

Configuraciones básicas de seguridad para Nginx y Apache

Andrés Guerrero Pinteño
Jose Carlos Baena Ariza

Índice del usuario

Nginx.....	3
Control de acceso HTTP.....	3
Ocultar la versión del servidor.....	4
Establecer límites para Buffer y tiempos de espera.....	5
Limitar el número de conexiones por IP.....	5
Configuración de seguridad para PHP.....	6
Cómo crear un certificado SSL (activar https) en Nginx.....	7
Crear el certificado SSL.....	7
Configurar Nginx para utilizar SSL.....	8
Apache.....	11
Ocultar la firma y versión del servidor.....	11
Deshabilitar el HTTP Trace.....	12
Defensa contra ataques SQL Injection.....	12
Configuración de ModSecurity.....	14
Bibliografía.....	16

En este documento vamos a configurar distintos tipos de seguridad que podemos aplicar a los servidores web nginx y apache. Iremos paso a paso en cada configuración, comenzando primero con la configuración del servidor web nginx y posteriormente con la configuración del servidor web apache.

1.Nginx.

1.1.Control de acceso HTTP.

La seguridad en nuestro servidor web es algo primordial y debemos tener en cuenta que directorios queremos que nuestro visitantes puedan visitar, para ello una práctica muy común es la de limitar el acceso a algunas de nuestras URLs mediante usuario y contraseña.

Este tipo de protección es necesaria cuando tenemos administradores web para algunas secciones que son de alto riesgo, como por ejemplo el administrador de base de datos de nuestro sitio web.

El control de acceso HTTP consiste en implementar un proceso de autenticación para el acceso a algunos archivos y carpetas, esto nos ayuda a proteger los distintos tipos de recursos que disponemos. Es fácil y rápido de implementar y está basado en cabeceras HTTP.

Los pasos a seguir para crear un control de acceso son los siguientes:

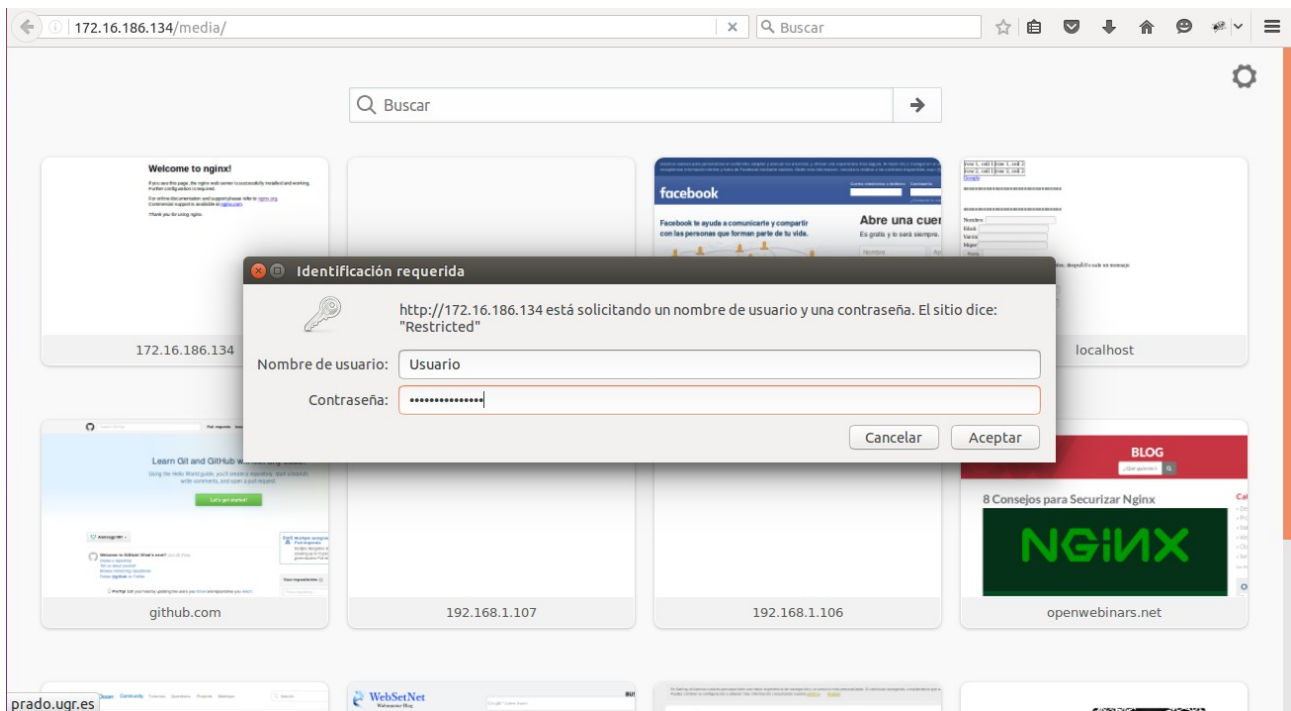
- 1- Vamos a generar el usuario y la contraseña que deseamos para acceder a la URL, para ello usaremos la librería Apache utils con el comando `htpasswd`. Este comando nos permite crear un archivo donde almacenaremos el usuario y la contraseña.

```
root@Usuario:/# htpasswd -c /etc/nginx/user_auth Usuario
New password:
Re-type new password:
Adding password for user Usuario
root@Usuario:/#
```

