FORTIFICANDO NGINX DE FORMA PROFESIONAL

Siguiendo las políticas CIS

JOSÉ MANUEL REDONDO LÓPEZ PROYECTO "*L-62 PRINCESA DE ASTURIAS*" v1.2




Fuente: Bing Chat AI

Departamento de Informática
Universidad de Oviedo

Universidad de Oviedo

Logo por: @creative_vanesa (Instagram)

¿QUÉ VAMOS A VER EN ESTA PRESENTACIÓN?

José Manuel
Redondo López

- En esta presentación veremos una serie de bloques
  - **La política de mejora de seguridad de Nginx del CIS** y por qué usar la misma
  - **Los conceptos básicos necesarios** para aplicar correctamente la política
  - **La lista de acciones implementables** que la política contempla
    - El documento de la política adjunta incluye información en profundidad de todas ellas: por qué son necesarias, como comprobarlas, como implementarlas…
  - **Acciones de administración adicionales**
    - Es una ampliación de la política propia de esta asignatura
    - Los contenidos para implementar las mismas se encuentran bien en el libro o en la propia presentación
    - Están enfocadas a dar servicios adicionales de forma correcta y segura
    - No forman parte de la política de seguridad, pero la complementan
- Todo el contenido para hacer los ejercicios se encuentra en esta presentación y en la política del CIS adjunta
  - No obstante, si surgen dudas se puede usar el siguiente libro online como referencia:
    - https://github.com/trimstray/nginx-admins-handbook

ÍNDICE

< Ir al Índice

Fuente: Bing Chat AI

Instalación >

Server blocks >

Contextos >

Conceptos básicos de Nginx

Antes de administrar...hay que conocer lo que se administra

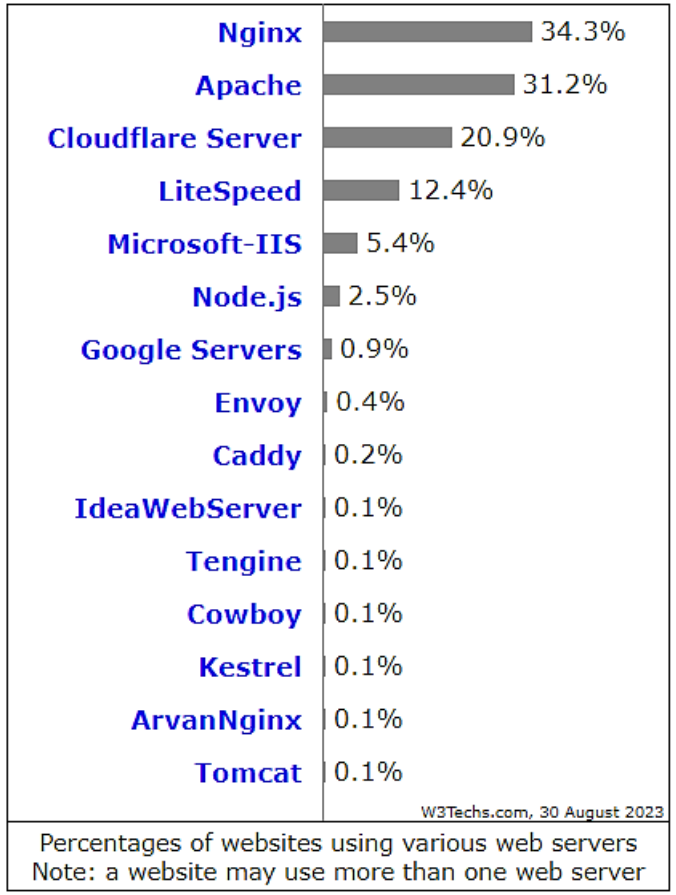
Departamento de Informática
Universidad de Oviedo

Universidad de Oviedo

¿Qué es Nginx?

José Manuel
Redondo López

- Es un servidor web que se ha convertido en uno de los más usados actualmente (el 2º)
- Algunas de las webs más grandes y con mayor tráfico de Internet usan **Nginx** como servidor
  - https://wappalyzer.com/applications/nginx
- Esto se debe a que normalmente consume menos recursos que **Apache 2**
- Además, es muy común usarlo como **reverse proxy** (de la misma forma que en **Apache 2**)
- Está disponible para **Linux** y para **Windows**
  - Antes de instalarlo, se debe configurar el SO base de forma segura (ya lo hemos visto)



¿Qué es Nginx?

José Manuel
Redondo López

- Hay dos versiones de **Nginx**, la versión **open source** (que veremos en esta presentación) y una de pago llamada **Nginx Plus**
- Ambas son idénticas, pero la versión de pago tiene funcionalidades adicionales, entre las que se encuentran, entre otras:
  - Balanceo de carga avanzado
  - Persistencia de sesiones
  - Integración con `mod_security`
  - Clustering activo/activo y activo/pasivo integrado
  - Control de salud de los servidores de un clúster
- **Las operaciones de clústering no están soportadas en la versión gratuita**
  - Si necesitamos un clúster de servidores, o bien pagamos la versión Plus o usamos Apache 2...
- **Más información**
  - https://www.nginx.com/products/nginx/compare-models/

¿Qué es Nginx?

José Manuel
Redondo López

- **Nginx tiene conceptos muy similares a Apache 2**
  - Saber manejar Apache 2 nos facilita la labor de trabajar con Nginx, dadas esas similitudes
- **Nos aprovecharemos de esto para hacer ciertas explicaciones sobre el servidor**
  - Elementos idénticos, elementos equivalentes...
- **Además, trataremos de cubrir los mismos casos usando, si es posible, los mismos productos o tecnologías**
- **Al acabar este tema, podréis administrar competentemente cualquier servidor web instalado en cerca del 90% de los servidores del mundo**
  - Apache 2 + Nginx
- **Esta presentación se concentra en distribuciones tipo Debian**
  - La política del CIS se centra en distribuciones tipo Red Hat
  - Casi todos los controles son los mismos o fácilmente traducibles

Instalación del servidor

Poniendo nuestro primer Nginx operativo

Instalación de Nginx

José Manuel
Redondo López

- Este servidor está disponible en los repositorios de software oficiales de **Ubuntu/Debian**, por lo que su instalación es bastante trivial
- Conviene hacer un `update` previo para instalar la versión más actual
  - `sudo apt-get update`
  - `sudo apt-get install nginx`
- A partir de ahora **Nginx** se actualizará con las actualizaciones del SO
- La instalación en distribuciones **Red Hat** es diferente
  - Está documentada en la política del CIS que usaremos luego

Configuración del firewall

José Manuel
Redondo López

- **Al instalarse Nginx, éste se registra como servicio en ufw, lo que permite configurarlo fácilmente**
  - Se habla de este firewall en la teoría de la asignatura
- **Lo primero es comprobar que se ha registrado:** `sudo ufw app list`
- **Al hacerlo veremos los siguientes perfiles asociados a Nginx**
  - `Nginx Full`: Abre los puertos 80 y 443 (TLS/SSL)
  - `Nginx HTTP`: Solo puerto 80
  - `Nginx HTTPS`: Solo puerto 443 (ideal para forzar el uso de https en todas las webs que alojemos)

Configuración del firewall

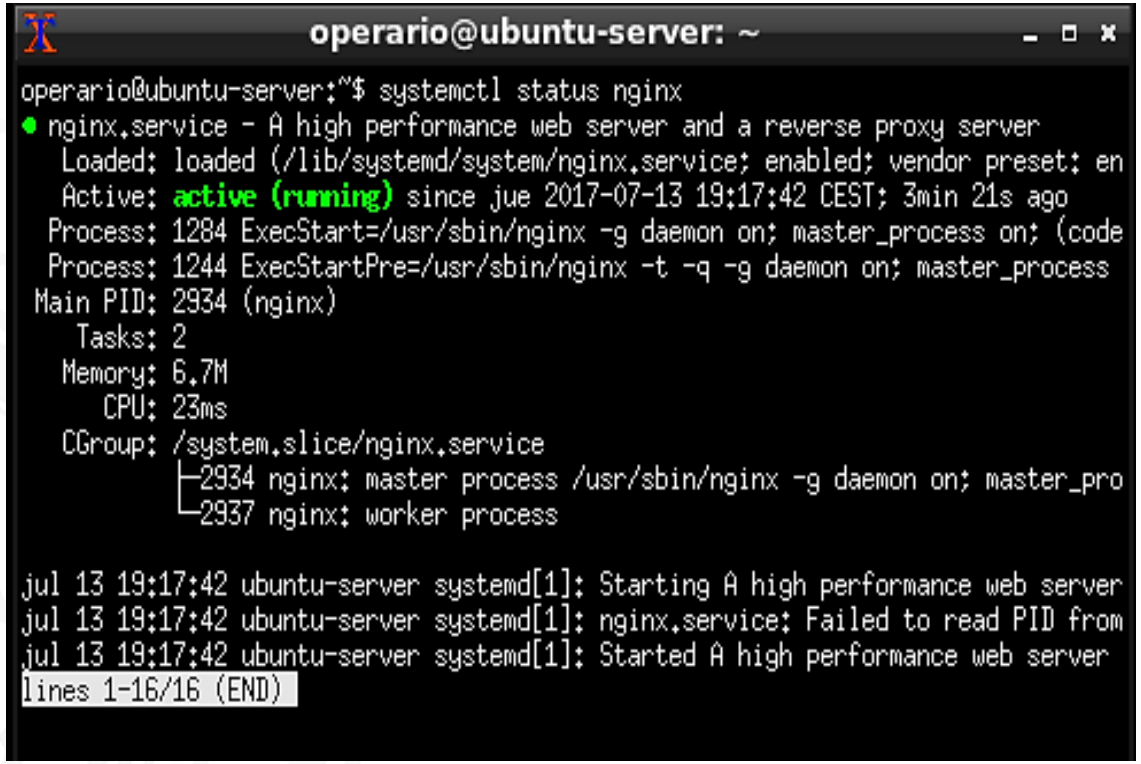
José Manuel
Redondo López

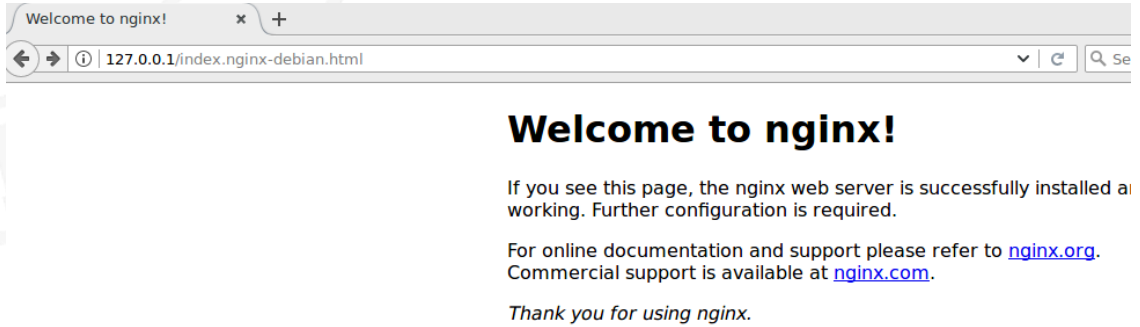
- Siempre debe activarse el perfil más restrictivo que nos dé lo que necesitemos
  - Como sabes, por temas de seguridad
- Para ello basta con hacer algo como
  - `sudo ufw allow 'Nginx HTTPS'` (en producción, en desarrollo puede usarse HTTP)
  - **Recuerda**: **¡NO se tienen servidores web en producción sin HTTPs!**
- Y comprobar que está correctamente configurado con: `sudo ufw status`

COMPROBACIÓN DE FUNCIONAMIENTO

José Manuel
Redondo López

- En este punto de la instalación, Nginx debería haberse auto-arrancado y estar en funcionamiento
- Al igual que con cualquier servicio, podemos comprobarlo con
  - `systemctl status nginx`
- Hecho esto, podemos acceder a la IP del servidor y ver la página por defecto de Nginx





Gestión del proceso Nginx

José Manuel
Redondo López

- **Nginx** es un proceso, por lo que se puede gestionar completamente, al igual que **Apache 2**, con `systemctl` (no sólo ver su status)
  - **Parar**: `sudo systemctl stop nginx`
  - **Arrancar**: `sudo systemctl start nginx`
  - **Reiniciar**: `sudo systemctl restart nginx`
  - **Cada cambio en la configuración de Nginx que veamos requiere una recarga** del servidor: `sudo systemctl reload nginx`
- Finalmente, aunque **Nginx** se configura para arrancar automáticamente en cada inicio, esto puede cambiarse mediante: `sudo systemctl disable nginx`
  - Si deseamos que el inicio sea automático de nuevo basta con hacer: `sudo systemctl enable nginx`

Ficheros importantes (Debian)

José Manuel
Redondo López

- **Al igual que con Apache, es importante tener claras una serie de localizaciones**
  - `/var/www/html`: Directorio donde están por defecto las webs que sirve. Es configurable
  - **Distintas partes de la configuración del servidor**
    - `/etc/nginx`: Carpeta donde se guarda la configuración de Nginx
    - `/etc/nginx/nginx.conf`: Fichero principal de configuración global (como `apache2.conf`)
    - `/etc/nginx/sites-available`: Contiene las diferentes configuraciones de las webs que se pueden servir, llamadas "server blocks" (similares a los hosts virtuales de Apache 2)
    - `/etc/nginx/sites-enabled`: Ficheros de configuración de server blocks en `sites-available` enlazados, representando las configuraciones de todas las webs que el servidor sirve realmente
    - `/etc/nginx/snippets`: Contiene ficheros con fragmentos de configuraciones que pueden ser usados en otras configuraciones, refactorizándolas
  - **Logs del servidor**
    - `/var/log/nginx/access.log`: Registro de cada acceso al servidor
    - `/var/log/nginx/error.log`: Registro de cada error que ocurra en el servidor

Ficheros importantes (Red Hat)

José Manuel
Redondo López

- **El ejecutable de Nginx en distribuciones Red Hat suele venir con todos los módulos compilados para que su rendimiento sea superior**
  - Eso quiere decir que **cada cambio en los módulos requiere una recompilación del ejecutable**
  - **Perdemos flexibilidad, ganamos en rendimiento**
  - Las páginas web están en `/usr/share/nginx/html`
  - El resto de los directorios son los mismos que en el caso de Debian
  - No hay `sites-enabled` ni `sites-available`, la configuración debe ser modularizada a mano
  - Si por cambiar los módulos de Nginx hemos recompilado las fuentes, el usuario por defecto ya no será `nginx`, sino `nobody`
    - Debe cambiarse en aquellos procedimientos de seguridad donde aparezca mención al mismo
  - Ante cambios de configuración, para que sean leídos hay que recargar la configuración con `/usr/sbin/nginx -s reload`
  - Aunque normalmente no tendremos que hacerlo, aquí puede verse como abrir servicios en el firewall del SO: https://linuxconfig.org/install-nginx-on-redhat-8

Sobre la recompilación de las fuentes de Nginx

José Manuel
Redondo López

- **Aunque la recompilación del ejecutable de `nginx` para personalizar los módulos que incluimos es viable, recompilar `nginx` con RHEL 8 (y quizá otros SO tipo Red Hat) puede darnos problemas**
  - Por ejemplo, la recompilación conlleva también la recompilación del módulo SSL, para soportar `https`
  - Pero sin embargo esto en ocasiones rompe la ejecución de los gestores de paquetes `yum` o `dnf`
    - ¡No podremos actualizar el SO!
    - También se cambian rutas por defecto, usuarios y pueden aparecer incompatibilidades varias
- **Por ese motivo, y por simplicidad, en esta asignatura se recomienda no modificar los módulos que vienen compilados por defecto con `nginx` bajo SO Red Hat**
  - O aquellos cuya instalación requiera una recompilación en cualquier SO por no ser dinámicos (`mod_security`, `pagespeed`)
- **Esto conlleva no poder usar los módulos**
  - `auth_pam`: perdiendo acceso a varios mecanismos de autenticación integrados
  - `mod_security`: si bien en este caso es mejor usar la versión de pago Nginx Plus
  - `pagespeed`

Server blocks

Las unidades que encapsulan cada web servida por Nginx

Manejando server blocks

José Manuel
Redondo López

- Como dijimos los server blocks son similares a los hosts virtuales de Apache 2
  - Contienen los detalles de configuración de cada sitio servido
  - Permitiendo que un mismo servidor sirva varios
- Por defecto, tiene un único server block cuyo contenido se sirve de `/var/www/html`
- Para poder servir varias webs debemos
  - Crear directorios separados para cada uno, dentro de este directorio base, que contengan sus ficheros
  - Crear la configuración para cada una de estas localizaciones en su server block correspondiente
- Vamos a hacer un ejemplo que ilustre cómo hacer todo el proceso

Manejando server blocks

José Manuel
Redondo López

- Creamos los directorios
  - `sudo mkdir -p /var/www/html/web1.com/html`
  - `sudo mkdir -p /var/www/html/web2.com/html`
- Cambiamos el propietario de estos directorios por nuestro usuario, para no tener que hacer `sudo` en cada operación
  - Para ello hacemos uso de la variable de entorno `$USER`, que contiene nuestro usuario actual
  - `sudo chown -R $USER:$USER /var/www/html/web1.com/html`
  - `sudo chown -R $USER:$USER /var/www/html/web2.com/html`

Manejando server blocks

José Manuel
Redondo López

- Al igual que en el caso de Apache 2, se usa un usuario estándar (también llamado `www-data`) para el acceso anónimo al contenido de las webs
- Por tanto, es probable que haya que configurar los permisos algo más en detalle para adaptarse a las necesidades de cada una de ellas
  - Especialmente en el caso de webs dinámicas que usan ciertas tecnologías
  - Para lo cual debemos consultar los manuales de cada tecnología que usemos

Manejando server blocks

José Manuel
Redondo López

- Una vez hecho esto, debemos copiar el contenido a servir por cada web en cada uno de los directorios creados
- Al igual que Apache 2, por defecto se sirven los documentos `index.html` presentes en una localización a la que se accede vía web
- Hecho esto, ya estamos en disposición de configurar los server blocks de cada web:
  - Para ello copiamos la configuración del server block por defecto: `sudo cp /etc/nginx/sites-available/default /etc/nginx/sites-available/web1.com`
  - Y modificamos lo necesario para nuestro primer server block
- En la siguiente transparencia vemos un ejemplo de cómo es el fichero del server block por defecto
  - Que iremos modificando paso a paso para adaptarlo a nuestro nuevo server block

MANEJANDO SERVER BLOCKS: DEFAULT SERVER BLOCK

José Manuel
Redondo López

```
server {
        listen 80 default_server;
        listen [::]:80 default_server;

        root /var/www/html;
        index index.html index.htm index.nginx-debian.html;

        server_name _;

        location / {
                try_files $uri $uri/ =404;
        }
}
```

