Universidad Europea
Proyecto de Fin de Grado

Reverse Proxy con capacidades de Firewall de aplicación web y aceleración TLS

Alumno: Pedro Pozuelo Rodríguez
Directora: Ana del Valle Corrales Paredes
Titulación: Grado en Ingeniería Informática
Fecha: 14 de julio de 2019

Índice general

Capítulo 1

Introducción

En los últimos años, la mayoría de los ataques en Internet se realizan contra aplicaciones web, con lo que es cada vez más importante contar con una solución que sea capaz de analizar el tráfico web y proteger estas aplicaciones.

Y, no importa el proceso de desarrollo, no importa el número o la calidad de los controles de seguridad desplegados, no existe una aplicación completamente segura; tal como afirma el adagio:

"La seguridad 100% no existe."

Con este punto de partida, y teniendo en cuenta el uso masivo que las aplicaciones hacen del Protocolo de transferencia de hipertexto (en adelante HTTP, de sus siglas en inglésHypertext Transfer Protocol) y del Protocolo seguro de transferencia de hipertexto (en adelante HTTPS, de sus siglas en inglés Hypertext Transfer Protocol Secure), es inmediato identificar porqué la mayoría de los ataques actuales se realizan contra aplicaciones publicadas a través de estos protocolos.

Para protegerlas existen múltiples mecanismos de seguridad, desde herramientas que monitorizan el uso indebido de datos como el Software de prevención de pérdida de datos (en adelante DLP, de sus siglas en inglés Data loss prevention), herramientas de red como son el firewall tradicional o herramientas más dirigidas como son los firewall de aplicación web (en adelante WAF, de sus siglas en inglés, Web Application Firewall).

En entornos con una gran infraestructura, que generen una facturación importante o con un alto riesgo, es habitual que se desplieguen las herramientas mencionadas y muchas otras, incluso herramientas redundantes siguiendo el principio de Defensa en Profundidad, el cual aboga por desplegar múltiples sistemas de protección con el fin de que si uno falla otro pueda detener el ataque.

Pero, en entornos en los que los recursos son más limitados, no es viable desplegar todos los controles de seguridad que serían deseables. Por recursos no sólo se entiende económicamente limitados como para invertir en tecnologías, si no también por personas con el tiempo, el conocimiento y la experiencia como para desplegar y mantenerlas.

En estos entornos es habitual que los controles de seguridad se limiten a un firewall de red o, si se dispone de algo más de presupuesto, un sistema de Gestión Unificada de Amenazas (más conocido como UTM, de sus siglas en inglésUnified Threat Management [1]).

Este tipo de soluciones no son capaces de ofrecer una protección adecuada contra ataques en capa de aplicación. Por ejemplo, los firewall tradicionales son capaces de analizar el tráfico de red en capa 3 del modelo TCP/IP[2] (equivalente a las capas 3 y 4 del *modelo OSI[3]*). Esto implica que , cuando se publica un servicio web, dichos firewalls permitirán todo el tráfico dirigido a estos servicios, con independencia de que se trate de una petición legítima, una petición incorrectamente formada o un ataque.

Para proteger estos servicios web adecuadamente es necesario disponer de tecnologías con capacidad de analizar el tráfico en la capa de aplicación siguiendo la lógica propia del servicio a proteger, como los mencionados WAF.

Otro aspecto que se debe tener en cuenta consiste en la migración constante que se está produciendo de forma generalizada de tráfico sin cifrar - también conocido como en texto plano - a tráfico cifrado HTTPS en el que se encapsula el tráfico HTTP en un canal SSL/TLS (Secure Sockets Layer/Transport Layer Security respectivamente [4]). Históricamente se utilizaban múltiples herramientas perimetrales como son los sistema de detección de intrusiones (en adelante NIDS, de sus siglas en inglés Network Intrusion Detection System) o los mencionados firewalls. Una de las limitaciones que tienen este tipo de herramientas es que no participan en la negociación SSL/TLS, por lo que no son capaces de descifrar el tráfico y no pueden analizar el contenido de las peticiones o sus respuestas. Estas herramientas son, por lo tanto, cada vez menos efectivas a la hora de detectar y bloquear amenazas, pues carecen de la visibilidad adecuada en un gran porcentaje del tráfico.

Por último, se debe considerar otro cambio significativo que se está produciendo en estos últimos años: Cada vez es más habitual desplegar las aplicaciones en La nube (o Cloud, se utilizarán ambos términos indistintamente debido a lo extendido de ambos términos). En estos entornos se diluyen los conceptos de red perimetral o segmentación de red y es más complejo desplegar controles de seguridad perimetrales como los mencionados firewalls, NIDS o UTM. En estos entornos muchos de los componentes de la infraestructura son transparentes para el cliente y no es posible desplegar mecanismos en estas capas.

A modo de referencia, en la tabla Responsabilidades compartidas en el Cloud vemos como los únicos elementos sobre los que se mantiene el control en los distintos modelos del Cloud son la capa de aplicación y los datos (con la excepción del modelo de Software como un Servicio, en adelante SaaS, de sus siglas en inglés Software as a Service).

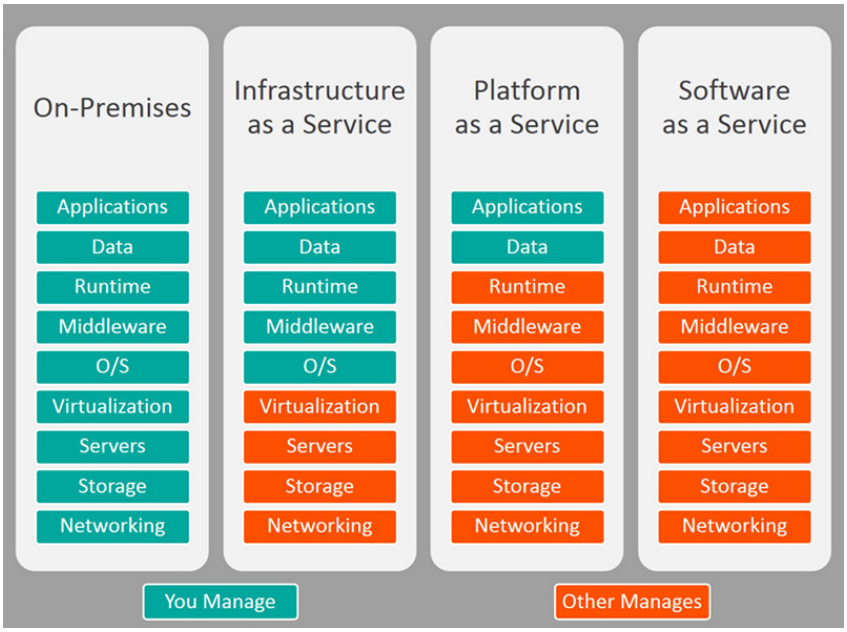


Figura 1.1: Responsabilidades compartidas en los modelos Cloud (fuente MR Informática [5]).

Es por lo tanto, cada vez más importante contar con herramientas que sean capaces de descifrar y analizar el tráfico HTTPS y, según se verá con más detalle en la sección Estado del arte, estas soluciones implican o bien un desembolso económico significativo en el caso de las soluciones privativas o bien se despliegan como módulos de la propia aplicación web en las soluciones de *software libre[6]*.

En este segundo grupo, las soluciones requieren un ejercicio de integración con las aplicaciones

web y consumen recursos del servidor que pueden impactar en el rendimiento. Adicionalmente, para la implementación y configuración adecuada de estos módulos se requiere de una persona con un conocimiento profundo de la seguridad de aplicaciones web que se mantenga informado de las últimas novedades del sector y que administre y actualice el entorno según se detectan nuevos ataques y mecanismos paliativos. Debido a estos factores y la complejidad que el WAF añade a la plataforma web, es habitual que cuando una plataforma web requiere de una actualización importante o una migración de tecnologías, el modulo de WAF se desactive o elimine completamente.

Como respuesta a esta necesidad surge el presente proyecto, cuyo objetivo es construir una solución de software libre con capacidades de WAF, aceleración SSL/TLS, fácilmente desplegable y que minimice el esfuerzo y el impacto que dicha solución tiene sobre la plataforma web actual o futura.

Capítulo 2

Estado del arte

Tal como se ha comentado en el capítulo anterior, un elemento clave para proteger las aplicaciones web es el WAF.

En primer lugar se ha procedido a realizar una análisis comparativo de las diferentes soluciones disponibles en el mercado. Para ello, se ha utilizado el documento de buenas prácticas de OWASP[7] como referencia.

De forma específica, el criterio seguido es que la solución debe ser capaz de proteger la plataforma web implementando una mayoría de los controles de seguridad (referenciados como *Countermeasure* en la página de OWASP[7, apartado A3.2]).

Dentro de las soluciones disponibles, se pueden distinguir principalmente entre soluciones privativas y soluciones de software libre. Este criterio no se circunscribe exclusivamente al modelo de licencias si no que está íntimamente ligado al coste asociado tal como se verá.

2.1 Soluciones WAF privativas

Las soluciones de WAF privativas se caracterizan por emplear un modelo de *licenciamiento privativo[8]*. Se trata de soluciones con un elevado coste y con la imposibilidad de acceder al código o modificarlo. Así mismo, ofrecen una serie de funcionalidades adicionales y una mayor capacidad de procesamiento.

Existen dos arquitecturas principales en este tipo de WAF:

Por un lado existen modelos WAF desplegados en las instalaciones del fabricante y gestionados por él. Este tipo de soluciones suelen estás alojadas en el Cloud (en el modelo de distribución SaaS.

En segundo lugar, están los WAF de tipo appliance o máquina virtual. Se trata de máquinas dedicadas en las que el software requiere de una máquina específica proporcionada por el fabricante. Si bien hardware y software se adquieren conjuntamente, es posible acceder a nuevas funcionalidades adquiriendo nuevas licencias.

2.1.1 Soluciones WAF SaaS

Dentro de las soluciones WAF SaaS, destacan y se han analizado *Cloud Web Application Firewall* [9] de Cloudflare[10] (*Cloudflare* en adelante), *Kona WAF[11]* de *Akamai[12]* e *Incapsula[13]*.

Habitualmente, estos proveedores no se limitan a ofrecer servicios WAF, pues por su infraestructura permite añadir funcionalidades adicionales como son las siguientes.

- Red de distribución de contenidos (en adelante CDN, de sus siglas en inglés, Content Delivery Network).
- Protección contra ataques de denegación de servicio (en adelante DoS, de sus siglas en inglés, Denial-of-service) en capa de aplicación.
- Habilitar el caché de contenido estático.
- Suscripción a listas de reputación de IP, dominios o Localizador de recursos uniforme (en adelante URL, de sus siglas en inglés Uniform Resource Locator).
- Bloqueo de bots maliciosos.
- Sistema de creación de informes.

De hecho, la funcionalidad CDN es el servicio mínimo que se puede contratar a Akamai y Cloudflare; pues es su nicho mercado y su producto principal, siendo el servicio WAF una funcionalidad que ofrecen a sus cliente para dar un valor añadido. Incapsula, por el contrario, proviene de soluciones WAF tipo appliance y su modelo de negocio está más enfocado a estos servicios.

Uno de las principales características que comparten estos proveedores es el modelo de negocio. En todos los casos el coste está asociado al volumen de tráfico que se genere, ya sea en caudal de datos (en adelante Throughput), como es el caso de Incapsula, o por volumen mensual de datos en el caso de Akamai y Cloudflare.

Modo de licenciamiento y coste

En la mayoría de las soluciones no existe un precio oficial de mercado proporcionado por los proveedores y, en estos casos, se ha optado por incluir referencias externas con el fin de disponer información del coste aproximado de estas soluciones.

A modo de referencia, se puede consultar los precios de Akamai y Cloudflare en la tabla de precios CDN. Estos precios se corresponden con sus servicios de CDN y podrá ser superior si se añaden funcionalidades como WAF. Adicionalmente, se debe tener en cuenta que se trata de precios estimativos proporcionados por terceros, pues en el caso de Akamai la lista de precios no es pública y ofrecen un coste ajustado a cada cliente.

	Akamai CDN	**CloudFlare CDN**
6 TB plan	900 USD al mes aprox.	750 USD al mes aprox.
25 TB plan	2800 USD al mes aprox.	2800 USD al mes aprox.
50 TB plan	5500 USD al mes aprox.	5000+ USD al mes aprox.
100 TB plan	8000 USD al mes aprox.	5000+ USD al mes aprox.

Cuadro 2.1: Precios de CDN[14] (consultado en abril de 2019)

En el caso de Incapsula, su solución más económica - el plan *PRO* - tiene un coste de 59 USD por web al mes[15]. Dicha solución soporta SSL de manera limitada y es necesario contratar un plan superior en caso de que se requiera dar servicio a clientes que no soporten la extensión TLS SNI o se requieran certificados con validación extendida (en adelante EV, de sus siglas en inglés, Extended Validation). En esta situación habría que contratar el plan *business* que tiene un coste de 299 USD por web al mes[15].

Implementación y operación

Independientemente de la solución, grosso modo estos son los pasos a realizar para implantar este tipo de soluciones:

1. Cambio en la gestión de certificados SSL.

   Dado que la mayoría de las aplicaciones web deben soportar SSL, es necesario generar nuevos certificados para que el proveedor pueda publicar los servicios web de manera confiable. En la mayoría de los casos el fabricante será responsable del mantenimiento y la operación de dichos certificados.

2. Preparación de un entorno de pruebas - *staging* - en el que se puedan probar las aplicaciones web de manera interna sin impactar a los clientes. Para ello, habitualmente, se redireccionan los dominios a probar en el fichero *hosts* de la máquina cliente.

3. Cambio del direccionamiento DNS.

   Una vez se ha validado que la solución web y el WAF funcionan adecuadamente, se procede a cambiar el direccionamiento ofrecido a nivel DNS para que los clientes se conecten a la plataforma WAF en lugar de a la aplicación web.

La operación de estas plataformas es realizada por el proveedor, por lo que los clientes no necesitan disponer del conocimiento o el tiempo necesario para mantener o actualizar la plataforma WAF.

Ventajas

Una de las principales ventajas que tiene este tipo de soluciones consiste en su independencia respecto a la infraestructura de la aplicación web.

Esta independencia permite realizar cambios en cualquier de las soluciones - WAF o plataforma Web - sin que afecte a la otra. Ya sean cambios en la operación diaria, migraciones de software o rediseño de la arquitectura.

La mencionada independencia no se limita a independencia tecnológica; el hecho de que el WAF y la plataforma web sean completamente independientes también permite asignar roles independientes a cada entorno, lo cual permite implementar seguridad basada en roles (en adelante RBAC, de sus siglas en inglés, role-based access control). Esto no sólo permite mejorar la seguridad del entorno, si no que además evita que el desarrollador web o el administrador de la plataforma web tenga que conocer en detalle la configuración del WAF y viceversa.

Por otro lado, este tipo de soluciones son muy sencillas de implementar, tal como se ha visto en la sección anterior.

Otra ventaja radica en la aplicación de nuevas reglas de seguridad de forma transparente para el cliente. No es necesario dar seguimiento a las últimas vulnerabilidades web que se publican o idear qué reglas o firmas son necesarias, pues el proveedor se hará cargo de su implementación y mantenimiento.

Las soluciones WAF SaaS también permiten contratar diversas modalidades de soporte que garanticen respuesta 24/7 en caso de que se produzca un incidente con el servicio.

Por último, es posible beneficiarse de las funcionalidades adicionales ya mencionadas para mejorar el estado de la seguridad de la plataforma o mejorar la experiencia del usuario.

Desventajas

Una de las principales desventajas radica en el coste económico. Este tipo de soluciones tienen un elevado coste. Esto implica que este tipo de WAF sólo son viables económicamente en portales que generen un beneficio económico importante o aquellos en los que la empresa/entidad responsable de la aplicación pueda asumir su inversión.

Aunque este tipo de soluciones disponen de modalidades relativamente económicas (ver Modo de licenciamiento y coste), lo cierto es que estas modalidades están muy limitadas y es necesario contratar funcionalidades adicionales en la mayoría de los casos. Es un modelo económico muy dirigido a las ofertas personalizadas y suele ser habitual requerir el modelo de licenciamiento *Enterprise* junto con ciertas licencias adicionales, lo cual encarece todavía más el servicio.

En cualquier caso, este tipo de soluciones no están al alcance de pequeñas o medianas empresas o de particulares.

Otra desventaja que tienen este tipo de soluciones consiste en la pocas posibilidades de personalizar las reglas o las firmas a nuestras necesidades. La arquitectura de este tipo de plataformas SaaS consiste en que múltiples clientes comparten la misma plataforma, para lo cual el proveedor requiere mantener un sistema homogéneo para todos los clientes y esto evita que se pueda personalizar el WAF según nuestras necesidades. A modo de ejemplo, en los servicios estándar de este tipo de soluciones no es posible configurar reglas para filtrar las cabeceras HTTP o los parámetros de tipo query en las URL si nuestra aplicación utilizada *Path Parameters* o *URL Routing*, lo que dejaría expuesta la aplicación web a ataques de inyección de código.

2.1.2 Soluciones WAF tipo Appliance

Otra modalidad de soluciones WAF son los de tipo appliance. Dentro de las opciones disponibles en el mercado se han analizado *Imperva WAF Gateway[16]* (*Imperva* en adelante) y *Fortiweb[17]* de la empresa *Fortinet[18]*.

El modelo de negocio tradicional consiste en adquirir una máquina física junto con un paquete de licencias, aunque en los últimos años se han incorporado soluciones virtuales en los catálogos de los principales proveedores de Cloud.

Al igual que sucede con las soluciones SaaS, los proveedores de este tipo de WAF también incluyen mecanismos de seguridad adicionales que no son propiamente funcionalidades WAF. Si bien estas funcionalidades dependen en gran medida del proveedor, a continuación se enumeran algunas de las más interesantes:

- Crear perfiles de las aplicaciones web y filtrar las peticiones web en función de los parámetros permitidos.
- Parcheo virtual de vulnerabilidades mediante la integración del WAF con programas de escaneo de vulnerabilidades.
- Suscripción a listas de reputación de IP, dominios o URL.
- Aceleración TLS.

  Dado que el dispositivo suele estar en la misma red que la aplicación web, es posible que el WAF realice el descifrado del tráfico SSL/TLS y envíe el tráfico sin cifrar a la aplicación web, lo que permite liberar los recursos asignados al cifrado y descifrado en la aplicación web.
- Bloqueo de bots maliciosos.
- Sistema de creación de informes.

- Antivirus.

Modo de licenciamiento y coste

Al igual que sucede con las soluciones SaaS, en las soluciones appliance los proveedores no publican abiertamente el coste que tienen sus productos y optan por realizar presupuestos personalizados dependiendo de las necesidades de cada cliente. Al igual que en el análisis del modelo SaaS se ha optado por incluir referencias externas con el fin de mostrar el coste que tienen este tipo de soluciones.

En el caso de Imperva, su oferta está enfocada a soluciones WAF y firewall de base de datos (en adelante DBF, de sus siglas en inglés, Database Firewall). Se trata de la misma compañía que ha desarrollado y comercializa Incapsula, siendo ésta la alternativa SaaS a Imperva.