- 2- Una vez hecho el archivo con el usuario y la contraseña, tenemos que indicarle a Nginx que debe utilizar dicho archivo para proteger la ubicación que hayamos decidido, esto agregará una interfaz nueva en el front end del site que protegerá nuestros recursos, veamos en la siguiente imagen una porción del archivo de configuración donde aplicamos esto.

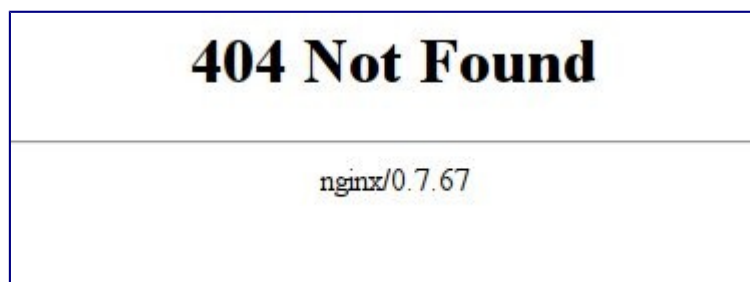
```
location /media {
    try_files $uri $uri/ =404;
    auth_basic "Restricted";
    auth_basic_user_file /etc/nginx/user_auth;
}
```

Aquí vemos como en el bloque location, le indicamos que para acceder a la raíz del site debe aplicarse la directiva `auth_basic` y le indicamos la ruta del archivo que hemos generado en el paso 1, ya con ello tendremos nuestra capa de seguridad establecida.



1.2. Ocultar la versión del servidor.

Cuando intentamos entrar a un directorio inexistente, una página no permitida o un error de servidor, tenemos los típicos errores 404, 403 o 500 respectivamente:



Como vemos, se muestra la versión del servidor, lo cual puede ser útil para algún atacante, pues puede encontrar vulnerabilidades específicas para la versión en cuestión.

Para solucionar esto, debemos modificar el archivo de configuración de nginx */etc/nginx/nginx.conf* y añadir la siguiente línea en la sección *http*:

server_tokens off;

Reiniciamos/recargamos el servidor web, y obtenemos lo siguiente:



1.3. Establecer límites para Buffer y tiempos de espera

Ahora vamos a configurar nginx para evitar ataque de desbordamiento de Buffer para ello creamos el archivo de configuración **/etc/nginx/conf.d/bufer.conf** y limitamos la configuración del Buffer para todos los clientes de la siguiente forma:

client_body_buffer_size 1K;

client_header_buffer_size 1k;

client_max_body_size 1k;

large_client_header_buffers 2 1k;

- **Client_body_buffer_size 1K** (Por defecto es 8k o 16k). Esta directiva especifica la solicitud del cliente respecto al tamaño del bufer en el cuerpo del mensaje.
- **Client_header_buffer_size 1k** – Esta directiva establece el tamaño headerbuffer para el encabezado de la solicitud del cliente. Para la inmensa mayoría de las solicitudes de un tamaño de búfer de 1K es suficiente. Aumente este si tiene un encabezado personalizado o una cookie enviado desde el cliente.
- **Client_max_body_size 1k**– Asigna tamaño de cuerpo máximo de solicitud de cliente , indicado por la línea Content-Length en el encabezado de la solicitud . Si el tamaño es mayor al que se da, entonces el cliente obtiene el error ” Entidad de solicitud demasiado grande” (413) . Se puede aumentar en caso de POST grandes.
- **Large_client_header_buffers 2 1k** – Asigna el número y el tamaño máximo de buffers para grandes cabeceras para leer desde la solicitud del cliente. Por defecto, el tamaño de un búfer es igual al tamaño de la página , según la plataforma , ya sea este de 4K u 8K , Esto también ayudará a combatir los bots malignos y ataques de denegación de servicio.

También es necesario controlar los tiempos de espera para mejorar el rendimiento del servidor y desconectar clientes:

client_body_timeout 10

client_header_timeout 10;

keepalive_timeout 5 5;

send_timeout 10;

- **client_body_timeout 10;** – Directiva establece el tiempo de espera leer el cuerpo de la solicitud del cliente. El tiempo de espera se establece sólo si un cuerpo no es conseguir en un readstep. Si después de este tiempo el cliente envia nada, Nginx devuelve error “Solicitud de tiempo muerto” (408). El valor predeterminado es 60.
- **client_header_timeout 10;** – Directiva asigna tiempo de espera con la lectura del título de la petición del cliente. El tiempo de espera se establece sólo si un encabezado no es conseguir en un readstep. Si después de este tiempo el cliente envia nada, Nginx devuelve error “Solicitud de tiempo muerto” (408).