Manejando server blocks: Creación de un server block

José Manuel
Redondo López

- **Sólo uno de los server blocks de Nginx puede tener la opción `default_server` activa**
  - El `default_server` es el server block que se cargará en caso de que se intente servir una petición que no encaje con ninguno de los server blocks registrados en el servidor
- **Podemos dejar el server block por defecto para estas funciones, o bien eliminarlo y activar como `default_server` uno de los que vamos a crear**
  - En todo caso, es importante que el fichero de configuración del que estamos creando no lleve `default_server` si ya hay otro que lo hace

```
server {
        listen 80;
        listen [::]:80;

        . . .
}
```

Manejando server blocks: Creación de un server block

José Manuel
Redondo López

- La siguiente modificación a realizar es cambiar el parámetro `root` para que encaje con el directorio que hemos creado antes para ese server block

```
server {
        listen 80;
        listen [::]:80;

        root /var/www/html/web1.com/html;

}
```

Manejando server blocks: Creación de un server block

José Manuel
Redondo López

- Luego, tenemos que cambiar el parámetro `server_name`
  - Para que sea el nombre del dominio asignado a nuestro primer server block
  - También podemos añadirle alias para dicho nombre
- El resto de configuración podemos dejarla como en la imagen si no lo estuviera ya

```
server {
        listen 80;
        listen [::]:80;

        root /var/www/example.com/html;
        index index.html index.htm index.nginx-debian.html;

        server_name web1.com www.web1.com;
        location / {
                try_files $uri $uri/ =404;
        }
}
```

Manejando server blocks: Creación de un server block

José Manuel
Redondo López

- Ahora hacemos lo mismo para nuestro segundo server block
  - `sudo cp /etc/nginx/sites-available/web1.com /etc/nginx/sites-available/web2.com`
- Siguiendo los pasos vistos, deberíamos tener un fichero como este

```
server {
        listen 80;
        listen [::]:80;

        root /var/www/web2.com/html;
        index index.html index.htm index.nginx-debian.html;

        server_name web2.com www.web2.com;

        location / {
                try_files $uri $uri/ =404;
        }
}
```

Manejando server blocks: Creación de un server block

José Manuel
Redondo López

- **Ahora ya podemos activar ambos server blocks**
  - Mediante la creación de un enlace dinámico a los ficheros de configuración en el directorio que hemos mencionado (solo en Debian)
  - `sudo ln -s /etc/nginx/sites-available/web1.com /etc/nginx/sites-enabled/`
  - `sudo ln -s /etc/nginx/sites-available/web2.com /etc/nginx/sites-enabled/`
- **En Red Hat, la configuración se puede modularizar con la directiva `include`:**
  - https://www.nginx.com/resources/wiki/start/topics/examples/full/
- **Con esto tendremos tres server blocks operativos**
  - `web1.com`: Para peticiones a `web1.com` y `www.web1.com`
  - `web2.com`: Para peticiones a `web2.com` y `www.web2.com`
  - `default`: Para peticiones HTTP (puerto 80) que no encajen con las dos anteriores
- **Para evitar un posible problema de memoria que pueda aparecer por haber añadido más nombres de servidor, debemos abrir `/etc/nginx/nginx.conf`**
  - Y descomentar la directiva `server_names_hash_bucket_size`

Manejando server blocks: Creación de un server block

José Manuel
Redondo López

- Ahora ya podemos reiniciar **Nginx**
  - No obstante, antes de hacerlo conviene **comprobar que el fichero de configuración no tiene ningún error** con: `sudo nginx -t`
- Si no se detecta ningún error, podemos reiniciar el servidor
  - `sudo systemctl restart nginx`
- Y ahora nuestro servidor podrá atender estos nombres de dominio
- No obstante, en nuestras pruebas no tenemos servidor DNSç
  - Por lo que hay que hacer una configuración adicional **cambiando el fichero** `/etc/hosts` (en Linux)
  - O equivalente en Windows

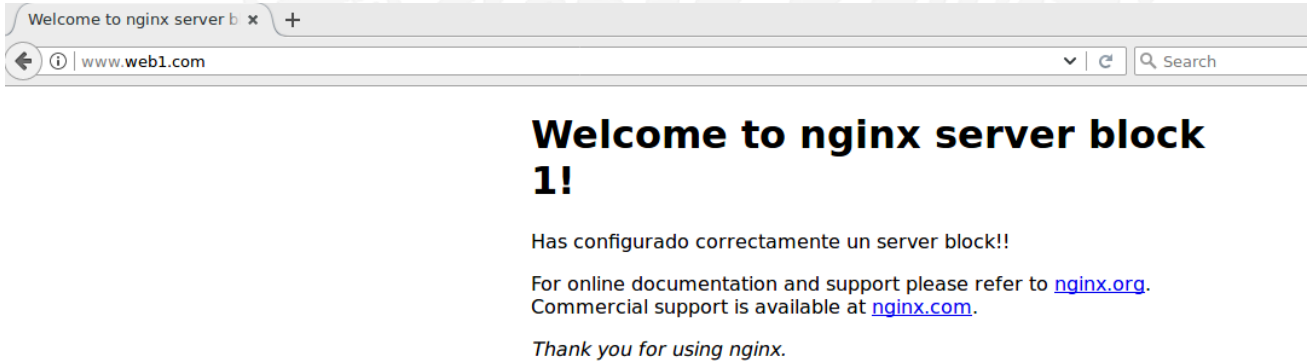
Manejando server blocks: Pruebas de funcionamiento

José Manuel
Redondo López

- Ejemplo de modificación de fichero `/etc/hosts` bajo **Linux**:

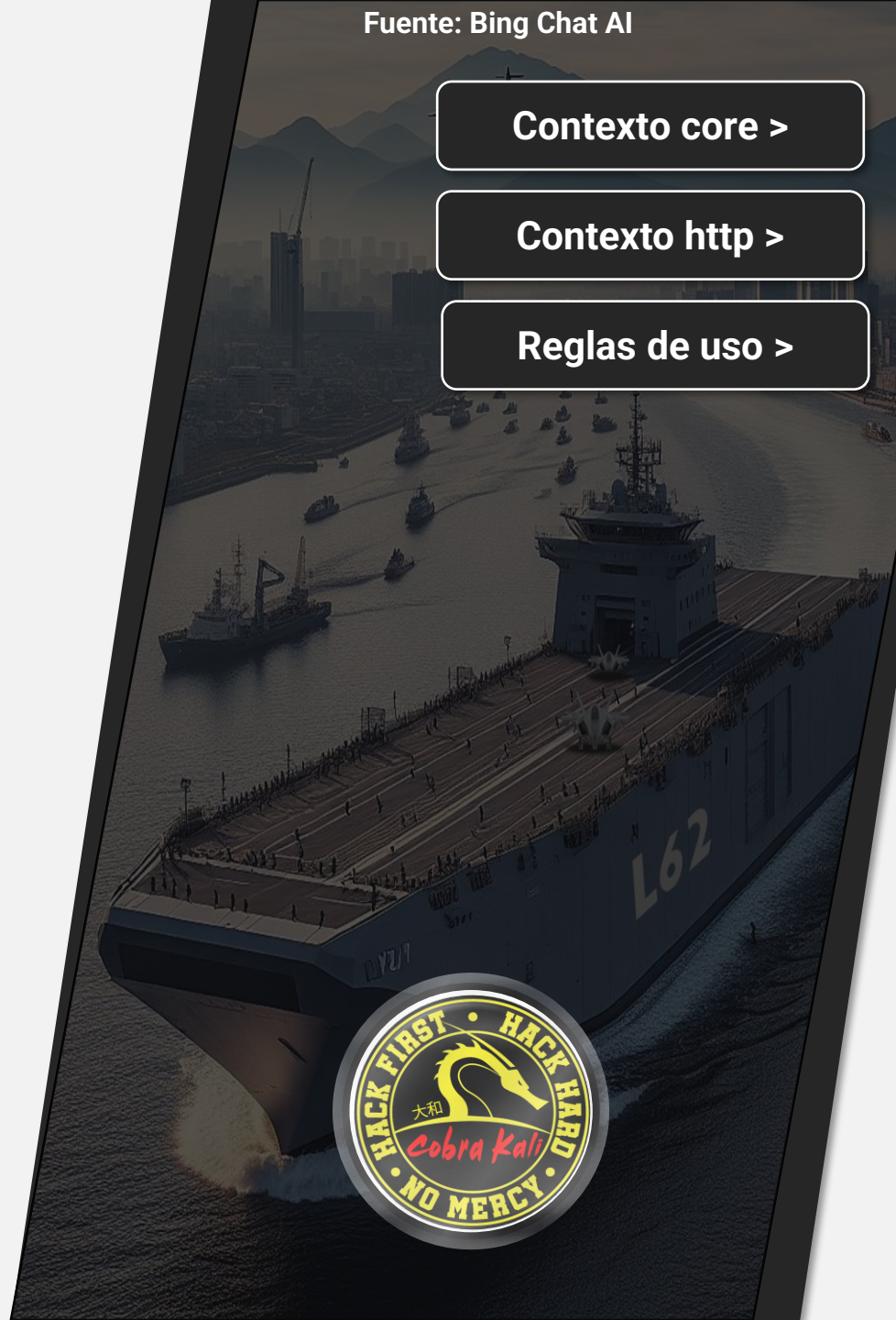
```
127.0.0.1   localhost
. . .
127.0.0.1 web1.com www.web1.com
127.0.0.1 web2.com www.web2.com
```

- Hechos estos cambios, ya podemos probar los nombres de dominio en nuestra máquina cliente



Contextos de configuración

Cómo se organizan los ficheros de configuración de Nginx



Contextos de configuración de Nginx

José Manuel
Redondo López

- **Para poder seguir trabajando con Nginx es muy importante entender cómo se estructuran sus ficheros de configuración**
- **Los ficheros de configuración de Nginx tienen forma de árbol**
  - Formado por **bloques entre { }** llamados **contextos**
  - Cada contexto representa **un área** en el que la serie de configuraciones que contiene se aplica
  - Los contextos pueden **anidarse** unos dentro de otros, formando la estructura de la configuración
    - Si un contexto está dentro de otro, hereda todas las directivas de su contenedor
    - Además, un contexto "hijo" puede sobrescribir cualquier valor de las directivas heredada de sus padres
- **Muchas directivas están pensadas para usarlas solo en ciertos contextos**
  - Ngnix dará un **error de configuración** si tratamos de usarlas en otros
  - La documentación de la directiva contiene esta información
- **Hay 3 contextos importantes (o "core") en la configuración de Nginx: Principal, events y http**
  - Y del http luego dependen otros que son importantes para configurar las web

Contextos “core”

Contextos de configuración "Core" de Nginx: Principal

José Manuel
Redondo López

- **El único contexto que no está contenido en otro**
  - Cualquier directiva **que exista fuera de un bloque de un contexto** pertenece al contexto principal
  - Contiene configuraciones que **afectan a toda una aplicación**
  - Si se importa un fichero de configuración desde otro, las directivas del main context del fichero importado pasan a formar parte del contexto que hace la importación del fichero
    - No se añaden al main context del fichero que lo importa
  - Muchas de sus directivas **no pueden ser heredadas**
    - Los contextos hijos no pueden redefinirlas
  - Contiene configuraciones como
    - El usuario y grupo que usan los procesos worker
    - El nº de procesos worker
    - Afinidad por determinadas CPUs
    - El fichero de log de errores
    - …

```
# Todo lo definido aquí (fuera
de cualquier contexto)
pertenece al principal
. . .
http {
    . . .
}
```

Contextos de configuración "Core" de Nginx: Events

José Manuel
Redondo López

- **Solo puede haber uno en toda la configuración de Nginx**
  - Con él se configura **cómo gestiona las conexiones el servidor** a nivel general
  - Nginx usa un modelo **basado en eventos** para procesar las conexiones
  - En este contexto se establecen las técnicas para gestionar las distintas conexiones
    - O modificar la forma en la que estas técnicas se implementan
  - Normalmente la forma de procesar conexiones se selecciona automáticamente a la más eficiente en función del SO usado
  - En esta sección también se configuran aspectos de funcionamiento de los worker processes
    - Como, por ejemplo, si gestionan varias conexiones a la vez

```
# Contexto principal
…
events {
        . . .
}
```

Contextos de configuración "Core" de Nginx: http

José Manuel
Redondo López

- **Cuando Nginx se usa como servidor web o proxy, la mayoría de la configuración se incluirá como parte de este contexto**
  - Define **cómo gestiona el servidor cualquier conexión tipo HTTP o HTTPs**
  - Se debe definir como parte del contexto principal (no anidado)
  - Se usa para establecer cómo se comportan por defecto todos los server blocks que definamos en el servidor
    - Cada server block puede dar un valor concreto a muchas de las directivas que definamos aquí
  - Contiene directivas como
    - Localización por defecto de logs de peticiones y error
    - Páginas de error
    - Compresión de peticiones
    - Configurar el parámetro Keep Alive
    - Reglas de optimización
    - …
  - **Más información**
    - http://nginx.org/en/docs/http/ngx_http_core_module.html#http

```
# Contexto principal
…
events {
    . . .
}
http {
    . . .
}
```

Otros Contextos de configuración de Nginx

José Manuel
Redondo López

- **Los siguientes contextos dependen de módulos opcionales, se usan solo en determinadas circunstancias o bien de manera minoritaria**
  - `split_clients`: Divide las peticiones de los clientes en categorías etiquetándolas con variables
    - Permitiendo server contenidos diferentes a distintos clientes
  - `geo`: Asigna un valor a una variable en función de la IP con la que un cliente se conecta
  - `types`: Mapea tipos MIME a extensiones de fichero asociados con los mismos.
    - Normalmente Nginx hace esto en un fichero aparte que se carga desde el `nginx.conf`
  - `upstream`: Contexto para designar servidores que pueden usarse como parte de un proxy
    - http://nginx.org/en/docs/http/ngx_http_upstream_module.html
  - `mail`: Contexto para usar Nginx como servidor de correo
    - https://www.nginx.com/resources/admin-guide/mail-proxy/
  - `if`: Contexto para emplear estructuras de control dentro de los ficheros de configuración de Nginx
    - https://www.nginx.com/resources/wiki/start/topics/depth/ifisevil/
  - `limit_except`: Restringe el conjunto de métodos HTTP admitidos dentro de una localización
    - http://nginx.org/en/docs/http/ngx_http_core_module.html#limit_except
  - **Más información**
    - https://www.digitalocean.com/community/tutorials/understanding-the-nginx-configuration-file-structure-and-configuration-contexts

Contextos que se declaran dentro del contexto “http”

Contextos de configuración "http" de Nginx: "server"

José Manuel
Redondo López

- **Uno por cada server block**
  - Cada petición solo puede ser atendida por la configuración de un solo contexto server
- **Se usa información de la petición para seleccionar el adecuado, según una de estas directivas de `server`**
  - `listen`: La dirección IP / Puerto que el contexto server atiende
  - `server_name`: Ante múltiples bloques `server` asociados a la misma IP y puerto, Nginx usa la cabecera Host Header para seleccionar aquella que encaje con el valor de esta directiva
- **Las directivas de `server`**
  - Pueden redefinir muchas de las definidas en `http`
  - Además de configurar ficheros para intentar responder a ciertas peticiones (`try_files`), hacer redirecciones (`return`), reescrituras (`rewrite`) y modificar valores (`set`)
- **Más información**
  - http://nginx.org/en/docs/http/ngx_http_core_module.html#server

```
# main context
http {
    # http context
    server {
        # Primer contexto server
    }
    server {
        # Segundo contexto server
    }
}
```

Contextos de configuración "Http" de Nginx: "Location"

José Manuel
Redondo López

- **Se pueden definir varios contextos `location` dentro de uno `server`**
  - Cada uno para un tipo de petición
  - La localización se seleccionará a través de un determinado algoritmo
- **Los contextos `location` deben estar dentro de contextos `server`**
  - Pueden anidarse, para ir creando conjuntos de reglas más especializados a medida que se llega a localizaciones más concretas
- **Más información**
  - http://nginx.org/en/docs/http/ngx_http_core_module.html#location

```
# main context
server {
    # server context
    location /match/criteria {
        # Primer contexto location
    }
    location /other/criteria {
        # Segundo contexto location
        location nested_match {
            # Primer contexto location anidado
        }
        location other_nested {
            # Segundo contexto location anidado
        }
    }
}
```

Contextos de configuración de Nginx: Contexto "Location"

José Manuel
Redondo López

- **Para seleccionar una `location` concreta una vez determinado el contexto `server` adecuado, se mira a la URI de la petición tras el nombre DNS o la combinación IP:puerto**
  - Ej.: Si la petición es http://www.miweb1.com/foro en el puerto 80...
  - Se seleccionará el contexto `server` www.miweb1.com asociado a HTTP y dentro del mismo la localización `/foro`, si existe entre las definidas para ese `server`
- **Muchas de las directivas definidas en este contexto también lo están en contextos padres**
  - Por lo que pueden sobrescribirlas
- **Las directivas típicas de este contexto son**
  - Los `alias` (para poder acceder a localizaciones fuera del document root)
  - Marcar localizaciones como solo accesibles a nivel interno (`internal`)
  - Y crear proxies en localizaciones concretas

Contextos de configuración "Http" de Nginx: "Location"

José Manuel
Redondo López

- Las reglas de procesamiento y selección de contextos `location` son claves para entender el funcionamiento de Nginx
  - Toda la información está aquí: https://www.digitalocean.com/community/tutorials/understanding-nginx-server-and-location-block-selection-algorithms
- La sintaxis de un contexto `location` es la siguiente

```
location <modificador opcional> <localización> {. . . }
```

- La localización es la cadena a comparar con la URI de la petición

Contextos de configuración "Http" de Nginx: "Location"

José Manuel
Redondo López

- El modificador afecta a como se hace la comparación entre la URI y la cadena
- Existen las siguientes opciones
  - **Dejarlo en blanco**: En caso de no especificarse, la localización se interpreta como un prefijo, es decir, que se comparará el principio de la URI para ver si encaja con la localización o no
  - **=**: Indica que la comparación se hace de forma exacta, es decir, que la URI debe encajar exactamente con la localización especificada
  - **~**: La localización se interpretará como una expresión regular, pero sensible a mayúsculas
  - **~***: Ídem al anterior, pero sin sensibilidad a mayúsculas
  - **^~**: Si el bloque que lleva este modificador es el que mejor encaja con la URI sin interpretarse la localización como una expresión regular
    - No se procesarán más expresiones regulares para determinar si otra localización encaja con la misma

Contextos de configuración "Http" de Nginx: "Location"

José Manuel
Redondo López

- Esta localización encaja con peticiones a `/clientes`, `/clientes/martinez/index.html` o `/clientes/index.html`

```
location /clientes {. . .}
```

- Esta localización encaja con `/clientes`, pero no con `/clientes/martinez/index.html` o `/clientes/index.html`

```
location = /clientes {. . .}
```

