

CCN-CERT BP/07

Recomendaciones implementación HTTPS



Noviembre 2017



LIMITACIÓN DE RESPONSABILIDAD

El presente documento se proporciona de acuerdo con los términos en él recogidos, rechazando expresamente cualquier tipo de garantía implícita que se pueda encontrar relacionada. En ningún caso, el Centro Criptológico Nacional puede ser considerado responsable del daño directo, indirecto, fortuito o extraordinario derivado de la utilización de la información y software que se indican incluso cuando se advierta de tal posibilidad¹.

AVISO LEGAL

Quedan rigurosamente prohibidas, sin la autorización escrita del Centro Criptológico Nacional, bajo las sanciones establecidas en las leyes, la reproducción parcial o total de este documento por cualquier medio o procedimiento, comprendidos la reprografía y el tratamiento informático, y la distribución de ejemplares del mismo mediante alquiler o préstamo públicos.

¹ Raúl Siles, fundador y analista de seguridad de DinoSec, ha participado en la elaboración y modificación del presente documento y sus anexos.

ÍNDICE

1. SOBRE CCN-CERT.....	5
2. RESUMEN EJECUTIVO	5
3. BUENAS PRÁCTICAS Y RECOMENDACIONES PARA LA IMPLEMENTACIÓN DE HTTPS ...	8
3.1 Acceso a los puertos TCP/IP asociados a los servicios web.....	9
3.1.1 Recomendaciones de acceso a los puertos TCP/IP de los servicios web	10
3.2 Generación de las claves criptográficas	11
3.2.1 Recomendaciones para la generación de las claves criptográficas	13
3.3 Gestión de los certificados digitales.....	13
3.3.1 Tipos de certificados digitales.....	14
3.3.2 Emisión de certificados digitales: Autoridades Certificadoras (CAs).....	14
3.3.3 Entidad asociada al certificado digital	16
3.3.4 Validez del certificado digital	17
3.3.5 Renovación de los certificados digitales	18
3.3.6 Certificate Transparency (CT).....	19
3.3.7 CAA (Certification Authority Authorization)	21
3.3.8 Restricción de los certificados digitales legítimos: HPKP	22
3.3.9 Recomendaciones para la gestión de los certificados digitales	27
3.4 Protocolos SSL y TLS.....	28
3.4.1 Versiones del protocolo SSL y TLS	28
3.4.2 SNI (Server Name Indication)	29
3.4.3 Renegociación	30
3.4.4 Degradación de la versión del protocolo TLS.....	31
3.4.5 HTTP/2	31
3.4.6 Recomendaciones de los protocolos SSL y TLS	31
3.5 Algoritmos y <i>suites</i> criptográficas	32
3.5.1 Forward secrecy	32
3.5.2 Intercambio de claves robusto.....	32
3.5.3 Fortaleza de los parámetros Diffie-Hellman (DH)	34
3.5.4 Preferencia de las <i>suites</i> criptográficas.....	35
3.5.5 Fortaleza de los algoritmos y <i>suites</i> criptográficas	35
3.5.6 Recomendaciones de los algoritmos y <i>suites</i> criptográficas.....	39
3.6 Mecanismos de revocación de certificados digitales	40
3.6.1 Certificate Revocation Lists (CRLs).....	40

3.6.2	Online Certificate Status Protocol (OCSP)	41
3.6.3	OCSP Stapling y OCSP Must-Staple.....	42
3.6.4	Recomendaciones para la revocación de certificados digitales	44
3.7	Contenidos y aplicaciones web HTTPS.....	44
3.7.1	Prioridad en los buscadores web.....	44
3.7.2	Rendimiento	45
3.7.2.1	TLS False Start.....	46
3.7.2.2	TLS session resumption o caching, y TLS session tickets	47
3.7.2.3	Preconnect	48
3.7.3	Forzando el uso (por defecto) de HTTPS	48
3.7.4	Forzando el uso estricto (por defecto) de HTTPS: HSTS.....	48
3.7.5	Evitando contenidos mixtos HTTP y HTTPS	50
3.7.6	Content Security Policy (CSP)	51
3.7.7	Ausencia de funcionalidades avanzadas en entornos HTTP	55
3.7.8	Evitando cachear contenidos sensibles	55
3.7.9	Inspección del tráfico HTTPS	56
3.7.10	Cookies seguras.....	57
3.7.11	Recomendaciones para los contenidos y aplicaciones web HTTPS.....	58
3.8	Vulnerabilidades en HTTPS	59
3.8.1	Recomendaciones de las vulnerabilidades en HTTPS.....	62
3.9	Compatibilidad con los navegadores y clientes web	62
4.	DECÁLOGO DE RECOMENDACIONES	64
ANEXO A.	REQUISITOS PARA EL ANÁLISIS DE SITIOS WEB HTTPS.....	65
ANEXO B.	SUITES CRIPTOGRÁFICAS RECOMENDADAS	67
ANEXO C.	REFERENCIAS	68

1. SOBRE CCN-CERT

El CCN-CERT (www.ccn-cert.cni.es) es la Capacidad de Respuesta a Incidentes de Seguridad de la Información del Centro Criptológico Nacional, CCN (www.ccn.cni.es). Este servicio se creó en el año 2006 como el **CERT Gubernamental/Nacional** español y sus funciones quedan recogidas en la Ley 11/2002 reguladora del Centro Nacional de Inteligencia, el RD 421/2004 regulador del CCN y en el RD 3/2010, de 8 de enero, regulador del Esquema Nacional de Seguridad (ENS), modificado por el RD 951/2015 de 23 de octubre.

De acuerdo a todas ellas, es competencia del CCN-CERT la gestión de ciberincidentes que afecten a **sistemas del sector público**, a **empresas y organizaciones de interés estratégico** para el país y a cualquier sistema clasificado. Su misión, por tanto, es contribuir a la mejora de la ciberseguridad española, siendo el centro de alerta y respuesta nacional que coopere y ayude a responder de forma rápida y eficiente a los ciberataques y a afrontar de forma activa las ciberamenazas.

2. RESUMEN EJECUTIVO

La proliferación de las tecnologías web en los últimos años, junto al aumento de las capacidades, funcionalidades, prestaciones y posibilidades de utilización de las mismas, permitiendo la realización de prácticamente cualquier gestión o tarea de la vida diaria, tanto personal como profesional, hace necesario plantearse la importancia de hacer uso de comunicaciones web lo más seguras posibles, es decir, basadas únicamente en el uso del protocolo HTTPS, que proporciona tanto mecanismos de autenticación como de cifrado e integridad.

Es por tanto necesario potenciar el uso de tecnologías web con un mayor nivel de seguridad a través de HTTPS, frente a HTTP, tanto dentro de los entornos corporativos como para el uso particular de las mismas, e independientemente de si son accedidas desde ordenadores tradicionales como desde dispositivos móviles, o desde cualquier otro dispositivo tecnológico, como los asociados al Internet de las Cosas (IoT, Internet of Things).

La evolución de HTTPS en la industria ha sufrido avances muy significativos durante los últimos años, especialmente en 2016 y 2017 [Ref – 2], dónde por primera vez en la historia, los datos de telemetría recopilados por Mozilla muestran que en octubre de 2016 el tráfico HTTPS global de Internet superó al tráfico HTTP (más del 50%). Esa tendencia alcista del uso de HTTPS frente a HTTP continua a lo largo del año 2017, habiéndose alcanzado casi el 55% en el mes de marzo².

² https://ipvsx/telemetry/general-v2.html?channels=release&measure=HTTP_PAGELOAD_IS_SSL&target=1&absolute=0&relative=1

Yesterday, for the first time, @Mozilla telemetry shows more than 50% of page loads were encrypted with HTTPS.

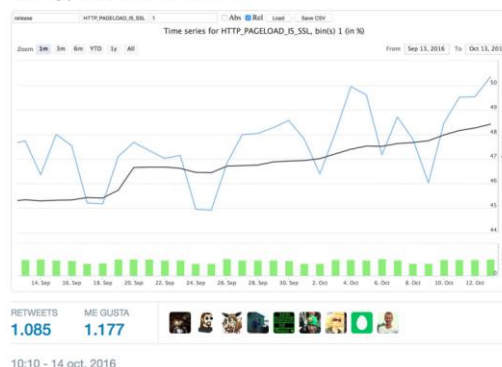


Figura 1.- Tráfico global de Internet según Mozilla: HTTPS vs. HTTP. Fuente: Mozilla³

Uno de los principales motivos por los que se debe hacer uso de HTTPS para el acceso a entornos, servidores y aplicaciones web son las características de seguridad adicionales proporcionadas por este protocolo empleado para establecer comunicaciones seguras, donde cabe destacar la posibilidad por parte del usuario de confirmar que se está conectando al entorno web deseado (autenticación e identificación) y la ventaja de que todo el tráfico intercambiado entre el usuario y el entorno web esté cifrado, asegurando a la vez la integridad de los datos transmitidos, y evitando que el tráfico pueda ser interceptado y manipulado por un tercero.

En la actualidad, todo el tráfico web intercambiado mediante el protocolo HTTP debería ser considerado sensible, incluso al hacer uso de servidores y aplicaciones web públicos donde no es necesario autenticarse, ya que en todo momento se debe velar por la privacidad del usuario y por la protección de los datos intercambiados. Por este motivo, es necesario generalizar el uso del protocolo HTTPS a todos los entornos web, tanto públicos como privados (o internos), de cualquier organización o compañía.

Un ejemplo de esta tendencia en la industria, desde el punto de vista de servicios ampliamente utilizados en Internet, se materializó en octubre de 2011, cuando Google convirtió su buscador web al protocolo HTTPS⁴, un servicio disponible públicamente (donde no es necesario autenticarse), pero en el que se deben de proteger los términos de búsqueda personalizados empleados por los usuarios, velando por su privacidad. Por otro lado, desde el punto de vista de los clientes web, en enero de 2017 los principales navegadores web (como, por ejemplo, Firefox 51⁵ y Chrome 56⁶) han comenzado a resaltar negativamente los servicios web que no hacen uso de HTTPS y, especialmente, los que solicitan información sensible al usuario, como por ejemplo credenciales o tarjetas de crédito, destacándolos en color rojo y con el mensaje de sitio web "No seguro" (o "Inseguro").

³ <https://twitter.com/0xjosh/status/786971412959420424>, <https://twitter.com/letsencrypt/status/786977436109934592>

⁴ <https://googleblog.blogspot.com.es/2011/10/making-search-more-secure.html>

⁵ <https://blog.mozilla.org/security/2017/01/20/communicating-the-dangers-of-non-secure-http/>

⁶ <https://security.googleblog.com/2016/09/moving-towards-more-secure-web.html>

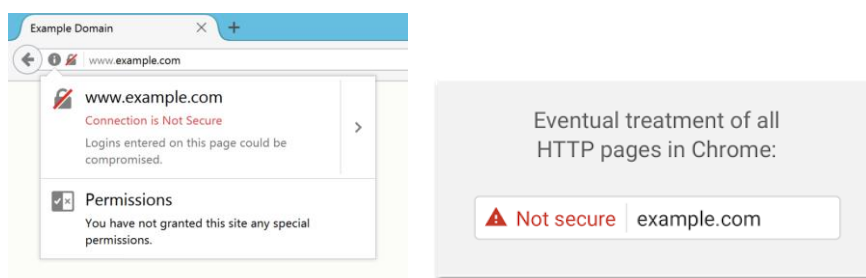


Figura 2.- Mensajes indicando que un servidor web HTTP no es seguro. Fuente: Firefox & Chrome

Pese a que las tecnologías web se utilizan habitualmente para establecer comunicaciones personales y profesionales, privadas y relevantes, para el intercambio de información sensible, y para el acceso a numerosos servicios, posiblemente debido a su utilización generalizada y a su facilidad de uso, el nivel de percepción de la amenaza de seguridad real existente al no hacerse uso de HTTPS no ha tenido la suficiente trascendencia en los usuarios finales y en las organizaciones. Los entornos web de las organizaciones, y las comunicaciones HTTP por parte de los usuarios, suelen ser objeto de numerosos ataques que, como resultado, permiten el acceso a información sensible, incluyendo datos personales e, incluso, identificadores de sesión y credenciales que permitirían suplantar a sus propietarios.

Debe tenerse en cuenta que HTTPS únicamente mitiga algunos ataques específicos asociados a las tecnologías web, siendo necesario establecer otras medidas de seguridad adicionales para proteger los servidores y aplicaciones web frente a otros ataques, como inyección SQL (SQLi), XSS (Cross-Site Scripting), CSRF (Cross-Site Request Forgery), ataques sobre la gestión de sesiones, LFI/RFI, etc.

La concienciación sobre el uso exclusivo de entornos web mediante HTTPS por parte de los usuarios y las buenas prácticas en la implementación de HTTPS (diseño, configuración y mantenimiento) en los entornos web por parte de sus responsables, constituyen la mejor defensa para prevenir y mitigar este tipo concreto de incidentes y amenazas web. El presente documento tiene como objetivo describir estas prácticas con el fin de ayudar a los responsables de los entornos web a proteger tanto sus servidores y aplicaciones web como a los usuarios finales, profundizando en la configuración y utilización de los mecanismos de protección existentes en la actualidad.

Para ello, se proporcionará un conjunto detallado de pautas y recomendaciones técnicas de seguridad que permitan mejorar la seguridad e incrementar las capacidades de protección de la implementación de HTTPS en entornos web. Adicionalmente, el apartado "ANEXO C. REFERENCIAS" proporciona diversas referencias y documentación para profundizar en todos los aspectos de seguridad descritos a lo largo del presente informe⁷, complementando las numerosas referencias existentes en el pie de página dentro de los diferentes apartados del informe.

El presente informe de Buenas Prácticas, "Recomendaciones de implementación de HTTPS" (CCN-CERT BP-01/17), debido a su carácter más técnico, está orientado a un público objetivo ligeramente diferente al del resto de informes de Buenas Prácticas del CCN-CERT [Ref – 1], cuyo

⁷ Destaca especialmente el libro "Bulletproof SSL and TLS" [Ref - 10].

público objetivo es el usuario final de las tecnologías analizadas. En este caso, el público objetivo son los administradores y/o responsables técnicos de seguridad de los entornos, servidores y aplicaciones web que deben de evaluar e implementar soporte para el protocolo HTTPS de la manera más segura posible.

3. BUENAS PRÁCTICAS Y RECOMENDACIONES PARA LA IMPLEMENTACIÓN DE HTTPS

El presente informe proporciona detalles técnicos para la implementación y despliegue seguro del protocolo HTTPS, basado en TLS (Transport Layer Security), en entornos, servidores y aplicaciones web. Con el objetivo de que resulte más sencillo para el lector comprender el porqué de las diversas recomendaciones de seguridad que se describirán a continuación, en algunos casos se realizará una breve descripción, o se proporcionarán referencias, a las amenazas e incidentes de seguridad que afectan a las implementaciones de HTTPS en entornos web en la actualidad, y a alguna de las técnicas más utilizadas por los atacantes.

El conjunto de recomendaciones mostrado se encuentra dividido en múltiples apartados, cada uno de ellos relacionado con diferentes elementos, capacidades y funcionalidades asociadas a las implementaciones de HTTPS, incluyendo el acceso a través de los puertos TCP/IP asociados a los servicios web, la generación de las claves, la gestión de los certificados digitales y otras consideraciones asociadas a la infraestructura global de clave pública (PKI, Public Key Infrastructure) de Internet, los protocolos SSL y TLS y sus diferentes versiones, los algoritmos y las *suítes* criptográficas, los mecanismos de revocación de certificados digitales, recomendaciones para publicar los contenidos web y desplegar las aplicaciones web mediante HTTPS, etc. Cada apartado dispone al final de un resumen de las recomendaciones principales.

El objetivo de estas recomendaciones es que el administrador y/o responsable técnico de seguridad de los entornos, servidores y aplicaciones web pueda aumentar el nivel de protección y de robustez de los servicios web en relación a la implementación y soporte disponible para el protocolo HTTPS, tanto desde el punto de vista de su diseño y configuración, como de su gestión diaria y mantenimiento, evitando así que dichos servicios web sean víctima de ataques conocidos sobre HTTP y HTTPS. Asimismo, en todo momento se tendrán en cuenta las posibles implicaciones de cómo los mecanismos de seguridad propuestos pueden afectar al rendimiento de las comunicaciones del entorno web, proporcionándose recomendaciones para mejorar el rendimiento del tráfico web incluso cuando se hace uso de HTTPS.

Debe tenerse en cuenta que algunas de las características descritas y, por tanto, de las recomendaciones de seguridad planteadas, son dependientes del tipo de sistema operativo empleado por el entorno web (Linux, BSD, Unix, Windows, etc.), del tipo de servidor web empleado (Apache, nginx, IIS, etc.), del tipo de servidor de aplicación (Weblogic, Websphere, JBoss/WildFly, Tomcat, etc.), así como de las versiones de todos estos elementos, y de la existencia de otros componentes perimetrales, como balanceadores de carga, redes de entrega de contenidos (CDN, Content Delivery Network), proxies inversos, cortafuegos (firewalls), cortafuegos de aplicación web (WAF, Web Application Firewall), concentradores o terminadores de HTTPS, etc. Por tanto, no necesariamente todas las recomendaciones ofrecidas serán de aplicación para todos los componentes existentes en el entorno web. En cualquier caso, se recomienda aplicar el mayor número de recomendaciones posible.

Las referencias y los ejemplos técnicos más específicos de configuración reflejados a lo largo del presente informe hacen uso de OpenSSL⁸, al ser potencialmente la librería (de código abierto) que implementa los protocolos SSL y TLS más comúnmente utilizada, y Apache⁹, como el servidor web (también de código abierto) más usado en sitios web activos en Internet actualmente y a lo largo de los últimos años (con casi un 46% de cuota de mercado en febrero de 2017¹⁰). Por tanto, los administradores y/o responsables técnicos de seguridad de los entornos, servidores y aplicaciones web que tienen que evaluar e implementar soporte para el protocolo HTTPS deben de identificar detalladamente, en cada caso, las posibilidades existentes y las particularidades de configuración específicas sobre las tecnologías web utilizadas en el entorno donde será aplicado el presente informe.

Desde el punto de vista de seguridad, se recomienda implantar la totalidad de los mecanismos de seguridad descritos de manera simultánea y complementaria, aplicando el principio de defensa en profundidad. Debe tenerse en cuenta que no todos los navegadores y clientes web disponen de soporte para los diferentes estándares y mecanismos de seguridad sugeridos. Por tanto, la aplicación de múltiples medidas complementarias, y aunque en ocasiones puedan parecer repetitivas (como, por ejemplo, la utilización de una redirección 301 desde HTTP hacia HTTPS, el uso de la cabecera HSTS o el uso de la cabecera CSP para actualizar las peticiones inseguras HTTP a peticiones HTTPS), permitirá acomodar el entorno web a las capacidades disponibles en los diferentes clientes que accederán al mismo.

Una vez aplicadas todas o la mayoría de recomendaciones sugeridas a lo largo del presente informe, se recomienda verificar el estado global de la implementación HTTPS para los entornos, servidores y aplicaciones web públicos mediante el servicio "SSL Server Test" [Ref - 4] de Qualys SSL Labs. A la hora de hacer uso del servicio, se recomienda seleccionar la opción "*Do not show the results on the boards*" antes de llevar a cabo el análisis, con el objetivo de no aparecer en el listado público de los sitios web analizados más recientemente, o con mejores o peores resultados¹¹. El objetivo del análisis debería centrarse en obtener una calificación final de A o A+. Cualquier calificación menor debería considerarse una anomalía o debilidad, y debería ser subsanada. Se debe tener en cuenta que una calificación de A, a día de hoy puede llevar asociada una calificación de B dentro de unos meses, ya que los mecanismos de seguridad de HTTPS y los requisitos mínimos asociados a las buenas prácticas están en continua evolución, mejora y revisión [Ref - 21].

3.1 Acceso a los puertos TCP/IP asociados a los servicios web

Los servicios web están asociados por defecto al puerto TCP/80 en el caso del protocolo HTTP, y al puerto TCP/443 en el caso del protocolo HTTPS, que adicionalmente hace uso de mecanismos de autenticación y cifrado mediante TLS.

⁸ <https://www.openssl.org>

⁹ <https://www.apache.org> (<https://httpd.apache.org>)

¹⁰ <https://news.netcraft.com/archives/2017/02/27/february-2017-web-server-survey.html>

¹¹ Debe tenerse en cuenta que nada impide que un tercero proporcione la dirección (o *hostname*) de un servidor web propio a través de este servicio, sin seleccionar dicha opción, por lo que el mismo aparecería listado en las diferentes categorías mencionadas, públicamente disponibles.

A la hora de desplegar un servicio web, y siguiendo la tendencia de la industria centrada en convertir y basar la web únicamente en el protocolo HTTPS (eliminando lo más posible el uso del protocolo HTTP), se recomienda hacer un uso exclusivo de HTTPS, por lo que es necesario disponer del puerto TCP/443 abierto en el entorno web.

Sin embargo, a día de hoy los navegadores y clientes web hacen uso por defecto del protocolo HTTP y el puerto TCP/80 cuando no se especifica el protocolo a emplear ("http://" o "https://") como, por ejemplo, cuando el usuario únicamente proporciona el nombre del servidor web con el que desea conectar (por ejemplo, www.dominio.es)¹².

Con el objetivo de proporcionar la mayor accesibilidad y disponibilidad posible del entorno web, se recomienda disponer del puerto TCP/80 también abierto¹³, pero configurando el servidor o aplicación web para que se lleve a cabo una redirección automática (de tipo 301 o 302) desde HTTP hacia HTTPS ante cualquier petición (intento de conexión) en el puerto TCP/80. Específicamente se recomienda hacer uso de una redirección de tipo 301 ("301 Moved Permanently") de HTTP hacia HTTPS¹⁴, ya que este tipo de respuesta puede ser cacheada y aplicada continuamente por los clientes y navegadores web (ver apartado "3.7.3. Forzando el uso (por defecto) de HTTPS").

Alternativamente, se podría emplear una implementación más restrictiva donde el puerto TCP/80 permanezca cerrado o filtrado, y únicamente se disponga de acceso al puerto TCP/443. En este caso, por accesibilidad, todas las referencias al servidor web deberán especificar el protocolo HTTPS mediante "https://", como, por ejemplo, todos los enlaces existentes en "www.dominio.es" que apuntan a "<https://sede.dominio.es>". En caso contrario, si se cierra el puerto TCP/80 y un usuario simplemente teclea la dirección (URL) del servidor o aplicación web en la barra de dirección de su navegador web, obtendrá un error de conexión. Si no se dispone de control sobre todas las referencias externas que apuntan al servidor web (situación habitual cuando existen referencias de terceros apuntando a él), o en caso de duda, se recomienda abrir el puerto TCP/80 y configurar la redirección automática de HTTP a HTTPS previamente descrita.

Este escenario donde se recomienda disponer del puerto TCP/80 abierto, con una redirección automática de HTTP a HTTPS, también se debería emplear incluso en escenarios más seguros donde se hace uso de HSTS, e incluso si se ha añadido el dominio en la lista precargada mediante la directiva "preload" (ver apartado "3.7.4. Forzando el uso estricto (por defecto) de HTTPS: HSTS"), ya que podrían presentarse situaciones donde un cliente no hace uso de la lista precargada de HSTS, o emplea una versión del navegador web que no incluía todavía el dominio bajo estudio en dicha lista.

3.1.1 Recomendaciones de acceso a los puertos TCP/IP de los servicios web

El resumen de recomendaciones de acceso a los puertos TCP/IP asociados a los servicios web incluye:

- Disponer del puerto estándar TCP/443, asociado al protocolo HTTPS, abierto.

¹² Únicamente, cuando en un hipotético futuro los navegadores web hagan uso del puerto TCP/443 (HTTPS) por defecto, en lugar de HTTP, será posible plantearse el cerrar el puerto TCP/80 con garantías elevadas de accesibilidad.

¹³ <https://scotthelme.co.uk/why-closing-port-80-is-bad-for-security/>

¹⁴ <https://support.google.com/webmasters/answer/6073543?hl=en>

- Disponer del puerto TCP/80 también abierto, pero configurar el servidor o aplicación web para que se lleve a cabo una redirección automática de tipo 301 ("301 Moved Permanently") desde HTTP hacia HTTPS ante cualquier petición en el puerto TCP/80.

3.2 Generación de las claves criptográficas

La correcta generación de las claves privadas criptográficas asociadas a los certificados digitales empleados en HTTPS, y concretamente la adecuada selección del algoritmo de generación de las claves y de su tamaño (o longitud), es fundamental para la seguridad de HTTPS. Debe tenerse en cuenta que el punto más débil relativo a las claves suele estar asociado a la gestión de las mismas y a la tarea diaria de mantener las claves privadas en secreto.

Las claves criptográficas, y su vinculación al nombre de la entidad representada por el servidor web a través de un certificado digital, constituyen la identidad criptográfica del servidor web en HTTPS.

El algoritmo más ampliamente utilizado en la actualidad para la generación de las claves privadas, y sus claves públicas asociadas, es RSA (Rivest, Shamir y Adleman). Se recomienda hacer uso de claves RSA con una longitud mínima de 2.048 bits (que proporcionan 112 bits de seguridad). El uso de claves mayores, por ejemplo, de 3.072 bits puede tener un impacto significativo en el rendimiento del entorno web.

En el futuro, es probable que se extienda aún más el uso de claves ECDSA (Elliptic Curve Digital Signature Algorithm) por encima de RSA, a veces referenciadas como ECC (Elliptic Curve Cryptography)¹⁵, ya que proporcionan un mayor rendimiento que RSA para claves de longitud y nivel de seguridad equivalente. Se recomienda hacer uso de claves ECDSA con una longitud mínima de 256 bits (que proporcionan 128 bits de seguridad y el doble de velocidad que una clave RSA de 2.048 bits, y unas 6 veces más de velocidad que una clave RSA equivalente de 3.072 bits).

Se debe de buscar siempre el equilibrio entre seguridad y rendimiento, evitando excederse en introducir "demasiada" seguridad, por lo que, en la actualidad, para entornos web estándar se recomienda hacer uso de claves RSA de 2.048 bits y claves ECDSA de 256 bits. Desde el punto de vista de seguridad y rendimiento, es preferible hacer uso de ECDSA frente a RSA.