- **keepalive_timeout 5 5;** – El primer parámetro asigna el tiempo de espera para las conexiones keep-alive con el cliente. El servidor cerrará las conexiones después de este tiempo. El segundo parámetro opcional asigna el valor del tiempo en la cabecera del Keep-Alive: timeout = tiempo de la respuesta. Este cabezal puede convencer a algunos navegadores para cerrar la conexión, para que el servidor no tiene que. Sin este parámetro, Nginx no envía un encabezado Keep-Alive (Aunque esto no es lo que hace una conexión “keepalive”).
- **send_timeout 10;** – Directiva asigna tiempo de espera de respuesta al cliente. Tiempo de espera se establece no en toda transferencia de respuesta, Pero sólo entre dos operaciones de lectura, Si después de este tiempo cliente llevará nada, Entonces nginx está cerrando la conexión.

1.4. Limitar el número de conexiones por IP.

Para controlar las conexiones de cliente en nginx se utiliza el modulo `ngxhttplimitzone` que limita el numero de conexiones simultaneas. Esta configuración ayuda a evitar ataques de denegación de servicio.

Para esto, haremos uso de la línea con el parámetro ‘`limit_conn_zone`’ (fuera del bloque de servidor del archivo de configuración) o la que contiene ‘`limit_conn`’ (dentro del bloque servidor en el archivo de configuración).

```
limit_conn_zone $binary_remote_addr zone=addr:5m
```

```
limit_conn addr 1;
```

En este caso, hemos establecido que no se podrá hacer más de 1 conexiones por IP.

- **Limit_zone slimits:** describe la zona en la cual los estados de la sesión son almacenados. 1m puede manejar 32000 sesiones con 32 bytes/sesión, se establece en 5 x 32000 sesión.
- **Limit_conn slimits:** control del número máximo de conexiones simultáneas para una sesión, es decir, restringe la cantidad de conexiones de una sola dirección IP.

```
limit_conn_zone $binary_remote_addr zone=alpha:5m;

server {
    limit_conn alpha 1;
    listen 80 default_server;
    listen [::]:80 default_server;
```

Utilizamos el comando `siege` para enviar peticiones simultaneas al servidor y obtenemos el siguiente resultado en el cual han fallado bastantes peticiones al servidor y no ha podido completar el tiempo de ejecución.

```
siege -b -t60S -v http://172.16.186.134/
```

```
Transactions:          66395 hits
Availability:          98.48 %
Elapsed time:          14.79 secs
Data transferred:     24.52 MB
Response time:         0.00 secs
Transaction rate:      4489.18 trans/sec
Throughput:            1.66 MB/sec
Concurrency:           14.91
Successful transactions: 66395
Failed transactions:    1024
Longest transaction:    0.02
Shortest transaction:   0.00

FILE: /var/log/siege.log
You can disable this annoying message by editing
the .siegerc file in your home directory; change
the directive 'show-logfile' to false.
[error] unable to create log file: /var/log/siege.log: Permission denied
```

1.5. Configuración de seguridad para PHP.

Para poder mejorar la seguridad de nuestro servidor, debemos mejorar también la seguridad de PHP. Para ello vamos a editar el archivo **/etc/php.ini** como sigue:

No permitir las funciones peligrosas

disable_functions = phpinfo, sistema, correo, Exec

Tiempo máximo de ejecución de cada script, en segundos

max_execution_time = 30

Cantidad máxima de tiempo que cada script puede pasar análisis de petición de datos

max_input_time = 60

Cantidad máxima de memoria que puede consumir un script (8MB)

memory_limit = 8M

Tamaño máximo de datos puesto que PHP aceptará.

post_max_size = 8M

Si va a permitir HTTP file uploads.

file_uploads = Apagado

Máximo permitido de tamaño para archivos subidos.

upload_max_filesize = 2M

No exponga los mensajes de error PHP a usuarios externos

display_errors = Apagado

Active el modo seguro

modo seguro = En

Sólo permiten el acceso a archivos ejecutables en el directorio aislado

safe_mode_exec_dir = php-necesaria-ejecutables-trayectoria

```
# Limitar el acceso externo al entorno de PHP
safe_mode_allowed_env_vars = PHP_

# Restringir la fuga de información de PHP
expose_php = Apagado

# Registrar todos los errores
log_errors = En

# No se registran globals para entrada de datos
register_globals = Apagado

# Minimizar el tamaño admisible de correo PHP
post_max_size = 1K

# Asegúrese de que php redirige apropiadamente
cgi.force_redirect = 0

# No permitir subir a menos necesario
file_uploads = Apagado

# Evite abrir archivos remotos
allow_url_fopen = Apagado
```

1.6. Cómo crear un certificado SSL (activar https) en Nginx

TLS, o la seguridad de capa de transporte, y su predecesor **SSL**, siglas de Secure Sockets Layer, son protocolos web utilizan para envolver el tráfico normal en una protegida, envoltura de cifrado.