- Esta localización encaja con peticiones a imágenes como `magneto.jpg`, pero no `ciclope.GIF`:

```
location ~ \.(jpe?g|png|gif|ico)$ {. . .}
```

- Esta sin embargo encaja con ambas imágenes

```
location ~* \.(jpe?g|png|gif|ico)$ {    . . .}
```

- Esta localización encaja con `/coches/honda_civic.html` y no se hará ningún procesamiento de expresiones regulares en otros bloques `location` para determinar si otro encaja con la URI

```
location ^~ /coches {. . .}
```

Contextos de configuración "http" de Nginx: "location"

José Manuel
Redondo López

- **Nginx sigue un proceso para determinar qué localización es la más adecuada para servir una petición y, por tanto, su contexto `server` adecuado**
  - Primero comprueba las localizaciones **que no implican usar expresiones regulares** para determinar si encajan contra la URI de la petición
  - Dentro de las mismas, comprueba **aquellas que usan el modificador `=`**
    - Si encuentra una que encaja exactamente, esta será la elegida para procesar la petición
  - En caso contrario, evalúa las otras localizaciones que no encajan exactamente, seleccionando aquella que **tenga la especificación de localización más larga** que encaje con la URI
    - Si esta tiene el modificador `^~`, sirve la petición
  - En caso contrario, se guarda para futuros usos
  - Hecho esto, **se evalúan las localizaciones que implican el uso de expresiones regulares**
    - Favoreciendo aquellas que aparezcan en las localizaciones más largas
  - La primera que encaje con la URI es la que servirá la petición
  - En caso de que no haya ninguna, se servirá la guardada en un paso anterior

Contextos de configuración "http" de Nginx: "location"

José Manuel
Redondo López

- **Por tanto**
  - Nginx **sirve las localizaciones que tienen expresiones regulares preferentemente** frente a las que no la tienen
  - Este comportamiento puede alterarse con los modificadores `=` y `^~`
  - Aunque se seleccionan las localizaciones basadas en su longitud, **la posición de las mismas en el fichero de configuración** puede afectar al resultado
    - Ya que la primera localización que encaje con la URI es la que sirve la petición

Reglas de uso de los contextos de configuración de Nginx

REGLAS DE USO

José Manuel
Redondo López

- Es conveniente aplicar las directivas en contextos lo más globales posible
- Especialmente cuando una directiva es válida en más de un contexto
- Esto es conveniente para
  - Evitar repetir las mismas configuraciones, al heredarse siempre de la global
  - Si en una localización concreta tenemos que cambiar algo, simplemente lo redefinimos solo donde nos haga falta
  - Si en una localización necesitamos un valor para una directiva, tendremos uno por defecto heredado de la global, no incurriendo en un error por faltar un valor

NO

```
http {
    server {
        location / {
            root /var/www/html;
            . . .
        }
        location /another {
            root /var/www/html;
            . . .
        }
    }
}
```

SI

```
http {
    root /var/www/html;
    server {
        location / {
            . . .
        }
        location /another {
            . . .
        }
    }
}
```

Reglas de uso

José Manuel
Redondo López

- La mayor parte de las veces el nivel `server` es el más adecuado para declarar directivas, para maximizar las ventajas mencionadas
- Si tenemos que tomar decisiones de configuración en función de algún valor en la petición del cliente, es mejor usar múltiples `location` que recurrir al `if`
  - Se ha demostrado que el uso de `if` es más propenso a errores
  - Mejora la legibilidad
  - Las peticiones se procesan de forma óptima
- Finalmente, es importante no cometer alguno de los errores habituales con la configuración del servidor
  - https://www.nginx.com/resources/wiki/start/topics/tutorials/config_pitfalls/
- Y también debemos tener en cuenta las mismas consideraciones finales que hicimos en el tema de Apache 2

Ejemplo de configuración completo analizado: ngnix.conf

José Manuel
Redondo López

- La siguiente configuración se muestra a modo de ejemplo de las partes que hemos visto hasta ahora
  - Con otras que están también en Apache 2 y también algunas que veremos posteriormente
- Su función es la de pasar todas las peticiones a servidores en el backend (Nginx hace de proxy)
  - Salvo las imágenes y las peticiones que empiecen por /download/

```
 # Main context
user  www-data; # Usuario propietario de los worker processes (por defecto)
worker_processes  auto; # Si la máquina es potente, puede asignarse un nº alto. Si no, mejor dejarlo auto
pid /var/run/nginx.pid; # Pid del proceso Nginx. Es mejor no cambiar esta configuración
# Cambia el nivel de log a info. Valores: [ debug | info | notice | warn | error | crit ]
# Consultar https://www.nginx.com/resources/admin-guide/logging-and-monitoring/
error_log  /var/log/nginx.error_log  info;
events { # Contexto events. Normalmente no tendríamos porqué tocar estos valores
    worker_connections   768;
    # Forma de procesar las peticiones. Valores: [ kqueue | epoll | /dev/poll | select | poll ];
    # http://nginx.org/en/docs/events.html
    use kqueue; # Conviene dejar el valor que el sistema nos aconseje en la instalación
}
```

EJEMPLO DE CONFIGURACIÓN COMPLETO ANALIZADO

José Manuel
Redondo López

```
http { # Contexto http
    include    conf/mime.types; #Cómo incluir otro fichero
    default_type  application/octet-stream;
    log_format main      '$remote_addr - $remote_user [$time_local] '
                         '"$request" $status $bytes_sent '
                         '"$http_referer" "$http_user_agent" '
                         '"$gzip_ratio"';
    log_format download  '$remote_addr - $remote_user [$time_local] '
                         '"$request" $status $bytes_sent '
                         '"$http_referer" "$http_user_agent" '
                         '"$http_range" "$sent_http_content_range"';
    client_header_timeout  3m;
    client_body_timeout    3m;
    send_timeout           3m;
    client_header_buffer_size    1k;
    large_client_header_buffers  4 4k;

    gzip on;  #Habilitar compresión de peticiones
    gzip_min_length  1100;
    gzip_buffers     4 8k;
    gzip_types       text/plain;

    output_buffers   1 32k;
    postpone_output  1460;

    sendfile         on;
    tcp_nopush       on;
    tcp_nodelay      on;
    send_lowat       12000;

    keepalive_timeout  75 20; # Configuración keep-alive TCP

    #lingering_time     30;
    #lingering_timeout  10;
    #reset_timedout_connection  on;
… (al final se incluyen los .conf de sites-enabled, entre otros)
```

EJEMPLO DE CONFIGURACIÓN COMPLETO ANALIZADO: DEFAULT

José Manuel
Redondo López

```
server { # Contexto server
        listen          www.web1.com; # Si tenemos DNS
        server_name     www.web1.com;
        # Log de accesos (ver https://www.nginx.com/resources/admin-guide/logging-and-monitoring/)
        access_log    /var/log/nginx.access_log  main;

        location / { # Primer contexto location
            # Configuramos un proxy
             proxy_pass          http://127.0.0.1/;
             proxy_redirect      off;
             # Cabeceras que recibirá el backend
             proxy_set_header    Host             $host;
             proxy_set_header    X-Real-IP        $remote_addr;
              client_max_body_size       10m;
             client_body_buffer_size    128k;
             client_body_temp_path  /var/nginx/client_body_temp;

             proxy_connect_timeout       70;
             proxy_send_timeout          90;
             proxy_read_timeout          90;
             proxy_send_lowat            12000;

             proxy_buffer_size           4k;
             proxy_buffers               4 32k;
             proxy_busy_buffers_size     64k;
             proxy_temp_file_write_size 64k;

             proxy_temp_path             /var/nginx/proxy_temp;
             charset  koi8-r;
        }
```

EJEMPLO DE CONFIGURACIÓN COMPLETO ANALIZADO: DEFAULT

José Manuel
Redondo López

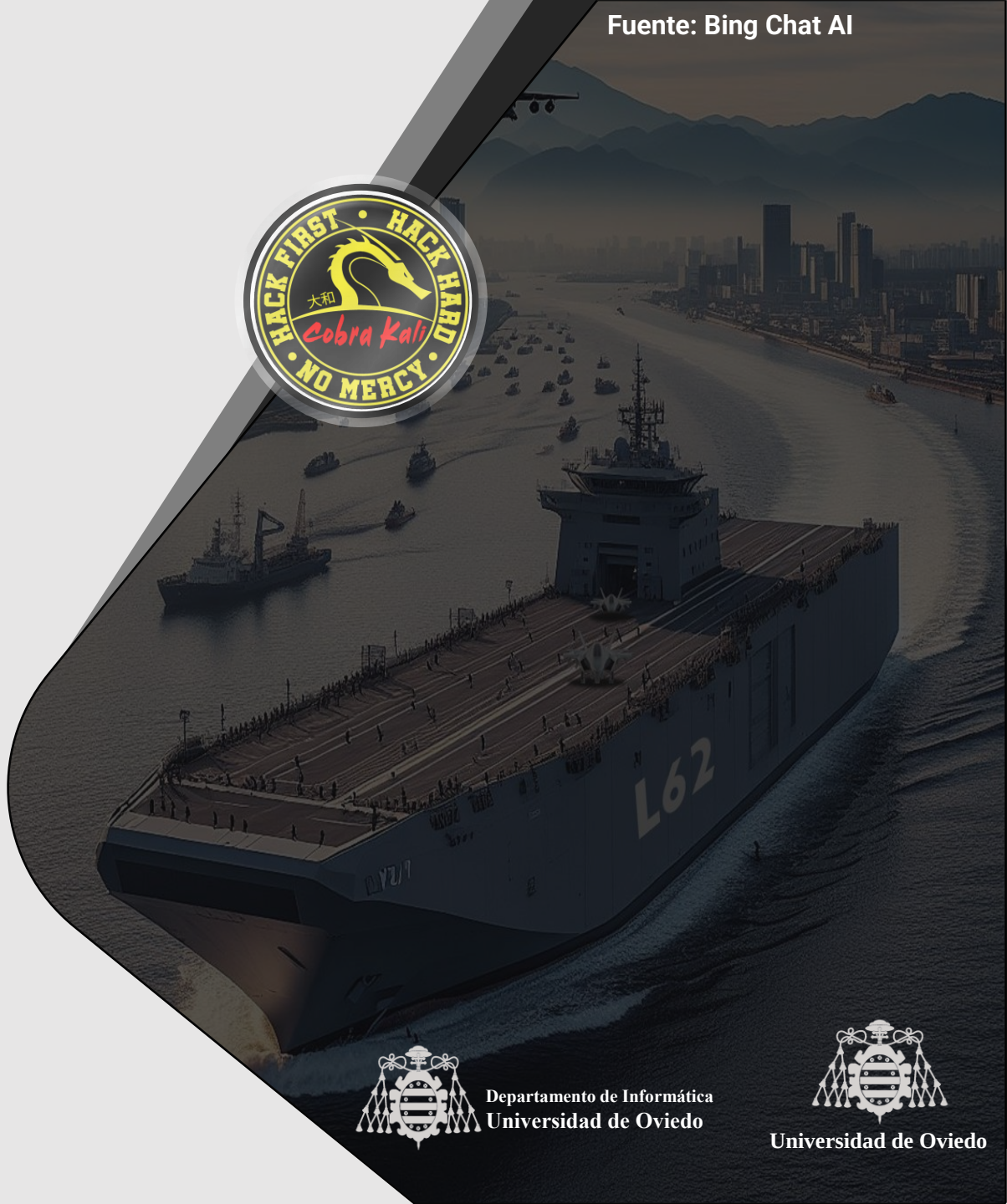
```
error_page  404  /404.html;
# location página de error
location /404.html {
    root  /spool/www;
}
# Uso de reescritura en este location
location /v_previa/ {
    rewrite   ^/v_previa/(.*)$  /release/$1  permanent;
}
# Location que representa imágenes
location ~* \.(jpg|jpeg|gif)$ {
    root         /spool/www;
    access_log   off;
    expires      30d;
}
```

```
# Location /download
location /download/ {
    valid_referers none blocked server_names *.web1.com;

    if ($invalid_referer) { # Acceso limitado a nuestro DNS
        #rewrite   ^/   http://www.web1.com/;
        return   403;
    }

    #rewrite_log  on;
    # Otro ejemplo de reescritura de URLs
    rewrite ^/(download/.*)/mp3/(.*)\..*$
            /$1/mp3/$2.mp3                    break;

    root      /spool/www;
    autoindex  on; # Desactivamos índice de directorios
    access_log /var/log/nginx-download.access_log download;
 }
}
}
```

< Ir al Índice

Proxies con Nginx

Haciendo a Nginx una “barrera”



Departamento de Informática
Universidad de Oviedo

Universidad de Oviedo

Conceptos básicos de proxies con Nginx

José Manuel
Redondo López

- **Nginx viene preparado para ser un reverse proxy fácilmente**
  - Distribuir el procesamiento de peticiones entre distintos servidores sin que el cliente lo sepa
  - Si Nginx se configura como un proxy, manda la petición al servidor indicado y guarda la respuesta
  - Para enviarla de nuevo al cliente sin que tengamos que hacer nada más
- **Veremos unos conceptos básicos para poner en marcha un proxy típico**
  - Podemos ampliar información en: https://www.nginx.com/resources/admin-guide/reverse-proxy/
- **Se pueden enviar peticiones a servidores HTTP (Nginx o no) y no HTTP**
  - Usando un determinado protocolo de entre los soportados
  - FastCGI, uwsgi, SCGI y memcached
- **Para pasar una petición HTTP a un servidor se usa la directiva `proxy_pass` dentro de un contexto `location`**

```
location /operaciones/clientes/ {
    proxy_pass http://www.gestionclientes.com/procesamiento/;
}
```

Conceptos básicos de proxies con Nginx

José Manuel
Redondo López

- Se admiten también conjuntos de IPs y puertos

```
location ~ \.php {
    proxy_pass http://127.0.0.1:8000;
}
```

- En el primer ejemplo la dirección del proxy lleva la cadena `/procesamiento/`
- Así se reemplaza la parte de la URI que encaja con el contexto `location` por dicha cadena
  - Es decir que una petición a `/operaciones/clientes/cliente_alta.html` se pasará a `http://www.gestionclientes.com/procesamiento/cliente_alta.html`
- Otras directivas que soportan otros protocolos son:
  - `fastcgi_pass` para pasar peticiones a un servidor que entienda el protocolo FastCGI
  - `uwsgi_pass` para pasar peticiones a un servidor que entienda el protocolo uwsgi
  - `scgi_pass` para pasar peticiones a un servidor que entienda el protocolo SCGI
  - `memcached_pass` para pasar peticiones a un servidor que entienda el protocolo de memcached
- La directiva `proxy_pass` puede pasar una petición a un grupo de servidores
  - Las peticiones entre ellos se distribuyen de acuerdo al algoritmo de balanceo de carga configurado
  - Veremos un ejemplo al final de la presentación

Conceptos básicos de proxies con Nginx

José Manuel
Redondo López

- Cuando Nginx hace de proxy, redefine dos cabeceras de la petición: `Host` y `Connection`
  - `Host` pasa a valer el valor de la variable `$proxy_host`
  - `Connection` pasa a valer `close`
  - Además, elimina las cabeceras sin contenido
- Se pueden modificar cabeceras a voluntad mediante `proxy_set_header`. Por ejemplo:

```
location /operaciones/clientes/ {
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_pass http://localhost:8000;
}
```

- Por lo dicho anteriormente, si no queremos que una cabecera pase al servidor al que se dirige la petición, basta con hacer que su valor sea la cadena vacía

```
location /operaciones/clientes/ {
    proxy_set_header Accept-Encoding "";
    proxy_pass http://localhost:8000;
}
```

Conceptos básicos de proxies con Nginx

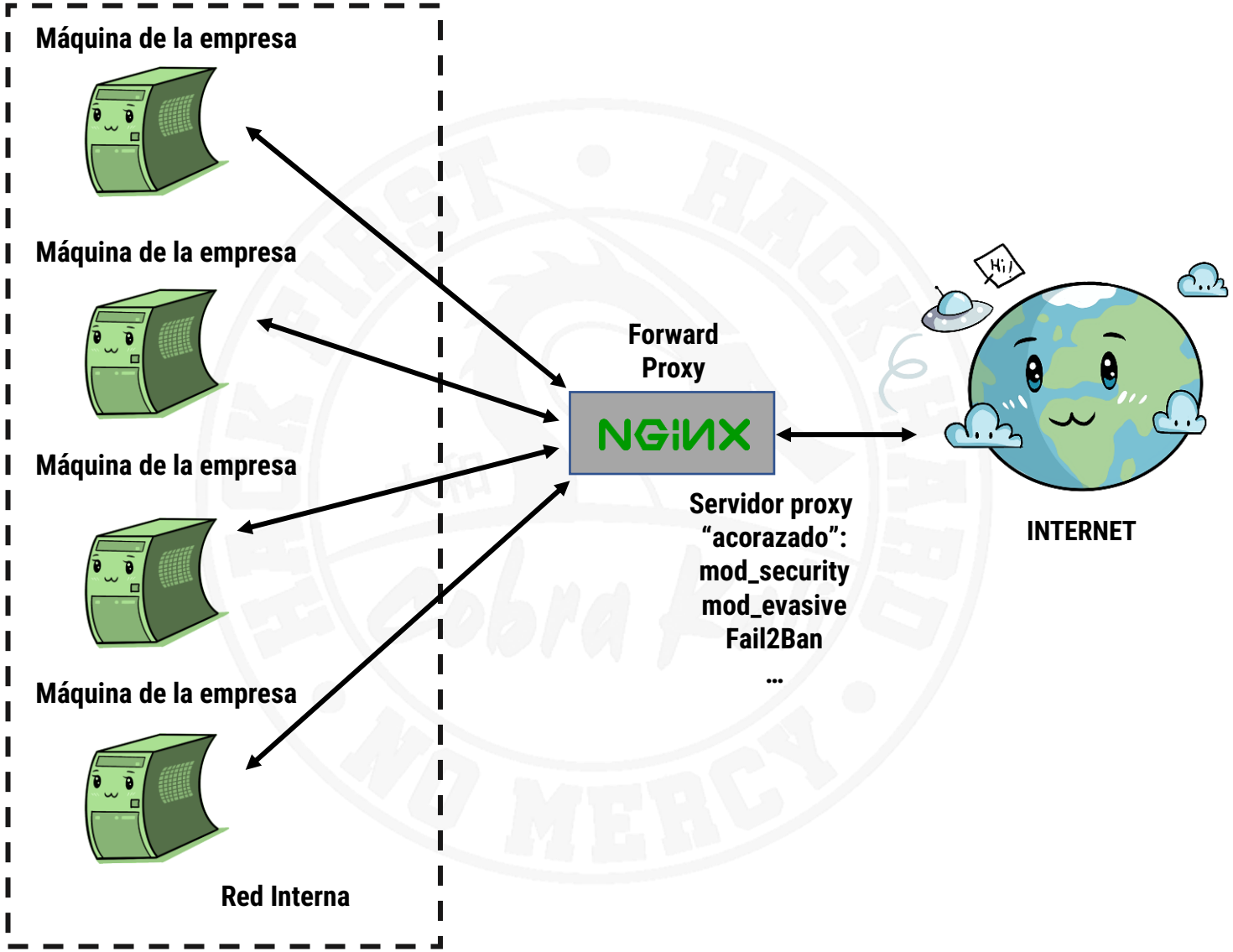
José Manuel
Redondo López

- Si el servidor proxy tiene varios interfaces de red, normalmente es necesario escoger cuál de ellos es el indicado para dirigirse a los servidores que recibirán las peticiones
- Esto se logra con la directiva `proxy_bind`, indicando la IP del interfaz de red adecuado

```
location /clientes/ {
    proxy_bind 127.0.0.1;
    proxy_pass http://miservidor.com/gestion_clientes/;
}

location /facturas/ {
    proxy_bind 127.0.0.2;
    proxy_pass http://miservidor.com/gestion_facturas/;
}
```

