TELEMÁTICA
REVISTA DIGITAL de las Tecnologías de la Información y las Telecomunicaciones
Revista Telemática. Vol.20 No.1 , Enero-Marzo, 2021, p.12-26 ISSN 1729-3804

# CONFIGURACIONES INTERNAS PARA EL FORTALECIMIENTO DE LA SEGURIDAD EN NGINX

**Ing. Leonardo Aguilera Blanco[1], Ing. Leobel Rodríguez Chang[2], MSc. Henry Raúl González Brito[3]**

[1,2,3] Universidad de las Ciencias Informáticas, UCI, Carretera a San Antonio de los Baños, km 2 ½, Boyeros, La Habana, Cuba.
[1]laguilera@uci.cu, [2]lrchang@uci.cu, [3]henryraul@uci.cu

**RESUMEN**

La propia evolución de las tecnologías, en su desarrollo acelerado, ha propiciado el surgimiento de ataques cada vez más complejos a los sistemas de informacion. Los servidores web son la base sobre la que se sustentan diferentes procesos, además de interactuar directamente con el sistema operativo, razon por la cual se convierten en un objetivo atractivo para los ciberatacantes. El denominado servidor Nginx, de aplicación en la web, ha alcanzado una gran popularidad en los últimos años , lo que a su vez, lo ha convertido en un objetivo constante para la búsqueda de vulnerabilidades. Un análisis de servidores web populares de código abierto reveló importantes agujeros de seguridad, a pesar de los esfuerzos obvios de las comunidades de desarrolladores, generandose vulnerabilidades, no previstas, que impactan sobre sus usuarios cuando los mismos no son expertos en temas de seguridad. Para lograr una garantia adecuada en este campo es necesario aplicar medidas y configuraciones adicionales a las establecidas por defecto durante un despliegue inicial. Por ello el objetivo de esta investigación fue organizar y aplicar medidas para el fortalecimiento de la seguridad a través de los diferentes mecanismos de configuración para el servidor web Nginx. Estas medidas fueron introducidas en aplicaciones disponibles en Internet en el año 2018, siendo efectivas para garantizar la confidencialidad, integridad y disponibilidad de la información ante ataques, riesgos y amenazas de todo tipo, contribuyendo de este modo al proceso de informatización segura que se lleva a cabo en el país.

**PALABRAS CLAVES:** Despliegue Seguro, Fortalecimiento, Nginx, Seguridad Informática, Servidor web.

# INTERNAL CONFIGURATIONS FOR THE HARDENING SECURITY IN NGINX

**ABSTRACT**

The notorious evolution of technologies has led to the rising of increasingly complex cybernetic attacks. Web servers are the foundations supporting these technologies , interacting directly, besides, with the operating system which makes them an attractive target for cyber attackers. The denominated Nginx server has reached recently a great popularity, becoming thus a constant target for the search for vulnerabilities in such a way that analysis of popular open source web servers revealed important security holes, despite the obvious efforts of their developer communities.

These vulnerabilities leave both applications and their no expert users in a "security limbo" opened to exploitation, for what it is almost mandatory and necessary, to apply additional measures and configurations to those established by default during an initial deployment.

Therefore, the objective of this research was to organize and implement measures to harden security through the different configuration mechanisms for the Nginx web server. These measures were applied in applications available on the Internet in 2018, being effective to guarantee the confidentiality, integrity and availability of information in the face of attacks, risks and threats of all kinds, thus contributing to the secure computerization process that is taking place in the country.

**INDEX TERMS:** Hardening, Informatic security, Nginx, Secure Deployment, Web Server.

## 1. INTRODUCCIÓN

La incorporación de las nuevas tecnologías a la sociedad moderna ha provocado una mayor agilización en las actividades y procesos. Internet ha servido como una poderosa herramienta especialmente para el área de la

Manuscrito recibido: 2020-09-05, aceptado: 2020-11-22
Sitio web:http://revistatelematica.cujae.edu.cu/index.php/tele

TELEMÁTICA
REVISTA DIGITAL de las Tecnologías de la Información y las Telecomunicaciones
Revista Telemática. Vol.20 No.1 , Enero- Marzo, 2021, p.12-26 ISSN 1729-3804

comunicación y la información, posibilitando a las personas un acceso más fácil, menos restringido y con mayor alcance en el menor tiempo, de tal forma que es considerado como un medio global de comunicación cotidiana.
Casi todas estas necesidades son resueltas a través de la web que a su vez es sustentada por los servidores web. La mayor parte de la información está disponible a través de estas, por lo que se vuelven objetivos de cibercriminales que buscan sustraer, modificar o destruir la información de los usuarios o instituciones [1, 2].

Nginx es un servidor web de código abierto que acelera la entrega de contenido y aplicaciones, mejora la seguridad, facilita la disponibilidad y la escalabilidad para las aplicaciones web. Actualmente es el segundo servidor web más utilizado en el mundo y el 2do más utilizado en Cuba [3-5]. Como cualquier tecnología, Nginx posee vulnerabilidades (Fig.1.) que pueden ser aprovechadas por un ciberatacante para comprometer a la aplicación web y acceder a información sensible, permitiendo además afectar al sistema operativo, el sistema gestor de bases de datos y las aplicaciones web alojadas [6-8].

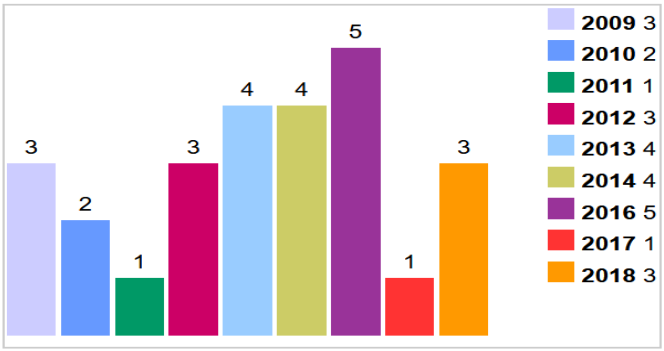


Figura 1: Cantidad de vulnerabilidades de Nginx por año. Fuente: https://www.cvedetails.com

Las vulnerabilidades del núcleo de Nginx son resueltas a través de nuevas versiones liberadas, las cuales a su vez contienen nuevas vulnerabilidades por descubrir y resolver [9]. Actualmente, se presentan dos escenarios en los cuales, las vulnerabilidades son la mayor amenaza de seguridad en los servidores web:

- **Base tecnológica desactualizada:** Constituye el escenario más común y está dado por la ausencia de prácticas de administración reconocidas, las cuales establecen que las aplicaciones y servicios deben ser actualizados cada vez que se libera una nueva versión pues representa la primera línea defensiva ante los intentos de explotación [10].