Imperva ofrece diversos modelos de appliances según el throughput que son capaces de gestionar, desde 500 Mbps en el modelo más básico - X2010 o X2020 - hasta los 10 Gbps en el modelo X10K.

El coste del appliance de 500 Mbps es de 4200 USD (según [19] y [20]), a lo que hay que sumar el coste anual de licencias y mantenimiento. La licencia necesaria para este modelo tiene un coste que puede ir desde 4800 USD[21] hasta 9600 USD[22]. Por lo tanto, la opción más económica requiere una inversión inicial de 9000 USD y un coste anual mínimo de 4800 USD.

En el caso de que la plataforma web esté alojada en el Cloud (por ejemplo AWS), la opción más económica ofrecida por Imperva tiene un coste mínimo de 8927 USD anuales para una instancia con capacidad de hasta 100 Mbps[23] o 21567 USD por año para el equivalente de la opción appliance de 500 Mbps[24].

Otra solución appliance que se ha analizado es Fortiweb. Si bien sigue un modelo de negocio similar a Imperva, dispone de modelos más económicos. En la tabla de precios Fortiweb se recogen los modelos appliance más económicos ofrecidos por Fortinet.

Modelo	Throughput	Coste de appliance	Coste licencia básica	Coste total
FortiWeb-100D	25 Mbps	5034 USD[25]	755 USD[25]	5789 USD
FortiWeb-400D	100 Mbps	9194 USD[26]	1572 USD[26]	10766 USD
FortiWeb-600D	250 Mbps	14000 USD[27]	2100 USD[27]	16100 USD

Cuadro 2.2: Precios de Fortiweb

La solución AWS de Fortiweb tiene un coste de 5374 USD[28] al año.

Implementación y operación

La implementación de las soluciones tipo appliance es más compleja que en el modelo SaaS debido a que el WAF será parte de la arquitectura del cliente y es necesario analizarla y adaptarla con el fin de incluir este nuevo elemento.

Los pasos que se deben realizar para implantar un WAF de tipo appliance en una arquitectura son los siguientes:

1. Evaluar la arquitectura actual e identificar los potenciales puntos en los que se podría desplegar el WAF.

   Algunos de los puntos de conexión donde se suelen desplegar WAF de este tipo son inmediatamente después del firewall de red o inmediatamente antes de los balanceadores de carga de aplicación, pero puede variar significativamente según la arquitectura. Especialmente se debe tener en cuenta los siguientes elementos:

- Tolerancia frente a fallos (en adelante *failover*).
- Tipo de enrutamiento: estático o dinámico, unicast o multicast, etc.
- Sistemas distribuidos o redundantes.
- Balanceadores de red o de aplicación.
- Volumen de tráfico en los distintos puntos de red a evaluar.

  Por ejemplo, si se instala el WAF en el punto de entrada de una DMZ, el appliance debe ser capaz de gestionar el throughput agregado de todos los servicios publicados en dicha DMZ. Sin embargo, si se instala inmediatamente antes de una aplicación web, el WAF sólo debe analizar el tráfico de dicha aplicación. Por contra, si se despliega una nueva aplicación web es posible que ésta no esté protegida por el WAF.
- Lógica de la aplicación web.

  A modo de ejemplo, en una arquitectura en la que se disponga de un servidor web para servir contenido estático, es posible configurar el WAF para que no acepte el paso de parámetros o que sólo proteja el servidor web de contenido dinámico si se decide aceptar el riesgo asociado.
- Aplicación alojada en un único centro de datos (en adelante CPD) o en varios.

2. Analizar dichos puntos y evaluar el modo de despliegue en el que se desplegará el WAF. Los modos más comunes son modo transparente, en el que el WAF no participa en las capas 3 a 7 del modelo OSI, o como proxy web explícito, en cuyo caso el WAF participa en las capas 3 y 4 y opcionalmente en la capa de aplicación.

3. Evaluar el impacto que este cambio tendrá en el desempeño de la aplicación web, entre otros se debe evaluar la latencia que añade a la red y a la aplicación o cómo afecta al throughput que deben soportar los distintos componentes.

4. Adaptar el diseño de red incluyendo los nuevos elementos.

5. Elegir el o los modelos de appliance que mejor cumple las necesidades del nuevo diseño.

6. Desplegar los nuevos elementos. Típicamente este punto comprende las siguientes actividades:

   (a) Instalación de la solución en el bastidor del CPD (en adelante rack).

   (b) Conexión y configuración de las interfaces de gestión y de los elementos de red necesarios.

   (c) Instalación de los certificados SSL y configuración inicial de las funcionalidades WAF.

   (d) Preparación de un entorno de pruebas de forma similar a la indicada para WAF de tipo SaaS.

   (e) Cambio del direccionamiento de DNS o de IP según proceda.

   Una vez se ha validado que la solución web y el WAF funcionan adecuadamente, se procede a cambiar el direccionamiento ofrecido a nivel de red para que el tráfico de la aplicación web se enrute a través del WAF.

Si se comparan estas actividades con las equivalente de la solución WAF SaaS, esta solución es más compleja de desplegar y se deben tener en cuenta más factores. Esto es así debido a que al elegir esta solución se debe modificar la arquitectura de red y se debe tener en cuenta cómo el WAF va a impactar a la plataforma web.

Por otro lado, dado que la administración y mantenimiento no están delegados en una empresa externa, se debe disponer de las personas adecuadas - con conocimiento, experiencia y tiempo - para administrar y operar el WAF.

Ventajas

Las soluciones WAF de tipo appliance son más personalizables que las soluciones WAF SaaS. Esto es así debido a que son dispositivos dedicados para nuestra plataforma web, y por lo tanto es posible crear reglas específicas que se adapten a nuestras necesidades.

Estas soluciones también cuentan con la ventaja de que toda la información está en nuestras instalaciones, lo cual permite tener mayor control de la información y puede simplificar el cumplimiento de ciertas regulaciones, como son el reglamento europeo RGDP o el Estándar de Seguridad de Datos para la Industria de Tarjeta de Pago (en adelante PCI DSS, de sus siglas en inglés, Payment Card Industry Data Security Standard).

Una ventaja que comparten con las soluciones SaaS es su independencia del software utilizado en la plataforma web. Esto es así debido a que se despliegan como un elemento perimetral y no está conectado con la plataforma web a nivel de aplicación.

Igualmente, comparten las ventajas de independencia operacional; aunque en el caso de los WAF de appliance esta independencia es prácticamente obligatoria debido a que requiere un mayor conocimiento especializado tal como se verá en la siguiente sección.

Al igual que los WAF SaaS, en este tipo de soluciones requiere un servicio de mantenimiento o de suscripción; esto permite que no sea necesario mantenerse al día de las últimas vulnerabilidades y es posible abstraerse parcialmente de cómo proteger la plataforma, pues lo proveedores mantienen las reglas actualizadas como parte del servicio contratado.

Por último, las soluciones WAF de tipo appliance también permiten añadir algunos de los mecanismos de seguridad que no son propiamente de WAF que se han comentado anteriormente.

Desventajas

Tal como se ha anticipado, implementar y administrar este tipo de soluciones requiere de ciertos conocimientos en materia de seguridad, tanto acerca de las técnicas ofensivas más frecuentes, como los mecanismos necesarios para proteger la infraestructura.

Por otro lado, la capacidad de crear nuevas reglas en este tipo de entornos es limitada. Si bien se menciona como ventaja que estos WAF permiten mayor versatilidad que las soluciones SaaS, hay que tener en cuenta que todas las soluciones de este tipo hacen uso de licencias de software privativas, con las restricciones que este tipo de licencias implica: No es posible acceder al código fuente o modificarlo para implantar nuevas funcionalidades o solucionar fallos y el ciclo de vida del appliance es el que el fabricante impone.

Este problema se agrava en las soluciones de tipo appliance debido a que no es infrecuente que el fabricante imponga la renovación de hardware o software de forma agresiva que implique realizar inversiones adicionales no planificadas.

Otra desventaja de este tipo de soluciones es su elevado coste económico. Si bien el coste recurrente suele ser inferior a las soluciones SaaS equivalentes, sigue siendo un coste elevado; por otro lado, las soluciones WAF de tipo appliance requieren la compra de los dispositivos, lo que supone un mayor coste de inversión inicial que en las soluciones SaaS.

Al igual que sucede con las soluciones SaaS, este tipo de soluciones no están al alcance de pequeñas o medianas empresas o de particulares.

2.2 Soluciones WAF de software libre

Dentro de las soluciones WAF de software libre, se han evaluado las siguientes:

- IronBee[29].
- WebCastellum[30].
- RAPTOR[31].
- NAXSI[32].
- OpenWAF[33].
- FreeWAF[34].
- Shadow Daemon[35].
- AQTRONiX WebKnight[36].
- Vulture[37].
- ModSecurity [38].

Entre ellas destaca ModSecurity por ser la solución de software libre más extendida y activa de la comunidad e implementa un número significativo de los controles de seguridad deseables en un WAF.

También destacan OpenWAF y FreeWAF (también conocido como *lua-resty-waf*) debido a que tienen un planteamiento y unas funcionalidades muy interesantes.

Éstas y las demás soluciones se evaluarán en la posterior fase de análisis.

Modo de licenciamiento y coste

En la tabla resumen de WAF de software libre se muestra una comparativa de las diferentes soluciones:

Solución	Licencia	Coste	Soporte
IronBee	Apache License 2.0	Gratuito	No
WebCastellum	Eclipse Public License	Gratuito	No
RAPTOR	GNU GPL 2.0	Gratuito	No
NAXSI	GNU GPL 3.0	Gratuito	Comunidad
OpenWAF	BSD license	Gratuito	Comunidad
FreeWAF	GNU GPL 3.0	Gratuito	Comunidad
Shadow Daemon	GNU GPL 2.0	Gratuito	Comunidad
WebKnight	GNU GPL	145 USD / año	Proveedor
Vulture	No se especifica	Gratuito	Proveedor(de pago)
ModSecurity	Apache License 2.0	Gratuito y de pago*	Comunidad o Proveedor

Cuadro 2.3: Modos de licenciamiento y costes de WAF de software libre

* Modsecurity ofrece la solución WAF de forma gratuita, que cuenta con soporte por parte de la comunidad, y una versión de pago por 495 USD al año que ofrece soporte por parte de Trustware[39] y un conjunto mayor de reglas [40].

Implementación y operación

Las soluciones WAF de software libre se implementan, en la mayoría de los casos, como módulos adicionales al servidor de aplicación web, ya sea Apache HTTP Server[41] (en adelante, Apache), Nginx[42], Internet Information Services[43], etc.

Esto implica que el WAF debe ejecutarse como parte del servicio web y su configuración y operación depende del administrador de la plataforma web.

Si en los WAF de tipo appliance se ha visto que las tareas de implantación tienen cierta complejidad en lo relativo a analizar la plataforma web desde un punto de vista de red, en el caso de los WAF que se ejecutan como parte del servicio web la complejidad radica en que el WAF debe integrarse dentro de la plataforma web.

Este tipo de WAF requiere que se revise el dimensionamiento de la plataforma web debido a que consumen recursos del servidor y puede afectar a su rendimiento.

Estas son las tareas que se deben abordar de forma genérica a la hora de implantar un WAF de software libre:

1. Evaluar la plataforma web actual e identificar qué soluciones WAF son compatibles con el servicio web.
2. Desplegar un un entorno de pruebas equivalente a la plataforma web actual o prueba de concepto (en adelante PoC, de sus siglas en inglés, Proof of concept).
3. Desplegar el WAF en el entorno de pruebas.
4. Realizar pruebas exhaustivas sobre el nuevo entorno, especialmente pruebas funcionales y de rendimiento.
5. Evaluar el impacto del WAF, entre otros se debe evaluar errores en la lógica de aplicación, latencia que añade, aumento en el consumo de recursos o cómo afecta al throughput soportado por la plataforma.

   Tradicionalmente este punto y el anterior se deberán ejecutar de forma reiterativa hasta que los resultados sean concluyentes y la nueva configuración se considere suficientemente robusta como para desplegarla en producción.
6. Desplegar el WAF en el entorno de producción (si la plataforma lo permite, se recomienda realizar el despliegue de forma escalonada) y realizar un conjunto de pruebas similar a las realizadas en el entorno de pruebas.

Aunque en este tipo de soluciones se han enumerado menos pasos que en las soluciones anteriores, esto es debido a que las tareas dependen en gran medida del software y la plataforma elegidos, por lo que las actividades se han identificado a más alto nivel.

Ventajas

Los WAF de software libre son más económicos que las alternativas privativas. Si bien es cierto que esto no quiere decir que sean gratis, pues requieren personas que los administren y consumen una serie de recursos de la plataforma web.

Pero, aun eligiendo alguna de las opciones de pago y con soporte como ModSecurity, el coste en licencias es muy inferior a las otros modelos privativos.

Debido al tipo de licenciamiento, estas soluciones se distribuyen como software libre, con todas las ventajas inherentes a este tipo de software entre las que destaca desde un punto de vista funcional el

acceso al código fuente y la capacidad de modificarlo de acuerdo a nuestras necesidades entre otras. Está fuera del alcance del proyecto evaluar las ventajas generales del software libre frente a otro tipo de licencias.

No en todos los casos, pero en la mayoría de las soluciones analizadas el equipo de desarrollo es un conjunto de individuos pertenecientes a distintos ámbitos o empresas, por lo que se elimina la estricta dependencia del proveedor. Si bien este modelo tiene sus ventajas e inconvenientes, lo cierto es que se elimina la dependencia de realizar una migración del software si se considera conveniente (por ejemplo, para alinear dicha migración según una planificación propia en lugar de una planificación impuesta).

Se puede pues decir que estas soluciones son más adaptables a las necesidades específicas de cada entorno.

Desventajas

Este tipo de soluciones son más difíciles de implementar y de mantener. El hecho de que estén integradas como parte de la plataforma web hace que sea complejo diferenciar los roles del administrador de la plataforma web del administrador del WAF. Por lo tanto, la misma persona debe tener conocimiento de ambas plataformas y ambos campos del conocimiento, lo cual no es común y puede provocar errores de configuración de alguna de las plataformas.

Precisamente debido a la gran dependencia existente entre el software WAF y de la plataforma web, es necesario analizar en detalle las configuraciones y las reglas que se habilitarán, pues el proceso de depuración de errores es más complejo y los fallos son más difíciles de identificar y solucionar.

Las actividades de actualización de componentes y migraciones de software son así mismo más complejas, pues se deben actualizar o migrar ambas plataformas conjuntamente.

Por otro lado, el soporte en este tipo de soluciones puede ser de menor calidad que en las alternativas privativas. Al fin y al cabo en muchos casos el soporte depende de la comunidad y si se elige un WAF que tenga una comunidad reducida, es probable que no se obtenga una respuesta inmediata. Por supuesto, si se elige una solución con soporte o se contrata un servicio de consultoría, se puede paliar esta desventaja. Si se trata de un entorno de producción en el que el tiempo de caída del servicio es crítico, se recomienda contratar algún servicio de soporte o consultoría.

2.3 Comparativa de soluciones WAF

En la tabla comparativa de las soluciones WAF se recoge un resumen de las características diferenciadoras de las soluciones evaluadas.

Características	WAF SaaS	WAF Appliance	WAF Software libre
Independencia de la plataforma web	⇑ Muy buena	⇑ Buena	⇓ Mala
Independencia operacional (RBAC)	⇑ Muy buena	⇑ Buena	⇓ Mala
Complejidad de despliegue y administración	⇑ Muy baja	⇑ Baja	⇓ Alta
Coste económico	⇓ Alto	⇓ Alto	⇑ Bajo
Soporte técnico	⇑ Bueno	⇑ Bueno	⇓ Limitado
Información accesible por terceros	⇓ Sí	⇑ No	⇑ No
Adaptabilidad / Personalización	⇓ Muy baja	⇓ Baja	⇑ Alta
Acceso al código fuente	⇓ No	⇓ No	⇑ Sí
Funcionalidades adicionales (por defecto)	⇑ Muy buenas	⇑ Buenas	⇓ Limitadas

Figura 2.1: Comparativa de las soluciones WAF

En la tabla comparativa con propuesta se muestra una comparativa incluyendo la solución propuesta en el presente proyecto.

Características	WAF SaaS	WAF Appliance	WAF Software libre	Propuesta
Independencia de la plataforma web	⇑ Muy buena	⇑ Buena	⇓ Mala	⇑ Buena
Independencia operacional (RBAC)	⇑ Muy buena	⇑ Buena	⇓ Mala	⇑ Buena
Complejidad de despliegue y administración	⇑ Muy baja	⇑ Baja	⇓ Alta	⇒ Media
Coste económico	⇓ Alto	⇓ Alto	⇑ Bajo	⇑ Bajo
Soporte técnico	⇑ Bueno	⇑ Bueno	⇓ Limitado	⇓ Limitado
Información accesible por terceros	⇓ Sí	⇑ No	⇑ No	⇑ No
Adaptabilidad / Personalización	⇓ Muy baja	⇓ Baja	⇑ Alta	⇑ Alta
Acceso al código fuente	⇓ No	⇓ No	⇑ Sí	⇑ Sí
Funcionalidades adicionales (por defecto)	⇑ Muy buenas	⇑ Buenas	⇓ Limitadas	⇑ Buenas

Figura 2.2: Comparativa de las soluciones WAF incluyendo la solución propuesta

Capítulo 3

Requisitos y casos de uso

3.1 Requisitos candidatos

En primer lugar se identifican los requisitos que se intentarán cumplir con la solución propuesta.

Se agrupan los requisitos según estén más orientados al componente WAF o al componente de TLS.

Requisitos orientados principalmente al componente WAF:

WAF-Req1. La solución debe poder ejecutarse en un sistema operativo o máquina independiente de la plataforma de la aplicación web con el objetivo de garantizar independencia en las tareas de administración y permitir aplicar un modelo RBAC.

WAF-Req2. La solución debe disponer de un conjunto básico de políticas de auditoría o bloqueo que permitan proteger la aplicación web frente a los ataques más comunes.

WAF-Req3. La plataforma debe permitir implementar parches virtuales frente a ataques conocidos.

WAF-Req4. La solución debe permitir la elaboración de reglas personalizadas según las necesidades específicas de la plataforma del cliente.

WAF-Req5. La plataforma debe ser compatible con el modelo de licencias de *software libre[6]* tipo Licencia Pública General de GNU (en adelánte GPL, de sus siglas en inglés GNU General Public License[44, Licencia GPL]) o Licencia Pública General Reducida de GNU (en adelánte LGPL, de sus siglas en inglés GNU Lesser General Public License[45, Licencia LGPL]).

WAF-Req6. La plataforma debe ser fácilmente integrable en la plataforma web del cliente.

Tal como se ha detallado en la sección Estado del arte, el despliegue y la operación de los WAF de software libre es complejo, y esta complejidad es uno de las causas de su escasa adopción. Es por ello que se considera de vital importancia para su adopción que la solución ofrecida sea fácil de desplegar u operar. Para ello, un factor determinante es minimizar los cambios necesarios en la infraestructura previa y las actividades de administración de la plataforma.