FORWARD PROXY

José Manuel
Redondo López



Forward proxy

José Manuel
Redondo López

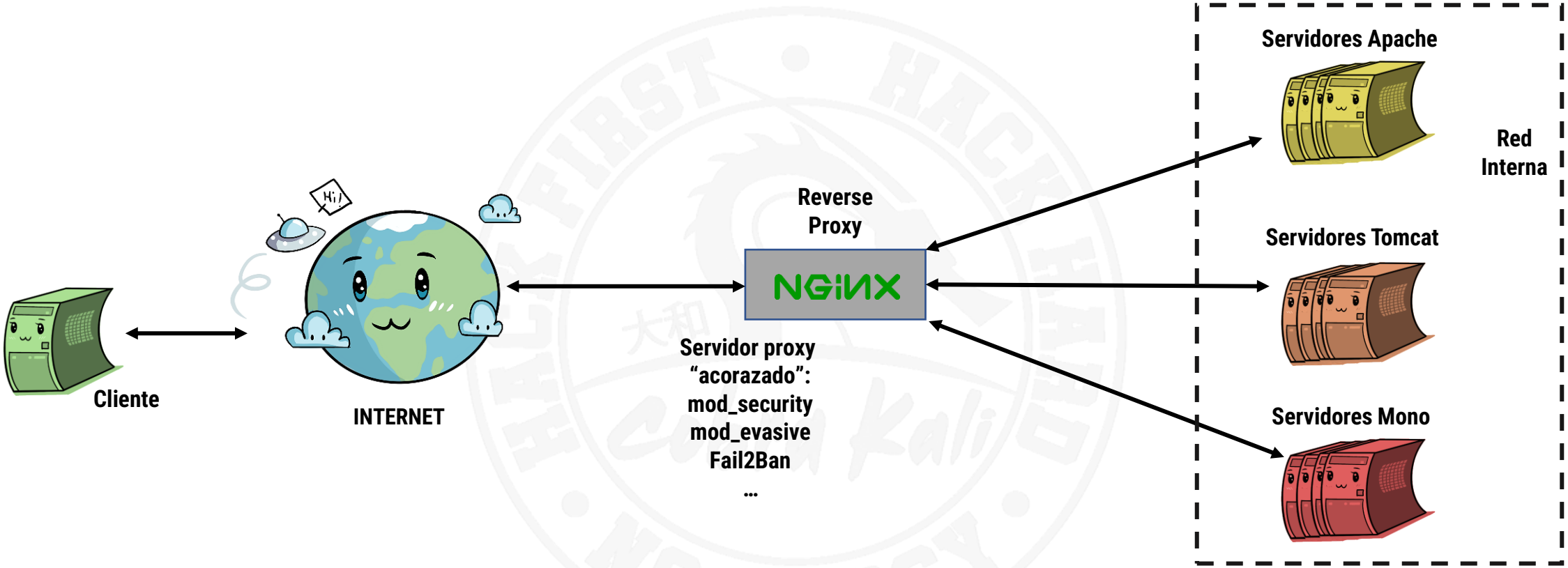
- Aunque **Nginx** facilita la creación de **reverse proxys**, también es posible configurarlo como un **forward proxy** con una configuración como esta

```
server {
    listen       8080; #Puerto 8080, al que deberemos redirigir nuestro navegador

    location / {
        resolver 8.8.8.8; #DNS que resolverá las peticiones
        proxy_pass http://$http_host$uri$is_args$args;
    }
}
```

- Además, ahora también se podrá manipular las peticiones que nos lleguen, como se muestra en el siguiente enlace
  - https://ef.gy/using-nginx-as-a-proxy-server

Reverse Proxy

José Manuel
Redondo López



Reverse Proxy

José Manuel
Redondo López

- Vemos que es sencillo hacer que **Nginx** pase una petición a una `location` a otro servidor
- También hacer que **sea a un grupo de servidores** con el contexto `upstream`
- Los grupos pueden estar compuestos por servidores HTTP en diferentes puertos, TCP o sockets UNIX, incluso mezclados

```
upstream backend {
    server backend1.miservidor.com weight=3;
    server 127.0.0.1:8080  max_fails=3 fail_timeout=30s;
    server unix:/tmp/backend3;
    server backup1.miservidor.com  backup;
}
```

- Las peticiones entre los mismos se distribuyen por el método **round-robin** con pesos (`weight`):
  - En el ejemplo, de cada 5 peticiones 3 se pasarán al 1er servidor, la siguiente al 2º y la siguiente al 3º
  - Si hay un error con cualquiera de ellos se pasará al servidor siguiente en la lista
  - Si ninguno responde, entonces se pasará la petición al indicado como `backup`

Reverse Proxy

José Manuel
Redondo López

- Un ejemplo sencillo de uso de `upstream` es el siguiente:

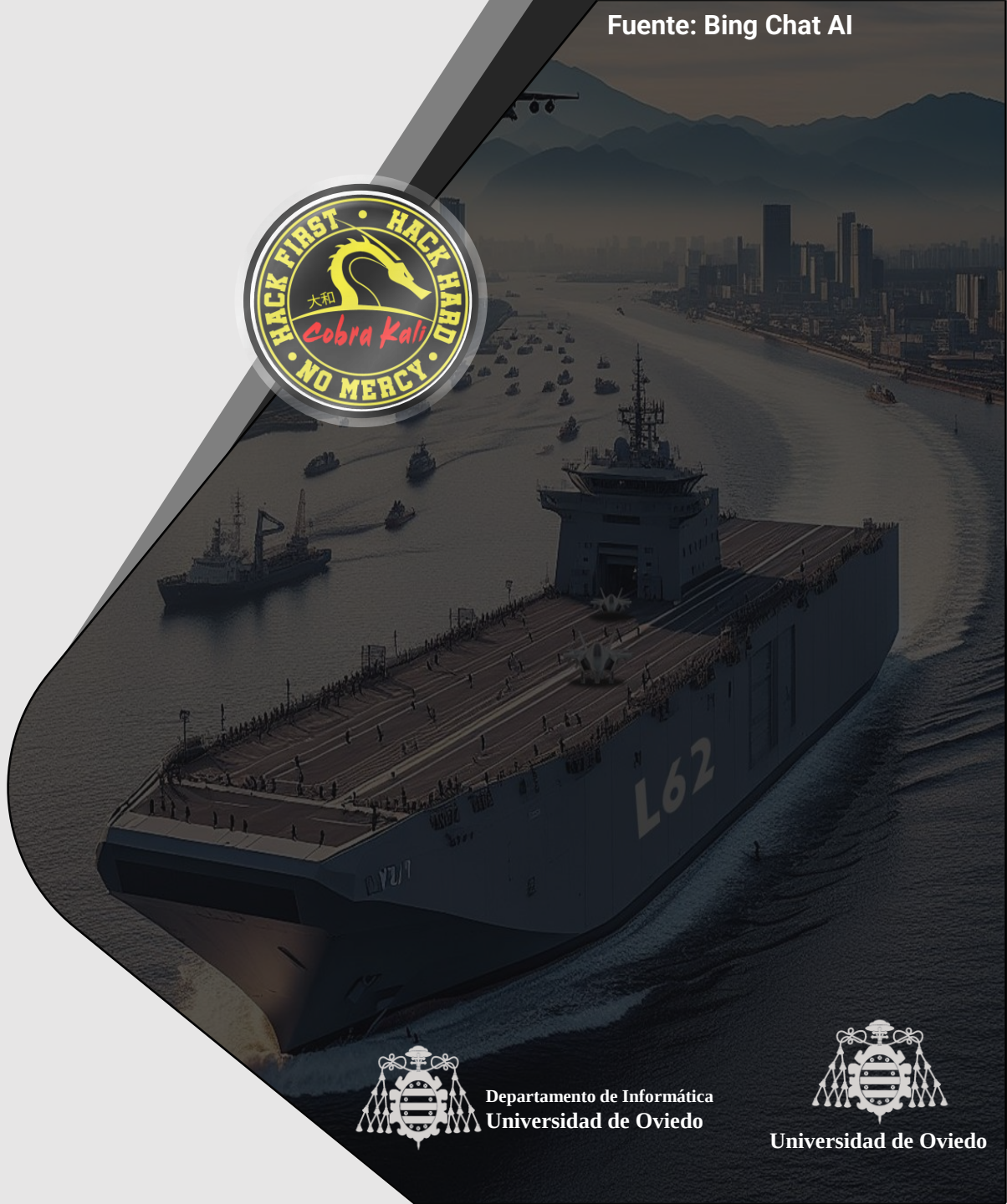
```
http {
 upstream aplicaciongestion {
  server srv1.miservidor.com;
  server srv2.miservidor.com;
  server srv3.miservidor.com;
 }
 server {
  listen 80;
  location / {
   proxy_pass http://aplicaciongestion;
  }
 }
}
```

- Se puede ampliar esta información consultando el siguiente enlace
  - https://umbrella.cisco.com/blog/blog/2015/11/03/lets-talk-about-proxies-pt-2-nginx-as-a-forward-http-proxy/
- Esto es un ejemplo de uso de `upstream`, podemos encontrar más información en
  - http://nginx.org/en/docs/http/ngx_http_upstream_module.html

< Ir al Índice

Implementando el CIS Benchmark de Nginx

Catálogo de acciones a realizar



Departamento de Informática
Universidad de Oviedo

Universidad de Oviedo

Políticas para Nginx

José Manuel
Redondo López

- **NOTA**: Si solo estás cursando la “L-62 Princesa de Asturias”, puedes saltarte esta sección
- El **CIS Benchmark** para **Nginx** es esencialmente lo mismo que el de **Apache**, y todo lo que hemos visto se aplica aquí también
- El único cambio reseñable son los perfiles del **benchmark**, que tienen un apartado especial para proxies
  - Esto es debido a que **Nginx** habitualmente se usa como balanceador de carga y como proxy, además de como servidor web
  - Debido su bajo consumo de recursos, es ideal para estas funciones
- Por ello, el benchmark sigue definiendo dos niveles de seguridad, pero dentro de cada uno contempla **los tres perfiles**

Niveles de seguridad para Nginx

José Manuel
Redondo López

- **Por tanto, este benchmark también define dos niveles**
  - **Nivel 1**: Los elementos de este perfil tienen la intención de
    - Ser práctico y prudente en la aplicación de medidas de seguridad, aplicando de forma contenida aquellas que puedan afectar de forma crítica a una funcionalidad
    - Proporcionar un beneficio de seguridad claro
    - No inhibir la utilidad de la tecnología más allá de lo aceptable
    - Hay 3 variantes de este nivel según la función que cumpla Nginx: Level 1 – Webserver, Level 1 - Proxy y Level 1 - Loadbalancer
  - **Nivel 2**: Amplía el "Nivel 1“ con elementos que presentan 1 o más de estas características
    - Están destinados a entornos o casos de uso donde la seguridad es primordial
    - Actúan como defensa en profundidad
    - A consecuencia de ello, puede afectar negativamente la utilidad o el rendimiento de la tecnología
    - Hay 3 variantes de este nivel según la función que cumpla Nginx: Level 2 - Webserver, Level 2 – Proxy y Level 2 - Loadbalancer

Política CIS Benchmark v1.0 para Nginx

José Manuel
Redondo López

- En esta asignatura tenemos un catálogo de ejercicios compuesto por la checklist de la política **CIS Benchmark v1.0**
- Adicionalmente, se proponen una serie de ampliaciones de seguridad que van más allá de lo contemplado por esta política
- Esto se detalla en la sección siguiente
- Aplicar la política en sí consiste en rellenar las tablas siguientes, que contemplan una serie de acciones divididas en categorías
- Esta tabla está sacada directamente del documento de la política adaptado suministrado

Para implementar la política...

José Manuel
Redondo López

- La política contempla todos los SO, pero está enfocada a Linux tipo **Red Hat**
- La versión que os entregamos se ha revisado y en algunos apartados hay una sección de **Notes** con ciertas correcciones y avisos
  - Conviene leerla **SIEMPRE** antes de empezar a hacer lo que indica la sección
- El menú es navegable para poder usar la política más eficientemente
- Finalmente, debemos recordar que aunque tengan funciones equivalentes, los servidores no tienen una organización de directorios idéntica en **Debian** y **Red Hat**

Política CIS Benchmark v2.0.1 para Nginx

José Manuel
Redondo López

		YES	NO
1	**Initial Setup**		
1.1	**Installation**		
1.1.1	Ensure NGINX is installed (Scored)		
1.1.2	Ensure NGINX is installed from source (Not Scored)		
1.2	**Configure Software Updates**		
1.2.1	Ensure package manager repositories are properly configured (Not Scored)		
1.2.2	Ensure the latest software package is installed (Not Scored)		
2	**Basic Configuration**		
2.1	**Minimize NGINX Modules**		
2.1.1	Ensure only required modules are installed (Not Scored)		
2.1.2	Ensure HTTP WebDAV module is not installed (Scored)		
2.1.3	Ensure modules with gzip functionality are disabled (Scored)		
2.1.4	Ensure the autoindex module is disabled (Scored)		
2.2	**Account Security**		
2.2.1	Ensure that NGINX is run using a non-privileged, dedicated service account (Not Scored)		
2.2.2	Ensure the NGINX service account is locked (Scored)		
2.2.3	Ensure the NGINX service account has an invalid shell (Scored)		

Política CIS Benchmark v2.0.1 para Nginx

José Manuel
Redondo López

		YES	NO
2.3	**Permissions and Ownership**		
2.3.1	Ensure NGINX directories and files are owned by root (Scored)		
2.3.2	Ensure access to NGINX directories and files is restricted (Scored)		
2.3.3	Ensure the NGINX process ID (PID) file is secured (Scored)		
2.3.4	Ensure the core dump directory is secured (Not Scored)		
2.4	**Network Configuration**		
2.4.1	Ensure NGINX only listens for network connections on authorized ports (Not Scored)		
2.4.2	Ensure requests for unknown host names are rejected (Not Scored)		
2.4.3	Ensure keepalive_timeout is 10 seconds or less, but not 0 (Scored)		
2.4.4	Ensure send_timeout is set to 10 seconds or less, but not 0 (Scored)		
2.5	**Information Disclosure**		
2.5.1	Ensure server_tokens directive is set to `off` (Scored)		
2.5.2	Ensure default error and index.html pages do not reference NGINX (Scored)		
2.5.3	Ensure hidden file serving is disabled (Not Scored)		
2.5.4	Ensure the NGINX reverse proxy does not enable information disclosure (Scored)		

Política CIS Benchmark v2.1.0 para Nginx

José Manuel
Redondo López

		YES	NO
3	**Logging**		
3.1	Ensure detailed logging is enabled (Not Scored)		
3.2	Ensure access logging is enabled (Scored)		
3.3	Ensure error logging is enabled and set to the info logging level (Scored)		
3.4	Ensure log files are rotated (Scored)		
3.5	Ensure error logs are sent to a remote syslog server (Not Scored)		
3.6	Ensure access logs are sent to a remote syslog server (Not Scored)		
3.7	Ensure proxies pass source IP information (Scored)		

Política CIS Benchmark v2.1.0 para Nginx

José Manuel
Redondo López

		YES	NO
4	**Encryption**		
4.1	**TLS / SSL Configuration**		
4.1.1	Ensure HTTP is redirected to HTTPS (Scored)		
4.1.2	Ensure a trusted certificate and trust chain is installed (Not Scored)		
4.1.3	Ensure private key permissions are restricted (Scored)		
4.1.4	Ensure only modern TLS protocols are used (Scored)		
4.1.5	Disable weak ciphers (Scored)		
4.1.6	Ensure custom Diffie-Hellman parameters are used (Scored)		
4.1.7	Ensure Online Certificate Status Protocol (OCSP) stapling is enabled (Scored)		
4.1.8	Ensure HTTP Strict Transport Security (HSTS) is enabled (Scored)		
4.1.9	Ensure upstream server traffic is authenticated with a client certificate (Scored)		
4.1.10	Ensure the upstream traffic server certificate is trusted (Not Scored)		
4.1.11	Ensure your domain is preloaded (Not Scored)		
4.1.12	Ensure session resumption is disabled to enable perfect forward security (Scored)		
4.1.13	Ensure HTTP/2.0 is used (Not Scored)		
4.1.14	Ensure only Perfect Forward Secrecy Ciphers are Leveraged		

Política CIS Benchmark v2.1.0 para Nginx

José Manuel
Redondo López

		YES	NO
5	**Request Filtering and Restrictions**		
5.1	**Access Control**		
5.1.1	Ensure allow and deny filters limit access to specific IP addresses (Not Scored)		
5.1.2	Ensure only whitelisted HTTP methods are allowed (Not Scored)		
5.2	**Request Limits**		
5.2.1	Ensure timeout values for reading the client header and body are set correctly (Scored)		
5.2.2	Ensure the maximum request body size is set correctly (Scored)		
5.2.3	Ensure the maximum buffer size for URIs is defined (Scored)		
5.2.4	Ensure the number of connections per IP address is limited (Not Scored)		
5.2.5	Ensure rate limits by IP address are set (Not Scored)		
5.3	**Browser Security**		
5.3.1	Ensure X-Frame-Options header is configured and enabled (Scored)		
5.3.2	Ensure X-Content-Type-Options header is configured and enabled (Scored)		
5.3.3	Ensure the X-Xss-Protection Header is enabled and configured properly (Scored)		
5.3.4	Ensure that Content Security Policy (CSP) is enabled and configured properly (Not Scored)		
5.3.5	Ensure the Referrer Policy is enabled and configured properly (Not Scored)		
6	**Mandatory Access Control**		