- **Vulnerabilidades descubiertas sin parches liberados:** Este es el peor escenario ya que los ciberdelincuentes conocen de la existencia de vulnerabilidades que pueden explotar y las aplicaciones web están desprotegidas al no contar con un método para resolver el problema de seguridad presente. Despublicarlo no es una opción en muchos casos debido a que afecta desde los procesos de organizaciones a los cuales tributa hasta el mismo posicionamiento alcanzado en motores de búsqueda [11].

La aplicación de buenas prácticas de vigilancia tecnológica, mediante la suscripción a los canales de avisos de seguridad de las tecnologías y complementos usados resuelve el primer escenario, sin embargo, el segundo es más difícil de enfrentar en la medida en que es imposible determinar cuál será la vulnerabilidad y el vector de ataque que será empleado. Por este motivo, es importante que el servidor web Nginx incorpore medidas de seguridad proactivas para limitar al máximo la explotación de las vulnerabilidades que puedan estar presentes y en el caso de que esto ocurra, impedir que los ciberdelincuentes puedan escalar privilegios que les permitan afectar al servidor web en su conjunto. En la medida en que se interpongan líneas de defensas entre el servidor web y los ciberatacantes, menores serán las posibilidades de que estos tengan éxito.

Nginx, es un servidor web que requiere dinamismo y flexibilidad con las configuraciones de sus usuarios, no debe contener por defecto, medidas de seguridad restrictivas que acoten su mercado de usuarios. No obstante, esto no impide que se puedan aplicar medidas personalizadas que posibiliten incrementar la protección de una instalación de Nginx y prepararla para los constantes ciberataques basados en botnets [12] que se producen en Internet [13].

Las medidas de seguridad a nivel de permisos en el sistema de archivos y usuarios del sistema operativo, configuraciones de reglas de acceso del servidor web, aplicaciones de firewall y sistemas de detección de intrusos son ampliamente conocidas. A pesar de ello, las estadísticas muestran como existe un crecimiento en los ciberataques contra aplicaciones web y esto está dado porque la seguridad debe concebirse como un proceso y no un producto y deben aplicarse medidas a diferentes niveles de software y hardware, más allá de la existencia de soluciones que prometen resolver todos estos problemas. Archivos de Nginx como nginx.conf, default y buffer.conf pueden ser configurados de manera tal que puede incrementarse la seguridad de forma sustancial, creándose de este modo un fortalecimiento de la seguridad del núcleo de Nginx que se adiciona a las medidas ya aplicadas mediante componentes y aplicaciones externas.

En correspondencia con lo anterior, el objetivo de la investigación fue organizar y aplicar medidas para el fortalecimiento de la seguridad a bajo nivel y basadas en la modificación de los archivos nginx.conf, default y buffer.conf en instalaciones de Nginx. Su empleo en varias aplicaciones web publicadas en Internet a lo largo del año 2018, permite afirmar que son efectivas para garantizar una seguridad adecuada y proactiva antes los ataques, riesgos y amenazas de todo tipo, contribuyendo de este modo al proceso de informatización segura que se lleva a cabo en el país.

## 2. FORMALIZACIÓN DE LOS CONTROLES PARA EL FORTALECIMIENTO DE LA SEGURIDAD EN NGINX

En la investigación se presentan un conjunto de controles para el fortalecimiento de la seguridad en el servidor web Nginx (Fig. 2.). Estos controles se basan en configuraciones internas que se aplican en tres dimensiones: la seguridad en la gestión de las peticiones HTTP recibidas, la seguridad en la gestión de las respuestas HTTP enviadas y las configuraciones bases de auditoría e interacción con el sistema operativo. Para ello se proponen 12 controles que serán explicados en las siguientes secciones. En esta investigación la versión de Nginx empleada fue la 1.12, instalada sobre Ubuntu Server 18.04 y se utilizó la versión de PHP 7.2.