A modo de referencia, una clave RSA de 2.048 bits sería equivalente a una clave ECDSA de 224 bits (ambas con 112 bits de seguridad), una clave RSA de 3.072 bits a una clave ECDSA de 256 bits (ambas con 128 bits de seguridad), una clave RSA de 7.680 bits a una clave ECDSA de 384 bits (ambas con 192 bits de seguridad), y una clave RSA de 15.360 bits a una clave ECDSA de 512 bits (ambas con 256 bits de seguridad)¹⁶. El número de bits de seguridad sería a su vez equivalente al tamaño en bits recomendado para las claves empleadas en criptografía simétrica (ej. AES), y el número de bits de ECDSA equivalente (aproximadamente) a la longitud de los tipos de hashes recomendados para una robustez similar (ej. ECDSA 256 bits y SHA-256)¹⁷.

El principal inconveniente de ECDSA en la actualidad es que no todos los navegadores y clientes web (agentes de usuario) lo soportan, especialmente algunos más antiguos (como por ejemplo

¹⁵ <https://blog.cloudflare.com/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/>

¹⁶ <https://casecurity.org/2014/06/10/benefits-of-elliptic-curve-cryptography/>

¹⁷ <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf>

IE ≤ 8 o Chrome en Windows XP, u OpenSSL $< 1.0.0$). Es posible evaluar los algoritmos o suites criptográficas soportadas por un cliente web específico, incluyendo el soporte para claves RSA y/o ECDSA, a través del servicio "SSL Client Test" de Qualys SSL Labs [Ref - 3], pudiéndose seleccionar cada uno de los clientes y navegadores web para consultar todas sus características relacionadas con HTTPS [Ref - 6], o mediante "SSL Cipher Suite Details of Your Browser" [Ref - 5].

Existe la posibilidad de hacer uso de claves RSA y ECDSA simultáneamente, con el objetivo de proporcionar el mayor nivel de seguridad posible a cada tipo de cliente web. Sin embargo, esta posibilidad depende de las tecnologías y plataformas utilizadas en la implementación de HTTPS, por lo que se recomienda verificar la existencia de soporte para el uso simultáneo de claves diferentes en el entorno web donde se aplicará el presente informe. Por ejemplo, Apache permite este tipo de configuración, empleando distintas claves RSA y ECDSA, y por tanto certificados digitales diferentes, simultáneamente.

A la hora de seleccionar y generar las claves es necesario evaluar las opciones que son razonablemente seguras en la actualidad, y que también lo serán hasta que la clave sea reemplazada, ya que permitirán determinar el periodo de tiempo durante el cual las claves permanecerán razonablemente protegidas antes de ser reemplazadas.

Debe tenerse en cuenta que, en la mayoría de ataques e incidentes de seguridad relacionados con criptografía, se acaba evitando el cifrado, en lugar de romperlo o comprometerlo, ya que es una tarea mucho más sencilla. Por tanto, se recomienda realizar una correcta gestión de las claves, manteniendo en secreto en todo momento las claves privadas, haciendo uso de un buen generador de números aleatorios o PRNG (pseudo-random number generator), protegiendo las claves con una contraseña robusta (*passphrase*) y almacenándolas de forma segura (por ejemplo, empleando un HSM, Hardware Security Module), realizando copias de seguridad o *backups* seguros de las claves, y renovando las claves periódicamente.

En ningún caso se debe delegar el proceso de generación de la clave privada a la Autoridad Certificadora (Certification Authority, en adelante, CA) y, en su lugar, debe de ser generada localmente y proporcionar a la CA una solicitud de firma de un certificado (CSR, Certificate Signing Request), para que la CA proceda a la emisión del certificado digital asociado. La CSR debe generarse empleando al menos SHA-256 como algoritmo de firma o *hashing*. Se debe adicionalmente aplicar el principio de mínimo privilegio, exponiendo la clave privada al menor número posible de sistemas y personas.

Con el objetivo de que la generación aleatoria de los números asociados a las claves criptográficas sea lo mejor posible, y para minimizar la exposición de estas, se recomienda hacer uso de un equipo *offline*, que no esté expuesto a ninguna red de comunicaciones, y con suficiente entropía (por ejemplo, no generando la clave nada más reiniciar dicho equipo, sino tras largos periodos de uso), con idea de generar claves suficientemente robustas.

El almacenamiento de las claves debe ser protegido empleando una contraseña robusta (*passphrase*) y, preferiblemente, haciendo uso de módulos de seguridad hardware específicos (HSM, Hardware Security Module), en lugar de disponer de una o múltiples copias de las claves en sistemas de ficheros estándar de uno o varios servidores. Asimismo, la utilización de un mismo conjunto de claves, y de sus certificados digitales asociados, en servidores web con

distinto nivel de seguridad y criticidad es muy problemático, ya que un incidente de seguridad en uno de los servidores expondría también al resto de servidores.

Por último, se debería llevar a cabo una correcta gestión de las claves, registrando cuando son generadas, puestas en producción y retiradas o renovadas finalmente, empleando un proceso de borrado seguro. Las claves deberían renovarse periódicamente (y a ser posible, de manera automática), con el objetivo de que no sean utilizadas durante muy largos periodos de tiempo, y para minimizar el impacto asociado en caso de sufrir un incidente de seguridad o cualquier otro evento que pueda exponer las mismas. Tras un incidente de seguridad, se deberían revocar los certificados digitales previos, generarse nuevas claves, y solicitar la emisión de nuevos certificados digitales para estas nuevas claves.

Una práctica común es renovar las claves criptográficas cada vez que se renueva el certificado digital asociado (ver apartado "3.3.5. Renovación de los certificados digitales"), aunque la introducción de nuevos mecanismos de seguridad como HPKP (ver apartado "3.3.8. Restricción de los certificados digitales legítimos: HPKP"), pueden influir en el periodo y/o procedimiento de renovación de las claves criptográficas (ya que los pines están asociados a la clave criptográfica, siendo necesario actualizarlos si la clave es renovada).

3.2.1 Recomendaciones para la generación de las claves criptográficas

El resumen de recomendaciones para la generación de las claves criptográficas incluye:

- Hacer uso de claves RSA de 2.048 bits de longitud.
- Alternativamente, o complementariamente, hacer uso de claves ECDSA de 256 bits de longitud, ya que ofrecen mayor seguridad y rendimiento que las claves RSA.
- Se recomienda buscar el equilibrio entre seguridad y rendimiento, evitando excederse en introducir "demasiada" seguridad, por lo que actualmente es preferible hacer uso de claves ECDSA (256 bits) frente a RSA (2.048 bits).
- Hacer uso simultáneo, si las tecnologías web empleadas lo soportan, de claves ECDSA y RSA.
- Hacer uso de un buen generador de números aleatorios para generar las claves.
- El almacenamiento de las claves debe ser protegido empleando una contraseña robusta (passphrase) y, alternativamente, mediante módulos de seguridad hardware específicos (HSM, Hardware Security Module).
- Realizar una correcta gestión de las claves: mantener en secreto las claves privadas, disponer de copias de seguridad o backups seguros de las claves, renovar las claves periódicamente, emplear un proceso de borrado seguro, renovar las claves tras un incidente de seguridad, exponer las claves privadas al menor número posible de sistemas y personas, etc.

3.3 Gestión de los certificados digitales

El presente apartado incluye recomendaciones relacionadas con múltiples aspectos asociados con la correcta selección, obtención y gestión de los certificados digitales utilizados por HTTPS y vinculados a las claves criptográficas previamente mencionadas (ver apartado "3.2. Generación de las claves"). Disponer de claves criptográficas y certificados digitales robustos y seguros evita que un potencial atacante pueda suplantar fácilmente el entorno web.

3.3.1 Tipos de certificados digitales

Actualmente existen tres tipos de certificados digitales, ordenados por el nivel de seguridad o confianza que proporcionan y por su coste: validado por dominio (DV, Domain Validated), validado por organización (OV, Organization Validated), y con validación extendida (EV, Extended Validation).

Los certificados DV se validan automáticamente y son los más económicos, mientras que los EV aplican procedimientos de validación estandarizados (por el CAB Forum o CA/Browser Forum¹⁸), son más caros, y son tratados de manera diferente por los navegadores web, mostrando la barra de dirección (o URL) en color verde y reflejando información de la organización propietaria del certificado digital junto a la dirección web (o URL).

La utilización de un tipo u otro de certificado digital depende de múltiples factores, incluyendo tanto los económicos como los asociados a la imagen de la organización y de la confianza que se desea transmitir de cara a los usuarios del servidor o aplicación web.

3.3.2 Emisión de certificados digitales: Autoridades Certificadoras (CAs)

El presente informe se centra en la implementación de HTTPS en servidores web públicos, por lo que es obligatorio disponer de un certificado digital emitido por una CA pública de confianza y ampliamente reconocida. No es posible, por tanto, para la correcta implementación de HTTPS, hacer uso de certificados digitales autofirmados ni de CAs privadas.

En el caso del Esquema Nacional de Seguridad (ENS), se recomienda utilizar un prestador de servicios de confianza.

Incluso para los servidores web internos, se recomienda disponer de un certificado digital emitido por una CA pública de confianza, o en su defecto, por una CA privada pero también ampliamente reconocida por los clientes web.

A la hora de solicitar un certificado digital a una CA pública, es necesario ser consciente de que se está estableciendo una estrecha relación a medio o largo plazo con ésta, especialmente si se hace uso de nuevos mecanismos de seguridad como HPKP (ver apartado "3.3.8. Restricción de los certificados digitales legítimos: HPKP").

Por tanto, se recomienda evaluar y seleccionar la CA en base a criterios objetivos en función de la naturaleza, criticidad y nivel de seguridad del servidor web en el que se desea hacer uso de HTTPS. Entre estos criterios se debería evaluar en nivel de consolidación y madurez de la CA, su reputación en la industria, su reconocimiento como CA de confianza en múltiples navegadores y clientes web, las capacidades de gestión de numerosos certificados digitales, el tipo de servicio administrativo y de soporte técnico proporcionado por la CA, su posición en la industria y su disposición a adoptar nuevas tecnologías y estándares (como por ejemplo CT - ver apartado "3.3.6. Certificate Transparency (CT)", CAA - ver apartado "3.3.7. CAA (Certification Authority Authorization)", OCSP Stapling - ver apartado "3.6.3. OCSP Stapling", la posibilidad de obtener certificados digitales basados en claves RSA y ECDSA - ver apartado "3.2. Generación de las claves criptográficas", etc.), y por supuesto, el nivel de seguridad y mecanismos de protección de

¹⁸ <https://cabforum.org>

la propia CA y su transparencia a la hora de gestionar incidentes de seguridad (en base al histórico de incidentes pasados).

Otro elemento interesante a considerar durante la evaluación de la CA a seleccionar son los mecanismos de revocación de certificados digitales disponibles (ver apartado "3.6. Mecanismos de revocación de certificados digitales"), debiendo ofrecer como mínimo CRL y OCSP, junto al análisis de la disponibilidad y el rendimiento ofrecido para estos servicios.

Alternativamente a las CAs comerciales, en la actualidad se dispone de una CA sin ánimo de lucro denominada Let's Encrypt [Ref - 11], cuyo objetivo es generalizar el uso de HTTPS en Internet mediante la emisión de certificados digitales gratuitos (de tipo DV). Let's Encrypt representa una alternativa válida para la obtención de certificados digitales legítimos y ampliamente reconocidos, y se caracteriza por emplear un modelo con un corto periodo de renovación de los certificados (actualmente establecido en 90 días, 3 meses¹⁹) con el objetivo de minimizar el impacto de un posible incidente de seguridad y de promover la automatización del proceso de solicitud y renovación de los certificados digitales.

A modo de referencia²⁰, en febrero de 2016 Let's Encrypt introdujo soporte para la generación de certificados digitales basados en claves ECDSA²¹, y antes de septiembre de 2017 anunció la disponibilidad de una CA raíz y CAs intermedias también basadas en claves ECDSA, pudiéndose disponer así de una cadena de certificación (o de confianza) completa basada en ECDSA.

Adicionalmente a la selección de la CA que emitirá los certificados digitales, una vez emitidos, la CA incluye su propia firma en el certificado para probar su validez. Para ello hace uso de un algoritmo de firma, considerándose hoy en día que únicamente debe disponerse de certificados digitales firmados mediante SHA-256, o algoritmos más robustos (ej. SHA-384, SHA-512...), también referenciados como SHA-2. En ningún caso se debería hacer uso de SHA-1 o MD5 para la firma del certificado.

Debido a que la solicitud de un certificado mediante una CSR no permite especificar el algoritmo de firma que se desea que sea utilizado por la CA, se recomienda comprobar con la CA, antes de realizar la petición del certificado digital, qué algoritmos está empleando actualmente para la firma de certificados digitales. Adicionalmente, también se recomienda comprobar con la CA cuál será la cadena completa de certificación (o de certificados) asociada al certificado digital solicitado, es decir, la lista completa de CAs intermedias asociadas a la emisión del certificado desde el mismo hasta la CA raíz, e identificar la propia CA raíz que será asignada al certificado digital solicitado.

Es importante clarificar que la recomendación de no utilizar SHA-1 es de aplicación para la firma de los certificados digitales raíz de las propias CAs. Los algoritmos de firma utilizados en el pasado como MD5 o SHA-1 son considerados inseguros y no deberían ser utilizados para firmar certificados digitales:

¹⁹ <https://letsencrypt.org/2015/11/09/why-90-days.html>

²⁰ <https://scotthelme.co.uk/ecdsa-certificates/>

²¹ <https://letsencrypt.org/upcoming-features/>

- MD5 fue diseñado para proporcionar 64 bits de seguridad y fue completamente vulnerado, de forma práctica sobre certificados digitales empleados para HTTPS, en el año 2009²².
- SHA-1 fue diseñado para proporcionar 80 bits de seguridad, aunque realmente proporciona 61 bits de seguridad efectiva. Desde enero de 2016 las CAs no deberían emitir certificados digitales firmados con SHA-1²³ (bajo petición inicial de Microsoft), y finalmente en enero/febrero de 2017 los principales navegadores web (Firefox 51²⁴, Edge & IE 11²⁵, o Chrome 56²⁶) dejaron de admitir como válidos los certificados digitales firmados con SHA-1.

En febrero de 2017 Google, junto al CWI Institute en Ámsterdam, publicaron un ataque práctico contra SHA-1 [Ref - 12].

3.3.3 Entidad asociada al certificado digital

Dentro de los diferentes campos o elementos de seguridad incluidos en un certificado digital (ver apartado "3.3.4. Validez del certificado digital"), cabe destacar el nombre o identificador de la entidad (entorno, servidor o aplicación web) a la que está asociada el certificado.

Cuando un navegador o cliente web se conecta mediante HTTPS al servidor web, verifica que el nombre de la entidad asociada al certificado digital corresponda con el nombre empleado en la dirección web (o URL) visitada. En caso contrario, se generarán errores del certificado que pueden confundir a los usuarios y reducir la confianza en el entorno web.

Es por tanto fundamental asegurarse de que los nombres y entidades reflejados en el certificado digital coinciden y dan cobertura (representan) a todos los servidores web que hacen uso de dicho certificado, ya sea con una referencia directa o única (como `www.dominio.es`), mediante una lista con todos los nombres de los servidores web (por ejemplo, `www.dominio.es`, `portal.dominio.es` y `sede.dominio.es`), como a través de un certificado digital comodín que englobe a todos los subdominios del dominio principal (como `*.dominio.es`)²⁷. La principal consideración a la hora de hacer uso de un certificado digital comodín es que el mismo sólo debería ser usado en servidores web con el mismo nivel de seguridad, para minimizar el impacto de un posible incidente de seguridad (y el acceso a las claves privadas vinculadas al certificado digital) y, de nuevo, debería aplicarse el principio de mínimo privilegio, intentando reducir la necesidad de tener que compartir el certificado digital y la clave privada entre numerosos departamentos, sistemas y personas.

Aunque también se dispone de la posibilidad de obtener certificados digitales multi-dominio, es decir, válidos para servidores web de diferentes dominios, por el mismo motivo (promover su uso en entornos con el mismo nivel de seguridad y disminuir las necesidades de compartir el

²² <https://documents.epfl.ch/users/l/le/lenstra/public/papers/lat.pdf>

²³ <https://threatpost.com/sha-1-end-times-have-arrived/>

²⁴ <https://blog.mozilla.org/security/2016/10/18/phasing-out-sha-1-on-the-public-web/>

²⁵ <https://blogs.windows.com/msedgedev/2016/11/18/countdown-to-sha-1-deprecation/>

²⁶ <https://security.googleblog.com/2016/11/sha-1-certificates-in-chrome.html>

²⁷ El uso de certificados digitales comodín es habitual en CDNs que gestionan el tráfico HTTPS destinado a múltiples servidores web de una misma organización.

certificado digital y la clave privada), se recomienda independizar los certificados digitales para cada dominio.

Por otro lado, se recomienda asegurarse de que el certificado digital incluye tanto el nombre del servidor web representado (o la lista de servidores web, o el certificado comodín) como el de su dominio, es decir, `www.dominio.es` y `dominio.es` (sin el prefijo "www"), o `*.dominio.es` y `dominio.es` (en el caso de usar un certificado digital comodín). Este escenario es habitual para los certificados digitales emitidos hoy en día (pero debe ser verificado explícitamente), ya que habitualmente ambos nombres hacen referencia en el servicio de DNS a la misma dirección IP, y por tanto al mismo servidor. Como regla general, el certificado digital debería ser válido para todos los nombres existentes en el servicio DNS para el servidor web asociado.

Antiguamente el nombre de la entidad asociada al certificado digital estaba reflejado en el campo "CN" (Common Name o `commonName`) del certificado. Posteriormente, el RFC 2818²⁸ sugería identificar el nombre a través del campo "SAN" (Subject Alternative Name, extensión "subjectAltName" de tipo "dNSName") y, alternativamente en su ausencia, a través del "CN". Este RFC, publicado en mayo de 2000, ya priorizaba el "SAN" frente al "CN", menospreciando este último. Pese a ello, numerosos clientes web han soportado ambos campos a lo largo de los últimos años.

Sin embargo, desde Firefox 48²⁹ y Chrome 58³⁰ no se dispone de soporte para la identificación del nombre mediante el "CN", generándose un error de certificado en caso de no encontrarse el nombre del servidor web esperado en el campo "SAN". Se recomienda por tanto hacer uso únicamente del campo "SAN" para reflejar la lista de nombres de las entidades asociadas al (o representadas por el) certificado digital.

3.3.4 Validez del certificado digital

Los certificados digitales X.509 disponen de numerosos campos o elementos de seguridad que son evaluados para determinar la validez del certificado por parte de un navegador o cliente web. Por este motivo, es necesario asegurarse en todo momento de que todos sus valores son válidos, incluyendo:

- Nombre de la entidad o servidor web (ver apartado "3.3.3. Entidad asociada al certificado digital").
- Periodo de validez o expiración del certificado (con la fecha de inicio y finalización de dicho periodo).
- CA que ha emitido y firmado el certificado (ver apartado "3.3.2. Emisión de certificados digitales: Autoridades Certificadoras (CAs)").
- Revocación del certificado (ver apartado "3.6. Mecanismos de revocación de certificados digitales").
- Propósito del certificado: Autenticación del servidor (asociado a los servidores web objeto del presente informe), del cliente, firma de código ejecutable o software, correo

²⁸ <https://tools.ietf.org/html/rfc2818>

²⁹ https://bugzilla.mozilla.org/show_bug.cgi?id=1245280

³⁰ <https://www.chromestatus.com/features/4981025180483584>

electrónico, etc. (esta clasificación es claramente visible especialmente en entornos Windows).

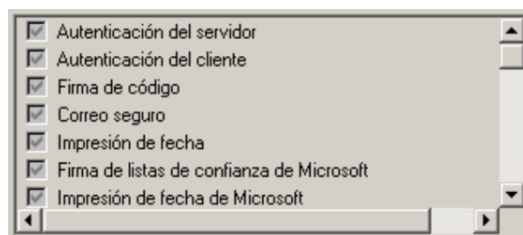


Figura 3.- Diferentes propósitos de utilización de un certificado digital. Fuente: Microsoft

Más específicamente, cuando se evalúa la validez de un certificado digital X.509 es necesario también comprobar que la totalidad de la cadena de certificación es válida, y más concretamente, que no está incompleta. Se recomienda que el servidor web proporcione a los clientes web la cadena de certificación completa (evitando así que estos la tengan que resolver y reconstruir de manera autónoma, situación que puede dar lugar a errores), proporcionando todos y cada uno de los certificados digitales emitidos o asociados a la CA en el orden correcto, es decir, desde la CA intermedia inmediatamente posterior a la CA raíz, hasta el certificado digital del propio servidor web. La cadena de certificación debe de incluir únicamente los certificados digitales necesarios; por motivos de rendimiento no se recomienda añadir el certificado de la CA raíz a la cadena, ya que éste ya debe de ser conocido por los navegadores y clientes web (al estar disponible en su almacén de CAs de confianza).

Es posible evaluar la cadena de certificación del servidor web con herramientas web como "Certificate Analyser"³¹.

3.3.5 Renovación de los certificados digitales

El periodo estándar habitual de validez o renovación de los certificados oscila actualmente entre 1, 2 o 3 años, dependiendo del tipo de certificado (DV, OV o EV). Lógicamente, en el periodo a seleccionar influye el coste económico (siendo habitual disponer de algún tipo de descuento por parte de las CAs para periodos más amplios) y la frecuencia con la que se desea llevar a cabo las gestiones administrativas de renovación asociadas.

Se recomienda renovar los certificados digitales anualmente, e incluso más frecuentemente si es posible automatizar el proceso de renovación (ver apartado "3.3.2. Emisión de certificados digitales: Autoridades Certificadoras (CAs)" y las referencias a Let's Encrypt). Si se considera que es muy complejo revocar completamente y de manera efectiva y fiable un certificado digital comprometido, desde el punto de vista práctico, los certificados digitales con una vida más reducida serían considerados más seguros.

En marzo de 2017 se lanzó un proceso de votación en el CAB Forum (Ballot 193) para definir un nuevo estándar que establezca que todos los certificados digitales (emitidos a partir del 1 de marzo de 2018) tendrán un periodo de validez máximo de 825 días³² (aproximadamente 27 meses, es decir, 2 años y tres meses), frente al periodo máximo actual de 39 meses (es decir, 3

³¹ https://report-uri.io/home/certificate_analyser/

³² <https://cabforum.org/pipermail/public/2017-March/010024.html>

años y tres meses). El resultado de la votación ratificó que el nuevo periodo de validez máximo de los certificados digitales emitidos a partir del 1 de marzo de 2018 se establece en 825 días³³.

Los certificados digitales deben de ser renovadas periódicamente y siempre antes de su vencimiento. Cada vez que se renueva el certificado digital, se recomienda también renovar la cadena de certificación completa, incluyendo todas las CAs intermedias (cuyos certificados digitales también han podido expirar), para proporcionar así a los clientes web una lista completa y actualizada de la cadena de certificación. Asimismo, se recomienda renovar las claves criptográficas cada vez que se renueva el certificado digital asociado, salvo que se esté haciendo uso de HPKP (ver apartado "3.3.8. Restricción de los certificados digitales legítimos: HPKP").

Se recomienda desplegar el nuevo certificado digital, una vez renovado, aproximadamente una semana después de haberlo obtenido, intentando evitar problemas con clientes web cuya referencia horaria no esté correctamente configurada y para dar suficiente margen a la CA a propagar el nuevo certificado como válido en su servicio OCSP. Adicionalmente, se debe tener en cuenta que no es recomendable revocar el certificado digital previo inmediatamente tras ser renovado, sino que haya cierto solapamiento en el periodo de validez de ambos certificados para permitir un proceso de migración fluido de uno a otro.

3.3.6 Certificate Transparency (CT)

El estándar Certificate Transparency (CT) [Ref- 13] de Google, o Transparencia en los Certificados, establece un entorno (o *framework*) abierto que permite la supervisión, monitorización y auditoría (en el proceso de emisión) de los certificados digitales por parte de las CAs, casi en tiempo real. La propuesta inicial se está convirtiendo en un estándar de aplicación en toda la industria, asociado al RFC 6962 [Ref - 14].

Las CAs han demostrado a lo largo de los últimos años ser vulnerables al uso inadecuado y a la manipulación de sus procedimientos de emisión de certificados digitales. CT permite identificar la emisión de nuevos certificados, ya que estos deben ser reflejados por las CAs en un registro públicamente verificable, y cuya integridad no puede ser manipulada, ya que hace uso de mecanismos criptográficos que sólo permiten añadir registros (pero no borrarlos o modificarlos)³⁴. CT permite la identificación temprana de múltiples escenarios no deseados como, por ejemplo, cuando una CA emite un certificado por error, cuando una CA (que, por ejemplo, ha sido adquirida por otra) emite certificados no solicitados, o cuando una CA se vuelve maliciosa, o ha sido comprometida, y emite certificados para suplantar a ciertos sitios web. La detección temprana permite mitigar el impacto de los certificados ilegítimos y, a su vez, proporciona transparencia y capacidades de escrutinio públicas de la salud y la integridad de todo el ecosistema de la infraestructura global de clave pública (PKI) de Internet.

Una vez se ha identificado mediante CT la emisión de un certificado digital no legítimo, es posible llevar a cabo acciones para revocar el mismo lo antes posible (ver apartado "3.6. Mecanismos de revocación de certificados digitales") y, en caso de ser necesario, sustituirlo por un certificado digital nuevo legítimo.

³³ <https://cabforum.org/pipermail/public/2017-March/010098.html>

³⁴ Para ello se usan "Merkle hash trees": www.certificate-transparency.org/log-proofs-work

CT permite monitorizar los logs públicos e identificar la emisión de certificados digitales para los dominios propios sin la autorización del propietario del dominio, ya sea por parte de otras CAs diferentes a la CA utilizada actualmente, o incluso por esta. Para ello, se dispone de recursos [Ref - 15]³⁵ que permiten consultar los registros existentes para un dominio, o incluso para un nombre (o host) concreto, pudiendo incluirse en la consulta los certificados digitales ya caducados y los asociados a los subdominios. La respuesta obtenida permite identificar las CAs emisoras y los certificados registrados públicamente en CT.

Como referencia, el 13 de diciembre de 2016 se habían registrado casi 172 millones de entradas en los logs o registros de CT, mientras que el 16 de marzo de 2017 ese número se había incrementado a casi 249 millones de entradas.