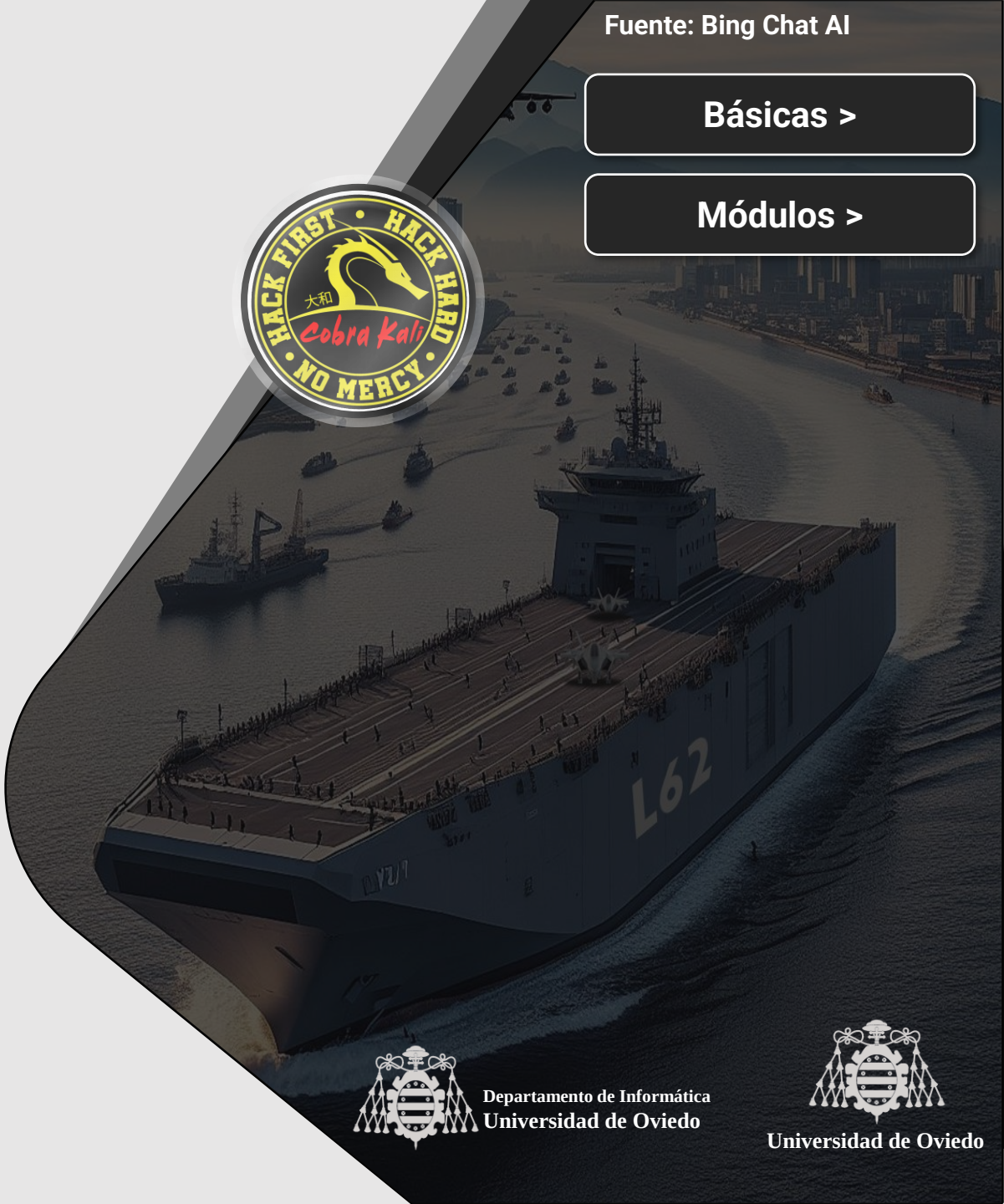
< Ir al Índice

Básicas >

Módulos >

Operaciones complementarias al CIS Benchmark

Configurando Nginx para tener un nivel de seguridad mayor



Departamento de Informática
Universidad de Oviedo

Universidad de Oviedo

¿Qué es esta sección?

José Manuel
Redondo López

- **El propósito de esta sección es doble**
  - Mostrar cómo se hacen algunas de las operaciones contempladas en el CIS Benchmark sobre sistemas basados en Debian
    - Ya que su implementación podría ser ligeramente diferente
    - **Las operaciones del benchmark siempre tendrán más prioridad que las que se encuentran en esta presentación**
  - Hacer operaciones adicionales que lleven a un sistema más seguro y mejor administrado
- **Por tanto, es un complemento y ampliación el benchmark visto hasta ahora**
- **Y capacidad para realizar los ejercicios de la asignatura correspondientes a Nginx**

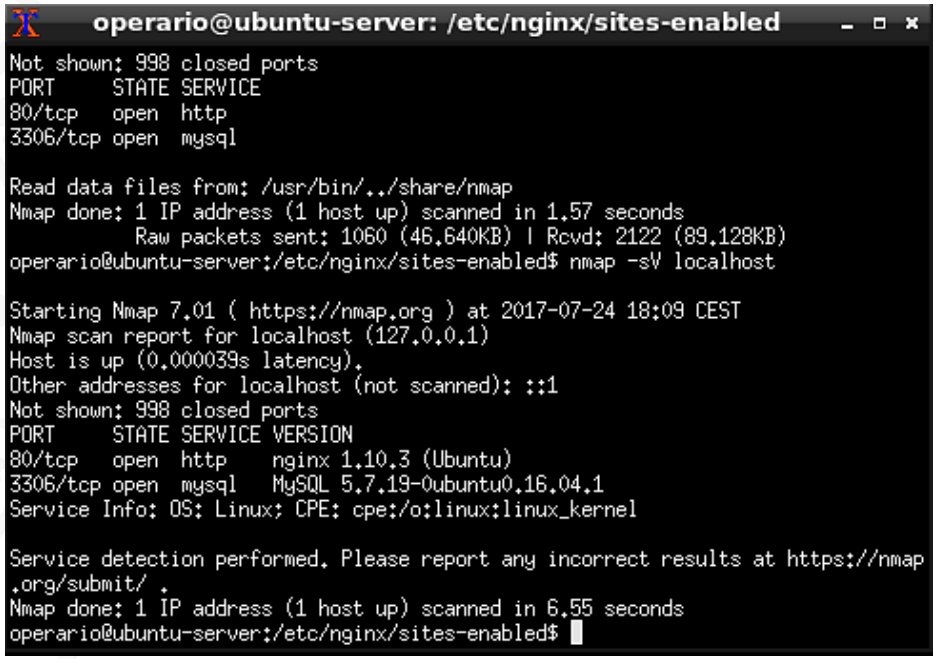
Opciones básicas

Configuración mínima recomendable para todo servidor

Ocultación de la versión del servidor

José Manuel
Redondo López

- Puede ocultarse la versión del servidor añadiendo esto a cualquier bloque `http`, `server` o `location`:
  - `server_tokens off;`
- Es posible también cambiar el contenido de esta cabecera modificándola en los fuentes del mismo
  - Y recompilando el servidor
  - https://serverfault.com/questions/214242/can-i-hide-all-server-os-info
- No obstante, es más fácil instalar el paquete `nginx-extras` (solo **Debian**) para modificar o borrar la cabecera `Server` de las peticiones HTTP
- Para ello, modificamos el `nginx.conf` así
  - Borrar la cabecera: `more_clear_headers Server;`
  - Modificarla por otro valor: `more_set_headers 'Server: que-mas-te-da-ho';`



HABILITAR / DESHABILITAR LISTADO DE DIRECTORIOS

José Manuel
Redondo López

- La directiva `autoindex on` u `off` habilita o deshabilita el listado de directorios
- Puede usarse en contextos `http`, `server` o `location`
- Dada su peligrosidad, viene desactivada por defecto
- Si queremos activarla tendremos que crear una configuración como ésta

```
server {
        listen   80;
        server_name www.web1.com;
        …
        location / { index  index.php index.html index.htm; }
        location /ficheros_subidos {
                autoindex on;
        }
}
```

Compresión de peticiones

José Manuel
Redondo López

- La compresión de peticiones viene activada por defecto en Nginx
- Puede aplicarse en contextos `http`, `server` y `location`
- Una configuración por defecto puede ser la siguiente:

```
gzip on;
gzip_disable "msie6";
gzip_vary on;
gzip_proxied any;
gzip_comp_level 6;
gzip_buffers 16 8k;
gzip_http_version 1.1;
gzip_types text/plain text/css application/json application/x-javascript text/xml
application/xml application/xml+rss text/javascript;
```

- Debemos recordar no intentar comprimir imágenes y sí ficheros de texto
- El conjunto de directivas de esta opción y su significado está aquí
  - http://nginx.org/en/docs/http/ngx_http_gzip_module.html

Compresión de peticiones: Opciones

José Manuel
Redondo López

- **Algunas de las directivas aceptadas para comprimir peticiones son**
  - `gzip on|off`: Habilita o deshabilita la compresión de peticiones
  - `gzip_buffers number size`: Determina cuantos buffers se usan para comprimir una petición y su tamaño
  - `gzip_comp_level level`: Modifica el nivel de compresión (de 1 a 9)
  - `gzip_disable regex`: Deshabilita la compresión de peticiones para aquellas cuyo `User-Agent` encajen con la especificado en las expresiones regulares
  - `gzip_min_length length`: Modifica el tamaño mínimo necesario para que una petición de comprima, mirando el campo `Content-Length` de la misma
  - `gzip_http_version 1.0|1.1`: Modifica la versión mínima de HTTP requerida para comprimir una petición
  - `gzip_proxied off|expired|no-cache|no-store|private|no_last_modified|no_etag|auth|any`: Habilita o deshabilita la compresión de peticiones para aquellas que pasen por un proxy
    - Se recomienda ver la documentación para entender los posibles valores aceptados
  - `gzip_types mime-type`: Habilita la compresión de peticiones con tipos MIME adicionales a `text/html`, que siempre se comprime. Si se especifica `*`, se comprime todo tipo de contenidos

Páginas de error personalizadas

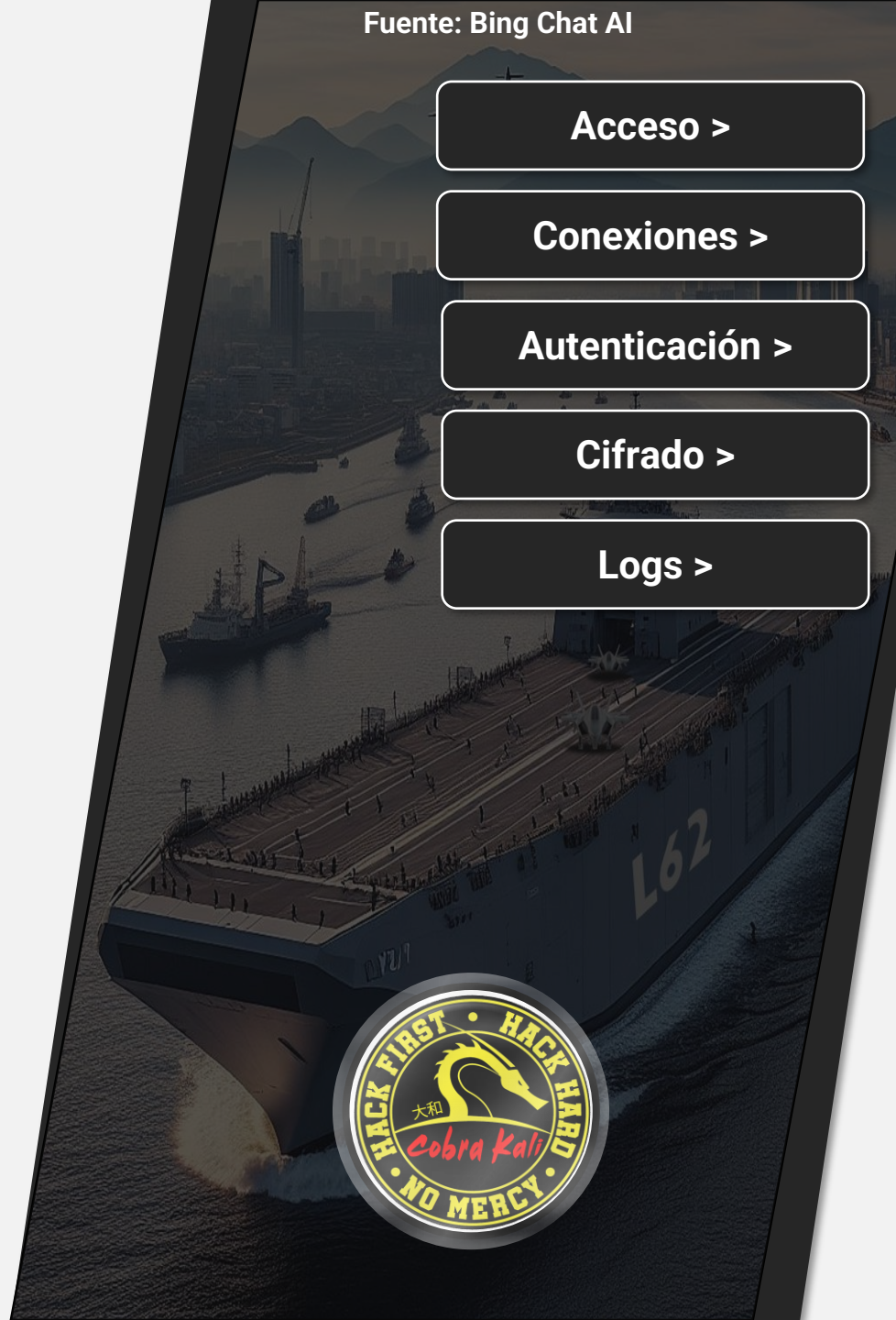
José Manuel
Redondo López

- **Para crear una página de error personalizada es necesario usar la directiva `error_page`**
  - En ella podemos asociar un código de error con una página a mostrar cuando éste se produzca
- **Las páginas de error propias no deberían ser accesibles directamente**
  - Para ello, las podemos crear en `/usr/share/nginx/html`
- **Una vez hecho, creamos una `location` para cada una y decimos que son internas**
  - Solo accesibles con redirecciones Nginx, no directamente
- **Además, en el ejemplo adjunto:**
  - Modificamos el `root` para que la página de error funcione, aunque cambiemos el `root` heredado de un contexto padre
  - Hacemos una llamada a un programa CGI inexistente en la localización `/testing` para provocar errores 500

```
error_page 404 /custom_404.html;
location = /custom_404.html {
        root /usr/share/nginx/html;
        internal;
}
error_page 500 502 503 504 /custom_50x.html;
location = /custom_50x.html {
        root /usr/share/nginx/html;
        internal;
}
location /testing {
        fastcgi_pass unix:/does/not/exist;
}
```

Módulos de Nginx

Ampliando las funcionalidades del servidor web



Módulos Nginx

José Manuel
Redondo López

- **Al igual que Apache, la funcionalidad de Nginx se puede extender con módulos**
  - Averiguar la localización de la que provienen las peticiones para llevarlas a páginas distintas en función de la misma
  - Redimensionar las imágenes para ahorrar ancho de banda
  - Incluir lenguajes de scripting
  - …
- **Hay dos tipos de módulos que se pueden cargar de manera dinámica en Nginx**
  - **Módulos de desarrolladores certificados**: https://www.nginx.com/products/modules/
  - **Módulos de terceros**: https://www.nginx.com/resources/wiki/modules/
  - En algunas distribuciones **no se pueden cargar**, se compilan con el ejecutable (rendimiento)
- **Si el módulo no está en estas listas, puede compilarse y cargarse dinámicamente tal y como indica el siguiente enlace**
  - https://www.nginx.com/resources/wiki/extending/converting/
- **Veremos cómo usar una serie de módulos con funcionalidades muy importantes**

Módulos Nginx

José Manuel
Redondo López

- Si se va a instalar un módulo de un desarrollador certificado, se deben seguir sus instrucciones de instalación específicas
- En caso de que sea un módulo de 3ºs, se instala usando su `module-name`
  - Que podemos encontrar en la entrada del módulo a instalar en el enlace de la transparencia anterior (`apt-get install module-name`)
- Tras eso, en el contexto principal de `/etc/nginx/nginx.conf`, añadimos una directiva `load_module` por cada módulo a añadir
  - Los módulos suelen instalarse en `/etc/nginx/modules`
  - `load_module modules/module-name.so;`
- Finalmente, comprobamos la configuración y recargamos Nginx
  - `nginx -t && nginx -s reload`
- En todo caso, y al igual que en Apache 2, siempre hay que instalar **LOS MÍNIMOS MÓDULOS IMPRESCINDIBLES**

Usuario del servidor, sobreescritura de configuración global, CGI y SSI

José Manuel
Redondo López

- **Nginx** no tiene ficheros `.htaccess` ni equivalentes, por lo que no se puede sobrescribir la configuración global de la forma que lo hace **Apache 2**
- Habilitar **Server Side Includes** (SSI) es con el módulo `ngx_http_ssi_module`,
  - Aunque no implementa todos sus comandos
  - http://nginx.org/en/docs/http/ngx_http_ssi_module.html
  - Por los motivos dichos, **no debe ser instalado** si no es estrictamente necesario
- El soporte del estándar CGI (concretamente su implementación **FastCGI**) se consigue instalando el paquete `fcgiwrap`
  - https://wiki.debian.org/nginx/FastCGI
  - Nuevamente, no debe usarse si no es estrictamente necesario
- Por defecto **Nginx** se ejecuta bajo el usuario `www-data` (usuario y grupo)
  - Por tanto, deben tenerse las mismas precauciones con la asignación de permisos a ficheros / directorios que en **Apache 2**

Restricción de acceso

Restricción de acceso

José Manuel
Redondo López

- **Se puede restringir el acceso a partes de un servidor gracias al módulo `ngx_http_access_module` por estos criterios**
  - IP del cliente (incluyendo subredes)
  - Listas de IPs
  - Nombres de dominio
  - IPV6
  - No se permite restricción por nombre DNS por la penalización de rendimiento que supone de forma nativa, pero si con el módulo `ngx_http_rdns_module`
- **Las reglas de acceso se comprueban en orden**
  - Hasta que encaja la primera
  - En el ejemplo se permite el acceso a las redes IPV4 `20.2.1.0/16` y `192.168.1.0/24` excepto a la dirección `192.168.1.1`
  - También se permite acceder a la red IPv6 `2001:0db8::/32`
- **La sintaxis es:** `allow/deny dirección IP | CIDR | unix: | all;`
- `unix` **representa a los sockets Unix**
  - Conexiones entre aplicaciones en el propio servidor

```
location / {
    deny  192.168.1.1;
    allow 192.168.1.0/24;
    allow 20.2.1.0/16;
    allow 2001:0db8::/32;
    deny all;
}
```