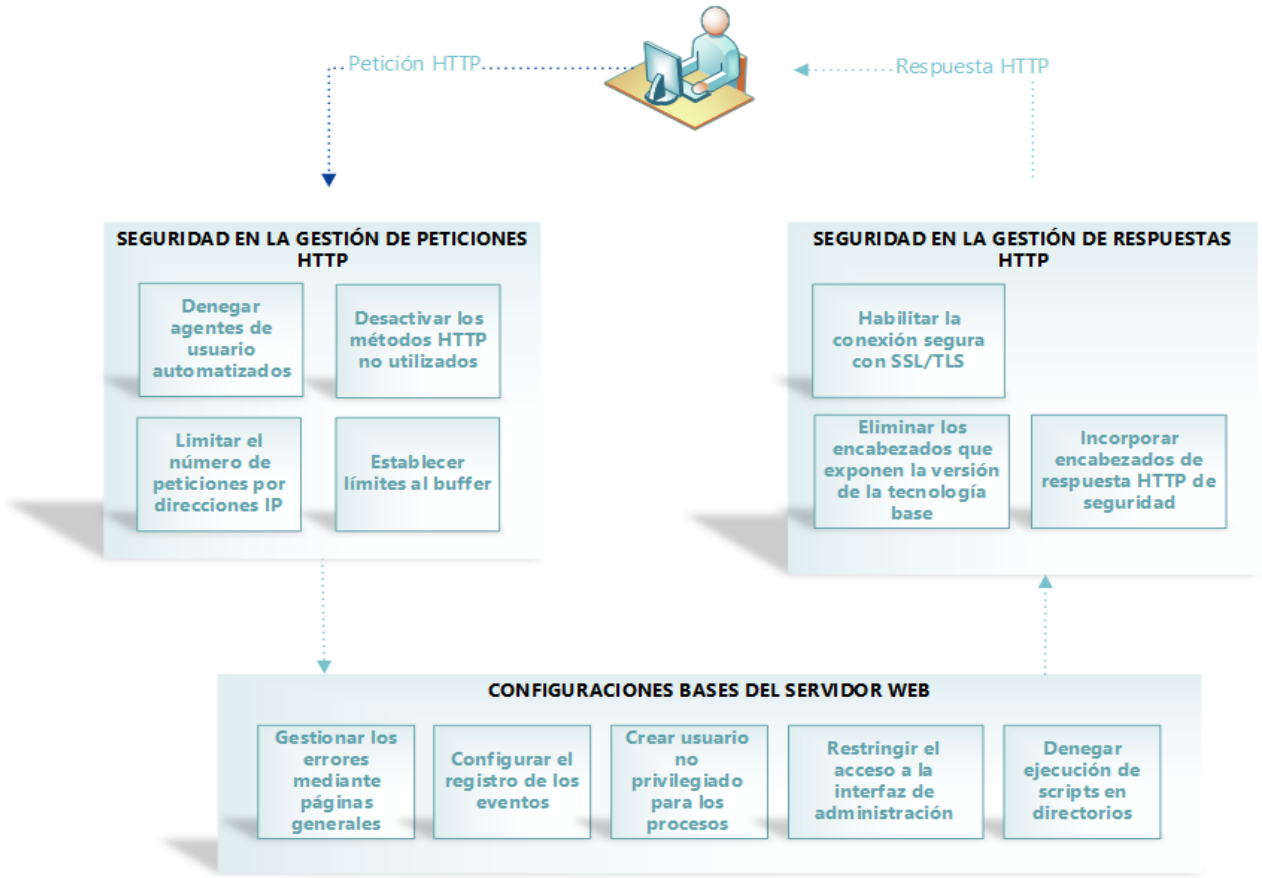


Figura 2: Relación entre los diferentes controles propuestos. Elaboración propia.

TELEMÁTICA
REVISTA DIGITAL de las Tecnologías de la Información y las Telecomunicaciones
Revista Telemática. Vol.20 No.1 , Enero- Marzo, 2021, p.12-26 ISSN 1729-3804

**Configuraciones bases del servidor web**

El servidor web requiere un conjunto de configuraciones que establezcan límites en la información y permisos que brinda. Se debe impedir que existan configuraciones predefinidas y conocidas por los ciberatacantes. Tampoco se deben utilizar usuarios y rutas por defecto, dejando brechas de seguridad que podrían expandir las posibilidades a una vulnerabilidad. Los controles que se mencionan a continuación constituyen medidas para el aseguramiento del entorno del servidor.

**Gestionar los errores mediante páginas generales**

Por defecto, el servidor Nginx al cargar páginas de error como 403 (acceso prohibido) o 404 (no encontrado) muestra información que puede ser relevante para un atacante. Por ejemplo, en el caso de los errores 403, aunque el ciberatacante no logre entrar sabe que el archivo o directorio existe, por lo que posibilitaría un reconocimiento de la aplicación web [15]. Lo recomendable para los errores es crear una página general para los errores.
Para cambiar estas páginas de respuestas se modifica el archivo default que se encuentra en la ruta /etc/nginx/sites-enabled/ y en el bloque server se adiciona lo siguiente:

```
server {
    ...
    error_page 401 403 404 /error.html;
    ...
}
```

Dicha configuración se puede utilizar para cualquier respuesta del servidor de los códigos 4XX y 5XX, que constituyen los códigos de respuesta para los errores del cliente y el servidor [16].
El archivo error.html debe estar ubicado en /var/www/html/ o en un subdirectorio dentro de esta ruta. Ejemplo de página web genérica de error:

```
<html>
    <title>Página de error</title>
    <h1>Error</h1>
    <p>Lo sentimos mucho, ha ocurrido un error, disculpe las molestias.</p>
</html>
```

**Configurar el registro de los eventos**

Se deben configurar los registros de eventos para que se registren todas las acciones realizadas en el servidor, esto ayudará a identificar fallos e información relevante sobre los ataques [21]. Para configurar el registro de eventos se tienen que tener en cuenta las siguientes directivas:

- La directiva **error_log** registra los problemas al iniciar el servidor o durante su operación. Por defecto la ruta donde se guardan los registros de errores es /var/log/nginx/error.log
- La directiva **access_log** permite configurar el servidor para que registre la información del registro de acceso. Por defecto la ruta donde se guardan los registros de acceso es /var/log/nginx/access.log. Se recomienda modificar la ruta y los nombres de los archivos de los registros.

Para guardar los registros del ejemplo se debe crear un nuevo directorio llamado sitioweb.

```
mkdir -p /var/log/nginx/sitioweb
```

Para cambiar la ubicación por defecto se modifica el archivo /etc/nginx/nginx.conf de la siguiente forma:

```
access_log /var/log/nginx/sitioweb/access.log;
error_log /var/log/nginx/sitioweb/error.log;
```

**Crear usuario no privilegiado para los procesos**

Por defecto el servidor Nginx utiliza el usuario www-data y grupo www-data. Por razones de seguridad es recomendado crear una cuenta no privilegiada para los procesos de Nginx, por ejemplo: http-web. Para realizar esto a continuación, creamos el grupo y usuario http-web de la siguiente forma:

```
groupadd http-web
useradd -d /var/www/ -g http-web -s /bin/nologin http-web
```

Luego necesitamos indicarle al servidor Nginx que utilice el usuario anteriormente creado, para ello modificamos el archivo /etc/nginx/nginx.conf en user sustituyendo www-data, usuario por defecto, por nuestro nuevo usuario http-web, quedando de la siguiente forma:

```
user http-web;
```

**Restringir el acceso a la interfaz de administración**

Debe restringirse el acceso al panel de administración y solo debe permitirse la conexión desde los puestos de trabajo del personal. En caso de que Nginx necesite procesar ficheros PHP se debe incluir la siguiente configuración en el archivo /etc/nginx/sites-enabled/default:

```
location ~ \.php$ {
   include snippets/fastcgi-php.conf;
   fastcgi_pass unix:/var/run/php/php7.2-fpm.sock;
}
```

El bloque location permite especificar directivas en esa URI. Ejemplos:

- **Restringir el acceso por IP:** Para restringir el acceso por IP se hace uso de las directivas allow y deny. La directiva allow permite aceptar las direcciones IP o las subredes que se especifiquen; y la directiva deny denegar por IP, por subred, o a todos. Estas directivas son interpretadas en el orden que aparezcan. Ejemplos:

```
allow 192.168.56.101;
allow 192.168.33.0/24;
deny all;
```

- **Cambiar los códigos de respuesta:** Esto se realiza a través de la directiva error page código respuesta URI en donde se especifica una URI que se mostrará para los códigos de respuesta especificados. Ejemplos:

```
error_page 404 /404.html;
error_page 500 502 503 /50x.html;
error_page 403 http://example.com;
```

Especificar otros bloques location: Esto se utiliza para establecer reglas específicas para un enlace o expresiones regulares de un enlace. Por ejemplo el (location ~ \.php) que se utiliza para que sean procesados los ficheros PHP. Debido a que los bloques location cuando utilizan las expresiones regulares no siguen las reglas de otros bloques location, por lo que si se necesita procesar los ficheros PHP se debe poner este bloque dentro de los bloques location con expresiones regulares adicionalmente del que existe al nivel de la raíz. Ejemplo de location:

```
location ~ /URI {
   allow 127.0.0.1;
   deny all;
   location ~ \.php$ {
      include snippets/fastcgi-php.conf;
      fastcgi_pass unix:/var/run/php/php7.2-fpm.sock;
   }
}
```

El bloque location utiliza expresiones regulares para especificar las URI. Estos son:

- En blanco si no hay modificadores presentes, el URI solicitado se comparará con el URI del bloque.
- = Esta expresión se usa para hacer coincidir exactamente el URI del bloque con el solicitado.