WAF-Req7. Escalabilidad. La plataforma debe permitir la ampliación o reducción de recursos de forma dinámica. Debido a que la solución está situada entre la plataforma web y las peticiones de los clientes, es clave que la solución WAF pueda ampliarse o reducirse según sea necesario.

WAF-Req8. La plataforma debe generar logs de seguridad exportables a soluciones externas de gestión de información y eventos de seguridad (en adelante SIEM de sus siglas en inglés, Security information and event management).

A continuación se recogen los requisitos asociados al componente de TLS:

TLS-Req1. La solución debe poder participar en la negociación TLS, presentando certificados confiables a los clientes.

TLS-Req2. La solución deber poder gestionar los certificados presentados a los clientes incluyendo soporte a la extensión SNI de TLS.

TLS-Req3. La plataforma debe soportar TLS versión 1.3, HTTP/2 y otros elementos incluidos en las *buenas prácticas de TLS[46]*.

TLS-Req4. Debe permitir aplicar soluciones de SSL offloading, entre el WAF y los frontales de la plataforma web, o permitir cifrado punto a punto.

En caso de utilizar la funcionalidad de SSL offloading:

a) Esta opción es recomendable en entornos controlados en los que prima el rendimiento.

b) Las comunicación entre el WAF y la plataforma web debe permitir tráfico HTTP.

En caso de cifrado punto a punto:

a) Esta opción es recomendable en las soluciones en las que prime la seguridad o en aquellos escenarios en los que no se tenga el control de elementos intermedios, como por ejemplo en entornos Cloud o multi-datacenter.

b) Las comunicación entre el WAF y la plataforma web debe permitir tráfico HTTPS.

c) El WAF debe confiar en la CA que firma los certificados de la plataforma web o, alternativamente, en los certificados hoja.

3.2 Identificación de actores

Se han identificado los siguientes actores (algunos de ellos se identifican también en inglés debido a que múltiples referencias los mencionan en inglés).

- **Cliente (Client en inglés)**. Se utiliza este término para referirse al cliente web que consume los servicios HTTP o HTTPS.
- **Atacante (Black Hat Hacker en inglés)**. Se trata de un tipo de cliente malintencionado.
- **Plataforma web (Web Servers en inglés)**. Se considera toda la infraestructura necesaria para servir los contenidos web. En esta infraestructura no se incluyen los elementos desarrollados en el presente proyecto.
- **Autoridad de certificación** (en adelante CA). Es el elemento encargado de firmar los certificados TLS. El cliente debe confiar en la CA o en los certificados hoja alternativamente. En caso de cifrado punto a punto el WAF deberá confiar en los certificados presentados por la plataforma web.
- **Plataforma WAF+TLS**. Se trata de la plataforma propuesta en el presente proyecto.

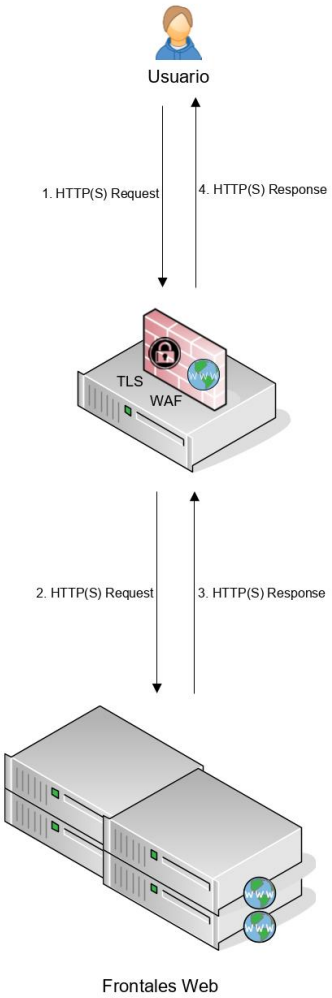
3.3 Casos de uso

Estos son los casos identificados.

1. Caso de uso: Petición identificada como legítima de recurso web.

   - Actores: Cliente, plataforma web, plataforma WAF+TLS.
   - Descripción: Se trata del caso de uso que sucede con mayor frecuencia, pues representa las peticiones legítimas de clientes a la plataforma web. Se pueden representar las comunicaciones en los siguientes pasos:
     (a) El cliente realiza una petición web a la solución propuesta en el presente proyecto.
     (b) La plataforma WAF+TLS evalúa la petición, la considera legítima y la envía a la plataforma web.
     (c) La plataforma web recibe la petición y envía una respuesta a la plataforma WAF+TLS.
     (d) La plataforma WAF+TLS recibe la respuesta y se la envía al cliente.

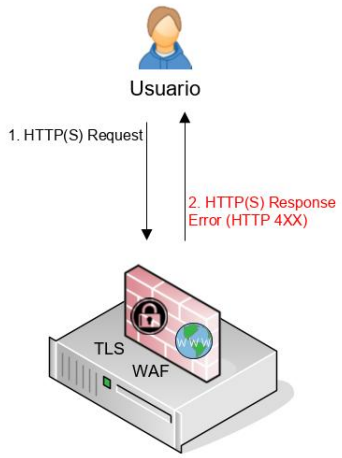
     Se debe tener en cuenta que en este caso de uso se incluyen peticiones legítimas y falsos negativos cuando el WAF falla al detectar un ataque.
   - Diagrama:



2. Caso de uso: Petición identificada como no legítima de recurso web.

   - Actores: Atacante, plataforma WAF+TLS.
   - Descripción: Este caso de uso se dará siempre que la plataforma WAF+TLS reciba una petición y la considere un ataque. En este caso se pueden identificar los siguientes pasos:
     (a) El atacante realiza una petición web a la plataforma WAF+TLS.
     (b) La plataforma WAF+TLS recibe la petición, la evalúa y, considerándola como ataque, envía un mensaje de error al atacante.

     A tener en cuenta que la plataforma WAF+TLS ha considerado que el cliente es un atacante y este caso de uso se dará tanto en ataque reales como con falsos positivos (cuando se diagnostica como ataque a una petición legítima).
   - Diagrama:

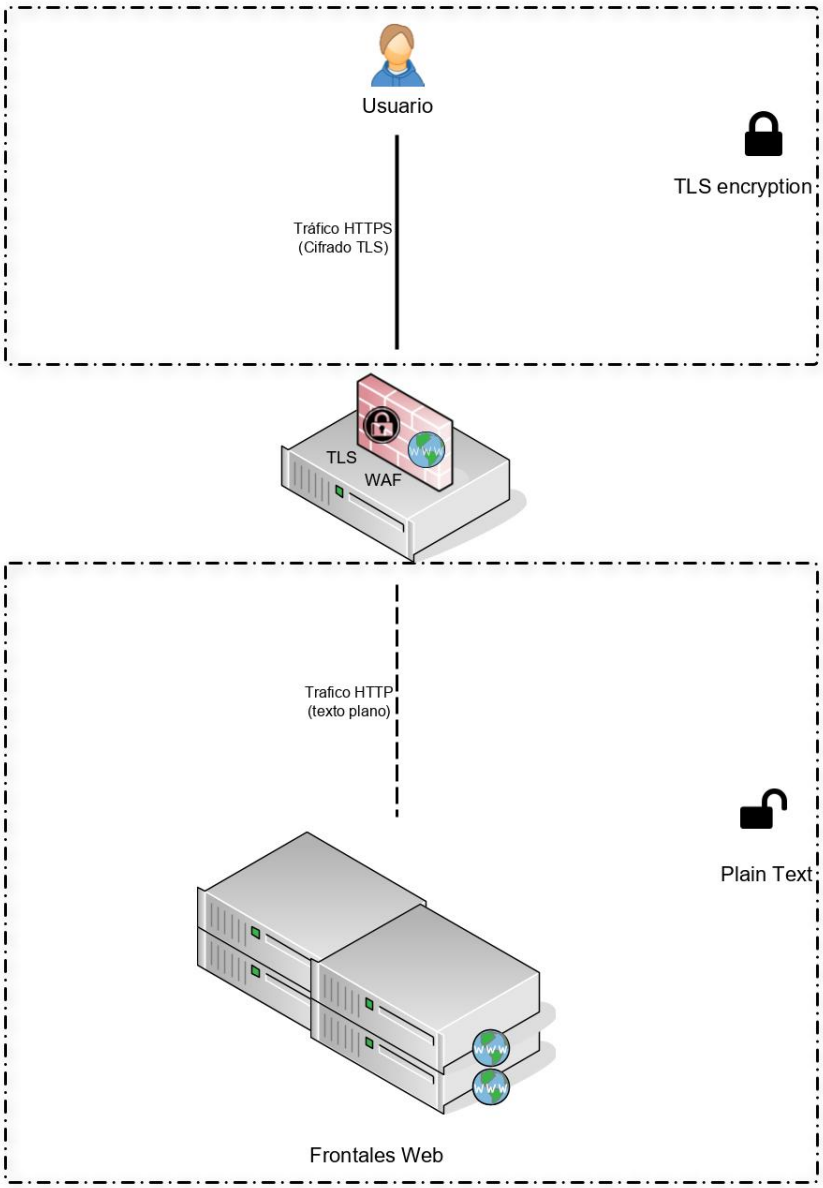


3. Caso de uso: Petición HTTPS en un entorno con TLS offloading.

    - Actores: Cliente, plataforma web, plataforma WAF+TLS.
    - Descripción: Se trata de una derivada del Caso de primer caso de uso. En este escenario las peticiones HTTPS realizadas por el cliente se envían sin cifrar (o en texto plano) a la plataforma web.

      Con ello se consigue reducir la carga que debe soportar la plataforma web y optimizar sus recursos.

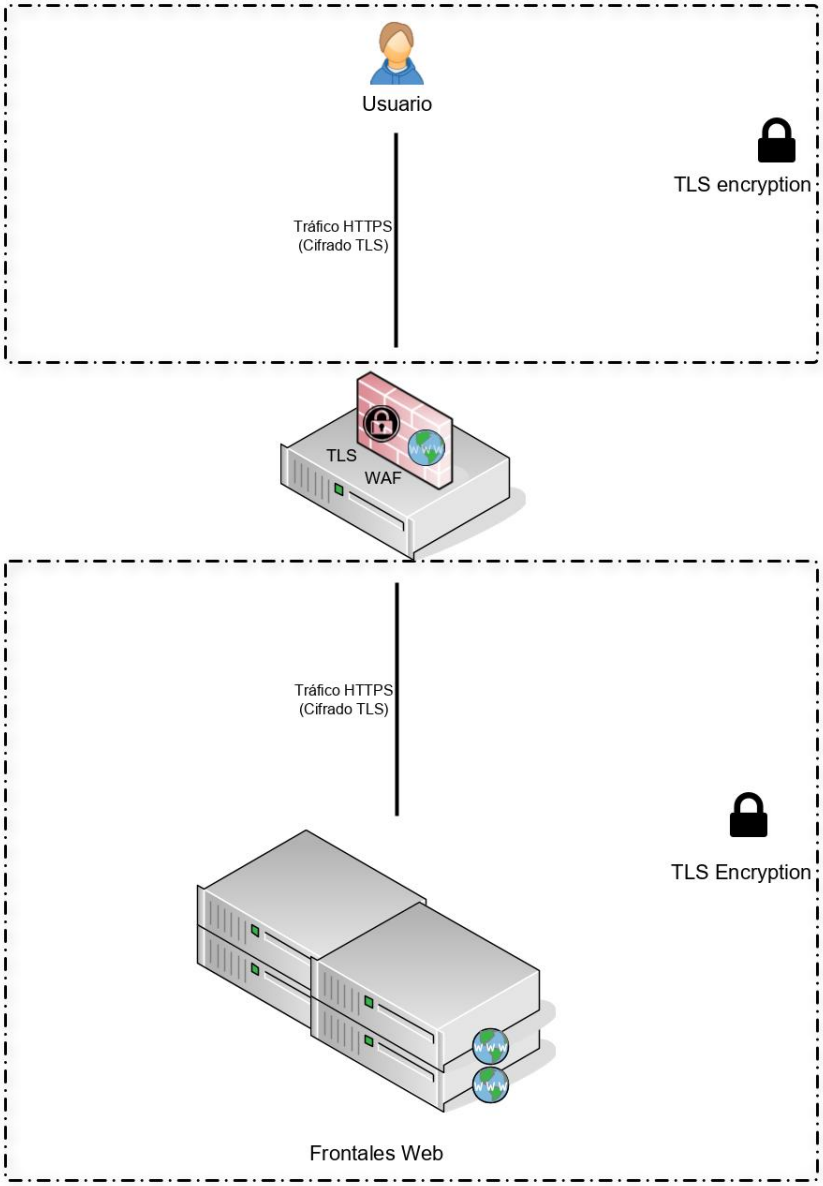
      Sin embargo, también se aumenta el riesgo a un ataque en caso de que un potencial atacante tuviese acceso a la infraestructura situada entre la solución WAF y la plataforma web. Es por ello que esta solución sólo se recomienda en escenarios en los que los elementos situados entre ambos estén protegidos y monitorizados adecuadamente.
    - Diagrama:



4. Caso de uso: Petición HTTPS con cifrado punto a punto.

    - Actores: Cliente, plataforma web, plataforma WAF+TLS.
    - Descripción: Se trata de un caso de uso alternativo al anterior. En este escenario se mantienen las comunicaciones cifradas también entre el WAF y la plataforma web, con lo que se consigue el cifrado punto a punto.

      Para ello se establecen dos túneles TLS: El primero entre el cliente y la plataforma WAF y un segundo túnel entre el WAF y la plataforma web.
    - Diagrama:



5. Caso de uso: Petición de validación y firma de certificado TLS a una CA de confianza.

   - Actores: Certification Authority, plataforma WAF+TLS.
   - Descripción: Este caso de uso tiene como objetivo obtener un certificado digitalmente por una CA que sea de confianza para el cliente. Se puede dividir en dos flujos similares.

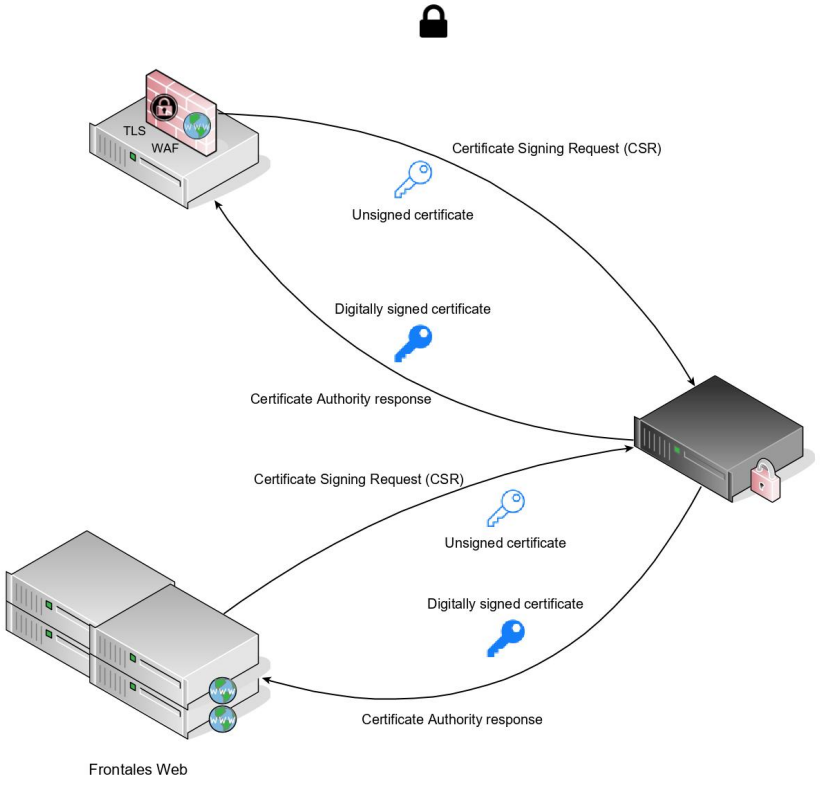
     El primer flujo se da entre el WAF y la CA siguiendo los siguientes pasos:

     (a) El WAF - o un administrador del WAF - genera un certificado CSR [47, Wikipedia] asociado a su clave privada que siga el *estándar X.509 v3 [48, IETF 5280]*.
     (b) Se envía el certificado a la CA solicitando que ésta lo firme.
     (c) La CA valida que la petición es legítima y firma el certificado con su clave raíz o, alternativamente, una clave intermedia.
     (d) La CA envía el certificado firmado digitalmente el WAF.
     (e) Una vez el WAF disponga de dicho certificado, este puede presentar sus servicios web a los clientes y estos podrán validar al WAF durante la negociación TLS si confían en el certificado raíz de la CA o en la jerarquía de certificados.

     El segundo flujo es opcional. Se trataría de un flujo similar al descrito para el WAF pero el solicitante de certificado sería la plataforma web en este caso.

     Se considera que es opcional debido a que la plataforma web sólo estará expuesta al WAF y se podría crear una relación de confianza sin necesidad de hacer uso de una CA externa. No obstante, dado que actualmente obtener este tipo de certificados no tiene un coste económico, y que tampoco debería tener un impacto operacional significativo, se recomienda seguir el mismo proceso que para el WAF y hacer uso de certificados firmados digitalmente por un CA.

     En este proyecto, para la gestión de certificados del WAF se intentará hacer uso del protocolo ACME (Automatic Certificate Management Environment[49, Estándar ACME]) mediante *Let's Encrypt [50]* con el objetivo de automatizar las actividades de creación y renovado de certificados.
   - Diagrama:



Capítulo 4

Diseño de la solución

Como es habitual en los Proyectos de Fin de Grado, uno de los retos más interesantes que se encuentran consiste en afrontar, abrazar y controlar la incertidumbre sobre cómo será la solución final; dicha incertidumbre es inherente a todo proyecto de software, pero en un PFG es mayor debido a que la investigación tiene a su vez un mayor peso y es determinante.

En el presente proyecto, a la hora de afrontar inicialmente el diseño de la solución se desconoce qué tecnologías se van a utilizar específicamente y éstas se deciden tras evaluarlas tanto individualmente como su capacidad de integración entre ellas y con elementos externos.

Para ello se sigue un modelo de desarrollo evolutivo similar al modelo de prototipos. Para lo cual se parte de la definición de un diseño a alto nivel en el que se definen los componentes y sus funciones más representativas. A continuación se empieza a realizar un proceso iterativo en el que se van probando los componentes y añadiendo nuevos elementos. El resultado de dicha fase es una solución funcional inicial que cumpla los requisitos definidos. Sobre dicho modelo se realiza una revisión de calidad que permita contar con una solución suficientemente madura y estable.

No obstante, dado que se espera que la solución sea algo en constante cambio, se ha elegido una metodología de desarrollo ágil [51], tal como se describe más adelante.

4.1 Componentes principales

Estos son los componentes que se han identificado para la solución:

- El primer componente que se ha identificado es un WAF con licencia de software libre. Este elemento es el componente más importante de la solución y marca en gran medida los demás componentes que participen en la capa de aplicación.