Complementariamente, existen servicios de monitorización de CT para poder ser informado de cambios en los certificados digitales asociados a un dominio, como por ejemplo Computest Suricat (<https://suricat.io>), o Cert Spotter (<https://sslmate.com/certspotter/>).

Busca todos los certificados emitidos para un nombre de host determinado que se incluyan en los registros públicos de Transparencia en los certificados.

☒ Incluir certificados caducados
☒ Incluir subdominios

Autoridades de certificación emisoras

Emisor ⓘ	N.º de certificados emitidos ⓘ	
C=ES, O=FNMT, OU=FNMT Clase 2 CA	1	Filtrar

Certificados

Asunto ⓘ	Emisor ⓘ	N.º de nombres de DNS ⓘ	Válido desde ⓘ	Válido hasta ⓘ	N.º de registros de transparencia de certificación ⓘ	
www.ccn-cert.cni.es	C=ES, O=FNMT, OU=FNMT Clase 2 CA	1	Mar 3, 2011	Mar 3, 2015	3	Mostrar detalles

Figura 4.- Resultados de la consulta a CT para www.ccn-cert.cni.es. Fuente: Google

El navegador web Chrome requiere que todos los certificados de tipo EV estén disponibles en CT desde el 1 de enero de 2015³⁶ y, debido a irregularidades de Symantec como CA, Chrome 53 (desde octubre de 2015) requiere el uso de CT para todos los certificados digitales emitidos por Symantec posteriores al 1 de junio de 2016³⁷. Adicionalmente, desde octubre de 2017, Chrome requerirá CT para todos los nuevos certificados digitales³⁸.

El entorno de CT está compuesto por tres componentes principales³⁹:

- Logs o registros de certificados: Servicio que se encarga de mantener los registros públicos criptográficos con los certificados digitales, dónde sólo se pueden añadir registros (proporcionados habitualmente por las CAs).
- Monitores: Servicio que se conecta a los registros públicos en busca de certificados sospechosos.

³⁵ <https://www.entrust.com/ct-search/>

³⁶ <https://sites.google.com/site/certificatetransparency/ev-ct-plan>

³⁷ <https://security.googleblog.com/2015/10/sustaining-digital-certificate-security.html>

³⁸ <https://groups.google.com/a/chromium.org/forum/#!topic/ct-policy/78N3SMcqUGw>

³⁹ <http://www.certificate-transparency.org/what-is-ct>

- Auditores: Servicio que, por un lado, verifica que los registros públicos funcionan correctamente y son consistentes criptográficamente hablando y, por otro, que pueden verificar la existencia de un certificado concreto en el registro CT. Los navegadores web acabarán implementando estas capacidades.

Los servidores web HTTPS pueden proporcionar información sobre CT a los clientes web en sus respuestas, durante el establecimiento de la sesión TLS, empleando tres métodos: Extensión del certificado digital X.509v3 (proporcionado por la CA), extensión de TLS (disponible por ejemplo en Apache mediante el módulo "mod_ssl_ct"⁴⁰) u OCSP Stapling⁴¹ (también si es soportado por la CA).

El método más sencillo para proporcionar información de CT desde el servidor web es haciendo uso de una CA que emita los certificados digitales incluyendo dentro de los mismos una extensión X.509 que incorpora el *timestamp* de la firma o comprobante de la incorporación del certificado en el registro de CT (SCT, Signed Certificate Timestamp)⁴², evidenciando su existencia en el registro de CT. Sin embargo, es preferible hacer uso de una nueva extensión de TLS creada para este propósito o de OCSP Stapling (si la CA proporciona soporte para incluir SCT en la respuesta OCSP), ya que ambos métodos permiten incluir nuevos valores de SCT para nuevos registros (añadiéndose a, o sustituyendo, los valores previos) sin necesidad de renovar el certificado digital.

Adicionalmente, en enero de 2017 Chrome propuso⁴³ la creación de una nueva cabecera HTTP, denominada "Expect-CT" [Ref - 16], que indica que el servidor web dispone de soporte de CT y que el cliente debería esperar recibir un valor de SCT válido. Al igual que con otras cabeceras HTTP de seguridad como HPKP (ver apartado "3.3.8. Restricción de los certificados digitales legítimos: HPKP") o CSP (ver apartado "3.7.6. Content Security Policy (CSP)"), "Expect-CT" puede ser utilizada de manera informativa generándose notificaciones desde los navegadores web (hacia la referencia especificada por la directiva "report-uri"), para detectar la ausencia de SCTs válidos, o de manera estricta, bloqueando las conexiones HTTPS que estén asociadas a un certificado digital que no es compatible con CT. Para ello, "Expect-CT" no hace uso de dos cabeceras HTTP diferentes, como ocurre en el caso de HPKP o CSP, sino que emplea la directiva opcional "enforce" dentro de la misma cabecera.

Dado que Chrome requerirá CT para todos los nuevos certificados en octubre de 2017, esta cabecera permite proteger también los certificados antiguos (emitidos antes de octubre de 2017) o certificados nuevos, pero que han sido emitidos por una CA maliciosa con fechas previas.

3.3.7 CAA (Certification Authority Authorization)

El estándar Certification Authority Authorization (CAA) [Ref- 17] establece un nuevo tipo de registro en el servicio de resolución de nombres DNS (RR, *resource record*, de tipo 257) que permite al propietario de un dominio especificar que una o más CAs tienen la autoridad para

⁴⁰ https://httpd.apache.org/docs/trunk/mod/mod_ssl_ct.html

⁴¹ <http://www.certificate-transparency.org/resources-for-site-owners>

⁴² <https://www.certificate-transparency.org/how-ct-works>

⁴³ <https://groups.google.com/a/chromium.org/forum/#!topic/blink-dev/tgn5R-58iek>

emitir certificados digitales para dicho dominio y que, por tanto, cualquier otra CA no está autorizada a emitirlos.

Por tanto, CAA vincula el servicio DNS con el proceso de emisión de los certificados digitales empleados, por ejemplo, en HTTPS, permitiendo hacer uso del servicio DNS como un mecanismo de validación externo asociado al protocolo HTTPS y, en concreto, a la gestión y emisión de certificados digitales.

CAA permite a una CA (bien intencionada) añadir controles (administrativos) adicionales para reducir el riesgo de emitir certificados digitales de manera no intencionada y contraria a lo estipulado por el propietario del dominio web.

En febrero de 2017 se lanzó un proceso de votación en el CAB Forum (Ballot 187) para establecer que el estándar CAA fuera de obligado cumplimiento para las CAs⁴⁴, debiendo éstas verificar y comprobar los registros CAA del DNS antes de emitir un certificado digital nuevo. El resultado de la votación ratificó que la comprobación de CAA es de obligado cumplimiento para las CAs⁴⁵, de manera efectiva desde el 8 de septiembre de 2017 [Ref - 18]. En caso de que la configuración de CAA no permita a una CA emitir certificados digitales para ese dominio, el proceso de emisión deberá ser interrumpido (y potencialmente revisado manualmente por el personal de la CA).

Estas comprobaciones introducen mejoras notables en la infraestructura global de clave pública (PKI) de Internet, compuesta por todas las CAs públicas de confianza, evitando que el ecosistema y la confianza en las CAs públicas sea tan débil como la más débil de las CAs.

Los registros CAA proporcionan granularidad para determinar las CAs asociadas al dominio ("issue"), las CAs asociadas a los certificados comodín ("issuewild"), ver apartado "3.3.3. Entidad asociada al certificado digital", y proporcionan a su vez capacidades de notificación en caso de errores o peticiones de certificados no válidas ("iodef"), mediante una URL o mediante una dirección de correo electrónico.

El formato de los registros CAA incluye el dominio asociado a los certificados digitales y que establece la restricción en el DNS, y el nombre del dominio de la CA autorizada a emitir certificados digitales para ese dominio⁴⁶:

dinosec.com.	IN	CAA	128	issue	"letsencrypt.org"
dinosec.com.	IN	CAA	128	issuewild	"letsencrypt.org"
dinosec.com.	IN	CAA	128	iodef	"mailto:info@dinosec.com"

El estándar CAA es suficientemente flexible y permite definir propiedades adicionales futuras mediante pares de nombre (de variable) y valor.

3.3.8 Restricción de los certificados digitales legítimos: HPKP

La cabecera de seguridad del protocolo HTTP conocida como HPKP o PKP (HTTP Public Key Pinning) [Ref - 19] permite al servidor web declarar que un certificado digital para dicho servidor

⁴⁴ <https://cabforum.org/pipermail/public/2017-February/009722.html>

⁴⁵ <https://cabforum.org/pipermail/public/2017-March/009988.html>

⁴⁶ El valor 128 añade un flag de criticidad para esa directiva DNS, lo que permitirá añadir nuevas semánticas en futuras versiones de CAA, e indicar que la directiva debe de ser entendida por la CA de cara a poder procesar los registros de CAA asociados y proceder a emitir el certificado digital.

web sólo debe ser considerado válido si está incluido en la lista específica de certificados (pines) listados en dicha cabecera. Como resultado, los navegadores y clientes web con soporte para HPKP no se conectarán al entorno web si el certificado digital recibido (o algún certificado de la cadena de certificación, como, por ejemplo, los de las CAs intermedias) no coincide con los pines recibidos.

HPKP es probablemente uno de los mecanismos de seguridad más estrictos disponibles en la actualidad, e introduce el riesgo de bloquear el acceso al entorno web si no es configurado correctamente, por lo que debe de ser implementado con cautela y evaluando detalladamente su implantación. Especialmente, debería ser utilizado si se considera que existe el riesgo de que el entorno web pueda ser suplantado mediante un certificado fraudulento o no legítimo. Para todas las medidas de protección sugeridas a lo largo del presente informe se recomienda evaluar el beneficio de seguridad obtenido frente a la dificultad o complejidad de implementación y gestión⁴⁷. Sin embargo, HPKP puede ser implementado sin ningún riesgo empleando únicamente sus capacidades de notificación (descritas posteriormente), lo que permite hacer uso de los navegadores y clientes web como sensores para la detección de ataques de suplantación en caso de identificar certificados digitales no legítimos.

El objetivo de HPKP es mitigar el riesgo asociado a una de las principales debilidades del ecosistema global de clave pública (PKI) de Internet, el hecho de que cualquier CA pública (existen cientos de ellas actualmente) puede generar un certificado digital válido para cualquier entorno web de Internet, y sin la autorización expresa de su propietario. HPKP previene, en el propio cliente web, ataques de tipo MitM (Man-in-the-Middle)⁴⁸ que hacen uso de certificados válidos pero no legítimos o autorizados, es decir, que hayan sido emitidos por alguna de las CAs públicamente reconocidas, aunque su emisión no haya sido autorizada por el propietario del servidor web. En realidad, HPKP permite eliminar la confianza existente por defecto en las CAs y en la infraestructura global de clave pública (PKI) de Internet, y confiar únicamente en un conjunto muy específico de certificados digitales (pines).

Esta medida de seguridad se ha diseñado a raíz del incremento de los incidentes de seguridad sobre las CAs durante los últimos años, con casos muy conocidos como el de DigiNotar o Comodo en el año 2011, o StartCom y WoSign posteriormente en el año 2016 (con múltiples violaciones de las reglas de juego de las CAs), y que han permitido a los atacantes (o a las CAs) generar certificados digitales para numerosos sitios web no legítimos, pero perfectamente válidos para todos los navegadores y clientes web.

La cabecera HPKP debe de ser incluida en todas y cada una de las respuestas HTTPS generadas por el servidor o aplicación web:

```
Public-Key-Pins: pin-sha256="<base64-A==>"; pin-sha256="<base64-B==>";  
...; max-age=5184000 [; includeSubDomains] [; report-uri="<URL>"]
```

NOTA: En el caso de HPKP, al igual que para CSP, es posible aplicar de manera estricta y efectiva la política a la vez que se reciben informes de violación de la misma (mediante "report-uri"). Esta opción es muy interesante para identificar ataques que hagan uso de certificados no legítimos.

⁴⁷ https://wiki.mozilla.org/Security/Guidelines/Web_Security

⁴⁸ Debido principalmente al compromiso de una CA, o a un comportamiento malintencionado o anómalo por parte de una CA.

Debe tenerse en cuenta que el navegador web refuerza el uso de HTTPS para un conjunto específico de certificados digitales durante el periodo indicado por la directiva "max-age" (especificado en segundos) de la cabecera HPKP. Se recomienda emplear un valor de "max-age" de 5184000 (60 días o 2 meses), tal como recomienda el RFC, con el objetivo de encontrar un equilibrio entre seguridad y disponibilidad.

Para minimizar el posible impacto de una configuración incorrecta al comenzar el despliegue de HPKP, se recomienda incrementar progresivamente el valor de "max-age", tal como se sugiere para la cabecera HSTS (ver apartado "3.7.4. Forzando el uso estricto (por defecto) de HTTPS: HSTS"). Simultáneamente, se deberían monitorizar exhaustivamente los accesos a la URL de informes indicada por la directiva "report-uri".

La directiva "report-uri" hace referencia a la dirección web dónde los navegadores y clientes web deben reportar errores en la validación de los pines, tanto en el modo estándar como en el modo de notificación de HPKP, empleando el formato definido por el RFC. Se recomienda emplear una referencia web HTTPS asociada a un servidor web distinto al que ha enviado la cabecera HPKP, ya que, en caso contrario, no será posible para el navegador web remitir el informe debido al error con el pin. Existe la posibilidad de hacer uso de un servidor web propio específico, en el mismo o en otro dominio, para recibir estos informes, o de hacer uso de un servicio de terceros para este propósito como, por ejemplo, "Report URI"^{49 50}.

Desde el punto de vista de seguridad, en el caso de especificar pines asociados a certificados comodín o a CAs raíz o intermedias, por simplicidad, HPKP debería de ser implementado para la totalidad del dominio, y no sólo para servidores web individuales. Para ello, se debe de hacer uso de la directiva "includeSubDomains".

Adicionalmente, al igual que para HSTS (ver apartado "3.7.4. Forzando el uso estricto (por defecto) de HTTPS: HSTS"), es importante resaltar que HPKP es un mecanismo de seguridad de tipo TOFU (Trust On First Use), es decir, que no es de aplicación la primera vez que el cliente web se conecta al servidor web. En el caso de HPKP, sin embargo, no se dispone de una lista precargada pública como ocurre con HSTS. Internamente los navegadores web sí disponen de una lista estática o conjunto de pines (static_spki_hashes), conocida como "Static Public Key Pinning" que aplica a algunos dominios asociados a las compañías vinculadas a los propios navegadores web (Google, Firefox, etc.) o, excepcionalmente, a otras grandes compañías y dominios incluidas por estas (Facebook, Twitter, etc.)⁵¹.

Adicionalmente, al igual que con otras cabeceras HTTP de seguridad como CSP (ver apartado "3.7.6. Content Security Policy (CSP)"), y de nuevo para minimizar el posible impacto de una configuración incorrecta al empezar el despliegue de HPKP, se recomienda comenzar haciendo uso de las capacidades de notificación de HPKP mediante la cabecera extendida "...-Report-Only" y la directiva "report-uri". Deberían hacerse pruebas inicialmente con esta cabecera para identificar el correcto funcionamiento de la política. Si un navegador o cliente web, con soporte para HPKP Report-Only⁵² (como por ejemplo Chrome 46 u Opera 33⁵³), recibe un certificado

⁴⁹ <https://report-uri.io>

⁵⁰ <https://scotthelme.co.uk/report-uri-io-new-features-new-reports-new-tools/>

⁵¹ <https://community.qualys.com/thread/17152-what-is-static-public-key-pinning>

⁵² <https://developer.mozilla.org/es/docs/Web/HTTP/Headers/Public-Key-Pins-Report-Only>

⁵³ No soportada aún en Firefox: https://bugzilla.mozilla.org/show_bug.cgi?id=1091177

digital no legítimo, es decir, que no está incluido en los pines de la cabecera HPKP, no bloqueará el tráfico HTTPS (tal como ocurre con la cabecera HPKP estándar, que sí aplica la política HPKP sin excepción), pero generará un informe dirigido a la URL indicada en dicha cabecera por la directiva "report-uri". En este escenario, de manera efectiva, los clientes web son empleados como sensores para la detección de ataques de suplantación del entorno web:

```
Public-Key-Pins-Report-Only: pin-sha256="<base64-A==>"; pin-  
sha256="<base64-B==>"; report-uri="https://dominioinformes.es/hpkp-  
report"
```

NOTA: En caso de usar "...-Report-Only" no es necesario hacer uso de la directiva "max-age".

Se recomienda comenzar haciendo uso de la cabecera HPKP sólo en un recurso web concreto, hasta verificar su correcta configuración, con el objetivo de evitar recibir multitud de informes de violación o de errores en la validación de los pines.

Dos elementos fundamentales a considerar a la hora de implementar HPKP (descritos a continuación) son como generar los pines para los certificados digitales existentes, y sobre qué certificado digital de la cadena de certificación actual se debería asociar el pin reflejado en la cabecera HPKP.

A la hora de crear los pines, o referencias a los certificados digitales legítimos, existen teóricamente dos opciones: referenciar directamente al certificado digital, o a la clave pública incluida en el certificado digital. HPKP referencia la clave pública, en concreto el campo "Subject Public Key Info" (SPKI) de los certificados digitales X.509, para evitar depender de la renovación, y otras tareas de gestión, de los certificados digitales. La estructura SPKI incluye la clave pública junto a ciertos metadatos, como el algoritmo criptográfico empleado para su generación (por ejemplo, RSA).

La verificación de HPKP en los navegadores o clientes web requiere, por tanto, que la cadena de certificación actual presentada al navegador o cliente web por el entorno, servidor o aplicación web deba de incluir al menos una estructura SPKI cuyo *fingerprint* (o huella) coincida con el *fingerprint* de uno de los pines existentes en la cabecera HPKP.

La cabecera HPKP debe de incluir, como mínimo, dos pines (reflejados anteriormente como A y B): uno principal, vinculado al valor SPKI de uno de los certificados digitales de la cadena de certificación actual, y uno de *backup* o respaldo, que no debe de estar en la cadena de certificación actual. El pin de *backup* puede ser usado, por ejemplo, para renovar la clave criptográfica (pública y privada) del servidor, y su certificado digital asociado, tras un incidente de seguridad donde se comprometa el pin principal. Adicionalmente es posible configurar más pines en la cabecera HPKP, tanto principales como de *backup*.

El valor reflejado por la cabecera HPKP en la directiva "pin-sha256", asociado al valor del pin, corresponde a un hash SHA-256 del valor del campo SPKI en base64. La presentación "HTTPS: TLS, no SSL" [Ref - 2] proporciona los detalles necesarios para generar los pines, tanto el principal como los de *backup* o respaldo, mediante OpenSSL.

Desde el punto de vista de la gestión y renovación de las claves criptográficas (ver apartado "3.2. Generación de las claves criptográficas"), debe tenerse en cuenta que una práctica común es renovar las claves criptográficas cada vez que se renueva el certificado digital asociado (ver apartado "3.3.5. Renovación de los certificados digitales", como ocurre por defecto con la CA

Let's Encrypt). Sin embargo, debido a que los pines de HPKP están asociados a la clave criptográfica (o clave pública), sería necesario actualizarlos si la clave es renovada. En este caso, se recomienda no renovar la clave criptográfica durante el proceso estándar de renovación del certificado digital, pudiendo así mantener el valor de los pines existente.

A la hora de asociar el pin reflejado en la cabecera HPKP a un certificado digital de la cadena de certificación actual, existen varias posibilidades, siendo la más restrictiva la que prefiere crear un pin para la clave pública (y, en consecuencia, el certificado digital final) del servidor web, y la más permisiva la que asocia el pin a la clave pública de la CA raíz. Alternativamente, es posible asociar el pin a la clave pública de cualquiera de las CAs intermedias existentes en la cadena de certificación actual. Se debe encontrar el equilibrio entre el nivel de seguridad (o restricción) que se desea establecer mediante HPKP, los posibles riesgos de accesibilidad, y las tareas de gestión asociadas a las claves criptográficas. Las consideraciones a tener en cuenta aplican tanto al pin (o pines) principales, como a los pines de *backup*.

El escenario más seguro y restrictivo es el que asocia el pin a la clave pública del propio servidor web. Sin embargo, debe de ser gestionado cuidadosamente para modificar los pines adecuadamente, anticipándose al cambio y/o renovación de la clave, e incluir pines para todas las claves asociadas al servidor web (por ejemplo, si se hace uso de claves RSA y ECDSA simultáneamente; ver apartado "3.2. Generación de las claves criptográficas").

El escenario más flexible y permisivo es el que asocia el pin a la clave pública de una CA intermedia, o más aún, de la CA raíz. Sin embargo, también debe de ser gestionado cuidadosamente para poder mantener los pines actuales (si no se desea modificarlos), especialmente durante el proceso de renovación del certificado digital del servidor web, ya que las CAs habitualmente disponen de varias CAs raíz e intermedias. Por tanto, es importante asegurarse de que el nuevo certificado digital será emitido por la misma CA. Asimismo, las CAs establecen relaciones de confianza entre sí, firmando algunos certificados digitales de manera cruzada, lo que implica que puede haber varias cadenas de certificación válidas para un mismo certificado digital. En este caso es necesario disponer de pines para todas y cada una de las cadenas de certificación válidas posibles.

Intentando buscar un equilibrio entre ambos escenarios, y las ventajas de cada uno de ellos, una opción es crear el pin para la primera CA intermedia, es decir, aquella que ha emitido directamente el certificado digital actual del servidor web.

Por otro lado, el pin (o pines) de *backup* pueden estar asociados a la misma CA, o aún más recomendable, a otra CA diferente, en caso de que la CA actual se vea comprometida. Se recomienda también mantener *offline* la clave privada y el certificado digital asociados al pin de *backup*, es decir, en una ubicación segura donde sea más difícil que estos sean comprometidos.

Finalmente, se debe tener en cuenta que HPKP no ofrece protección frente a la manipulación local de las CAs en los navegadores y clientes web, como por ejemplo cuando un usuario importa una nueva CA raíz. Los certificados digitales asociados a dicha CA raíz serán considerados como válidos, independientemente de si coinciden o no con los pines de la cabecera HPKP, escenario que permite el uso de proxies web locales y de soluciones empresariales de inspección e interceptación del tráfico HTTPS (ver apartado "3.7.9. Inspección del tráfico HTTPS").

3.3.9 Recomendaciones para la gestión de los certificados digitales

El resumen de recomendaciones para la gestión de los certificados digitales, donde es necesario tener en cuenta múltiples aspectos, incluye:

- Para los servidores web expuestos públicamente es necesario disponer de un certificado digital emitido por una CA pública de confianza y ampliamente reconocida, no siendo válido hacer uso de certificados digitales autofirmados, ni de CAs privadas.
- Evaluar y seleccionar el tipo adecuado de certificado digital (DV, OV o EV) que se desea obtener para cada servidor o aplicación web.
- Evaluar y seleccionar la CA que emitirá el certificado empleando múltiples criterios objetivos, como su reputación, la funcionalidad que ofrece, su adopción de nuevos estándares, los mecanismos de revocación de certificados que implementa, etc.
- Seleccionar el periodo de renovación del certificado digital y renovar los certificados periódicamente y siempre antes de su vencimiento.
- Hacer uso de certificados digitales firmados mediante SHA-256 (o superior). En ningún caso hacer uso de firmas basadas en SHA-1 o MD5.
- Identificar y configurar adecuadamente (en el orden correcto) la cadena completa de certificación (o de certificados) asociada al certificado digital, desde la CA raíz, pasando por todas las CAs intermedias, hasta el certificado final.
- Seleccionar el tipo adecuado de certificado digital respecto al nombre de la entidad representada por éste, es decir, referencia directa o única, lista de nombres, certificado comodín, o multi-dominio.
- El certificado digital debería incluir tanto el nombre del servidor web representado ("www", o la lista de servidores web, o el comodín) como de su dominio.
- El nombre de la entidad debe estar reflejado en el certificado digital a través del campo "SAN" (Subject Alternative Name, extensión "subjectAltName") en lugar de mediante el "CN" (Common Name).
- Asegurar la validez del certificado digital en todo momento, incluyendo el nombre de la entidad, periodo de validez, CA emisora del certificado, revocación del certificado y propósito.
- Asociar el certificado digital a una CA que haga uso de Certificate Transparency (CT) y confirmar que el certificado ha sido registrado en los logs públicos de CT.
- Monitorizar los logs o registros de CT para identificar la emisión de certificados digitales de dominios propios de manera ilegítima por parte de otras CAs, sin la autorización del propietario del dominio. Complementariamente, hacer uso de servicios de monitorización de CT para ser informado de cambios en los certificados digitales asociados a un dominio.
- Proporcionar información de registro en CT (mediante SCT) en las respuestas HTTPS del servidor web hacia los clientes web mediante una extensión del certificado digital X.509v3, (o preferiblemente) mediante una extensión específica de TLS o empleando OCSP Stapling.
- Se recomienda valorar la utilización de la cabecera HTTP "Expect-CT" para declarar la necesidad de usar certificados digitales reconocidos en CT. En su defecto, hacer uso de las capacidades de notificación de "Expect-CT" para identificar el uso de certificados digitales no válidos desde el punto de vista de CT.

- Configurar en el servicio de resolución de nombres DNS los registros CAA correspondientes a la lista de CAs asociadas a los certificados digitales del dominio.
- Asociar el certificado digital a una CA que haga uso de CAA (Certification Authority Authorization).
- Hacer uso de las capacidades de notificación de CAA para identificar errores o peticiones de certificados no válidas.
- Hacer uso de HPKP (HTTP Public Key Pinning) para establecer los certificados digitales reconocidos asociados al entorno, servidor o aplicación web. En su defecto, hacer uso de las capacidades de notificación de HPKP para identificar el uso de certificados digitales válidos, pero no legítimos o autorizados, contra el entorno web.
- En el caso de hacer uso de HPKP, se recomienda emplear un valor de "max-age" de 5184000 (60 días o 2 meses), empezar haciendo un uso limitado de la cabecera HPKP en un recurso concreto, e ir incrementando progresivamente el valor de "max-age".
- Si se está haciendo uso de la directiva "report-uri" de HPKP, se debe emplear una referencia web HTTPS asociada a un servidor web distinto al que ha enviado la cabecera HPKP.
- HPKP debería de ser implementado para la totalidad del dominio mediante la directiva "includeSubDomains".
- HPKP debe de incluir, como mínimo, dos pines: uno principal, asociado a uno de los certificados digitales de la cadena de certificación actual, y uno de backup, que no debe de estar en la cadena de certificación actual.
- Evaluar y seleccionar el certificado digital de la cadena de certificación actual que será utilizado para la generación de los pines de HPKP, pudiendo elegir un escenario más restrictivo (servidor web final), intermedio (CAs intermedias) o más permisivo (CA raíz).
- Seleccionar la CA vinculada a los pines de backup de HPKP y proteger convenientemente la clave privada asociada a estos pines.

