SSL_CERT%%;
    ssl_certificate_key %%NGINX_SSL_KEYS%%;

    # enable session resumption to improve https performance
    # http://vincent.bernat.im/en/blog/2011-ssl-session-reuse-rfc5077.html
    ssl_session_cache shared:SSL:50m;
    ssl_session_timeout 1d;
    ssl_session_tickets off;

    # Diffie-Hellman parameter for DHE ciphersuites, recommended 4096 bits
    ssl_dhparam /etc/nginx/conf.d/dhparam.pem;

    # enables server-side protection from BEAST attacks
    # http://blog.ivanristic.com/2013/09/is-beast-still-a-threat.html
    ssl_prefer_server_ciphers on;

    # disable SSLv3(enabled by default since nginx 0.8.19) since it's less secure then
TLS http://en.wikipedia.org/wiki/Secure_Sockets_Layer#SSL_3.0
    #ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3;
    ssl_protocols TLSv1.2 TLSv1.3;

    # ciphers chosen for forward secrecy and compatibility
    #
http://blog.ivanristic.com/2013/08/configuring-apache-nginx-and-openssl-for-forward-se
crecy.html
    ssl_ciphers
'ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA2
56:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA3
84:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE
-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA
:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA:ECDHE-RSA-AES256-SHA:DHE-RSA-AES128-
SHA256:DHE-RSA-AES128-SHA:DHE-RSA-AES256-SHA256:DHE-RSA-AES256-SHA:ECDHE-ECDSA-DES-CBC
3-SHA:ECDHE-RSA-DES-CBC3-SHA:EDH-RSA-DES-CBC3-SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:
AES128-SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:DES-CBC3-SHA:!DSS';

    # enable ocsp stapling (mechanism by which a site can convey certificate
revocation information to visitors in a privacy-preserving, scalable manner)
```

```
    # http://blog.mozilla.org/security/2013/07/29/ocsp-stapling-in-firefox/
    resolver 8.8.8.8 8.8.4.4;
    ssl_stapling on;
    ssl_stapling_verify on;
    ssl_trusted_certificate %%NGINX_SSL_CERT%%;

    # config to enable HSTS(HTTP Strict Transport Security)
https://developer.mozilla.org/en-US/docs/Security/HTTP_Strict_Transport_Security
    # to avoid ssl stripping https://en.wikipedia.org/wiki/SSL_stripping#SSL_stripping
    # also https://hstspreload.org/
    # comment this out if your backend service doesn't add this header
    # add_header Strict-Transport-Security "max-age=31536000; includeSubdomains;
preload";

    # ... the rest of your configuration
    location / {
        proxy_set_header Host $host;
        proxy_cache_valid 200 7d;
        proxy_cache_use_stale error timeout invalid_header updating http_500 http_502
http_503 http_504;
        proxy_set_header X-Forwarded-Proto https;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_redirect off;

        proxy_pass http://%%LOCAL_IP%%:%%HTTP_PORT%%;
    }

    location /soap {
        proxy_set_header Host $host;
        proxy_cache_valid 200 7d;
        proxy_cache_use_stale error timeout invalid_header updating http_500 http_502
http_503 http_504;
        proxy_set_header X-Forwarded-Proto https;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_redirect off;

        proxy_pass https://%%LOCAL_IP%%:%%SOAP_PORT%%/;
    }

}
```