  Durante la fase de construcción se evalúan qué componentes WAF cumplen los requisitos y se elige un candidato.

- El segundo componente de la solución es el software responsable de gestionar las comunicaciones HTTPS. Este software es el encargado de cifrar y descifrar el tráfico SSL / TLS.

- El tercer componente es el software de virtualización o de contenedores. Uno de los puntos clave de la solución es garantizar la facilidad para implementarla de forma poco intrusiva. Para ello se considera que un software de contenedores como Docker [52]. Este tipo de soluciones está ampliamente asentada en el mercado y existe un movimiento significativo de migración de los entornos tradicionales a entornos de contenedores.

- Otro componente estrechamente relacionado con el componente anterior es el software automatización de despliegue y gestión de la solución. Este componente es opcional y por lo tanto se tendrá en cuenta que la opción deber ser sustituible en diferentes implementaciones con el fin de garantizar la máxima portabilidad de la solución.
- Otro componente del que depende la solución es el conector con los elementos de almacenamiento persistente. Este componente es el encargado de proporcional un servicio de compartir información entre recursos y garantizar la perdurabilidad de la información necesaria. Se hará foco en reducir el uso o impacto de este componente al mínimo y que todo el entorno pueda ser reconstruido con el menor esfuerzo posible con el fin de ofrecer un entorno dinámico siguiendo los principios CI/CD ya mencionados.
- Un componente opcional a considerar es el gestor de certificados HTTPS. Este componente será el responsable de aprovisionar, renovar y revocar certificados que sean confiables para los clientes, por lo que deberá comunicarse con una entidad certificadora.
- Por último, tenemos dos componentes directamente relacionados con la personalización de la herramienta por el usuario final. Se trata de las reglas de auditoría o bloqueo del WAF, así como las reglas de configuración del software de TLS. Estos componentes serán los ficheros o parámetros de configuración de los dos primeros componentes que se han identificado.

4.2 Diseño de alto nivel

Partiendo de los requisitos definidos, se plantea el diseño a alto nivel tal como se muestra en el Diagrama de alto nivel de la solución en el que se reflejan los componentes de la solución y los principales actores.

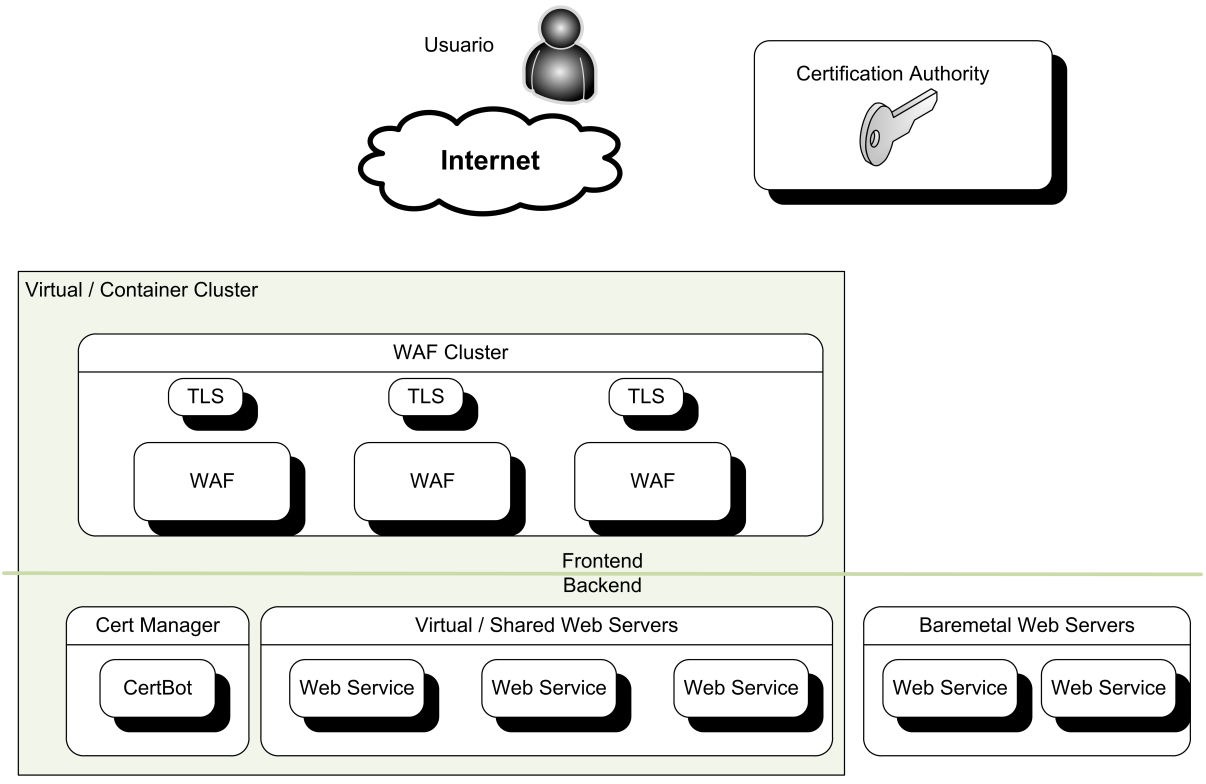


Figura 4.1: Diseño de alto nivel

4.3 Construcción de la solución

Tal como se comentaba anteriormente, debido al alto grado de incertidumbre que se tiene del aspecto final de la solución, junto con la necesidad de poder realizar cambios sobre el modelo de forma rápida, se ha optado por seguir una metodología de desarrollo ágil [51].

Durante el proceso de desarrollo del proyecto se utilizará como referencia o mantra el Manifiesto por el Desarrollo Ágil de Software y como modelo de liberación de versiones se siguen los principios de integración continua, entrega continua (en adelante CI/CD, de sus siglas en inglés, Continuous Integration and Continuous Deployment or Continuous Delivery [53, What is CI/CD]).

4.4 Análisis y selección de tecnologías

En esta sección se evalúan las distintas opciones disponibles y se elige una tecnología candidata sobre la que construir la solución.

4.4.1 Web Application Firewall

En primer lugar, de acuerdo a los requerimientos establecidos - La plataforma debe ser compatible con el modelo de licencias de software libre - se han descartado las soluciones WAF privativas y aquellas que no estén licenciadas bajo una licencia del tipo GPL. En este segundo grupo se encuentran WebKnight - sólo compatible con el software de aplicación web Internet Information Services[43] sobre plataformas Microsoft Windows - y Vulture, con licenciamiento BSD y que requiere un sistema operativo FreeBSD.

Otras soluciones se han descartado debido a que están discontinuadas desde hace tiempo, como IronBee que no se ha actualizado desde hace más de 3 años y WebCastellum desde hace 5 años.

Otra tecnología candidata que se ha descartado ha sido RAPTOR debido a que no soporta tráfico SSL/TLS (La solución debe poder participar en la negociación TLS).

NAXSI se ha considerado que no cumple los requisitos básicos que debe tener un WAF actualmente (La solución debe disponer de un conjunto básico de políticas de auditoría o bloqueo que permitan proteger la aplicación web frente a los ataques más comunes), pues sólo implementa dos controles de seguridad - de los referenciados en el documento de buenas prácticas de OWASP[7, apartado A3.2]); concretamente, sólo protege ataques de tipo *Cross-site scripting* (en adelante XSS[54]) e inyecciones SQL (en adelante SQLi[55]). Además, se han producido duras críticas acerca de las funcionalidades que ofrece [56].

OpenWAF por su parte es una iniciativa con un planteamiento y unas funcionalidades muy interesantes, pero que lleva más de dos años sin publicar una nueva versión y además carece de suficiente madurez para considerarse su uso en un entorno de producción (la última versión publicada es la 0.0.4).

Respecto a FreeWAF (también conocido como *lua-resty-waf*) está en una situación muy similar a OpenWAF, pues su última versión tiene más de 2 años y se trata de la versión 0.11.1[57].

En el caso de Shadow Daemon, no se cumple el requisito de independencia (La solución debe poder ejecutarse en un sistema operativo o máquina independiente de la plataforma de la aplicación web), pues se trata de un conector de aplicación web con una funcionalidades interesantes, pero por diseño requiere de su integración con el framework de desarrollo y sólo es compatible con PHP, Perl y Python. Por lo tanto no es posible implementarlo como elemento externo e independiente de la plataforma web.

Por último, se ha considerado ModSecurity. Esta tecnología está disponible en dos ramas de desarrollo, la rama 2.x (versión 2.9.3 en el momento en el que se escribe este documento) y la rama 3.x

(versión 3.0.3 actualmente).

La rama no cumple los requisitos de independencia (La solución debe poder ejecutarse en un sistema operativo o máquina independiente de la plataforma de la aplicación web), debido a que se despliega como un módulo de Apache y no permite que su despliegue sea agnóstico de la plataforma web; por lo tanto se descarta esta versión de ModSecurity.

La rama 3.x por el contrario incorpora entre sus novedades una reescritura completa del código y la posibilidad de enlazarse a distintas tecnologías de servicios web.

Cabe mencionar que las tecnologías que abiertamente no cumplen los requisitos definidos se han descartado, pero aquellas que cumplen con los requisitos pero que llevan tiempo sin actualizarse o están en una fase temprana de desarrollo se mantienen como opciones alternativas en caso de que la opción candidata no cumpla con las expectativas.

Tecnología candidata

Se ha elegido como tecnología candidata ModSecurity en su versión 3.x, pues cumple todos los requisitos y es un referente entre las tecnologías de WAF de software libre.

Servicio web

Uno de los componentes sobre los que se debe apoyar el WAF es el servicio web. Este componente se comporta como un interfaz entre el WAF y las peticiones y respuestas del cliente y el servicio web.

Se han identificado los siguientes servicios web compatibles con ModSecurity en su versión 3:

- Internet Information Services (IIS) [43].
- Apache HTTP Server (Apache) [41].
- Nginx [42].

IIS se ha descartado debido a que sólo es compatible con plataformas Windows, tal como se comentó en la sección Estado del arte.

Tanto Apache como Nginx cumplen los requisitos funcionales, pero Nginx tiene mejores capacidades de proxy inverso.

Respecto al licenciamiento, Nginx utiliza *Licencia BSD simplificada o licencia FreeBSD (de 2 cláusulas)* [58], la cual es compatible con GPL.

Tecnología candidata

Se ha elegido como servicio web Nginx.

4.4.2 Software criptográfico

Uno de los puntos clave para proporcionar un canal cifrado y seguro alineado con las mencionadas mejores prácticas de TLS [46] o de PCI DSS [59] es el uso de TLS en su versión 1.3 y HTTP/2.

Se han descartado por lo tanto las soluciones que no cumplen estos requisitos, como por ejemplo LibreSSL [60].

Una vez realizado este primer filtro, se han seleccionado las siguientes opciones:

- GnuTLS [61].
- OpenSSL [62].
- Network Security Services (NSS [63]).

A la hora de valorar estas opciones se ha identificado que a nivel de licenciamiento, sólo GnuTLS utiliza una licencia de tipo GPL.

Sin embargo, uno de los componentes previamente seleccionados - Nginx - sólo es compatible con OpenSSL [64].

OpenSSL por su parte tiene una licencia doble, licencia [65, Apache License 1.0] y [66, licencia SSLeay], esta última heredada del proyecto original en el que se basa OpenSSL.

Para resolver este problema se han identificado diferentes opciones:

1. Cambiar Nginx por una alternativa que soporte GnuTLS.
2. Desplegar los componentes de cifrado y de WAF en sistemas independientes.
3. Revisar la compatibilidad de licencias.

Para evaluar la viabilidad de la primera opción se han identificado qué servicios son compatibles con GnuTLS. Según la Wikipedia [67] este proyecto es usado principalmente en clientes TLS y sólo es compatible con el servicio web Apache, el cual a su vez tiene el mismo tipo de licencia que OpenSSL (Apache License). Por lo que tiene la misma incompatibilidad de licencias ya mencionada.

Debido a la falta de compatibilidad de GnuTLS con otros servicios web, la segunda opción mantiene la dependencia de Apache y su licencia, por lo que se debe descartar igualmente esta opción.

Para evaluar la tercera opción, dado que la librería OpenSSL es una de las más utilizadas, se ha investigado cual es el acercamiento que otros proyectos de software libre.

El acercamiento de algunos proyectos es utilizar una licencia tipo GPL modificada en la que se añade una excepción para esta librería [68]. Esta solución la han seguido proyectos como *GNU Wget* y *climm*. Para ello se deben adjuntar archivos binarios dentro del proyecto y modificar la licencia GPL ofrecida por la GNU; si bien es una opción viable, no es la opción deseable.

Investigando cual es el acercamiento de otros proyectos para mantener su licencia GPL y utilizar OpenSSL, se ha identificado que OpenSSL ha iniciado un proceso de cambio de licencia para atajar este problema. Para ello han decidido cambiar su licencia por una compatible con GPL ([69, Anuncio del cambio de licencia]). El cambio de licencia se ha aplicado recientemente ([70, *Pull request del cambio*]) y ya es efectivo en las versiones proporcionadas por la versión *Buster* de Debian GNU/Linux.

Con este cambio se elimina la incompatibilidad de licencias que se había identificado.

Tecnología candidata

Se ha elegido como tecnología candidata OpenSSL, pues cumple todos los requisitos y es una de las librerías más extendidas y probadas.

4.4.3 Software de virtualización

A la hora de elegir software de virtualización, todas las soluciones que se conocen cumplen con los requisitos y objetivos del proyecto, por lo que se ha optado por Docker debido a que es una solución ampliamente utilizada y en constante expansión.

Tecnología candidata

La solución que se ha elegido es [52].

Dentro de cada instancia del contenedor Docker se desplegará Nginx y ModSecurity y OpenSSL:

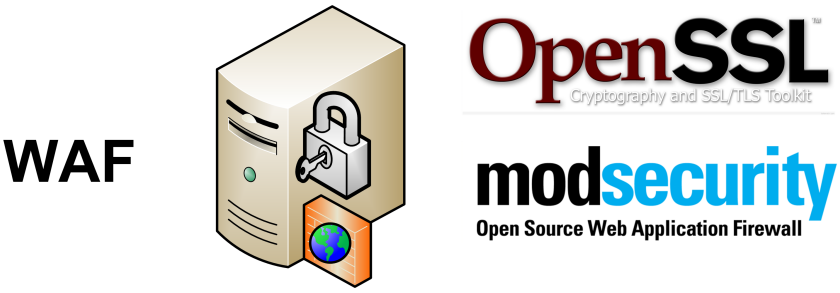


Figura 4.2: Componentes desplegados en cada instancia WAF

4.4.4 Software de automatización u orquestación

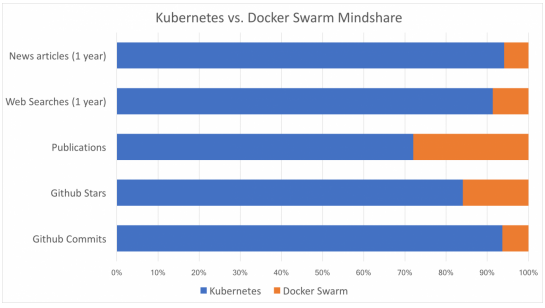
Esta tecnología será la responsable de garantizar el cumplimiento de los siguientes requisitos: La plataforma debe ser fácilmente integrable en la plataforma web del cliente. Escalabilidad. La plataforma debe permitir la ampliación o reducción de recursos de forma dinámica.

Para poder ofrecer estas funcionalidades, se han evaluado las siguientes tecnologías:

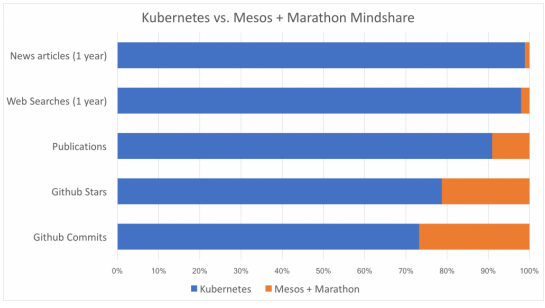
- Kubernetes (K8s) [71].
- Apache Mesos [72].
- Docker Swarm [73].

Todas las tecnologías cumplen las necesidades funcionales definidas y todas ellas tiene un licenciamiento compatible con las licencias de tipo GPL.

Para poder ofrecer que la solución sea fácilmente integrable, se ha evaluado el uso de estas soluciones y se ha comprobado como Kubernetes es la solución más ampliamente extendida (ver los gráficos Kubernetes y Docker Swarm y Kubernetes y Apache Mesos).



(a) Uso y popularidad de software de Kubernetes y Docker Swarm(fuente Platform9 [74])



(b) Uso y popularidad de software de Kubernetes y Apache Mesos(fuente Platform9 [75])

Si además comparamos la evolución que estas tecnologías han tenido históricamente, la diferencia es aun mayor como se puede comprobar en el gráfico de Popularidad de software de orquestación.

Cabe mencionar que existen soluciones Kubernetes adaptadas a los entornos de los principales proveedores del Cloud, como son *Azure Kubernetes Service (AKS) [77]* o *Amazon Elastic Kubernetes*

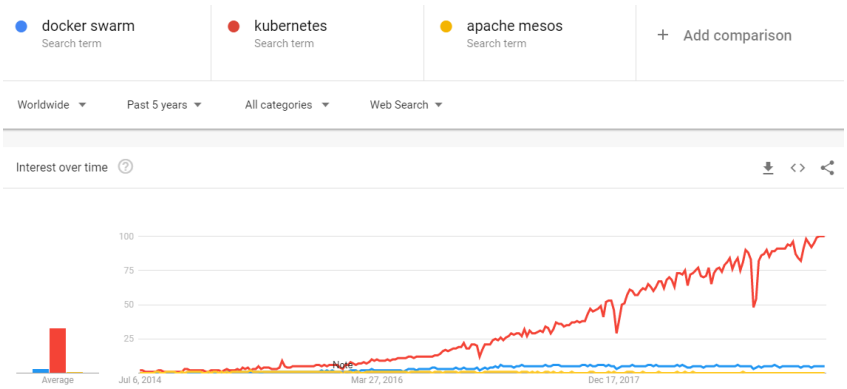


Figura 4.4: Histórico de popularidad de software de orquestación (fuente Google [76])

Service (Amazon EKS) [78]. Estas soluciones cuentan con la ventaja de que facilitan su despliegue y mantenimiento mediante el uso de herramientas propias, pero cuentan con una fuerte desventaja: Aumentan la cautividad de la solución con un único proveedor.

Sin embargo, la migración de una solución agnóstica del proveedor del Cloud a un proveedor específico es sencilla.

Por lo tanto se ha optado por no utilizar ninguna solución específica del Cloud y utilizar la tecnología Kubernetes sin funcionalidades añadidas como son los *Ingress Controllers [79]* o API de proveedores del Cloud. Esto permitirá que la solución sea fácilmente desplegable en el mayor número de entornos posible.