3.4 Protocolos SSL y TLS

El presente apartado profundiza en todas las variantes y en las funcionalidades adicionales asociadas al protocolo obsoleto SSL (Secure Sockets Layer) y a su sustituto, TLS (Transport Layer Security).

3.4.1 Versiones del protocolo SSL y TLS

La selección de los protocolos SSL y TLS, y de las versiones específicas de estos, a utilizar por el servidor o aplicación web estará influenciada por los requisitos de seguridad e interoperabilidad del entorno web, es decir, cuáles son los navegadores y clientes web a los que se les quiere dar soporte y permitir el acceso a dicho entorno web con el mayor nivel de seguridad posible.

Las diferentes versiones del protocolo TLS han ido introduciendo mejoras de seguridad. Por ejemplo, TLS 1.0 añadió el uso de funciones pseudo-aleatorias (PRF, Pseudo-Random Function) basadas en HMAC, empleándose también PRF para generar el secreto maestro. Las capacidades de *padding* también fueron mejoradas en TLS 1.0 respecto a SSL 3.0. TLS 1.1 principalmente añadió mejoras de seguridad sobre TLS 1.0, como el uso de cifrado CBC con vectores de

inicialización (IVs) explícitos en cada registro TLS, en lugar de IVs predecibles, y de nuevo, mejoras en *padding*, e incluyó las extensiones TLS (RFC 3546⁵⁴). TLS 1.2 añadió cifrado autenticado (AEAD), soporte para HMAC-SHA256 (siendo éste empleado para la función PRF estándar y así para los algoritmos de firma, aunque se dispone de flexibilidad para utilizar otras funciones PRF), una extensión para que ambos extremos puedan indicar y negociar que algoritmos de firma y *hashing* soportan, y eliminó el uso de algoritmos criptográficos débiles.

Se desaconseja hacer uso del protocolo SSL, ya que se considera completamente vulnerable desde su versión 2.0, debido a vulnerabilidades previas, pero especialmente tras el descubrimiento de la vulnerabilidad DROWN en marzo de 2016, hasta su última versión 3.0, tras el descubrimiento de la vulnerabilidad POODLE en octubre de 2014 (ver apartado "3.8. Vulnerabilidades en HTTPS").

Por tanto, en el caso más tolerante desde el punto de vista de la interoperabilidad, el entorno web sólo debería soportar las diferentes versiones del protocolo TLS. Sin embargo, debido a las debilidades existentes en la versión 1.0 de este protocolo, como (entre otras) BEAST, publicada en junio de 2011 (ver apartado "3.8. Vulnerabilidades en HTTPS"), se recomienda buscar un equilibrio entre seguridad e interoperabilidad y soportar únicamente TLS 1.1 y TLS 1.2 (RFC 5246⁵⁵), junto a futuras versiones del protocolo, como TLS 1.3 [Ref - 22]⁵⁶.

Como referencia, todos los sitios web que aceptan pagos con tarjeta de crédito y que, por tanto, están regulados bajo PCI (Payment Card Industry), deben de eliminar el soporte de TLS 1.0 antes de julio de 2018, fecha que inicialmente estaba fijada para julio de 2016⁵⁷ (lo que denota la obsolescencia de esta versión del protocolo, que fue publicada en enero de 1999⁵⁸).

TLS 1.2 es la única versión del protocolo que soporta algoritmos criptográficos modernos, como los asociados al cifrado autenticado (Authenticated Encryption, AE), o AEAD (ver apartado "3.5.5. Fortaleza de los algoritmos y *suites* criptográficas").

La versión 1.3 del protocolo TLS [Ref - 22] está siendo ratificada durante el año 2017, disponiéndose ya de soporte en Chrome 56 y Firefox 52, u OpenSSL 1.1.1-dev (versión de desarrollo).

3.4.2 SNI (Server Name Indication)

SNI (Server Name Indication) [Ref - 8] es una extensión del protocolo TLS que permite a los clientes web indicar el nombre del servidor web (hostname) al que desean conectarse durante el proceso inicial de establecimiento de la conexión HTTPS empleando TLS (*handshake*).

Las extensiones TLS son un mecanismo disponible para extender la funcionalidad de TLS sin tener que modificar el protocolo. Las extensiones tienen implicaciones relevantes positivas desde el punto de vista de seguridad, y son empleadas para funcionalidades como SNI, OCSP Stapling ("3.6.3. OCSP Stapling"), los tickets de sesión ("3.7.2.2. TLS session resumption o

⁵⁴ <https://tools.ietf.org/html/rfc3546>

⁵⁵ <https://tools.ietf.org/html/rfc5246>

⁵⁶ <https://datatracker.ietf.org/wg/tls/documents/>

⁵⁷ <https://blog.pcisecuritystandards.org/migrating-from-ssl-and-early-tls>

⁵⁸ <https://tools.ietf.org/html/rfc2246>

caching, y TLS session tickets"), o los mecanismos de renegociación segura ("3.4.3. Renegociación"), entre otros.

Mediante el uso de SNI es posible hacer uso de HTTPS en entornos con servidores web o *hosts* virtuales (HTTP/1.1), es decir, cuando diferentes servidores web (hostnames) coexisten en un mismo servidor físico y hacen uso de la misma dirección IP. SNI permite por tanto disponer de diferentes certificados digitales para cada uno de los servidores web, todos ellos proporcionados desde la misma dirección IP y puerto HTTPS (TCP/443).

Si no se dispone de soporte para SNI, sólo es posible hacer uso de un servidor web (o un grupo muy reducido de servidores) por dirección IP, al tener asociado sólo un único certificado digital. Se podría compartir dicho certificado digital entre varios servidores web asociados a esa dirección IP, opción que sólo es recomendada si se trata de un certificado digital (por ejemplo, comodín) que sea válido para todos los diferentes servidores web de un mismo dominio disponibles en dicha dirección IP.

Debido a que SNI únicamente no está soportado en versiones antiguas de Android $\leq 2.3.x$, Java ≤ 6 y en Internet Explorer en Windows XP (cuyo soporte finalizó en 2014), desde el punto de vista de su interoperabilidad, es aceptable su utilización.

El único inconveniente de SNI es que el nombre del servidor web es desvelado en claro por el cliente web en el tráfico de red antes de establecerse el túnel cifrado mediante HTTPS, aunque en la mayoría de los escenarios esto no es un inconveniente, ya que el nombre del servidor web es conocido públicamente o se desvela mediante el servicio de DNS.

En el caso de disponer de varios servidores web o *hosts* virtuales vinculados a una misma dirección IP, se recomienda disponer de soporte para SNI en la implementación de HTTPS.

3.4.3 Renegociación

La implementación del protocolo TLS proporciona mecanismos de renegociación, que permiten tanto al cliente como al servidor renegociar los detalles y parámetros de una sesión TLS ya establecida, como, por ejemplo, al hacer uso de certificados digitales cliente, como los asociados al DNI electrónico (DNIe) o a los certificados de usuario.

Los mecanismos de renegociación de TLS eran inseguros, facilitando la realización de ataques de interceptación (MitM) y de denegación de servicio (DoS, Denial of Service) sobre el servidor web, ya que la operación de renegociación requiere disponer de más recursos en el servidor que en el cliente.

Se recomienda hacer uso de una implementación TLS que haga uso de los mecanismos de renegociación seguros definidos en el RFC 5746⁵⁹ y, preferiblemente, que la renegociación sea siempre iniciada por el servidor. En caso de que no se haga uso de certificados digitales cliente, se recomienda deshabilitar los mecanismos de renegociación completamente. Al menos, se deberían deshabilitar los mecanismos que permiten que la renegociación sea iniciada por el cliente. Por ejemplo, Apache no proporciona soporte para renegociación iniciada por el cliente desde la versión 2.2.15.

⁵⁹ <https://tools.ietf.org/html/rfc5746>

3.4.4 Degradación de la versión del protocolo TLS

Una de las mayores debilidades asociadas a HTTPS es la posibilidad de que, durante el proceso de negociación asociado al establecimiento de la sesión cifrada, un potencial atacante pueda forzar al cliente y/o al servidor a hacer uso de una versión previa (*downgrade*), y vulnerable, de los protocolos SSL o TLS. Esta técnica fue utilizada, por ejemplo, en el ataque POODLE para degradar la conexión a la versión 3.0 de SSL, incluso desde versiones más seguras del protocolo, como TLS 1.2.

Para evitar este tipo de escenarios de ataque, se recomienda implementar en el servidor web TLS Fallback Signaling Cipher Suite Value (SCSV), RFC 7507⁶⁰, una extensión del protocolo TLS que previene la realización de ataques de degradación del protocolo a emplear. Esta extensión está soportada desde Chrome 33⁶¹ y Firefox 35⁶², y a partir de la versión 1.0.1j de OpenSSL.

3.4.5 HTTP/2

En la actualidad, el mayor impacto en el rendimiento de TLS (ver apartado "3.7.2. Rendimiento") no suele estar asociado a operaciones criptográficamente costosas desde el punto de vista del consumo de CPU, sino a la latencia de la red.

Por este motivo, y para mejorar el rendimiento del entorno web, se recomienda establecer el número mínimo posible de conexiones HTTPS (y, por tanto, TCP). Para ello se puede incrementar la duración en el tiempo de las sesiones TCP (mediante el parámetro TCP keep alive) o hacer uso de HTTP/2, protocolo que mantiene una única sesión TCP en el servidor web por cada cliente web. HTTP/2 está soportado en la actualidad por la mayoría de navegadores web modernos⁶³.

Adicionalmente, la distancia entre el cliente y el servidor web es un factor crítico de cara a la latencia de las comunicaciones. Esta distancia se puede optimizar mediante el uso de CDNs (Content Delivery Networks) con capacidades HTTPS, aunque se recomienda evaluar los detalles de implementación de HTTPS de este tipo de servicios, y las diferentes modalidades que ofrecen⁶⁴.

3.4.6 Recomendaciones de los protocolos SSL y TLS

El resumen de recomendaciones de los protocolos SSL y TLS incluye:

- Deshabilitar completamente el soporte para el protocolo SSL 2.0 y SSL 3.0.
- Hacer uso únicamente de las versiones 1.1 y 1.2 del protocolo TLS.
- Llevar a cabo las acciones necesarias para soportar TLS 1.3 tan pronto se ratifique esta nueva versión del protocolo a lo largo del año 2017.
- Disponer de soporte para SNI en la implementación de HTTPS en el caso de disponer de varios servidores web o hosts virtuales vinculados a una misma dirección IP.

⁶⁰ <https://tools.ietf.org/html/rfc7507>

⁶¹ <https://www.imperialviolet.org/2014/02/27/tlssymmetriccrypto.html>

⁶² <https://blog.mozilla.org/security/2014/10/14/the-poodle-attack-and-the-end-of-ssl-3-0/>

⁶³ <http://caniuse.com/#feat=http2>

⁶⁴ <https://www.troyhunt.com/cloudflare-ssl-and-unhealthy-security-absolutism/>

- Deshabilitar los mecanismos de renegociación de TLS, especialmente, la renegociación que puede ser iniciada por el cliente. En caso de ser necesarias estas capacidades, habilitar la renegociación iniciada por el servidor y hacer uso de los mecanismos de renegociación segura de TLS.
- Habilitar la extensión del protocolo TLS "TLS Fallback Signaling Cipher Suite Value" (SCSV) para prevenir ataques de degradación (downgrade) del protocolo TLS.
- Evaluar la posibilidad de implementar HTTP/2 en el servidor web, coexistiendo con la versión HTTP/1.1 del protocolo, debido a las ventajas de rendimiento que presenta al utilizar HTTPS.

3.5 Algoritmos y suites criptográficas

La configuración de los algoritmos y *suites* criptográficas soportadas por el entorno web basado en HTTPS es uno de los elementos más complejos a tener en cuenta en la implementación de HTTPS, ya que siempre debe de buscar el equilibrio entre el nivel de seguridad proporcionado (en base a la fortaleza de los algoritmos y protocolos criptográficos empleados, y de las novedades en la industria y expectativas a medio y largo plazo de estos), el rendimiento del entorno web, y la interoperabilidad con múltiples navegadores y clientes web.

3.5.1 Forward secrecy

El intercambio de claves basado en RSA no proporciona forward secrecy (también referenciado como PFS, Perfect Forward Secrecy), es decir, una propiedad criptográfica cuyo objetivo es evitar que el tráfico cifrado capturado en el pasado pueda ser descifrado posteriormente si se comprometen las claves en el futuro. Expresado de otro modo, el compromiso u obtención de las claves de cifrado actuales no permite comprometer u obtener las claves de cifrado utilizadas en el pasado. Básicamente, forward secrecy evita que el tráfico cifrado dependa de la clave privada. En su lugar, deberían usarse algoritmos basados en Diffie-Hellman (DH), como por ejemplo DHE (DH Ephemeral, o EDH) o ECDHE (Elliptic Curve DHE, o ECDH), que hacen uso de parámetros DH efímeros (y que dan lugar a claves efímeras). Desde el punto de vista de seguridad y rendimiento, es preferible hacer uso de ECDHE frente a DHE.

Forward secrecy asegura que cada sesión TLS de cada cliente, incluso hacia el mismo servidor, hace uso de claves criptográficas diferentes, mientras que sin forward secrecy (por ejemplo, al usar RSA), las claves de todas las sesiones dependen de la clave privada del servidor web, y podrían ser descifradas disponiendo de esta.

RSA, sin embargo, puede seguir siendo utilizado para la generación de las claves asociadas a los certificados digitales. Es importante diferenciar entre los algoritmos y claves criptográficas empleadas para la identificación de los extremos y la generación de los certificados digitales asociados (ver apartado "3.2. Generación de las claves criptográficas"), y los algoritmos empleados para el intercambio de claves durante el establecimiento de una sesión TLS (ver apartado "3.5.2. Intercambio de claves robusto"), dónde debe de asegurarse el uso de forward secrecy.

3.5.2 Intercambio de claves robusto

El apartado "3.5.1. Forward secrecy" describe en detalle la principal propiedad necesaria durante el intercambio de claves al establecerse una sesión TLS: forward secrecy. Para ello, no se deben

hacer uso de *suites* criptográficas que hagan uso de un intercambio de claves basado en RSA, sino que se debe emplear la variante de curva elíptica (ECDHE) o, en su defecto, la variante clásica del intercambio de claves mediante Diffie-Hellman (DHE), empleando claves efímeras (ephemeral). En realidad, RSA se emplea para transportar la clave maestra (o de sesión), mientras que (EC)DHE se emplea para negociar dicha clave maestra, empleando forward secrecy.

Se dispone por tanto de tres algoritmos de intercambio de claves (en orden inverso de prioridad): RSA, DHE y ECDHE. En resumen, se debería realizar el intercambio de claves mediante ECDHE con todos los navegadores y clientes web modernos, y soportar alternativamente (por compatibilidad) DHE para hacer uso de forward secrecy con navegadores y clientes web más antiguos (como Android < 3.0, Java < 7 u OpenSSL < 1.0.0).

La nomenclatura empleada para los algoritmos de intercambio de claves (ver apartado "3.5.5. Fortaleza de los algoritmos y *suites* criptográficas") incluye el algoritmo para el intercambio de clave (RSA, DHE y ECDHE) y el método de autenticación (en base al tipo de claves empleadas para generar los certificados digitales: RSA o ECDSA). Las combinaciones más comunes, en orden inverso de prioridad, son:

- RSA: Intercambio de clave RSA con autenticación RSA
- DHE_RSA: Intercambio de clave DH efímero con autenticación RSA.
- ECDHE_RSA: Intercambio de clave ECDH efímero con autenticación RSA.
- ECDHE_ECDSA: Intercambio de clave ECDH efímero con autenticación ECDSA.

NOTA: Adicionalmente, se dispone recientemente en TLS de suites criptográficas que hacen uso de criptografía post-cuántica (CECPQ1) para el intercambio de claves⁶⁵, que combinan el algoritmo NewHope con las curvas elípticas X25519.

Tras el intercambio de claves entre el cliente y el servidor, independientemente del método empleado, ambos extremos dispondrán de un secreto o clave maestra precompartida (*premaster secret* o *premaster key*). Tras aplicar el algoritmo de intercambio de claves seleccionado, y debido a que cada algoritmo puede manejar longitudes de claves diferentes, el valor de salida es procesado mediante una función pseudo-aleatoria (PRF, Pseudo-Random Function) para obtener siempre un secreto maestro de 48 bytes (384 bits).

En TLS, el intercambio de claves va ligado al proceso de autenticación, empleándose la clave pública (RSA o ECDSA), tanto para verificar que se está estableciendo una comunicación con la entidad deseada, como para la derivación del secreto maestro a través del algoritmo de intercambio de claves elegido. En el caso de emplearse RSA, el cliente genera el secreto maestro y lo envía cifrado con la clave pública del servidor; sólo éste podrá descifrarlo con la correspondiente clave privada, autenticándose implícitamente. En el caso de emplearse DHE o ECDHE, los parámetros DH enviados por el servidor están firmados con su clave privada, de forma que el cliente pueda verificar la firma con la clave pública correspondiente, y confirmar que está comunicándose con el servidor esperado (autenticándolo).

⁶⁵ <https://www.imperialviolet.org/2016/11/28/cecpq1.html>

Como recomendación general, en el caso de ECDHE se deberían de utilizar algoritmos de intercambio de claves de 256 bits y, en el caso de DHE, de 2.048 bits (ver apartado "3.5.3. Fortaleza de los parámetros Diffie-Hellman (DH)").

Se recomienda por tanto hacer uso de ECDHE empleando, por ejemplo, la curva "secp256r1" (denominada "prime256v1" en OpenSSL y también conocida como P-256, y ampliamente soportada por los clientes web), que proporciona 128 bits de seguridad para el intercambio de claves. Alternativamente se puede hacer uso de la curva "secp384r1" (también conocida como P-384 y también generalmente soportada, RFC 4492⁶⁶), pero cuyo rendimiento es inferior respecto a P-256 y tampoco proporciona una mejora de seguridad tan significativa como para justificar su uso. Alternativamente se podría hacer uso de la curva "secp521r1", pero cuyo soporte está más limitado.

Estas dos primeras curvas, definidas por NIST, deberían ser utilizadas con cautela, ya que no está claramente establecido como se han seleccionado sus parámetros y, por tanto, podrían presentar debilidades sólo conocidas por sus diseñadores. Alternativamente, se han popularizado recientemente dos curvas estándar conocidas como Curve25519 (X2559) y Curve448 (X448), incluso para su utilización en TLS^{67 68}.

Alternativamente, para aquellos casos en los que no es posible para el cliente web hacer uso de ECDHE, se recomienda proporcionar soporte para DHE empleando parámetros robustos de 2.048 bits (ver apartado "3.5.3. Fortaleza de los parámetros Diffie-Hellman (DH)").

3.5.3 Fortaleza de los parámetros Diffie-Hellman (DH)

La longitud o el tamaño de los parámetros Diffie-Hellman (DH) debería ser de al menos 2.048 bits y, por otro lado, se recomienda que coincida con la longitud o fortaleza de la clave privada correspondiente.

Algunas incompatibilidades de clientes web conocidas incluyen Java 6, que sólo soporta parámetros DH de 1.024 bits. Este también es el caso de Java 7, pero no siendo conflictivo desde el punto de vista de seguridad, ya que también soporta ECDHE.

En cualquier caso, no se recomienda hacer uso de grupos DH conocidos o pre-configurados (de 1.024 bits), como por ejemplo los recomendados en el RFC 3526⁶⁹, o de parámetros DH de 768 bits o menos, especialmente, tras el descubrimiento de la vulnerabilidad Logjam en enero de 2015 (ver apartado "3.8. Vulnerabilidades en HTTPS"). Asimismo, las implementaciones de HTTPS deben de ser muy cuidadosas en la selección de los parámetros DH, ya que como demostró la vulnerabilidad de triple handshake en 2014 (ver apartado "3.8. Vulnerabilidades en HTTPS"), era incluso posible para un servidor malicioso seleccionar parámetros DH inseguros que ni siquiera eran números primos.

Hasta la introducción del RFC 7919⁷⁰ (agosto 2016), los clientes no podían especificar sus requisitos de seguridad respecto a los parámetros DH durante el proceso de establecimiento y

⁶⁶ <https://tools.ietf.org/html/rfc4492>

⁶⁷ <https://tools.ietf.org/html/draft-ietf-tls-curve25519-01>

⁶⁸ <https://tools.ietf.org/html/draft-ietf-tls-rfc4492bis-15>

⁶⁹ <https://tools.ietf.org/html/rfc3526>

⁷⁰ <https://tools.ietf.org/html/rfc7919>

negociación de la sesión TLS, debiendo aceptar los proporcionados por el servidor. En base al RFC 7919 se recomienda, en lugar de usar grupos DH pre-configurados o de generar grupos propios con "openssl dhparam", hacer uso de los grupos DH pre-definidos en dicho estándar, ya que éstos han sido auditados desde el punto de vista de seguridad: ffdhe2048, ffdhe3072 o ffdhe4096⁷¹.

3.5.4 Preferencia de las *suites* criptográficas

Una de las funcionalidades fundamentales del protocolo TLS es la capacidad del cliente y el servidor HTTPS para negociar, durante el establecimiento de la sesión TLS (o TLS *handshake*), el conjunto de *suites* criptográficas que podrían ser utilizadas por ambas partes, y llegar a un acuerdo o consenso de cuál es la mejor opción disponible en función de las prioridades o preferencias de los dos extremos. Esta negociación permite que el servidor o aplicación web siempre intente hacer uso de la opción más segura disponible con cada uno de los clientes web que se conectan a él.

Por tanto, se recomienda configurar el servidor web indicando en primer lugar las *suites* criptográficas más robustas, listadas en orden descendente de preferencia según el nivel de seguridad que ofrecen.

3.5.5 Fortaleza de los algoritmos y *suites* criptográficas

Las *suites* criptográficas empleadas por TLS definen como se llevarán a cabo todas las comunicaciones entre cliente y servidor mediante HTTPS (empleando múltiples componentes o algoritmos criptográficos), incluyendo el proceso de autenticación para confirmar que la comunicación se realiza con el servidor o aplicación web legítimo, hasta todo el intercambio de datos cifrado posterior.

Si alguno de los algoritmos criptográficos empleados es débil o inseguro, debe de ser reemplazado por uno más moderno. Los siguientes algoritmos criptográficos no deberían ser utilizados en la actualidad:

- El cifrador⁷² (y mecanismo de autenticación) NULL, ya que no proporciona capacidades de cifrado (ni de autenticación).
- Las *suites* basadas en DH anónimo (ADH, Anonymous Diffie-Hellman) ya que no proporcionan capacidades de autenticación.
- Las *suites* criptográficas con algoritmos de cifrado débiles basados en claves con fortaleza criptológica inferior a 100 bits.
- Las *suites* criptográficas con algoritmos de cifrado permitidos para su exportación (ver ataque FREAK en el apartado "3.8. Vulnerabilidades en HTTPS").
- El algoritmo de cifrado RC4, ya que es inseguro.
- El algoritmo de cifrado DES, ya que es inseguro y lento.
- Los algoritmos de *hashing* MD5, RIPMED-160 y SHA-1, ya que son inseguros o débiles.

⁷¹ https://wiki.mozilla.org/Security/Server_Side_TLS

⁷² El término 'cifrador' es empleado en ocasiones para abreviar 'algoritmo de cifrado'.

TLS podría ser considerado como un *framework* que permite negociar y hacer uso de diferentes protocolos criptográficos, seleccionando los componentes y parámetros que definirán el tipo de seguridad a emplear. Las suites criptográficas empleadas por TLS definen en su nombre un conjunto de algoritmos criptográficos asociados a los diferentes componentes y fases del protocolo TLS. Entre ellos, se especifica (en el siguiente orden, como se muestra en la siguiente figura [Ref - 10]) el método de intercambio de claves, el método (o claves, y certificados) de autenticación, el algoritmo de cifrado simétrico y la longitud de las claves de cifrado a emplear, junto a su modo de uso (en caso de existir varios) y el algoritmo de *hashing* para el MAC (o el algoritmo de la función pseudo-aleatoria, PRF, empleado por ejemplo en los algoritmos de tipo AEAD, que no hacen uso de un MAC a nivel de TLS):

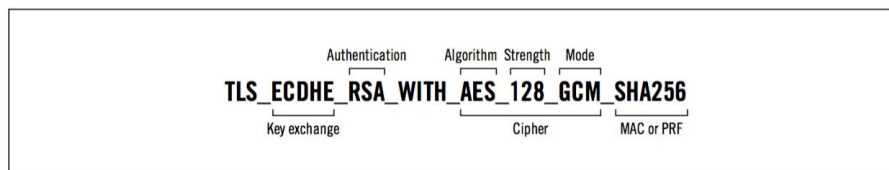


Figura 5.- Nomenclatura estándar (IANA) de las suites criptográficas empleadas por TLS. Fuente: [Ref - 10]

Por ejemplo, las siguientes *suites* criptográficas emplean los algoritmos detallados a continuación. Todas las *suites* criptográficas del protocolo TLS comienzan por "TLS":

Suite:	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS con intercambio de claves mediante ECDHE, empleando autenticación mediante claves (y certificados digitales) ECDSA, con el algoritmo de cifrado AES de 128 bits en modo GCM (Galois/Counter Mode) y utilizando SHA-256 como función pseudo-aleatoria (PRF).	
Suite:	TLS_DHE_RSA_WITH_AES_256_CBC_SHA
TLS con intercambio de claves mediante DHE, empleando autenticación mediante claves (y certificados digitales) RSA, con el algoritmo de cifrado AES de 256 bits en modo CBC (Cipher Block Chaining) y utilizando SHA-1 como algoritmo de <i>hashing</i> para el cálculo del MAC.	