RESTRICCIÓN DE ACCESO A TIPOS DE FICHERO

José Manuel
Redondo López

- Puede impedirse que se sirvan determinados tipos de archivo por medio de contextos, expresiones regulares y restricciones de acceso
- Por ejemplo, esta configuración limita el acceso a una gran cantidad de extensiones de archivos en cualquier localización del servidor:

```
location ~* \.(7z|bak|bash|bz2|config|dist|engine|fla|git|gz|inc|inc|info|ini|install|iso|log
|make|module|profile|psd|py|rar|rb|sh|sql|swp|tar|zip)$ {
    deny all;
}
```

- Este ejemplo prohíbe el acceso a ficheros `.bak`, `.zip` y ficheros del sistema en cualquier localización del servidor (cuyo nombre empieza por `.`)

```
location ~* \.(bak|zip)$|/\. {
    deny all;
}
```

- Con esto es sencillo impedir que se lean potenciales ficheros de configuración que estén accesibles vía web

Limitación de conexiones

LIMITACIÓN DE Nº DE CONEXIONES

José Manuel
Redondo López

- La directiva `limit_conn_zone` se usa para limitar el nº de conexiones
- Con ella definimos un parámetro a usar para limitar las conexiones
  - Y un **nombre** que hace de clave para la zona de memoria compartida que se usará para contar cuantos accesos hace cada cliente en función del parámetro escogido
  - Finalmente, el tamaño de esa zona de memoria
- Debe definirse dentro de un contexto `http`
- Hecho esto, se usa la directiva `limit_conn` dentro de un contexto `location`, `server` o `http`
  - Especificando el nombre de la zona y el nº de conexiones/seg. admitidas
- Esta opción, junto con la limitación de la ratio de peticiones y del ancho de banda pueden servir para escudarnos en la medida de lo posible contra ataques DoS
  - Equivale al `mod_evasive` de Apache 2, módulo que no tiene Nginx

LIMITACIÓN DE Nº DE CONEXIONES

José Manuel
Redondo López

- **Ejemplo**: Limitar la localización /descargas a una conexión por IP, usando 10Mb para guardar los contadores de accesos de los clientes

```
limit_conn_zone $binary_remote_addr zone=addr:10m;
…
location /download/ {
    limit_conn addr 1;
}
```

- **Ejemplo**: Limitar las conexiones de un mismo servidor a 1000

```
http {
    limit_conn_zone $server_name zone=servers:10m;
    server {
        limit_conn servers 1000;
    }
}
```

LIMITACIÓN DE RATIO DE PETICIONES

José Manuel
Redondo López

- Para limitar la ratio de peticiones, hay que usar un par de directivas `limit_req_zone` y `limit_req` muy similares a las anteriores, dentro de los mismos contextos mencionados
  - Con la primera, nuevamente especificamos **la variable** que controlará la ratio de peticiones para cada cliente, un **nombre clave de la zona de memoria** que contendrá los contadores **y la ratio** por segundo (r/s) o por minuto (r/m) admitido

```
limit_req_zone $binary_remote_addr zone=one:10m rate=1r/s;
```

  - Una vez definido el nombre de la zona de memoria basta con usar la directiva `limit_req` con el nombre especificado y el número de peticiones que pueden quedar a la espera de ser procesadas (`burst`)

```
location /search/ {
    limit_req zone=one burst=5;
}
```

- Si se define el parámetro `burst`, el procesamiento de las peticiones en espera se retrasa para que la ratio de peticiones sea el especificado
  - Salvo que se añada `nodelay` a esta línea
- Cualquier petición que supere la ratio de peticiones especificado + el nº de peticiones en espera dará un error 503

LIMITACIÓN DE ANCHO DE BANDA

José Manuel
Redondo López

- La directiva `limit_rate` limita el ancho de banda por conexión al servidor

```
location /download/ {
    limit_rate 50k;
}
```

- El ejemplo anterior limita cada conexión que establezca un cliente a un ancho de banda de 50Kb/seg
- Como todo cliente puede establecer múltiples conexiones, se puede combinar con una limitación del nº de conexiones ya visto
- También es posible establecer un límite tras transferir una determinada cantidad de información con la directiva `limit_rate_after`

```
limit_rate_after 500k;
limit_rate 20k;
```

Autenticación

AUTENTICACIÓN

José Manuel
Redondo López

- Recordamos que consiste en requerir una identidad de usuario para ver ciertos contenidos
- Se puede conseguir con medios locales o externos
  - **Locales**: Ficheros de diferente tipo existentes en el disco duro de la máquina
    - Basic, Digest, sobre `/etc/shadow`…
  - **Externos**: Datos de usuarios en LDAPs, Active Directory, BBDD MySQL,…
- Nginx soporta los mismos modos de autenticación que Apache 2 a través de módulos:
  - **Basic**: https://www.nginx.com/resources/admin-guide/restricting-access-auth-basic/
  - **Contra usuarios del SO y contra LDAP** (a través de PAM, Pluggable Authentication Modules):
    - http://www.doublecloud.org/2014/01/nginx-with-pam-authentication/
    - https://unix.stackexchange.com/questions/165520/authenticate-users-via-ldap-using-nginx

AUTENTICACIÓN BÁSICA

José Manuel
Redondo López

- La autenticación básica en **Nginx** funciona de la misma forma que la que vimos con **Apache 2**, incluso usando las mismas herramientas
- Estas se pueden instalar con el paquete `apache2-utils`, sin falta de instalar **Apache 2**
  - En **RHEL** la utilidad `htpasswd` ya está instalada de base
- Para crear el fichero de pares usuario / clave debe usarse la opción `-c`

```
sudo htpasswd -c /etc/nginx/.htpasswd usuario1
```

- Y luego ya crear todos los usuarios / claves que sean necesarios:

```
sudo htpasswd /etc/nginx/.htpasswd usuario2
```

- Ahoya ya podemos proteger el acceso a cualquier localización con las directivas siguientes:

```
 location /protegido {
    auth_basic “Area restringida”; #Activa autenticación básica y especifica el mensaje
que aparecerá cuando se pida usuario y clave
    auth_basic_user_file /etc/nginx/.htpasswd; # Path al fichero donde están usuario y
clave
    }
```

AUTENTICACIÓN BÁSICA

José Manuel
Redondo López

- Otra opción común es limitar el acceso a toda una web salvo determinadas áreas
- Esto puede lograrse gracias a la forma de trabajar que tienen los contextos, redefiniendo el tipo de autenticación cuando sea necesario

```
server {
    auth_basic            " Area restringida";
    auth_basic_user_file /etc/nginix/htpasswd;

    location /public/ {
        auth_basic off;
    }
}
```

- Este ejemplo serviría los ficheros `index.html` y `/help/manuals.zip` sin pedir usuario y clave

```
location ~ (/index\.html|/help/manuals\.zip)
{
   auth_basic off;
    allow all; # Allow all para ver el contenido
}
```

Autenticación con PAM y usuarios del SO

José Manuel
Redondo López

- **Al igual que con Apache, el uso de PAM (Pluggable Authentication Modules) permite a Nginx autenticar usuarios sobre diversas fuentes**
  - Como un LDAP o los usuarios del SO
  - Este último ejemplo será el que veamos
- **Para ello es necesario instalar el paquete `nginx-extras`**
  - Hecho esto, es necesario que el usuario por defecto que se usa para servir las peticiones en Nginx pueda acceder al fichero `shadow`
  - La forma más sencilla de hacerlo es añadirlo al grupo `shadow`

```
usermod -a -G shadow www-data
```

- **En sistemas Red Hat, se requiere recompilar el ejecutable con el soporte de este módulo de 3ºs**
  - https://github.com/sto/ngx_http_auth_pam_module
  - Y hacer que el usuario por defecto (`nobody` o `nginx`) pueda leer el fichero `/etc/shadow` incluyéndolo en el grupo `root`
  - Dado que es un proceso potencialmente inseguro, **se desaconseja usarlo**

AUTENTICACIÓN CON PAM Y USUARIOS DEL SO

José Manuel
Redondo López

- Luego creamos un fichero en `/etc/pam.d` con el siguiente contenido
  - Llamaremos al fichero `nginx`

```
@include common-auth
```

- Ahora basta con modificar la localización que queramos en `/etc/nginx/nginx.conf` para que use este tipo de autenticación

```
location / {
    auth_pam    “Zona segura";
    auth_pam_service_name   "nginx";
}
```

- Conviene recordar que **ninguno de estos métodos cifra sus transmisiones**, por lo que es necesario cifrarlos mediante el uso de TLS, que veremos a continuación

AUTENTICACIÓN BÁSICA Y RESTRICCIONES DE ACCESO

José Manuel
Redondo López

- **Es sencillo combinar autenticación y las restricciones de accesos vistas para solo permitir el acceso a una localización a usuarios autenticados que vengan de IPs autorizadas**
  - El ejemplo siguiente permite accede a los usuarios de la red `192.168.1.1/24` salvo a la IP `192.168.1.2`
  - La directiva `safisfy all` especifica que los usuarios deben estar autorizados Y proceder de una IP permitida
  - Si en lugar de `all` se especifica `any`, entonces basta con que el usuario cumpla una de las dos condiciones

```
location /status {
    satisfy all;
    deny  192.168.1.2;
    allow 192.168.1.1/24;
    allow 127.0.0.1;
    deny  all;
    auth_basic            “Protegido";
    auth_basic_user_file /etc/nginx/htpasswd;
}
```

Cifrado de conexiones

SOBRE CIFRADO…

José Manuel
Redondo López

- Ya hemos hablado de la importancia del **buen** cifrado
- Es especialmente importante en el caso de los dos métodos de autenticación anteriores, donde los datos del usuario se transmiten en texto plano
- **Nginx** por supuesto soporta el uso de TLS con mecanismos de cifrado fuertes y no obsoletos
- A continuación, veremos cómo poner en marcha HTTPs con un certificado auto-firmado
  - **Más información**:
    - http://nginx.org/en/docs/http/configuring_https_servers.html
    - https://www.digitalocean.com/community/tutorials/how-to-create-a-self-signed-ssl-certificate-for-nginx-in-ubuntu-16-04

Cifrado: Crear un certificado autofirmado

José Manuel
Redondo López

- Podemos crear un certificado auto-firmado con OpenSSL así

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/ssl/private/nginx-self.key -out /etc/ssl/certs/nginx-self.crt
```

- `req -x509` especifica que el certificado autofirmado será de tipo X.509 y que no generará una certificate signing request (CSR), al ser auto-firmado
- `-nodes` Indica a OpenSSL que no se proteja el fichero con el certificado con una clave, para que Nginx pueda leerlo directamente
- `-days 365` Nº de días durante los cuales el certificado se considera válido
- `-newkey rsa:2048` Indica que queremos generar un certificado y una nueva clave de firmado para el certificado al mismo tiempo
  - La clave generada será de tipo RSA y de 2048 bits
  - Debemos mirar cuál es la longitud recomendada en la actualidad
- `-keyout` Especifica donde se genera el fichero con la clave privada generada

Cifrado: Crear un certificado autofirmado

José Manuel
Redondo López

- La ejecución de esta orden generará un fichero con la clave privada y un certificado en subdirectorios de `/etc/ssl`
- El proceso de creación hará además una serie de preguntas a responder
  - La más importante es la que se refiere al `Common Name`, donde debemos responder el nombre DNS o la IP del servidor (como es nuestro caso)
- Para mejorar la seguridad de las comunicaciones, activaremos PFS (**Perfect Forward Secrecy**
  - https://es.wikipedia.org/wiki/Perfect_forward_secrecy
  - Mediante la siguiente orden

```
sudo openssl dhparam -out /etc/ssl/certs/dhparam.pem 2048
```

Configurando Nginx para usar TLS

José Manuel
Redondo López

- Generados los certificados, debemos modificar la configuración de Nginx para que los use. Para ello:
  - Generaremos una configuración que indique **donde está nuestra clave privada y el certificado**
  - Generaremos otra configuración que configure un **cifrado fuerte y adecuado**
  - Modificaremos nuestros server blocks para que usen las dos configuraciones anteriores, reutilizando las mismas
- Para ello, lo primero es crear el fichero `/etc/nginx/snippets/self-signed.conf` con el siguiente contenido

```
ssl_certificate /etc/ssl/certs/nginx-self.crt;
ssl_certificate_key /etc/ssl/private/nginx-self.key;
```

- Como dijimos anteriormente, el directorio `snippets` es el adecuado para crear configuraciones reutilizables

Configurando Nginx para usar TLS

José Manuel
Redondo López

- Ahora haremos lo propio con los parámetros indicados para un cifrado fuerte
  - En el fichero `/etc/nginx/snippets/ssl-params.conf`
- Para ello podemos usar la configuración “**Modern**” de https://ssl-config.mozilla.org/
  - Seleccionando las versiones correctas del Nginx y OpenSSL
  - Estudiando como añadir el fichero `dhparam.pem` generado anteriormente para usar PFS

mozilla SSL Configuration Generator

Server Software

○ Apache	○ MySQL
○ AWS ALB	◉ nginx
○ AWS ELB	○ Oracle HTTP
○ Caddy	○ Postfix
○ Dovecot	○ PostgreSQL
○ Exim	○ ProFTPD
○ Go	○ Redis
○ HAProxy	○ Squid
○ Jetty	○ Tomcat
○ lighttpd	○ Traefik

Mozilla Configuration

- ◉ Modern
  Services with clients that support TLS 1.3 and don't need backward compatibility
- ○ Intermediate
  General-purpose servers with a variety of clients, recommended for almost all systems
- ○ Old
  Compatible with a number of very old clients, and should be used only as a last resort

Environment

Server Version	1.17.7
OpenSSL Version	1.1.1k

Miscellaneous

- ☑ HTTP Strict Transport Security
  This also redirects to HTTPS, if possible
- ☑ OCSP Stapling

nginx 1.17.7, modern config, OpenSSL 1.1.1k

Supports Firefox 63, Android 10.0, Chrome 70, Edge 75, Java 11, OpenSSL 1.1.1, Opera 57, and Safari 12.1

```
# generated 2023-09-02, Mozilla Guideline v5.7, nginx 1.17.7, OpenSSL 1.1.1k, modern configuration
# https://ssl-config.mozilla.org/#server=nginx&version=1.17.7&config=modern&openssl=1.1.1k&guideline=5.7
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    location / {
        return 301 https://$host$request_uri;
    }
}

server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;

    ssl_certificate /path/to/signed_cert_plus_intermediates;
    ssl_certificate_key /path/to/private_key;
    ssl_session_timeout 1d;
    ssl_session_cache shared:MozSSL:10m;  # about 40000 sessions
    ssl_session_tickets off;

    # modern configuration
```

ACTIVANDO TLS

José Manuel
Redondo López

- Un certificado autofirmado no usará el método SSL stapling para averiguar si ha sido revocado, pero simplemente generará un warning por lo que no es necesario desactivarlo
- Ahora ya podemos configurar el sitio web que queramos para que use TLS

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    server_name <IP o nombre DNS>;
    return 302 https://$server_name$request_uri;
}
server {
    listen 443 ssl http2 default_server;
    listen [::]:443 ssl http2 default_server;
    include snippets/self-signed.conf;
    include snippets/ssl-params.conf;
}
```

Activando TLS

José Manuel
Redondo López

- En la configuración anterior, cualquier acceso por HTTP se redirecciona automáticamente a HTTPs
- Es decir, el tráfico cifrado es obligatorio, lo cual debería ser el comportamiento por defecto en la mayoría de las ocasiones
- Además, se ha configurado el acceso vía HTTPs para usar HTTP/2, más seguro y eficiente
  - Nginx soporta esta versión del protocolo directamente
  - También podemos investigar cómo usar el más moderno HTTP/3 si ya no tiene soporte experimental
    - https://es.wikipedia.org/wiki/HTTP/3
    - https://nginx.org/en/docs/http/ngx_http_v3_module.html
- Y, además, estamos usando las configuraciones vistas anteriormente
- Con esto el servidor complete tendría tráfico cifrado de forma adecuada

ACTIVANDO TLS

José Manuel
Redondo López

- Si por alguna razón necesitásemos mantener el acceso por HTTP y HTTPs, la configuración sería la siguiente
- No obstante, esta forma de proceder NO se recomienda

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    listen 443 ssl http2 default_server;
    listen [::]:443 ssl http2 default_server;