TELEMÁTICA
REVISTA DIGITAL de las Tecnologías de la Información y las Telecomunicaciones
Revista Telemática. Vol.20 No.1 , Enero- Marzo, 2021, p.12-26 ISSN 1729-3804

- ~ El signo de tilde se usa para la coincidencia de expresión regular que distingue entre mayúsculas y minúsculas con un URI solicitado.
- ~ * La tilde seguida de un signo de asterisco se usa para la coincidencia de expresiones regulares que no distingue entre mayúsculas y minúsculas con el URI solicitado.
- ^ ~ Esta expresión se utiliza para realizar la mayor coincidencia de expresión no regular contra el URI solicitado. Si el URI solicitado golpea tal bloque de ubicación, no se realizará ninguna otra coincidencia.

A continuación, se describe un ejemplo de cómo restringir el acceso a la interfaz de administración de WordPress:
Para restringir los accesos se debe modificar el archivo /etc/nginx/sites-enabled/default y poner la siguiente configuración:

1) Bloqueando acceso al panel de administración

```
location ^~ /wordpress/wp-admin/{
    error_page 403 https://192.168.56.101/wordpress;
    allow 127.0.0.1;
    deny all;
    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php7.2-fpm.sock;
    }
}
```

2) Bloqueando acceso al panel de autenticación

```
location ^~ /wordpress/wp-login.php{
    error_page 403 https://192.168.56.101/wordpress;
    allow 127.0.0.1;
    deny all;
    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php7.2-fpm.sock;
    }
}
```

La directiva **allow** permite el acceso HTTP hacia el servidor web desde la dirección IP 127.0.0.1, en caso de conectarse desde otra dirección IP se redireccionaría a la dirección https://192.168.56.101/wordpress debido a que el resto de direcciones IP tienen el acceso prohibido debido a la directiva **deny all**. Además, con la directiva **error_page** modificamos la página de respuesta al producirse un código 403 (Prohibido) hacia dicha dirección.

**Denegar ejecución de scripts en directorios**

La ejecución de script remota, es uno de los ataques más comunes que se realizan, constituye en la subida de un fichero hacia el servidor web. Luego es posible acceder a él a través de la URL ejecutando acciones remotas en el servidor web. A través de estos es posible crear usuarios, interactuar con los procesos del sistema, crear puertas traseras, acceder a otros sitios webs, entre otras acciones [29].

Para denegar la ejecución de scripts en los directorios se debe modificar el archivo /etc/nginx/sites-enabled/default adicionando el siguiente código:

```
location ~* /(images|cache|media|logs|tmp)/.*.(php|pl|py|jsp|asp|sh|cgi)$ {
    return 403;
}
```

En el código anterior se bloquea el acceso a cualquier archivo que tenga extensión php, html, cgi, perl, python, java, bash o exe a través del navegador para aquellas URI que contengan las palabras images, cache, media, logs y tmp. Las extensiones y los directorios pueden ser sustituidos o adicionar aquellas direcciones en la que los usuarios puedan subir ficheros.

**Seguridad en la gestión de respuestas HTTP**

El siguiente grupo de controles están orientados al fortalecimiento de la respuesta HTTP devuelta por el servidor web a través de solicitudes y previenen la fuga de información del servidor, como la versión de Nginx empleada. Se explica como realizar un cifrado de la comunicación a través de HTTPS, creando un certificado digital y redireccionando el tráfico HTTP hacia HTTPS. También se incluyen encabezados que previenen ataques como XSS (Cross Site Scripting), Clickjacking, Hijacking, etc.

**Eliminar los encabezados que exponen la versión de la tecnología base**

En su configuración inicial Nginx muestra la versión del servidor, el sistema operativo, la dirección IP y el puerto por el que escucha en páginas de error y encabezados HTTP. Esto constituye un riesgo de seguridad debido a que le facilita al atacante el reconocimiento de la estructura, así como determinar el rango de vulnerabilidades y herramientas que debe utilizar [14].

Para evitar esta fuga de información se modifica en la ruta /etc/nginx/ el archivo nginx.conf, y dentro del bloque http, agregue la siguiente línea:

```
http {
    ...
    server_tokens off;
    ...
}
```

La directiva **server tokens** identifica la información que Nginx envía sobre sí mismo a los clientes en las peticiones.

**Habilitar la conexión segura con SSL/TLS**

En la actualidad, es un requisito obligatorio proteger el canal de transmisión entre el agente de usuario y el servidor web, para ello se emplean protocolos de encriptación SSL/TLS combinados con HTTP (HTTPS) [22]. Para habilitar la conexión segura con SSL/TLS es necesario seguir los siguientes pasos:

1) **Crear certificado digital**

Para crear el certificado es necesario crear el directorio /etc/nginx/ssl/ para guardar los archivos de configuración de la siguiente forma:

```
mkdir -p /etc/nginx/ssl
```

A continuación, se establecerá el certificado y la llave que serán empleados por el servidor web:

```
sudo openssl req -x509 -nodes -sha256 -days 365 -newkey rsa:4096 -keyout /etc/nginx/ssl/server.key -out /etc/nginx/ssl/server.crt
```

El certificado y la llave se crean en la ruta /etc/nginx/ssl/server.key y /etc/nginx/ssl/server.crt para su uso. En caso de que ya se posea un certificado y una llave se debe omitir el paso anterior y guardar los archivos en el directorio /etc/nginx/ssl/ creado o modificar la ruta hacia estos en los siguientes pasos. Posteriormente se fortalece los parámetros de cifrado del método Diffie-Hellman [23], debido a que este utiliza por defecto valores de 1024 bits que pueden ser vulnerados, se recomienda utilizar como valor 4096 [24]:
sudo openssl dhparam -out /etc/nginx/ssl/dhparam.pem 4096
Este comando hará algunas preguntas simples sobre los detalles del sitio o empresa. Después de eso, creará una clave cifrada RSA de 2048 bits en el archivo /etc/nginx/ssl/server.key y un certificado
SHA256 en el archivo /etc/nginx/ssl/server.crt.

2) **Activar configuración SSL/TLS**

Se deben establecer directivas para activar la configuración SSL/TLS. Para ello en el archivo /etc/nginx/sites-enabled/default se realizan los siguientes pasos.
Primero se especifica el puerto para el servicio SSL/TLS dentro del bloque server.