La parte más compleja es la asociada a la parte final del nombre, es decir, al algoritmo de *hashing* para el cálculo del MAC, o al algoritmo de la función pseudo-aleatoria (PRF), ya que en TLS 1.2 y debido a la flexibilidad que esta versión del protocolo proporciona, puede tener distintos significados. Por ejemplo, para las suites AEAD (como AES/GCM y ChaCha20-Poly1305) la última parte refleja la función PRF. Para suites definidas antes de TLS 1.2, el nombre de la *suite* define el algoritmo de *hashing* para el (cálculo del) MAC en TLS (por ejemplo, SHA-1), no reflejándose la función PRF, al ser la versión del protocolo (no reflejada en el nombre de la *suite*) la que la define, ya que, por ejemplo, una suite puede hacer uso de HMAC-SHA256 como PRF para TLS 1.2, pero de HMAC-SHA1 como PRF para TLS 1.0 [Ref - 10].

Se debe tener en cuenta que OpenSSL hace uso de una nomenclatura para referenciar a las *suites* criptográficas ligeramente diferente a la definida en el estándar de IANA [Ref - 23] y empleada por otras tecnologías web, como por ejemplo IIS. A continuación, se muestran dos ejemplos de cada formato de nomenclatura (que como se puede apreciar, son bastante similares, y referencian el mismo código o identificador único de cada *suite* criptográfica):

Código:	0xC0,0x2B ó 0xC02B (RFC 5289)
OpenSSL:	ECDHE-ECDSA-AES128-GCM-SHA256

IANA:	TLS ECDHE ECDSA WITH AES 128 GCM SHA256
Código:	0x00,0x9E ó 0x009E (RFC 5288)
OpenSSL:	DHE-RSA-AES128-GCM-SHA256
IANA:	TLS DHE RSA WITH AES 128 GCM SHA256

NOTA: Se dispone de una comparativa de la nomenclatura de IANA con la empleada por otras librerías SSL/TLS, como GnuTLS, NSS y OpenSSL⁷¹.

Se recomienda por tanto hacer uso de suites criptográficas que comiencen por "ECDHE-RSA-..." (preferiblemente) o "DHE-RSA-...", en el caso de usar claves criptográficas RSA, y por "ECDHE-ECDSA-...", en el caso de usar claves criptográficas ECDSA (ver apartado "3.2. Generación de las claves criptográficas"). El "ANEXO B. SUITES CRIPTOGRÁFICAS RECOMENDADAS" proporciona un listado inicial de referencia con las *suites* criptográficas preferidas para una configuración inicial del servidor web HTTPS (en formato OpenSSL).

Debe tenerse en cuenta que (en el futuro), al hacer uso del protocolo TLS 1.3, éste no negocia el algoritmo de intercambio de claves como parte de las suites criptográficas, sino que el algoritmo de intercambio de claves se negocia aparte (admitiéndose únicamente en TLS 1.3 algoritmos que hacen uso de forward secrecy, como ECDHE y FFDHE, Finite Field Diffie-Hellman Ephemeral, RFC 7919⁷⁰, de agosto de 2016). Por tanto, las suites criptográficas soportadas por TLS 1.3⁷³ no reflejan en su nombre o definición el algoritmo de intercambio de claves a emplear. Por ejemplo:

Código:	0x13,0x01 ó 0x1301 (RFC5116)
IANA:	TLS AES 128 GCM SHA256
Código:	0x13,0x02 ó 0x1302 (RFC5116)
IANA:	TLS AES 256 GCM SHA384
Código:	0x13,0x03 ó 0x1303 (RFC7539)
IANA:	TLS CHACHA20 POLY1305 SHA256

TLS puede hacer uso de numerosos algoritmos de cifrado, tanto de flujo (*stream*), como de bloque (*block*), como de cifrado autenticado (AEAD) en TLS 1.2, como por ejemplo 3DES, RC4, AES, CAMELLIA, ChaCha20, etc. Dentro de los algoritmos de cifrado, se recomienda hacer uso de aquellos que proporcionan cifrado autenticado (Authenticated Encryption, AE), es decir, que combinan nativamente cifrado e integridad con la autenticación de los datos, conocidos habitualmente como AEAD (Authenticated Encryption with Associated Data). Estos algoritmos modernos también hacen uso de claves robustas y de forward secrecy (ver apartado "3.5.1. Forward secrecy"). Por ejemplo, AES en modo GCM (Galois/Counter Mode), frente a los modos CBC (Cipher Block Chaining) de AES, más inseguros, proporciona estas características.

Desde el punto de vista del rendimiento, numerosas CPUs en la actualidad disponen de soporte para operaciones que permiten ejecutar el algoritmo AES en hardware (AES-NI⁷⁴), agilizando y optimizando las operaciones de cifrado y descifrado.

La validación de la integridad de los datos intercambiados en TLS, a través de un MAC (Message Authentication Code), va implícita junto al proceso de cifrado.

⁷³ <https://tswg.github.io/tls13-spec/#rfc.appendix.B.4>

⁷⁴ AES-NI: (Intel) Advanced Encryption Standard New Instructions

Tras obtener el secreto maestro, éste (junto a dos semillas aleatorias del cliente y el servidor) es procesado mediante una función pseudo-aleatoria (PRF, Pseudo-Random Function) para obtener el material criptográfico necesario en función de la *suite* criptográfica a emplear, y principalmente, de los algoritmos de cifrado. Debido a que cada algoritmo de cifrado requiere un material criptográfico de longitud diferente, el valor de salida será adaptado en base a la *suite* negociada.

Se recomienda por tanto hacer uso de cifrado AEAD mediante AES 128 ó 256 bits en modo GCM para TLS (1.2), tal como se define en el RFC 5288⁷⁵, de agosto de 2008. Adicionalmente, y una vez se estandarice su uso en TLS, puede hacerse uso del cifrador de flujo ChaCha20, junto al autenticador Poly1305, tal como se define en el RFC 7905⁷⁶, de junio de 2016.

Desde el punto de vista de los navegadores y clientes web es posible evaluar su soporte para algoritmos y *suites* criptográficas robustas, o débiles y vulnerables, mediante el servicio "SSL Client Test" de Qualys SSL Labs [Ref - 3], y desde el punto de vista de una mala implementación, mediante el servicio "BadSSL" [Ref - 20] y, en concreto, su "Dashboard"⁷⁷. El servicio "BadSSL" permite verificar el comportamiento de los clientes web frente a configuraciones incorrectas o no deseadas de HTTPS. El dominio asociado, "badssl.com", proporciona multitud de servidores (y *hostnames*) diferentes, cada uno de ellos con debilidades y vulnerabilidades específicas, permitiendo comprobar las capacidades de conexión HTTPS de un cliente web en este tipo de escenarios vulnerables, así como obtener (desde el punto de vista del usuario) el mensaje de error, alerta o aviso concreto generado en cada situación no deseada.

En resumen, de nuevo, se debe de buscar siempre el equilibrio entre seguridad y rendimiento, por lo que en la actualidad para entornos web estándar, para la generación de las claves criptográficas se recomienda hacer uso de claves ECDSA de 256 bits y claves RSA de 2.048 bits (ver apartado "3.2. Generación de las claves criptográficas"), para el intercambio de claves se recomienda hacer uso de ECDHE de 256 bits y DHE de 2.048 bits (ver apartado "3.5.2. Intercambio de claves robusto") y, como recomendación general, se deberían utilizar algoritmos de cifrado simétrico que proporcionen, al menos, 128 bits de seguridad (debiendo valorarse el impacto en el rendimiento de los algoritmos de cifrado simétrico de 256 bits, respecto al incremento en seguridad que proporcionan). Asimismo, como recomendación general, se deberían utilizar algoritmos de *hashing* basados en SHA-256, debiendo emplearse algoritmos más robustos (ej. SHA-384, SHA-512...) en casos muy concretos, que requieran un mayor grado de seguridad.

Debe tenerse en cuenta que a la hora de hacer uso de *suites* criptográficas que utilizan el algoritmo SHA-1 de *hashing* para el MAC (reflejado al final de su nombre simplemente como "SHA"), en realidad lo utilizan con una clave en su variante HMAC (keyed-Hash Message Authentication Code), es decir HMAC-SHA1, para el que no se conocen ataques y que, por tanto, se puede utilizar con garantías de seguridad. (a diferencia del uso de SHA-1 sin HMAC).

⁷⁵ "AES Galois Counter Mode (GCM) Cipher Suites for TLS": <https://tools.ietf.org/html/rfc5288>

⁷⁶ "ChaCha20-Poly1305 Cipher Suites for TLS": <https://tools.ietf.org/html/rfc7905>

⁷⁷ <https://badssl.com/dashboard/>

3.5.6 Recomendaciones de los algoritmos y suites criptográficas

El resumen de recomendaciones de los algoritmos y suites criptográficas incluye:

- Utilizar algoritmos de intercambio de claves que proporcionen forward secrecy, preferiblemente ECDHE, o alternativamente DHE, frente a RSA.
- En el caso de ECDHE se deberían de utilizar algoritmos de intercambio de claves de 256 bits y, en el caso de DHE, de 2.048 bits.
- Si se hace uso de ECDHE, se debe de evaluar en detalle el tipo de curvas elípticas a utilizar, disponiéndose de un par ampliamente soportadas, como secp256r1(P-256) y secp384r1 (P-384), definidas por NIST y que no han sido analizadas públicamente de manera exhaustiva, y de nuevas curvas consideradas robustas, como Curve25519 (X2559) y Curve448 (X448), cuyo soporte se está empezando a popularizar.
- En el caso de hacer uso de DHE, el tamaño de los parámetros DH debería ser de al menos 2.048 bits, y se recomienda que coincida con la longitud o fortaleza de la clave privada correspondiente.
- En el caso de hacer uso de DHE, no se recomienda hacer uso de grupos DH conocidos o pre-configurados, como los del RFC 3526, o generar grupos propios. En su lugar, es preferible hacer uso de los grupos DH pre-definidos en el RFC 7919: ffdhe2048, ffdhe3072 o ffdhe4096.
- Configurar el servidor web dando preferencia a las suites criptográficas más robustas, listadas en orden descendente de preferencia según el nivel de seguridad que ofrecen.
- No hacer uso del cifrador NULL, de suites basadas en ADH, de algoritmos de cifrado con claves con fortaleza criptológica inferior a 100 bits, de suites con algoritmos de cifrado permitidos para su exportación, de los cifradores RC4 y 3DES, de los algoritmos de *hashing* MD5, RIPEMD-160 y SHA-1, etc.
- A la hora de priorizar las suites criptográficas de TLS, respecto al algoritmo de intercambio de claves y de autenticación, se recomienda hacer uso de suites criptográficas que comiencen por "ECDHE-ECDSA-...", en el caso de usar claves criptográficas ECDSA, y por "ECDHE-RSA-..." (preferiblemente) o "DHE-RSA-...", en el caso de usar claves criptográficas RSA.
- Hacer uso de algoritmos de cifrado que proporcionen cifrado autenticado (Authenticated Encryption, AE) en TLS 1.2, conocidos habitualmente como AEAD (Authenticated Encryption with Associated Data), como por ejemplo AES 128 ó 256 bits en modo GCM (Galois/Counter Mode), frente a los modos CBC (Cipher Block Chaining) de AES, o el cifrador de flujo ChaCha20, junto al autenticador Poly1305.
- Se deberían utilizar algoritmos de cifrado simétrico que proporcionen, al menos, 128 bits de seguridad (debiendo valorarse el impacto en el rendimiento de los algoritmos de cifrado simétrico de 256 bits, respecto al incremento en seguridad que proporcionan).
- Se deberían utilizar algoritmos de *hashing* basados en SHA-256, debiendo emplearse algoritmos más robustos (ej. SHA-384, SHA-512...) en casos muy concretos, que requieran un mayor grado de seguridad.

3.6 Mecanismos de revocación de certificados digitales

Los mecanismos de revocación de certificados digitales permiten al propietario de los mismos indicar a los usuarios (navegadores y clientes web) que el certificado no tiene validez, ya que, probablemente, ha sido reemplazado por otro nuevo.

Los certificados digitales deben de incluir mecanismos de revocación como CRLs (Certificate Revocation Lists) u OSCP (Online Certificate Status Protocol) o, preferiblemente, debe hacerse uso de OSCP Stapling. El propio certificado digital proporciona las instrucciones o referencias para hacer uso de los dos primeros mecanismos a la hora de verificar la validez del certificado y si éste ha sido o no revocado:

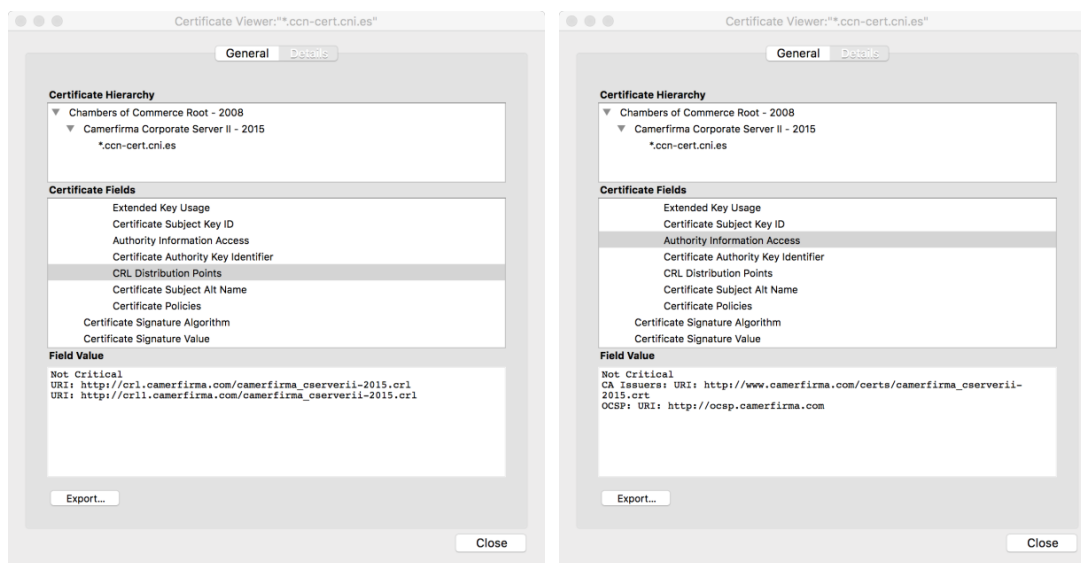


Figura 6.- Información de revocación incluida en los certificados digitales.

3.6.1 Certificate Revocation Lists (CRLs)

El mecanismo CRL (Certificate Revocation List), RFC 6818⁷⁸, permite a los navegadores y clientes web verificar la validez del estado de un certificado digital mediante la descarga de la lista de revocación (en adelante lista CRL), que es mantenida por la CA que emitió dicho certificado. Si el número de serie del certificado aparece en la lista, quiere decir que ha sido revocado.

La referencia a la lista CRL se encuentra en el campo "CRL Distribution Points" del certificado digital.

Las principales limitaciones de CRL son su escalabilidad, debido al tamaño de las listas de revocación, que se incrementan con cada nuevo certificado revocado, la ausencia de información reciente o actualizada (ya que debido al tamaño de las listas, los clientes web suelen mantener una copia local en caché), las implicaciones en el rendimiento a la hora de completar el *handshake* TLS y verificar el certificado (considerando que es necesario descargar la lista CRL en el momento de la conexión), y la gestión de fallos en la descarga de la lista CRL.

Si por cualquier motivo no es posible descargar la lista CRL, lo habitual es que los clientes web hagan uso de la modalidad de fallo permisivo (*soft fail*) y consideren el certificado como válido.

⁷⁸ <https://tools.ietf.org/html/rfc6818>

La utilización de CRLs por parte de los navegadores y clientes web hoy en día tiende al desuso, o a escenarios de uso muy específicos (ej. certificados EV) o no generalizados, empleado casi siempre como último recurso de comprobación.

3.6.2 Online Certificate Status Protocol (OCSP)

El mecanismo OCSP (Online Certificate Status Protocol), RFC 6960⁷⁹, permite a los navegadores y clientes web verificar la validez del estado de un certificado digital mediante la realización de una consulta en tiempo real al servicio OCSP (conocido como servicio de respuestas OCSP u *OCSP responder*), que es mantenido por la CA que emitió dicho certificado. La consulta al servicio OCSP se lleva a cabo sólo para el número de serie del certificado, optimizando el ancho de banda necesario para comprobar el estado de la revocación, y la respuesta obtenida permite saber si ha sido revocado o no.

La referencia al servicio OCSP se encuentra en el campo "(Certificate) Authority Information Access" (en la línea "OCSP") del certificado digital.

Las principales limitaciones de OCSP son que la CA debe disponer de un servicio OCSP rápido, fiable y actualizado, siempre disponible y que maneje altas cargas de trabajo puntuales, y de nuevo, las implicaciones en el rendimiento a la hora de completar el *handshake* TLS y verificar el certificado (considerando que es necesario recibir la respuesta del servicio OCSP en el momento de la conexión), junto a la gestión de fallos de acceso al servicio OCSP. Por otro lado, y desde el punto de vista de la privacidad, la CA puede conocer que sitios web están siendo visitados por un cliente web.

Si por cualquier motivo no es posible acceder al servicio OCSP, lo habitual es que los clientes web hagan uso de la modalidad de fallo permisivo (*soft fail*) y consideren el certificado como válido. Es relativamente sencilla la realización de ataques (mediante errores TCP, errores HTTP o incluso errores contemplados en el propio protocolo OCSP) que evitan que el cliente web reciba la respuesta del servicio OCSP.

Las respuestas OCSP pueden ser válidas hasta un máximo de 10 días para los certificados digitales finales (habitualmente unos 7 días), y hasta 12 meses para los certificados digitales de CAs intermedias. Adicionalmente, ciertas optimizaciones de rendimiento aplicadas sobre OCSP facilitan la realización de ataques de *replay*.

A la hora de seleccionar una CA, se recomienda verificar que la CA empleada para la emisión de los certificados digitales (ver apartado "3.3.2. Emisión de certificados digitales: Autoridades Certificadoras (CAs)") dispone de un servicio OCSP rápido y fiable.

De nuevo al igual que con las CRLs, la utilización de OCSP por parte de los navegadores y clientes web hoy en día es muy variada, y no suele estar activa por defecto, o sólo en escenarios de uso muy específicos (ej. certificados EV) o no generalizados, por lo que no se puede considerar efectiva, tal como demuestra la aparición de otras alternativas propietarias como CRLSets en Chrome o OneCRL en Firefox.

Si un cliente web hace uso de OCSP, cada vez que se conecta al servidor web también se tendrá que conectar a la CA para verificar la validez actual del certificado digital. OCSP stapling (ver

⁷⁹ <https://tools.ietf.org/html/rfc6960>

apartado "3.6.3. OCSP Stapling") permite evitar esta dependencia continua de la CA (con las implicaciones positivas asociadas de disponibilidad, rendimiento y privacidad).

3.6.3 OCSP Stapling y OCSP Must-Staple

Si un cliente web soporta y solicita OCSP Stapling, y el servidor web también implementa OCSP Stapling, cada vez que el cliente se conecta al servidor web, éste podrá proporcionarle directamente la respuesta OCSP para que el cliente pueda verificar la validez actual del certificado digital, evitando así la mayoría de implicaciones negativas asociadas al uso de OCSP o de CRL.

OCSP Stapling⁸⁰, RFC 6066⁸¹, fue diseñado debido a que ni CRL ni OCSP proporcionan la seguridad y el rendimiento deseados para la verificación de la revocación de certificados en las conexiones HTTPS. En la mayoría de los casos, las comprobaciones de revocación no han sido efectivas, tal como se demostró en el año 2011 al ser necesaria la distribución de actualizaciones para los navegadores y clientes web de cara a añadir a las listas negras de revocación los certificados falsos generados tras el compromiso de varias CAs. Con OCSP Stapling en lugar de ser el cliente web el que debe encargarse de verificar el estado de la revocación, es el servidor web el que lo hace periódicamente y proporciona los detalles al cliente.

La extensión TLS asociada a OCSP Stapling permite al servidor web proporcionar información lo más actualizada posible sobre el estado de la revocación de un certificado digital al cliente web, enviada (adjuntada o grapada, "stapled", de ahí su nombre) durante el establecimiento de la conexión o *handshake* TLS, desde el propio servidor web (sin depender de la CA en el momento de la conexión). Para ello, el servidor obtiene periódicamente el estado del certificado mediante una petición OCSP al servicio de la CA, recibiendo una respuesta firmada por la CA y con una marca de tiempo (*timestamp*). Cuando el cliente web recibe dicha respuesta durante el establecimiento de la conexión TLS, puede verificar la firma de la respuesta OCSP y, por tanto, la validez del certificado. El cliente también puede confiar en la respuesta ya que ésta tiene una validez limitada en el tiempo (en base al *timestamp*).

Si por cualquier motivo no es posible obtener la respuesta OCSP Stapling, y el servidor web declara que hace uso de este mecanismo de revocación (mediante la directiva OCSP Must-Staple, descrita posteriormente), los clientes web podrán hacer uso de la modalidad de fallo restrictivo (*hard fail*) y no considerar el certificado como válido.

En el escenario de uso inicial de OCSP Stapling, el cliente web no puede saber si el sitio web soporta OCSP Stapling y si, por tanto, debe esperar recibir una respuesta OCSP en el *handshake* TLS. Para solucionar este escenario surgió OCSP Must-Staple.

OCSP Must-Staple⁸², RFC 7633⁸³, define un nuevo ajuste asociado a los certificados digitales X.509 que debe ser activado por la CA, indicando al navegador o cliente web que el certificado digital debe ser ofrecido a través de una sesión TLS que incluya una respuesta OCSP Stapling. En

⁸⁰ <https://scotthelme.co.uk/ocsp-stapling-speeding-up-ssl/>

⁸¹ <https://tools.ietf.org/html/rfc6066>

⁸² <https://scotthelme.co.uk/ocsp-must-staple/>

⁸³ <https://tools.ietf.org/html/rfc7633>

caso contrario, el cliente web debe de considerar que el certificado digital no es válido (*hard fail*).

El proceso empleado para generar una solicitud de firma de un certificado (CSR), destinado a la CA, debe de incluir la extensión OCSP Must-Staple.

Debido a que la combinación de OCSP Stapling junto a OCSP Must-Staple es más restrictiva que las opciones de revocación disponibles previamente, y a que podrían presentarse escenarios que afecten a la disponibilidad del entorno web si no se implementan correctamente (al igual que ocurre con HPKP o CSP), se recomienda implementarlo con cautela y evaluando detalladamente su implantación. Para ello, se dispone de un nuevo mecanismo de notificación, denominado OCSP Expect-Staple⁸⁴ que permite al propietario de un sitio web monitorizar el funcionamiento de OCSP Stapling antes de su activación de manera estricta mediante OCSP Must-Staple. Sus capacidades de monitorización son similares a las asociadas a la directiva "report-uri" de cabeceras HTTP de seguridad como HPKP (ver apartado "3.3.8. Restricción de los certificados digitales legítimos: HPKP") o CSP (ver apartado "3.7.6. Content Security Policy (CSP)").

La lista precargada de HSTS (ver apartado "3.7.4. Forzando el uso estricto (por defecto) de HTTPS: HSTS") se está convirtiendo en la ubicación de referencia en los navegadores web para reflejar los requisitos de seguridad de los entornos web. Dicha lista no sólo contiene la directiva propia de HSTS, sino también los pines prefijados de HPKP (para ciertos dominios únicamente), Expect-CT (ver apartado "3.3.6. Certificate Transparency (CT)") y OCSP Expect-Staple. En el momento de elaboración del presente informe, el proceso para añadir OCSP Expect-Staple al entorno web se lleva a cabo manualmente con los responsables de la lista precargada de Chromium/Chrome.

La definición de OCSP Expect-Staple en dicha lista incluye la declaración de que se espera que las respuestas del entorno web hagan uso de OCSP Stapling e incluyan una respuesta OCSP, la URL a la que remitir informes de errores asociados a OCSP Expect-Staple, y si la directiva aplica igualmente a todos los subdominios (includeSubDomains).

Cuando la directiva OCSP Expect-Staple está activa, si el navegador web establece una conexión con el entorno web pero no recibe una respuesta OCSP válida mediante OCSP Stapling, generará un informe notificando esta situación a la URL previamente definida, empleando el formato descrito en la especificación⁸⁵. El informe no sólo notifica si la respuesta OCSP no ha sido recibida (ausente), sino también si se ha recibido, pero no es válida por otros motivos (ha expirado, no aplica para el certificado digital recibido, su formato es incorrecto, etc.). Estos detalles proporcionan la información necesaria para configurar e implementar un entorno de OCSP Stapling óptimo.

Inicialmente, OCSP Stapling sólo podía proporcionar información sobre el certificado digital del servidor web, pero el RFC 6961⁸⁶ (de junio de 2013) extiende sus capacidades para proporcionar detalles de revocación de múltiples certificados simultáneamente, como por ejemplo los asociados a las CAs intermedias reflejadas en la cadena de certificación.

⁸⁴ <https://scotthelme.co.uk/ocsp-expect-staple/>

⁸⁵ https://docs.google.com/document/d/1aISgJIWglcOAhqNfK-2vtQI-_dWAapc-VLDh-9-BE/edit#heading=h.rkpittae54q

⁸⁶ <https://tools.ietf.org/html/rfc6961>

La ventaja de hacer uso de OCSP Stapling junto a OCSP Must-Staple es que se reduce significativamente el tiempo en el que los clientes web son notificados de un potencial compromiso relativo a un certificado digital, pasando de un máximo de 39 meses (periodo máximo de validez de los certificados digitales, u 825 días con la nueva propuesta a partir de marzo de 2018, ver apartado "3.3.5. Renovación de los certificados digitales"), al máximo asociado al tiempo de vida de una respuesta OCSP, es decir, normalmente un máximo de 7 días.