Tecnología candidata

Se ha elegido Kubernetes [71] como tecnología candidata para implementar la automatización de la plataforma.

Kubernetes será responsable de desplegar las instancias WAF que sean necesarias, garantizar su disponibilidad y volver a desplegarlas si hubiese algún problema con alguna de ellas.

4.4.5 Servicio de almacenamiento

El objetivo de este componente es garantizar un servicio de intercambio de información entre los distintos componentes de la misma y, en caso de que sea necesario, el almacenamiento persistente de la información. Pero, tal como se ha comentado anteriormente, se intentará minimizar su uso. De ser posible se utilizará alguna solución ya presente en la infraestructura.

Entre las soluciones a evaluar existen dos aproximaciones posibles: Servicios tradiciones como Samba [80] o NFS (estándar RFC7530 [81]) o servicios del Cloud como Amazon S3[82] o Google Storage[83].

Tal como se ha identificado a la hora de identificar la tecnología de automatización, las soluciones proporcionadas por los fabricantes del Cloud limitan las posibles plataformas que adopten la solución, por lo que se prefiere utilizar servicios de almacenamiento e intercambio de archivos independientes del proveedor.

Tanto Samba como NFS cubren los requisitos, por lo que cualquiera de estas tecnologías son válidas. Sin embargo, dentro de los entornos de contenedores el protocolo NFS está más extendido.

Tecnología candidata

Se utilizará por tanto un servicio de almacenamiento basado en [81], preferiblemente un recurso externo a la solución propuesta.

4.4.6 Políticas o reglas de auditoría

Tecnología candidata

4.5 Arquitectura

Tal como se ha comentado anteriormente, se procede a construir la solución de forma iterativa.

Debido a las funcionalidades añadidas por las últimas versiones del software, se utilizarán las últimas versiones estables de todo el software, para lo cual se descargará el código fuente y compilará según se requiera.

Esto permite a su vez minimizar las dependencias entre la solución y el sistema operativo, lo que facilitará que ésta se adapte a otros sistemas si así se requiere.

No obstante, se utilizará como base un sistema operativo Debian GNU/Linux.

4.5.1 Web Application Firewall

El primer componente que se debe construir es el WAF. Para ello, se han definido las siguientes capas de seguridad:

1. **ModSecurity**. Es la primera capa de seguridad y la funcionalidad que propiamente se considera como WAF.

   Se utilizarán las reglas de WAF definidas por OWASP [84].

2. **Seguridad de la Cookie**. Se aplicarán una serie de atributos con el fin de evitar el robo o uso indebido de cookies.

3. **Cabeceras HTTP de seguridad**. Se aplicarán una serie de cabeceras HTTP que protegen contra diversos ataques.

4. **Cabecera HTTP Strict Transport Security (HSTS)**. Esta cabecera HTTP evita que se realicen peticiones mediante canales no cifrados.

5. **HTTP/2**. La versión 2 de HTTP aporta una serie de mejoras respecto a las versiones anteriores en materia de seguridad y de rendimiento.

El WAF constará pues de los componentes identificados en el Diagrama del WAF

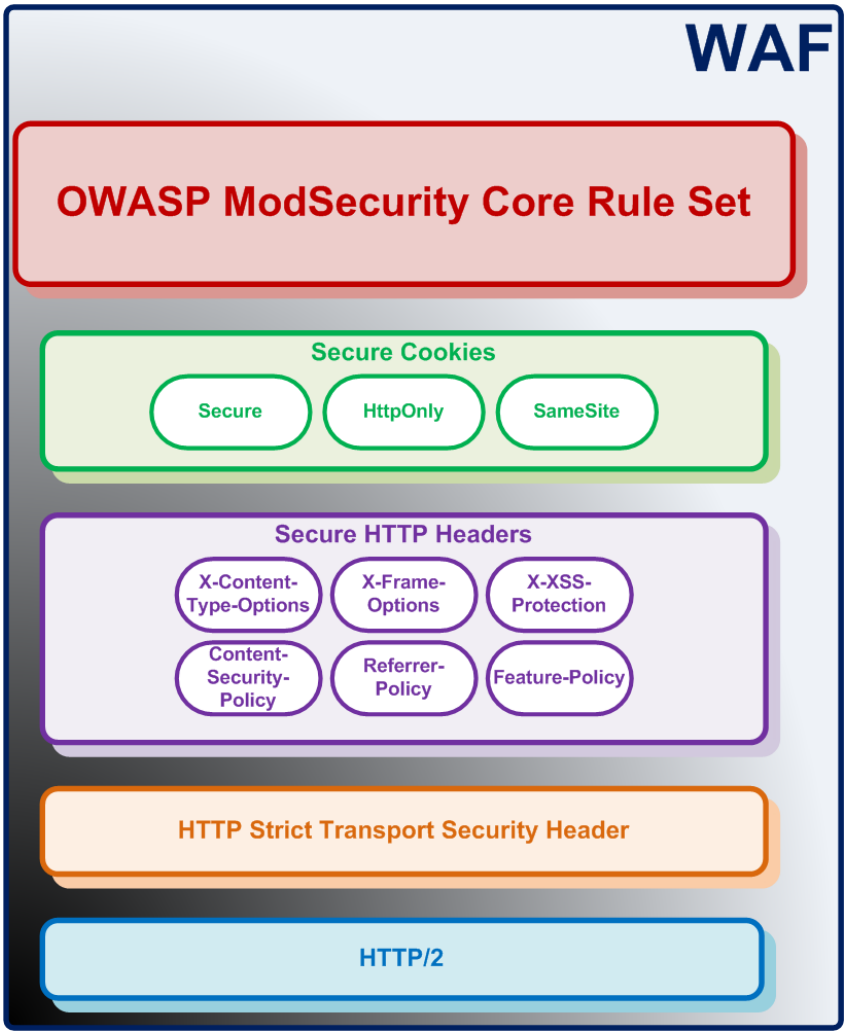


Figura 4.5: Diseño de alto nivel

Construcción de ModSecurity

Se procede a construir el componente base sobre el que se basarán los demás componentes, esto es un WAF sobre Nginx.

En primer lugar se debe instalar el conector de ModSecurity.

```
1 cd /opt/
2 git clone https://github.com/SpiderLabs/ModSecurity-nginx
```

A continuación se procede a descargar e instalar la última versión disponible de Nginx.

```
wget http://nginx.org/download/nginx-${NGINX_VERSION}.tar.gz
tar xvzf nginx-${NGINX_VERSION}.tar.gz
cd nginx-${NGINX_VERSION}
./configure \
  --with-compat --add-dynamic-module=../ModSecurity-nginx \
  --user=www-data \
  --group=www-data \
  --without-mail_pop3_module \
  --without-mail_imap_module \
  --without-mail_smtp_module \
  --with-file-aio \
  --with-http_ssl_module \
  --with-stream \
  --with-stream_ssl_module \
  --with-http_stub_status_module \
  --with-http_gzip_static_module
make
make install
```

Se descarga e instala ModSecurity.

```
cd /opt/ \
git clone https://github.com/SpiderLabs/ModSecurity \
cd ModSecurity/ \
git checkout -B v3/master origin/v3/master \
sh build.sh \
git submodule init \
git submodule update \
./configure \
make \
make install \
```

Por último, se configura ModSecurity con una regla de bloqueo que permita realizar pruebas.

```
#Configure ModSecurity
mkdir -p ${ModSecPath}
cd ${ModSecPath}
cat >> ${ModSecPath}/main.conf <<EOL
# From https://github.com/SpiderLabs/ModSecurity/blob/master/
# modsecurity.conf-recommended
#
# Edit to set SecRuleEngine On
Include "${ModSecPath}/modsecurity.conf"
# Basic test rule
SecRule ARGS:testparam "@contains test" "id:1234,deny,status:403"
EOL

# Set ModSecurity in enforce mode:
cp /opt/ModSecurity/modsecurity.conf-recommended ${ModSecPath}/modsecurity.conf
sed -i 's/SecRuleEngine DetectionOnly/SecRuleEngine On/' ${ModSecPath}/modsecurity.conf

# Load ModSecurity module in nginx:
loadmodule="load_module ${ModSecPath}/ngx_http_modsecurity_module.so;"
sed -i '\-^pid-s-$-\n\n'"${loadmodule}"'\n-' ${nginxConfFile}

# Enable ModSecurity in nginx server sections:
enablemodsecurity="modsecurity_rules_file ${ModSecPath}/main.conf;"
sed -i '\-^http-s-$-\n\t'"${enablemodsecurity}"'\n-' ${nginxConfFile}
enablemodsecurity="modsecurity on;"
sed -i '\-^http-s-$-\n\t'"${enablemodsecurity}"'-' ${nginxConfFile}


# Copy nginx modules
cp /opt/nginx-${NGINX_VERSION}/objs/ngx_http_modsecurity_module.so ${ModSecPath}
cp /opt/ModSecurity/unicode.mapping ${ModSecPath}
```

El WAF envía una respuesta HTTP 403 cuando se aplica dicha regla a la petición tal como se ve en la captura (Respuesta de WAF a una petición no legítima).

Figura 4.6: Respuesta de WAF a una petición no legítima.

Soporte HTTP/2

El siguiente componente que se debe añadir soporte en Nginx mediante el parámetro --with-http_v2_module.

Por lo que se debe compilar Nginx con los siguientes parámetros:

```
./configure \
  --with-compat --add-dynamic-module=../ModSecurity-nginx \
  --user=www-data \
  --group=www-data \
  --without-mail_pop3_module \
  --without-mail_imap_module \
  --without-mail_smtp_module \
  --with-file-aio \
  --with-http_ssl_module \
  --with-http_v2_module \
  --with-stream \
  --with-stream_ssl_module \
  --with-http_stub_status_module \
  --with-http_gzip_static_module
make
make install
```

A continuación se debe habilitar el protocolo en el fichero de configuración de nginx (*nginx.conf*).

Para ello, se modifica el parámetro *listen* del bloque *server* destinado a HTTPS:

```
listen       443 ssl http2 default_server;
listen       [::]:443 ssl http2 default_server;
```

La segunda línea se añade con el fin de que la solución tenga soporte para IPv6.

Cabecera HTTP Strict Transport Security

La cabecera HTTP Strict Transport Security [85](HSTS en adelante) permite implantar una política de seguridad que obliga al cliente a comunicarse mediante canales cifrados SSL/TLS.

Esto permite proteger el tráfico frente a ataques que puedan interceptar las comunicaciones como por ejemplo un *ataque de intermediario*MitM (en adelante MitM, del inglés Man-in-the-middle).

Para habilitar esta política, se añade la siguiente cabecera a la configuración:

```
add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;
```

Los parámetros especificados son los siguientes:

- max-age=31536000. Indica la duración de la política en el cliente web. En este caso un año.
- includeSubDomains. Indica que dicha política se debe aplicar a todos los subdominios.
- always. Indica que la política se debe añadir a cualquier tipo de respuesta HTTP.

Cabecera de seguridad *X-Content-Type-Options*

La siguiente cabecera que se añade es *X-Content-Type-Options* [86]. Esta cabecera previene que un cliente pueda realizar *MIME type sniffing* [87]. Esta funcionalidad representa un riesgo de seguridad debido a que permite potencialmente transformar un tipo MIME no ejecutable en ejecutable.

Para habilitar esta política, se añade la siguiente cabecera a la configuración:

```
1 add_header X-Content-Type-Options "nosniff" always;
```

El parámetro nosniff previene que el cliente aplique un tipo MIME diferente al indicado por el servidor web.

El parámetro always tiene un comportamiento similar a lo descrito en otras cabeceras.

Cabecera de seguridad *X-Frame-Options*

Otra cabecera de seguridad a habilitar es *X-Frame-Options* [88]. Esta cabecera previene que la petición se enmarque dentro de un frame/iframe de un dominio que no haya sido autorizado, una técnica habitual en los ataques del tipo Clickjacking.

Para habilitar esta política, se añade la siguiente cabecera a la configuración:

```
1 add_header X-Frame-Options SAMEORIGIN always;
```

El parámetro SAMEORIGIN indica que las peticiones sólo podrán realizarse en frames/iframes pertenecientes al mismo dominio.

Cabecera de seguridad *X-XSS-Protection*

La siguiente cabecera que se habilita es *X-XSS-Protection* [89]. Esta cabecera indica al navegador web que habilite y aplique el filtro contra vulnerabilidades del tipo Cross-site scripting[54].

Para habilitar esta política, se añade la siguiente cabecera a la configuración:

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

El parámetro 1; mode=block indica que el filtro contra vulnerabilidades XSS debe estar activado obligatoriamente.

Cabecera de seguridad *Referrer-Policy*

La cabecera *Referrer-Policy* [90] tiene como función limitar qué información se envía en la cabecera *Referrer*.

Esta cabecera se envía cuando el usuario hace click en un enlace y, por defecto, se envía la URL completa de la página actual a la nueva página. Esto implica que, si existe algún tipo de información en la URL actual, la página web destino tendrá acceso a ella.

Este comportamiento puede aprovecharse para realizar ataques de falsificación de petición en sitios cruzados (en adelante XSRF, del inglés Cross-site request forgery (XSRF)) o Phishing entre otros.

Para habilitar esta política, se añade la siguiente cabecera a la configuración:

```
1 add_header Referrer-Policy "strict-origin-when-cross-origin";
```

El parámetro strict-origin-when-cross-origin indica lo siguiente:

- Si la dirección destino es más insegura que la dirección actual (HTTPS → HTTP), no se envía ningún dato en la cabecera *Referrer*.
- Si la dirección destino pertenece al mismo dominio, se envía la URL completa.
- Si la dirección destino pertenece a un dominio diferente, se enviará el dominio de la dirección actual y se eliminará cualquier parámetro o ruta de la URL.

Cabecera *Feature-Policy*

La cabecera *Feature-Policy* [91] define las funcionalidades permitidas. Se trata de una política bastante reciente cuyo borrador ha sido publicado por el W3C (World Wide Web Consortium) en abril de 2019 [92].

Actualmente se han publicado las siguientes directivas:

- ambient-light-sensor
- autoplay
- accelerometer
- camera
- display-capture
- document-domain
- encrypted-media
- execution-while-not-rendered
- execution-while-out-of-viewport
- fullscreen
- geolocation
- loading-frame-default-eager
- loading-image-default-eager
- gyroscope
- magnetometer
- microphone
- midi
- payment
- picture-in-picture
- speaker
- sync-xhr
- usb
- wake-lock
- vr / xr

Se han definido las siguientes políticas con el fin de garantizar compatibilidad sin sacrificar la seguridad.

```
1
2  set $FeaturePolicy '';
3  set $FeaturePolicy "${FeaturePolicy}geolocation            'self';";
4  set $FeaturePolicy "${FeaturePolicy}sync-xhr               'self';";
5  set $FeaturePolicy "${FeaturePolicy}payment                'self';";
6  set $FeaturePolicy "${FeaturePolicy}camera                 'none';";
7  set $FeaturePolicy "${FeaturePolicy}usb                    'none';";
8  set $FeaturePolicy "${FeaturePolicy}magnetometer           'none';";
9  set $FeaturePolicy "${FeaturePolicy}accelerometer          'self';";
10 set $FeaturePolicy "${FeaturePolicy}gyroscope              'self';";
11 set $FeaturePolicy "${FeaturePolicy}speaker                'self';";
12 set $FeaturePolicy "${FeaturePolicy}ambient-light-sensor   'none';";
13 set $FeaturePolicy "${FeaturePolicy}microphone             'none'";
```

Los valores elegidos son:

- 'self'. Indica que la funcionalidad está permitida en las peticiones al dominio y en peticiones dentro de los iframes que pertenezcan al mismo dominio.

- 'none'. Indica que la funcionalidad no está permitida.

Se espera ir ajustando los parámetros a medida que se implementen estas políticas en distintos entornos y evolucionen las directivas.

Cabecera de Políticas de Seguridad de Contenido

La cabecera de Políticas de Seguridad de Contenido [93] (en adelante CSP, del inglés Content-Security-Policy) define una serie de directivas de seguridad que permiten proteger la plataforma de ataques Cross-site scripting[54] o Inyección SQL[55].

Estas políticas son muy versátiles pero también pueden provocar que el contenido no se entregue adecuadamente.

Para prevenirlo, se ha identificado el contenido multimedia y las redes sociales más comunes:

- 500px.com
- 500px.org
- cloudflare.com
- facebook.com
- facebook.net
- flickr.com
- google-analytics.com
- googleapis.com
- google.com
- imgur.com
- jquery.com
- reddit.com
- staticflickr.com
- youtube.com

Utilizando estos servicios como referencia, se han codificado las siguientes listas:

```
set $defaultSrcDomains '';
set $defaultSrcDomains "${defaultSrcDomains} https://google.com";
set $defaultSrcDomains "${defaultSrcDomains} https://youtube.com";
set $defaultSrcDomains "${defaultSrcDomains} https://facebook.com";
set $defaultSrcDomains "${defaultSrcDomains} https://fonts.google.com";
set $defaultSrcDomains "${defaultSrcDomains} https://fonts.googleapis.com";
set $defaultSrcDomains "${defaultSrcDomains} https://ajax.googleapis.com";
set $defaultSrcDomains "${defaultSrcDomains} https://www.google-analytics.com";
set $defaultSrcDomains "${defaultSrcDomains} https://cdnjs.cloudflare.com";
set $defaultSrcDomains "${defaultSrcDomains} https://code.jquery.com";
set $defaultSrcDomains "${defaultSrcDomains} https://connect.facebook.net";
set $defaultSrcDomains "${defaultSrcDomains} https://imgur.com";
set $defaultSrcDomains "${defaultSrcDomains} https://i.imgur.com";
set $defaultSrcDomains "${defaultSrcDomains} https://s.imgur.com";
set $defaultSrcDomains "${defaultSrcDomains} https://500px.com";
set $defaultSrcDomains "${defaultSrcDomains} https://drscdn.500px.org";
set $defaultSrcDomains "${defaultSrcDomains} https://www.reddit.com";
set $defaultSrcDomains "${defaultSrcDomains} https://www.flickr.com";
set $defaultSrcDomains "${defaultSrcDomains} https://c1.staticflickr.com";

set $ImageSrcDomains '';
set $ImageSrcDomains "${ImageSrcDomains} https://ssl.google-analytics.com";
set $ImageSrcDomains "${ImageSrcDomains} https://s-static.ak.facebook.com";

set $frameSrcDomains '';
set $frameSrcDomains "${frameSrcDomains} https://www.facebook.com";
set $frameSrcDomains "${frameSrcDomains} https://s-static.ak.facebook.com";

set $scriptSrcDomains '';
set $scriptSrcDomains "${scriptSrcDomains} https://ssl.google-analytics.com";
set $scriptSrcDomains "${scriptSrcDomains} https://connect.facebook.net";
```

Por último, se ha configurado y habilitado la cabecera CSP como se muestra a continuación:

```
set $CSP '';
set $CSP "${CSP}default-src      'self' $defaultSrcDomains";
set $CSP "${CSP}object-src       'none';";
set $CSP "${CSP}frame-src        $frameSrcDomains;";
set $CSP "${CSP}img-src          data: $ImageSrcDomains;";
set $CSP "${CSP}script-src       $scriptSrcDomains 'unsafe-inline';";
set $CSP "${CSP}style-src        https://fonts.googleapis.com;";
set $CSP "${CSP}font-src         https://themes.googleusercontent.com;";

add_header Content-Security-Policy "$CSP" always;
```

Control de seguridad de *Cookies*

Otro elemento que se deben proteger son las cookies de la plataforma web (), para ello el WAF modificará las cookies de la plataforma web añadiendo los siguientes atributos si no estuviesen previamente:

- Secure. Previene que la cookie sea enviada mediante canales no cifrados.
- HttpOnly. Previene que la cookie sea accesible por una medio que no sea HTTP/HTTPS. Por lo tanto la cookie no puede ser accesible por un script en el navegador, lo que proporciona protección contra ataques de XSS.
- SameSite. Previene que la cookie se envíe a un dominio diferente, por lo que ayuda a proteger la plataforma frente a ataques Cross-site request forgery

  Debido a que es una práctica habitual el uso de la misma cookie en distintos dominios se ha optado por aplicar este filtro en una modalidad más permisiva: SameSite=Lax. Esta opción evita que se envíe la cookie en peticiones POST entre otras.