TELEMÁTICA
REVISTA DIGITAL de las Tecnologías de la Información y las Telecomunicaciones
Revista Telemática. Vol.20 No.1 , Enero- Marzo, 2021, p.12-26 ISSN 1729-3804

```
server {
    ...
    listen 443 ssl;
    listen [::]:443 ssl;
    ...
}
```

Después se debe añadir el siguiente código en el archivo /etc/nginx/sites-enabled/default dentro del bloque server luego del código anterior:

```
server {
    ...
    ssl_dhparam /etc/nginx/ssl/dhparam.pem;
    ssl_certificate /etc/nginx/ssl/nginx.crt;
    ssl_certificate_key /etc/nginx/ssl/nginx.key;
    ...
}
```

A continuación se describen las directivas y atributos utilizados en el código anterior:

- La directiva **ssl dhparam:** Especifica el archivo con los parámetros de cifrados que utilizan el protocolo Diffie-Hellman.
- La directiva **ssl certificate:** Permite la utilización de certificados. Debe actualizarse si se emplean otros certificados.
- La directiva **ssl certificate key:** Especifica la llave privada SSL/TLS que se ha generado previamente.

Se especifica la versión de SSL/TLS que se va a emplear a través de la directiva ssl protocol:

```
server {
    ...
    ssl_protocols TLSv1.2;
    ...
}
```

Luego a través de la directiva ssl ciphers se especifica el algoritmo de cifrado a utilizar y mediante la directiva ssl prefer server ciphers se verifica que el agente de usuario cumpla con la configuración de cifrado del servidor web.

```
server {
    ...
    ssl_prefer_server_ciphers on;
    ssl_ciphers 'EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH';
    ...
}
```

Después de realizadas las modificaciones el archivo de configuración HTTPS /etc/nginx/sites-enabled/default queda de la siguiente forma:

```
server {
    ...
    listen 443 ssl;
    listen [::]:443 ssl;
    ssl_dhparam /etc/nginx/ssl/dhparam.pem;
    ssl_certificate /etc/nginx/ssl/nginx.crt;
    ssl_certificate_key /etc/nginx/ssl/nginx.key;
    ssl_protocols TLSv1.2;
    ssl_prefer_server_ciphers on;
    ssl_ciphers 'EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH';
```

```
    ...
}
```

### 3) Redireccionar conexiones HTTP hacia HTTPS

Para forzar el uso de HTTPS se debe modificar el archivo /etc/nginx/sites-enabled/default en el que se establecen los puertos de escucha para HTTP y HTTPS en el mismo bloque server, sin embargo, para realizar el redireccionamiento es necesario separar estos servicios en diferentes bloques, por lo que se debe crear un bloque server para la configuración HTTP y mantener la existente solo para peticiones HTTPS. Luego de separadas las configuraciones para las peticiones HTTP y HTTPS, se debe redirigir las peticiones HTTP hacia HTTPS en el bloque server de la configuración HTTP a través de la regla siguiente:

```
rewrite ^ https://$host$request_uri? permanent;
```

Al agregar la regla anterior, el archivo /etc/nginx/sites-enabled/default queda de la siguiente forma:

```
server {
    listen 80;
    listen [::]:80;
    server_name 192.168.56.101;
    rewrite ^ https://$host$request_uri? permanent;
}
server {
    listen 443 ssl;
    listen [::]:443 ssl;
    ssl_dhparam /etc/nginx/ssl/dhparam.pem;
    ssl_certificate /etc/nginx/ssl/nginx.crt;
    ssl_certificate_key /etc/nginx/ssl/nginx.key;
    ssl_protocols TLSv1.2;
    ssl_prefer_server_ciphers on;
    ssl_ciphers 'EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH';
    server_name 192.168.56.101;
    root /var/www/html;
    index index.php index.html index.htm index.nginx-debian.html;
    location / {
        try_files $uri $uri/ =404;
    }
    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php7.2-fpm.sock;
    }
}
```

**Incorporar encabezados de respuesta HTTP de seguridad**

El protocolo HTTP tiene campos de encabezados especializados en mejorar la seguridad de las transacciones HTTP. A pesar de ello, es difícil encontrar servidores y aplicaciones web que incorporen por defecto las configuraciones necesarias para enviarlos. Los campos de encabezados recomendados son:

- **Strict-Transport-Security:** Instruye al agente de usuario para que la conexión se realice a través de HTTPS en lugar de HTTP. Es común encontrar aplicaciones web que permiten el uso de ambas conexiones, dejando en manos del usuario la responsabilidad de indicar la forma de hacerlo [25].
- **X-Frame-Options:** Especifica que no se puede embeber la aplicación web en una etiqueta HTML frame. Esto garantiza en gran medida la protección ante ataques de secuestros de clic, más conocidos por su nombre en inglés de clickjacking [26], los cuales se utilizan mayormente para el robo de credenciales de usuarios.

TELEMÁTICA
REVISTA DIGITAL de las Tecnologías de la Información y las Telecomunicaciones
Revista Telemática. Vol.20 No.1 , Enero- Marzo, 2021, p.12-26 ISSN 1729-3804

- **X-Xss-Protection:** Contribuye a evitar los ataques de tipo XSS [27] mediante la activación del filtro relacionado que poseen los navegadores web modernos.

- **X-Content-Type-Options:** Instruye al navegador web que no cargue las hojas de estilo ni los scripts dañinos basados en la confusión de tipos MIME [28].

Para incorporar el encabezado de Strict-Transport-Security es necesario tener configurado en el servidor Nginx el SSL/TLS, debido a que fuerza la conexión por HTTPS y no aceptará otra conexión devolviendo un código de respuesta de 400 (Bad Request).
Adicionar en el archivo /etc/nginx/sites-enabled/default dentro del bloque server las siguientes directivas:

```
server{
   ...
   add_header X-Frame-Options SAMEORIGEN;
   add_header Strict-Transport-Security max-age=2592000;
   add_header X-XSS-Protection ’1; mode=block’;
   add_header X-Content-Type-Options nosniff;
   ...
}
```

**Seguridad en la gestión de peticiones HTTP**

Se deben establecer medidas para evitar que se utilicen métodos HTTP que podrían modificar contenido sin autorización o producir peticiones ilimitadas afectando la disponibilidad y el comportamiento del servidor web. A continuación se detallan controles para prevenir el uso malintencionado de las peticiones HTTP.