3.6.4 Recomendaciones para la revocación de certificados digitales

El resumen de recomendaciones para la revocación de certificados digitales incluye:

- Los certificados digitales deben de incluir mecanismos de revocación como CRLs (Certificate Revocation Lists) u OCSP (Online Certificate Status Protocol) o, preferiblemente, debe hacerse uso de OCSP Stapling.
- Al seleccionar una CA, se recomienda verificar que dispone de un servicio OCSP rápido y fiable.
- Hacer uso de la extensión TLS OCSP Stapling para proporcionar a los clientes información actualizada sobre el estado de la revocación de un certificado digital.
- Hacer uso de una CA que permita añadir la directiva OCSP Must-Staple a los certificados digitales, para permitir a los clientes web considerar el certificado como no válido si no es posible obtener una respuesta OCSP Stapling válida.
- Hacer uso de OCSP Must-Staple. Complementariamente, hacer uso de las capacidades de notificación de OCSP Expect-Staple, que permiten monitorizar el funcionamiento de OCSP Stapling antes de su activación de manera estricta mediante OCSP Must-Staple. Estas capacidades, aún disponibles de manera experimental, permiten hacer un uso progresivo y controlado de OCSP Must-Staple.
- Proporcionar detalles de revocación de múltiples certificados simultáneamente, como por ejemplo los asociados a las CAs intermedias, mediante OCSP Stapling (si se considera necesario).

3.7 Contenidos y aplicaciones web HTTPS

Los siguientes apartados proporcionan numerosas recomendaciones que afectan al diseño y configuración tanto del servidor web y sus contenidos, como de las aplicaciones web, todos ellos centrados en proporcionar la máxima cobertura para ofrecer todos los contenidos disponibles en el entorno web a través de HTTPS.

3.7.1 Prioridad en los buscadores web

Uno de los objetivos de la mayoría de sitios web es estar bien posicionados en los buscadores de Internet (como Google, Yahoo!, Bing, etc.) de cara a poder ser fácilmente encontrados e identificados por los potenciales usuarios.

En agosto de 2014, Google anunció [Ref - 24] que daría más relevancia en el posicionamiento (o *ranking*) de las búsquedas, asociados a los algoritmos de SEO (Search Engine Optimization), a los sitios web que hacían uso de HTTPS frente a los que empleaban HTTP. Posteriormente, en diciembre de 2015 Google anunció que la búsqueda e indexación de páginas web se realizaría

por defecto para la versión HTTPS, es decir, comenzando por las páginas HTTPS frente a HTTP⁸⁷, incluso si no se han identificado referencias a HTTPS.

Por este motivo, con el objetivo de facilitar la indexación de los contenidos del sitio web, se recomienda no añadir la directiva "noindex" para las URLs que hacen uso de HTTPS, ni a través de la etiqueta HTML "<meta>", ni de las cabeceras HTTP.

Adicionalmente, la política establecida en el fichero "robots.txt" debe permitir la indexación de los contenidos HTTPS del servidor web por parte de los buscadores web.

También relacionado con los buscadores web y los algoritmos de SEO, en relación a la existencia de contenidos duplicados y de la redirección de cualquier recurso desde HTTP hacia HTTPS (ver apartado "3.7.3. Forzando el uso (por defecto) de HTTPS"), se recomienda hacer uso de la etiqueta "rel=canonical"⁸⁸⁸⁹, que permite definir cuál es el recurso canónico o de referencia dentro del servidor web para contenidos similares (o duplicados).

La etiqueta "rel=canonical" debe ser añadida en la cabecera "<head>" de todas las páginas web duplicadas (e incluso de las páginas web únicas o no duplicadas), especificando la página web HTTPS de referencia para los buscadores web:

```
<link rel="canonical" href="https://www.dominio.es/p.php?i=1" />
```

En caso de que la redirección global de HTTP hacia HTTPS no aplique a todos los recursos de la aplicación web, esta etiqueta permite añadir una indicación adicional de la intencionalidad de utilizar HTTPS, de cara a la indexación por parte de los buscadores web.

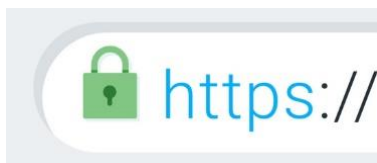


Figura 7.- Prioridad de sitios web HTTPS en los buscadores web. Fuente: Google

3.7.2 Rendimiento

En el pasado uno de los argumentos en contra del uso de HTTPS frente a HTTP ha sido el impacto en el rendimiento del servidor o aplicación web. Las numerosas mejoras y optimizaciones tanto en los protocolos asociados a HTTPS, como en las implementaciones [Ref - 9], han permitido incrementar significativamente el rendimiento de HTTPS y el tiempo de carga de las páginas web que hacen uso de este protocolo, e incluso que éste supere el rendimiento de HTTP, especialmente si se hace uso de HTTP/2 (ver apartado "3.4.5. HTTP/2").

Dentro de las mejoras de rendimiento y optimizaciones (aplicadas sobre el uso de CPU, la latencia de la conexión, etc.) disponible se encuentran las optimizaciones del *TLS handshake* (1-RTT), del *TLS record size* y de la cadena de certificación, *TLS session resumption* o *TLS session caching* (*TLS session tickets*), *TLS False Start*, *TCP Fast Open*, *TLS early termination*, *OCSP Stapling*, etc., junto a nuevas características que serán incorporadas en TLS 1.3, como 0-RTT o zero-RTT.

⁸⁷ <https://security.googleblog.com/2015/12/indexing-https-pages-by-default.html>

⁸⁸ <https://webmasters.googleblog.com/2009/02/specify-your-canonical.html>

⁸⁹ <https://webmasters.googleblog.com/2013/04/5-common-mistakes-with-relcanonical.html>

Se dispone de numerosos recursos⁹⁰ para evaluar el rendimiento de HTTP frente a HTTPS, como por ejemplo "HTTP vs HTTPS Test"⁹¹.

Debido a ataques como CRIME (ver apartado "3.8. Vulnerabilidades en HTTPS"), se recomienda deshabilitar las capacidades de compresión de TLS del servidor web. Sin embargo, se dispone de la posibilidad de hacer uso de mecanismos de compresión Brotli [Ref - 25], pero únicamente para conexiones HTTPS, dónde un cliente web puede indicar que dispone de soporte a través de la cabecera HTTP "Accept-encoding: br". Estas capacidades fueron incluidas en Firefox 44⁹², Chrome 55⁹³, Edge 15 u Opera 38⁹⁴. Brotli proporciona mejoras en torno al 20-26% respecto a otros algoritmos de compresión existentes, lo que reduce el ancho de banda utilizado para las comunicaciones web, y agiliza el tiempo de carga de las páginas y recursos web. Se recomienda verificar si se dispone de soporte para compresión Brotli en el servidor web, y habilitarla en caso de estar disponible.

3.7.2.1 TLS False Start

El establecimiento de una sesión TLS, mediante un *TLS handshake* estándar, requiere hacer uso de 2-RTT (roundtrips, o intercambios de ida y vuelta) entre el cliente y el servidor [Ref - 9], afectando al rendimiento de las conexiones HTTPS.

TLS False Start es una extensión del protocolo TLS que permite la transmisión de datos de aplicación cifrados cuando la sesión TLS está sólo establecida parcialmente, lo que permite reducir el establecimiento de la sesión TLS a 1-RTT. *TLS False Start* optimiza el rendimiento de TLS sacrificando ligeramente su seguridad, ya que los datos se envían antes de verificar la integridad del *TLS handshake*.

TLS False Start está soportada por todos los navegadores y clientes web modernos, pero para ser utilizada con confianza y contrarrestar el sacrificio en seguridad mencionado, estos verifican que el servidor web haga uso de suites de cifrado de (al menos) 128 bits, con forward secrecy, y de la extensión TLS NPN⁹⁵ (Next Protocol Negotiation, previamente) o ALPN (Application Layer Protocol Negotiation).

Si se combina *TLS False Start* junto a *TLS session resumption* (o *TLS session caching*, ver apartado "3.7.2.2. TLS session resumption o caching, y TLS session tickets") es posible hacer uso de TLS *handshakes* de 1-RTT para nuevas sesiones, y optimizar los cálculos criptográficos para sesiones posteriores (resumidas).

Complementariamente, y para poder comparar las versiones del protocolo TLS entre sí, uno de los objetivos de diseño de TLS 1.3 (ver apartado "3.4.1. Versiones del protocolo SSL y TLS") es disponer de capacidades para poder establecer sesiones TLS nuevas en 1-RTT y resumir sesiones en 0-RTT.

⁹⁰ <https://scotthelme.co.uk/still-think-you-dont-need-https/>

⁹¹ <https://www.httpvshhttps.com>

⁹² <https://www.mozilla.org/en-US/firefox/44.0/releasenotes/>

⁹³ <https://groups.google.com/a/chromium.org/forum/#!searchin/blink-dev/brotli/blink-dev/JufzX024oy0/WEOGbN43AwAJ>

⁹⁴ <http://caniuse.com/#feat=brotli>

⁹⁵ <https://www.imperialviolet.org/2012/04/11/falsestart.html>

3.7.2.2 TLS session resumption o caching, y TLS session tickets

Si un cliente se ha comunicado previamente con el servidor mediante TLS, es posible establecer una nueva sesión de manera abreviada, y reducir el uso de CPU al reutilizarse parámetros criptográficos de la sesión previa.

TLS session resumption, también referido como *TLS session caching* (RFC 5077⁹⁶), es un mecanismo que optimiza el rendimiento de TLS sacrificando ligeramente su seguridad. Al establecer una sesión TLS, el cliente y el servidor negocian y derivan una clave maestra, una operación costosa criptográficamente hablando. Si se hace uso de *TLS session caching*, las conexiones posteriores que forman parte de la misma sesión harán uso de la misma clave maestra, reduciendo el tiempo de conexión (a 1-RTT). Si un potencial atacante obtiene la clave maestra, podría descifrar todas las conexiones asociadas a una misma sesión. Sin embargo, debido a que las sesiones TLS no deberían perdurar a lo largo del tiempo, es aceptable su utilización desde el punto de vista de seguridad, teniendo en cuenta los beneficios de rendimiento obtenidos.

Se recomienda establecer el periodo máximo en el que se cachearán las sesiones TLS, empleando por ejemplo un día (24 horas) como referencia. Asimismo, se recomienda no compartir la caché de sesiones TLS entre diferentes entornos, servidores o aplicaciones web (en entornos web de *hosting* compartido).

Las sesiones son identificadas tanto en el cliente como en el servidor mediante un identificador de sesión (Session ID - RFC 5246⁵⁵), intercambiado entre ambas partes durante el establecimiento inicial de la sesión. Al hacer uso de *TLS session caching*, tanto el cliente como el servidor deben mantener la información del estado de la sesión durante un tiempo.

La alternativa (para evitar el almacenamiento en el lado del servidor, especialmente cuando éste gestiona miles o millones de clientes web) es hacer uso de tickets de sesión (*TLS session tickets* - RFC 5077⁹⁶), donde todo el estado de la sesión se mantiene en el cliente (también conocido como *stateless TLS session resumption*). Desde el punto de vista de seguridad, se desaconseja el uso de *TLS session tickets*, ya que exponen todos los detalles criptográficos del estado de la sesión a través del canal de comunicación, protegidos únicamente con una clave de ticket, que, en algunos casos, puede ser más débil que las claves de la propia sesión. Por ejemplo, OpenSSL hace uso de AES 128 bits para las claves de ticket y, por defecto, reutiliza las claves de ticket entre sesiones, que son creadas al arrancar el servidor web y no son rotadas (pudiendo ser utilizadas durante muy largos periodos de tiempo y anulando de manera efectiva *forward secrecy*). Asimismo, en el caso de usar *TLS session tickets* se recomienda no compartir la clave de ticket entre diferentes entornos, servidores o aplicaciones web (en entornos web de *hosting* compartidos).

Por ejemplo, Apache 2.4.12+ dispone de capacidades avanzadas para la gestión de la cache de sesiones TLS, la capacidad de deshabilitar los tickets de sesión y la gestión de las claves de ticket.

⁹⁶ <https://tools.ietf.org/html/rfc5077>

3.7.2.3 Preconnect

La etiqueta HTML `<link>`⁹⁷ especifica relaciones entre el documento web actual y otros recursos externos, como por ejemplo hojas de estilo (CSS o *stylesheets*), pero también puede ser utilizada para referenciar otros recursos web (por ejemplo, externos).

La directiva HTML "preconnect"⁹⁸ [Ref - 26] de `<link>` (`<link rel="preconnect" href="https://...">`) permite indicarle al navegador o cliente web que debe de abrir prematuramente una conexión HTTPS hacia el servidor y puerto indicados para cargar recursos que serán referenciados posteriormente. Estas indicaciones previas mejoran el rendimiento, llevando a cabo la resolución DNS, establecimiento de la conexión TCP y la negociación mediante TLS anticipadamente. La directiva "preconnect" está soportada en Chrome 46⁹⁹, Firefox 39¹⁰⁰, y Opera 33¹⁰¹, y complementa otras directivas web para mejorar el rendimiento como "preload", "prefetch" o "prerender"¹⁰².

3.7.3 Forzando el uso (por defecto) de HTTPS

Uno de los problemas que todavía subyacen en las tecnologías web es que aún por defecto se hace uso de HTTP, y no de HTTPS, siendo éste una característica opcional. Con el objetivo de implementar un servidor o aplicación web únicamente disponible por HTTPS, se recomienda redirigir automáticamente todo el tráfico HTTP (TCP/80) hacia HTTPS (TCP/443; ver apartado "3.1. Acceso a los puertos TCP/IP asociados a los servicios web").

En ningún caso lo que debe ocurrir es que un acceso por parte de un usuario a la versión HTTPS del entorno, servidor o aplicación web sea redireccionado a la versión HTTP, que no hace uso de mecanismos de autenticación y/o cifrado.

3.7.4 Forzando el uso estricto (por defecto) de HTTPS: HSTS

La cabecera de seguridad del protocolo HTTP conocida como HSTS o STS (HTTP Strict Transport Security) [Ref - 27] permite al servidor web declarar (e informar a los clientes web) que sólo está accesible mediante HTTPS (y no mediante HTTP). Como resultado, los navegadores y clientes web con soporte para HSTS no generarán ninguna petición HTTP hacia el entorno web, empleando automáticamente HTTPS en todo momento.

HSTS previene errores de configuración e implementación, y ataques de tipo MitM tanto pasivos, basados en la captura de tráfico no cifrado, como activos, conocidos como SSLstrip, es decir, donde un potencial atacante fuerza al cliente web víctima a conectarse al servidor web empleando HTTP, sin hacer uso de cifrado, en lugar de HTTPS y, desvelando como resultado información sensible, como por ejemplo credenciales, cookies o identificadores de sesión, etc.

Adicionalmente, el uso de HSTS mitiga los ataques debidos a la posibilidad de que el usuario acepte certificados digitales inválidos. Las alertas y errores de seguridad en la validación de los

⁹⁷ <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/link>

⁹⁸ <https://www.igvita.com/2015/08/17/eliminating-roundtrips-with-preconnect/>

⁹⁹ <https://www.chromestatus.com/feature/5560623895150592>

¹⁰⁰ https://bugzilla.mozilla.org/show_bug.cgi?id=1135160

¹⁰¹ <http://caniuse.com/#search=preconnect>

¹⁰² <https://www.keycdn.com/blog/resource-hints/>

certificados digitales en los navegadores web no pueden ser ignorados o evitados por los usuarios cuando se hace uso de HSTS.

La cabecera HSTS debe de ser incluida en todas y cada una de las respuestas HTTPS generadas por el servidor o aplicación web:

```
Strict-Transport-Security: max-age=31536000 [; includeSubDomains]
[; preload]
```

Debe tenerse en cuenta que el navegador web refuerza el uso de HTTPS para un servidor web concreto durante el periodo indicado por la directiva "max-age" (especificado en segundos) de la cabecera HSTS.

Se recomienda que el valor de "max-age" sea de al menos 10886400 (18 semanas), ya que este es el valor mínimo requerido para ser admitido en la lista precargada (descrita posteriormente). Sin embargo, sería recomendable hacer uso de periodos temporales mayores de, al menos, 6 ó 12 meses (ó 365 días: "31536000").

Para minimizar el posible impacto de una configuración incorrecta al comenzar el despliegue de HSTS, se recomienda incrementar progresivamente el valor de "max-age", comenzando con un valor pequeño (5 minutos: "300"), y aumentándolo a valores mayores (1 semana: "604800", 1 mes: "2592000", y finalmente a su valor definitivo). Simultáneamente, se deberían monitorizar exhaustivamente los logs de acceso del servidor web en busca de peticiones recibidas por HTTP (y no por HTTPS).

Desde el punto de vista de seguridad, HSTS debe de ser implementado para la totalidad del dominio, y no sólo para servidores web individuales. Para ello, se debe de hacer uso de la directiva "includeSubDomains". Para evitar problemas de accesibilidad, todos los servidores y aplicaciones web del dominio deben hacer uso de HTTPS y disponer de certificados digitales válidos. La aplicación de HSTS a todos los subdominios mitiga ataques basados en la manipulación del servicio DNS y en la generación de nuevos nombres de servidores web inexistentes en realidad, y sobre los que no se haría uso de conexiones HTTPS (al no existir, no existe en los clientes web una cabecera HSTS, o una directiva específica en la lista precargada, para ellos).

Adicionalmente, es importante resaltar que HSTS es un mecanismo de seguridad de tipo TOFU (Trust On First Use), es decir, que no es de aplicación la primera vez que el cliente web se conecta al servidor web. Para mitigar esta debilidad, la directiva "preload" indica el interés y la posibilidad de que el servidor web o el dominio asociado haya sido incluido en la lista precargada de HSTS de los navegadores web, "HSTS Preload List". En realidad, la lista precargada únicamente contempla dominios completos, no siendo posible añadir servidores web individualmente.

La lista precargada de HSTS [Ref - 28] en una lista interna existente en los navegadores web, y gestionada por Chrome/Google, que es empleada como referencia por el resto de navegadores web para determinar qué dominios desean hacer uso exclusivo de HTTPS y, por tanto, para los que el navegador web nunca generará tráfico HTTP (evitando así ataques incluso en la primera conexión del cliente web).

Para que un dominio pueda ser admitido en la lista precargada debe de cumplir unos requisitos mínimos [Ref - 28]¹⁰³ (desde febrero de 2016), similares a los descritos previamente, y mantenerlos a partir del momento de su inclusión en la lista. Es posible comprobar la lista precargada completa [Ref - 29], incluyendo todos los dominios existentes en la misma, así como la lista pendiente [Ref - 30] que será incluida en futuras versiones de los navegadores web. Se recomienda monitorizar estas listas, especialmente esta última, una vez se ha solicitado la inclusión de un dominio en la misma.

En resumen, la activación de HSTS debe de entenderse como una declaración a futuro por parte del responsable del servidor web y, preferiblemente, del dominio, de que se soportará HTTPS durante toda la vida en la totalidad del dominio (*. dominio.es). Una vez el dominio ha sido añadido a la lista precargada, no es sencilla su eliminación¹⁰⁴.

3.7.5 Evitando contenidos mixtos HTTP y HTTPS

El objetivo de la implementación HTTPS propuesta por el presente informe es que los entornos, servidores o aplicaciones web hagan uso de HTTPS para la totalidad de sus contenidos, sin excepción.

Las páginas web con contenidos mixtos [Ref - 31] (o mezclados) son aquellas que, aunque sean servidas inicialmente mediante HTTPS, incluyen referencias a contenidos y a otros recursos web empleando HTTP (es decir, una conexión en texto plano sin cifrar ni autenticar), como, por ejemplo, a imágenes, otras páginas web, *frames* o *iframes*, scripts con código Javascript, hojas de estilo (CSS), etc. Desafortunadamente, una única referencia que no haga uso de HTTPS podría ser interceptada y manipulada por un potencial atacante para comprometer la sesión HTTPS del usuario.

Habitualmente se definen dos tipos de contenidos mixtos: los pasivos o menos relevantes, ya que no pueden interactuar con el resto de la página web, como, por ejemplo, imágenes y contenidos multimedia, y los activos o más críticos (por sus capacidades de interacción con el resto de la página web), como por ejemplo scripts, contenidos y etiquetas HTML (*iframes*), hojas de estilo (CSS), objetos Flash o Java, etc. La inclusión de contenidos activos permitiría a un potencial atacante tomar control completo de la sesión y poder realizar cualquier acción en el entorno web suplantando la identidad del usuario víctima. Habitualmente, este contenido activo mixto es bloqueado por los navegadores web¹⁰⁵. Sin embargo, incluso a través de los contenidos mixtos pasivos sería posible manipular al usuario en base a los contenidos de las imágenes que éste visualiza (mediante ingeniería social), enviar imágenes que incluyan exploits, o realizar ataques de *content sniffing* dónde una imagen podría ser interpretada por el navegador o cliente web como un script, junto a la posibilidad de afectar la privacidad del usuario y hacer un seguimiento de sus actividades.

Se recomienda revisar todos los contenidos y código asociados al entorno, aplicación y páginas web (estáticas y dinámicas) y sustituir todas las referencias a "http://" por "https://" (o por referencias relativas o absolutas sin especificar el protocolo), exceptuando enlaces externos a

¹⁰³ <https://hstspreload.org/#submission-requirements>

¹⁰⁴ <https://hstspreload.org/#removal>

¹⁰⁵ Dependiendo del tipo de navegador web empleado y de su versión.

otras páginas web (en el atributo "href" de la etiqueta anchor "<a>"), que, salvo excepciones, no se consideran contenidos mixtos (ya que hacen que el navegador web navegue a una página web distinta). Se debe por tanto diferenciar entre enlaces web (a sitios web externos fuera del control del entorno web) y la inclusión de recursos web, integrados en el propio entorno web (tanto propios como externos). Son especialmente relevantes los contenidos que son cargados como parte de la página web, como hojas de estilo (CSS), imágenes, scripts, contenido HTML (iframes), etc.¹⁰⁶. El proceso de conversión de todos los contenidos web a HTTPS puede ser complejo y tedioso dependiendo de la complejidad y extensión del sitio web, de los contenidos existentes actualmente y de otras posibles dependencias¹⁰⁷.

Además de las referencias propias, es decir, asociadas al mismo entorno web, se debe prestar especial atención a las referencias a recursos pertenecientes a terceros (como librerías, servicios de anuncios, mapas, integración con redes sociales, o cualquier otra funcionalidad), ya que se dispone de menos control sobre ellos de cara a asegurar el uso exclusivo de HTTPS. Todos los recursos de terceros deberían de ser referenciados mediante enlaces HTTPS e, incluso, se debería verificar su validez e integridad (frente a manipulaciones inesperadas) a través de un nuevo estándar denominado SubResource Integrity (SRI)¹⁰⁸.

Es muy importante ser consciente de que la inclusión de recursos y componentes de servicios web de terceros en las páginas web propias crea un vínculo de confianza muy estrecho entre distintos sitios web. Por ejemplo, la inclusión de código Javascript de un tercero en la aplicación web hace que dicho código deba ser considerado como propio, ya que cualquier acción malintencionada del mismo, vulnerabilidad o posible manipulación, tendría control completo de la aplicación web y un efecto negativo directo sobre la misma.

Asimismo, la utilización o carga de contenidos mixtos (especialmente los considerados activos) puede no estar permitida por los navegadores o clientes web, imponiendo cada uno de ellos diferentes criterios y restricciones al respecto, que pueden ser evaluados a través del servicio "SSL Client Test" [Ref - 3] ("Mixed Content Handling").

También es posible evaluar la existencia de contenidos mixtos en un entorno web HTTPS mediante herramientas como "Mixed Content Scan"¹⁰⁹.

El objetivo de evitar contenidos mixtos HTTP y HTTPS puede ser reforzado y simplificado mediante CSP (ver apartado "3.7.6. Content Security Policy (CSP)"), especialmente para entornos web de mayor tamaño y más complejos, y mediante la utilización de HSTS (ver apartado "3.7.4. Forzando el uso estricto (por defecto) de HTTPS: HSTS").

3.7.6 Content Security Policy (CSP)

El estándar web conocido como Content Security Policy (CSP) [Ref - 32], aparte de sus múltiples usos para mitigar muchos otros tipos de ataques asociados a tecnologías web (como por

¹⁰⁶ <https://developers.google.com/web/fundamentals/security/prevent-mixed-content/what-is-mixed-content#mixed-content-types--security-threats-associated>

¹⁰⁷ <https://developers.google.com/web/fundamentals/security/prevent-mixed-content/fixing-mixed-content>

¹⁰⁸ <https://www.w3.org/TR/SRI/>

¹⁰⁹ <https://github.com/bramus/mixed-content-scan>

ejemplo Cross-Site Scripting, XSS), puede ser utilizado para mejorar la implementación de HTTPS.

CSP es un mecanismo que permite restringir el comportamiento de los navegadores y clientes web, definiendo una política que facilite controlar como se obtienen (o cargan) los diferentes recursos que forman parte de una página web, indicando cuáles de ellos son permitidos y desde qué orígenes. Para su implementación se emplea la cabecera HTTP "Content-Security-Policy"¹¹⁰. Se recomienda hacer una utilización mucho más amplia y rigurosa de CSP¹¹¹ para la protección de los entornos web de la que será detallada a continuación, que únicamente se centra en los aspectos relacionados con el uso de HTTPS. La herramienta "CSP Builder"¹¹² puede ser empleada para crear la política CSP, mientras que "CSP Analyser"¹¹³ se puede emplear para analizar una política CSP existente.

CSP ha evolucionado significativamente a lo largo de los últimos años, identificado inicialmente por la versión 1.0 del estándar [Ref - 33], soportada por la mayoría de navegadores web hoy en día¹¹⁴ (salvo IE¹¹⁵). Posteriormente, la versión 2 o "Level 2" de CSP [Ref - 34], añadió nuevas directivas, características y mecanismos de protección, algunos de ellos asociados a HTTPS y que serán descritos posteriormente. El soporte de todas las características de CSP Level 2 en los navegadores web modernos está bastante extendido, pero es más limitado¹¹⁴. En la actualidad, está en proceso de revisión y diseño la versión 3 o "Level 3" de CSP [Ref - 35], incluyendo nuevas directivas y cambios respecto a las versiones previas.

En el caso de HTTPS, CSP puede ser utilizado para evitar la obtención de recursos mediante enlaces HTTP que no emplean HTTPS, impidiendo así el uso de contenidos mixtos (ver apartado "3.7.5. Evitando contenidos mixtos HTTP y HTTPS").