Utilizando esta tecnología, los servidores pueden enviar tráfico de forma segura entre el servidor y el cliente sin la preocupación de que los mensajes serán interceptados y leídos por un tercero. El sistema de certificados también ayuda a los usuarios en la verificación de la identidad de los sitios que se conectan con.

1.6.1. Crear el certificado SSL

Podemos empezar por la creación de un directorio que se utiliza para mantener toda nuestra información de SSL. Debemos crear esta bajo el directorio de configuración de Nginx:

```
sudo mkdir /etc/nginx/ssl
```

Ahora que tenemos un lugar para colocar nuestros archivos, podemos crear los archivos de claves y certificados SSL en un solo movimiento, escribiendo:

```
sudo openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout  
/etc/nginx/ssl/nginx.key -out /etc/nginx/ssl/nginx.crt
```

Se le pedirá una serie de preguntas. Antes de que vayamos más de eso, vamos a echar un vistazo a lo que está sucediendo en el comando que estamos emitiendo:

- **openssl:** Se trata de la herramienta de línea de comandos básicos para crear y gestionar certificados de OpenSSL, llaves y otros archivos.
- **req:** Este subcomando especifica que queremos usar solicitud de certificado de firma X.509 gestión (RSE). El "X.509" es un estándar de la infraestructura de clave pública que SSL y TLS se adhiere a por su llave y gestión de certificados. Queremos crear un nuevo certificado X.509, por lo que estamos utilizando este subcomando.
- **-x509:** Esto modifica aún más el subcomando anterior diciendo la utilidad que queremos hacer un certificado autofirmado en lugar de generar una solicitud de firma de certificado, como normalmente sucedería.
- **-nodes:** Esto le dice OpenSSL para saltarse la opción de asegurar nuestro certificado con una frase de contraseña. Necesitamos Nginx para poder leer el archivo, sin intervención del usuario, cuando el servidor se inicia. Una frase de paso sería evitar que esto suceda, porque tendríamos que entrar en él después de cada reinicio.
- **-days 365:** Esta opción establece la cantidad de tiempo que el certificado se considerará válido. Nos pusimos por un año aquí.
- **rsa -newkey: 2048:** Esto especifica que queremos generar un nuevo certificado y una nueva clave al mismo tiempo. No hemos creado la clave que se requiere para firmar el certificado en un paso anterior, por lo que necesitamos para crear junto con el certificado. El `rsa:2048` parte dice que para hacer una clave RSA que es de 2048 bits de longitud.
- **-keyout:** Esta línea le dice OpenSSL dónde colocar el archivo de clave privada generada que estamos creando.
- **salida privado:** Esto le dice OpenSSL dónde colocar el certificado que estamos creando.

```

usuario@Usuario:~$ sudo openssl req -x509 -nodes -days 1095 -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]:ES
State or Province Name (full name) [Some-State]:Granada
Locality Name (eg, city) []:Granada
Organization Name (eg, company) [Internet Widgits Pty Ltd]:UGR
Organizational Unit Name (eg, section) []:ETSIIT
Common Name (e.g. server FQDN or YOUR name) []:dominio.com
Email Address []:admin@dominio.com

```

1.6.2. Configurar Nginx para utilizar SSL

Hemos creado nuestros archivos de claves y certificados bajo el directorio de configuración de Nginx. Ahora sólo tenemos que modificar nuestra configuración del archivo `/etc/nginx/sites-enabled/default` en la sección `server` para aprovecharse de ellos ajustando nuestros archivos de bloques de servidor. Lo único que tendríamos que hacer para conseguir trabajo SSL en este mismo