Nginx por defecto soporta esta funcionalidad para la pagina raíz ("/"), pero esta protección no se aplica a las demás páginas.

Para solucionarlo se hace uso de un módulo externo, nginx_cookie_flag_module [94], con este módulo se pueden habilitar los atributos identificados a todas las cookies con independencia de la URL.

En primer lugar se debe descargar el módulo.

```
cd /opt/
git clone https://github.com/AirisX/nginx_cookie_flag_module.git
```

A continuación se debe modificar la compilación de Nginx añadiendo el siguiente parámetro `--add-module=../nginx_cookie_flag_module`.

Y se compila Nginx con los siguientes parámetros:

```
./configure \
  --with-compat --add-dynamic-module=../ModSecurity-nginx \
  --add-module=../nginx_cookie_flag_module \
  --user=www-data \
  --group=www-data \
  --without-mail_pop3_module \
  --without-mail_imap_module \
  --without-mail_smtp_module \
  --with-file-aio \
  --with-http_ssl_module \
  --with-http_v2_module \
  --with-stream \
  --with-stream_ssl_module \
  --with-http_stub_status_module \
  --with-http_gzip_static_module
```

4.5.2 Software criptográfico

Se procede a configurar las comunicaciones HTTPS.

Para ello, lo primero será crear un certificado auto-firmado de pruebas con su correspondiente clave privada.

```
# Temporal self-signed certificates. They will be replaced after Let's Encrypt service is running.
commonName="www.fakedomain.com"
country="ES"
organization="PFG WAF"
organizationalUnit="PFG WAF Test"
state="Madrid"
emailAddress="info@fakedomain.com"
certFile="testCert"
openssl req -batch -nodes -newkey rsa:2048 -keyout "${certFile}.key" -out "${certFile}.csr" -subj "${subject}"
openssl x509 -req -days 365 -in "${certFile}.csr" -signkey "${certFile}.key" -sha256 -out "${certFile}.crt"
openssl x509 -in "${certFile}.crt" -out "${certFile}.pem" -outform PEM
```

A continuación se debe configurar Nginx para que ofrezca este certificado a los clientes.

```
  listen          443 ssl http2 default_server;
  listen          [::]:443 ssl http2 default_server;
  server_name <SET_DOMAIN> *.<SET_DOMAIN>;
  ssl_certificate       /usr/local/nginx/certs/testCert.pem;
  ssl_certificate_key   /usr/local/nginx/certs/testCert.key;

  ssl_session_cache     shared:SSL:1m;
  ssl_session_timeout   5m;

  ssl_protocols TLSv1.1 TLSv1.2 TLSv1.3;
  ssl_ciphers   HIGH:!aNULL:!MD5;
  ssl_prefer_server_ciphers   on;
```

Siendo:

- listen. Indica el puerto donde el servicio está escuchando sobre IPv4 e IPv6, usando canales cifrados SSL, HTTP/2 y siendo el servicio por defecto en caso.
- ssl_certificate . Indica la clave privada y el certificado que se han creado anteriormente.
- ssl_session_cache. Indica cómo gestionar la caché de las sesiones SSL.
- ssl_session_timeout. Indica el tiempo de vida de la sesión SSL en caso de inactividad. Dado que la negociación SSL tiene un coste computacional elevado, es preferible mantener las sesiones inactivas en caso de que se vayan a retomar.
- ssl_protocols. Indica las versiones de TLS compatibles. Por defecto se utiliza la versión superior que soporte el cliente.
- ssl_ciphers. Indica los algoritmos de cifrados soportados. Se desactivan algoritmos con cifrado NULL y MD5 explícitamente y sólo se permiten algoritmos considerados con seguridad criptográfica alta.
- ssl_prefer_server_ciphers . Indica que el WAF será el responsable de elegir el algoritmo de cifrado utilizado.

4.5.3 Contenedores de software

Se ha elegido Docker como tecnología de contenedores.

Para construir y probar la solución se han definido los siguientes contenedores:

- *ccwaf*. Contenedor sobre el que se despliegan los componentes WAF vistos.
- *letsencrypt*. Contenedor sobre el que se configurará el gestor de certificados que aprovisionará a los demás servicios web con certificados creador por una entidad certificadora de confianza.
- *basicwebserver*. Contenedor en el que se desplegará un servicio web básico. Este elemento se podrá eliminar una vez la solución esté funcionando en el entorno de producción, pero se distribuirá como parte de la solución para facilitar las tareas de configuración y depuración de errores.

Todos los contenedores se han construido sobre una imagen base de Debian GNU/Linux del repositorio de Docker Hub (*debian:buster-slim* [95]).

Se puede consultar el código en el apéndice (fichero Dockerfile del WAF).

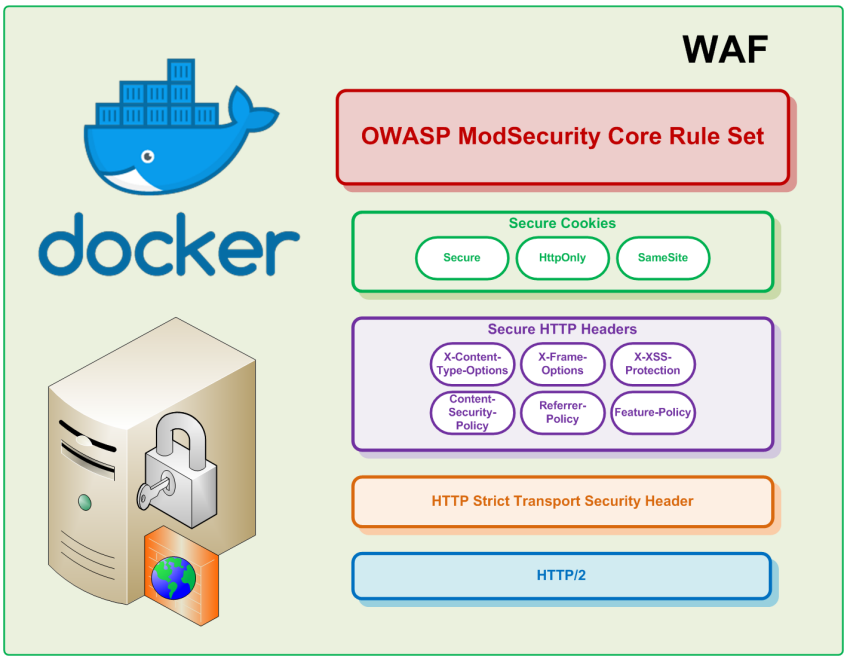


Figura 4.7: Controles de seguridad desplegados en el contenedor Docker.

4.5.4 Software de gestión de certificados HTTPS

Se ha elegido Let's Encrypt para gestionar los certificados, pues permite que la gestión sea automática y no tiene coste económico.

En el diagrama de Let's Encrypt se puede ver cómo se han definido las comunicaciones entre la instancia de Let's Encrypt desplegada en el cluster de Kubernetes y el servicio SaaS ofrecido por el proveedor.

El modo de funcionamiento es el siguiente:

1. La instancia de Docker crea una clave privada.
2. La instancia de Docker realiza una petición de certificado al servicio SaaS.
3. El servicio SaaS le comunica una variable temporal no predecible (tipo NONCE) al cliente Let's Encrypt.

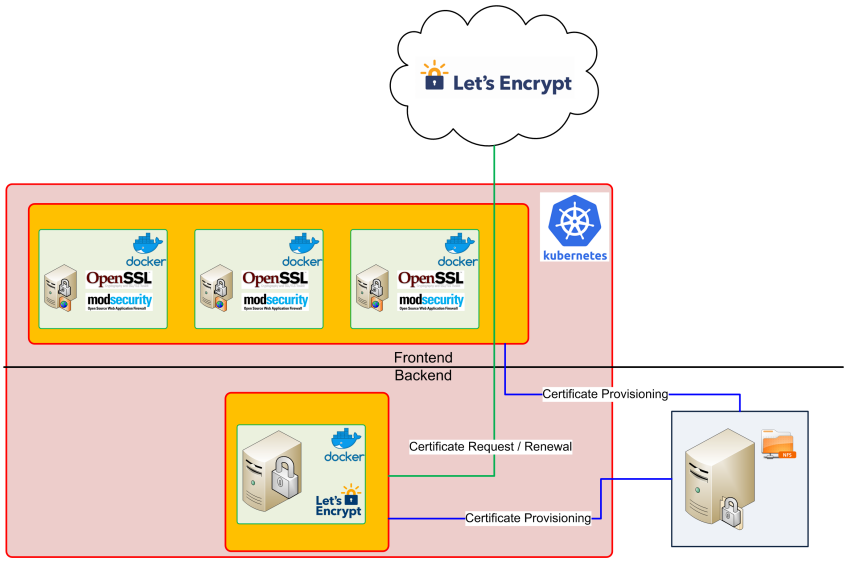


Figura 4.8: Diagrama de comunicaciones de Let's Encrypt.

4. La instancia de Docker crea un fichero en la dirección URL propuesta con el valor propuesto.
5. La instancia Docker responde al servicio SaaS con el fichero temporal firmado con su clave privada.
6. El servicio SaaS realiza una petición HTTP para comprobar que la instancia de Docker tiene control sobre el dominio sobre el que se pide el certificado.

   La taxonomía de la petición es http ://< dominio >:80/. well−known/acme−challenge/<variable−NONCE>.
7. El servicio SaaS valida la respuesta HTTP y, a continuación, valida al agente Let's Encrypt.

Una vez el agente ha sido autorizado, éste podrá pedir un nuevo certificado, renovarlo o revocarlo.

4.5.5 Software de automatización u orquestación

Se ha elegido Kubernetes para gestionar y automatizar la solución.

Para ello, se han definido los siguientes elementos:

- Para el WAF:
  - *ccwaf deployment*. Es el componente encargado de desplegar las instancias de WAF.
  - *ccwaf service*. Es el componente encargado de publicar los PODs desplegados y ofrecer abstracción a los servicios externos.
- Para el servidor web:
  - *basicwebserver deployment*. Es el componente encargado de desplegar las instancias de servicio web.
  - *basicwebserver service*. Es el componente encargado de publicar los PODs desplegados.
- Para el servidor de Let's Encrypt:
  - *basicwebserver deployment*. Es el componente encargado de desplegar la instancia de Let's Encrypt.

– *letsencrypt service*. Es el componente encargado de publicar los PODs desplegados.

A continuación se muestran los aspectos más representativos de estos elementos:

ccwaf deployment

Listing 4.1: ../../CircumCaetera/Kubernetes/kub_waf-server.yaml.template

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kubewaf-deployment
  labels:
    app: kubewaf
    service: kubewaf-service
    tier: frontend
spec:
  replicas: 2
  selector:
    matchLabels:
      app: kubewaf
  template:
    metadata:
      labels:
        app: kubewaf
    spec:
      containers:
      - name: kubewaf
        image: <SET_REGISTRY>/<SET_OWNER>/ccwaf:<SET_RELEASE>
        volumeMounts:
          - name: nfs-volume
            mountPath: /exports
        ports:
        - name: port80
          containerPort: 80
          protocol: TCP
        - name: port443
          containerPort: 443
          protocol: TCP
      imagePullSecrets:
        - name: docscret
      volumes:
        - name: nfs-volume
          nfs:
            server: 10.231.123.166
            path: /
```

Se destacan los siguientes elementos:

replicas : 2. Indica el número de instancias que se despliegan. En las pruebas se han desplegado entre dos y cinco instancias con el fin de validar que la plataforma.

image: <SET_REGISTRY>/<SET_OWNER>/ccwaf:<SET_RELEASE>. Indica el repositorio desde el que se descargarán las imágenes. En la elaboración del proyecto se ha utilizado un repositorio propio, pero se compartirá a un repositorio público tras realizar la entrega.

volumeMounts. Indica el servicio de ficheros que se utilizará para compatir certificados. Este recurso debe ser inmutable y tolerante a reconstrucciones del entorno, por lo que se ha utilizado un servicio NFS externo a la plataforma de Kubernetes.

ccwaf service

Listing 4.2: ../../CircumCaetera/Kubernetes/kub_waf-server.yaml.template

```
apiVersion: v1
kind: Service
metadata:
  name: kubewaf-service
  labels:
    app: kubewaf
    tier: frontend
spec:
  #type: LoadBalancer
  type: NodePort
  selector:
    app: kubewaf
  ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: port80
    nodePort: 30080
  - name: https
    port: 443
    protocol: TCP
    targetPort: port443
    nodePort: 30443
```

En esta configuración destaca el parámetro `type: NodePort`, pues el valor *NodePort* indica que el WAF será accesible externamente desde Internet.

basicwebserver deployment

Listing 4.3: ../../CircumCaetera/Kubernetes/kub_basicwebserver-server.yaml.template

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kubebasicwebserver-deployment
  labels:
    app: kubebasicwebserver
    service: kubebasicwebserver-service
    tier: backend
spec:
  replicas: 3
  selector:
    matchLabels:
      app: kubebasicwebserver
  template:
    metadata:
      labels:
        app: kubebasicwebserver
    spec:
      containers:
      - name: kubebasicwebserver
        image: <SET_REGISTRY>/<SET_OWNER>/basicwebserver:<SET_RELEASE>
        volumeMounts:
          - name: nfs-volume
            mountPath: /exports
        ports:
        - name: port80
          containerPort: 80
          protocol: TCP
        - name: port443
          containerPort: 443
          protocol: TCP
      imagePullSecrets:
        - name: docscret
      volumes:
        - name: nfs-volume
          nfs:
            server: 10.231.123.166
            path: /
```

Se puede ver que este elemento es similar al desplegado para el WAF. En esta ocasión se han desplegado entre dos y cinco instancias en las pruebas con el fin de validar que el entorno distribuido funciona correctamente.

basicwebserver service

Listing 4.4: ../../CircumCaetera/Kubernetes/kub_basicwebserver-server.yaml.template

```
apiVersion: v1
kind: Service
metadata:
  name: kubebasicwebserver-service
  labels:
    app: kubebasicwebserver
    tier: backend
spec:
  type: ClusterIP
  selector:
    app: kubebasicwebserver
  ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: port80
  - name: https
    port: 443
    protocol: TCP
    targetPort: port443
```

Se puede comprobar como en este servicio se ha configurado el siguiente parámetro: `type: ClusterIP`.

El utilizar el valor *ClusterIP* se define que las instancias gestionadas por este servicio sólo son accesibles internamentes al cluster de Kunernetes y, por lo tanto, no son accesibles desde Internet.

Con esto se garantiza que las peticiones que se realicen a la plataforma web deben provenir forzosamente del WAF, con lo que **se garantiza que los mecanismos de protección del WAF serán aplicados**.

basicwebserver deployment

Listing 4.5: ../../CircumCaetera/Kubernetes/kub_basicwebserver-server.yaml.template

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kubebasicwebserver-deployment
  labels:
    app: kubebasicwebserver
    service: kubebasicwebserver-service
    tier: backend
spec:
  replicas: 3
  selector:
    matchLabels:
      app: kubebasicwebserver
  template:
    metadata:
      labels:
        app: kubebasicwebserver
    spec:
      containers:
      - name: kubebasicwebserver
        image: <SET_REGISTRY>/<SET_OWNER>/basicwebserver:<SET_RELEASE>
        volumeMounts:
          - name: nfs-volume
            mountPath: /exports
        ports:
        - name: port80
          containerPort: 80
          protocol: TCP
        - name: port443
          containerPort: 443
          protocol: TCP
      imagePullSecrets:
        - name: docscret
      volumes:
        - name: nfs-volume
          nfs:
            server: 10.231.123.166
            path: /
```

En esta ocasión se ha desplegado una única réplica, pues este servicio está dedicado a la gestión de certificados, un servicio al que los usuarios no tienen acceso y con un consumo menor de recursos.

letsencrypt service

Listing 4.6: ../../CircumCaetera/Kubernetes/kub_basicwebserver-server.yaml.template