Por un lado, mediante la directiva "default-src" de CSP se le puede indicar al navegador web desde donde se le permite cargar los recursos (incluyendo imágenes, scripts, CSS, etc.) por defecto asociados al servidor o aplicación web. Si su valor es "https:" se indica que sólo podrán ser obtenidos mediante HTTPS, y no mediante HTTP, independientemente del dominio desde el que se obtengan (esta sencilla política CSP no está aplicando ninguna restricción a este respecto). La política se puede extender también a los formularios web mediante la directiva "form-action" con el mismo valor.

```
Content-Security-Policy: default-src https;; form-action https;
```

NOTA: En el caso de CSP, al igual que para HPKP, es posible aplicar de manera estricta y efectiva la política a la vez que se reciben informes de violación de la misma (mediante "report-uri"). Esta opción (descrita posteriormente junto a "...-Report-Only") es muy interesante para identificar recursos que no deberían estar siendo referenciados, o ataques que hayan incluido nuevos contenidos web no autorizados.

¹¹⁰ <https://scotthelme.co.uk/content-security-policy-an-introduction/>

¹¹¹ <https://scotthelme.co.uk/csp-cheat-sheet/>

¹¹² <https://report-uri.io/home/generate>

¹¹³ <https://report-uri.io/home/analyse>

¹¹⁴ <http://caniuse.com/#search=csp> (<http://caniuse.com/#feat=contentsecuritypolicy> y <http://caniuse.com/#feat=contentsecuritypolicy2>)

¹¹⁵ IE hace uso de la antigua cabecera HTTP "X-Content-Security-Policy".

Mediante la anterior política CSP, el navegador web no cargará ningún recurso, ni enviará ningún formulario, que no haga uso de HTTPS. La directiva "default-src" especifica la política de seguridad que se debe aplicar por defecto, es decir, para todos los tipos de recursos que no disponen de directivas que definan una política CSP específica para otros contenidos (fuentes, imágenes y favicons, multimedia - audio y vídeo, elementos hijos - como frames e iframes, objetos, scripts - Javascript, estilos, conexiones - como AJAX (XMLHttpRequest) y WebSockets, etc.).

Adicionalmente, al igual que con otras cabeceras HTTP de seguridad como HPKP (ver apartado "3.7.4. Forzando el uso estricto (por defecto) de HTTPS: HSTS"), y de nuevo para minimizar el posible impacto de una configuración incorrecta al comenzar el despliegue de CSP, se recomienda empezar haciendo uso de las capacidades de notificación de CSP mediante la cabecera extendida "...-Report-Only" y la directiva "report-uri". Se recomienda hacer pruebas inicialmente con esta cabecera para identificar el correcto funcionamiento de la política. Si un navegador o cliente web recibe la política CSP Report-Only¹¹⁶, y el contenido web referencia un recurso que no hace uso de HTTPS, no bloqueará el tráfico HTTPS (tal como ocurre con la cabecera CSP estándar, que sí aplica la política CSP sin excepción), pero generará un informe dirigido a la URL indicada en dicha cabecera por la directiva "report-uri". En este escenario, de manera efectiva, los clientes web son empleados como sensores para la detección de acceso a recursos mediante HTTP:

```
Content-Security-Policy-Report-Only: default-src https;; form-action https;; report-uri="https://dominioinformes.es/csp-report"
```

A diferencia de HPKP, en el caso de CSP sí se puede emplear para la directiva "report-uri" una referencia web HTTPS asociada al mismo servidor web que ha enviado la cabecera CSP (o a uno diferente), ya que en este caso no habrá conflictos en el navegador web para remitir el informe (tal como podía ocurrir en HPKP).

En el caso de CSP es posible hacer uso simultáneo de las cabeceras "Content-Security-Policy" y "Content-Security-Policy-Report-Only". La primera de ellas aplicaría de manera efectiva la política CSP actual, y la segunda permitiría verificar posibles violaciones de una nueva política CSP que se quiera introducir en el entorno web.

Se recomienda comenzar haciendo uso de la cabecera CSP en un recurso (o un conjunto reducido de recursos) web concreto(s), hasta verificar su correcta configuración, con el objetivo de evitar recibir multitud de informes de violación de la política CSP. A diferencia de HSTS, CSP es una política que se aplica individualmente para cada página web (y que no es cacheada por el navegador web), por lo que todos y cada uno de los recursos web que se desea proteger deben de incluir la cabecera CSP en su respuesta. El impacto si se comenten errores de configuración de la política es mínimo, ya que sólo afectarían a cada recurso individualmente y una única vez.

El objetivo final de la migración a HTTPS es la modificación de todos los contenidos del entorno, servidor o aplicación web para que todas las referencias hagan uso de HTTPS en lugar de HTTP. Sin embargo, este proceso de migración puede ser complejo, ya que es necesario actualizar todas las referencias tanto estáticas (contenidos en ficheros) como dinámicas (es decir,

¹¹⁶ <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy-Report-Only>

contenidas en bases de datos), hacia "https://". Se recomienda hacer uso de referencias relativas (como "./directorio/recurso.html"), absolutas (como "/directorio/recurso.html") o, al menos, emplear referencias que no especifiquen el protocolo (como "//dominio.es/directorio/recurso.html"), para generalizar el uso de HTTPS (y evitar referencias directas a HTTP). El uso de las capacidades de notificación de CSP mencionadas previamente puede ser crucial para identificar los recursos que aún hacen uso de HTTP ("http://")¹¹⁷.

Adicionalmente, durante el proceso de migración (y posteriormente), la directiva "upgrade-insecure-requests" [Ref - 36] (asociada a CSP [Ref - 33]) puede ser de gran ayuda. Mediante "upgrade-insecure-requests", el entorno web indicará a los navegadores web que disponen de soporte para esta directiva¹¹⁸ (como por ejemplo Firefox 42, Chrome 43 u Opera 30) que actualicen cualquier referencia HTTP hacia HTTPS, tanto para los contenidos del propio entorno web como de terceros, antes de enviar la petición asociada. El único problema que puede surgir al hacer uso de esta directiva es que los recursos que no están disponibles mediante HTTPS no podrán ser cargados. Los navegadores que disponen de soporte para esta directiva lo indican mediante la cabecera "Upgrade-Insecure-Requests: 1" en sus peticiones HTTP.

```
Content-Security-Policy: upgrade-insecure-requests; default-src https;; form-action https;; report-uri="<URL>"
```

Otra directiva asociada a "upgrade-insecure-requests", pero más restrictiva, es "block-all-mixed-content" [Ref - 31]¹¹⁹ (asociada a CSP 3 [Ref - 35]). Mediante "block-all-mixed-content", el entorno web indicará a los navegadores web que disponen de soporte para esta directiva que no carguen ningún contenido mediante HTTP si la página web ha sido obtenida mediante HTTPS. Esta directiva no permitirá la realización de ninguna petición mediante HTTP. En el caso de estar haciendo uso de "upgrade-insecure-requests" junto a "block-all-mixed-content", esta última directiva no será de aplicación, siendo más permisivo el navegador web en la carga de contenidos mediante HTTP, e intentando al menos obtenerlos mediante HTTPS.

```
Content-Security-Policy: block-all-mixed-content;
```

NOTA: La presencia de múltiples cabeceras CSP en una misma respuesta HTTP hace que el navegador web combine todas las políticas recibidas, aplicando de manera restrictiva el mínimo común múltiplo de todas ellas¹⁰⁶.

En resumen, y como política sencilla de ejemplo centrada en HTTPS, CSP puede ser empleado para restringir el uso de contenidos mixtos de terceros, forzando el uso de HTTPS, un escenario donde HSTS no puede imponer restricciones (ver apartado "3.7.4. Forzando el uso estricto (por defecto) de HTTPS: HSTS"), haciendo uso de la siguiente política de seguridad:

```
Content-Security-Policy: upgrade-insecure-requests; default-src https: 'unsafe-inline' 'unsafe-eval'; form-action https;; connect-src https: wss;; report-uri="https://dominioinformes.es/csp-report"
```

¹¹⁷ <https://scotthelme.co.uk/fixing-mixed-content-with-csp/>

¹¹⁸ <http://caniuse.com/#feat=upgradeinsecurerequests>

¹¹⁹ <https://w3c.github.io/webappsec-mixed-content/#block-all-mixed-content>

NOTA: La política del ejemplo superior¹²⁰ especifica que cualquier referencia de AJAX (XMLHttpRequest), fetch, EventSource, y, especialmente, WebSockets, debe también hacer uso de HTTPS o WSS (WebSockets Secure), mediante "connect-src".

3.7.7 Ausencia de funcionalidades avanzadas en entornos HTTP

Algunos navegadores y clientes web están evaluando la posibilidad y aplicando medidas para no permitir el acceso a funcionalidades avanzadas disponibles en el navegador para sitios web basados en HTTP, sólo pudiendo ser éstas utilizadas a través de HTTPS (englobado dentro de los denominados orígenes seguros) [Ref - 37].

Dentro de la lista de funcionalidades avanzadas propuestas por Chrome, se encuentran las relacionadas con la geolocalización, la orientación y el movimiento del dispositivo del usuario, el acceso a la cámara, notificaciones, etc.

3.7.8 Evitando cachear contenidos sensibles

El objetivo del presente informe es que todo el tráfico asociado a las tecnologías web haga uso de HTTPS, siendo necesario por tanto diferenciar entre contenido sensible y contenido público. Como sugerencia general, se recomienda no almacenar localmente en disco, o cachear, en los clientes web ningún contenido sensible intercambiado por el entorno, servidor o aplicación web a través de HTTPS.

Para ello, es necesario hacer uso de diferentes cabeceras HTTP (con directivas de caché) que indiquen a los navegadores y clientes web estándar que no deben almacenar una copia local, o cachear, ciertos contenidos más críticos¹²¹. Las cabeceras HTTP a emplear varían en función de los navegadores y clientes web empleados para acceder a los contenidos web (debido a que no todos ellos implementan correctamente los estándares¹²²), por lo que se recomienda especificar todas las siguientes directivas en todos los recursos con contenido sensible intercambiados mediante HTTPS:

```
Cache-Control: no-store, no-cache, must-revalidate, private
Pragma: no-cache
Expires: 0
```

Las directivas para el control de caché de contenidos web están definidas actualmente en el RFC 7234 [Ref - 38].

Por el contrario, los recursos y contenidos públicos deben de ser clasificados convenientemente para que puedan ser cacheados incluso al hacer uso de HTTPS:

```
Cache-Control: public
```

¹²⁰ La política de ejemplo mostrada podría ser más estricta, y no permitir la carga y la utilización de contenidos propios mediante HTTPS catalogados como "unsafe", es decir, para recursos como scripts (<script>) y estilos (<style>) definidos en la propia página web ("unsafe-inline"), así como llamadas Javascript a "eval ()" y similares ("unsafe-eval"), eliminando de "default-src https:" las directivas 'unsafe-inline' y 'unsafe-eval'. Estas directivas no están directamente relacionadas con HTTPS.

¹²¹ https://securityevaluators.com/knowledge/case_studies/caching/

¹²² <http://stackoverflow.com/questions/49547/how-to-control-web-page-caching-across-all-browsers/5493543#5493543>

3.7.9 Inspección del tráfico HTTPS

Aunque de manera global se recomienda el uso exclusivo de HTTPS para el intercambio de contenidos web, objetivo principal del presente informe, podrían existir escenarios dónde puede ser necesario inspeccionar o monitorizar el tráfico web con autorización, por ejemplo, para detectar la presencia de tráfico o contenidos maliciosos, o de ataques que podrían ser encubiertos a través del tráfico cifrado.

La inspección (o interceptación) del tráfico HTTPS, o TLS, especialmente en la actualidad dónde la tendencia es hacer un uso generalizado de estos protocolos, es siempre controvertida, existiendo tanto motivos a favor como en contra. Habitualmente, esta capacidad es necesaria por parte de las soluciones empresariales de monitorización de tráfico (como sistemas de detección de intrusos, IDS) y de las redes de comunicaciones, en concreto, en las redes internas y privadas de las organizaciones, o en las infraestructuras perimetrales de conexión a Internet. Su implantación se puede llevar a cabo tanto desde la propia red (empleando proxies de interceptación) y/o en los navegadores o clientes web, según su implementación.

Algunos escenarios específicos donde la inspección del tráfico HTTPS puede ser positiva desde el punto de vista defensivo serían la identificación de descargas de malware, la detección de tráfico dirigido a servidores de Comando & Control (C&C) de una *botnet*, escenarios de exfiltración de datos desde una organización, etc. Sin las capacidades de inspeccionar los contenidos de las conexiones HTTPS, sólo es posible obtener detalles de los metadatos de la conexión, es decir, las direcciones IP involucradas, los nombres en el DNS y los diferentes servicios empleados (si fuesen distintos a TCP/443), u obtener ciertos patrones de tráfico. En cualquier caso, las soluciones de inspección deben de ser completamente transparentes para el usuario y no afectar al resto de recomendaciones de concienciación y buenas prácticas de comportamiento que se han transmitido a los usuarios para poder identificar ataques reales sobre HTTPS¹²³.

Desafortunadamente, el uso generalizado de soluciones de inspección del tráfico HTTPS está ampliamente generalizado (entre un 4 y un 11% en conexiones públicas HTTPS en Internet), y en numerosas ocasiones su correcta implementación disminuye el nivel de seguridad de las comunicaciones HTTPS (en un 62% de los casos), por ejemplo, haciendo uso de algoritmos o *suites* criptográficas débiles, o incluso introduce vulnerabilidades de seguridad adicionales (en un 58% de los casos), por ejemplo, no validando correctamente los certificados digitales¹²⁴. Si se está evaluando la implantación de una solución de este tipo, se recomienda evaluar cuidadosamente las ventajas e inconvenientes de la misma. En caso de disponer de cualquier solución de inspección de tráfico HTTPS, se recomienda al menos evaluar parcialmente el comportamiento de dicha solución mediante el servicio "BadSSL" [Ref - 20].

Por otro lado, la inspección del tráfico HTTPS es por naturaleza contraria al conjunto global de recomendaciones proporcionados a lo largo del presente informe, inutilizando los diferentes mecanismos y capacidades de protección sugeridos, y facilitando la posibilidad de disponer de acceso al tráfico sensible que debería estar protegido, para su inspección y/o manipulación, y a

¹²³ <https://www.us-cert.gov/ncas/alerts/TA17-075A>

¹²⁴ <https://jhalderm.com/pub/papers/interception-ndss17.pdf>

suplantar a los extremos de la comunicación¹²⁵. Las soluciones de inspección de tráfico llevan a cabo de manera técnica y efectiva (aunque se realice con autorización, e independientemente del tráfico intercambiado: personal, profesional, etc.) un ataque de Man-in-the-Middle (MitM), amenaza que de hecho se intenta mitigar mediante el uso de HTTPS.

Debe tenerse en cuenta que con la sofisticación y avances actuales de los mecanismos de protección asociados a HTTPS, las soluciones de inspección no sólo requieren modificaciones en la arquitectura de la red, sino también en la gestión de las conexiones HTTPS en los extremos de la comunicación, tanto cliente como servidor, ya que si no se dificultará su activación (por ejemplo, si se hace uso de Certificate Transparency o de HPKP, en este último caso, salvo que se manipulen localmente las CAs en los navegadores y clientes web, tanto corporativos como personales).

Por último, incluso en la definición y el proceso de ratificación del estándar TLS 1.3 [Ref - 22], diferentes grupos (como, por ejemplo, de la industria financiera¹²⁶) han reflejado su preocupación debido a la eliminación en TLS 1.3 del intercambio de claves mediante RSA (permitiendo únicamente DHE y ECDHE empleando forward secrecy), lo que elimina la posibilidad de inspeccionar y descifrar el tráfico TLS en las redes internas y centros de datos. Por este motivo, se ha propuesto incluso un nuevo estándar que permita una configuración de los servidores HTTPS que haga uso de una clave DH o ECDH estática, en lugar de efímera, para todas las conexiones TLS, lo que permitiría monitorizar el tráfico asociado a dichos servidores¹²⁷.

3.7.10 Cookies seguras

Independientemente de si se hace uso de otros mecanismos de seguridad descritos a lo largo del presente informe, como por ejemplo HSTS (ver apartado "3.7.4. Forzando el uso estricto (por defecto) de HTTPS: HSTS"), se recomienda establecer la directiva "Secure" en todas las cookies empleadas por la aplicación web con el objetivo de que las mismas sólo sean enviadas por los navegadores y clientes web a través de una conexión HTTPS, y nunca por HTTP.

Incluso haciendo uso de HSTS, existen escenarios donde podría ser posible obtener el valor de una cookie si no dispone de la directiva "Secure", como por ejemplo en la primera conexión al entorno web, o al establecerse conexiones a otros servidores del mismo dominio. Es especialmente relevante el uso de la directiva "includeSubDomains" de HSTS si se hace uso de cookies de dominio ("domain"), cuyo alcance y aplicación incluye la totalidad del dominio (en lugar de únicamente la aplicación web que ha fijado el valor de la cookie).

Se recomienda hacer uso del resto de directivas de protección de las cookies¹²⁸, aunque estas no estén directamente vinculadas a HTTPS (ej. HttpOnly o Same-site cookies¹²⁹). En el caso de la propuesta asociada a los prefijos de cookies¹³⁰, tanto el prefijo "__Secure-" como el prefijo "__Host-", sí requieren que la cookie disponga de la directiva "Secure" y haya sido servida mediante HTTPS.

¹²⁵ <https://blog.hboeck.de/archives/875-TLS-interception-considered-harmful-video-and-slides.html>

¹²⁶ <https://www.ietf.org/mail-archive/web/tls/current/msg21275.html>

¹²⁷ <https://tools.ietf.org/html/draft-green-tls-static-dh-in-tls13-00>

¹²⁸ https://www.owasp.org/index.php/Session_Management_Cheat_Sheet

¹²⁹ <https://tools.ietf.org/html/draft-west-first-party-cookies-07>

¹³⁰ <https://tools.ietf.org/html/draft-ietf-httpbis-cookie-prefixes-00>

3.7.11 Recomendaciones para los contenidos y aplicaciones web HTTPS

El resumen de recomendaciones para los contenidos y aplicaciones web HTTPS incluye:

- El uso de HTTPS permite a los sitios web tener más relevancia en el posicionamiento (o ranking) de los buscadores web, como Google.
- Los buscadores web priorizan la indexación de páginas web por defecto empleando la versión HTTPS de los sitios web, por lo que se recomienda no añadir la directiva "noindex" para las URLs que hacen uso de HTTPS.
- La política establecida en el fichero "robots.txt" debe permitir la indexación de los contenidos HTTPS del servidor web por parte de los buscadores web.
- Se recomienda hacer uso de la etiqueta "rel=canonical" para definir que los recursos canónicos o de referencia dentro del servidor web son los que hacen uso de HTTPS (tanto únicos como duplicados).
- Evaluar en detalle la implementación de mejoras de rendimiento de TLS, como las optimizaciones del TLS *handshake* (1-RTT), del TLS record size y de la cadena de certificación, TLS session resumption o TLS session caching (TLS session tickets), TLS False Start, TCP Fast Open, TLS early termination, OCSP Stapling, etc.
- Hacer uso de mecanismos de compresión Brotli mediante HTTPS.
- Hacer uso de TLS False Start junto a los mecanismos de seguridad que permiten utilizarla con confianza (cifrado de 128 bits, con forward secrecy y NPN/ALPN).
- Hacer uso de TLS session resumption (o TLS session caching) estableciendo el periodo máximo en el que se cachearán las sesiones TLS, por ejemplo, a un día (24 horas). No compartir la caché de sesiones TLS entre diferentes entornos web de hosting compartidos.
- No hacer uso de TLS session tickets, y en caso de utilizarlos, hacer uso de claves de ticket robustas, que no sean reutilizadas frecuentemente, y que no sean compartidas entre diferentes entornos web de hosting compartidos.
- Se recomienda hacer uso de la directiva "preconnect" de la etiqueta HTML <link> para indicar a los clientes web que abran prematuramente una conexión HTTPS, para cargar recursos que serán referenciados posteriormente.
- Se recomienda redirigir automáticamente todo el tráfico HTTP hacia HTTPS mediante una redirección 301. En ningún caso los accesos a la versión HTTPS del entorno web deben ser redireccionados a la versión HTTP.
- Hacer uso de HSTS (HTTP Strict Transport Security) para declarar que el servidor web sólo está accesible mediante HTTPS.
- En el caso de hacer uso de HSTS, se recomienda emplear un valor de "max-age" de al menos 10886400 (18 semanas), y preferiblemente de 6 o 12 meses ("31536000"). HSTS puede ser introducido incrementando progresivamente el valor de "max-age".
- HSTS debería de ser implementado para la totalidad del dominio mediante la directiva "includeSubDomains".

- Se debe de incluir la directiva "preload" en la configuración HSTS y remitir el dominio para que sea incluido en la lista precargada pública de HSTS.
- Se recomienda revisar todos los contenidos y código asociados al entorno, aplicación y páginas web (estáticas y dinámicas) y sustituir todas las referencias a "http://" por "https://" (o por referencias relativas o absolutas sin especificar el protocolo), evitando el uso de contenidos mixtos.
- Verificar sula validez e integridad de los recursos de terceros a través de SubResource Integrity (SRI).
- Hacer uso de Content Security Policy (CSP) para mitigar el riesgo de utilización de contenidos mixtos, indicando que por defecto todos los recursos deberían hacer uso de HTTPS. En su defecto, hacer uso de las capacidades de notificación de CSP para identificar el uso de recursos web que continúan empleando HTTP.
- Se recomienda empezar haciendo un uso limitado de la cabecera CSP en un recurso concreto, e ir incrementando progresivamente su utilización al resto de recursos web.
- En concreto, se pueden mejorar la política CSP mediante las directivas "upgrade-insecure-requests" (más permisiva) y "block-all-mixed-content" (más estricta), para asegurar el uso de HTTPS en todo el entorno web.
- Se recomienda hacer uso de referencias relativas o absolutas a los recursos web, pero que en cualquier caso no especifiquen el protocolo, para generalizar el uso de HTTPS (y evitar referencias directas a HTTP).
- Se recomienda evitar que los clientes web almacenen localmente en disco, en la caché, ningún contenido sensible intercambiado por el entorno web a través de HTTPS, haciendo uso de diferentes cabeceras HTTP con directivas de caché.
- Por el contrario, los recursos y contenidos públicos deben de ser clasificados convenientemente para que puedan ser cacheados incluso al hacer uso de HTTPS.
- En el caso de hacer uso de alguna solución de inspección de tráfico HTTPS, se debe verificar que es transparente para el usuario, no afectando al resto de buenas prácticas, y es necesario evaluar cuidadosamente las ventajas e inconvenientes de la misma, verificando especialmente su correcta implementación para que no disminuya el nivel de seguridad de las comunicaciones HTTPS o introduzca vulnerabilidades de seguridad adicionales.
- Se recomienda establecer la directiva "Secure" en todas las cookies empleadas por la aplicación web con el objetivo de que las mismas sólo sean enviadas a través de una conexión HTTPS.
- Se recomienda hacer uso del resto de directivas de protección de las cookies, tanto las no están directamente vinculadas a HTTPS (ej. HttpOnly o Same-site cookies), como las que sí lo están (ej. prefijos "__Secure-" y "__Host-").

3.8 Vulnerabilidades en HTTPS

El presente apartado proporciona una breve lista, y una descripción muy resumida, de las vulnerabilidades y ataques (término empleado de aquí en adelante) más relevantes sufridos por

los protocolos e implementaciones asociados a HTTPS en los últimos años [Ref - 10], y referenciados o mencionados a lo largo del presente informe.

Algunos de estos ataques podían ser mitigados simplemente actualizando los componentes empleados en la implementación de HTTPS (librerías, servidores web, etc.), o modificando una configuración insegura, mientras que otros han provocado cambios drásticos en el diseño del protocolo TLS, o han hecho obsoletos (y permanentemente inseguros) versiones de los protocolos y/o de los algoritmos y *suites* criptográficas.

Por tanto, como recomendación general, siempre se debe disponer de la versión más actualizada (que al menos solucione las vulnerabilidades de seguridad conocidas) de las librerías TLS, servidores y aplicaciones web empleados en la implementación de HTTPS.

Se dispone de una lista de vulnerabilidades y eventos relevantes en la industria relacionados con HTTPS, desde 1994 hasta la actualidad, en "SSL/TLS and PKI History" [Ref - 39].

Los mecanismos de renegociación insegura de TLS (2009) permiten la realización de ataques MitM y de denegación de servicio (DoS, Denial of Service) sobre el servidor web (ver apartado "3.4.3. Renegociación").

BEAST (2011) es un ataque contra TLS 1.0 (y versiones previas de SSL) y los algoritmos de cifrado en modo CBC (AES, DES, etc.), debido al uso de vectores de inicialización (IVs, Initialization Vectors) predecibles en los cifradores por bloques, por lo que se recomienda hacer uso de TLS 1.1, dar prioridad a *suites* criptográficas basadas en cifradores de flujo (en lugar de bloque) como RC4 (aunque no está recomendado en la actualidad debido a otras debilidades descubiertas en 2013 y 2015, ver RFC 7465¹³¹) y, preferiblemente, AES-GCM con TLS 1.2¹³².

CRIME (2012) es un ataque contra los mecanismos de compresión a nivel de TLS que desvela información asociada a la sesión cifrada, por lo que se recomienda deshabilitar las capacidades de compresión de TLS del servidor web¹³³.

Lucky 13 (2013) es un ataque, de nuevo, contra las *suites* criptográficas de cifrado en modo bloque con CBC, basado en el análisis estadístico e identificación de pequeñas diferencias en los tiempos de las operaciones de cifrado (ataques conocidos como *padding oracle attacks*). De nuevo, se recomienda evitar el uso de *suites* CBC y emplear en su lugar *suites* GCM con TLS 1.2.

BREACH (2013) es un ataque contra los mecanismos de compresión por encima de TLS (extendiendo el ataque CRIME previo), por lo que se recomienda deshabilitar las capacidades de compresión web estándar de HTTP del servidor web (ampliamente utilizadas) y, en caso de no ser posible, aplicar otras técnicas de mitigación más elaboradas¹³⁴, como por ejemplo HTTP Chunked Encoding. También se puede hacer uso en su lugar de las capacidades de compresión Brotli. Otra alternativa es deshabilitar los mecanismos de compresión del servidor o aplicación web únicamente cuando se reciben peticiones desde otros sitios web, en base a la cabecera Referrer (o si ésta no está disponible).