bloque de servidor, al tiempo que permite conexiones HTTP regulares, es añadir una de estas líneas como vemos en la figura:

```
#      SSL configuration

      listen 443 ssl;
      ssl_certificate /etc/nginx/ssl/nginx.crt;
      ssl_certificate_key /etc/nginx/ssl/nginx.key;

      server_name dominio.com;
```

```
limit_conn_zone $binary_remote_addr zone=alpha:5m;

server {
    limit_conn alpha 2;
    listen 80 default_server;
    listen [::]:80 default_server;

#      SSL configuration

      listen 443 ssl;
      ssl_certificate /etc/nginx/ssl/nginx.crt;
      ssl_certificate_key /etc/nginx/ssl/nginx.key;

      server_name dominio.com;

      root /var/www/html;

      # Add index.php to the list if you are using PHP
      index index.html index.htm index.nginx-debian.html;

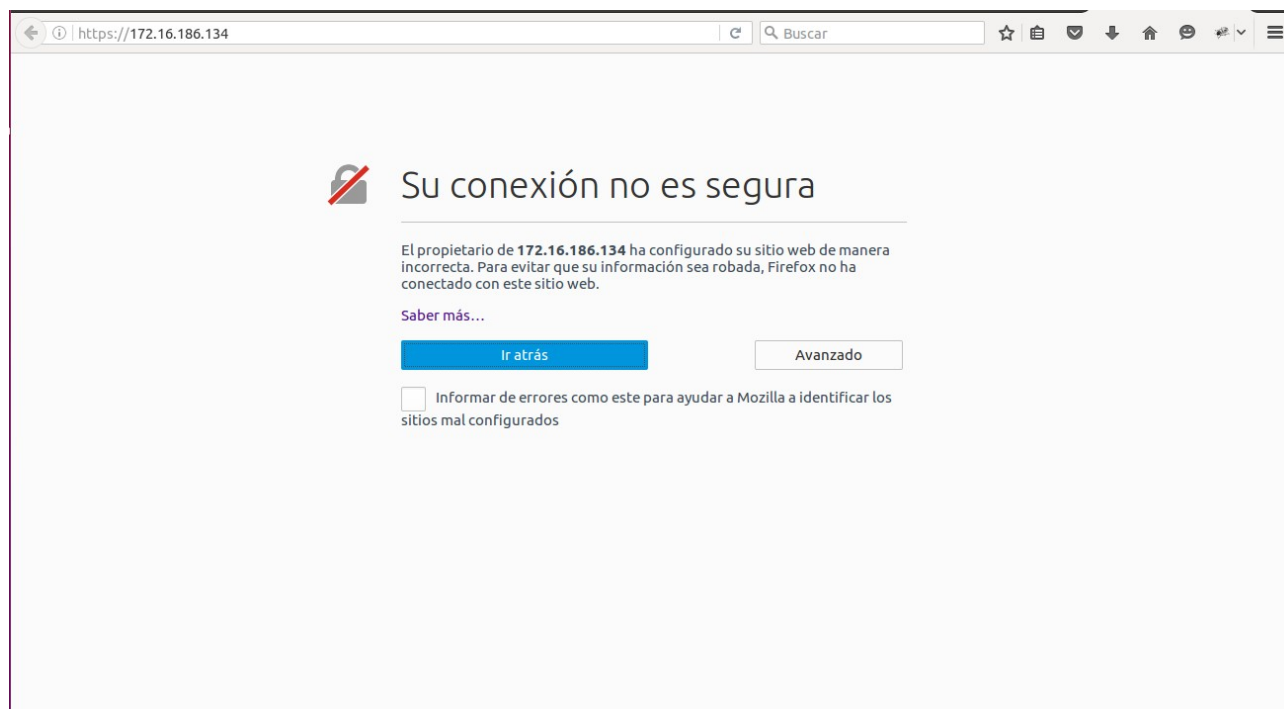
      location / {
          # First attempt to serve request as file, then
          # as directory, then fall back to displaying a 404.
          try_files $uri $uri/ =404;
      }

      location /media {
          try_files $uri $uri/ =404;
          auth_basic "Restricted";
          auth_basic_user_file /etc/nginx/user_auth;
      }
}
```

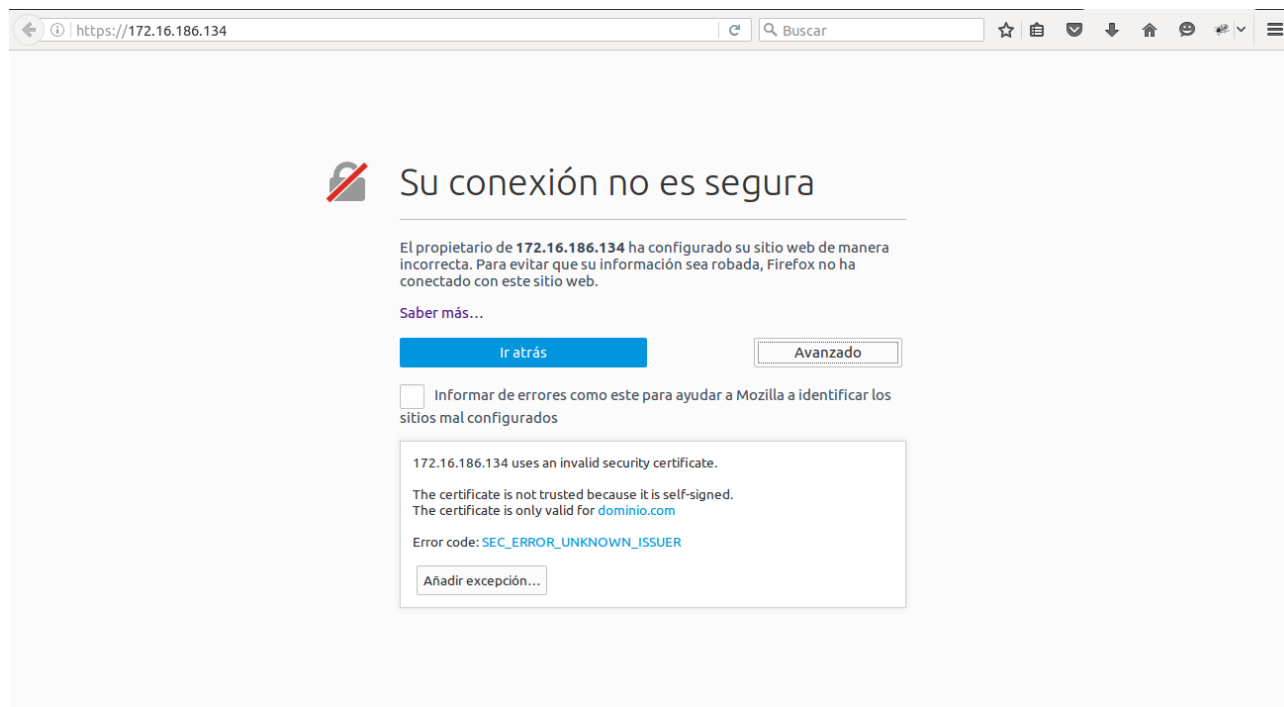
Cuando haya terminado, guarde y cierre el archivo. Ahora, todo lo que tienes que hacer es reiniciar Nginx para usar la nueva configuración:

service nginx restart

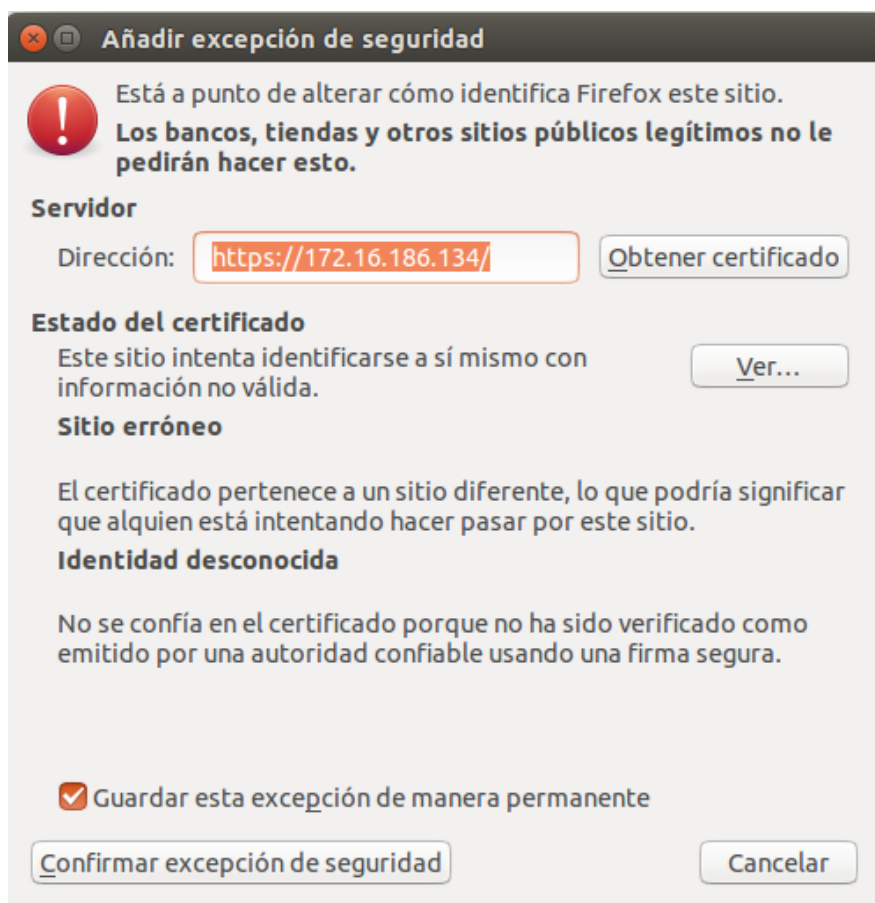
Ahora, podemos comprobar si nuestro servidor puede utilizar SSL para comunicarse. Haga esto especificando el https en vez del protocolo http protocolo. Usted probablemente recibirá una advertencia en su navegador web que se ve algo como esto:



Hacemos click en el botón “Avanzado” y nos saldrá un botón para añadir excepción debido a que no tenemos el certificado requerido por el servidor:



Lo Ultimo que tendremos que hacer será confirmar la excepción de seguridad para obtener el certificado del sitio web:



2. Apache.

2.1. Ocultar la firma y versión del servidor.

Como ya mencionamos en el mismo apartado de nginx, la firma y la versión de nuestro servidor son informaciones valiosas para cualquier atacante, por lo tanto, vamos a realizar la misma práctica de seguridad con apache.