**1) Desactivar los métodos HTTP no utilizados**

HTTP define un conjunto de métodos de petición para indicar la acción que se desea realizar para un recurso determinado [17]. Debido a riesgos de seguridad y uso malintencionado de estos métodos [18] es recomendable especificar en el servidor que métodos HTTP se van a utilizar y denegar los otros. Para especificar los métodos permitidos se modifica el archivo de configuración /etc/nginx/sites-enabled/default adicionando dentro de los bloques location el siguiente fragmento de código para aquellos URI que se deseen desactivar los métodos HTTP no utilizados:

```
location /URI {
   ...
   limit_except GET POST {
      deny all;
   }
   ...
}
```

En el bloque **limit_except** se especifican separados por un espacio los métodos HTTP que se desean permitir.

**Establecer límites al buffer**

Se deben establecer límites a la cantidad de conexiones simultáneas y ejercer un uso eficiente de los recursos del servidor para evitar ataques de desbordamiento de Buffer (buffer overflow) [19] y/o denegación de servicio (denial of service) [20]. Nginx permite establecer directrices para modificar como se emplean estos recursos, para ello se debe adicionar un nuevo archivo en la ruta /etc/nginx/conf.d/ con el nombre buffer.conf y utilizar las directrices siguientes, que se adicionan teniendo en cuenta las necesidades y recursos del servidor web:

- **client body buffer size:** Esta directiva permite especificar el tamaño del cuerpo de la solicitud del cliente. El valor predeterminado es 8k o 16k.

```
...
client_body_buffer_size 1k;
...
```

- **client header buffer size:** Esta directiva permite establecer el tamaño del encabezado para el encabezado de la solicitud del cliente. Para la mayoría de las solicitudes, un tamaño de buffer de 1 K es suficiente. Aumente esto si tiene un encabezado personalizado o una cookie con gran tamaño enviada desde el cliente.

  ...
  *client_header_buffer_size 1k;*
  ...
- **client max body size:** Esta directiva permite asignar el tamaño de cuerpo máximo aceptado de la solicitud del cliente, indicado por la línea Content-Length en el encabezado de la solicitud. Si el tamaño es mayor que el dado, el cliente recibe el error "Solicitud de entidad demasiado grande" (413). Aumente esto cuando esté recibiendo cargas de archivos a través del método POST.

  ...
  *client_max_body_size 1k;*
  ...
- **large client header buffers:** Esta directiva permite asignar el número y el tamaño de los buffers utilizado para leer los encabezados de gran tamaño de solicitudes del cliente. El tamaño por defecto es 8K.

  ...
  *large_client_header_buffers 2 1k;*
  ...
- **client body timeout:** La directiva define un tiempo de espera para leer el cuerpo de la solicitud del cliente. Si después de este tiempo el cliente no realiza ninguna petición, Nginx envía el código de respuesta "Tiempo de espera de solicitud" (408). El valor predeterminado es 60.

  ...
  *client_body_timeout 10;*
  ...
- **client header timeout:** La directiva asigna un tiempo de espera para leer el encabezado de la solicitud del cliente. Si después de este tiempo el cliente no realiza ninguna petición, Nginx envía el código de respuesta "Tiempo de espera de solicitud"(408).

  ...
  *client_header_timeout 10;*
  ...
- **keepalive timeout:** El primer parámetro asigna el tiempo de espera para las conexiones de keepalive con el cliente. El servidor cerrará las conexiones después de este tiempo. El segundo parámetro opcional asigna el valor de tiempo en el encabezado KeepAlive: timeout = time de la respuesta.

  ...
  *keepalive_timeout 5 5;*
  ...
- **send timeout:** Esta directiva asigna el tiempo de espera de respuesta al cliente. El tiempo de espera se establece entre dos operaciones de escritura, si después de este tiempo el cliente no recibe ninguna respuesta, entonces el servidor Nginx está cerrando la conexión.

  ...
  *send_timeout 10;*
  ...

**Denegar agentes de usuario automatizados**

Todas las herramientas utilizan agentes de usuarios para interactuar con el servidor web, y las automatizadas no son la excepción, denegar los agentes automatizados más comunes que se utilizan para realizar fingerprint o ataques hacia el servidor web puede mitigar estas acciones. A pesar de que esta cadena de caracteres puede ser modificada de su valor por defecto, un alto número de usuarios no lo realiza, por lo que para usuarios más avanzados no constituiría una fuerte medida de seguridad sin embargo sirve para forzar al atacante a realizar más acciones y entorpecer más el proceso de detección y ataque hacia el servidor web [32].

Para denegar que agentes de usuario automatizados realicen peticiones a la aplicación web se debe utilizar la siguiente configuración en el archivo /etc/nginx/sites-enabled/default:

```
if ($http_user_agent ~* (acunetix|curl) ) {
    return 403;
}
```

Si se desea denegar más agentes de usuarios se deben agregar separados por "|".

TELEMÁTICA
REVISTA DIGITAL de las Tecnologías de la Información y las Telecomunicaciones
Revista Telemática. Vol.20 No.1 , Enero- Marzo, 2021, p.12-26 ISSN 1729-3804

Algunos ejemplos de agentes de usuarios son: sqlmap, nikto, metasploit, hping3, maltego, nessus, webscarab, sqlsus, sqlninja, aranchni, netsparker, nmap, dirbuster, zenmap, hydra, owaspzap, w3af, vega, msnbot, Purebot, Baiduspider, Lipperhey, Mail.Ru, scrapbot, burpsuite, aircrack-ng, whatweb, medusa.

**Limitar el número de peticiones por direcciones IP**

Es recomendado reducir la cantidad de memoria que se consume por las peticiones y reducir la frecuencia de estas, ya que esta constituye la forma más común de realizar ataques de denegación de servicios [33]. Por lo que se le debe limitar el número de peticiones permitidas por una misma dirección.

Para ello es necesario crear una zona modificando el archivo /etc/nginx/nginx.conf agregando la siguiente configuración:

```
http{
   ...
   limit_req_zone $binary_remote_addr zone=one:10m rate=10r/s;
   ...
}
```

La directiva **limit_req_zone** define los siguientes parámetros para la limitación de la velocidad de las solicitudes y se define en el bloque **http**:

- **Key:** Define la característica de solicitud contra la cual se aplica el límite. En el ejemplo, la variable $ binary remote addr, contiene una representación binaria de la dirección IP de un cliente. Esto significa que estamos limitando cada dirección IP única a la tasa de solicitud definida por el tercer parámetro.