    server_name <IP o nombre DNS>;
    include snippets/self-signed.conf;
    include snippets/ssl-params.conf;
```

- Debemos comprobar que el firewall tiene abierto el puerto 443 y reiniciar Nginx
- Además, si la configuración funciona es mejor hacer una redirección permanente (301) en lugar de la 302

Logs


Fuente: Bing Chat AI
L62
HACK FIRST
HACK HARD
NO MERCY
大和
Cobra Kali

Análisis de Logs: uso de Awstats

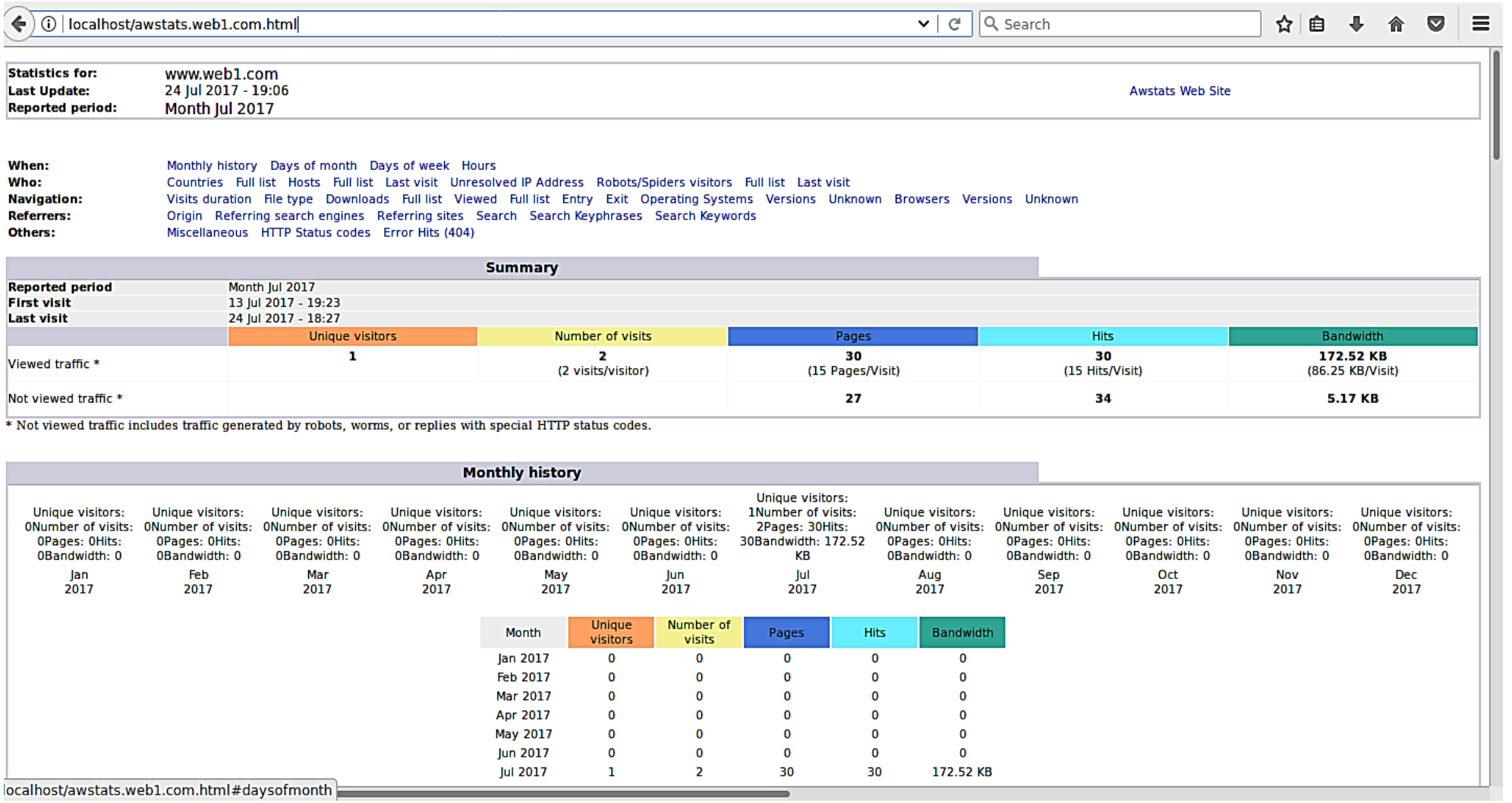
José Manuel
Redondo López

- **Awstats puede usarse con los logs de Nginx**
  - Para Linux tipo Red Hat: https://tecadmin.net/steps-to-configure-awstats-on-centos-and-rhel-system/
- **El siguiente tutorial muestra cómo usar awstats en un servidor en producción con logs rotatorios**
  - https://luxagraf.net/src/awstats-nginx-ubuntu-debian
  - Es preferible hacer una copia del fichero de configuración central al local
  - Y añadir `LogFormat=4` a los ficheros de configuración de los sitios sobre los que queramos usar la herramienta
- **Nosotros usaremos la opción más simple vista con Apache 2 para ver los resultados de awstats**
  - Las imágenes adjuntas muestran como generar una página HTML estática con los resultados

Análisis de Logs: uso de Awstats

José Manuel
Redondo López

- Debemos recordar que este tipo de información no debe ser expuesta al público



< Ir al Índice

Fuente: Bing Chat AI

Mod_security >

DDoS >

Lenguajes y URLs >

Otros módulos >

Módulos avanzados de Nginx

Extendiendo la funcionalidad del servidor al máximo

Departamento de Informática
Universidad de Oviedo

Universidad de Oviedo

Mod_security

Un WAF para nuestro Nginx

MOD_SECURITY

José Manuel
Redondo López

- **Nginx también puede hacer uso de este módulo de seguridad avanzado que ya vimos con Apache, con las mismas funcionalidades**
- **Pero o bien usamos la versión de pago (Nginx Plus) o recompilamos Nginx con el soporte para dicho módulo activo**
  - La aproximación más conveniente es compilar un Nginx con `mod_security` en un servidor aislado, que pueda hacer de proxy de otros que no lo tengan añadido
  - O bien usar un Nginx + `mod_security` como base para hacer el resto de las operaciones de configuración del servidor
  - No obstante, compilar Nginx es un trabajo **largo y laborioso**
    - https://www.linuxbabe.com/security/modsecurity-nginx-debian-ubuntu
- **Otras opciones**
  - Usar Apache 2 + `mod_security2`, más sencillo de instalar
  - Usar **un contenedor Docker** ("L-62 Princesa de Asturias") con todo el software y reglas ya instalado (**recomendado**)
    - https://hub.docker.com/r/owasp/modsecurity-crs

Protección contra DDoS

Módulos que pueden evitar que nos colapsen el servidor

Fail2Ban

José Manuel
Redondo López

- **Este programa, que detecta y bloquea intentos de intrusión analizando el log de peticiones/errores, puede usarse también con Nginx**
- **El modo de instalación es el ya visto:** `sudo apt-get install fail2ban`
  - Podemos seguir el siguiente tutorial
    - https://www.digitalocean.com/community/tutorials/how-to-protect-an-nginx-server-with-fail2ban-on-ubuntu-20-04
- **Los parámetros `findtime` y `maxretry`**
  - Si un cliente hace más de `maxretry` operaciones maliciosas en el intervalo de tiempo especificado por `findtime` en segundos, se bloqueará
  - Ej.: Intentar autenticarse erróneamente más de 6 veces en una hora

```
findtime = 3600
maxretry = 6
```

Fail2Ban

José Manuel
Redondo López

- **No obstante, se debe modificar la configuración para indicar a `fail2ban` qué logs del servidor debe monitorizar**
  - Y qué hacer cuando detecte alguna operación maliciosa
- **Esto puede hacerse copiando el fichero `/etc/fail2ban/jail.conf` a `/etc/fail2ban/jail.local`**
  - Para que actualizaciones del producto no sobrescriban el fichero por defecto de configuración
- **Cosas a cambiar en `/etc/fail2ban/jail.local`**
  - **Qué clientes no están afectados** por `fail2ban`: Es conveniente añadir la propia IP del servidor para evitar auto-bloquearnos

    ```
    ignoreip = 127.0.0.1/8
    ```
  - **El tiempo en segundos** que un cliente que ha efectuado una operación maliciosa está **bloqueado**. Por defecto es 10 minutos

    ```
    bantime = 3600
    ```

Fail2Ban y monitorización de logs

José Manuel
Redondo López

- Como vimos en la parte de Apache 2, las acciones de `fail2ban` se configuran por medio de jails
  - Cada jail monitoriza los logs del servidor en busca de patrones de comportamientos concretos
  - Cada jail se configura en `/etc/fail2ban/jail.local` y comienza por una cabecera con su nombre entre `[ ]` seguido de una serie de configuraciones
- El siguiente ejemplo es una jail llamada `nginx-http-auth` para monitorizar intentos de acceso erróneos

```
[nginx-http-auth]
enabled  = true
filter   = nginx-http-auth
port     = http,https
logpath  = /var/log/nginx/error.log
```

- Esta jail es la única incluida en Ubuntu específica de Nginx, pero podemos crear más como veremos en las siguientes transparencias

FAIL2BAN Y JAILS PERSONALIZADAS

José Manuel
Redondo López

- La siguiente jail bloquea a clientes que buscan ciertos scripts en las webs para ejecutar exploits

```
[nginx-noscript]
enabled  = true
port     = http,https
filter   = nginx-noscript
logpath  = /var/log/nginx/access.log
maxretry = 6
```

- Esta jail está preparada para bloquear peticiones maliciosas de bots conocidas

```
[nginx-badbots]
enabled  = true
port     = http,https
filter   = nginx-badbots
logpath  = /var/log/nginx/access.log
maxretry = 2
```

Fail2Ban y jails personalizadas

José Manuel
Redondo López

- Si no se sirven contenidos desde directorios personalizados de los usuarios, esta jail prohíbe este tipo de accesos

```
[nginx-nohome]
enabled  = true
port     = http,https
filter   = nginx-nohome
logpath  = /var/log/nginx/access.log
maxretry = 2
```

- Y. finalmente, esta jail evitará que nuestro Nginx se use como un proxy abierto

```
[nginx-noproxy]
enabled  = true
port     = http,https
filter   = nginx-noproxy
logpath  = /var/log/nginx/access.log
maxretry = 2
```

Fail2Ban y jails personalizadas

José Manuel
Redondo López

- **Las jails personalizadas que hemos visto no están completas si no se añaden filtros que correspondan a cada una de ellas**
  - Estos filtros son los patrones con expresiones regulares que `fail2ban` debe buscar en los logs para bloquear o no las peticiones de los clientes
  - Todos los filtros están en el directorio `/etc/fail2ban/filter.d`
- **Abrimos el único suministrado por Ubuntu (`nginx-http-auth.conf`)**
  - Y añadimos una línea más debajo de `failregex` que contemple que el usuario no introduzca ninguna clave

```
[Definition]
failregex = ^ \[error\] \d+#\d+: \*\d+ user "\S+":? (password mismatch|was not found in ".*"), client:
<HOST>, server: \S+, request: "\S+ \S+ HTTP/\d+\.\d+", host: "\S+"\s*$
            ^ \[error\] \d+#\d+: \*\d+ no user/password was provided for basic authentication, client:
<HOST>, server: \S+, request: "\S+ \S+ HTTP/\d+\.\d+", host: "\S+"\s*$
ignoreregex =
```

Fail2Ban y jails personalizadas

José Manuel
Redondo López

- Con la distribución de `fail2ban` tenemos un fichero `apachapache-badbots.conf`
- Haremos una copia con otro nombre para poder usarlo con la jail definida anteriormente, aunque podríamos usarlo tal cual: `sudo cp apache-badbots.conf nginx-badbots.conf`
- Seguidamente editamos `nginx-noscript.conf` dentro del directorio mencionado
  - Para bloquear extensiones de fichero de lenguajes que no tendríamos instalados en el servidor

```
[Definition]
failregex = ^<HOST> -.*GET.*(\.php|\.asp|\.exe|\.pl|\.cgi|\.scgi)
ignoreregex =
```

- Es conveniente darse cuenta de que si alguien nos pide una extensión que no tenemos instalada, no será una petición legítima
  - Sino alguien buscando información de nuestro servidor, de ahí que se bloquee

FAIL2BAN Y JAILS PERSONALIZADAS

José Manuel
Redondo López

- Creamos ahora un fichero `nginx-nohome.conf` para evitar accesos a directorios web de usuarios

```
[Definition]
failregex = ^<HOST> -.*GET .*/~.*
ignoreregex =
```

- Finalmente, creamos el fichero `nginx-noproxy.conf` para evitar intentos de usar nuestro servidor como proxy

```
[Definition]
failregex = ^<HOST> -.*GET http.*
ignoreregex =
```

- Terminada la definición de jails y reglas, debemos reiniciar el servidor para que tengan efecto

Lenguajes y URLs

Funcionalidades para manipular las peticiones y respuestas

Lenguajes del servidor

José Manuel
Redondo López

- **Al igual que con Apache 2, lo mejor es instalar 1 sólo lenguaje en cada servidor**
  - Por los principios de seguridad vistos
- **Si no es posible, mantener server blocks separados para cada lenguaje**
- **El soporte de lenguajes de servidor requiere la instalación de módulos:**
  - **PHP** (LEMP Stack): https://www.digitalocean.com/community/tutorials/como-instalar-linux-nginx-mysql-php-lemp-stack-in-ubuntu-16-04-es
  - **JSP** (en combinación con Tomcat): http://www.programering.com/a/MDMwIDNwATc.html
  - **.NET**: Para usar páginas .aspx y similares debe instalarse el producto FastCGI Mono Server y hacer que Nginx le pase peticiones al mismo
  - **Más información:**
    - http://www.mono-project.com/docs/web/fastcgi/
    - https://www.nginx.com/resources/wiki/start/topics/examples/mono/
    - http://www.mono-project.com/docs/web/fastcgi/nginx/

REWRITE RULES

José Manuel
Redondo López

- **Nginx** tiene un motor de reglas para la reescritura de URLs con una funcionalidad comparable a la de `mod_rewrite` de **Apache 2**
- Se implementa en el módulo `ngx_http_rewrite_module`
  - https://nginx.org/en/docs/http/ngx_http_rewrite_module.html
- Podemos encontrar una descripción de sus funcionalidades en: https://www.nginx.com/blog/creating-nginx-rewrite-rules/
- Ejemplos de uso
  - **Reescritura de URLs** para dejarlas más “bonitas”/usables/entendibles
  - **Eliminar partes** de la dirección
  - **Evitar el *hotlinking***
  - Acceder a **páginas de error personalizadas**

REWRITE RULES

José Manuel
Redondo López

- **Proteger el acceso** a directorios y tipos de fichero concretos
- **Restringir los métodos HTTP** admitidos
- **Prohibir el acceso** mediante proxies
- **Prohibir versiones concretas** del protocolo HTTP
- **Limitar los caracteres** presentes en las URI
- **Prohibir peticiones erróneas**/corruptas/mal formadas
- **Manipulación de los parámetros** de la petición (*querystring*)
- **Bloquear a clientes** que se conectan desde una lista de IPs concretas (**lista de bloqueo**)
  - Y lo contrario (**lista de permiso**) también
  - Por ejemplo, como lo que hicimos para clientes provenientes de la red ToR

REWRITE RULES

José Manuel
Redondo López

- Es posible convertir reglas de `mod_rewrite` en **rewrite rules** de **Nginx**
  - https://www.nginx.com/blog/converting-apache-to-nginx-rewrite-rules/
- Sin embargo, para entender este proceso es necesario entender las directivas siguientes
  - `return`
  - `rewrite`
  - `try_files`
- La última directiva no pertenece al módulo de reescritura de URLs, sino al `ngx_http_core_module` de **Nginx**
  - http://nginx.org/en/docs/http/ ngx_http_core_module.html
- Veremos la descripción de las mismas a continuación

La directiva return

José Manuel
Redondo López

- Esta directiva se encuentra dentro de contextos `server` o `location` y define la URL correcta a la hora de atender la petición que representan estos contextos
- Este ejemplo indica que cualquier petición HTTP o HTTPs a www.dominioviejo.com se transforme en una petición a www.dominionuevo.com
  - Manteniendo el protocolo (variable `$scheme`) y la URI de la petición (variable `$request_uri`):

```
server {
    listen 80;
    listen 443 ssl;
    server_name www.dominioviejo.com;
    return 301 $scheme://www.dominionuevo.com$request_uri;
}
```

- Más información: http://nginx.org/en/docs/http/ngx_http_rewrite_module.html#return

La directiva rewrite

José Manuel
Redondo López

- Directiva para hacer reescrituras avanzadas de URLs
- Se debe especificar en los mismos contextos que la directiva `return` y tiene la siguiente sintaxis: `rewrite regex URL [flag];`
  - `regex`: Indica a Nginx que solo debe hacer la reescritura si la URL encaja con esta expresión regular (además de con el contexto `server` o `location` que la contiene)
  - `URL`: es la URL reescrita a la que redirigirse
  - `flag` (opcional) según su valor indica que
    - `last`: Detiene el procesamiento del conjunto de directivas actual y busca una nueva `location` en todo el fichero que encaje con la URL cambiada
    - `break`: Detiene el procesamiento del conjunto de reglas actual
    - `redirect`: Devuelve un código de redirección 302, usando si la URL de reemplazo no empieza por `http://`, `https://`, o `$scheme`
    - `permanent`: Devuelve un código de redirección permanente (301)

La directiva rewrite

José Manuel
Redondo López

- Esta directiva no detiene el procesamiento como la `return` ni manda una redirección al cliente salvo que se especifique con flags
- Si no se envía una redirección, Nginx procesa todo el fichero de configuración en busca de directivas que están definidas en el módulo `ngx_http_rewrite_module`
- Procesa todas las que encuentra en orden, de forma que, si una URL reescrita encaja con una directiva, hace la acción pertinente y sigue procesando
  - De esta forma se pueden dar un gran nº de reescrituras encadenadas
- Por este motivo, es muy importante pensar bien el orden de las directivas `rewrite`, ya que se puede entrar en un procesamiento en bucle
- El módulo tiene un límite de 10 reescrituras

La directiva rewrite

José Manuel
Redondo López

- El siguiente ejemplo captura peticiones que lleven `/multimedia` y luego las cadenas `/audio/` o `/video/`
- Reemplaza dichas URL por otra que sustituya esas cadenas por `/mp3/` o `/mp4/` y añade la extensión `.mp3` o `.mp4`
- Las variables `$1` y `$2` contienen los elementos que no cambian
- La URL `/multimedia/verano/media/despacito` se convierte en `/multimedia/verano/mp3/despacito.mp3`
- Si el fichero tiene otra extensión, la reemplaza por la indicada en cada caso

```
server {
    rewrite ^(/multimedia/.*)/audio/(\w+)\.?.*$ $1/mp3/$2.mp3 last;
    rewrite ^(/multimedia/.*)/video/(\w+)\.?.*$ $1/mp4/$2.mp4 last;
    return  403;
}
```

La directiva rewrite

José Manuel
Redondo López

- Si estas reglas anteriores se crean dentro de una `location /multimedia`, el procesamiento entrará en bucle y tras 10 reescrituras devolverá un error 500
  - Para evitarlo, tendríamos que reemplazar `last` por `break`
- El `return` final indica que si la URL no encaja con ninguno de los patrones de las directivas `rewrite` se retorne un código 403 al cliente
- `rewrite` por sí solo puede enviar 301 o 302, tenemos que recurrir a este "truco" para devolver otros códigos
- Más información: http://nginx.org/en/docs/http/ngx_http_rewrite_module.html#rewrite

La directiva try_files

José Manuel
Redondo López

- **Esta directiva, que también vale en contextos `server` o `location`, acepta una lista de ficheros y/o directorios y una URL final: `try_files file ... uri;`**
  - Nginx comprueba la existencia de los ficheros o directorios (nombres acabados en /) indicados en orden
  - Para ello construye el path completo de los mismos usando las directivas `root` y `alias` existentes
  - Sirve el primero que encuentra en la lista identificada
  - Si ninguno de los ficheros o directorios encaja, se hace una redirección a la URI
- **Para que funcione, debe especificarse un bloque `location` que capture la redirección a esta URI final**
  - Este bloque `location` puede ser una localización con nombre (cuyo especificador de localización comienza por `@`)
  - Suele usarse la variable `$uri` para hacer la redirección, al contener la parte de la URL que va detrás del nombre del dominio

La directiva try_files

José Manuel
Redondo López

- **El siguiente ejemplo indica que si se intenta accede a una imagen o carpeta no existente en la localización `/imagenes/`, se devuelva una imagen por defecto**
  - Nginx intenta primero www.midominio.com/imagenes/foto.jpg
  - Si no existe, luego intenta www.midominio.com/imagenes/foto.jpg/
  - Si no existe, sirve www.midominio.com/imagenes/default.jpg indicando además que se meta en una caché durante 30 segundos

```
location /images/ {
    try_files $uri $uri/ /images/default.jpg;
}
location = /images/default.jpg {
    expires 30s;
}
```

- **Más información:** http://nginx.org/en/docs/http/ngx_http_core_module.html#try_files

Ejemplos de rewrite rules

Ejemplos: Cambiando el nombre de dominio

José Manuel
Redondo López

- Consiste en una redirección de un dominio viejo a uno nuevo, usando variables del servidor ya vistas