Para deshabilitar la firma y la versión debemos acceder al archivo de configuración **etc/apache2/conf-enabled/security.conf**, buscar la variable 'ServerSignature' y ponerla en off.

```
# Optionally add a line containing the server version and virtual host
# name to server-generated pages (internal error documents, FTP directory
# listings, mod_status and mod_info output etc., but not CGI generated
# documents or custom error documents).
# Set to "EMail" to also include a mailto: link to the ServerAdmin.
# Set to one of: On | Off | EMail
ServerSignature Off
#ServerSignature On
```

También debemos cambiar la variable 'ServerTokens' a prod.

```
# ServerTokens
# This directive configures what you return as the Server HTTP response
# Header. The default is 'Full' which sends information about the OS-Type
# and compiled in modules.
# Set to one of: Full | OS | Minimal | Minor | Major | Prod
# where Full conveys the most information, and Prod the least.
#ServerTokens Minimal
#ServerTokens OS
#ServerTokens Full
ServerTokens Prod
```

2.2. Deshabilitar el HTTP Trace.

El HTTP Trace se usa para devolver la información recibida. Puede ser modificado para que devuelva cookies HTTP, dando la posibilidad a que nos roben la sesión HTTP. También puede ser usado para ataques de Cross Site Scripting. Por lo tanto vamos a desactivarlo.

Accedemos al archivo de configuración **etc/apache2/conf-enabled/security.conf** y cambiamos la variable 'TraceEnable' a off.

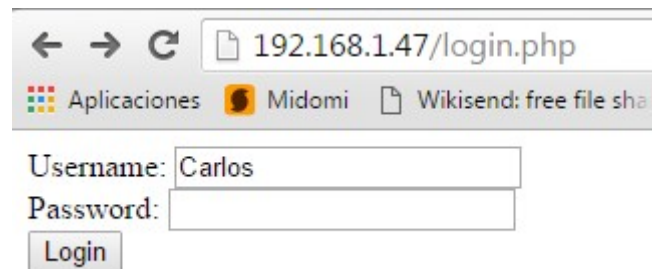
```
# Allow TRACE method
#
# Set to "extended" to also reflect the request body (only for testing and
# diagnostic purposes).
#
# Set to one of: On | Off | extended
TraceEnable Off
#TraceEnable On_
```

2.3. Defensa contra ataques SQL Injection.

Para empezar vamos a crear un script en php que será un simple formulario con login y contraseña que será vulnerable a un ataque por SQL Injection.

```
<html>
<body>
    <form action="" method="post">
        Username: <input type="text" name="username"/><br />
        Password: <input type="password" name="password"/><br />
        <input type="submit" name="login" Value="Login"/>
    </form>
<?php
    if(isset($_POST['login']))
    {
        $username = $_POST['username'];
        $password = $_POST['password'];
        $con = mysqli_connect('192.168.1.47', 'carlos', '1234', 'sample');
        $result = mysqli_query($con, "SELEC * FROM 'users'
            WHERE username='$username' AND password='$password'");
        if(mysqli_num_rows($result) == 0)
            echo 'Invalid username or password';
        else
            echo '<h1>Logged in</h1><p>A secret fo you....</p>';
    }
?>
</body>
</html>
```

Este script muestra en nuestro navegador dos campos con login y contraseña que podremos rellenar e interactuar con el.



← → ↻ 192.168.1.47/login.php

Aplicaciones Midomi Wikisend: free file sha

Username: Carlos

Password:

Login

A continuación, vamos a crear una base de datos para probar nuestro script. Para ello introducimos el comando 'mysql -u root -p' e introducimos la contraseña del usuario root que pusimos cuando instalamos sql.

Después, creamos la base de datos 'sample'.

```
mysql> create database sample;
Query OK, 1 row affected (0.04 sec)

mysql> connect sample;
Connection id: 38
Current database: sample
```

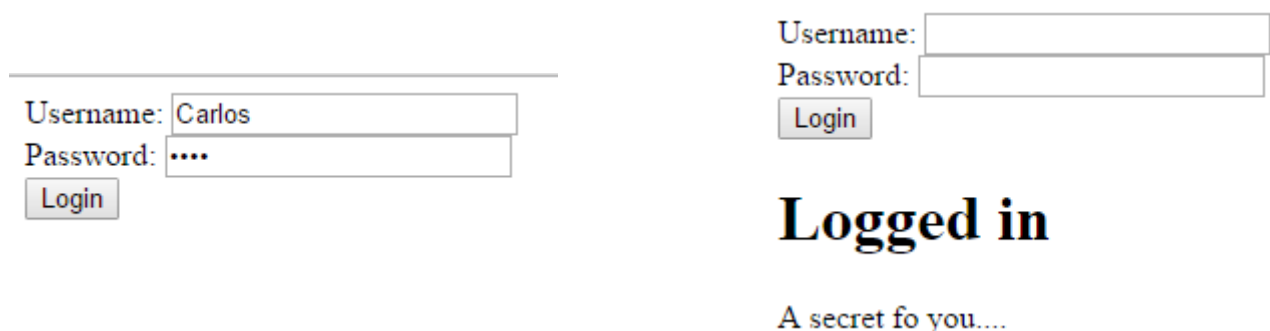
Creamos las tablas de nuestra base de datos e introducimos datos en ellas.

```
mysql> create table users(username VARCHAR(100),password VARCHAR(100));
Query OK, 0 rows affected (0.09 sec)

mysql> insert into users values('Carlos','1234');
Query OK, 1 row affected (0.06 sec)

mysql> insert into users values('Andres','0000');
Query OK, 1 row affected (0.06 sec)
```

Una vez creado todo lo relacionado con la base de datos, podemos probar nuestro script, utilizando un login y contraseña que hayamos añadido a la base de datos.