- **Zone:** Define la zona de memoria compartida utilizada para almacenar el estado de cada dirección IP y la frecuencia con la que ha accedido a una URL limitada a la solicitud. Mantener la información en la memoria compartida significa que se puede compartir entre los procesos de trabajo de Nginx. La definición tiene dos partes: el nombre de la zona identificado por zone = keyword y el tamaño después de los dos puntos. La información del estado para aproximadamente 16,000 direcciones IP toma 1 megabyte, por lo que en la zona ejemplo establecida a 10 megabytes puede almacenar aproximadamente 160,000 direcciones. Si el almacenamiento se agota cuando Nginx necesita agregar una nueva entrada, elimina la entrada más antigua. Si el espacio liberado aún no es suficiente para acomodar el nuevo registro, Nginx devuelve el código de estado 503 (Servicio no disponible temporalmente). Además, para evitar que la memoria se agote, cada vez que Nginx crea una nueva entrada, elimina hasta dos entradas que no se han utilizado en los 60 segundos anteriores.

- **Rate:** Establece la tasa de solicitud máxima. En el ejemplo, la tasa no puede exceder las 10 solicitudes por segundo. Nginx rastrea las solicitudes en milisegundos, por lo que este límite corresponde a 1 solicitud cada 100 milisegundos. La directiva **limit_req_zone** establece los parámetros para la limitación de la tasa y la zona de memoria compartida, pero en realidad no limita la tasa de solicitud. Para eso necesita aplicar el límite a una ubicación específica o bloque de servidor incluyendo una directiva limit_req allí.

Por lo que luego de definida la zona es aplicada a través de la directiva limit_req en el archivo /etc/nginx/sites-enabled/default dentro del bloque location especificando la URI a la que se le desee aplicar la zona:

```
server{
   ...
   location /URI {
      ...
      limit_req zone=one burst=5 nodelay;
      limit_req_status 403;
      ...
   }
   ...
}
```

A continuación se describen las directivas y atributos utilizados en el código anterior:

- La directiva **limit req** permite establecer una zona declarada anteriormente en un contexto determinado.
- El atributo burst define la cantidad de solicitudes que un cliente puede hacer por encima de la especificada por la zona, en el caso del ejemplo la frecuencia es de 10 solicitudes por segundo, la solicitud que es recibida después se coloca en cola, en este caso el tamaño máximo de la cola es 5. Si la cantidad de peticiones supera la cola entonces Nginx devuelve el código de estado 503 (Servicio no disponible temporalmente).
- El atributo **nodelay** permite que cuando es liberado un espacio de la zona se procesa la primera petición en cola, en caso contrario no se procesan las peticiones de la cola hasta que no sean atendidos todos los elementos de la zona.
- La directiva **limit_req_status** permite establecer el código de respuesta obtenido luego de haber superado el límite de peticiones de la zona y en cola.

## 3. CONCLUSIONES

Se presentaron una serie de medidas para incrementar la seguridad a bajo nivel en instalaciones de Nginx. La gestión de estas debe ser parte indispensable y cotidiana de los administradores de este tipo de servidor web.

Con la propuesta de fortificación los administradores de servidores de Nginx pueden:

- Eludir los ataques automatizados más comunes.
- Disminuir la superficie de ataque.
- Minimizar los ataques de fuerza bruta contra paneles de autenticación y administración.
- Aumentar la seguridad de las transacciones HTTP.
- Impedir el reconocimiento de la estructura del portal.
- Mitigar ataques de desbordamiento de buffer y de denegación de servicios.
- Mantener identificado al usuario y a las acciones que se realizan en nuestro servidor web.

## RECOMENDACIONES

Como trabajo futuro se propone el desarrollo de una aplicación que permita evaluar la configuración de una instalación de Nginx y emita un reporte con las principales recomendaciones de fortalecimiento de la seguridad para su posterior aplicación automática.

## RECONOCIMIENTOS

Esta investigación ha sido llevada a cabo en el marco del proyecto “Metodología Ágil para pruebas de penetración en aplicaciones web (MAPPAW)” el cual formar parte del Programa de Prioridad Nacional de Ciencia, Tecnología e Innovación “Informatización de la Sociedad”.

## REFERENCIAS

[1] M. Meeker and L. Wu, "Internet trends 2018," Kleiner Perkins, 2018.

[2] G. Tsakalidis, K. Vergidis, M. Madas, and M. Vlachopoulou, "Cybersecurity threats: a proposed system for assessing threat severity," in *Proceedings of the the forth international conference on decision support system technology-ICDSST 2018*, Heraklion, Greece, 22-25 May 2018.

[3] R. Soni, *Nginx: From Beginner to Pro*. Berkeley, CA: Apress, 2016, p. 240.

[4] BuiltWith®. "Web Server Usage Distribution in Cuba." https://trends.builtwith.com/web-server/country/Cuba (accessed 3 sept, 2020).

[5] BuiltWith®. "Web Server Usage Distribution in the Top 1 Million Sites." https://trends.builtwith.com/web-server (accessed 3 sept, 2020).

[6] C. Josiline Phiri, "Adoption of Open Source Software in Libraries in Developing Countries," *International Journal of Library and Information Services (IJLIS),* vol. 7, no. 1, pp. 15-29, 2018, doi: 10.4018/IJLIS.2018010102.

[7] P. Kamat and A. Singh Gautam, "Recent trends in the era of cybercrime and the measures to control them," in *Handbook of e-business security*, J. M. R. S. Tavares Ed. Boca Raton, FL: CRC Press, 2018, pp. 243-258.

[8] N. H. AlMheiri, A. Rajan, and V. Akre, "Framework for Open Source Software implementation in the Government Sector of Dubai," in *2018 Fifth HCT Information Technology Trends (ITT)*, 28-29 November 2018: IEEE, pp. 71-76, doi: 10.1109/CTIT.2018.8649506.

[9] P. Khandelwal, K. Srivastava, and IT, "Security Vulnerabilities of Websites and Challenges in Combating these Threats," *IITM Journal of Management,* vol. 8, no. 1, pp. 32-36, 2017.