¹³¹ <https://tools.ietf.org/html/rfc7465>

¹³² <https://blog.qualys.com/ssllabs/2011/10/17/mitigating-the-beast-attack-on-tls>

¹³³ <https://blog.qualys.com/ssllabs/2012/09/14/crime-information-leakage-attack-against-ssltls>

¹³⁴ <https://blog.qualys.com/ssllabs/2013/08/07/defending-against-the-breach-attack>

TIME (2013) es un ataque similar a BREACH, también centrado en los mecanismos de compresión, pero para el que no se publicó una prueba de concepto o herramienta práctica de ataque.

El ataque de triple *handshake* (2014) permite la manipulación de la sesión TLS cuando se hace uso de certificados digitales cliente, es decir, de las capacidades de renegociación (seguras) de TLS. Asimismo, este ataque demostró cómo era posible para un servidor malicioso seleccionar parámetros DH inseguros que ni siquiera eran números primos, lo que permitiría romper fácilmente el intercambio de clave DH. Las implementaciones modernas de TLS, tanto cliente como servidor, han incorporado mecanismos de defensa, por lo que se recomienda actualizar todos los componentes asociados a HTTPS.

Heartbleed (2014) es un ataque específico de la librería OpenSSL que permitía obtener fragmentos de la memoria del servidor (en bloques de 64 KB) mediante la manipulación de la funcionalidad de TLS heartbeat. Heartbleed puede ser mitigado actualizando la versión de OpenSSL empleada.

POODLE (2014) es un ataque contra SSL 3.0 que permite calcular el contenido de una conexión cifrada con SSL (especialmente con algoritmos de cifrado en modo CBC, una vez más, debido a la dificultad de implementarlos correctamente), así como forzar un ataque mediante técnicas de *downgrade* de la versión del protocolo SSL/TLS, para pasar de usar TLS 1.x a usar SSL 3.0¹³⁵¹³⁶.

Logjam (2015) es un ataque sobre los parámetros Diffie-Hellman (DH), y la posibilidad de hacer uso de un intercambio de claves débil mediante DH¹³⁷, como por ejemplo DH de 768 o 512 bits (o incluso, para atacantes más avanzados, DH de 1.024 bits si se hace uso de grupos DH conocidos). Un atacante puede forzar, mediante un ataque MitM y técnicas de *downgrade*, el uso de una *suite* criptográfica débil que le permita descifrar el tráfico intercambiado. Logjam es un ataque similar a FREAK, pero centrado en el intercambio de claves mediante DHE en lugar de mediante RSA.

FREAK (2015) es un ataque centrado en las implementaciones TLS de los clientes que permite interceptar las comunicaciones HTTP y forzar (mediante técnicas de *downgrade*) a que se usen algoritmos de cifrado débiles, por ejemplo, RSA de 512 bits. Afecta a cualquier servidor que hace uso de suites criptográficas permitidas para su exportación¹³⁸.

DROWN (2016) es un ataque contra SSL 2.0 que permite descifrar el tráfico HTTPS mediante la realización de peticiones a un servidor con SSL 2.0 que utiliza la misma clave¹³⁹. Lo novedoso de esta vulnerabilidad es que permite incluso atacar servidores que no soportan SSL 2.0 si reutilizan la misma clave que otro servidor que sí soporta esta versión del protocolo SSL. Pese a haber sido publicado en 2016, lamentablemente se estimó que, en el momento de su publicación, el 33% de los servidores web públicos HTTPS estaban afectados por ella, debido a que aún proporcionaban soporte para SSL 2.0.

¹³⁵ <https://security.googleblog.com/2014/10/this-poodle-bites-exploiting-ssl-30.html>

¹³⁶ <https://blog.qualys.com/ssllabs/2014/10/15/ssl-3-is-dead-killed-by-the-poodle-attack>

¹³⁷ <https://weakdh.org>

¹³⁸ <https://freakattack.com> (<https://censys.io/blog/freak>)

¹³⁹ <https://drownattack.com>

3.8.1 Recomendaciones de las vulnerabilidades en HTTPS

El resumen de recomendaciones de las vulnerabilidades en HTTPS, en caso de no haber sido ya mencionadas en apartados previos, incluye:

- Disponer de la versión más actualizada (que al menos solucione las vulnerabilidades de seguridad conocidas) de las librerías TLS, servidores y aplicaciones web empleados en la implementación de HTTPS.
- Deshabilitar las capacidades de compresión de TLS del servidor web.
- Evitar el uso de suites criptográficas de cifrado en modo bloque con CBC.
- Deshabilitar las capacidades de compresión web estándar de HTTP del servidor web y, en caso de no ser posible, aplicar otras técnicas de mitigación más elaboradas o específicas. También se puede hacer uso en su lugar de las capacidades de compresión Brotli.

3.9 Compatibilidad con los navegadores y clientes web

Aunque desde el punto de vista de seguridad convendría premiar los mecanismos de seguridad y de protección más modernos y avanzados, es necesario encontrar un equilibrio entre el nivel de seguridad proporcionado mediante HTTPS y la compatibilidad o interoperabilidad con el mayor número de clientes web posible.

Para ello se recomienda, antes de implementar una nueva medida de seguridad relacionada con HTTPS, evaluar el soporte existente para la misma en la mayoría de navegadores o clientes web (agentes de usuario), tanto tradicionales (por ejemplo, Chrome, Edge, Firefox, IE, Opera, Safari, etc.) como móviles (por ejemplo, navegador web Android (<= 4.3), Chrome (Android e iOS), Safari Mobile (iOS), Firefox mobile, etc.).

Mediante el servicio web "Can I use...?" [Ref – 7] es posible evaluar el soporte para multitud de tecnologías y capacidades web en los navegadores y clientes web, como, por ejemplo:

- Diferentes versiones del protocolo TLS (1.0, 1.1, 1.2, etc.) y otras funcionalidades específicas de TLS, como SNI (ver apartado "3.4.2. SNI (Server Name Indication)"), directivas como "preconnect" (ver apartado "3.7.2.3. Preconnect"), o suites criptográficas específicas, como "ChaCha20-Poly1305":
 - <http://caniuse.com/#search=TLS>
 - <http://caniuse.com/#search=preconnect>
 - <http://caniuse.com/#feat=chacha20-poly1305>
- Soporte para HSTS o STS (HTTP Strict Transport Security):
 - <http://caniuse.com/#search=hsts>
- Soporte para HPKP o PKP (HTTP Public Key Pinning):
 - <http://caniuse.com/#search=hpkp>
- Soporte para compresión Brotli:
 - <http://caniuse.com/#search=brotli>
- Soporte para HTTP/2:
 - <http://caniuse.com/#feat=http2>

- Soporte para (diferentes versiones) de CSP (Content Security Policy) y para otras características asociadas:
 - <http://caniuse.com/#search=csp>
 - <http://caniuse.com/#feat=contentsecuritypolicy>
 - <http://caniuse.com/#feat=contentsecuritypolicy2>
 - <http://caniuse.com/#feat=upgradeinsecurerequests>

Adicionalmente, mediante el servicio web "User Agent Capabilities" [Ref – 6] es posible identificar el soporte de los diferentes navegadores y clientes web, incluyendo también *bots* como los empleados por los buscadores de Internet (como Baidu, Bing, Googlebot, Yahoo!, Yandexbot, etc.) o herramientas y librerías SSL/TLS (Java, OpenSSL, Tor, etc.), para las capacidades fundamentales de seguridad asociadas a HTTPS: versión 1.2 de TLS (ver apartado "3.4.1. Versiones del protocolo SSL y TLS"), SNI (ver apartado "3.4.2. SNI (Server Name Indication)"), forward secrecy (ver apartado "3.5.1. Forward secrecy"), Stapling (ver apartado "3.6.3. OCSP Stapling") y tickets de sesión (ver apartado "3.7.2.2. TLS session resumption o caching, y TLS session tickets").

Con el objetivo de evaluar el posible impacto que puede tener un cambio en los mecanismos de seguridad de HTTPS del servidor web sobre los clientes web que hacen uso del mismo, se recomienda analizar los logs del servidor web e identificar, en base al "User Agent" (agente de usuario), qué porcentaje de clientes web de un tipo determinado hacen uso del servicio, pudiendo así evaluarse y cuantificarse el impacto negativo que podría tener el cambio planificado.

4. DECÁLOGO DE RECOMENDACIONES

	Decálogo de seguridad para la implementación de HTTPS
1	<p>Configurar el entorno, servidor o aplicación web para que se lleve a cabo una redirección automática de tipo 301 desde HTTP hacia HTTPS para cualquier petición web.</p> <p>Revisar todos los contenidos y código asociados al entorno, aplicación y páginas web (estáticas y dinámicas) y sustituir todas las referencias a "http://" por "https://" (o por referencias relativas o absolutas sin especificar el protocolo), evitando el uso de contenidos mixtos.</p>
2	Hacer uso de HSTS (HTTP Strict Transport Security) para declarar que el servidor web sólo está accesible mediante HTTPS.
3	Hacer uso de Content Security Policy (CSP) para evitar la utilización de contenidos mixtos, indicando que, por defecto, todos los recursos deberían obtenerse mediante HTTPS. Ampliar la política de CSP mediante las directivas "upgrade-insecure-requests" o "block-all-mixed-content".
4	Hacer uso únicamente del protocolo TLS, y preferiblemente de las versiones 1.1 y 1.2 del mismo (y de la versión TLS 1.3 próximamente).
5	<p>Hacer uso de claves criptográficas ECDSA de 256 bits y/o RSA de 2.048 bits para los certificados digitales (X.509).</p> <p>Evaluar y seleccionar cuidadosamente el tipo de certificado digital a emplear, firmado mediante SHA-256, la CA que lo emitirá, el periodo de renovación, configurar adecuadamente la cadena completa de certificación, seleccionar la entidad (o entidades) representada(s) por el certificado, y asegurar su validez en todo momento.</p>
6	<p>Utilizar algoritmos de intercambio de claves que proporcionen <i>forward secrecy</i>, preferiblemente ECDHE (256 bits), o alternativamente DHE (2.048 bits), frente a RSA.</p> <p>Configurar el servidor web priorizando las suites criptográficas más robustas, listadas en orden de preferencia según el nivel de seguridad que ofrecen y que comiencen por "ECDHE-ECDSA-...", "ECDHE-RSA-..." o "DHE-RSA-...".</p>
7	Hacer uso de algoritmos de cifrado que proporcionen cifrado autenticado AEAD, como por ejemplo AES en modo GCM o ChaCha20 junto a Poly1305. Se deberían utilizar algoritmos de cifrado simétrico que proporcionen, al menos, 128 bits de seguridad. Se deberían utilizar algoritmos de <i>hashing</i> basados en SHA-256 (o superior).
8	<p>Los certificados digitales deben de incluir mecanismos de revocación como CRLs u OCSP o, preferiblemente, OCSP Stapling.</p> <p>Hacer uso de una CA que permita añadir la directiva OCSP Must-Staple a los certificados digitales. Complementariamente, hacer uso de las capacidades de notificación de OCSP Expect-Staple.</p>
9	<p>Hacer uso de HPKP (HTTP Public Key Pinning), o al menos de sus capacidades de notificación para identificar el uso de certificados digitales ilegítimos.</p> <p>Hacer uso de los nuevos mecanismos de seguridad para la emisión de certificados, como Certificate Transparency (CT), incluyendo la cabecera Expect-CT y utilizando las capacidades de los servidores web para proporcionar respuestas SCT, y de los registros DNS CAA.</p>
10	Disponer de la versión más actualizada (que al menos solucione las vulnerabilidades de seguridad conocidas) de las librerías TLS, servidores y aplicaciones web empleados en la implementación de HTTPS.

Figura 8.- Decálogo de seguridad para la implementación de HTTPS

ANEXO A. REQUISITOS PARA EL ANÁLISIS DE SITIOS WEB HTTPS

Los siguientes requisitos deben ser tenidos en cuenta por los responsables de los entornos web (servidores, aplicaciones, etc.) que implementan el protocolo HTTPS, de cara a su análisis dentro de los estudios realizados por el CCN-CERT para evaluar la implementación de HTTPS y el estado actual del ecosistema HTTPS en el sector público.

El estudio se centra única y exclusivamente en el análisis de la implementación de SSL/TLS (Secure Sockets Layer/Transport Layer Security) sobre los servicios web (HTTP y HTTPS), no extendiéndose el uso de SSL/TLS a otros servicios como, por ejemplo, SMTPS, IMAPS, POP3S, redes privadas virtuales (VPNs, Virtual Private Networks), etc.

Aunque parezca obvio, en primer lugar, se debe asegurar la disponibilidad del entorno web objetivo en todo momento, verificándose que al menos está disponible a través del servicio de resolución de nombres o sistema de nombres de dominio (DNS, Domain Name System), y que al menos está ofreciendo servicio web HTTPS a través del puerto TCP/443 (debiendo estar este puerto siempre abierto). Opcionalmente, se evaluará si el entorno web objetivo está también ofreciendo servicio web HTTP a través del puerto TCP/80, pudiendo estar este puerto abierto, filtrado o cerrado.

El análisis se llevará a cabo a través de conexiones TCP/IP empleando la versión 4 del protocolo IP (IPv4), por lo que el entorno web objetivo debe de disponer de presencia y de una dirección IP pública (IPv4) en Internet, independientemente de su presencia en otras redes basadas en la versión 6 del protocolo IP (IPv6).

Se debe de identificar el entorno web objetivo mediante su nombre o *hostname* (ej. *www.dominio.es*), no debiendo ser identificado por su dirección IP (ej. *10.10.10.10*) ya que, por ejemplo, el uso de la cabecera HTTP de seguridad HSTS no está permitido para direcciones IP.

No se debe ofrecer el servicio HTTPS a través de un puerto no estándar distinto a TCP/443 (ej. TCP/444).

Uno de los primeros comportamientos a evaluar será la existencia de redirecciones automáticas de las conexiones establecidas mediante HTTP (TCP/80) hacia HTTPS (TCP/443).

Asimismo, se evaluará la validez de la cadena de certificación o de certificados digitales (X.509) empleados por el entorno web, desde el certificado de la CA raíz hasta el certificado propio del entorno web. Entre otros detalles, se evaluará:

- El nombre de la entidad asociada al certificado digital y su validez respecto al identificador del entorno web analizado (*hostname*).
- El periodo de validez (o periodo de expiración) del certificado digital.
- La validez de la cadena de certificación, desde la CA raíz hasta el certificado digital propio, pasando por las diferentes CAs intermedias.

Adicionalmente, se evaluarán múltiples detalles técnicos asociados a HTTPS de los entornos web objetivo como, por ejemplo, el estado de la implementación de estándares como HTTP Strict Transport Security (HSTS), HTTP Public Key Pinning (HPKP), OSCP Stapling, las diferentes versiones de los protocolos SSL y TLS soportadas, etc.

El estudio se llevará a cabo mediante la utilización de herramientas de escaneo y análisis propias, junto a la evaluación realizada mediante el servicio "SSL Server Test" [Ref - 4] de Qualys SSL Labs y, en concreto, empleando las APIs de automatización disponibles en dicho servicio.

Este servicio puede ser también utilizado manualmente por los responsables de los entornos web para autoevaluar el estado de los mismos, desde la dirección web "https://www.ssllabs.com/ssltest/". Se recomienda siempre seleccionar la opción "*Do not show the results on the boards*" antes de comenzar el análisis para que el entorno web no aparezca listado públicamente a través de dicho servicio.

En resumen, los Organismos interesados en participar en los estudios realizados por el CCN-CERT para evaluar la implementación de HTTPS y el estado actual del ecosistema HTTPS en el sector público deberán de identificar los entornos web objetivo del análisis facilitando el dominio principal (ej. cni.es) e identificando los servidores y aplicaciones web (mediante el siguiente formato):

- Nombre del servidor web (ej. www.ccn-cert.cni.es), sin incluir el protocolo (ej. http:// o https://) ni tampoco ningún recurso concreto (ej. www.ccn-cert.cni.es/cursos/).

Para recopilar la lista completa de entornos web objetivo que serán analizados durante el estudio se dispone de un documento¹⁴⁰ que deberá ser completado por el responsable del Organismo, proporcionando una pestaña diferente para cada dominio principal que desea incluir en el análisis, incluyendo en la misma todos los servidores web de dicho dominio que formarán parte del estudio.

Adicionalmente, aparte del análisis HTTPS de un servidor o sitio web específico, como por ejemplo su Sede Electrónica (https://sede.ministerio.es), se recomienda extender el uso de HTTPS a la totalidad del dominio, es decir, al servidor web por defecto ("www"), al propio dominio y a todos sus subdominios (ej. https://ministerio.es, https://www.ministerio.es y/o https://<subdominio>. ministerio.es).

El estudio de la implementación de HTTPS y del estado actual del ecosistema HTTPS en el sector público se centrará tanto en evaluar el estado global de la implementación HTTPS en el dominio principal (incluyendo el servidor web por defecto, "www", y sus subdominios), como en los entornos y servidores web concretos facilitados por sus responsables.

¹⁴⁰ Por ejemplo, hoja de cálculo Microsoft Excel que permita recopilar los datos del Organismo, su responsable y los servidores web objetivo: "CCN-CERT_EntornosWebObjetivo_HTTPS_2017_v1.0.xlsx".

ANEXO B. SUITES CRIPTOGRÁFICAS RECOMENDADAS

La siguiente lista proporciona un conjunto de *suites* criptográficas recomendadas inicialmente para la configuración de la implementación HTTPS, tanto para claves RSA como ECDSA, en formato OpenSSL, clasificadas en orden de preferencia según su nivel de seguridad y de interoperabilidad [Ref - 10]¹⁴¹:

```
ECDHE-ECDSA-CHACHA20-POLY1305
ECDHE-ECDSA-AES128-GCM-SHA256
ECDHE-ECDSA-AES256-GCM-SHA384
ECDHE-ECDSA-AES128-SHA256
ECDHE-ECDSA-AES256-SHA384
ECDHE-RSA-CHACHA20-POLY1305
ECDHE-RSA-AES128-GCM-SHA256
ECDHE-RSA-AES256-GCM-SHA384
ECDHE-RSA-AES128-SHA256
ECDHE-RSA-AES256-SHA384
DHE-RSA-AES128-GCM-SHA256
DHE-RSA-AES256-GCM-SHA384
DHE-RSA-AES128-SHA256
DHE-RSA-AES256-SHA256
```

NOTA: En la lista anterior, de manera conservadora, se da prioridad a AES128 y a CHACHA20 sobre AES256 asumiendo que no todos los clientes web disponen de soporte para AES-NI⁷⁴ (aunque la mayoría de cliente modernos sí disponen del mismo). En caso contrario, se podría dar prioridad a AES256 (en letra cursiva) sobre el resto de opciones.

NOTA: Asimismo, se da prioridad a las claves ECDSA por encima de las claves RSA en todos los casos, pero también se podría alternar la misma *suite* para ECDSA y, a continuación, para RSA (con prioridad sobre el resto de *suites* basadas en ECDSA).

Alternativamente, la herramienta "Mozilla SSL Configuration Generator"¹⁴² puede ser empleada para la generación de la configuración HTTPS inicial de diferentes servidores web, pudiendo especificar el nivel de seguridad deseado (o el nivel de compatibilidad requerido con navegadores y clientes web modernos, o más antiguos⁷¹) y otras características asociadas, incluyendo la versión del servidor web y de OpenSSL. Una vez generada la configuración, se recomienda personalizar y parametrizar la misma en función de las preferencias y requisitos del entorno web.

¹⁴¹ <https://github.com/ssllabs/research/wiki/SSL-and-TLS-Deployment-Best-Practices>

¹⁴² <https://mozilla.github.io/server-side-tls/ssl-config-generator/>

ANEXO C. REFERENCIAS

[Ref – 1]	"Informes de Buenas Prácticas (BP)". CCN-CERT. Informes 2017 https://www.ccn-cert.cni.es/informes/informes-ccn-cert-buenas-practicas-bp.html
[Ref – 2]	"HTTPS: TLS, no SSL". "Has Tenido Tiempo Para Securizarlo: Te Lo Suplico, no Sigas Solo Leyéndolo". Raúl Siles (DinoSec). X Jornadas STIC CCN-CERT. Presentación Diciembre 2016 https://www.ccn-cert.cni.es/documentos-publicos/x-jornadas-stic-ccn-cert/1942-p2-05-https-tls-ssl-x-jornadas-stic-ccn-cert-dinosec/file.html https://www.ccn-cert.cni.es/xjornadas
[Ref – 3]	"SSL Client Test". Qualys SSL Labs. Servicio web https://www.ssllabs.com/ssltest/viewMyClient.html
[Ref – 4]	"SSL Server Test". Qualys SSL Labs. Servicio web https://www.ssllabs.com/ssltest/
[Ref – 5]	"SSL Cipher Suite Details of Your Browser". DC Sec. Web https://cc.dcsec.uni-hannover.de
[Ref – 6]	"User Agent Capabilities". Qualys SSL Labs. Web https://www.ssllabs.com/ssltest/clients.html
[Ref – 7]	"Can I Use...?" Servicio web http://caniuse.com
[Ref – 8]	"Transport Layer Security (TLS) Extensions: Extension Definitions" (SNI). RFC 6066. IETF. Web January 2011 https://tools.ietf.org/html/rfc6066
[Ref – 9]	"TLS has exactly one performance problem: it is not used widely enough. Everything else can be optimized.". Ilya Grigorik. Web https://istlsfastyet.com https://hpbn.co (https://hpbn.co/transport-layer-security-tls/#optimizing-for-tls)
[Ref – 10]	"Bulletproof SSL and TLS". Ivan Ristic. Feisty Duck. Libro (book) https://www.feistyduck.com/books/bulletproof-ssl-and-tls/
[Ref – 11]	"Let's Encrypt". Web https://letsencrypt.org

[Ref – 12]	"Announcing the first SHA1 collision". Google Security Blog. Blog Post February 2017 https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html https://shattered.io
[Ref – 13]	"Certificate Transparency (CT)". Google. Web http://www.certificate-transparency.org
[Ref – 14]	"Certificate Transparency". RFC 6962. IETF. Web June 2013 https://tools.ietf.org/html/rfc6962
[Ref – 15]	"Transparencia en los Certificados". Google. Servicio web https://www.google.com/transparencyreport/https/ct/
[Ref – 16]	"Expect-CT Extension for HTTP". IETF. Web December 2016 (Draft 01: draft-stark-expect-ct-01) https://datatracker.ietf.org/doc/draft-stark-expect-ct https://tools.ietf.org/html/draft-stark-expect-ct-01
[Ref – 17]	"DNS Certification Authority Authorization (CAA) Resource Record". RFC 6844. IETF. Web January 2013 https://tools.ietf.org/html/rfc6844
[Ref – 18]	"CAA Mandated by CA/Browser Forum". Ivan Ristic. Qualys. Blog Post March 2017 https://blog.qualys.com/ssllabs/2017/03/13/caa-mandated-by-cabrowser-forum
[Ref – 19]	"Public Key Pinning Extension for HTTP". RFC 7469. IETF. Web April 2015 https://tools.ietf.org/html/rfc7469
[Ref – 20]	"BadSSL". Servicio web https://badssl.com https://github.com/chromium/badssl.com
[Ref – 21]	"SSL Server Rating Guide (SSL Server Test)". Qualys SSL Labs." Web 19 January 2017 (version 2009n) https://github.com/ssllabs/research/wiki/SSL-Server-Rating-Guide
[Ref – 22]	"The Transport Layer Security (TLS) Protocol Version 1.3". RFC nnnn. IETF. Web March 2017 (Draft 19: draft-ietf-tls-tls13-19) https://tools.ietf.org/html/draft-ietf-tls-tls13-19
[Ref – 23]	"Transport Layer Security (TLS) Parameters" (TLS Cipher Suite Registry). IANA.

	Web https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml
[Ref – 24]	"HTTPS as a ranking signal". Google. Web August 2014 https://webmasters.googleblog.com/2014/08/https-as-ranking-signal.html
[Ref – 25]	"Brotli Compression". Scott Helme. Web September 2016 https://scotthelme.co.uk/brotli-compression/
[Ref – 26]	"Resource Hints". W3C. Web March 2017 https://www.w3.org/TR/resource-hints/#preconnect
[Ref – 27]	"HTTP Strict Transport Security (HSTS)". RFC 6797. IETF. Web November 2012 https://tools.ietf.org/html/rfc6797
[Ref – 28]	"HSTS Preload List". Google. Servicio web https://hstspreload.org https://opensource.google.com/projects/hstspreload
[Ref – 29]	"HTTP Strict Transport Security: Full List". Google. Web https://www.chromium.org/hsts https://cs.chromium.org/chromium/src/net/http/transport_security_state_static.json
[Ref – 30]	"HSTS Preload Pending List". Google. Web https://hstspreload.org/api/v2/pending
[Ref – 31]	"Mixed Content (Block All Mixed Content)". W3C. Web August 2016 https://www.w3.org/TR/mixed-content/
[Ref – 32]	"Content Security Policy Reference". Web https://content-security-policy.com/
[Ref – 33]	"Content Security Policy 1.0". W3C. Web February 2015 https://www.w3.org/TR/CSP1/ https://www.w3.org/TR/2012/CR-CSP-20121115/ (November 2012)
[Ref – 34]	"Content Security Policy Level 2 (2.0)". W3C. Web December 2016 https://www.w3.org/TR/CSP2/

[Ref – 35]	"Content Security Policy Level 3 (3.0)". W3C. Web September 2016 (borrador) https://www.w3.org/TR/CSP3/ https://www.w3.org/TR/CSP/ https://w3c.github.io/webappsec-csp/ (documento de trabajo)
[Ref – 36]	"Upgrade Insecure Requests". W3C. Web October 2015 https://www.w3.org/TR/upgrade-insecure-requests/
[Ref – 37]	"Deprecating Powerful Features on Insecure Origins". The Chromium Projects. Web https://www.chromium.org/Home/chromium-security/deprecating-powerful-features-on-insecure-origins https://www.chromium.org/Home/chromium-security/prefer-secure-origins-for-powerful-new-features
[Ref – 38]	"Hypertext Transfer Protocol (HTTP/1.1): Caching". RFC 7234. IETF. Web June 2014 https://tools.ietf.org/html/rfc7234
[Ref – 39]	"SSL/TLS and PKI History". Ivan Ristic. Feisty Duck. Web https://www.feistyduck.com/ssl-tls-and-pki-history/
[Ref – 40]	"OpenSSL Cookbook". Ivan Ristic. Feisty Duck. Libro (book) https://www.feistyduck.com/books/openssl-cookbook/