Username: Carlos

Password:

Login

Username:

Password:

Login

Logged in

A secret fo you....

A pesar de todo, nuestro script sigue siendo vulnerable a los ataques por SQL Injection, por lo tanto vamos a usar modsecurity, el cual vamos a explicar como usar en el siguiente apartado, para defendernos contra dichos ataques.

Añadimos la regla 'modsecurity_crs_41_sql_injection_attacks.conf' a la carpeta **usr/share/modsecurity-crs/activated_rules**.

```
root@ubuntu:/usr/share/modsecurity-crs/activated_rules# ln -s /usr/share/modsecurity-crs/base_rules/modsecurity_crs_41_sql_injection_attacks.conf
root@ubuntu:/usr/share/modsecurity-crs/activated_rules# ls
modsecurity_crs_41_sql_injection_attacks.conf  README
```

Una vez añadida y reiniciado el servicio de apache, podemos probar nuestro ataque por SQL Injection básico.

Username:

Password:

Login

Username:

Password:

Login

Invalid username or password

2.4. Configuración de ModSecurity.

En ModSecurity existen una gran cantidad de reglas que podemos activar para que nuestro servidor apache sea más seguro. Para poder utilizar dichas reglas primero debemos indicarles a apache donde encontrarlas.

Para ello necesitamos modificar el archivo de configuración **etc/apache2/mods-enabled/modsecurity.conf** y añadir las siguientes líneas.

```
<IfModule security2_module>
    # Default Debian dir for modsecurity's persistent data
    SecDataDir /var/cache/modsecurity

    Include "/usr/share/modsecurity-crs/*.conf"
    Include "/usr/share/modsecurity-crs/activated_rules/*.conf"

    # Include all the *.conf files in /etc/modsecurity.
    # Keeping your local configuration in that directory
    # will allow for an easy upgrade of THIS file and
    # make your life easier
    IncludeOptional /etc/modsecurity/*.conf
</IfModule>
```

```
root@ubuntu:/usr/share/modsecurity-crs# cd activated_rules/
root@ubuntu:/usr/share/modsecurity-crs/activated_rules# ln -s /usr/share/modsecurity-crs/experimental_rules/modsecurity_crs_11_brute_force.conf
root@ubuntu:/usr/share/modsecurity-crs/activated_rules# ln -s /usr/share/modsecurity-crs/experimental_rules/modsecurity_crs_11_dos_protection.conf
root@ubuntu:/usr/share/modsecurity-crs/activated_rules# ln -s /usr/share/modsecurity-crs/experimental_rules/modsecurity_crs_11_slow_dos_protection.conf
root@ubuntu:/usr/share/modsecurity-crs/activated_rules# ls
modsecurity_crs_11_brute_force.conf
modsecurity_crs_11_dos_protection.conf
modsecurity_crs_11_slow_dos_protection.conf
modsecurity_crs_23_request_limits.conf
modsecurity_crs_41_sql_injection_attacks.conf
README
```

Una vez hecho esto, ya podemos activar las reglas que veamos necesarias para nuestro servidor, en nuestro caso hemos activado las siguientes.

```
root@ubuntu:/usr/share/modsecurity-crs# cd activated_rules/
root@ubuntu:/usr/share/modsecurity-crs/activated_rules# ln -s /usr/share/modsecurity-crs/base_rules/modsecurity_crs_23_request_limits.conf
root@ubuntu:/usr/share/modsecurity-crs/activated_rules# ls
modsecurity_crs_23_request_limits.conf      README
modsecurity_crs_41_sql_injection_attacks.conf
```

Como podemos ver, usamos el comando 'ln -s /usr/share/modsecurity-crs/(carpeta donde se encuentra)/(regla que deseamos)'. Una vez añadidas las reglas que veamos convenientes tendremos que reiniciar apache para que se apliquen a nuestro servidor.

3. Bibliografía.

- [1] https://www.digitalocean.com/community/tutorials/how-to-set-up-mod_security-with-apache-on-debian-ubuntu
- [2] <https://www.jsitech.com/linux/recomendaciones-seguridad-servidor-web-apache/>
- [3] <http://www.solvetic.com/tutoriales/article/802-nginx-seguridad-adicional/>
- [4] <https://openwebinars.net/consejos-para-securizar-nginx/>
- [5] <http://alejovazquez.blogspot.com.es/2015/04/como-crear-un-certificado-ssl-activar.html>