[10] I. Muscat, "Web vulnerabilities: identifying patterns and remedies," *Network Security,* vol. 2016, no. 2, pp. 5-10, 2016/02/01/ 2016, doi: 10.1016/S1353-4858(16)30016-2.
[11] E. Erturk, "Cloud Computing and Cybersecurity Issues Facing Local Enterprises," in *Cloud Security: Concepts, Methodologies, Tools, and Applications*: IGI Global, 2019, pp. 947-969.
[12] E. Bertino and N. Islam, "Botnets and Internet of Things Security," *Computer,* vol. 50, no. 2, pp. 76-79, 2017, doi: 10.1109/MC.2017.62.
[13] R. Chiesa and M. De Luca Saggese, "Data Breaches, Data Leaks, Web Defacements: Why Secure Coding Is Important," in *Proceedings of 4th International Conference in Software Engineering for Defence Applications*, Cham, P. Ciancarini, A. Sillitti, G. Succi, and A. Messina, Eds., May 2016: Springer International Publishing, pp. 261-271, doi: 10.1007/978-3-319-27896-4_22.
[14] A. Lavrenovs and F. J. R. Melón, "HTTP security headers analysis of top one million websites," in *2018 10th International Conference on Cyber Conflict (CyCon)*, 29 May-1 June 2018, pp. 345-370, doi: 10.23919/CYCON.2018.8405025.
[15] T. Book, M. Witick, and D. Wallach, "Automated generation of web server fingerprints," *ArXiv preprint,* vol. abs/1305.0245, 2013.
[16] N. W. Group. "RFC 2616 Hypertext Transfer Protocol--HTTP/1.1." https://www.w3.org/Protocols/rfc2616/rfc2616.html (accessed 03/09, 2020).
[17] R. Fielding and J. Reschke. "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content." Internet Engineering Task Force (IETF). https://tools.ietf.org/html/rfc7231 (accessed 03/09, 2020).
[18] Y. Wang, W. Cai, P. Lyu, and W. Shao, "A Combined Static and Dynamic Analysis Approach to Detect Malicious Browser Extensions," *Security Communication Networks,* vol. 2018, 2018, doi: 10.1155/2018/7087239.
[19] J. Fu, R. Jin, Y. Lin, B. Jiang, and Z. Guo, "Function Risk Assessment Under Memory Leakage," in *2018 International Conference on Networking and Network Applications (NaNA)*, 12-15 October 2018, pp. 284-291, doi: 10.1109/NANA.2018.8648754.
[20] G. Dayanandam, T. V. Rao, D. Bujji Babu, and S. Nalini Durga, "DDoS Attacks—Analysis and Prevention," in *Innovations in Computer Science and Engineering*, Singapore, H. S. Saini, R. Sayal, A. Govardhan, and R. Buyya, Eds., 18-19 August 2019: Springer Singapore, pp. 1-10, doi: 10.1007/978-981-10-8201-6_1.
[21] (2006). *Guide to computer security log management. Recommendations of the National Institute of Standards and Technology (NIST Special Publication 800-92)*. [Online] Available: http://csrc.nist.gov/publications/nistpubs/800-92/SP800-92.pdf
[22] Z. Durumeric *et al.*, "The Security Impact of HTTPS Interception," in *NDSS*, San Diego, CA, 26 February –1 March 2017: Internet Society, pp. 1-14, doi: 10.14722/ndss.2017.23456.
[23] M. S. K. Dabhade and M. Kshirsagar, "Data Security in Cloud Using Aggregate Key and Diffie-Hellman Algorithm," *International Journal of Computer Science and Mobile Computing,* vol. 4, no. 4, pp. 906-923, 2015.
[24] V. Revuelto and K. Socha, "Weaknesses in Diffie-Hellman Key Exchange Protocol," CERT-EU Computer Emergency Response Team, EE.UU, July 7 2016.
[25] L. Chang, H.-C. Hsiao, W. Jeng, T. H.-J. Kim, and W.-H. Lin, "Security Implications of Redirection Trail in Popular Websites Worldwide," presented at the Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, April, 2017.
[26] K. Jamwal and L. S. Sharma, "Clickjacking Attack: Hijacking User's Click," *International Journal of Advanced networking Applications,* vol. 10, no. 1, pp. 3735-3740, 2018.
[27] L. Weichselbaum, M. Spagnuolo, and A. Janc, "Adopting Strict Content Security Policy for XSS Protection," in *2016 IEEE Cybersecurity Development (SecDev)*, 3-4 November 2016, pp. 149-149, doi: 10.1109/SecDev.2016.039.
[28] C. Astudillo, F. Carvajal, J. P. Carvallo, E. Crespo-Martínez, M. Orellana, and R. Vintimilla, "Acometer contra un ERP con Software Libre," *Enfoque UTE,* vol. 9, pp. 138-148, 2018, doi: 10.29019/enfoqueute.v9n1.253.
[29] Y. Zheng and X. Zhang, "Path sensitive static analysis of web applications for remote code execution vulnerability detection," in *2013 35th International Conference on Software Engineering (ICSE)*, 18-26 May 2013, pp. 652-661, doi: 10.1109/ICSE.2013.6606611.

## SOBRE LOS AUTORES

**Leonardo Aguilera Blanco**: Graduado de Ingeniero en Ciencias Informáticas por la Universidad de las Ciencias Informáticas en el 2018. Actualmente se desempeña como especialista general en la Dirección de Seguridad

Informática, enfocándose en la realización de pruebas de penetración. Ha contribuido en la impartición de cursos de posgrado en este campo. Sus áreas de investigación están relacionadas con la seguridad en aplicaciones web y metodologías de pentesting. Coautor del libro Administración Segura de Gestión de Contenidos Web. ORCID: 0000-0002-9609-8967.

**Leobel Rodríguez Chang**: Graduado de Ingeniero en Ciencias Informáticas por la Universidad de las Ciencias Informáticas en el 2017. Actualmente se desempeña como especialista general en la Dirección de Seguridad Informática, enfocándose en la realización de pruebas de penetración. Ha contribuido en la impartición de cursos de posgrado en este campo. Sus áreas de investigación están relacionadas con la seguridad en aplicaciones web y metodologías de pentesting. Coautor del libro Administración Segura de Gestión de Contenidos Web. ORCID: 0000-0002-6724-1787

**Henry Raúl González Brito**: Graduado de Ingeniero Informático por la Universidad de Camagüey y la CUJAE en el año 2005 y Máster en Gestión de Proyectos Informáticos en el 2012 por la Universidad de Ciencias Informáticas. Integra el claustro de varias maestrías impartiendo posgrados en la temática de Seguridad Informática. Actualmente es subdirector del Centro de Telemática (TLM) de la UCI y coordinador de la Especialidad de Posgrado en Seguridad Informática. Sus áreas de investigación están relacionadas con la seguridad en aplicaciones web y metodologías de pentesting. Coautor del libro Administración Segura de Gestión de Contenidos Web. ORCID: 0000-0002-3226-9210

## CONFLICTO DE INTERESES

No existe conflicto de intereses de los autores o de las instituciones a las cuales pertenece en relación al contenido del artículo aquí reflejado.

## CONTRIBUCIONES DE LOS AUTORES

- **Leonardo Aguilera Blanco**: Conceptualización, preparación, creación y desarrollo del artículo.
- **Ing. Leobel Rodríguez Chang**: Contribución a la idea y organización del artículo, sugerencias acertadas para la conformación de la versión final.
- **Henry Raúl González Brito**: Revisión crítica de cada una de las versiones del borrador del artículo y aprobación de la versión final a publicar.