```
apiVersion: v1
kind: Service
metadata:
  name: kubebasicwebserver-service
  labels:
    app: kubebasicwebserver
    tier: backend
spec:
  type: ClusterIP
  selector:
    app: kubebasicwebserver
  ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: port80
  - name: https
    port: 443
    protocol: TCP
    targetPort: port443
```

Al igual que en el servicio web, se ha optado por proteger el servicio Let's Encrypt con el WAF, por lo que se ha definido nuevamente `type: ClusterIP`.

4.5.6 Diseño final

Tras construir todos los componentes necesarios, las peticiones HTTP y HTTPS son tal como se indica en el Diagrama de peticiones HTTP/HTTPS en la solución.

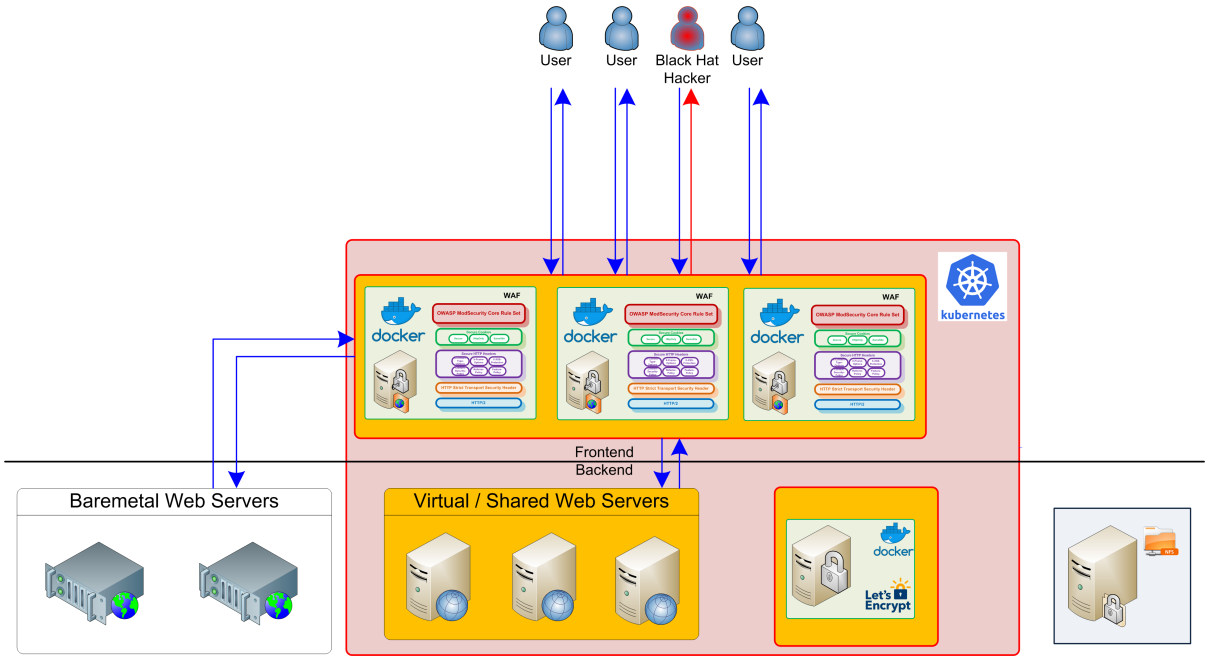


Figura 4.9: Diagrama de peticiones HTTP y HTTPS

Acrónimos

ACME Automatic Certificate Management Environment[49, Estándar ACME]. 23

CA Certification Authority. 23

CDN Content Delivery Network. 7

CI/CD Continuous Integration and Continuous Deployment or Continuous Delivery [53, What is CI/CD]. 25, 26

CSP Content-Security-Policy. 37, 38, 56

DBF Database Firewall. 10

DLP Data loss prevention. 3

DoS Denial-of-service. 7

EV Extended Validation. 7

GPL GNU General Public License[44, Licencia GPL]. 17

HSTS HTTP Strict Transport Security. 31, 34

HTTP Hypertext Transfer Protocol. 3

HTTPS Hypertext Transfer Protocol Secure. 3

LGPL GNU Lesser General Public License[45, Licencia LGPL]. 17

MitM Man-in-the-middle. 34

NIDS Network Intrusion Detection System. 4

OWASP Open Web Application Security Project. 52, 54, 56

PCI DSS Payment Card Industry Data Security Standard. 12

PoC Proof of concept. 14

RBAC role-based access control. 8

RGDP Reglamento General de Protección de Datos. 12

SaaS Software as a Service. 4, 6

SIEM Security information and event management. 17

SNI Server Name Indication[96, Wikipedia]. 7, 18

SQLi Inyección SQL[55]. 26, 37, 50

SSL Secure Sockets Layer. 4

TLS Transport Layer Security. 4

URL Uniform Resource Locator. 7

UTM Unified Threat Management. 3

W3C World Wide Web Consortium. 36

WAF Web Application Firewall. 3, 31, 32

XSRF Cross-site request forgery. 35, 38, 50

XSS Cross-site scripting[54]. 26, 35, 37, 38, 50

Glosario

es una pequeña información enviada por un sitio web y almacenada en el navegador del usuario, de manera que el sitio web puede consultar la actividad previa del navegador [97, Wikipedia]. 38, 50

Inyección SQL[55] es un método de infiltración de código intruso que se vale de una vulnerabilidad informática presente en una aplicación en el nivel de validación de las entradas para realizar operaciones sobre una base de datos [98, Wikipedia]. 37

Cross-site request forgery es un tipo de exploit malicioso de un sitio web en el que comandos no autorizados son transmitidos por un usuario en el cual el sitio web confía. [99, Wikipedia].. 38

Cross-site scripting[54] es un tipo de vulnerabilidad informática o agujero de seguridad típico de las aplicaciones Web, que puede permitir a una tercera persona inyectar en páginas web visitadas por el usuario código JavaScript o en otro lenguaje similar (ej: VBScript). [100, Wikipedia].. 35, 37

Atacante El atacante es un individuo u organización que intenta obtener el control de un sistema informático para utilizarlo con fines maliciosos, robo de información o de hacer daño a su objetivo. [101, Wikipedia]. 18, 20

CA En criptografía, las expresiones autoridad de certificación, o certificadora, o certificante, o las siglas AC o CA (por la denominación en idioma inglés Certification Authority), señalan a una entidad de confianza, responsable de emitir y revocar los certificados, utilizando en ellos la firma electrónica, para lo cual se emplea la criptografía de clave pública. [102, Wikipedia]. 18

Clickjacking es una técnica maliciosa para engañar a usuarios de Internet con el fin de que revelen información confidencial o tomar control de su ordenador cuando hacen clic en páginas web aparentemente inocentes. [103, Wikipedia]. 35

Defensa en Profundidad El concepto de defensa en profundidad se basa en la premisa de que todo componente de un sistema puede ser vulnerado, y por tanto no se debe delegar la seguridad de un sistema en un único método o componente de protección. [104, Wikipedia]. 3

La nube La computación en la nube (del inglés cloud computing), conocida también como servicios en la nube, informática en la nube, nube de cómputo, nube de conceptos o simplemente «la nube», es un paradigma que permite ofrecer servicios de computación a través de una red, que usualmente es Internet. [105, Wikipedia]. 4

Manifiesto por el Desarrollo Ágil de Software Estamos descubriendo formas mejores de desarrollar software tanto por nuestra propia experiencia como ayudando a terceros. A través de este trabajo hemos aprendido a valorar:

Individuos e interacciones sobre procesos y herramientas
Software funcionando sobre documentación extensiva
Colaboración con el cliente sobre negociación contractual
Respuesta ante el cambio sobre seguir un plan

Esto es, aunque valoramos los elementos de la derecha, valoramos más los de la izquierda. [106] 26

MitM es un ataque en el que se adquiere la capacidad de leer, insertar y modificar a voluntad. El atacante debe ser capaz de observar e interceptar mensajes entre las dos víctimas y procurar que ninguna de las víctimas conozca que el enlace entre ellos ha sido violado. [107, Wikipedia].. 34

Phishing denomina un modelo de abuso informático y que se comete mediante el uso de un tipo de ingeniería social, caracterizado por intentar adquirir información confidencial de forma fraudulenta [108, Wikipedia].. 35

Throughput La tasa de transferencia efectiva (en inglés throughput) es el volumen de trabajo o de información neto que fluye a través de un sistema, como puede ser una red de computadoras. [109, Wikipedia]. 7

Bibliografía

[1] Wikipedia. *Unified Threat Management*
. URL: https://es.wikipedia.org/wiki/Unified_Threat_Management.

[2] Wikipedia. *Modelo TCP/IP*
. URL: https://es.wikipedia.org/wiki/Modelo_TCP/IP.

[3] *Estándar del modelo OSI, del inglés* Open Systems Interconnection model. *ISO/IEC standard 7498-1:1994*
. URL: http://standards.iso.org/ittf/PubliclyAvailableStandards/s020269_ISO_IEC_7498-1_1994(E).zip.

[4] Wikipedia. *Transport Layer Security*
. URL: https://es.wikipedia.org/wiki/Transport_Layer_Security.

[5] S.L. MR Informatica. *Algunas notas sobre Cloud Computing*
. URL: https://mrinformatica.es/algunas-notas-sobre-cloud-computing/.

[6] *¿Qué es el software libre?*
. URL: https://www.gnu.org/philosophy/free-sw.es.html.

[7] Open Web Application Security Project (OWASP. *OWASP Best Practices: Use of Web Application Firewalls*
. URL: https://www.owasp.org/index.php/Category:OWASP_Best_Practices:_Use_of_Web_Application_Firewalls.

[8] *Software propietario o privativo*
. URL: https://www.definicionabc.com/tecnologia/software-propietario.php.

[9] *Cloudflare WAF*
. URL: https://www.cloudflare.com/waf/.

[10] *Cloudflare*
. URL: https://www.cloudflare.com/.

[11] *Kona WAF*
. URL: https://www.akamai.com/uk/en/resources/waf.jsp.

[12] *Akamai*
. URL: https://www.akamai.com/es/es/.

[13] *Incapsula Web Application Firewall*
. URL: https://www.incapsula.com/website-security/web-application-firewall.html.

[14] *CDN Cost Comparison - Global Traffic Monthly Plans*
. URL: https://www.cdn77.com/compare-cdn-providers.

[15] *Imperva Incapsula. Pricing and plans*
. URL: https://www.incapsula.com/pricing-and-plans.html.

[16] *Imperva WAF Gateway*
. URL: https://www.imperva.com/products/on-premises-waf/.

[17] *FortiWeb: Web Application Firewall*
. URL: https://www.fortinet.com/products/web-application-firewall/fortiweb.html.

[18] *Fortinet*
. URL: https://www.fortinet.com/.

[19] *Precio de appliance Imperva, modelo X2020*
. URL: https://www.comparitech.com/net-admin/best-web-application-firewall/.

[20] *Precio de appliance Imperva, modelo X2010*
. URL: https://searchsecurity.techtarget.com/feature/Comparing-the-best-Web-application-firewalls-in-the-industry.

[21] *Precio de licencias anuales de Imperva, modelo X2010*
. URL: https://www.globenetstore.com/shop/search.aspx?search=SS-WAF-X21-SL.

[22] *Precio de licencias anuales de Imperva, modelos X2500 y X4500 (fichero de Microsoft Excel)*
. URL: https://cdn2.hubspot.net/hubfs/2539908/Imperva%20Price%20List.xlsx.

[23] *SecureSphere WAF AV1000 Gateway for AWS*
. URL: https://aws.amazon.com/marketplace/pp/B00UAWMZ1U?qid=1555322432672.

[24] *SecureSphere WAF AV2500 Gateway for AWS*
. URL: https://aws.amazon.com/marketplace/pp/B00UAWN0FU?qid=1555323193972.

[25] *AVFirewalls.com. Fortinet FortiWeb 100D*
. URL: http://www.avfirewalls.com/FortiWeb-100D.asp.

[26] *AVFirewalls.com. Fortinet FortiWeb 400D*
. URL: http://www.avfirewalls.com/FortiWeb-400D.asp.

[27] *Real Data Solutions. Fortinet FortiWeb 600D*
. URL: http://realdatasolutions.net/index.php?id_product=220&controller=product.

[28] *AWS Marketplace. Fortinet FortiWeb Web Application Firewall WAF VM*
. URL: https://aws.amazon.com/marketplace/pp/B00L9JODAE?ref=_ptnr_ftnt_web_fortiweb.

[29] *Página oficial de IronBee*
. URL: https://github.com/ironbee/ironbee.

[30] *Repositorio de código oficial de WebCastellum*
. URL: https://sourceforge.net/p/webcastellum/code/HEAD/tree/.

[31] *Repositorio de código oficial de Raptor WAF*
. URL: https://github.com/CoolerVoid/raptor_waf.

[32] *Página oficial de NAXSI*
. URL: https://github.com/nbs-system/naxsi.

[33] *Página oficial de OpenWAF*
. URL: https://github.com/titansec/OpenWAF.

[34] *Blog oficial de FreeWAF / lua-resty-waf*
. URL: https://www.cryptobells.com/reintroducing-lua-resty-waf/.

[35] *Página oficial de Shadow Daemon*
. URL: https://shadowd.zecure.org/overview/introduction/.

[36] *Página oficial de AQTRONiX WebKnight*
. URL: https://www.aqtronix.com/?PageID=99.

[37] *Página oficial de Vulture WAF*
. URL: https://www.vultureproject.org/.

[38] *Página oficial de Modsecurity*
. URL: https://www.modsecurity.org/.

[39] *Página oficial de Trustware*
. URL: https://www.trustwave.com/en-us/.

[40] *Modalidades de soporte de Modsecurity*
. URL: https://www.modsecurity.org/help.html.

[41] Apache Software Foundation. *Página oficial del servidor web Apache HTTP Server*
. URL: https://httpd.apache.org/.

[42] *Página oficial del servidor web Nginx*
. URL: http://nginx.org/.

[43] *Página oficial del servidor web Microsoft Internet Information Services*
. URL: https://www.iis.net/.

[44] *GNU General Public License*
. URL: https://www.gnu.org/licenses/gpl-3.0.html.

[45] *GNU Lesser General Public License*
. URL: https://www.gnu.org/licenses/lgpl-3.0.html.

[46] *SSL and TLS Deployment Best Practices*
. URL: https://github.com/ssllabs/research/wiki/SSL-and-TLS-Deployment-Best-Practices.

[47] Wikipedia. *Certificate signing request*
. URL: https://en.wikipedia.org/wiki/Certificate_signing_request.

[48] Internet Engineering Task Force (IETF). *Estándar RFC5280. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*
. URL: https://tools.ietf.org/rfc/rfc5280.txt.

[49] Internet Engineering Task Force (IETF). *Estándar RFC8555. Automatic Certificate Management Environment (ACME)*
. URL: https://tools.ietf.org/html/rfc8555.

[50] *Página oficial de Let's Encrypt*
. URL: https://letsencrypt.org/.

[51] Wikipedia. *Desarrollo ágil de software*
. URL: https://es.wikipedia.org/wiki/Desarrollo_%C3%A1gil_de_software.

[52] Inc. Docker. *Página oficial de Docker*
. URL: https://www.docker.com/.

[53] *What is CI/CD - all you need to know*
. URL: https://codilime.com/what-is-ci-cd-all-you-need-to-know/.

[54] Open Web Application Security Project (OWASP. *Artículo en OWASP de ataques CSS*
. URL: https://www.owasp.org/index.php/Cross-site_Scripting_%28XSS%29.

[55] Open Web Application Security Project (OWASP. *Artículo en OWASP de ataques de inyección SQL*
. URL: https://www.owasp.org/index.php/SQL_Injection.

[56] *Exploring Naxsi (A Bit)*
. URL: https://www.cryptobells.com/exploring-naxsi-a-bit/.

[57] *FreeWAF changelog*
. URL: https://github.com/p0pr0ck5/lua-resty-waf/releases.

[58] proyecto FreeBSD. *Licencia BSD simplificada o licencia FreeBSD (de 2 cláusulas)*
. URL: https://es.wikipedia.org/wiki/Licencia_BSD#Licencia_BSD_simplificada_o_licencia_FreeBSD_(de_2_cl%C3%A1usulas).

[59] *TLS compatibility with PCI DSS (Payment Card Industry Data Security Standard)*
. URL: https://blog.wao.io/tls-compatibility-with-pci-dss/.

[60] OpenBSD project. *Página oficial del proyecto LibreSSL*
. URL: https://www.libressl.org/.

[61] *Página oficial del proyecto GnuTLS*
. URL: https://www.gnutls.org/.

[62] *Página oficial del proyecto OpenSSL*
. URL: https://www.openssl.org/.

[63] Mozilla. *Página oficial del proyecto Network Security Services (NSS)*
. URL: https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS.

[64] *Página web del módulo SSL de Nginx*
. URL: http://nginx.org/en/docs/http/ngx_http_ssl_module.html.

[65] Apache Software Foundation. *Apache License 1.0*
. URL: https://www.apache.org/licenses/LICENSE-1.0.

[66] SSLeay. *SSLeay license*
. URL: https://www.openssl.org/source/license-openssl-ssleay.txt.

[67] Wikipedia. *Página oficial del proyecto LibreSSL*
. URL: https://en.wikipedia.org/wiki/GnuTLS.

[68] Wikipedia. *OpenSSL - Apartado* Licensing
. URL: https://en.wikipedia.org/wiki/OpenSSL#Licensing.

[69] OpenSSL Blog. *License Agreements and Changes Are Coming*
. URL: https://www.openssl.org/blog/blog/2015/08/01/cla/.

[70] Nginx GitHub repository. *Pull request del cambio de licencia en Nginx*
. URL: https://www.openssl.org/blog/blog/2015/08/01/cla/.

[71] Google. *Página oficial de Kubernetes (K8s)*
. URL: https://kubernetes.io/.

[72] Apache Software Foundation. *Página oficial de Apache Mesos)*
. URL: http://mesos.apache.org/.

[73] Inc. Docker. *Página oficial de Docker Swarm)*
. URL: https://github.com/docker/swarm.

[74] Platform9 Blog. *Comparativa de popularidad Kubernetes y Docker Swarm*
. URL: https://platform9.com/blog/kubernetes-docker-swarm-compared/.

[75] Platform9 Blog. *Comparativa de popularidad Kubernetes y Apache Mesos*
. URL: https://platform9.com/blog/kubernetes-vs-mesos-marathon/.

[76] Google Trends. *Comparativa de popularidad Kubernetes, Apache Mesos y Docker Swarm*
. URL: https://trends.google.com/trends/explore?date=today%205-y&q=docker%20swarm,kubernetes,apache%20mesos.

[77] Microsoft Azure. *Página oficial de Azure Kubernetes Service (AKS)*
. URL: https://docs.microsoft.com/en-us/azure/aks/.

[78] Amazon. *Página oficial de Amazon Elastic Kubernetes Service (Amazon EKS)*
. URL: https://aws.amazon.com/es/eks/.

[79] Google. *Página oficial de Kubernetes Ingress Controllers*
. URL: https://kubernetes.io/docs/concepts/services-networking/ingress-controllers/.

[80] *Página oficial de Samba*
. URL: https://www.samba.org/.

[81] Internet Engineering Task Force (IETF). *Estándar RCF7530. Network File System (NFS) Version 4 Protocol*
. URL: https://tools.ietf.org/html/rfc7530.

[82] Amazon. *Página oficial de Amazon S3*
. URL: https://aws.amazon.com/s3/.

[83] Google. *Página oficial de Google Storage*
. URL: https://cloud.google.com/storage/.

[84] Open Web Application Security Project (OWASP. *OWASP ModSecurity Core Rule Set (CRS)*
. URL: https://www.owasp.org/index.php/Category:OWASP_ModSecurity_Core_Rule_Set_Project.

[85] Wikipedia. *HTTP Strict Transport Security*
. URL: https://es.wikipedia.org/wiki/HTTP_Strict_Transport_Security.

[86] Mozilla. *X-Content-Type-Options*
. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options.

[87] Mozilla. *MIME sniffing*
. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types#MIME_sniffing.

[88] WP Doctor. *Cabecera X-Frame-Options para mejorar la seguridad de tu web*
. URL: https://www.wpdoctor.es/cabecera-x-frame-options-para-mejorar-la-seguridad-de-tu-web/.

[89] Mozilla. *X-XSS-Protection*
. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-XSS-Protection.

[90] Mozilla. *Referrer-Policy*
. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Referrer-Policy.

[91] Mozilla. *Feature Policy*
. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Feature-Policy.

[92] World Wide Web Consortium. *Feature Policy. W3C First Public Working Draft*
. URL: https://www.w3.org/TR/2019/WD-feature-policy-1-20190416/.

[93] Mozilla. *Content-Security-Policy (CSP)*
. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP.

[94] AirisX. *The Nginx module for adding cookie flag*
. URL: https://github.com/AirisX/nginx_cookie_flag_module.

[95] Docker Hub. *Debian. Docker Official Images*
. URL: https://hub.docker.com/_/debian/.

[96] Wikipedia. *Server Name Indication*
. URL: https://es.wikipedia.org/wiki/Server_Name_Indication.

[97] Wikipedia. *Cookie (informática)*
. URL: https://es.wikipedia.org/wiki/Cookie_(inform%C3%A1tica).

[98] Wikipedia. *Inyección SQL*
. URL: https://es.wikipedia.org/wiki/Inyecci%C3%B3n_SQL.

[99] Wikipedia. *Cross-site request forgery*
. URL: https://es.wikipedia.org/wiki/Cross-site_request_forgery.

[100] Wikipedia. *Cross-site scripting*
. URL: https://es.wikipedia.org/wiki/Cross-site_scripting.

[101] Wikipedia. *Ciberataque*
. URL: https://es.wikipedia.org/wiki/Ciberataque.

[102] Wikipedia. *Autoridad de certificación*
. URL: https://es.wikipedia.org/wiki/Autoridad_de_certificaci%C3%B3n.

[103] Wikipedia. *Clickjacking*
. URL: https://es.wikipedia.org/wiki/Clickjacking.

[104] Wikipedia. *Defensa en profundidad en seguridad informática*
. URL: https://es.wikipedia.org/wiki/Defensa_en_profundidad#Defensa_en_profundidad_en_seguridad_inform%C3%A1tica.

[105] Wikipedia. *Computación en la nube*
. URL: https://es.wikipedia.org/wiki/Computaci%C3%B3n_en_la_nube.

[106] *Manifiesto por el Desarrollo Ágil de Software*
. URL: http://agilemanifesto.org/iso/es/manifesto.html.

[107] Wikipedia. *Man-in-the-middle attack*
. URL: https://en.wikipedia.org/wiki/Man-in-the-middle_attack.

[108] Wikipedia. *Phishing*
. URL: https://es.wikipedia.org/wiki/Phishing.

[109] Wikipedia. *Throughput*
. URL: https://es.wikipedia.org/wiki/Throughput.

Anexos

Anexo A

Contrucción de WAF en Docker

../../CircumCaetera/Docker/ccwaf/Dockerfile.template