```
server {
    listen 80;
    listen 443 ssl;
    server_name www.dominioviejo.com dominioviejo.com;
    return 301 $scheme://www.dominionuevo.com$request_uri;
}
```

- Esta forma de redirigir es adecuada si la estructura del nuevo dominio es idéntica a la del viejo
  - Ej.: Redirecciones de www.dominioviejo.com/usuarios se hacen a www.dominionuevo/usuarios
- Si no es así, es mejor quitar el parámetro `$request_uri`

EJEMPLOS: QUITANDO EL PREFIJO WWW

José Manuel
Redondo López

- Añadir www

```
server {
    listen 80;
    listen 443 ssl;
    server_name dominio.com;
    return 301 $scheme://www.dominio.com$request_uri;
}
```

- Quitar www

```
server {
    listen 80;
    listen 443 ssl;
    server_name www.dominio.com;
    return 301 $scheme://dominio.com$request_uri;
}
```

Ejemplos: Redirección de tráfico

José Manuel
Redondo López

- Este ejemplo redirige el tráfico que va a una página desconocida del servidor a la página principal del mismo
- Se supone que la petición no encaja con ningún bloque `server` o `location` existente
- Todas las peticiones que no encajen con ningún bloque acaban aquí
  - Gracias al uso del parámetro `default_server` visto
  - Y el uso del nombre de servidor _, que no especifica ningún nombre de servidor particular

```
server {
    listen 80 default_server;
    listen 443 ssl default_server;
    server_name _;
    return 301 $scheme://www.domain.com;
}
```

Ejemplos: Prohibir peticiones a ficheros no permitidos

José Manuel
Redondo López

- Ya hemos visto un ejemplo de cómo no permitir acceso a extensiones de ficheros que no soportamos para cortar posibles intentos de exploración del servidor
- Pero en este caso en lugar de denegar el acceso simplemente devolveremos un código de error

```
location ~ \.(aspx|php|jsp|cgi)$ {
    return 410;
}
```

- El código 410 indica al cliente que el recurso pedido está permanentemente fuera de servicio, para que no vuelvan a hacer la petición
  - Es la ventaja sobre el código 404 Not Found
- Una denegación de acceso (`deny all`) devuelve un código 403, indicando que el recurso existe, pero que no se tiene permiso para el
  - Por tanto, esta opción es mejor para indicar que algo no es accesible (no reconocemos su existencia)

Ejemplos: Redirección de peticiones

José Manuel
Redondo López

- En este ejemplo, una petición a una URL con un código (http://dominio.com/clientes/123) se convierte a un equivalente
- De forma que se pase a una página que controla qué hacer con cada uno de estos códigos (http://dominio.com/clientes.html?id=123)

```
rewrite ^/clientes/(.*)$ /clientes.html?id=$1 last;
```

REWRITE RULES: IF

José Manuel
Redondo López

- `if` es una directiva de las **rewrite rules** que permite evaluar una condición
- Si es cierta, se ejecutan las directivas contenidas dentro del bloque `{}` que contiene
- Las condiciones pueden ser
  - El nombre de una variable, que se evalúa a **falso** si está vacía o vale "0"
  - Comparar el valor de una variable con una cadena **usando los operadores "=" y "!="**
  - Comprobar el valor de una variable con una expresión regular con los operadores "`~`" (**sensible a mayúsculas**) o "`~*`" (lo contrario)
    - Las expresiones regulares pueden capturar partes que se usen posteriormente con las variables **`$1`**..**`$9`**
    - Existen también los operadores negativos equivalentes "`!~`" y "`!~*`"
    - Si una expresión regular incluye los caracteres "}" o ";", debe ponerse entre comillas dobles

REWRITE RULES: IF

José Manuel
Redondo López

- **Más operadores**
  - Comprobar si un fichero existe o no con los operadores "`-f`" and "`!-f`"
  - Ídem para un directorio con "`-d`" y "`!-d`"
  - Comprobar la existencia de un fichero, directorio o enlace simbólico con los operadores "`-e`" y "`!-e`"
  - Comprobar si existe un ejecutable con los operadores "`-x`" and "`!-x`"
- **El problema de usar if es que el propio fabricante lo desaconseja por la dificultad que tiene usarlo correctamente:**
  - https://www.nginx.com/resources/wiki/start/topics/depth/ifisevil/
- **Es más, recomienda cambiarlo por combinaciones de `return`, `rewrite` y `try_files`**
- **No obstante, la documentación complete de la directiva se puede encontrar aquí:**
  - http://nginx.org/en/docs/http/ngx_http_rewrite_module.html#if
- **Veremos un ejemplo de uso de `if` posteriormente para evitar hotlinking**

EVITAR EL USO DE MÉTODOS HTTP NO AUTORIZADOS

José Manuel
Redondo López

- En lugar de usar rewrite rules para limitar los métodos HTTP permitidos en un servidor o localización, usaremos la directiva `limit_except`
  - http://nginx.org/en/docs/http/ngx_http_core_module.html#limit_except
- Las directivas dentro de un bloque `limit_except` solo se aplican si la petición a una localización se hace con un método HTTP distinto a los indicados en la directiva
- El siguiente ejemplo prohíbe por ejemplo el uso de POST en la localización `/clientes`

```
location /clientes {
    limit_except GET HEAD {
        deny all;
    }
}
```

Evitar Hotlinking

José Manuel
Redondo López

- En lugar de usar **rewrite rules** para evitar **hotlinking**, como con **Apache 2**, vamos a usar la directiva `valid_referers` del módulo `ngx_http_referer_module`
  - http://nginx.org/en/docs/http/ngx_http_referer_module.html

```
valid_referers none blocked server_names *.midominio.com dominio.*
www.midominio.org/cuentas/ ~\.google\.;
if ($invalid_referer) {
    return 403;
}
```

- Los parámetros admitidos por esta directiva son
  - `none`: La petición HTTP no tiene la cabecera `Referer`
  - `blocked`: La petición HTTP tiene la cabecera `Referer`, pero la ha borrado por un **firewall** o proxy
  - `server_names`: La cabecera `Referer` tiene uno de los nombres asignados al servidor **Nginx**
  - **Cadena de caracteres**: Define un nombre de servidor y un prefijo de la URI opcional. Admite * al principio y al final. Si el `Referer` tiene un nº de puerto, se ignora
  - **Expresión regular**: Se identifica por empezar por "`~`"
    - La expresión regular se comprueba con el texto que aparezca detrás de "`http://`" o "`https://`"

Evitar Hotlinking

José Manuel
Redondo López

- Ejemplo que evita **hotlinking** de todas las imágenes de una web

```
location ~ .(gif|png|jpe?g)$ {
     valid_referers none blocked miweb.com *.miweb.com;
     if ($invalid_referer) {
        return   403;
    }
}
```

- Hemos incluido en `valid_referers` el nombre DNS de nuestra propia web
  - Y una serie de extensiones comunes en el contexto `location`
- Se puede hacer una redirección a una imagen estándar de respuesta, como hicimos en el caso de **Apache 2**
  - Esta variante protege las imágenes de un directorio concreto

```
location /pictures/ {
     valid_referers none blocked miweb.com *.miweb.com;
     if ($invalid_referer) {
        return   403;
    }
}
```

Otros módulos de seguridad y mejora de las peticiones

Configurando Nginx de forma aún más avanzada

Evitar el acceso desde la red ToR

José Manuel
Redondo López

- Esta operación que ya vimos con **Apache 2** se puede hacer gracias a las capacidades de inclusión de ficheros de **Nginx** en lugar de con **rewrite rules**
- Basta con denegar cada una de las IPs de los nodos de salida que podemos obtener de la localización anterior: https://www.dan.me.uk/torlist/?exit
  - Este fichero debe procesarse para añadir la directiva deny delante de cada IP y finalizar cada línea en `;`
    - Ej.: `deny 103.10.197.50;`
  - Hecho esto, basta con incluir el fichero para lograr el efecto deseado

```
server {
    include /etc/nginx/tor.lst;
    …
}
```

- Para mantenerla actualizada habría que programar un **script** que se encargue de descargar el fichero, procesarlo para añadir los `deny` adecuados y recargar **Nginx**
  - O usar un **script** ya hecho para este propósito: https://github.com/gongled/nginx-tor-sync
  - Esta aproximación se puede usar para bloquear listas de IPs arbitrarias

Bloqueo de User Agent Strings

José Manuel
Redondo López

- El bloqueo de User Agent Strings que se puedan considerar visitantes no deseados se puede realizar examinando el valor de la variable `$http_user_agent`
- El nº de posibles User Agent Strings es muy alto
  - Es aconsejable usar un procedimiento que permita especificar un gran nº de ellos con la mayor facilidad
- Para ello, crearemos un mapa en Nginx de la siguiente forma:
  - Creamos un fichero llamado `/etc/nginx/useragent.rules`
  - Escribimos lo que aparece en el cuadro adjunto

```
map $http_user_agent $badagent
{
        default          0;
        ~*malicious      1;
        ~*backdoor       1;
        ~*netcrawler     1;
        ~Antivirx        1;
        ~Arian           1;
        ~webbandit       1;
   …(otros user agents)
}
```

Bloqueo de User Agent Strings

José Manuel
Redondo López

- Al igual que lo visto anteriormente, los operadores '~*' y '~' usarán expresiones regulares no sensibles a mayúsculas (o no) para determinar si un agente está o no incluido en el mapa
  - Hecho esto, en nuestro contexto `http` debemos cargar el fichero (siempre antes que cualquier `server`)

```
http {
    .....
    include /etc/nginx/useragent.rules
}
```

- Y finalmente, dentro de una sección server lo location podemos usar el mapa para saber si el agente tiene o no permitida la navegación

```
server {
    ....
    if ($badagent) {
        return 403;
    }
    ....
}
```

- Otra opción es bloquear un nº muy grande de User Agent Strings considerados maliciosos o inadecuados usando el trabajo de este enlace
  - https://github.com/mariusv/nginx-badbot-blocker

Pagespeed

José Manuel
Redondo López

- **Pagespeed también está disponible para Nginx, con la misma funcionalidad vista**
- **Su instalación requiere la compilación de Nginx con él activo, lo cual puede hacerse siguiendo las instrucciones de este tutorial**
  - https://www.tecmint.com/install-ngx_pagespeed-to-optimize-nginx-performance-on-ubuntu/
  - Si hemos realizado una compilación de Nginx previa, conviene usar ese ejecutable como base para la instalación, añadiendo el módulo a sus fuentes
- **Una vez activo, podemos hacer lo mismo que en el caso de Apache 2**
  - https://modpagespeed.com/doc/config_filters
- **Desgraciadamente, el soporte de Nginx para Pagespeed compilado desde las fuentes es un proceso muy propenso a errores**
  - https://www.getpagespeed.com/server-setup/nginx/how-to-install-google-pagespeed-ngx_pagespeed-module-for-nginx-in-rhel-8-centos-8
  - Se recomienda buscar un contenedor con él instalado fiable (**"L-62 Princesa de Asturias"**)
- **Otra opción es instalar un proxy Apache 2 como frontend con este módulo activo**

WebDAV (*Web-based Distributed Authoring and Versioning*)

José Manuel
Redondo López

- **Nginx** tiene un módulo que nos permite usar **WebDAV**, `ngx_http_dav_module`
- Como indica el **CIS Benchmark**, conviene evitar instalarlo si no es estrictamente necesario
- **Más información**
  - http://nginx.org/en/docs/http/ngx_http_dav_module.html
  - https://opensource.ncsa.illinois.edu/confluence/display/ERGO/Creating+a+WebDAV+repository+server+with+NGINX

GEOLOCALIZACIÓN

José Manuel
Redondo López

- **Finalmente, también es posible geolocalizar las peticiones de los clientes con el módulo `ngx_http_geoip_module`**
  - **Más información:** http://nginx.org/en/docs/http/ngx_http_geoip_module.html
- **Y por supuesto, también podemos limitar el acceso en función de esta información**
  - **Más información**
    - https://www.howtoforge.com/nginx-how-to-block-visitors-by-country-with-the-geoip-module-debian-ubuntu
  - Podemos encontrar una lista de códigos de países en
    - http://dev.maxmind.com/geoip/legacy/codes/iso3166/

GEOLOCALIZACIÓN

José Manuel
Redondo López

- El módulo de geolocalización de **Nginx** depende una base de datos llamada **GeoIP** que se puede instalar de dos formas
- Como paquete: `apt-get install geoip-database libgeoip1`
- Esto creará la base de datos en `/usr/share/GeoIP/GeoIP.dat`
- Descargándosela de la web oficial y colocándola en el lugar adecuado:

```
cd /usr/share/GeoIP/
wget http://geolite.maxmind.com/download/geoip/database/GeoLiteCountry/GeoIP.dat.gz
gunzip GeoIP.dat.gz
```

- Con esto nos aseguramos de tener la base de datos más actualizada posible

GEOLOCALIZACIÓN

José Manuel
Redondo López

- Hecho esto, creamos un mapa como este dentro de un contexto `http`

```
geoip_country /usr/share/GeoIP/GeoIP.dat;
map $geoip_country_code $allowed_country {
    default yes;
    FK no;
    FM no;
}
```

- Esto permite el acceso a clientes de todos los países excepto los listados
- Lo contrario sería

```
geoip_country /usr/share/GeoIP/GeoIP.dat;
map $geoip_country_code $allowed_country {
    default no;
    FM yes;
    EH yes;
}
```

GEOLOCALIZACIÓN Y RESTRICCIÓN DE ACCESO POR PAÍSES

José Manuel
Redondo López

- También podemos escribir nuestra configuración en un fichero independiente y cargarlo con include como vimos anteriormente
- Hecho esto, vamos al contexto `server` o `location` a proteger y hacemos lo siguiente

```
if ($allowed_country = no) {
        return 444;
}
```

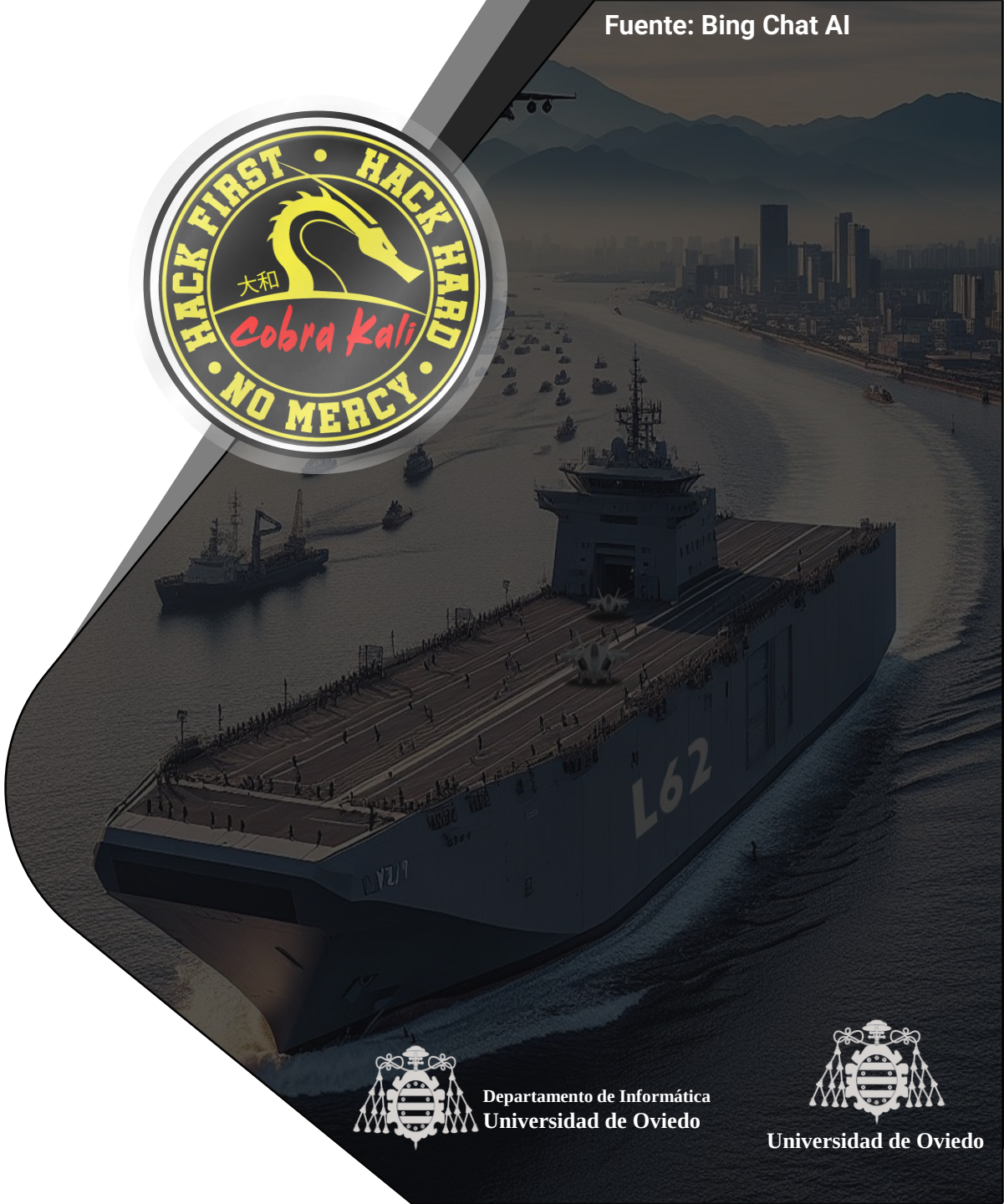
- El código 444 corta la conexión sin enviar respuesta a todos los países que no estén autorizados a acceder al contexto protegido
- El módulo **GeoIP no viene compilado con `nginx` en RHEL** y otros **Red Hat**
  - y requiere una recompilación del ejecutable después de instalar las bases de datos vistas

HTTP/3

José Manuel
Redondo López

- **HTTP/3 es la tercera versión del protocolo HTTP**
- **Está desarrollado sobre el protocolo de transporte QUIC de Google**
  - Este protocolo **usa por debajo el protocolo UDP**, en lugar del TCP anterior
  - Por ese motivo, **es más rápido**
    - Especialmente en redes que tienen alta probabilidad de perder paquetes y con mucha latencia
- **Otras características**
  - Mejora el tiempo de establecimiento de conexiones y el control de congestiones
  - Incorpora las **medidas de seguridad** de HTTP/2 y más
    - Siendo la más importante el **cifrado obligatorio de conexiones** con TLS 1.3
    - Mejora la seguridad contra ataques como la falsificación de conexiones
      - Al cifrar más partes de los datos enviados como parte del protocolo y su lógica de funcionamiento
- **Nginx soporta oficialmente HTTP/3 desde su versión 1.25.1**
  - Si no se disponible de esa versión, se puede compilar con él
    - https://codedamn.com/news/backend/leveraging-http3-with-nginx
  - **Configuración**: https://www.nginx.com/blog/binary-packages-for-preview-nginx-quic-http3-implementation/

FORTIFICANDO NGINX DE FORMA PROFESIONAL



Departamento de Informática
Universidad de Oviedo

Universidad de Oviedo