```
FROM debian:buster-slim

ARG USER
ENV USER ${USER:-waf}

ENV USER_HOME       "/home/${USER}"

ENV NGINX_VERSION 1.16.0
ENV nginxPath       "/usr/local/nginx"
ENV nginxConfFile "$nginxPath/conf/nginx.conf"
ENV nginxConfDir  "$nginxPath/conf"
ENV nginxExecPath "${nginxPath}/sbin"
ENV ModSecPath      "$nginxPath/conf/modsec"
ENV certsPath       "$nginxPath/certs"
ENV nginxlogsPath "$nginxPath/logs"
ENV indexFile       "/usr/local/nginx/html/index.html"

ENV DOMAINS         "<SET_DOMAIN>"

COPY ./files/build/ /

RUN set -eu \
    && echo 'debconf debconf/frontend select Noninteractive' | debconf-set-selections \
    && DEBIAN_FRONTEND=noninteractive \
    && apt-get update \
    && apt-get install -y --no-install-recommends apt-utils \
    && apt-get install -y   git \
                            iputils-ping \
                            iproute2 \
                            inetutils-traceroute \
                            dnsutils \
                            tcptrace \
                            tcpdump \
                            less \
                            net-tools \
                            openssl \
                            openssh-server \
                            nfs-common \
                            screen \
                            sudo \
                            vim \
                            wget \
                            # Packages for nginx & modSecurity
                            autoconf \
                            automake \
                            build-essential \
                            bison \
                            curl \
                            dh-autoreconf \
                            doxygen \
                            flex \
```

```
                            g++ \
                            libcurl4-gnutls-dev \
                            libgeoip-dev \
                            libpcre++-dev \
                            libssl-dev \
                            libtool \
                            libxml2 \
                            libyajl-dev \
                            libxml2-dev \
                            liblmdb-dev \
                            pkgconf \
                            zlib1g-dev \
    && rm -rf /var/lib/apt/lists/* /tmp/* \
    && DEBIAN_FRONTEND=dialog \
    && mkdir -p /run/sshd \
    && useradd ${USER} -d ${USER_HOME} -m --shell /bin/bash -G sudo \
    && chown -R ${USER}:${USER} ${USER_HOME} \
    && echo "root:Docker!" | chpasswd \
    && echo "waf:waf!" | chpasswd \
    # Install modSecurity
    && cd /opt/ \
    && git clone https://github.com/SpiderLabs/ModSecurity \
    && cd ModSecurity/ \
    && git checkout -B v3/master origin/v3/master \
    && sh build.sh \
    && git submodule init \
    && git submodule update \
    && ./configure \
    && make \
    && make install \
    && chmod 700 /run.sh

# Nginx
COPY ./files/nginx_conf/ ${nginxConfDir}/

RUN set -eu \
    && chmod 700 /build_nginx.sh \
    && /build_nginx.sh

# modSecurity
RUN set -eu \
    && chmod 700 /build_modsecurity.sh \
    && /build_modsecurity.sh

ENV PATH ${nginxExecPath}:$PATH


EXPOSE 22
EXPOSE 80
EXPOSE 443

ENTRYPOINT /run.sh
```

../../CircumCaetera/Docker/ccwaf/files/nginx_conf/nginx.conf

```
user www-data;
worker_processes auto;

error_log  logs/error.log  info;
# Sending logs to standard error so we can see them from docker logs or kubernets logs
error_log  /dev/stderr     info;

pid /run/nginx.pid;

events {
  worker_connections  1024;
}

http {
  #log
  log_format regular '$host $remote_addr - "$http_x_forwarded_for" $remote_user [$time_local]
      "$request" $status $body_bytes_sent "$http_referer" "$http_user_agent" $request_time';
  log_format front_verb    '$host $remote_addr - "$http_x_forwarded_for" $remote_user [
```

```
      $time_local] "$request" $status $body_bytes_sent "$http_referer" "$http_user_agent"
      $request_time'
    '$request_length $bytes_sent "$http_authorization" "$http_x_forwarded_proto" $server_name'
      ;

  log_format brief '$remote_addr "$request" $status';

  ## SSL (regular chunck to be added)
  log_format SSLclient '"$ssl_client_s_dn($ssl_client_serial)"';

  ## Upstream logs (regular chunck to be added)
  log_format upstreamlog '$msec $server_name $upstream_addr $upstream_response_time';

  ## Backend log
  log_format backend  '    $remote_addr - "$http_x_real_ip"       $remote_user [$time_local] "
      $request" $status $body_bytes_sent "$http_referer" "$http_user_agent"';

  log_format back_verb '    $remote_addr - "$http_x_real_ip"       $remote_user [$time_local]
      "$request" $status $body_bytes_sent "$http_referer" "$http_user_agent" $request_time'
    '$request_length $bytes_sent "$http_authorization" "$http_x_forwarded_proto" $server_name'
      ;


  access_log logs/$host.access.log;
  access_log logs/$host.access_verbose.log front_verb;
  # Sending logs to standard output so we can see them from docker logs or kubernets logs
  access_log /dev/stdout front_verb;

  # Map upgrade and connection HTTP headers in case we use WebSockets
  # Check location example in default.conf
  map $http_upgrade $connection_upgrade {
    default upgrade;
    ''      close;
  }

  include       mime.types;
  default_type  application/octet-stream;

  sendfile        on;

  keepalive_timeout  65;

  # Reverse Proxy headers
  include /usr/local/nginx/conf/snippets/proxy_headers.conf;

  # Hide Server and other common identification headers
  include /usr/local/nginx/conf/snippets/hide_headers.conf;

  # HTTP protocol version for proxying
  proxy_http_version 1.1;

  # Disabling buffering (needed for real-time interactions, e.g. Node.js). Need testing.
  proxy_buffering off;

  # Adding resolver directive so proxy_pass name is updated automatically
  # It needs to be replaced by Kunernetes DNS service IP.
  # Script kube_launchNewContainer.sh should take care of it.
  # Or, if you are not using the script, just can get the IP executing the following command:
  # kubectl describe svc -lk8s-app=kube-dns -n kube-system | sed -re '/IP:/!d;s/.* //')"
  #
  #resolver <KUBEDNS_SERVICE_IP>;
  resolver 10.96.0.10;

  # HTTP/HTTPS sites
  include /usr/local/nginx/conf/sites-enabled/*.conf;
}
```

../../CircumCaetera/Docker/ccwaf/files/nginx_conf/sites-available/default.conf.template

```
server {
  listen        80 default_server;
  listen        [::]:80 default_server;
  server_name <SET_DOMAIN> *.<SET_DOMAIN>;
```

```
5
6     # It needs to be replaced by Let's Encrypts service IP.
7     # Script kube_launchNewContainer.sh should take care of it.
8     # Or, if you are not using the script, just can get the IP executing the following command:
9     # kubectl describe svc -lapp=kubeletsencrypt  | sed -re '/IP:/!d;s/.* //'
10    #
11    #set $upstream_endpoint http://<KUBELETSENCRYPT_SERVICE_IP>;
12    set $upstream_endpoint http://10.105.221.243;
13
14    location = / {
15      # Redirect plain text traffic to HTTPS
16      return 301 https://$server_name$request_uri;
17    }
18
19    location ^~ /.well-known/acme-challenge/ {
20        proxy_pass $upstream_endpoint$request_uri;
21    }
22
23    location = /.well-known/acme-challenge/ {
24      return 404;
25    }
26  }
27
28  server {
29    listen        443 ssl http2 default_server;
30    listen        [::]:443 ssl http2 default_server;
31    server_name <SET_DOMAIN> *.<SET_DOMAIN>;
32
33    # Security headers
34    include /usr/local/nginx/conf/snippets/security_headers.conf;
35
36    # Securying Cookies
37    set_cookie_flag * HttpOnly secure SameSite=Lax;
38
39    # It needs to be replaced by the upstream service IP (the web server behind the WAF).
40    # Script kube_launchNewContainer.sh should take care of it.
41    # Or, if you are not using the script, just can get the IP executing the following command:
42    # kubectl describe svc -lapp=kubebasicwebserver  | sed -re '/IP:/!d;s/.* //'
43    #
44    #set $upstream_endpoint http://<KUBEWAF_SERVICE_SERVICE_HOST>;
45    set $upstream_endpoint https://10.99.145.254;
46
47    ssl_certificate      /usr/local/nginx/certs/testCert.pem;
48    ssl_certificate_key  /usr/local/nginx/certs/testCert.key;
49
50    ssl_session_cache    shared:SSL:1m;
51    ssl_session_timeout  5m;
52
53    ssl_protocols TLSv1.1 TLSv1.2 TLSv1.3;
54    ssl_ciphers  HIGH:!aNULL:!MD5;
55    ssl_prefer_server_ciphers  on;
56
57    location / {
58      proxy_pass $upstream_endpoint$request_uri;
59    }
60
61    ###################################
62    ############ WebSockets ############
63    #
64    location /websocket {
65      proxy_pass $upstream_endpoint$request_uri;
66      proxy_set_header Upgrade $http_upgrade;
67      proxy_set_header Connection "upgrade";
68      include /usr/local/nginx/conf/snippets/proxy_headers.conf;
69    ###################################
70    }
71  }
```

../../CircumCaetera/Docker/ccwaf/files/nginx_conf/snippets/hide_headers.conf

```
1  server_tokens off;
2  proxy_hide_header       X-Powered-By;
3  proxy_hide_header       X-Runtime;
```

```
4 proxy_hide_header        Server;
```

../../CircumCaetera/Docker/ccwaf/files/nginx_conf/snippets/proxy_headers.conf

```
1 proxy_set_header Host $host;
2 proxy_set_header X-Real-IP $remote_addr;
3 proxy_set_header X-Forwarded-For   $proxy_add_x_forwarded_for;
4 proxy_set_header X-Forwarded-Proto $scheme;
5 proxy_set_header X-Forwarded-Host  $host;
6 proxy_set_header X-Forwarded-Port  $server_port;
```

../../CircumCaetera/Docker/ccwaf/files/nginx_conf/snippets/security_headers.conf

```
1  # Remove headers sent by upstream server.
2  proxy_hide_header Strict-Transport-Security;
3  proxy_hide_header X-Content-Type-Options;
4  proxy_hide_header X-Frame-Options;
5  proxy_hide_header X-XSS-Protection;
6
7
8  # Alternative in case we need preserving upstream header:
9  # Source: https://serverfault.com/a/598106/530594
10
11 #map $upstream_http_strict_transport_security $sts {
12 #   '' "max-age=31536000; includeSubDomains" always;
13 #}
14 #
15 #add_header Strict-Transport-Security $sts;
16
17
18 #Set up HSTS
19 add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;
20
21 # Additional security headers
22 add_header X-Content-Type-Options "nosniff" always;
23 add_header X-Frame-Options SAMEORIGIN always;
24 add_header X-XSS-Protection "1; mode=block" always;
25
26
27 # Content-Security-Policy. It is quite open but should be tailored based on the web platform.
28 # References:
29 # https://github.com/OWASP/CheatSheetSeries/blob/e791059427dbaab6a21f642ff2fb5c27de5d4c43/
       cheatsheets/Content_Security_Policy_Cheat_Sheet.md
30 # https://content-security-policy.com/
31 # https://www.html5rocks.com/en/tutorials/security/content-security-policy/
32 #
33 # Online CSP validators:
34 # https://csp-evaluator.withgoogle.com/
35 # https://cspvalidator.org/
36 #
37 # Side note: We use variable nesting for clarity since all the parameters must be in one line.
38 # Kudos to https://serverfault.com/a/854238/530594
39
40 set $defaultSrcDomains '';
41 set $defaultSrcDomains "${defaultSrcDomains} https://google.com";
42 set $defaultSrcDomains "${defaultSrcDomains} https://youtube.com";
43 set $defaultSrcDomains "${defaultSrcDomains} https://facebook.com";
44 set $defaultSrcDomains "${defaultSrcDomains} https://fonts.google.com";
45 set $defaultSrcDomains "${defaultSrcDomains} https://fonts.googleapis.com";
46 set $defaultSrcDomains "${defaultSrcDomains} https://ajax.googleapis.com";
47 set $defaultSrcDomains "${defaultSrcDomains} https://www.google-analytics.com";
48 set $defaultSrcDomains "${defaultSrcDomains} https://cdnjs.cloudflare.com";
49 set $defaultSrcDomains "${defaultSrcDomains} https://code.jquery.com";
50 set $defaultSrcDomains "${defaultSrcDomains} https://connect.facebook.net";
51 set $defaultSrcDomains "${defaultSrcDomains} https://imgur.com";
52 set $defaultSrcDomains "${defaultSrcDomains} https://i.imgur.com";
53 set $defaultSrcDomains "${defaultSrcDomains} https://s.imgur.com";
54 set $defaultSrcDomains "${defaultSrcDomains} https://500px.com";
55 set $defaultSrcDomains "${defaultSrcDomains} https://drscdn.500px.org";
56 set $defaultSrcDomains "${defaultSrcDomains} https://www.reddit.com";
57 set $defaultSrcDomains "${defaultSrcDomains} https://www.flickr.com";
58 set $defaultSrcDomains "${defaultSrcDomains} https://c1.staticflickr.com";
```

```
59
60  set $ImageSrcDomains '';
61  set $ImageSrcDomains "${ImageSrcDomains} https://ssl.google-analytics.com";
62  set $ImageSrcDomains "${ImageSrcDomains} https://s-static.ak.facebook.com";
63
64  set $frameSrcDomains '';
65  set $frameSrcDomains "${frameSrcDomains} https://www.facebook.com";
66  set $frameSrcDomains "${frameSrcDomains} https://s-static.ak.facebook.com";
67
68  set $scriptSrcDomains '';
69  set $scriptSrcDomains "${scriptSrcDomains} https://ssl.google-analytics.com";
70  set $scriptSrcDomains "${scriptSrcDomains} https://connect.facebook.net";
71
72  # script-src note:
73    # 'unsafe-inline' should be removed but we would need to add all the authorized scripts and
        their hashes in order to do so.
74    # Example of hashes taken from the script defined in index.html (taken from Google Chrome
        Browser, Web Developer -> Console):
75    # sha256-tal7GbviAXdYCAc6NunaQupSR3mKRqGXRcvPRMdJb0o and sha256-/DXjfYUUJWnYrCRgtOE0sbgYb+
        loFr5i0G0HlISdPrU
76
77  set $CSP '';
78  set $CSP "${CSP}default-src      'self' $defaultSrcDomains";
79  set $CSP "${CSP}object-src       'none';";
80  set $CSP "${CSP}frame-src        $frameSrcDomains;";
81  set $CSP "${CSP}img-src          data: $ImageSrcDomains;";
82  set $CSP "${CSP}script-src       $scriptSrcDomains 'unsafe-inline';";
83  #set $CSP "${CSP}script-src      $scriptSrcDomains 'sha256-
      tal7GbviAXdYCAc6NunaQupSR3mKRqGXRcvPRMdJb0o=';";
84  #set $CSP "${CSP}style-src        https://fonts.googleapis.com 'sha256-/
      DXjfYUUJWnYrCRgtOE0sbgYb+loFr5i0G0HlISdPrU=';";
85  set $CSP "${CSP}style-src        https://fonts.googleapis.com;";
86  set $CSP "${CSP}font-src         https://themes.googleusercontent.com;";
87
88  add_header Content-Security-Policy "$CSP" always;
89
90
91  # Referer-Policy
92  # References:
93  # https://www.owasp.org/index.php/OWASP_Secure_Headers_Project#rp
94  # strict-origin-when-cross-origin means: Send a full URL when performing a same-origin request
      , only send the origin of the document to a-priori as-much-secure destination (HTTPS->
      HTTPS), and send no header to a less secure destination (HTTPS->HTTP).
95
96  add_header Referrer-Policy "strict-origin-when-cross-origin";
97
98
99  # Feature-Policy:
100 # References:
101 # https://github.com/w3c/webappsec-feature-policy/blob/master/features.md
102 # https://developer.mozilla.org/en-US/docs/Web/HTTP/Feature_Policy#Browser_compatibility
103
104 set $FeaturePolicy '';
105 set $FeaturePolicy "${FeaturePolicy}geolocation            'self';";
106 set $FeaturePolicy "${FeaturePolicy}sync-xhr               'self';";
107 set $FeaturePolicy "${FeaturePolicy}payment                'self';";
108 set $FeaturePolicy "${FeaturePolicy}camera                 'none';";
109 set $FeaturePolicy "${FeaturePolicy}usb                    'none';";
110 set $FeaturePolicy "${FeaturePolicy}magnetometer           'none';";
111 set $FeaturePolicy "${FeaturePolicy}accelerometer          'self';";
112 set $FeaturePolicy "${FeaturePolicy}gyroscope              'self';";
113 set $FeaturePolicy "${FeaturePolicy}speaker                'self';";
114 set $FeaturePolicy "${FeaturePolicy}ambient-light-sensor   'none';";
115 set $FeaturePolicy "${FeaturePolicy}microphone             'none'";
116
117 add_header Feature-Policy "${FeaturePolicy}";
```